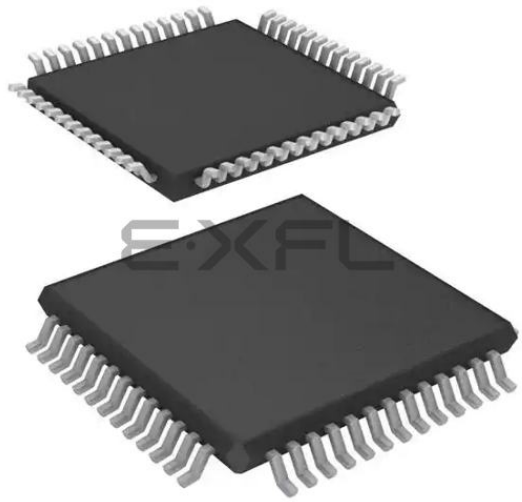


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	8032
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, IrDA, SPI, UART/USART, USB
Peripherals	LVD, POR, PWM, WDT
Number of I/O	35
Program Memory Size	80KB (80K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K 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/upsd3422ev-40t6

Table 2. Pin definitions (continued)

Port pin	Signal name	80-pin No.	52-pin No.(1)	In/out	Function		
					Basic	Alternate 1	Alternate 2
ALE		4	N/A	O	Address Latch signal, external bus		
RESET_IN		68	44	I	Active low reset input		
XTAL1		48	31	I	Oscillator input pin for system clock		
XTAL2		49	32	O	Oscillator output pin for system clock		
DEBUG		8	5	I/O	I/O to the MCU debug unit		
PA0		35	N/A	I/O	General I/O port pin		All Port A pins support: – PLD Macrocell 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		
PB0		80	52	I/O	General I/O port pin		All Port B pins support: – PLD Macrocell outputs, or – PLD inputs, or – Latched address out (A0-A7 or A8-A15)
PB1		78	51	I/O	General I/O port pin		
PB2		76	50	I/O	General I/O port pin		
PB3		74	49	I/O	General I/O port pin		
PB4		73	48	I/O	General I/O port pin		
PB5		71	46	I/O	General I/O port pin		
PB6		67	43	I/O	General I/O port pin		
PB7		66	42	I/O	General I/O port pin		
JTAGTMS	TMS	20	13	I	JTAG pin (TMS)		
JTAGTCK	TCK	17	12	I	JTAG pin (TCK)		
PC2		16	11	I/O	General I/O port pin		PLD Macrocell output, or PLD input
PC3	TSTAT	15	N/A	I/O	General I/O port pin	Optional JTAG Status (TSTAT)	PLD, Macrocell output, or PLD input
PC4	TERR	9	N/A	I/O	General I/O port pin	Optional JTAG Status ($\overline{\text{TERR}}$)	PLD, Macrocell output, or PLD input
JTAGTDI	TDI	7	4	I	JTAG pin (TDI)		
JTAGTDO	TDO	6	3	O	JTAG pin (TDO)		

7.4 Accumulator (ACC)

This is an 8-bit general purpose register which holds a source operand and receives the result of arithmetic operations. The ACC register can also be the source or destination of logic and data movement operations. For MUL and DIV instructions, ACC is combined with the B register to hold 16-bit operands. The ACC is referred to as “A” in the MCU instruction set.

7.5 B register (B)

The B register is a general purpose 8-bit register for temporary data storage and also used as a 16-bit register when concatenated with the ACC register for use with MUL and DIV instructions.

7.6 General purpose registers (R0 - R7)

There are four banks of eight general purpose 8-bit registers (R0 - R7), but only one bank of eight registers is active at any given time depending on the setting in the PSW word (described next). R0 - R7 are generally used to assist in manipulating values and moving data from one memory location to another. These register banks physically reside in the first 32 locations of 8032 internal DATA SRAM, starting at address 00h. At reset, only the first bank of eight registers is active (addresses 00h to 07h), and the stack begins at address 08h.

7.7 Program status word (PSW)

The PSW is an 8-bit register which stores several important bits, or flags, that are set and cleared by many 8032 instructions, reflecting the current state of the MCU core. *Figure 11 on page 40* shows the individual flags.

7.7.1 Carry flag (CY)

This flag is set when the last arithmetic operation that was executed results in a carry (addition) or borrow (subtraction). It is cleared by all other arithmetic operations. The CY flag is also affected by Shift and Rotate Instructions.

7.7.2 Auxiliary carry flag (AC)

This flag is set when the last arithmetic operation that was executed results in a carry into (addition) or borrow from (subtraction) the high-order nibble. It is cleared by all other arithmetic operations.

7.7.3 General purpose flag (F0)

This is a bit-addressable, general-purpose flag for use under software control.

7.7.4 Register bank select flags (RS1, RS0)

These bits select which bank of eight registers is used during R0 - R7 register accesses (see *Table 4*)

10 UPSD34xx instruction set summary

Tables *6* through *11* list all of the instructions supported by the UPSD34xx, including the number of bytes and number of machine cycles required to implement each instruction. This is the standard 8051 instruction set.

The meaning of “machine cycles” is how many 8032 MCU core machine cycles are required to execute the instruction. The “native” duration of all machine cycles is set by the memory wait state settings in the SFR, BUSCON, and the MCU clock divider selections in the SFR, CCON0 (i.e. a machine cycle is typically set to 4 MCU clocks for a 5 V UPSD34xx). However, an individual machine cycle may grow in duration when either of two things happen:

1. a stall is imposed while loading the 8032 Pre-Fetch Queue (PFQ); or
2. the occurrence of a cache miss in the Branch Cache (BC) during a branch in program execution flow.

See *Section 5: 8032 MCU core performance enhancements on page 33* or more details.

But generally speaking, during typical program execution, the PFQ is not empty and the BC has no misses, producing very good performance without extending the duration of any machine cycles.

Table 6. Arithmetic instruction set

Mnemonic[1] and use		Description	Length/cycles
ADD	A, Rn	Add register to ACC	1 byte/1 cycle
ADD	A, Direct	Add direct byte to ACC	2 byte/1 cycle
ADD	A, @Ri	Add indirect SRAM to ACC	1 byte/1 cycle
ADD	A, #data	Add immediate data to ACC	2 byte/1 cycle
ADDC	A, Rn	Add register to ACC with carry	1 byte/1 cycle
ADDC	A, direct	Add direct byte to ACC with carry	2 byte/1 cycle
ADDC	A, @Ri	Add indirect SRAM to ACC with carry	1 byte/1 cycle
ADDC	A, #data	Add immediate data to ACC with carry	2 byte/1 cycle
SUBB	A, Rn	Subtract register from ACC with borrow	1 byte/1 cycle
SUBB	A, direct	Subtract direct byte from ACC with borrow	2 byte/1 cycle
SUBB	A, @Ri	Subtract indirect SRAM from ACC with borrow	1 byte/1 cycle
SUBB	A, #data	Subtract immediate data from ACC with borrow	2 byte/1 cycle
INC	A	Increment A	1 byte/1 cycle
INC	Rn	Increment register	1 byte/1 cycle
INC	direct	Increment direct byte	2 byte/1 cycle
INC	@Ri	Increment indirect SRAM	1 byte/1 cycle
DEC	A	Decrement ACC	1 byte/1 cycle
DEC	Rn	Decrement register	1 byte/1 cycle
DEC	direct	Decrement direct byte	2 byte/1 cycle

Table 6. Arithmetic instruction set (continued)

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

1. All mnemonics copyrighted ©Intel Corporation 1980.

Table 7. Logical instruction set

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

1. All mnemonics copyrighted ©Intel Corporation 1980.

17 I/O ports of mcu module

The MCU module has three 8-bit I/O ports: Port 1, Port 3, and Port 4. The PSD module has four other I/O ports: Port A, B, C, and D. This section describes only the I/O ports on the MCU module.

I/O ports will function as bidirectional general-purpose I/O (GPIO), but the port pins can have alternate functions assigned at run-time by writing to specific SFRs. The default operating mode (during and after reset) for all three ports is GPIO input mode. Port pins that have no external connection will not float because each pin has an internal weak pull-up (~150 kΩ) to V_{CC}.

I/O ports 3 and 4 are 5 V tolerant, meaning they can be driven/pulled externally up to 5.5 V without damage. The pins on Port 4 have a higher current capability than the pins on Ports 1 and 3.

Three additional MCU ports (only on 80-pin UPSD34xx devices) are dedicated to bring out the 8032 MCU address, data, and control signals to external pins. One port, named MCUAD[7:0], has eight multiplexed address/data bidirectional signals. The third port has MCU bus control outputs: read, write, program fetch, and address latch. These ports are typically used to connect external parallel peripherals and memory devices, but they may NOT be used as GPIO. Notice that the eight upper address signals do not come out to pins on the port. If high-order address signals are required on external pins (MCU addresses A[15:8]), then these address signals can be brought out as needed to PLD output pins or to the Address Out mode pins on PSD module ports. See PSD module section, "*Section 28.5.40: Latched address output mode on page 238* for details.

Figure 15 on page 80 represents the flexibility of pin function routing controlled by the SFRs. Each of the 24 pins on three ports, P1, P3, and P4, may be individually routed on a pin-by-pin basis to a desired function.

17.1 MCU port operating modes

MCU port pins can operate as GPIO or as alternate functions (see *Figure 16 on page 80* through *Figure 18 on page 81*).

Depending on the selected pin function, a particular pin operating mode will automatically be used:

- GPIO - Quasi-bidirectional mode
- UART0, UART1 - Quasi-bidirectional mode
- SPI - Quasi-bidirectional mode
- I2C - Open drain mode
- ADC - Analog input mode
- PCA output - Push-Pull mode
- PCA input - Input only (Quasi-bidirectional)
- Timer 0,1,2 - Input only (Quasi-bidirectional)

Figure 28. Timer 2 in auto-reload mode

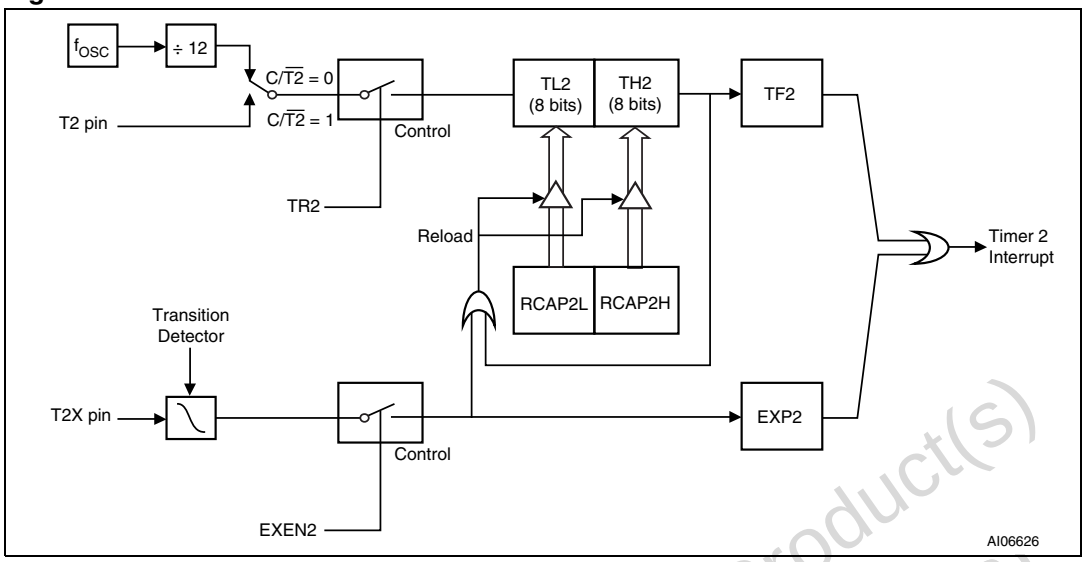


Figure 29. Timer 2 in baud rate generator mode

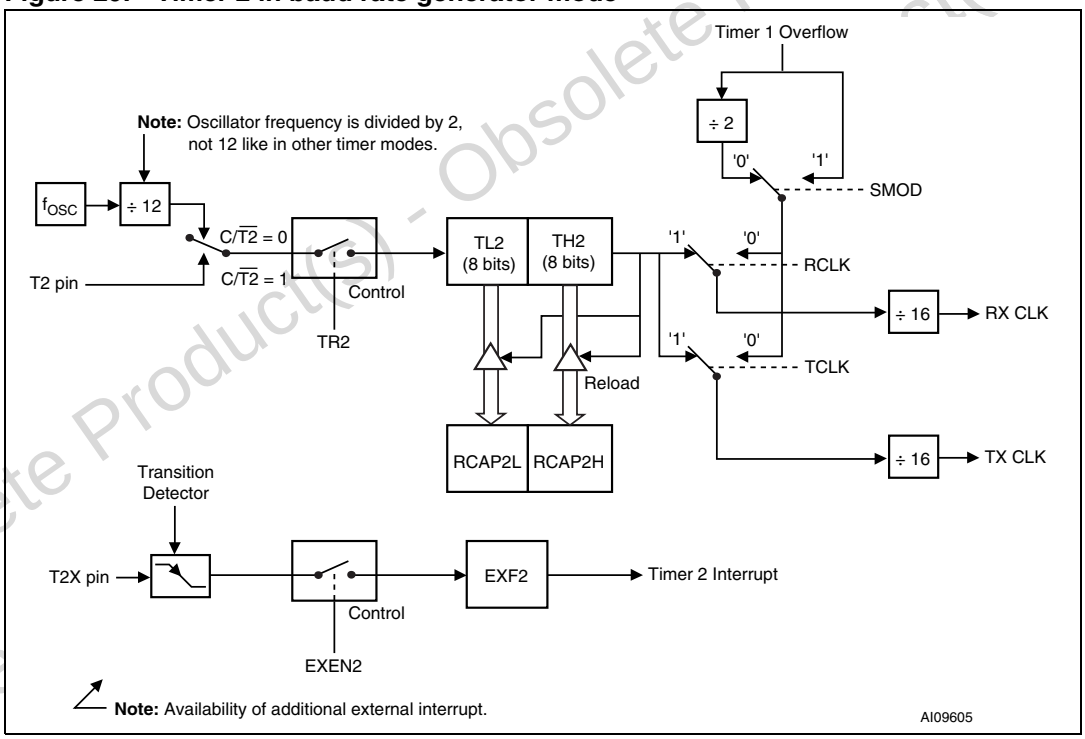


Figure 30. UART mode 0, block diagram

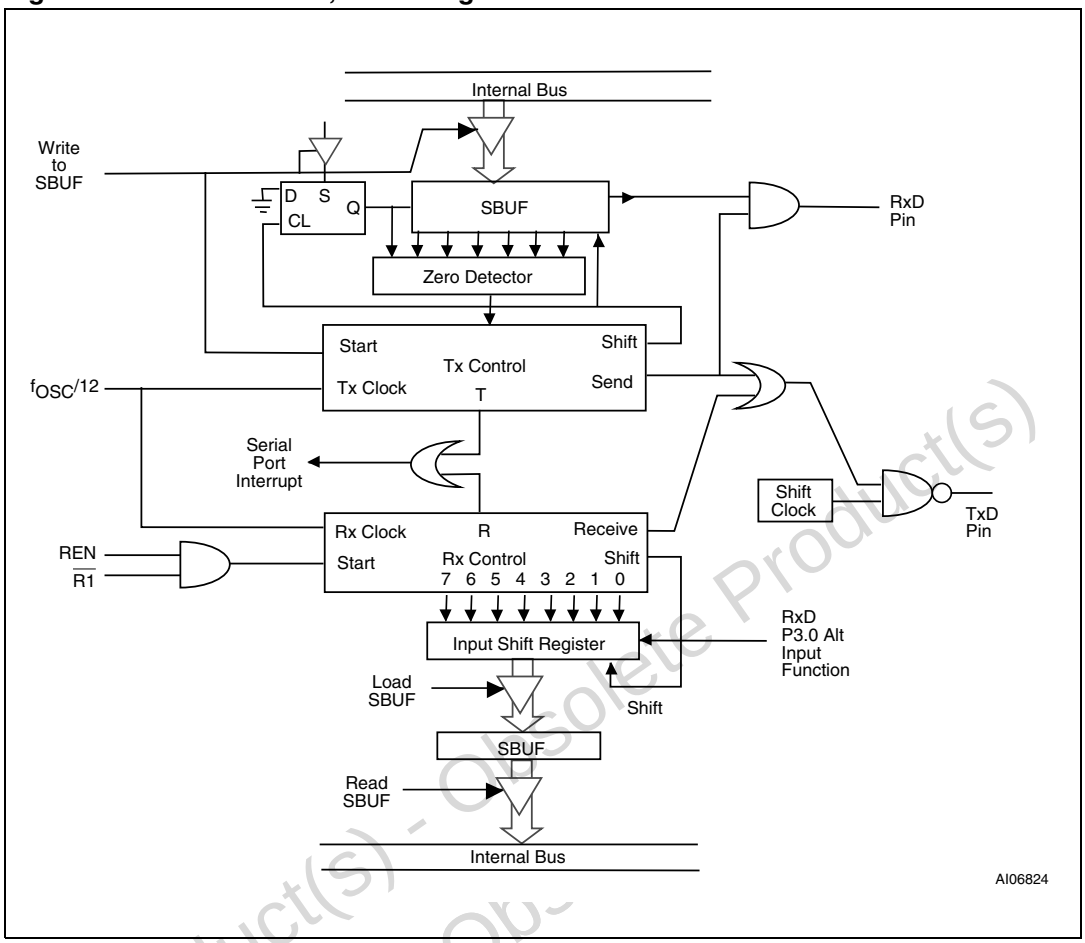


Figure 31. UART mode 0, timing diagram

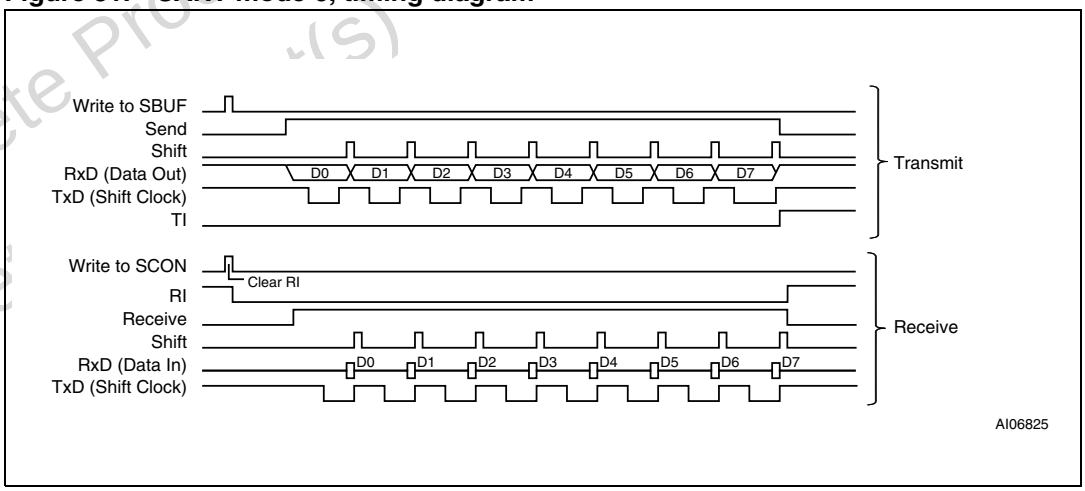
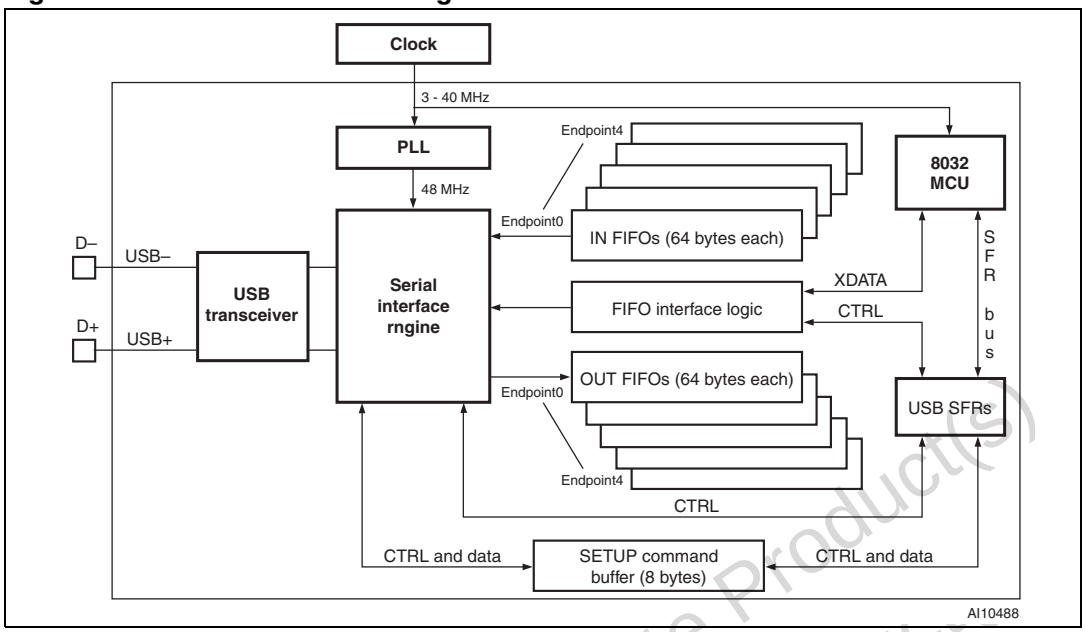


Figure 48. USB module block diagram



25.1 Basic USB concepts

The Universal Serial Bus (USB) is more complex than the standard serial port and requires familiarity with the specification to fully understand how to use the USB peripheral in the UPSD34xx. The USB specification is available on the Internet at **http://www.usb.org.** Some basic concepts will be presented in this section but knowledge of the USB specification is required.

In a USB system, there is only one master and the master is the host computer. The host controls all activity on the bus and devices respond to requests from the host. The only exception is when a device has been put into a low power suspend mode by the host. In this case, the device can signal a remote wakeup. Outside of that exception, all activity is controlled and initiated by the host. The host-centric model versus a peer-to-peer model provides the best way to develop low cost peripherals by keeping the complex control logic on the host side. The UPSD34xx is a peripheral (non-host) device.

25.1.1 Communication flow

The USB provides a means for communication between host (client) software and a function on a USB device. Functions can have different requirements for the communication flow depending on the client software to the USB function interaction. With USB, the various communication flows are separated to provide better bus utilization. For example, one communication flow is used for managing the device while another is for transferring data related to the operation of the device. Some bus access is used for each communication flow with each flow terminated at an endpoint on a device. Each endpoint has various aspects associated with the communication flow. A USB device looks like a collection of endpoints to the USB system.

- USB endpoint control register (UCON)

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

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

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

Table 127. UCON register bit definition

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

27 Programmable counter array (PCA) with PWM

There are two Programmable Counter Array blocks (PCA0 and PCA1) in the UPSD34xx. A PCA block consists of a 16-bit up-counter, which is shared by three TCM (Timer Counter Module). A TCM can be programmed to perform one of the following four functions:

1. Capture Mode: capture counter values by external input signals
2. Timer Mode
3. Toggle Output Mode
4. PWM Mode: fixed frequency (8-bit or 16-bit), programmable frequency (8-bit only)

27.1 PCA block

The 16-bit Up-Counter in the PCA block is a free-running counter (except in PWM Mode with programmable frequency). The Counter has a choice of clock input: from an external pin, Timer 0 Overflow, or PCA Clock.

A PCA block has 3 Timer Counter Modules (TCM) which share the 16-bit Counter output. The TCM can be configured to capture or compare counter value, generate a toggling output, or PWM functions. Except for the PWM function, the other TCM functions can generate an interrupt when an event occurs.

Every TCM is connected to a port pin in Port 4; the TCM pin can be configured as an event input, a PWMs, a Toggle Output, or as External Clock Input. The pins are general I/O pins when not assigned to the TCM.

The TCM operation is configured by Control registers and Capture/Compare registers. *Table 143 on page 182* lists the SFR registers in the PCA blocks.

Figure 57. PCA0 block diagram

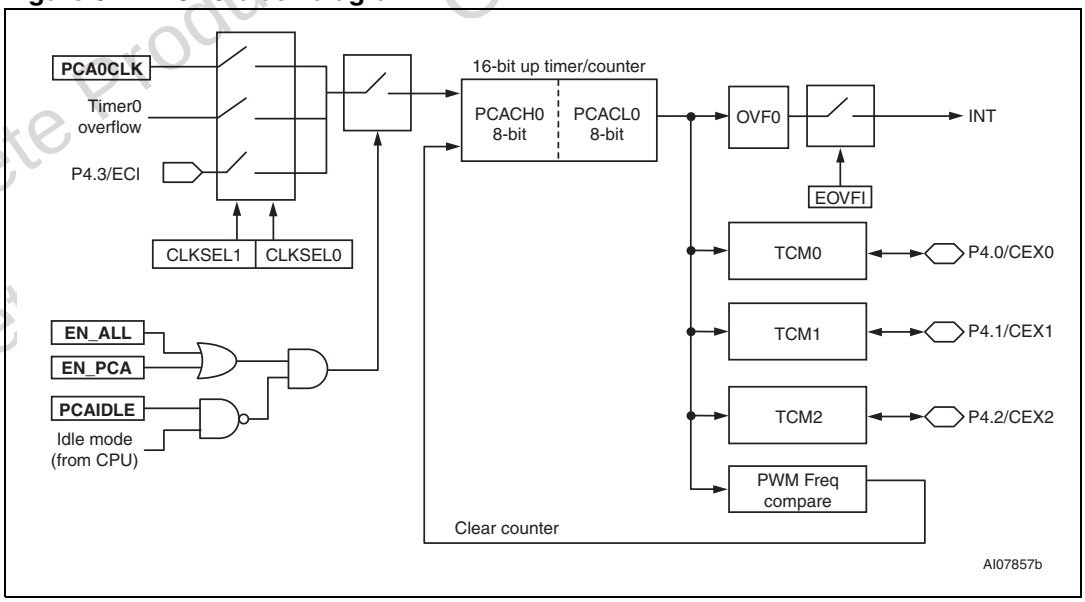
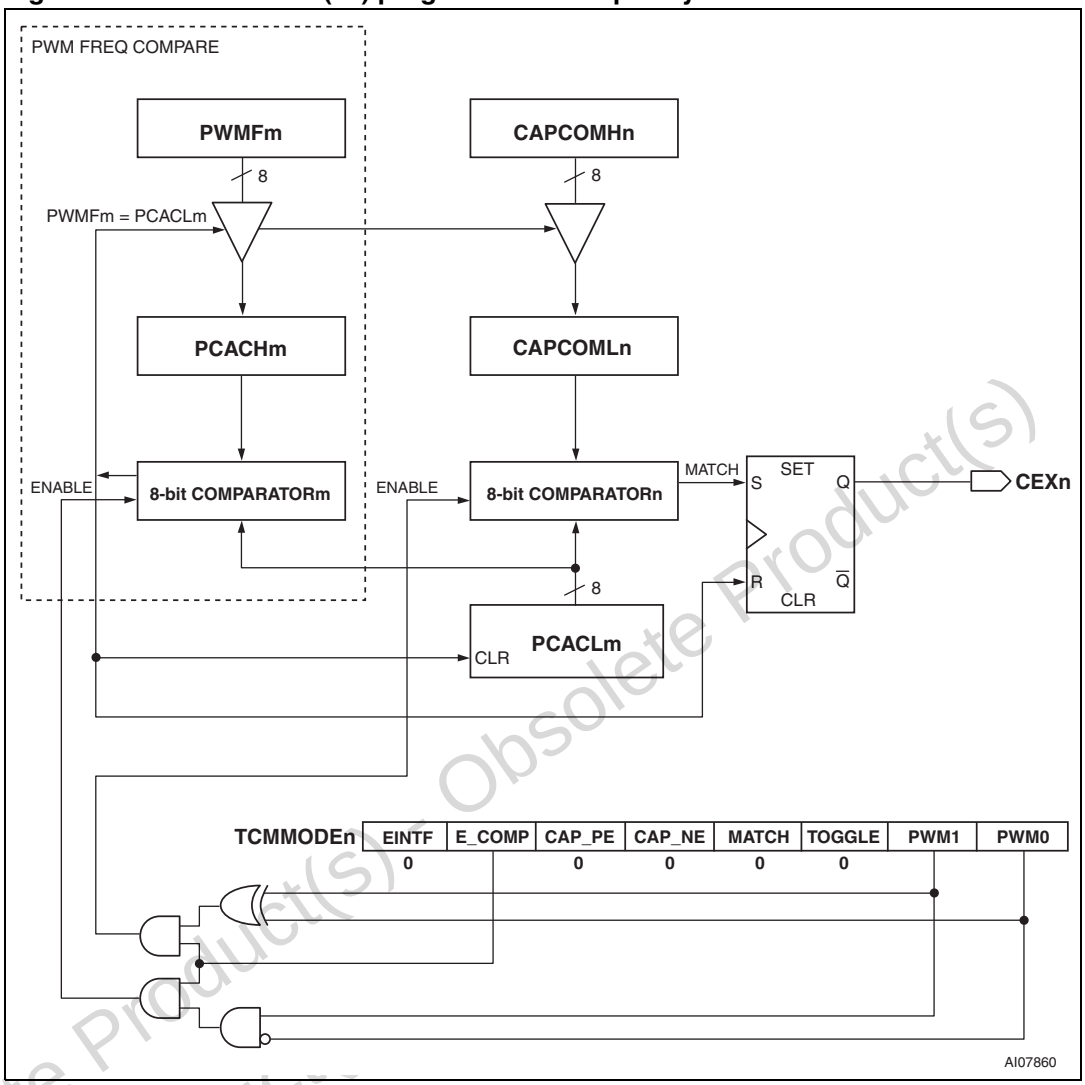


Figure 60. PWM mode - (x8) programmable frequency



1. m = 0: n = 0, 1, or 2
   m = 1: n = 3, 4, or 5

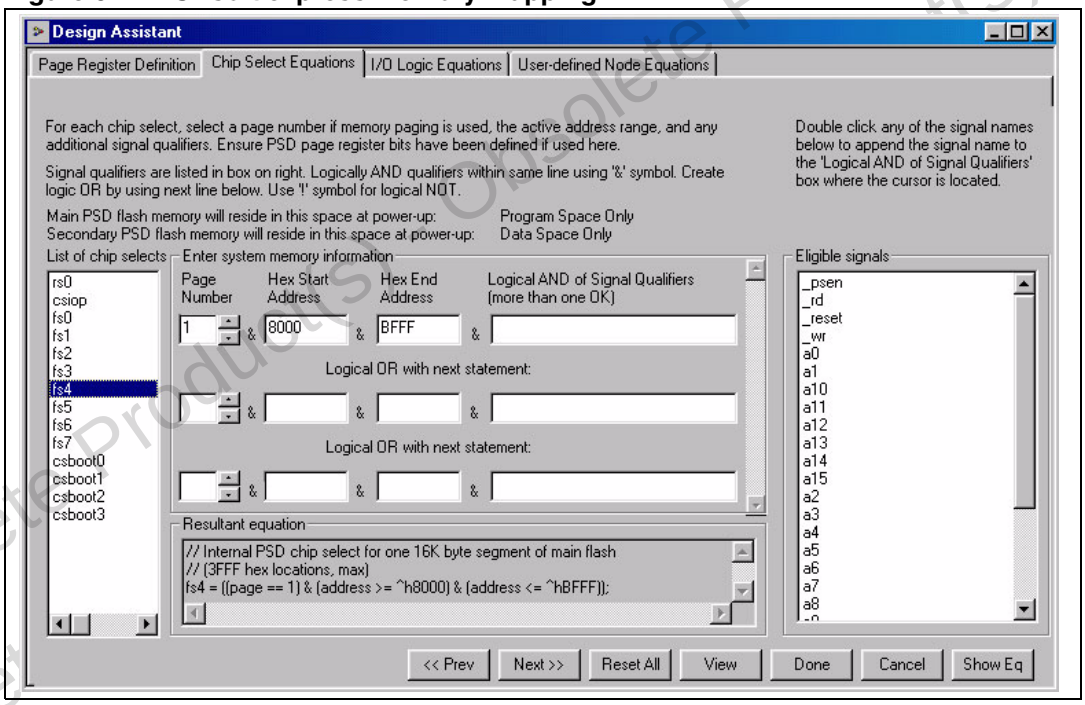
27.9 PWM mode - fixed frequency, 16-bit

The operation of the 16-bit PWM is the same as the 8-bit PWM with fixed frequency. In this mode, one or all the TCM can be configured to have a fixed frequency PWM output on the port pins. The PWM frequency is depending on the clock input frequency to the 16-bit Counter. The duty cycle of each TCM module can be specified in the CAPCOMHn and CAPCOMLn registers. When the 16-bit PCA_Counter is equal or greater than the values in registers CAPCOMHn and CAPCOMLn, the PWM output is switched to a high state. When the PCA_Counter overflows, CEXn is asserted low.

Table 159. HDL statement example generated from PSDsoft express 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 64. PSDsoft express memory mapping



28.2.4 EEPROM emulation

EEPROM emulation is needed if it is desired to repeatedly change only a small number of bytes of data in Flash memory. In this case EEPROM emulation is needed because although Flash memory can be written byte-by-byte, it must be erased sector-by-sector, it is not erasable byte-by-byte (unlike EEPROM which is written AND erased byte-by-byte). So changing one or two bytes in Flash memory typically requires erasing an entire sector each time only one byte is changed within that sector.

However, two of the 8 Kbyte sectors of Secondary Flash memory may be used to emulate EEPROM by using a linked-list software technique to create a small data set that is

maintained by alternating between the two flash sectors. For example, a data set of 128 bytes is written and maintained by software in a distributed fashion across one 8 Kbyte sector of Secondary Flash memory until it becomes full. Then the writing continues on the other 8 Kbyte sector while erasing the first 8 Kbyte sector. This process repeats continuously, bouncing back and forth between the two 8 Kbyte sectors. This creates a wear-leveling effect, which increases the effective number of erase cycles for a data set of 128 bytes to many times more than the base 100 000 erase cycles of the Flash memory. EEPROM emulation in Flash memory is typically faster than writing to actual EEPROM memory, and more reliable because the last known value in a data set is maintained even if a WRITE cycle is corrupted by a power outage. The EEPROM emulation function can be called by the user's firmware, making it appear that the user is writing a single byte, or data record, thus hiding all of the data management that occurs within the two 8 Kbyte Flash sectors. EEPROM emulation firmware for the UPSD34xx is available from www.st.com/psm.

28.2.5 Alternative mapping schemes

Here are more possible memory maps for the UPSD3433.

Note: Mapping examples would be slightly different for UPSD3433 and UPSD3434, because of the different sizes of individual Flash memory sectors.

- ***Figure 65*** Place the larger Main Flash Memory into program space, but split the Secondary Flash in half, placing two of its sectors into XDATA space and remaining two sectors into program space. This method allows the designer to put IAP code (or boot code) into two sectors of Secondary Flash in program space, and use the other two Secondary Flash sectors for data storage, such as EEPROM emulation in XDATA space.
- ***Figure 66*** Place both the Main and Secondary Flash memories into program space for maximum code storage, with no Flash memory in XDATA space.

Figure 65. Mapping: split second Flash in half

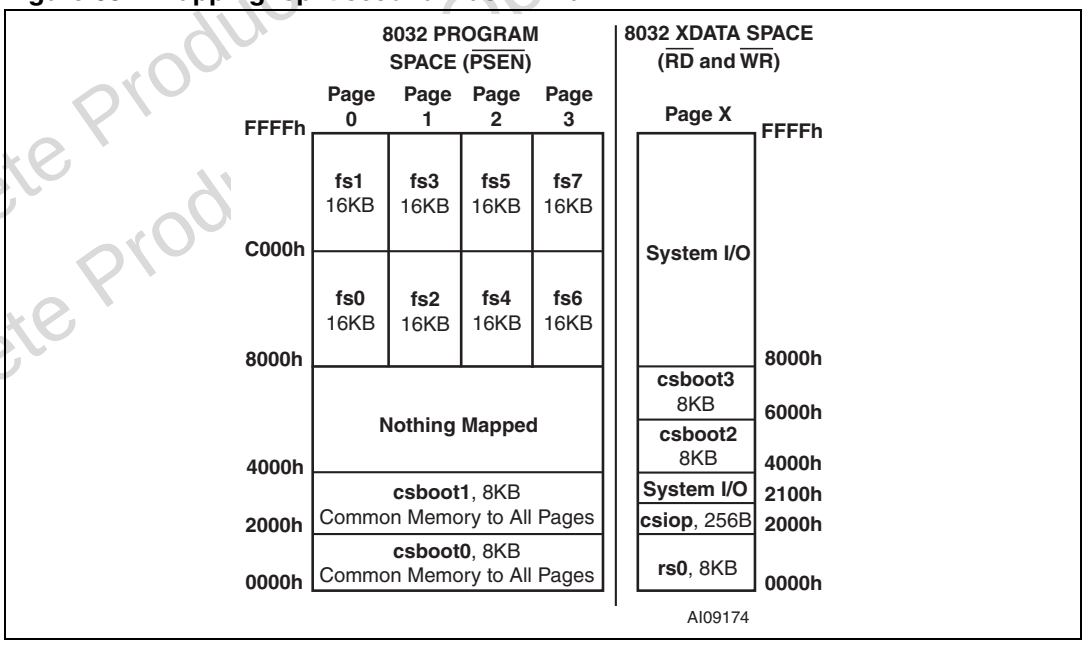


Figure 73. DPLD and GPLD

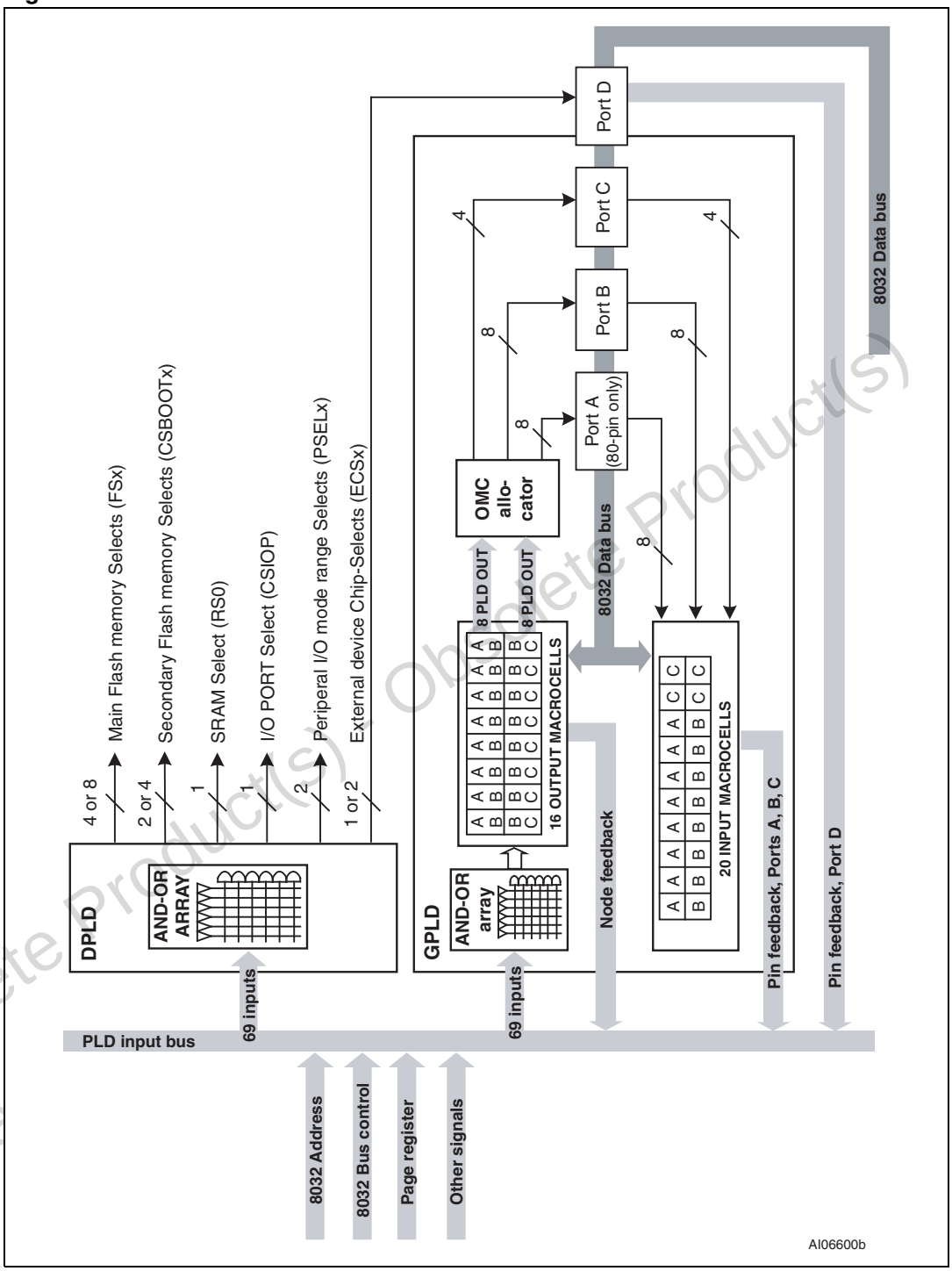


Figure 75. GPLD: one OMC, one IMC, and one I/O port (typical pin, port A, B, or C)

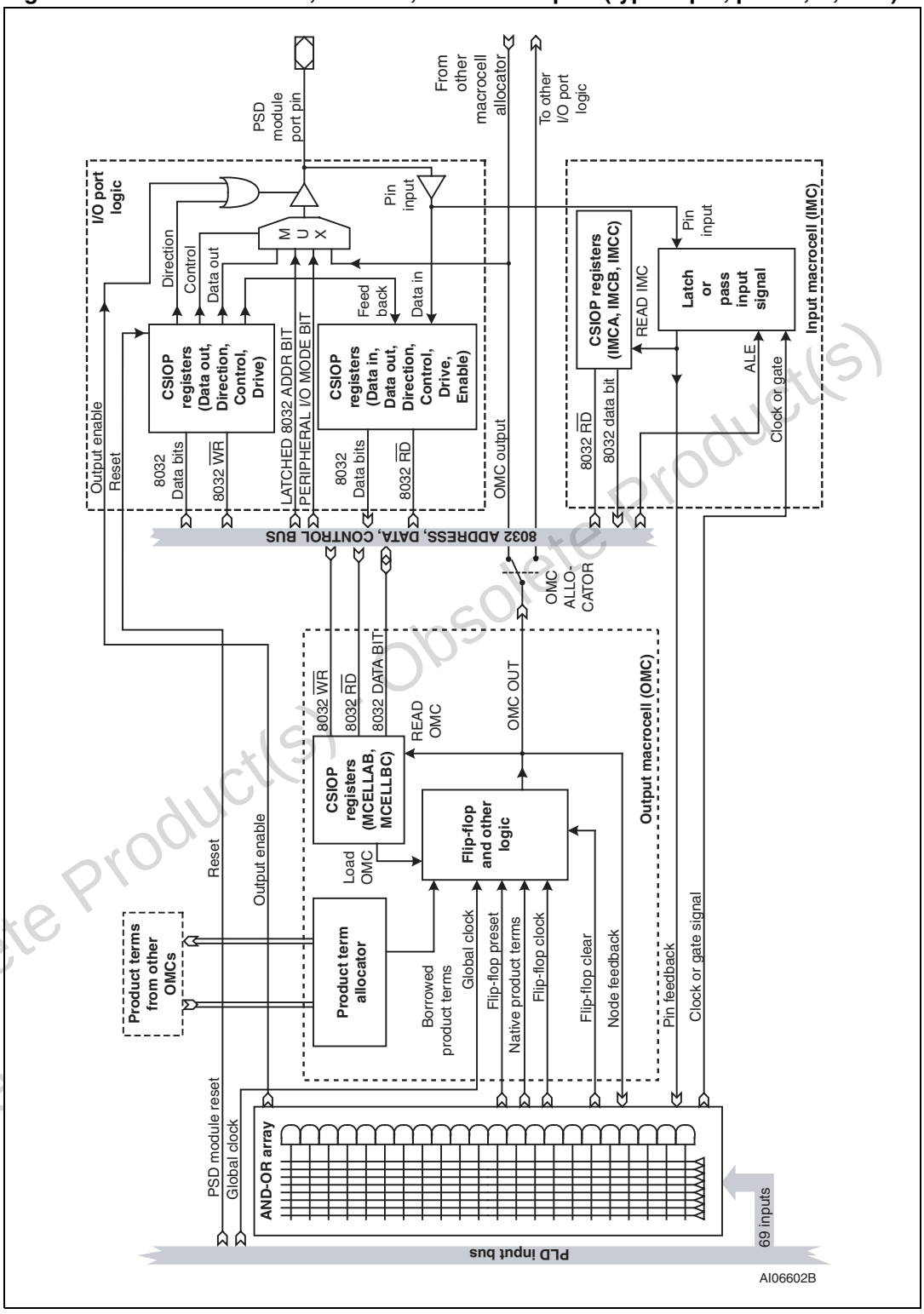
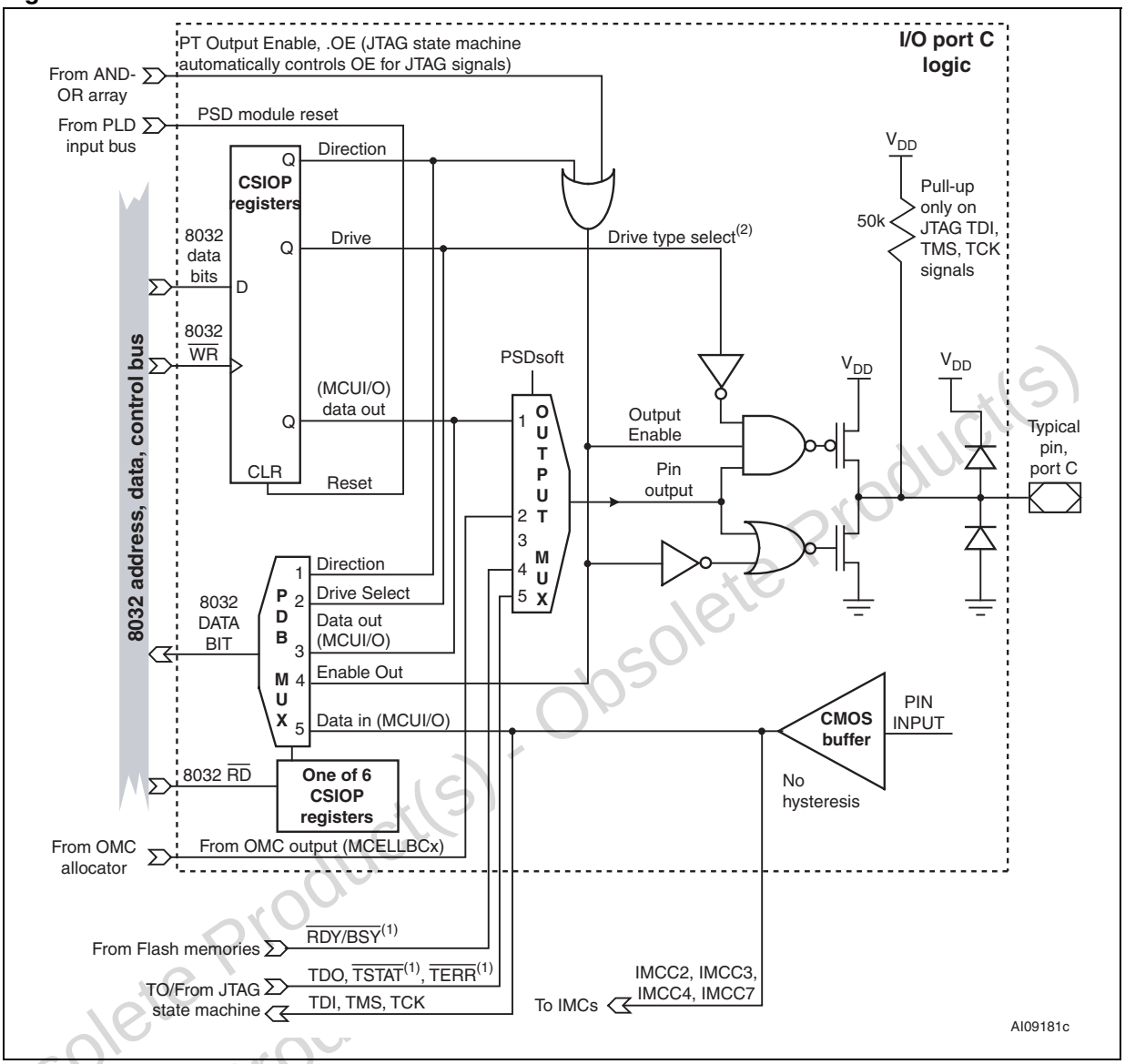


Figure 86. Port C structure



Note: 1 Optional function on a specific Port C pin.

28.5.51 Port D structure

Port D has two I/O pins (PD1, PD2) on 80-pin UPSD34xx devices, and just one pin (PD1) on 52-pin devices, supporting the following operating modes:

- MCU I/O Mode
- DPLD Output Mode for External Chip Selects, ECS1, ECS2. This does not consume OMCs in the GPLD.
- PLD Input Mode – direct input to the PLD Input Bus available to DPLD and GPLD. Does not use IMCs

See *Figure 87 on page 247* for detail.

28.6.9 JTAG security setting

A programmable security bit in the PSD module protects its contents from unauthorized viewing and copying. The security bit is set by clicking on the “Additional PSD Settings” box in the main flow diagram of PSDsoft Express, then choosing to set the security bit. Once a file with this setting is programmed into a UPSD34xx using JTAG ISP, any further attempts to communicate with the UPSD34xx using JTAG will be limited. Once secured, the only JTAG operation allowed is a full-chip erase. No reading or modifying Flash memory or PLD logic is allowed. Debugging operations to the MCU module are also not allowed. The only way to defeat the security bit is to perform a JTAG ISP full-chip erase operation, after which the device is blank and may be used again. The 8032 on the MCU module will always have access to PSM Module memory contents through the 8-bit 8032 data bus connecting the two die, even while the security bit is set.

28.6.10 Initial delivery state

When delivered from STMicroelectronics, UPSD34xx devices are erased, meaning all Flash memory and PLD configuration bits are logic '1.' Firmware and PLD logic configuration must be programmed at least the first time using JTAG ISP. Subsequent programming of Flash memory may be performed using JTAG ISP, JTAG debugging, or the 8032 may run firmware to program Flash memory (IAP).

Table 210. Major parameters

Parameter	Test conditions/comments	5.0 V value	3.3 V value	Unit
Operating voltage	–	4.5 to 5.5 (PSD); 3.0 to 3.6 (MCU)	3.0 to 3.6 (PSD and MCU)	V
Operating temperature	–	–40 to 85	–40 to 85	°C
MCU frequency	8 MHz (min) for I^2C	3 Min, 40 Max	3 Min, 40 Max	MHz
Operating current, typical(1) (20% of PLD used; 25°C operation. Bus control signals are blocked from the PLD in Non-turbo mode.)	40 MHz crystal, Turbo	79	63	mA
	40 MHz crystal, non-Turbo	71	58	mA
	8 MHz crystal, Turbo	32	24	mA
	8 MHz crystal, non-Turbo	17.7	14	mA
Idle current, typical (20% of PLD used; 25°C operation)	40 MHz crystal divided by 2048 internally. All interfaces are disabled.	19	18	mA
Standby current, typical	Power-down mode needs reset to exit.	140	120	µA
I/O sink/source current, ports A, B, C, and D	V_{OL} = 0.45 V (max); V_{OH} = 2.4 V (min)	I_{OL} = 8 (max); I_{OH} = –2 (min)	I_{OL} = 4 (max); I_{OH} = –1 (min)	mA
I/O sink/source current, port 4	V_{OL} = 0.6 V (max); V_{OH} = 2.4 V (min)	I_{OL} = 10 (max); I_{OH} = –10 (min)	I_{OL} = 10 (max); I_{OH} = –10 (min)	mA
PLD macrocells	For registered or combinatorial logic	16	16	–
PLD inputs	Inputs from pins, feedback, or MCU addresses	69	69	–
PLD outputs	Output to pins or internal feedback	18	18	–
PLD propagation delay, typical, Turbo mode	PLD input to output	15	22	ns

1. Operating current is measured while the UPSD34xx is executing a typical program at 40 MHz.

Table 211. MCU module DC characteristics

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
V_{CC}, AV_{CC}	Supply voltage(1)		3.0		3.6	V
V_{IH}	High level input voltage (ports 0, 1, 3, 4, XTAL1, RESET) 5 V tolerant - max voltage 5.5 V	3.0 V < V_{CC} < 3.6 V	0.7V_{CC}		5.5(2)	V
V_{IL}	Low level input voltage (ports 0, 1, 3, 4, XTAL1, RESET)	3.0 V < V_{CC} < 3.6 V	V_{SS} – 0.5		0.3V_{CC}	V
V_{OL1}	Output low voltage (port 4)	I_{OL} = 10 mA			0.6	V
						V

Table 230. Port A peripheral data mode READ timing (3 V PSD module)

Symbol	Parameter	Conditions	Min	Max	Turbo off	Unit
$t_{AVQV-PA}$	Address valid to data valid	(1)		50	+ 15	ns
$t_{SLQV-PA}$	$\overline{CSI}$ valid to data valid			37	+ 15	ns
$t_{RLQV-PA}$	$\overline{RD}$ to data valid	(2)		45		ns
$t_{DVQV-PA}$	Data in to data out valid			38		ns
$t_{RHQZ-PA}$	$\overline{RD}$ to data High-Z			36		ns

1. Any input used to select Port A data peripheral mode.
2. Data is already stable on Port A.

Figure 106. Peripheral I/O WRITE timing

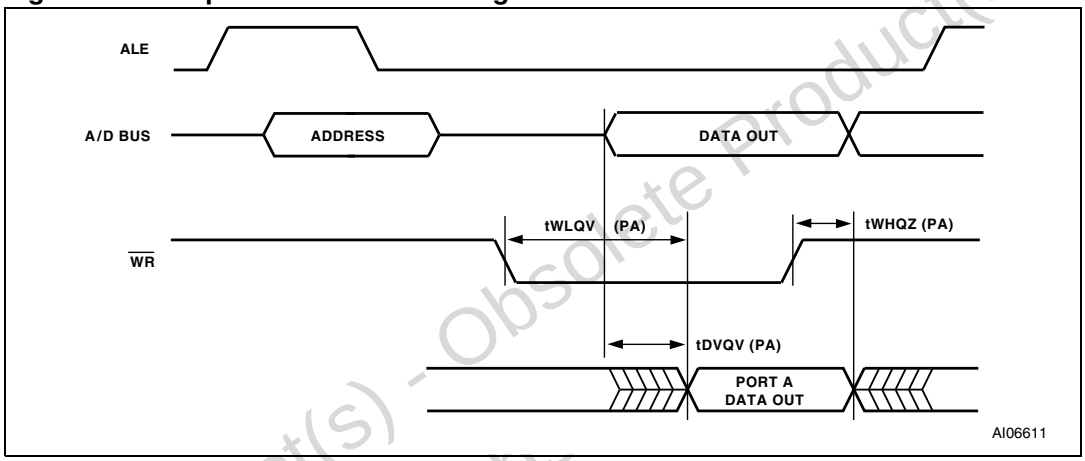


Table 231. Port A peripheral data mode WRITE timing (5 V PSD module)

Symbol	Parameter	Conditions	Min	Max	Unit
$t_{WLQV-PA}$	$\overline{WR}$ to data propagation delay			25	ns
$t_{DVQV-PA}$	Data to port A data propagation delay	(1)		22	ns
$t_{WHQZ-PA}$	$\overline{WR}$ Invalid to port a tri-state			20	ns

1. Data stable on Port 0 pins to data on Port A.

Table 232. Port A peripheral data mode WRITE timing (3 V PSD module)

Symbol	Parameter	Conditions	Min	Max	Unit
$t_{WLQV-PA}$	$\overline{WR}$ to data propagation delay			42	ns
$t_{DVQV-PA}$	Data to port A data propagation delay	(1)		38	ns
$t_{WHQZ-PA}$	$\overline{WR}$ Invalid to port a tri-state			33	ns

1. Data stable on Port 0 pins to data on Port A.