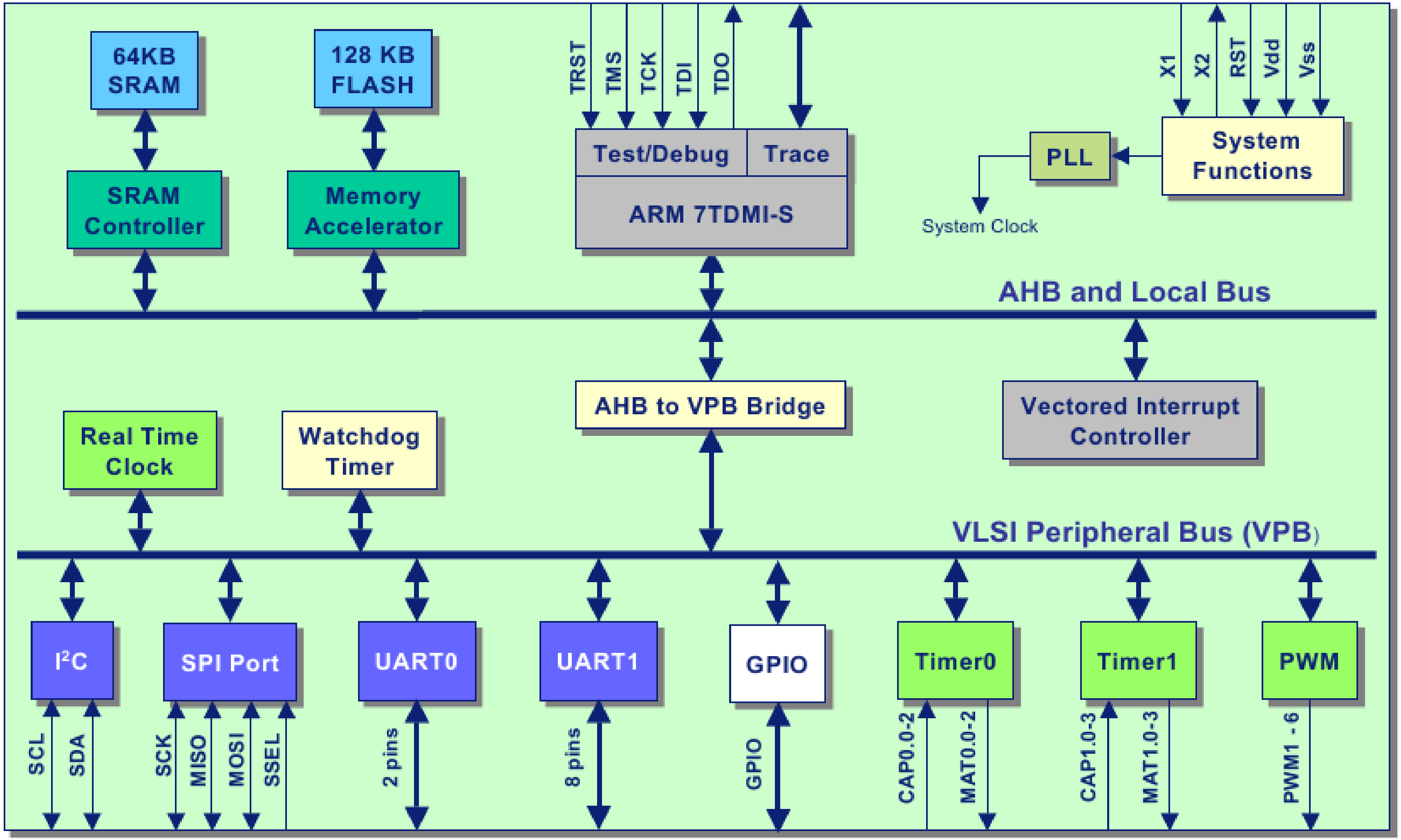
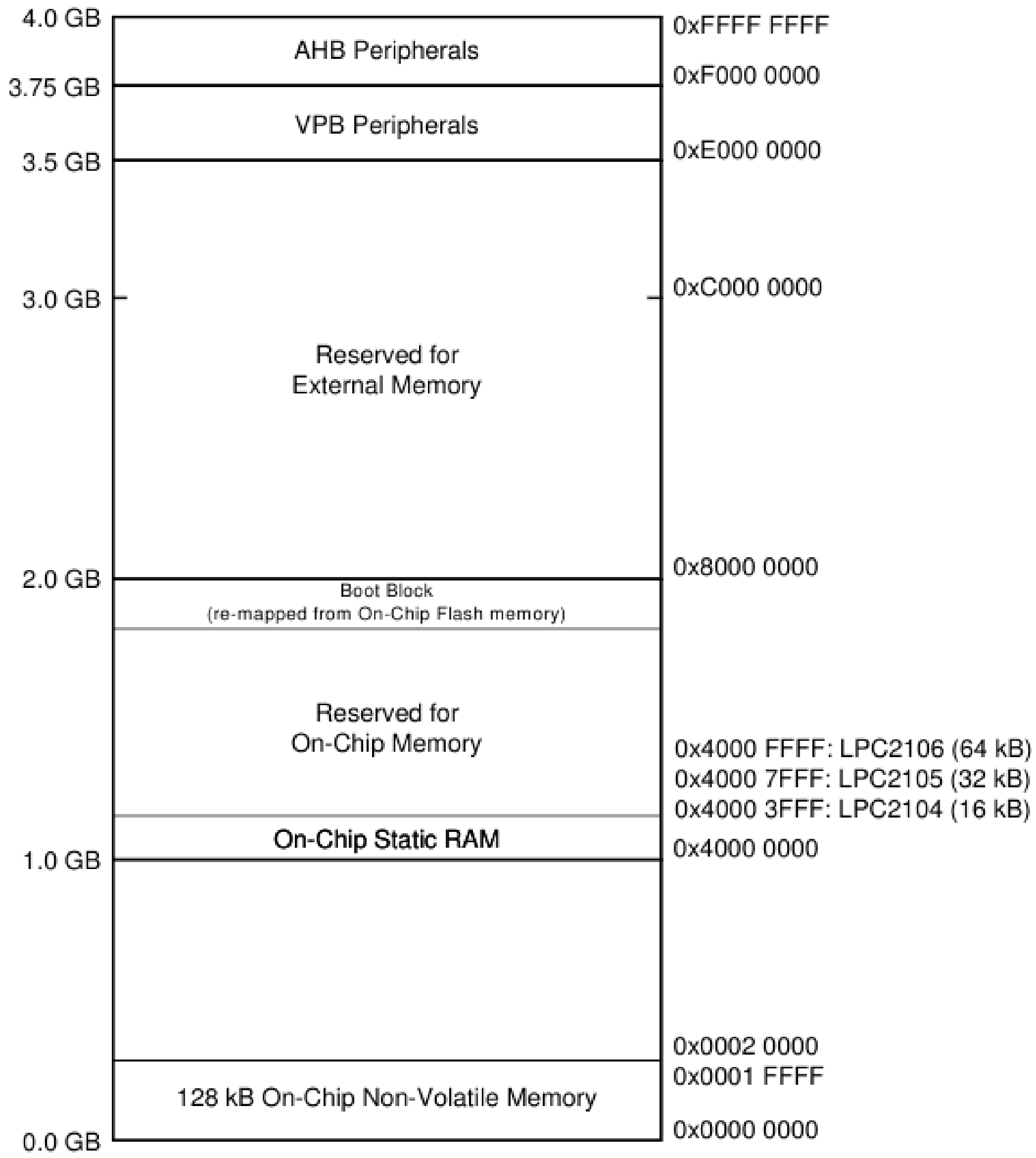
ARM7TDMI

Microcontroller LPC2106

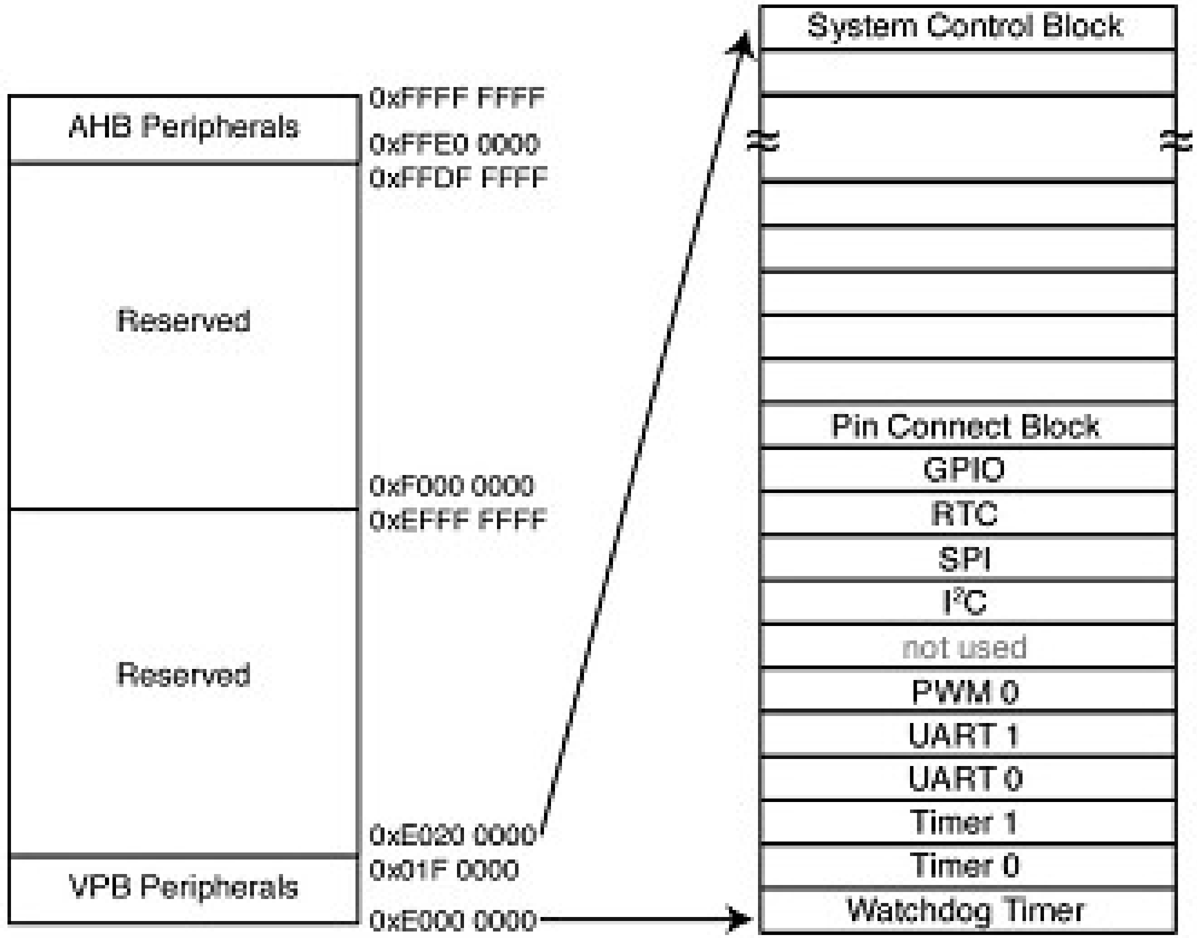
Block diagram



Memory Map


4.0 GB
3.75 GB
3.5 GB
3.0 GB
2.0 GB
1.0 GB
0.0 GB
AHB Peripherals
VPB Peripherals
Reserved for
External Memory
Boot Block
(re-mapped from On-Chip Flash memory)
Reserved for
On-Chip Memory
On-Chip Static RAM
128 kB On-Chip Non-Volatile Memory
0xFFFF FFFF
0xF000 0000
0xE000 0000
0xC000 0000
0x8000 0000
0x4000 FFFF: LPC2106 (64 kB)
0x4000 7FFF: LPC2105 (32 kB)
0x4000 3FFF: LPC2104 (16 kB)
0x4000 0000
0x0002 0000
0x0001 FFFF
0x0000 0000

VLSI Peripheral Bus - VPB



Pin Connect Block

Address	Name	Description	Access
0xE002C000	PINSEL0	Pin function select register 0	Read/Write
0xE002C004	PINSEL1	Pin function select register 1	Read/Write

Pinsel0 and Pinsel1 Values		Function	Value after Reset
0	0	Primary (default) function, typically GPIO Port	00
0	1	First alternate function	
1	0	Second alternate function	
1	1	Reserved	

Pin Configuration

PINSEL0	Pin Name	Function when 00	Function when 01	Function when 10	Function when 11	Reset Value
1:0	P0.0	GPIO Port 0.0	TxD (UART 0)	PWM1	Reserved	0
3:2	P0.1	GPIO Port 0.1	RxD (UART 0)	PWM3	Reserved	0
5:4	P0.2	GPIO Port 0.2	SCL (I^2C)	Capture 0.0 (Timer 0)	Reserved	0
7:6	P0.3	GPIO Port 0.3	SDA (I^2C)	Match 0.0 (Timer 0)	Reserved	0
9:8	P0.4	GPIO Port 0.4	SCK (SPI)	Capture 0.1 (Timer 0)	Reserved	0
11:10	P0.5	GPIO Port 0.5	MISO (SPI)	Match 0.1 (Timer 0)	Reserved	0
13:12	P0.6	GPIO Port 0.6	MOSI (SPI)	Capture 0.2 (Timer 0)	Reserved	0
15:14	P0.7	GPIO Port 0.7	SSEL (SPI)	PWM2	Reserved	0
17:16	P0.8	GPIO Port 0.8	TxD UART 1	PWM4	Reserved	0
19:18	P0.9	GPIO Port 0.9	RxD (UART 1)	PWM6	Reserved	0
21:20	P0.10	GPIO Port 0.10	RTS (UART1)	Capture 1.0 (Timer 1)	Reserved	0
23:22	P0.11	GPIO Port 0.11	CTS (UART1)	Capture 1.1 (Timer 1)	Reserved	0
25:24	P0.12	GPIO Port 0.12	DSR (UART1)	Match 1.0 (Timer 1)	Reserved	0
27:26	P0.13	GPIO Port 0.13	DTR (UART 1)	Match 1.1 (Timer 1)	Reserved	0
29:28	P0.14	GPIO Port 0.14	CD (UART 1)	EINT1	Reserved	0
31:30	P0.15	GPIO Port 0.15	RI (UART1)	EINT2	Reserved	0

Pin Configuration

PINSEL1	Pin Name	Function when 00	Function when 01	Function when 10	Function when 11	Reset Value
1:0	P0.16	GPIO Port 0.16	EINT0	Match 0.2 (Timer 0)	Reserved	0
3:2	P0.17	GPIO Port 0.17	Capture 1.2 (Timer 1)	Reserved	Reserved	0
5:4	P0.18	GPIO Port 0.18	Capture 1.3 (Timer 1)	Reserved	Reserved	0
7:6	P0.19	GPIO Port 0.19	Match 1.2 (Timer 1)	Reserved	Reserved	0
9:8	P0.20	GPIO Port 0.20	Match 1.3 (Timer 1)	Reserved	Reserved	0
11:10	P0.21	GPIO Port 0.21	PWM5	Reserved	Reserved	0
13:12	P0.22	GPIO Port 0.22	Reserved	Reserved	Reserved	0
15:14	P0.23	GPIO Port 0.23	Reserved	Reserved	Reserved	0
17:16	P0.24	GPIO Port 0.24	Reserved	Reserved	Reserved	0
19:18	P0.25	GPIO Port 0.25	Reserved	Reserved	Reserved	0
21:20	P0.26	GPIO Port 0.26	Reserved	Reserved	Reserved	0
23:22	P0.27	GPIO Port 0.27	TRST	Reserved	Reserved	0
25:24	P0.28	GPIO Port 0.28	TMS	Reserved	Reserved	0
27:26	P0.29	GPIO Port 0.29	TCK	Reserved	Reserved	0
29:28	P0.30	GPIO Port 0.30	TDI	Reserved	Reserved	0
31:30	P0.31	GPIO Port 0.31	TDO	Reserved	Reserved	0

General Purpose I/O - GPIO

Address	Name	Description	Access
0xE0028000	IOPIN	GPIO Pin value register. The current state of the port pins can always be read from this register, regardless of pin direction and mode.	Read Only
0xE0028004	IOSET	GPIO 0 Output set register. This register controls the state of output pins in conjunction with the IOCLR register. Writing ones produces highs at the corresponding port pins. Writing zeroes has no effect.	Read/Set
0xE0028008	IODIR	GPIO 0 Direction control register. This register individually controls the direction of each port pin.	Read/Write
0xE002800C	IOCLR	GPIO 0 Output clear register. This register controls the state of output pins. Writing ones produces lows at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing zeroes has no effect.	Clear Only

General Propose I/O - GPIO

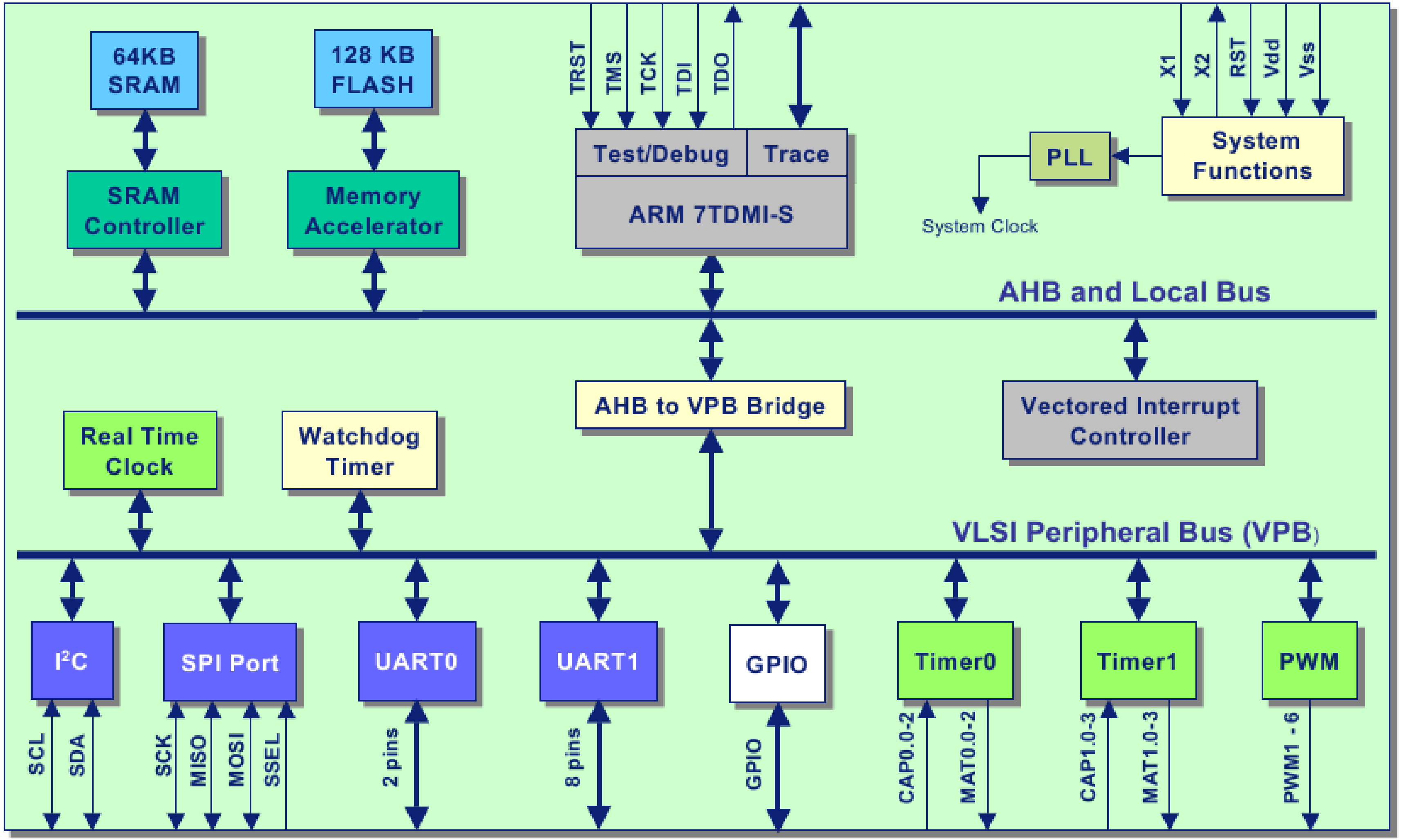
IOPIN	Description	Value after Reset
31:0	GPIO pin value bits. Bit 0 corresponds to P0.0 ... Bit 31 corresponds to P0.31	Undefined

IOSET	Description	Value after Reset
31:0	Output value SET bits. Bit 0 corresponds to P0.0 ... Bit 31 corresponds to P0.31	0

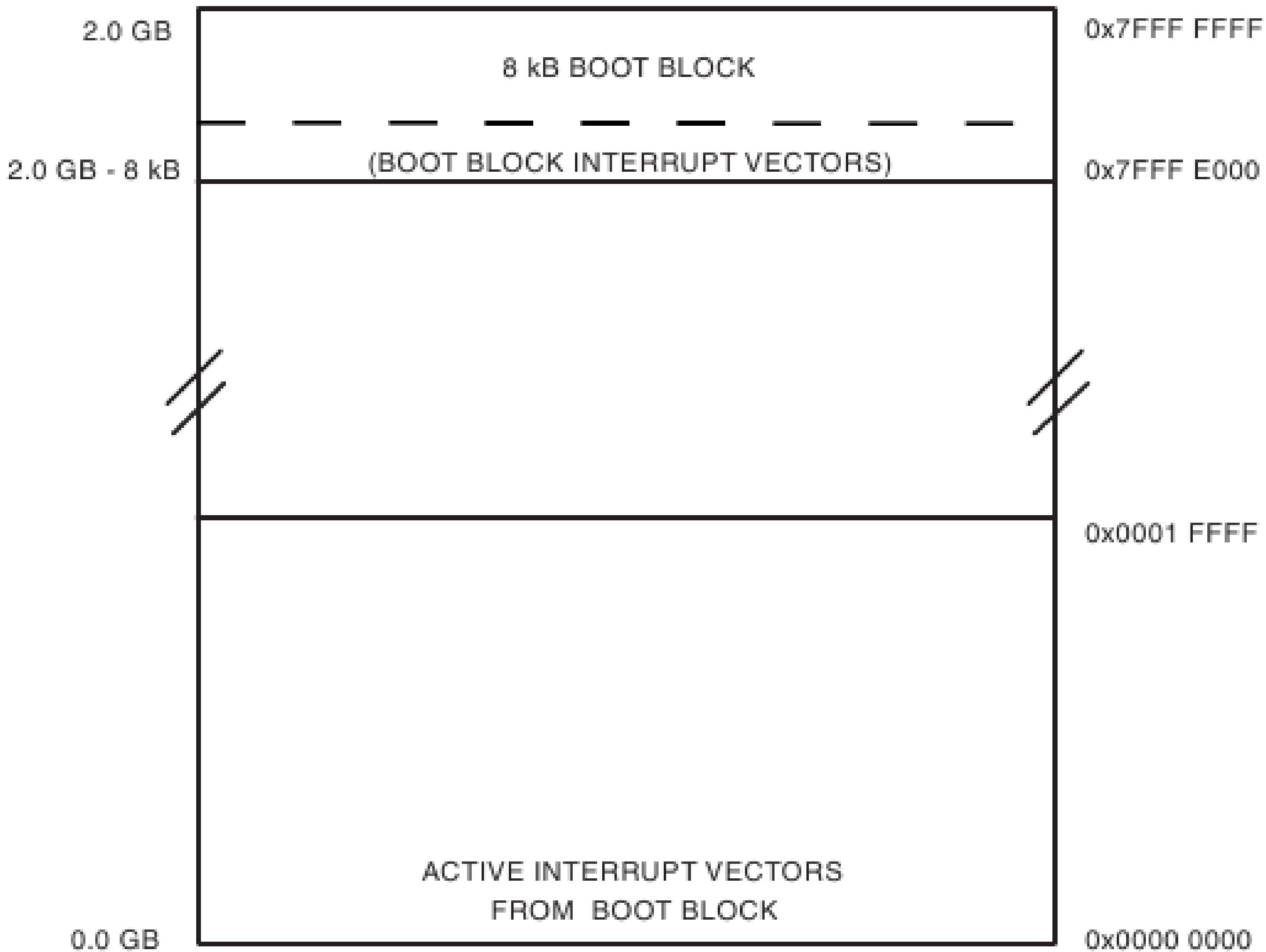
IOCLR	Description	Value after Reset
31:0	Output value CLEAR bits. Bit 0 corresponds to P0.0 ... Bit 31 corresponds to P0.31	0

IODIR	Description	Value after Reset
31:0	Direction control bits (0 = INPUT, 1 = OUTPUT). Bit 0 controls P0.0 ... Bit 31 controls P0.31	0

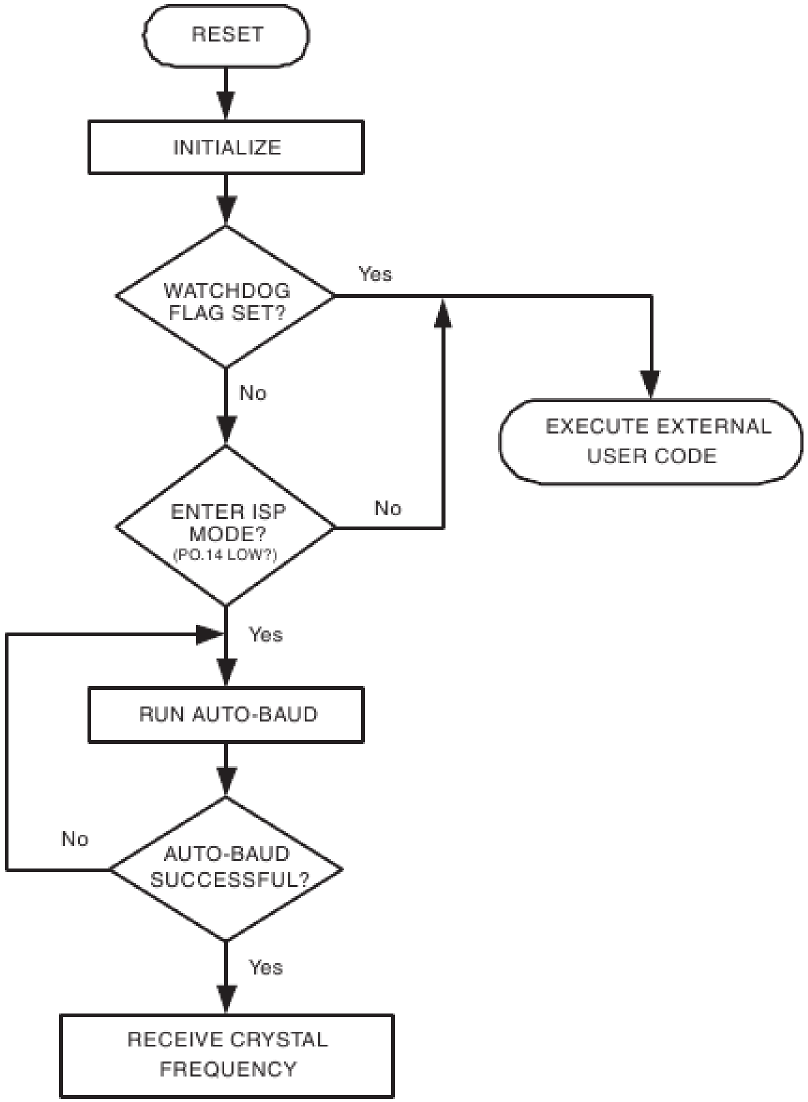
Block diagram



Memory Map (after Reset)



Boot Process



Execptions Vector locations

Address	Exception
0x0000 0000	Reset
0x0000 0004	Undefined Instruction
0x0000 0008	Software Interrupt
0x0000 000C	Prefetch Abort (instruction fetch memory fault)
0x0000 0010	Data Abort (data access memory fault)
0x0000 0014	Reserved *
0x0000 0018	IRQ
0x0000 001C	FIQ

- 0x0000 0014 – Checksum of all others positions

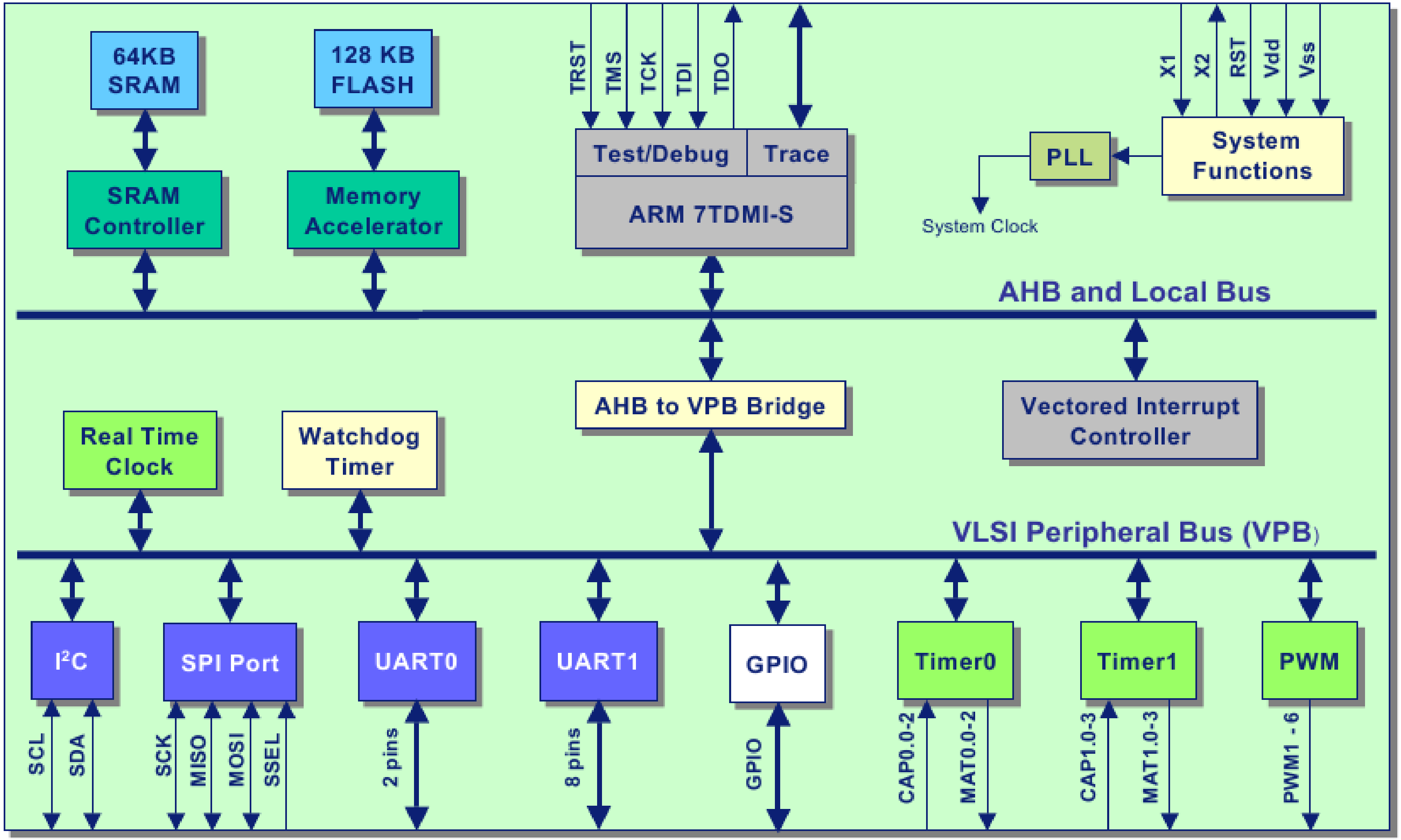
  0x0000 0000 – 0x0000 001C

Memory remap

Mode	Activation	Usage
Boot Loader mode	Hardware activation by any Reset	The Boot Loader always executes after any reset. The Boot Block interrupt vectors are mapped to the bottom of memory to allow handling exceptions and using interrupts during the Boot Loading process.
User Flash mode	Software activation by Boot code	Activated by Boot Loader when a valid User Program Signature is recognized in memory and Boot Loader operation is not forced. Interrupt vectors are not re-mapped and are found in the bottom of the Flash memory.
User RAM mode	Software activation by User program	Activated by a User Program as desired. Interrupt vectors are re-mapped to the bottom of the Static RAM.

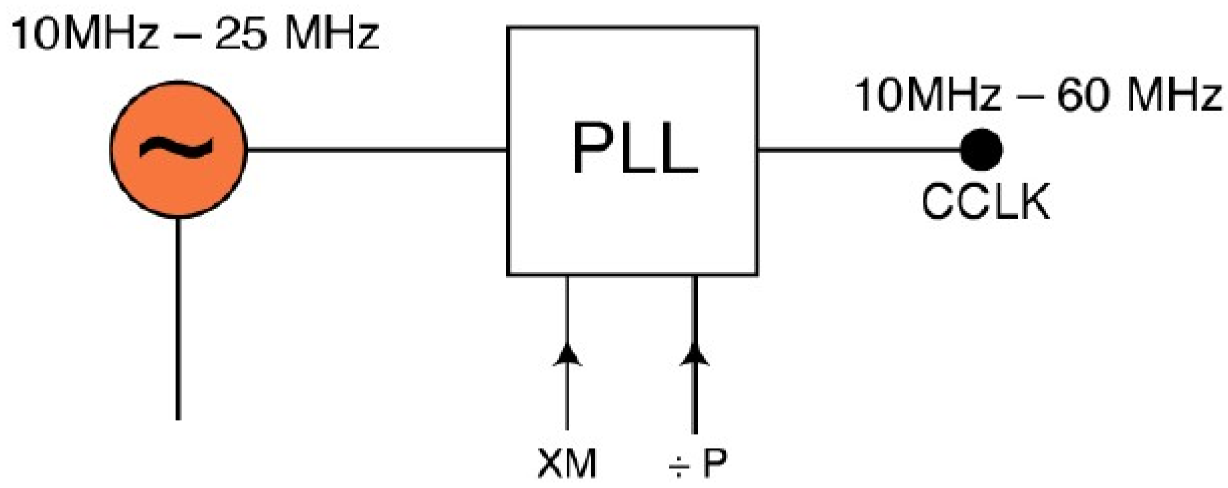
Memory Mapping Control				
0xE01FC040	MEMMAP	Memory mapping control.	R/W	0

Block diagram

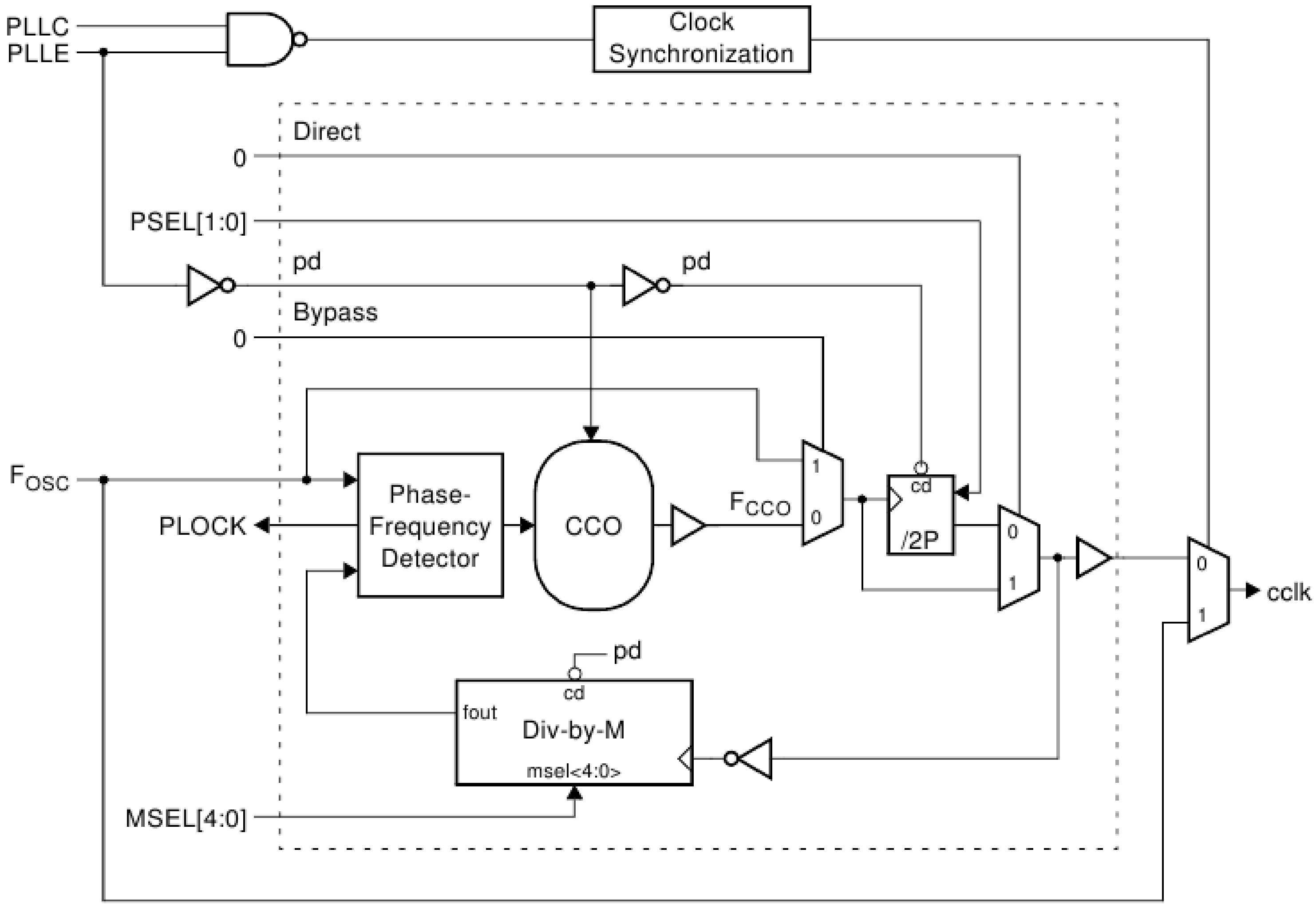


PLL

- Input clock frequency range:
  - 10 MHz to 25 MHz.
- Working clock frequency range (cclk):
  - 10 MHz to 60 MHz



PLL Block diagram



PLL Registers

Address	Name	Description	Access
0xE01FC080	PLLCON	Holding register for updating PLL control bits. Values written to this register do not take effect until a valid PLL feed sequence has taken place.	R/W
0xE01FC084	PLLCFG	Holding register for updating PLL configuration values. Values written to this register do not take effect until a valid PLL feed sequence has taken place.	R/W
0xE01FC088	PLLSTAT	Read-back register for PLL control and configuration information. If PLLCON or PLLCFG have been written to, but a PLL feed sequence has not yet occurred, they will not reflect the current PLL state. Reading this register provides the actual values controlling the PLL, as well as the status of the PLL.	RO
0xE01FC08C	PLLFEED	This register enables loading of the PLL control and configuration information from the PLLCON and PLLCFG registers into the shadow registers that actually affect PLL operation.	WO

- To changes PLLCON and PLLCFG
  - PLLFEED must be writen with sequence
    - 0xAA
    - 0x55

PLLCON

PLLCON	Function	Description	Reset Value
0	PLLE	PLL Enable. When one, and after a valid PLL feed, this bit will activate the PLL and allow it to lock to the requested frequency. See PLLSTAT register, Table 15.	0
1	PLLC	PLL Connect. When PLLC and PLLE are both set to one, and after a valid PLL feed, connects the PLL as the clock source for the LPC2106/2105/2104. Otherwise, the oscillator clock is used directly by the LPC2106/2105/2104. See PLLSTAT register, Table 15.	0
7:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PLLCFG

PLLCFG	Function	Description	Reset Value
4:0	MSEL4:0	PLL Multiplier value. Supplies the value "M" in the PLL frequency calculations.	0
6:5	PSEL1:0	PLL Divider value. Supplies the value "P" in the PLL frequency calculations.	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

cclk = M * Fosc or cclk = Fcco / (2 * P)

Fcco = Fosc * M * (2 * P)

- 10 MHz <= Fosc <= 25 MHz
- 10 MHz <= cclk <= 60 MHz
- 156 MHz <= Fcco <= 320 MHz

PLLCFG

PLLCFG	Function	Description	Reset Value
4:0	MSEL4:0	PLL Multiplier value. Supplies the value "M" in the PLL frequency calculations.	0
6:5	PSEL1:0	PLL Divider value. Supplies the value "P" in the PLL frequency calculations.	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PSEL Bits (PLLCFG bits 6:5)	Value of P
00	1
01	2
10	4
11	8

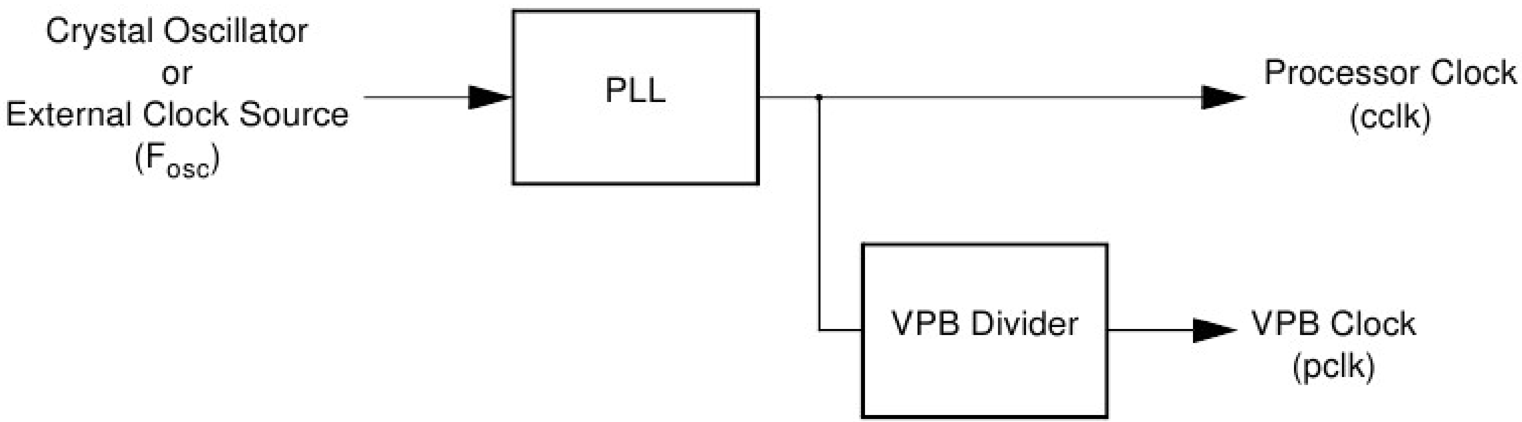
MSEL Bits (PLLCFG bits 4:0)	Value of M
00000	1
00001	2
00010	3
00011	4
...	...
11110	31
11111	32

PLLSTAT

PLLSTAT	Function	Description	Reset Value
4:0	MSEL4:0	Read-back for the PLL Multiplier value. This is the value currently used by the PLL.	0
6:5	PSEL1:0	Read-back for the PLL Divider value. This is the value currently used by the PLL.	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	PLLE	Read-back for the PLL Enable bit. When one, the PLL is currently activated. When zero, the PLL is turned off. This bit is automatically cleared when Power Down mode is activated.	0
9	PLLC	Read-back for the PLL Connect bit. When PLLC and PLLE are both one, the PLL is connected as the clock source for the LPC2106/2105/2104. When zero, the PLL is bypassed and the oscillator clock is used directly by the LPC2106/2105/2104. This bit is automatically cleared when Power Down mode is activated.	0
10	PLOCK	Reflects the PLL Lock status. When zero, the PLL is not locked. When one, the PLL is locked onto the requested frequency.	0
15:11	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

VPBDIV

- Relationship between the processor clock (cclk) and the clock used by peripheral devices (pclk)
  - provides peripherals with desired pclk via VPB bus
  - allow power savings when an application does not require any peripherals to run at the full processor rate.

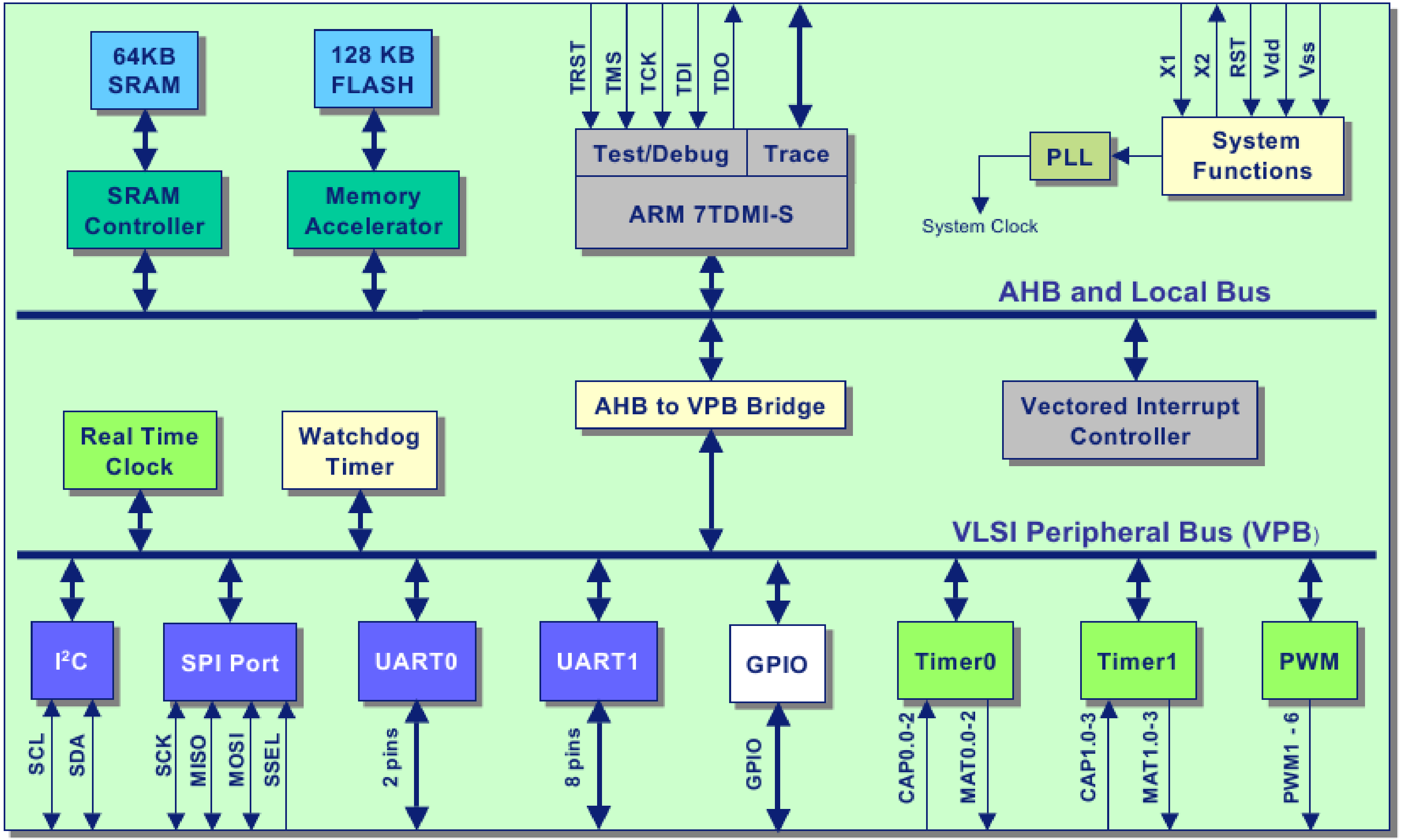


VPBDIV

Address	Name	Description	Access
0xE01FC100	VPBDIV	Controls the rate of the VPB clock in relation to the processor clock.	R/W

VPBDIV	Function	Description	Reset Value
1:0	VPBDIV	The rate of the VPB clock is as follows: 0 0: VPB bus clock is one fourth of the processor clock. 0 1: VPB bus clock is the same as the processor clock. 1 0: VPB bus clock is one half of the processor clock. 1 1: Reserved. If this value is written to the VPBDIV register, it has no effect (the previous setting is retained).	0
7:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Block diagram



Memory Accelerator Module

- Attempts to have the next ARM instruction that will be needed in its latches in time to prevent CPU fetch stalls.

  - Split the Flash memory into two banks
    - Independent accesses.
    - Prefetch Buffer and Branch Trail Buffer

  - Sequential code execution
    - one Flash bank - current instruction
    - other Flash bank - next sequential code line

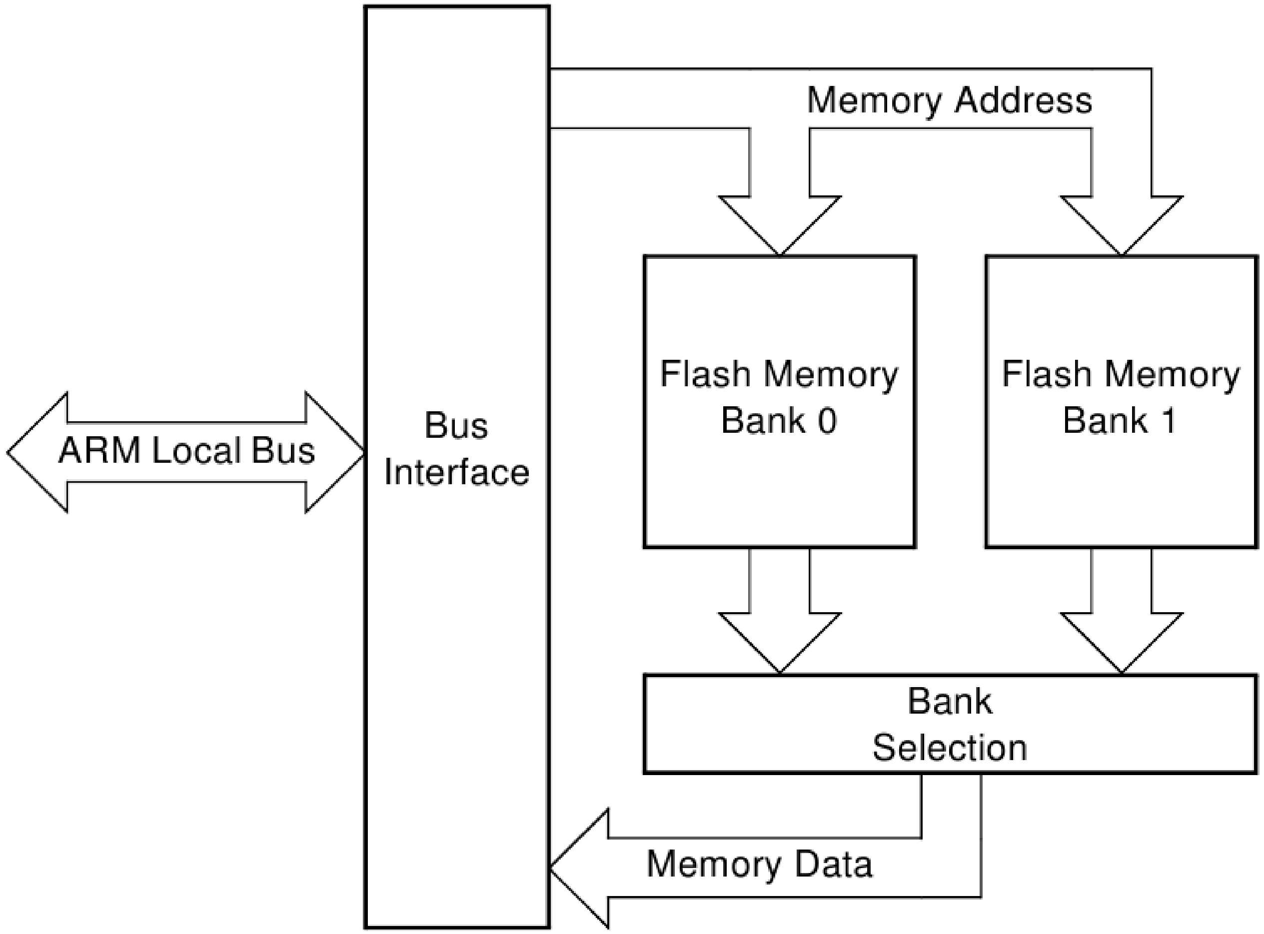
Memory Accelerator Module

- Branches cause a break in the sequential flow of instruction fetches.
  - In Branch Trail Buffers - execution continues without the need for a Flash read cycle.
  - Outside Branch Trail and Prefetch buffers - one Flash Access cycle is needed to load the Branch Trail buffers.

- Allows faster access to data in sequencial accesses

- There is no prefetch function for data accesses.

Memory Accelerator Module



MAM Registers

Address	Name	Description	Access	Reset Value*
MAM				
0xE01FC000	MAMCR	Memory Accelerator Module Control Register. Determines the MAM functional mode, that is, to what extent the MAM performance enhancements are enabled. See Table 28.	R/W	0
0xE01FC004	MAMTIM	Memory Accelerator Module Timing control. Determines the number of clocks used for Flash memory fetches (1 to 7 processor clocks).	R/W	0x07

MAMCR

MAMCR	Function	Description	Reset Value
1:0	MAM mode control	These bits determine the operating mode of the MAM as follows: 0 0 - MAM functions disabled. 0 1 - MAM functions partially enabled. 1 0 - MAM functions fully enabled. 1 1 - reserved	0
7:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

- Disable - All memory requests result in a Flash read operation
- Partially enabled - Sequential instruction accesses
- Fully enabled - Any memory request (code or data)

MAMTIM

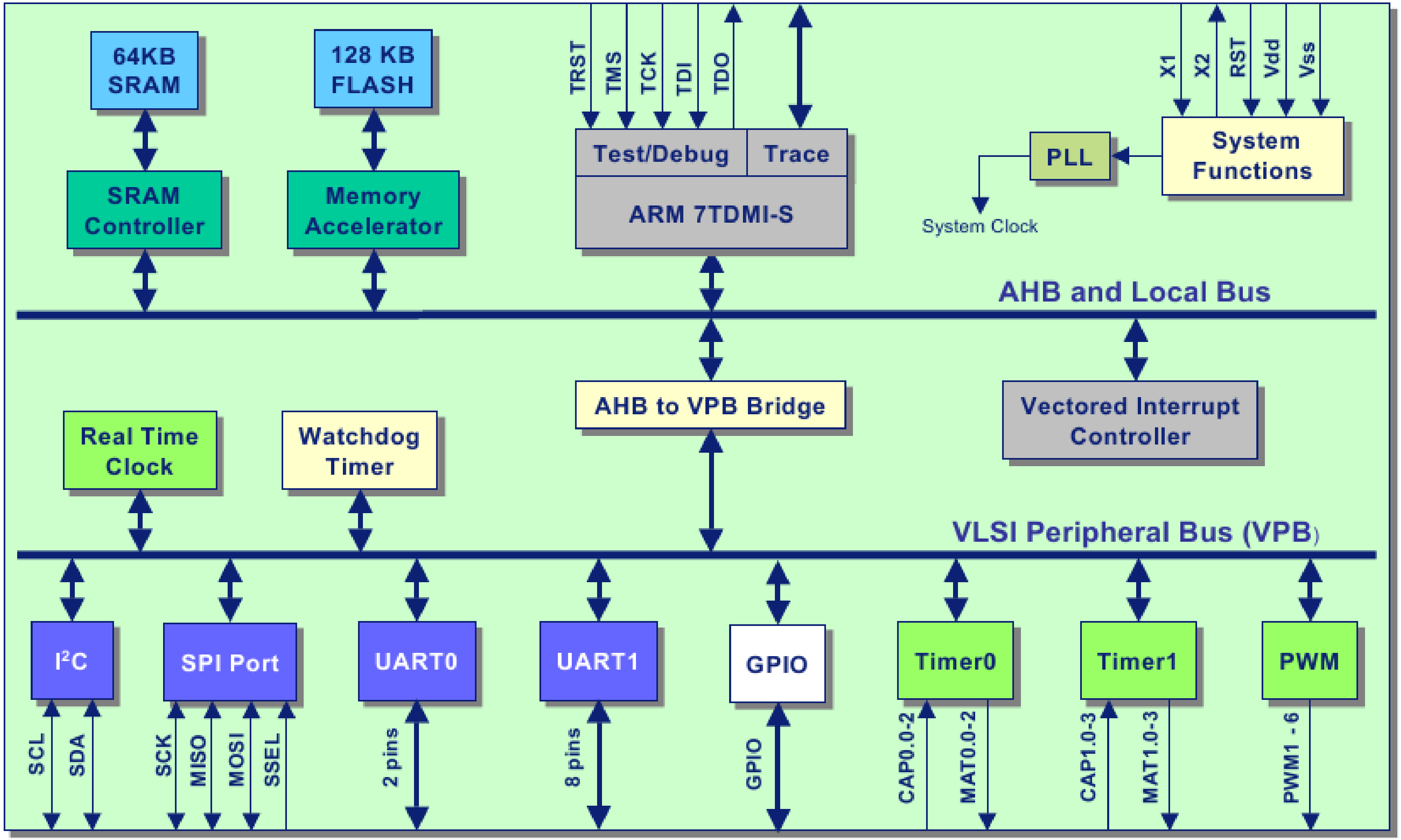
MAMTIM	Function	Description	Reset Value
2:0	MAM Fetch Cycle timing	These bits set the duration of MAM Flash fetch operations as follows: 0 0 0 = 0 - Reserved. 0 0 1 = 1 - MAM fetch cycles are 1 processor clock (cclk) in duration. 0 1 0 = 2 - MAM fetch cycles are 2 processor clocks (cclks) in duration. 0 1 1 = 3 - MAM fetch cycles are 3 processor clocks (cclks) in duration. 1 0 0 = 4 - MAM fetch cycles are 4 processor clocks (cclks) in duration. 1 0 1 = 5 - MAM fetch cycles are 5 processor clocks (cclks) in duration. 1 1 0 = 6 - MAM fetch cycles are 6 processor clocks (cclks) in duration. 1 1 1 = 7 - MAM fetch cycles are 7 processor clocks (cclks) in duration. **Warning:** Improper setting of this value may result in incorrect operation of the device.	0x07
7:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

- system clock < 20 MHz → MAMTIM = 1
- 20 MHz < system clock < 40 MHz → MAMTIM = 2
- system clock > 40 MHz → MAMTIM = 3

StartOsc

```
void StartOsc(void) {
    PLLCFG = 0x23;          // Setting M and P values
    PLLFEED = 0xAA; PLLFEED = 0x55;
    PLLCON = 0x1;           // Enabling the PLL
    PLLFEED = 0xAA; PLLFEED = 0x55;
    While ( !(PLLSTAT & PLOCK) );    // Wait for the PLL to lock
    PLLCON = 0x3;           // Connect the PLL as the clock source
    PLLFEED = 0xAA; PLLFEED = 0x55;
    MAMCR = 0x2;            // Enabling MAM and setting number
    MAMTIM = 0x4;           // of clocks used for Flash memory fetch
    VPBDIV = 0x1;           // Setting pclk to cclk
}
```

Block diagram

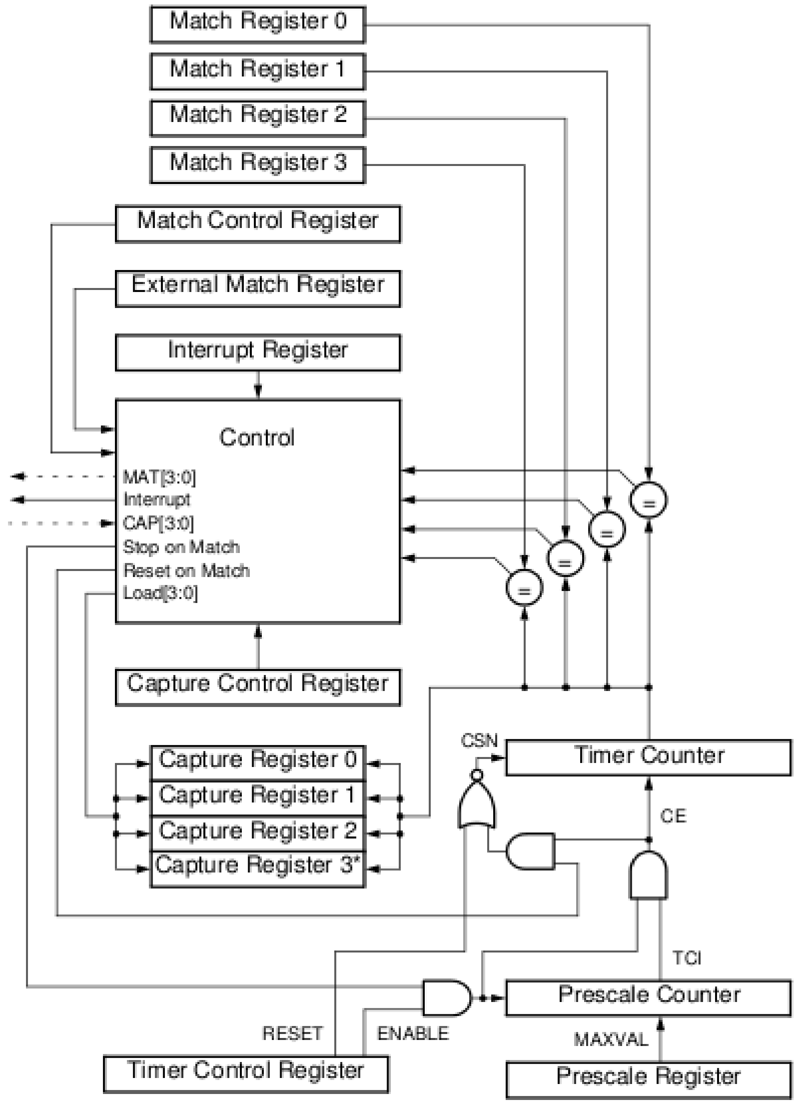


Timers

- 32-bit programmable with Prescaler.
- Capture
  - snapshot of the timer valueCan generate interrupt.
- Match
  - Continuous operation with interrupt generation.
  - Stop timer with interrupt generation.
  - Reset timer with interrupt generation.
- External outputs capabilities
  - Set low.
  - Set high.
  - Toggle.

Block Diagram

- tick rate controlled by prescaler
- prescale counter increment on PCLK
  - until value of prescaler register
    - timer counter is incremented by one
    - prescale counter resets and restart.
- has up to four capture channels
  - allow capture the value of the timer counter



External pin interface

Pin name	Pin direction	Pin Description
CAP0.2..0 CAP1.3..0	Input	**Capture Signals**- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt.
MAT0.0 MAT1.0	Output	**External Match Output 0/1-** When match register 0/1 (MR0/1) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output.
MAT0.1 MAT1.1	Output	**External Match Output 1-** See the MAT0/MAT1 description above.
MAT0.2 MAT1.2	Output	**External Match Output 2-** See the MAT0/MAT1 description above.
MAT1.3	Output	**External Match Output 3-** See the MAT1 description above.

Config & Status Registers

Generic Name	Timer 0 Address & Name	Timer 1 Address & Name	Description	Access	Reset Value*
IR	0xE0004000 T0IR	0xE0008000 T1IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight (Timer 1) or seven (Timer 0) possible interrupt sources are pending.	R/W	0
TCR	0xE0004004 T0TCR	0xE0008004 T1TCR	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	R/W	0
TC	0xE0004008 T0TC	0xE0008008 T1TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of pclk. The TC is controlled through the TCR.	RW	0
PR	0xE000400C T0PR	0xE000800C T1PR	Prescale Register. The TC is incremented every PR+1 cycles of pclk.	R/W	0
PC	0xE0004010 T0PC	0xE0008010 T1PC	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented.	R/W	0

Timer Counter (TC)

- 32-bit Timer Counter

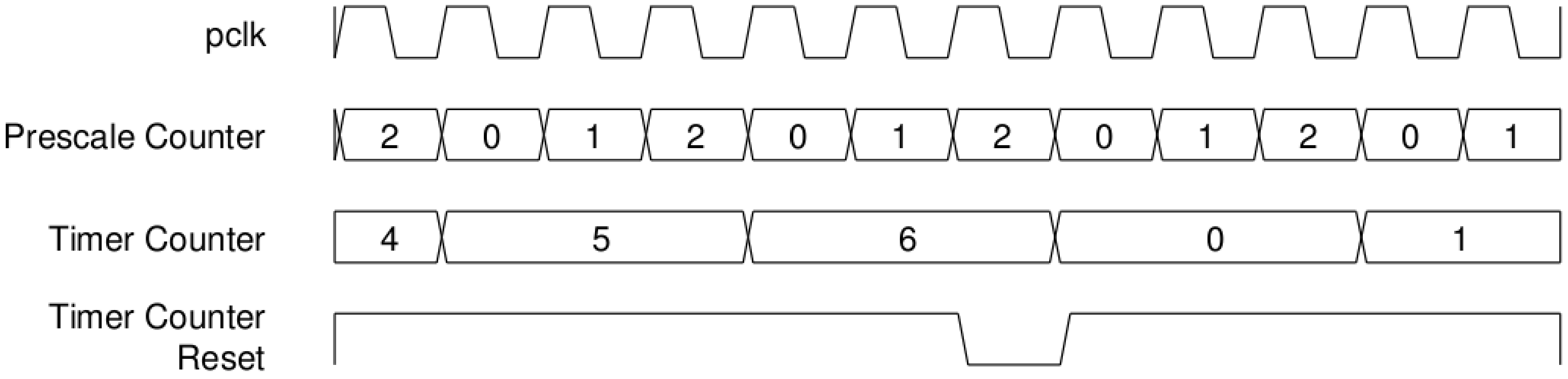
- incremented when the Prescale Counter (PC) reaches its terminal count.
  - Unless it is reset before reaching its upper limit, the TC will count up through the value 0xFFFFFFFF and then wrap back to the value 0x00000000.

- The upper limit reaching event does not cause an interrupt
  - A Match register can be used to detect an overflow.

Prescale

- Prescale Register (PR)
  - The 32-bit Prescale Register specifies the maximum value for the Prescale Counter.
- Prescale Counter Register (PC)
  - 32-bit Prescale Counter controls division of PCLK by some constant value before it is applied to the Timer Counter.
  - Allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows.
  - Is incremented on every PCLK.
    - When it reaches the value stored in the Prescale Register
      - Timer Counter is incremented
      - Prescale Counter is reset on the next PCLK.
        - TC is increment on every PCLK when PR = 0
        - TC is increment on every 2 pclks when PR = 1

Example Timer Operation



- PR = 2
- MR0 = 6
- Reset on match

Match Registers

Generic Name	Timer 0 Address & Name	Timer 1 Address & Name	Description	Access	Reset Value*
MCR	0xE0004014 T0MCR	0xE0008014 T1MCR	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	R/W	0
MR0	0xE0004018 T0MR0	0xE0008018 T1MR0	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	R/W	0
MR1	0xE000401C T0MR1	0xE000801C T1MR1	Match Register 1. See MR0 description.	R/W	0
MR2	0xE0004020 T0MR2	0xE0008020 T1MR2	Match Register 2. See MR0 description.	R/W	0
MR3	0xE0004024 T0MR3	0xE0008024 T1MR3	Match Register 3. See MR0 description.	R/W	0

Capture Registers

Generic Name	Timer 0 Address & Name	Timer 1 Address & Name	Description	Access	Reset Value*
CCR	0xE0004028 T0CCR	0xE0008028 T1CCR	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	R/W	0
CR0	0xE000402C T0CR0	0xE000802C T1CR0	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the capture[0] signal.	RO	0
CR1	0xE0004030 T0CR1	0xE0008030 T1CR1	Capture Register 1. See CR0 description.	RO	0
CR2	0xE0004034 T0CR2	0xE0008034 T1CR2	Capture Register 2. See CR0 description.	RO	0
CR3	0xE0004038 T0CR3	0xE0008038 T1CR3	Capture Register 3. See CR0 description. Not usable on Timer 0.	RO	0
EMR	0xE000403C T0EMR	0xE000803C T1EMR	External Match Register. The EMR controls the external match pins MATn.	R/W	0

Timer Control Register

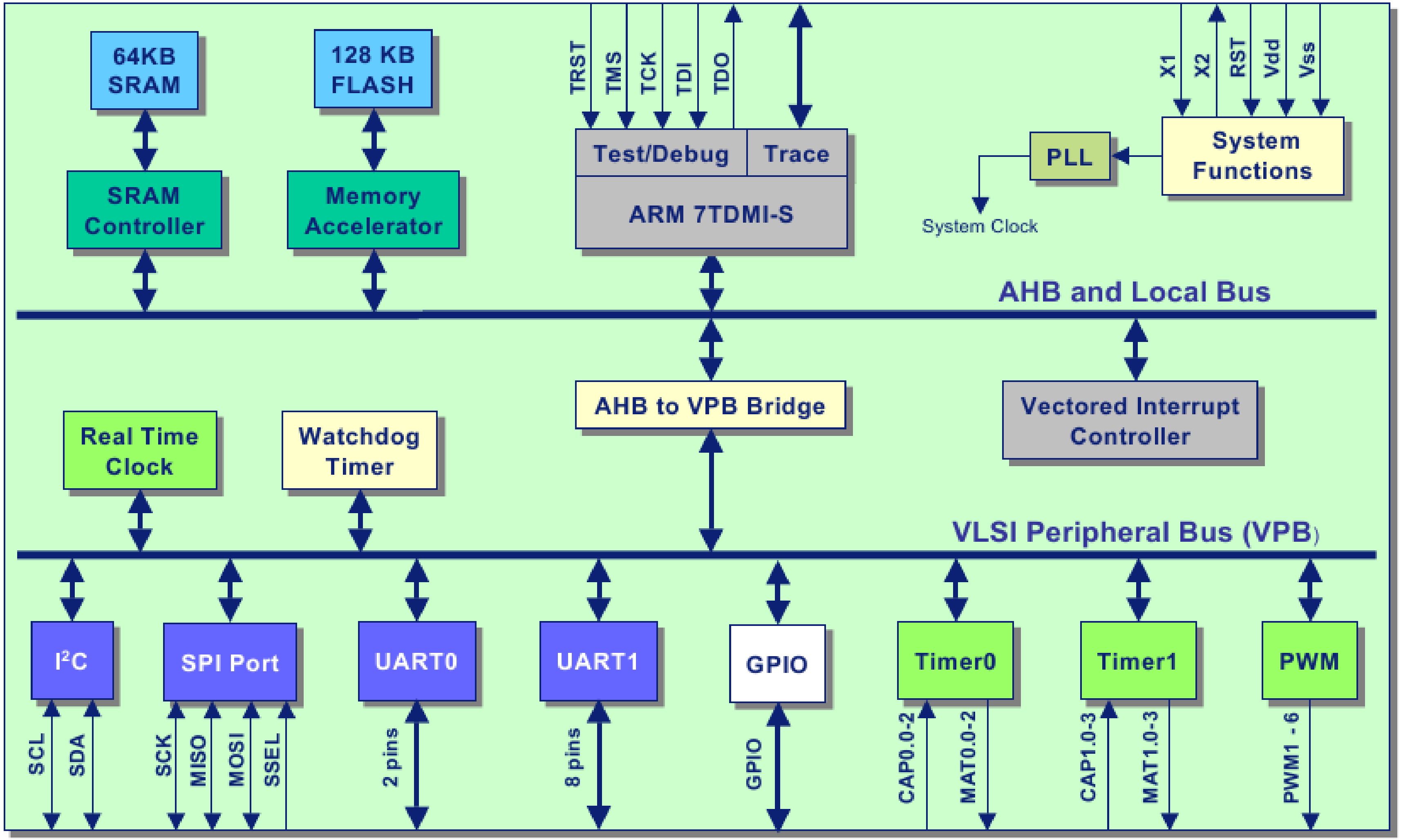
TCR	Function	Description	Reset Value
0	Counter Enable	When one, the Timer Counter and Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of pclk. The counters remain reset until TCR[1] is returned to zero.	0

Delay 1s

```c
#define TIMER0_IR     (*((volatile unsigned long *) 0xE0004000))
#define TIMER0_TCR    (*((volatile unsigned long *) 0xE0004004))
#define TIMER0_PR     (*((volatile unsigned long *) 0xE000400C))
#define TIMER0_MCR    (*((volatile unsigned long *) 0xE0004014))
#define TIMER0_MR0    (*((volatile unsigned long *) 0xE0004018))
#define TIMER0_EMR    (*((volatile unsigned long *) 0xE000403C))

void delay()
{
        TIMER0_TCR = 2; TIMER0_EMR = 0;
        TIMER0_PR = 0; TIMER0_MR0 = PCLK;
        TIMER0_MCR = TIMER_INT_MASK | TIMER_RESTART_MASK;
        TIMER0_TCR = 1;
        do {
        } while ( !(TIMER0_IR & TIMER_INT_MASK) );
        TIMER0_IR = TIMER_INT_MASK;
}
```

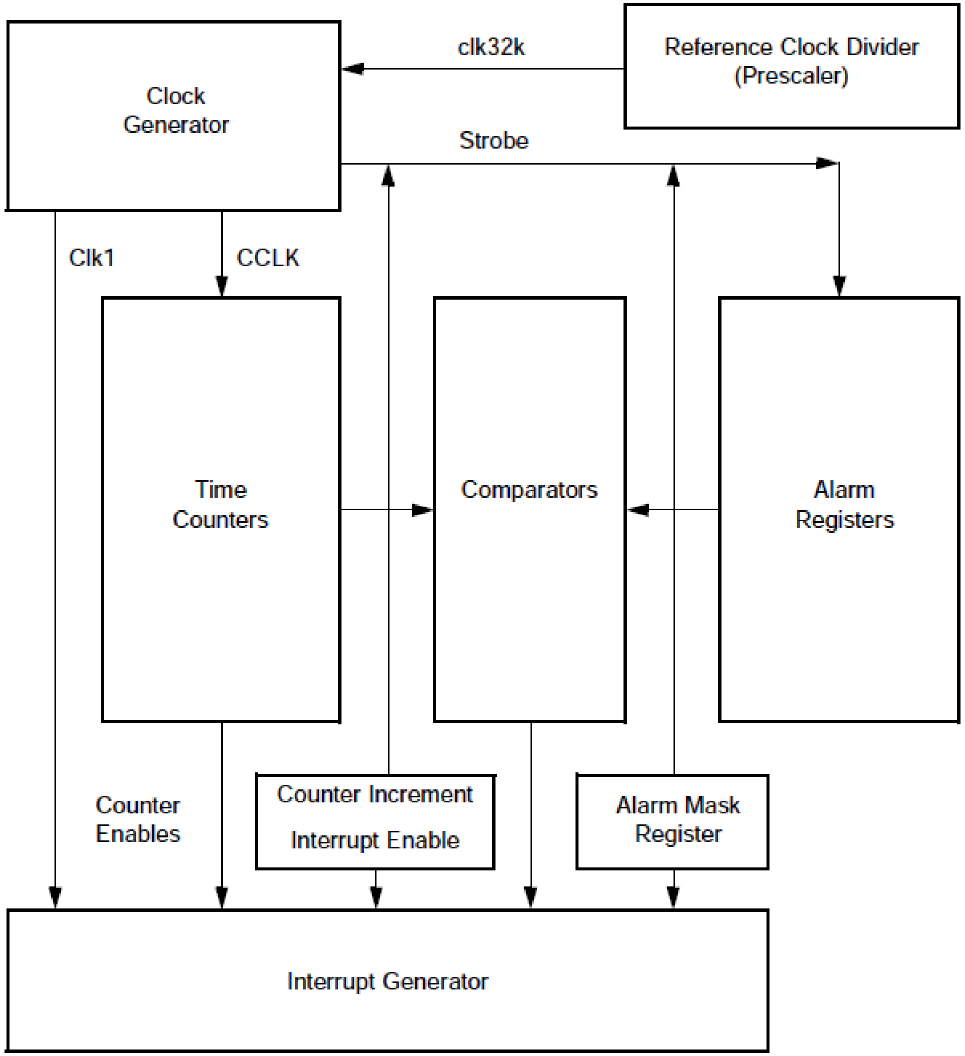
Block diagram



RTC – Real Time Clock

- Set of counters to measure time
  - Measures the passage of time to maintain a calendar and clock
    - Seconds, Minutes, Hours
    - Day of Month, Month, Year, Day of Week, and Day of Year.
- Set of alarms
- Ultra low power
- Programmable reference clock divider allows adjustment of the RTC to match various crystal frequencies.

RTC - Architecture



RTC - Registers

Address	Name	Size	Description	Access
0xE0024000	ILR	2	Interrupt Location. Reading this location indicates the source of an interrupt. Writing a one to the appropriate bit at this location clears the associated interrupt.	RW
0xE0024004	CTC	15	Clock Tick Counter. Value from the clock divider.	RO
0xE0024008	CCR	4	Clock Control Register. Controls the function of the clock divider.	RW
0xE002400C	CIIR	8	Counter Increment Interrupt. Selects which counters will generate an interrupt when they are incremented.	RW
0xE0024010	AMR	8	Alarm Mask Register. Controls which of the alarm registers are masked.	RW
0xE0024014	CTIME0	32	Consolidated Time Register 0	RO
0xE0024018	CTIME1	32	Consolidated Time Register 1	RO
0xE002401C	CTIME2	32	Consolidated Time Register 2	RO

RTC – Time Registers

Address	Name	Size	Description	Access
0xE0024020	SEC	6	Seconds value in the range of 0 to 59.	R/W
0xE0024024	MIN	6	Minutes value in the range of 0 to 59.	R/W
0xE0024028	HOUR	5	Hours value in the range of 0 to 23.	R/W
0xE002402C	DOM	5	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year). [1]	R/W
0xE0024030	DOW	3	Day of week value in the range of 0 to 6. [1]	R/W
0xE0024034	DOY	9	Day of year value in the range of 1 to 365 (366 for leap years). [1]	R/W
0xE0024038	MONTH	4	Month value in the range of 1 to 12.	R/W
0xE002403C	YEAR	12	Year value in the range of 0 to 4095.	R/W

RTC – Alarms Registers

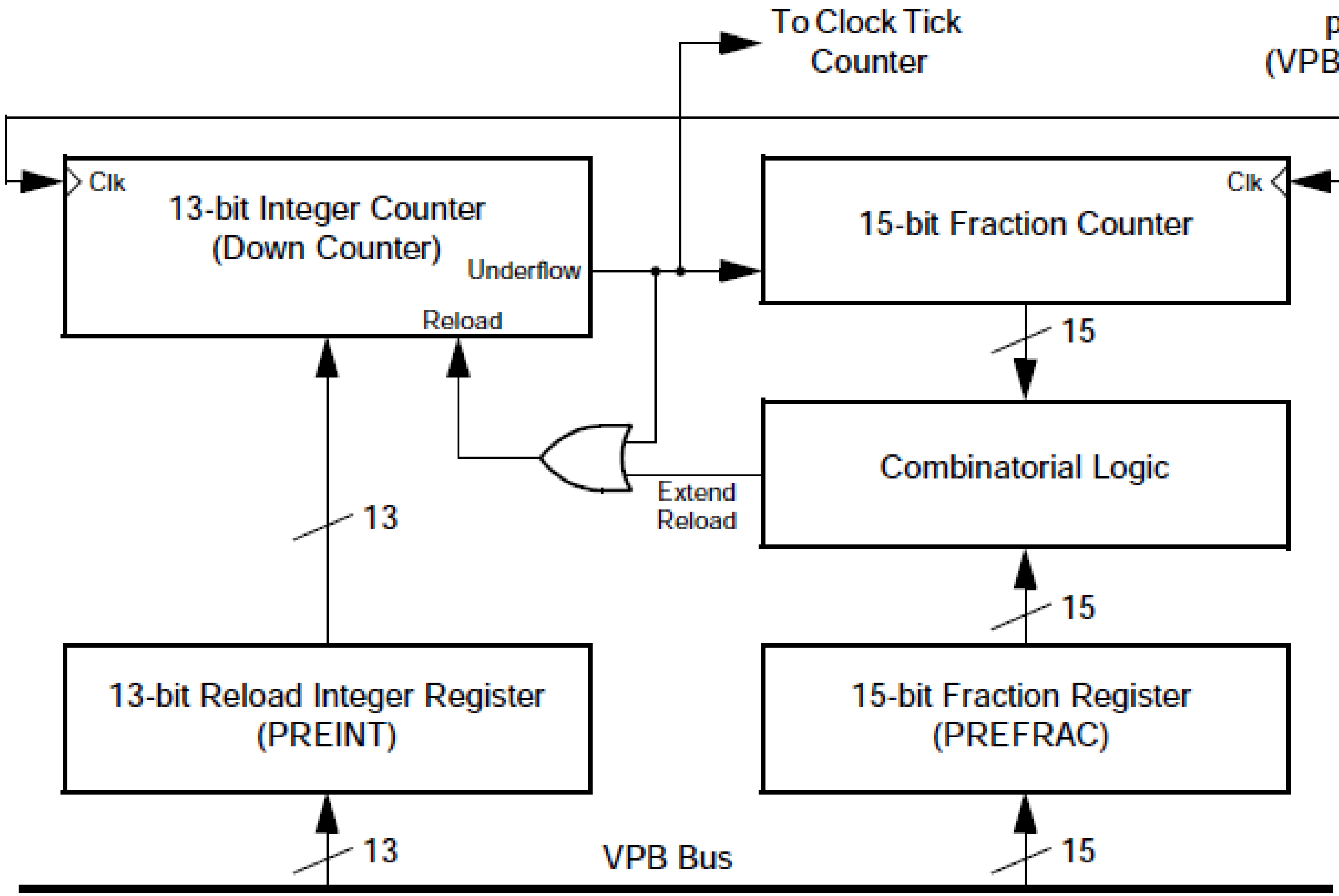
AMR	Function	Description
0	AMRSEC	When one, the Second value is not compared for the alarm.
1	AMRMIN	When one, the Minutes value is not compared for the alarm.
2	AMRHOUR	When one, the Hour value is not compared for the alarm.
3	AMRDOM	When one, the Day of Month value is not compared for the alarm.
4	AMRDOW	When one, the Day of Week value is not compared for the alarm.
5	AMRDOY	When one, the Day of Year value is not compared for the alarm.
6	AMRMON	When one, the Month value is not compared for the alarm.
7	AMRYEAR	When one, the Year value is not compared for the alarm.

RTC – clock dividers

Address	Name	Size	Description	Access
0xE0024080	PREINT	13	Prescale Value, integer portion	R/W
0xE0024084	PREFRAC	15	Prescale Value, fractional portion	R/W

- Reference clock divider
  - Generate a 32.768 kHz from pclk

- PREINT = int (pclk / 32768) – 1
  - PREINT > 1
- PREFRAC = pclk - ((PREINT + 1) x 32768)


To Clock Tick Counter
pclk (VPB Clock)
Clk
13-bit Integer Counter (Down Counter)
Underflow
Reload
Clk
15-bit Fraction Counter
15
Combinatorial Logic
Extend Reload
13
15
13-bit Reload Integer Register (PREINT)
15-bit Fraction Register (PREFRAC)
13
VPB Bus
15

RTC – CTIME0 Register

CTIME0	Function	Description
31:27	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
26:24	Day of Week	Day of week value in the range of 0 to 6.
23:21	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
20:16	Hours	Hours value in the range of 0 to 23.
15:14	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
13:8	Minutes	Minutes value in the range of 0 to 59.
7:6	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
5:0	Seconds	Seconds value in the range of 0 to 59.

RTC – CTIME1/2 Registers

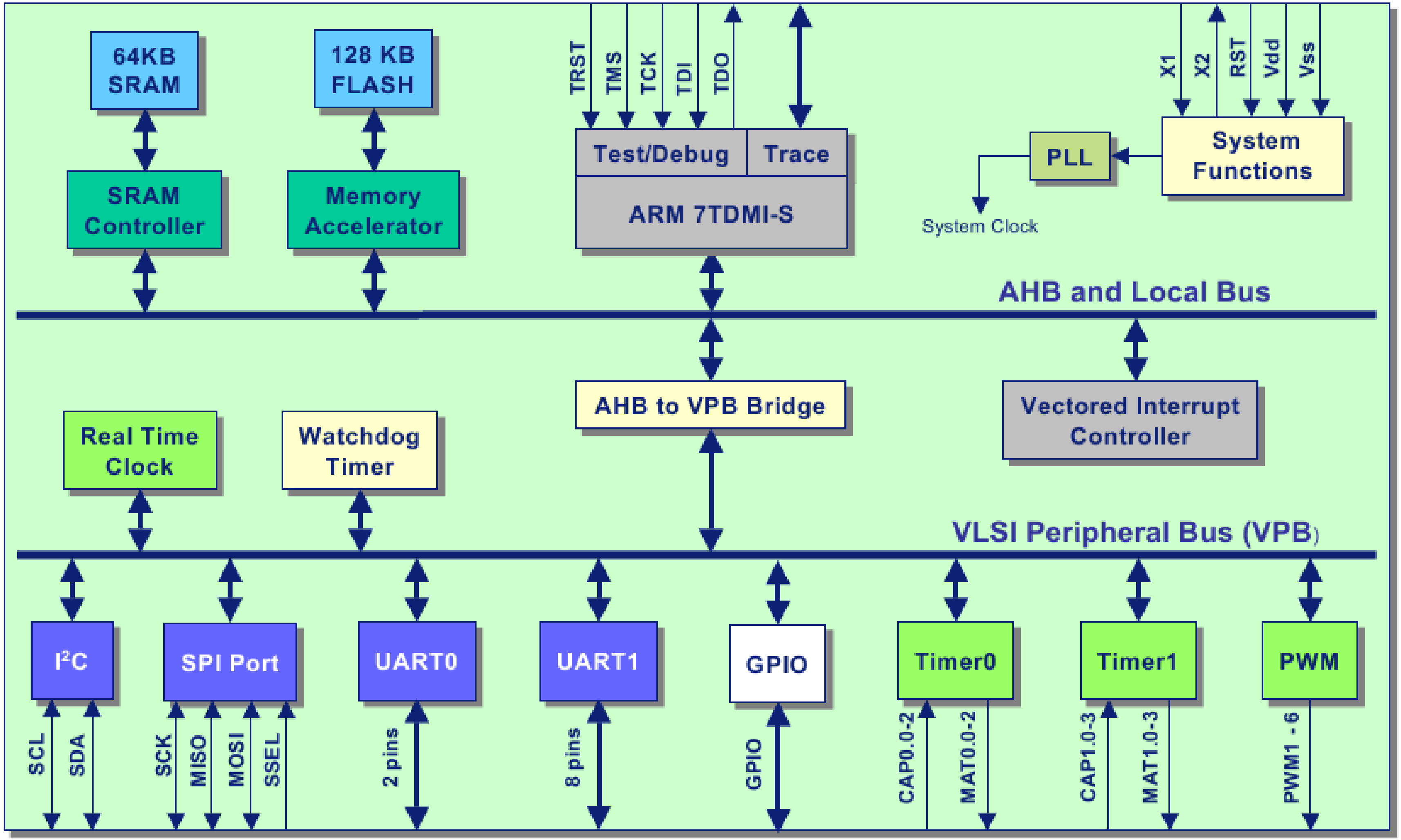
CTIME1	Function	Description
31:28	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
27:16	Year	Year value in the range of 0 to 4095.
15:12	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
11:8	Month	Month value in the range of 1 to 12.
7:5	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
4:0	Day of Month	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year).

CTIME2	Function	Description
11:0	Day of Year	Day of year value in the range of 1 to 365 (366 for leap years).
31:12	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

RTC – Sample code

```
void InitRtc(void) {
	PCONP |= (1 << 10);
	RTC_CCR = 0x2; /* Disable time counters and reset CTC */
	RTC_PREINT = (GetPclk() / 32768) - 1; /* Set PREINT */
	RTC_PREFRAC = GetPclk() - ((RTC_PREINT + 1) * 32768); /* Set PREFRAC */
	RTC_ILR = 0x3; /* Clear interrupts */
	RTC_ILR = 0x0; /* Disable interrupts */
	RTC_CIIR = 0x0; /* CIIR disabled */
	RTC_AMR = 0xFF; /* Alarm disabled */
	/* dummy initialization 00:00:00 1-Jan-1900*/
	SetTime(0, 0, 0);
	SetDate(1, 1, 1900);
	RTC_CCR = 0x1; /* Enable time counters and CTC */
}
```

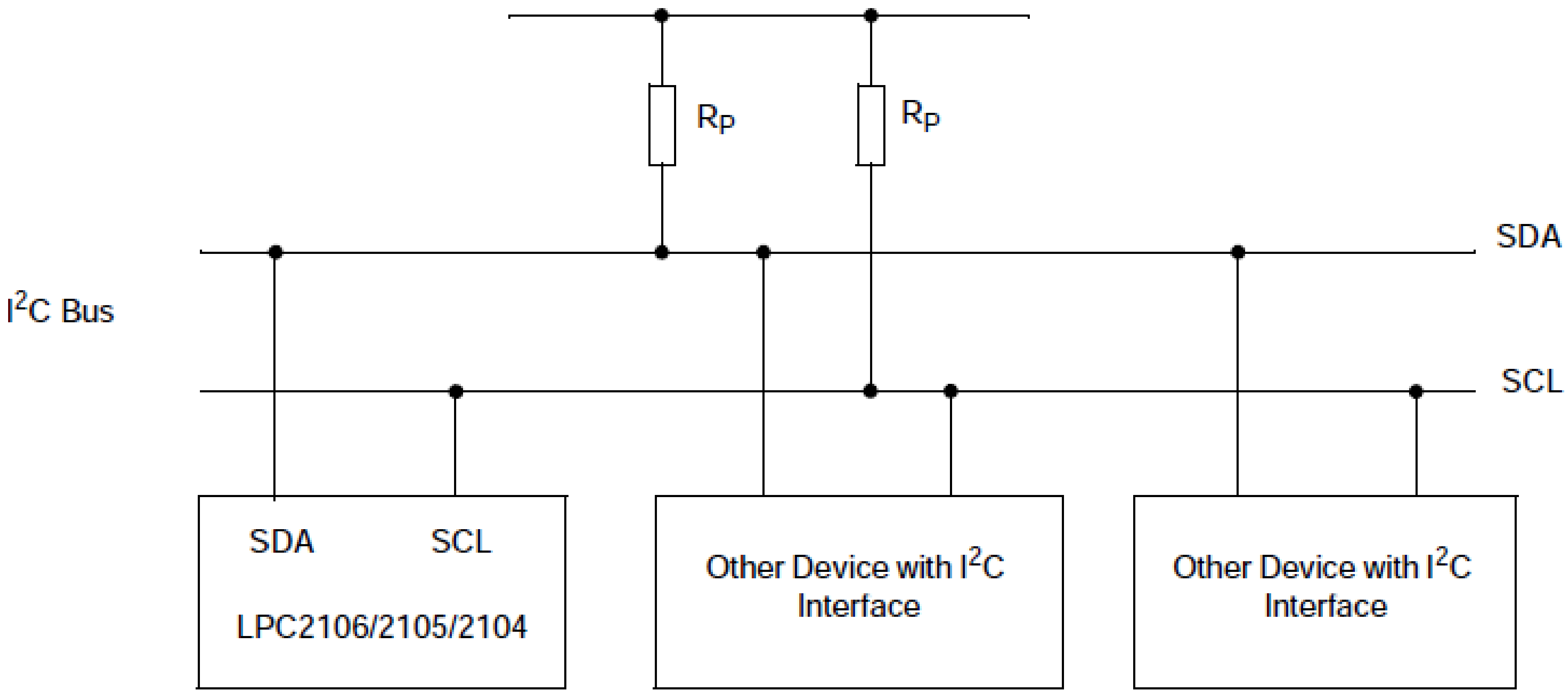
Block diagram



I2C Interface

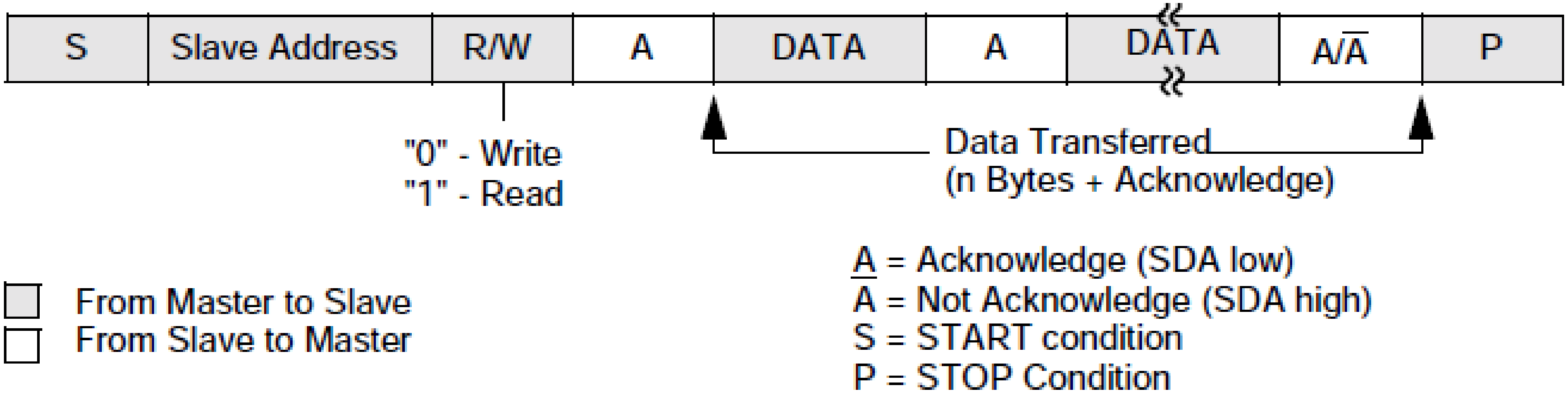
- Standard I2C compliant bus interface.
- Configurable as Master, Slave, or Master/Slave.
- Programmable clocks
- Bidirectional data transfer between masters and slaves.
- Multi-master bus (no central master).

I2C Bus configuration



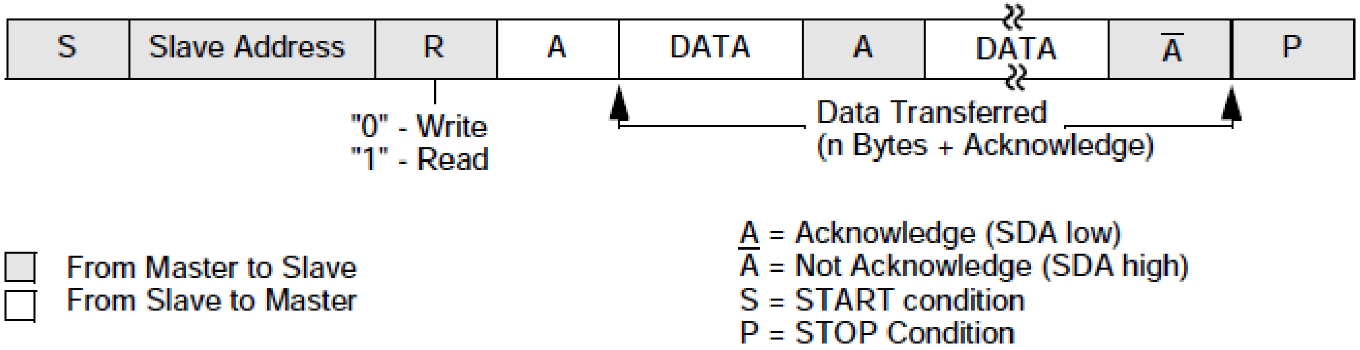
I2C Data transfers

- Master → Slave
  - First byte - slave address
  - Next bytes - data
  - Slave returns an acknowledge bit after each received byte.

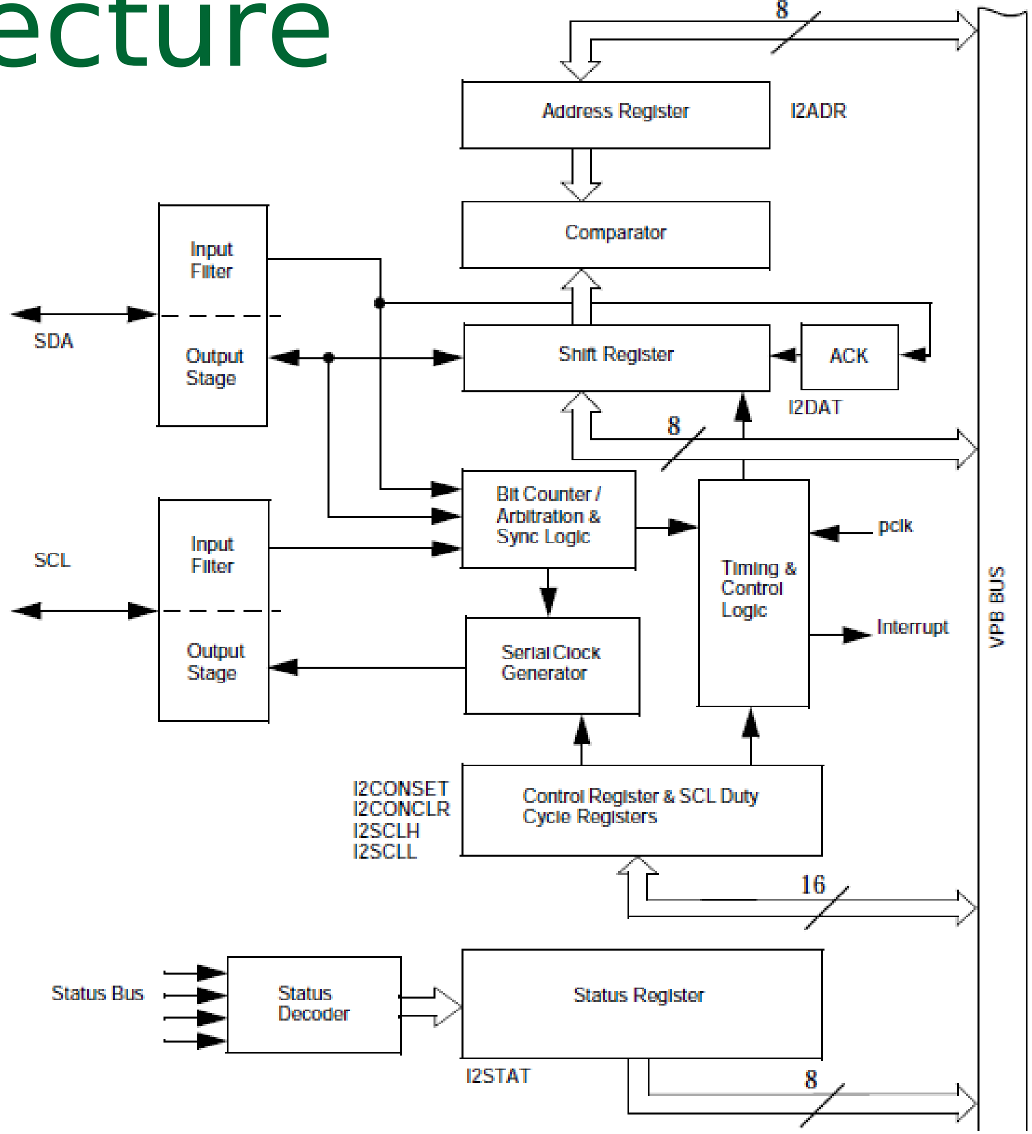


I2C Data transfers

- Slave → Master
  - First byte - slave address
    - transmitted by the master
    - slave returns acknowledge bit.
  - Next bytes – data
    - master returns acknowledge bit after all received bytes
    - last received byte returns “not acknowledge”



I2C Architecture



I2C - Registers

Address	Name	Description	Access	Reset Value*
0xE001C000	I2CONSET	I^2C Control Set Register	Read/Set	0
0xE001C004	I2STAT	I^2C Status Register	Read Only	0xF8
0xE001C008	I2DAT	I^2C Data Register	Read/Write	0
0xE001C00C	I2ADR	I^2C Slave Address Register	Read/Write	0
0xE001C010	I2SCLH	SCL Duty Cycle Register High Half Word	Read/Write	0x04
0xE001C014	I2SCLL	SCL Duty Cycle Register Low Half Word	Read/Write	0x04
0xE001C018	I2CONCLR	I^2C Control Clear Register	Clear Only	NA

I2C - I2CONSET

I2CONSET	Function	Description	Reset Value
0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
1	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AA	Assert acknowledge flag	0
3	SI	I^2C interrupt flag	0
4	STO	STOP flag	0
5	STA	START flag	0
6	I2EN	I^2C interface enable	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

I2C – I2CONSET

- AA - Assert Acknowledge Flag
  - When 1 → acknowledge (low level to SDA)
    - Address in the Slave Address Register has been received.
    - General call address has been received while bit GC is set.
    - Data byte received while the I2C is in the master receiver
    - Data byte received while the I2C is in the address slave receiver mode
  - When 0 → not acknowledge (high level to SDA)
    - Data byte has been received while the I2C is in the master receiver mode.
    - data byte received while the I2C is in the address slave receiver mode.

I2C – I2CONSET

- SI - Interrupt Flag
  - Set when one of the 25 possible I2C states is entered
- STO - STOP flag
  - transmit a STOP condition in master mode
  - recover from an error condition in slave mode
- STA - START flag
  - transmit a START condition or transmit a repeated START condition if it is already in master mode.
- I2EN - Interface Enable
  - 1 - I2C function is enabled

I2C - I2CONCLR

I2CONCLR	Function	Description	Reset Value
0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
1	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AAC	Assert Acknowledge Clear bit. Writing a 1 to this bit clears the AA bit in the I2CONSET register. Writing 0 has no effect.	NA
3	SIC	I^2C Interrupt Clear Bit. Writing a 1 to this bit clears the SI bit in the I2CONSET register. Writing 0 has no effect.	NA
4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
5	STAC	Start flag clear bit. Writing a 1 to this bit clears the STA bit in the I2CONSET register. Writing 0 has no effect.	NA
6	I2ENC	I^2C interface disable. Writing a 1 to this bit clears the I2EN bit in the I2CONSET register.Writing 0 has no effect.	NA
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

I2C - I2STAT

I2STAT	Function	Description	Reset Value
2:0	Status	These bits are always 0	0
7:3	Status	Status bits	1

- 26 code states
  - 0xF8 – no relevant information (SI = 0)

http://www.semiconductors.philips.com/acrobat/various/8XC552_562OVERVIEW_2.pdf

I2C – Duty cycle register

- Bit Frequency = PCLK / (I2SCLH + I2SCLL)

I2SCLH	Function	Description	Reset Value
15:0	Count	Count for SCL HIGH time period selection	0x 0004

I2SCLL	Function	Description	Reset Value
15:0	Count	Count for SCL LOW time period selection	0x 0004

I2C – Sample code

```
void i2c_init() {
IOSET = SDA | SCL;
IODIR |= IODIR | SDA | SCL;
}
void write_bit(unsigned char d) {
        if (d & 1) IOSET = SDA;
        else IOCLR = SDA;
        delay(CLK_LOW); IOSET = SCL;
        delay(CLK_HIGH); IOCLR = SCL;
}
void i2c_master_ack() { write_bit(0); }
void i2c_master_nack() { write_bit(1); }
```

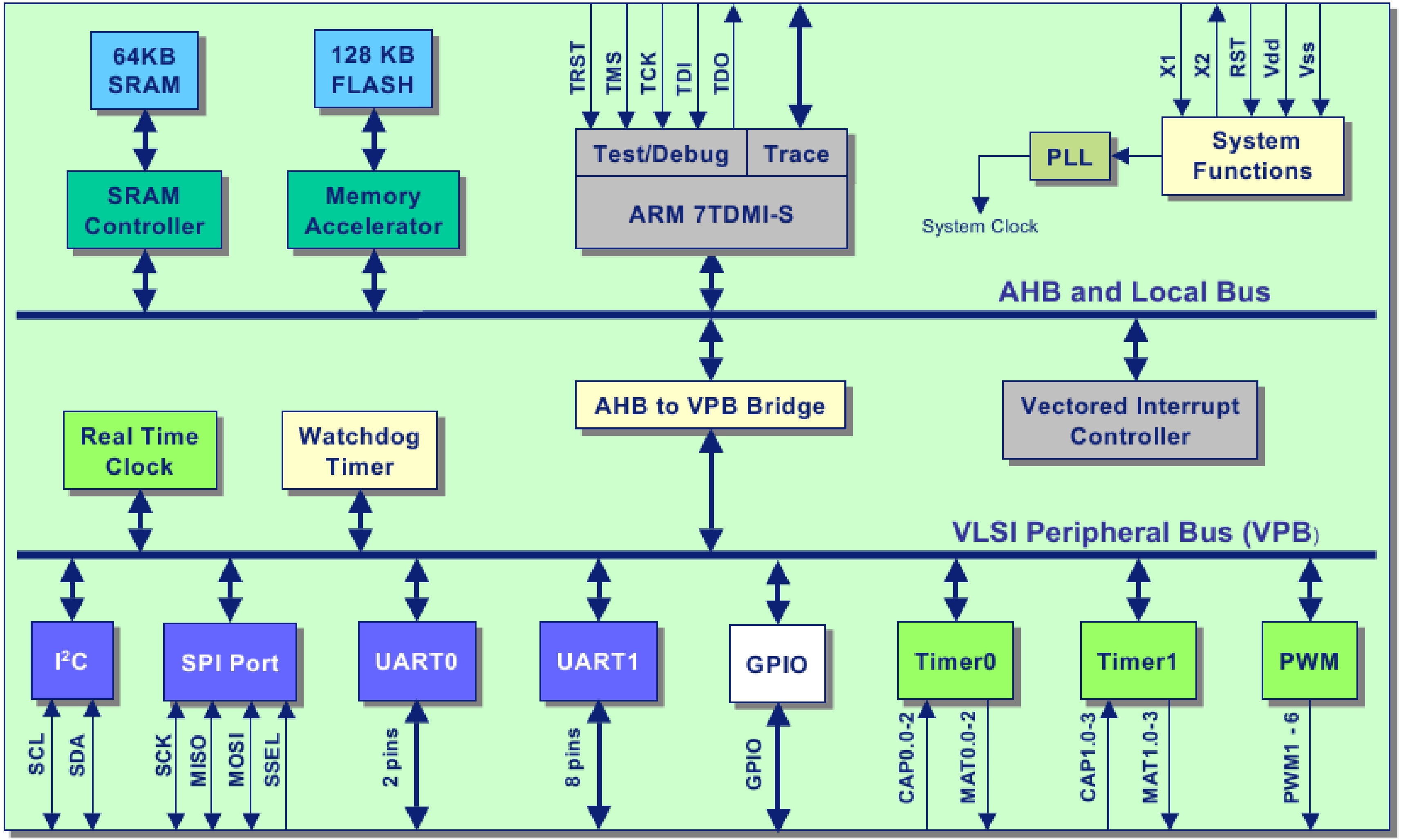
```
void i2c_start() {
        IOSET = SDA | SCL;
        delay(CLK_START_SETUP);
        IOCLR = SDA;
        delay(CLK_START_HOLD);
        IOCLR = SCL;
}
void i2c_stop() {
        IOCLR = SDA | SCL;
        delay(CLK_DELAY);
        IOSET = SCL;
        delay(CLK_DELAY);
}
```

```
unsigned char read_bit() {
	int val;
	IOSET = SDA;
	delay(CLK_LOW);
	IOSET = SCL;
	delay(CLK_HIGH);
	val = IOPIN;
	IOCLR = SCL;
	return (val & SDA);
}
```

```
void i2c_write_byte(unsigned char data) {
	int i;
	for (i = 0; i < 8; ++i)
		write_bit(data >> (7 - i));
}

unsigned char i2c_read_byte() {
	unsigned char val = 0;
	int i;
	for (i = 0; i < 8; ++i)
		val = (val << 1) + read_bit();
	return val;
}
```

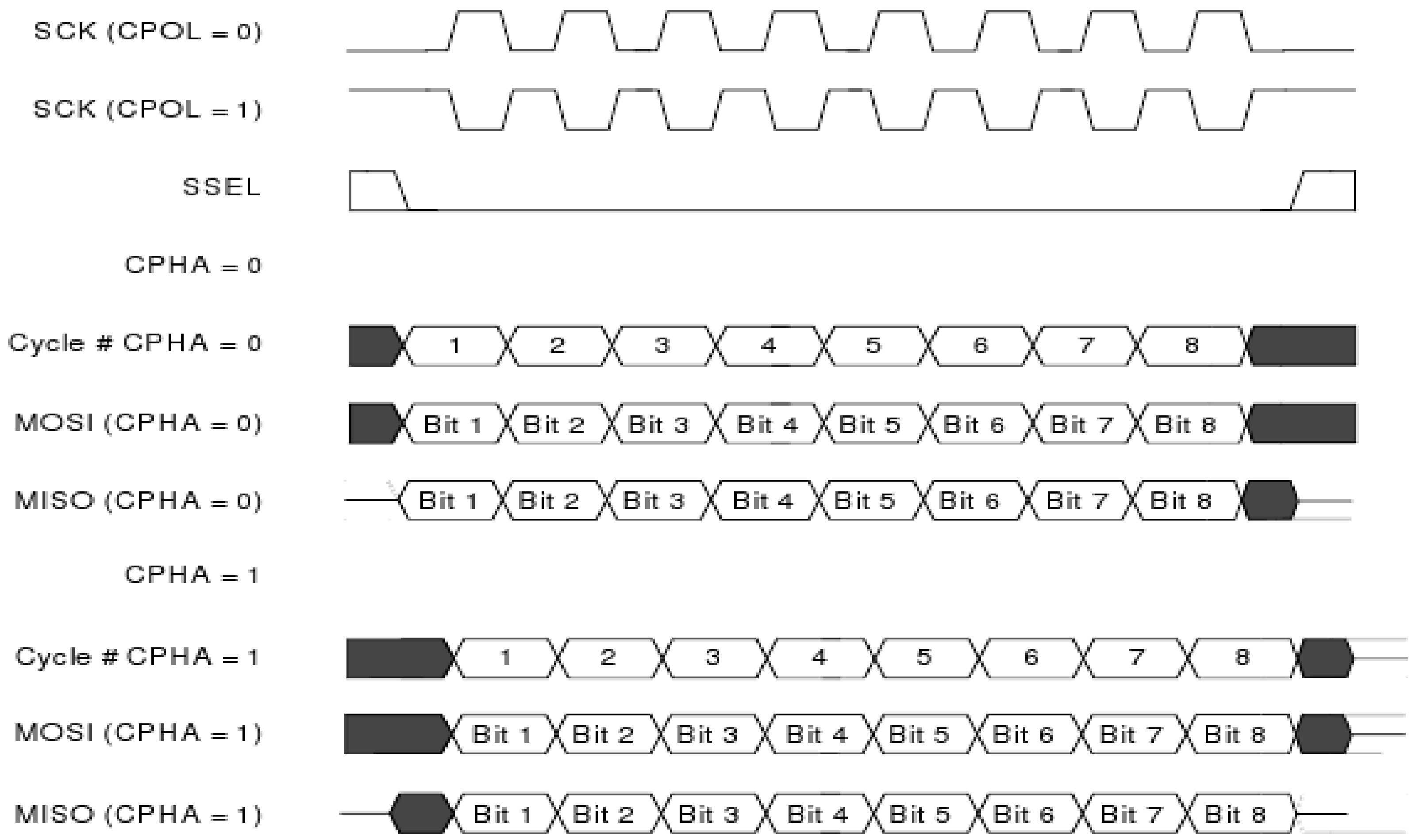
Block diagram



SPI :: Features

- Compliant with Serial Peripheral Interface (SPI) specification.
- Synchronous, Serial, Full Duplex Communication.
- Combined SPI master and slave.
- Maximum data bit rate of one eighth of the input clock rate.

SPI :: Description



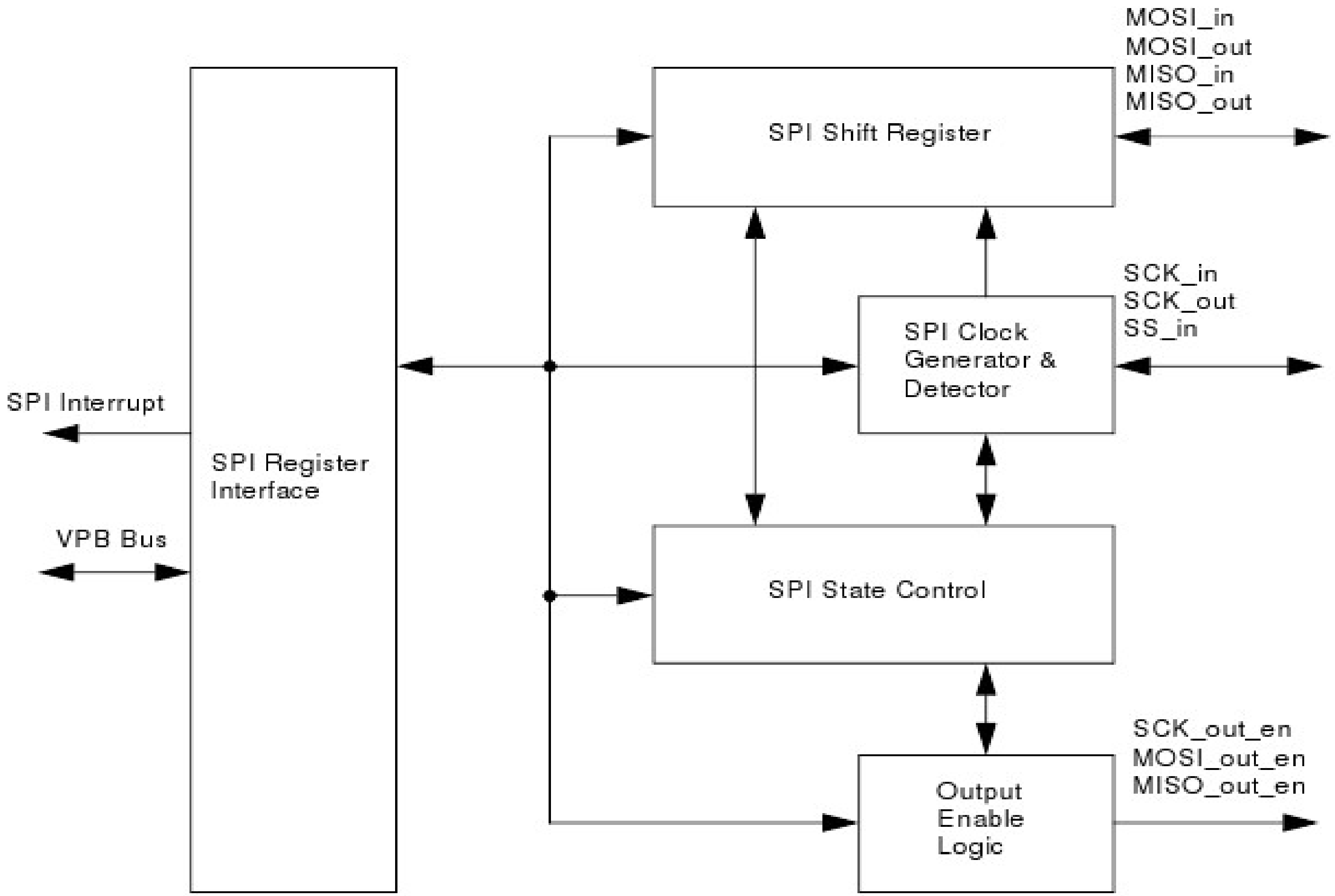
SPI :: Data Transfer

CPOL And CPHA Settings	First Data Driven	Other Data Driven	Data Sampled
CPOL = 0, CPHA = 0	Prior to first SCK rising edge	SCK falling edge	SCK rising edge
CPOL = 0, CPHA = 1	First SCK rising edge	SCK rising edge	SCK falling edge
CPOL = 1, CPHA = 0	Prior to first SCK falling edge	SCK rising edge	SCK falling edge
CPOL = 1, CPHA = 1	First SCK falling edge	SCK falling edge	SCK rising edge

SPI :: Interface

Pin Name	Type	Pin Description
SCK	Input/ Output	**Serial Clock.** The SPI is a clock signal used to synchronize the transfer of data across the SPI interface. The SPI is always driven by the master and received by the slave. The clock is programmable to be active high or active low. The SPI is only active during a data transfer. Any other time, it is either in its inactive state, or tri-stated.
SSEL	Input	**Slave Select.** The SPI slave select signal is an active low signal that indicates which slave is currently selected to participate in a data transfer. Each slave has its own unique slave select signal input. The SSEL must be low before data transactions begin and normally stays low for the duration of the transaction. If the SSEL signal goes high any time during a data transfer, the transfer is considered to be aborted. In this event, the slave returns to idle, and any data that was received is thrown away. There are no other indications of this exception. This signal is not directly driven by the master. It could be driven by a simple general purpose I/O under software control.
MISO	Input/ Output	**Master In Slave Out.** The MISO signal is a unidirectional signal used to transfer serial data from the slave to the master. When a device is a slave, serial data is output on this signal. When a device is a master, serial data is input on this signal. When a slave device is not selected, the slave drives the signal high impedance.
MOSI	Input/ Output	**Master Out Slave In.** The MOSI signal is a unidirectional signal used to transfer serial data from the master to the slave. When a device is a master, serial data is output on this signal. When a device is a slave, serial data is input on this signal.

SPI :: Architecture



SPI :: Registers

Address	Name	Description	Access	Reset Value*
0xE0020000	SPCR	SPI Control Register. This register controls the operation of the SPI.	Read/Write	0
0xE0020004	SPSR	SPI Status Register. This register shows the status of the SPI.	Read Only	0
0xE0020008	SPDR	SPI Data Register. This bi-directional register provides the transmit and receive data for the SPI. Transmit data is provided to the SPI by writing to this register. Data received by the SPI can be read from this register.	Read/Write	0
0xE002000C	SPCCR	SPI Clock Counter Register. This register controls the frequency of a master's SCK.	Read/Write	0
0xE002001C	SPINT	SPI Interrupt Flag. This register contains the interrupt flag for the SPI interface.	Read/Write	0

SPI :: Control Register

SPCR	Function	Description	Reset Value
2:0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	CPHA	Clock phase control determines the relationship between the data and the clock on SPI transfers, and controls when a slave transfer is defined as starting and ending. When 1, data is sampled on the second clock edge of the SCK. A transfer starts with the first clock edge, and ends with the last sampling edge when the SSEL signal is active. When 0, data is sampled on the first clock edge of SCK. A transfer starts and ends with activation and deactivation of the SSEL signal.	0
4	CPOL	Clock polarity control. When 1, SCK is active low. When 0, SCK is active high.	0
5	MSTR	Master mode select. When 1, the SPI operates in Master mode. When 0, the SPI operates in Slave mode.	0
6	LSBF	LSB First controls which direction each byte is shifted when transferred. When 1, SPI data is transferred LSB (bit 0) first. When 0, SPI data is transferred MSB (bit 7) first.	0
7	SPIE	Serial peripheral interrupt enable. When 1, a hardware interrupt is generated each time the SPIF or MODF bits are activated. When 0, SPI interrupts are inhibited.	0

SPI :: Status Register

SPSR	Function	Description	Reset Value
2:0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	ABRT	Slave abort. When 1, this bit indicates that a slave abort has occurred. This bit is cleared by reading this register.	0
4	MODF	Mode fault. when 1, this bit indicates that a Mode fault error has occurred. This bit is cleared by reading this register, then writing the SPI control register.	0
5	ROVR	Read overrun. When 1, this bit indicates that a read overrun has occurred. This bit is cleared by reading this register.	0
6	WCOL	Write collision. When 1, this bit indicates that a write collision has occurred. This bit is cleared by reading this register, then accessing the SPI data register.	0
7	SPIF	SPI transfer complete flag. When 1, this bit indicates when a SPI data transfer is complete. When a master, this bit is set at the end of the last cycle of the transfer. When a slave, this bit is set on the last data sampling edge of the SCK. This bit is cleared by first reading this register, then accessing the SPI data register. **Note:** this is not the SPI interrupt flag. This flag is found in the SPINT registrer.	0

SPI :: Clock Register

SPCCR	Function	Description	Reset Value
7:0	Counter	SPI Clock counter setting	0

- Number of PCLK cycles that make up an SPI clock.
- Value must always be an even number
- Value must always be greater than or equal to 8

SPI :: Sample Code

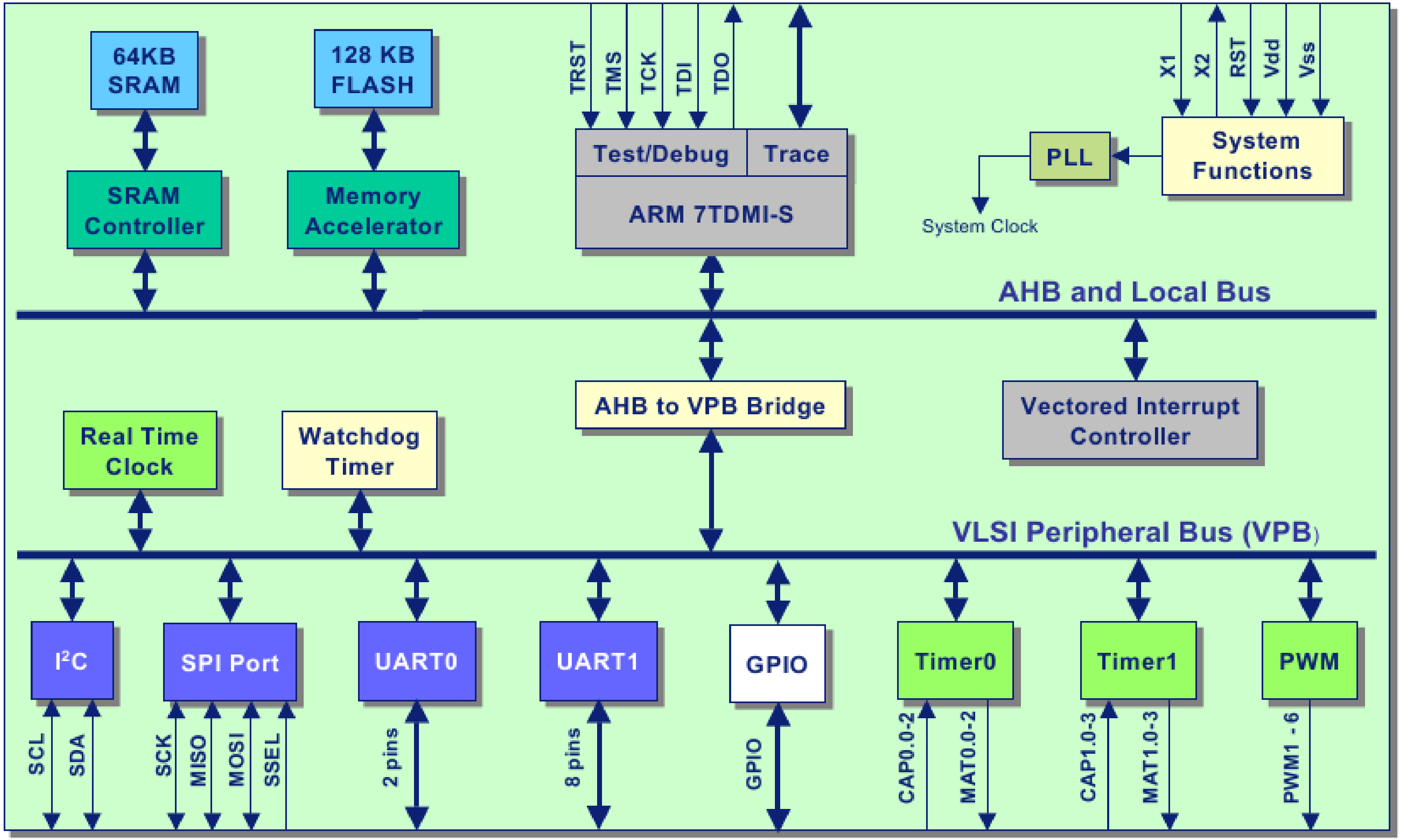
```
void SPIInit(void)
{
    unsigned char i;

    PINSEL0 = (1 << 14) | (1 << 12) | (1 << 10) | (1 << 8);  // set pins to SPI
    S0SPCCR = 8;                    // configure spi clock
    S0SPCR  = S0SPCR_MSTR;          // master mode, CPOL = 0, CCPHA = 0,  MSB first
    i = S0SPSR;                     // read SPI status reg to clear the flags.
}

int SPITransfer(char* txBuffer, char *rxBuffer, int len)
{
    int i;

    for (i = 0; i < len; i++) {
      S0SPDR = *txBuffer++;      // load spi tx reg
      while (!(S0SPSR));          // wait for transmission to complete
      *rxBuffer++ = S0SPDR;      // read data from SPI data reg
    }
    return i;
}
```

Block diagram

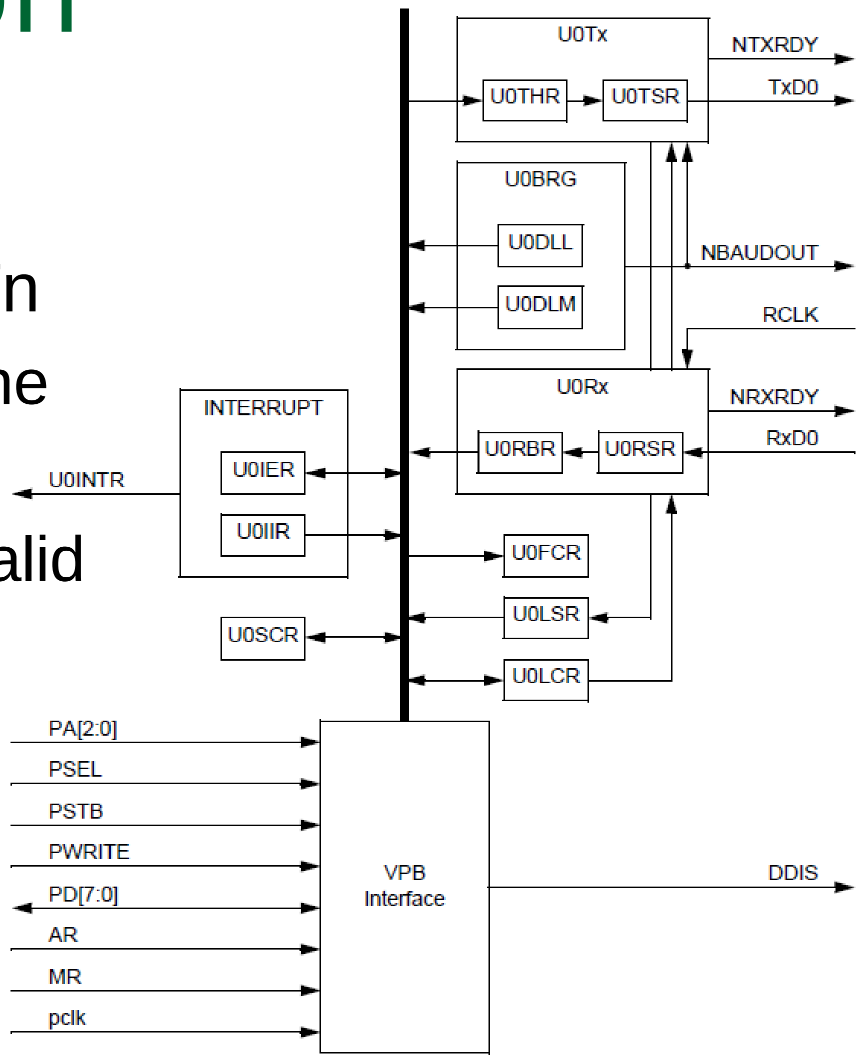


UART :: Features

- 16 byte Receive and Transmit FIFOs.
- Register locations conform to ‘550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- Built-in baud rate generator.
- UART1 - Standard modem interface signals

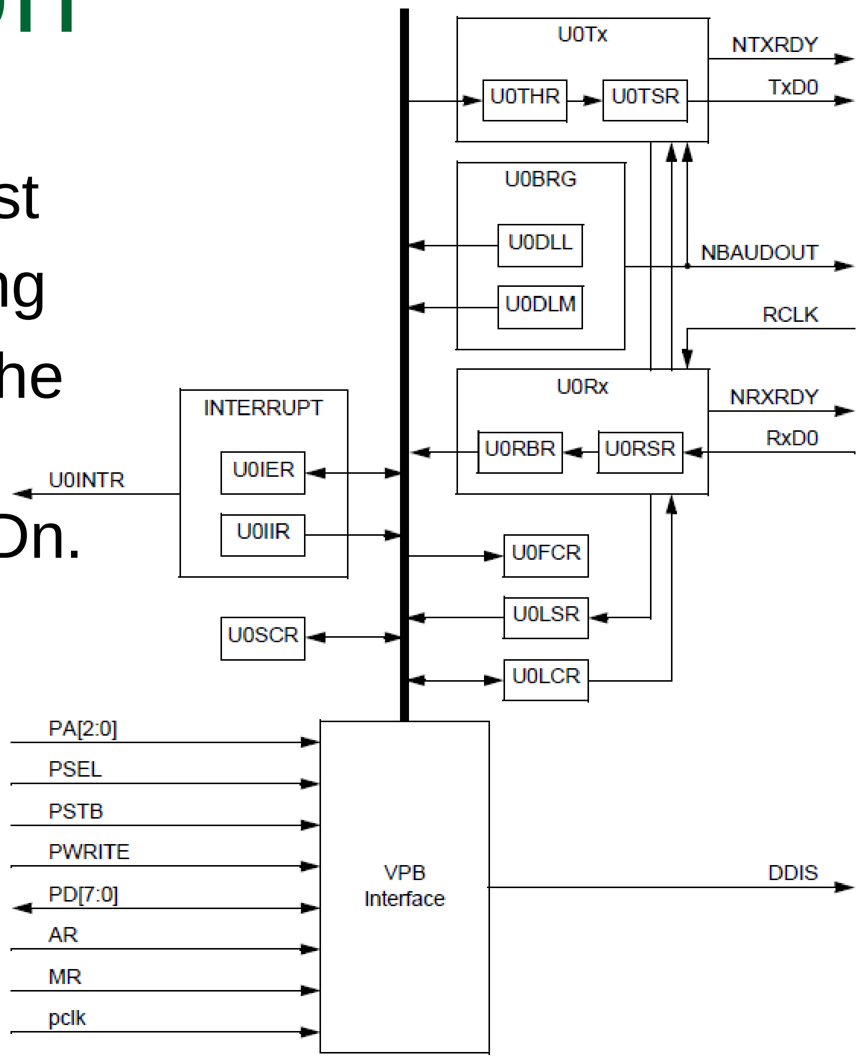
UART :: Description

- VPB interface
  - link between host and UARTn
- UnRx, monitors the serial input line
  - RxDn, for valid input.
  - The UARTn UnRSR accepts valid characters via RxDn.
  - Valid character is passed to UARTn Rx Buffer
- FIFO to await access by the host via the generic host interface.



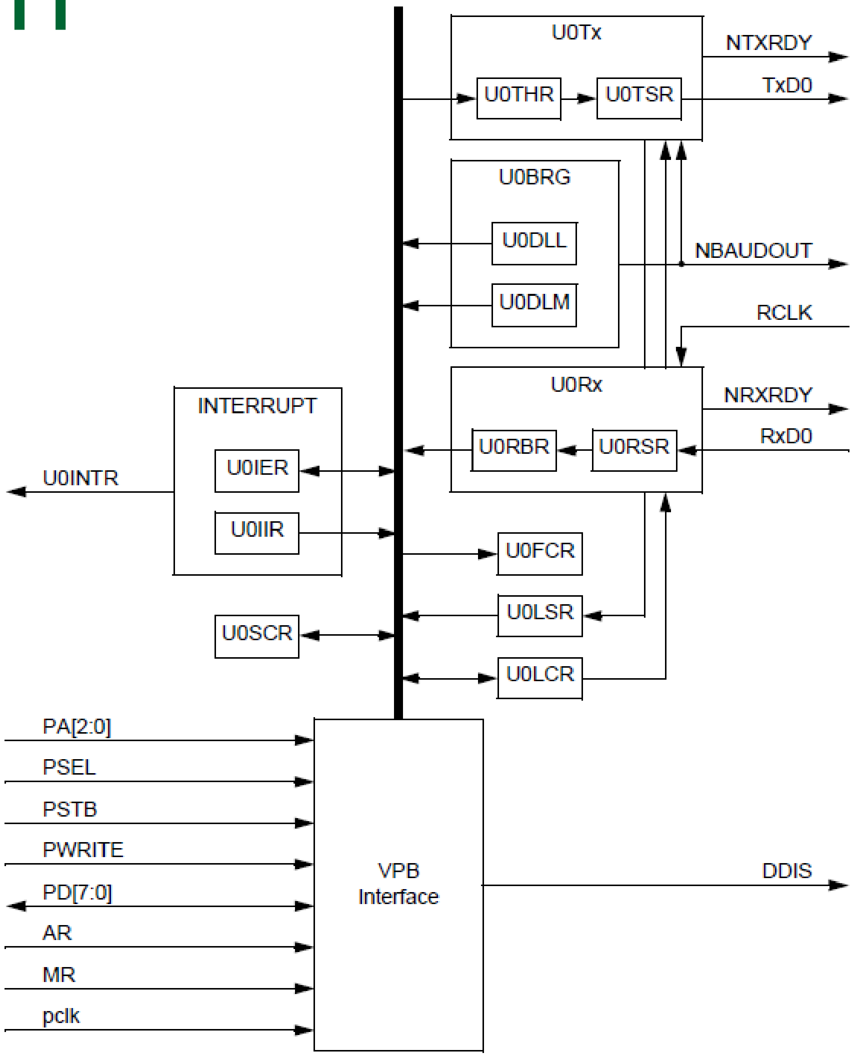
UART :: Description

- UnTx, accepts data written by host
  - buffers in the UARTn Tx Holding
- UnTSR reads the data stored in the UnTHR
  - transmit via the output pin, TxDn.



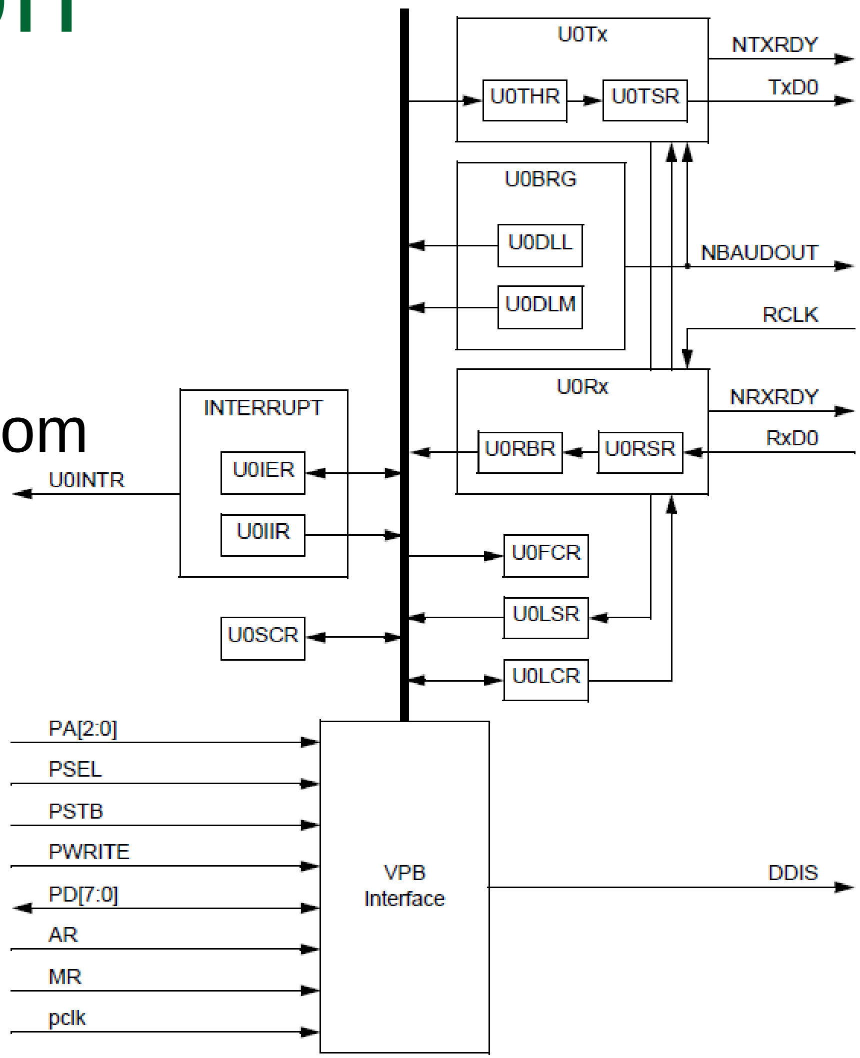
UART :: Description

- UnBRG generates the timing
  - Clock input source is the VPB clock - PCLK.
  - Clock is divided
    - UnDLL and UnDLM regis
    - 16x oversample clock, NBAUDOUT.



UART :: Description

- The interrupt interface contains registers UnIER and UnIIR.
- The interrupt interface receives several one clock wide enables from the UnTx and UnRx blocks.
- UnLSR, status information from UnTx and UnRx
- UnLCR, control information for UnTx and UnRx



UART :: Pin Description

Pin Name	Type	Description
RxD0	Input	**Serial Input**. Serial receive data.
TxD0	Output	**Serial Output**. Serial transmit data.

UART :: Registers

Address Offset	Name	Description	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Access	Reset Value*
0xE000C000 DLAB = 0	U0RBR	Receiver Buffer Register	MSB			READ DATA				LSB	RO	un-defined
0xE000C000 DLAB = 0	U0THR	Transmit Holding Register	MSB			WRITE DATA				LSB	WO	NA
0xE000C004 DLAB = 0	U0IER	Interrupt Enable Register	0	0	0	0	0	Enable Rx Line Status Interrupt	Enable THRE Interrupt	Enable Rx Data Available Interrupt	R/W	0
0xE000C008	U0IIR	Interrupt ID Register	FIFOs Enabled		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01
0xE000C008	U0FCR	FIFO Control Register	Rx Trigger		Reserved		-	Tx FIFO Reset	Rx FIFO Reset	FIFO Enable	WO	0
0xE000C00C	U0LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Number of Stop Bits	Word Length Select		R/W	0
0xE000C014	U0LSR	Line Status Register	Rx FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60
0xE000C01C	U0SCR	Scratch Pad Register	MSB							LSB	R/W	0
0xE000C000 DLAB = 1	U0DLL	Divisor Latch LSB	MSB							LSB	R/W	0x01
0xE000C004 DLAB = 1	U0DLM	Divisor Latch MSB	MSB							LSB	R/W	0

UART :: Line Control Register

U0LCR	Function	Description	Reset Value
1:0	Word Length Select	00: 5 bit character length 01: 6 bit character length 10: 7 bit character length 11: 8 bit character length	0
2	Stop Bit Select	0: 1 stop bit 1: 2 stop bits (1.5 if U0LCR[1:0]=00)	0
3	Parity Enable	0: Disable parity generation and checking 1: Enable parity generation and checking	0
5:4	Parity Select	00: Odd parity 01: Even parity 10: Forced "1" stick parity 11: Forced "0" stick parity	0
6	Break Control	0: Disable break transmission 1: Enable break transmission. Output pin UART0 TxD is forced to logic 0 when U0LCR6 is active high.	0
7	Divisor Latch Access Bit	0: Disable access to Divisor Latches 1: Enable access to Divisor Latches	0

UART :: Divisor Register

U0DLL	Function	Description	Reset Value
7:0	Divisor Latch LSB Register	The UART0 Divisor Latch LSB Register, along with the U0DLM register, determines the baud rate of the UART0.	0x01

U0DLM	Function	Description	Reset Value
7:0	Divisor Latch MSB Register	The UART0 Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UART0.	0

UART :: Line Status Register

U0LSR	Function	Description	Reset Value
0	Receiver Data Ready (RDR)	0: U0RBR is empty 1: U0RBR contains valid data U0LSR0 is set when the U0RBR holds an unread character and is cleared when the UART0 RBR FIFO is empty.	0
1	Overrun Error (OE)	0: Overrun error status is inactive. 1: Overrun error status is active. The overrun error condition is set as soon as it occurs. An U0LSR read clears U0LSR1. U0LSR1 is set when UART0 RSR has a new character assembled and the UART0 RBR FIFO is full. In this case, the UART0 RBR FIFO will not be overwritten and the character in the UART0 RSR will be lost.	0
2	Parity Error (PE)	0: Parity error status is inactive. 1: Parity error status is active. When the parity bit of a received character is in the wrong state, a parity error occurs. An U0LSR read clears U0LSR2. Time of parity error detection is dependent on U0FCR0. A parity error is associated with the character being read from the UART0 RBR FIFO.	0
3	Framing Error (FE)	0: Framing error status is inactive. 1: Framing error status is active. When the stop bit of a received character is a logic 0, a framing error occurs. An U0LSR read clears U0LSR3. The time of the framing error detection is dependent on U0FCR0. A framing error is associated with the character being read from the UART0 RBR FIFO. Upon detection of a framing error, the Rx will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error.	0

UART :: Line Status Register

4	Break Interrupt (BI)	0: Break interrupt status is inactive. 1: Break interrupt status is active. When RxD0 is held in the spacing state (all 0's) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RxD0 goes to marking state (all 1's). An U0LSR read clears this status bit. The time of break detection is dependent on U0FCR0. The break interrupt is associated with the character being read from the UART0 RBR FIFO.	0
5	Transmitter Holding Register Empty (THRE)	0: U0THR contains valid data. 1: U0THR is empty. THRE is set immediately upon detection of an empty UART0 THR and is cleared on a U0THR write.	1
6	Transmitter Empty (TEMT)	0: U0THR and/or the U0TSR contains valid data. 1: U0THR and the U0TSR are empty. TEMT is set when both U0THR and U0TSR are empty; TEMT is cleared when either the U0TSR or the U0THR contain valid data.	1
7	Error in Rx FIFO (RXFE)	0: U0RBR contains no UART0 Rx errors or U0FCR0=0. 1: UART0 RBR contains at least one UART0 Rx error. U0LSR7 is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the U0RBR. This bit is cleared when the U0LSR register is read and there are no subsequent errors in the UART0 FIFO.	0

UART :: Interrupt Enable Register

U0IER	Function	Description	Reset Value
0	RBR Interrupt Enable	0: Disable the RDA interrupt. 1: Enable the RDA interrupt. U0IER0 enables the Receive Data Available interrupt for UART0. It also controls the Character Receive Time-out interrupt.	0
1	THRE Interrupt Enable	0: Disable the THRE interrupt. 1: Enable the THRE interrupt. U0IER1 enables the THRE interrupt for UART0. The status of this interrupt can be read from U0LSR5.	0
2	Rx Line Status Interrupt Enable	0: Disable the Rx line status interrupts. 1: Enable the Rx line status interrupts. U0IER2 enables the UART0 Rx line status interrupts. The status of this interrupt can be read from U0LSR[4:1].	0
7:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

UART :: Interrupt Ident. Register

U0IIR	Function	Description	Reset Value
0	Interrupt Pending	0: At least one interrupt is pending. 1: No pending interrupts. Note that U0IIR0 is active low. The pending interrupt can be determined by evaluating U0IER3:1.	1
3:1	Interrupt Identification	011: 1. Receive Line Status (RLS) 010: 2a.Receive Data Available (RDA) 110: 2b.Character Time-out Indicator (CTI) 001: 3. THRE Interrupt. U0IER3 identifies an interrupt corresponding to the UART0 Rx FIFO. All other combinations of U0IER3:1 not listed above are reserved (000,100,101,111).	0
5:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFO Enable	These bits are equivalent to U0FCR0.	0

UART :: FIFO Control Register

U0FCR	Function	Description	Reset Value
0	FIFO Enable	Active high enable for both UART0 Rx and Tx FIFOs and U0FCR7:1 access. This bit must be set for proper UART0 opearation. Any transition on this bit will automatically clear the UART0 FIFOs.	0
1	Rx FIFO Reset	Writing a logic 1 to U0FCR1 will clear all bytes in UART0 Rx FIFO and reset the pointer logic. This bit is self-clearing.	0
2	Tx FIFO Reset	Writing a logic 1 to U0FCR2 will clear all bytes in UART0 Tx FIFO and reset the pointer logic. This bit is self-clearing.	0
5:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	Rx Trigger Level Select	00: trigger level 0 (default='h1) 01: trigger level 1 (default='h4) 10: trigger level 2 (default='h8) 11: trigger level 3 (default='he) These two bits determine how many receiver UART0 FIFO characters must be written before an interrupt is activated. The four trigger levels are defined by the user at compilation allowing the user to tune the trigger levels to the FIFO depths chosen.	0

UART :: Interrupt Handling

U0IIR[3:0]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset
0001	-	none	none	-
0110	Highest	Rx Line Status / Error	OE or PE or FE or BI	U0LSR Read
0100	Second	Rx Data Available	Rx data available or trigger level reached in FIFO (U0FCR0=1)	U0RBR Read or UART0 FIFO drops below trigger level
1100	Second	Character Time-out Indication	Minimum of one character in the Rx FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) X 7 - 2] X 8 + {(trigger level - number of characters) X 8 + 1] RCLKs	U0 RBR Read
0010	Third	THRE	THRE	U0IIR Read (if source of interrupt) or THR write

note: values "0000", “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.

UART :: Sample Code

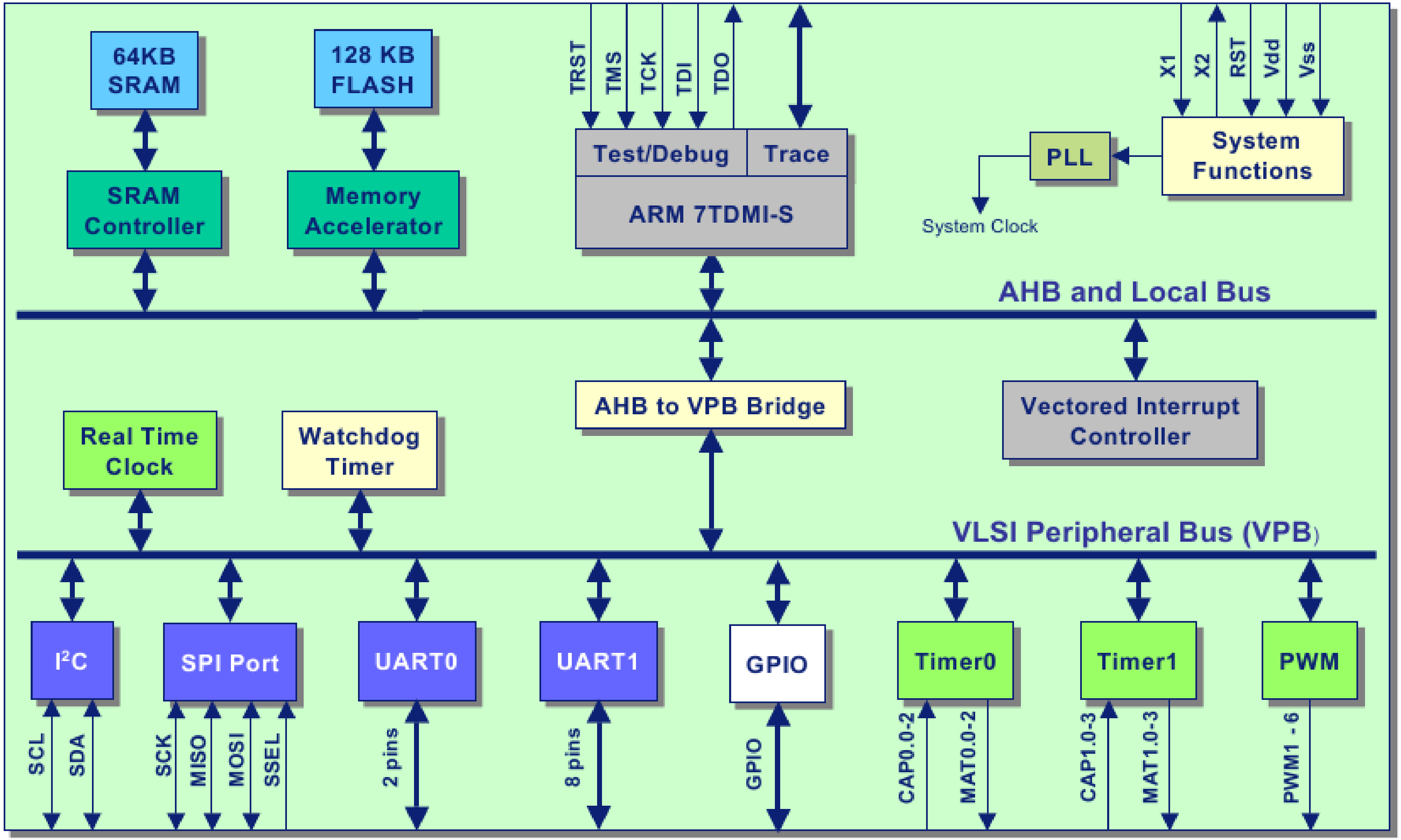
```
void UART0Init(unsigned int baud)
{
  unsigned int divisor = PCLK / (16 * baud);

  PINSEL0 = (PINSEL0 & ~0xF) | (1 << 2) | (1 << 0);    //set pins TX0 and RXD0
  UART0_LCR = 0x80;                             // enable DLAB
  UART0_DLL = divisor & 0xFF;                   //divisor
  UART0_DLM = (divisor >> 8) & 0xFF;
  UART0_LCR = 0x03;  // 8 bit, 1 stop bit, no parity, disable DLAB
}

void UART0WriteChar(unsigned char ch)
{
  while ((UART0_LSR & 0x20) == 0); // wait for THRE
  UART0_THR = ch;
}

unsigned char UART0ReadChar(void)
{
  while ((UART0_LSR & 0x01) == 0);   // wait for DR
  return  UART0_RBR;
}
```

Block diagram



Watchdog :: Features

- Internally resets chip if not periodically reloaded
- Debug mode
- Enabled by software
  - hardware reset
- Incorrect/Incomplete feed sequence
  - reset/interrupt
- Flag to indicate Watchdog reset
- Programmable 32-bit timer with internal pre-scaler
- Selectable time period

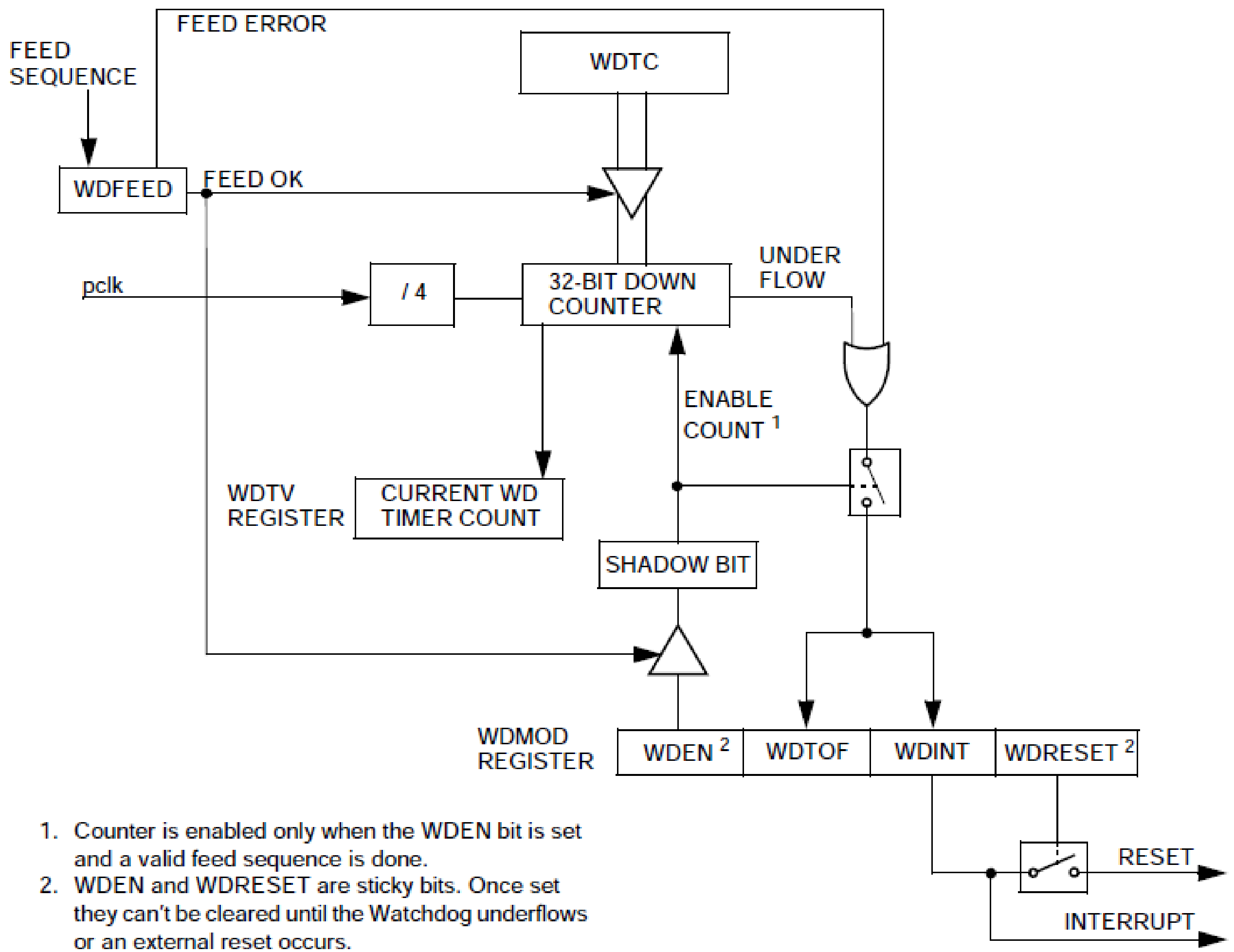
Watchdog :: Description

- Consists:
  - a divide by 4 fixed pre-scaler
  - a 32-bit counter.
  - The clock is fed to the timer via a pre-scaler.
- The timer decrements when clocked
  - The minimum value from which the counter decrements is 0xFF.
  - Setting a value lower than 0xFF causes 0xFF to be loaded in the counter.
    - Minimum interval is (tpclk x 256 x 4)
    - Maximum interval is (tpclk x 232 x 4) in multiples of (tpclk x 4)

Watchdog :: Usage

- Set the Watchdog timer constant reload
- Setup mode
- Start the Watchdog
  - writing 0xAA, 0x55 to the WDFEED register.
- Watchdog should be fed again before the counter underflows
- When the Watchdog counter underflows
  - program counter will start from 0x00000000

Watchdog :: Architecture



1. Counter is enabled only when the WDEN bit is set and a valid feed sequence is done.
2. WDEN and WDRESET are sticky bits. Once set they can't be cleared until the Watchdog underflows or an external reset occurs.

Watchdog :: Registers

Address	Name	Description	Access	Reset Value*
0xE0000000	WDMOD	Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer.	Read/Set	0
0xE0000004	WDTC	Watchdog timer constant register. This register determines the time-out value.	Read/Write	0xFF
0xE0000008	WDFEED	Watchdog feed sequence register. Writing AAh followed by 55h to this register reloads the Watchdog timer to its preset value.	Write Only	NA
0xE000000C	WDTV	Watchdog timer value register. This register reads out the current value of the Watchdog timer.	Read Only	0xFF

Watchdog :: WDMOD

WDMOD	Function	Description	Reset Value
0	WDEN	Watchdog interrupt enable bit (Set only)	0
1	WDRESET	Watchdog reset enable bit (Set Only)	0
2	WDTOF	Watchdog time-out flag	0 (Only after external reset)
3	WDINT	Watchdog interrupt flag (Read Only)	0
7:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

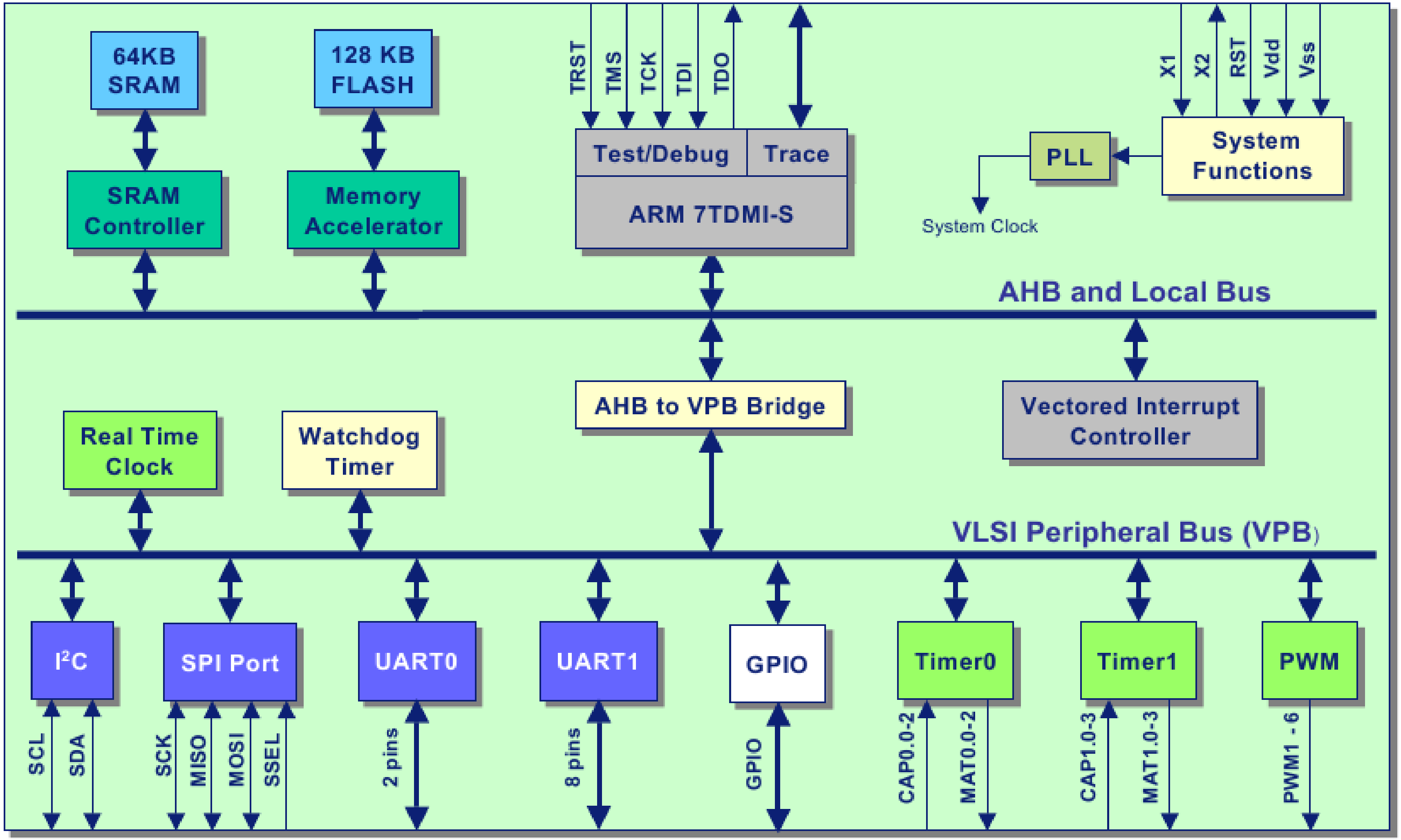
Watchdog :: Sample Code

```
void WDTInit(void) {
	WDTC = WDT_FEED_VALUE;
	WDMOD = WDEN | WDRESET;
	WDFEED = 0xAA;
	WDFEED = 0x55;
}
```

```
void WDTFeed(void) {
	WDFEED = 0xAA;
	WDFEED = 0x55;
}
```

```
void WDTHandler(void) {
	WDMOD &= ~WDTOF;
	wdt_counter++;
}
```

Block diagram



ARM - Excepções

- Tipos (8) – exemplos
  - Interrupções
  - Instrução invalida
  - Reset, etc.
- Estado do processador tem de ser preservado
- Varias excepções em simultâneo

Processamento

- Execução em endereço fixo em memoria
  - Vectores de excepção

Processamento

- Modo de execução muda de acordo com a excepção
  - Codificado em CPSR[0..4] o modo de execução
  - Registos alternativos
    - O banco de registos é do modo em execução

31	30	29	28	27	26 8	7	6	5	4	3	2	1	0
N	Z	C	V	Q	DNM(RAZ)	I	F	T	M4	M3	M2	M1	M0

Excepções / Modos de Execução

Endereço	Excepção	Modo
0x00000000	Reset	Supervisor
0x00000004	Inst. Indef.	Não def.
0x00000008	Intr. Soft	Supervisor
0x0000000C	Abort(prefetch)	Abort
0x00000010	Abort(data)	Abort
0x00000014	Reservado	
0x00000018	IRQ	IRQ
0x0000001C	FIQ	FIQ

Processamento: chamada

- Execução
  - Guardar no R14 (LR) do modo de excepção o PC para retorno
  - Guardar no SPSR do modo de excepção o CPSR
  - No CPSR afectar
    - Os bits do modo de excepção
    - Inibir IRQ
    - Inibir FIQ (se excepção FIQ)
  - Afectar o PC com o vector de excepção

Processamento: retorno

- Retorno
  - Repor o CPSR
  - Afectar o PC com o valor de R14 do modo de excepção
    - Estas acções devem ser indivisível

Exemplos de Código de Retorno

- Utilizando o sufixo 'S'

```
...
subs PC, LR, #4
```

- Utilizando o oper. '^'

```
sub   LR, LR, #4
stmfd SP!, {regs, LR}
... processamento
ldmfd SP!, {regs, PC}^
```

Registos / Modo de Execução

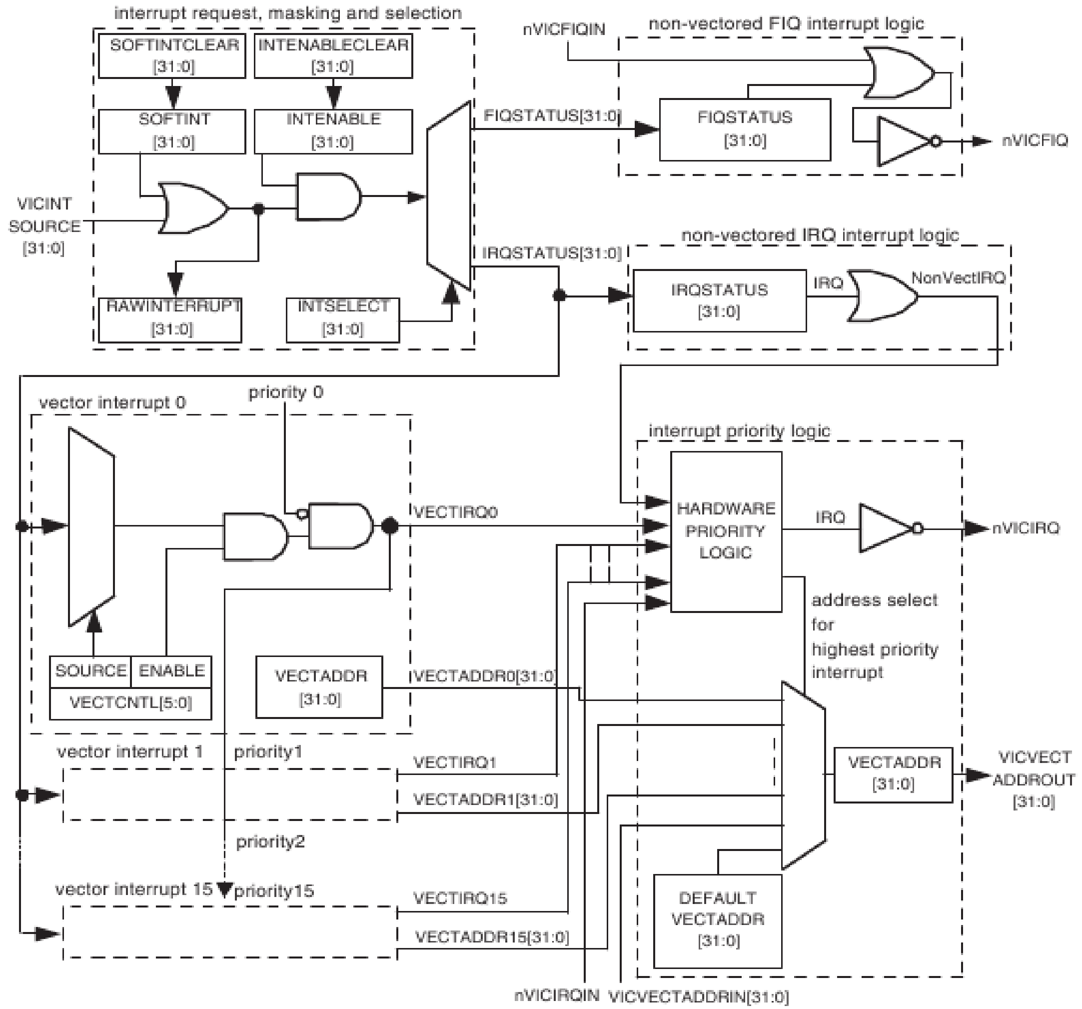
System & User	FIQ	Supervisor	Abort	IRQ	Undefined
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8_fiq	R8	R8	R8	R8
R9	R9_fiq	R9	R9	R9	R9
R10	R10_fiq	R10	R10	R10	R10
R11	R11_fiq	R11	R11	R11	R11
R12	R12_fiq	R12	R12	R12	R12
R13	R13_fiq	R13_svc	R13_abt	R13_irq	R13_und
R14	R14_fiq	R14_svc	R14_abt	R14_irq	R14_und
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)

ARM State Program Status Registers

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

= banked register

Vectored Interrupt Control



VIC :: Registers Addresses

Address	Name	Description	Access	Reset Value*
0xFFFF F000	VICIRQStatus	IRQ Status Register. This register reads out the state of those interrupt requests that are enabled and classified as IRQ.	RO	0
0xFFFF F004	VICFIQStatus	FIQ Status Requests. This register reads out the state of those interrupt requests that are enabled and classified as FIQ.	RO	0
0xFFFF F008	VICRawIntr	Raw Interrupt Status Register. This register reads out the state of the 32 interrupt requests / software interrupts, regardless of enabling or classification.	RO	0
0xFFFF F00C	VICIntSelect	Interrupt Select Register. This register classifies each of the 32 interrupt requests as contributing to FIQ or IRQ.	R/W	0
0xFFFF F010	VICIntEnable	Interrupt Enable Register. This register controls which of the 32 interrupt requests and software interrupts are enabled to contribute to FIQ or IRQ.	R/W	0
0xFFFF F014	VICIntEnClr	Interrupt Enable Clear Register. This register allows software to clear one or more bits in the Interrupt Enable register.	W	0
0xFFFF F018	VICSoftInt	Software Interrupt Register. The contents of this register are ORed with the 32 interrupt requests from various peripheral functions.	R/W	0
0xFFFF F01C	VICSoftIntClear	Software Interrupt Clear Register. This register allows software to clear one or more bits in the Software Interrupt register.	W	0
0xFFFF F020	VICProtection	Protection enable register. This register allows limiting access to the VIC registers by software running in privileged mode.	R/W	0
0xFFFF F030	VICVectAddr	Vector Address Register. When an IRQ interrupt occurs, the IRQ service routine can read this register and jump to the value read.	R/W	0
0xFFFF F034	VICDefVectAddr	Default Vector Address Register. This register holds the address of the Interrupt Service routine (ISR) for non-vectored IRQs.	R/W	0

VIC :: Registers

Address	Name	Description	Access	Reset Value*
0xFFFF F100	VICVectAddr0	Vector address 0 register. Vector Address Registers 0-15 hold the addresses of the Interrupt Service routines (ISRs) for the 16 vectored IRQ slots.	R/W	0
0xFFFF F104	VICVectAddr1	Vector address 1 register	R/W	0
0xFFFF F108	VICVectAddr2	Vector address 2 register	R/W	0
0xFFFF F10C	VICVectAddr3	Vector address 3 register	R/W	0
0xFFFF F110	VICVectAddr4	Vector address 4 register	R/W	0
0xFFFF F114	VICVectAddr5	Vector address 5 register	R/W	0
0xFFFF F118	VICVectAddr6	Vector address 6 register	R/W	0
0xFFFF F11C	VICVectAddr7	Vector address 7 register	R/W	0
0xFFFF F120	VICVectAddr8	Vector address 8 register	R/W	0
0xFFFF F124	VICVectAddr9	Vector address 9 register	R/W	0
0xFFFF F128	VICVectAddr10	Vector address 10 register	R/W	0
0xFFFF F12C	VICVectAddr11	Vector address 11 register	R/W	0
0xFFFF F130	VICVectAddr12	Vector address 12 register	R/W	0
0xFFFF F134	VICVectAddr13	Vector address 13 register	R/W	0
0xFFFF F138	VICVectAddr14	Vector address 14 register	R/W	0
0xFFFF F13C	VICVectAddr15	Vector address 15 register	R/W	0

VIC :: Registers Addresses

Address	Name	Description	Access	Reset Value*
0xFFFF F200	VICVectCntl0	Vector control 0 register. Vector Control Registers 0-15 each control one of the 16 vectored IRQ slots. Slot 0 has the highest priority and slot 15 the lowest.	R/W	0
0xFFFF F204	VICVectCntl1	Vector control 1 register	R/W	0
0xFFFF F208	VICVectCntl2	Vector control 2 register	R/W	0
0xFFFF F20C	VICVectCntl3	Vector control 3 register	R/W	0
0xFFFF F210	VICVectCntl4	Vector control 4 register	R/W	0
0xFFFF F214	VICVectCntl5	Vector control 5 register	R/W	0
0xFFFF F218	VICVectCntl6	Vector control 6 register	R/W	0
0xFFFF F21C	VICVectCntl7	Vector control 7 register	R/W	0
0xFFFF F220	VICVectCntl8	Vector control 8 register	R/W	0
0xFFFF F224	VICVectCntl9	Vector control 9 register	R/W	0
0xFFFF F228	VICVectCntl10	Vector control 10 register	R/W	0
0xFFFF F22C	VICVectCntl11	Vector control 11 register	R/W	0
0xFFFF F230	VICVectCntl12	Vector control 12 register	R/W	0
0xFFFF F234	VICVectCntl13	Vector control 13 register	R/W	0
0xFFFF F238	VICVectCntl14	Vector control 14 register	R/W	0
0xFFFF F23C	VICVectCntl15	Vector control 15 register	R/W	0

VIC :: Registers

VICIntEnable	Function	Reset Value
31:0	When this register is read, 1s indicate interrupt requests or software interrupts that are enabled to contribute to FIQ or IRQ. When this register is written, ones enable interrupt requests or software interrupts to contribute to FIQ or IRQ, zeroes have no effect. See the VICIntEnClear register (Table 46 below), for how to disable interrupts.	0

VICIntEnClear	Function	Reset Value
31:0	1: writing a 1 clears the corresponding bit in the Interrupt Enable register, thus disabling interrupts for this request. 0: writing a 0 leaves the corresponding bit in VICIntEnable unchanged.	0

VICIntSelect	Function	Reset Value
31:0	1: the interrupt request with this bit number is assigned to the FIQ category. 0: the interrupt request with this bit number is assigned to the IRQ category.	0

VIC :: Registers

VICVectCntl0-15	Function	Reset Value
5	1: this vectored IRQ slot is enabled, and can produce a unique ISR address when its assigned interrupt request or software interrupt is enabled, classified as IRQ, and asserted.	0
4:0	The number of the interrupt request or software interrupt assigned to this vectored IRQ slot. As a matter of good programming practice, software should not assign the same interrupt number to more than one enabled vectored IRQ slot. But if this does occur, the lower-numbered slot will be used when the interrupt request or software interrupt is enabled, classified as IRQ, and asserted.	0

VICVectAddr0-15	Function	Reset Value
31:0	When one or more interrupt request or software interrupt is (are) enabled, classified as IRQ, asserted, and assigned to an enabled vectored IRQ slot, the value from this register for the highest-priority such slot will be provided when the IRQ service routine reads the Vector Address register (VICVectAddr).	0

VICDefVectAddr	Function	Reset Value
31:0	When an IRQ service routine reads the Vector Address register (VICVectAddr), and no IRQ slot responds as described above, this address is returned.	0

VIC :: Registers

VICVectAddr	Function	Reset Value
31:0	If any of the interrupt requests or software interrupts that are assigned to a vectored IRQ slot is (are) enabled, classified as IRQ, and asserted, reading from this register returns the address in the Vector Address Register for the highest-priority such slot. Otherwise it returns the address in the Default Vector Address Register. Writing to this register does not set the value for future reads from it. Rather, this register should be written near the end of an ISR, to update the priority hardware.	0

VICIRQStatus	Function	Reset Value
31:0	1: the interrupt request with this bit number is enabled, classified as IRQ, and asserted.	0

VICFIQStatus	Function	Reset Value
31:0	1: the interrupt request with this bit number is enabled, classified as FIQ, and asserted.	0

VIC :: Interrupt Source

Block	Flag(s)	VIC Channel #
WDT	Watchdog Interrupt (WDINT)	0
-	Reserved for software interrupts only	1
ARM Core	Embedded ICE, DbgCommRx	2
ARM Core	Embedded ICE, DbgCommTx	3
Timer 0	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	4
Timer 1	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	5
UART 0	Rx Line Status (RLS) Transmit Holding Register empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI)	6
UART 1	Rx Line Status (RLS) Transmit Holding Register empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Status Interrupt (MSI)	7
PWM0	Match 0 - 6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6) Capture 0 - 3 (CR0, CR1, CR2, CR3)	8
I2C	SI (state change)	9
SPI	SPIF, MODF	10
-	reserved	11
PLL	PLL Lock (PLOCK)	12
RTC	RTCCIF (Counter Increment), RTCALF (Alarm)	13
System Control	External Interrupt 0 (EINT0)	14
System Control	External Interrupt 1 (EINT1)	15
System Control	External Interrupt 2 (EINT2)	16

VIC :: Sample code

```
VICIntSelect &= ~(1 << VIC_TIMER0);          // TIMER0 selected as IRQ
VICVectCntl0 = VIC_ENABLE | VIC_TIMER0;      // enable TIMER0 in 0
VICVectAddr0 = (unsigned long) timer0ISRH;  // set intr vector in 0
VICIntEnable = 1 << VIC_TIMER0;              // TIMER0 intr enabled
```

```
timer0ISRH:
        sub     lr, lr, #4
        stmfd   sp!, {r0-r12, lr}
        ldr     r1, =timer0ISR
        mov     lr, pc
        bx      r1
        ldmfd   sp!, {r0-r12, pc}^
```

```
ldr     r0, =__stack_top__
msr     CPSR_c, #MODE_IRQ|I_BIT|F_BIT
        /* IRQ Mode */
mov     sp, r0
```

```
.word    0xb9205f80
ldr      PC, [PC,#-0xff0]
```

```
void timer0ISR(void) {
        sysClk++;
        TIMER0_IR = 0x01; // clear interrupt
        VICVectAddr = 0; // end of interrupt - dummy write
}
```