BITS Pilani presentation

BITS Pilani
Pilani Campus

Ramani Kalpathi
Automotive Electronics

innovate achieve lead

BITS Pilani

Pilani Campus

AEL ZG512 Embedded Systems
Lecture No. 4

Embedded Architecture 2 – ARM Based Controller

innovate achieve lead

Introduction to LPC23xx

- AMBA Bus Architecture
- GPIO, Timer, Watch dog
- Interrupt Handling -VIC , ADC/DAC

Introduction to LPC23xx

- LPC23xx series are ARM-based microcontrollers for applications requiring serial communications.
- Incorporate a 10/100 Ethernet MAC, USB 2.0 Full Speed interface, four UARTs, two CAN channels, an SPI interface, two Synchronous Serial Ports (SSP), three I2C interfaces, an I2S interface, and a MiniBus (8-bit data/16-bit address parallel bus).
- LPC2361/62
- LPC2364/65/66/67/68
- LPC2377/78
- LPC2387
- LPC2388

LPC 23XX - General features

General features

- ARM7TDMI-S processor, running at up to 72 MHz.
- Up to 512 kB on-chip Flash Program Memory with In-System Programming (ISP)
- Up to 64 kB of SRAM on the ARM local bus, 16 kB Static RAM for Ethernet interface, 8 kB RAM for USB interface.
- Dual AHB system that provides for simultaneous Ethernet DMA, USB DMA, and program execution from on-chip flash with no contention between those functions.
- Advanced Vectored Interrupt Controller, supporting up to 32 vectored interrupts.
- DMA controller (GPDMA) on AHB that can be used with the SSPserial interfaces, the I2S port, and the SD/MMC card port, as well as for memory-to-memory transfers.

LPC 2367/68 for example..

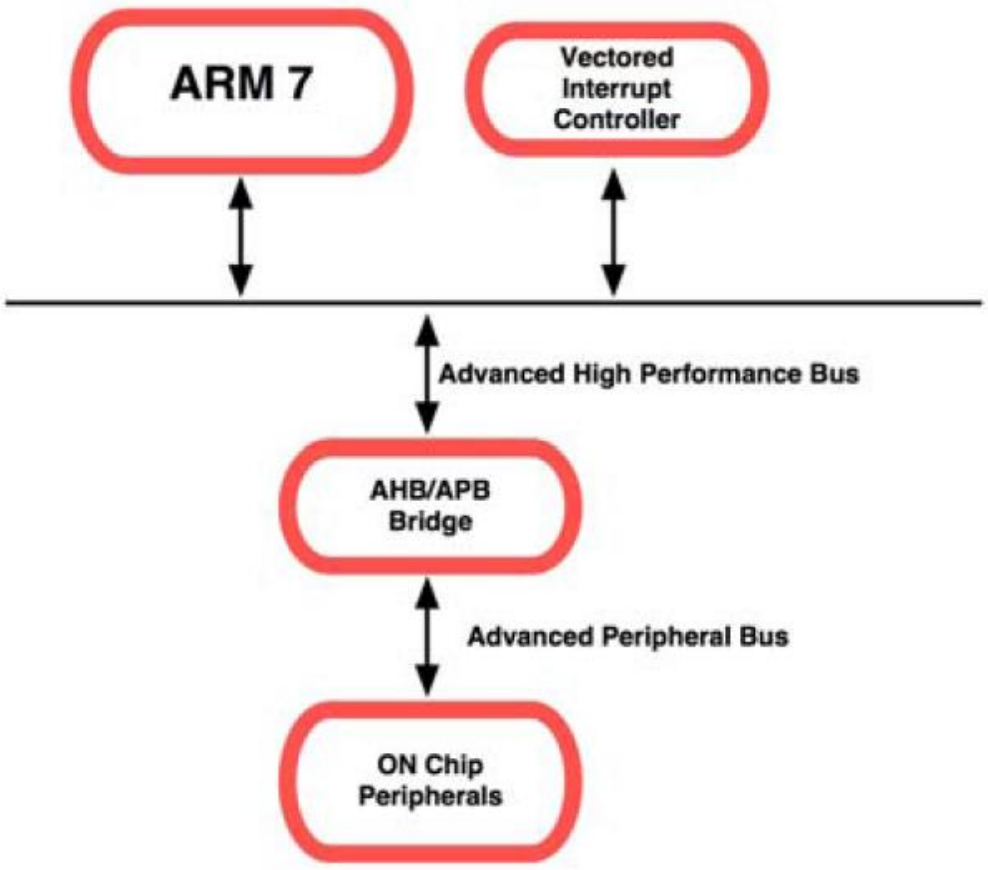
- Secure Digital (SD) /MultiMediaCard (MMC) memory card interface.
- Up to 70 I/O pins, 10 bit A/D converter 8 channels, 10 bit D/A
- Four timers with two capture inputs, four compare outputs
- PWM support for three-phase motor control
- Real-Time Clock (RTC) with 2 kB Static RAM storage
- Watch dog timer, Standard ARM Test/Debug interface
- Single 3.3 V power supply (3.0 V to 3.6 V).
- Idle, Sleep, Power-down, and Deep power-down modes.
- Four external interrupt inputs
- Brownout detection, Power On Reset (POR)
- 4 MHz internal RC oscillator

LPC23XX Overview

innovate achieve lead

Part	Local bus SRAM (kB)	Flash (kB)	EMC	USB/ GP SRAM (kB)	USB device	USB host/ OTG	Ethernet	Ethernet GP SRAM (kB)	CAN channels	SD/ MMC	ADC channels	GPIO pins
LPC2361	8	64	no	8	yes	yes	no	16	2	no	6	70
LPC2362	32	128	no	8	yes	yes	yes	16	2	no	6	70
LPC2364	8	128	no	8	yes	no	yes	16	2	no	6	70
LPC2365	32	256	no	8	no	no	yes	16	-	no	6	70
LPC2366	32	256	no	8	yes	no	yes	16	2	no	6	70
LPC2367	32	512	no	8	no	no	yes	16	-	yes	6	70
LPC2368	32	512	no	8	yes	no	yes	16	2	yes	6	70
LPC2377	32	512	Mini	8	no	no	yes	16	-	yes	8	104
LPC2378	32	512	Mini	8	yes	no	yes	16	2	yes	8	104
LPC2387	64	512	no	16	yes	yes	yes	16	2	yes	6	70
LPC2388	64	512	Mini	16	yes	yes	yes	16	2	yes	8	104

Bus Interface



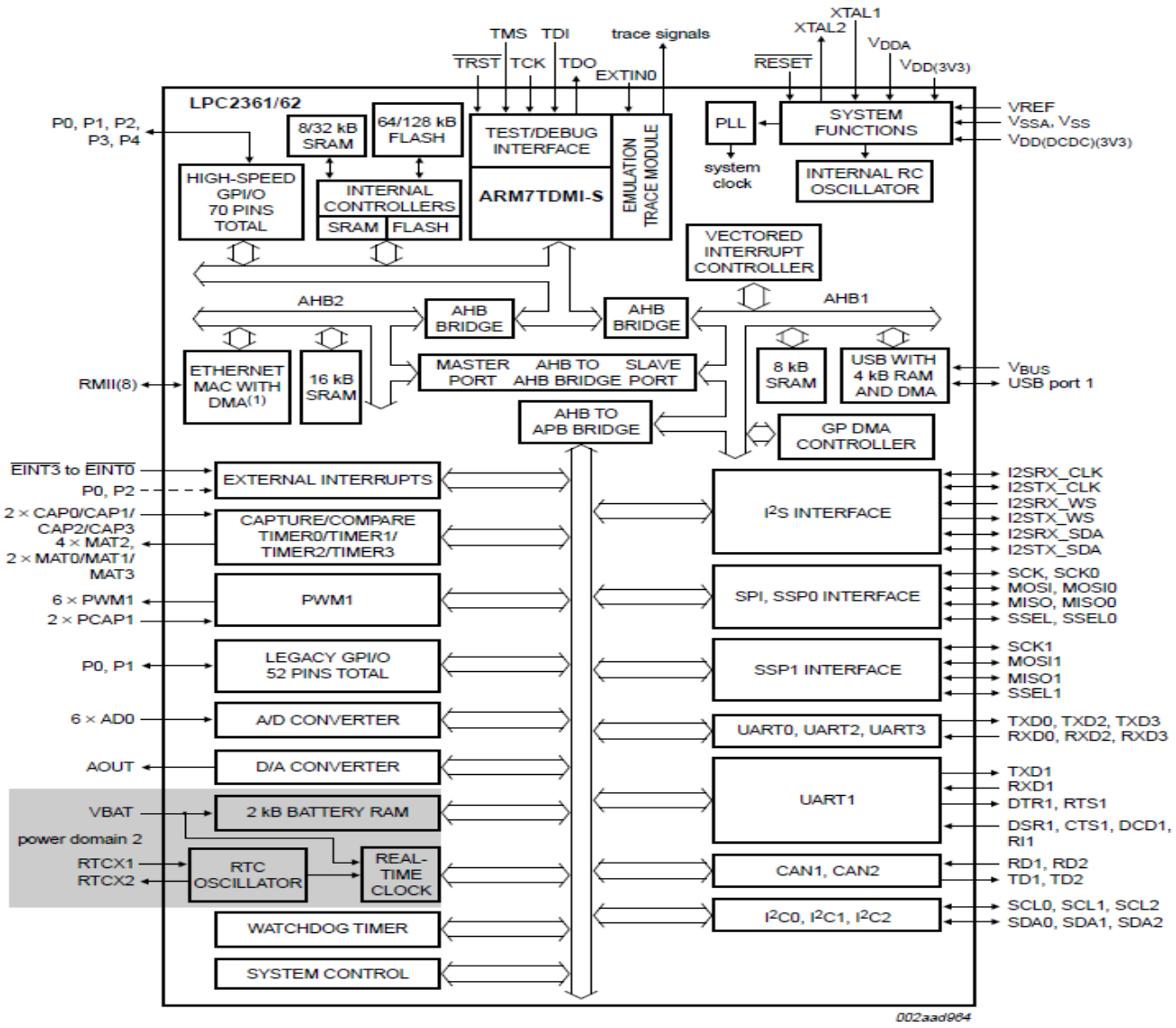
AMBA Bus interfaces

- AMBA – Advanced Microcontroller Bus Architecture
- Advanced High-performance Bus (AHB) interfacing to high speed on-chip peripherals and external memory
- Advanced Peripheral Bus (APB) for connection to other on-chip peripheral functions.
- Arm processor core is a bus master – capable of initiating a bus transfer, peripherals are bus slaves
- ‘ARM7 Local Bus for closely coupled, high speed access to the majority of on-chip memory
- The microcontroller implements two AHB buses in order to allow the Ethernet block to operate without interference caused by other system activity. The primary AHB, referred to as AHB1, includes the Vectored Interrupt Controller, General Purpose DMA Controller, External Memory Controller, USB interface, and 8/16 kB SRAM primarily intended for use by the USB.

AHB-APB Bus

- The second AHB, referred to as AHB2, includes only the Ethernet block and an associated 16 kB SRAM.
- Bus masters with access to AHB1 are the ARM7 itself, the USB block, the General Purpose DMA function, and the Ethernet block (via the bus bridge from AHB2).
- AHB peripherals are allocated a 2 MB range of addresses at the very top of the 4 GB ARM memory space.
- Each AHB peripheral is allocated a 16 kB address space within the AHB address space.

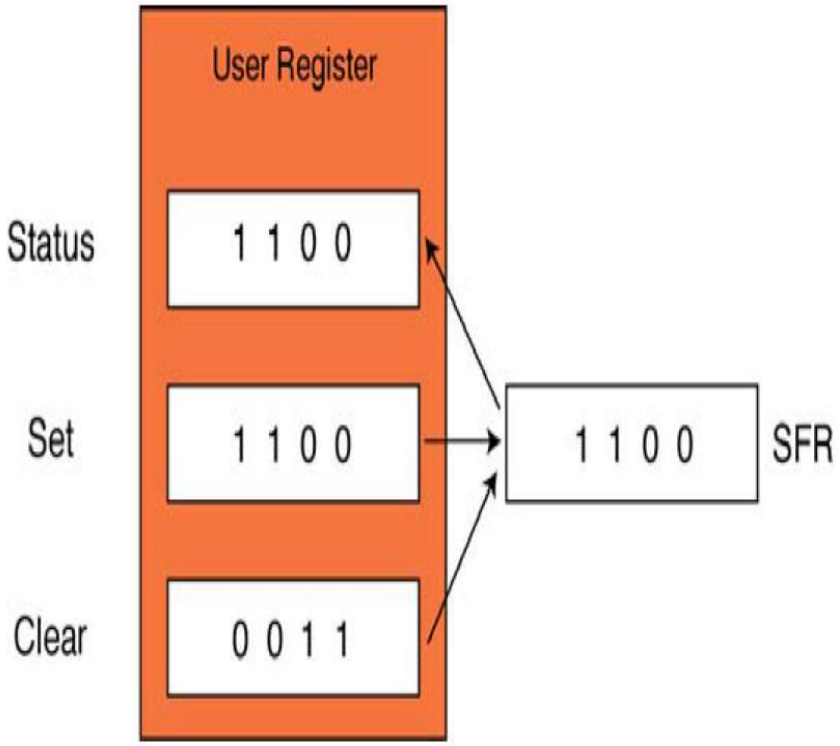
LPC 23XX Block Diagram



General Purpose Input Output

innovate achieve lead

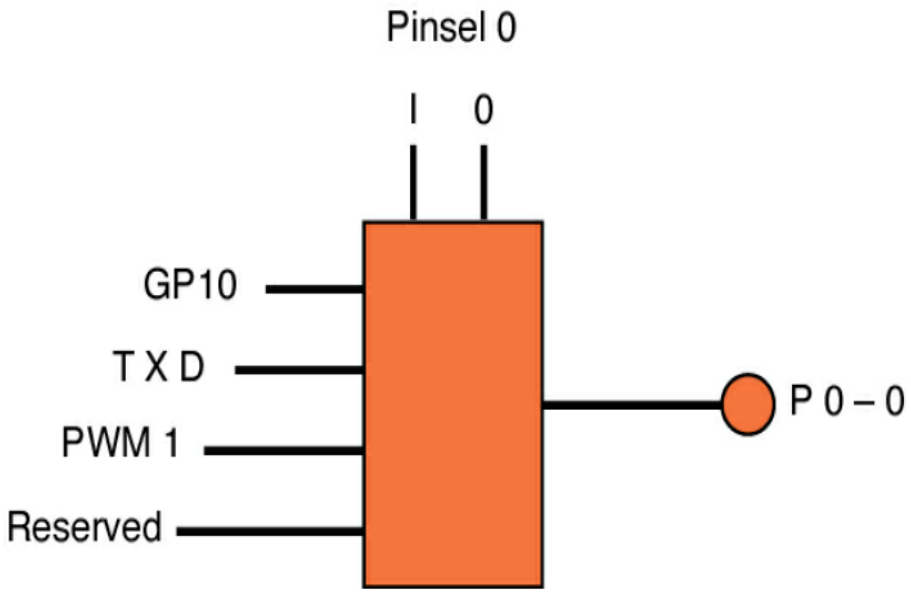
Address range	General use	Address range details and description	
0x0000 0000 to 0x3FFF FFFF	on-chip NV memory and fast I/O	0x0000 0000 - 0x0007 FFFF	flash memory (up to 512 kB)
		0x3FFF C000 - 0x3FFF FFFF	fast GPIO registers



As a general rule all Special Function Registers originating from ARM are controlled by three registers: a Set, Clear and Status register.

NB To clear bits you must write a logic 1 to the relevant bit in the Clear Register.

Configuration for GPIO



The Pin Select module allows each I/O pin to be multiplexed between one of four peripherals.

GPIO on LPC23XX

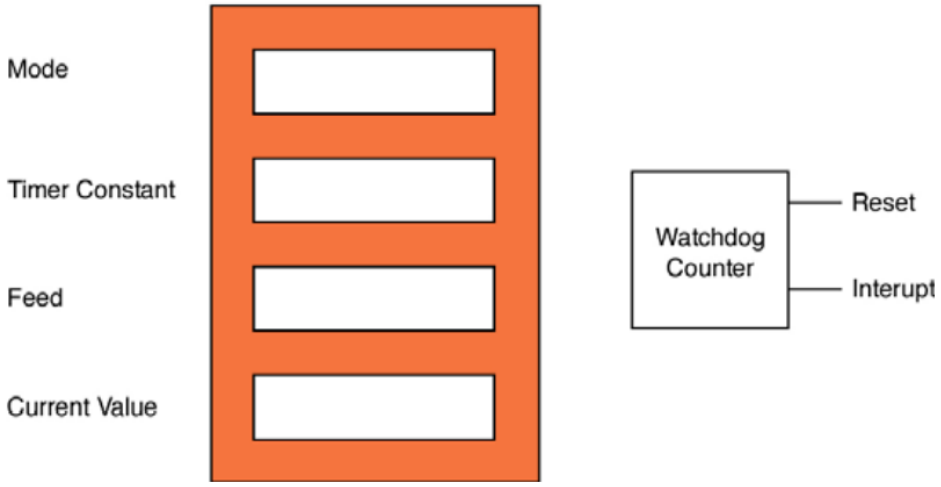
- LPC23XX have five ports of 32 bits each - 160 pins.
- Port 0, Port1 have control through legacy control registers using the APB bus which is quite slow.
- A second set of registers that can control the ports in a fast mode is provided called Fast GPIO control registers. The Fast GPIO registers also have a mask capability.
- The FIODIR allows each pin to be configured as an input (0) or an output(1)
- If the pin is configured as an output, the FIOSET or FIOCLR can be used to set the bit or clear the bit by writing a 1 in the appropriate register. The state of the pin can be read by reading the FIOPIN register.
- If the mask bit in the FIOMASK register is set to 0, then FIOSET, FIOCLR or FIOPIN can be updated.

Toggling GPIO pins

```
int main (void)
{
unsigned int delay;
IODIR1 = 0xFFFFFFFF; //Direction to output
 while(1){
Delay = 10000;
while(--delay);
IOCLR1 = 0xFFFFFFFF;
Delay = 10000;
while(--delay);
IOSET1 = 0xFFFFFFFF;
 }
}
```

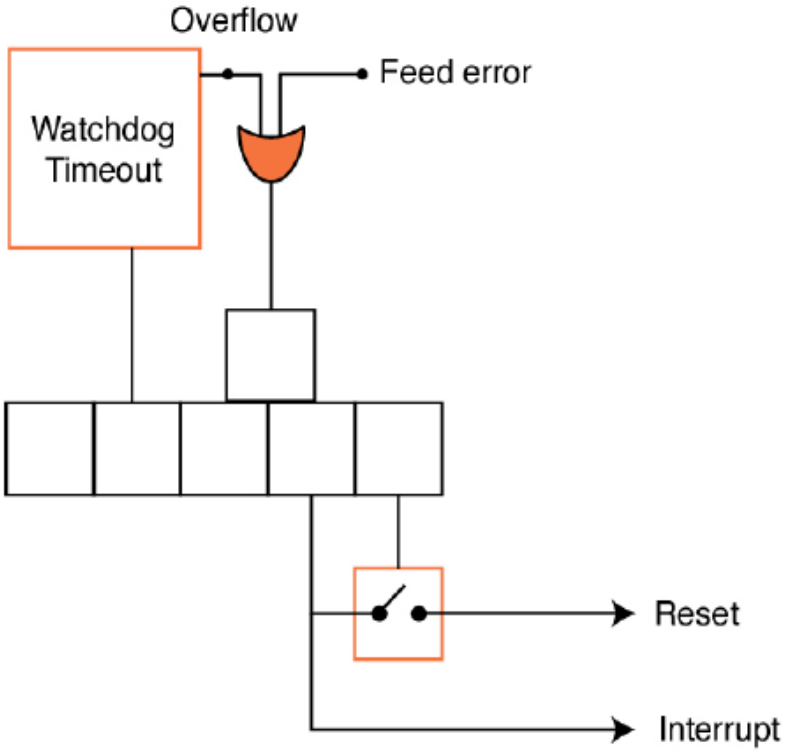
Watchdog Timer

- Watchdog timer that will help in system recovery if the program has crashed.
- Watchdog may be clocked from internal RC oscillator, RTC oscillator or Pclk.
- The watchdog has four registers.
- The timeout period is set by the value loaded into Watchdog constant register WDTCR.
- Wdperiod = Twdck x WDTC x 4



The on-chip watchdog can force a processor reset or interrupt. In the case of a watchdog reset, a flag is set so your code can stop a "soft reset".

WDT Mode Register



The watchdog mode register allows configuration of the watchdog action on underflow (reset or interrupt).

WDT Feed Register

- Once the watchdog timer constant and mode registers are configured, the watchdog is kick-started by a feed sequence of writing 0XAA followed by 0x55 into the feed register.
- The watchdog needs regular feed sequence during the course of normal working of the program to prevent it from counting down to zero and resetting the controller.
- *Under an abnormal operation that causes the program to execute program memory that is not defined by the programmer, the regular feed sequences are not executed causing the watchdog timer to reset the controller and initialize the controller to a known startup sequence.*

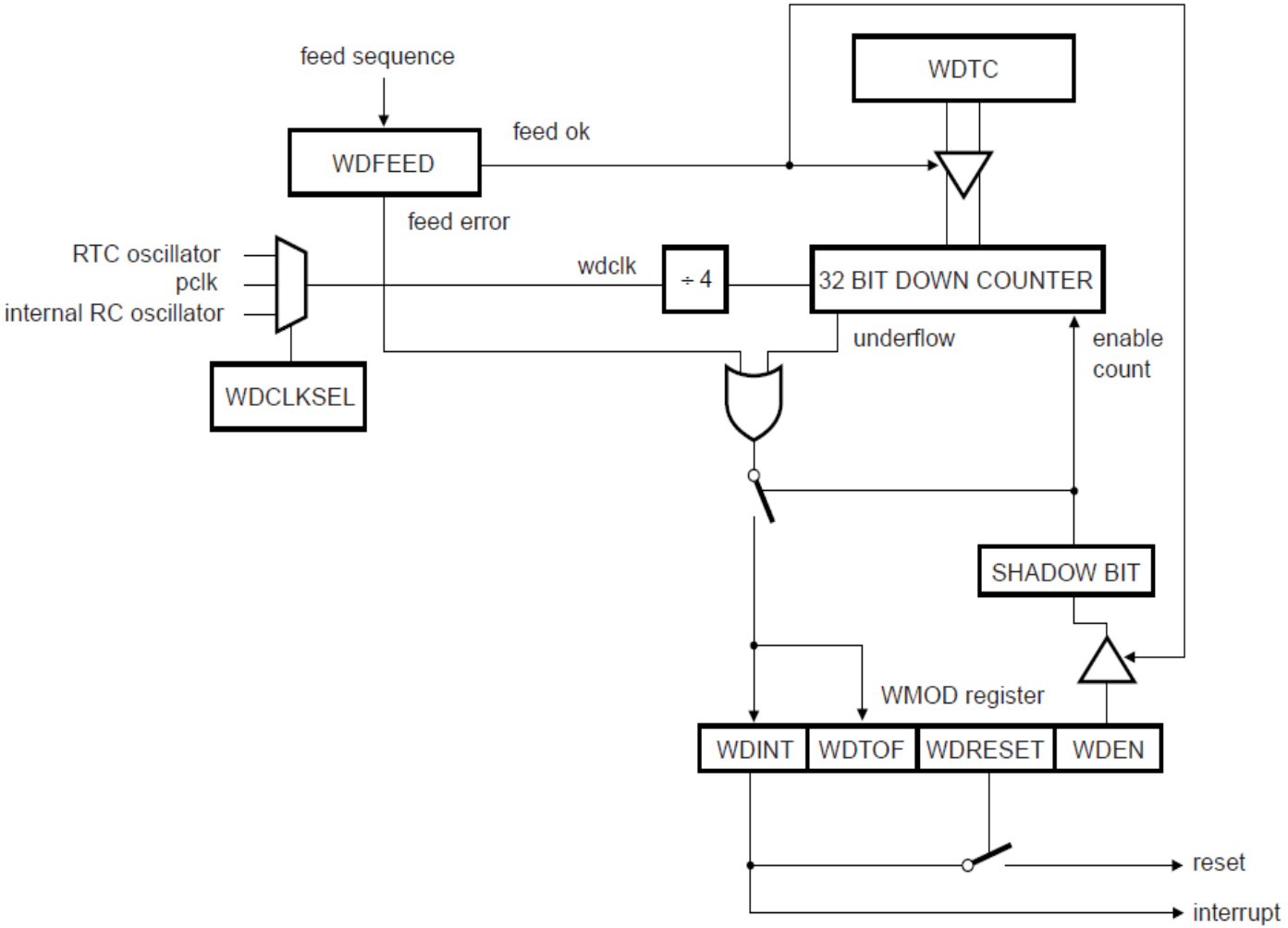
WDT Registers

Table 494. Watchdog register map

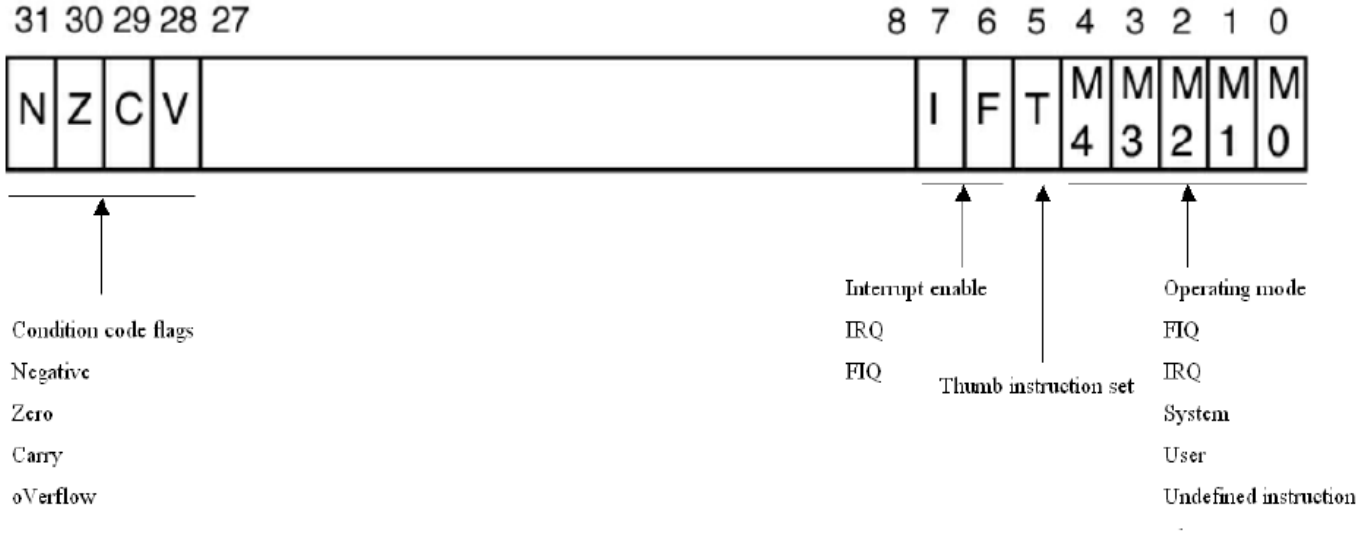
Name	Description	Access	Reset Value[1]	Address
WDMOD	Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer.	R/W	0	0xE000 0000
WDTC	Watchdog timer constant register. This register determines the time-out value.	R/W	0xFF	0xE000 0004
WDFEED	Watchdog feed sequence register. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in WDTC.	WO	NA	0xE000 0008
WDTV	Watchdog timer value register. This register reads out the current value of the Watchdog timer.	RO	0xFF	0xE000 000C
WDCLKSEL	Watchdog clock source selection register.	R/W	0	0xE000 0010

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Block Diagram of WDT



Vectored Interrupt Controller



The Current Program Status Register contains condition code flags which indicate the result of data processing operations and User flags which set the operating mode and enable interrupts. The T bit is for reference only.

- I and F are interrupt enable bits in the Current Program Status Register which when set to 1 will disable the interrupts.
- When an exception occurs, the CPU will change modes and the Program Counter will be forced to an exception vector. The vector table starts at zero with the reset vector and then has an exception vector every four bytes.

Vector Table

Exception	Mode	Address
Reset	Supervisor	0x00000000
Undefined instruction	Undefined	0x00000004
Software interrupt (SWI)	Supervisor	0x00000008
Prefetch Abort (instruction fetch memory abort)	Abort	0x0000000C
Data Abort (data access memory abort)	Abort	0x00000010
IRQ (interrupt)	IRQ	0x00000018
FIQ (fast interrupt)	FIQ	0x0000001C

Each operating mode has an associated interrupt vector. When the processor changes mode the PC will jump to the associated vector.

NB: there is a missing vector at 0x00000014.

Priority of Exceptions

Priority	Exception
Highest 1	Reset
2	Data Abort
3	FIQ
4	IRQ
5	Prefetch Abort
Lowest 6	Undefined instruction SWI

Each of the exception sources has a fixed priority. The on-chip peripherals are served by FIQ and IRQ interrupts. Each peripheral's priority may be assigned within these groups.

Exception Processing

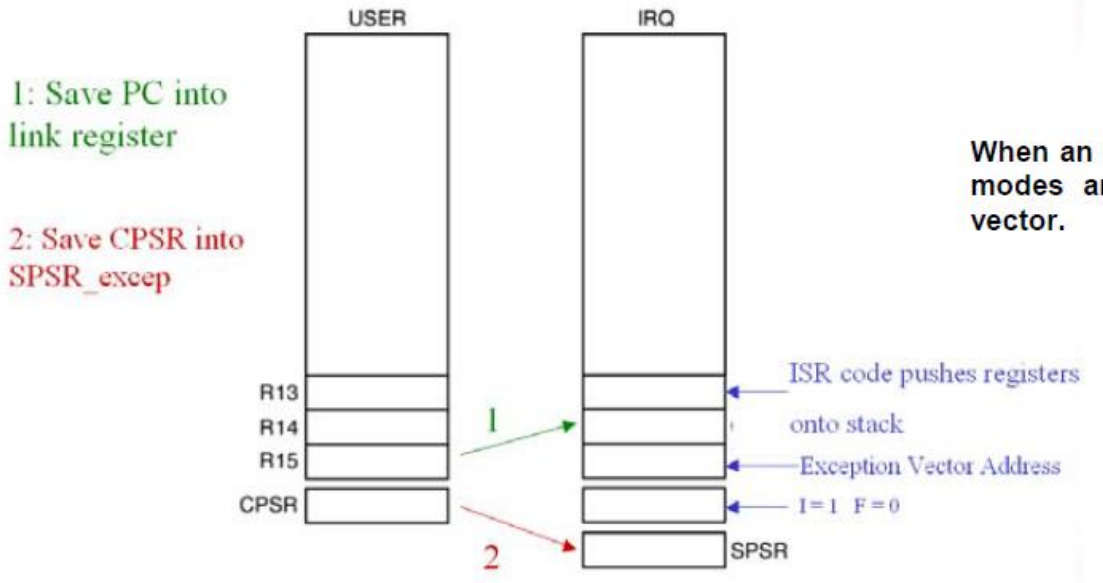
When an exception (IRQ) occurs, the following action takes place.

- The address of the next instruction to be executed (PC+4) is saved in the Link Register.
- CPSR is copied into the SPSR of the Exception Mode that is about to be entered (SPSR_irq).
- The PC is then filled with the address of the Exception Mode Interrupt Vector - 0x00000018.
- The Mode is changed to IRQ mode which causes R13 and R14 to be replaced by IRQ R13 and R14 registers.
- On entry to the IRQ mode, the I bit in CPSR is set, causing the IRQ interrupt line to be disabled.
- From the Exception vector the code will jump to exception ISR (Interrupt Service Routine).

…Exception Processing

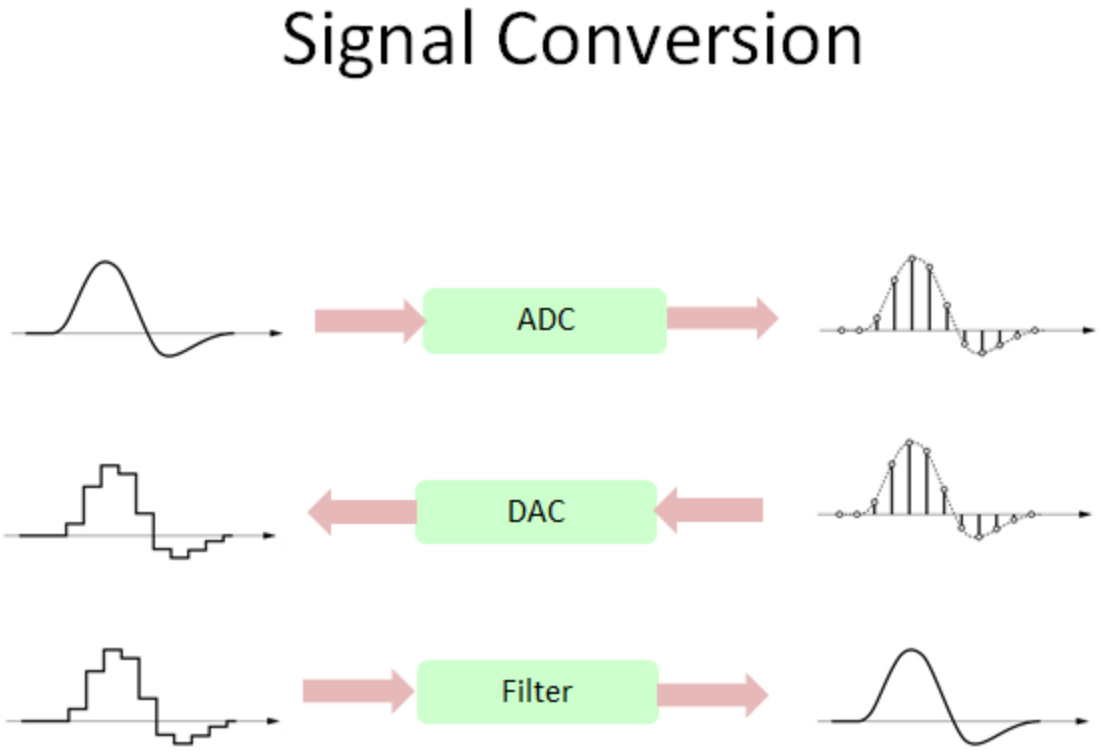
- The code must preserve R0-R12 registers that the ISR will use by pushing them onto the IRQ stack. Once this is done the exception will be processed.
- Once the exception code has been processed, it must return to the User mode and continue where it left off. The ARM set does not contain a Return instruction like other processors and hence manipulating the PC has to be done by regular instructions.

…Exception Processing

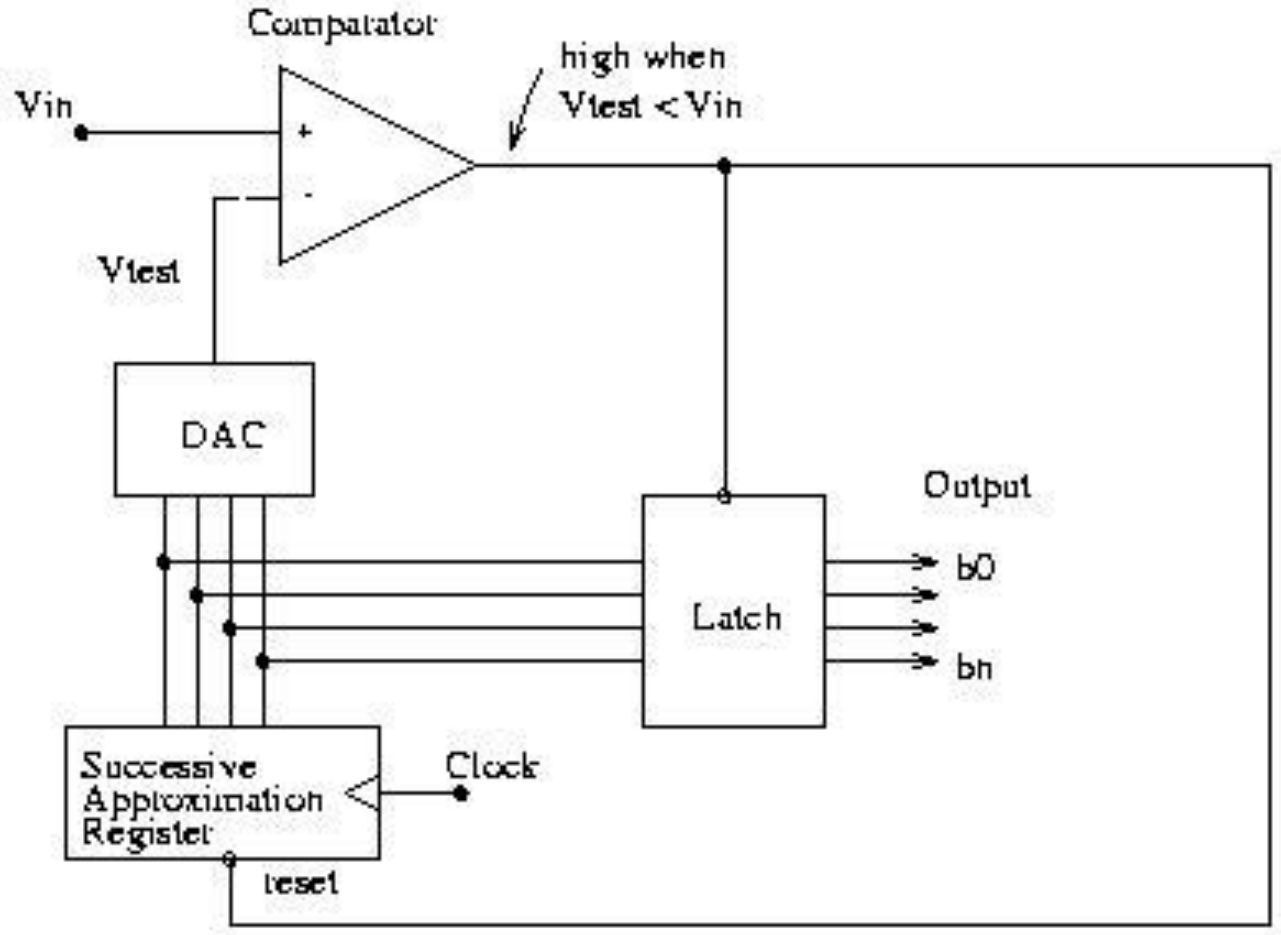


When an exception occurs the CPU will change modes and jump to the associated interrupt vector.

Analog to Digital Conversion

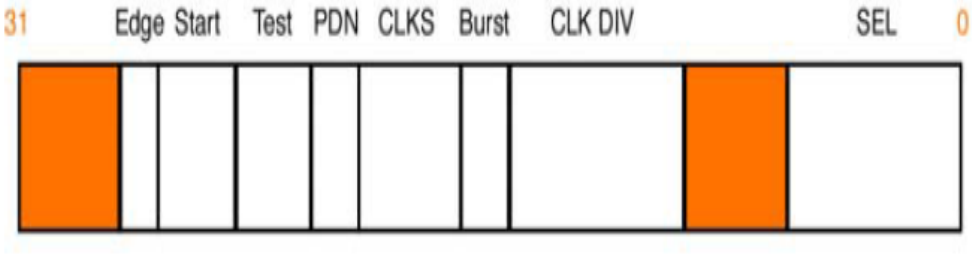


Successive Approximation ADC



Analog to Digital Converter

- The A/D Converter on LPC23XX devices is a 10-bit successive approximation converter with a conversion time of 2.44usec.
- The A/D converter has 6 or 8 multiplexed inputs depending on the device.
- The A/D control register needs to be programmed by setting up the peripheral clock and it must be divided down from the clock input frequency to below 4.5MHz.



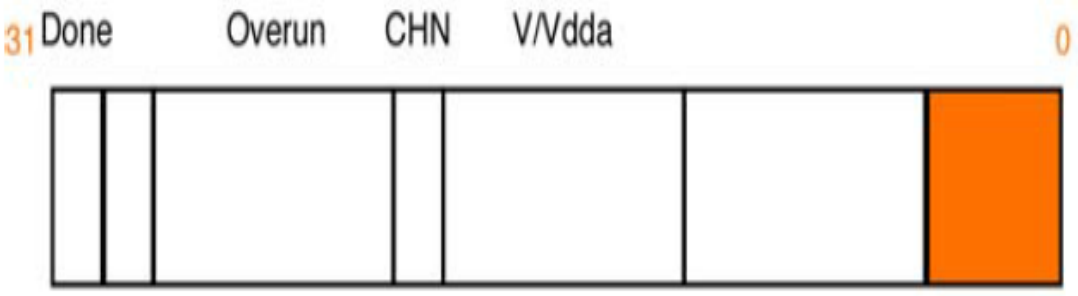
AD Control register: The control register determines the conversion mode, channel and resolution.

Analog to Digital Converter

CLKDIV = (PCLK/ADCLK) -1

- Logic 1 in PDN bit will turn ON the AD converter module.
- The resolution can be programmed to 10 bit resolution by programming the CLKS field.
- The AD conversion can be selected by hardware based conversion or software based conversion.
- The hardware mode can be selected and the AD conversion started after selecting the number of channels.
- The AD conversion result is available in the AD Global register and also in the result register ADDR0 to ADDR7.

Analog to Digital Converter



AD data register: The data register contains the conversion result, channel overrun error and conversion done flag.

At the end of the conversion…

- Done bit is set
- global enable, AD interrupt enable bits are set in the AD interrupt register.
- The result is stored in the V/Vdda as a ratio of the voltage input on the analog channel to the voltage on the Analog voltage power supply input pin.
- Channel number is also stored in the CHN field.
- If the conversion result is not read by the end of next conversion the **result will** be overwritten and the Overrun bit is set.

Analog To Digital Converter

The AD conversion is started by writing 0x0x1 to the START field. The channel is first selected using the SEL field. A single conversion happens and the result is stored in the corresponding ADDR result register.

The DONE bit in the ADDR bit is polled to see if the conversion is completed or the AD interrupt bit can be used to signal a conversion complete status. A conversion event can also be synchronized to start when there is a timer match event occurs on Timer0 or Timer1. The AD event can also be synchronized to happen upon the presence of an edge occurrence on P0.16 or P0.22.

ADC Converter code

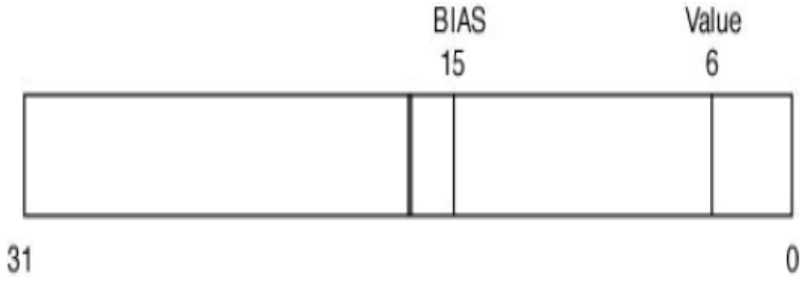
```
VPBDIV = 0x02;              //Set the Pclk to 30 MHz
IODIR1 = 0x00FF0000;        // P1.16..23 defined as Outputs
ADCR   = 0x00270601;        // Setup A/D: 10-bit AIN0 @ 3MHz
ADCR   |= 0x01000000;       // Start A/D Conversion

while(1)
{

do
{
    val = ADDR;       // Read A/D Data Register
}
```

Digital to Analog Converter

- LPC23xx devices have a 10 bit Digital to Analog converter using a single register.
- The DAC is enabled by writing to bits 20 and 21 of PINSEL1 and converting pin 0.26 from GPIO to AOUT function.
- The pin is also shared by A to D converter.
- The VALUE bits in the control register are written. If the BIAS bit is set to 1, the conversion time is 2.5usec and it can drive 700uA of current from this pin.
- If the BIAS is set to 0, the conversion time is 1usec and the drive current is 350uA.



The DAC is controlled by a single register. The value to be converted is written here along with the bias value.