Lec#09 (part 4) More on Interrupts

Last update: Cristinel Ababei, Feb. 2013

1. MCB1700 Board Quick Overview

The hardware block diagram in Fig.1 [1] displays input, configuration, power system, and User I/O on the MCB1700 board.

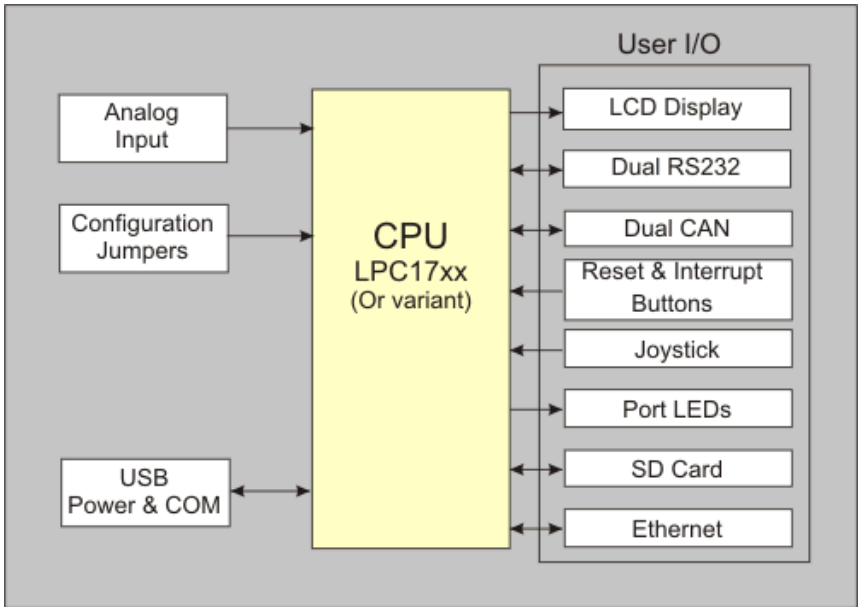


Figure 1 Main components of MCB1700 board

Microcontroller: NXP LPC1768
XTAL Frequency: 12 MHz
Peripherals on board:
2 × RS232 Interfaces,
2 × CAN Interfaces,
1 × Ethernet Interface,
1 × JTAG Interface,
1 × ETM Interface,
2 × Cortex Debug Interfaces,
1 × LCD Display,
1 × USB Device/Host/OTG Interface,
1 × Analog Output (connected to speaker by default)
1 × Analog Input (connected to potentiometer by default)

2. LPC17xx MCU Overview

The LPC1700 series of low power cost-effective Cortex-M3 microcontrollers feature best-in-class peripheral support such as Ethernet, USB 2.0 Host/OTG/Device, and CAN 2.0B.
Operating at speeds up to 120 MHz, they have up to 512 KB of FLASH, up to 64 KB of SRAM, 12-bit A/D and 10-bit D/A converters as well as an internal RC oscillator.

Block diagram shown in Fig.2 [2].

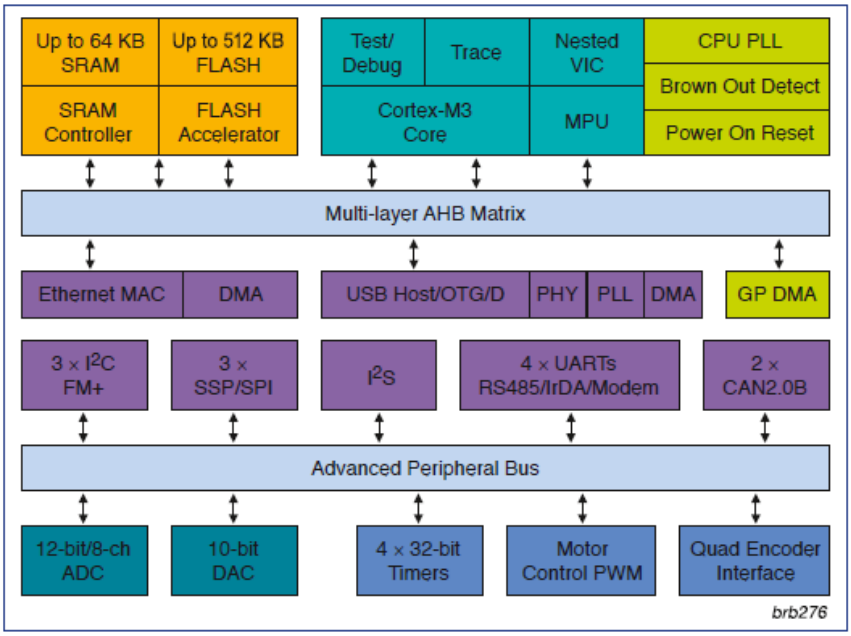


Figure 2 LPC17xx block diagram.

Simplified block diagrams of LPC1768 MCU are shown in Fig.3,4 [3].

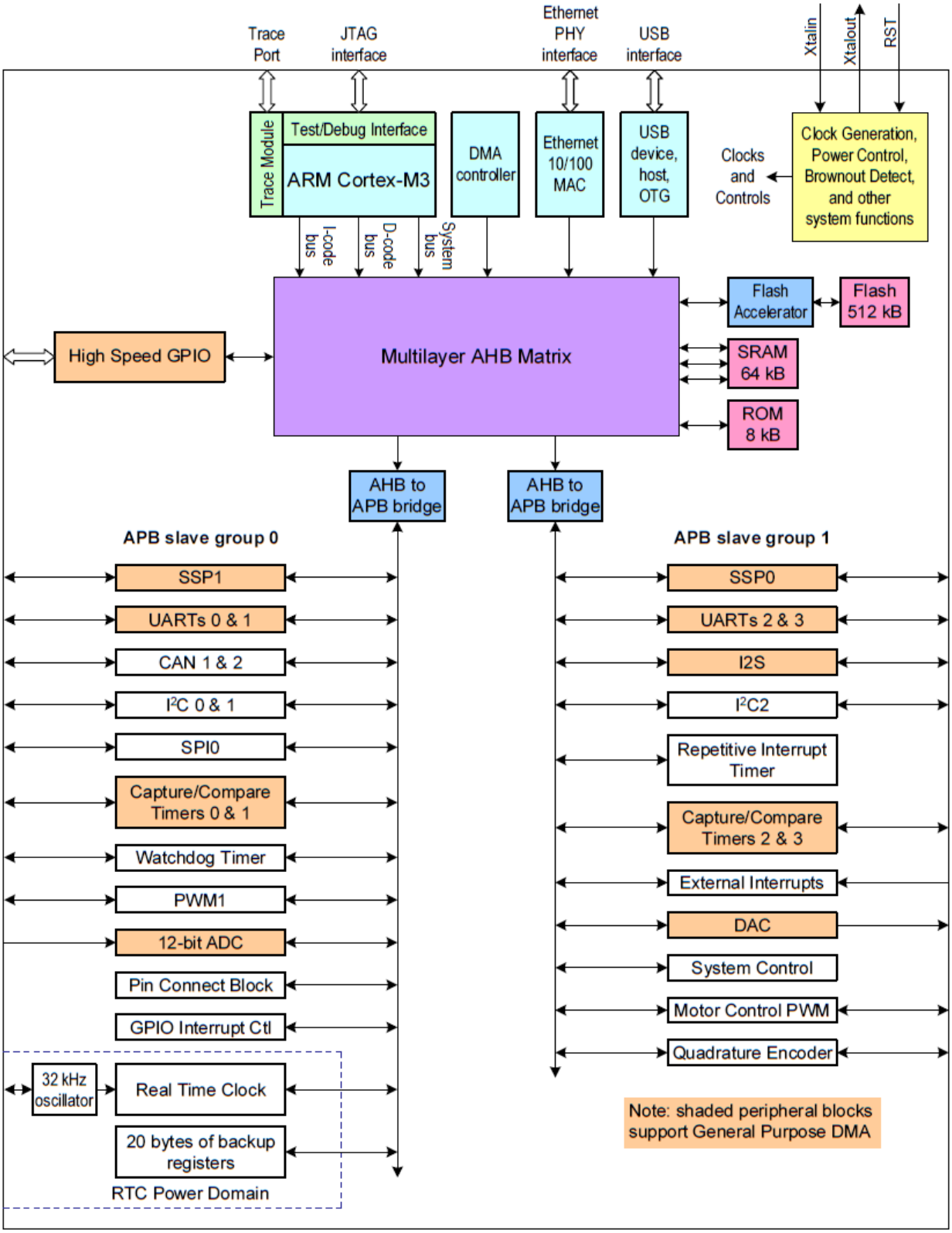


Figure 3 Simplified block diagram of LPC1768 MCU.

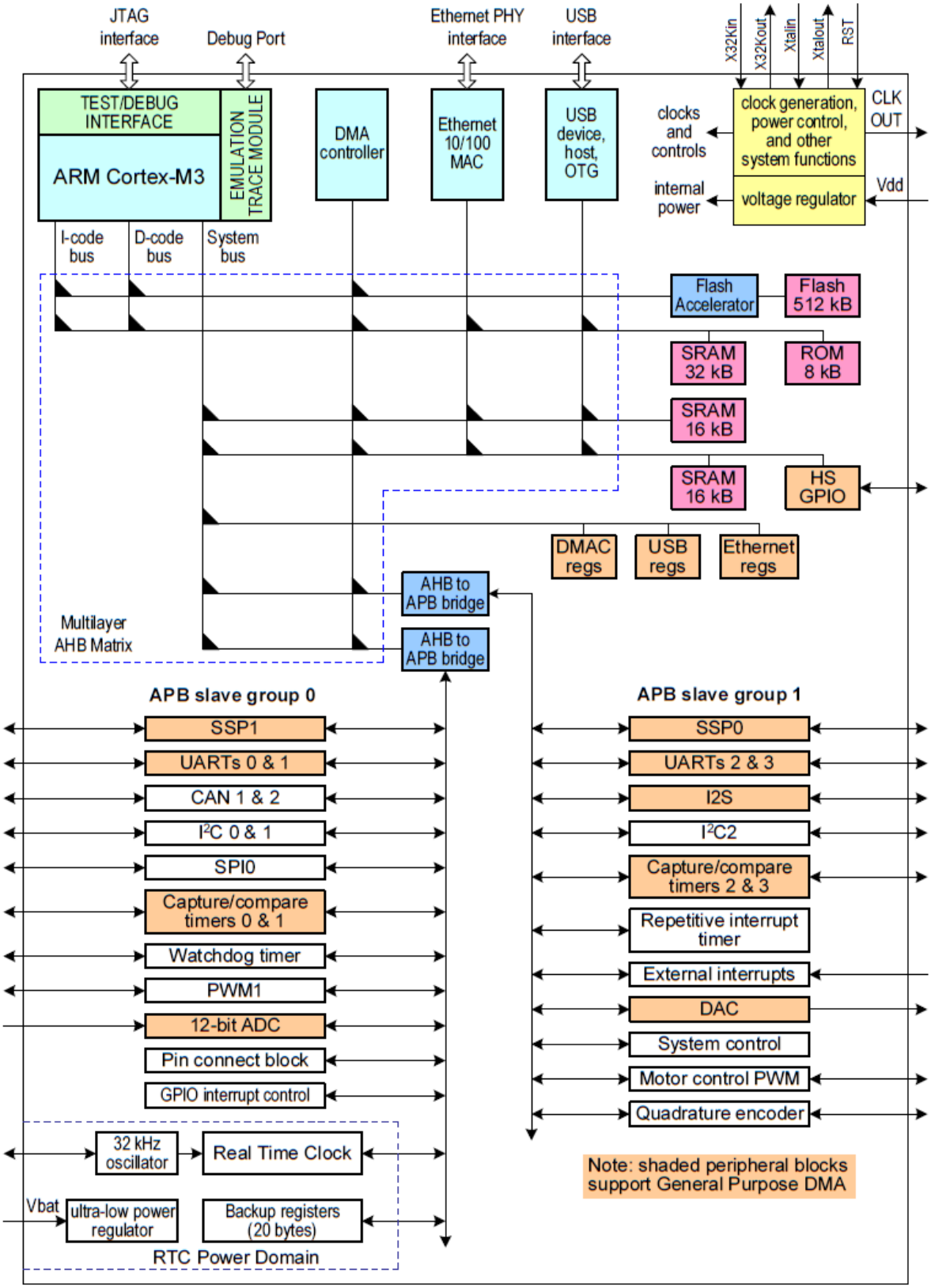


Figure 4 LPC1768 block diagram, CPU, and buses.

3. Exception and Interrupts: Nested Vectored Interrupt Controller (NVIC)

--Nested Vectored Interrupt Controller that is an integral part of the ARM Cortex-M3
--Tightly coupled interrupt controller provides low interrupt latency
--Controls system exceptions and peripheral interrupts
--In the LPC17xx, the NVIC supports 35 vectored interrupts
--32 programmable interrupt priority levels, with hardware priority level masking
--Relocatable vector table
--Non-Maskable Interrupt
--Software interrupt generation

Table 50 from User Manual [3] lists the interrupt sources for each peripheral function. Each peripheral device may have one or more interrupt lines to the Vectored Interrupt Controller. Each line may represent more than one interrupt source, as noted.

Exception numbers relate to where entries are stored in the exception vector table. Interrupt numbers are used in some other contexts, such as software interrupts.

Table 50. Connection of interrupt sources to the Vectored Interrupt Controller

Interrupt ID	Exception Number	Vector Offset	Function	Flag(s)
0	16	0x40	WDT	Watchdog Interrupt (WDINT)
1	17	0x44	Timer 0	Match 0 - 1 (MR0, MR1) Capture 0 - 1 (CR0, CR1)
2	18	0x48	Timer 1	Match 0 - 2 (MR0, MR1, MR2) Capture 0 - 1 (CR0, CR1)
3	19	0x4C	Timer 2	Match 0-3 Capture 0-1
4	20	0x50	Timer 3	Match 0-3 Capture 0-1
5	21	0x54	UART0	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO)
6	22	0x58	UART1	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Control Change End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO)
7	23	0x5C	UART 2	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO)
8	24	0x60	UART 3	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO)
9	25	0x64	PWM1	Match 0 - 6 of PWM1 Capture 0-1 of PWM1

10	26	0x68	I^2C0	SI (state change)
11	27	0x6C	I^2C1	SI (state change)
12	28	0x70	I^2C2	SI (state change)
13	29	0x74	SPI	SPI Interrupt Flag (SPIF) Mode Fault (MODF)
Interrupt ID	**Exception Number**	**Vector Offset**	**Function**	**Flag(s)**
14	30	0x78	SSP0	Tx FIFO half empty of SSP0 Rx FIFO half full of SSP0 Rx Timeout of SSP0 Rx Overrun of SSP0
15	31	0x7C	SSP 1	Tx FIFO half empty Rx FIFO half full Rx Timeout Rx Overrun
16	32	0x80	PLL0 (Main PLL)	PLL0 Lock (PLOCK0)
17	33	0x84	RTC	Counter Increment (RTCCIF) Alarm (RTCALF)
18	34	0x88	External Interrupt	External Interrupt 0 (EINT0)
19	35	0x8C	External Interrupt	External Interrupt 1 (EINT1)
20	36	0x90	External Interrupt	External Interrupt 2 (EINT2)
21	37	0x94	External Interrupt	External Interrupt 3 (EINT3). **Note:** EINT3 channel is shared with GPIO interrupts
22	38	0x98	ADC	A/D Converter end of conversion
23	39	0x9C	BOD	Brown Out detect
24	40	0xA0	USB	USB_INT_REQ_LP, USB_INT_REQ_HP, USB_INT_REQ_DMA
25	41	0xA4	CAN	CAN Common, CAN 0 Tx, CAN 0 Rx, CAN 1 Tx, CAN 1 Rx
26	42	0xA8	GPDMA	IntStatus of DMA channel 0, IntStatus of DMA channel 1
27	43	0xAC	I^2S	irq, dmareq1, dmareq2
28	44	0xB0	Ethernet	WakeupInt, SoftInt, TxDoneInt, TxFinishedInt, TxErrorInt, TxUnderrunInt, RxDoneInt, RxFinishedInt, RxErrorInt, RxOverrunInt.
29	45	0xB4	Repetitive Interrupt Timer	RITINT
30	46	0xB8	Motor Control PWM	IPER[2:0], IPW[2:0], ICAP[2:0], FES
31	47	0xBC	Quadrature Encoder	INX_Int, TIM_Int, VELC_Int, DIR_Int, ERR_Int, ENCLK_Int, POS0_Int, POS1_Int, POS2_Int, REV_Int, POS0REV_Int, POS1REV_Int, POS2REV_Int
32	48	0xC0	PLL1 (USB PLL)	PLL1 Lock (PLOCK1)
33	49	0xC4	USB Activity Interrupt	USB_NEED_CLK
34	50	0xC8	CAN Activity Interrupt	CAN1WAKE, CAN2WAKE

As we can see, the LPC1768 microprocessor can have many sources of interrupts. Selected GPIO pins can also be set to generate interrupts.

4. Input/Output Ports

Detailed information on this topic can be found in chapters 7,8, and 9 of the LPC17xx user manual [3].

a) Pin function:

The LPC1768 microprocessor on the MCB1700 board has 100 pins. Fig.5 [4] shows the functionality of most of these pins. **Most pins can have more than one function.** For example, pin number 73, named as P2.2, can take four different roles as indicated by P2.2/PWM1.3/CTS1/TRACEDATA3.

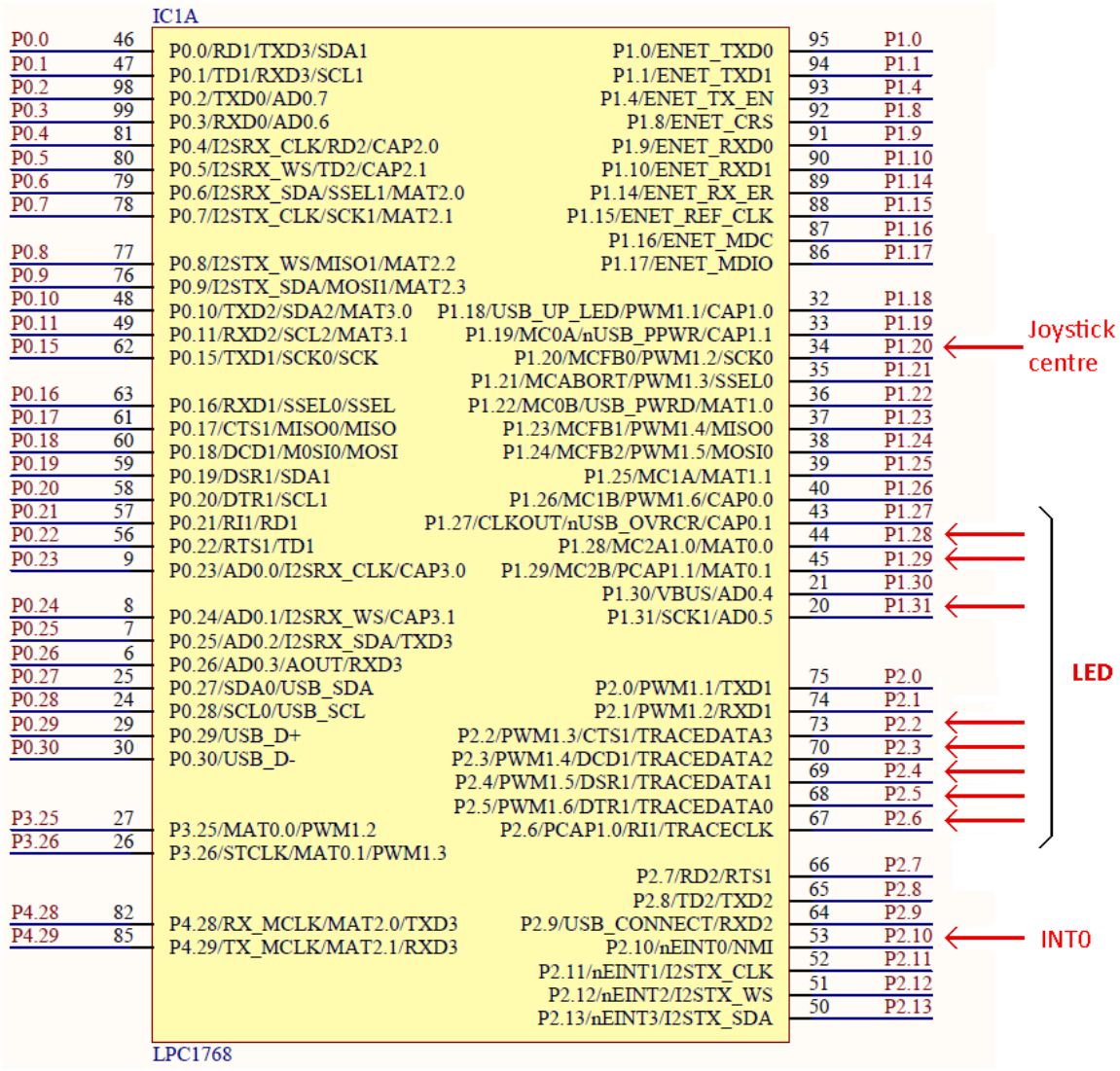


Figure 5 Pins of LPC178.

Many pins of LPC1768 can have up to four different functionalities. For each pin there is a two-bit field in a register named **PINSELx which controls the functionality of the pin**. Table D.1 shows how the two bits should be set to define the functionality of a pin. For more information refer to tables 79-86 of the LPC17xx user manual.

Many times we'll define the pins we'll work with as GPIO (general purpose input output) – the default functionality. We saw an example of using a pin as a GPIO in the Blinky examples of lab#1.

There are ten PINSEL (pin select) registers. Two of them, PINSEL3 and PINSEL4, are used for pins connected to LEDs. The PINSEL3 (address 0x4002 C00C) controls P1[31:16] pins and the PINSEL4 (0x4002 C010) controls P2[15:0] pins. Table D.2 shows pin function select for PINSEL3 (partial), and Table D.3 shows pin function select for PINSEL4 (partial).

Table D.1 – Pin function select register bits

PINSEL0..9 values	Function	Value after reset
00	Primary (default) function, typically GPIO port	00
01	First alternate function	
10	Second alternate function	
11	Third alternate function	

Table D.2- Pin function select register 3

PINSEL3	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value	LED
19:18	P1.25	GPIO Port 1.25	MCOA1	Reserved	MAT1.1	00	
21:20	P1.26	GPIO Port 1.26	MCOB1	PWM1.6	CAP0.0		
23:22	P1.27	GPIO Port 1.27	CLKOUT	USB_OVRCR	CAP0.1		
25:24	P1.28	GPIO Port 1.28	MCOA2	PCAP1.0	MAT0.0		LED[7]
27:26	P1.29	GPIO Port 1.29	MCOB2	PCAP1.1	MAT0.1		LED[6]
29:28	P1.30	GPIO Port 1.30	Reserved	V_{BUS}	AD0.4		
31:30	P1.31	GPIO Port 1.31	Reserved	SCK1	AD0.5		LED[5]

Table D.3- Pin function select register 4

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value	LED
1:0	P2.0	GPIO Port 2.0	PWM1.1	TXD1	Reserved		
3:2	P2.1	GPIO Port 2.1	PWM1.2	RXD1	Reserved		

5:4	P2.2	GPIO Port 2.2	PWM1.3	CTS1	Reserved		LED[4]
7:6	P2.3	GPIO Port 2.3	PWM1.4	DCD1	Reserved		LED[3]
9:8	P2.4	GPIO Port 2.4	PWM1.5	DSR1	Reserved	00	LED[2]
11:10	P2.5	GPIO Port 2.5	PWM1.6	DTR1	Reserved		LED[1]
13:12	P2.6	GPIO Port 2.6	PCAP1.6	RI1	Reserved		LED[0]
15:14	P2.7	GPIO Port 2.7	RD2	RTS1	Reserved		
17:16	P2.8	GPIO Port 2.8	TD2	TXD2	ENET_MDC		

b) Pin mode:

Each GPIO pin can be either input or output and can be configured to use a pull-up resistor, a pull-down resistor, or no resistor at all. "Fig.35" below, taken from NXP's LPC17xx datasheet, shows the general structure for a GPIO pin.

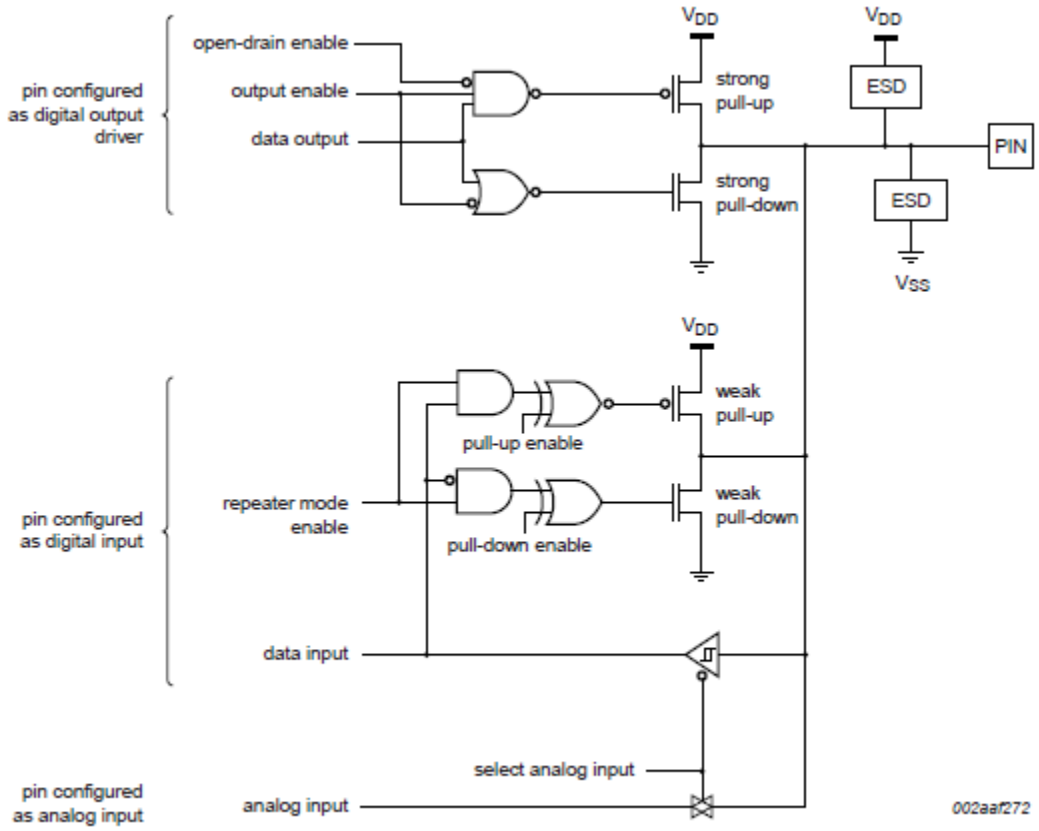


Fig 35. Standard I/O pin configuration with analog input

For configuration as output, the pull-up "resistor" can generate a "high" (or logic 1) state, while the pull-down "resistor" can generate a "low" (or logic 0) state.

There is a two bit field in the register named **PINMODEx** that determines the GPIO pin configuration. The PINMODE registers control the input mode of all ports. This includes the use of the on-chip pull-up/pull-down resistor feature and a special open-drain operating mode. The on-chip pull-up/pull-down resistor can be selected for every port pin regardless of the function on this pin with the exception of the I2C pins for the I2C0 interface and the USB pins. The next table shows how the configuration is done.

Table 76. Pin Mode Select register Bits

PINMODE0 to PINMODE9 Values	Function	Value after Reset
00	Pin has an on-chip pull-up resistor enabled.	00
01	Repeater mode (see text below).	
10	Pin has neither pull-up nor pull-down resistor enabled.	
11	Pin has an on-chip pull-down resistor enabled.	

For more information, refer to tables 87-93 of the LPC17xx User manual.

c) Using the GPIO:

The following table lists the registers associated with GPIOs. When using a given pin as I/O we write and read from these registers' individual bits.

Table 101. GPIO register map (local bus accessible registers - enhanced GPIO features)

Generic Name	Description	Access	Reset value[1]	PORTn Register Name & Address
FIODIR	Fast GPIO Port Direction control register. This register individually controls the direction of each port pin.	R/W	0	FIO0DIR - 0x2009 C000 FIO1DIR - 0x2009 C020 FIO2DIR - 0x2009 C040 FIO3DIR - 0x2009 C060 FIO4DIR - 0x2009 C080
FIOMASK	Fast Mask register for port. Writes, sets, clears, and reads to port (done via writes to FIOPIN, FIOSET, and FIOCLR, and reads of FIOPIN) alter or return only the bits enabled by zeros in this register.	R/W	0	FIO0MASK - 0x2009 C010 FIO1MASK - 0x2009 C030 FIO2MASK - 0x2009 C050 FIO3MASK - 0x2009 C070 FIO4MASK - 0x2009 C090
FIOPIN	Fast Port Pin value register using FIOMASK. The current state of digital port pins can be read from this register, regardless of pin direction or alternate function selection (as long as pins are not configured as an input to ADC). The value read is masked by ANDing with inverted FIOMASK. Writing to this register places corresponding values in all bits enabled by zeros in FIOMASK. **Important:** if an FIOPIN register is read, its bit(s) masked with 1 in the FIOMASK register will be read as 0 regardless of the physical pin state.	R/W	0	FIO0PIN - 0x2009 C014 FIO1PIN - 0x2009 C034 FIO2PIN - 0x2009 C054 FIO3PIN - 0x2009 C074 FIO4PIN - 0x2009 C094
FIOSET	Fast Port Output Set register using FIOMASK. This register controls the state of output pins. Writing 1s produces highs at the corresponding port pins. Writing 0s has no effect. Reading this register returns the current contents of the port output register. Only bits enabled by 0 in FIOMASK can be altered.	R/W	0	FIO0SET - 0x2009 C018 FIO1SET - 0x2009 C038 FIO2SET - 0x2009 C058 FIO3SET - 0x2009 C078 FIO4SET - 0x2009 C098
FIOCLR	Fast Port Output Clear register using FIOMASK. This register controls the state of output pins. Writing 1s produces lows at the corresponding port pins. Writing 0s has no effect. Only bits enabled by 0 in FIOMASK can be altered.	WO	0	FIO0CLR - 0x2009 C01C FIO1CLR - 0x2009 C03C FIO2CLR - 0x2009 C05C FIO3CLR - 0x2009 C07C FIO4CLR - 0x2009 C09C

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

Selected GPIO pins can also be set to generate interrupts. The push button INT0 is connected to pin P2.10 of the LPC1768 microprocessor. This pin can be a source of external interrupts to the MCU. The table below shows different functionalities that can be assigned to P2.10 pin.

Table E.1 – Pin functions for P2.10

Bits 21:20 of **PINSEL4**	**Function**	**Value after reset**
00	GPIO P2.10 pin (default)	00
01	$\overline{EINT0}$	
10	NMI	
11	Reserved	

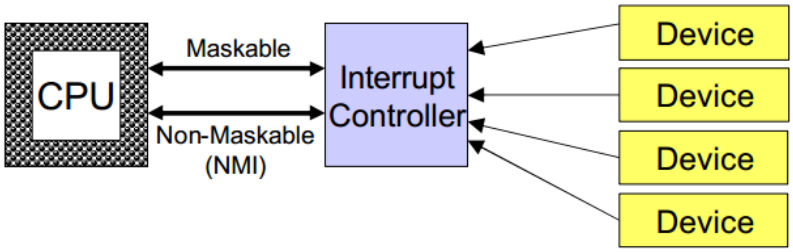
If you plan to use P2.10 as GPIO, then you should also enable this source of interrupt as described in section 9.5.6 of the LPC17xx user manual. Note that you can set the P2.10 pin to be sensitive to either the rising edge or the falling edge. More information on clearing the interrupt pending bit can be found in table 123 in section 9.5.6.1, page 139 of the user manual.

5. Polling vs. Interrupts

"Polling is like picking up your phone every few seconds to see if you have a call."

Polling the device usually means reading its status register every so often until the device's status changes to indicate that it has completed some request. It takes CPU time even when no requests are pending. However, it can be efficient if events arrive rapidly.

Interrupts are an alternative to polling. Each device is given a wire (interrupt line) that it can use to signal the processor. When an interrupt is signaled, the processor executes a routine called an interrupt handler to deal with the interrupt. This approach has no overhead when no requests are pending.



A nonmaskable interrupt (NMI) is an interrupt that cannot be ignored by standard interrupt masking techniques in the system. It is typically used to signal attention for non-recoverable hardware errors. On receipt of an NMI request, immediate execution of the NMI handler is guaranteed unless the system is completely locked up. NMI is very important for many safety-critical applications.

6. References

[1] Block diagram of MCB1700 board;
http://www.keil.com/support/man/docs/mcb1700/mcb1700_to_block.htm
[2] Cortex-M3 based microcontrollers with Ethernet, USB, CAN and 12-bit ADC;
http://www.nxp.com/documents/leaflet/75016846.pdf
[3] LPC17xx User Manual;

http://www.nxp.com/documents/user_manual/UM10360.pdf
[4] Schematic Diagram for the MCB1700 board;
http://www.keil.com/mcb1700/mcb1700-schematics.pdf
[5] ECE 222 lab manual at UWaterloo; https://ece.uwaterloo.ca/~ece222/ECE222_Manual_DRAFT.pdf
[6] ECE 455 lab manual at UWaterloo; https://ece.uwaterloo.ca/~ece455/lab_manual.pdf