Laboratorio 8

R. Ferrero, P. Bernardi

Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)

Torino - Italy

NXP LPC1768

- Il materiale dell'esercitazione comprende:
  - LPC176x_USER_MANUAL.pdf
  - HY-LandTiger_BOARD_SCHEMATIC.pdf
  - progetto Sample

Diagramma a blocchi del SoC (p.9)

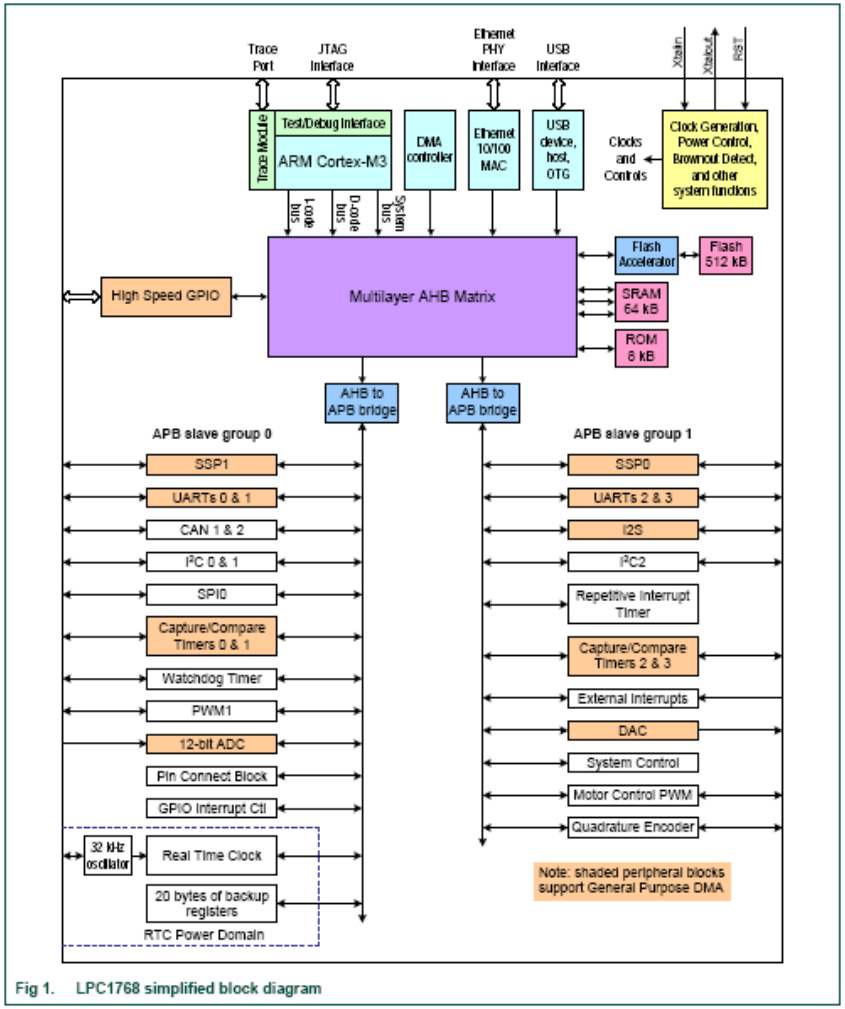


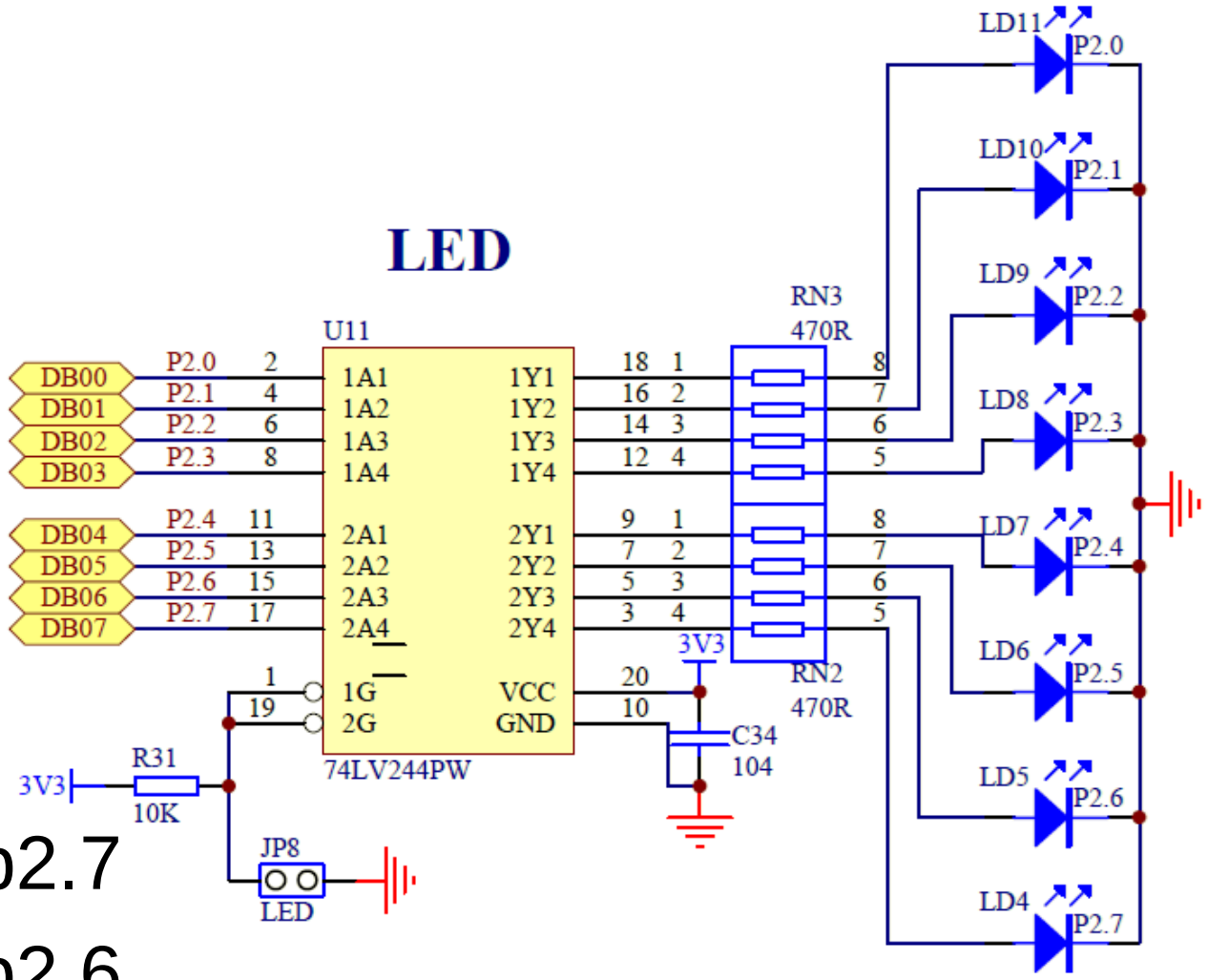
Fig 1. LPC1768 simplified block diagram

Trace Port
JTAG Interface
Ethernet PHY Interface
USB Interface
Xtalin
Xtalout
RST
Trace Module
Test/Debug Interface
ARM Cortex-M3
DMA controller
Ethernet 10/100 MAC
USB device, host, OTG
Clocks and Controls
Clock Generation, Power Control, Brownout Detect, and other system functions
I-code bus
D-code bus
System bus
High Speed GPIO
Multilayer AHB Matrix
Flash Accelerator
Flash 512 kB
SRAM 64 kB
ROM 8 kB
AHB to APB bridge
AHB to APB bridge
APB slave group 0
SSP1
UARTs 0 & 1
CAN 1 & 2
I²C 0 & 1
SPI0
Capture/Compare Timers 0 & 1
Watchdog Timer
PWM1
12-bit ADC
Pin Connect Block
GPIO Interrupt Ctl
32 kHz oscillator
Real Time Clock
20 bytes of backup registers
RTC Power Domain
APB slave group 1
SSP0
UARTs 2 & 3
I2S
I²C2
Repetitive Interrupt Timer
Capture/Compare Timers 2 & 3
External Interrupts
DAC
System Control
Motor Control PWM
Quadrature Encoder
Note: shaded peripheral blocks support General Purpose DMA

Progetto Sample: contenuto

- file di avvio del chip:
  - startup_LPC17xx.s
- librerie di sistema
  - core_cm3.c
  - system_LPC17xx.c
- Librerie di alcuni periferici:
  - *peripheral*.h /* prototipi */
  - lib_*peripheral*.c /* funzioni di base */
  - IRQ_*peripheral*.c /* interrupt service routine */
  - funct_*peripheral*.c /* funzioni utente */

Led



- Led4 -> p2.7
- Led5 -> p2.6
- Led11-> p2.0

Pin connect block (pag. 119)

Table 84. Pin function select register 4 (PINSEL4 - address 0x4002 C010) bit description

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P2.0	GPIO Port 2.0	PWM1.1	TXD1	Reserved	00
3:2	P2.1	GPIO Port 2.1	PWM1.2	RXD1	Reserved	00
5:4	P2.2	GPIO Port 2.2	PWM1.3	CTS1	Reserved [2]	00
7:6	P2.3	GPIO Port 2.3	PWM1.4	DCD1	Reserved [2]	00
9:8	P2.4	GPIO Port 2.4	PWM1.5	DSR1	Reserved [2]	00
11:10	P2.5	GPIO Port 2.5	PWM1.6	DTR1	Reserved [2]	00
13:12	P2.6	GPIO Port 2.6	PCAP1.0	RI1	Reserved [2]	00
15:14	P2.7	GPIO Port 2.7	RD2	RTS1	Reserved	00
17:16	P2.8	GPIO Port 2.8	TD2	TXD2	ENET_MDC	00
19:18	P2.9	GPIO Port 2.9	USB_CONNECT	RXD2	ENET_MDIO	00
21:20	P2.10	GPIO Port 2.10	$\overline{\text{EINT0}}$	NMI	Reserved	00
23:22	P2.11 [1]	GPIO Port 2.11	$\overline{\text{EINT1}}$	Reserved	I2STX_CLK	00
25:24	P2.12 [1]	GPIO Port 2.12	$\overline{\text{EINT2}}$	Reserved	I2STX_WS	00
27:26	P2.13 [1]	GPIO Port 2.13	$\overline{\text{EINT3}}$	Reserved	I2STX_SDA	00
31:28	-	Reserved	Reserved	Reserved	Reserved	0

General purpose I/O (pag. 131)

Table 102. 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.

FIODIR (pag. 132)

Table 104. Fast GPIO port Direction register FIO0DIR to FIO4DIR - addresses 0x2009 C000 to 0x2009 C080) bit description

Bit	Symbol	Value	Description	Reset value
31:0	FIO0DIR FIO1DIR FIO2DIR FIO3DIR FIO4DIR		Fast GPIO Direction PORTx control bits. Bit 0 in FIOxDIR controls pin Px.0, bit 31 in FIOxDIR controls pin Px.31.	0x0
		0	Controlled pin is input.	
		1	Controlled pin is output.	

Esercizio 1

- Aggiungere al file led.h il prototipo `void led4and11_On(void);`
- Aggiungere al gruppo ‘led’ il file funct_led.c
- Implementare in funct_led.c la funzione `led4and11_On(void)`, che accende i led 4 e 11 agendo sul registro FIOSET.
- Nota: lo stato (acceso/spento) degli altri led non deve essere modificato.
- Testare la funzione richiamandola dal main.

Esercizio 2

- Aggiungere al file led.h il prototipo `void led4_Off(void);`
- Implementare in funct_led.c la funzione `led4_Off(void)`, che spegne il led 4 agendo sul registro FIOCLR.
- Nota: lo stato (acceso/spento) degli altri led non deve essere modificato.
- Testare la funzione richiamandola dal main.

Esercizio 3

- Aggiungere al file led.h il prototipo `void ledEvenOn_OddOf(void);`
- Implementare in funct_led.c la funzione `ledEvenOn_OddOf(void)`, che accende i led di indice pari e spegne quelli di indice dispari, agendo sul registro FIOPIN.
- Testare la funzione richiamandola dal main.

Esercizio 4

- Aggiungere al file led.h il prototipo `void LED_On(unsigned int num);`
- Implementare in funct_led.c la funzione `void LED_On(unsigned int num)` che accende il led passato come parametro:
  - num = 0 -> led 4
  - num = 1 -> led 5
  - num = 7 -> led 11
- Testare la funzione richiamandola dal main.

Esercizio 5

- Aggiungere al file led.h il prototipo `void LED_Off(unsigned int num);`
- Implementare in funct_led.c la funzione `void LED_Off(unsigned int num)` che spegne il led passato come parametro:
  - num = 0 -> led 4
  - num = 1 -> led 5
  - num = 7 -> led 11
- Testare la funzione richiamandola dal main.