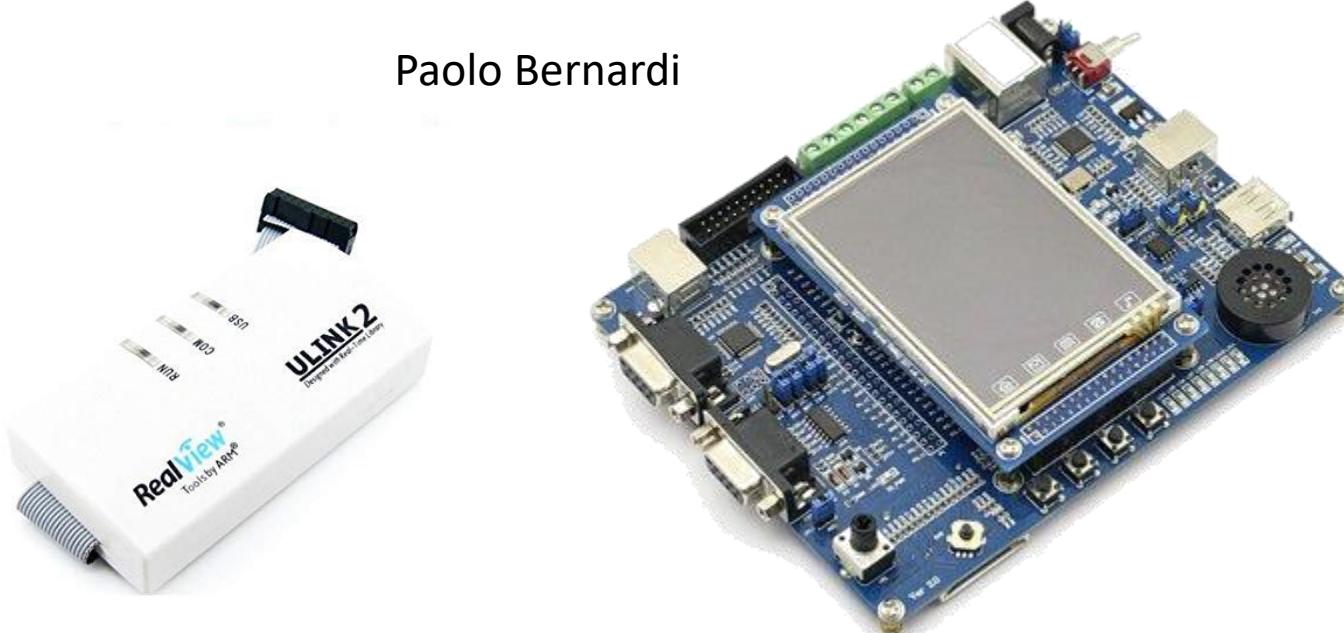


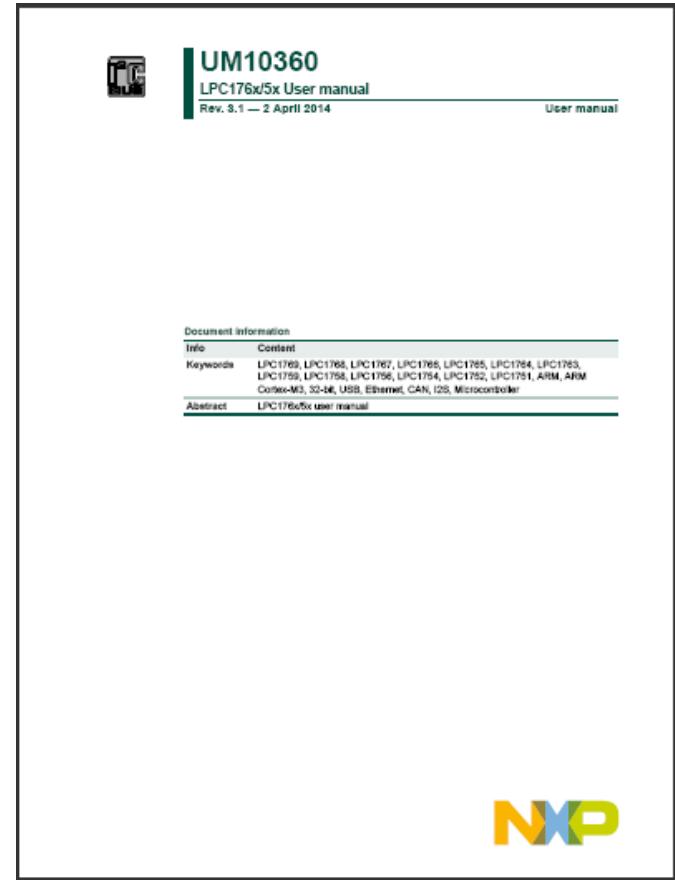
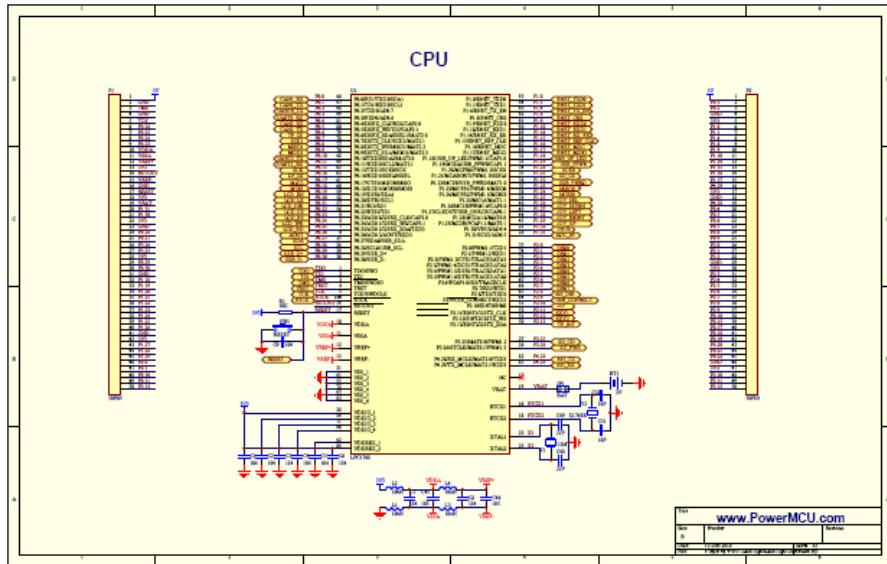
NXP LPC1768 fast usage guide

Paolo Bernardi



Available resources

- *Reference manual:*
 - LPC176x_USER_MANUAL.pdf
- Schematic:
 - HY-LandTiger_BOARD_SCHEMATIC.pdf
- Example:
 - Sample project



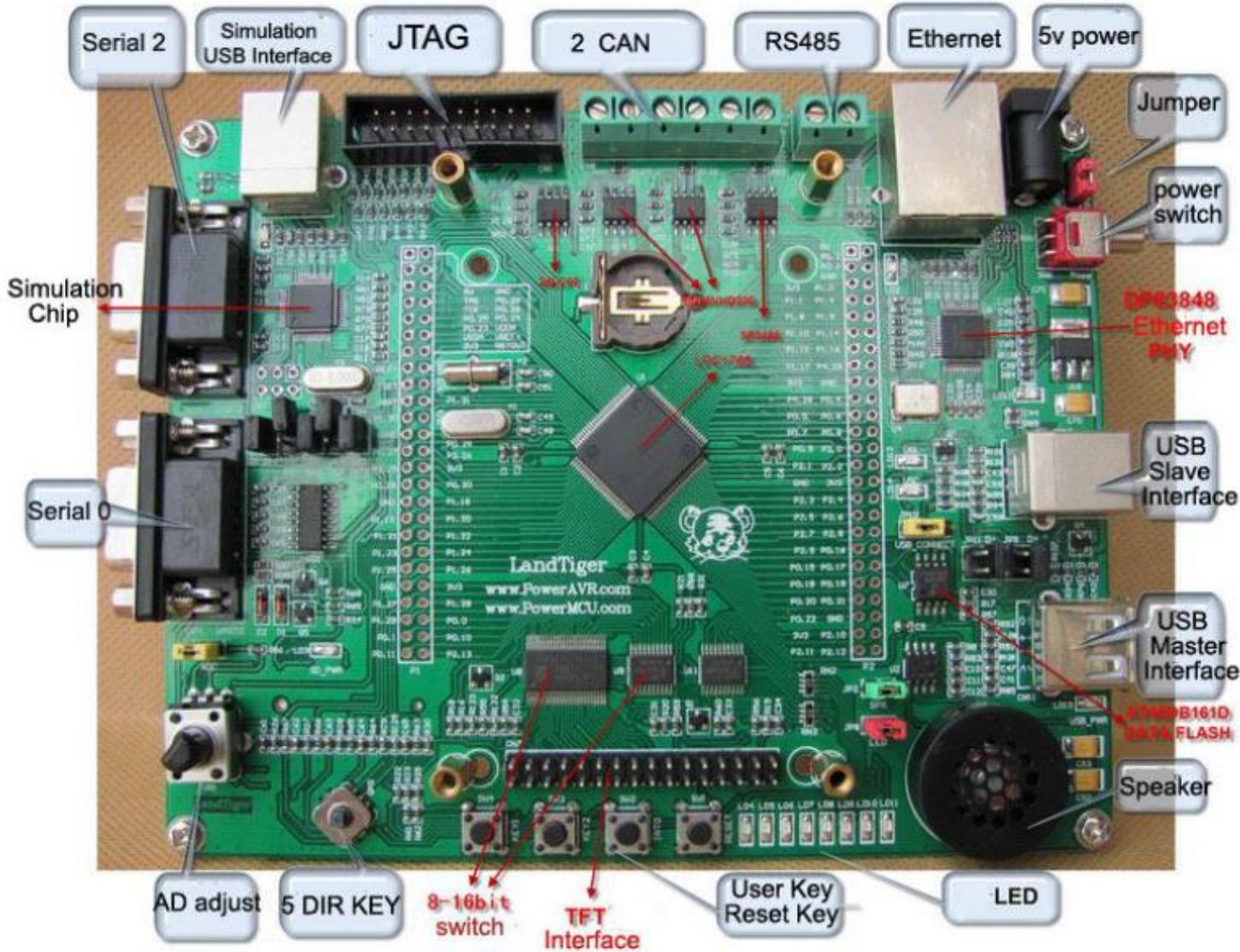
Composition of the chip

- ARM 32-bit Cortex-M3 Microcontroller with MPU, CPU clock up to 100MHz
- 512kB on-chip Flash ROM with enhanced Flash Memory Accelerator,
- Four 32-bit **Timers** with capture/compare, Standard PWM Timer block,
- 64kB RAM, **Nested Vectored Interrupt Controller**,
- Eight channel General purpose DMA controller, AHB Matrix, APB,
- **System Tick Timer**, Repetitive Interrupt Timer, Brown-out detect circuit

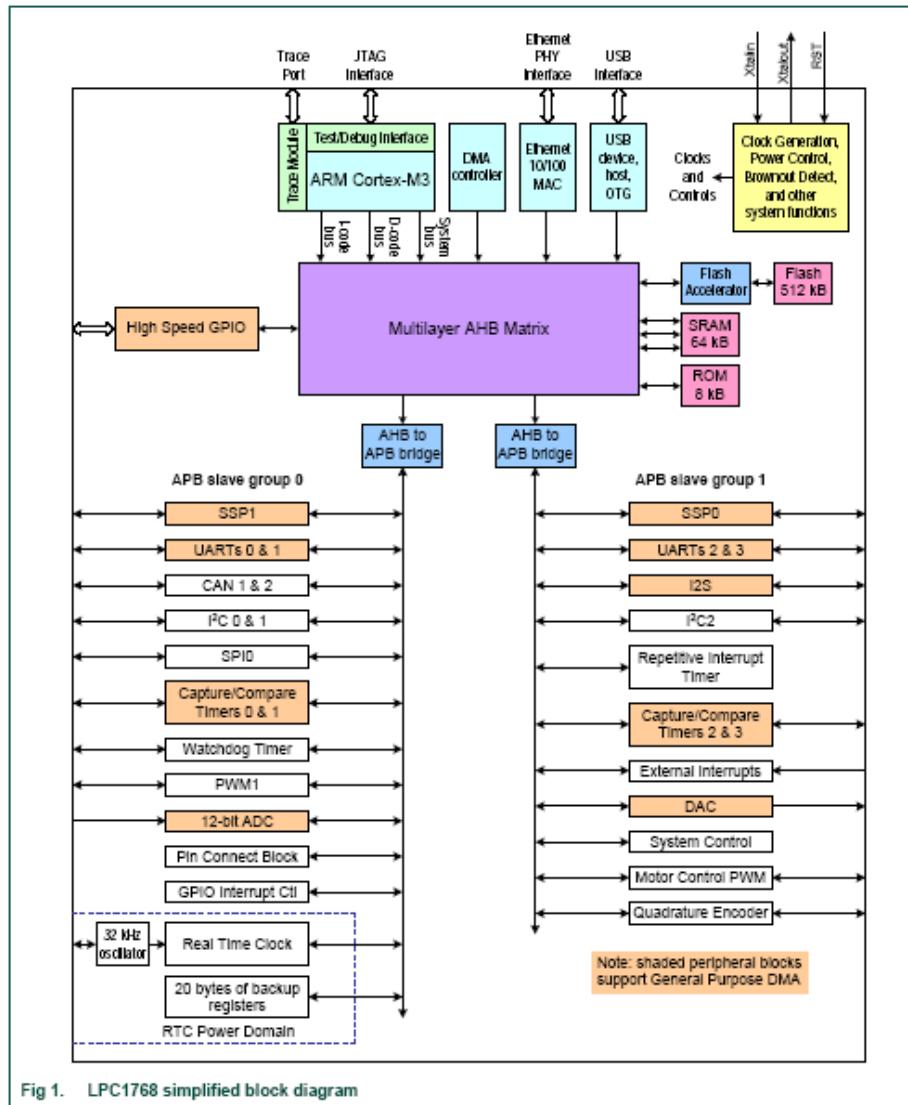
Composition of the chip (II)

- Ethernet 10/100 MAC with RMII interface and dedicated DMA,
- USB 2.0 full-speed Device controller and Host/OTG controller with DMA,
- CAN 2.0B with two channels, **Four UARTs**, one with full Modem interface,
- Three I²C serial interfaces, Three SPI/SSP serial interfaces, I²S interface,
- **General purpose I/O pins, 12-bit ADC with 8 channels, 10-bit DAC**
- Motor control PWM for three-phase Motor control, Quadrature Encoder
- Watchdog Timer, Real Time Clock with optional Battery backup
- Power-On Reset, **Power Management Unit**, Wakeup Interrupt Controller
- Crystal oscillator, 4MHz internal RC oscillator, **PLL**

Board composition



Block diagram of the Chip (user manual pg.9)

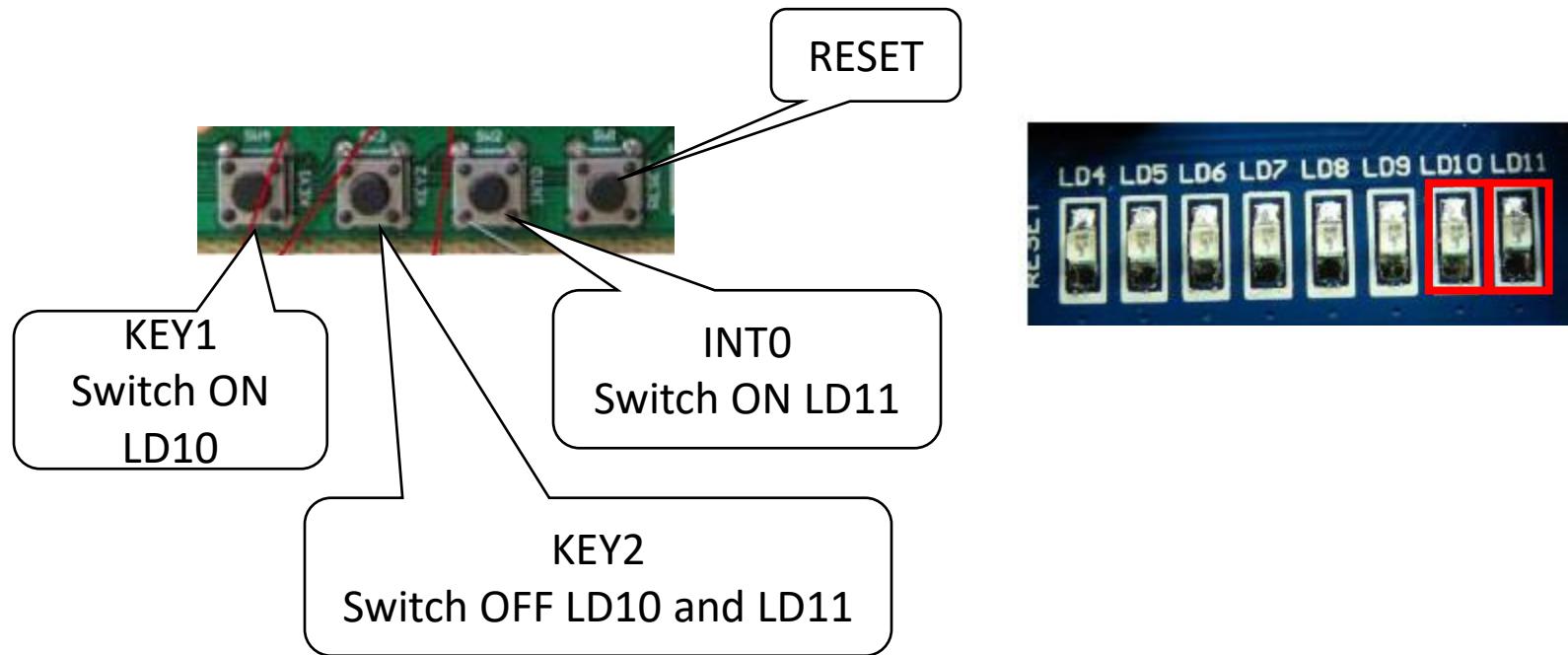


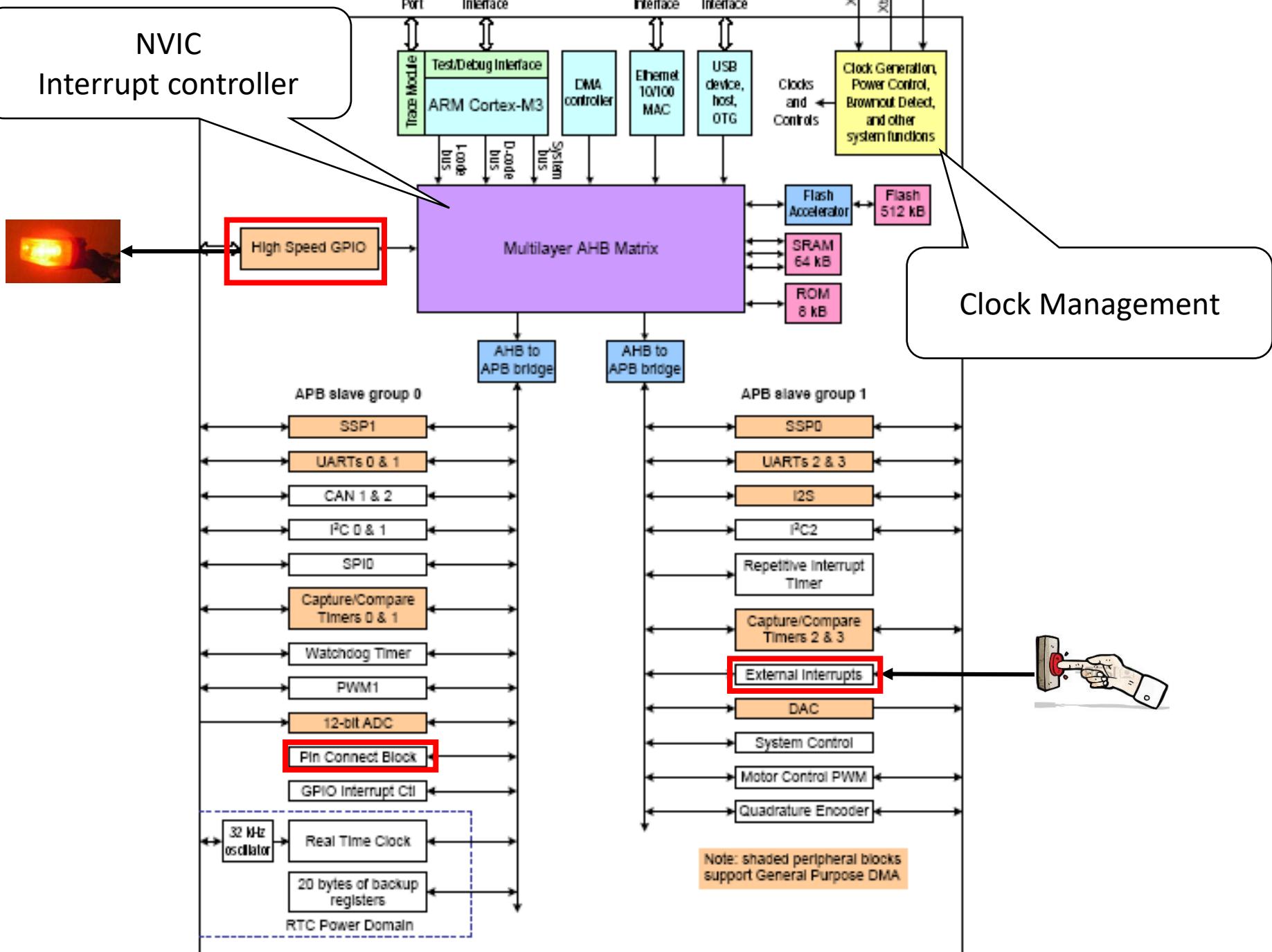
Sample project

- The following slides provide a top-down view of the Sample project which includes many files
- It is a base project that includes
 - boot of the chip
 - startup_LPC17xx.s
 - includes libraries
 - core_cm3.c
 - system_LPC17xx.h & system_LPC17xx.c
- The libraries of some peripheral core according to the following convention
 - *peripheral.h* /* prototypes */
 - *lib_peripheral.c* /* base functions */
 - *IRQ_peripheral.c* /* interrupt service routines */
 - *funct_peripheral.c* /* advanced user functions */

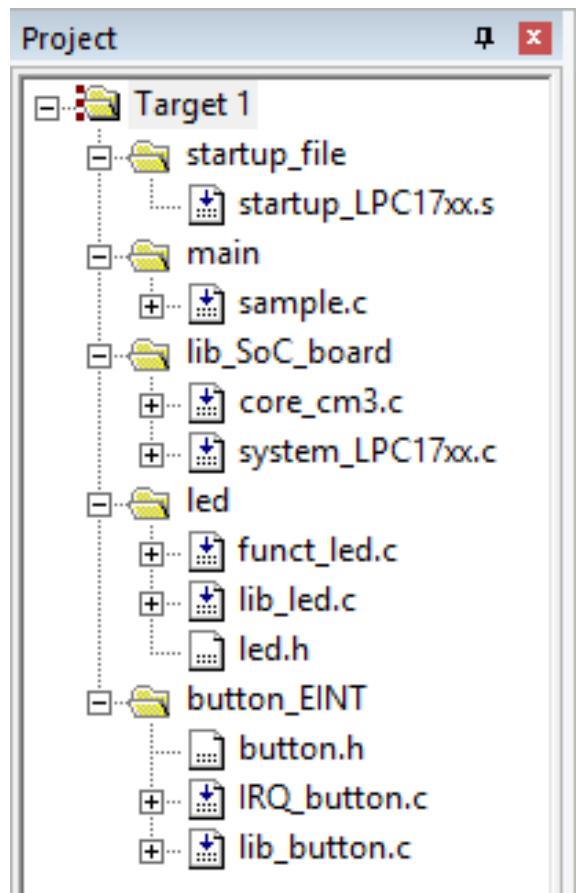
Sample project (II)

- The functionality of the project is the following
 - Pression of button INT0 provokes the LD11 to switch ON
 - Pression of button KEY1 provokes the LD10 to switch ON
 - Pression of button KEY2 provokes the LD10 and LD11 to switch OFF





General view of the project



- The set of *lib_SoC_board* files provides all information needed to
 - Address system-on-chip peripheral registers
 - Setup clock distribution of the system
 - Manage interrupt enable and priority
- led and button_EINT libraries are
 - Setting the system to react to the CPU requests
 - Handling external interrupts
 - Providing functions to simplify programmers work.

lib_SoC_board libraries

lpc17xx.h, sys_lpc17xx.c and core_cm3.h

lib_SoC_board libraries lpc17xx.h, sys_lpc17xx.c and core_cm3.h

- Defines constants that store main memory addresses at C language level for accessing peripheral registers

```
915  **** Peripherals memory map ****
916  /*
917  ****
918  /* Base addresses
919  #define LPC_FLASH_BASE          (0x00000000UL)
920  #define LPC_RAM_BASE            (0x10000000UL)
921  #ifdef __LPC17XX_REV00
922  #define LPC_AHBRAM0_BASE        (0x20000000UL)
923  #define LPC_AHBRAM1_BASE        (0x20004000UL)
924  #else
925  #define LPC_AHBRAM0_BASE        (0x2007C000UL)
926  #define LPC_AHBRAM1_BASE        (0x20080000UL)
927  #endif
928  #define LPC_GPIO_BASE           (0x2009C000UL)
929  #define LPC_APBO_BASE           (0x40000000UL)
930  #define LPC_APB1_BASE           (0x40080000UL)
931  #define LPC_AHB_BASE             (0x50000000UL)
932  #define LPC_CM3_BASE             (0xE0000000UL)
933
```

lib_SoC_board libraries lpc17xx.h, sys_lpc17xx.h

```

915  **** Peripherals memory map ****
916  /*
917  **** Base addresses ****
918  */
919 #define LPC_FLASH_BASE      (0x00000000UL)
920 #define LPC_RAM_BASE        (0x10000000UL)
921 #ifdef __LPC17XX_REV00
922 #define LPC_AHBRAM0_BASE    (0x20000000UL)
923 #define LPC_AHBRAM1_BASE    (0x20004000UL)
924 #else

```

Table 3. LPC176x/5x memory usage and details

Address range	General Use	Address range details and description	
0x0000 0000 to 0x1FFF FFFF	On-chip non-volatile memory	0x0000 0000 - 0x0007 FFFF	For devices with 512 kB of flash memory.
		0x0000 0000 - 0x0003 FFFF	For devices with 256 kB of flash memory.
		0x0000 0000 - 0x0001 FFFF	For devices with 128 kB of flash memory.
		0x0000 0000 - 0x0000 FFFF	For devices with 64 kB of flash memory.
		0x0000 0000 - 0x0000 7FFF	For devices with 32 kB of flash memory.
	On-chip SRAM	0x1000 0000 - 0x1000 7FFF	For devices with 32 kB of local SRAM.
		0x1000 0000 - 0x1000 3FFF	For devices with 16 kB of local SRAM.
		0x1000 0000 - 0x1000 1FFF	For devices with 8 kB of local SRAM.
	Boot ROM	0x1FFF 0000 - 0x1FFF 1FFF	8 kB Boot ROM with flash services.
0x2000 0000 to 0x3FFF FFFF	On-chip SRAM (typically used for peripheral data)	0x2007 C000 - 0x2007 FFFF	AHB SRAM - bank 0 (16 kB), present on devices with 32 kB or 64 kB of total SRAM.
		0x2008 0000 - 0x2008 3FFF	AHB SRAM - bank 1 (16 kB), present on devices with 64 kB of total SRAM.
	GPIO	0x2009 C000 - 0x2009 FFFF	GPIO.
	APB Peripherals	0x4000 0000 - 0x4007 FFFF	APB0 Peripherals, up to 32 peripheral blocks, 16 kB each.
		0x4008 0000 - 0x400F FFFF	APB1 Peripherals, up to 32 peripheral blocks, 16 kB each.
0xE000 0000 to 0xE00F FFFF	AHB peripherals	0x5000 0000 - 0x501F FFFF	DMA Controller, Ethernet interface, and USB interface.
	Cortex-M3 Private Peripheral Bus	0xE000 0000 - 0xE00F FFFF	Cortex-M3 related functions, includes the NVIC and System Tick Timer.

lib_SoC_board libraries

lpc17xx.h, sys_lpc17xx.c and core_cm3.h

- For every peripheral block, it defines a start address based on main memory constant

```
929 #define LPC_APB0_BASE          (0x400000000UL)
930 #define LPC_APB1_BASE          (0x400800000UL)
931 #define LPC_AHB_BASE           (0x500000000UL)
932 #define LPC_CM3_BASE           (0xE00000000UL)
933
934 /* APB0 peripherals */                                 */
935 #define LPC_WDT_BASE            (LPC_APB0_BASE + 0x00000)
936 #define LPC_TIM0_BASE           (LPC_APB0_BASE + 0x04000)
937 #define LPC_TIM1_BASE           (LPC_APB0_BASE + 0x08000)
938 #define LPC_UART0_BASE          (LPC_APB0_BASE + 0x0C000)
939 #define LPC_UART1_BASE          (LPC_APB0_BASE + 0x10000)
940 #define LPC_PWM1_BASE           (LPC_APB0_BASE + 0x18000)
941 #define LPC_I2C0_BASE           (LPC_APB0_BASE + 0x1C000)
942 #define LPC_SPI_BASE            (LPC_APB0_BASE + 0x20000)
943 #define LPC_RTC_BASE             (LPC_APB0_BASE + 0x24000)
944 #define LPC_GPIOINT_BASE         (LPC_APB0_BASE + 0x28080)
945 #define LPC_PINCON_BASE          (LPC_APB0_BASE + 0x2C000)  */
946 #define LPC_SSPI_BASE            (LPC_APB0_BASE + 0x30000)
947 #define LPC_ADC_BASE             (LPC_APB0_BASE + 0x34000)
948 #define LPC_CANAF_RAM_BASE       (LPC_APB0_BASE + 0x38000)
949 #define LPC_CANAF_BASE           (LPC_APB0_BASE + 0x3C000)
950 #define LPC_CANCR_BASE          (LPC_APB0_BASE + 0x40000)
951 #define LPC_CAN1_BASE            (LPC_APB0_BASE + 0x44000)
```

lib_SoC_board libraries
`lpc17xx.h`, `sys_lpc17xx.c`

Table 4. APB0 peripherals and base addresses

APB0 peripheral	Base address	Peripheral name	BASE
0	0x4000 0000	Watchdog Timer	AM_BASE
1	0x4000 4000	Timer 0	ASE
2	0x4000 8000	Timer 1	ASE
3	0x4000 C000	UART0	SE
4	0x4001 0000	UART1	SE
5	0x4001 4000	reserved	
6	0x4001 8000	PWM1	
7	0x4001 C000	I ² C0	
8	0x4002 0000	SPI	
9	0x4002 4000	RTC	
10	0x4002 8000	GPIO interrupts	
11	0x4002 C000	Pin Connect Block	
12	0x4003 0000	SSP1	
13	0x4003 4000	ADC	
14	0x4003 8000	CAN Acceptance Filter RAM	
15	0x4003 C000	CAN Acceptance Filter Registers	
16	0x4004 0000	CAN Common Registers	
17	0x4004 4000	CAN Controller 1	
18	0x4004 8000	CAN Controller 2	
19 to 22	0x4004 C000 to 0x4005 8000	reserved	
23	0x4005 C000	I ² C1	
24 to 31	0x4006 0000 to 0x4007 C000	reserved	

934	/* APB0 peripherals	
935	#define LPC_WDT_BASE	(LPC_APB0_BASE + 0x00000)
936	#define LPC_TIM0_BASE	(LPC_APB0_BASE + 0x04000)
937	#define LPC_TIM1_BASE	(LPC_APB0_BASE + 0x08000)
938	#define LPC_UART0_BASE	(LPC_APB0_BASE + 0x0C000)
939	#define LPC_UART1_BASE	(LPC_APB0_BASE + 0x10000)
940	#define LPC_PWM1_BASE	(LPC_APB0_BASE + 0x18000)
941	#define LPC_I2C0_BASE	(LPC_APB0_BASE + 0x1C000)
942	#define LPC_SPI_BASE	(LPC_APB0_BASE + 0x20000)
943	#define LPC_RTC_BASE	(LPC_APB0_BASE + 0x24000)
	BASE	(LPC_APB0_BASE + 0x28080)
	BASE	(LPC_APB0_BASE + 0x2C000)
	SE	(LPC_APB0_BASE + 0x30000)
	I	(LPC_APB0_BASE + 0x34000)
	AM_BASE	(LPC_APB0_BASE + 0x38000)
	ASE	(LPC_APB0_BASE + 0x3C000)
	ASE	(LPC_APB0_BASE + 0x40000)
	SE	(LPC_APB0_BASE + 0x44000)
	SE	(LPC_APB0_BASE + 0x48000)
	SE	(LPC_APB0_BASE + 0x5C000)

lib_SoC_board libraries

lpc17xx.h, sys_lpc17xx.c and core_cm3.h

- Address definition of each single peripheral block

```
934 /* APB0 peripherals
935 #define LPC_WDT_BASE (LPC_APB0_BASE + 0x00000)
936 #define LPC_TIM0_BASE (LPC_APB0_BASE + 0x04000)
937 #define LPC_TIM1_BASE (LPC_APB0_BASE + 0x08000)
938 #define LPC_UART0_BASE (LPC_APB0_BASE + 0x0C000)
939 #define LPC_UART1_BASE (LPC_APB0_BASE + 0x10000)
940 #define LPC_PWM1_BASE (LPC_APB0_BASE + 0x18000)
941 #define LPC_I2C0_BASE (LPC_APB0_BASE + 0x1C000)
942 #define LPC_SPI_BASE (LPC_APB0_BASE + 0x20000)
943 #define LPC_RTC_BASE (LPC_APB0_BASE + 0x24000)
944 #define LPC_GPIOINT_BASE (LPC_APB0_BASE + 0x28080)
945 #define LPC_PINCON_BASE (LPC_APB0_BASE + 0x2C000) -----
946 #define LPC_SSP1_BASE (LPC_APB0_BASE + 0x30000)
947 #define LPC_ADC_BASE (LPC_APB0_BASE + 0x34000)
948 #define LPC_CANAF_RAM_BASE (LPC_APB0_BASE + 0x38000)
949 #define LPC_CANAF_BASE (LPC_APB0_BASE + 0x3C000)
950 #define LPC_CANCR_BASE (LPC_APB0_BASE + 0x40000)
951 #define LPC_CAN1_BASE (LPC_APB0_BASE + 0x44000)
952 #define LPC_CAN2_BASE (LPC_APB0_BASE + 0x48000)
953 #define LPC_I2C1_BASE (LPC_APB0_BASE + 0x5C000)
...  
-----
```

lib_SoC_board libraries lpc17xx.h, sys_lpc17xx.c and core_cm3.h

Accessible name
from C code

Cast as a pointer to
the relative type

Address of every
specific SoC resource

```
989  **** Peripherals Declaration ****
990  /*
991  ****
992 #define LPC_SC          ((LPC_SC_TypeDef *) LPC_SC_BASE)
993 #define LPC_PINCON        ((LPC_PINCON_TypeDef *) LPC_PINCON_BASE)
994 #define LPC_GPIO0          ((LPC_GPIO_TypeDef *) LPC_GPIO0_BASE)
995 #define LPC_GPIO1          ((LPC_GPIO_TypeDef *) LPC_GPIO1_BASE)
996 #define LPC_GPIO2          ((LPC_GPIO_TypeDef *) LPC_GPIO2_BASE)
997 #define LPC_GPIO3          ((LPC_GPIO_TypeDef *) LPC_GPIO3_BASE)
998 #define LPC_GPIO4          ((LPC_GPIO_TypeDef *) LPC_GPIO4_BASE)
999 #define LPC_WDT           ((LPC_WDT_TypeDef *) LPC_WDT_BASE)
1000 #define LPC_TIM0          ((LPC_TIM_TypeDef *) LPC_TIM0_BASE)
1001 #define LPC_TIM1          ((LPC_TIM_TypeDef *) LPC_TIM1_BASE)
1002 #define LPC_TIM2          ((LPC_TIM_TypeDef *) LPC_TIM2_BASE)
1003 #define LPC_TIM3          ((LPC_TIM_TypeDef *) LPC_TIM3_BASE)
1004 #define LPC_RIT            ((LPC_RIT_TypeDef *) LPC_RIT_BASE)
1005 #define LPC_UART0          ((LPC_UART_TypeDef *) LPC_UART0_BASE)
1006 #define LPC_UART1          ((LPC_UART1_TypeDef *) LPC_UART1_BASE)
1007 #define LPC_UART2          ((LPC_UART_TypeDef *) LPC_UART2_BASE)
1008 #define LPC_UART3          ((LPC_UART_TypeDef *) LPC_UART3_BASE)
1009 #define LPC_PWM1          ((LPC_PWM_TypeDef *) LPC_PWM1_BASE)
```

C libraries

lpc17xx.h, sys_lpc17xx.c and core_cm3.h

```
160  /*----- Pin Connect Block (PINCON) -----*/
161  /** @brief Pin Connect Block (PINCON) register structure definition */
162  typedef struct
163  {
164      __IO uint32_t PINSEL0;
165      __IO uint32_t PINSEL1;
166      __IO uint32_t PINSEL2;
167      __IO uint32_t PINSEL3;
168      __IO uint32_t PINSEL4; [Red Box]
169      __IO uint32_t PINSEL5;
170      __IO uint32_t PINSEL6;
171      __IO uint32_t PINSEL7;
172      __IO uint32_t PINSEL8;
173      __IO uint32_t PINSEL9;
174      __IO uint32_t PINSEL10;
175      uint32_t RESERVED0[5];
176      __IO uint32_t PINMODE0;
177      __IO uint32_t PINMODE1;
178      __IO uint32_t PINMODE2;
179      __IO uint32_t PINMODE3;
180      __IO uint32_t PINMODE4;
181      __IO uint32_t PINMODE5;
182      __IO uint32_t PINMODE6;
183      __IO uint32_t PINMODE7;
184      __IO uint32_t PINMODE8;
185      __IO uint32_t PINMODE9;
186      __IO uint32_t PINMODE_OD0;
187      __IO uint32_t PINMODE_OD1;
188      __IO uint32_t PINMODE_OD2;
189      __IO uint32_t PINMODE_OD3;
190      __IO uint32_t PINMODE_OD4;
191      __IO uint32_t T2CPADCFG;
192 } LPC_PINCON_TypeDef;
```

C libraries

lpc17xx.h, sys_lpc17xx.c and core_cm3.h

Table 79. Pin Connect Block Register Map

Name	Description	Access	Reset Value ^[1]	Address
PINSEL0	Pin function select register 0.	R/W	0	0x4002 C000
PINSEL1	Pin function select register 1.	R/W	0	0x4002 C004
PINSEL2	Pin function select register 2.	R/W	0	0x4002 C008
PINSEL3	Pin function select register 3.	R/W	0	0x4002 C00C
PINSEL4	Pin function select register 4	R/W	0	0x4002 C010
PINSEL7	Pin function select register 7	R/W	0	0x4002 C01C
PINSEL8	Pin function select register 8	R/W	0	0x4002 C020
PINSEL9	Pin function select register 9	R/W	0	0x4002 C024
PINSEL10	Pin function select register 10	R/W	0	0x4002 C028
PINMODE0	Pin mode select register 0	R/W	0	0x4002 C040
PINMODE1	Pin mode select register 1	R/W	0	0x4002 C044
PINMODE2	Pin mode select register 2	R/W	0	0x4002 C048
PINMODE3	Pin mode select register 3.	R/W	0	0x4002 C04C
PINMODE4	Pin mode select register 4	R/W	0	0x4002 C050
PINMODE5	Pin mode select register 5	R/W	0	0x4002 C054
PINMODE6	Pin mode select register 6	R/W	0	0x4002 C058
PINMODE7	Pin mode select register 7	R/W	0	0x4002 C05C
PINMODE9	Pin mode select register 9	R/W	0	0x4002 C064
PINMODE_OD0	Open drain mode control register 0	R/W	0	0x4002 C068
PINMODE_OD1	Open drain mode control register 1	R/W	0	0x4002 C06C
PINMODE_OD2	Open drain mode control register 2	R/W	0	0x4002 C070
PINMODE_OD3	Open drain mode control register 3	R/W	0	0x4002 C074
PINMODE_OD4	Open drain mode control register 4	R/W	0	0x4002 C078
I2CPADCFG	I ² C Pin Configuration register	R/W	0	0x4002 C07C

```

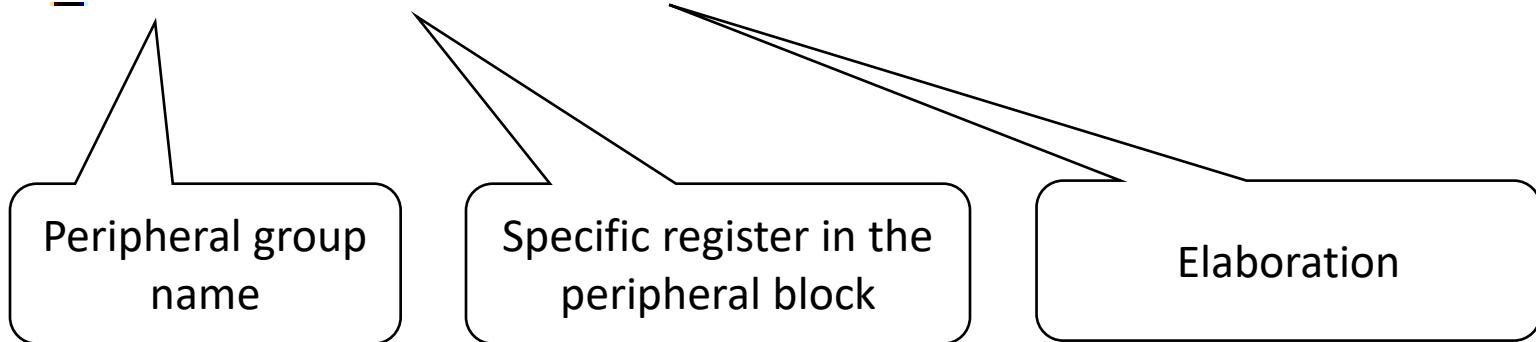
160  /*----- Pin Connect Blc
161  /** @brief Pin Connect Block (P
162  typedef struct
163  {
164      __IO uint32_t PINSEL0; base
165      __IO uint32_t PINSEL1; +4
166      __IO uint32_t PINSEL2; +4
167      __IO uint32_t PINSEL3; +4
168      __IO uint32_t PINSEL4;
169      __IO uint32_t PINSEL5;
170      __IO uint32_t PINSEL6;
171      __IO uint32_t PINSEL7;
172      __IO uint32_t PINSEL8;
173      __IO uint32_t PINSEL9;
174      __IO uint32_t PINSEL10;
175      uint32_t RESERVED0[5];
176      __IO uint32_t PINMODE0;
177      __IO uint32_t PINMODE1;
178      __IO uint32_t PINMODE2;
179      __IO uint32_t PINMODE3;
180      __IO uint32_t PINMODE4;
181      __IO uint32_t PINMODE5;
182      __IO uint32_t PINMODE6;
183      __IO uint32_t PINMODE7;
184      __IO uint32_t PINMODE8;
185      __IO uint32_t PINMODE9;
186      __IO uint32_t PINMODE_OD0;
187      __IO uint32_t PINMODE_OD1;
188      __IO uint32_t PINMODE_OD2;
189      __IO uint32_t PINMODE_OD3;
190      __IO uint32_t PINMODE_OD4;
191      __IO uint32_t I2CPADCFG;
192 } LPC_PINCON_TypeDef;
```

C libraries

lpc17xx.h, sys_lpc17xx.c and core_cm3.h

- This setup permits to use simple symbols to address peripheral registers that would be much difficult to manage

```
LPC_PINCON->PINSEL4 |= 0xFFFF0000;
```



C libraries

lpc17xx.h, sys_lpc17xx.c and core_cm3.h

- Mainly devoted to initialize the system-on-chip clock frequencies

```
377 /*-----  
378  Define clocks  
379  -----*/  
380 #define XTAL      (12000000UL)      /* Oscillator frequency */  
381 #define OSC_CLK   (XTAL)           /* Main oscillator frequency */  
382 #define RTC_CLK   (32000UL)         /* RTC oscillator frequency */  
383 #define IRC_OSC   (4000000UL)        /* Internal RC oscillator frequency */  
384  
  
392 /**  
393  * Initialize the system  
394  *  
395  * @param  none  
396  * @return none  
397  *  
398  * @brief   Setup the microcontroller system.  
399  *          Initialize the System and update the SystemFrequency variable.  
400  */  
401 void SystemInit (void)  
402 {  
403 #if (CLOCK_SETUP)                  /* Clock Setup */  
404     LPC_SC->SCS      = SCS_Val;  
405     if (SCS_Val & (1 << 5)) {       /* If Main Oscillator is enabled */  
406         while ((LPC_SC->SCS & (1<<6)) == 0);/* Wait for Oscillator to be ready */  
407     }  
408 }
```

C libraries

lpc17xx.h, sys_lpc17xx.c and core_cm3.h

- The SystemInit() function is called from the main program

```
20  /*-----  
21   Main Program  
22  -----*/  
23 int main (void) {  
24  
25     SystemInit();          /* System Initialization (i.e., PLL) */  
26     LED_init();            /* LED Initialization */  
27     BUTTON_init();         /* BUTTON Initialization */  
28  
29     while (1) {           /* Loop forever */  
30     }  
31  
32 }  
33
```

C libraries

lpc17xx.h, sys_lpc17xx.c and core_cm3.h

- Similarly to lpc17xx.* files, it defines some constants and functions at CPU core level.

```
129  /** \brief Union type to access the Application Program Status Register (APSR).  
130  */  
131  typedef union  
132  {  
133      struct  
134      {  
135 #if __CORTEX_M != 0x04  
136         uint32_t _reserved0:27;           /*!< bit: 0..26 Reserved */  
137 #else  
138         uint32_t _reserved0:16;          /*!< bit: 0..15 Reserved */  
139         uint32_t GE:4;                  /*!< bit: 16..19 Greater than or Equal flags */  
140         uint32_t _reserved1:7;          /*!< bit: 20..26 Reserved */  
141 #endif  
142         uint32_t Q:1;                  /*!< bit: 27 Saturation condition flag */  
143         uint32_t V:1;                  /*!< bit: 28 Overflow condition code flag */  
144         uint32_t C:1;                  /*!< bit: 29 Carry condition code flag */  
145         uint32_t Z:1;                  /*!< bit: 30 Zero condition code flag */  
146         uint32_t N:1;                  /*!< bit: 31 Negative condition code flag */  
147     } b;  
148     uint32_t w;  
149 } APSR_Type;
```

Program Status register
high-level usage

C libraries

lpc17xx.h, sys_lpc17xx.c and core_cm3.h

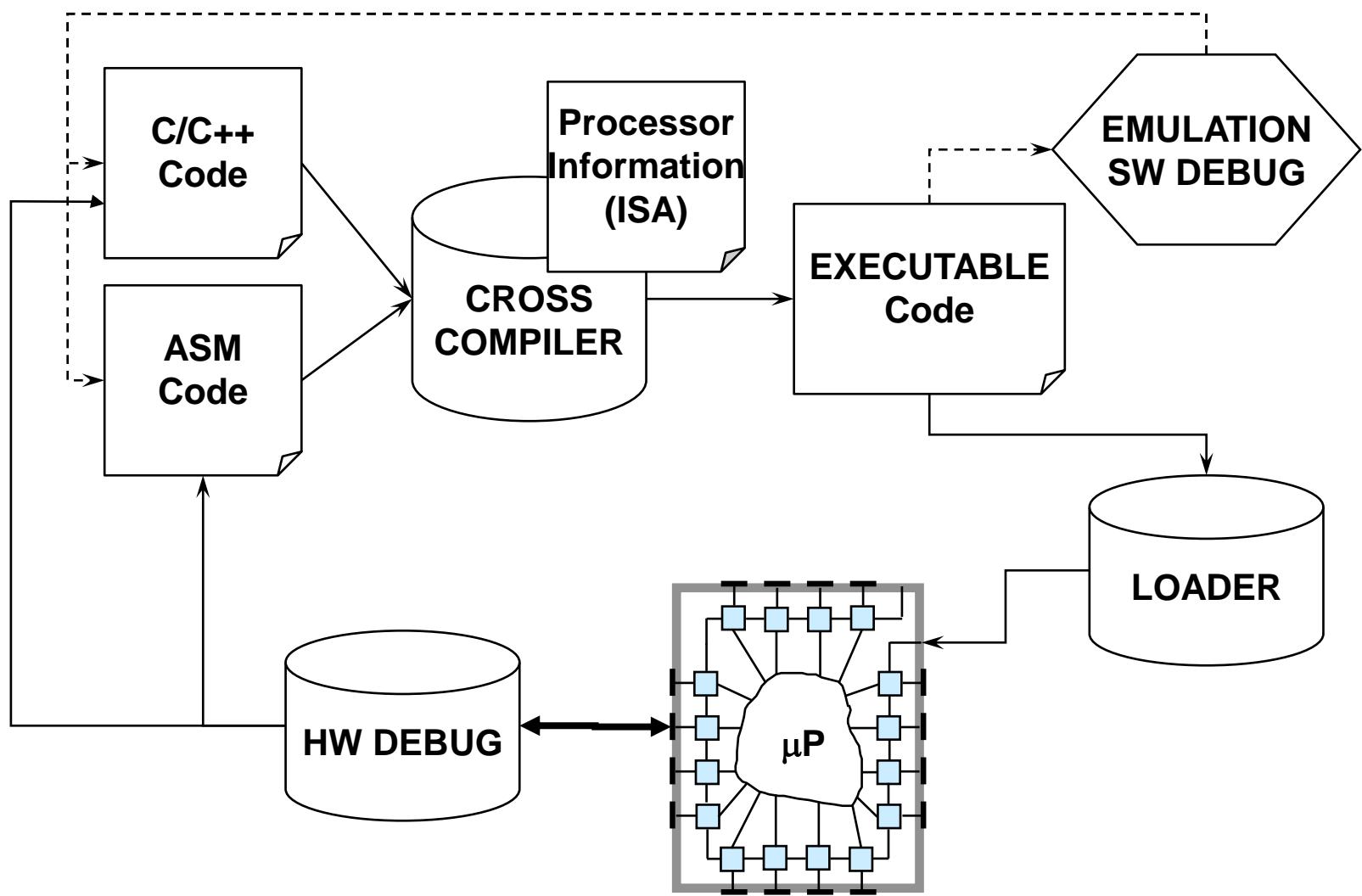
- Similarly to lpc17xx.* files, it defines some constants and functions at CPU core level.

```
935  /** \brief Enable External Interrupt
936
937     This function enables a device specific interrupt in the NVIC interrupt
938     The interrupt number cannot be a negative value.
939
940     \param [in]      IRQn  Number of the external interrupt to enable
941 */
942 static __INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)
943 {
944     NVIC->ISER[((uint32_t)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F));
945 }
946
```

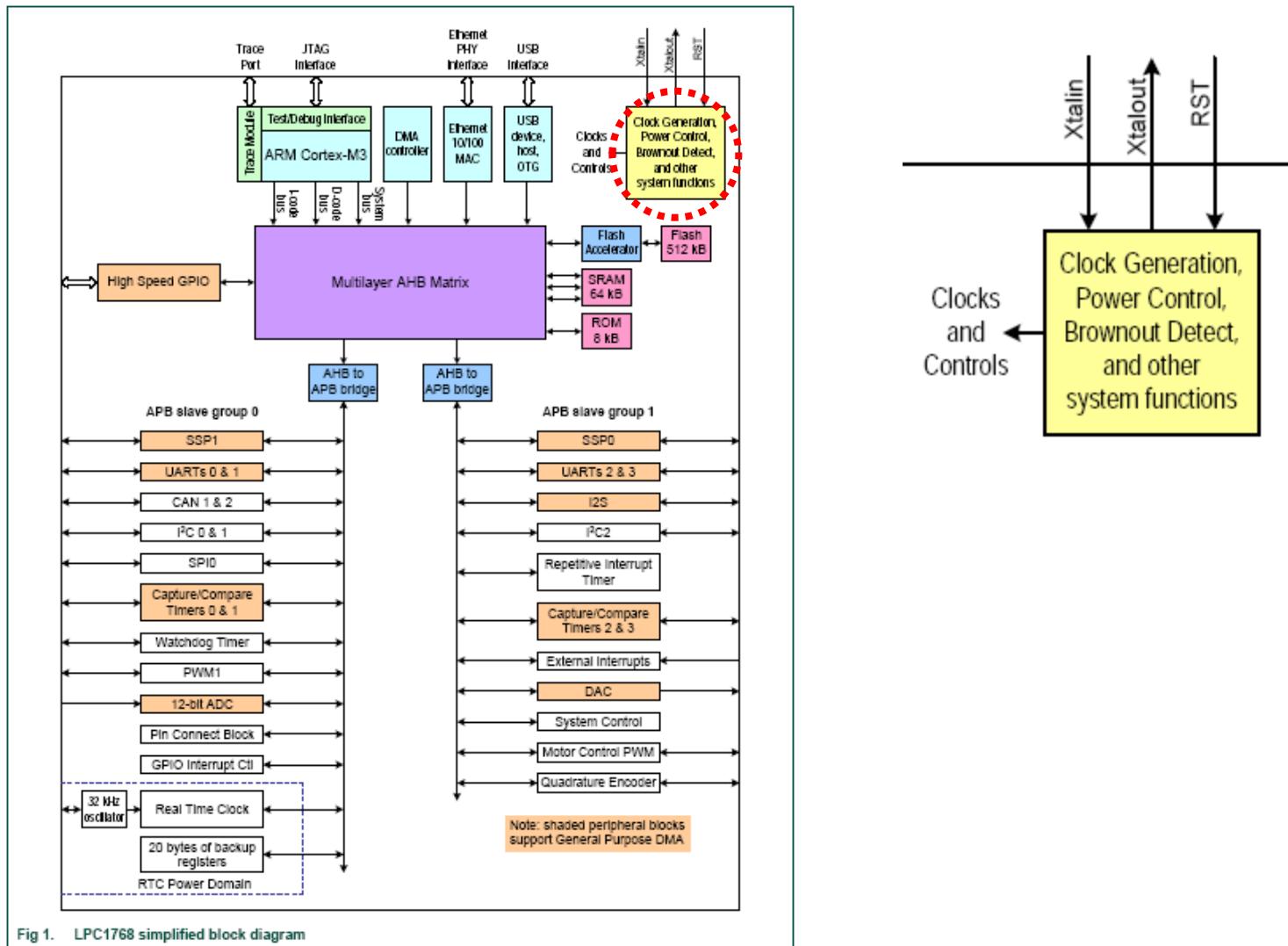
Nested Interrupt controller
peripheral block

Specific register to configure to
enable an external interrupt source

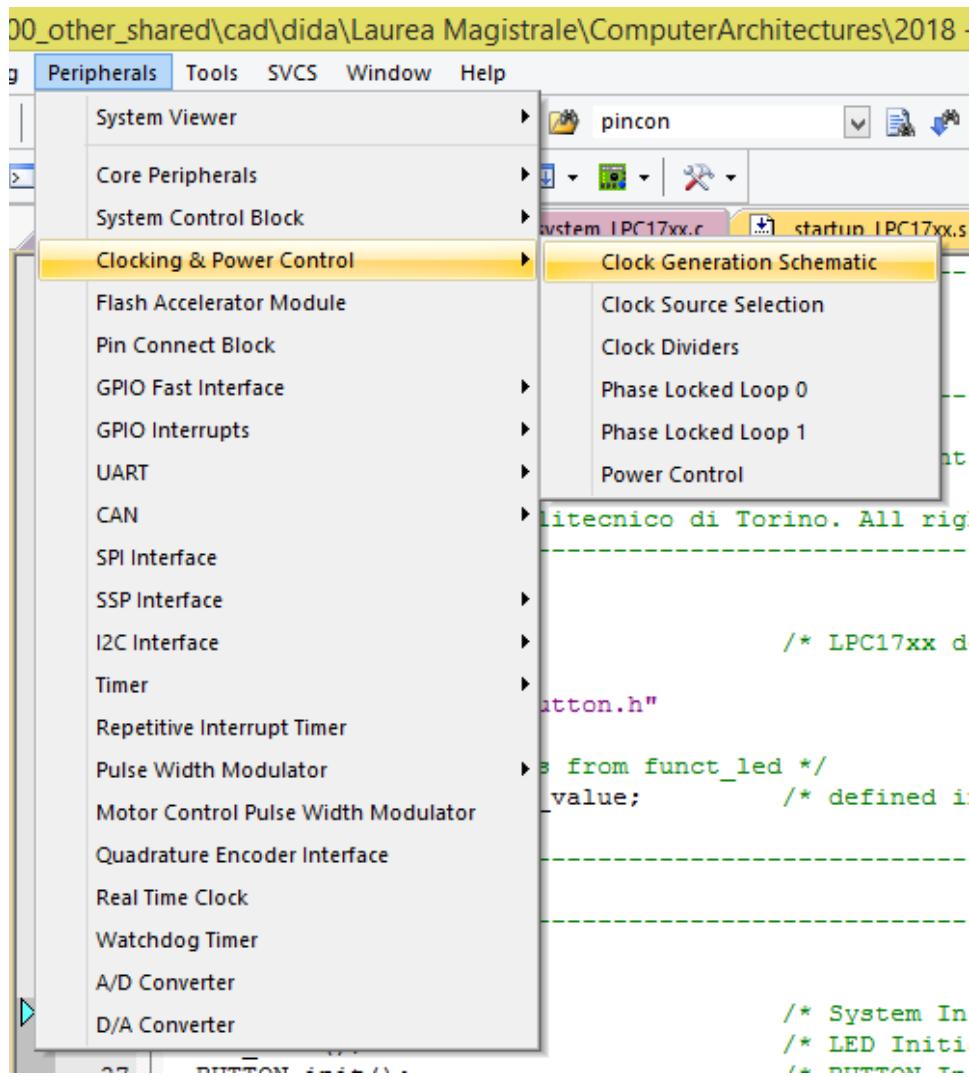
Tool chain



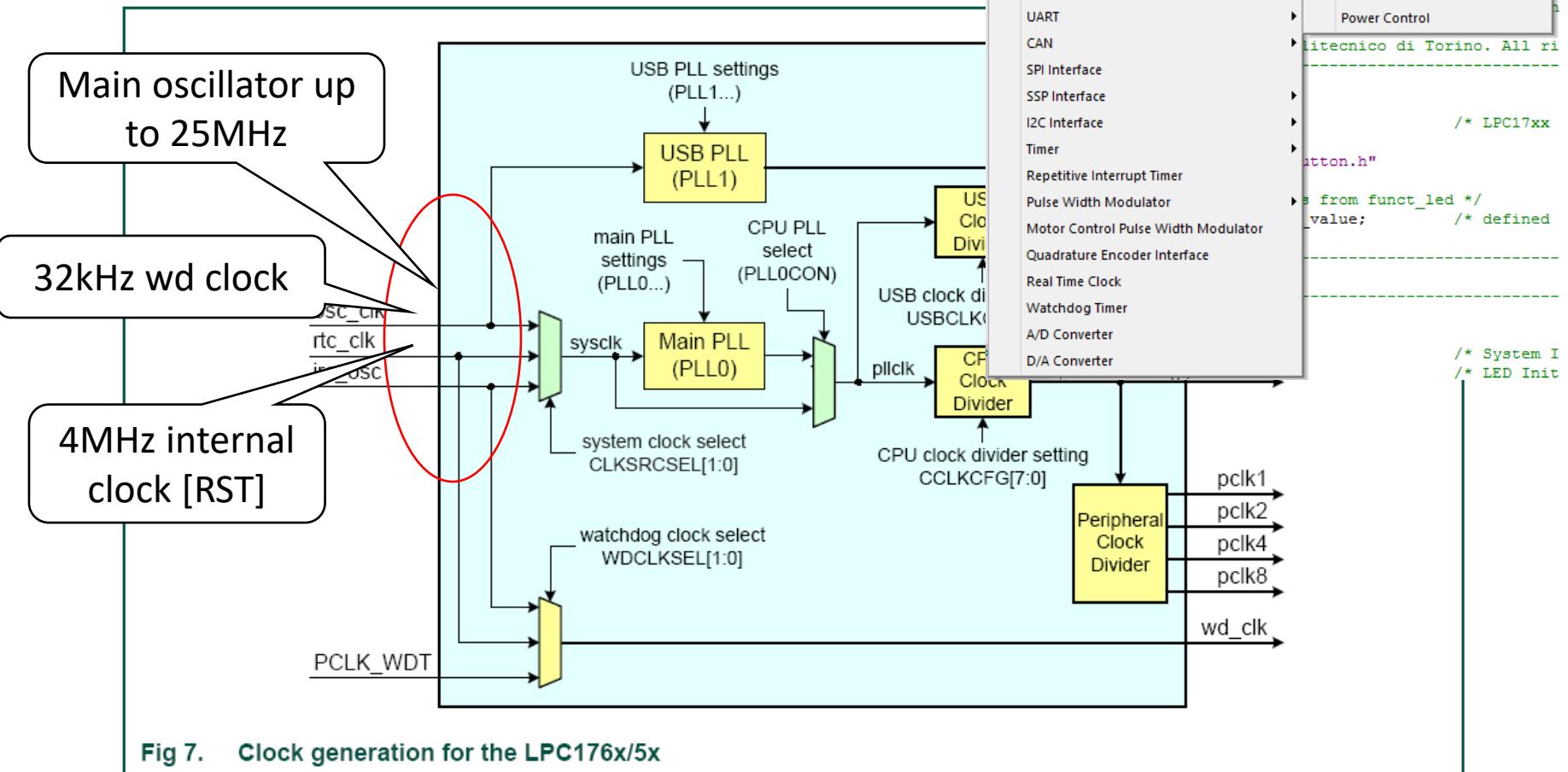
Clock distribution setup



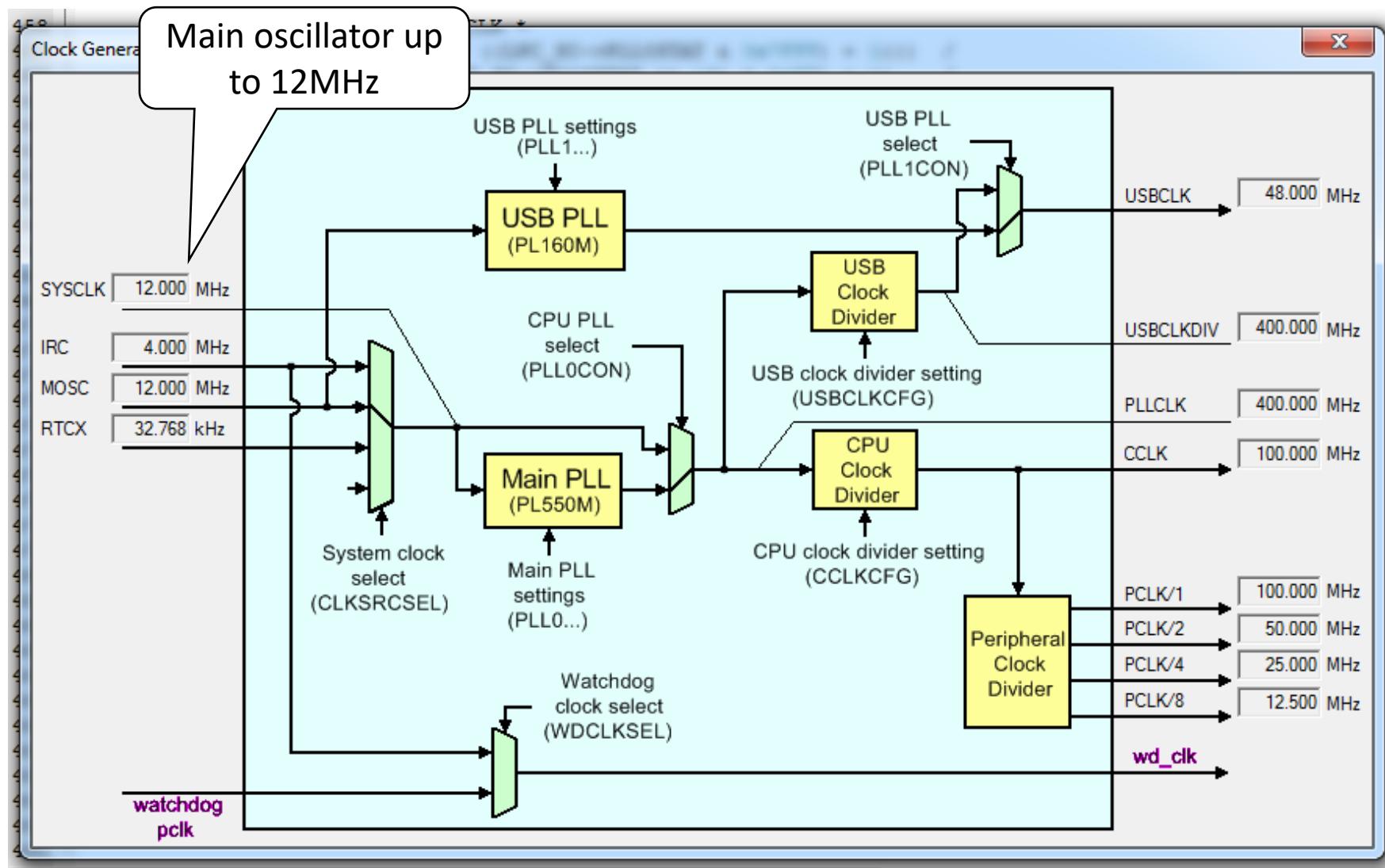
Debug view



Clock selection



Current values in sample



Current configuration

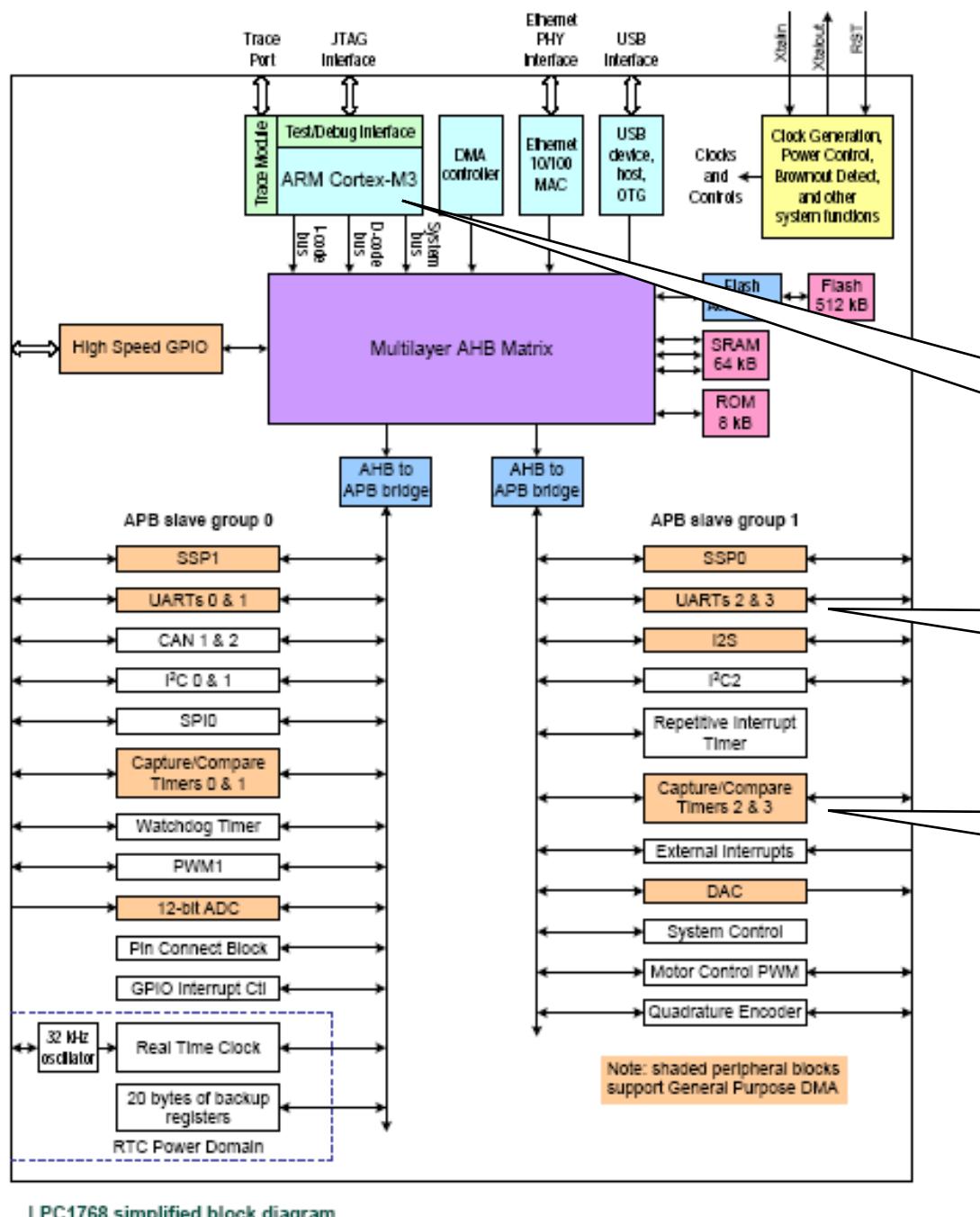


Fig 1. LPC1768 simplified block diagram

Clock setup configuration

The screenshot shows a software development environment with multiple tabs at the top: 'startup_LPC17xx.s', 'lib_led.c', 'system_LPC17xx.c', 'startup_LPC17xx.s', 'sample.c', and 'core_cm3.c'. The 'system_LPC17xx.c' tab is active. Below the tabs, the code for the 'SystemInit' function is displayed:

```
392  /**  
393   * Initialize the system  
394   *  
395   * @param  none  
396   * @return none  
397   *  
398   * @brief  Setup the microcontroller system.  
399   *          Initialize the System and update the SystemFrequency variable.  
400   */  
401 void SystemInit (void)  
402 {  
403 #if (CLOCK_SETUP)  
404     LPC_SC->SCS      = SCS_Val;                      /* Clock Setup */  
405     if (SCS_Val & (1 << 5)) {                         /* If Main Oscillator is enabled */  
406         while ((LPC_SC->SCS & (1<<6)) == 0);/* Wait for Oscillator to be ready */  
407     }  
408  
409     LPC_SC->CCLKCFG  = CCLKCFG_Val;                  /* Setup Clock Divider */  
410  
411     LPC_SC->PCLKSEL0 = PCLKSEL0_Val;                 /* Peripheral Clock Selection */  
412     LPC_SC->PCLKSEL1 = PCLKSEL1_Val;  
413 }
```

The 'Text Editor' tab at the bottom left is circled in red.

Clock setup configuration wizard (II)

The screenshot shows the Keil MDK-ARM IDE interface with the 'system_LPC17xx.c' tab selected in the top menu bar. Below the tabs, there are buttons for 'Expand All', 'Collapse All', 'Help', and 'Show Grid'. The main window displays a hierarchical tree view under the 'Option' column, with the 'Value' column showing checked boxes for most entries. The tree structure includes:

- Clock Configuration** (checked):
 - System Controls and Status Register ...
 - OSCRANGE: Main Oscillator Ran... 1 MHz to 20 MHz
 - OSCEN: Main Oscillator Enable (checked)
- PLL0 Configuration (Main PLL) (checked):
 - PLL0 Configuration Register (PLL...
 - MSEL: PLL Multiplier Selection 0x0064
 - NSEL: PLL Divider Selection 0x0006
- PLL1 Configuration (USB PLL) (checked):
 - CPU Clock Configuration Register (C...
 - USB Clock Configuration Register (U...
 - Peripheral Clock Selection Register 0 ...
 - Peripheral Clock Selection Register 1 ...

At the bottom of the window, the text 'Peripheral Clock Selection Register 0 (PCLKSEL0)' is displayed, followed by a red oval highlighting the 'Configuration Wizard' tab in the bottom navigation bar.

Configuration Wizard Annotations

- **Configuration Wizard Annotations** consist of annotation items and annotation modifiers. They create GUI-like elements in IDEs for configuration files.
- Using a GUI-like approach makes it easier for the user to check and adapt configuration files to the application needs.

ARM environment

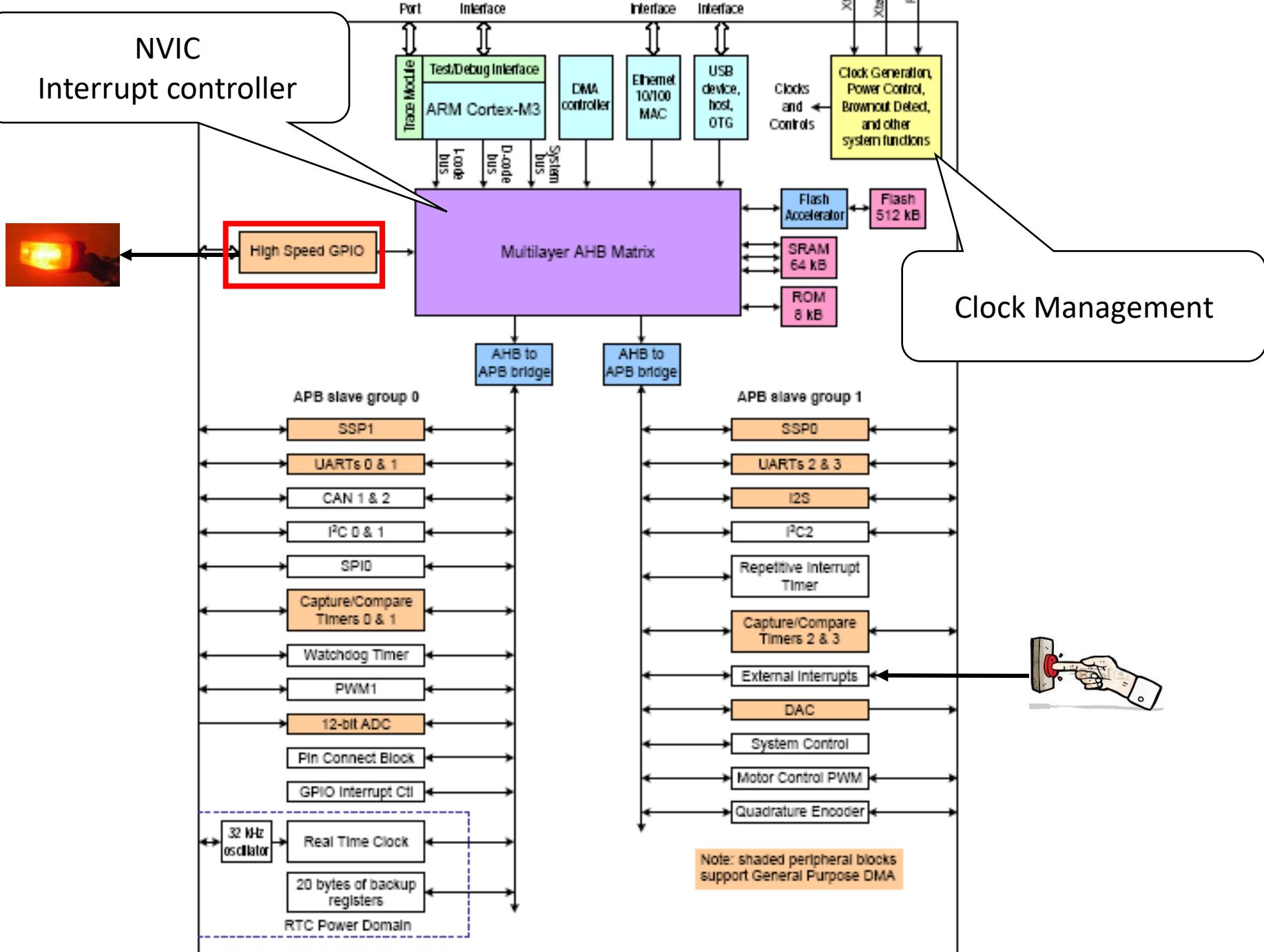
- The Configuration Wizard section must begin within the first 100 lines of code and must start with the following comment line:
 - // <<< Use Configuration Wizard in Context Menu >>>
- The Configuration Wizard section can end with the following optional comment:
 - // <<< end of configuration section >>>
- Annotations are written as comments in the code. Each annotation line must start with a double backslash (//).
- By default, it is the next code symbol that follows the annotation to be modified.
 - It is possible to add a “skip-value” to omits a number of code symbols. This overwrites the previous rule.
- A descriptive text can be added to items.

Lists of the main Configuration Wizard Annotations

- `<h>` : *Heading*. Creates a header section. All items and options enclosed by `<h>` and `</h>` belong to one group and can be expanded. This entry makes no changes to code symbols. It is just used to group other items and modifiers.
- `<e>*` : *Heading with Enable*. Creates a header section with a checkbox to enable or disable all items and options enclosed by `<e>` and `</e>`.
- `<e.i>*` : *Heading with Enable*: modifies a specific bit (*i*) (example: `<e.4>` - changes bit 4 of a value).
- `<i>` : Tooltip help

Lists of the main Configuration Wizard Annotations

- $\langle o \rangle^*$: Option with selection or number entry.
 - // $\langle o \rangle$ Round-Robin Timeout [ticks] <1-1000>
 - The example creates an option with the text *Round-Robin Timeout [ticks]* and a field to enter values that can range between [1..1000].
- $\langle o.x..y \rangle^*$: Option Modify a range of bits. (example: $\langle o.4..5 \rangle$ - bit 4 to 5).
 - // $\langle o.0..15 \rangle$ Language ID <0x0000-0xFCFF>
- $\langle oi \rangle$: Skip i items. Can be applied to all annotation items marked with a *

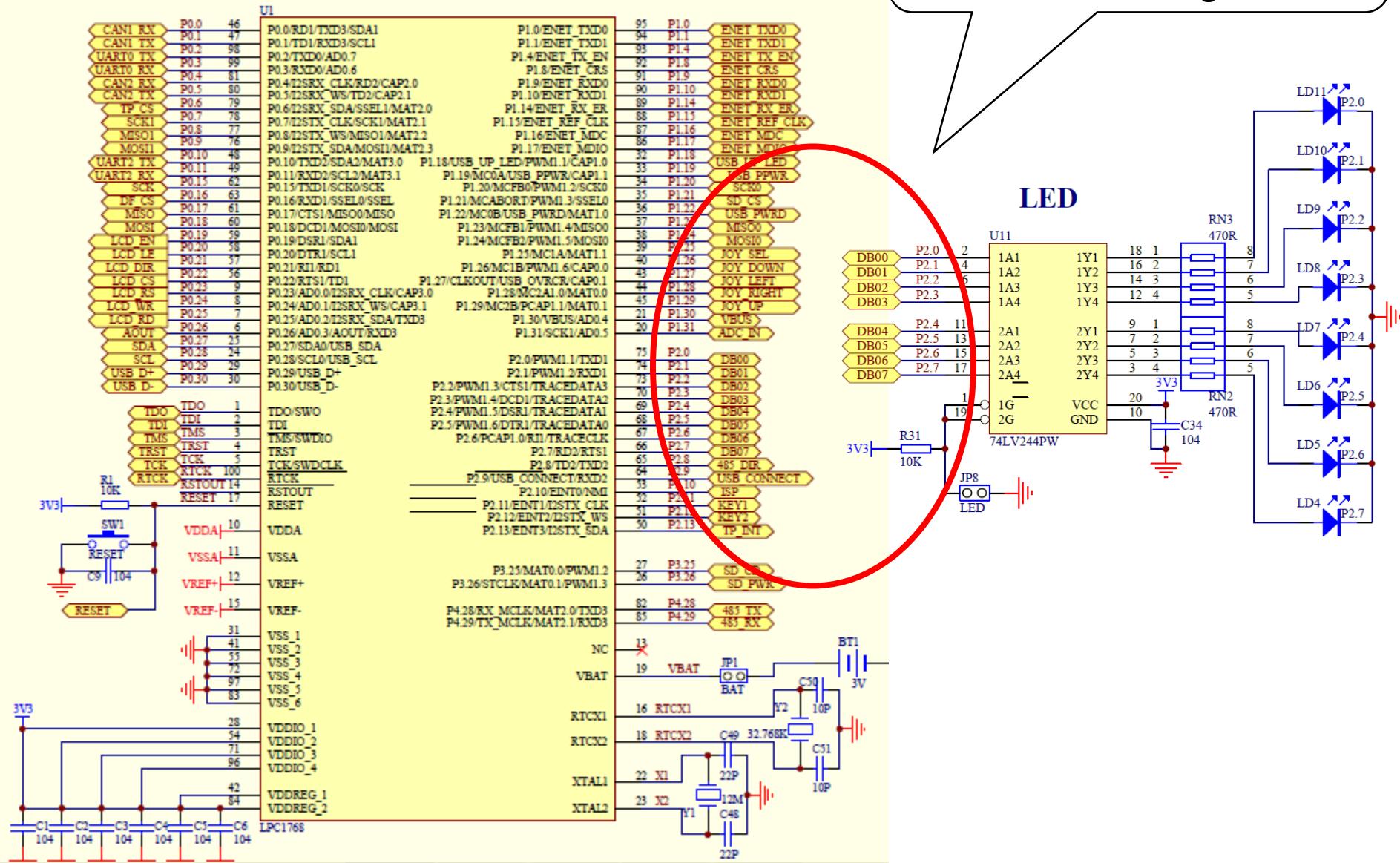


Reading the board schematic

- It is important because
 - the SoC description is not including the description of the board composition
 - It is not know how the board components are connected to the CPU

EXAMPLE SoC

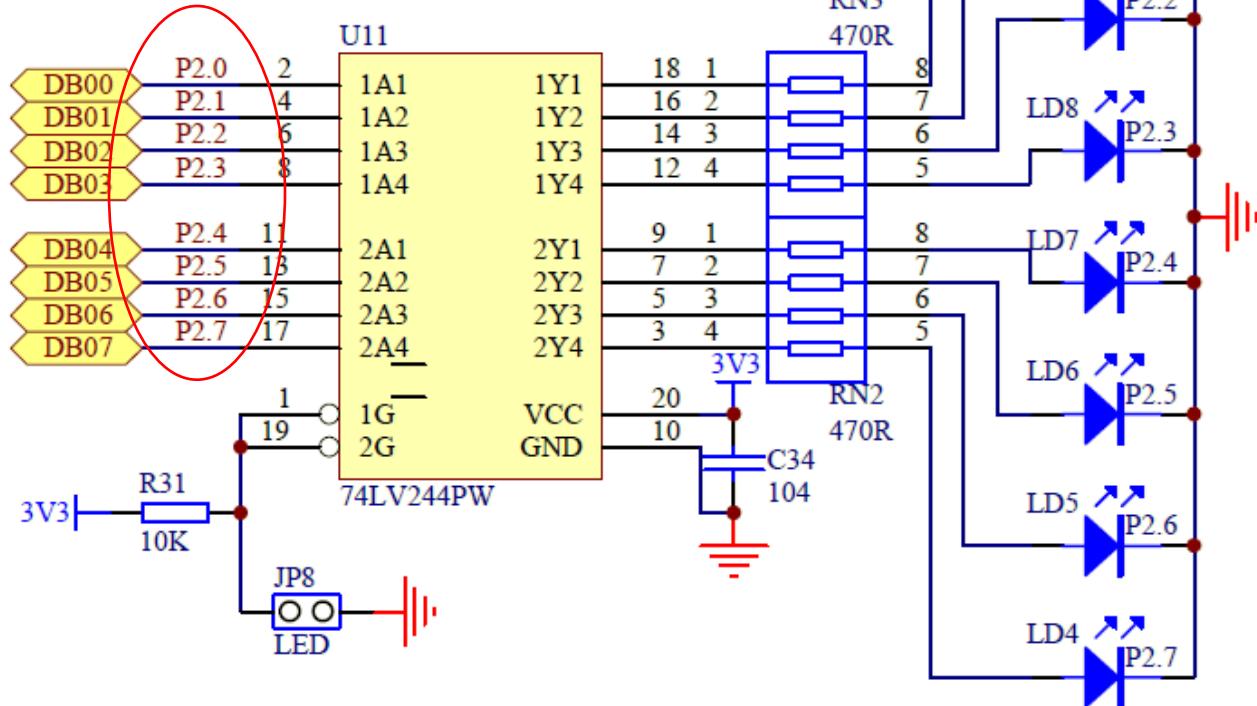
Connection of LEDs to CPU
need to be found in the
schematic diagram



LEDS

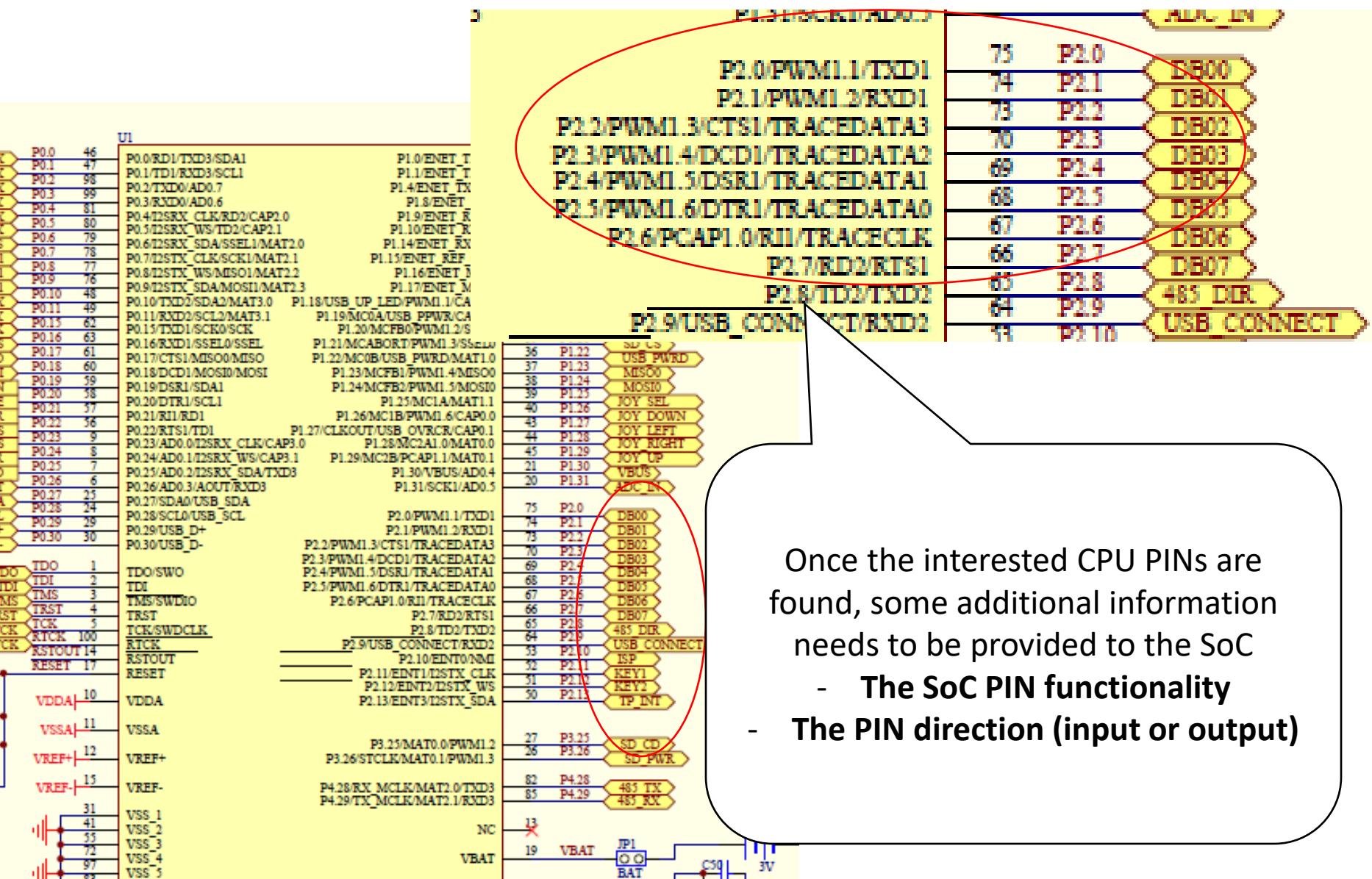
Forced 1 => on
Forced 0 => off

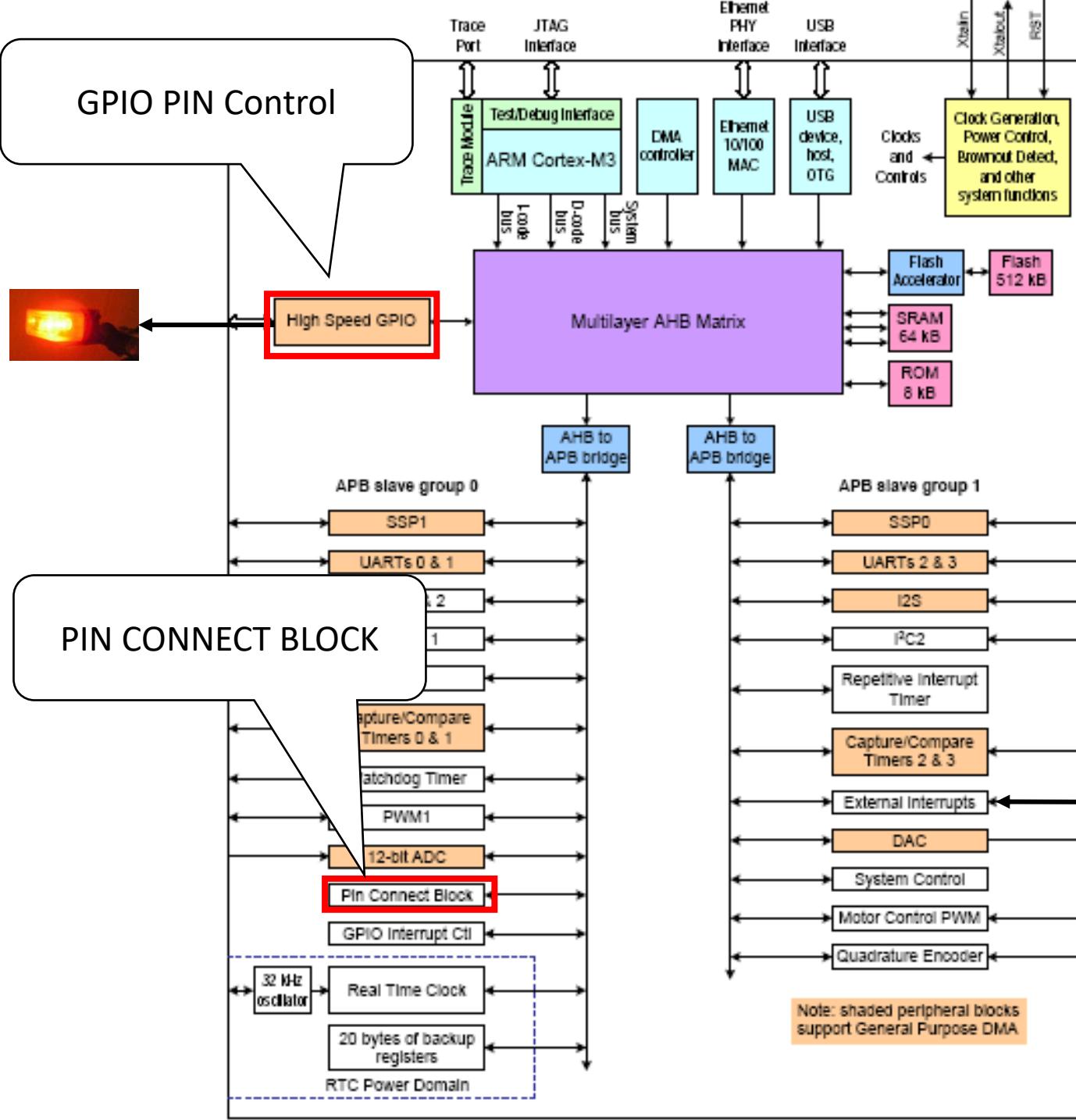
- Look at the schematic
LED



- LD4 → p2.7 LD5 → p2.6 ... LD11 → p2.0

PIN selection





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 	00
7:6	P2.3	GPIO Port 2.3	PWM1.4	DCD1	Reserved 	00
9:8	P2.4	GPIO Port 2.4	PWM1.5	DSR1	Reserved 	00
11:10	P2.5	GPIO Port 2.5	PWM1.6	DTR1	Reserved 	00
13:12	P2.6	GPIO Port 2.6	PCAP1.0	RI1	Reserved 	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	EINT0	NMI	Reserved	00
23:22	P2.11 	GPIO Port 2.11	EINT1	Reserved	I2STX_CLK	00
25:24	P2.12 	GPIO Port 2.12	EINT2	Reserved	I2STX_WS	00
27:26	P2.13 	GPIO Port 2.13	EINT3	Reserved	I2STX_SDA	00
31:28	-	Reserved	Reserved	Reserved	Reserved	0

General purpose I/O (GPIO)

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

Direction
In/out

If input,
Read pin
value

If output,
set/clr

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

lib_led.c – setup PinSel and direction

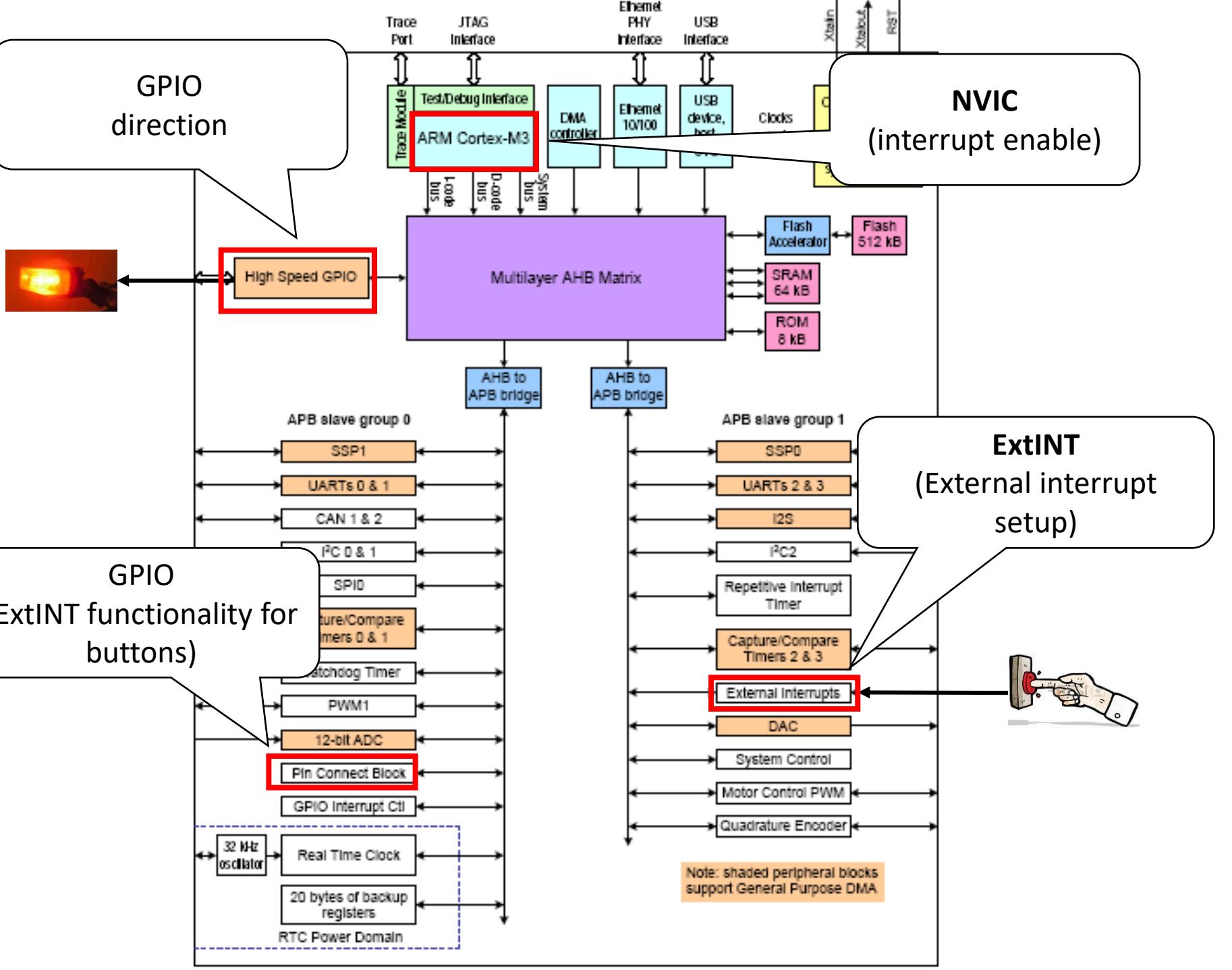
PIN mode GPIO (00b value per P2.0 to P2.7)

```
20 void LED_init(void) {  
21  
22     LPC_PINCON->PINSEL4 &= 0xFFFF0000; /  
23     LPC_GPIO2->FIODIR    |= 0x000000FF; /  
24     /* LPC_GPIO2->FIOSET    = 0x000000FF;  
25     LPC_GPIO2->FIOCLR    = 0x000000FF; /  
26  
27     led_value = 0;  
28 }
```

Lib_led – setup PinSel and direction

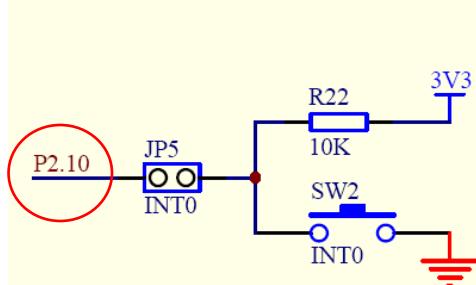
P2.0...P2.7 Output (LEDs on PORT2 defined as Output)

```
20 void LED_init(void) {  
21  
22     LPC_PINCON->PINSEL4 &= 0xFFFF0000; //  
23     LPC_GPIO2->FIODIR |= 0x000000FF; //  
24     /* LPC_GPIO2->FIOSET = 0x000000FF;  
25     LPC_GPIO2->FIOCLR = 0x000000FF; //  
26  
27     led_value = 0;  
28 }
```

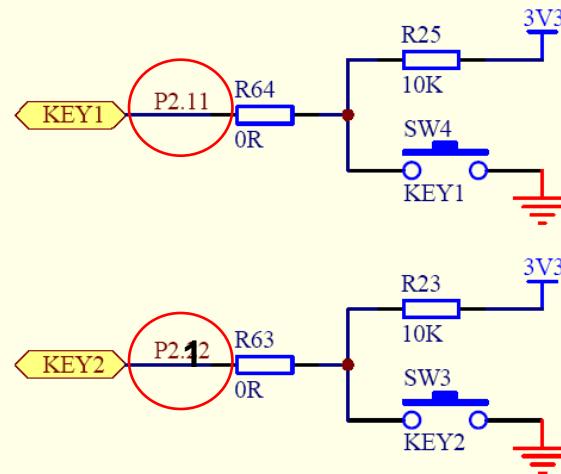


Buttons

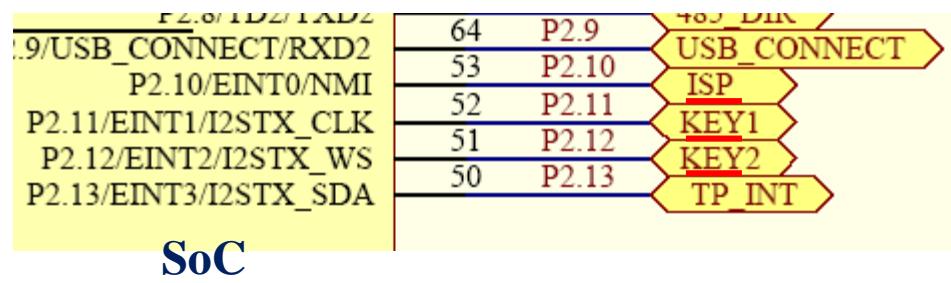
- Look at the schematic



KEY



- INT0 → p2.10
- KEY1 → p2.11
- KEY2 → p2.12



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 	00
7:6	P2.3	GPIO Port 2.3	PWM1.4	DCD1	Reserved 	00
9:8	P2.4	GPIO Port 2.4	PWM1.5	DSR1	Reserved 	00
11:10	P2.5	GPIO Port 2.5	PWM1.6	DTR1	Reserved 	00
13:12	P2.6	GPIO Port 2.6	PCAP1.0	RI1	Reserved 	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	EINT0	NMI	Reserved	00
23:22	P2.11 	GPIO Port 2.11	EINT1	Reserved	I2STX_CLK	00
25:24	P2.12 	GPIO Port 2.12	EINT2	Reserved	I2STX_WS	00
27:26	P2.13 	GPIO Port 2.13	EINT3	Reserved	I2STX_SDA	00
31:28	-	Reserved	Reserved	Reserved	Reserved	0

+ [] Chapter 1: LPC176x/5x Introductory information
+ [] Chapter 2: LPC176x/5x Memory map
- [] Chapter 3: LPC176x/5x System control
[] 3.1 Introduction
[] 3.2 Pin description
[] 3.3 Register description
- [] 3.4 Reset
[] 3.5 Brown-out detection
- [] 3.6 External interrupt inputs
[] 3.7 Other system controls and status flags
+ [] Chapter 4: LPC176x/5x Clocking and power control
+ [] Chapter 5: LPC176x/5x Flash accelerator
+ [] Chapter 6: LPC176x/5x Nested Vectored Interrupt Controller
+ [] Chapter 7: LPC176x/5x Pin configuration

External Interrupt mode

Bit	Symbol	Value	Description	Reset value
0	EXTMODE0	0	Level-sensitivity is selected for EINT0.	0
		1	EINT0 is edge sensitive.	

Bit	Symbol	Value	Description	Reset value
0	EXTPOLAR0	0	EINT0 is low-active or falling-edge sensitive (depending on EXTmode0).	0
		1	EINT0 is high-active or rising-edge sensitive (depending on EXTmode0).	

Table 9. External Interrupt registers

Name	Description	Access	Reset value	Address
EXTINT	The External Interrupt Flag Register contains interrupt flags for EINT0, EINT1, EINT2 and EINT3. See Table 10 .	R/W	0x00	0x400F C140
EXTMODE	The External Interrupt Mode Register controls whether each pin is edge- or level-sensitive. See Table 11 .	R/W	0x00	0x400F C148
EXTPOLAR	The External Interrupt Polarity Register controls which level or edge on each pin will cause an interrupt. See Table 12 .	R/W	0x00	0x400F C14C

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

lib_button.c – setup PinSel, direction and ExtINT config

```
5  /** @bri */
6  */
7
8 void BUTTON_init(void) {
9
10    LPC_PINCON->PINSEL4 |= (1 << 20);
11    LPC_GPIO2->FIODIR  &= ~(1 << 10);
12
13    LPC_PINCON->PINSEL4 |= (1 << 22);
14    LPC_GPIO2->FIODIR  &= ~(1 << 11);
15
16    LPC_PINCON->PINSEL4 |= (1 << 24);
17    LPC_GPIO2->FIODIR  &= ~(1 << 12);
18
19    LPC_SC->EXTMODE = 0x7;
20
21    NVIC_EnableIRQ(EINT2_IRQn);
22    NVIC_EnableIRQ(EINT1_IRQn);
23    NVIC_EnableIRQ(EINT0_IRQn);
24 }
```

EXTINT functionality and direction input

lib_button.c – setup PinSel, direction and ExtINT config

```
5  /***
6   * @brief Function that initializes Button
7   */
8 void BUTTON_init(void) {
9
10    LPC_PINCON->PINSEL4      |= (1 << 20);      /
11    LPC_GPIO2->FIODIR       &= ~(1 << 10);      /
12
13    LPC_PINCON->PINSEL4      |= (1 << 24);      /
14    LPC_GPIO2->FIODIR       &= ~(1 << 12);      /
15
16    LPC_SC->EXTMODE = 0x7;
17
18
19
20
21    NVIC_EnableIRQ(EINT2_IRQn);      /
22    NVIC_EnableIRQ(EINT1_IRQn);      /
23    NVIC_EnableIRQ(EINT0_IRQn);      /
24 }
```

EXTINT pins mode: edge sensitive

lib_button.c – setup PinSel, direction and ExtINT config

```
5  /***
6   * @brief Function that initializes Button
7   */
8 void BUTTON_init(void) {
9
10    LPC_PINCON->PINSEL4 |= (1 << 20); /
11    LPC_GPIO2->FIODIR &= ~(1 << 10); /
12
13    LPC_PINCON->PINSEL4 |= (1 << 22); /
14    LPC_GPIO2->FIODIR &= ~(1 << 11); /
15
16    LPC
17    LPC
18
19    LPC_SC->EXTMC = 0x7;
20
21    NVIC_EnableIRQ(EINT2_IRQn);
22    NVIC_EnableIRQ(EINT1_IRQn);
23    NVIC_EnableIRQ(EINT0_IRQn);
24 }
```

Nested Vectored Interrupt Controller (NVIC)
selective enable of external interrupts

Interruption handler and IVT

```
6 void EINT0_IRQHandler (void)           ←
7 {
8     LED_On(0);
9     LPC_SC->EXTINT |= (1 << 0);      /* clear pending interrupt
10 }
```

```
52             AREA    RESET, DATA, READONLY
53             EXPORT   __Vectors
54
55 __Vectors    DCD      _initial_sp          ; Top of Stack
56                 DCD      Reset_Handler       ; Reset Handler
57                 DCD      NMI_Handler         ; NMI Handler
58                 DCD      HardFault_Handler    ; Hard Fault Handler
59                 DCD      MemManage_Handler    ; MPU Fault Handler
60                 DCD      BusFault_Handler     ; Bus Fault Handler
61                 DCD      UsageFault_Handler  ; Usage Fault Handler
62                 DCD      0
63         91             DCD      EINT0_IRQHandler    ; 34: External Interrupt 0
64         92             DCD      EINT1_IRQHandler    ; 35: External Interrupt 1
65         93             DCD      EINT2_IRQHandler    ; 36: External Interrupt 2
66         94             DCD      EINT3_IRQHandler    ; 37: External Interrupt 3
67         95             DCD      0
68             DCD      0
69             DCD      PendSV_Handler       ; PendSV Handler
70             DCD      SysTick_Handler      ; SysTick Handler
71
72             ; External Interrupts
73             DCD      WDT_IRQHandler      ; 16: Watchdog Timer
74             DCD      TIMER0_IRQHandler   ; 17: Timer0
75             DCD      TIMER1_IRQHandler   ; 18: Timer1
```

Overall view (MAIN PROGRAM)

```
20 /*-----  
21  Main Program  
22 -----*/  
23 int main (void) {  
24  
25     SystemInit();          /* System Initialization (i.e., PLL) */  
26     LED_init();            /* LED Initialization */  
27     BUTTON_init();         /* BUTTON Initialization */  
28  
29     while (1) {           /* Loop forever */  
30         ...  
31     }  
32 }
```

WAIT FOR
external
INTERRUPT

BUTTON INIT

LED INIT

CLOCK INIT

How to trigger an external interruption

The screenshot shows the µVision4 IDE interface. The left sidebar is titled "Peripherals" and lists various hardware components. The "GPIO Fast Interface" option is selected, and its sub-menu shows Port 0, Port 1, Port 2 (which is currently selected), Port 3, and Port 4. The main workspace displays a C code file for a sample application. A context menu is open over the code, specifically over the declaration of the GPIO2 structure. This context menu is titled "General Purpose Input/Output 2 (GPIO 2) - Fast Interface". It contains fields for FIO2DIR (0x000000FF), FIO2MASK (0x00000000), FIO2SET (0x00000000), FIO2CLR (0x00000000), FIO2PIN (0x00002300), and Pins (0x00003F00). To the right of these fields is a bit mask table with columns for Bits 31, 24, 23, 16, 15, 8, 7, and 0. The bit mask for the Pins field is shown as 0x00003F00, with the 8 least significant bits set to 1 (indicated by checked boxes) and the others set to 0.

```
ed through EINT buttons

Port 0
Port 1
Port 2
Port 3
Port 4

without warranties
Torino. All rights
/* LPC17xx defi
utton.h"

    from funct_led */
    value;      /* defined in funct_led
    */

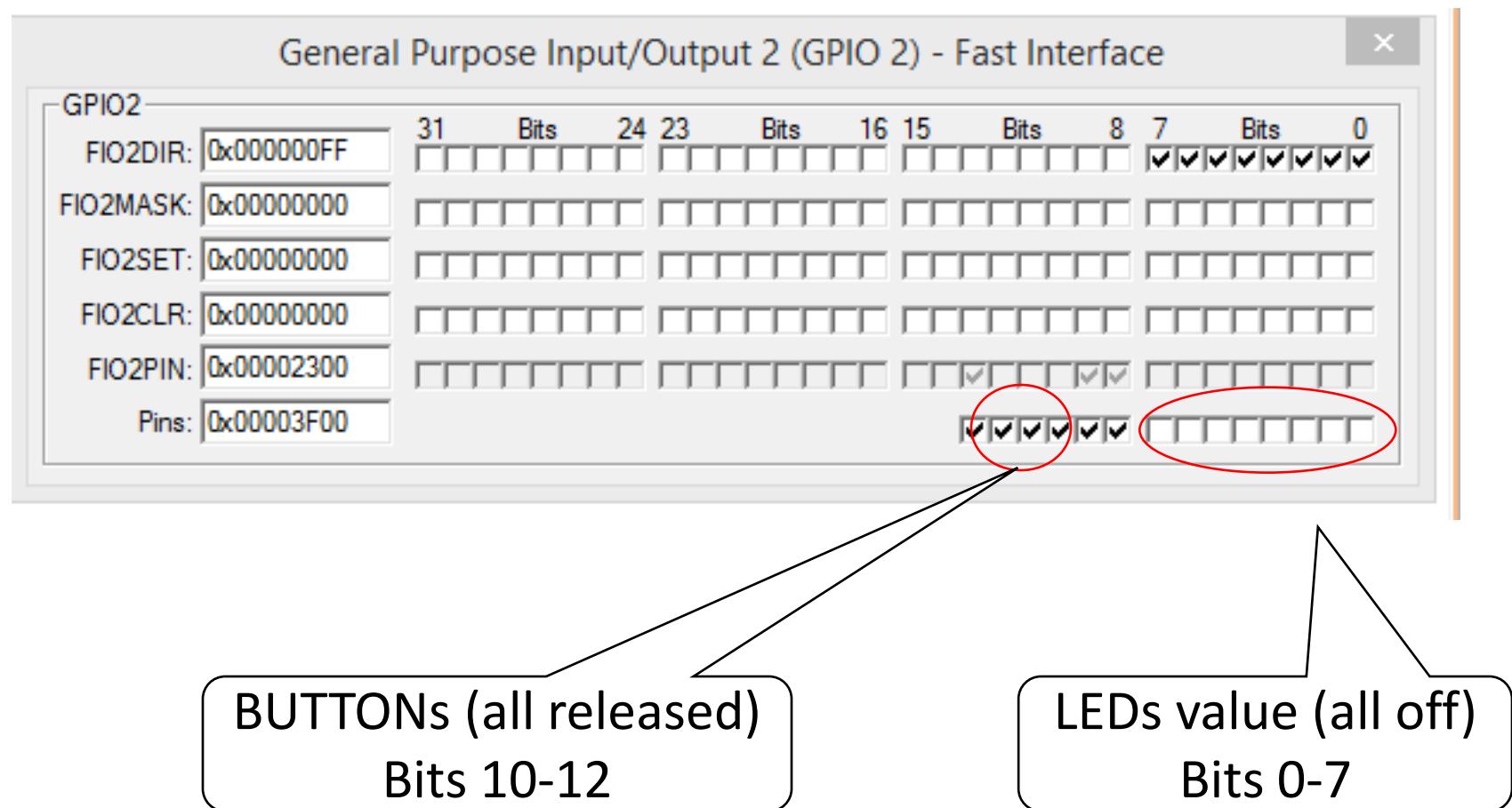
    /* System Initialization (i.e., PLL) */
    /* LED Initialization
    /* BUTTON Initialization
    */

    while (1) {
        /* Loop forever
    }

BUTTON_init();

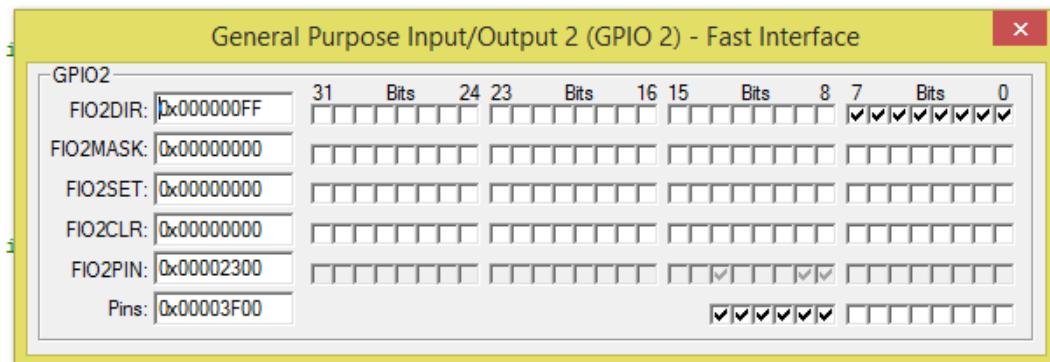
```

GPIO mask

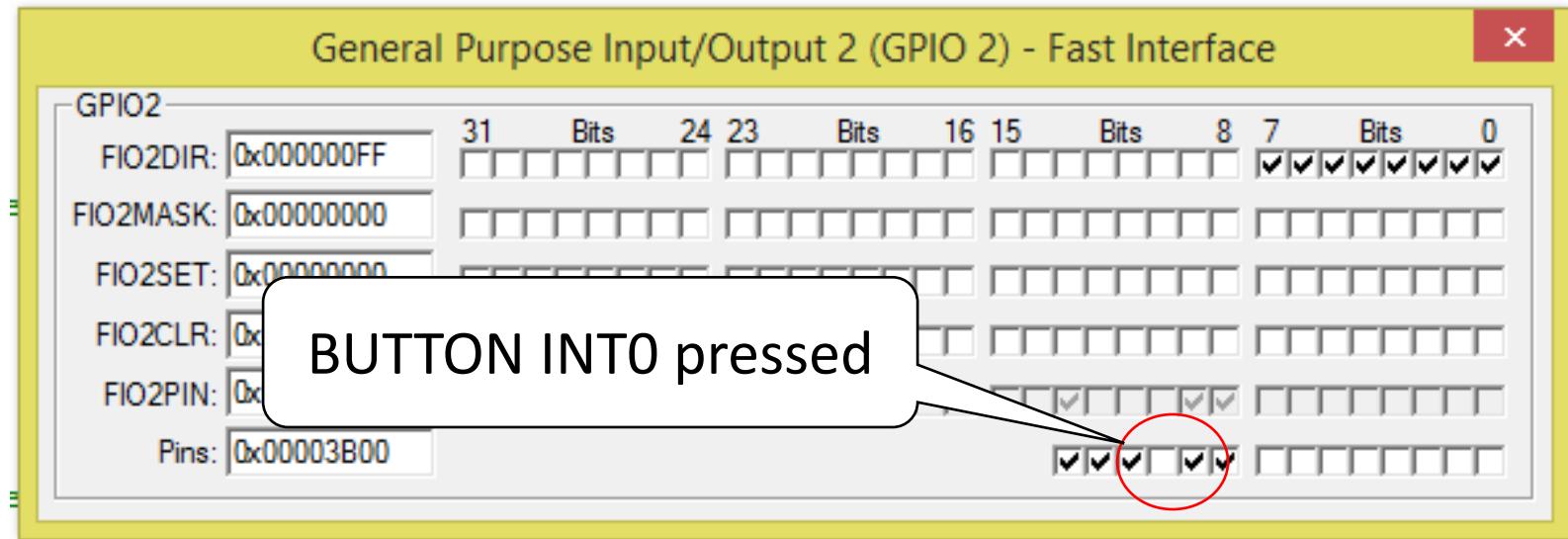


How to trigger an external interrupt

```
5 void EINT0_IRQHandler (void)
6 {
7     LED_On(0);
8     LPC_SC->EXTINT &= (1 << 0);      /* clear pending i
9 }
10
11
12
13 void EINT1_IRQHandler (void)
14 {
15     LED_On(1);
16     LPC_SC->EXTINT &= (1 << 1);      /* clear pending i
17 }
18
19 void EINT2_IRQHandler (void)
20 {
21     LED_Off(0);
22     LED_Off(1);
23     LPC_SC->EXTINT &= (1 << 2);      /* clear pending interrupt      */
24 }
```



How to trigger an external interrupt



```
6 void EINT0_IRQHandler (void)
7 {
8     LED_On(0);
9     LPC_SC->EXTINT &= (1 << 0); /* clear pending interrupt
10 }
11
12 }
```

How to trigger an external interrupt

- Try the hardware debug with KEIL