NXP LPC1768

R. Ferrero, P. Bernardi

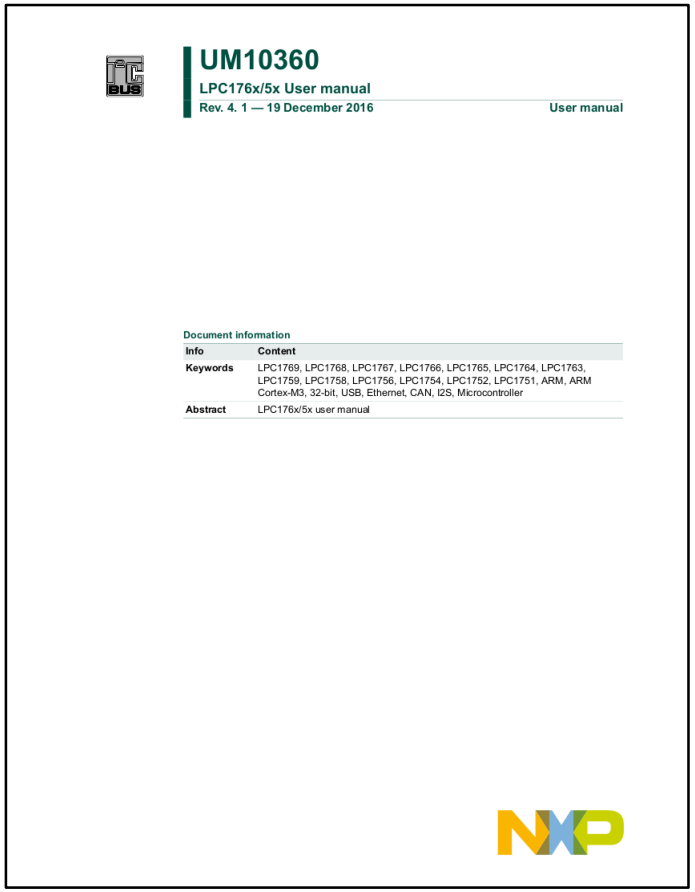
Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)

Torino - Italy

Available resources

- **User manual**
- Sample project
- Schematic



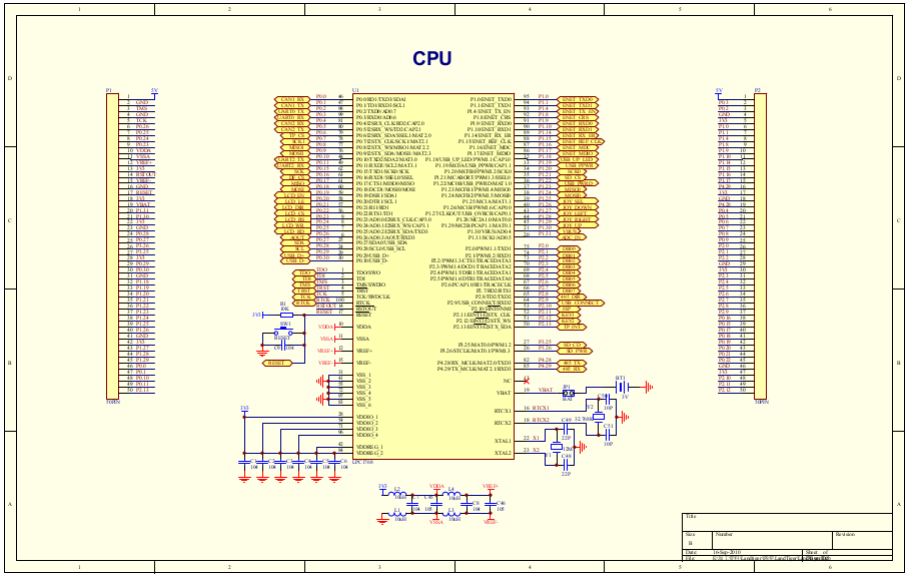
UM10360

LPC176x/5x User manual

Rev. 4. 1 — 19 December 2016 User manual

Document information

Info	Content
Keywords	LPC1769, LPC1768, LPC1767, LPC1766, LPC1765, LPC1764, LPC1763, LPC1759, LPC1758, LPC1756, LPC1754, LPC1752, LPC1751, ARM, ARM Cortex-M3, 32-bit, USB, Ethernet, CAN, I2S, Microcontroller
Abstract	LPC176x/5x user manual



Features (user manual, page 6)

- ARM 32-bit Cortex-M3 Microcontroller with MPU (memory protection unit)
- CPU clock up to 100MHz
- 512kB on-chip Flash ROM with enhanced Flash Memory Accelerator 闪存加速器
- 64kB RAM
- **Nested Vectored Interrupt Controller**
- Eight channel General purpose DMA controller
- **AHB Matrix, APB**

Features (user manual, page 6)

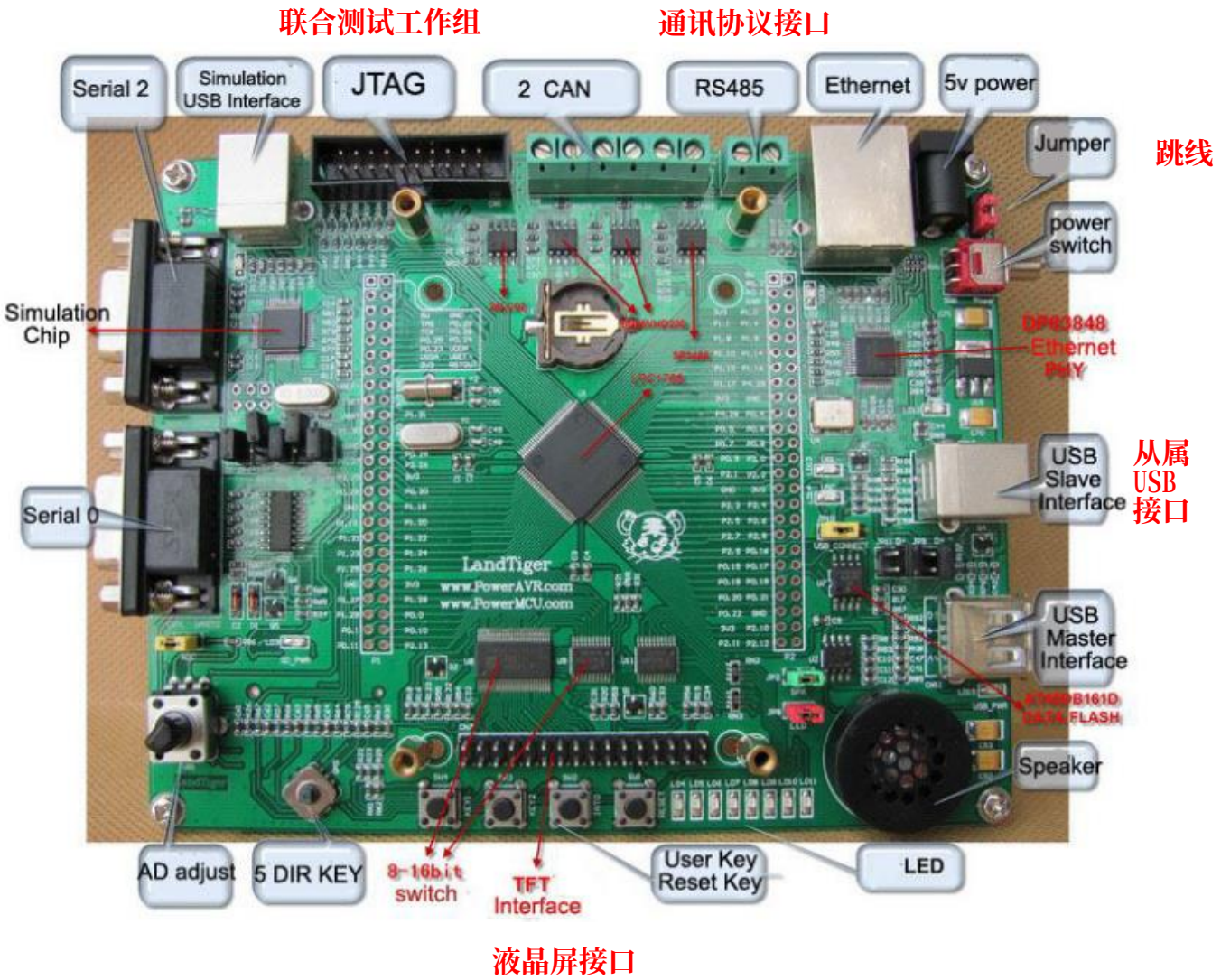
RMII：精简介质独立接口

- Ethernet 10/100 MAC with RMII interface and dedicated DMA DMA：直接存储器存取
- USB 2.0 full-speed Device controller and Host/OTG controller with DMA
- CAN 2.0B with two channels 两个信道
- **Four UARTs**, one with full Modem interface
- Three I2C serial interfaces UART：通用异步收发器
- Three SPI/SSP serial interfaces
- I2S interface

Features (user manual, page 6)

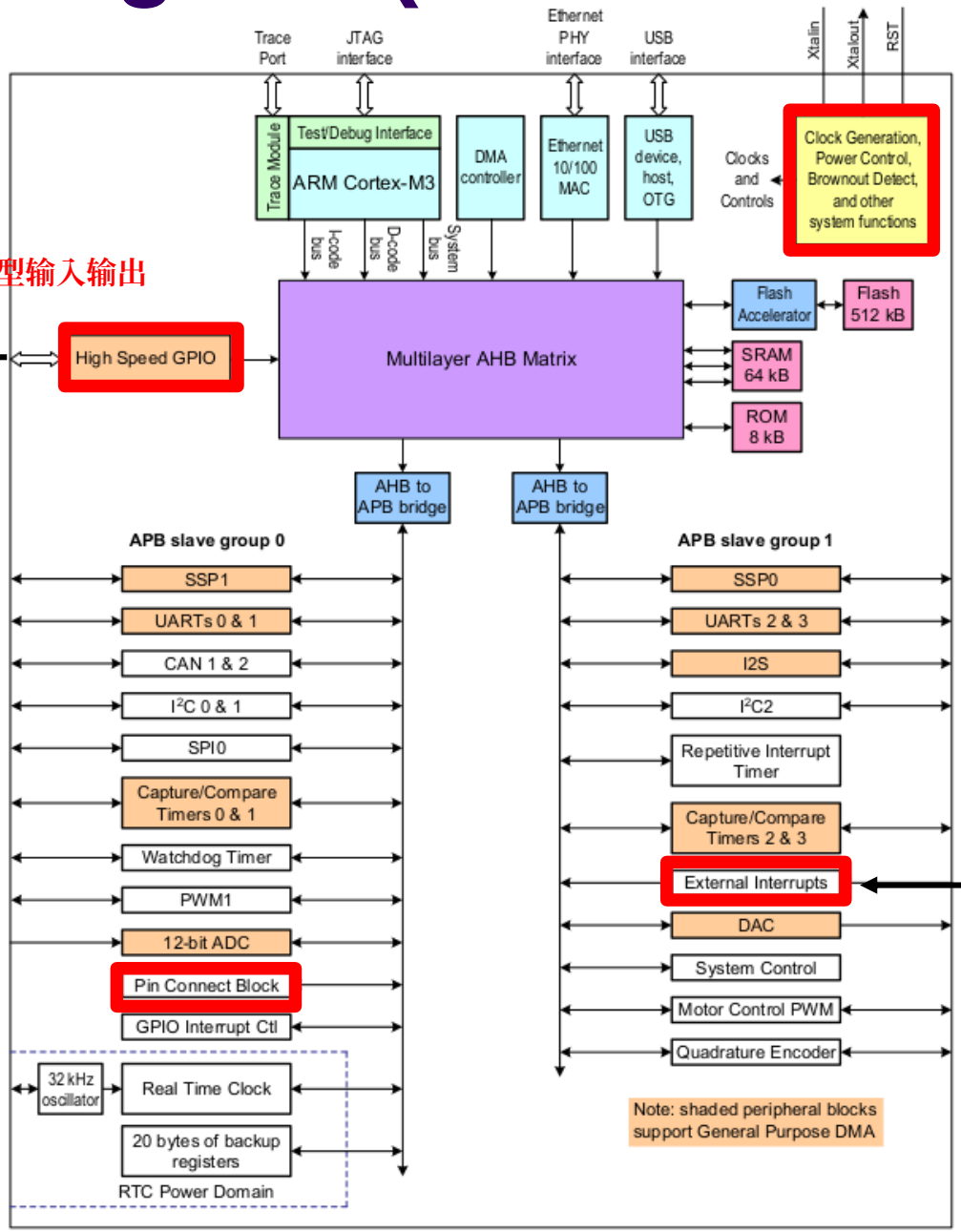
- **General purpose I/O pins**
- 12-bit ADC with 8 channels, 10-bit DAC
- **Four 32-bit Timers** with capture/compare
- Standard PWM Timer block
- Motor control PWM for three-phase Motor control
- Quadrature Encoder
- Watchdog Timer
- Real Time Clock with optional Battery backup
- System Tick Timer, Repetitive Interrupt Timer, ...

Board composition



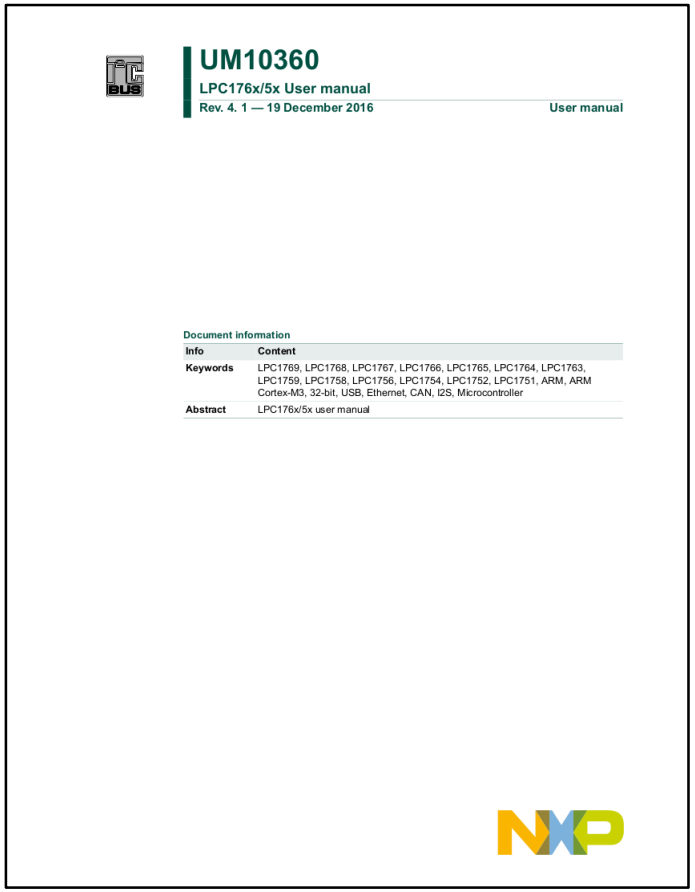
Block diagram (user manual p. 10)

GPIO：通用型输入输出



Available resources

- User manual
- **Sample project**
- Schematic



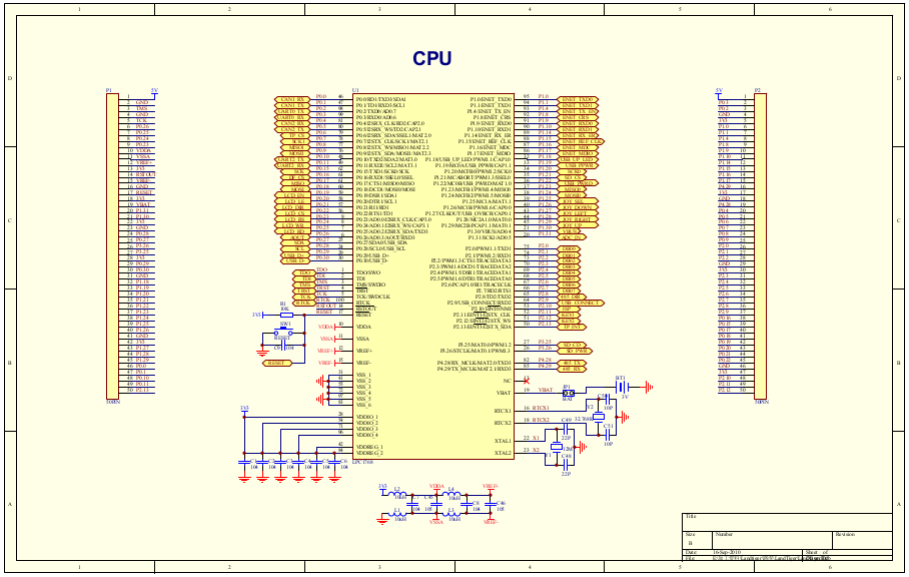
UM10360

LPC176x/5x User manual

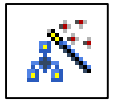
Rev. 4. 1 — 19 December 2016 User manual

Document information

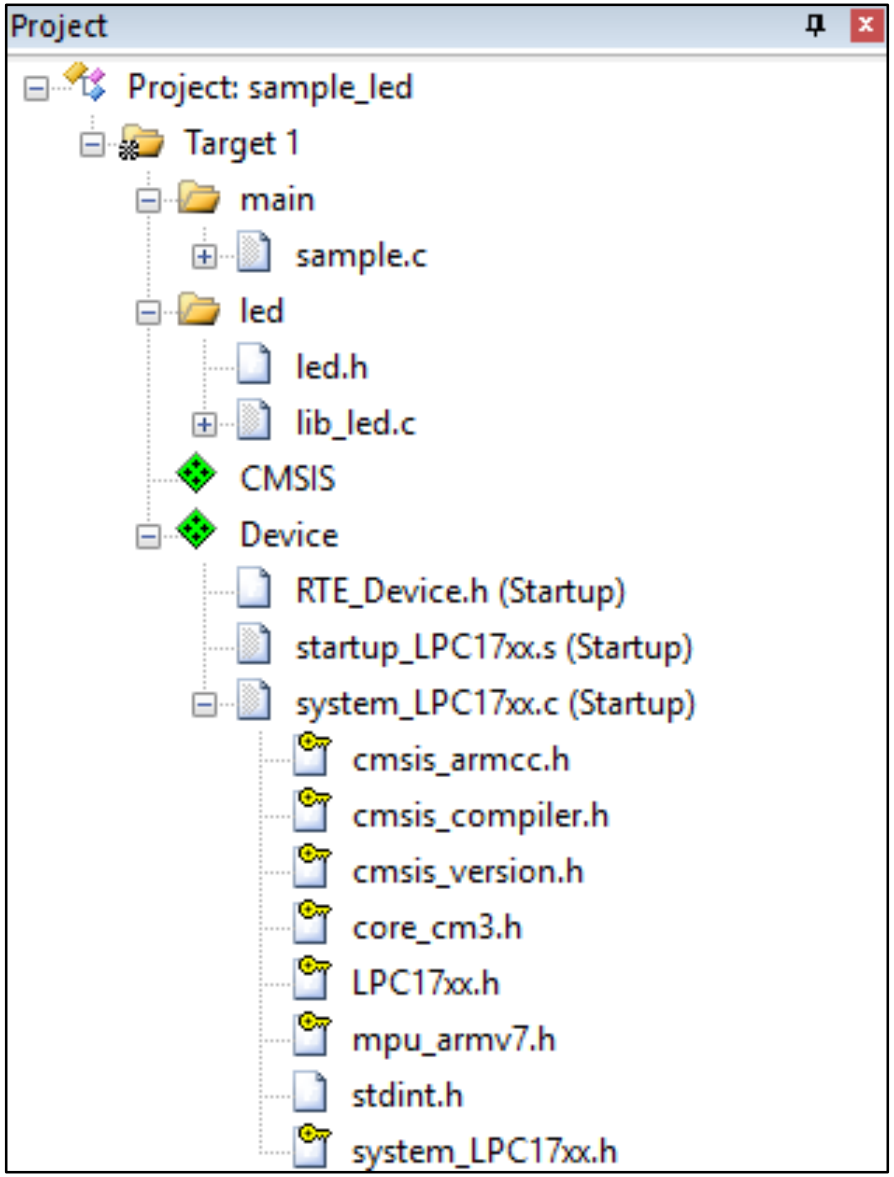
Info	Content
Keywords	LPC1769, LPC1768, LPC1767, LPC1766, LPC1765, LPC1764, LPC1763, LPC1759, LPC1758, LPC1756, LPC1754, LPC1752, LPC1751, ARM, ARM Cortex-M3, 32-bit, USB, Ethernet, CAN, I2S, Microcontroller
Abstract	LPC176x/5x user manual



Creation of sample project

- Add a new group beside the default one
  - Right click on Target 1 -> Add Group
- Optional: rename the groups as *main* and *led*
  - Right click on a group -> Manage Project items
- Copy sample.c and led folder in the project folder; add files to the corresponding group
  - Right click on a group -> Add Existing Files
- Options for target  -> Debug
  - Dialog DLL: DARMP1.DLL (Simulator) or TARMP1.DLL (ULINK2/ME Cortex Debugger)
  - Parameter: -pLPC1768 (both)

Project view



Files in sample projects

- startup_LPC17xx.s
  - stack and heap allocation, IVT, reset_handler
- system_LPC17xx.c
  - setup of clock distribution for the system
  - *SystemInit()* function called by reset_handler
- sample.c
  - *main* function called by reset_handler
- led.c and led.h
  - functions to configure and deal with leds
- system libraries: lpc17xx.h

lpc17xx.h: addressing memory

1) Constants are defined at C language level for addressing main blocks of memory.

```
/*****************************************************
/*                          Peripheral memory map
/*****************************************************
/* Base addresses
#define LPC_FLASH_BASE        (0x00000000UL)
#define LPC_RAM_BASE          (0x10000000UL)
#ifdef __LPC17XX_REV00
#define LPC_AHBRAM0_BASE      (0x20000000UL)
#define LPC_AHBRAM1_BASE      (0x20004000UL)
#else
#define LPC_AHBRAM0_BASE      (0x2007C000UL)
#define LPC_AHBRAM1_BASE      (0x20080000UL)
#endif
#define LPC_GPIO_BASE         (0x2009C000UL)
#define LPC_APB0_BASE         (0x40000000UL)
#define LPC_APB1_BASE         (0x40080000UL)
#define LPC_AHB_BASE          (0x50000000UL)
#define LPC_CM3_BASE          (0xE0000000UL)
```

Memory map (page 14)

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.
0x4000 0000 to 0x5FFF FFFF	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.
	AHB peripherals	0x5000 0000 - 0x501F FFFF	DMA Controller, Ethernet interface, and USB interface.
0xE000 0000 to 0xE00F FFFF	Cortex-M3 Private Peripheral Bus	0xE000 0000 - 0xE00F FFFF	Cortex-M3 related functions, includes the NVIC and System Tick Timer.

lpc17xx.h: addressing peripherals

2) A start address is defined for every peripheral, based on main blocks of memory).

```
/* APB0 peripherals
#define LPC_WDT_BASE          (LPC_APB0_BASE + 0x00000)
#define LPC_TIM0_BASE         (LPC_APB0_BASE + 0x04000)
#define LPC_TIM1_BASE         (LPC_APB0_BASE + 0x08000)
#define LPC_UART0_BASE        (LPC_APB0_BASE + 0x0C000)
#define LPC_UART1_BASE        (LPC_APB0_BASE + 0x10000)
#define LPC_PWM1_BASE         (LPC_APB0_BASE + 0x18000)
#define LPC_I2C0_BASE         (LPC_APB0_BASE + 0x1C000)
#define LPC_SPI_BASE          (LPC_APB0_BASE + 0x20000)
#define LPC_RTC_BASE          (LPC_APB0_BASE + 0x24000)
#define LPC_GPIOINT_BASE      (LPC_APB0_BASE + 0x28080)
#define LPC_PINCON_BASE       (LPC_APB0_BASE + 0x2C000)
#define LPC_SSP1_BASE         (LPC_APB0_BASE + 0x30000)
#define LPC_ADC_BASE          (LPC_APB0_BASE + 0x34000)
#define LPC_CANAF_RAM_BASE    (LPC_APB0_BASE + 0x38000)
#define LPC_CANAF_BASE        (LPC_APB0_BASE + 0x3C000)
#define LPC_CANCR_BASE        (LPC_APB0_BASE + 0x40000)
#define LPC_CAN1_BASE         (LPC_APB0_BASE + 0x44000)
#define LPC_CAN2_BASE         (LPC_APB0_BASE + 0x48000)
#define LPC_I2C1_BASE         (LPC_APB0_BASE + 0x5C000)
```

APB peripheral addresses (page 16)

Table 4. APB0 peripherals and base addresses

APB0 peripheral	Base address	Peripheral name
0	0x4000 0000	Watchdog Timer
1	0x4000 4000	Timer 0
2	0x4000 8000	Timer 1
3	0x4000 C000	UART0
4	0x4001 0000	UART1
5	0x4001 4000	reserved
6	0x4001 8000	PWM1
7	0x4001 C000	I^2C0
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^2C1
24 to 31	0x4006 0000 to 0x4007 C000	reserved

lpc17xx.h: addressing registers

3) A structured list is defined for every peripheral, in order to access its registers.

```
/*------------- Pin Connect Block (PINCON) ----------------------------------*/
/** @brief Pin Connect Block (PINCON) register structure definition */
typedef struct
{
  __IO uint32_t PINSEL0;
  __IO uint32_t PINSEL1;
  __IO uint32_t PINSEL2;
  __IO uint32_t PINSEL3;
  __IO uint32_t PINSEL4;
  __IO uint32_t PINSEL5;
  __IO uint32_t PINSEL6;
  __IO uint32_t PINSEL7;
  __IO uint32_t PINSEL8;
  __IO uint32_t PINSEL9;
  __IO uint32_t PINSEL10;
       uint32_t RESERVED0[5];
  __IO uint32_t PINMODE0;
  __IO uint32_t PINMODE1;
  __IO uint32_t PINMODE2;
  __IO uint32_t PINMODE3;
  __IO uint32_t PINMODE4;
  __IO uint32_t PINMODE5;
  __IO uint32_t PINMODE6;
  __IO uint32_t PINMODE7;
  __IO uint32_t PINMODE8;
  __IO uint32_t PINMODE9;
  __IO uint32_t PINMODE_OD0;
  __IO uint32_t PINMODE_OD1;
  __IO uint32_t PINMODE_OD2;
  __IO uint32_t PINMODE_OD3;
  __IO uint32_t PINMODE_OD4;
  __IO uint32_t I2CPADCFG;
} LPC_PINCON_TypeDef;
```

System libraries: lpc17xx.h (III)

```
989  /******************************************************************************/
990  /*                        Peripheral declaration                              */
991  /******************************************************************************/
992  #define LPC_SC              ((LPC_SC_TypeDef        *) LPC_SC_BASE       )
993  #define LPC_GPIO0           ((LPC_GPIO_TypeDef      *) LPC_GPIO0_BASE    )
994  #define LPC_GPIO1           ((LPC_GPIO_TypeDef      *) LPC_GPIO1_BASE    )
995  #define LPC_GPIO2           ((LPC_GPIO_TypeDef      *) LPC_GPIO2_BASE    )
996  #define LPC_GPIO3           ((LPC_GPIO_TypeDef      *) LPC_GPIO3_BASE    )
997  #define LPC_GPIO4           ((LPC_GPIO_TypeDef      *) LPC_GPIO4_BASE    )
998  #define LPC_WDT             ((LPC_WDT_TypeDef       *) LPC_WDT_BASE      )
999  #define LPC_TIM0            ((LPC_TIM_TypeDef       *) LPC_TIM0_BASE     )
1000 #define LPC_TIM1            ((LPC_TIM_TypeDef       *) LPC_TIM1_BASE     )
1001 #define LPC_TIM2            ((LPC_TIM_TypeDef       *) LPC_TIM2_BASE     )
1002 #define LPC_TIM3            ((LPC_TIM_TypeDef       *) LPC_TIM3_BASE     )
1003 #define LPC_RIT             ((LPC_RIT_TypeDef       *) LPC_RIT_BASE      )
1004 #define LPC_UART0           ((LPC_UART_TypeDef      *) LPC_UART0_BASE    )
1005                                                   Def
1006                                                   ef
1007                                                   ef
1008                                                   f
1009                                                   f
1010 #define LP      C1          ((LPC_I      peDef      *)      I2C1_BASE    )
1011 #define LPC     C2          ((LPC_I2     peDef      *)      I2C2_BASE    )
1012 #define LPC     S           ((LPC_I2S    peDef      *) I    I2S_BASE     )
1013 #define LPC_S   I           ((LPC_SPI    peDef      *) L    SPI_BASE     )
1014 #define LPC_R   C           ((LPC_RTC_T  peDef      *) LP   RTC_BASE     )
1015 #define LPC_GPIOINT         ((LPC_GPIOINT_TypeDef   *) LPC_GPIOINT_BASE  )
1016 #define LPC_PINCON          ((LPC_PINCON_TypeDef    *) LPC_PINCON_BASE   )
1017 #define LPC_SSP0            ((LPC_SSP_TypeDef       *) LPC_SSP0_BASE     )
```

Accessible name from C code

Cast as a pointer to the relative type

Address of every specific SoC resource

Pin connect block (page 117)

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^2C Pin Configuration register	R/W	0	0x4002 C07C

```
/*-------------- Pin Connect Blo
/** @brief Pin Connect Block (P
typedef struct
{
  __IO uint32_t PINSEL0;
  __IO uint32_t PINSEL1;
  __IO uint32_t PINSEL2;
  __IO uint32_t PINSEL3;
  __IO uint32_t PINSEL4;
  __IO uint32_t PINSEL5;
  __IO uint32_t PINSEL6;
  __IO uint32_t PINSEL7;
  __IO uint32_t PINSEL8;
  __IO uint32_t PINSEL9;
  __IO uint32_t PINSEL10;
       uint32_t RESERVED0[5];
  __IO uint32_t PINMODE0;
  __IO uint32_t PINMODE1;
  __IO uint32_t PINMODE2;
  __IO uint32_t PINMODE3;
  __IO uint32_t PINMODE4;
  __IO uint32_t PINMODE5;
  __IO uint32_t PINMODE6;
  __IO uint32_t PINMODE7;
  __IO uint32_t PINMODE8;
  __IO uint32_t PINMODE9;
  __IO uint32_t PINMODE_OD0;
  __IO uint32_t PINMODE_OD1;
  __IO uint32_t PINMODE_OD2;
  __IO uint32_t PINMODE_OD3;
  __IO uint32_t PINMODE_OD4;
  __IO uint32_t I2CPADCFG;
} LPC_PINCON_TypeDef;
```

base
+4
+4
+4
.....

Use of definitions in lpc17xx.h

- Symbols defined in lpc17xx.h simplify addressing peripheral registers in the project.
- Example in lib_led.c:

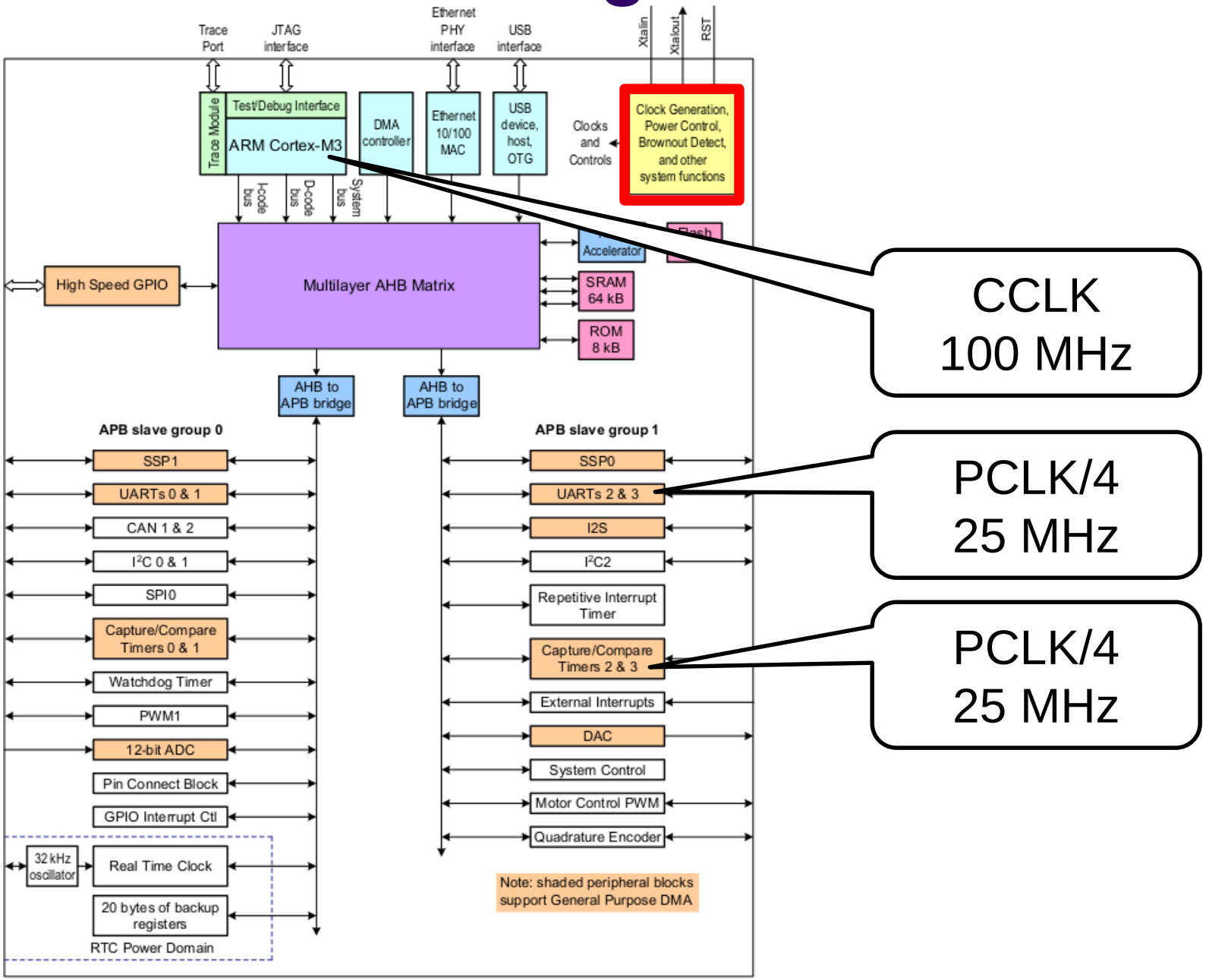
```
LPC_PINCON->PINSEL4 &= 0xFFFF0000;
```

Peripheral group name

Specific register of the peripheral

Elaboration

Clock configuration



Clock selection

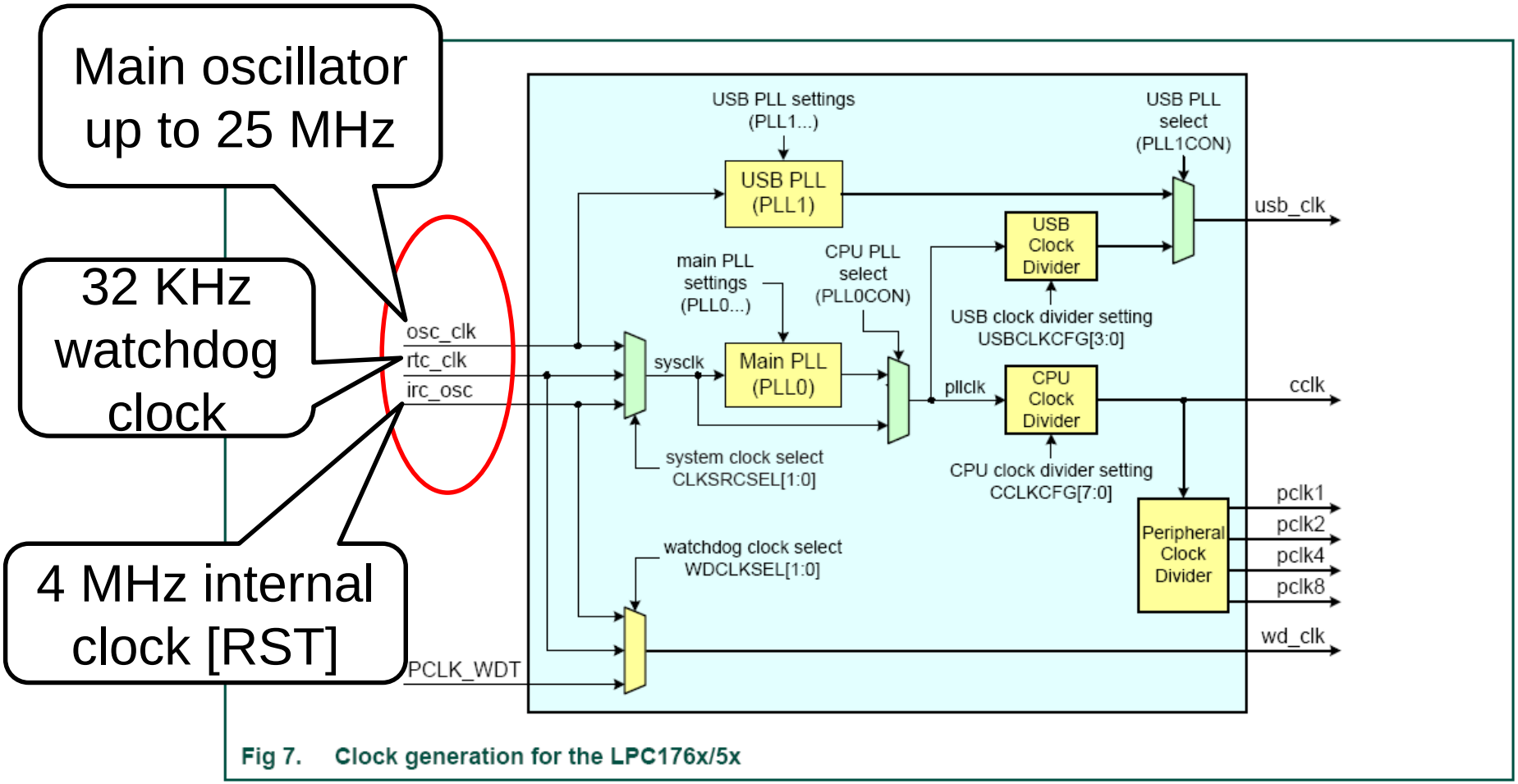
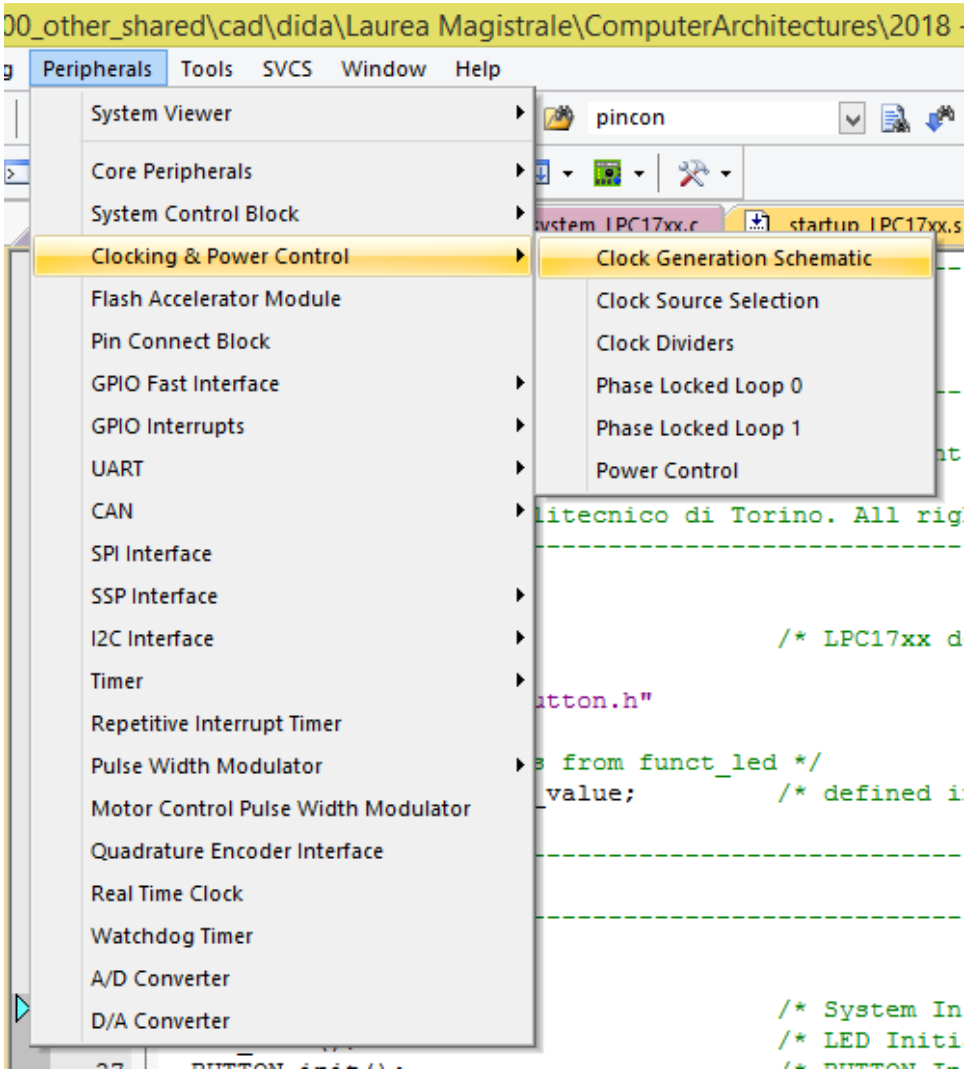
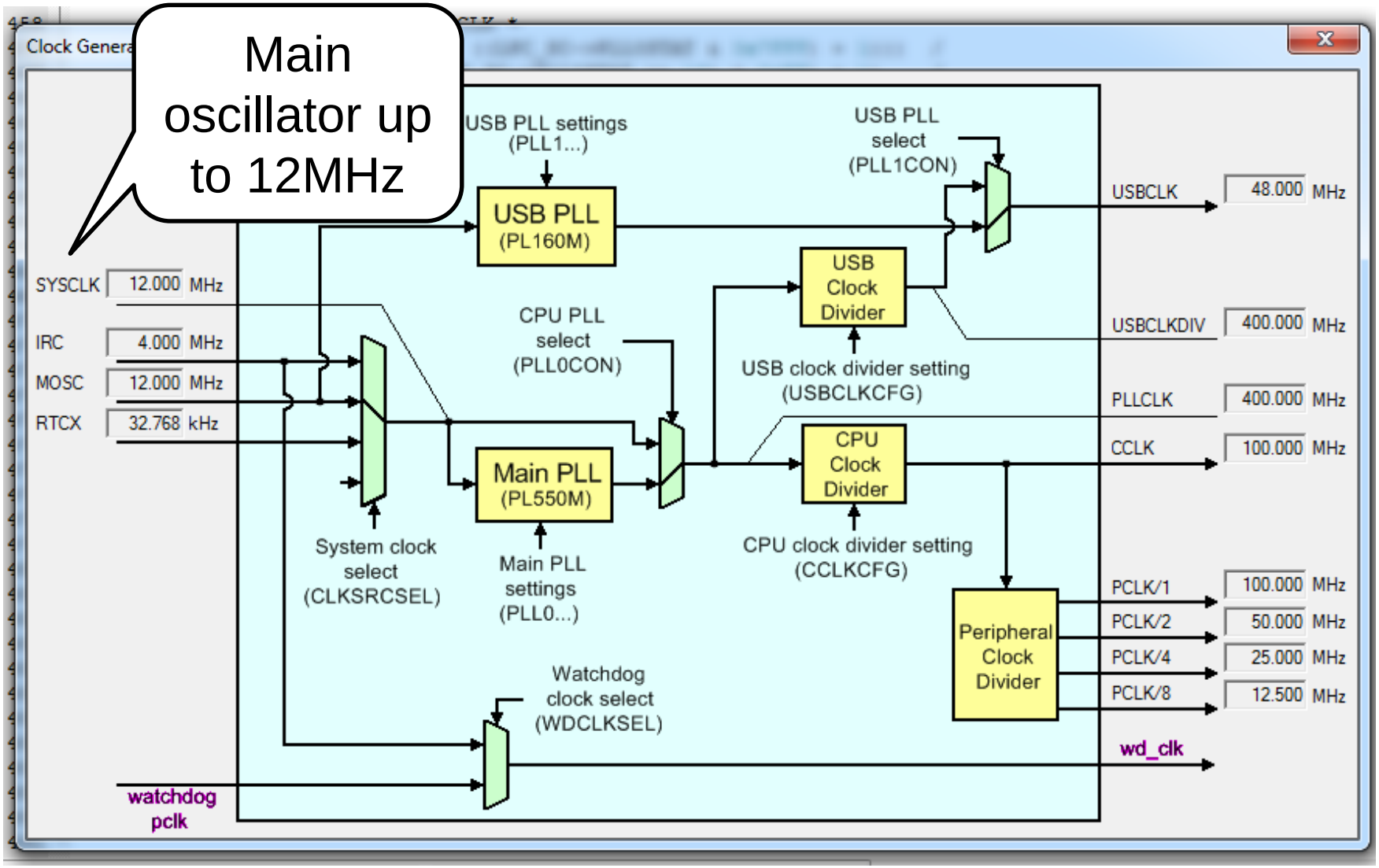


Fig 7. Clock generation for the LPC176x/5x

Debug view


00_other_shared\cad\dida\Laurea Magistrale\ComputerArchitectures\2018
Peripherals
Tools
SVCS
Window
Help
System Viewer
Core Peripherals
System Control Block
Clocking & Power Control
Flash Accelerator Module
Pin Connect Block
GPIO Fast Interface
GPIO Interrupts
UART
CAN
SPI Interface
SSP Interface
I2C Interface
Timer
Repetitive Interrupt Timer
Pulse Width Modulator
Motor Control Pulse Width Modulator
Quadrature Encoder Interface
Real Time Clock
Watchdog Timer
A/D Converter
D/A Converter
Clock Generation Schematic
Clock Source Selection
Clock Dividers
Phase Locked Loop 0
Phase Locked Loop 1
Power Control
pincon
/* LPC17xx d
/* System In
/* LED Initi

Clock values after *SystemInit()*



system_LPC17xx.c: text view

startup_LPC17xx.s | lib_led.c | system_LPC17xx.c | startup_LPC17xx.s | sample.c | core_cm3.c

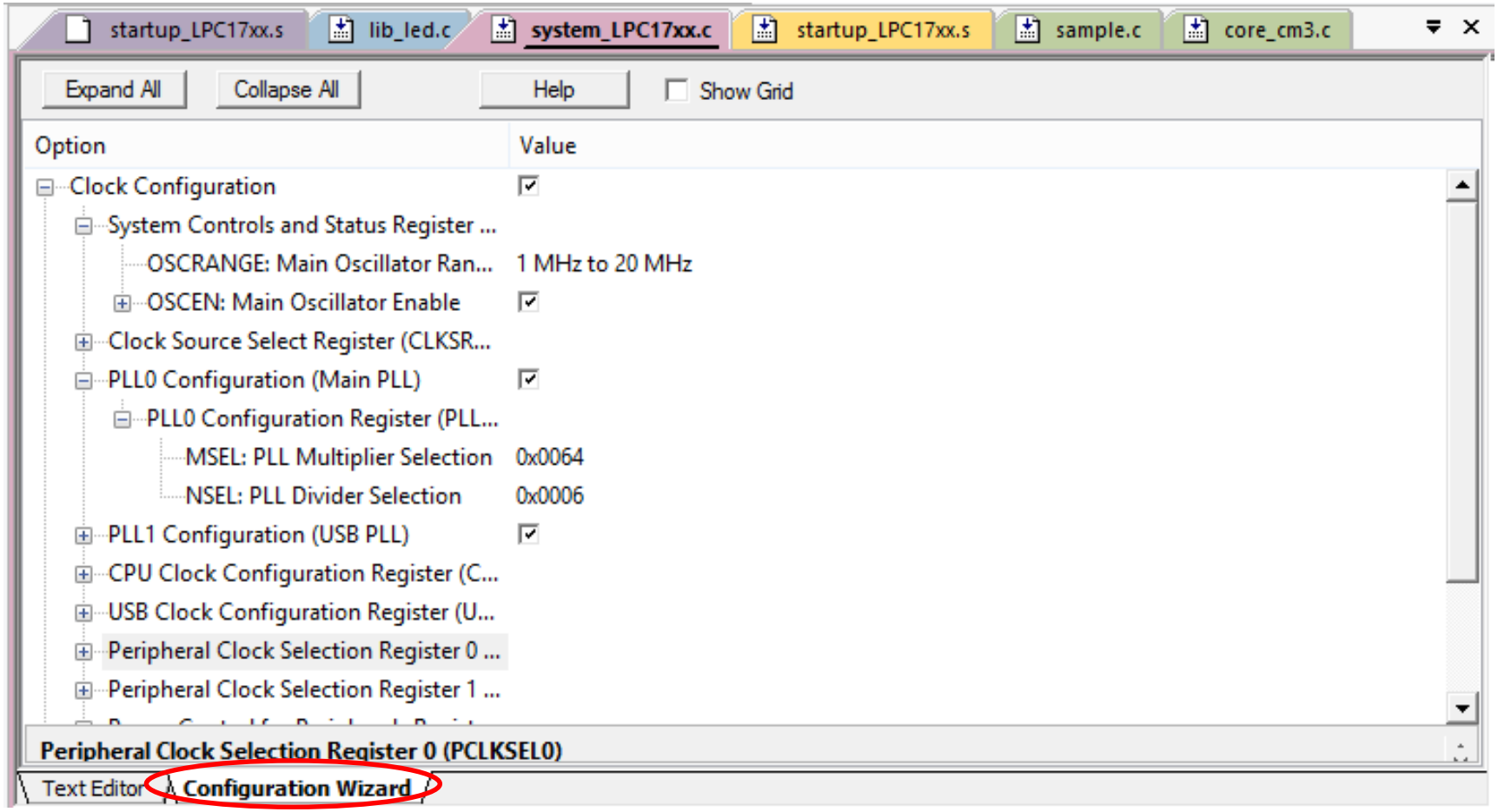
```c
/**
 * Initialize the system
 *
 * @param  none
 * @return none
 *
 * @brief  Setup the microcontroller system.
 *         Initialize the System and update the SystemFrequency variable.
 */
void SystemInit (void)
{
#if (CLOCK_SETUP)                          /* Clock Setup                        */
  LPC_SC->SCS       = SCS_Val;
  if (SCS_Val & (1 << 5)) {                /* If Main Oscillator is enabled      */
    while ((LPC_SC->SCS & (1<<6)) == 0);/* Wait for Oscillator to be ready    */
  }

  LPC_SC->CCLKCFG   = CCLKCFG_Val;         /* Setup Clock Divider                */

  LPC_SC->PCLKSEL0  = PCLKSEL0_Val;        /* Peripheral Clock Selection         */
  LPC_SC->PCLKSEL1  = PCLKSEL1_Val;
```

Text Editor | Configuration Wizard

system_LPC17xx.c: wizard



Configuration Wizard Annotations

配置注释向导

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

Configuration Wizard Annotations

- The Configuration Wizard section must begin within the first 100 lines of code and must start with:

  // <<< Use Configuration Wizard in Context Menu >>>

- The optional end of the Configuration Wizard section is :

  // <<< end of configuration section >>>

- Annotations are written as comments in the code: it 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.

Configuration Wizard Annotations

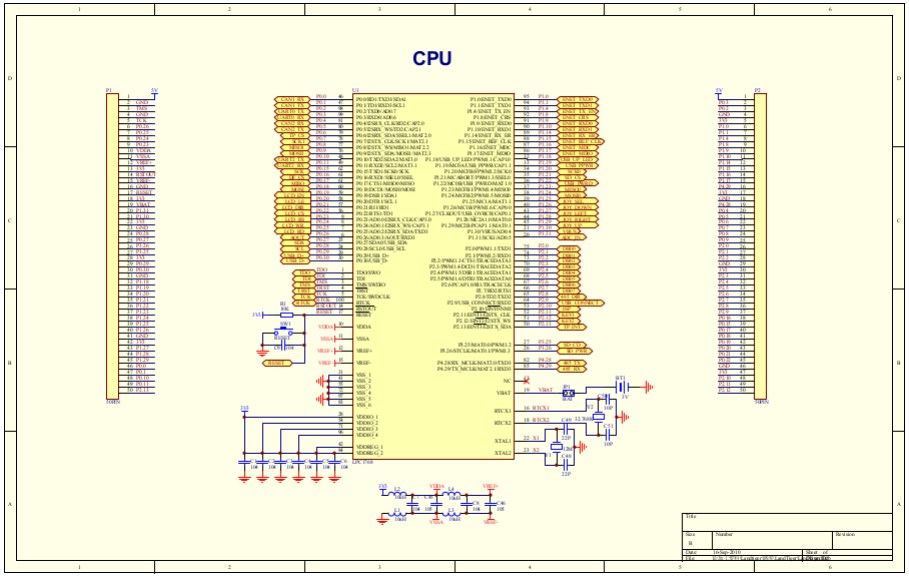
- <h> : *Heading*. Creates a header section. All items and options enclosed by <h> and </h> belong to one group and can be expanded. This entry makes no changes to code symbols. It is just used to group other items and modifiers.
- <e>* : *Heading with Enable*. Creates a header section with a checkbox to enabled or disabled 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

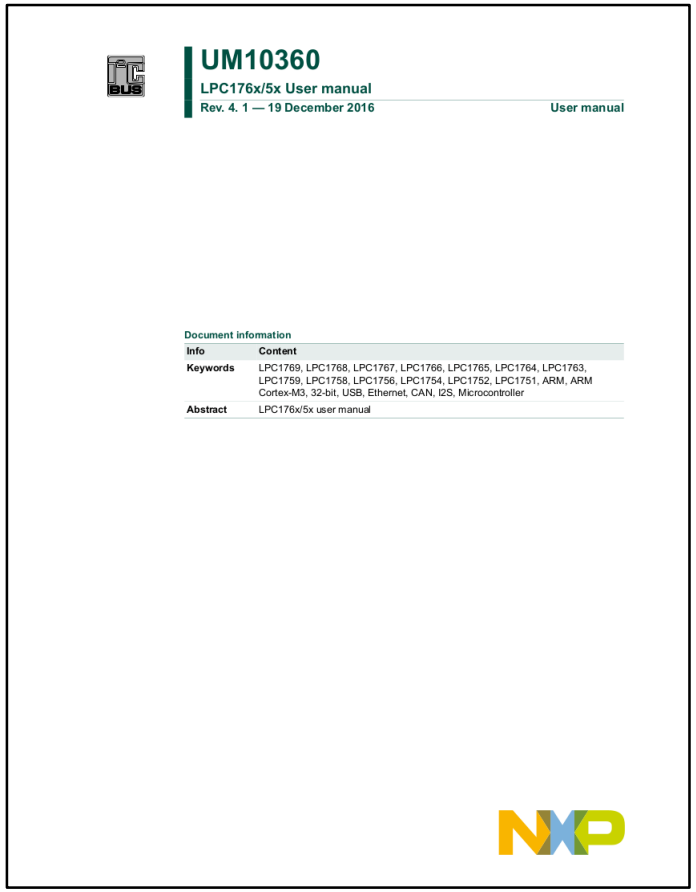
Configuration Wizard Annotations

- <o>* : Option with selection or number entry.
  - // <o>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].
- <o.*x*..*y*>* : Option Modify a range of bits. (example: <o.4..5> - bit 4 to 5).
  - // <o.0..15>Language ID <0x0000-0xFCFF>
- <o*i*> *:* Skip *i* items. Can be applied to all annotation items marked with a *

Available resources

- User manual
- Sample project
- **Schematic**





UM10360

LPC176x/5x User manual

Rev. 4. 1 — 19 December 2016 User manual

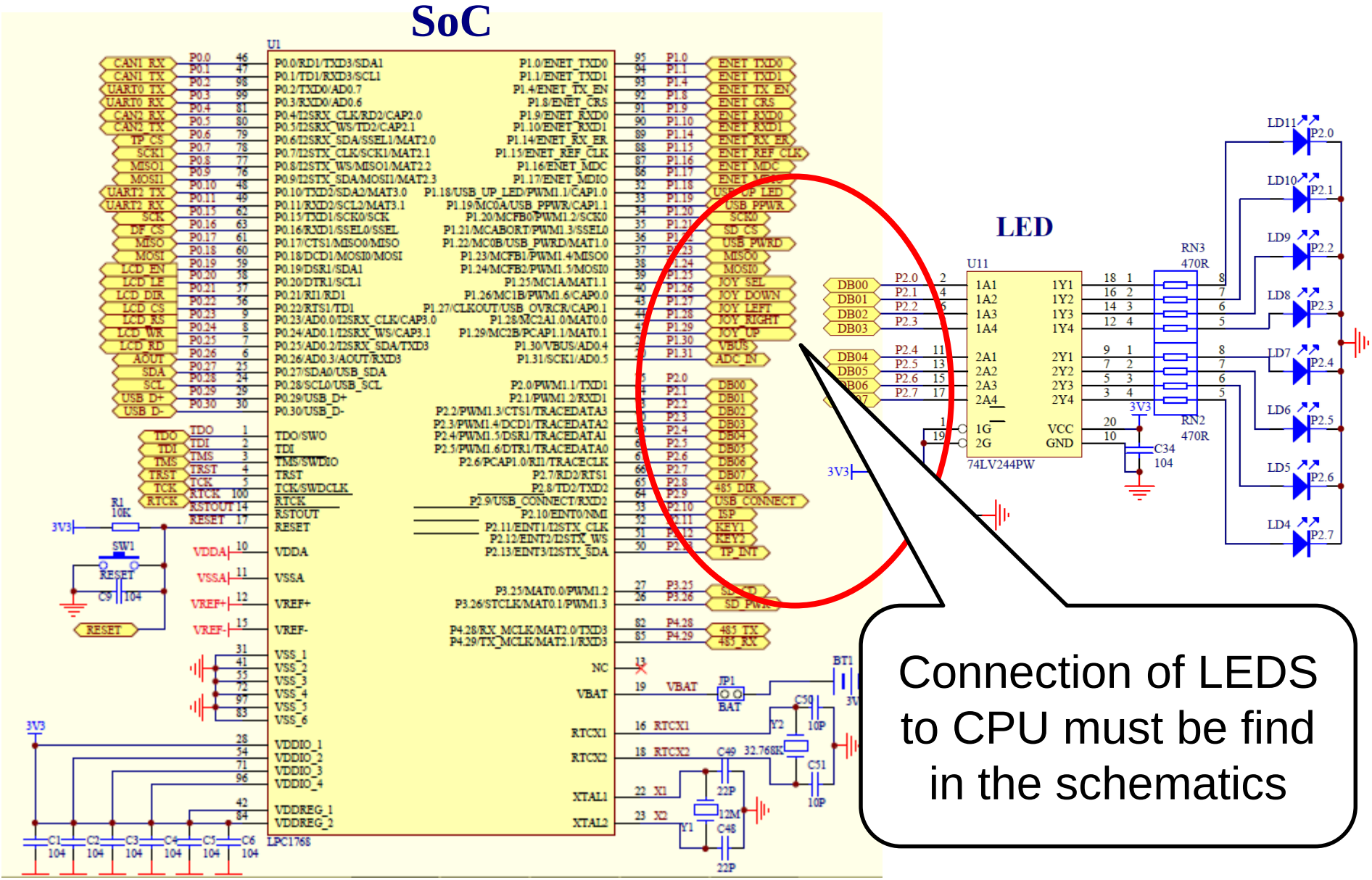
Document information

Info	Content
Keywords	LPC1769, LPC1768, LPC1767, LPC1766, LPC1765, LPC1764, LPC1763, LPC1759, LPC1758, LPC1756, LPC1754, LPC1752, LPC1751, ARM, ARM Cortex-M3, 32-bit, USB, Ethernet, CAN, I2S, Microcontroller
Abstract	LPC176x/5x user manual

Board schematic

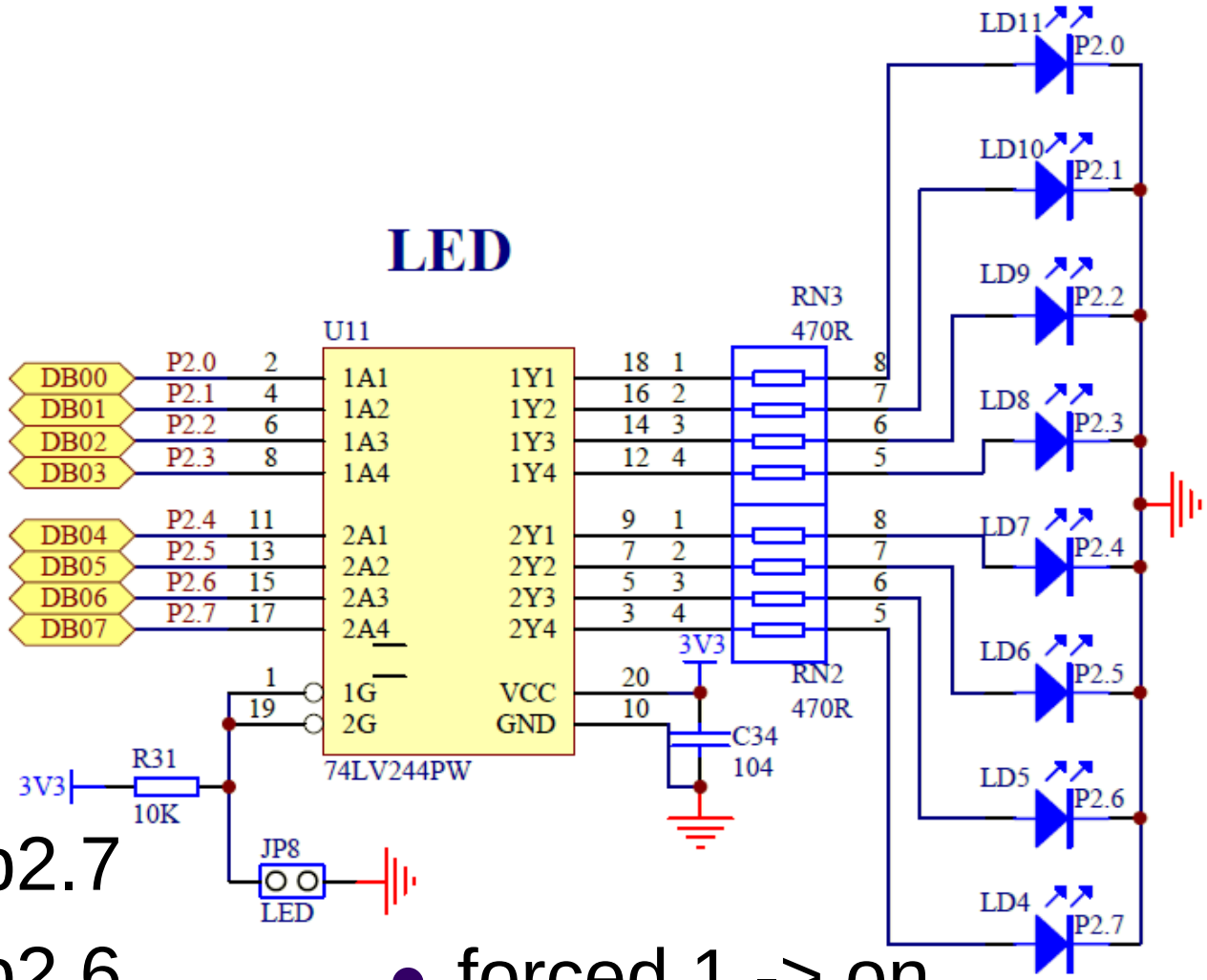
- The user manual of the SoC does not describe:
  - the list of components in the board
  - how the board components are connected to the SoC.
- This information can be obtained only by reading the board schematic.

Example



Connection of LEDS to CPU must be find in the schematics

Leds

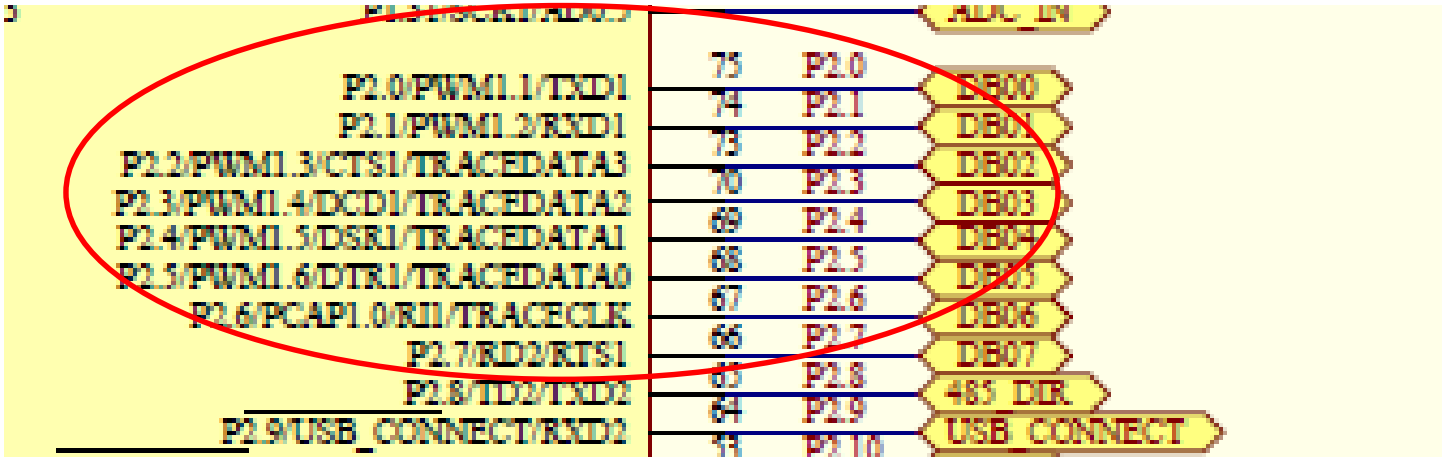


- Led4 -> p2.7
- Led5 -> p2.6
- Led11-> p2.0

- forced 1 -> on
- forced 0 -> off

Pin selection and direction

- After discovering the interested CPU pins, you have to provide additional information to the SoC:
  - the pin functionality
  - the pin direction (input or output)



Pin connect block (page 114)

Table 75. Summary of PINSEL registers

Register	Controls	Table
PINSEL0	P0[15:0]	Table 80
PINSEL1	P0 [31:16]	Table 81
PINSEL2	P1 [15:0] (Ethernet)	Table 82
PINSEL3	P1 [31:16]	Table 83
PINSEL4	P2 [15:0]	Table 84
PINSEL5	P2 [31:16]	not used
PINSEL6	P3 [15:0]	not used
PINSEL7	P3 [31:16]	Table 85
PINSEL8	P4 [15:0]	not used
PINSEL9	P4 [31:16]	Table 86
PINSEL10	Trace port enable	Table 87

Bits of PINSEL4 (page 120)

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 (page 132)

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 register (page 133)

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.	

lib_led.c: LED_init

```
void LED_init(void)
{
    // PIN mode GPIO:
    // P2.0-P2.7 are set to zero
    LPC_PINCON->PINSEL4 &= 0xFFFF0000;
    // P2.0-P2.7 on PORT2 defined as
    // Output
    LPC_GPIO2->FIODIR |= 0x000000FF;
}
```

lib_led.c: all LED on and off

```
void all_LED_on(void)
{
    LPC_GPIO2->FIOSET = 0x000000FF;
}

void all_LED_off(void)
{
    LPC_GPIO2->FIOCLR = 0x000000FF;
}
```

Check led status with software debug