PINSEL
#include "lpc17xx_pinsel.h"

PINSEL_CFG_Type pin_x;
- Portnum
- Pinnum
- Pinmode //PINSEL_PINMODE_PU PD TS
- Funcnum //
- OpenDrain
PINSEL_PINMODE_NORMAL/OPENDRAIN
PINSEL_ConfigPin(&x)

EXTINT
#include "lpc17xx_exti.h"

void EXTI_SetMode(EXTI_EINTX, mode)
Mode:
EXTI_MODE_LEVEL _SENSITIVE
EXTI_MODE_ EDGE_SENSITIVE
void EXTI_SetPolarity(EXTI_EINTX, polarity))
polarity:
EXTI_POLARITY_HIGH_ACTIVE_OR_RISING_EDGE

PCLK
#include "lpc17xx_exti.h"
void CLKPWR_SetPCLKDiv (uint32_t ClkType, uint32_t DivVal)
ClkType:
CLKPWR_PCLKSEL_Periferico
DivVal:
CLKPWR_PCLKSEL_CCLK_DIV_8 (3)
CLKPWR_PCLKSEL_CCLK_DIV_4 (0)
CLKPWR_PCLKSEL_CCLK_DIV_2 (1)
CLKPWR_PCLKSEL_CCLK_DIV_1 (2)
uint32_t CLKPWR_GetPCLKSEL (uint32_t ClkType)
CLKPWR_GetPCLK (uint32_t ClkType) //este

GPIO
#include "lpc17xx_gpio.h"

void GPIO_SetDir(uint8_t portNum, uint32_t bitValue, uint8_t dir)
void GPIO_SetValue(uint8_t portNum, uint32_t bitValue)
void GPIO_ClearValue(uint8_t portNum, uint32_t bitValue)
uint32_t GPIO_ReadValue(uint8_t portNum)
GPIO_IntCmd(uint8_t portNum, uint32_t bitValue, uint8_t edgeState)
- 0: Rising edge
- 1: Falling edge
void GPIO_ClearInt(uint8_t portNum, uint32_t bitValue)
void FIO_SetMask(uint8_t portNum, uint32_t bitValue, uint8_t maskValue)
void FIO_HalfWordSetDir(uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue, uint8_t dir)
halfwordNum: 0 (lower) or 1(upper)
bitValue: los 16 bits afectados
void FIO_HalfWordSetMask(uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue, uint8_t maskValue)
void FIO_HalfWordSetValue(uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue)
void FIO_HalfWordClearValue(uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue)
uint16_t FIO_HalfWordReadValue(uint8_t portNum, uint8_t halfwordNum)
void FIO_ByteSetDir(uint8_t portNum, uint8_t byteNum, uint8_t bitValue, uint8_t dir)
void FIO_ByteSetMask(uint8_t portNum, uint8_t byteNum, uint8_t bitValue, uint8_t maskValue)
void FIO_ByteSetValue(uint8_t portNum, uint8_t byteNum, uint8_t bitValue)
void FIO_ByteClearValue(uint8_t portNum, uint8_t byteNum, uint8_t bitValue)
uint8_t FIO_ByteReadValue(uint8_t portNum, uint8_t byteNum)

TIMER
#include "lpc17xx_timer.h"

TIM_TIMERCFG_Type PrescaleOption (TIM_PRESCALE_TICKVAL, TIM_PRESCALE_USVAL), PrescaleValue
TIM_MATCHCFG_Type MatchChannel IntOnMatch ResetOnMatch StopOnMatch ExtMatchOutputType MatchValue
TIM_EXTMATCH_NOTHING
TIM_EXTMATCH_LOW:
TIM_EXTMATCH_HIGH:
TIM_EXTMATCH_TOGGLE:

TIM_CAPTURECFG_Type CaptureChannel (0 o 1) – RisingEdge – FallingEdge - IntOnCaption

TIM_Init(LPC_TIMX, mode, &TIM_TIMERCFG_Type) //setea PCLK a CCLK/4 //TIMx->PR = pTimeCfg->PrescaleValue -1 ;
Mode:
- TIM_TIMER_MODE: Timer mode
- TIM_COUNTER_RISING_MODE: Counter rising mode
- TIM_COUNTER_FALLING_MODE: Counter falling mode
- TIM_COUNTER_ANY_MODE:Counter on both edges

TIM_ConfigMatch(LPC_TIMX, & TIM_MATCHCFG_Type)
TIM_ConfigCapture(LPC_TIMX, & TIM_CAPTURECFG_Type)
TIM_GetCaptureValue(LPC_TIMX, CaptureChannel)
CaptureChannel:
TIM_COUNTER_INCAP0
TIM_COUNTER_INCAP1
TIM_Cmd(LPC_TIMX, ENABLE)
TIM_ResetCounter(LPC_TIMX)
TIM_GetIntStatus(LPC_TIMX, channelNumber)
channelNumber:
TIM_MR0_INT
TIM_MR1_INT
TIM_MR2_INT
TIM_MR3_INT
TIM_CR0_INT
TIM_CR1_INT
void TIM_ClearIntPending(LPC_TIMX, TIM_INT_TYPE IntFlag)
TIM_MR0_INT: Interrupt for Match channel 0
TIM_MR1_INT: Interrupt for Match channel 1
TIM_MR2_INT: Interrupt for Match channel 2
TIM_MR3_INT: Interrupt for Match channel 3
TIM_CR0_INT: Interrupt for Capture channel 0
TIM_CR1_INT: Interrupt for Capture channel 1

ADC
#include "lpc17xx_adc.h"

ADC_Init(LPC_ADC, rate); //ajusta los bits CLKDIV (divisor del ADC) según temp = (CLKPWR_GetPCLK() /(rate * 65)) - 1;
//tiene que ser <= 200KHz
ADC_IntConfig(LPC_ADC, ADC_ADINTENX, ENABLE); //configura interrupcion por canal 0-7
ADC_ChannelCmd(LPC_ADC, channelNumber, ENABLE); //habilita canal 0-7
ADC_BurstCmd(LPC_ADC, 1); //1: Set Burst mode //si no se usa en modo burst usar **ADC_StartCmd()** en main
ADC_EdgeStartConfig(LPC_ADC, EDGE) // 0-> Rising, 1-> Falling USAR SOLO EN MODO NO BURST, y NO START NOW
ADC_StartCmd(LPC_ADC, START_MODE) //USAR SOLO EN MODO NO BURST START MODE entre 0-7
START_MODE:
- ADC_START_CONTINUOUS
- ADC_START_NOW
- ADC_START_ON_EINT0
- ADC_START_ON_CAP01
- ADC_START_ON_MAT01
- ADC_START_ON_MAT03
- ADC_START_ON_MAT10
- ADC_START_ON_MAT11
ADC_ChannelGetStatus(LPC_ADC, channelNumber, ADC_DATA_DONE) //channel 0-7
ADC_ChannelGetData((LPC_ADC, channelNumber) //channel 0-7
NVIC_EnableIRQ(ADC_IRQn);
NVIC_DisableIRQ(ADC_IRQn);

DAC
#include "lpc17xx_dac.h"

void DAC_Init(LPC_DAC) //setea bias con max corriente (700uA)
void DAC_UpdateValue(LPC_DAC)
void DAC_SetBias(LPC_DAC,uint32_t bias)
bias : 0 is 700 uA , 1MHz | 1 is 350 uA 400KHz
void DAC_ConfigDAConverterControl (LPC_DAC ,DAC_CONVERTER_CFG_Type)
- DBLBUF_ENA : enable/disable DACR double buffering feature
- CNT_ENA : enable/disable timer out counter
- DMA_ENA : enable/disable DMA access
void DAC_SetDMATimeOut(LPC_DAC, uint32_t time_out) // time out to reload for interrupt/DMA counter
//time out, tiempo al que desborda el timer del dac para que el dma sepa cuando mandar una nueva muestra al dac
//tiene que ser mayor que el tiempo de establecimiento del dac

DMA
#include "lpc17xx_gpdma.h"
GPDMA_LLY_Type name;
Name.SrcAddr = uint32_t(direccionOrigen) //puede ser nombre de arreglo
Name.DstAddr uint32_t(direccionDest) //puede ser el registro DACR donde esta el value que se va a sacar por un AOUT
Name.NextLLI = (uint_32) &name
Name.Control = //configurar transfer size (cant de datos a transf), SBsize DBsize, Swidth Dwidth, Si, Di //table 564
//no tiene function propia, se usa como posible valor del campo DMALLI de la estructura de configuración de canal

GPDMA_Init() //inicia el controlador GPDMA
GPDMA_Channel_CFG_Type nombre
Nombre.ChannelNum = // 0-7
Nombre.SrcMemAddr = //puede ser nombre de arreglo
Nombre.DstMemAddr = 0 //el 0 indica que el destino es un periferico, no otra pos de memoria, en la lista previa se debe definir el perif de destino de los dato
Nombre.TransferSIZE = //mismo configurado en los bits de control de la estrucuura previa
Nombre.TransferWidth = 0 // solo usado para transferencia M2M
Nombre.TransferType = **GPDMA_TRANSFERTYPE**_M2P, M2M, P2M, P2P
Nombre.SrcConn = 0 //puede ser un periferico
Nombre.DstConn = // puede ser un periferico
Nombre.DMALLI = //puede ser la dirección a la estructura previa (si se esta trabajando con una lista), de lo contrario va en 0
GPDMA_Setup(&nombre)
//los campos de config que no se usan van en 0

GPDMA_ChannelCmd(0, ENABLE) // habilita el DMA, usar luego de haber configurado todo

GPDMA_CONN_SSP0_Tx
GPDMA_CONN_SSP0_Rx
GPDMA_CONN_SSP1_Tx
GPDMA_CONN_SSP1_Rx
GPDMA_CONN_ADC
GPDMA_CONN_I2S_Channel_0
GPDMA_CONN_I2S_Channel_1
GPDMA_CONN_DAC
GPDMA_CONN_UART0_Tx
GPDMA_CONN_UART0_Rx
GPDMA_CONN_UART1_Tx
GPDMA_CONN_UART1_Rx
GPDMA_CONN_UART2_Tx
GPDMA_CONN_UART2_Rx
GPDMA_CONN_UART3_Tx
GPDMA_CONN_UART3_Rx
GPDMA_CONN_MAT0_0
GPDMA_CONN_MAT0_1
GPDMA_CONN_MAT1_0
GPDMA_CONN_MAT1_1
GPDMA_CONN_MAT2_0
GPDMA_CONN_MAT2_1
GPDMA_CONN_MAT3_0
GPDMA_CONN_MAT3_1