```
#include "lpc17xx_dac.h"
#include "lpc17xx_libcfg_default.h"
#include "lpc17xx_pinsel.h"
#include "lpc17xx_gpdma.h"

#define DMA_SIZE                60
#define NUMERO_MUESTRAS 60
#define FRECUENCIA_SENOIDAL     60
#define CLOCK_DAC_MHZ   25 //CCLK divided by 4

GPDMA_Channel_CFG_Type GPDMACfg;        //Estructura de configuracion
del DMA


int main(void)
{
        PINSEL_CFG_Type PinCfg;
        DAC_CONVERTER_CFG_Type DAC_Struct;
        GPDMA_LLI_Type DMA_LLI_Struct;
        uint32_t tmp;
        uint32_t i;
        uint32_t seno_0_a_90[16]={\
                        0,1045,2079,3090,4067,\
                        5000,5877,6691,7431,8090,\
                        8660,9135,9510,9781,9945,10000\
        };

        uint32_t TABLA_DAC[NUMERO_MUESTRAS];


        PinCfg.Funcnum = 2;
        PinCfg.OpenDrain = 0;
        PinCfg.Pinmode = 0;
        PinCfg.Pinnum = 26;
        PinCfg.Portnum = 0;
        PINSEL_ConfigPin(&PinCfg);


        //
        for(i=0;i<NUMERO_MUESTRAS;i++)
        {
                if(i<=15)       //De 0 a 90°
                {
                        TABLA_DAC[i] = 512 + 512*seno_0_a_90[i]/10000;
                        if(i==15) TABLA_DAC[i]= 1023;
                }
                else if(i<=30)  //De 90° a 180°
                {
                        TABLA_DAC[i] = 512 + 512*seno_0_a_90[30-
i]/10000;
                }
                else if(i<=45)  //De 180° a 270°
                {
                        TABLA_DAC[i] = 512 - 512*seno_0_a_90[i-
30]/10000;
                }
```

```
                else    //De 270° a 360°
                {
                        TABLA_DAC[i] = 512 - 512*seno_0_a_90[60-
i]/10000;
                }
                TABLA_DAC[i] = (TABLA_DAC[i]<<6);
        }

        //Configuracion de la lista del DMA
        DMA_LLI_Struct.SrcAddr= (uint32_t)TABLA_DAC;    //Direccion de
los datos fuente
        DMA_LLI_Struct.DstAddr= (uint32_t)&(LPC_DAC->DACR);
//Destino: DAC
        DMA_LLI_Struct.NextLLI= (uint32_t)&DMA_LLI_Struct;      //Solo
un juego de datos
        DMA_LLI_Struct.Control= DMA_SIZE
                                                                | (2<<18)
//Fuente: 32bits
                                                                | (2<<21)
//Destino: 32bit
                                                                | (1<<26)
//Incremento automático de la fuente
                                                                ;

        GPDMA_Init();   //Inicializa el modulo DMA

        GPDMACfg.ChannelNum = 0;        //Canal 0
        GPDMACfg.SrcMemAddr = (uint32_t)(TABLA_DAC);    //Origen
        GPDMACfg.DstMemAddr = 0;        //Destino = es un periferico,
no es memoria
        GPDMACfg.TransferSize = DMA_SIZE;       //Tamaño de la
transferencia
        GPDMACfg.TransferWidth = 0;     //No usado
        GPDMACfg.TransferType = GPDMA_TRANSFERTYPE_M2P; //Tipo de
transferencia = Memory 2 Peripheral
        GPDMACfg.SrcConn = 0;   //La fuente es memoria => no
connection
        GPDMACfg.DstConn = GPDMA_CONN_DAC;      //Destino : conexión
al DAC
        //GPDMACfg.DMALLI = 0;  //Lista de enlace del DMA
        GPDMACfg.DMALLI = (uint32_t)&DMA_LLI_Struct;    //Lista de
enlace del DMA

        GPDMA_Setup(&GPDMACfg); //Configura el DMA

        DAC_Struct.CNT_ENA =SET;                //Activa el modo
timeout
        DAC_Struct.DMA_ENA = SET;               //Activa el modo
        DAC_Init(LPC_DAC);

        //Calculamos el tiempo de actualizacion de muestra

        tmp =
(CLOCK_DAC_MHZ*1000000)/(FRECUENCIA_SENOIDAL*NUMERO_MUESTRAS);
        DAC_SetDMATimeOut(LPC_DAC,tmp); //Fija el timeout, separacion
entre muestras
        DAC_ConfigDAConverterControl(LPC_DAC, &DAC_Struct);
```

```
//Configura el DAC

        GPDMA_ChannelCmd(0, ENABLE);    //Enciende el modulo DMA

//      __asm(  "halt:"
//                      "WFI \n"
//                      "B halt");


        return 1;
}
```