[14:01, 22/10/2019] Angeloff: //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
                                                        ;
```

[14:01, 22/10/2019] Angeloff: const uint32_t origen [514]= {0x10000800};

```
const uint32_t destino [514]={0x10002800};
volatile uint32_t Channel0_TC;
volatile uint32_t Channel0_Err;

int main(void)
{
	uint8_t i=0;
	confDMA();
	Channel0_TC = 0;
	Channel0_Err = 0;
	for(i=0;i<514;i++){
		origen[i]= 0x10000800+i;
		destino[i]= 0x10002800+i;
	}
	GPDMA_ChannelCmd(0, ENABLE);
	NVIC_EnableIRQ(DMA_IRQn);
	while ((Channel0_TC == 0) && (Channel0_Err == 0)){
		//TERMINO DE TRANSFERIR
```

```
	}
	while(1);
	return 0;
}

void confDMA(void){
	GPDMA_Channel_CFG_Type GPDMACfg;
	NVIC_DisableIRQ(DMA_IRQn);

	GPDMA_Init(); //1 encienda el dma
	GPDMACfg.ChannelNum = 0; //2 elegir el canar
	GPDMACfg.SrcMemAddr = (uint32_t)origen; //3 fuente
	GPDMACfg.DstMemAddr = (uint32_t)destino; //4 destino

	GPDMACfg.TransferSize = 514; //7 establecer el tamaño de rafaga
	GPDMACfg.TransferWidth = GPDMA_WIDTH_HALFWORD;
	GPDMACfg.TransferType = GPDMA_TRANSFERTYPE_M2P; //8 memoria memoria
	GPDMACfg.SrcConn = GPDMA_CONN_MAT1_0;//5 quien va a activar la transmision por
DMA
	GPDMACfg.DstConn = 0;
	// Linker List Item - unused
	GPDMACfg.DMALLI = 0;
	// Setup channel with given parameter
	GPDMA_Setup(&GPDMACfg);
	}
void DMA_IRQHandler (void)
{
	if (GPDMA_IntGetStatus(GPDMA_STAT_INT, 0)){ //check interrupt status on channel 0
		if(GPDMA_IntGetStatus(GPDMA_STAT_INTTC, 0)){
```

```
                GPDMA_ClearIntPending (GPDMA_STATCLR_INTTC, 0);
                Channel0_TC++;
                }
                if (GPDMA_IntGetStatus(GPDMA_STAT_INTERR, 0)){
                GPDMA_ClearIntPending (GPDMA_STATCLR_INTERR, 0);
                Channel0_Err++;
                }
        }
return;
}
[14:01, 22/10/2019] Angeloff: const uint32_t origen [2049]= {0x10000800};
const uint32_t destino [2049]={0x10002800};
volatile uint32_t Channel0_TC;
volatile uint32_t Channel0_Err;

uint32_t *ptr = origen[0];

int main(void)
{
        uint8_t i=0;
        confDMA();
        Channel0_TC = 0;
        Channel0_Err = 0;
        GPDMA_ChannelCmd(0, ENABLE);
        NVIC_EnableIRQ(DMA_IRQn);
        while ((Channel0_TC == 0) && (Channel0_Err == 0)){
                //TERMINO DE TRANSFERIR
        }
        while(1);
```

```
	return 0;
}

void confDMA(void){
	GPDMA_Channel_CFG_Type GPDMACfg;
	NVIC_DisableIRQ(DMA_IRQn);

	GPDMA_Init(); //1 encienda el dma
	GPDMACfg.ChannelNum = 0; //2 elegir el canar
	GPDMACfg.SrcMemAddr = (uint32_t)origen; //3 fuente
	GPDMACfg.DstMemAddr = (uint32_t)destino; //4 destino

	GPDMACfg.TransferSize = 2049; //7 establecer el tamaño de rafaga
	GPDMACf…
[14:01, 22/10/2019] Angeloff: if (GPDMAChannelConfig->SrcConn > 15)
	{
		LPC_SC->DMAREQSEL |= (1<<(GPDMAChannelConfig->SrcConn - 16));
	} else {
		LPC_SC->DMAREQSEL &= ~(1<<(GPDMAChannelConfig->SrcConn - 8));
	}
[14:01, 22/10/2019] Angeloff: void configDMA(uint32_t *origen,uint32_t *destino,uint32_t
tamano_de_palabras_de32b ){
	GPDMA_Init();
	GPDMA_Channel_CFG_Type channel_cfg;
	channel_cfg.ChannelNum = 0;
	channel_cfg.TransferType = GPDMA_TRANSFERTYPE_P2M;
	channel_cfg.TransferSize = tamano_de_palabras_de16b;
	channel_cfg.SrcMemAddr = origen;
	channel_cfg.DstMemAddr = destino;
```

```
channel_cfg.TransferWidth = GPDMA_WIDTH_HALFWORD;
channel_cfg.SrcConn = GPDMA_CONN_MAT1_0;
channel_cfg.DstConn = 0;
channel_cfg.DMALLI = 0;
GPDMA_Setup(&channel_cfg);
LPC_GPDMACH0->DMACCControl |= GPDMA_DMACCxControl_SI;
LPC_GPDMACH0->DMACCSrcAddr = origen;
return;
```