Perifèrics
Practica 2

Angel Farre
Enric Gutiérrez

INDEX

1. Descripció de l'arquitectura del programa

Inicialment les configuracions dels diferents mòduls emprats a la practica son cridats en el main del programa. Aquests mòduls son el **timer2**, el **DAC**, la **DMA** i el **ADC**, els quals explicarem amb mes detall en els punts Diagrama de mòduls i Descripció dels mòduls del controlador.

El timer 2 l'hem emprat com a comptador free running per així poder comptar temps dins de la interrupció del timer2 i ser capaços de determinar el nombre de pulsacions fetes per l'usuari mitjançant el polsador. Aquest paràmetre ens modifica una variable que correspon a la velocitat a la que treballarem mes endavant amb el ADC i el DAC.

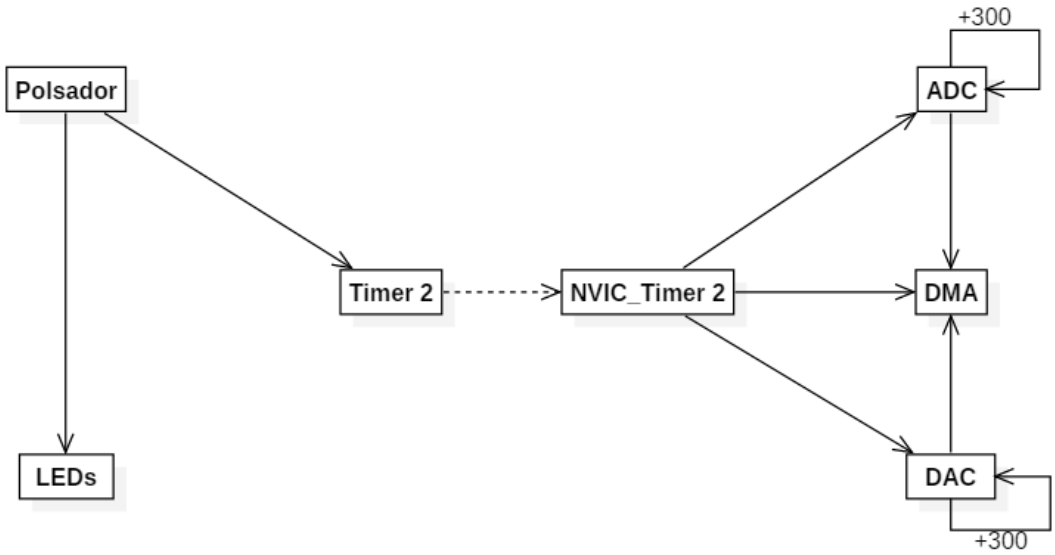
Quan l'usuari ha configurat aquest paràmetre (si ho fa ja que no es necessari, sent el valor per defecte de 10us) i prem el polsador mes de dos segons i mig, iniciem la conversió del ADC.

Aquesta crida es troba dins de la interrupció del timer 2 també. Al tenir el ADC configurat de forma que nomes faci una sola conversió, es fan les 300 conversions necessàries de les mostres dins d'un bucle.

Al finalitzar la conversió, els valors adquirits es mouen de la regió de memòria actual a una altra fent us de la DMA. Aquesta acció es realitza mitjançant la DMA ja que es aproximadament unes 3 vegades mes rapides que fent-ho mitjançant la CPU. Tant la regió de memòria inicial com la nova regió de memòria a la que desplacem les dades es arbitraria, sent la placa mateixa la que decideix la ubicació exacta.

2. Diagrama de mòduls

En la Il·lustració 1 - Diagrama general dels mòduls emprats en la pràctica següent, es pot observar les relacions existents entre els diferents mòduls. Addicionalment, es mostra a continuació de la imatge quina es la configuració de cada mòdul per a esclarir el seu funcionament:



Il·lustració 1 - Diagrama general dels mòduls emprats en la pràctica

- Timer 2: Free running timer per controlar el temps que l'usuari prem el boto i modificat a mitja execució per a generar interrupcions i generar peticions de ADC.
    - Configuració:

```
TIM_TimeBaseInitTypeDef SetupTimer;
/* Enable timer 2, using the Reset and Clock Control register */
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
SetupTimer.TIM_Prescaler = 0x0000;
SetupTimer.TIM_CounterMode = TIM_CounterMode_Up;
SetupTimer.TIM_Period = 0xFFFFFFFF;
SetupTimer.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseInit(TIM2, &SetupTimer);
TIM_Cmd(TIM2, ENABLE); /* start counting by enabling CEN in CR1 */
```

- ADC: Conversio de dades de Analògic a Digital.
    - Configuració:

```
/* ADC Common Init
**********************************************************/
  ADC_CommonInitStructure.ADC_Mode = ADC_Mode_Independent;
  ADC_CommonInitStructure.ADC_Prescaler = ADC_Prescaler_Div2;
  ADC_CommonInitStructure.ADC_DMAAccessMode = ADC_DMAAccessMode_Disabled;
  ADC_CommonInitStructure.ADC_TwoSamplingDelay =
ADC_TwoSamplingDelay_5Cycles;
  ADC_CommonInit(&ADC_CommonInitStructure);

  /* ADC3 Init
****************************************************************/
  ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b;
  ADC_InitStructure.ADC_ScanConvMode = DISABLE;
```

```
  ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
  ADC_InitStructure.ADC_ExternalTrigConvEdge =
ADC_ExternalTrigConvEdge_None;
  ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_T1_CC1;
  ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStructure.ADC_NbrOfConversion = 1;
  ADC_Init(ADC3, &ADC_InitStructure);
```

- DAC: Conversio de dades de Digital a Analògic
    - Configuració:

```
  /* Set DAC options */
  DAC_InitStruct.DAC_Trigger = DAC_Trigger_None;
  DAC_InitStruct.DAC_WaveGeneration = DAC_WaveGeneration_None;
  DAC_InitStruct.DAC_OutputBuffer = DAC_OutputBuffer_Enable;
```

- DMA: Direct Memory Access, utilitzat per escriure i llegir de la SDRAM.
    - Configuració DMA per ADC:

```
  /* DMA2 Stream0 channel2 configuration */
  DMA_InitStructure.DMA_Channel = DMA_Channel_2;
  DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)ADC3_DR_ADDRESS;
  DMA_InitStructure.DMA_Memory0BaseAddr = (uint32_t)&ADC3ConvertedValue;
  DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralToMemory;
  DMA_InitStructure.DMA_BufferSize = 1;
  DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
  DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Disable;
  DMA_InitStructure.DMA_PeripheralDataSize =
DMA_PeripheralDataSize_HalfWord;
  DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
  DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
  DMA_InitStructure.DMA_Priority = DMA_Priority_Medium;
  DMA_InitStructure.DMA_FIFOMode = DMA_FIFOMode_Disable;
  DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_HalfFull;
  DMA_InitStructure.DMA_MemoryBurst = DMA_MemoryBurst_Single;
  DMA_InitStructure.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;
  DMA_Init(DMA2_Stream0, &DMA_InitStructure);
  DMA_Cmd(DMA2_Stream0, ENABLE);
```

    - Configuració DMA per Memory2Memory:

```
DMA_InitStructure.DMA_Channel = DMA_Channel_2;
   DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&totalMostres;
   DMA_InitStructure.DMA_Memory0BaseAddr = (uint32_t)&destination;
   DMA_InitStructure.DMA_DIR = DMA_DIR_MemoryToMemory;
   DMA_InitStructure.DMA_BufferSize = ARRAYSIZE;
   DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Enable;
   DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
   DMA_InitStructure.DMA_PeripheralDataSize =
DMA_PeripheralDataSize_HalfWord;
   DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
   DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
   DMA_InitStructure.DMA_Priority = DMA_Priority_Medium;
   DMA_InitStructure.DMA_FIFOMode = DMA_FIFOMode_Enable;
   DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_Full;
   DMA_InitStructure.DMA_MemoryBurst = DMA_MemoryBurst_INC8;
   DMA_InitStructure.DMA_PeripheralBurst = DMA_PeripheralBurst_INC8;
```

```
DMA_Init(DMA2_Stream3, &DMA_InitStructure);
```

3. Descripció dels mòduls del controlador

Tal i com hem mostrat en la figura **¡Error! No se encuentra el origen de la referencia.**, el funcionament de la practica es el següent:

- Timer 2:
  Timer configurat com a free running timer sense interrupció. En el main, dins del bucle infinit, es consulta si el polsador esta premut consultant el pin adient. Si esta premut, s'agafa el temps de referencia actual del timer 2 i ens enbuclem de nou fins que el usuari deixa de prémer el boto.

  Al deixar de prémer el boto, agafem de nou la referencia actual del timer 2 i es fa la diferencia amb el moment en que l'usuari ha premut el boto inicialment.

  Amb això podem saber el temps que l'usuari ha tingut premut el boto i actuar de forma corresponent.

  El timer 2 es reconfigura al prémer el boto durant mes de dos segons i mig per a que generi interrupcions. D'aquesta forma a cada interrupció generada pel timer, es fa una crida al ADC per a que realitzi una conversió. La interrupció assegura que el ADC tingui temps suficient per a realitzar la conversió.

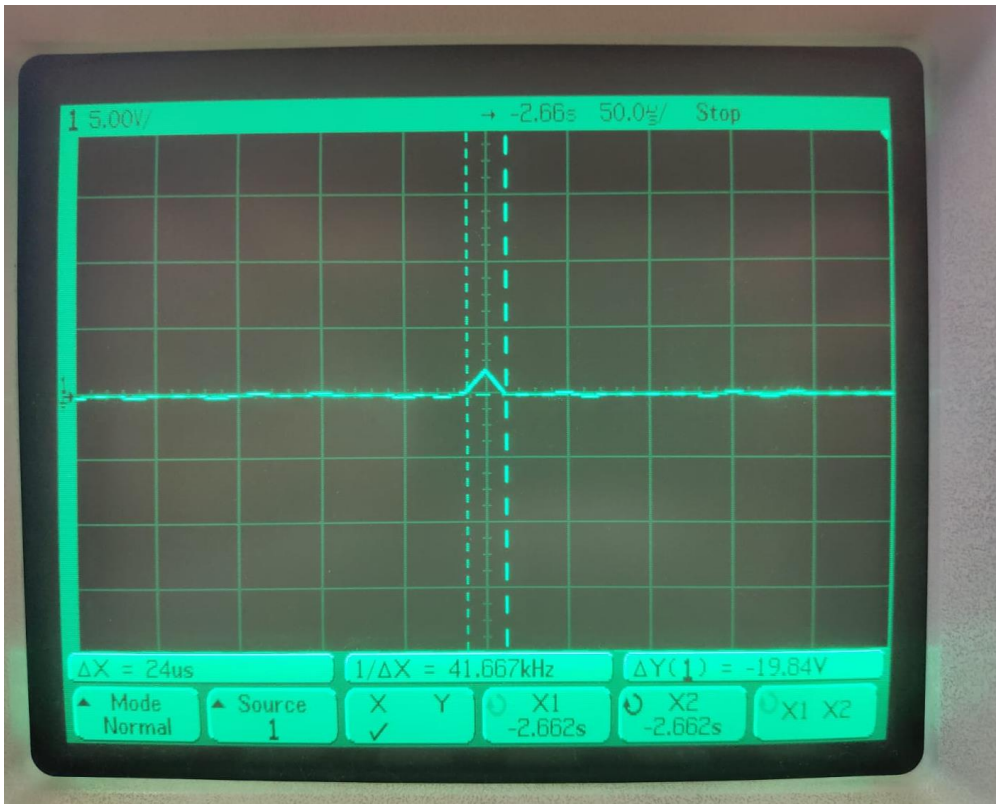
- ADC:
  El ADC (Analog to Digital Converter) esta configurat de forma que realitzi una sola conversió. Com a bona praxis, les crides al ADC estan dins de la interrupció del timer així garantim una conversió espaïda i sense solapaments.
  Al estar el pin d'entrada del ADC al aire, això genera uns valors de conversió compresos entre 800 i 1000.

- DMA:
  La DMA (Direct Memory Access) es configura de forma anàloga en tots els casos. L'única diferencia que hi ha es en el camp `DMA_InitStructure.DMA_DIR`, per indicar la direcció de la conversió (MemoryToMemory o PeripherailToMemory) i en el camp `DMA_InitStructure.DMA_BufferSize` el qual indica la mida de dades a convertir. Es podria haver mantingut aquest segon camp constant a un valor de 300, però la conversió del ADC s'ha realitzat de dada a dada i s'ha mantingut aquest format.

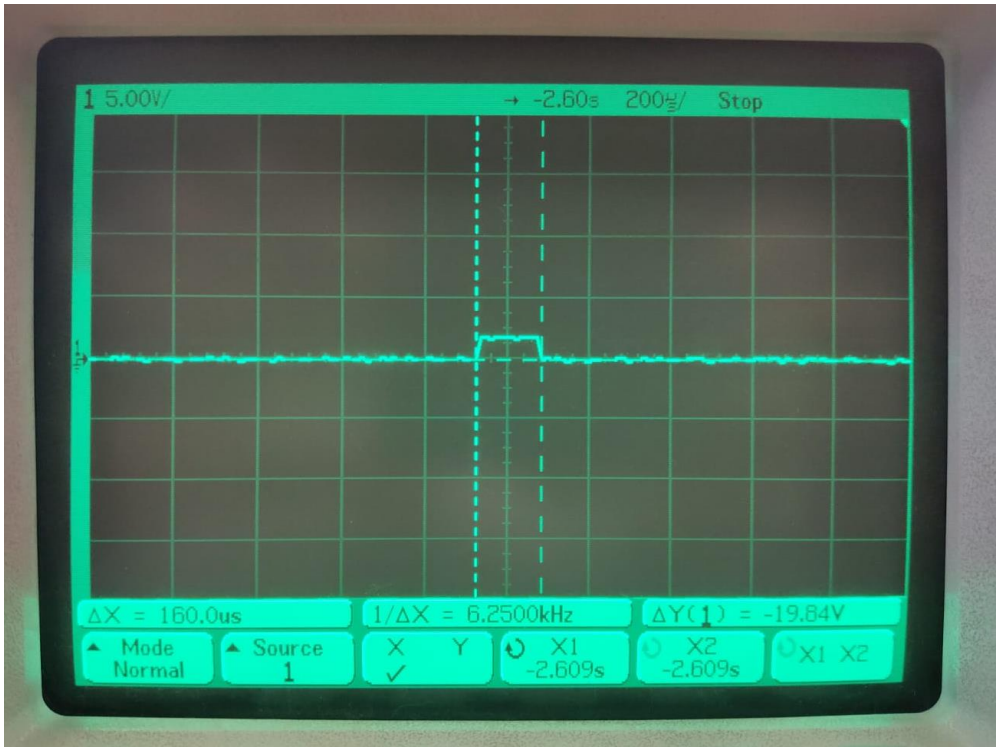
- DAC:
  El DAC (Digital to Analog Converter) s'empra en la fase final per comprovar que les dades obtingudes amb el ADC i tractades amb la DMA son correctes. Per a fer-ho, es fa us d'un sol canal del DAC, el canal1. De forma anàloga al emmagatzematge de dades per part del ADC, el DAC llegeix les dades una a una i les mostra de forma cíclica. S'ha decidit mostrar-les de forma cíclica ja que les dades es recopilen, i per tant mostren, en un espai molt petit de temps i de no ser el seu mostratge cíclic, no s'apreciaria.

4. Resultat de les qüestions plantejades

- Durada de la transferència MemoryToMemory. Tal i com mostra la il·lustració següent, es aproximadament d'uns 24us.

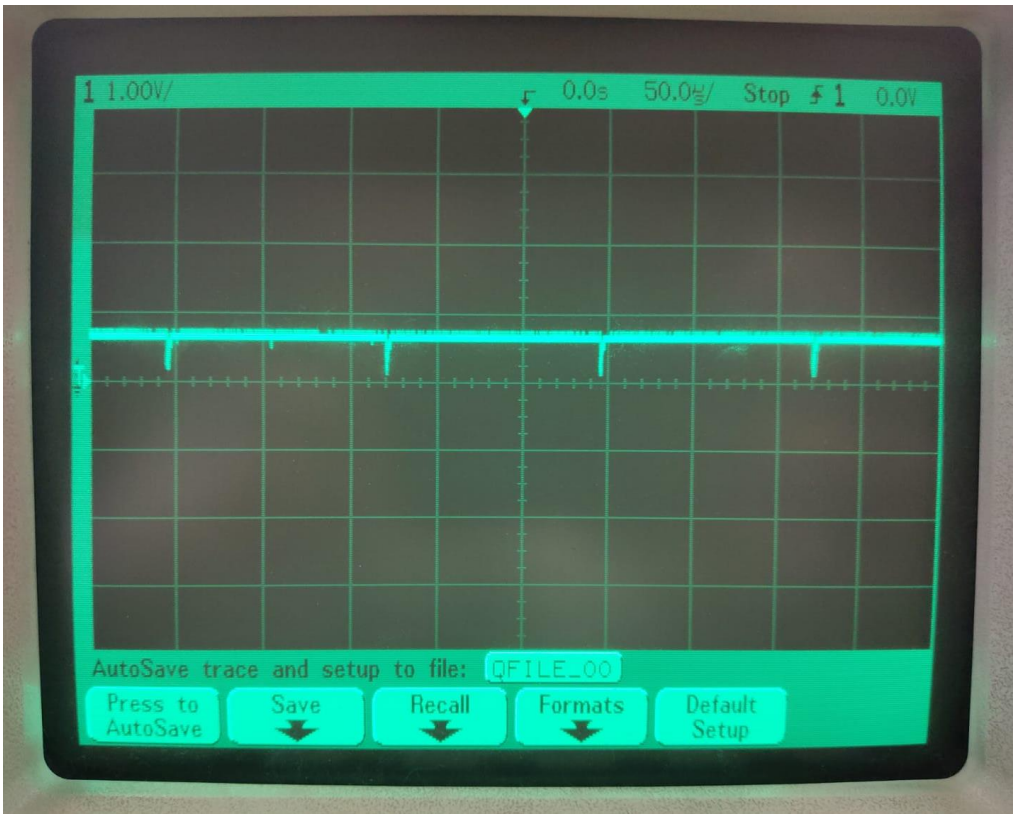


- Durada de la transferència MemoryToMemory mitjançant l'us de la CPU. Tal i com mostra la il·lustració següent, es aproximadament d'uns 160us, 6.6 vegades superiors al cas anterior.



5. Captures de pantalla i formes d'ona

A continuació es mostra una captura del output del DAC si hem fet servir la placa. Es pot observar que tal i com hem mencionat anteriorment, els resultats es mostren de forma cíclica per a que es pugin apreciar.



6. Conclusions

En aquesta practica hem treballat amb el DAC, un mòdul que sempre s'ha explicat teòricament i mai havíem tingut que aplicar fins avui. La conversió del ADC no es nova, però si que ho es la forma particular en la que s'ha implementat en aquesta practica, ja que no s'emmagatzema de forma unitària en un array, si no que es realitza dada a dada.

S'ha de mencionar també la significant diferencia entre moure dades entre espais de memòria fent servir la CPU o la DMA (accedint directament a memòria). Com hem demostrat, el accés directe a memòria mitjançant la DMA es significativament rapida i en sistemes que tinguin requisits de resposta elevats, pot ser força interessant aplicar el que s'ha après aquí.

Per últim, el disseny de les llibreries i la versatilitat de la placa facilita bastant treballar amb ella de forma interna sense tenir que desenvolupar un codi llarg i complex.