UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Programación GPIO (General Purpose Input/Output) en los microcontroladores ARM Cortex-M: Salida digital

Apellidos, nombre	Capella Hernández, Juan Vicente (jcapella@disca.upv.es)
Departamento	Informática de Sistemas y Computadores (DISCA)
Centro	Universitat Politècnica de València

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

1 Resumen de las ideas clave

En este artículo se introduce al lector en la programación de la GPIO (General Purpose Input/Output) de los microcontroladores ARM Cortex-M, utilizando bibliotecas basadas en el estándar CMSIS (Cortex Microcontroller Software Interface Standard), para realizar operaciones de salida digital. Se mostrarán y explicarán a lo largo del artículo ejemplos en lenguaje C utilizándose el entorno de desarrollo Keil uVision. Todo ello de forma guiada, paso a paso, para que pueda seguirlo y aplicarlo a la vez.

2 Introducción

La salida digital es un mecanismo básico de cualquier microcontrolador que permite generar señales en dos posibles estados, que serán equivalentes a un "1" o a un "0" en la parte software.

De esta manera, conociendo el mecanismo de salida digital, cómo configurarlo y programarlo en los microcontroladores ARM Cortex-M [1, 2] de St, podremos actuar sobre los dispositivos que conectemos al microcontrolador, y por ejemplo encender/ apagar leds, abrir/cerrar válvulas, activar/desactivar motores, y un largo etcétera.

Además todo ello lo haremos recurriendo a bibliotecas basadas en CMSIS [3] que proporciona el fabricante, en este caso St. Estas bibliotecas lo que hacen es crear una abstracción de los periféricos, en este caso de entrada/salida (GPIO) que facilita su utilización por parte del programador.

Indicar finalmente que las señales digitales de un microcontrolador cualquiera se organizan en puertos, que son agrupaciones de 8, 16 o 32 líneas en función del modelo de microcontrolador. En la familia que nos ocupa (STM32), las agrupaciones son de 16 líneas que se corresponden internamente a una palabra binaria de 16 bits, denotándose los puertos mediante letras (A,B,C..), así por ejemplo para referirnos al pin 15 del puerto A utilizaremos: PA15.

3 Objetivos

Una vez que el alumno se lea con detenimiento este documento, será capaz de:

- Aplicar el concepto de salida digital.
- Configurar los pines necesarios de un microcontrolador ARM Cortex-M de St como salida digital.
- Desarrollar programas en C que gestionen adecuadamente salidas digitales.

4 Desarrollo

A continuación se desarrollarán cada uno de los aspectos indicados en la introducción y objetivos, realizando las explicaciones de la forma más práctica y guiada posible.

4.1 Preparación (configuración inicial)

Consultar el esquemático de la placa Discovery para tener clara la electrónica de los 4 LEDs de que dispone y saber en qué líneas están conectados y cómo se atacan. Estos aspectos habrá que tenerlos claros, para en los apartados posteriores de la práctica, poder configurar los pines correspondientes de forma adecuada. El esquemático y demás información está disponible en el manual de la placa, pudiéndose encontrar también en la figura 1 la sección del esquemático respecto a los leds de la placa de evaluación DISCOVERY.

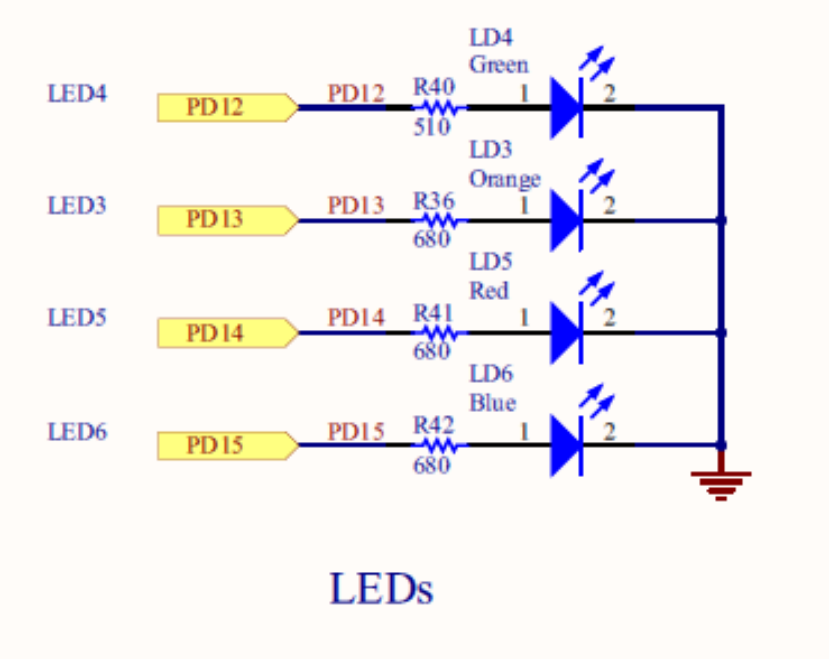

Imagen 1. Esquemático leds placa evaluación DISCOVERY

4.2 Salida digital: configuración

Para poder escribir una salida digital, que nos permita p.e manejar un LED, necesitamos en primer lugar activar y configurar adecuadamente el puerto GPIO donde se encuentre el pin/es que vayamos a utilizar. Para ello deberemos llamar a la función: `void RCC_AHB1PeriphClockCmd (uint32_t RCC_AHB1Periph, FunctionalState NewState)`, indicando en el primer parámetro el puerto que queremos activar y como segundo parámetro ENABLE. Con ello daremos reloj al periférico.

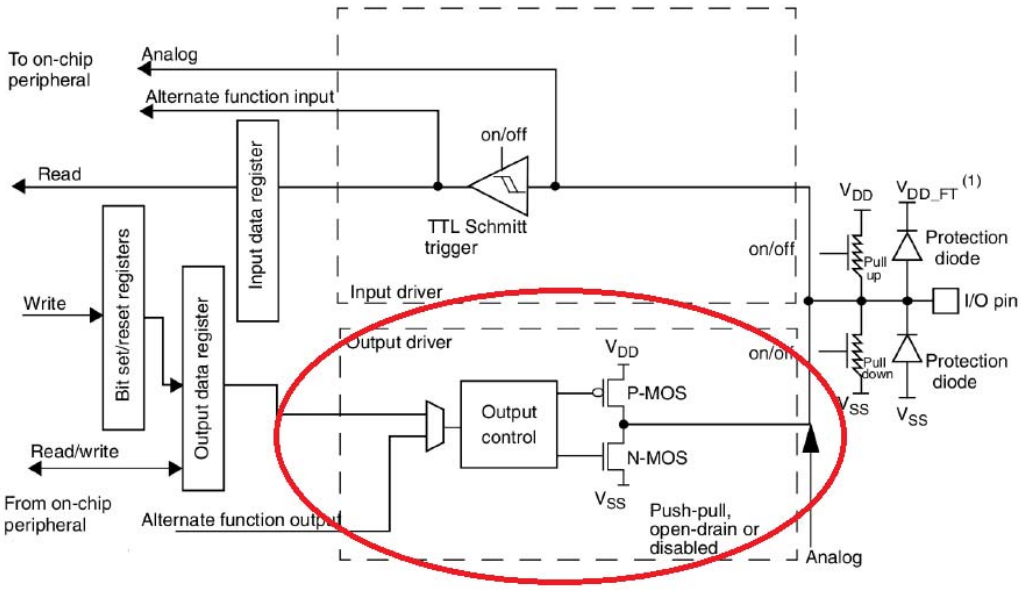

Imagen 2. Estructura interna pines microcontroladores STM32F

A continuación tendremos que indicar la configuración que queremos darle a los pines de dicho puerto rellenando los campos correspondientes de la estructura `GPIO_InitTypeDef GPIO_InitStructure`, en la que indicaremos qué pines queremos configurar y cómo queremos configurarlos. Para rellenar adecuadamente dicha estructura tenemos que tener en mente la estructura interna de los pines del microcontrolador, para el caso que nos ocupa STM32F la podemos ver en la figura 2. Básicamente el driver de salida lo podemos configurar en modo "Push-Pull" ó "OpenDrain", pudiéndonos adaptar a gran variedad de aplicaciones, además es posible también configurar las resistencias de pull-up y pulldown incluidas en la celda de cada pin.

Finalmente invocaremos la función `void GPIO_Init (GPIO_TypeDef *GPIOx, GPIO_InitTypeDef *GPIO_InitStruct)`, donde indicaremos como primer parámetro el puerto GPIO que estamos configurando (p.e GPIOD) y como segundo la estructura que hemos rellenado en el paso anterior. Las distintas maneras de gestionar la célula de salida digital se describen en el manual de referencia, en nuestro caso, para la familia STM32F4x podemos consultar [4]. En el apartado 4.4 podemos ver un ejemplo completo y comentado, donde se configura un pin como salida digital.

4.3 Salida digital: manejo

Ahora ya podríamos utilizar las diferentes funciones de las bibliotecas CMSIS disponibles [5, 6] para poder escribir en el pin/es que hayamos configurado como salida digital, siendo de destacar las siguientes:

- `void GPIO_SetBits (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)`

Pone a "1" los bits indicados.

- `void GPIO_ResetBits (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)`

  Pone a "0" los bits indicados.

- `void GPIO_WriteBit (GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, BitAction BitVal)`

  Escribe al valor indicado el bit seleccionado.

- `void GPIO_Write (GPIO_TypeDef *GPIOx, uint16_t PortVal)`

  Escribe el dato (uint16_t) en el Puerto indicado.

4.4 Un ejemplo completo

A continuación se muestra un ejemplo comentado donde podrás ver cómo se realiza la configuración de un pin como salida digital, así como se puede gestionar dicha línea una vez configurada, aplicándose las funciones explicadas en los apartados anteriores, y a partir del cual podrás ir probando otras configuraciones y combinaciones.

```
#include <stdint.h>

#include "stm32f4xx_rcc.h"
#include "stm32f4xx_gpio.h"



//Configuración del pin donde está conectado el LED azul (PD15):
void led_inicializar(void)
{
   GPIO_InitTypeDef  GPIO_InitStructure;  //estructura para especificar la config.

   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE); // damos reloj al periférico

   GPIO_StructInit(&GPIO_InitStructure);   // establece los valores por defecto

   /* dar los detalles del puerto/pin */
   GPIO_InitStructure.GPIO_Pin   = GPIO_Pin_15;       // pin que vamos a configurar
   GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_OUT;     // se desea configurar como salida
   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;      // modo push-pull
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_25MHz;   // frec. máxima para el puerto
   GPIO_InitStructure.GPIO_PuPd  = GPIO_PuPd_NOPULL;  // desactivar resistencias pullup

   GPIO_Init(GPIOD, &GPIO_InitStructure);           // hacer efectiva la configuración
}
```

```
//Para encender el LED:
void led_on(void)
{
    GPIO_SetBits(GPIOD, GPIO_Pin_15);     // poner a "1" el pin PD15
}

//Para apagar el LED:
void led_off(void)
{
    GPIO_ResetBits(GPIOD, GPIO_Pin_15);     // poner a "0" el pin PD15
}

//Para hacer retardos
void retardo(uint32_t nCount)
{
  while(nCount--){}
}

// Programa principal:
int main (void)
{
   led_inicializar();

   while (1) {
        led_on();
        retardo(1000000);
        led_off();
        retardo(1000000);
   }

}
```

5 Cierre

A lo largo de este objeto de aprendizaje hemos tratado las cuestiones básicas relacionadas con la configuración y gestión de salida digital en microcontroladores de la familia ARM Cortex-M.

Para comprobar que realmente has aprendido las bases de la programación de la salida digital en microcontroladores ARM Cortex-M de St es el momento de que te pongas manos a la obra e intentes crear un proyecto donde configures uno o varios pines de un puerto (GPIOA, GPIOB,..) como se ha explicado a lo largo del artículo y desarrolles un programa que vaya cambiando el estado de dicho/s pin/es cada cierto tiempo. Si a dicho/s pin/es conectas por ejemplo un led podrás

verificar el correcto funcionamiento del sistema (también podrías utilizar un osciloscopio).

¡¡ÁNIMO!!

6 Bibliografía

[1] *ARM Limited. Cortex-M4 technical reference manual, 2010.* Disponible en URL: http://www.arm.com/

[2] Jonathan W. Valvano: *Introduction to the Arm Cortex-M microcontrollers*. Fifth edition, 2015.

[3] *CMSIS-CORE support for ARM Cortex-M processor-based devices*. Disponible en URL: http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php

[4] *Microelectronics, St. RM0090 Reference manual for STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx and STM32F43xxx*, 2014. Disponible en URL: http://www.st.com/st-webui/static/active/en/resource/technical/document/reference_manual/DM00031020.pdf

[5] *StMicroelectronics. St STM32F4 DSP and standard peripherals library*, 2014.

[6] *Microelectronics, St. STM32F3xxx and STM32F4xxx Cortex-M4 programming Manual*, 2014. Disponible en URL: http: //www. st.com/web/en/resource/technical/document/programming_manual/DM0004698.pdf