|  |
| --- |
|  |
| DESARROLLO TP-UAR DALI |
|  |

|  |
| --- |
| Javier Riego Barcia  1-3-2021 |



# Introducción

Desarrollo de una TP-UART para equipos DALI. Programa para la recepción, lectura y envío de datos mediante protocolo DALI.

Sistema capad de leer y de enviar señales cumpliendo con la normativa DALI establecida. La normativa DALI se compone de tres capas principales:

**CAPA 1xx: REQUERIMIENTOS GENERALES**

* CAPA 101: Componentes del Sistema (General)
* CAPA 102: Balastros
* CAPA 103: Dispositivos de control

**CAPA 2xx y 3xx: REQUERIMIENTOS PARTICULARES**

En base a esta normativa el sistema ha de cumplir con unos ciertos requerimientos para considerarse un equipo DALI en correcto funcionamiento.

Dicho sistema ha sido desarrollado para el microcontrolador “STM32G070”, del fabricante *STMicroelectronics*. Para sus pruebas se ha utilizado la tarjeta de desarrollo “Núcleo-G070RB”.

## Conexiones al microcontrolador y módulos activos

|  |  |  |
| --- | --- | --- |
| PIN / MÓDULO | CONFIGURACIÓN | FUNCIÓN |
| PB4 | OUTPUT | Salida de datos al bus DALI (Tx) |
| PB5 | INPUT RISING/FALLING EDGE INTERRUPTS | Entrada de datos del bus DALI (Rx) |
| TIMER 3 | AUTORRELOAD = ENABLED | Detección de Send-Twice frames y fallos del sistema tras un Power Down |
| TIMER 6 | AUTORELOAD = ENABLED | Envío de tramas |
| TIMER 14 | AUTORELOAD = ENABLED  PRESCALER = 15  COUNTERMODE = UP | Recepción de tramas |

Tabla .: Conexiones y módulos del microcontrolador utilizados

# Recepción de señales

Las señales en un bus DALI son enviadas por el BUS como trenes de pulsos con codificación Manchester. Estas señales varían entre los dos estados lógicos LOW (0V) [ACTIVE STATE] y HIGH (16V) [IDLE STATE]. Mediante un transceptor se adaptan dichos niveles de tensión al rango de 0 a 3.3V, operables desde un microcontrolador.

Las tramas o “*frames*” recibidos serás de 16 o 24 bits, y han de cumplir con las especificaciones temporales mostradas en la Figura 1, Figura 2 y Figura 3:

Tabla

Descripción generada automáticamente

Figura : Tiempos de recepción con flanco en la mitad del tiempo de bit

Tabla

Descripción generada automáticamente

Figura : Tiempos de recepción con flanco en la mitad del tiempo de bit

Tabla

Descripción generada automáticamente

Figura : Tiempos de establecimiento entre diferentes tipos de trama

Cada frame en DALI tiene la siguiente estructura:

* 1 BIT START: Al estar el bus en IDLE STATE [“1”], comienza con un flanco de bajada y posteriormente la codificación Manchester del estado lógico “1” [Flanco ascendente], vuelta a IDLE STATE.
* 16 o 24 BITS DE DATOS
* 2 BITS STOP: Retorno a IDLE STATE [“1”]

## Detección y lectura de tramas recibidas

Para la lectura y posterior decodificación de la señal recibida se ha llevado a cabo la siguiente metodología:

Mediante interrupciones por cambio de nivel en PB5, se detecta cada uno de los flancos de la señal de entrada (subida y bajada). Con cada flanco se almacena en un array la información más relevante relativa a este, tiempo que ha tardado en darse y el valor tras dicho flanco.

Mediante la utilización del “Timer 14” del microcontrolador, se da inicio a una temporización con fada flanco recibido, de esta forma podremos monitorizar errores temporales en la señal de entrada, así como detectar Power Down o determinar cuando se ha terminado de recibir un frame.

El proceso de detección de flancos es el mostrado en el siguiente diagrama de flujo:

Diagrama

Descripción generada automáticamente

Figura : Diagrama de flujo de la detección de flancos

Con cada flanco descendente, se comprueba si es el primer flanco de la trama, ya que todas las tramas DALI comienzan con este tipo de flanco. Si es el primero se inicializan las variables y condiciones pertinentes para comenzar a recibir el frame de forma correcta.

Tras haber recibido el flanco, determinado de qué tipo de flanco se trata y si es el primero o no de la trama, se realiza la llamada al controlador o manejador de flancos, mediante el cual, se determina y almacena la posición del flanco dentro del tiempo de bit (al inicio o en medio), y el tiempo transcurrido entre cada flanco. Esto es algo muy importante, ya que el flanco que nos proporciona información relevante acerca del dato que se pretende leer es el flanco central (Manchester), que determina el valor del bit recibido; si el flanco es ascendente, el dato recibido es un “1”, si el flanco es descendente, el dato recibido es un “0”.

Gráfico

Descripción generada automáticamente con confianza media

Figura : Codificación Manchester

El diagrama de flujo mostrado en la Figura 6, muestra el proceso de tratamiento y almacenamiento de cada mitad de bit recibido (todos los tiempos están en microsegundos), para su posterior procesado y decodificación.

La configuración del TIMER 14 para el conteo del tiempo de bit y la detección de POWER DOWN y STOP CONDITION tras la recepción de un frame ha sido la siguiente:

* CounterMode = TIM\_COUNTERMODE\_UP
* Period = 65535
* Prescaler = 15
* ClockDivision = TIM\_CLOCKDIVISION\_DIV1
* AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_ENABLE

Se ha configurado un prescaler de 15 en función a la siguiente fórmula (disponible en el datasheet) para reducir la frecuencia del reloj de 16 MHz a 1MHz, y realizar así el conteo en microsegundos [1 TICK = 1us].

Diagrama

Descripción generada automáticamente

Figura : Diagrama de flujo del manejo de flancos "EdgeHandler()"

## Decodificación Manchester de la trama recibida

Una vez almacenada la información de cada medio bit en un array de datos, se analiza la tabla en busca de flancos ascendentes (cambio de “0” a “1” lógico), en codificación Manchester un flanco ascendente supone un “1”. Encontrado un “1”, mediante una serie de operaciones lógicas de rotación y adición, se colocará en el lugar que corresponda de la trama. La información extraída es leída de 8 en 8 bits (1 byte) y almacenada en un array (rx\_buffer) de 3 bytes, en el que cada byte contendrá:

* rx\_buffer[0] = ADDRESS
* rx\_buffer[1] = DATA1
* rx\_buffer[2] = DATA2

Diagrama

Descripción generada automáticamente

Figura : Diagrama de flujo procesamiento del frame recibido “processRxFrame()”