D-BUGERZ SCHOOL

ARM INTERFACING BOOTCAMP #02

Final Project

Fady Emad

Arm Bootcamp

9/20/24

CONTENTS

1. CASE STUDY

1. SYSTEM OVERVIEW

This project involves the development of an automotive embedded system using two Electronic Control Units (ECUs) connected via Serial Peripheral Interface (SPI). The main tasks of the ECUs are divided as follows:

- ECU1: Controls the state machines and sends UART messages to a PC.
- ECU2: Executes the state machines based on the SPI message received from ECU1.

MAIN COMPONENTS:

- ECU1: Reads inputs (switches) and communicates with ECU2 via SPI. It also sends UART messages to a PC, which indicates the current system status and any errors.
- ECU2: Executes state machine logic based on commands received from ECU1 over SPI.

2. FUNCTIONAL REQUIREMENTS

2.1 ECU1:

Reads on-board switches (SW1 & SW2).

- Sends SPI messages every 50 ms to control ECU2.
- Transmits state change information to the PC via UART after every state transition.

2.2 ECU2:

- Processes SPI commands every 10 ms and send data to UART also.
- Executes state transitions based on received commands:
- -SW1: Triggers a clockwise state transition every 1000 ms.
  -SW2: Triggers an anticlockwise state transition every 1000 ms.
  -Both pressed: Resets both ECUs to the initial state (RED).
  -If no state change is detected for more than 10 seconds, ECU1 commands ECU2 to enter the IDLE state (WHITE).

2. ASSUMPTIONS

RTOS

- Assume scheduler of type priority based non preemptive

A. ECU 1

#INCLUDE

```c
/* UTILES_LIB */
#include "STD_TYPES.h"
#include "BIT_MATH.h"
#include "STM32F103x6.h"

/* MCAL */
#include "STM32_F103C6_GPIO_Driver.h"
#include "STM32_F103C6_SPI_Driver.h"
#include "STM32_F103C6_USART_Driver.h"
#include "STM32_F103C6_SYSTICK_DRIVER.h"
#include "Timer.h"
#include "PB_interface.h"

/* RTOS STACK */
#include "KERNEL_interface.h"

```

ENUMS

```c
typedef enum Buttom_status{
    Idle,
    NoPress,
    PB1_Press,
    PB2_Press,
    TwoPress,
    Error
}Buttom_status_t;

```

MACROS

```c
#define PB1             GPIO_PIN_1
#define PB2             GPIO_PIN_2
```

GLOBAL VARIABLES

```c
volatile u8 PB1_Flag = Idle;
volatile u8 PB2_Flag = Idle;
u8 ch= Idle;
u8 ch_temp= Idle;
u8 LastProcess;
u8 lastMessage = Idle;

```

PROTOTYPES

```c
/**================================================================
  * @Fn                         - main
  * @brief                      - main function inside tasks is
                                created and RTOS then starts
  * @param [in]                 - void
  * @retval                     - integer
  * Note                        - none
  */
int main(void);
/**================================================================
  * @Fn                         - init
  * @brief                      - Initialize {RCC for GPIOs used
                                ,Systick,SPI, UART & timer2  }
  * @param [in]          - void
  * @retval                     - void
  * Note                        - none
  */
void init(void);
/**================================================================
  * @Fn                         - Read_Button_State
  * @brief                      - Reads buttons and raise flags
                                according to pressed button
  * @param [in]          - void
  * @retval                     - void
  * Note                        - none
  */
```

```
void Read_Button_State(void);



/**=============================================================
  * @Fn                           - SPI_Send_Message
  * @brief                      - send spi message to other ECU with
                                the state of last buttons pressed
  * @param [in]                   - void
  * @retval                       - void
  * Note                          - none
  */
void SPI_Send_Message(void);

/**=============================================================
  * @Fn                           - Uart_Send_State
  * @brief                        - send UART message to PC with the
                                  last button pressed
  * @param [in]                   - void
  * @retval                       - void
  * Note                          - none
  */
void Uart_Send_State(void);

```

B. ECU2

#INCLUDES

```
/* UTILES_LIB */
#include "STD_TYPES.h"
#include "BIT_MATH.h"
u8 ch,ch1;
u32 count=0;

/* MCAL */
#include "STM32_F103C6_GPIO_Driver.h"
#include "STM32_F103C6_SPI_Driver.h"
#include "STM32_F103C6_USART_Driver.h"
#include "STM32_F103C6_SYSTICK_DRIVER.h"

/* RTOS STACK */
```

```c
#include "KERNEL_interface.h"

```

MACROS

```c
#define LED_Red          GPIO_PIN_0
#define LED_Green GPIO_PIN_1
#define LED_Blue  GPIO_PIN_2

```

ENUMS

```c
typedef enum{
   NOPress,
   CW,
   CCW,
   _ERROR
}ProcessState_t;

typedef enum Receive{
   Ideal,
   NoPress,
   PB1_Press,
   PB2_Press,
   TwoPress,
   Error
}Receive_status_t;

```

GLOBAL VARIABLES

```c
u8 ch,STATE;

```

PROTOTYPES

```c
/**=================================================================
 * @Fn                              - main
 * @brief                           - Calls Init function and creates
                                    tasks required then start scheduler
 * @param [in]                      - void
 * @retval                          - integer
 * Note                             - none
 */
int main(void);
/**================================================================
 * @Fn                             - Init
 * @brief                          - Initializes RCC , Leds, timer2, Systick ,SPI and UART
 * @param [in]                     - void
 * @retval                         - void
 * Note                            - none
 */
void Init(void);
/**================================================================
 * @Fn                             - LEDS_Proccess
 * @brief                          - This function proccess the current leds state
 * @param [in]                     - void
 * @retval                         - void
 * Note                            - none
 */
void LEDS_Proccess(void);
/**================================================================
 * @Fn                             - SPIReceive
 * @brief                          - Receive SPI message from Master and Change state of Leds according to it
 * @param [in]                     - void
 * @param [in]                     - void
 * @retval                         - none
 * Note                            -
 */
void SPIReceive(void);
```