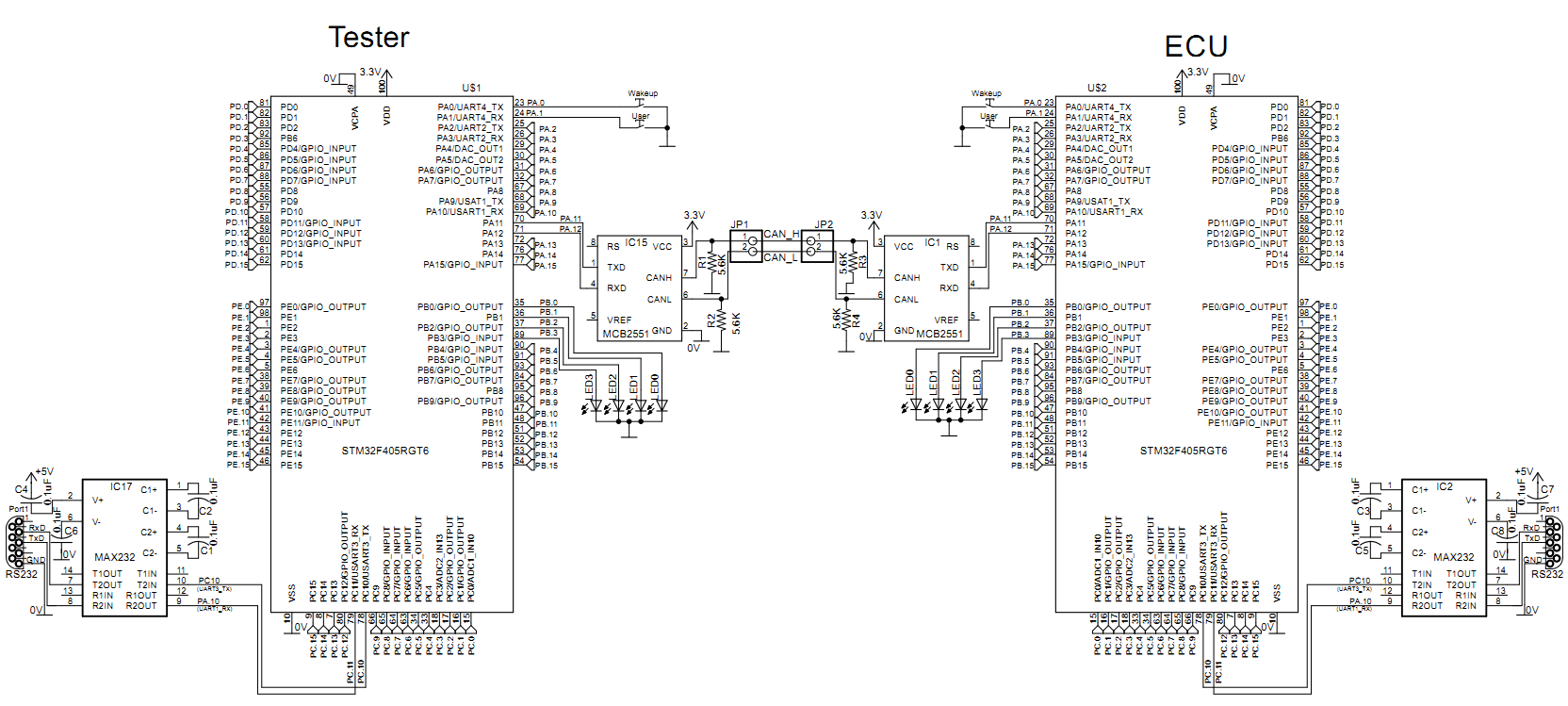
**I. Hardware**



**II. Software**

**A. GPIO\_ Configuration for Led (LED1->LED4)**

void GPIO\_Configuration(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOB, ENABLE);

//LED1 (PC9) LED2 (PC10) LED3 (PC11) LED4 (PC12)

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0 | GPIO\_Pin\_1| GPIO\_Pin\_2| GPIO\_Pin\_3;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_OUT;

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_100MHz;

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

}

**B. UART3\_ Configuration for monitoring**

void USART\_Configuration(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

USART\_InitTypeDef USART\_InitStructure;

RCC\_AHB1PeriphClockCmd(Open\_USARTx\_TX\_GPIO\_CLK,ENABLE);

RCC\_AHB1PeriphClockCmd(Open\_USARTx\_RX\_GPIO\_CLK,ENABLE);

#if defined OpenUSART1 || defined OpenUSART6

RCC\_APB2PeriphClockCmd(Open\_USARTx\_CLK,ENABLE);

#else

RCC\_APB1PeriphClockCmd(Open\_USARTx\_CLK,ENABLE);

#endif

GPIO\_PinAFConfig(Open\_USARTx\_TX\_GPIO\_PORT, Open\_USARTx\_TX\_SOURCE, Open\_USARTx\_TX\_AF);

GPIO\_PinAFConfig(Open\_USARTx\_RX\_GPIO\_PORT, Open\_USARTx\_RX\_SOURCE, Open\_USARTx\_RX\_AF);

/\*

\* Open\_USARTx\_TX -> PA9 , Open\_USARTx\_RX -PA10

\*/

GPIO\_InitStructure.GPIO\_Pin = Open\_USARTx\_TX\_PIN;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF;

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;

GPIO\_Init(Open\_USARTx\_TX\_GPIO\_PORT, &GPIO\_InitStructure);

GPIO\_InitStructure.GPIO\_Pin = Open\_USARTx\_RX\_PIN;

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_OD;

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

GPIO\_Init(Open\_USARTx\_RX\_GPIO\_PORT, &GPIO\_InitStructure);

#if defined HwFlowControl && !defined OpenUART4 && !defined OpenUART5

RCC\_AHB1PeriphClockCmd(Open\_USARTx\_RTS\_GPIO\_CLK,ENABLE);

RCC\_AHB1PeriphClockCmd(Open\_USARTx\_CTS\_GPIO\_CLK,ENABLE);

GPIO\_PinAFConfig(Open\_USARTx\_RTS\_GPIO\_PORT, Open\_USARTx\_RTS\_SOURCE, Open\_USARTx\_RTS\_AF);

GPIO\_PinAFConfig(Open\_USARTx\_CTS\_GPIO\_PORT, Open\_USARTx\_CTS\_SOURCE, Open\_USARTx\_CTS\_AF);

GPIO\_InitStructure.GPIO\_Pin = Open\_USARTx\_RTS\_PIN;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF;

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;

GPIO\_Init(Open\_USARTx\_RTS\_GPIO\_PORT, &GPIO\_InitStructure);

GPIO\_InitStructure.GPIO\_Pin = Open\_USARTx\_CTS\_PIN;

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_OD;

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

GPIO\_Init(Open\_USARTx\_CTS\_GPIO\_PORT, &GPIO\_InitStructure);

#endif

/\*

USARTx configured as follow:

- BaudRate = 115200 baud

- Word Length = 8 Bits

- One Stop Bit

- No parity

- Hardware flow control disabled (RTS and CTS signals)

- Receive and transmit

\*/

USART\_InitStructure.USART\_BaudRate = 115200;

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;

USART\_InitStructure.USART\_Parity = USART\_Parity\_No;

#if defined HwFlowControl && !defined OpenUART4 && !defined OpenUART5

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_RTS\_CTS;

#else

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

#endif

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;

USART\_Init(Open\_USARTx, &USART\_InitStructure);

/\* Enable the Open\_USART Transmit interrupt: this interrupt is generated when the

Open\_USARTx transmit data register is empty \*/

USART\_ITConfig(Open\_USARTx,USART\_IT\_RXNE,ENABLE);

USART\_Cmd(Open\_USARTx, ENABLE);

}

void USART\_NVIC\_Config(void)

{

NVIC\_InitTypeDef NVIC\_InitStructure;

/\* Enable the USARTx Interrupt \*/

NVIC\_InitStructure.NVIC\_IRQChannel = Open\_USARTx\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 1;

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0;

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

NVIC\_Init(&NVIC\_InitStructure);

**}**

**C. Can\_configuration**

void CAN\_Config(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

/\* CAN GPIOs configuration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Enable GPIO clock \*/

RCC\_AHB1PeriphClockCmd(Open\_CAN\_GPIO\_CLK, ENABLE);

RCC\_APB1PeriphClockCmd(Open\_CAN\_CLK, ENABLE);

/\* Connect CAN pins to AF9 \*/

GPIO\_PinAFConfig(Open\_CAN\_GPIO\_PORT, Open\_CAN\_RX\_SOURCE, Open\_CAN\_AF\_PORT);

GPIO\_PinAFConfig(Open\_CAN\_GPIO\_PORT, Open\_CAN\_TX\_SOURCE, Open\_CAN\_AF\_PORT);

/\* Configure CAN RX and TX pins \*/

GPIO\_InitStructure.GPIO\_Pin = Open\_CAN\_RX\_PIN | Open\_CAN\_TX\_PIN;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP;

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP;

GPIO\_Init(Open\_CAN\_GPIO\_PORT, &GPIO\_InitStructure);

/\* CAN configuration \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Enable CAN clock \*/

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_CAN1, ENABLE);

RCC\_APB1PeriphClockCmd(Open\_CAN\_CLK, ENABLE);

/\* CAN register init \*/

CAN\_DeInit(Open\_CANx);

CAN\_StructInit(&CAN\_InitStructure);

/\* CAN cell init \*/

CAN\_InitStructure.CAN\_TTCM = DISABLE;

CAN\_InitStructure.CAN\_ABOM = DISABLE;

CAN\_InitStructure.CAN\_AWUM = DISABLE;

CAN\_InitStructure.CAN\_NART = DISABLE;

CAN\_InitStructure.CAN\_RFLM = DISABLE;

CAN\_InitStructure.CAN\_TXFP = DISABLE;

CAN\_InitStructure.CAN\_Mode = CAN\_Mode\_Normal;

CAN\_InitStructure.CAN\_SJW = CAN\_SJW\_1tq;

/\* CAN Baudrate = 1MBps (CAN clocked at 30 MHz) \*/

CAN\_InitStructure.CAN\_BS1 = CAN\_BS1\_6tq;

CAN\_InitStructure.CAN\_BS2 = CAN\_BS2\_8tq;

CAN\_InitStructure.CAN\_Prescaler = 2;

CAN\_Init(Open\_CANx, &CAN\_InitStructure);

/\* CAN filter init \*/

#ifdef OpenCAN1

CAN\_FilterInitStructure.CAN\_FilterNumber = 0;

#endif

#ifdef OpenCAN2

CAN\_FilterInitStructure.CAN\_FilterNumber = 14;

#endif

CAN\_FilterInitStructure.CAN\_FilterMode = CAN\_FilterMode\_IdMask;

CAN\_FilterInitStructure.CAN\_FilterScale = CAN\_FilterScale\_32bit;

CAN\_FilterInitStructure.CAN\_FilterIdHigh = 0x0000;

CAN\_FilterInitStructure.CAN\_FilterIdLow = 0x0000;

CAN\_FilterInitStructure.CAN\_FilterMaskIdHigh = 0x0000;

CAN\_FilterInitStructure.CAN\_FilterMaskIdLow = 0x0000;

CAN\_FilterInitStructure.CAN\_FilterFIFOAssignment = 0;

CAN\_FilterInitStructure.CAN\_FilterActivation = ENABLE;

CAN\_FilterInit(&CAN\_FilterInitStructure);

/\* Transmit Structure preparation \*/

TxMessage.StdId = 0x321;

TxMessage.ExtId = 0x01;

TxMessage.RTR = CAN\_RTR\_DATA;

TxMessage.IDE = CAN\_ID\_STD;

TxMessage.DLC = 1;

/\* Enable FIFO 0 message pending Interrupt \*/

CAN\_ITConfig(Open\_CANx, CAN\_IT\_FMP0, ENABLE);

}

\* Configures the NVIC for CAN.

\*/

void NVIC\_Config(void)

{

NVIC\_InitTypeDef NVIC\_InitStructure;

#ifdef OpenCAN1

NVIC\_InitStructure.NVIC\_IRQChannel = CAN1\_RX0\_IRQn;

#else /\* USE\_CAN2 \*/

NVIC\_InitStructure.NVIC\_IRQChannel = CAN2\_RX0\_IRQn;

#endif /\* OpenCAN1 \*/

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0x0;

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0x0;

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

NVIC\_Init(&NVIC\_InitStructure);

}

## Example:

## Diagnostic CAN ID

Request Id: 0x712

Tester send request via Request Id to ECU

Tester receives Id from ECU response and then show the value on monitor (PC) via UART3

int main(void)

{

GPIO\_Configuration();

USART\_Configuration();

NVIC\_Config();

CAN\_Config();

//show information on monitor (PC) via UART3

printf("\r\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r\n");

printf("CAN-Bus Test \r\n");

printf("CAN-Bus Speed 100kHz \r\n");

/\* Infinite loop \*/

while (1)

{

if( CanFlag == ENABLE )

{

CanFlag = DISABLE;

printf("CAN Receive Data \r\n");

printf("CAN ID %x \r\n",CAN\_ID);

printf("CAN\_DATA0 %x \r\n",CAN\_DATA0);

printf("CAN\_DATA1 %x \r\n",CAN\_DATA1);

printf("CAN\_DATA2 %x \r\n",CAN\_DATA2);

printf("CAN\_DATA3 %x \r\n",CAN\_DATA3);

printf("CAN\_DATA4 %x \r\n",CAN\_DATA4);

printf("CAN\_DATA5 %x \r\n",CAN\_DATA5);

printf("CAN\_DATA6 %x \r\n",CAN\_DATA6);

printf("CAN\_DATA7 %x \r\n",CAN\_DATA7);

}

CanWriteData(0x712);

if( Display )

{

/\*====LED-ON=======\*/

GPIO\_SetBits(GPIOB , GPIO\_Pin\_0);

GPIO\_SetBits(GPIOB , GPIO\_Pin\_1);

GPIO\_SetBits(GPIOB , GPIO\_Pin\_2);

GPIO\_SetBits(GPIOB , GPIO\_Pin\_3);

}

else

{

/\*====LED-OFF=======\*/

GPIO\_ResetBits(GPIOB , GPIO\_Pin\_0);

GPIO\_ResetBits(GPIOB , GPIO\_Pin\_1);

GPIO\_ResetBits(GPIOB , GPIO\_Pin\_2);

GPIO\_ResetBits(GPIOB , GPIO\_Pin\_3);

}

Display = ~Display;

Delay(); /\* delay 200ms \*/

Delay(); /\* delay 200ms \*/

Delay(); /\* delay 200ms \*/

Delay(); /\* delay 200ms \*/

}

}

**Refer to attached file**

## Practice

In this practice,

- Tester board will send request to ECU board read Wakeup button value and User button value.

- ECU board receives request, Wakeup button value and User button value are stored in “Data7” of frame.

- Tester board receives Wakeup button value and User button value from ECU board response and then show the value on LED1 and LED2.

- Tester board will send request to read Wakeup button value and User button value for each 1 second.

*ID of Tester i: 0xF001*