EEE3088F - Group 7 - Code Reference

MICROCONTROLLER FUNCTIONS

Global variables used by the program:

```
40
41 /* Private variables ---------------------------------------------------------*/
42  ADC_HandleTypeDef hadc;
43
44 I2C_HandleTypeDef hi2c1;
45
46 TIM_HandleTypeDef htim2;
47
48 UART_HandleTypeDef huart1;
49
50 /* USER CODE BEGIN PV */
51
52 uint8_t index = 0;
53
54 //Setup variables for reading and writing
55 uint16_t EEPROM_DEVICE_ADDR = 0x0 ; //Address of EEPROM device on I2C bus
56 uint16_t madd = 0x00; //Memory address variable containing a starting memory address for a location of memory in the EEPROM
57
58
```

```
void HAL_TIM_PeriodElapsedCallback(TIM_handleTypeDef *htim)
Function that runs when Tim2 Interrupt is triggered

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
Function triggered by user typing "Hello HAT"

void debugPrintln(UART_HandleTypeDef *uarthandle, char_out[ ])
Function that improves output to screen

int main(void)
Main function
```

MICROCONTROLLER CODE

The following code runs upon the Tim2 interrupt triggering

```
413 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
414     {
415         uint8_t light_reading = 0
416         uint8_t *sData; //Pointer to sending Data variable
417 
418         Presence = DS18B20_Start ();
419         HAL_Delay (1);
420 
421         DS18B20_Write (0xCC);
422         //write sequence to sensor to skip ROM
423 
424         DS18B20_Write (0x44);
425         // write sequence to convert temperature from the sensors internal memory
426 
427         HAL_Delay (800);
428 
429 
430 
431         Presence = DS18B20_Start ();
432         HAL_Delay(1);
433 
434         DS18B20_Write (0xCC);
435         //write sequence to sensor to skip ROM
436 
437         DS18B20_Write (0xBE);
438         // Read  sensor Scratch-pad (memory)
439 
440         Temp_byte1 = DS18B20_Read();
441         // Read  sensor Scratch-pad (memory) and store first byte
442 
443         Temp_byte2 = DS18B20_Read();
444         // Read  sensor Scratch-pad (memory) and store second byte
445 
446         TEMP = (Temp_byte2<<8)|Temp_byte1;
447         Temperature = (float)TEMP/16;
448 
449         HAL_ADC_Start(&hadc);
450         while(HAL_ADC_PollForConversion (&hadc, 1000) != HAL_OK);
451         light_reading = HAL_ADC_GetValue(&hadc);
452 
453 
454         HAL_GPIO_WritePin(GPIOA, WP_Pin, 1)
455 
456         // Write index to first byte and increment address.
457         sData = &index;
458         I2CReturn = HAL_I2C_Mem_Write(&hi2c1, EEPROM_DEVICE_ADDR, madd, 2, sData, 1, HAL_MAX_DELAY);
459         madd++;
460 
461         // Write light reading to second byte and incrememement address.
462         sData = &light_reading;
463         I2CReturn = HAL_I2C_Mem_Write(&hi2c1, EEPROM_DEVICE_ADDR, madd, 2, sData, 1, HAL_MAX_DELAY);
464         madd++;
465 
466         HAL_GPIO_WritePin(GPIOA, WP_Pin, 0)
467 
468 
469     }
470
```

This function triggers when the user types “Hello HAT” into their UART console

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
    {
        char str[60];

        uint8_t data_index = 0x0;
        uint8_t data_val = 0x0;
        uint8_t *rdata = &data_val; // Pointer to received data

        debugPrintln(&huart1, "Welcome to your HAT.\n\nThese are the readings taken while you were away.");

        for(int i=0; i < index; i=i+3)
        {
            I2CReturn = HAL_I2C_Mem_Read(&hi2c1, EEPROM_DEVICE_ADDR, i, 2, rdata, 1, HAL_MAX_DELAY); // read index from EEPROM

            memset(str, 0, sizeof(str));
            sprintf(str, "%d - Light - Value: 0x%X", i/3, data_val);
            debugPrintln(&huart1, str);

            I2CReturn = HAL_I2C_Mem_Read(&hi2c1, EEPROM_DEVICE_ADDR, i+1, 2, rdata, 1, HAL_MAX_DELAY); // read index from EEPROM

            memset(str, 0, sizeof(str));
            sprintf(str, "%d - Reading No: 0x%X - Value: 0x%X", i/3, data_index, data_val);
            debugPrintln(&huart1, str);
        }

        while(HAL_GPIO_ReadPin(GPIOA, USB_DETECT_Pin))
        {
            if(__HAL_TIM_GET_FLAG(htim, TIM_FLAG_UPDATE))
            {
                uint8_t light_reading = 0
                        uint8_t *sData; //Pointer to sending Data variable

                        Presence = DS18B20_Start ();
                        HAL_Delay (1);

                        DS18B20_Write (0xCC);
                        //write sequence to sensor to skip ROM

                        DS18B20_Write (0x44);
                        // write sequence to convert temperature from the sensors internal memory

                        HAL_Delay (800);

                        Presence = DS18B20_Start ();
                        HAL_Delay(1);

                        DS18B20_Write (0xCC);
                        //write sequence to sensor to skip ROM

                        DS18B20_Write (0xBE);
                        // Read  sensor Scratch-pad (memory)

                        Temp_byte1 = DS18B20_Read();
                        // Read  sensor Scratch-pad (memory) and store first byte

                        Temp_byte2 = DS18B20_Read();
                        // Read  sensor Scratch-pad (memory) and store second byte

                        TEMP = (Temp_byte2<<8)|Temp_byte1;
                        Temperature = (float)TEMP/16;

                        sData = &Temperature;
                        I2CReturn = HAL_I2C_Mem_Write(&hi2c1, EEPROM_DEVICE_ADDR, madd, 2, sData, 1,   HAL_MAX_DELAY);
                        madd= madd+2;

                        HAL_ADC_Start(&hadc);
                        while(HAL_ADC_PollForConversion (&hadc, 1000) != HAL_OK);
                        light_reading = HAL_ADC_GetValue(&hadc);

                        HAL_GPIO_WritePin(GPIOA, WP_Pin, 1)

                        // Write light reading to second byte and incrememement address.
                        sData = &light_reading;
                        I2CReturn = HAL_I2C_Mem_Write(&hi2c1, EEPROM_DEVICE_ADDR, madd, 2, sData, 1, HAL_MAX_DELAY);
                        madd++;

                        HAL_GPIO_WritePin(GPIOA, WP_Pin, 0);
            }
        }

          char UART1_rxBuffer[12];
          sprintf(UART1_rxBuffer, "Hello HAT\n");
          HAL_UART_Receive_IT (&huart1, UART1_rxBuffer, 12);
    }
```

This function streamlines the process of printing information to the screen

```
557 |
558 void debugPrintln(UART_HandleTypeDef *uart_handle, char _out[])
559     {
560         HAL_UART_Transmit(uart_handle, (uint8_t *) _out, strlen(_out), 60);
561         char newline[2] = "\r\n";
562         HAL_UART_Transmit(uart_handle, (uint8_t *) newline, 2, 10);
563     }
```

Main function

The UART is set to wake up on a specific phrase “Hello Hat”. The ticks are suspended. This is done to prevent the system clock from waking the STM during sleep mode.

```
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_ADC_Init();
  MX_I2C1_Init();
  MX_TIM2_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */

  char UART1_rxBuffer[12];
  sprintf(UART1_rxBuffer, "Hello HAT\n");
  HAL_UART_Receive_IT (&huart1, UART1_rxBuffer, 12);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
      HAL_SuspendTick();
      HAL_PWR_EnterSLEEPMode(PWR_MAINREGULATOR_ON, PWR_SLEEPENTRY_WFI);
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

SENSOR FUNCTIONS

```
uint8_t DS18B20_Start (void)
```

Initialises the digital sensor

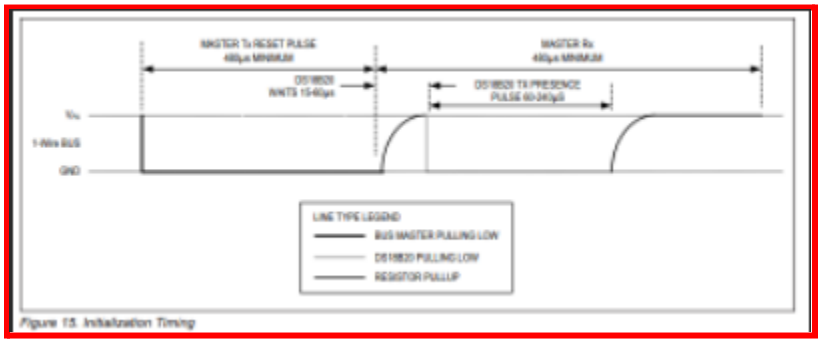
```
void DS18B20_Write (uint8_t data)
```

Takes a byte of data (as an argument) and writes it to the digital sensor

```
uint8_t read (void)
```

Returns a byte from the digital sensors built in memory

INITIALISING DIGITAL SENSOR

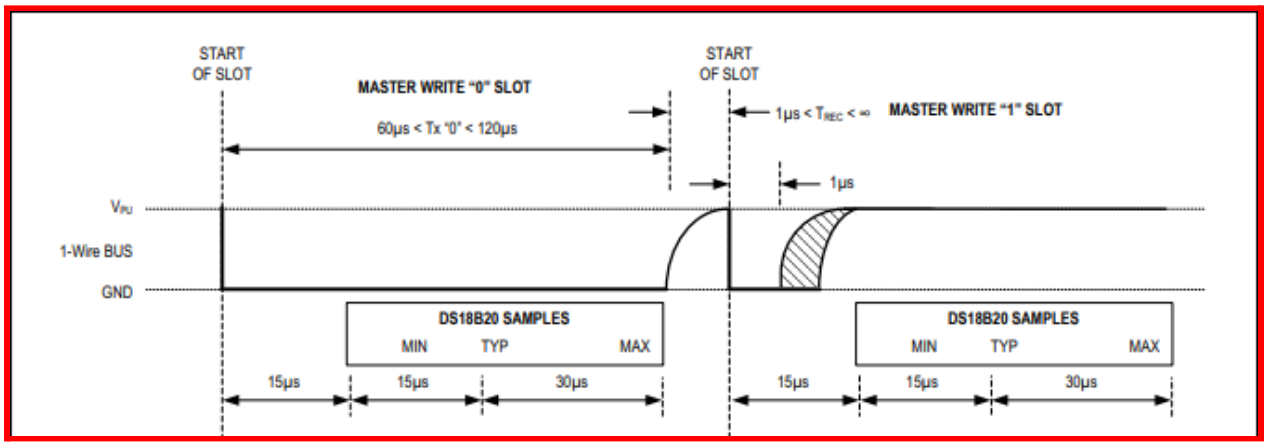


- Each time it's used, the sensor needs initialising
- According to the datasheet, the initialization is done by pulling the data pin LOW for 480 μs and then reading the pin for the presence pulse sent by the sensor. The below function does this

```
uint8_t DS18B20_Start (void)
{
	uint8_t Response = 0;
	Set_Pin_Output(DS18B20_PORT, DS18B20_PIN);   // set the pin as output
	HAL_GPIO_WritePin (DS18B20_PORT, DS18B20_PIN, 0);  // pull the pin
low
	delay (480);   // delay according to data sheet

	Set_Pin_Input(DS18B20_PORT, DS18B20_PIN);    // set the pin as input
	delay (80);    // delay according to data sheet

	if (!(HAL_GPIO_ReadPin (DS18B20_PORT, DS18B20_PIN))) Response = 1;
```

```
// if the pin is low i.e the presence pulse is detected
      else Response = -1;

      delay (400); // 480 us delay totally.

      return Response;}
```

WRITING TO THE SENSOR

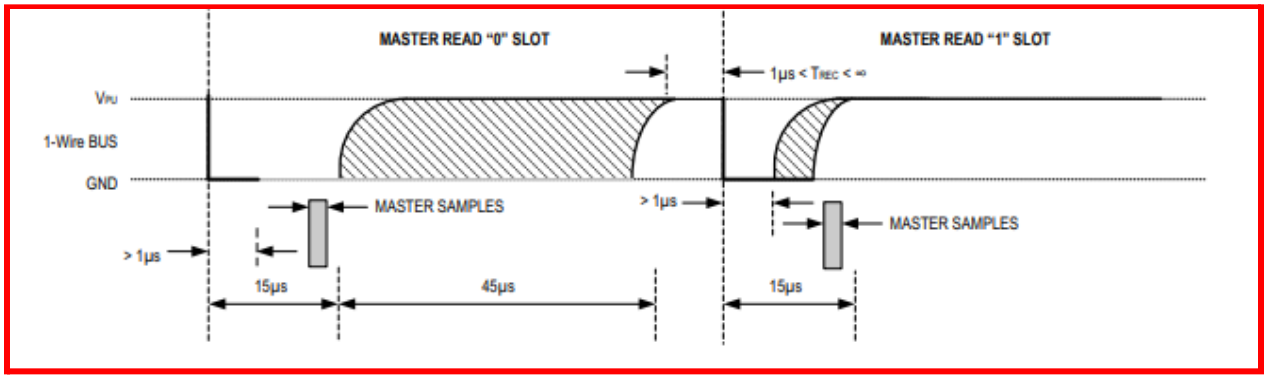


- To generate a Write 1 the master must pull the line low and then release it within 15µs.
- When the line is released the pullup resistor will pull the bus high again
- To generate a Write 0 time slot, after pulling the line low, the master must continue to hold the line low for the duration of the time slot (at least 60µs)
- The function below achieves this

```
void DS18B20_Write (uint8_t data)
{
      Set_Pin_Output(DS18B20_PORT, DS18B20_PIN);  // set as output

      for (int i=0; i<8; i++)
      {

            if ((data & (1<<i))!=0)  // if the bit is high
            {
                  // write 1

                  Set_Pin_Output(DS18B20_PORT, DS18B20_PIN);  // set as
```

```
output
                HAL_GPIO_WritePin (DS18B20_PORT, DS18B20_PIN, 0);  //
pull the pin LOW
                delay (1);  // wait for 1 us

                Set_Pin_Input(DS18B20_PORT, DS18B20_PIN);  // set as
input
                delay (50);  // wait for 60 us
            }

            else  // if the bit is low
            {
                // write 0

                Set_Pin_Output(DS18B20_PORT, DS18B20_PIN);
                HAL_GPIO_WritePin (DS18B20_PORT, DS18B20_PIN, 0);  //
pull the pin LOW
                delay (50);  // wait for 60 us

                Set_Pin_Input(DS18B20_PORT, DS18B20_PIN);
            }
      }
}
```

READING THE TEMPERATURE VALUES



- A read time slot is initiated by the master device (STM) pulling the data line low for a minimum of 1µs and then releasing it.
- After the STM initiates the read time slot, the DS18B20 will begin transmitting a 1 or 0 on the line. It transmits a 1 by leaving the bus high and transmits a 0 by pulling the bus low.

- When transmitting a 0, the sensor will release the bus by the end of the time slot, and the bus will be pulled back to its high idle state by the pull-up resistor

```
uint8_t read (void)
{
	uint8_t value=0;
	gpio_set_input ();

	for (int i=0;i<8;i++)
	{
		gpio_set_output ();   // set as output

		HAL_GPIO_WritePin (GPIOA, GPIO_PIN_1, 0);  // pull the data pin
LOW
		delay (2);  // wait for 2 us

		gpio_set_input ();  // set as input
		if (HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_1))  // if the pin is
HIGH
		{
			value |= 1<<i;  // read = 1
		}
		delay (60);  // wait for 60 us
	}
	return value;
}
```