ООО «Группа индустриальных технологий»

УТВЕРЖДАЮ

Генеральный директор

_____ К.Н. Мигун

«____» _____ 2021 г.

УЗЕЛ ПЕЧАТНЫЙ СИГНАЛИЗАЦИИ

Встроенное программное обеспечение

Текст программы

РОФ.ГРЛМ.03003-01 12 01

Листов 19

2021

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

АННОТАЦИЯ

В данном документе содержится текст программы встроенного программного обеспечения узла печатного сигнализации ГРЛМ.468232.002.

В разделе «Необходимый комплект оборудования и ПО» указаны технические и программные средства, необходимые для просмотра электронной записи исходного кода встроенного программного обеспечения изделия.

В разделе «Характеристики записи текста программы» указаны характеристики исходного кода встроенного программного обеспечения изделия, включая язык программирования, на котором написана программа, и место размещения электронной версии текста программы.

В разделе «Фрагмент текста программы» приведен фрагмент исходного кода встроенного программного обеспечения узла печатного сигнализации.

1 НЕОБХОДИМЫЙ КОМПЛЕКТ ОБОРУДОВАНИЯ И ПО

Для просмотра файла, содержащего текст программы с исходным кодом, требуется следующий комплект оборудования и программного обеспечения:

- персональный компьютер с операционной системой Windows не ниже Windows 7;
- установленное программное обеспечение QT Creator, включающее в себя редактор кода, справочник, графические средства разработки интерфейса Qt Designer и графический интерфейс отладчика приложений;
- текстовый редактор для просмотра ASCII-текстовых файлов.

2 ХАРАКТЕРИСТИКИ ЗАПИСИ ТЕКСТА ПРОГРАММЫ

Исходный код программы написан на языках C++, Python.

Текст программы включает в себя исходный код встроенного программного обеспечения узла печатного сигнализации с комментариями, в которых указаны функциональное назначение представленных процедур.

Текст программы оформлен в форме электронного текстового файла, выполненного с использованием стандартной кодировки ASCII.

Размещение файла: http://gitlab.git-holding.ru:9071/git/meta-git

3 ФРАГМЕНТ ТЕКСТА ПРОГРАММЫ

```
#include "gpio.h"
#include "main.h"
#include "usb.h"
#include <string>
#include <vector>
#include <string.h>
#include <chrono>

#define GIT_VIDEO
//#define GIT_PDKV

int main (int argc, char** argv) {

    __disable_irq();
    RCC_init();
    GPIO_init();
    __enable_irq();

  TIM1_init();
//  IWDT_init();
```

```
    VCom_Configuration();
    USB_CDC_Init(Buffer, 165, SET);
    Setup_CPU_Clock();
    Setup_USB();

    LCD_init();
    Check();

    while (1)
    {
      while(Buffer[0] == 0x00)
      {
        firstScreen();
        delay_ms(1000);
        LCD_set_line(4); LCD_write_string((char*)"                ");
        delay_ms(1000);
//        IWDG_ReloadCounter();    //сбрасываем IWDT
      }

        parsingBuffer();

        if(BufferLCD[0] == 49) //если 1 то горит первый светодиод
        {
          PORT_SetBits(MDR_PORTB, LED2_REC);
        }
        if(BufferLCD[0] == 48)
        {
          PORT_ResetBits(MDR_PORTB, LED2_REC);
        }
        if(BufferLCD[1] == 49) //если 1 то горит второй светодиод
        {
          PORT_SetBits(MDR_PORTB, LED1_ERROR);
        }
        if(BufferLCD[1] == 48)
        {
          PORT_ResetBits(MDR_PORTB, LED1_ERROR);
        }
        if (BufferLCD[2] == 49) //Постоянный сигнал
        {
          PORT_SetBits(MDR_PORTB, BUZZER);
        }
        if(BufferLCD[2] == 48)
        {
          PORT_ResetBits(MDR_PORTB, BUZZER);
        }

        while((Buffer[0] == 55) && (Buffer[1] == 55) && (Buffer[2] == 55))
```

```
            {
               PORT_ResetBits(MDR_PORTB, LED2_REC);
               PORT_ResetBits(MDR_PORTB, LED1_ERROR);
               PORT_ResetBits(MDR_PORTB, BUZZER);
               LCD_set_line(3); LCD_write_string((char*)"                ");
               firstScreen();
               delay_ms(1000);
               LCD_set_line(4); LCD_write_string((char*)"                ");
               delay_ms(1000);
            }

      //       checkLCD1();
      //       checkLCD2();
      //       checkBUZZER();

            char  str1[21]; //4 строки на экране
            char  str2[21];
            char  str3[21];
            char  str4[21];

            memcpy (str1, BufferLCD + 3, 20);
            memcpy (str2, BufferLCD + 23 , 20);
            memcpy (str3, BufferLCD + 43 , 20);
            memcpy (str4, BufferLCD + 63 , 20);

            PrintString1(str1);
            PrintString2(str2);
            PrintString3(str3);
            PrintString4(str4);

//            IWDG_ReloadCounter();    //сбрасываем IWDT
//         }
         }
}

void parsingBuffer()
{
   uint8_t* pBuff = Buffer;
   uint8_t* pBuffLCD = BufferLCD;

   for (int numSymbols = 0; numSymbols < 83; ++numSymbols)
   {
      uint8_t temp = *pBuff;
      if ((temp & 0xD0) != 0xD0)
      {
         *pBuffLCD = *pBuff;
         ++pBuff;
```

```
      ++pBuffLCD;
   }
   else //unicode
   {
      if (*pBuff == 0xD0)
      {
         ++pBuff;
         if (*pBuff == 0x90) *pBuffLCD = 0x41;     //A
         else if (*pBuff == 0x81) *pBuffLCD = 0xA2;//Ё
         else if (*pBuff == 0x91) *pBuffLCD = 0xA0;//Б
         else if (*pBuff == 0x92) *pBuffLCD = 0x42;//В
         else if (*pBuff == 0x93) *pBuffLCD = 0xA1;//Г
         else if (*pBuff == 0x94) *pBuffLCD = 0xE0;//Д
         else if (*pBuff == 0x95) *pBuffLCD = 0x45;//Е
         else if (*pBuff == 0x96) *pBuffLCD = 0xA3;//Ж
         else if (*pBuff == 0x97) *pBuffLCD = 0xA4;//З
         else if (*pBuff == 0x98) *pBuffLCD = 0xA5;//И
         else if (*pBuff == 0x99) *pBuffLCD = 0xA6;//Й
         else if (*pBuff == 0x9A) *pBuffLCD = 0x4B;//К
         else if (*pBuff == 0x9B) *pBuffLCD = 0xA7;//Л
         else if (*pBuff == 0x9C) *pBuffLCD = 0x4D;//М
         else if (*pBuff == 0x9D) *pBuffLCD = 0x48;//Н
         else if (*pBuff == 0x9E) *pBuffLCD = 0x4F;//О
         else if (*pBuff == 0x9F) *pBuffLCD = 0xA8;//П
         else if (*pBuff == 0xA0) *pBuffLCD = 0x50;//Р
         else if (*pBuff == 0xA1) *pBuffLCD = 0x43;//С
         else if (*pBuff == 0xA2) *pBuffLCD = 0x54;//Т
         else if (*pBuff == 0xA3) *pBuffLCD = 0xA9;//У
         else if (*pBuff == 0xA4) *pBuffLCD = 0xAA;//Ф
         else if (*pBuff == 0xA5) *pBuffLCD = 0x58;//Х
         else if (*pBuff == 0xA6) *pBuffLCD = 0xE1;//Ц
         else if (*pBuff == 0xA7) *pBuffLCD = 0xAB;//Ч
         else if (*pBuff == 0xA8) *pBuffLCD = 0xAC;//Ш
         else if (*pBuff == 0xA9) *pBuffLCD = 0xE2;//Щ
         else if (*pBuff == 0xAA) *pBuffLCD = 0xAD;//Ъ
         else if (*pBuff == 0xAB) *pBuffLCD = 0xAE;//Ы
         else if (*pBuff == 0xAC) *pBuffLCD = 0x08;//Ь
         else if (*pBuff == 0xAD) *pBuffLCD = 0xAF;//Э
         else if (*pBuff == 0xAE) *pBuffLCD = 0xB0;//Ю
         else if (*pBuff == 0xAF) *pBuffLCD = 0xB1;//Я
         else if (*pBuff == 0xB0) *pBuffLCD = 0x61;//а
         else if (*pBuff == 0xB1) *pBuffLCD = 0xB2;//б
         else if (*pBuff == 0xB2) *pBuffLCD = 0xB3;//в
         else if (*pBuff == 0xB3) *pBuffLCD = 0xB4;//г
         else if (*pBuff == 0xB4) *pBuffLCD = 0xE3;//д
         else if (*pBuff == 0xB5) *pBuffLCD = 0x65;//е
         else if (*pBuff == 0xB6) *pBuffLCD = 0xB6;//ж
```

```
            else if (*pBuff == 0xB7) *pBuffLCD = 0xB7;//з
            else if (*pBuff == 0xB8) *pBuffLCD = 0xB8;//и
            else if (*pBuff == 0xB9) *pBuffLCD = 0xB9;//й
            else if (*pBuff == 0xBA) *pBuffLCD = 0xBA;//к
            else if (*pBuff == 0xBB) *pBuffLCD = 0xBB;//л
            else if (*pBuff == 0xBC) *pBuffLCD = 0xBC;//м
            else if (*pBuff == 0xBD) *pBuffLCD = 0xBD;//н
            else if (*pBuff == 0xBE) *pBuffLCD = 0x6F;//о
            else if (*pBuff == 0xBF) *pBuffLCD = 0xBE;//п
          }
          else if (*pBuff == 0xD1)
          {
            ++pBuff;
            if (*pBuff == 0x80) *pBuffLCD = 0x70;     //р
            else if (*pBuff == 0x91) *pBuffLCD = 0xB5;//ё
            else if (*pBuff == 0x81) *pBuffLCD = 0x63;//с
            else if (*pBuff == 0x82) *pBuffLCD = 0xBF;//т
            else if (*pBuff == 0x83) *pBuffLCD = 0x79;//у
            else if (*pBuff == 0x84) *pBuffLCD = 0xE4;//ф
            else if (*pBuff == 0x85) *pBuffLCD = 0x78;//х
            else if (*pBuff == 0x86) *pBuffLCD = 0xE5;//ц
            else if (*pBuff == 0x87) *pBuffLCD = 0xC0;//ч
            else if (*pBuff == 0x88) *pBuffLCD = 0xC1;//ш
            else if (*pBuff == 0x89) *pBuffLCD = 0xE6;//щ
            else if (*pBuff == 0x8A) *pBuffLCD = 0xC2;//ъ
            else if (*pBuff == 0x8B) *pBuffLCD = 0xC3;//ы
            else if (*pBuff == 0x8C) *pBuffLCD = 0xC4;//ь
            else if (*pBuff == 0x8D) *pBuffLCD = 0xC5;//э
            else if (*pBuff == 0x8E) *pBuffLCD = 0xC6;//ю
            else if (*pBuff == 0x8F) *pBuffLCD = 0xC7;//я
          }
          else //другой символ
          {
            ++pBuff;
            *pBuffLCD = 0x20;
          }

          ++pBuff;
          ++pBuffLCD;
        }
      }

      USB_CDC_Reset(); //Экран заполнен
    }

    void firstScreen()
    {
```

```
#ifdef GIT_PDKV
   LCD_set_line(1);
   LCD_write_string((char*)"    GIT-COMM IPS    ");
   LCD_set_XY(9, 2);
   LCD_write_data(0xA8);//П
   LCD_write_data(0xE0);//Д
   LCD_write_data(0x4B);//К
   LCD_write_data(0x42);//В
   LCD_set_XY(6, 4);
   LCD_write_data(0xA4);//З
   LCD_write_data(0x61);//а
   LCD_write_data(0xB4);//г
   LCD_write_data(0x70);//р
   LCD_write_data(0x79);//у
   LCD_write_data(0xB7);//з
   LCD_write_data(0xBA);//к
   LCD_write_data(0x61);//а
   LCD_write_string((char*)"...");
#endif
#ifdef GIT_VIDEO
   LCD_set_line(1);
   LCD_write_string((char*)"   GIT-VIDEO  SURV  ");
   LCD_set_line(2);
   LCD_write_string((char*)"      GDMX S       ");
   LCD_set_XY(6, 4);
   LCD_write_data(0xA4);//З
   LCD_write_data(0x61);//а
   LCD_write_data(0xB4);//г
   LCD_write_data(0x70);//р
   LCD_write_data(0x79);//у
   LCD_write_data(0xB7);//з
   LCD_write_data(0xBA);//к
   LCD_write_data(0x61);//а
   LCD_write_string((char*)"...");
#endif
}

#include "main.h"
#include <chrono>

void checkLCD1()
{
   if(Buffer[0] == 49) //если 1 то горит первый светодиод
   {
      PORT_SetBits(MDR_PORTB, LED2_REC);
   }
   if(Buffer[0] == 48) //если 0 то гасим
```

```
    {
      PORT_ResetBits(MDR_PORTB, LED2_REC);
    }

  }

  void checkLCD2()
  {
    if(Buffer[1] == 49) //если 1 то горит второй светодиод
    {
      PORT_SetBits(MDR_PORTB, LED1_ERROR);
    }
    if(Buffer[1] == 48) //если 0 то гасим
    {
      PORT_ResetBits(MDR_PORTB, LED1_ERROR);
    }
  }

  void checkBUZZER()
  {
  #if 0
```

Плата индикации должна обеспечивать выдачу звуковых сигналов в следующих случаях:

0. Выключен
1. Постоянный сигнал
2. Два сигнала (звучание в течение 0,5с с интервалом 0,5с).
3. Три сигнала (звучание в течение 0,5с с интервалом 0,5с).
4. Четыре сигнала (звучание в течение 0,5с с интервалом 0,5с).
5. Один сигнал (звучание в течение 0,5с с интервалом 60с).
6. Непрерывная последовательность (звучание в течение 0,5с с интервалом 0,5с).

```
  #endif
    if (Buffer[2] == 49) //Постоянный сигнал
    {
      PORT_SetBits(MDR_PORTB, BUZZER);
    }
    if(Buffer[2] == 48) //если 0 то гасим
    {
      PORT_ResetBits(MDR_PORTB, BUZZER);
    }

  }

  void checkCMD()
  {
    if((Buffer[0] == 55) && (Buffer[1] == 55) && (Buffer[2] == 55))
    {
```

```
        firstScreen();
    }

}

#include "MDR32Fx.h"
#include "MDR32F9Qx_rst_clk.h"
#include "MDR32F9Qx_port.h"
#include "main.h"
#include "gpio.h"
#include "usb.h"
#include <string>

void Check() {
      LCD_control(1,0,0);          //включение дисплея и выбор курсора
      LCD_entry_mode(1,0);      //установка направления сдвига
      LCD_clear_display();        //установка курсора в начало
    LCD_function(1,0);          //страница знакогенератора 0
      LCD_clear_display();

//    PORT_SetBits(MDR_PORTB, LED1_ERROR);//проверка светодиодов
//    PORT_SetBits(MDR_PORTB, LED2_REC);
//    delay_ms(1000);
//    PORT_ResetBits(MDR_PORTB, LED1_ERROR);
//    PORT_ResetBits(MDR_PORTB, LED2_REC);
#if 0
    LCD_set_line(1);
    LCD_write_data(0x41);//A
    LCD_write_data(0xA0);//Б
    LCD_write_data(0x42);//B
    LCD_write_data(0xA1);//Г
    LCD_write_data(0xE0);//Д
    LCD_write_data(0x45);//E
    LCD_write_data(0xA2);//Ё
    LCD_write_data(0xA3);//Ж
    LCD_write_data(0xA4);//З
    LCD_write_data(0xA5);//И
    LCD_write_data(0xA6);//Й
    LCD_write_data(0x4B);//K
    LCD_write_data(0xA7);//Л
    LCD_write_data(0x4D);//M
    LCD_write_data(0x48);//H
    LCD_write_data(0x4F);//O
    LCD_write_data(0xA8);//П
    LCD_write_data(0x50);//P
    LCD_write_data(0x43);//C
    LCD_write_data(0x54);//T
```

```
LCD_set_line(2);
LCD_write_data(0xA9);//У
LCD_write_data(0xAA);//Ф
LCD_write_data(0x58);//Х
LCD_write_data(0xE1);//Ц
LCD_write_data(0xAB);//Ч
LCD_write_data(0xAC);//Ш
LCD_write_data(0xE2);//Щ
LCD_write_data(0xAD);//Ъ
LCD_write_data(0xAE);//Ы
LCD_write_data(0x08);//Ь
LCD_write_data(0xAF);//Э
LCD_write_data(0xB0);//Ю
LCD_write_data(0xB1);//Я
LCD_write_data(0x61);//а
LCD_write_data(0xB2);//б
LCD_write_data(0xB3);//в
LCD_write_data(0xB4);//г
LCD_write_data(0xE3);//д
LCD_write_data(0x65);//е
LCD_write_data(0xB5);//ё

LCD_set_line(3);
LCD_write_data(0xB6);//ж
LCD_write_data(0xB7);//з
LCD_write_data(0xB8);//и
LCD_write_data(0xB9);//й
LCD_write_data(0xBA);//к
LCD_write_data(0xBB);//л
LCD_write_data(0xBC);//м
LCD_write_data(0xBD);//н
LCD_write_data(0x6F);//о
LCD_write_data(0xBE);//п
LCD_write_data(0x70);//р
LCD_write_data(0x63);//с
LCD_write_data(0xBF);//т
LCD_write_data(0x79);//у
LCD_write_data(0xE4);//ф
LCD_write_data(0x78);//х
LCD_write_data(0xE5);//ц
LCD_write_data(0xC0);//ч
LCD_write_data(0xC1);//ш
LCD_write_data(0xE6);//щ

LCD_set_line(4);
LCD_write_data(0xC2);//ъ
```

```
    LCD_write_data(0xC3);//ы
    LCD_write_data(0xC4);//ь
    LCD_write_data(0xC5);//э
    LCD_write_data(0xC6);//ю
    LCD_write_data(0xC7);//я

//    проверка зуммера
    PORT_SetBits(MDR_PORTB, BUZZER);
    delay_ms(200);
    PORT_ResetBits(MDR_PORTB, BUZZER);
#endif

}

void PrintString1(char* str) {
    LCD_set_XY(1, 1);
    std::string ss{str, 20};
    LCD_write_string((char*)ss.c_str());
//    delay_ms(10);
}

void PrintString2(char* str) {
    LCD_set_XY(1, 2);
    std::string ss{str, 20};
    LCD_write_string((char*)ss.c_str());
//    delay_ms(10);
}

void PrintString3(char* str) {
    LCD_set_XY(1, 3);
    std::string ss{str, 20};
    LCD_write_string((char*)ss.c_str());
//    delay_ms(10);
}

void PrintString4(char* str) {
    LCD_set_XY(1, 4);
    std::string ss{str, 20};
    LCD_write_string((char*)ss.c_str());
//    delay_ms(10);
}

#include "usb.h"
#include "main.h"
#include <string>

USB_Clock_TypeDef USB_Clock_InitStruct;
```

```
USB_DeviceBUSParam_TypeDef USB_DeviceBUSParam;

#ifdef USB_VCOM_SYNC
  volatile uint32_t PendingDataLength = 0;
#endif /* USB_VCOM_SYNC */

/* USB protocol debugging */
#ifdef USB_DEBUG_PROTO
  #define USB_DEBUG_NUM_PACKETS   100
  typedef struct {
    USB_SetupPacket_TypeDef packet;
    uint32_t address;
  } TDebugInfo;
  static TDebugInfo SetupPackets[USB_DEBUG_NUM_PACKETS];
  static uint32_t SPIndex;
#endif /* USB_DEBUG_PROTO */

void Setup_CPU_Clock(void)
{
  /* Enable HSE */
  RST_CLK_HSEconfig(RST_CLK_HSE_ON);

  while(RST_CLK_HSEstatus() != SUCCESS)
  {
    delay_ms(100);
    LCD_write_string((char*)"Error CPU_Clock");
  }

  /* CPU_C1_SEL = HSE */
  RST_CLK_CPU_PLLconfig(RST_CLK_CPU_PLLsrcHSEdiv1,
RST_CLK_CPU_PLLmul10);
  RST_CLK_CPU_PLLcmd(ENABLE);

  while(RST_CLK_CPU_PLLstatus() != SUCCESS)
  {
    delay_ms(100);
    LCD_write_string((char*)"Error CPU_Clock");
  }

  /* CPU_C3_SEL = CPU_C2_SEL */
  RST_CLK_CPUclkPrescaler(RST_CLK_CPUclkDIV1);
  /* CPU_C2_SEL = PLL */
  RST_CLK_CPU_PLLuse(ENABLE);
  /* HCLK_SEL = CPU_C3_SEL */
  RST_CLK_CPUclkSelection(RST_CLK_CPUclkCPU_C3);
}
```

```
void Setup_USB(void)
{
  /* Включение тактирования */
  RST_CLK_PCLKcmd(RST_CLK_PCLK_USB, ENABLE);

  /* Выбор источника тактирования USB */
  USB_Clock_InitStruct.USB_USBC1_Source = USB_C1HSEdiv2;

  /* Выбор коэффициента умножения схемы PLL для USB */
  USB_Clock_InitStruct.USB_PLLUSBMUL    = USB_PLLUSBMUL12;

  /* Выбор режима USB FULL Speed */
  USB_DeviceBUSParam.MODE  = USB_SC_SCFSP_Full;

  /* Выбор скорости 12мБит в сек */
  USB_DeviceBUSParam.SPEED = USB_SC_SCFSR_12Mb;

  /* Подтягивание линии DP к питанию */
  USB_DeviceBUSParam.PULL  = USB_HSCR_DP_PULLUP_Set;

  /* Инициализация USB с заданными параметрами */
  USB_DeviceInit(&USB_Clock_InitStruct, &USB_DeviceBUSParam);

  /* Разрешение всех видов прерываний от USB */
  USB_SetSIM(USB_SIS_Msk);

  /* Включение питания USB и разрешение передачи и приема данных */
  USB_DevicePowerOn();

  /* Включение прерываний USB */
#ifdef USB_INT_HANDLE_REQUIRED
  NVIC_EnableIRQ(USB_IRQn);
#endif /* USB_INT_HANDLE_REQUIRED */

  USB_DEVICE_HANDLE_RESET;
}

/* Данная процедура автоматически вызывается при приеме данных по USB
*/
USB_Result USB_CDC_RecieveData(uint8_t* Buffer, uint32_t Length)
{
  USB_Result result;

#ifdef USB_DEBUG_PROTO
  ReceivedByteCount += Length;
#endif /* USB_DEBUG_PROTO */
```

```
      /* Передача (одного байта?) назад на устройство */
    //   result = USB_CDC_SendData(Buffer, Length);

    #ifdef USB_DEBUG_PROTO
      if (result == USB_SUCCESS)
      {
        SentByteCount += Length;
      }
    #ifndef USB_VCOM_SYNC
      else
      {
        SkippedByteCount += Length;
      }
    #endif /* !USB_VCOM_SYNC */
    #endif /* USB_DEBUG_PROTO */

    #ifdef USB_VCOM_SYNC
      if (result != USB_SUCCESS)
      {
        /* If data cannot be sent now, it will await nearest possibility
        * (see USB_CDC_DataSent) */
        PendingDataLength = Length;
      }
      return result;
    #else
      return USB_SUCCESS;
    #endif /* USB_VCOM_SYNC */
    }

    #ifdef USB_VCOM_SYNC

    /**
     * @brief   USB_CDC_HANDLE_DATA_SENT implementation - sending of pending data
     * @param  None
     * @retval @ref USB_Result.
     */
    USB_Result USB_CDC_DataSent(void)
    {
      USB_Result result = USB_SUCCESS;

      if (PendingDataLength)
      {
        result = USB_CDC_SendData(Buffer, PendingDataLength);
    #ifdef USB_DEBUG_PROTO
        if (result == USB_SUCCESS)
        {
```

```
            SentByteCount += PendingDataLength;
          }
          else
          {
            SkippedByteCount += PendingDataLength;
          }
        #endif /* USB_DEBUG_PROTO */
          PendingDataLength = 0;
          USB_CDC_ReceiveStart();
        }
        return USB_SUCCESS;
      }

      #endif /* USB_VCOM_SYNC */

      #ifdef USB_CDC_LINE_CODING_SUPPORTED

      /**
       * @brief  USB_CDC_HANDLE_GET_LINE_CODING implementation example
       * @param  wINDEX: Request value 2nd word (wIndex)
       * @param  DATA: Pointer to the USB_CDC Line Coding Structure
       * @retval @ref USB_Result.
       */

      USB_Result USB_CDC_GetLineCoding(uint16_t wINDEX,
USB_CDC_LineCoding_TypeDef* DATA)
      {
        assert_param(DATA);
        if (wINDEX != 0)
        {
          /* Invalid interface */
          return USB_ERR_INV_REQ;
        }

        /* Just store received settings */
        *DATA = LineCoding;
        return USB_SUCCESS;
      }

      /**
       * @brief  USB_CDC_HANDLE_SET_LINE_CODING implementation example
       * @param  wINDEX: Request value 2nd word (wIndex)
       * @param  DATA: Pointer to the USB_CDC Line Coding Structure
       * @retval @ref USB_Result.
       */
      USB_Result USB_CDC_SetLineCoding(uint16_t wINDEX, const
USB_CDC_LineCoding_TypeDef* DATA)
```

```
{
  assert_param(DATA);
  if (wINDEX != 0)
  {
    /* Invalid interface */
    return USB_ERR_INV_REQ;
  }

  /* Just send back settings stored earlier */
  LineCoding = *DATA;
  return USB_SUCCESS;
}

#endif /* USB_CDC_LINE_CODING_SUPPORTED */

#ifdef USB_DEBUG_PROTO

/**
 * @brief  Overwritten USB_DEVICE_HANDLE_SETUP default handler - to dump received setup packets
 * @param  EPx: USB Control EndPoint (EP0) number
 * @param  USB_SetupPacket: Pointer to a USB_SetupPacket_TypeDef structure
 *         that contains received setup packet contents (on success)
 * @retval @ref USB_Result.
 */
USB_Result USB_DeviceSetupPacket_Debug(USB_EP_TypeDef EPx, const USB_SetupPacket_TypeDef* USB_SetupPacket)
{

#ifdef USB_DEBUG_PROTO
  SetupPackets[SPIndex].packet = *USB_SetupPacket;
  SetupPackets[SPIndex].address = USB_GetSA();
  SPIndex = (SPIndex < USB_DEBUG_NUM_PACKETS ? SPIndex + 1 : 0);
#endif /* USB_DEBUG_PROTO */

  return USB_DeviceSetupPacket(EPx, USB_SetupPacket);
}

#endif /* USB_DEBUG_PROTO */

/**
 * @brief  Reports the source file ID, the source line number
 *         and expression text (if USE_ASSERT_INFO == 2) where
 *         the assert_param error has occurred.
 * @param  file_id: pointer to the source file name
 * @param  line: assert_param error line source number
 * @param  expr:
```

```
  * @retval None
  */
#if (USE_ASSERT_INFO == 1)
void assert_failed(uint32_t file_id, uint32_t line)
{
  /* User can add his own implementation to report the source file ID and line number.
     Ex: printf("Wrong parameters value: file Id %d on line %d\r\n", file_id, line) */

  /* Infinite loop */
  while (1)
  {
  }
}
#elif (USE_ASSERT_INFO == 2)
void assert_failed(uint32_t file_id, uint32_t line, const uint8_t* expr);
{
  /* User can add his own implementation to report the source file ID, line number and
     expression text.
     Ex: printf("Wrong parameters value (%s): file Id %d on line %d\r\n", expr, file_id, line) */

  /* Infinite loop */
  while (1)
  {
  }
}
#endif /* USE_ASSERT_INFO */

/** @} */ /* End of group USB_Virtual_COM_Port_Echo_92 */

/** @} */ /* End of group __MDR1986VE92_EVAL */

/** @} */ /* End of group __MDR32Fx_StdPeriph_Examples */
```

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ докум.	Входящий № сопроводительного докум. и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					