**ППРИЛОЖЕНИЕ Г – Исходный код программы**

Исходный код МК:

Main.c:

/\* Includes ------------------------------------------------------------------\*/

#include "MDR32F9Qx\_config.h" // Keil::Device:Startup

#include "MDR32Fx.h"

#include "MDR32F9Qx\_uart.h"

#include "MDR32F9Qx\_port.h"

#include "MDR32F9Qx\_rst\_clk.h"

#include "1986BE9x\_it.h"

#include "MDR32F9Qx\_ssp.h" // Keil::Drivers:SSP

#include "MDR32F9Qx\_can.h" // Keil::Drivers:CAN

#include "mlt\_lcd.h"

#define Ram

#define BLINK\_NUM 50

#define BLINK\_DELAY 20000

#define DELAY(T) for (int i = T; i > 0; i--)

#define LED1 PORT\_Pin\_0

#define LED2 PORT\_Pin\_1

int count\_title = 0;

int MenuMainItem = 0;

int CursorPosItem = 0;

char UartFlag = 0;

char SSPFlag = 0;

char TestSSPDone = 0;

char TestCanDone = 0;

char TestUartDone = 0;

char USBFlag = 0;

char CanFlag = 0;

char FullTestFlag = 0;

char FullTestUSBGetFlag = 0;

char FullTestCANFlag = 0;

char FullTestUARTFlag = 0;

char FullTestSSPFlag = 0;

char FullTestUSBSendFlag = 0;

char FullTestUSBInfoFlag = 0;

char FullTestUSBSendInfoFlag = 0;

char FullTestUARTGetFlag = 0;

uint8\_t USBInfo;

uint8\_t USBFinalInfo;

\_\_IO uint32\_t tx\_buf = 1;

static PORT\_InitTypeDef PortInit;

//Задержка

void Delay(\_\_IO uint32\_t nCount)

{

for (; nCount != 0; nCount--);

}

//Настройка пинов для LCD

void MltPinCfg (void)

{

PortInit.PORT\_PULL\_UP = PORT\_PULL\_UP\_OFF;

PortInit.PORT\_PULL\_DOWN = PORT\_PULL\_DOWN\_OFF;

PortInit.PORT\_PD\_SHM = PORT\_PD\_SHM\_OFF;

PortInit.PORT\_PD = PORT\_PD\_DRIVER;

PortInit.PORT\_GFEN = PORT\_GFEN\_OFF;

PortInit.PORT\_Pin = (PORT\_Pin\_0 | PORT\_Pin\_1 | PORT\_Pin\_2 | PORT\_Pin\_3 | PORT\_Pin\_4 | PORT\_Pin\_5);

PortInit.PORT\_OE = PORT\_OE\_IN;

PortInit.PORT\_FUNC = PORT\_FUNC\_PORT;

PortInit.PORT\_MODE = PORT\_MODE\_DIGITAL;

PortInit.PORT\_SPEED = PORT\_SPEED\_SLOW;

PORT\_Init(MDR\_PORTA, &PortInit);

PortInit.PORT\_Pin = (PORT\_Pin\_2 | PORT\_Pin\_3);

PORT\_Init(MDR\_PORTF, &PortInit);

PortInit.PORT\_Pin = (PORT\_Pin\_7 | PORT\_Pin\_8 | PORT\_Pin\_9 | PORT\_Pin\_10);

PortInit.PORT\_OE = PORT\_OE\_OUT;

PORT\_Init(MDR\_PORTB, &PortInit);

PortInit.PORT\_Pin = (PORT\_Pin\_0 | PORT\_Pin\_1);

PORT\_Init(MDR\_PORTC, &PortInit);

}

//Настройка пинов для LEDов

void LedPinGfg (void)

{

PortInit.PORT\_Pin = (PORT\_Pin\_0 | PORT\_Pin\_1);

PortInit.PORT\_OE = PORT\_OE\_OUT;

PortInit.PORT\_FUNC = PORT\_FUNC\_PORT;

PortInit.PORT\_MODE = PORT\_MODE\_DIGITAL;

PortInit.PORT\_SPEED = PORT\_SPEED\_SLOW;

PORT\_Init(MDR\_PORTC, &PortInit);

}

//Настройка пинов для кнопок

void ButtonPinGfg (void)

{

PortInit.PORT\_Pin = (PORT\_Pin\_5 | PORT\_Pin\_6);

PortInit.PORT\_OE = PORT\_OE\_IN;

PortInit.PORT\_FUNC = PORT\_FUNC\_PORT;

PortInit.PORT\_MODE = PORT\_MODE\_DIGITAL;

PortInit.PORT\_SPEED = PORT\_SPEED\_SLOW;

PORT\_Init(MDR\_PORTB, &PortInit);

PortInit.PORT\_Pin = (PORT\_Pin\_2);

PORT\_Init(MDR\_PORTC, &PortInit);

PortInit.PORT\_Pin = (PORT\_Pin\_1 | PORT\_Pin\_3);

PORT\_Init(MDR\_PORTE, &PortInit);

}

//Настройка и запуск таймера

void Tim1Setup(void)

{

//Разрешение тактирвоания

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_TIMER1, ENABLE);

MDR\_RST\_CLK->TIM\_CLOCK = 0x07000000;

MDR\_TIMER1->CNTRL = 0x00000000;

MDR\_TIMER1->CNT = 0x00000000; //стартовое число

MDR\_TIMER1->PSG = 0x0000003F; //предделитель

MDR\_TIMER1->ARR = 0x0000AFFF; //конечное число

MDR\_TIMER1->CH1\_CNTRL = 0x00000000;

MDR\_TIMER1->CH2\_CNTRL = 0x00000000;

MDR\_TIMER1->CH3\_CNTRL = 0x00000000;

MDR\_TIMER1->CH4\_CNTRL = 0x00000000;

MDR\_TIMER1->CH1\_CNTRL1 = 0x00000000;

MDR\_TIMER1->CH2\_CNTRL1 = 0x00000000;

MDR\_TIMER1->CH3\_CNTRL1 = 0x00000000;

MDR\_TIMER1->CH4\_CNTRL1 = 0x00000000;

MDR\_TIMER1->STATUS = 0x00000000;

MDR\_TIMER1->IE = 0x00000002;

MDR\_TIMER1->CNTRL |= 0x00000001;

}

//Тест CAN

int CanTest (void)

{

CAN\_ports\_ini();

CAN\_Setup();

CAN\_TxMsgTypeDef TxMsg;

DELAY(1000000);

//Передача данных (инициализация сообщения)

TxMsg.IDE = CAN\_ID\_EXT;

TxMsg.DLC = 0x04;

TxMsg.PRIOR\_0 = DISABLE;

TxMsg.ID = 0x15555555;

TxMsg.Data[1] = 0;

TxMsg.Data[0] = 0x12345678;

CAN\_Transmit(MDR\_CAN1, tx\_buf, &TxMsg);

TestCanDone = 1;

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit;

}

} while (1);

exit: return 0;

}

// Тест UART

int UartTest (void)

{

uint8\_t DataByte = 'm';

Uart2PinCfg();

Uart2Setup();

DELAY(1000000);

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit;

}

} while (UART\_GetFlagStatus (MDR\_UART2, UART\_FLAG\_TXFE)!= SET);

// Передача данных

UART\_SendData (MDR\_UART2,DataByte);

TestUartDone = 1;

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit;

}

} while (1);

exit: return 0;

}

//Тест SSP

void SSPTest(void) {

uint16\_t Src1 = 0x0002;

uint16\_t Dst1 = 0x0000;

PortsSSP2Init();

SSP2StructInit();

DELAY(1000000);

while (SSP\_GetFlagStatus(MDR\_SSP2, SSP\_FLAG\_TFE) == RESET)

{

}

//Отправка данных

SSP\_SendData(MDR\_SSP2, Src1);

TestSSPDone = 1;

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit;

}

} while (1);

exit: return 0;

}

//Тест USB

void USBTest (void)

{

USB\_main();

while(1)

{

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit;

}

} while (1);

}

exit:

//Возвращение к нормальной тактовой частоте

USB\_off();

NVIC\_DisableIRQ(USB\_IRQn);

return 0;

}

//Полный тест

void FullTest (void)

{

//Получение данных по USB

FullTestUSBGetFlag = 1;

USB\_main();

while(1)

{

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit;

}

} while (FullTestUSBInfoFlag != 1);

USB\_off();

NVIC\_DisableIRQ(USB\_IRQn);

DELAY(1000000);

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit0;

}

} while ( PORT\_ReadInputDataBit(MDR\_PORTC,PORT\_Pin\_2));

//Отправка данных по SSP

FullTestUSBGetFlag = 0;

FullTestUSBInfoFlag = 0;

USBInfo = USBInfo + 1;

FullTestSSPFlag = 1;

PortsSSP2Init();

SSP2StructInit();

while (SSP\_GetFlagStatus(MDR\_SSP2, SSP\_FLAG\_TFE) == RESET)

{

}

SSP\_SendData(MDR\_SSP2, USBInfo);

DELAY(1000000);

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit0;

}

} while ( PORT\_ReadInputDataBit(MDR\_PORTC,PORT\_Pin\_2));

//Получение данных по UART

FullTestSSPFlag = 0;

FullTestCANFlag = 0;

FullTestUARTFlag = 1;

FullTestUARTGetFlag = 0;

Uart2PinCfg();

Uart2Setup();

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit0;

}

} while (UART\_GetFlagStatus (MDR\_UART2, UART\_FLAG\_RXFF)!= SET);

// Отправка данных по USB

USBInfo = UART\_ReceiveData (MDR\_UART2);

FullTestUARTGetFlag = 1;

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit0;

}

} while ( PORT\_ReadInputDataBit(MDR\_PORTC,PORT\_Pin\_2));

FullTestUARTFlag = 0;

FullTestUARTGetFlag = 0;

FullTestUSBSendInfoFlag = 1;

FullTestUSBInfoFlag = 0;

USB\_main();

USBFinalInfo = USBInfo + 1;

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit;

}

} while (FullTestUSBInfoFlag != 1);

USB\_off();

NVIC\_DisableIRQ(USB\_IRQn);

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit0;

}

} while ( PORT\_ReadInputDataBit(MDR\_PORTC,PORT\_Pin\_2));

//Отправка данных по CAN

FullTestUSBSendInfoFlag = 0;

FullTestUSBInfoFlag = 0;

CAN\_ports\_ini();

CAN\_Setup();

DELAY(1000000);

USBInfo = USBFinalInfo + 1;

CAN\_TxMsgTypeDef TxMsg;

DELAY(1000000);

FullTestCANFlag = 1;

TxMsg.IDE = CAN\_ID\_EXT;

TxMsg.DLC = 0x04;

TxMsg.PRIOR\_0 = DISABLE;

TxMsg.ID = 0x15555555;

TxMsg.Data[1] = 0;

TxMsg.Data[0] = USBInfo;

CAN\_Transmit(MDR\_CAN1, tx\_buf, &TxMsg);

do

{

if (PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_3) == 0)

{

goto exit0;

}

} while (1);

}

exit:

USB\_off();

NVIC\_DisableIRQ(USB\_IRQn);

return 0;

exit0:

return 0;

}

//Основная программа

int main (void)

{

char s1;

//Натсройка тактирования

RST\_CLK\_LSEconfig(RST\_CLK\_LSE\_ON);

while (RST\_CLK\_LSEstatus() != SUCCESS);

RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);

while (RST\_CLK\_HSEstatus() != SUCCESS);

RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSIdiv2, RST\_CLK\_CPU\_PLLmul1);

RST\_CLK\_CPU\_PLLcmd(ENABLE);

if (RST\_CLK\_CPU\_PLLstatus() != SUCCESS)

{

/\* Trap \*/

while (1)

{

}

}

RST\_CLK\_CPUclkPrescaler(RST\_CLK\_CPUclkDIV1);

RST\_CLK\_CPU\_PLLuse(ENABLE);

RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTA, ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTB, ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTC, ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTD, ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTE, ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTF, ENABLE);

MltPinCfg ();

ButtonPinGfg ();

LcdInit();

do

{

s1 = ReadStatus(1);

} while (s1 & 0x80 != 0x00);

DispOn (1);

do

{

s1 = ReadStatus(2);

} while (s1 & 0x80 != 0x00);

DispOn (2);

LcdClearChip (1);

LcdClearChip (2);

SCB->AIRCR = 0x05FA0000 | 0x00000500;//configure

SCB->VTOR = 0x08000000; //NVIC

NVIC->ISER[0] = 0x00004000;

Tim1Setup();

while (1)

{

//Выбор

if ( ! PORT\_ReadInputDataBit(MDR\_PORTC,PORT\_Pin\_2) )

{

while ( ! PORT\_ReadInputDataBit(MDR\_PORTC,PORT\_Pin\_2) ) {};

if (MenuMainItem == 0)

{

CanFlag = 1;

CanTest();

CanFlag = 0;

TestCanDone = 0;

}

if (MenuMainItem == 1)

{

SSPFlag = 1;

SSPTest();

SSPFlag = 0;

TestSSPDone = 0;

}

if (MenuMainItem == 2)

{

USBFlag = 1;

USBTest();

USBFlag = 0;

}

if (MenuMainItem == 3)

{

UartFlag = 1;

UartTest();

UartFlag = 0;

TestUartDone = 0;

}

if (MenuMainItem == 4)

{

FullTestFlag = 1;

FullTest();

FullTestFlag = 0;

FullTestCANFlag = 0;

FullTestUSBInfoFlag = 0;

}

}

//Вверх

else if (!PORT\_ReadInputDataBit(MDR\_PORTB,PORT\_Pin\_5))

{

while ( ! PORT\_ReadInputDataBit(MDR\_PORTB,PORT\_Pin\_5) ) {};

if (MenuMainItem != 0) MenuMainItem--;

if (CursorPosItem != 0) CursorPosItem--;

}

//Вниз

else if (!PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_1))

{

while ( ! PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_1) ) {};

if (MenuMainItem != 4) MenuMainItem++;

if (CursorPosItem != 2) CursorPosItem++;

}

}

}

1986BE9x\_it.c:

/\* Includes ------------------------------------------------------------------\*/

#include "MDR32F9Qx\_can.h" // Keil::Drivers:CAN

#include "1986BE9x\_it.h"

#include "mlt\_lcd.h"

#include "font.h"

/\* Private typedef -----------------------------------------------------------\*/

/\* Private define ------------------------------------------------------------\*/

/\* Private macro -------------------------------------------------------------\*/

/\* Private variables ---------------------------------------------------------\*/

int extern count\_title;

int extern MenuMainItem;

int extern CursorPosItem;

char extern UartFlag;

char extern TestCanDone;

char extern CanFlag;

char extern TestUartDone;

char extern USBFlag;

char extern SSPFlag;

char extern TestSSPDone;

char extern FullTestFlag;

char extern FullTestCANFlag;

char extern FullTestCANGetFlag;

char extern FullTestUARTFlag;

char extern FullTestSSPFlag;

char extern FullTestSSPGetFlag;

uint8\_t extern USBInfo;

int count\_main = 0;

\_\_IO uint32\_t extern rx\_buf;

//Константы для отображения на дисплей

uint8\_t\* main\_string[] =

{

cyr\_K, cyr\_u, cyr\_r, cyr\_s, cyr\_o, cyr\_v, cyr\_a, cyr\_ya, sym\_sp,

cyr\_r, cyr\_a, cyr\_b, cyr\_o, cyr\_t, cyr\_a, sym\_sp, cyr\_p, cyr\_o, sym\_sp,

cyr\_k, cyr\_u, cyr\_r, cyr\_s, cyr\_u, sym\_sp, cyr\_M, cyr\_i, cyr\_k, cyr\_r, cyr\_o,

cyr\_p, cyr\_r, cyr\_o, cyr\_c, cyr\_e, cyr\_s, cyr\_s, cyr\_o, cyr\_r, cyr\_n,

cyr\_y, cyr\_e, sym\_sp, cyr\_s, cyr\_i, cyr\_s, cyr\_t, cyr\_e, cyr\_m, cyr\_y

};

uint8\_t\* menu\_string[] =

{

sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,lat\_C,lat\_A,lat\_N,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,

sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,lat\_S,lat\_S,lat\_P,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,

sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,lat\_U,lat\_S,lat\_B,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,

sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,lat\_U,lat\_A,lat\_R,lat\_T,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,

sym\_sp,sym\_sp,sym\_sp,sym\_sp,lat\_F,lat\_U,lat\_L,lat\_L,sym\_sp,lat\_T,lat\_E,lat\_S,lat\_T,sym\_sp,sym\_sp,sym\_sp

};

uint8\_t\* numbers\_string[] =

{

dig\_0,dig\_0,dig\_0,dig\_1,dig\_0,dig\_2,dig\_0,dig\_3,dig\_0,dig\_4,dig\_0,dig\_5,dig\_0,dig\_6,dig\_0,dig\_7,dig\_0,dig\_8,dig\_0,dig\_9,dig\_0,lat\_A,dig\_0,lat\_B,dig\_0,lat\_C,dig\_0,lat\_D,dig\_0,lat\_E,dig\_0,lat\_F,

dig\_1,dig\_0,dig\_1,dig\_1,dig\_1,dig\_2,dig\_1,dig\_3,dig\_1,dig\_4,dig\_1,dig\_5,dig\_1,dig\_6,dig\_1,dig\_7,dig\_1,dig\_8,dig\_1,dig\_9,dig\_1,lat\_A,dig\_1,lat\_B,dig\_1,lat\_C,dig\_1,lat\_D,dig\_1,lat\_E,dig\_1,lat\_F,

dig\_2,dig\_0,dig\_2,dig\_1,dig\_2,dig\_2,dig\_2,dig\_3,dig\_2,dig\_4,dig\_2,dig\_5,dig\_2,dig\_6,dig\_2,dig\_7,dig\_2,dig\_8,dig\_2,dig\_9,dig\_2,lat\_A,dig\_2,lat\_B,dig\_2,lat\_C,dig\_2,lat\_D,dig\_2,lat\_E,dig\_2,lat\_F,

dig\_3,dig\_0,dig\_3,dig\_1,dig\_3,dig\_2,dig\_3,dig\_3,dig\_3,dig\_4,dig\_3,dig\_5,dig\_3,dig\_6,dig\_3,dig\_7,dig\_3,dig\_8,dig\_3,dig\_9,dig\_3,lat\_A,dig\_3,lat\_B,dig\_3,lat\_C,dig\_3,lat\_D,dig\_3,lat\_E,dig\_3,lat\_F,

dig\_4,dig\_0,dig\_4,dig\_1,dig\_4,dig\_2,dig\_4,dig\_3,dig\_4,dig\_4,dig\_4,dig\_5,dig\_4,dig\_6,dig\_4,dig\_7,dig\_4,dig\_8,dig\_4,dig\_9,dig\_4,lat\_A,dig\_4,lat\_B,dig\_4,lat\_C,dig\_4,lat\_D,dig\_4,lat\_E,dig\_4,lat\_F,

dig\_5,dig\_0,dig\_5,dig\_1,dig\_5,dig\_2,dig\_5,dig\_3,dig\_5,dig\_4,dig\_5,dig\_5,dig\_5,dig\_6,dig\_5,dig\_7,dig\_5,dig\_8,dig\_5,dig\_9,dig\_5,lat\_A,dig\_5,lat\_B,dig\_5,lat\_C,dig\_5,lat\_D,dig\_5,lat\_E,dig\_5,lat\_F,

dig\_6,dig\_0,dig\_6,dig\_1,dig\_6,dig\_2,dig\_6,dig\_3,dig\_6,dig\_4,dig\_6,dig\_5,dig\_6,dig\_6,dig\_6,dig\_7,dig\_6,dig\_8,dig\_6,dig\_9,dig\_6,lat\_A,dig\_6,lat\_B,dig\_6,lat\_C,dig\_6,lat\_D,dig\_6,lat\_E,dig\_6,lat\_F,

dig\_7,dig\_0,dig\_7,dig\_1,dig\_7,dig\_2,dig\_7,dig\_3,dig\_7,dig\_4,dig\_7,dig\_5,dig\_7,dig\_6,dig\_7,dig\_7,dig\_7,dig\_8,dig\_7,dig\_9,dig\_7,lat\_A,dig\_7,lat\_B,dig\_7,lat\_C,dig\_7,lat\_D,dig\_7,lat\_E,dig\_7,lat\_F,

dig\_8,dig\_0,dig\_8,dig\_1,dig\_8,dig\_2,dig\_8,dig\_3,dig\_8,dig\_4,dig\_8,dig\_5,dig\_8,dig\_6,dig\_8,dig\_7,dig\_8,dig\_8,dig\_8,dig\_9,dig\_8,lat\_A,dig\_8,lat\_B,dig\_8,lat\_C,dig\_8,lat\_D,dig\_8,lat\_E,dig\_8,lat\_F,

dig\_9,dig\_0,dig\_9,dig\_1,dig\_9,dig\_2,dig\_9,dig\_3,dig\_9,dig\_4,dig\_9,dig\_5,dig\_9,dig\_6,dig\_9,dig\_7,dig\_9,dig\_8,dig\_9,dig\_9,dig\_9,lat\_A,dig\_9,lat\_B,dig\_9,lat\_C,dig\_9,lat\_D,dig\_9,lat\_E,dig\_9,lat\_F,

lat\_A,dig\_0,lat\_A,dig\_1,lat\_A,dig\_2,lat\_A,dig\_3,lat\_A,dig\_4,lat\_A,dig\_5,lat\_A,dig\_6,lat\_A,dig\_7,lat\_A,dig\_8,lat\_A,dig\_9,lat\_A,lat\_A,lat\_A,lat\_B,lat\_A,lat\_C,lat\_A,lat\_D,lat\_A,lat\_E,lat\_A,lat\_F,

lat\_B,dig\_0,lat\_B,dig\_1,lat\_B,dig\_2,lat\_B,dig\_3,lat\_B,dig\_4,lat\_B,dig\_5,lat\_B,dig\_6,lat\_B,dig\_7,lat\_B,dig\_8,lat\_B,dig\_9,lat\_B,lat\_A,lat\_B,lat\_B,lat\_B,lat\_C,lat\_B,lat\_D,lat\_B,lat\_E,lat\_B,lat\_F,

lat\_C,dig\_0,lat\_C,dig\_1,lat\_C,dig\_2,lat\_C,dig\_3,lat\_C,dig\_4,lat\_C,dig\_5,lat\_C,dig\_6,lat\_C,dig\_7,lat\_C,dig\_8,lat\_C,dig\_9,lat\_C,lat\_A,lat\_C,lat\_B,lat\_C,lat\_C,lat\_C,lat\_D,lat\_C,lat\_E,lat\_C,lat\_F,

lat\_D,dig\_0,lat\_D,dig\_1,lat\_D,dig\_2,lat\_D,dig\_3,lat\_D,dig\_4,lat\_D,dig\_5,lat\_D,dig\_6,lat\_D,dig\_7,lat\_D,dig\_8,lat\_D,dig\_9,lat\_D,lat\_A,lat\_D,lat\_B,lat\_D,lat\_C,lat\_D,lat\_D,lat\_D,lat\_E,lat\_D,lat\_F,

lat\_E,dig\_0,lat\_E,dig\_1,lat\_E,dig\_2,lat\_E,dig\_3,lat\_E,dig\_4,lat\_E,dig\_5,lat\_E,dig\_6,lat\_E,dig\_7,lat\_E,dig\_8,lat\_E,dig\_9,lat\_E,lat\_A,lat\_E,lat\_B,lat\_E,lat\_C,lat\_E,lat\_D,lat\_E,lat\_E,lat\_E,lat\_F,

lat\_F,dig\_0,lat\_F,dig\_1,lat\_F,dig\_2,lat\_F,dig\_3,lat\_F,dig\_4,lat\_F,dig\_5,lat\_F,dig\_6,lat\_F,dig\_7,lat\_F,dig\_8,lat\_F,dig\_9,lat\_F,lat\_A,lat\_F,lat\_B,lat\_F,lat\_C,lat\_F,lat\_D,lat\_F,lat\_E,lat\_F,lat\_F

};

uint8\_t\* usb\_string1[] =

{

sym\_sp,sym\_sp,sym\_sp,sym\_sp,cyr\_T,cyr\_e,cyr\_s,cyr\_t,sym\_sp,lat\_U,lat\_S,lat\_B,sym\_sp,sym\_sp,sym\_sp,sym\_sp

};

uint8\_t\* usb\_string2[] =

{

sym\_sp,cyr\_p,cyr\_r,cyr\_o,cyr\_v,cyr\_o,cyr\_d,cyr\_i,cyr\_t,cyr\_s,cyr\_ya,sym\_sp,cyr\_n,cyr\_a,sym\_sp,sym\_sp

};

uint8\_t\* usb\_string3[] =

{

sym\_sp,sym\_sp,cyr\_p,cyr\_l,cyr\_a,cyr\_t,cyr\_e,sym\_sp,lat\_m,lat\_a,lat\_s,lat\_t,lat\_e,lat\_r,sym\_sp,sym\_sp

};

uint8\_t\* uart\_string[] =

{

sym\_sp,sym\_sp,sym\_sp,cyr\_T,cyr\_e,cyr\_s,cyr\_t,sym\_sp,lat\_U,lat\_A,lat\_R,lat\_T,sym\_sp,sym\_sp,sym\_sp,sym\_sp

};

uint8\_t\* ssp\_string[] =

{

sym\_sp,sym\_sp,sym\_sp,sym\_sp,cyr\_T,cyr\_e,cyr\_s,cyr\_t,sym\_sp,lat\_S,lat\_S,lat\_P,sym\_sp,sym\_sp,sym\_sp,sym\_sp

};

uint8\_t\* pass\_string[] =

{

cyr\_D,cyr\_a,cyr\_n,cyr\_n,cyr\_y,cyr\_e,sym\_sp,cyr\_p,cyr\_e,cyr\_r,cyr\_e,cyr\_d,cyr\_a,cyr\_n,cyr\_y,sym\_sp

};

uint8\_t\* get\_string[] =

{

cyr\_D,cyr\_a,cyr\_n,cyr\_n,cyr\_y,cyr\_e,sym\_sp,cyr\_p,cyr\_o,cyr\_l,cyr\_u,cyr\_ch,cyr\_e,cyr\_n,cyr\_y,sym\_sp

};

uint8\_t\* wait\_string[] =

{

cyr\_O,cyr\_zh,cyr\_i,cyr\_d,cyr\_a,cyr\_n,cyr\_i,cyr\_e,sym\_sp,cyr\_d,cyr\_a,cyr\_n,cyr\_n,cyr\_y,cyr\_kh,sym\_sp

};

uint8\_t\* fail\_string[] =

{

cyr\_D,cyr\_a,cyr\_n,cyr\_n,cyr\_y,cyr\_e,sym\_sp,cyr\_o,cyr\_sh,cyr\_i,cyr\_b,cyr\_o,cyr\_ch,cyr\_n,cyr\_y,sym\_sp

};

uint8\_t\* can\_string[] =

{

sym\_sp,sym\_sp,sym\_sp,sym\_sp,cyr\_T,cyr\_e,cyr\_s,cyr\_t,sym\_sp,lat\_C,lat\_A,lat\_N,sym\_sp,sym\_sp,sym\_sp,sym\_sp

};

uint8\_t\* empty\_string[] =

{

sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp

};

uint8\_t\* info\_string[] =

{

sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp,sym\_sp

};

uint8\_t\* InfoString(uint8\_t\* RecievedData)

{

int value = RecievedData;

info\_string[7] = numbers\_string[value \* 2];

info\_string[8] = numbers\_string[value \* 2 + 1];

return info\_string;

}

//Прерывание CAN

void CAN1\_IRQHandler(void)

{

CAN\_RxMsgTypeDef RxMessage;

CAN\_GetRawReceivedData(MDR\_CAN1, rx\_buf, &RxMessage);

if (FullTestFlag == 0)

{

if((RxMessage.Rx\_Header.ID==0x15555555) && (RxMessage.Rx\_Header.IDE==CAN\_ID\_EXT)

&& (RxMessage.Rx\_Header.DLC==4) && (RxMessage.Data[0]==0x12345678))

{

TestCanDone = 1;

}

else

{

TestCanDone = 2;

}

}

else

{

USBInfo = RxMessage.Data[0];

FullTestCANGetFlag = 1;

}

CAN\_ITClearRxTxPendingBit(MDR\_CAN1, rx\_buf, CAN\_STATUS\_RX\_READY);

}

//Прерывание по таймеру

void Timer1\_IRQHandler(void)

{

int i;

MDR\_TIMER1->STATUS = 0;

if (count\_title == 0)

{

count\_title = 1;

LcdPutChar (cyr\_SH, 0, 0); LcdPutChar (cyr\_a, 1, 0); LcdPutChar (cyr\_sh, 2, 0);

LcdPutChar (cyr\_k, 3, 0); LcdPutChar (cyr\_i, 4, 0); LcdPutChar (cyr\_n, 5, 0);

LcdPutChar (cyr\_I, 10, 0); LcdPutChar (cyr\_U, 11, 0); LcdPutChar (dig\_6, 12, 0);

LcdPutChar (sym\_def, 13, 0); LcdPutChar (dig\_7, 14, 0); LcdPutChar (dig\_3, 15, 0);

//LcdPutChar (cyr\_r, 14, 0);

}

LcdPutChar (lat\_s, 11, 1); LcdPutChar (lat\_l, 12, 1);

LcdPutChar (lat\_a, 13, 1); LcdPutChar (lat\_v, 14, 1); LcdPutChar (lat\_e, 15, 1);

//Бегущая строка

while (!LcdScrollString (main\_string,7,50,count\_main));

count\_main++;

if (count\_main == 55) count\_main = 0;

//

if (CanFlag == 1)

{

LcdPutString (can\_string, 3);

LcdPutString (empty\_string, 4);

if (TestCanDone == 1)

{

LcdPutString (get\_string, 5);

}

else if (TestCanDone == 0)

{

LcdPutString (wait\_string, 5);

}

else if (TestCanDone == 2)

{

LcdPutString (fail\_string, 5);

}

}

else if (SSPFlag == 1)

{

LcdPutString (ssp\_string, 3);

LcdPutString (empty\_string, 4);

if (TestSSPDone == 1)

{

LcdPutString (get\_string, 5);

}

else if (TestSSPDone == 0)

{

LcdPutString (wait\_string, 5);

}

else if (TestSSPDone == 2)

{

LcdPutString (fail\_string, 5);

}

}

else if (USBFlag == 1)

{

LcdPutString (usb\_string1, 3);

LcdPutString (usb\_string2, 4);

LcdPutString (usb\_string3, 5);

}

else if (UartFlag == 1)

{

LcdPutString (uart\_string, 3);

LcdPutString (empty\_string, 4);

if (TestUartDone == 1)

{

LcdPutString (get\_string, 5);

}

else

{

LcdPutString (wait\_string, 5);

}

}

else if (FullTestFlag == 1)

{

if (FullTestCANFlag == 1)

{

if (FullTestCANGetFlag == 0)

{

LcdPutString (can\_string, 3);

LcdPutString (wait\_string, 4);

LcdPutString (empty\_string, 5);

}

else

{

LcdPutString (can\_string, 3);

LcdPutString (get\_string, 4);

LcdPutString (InfoString(USBInfo), 5);

}

}

else if (FullTestSSPFlag == 1)

{

if (FullTestSSPGetFlag == 0)

{

LcdPutString (ssp\_string, 3);

LcdPutString (wait\_string, 4);

LcdPutString (empty\_string, 5);

}

else

{

LcdPutString (ssp\_string, 3);

LcdPutString (get\_string, 4);

LcdPutString (InfoString(USBInfo), 5);

}

}

else if (FullTestUARTFlag == 1)

{

LcdPutString (uart\_string, 3);

LcdPutString (pass\_string, 4);

LcdPutString (InfoString(USBInfo), 5);

}

}

else

{

if (CursorPosItem == 0)

{

for (i=0;i<16;i++) LcdPutChar(menu\_string[16\*(MenuMainItem+0)+i],i,3);

for (i=0;i<16;i++) LcdPutChar(menu\_string[16\*(MenuMainItem+1)+i],i,4);

for (i=0;i<16;i++) LcdPutChar(menu\_string[16\*(MenuMainItem+2)+i],i,5);

LcdPutChar (cursor, 15, 3);

}

if (CursorPosItem == 1)

{

for (i=0;i<16;i++) LcdPutChar(menu\_string[16\*(MenuMainItem-1)+i],i,3);

for (i=0;i<16;i++) LcdPutChar(menu\_string[16\*(MenuMainItem+0)+i],i,4);

for (i=0;i<16;i++) LcdPutChar(menu\_string[16\*(MenuMainItem+1)+i],i,5);

LcdPutChar (cursor, 15, 4);

}

if (CursorPosItem == 2)

{

for (i=0;i<16;i++) LcdPutChar(menu\_string[16\*(MenuMainItem-2)+i],i,3);

for (i=0;i<16;i++) LcdPutChar(menu\_string[16\*(MenuMainItem-1)+i],i,4);

for (i=0;i<16;i++) LcdPutChar(menu\_string[16\*(MenuMainItem+0)+i],i,5);

LcdPutChar (cursor, 15, 5);

}

}

}

CAN\_main.c

#include "MDR32Fx.h"

#include "MDR32F9Qx\_config.h"

#include "MDR32F9Qx\_can.h"

#include "MDR32F9Qx\_rst\_clk.h"

#include "MDR32F9Qx\_port.h"

typedef enum {FAILED = 0, PASSED = !FAILED} TestStatus;

#define LED1 PORT\_Pin\_0

#define LED2 PORT\_Pin\_1

\_\_IO uint32\_t ret = 0;

\_\_IO uint32\_t extern rx\_buf;

\_\_IO uint32\_t extern tx\_buf;

volatile TestStatus TestRx;

PORT\_InitTypeDef PORT\_InitStructure;

//Инциализация портов CAN

void CAN\_ports\_ini(void)

{

PORT\_InitTypeDef PortInit;

PortInit.PORT\_PULL\_UP = PORT\_PULL\_UP\_OFF;

PortInit.PORT\_PULL\_DOWN = PORT\_PULL\_DOWN\_OFF;

PortInit.PORT\_PD\_SHM = PORT\_PD\_SHM\_OFF;

PortInit.PORT\_PD = PORT\_PD\_DRIVER;

PortInit.PORT\_GFEN = PORT\_GFEN\_OFF;

PortInit.PORT\_FUNC = PORT\_FUNC\_ALTER;

PortInit.PORT\_SPEED = PORT\_SPEED\_MAXFAST;

PortInit.PORT\_MODE = PORT\_MODE\_DIGITAL;

PortInit.PORT\_OE = PORT\_OE\_IN;

PortInit.PORT\_Pin = PORT\_Pin\_7;

PORT\_Init(MDR\_PORTA, &PortInit);

PortInit.PORT\_OE = PORT\_OE\_OUT;

PortInit.PORT\_Pin = PORT\_Pin\_6;

PORT\_Init(MDR\_PORTA, &PortInit);

}

//Инициация CAN

void CAN\_Setup(void)

{

CAN\_InitTypeDef sCAN;

RST\_CLK\_CPU\_PLLconfig (RST\_CLK\_CPU\_PLLsrcHSIdiv2,0);

uint32\_t i = 0;

RST\_CLK\_PCLKcmd((RST\_CLK\_PCLK\_RST\_CLK | RST\_CLK\_PCLK\_CAN1),ENABLE);

CAN\_BRGInit(MDR\_CAN1,CAN\_HCLKdiv1);

CAN\_DeInit(MDR\_CAN1);

CAN\_StructInit (&sCAN);

sCAN.CAN\_ROP = ENABLE;

sCAN.CAN\_SAP = ENABLE;

sCAN.CAN\_STM = DISABLE;

sCAN.CAN\_ROM = DISABLE;

sCAN.CAN\_PSEG = CAN\_PSEG\_Mul\_2TQ;

sCAN.CAN\_SEG1 = CAN\_SEG1\_Mul\_3TQ;

sCAN.CAN\_SEG2 = CAN\_SEG2\_Mul\_2TQ;

sCAN.CAN\_SJW = CAN\_SJW\_Mul\_2TQ;

sCAN.CAN\_SB = CAN\_SB\_3\_SAMPLE;

sCAN.CAN\_BRP = 1;

CAN\_Init (MDR\_CAN1,&sCAN);

CAN\_Cmd(MDR\_CAN1, ENABLE);

NVIC\_EnableIRQ(CAN1\_IRQn);

CAN\_ITConfig( MDR\_CAN1, CAN\_IT\_GLBINTEN | CAN\_IT\_RXINTEN, ENABLE);

CAN\_RxITConfig( MDR\_CAN1 ,(1<<rx\_buf), ENABLE);

CAN\_TxITConfig( MDR\_CAN1 ,(1<<tx\_buf), ENABLE);

CAN\_Receive(MDR\_CAN1, rx\_buf, DISABLE);

}

SSP\_main.c

//#include "MDR32F9Qx\_board.h"

#include "MDR32F9Qx\_config.h"

#include "MDR32Fx.h"

#include "MDR32F9Qx\_ssp.h"

#include "MDR32F9Qx\_rst\_clk.h"

#include "MDR32F9Qx\_port.h"

SSP\_InitTypeDef sSSP;

PORT\_InitTypeDef PORT\_InitStructur;

uint16\_t SrcBuf1;

uint16\_t SrcBuf2;

uint8\_t TxIdx = 0, RxIdx = 0;

//Настройка портов SSP

void PortsSSP2Init(void) {

PORT\_InitStructur.PORT\_Pin = (PORT\_Pin\_2 | PORT\_Pin\_3 | PORT\_Pin\_5);

PORT\_InitStructur.PORT\_OE = PORT\_OE\_IN;

PORT\_InitStructur.PORT\_FUNC = PORT\_FUNC\_ALTER;

PORT\_InitStructur.PORT\_MODE = PORT\_MODE\_DIGITAL;

PORT\_InitStructur.PORT\_SPEED = PORT\_SPEED\_FAST;

PORT\_Init(MDR\_PORTD, &PORT\_InitStructur);

PORT\_InitStructur.PORT\_OE = PORT\_OE\_OUT;

PORT\_InitStructur.PORT\_Pin = (PORT\_Pin\_6);

PORT\_Init(MDR\_PORTD, &PORT\_InitStructur);

}

//Настройка структуры SSP

void SSP2StructInit(void) {

SSP\_DeInit(MDR\_SSP2);

RST\_CLK\_PCLKcmd((RST\_CLK\_PCLK\_RST\_CLK | RST\_CLK\_PCLK\_SSP1 | RST\_CLK\_PCLK\_SSP2),ENABLE);

SSP\_BRGInit(MDR\_SSP2,SSP\_HCLKdiv16);

SSP\_StructInit (&sSSP);

sSSP.SSP\_SCR = 0x10;

sSSP.SSP\_CPSDVSR = 12;

sSSP.SSP\_Mode = SSP\_ModeSlave;

sSSP.SSP\_WordLength = SSP\_WordLength16b;

sSSP.SSP\_SPH = SSP\_SPH\_1Edge;

sSSP.SSP\_SPO = SSP\_SPO\_Low;

sSSP.SSP\_FRF = SSP\_FRF\_SPI\_Motorola;

sSSP.SSP\_HardwareFlowControl = SSP\_HardwareFlowControl\_SSE;

SSP\_Init (MDR\_SSP2,&sSSP);

SSP\_Cmd(MDR\_SSP2, ENABLE);

}

UART\_main.c:

//#include "MDR32F9Qx\_board.h"

#include "MDR32F9Qx\_config.h"

#include "MDR32Fx.h"

#include "MDR32F9Qx\_uart.h"

#include "MDR32F9Qx\_port.h"

#include "MDR32F9Qx\_rst\_clk.h"

static PORT\_InitTypeDef PortInit;

static UART\_InitTypeDef UART\_InitStructure;

// Настройка портов для UART

void Uart2PinCfg(void)

{

PortInit.PORT\_PULL\_UP = PORT\_PULL\_UP\_OFF;

PortInit.PORT\_PULL\_DOWN = PORT\_PULL\_DOWN\_OFF;

PortInit.PORT\_PD\_SHM = PORT\_PD\_SHM\_OFF;

PortInit.PORT\_PD = PORT\_PD\_DRIVER;

PortInit.PORT\_GFEN = PORT\_GFEN\_OFF;

PortInit.PORT\_FUNC = PORT\_FUNC\_OVERRID;

PortInit.PORT\_SPEED = PORT\_SPEED\_MAXFAST;

PortInit.PORT\_MODE = PORT\_MODE\_DIGITAL;

PortInit.PORT\_OE = PORT\_OE\_IN;

PortInit.PORT\_Pin = PORT\_Pin\_0;

PORT\_Init(MDR\_PORTF, &PortInit);

PortInit.PORT\_OE = PORT\_OE\_OUT;

PortInit.PORT\_Pin = PORT\_Pin\_1;

PORT\_Init(MDR\_PORTF, &PortInit);

}

//Настройка структуры UART

void Uart2Setup(void)

{

RST\_CLK\_CPU\_PLLconfig (RST\_CLK\_CPU\_PLLsrcHSIdiv2,0);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_UART2, ENABLE);

UART\_BRGInit(MDR\_UART2, UART\_HCLKdiv1);

UART\_InitStructure.UART\_BaudRate = 9600;

UART\_InitStructure.UART\_WordLength = UART\_WordLength8b;

UART\_InitStructure.UART\_StopBits = UART\_StopBits1;

UART\_InitStructure.UART\_Parity = UART\_Parity\_No;

UART\_InitStructure.UART\_FIFOMode = UART\_FIFO\_OFF;

UART\_InitStructure.UART\_HardwareFlowControl = UART\_HardwareFlowControl\_RXE | UART\_HardwareFlowControl\_TXE;

UART\_Init (MDR\_UART2,&UART\_InitStructure);

UART\_Cmd(MDR\_UART2,ENABLE);

}

USB\_main.c:

#include "MDR32Fx.h"

#include "MDR32F9Qx\_usb\_handlers.h"

#include "MDR32F9Qx\_rst\_clk.h"

#define BUFFER\_LENGTH 100

USB\_Clock\_TypeDef USB\_Clock\_InitStruct;

USB\_DeviceBUSParam\_TypeDef USB\_DeviceBUSParam;

static uint8\_t Buffer[BUFFER\_LENGTH];

char extern FullTestFlag;

char extern FullTestUSBInfoFlag;

char extern FullTestUSBSendInfoFlag;

uint8\_t extern USBInfo;

uint8\_t extern USBFinalInfo;

uint8\_t\* Buf = 0x00000000;

static TDebugInfo SetupPackets[USB\_DEBUG\_NUM\_PACKETS];

static uint32\_t SPIndex;

static uint32\_t ReceivedByteCount, SentByteCount, SkippedByteCount;

#define DELAY(T) for (int i = T; i > 0; i--)

static void Setup\_USB(void);

static void VCom\_Configuration(void);

//Запуск USB

int USB\_main(void)

{

VCom\_Configuration();

USB\_CDC\_Init(Buffer, 1, SET);

Setup\_CPU\_Clock();

Setup\_USB();

/\* Main loop \*/

return 0;

}

//Отключение USB, возврат нормлаьного тактирования

void USB\_off(void)

{

USB\_DevicePowerOff();

RST\_CLK\_LSEconfig(RST\_CLK\_LSE\_ON);

while (RST\_CLK\_LSEstatus() != SUCCESS);

RST\_CLK\_HSEconfig(RST\_CLK\_HSE\_ON);

while (RST\_CLK\_HSEstatus() != SUCCESS);

RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv2, RST\_CLK\_CPU\_PLLmul1);

RST\_CLK\_CPU\_PLLcmd(ENABLE);

if (RST\_CLK\_CPU\_PLLstatus() != SUCCESS)

{

/\* Trap \*/

while (1)

{

}

}

RST\_CLK\_CPUclkPrescaler(RST\_CLK\_CPUclkDIV1);

RST\_CLK\_CPU\_PLLuse(ENABLE);

RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTA, ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTB, ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTC, ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTD, ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTE, ENABLE);

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTF, ENABLE);

}

void Setup\_CPU\_Clock(void)

{

RST\_CLK\_CPU\_PLLconfig(RST\_CLK\_CPU\_PLLsrcHSEdiv1, RST\_CLK\_CPU\_PLLmul10);

RST\_CLK\_CPU\_PLLcmd(ENABLE);

if (RST\_CLK\_CPU\_PLLstatus() != SUCCESS)

{

/\* Trap \*/

while (1)

{

}

}

RST\_CLK\_CPUclkPrescaler(RST\_CLK\_CPUclkDIV1);

RST\_CLK\_CPU\_PLLuse(ENABLE);

RST\_CLK\_CPUclkSelection(RST\_CLK\_CPUclkCPU\_C3);

}

//Включение и настройка USB

void Setup\_USB(void)

{

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_USB, ENABLE);

USB\_Clock\_InitStruct.USB\_USBC1\_Source = USB\_C1HSEdiv2;

USB\_Clock\_InitStruct.USB\_PLLUSBMUL = USB\_PLLUSBMUL12;

USB\_DeviceBUSParam.MODE = USB\_SC\_SCFSP\_Full;

USB\_DeviceBUSParam.SPEED = USB\_SC\_SCFSR\_12Mb;

USB\_DeviceBUSParam.PULL = USB\_HSCR\_DP\_PULLUP\_Set;

USB\_DeviceInit(&USB\_Clock\_InitStruct, &USB\_DeviceBUSParam);

//Разрешение прерываний

USB\_SetSIM(USB\_SIS\_Msk);

USB\_DevicePowerOn();

USB\_DEVICE\_HANDLE\_RESET;

}

//Отправка данных

USB\_Result SendInfo(uint8\_t\* Data)

{

USB\_Result result;

Buf[0] = Data;

USB\_CDC\_SendData(Buf,0x00000001);

return result;

}

//Прием данных

USB\_Result USB\_CDC\_RecieveData(uint8\_t\* Buffer, uint32\_t Length)

{

USB\_Result result;

if (FullTestFlag == 1)

{

if (FullTestUSBSendInfoFlag == 0)

{

USBInfo = Buffer[0];

FullTestUSBInfoFlag = 1;

}

else

{

Buffer[0] = USBFinalInfo;

result = USB\_CDC\_SendData(Buffer, Length);

FullTestUSBInfoFlag = 1;

}

}

else

{

/\* Send back received data portion \*/

result = USB\_CDC\_SendData(Buffer, Length);

}

//Проверка отправки данных

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

// USB\_CDC\_HANDLE\_GET\_LINE\_CODING пример

USB\_Result USB\_CDC\_GetLineCoding(uint16\_t wINDEX, USB\_CDC\_LineCoding\_TypeDef\* DATA)

{

assert\_param(DATA);

if (wINDEX != 0)

{

//Неверный интерфейс

return USB\_ERR\_INV\_REQ;

}

//Сохранить полученные значения

\*DATA = LineCoding;

return USB\_SUCCESS;

}

//USB\_CDC\_HANDLE\_SET\_LINE\_CODING пример

USB\_Result USB\_CDC\_SetLineCoding(uint16\_t wINDEX, const USB\_CDC\_LineCoding\_TypeDef\* DATA)

{

assert\_param(DATA);

if (wINDEX != 0)

{

return USB\_ERR\_INV\_REQ;

}

// Отправка полученных ранее значений

LineCoding = \*DATA;

return USB\_SUCCESS;

}

Исходный код драйвера:

; MDRVComPort.inf - Virtual COM Port to USB

; Driver for Windows NT family

;

; Copyright 2011 Milandr

;

[Version]

Signature="$Windows NT$"

Class=Ports

ClassGUID={4d36e978-e325-11ce-bfc1-08002be10318}

Provider=%MDR%

DriverVer=01/06/2011,1.0.0

[DestinationDirs]

DefaultDestDir=12

[Manufacturer]

%MDR%=Milandr

[Milandr]

%MDR\_CDC&VID\_0483&PID\_F125.DeviceDesc%=MDRVCom,USB\VID\_0483&PID\_F125

[MDRVCom.NT]

CopyFiles=MDRVCom.NT.Copy

AddReg=MDRVCom.NT.AddReg

[MDRVCom.NT.Copy]

usbser.sys

[MDRVCom.NT.AddReg]

HKR,,DevLoader,,\*ntkern

HKR,,NTMPDriver,,usbser.sys

HKR,,EnumPropPages32,,"MsPorts.dll,SerialPortPropPageProvider"

[MDRVCom.NT.Services]

AddService=usbser, 0x00000002, Service\_Inst

[Service\_Inst]

DisplayName=%MDRVCom.SvcDesc%

ServiceType=1

StartType=3

ErrorControl=1

ServiceBinary=%12%\usbser.sys

LoadOrderGroup=Base

[Strings]

MDR="Milandr"

MDR\_CDC&VID\_0483&PID\_F125.DeviceDesc="Milandr Virtual COM port"

MDRVCom.SvcDesc="Milandr Virtual COM port"