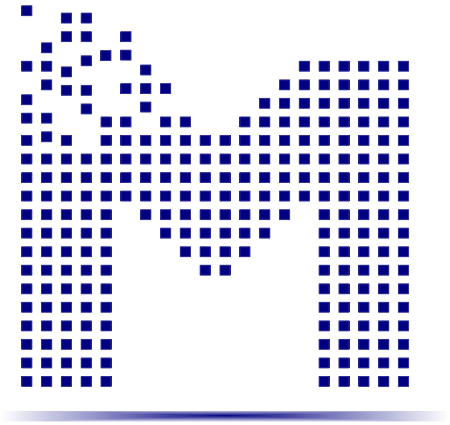


НПФ «Мехатроника-Про»

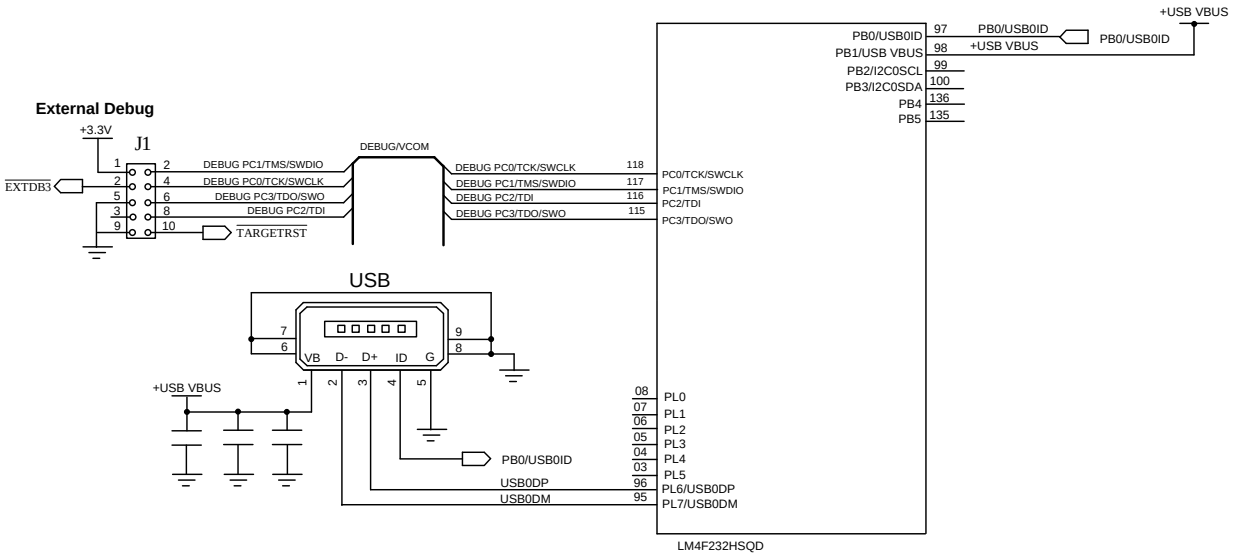
Описание Стартового проекта LM4F23x в среде Code Composer Studio 5.xx

Содержание

Описание стартового проекта

Проект «**StartUp**» предназначен для конфигурации периферии микроконтроллера и запуска основных задач. Данный проект содержит следующие основные модули:

- «**main.c**» – главный модуль, содержащий секцию инициализации, фоновую задачу и обработчики прерываний;
- «**startup.c**» – модуль, содержащий таблицу векторов прерываний и базовые функции их обработки;
- «**usb_dev.c**» – модуль реализации коммуникации по USB;
- «**usb_structs.c**» – модуль структур дескрипторов USB устройства;
- «**LM4F232H5QD.cmd**» – командный файл распределения адресного пространства;
- «**target_config.ccxml**» – конфигурационный файл, содержащий тип устройства и средство отладки
- «**usblib-cm4f.lib**» – библиотека функций для коммуникации по USB (из пакета StellarisWare);
- «**driverlib-cm4f.lib**» – библиотека функций для конфигурации и работы с периферией процессора (из пакета StellarisWare).



Основной модуль «main» содержит следующие функции:

- «**main**» – базовая функция, управление на которую передается системой после секции старта кода.

  В данной функции осуществляется вызов функций: конфигурации оборудования, инициализация коммуникации, при наличии драйвера памяти функции – чтение конфигурации операционной среды MexBIOS, функции задания конфигурации ядра ОС (**MBS_Create**) и функции разрешения работы прерываний.
  Кроме того, осуществляется запуск фоновой задачи (бесконечного цикла), в которой осуществляется вызов функций установки конфигурации и запуска фоновых задач MexBIOS (**MBS_Init**, **MBS_TaskExecute**).

- «**SysTickIntHandler**» – обработчик прерывания для формирования основной частоты расчета (периода дискретизации системы).

В данной функции осуществляется вызов функций: запуск периодических задач ОС (**MBS_IsrExecute**), записи конфигурации в энергонезависимую память при наличии команды (**WriteMexBiosConfig**), а также выполняется расчет загрузки системы и передача информации в ядро ОС (через **MbsCpuLoad**).

В секции инициализации выполняются следующие действия:

- Разрешается работы модуля FPU (для работы с floating-point)

```
36      // The FPU is enabled
37      FPUEnable();
38      FPULazyStackingEnable();
```

- Выбирается источник тактовой частоты (внешний кварц) и конфигурация для работы на частоте 80 МГц

```
40      // Set the clocking to run directly from the crystal
41      ROM_SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL |
42                         SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);
```

- Конфигурация таймера для основного прерывания, определяющего частоту дискретизации (HZ = 10000 Гц)

```
53      // Enable the system tick.
54      g_uiSysTickPeriod = ROM_SysCtlClockGet() / HZ;
55      ROM_SysTickPeriodSet(g_uiSysTickPeriod);
56      ROM_SysTickIntEnable();
57      ROM_SysTickEnable();
```

- Глобально разрешаются прерывания

```
80      // Enable interrupts to the processor.
81      ROM_IntMasterEnable();
```

Подключение ядра MexBIOS в проект пользователя

Ядро MexBIOS может быть подключено в проект, созданный пользователем. Для этого необходимо выполнить действия, представленные в данном разделе.

Подключение драйвера коммуникации

Драйвер «**usblib**» предназначен для загрузки конфигурации MexBIOS в микроконтроллер. Для его подключения необходимо выполнить следующие этапы:

1) Подключение библиотеки драйвера к проекту

«Project» → «Properties» → «C/C++ Build» → «Tool Settings» → «TMS470 Linker» → «File Search Path» → добавить библиотеку «usblib-cm4f.lib» и прописать путь для неё «"..\..\..\..\Generic\StellarisWare\lib»

2) Подключить в модуле «main.c» заголовочные файлы драйвера

```
16 #include "usblib.h"
17 #include "usbdbulk.h"
18 #include "usb_structs.h"
```

3) Объявить прототип функции обработки прерывания в модуле конфигурации «**startup.c**»

```
27 //*****************************************************************************
28 //
29 // External declarations for the interrupt handlers used by the application.
30 //
31 //*****************************************************************************
32 extern void SysTickIntHandler(void);
33 extern void USB0DeviceIntHandler(void);
```

4) Добавить таблицу векторов прерываний

```
42 #pragma DATA_SECTION(g_pfnVectors, ".intvecs")
43 void (* const g_pfnVectors[])(void) =
44 {
45     (void (*)(void))((unsigned long)&__STACK_TOP),
   ...
105     IntDefaultHandler,                      // Hibernate
106     USB0DeviceIntHandler,                   // USB0
107     IntDefaultHandler,                      // PWM Generator 3
```

5) В модуле «**main.c**» задать конфигурацию ножек для работы с USB (PB0, PB1, PL6 и PL7)

```
47     // Enable the GPIO peripheral used for USB, and configure the USB pins.
48     ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
49     ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOL);
50     ROM_GPIOPinTypeUSBAnalog(GPIO_PORTB_BASE, GPIO_PIN_0 | GPIO_PIN_1);
51     ROM_GPIOPinTypeUSBAnalog(GPIO_PORTL_BASE, GPIO_PIN_6 | GPIO_PIN_7);
```

6) Инициализировать буферы **USB** протокола

```
59      // Initialize the transmit and receive buffers.
60      USBBufferInit((tUSBBuffer *)&g_sTxBuffer);
61      USBBufferInit((tUSBBuffer *)&g_sRxBuffer);
```

7) Вызвать функцию инициализации устройства

```
63      // Pass our device information to the USB library and
64      // place the device on the bus.
65       USBDBulkInit(0, (tUSBDBulkDevice *)&g_sBulkDevice);
```

8) В модуле «**usb_dev.c**» объявить обработчик прерываний по приему данных

```
150 // Handles bulk driver notifications related to the receive channel (data from
151 // the USB host).
152 unsigned long RxHandler(void *pvCBData, unsigned long ulEvent,
153     unsigned long ulMsgValue, void *pvMsgData)
```

В данной функции по событию USB_EVENT_RX_AVAILABLE (наличие входного пакета данных) осуществлен вызов функции обработки принятого пакета «**UpdateData**».

9) Объявить обработчик прерывания при передаче

```
199 // Handles bulk driver notifications related to the transmit channel (data to
200 // the USB host).
201 unsigned long TxHandler(void *pvCBData, unsigned long ulEvent,
202     unsigned long ulMsgValue, void *pvMsgData)
```

10) Объявить функцию обработки принятого пакета «**UpdateData**»

```
044 // Update receiving data and send back new to the host.
045 static unsigned long UpdateData(tUSBDBulkDevice *psDevice, unsigned char *pcData,
046 unsigned long ulNumBytes)
```

В данной функции обрабатывается принятый буфер с устройства USB-Host. Протокол данных MexBIOS инкапсулируется в USB протокол, а буфер данных содержит следующие поля:

- «**ucFunction**» (1 байт) – Код функции. Требуемая операция (USB_FUNC_READ=1 – чтение данных, USB_FUNC_WRITE=2 – запись данных);
- «**usDataAddr**» (2 байта) – Начальный адрес. Это значение, определяющее смещение в базе данных параметров устройства;
- «**usDataCount**» (2 байта) – Количество данных в словах;
- «**usExtData**» – указатель на массив данных, получаемый в результате обработки заданной функции.

В случае успеха формируется ответный кадр данных, иначе формируется кадр, содержащий поля: код функции и значение ошибки «**ucException**» (значения ошибок смотри в объявлениях «**Exception codes**»).

Кадр данных записывается для дальнейшей передачи в выходной буфер «**g_sTxBuffer**» с помощью функции «**USBBufferDataWritten**».

Подключение ядра MexBIOS

Для подключения необходимо выполнить следующие этапы:

1) Подключить в модуле «**main.c**» заголовочный файл

```
19 #include "MBS_Import.h"
```

Примечание: Данной файл автоматически генерируется в папке библиотеки блоков «\NPF Mechatronica-pro\MexBIOS Development Studio\Extend\LM4F23x»

2) В функции «**UpdateData**» модуля «**usb_dev.c**» вызвать функцию получения указателя на буфер данных

```
088     usExtData = (unsigned short *)MBS_GetData(usDataAddr, usDataCount,
        (short*)&usMode);
```

В данную функцию необходимо передать адрес на локальную переменную «**usMode**». Данный параметр определяет тип указателя данных:

- «usMode=0» – значение переменной «usExtData» содержит указатель на массив (буфер) данных. Доступ к параметрам осуществляется с помощью инструкции:

```
usTemp = *(usExtData + ucIndex);
```

- «usMode=1» – значение переменной «usExtData» содержит указатель на массив (буфер) указателей глобальных переменных ядра ОС. Доступ к параметрам осуществляется с помощью инструкций:

```
long **Gdata = (long **)usExtData;
usTemp = *((short *)(*(Gdata + usIndex)));      - для младшего слова
usTemp = *((short *)(*(Gdata + usIndex)) + 1); - для старшего слова
```

3) Выполнить операцию чтения конфигурации из энергонезависимой памяти в модуле «main.c»

```
66  // If memory drive enable execute function for
67  // reading MexBIOS configuration
68  #if 0
69  ReadMexBiosConfig(Address, MbsMtxAddr, MbsMtxSize);
70  #endif
```

Примечание: Здесь приведен пример вызова такой функции. Параметры этой функции «**MbsMtxAddr**» и «**MbsMtxSize**» определены в заголовочном файле «**MBS_Import.h**». Параметр «**Address**» является начальным адресом конфигурации MexBIOS в энергонезависимой памяти и может быть назначен по своему усмотрению.

4) Вызвать в секции инициализации функцию «**MBS_Create**» и передать параметры в ядро MexBIOS. После этого подать команду на запуск **(*MbsEnable=1).**

```
72  // If MexBIOS enable execute function for
73  // setting it base parameters
74  #ifdef MBS_INCLUDE
75  MBS_Create();
76  *MbsAppVersion = VERSION;   // set project version to dislay in the
studio
77  *MbsEnable = 1;             // command to startup
78  #endif
```

5) В фоновой задаче вызвать функции задании конфигурации и исполнения фоновых задач

```
83      // Loop forever
84      while(1)
85      {
86       // If MexBIOS enable execute functions for
87       // setting configuration and execute tasks
88       #ifdef MBS_INCLUDE
89       MBS_Init();
90       MBS_TaskExecute();
91       #endif
92      }
```

6) В обработчике основного прерывания «**SysTickHandler**» вызвать обработчик периодических задач ядра

```
105     // Execute function with interrupt source
106     MBS_IsrExecute();
```

7) По команде выполнить операцию записи конфигурации в энергонезависимую память

```
108     // If memory driver enable check write command
109     #if 0
110     if (*MbsWriteCmd)
111     {
112         // Write MexBIOS configuration in memory
113         WriteMexBiosConfig(Address, MbsMtxAddr, MbsMtxSize);
114         *MbsWriteCmd = 0;
115     }
116     #endif
```

8) Выполнить расчет загрузки системы и передать значение в ядро для отображения в среде разработки

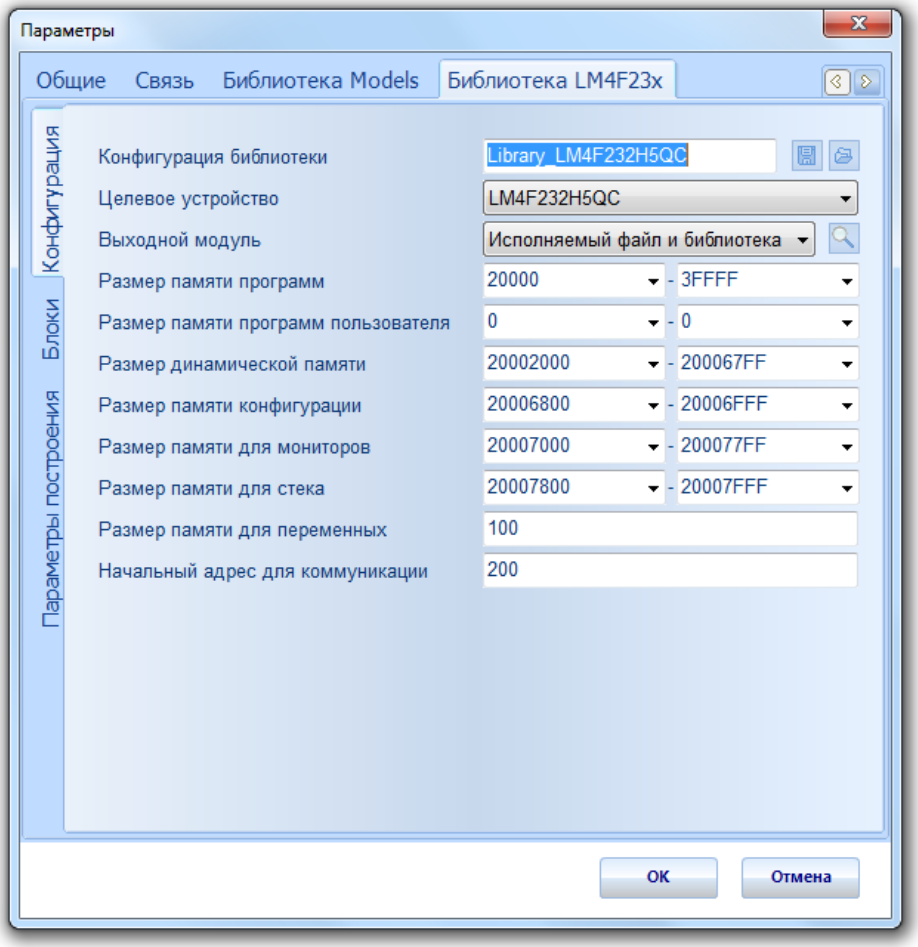
```
099     g_uiSysTickTimer = ROM_SysTickValueGet();
  ...
118     // CPU load calculation
119     g_uiSysTickTimer = g_uiSysTickTimer - ROM_SysTickValueGet();
120     if ((int)g_uiSysTickTimer < 0) g_uiSysTickTimer += g_uiSysTickPeriod;
121     *MbsCpuLoad = (long)(100.0*g_uiSysTickTimer/g_uiSysTickPeriod);
```

9) Выделить адресное пространство памяти программ и данных в командном файле для ядра MexBIOS

```
MEMORY
{
    /* Application stored in and executes from internal flash */
    FLASH (RX) : origin = APP_BASE,   length = 0x00020000
/*  FLASH (RX) : origin = 0x00020000, length = 0x00020000 Reserved for MexBIOS */

    /* Application uses internal RAM for data */
    VECS (RWX) : origin = RAM_BASE,            length = 0x0000026C
    SRAM (RWX) : origin = RAM_BASE+0x0000026C, length = 0x00002000-0x0000026C
/*  SRAM (RWX) : origin = 0x20002000, length = 0x00006000 Reserved for MexBIOS */
}
```

Данные настройки необходимо выставлять в соответствии с настройками, выставленными в окне «**Параметры**» на вкладке «**Библиотека LM4F232x**».



Распределение Flash-памяти:

- для библиотеки и ядра MexBIOS – 0x20000 ÷ 0x3FFFF
- для стартового проекта – 0x00000 ÷ 0x1FFFF

Распределение RAM-памяти:

- для библиотеки и ядра MexBIOS – 0x20002000 ÷ 0x20007FFF
- для стартового проекта – 0x20000000 ÷ 0x20001FFF

Настройка проекта в MexBIOS Development Studio

Для работы с платой необходимо загрузить стартовый проект и библиотеку блоков в память процессора.

Также в случае изменения состава библиотеки (добавление или исключение блоков, добавление новых блоков, редактирования существующих) необходимо произвести компиляцию библиотеки и стартового проекта.

В данном разделе показаны необходимые настройки для компиляции библиотеки блоков и стартового проекте. Если настройки выполнены верно, то **MexBIOS Development Studio** после нажатия кнопки **Построить** проведёт компиляцию стартового проекта и библиотеки блоков.

В случае успешной компиляции необходимо загрузить стартовый проект и библиотеку блоков по приведённой ниже инструкции в память контроллера, после чего можно приступать к составлению схем в MexBIOS Development Studio.

Настройка подключения компилятора

Для микропроцессоров серии **LM4F23x Texas Instruments** не предлагает компилятор в свободном доступе как **Code Generation Tools C2000**.

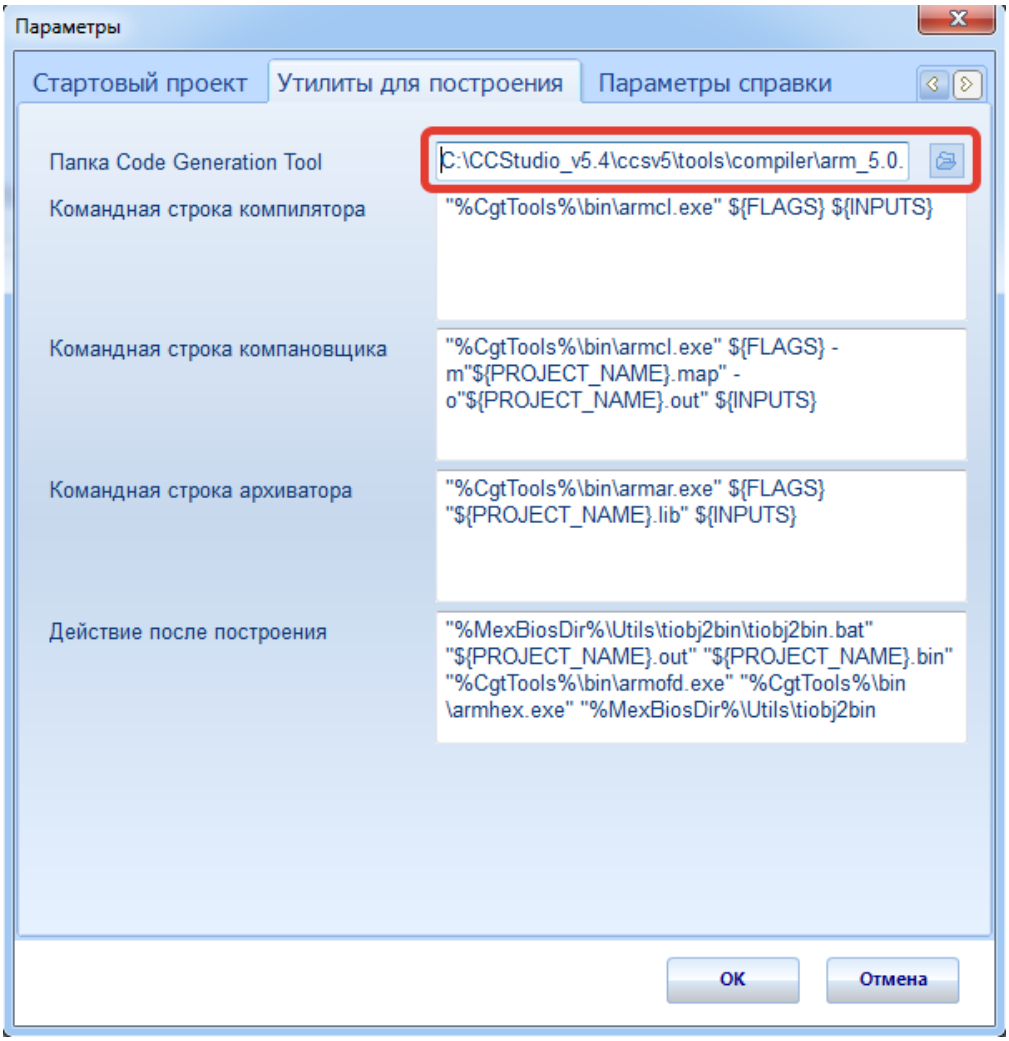
Если имеется отладочный комплект фирмы TI с диском CCS, то компилятор можно найти на диске по следующему пути:

E:\Tools\CCS\install_images\cgt

файл **ti_cgt_tms470_4.6.3_setup_win32.exe.**

Если нет такового, то необходимо скачать версию CCS для работы с процессорами серии LM4F23x и указать в Options путь к компилятору:

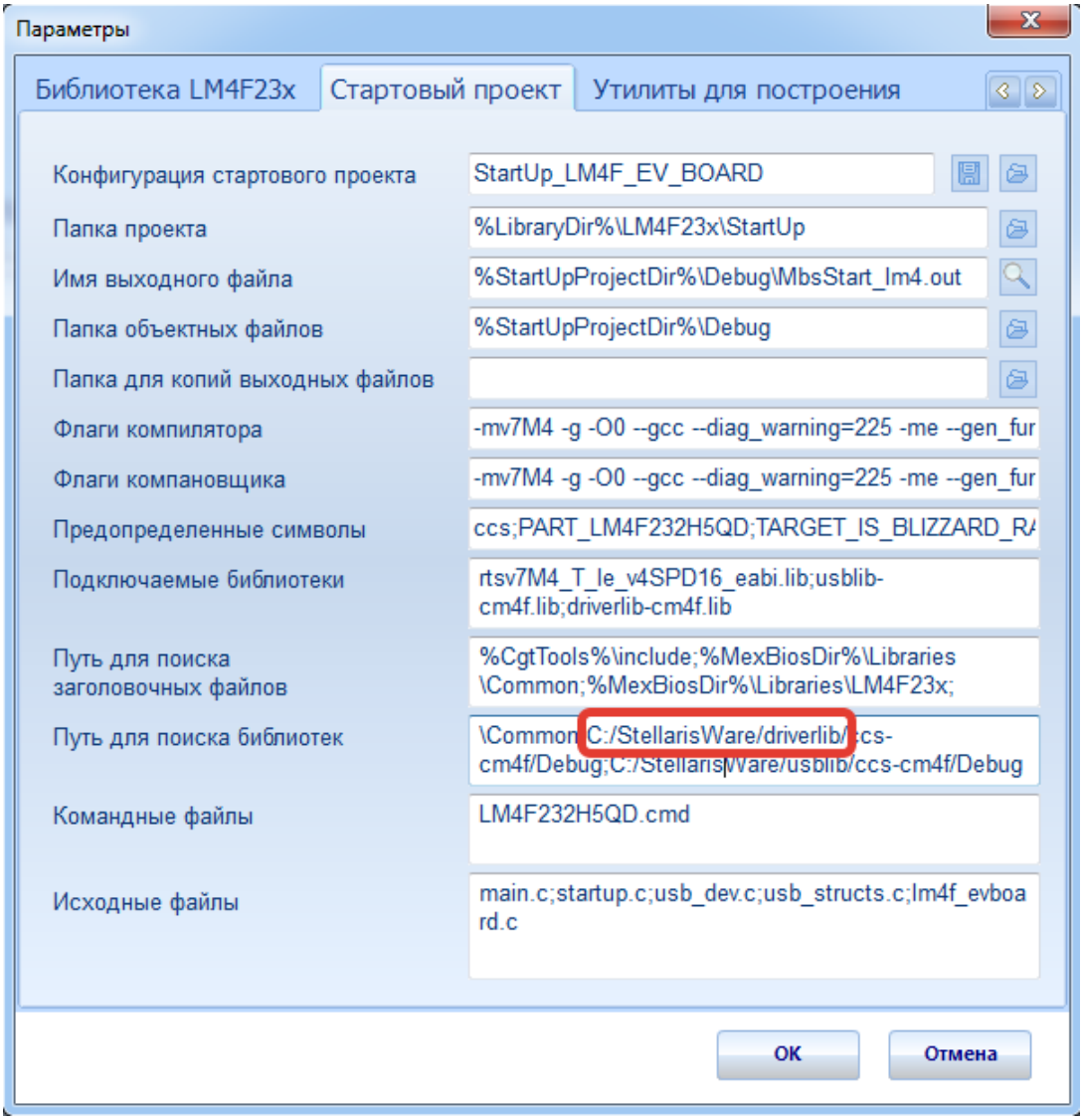
C:\CCStudio_v5.4\ccsv5\tools\compiler\arm_5.0.4



Для установки компилятора необходимо загрузить пробную версию CCS 5.x.

Компиляция библиотеки и стартового проекта в MexBIOS

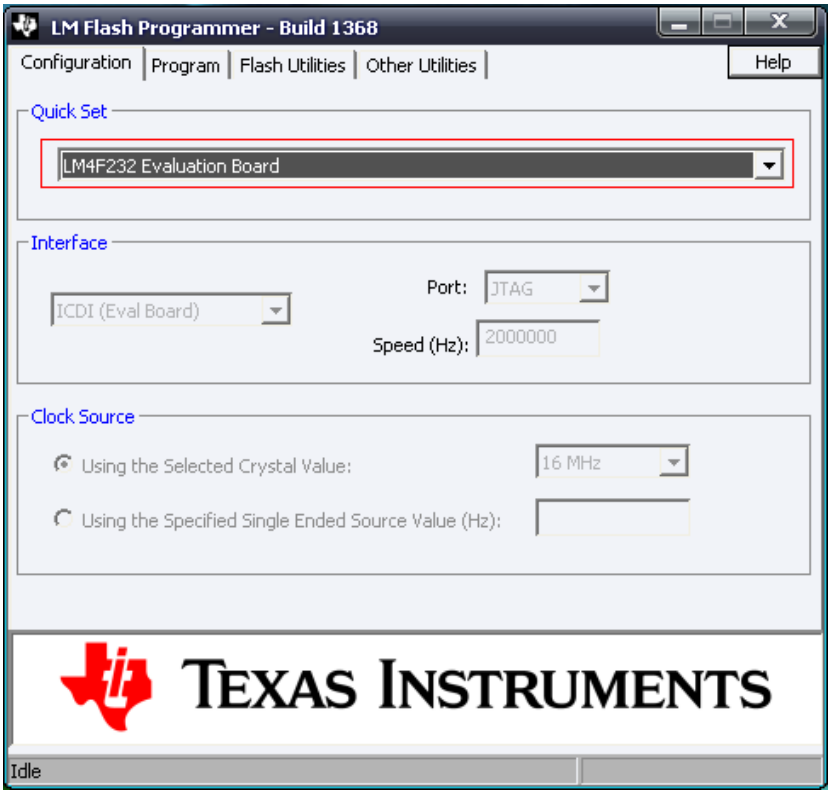
Перед построением убедитесь, что установлена утилита **StellarisWare.** Путь установки : **C:/StellarisWare/**

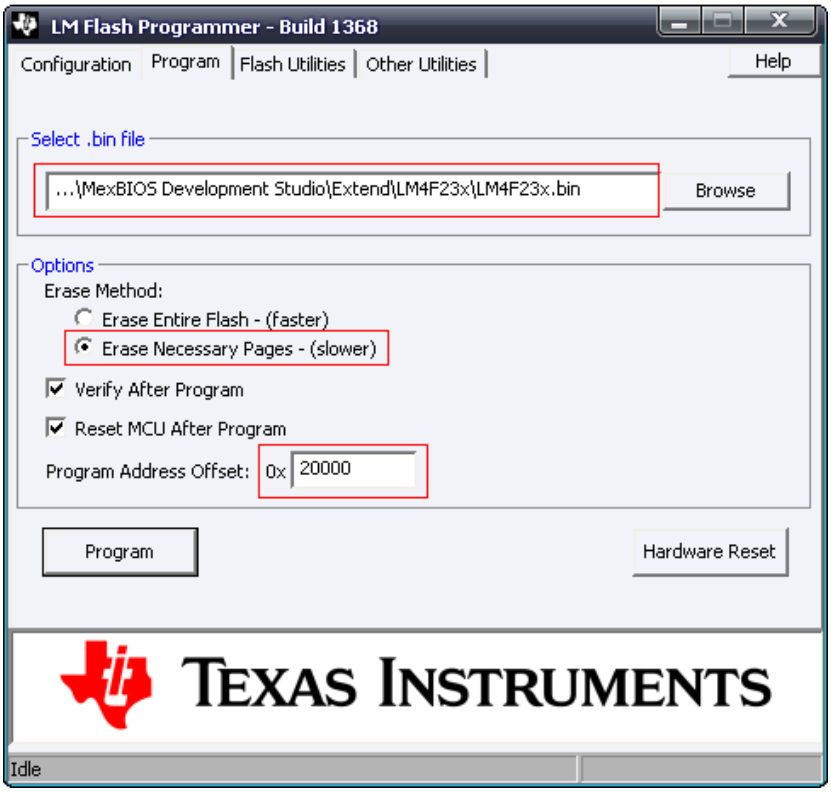


Выполните построение библиотеки блоков и стартового проекта, нажав во вкладке **Библиотека** кнопку **Построить**.

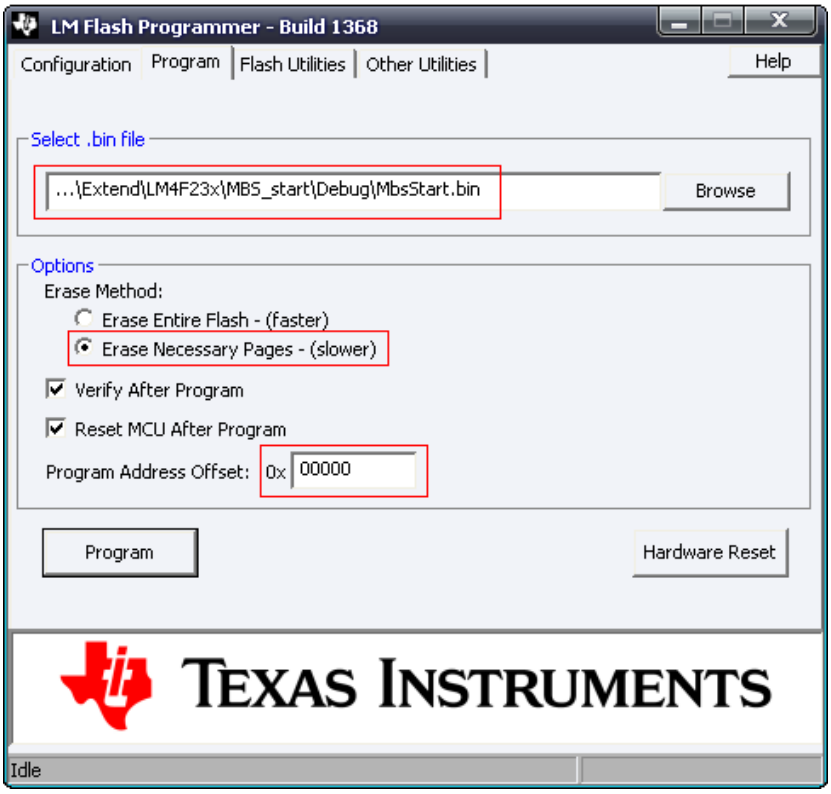
Загрузка стартового проекта и библиотеки блоков в память процессора

Загрузить с помощью утилиты «**LM Flash Programmer**» библиотеки и ядра MexBIOS в соответствии с заданными настройками





10) Загрузить стартовый проект

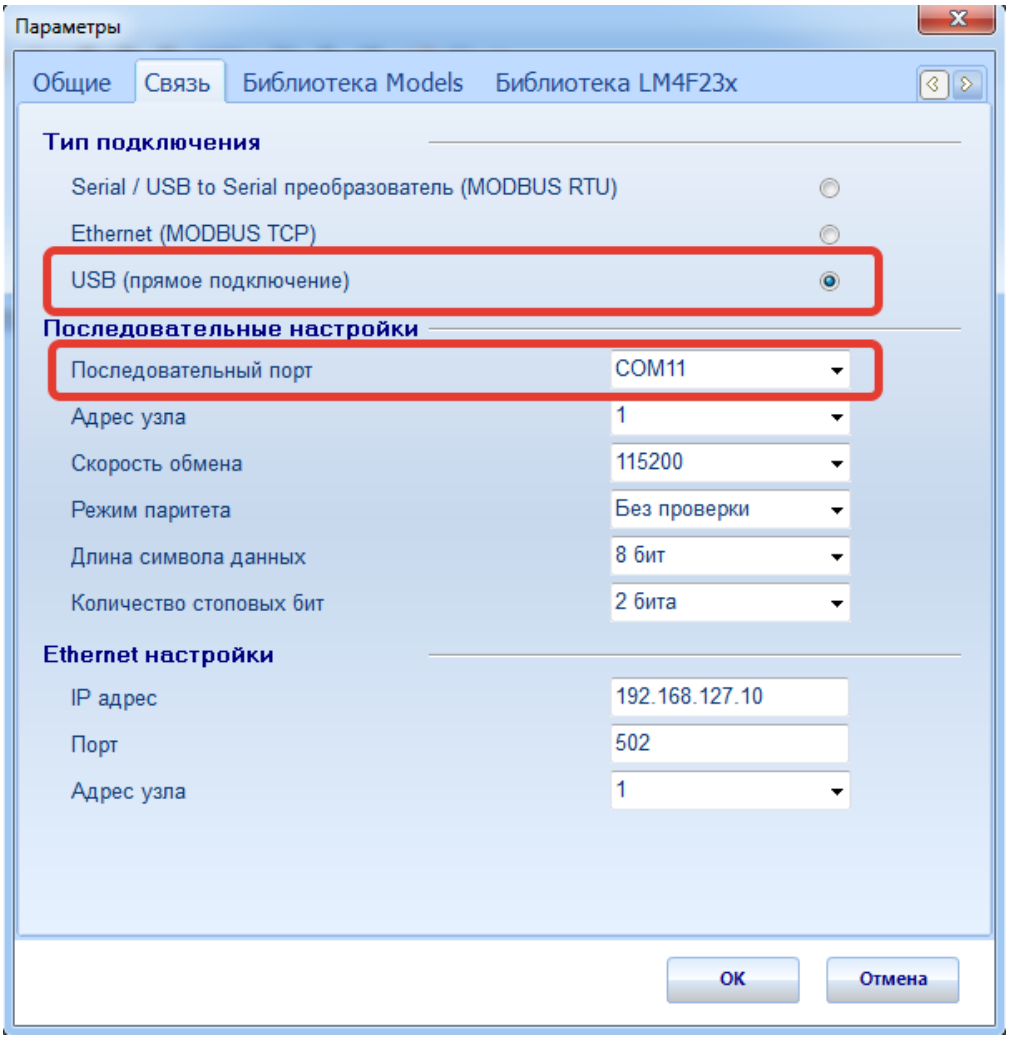


Подключение к плате LM4F232H5QD

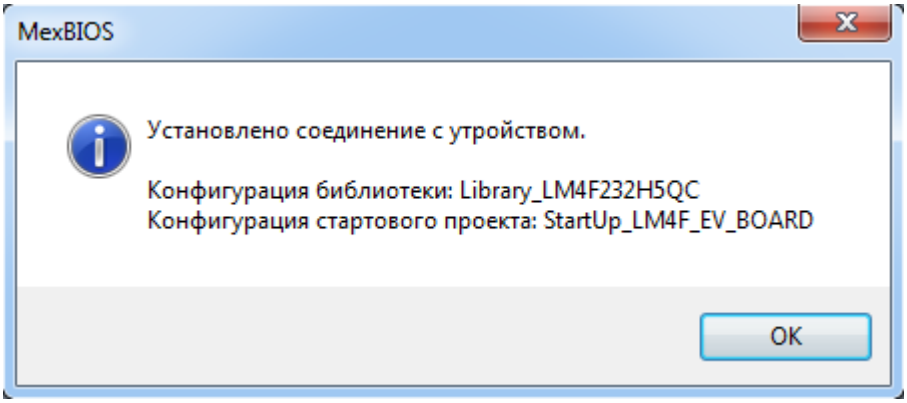
Подключите питание к плате через miniUSB-кабель. Убедитесь, что LED индикатор POWER загорелся.

Подключите прямое питание процессора через micro-USB.

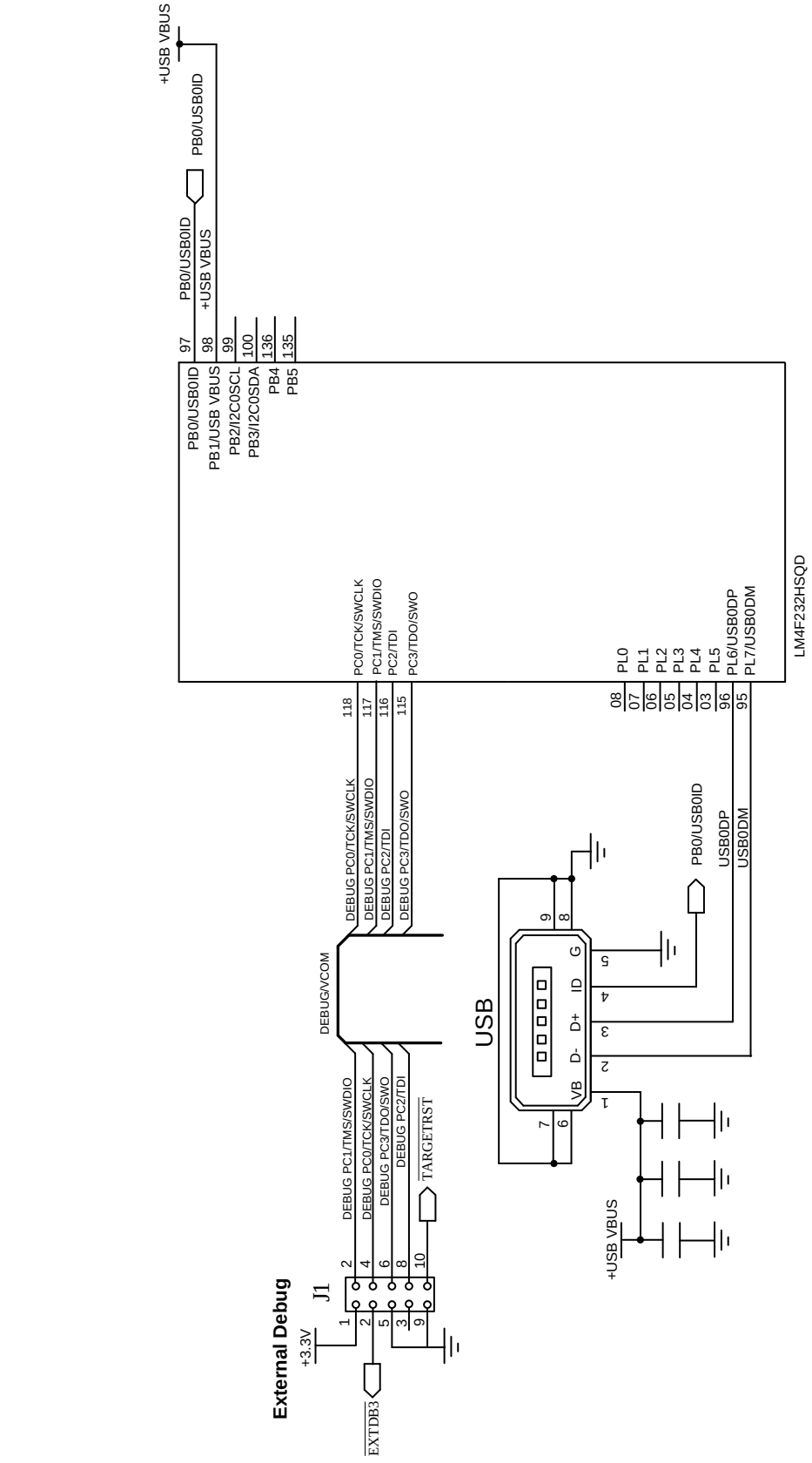
Выберите тип подключения и последовательные настройки:



Во вкладке **Устройство** нажмите кнопку **Подключиться**. В результате успеха появится сообщение:



Приложение


+USB VBUS
PB0/USB0ID
97
98
99
100
136
135
PB0/USB0ID
+USB VBUS
PB0/USB0ID
PB1/USB VBUS
PB2/I2C0SCL
PB3/I2C0SDA
PB4
PB5
LM4F232HSQD
PC0/TCK/SWCLK
PC1/TMS/SWDIO
PC2/TDI
PC3/TDO/SWO
118
117
116
115
PL0
PL1
PL2
PL3
PL4
PL5
PL6/USB0DP
PL7/USB0DM
08
07
06
05
04
03
96
95
DEBUG/VCOM
DEBUG PC0/TCK/SWCLK
DEBUG PC1/TMS/SWDIO
DEBUG PC2/TDI
DEBUG PC3/TDO/SWO
USB
PB0/USB0ID
USB0DP
USB0DM
9
8
5
4
3
2
1
7
6
G
ID
D+
D-
VB
+USB VBUS
External Debug
J1
+3.3V
2
4
6
8
10
1
2
5
3
9
DEBUG PC1/TMS/SWDIO
DEBUG PC0/TCK/SWCLK
DEBUG PC3/TDO/SWO
DEBUG PC2/TDI
TARGETRST
EXTDB3