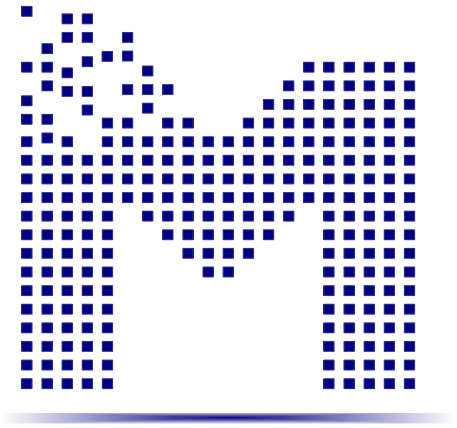


НПФ «Мехатроника-Про»

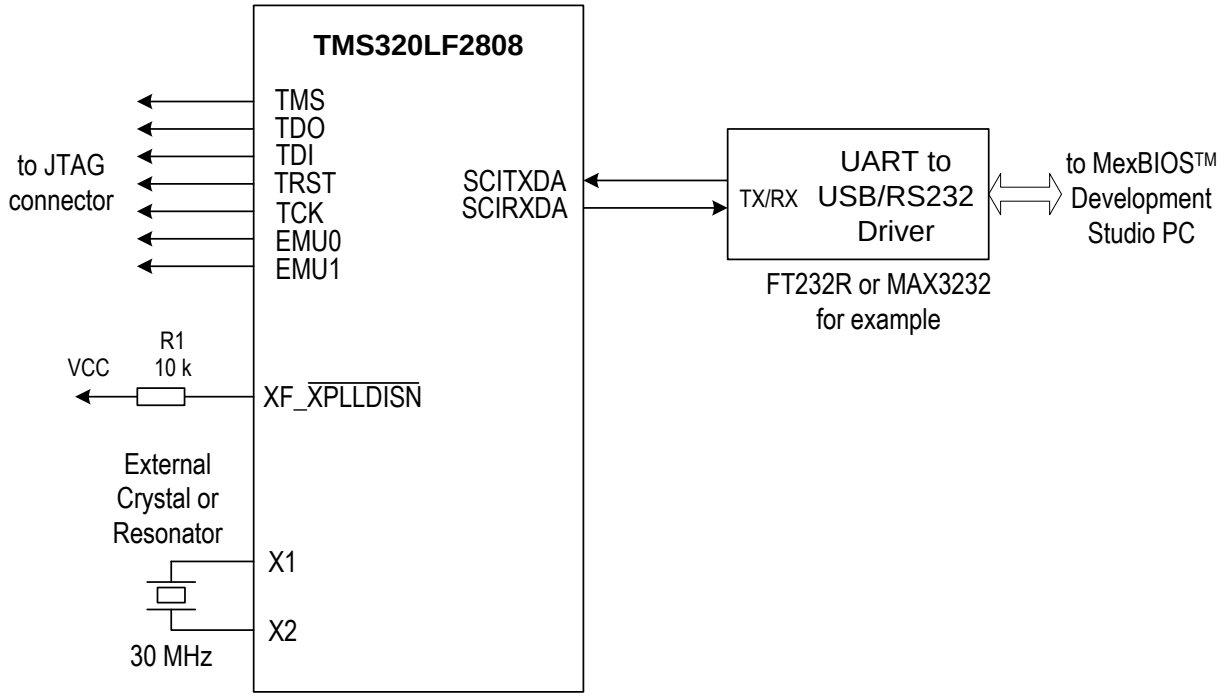
Описание Стартового проекта TMS320F280x в среде Code Composer Studio 3.1

Содержание

1. Описание стартового проекта

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

- «**DSP280x_CodeStartBranch.asm**» – модуль, содержащий секцию точки входа в программу для ее запуска;
- «**DSP280x_GlobalVariableDefs.c**» – модуль объявления периферийных регистров микроконтроллера;
- «**DSP280x_Init.c**» – модуль конфигурации оборудования;
- «**main.c**» – главный модуль, содержащий секцию инициализации, фоновую задачу и обработчики прерываний;
- «**drv_comm_280x.lib**» – библиотека драйвера коммуникации;
- «**config.h**» – модуль задания базовой конфигурации и определений.



1.1. Модуль конфигурации оборудования «DSP280x_Init»

Содержит следующие функции:

- «**InitSysCtrl**» – инициализация системных управляющих регистров (отключение сторожевого таймера, конфигурация **PLL**, разрешение работы встроенных модулей периферии);
- «**InitGpio**» – конфигурация ножек микроконтроллера (по умолчанию все задаются как входы);
- «**InitPieCtrl**» – конфигурация регистров управления менеджером прерываний;
- «**InitPieVectTable**» – инициализация таблицы векторов прерываний;
- «**InitXintf**» – инициализация внешнего интерфейса;
- «**InitEvRegs**» – конфигурация менеджеров событий (например, для работы **PWM**);
- «**InitCpuTimers**» – инициализация структур управления системными **CPU** таймерами;
- «**ConfigCpuTimer**» – конфигурация регистров заданного **CPU** таймера;
- «**MemCopy**» – функция копирования секции памяти;
- «**InitFlash**» – инициализация регистров управления **FLASH**.

1.2. Основной модуль «main.c»

Содержит следующие функции:

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

  В данной функции осуществляется вызов функций: конфигурации оборудования (**InitHardware**), инициализация коммуникации (**SerialCommInit**), при наличии драйвера памяти функции чтения конфигурации операционной среды MexBIOS (**ReadMexBiosConfig**), 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**).

- «**CpuTimer0IsrHandler**» – обработчик прерывания для формирования основной частоты расчета (периода дискретизации системы).

  В данной функции осуществляется вызов функций: функции реализации коммуникации по последовательному интерфейсу для загрузки конфигурации MexBIOS™ в микроконтроллер (**SerialCommTimings**), 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**).

- «**SciaRxIsrHandler**» – обработчик прерывания по приему данных.

  В данной функции осуществляется вызов соответствующей функции из драйвера коммуникации.

- «**SciaTxIsrHandler**» – обработчик прерывания при передаче данных

В данной функции осуществляется вызов соответствующей функции из драйвера коммуникации.

1.3. Модуль конфигурации «config.h»

В данном модуле осуществляется подключение необходимых заголовочных файлов, а также определение системных констант:

- «**VERSION**» – версия проекта;
- «**SYSCLK**» – системная тактовая частота определяющая количество исполняемых операций в секунду;
- «**CLKIN**» – частота внешнего кварца (для определения множителя частоты);
- «**HZ**» – частота основного прерывания (для модуля CPU-Timer0).

2. Подключение драйвера коммуникации

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

1) Подключение библиотеки драйвера к проекту

«**Project**» → «**Add Files to Project …**» → выбрать из директории «**lib**» папки проекта библиотеку «**drv_comm_280x.lib**»

2) Подключить в модуле «**config**» заголовочный файл драйвера

```
#include "drv_comm.h"        // Communication library header
```

3) Объявить прототипы функций обработки прерываний в модуле конфигурации «InitHardware»

```
// interrupt handler prototypes
interrupt void CpuTimer0IsrHandler(void);
interrupt void SciaRxIsrHandler(void);
interrupt void SciaTxIsrHandler(void);
interrupt void PIE_RESERVED(void);
```

4) Добавить их в таблицу векторов прерываний

```
// User interrupts that are used in this project are re-mapped to
// ISR functions found within this file
EALLOW;
PieVectTable.TINT0  = &CpuTimer0IsrHandler;
PieVectTable.RXAINT = &SciaRxIsrHandler;
PieVectTable.TXAINT = &SciaTxIsrHandler;
EDIS;
```

5) Разрешить прерывания по приему и передаче данных в менеджере обработки прерываний:

```
// Enable user interrupts
IER |= (M_INT1|M_INT9);

PieCtrlRegs.PIEIER1.bit.INTx7 = 1;   // TINT0
PieCtrlRegs.PIEIER9.bit.INTx1 = 1;   // SCIRXINTA
PieCtrlRegs.PIEIER9.bit.INTx2 = 1;   // SCITXINTA
```

6) Выставить ножки микроконтроллера на работу функции в режиме SCI

```
EALLOW;

// User configuration
GpioMuxRegs.GPFMUX.bit.SCITXDA_GPIOF4 = 1;    // SCIARX
GpioMuxRegs.GPFMUX.bit.SCIRXDA_GPIOF5 = 1;    // SCIATX

EDIS;
```

7) В модуле «main» объявить структуру конфигурации и задать ее параметры:

```
// include base header file
#include "config.h"

// Communication parameters
TDrvCommParams CommParams = {
   0,                  // UART identificator - SCIA
   1,                  // Slave adrress
   1152,               // Baudrate / 100
   SYSCLK/8/115200-1,  // Baudrate register value
   0,                  // Parity mode - None
   5,                  // Delay on byte reception (5/10000=0.0005 s)
   0,                  // Delay before transmition begins (0 s)
   HZ/1000,            // Timing function frequency in kHz
   0,                  // Transmition control function
   #ifdef MBS_INCLUDE
   (TDrvCommCallBack)MBS_GetDataAddr // Callback function
   #else
   0
   #endif
};
```

Примечание. В данном примере для обработки принятого кадра по последовательному интерфейсу используется функция ядра **MexBIOS** (**MBS_GetData**). Пользователь при этом может создать собственную функцию и передать ее в драйвер коммуникации.

8) В функции «main» добавить инициализацию

```
// main function
void main(void)
{
  // hardware initialization
  InitHardware();

  // comminication initialization
  SerialCommInit(&CommParams);
```

9) В обработчике основного системного прерывания добавить функцию реализации

```
interrupt void CpuTimer0IsrHandler(void)
{
  // clear timer counter for cpu load calculation
  CpuTimer1Regs.TIM.all = 0;

  // increment interrupt counter
  CpuTimer0.InterruptCount++;

  // execute function for comminication
  SerialCommTimings();
```

10) Объявить обработчики прерываний по приему и передаче данных и вызвать в них соответствующие функции:

```
// SCIA recieve interrupt handler
interrupt void SciaRxIsrHandler(void)
{
  // run apopriate function from communication driver
  SerialCommRxHandler();
  // acknoledge interrupt
  PieCtrlRegs.PIEACK.bit.ACK9 = 1;
}
// SCIA transmit interrupt handler
interrupt void SciaTxIsrHandler(void)
{
  // run apopriate function from communication driver
  SerialCommTxHandler();
  // acknoledge interrupt
  PieCtrlRegs.PIEACK.bit.ACK9 = 1;
}
```

3. Подключение ядра MexBIOS

Для подключения необходимо выполнить следующие этапы:

1) Подключить в модуле «**config.h**» заголовочный файл

```
#include "MBS_Import.h"      // MexBIOS library import header
```

Примечание. Данный файл автоматически генерируется в папке библиотеки блоков «\NPF Mechatronica-pro\MexBIOS Development Studio\Extend\TMS320F280x»

2) В структуру конфигурации коммуникации добавить адрес

```
// Communication parameters
TDrvCommParams CommParams = {
  0,                  // UART identificator - SCIA
  1,                  // Slave adrress
  1152,               // Baudrate / 100
  SYSCLK/8/115200-1,  // Baudrate register value
  0,                  // Parity mode - None
  5,                  // Delay on byte reception (5/10000=0.0005 s)
  0,                  // Delay before transmition begins (0 s)
  HZ/1000,            // Timing function frequency in kHz
  0,                  // Transmition control function
  #ifdef MBS_INCLUDE
  (TDrvCommCallBack)MBS_GetDataAddr // Callback function
  #else
  0
  #endif
};
```

3) Выполнить операцию чтения конфигурации из энергонезависимой памяти

```
// if memory drive enable execute function for
// reading MexBIOS configuration
#if 0
ReadMexBiosConfig(Address, MbsMtxAddr, MbsMtxSize);
#endif
```

Примечание. Здесь приведен пример вызова такой функции. Параметры этой функции «**MbsMtxAddr**» и «**MbsMtxSize**» определены в заголовочном файле «**MBS_Import.h**», а параметр «**Address**» может назначить 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).

```
// if MexBIOS enable execute function for
// setting it base parameters
#ifdef MBS_INCLUDE
MBS_Create();
*MbsAppVersion = VERSION;   // set project version to dislay in the studio
*MbsEnable = 1;             // command to startup
#endif
```

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

```
// run forewer
while(1)
{
   // if MexBIOS enable execute functions for
   // setting configuration and execute tasks
   #ifdef MBS_INCLUDE
   MBS_Init();
   MBS_TaskExecute();
   #endif
}
```

6) В обработчике основного прерывания «CpuTimer0IsrHandler» вызвать обработчик периодических задач ядра

```
// execute function with interrupt source
MBS_IsrExecute();
```

7) По команде выполнить операцию записи конфигурации в энергонезависимую память

```
// if memory driver enable check write command
#if 0
if (*MbsWriteCmd)
{
   // write MexBIOS configuration in memory
   WriteMexBiosConfig(Address, MbsMtxAddr, MbsMtxSize);
   *MbsWriteCmd = 0;
}
#endif
```

Примечание. Здесь приведен пример вызова такой функции. Параметры этой функции «**MbsMtxAddr**» и «**MbsMtxSize**» определены в заголовочном файле «**MBS_Import.h**», а параметр «**Address**» может назначить по своему усмотрению.

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

```
// CPU load calculation
      *MbsCpuLoad = (CPUL_GAIN * (-(Uint16)CpuTimer1Regs.TIM.all)) >> 21;
```

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

```
MEMORY
{
PAGE 0: /* Program Memory */
      M1SARAM           : origin = 0x000480, length = 0x000380
      OTP               : origin = 0x3D7800, length = 0x000400
      FLASH             : origin = 0x3F4000, length = 0x003F80
/*    FLASH             : origin = 0x3E8000, length = 0x00C000    */
      CSM_PWL           : origin = 0x3F7FF8, length = 0x000008
      CSM_RSVD          : origin = 0x3F7F80, length = 0x000076
      BEGIN             : origin = 0x3F7FF6, length = 0x000002
      BOOTROM           : origin = 0x3FF000, length = 0x000FC0
      RESET             : origin = 0x3FFFC0, length = 0x000002
      VECTORS           : origin = 0x3FFFC2, length = 0x00003E

PAGE 1: /* Data Memory */
/*    M0SARAM           : origin = 0x000000, length = 0x000400    */
      BOOT_RSVD         : origin = 0x000400, length = 0x000080
/*    L0SARAM           : origin = 0x008000, length = 0x001000    */
      L1SARAM           : origin = 0x009000, length = 0x001000
      H0SARAM           :origin = 0x00A000, length = 0x002000
}
```

Данные настройки необходимо выставлять в соответствии с настройки выставленными в окне «**Options**» на вкладке **Edit**.