Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет ИУ «Информатика и системы управления»

Кафедра ИУ-3 «Информационные системы и телекоммуникации»

Методические указания
по выполнению лабораторной работы 3
«Исследование устройств обработки сигналов»
по дисциплине «Микропроцессорные устройства обработки сигналов»

Для студентов, обучающихся по направлениям
2304002468, 2304007468 и 2302010065

Продолжительность работы 8 часов

Составители:
проф. каф. ИУЗ, д.т.н. В.С. Выхованец
преп. каф. ИУЗ А.И. Германчук

Москва, 2020

Содержание

1 Общие указания

Целью работы является разработка и исследование программы обработки сигналов на языке программирования Си, реализующей звуковой эффект, определенный в индивидуальном задании. При выполнении лабораторной работы используется экспериментальная плата TMS320C5515™ Evaluation Module™ (EVM) компании Spectrum Digital Incorporated® [1], показанная на рис. 1.

Для подготовки к лабораторной работе необходимо:

– повторить материал лекций по архитектуре и организации микропроцессора TMS320C5515 и его программированию и изучить исходные тексты программ из приложений;

– разработать исходный текст функции на языке Си, реализующей алгоритм обработки данных, заданный в индивидуальном задании.

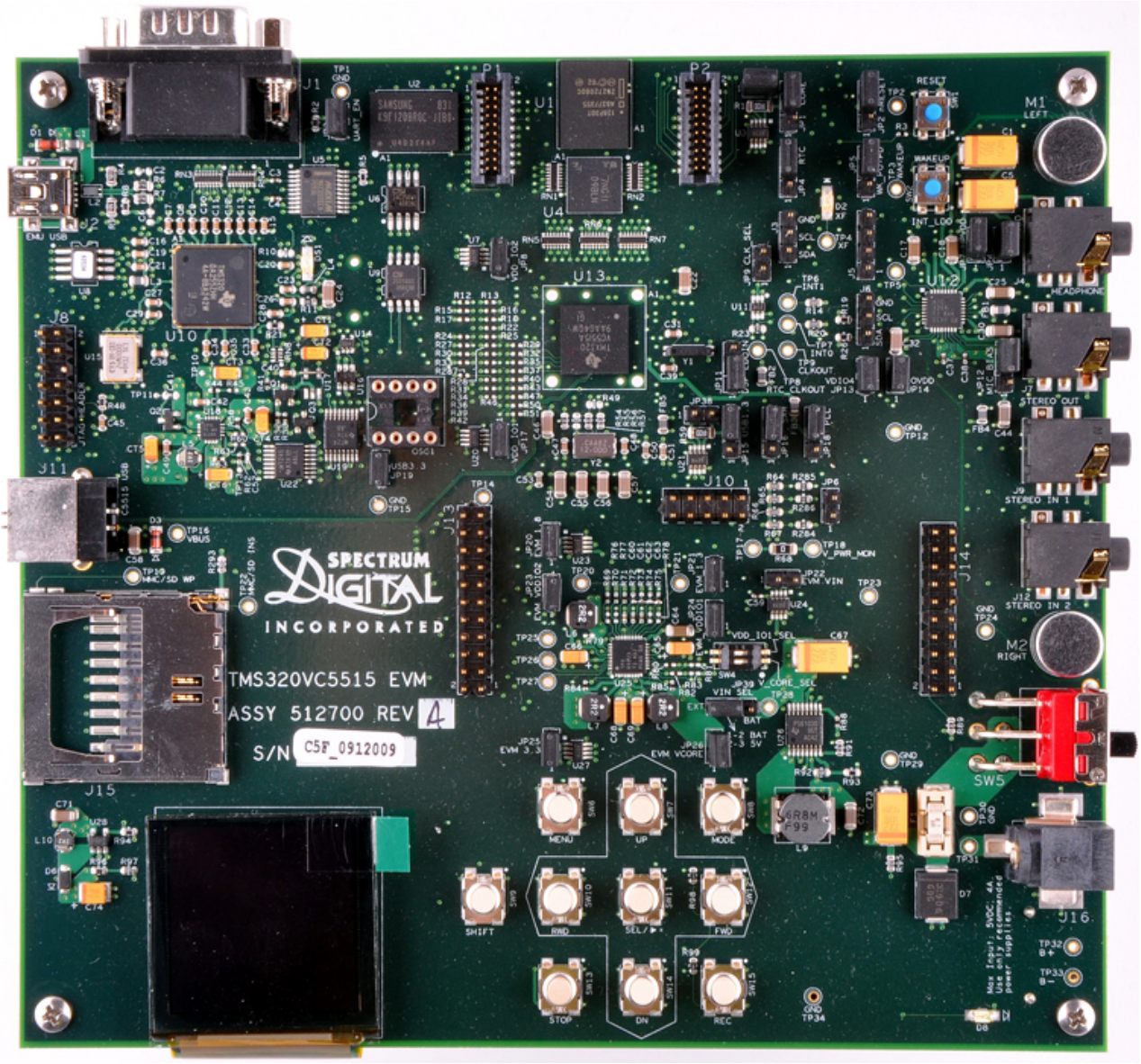

Рисунок 1 – Общий вид экспериментальной платы EVM5515

2 Выполнение лабораторной работы

2.1 Подключение экспериментальной платы

2.1.1 Установите исходное состояние перемычек экспериментальной платы (рис. 2):

– JP2 (nRESET Select) в позиции 1–2;

– JP3 (UART_EN) закорочен;

– JP5 (WK_PU_PD_SEL) в позиции 2–3;

– JP6 (LDO_EN) закорочен;

– JP9 (CLK_SEL) открыт;

– JP12 (MIC_BIAS) в позиции 2–3;

– JP39 (VIN Select) в позиции 2–3.

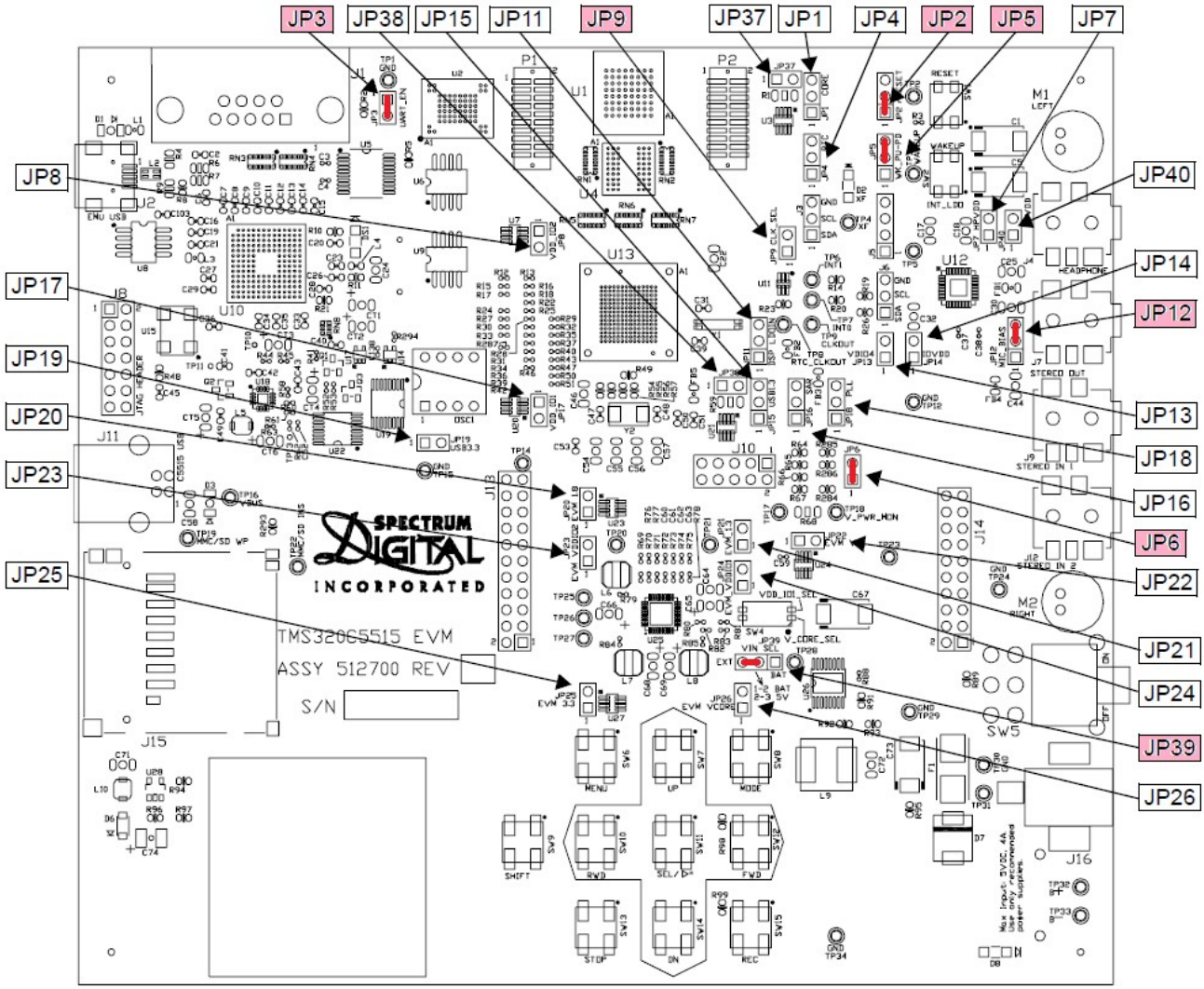

Рисунок 2 – Перемычки экспериментальной платы EVM5515

2.1.2 Подключите выход блока питания (рис. 3, а) к разъему экспериментальной платы J16, а интерфейсным кабелем USB (рис. 3, б) соедините разъему J2 экспериментальной платы с разъемом USB компьютера.

а) б)

Рисунок 3 – Блок питания и интерфейсный кабель USB

2.1.3 Подключите блок питания к сети и включите питание платы переключателем SW5.

2.2 Создание проекта

2.2.1 Для создания проекта выберите пункты меню *File – New – CCS Project*. Откроется форма нового проекта *New CCS Project*.

2.2.2 В поле *Project Name* введите имя проекта, например, MPUOS_LAB3. При установке флага Use default location проект будет создан в папке текущего пользователя. В противном случае место размещения папки проекта в файловой системе следует выбрать после нажатия на кнопку *Browse*.

2.2.3 В ниспадающем списке *Output Type* выберите *Executable* для создания исполняемой программы.

2.2.4 В разделе *Device* в ниспадающем списке *Family* выберите семейство микропроцессоров *C5500*, в списке *Variant* – модель микропроцессора *TMS320C5515*. а в списке *Connection* – тип экспериментальной платы *Spectrum Digital DSK-EVM-eZdsp onboard USB Emulator*.

2.2.5 В разделе *Project Template and Example* выберите пустой проект *Empty Project*. и нажмите кнопку *Finish*. CCS создаст проект, который отобразится в окне Project Explorer.

2.2.6 Для включения в проект новых файлов через оконное меню *View* откройте панель *Project Explorer* и в контекстном меню, открывающемся при нажатии правой кнопки мыши на имени созданного проекта, выберите пункт *New* и создайте файлы модулей программы и командный файл компоновщика (см. приложения). Для добавления в проект существующих

файлов в контекстном меню проекта выберите пункт меню *Add Files* и в появившемся окне укажите место размещения включаемых в проект файлов.

Примечание. Папка с тестовым проектом для среды разработки программ CCS 5 размещена на сайте дисциплины в разделе «Программные средства», пункт Р05.

2.3 Выбор целевой платформы

2.3.1 До отладки проекта необходимо выбрать и сконфигурировать целевую платформу, на которой будет выполняться проект. Целевой платформой должен быть эмулятор. В этом случае к компьютеру должна быть подключена экспериментальная плата.

2.3.2 Для выбора целевой платформы создайте конфигурационный файл, для чего в контекстном меню проекта выберите пункт *File – New – Target Configuration File*, или активизируйте окно текущего конфигурационного файла через его контекстное меню на панели *Project Explorer*, пункт меню *Open*. В открывшемся окне задайте имя конфигурационного файла, например, *TMS320C5515*, к которому будет добавлено стандартное расширение *.ccxml*.

2.3.3 Во вкладке с именем конфигурационного файла (*TMS320C5515.ccxml*) выберите тип экспериментальной платы *Spectrum Digital DSK-EVM-eZdsp onboard USB Emulator* из ниспадающего списка *Connection,* а в списке *Board or Device* установите флаг модели микропроцессора *TMS320C5515*.

2.3.4 Завершите конфигурирование, нажав на кнопку *Save* в окне конфигурационного файла. Проконтролируйте, что конфигурационный файл появился в списке файлов проекта с установленным атрибутом активности *Active/Default*.

2.4 Сборка и отладка проекта

2.4.1 Выполните сборку проекта, выбрав пункт меню *Project – Build Project*. Исправьте обнаруженные ошибки компиляции (ассемблирования) и повторите сборку снова.

2.4.2 В случае необходимости измените параметры сборки, выбрав в контекстном меню проекта (панель *Project Explorer*) пункт *Show Build Settings*. Аналогичные настройки через контекстное меню, открывающееся при нажатии правой кнопки мыши на имени файла, можно выполнить и для каждого файла проекта.

2.4.3 Для запуска CCS в режиме отладки выберите пункт меню *Run – Debug*. По этой команде запуститься отладчик и откомпилированный ранее загрузочный файл с именем проекта и расширением *.out* загрузится для исполнения в память целевой платформы.

2.4.4 Для выполнения загруженной программы по шагам используйте пункты меню *Run – Step Into* (F5) или *Run – Step Over* (F6), а для запуска программы – *Run – Resume* (F8).

2.4.5 При пошаговой отладке программы содержимое регистров процессора контролируйте в окне *Registers*, содержимое ячеек памяти – в окне *Memory*, текст программы с подсвеченным текущим оператором – в окне редактора соответствующего исходного файла, а исполняемый код программы на языке ассемблера – в окне *Disassemly*. Для открытия этих окон необходимо выбрать соответствующие пункты в меню *View*.

2.4.6 Для ускорения прохода отладчика до требуемого оператора (команды) используйте точки останова. Для задания точки останова установите фокус ввода на требуемый оператор (команду) и выберите в меню *Run* пункт *New Breakpoint* или *Toggle Breakpoint*. После установки точек останова используйте команду запуска программы *Run – Resume* (F8), которая приведет к выполнению программы до тех пор, пока не наступит очередь выполнения команды, помеченной точкой останова.

2.5 Исследование звукового эффекта

2.5.1 Подключите наушники к гнезду оценочной плате.

2.5.2 Закомментируйте в главной функции main.c строки, ответственные за вывод на наушники звука с микрофона.

2.5.3 Запустите программу, выбрав пункт меню *Run – Resume*, и прослушайте звук частотой 1 кГц в наушниках.

2.5.4 Подключите к оценочной плате микрофон (источник стереозвука к стереовходу 1).

2.5.5 Модифицируйте главную функцию main.c для вывода на наушники звука с микрофона (стерео входа 1).

2.5.6 Запустите программу и прослушайте в наушниках звук, снимаемый с микрофона.

2.5.7 Модифицируйте главную функцию main.c в соответствии с требованиями индивидуального задания.

2.5.8 Запустите программу и прослушайте звук с исследуемым звуковым эффектом.

3 Требования к отчету

Отчет по домашнему заданию должен содержать:

- титульный лист;
- содержание;
- описание звукового эффекта;

– результаты экспериментального исследования;
– выводы и рекомендации;
– список использованной литературы;
– приложение (текст с разработанной программой на языке Си).

Оформление отчета осуществляется в соответствии с ГОСТ 2.105-95 [3], список литературы по ГОСТ Р 7.0.5–2008 [4].

Список литературы

[1] TMS320C5515 Evaluation Module (EVM). Technical Reference. – Spectrum Digital, 2010. – 76 p.

[2] DAFX – Digital Audio Effects / Ed. Udo Zolzer. – Chichester: John Wiley & Sons, 2002. – 554 p.

[3] ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам. – М.: Изд-во стандартов, 2012. – 26 с.

[4] ГОСТ Р 7.0.5–2008. Библиографическая ссылка. Общие требования и правила составления. – М.: Изд-во стандартов, 2009. – 23 с.

Приложение А

Командный файл компоновщика c5515.cmd

```
-stack 0x2000
-sysstack 0x1000
-heap 0x2000
-c
-u _Reset

MEMORY
{
PAGE 0:
MMR (RWIX): origin = 0x000000, length = 0x0000c0
DARAM0 (RWIX): origin = 0x0000c0, length = 0x00ff40
SARAM0 (RWIX): origin = 0x010000, length = 0x010000
SARAM1 (RWIX): origin = 0x020000, length = 0x020000
SARAM2 (RWIX): origin = 0x040000, length = 0x00FE00
VECS (RWIX): origin = 0x04FE00, length = 0x000200
PDROM (RIX): origin = 0xff8000, length = 0x008000
PAGE 2:
IOPORT (RWI) : origin = 0x000000, length = 0x020000
}

SECTIONS
{
.text >> SARAM1|SARAM2|SARAM0
.stack > DARAM0
.sysstack > DARAM0
.data >> DARAM0|SARAM0|SARAM1
.bss >> DARAM0|SARAM0|SARAM1
.const >> DARAM0|SARAM0|SARAM1
.sysmem > DARAM0|SARAM0|SARAM1
.switch > SARAM2
.cinit > SARAM2
.pinit > SARAM2
.cio > SARAM2
.args > SARAM2
vectors > VECS
.ioport > IOPORT PAGE 2
}
```

Приложение В

Заголовочный файл c5515.h

```
#ifndef STK5505_
#define STK5505_

#define Uint32  unsigned long
#define Uint16  unsigned short
#define Uint8   unsigned char
#define Int32   int
#define Int16   short
#define Int8    char

#define SW_BREAKPOINT      while(1);

#define SYS_EXBUSSEL       *(volatile ioport Uint16*)(0x1c00)
#define SYS_PCGCR1         *(volatile ioport Uint16*)(0x1c02)
#define SYS_PCGCR2         *(volatile ioport Uint16*)(0x1c03)
#define SYS_PRCNTR         *(volatile ioport Uint16*)(0x1c04)
#define SYS_PRCNTRLR       *(volatile ioport Uint16*)(0x1c05)
#define SYS_GPIO_DIR0      *(volatile ioport Uint16*)(0x1c06)
#define SYS_GPIO_DIR1      *(volatile ioport Uint16*)(0x1c07)
#define SYS_GPIO_DATAIN0   *(volatile ioport Uint16*)(0x1c08)
#define SYS_GPIO_DATAIN1   *(volatile ioport Uint16*)(0x1c09)
#define SYS_GPIO_DATAOUT0  *(volatile ioport Uint16*)(0x1c0a)
#define SYS_GPIO_DATAOUT1  *(volatile ioport Uint16*)(0x1c0b)
#define SYS_OUTDRSTR       *(volatile ioport Uint16*)(0x1c16)
#define SYS_SPPDIR         *(volatile ioport Uint16*)(0x1c17)

Int16 EVM5515_init( );
void EVM5515_wait( Uint32 delay );
void EVM5515_waitusec( Uint32 usec );

#endif
```

Приложение С

Модуль микропроцессора c5515.c

```
#include "C5515.h"

void c5515_wait( Uint32 delay )
{
    volatile Uint32 i;
    for ( i = 0 ; i < delay ; i++ ){ };
}

void c5515_waitusec( Uint32 usec )
{
    c5515_wait( (Uint32)usec * 8 );
}

Int16 c5515_init( )
{
    /* Enable clocks to all peripherals */
    SYS_PCGCR1 = 0x0000;
    SYS_PCGCR2 = 0x0000;

    return 0;
}
```

Приложение D

Заголовочный файл приборного интерфейса i2c.h

```
#ifndef I2C_
#define I2C_

#include "C5515.h"

#define I2C_IER            *(volatile ioport Uint16*)(0x1A04)
#define I2C_STR            *(volatile ioport Uint16*)(0x1A08)
#define I2C_CLKL           *(volatile ioport Uint16*)(0x1A0C)
#define I2C_CLKH           *(volatile ioport Uint16*)(0x1A10)
#define I2C_CNT            *(volatile ioport Uint16*)(0x1A14)
#define I2C_DRR            *(volatile ioport Uint16*)(0x1A18)
#define I2C_SAR            *(volatile ioport Uint16*)(0x1A1C)
#define I2C_DXR            *(volatile ioport Uint16*)(0x1A20)
#define I2C_MDR            *(volatile ioport Uint16*)(0x1A24)
#define I2C_EDR            *(volatile ioport Uint16*)(0x1A2C)
#define I2C_PSC            *(volatile ioport Uint16*)(0x1A30)

#define MDR_STT            0x2000
#define MDR_TRX            0x0200
#define MDR_MST            0x0400
#define MDR_IRS            0x0020
#define MDR_FREE           0x4000
#define STR_XRDY           0x0010
#define STR_RRDY           0x0008
#define MDR_STP            0x0800

Int16 I2C_init ( );
Int16 I2C_close( );
Int16 I2C_read( Uint16 i2c_addr, Uint8* data, Uint16 len );
Int16 I2C_write( Uint16 i2c_addr, Uint8* data, Uint16 len );

#endif
```

Приложение Е

Модуль приборного интерфейса i2c.c

```
#include "i2c.h"

Int32 i2c_timeout = 0x0fff;

/* ------------------------------------------------------------------------*
*  _I2C_init( )                                                            *
*      Enable and initalize the I2C module                                 *
*      The I2C clk is set to run at 20 KHz                                 *
* ------------------------------------------------------------------------ */
Int16 I2C_init( )
{
    I2C_MDR = 0x0400;        // Reset I2C
    I2C_PSC   = 15;          // Config prescaler for 100MHz
    I2C_CLKL  = 25;          // Config clk LOW for 100kHz
    I2C_CLKH  = 25;          // Config clk HIGH for 100kHz
    I2C_MDR   = 0x0420;      // Release reset; Master, Transmitter, 7-bit address
    return 0;
}
/* ------------------------------------------------------------------------*
*  _I2C_close( )                                                           *
* ------------------------------------------------------------------------ */
Int16 I2C_close( )
{
    I2C_MDR = 0;              // Reset I2C
    return 0;
}
/* ------------------------------------------------------------------------ *
*  _I2C_reset( )                                                           *
*      Сброс I2C                                                           *
* ------------------------------------------------------------------------  */
Int16 I2C_reset( )
{
    I2C_close( );
    I2C_init( );
    return 0;
}
/* ------------------------------------------------------------------------ *
 *  _I2C_write( i2c_addr, data, len )                                       *
 *      I2C write in Master mode                                            *
 *      i2c_addr    <- I2C slave address                                    *
 *      data        <- I2C data ptr                                         *
 *      len         <- # of bytes to write                                  *
* ------------------------------------------------------------------------ */
Int16 I2C_write( Uint16 i2c_addr, Uint8* data, Uint16 len )
{
    Int16 timeout, i;
   //I2C_IER = 0x0000;
    I2C_CNT = len;                    // Set length
    I2C_SAR = i2c_addr;               // Set I2C slave address
    I2C_MDR = MDR_STT                 // Set for Master Write
              | MDR_TRX               // Передатчик
              | MDR_MST               // Ведущее устройство
              | MDR_IRS               // Разрешить работу
              | MDR_FREE;             // Работа совместно с эмулятором
    c5515_wait(100);                  // Задержка перед передачей
    for ( i = 0; i < len; i++ )
    {
       I2C_DXR = data[i];             // Запись байта в регистр передатчика
       timeout = 0x7fff;              // i2c_timeout
       // Ожидание передачи данных
       do {
```

```
            if ( timeout-- < 0  )
            {
                I2C_reset( );
                return -1;
            }
        } while ( ( I2C_STR & STR_XRDY ) == 0 );// Wait for Tx Ready
    }
    I2C_MDR |= MDR_STP;             // Генерация STOP
    c5515_waitusec(1000);           // Перерыв в линии
    return 0;
}
/* ------------------------------------------------------------------------ *
 *  _I2C_read( i2c_addr, data, len )                                        *
 *      I2C read in Master mode                                             *
 *      i2c_addr    <- I2C slave address                                    *
 *      data        <- I2C data ptr                                         *
 *      len         <- # of bytes to write                                  *
 *      Returns:    0: PASS                                                 *
 *                 -1: FAIL Timeout                                         *
 * ------------------------------------------------------------------------ */
Int16 I2C_read( Uint16 i2c_addr, Uint8* data, Uint16 len )
{
     // Локальные данные
    Int32 timeout, i;
    I2C_CNT = len;                      // Задание длины данных в посылках
    I2C_SAR = i2c_addr;                 // Задать адрес ведомого
    I2C_MDR = MDR_STT                   // Set for Master Read
              | MDR_MST                 // Ведущее устройство
              | MDR_IRS                 // Разрешить работу
              | MDR_FREE;               // Работа совместно с эмулятором
    c5515_wait( 10 );                   // Задержка перед приемом
    // Прием данных
    for ( i = 0 ; i < len ; i++ )
    {
        timeout = i2c_timeout;
        // Ожидание приема данных
        do {
            if ( timeout-- < 0 )
            {
                I2C_reset( );
                return -1;
            }
        } while ( ( I2C_STR & STR_RRDY ) == 0 );
        data[i] = I2C_DRR;              // Получить данные
    }
    I2C_MDR |= MDR_STP;                 // Генерация STOP
    c5515_waitusec(10);                 // Перерыв в линии
    return 0;
}
```

Приложение F

Заголовочный файл звукового интерфейса i2s.h

```
#ifndef I2S_H_
#define I2S_H_
* ------------------------------------------------------------------------- */
#define I2S0_CR            *(volatile ioport Uint16*)(0x2800)
#define I2S0_SRGR          *(volatile ioport Uint16*)(0x2804)
#define I2S0_W0_LSW_W      *(volatile ioport Uint16*)(0x2808)
#define I2S0_W0_MSW_W      *(volatile ioport Uint16*)(0x2809)
#define I2S0_W1_LSW_W      *(volatile ioport Uint16*)(0x280C)
#define I2S0_W1_MSW_W      *(volatile ioport Uint16*)(0x280D)
#define I2S0_IR            *(volatile ioport Uint16*)(0x2810)
#define I2S0_ICMR          *(volatile ioport Uint16*)(0x2814)
#define I2S0_W0_LSW_R      *(volatile ioport Uint16*)(0x2828)
#define I2S0_W0_MSW_R      *(volatile ioport Uint16*)(0x2829)
#define I2S0_W1_LSW_R      *(volatile ioport Uint16*)(0x282C)
#define I2S0_W1_MSW_R      *(volatile ioport Uint16*)(0x282D)
* ------------------------------------------------------------------------- */
#define I2S2_CR            *(volatile ioport Uint16*)(0x2A00)
#define I2S2_SRGR          *(volatile ioport Uint16*)(0x2A04)
#define I2S2_W0_LSW_W      *(volatile ioport Uint16*)(0x2A08)
#define I2S2_W0_MSW_W      *(volatile ioport Uint16*)(0x2A09)
#define I2S2_W1_LSW_W      *(volatile ioport Uint16*)(0x2A0C)
#define I2S2_W1_MSW_W      *(volatile ioport Uint16*)(0x2A0D)
#define I2S2_IR            *(volatile ioport Uint16*)(0x2A10)
#define I2S2_ICMR          *(volatile ioport Uint16*)(0x2A14)
#define I2S2_W0_LSW_R      *(volatile ioport Uint16*)(0x2A28)
#define I2S2_W0_MSW_R      *(volatile ioport Uint16*)(0x2A29)
#define I2S2_W1_LSW_R      *(volatile ioport Uint16*)(0x2A2C)
#define I2S2_W1_MSW_R      *(volatile ioport Uint16*)(0x2A2D)
* ------------------------------------------------------------------------- */
#endif /* I2S_H_ */
```

Приложение G

Заголовочный файл входов-выходов общего назначения gpio.h

```
#ifndef GPIO_
#define GPIO_
#include "C5515.h"
* ------------------------------------------------------------------------ */
#define GPIO_IN                 1
#define GPIO_OUT                0
* ------------------------------------------------------------------------ */
#define GPIO0                   0x00
#define GPIO1                   0x01
#define GPIO2                   0x02
#define GPIO3                   0x03
#define GPIO4                   0x04
#define GPIO5                   0x05
#define GPIO6                   0x06
#define GPIO7                   0x07
#define GPIO8                   0x08
#define GPIO9                   0x09
#define GPIO10                  0x0A
#define GPIO11                  0x0B
#define GPIO12                  0x0C
#define GPIO13                  0x0D
#define GPIO14                  0x0E
#define GPIO15                  0x0F
#define GPIO16                  0x10
#define GPIO17                  0x11
#define GPIO18                  0x12
#define GPIO19                  0x13
#define GPIO20                  0x14
#define GPIO21                  0x15
#define GPIO22                  0x16
#define GPIO23                  0x17
#define GPIO24                  0x18
#define GPIO25                  0x19
#define GPIO26                  0x1A
#define GPIO27                  0x1B
#define GPIO28                  0x1C
#define GPIO29                  0x1D
#define GPIO30                  0x1E
#define GPIO31                  0x1F
* ------------------------------------------------------------------------ */
Int16 EVM5515_GPIO_init         ( );
Int16 EVM5515_GPIO_setDirection ( Uint16 number, Uint16 direction );
Int16 EVM5515_GPIO_setOutput    ( Uint16 number, Uint16 output );
Int16 EVM5515_GPIO_getInput     ( Uint16 number );

#endif
```

Приложение Н

Модуль входов-выходов общего назначения gpio.c

```
#include "gpio.h"
* ------------------------------------------------------------------------ */
Int16 c5515_GPIO_init()
{
    return 0;
}
* ------------------------------------------------------------------------ */
Int16 c5515_GPIO_setDirection( Uint16 number, Uint16 direction )
{
    Uint32 bank_id = ( number >> 4);
    Uint32 pin_id  = ( 1 << ( number & 0xF ) );
    if (bank_id == 0)
        if ((direction & 1) == GPIO_IN)
            SYS_GPIO_DIR0 &= ~pin_id;
        else
            SYS_GPIO_DIR0 |= pin_id;
    if (bank_id == 1)
        if ((direction & 1) == GPIO_IN)
            SYS_GPIO_DIR1 &= ~pin_id;
        else
            SYS_GPIO_DIR1 |= pin_id;
    return 0;
}
* ------------------------------------------------------------------------ */
Int16 c5515_GPIO_setOutput( Uint16 number, Uint16 output )
{
    Uint32 bank_id = ( number >> 4 );
    Uint32 pin_id  = ( 1 << ( number & 0xF ) );
    if (bank_id == 0)
        if ((output & 1) == 0)
            SYS_GPIO_DATAOUT0 &= ~pin_id;
        else
            SYS_GPIO_DATAOUT0 |= pin_id;
    if (bank_id == 1)
        if ((output & 1) == 0)
            SYS_GPIO_DATAOUT1 &= ~pin_id;
        else
            SYS_GPIO_DATAOUT1 |= pin_id;
    return 0;
}
* ------------------------------------------------------------------------ */
Int16 c5515_GPIO_getInput( Uint16 number )
{
    Uint32 input;
    Uint32 bank_id = ( number >> 4 );
    Uint32 pin_id  = ( number & 0xF );
    if (bank_id == 0)
        input = (SYS_GPIO_DATAIN0 >> pin_id) & 1;
    if (bank_id == 1)
        input = (SYS_GPIO_DATAIN1 >> pin_id) & 1;
    return input;
}
* ------------------------------------------------------------------------ */
```

Приложение I

Модуль звукового кодека aic3204.c

```
#include "C5515.h"
#include "gpio.h"
#include "i2c.h"
#include "i2s.h"
#include "stdio.h"

#define Rcv 0x08
#define Xmit 0x20
#define AIC3204_I2C_ADDR 0x18
/* ------------------------------------------------------------------------ */
Int16 AIC3204_rget(  Uint16 regnum, Uint16* regval )
{
    Int16 retcode = 0;
    Uint8 cmd[2];

    cmd[0] = regnum & 0x007F;       // 7-bit Адрес устройства
    cmd[1] = 0;

    retcode |= I2C_write( AIC3204_I2C_ADDR, cmd, 1 );
    retcode |= I2C_read( AIC3204_I2C_ADDR, cmd, 1 );

    *regval = cmd[0];
    c5515_wait( 10 );
    return retcode;
}
/* ------------------------------------------------------------------------ */
Int16 AIC3204_rset( Uint16 regnum, Uint16 regval )
{
    Uint8 cmd[2];
    cmd[0] = regnum & 0x007F;       // 7-bit Адресс регистра
    cmd[1] = regval;                // 8-bit Данные регистра

    return I2C_write( AIC3204_I2C_ADDR, cmd, 2 );
}
/* --- Программа конфигурации «тон 1 кГц на наушники» --------------------- */
Int16 aic3204_sin( )
{
    AIC3204_rset(  0, 0x00 );     // Select page 0
    AIC3204_rset(  1, 0x01 );     // Reset codec
    AIC3204_rset(  0, 0x01 );     // Select page 1
    AIC3204_rset(  1, 0x08 );     // Disable crude AVDD generation from DVDD
    AIC3204_rset(  2, 0x00 );     // Enable Analog Blocks
    // PLL and Clocks config and Power Up
    AIC3204_rset(  0, 0x00 );     // Select page 0
    AIC3204_rset( 27, 0x00 );     // BCLK and WCLK is set as i/p to AIC3204(Slave)
    AIC3204_rset(  4, 0x07 );     // PLLCLK <- BCLK and CODEC_CLKIN <-PLL CLK
    AIC3204_rset(  6, 0x20 );     // PLL setting: J = 32
    AIC3204_rset(  7, 0 );        // PLL setting: HI_BYTE(D)
    AIC3204_rset(  8, 0 );        // PLL setting: LO_BYTE(D)
    // For 48 KHz sampling
    AIC3204_rset(  5, 0x92 );     // PLL setting: Power up PLL, P=1 and R=2
    AIC3204_rset( 13, 0x00 );     // DOSR = 128 decimal DAC oversamppling
    AIC3204_rset( 14, 0x80 );     // DOSR = 128 decimal or 0x0080
    AIC3204_rset( 20, 0x80 );     // AOSR for AOSR = 128 decimation filters 1 to 6
    AIC3204_rset( 11, 0x84 );     // Power up NDAC and set NDAC value to 4
    AIC3204_rset( 12, 0x82 );     // Power up MDAC and set MDAC value to 2
    AIC3204_rset( 18, 0x84 );     // Power up NADC and set NADC value to 4
    AIC3204_rset( 19, 0x82 );     // Power up MADC and set MADC value to 2
    // DAC ROUTING and Power Up
    AIC3204_rset(  0, 0x01 );     // Select page 1
    AIC3204_rset( 12, 0x08 );     // LDAC AFIR routed to HPL
```

```
    AIC3204_rset( 13, 0x08 );     // RDAC AFIR routed to HPR
    AIC3204_rset(  0, 0x00 );     // Select page 0
    AIC3204_rset( 64, 0x02 );     // Left vol=right vol
    AIC3204_rset( 65, 0x00 );     // Left DAC gain to 0dB VOL; Right tracks Left
    AIC3204_rset( 63, 0xd4 );     // Power up left,right paths and set channel
    AIC3204_rset(  0, 0x01 );     // Select page 1
    AIC3204_rset( 16, 0x00 );     // Unmute HPL , 0dB gain
    AIC3204_rset( 17, 0x00 );     // Unmute HPR , 0dB gain
    AIC3204_rset(  9, 0x30 );     // Power up HPL,HPR
    AIC3204_rset(  0, 0x00 );     // Select page 0
    c5515_wait( 500 );            // Wait
    // ADC ROUTING and Power Up
    AIC3204_rset(  0, 0x01 );     // Select page 1
    AIC3204_rset( 52, 0x0C );     // STEREO 1 Jack
    // IN2_L to LADC_P through 40 kohm
    AIC3204_rset( 55, 0x0C );     // IN2_R to RADC_P through 40 kohmm
    AIC3204_rset( 54, 0x03 );     // CM_1 (common mode) to LADC_M through 40 kohm
    AIC3204_rset( 57, 0xC0 );     // CM_1 (common mode) to RADC_M through 40 kohm
    AIC3204_rset( 59, 0x00 );     // MIC_PGA_L unmute
    AIC3204_rset( 60, 0x00 );     // MIC_PGA_R unmute
    AIC3204_rset(  0, 0x00 );     // Select page 0
    AIC3204_rset( 81, 0xc0 );     // Powerup Left and Right ADC
    AIC3204_rset( 82, 0x00 );     // Unmute Left and Right ADC
    AIC3204_rset( 0,  0x00 );
    c5515_wait( 200 );  // Wait
    return 0;
}
/* --- Программа конфигурации «с микрофона на наушники»-------------------- */
Int16 aic3204_mic( )
{
    AIC3204_rset(  0, 0x00 );     // Select page 0
    AIC3204_rset(  1, 0x01 );     // Reset codec
    AIC3204_rset(  0, 0x01 );     // Select page 1
    AIC3204_rset(  1, 0x08 );     // Disable crude AVDD generation from DVDD
    AIC3204_rset(  2, 0x00 );     // Enable Analog Blocks
    // PLL and Clocks config and Power Up
    AIC3204_rset(  0, 0x00 );     // Select page 0
    AIC3204_rset( 27, 0x00 );     // BCLK and WCLK is set as i/p to AIC3204(Slave)
    AIC3204_rset(  4, 0x07 );     // PLLCLK <- BCLK and CODEC_CLKIN <-PLL CLK
    AIC3204_rset(  6, 0x20 );     // PLL setting: J = 32
    AIC3204_rset(  7, 0 );        // PLL setting: HI_BYTE(D)
    AIC3204_rset(  8, 0 );        // PLL setting: LO_BYTE(D)
    // For 48 KHz sampling
    AIC3204_rset(  5, 0x92 );     // PLL setting: Power up PLL, P=1 and R=2
    AIC3204_rset( 13, 0x00 );     // Hi_Byte(DOSR) = 128 DAC oversamppling
    AIC3204_rset( 14, 0x80 );     // Lo_Byte(DOSR) DOSR = 128
    AIC3204_rset( 20, 0x80 );     // AOSR = 128 decimal for filters 1 to 6
    AIC3204_rset( 11, 0x84 );     // Power up NDAC and set NDAC value to 4
    AIC3204_rset( 12, 0x82 );     // Power up MDAC and set MDAC value to 2
    AIC3204_rset( 18, 0x84 );     // Power up NADC and set NADC value to 4
    AIC3204_rset( 19, 0x82 );     // Power up MADC and set MADC value to 2
    // DAC ROUTING and Power Up
    AIC3204_rset(  0, 0x01 );     // Select page 1
    AIC3204_rset( 12, 0x08 );     // LDAC AFIR routed to HPL
    AIC3204_rset( 13, 0x08 );     // RDAC AFIR routed to HPR
    AIC3204_rset(  0, 0x00 );     // Select page 0
    AIC3204_rset( 64, 0x02 );     // Left vol=right vol
    AIC3204_rset( 65, 0x00 );     // Left DAC gain to 0dB VOL; Right tracks Left
    AIC3204_rset( 63, 0xd4 );     // Power up left,right and set channel
    AIC3204_rset(  0, 0x01 );     // Select page 1
    AIC3204_rset( 16, 0x06 );     // Unmute HPL , 6dB gain
    AIC3204_rset( 17, 0x06 );     // Unmute HPR , 6dB gain
    AIC3204_rset(  9, 0x30 );     // Power up HPL,HPR
    AIC3204_rset(  0, 0x00 );     // Select page 0
    c5515_wait( 500 );            // Wait
    // ADC ROUTING and Power Up
```

```
    AIC3204_rset(  0, 0x01 );     // Select page 1
    AIC3204_rset( 51, 0x40 );     // SetMICBIAS
    AIC3204_rset( 52, 0xc0 );     // STEREO 1 Jack
    // IN2_L to LADC_P through 40 kohm
    AIC3204_rset( 55, 0xc0 );     // IN2_R to RADC_P through 40 kohmm
    AIC3204_rset( 54, 0x03 );     // CM_1 (common mode) to LADC_M through 40 kohm
    AIC3204_rset( 57, 0xc0 );     // CM_1 (common mode) to RADC_M through 40 kohm
    AIC3204_rset( 59, 0x5f );     // MIC_PGA_L unmute
    AIC3204_rset( 60, 0x5f );     // MIC_PGA_R unmute
    AIC3204_rset(  0, 0x00 );     // Select page 0
    AIC3204_rset( 81, 0xc0 );     // Powerup Left and Right ADC
    AIC3204_rset( 82, 0x00 );     // Unmute Left and Right ADC
    AIC3204_rset( 0,  0x00 );
    c5515_wait( 200 );            // Wait
    return 0;
}
/* ---------------------------------------------------------------------------- */
Int16 aic3204_stereo_in1()
{
    AIC3204_rset(  0, 0x00 );     // Select page 0
    AIC3204_rset(  1, 0x01 );     // Reset codec
    AIC3204_rset(  0, 0x01 );     // Select page 1
    AIC3204_rset(  1, 0x08 );     // Disable crude AVDD generation from DVDD
    AIC3204_rset(  2, 0x00 );     // Enable Analog Blocks
    // PLL and Clocks config and Power Up
    AIC3204_rset(  0, 0x00 );     // Select page 0
    AIC3204_rset( 27, 0x00 )      // BCLK and WCLK is set as i/p to AIC3204(Slave)
    AIC3204_rset(  4, 0x07 );     // PLLCLK <- BCLK and CODEC_CLKIN <-PLL CLK
    AIC3204_rset(  6, 0x20 );     // PLL setting: J = 32
    AIC3204_rset(  7, 0 );        // PLL setting: HI_BYTE(D)
    AIC3204_rset(  8, 0 );        // PLL setting: LO_BYTE(D)
    // For 48 KHz sampling
    AIC3204_rset(  5, 0x92 );     // PLL setting: Power up PLL, P=1 and R=2
    AIC3204_rset( 13, 0x00 );     // Hi_Byte(DOSR) = 128 decimal DAC oversamppling
    AIC3204_rset( 14, 0x80 );     // Lo_Byte(DOSR) = 128 decimal or 0x0080
    AIC3204_rset( 20, 0x80 );     // AOSR = 128 decimal for filters 1 to 6
    AIC3204_rset( 11, 0x84 );     // Power up NDAC and set NDAC value to 4
    AIC3204_rset( 12, 0x82 );     // Power up MDAC and set MDAC value to 2
    AIC3204_rset( 18, 0x84 );     // Power up NADC and set NADC value to 4
    AIC3204_rset( 19, 0x82 );     // Power up MADC and set MADC value to 2
    // DAC ROUTING and Power Up
    AIC3204_rset(  0, 0x01 );     // Select page 1
    AIC3204_rset( 12, 0x08 );     // LDAC AFIR routed to HPL
    AIC3204_rset( 13, 0x08 );     // RDAC AFIR routed to HPR
    AIC3204_rset(  0, 0x00 );     // Select page 0
    AIC3204_rset( 64, 0x02 );     // Left vol=right vol
    AIC3204_rset( 65, 0x00 );     // Left DAC gain to 0dB VOL; Right tracks Left
    AIC3204_rset( 63, 0xd4 );     // Power up left,right and set channel
    AIC3204_rset(  0, 0x01 );     // Select page 1
    AIC3204_rset( 16, 0x00 );     // Unmute HPL , 0dB gain
    AIC3204_rset( 17, 0x00 );     // Unmute HPR , 0dB gain
    AIC3204_rset(  9, 0x30 );     // Power up HPL,HPR
    AIC3204_rset(  0, 0x00 );     // Select page 0
    c5515_wait( 500 );            // Wait
    // ADC ROUTING and Power Up
    AIC3204_rset(  0, 0x01 );     // Select page 1
    AIC3204_rset( 52, 0x30 );     // STEREO 1 Jack
    // IN2_L to LADC_P through 40 kohm
    AIC3204_rset( 55, 0x30 );     // IN2_R to RADC_P through 40 kohmm
    AIC3204_rset( 54, 0x03 );     // CM_1 (common mode) to LADC_M through 40 kohm
    AIC3204_rset( 57, 0xC0 );     // CM_1 (common mode) to RADC_M through 40 kohm
    AIC3204_rset( 59, 0x0f );     // MIC_PGA_L unmute
    AIC3204_rset( 60, 0x0f );     // MIC_PGA_R unmute
    AIC3204_rset(  0, 0x00 );     // Select page 0
    AIC3204_rset( 81, 0xc0 );     // Powerup Left and Right ADC
    AIC3204_rset( 82, 0x00 );     // Unmute Left and Right ADC
```

```
    AIC3204_rset( 0,  0x00 );
    c5515_wait( 200 );            // Wait
    return 0;
}
```

Приложение J

Главный модуль main.c

```
#include "C5515.h"
#include "gpio.h"
#include "i2c.h"
#include "i2s.h"
#include "stdio.h"

#define AIC3204_I2C_ADDR  0x18
#define Rcv               0x08
#define Xmit              0x20

extern Int16 aic3204_mic();
extern Int16 aic3204_stereo_in1();
extern Int16 aic3204_sin();

Int16 sinetable[48] = {
     0x0000, 0x10b4, 0x2120, 0x30fb, 0x3fff, 0x4dea, 0x5a81, 0x658b, 0x6ed8,
     0x763f, 0x7ba1, 0x7ee5, 0x7ffd, 0x7ee5, 0x7ba1, 0x76ef, 0x6ed8, 0x658b,
     0x5a81, 0x4dea, 0x3fff, 0x30fb, 0x2120, 0x10b4, 0x0000, 0xef4c, 0xdee0,
     0xcf06, 0xc002, 0xb216, 0xa57f, 0x9a75, 0x9128, 0x89c1, 0x845f, 0x811b,
     0x8002, 0x811b, 0x845f, 0x89c1, 0x9128, 0x9a76, 0xa57f, 0xb216, 0xc002,
     0xcf06, 0xdee0, 0xef4c };

Int16 j, i = 0;
Int16 sample, left, right;

Int16 chorus_voices = 2;
Int16 chorus_width = 4040;

//отсчеты после поимененого эффекта
Int16 out_left = 0, out_right = 0;

Int16 array_size = 4800;

Int16 arrayLeft_delay[4800] = {0};
Int16 arrayRight_delay[4800] = {0};

Int16 index_delay = 0;                    // вычисляемая задержка
Int16 accumulator_left = 0;               // аккумулятор для левого сигнала
Int16 accumulator_right = 0;              // аккумулятор для правого сигнала
/* ------------------------------------------------------------------------ */
void effect(Int16 left_ch, Int16 right_ch, Int16 voices, Int16 chorus_width)
{
     Int16 v = 0;                          //счетчик голосов
     arrayLeft_delay[sample] = left_ch;    // буфер задержки левого канала
     arrayRight_delay[sample] = right_ch;  // буфер задержки правого канала
     out_left = left_ch;
     out_right = right_ch;
     for (v = 1; v < voices; v++)
     {
          // обнуляем аккумуляторы
          accumulator_left = 0;
          accumulator_right = 0;
          // находим номер отсчета в буфере задержки
          index_delay = (array_size + sample - (chorus_width*v)) % array_size;
          // суммируем отсчеты
          accumulator_left += arrayLeft_delay[index_delay];
          accumulator_right += arrayRight_delay[index_delay];
     }
     // суммируем с текущим значением отсчета
     out_left += accumulator_left;
     out_right += accumulator_right;
```

```
    // делим на количество голосов для того чтобы не было перегрузки звука
    out_left = out_left / voices;
    out_right = out_right / voices;
}
/* ------------------------------------------------------------------------ */
void main(void)
{
    /* Initialize BSL */
    c5515_init();
    /* Configure Parallel Port */
    SYS_EXBUSSEL &= ~0x7000;          //
    SYS_EXBUSSEL |= 0x1000;           // Configure Parallel Port for I2S2
    /* Configure Serial Port */
    SYS_EXBUSSEL &= ~0x0C00;          //
    SYS_EXBUSSEL |= 0x0400;           // Serial Port mode 1 (I2S1 and GP[11:10]).
    c5515_GPIO_init();
    c5515_GPIO_setDirection(GPIO10, GPIO_OUT);
    c5515_GPIO_setOutput(GPIO10, 1);  // Take AIC3201 chip out of reset
    I2C_init();                       // Initialize I2C
    /* I2S settings */
    I2S2_SRGR = 0x0015;
    I2S2_ICMR = 0x0028;               // Enable interrupts
    I2S2_CR = 0x8012;                 // 16-bit word, Master, enable I2C
// Режим генерация тона
/*
     aic3204_sin();
     for ( i = 0 ; i < 5 ; i++ )
     {
         for ( j = 0 ; j < 1000 ; j++ )
         {
             for ( sample = 0 ; sample < 48 ; sample++ )
             {
                 while((Xmit & I2S2_IR) == 0);  // Wait for interrupt
                 I2S2_W0_MSW_W = (sinetable[sample]) ;
                 I2S2_W0_LSW_W = 0;
                 I2S2_W1_MSW_W = (sinetable[sample]) ;
                 I2S2_W1_LSW_W = 0;
             }
         }
     }
*/
// Режим «с микрофона на наушники»
/*
    aic3204_mic();
*/

// Режим стерео-входа
    aic3204_stereo_in1();
    // необходимо чтобы 5*1000*sample было кратно 240 000
    for (i = 0; i < 5; i++)
    {
        for (j = 0; j < 1000; j++)
        {
            for (sample = 0; sample < array_size; sample++) {
                // Wait for receive interrupt to be pending
                while ((Rcv & I2S2_IR) == 0);
                left = I2S2_W0_MSW_R;
                right = I2S2_W1_MSW_R;
                // функция эффекта
                effect(left, right, chorus_voices, chorus_width);
                // Wait for receive interrupt to be pending
                while ((Xmit & I2S2_IR) == 0);
                I2S2_W0_MSW_W = out_left;
                I2S2_W1_MSW_W = out_right;
        }
    }
```

```
    I2S0_CR = 0x00;
    c5515_GPIO_setOutput(GPIO26, 0);
    SW_BREAKPOINT;
}
```