ЦИФРОВОЙ СИНТЕЗАТОР 1508ПЛ8Т

УПРАВЛЕНИЕ МИКРОСХЕМОЙ ЧЕРЕЗ ПОРТ SPI

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ

Данный документ описывает работу с микросхемой 1508ПЛ8Т, подключенной к процессору серии «Мультикор» (на примере процессора 1892ВМ10Я) через порт SPI.

Приведены схемы подключения микросхем 1508ПЛ8Т и 1892ВМ10Я, отладочных модулей, настройка порта SPI процессора, функции для доступа к регистрам микросхемы 1508ПЛ8Т, настройка режимов синтеза гармонического сигнала, переключения между профилями, линейно-частотной модуляции (ЛЧМ). Также приведены формулы расчета частот и длительностей фаз.

2. СХЕМЫ ПОДКЛЮЧЕНИЯ

2.1 Схема подключения микросхем 1508ПЛ8Т и 1892ВМ10Я

В процессорах серии «Мультикор» SPI реализован в составе порта MFBSP. Порт MFBSP имеет десять выводов, из которых в режиме SPI используются шесть. Оставшиеся четыре вывода могут быть использованы как GPIO. Для подключения микросхемы 1508ПЛ8Т задействуются четыре вывода SPI из шести.

При работе с микросхемой 1508ПЛ8Т через порт SPI необходимо обеспечить на ее входах CSn, RDn, WRn высокий (неактивный) уровень.

Уровни логической единицы/нуля цифровых входов/выходов у микросхем 1508ПЛ8Т и 1892ВМ10Я совместимы, поэтому дополнительной обвязки при подключении не требуется.

Схема подключения приведена на рисунке 2.1.

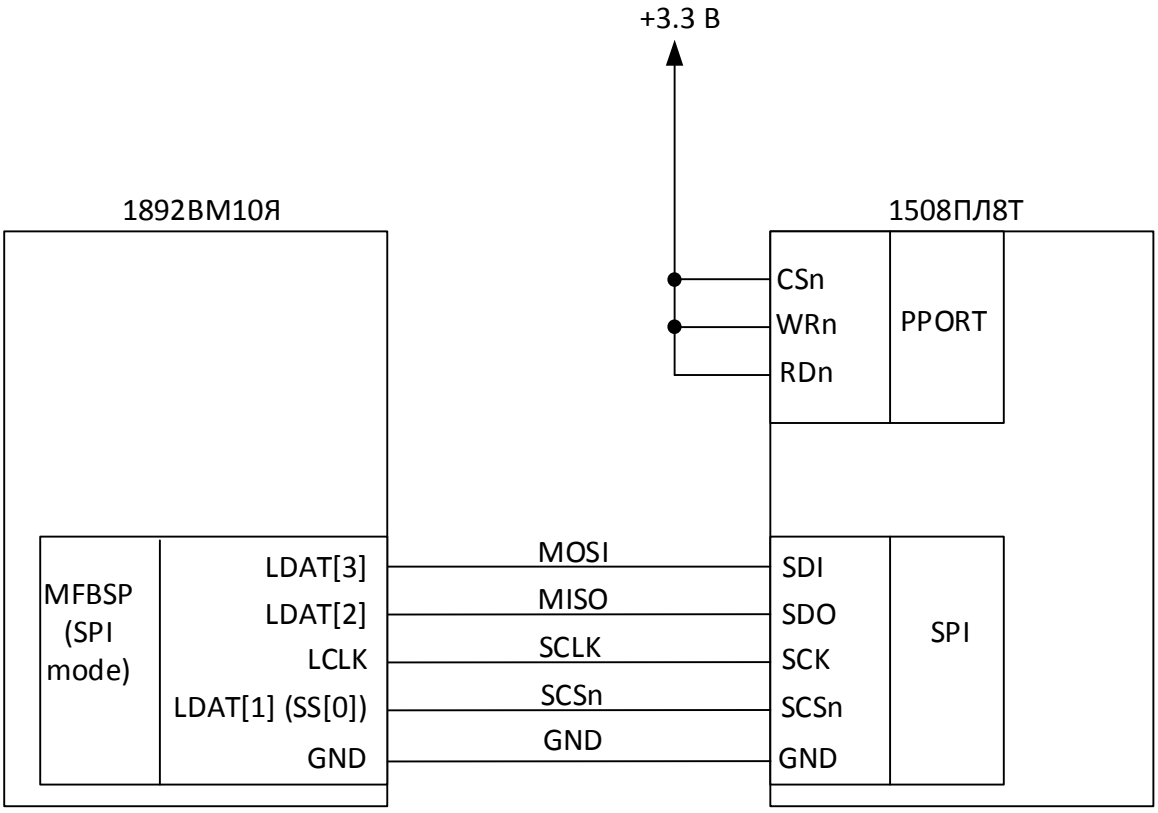

Рисунок 2.1. Схема подключения микросхемы 1508ПЛ8Т к процессору 1892ВМ10Я через порт SPI

2.2 Схема подключения модулей

Для отладки и макетирования могут быть использованы отладочный модуль NVCom-02TEM-3U для микросхемы 1892ВМ10Я (производство АО НПЦ «ЭЛВИС») и отладочная плата для микросхемы 1508ПЛ8Т (производство ООО «Радиокомп»).

Схема подключения модулей приведена на Рисунок 2.2. Положение переключателей на плате см. в руководстве пользователя.

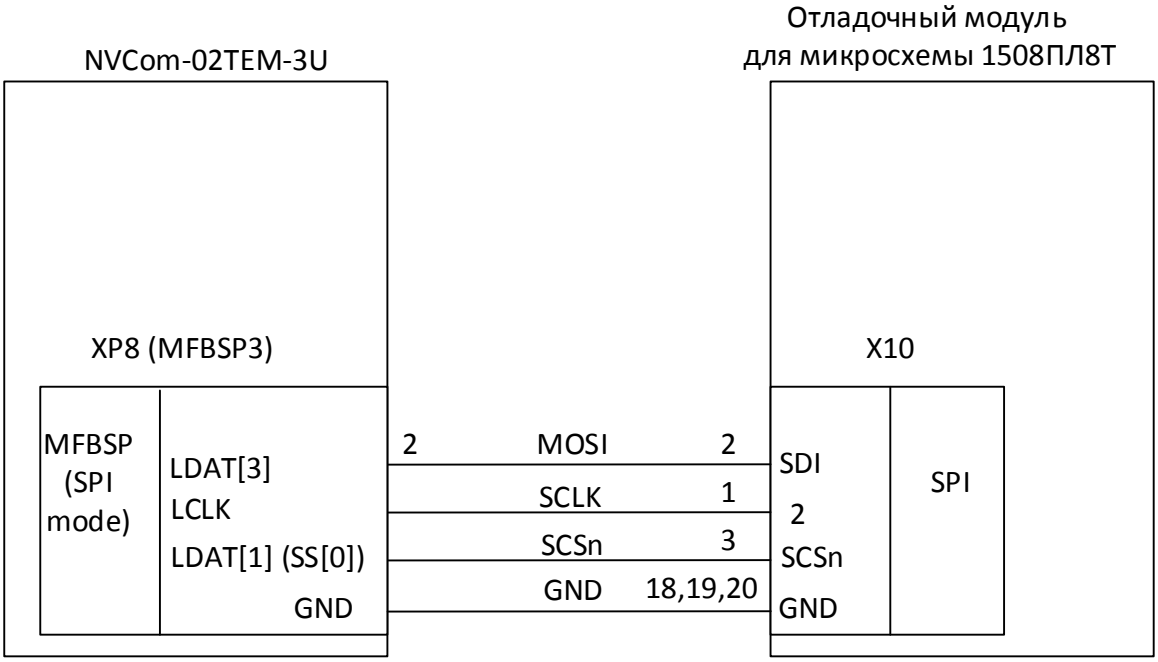

Рисунок 2.2

На отладочной плате производства ООО «Радиокомп» сигнал SDO (MISO) не выведен. При отсутствии этого вывода читать регистры микросхему 1508ПЛ8Т по SPI не представляется возможным, но ничего не мешает производить в них запись.

3. НАСТРОЙКА ПОРТА MFBSP ПРОЦЕССОРА 1892ВМ10Я

При подготовке данного документа использовалось подключение, в котором синтезатор 1508ПЛ8Т был соединен с портом MFBSP3 процессора 1892ВМ10Я, поэтому приведенный ниже код работает именно с этим портом. При подключении к любому другому порту MFBSP данный код останется неизменным, за исключением номера порта.

```
void SPI_Init() {
     MFBSP3.CSR.bits.SPI_I2S_EN = 1;
//Включен режим I2S/SPI. Появляется 201 в TX

     MFBSP3.DIR.bits.TD_DIR = 1;
// MOSI- выход. MISO - вход по умолчанию

     MFBSP3.DIR.bits.TCS_DIR = 1;// SS[0] - выход
     MFBSP3.DIR.bits.TCLK_DIR =1;// TSCK - выход
     MFBSP3.DIR.bits.RCLK_DIR =0;// RSCK - формируется приемником
     MFBSP3.DIR.bits.RD_DIR = 0;    // MISO - вход по умолчанию

// настройка передатчика
     MFBSP3.TCTR.bits.SS_0 = 1;      // Используется SS[0] в качестве
CS
     MFBSP3.TCTR.bits.TWORDLEN = 23; // длина слова - 24 разряда
     MFBSP3.TCTR.bits.TMBF = 1;      // старшим битом вперед
     MFBSP3.TCTR.bits.TWORDCNT = 0; // одно слово во фрейме – чтобы
сигнал SS[0]
                                    // переключался в единицу после
передачи каждого слова
              MFBSP3.TCTR.bits.TNEG = 1;      // TNEG=1, TDEL=1. При
данном сочетании выдача данных
              MFBSP3.TCTR.bits.TDEL = 1;      // происходит по заднему
фронту TSCK
                                                              //
исходное состояние TSCK перед началом передачи - высокое.
              MFBSP3.TCTR.bits.TMODE = 1;     // режим SPI (TMODE=0 –
режим I2S)
              MFBSP3.TCTR.bits.TPACK = 0;

              MFBSP3.TCTR_RATE.data = (FREQ/(2*SPI_FRQ)) - 1; //
частота работы SPI

// настройка приемника
              MFBSP3.RCTR.bits.RWORDLEN = 23;// длина принимаемого
слова - 24 разряда
              MFBSP3.RCTR.bits.RMBF = 1;       // старшим битом вперед
             MFBSP3.RCTR.bits.RWORDCNT=0; // 1 слово во фрейме

// RSCK и SS[0] - берутся от передатчика
     MFBSP3.RCTR.bits.RCS_CP = 1;
     MFBSP3.RCTR.bits.RCLK_CP = 1;

     MFBSP3.RCTR.bits.RMODE = 1;    // режим SPI
```

```
     MFBSP3.RCTR.bits.RNEG = 1;//
     MFBSP3.RCTR.bits.RDEL = 1; // приеме данных происходит по
//переднему фронту RSCK
     // а исходное состояние RSCK перед началом приема - высокое.
// RSCK и SS[0] - берутся от передатчика

     MFBSP3.RCTR.bits.REN = 1;     // приемник включен
     MFBSP3.RSTART.data = 1;
     MFBSP3.TSTART.data = 1;
     MFBSP3.TCTR.bits.TEN = 1;       // TEN - передатчик включен
// включение передатчика необходимо делать после всех остальных
настроек
}
```

4. ФУНКЦИИ ЗАПИСИ И ЧТЕНИЯ ПО SPI

Для отправки данных в порт SPI необходимо совершить запись слова в регистр TX_MFBSP соответствующего порта. Записывать необходимо 32-разрядное слово. Поскольку в настройках SPI указана длина слова 24 бита, старшие 8 разрядов записанного 32-разрядного слова будут отброшены передатчиком, а младшие 24 – отправлены в линию MOSI. При передаче слова передатчик SPI автоматически установит в ноль сигнал SS[0] (при настройках, использованных в указанной функции инициализации). После записи слова в регистр TX_MFBSP необходимо дождаться, пока передающий буфер порта MFBSP освободится (то есть, слово физически будет отправлено в канал).

При передаче порт SPI формирует 24 импульса частоты TCLK (SCK – в терминах микросхемы 1508ПЛ8Т). По этим же 24 импульсам сдвиговый регистр микросхемы 1508ПЛ8Т выдвигает свое значение на линию SDO (MISO). Приемник SPI микросхемы 1892ВМ10Я фиксирует состояния линии MISO на каждом импульсе SCK и полученное слово сохраняет в буфер порта MFBSP, доступный через чтение регистра RX_MFBSP. Таким образом, при записи каждого слова в SPI, в буфере порта MFBSP появляется новое слово данных.

4.1 Функция записи в адресное пространство 1508ПЛ8Т

```
void SYNT1_write(unsigned int addr, unsigned int val){
// Функция записи в адресное пространство DDS

     MFBSP3.TX.data = COMMAND_SETA|(addr & 0xFFFF);
     while (!(MFBSP3.TSR.bits.TBE));

     MFBSP3.TX.data = COMMAND_WR|(val & 0xFFFF);
     while (!(MFBSP3.TSR.bits.TBE));

}
```

4.2 Функция чтения из регистров адресного пространства 1508ПЛ8Т

```
unsigned int SYNT1_read(unsigned int addr){

     MFBSP3.TX.data = COMMAND_SETAFT|(addr & 0xFFFF);

     while (!(MFBSP3.TSR.bits.TBE));
      while (!(MFBSP3.CSR.data & 1<<4));     //ожидание получения
новых данных
           return (MFBSP3.RX.data & 0xFFFF); // приходят данные от
предыдущей команды, это не ответ на команду чтения
}
```

Приведенных выше функций достаточно, чтобы полноценно работать с микросхемой 1508ПЛ8Т через последовательный порт SPI.

5. ПРИМЕРЫ РАБОТЫ С МИКРОСХЕМОЙ 1508ПЛ8Т

Код, приведенный ниже, демонстрирует пример записи и чтения регистров синтезатора 1508ПЛ8Т.

Регистры микросхемы 1508ПЛ8Т прописаны в заголовочном файле dds.h, в котором его названия совпадают с наименованиями в техническом описании.

```
// файл со ссылками на макроопределения регистров, в том числе нужных
нам CLK_EN, DIR_MFBSP1, GPIO_DR1
#include "multicore/nvcom02t.h"
// файл со ссылками адресов регистров микросхемы 1508ПЛ8Т и команд
#include "multicore/dds.h"

// установка длительности паузы в тактах частоты CPU, 1 секунда
#define TIMEOUT 100000000

// частота работы SPI в Гц
unsigned int SPI_FRQ = 6000000;
// частота работы процессора в герцах
#define FREQ 240000000

unsigned int SYNT1_read(unsigned int addr){

      MFBSP3.TX.data = COMMAND_SETAFT|(addr & 0xFFFF);

      while (!(MFBSP3.TSR.bits.TBE));
      while (!(MFBSP3.CSR.data & 1<<4)); //ожидание получения новых
//данных
      return (MFBSP3.RX.data & 0xFFFF); // приходят данные от
//предыдущей команды, это не ответ на команду чтения
}


void SYNT1_write(unsigned int addr, unsigned int val){ // Функция
//записи в адресное пространство DDS

      MFBSP3.TX.data = COMMAND_SETA|(addr & 0xFFFF);
      while (!(MFBSP3.TSR.bits.TBE));

      MFBSP3.TX.data = COMMAND_WR|(val & 0xFFFF);
      while (!(MFBSP3.TSR.bits.TBE));

}

void SYNT1_RESET(){
      SYNT1_write(0x0,0x78); //RESET
}

void SPI_Init() {
      MFBSP3.CSR.bits.SPI_I2S_EN = 1; // включен режим I2S/SPI
```

```
//Появляется 201 в TX

     MFBSP3.DIR.bits.TD_DIR = 1; // MOSI- выход. MISO - //вход по
умолчанию
     MFBSP3.DIR.bits.TCS_DIR = 1;   // SS[0] - выход
     MFBSP3.DIR.bits.TCLK_DIR =1;   // TSCK - выход
     MFBSP3.DIR.bits.RCLK_DIR =0;   // RSCK - формируется приемником
     MFBSP3.DIR.bits.RD_DIR = 0;    // MISO - вход по умолчанию
// настройка передатчика
     MFBSP3.TCTR.bits.SS_0 = 1; //Используется SS[0] в качестве CS
     MFBSP3.TCTR.bits.TWORDLEN = 23; // длина слова - 24 разряда
     MFBSP3.TCTR.bits.TMBF = 1; // старшим битом вперед
     MFBSP3.TCTR.bits.TWORDCNT = 0;// одно слово во фрейме – чтобы
//сигнал SS[0]
// переключался в единицу после передачи каждого слова
     MFBSP3.TCTR.bits.TNEG = 1;      // TNEG=1, TDEL=1. При данном
сочетании выдача данных
     MFBSP3.TCTR.bits.TDEL = 1;      // происходит по заднему фронту
TSCK
     // исходное состояние TSCK перед началом передачи - высокое.
     MFBSP3.TCTR.bits.TMODE = 1; // режим SPI (TMODE=0 – режим I2S)
     MFBSP3.TCTR.bits.TPACK = 0;

     MFBSP3.TCTR_RATE.data = (FREQ/(2*SPI_FRQ)) - 1;//частота работы
//SPI

// настройка приемника
     MFBSP3.RCTR.bits.RWORDLEN = 23;// длина принимаемого слова - 24
разряда
     MFBSP3.RCTR.bits.RMBF = 1;    // старшим битом вперед
     MFBSP3.RCTR.bits.RWORDCNT=0; // 1 слово во фрейме

// RSCK и SS[0] - берутся от передатчика
     MFBSP3.RCTR.bits.RCS_CP = 1;
     MFBSP3.RCTR.bits.RCLK_CP = 1;

     MFBSP3.RCTR.bits.RMODE = 1;    // режим SPI

     MFBSP3.RCTR.bits.RNEG = 1; // RNEG=1, RDEL=1. При данном
//сочетании захват данных при
     MFBSP3.RCTR.bits.RDEL = 1; // приеме данных происходит по
//переднему фронту RSCK
// а исходное состояние RSCK перед началом приема - высокое.
// RSCK и SS[0] - берутся от передатчика

     MFBSP3.RCTR.bits.REN = 1;     // приемник включен
     MFBSP3.RSTART.data = 1;
     MFBSP3.TSTART.data = 1;
     MFBSP3.TCTR.bits.TEN = 1; // TEN - передатчик включен
     // Такой порядок включения рекомендован разработчиком
}

int main() {

     unsigned int rcv_array[12];
     unsigned int i;
```

```
      SPI_Init();

      rcv_array[0]=SYNT1_read(SYNC); // в этот элемент массива
//попадёт значение 0x201, DEVID
      rcv_array[1]=SYNT1_read(CTR); // 0x4800 значение SYNC - результат
//предыдущего чтения (SYNC) из сдвигового регистра
      rcv_array[2]=SYNT1_read(DEVID);// 0x2 (CTR)
      rcv_array[3]=SYNT1_read(SYNC);// 0x201 (DEVID)
      // Запись и чтение из TC_L
      SYNT1_write(SEL_REG,0xCFCF);// после функции записи (команды SETA
//и WR) DDS дважды выдаст со сдвигового регистра наружу
      // значение адреса регистра, в который производится запись
      // в данном случае адрес SEL_REG (0x2)
      rcv_array[4]=SYNT1_read(SEL_REG);// 0x4800 (SYNC)
      rcv_array[5]=SYNT1_read(SYNC);   // 0x2 см функцию SYNT1_write
      rcv_array[6]=SYNT1_read(SYNC);   // 0x2 см функцию SYNT1_write
      rcv_array[7]=SYNT1_read(SYNC);   //  0xCFCF - значение,
//прочитанное из регистра SEL_REG окажется в этом элементе массива
      SYNT1_RESET();                   // программный сброс
      rcv_array[8]=SYNT1_read(SEL_REG);// 0x4800 (SYNC)
      rcv_array[9]=SYNT1_read(CTR);    // 0x4800 (SYNC)
      rcv_array[10]=SYNT1_read(CTR);   // 0x4800 (SYNC)
      rcv_array[11]=SYNT1_read(CTR);   // 0x0 значение SEL_REG=0  как
результат программного сброса


      while (1) ;

}
```

В результате этих действий в массиве rcv_array должны оказаться следующие значения:

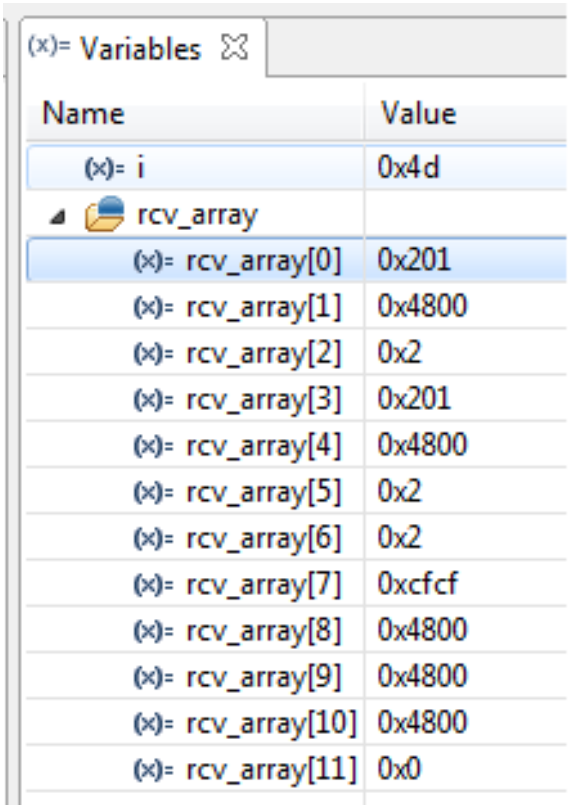

Рисунок 5.1. Результат выполнения примера записи и чтения из регистров

1. В результате первого чтения регистра SYNC в элементе rcv_array(0) оказывается значение 0x201 так как это результат первоначального обращения к микросхеме 1508ПЛ8Т, по первому значению она выдает DEVID.

2. В результате следующего чтения приходит значение 0x4800, что является значением регистра SYNC по сбросу. То есть, здесь и при дальнейших чтениях в элементы массива rcv_array[1] - rcv_array[4] попадает результат предыдущего чтения.

3. В rcv_array[5], rcv_array[6] – значение адреса регистра SEL_REG, так как предыдущей командой в него происходила запись и адрес именно этого регистра дважды попал в сдвиговый регистр микросхемы согласно таблице 6.1 Технического описания.

4. В rcv_array[7] – прочитанное значение из регистра SEL_REG, записанное нами ранее.

5. В rcv_array[8], rcv_array[9], rcv_array[10] – результат троекратного чтения регистра SYNC, и так как в него ничего записано не было, читается значение 0x4800 – значение по умолчанию.

6. В rcv_array[11] - значение SEL_REG=0 как результат программного сброса.

Пример для MCStudio4 доступен по ссылке http://multicore.ru/mc/software/test_synt1.zip

5.1 Запись одного профиля и демонстрация синтеза гармонического сигнала (синусоиды)

В этом разделе продемонстрирован пример, позволяющий получить синусоиду как на рисунке ниже.

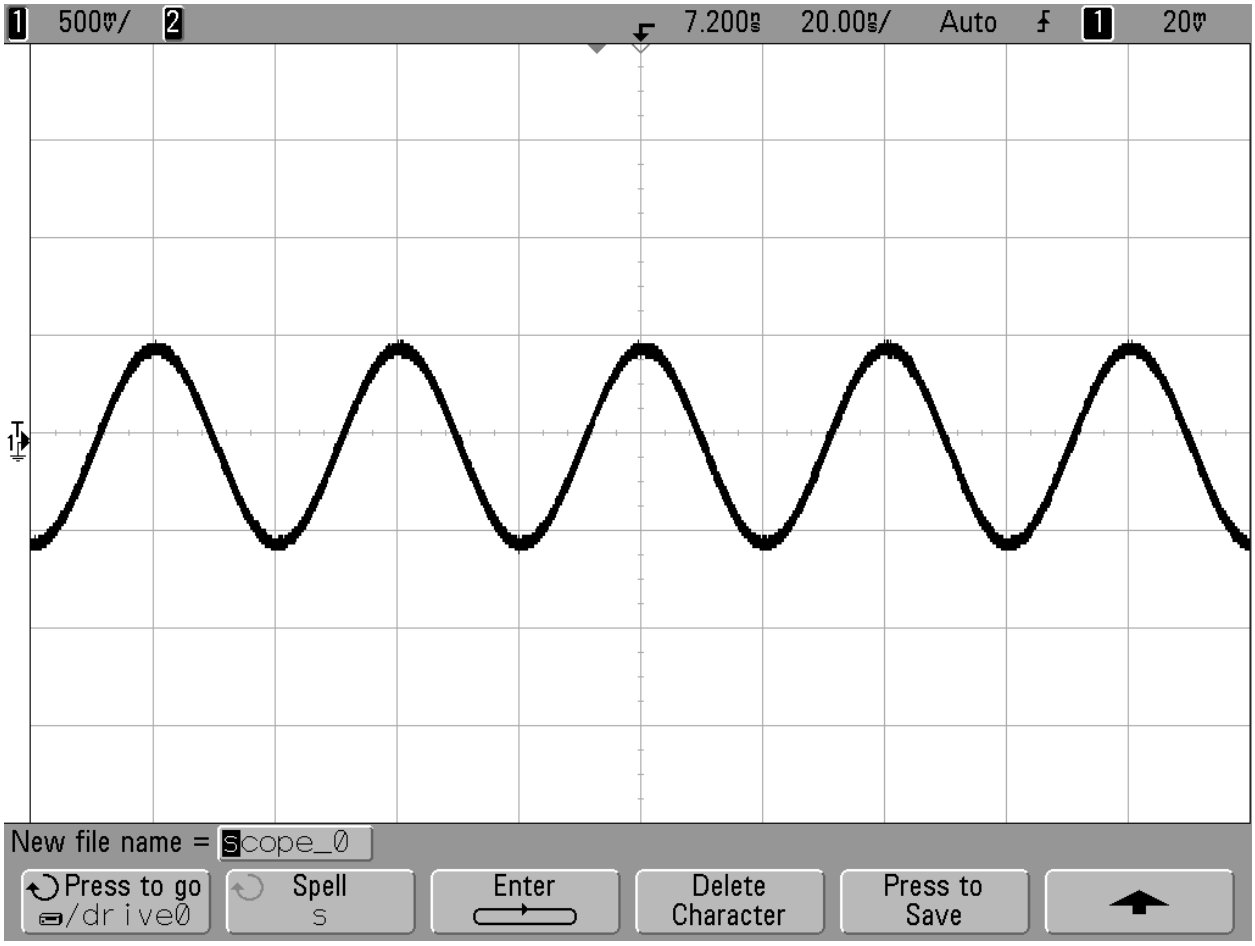

Рисунок 5.2. Гармонический сигнал 50МГц

5.1.1 Расчет выходной частоты синтезатора

Выходную частоту задают регистры Ch*x*_dPh*y*_H, Ch*x*_dPh*y*_M, Ch*x*_dPh*y*_L.

Значение выходной частоты определяется соотношением:

$$F_{out} = \frac{F_H * 2^{32}\ F_M * 2^{16}\ F_L}{2^{48}} * F_{clk},$$

где :

F_{out} — синтезируемая частота,

F_{clk} — тактовая частота ЦАП,

F_H= Ch*x*_dPh*y*_H,

F_M= Chx_dPhy_M,

F_L = Chx_dPhy_L.

Чтобы рассчитать значения регистров, нужно:

1. Выходную частоту F_{out} (ту, которую требуется получить) поделить на тактовую F_{clk}.
2. Результат умножить на 2^{48}.
3. Получившееся значение перевести в шестнадцатеричную систему исчисления.
4. Результат – это значения регистров Chx_dPhy_H, Chx_dPhy_M, Chx_dPhy_L друг за другом.

Например,

Нам необходимо получить на выходе микросхемы частоту в 25МГц. Входная тактовая частота – 8000МГц.

1. 25/800= 0,03125
2. 0,03125*2^{48}= 8796093022208
3. 8796093022208_{10}=80000000000_{16}
4. Chx_dPhy_H=0x0800, Chx_dPhy_M=0x0000, Chx_dPhy_L=0x0000.

Пример синтеза гармонического сигнала 25МГц:

```
/*****************************************************************
 *                  harmonic signal synthesis
 *                      NVCom-02T-1508PL8T                       *
 *      Демонстрирует работу с микросхемой 1508PL8T по SPI:      *
 *      Запись одного профиля и                                  *
 *      демонстрация синтеза гармонического сигнала (синусоиды).*
 *      Результатом исполнения этого примера является
 *      демонстрация синусоиды 50МГц и амплитудой ~1В
 *      Выводы разъемов X10 и XP8                                *
 *      модулей 1508ПЛ8Т(Радиокомп) и NVCom-02TEM-3U             *
 *      должный быть соединены следующим образом:
 *      X10          XP8
 *      SCK          12 (LCLKn)
 *      SDI(MOSI)    4  (LDAT[3])
 *      SCS          2  SS_0(LDAT[1])
 *      S23(MISO)    3  (LDAT[2])
 *      GND          14 (GND)
 *                                                               *
 *****************************************************************/



// файл со ссылками на макроопределения регистров, в том числе нужных
нам CLK_EN, DIR_MFBSP1, GPIO_DR1
#include "multicore/nvcom02t.h"
// файл со ссылками адресов регистров микросхемы 1508ПЛ8Т и команд
#include "multicore/dds.h"

// установка длительности паузы в тактах частоты CPU, 1 секунда
```

```
#define TIMEOUT 100000000

// частота работы SPI в Гц
unsigned int SPI_FRQ = 6000000;
// частота работы процессора в герцах
#define FREQ 240000000

unsigned int SYNT1_read(unsigned int addr){

     MFBSP3.TX.data = COMMAND_SETAFT|(addr & 0xFFFF);

     while (!(MFBSP3.TSR.bits.TBE));
      while (!(MFBSP3.CSR.data & 1<<4));     //ожидание получения
новых данных
          return (MFBSP3.RX.data & 0xFFFF); // приходят данные от
предыдущей команды, это не ответ на команду чтения
}


void SYNT1_write(unsigned int addr, unsigned int val){ // Функция
записи в адресное пространство DDS

     MFBSP3.TX.data = COMMAND_SETA|(addr & 0xFFFF);
     while (!(MFBSP3.TSR.bits.TBE));

     MFBSP3.TX.data = COMMAND_WR|(val & 0xFFFF);
     while (!(MFBSP3.TSR.bits.TBE));

}

void SYNT1_RESET(){
     SYNT1_write(0x0,0x78); //RESET
}

void SPI_Init() {
             MFBSP3.CSR.bits.SPI_I2S_EN = 1; // включен режим I2S/SPI
Появляется 201 в TX

             MFBSP3.DIR.bits.TD_DIR = 1;    // MOSI- выход. MISO -
вход по умолчанию
             MFBSP3.DIR.bits.TCS_DIR = 1;   // SS[0] - выход
             MFBSP3.DIR.bits.TCLK_DIR =1;   // TSCK - выход
             MFBSP3.DIR.bits.RCLK_DIR =0;   // tiu RSCK - формируется
приемником (то есть, нами же)
             MFBSP3.DIR.bits.RD_DIR = 0;    // tiu MISO - вход по
умолчанию
// настройка передатчика
             MFBSP3.TCTR.bits.SS_0 = 1;      // Используется SS[0] в
качестве CS
             MFBSP3.TCTR.bits.TWORDLEN = 23; // длина слова - 24
разряда
             MFBSP3.TCTR.bits.TMBF = 1;      // старшим битом вперед
             MFBSP3.TCTR.bits.TWORDCNT = 0; // одно слово во фрейме –
чтобы сигнал SS[0]
                                  // переключался в единицу после
передачи каждого слова
```

```
            MFBSP3.TCTR.bits.TNEG = 1;      // TNEG=1, TDEL=1. При
данном сочетании выдача данных
            MFBSP3.TCTR.bits.TDEL = 1;      // происходит по заднему
фронту TSCK
                                                              //
исходное состояние TSCK перед началом передачи - высокое.
            MFBSP3.TCTR.bits.TMODE = 1;     // режим SPI (TMODE=0 –
режим I2S)
            MFBSP3.TCTR.bits.TPACK = 0;

            MFBSP3.TCTR_RATE.data = (FREQ/(2*SPI_FRQ)) - 1; //
частота работы SPI

// настройка приемника
            MFBSP3.RCTR.bits.RWORDLEN = 23;// длина принимаемого
слова - 24 разряда
            MFBSP3.RCTR.bits.RMBF = 1;       // старшим битом вперед
           MFBSP3.RCTR.bits.RWORDCNT=0; // 1 слово во фрейме

// RSCK и SS[0] - берутся от передатчика
            MFBSP3.RCTR.bits.RCS_CP = 1;
            MFBSP3.RCTR.bits.RCLK_CP = 1;

            MFBSP3.RCTR.bits.RMODE = 1;    // режим SPI



            MFBSP3.RCTR.bits.RNEG = 1;       // RNEG=1, RDEL=1. При
данном сочетании захват данных при
            MFBSP3.RCTR.bits.RDEL = 1;       // приеме данных
происходит по переднему фронту RSCK

// а исходное состояние RSCK перед началом приема - высокое.

// RSCK и SS[0] - берутся от передатчика

            MFBSP3.RCTR.bits.REN = 1;        // приемник включен
            MFBSP3.RSTART.data = 1;
            MFBSP3.TSTART.data = 1;
            MFBSP3.TCTR.bits.TEN = 1;       // TEN - передатчик
включен


                                            // Такой порядок включения
рекомендован разработчиком
}

unsigned int GetCP0_Count() {
                        unsigned int result;
                        asm volatile ("mfc0 %0, $9" :"=r"(result));
                        return result;
}

void SetCP0_Count(unsigned int value) {
                asm volatile ("mtc0 %0, $9" ::"r"(value));
}
```

```
unsigned int val_read = 0;
unsigned int val_read2 = 0;
unsigned int val_read3 = 0;

int main() {

      unsigned int rcv_array[24];
      unsigned int i;
      SPI_Init();

      SYNT1_write(SEL_REG,0x0000);
      SYNT1_write(CTR,0x1000);
      SYNT1_write(SYNC,0x0000);
      //SYNT1_write(T_CAPTURE,0x1BAD);
      SYNT1_write(TC_L,0x0000);
      SYNT1_write(TC_H,0x0000);
      SYNT1_write(CH1_TSW,0x0000);


      // Задается частота
      //Это частота 25МГц
      SYNT1_write(CH1_dPh0_L,0x0000);
      SYNT1_write(CH1_dPh0_M,0x0000);
      SYNT1_write(CH1_dPh0_H,0x0800);

      SYNT1_write(CH1_P0,0x0000);
      SYNT1_write(CH1_Mul0,0x7fff);
      SYNT1_write(CH1_Offset0,0x0000);

      // Ниже идет чтение записанных регистров
      rcv_array[0]=SYNT1_read(CH1_dPh0_H);
      rcv_array[1]=SYNT1_read(CH1_dPh0_H);
      rcv_array[2]=SYNT1_read(CH1_dPh0_H);
      rcv_array[3]=SYNT1_read(CH1_dPh0_H);
      rcv_array[4]=SYNT1_read(CH1_dPh0_H);
      rcv_array[5]=SYNT1_read(CH1_dPh0_H);
      rcv_array[6]=SYNT1_read(CH1_dPh0_H);
      rcv_array[7]=SYNT1_read(CH1_dPh0_H);
      rcv_array[8]=SYNT1_read(CH1_dPh0_H);
            rcv_array[9]=SYNT1_read(CH1_dPh0_H);
            rcv_array[10]=SYNT1_read(CH1_dPh0_H);
            rcv_array[11]=SYNT1_read(CH1_dPh0_H);
            rcv_array[12]=SYNT1_read(CH1_dPh0_H);
            rcv_array[13]=SYNT1_read(CH1_dPh0_H);
            rcv_array[14]=SYNT1_read(CH1_dPh0_H);
            rcv_array[15]=SYNT1_read(CH1_dPh0_H);
            rcv_array[16]=SYNT1_read(CH1_dPh0_H);
                  rcv_array[17]=SYNT1_read(CH1_dPh0_H);
                  rcv_array[18]=SYNT1_read(CH1_dPh0_H);
                  rcv_array[19]=SYNT1_read(CH1_dPh0_H);
                  rcv_array[20]=SYNT1_read(CH1_dPh0_H);
                  rcv_array[21]=SYNT1_read(CH1_dPh0_H);
                  rcv_array[22]=SYNT1_read(CH1_dPh0_H);
                  rcv_array[23]=SYNT1_read(CH1_dPh0_H);
```

```
	while (1) ;

}
```

Пример для MCStudio4 доступен по ссылке
http://multicore.ru/mc/software/harmonic_signal_synthesis.zip

5.2 Запись двух профилей, переключение между ними и демонстрация получившейся модуляции

Пример, приведенный ниже, демонстрирует модуляцию гармонического сигнала с помощью переключения профилей. Профили переключаются в регистре SEL_REG.

В профиле номер 0 задана частота 25МГц, в профиле номер 1 – частота 50МГц.

```
int main() {

	unsigned int rcv_array[24];
	unsigned int i;
	SPI_Init();

	SYNT1_write(SEL_REG,0x0000); //по умолчанию для первого канала
выбран профиль 0
	SYNT1_write(CTR,0x1000);
	SYNT1_write(SYNC,0x0000);
	//SYNT1_write(T_CAPTURE,0x1BAD);
	SYNT1_write(TC_L,0x0000);
	SYNT1_write(TC_H,0x0000);
	SYNT1_write(CH1_TSW,0x0000);

	SYNT1_write(CH1_P0,0x0000);
	SYNT1_write(CH1_Mul0,0x7fff);
	SYNT1_write(CH1_Offset0,0x0000);

	// Задается частота 0-го профиля
	//Это частота 25МГц
	SYNT1_write(CH1_dPh0_L,0x0000);
	SYNT1_write(CH1_dPh0_M,0x0000);
	SYNT1_write(CH1_dPh0_H,0x0800);

	// Задается частота 1-го профиля
	//Это частота 50МГц
	SYNT1_write(CH1_Mul1,0x7fff); //задается амплитуда
	SYNT1_write(CH1_dPh1_L,0x0000);
	SYNT1_write(CH1_dPh1_M,0x0000);
	SYNT1_write(CH1_dPh1_H,0xF000);

	while (1) {

	SYNT1_write(SEL_REG,0x0000); //для первого канала выбран профиль
0
	while (GetCP0_Count()<TIMEOUT) ; // через 1 секунду
	SYNT1_write(SEL_REG,0x0001); //для первого канала выбран профиль
1
```

```
        while (GetCP0_Count()<TIMEOUT) ; // через 1 секунду
        }

}
```

На рисунке ниже показан результирующий сигнал, по которому можно наблюдать переключение с профиля с записанными 50МГц на профиль с 25МГц.

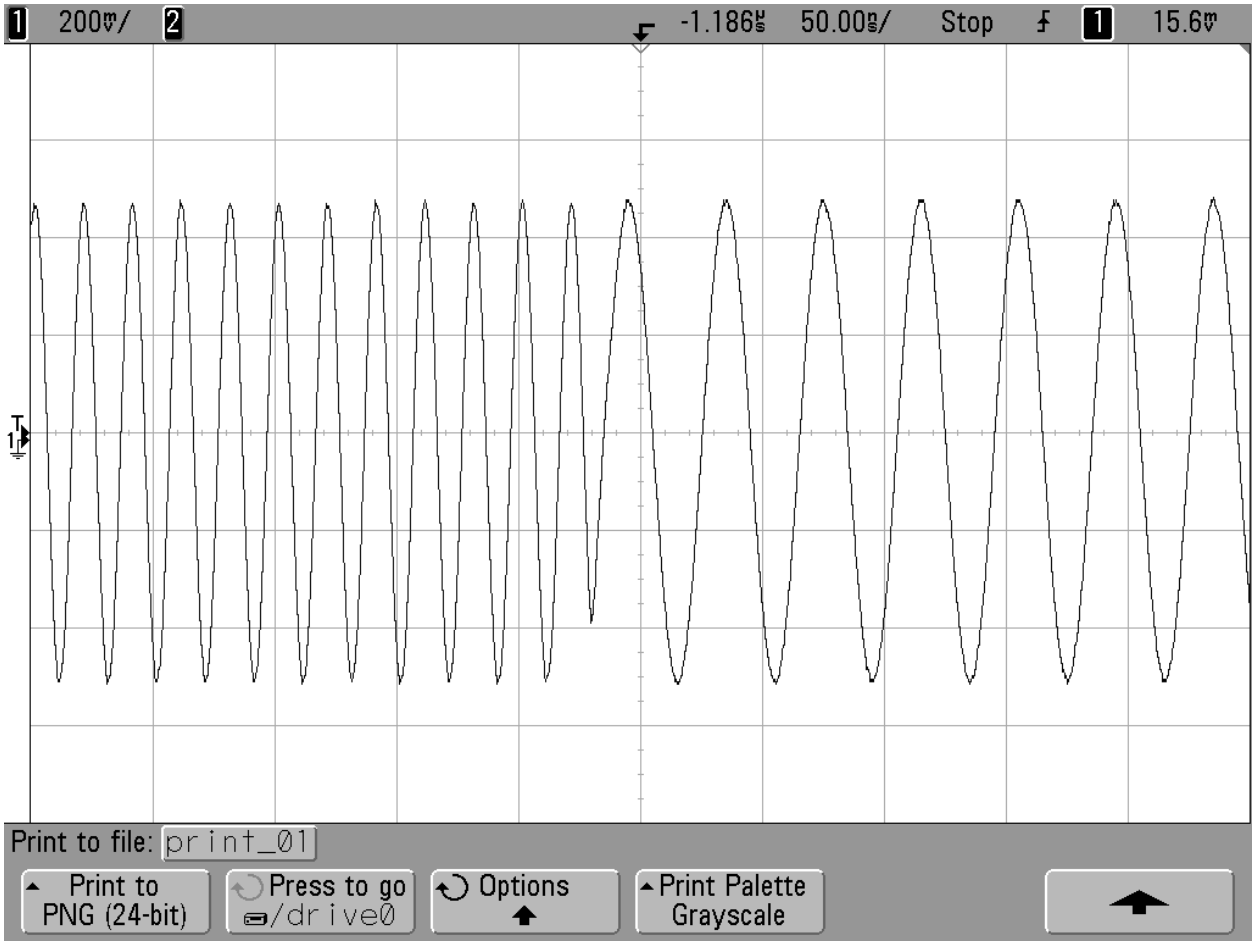

Рисунок 5.3. Модуляция гармонического сигнала

Пример для MCStudio4 доступен по ссылке
http://multicore.ru/mc/software/harm_signal_modulation.zip

5.3 Реализация ЛЧМ и ее демонстрация

В данном разделе продемонстрируем ЛЧМ-сигнал, формирующийся из 4-х последовательных стадий:

- в стадиях 1 и 3 происходит равномерное нарастание частоты с 25МГц до 50МГц,
- в стадиях 2 и 4 частота нулевая,
- длительность каждой стадии составляет 1 мкс.

Таким образом, в течение 1 мкс происходит нарастание частоты, затем 1мкс – перерыв в сигнале, затем снова нарастание с 25 до 50 Мгц, затем такой же перерыв.

5.3.1 Длительность стадии

Приведем расчет длительности стадии.

В регистрах с наименованием типа CHx_LS_TPHy_L(M, H) (регистры длительности 1, 2, 3, 4 стадий) содержится время длительности каждой из стадий в тактах частоты, поступающей на входы CLKDP и CLKDM (тактовой частоты микросхемы 1508ПЛ8Т), которое рассчитывается по следующей формуле:

(желаемое время стадии)* Fclk/4;

Для стадии длительностью 1мкс при тактовой частоте 800МГц:

1*10^(-6)*800*10^ 6 /4 = 800/4 =200;

0xC8 – в шестнадцатеричном формате.

При отключенной коррекции (бит corr_enable=0 регистра CHx_LS_CTR), параметры фазы, амплитуды и постоянного смещения синтезируемого сигнала берутся из профиля с номерами 1, 2, 3, 0 для стадий 1-4 соответственно. Поэтому при отключении коррекции необходимо записать значения амплитуды, начальной фазы и частоты в регистры перечисленных профилей.

Код программы на Си с учетом вышесказанного будет выглядеть так:

```
SYNT1_write(SEL_REG,0x0000); //по умолчанию для первого канала выбран профиль 0
     SYNT1_write(CTR,0x1000);
     SYNT1_write(SYNC,0x0000);
     SYNT1_write(TC_L,0x0000);
     SYNT1_write(TC_H,0x0000);
     SYNT1_write(CH1_TSW,0x0000);

//Необходимо задать значения профилей с 0-го по 4-ый, так как
//при отключенном режиме корреции параметры фазы, амплитуды и
//постоянного смещения
//синтезируемого сигнала берутся из профиля с номерами 1, 2, 3, 0
//для стадий 1-4 соответственно.
     SYNT1_write(CH1_P0,0x0000);
     SYNT1_write(CH1_Mul0,0x7fff);
     SYNT1_write(CH1_Offset0,0x0000);

     SYNT1_write(CH1_P1,0x0000);
     SYNT1_write(CH1_Mul1,0x7fff);
     SYNT1_write(CH1_Offset1,0x0000);

// Записываем во все профили
     SYNT1_write(CH1_dPh_all_L,0x0000);
     SYNT1_write(CH1_dPh_all_M,0x0000);
     SYNT1_write(CH1_dPh_all_H,0x0000);
     SYNT1_write(CH1_P_all,0x0000);
     SYNT1_write(CH1_Mul_all,0x7fff);
     SYNT1_write(CH1_Offset_all,0x0000);

// Задается начальная частота стадии 1
     SYNT1_write(CH1_LS_F1_L,0x0000);
```

```
      SYNT1_write(CH1_LS_F1_M,0x0000);
      SYNT1_write(CH1_LS_F1_H,0x0800);

// Задается начальная частота стадии 3
     SYNT1_write(CH1_LS_F2_L,0x0000);
      SYNT1_write(CH1_LS_F2_M,0x0000);
      SYNT1_write(CH1_LS_F2_H,0x0800);

// Задается приращение фазы (0)
      SYNT1_write(CH1_dPh0_L,0x0000);
      SYNT1_write(CH1_dPh0_M,0x0000);
      SYNT1_write(CH1_dPh0_H,0x0000);

// Задается приращение частоты в 1-ой и 2-ой стадиях ЛЧМ
      SYNT1_write(CH1_LS_dFq1_L,0xA3D7);
      SYNT1_write(CH1_LS_dFq1_M,0x3D70);
      SYNT1_write(CH1_LS_dFq1_H,0x0000A); // 0x000A_3D70_A3D7

      SYNT1_write(CH1_LS_dFq2_L,0xA3D7);
      SYNT1_write(CH1_LS_dFq2_M,0x3D70);
      SYNT1_write(CH1_LS_dFq2_H,0x000A);

// Задается длительность каждой стадии ЛЧМ
      SYNT1_write(CH1_LS_TPH1_L,0x00C8); // 1 мкс: 1*10^(-6)*800*10^ 6
/4 = 800/4 = 200 0xC8 в результате 1мкс
      SYNT1_write(CH1_LS_TPH1_M,0x0000);
      SYNT1_write(CH1_LS_TPH1_H,0x0000);

      SYNT1_write(CH1_LS_TPH2_L,0x00C8);
      SYNT1_write(CH1_LS_TPH2_M,0x0000);
      SYNT1_write(CH1_LS_TPH2_H,0x0000);

      SYNT1_write(CH1_LS_TPH3_L,0x00C8);
      SYNT1_write(CH1_LS_TPH3_M,0x0000);
      SYNT1_write(CH1_LS_TPH3_H,0x0000);

      SYNT1_write(CH1_LS_TPH4_L,0x00C8);
      SYNT1_write(CH1_LS_TPH4_M,0x0000);
      SYNT1_write(CH1_LS_TPH4_H,0x0000);

// Включение частоты
      SYNT1_write(CH1_LS_CTR,0xBC70);
      SYNT1_write(CLR,0x0010);
      while (1);

}
```

Пример для MCStudio4 доступен по ссылке http://multicore.ru/mc/software/LFM.zip

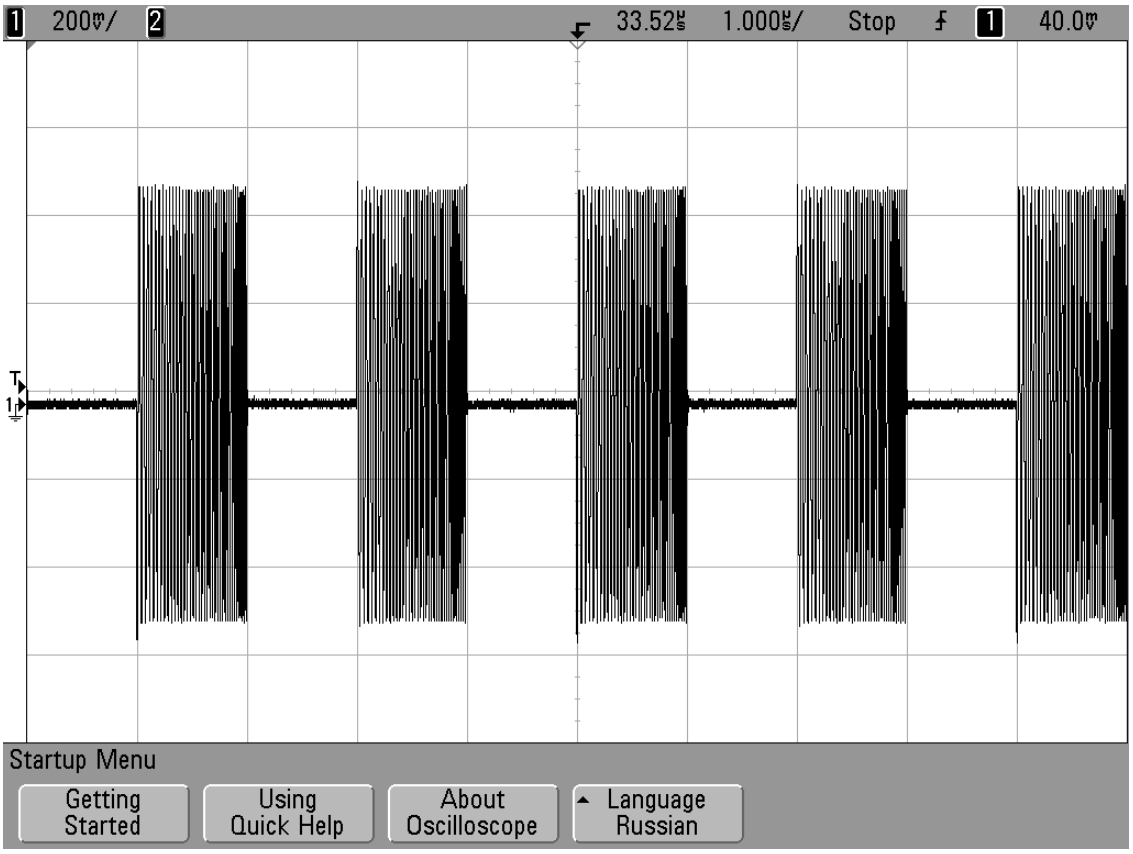

Рисунок 5.4. Чередование стадий ЛЧМ длительностью 1 мкс каждая

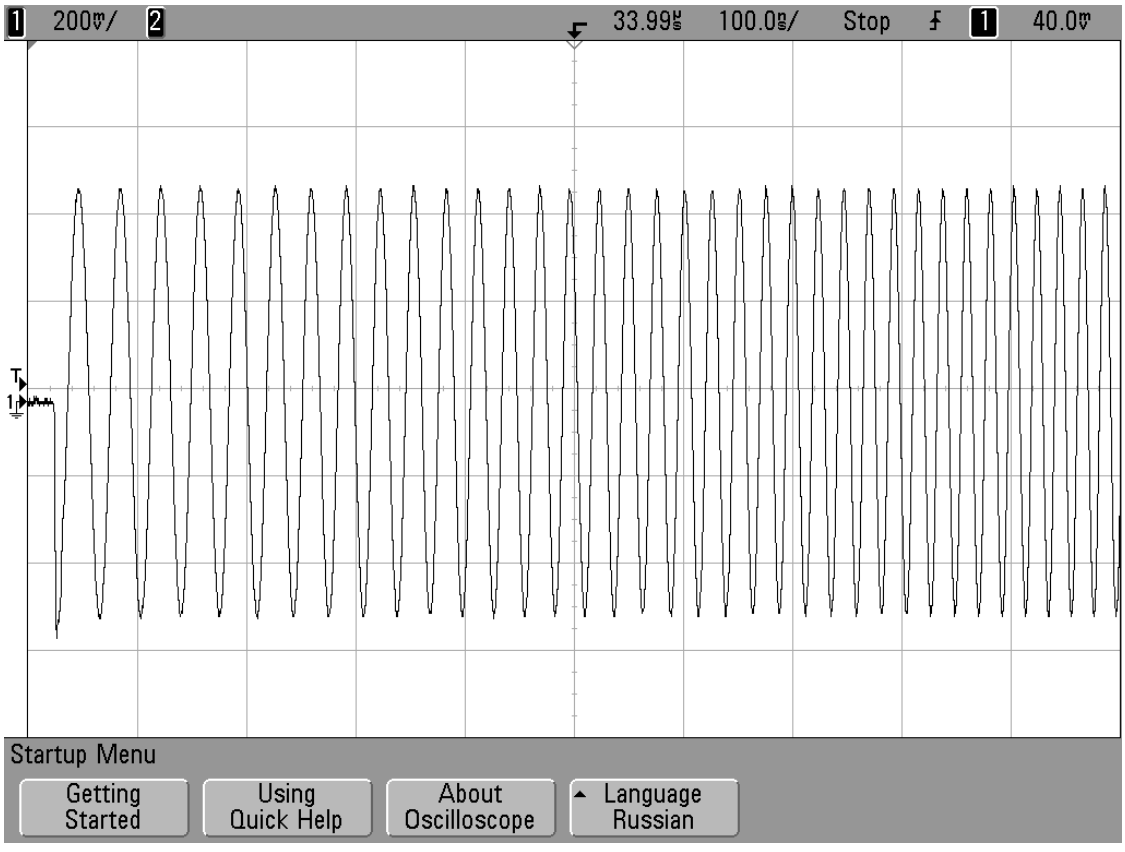

Рисунок 5.5. Плавное увеличение частоты с 25МГц до 50МГц

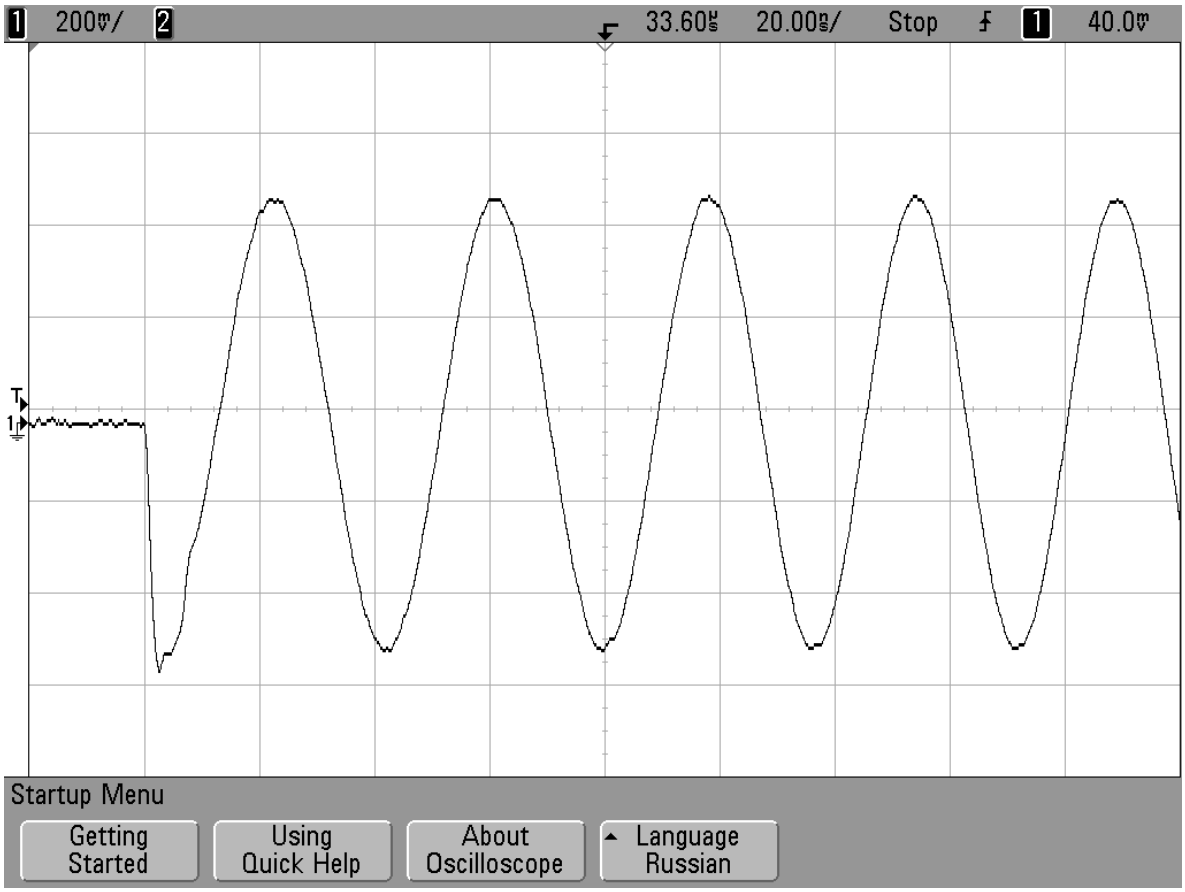

Рисунок 5.6. Начало стадии 1 или 3 – сигнал 25МГц

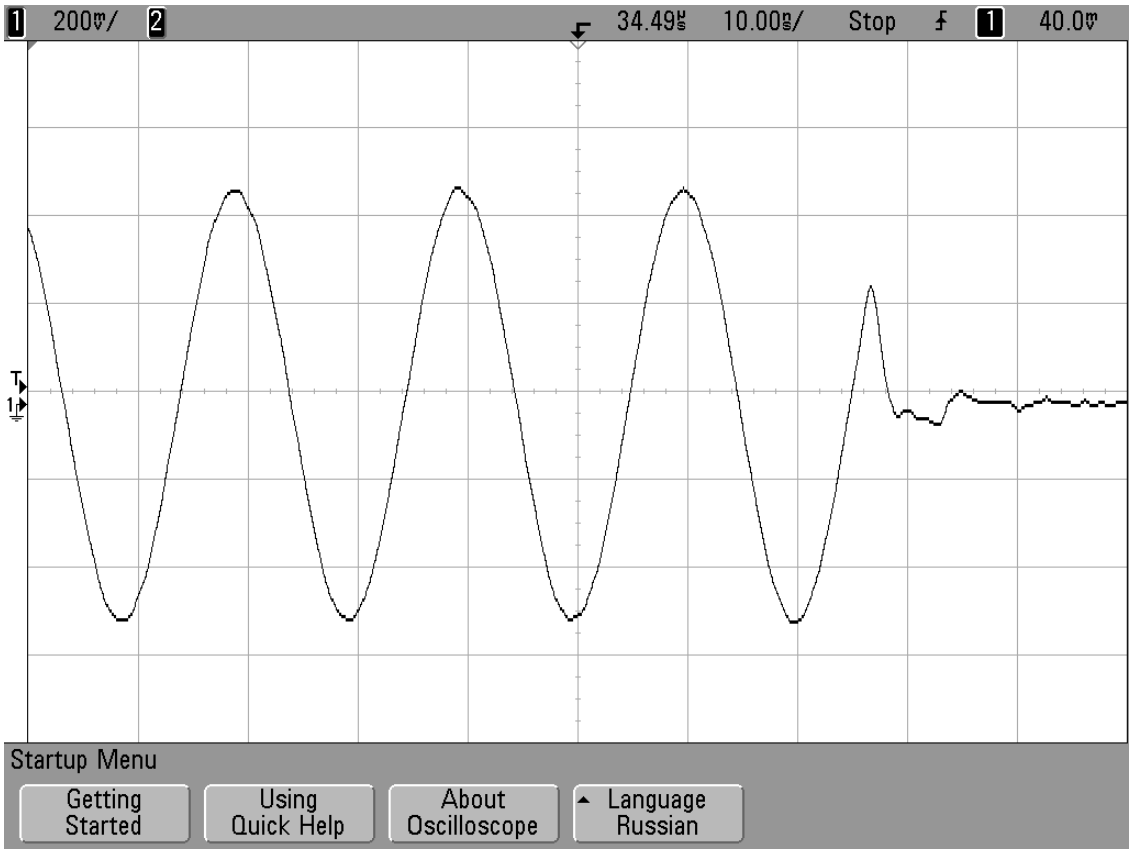

Рисунок 5.7. Окончание стадии 1 или 3 – сигнал 50МГц

5.3.2 Неравномерность амплитуды из-за больших скачков частоты

Если задано слишком быстрое изменение частоты, например, в регистры приращения частоты записаны следующие значения:

```
// Задается приращение частоты в 1-ой и 2-ой стадиях ЛЧМ
     SYNT1_write(CH1_LS_dFq1_L,0x0000);
     SYNT1_write(CH1_LS_dFq1_M,0x0000);
     SYNT1_write(CH1_LS_dFq1_H,0x0100);

     SYNT1_write(CH1_LS_dFq2_L,0x0000);
     SYNT1_write(CH1_LS_dFq2_M,0x0000);
     SYNT1_write(CH1_LS_dFq2_H,0x0100);
```

то результирующий сигнал может выглядеть как на рисунке ниже.

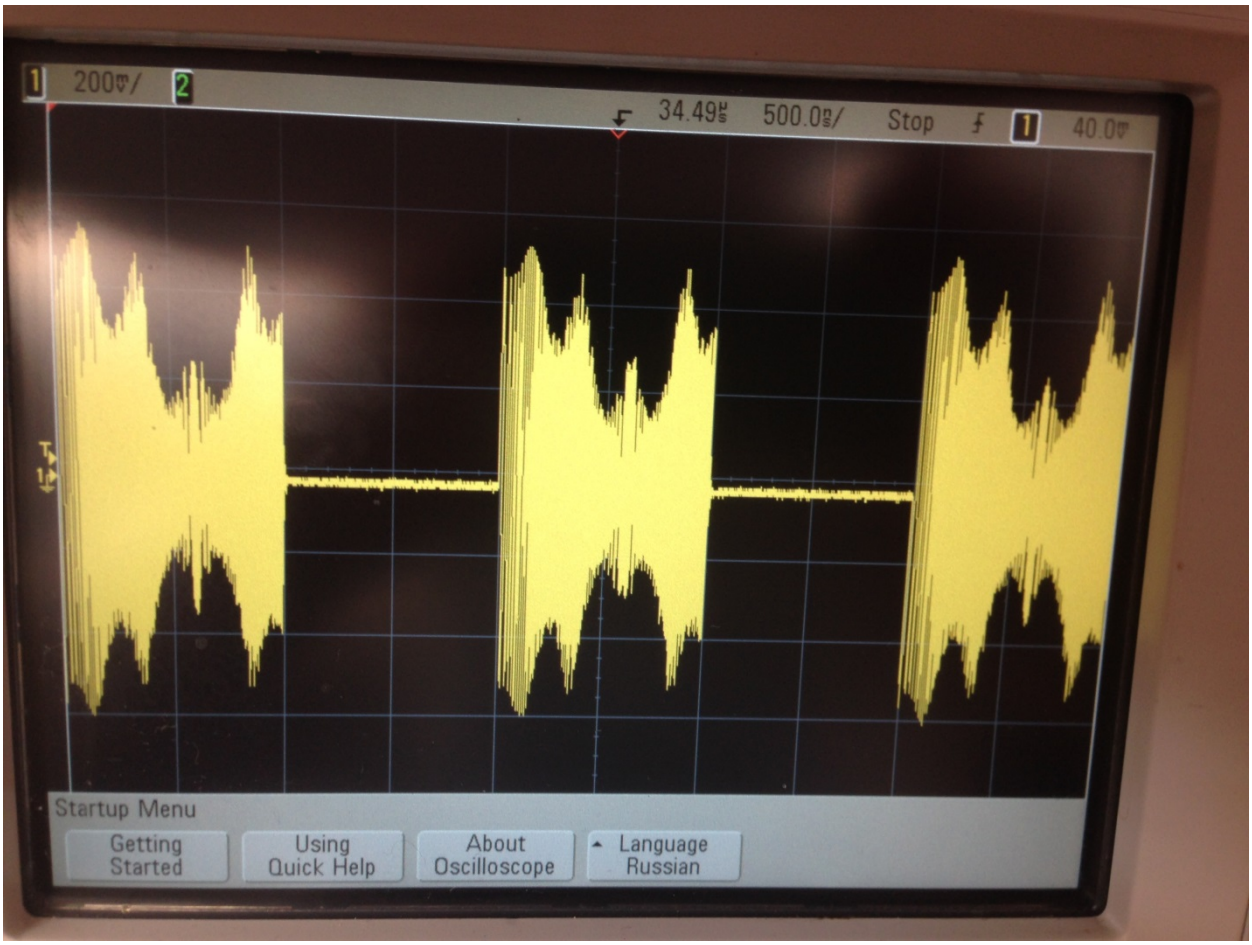

Рисунок 5.8. Амплитуда сигнала при слишком быстром изменении частоты

Если частота быстро меняется в широких пределах, на выходе без фильтрации может наблюдаться взаимодействие основного тона и зеркальных составляющих. Это те составляющие на спектроскопе, которые находятся выше частоты Найквиста (половины частоты дискретизации).

6. ИСТОРИЯ ИЗМЕНЕНИЙ

6.1 Изменения от 14.08.2019

Добавлены ссылки на примеры для MCStudio4.