Разработка схемы и программы в части сопряжения микропроцессора с элементами ввода информационного сигнала (ЦАП)

1. Сведения о ЦАП

Цифро-аналоговый преобразователь — устройство для перевода цифровых данных в аналоговый сигнал. В рамках данной работы будет использован AD5310

AD5310 – микросхема, содержащая один ЦАП, обеспечивающий формирование выходного напряжения 10-разрядным двоичным кодом, питающийся однополярным напряжением от +2.5 В до +5.5 В и потреблением 115 мкА при 3В питания. В AD5310 встроен 3-проводный последовательный интерфейс, который работает на частоте до 30МГц и совместим со стандартными интерфейсами SPI, QSPI, MICROWIRE и DSP.

Характеристики:

- Один 10-разрядный ЦАП
- Поставка в 6-выводном SOT-23 и 8-выводном µSOIC корпусах
- Малое энергопотребление: 140 мкА при 5 В питания
- Режим пониженного потребления: 200 нА при 5В, 50 нА при 3 В
- Питание от источника напряжением от +2.5В до +5.5В
- Гарантируется равномерность дифференциальной нелинейности для любых значений входного кода
- В качестве опорного напряжения используется напряжение питания
- Сброс в ноль выходного напряжения при подаче питания
- Три режима пониженного энергопотребления
- Экономичный последовательный интерфейс с триггерами Шмитта на входах
- Встроенный усилитель-повторитель выходного напряжения
- Вход SYNC для прерывания записи.

Структурная схема устройства изображена на рисунке 1, а на рисунке 2 показано расположение выводов ЦАПа.

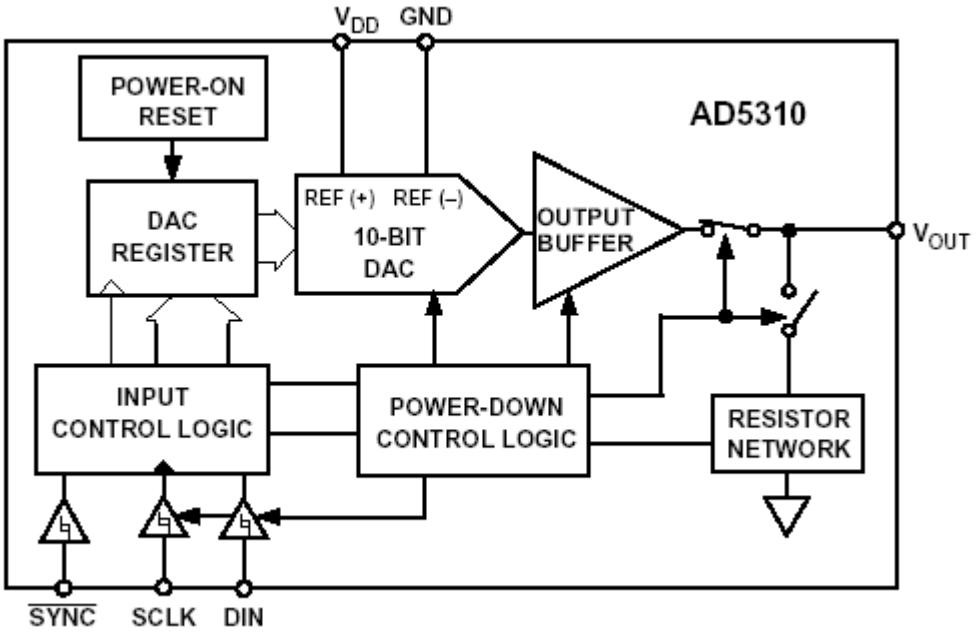

Рисунок 1 – Структурная схема устройства

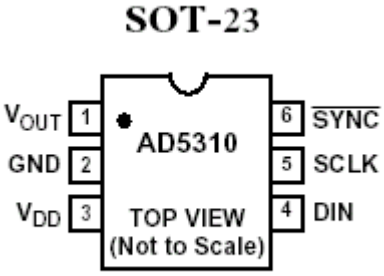

Рисунок 2 – Расположение выводов ЦАПа

Описание выводов:

- V_{DD} – Вход питания
- V_{OUT} – Выходное усиленное напряжение
- GND – Общий для всех частей микросхемы
- SYNC - Управляющий вход с активным низким уровнем (выбор микросхемы).
- SCLK – Вход синхронизации последовательной связи
- DIN – Ввод последовательных данных[1]

Временная диаграмма передачи данных на ЦАП показана на рисунке 3. Можно увидеть, что чтобы передать данные вывод SYNC должен быть притянут к земле во время передачи всех 16 бит.

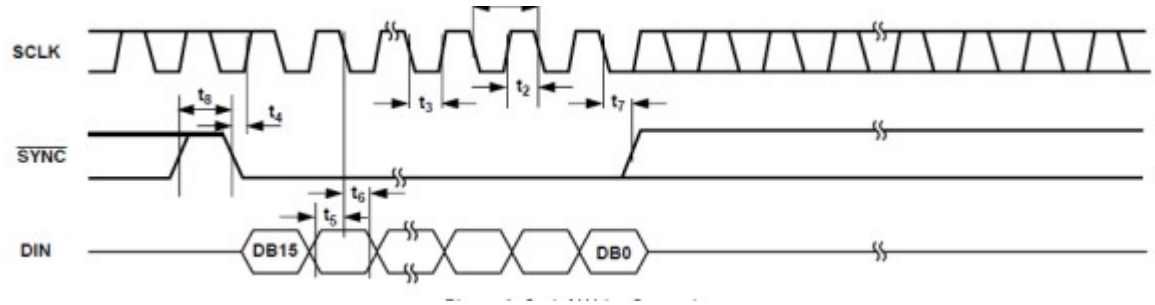

Рисунок 3 – Временная диаграмма передачи данных на ad5310

Конкретное содержание 16 бит передаваемых данных показано на рисунке 4. Биты PD1 и PD0 – задают режим работы ЦАП, мы будем использовать режим по умолчанию, так что там всегда будут нули. Биты с D9 до D0 – биты данных, через которые задаётся значение формируемое на выходе ЦАП. Обратите внимание, что нулевой бит находится справа.

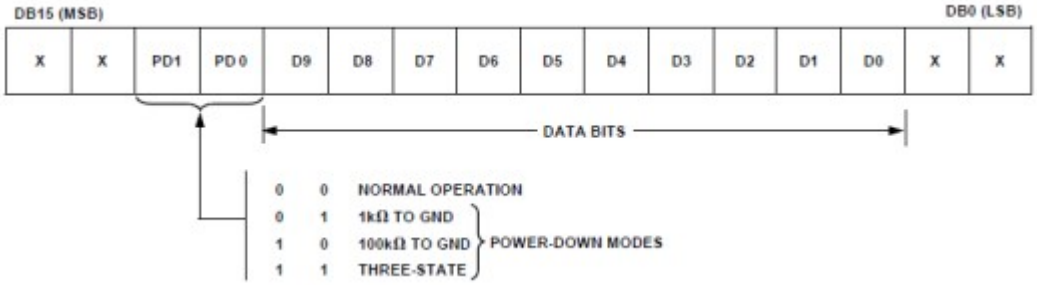

Рисунок 4 – Содержимое входного регистра данных

2. Сведения о передатчике значений

Для передачи данных ЦАП был выбран синхронный последовательный порт из-за его простоты и понятности. В AT89C51ED2 он реализован в виде нулевого режима работы последовательного порта. В рамках этого режима информация передаётся, и принимается через вывод входа приемника RxD. Принимаются или передаётся 8 бит данных. Для нас это значит, что передаваемые данные придётся дробить на два байта. Через вывод TxD выдаются импульсы синхронизации, которые сопровождают каждый бит. Скорость передачи фиксирована и составляет 1/6 $F_{ген}$. Для его настройки в выбранном микроконтроллере вообще не нужно ничего никуда записывать.

Передача по последовательному порту начинается после записи байта в регистр данных SBUF. Временная диаграмма сигнала, вырабатываемого последовательным портом микроконтроллера при передаче восьми бит данных приведена на рисунке 5. Снова обращаю внимание на положение нулевого бита в передаваемом байте. В данном случае он находится слева. Это значит, что чтобы корректно передать на ЦАП значение придётся реверсировать порядок битов. Так же, на временной диаграмме можно

заметить, что после окончания передачи устанавливается флаг TI, сигнализирующий об окончании передачи, для того, чтобы последовательный порт работал корректно его нужно сбрасывать по окончанию передачи байта.

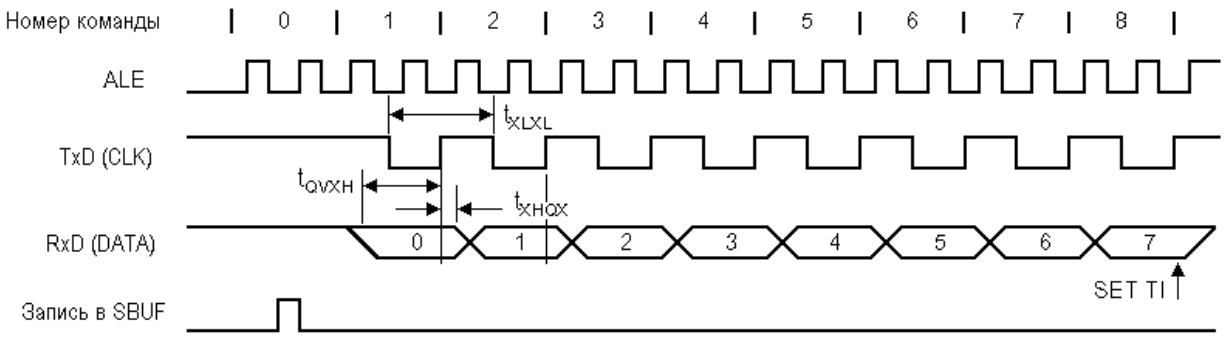

Рисунок 5 – Временная диаграмма работы последовательного порта в нулевом режиме после записи передаваемого байта в регистр данных SBUF.

3. Написание программы для работы с ЦАП

Итак, чтобы послать корректные данные на ЦАП нужно решить несколько задач:

- Разбить значение на байты
- Расположить значение внутри двух байтов так, чтобы оно соответствовало порядку битов данных в регистре данных ЦАП
- Реверсировать порядок битов значения

Для того, чтобы это сделать было создано несколько функций и макросов. Содержание кода этапа можно найти в приложении 1.

Рассмотрим происходящее внутри кода.

Чтобы передача данных не затягивалась было решено вынести разбивку и форматирование переселаемого значения в начало функции по пересылке значения shift_transmit. Следующая строчка реверсирует порядок битов значения, вызывая функцию reverse:

```
data_to_transmit = reverse(data_to_transmit);
```

Функция reverse довольно незамысловата. На входе она принимает число, а на выходе выдаёт его копию, но с зеркально расположенными битами. Например, если на вход поступит «0111101010», то на выходе будет уже «0101011110»

Далее, разобьём значение на два бита и расположим их в требуемых местах байтов данных с помощью вызовов макросов extract_first_byte и

extract_second_byte, которые просто выполняют побитовый сдвиг влево или в право

```
first_byte = extract_first_byte(data_to_transmit);
second_byte = extract_second_byte(data_to_transmit);
```

Таким образом, наши биты готовы к отправке. Обнуляем вывод sync, чтобы сигнализировать ЦАП, что передача данных началась:

```
sync = 0;
```

Далее мы записываем первый бит данных в SBUF, ждём окончания передачи и обнуляем флаг окончания передачи:

```
sync = 0;
SBUF = first_byte;
while(TI==0);
TI = 0; TI = 0;
```

Проделываем тоже самое с вторым байтом данных и устанавливаем выход sync в единицу, сигнализируя ЦАП, что передача значения окончена

```
SBUF = second_byte;
while(TI==0);
TI = 0; TI = 0;
sync = 1;
```

Для того, чтобы продемонстрировать работоспособность написанного кода передадим на ЦАП пилообразный сигнал. Для этого соберём схему в proteus как показано на рисунке 6 и загрузим код из приложения 1. На выходе получиться пилообразный сигнал изменяющийся в диапазоне от 0 до 5 В. Выходной сигнал показан на рисунке 7.

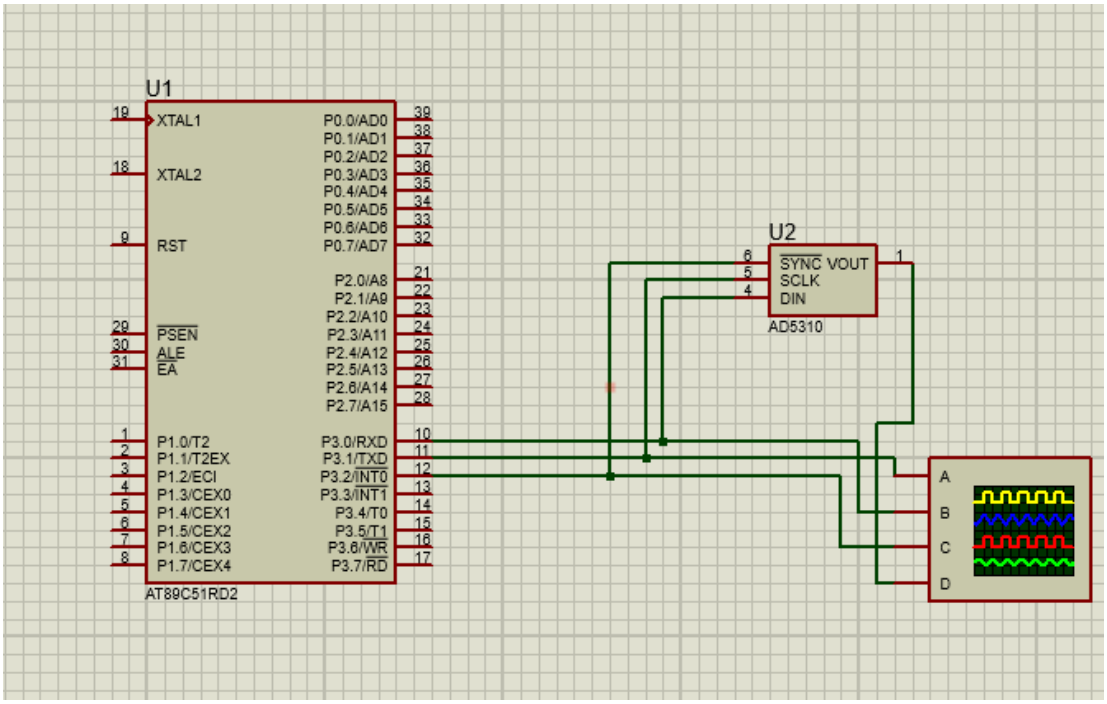

Рисунок 6 – Схема подключения ЦАП к микроконтроллеру

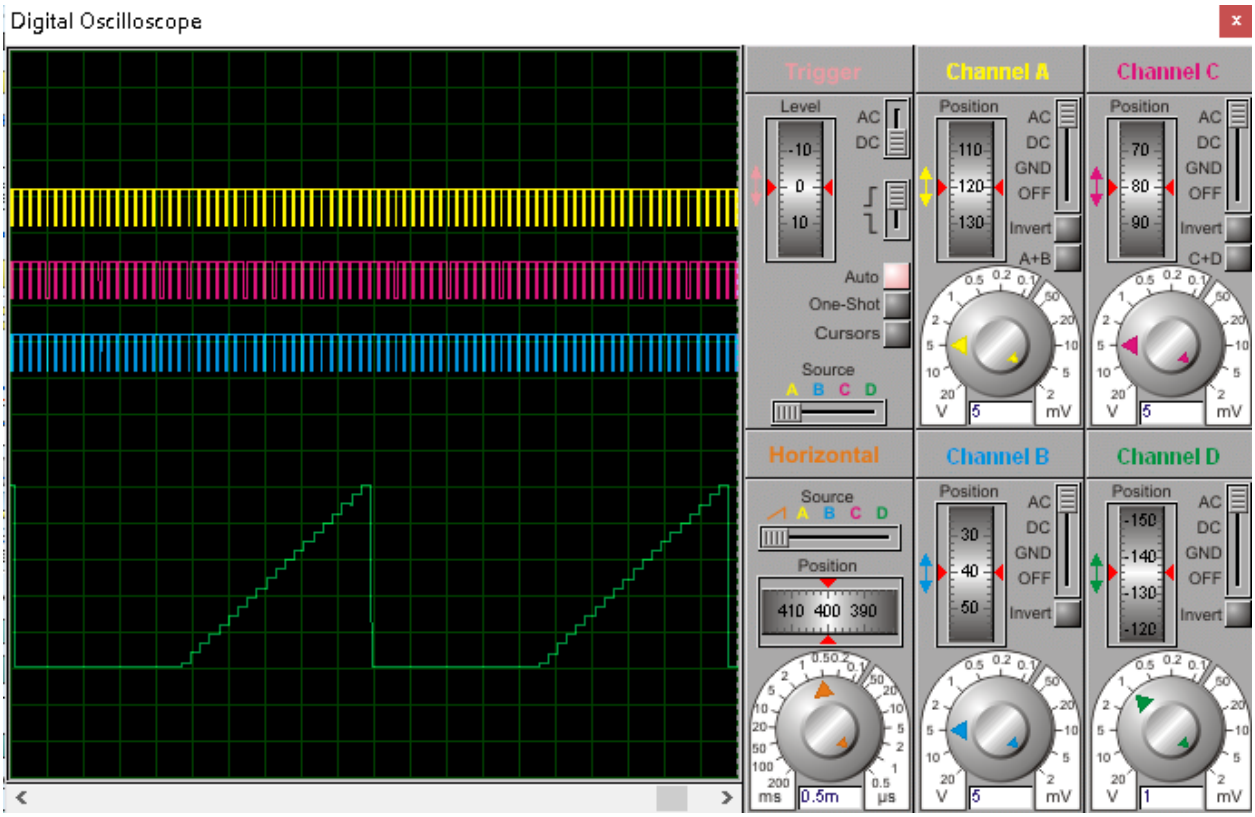

Рисунок 7 – Осциллограмма выходного сигнала ЦАП(зелёным)

Список источников

1. AD5310 Datasheet. Электронный ресурс URL:https://www.analog.com/media/en/technical-documentation/data-sheets/ad5310.pdf (Дата обращения 11.02.2021)

2. Последовательный порт микроконтроллера 8051. Электронный ресурс URL:https://digteh.ru/MCS51/PoslPort.php (Дата обращения 11.02.2021)

ПРИЛОЖЕНИЕ 1

```
#include <at89c51xd2.h>

#define sync P3_2
#define extract_first_byte(value) (value & 0xF) << 4
#define extract_second_byte(value) value >> 4
#define SAWTOOTH_LENGTH 39

unsigned int reverse(unsigned int value)
{
	unsigned int result = 0;
	if ( value & 0x200 ) result |= 0x001;
	if ( value & 0x100 ) result |= 0x002;
	if ( value & 0x080 ) result |= 0x004;
	if ( value & 0x040 ) result |= 0x008;
	if ( value & 0x020 ) result |= 0x010;
	if ( value & 0x010 ) result |= 0x020;
	if ( value & 0x008 ) result |= 0x040;
	if ( value & 0x004 ) result |= 0x080;
	if ( value & 0x002 ) result |= 0x100;
	if ( value & 0x001 ) result |= 0x200;
	return result;
}

void shift_transmit(unsigned int data_to_transmit){
	char first_byte;
	char second_byte;

	data_to_transmit = reverse(data_to_transmit);

	first_byte = extract_first_byte(data_to_transmit);
	second_byte = extract_second_byte(data_to_transmit);

	sync = 0;
	SBUF = first_byte;
	while(TI==0);
	TI = 0; TI = 0;

	SBUF = second_byte;
	while(TI==0);
	TI = 0; TI = 0;
	sync = 1;
}

void main (void){
```

```
	unsigned int sawtooth [SAWTOOTH_LENGTH] = {0,  0,  0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0,

                                                0,  0,  0,  0,  0,  0,  26,
78, 131, 183, 236, 288, 341,

                                                393, 446, 498, 551, 603, 656, 708,
761, 813, 866, 918, 971,1023};
	int sawtooth_index;

	sync = 0;

	while(1){
		for(sawtooth_index = 0; sawtooth_index < SAWTOOTH_LENGTH;
sawtooth_index++){
			shift_transmit(sawtooth[sawtooth_index]);
		}
	}
}
```