МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра компьютерной физики и метаматериалов на базе Саратовского филиала Института радиотехники и электроники им. В.А. Котельникова РАН

Исследование и доработка компьютерного программного обеспечения для микроконтроллера, выполняющего измерения временных интервалов отражённых волн

АВТОРЕФЕРАТ

студента 4 курса 432 группы

направления 03.03.02 «Физика» физического факультета

Шуршилина Павла Владимировича

Научный руководитель

профессор, канд. ф.-м. н В.В. Петров

31.05.2019

Зав. кафедрой

профессор, д.ф.-м.н. В.М. Аникин

31.05.2019

Саратов 2019

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность. На текущий момент создано огромное количество микроконтроллеров, отличающихся архитектурой процессора, типом, размером и объёмом встроенной памяти, набором периферийных устройств, количеством цифровых и аналоговых вводов и выводов. В зависимости от возможностей микроконтроллера они применяются как в вычислительных устройствах, например калькуляторах, материнских платах персональных компьютеров, так и в электронике и различной бытовой технике, использующей электронные системы управления. В промышленности применение микроконтроллеров особенно часто встречается в системах управления различными станками, системах автоматизации производства и т.д.. В большинстве случаев, в промышленности используют более простые микроконтроллеры. Это связанно с тем, что от них не требуется большая производительность, но важна низкая стоимость. С другой стороны, микроконтроллеры общего пользования почти всегда требуют серьёзные вычислительные мощности для работы с большими потоками данных в режиме реального времени.

Цель выпускной квалификационной работы (ВКР). Основной целью работы является исследование и доработка программного обеспечения, для обработки данных, полученных с ультразвуковых датчиков. Аппаратными компонентами работы являются ультразвуковой датчик, работающий в определённом диапазоне частот и плата TDC1000-TDC7200EVM. Данная работа позволит использовать результаты измерений в реальном времени на компьютере, что даёт большие возможности в создании систем мониторинга, возможности создания счётчика для измерения проходящего сквозь него количества газа или жидкости.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Основное содержание. Выпускная квалификационная работа содержит лист определений, обозначений и сокращений, введение, два основных раздела, заключение, список использованных источников из 18 наименований.

В разделе 1 разбираются особенности программного обеспечения для микроконтроллера.

В **п. 1.1** дается краткое описание используемого микроконтроллера, основных особенностей и характеристик линейки данного микроконтроллера, возможностей данного компонента в плане энергосбережения, а так же

использования USB и API для облегчения программирования устройств, передающих информацию с микроконтроллера на персональный компьютер.

В п. 1.2 рассматривается принцип работы платы TDC1000-TDC7200EVM и основных её компонентов: TDC1000 и TDC7200.

TDC 1000 представляет собой аналоговый узел, включающий в себя передатчик и приёмник. Передатчик TDC1000 создан для формирования возбуждающих импульсов для разных типов излучателей с рабочими частотами от 31,5 кГц до 4 МГц. Микросхема позволяет формировать импульсы с различной огибающей. Приемный тракт включает усилитель с программируемым коэффициентом усиления, компаратор и выходной мультиплексор, формирующий выходные импульсы СТАРТ-СТОП для внешнего измерителя временных интервалов TDC7200. TDC 7200 предназначен для измерения между посланными и отражёнными импульсами, (time-of-flight) для ультразвуковых измерителей. TDC7200 имеет внутренний источник базового временного интервала, с автоматической компенсацией смещения при изменении температуры. Это позволило добиться пикосекундой точности преобразования «time-to-digital».

В исследуемой плате TDC 1000 отвечает за отправку и приём отраженного сигнала. Как после отправки, так и после приёма TDC 1000 подаёт сигнал на TDC 7200, который рассчитывает временной интервал между моментом принятия сигналов СТАРТ-СТОП.

TDC7200 является компонентом, предназначенным лишь для сверх точного измерения временных интервалов. Разрешение устройства 35 пс.

Принцип работы TDC 1000 и TDC 7200 нагляднее всего рассматривать на схеме (рисунок 1). Получив от микроконтроллера команду на проведение измерений, TDC 1000 формирует импульс(или последовательность импульсов), одновременно отправляя сигнал СТАРТ TDC 7200, который ,получив сигнал, запускает отсчёт времени. TDC 1000, перешедший в режим сканирования, ожидает отражённую волну. В следствии того, что распространение волны происходит в среде, отраженная волна, возникает эффект собственного резонанса, следовательно волна имеет синусоидальный вид, и поэтому, отличить шумы от отраженного сигнала становится сложнее. Поэтому, TDC 1000 срабатывает в тот момент, когда амплитуда волны достигает амплитуды исходного сигнала. В этот момент TDC 1000 отправляет TDC 7200 сигнал СТОП, после чего останавливается таймер и полученные замеры времени, а так же температуры среды, если подключен датчик температуры, передаются

управляющему микроконтроллеру, который и занимается обработкой данных.

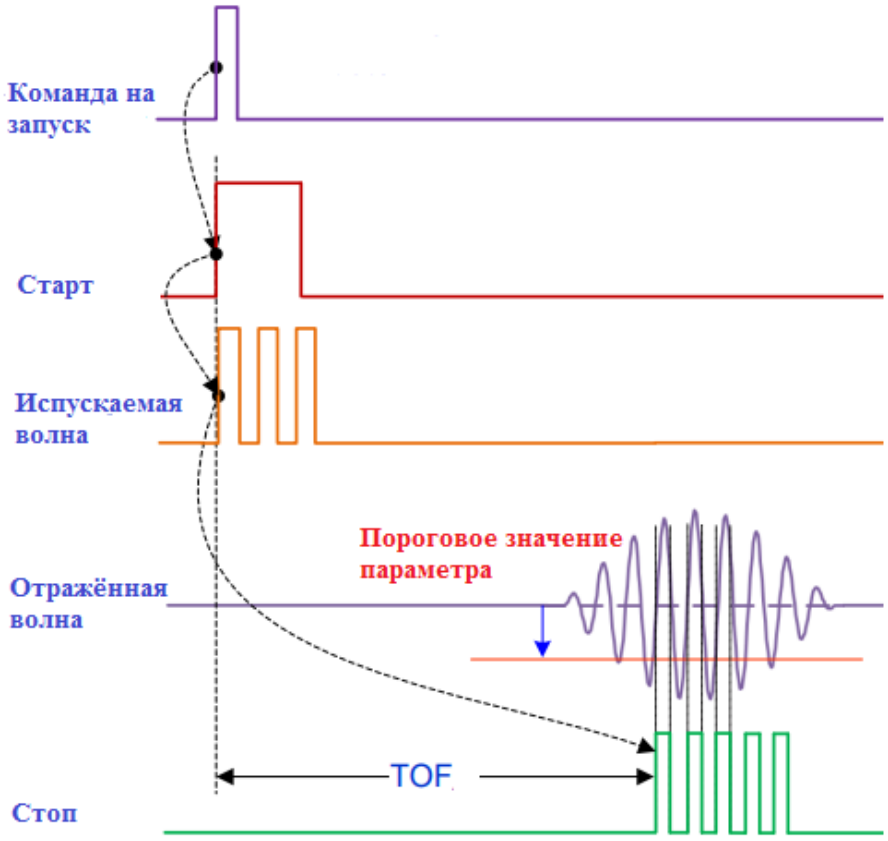

Рисунок 1. Схема работы TDC1000 и TDC7200

TDC 1000 и TDC 7200 имеют несколько изменяемых параметров для работы. К ним можно отнести – количество испускаемых импульсов, точность измерения, однократная или многократная отправка сигнала «стоп» и многое другое. Следовательно, в связи с большим количеством изменяемых параметров, происходит усложнение программного обеспечения микроконтроллера(msp430f5528).

В п. 1.3 проводится исследование особенностей программного обеспечения.

Перед работой непосредственно над созданием ПО для микроконтроллера, следует определить требуемый функционал программы. В случае с платой TDC1000-TDC7200EVM, данный функционал должен включать в себя:

- Запуск компонентов платы.
- Установка стартовых параметров работы TDC1000 и TDC 7200.
- Запуск измерений временных интервалов.

- Получение и обработка микроконтроллером результатов измерения временных интервалов.
- Передача данных на ПК через интерфейс UART и(или) USB.
- Отображение результатов на ПК
- Реализация однократных и многократных измерений.
- Возможность сохранения и загрузки последней рабочей конфигурации.

Содержание данного раздела работы – изучение вышеперечисленных функций и практическая реализация следующих из них:

- Запуск компонентов платы.
- Запуск измерений временных интервалов.
- Передача данных на ПК и взаимодействие через интерфейс USB.
- Отображение результатов на ПК

В связи с тем, что полная версия кода для этого микроконтроллера имеет слишком размер, приведение его в полной версии нецелесообразно. Поэтому, программное обеспечение для MSP430F5528 будет рассматриваться фрагментами, выполняющими ключевые функции.

Как и в любом проекте на C, данная программа начинает свою работу с функции “int_main()”

```
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;                                   // Stop WDT
    InitMCU();
    if (TDC1000_UART_Stream)
    {
        // Init UART
        InitUART();
        sprintf((char *) outString, "Hello World \r\n");
        putsUART((unsigned char *) outString, strlen((char *) outString));
    }

    TI_TDC7200_SPISetup();                      // Initilaize MSP430 - 7200 SPI Block

    TI_TDC1000_SPISetup();                      // Initilaize MSP430 -1000 SPI Block
```

Рисунок 2. Стартовая функция – main.c

На коде, представленном на рисунке 2, последовательно происходит выполнение нескольких команд. Первым делом, при работе с данным микроконтроллером выключить, включенный по умолчанию сторожевой таймер (WDTCTL = WDTPW + WDTHOLD;). Этот таймер отвечает за перезагрузку

микроконтроллера если таймер произвёл отсчёт до нуля. В данном случае использование данного таймера будет мешать работе программы. После отключения таймера происходит запуск UART (InitUART();) из подключенной библиотеки, и выводом тестовой строки, а так же последующая инициализация связи микроконтроллера MSP430F5528 с периферийным TDC7200 и TDC 1000 посредством дуплексного протокола SPI(TI_TDC7200_SPISetup();TI_TDC1000_SPISetup();).

```
void InitMCU(void)
{

    __disable_interrupt();
    init_port_pins();
    SetVCore(3);
    Init_Clock();
```

Рисунок 3. Функция InitMCU(void)

```
    USB_init();
    USB_setEnabledEvents(
            kUSB_VbusOnEvent + kUSB_VbusOffEvent + kUSB_receiveCompletedEvent
                    + kUSB_dataReceivedEvent + kUSB_UsbSuspendEvent
                    + kUSB_UsbResumeEvent +
                     kUSB_UsbResetEvent);

    //Check if we're already physically attached to USB, and if so, connect to it
    //This is the same function that gets called automatically when VBUS gets attached.
    if (USB_connectionInfo() & kUSB_vbusPresent)
    {
        USB_handleVbusOnEvent();
    }
```

Рисунок 4. Инициация USB

Функция” initMCU()” вызывалась в прошлой части кода . В данном фрагменте, представленном на рисунке 4.1 происходит отключение глобальных прерываний(__disable_interrupt();), инициация pin портов(init_port_pins();), инициацию часов(Init_Clock();). Далее происходит инициализация USB (USB_init();) и последовательный вызов набора функций для настройки и управления. После чего посредством команды “if” происходит проверка подключения по USB (USB_connectionInfo() & kUSB_vbusPresent), и запуск vbus(USB_handleVbusOnEvent();), что отвечает за распиновку USB.

Дальнейшая часть кода не приводится из-за его большого объёма. Данный код является проверочным, и выполняет функцию итоговой проверки перед запуском. В зависимости от параметров, или возможных сбоев в программной или аппаратной части, данный код либо начнет измерение, либо предпримет действие для устранения возникшего сбоя. Если ошибок не обнаружено, происходит либо одиночный запуск измерений.

```
(TDC1000_UART_Stream)

 if (!TDC1000_MSP430Timer_TDC)
     tdc7200_calc(ubuf);
 else
     msp430tdc_calc(ubuf);
```

Рисунок 5. Запуск измерений

В зависимости от параметра TDC1000_MSP430Timer_TDC происходит запуск измерений либо с использованием TDC7200(tdc7200_calc(ubuf);), либо на встроенном в микроконтроллере измерителе времени(msp430tdc_calc(ubuf);)

```
#define MSP430TIMER_TDC_CLK_PERIOD 1000/12.0

extern uint8_t outString[];

//---------------------------------------------------------------------------
void msp430tdc_calc(uint8_t *buf)
{
    uint16_t tst, tsp;
    uint16_t ovf;
    float st2sp;

    tst = buf[0] + (buf[1] << 8);
    tsp = buf[2] + (buf[3] << 8);
    ovf = buf[4];
#if
    st2sp = (((ovf * 65536) + tsp) - tst) * MSP430TIMER_TDC_CLK_PERIOD;

    sprintf((char *) outString, "Start_to_Stop1: %4.6f \r\n", st2sp);
    putsUART((unsigned char *) outString, strlen((char *) outString));

}
```

Рисунок 6. Измерение с использованием MSP430f5528

В результате выполнение данного кода происходит вычисление временной задержки с использованием встроенных в микроконтроллер часов, и запись результата в переменной(st2sp) , а так же последующая передача её по UART.

```
void tdc7200_calc(uint8_t *buf)
{
    tdc7200_build_mreg(buf);
    tdc7200_start2stopn();
    print_start2stopn();

}
```

Рисунок 7. Главная функция измерений на TDC7200

Функция, которую мы вызываем для расчётов временных интервалов, является лишь промежуточной. Из-за сложности работы с TDC7200 пришлось разделить код на несколько отдельных функций и поэтому tdc7200_calc(uint8_t *buf) выполняет роль последовательного вызова других функций.

```
void tdc7200_build_mreg(uint8_t *buf)
{
    uint8_t i;
    //mreg[0]  = Time1,      mreg[1] = Clk_count1, mreg[2] = Time2
    //mreg[3]  = Clk_count2, mreg[4] = Time3,      mreg[5] = Clk_count3
    //mreg[6]  = Time4,      mreg[7] = Clk_count4, mreg[8] = Time5
    //mreg[9]  = Clk_count5, mreg[10] = Time6,     mreg[11] = Calibration1
    //mreg[12] = Calibration2
    for (i = 0; i < MEAS_RESULT_REG_NUM; i++)
    {
        meas_result_regrs[i] = (((uint32_t) buf[3 * i]) << 16)
                + (((uint32_t) buf[(3 * i) + 1]) << 8)
                + (((uint32_t) buf[(3 * i) + 2]));
    }
```

Рисунок 8. Функция tdc7200_bui;d_mreg

```
void tdc7200_start2stopn(void)
{
    uint8_t n;

    tdc7200_normlsb();

    //mreg[0]  = Time1,      mreg[1] = Clk_count1, mreg[2] = Time2
    //mreg[3]  = Clk_count2, mreg[4] = Time3,      mreg[5] = Clk_count3
    //mreg[6]  = Time4,      mreg[7] = Clk_count4, mreg[8] = Time5
    //mreg[9]  = Clk_count5, mreg[10] = Time6,     mreg[11] = Calibration1
    //mreg[12] = Calibration2

    //st2sp[0] = nlsb * (Time1 - Time2) + tdc_clk_period * Clk_count1
    //st2sp[1] = nlsb * (Time1 - Time3) + tdc_clk_period * Clk_count2
    //st2sp[2] = nlsb * (Time1 - Time4) + tdc_clk_period * Clk_count3
    //st2sp[3] = nlsb * (Time1 - Time5) + tdc_clk_period * Clk_count4
    //st2sp[4] = nlsb * (Time1 - Time6) + tdc_clk_period * Clk_count5

    // do for 5 stops
    for (n = 0; n < MAX_STOPS; n++)
        start2stop[n] = norm_lsb
                * (meas_result_regrs[0] - meas_result_regrs[2 * (n + 1)])
                + tdc_clk_period * meas_result_regrs[2 * (n + 1) - 1];

}
```

Рисунок 9. Функция TDC7200_start2stapn

```
void tdc7200_normlsb(void)
{
    uint8_t tdcbyte, cal2cy;

    // read config2 register - use local saved data
    tdcbyte = (TDC7200_reg_local_copy[TI_TDC7200_CONFIG2_REG] & CAL2MASK)
            >> CAL2SHFT;
    switch (tdcbyte)
    {case 0:
        cal2cy = 2;
        break;
    case 1:
        cal2cy = 10;
        break;
    case 2:
        cal2cy = 20;
        break;
    case 3:
        cal2cy = 40;
    default:
        cal2cy = 10;
    }
    ccnt = (float) (meas_result_regrs[12] - meas_result_regrs[11])
            / (cal2cy - 1);
    norm_lsb = tdc_clk_period / ccnt;
}
```

Рисунок 10. Функция tdc7200_normlsb

Код приведённый на рисунках 8 – 10 производит серию вычислений для расчёта временного интервала. На изображении 8.3 происходит перевод в другую систему измерения, а на 8, 9 – итоговый расчёт. Представленный код лишь производит вычисление, но не передаёт результат.

```
void print_start2stopn(void)
{    uint8_t i;
if 1
    for (i = 0; i < MAX_STOPS; i++)
    {
        sprintf((char *) outString, "Start_to_Stop:     ", i + 1,
                start2stop[i]);
        putsUART((unsigned char *) outString, strlen((char *) outString));
    }
    sprintf((char *) outString, " \r\n");
    putsUART((unsigned char *) outString, strlen((char *) outString));
}
```

Рисунок 11. Вывод результатов расчёта через UART

Функция print_start2stopn() выполняет роль передачи полученных временных интервалов, в зависимости от их колличества(MAX_STOPS).

Результатом работы программы является передача числа, обозначающая временной интервал между испускаемой и поглощаемой волной. Представленная программа выполняет лишь некоторые функции - установку параметров, запуск измерений, вывод их по протоколу UART.

В разделе 2 проводится доработка программного обеспечения ПО и практическая проверка эффективности использования TDC7200.

Необходимость доработки связана с тем, что рассмотренная в разделе 1 версия имеет недостатки. Данный код при запуске сразу производит измере-

ние, основываясь на начальных параметрах. Также отсутствует возможность менять параметры при помощи программы на ПК. Таким образом, для изменения параметров работы требуется постоянная перепрошивка микроконтроллера. Это является большой проблемой для практического применения данной работы.

Таким образом целью доработки является рассмотрение возможности взаимодействия с микроконтроллером по USB, как для изменения начальных параметров, так и для получение команды на проведение измерений(отключение автоматического старта измерений, ожидание команды от ПК)

Решением может являться изменение готового кода для работы с API функциями библиотеки USB. Тогда, используя готовые функции, можно будет изменять переменные данной программы. Для решения проблемы начального запуска возможно создание вечного цикла, проверяющего команду СТАРТ с определённым временным интервалом.

Данную проверку можно реализовать добавив в самое начало кода цикл с проверкой на команду, отвечающую за запуск измерений, и изменяющуюся с ПК

Изменение переменной “start” посредством USB на значение “ true” начнёт выполнение кода в зависимости от заданных переменных. Таким образом мы получаем возможность в любой момент не выполняемой программы изменять переменные, и запускать измерения с ПК.

Так как мы уже подключили все библиотеки для взаимодействия по USB, а все переменные подразумевают возможность их удалённого изменения, то реализация работы с ПК лежит не на стороне микроконтроллера.

В этой части работы осуществляется практическая проверка эффективности использования связки TDC1000 и TDC 7200, вместо использования встроенного в микроконтроллер измерителя времени. Для практического сравнения мы сначала будем рассчитывать временные интервалы, используя TDC7200, а потом только микроконтроллер. Для того чтобы провести такие измерения мы немного изменим код, проведя серию отдельных измерений с небольшой паузой между ними. Для этого мы изменим код на рисунке 5, убрав проверку “if” и добавив цикл:

```
for (i = 0; i <= 3; i++)

{ tdc7200_calc(ubuf);Pause(1);msp430tdc_calc(ubuf);pause(3);}
```

Данная последовательность команд приведёт к серии из 3 измерений поочерёдно каждым методом. Далее мы рассчитаем расстояние между датчиком и объектом(l) посредством формулы l=0,5*t*v, где v – скорость звука в среде. Результаты этих измерений записаны в следующую таблицу:

l, см	l_1, см	Δl_1, см	$\lvert\Delta l_1 - l_1\rvert$, см	$\Delta\lvert\Delta l_1 - l_1\rvert$, см	l_2, см	Δl_2, см	$\lvert\Delta l_2 - l_2\rvert$, см	$\Delta\lvert\Delta l_2 - l_2\rvert$, см
3	3.03	2.99	0.04	0.035	3.15	2.96	0.18	0.113
	2.96		0.03		2.94		0.02	
	2.99		0		2.8		0.14	
4	4.05	4.01	0.04	0.083	3.88	4.06	0.18	0.12
	3.88		0.13		4.18		0.12	
	4.10		0.09		4.12		0.06	
5	4.9	5.00	0.1	0.046	5.24	5.13	0.11	0.076
	5.12		0.12		5.02		0.11	
	4.99		0.1		5.12		0.01	
7	7.14	7.04	0.1	0.07	6.68	6.88	0.2	0.2
	7.04		0		7.24		0.36	
	6.95		0.11		6.82		0.06	

l_1 – данные с TDC7200, l_2 – с встроенного в MSP430 измерителя времени.

Рассматривая данные таблицы и проведя несложные расчёты, мы получим, что отклонение от средней погрешности при использовании связки TDC1000-7200 составляет 1.16%, 2.075%, 0.92%, 0.01% соответственно. Таким образом, средняя погрешность при использовании TDC7200 составила 1.04%.

В случае с использованием для измерения временных интервалов MSP430f5528 мы получаем 3.767%, 3%, 1.52%, 2.857%. Средняя погрешность составляет 2.79%. Сравнивая результаты, сразу заметно, что использование TDC1000 и TDC7200 даёт более чем в 2.5 раза точный результат, чем в системах, где используется встроенный в микроконтроллер измеритель времени. Это особенно актуально в системах, где требуется очень точное измерение, например счётчиках газа или воды.

ВЫВОДЫ

В данной работе проведены исследования программного обеспечения для платы TDC1000-7200EVM. В ходе этой работы была исследованы основные принципы работы микроконтроллера MSP430F5528, а так же компонен-

тов платы - TDC1000 и TDC 7200, изучены базовые принципы работы с линейкой микроконтроллеров MSP430x5xx, и его взаимодействие его с периферийными устройствами, рассмотрена возможность прямого управления микроконтроллера посредством персонального компьютера.

Результатом работы является редактирование ПО для микроконтроллера, с целью вывода данных на персональный компьютер, и последующим сравнением эффективности использования разных методов измерения временных интервалов отражённых волн. Результатом сравнения была подтверждена эффективность использования TDC7200 для измерения временных интервалов, относительно использования микроконтроллера.

Список используемых источников

1. http://www.ti.com/tool/tdc1000-tdc7200evm?keyMatch=tdc1000-tdc7200EVM
2. http://www.ti.com/product/TDC1000?keyMatch=tdc1000&tisearch=Search-EN-Products
3. Семенов Б.Ю. Микроконтроллеры MSP430. Первое знакомство, М.: Изд-во «Солон-пресс», 2006.
4. http://www.ti.com/lit/sg/slab055/slab055.pdf
5. http://www.ti.com/lit/an/snaa284a/snaa284a.pdf
6. http://www.ti.com/lit/ds/symlink/tdc7200.pdf
7. http://www.ti.com/lit/ug/sniu021a/sniu021a.pdf
8. https://www.compel.ru/lib/ne/2009/11/4-msp430f5xxx-samyie-energoekonomichnyie-v-mire-mikrokontrolleryi-s-usb-interfeysom
9. https://habr.com/ru/post/154155/
10. http://we.easyelectronics.ru/blog/msp430
11. http://mspsci.blogspot.com/2010/07/tutorial-02-msp430-township-and.html
12. http://www.ti.com/lit/zip/snac064
13. http://www.ti.com/lit/an/snaa284a/snaa284a.pdf
14. http://www.ti.com/lit/an/snaa230/snaa230.pdf
15. http://www.ti.com/lit/an/snaa220a/snaa220a.pdf
16. http://www.ti.com/lit/an/snia020/snia020.pdf
17. https://www.alldatasheet.com/datasheet-pdf/pdf/390620/TI/MSP430F5529IPN.html
18. https://habr.com/ru/posl/137205/