После этого, генерируем файл, ответственный за программную часть ViewController. В этом файле будут производиться все процессы и процедуры, которые необходимы для корректной работы данного View.

По предыдущему же паттерну, разрабатываются оставшиеся файлы программного кода и требующиеся представления, обеспечивающие функционал приложения.

***

1. Структура и интерпретация компьютерных программ / Харольд Абельсон, Джеральд Сассман - Добросвет, КДУ, 2018. - 608 с.
2. Cocoa Design Patterns / Eric Buck, Donald Yactman, 2009. - 456 с.
3. Core Data Programming Guide: Persistent Store Types and Behaviors [Электронный ресурс] / Apple Inc. – Режим доступа: <https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/CoreData/>. – Загл. с экрана. – яз. англ.
4. Foundation Functions Reference – Apple Developer [Электронный ресурс] / Apple Inc. – Режим доступа: https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Miscellaneous/Foundation_Functions/. – Загл. с экрана. – яз. англ.
5. GitHub – GitHub [Электронный ресурс] / GitHub Inc. – Режим доступа: <https://github.com/>. – Загл. с экрана. – яз. англ.
6. iOS 11 App Development Essentials / Neil Smyth, 2018. - 812 с.
7. iOS Apprentice / Matthijs Hollemans, 2013. - 816 с.
8. iOS Programming: The Big Nerd Ranch Guide / Christian Keur, Aaron Hillegass, 2010. - 416 с.
9. iPhone App Development: The Missing Manual / Craig Hockenberry, 2010. - 352 с.
10. Model-View-Controller - Википедия [Электронный ресурс] / Wikimedia Foundation, Inc. – Режим доступа: <https://ru.wikipedia.org/wiki/Model-View-ControllerMVC>. – Загл. с экрана. – яз. рус., англ.
11. Pro Swift / Paul Hudson, 2016. - 147 с.
12. Push-уведомления - Википедия [Электронный ресурс] / Wikimedia Foundation, Inc. – Режим доступа: http://ru.wikipedia.org/wiki/Push_уведомления. – Загл. с экрана. – яз. рус., англ.
13. Госдума приняла закон об изучении родных языков <http://www.segodnia.ru/print/204564>

Каргин М.С., Сагаева И.Д.

Реализация программной множественной тахометрии на микроконтроллерах

*Саратовский государственный университет
(Россия, Саратов)*

doi: 10.18411/lj-06-2021-61

Аннотация

Статья посвящена практической реализации программной тахометрии для нескольких каналов. В статье сравнивается аппаратный и программный подход к измерению частоты сигнала, а также описан блок кода на языке C/C++, реализующий программный подход. В заключение приведен способ определения рациональности использования программного подхода перед аппаратным.

Ключевые слова: тахометрия, частотомер, микроконтроллеры, алгоритмы.

Abstract

The article is devoted to the practical implementation of software tachometry for several channels. The article compares the hardware and software approaches to measuring the signal frequency, and also describes a block of C / C ++ code that implements the software approach. In conclusion, a method for determining the rationality of using the software approach over the hardware one is presented.

Keywords: tachometry, frequency meter, microcontrollers, algorithms.

Для различных производственных и бытовых задач часто возникает потребность произвести измерение частоты некоторого физического процесса. Существует

множество классификаций таких задач: измерение синусоидальных колебаний, гармонических составляющих, дискретных событий. В рамках статьи будет рассмотрена задача по измерению частот многих дискретных сигналов при помощи одной аппаратной платформы (контроллера).

Практически задача возникла во время разработки устройства, одной из функций которого являлся замер скорости вращения нескольких разных двигателей, у каждого из которых был отдельный дискретный канал обратной связи.

Классическая реализация тахометрии является аппаратной: измеряемый канал соединяется с генератором внешних прерываний контроллера, и во время прохождения фронта вызывается аппаратное прерывание. Частота дискретного сигнала измеряемого канала обратно связана с разностью значений внутреннего таймера контроллера между двумя вызовами прерываний и вычисляется по формуле:

$f_m = f_s / (2 * ps * (t_2 - t_1))$, где

f_m (measured frequency) – частота измеряемого сигнала

f_s (sync frequency) – тактовая частота контроллера

t_1 – значение регистра состояния таймера в момент прохождения первого фронта

t_2 – значение регистра состояния таймера в момент прохождения второго (противоположного) фронта

ps (prescaler) – множитель таймера, отображающий количество тактов, совершаемых контроллером между инкрементами таймера.

Так, например, если таймер с предделителем 16 поймал два фронта на значениях регистра таймера 1000 и 9000, а тактовая частота составляла 8 МГц, то частота измеряемого сигнала равна 62,5 Гц.

Проблема измерения частоты нескольких дискретных каналов заключается в том, что контроллер имеет ограниченное число генераторов внешних прерываний, как правило один или два. Если количество каналов превышает количество доступных для использования внешних прерываний, то существует два способа реализовать множественную тахометрию: аппаратный (использование мультиплексоров) и программный. Первый способ не является предметом данной статьи, поэтому рассмотрим второй способ.

Для конечного вычисления частоты будем также использовать формулу, приведенную выше. Необходимо завести прерывание по таймеру, частота вызова которого будет достаточной для периодических измерений: так, например, если сигнал от десяти источников необходимо актуализировать раз в секунду, то частоту прерываний по таймеру следует выставить равной 100 Гц. Слишком высокая частота вызова прерываний будет ограничивать возможности программного фильтра ошибок и минимально возможную измеряемую частоту сигнала, а слишком низкая частота вызова прерываний будет увеличивать период измерений частоты одного канала.

Один раз за прерывание будет измеряться частота сигнала одного канала (входа контроллера). Для максимальной точности необходимо провести синхронизацию: запомнить состояние канала и запустить цикл отслеживания его изменения на противоположное состояние. Для того, чтобы ограничить этот цикл и предотвратить зависание, необходимо добавить условие по допустимому времени замера на канал: если регистр значения внутреннего таймера превысил некоторое заданное нами число, то считаем, что частота равна нулю и выходим из цикла.

Если смена состояния сигнала всё таки произошла за отведенное время, то этот момент и будет точкой синхронизации: необходимо занести значение внутреннего таймера в переменную t_1 . Далее в таком же цикле ожидаем следующего фронта и заносим его значение в переменную t_2 . Подставив полученные значения t_1 и t_2 в формулу получим частоту дискретного сигнала на измеряемом канале.

Для того, чтобы избежать грубых ошибок измерений, следует ввести хотя бы простейший программный фильтр, который будет вычислять среднее значение частоты. Вычисление среднего значения на основе единичных замеров может занимать слишком много времени, поэтому рациональнее посчитать время нескольких фронтов и уже после поделить разность значений таймера на количество фронтов. Количество фронтов программного фильтра выбирается исходя из максимально допустимого времени замера на канал. Результат программного фильтра помещается в массив значений частот каналов на соответствующую каналу позицию и уже может использоваться в основном коде.

Пример реализации для десяти каналов (измерение оборотов двигателей), $f_m = 100..250$ Гц, $f_s = 8 \cdot 10^6$ Гц, $p_s = 2$:

```
//данный фрагмент кода выполняется в прерывании по таймеру
//int counter – количество дискретных смен уровня
//measure – текущий дискретный уровень
//readFB(pin) – глобальная функция, возвращающая текущий дискретный
уровень канала «pin»
//chNum – глобальная переменная, номер исследуемого канала
//TCNT3 – регистр текущего значения аппаратного таймера
//point – переменная для хранения промежуточного значения TCNT3
//TCNT_limit – define, 1500, количество тиков таймера, после которого считаем,
что текущая частота равна нулю
//fbMas[10] – массив результатов замеров частоты десяти каналов
//по первому замеру происходит синхронизация
counter = 1;
//делаем замер
measure = readFB(chNum);
//ждем фронта
while (readFB(chNum) == measure) if (TCNT3 > 1500) break;
//запоминаем состояние таймера на момент первого фронта
point = TCNT3;
//т.к. был фронт, то состояние замера поменялось
measure = !measure;
//отсчитываем еще 9 фронтов
while (counter < 10)
{
    if (readFB(chNum) != measure)
    {
        counter++;
        measure = !measure;
    }
    //защита от зависания если частота менее 100Гц
    if (TCNT3 > TCNT_limit) break;
}
//если частота выше минимума, то вычисляем частоту и записываем в массив
if (TCNT3 < TCNT_limit) fbMas[chNum] = int(156200/(TCNT3-point)/4);
//иначе записываем в массив, что частота равна нулю
else fbMas[chNum] = 0;
//в следующем прерывании будем смотреть частоту следующего канала
chNum++;
//если каналы кончились, начинаем с первого
if (chNum > 10) chNum = 1;
//конец сегмента кода
```

«Чистые» вычисления исполняемого кода внутри прерывания по таймеру (без программного фильтра) занимают до 100 тактов однопоточного АЛУ. Отсюда вычисляется максимальная измеряемая частота сигнала $\max(fm) = fs/(100 \cdot n)$, где n – порядок фильтра.

Минимальное значение частоты измеряемого сигнала ограничена периодом вызова прерываний по таймеру (100Гц в приведенном примере) и настраивается исходя из условий решаемой задачи. Для определения необходимо установить количество каналов N , а также ft (transmit frequency) – то, с какой минимальной частотой нам необходимо получать информацию о сигналах. В таком случае определим $\min(fm) = ft \cdot N$. В случае, если код не фиксирует дважды смены дискретного значения за время $1/\min(fm)$, то считается, что $fm = 0$.

Для наглядности представим примеры зависимости $\max(fm)$ и $\min(fm)$ в виде таблицы 1 и таблицы 2 соответственно:

Таблица 1

Зависимость $\max(fm)$ от fs и n

Model	fs, kHz	n	max(fm), kHz
Atmega 328 (int osc)	4000	1	40
Atmega 328 (int osc)	4000	10	4
Atmega 328 (ext osc)	16000	1	160
Atmega 328 (ext osc)	16000	10	16
STM32 Cortex-M7F	480000	1	4800
STM32 Cortex-M7F	480000	10	480

Таблица 2

Зависимость $\min(fm)$ от N и ft

N	1	1	1	10	10	10	100	100	100
ft, Hz	1	10	100	1	10	100	1	10	100
min(fm), Hz	1	10	100	10	100	1000	100	1000	10000

Вышеуказанные данные можно применить для оценочного определения возможности использовать предлагаемый метод для реализации конкретной задачи.

***

1. Частотомер [Электронный ресурс]: Материал из Википедии — свободной энциклопедии. –URL: <https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%BE%D0%BC%D0%B5%D1%80> (дата обращения 27.05.2021). - Загл. с экрана. –Яз. рус.
2. Atmel ATmega328 [Электронный ресурс] / Microchip [Электронный ресурс] : Datasheet. -URL: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (дата обращения 27.05.2021). - Загл. с экрана. -Яз. англ.
3. STM32F7 Series and STM32H7 Series Cortex®-M7 processor [Электронный ресурс] : PM0253 Programming manual. -URL: https://www.st.com/resource/en/programming_manual/dm00237416-stm32f7-series-and-stm32h7-series-cortexm7-processor-programming-manual-stmicroelectronics.pdf (дата обращения 27.05.2021). - Загл. с экрана. -Яз. англ.

Кочкарова П.А., Катчиева А.К.

Факторы, оказывающие влияние на цифровизацию банковской деятельности

*Северо-Кавказская государственная академия
(Россия, Черкесск)*

doi: 10.18411/lj-06-2021-62

Аннотация

Цифровизация банковской деятельности является современным трендом в развитии кредитных организаций. Российские банки активно участвуют в этом