# Проектирование на программируемых системах на кристалле PSoC Cypress. Часть 6. Клавиатуры, выполненные по технологии CapSense

**Дмитрий КИЛОЧЕК**
Dmitry.Kilochek@macrogroup.ru

**Продолжая цикл статей [1], посвященных программируемым системам на кристалле PSoC фирмы Cypress, расскажем об одном из популярных применений данных систем, а именно об обработке сигналов с емкостных датчиков прикосновения и приближения — технологии CapSense.**

Прежде чем приступать к рассмотрению технологии CapSense, вкратце рассмотрим теоретические основы измерений сигналов с емкостных датчиков прикосновения.

Любой проводящий объект, имеющий заряд *Q*, создает вокруг себя электрическое поле. Потенциал поля *Ф* прямо пропорционален заряду:

$$Ф = Q/C,$$

где емкость *C* — коэффициент, связывающий потенциал объекта с его зарядом и зависящий только от геометрических свойств объекта и от окружающего его диэлектрика. Собственную емкость имеет любое проводящее тело. Так, например, емкость человеческого тела составляет 100–300 пФ.

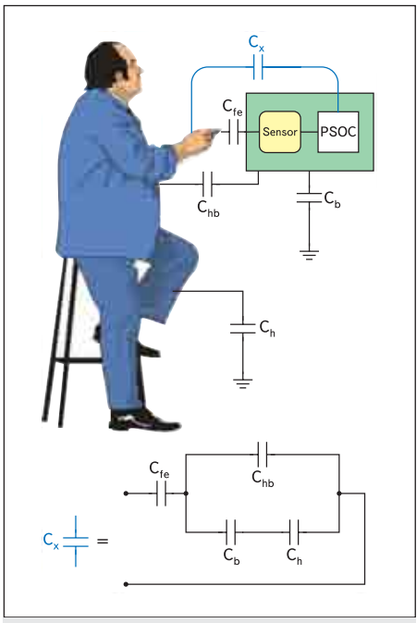


**Рис. 1.** Емкость прикосновения

При измерении емкости сенсора следует принять во внимание следующие компоненты (рис. 1):

- Емкость «палец–сенсор» *Cfe* имеет значения 0,1–10 пФ в зависимости от размеров электрода и толщины покрытия, а также диэлектрической постоянной электрода.
- Емкость тела человека *Ch* — 100–300 пФ.
- Емкость платы *Cb* — 10–20 пФ для небольших плат без внешних подключений и свыше 1000 пФ, если устройство подключено, например, к внешнему блоку питания.
- Емкость «человек–плата» *Chb* — примерно 1–20 пФ в зависимости от размеров, толщины покрытия и положения рук.

Емкость прикосновения *Cx* представляет собой последовательное соединение емкости *Cfe* с эквивалентной емкостью заземления:

$$Cg = Chb + (Ch \times Cb)/(Ch + Cb),$$
$$Cx = (Cg \times Cfe)/(Cg + Cfe).$$

Из этого выражения видно, что емкость прикосновения *Cx* меньше емкости «палец–сенсор» *Cfe*. В большинстве случаев разница невелика, однако об этом уравнении надо помнить в случае, если в устройстве предполагается использование внешних дополнительных подключений, которые могут изменить эквивалентную емкость *Cg*.

Таким образом, задачей системы CapSense является определение небольших изменений емкости прикосновения (*Cx* = 0,1–10 пФ) при наличии собственной большой паразитной емкости сенсора (10–300 пФ).

Паразитную емкость составляют емкости электродов сенсоров, емкость между сенсором и полигонами «земли» на плате, взаимная емкость соединений на поверхности печатной платы, емкость контакта PSoC. Емкость сенсора, измеряемая PSoC, равна сумме емкости прикосновения и паразитной емкости:

$$Cs = Cx + Cpar.$$

В библиотеке пользовательских модулей в PSoC Designer существуют три элемента, предназначенных для определения изменения емкости сенсоров, выполненных в виде проводящих полигонов на печатной плате. Такими модулями являются модули CSR, CSD и CSA.

## Модуль CSR

Модуль CSR (Capacitive Sensor Relaxation Oscillator) является первым модулем, созданным для измерения емкости для технологии CapSense [2, 3]. Он представляет собой релаксационный генератор (Relaxation Oscillator) в совокупности со схемой преобразования «частота–код» (рис. 2).

Компонентами генератора являются емкость сенсора, источник тока, компаратор и ключ для сброса. Напряжение на конденсаторе, образованном проводящей структурой сенсора, изменяется по следующему закону:

$$v(t) = (Ic \times t)/C,$$

где *Ic* — ток заряда.

Это линейно нарастающее напряжение сравнивается компаратором с пороговым значением *Vbg*, образованным внутренним генератором опорных напряжений. Время, необходимое для того, чтобы напряжение на конденсаторе достигло порога, определяется следующей формулой:

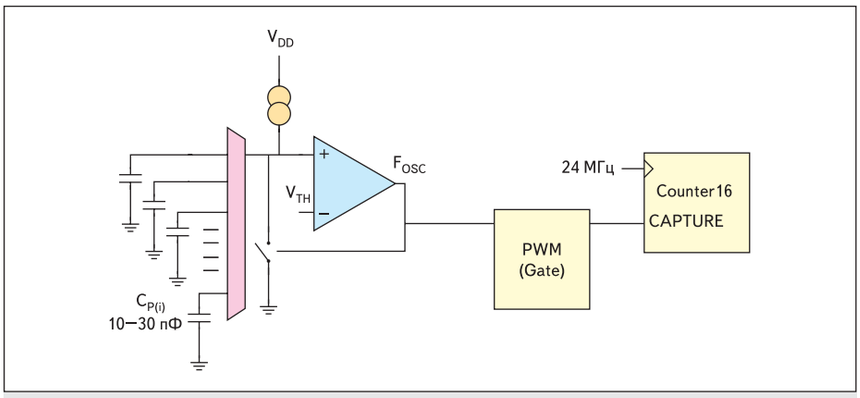


**Рис. 2.** Блок-схема модуля CSR

$$t = (Cx \times Vbg)/Ic.$$

Когда напряжение достигает порогового уровня, компаратор сбрасывает напряжение на конденсаторе. Таким образом, частота работы такого генератора равна:

$$f = Ic/(C \times Vbg).$$

Временная диаграмма работы показана на рис. 3.

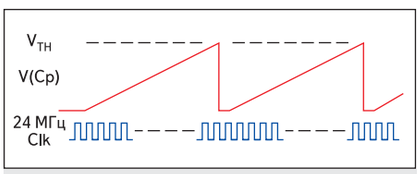


**Рис. 3.** Временная диаграмма работы модуля CSR

Когда проводящий объект (палец) прикасается к сенсору, емкость сенсора увеличивается и, следовательно, уменьшается частота генератора. Импульсы от генератора поступают на ШИМ, с выхода которого сигнал подается на разрешающий вход счетчика импульсов системной частоты (обычно 24 МГц). Таким образом измеряется несколько периодов частоты релаксационного генератора. Этот подход позволяет увеличить чувствительность и разрядность.

При первом запуске все сенсоры сканируются, и измеренное в отсутствие прикосновения значение сохраняется в качестве базового (**baseline**). В дальнейшем по разнице измеренного значения относительно базового принимается решение о наличии прикосновения.

Для того чтобы определение прикосновения было надежным, необходимо, чтобы собственная емкость сенсора, а также относящиеся к ней паразитные емкости (емкость соединительного проводника, емкость контакта микросхемы и т. п.) были как можно меньше. Кроме того, требуется довольно длительное время измерения (длительность импульса ШИМ), чтобы разница между базовым значением и измеренным стала достаточно ощутимой.

Модуль CSR, реализуя довольно простой алгоритм, тем не менее, не показал достаточной надежности. Для его замены были предложены два других модуля — CSD и CSA, имеющие улучшенные характеристики [5–7]. Модуль CSD реализуется на микросхемах тех же серий, что и CSR — CY8C21x34 и CY8C24x94. Для модуля CSA была специально разработана серия CY8C20x34. В отличие от CSR, оба модуля требуют подключения внешних компонентов.

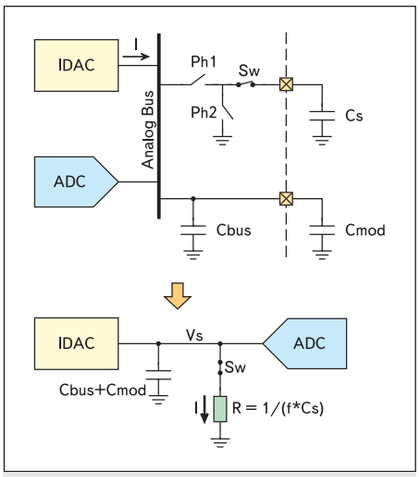


**Рис. 5.** Схема работы модуля CSA

## Модуль CSA

Модуль CSA (CapSense Successive Approximation) использует для измерения преобразование емкости в напряжение и оцифровку полученного значения с помощью АЦП (рис. 4).

Конденсатор *Cs*, образованный сенсором, периодически либо заряжается от аналоговой шины, либо разряжается на «землю». Переключаемый конденсатор образует цепь с эквивалентным сопротивлением, через который стекает заряд с параллельно включенных собственной емкости аналоговой шины *Cbus* и подключаемого внешнего конденсатора *Cmod* (рис. 5). Напряжение на этом резисторе определяется следующим выражением:

$$Vs = I \times 1/(f \times Cs),$$

где $f$ — частота переключения конденсатора, равная системной частоте.

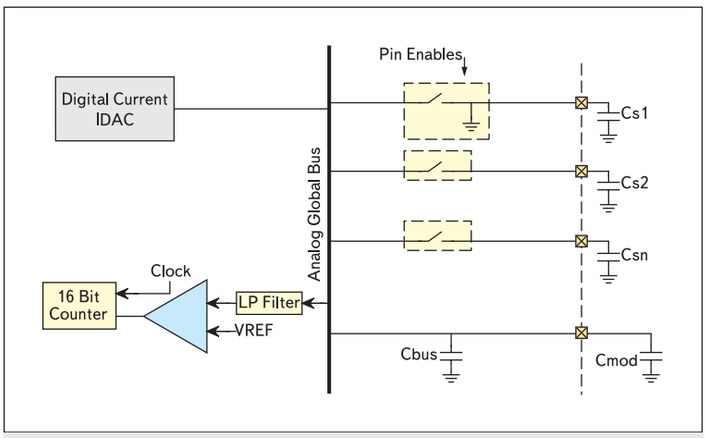


**Рис. 4.** Блок-схема модуля CSA

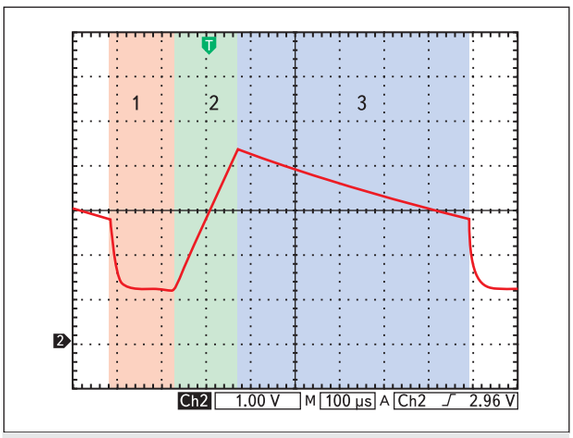


**Рис. 6.** Осциллограмма напряжения на конденсаторе *Cmod*

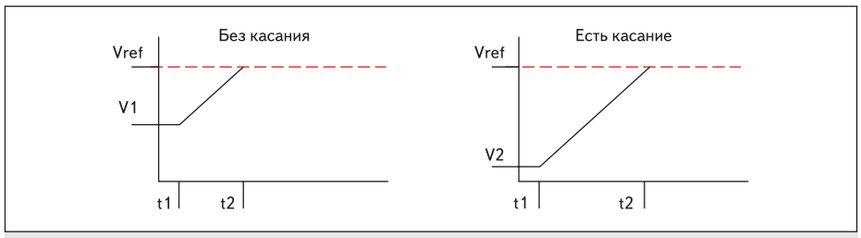


**Рис. 7.** Определение прикосновения

Конденсаторы *Cbus* и *Cmod* выступают в роли фильтрующих, сглаживая уровень напряжения *Vs*. Большее значение *Cmod* означает лучшую стабильность результата, но и требует большего времени на переходный процесс, поэтому величина *Cmod* варьируется в зависимости от требуемой скорости работы системы.

После того как переключаемый конденсатор прошел через некоторое число циклов заряда-разряда, достаточное для установления напряжения (этап 1 на рис. 6), он отключается от аналоговой шины (ключ *Sw*), и сохраненное конденсаторами *Cbus* и *Cmod* напряжение подвергается оцифровке (этап 2).

Напряжение на *Cbus+Cmod* начинает нарастать благодаря тому, что они остаются подключенными к источнику IDAC. Время, необходимое для достижения уровня *Vref* и выраженное в тактах системной частоты, является результатом измерения (рис. 7).

Данный метод показывает лучшие характеристики по сравнению с CSR, но в отличие от рассматриваемого далее модуля CSD, не обладает встроенными средствами для обеспечения надежной работы в условиях повышенной влажности, когда на клавиатуре могут образовываться капли воды или водяная пленка. Также этот метод, в отличие от CSD, не позволяет использовать в качестве сенсоров проводящие материалы с высоким сопротивлением.

## Модуль CSD

Метод измерения емкости, используемый в модуле CSD (Capacitive Sensing using a Sigma-Delta Modulator) (рис. 8, 9), заключается в следующем. Конденсатор *Cs*, образованный сенсором, выступает в роли переключаемого конденсатора и образует эквивалентную схему резистора *Rc*, через который заряжается модулирующий конденсатор *Cmod*.

В качестве источника частоты для управления ключами, связанными с переключаемым конденсатором, могут использоваться три варианта схемы:

- 16-разрядный генератор псевдослучайных чисел — PRS16;
- 8-разрядный PRS (только в CY8C24x94) — PRS8;
- PRS8 с предварительным делителем частоты.

Использование 16-разрядного генератора псевдослучайных чисел позволяет расширить спектр сигнала, управляющего ключами, и, как следствие, получить лучшую помехозащищенность и уменьшить уровень электромагнитного излучения. Этот вариант рекомендуется, если разрабатываемое устройство должно проходить тесты на электромагнитную совместимость или надежно работать в условиях внешних электромагнитных помех.

Схема тактирования с 8-разрядным PRS использует только один аппаратный цифровой блок, но за счет более короткой последовательности псевдослучайных чисел имеет худшую помехоустойчивость, чем в первом случае.

Схема с 8-разрядным PRS и 8-разрядным счетчиком в качестве делителя частоты позволяет легко регулировать скорость работы. Основное применение данной схемы — измерение емкости при использовании в качестве сенсоров материалов с высоким сопротивлением, например, токопроводящих тонких прозрачных пленок, используемых в сенсорных дисплеях. Так как для правильного определения емкости сенсор должен быть полностью заряжен и разряжен за один такт (фазы $Ph_1$ и $Ph_2$), то для материалов с высоким сопротивлением работа на высоких частотах невозможна. Эта схема применяется и в том случае, когда требуется низкая частота, например для снижения энергопотребления и электромагнитного излучения.

Схема сигма-дельта модулятора приведена на рис. 10. Когда напряжение *Vcmod* на модулирующем конденсаторе достигает опор-

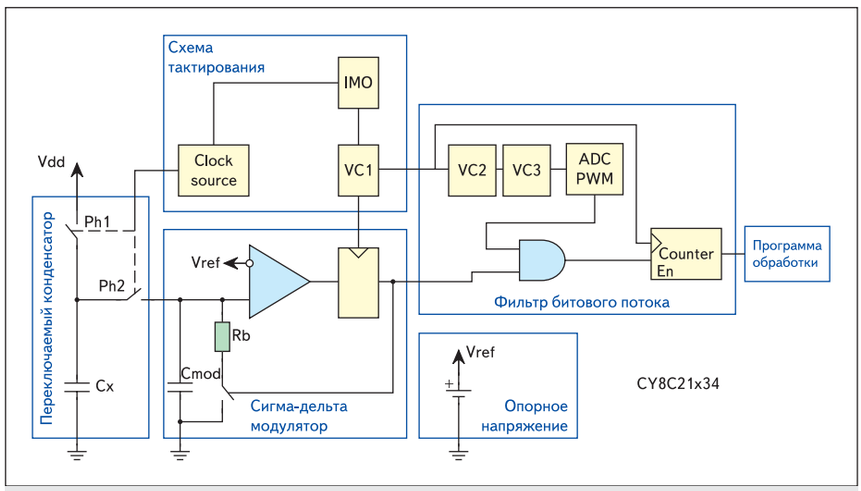


**Рис. 8.** Блок-схема модуля CSD для серии CY8C21x34

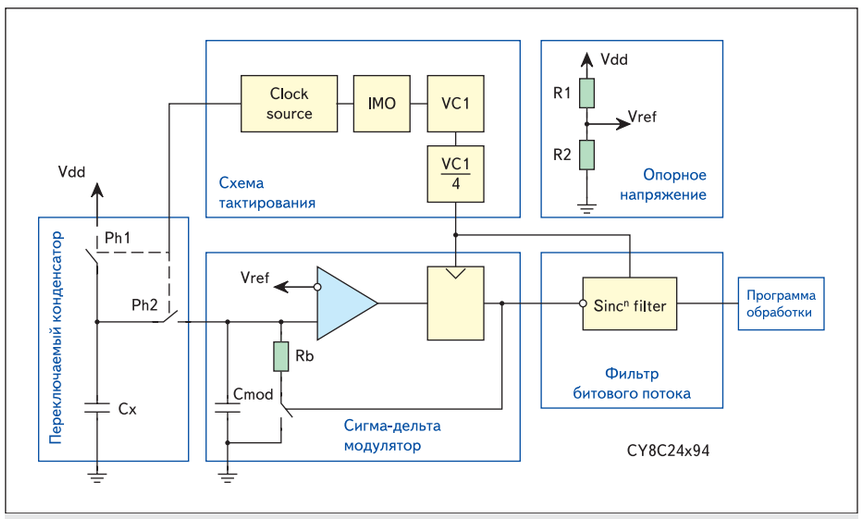


**Рис. 9.** Блок-схема модуля CSD для серии CY8C24x94

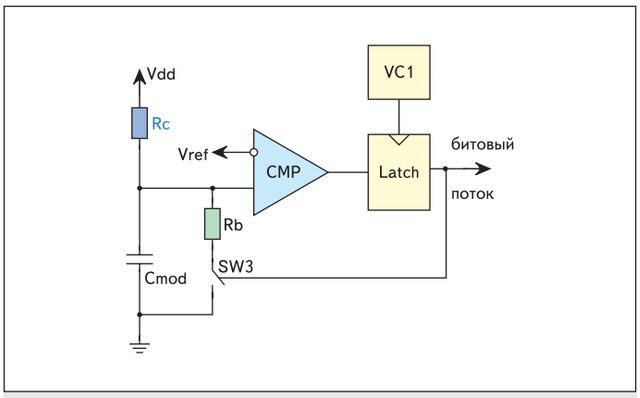


**Рис. 10.** Сигма-дельта модулятор модуля CSD

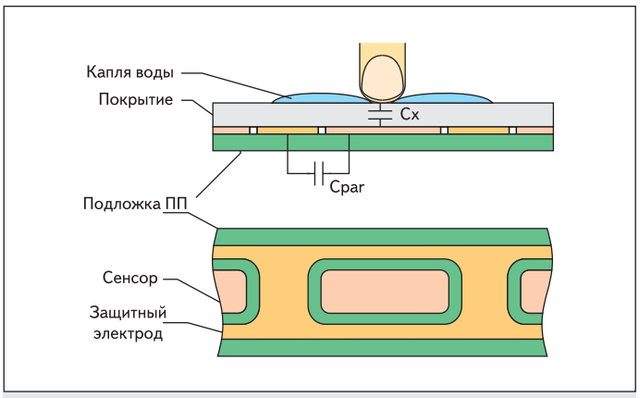


**Рис. 11.** Защитный электрод

ного значения *Vref*, компаратор срабатывает и разряжает конденсатор через резистор *Rb*. Разряд прекращается, если напряжение падает ниже уровня *Vref*. Затем цикл повторяется. Триггер, входящий в состав компаратора, ограничивает минимальное время заряда и разряда. Таким образом модулятор поддерживает напряжение *Vcmod* как можно более близким к *Vref*.

Ток через «виртуальный» резистор *Rc* определяется следующим выражением:

$$Ic = fs \times Cs \times (Vdd - Vcmod).$$

Так как этот ток зависит от напряжения питания *Vdd*, то и опорное напряжение *Vref* должно быть пропорционально *Vdd*:

$$Vref = k \times Vdd.$$

Это необходимо для того, чтобы результат вычислений в меньшей степени зависел от нестабильности питающего напряжения.

Ток разряда через резистор *Rb* зависит от скважности *d* выходного сигнала сигма-дельта модулятора:

$$Irb = (d \times Vcmod)/Rb.$$

Поскольку модулятор стремится поддерживать напряжение *Vcmod* равным *Vref*, то токи заряда и разряда также оказываются равными. Исходя из этого, можно получить уравнение следующего вида:

$$fs \times Cs \times (Vcmod/k - Vcmod) = \\ = (d \times Vcmod)/Rb,$$

откуда

$$d = fs \times Cs \times Rb \times (1/k - 1).$$

Таким образом, именно скважность *d* и несет информацию об измеряемой емкости.

Величина скважности измеряется при помощи цифрового фильтра. Поскольку серия CY8C21x34 не имеет выделенного блока децимации, то в версии модуля CSD для этой серии (рис. 8) фильтр реализуется с помощью 8-разрядного аппаратного счетчика и схемы, задающей интервал измерения. При этом для экономии аппаратных ресурсов часть действий реализуется программно. Так, например, счетчик расширяется до 16 разрядов благодаря программной обработке прерывания по переполнению. В серии CY8C24x94 аппаратный блок децимации используется для реализации цифрового SincN фильтра (рис. 9). Выход фильтра является значением, пропорциональным измеряемой емкости.

Опорное напряжение компаратора определяет чувствительность, так как скважность модулятора обратно пропорциональна опорному напряжению. В качестве источника опорного напряжения может выступать несколько вариантов схем.

Для серии CY8C21x34:

- внутренний источник, основанный на величине запрещенной зоны, *Vbg* = 1,3 В;
- аналоговый фильтр на блоке с переключаемыми конденсаторами для сигнала от PRSPWM или от предделителя;
- внешний резистивный делитель;
- внешний RC-фильтр сигнала от PRSPWM или от делителя частоты.

В микросхемах серии CY8C24x94 используется резистивный делитель внутри аналогового блока CT.

Первый вариант является простым решением, но показывает плохие результаты, если питающее напряжение нестабильно. Во всех остальных вариантах опорное напряжение пропорционально напряжению питания, и поэтому выход сигма-дельта модулятора мало зависит от нестабильности питания.

Как уже упоминалось выше, модуль CSD имеет встроенные средства, позволяющие минимизировать влияние возможного попадания на клавиатуру капель воды и водяной пленки на результаты сканирования сенсоров [13]. Для этого используется дополнительный полигон металлизации, окружающий сенсоры — защитный электрод (рис. 11). Когда капли воды или водяная пленка находятся на поверхности клавиатуры между сенсором и защитным электродом, взаимосвязь между ними увеличивается, и, соответственно, увеличивается паразитная емкость *Cpar*.

Для управления защитным электродом используется тот же тактирующий сигнал, что и для ключей, управляющих зарядом сенсора *Cs* (рис. 12).

Ток модулятора:

$$Imod = Ics - Icpar = \\ = fs \times Cs(Vdd - Vcmod) - f \times Cpar \times Vcmod.$$

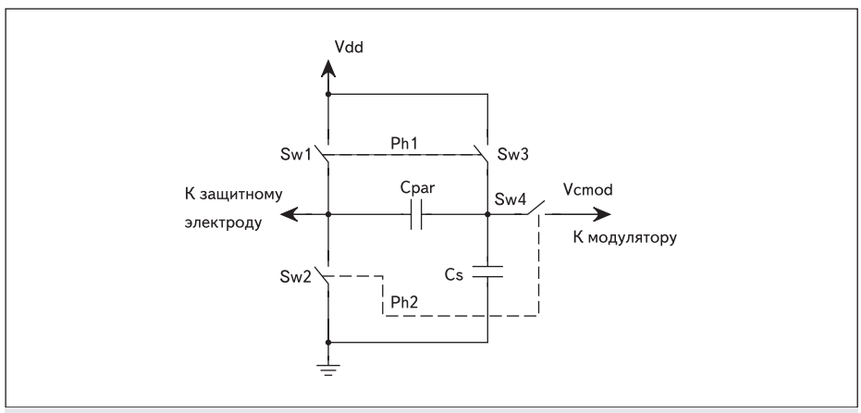


**Рис. 12.** Схема подключения защитного электрода

Из этого выражения видно, что ток модулятора уменьшается, если увеличивается *Cpar*. Это позволяет в случае необходимости разделить при программной обработке сигналы, вызываемые каплями воды и прикосновением пальца. Уменьшение опорного напряжения *Vref* позволяет уменьшить влияние паразитной емкости *Cpar* на ток модулятора. При настройке клавиатуры при практических тестах опорное напряжение может быть выбрано таким образом, что приращение результата измерений под действием капель воды будет близким нулю или даже немного отрицательным.

Модуль CSD имеет следующие параметры для настройки.

- **Finger Threshold.** Порог срабатывания при прикосновении. Этот параметр используется для определения состояния каждого сенсора. Для несгруппированных сенсоров (обычных кнопок, а не полос прокрутки) этот параметр является переменной, хранящейся в массиве baBtnFThreshold[], и может быть задан индивидуально для каждого сенсора при настройке. Возможные значения — от 5 до 255.
- **Noise Threshold.** Порог шума. Для отдельных сенсоров этот параметр определяет значение, свыше которого не будет обновляться уровень baseline. Для элементов полос прокрутки этот параметр устанавливает пороговое значение результата, которое будет использовано для вычисления позиции. Возможные значения — от 5 до 255.
- **Debounce.** Этот параметр добавляет счетчик, реализующий гистерезис активного состояния сенсора. Это необходимо для устранения возможных помех, связанных с электростатическими разрядами. Счетчик инкрементируется при вызове функций API, возвращающих состояние сенсора. Возможные значения от 1 до 255.
- **BaselineUpdate Threshold.** Если результат измерения оказывается свыше опорного значения, но разница между ними ниже порога шума, то при параметре Sensors Autoreset, установленном в Disabled, эта разница аккумулируется. Если накопившееся значение превышает значение параметра BaselineUpdate Threshold, то уровень baseline увеличивается на какое-то значение, а накопленное значение сбрасывается. Возможные значения — от 0 до 255. Большие значения означают медленное изменение уровня baseline.
- **NegativeNoiseThreshold.** Этот параметр задает порог отрицательной разницы результата и уровня baseline, свыше которого не будет происходить обновление baseline. Однако если результат оказывается ниже baseline в течение нескольких выборок, число которых определяется параметром LowBaselineReset, то уровень baseline будет сброшен.
- **LowBaselineReset.** Этот параметр работает совместно с NegativeNoiseThreshold и предназначен в первую очередь для того, чтобы отработать ситуацию, когда было прикосновение пальца в момент инициализации устройства и уровень baseline был вычислен неверно.
- **Sensors Autoreset.** Этот параметр определяет, будет ли уровень baseline обновляться постоянно (Enabled) или только когда текущая разница результата и baseline ниже Noise Threshold (Disabled). В первом случае максимальное времянахождение сенсора в активном состоянии ограничивается 5–10 секундами. Рекомендуется оставлять этот параметр в состоянии Enabled, за исключением случаев, когда требуется, чтобы сенсоры долго находились в активном состоянии. Рис. 13 иллюстрирует работу при разных установках этого параметра.
- **Hysteresis.** Значение этого параметра вычитается из Finger Threshold или добавляется к нему в зависимости от того, активен или неактивен сенсор в данный момент. Возможное значение — от 0 до 255, но оно должно быть меньше, чем значение параметра Finger Threshold. Правильный выбор этих параметров позволяет эффективно скомпенсировать влияние окружающей среды (температуры, влажности), а также сигналов помех (ESD, нестабильность питания).
- **Scanning Speed.** Этот параметр устанавливает скорость сканирования сенсоров как быструю (Fast), нормальную (Normal) и медленную (Slow). Медленная скорость улучшает соотношение «сигнал/шум», а также в меньшей степени нагружает процессор.
- **Resolution.** Данный параметр определяет разрядность результата в битах. Вариант схемы с тактированием от PRS16 допускает разрядность от 9 до 16 бит, остальные варианты — только 8, 10, и 12. Увеличение разрядности увеличивает чувствительность и улучшает «сигнал/шум». Высокая разрядность используется при реализации датчика приближения. Разрядность 16 бит, медленная скорость сканирования и 20-сантиметровая проволочная антенна позволяют определить присутствие руки на расстоянии свыше 20 см.
- **Modulator Capacitor Pin.** Параметр устанавливает контакт, к которому подключается конденсатор модулятора (Cmod).
- **Feedback Resistor Pin.** Этот параметр задает контакт для подключения резистора Rb.
- **Reference.** Параметр, определяющий источник опорного напряжения для компаратора в версии CSD для CY8C21x34.
- **Ref Value.** Параметр устанавливает уровень *Vref*, если используется источник, допускающий выбор уровня напряжения. Значение 0 соответствует минимуму, равному 1/4 *Vdd*. 8 — максимуму в 3/4 *Vdd*. С увеличением *Vref* уменьшается чувствительность, но увеличивается влияние защитного электрода. Если в проекте есть сенсоры с различной емкостью, то количество отсчетов в результате может быть сбалансировано установкой большего *Vref* для сенсоров с большей емкостью.
- **Prescaler Period.** Период делителя частоты в версиях с PRS8. Рекомендуемые значения равны $2^n$–1 для получения максимального соотношения «сигнал/шум».

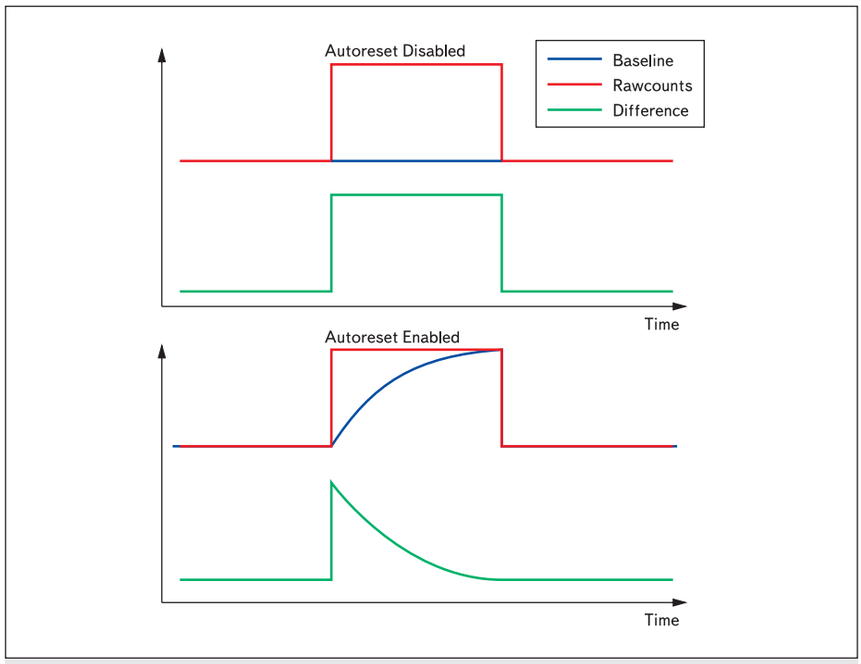


**Рис. 13.** Работа алгоритма CSD с отключенным автосбросом сенсора (вверху) и с включенным (внизу)

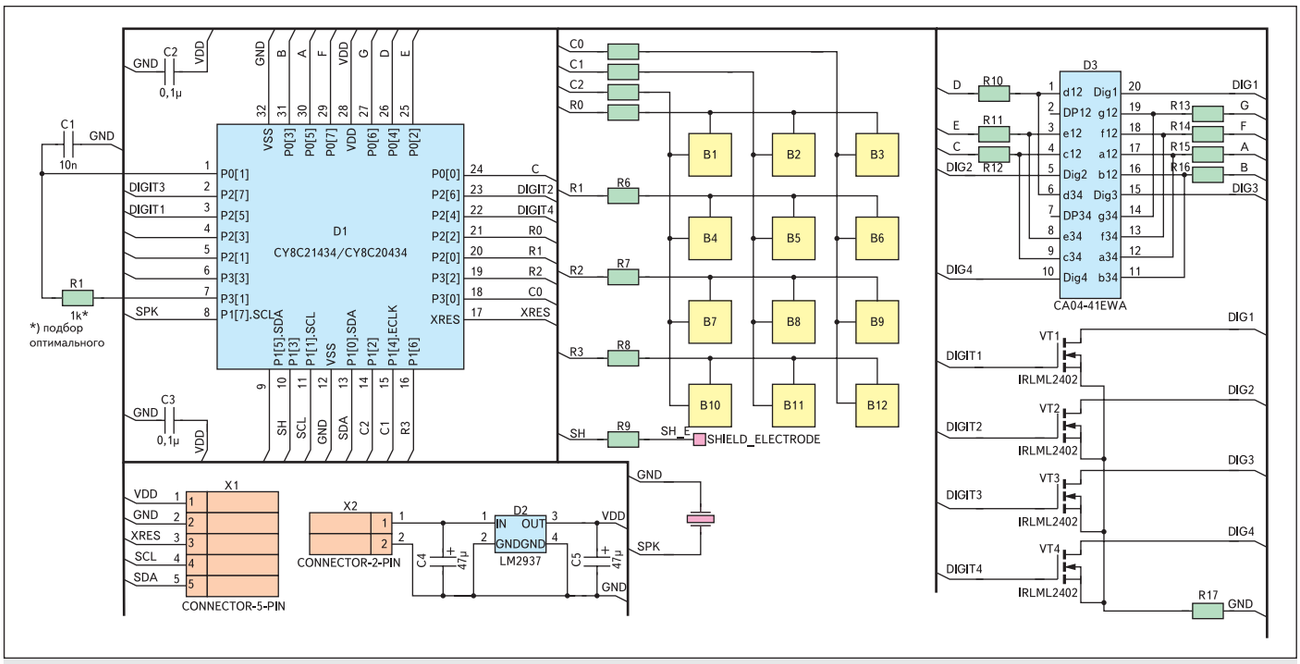


**Рис. 14.** Схема клавиатуры

- **PRS Polynomial.** Длина полинома PRS в версии PRS8. Короткая (Short) дает лучшее отношение «сигнал/шум», но устройство может быть более чувствительно к внешним помехам.
  Длинная (Long) ухудшает отношение «сигнал/шум», но устройство более устойчиво к внешним помехам.
- **ShieldElectrodeOut.** Задает выходную цифровую шину, которая будет использоваться для подключения защитного электрода.

Параметры настройки модулей CSR и CSA с похожими названиями имеют в общем-то такой же смысл.

В качестве примера использования технологии CapSense рассмотрим проект клавиатуры, которая может применяться в панелях систем контроля доступа, в домофонах, банкоматах и т. п. (рис. 14, 15). Клавиатура содержит 12 клавиш, которые, например, могут выступать в качестве цифровых (0–9), плюс клавиши ввода (вызова) и сброса. За основу был взят матричный способ реализации [15]. Введенные цифры отображаются на 4-символьном 7-сегментном светодиодном индикаторе (D3). Звуковой пьезоизлучатель (U1) используется для озвучивания нажатия клавиш, заменяя тем самым тактильный эффект.

Так как возможная область применения подразумевает использование клавиатуры вне помещений, то был выбран модуль CSD, имеющий встроенные средства защиты от водяных капель. Проходные резисторы R2 — R9 совместно с собственной емкостью контактов микросхемы образуют НЧ-фильтры для уменьшения влияния высокочастотных помех, например от мобильных телефонов.

**Рис. 15.** Внешний вид клавиатуры с использованием технологии CapSense

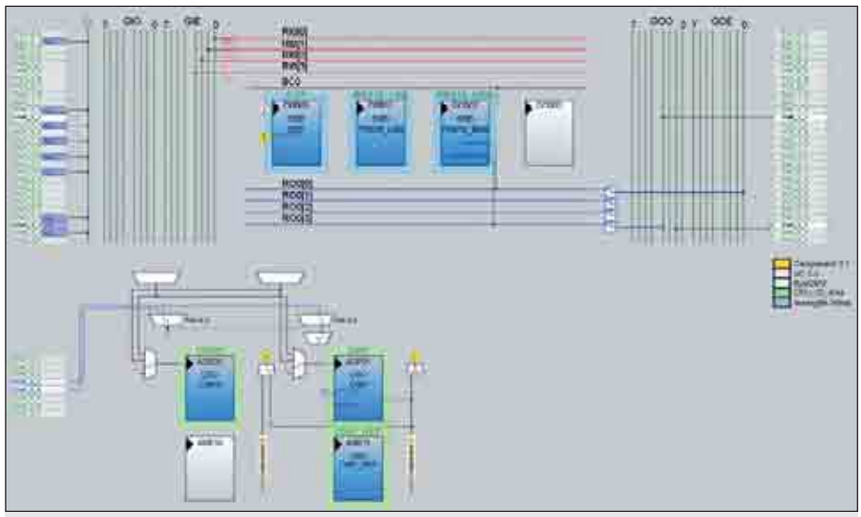

**Рис. 16.** Размещение модуля CSD в среде программирования PSoC Designer

Номинал этих резисторов выбирается в диапазоне от 100 Ом до 1 кОм, однако чем больше номинал резистора, тем меньше должна быть скорость сканирования сенсоров.

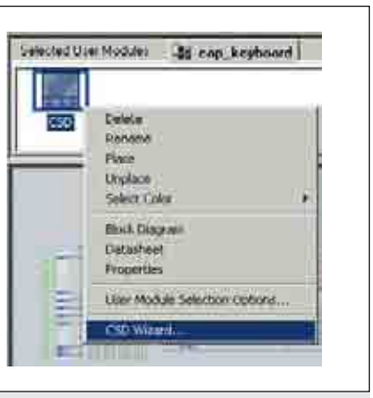


**Рис. 17.** Вызов мастера настройки модуля CSD

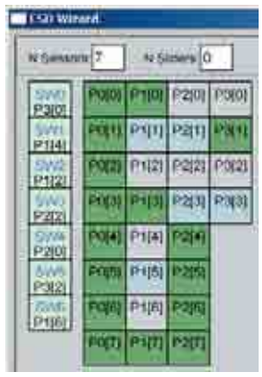


**Рис. 18.** Мастер настройки модуля CSD

**Рис. 19.** Клавиатура с установленным тестовым покрытием

Программирование PSoC и настройка клавиатуры начинается с выбора модуля CSD в САПР PSoC Designer и установки его на массиве аналоговых и цифровых блоков (рис. 16).

Тип схемы при первых экспериментах лучше выбрать с использованием PRS16. Следующим шагом является вызов мастера настройки CSD Wizard (рис. 17).

Подобный мастер существует и для модулей CSR и CSA. При запуске мастер предлагает задать количество обрабатываемых сенсоров. В нашем примере матрица клавиатуры реализована как 4×3, то есть для ее реализации необходимо 7 сенсоров. Мастер настройки позволяет назначить в качестве входов от сенсоров необходимые контакты микросхемы, выбрав их из списка (рис. 18). Зеленым цветом обозначаются контакты, уже имеющие имена и заблокированные для выбора. Синим — контакты, которые можно выбрать. Серым цветом в списке отображаются уже выбранные контакты. Назначение контакта осуществляется перетаскиванием соответствующего ему поля из списка справа (имена Px[y]) на поле с необходимым номером сенсора в левом столбце (SW0–SW6).

Результатом работы мастера являются внутренние настроечные таблицы, используемые при работе модуля.

Следующим шагом задаются параметры модуля. В описании модуля рекомендуется начинать настройку с задания следующих значений параметров:

```
FingerThreshold = 40
NoiseThreshold = 20
BaselineUpdateThreshold = 200
Sensor Autoreset = Enabled
Hysteresis = 10
Debounse = 3
NegativeNoiseThreshold = 20
LowBaselineReset = 50
Scannning Speed = Fast
Resolution = 9
Ref Value = 2
```

После установки параметров следует сгенерировать шаблон прошивки PSoC, добавить в него необходимые функции по обработке и скомпилировать. Подробно о создании проекта было рассказано в предыдущих статьях цикла [1].

Перед настройкой клавиатуры ее необходимо подготовить — приклеить покрытие с помощью клея или адгезивной пленки, устранив любой воздушный зазор, так как он сильно ухудшает чувствительность (рис. 19).

Кроме задания параметров необходимо установить внешние компоненты, которые использует модуль CSD — конденсатор модулятора *Cmod* и резистор *Rb*. Конденсатор модулятора рекомендуется выбирать в диапазоне от 4,7 до 47 нФ. Оптимальное значение выбирается в ходе экспериментов для получения оптимального соотношения «сигнал/шум». Как правило, в большинстве случаев керамический конденсатор емкостью 10 нФ дает хорошие результаты. Сопротивление резистора *Rb* зависит от емкости сенсора *Cs* и выбирается при настройке клавиатуры. Обычно величина резистора лежит в диапазоне 500 Ом–10 кОм.

Для мониторинга результатов сканирования удобно использовать утилиту (рис. 20) из документа AN2397 [14], который доступен на сайте Cypress. Подключение настраиваемой клавиатуры к компьютеру проще всего выполнить через интерфейс RS-232, используя для преобразования уровней сигналов хорошо известную несложную схему на основе MAX232 или аналогах. В документе AN2397 приведен в качестве примера несложный проект, интерфейс UART в котором реализован программно. Эти готовые функции удобно использовать для передачи результатов в компьютер при отладке.

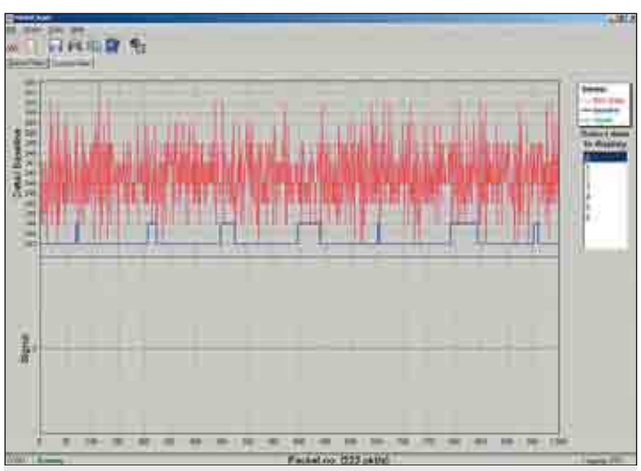

**Рис. 20.** Программа для отладки модуля CSD

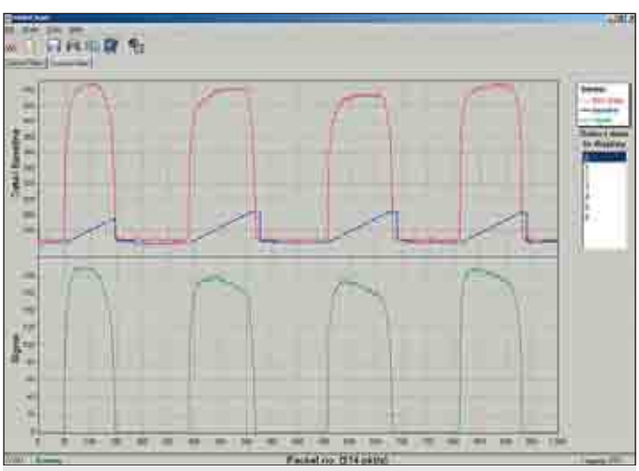

**Рис. 21.** Сигнал при периодичном касании сенсора без покрытия

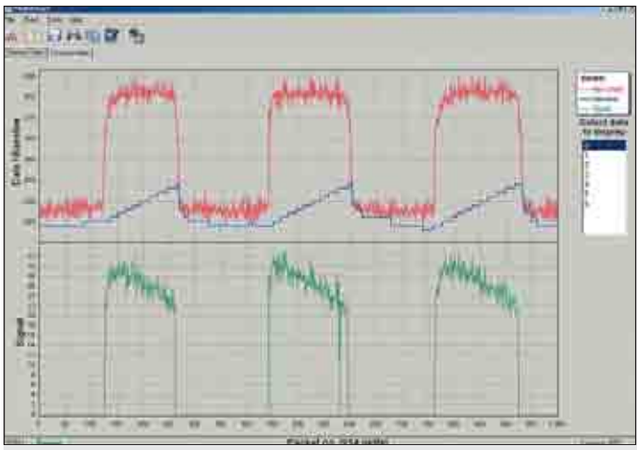

**Рис. 22.** Сигнал с сенсора при установленном покрытии

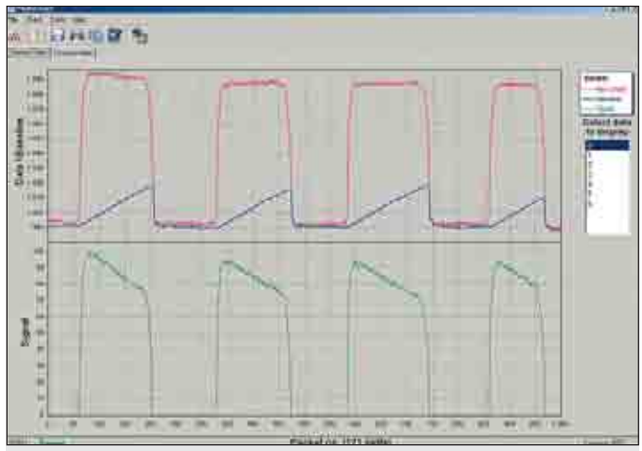

**Рис. 23.** Показания при увеличенной разрядности

Рис. 20 демонстрирует сигнал с сенсора при отсутствии прикосновения. Наблюдая за результатами, необходимо подобрать значение *Rb* так, чтобы при касании сенсора результат (Raw Data) составлял 60–70% от максимально возможного при выбранной разрядности. На следующем этапе настройки с помощью небольшого проводящего предмета имитируют легкое прикосновение пальца. В том случае, если результат будет иметь соотношение «сигнал/шум» хуже, чем минимально рекомендуемое 5:1 [11], то следует увеличить разрядность модуля CSD, изменив параметр Resolution. Однако увеличенная разрядность приведет к снижению скорости сканирования сенсоров, поэтому, если скорость работы важна для конкретного применения, можно применить схему CSD с PRS8. В свою очередь, как уже упоминалось выше, применение PRS8 увеличит чувствительность к внешним помехам.

В ходе экспериментов по настройке рассматриваемой клавиатуры был выбран *Rb*, равный 2,7 кОм, что дает результат примерно на уровне 80–85% от максимального в отсутствие покрытия на клавиатуре (рис. 15, 21) и всего 50% с установленным покрытием (рис. 19, 22).

В первом случае размах сигнала оказывается достаточным для надежного определения срабатывания сенсора, во втором же случае показания лежат ниже FingerThreshold+Hysteresis, поэтому сенсор считается неактивным. Увеличить чувствительность можно, подняв разрядность, например, до 11 бит — результат показан на рис. 23.

Выбор оптимальной частоты сканирования сенсоров позволяет снизить влияние возможной нестабильности напряжения питания. Таким образом, как видно из сказанного выше, процесс настройки весьма итеративный. Безошибочной работы клавиатуры добиваются подбором разрядности, значений порогов и гистерезиса, скорости сканирования. Кроме того, надежная работа может быть обеспечена и дополнительной программной обработкой. Например, для случая с матричной клавиатурой, для определения нажатой клавиши необходимо одновременное срабатывание двух сенсоров. В листинге приведен главный алгоритм программы, демонстрирующий необходимые действия для определения номера сработавшей клавиши. Полностью проект можно найти на сайте **w ww.macrogoup. ru** в разделе, посвященном PSoC.

```
#include <m8c.h>        // part specific constants and macros
#include «PSoCAPI.h»    // PSoC API definitions for all User Modules
#include «led.h»
#include «speaker.h»
#include «tx8.h»
#include «sleep.h»

BYTE DeviceStatus;
#define btn_error 0x04

void main()
{
   BYTE btn=0; //код нажатой клавиши

   DeviceStatus = 0;
   LED_Start();

   INT_MSK0 = INT_MSK0_SLEEP;
   M8C_EnableGInt;

//запуск и инициализация модуля CSD
   CSD_Start();
   CSD_InitializeBaselines();
   CSD_SetDefaultFingerThresholds();

   TX8_Start();

while (1)
{

//сканирование сенсоров
CSD_ScanAllSensors();
CSD_UpdateAllBaselines();

//-------------------------------------
//вывод тестовой информации
   TX8_PutCRLF();
   TX8_Write((char *) (CSD_waSnsResult), CSD_TotalSensorCount*2);
   TX8_Write((char *) (CSD_waSnsBaseline), CSD_TotalSensorCount*2);
   TX8_Write((char *) (CSD_waSnsDiff), CSD_TotalSensorCount*2);

   TX8_PutChar(0);
   TX8_PutChar((CHAR)0xFF);
   TX8_PutChar((CHAR)0xFF);
//-------------------------------------

//проверяем какие сенсоры активны и определяем нажатую клавишу
   if(CSD_bIsAnySensorActive())
   {
      DeviceStatus &= ~btn_error;
      if(CSD_bIsSensorActive(0))
      {
         if(CSD_bIsSensorActive(3))
         {
            btn=0;
            Speaker_Freq = SPK_3000_Hz;
         }
         else if(CSD_bIsSensorActive(4))
         {
            btn=1;
            Speaker_Freq = SPK_1500_Hz;
         }
         else if(CSD_bIsSensorActive(5))
         {
            btn=2;
            Speaker_Freq = SPK_1000_Hz;
         }
         else if(CSD_bIsSensorActive(6))
         {
            btn=3;
            Speaker_Freq = SPK_750_Hz;
         }
         else DeviceStatus |= btn_error;
      }
      else if(CSD_bIsSensorActive(1))
      {
         if(CSD_bIsSensorActive(3))
         {
            btn=4;
            Speaker_Freq = SPK_600_Hz;
         }
         else if(CSD_bIsSensorActive(4))
         {
            btn=5;
            Speaker_Freq = SPK_429_Hz;
         }
         else if(CSD_bIsSensorActive(5))
         {
            btn=6;
            Speaker_Freq = SPK_333_Hz;
         }
         else if(CSD_bIsSensorActive(6))
         {
            btn=7;
            Speaker_Freq = SPK_273_Hz;
         }
         else DeviceStatus |= btn_error;
      }
      else if(CSD_bIsSensorActive(2))
      {
         if(CSD_bIsSensorActive(3))
         {
            btn=8;
            Speaker_Freq = SPK_231_Hz;
         }
         else if(CSD_bIsSensorActive(4))
         {
            btn=9;
            Speaker_Freq = SPK_200_Hz;
         }
         else if(CSD_bIsSensorActive(5))
         {
            LED_Str[2]=0x4B;//'C'
            LED_Str[1]=0x12;//'A'
            LED_Str[0]=0x13;//'P'
            btn=5;//'S'
            Speaker_Freq = SPK_150_Hz;
         }
         else if(CSD_bIsSensorActive(6))
         {
            btn=16;
```

```
                LED_Str[2]=0xFF;
                LED_Str[1]=0xFF;
                LED_Str[0]=0xFF;
                Speaker_Freq = SPK_100_Hz;
            }
            else DeviceStatus |= btn_error;
        }
        else DeviceStatus |= btn_error;

        if(!(DeviceStatus & btn_error))
        {
//если ошибок нет — выводим цифру на LED-индикатор и озвучиваем нажатие
            LED_ShiftStr(LED_Table[btn]);

//озвучка на время тестирования отключена
//          Speaker_Sound(1500);
        }

        //ждем пока не отпусятят клавишу
        do
        {
            CSD_ScanAllSensors();
            CSD_UpdateAllBaselines();

            //-----------------------------------
            //вывод тестовой информации
            TX8_PutCRLF();
            TX8_Write((char *) (CSD_waSnsResult), CSD_TotalSensorCount*2);
            TX8_Write((char*)(CSD_waSnsBaseline), CSD_TotalSensorCount*2);
            TX8_Write((char *) (CSD_waSnsDiff), CSD_TotalSensorCount*2);

            TX8_PutChar(0);
            TX8_PutChar((CHAR)0xFF);
            TX8_PutChar((CHAR)0xFF);
            //-----------------------------------
        }
        while(CSD_bIsAnySensorActive());
    }
  }
}
```

**Листинг 1**

На основе данной технологии «Макро Групп» и «Абрис-KEY» запустили совместный проект: бесконтактные клавиатуры на основе технологии CapSense на микросхемах PSoC Cypress. Отличительные особенности данных клавиатур:

- В отличие от традиционных кнопочных переключателей и клавиатур, данные клавиатуры более технологичны: кнопками и слайдерами клавиатуры являются емкостные сенсоры, выполненные в виде металлизации нужной геометрии и размеров по стандартной технологии изготовления контактных площадок на печатных платах.
- Клавиатуры могут быть различных размеров и с разным количеством кнопок, что позволяет интегрировать их в любые устройства. По желанию заказчика клавиатура может быть совмещена с основной печатной платой прибора, с одной стороны платы будут установлены электронные компоненты, а на другой стороне выполнена клавиатура.
- С помощью той же микросхемы PSoC Cypress, на которой реализована клавиатура, можно также осуществлять вывод информации, например, на жидкокристаллический или OLED-индикатор.
- Благодаря универсальности PSoC Cypress клавиатура может быть подключена через интерфейсы $I^2C$, UART или USB.
- Тактильный эффект может быть реализован и подобран по желанию заказчика за счет использования микродинамика.
- Благодаря технологии CapSense, обеспечивающей высокую чувствительность сенсоров, в качестве защитного покрытия клавиатуры может использоваться стекло или полимерный материал толщиной до 15 мм.
- Высокая надежность и ресурс работы.

## Литература


1. Килочек Д. Проектирование на программируемых системах на кристалле PSoC Cypress // Компоненты и технологии. 2006. № 4, 6, 8, 12; 2007. № 3.
2. AN2233a. Capacitive Switch Scan.
3. AN2355. Calibrating CapSense with the CSR User Module .
4. AN2277. Capacitive Front Panel Display Demonstration.
5. AN14459. CapSense Device and Method Selection Guide.
6. AN2393. Migrating from CSR to CSA.
7. AN2408. Migrating from CSR to CSD.
8. AN2394. CapSense Best Practices.
9. AN2292. Layout Guidelines for PSoC™ CapSense™.
10. AN2318. EMC Design Considerations for PSoC CapSense Applications.
11. AN2403. Signal-to-Noise Ratio Requirement for CapSense Applications.
12. AN2360. Power Consumption and Sleep Considerations in Capacitive Sensing Applications.
13. AN2398. Waterproof Capacitance Sensing.
14. AN2397. CapSense Data Viewing Tool.
15. AN2407. PC-Compatible USB CapSense Matrix Keyboard.
16. PSoC Mixed Signal Array Technical Reference Manual (TRM).
17. CapSense Relaxation Oscillator Data Sheet.
18. CapSense Sigma-Delta Data Sheet.
19. CapSense Successive Approximation Data Sheet.