C8051F700 Capacitive Touch Sense Overview

In this section, we are going to cover the Silicon Labs Capacitive Touch Sense solution.

Agenda

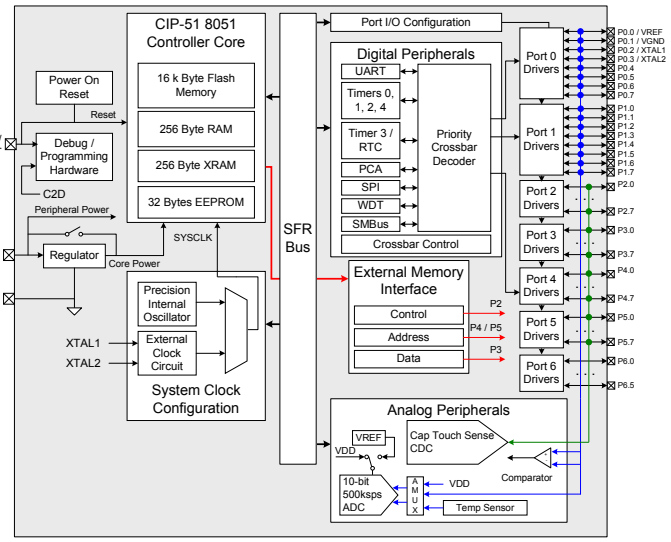
- C8051F700 block diagram
- Capacitive sensing overview
- C8051F700 capacitive sensing module features
- Summary
- Where to learn more

2

We are going to look at the new C8051F700 devices in this module. We will first take a look at the high level block diagram and then dive into some of the new features of these devices that make them excellent choices for a wide range of capacitive sensing applications.

Introducing The Cost Optimized C8051F700

- New patented capacitive touch sense
  - True capacitance-to-digital converter
  - Robust and responsive
  - Easy to use
- High performance MCU
  - 25 MHz 8051 CPU
  - Best in class ADC
  - 16kB flash
  - 32B data-EEPROM
- 54 multi-function GPIO
  - User configured as digital or analog
  - Digital crossbar assigns pins
  - Up to 32 capacitive touch sense inputs
  - Available in TQFP64, TQFP48, and QFN48 (7x7 mm) packages



SILICON LABS

3

Let's have a look at the features of the new C8051F700 microcontroller family. Here are a few highlights. First the Silicon labs patented Capacitance to Digital converter enables accurate, responsive and very reliable capacitive touch sense capability. Next the classic Silicon Labs performance you expect. A fast 25 MIPS CPU, best in class analog functions, such as the accurate 10 bit analog to digital converter with internal voltage reference. 2% calibrated internal precision oscillator. Up to 16Kbytes of in system programmable flash and 32 bytes of EEPROM with 100,000 cycle endurance guaranteed.

As usual, you don't have to choose between I2C, SPI or UART as all are included as independent functions. There is also the timers and PCA as well as the external memory interface. Finally, there are 54 general purpose I/O pins. You can configure most inputs to be analog inputs to the ADC, comparator or capacitance to digital converter via software and digital functions can be enabled through the crossbar.

C8051F700 Product Family Selector Guide

Part Number	MIPS (peak)	FLASH Memory (bytes)	Data EEPROM (bytes)	RAM (bytes)	Digital Port I/O Pins	Serial Buses	Timers (16-bit)	PCA Chnls	Internal Osc	Cap Touch Sense	ADC0	Temp Sensor	VREF	Comp.	Package
C8051F700-GQ	25	15kB	32	512	54	UART, I2C, SPI	4	3	2%	Y	10-Bit	Y	Y	1	QFP64
C8051F701-GQ	25	15kB	32	512	54	UART, I2C, SPI	4	3	2%	Y	-			1	QFP64
C8051F702-GQ	25	16kB	-	512	54	UART, I2C, SPI	4	3	2%	Y	10-Bit	Y	Y	1	QFP64
C8051F703-GQ	25	16kB	-	512	54	UART, I2C, SPI	4	3	2%	Y	-			1	QFP64
C8051F704-GQ	25	15kB	32	512	39	UART, I2C, SPI	4	3	2%	Y	10-Bit	Y	Y	1	QFP48
C8051F704-GM	25	15kB	32	512	39	UART, I2C, SPI	4	3	2%	Y	10-Bit	Y	Y	1	QFN48
C8051F705-GQ	25	15kB	32	512	39	UART, I2C, SPI	4	3	2%	Y	-			1	QFP48
C8051F705-GM	25	15kB	32	512	39	UART, I2C, SPI	4	3	2%	Y	-			1	QFN48
C8051F706-GQ	25	16kB	-	512	39	UART, I2C, SPI	4	3	2%	Y	10-Bit	Y	Y	1	QFP48
C8051F706-GM	25	16kB	-	512	39	UART, I2C, SPI	4	3	2%	Y	10-Bit	Y	Y	1	QFN48
C8051F707-GQ	25	16kB	-	512	39	UART, I2C, SPI	4	3	2%	Y	-			1	QFP48
C8051F707-GM	25	16kB	-	512	39	UART, I2C, SPI	4	3	2%	Y	-			1	QFN48
C8051F708-GQ	25	8kB	32	512	54	UART, I2C, SPI	4	3	2%	Y	10-Bit	Y	Y	1	QFP64
C8051F709-GQ	25	8kB	32	512	54	UART, I2C, SPI	4	3	2%	Y	-			1	QFP64
C8051F710-GQ	25	8kB	-	512	54	UART, I2C, SPI	4	3	2%	Y	10-Bit	Y	Y	1	QFP64
C8051F711-GQ	25	8kB	-	512	54	UART, I2C, SPI	4	3	2%	Y	-			1	QFP64
C8051F712-GQ	25	8kB	32	512	39	UART, I2C, SPI	4	3	2%	Y	10-Bit	Y	Y	1	QFP48
C8051F712-GM	25	8kB	32	512	39	UART, I2C, SPI	4	3	2%	Y	10-Bit	Y	Y	1	QFN48
C8051F713-GQ	25	8kB	32	512	39	UART, I2C, SPI	4	3	2%	Y	-			1	QFP48
C8051F713-GM	25	8kB	32	512	39	UART, I2C, SPI	4	3	2%	Y	-			1	QFN48
C8051F714-GQ	25	8kB	-	512	39	UART, I2C, SPI	4	3	2%	Y	10-Bit	Y	Y	1	QFP48
C8051F714-GM	25	8kB	-	512	39	UART, I2C, SPI	4	3	2%	Y	10-Bit	Y	Y	1	QFN48
C8051F715-GQ	25	8kB	-	512	39	UART, I2C, SPI	4	3	2%	Y	-			1	QFP48
C8051F715-GM	25	8kB	-	512	39	UART, I2C, SPI	4	3	2%	Y	-			1	QFN48

- 24 Unique part numbers!
  - Choice flash size
  - Can select EEPROM (in larger flash size, EEPROM is traded for 1 kB flash)
  - ADC or no-ADC
  - Capacitive touch sense option

SILICON LABS

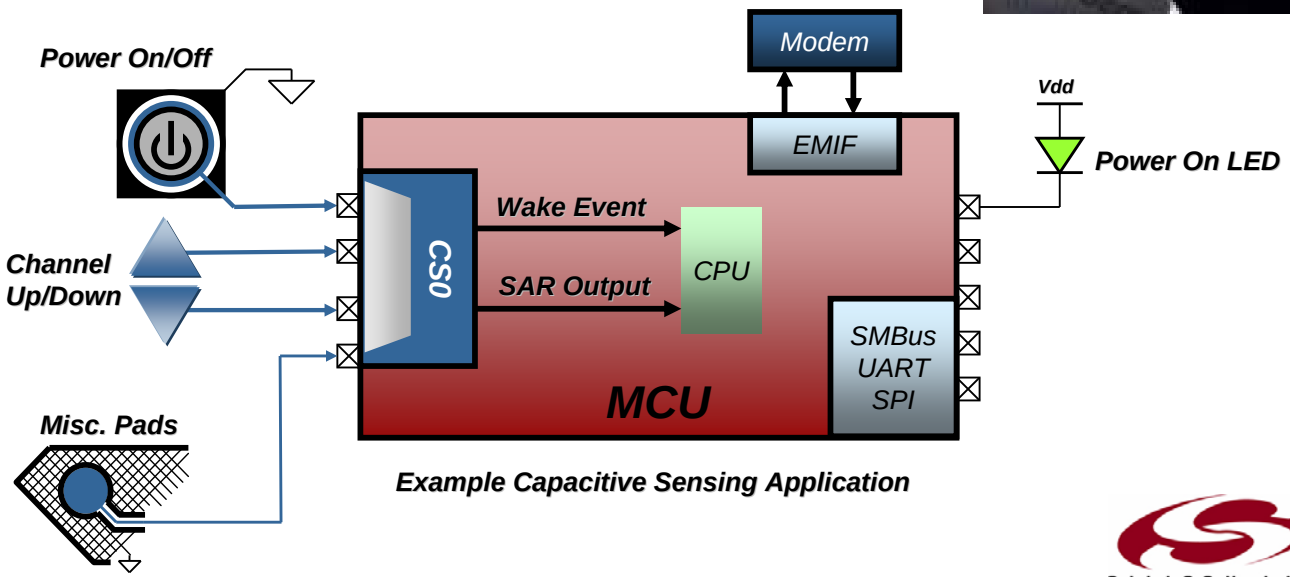
4

The C8051F700 family gives you choices in memory size, analog function, and EEPROM. In addition, this is offered in three different package options.


SILICON LABS
Capacitive Sensing

Why Use Capacitive Sensing?

- Capacitive sensing provides switch input using standard PCB traces
  - Reduces mechanical failures
  - Low-cost because the switch is as simple as a trace on a PCB



Example Capacitive Sensing Application

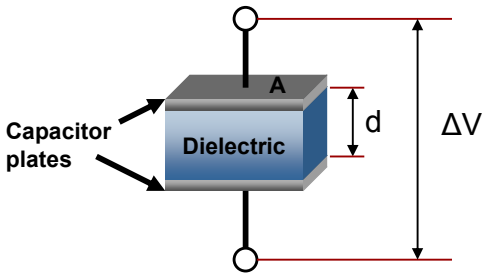
SILICON LABS

6

Here is an example where the capacitive sensing for touch applications could provide a cost reduction and increased reliability by eliminating the mechanical switches commonly found in these applications. Capacitive sensing can provide the user interface for a wide range of applications ranging from consumer goods to industrial applications.

What Are We Measuring?

- Capacitance
  - If two metal plates are placed with a gap between them and a voltage is applied to one of the plates, an electric field will exist between the plates
  - Any two conductive objects can be used for the plates of the capacitor
  - When using capacitive sensing for touch applications the capacitor is typically formed by a PCB element and the conductive object in proximity
- Capacitor parameters
  - Area – capacitance is directly proportional to the area of the plates
  - Dielectric – capacitance is directly proportional to the relative permittivity of the material between the plates (air, glass, plastic, etc.)
  - Distance – capacitance is indirectly proportional to the distance between the two plates
- The capacitance of a parallel-plate capacitor is given by:



$$C = \epsilon_0 \epsilon_r * (A / d)$$

where:
ϵ_0 is the permittivity of free space
ϵ_r is the relative permittivity of the dielectric material
A is the area of the plates
D is the distance between the plates

SILICON LABS

7

When using capacitive sensing for touch applications the PCB trace typically acts as one plate of the capacitor. When a conductive object (such as a finger for touch sensing) comes into proximity it acts as the other plate and air is the dielectric. As the object moves closer to the pad the capacitance changes based on the equation for capacitance shown. The constants in the equation represent the value for the dielectric constant of the material, i.e., air, glass etc. The value for A is a measure of the area of the pad used for the sensor capacitor and d represents the distance between the two plates. So you can see, for touch applications, as the finger moves closer to a pad the value for d is reduced which then increases the capacitance.

ϵ_r is the relative static permittivity (sometimes called the dielectric constant) of the material between the plates

A is the area of each plate

D is the separation between the plates.

Putting Your Finger on the Switch

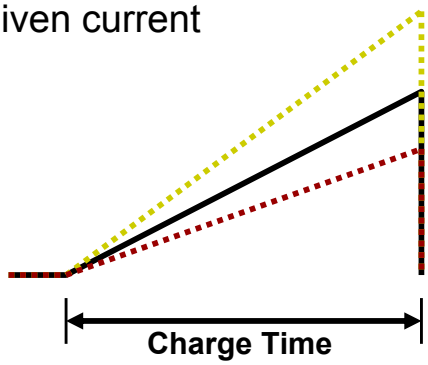
- Capacitance increases by touching the switch directly with a finger
- The change in voltage of a parallel-plate sensor capacitor is derived from the current of the capacitor given by:

$$I_{SENSOR} = C_{SENSOR} \frac{dV}{dt}$$

or

$$dV = \frac{I_{SENSOR}}{C_{SENSOR}} dt$$

- This additional capacitance reduces the voltage potential of the sensor capacitor at a given current



Idle Switch

Active Switch

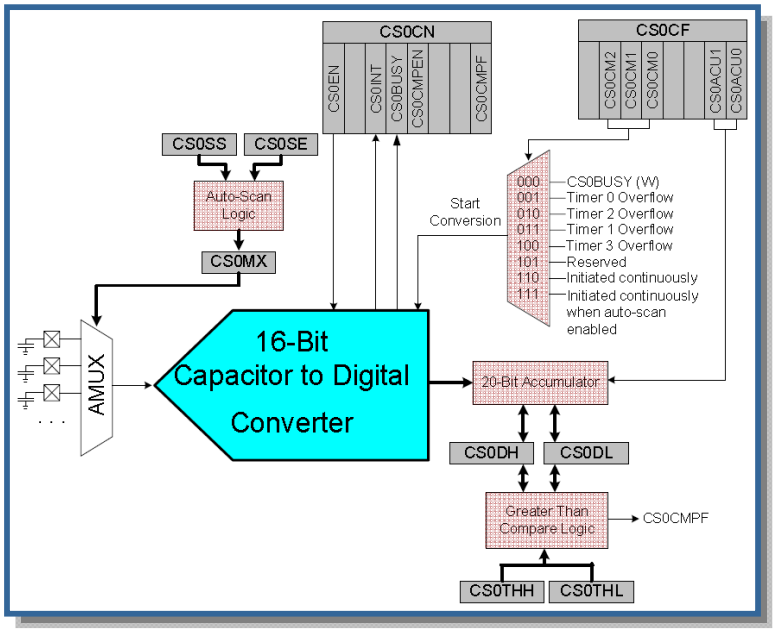
SILICON LABS

8

Remember that as the finger moves closer to the pad the capacitance changes based on the capacitance equation from the previous slide. Keeping in mind that as a finger moves closer to the pad the value for 'd' is reduced which then increases the capacitance. From the equation for the change in voltage above we note that as the capacitance increases the voltage potential for a given current decreases. This voltage potential change measured by the new capacitance to digital converter module of the C8051F700.

C8051F700 Capacitive Sensing Implementation

- *Capacitor charge timing using successive approximation (SAR)*
  - Varies the current through the sensor capacitor to match a reference ramp
- 16 bit SAR converter
- 32 channel input multiplexor
- Single channel or multi-channel scanning using auto scan
- Hardware accumulator
- Window logic threshold to trigger an event when an active state is detected



Let's take a look at how to implement the Capacitive Sensing using the C8051F700 family from Silicon Labs. The 16 bit SAR block itself is implemented as a complete stand alone block and generates it's own time base. This allows it to run autonomously from the CPU and provides the capability to wake the CPU from a low power state. Also integrated is a 20 bit accumulator that can add 1, 4, 8 or 16 scans and then provide the division using a simple shift function. Using this hardware accumulator and simple shift function provides a low overhead averaging function to reduce the affects of noise, thus increasing performance of the system. The window comparator is software programmable and is set based on the system level performance of the pad configurations. After the idle and signal levels are determined and a suitable threshold is obtained the window comparator can be set to trigger an event when the active switch threshold is met. This is useful to reduce CPU overhead and also for low power modes. The conversion of the CS0 module is capable of being generated from several sources including software trigger, all of the timers and auto-scanning. All of the required components for Capacitive sensing are integrated on chip. Therefore, we can connect the Capacitive Sense pad directly to the pin of the MCU.

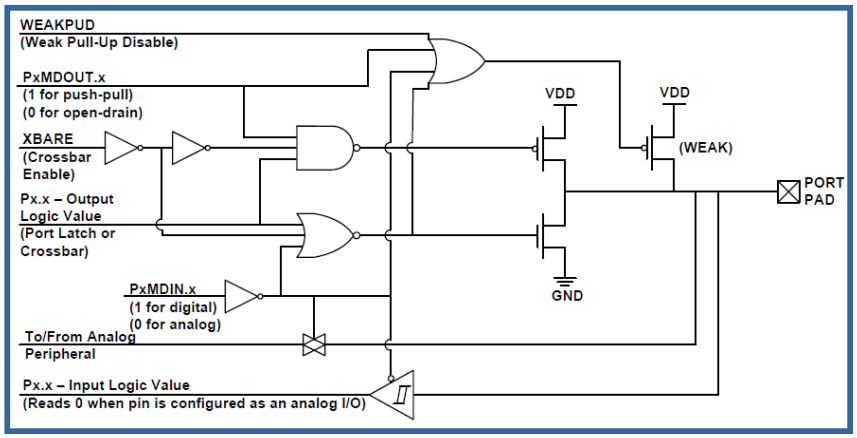
Auto-Scanning Programming

- ***Step 1***: Set port pins to analog inputs
- ***Step 2***: Enable auto-scan to convert from write to CS0BUSY by setting the CS0 start-of- conversion bits (CS0CF6:4) to 111 b
- ***Step 3***: Enable the CS0 module and the comparator
- ***Step 4***: Set the greater than threshold using CS0TH
- ***Step 5***: After enabling auto-scan, the starting end ending channels should be set to appropriate values in CS0SS and CS0SE
  - Writing to CS0SS when CS0CF[6:4] = 111 b copies the value to CS0MX
- ***Step 6***: Write to CS0BUSY to start conversions: CS0CN |= 0x10

Analog Input Pins

- Most port pins can be used for digital I/O
- Port pin configured as analog using PxMDIN register bits set to a ‘0’
- The output mode is selected to be open drain using the PxMDOUT bit set to a ‘0’
- The port latch bit should be set to a ‘1’

PxMDIN	PxMDOUT	Px	Description
0	0	1	Analog Input

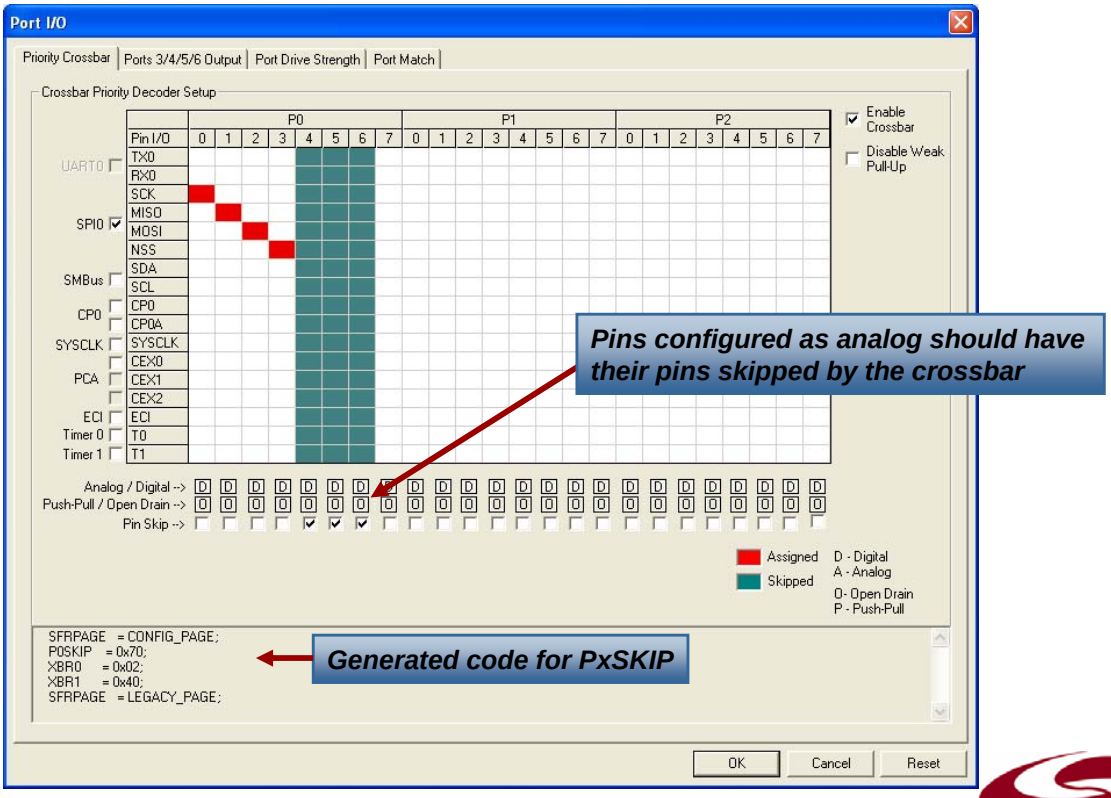


SILICON LABS

11

To configure the inputs as analog inputs the PxMDIN and PxMDOUT bits should be set to a ‘0’ and the port latch bits (Px) should be set to a ‘1’. These settings disable the output drivers as well as the weak pull-up and the digital input buffer.

Crossbar Pin Assignment Using Config Wizard



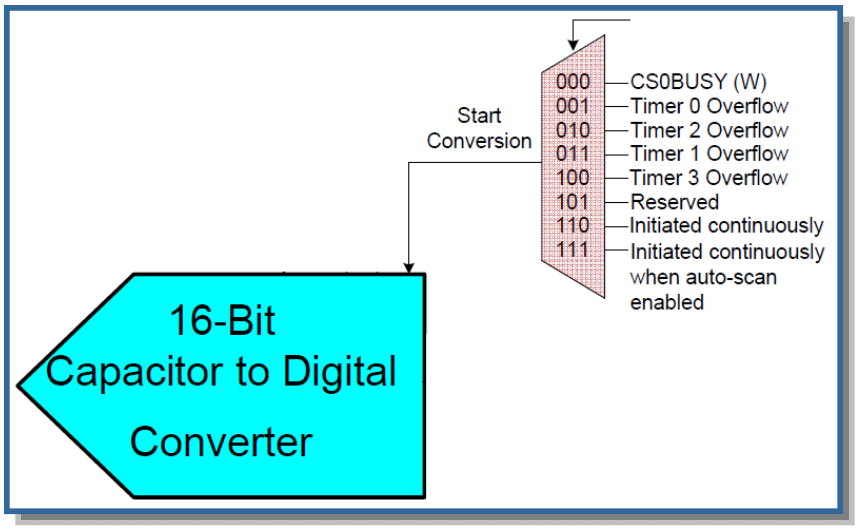
This diagram is taken from the C8051F700 family for reference. The crossbar works on a predefined priority order to allocate the digital peripheral I/O. When one of the analog functions are used for a pin we should set the PxSKIP bit for the pin so the crossbar does not allocate a peripheral input or output signal to the same pin.

Auto-Scanning Programming

- ***Step 1***: Set port pins to analog inputs
- ***Step 2***: Enable auto-scan to convert from write to CS0BUSY by setting the CS0 start-of- conversion bits (CS0CF6:4) to 111 b
- ***Step 3***: Enable the CS0 module and the comparator
- ***Step 4***: Set the greater than threshold using CS0TH
- ***Step 5***: After enabling auto-scan, the starting end ending channels should be set to appropriate values in CS0SS and CS0SE
  - Writing to CS0SS when CS0CF[6:4] = 111 b copies the value to CS0MX
- ***Step 6***: Write to CS0BUSY to start conversions: CS0CN |= 0x10

Start of Conversion

- Conversions initiated in one of seven ways
  - Determined by programmed state of the CS0 start of conversion bits (CS0CF6:4)
- Can be configured to be initiated continuously through one of two methods
  - CS0 can be configured to convert at a single channel continuously
  - CS0 can be configured to convert continuously with auto-scan enabled.
    - When configured to convert continuously, conversions will begin after the CS0BUSY bit in CS0CF has been set



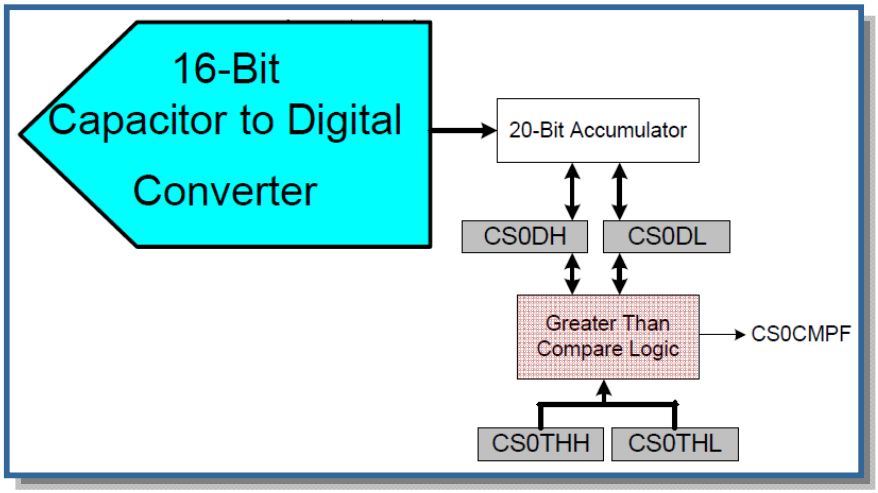
The capacitive sensing module can be triggered from a wide number of sources. Software can initiate a conversion on demand by setting the CS0BUSY bit. Hardware timers can also initiate a conversion using the timer overflow output for periodic measurements. When the accumulator is enabled continuous samples can be acquired and averaged in the hardware. Adding to that the auto-scan logic, we can continuously convert multiple channels. So as you can see there quite a few options for starting keypad scans.

Auto-Scanning Programming

- ***Step 1***: Set port pins to analog inputs
- ***Step 2***: Enable auto-scan to convert from write to CS0BUSY by setting the CS0 start-of- conversion bits (CS0CF6:4) to 111 b
- ***Step 3***: Enable the CS0 module and the comparator
- ***Step 4***: Set the greater than threshold using CS0TH
- ***Step 5***: After enabling auto-scan, the starting end ending channels should be set to appropriate values in CS0SS and CS0SE
  - Writing to CS0SS when CS0CF[6:4] = 111 b copies the value to CS0MX
- ***Step 6***: Write to CS0BUSY to start conversions: CS0CN |= 0x10

Comparator

- Compares the capacitor to digital converter output to the programmed value
  - Sets the output flag high when the sample is greater than the programmed value
    - For auto-accumulate the comparator will only update the compare flag after all samples have been accumulated via hardware
  - Can generate an interrupt
  - Can wake the CPU when the comparison is TRUE
    - Does not require interrupts to be enabled



16

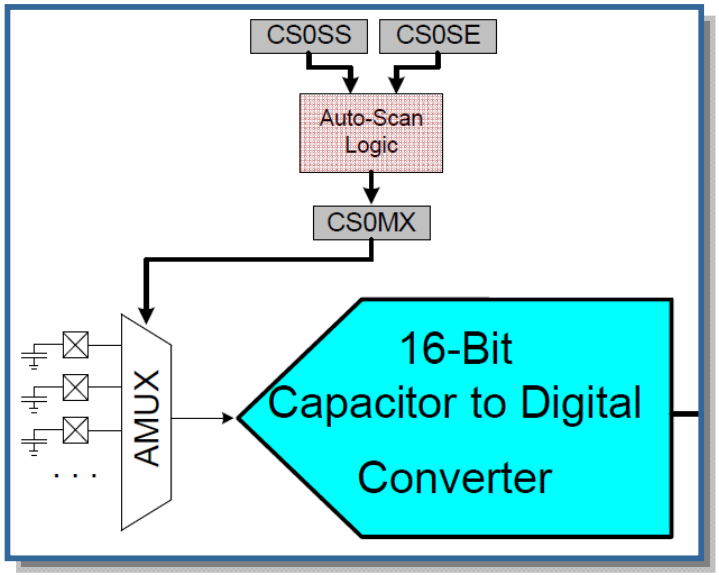
The capacitive sensing comparator compares either the single conversion or the averaged conversions (when the hardware accumulator is enabled) and generates an interrupt when the number of accumulated channels has exceeded the programmed limit. When the interrupt is generated the auto-scan logic is halted so that the CPU can read the mux value to determine which channel has triggered the event. Since the module is self timed and can automatically scan the channels the comparator enables the capacitive sensing module to wake the CPU from the sleep modes.

Auto-Scanning Programming

- ***Step 1***: Set port pins to analog inputs
- ***Step 2***: Enable auto-scan to convert from write to CS0BUSY by setting the CS0 start-of- conversion bits (CS0CF6:4) to 111 b
- ***Step 3***: Enable the CS0 module and the comparator
- ***Step 4***: Set the greater than threshold using CS0TH
- ***Step 5***: After enabling auto-scan, the starting end ending channels should be set to appropriate values in CS0SS and CS0SE
  - Writing to CS0SS when CS0CF[6:4] = 111 b copies the value to CS0MX
- ***Step 6***: Write to CS0BUSY to start conversions: CS0CN |= 0x10

Auto-Scanning

- Configured to automatically scan a sequence of contiguous CS0 input channels
  - Scans all channels defined between the start and the ending channels (CS0SS, CS0SE)
  - Remains enabled even in Suspend mode to enable “wake on touch”
  - Using auto-scan with the CS0 comparator interrupt enabled allows a system to detect a change in measured capacitance without requiring any additional dedicated MCU resources



18

The auto-scanning feature enables the automatic conversion of a sequence of channels. The starting port input and the ending port input are programmed into the SFRs to set the range of channels to be converted. Once started, the CS0 module will continuously convert the range of channels without CPU intervention. We mentioned that the CS0 module is also self-timed. With this autonomous operation the MCU can detect a change even while entering a low power mode where there is no CPU activity. The auto-scan logic can also be used while the accumulator is enabled to provide high SNR measurements across multiple channels.

Auto-Scan and Accumulate Summary

Auto-Scan Enabled	Accumulator Enabled	CS0 Conversion Complete Interrupt Behavior	CS0 Greater Than Interrupt Behavior	CS0MX Behavior
N	N	CS0INT Interrupt serviced after 1 conversion completes	Interrupt serviced after 1 conversion completes if value in CS0DH:CS0DL is greater than CS0THH:CS0THL	CS0MX unchanged, next conversion will be taken on channel *N*
N	Y	CS0INT Interrupt serviced after *M* conversions complete	Interrupt serviced after *M* conversions complete if value in 16-bit accumulator is greater than CS0THH:CS0THL	CS0MX unchanged, next conversion will be taken on channel *N*
Y	N	CS0INT Interrupt serviced after 1 conversion completes	Interrupt serviced after conversion completes if value in CS0DH:CS0DL is greater than CS0THH:CS0THL; Auto-Scan stopped	If Greater Than Comparator detects conversion value is greater than CS0THH:CS0THL, CMUX0 is left unchanged; otherwise, CMUX0 updates to *N+1* and wraps back to CS0SS after passing CS0SE
Y	Y	CS0INT Interrupt serviced after *M* conversions complete	Interrupt serviced after *M* conversions complete if value in 16-bit accumulator is greater than CS0THH:CS0THL; Auto-Scan stopped	If Greater Than Comparator detects conversion value is greater than CS0THH:CS0THL, CS0MX is left unchanged; otherwise, CS0MX updates to *N+1* and wraps back to CS0SS after passing CS0SE
M = Accumulator setting (1x, 4x, 8x, 16x) N = CS0 input channel selected in CS0MX				

19

In this slide we see a summary of how the auto scan hardware and the hardware accumulator operate together. If both settings are disabled then the module performs a single conversion on the selected channel upon receipt of the trigger. If the accumulator is enabled without auto-scan, then the module will perform a conversion upon receipt of the trigger source and then store it in the accumulator. Once the repeat count is met then the values are shifted and the interrupt is generated presenting the hardware averaged result. When the Auto-scan logic is enabled and the accumulator hardware is not enabled, then an interrupt is generated after each conversion while the auto-scan logic cycles the mux input to the next channel. This process continues through the range of channels and wraps back to the first channel. When both the auto-scan logic and the accumulator is enabled the autoscan logic sets the appropriate mux channel. After the programmed number of conversions have been accumulated an interrupt is generated and the auto-scan logic cycles to the next channel. This allows for the autonomous averaging of up to 16 channels greatly reducing CPU overhead.

CS0 Control: CS0CN Register

Bits	Name	Function
7	CS0EN	***CS0 Enable*** 0: CS0 disabled and in low-power mode 1: CS0 enabled and ready to convert
6	UNUSED	Unused. Read = 0b; Write = Don't care
5	CS0INT	***CS0 Interrupt Flag*** 0: CS0 has not completed a data conversion since the last time CS0INT was cleared 1: CS0 has completed a data conversion
4	CS0BUSY	***CS0 Busy*** Read: 0: CS0 conversion is complete or a conversion is not currently in progress 1: CS0 conversion is in progress Write: 0: No effect 1: Initiates CS0 conversion if CS0CM[2:0] = 000 b, 110 b, or 111 b
3	CS0CMPEN	***CS0 Digital Comparator Enable Bit*** Enables the digital comparator, which compares accumulated CS0 conversion output to the value stored in CS0THH:CS0THL 0: CS0 digital comparator disabled 1: CS0 digital comparator enabled
2:1	UNUSED	Unused. Read = 00b; Write = Don't care
0	CS0CMPF	***CS0 Digital Comparator Interrupt Flag*** 0: CS0 result is smaller than the value set by CS0THH and CS0THL since the last time CS0CMPF was cleared 1: CS0 result is greater than the value set by CS0THH and CS0THL since the last time CS0CMPF was cleared

SILICON LABS

We won't cover all of the register settings but can highlight a few here. See the data sheet for complete details. CS0BUSY is the bit that software can use to initiate conversions on demand as well as start the auto-scan. In the control register we also enable the peripheral and control the settings for the window comparator.

CS0 Configuration: CS0CF Register

Bits	Name	Function
7	UNUSED	*Unused.* Read = 0b; Write = Don't care
6:4	CS0CM[2:0]	CS0 Start of Conversion Mode Select 000: Conversion initiated on every write of '1' to CS0BUSY 001: Conversion initiated on overflow of Timer 0 010:Conversion initiated on overflow of Timer 2 011: Conversion initiated on overflow of Timer 1 100: Conversion initiated on overflow of Timer 3 101: Reserved 110: Conversion initiated continuously after writing 1 to CS0BUSY 111: Auto-scan enabled, conversions initiated continuously after writing 1 to CS0BUSY
3:2	UNUSED	*Unused.* Read = 00b; Write = Don't care
1:0	CS0ACU[1:0]	*CS0 Accumulator Mode Select* 00: Accumulate 1 sample 01: Accumulate 4 samples 10: Accumulate 8 samples 11: Accumulate 16 samples

The configuration register enables the use and control of the conversion trigger source and how many samples will be taken by the accumulator.

CS0 Code: I/O and CS0 Config

- ***Step 1***: Set port pins to analog inputs
- ***Step 2***: Enable auto-scan to convert from write to CS0BUSY
- ***Step 3***: Enable the CS0 module and the comparator
- ***Step 4***: Set the auto-scan channels
  - Writing to CS0SS when CS0CF[6:4] = 111 b copies the value to CS0MX
- ***Step 5***: Set the greater than threshold using CS0TH
- ***Step 6***: Write to CS0BUSY to start conversions: CS0CN |= 0x10

```
void PORT_Init (void)
{
   U8 SFRPAGE_save = SFRPAGE;          // Save the current SFRPAGE

   SFRPAGE = CONFIG_PAGE;

   P0SKIP = 0xFF;                      // Skip all P0, route PCA to P1.0
   P1MDOUT |= 0x01;                    // P1.0 is push-pull
   P2MDIN &= ~0x0F;                    // P2.0-P2.3 set to analog input for
                                       // CS0
   XBR1    = 0x41;                     // Enable crossbar and enable
                                       // weak pull-ups, 1 PCA channel
   SFRPAGE = SFRPAGE_save;
}
```

```
void CS0_Init (void)
{
   U8 SFRPAGE_save = SFRPAGE;      // Save the current SFRPAGE

   SFRPAGE = LEGACY_PAGE;
   CS0CF = 0x70;                   // Auto-scan enabled, Convert on CS0BUSY
   CS0CN = 0x88;                   // Enable CS0, comparator
   CS0SS = 0x00;                   // Set channel 0 as autoscan starting channel
   CS0SE = 0x03;                   // Set channel 1 as autoscan end channel
   CS0TH = SWITCH_THRESHOLD;       // Set comparator threshold value
   CS0CN |= 0x10;                  // Set CS0BUSY to begin conversions
   EIE2 |= 0x02;                   // Enable CS0 Greater than comp. interrupts
   SFRPAGE = CONFIG_PAGE;
   SFRPAGE = SFRPAGE_save;
}
```

Code examples can be found in the tools directory: Silabs/MCU/examples\C8051F70x_71x

22

Here we see some code examples for initializing the peripheral. The first step is making sure the I/Os are set to provide the analog input required to detect the capacitance change on the pin. Next we have to initialize the CS0 peripheral. Here is where we set the different operational parameters required by the application, like the auto-scan enable or accumulation of samples.

CS0 Code: CS0 Interrupt

- ***Step 1***: Clear the interrupt pending bit
- ***Step 2***: Auto-scan will stop on the channel when the greater than threshold is reached. Use CS0MX to select the channel
- ***Step 3***: Perform the function for the pad selected
- ***Step 4***: Restart auto-scan since CS0BUSY halts after the comparator output triggers an interrupt

```
INTERRUPT(CS0_ComparatorInterruptServiceRoutine, INTERRUPT_CS0_GRT)
{
   static S16 counter;
   CS0CN &= ~0x01;            // Acknowledge interrupt

   switch (CS0MX)             // Determine which button was pressed
   {
      case (0):
         PCA0CPH0 = 0xF0;     // Set PCA duty cycle to change
         break;               // LED's brightness
      case (1):
         PCA0CPH0 = 0xB0;
         break;
      case (2):
         PCA0CPH0 = 0x80;
         break;
      case (3):
         PCA0CPH0 = 0x40;
         break;
   };
   CS0CN |= 0x10;             // Set CS0Busy to start another scan
}
```

Benefits of the Implementation

- Increased sensitivity to change in capacitance
- Hardware accumulator
  - Decreases system noise
  - Reduces CPU overhead
- Insensitive to noise and supply voltage changes
  - Insensitive to 50/60 Hz noise due to sample time
  - Differential sampling doesn't require precise supply voltage
- No external hardware overhead
  - Directly connect the switch traces to the MCU port pins
  - No other external feedback resistors or capacitors are needed
- Low MCU overhead
  - Autonomous peripheral allows the MCU to go into low-power mode and wake up on a switch event
- Simple configuration
  - Easy to perform with any material on the switch
  - Software API and GUI support
    - Configuration wizard for capacitive sensing API
    - Human Interface Studio

Where Can I Learn More?

- Visit the Silicon Labs website to get more information on Silicon Labs products, technologies and tools

- The Education Resource Center training modules are designed to get designers up and running quickly on the peripherals and tools needed to get the design done

  - http://www.silabs.com/MCU
  - http://www.silabs.com/ERC
  - C8051F700 data sheet
  - Capacitive Sensing Overview online training module
  - AN367: *Understanding Capacitive Sensing Signal To Noise Ratios and Setting Reliable Thresholds*

- To provide feedback on this or any other training go to the training page and submit an email

SILICON LABS

25

Visit the Silicon Labs Education Resource Center to learn more about the MCU products


SILICON LABS
www.silabs.com/MCU