Ultra-low Power Sensing Applications with TI’s CC26xx/CC13xx family of SimpleLink CortexM SOCs.

Using the CC13x2/CC26x2 Sensor Controller

Merril Newman - Digital Field Applications
Schaumburg, Illinois

Note.. This presentation is a mix of slides and interactive content from www.ti.com. Look for slides marked “WebPath”
WiFi: AT&T pops up: *I have a coupon code* – clebh2019

Presentations: Will appear at ti.com/techdays

Survey: Customers will receive an email from TI on Wednesday asking them for Feedback
TI Training – Summary

CC13x2/CC26x2 Ultra-Low Power Sensing:
This session will showcase how the updated Sensor Controller in CC13x2/CC26x2 can be leveraged for ultra-low power sensing applications. By using the ULP-Sense BoosterPack with a CC1352 LaunchPad, several application examples will be discussed as well as demonstrated to showcase the ultra-low power consumption.

• What you’ll learn:
  – **What** the Sensor Controller is, **how** to use it, and which **tools** to use for development and evaluation.
  – Understand how this enables **ultra-low power operation** of your applications.
  – See how Sensor Controller **application examples** are implemented and how they can be evaluated.
Ultra-low Power Sensing Applications
*Using the CC13x2/CC26x2 Sensor Controller*

**Part 1: What is the Sensor Controller?**

The “Sensor Controller” is..
An advanced low-power peripheral included on..
TI’s CC26xx/CC13xx devices which are..
A part of TI’s [www.ti.com/simplelink](http://www.ti.com/simplelink) family of..
Cortex M3/M4F.. featuring both wired & wireless SOCs..
CC26xx/CC13xx Integrated Sensor Controller Engine

What are CC26xx / CC13xx devices? Let’s visit ti.com to see..

CC26xx / CC13xx are a part of TI’s www.ti.com/simplelink family of Cortex M3/M4F.. featuring both wired & wireless SOCs..
CC26xx/CC13xx Integrated Sensor Controller Engine

What is CC26xx / CC13xx? Let’s visit ti.com to see..

From [www.ti.com/simplelink](http://www.ti.com/simplelink), Click on “BLE Overview”.. Then Click on CC2642 “Download Datasheet”..
CC26xx/CC13xx Integrated Sensor Controller Engine

What is CC26xx / CC13xx? Let’s visit ti.com to see..

![Functional Block Diagram]

Figure 1-1. CC2642R Block Diagram

[Direct Link to Datasheet: http://www.ti.com/lit/ds/symlink/cc2642r.pdf]
Sensor Controller Engine

- 16 bit custom CPU
- Ultra Low Power Handling of
  - Analog sensors
  - Digital Sensors
- Flexible (programmable)
- Sequencing

ARM® Cortex®-M4F

Radio

SCE

Memory

Peripherals / modules
What is the Sensor Controller?

- **Active Mode**
  - 3 mA
  - Cortex M4F Active Mode 48 MHz

- **Standby Mode**
  - 0.9 uA
  - RTC running, RAM/CPU retention
What is the Sensor Controller?

- **Active Mode**
  - 3 mA
  - Cortex M4 Active Mode 48 MHz
  - SC Active Mode processing (24 MHz)
  - SC Low Power-Mode processing (2 MHz)
  - Capacitive Touch, two buttons
  - Flow Meter @ 16 Hz
  - Comp A Reading @ 100 Hz
  - SPI
  - ADC sampling @ 1 Hz

- **Standby Mode**
  - 0.9 uA
  - SC Standby Mode.. RTC running, RAM/CPU retention
### Sensor Controller Peripherals

#### Analog Domain
- **COMPA**: High speed Continuous time
- **COMPB**: Low power Clocked at 32 kHz
- **Analog MUX**

#### Current Source:
- 0.25 uA to 20 uA

#### ADC:
- 12 bit, 200 ksps

#### DAC:
- 8 bit

#### Digital Domain
- **16 bit CPU w/4k ULL RAM**
- **Pulse Counter**
- **Time-to-Digital Converter**
- **2x Timers**
- **1x Timer w/PWM + 4 channel Cap/Comp**
- **8 analog/digital + 23 digital GPIO**
- **SPI Master**
- **16x16b Multiply and 40b Accumulate**
Sensor Controller Peripherals: Low Power Mode

**Analog Domain**
- **COMPA:** High speed
  - Continuous time
- **COMPB:** Low power
  - Clocked at 32 kHz
- **Analog MUX**
- **Current Source:** 0.25 uA to 20 uA
- **ADC:** 12 bit, 200 ksamples per second
- **DAC:** 8 bit

**Digital Domain**
- **16 bit CPU w/4k ULL RAM**
- **Pulse Counter**
- **Time-to-Digital Converter**
- **2x Timers**
- **1x Timer w/PWM + 4 channel Cap/Comp**
- **8 analog/digital + 23 digital GPIO**
- **SPI Master**
- **16x16b Multiply and 40b Accumulate**
## Sensor Controller Peripherals: Standby Mode

### Analog Domain
- **COMPA**: High speed, Continuous time
- **COMPB**: Low power, Clocked at 32 kHz
- **Analog MUX**
- **ADC**: 12 bit, 200 ksps
- **DAC**: 8 bit

### Current Source
- 0.25 uA to 20 uA

### Digital Domain
- **16-bit CPU** w/4k ULL RAM
- **Pulse Counter**
- **Time-to-Digital Converter**
- **2x Timers**
- **1x Timer w/PWM + 4 channel Cap/Comp**
- **8 analog/digital + 23 digital GPIO**
- **SPI Master**
- **16x16b Multiply and 40b Accumulate**
Access to System Functionality

- **BATMON:**
  - Monitor the VDDS voltage and measure on-chip temperature
- **RTC:**
  - Timestamp measurement results
  - Calculate the time between wake-ups
- **VDDR recharge monitoring and triggering:**
  - Ensure that VDDR recharging current spikes do not occur during measurements
Ultra-low Power Sensing Applications

Using the CC13x2/CC26x2 Sensor Controller

Part 2: Sensor Controller Application Examples
Application Examples - BOOSTXL-ULPSENSE

First.. What is a LaunchPad & what is a BoosterPack? Lets go to [http://dev.ti.com/tirex](http://dev.ti.com/tirex) to see..

ULPXL-ULPSENSE is..

- Capacitive Touch
- Ultra-low Power UART RX and TX  
  – RX is possible in Standby mode  
  Sub-1 uA current consumption
- Readout of SPI sensors
- Read analog sensors
- Flow Metering

Mrn- WebPath .. [http://dev.ti.com/tirex](http://dev.ti.com/tirex) (or specifically..)

[http://dev.ti.com/tirex/explore/node?devtools=BOOSTXL-ULPSENSE&node=AEF1LxEBq1JoDkXd3YNvzA__FUz-xrs__LATEST](http://dev.ti.com/tirex/explore/node?devtools=BOOSTXL-ULPSENSE&node=AEF1LxEBq1JoDkXd3YNvzA__FUz-xrs__LATEST)
Application: Flow Meter

• The CPU enables many different implementations of Flow Metering solutions

• Implemented example with BOOSTXL-ULPSENSE:
  – COMPA
  – DAC as a reference
  – Pulse Counter
### Application: SW SAR ADC

**Wake-up Interval**

<table>
<thead>
<tr>
<th></th>
<th>1 Hz</th>
<th>20 Hz</th>
<th>100 Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW SAR ADC, SC Low-power Mode</td>
<td>0.96 uA</td>
<td>1.33 uA</td>
<td>3.00 uA</td>
</tr>
<tr>
<td>Regular ADC, SC Active Mode</td>
<td>1.4 uA</td>
<td>4.4 uA</td>
<td>17.4 uA</td>
</tr>
<tr>
<td>Regular ADC, System CPU</td>
<td>2.6 uA</td>
<td>28.7 uA</td>
<td>118.3 uA</td>
</tr>
</tbody>
</table>

**Vin, Vref, DAC**

**Sensor Controller Engine**

<table>
<thead>
<tr>
<th></th>
<th>Sampling Rate</th>
<th>ENOB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular ADC</td>
<td>200 kHz</td>
<td>10.1 bits</td>
</tr>
<tr>
<td>SW SAR ADC</td>
<td>8.8 kHz</td>
<td>6.3 bits</td>
</tr>
</tbody>
</table>
## Application: SPI 18-Bytes

<table>
<thead>
<tr>
<th>Wake-up Interval</th>
<th>1 Hz</th>
<th>20 Hz</th>
<th>100 Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC Low-power Mode, 1 MHz SPI Speed</td>
<td>1.0 uA</td>
<td>1.4 uA</td>
<td>3.0 uA</td>
</tr>
<tr>
<td>SC Active Mode, 12 MHz SPI Speed</td>
<td>1.5 uA</td>
<td>4.0 uA</td>
<td>15.6 uA</td>
</tr>
<tr>
<td>System CPU, 12 MHz SPI Speed</td>
<td>2.4 uA</td>
<td>25.4 uA</td>
<td>119.2 uA</td>
</tr>
<tr>
<td>MSP432, 16 MHz SPI Speed</td>
<td>1.2 uA</td>
<td>5.8 uA</td>
<td>26.2 uA</td>
</tr>
</tbody>
</table>

### 16 bit CPU w/4k ULL RAM

**SPI Master**

Low power SPI Accelerometer on BOOSTXL-ULPSENSE: 2 uA @ 100 Hz
Ultra-low Power Sensing Applications

*Using the CC13x2/CC26x2 Sensor Controller*

Part 3: Development Environment
Sensor Controller Studio

• Standalone tool for development and debugging of the Sensor Controller
  – Sensor Controller Task development and debugging
  – Using a simple programming language with syntax similar to C
  – Generates a driver interface for the System CPU

• Included application examples:
  – ADC logger, streamer, window monitor
  – SW SAR ADC
  – Button Debouncer
  – Capacitive touch
  – I2C readout (bit-banged)
  – LC Flow Meter
  – LED Blink, PWM
  – UART Emulation
  – SPI readout
Sensor Controller Programming Model

- A task is initialized from the System CPU. Then the task is kept alive by events or through scheduling.
- Sensor Controller tasks are fully independent and cannot transfer data or control each other.
- Up to 8 tasks are supported.
Resource Allocation

- Peripheral, I/O control, Power Management, Event trigger
- A resource can only be used by one Sensor Controller Task
- Resource sharing is only possible between one task and the System CPU
Data Structures

• **cfg:**
  – Set by the System CPU application before starting the task

• **input:**
  – Transfer data to the Sensor Controller task

• **output:**
  – Transfer data to the System CPU application
  – Supports Multi-Buffered Output Data Exchange

• **state:**
  – Internal variables to store the task’s state between iterations
**Task Code Blocks**

- **Initialization Code**
  - The system CPU starts the Sensor Controller by:
    - Loading the SC FW image into AUX RAM
    - Trigger the Initialization Code Block
  - The Initialization Code must schedule the first execution and/or setup the initial event trigger(s)

- **Execution Code**
  - Application code triggered synchronously by the RTC or System CPU
  - Keeps the task alive by scheduling the next execution and/or setup new event triggers

- **Event Handler Code**
  - Application code triggered by an event.
  - Up to 3 event handlers per project

- **Termination Code**
  - Executed when the System CPU stops a task
Run-Time Logging

• Graphical representation of data structures
• Modify and monitor variables
• Adjust parameters
• Tasks are running in full speed on Sensor Controller HW
• The **System CPU** is reading data from AUX RAM and transmitting over UART to Sensor Controller Studio
Task Testing

- Low-level debugging: step through the assembly code
- Graphical representation of data structures
- Modify and monitor constants and data structures
- Adjust parameters
- Sensor Controller Studio acts as the System CPU application and is responsible for controlling the Sensor Controller task
Sensor Controller Interface Driver

• Driver setup:
  – The AUX RAM image,
  – Data structure definitions
  – I/O mapping,

• A generic application programming interface (API) for:
  – Initializing the driver
  – Task control
  – Task data exchange
  – Un-initializing the driver

• An operating system abstraction layer (OSAL) that:
  – Enables seamless integration with the operating system and,
  – Power and clock management system running on the System CPU
Sensor Controller “How-To-Use Guide”

- The code generator outputs a tailored user guide for integrating the Sensor Controller project into an application running on the System CPU.

Include the SCIF Driver Header File

Add the following code to include the SCIF driver main header file:

```c
#include "scif.h"
```

Add SCIF Driver Callback Functions

Add these SCIF driver callback functions to the application:

```c
// SCIF driver callback: Task control interface ready (non-blocking task control
void scCtrlReadyCallback(void) {
}
```

```c
// SCIF driver callback: Sensor Controller task code has generated an alert Interrupt
void scTaskAlertCallback(void) {
}
```

Start Sensor Controller Tasks (Simplified)

The SCIF driver uses the task control interface to start and stop Sensor Controller tasks, and perform other task control operations. The Task Control section describes how to use the task control functions correctly, but for now we will only perform one operation that starts the Sensor Controller task.

To start the “Capacitive Touch” task:

```c
// Start the "Capacitive Touch" Sensor Controller task
scfStartTask.listBox[1] = SCIF_CAPACITIVE_TOUCH_TASK_ID;
```

Access Sensor Controller Data Structures from the Application

The application can directly access the Sensor Controller’s `cfg`, `input`, `output` and `state` data structures, which are located in the AUX RAM. This is the only and only mechanism for exchanging data between the Sensor Controller and the application.

Resource-specific data structure members should not be accessed directly, but rather through the resource provided API.

Capacitive Touch

The application will typically write to the `cfg` data structure before the task is started:

```c
scfTaskData.capacitiveTouch.cfg.inCapCurrent = ...;
scfTaskData.capacitiveTouch.cfg.minSepPeriod = ...;
```
Sensor Controller Studio – Live Demo..
Sensor Controller Studio – eLock Demo

DMM eLock Block Diagram

- CC1312R1
- CC1352R1
- MSP430FR2522 CapTIvate
- QVGA Display
- UART
- GUI Composer
- I2C
- SPI
- Sub-1GHz
- Bluetooth

Texas Instruments
Download Sensor Controller Studio.
www.ti.com/tool/sensor-controller-studio

TI Resources..
www.ti.com
www.ti.com/applications
www.ti.com/SimpleLink
http://dev.ti.com/tirex

Remember to look / post to..
http://e2e.ti.com