AGENDA

- ACHIEVING LOW POWER
- SYNERGY LOW POWER IMPLEMENTATION – HARDWARE
- SYNERGY LOW POWER IMPLEMENTATION – SOFTWARE
- SYNERGY LOW POWER DEMO
How do we achieve Low Power?

The combination of 4 key factors determines low power consumption

Average Current = (Active current \times \text{Active duty cycle}) + (Standby current \times \text{Standby duty cycle})

Keys to achieve low power consumption

- Low active power
- Low standby power (STOP/Halt, LVD, WDT, Interval timer)
- High processing throughput
- Minimum Active vs. Standby time
Synergy Low Power Implementation – Hardware

- **Process:**
  - S7 & S5 → 40nm, 2.7V-3.6V
  - S3 & S1 → 130nm, 1.6V-5.5V

- **Operating temperature range:**
  - -40°C to 105°C

**Four Microcontroller Series**

- **S1**
  - ARM Cortex M0+
  - 32 MHz
  - 128KB
  - Ultra-Low Power
  - Core Frequency Up to 32 MHz

- **S3**
  - ARM Cortex M4
  - 48 MHz
  - 1MB
  - High Efficiency
  - Core Frequency To 100 MHz

- **S5**
  - ARM Cortex M4
  - 120 MHz
  - 2MB
  - High Integration
  - Core Frequency To 200 MHz

- **S7**
  - ARM Cortex M4
  - 240 MHz
  - 4MB
  - High Performance
  - Core Frequency To 300 MHz
## Synergy MCU S3 Series

### 48-MHz ARM® Cortex®-M4 CPU

#### Memory
- Code Flash (1 MB)
- Data Flash (16 KB)
- SRAM (192 KB)
- Flash Cache
- Security MPU
- Memory Mirror Function

#### Analog
- 14-Bit A/D Converter (28 ch.)
- 12-Bit D/A Converter x2
- Low-Power Analog Comparator x2
- High-Speed Analog Comparator x2
- OPAMP x4
- Temperature Sensor

#### Timing & Control
- General PWM Timer 32-bit x10
- Asynchronous General Purpose Timer x2
- WDT

#### Connectivity
- USBFS
- CAN
- SDHI/MMC
- Serial Communications Interface x6
- IrDA Interface
- QSPI
- SPI x2
- IIC x3
- SSI x2
- External Memory Bus

#### System & Power Mgmt
- DMA Controller (4 ch.)
- Data Transfer Controller
- Event Link Controller
- Low Power Modes
- Multiple Clocks
- Port Function Select
- RTC
- SysTick

#### HMI
- Capacitive Touch Sensing Unit (35 ch.)
- Segment LCD Controller

#### Safety
- ECC in SRAM
- SRAM Parity Error Check
- Flash Area Protection
- ADC Diagnostics
- Clock Frequency Accuracy Measurement Circuit
- CRC Calculator
- Data Operation Circuit
- Port Output Enable for GPT
- IWDT

#### Security & Encryption
- 128-bit Unique ID
- TRNG
- AES (128/256)
- GHASH

---

© 2016 Renesas Electronics Corporation. All rights reserved.
Synergy Low Power Implementation – Hardware

Achieving Low Average Current consumption on Synergy via Hardware

![Diagram showing current consumption over time with active and standby states]

System & Power Mgmt

- DMA Controller (4 ch.)
- Data Transfer Controller
- Event Link Controller
- Low Power Modes
- Multiple Clocks
- Port Function Select
- RTC
- SysTick
Data Transfer Controller (DTC) (7,5,3,1)

Transfers data between memory and peripherals without CPU intervention.

DTC is activated by peripheral interrupts.

- **Normal Transfer Mode**
  - One data transfer per activation

- **Repeat Transfer Mode**
  - One data transfer per transfer request
  - Max. no. of repeat t/f: 256
  - Max. t/f data size = 256 x 32 bit = 1 KB

- **Block Transfer Mode**
  - One block data transfer per activation
  - Max. block size = 1 KB

Data transfer unit size: 1 byte (8 bits), 1 word (16 bits), 1 longword (32 bits)

- 1 to 256 data units (word) per block
- Supports chained transfers and sophisticated data scatter/gathering
- Interrupt generation upon transfer completion
- Event Link Controller function output
- Snooze Mode operation

Simplified implementation of the Data Transfer Controller

© 2016 Renesas Electronics Corporation. All rights reserved.
Event Link Controller (ELC) (7,5,3,1)

Allows direct interaction between different modules without CPU intervention

Snooze Mode operation

Routes source events generated by a peripheral to event inputs on other peripherals

Event signal can activate a peripheral for the desired operation

- Start/stop/clear timer, up/down counting
- Start ADC and DAC conversion
- Start cap touch measurement
- Start DMA/DTC transfer
- Issue interrupts to the CPU
- Change state of GPIOs

Peripheral classification based on event signals

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Event Link Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only Generate Event Signals</td>
<td></td>
</tr>
<tr>
<td>Only Accept Event Signals</td>
<td></td>
</tr>
<tr>
<td>Generate and Accept Event Signals</td>
<td></td>
</tr>
</tbody>
</table>
Low Power Modes (7,5,3,1)

Low power modes provide operational flexibility which can dramatically reduce current consumption.

Independent wake-up signals for data acquisition and data transmission peripherals

Allow operation of a peripheral while keeping the CPU and other peripherals disabled.

<table>
<thead>
<tr>
<th>Mode of Operation</th>
<th>Core</th>
<th>Flash</th>
<th>SRAM</th>
<th>RTC, AGT, Vbatt, LVD</th>
<th>Other Peripherals</th>
<th>IO Pins</th>
<th>Snooze</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Operating</td>
<td>Selectable</td>
<td>Selectable</td>
<td>Selectable</td>
<td>Selectable</td>
<td>Selectable</td>
<td>NA</td>
</tr>
<tr>
<td>Sleep</td>
<td>Clock Gated</td>
<td>Selectable</td>
<td>Selectable</td>
<td>Selectable</td>
<td>Selectable</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Software Standby</td>
<td>Clock Gated</td>
<td>Data Retained</td>
<td>Selectable</td>
<td>Clock Gated</td>
<td>State Retained</td>
<td>Available</td>
<td>NA</td>
</tr>
<tr>
<td>Deep Software Standby</td>
<td>Powered Off</td>
<td>Powered Off</td>
<td>Selectable</td>
<td>Powered Off</td>
<td>State Retained</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Comparison among various modes of operation

*Selectable means that operating or disable can be selected by control registers. Some modules are also controlled by module-stop bit.

![Power Consumption Throughput Graph]

<table>
<thead>
<tr>
<th>Mode of Operation</th>
<th>S7</th>
<th>S5</th>
<th>S3</th>
<th>S1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>18 mA</td>
<td>TBD</td>
<td>11.8 mA</td>
<td>3.43 mA</td>
</tr>
<tr>
<td>Sleep</td>
<td>15 mA</td>
<td>TBD</td>
<td>2.4 mA</td>
<td>0.58 mA</td>
</tr>
<tr>
<td>Software Standby</td>
<td>1.2 mA</td>
<td>TBD</td>
<td>0.9 µA</td>
<td>0.44 µA</td>
</tr>
<tr>
<td>Deep Software Standby</td>
<td>16 µA</td>
<td>TBD</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

Typical current consumption in various modes of operation (target values)

© 2016 Renesas Electronics Corporation. All rights reserved.
# Low Power Modes

<table>
<thead>
<tr>
<th>Mode of Operation →</th>
<th>Normal</th>
<th>Sleep</th>
<th>Software Standby</th>
<th>Battery Backup Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Test Conditions ↓</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Current Consumption (mA)</strong></td>
<td>11.8 mA</td>
<td>2.4 mA</td>
<td>0.9 µA</td>
<td>0.6 µA</td>
</tr>
<tr>
<td><strong>Vcc (V)</strong></td>
<td>3.3</td>
<td>3.3</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td><strong>Vbatt (V)</strong></td>
<td>3.3</td>
<td>3.3</td>
<td>3.3</td>
<td>3.0</td>
</tr>
<tr>
<td><strong>System Frequency (MHz)</strong></td>
<td>48</td>
<td>8</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Code</strong></td>
<td>While (1)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Measurement condition</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>Operating</td>
<td>Clock Gated</td>
<td>Clock Gated</td>
<td>Power Off</td>
</tr>
<tr>
<td><strong>Flash</strong></td>
<td>Operating</td>
<td>Disabled</td>
<td>Data Retained</td>
<td>Power Off</td>
</tr>
<tr>
<td><strong>SRAM</strong></td>
<td>Disabled</td>
<td>Disabled</td>
<td>Data Retained</td>
<td>Power Off</td>
</tr>
<tr>
<td><strong>RTC, LVD, OSC</strong></td>
<td>Clock Gated</td>
<td>Clock Gated</td>
<td>Clock Gated</td>
<td>Power Off</td>
</tr>
<tr>
<td><strong>Other Peripherals</strong></td>
<td>Disabled</td>
<td>Disabled</td>
<td>Clock Gated</td>
<td>Power Off</td>
</tr>
</tbody>
</table>

*Current consumption (target, typical) and test conditions for various modes of operation*
Power Control Modes (3,1)

The OPCCR register is used to reduce power consumption in normal operating mode and Sleep Mode.

Power consumption can be reduced according to the operating frequency:

- **High-Speed Operating Mode**
  - Normal Operating Mode up to max. frequency

- **Middle-Speed Mode**
  - Lower Power domain settings

- **Low-Voltage Mode**
  - 1.6V operation possible

- **Low-Speed Mode**
  - Up to 1 MHz operation possible / Flash P/E not possible

- **Subosc-Speed Mode**
  - 32 kHz operating frequency / Flash O/E not possible
  - Limitations in clock settings and Peripheral usage

<table>
<thead>
<tr>
<th>Operating power control mode</th>
<th>SOPCM bit</th>
<th>OPCM[1:0] bits</th>
<th>Power consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-speed mode</td>
<td>0</td>
<td>00b</td>
<td>High</td>
</tr>
<tr>
<td>Middle-speed mode</td>
<td>0</td>
<td>01b</td>
<td></td>
</tr>
<tr>
<td>Low-voltage mode</td>
<td>0</td>
<td>10b</td>
<td></td>
</tr>
<tr>
<td>Low-speed mode</td>
<td>0</td>
<td>11b</td>
<td></td>
</tr>
<tr>
<td>Subosc-speed mode</td>
<td>1</td>
<td>xxb</td>
<td></td>
</tr>
</tbody>
</table>

Max. frequencies in different Modes depending on Supply Voltage:

<table>
<thead>
<tr>
<th>ICLK</th>
<th>High-speed</th>
<th>Middle-speed</th>
<th>Low-voltage</th>
<th>Low-speed</th>
<th>Subspeed</th>
<th>Supply Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICLK</td>
<td>48</td>
<td>12</td>
<td>4</td>
<td>1</td>
<td>32 (kHz)</td>
<td>2.7 – 5.5V</td>
</tr>
<tr>
<td>ICLK</td>
<td>16</td>
<td>12</td>
<td>4</td>
<td>1</td>
<td>32 (kHz)</td>
<td>2.4 – 5.5V</td>
</tr>
<tr>
<td>ICLK</td>
<td>-</td>
<td>8</td>
<td>4</td>
<td>1</td>
<td>32 (kHz)</td>
<td>1.8 – 5.5V</td>
</tr>
<tr>
<td>ICLK</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>1</td>
<td>32 (kHz)</td>
<td>1.6 – 5.5V</td>
</tr>
</tbody>
</table>
Battery Backup

Battery backup area that can be separately powered by battery

When VCC voltage drop is detected, power source switched to dedicated battery backup power pin, VBATT

When VCC rises again, power source switched from VBATT to VCC pin

Features supported in battery backup area:

- LOCO* and Sub Oscillator
- RTC
- Time capture event input (RTCIC0~2)
- AGT* (Channel 0 & 1)
- VBATT low voltage detection*
- VBATT_R low voltage detection*
- Backup registers (512 B of backup memory)
- Wakeup control function*

* Available on S3 series only.
Snooze

Snooze feature provides operational flexibility to dramatically reduce current consumption

Allows operation of certain peripherals while keeping the CPU and other peripherals remain retained, for example:

- ADC, DAC, CTSU, SCI0, AGT
- ADC can sample analogue values while in Snooze
- SCI0 can receive addressed data while in Snooze
- DTC can service data in the background while in Snooze

Independent wake-up signals for data acquisition and data transmission peripherals
Clock Management (3)

Multiple clock sources enable flexible operation and optimization of devices across a variety of power and performance points.

- **Main Clock Oscillator**
  - 1 to 20 MHz resonator frequency
  - Up to 20 MHz external clock frequency
  - Drive capability switching
  - Automatic clock switching upon oscillation stop detection

- **Sub-Clock Oscillator**
  - Requires 32.768 kHz external crystal
  - Drive capability switching

- **PLL**
  - 4 to 12.5 MHz input frequency
  - 24 to 64 MHz output frequency

- **On-Chip Oscillator**
  - High-speed: 24, 32, 48, 64 MHz
  - Middle-speed: 8 MHz
  - Low-speed: 32.768 kHz

- **Independent Watchdog Timer Oscillator**
  - Dedicated for watchdog timer operation
  - 15 kHz

- **External Debugging Clock**
  - For JTAG, SWD
  - Up to 25 MHz
Realtime Clock (RTC) (7,5,3,1)

Integrated RTC with calendar functionality
Operates using a 128 Hz clock from either
- Sub-Clock Oscillator, or
- Low-Speed On-Chip Oscillator (LOCO)

**Modes of operation**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Details</th>
</tr>
</thead>
</table>
| Calendar Count Mode      | ✓ Supports 100 year calendar (2000 to 2099)
|                          | ✓ Automatic adjustment for leap year
|                          | ✓ 12/24 hour, AM/PM switching function
|                          | ✓ Year, month, day, date, minute and second counting |
| Binary Count Mode        | ✓ Count seconds serially in 32 bits |

Clock error correction function
1 or 64 Hz clock output via RTCOUT pin
Time (year, month, day, date, hour, minute and second) capture upon activity on RTC input pins
Send event signals to other peripherals while CPU is in sleep mode
Interrupt on alarm, periodic interrupt, and several other interrupts

Simplified block diagram of the RTC
Asynchronous General Purpose Timer (AGT) (7,5,3,1)

16-bit asynchronous timer for pulse output, external pulse width (period) measurement, and external event counting

Modes of operation

- Timer mode
  - Count source is counted
- Pulse Output mode
  - Count source is counted and output is inverted upon timer underflow
- Event Counter mode
  - An external event is counted
- Pulse Width Measurement mode
  - An external pulse width is measured
- Pulse Period Measurement mode
  - An external pulse period is measured

Multiple clock sources
- Peripheral clock (PCLK)
- Low-Speed On-Chip Oscillator (LOCO)
- Sub-clock (fSUB)
- Underflow signal of timer

Can be operated asynchronously in low power mode without the PCLK
Renesas SSP

ThreadX® RTOS
Multitasking real time kernel with preemptive scheduling and small memory footprint. Stable heartbeat of the system.

Stacks & Middleware
X-Ware™ for TCP/IP, USB, color graphics, and file system. Completely optimized and integrated.

Board Support Package
Customized for every Synergy hardware kit and MCU. Easily tailored for end-product.

Hardware Abstraction Layer (HAL) Drivers
Efficient drivers for all peripherals and system services. Eliminates deep study.

Synergy Software Package (SSP)

Software APIs
Standardized ‘C’ language APIs for X-Ware™, Application Framework, Middleware, Libraries, DSP, HAL, BSP, and MCU registers. Abstract the dependencies, ensure portability, and accelerate product development.

Add-Ons
Qualified Software Add-Ons (QSA)
Verified Software Add-Ons (VSA)

Libraries
Specialized software for DSP, touch, security, safety, and more.

Application Framework
System level services linking RTOS to HAL for inter-process messaging, security services, audio playback, serial comm, power mgmt, JPEG conversion, touch, and more. Saves time.

Software APIs

ThreadX® RTOS

FileX®

USBX™

GUIX™

NetX™

NetX Duo™

Application Framework

Functional Libraries

Hardware Abstraction Layer (HAL) Drivers

Board Support Package (BSP)

© 2016 Renesas Electronics Corporation. All rights reserved.
Synergy Low Power Implementation – Software

Achieving Low Average Current consumption on Synergy via Software

Synergy Software

- Software APIs
- Synergy Software Package (SSP)
  - RTOS
  - Middleware & Stacks
  - Application Framework
  - Functional Libraries
  - HAL Drivers
  - Board Support Package (BSP)
- Qualified Software Add-Ons (QSA)
- Verified Software Add-Ons (VSA)
Synergy Low Power Implementation – Software

The SSP supports and HAL Interfaces & Frameworks specific to Low Power

- HAL CGC Interface
- HAL Low Power Modes Interface
- HAL AGT Interface
- HAL ELC Interface
- HAL DTC Interface
- Framework Power Profiles Interface
- Framework Messaging Interface
Synergy Low Power Implementation – Demo

Device is in SWStandby – AGT1 is running on 32.768kHz LOCO

AGT1 Underflows every 1 second – Transition to Snooze mode

AGT1 Underflow is LINKED to the ADC via the ELC and triggers ADC - ADC does conversion

ADC Scan End Interrupt is serviced by the DTC (not the CPU) which transfers result to SRAM - Transition to Active

Check loop count

If < 30 counts, back to SWStandby

If = 30 transition tx data via SCI2 @ 57600 Baud, back to SWStandby
HAL CGC – Managing the clock tree

- Easy configuration of clocks via the ISDE Configurator.
- Everything is setup for you in the BSP on reset.
- APIs are provided for clock switching.
HAL CGC

What Does the CGC Module Do?

You can use the CGC interface to configure the clocks on the Synergy microcontroller as follows:

- Configure any of the available clocks (HOCO, MOCO, LOCO, Main Clock, PLL, Sub-Oscillator) as the system clock source.
- Configure the internal clocks (ICLK, PCLK etc.).
- Switch the clocks on and off.
- Configure the output clocks.
- Set up the Oscillation Stop Detection feature.
HAL CGC – Managing clocks in your application

Managing Clocks within the application

For example, switching from the MOCO to HO CO:

```c
    g_cgc_on_cgc.clockStart(CGC_CLOCK_HOCO, NULL); /* Start the HOCO */
    g_cgc_on_cgc.systemClockGet( NULL, &clock_dividers ); /* Get the current clock dividers */
    
    /* Modify the clock dividers to clock gear */
    clock_dividers.pclka_div = CGC_SYS_CLOCK_DIV_1; /* modify the PLCKA divider to / 1 */
    clock_dividers.pclkd_div = CGC_SYS_CLOCK_DIV_1; /* modify the PL CKD divider to / 1 */

    /* Set the new clock */
    g_cgc_on_cgc.systemClockSet( CGC_CLOCK_HOCO, &clock_dividers ); /* Set the clock and clock dividers */

    g_cgc_on_cgc.clockStop( CGC_CLOCK_MOCO ); /* Stop the previous clock if required*/
```
HAL Timer Interface

In our demo, we use the AGT to transition from SWStandby to Snooze and trigger the ADC conversion. The AGT is added via the configurator.
HAL Timer Interface

To use the AGT

```c
  g_timer.p_api->open( g_timer.p_ctrl, g_timer.p_cfg );
```

To start and stop the AGT

```c
  g_timer.p_api->start( g_timer.p_ctrl );
  g_timer.p_api->stop( g_timer.p_ctrl );
```
HAL ELC Interface

The ELC HAL Interface is a generic API that uses event requests to link different peripherals together without CPU intervention.

The ELC is part of the BSP and is initialised automatically.

Therefore to use it:

```c
  g_elc.p_api->linkSet( ELC_PERIPHERAL_ADC0, ELC_EVENT_AGT1_INT);
```

This creates the LINK to the ADC0 unit, to be triggered by the AGT1 underflow INTERRUPT.
HAL ADC Interface
HAL ADC Interface

/* Setup the ADC */
g_adc.p_api->open( g_adc.p_ctrl, g_adc.p_cfg ); // * Open */

g_adc.p_api->scanCfg( g_adc.p_ctrl, g_adc.p_channel_cfg ); // * Configure the channels to scan */

g_adc.p_api->scanStart( g_adc.p_ctrl ); // * Start – wait for trigger from ELC */
HAL Transfer Interface

What Does the Transfer Interface Do?

The HAL Transfer Interface is a generic interface for Transfer applications and supports the two Transfer peripherals available on the Synergy microcontroller hardware: DMAC and DTC.

Adding the interface to your project

Interface properties
HAL Transfer Interface

/* Setup the DTC */

```c
    g_transfer_ADC.p_api->open( g_transfer_ADC.p_ctrl, g_transfer_ADC.p_cfg );
```

```c
    g_transfer_ADC.p_api->reset( g_transfer_ADC.p_ctrl,
                                &R_S14ADC->ADDRn[13],  // source address
                                g_adc_results,         // destination address
                                g_transfer_ADC.p_cfg->p_info->length );  // number of transfers
```

```c
    g_transfer_ADC.p_api->enable( g_transfer_ADC.p_ctrl );
```
SSP – Active to Standby to Snooze

Considering our use case.

How does the SSP aid the transitions from Active to Standby and from Standby to Active?

Via the lpm (low Power Module) interface
HAL LPM Interface

What does the LPM interface do?
The LPM supports configuration of MCU operating modes and MCU low power modes using the LPM hardware peripheral.
The LPM supports operating modes low-voltage, low-speed, middle-speed, high-speed, and sub-oscillator mode.
The LPM supports low power modes deep standby, standby, sleep, and snooze.
The LPM driver supports disabling and enabling of the MCU's other hardware peripherals.
HAL LPM Interface

How do we use LPM in our demo?

/* Set the low power mode we wish to enter */
g_lpm.p_api->lowPowerCfg( LPM_LOW_POWER_MODE_STANDBY, LPM_OUTPUT_PORT_ENABLE_RETAIN, LPM_POWER_SUPPLY_DEEPCUT0, LPM_IO_PORT_NO_CHANGE);

/* Set the transition source to Snooze mode from SW Standby */
/* DTC will operate in Snooze */
/* AGT1 Underflow Snooze request */
/* leave the Snooze to SWStby blank */
g_lpm.p_api->snoozeEnable( LPM_SNOOZE_RXD0_FALLING_EDGE_IGNORE, LPM_SNOOZE_DTC_ENABLE, LPM_SNOOZE_REQUEST_AGT1_UNDERFLOW, 0);

/* We want to transition to active when the DTC has done its transfer. */
/* Specify this via the SELSR0 register */
/* No API for this at the moment. Direct register access! */
R_ICU->SELSR0_b.SELS = 0x29;

/* Enter sleep mode */
g_lpm.p_api->enterLowPowerMode();
Building on top of the HAL LPM Interface we have a Power Profile Framework.

The Framework Power Profiles Interface is a generic API that can be configured to allow the MCU to be placed in lower power Software Standby mode.

The module can be configured to run in one of three operating modes (Run, RTC and External Interrupt).

These modes determine which clocks and peripherals are disabled during Software Standby, as well as what the output pin states are prior to and after exiting Software Standby mode.

The Interface uses the RTC, LPM, IOPORT and CGC peripherals on the Synergy microcontroller hardware and provides an easy-to-use software interface to access the low power operating modes.
The Power Profile framework is still in development
It provides features beyond the LPM

Pin configurations pre and post software standby
Pre and Post software standby callback events for system management
Power Handling in Multi-Threaded applications

Up until now the demos have been simple single applications.

But what are the implications of power management in a multi threaded application?

If one thread is responsible for placing the device into a power saving mode, how does it communicate this to other threads?

How does it know that other threads have become “safe” for power saving?

The SSP has another Framework that simplifies this process.
Framework Messaging Interface

The Framework Messaging Interface is a lightweight and event-driven Framework API for passing messages between threads.

The Messaging Framework allows applications to communicate messages between two or more threads.

The Framework uses the ThreadX message queue primitive for message passing and provides more benefits than the ThreadX RTOS message queue services alone.

The Messaging Framework API is purely a software API and does not access any hardware peripherals.
Framework Messaging Interface

- LCD Thread
- SPI Temp Sensor Thread
- ADC Thread
- USB Thread
- Low Power Management Thread

SendPowerDownMessage
AllMessageACKsreceived

Messaging Framework
Synergy Low Power Implementation – After 60 mins, $I_{cc(Ave)}$

Device is in SWStandby – AGT1 is running on 32.768kHz LOCO

AGT1 Underflows every 1 second – Transition to Snooze mode

AGT1 Underflow is LINKED to the ADC via the ELC and triggers ADC - ADC does conversion

ADC Scan End Interrupt is serviced by the DTC (not the CPU) which transfers result to SRAM - Transition to Active

Check loop count

If < 30 counts, back to SWStandby

If = 30 transition tx data via SCI2 @ 57600 Baud, back to SWStandby
THANK YOU FOR YOUR ATTENTION

PLEASE REMEMBER TO COMPLETE THE FEEDBACK SURVEY IN YOUR SMARTPHONE APP