Application Report
SWRA537A–March 2017–Revised July 2018

Increase RAM Size on the CC2640R2F Bluetooth® low energy Wireless MCU

Marie Hernes

ABSTRACT

This application report documents ways to increase the available random access memory (RAM) for your application on the CC2640R2F SimpleLink™ Bluetooth low energy microcontroller. By moving initialized data or compiled code from the SRAM into other parts of the memory, the available SRAM is increased. This application report also highlights some of the tools found in the SimpleLink CC2640R2 software development kit (SDK) that enables optimization of the RAM used by heap and stack memory. Note that the features described in this document are not profiled in terms of power consumption or processor speed.

Contents

List of Figures

List of Tables

Trademarks

SimpleLink, Code Composer Studio are trademarks of Texas Instruments.
Bluetooth is a registered trademark of Bluetooth SIG, Inc.

1 CC2640R2F Memory

As illustrated in Figure 1, the CC2640R2F MCU has 20 KB SRAM. In addition, there is 128 KB flash and 8 KB cache. There is also ROM, which is pre-programmed. The CC2640R2F has a Sensor Controller with a 2 KB RAM, known as the AUX RAM. To the right, Figure 1 shows a device running the *simple peripheral* project from the BLE-Stack. In this case, all memory areas have been assigned and filled. The flash has been programmed. The SRAM contains the heap, the system stack and task stacks in addition to the bss section and initialized data. The AUX RAM is blacked out since *simple peripheral* is not using it.

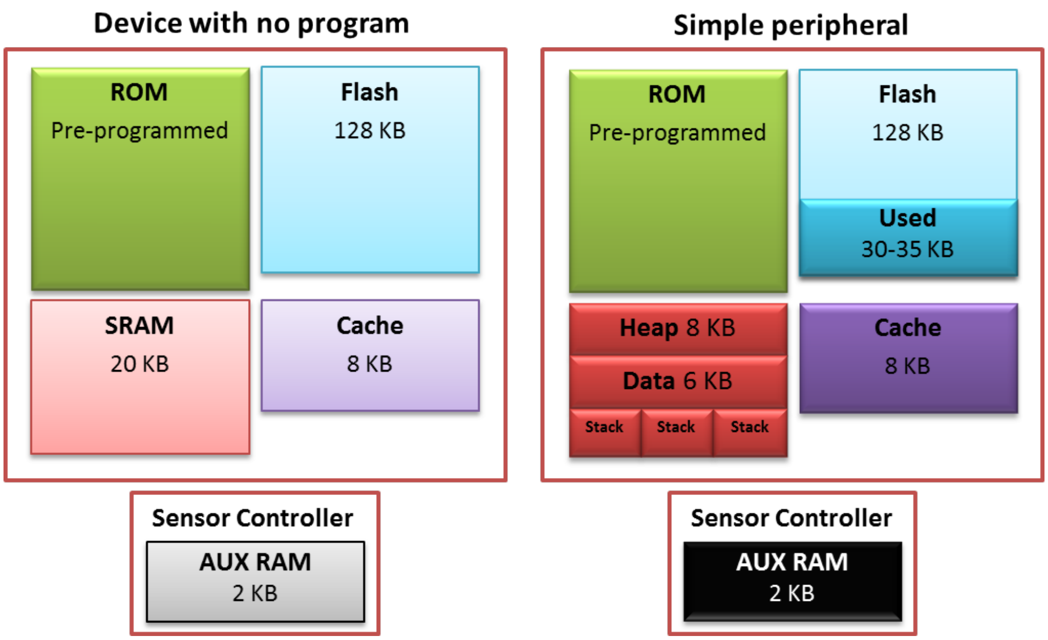

Figure 1. Memory Distribution in CC2640R2.

1.1 *Acronyms*

Table 1. Acronyms Used in This Document

Acronym	Description
AUX RAM	Auxiliary RAM
BLE	Bluetooth low energy
bss	Block started by symbol
GPRAM	General purpose RAM
KB	Kilobyte
MCU	Microcontroller unit
RAM	Random access memory
RF	Radio frequency
ROM	Read-only memory
ROV	RTOS object viewer
RTOS	Real time operating system
SRAM	Static RAM
ULL	Ultra low leakage

2 What is RAM?

RAM is a memory area that an application can use to store information at runtime. Flash memory, on the other hand, is commonly programmed before the application is initialized. An example of information your program might want to save at runtime is data from sensors. In addition to storing information runtime, the RAM is used to store initialized and uninitialized data (.data and .bss). The heap is also located in RAM. In CC2640R2 devices, the RAM is divided into static random access memory (SRAM) and a cache. The CC2640R2F has 20KB ultra low leakage (ULL) SRAM and an 8 KB cache. In addition, the Sensor Controller has a 2 KB RAM area called the AUX RAM.

Table 2 shows SRAM usage in the simple peripheral example project. .data, .bss and .stack are the largest sections occupying the SRAM. Note that specific numbers may vary depending on the application, the BLE Stack version and the compiler. Space allocated for the ICall heap is not shown as occupied. *Simple peripheral* uses the auto heap size functionality (see Section 5.2), thus the heap occupies roughly all the remaining space in the SRAM. Figure 2 is a screen shot of the memory allocation view in Code Composer Studio™ (CCS). It shows the largest objects occupying the SRAM. Again, the specific numbers will vary with application, compiler and BLE Stack version.

Table 2. Simple Peripheral SRAM Usage

Section	Description
ICall Heap (7.5 KB)	A heap that is used for dynamic memory allocation in the application and allocating message between the BLE stack task and the application task.
.vecs (0.2 KB)	Defined vectors.
.data (4.3 KB)	Initialized variables.
.bss (3.3 KB)	Uninitialized variables including task stacks and the TI-RTOS heap.
.stack (1 KB)	System stack. Used by TI-RTOS for *main()*, hardware interrupts and software interrupts.

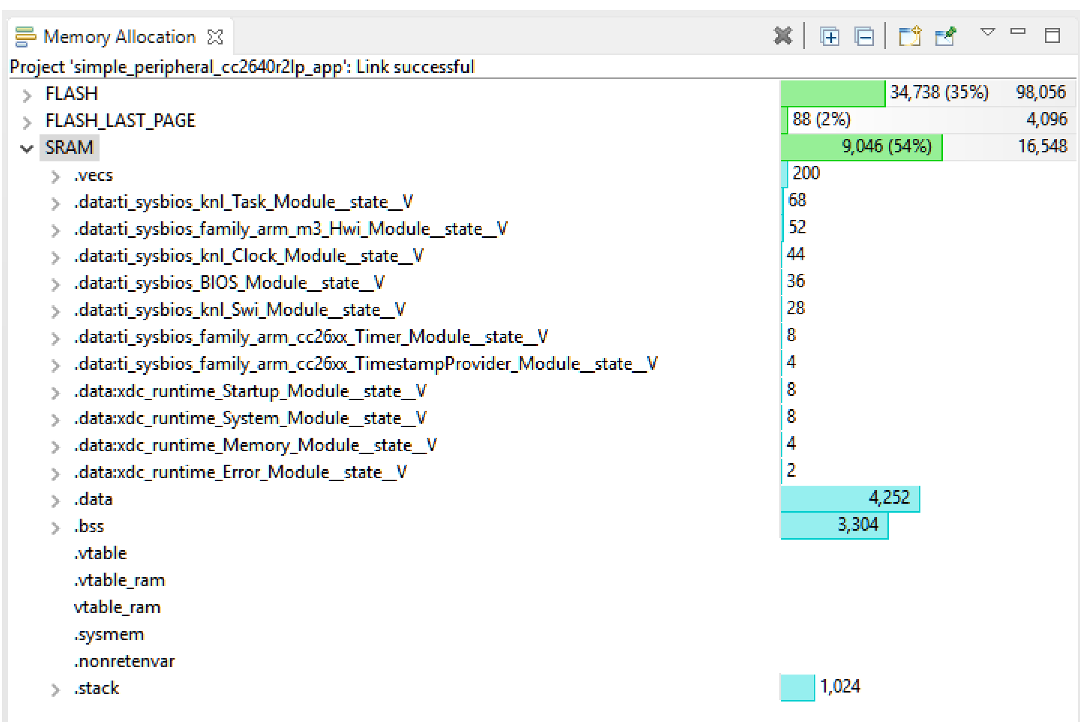

Figure 2. Simple Peripheral Memory Allocation View.

3 The Cache and GPRAM

3.1 What is the Cache?

The cache is an 8 KB section of the device's RAM reserved for the processor. The cache module temporarily stores data that has been read from the Flash memory, so that frequently used data does not have to be fetched from Flash on each access. This reduces the number of CPU wait-states and saves power. When the cache is not used, it is not powered. This is true for the Standby mode, and Idle modes where the cache is not in use.

On the CC2640R2F, the cache memory cells are is physically different from the SRAM. The SRAM is highly optimized for low current leakage in standby, while the cache is optimized for speed as well as low power consumption when active.

3.2 Using the Cache as RAM

The cache can be temporarily or permanently disabled. When the cache is disabled, the memory can be used as RAM.

If your application needs more memory, or if you need more space in SRAM, the cache can be re-purposed as RAM. This allows the linker [(1)] to store data in this section of the RAM. This section is referred to as the general-purpose RAM (GPRAM). This causes the program to run at a slightly reduced speed, and it increases the device power consumption in sleep. This is because the GPRAM has to be powered even when the device is sleeping in order to retain the data stored in GPRAM during sleep. The current consumption in stand-by mode with and without cache retained is listed in the *CC2640R2F SimpleLink™ Bluetooth® low energy Wireless MCU Data Sheet*.

When the program runs at a slightly decreased speed, the device to spends more time when active. This gives a higher overall power consumption. How this affects the device power consumption depends on the application. For some applications, the added power consumption is very small, but for processing intensive applications it is slightly higher. To verify your application current consumption, use the method described in *Measuring Bluetooth Low Energy Power Consumption*.

Alternatively the cache can be used as cache as normal, but when needed it can be switched to GPRAM to hold a temporary buffer. In this case the increased power consumption will be less, since the cache is used as normal and does not need to be retained in sleep (except when the cache is used as RAM.)

If you are ready to start using the cache as RAM, head over to the step-by-step instructions in the *Using the Cache as RAM* chapter of the *BLE-Stack User's Guide*.

4 Using the Sensor Controller AUX RAM

The Sensor Controller is an autonomous processor on the CC2640R2. The sensor controller is intended for interfacing external sensors and for collecting analog and digital data autonomously while the rest of the system is in sleep mode. You can read more about the Sensor Controller in the *CC13xx, CC26xx SimpleLink™ Wireless MCU Technical Reference Manual*. If your application is not using the Sensor Controller, its 2 KB RAM can be used for the main application. Please note that access to the Sensor Controller AUX RAM (auxiliary RAM) is a lot slower than access to other memory areas. This should be taken into consideration when deciding on what to store in the AUX RAM.

The *Using the AUX RAM as RAM* section of the *BLE-Stack User's Guide* contains instructions for using the AUX RAM.

(1) The linker is a tool that takes the fragmented, compiled code from the compiler, organizes it into one program and decides which part of the program should be stored where on the CC2640R2F device.

5 Manage Heap and Stack Sizes

5.1 Using ROV to Determine Peak Memory Usage

In TI-RTOS, each task has its own stack. In addition, there is a system stack that is used by RTOS for *main()*, hardware interrupts and software interrupts. On CC2640R2F, the system stack has a default size of 1 KB. However, this can be changed as described in the *System Stack* chapter in the *BLE-Stack User's Guide*.

The TI-RTOS object viewer (ROV) is a plug-in to both CCS and IAR with various functionality for debugging. To see how the ROV can be used to determine peak stack size, see the *TI-RTOS Object Viewer* chapter in the *BLE-Stack User's Guide*. Optimizing the stack size frees as much space in SRAM as possible. However, care must be taken to avoid stack overflow.

5.2 Manage the Heap Size

The ICall heap is used to allocate messages between the Bluetooth low energy stack task and the application task. It can also be used for dynamic memory allocation in the tasks. The ICall heap is implemented in the BLE Stack, so the following paragraph is not relevant for projects not imported from the BLE Stack (e.g. the *empty* project, which is imported from the *drivers* folder).

Most example applications in the BLE Stack use the heap auto size feature. This is enabled by the HEAPMGR_SIZE=0 define in the compiler predefined symbols. The heap auto size is an easy to use feature that automatically allocates memory for the heap based on the amount of available memory (memory that is not statically allocated). The auto heap size feature does not determine the amount of heap needed for the application, but this can be determined by using the heap manager metrics functionality. Using the heap manager metrics to configure the heap size is explained in the BLE-Stack User's Guide, under *Profiling the ICall Heap Manager (heapmgr.h)*.

6 References

1. *CC2640R2F SimpleLink™ Bluetooth® low energy Wireless MCU Data Sheet*
2. *Measuring Bluetooth Low Energy Power Consumption*
3. *BLE-Stack User's Guide*
4. *CC13xx, CC26xx SimpleLink™ Wireless MCU Technical Reference Manual*

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (March 2017) to A Revision	Page
• Added information for dynamic cache usage.	1

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265