AIROC™ CYW20819 and CYW20820 Application RAM

Defaults and optimization options

Abstract

The AIROC™ CYW20819 and CYW20820 devices provide 176 KB of read-write memory (RAM) that is shared by code in ROM, patches, libraries, and user applications. This document describes how the memory usage can be adjusted and optimized for specific use-cases.

The read-write memory is in two address ranges: 0x200000 – 0x228000 (160 KB) and 0x270000 – 0x274000 (16 KB). Of the 160 KB, 61 KB are used by ROM code directly. This usage includes variables and structures for code configuration and state, along with the buffer memory needed for data processing. Any parts of this 61 KB that can be modified requires patches to ROM code. After this portion is accounted for, about 115 KB of the 176 KB remains.

Some memory is also used for variables of patch code, leaving 114 KB. This is less than 1000 bytes and is expected to grow slowly as mandatory patches are introduced over time.

Given the default configuration of the **empty_wiced_bt** application , about 78 KB of the remaining 114 KB are used for dynamic allocations and application or library read-write memory, such as buffers and state variables. Note that application and library code and read-only memory are located in the on-chip flash memory and do not use RAM. This leaves about 36 KB of read-write memory for the application.

Table 1 RAM usage

RAM type	Size (KB)	Remaining from 176 KB	Comment
ROM use	61	115	
Patch use	1	114	
Block pools	31	83	Adjustable
Dynamic allocation	47	36	Adjustable
Application	Up to 36		

This document will describe the configuration settings available to manage the block pools and dynamic allocation areas, potentially freeing up more application memory.

Table of Contents

Abstract **1**
Table of Contents **2**
1 RAM usage overview **3**
1.1 Dynamic allocation 3
1.2 Block pools 3
2 Configurations for dynamic allocations **4**
2.1 Stack initialization 4
2.2 Memory "pre-init" API 4
3 Block pool configuration **5**
4 Tuning example **7**
5 Conclusion **9**
Revision history **10**

1 RAM usage overview

The CYW20819 RAM is used by code for read-write data storage, typically in initialized ".data" sections or zero initialized ".bss" sections. These types of sections are referenced by ROM, patch, library and application code. The boot process allocates and initializes these sections and they comprise the "ROM use", "Patch use" and "application" areas that are referenced in **Table 1**. Of these, only the application usage is under control of the application developer.

The additional memory areas "Block Pools" and "Dynamic allocation" are configurable, but the settings will affect the performance depending on the application requirements for Bluetooth® and other functionality. Configuration settings are applied from a .cgs file, as described in **Block pool configuration**.

1.1 Dynamic allocation

As an embedded system that is designed for long-term operation, memory allocations made by the CYW20819 code are often permanent. The dynamic memory manager will reserve memory, but will not support freeing and merging of allocation fragments. Alternatively, memory designed for allocation, freeing, and re-use is supported by block pools of uniform sizes. These pools are created within dynamically allocated memory.

Dynamic allocation from the two physical RAM blocks, 0x200000 – 0x228000 (160 KB) and 0x270000 – 0x274000 (16 KB), are managed with pointers to the base of free memory and the top of free memory. The amount of free memory available for allocation is the difference between these pointers. The base pointer is adjusted upward for most allocations. Dynamic allocations are attempted from the 16 KB RAM block in the address range 0x270000 – 0x274000 first. When blocks are too large to be allocated from the free area in that range, then the free area in the RAM block ranging from 0x200000 – 0x228000 is used for allocation. Note that this latter RAM area is already partially occupied by ROM, patch, library, and application data. Applications allocate this memory using `wiced_memory_permanent_allocate()`.

By the time the application, libraries, and stack are initialized, the first free memory block (0x270000 – 0x274000) is fully allocated. The difference between the base and top pointers for the second free memory area, the 0x200000 – 0x228000 block, is reported by `wiced_memory_get_free_bytes()`.

1.2 Block pools

Several block or buffer pools are allocated by ROM and stack code. These can often be optimized, but take care to avoid performance degradation and adequate buffering for edge conditions.

General use pools are allocated and created according to the "Dynamic Memory Pool" configuration, which provides the number of pools and a corresponding array setting the block size and count for each pool. The default configuration can be overridden by the application. The General Use pools provide buffer pools to ROM, library, and application code.

Another set of buffer pools are used for buffer communication over the Host Controller Interface (HCI). Four specific pools are created per the "ACL Pool" configuration and "LE ACL Pool" configuration. This is described in **Block pool configuration**. Finally, a set of buffer pools is allocated according to stack configuration by `wiced_bt_stack_init()`. These are used for application-specific purposes. The pool configuration is an array passed to the `wiced_bt_stack_init()` function. Note that devices other than the CYW20819 may use a different stack version that does not allocate application buffers with this configuration.

2 Configurations for dynamic allocations

Dynamic memory allocations are made during system boot-up as well as initialization of the stack. These can be controlled by direct stack configuration, use of the `wiced_memory_pre_init()` API, and a few other settings requiring ROM symbols.

2.1 Stack initialization

Some allocations are derived from the `wiced_bt_cfg_settings_t` parameter in the `wiced_bt_stack_init()` call. The configuration structure passed in the call is defined in *wiced_bt_cfg.h* and initialized for this example in *app_bt_cfg.c*. This data structure configures for various Bluetooth® protocols such as gatt (GATT, RFCOMM, L2CAP, AVDT, AVRC, etc), Low Energy scan, and advertising characteristics, along with more general settings like the number of simultaneous links and maximum size of the filter accept list.

Tuning these settings to cover the minimum number of channels, links, and protocols necessary for the application use cases will also minimize the size of the data structures to be dynamically allocated.

2.2 Memory "pre-init" API

An API is provided to configure several parameters used for allocation configuration during CYW20819 boot. Exercise caution when tuning these settings as they may affect performance.

```
typedef struct tag_mem_pre_init_control
{
    uint8_t max_ble_connections;
    uint8_t max_peripheral_piconet; /* use to reduce bt connections */
    uint8_t max_resolving_list;
    uint8_t onfound_list_len;
    uint8_t max_multi_adv_instances;
} WICED_MEM_PRE_INIT_CONTROL;

/**
 * set pre-init memory allocation parameters
 * call this from spar_crt_init function to set parameters prior to allocations
 */
void wiced_memory_pre_init(WICED_MEM_PRE_INIT_CONTROL *mem_pre_init);
```

These settings will affect allocations as follows:

Parameter	Default	Approximate allocation per count
max_ble_connections	8	1000
max_peripheral_piconet	4	1150
max_resolving_list	20	80
onfound_list_len	20	148
max_multi_adv_instances	8	232

To override the weakly defined default value from *spar_setup.c*, define the structure "`WICED_MEM_PRE_INIT_CONTROL g_mem_pre_init`" in the application code with desired values.

3 Block pool configuration

The General Use block pools are created according to the `g_foundation_config_DynamicMemory` configuration data structure. The pools are arranged as per the default value of `g_foundation_config_DynamicMemory` for the empty_wiced_bt code example. The total allocated memory for the General Use block pools for this setting is 7232 bytes.

Non-default configuration settings can be made by adding a configuration command to the application combined *.cgs file. For the CYW20819 case, this can be done by appending to the *mtb_shared\wiced_btsdk\dev-kit\baselib\20819A1\<ReleaseVersion>\COMPONENT_20819A1\platforms\CYW208XXA1.cgs* file.

```
ENTRY "Dynamic Memory Pool"
{
    "Number of pools" = 4
    "Size[0]" = 16
    "Count[0]" = 32
    "Die reserve[0]" = 3
    "Size[1]" = 48
    "Count[1]" = 36
    "Die reserve[1]" = 2
    "Size[2]" = 96
    "Count[2]" = 20
    "Die reserve[2]" = 1
    "Size[3]" = 268
    "Count[3]" = 10
    "Die reserve[3]" = 0
}
```

Table 2 General buffer pools

Name	Block size	Block count	Allocation
GEN0	16	32	640
GEN1	48	36	1872
GEN2	96	20	2000
GEN3	268	10	2720

In addition, four specific buffer pools for HCI communication are created as per the following configurations.

The configuration value for ACL Pools is set in the platform cgs file and can be modified there:

```
ENTRY "ACL Pool Configuration"
{
    "Host claim ACL Host to device size"   = 1021
    "ACL Host to device size"              = 1092
    "Unemployed entry"                     = 0
    "ACL Device to host size"              = 1068
    "Host Claim ACL Host to device count"  = 6
    "ACL Host to device count"             = 6
    "ACL Device to host count"             = 4
}
```

Block pool configuration

For LE Pools, the configuration is:

```
ENTRY "BLE ACL Pool Configuration"
{
    "BLE Host claim ACL Host to device size"    = 251
    "BLE ACL Host to device size"               = 264
    "BLE device claim ACL device to host size"  = 251
    "BLE ACL Device to host size"               = 264
    "BLE Host Claim ACL Host to device count"   = 6
    "BLE ACL Host to device count"              = 6
    "BLE ACL Device to host count"              = 8
}
```

This sets up the buffer pools as shown in **Table 3**. The total allocated is 14616 bytes.

Table 3 Specific use buffer pools

Name	Block size	Block count	Allocation
AD2H	1068	4	4288
AH2D	1092	6	6576
BD2H	264	8	2144
BH2D	264	6	1608

These pools can be monitored with `wiced_bt_get_buffer_usage()` as documented in the **AIROC™ memory API**.

Application-related pools are configured during `wiced_bt_stack_init()`. The size and number of the buffers in each pool are specified by the array passed to the `wiced_bt_cfg_buf_pool_t` argument.

```
const wiced_bt_cfg_buf_pool_t wiced_bt_cfg_buf_pools[WICED_BT_CFG_NUM_BUF_POOLS] =
{
 {64,  12}, /* Small Buffer Pool */
 {360, 6 }, /* Medium Buffer Pool (used for HCI & RFCOMM control messages, min recommended
size is 360) */
 {1056, 6}, /* Large Buffer Pool  (used for HCI ACL messages) */
 {1056, 0}, /* Extra Large Buffer Pool - Used for AVDT media packets and misc */
};
```

Table 4 Application buffer pools

Name	Block size	Block count	Alloc size	Use
gki 0	64	12	912	Small buffers
gki 1	360	6	2232	HCI & RFCOMM control messages
gki 2	1056	6	6408	HCI ACL messages
gki 3	1056	0	0	Used for AVDT media packets and miscellaneous

The pools can be tuned based on the application. The default case shown in **Table 4** allocates 9552 bytes for buffers. See **AIROC Application Buffer Pools** for more information.

The three types of buffer pools for this example code occupy a total of 31400 bytes.

4 Tuning example

The read-write memory available for the empty_wiced_bt application with default settings is about 36 KB. Using the configuration settings from this document and some assumptions about the application that will be created from the empty_wiced_bt code example, we can increase this memory. Note that the settings below are only examples and realistic choices for optimization depend on the application's desired function, performance, and safety margins.

For example, if we desire to use Bluetooth® Low Energy only, then the classic Bluetooth® data structures can be minimized. We can also override the default settings for `wiced_memory_pre_init()` as described in section **2.2** in the following manner:

```
WICED_MEM_PRE_INIT_CONTROL g_mem_pre_init =
{
    4,
    3,
    12,
    0,
    12
};
```

Adding this change increases the read-write memory available for the application to 44 KB.

The ACL and LE host communication buffer pools can be reduced along with the general buffer pool availability by editing *platforms/CYW208XXA1.cgs* as follows.

```
ENTRY "ACL Pool Configuration"
{
    "Host claim ACL Host to device size"   = 1021
    "ACL Host to device size"              = 264
    "Unemployed entry"                     = 0
    "ACL Device to host size"              = 512
    "Host Claim ACL Host to device count"  = 6
    "ACL Host to device count"             = 1
    "ACL Device to host count"             = 3
}
ENTRY "BLE ACL Pool Configuration"
{
    "BLE Host claim ACL Host to device size"    = 251
    "BLE ACL Host to device size"               = 264
    "BLE device claim ACL device to host size"  = 251
    "BLE ACL Device to host size"               = 264
    "BLE Host Claim ACL Host to device count"   = 2
    "BLE ACL Host to device count"              = 2
    "BLE ACL Device to host count"              = 8
}
```

Increasing the read-write memory to about 54 KB.

```
ENTRY "Dynamic Memory Pool"
{
    "Number of pools" = 4
    "Size[0]" = 16
    "Count[0]" = 32
    "Die reserve[0]" = 3
    "Size[1]" = 48
    "Count[1]" = 36
    "Die reserve[1]" = 2
    "Size[2]" = 96
    "Count[2]" = 20
    "Die reserve[2]" = 1
    "Size[3]" = 268
    "Count[3]" = 2
    "Die reserve[3]" = 0
}
```

Tuning example

Increasing the available read-write memory to 56 KB.

Next, focus on the settings passed to the stack-affecting allocations. Reduce the application buffer pools by modifying the `wiced_bt_cfg_buf_pools[]`.

```
const wiced_bt_cfg_buf_pool_t wiced_bt_cfg_buf_pools[WICED_BT_CFG_NUM_BUF_POOLS] =
{
 {64,   10}, /* Small Buffer Pool */
 {360, 6 }, /* Medium Buffer Pool (used for HCI & RFCOMM control messages, min recommended
size is 360) */
 {1056, 2}, /* Large Buffer Pool  (used for HCI ACL messages) */
 {1056, 0}, /* Extra Large Buffer Pool - Used for AVDT media packets and misc */
};
```

Increasing the available RAM to 58 KB.

Finally, try to optimize the configuration settings in `wiced_bt_cfg_settings`. Most of these settings are already at a minimum in the empty_wiced_bt code example. For example, reduce the `max_simultaneous_links` and `addr_resolution_db_size`.

```
const wiced_bt_cfg_settings_t wiced_bt_cfg_settings =
{
    .device_name                        = (uint8_t*)app_gap_device_name,
    .device_class                       = {0x00, 0x00, 0x00},
    .security_requirement_mask          = BTM_SEC_NONE,
    .max_simultaneous_links             = 2,
…
    .addr_resolution_db_size           = 4,
    .max_number_of_buffer_pools        = 4,
    .rpa_refresh_timeout               = WICED_BT_CFG_DEFAULT_RANDOM_ADDRESS_CHANGE_TIMEOUT,
    .ble_filter_accept_list_size       = 0,
    .default_ble_power_level           = 12
};
```

This increases the available RAM by 296 bytes to 58 KB.

5 Conclusion

Several techniques for read-write memory optimization were demonstrated using the empty_wiced_bt code example. By changing the configurations for buffer pools and dynamic allocations the read-write memory was increased from the default level of 36 KB to 58 KB.

Revision history

Document version	Date of release	Description of changes
**	2022-11-07	New user guide.
*A	2022-11-10	Updated header and fixed typos.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Go to www.infineon.com/support

Document reference

002-36601 Rev.*A

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.