AIROC™ CYW20xxx and CYW43012 Application RAM

Defaults and optimization options

Abstract

The AIROC™ CYW20706, CYW20719, CYW20721, CYW20819, CYW20820, CYW20835, and CYW43012 devices provide read-write memory (RAM) that is shared by code in the ROM, patches, libraries, and user applications. This document describes how the memory usage can be adjusted and optimized for specific use cases, optimizing the RAM available for user applications.

This document describes the configuration settings available to manage the block pools and dynamic allocation areas for the devices listed above, potentially freeing up more application memory. The CYW20819 device and the empty_wiced_bt code example (CE) are used as an example to demonstrate the configuration changes and results.

Table of contents

1 Preliminaries

Throughout this document, the AIROC™ Bluetooth® System-on-Chip device set listed above will be referred to as “devices”.

Note that this document applies to BTSTACK version 1 devices. The stack version can be found in the BSP Makefile, for example, `COMPONENTS+=btstack_v1`. Other devices may use a different stack version that may not allocate buffers as described in this document.

2 RAM usage overview

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

For example, the CYW20819 device provides 176 KB of RAM. This read-write memory is in two address ranges: 0x200000 – 0x228000 (160 KB) and 0x270000 – 0x274000 (16 KB). Of the 160 KB, 61 KB is used by ROM code directly. This usage includes variables and structures for code configuration and state, along with the buffer memory required for data processing. Any parts of this 61 KB that can be modified requires patches to the ROM code. After this portion is accounted for, approximately 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, approximately 78 KB of the remaining 114 KB is used for dynamic allocations and application or library read-write memory, such as buffers and state variables. Note that for the CYW20819 device, most of the application and library code and read-only memory are located in the on-chip flash memory and do not use the RAM. This leaves approximately 36 KB of read-write memory for the application.

Table 1 RAM usage for CYW20819

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		

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.

Physical RAM blocks are each device are listed in Table 2. Note that CYW20819 and CYW20820 include 256 KB of on-chip flash used for code and read-only data.

Table 2 RAM usage for the devices

Device	Range 1	Range 2	Total
CYW20706	0x200000 - 0x248000 (288 KB)	0xd0000 – 0xe0000 (64 KB)	352 KB
CYW20719	0x200000 - 0x270000 (448 KB)	0x270000 – 0x280000 (64 KB)	512 KB
CYW20721	0x200000 - 0x270000 (448 KB)	0x270000 – 0x280000 (64 KB)	512 KB
CYW20819	0x200000 – 0x228000 (160 KB)	0x270000 – 0x274000 (16 KB)	176 KB

Device	Range 1	Range 2	Total
CYW20820	0x200000 – 0x228000 (160 KB)	0x270000 – 0x274000 (16 KB)	176 KB
CYW20835	0x200000 - 0x250000 (320 KB)	0x270000 – 0x280000 (64 KB)	384 KB
CYW43012	0x200000 – 0x251000 (324 KB)	0x130000 - 0x140000 (64 KB)	388 KB

2.1 Dynamic allocation

Because an embedded system that is designed for long-term operation, memory allocations 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 reuse is supported by pools of uniform size blocks. These pools are created within the dynamically allocated memory.

Dynamic allocations from the physical RAM are managed with pointers to the base of the free memory and the top of the 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. Note that some RAM area is already occupied by the ROM, patch, library, and application data. Applications allocate this memory using `wiced_memory_permanent_allocate()`.

The difference between the base and top pointers for the free memory area is reported by `wiced_memory_get_free_bytes()`.

2.2 Block pools

Several block or buffer pools are allocated by the ROM and stack code. These can often be optimized, but be sure to avoid performance degradation and adequate buffering for edge conditions. For example, processing communications may involve bursts of activity that fill many storage buffers in a pool. The process that uses the data in the pools could have a latency that depends on other higher priority tasks. If a pool is depleted during such a burst, the communication data may not have a storage location and could be lost. The pool sizes should be determined to withstand such cases of high use and latencies.

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 the ROM, library, and application code.

Another set of buffer pools is 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—see Block pool configuration. Finally, a set of application-specific buffer pools is allocated according to the stack configuration by `wiced_bt_stack_init()`. The pool configuration is an array passed to the `wiced_bt_stack_init()` function.

3 Configurations for dynamic allocations

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

3.1 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 the empty_wiced_bt CE in *app_bt_cfg.c*. This data structure is configured for various Bluetooth® protocols (such as GATT, RFCOMM, L2CAP, AVDT, and AVRC), Low Energy scan, and advertising characteristics along with more general settings such as 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.

3.2 Memory "pre-init" API

An API is provided to configure several parameters used for allocation configuration during device boot. This API is declared and documented in the *wiced_memory_pre_init.h* header file. Note that this API has some variations between the devices. CYW20719, CYW20721, CYW20819, CYW20820, and CYW20835 have the same API, while APIs for CYW20706 and CYW43012 are slightly different. Exercise caution when tuning these settings because they may affect performance.

Continuing with the CYW20819 example:

```
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 the allocations as follows:

Table 3 WICED_MEM_PRE_INIT_CONTROL parameter information

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 linked default value from *spar_setup.c*, define the `WICED_MEM_PRE_INIT_CONTROL g_mem_pre_init` structure in the application code with the desired values.

CYW20706 has a similar style API, but the data structure members are different:

```
typedef struct tag_mem_pre_init_control
{
    UINT16 scanRssiThresholdDeviceListSize;
    UINT16 lm_cmdQueueAreaSize;
    UINT16  aclDownBufSize;
    UINT16  aclUpBufSize;
    UINT8   aclDownCount;
    UINT8   aclUpCount;
    UINT8   rmulpMaxLLConnection;
    UINT8   ulp_rl_maxSize;
} WICED_MEM_PRE_INIT_CONTROL;

void wiced_memory_pre_init(WICED_MEM_PRE_INIT_CONTROL *mem_pre_init);
```

Similar to the previously mentioned device API, the default values are set in a weakly linked structure. To override the default values, declare and initialize a non-weak structure in the application code.

Table 4 WICED_MEM_PRE_INIT_CONTROL parameter information

Parameter	Default	Approximate allocation per count
scanRssiThresholdDeviceListSize	25	7
lm_cmdQueueAreaSize	800	This is currently not modified.
aclDownBufSize	1092	1 * aclDownCount
aclUpBufSize	1064	1 * aclUpCount
aclDownCount	8	1092

Parameter	Default	Approximate allocation per count
aclUpCount	8	1064
rmulpMaxLLConnection	16	784
ulp_rl_maxSize	128	68

The CYW43012 API has only two parameters. The Resolving List (max number set by `num_ble_rl`) is used by the link layer to resolve Resolvable Private Addresses (RPA) used by advertisers, scanners, or initiators. The early initialization in *spar_setup.c* passes the weakly defined global variables as default parameters. Redeclare these variables in the application code as non-weak and initialize them to use values other than the default.

```
uint8_t g_wiced_memory_pre_init_enable __attribute__((weak)) = 0;
uint8_t g_wiced_memory_pre_init_max_ble_connections __attribute__((weak)) = 0;
uint8_t g_wiced_memory_pre_init_num_ble_rl __attribute__((weak)) = 0;


void wiced_memory_pre_init(uint32_t enable, /* Enable or disable this memory
tuning */
                           uint32_t max_ble_connections, /* can tune from 1 to
15 */
                           uint32_t num_ble_rl); /* default to 128, cat set
from 1 to 128 */
```

When `enable == 1`, the general buffer pools are modified from the default values:

```
const FOUNDATION_CONFIG_DYNAMIC_MEMORY_t _foundation_config_DynamicMemory
__attribute__((weak)) =
{
    /* .num_pools */ 5,
    {
        { /* .size */ 16,  /* .count */ 32, /* .die_reserve */ 3 },     /*
WAS 8/128 */
        { /* .size */ 32,  /* .count */ 36, /* .die_reserve */ 2 },     /*
WAS 32/32 */
        { /* .size */ 96,  /* .count */ 8,  /* .die_reserve */ 1 },     /*
WAS 96/55 */
        { /* .size */ 268, /* .count */ 8,  /* .die_reserve */ 1 },     /*
WAS 268/12 */
        { /* .size */ 572, /* .count */ 2,  /* .die_reserve */ 0 },     /*
WAS 512/1 */
    }
};
```

Table 5 WICED_MEM_PRE_INIT_CONTROL parameter information

Parameter	Default	Approximate allocation per count
`g_wiced_memory_pre_init_enable`	0	Set to 1, modifies pools to save 25724
`g_wiced_memory_pre_init_max_ble_connections`	15	997
`g_wiced_memory_pre_init_num_ble_rl`	128	68

4 Block pool configuration

4.1 Block pools for CYW20819

CYW20819 has general use block pools which 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 for CYW20819. The total allocated memory for the general use block pools for this setting is 7232 bytes. Devices except CYW20706 have a block pool configuration similar to CYW20819.

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 6 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
}
```

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 7. The total allocated is 14616 bytes.

Table 7 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 8 Application buffer pools

Name	Block size	Block count	Alloc size	Use
gki 0	64	12	912	Small buffers
gki 1	360	6	2232	HCI and 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 8 allocates 9552 bytes for buffers. See AIROC™ application buffer pools for more information.

The three types of buffer pools for this CYW20819 example code occupy a total of 31400 bytes.

4.2 CYW20706 block pools

CYW20706 is an older device with a configuration of block pools different from other devices. The general use pools are configured as shown in Table 9.

The general buffer pool is configured by the number of buffers. The size for each pool is fixed. An example of the configuration command, where the address is a fixed location and the pool buffer counts are 4-byte unsigned integers, here {48, 16, 10}:

```
ENTRY "Data" = "MM_NUM_OF_BLOCKS"
{
    "Address" = 0x00201984 # <<< 0x00201984 # <<< 0x00201984 # <<<
$AUTOGEN(ADDR{mm_numOfBlocks})
       "Data" =
        COMMENTED_BYTES
        {
            <hex>
                30 00 00 00   # MM_BLOCK_CATEGORY_0_COUNT
                10 00 00 00   # MM_BLOCK_CATEGORY_1_COUNT
                0a 00 00 00   # MM_BLOCK_CATEGORY_2_COUNT
        } END_COMMENTED_BYTES
}
```

Table 9 General buffer pool

Name	Block size	Block count	Allocation
GEN0	32	48	1728
GEN1	64	22	1496
GEN2	264	10	2680

The specific use buffer pools are shown in Table 10. They can be configured similar to the specific use buffer pools in Block pools for CYW20819.

```
ENTRY "ACL Pool Configuration"
{
    "Host claim ACL down payload size"  = 1021
    "ACL down buffer size"              = 1092
    "ACL up buffer size"                = 1064
    "Host claim ACL down payload count" = 6
    "ACL down buffer count"             = 8
    "Preserve UHE memory area"          = 1
    "ACL up buffer count"               = 8
}

ENTRY "BLE ACL Pool Configuration"
{
    "BLE Host claim ACL down payload size" = 251
    "BLE ACL down buffer size" = 260
    "BLE ACL up buffer size" = 264
    "BLE Host claim ACL down payload count" = 15
    "BLE ACL down buffer count" = 15
    "BLE ACL up buffer count" = 15
}
```

Table 10 Specific use buffer pool

Name	Block size	Block count	Allocation
mm_aclUpTcb	1068	1	1068
mm_aclDownTcb	1092	2	2192
mmulp_aclDownTcb	260	15	3960
uart_trans_pool	1032	2	2072
trans_pool	1032	2	2072

Application buffers are shown in Table 11. They are configured by the application code similar to the description in Block pools for CYW20819.

Table 11 Application buffer pool

Name	Block size		Block count	Alloc size	Use
gki 0	68		42	3360	Small buffers
gki 1	140		20	3040	HCI & RFCOMM control messages
gki 2	392		10	4040	HCI ACL messages

Name	Block size		Block count	Alloc size	Use
gki 3	1024		4	4144	Used for AVDT media packets and miscellaneous

The pools can be tuned based on the application. The default case shown in Table 11 allocates 14584 bytes for buffers. See AIROC™ application buffer pools for more information.

The three types of buffer pools for this CYW20706 example code occupy a total of 31852 bytes.

5 Tuning example

The read-write memory available for the *empty_wiced_bt* application with default settings running on CYW20819 is approximately 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, you can increase this memory. Note that the following settings are only examples; realistic choices for optimization depend on the application's desired function, performance, and safety margins.

For example, if you want to use Bluetooth® Low Energy only, the Bluetooth® Classic data structures can be minimized. You can also override the default settings for `wiced_memory_pre_init()` as described in Memory "pre-init" API 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
}
```

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 `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 `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,
```

Tuning example

```
    .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.

6 Conclusion

Several techniques for read-write memory optimization were demonstrated using the *empty_wiced_bt* code example for CYW20819. 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.

A similar API is provided for each of CYW20706, CYW20719, CYW20721, CYW20819, CYW20820, CYW20835, and CYW43012 devices.

Revision history

Document revision	Date	Description of changes
**	2022-11-07	New user guide.
*A	2022-11-10	Updated header and fixed typos.
*B	2023-03-09	Added support for other CYW20xxx and CYW43012 devices.

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

Edition 2023-03-09
Published by

Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
002-36601 Rev. *B

`Important notice
This document is for information purposes only and any information given herein shall in no event be regarded as a warranty, guarantee or description of any functionality, conditions and/or quality of our products or any suitability for a particular purpose with regard to the technical specifications of our products, we kindly ask you to refer to the relevant product data sheets provided by us. Our customers and their technical departments are required to evaluate the suitability of our products for the intended application.

We reserve the right to change this document and/or the information given herein at any time.

Warnings
Due to technical requirements, our 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 us in a written document signed by authorized representatives of Infineon Technologies, our products may not be used in any life endangering applications, including but not limited to medical, nuclear, military, life critical or any other applications where a failure of the product or any consequences of the use thereof can result in personal injury.