Qualcomm Technologies International, Ltd.

Configuring the Qualcomm BlueCore Technology Memory Allocator

Application Note

80-CT406-1 Rev. AK

October 25, 2017

Revision history

Revision	Date	Description
1	JUN 2009	Initial release. Alternative document number CS-00128085-AN
2	AUG 2009	Note existence of per-pool housekeeping overhead.
3	JUL 2010	Updated for 2010 SDKs and Qualcomm® BlueLab™ references removed.
4	JUL 2011	Updated to latest CSR™ style
5	JAN 2012	Updated to latest CSR style
6	DEC 2013	Updated to New CSR style
7	JUN 2016	Updated to conform to QTI standards; No technical content was changed in this document revision
8	APR 2017	Added notes about the VM slots restrictions
9	APR 2017	Editorial corrections
AK	OCT 2017	Added to Document Content Manager. DRN updated to use Agile number. No change to technical content.

Contents

Tables

1 BlueCore pool memory allocator - overview

From the Unified 23h release of Qualcomm® BlueCore™ technology firmware onwards, the BlueCore pool memory allocator (`pmalloc`) has been updated to make it easier for customers to alter the division of available memory to suit virtual machine (VM) applications.

NOTE The allocator should not be reconfigured for parts where a VM application is not in use.

2 The BlueCore pool memory allocator (pmalloc)

Each Qualcomm® BlueCore™ firmware build sets aside a fixed area of RAM for dynamic memory allocation. The size of this area is determined when the firmware is compiled.

This memory is managed by the firmware's `pmalloc` subsystem. `pmalloc` divides the available memory into a number of pools of varying sizes. It determines the sizes of the pools and the number of pools of a particular size at boot time, by reading persistent store keys (PS Keys).

At run time, when the firmware or VM application makes a request for a given number of words of memory, `pmalloc` allocates the smallest available pool of a size greater than or equal to the requested number of words.

For example, if `pmalloc` receives a request for 13 words of memory, and has pools of size 2, 4, 8, 10, 12, 16, and 20 words, it allocates a pool of the size 16, unless all such pools are taken, in which case it allocates one of the size 20 pools, assuming one is available. Hence the size of the largest pool determines the largest contiguous block of memory which `pmalloc` can allocate.

The PS Keys used to configure `pmalloc` default to settings that support operation with a large number of Bluetooth connections (typically 7, the maximum possible number of slaves for a given master). These default settings create a number of pools sized for the firmware's own per link data structures, but do not typically create many larger pools.

VM applications sometimes require larger contiguous blocks of memory for their own use. 3 and 4 describe how to configure PS Keys and reduce the number of supported Bluetooth connections to make space for larger pools.

3 PS Keys used to configure pmalloc

Prior to firmware release 23h, `pmalloc` was configured with PSKEY_PMALLOC_SIZES, which specified all the pools to be created by `pmalloc`.

This key was not designed to be changed by customers, except on advice from QTIL.

Some pools specified by the key were sized to fit critical structures in the firmware, but no externally visible information revealed which pools these were. Customers could not know which pools it was safe to eliminate to make way for larger pools.

Even altered versions of the key supplied by QTIL were specific to a particular firmware build, and were not generally suitable for copying for use with different firmware builds.

Nevertheless, bug reports received by QTIL showed this key was being altered by customers and copied between firmware builds, in order to free up larger pools for VM applications. To prevent the continued inappropriate use of PSKEY_PMALLOC_SIZES, the new system described in this document has been implemented in 23h firmware.

PSKEY_PMALLOC_SIZES has now been retired. Starting with firmware release 23h, it will be ignored by BlueCore firmware.

3.1 PS Key used to configure pmalloc from Unified 23h onwards

From 23h onwards the division of the available RAM into pools is specified in the following PS Keys:

- PSKEY_PMALLOC_BASE: The pools required for basic operation of the firmware, specified by QTIL.
- PSKEY_PMALLOC_PER_ACL: The pools required for a single ACL link, specified by QTIL.
- PSKEY_PMALLOC_PER_SCO: The pools required for a single SCO link, specified by QTIL.
- PSKEY_PMALLOC_APP (when the firmware supports VM applications): The pools required by the VM application, specified by the application author. The default for this key is empty.
- PSKEY_PMALLOC_EXTRA: Optional pools to fill the remaining RAM space, specified by QTIL.

Each of these keys is a list of (pool size, number of pools of this size) pairs. For example, specifying:

```
PSKEY_PMALLOC_BASE = 2 10 4 20 10 30
```

This example entry creates 10 pools of size 2, 20 pools of size 4 and 30 pools of size 10. `pmalloc` creates multiple instances of the pools specified by PSKEY_PMALLOC_PER_ACL and PSKEY_PMALLOC_PER_SCO, according to the number of supported ACL and SCO links, see Configuration of pmalloc at boot time.

The following restrictions apply to these keys:

- The pool sizes must be even. If an odd sized pool is specified, the firmware rounds up the pool size to the next even value and emits `FAULT_PMALLOC_ODD_POOL_SIZE`. See Notes and warnings when using pmalloc for guidance on how to see any faults emitted.
- The pool size and number of pools must not be zero. If a zero value is specified, the firmware emits `FAULT_PMALLOC_INVALID_POOL_ENTRY` and ignores that pair of values.
- The pool size must not exceed 1024 words. If a larger pool is specified, the firmware emits `FAULT_PMALLOC_INVALID_POOL_ENTRY` and ignores that pair of values.
- The PS Keys must specify pairs of values, so the overall length of each PS Key must be even. If the length of the key is odd, the firmware emits `FAULT_PMALLOC_INVALID_PSKEY_LENGTH` and ignores the PS Key.

For the numerical values of fault codes, see pmalloc panic and fault codes.

3.2 Configuration of pmalloc at boot time

At boot time, `pmalloc` creates the pools as follows:

- pmalloc attempts to create the pools specified in PSKEY_PMALLOC_BASE. If there is insufficient space for this, it emits `FAULT_PMALLOC_BASE_INSUFFICIENT_SPACE` and falls back to the value of PSKEY_PMALLOC_BASE in the firmware's PS Key defaults (that is, the psrom store described in *BCCMD Commands*).
- pmalloc reads PSKEY_MAX_ACLS to determine the number of supported ACL links. It creates an instance of the set of pools specified in PSKEY_PMALLOC_PER_ACL for each supported ACL link. If there is insufficient space for this, it emits `FAULT_PMALLOC_ACL_INSUFFICIENT_SPACE`. `pmalloc` only creates whole instances of PSKEY_PMALLOC_PER_ACL, that is either all the pools specified in an instance are created, or none.
- pmalloc reads PSKEY_MAX_SCOS to determine the number of supported SCO links. It repeats the process for PSKEY_PMALLOC_PER_ACL described above, but using PSKEY_PMALLOC_PER_SCO, and emits `FAULT_PMALLOC_SCO_INSUFFICIENT_SPACE` if it runs out of space.
- If the firmware supports VM applications, `pmalloc` attempts to create the pools specified by PSKEY_PMALLOC_APP. It creates the pools in order from the first (number, size) pair in the PS Key to the last. If at any point there is insufficient space for the remaining pools, `pmalloc` emits `FAULT_PMALLOC_APP_INSUFFICIENT_SPACE` and stops processing the PS Key.
- `pmalloc` attempts to create the pools specified in PSKEY_PMALLOC_EXTRA. It creates the pools in order from the first (number, size) pair in the PS Key to the last. If at any point there is insufficient space for the remaining pools, `pmalloc` stops processing but does not emit a fault.
- If there is any remaining space, `pmalloc` fills it with whole or partial instances of PSKEY_PMALLOC_PER_ACL, but does not emit any faults to indicate insufficient space.

  NOTE This behavior is not expected to be relevant in practice, and should not be relied upon.

4 How to configure larger pmalloc pools

Of the PS Keys used to configure pmalloc, only PSKEY_PMALLOC_APP, PSKEY_MAX_SCOS and PSKEY_MAX_ACLS may be altered without consulting QTIL.

The other PSKEY_PMALLOC keys exist mainly to make their values visible to customers. This is to allow customers to deduce the amount of memory available for use in PSKEY_PMALLOC_APP.

The values of the PSKEY_PMALLOC keys are calculated at firmware build time, so may differ between firmware builds. To obtain the values used in a particular firmware build follow the guidance in How to obtain the values of pmalloc PS Keys.

4.1 How to obtain the values of pmalloc PS Keys

To obtain the values of PS Keys, customers may use the programs supplied as part of Qualcomm® BlueSuite™ (v2.2 or later), or any other tool capable of issuing the appropriate PS BCCMDs (refer to the *HQ and BCCMD Protocols and Commands*).

NOTE Ensure that the firmware is present on BlueCore by flashing it (using the BlueFlash tool or DFU).

4.1.1 How to use PSTool to obtain PS Key values

Connect to BlueCore with **PSTool**. Select **Programmer ID's** from the **View** menu. Enter the PS Key name into the **Filter:** field, leaving out the PSKEY_ prefix (for example,. `PMALLOC_PER_ACL`). Click on the required key. The value of the key appears as a hexadecimal number on the right hand side of the window.

4.1.2 How to use BTCli to obtain PS Key values

Connect to BlueCore with **BTCli**. To obtain the value of a PS Key, type `psget` followed by the name of the key, leaving out the PSKEY_ prefix. The name must be in lower case. For example:

```
psget pmalloc_base
```

BTCli responds with something like:

```
bccmd_getresp sn:0x0002 ps ok pmalloc_base d len:0x0018 0x0002 0x002d
0x0004 0x0055 0x0006 0x0028 0x0008 0x004b 0x000a 0x002d 0x0010 0x000f
0x0014 0x0028 0x0020 0x0014 0x0024 0x000c 0x0046 0x000a 0x0062 0x0001
0x0070 0x0001
```

After the key length, `len:0x0018`, the key values are shown in hexadecimal.

4.2 How to use the memory in PSKEY_PMALLOC_EXTRA

When the memory used by the keys in a particular firmware build has been established, some of this memory can be claimed for use in PSKEY_PMALLOC_APP.

This may be done by taking memory from PSKEY_PMALLOC_EXTRA, or by reducing the maximum number of ACL or SCO links, taking memory from PSKEY_PMALLOC_PER_ACL or PSKEY_PMALLOC_PER_SCO, respectively.

NOTE `pmalloc` does not emit a fault if there is insufficient space to create the pools specified in PSKEY_PMALLOC_EXTRA. This is because some or all of the memory used by PSKEY_PMALLOC_EXTRA may be taken by PSKEY_MALLOC_APP without creating an error condition.

To calculate the total number of words available, sum all of the pools specified by PSKEY_PMALLOC_EXTRA:

```
words = sum([(pool_size * n_pools) for (pool_size, n_pools) in key])
```

For example, if

```
PSKEY_PMALLOC_EXTRA = 2 10 4 20 10 30
```

there are (2 * 10) + (4 * 20) + (10 * 30) = 400 words available. These could be rolled into a single large pool:

```
PSKEY_PMALLOC_APP = 400 1
```

or two smaller pools:

```
PSKEY_PMALLOC_APP = 200 2
```

or any combination of sizes that adds up to less than or equal to the total number of words used by PSKEY_PMALLOC_EXTRA.

NOTE Customers need not alter PSKEY_PMALLOC_EXTRA, as PSKEY_PMALLOC_APP is processed before PSKEY_PMALLOC_EXTRA.

4.2.1 VM memory slots

The memory (allocated by the firmware in response to a malloc call from the VM), is accessed through slots. The number of slots is limited. Therefore, it is possible to have a malloc failure even when there are sufficient pools from which to allocate the requested memory. This is due to the number of used slots exceeding the maximum number available.

Alos each slot allows access to a limited amount of memory. If a request is made for more memory than can be accessed by a single slot, multiple slots are allocated. For example, in Unified 28a1, the slot size is 256 words. If a request is made for a block of 300 words, then two slots are allocated to this request.

NOTE The number of slots allocated is kept to a minimum, that is the number of slots is defined by the requested memory size. This does not depend on the size of the memory pool from which the memory is allocated. For example, if the slot size is 256 words and 300 words is allocated from a pool size of 1024, only 2 slots are used not the 4 that would be required for access to the full 1024 words of the memory pool.

4.3 How to use the memory in PSKEY_PMALLOC_PER_ACL or PSKEY_PMALLOC_PER_SCO

Configuration of pmalloc at boot time describes how `pmalloc` creates an instance of the set of pools specified by PSKEY_PMALLOC_PER_ACL for each ACL link specified in PSKEY_MAX_ACLS. Similarly, `pmalloc` creates an instance of the set of pools specified by PSKEY_PMALLOC_PER_SCO for each SCO link specified in PSKEY_MAX_SCOS.

By reducing PSKEY_MAX_ACLS or PSKEY_MAX_SCOS, the memory used by some of these instances can be added to PSKEY_PMALLOC_APP. The memory used by each link can be calculated by summing the pools specified by PSKEY_PER_ACL and PSKEY_PER_SCO keys, as described in How to use the memory in PSKEY_PMALLOC_EXTRA.

For example, if:

```
PSKEY_PMALLOC_PER_ACL = 2 10 4 20 10 30
```

Then there are 400 words used by each ACL link. Reducing PSKEY_MAX_ACLS by 1 would make those 400 words available, reducing it by 2 would make 800 words available, and so on. Similar reasoning applies for PSKEY_PMALLOC_PER_SCO and PSKEY_MAX_SCOS.

NOTE When calculating suitable values of PSKEY_MAX_ACLS and PSKEY_MAX_SCOS, allow for transient needs as well as the more obvious ones. Typically this requires 2 additional ACLs to allow for remote name requests or service discovery, and 1 additional SCO connection used during transfers. A headset configured with PSKEY_MAX_ACLS or PSKEY_MAX_SCOS set to 1 will not be usable.

Be aware the firmware rejects further SCO or ACL connections when it has the maximum number specified by PSKEY_MAX_SCOS or PSKEY_MAX_ACLS.

4.4 Notes and warnings when using pmalloc

- When assigning a value to PSKEY_PMALLOC_APP, customers must obey the restrictions described in PS Keys used to configure pmalloc.
- The memory pools specified by PSKEY_PMALLOC_APP are not reserved exclusively for the VM application. They join a common set of pools available from `pmalloc`. Therefore, it is possible for a pool specified by PSKEY_PMALLOC_APP to be taken by the firmware if no other pools are available.
- The pools specified by PSKEY_PMALLOC_PER_ACL and PSKEY_PMALLOC_PER_SCO are based on estimates of the space required by the Bluetooth baseband firmware.
  When higher layers of the Bluetooth stack are present in the firmware, these layers also require dynamically allocated memory, which they take from the common pools. Because of this, it is not advisable for PSKEY_PMALLOC_APP to use all the space apparently freed by reducing PSKEY_MAX_ACLS to a low value.
  The system is designed for customers to restrict the number of ACL links in return for some extra space.
- Some of the RAM available to be allocated by this allocator is used for housekeeping. The overhead consists of a few words for each distinct pool size in the configuration (the exact amount is firmware-dependent).
- Thus, adding a pool size that does not already exist in the QTIL-supplied configuration can apparently reduce the available RAM by a few words.

- If the firmware runs out of pools at run time, it is likely to panic and reset BlueCore. The most likely panic code is `PANIC_HEAP_EXHAUSTION`, although `PANIC_INVALID_PAGE_TABLE_LOCATION` may also be seen.
  When BlueCore has rebooted after a panic, the panic code can be retrieved using the `Panic_Arg` BCCMD (refer to the *HQ and BCCMD Protocols and Commands*). The numerical values of the relevant panic codes are listed in Table A-2.
- When making changes to these PS Keys, or when receiving new firmware, it is the customer's responsibility to test that their device still meets their requirements. QTIL tests the firmware with these keys set to their supplied values.
- When making changes to these PS Keys, customers should verify that the IC does not emit any faults on boot. This can be done using BTCli. On connecting to BlueCore, **BTCli** reports any fault events which have occurred since boot, for example:
  - `hardware_error hc:0x63`
  - `hq sn:0x000d fault ok fi:0x0063 0x5400 0x1`
- In this example, `BTCli` reports fault `0x63, FAULT_PMALLOC_INVALID_POOL_ENTRY`, using the HCI hardware error event (refer to the *Bluetooth Specification v3.0*) and the fault HQ command (refer to the *HQ and BCCMD Protocols and Commands*).

NOTE In some versions of xIDE, faults emitted by the firmware also appear in the **Messages** tab.

A pmalloc panic and fault codes

Table A-1 Fault codes

Numerical Value	Fault
0x5e	FAULT_PMALLOC_BASE_INSUFFICIENT_SPACE
0x5f	FAULT_PMALLOC_ACL_INSUFFICIENT_SPACE
0x60	FAULT_PMALLOC_SCO_INSUFFICIENT_SPACE
0x61	FAULT_PMALLOC_APP_INSUFFICIENT_SPACE
0x62	FAULT_PMALLOC_INVALID_PSKEY_LENGTH
0x63	FAULT_PMALLOC_INVALID_POOL_ENTRY
0x64	FAULT_PMALLOC_ODD_POOL_SIZE

See PS Key used to configure pmalloc from Unified 23h onwards for the circumstances under which faults are emitted.

Table A-2 Panic codes

Numerical Value	Panic
0x33	PANIC_HEAP_EXHAUSTION
0x35	PANIC_INVALID_PAGE_TABLE_LOCATION

Document references

Document	Reference
HQ and BCCMD Protocols and Commands	80-CT714-1/CS-00227432-SP
Bluetooth Specification v3.0	Bluetooth Core Version 3.0 + HS, 21 April 2009.

Terms and definitions

Term	Definition
ACL	Asynchronous ConnectionLess
BCCMD	BlueCore Command
BlueCore	Group term for the range of QTIL Bluetooth wireless technology ICs
Bluetooth	Set of technologies providing audio and data transfer over short-range radio connections
DFU	Device Firmware Upgrade
HCI	Host Controller Interface
HQ	Host Query
IC	Integrated Circuit
ID	Identifier
PS	Persistent Store
QTIL	Qualcomm Technologies International, Ltd.
RAM	Random Access Memory
SCO	Synchronous Connection-Oriented
VM	Virtual Machine