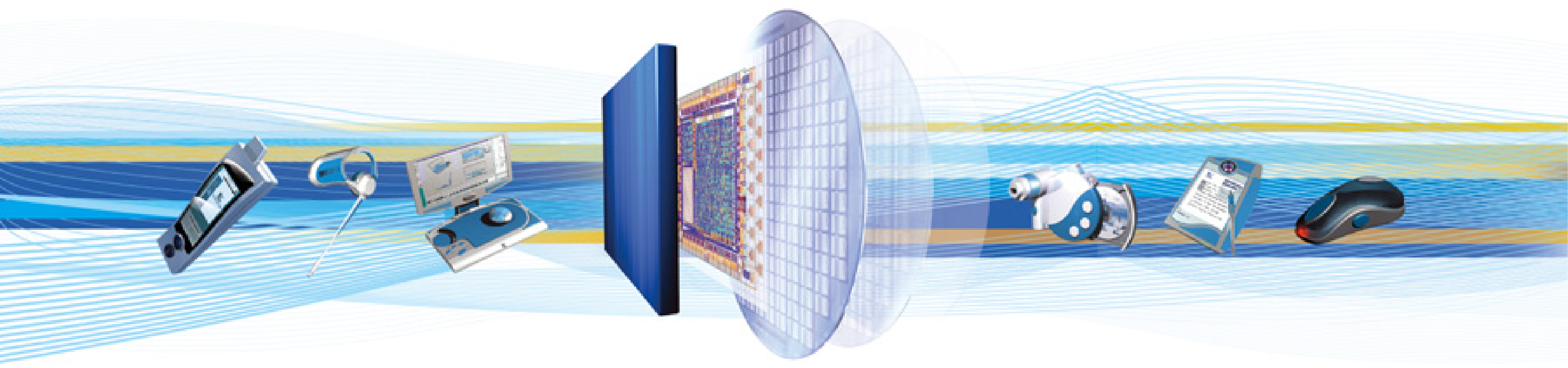
CSR Synergy Framework 3.1.0

Pmem

API Description

August 2011



Cambridge Silicon Radio Limited

Churchill House
Cambridge Business Park
Cowley Road
Cambridge   CB4 0WZ
United Kingdom

Registered in England and Wales 3665875

Tel: +44 (0)1223 692000
Fax: +44 (0)1223 692001
www.csr.com

Contents

1 Introduction

This document describes the functionality and interface provided by the CSR Synergy Pmem API. The Pmem API provides an interface for dynamic memory allocation and freeing.

Depending on the platform, memory management can be implemented in different ways. One way is to reuse the `malloc()` call if such exists. It must however be considered if memory allocation, fragmentation etc. is sufficiently efficient for real time systems. If this is not the case, one may use pre-allocated memory with memory allocated from a set of pools. If pre-allocated memory is used for it must be assured that sufficient memory is allocated in the pools. The major loss using pool based memory is the inability to handle requests for arbitrary-sized memory blocks and an inevitable overhead of memory allocated.

The memory management API is declared in the file `csr_pmem.h`.

2 Pmem API

2.1 Basic Types

The following basic macros must be defined in `csr_pmem.h`.

Prototype

```
#include "csr_pmem.h"

#define pnew(t)   ((t *) (CsrPmemAlloc(sizeof(t))))
#define zpnew(t)  ((t *) (CsrPmemZmalloc(sizeof(t))))
```

2.2 CsrPmemAlloc

Prototype

```
#include "csr_pmem.h"

void *CsrPmemAlloc(CsrSize size);
```

Description

This function will allocate a contiguous block of memory of at least the specified size in bytes and return a pointer to the allocated memory. This function is not allowed to return NULL. A size of 0 is a valid request, and a unique and valid (not NULL) pointer must be returned in this case.

Parameters

Type	Argument	Description
`CsrSize`	`size`	Size of memory requested. Note that a size of 0 is valid.

Returns

Pointer to allocated memory.

2.3 CsrPmemZalloc

Prototype

```
#include "csr_pmem.h"

void *CsrPmemZalloc(CsrSize size);
```

Description

This function is equivalent to CsrPmemAlloc, but the allocated memory is initialised to zero. This is usually implemented as a macro that wraps CsrPmemAlloc and CsrMemSet, but can be implemented in a more efficient way if needed. Please see the description of CsrPmemAlloc for exact behaviour.

Parameters

Type	Argument	Description
`CsrSize`	`size`	Size of memory requested. Note that a size of 0 is valid.

Returns

Pointer to allocated and zero-initialized memory.

2.4 CsrPmemFree

Prototype

```
#include "csr_pmem.h"
void CsrPmemFree(void *ptr);
```

Description

This function will deallocate a previously allocated block of memory.

Parameters

Type	Argument	Description
`void*`	`ptr`	Pointer to allocated memory.

Returns

None.

3 Optional debugging interface

An optional debugging interface exists which can be used for tracking e.g. memory leaks on platforms that don't have built in tools for this. This interface is **optional** and **disabled by default** and *should not be ported to a given platform unless explicitly enabled*.

It is essentially a different variant of CsrPmemAlloc() which takes two extra parameters, file and line, which can be used to identify the exact location in the source code where the allocation was requested.

The interface is enabled by setting the symbolic constant CSR_PMEM_DEBUG during compilation. Note that when this is done, CsrPmemAlloc() is exposed as a macro that calls CsrPmemAllocDebug() passing __FILE__ and __LINE__ to track the allocation with meaningful values.

3.1 CsrPmemAllocDebug

Prototype

```
#include "csr_pmem.h"

#ifdef CSR_PMEM_DEBUG
void *CsrPmemAllocDebug(CsrSize size, const CsrCharString *file, CsrUint32 line);
#define CsrPmemAlloc(size) CsrPmemAllocDebug((size), __FILE__, __LINE__)
#endif
```

Description

This function will allocate a contiguous block of memory of at least the specified size in bytes and return a pointer to the allocated memory. This function is not allowed to return NULL. A size of 0 is a valid request, and a unique and valid (not NULL) pointer must be returned in this case. It is possible to pass this function a file name and a line number to indicate where this allocation was performed from, and it is the intention that the standard C macros `__FILE__` and `__LINE__` should passed.

Parameters

Type	Argument	Description
`CsrSize`	`size`	Size of memory requested. Note that a size of 0 is valid and does not cause `NULL` to be returned.
`const CsrCharString*`	`file`	Name of file from which the allocation was made. `__FILE__` should be passed.
`CsrUint32`	`line`	Line number in above filename where allocation was made from. `__LINE__` should be used.

Returns

Pointer to allocated memory.

4 Memory Management Reference

4.1 Introduction

The CSR Synergy Framework reference ports contain an optional interface for installing hooks that trigger during memory allocation and deallocation which can be used e.g. to track memory leaks or provide runtime consistency checks. This interface is not to be used by any Synergy generic components intended for production use, and thus is not required to be ported. This section presents this reference interface.

Note: The reference API is contained in the file csr_pmem_hooks.h in the special include directory $(BSP_ROOT)/inc/platform. Interfaces in this folder may be platform dependent and they are not necessarily available with any BSP. Thus, the CSR Synergy Framework does not require porting of these interfaces.

4.2 CsrPmemInit

Prototype

```
#include "platform/csr_pmem_init.h"

void CsrPmemInit(void);
```

Description

Initializes the memory management reference implementation. Applications using the reference ports must call this function prior to using the memory allocation API.

Parameters

None.

Returns

None.

4.3 CsrPmemDeinit

Prototype

```
#include "platform/csr_pmem_init.h"

void CsrPmemDeinit(void);
```

Description

De-initializes the memory management reference implementation. Applications using the reference ports must call this function after using the memory allocation API.

Parameters

None.

Returns

None.

5 Memory allocation hooks

5.1 Introduction

The CSR Synergy Framework reference ports contain an interface for hooking into runtime memory allocations using an interface through which callback functions may be registered to be called during memory allocation and deallocation. These callbacks can e.g. log the pointers being allocated, perform accounting to track memory usage over time, or perform consistency checks.

In addition to registering the callbacks, it is possible to register two sizes:

- Header space
- Tail space

If a positive header space value is set, additional header space is allocated in front of the buffer that is passed to the caller of CsrPmemAlloc() which only the callback functions have access to. This space can be used to store accounting data (e.g. where the data was allocated) or other any other data that an application may need to identify the memory. If the header space is of an odd size, padding will be appended to ensure that the user data buffer is correctly aligned.

If a positive tail space value is set, additional space is allocated immediately following the user data buffer with no padding whatsoever. This may be used to insert canary values at the end of user allocations that may then be checked when the memory buffer is deallocated by the application. If the canary value is not found in the tail space, memory has been corrupted and this may be reported in an application-specific manner.

This interface is only enabled if the symbolic constant CSR_MEMALLOC_PROFILING is set during compilation, and by default it is disabled.

5.2 CsrPmemHookAlloc

Prototype

```
#include “csr_pmem_hook.h”

#ifdef CSR_MEMALLOC_PROFILING

typedef void (*CsrPmemHookAlloc)(void *hdr, void *buf, CsrSize count, CsrSize size,

    const CsrCharString *file, CsrUint32 line);

#endif
```

Description

The type of function pointer used for callbacks when allocating memory. Note that it is not passed a pointer to the tail space, but this pointer is simply an offset from the user data buffer and can thus be calculated if needed.

Parameters

Type	Argument	Description
`void*`	`hdr`	Pointer to the header space. Can be cast to any data type. Exclusive access by the callback functions, so a header with metadata can be stored here.
`void*`	`buf`	Pointer to the user data buffer.

CSR Synergy Framework 3.1.0 Pmem API

`CsrSize`	`count`	Number of elements of `size` bytes to allocate. For CsrPmem, this value will always be 1.
`CsrSize`	`size`	The allocation size requested by the application.
`const CsrCharString*`	`file`	A pointer to the file name from which the allocation was requested. May not be available, in which case a dummy file name is passed.
`CsrUint32`	`line`	The line number in the above file which contains the allocation call. If the filename is not available, 0 is passed.

Returns

None.

5.3 CsrPmemHookFree

Prototype

#include “csr_pmem_hook.h”

#ifdef CSR_MEMALLOC_PROFILING

typedef void (*CsrPmemHookFree)(void *hdr, void *buf);

#endif

Description

The type of function pointer used for callbacks when deallocating memory. It is only passed a pointer to the header space and the actual data pointer, so any data from the allocation that are needed here must be stored elsewhere, e.g. in the header. The tail space pointer may be calculated as an offset from the data buffer.

Parameters

Type	Argument	Description
`void*`	`hdr`	Pointer to the header space. Can be cast to any data type. Exclusive access by the callback functions, so a header with metadata can be stored here.
`void*`	`buf`	Pointer to the user data buffer.

Returns

None.

5.4 CsrPmemHookSet

Prototype

#include “csr_pmem_hook.h”

```
#ifdef CSR_MEMALLOC_PROFILING

void CsrPmemHookSet(CsrPmemHookAlloc cbAlloc, CsrPmemHookFree cbFree,

    CsrSize hdrSz, CsrSize tailSz);

#endif
```

Description

This function is used to install the memory hooks and set up the amount of header and tail space to reserve in allocations.

Note: This function must be called *only once*, and it must be called *before any allocations are made* using CsrPmemAlloc(). Essentially, this means it should be called in the low level application initialization code.

Parameters

Type	Argument	Description
`CsrPmemHookAlloc`	`cbAlloc`	Pointer to the header space. Can be cast to any data type. Exclusive access by the callback functions, so a header with metadata can be stored here.
`CsrPmemHookFree`	`cbFree`	Pointer to the user data buffer.
`CsrSize`	`hdrSz`	Amount of header space to allocate.
`CsrSize`	`tailSz`	Amount of tail space to allocate.

Returns

None.

6 Document References

[SYN-FRW-USERS-GUIDE]	CSR Synergy Framework Users Guide. Doc. gu-0001-users_guide

Terms and Definitions

Abbreviation	Explanation
CSR	Cambridge Silicon Radio

Document History

Revision	Date	History
1	23 NOV 09	Initial version
2	30 NOV 09	Ready for release 2.0.0
3	OCT 10	Ready for release 2.2.0
4	DEC 10	Ready for release 3.0.0
5	Aug 11	Ready for release 3.1.0

TradeMarks, Patents and Licences

Life Support Policy and Use in Safety-critical Compliance

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information.