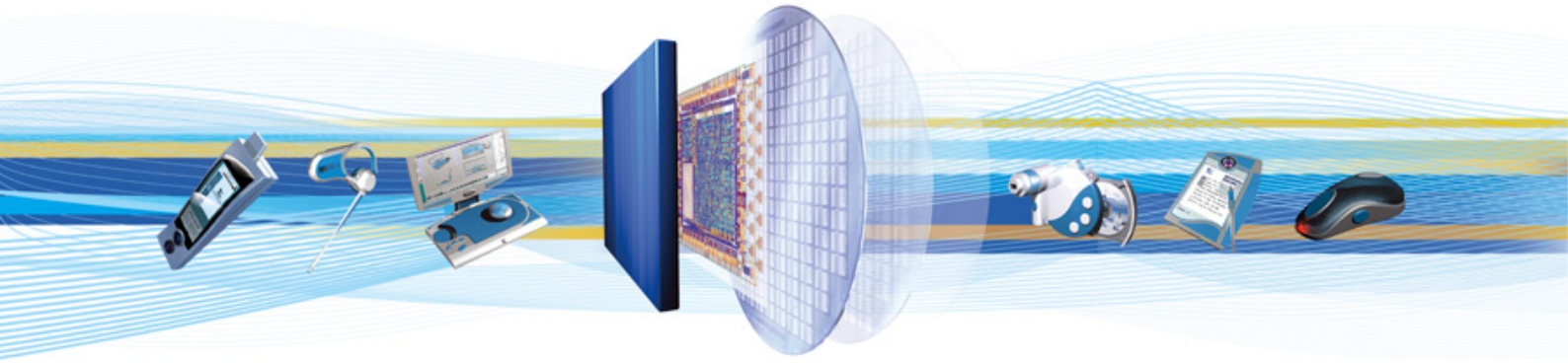
CSR Synergy Framework 3.1.0

ACL-buffering

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 ACL-buffering API. The ACL-buffering API provides an interface for subscribing to a particular ACL-handle and have packets delivered in batches.

On many systems, context switches may have a serious performance overhead. ACL-buffering can be used for buffering ACL-data and deliver it multiple packets at a time to reduce the number of context switches.

The ACL-buffering API must be provided as defined in the file `csr_aclbuf.h`. The ACL-buffering API is special in the sense that it is not based on named functions. Instead it relies on a structure containing function pointers that are used as the interface for the operations supported by the API.

2 ACL-buffering API

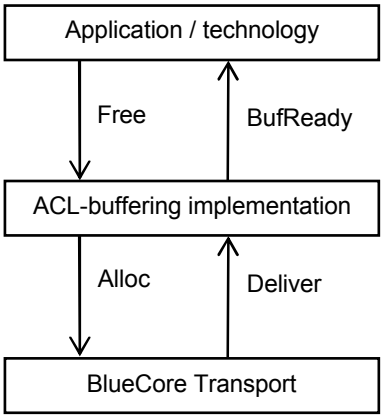
2.1 Fundamental Operations

ACL-buffering is based on the following fundamental operations from a user's point of view:

1. Register an ACL-handle subscription
2. Unregister a subscription
3. Free data
4. Flush buffered data and deliver it
5. Deinitialization

The basic idea is that an application subscribes a particular ACL-handle in order to obtain the data packets that are sent on that handle. An ACL-handle subscription is done by registering a number of callback functions with a given ACL-handle. These callbacks are then used by the ACL-buffering implementation to handle things such as memory allocations from the transport later and passing data to subscribers.

The cycle of ACL-buffering data buffers is like shown in the figure below:



Data is allocated by the BlueCore transports which later return them to the ACL-buffering implementation. When enough data (as determined by the ACL-buffering implementation) has been delivered for a given ACL-handle, it is passed to the subscriber of that handle via its registered `bufready`-callback. When the application is done with the data, it passes the memory buffers back to the ACL-buffer implementation.

2.2 CsrAclBuf

Prototype

```
#include "csr_aclbuf.h"

typedef struct CsrAclBuf

{

    CsrBool (*reg)(CsrUint16 chan,

                   CsrBool doBuffering,

                   CsrAclBufReady bufReady,
```

```
                    CsrAclBufSizer aclSizer,
                    void *privArg);
    void (*dereg)(CsrUint16 chan);
    void (*free)(CsrAclBufElm **elm);
    CsrAclBufElm *(*flush)(CsrUint16 chan);
    void (*deinit)(struct CsrAclBuf **acl);
} CsrAclBuf;
```

Description

The direct interface that technologies will be using for ACL-buffering is the `CsrAclBuf` which holds callback functions that are the entry points to the ACL-buffering implementation.

2.2.1 reg

Parameters

Type	Argument	Description
`CsrUint16`	`chan`	The ACL-channel to subscribe to.
`CsrBool`	`doBuffering`	Whether to buffer packets in batches or to immediately pass on packets.
`CsrAclBufReady`	`bufReady`	The callback function used for delivering packets.
`CsrAclBufSizer`	`aclSizer`	The callback function used for determining complete packet size.

Returns

A boolean indicating whether the registration was successful (TRUE) or not (FALSE).

2.2.2 dereg

Parameters

Type	Argument	Description
`CsrUint16`	`chan`	The ACL-channel to unsubscribe from.

Returns

Does not return a value.

2.2.3 free

Parameters

Type	Argument	Description

`CsrAclBufElm **`	`element`	A pointer to the first ACL-buffer element to free.

Returns

Does not return a value.

2.2.4 flush

Parameters

Type	Argument	Description
`CsrUint16`	`chan`	The ACL-channel to subscribe to.

Returns

A pointer to the first buffer element.

2.2.5 deinit

Parameters

Type	Argument	Description
`struct CsrAclBuf **`	`acl`	A pointer to the pointer to the struct holding the ACL-buffering implementation callback functions.

Returns

Does not return a value.

2.3 Basic Types

ACL-buffering is based on a structure containing function pointers to the individual ACL-buffering operations (as described above). A user of ACL-buffering is thus given a structure which it uses to call the actual ACL-buffering implementation that is being used.

The following types are defined in `csr_aclbuf.h`.

2.3.1 CsrAclBufElm

Prototype

```
#include "csr_aclbuf.h"

typedef struct CsrAclBufElm
{
    struct CsrAclBufElm *next;
    CsrUint16 chanAndFlags;
    CsrUint16 size;
    void *data;
} CsrAclBufElm;
```

Description

`CsrAclBufElm` is the fundamental data type for passing ACL-packets. It is a linked list of structures containing the following elements:

Parameters

Type	Member	Description
`struct CsrAclBufElm *`	`next`	Pointer to next element.
`CsrUint16`	`chanAndFlags`	The channel and flags part of the ACL-header.
`CsrUint16`	`size`	The size in bytes of the ACL-packet.
`void *`	`data`	The raw, unprocessed ACL-packet including headers.

2.3.2 CsrAclBufReady

Prototype

```
#include "csr_aclbuf.h"

typedef void (*CsrAclBufReady)(CsrAclBufElm *elm, void *priv);
```

Description

When enough packets have been buffered, the ACL-buffering implementation will call the associated buffer-ready callback function. The callback is of type `CsrAclBufReady`.

Parameters

Type	Argument	Description
`CsrAclBufElm *`	`elm`	A pointer to the first ACL-packet .
`void *`	`priv`	A pointer to the private data that was registered with the callback.

Returns

Does not return a value.

2.3.3 CsrAclBufSizer

Prototype

```
#include "csr_aclbuf.h"

typedef CsrInt32 (*CsrAclBufSizer)(CsrUint16 channelAndFlags, CsrUint16 size, void
*data);
```

Description

A transport that uses ACL-buffering first reads out the header of the ACL-packet and a certain number of the data bytes that follow it. Before allocating a buffer for the ACL-packet, the ACL-buffering implementation calls a sizer-function that has been registered with the ACL-handle of the ACL-packet. ACL-packets may contain data that has been fragmented across multiple ACL-packets. Sizer-functions should look at the ACL-header and any protocol headers contained within the payload for protocol level information to determine the size of the complete, reassembled packet.

Parameters

Type	Argument	Description
CsrUint16	channelAndFlags	The channel and flags part of an ACL-header.
CsrUint16	size	The amount of ACL-packet payload provided.
void *	data	A pointer to the ACL-packet payload.

Returns

The minimum size of the buffer that must be used for holding the ACL-packet including headers.

2.4 Implementation Notes

How the structure containing the function pointers for ACL-buffering operations is created is implementation-specific and no assumptions may be made about it in generic code. If a generic component needs to use ACL-buffering, it must provide an interface through which an application can provide it with a structure for the current implementation.

One example of how to implement initialization of ACL-buffering is shown below:

```
CsrAclBuf *CsrAclBufInit(void)
{
    CsrAclBuf *p = CsrPmemAlloc(sizeof(CsrAclBuf));

    p->reg=csrAclBufRegister;

    p->dereg=csrAclBufDeregister;

    p->free=csrAclBufFree;

    p->flush=csrAclBufFlush;

    p->deinit=csrAclBufDeinit;

    CsrAclBufLowerInit(csrAclBufAlloc, csrAclBufDeliver);

    return(p);
}
```

Another option may be to initialize a statically allocated structure and returning a pointer to it. Again, the details are not important as no generic code will make assumptions about it.

3 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	26 FEB 10	Initial version
2	20 APR 10	Ready for release 2.1.0
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

Unless otherwise stated, words and logos marked with ™ or ® are trademarks registered or owned by CSR plc or its affiliates. Bluetooth® and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc. and licensed to CSR. Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by CSR plc.

CSR reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, CSR cannot accept responsibility for any errors.

No statements or representations in this document are to be construed as advertising, marketing, or offering for sale in the United States imported covered products subject to the Cease and Desist Order issued by the U.S. International Trade Commission in its Investigation No. 337-TA-602. Such products include SiRFstarIII™ chips that operate with SiRF software that supports SiRFInstantFix™, and/or SiRFLoc® servers, or contains SyncFreeNav functionality.

Life Support Policy and Use in Safety-critical Compliance

CSR's products are not authorised for use in life-support or safety-critical applications. Use in such applications is done at the sole discretion of the customer. CSR will not warrant the use of its devices in such applications.

Performance and Conformance

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