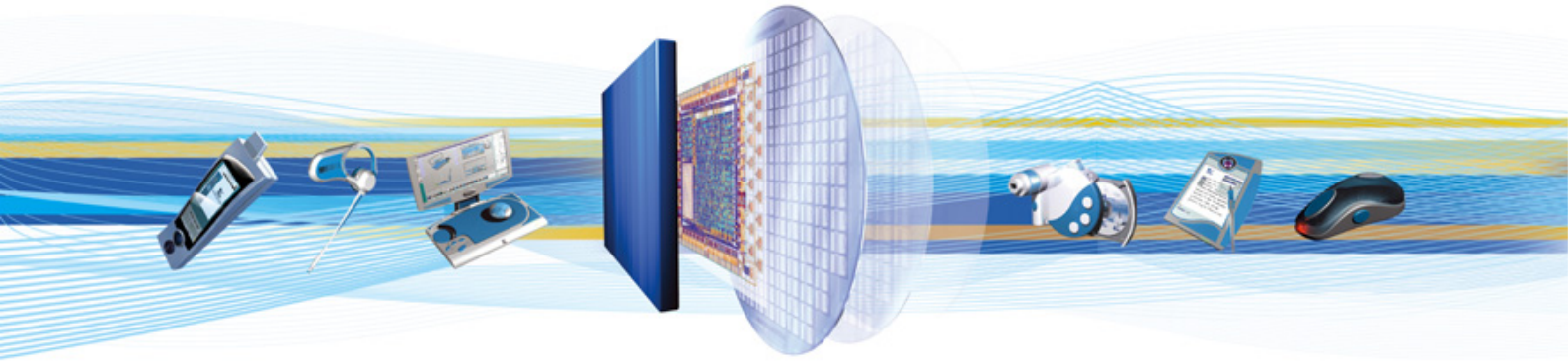
CSR Synergy Framework 3.1.0

Generic Scheduler

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

The CSR Synergy Framework contains a generic platform independent (GSP) implementation of the Scheduler API described in *CSR Synergy Framework Scheduler API* [SYN-FRW-SCHED-API]. The generic scheduler implementation is multi-instance allowing multiple scheduler instances running in separate OS threads.

Important: The generic scheduler requires a BSP implementation of parts of the *CSR Synergy Framework Extensions API* [SYN-FRW-EXTENSION-API]. The Extensions API services that are required are events, mutexes, threads, and memory management. These services are used as foundation for the generic scheduler implementation.

The Generic Scheduler API is identical to the Scheduler API described in *CSR Synergy Framework Scheduler API* [SYN-FRW-SCHED-API]. The Generic Scheduler API adopts, with minor changes, the reference interface for the task initialization and task scheduling presented in Section 4 of [SYN-FRW-SCHED-API]. It adopts completely the interface for Message Forwarding, Timer Handling, Memory Management, and Background Interrupts, presented in Sections 5-9 in [SYN-FRW-SCHED-API].

The next section describes the minor change in the scheduler initialization API.

Note: For a CSR Synergy Framework architectural overview, please refer to *CSR Synergy Framework Users Guide* [SYN-FRW-USERS-GUIDE].

2 Generic Scheduler Initialization

The generic scheduler allows multiple scheduler instances to be running in separate threads. Thus, the scheduler initialization function CsrInitSched has been extended to allow for specification of thread priority and stack size for a given scheduler instance.

Note: The generic scheduler initialization API is contained in file **`csr_gsched_init.h`** in the include directory `$(GSP_ROOT)/inc/gsched`. Source code for the generic scheduler implementation is contained in file **`csr_gsched.c`** in the `$(GSP_ROOT)/src/gsched` directory.

2.1 CsrInitSched

Prototype

```
#include "csr_gsched_init.h"

void *CsrInitSched(CsrUint16 id, CsrUint16 priority, CsrUint32 stackSize);
```

Description

This function initializes the scheduler implementation. I.e. it initializes the internal structures used by the scheduler like task queue structures, background interrupts, etc. The function returns the private scheduler instance data (as a void pointer), which must be passed to subsequent scheduler functions.

In order to differentiate between multiple schedulers or multiple threads it is necessary to know *where* a given task is run, i.e. in which scheduler it is run. To accomplish this a *scheduler identification* is used, which is the id that is passed as a parameter to `CsrInitSched()` (and also to the `CsrSchedRegisterTask()` function).

Each scheduler instance can be assigned a thread priority and a thread stack size. Priority and stack size values are passed unchanged to the CsrThreadCreate() call of the underlying Framework Extension API. Valid priority and stack size values are determined be the Framework Extensions API.

Parameters

Type	Argument	Description
`CsrUint16`	`id`	The scheduler identifier.
`CsrUint16`	`priority`	The scheduler thread instance priority
`CsrUint32`	`stackSize`	The scheduler thread instance stack size.

Returns

The private scheduler instance data as a void pointer.

2.2 CsrSchedRegisterExternalSend

Prototype

```
#include "csr_gsched_init.h"

void CsrSchedRegisterExternalSend(CsrBool (*f)(CsrSchedQid q,CsrUint16 mi, void
*mv));
```

Description

This function registers a callback function used for external communication (outside scheduler context). Valid external queue range starts at:

```
#define CSR_SCHED_QUEUE_EXTERNAL_LOWEST (0x7800).
```

Sending messages using queue ids in the external range causes the generic scheduler to call the registered callback.

Parameters

Type	Argument	Description
`CsrBool (*)(CsrSchedQid, CsrUint16, void *)`	`f`	The external send callback function.
`CsrSchedQid`	`q`	The destination queue. Valid external queue range starts at CSR_SCHED_QUEUE_EXTERNAL_LOWEST
`CsrUint16`	`mi`	Message integer part.
`void *`	`mv`	Message pointer part.

Returns

The private scheduler instance data as a void pointer.

3 Document References

[SYN-FRW-USERS-GUIDE]	CSR Synergy Framework Users Guide. Doc. gu-0001-users_guide
[SYN-FRW-SCHED-API]	CSR Synergy Framework Scheduler API. Doc. api-0004-sched
[SYN-FRW-EXTENSION-API]	CSR Synergy Framework Extensions API. Doc. api-0007-extensions

Terms and Definitions

Abbreviation	Explanation
CSR	Cambridge Silicon Radio

Document History

Revision	Date	History
1	20 OCT 09	Initial version
2	30 NOV 09	Ready for release 2.0.0
3	20 APR 10	Ready for release 2.1.0
4	OCT 2010	Ready for release 2.2.0
5	DEC 2010	Ready for release 3.0.0
6	Aug 2011	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.