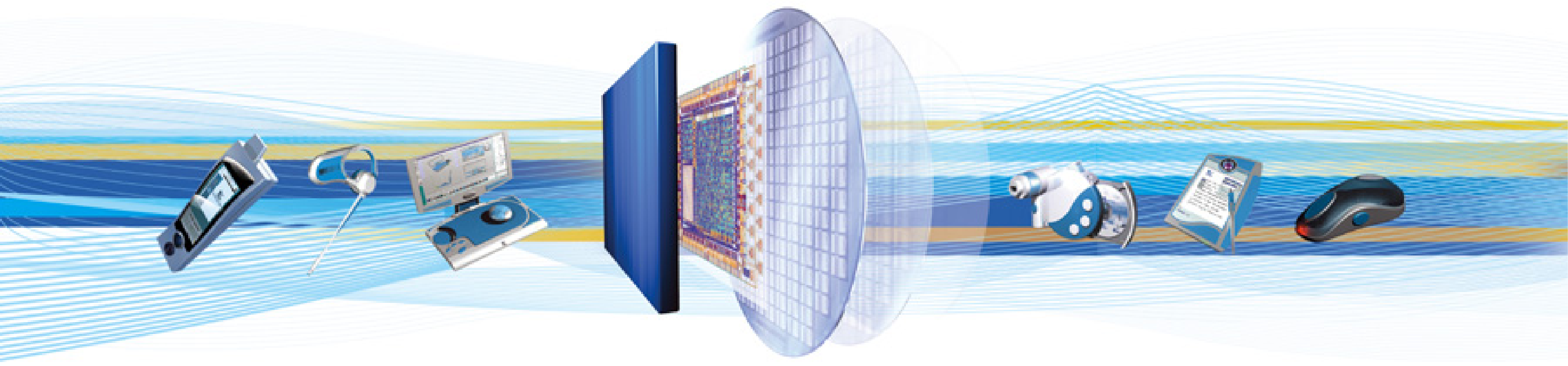
CSR Synergy Framework 3.1.0

HQ – Host Query

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

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Scope

This document describes the functionality and message interface used for receiving Host Query (HQ) commands from the BlueCore® chip.

Before reading this document, it is recommended to read [1] and [2].

2 Description

2.1 Introduction

The Host Query (HQ) protocol is used for accessing a command interpreter in the BlueCore® chip. The command interpreter present commands that allow the chip to signal, monitor and control its host. The protocol is not part of the Bluetooth standard. Examples of HQ commands available are:

- Booted: The BlueCore® chip can be instructed to send this command once it has finished booting.
- Delayed_Panic: The BlueCore® chip has encountered an irrecoverable error condition and is about to reboot.
- BER_Trigger: The BlueCore chip has received a packet with a Bit Error Rate (BER) that exceeds a configured threshold.

HQ is also required to support the DSPM functionality, please see [3]. For a full list of available commands, please refer to [1] and [2].

An application on top of the CSR Synergy Framework HW module can receive HQ commands sent from the BlueCore® chip using the HQ module, see Figure 1. To receive the commands, the application needs to communicate with the HQ module only, as it will handle all necessary communication through the HCI layer.

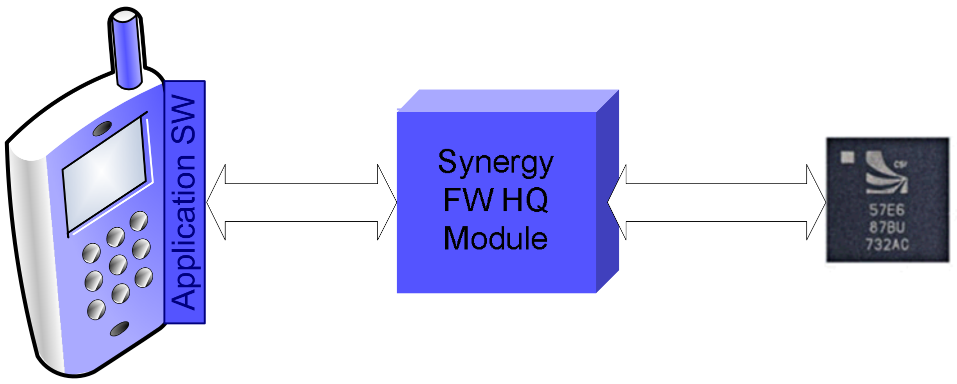


Figure 1: Typical interface

If an application wants to subscribe to a HQ command, it registers the appropriate HQ command with the HQ module, using a set of functions and primitives provided by the HQ module. These are described in chapter 3 of this document.

It is possible for more than one application to subscribe to a specific HQ command. If the HQ module receives an HQ command, which more than one application subscribe to, it will forward the command to all subscribers. If no subscribers to the command received exist, the HQ module will silently discard the command, unless it is an error command. An error command, which no-one subscribes to, will result in a panic in the HQ module. Error commands, which have subscribers, will be forwarded to these subscribers, and no panic will occur in the HQ module.

A common way of using the HQ module is depicted in Figure 2. In this figure it is shown that if an application wants to subscribe to a specific HQ command, it shall register that with the HQ module. The application does this by sending a CSR_HQ_REGISTER_REQ to the HQ module and after it receives a CSR_HQ_REGISTER_CFM back, it will receive the specific HQ commands when they arrive from the BlueCore® chip. There are no limits to how many HQ commands an application can receive.

When the application wants to unsubscribe from a specific HQ command, it shall deregister by sending a CSR_HQ_DEREGISTER_REQ to the HQ module. After receiving a successful CSR_HQ_DEREGISTER_CFM signal from the HQ module, the application will no longer receive any HQ commands of the type specified in the CSR_HQ_DEREGISTER_REQ signal.

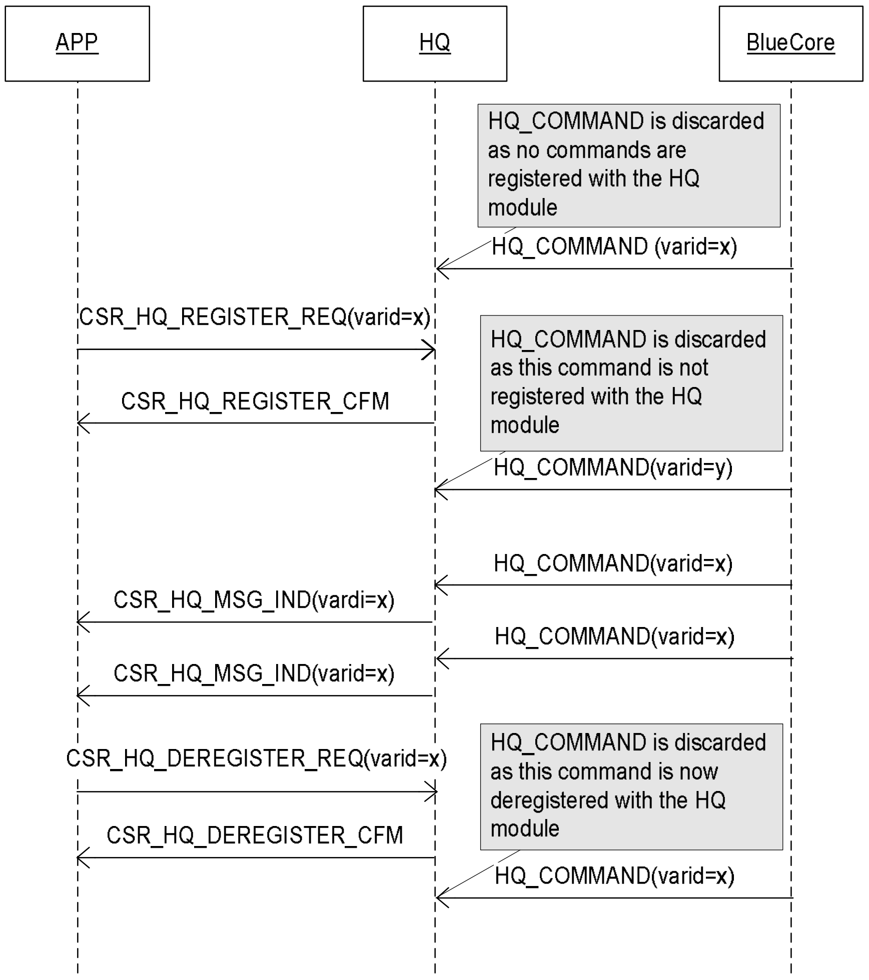

APP
HQ
BlueCore
HQ_COMMAND is discarded as no commands are registered with the HQ module
HQ_COMMAND (varid=x)
CSR_HQ_REGISTER_REQ(varid=x)
HQ_COMMAND is discarded as this command is not registered with the HQ module
CSR_HQ_REGISTER_CFM
HQ_COMMAND(varid=y)
HQ_COMMAND(varid=x)
CSR_HQ_MSG_IND(vardi=x)
HQ_COMMAND(varid=x)
CSR_HQ_MSG_IND(varid=x)
HQ_COMMAND is discarded as this command is now deregistered with the HQ module
CSR_HQ_DEREGISTER_REQ(varid=x)
CSR_HQ_DEREGISTER_CFM
HQ_COMMAND(varid=x)

Figure 2: Example of how the CSR Synergy Framework HQ module may be used

2.2 Sequence Overview

The HQ module will always (after it has been initialised by the scheduler) be able to do three jobs:

- Register which HQ commands an application wants to subscribe to
- De-register, i.e. end the subscription to some or all HQ commands
- Receive HQ commands from the BlueCore® chip and forward them to the applications (if they subscribe to the commands received)

For further details on how these jobs are done, please refer to section 3.

3 Interface Description

In this section a series of MSCs will be shown to explain the usage of the HQ module. The primitives and the functions available to the application are also described in the subsections of this chapter.

If CSR_BLUECORE_ONOFF is defined, and the BlueCore enters the deactivating state no further CSR_HQ_MSG_IND messages will be sent to any registered tasks until the BlueCore is again activated, and any CSR_HQ_MSG_RES messages will be discarded on reception. The registrations persist across BlueCore activation and deactivations and it is possible to register before the BlueCore is activated.

3.1 Registration

If an application wants to receive a particular HQ command from the BlueCore® chip, it must first register the command with the HQ module. A registration is initiated using one of the `CsrHqRegisterReqSend*` functions, which send a CSR_HQ_REGISTER_REQ primitive to the HQ module.

The prototypes for the `CsrHqRegisterReqSend*` functions are:

```
CsrHqRegisterReqSend(CsrQid pHandle, CsrHqVarIdType varId)

CsrHqRegisterReqSend2(CsrQid pHandle, CsrHqVarIdType varId, CsrBool response)
```

The arguments to the functions are described in Table 1.

Type	Argument	Description
CsrQid	phandle	Handle to application, i.e. the queue on which the CSR_HQ_REGISTER_CFM and CSR_HQ_MSG_INDs are put by the HQ module
CsrHqVarIdType	varId	The identification number for the HQ command that the application wants to subscribe to. For further information on the different HQ commands, please refer to [1].
CsrBool	response	Set to TRUE if the application needs to send a response to the HQ command. If set to FALSE, the HQ module will automatically (and immediately) send a successful response to the HQ command upon reception. When using the CsrHqRegisterReqSend function, this value will be fixed to FALSE, which yields backward compatible behaviour. Please note that only one entity can register for a particular HQ command with this parameter set to TRUE as only one entity can provide a response to any given HQ command.

Table 1: Arguments to `CsrHqRegisterReqSend*` functions

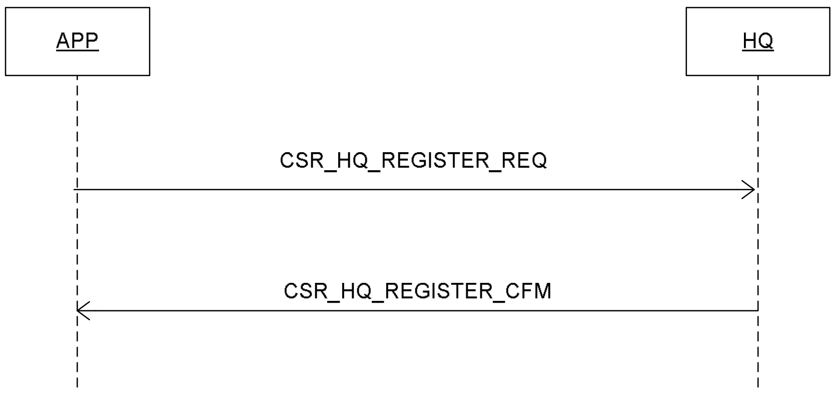


Figure 3: HQ registration

When the HQ module has finished handling the CSR_HQ_REGISTER_REQ, it will confirm the request by sending a CSR_HQ_REGISTER_CFM primitive, which contains the parameters found in Table 2 below. Note that the confirm signal is generated locally, as the registration is made in the HQ module. The reason for this is that the sorting of which HQ commands are forwarded (and to where) is done locally by the HQ module, as it will receive all HQ commands sent from the BlueCore® chip (see Section 3.3).

Currently the *result* parameter will always equal CSR_HQ_OK (0x0000) as no error can occur in the registration process. All other result codes are reserved for future use and will indicate that an error has occurred.

Type	Parameter	Description
CsrHqPrim	type	Signal identity – always CSR_HQ_REGISTER_CFM.
CsrHqVarIdType	varId	The identification number for the HQ command that the application wants to subscribe to. For further information on the different HQ commands, please refer to [1].
CsrResult	result	Result code. Currently this will always be CSR_RESULT_SUCCESS (0x0000) indicating a successful registration. All other result codes are reserved for future use and will indicate that an error has occurred.

Table 2: Parameters in a CSR_HQ_REGISTER_CFM primitive

3.2 Deregistration

The application can de-register a particular command with the HQ module by sending a CSR_HQ_DEREGISTER_REQ primitive. This is done using the `CsrHqDeregisterReqSend` function.

The prototype for the `CsrHqDeregisterReqSend` function is:

```
    CsrHqDeregisterReqSend(CsrQid pHandle, CsrHqVarIdType varId)
```

The arguments to the function are stated in Table 3.

Type	Argument	Description
CsrQid	phandle	Handle to application, i.e. the queue on which the CSR_HQ_DEREGISTER_CFM signal is put by the HQ module
CsrHqVarIdType	varId	The identification number for the HQ command that the application wants to subscribe to. For further information on the different HQ commands, please refer to [1].

Table 3: Arguments to `CsrHqDeregisterReqSend` function

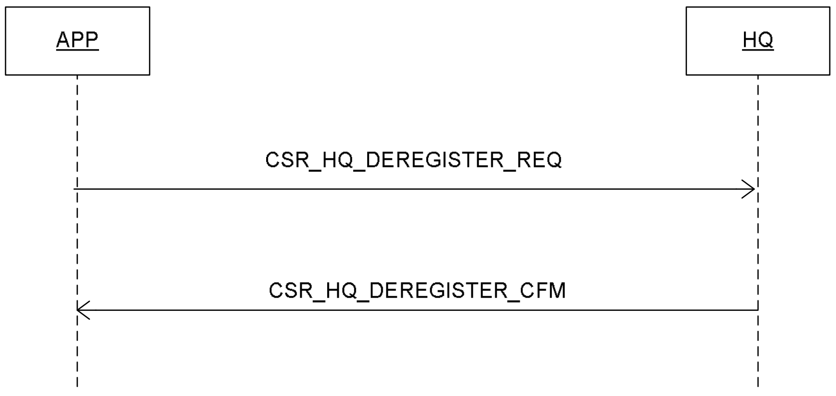


Figure 4: HQ Deregistration

When the HQ module has finished handling the CSR_HQ_DEREGISTER_REQ, it will confirm the request by sending a CSR_HQ_DEREGISTER_CFM primitive, which contains the parameters below. Note that the confirm signal is generated locally, as the deregistration is made in the in the same place as the registration (see Section 3.1).

If an error occurs during the deregistration procedure, the *result* parameter will different from CSR_HQ_OK (0x0000). Currently an error occurs only if the *varId* given in the CSR_HQ_DEREGISTER_REQ was not registered correctly in the HQ module. In this case *result* will equal CSR_HQ_NO_SUCH_VARID (0x0001).

Type	**Parameter**	**Description**
CsrHqPrimType	type	Signal identity – always CSR_HQ_DEREGISTER_CFM.
CsrHqVarIdType	varId	The identification number for the HQ command that the application wants to subscribe to. For further information on the different HQ commands, please refer to [1].
CsrResult	Result	Result code. Currently this will be CSR_RESULT_SUCCESS (0x0000) indicating a successful registration or CSR_HQ_RESULT_NO_SUCH_VARID (0x0001) if the varId was not registered. All other result codes are reserved for future use and will indicate that an error has occurred.

Table 4: Parameters in a CSR_HQ_DEREGISTER_CFM primitive

3.3 HQ Command Forwarding

Once the HQ module has been initialised, HQ commands from the BlueCore® chip may be received in the HQ module. When the HQ module receives a HQ command from the BlueCore® chip, it checks if the command should be forwarded to an application. If so, this is done using a CSR_HQ_MSG_IND primitive. Depending on the registration parameters, the application is expected to send a response after reception of the CSR_HQ_MSG_IND as the HQ module only sends the response to the BlueCore HQ_COMMAND by itself if no application has indicated it needs to send a response.

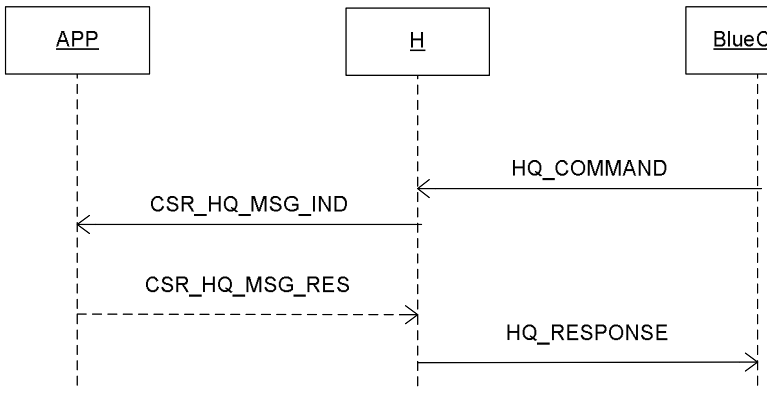


Figure 5: HQ message forwarding

The parameters of the CSR_HQ_MSG_IND primitive are found below.

Type	**Parameter**	**Description**
CsrHqPrimType	Type	Signal identity – always CSR_HQ_MSG_IND.
CsrUint16	cmdType	HQ command type. Possible values are 0x0000 (GETREQ) and 0x0002 (SETREQ). See [1] and [2] for further details.
CsrUint16	seqNo	Sequence number of the HQ command. This value is set solely by the BlueCore® chip.
CsrHqVarIdType	varId	varId of the HQ command
CsrUint16	payloadLength	Length of the HQ command payload pointer in bytes.

Type	Parameter	Description
CsrUint16	*payload	Pointer to the HQ command payload. For details on how to interpret the payload, please refer to [1] and [2].

Table 5: Parameters in a CSR_HQ_MSG_IND primitive

The parameters of the CSR_HQ_MSG_RES primitive are found below

Type	Parameter	Description
CsrHqPrimType	Type	Signal identity – always CSR_HQ_MSG_RES.
CsrUint16	cmdType	HQ command type. Possible values are 0x0001 (GETRESP). See [1] and [2] for further details.
CsrUint16	seqNo	Sequence number of the HQ response. This value is set solely by the BlueCore® chip and must match the sequence number of the corresponding HQ command.
CsrHqVarIdType	varId	varId of the HQ response. Must match the varId of the corresponding HQ command.
CsrUint16	status	Status field of the HQ message. See [1] and [2] for further details.
CsrUint16	payloadLength	Length of the HQ command payload pointer in bytes.
CsrUint16	*payload	Pointer to the HQ command payload. For details on how to interpret the payload, please refer to [1] and [2].

Table 6: Parameters in a CSR_HQ_MSG_RES primitive

Most of the parameters given in the CSR_HQ_MSG_IND and CSR_HQ_MSG_RES primitives are taken directly from the original HQ commands sent from BlueCore® chip. The structure of the HQ commands is depicted in Figure 6.

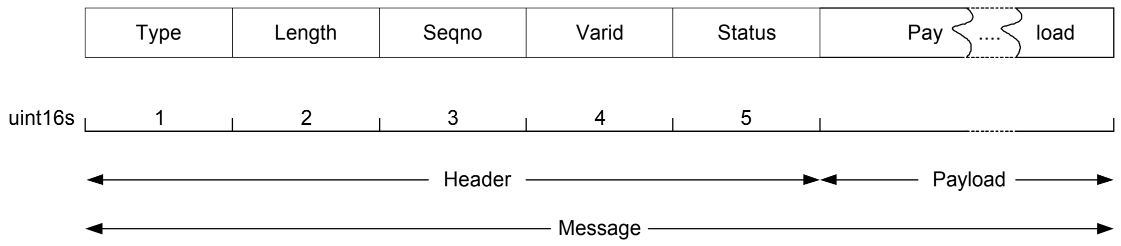


Figure 6: HQ command structure

The *type*, *seqno* and *varId* present in the CSR_HQ_MSG_IND and CSR_HQ_MSG_RES are all 16 bit variables which can be read directly.The data pointed to by the **payload* parameter is the payload of the original HQ command sent from the BlueCore® chip and *payloadLength* indicates the length of this payload in bytes, not in words as described by the protocol. For details on how to interpret the data pointed to by the **payload* parameter, please refer to [1] and [2].

When interpreting the payload, it should be noted that the HQ protocol is shaped to fit the XAP processor within the BlueCore® chip in which each data address accesses a 16-bit value. This means that the fundamental unit of storage in HQ messages is a 16-bit (unsigned) integer (uint16) rather than a byte.

8-bit integers are sent as 16-bit integers; their values are promoted to the larger integer size before transmission, whereas 16-bit integers are sent with the less significant byte first, see Figure 7. 32-bit integers are sent with a more awkward byte ordering found in Figure 8.

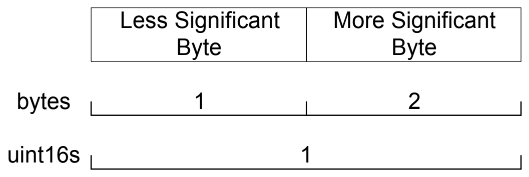


Figure 7: 16-bit HQ message format

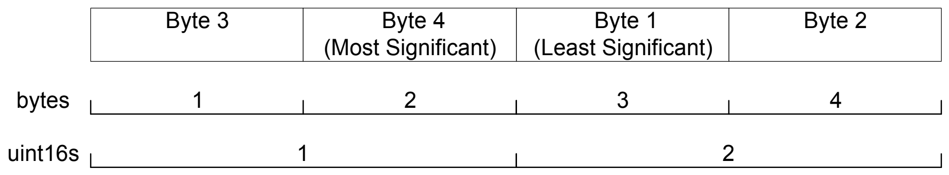


Figure 8: 32-bit HQ message format

4 Document References

Ref	**Title**
[1]	"HQ Commands" (bcore-sp-003Pd) available for download at www.csrsupport.com
[2]	"HQ Protocol" (bcore-sp-011Pb) available for download at www.csrsupport.com
[3]	"DSPM API" (api-0034-dspm)

Terms and Definitions

BlueCore®	Group term for CSR's range of Bluetooth wireless technology chips
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
CSR	Cambridge Silicon Radio
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards
HCI	Host Controller Interface
HQ	Host Query

Document History

Revision	Date	History
1	19 DEC 08	Ready for first Engineering Release
2	16 JAN 09	Ready for second Engineering Release
3	09 FEB 09	Ready for release 1.0.0
4	26 MAY 09	Ready for release 1.1.0
5	30 NOV 09	Ready for release 2.0.0
6	20 APR 10	Ready for release 2.1.0
7	OCT 10	Ready for release 2.2.0
8	DEC 10	Ready for release 3.0.0
9	May 11	Ready for release 3.1.0

TradeMarks, Patents and Licences

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.