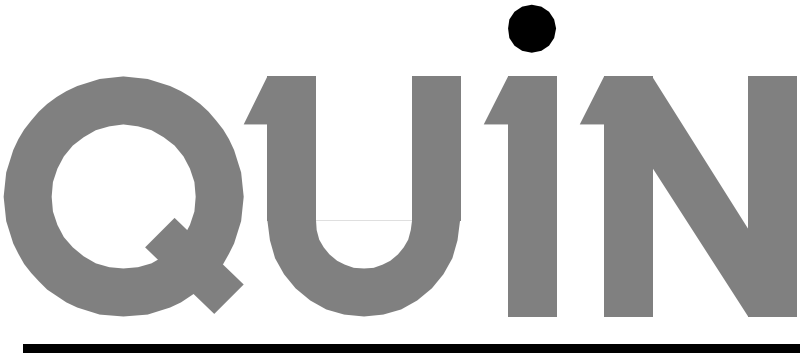

Quin Systems Limited
BMSG232C Serial Gateway
User Manual

MAN606

Issue 1.4
March 1998

Copyright Notice

Important Notice

Quin Systems reserves the right to make changes in the products described in this document in order to improve design or performance and for further product development. Examples given are for illustration only, and no responsibility is assumed for their suitability in particular applications. Reproduction of any part hereof without the prior written consent of Quin Systems is prohibited.

Although every attempt has been made to ensure the accuracy of the information in this document, Quin Systems assumes no liability for inadvertent errors. Suggestions for improvements in either the products or the documentation are welcome.

European Community Directives

This product has been tested in typical configurations and meets the protection requirements of the EMC Directive (89/336/EEC) as amended by Directives 92/31/EEC and 93/68/EEC, when fed from power supplies which themselves meet the requirements of Directives 89/336/EEC, 92/31/EEC and 93/68/EEC.

If this product is to be incorporated into a system for the control of machinery:

- it must not be relied upon to provide safety-critical features such as guarding or emergency stop functions.
- it must not be put into service until the machinery into which it has been incorporated has been declared in conformance with the Machinery Directive (89/392/EEC) as amended by Directives 91/368/EEC, 93/44/EEC and 93/68/EEC.

This product, as normally supplied, has low voltages accessible to touch and must be mounted within a suitable cabinet to meet any required IP rating to BS EN 60529.The installation instructions in this manual should be followed in constructing a system which complies with all relevant Directives.

Versions

This manual reflects the following versions of hardware and software:

- Intel iDCM44 version 2.1
- BMSG232C Serial Gateway Software version 1.3 (& newer)
- BMSG232 Bitbus Serial Module Issue C

Quin Systems Limited
Oaklands Business Centre,
Oaklands Park,
Wokingham,
Berkshire.
RG41 2FD

Telephone:	+44 (0) 118 977 1077
Facsimile:	+44 (0) 118 977 6728
Email:	support@quin.co.uk
WWW Site:	http://www.quin.co.uk
Compuserve:	100572,1631

ISSUE	*REVISION HISTORY*		*DATE*
Draft	Initial draft copy for comment	TAW	Jan 1992
1.0	Original released version	TAW	Aug 1992
1.1	Modified for compliance with EC Directives Minor modifications to text descriptions and format New title & manual number assigned	TJL	Apr 1996
1.2	Added description for ECHO command	TJL	Sept 1996
1.3	Added description of RAWMSG command	IDH	Oct. 1997
1.4	Revised to include comments for Euromap 15	IDH	Mar. 1998

Contents

1. Introduction

This document describes the operation and general use of the Quin Serial to Bitbus Gateway.

The Serial Gateway module uses a standard BMSG232C Serial Bitbus Module running custom software which acts as a server to pass messages between a Bitbus network and a terminal on the serial port. This allows a computer or terminal attached to the serial port to access Bitbus transparently. A high level monitor program is provided to allow access to the Bitbus network via a user friendly interface.

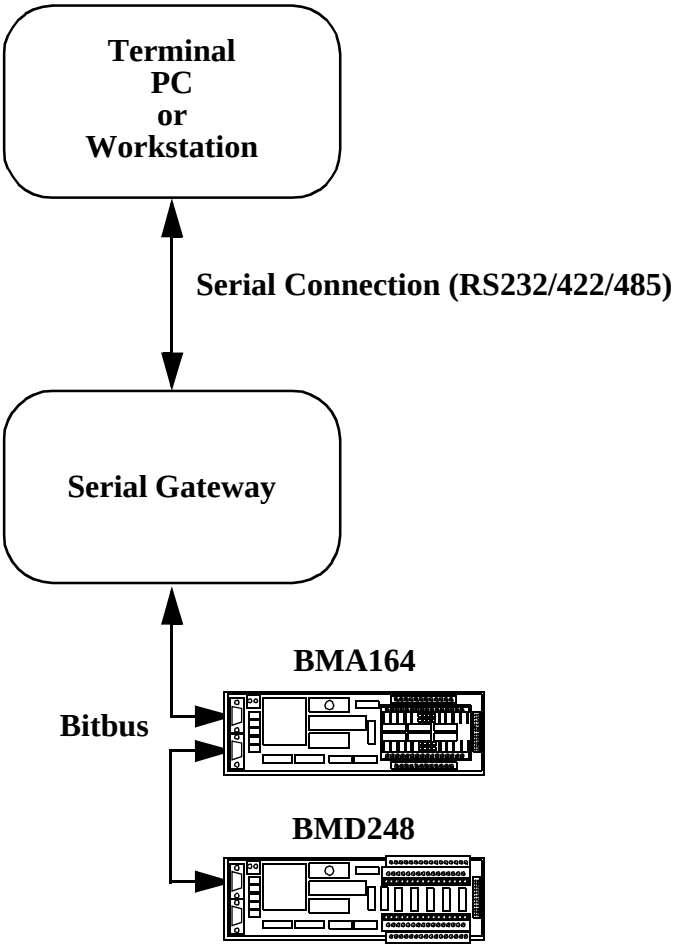

Figure 1. System Block Diagram.

2. General Description

2.1 BMSG232C Serial Gateway

The BMSG232C Serial Gateway comprises a standard BMS232C serial Bitbus module fitted with additional firmware.

The BMSG232C module provides a complete interface to the Intel Bitbus distributed network.The BMSG232C has its own Intel 8044 processor with private program and data memory. The 8044 is a single chip microcontroller device, with a built-in high speed serial interface dedicated to the Bitbus link. The variant used on the BMSG232C includes the Intel iDCM44 firmware package in ROM, comprising the iDCX51 real-time executive and the Remote Access and Control task (RAC). This provides support for up to eight tasks (including RAC), message passing between tasks, and message passing to remote tasks on other Bitbus nodes. It also provides several useful housekeeping facilities such as timers and interrupt handling. The standard Serial Bitbus module may be used either as a Bitbus master or slave node.

The 8044 has a built in high speed serial interface which supports a subset of the IBM SDLC protocol, and it is this interface which is used for the Bitbus communications. Use of the Bitbus serial interface is transparent to the user program as it is dealt with by the iDCX firmware. The electrical interface to Bitbus conforms to the RS485 standard, and in addition is isolated from the local power supply.

2.2 Serial Gateway Monitor

In addition to the standard serial driver tasks, the Serial Gateway is supplied with a front end monitor program which is designed to be run from serial port A, via an RS232 terminal or PC serial port.

The monitor program provides a user friendly front end to the RAC commands. Bitbus message and error handling is taken care of by the monitor program which, in verbose mode, provides the user with information in an easy-to-read format. This monitor program allows transparent access to any Bitbus module on the connected Bitbus network, via serial port A on the BMSG232C module.

3. Programming

3.1 General

The monitor program provides a user friendly interface to the Bitbus RAC commands and allows transparent gateway access to remote Bitbus I/O Modules.

The monitor is configured to be permanently executing and provides a user prompt and status or error reporting on the serial port, in response to user requests.

3.2 Configuration

The Serial Gateway module as supplied is configured to run with the following serial communications settings on port A:

Baud Rate 9600
1 Stop Bits
8 bit Char Length
No Parity
Hardware Handshake
RS232 DCE

All the above settings are fixed when the gateway firmware is compiled, excluding the RS232 configuration which is hardware dependent. If other settings are desired, for example a different Baud rate setting, please contact Quin Systems who can supply Eproms holding the desired settings.

The serial hardware interface as supplied is a fully functional RS232 DCE device, this can be changed to either RS422 or RS485 if required. It is recommended that the customer returns the unit to Quin Systems to carry this modification out. The unit would then re-enter Quin Systems QA procedures and would therefore be subject to full system testing after modification.

The standard configuration uses Channel/Port A (P4) on the BMSG232 module for use with the monitor program. With the Bitbus connectors on the left, Port A is the upper 25-way connector. Refer to the BMS232 Hardware Manual (MAN604) for more information on the hardware related aspects of the serial gateway, including full information on the serial interface.

3.3 Serial Gateway Monitor Commands

3.3.1 Introduction

The Monitor program provides a user shell, from which the operator enters strings of commands and is presented with information in a readable format. In the case where parameters are required for the individual commands, these are entered on the command line in the format specified below.

Generally commands that require parameters expect them directly after the command string with a SPACE (ASCII 20) character in front of the parameter. Multiple parameters are again delimited with SPACE characters. All numeric values are entered in Hexadecimal notation. A Carriage Return (ASCII 13) character terminates the command input. Alpha characters can be entered in upper or lower case.

Example 1: **rxmem 22 1000 8**

This command string instructs the serial gateway to read 8 bytes of data from external memory on the Bitbus module with node address 22H, starting at memory location 1000H.

Example 2: **smsg 14 2 A0 10 20**

This command string instructs the serial gateway to send a Bitbus message to module 14H (node address) task id. 2 with a Cmd/Resp field A0H and contents of 10H and 20H.
.

The following sections describe each of the commands supported by the gateway, providing a brief explanation and an example of the use each command.

NB: The gateway software will always address itself (the BMSG232C serial module) as node 0, regardless of the node address set on the hexadecimal switches.

3.3.2 Miscellaneous Commands

verbose

This command toggles the verbose mode On and Off. In verbose mode command replies are displayed in full with user messages enabled. The module will always power up with verbose mode On.

Example:
```
>verbose
```

Response:
```
OFF
or
ON
>
```


echo This command toggles the echo mode On and Off. With echo mode off command requests are NOT echoed back on the serial port.

Example:
```
>echo
```

Response:
```
OFF
or
ON
```

load ***<node address> <CR> <filename.hex>***

This command allows Intel format (*.HEX) program files to be downloaded to a programmable Bitbus node on the Bitbus network. It uses the RAC download code command, but automatically formats the program file correctly for Bitbus and provides full error checking.

The file is written to the serial port after the download command has been issued, i.e. after the Carriage Return (ASCII 13). The characters written during the download are not echoed. The download is terminated on the input parser recognising the Intel Hex file terminator

See the section 3.4 for a list of possible error codes.

Example:
```
>load 22
<filename.hex>
```

3.3.3 Raw Bitbus Message Commands

smsg ***<node address> <taskid> <cmd/resp> <data...>***

A raw message can be transmitted to any task on any node (if it exists). The command waits for a returned message and prints the response. If no message is received within the timeout period (approx. 250ms), the Serial Master node will clear the outstanding message and return a relevant error message. If a message is sent to a non-existing node or task, the command reply displays the full Bitbus message rather than the gateway error condition.

Example:
```
>smsg 33 0 0f 00 00 00 00 00 00
```


Response (verbose ON):

```
Returned Message
Length     : D
Flags      : 80
Dest node  : 33
S/D Tasks  : 30
Cmd/Resp   : 0
Contents[0]: 69
Contents[1]: 38
Contents[2]: 30
Contents[3]: 34
Contents[4]: 34
Contents[5]: 20
>
```

Response (verbose OFF):

```
D 80 33 30 0 69 38 30 34 34 20
>
```

rawmsg <length> <flags> <*node address*> <*taskid*> <*cmd/resp*> <*data...*>

This function is similar to **smsg** but allows all Bitbus message header fields to be specified. This permits messages to be sent to node extensions by specifying the correct ***flags***. A raw message can be transmitted to any task on any node (if it exists). The command waits for a returned message and prints the response. If no message is received within the timeout period (approx. 250ms), the Serial Master node will clear the outstanding message and return a relevant error message. If a message is sent to a non-existing node or task, the command reply displays the full Bitbus message rather than the gateway error condition.

Example:

```
>rawmsg 0B 10 33 0 0f 00 00 00 00 00 00
```

Response (verbose ON):

```
Returned Message
Length     : D
Flags      : 80
Dest node  : 33
S/D Tasks  : 30
Cmd/Resp   : 0
Contents[0]: 69
Contents[1]: 38
Contents[2]: 30
Contents[3]: 34
Contents[4]: 34
Contents[5]: 20
>
```

Response (verbose OFF):

```
D 80 33 30 0 69 38 30 34 34 20
>
```


3.3.4 RAC Command Functions

The standard Intel Bitbus Remote Access and Control (RAC) commands form the basis of communicating with slave modules over a Bitbus interconnect. The RAC interface provides memory and I/O read and write facilities, along with remote control and monitoring of tasks on slave nodes.

reset ***<nodeaddress>***

This command initiates a reset at Bitbus node **nodeaddress**. If the reset is successful the node is automatically sent offline to force a node resynchronization.

Example:
```
>reset 55
>
```

NB: After the BMSG232C module is reset (i.e. RESET 0) a character is required to be sent to the module via the RS232 port to "restart" the monitor program.

off ***<nodeaddress>***

resync ***<nodeaddress>***

This command sets the slave node **nodeaddress** to an offline state, this forces the master to resynchronize with the node next time it is sent a message.

Example:
```
>off 88
>
```

ctask ***<nodeaddress> <task descriptor address>***

This command creates a new user task on node **nodeaddress.** The address of the task's Initial Task Descriptor (ITD) is given by **itdaddress**. The command returns the task number (0-7) if successful.

Example:
```
>ctask 33 fff0
```

Response (verbose ON):
```
Task Id = 3
>
```



```
Response (verbose OFF):
T3
>
```

dtask <*nodeaddress*> <*taskid*>

This command deletes an existing task on node **nodeaddress**. The task ID of the task to be deleted is specified in **taskid**.

```
Example:
>dtask 11 2
>
```

get <*nodeaddress*>

This command returns the function ID list for node **nodeaddress**. The list is displayed as the FID structure.

```
Example:
>get 33

Response (verbose ON):
Task 0: 1
Task 1: E4
Task 2: E5
Task 3: 0
Task 4: 0
Task 5: 0
Task 6: 0
Task 7: 0
>

Response (verbose OFF):
1 E4 E5 0 0 0 0 0
>
```

protect <*nodeaddress*>

This command disables the remote access commands at node **nodeaddress**. When protected, the RAC service only recognizes the remote control commands and does not execute I/O, upload, download, read or write commands.

```
Example:
>protect 44
>
```


release *<nodeaddress>*

This command enables the remote access commands at node **nodeaddress**. When protected, the RAC service only recognizes the remote control commands and does not execute I/O, upload, download, read or write commands.

Example:
```
>release 44
>
```

wio *<nodeaddress> <ioaddress> <data...>*

This command writes a block of I/O data to node **nodeaddress**. The function writes consecutive bytes starting at the I/O port specified by **ioaddress**. This command can write up to 6 data bytes.

Note that this command behaves slightly differently to the standard RAC command in that it writes a block of data to consecutive addresses rather than writing data to (up to) 6 different addresses.

Example:
```
>wio 22 c3 88
>
```

rio *<nodeaddress> <ioaddress> <number of bytes>*

This command reads a block of I/O data from node **nodeaddress**. The function reads consecutive bytes starting from the I/O port specified by **ioaddress**. The returned data is displayed to the user. This command can return up to 6 data bytes.

Note that this command behaves slightly differently to the standard RAC command in that it reads a block of data from consecutive addresses rather than reading data from (up to) 6 different addresses.

Example:
```
>rio 22 c3 4
```

Response (verbose ON):
```
Addr = C3H : Data = 55H
Addr = C4H : Data = 12H
Addr = C5H : Data = 88H
Addr = C6H : Data = FFH
>
```



```
Response (verbose OFF):
C3 FF
C4 FF
C5 FF
C6 FF
>
```

wxmem <*nodeaddress*> <*memaddr*> <*data...*>

This command writes an array of bytes to a block of external data memory on node **nodeaddress**, starting at address **memaddr**. This command can write up to 11 data bytes.

```
Example:
>wxmem 22 1000 11 22 33 44
>
```

rxmem <*nodeaddress*> <*memaddr*> <*number of bytes*>

This command reads a block of data from external data memory on node **nodeaddress**, starting at address **memaddr**. This command can read up to 11 data bytes.

```
Example:
>rxmem 22 1000 4

Response (verbose ON):
Addr = 1000H : Data = 11H
Addr = 1001H : Data = 22H
Addr = 1002H : Data = 33H
Addr = 1003H : Data = 44H
>

Response (verbose OFF):
1000 11
1001 22
1002 33
1003 44
>
```

wimem <*nodeaddress*> <*intaddr*> <*data...*>

This command writes to a block of indirectly addressable internal data RAM on node **nodeaddress**. The command writes data bytes starting at internal address **intaddr.** It can write up to 6 data bytes

Note that this command behaves slightly differently to the standard RAC command in that it operates on consecutive addresses rather than on (up to) 6 different addresses.

Example:
```
>wimem 33 30 10 11 12 13
>
```

rimem <*nodeaddress*> <*intaddr*> <*nbytes*>

This command reads from a block of indirectly addressable internal data RAM on node **nodeaddress**. The command reads **nbytes** starting at internal address **intaddr**. It can read up to 6 bytes of data.

Note that this command behaves slightly differently to the standard RAC command in that it operates on consecutive addresses rather than on (up to) 6 different addresses.

Example:
```
>rimem 33 30 4
```

Response (verbose ON):
```
Addr = 30H : Data = 10H
Addr = 31H : Data = 11H
Addr = 32H : Data = 12H
Addr = 33H : Data = 13H
>
```

Response (verbose OFF):
```
30 10
31 11
32 12
33 13
>
```

wcmem <nodeaddress> <memory address> <data...>

This command writes to a block of program memory on node **nodeaddress**. The command writes data bytes starting at internal address **intaddr.** It can write up to 11 data bytes

Example:
```
>wcmem 55 100 1 2 3 4
>
```

rcmem <*nodeaddress*> <memaddr> <nbytes>

This command reads from a block of code from program memory on node **nodeaddress**. The command reads **nbytes** starting at internal address **memaddr**. It can read up to 11 bytes of data.

Example:
```
>rcmem 55 100 4
```


Response (verbose ON):

```
Addr = 100H : Data = 1H
Addr = 101H : Data = 2H
Addr = 102H : Data = 3H
Addr = 103H : Data = 4H
>
```

Response (verbose OFF):

```
100 1
101 2
102 3
103 4
>
```

nodeinfo <*nodeaddress*>

This command returns device related information for node **nodeaddress**. The information displayed includes the processor type and software version**.**

Example:

```
>nodeinfo 66
```

Response (verbose ON):

```
Processor: i8044
Version:   21
Config:    1
Buffer:    20
>
```

Response (verbose OFF):

```
i8044
21
1
20
>
```


3.4 Error Reporting

Errors are returned from the commands in the form of error message strings with decimal error codes, or if verbose mode is switched off, only the error code is reported as E<*error code*>.

Errors may come from one of two sources. Serial Gateway errors may occur with the parsing of command strings and parameters, or with communication between local tasks on the Gateway module. Bitbus errors may occur any time messages are sent over the Bitbus network and are the standard errors defined in the Bitbus specification.

The likely error codes reported by the gateway are described in the following sections.

3.4.1 Serial Gateway Errors

-1 or 255 E_PARAM Invalid Parameter

This error indicates that the software encountered an invalid parameter in the command string. If this is reported, check that the parameters typed conform to the format required and are within the range indicated (especially numbers are in HEX and are not too large).

-8 E_SYNTAX Invalid Command

This error is reported when the gateway parser fails to recognise a command string. If this is reported, check that the command string entered, matches that shown in the manual description (especially the spelling).

3.4.2 Gateway Download File Errors

1	Protocol error received on send / receive.
2	Invalid hex character
3	Not enough memory for a buffer
5	Invalid byte count
6	Invalid record
7	Invalid checksum
8	Invalid code found
9	Invalid address
10	Timeout met on sending/receiving message
11	Checksum and byte total do not match

12 File not Intel hex format
13 File does not exist
14 Error opening file
15 Error reading file
16 Not enough memory
17 Overflow on node
18 Bank overflow on node
19 No buffers on node
20 RAC protected on node
21 No task at address

3.4.3 Bitbus Errors

128 E_EXIST Task specified by TaskID not found.

The task ID specified in the **dtask** command does not exist.

129 E_MAXTASKS Already running eight tasks

A **ctask** command was made for a node which is already running the maximum number of tasks (8).

130 E_REGS Unable to assign register bank to task

The system was unable to assign the requested register bank to a new task on **ctask.**

131 E_FID Function ID zero not allowed

A **ctask** command was made for a task with a function ID of zero.

132 E_BUFFER Unable to assign buffer space

The system was unable to assign the requested buffer space.

133 E_STACK Unable to assign stack space

The system was unable to assign the requested stack space to a new task on **ctask.**

134 E_ITD No ITD pattern, or ITD address was zero

A **ctask** command call was made for a task with a invalid Initial Task Descriptor (ITD).

145 E_PROTOCOL Protocol error

A Bitbus protocol error occurred. This is the usual error returned when a node cannot be reached on the Bitbus interconnect.

147 E_DEVICE No destination device

An attempt was made to send a command to an unknown destination node.

149 E_PROTECT RAC is protected

An attempt was made to send an I/O, read, write, upload or download command while the destination node was in RAC protect mode.

150 E_UNKNOWN Unknown RAC command

An attempt was made to send an unknown command to the RAC task.

4. Command Summary

4.1 Serial Gateway Commands

verbose	Toggle verbose mode ON/OFF
echo	Toggle echo mode ON/OFF
load	Download an Intel Hex file to module
smsg	Send and receive a raw Bitbus message
reset	Reset Bitbus node
off	Force Bitbus node offline
ctask	Create user task
dtask	Delete user task
get	Get function Id.s
protect	Set RAC protect mode
release	Release RAC protect mode
rio	Read I/O block
wio	Write I/O block
rxmem	Upload memory
wxmem	Download memory
wimem	Write internal data
rimem	Read internal data
rcmem	Upload code
wcmem	Download code
nodeinfo	Get node configuration

S

W

