Quin Systems Limited

BMS232C Hardware Reference Manual

MAN 604

Issue 2.3

April 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 to the protection requirements of the EMC Directive (89/336/EEC) as amended by Directives 92/31/EEC and 93/68/EEC. If this poduct 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:

- Microsoft Windows 3.0 and above
- Microsoft Visual Basic version 1 and above
- Wonderware Intouch version 3 and above
- Intel iDCM44 version 2.1
- Keil ‘C’ Compiler Version 3.0 and above
- Keil C Linker/Locator version 2.1 and above
- BMS232 Hardware Issue C

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

National Tel: 01734 - 771077 International Tel: +44-1734-771077
National Fax: 01734 - 776728 International Fax: +44-1734-776728
email: support@quin.co.uk
Compuserve: 100572,1631

ISSUE	*REVISION HISTORY*		*DATE*
0.1	Draft Copy	TAW	Jan 1990
1.1	Original issue 1.1 for Issue A hardware	TAW	July 1990
2.0	Reissue for Issue B/C hardware	TAW	Nov 1991
2.1	Mods. for SSL	TAW	Jan 1993
2.2	Update	TAW	Sept 1993
2.3	Amendments	IDH	Apr 1998

Contents

List of Figures

1. Introduction

This manual describes the operation and capabilities of the BMS232C Bitbus Serial module. This module is available in the following format:

BMS232C: In addition to the XBus2 interface provided by the BUSBUF daughterboard, this module provides 2, galvanically isolated, software configurable serial communications ports.

BMSG232C: In addition to the XBus2 interface provided by the BUSBUF daughterboard, this module also provides 2, galvanically isolated, software configurable serial communications ports.

The part number BMS232C will be used throughout this manual to describe the generic part and all information provided within this manual refers to the generic module unless specifically stated otherwise.

External References:

001 iDCX51 Distributed Control Executive
Users Guide for Release 2.0
Part number 460367, *Intel Corporation*
Release 001 04/87

002 8044 Bitbus Enhanced Microcontroller
Programmer's Summary
Part Number 462390-001, *Intel Corporation*
Release 001

003 The Bitbus Interconnect Serial Control
Bus Specification
Part Number 280645-001, *Intel Corporation*
Release 001 Rev B 7/88

004 BMS232 Programmers Reference Manual
MA0204
Quin Systems Ltd.

005 Bitbus DDE manual
for Windows 3.1
Quin Systems Ltd.

006 Bitbus DLL manual
for Windows 3.1
Quin Systems Ltd.

1.1 General

The BMS232C is a compact DIN rail mounted Bitbus module which provides the user with a fully configured interface between the Bitbus high-speed serial network and the XBus2 parallel expansion bus. The BMS232C module is based on the Intel 8044BEM microcontroller with 32K RAM and 32K EPROM which together form a complete microcomputer system. This system runs the iDCX51 real-time, multi-tasking operating system. Each BMS232C has a preprogrammed Remote Access and Control (RAC) task which provides all of the RAC functionality as required by the Bitbus Interconnect specification and thereby ensures that the BMS232C is fully compatible with other manufacturer's Bitbus products.

The BMS232C is designed for operation in the industrial environment. It complies with all relevant EU EMC Directives and utilises galvanic isolation to ensure a high degree of noise immunity from the local environment. The RAC task enables the BMS232C to be fully applied from PC resident programs which utilise a Bitbus compliant driver program. The use of the Quin SYK-PC-CON or IBC-PC-CON master card together, with the Quin Bitbus libraries, enables rapid system development using standard PC development tools and languages. A second, Watch-Dog, task can be provided which continuously monitors the operation of the module and will cause a hardware reset should the module cease functioning. The module is also designed to facilitate the programming and execution of custom firmware which can be programmed into EPROM or downloaded via Bitbus and programmed into available RAM. In this way the operation and general functionality of the module can be changed on network initialisation or during operation. All module resident firmware comprises tasks running under iDCM44. These on-module tasks can perform **local** control and data acquisition functions which are capable of maintaining control in the event of a network failure.

Two additional tasks, Serial Driver and Interrupt Handler, are provided to control the operation of the two high speed bidirectional serial ports. These tasks handle all low-level control of the ports and provide a high-level user-interface which facilitates simple configuration of protocols, handshaking etc. The reader is referred to the BMS232C Programmers Reference Manual (MA0204) for further details. The XBus2 expansion bus permits the user to connect a selection of up to five of the large range of Quin Bitbus expansion modules. It also facilitates the interfacing of custom modules to the Bitbus network with full technical support available from Quin Systems.

The BMS232C may be used either as a Bitbus master or slave node. Throughout the remainder of this manual it is assumed that it is being used as a slave device, for further information on using the BMS232C module as a master Bitbus device the reader is referred to the BMSG232C Serial Gateway Reference Manual (MAN606).

1.2 Rated Performance

This section details the rated performance of the BMS232C and BMSG232C modules.

Table 1: Rated Performance

Bitbus Mode	Max. Baud Rate	RS232/485 Ports	Max. Baud Rate
Synchronous	2.4 Mbaud	2	38.4 kbaud
Asynchronous	375 kbaud	2	38.4 kbaud
Asynchronous	62.5 kbaud	2	38.4 kbaud

The BMS232C is designed to maintain its rated performance without the need for any user adjustments. This is achieved through the use of high precision, high stability components.

1.3 Physical Dimensions

The standard module is provided with D-Type connectors for all serial communications links. A two-part two-pin combicon connector is provided for application of the 24Vdc power supply. This facilitates pre-wiring of harnesses etc. prior to installation of the module. The table below shows the space required for installation.

Length	240mm
Width	77mm (raft base only)
Height	40mm (from top of DIN rail)
Weight	0.4 kg

Table 2: Module Dimensions

1.4 Power Supply Requirements

The power supply for the module is derived from the main 24Vdc supply via a 5W dc/dc converter which provides galvanic isolation of the 24V and regulation down to 5Vdc. The BMS232C only requires 50% of the converter's output capacity, this leaves a total of 2.5W for powering expansion modules connected to the XBus2 connector.

1.5 Environmental Specification

The table below shows the operating and storage environmental limits:

Temperature Ranges
Rated Performance 0 to 50 C
Storage -25 C to 70C
Rel Humidity 0 to 95% at 50 C

Table 3: Temperature Ranges

If necessary, the unit can be supplied in a suitable sealed cabinet. Please contact your sales office or Quin Systems directly for further details.

1.6 Conformity to EU Directives

If this product is 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 conformity with the Machinery Directive 89/392/EEC and/or its relevant amendments. These installation instructions should be followed in constructing a system which meets requirements.

This product has been tested to the protection requirements of the EMC Directive (89/336/EEC) as amended by Directives 92/31/EEC and 93/68/EEC. This product only uses low voltages, and is therefore exempt under the Low Voltage Directive (73/23/EEC) as amended by 93/68/EEC.

The 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.

2. Description of Operation

2.1 General

This section summarises the operation of the Bitbus Serial module. The following is a block diagram of the BMS232C.

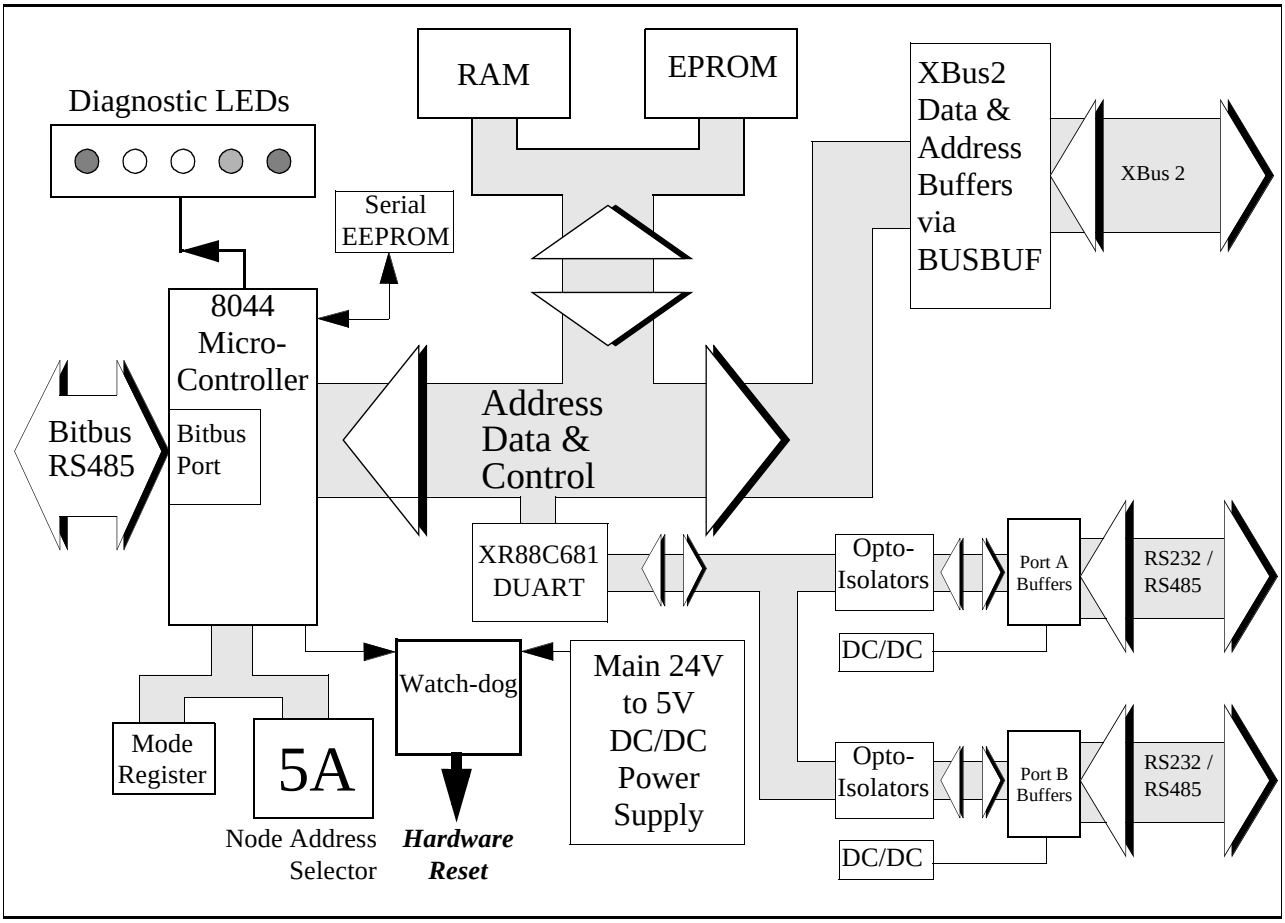


Figure 1. BMS232C: Block Diagram

The BMS232C module has its own processor and memory which form a complete microcomputer system. It is a complete intelligent Bitbus node, designed to execute local tasks. These task can be written to perform the low level serial data gathering and processing such as protocol conversion, for passing up to any master Bitbus Node, such as the IBC-PC-CON or SYK-PC-CON. The module may be used either as a Bitbus master or slave node.

The serial DUART device is mapped directly into the processor's Memory mapped I/O space, giving the application full flexibility in accessing the serial hardware if required. For the user's convenience, there is a serial driver task which gives the application task a much friendlier front end, and both hardware and software handshake control.

The local 8044 processor runs the iDCX51 operating system, providing support for up to eight concurrent tasks. Application software can be programmed into a 32k×8 EPROM on the module. A 32k×8 static RAM is used for local workspace memory, message buffers, data, etc. This memory has been arranged such that both the EPROM and SRAM appear in both the 8044's code and data space. This allows both the formation of read-only data areas and the ability to down-load executable code from the master node. In this way the operation and general function of the module can be changed while in operation.

The 8044BEM has an inbuilt 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 is in addition galvanically isolated from the remainder of the module.

The unit also possesses watch-dog circuitry which will invoke a full hardware reset should the Watch-Dog task fail to toggle the input once every 1.5 seconds. This watch-dog facility may be enabled or disabled by a jumper link.

2.2 iDCX Operating System

The iDCX Distributed Control Executive is a real-time, multi-tasking nucleus that operates on the MCS-51 family of micro-controllers, including the 8044BEM. iDCX51 provides all the supervisory services for real-time control systems that involve multiple control functions. For more information, please refer to [001] "iDCX51 Distributed Control Executive Users Guide for Release 2.0"

Within this programming environment, the individual control functions are classed as ‘tasks’ and eight such tasks can be active in the iDCX environment at any one time.

The Bitbus product line uses the 8044 Bitbus Enhanced Microcontroller running a pre-configured iDCX51 executive iDCM44. Communication between Bitbus nodes is achieved using the inbuilt serial interface unit (SIU) of the 8044BEM which provides all protocol and buffering operations for the Bitbus interconnect.

iDCM44 includes a preconfigured task 0 called the Remote Access and Control task (RAC). This task provides all the communications management and remote access and control which provides the application programmer with the greatest flexibility in writing tasks, as there is no distinction between external and internal Bitbus messages.

2.3 Watchdog Facility

The watchdog provides a mechanism by which the operation of the system is monitored to detect failures in either hardware or software. It is enabled (in hardware) by fitting a jumper link. The watchdog operates by means of a fixed timeout period, approximately 1.6 seconds, and if it is not triggered within this time it performs a hardware reset of the system.

The watchdog hardware is controlled by one otherwise unused output line from the 8044 processor. The Port line P1.2 is the watchdog trigger. When the watchdog is enabled, the watchdog trigger signal on P1.2 must change state at least once within each successive timeout period to prevent the system being reset by the watchdog.

The watchdog on the BMS232C module has an associated red LED :indicator in the located close to the watchdog jumper. This gives user confidence that the watchdog task is running. This LED may be used as a general purpose output indicator when the watchdog is not being used.Serial Interface.

The serial interface device consists of the XR88C681 dual channel UART device. This CMOS peripheral chip is pin and function compatible with the industry standard 2681 device and occupies 16 locations in the i8044’s dedicated I/O space above $FFF0 in external data space. With the I/O located here, it is possible to access the I/O locations both from local, on-module tasks, and remotely via the built in RAC commands.

The serial interface is available to the user in the form of two independent ports, which are individually configurable via appropriate jumpers and driver chips to be RS232, RS422 or RS485 compliant.

2.4 Diagnostic LEDs

To facilitate diagnosis of system faults the BMS232C is provided with five diagnostic LEDs (D1...D5). The operation of these LEDs is discussed below:

D1 & D3 These two LEDs (D1 green, D3 red) are used during system initialisation to indicate the state of various system components as shown in the following table

Table 4: Power-Up Test Sequence

State	D3 Red	D1 Green
Initialising	On	On
Test Start	Off	Off
Instruction Set	On	On
ROM Checksum	On	Off
Internal RAM	Off	Off
External RAM	Off	On

Once the module has booted these LEDs are available for use by custom applications.

D4 & D5 These yellow LEDs are illuminated while the module is transmitting Bitbus messages.

D3 The red WDOG LED indicates the state of the WDI input to the LTC690. It is illuminated when the WDI input is a logic ‘0’. When the watch-dog is enabled the WDI input must be toggled every 1.6 seconds to prevent a hardware reset. This LED would typically be flashed at a rate of approximately 1Hz.

3. Using the BMS232C Within the Bitbus Environment

3.1 Introduction to Bitbus

Bitbus is a high performance, serial Fieldbus network designed for communication between distributed process I/O and intelligent controllers. It uses the IEEE-1118 international open standard for the Bitbus Interconnect as specified by the Intel Corporation in 1984.

The IEEE1118 standard specifies the electrical and data link protocols for the network, along with the application message structure and protocol for use with a multi-tasking operating environment such as iDCX51/iDCM44.

The BMS232C is primarily intended to be used as a slave Bitbus node which is connected to a number of expansion modules via the XBus2 expansion bus. The BMS232C would typically form part of a distributed Bitbus network which would include a PC resident Master card as shown in the following figure:

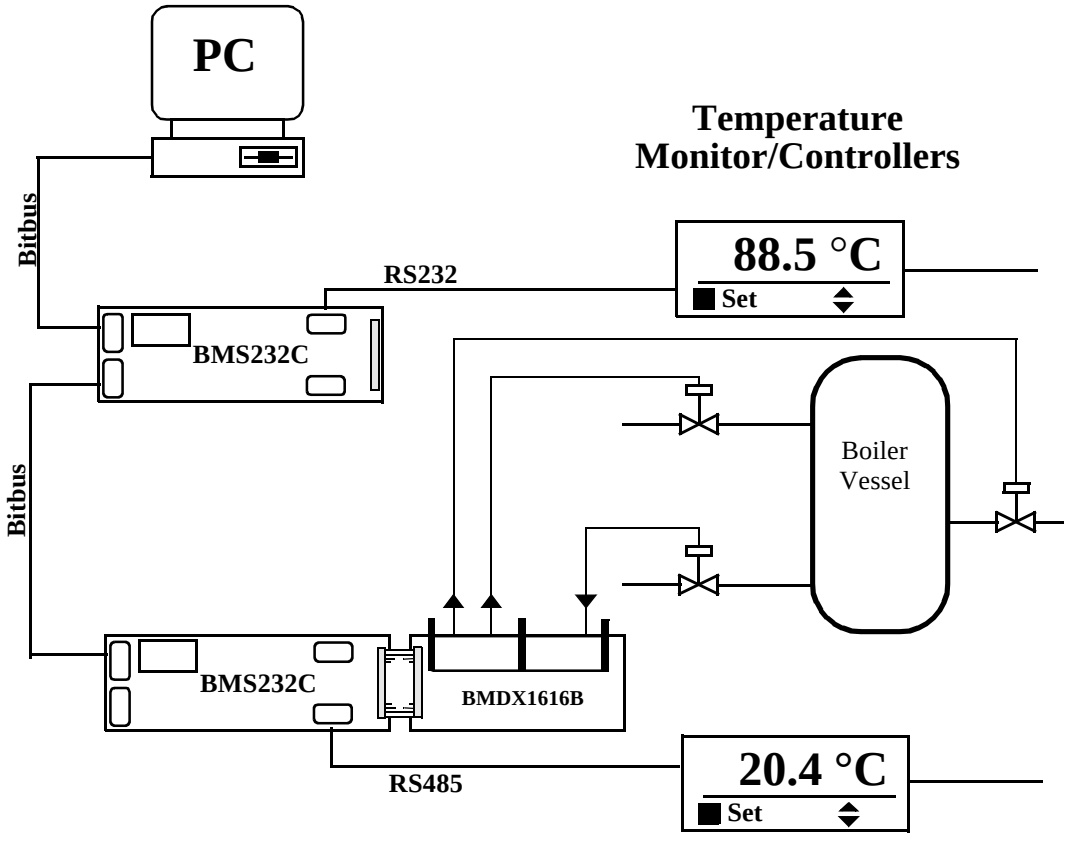


Figure 2. Typical BMS232C Installation

3.1.1 Physical Characteristics

The network uses a simple multi-drop bus configuration, with the standard RS-485 (twisted pair) specification cable link. This standard has been thoroughly tested and proven for industrial communications and is currently in world-wide use.

3.1.2 Modes of Operation

Bitbus supports two modes of operation, synchronous for very high speed performance and asynchronous (or self-clocked) for operation over large distances.

Synchronous mode of operation allows Bitbus to operate at speeds of 500kbits/sec or 2.4Mbits/sec, with a maximum of 28 nodes distributed over a physical distance of 30 metres.

Self-clocked operation allows Bitbus to be used for much larger distances, with the ability to link together up to 250 nodes over a distance of up to 13.2km. In this mode Bitbus can use multiple segments, linked via repeaters, to achieve a range of 13.2 km at 62.5kbits/sec or 3.3 km at 375kbits/sec.

3.1.3 Data Link Protocol

Bitbus uses a highly reliable Master-Slave protocol. This uses a single master node to control all communication on the network. Each message transaction is initiated by the master node by sending an order to a specified slave, then polling for a reply. Slaves only send messages on the network in response to an order from the master.

3.1.4 Message Specification

The Bitbus protocol uses a similar structure for both order and reply messages. The master starts a transaction by issuing an order to a slave. The slave must then respond with a reply back to the master. The application message structure is the same for all Bitbus communications and is shown in the following figure:

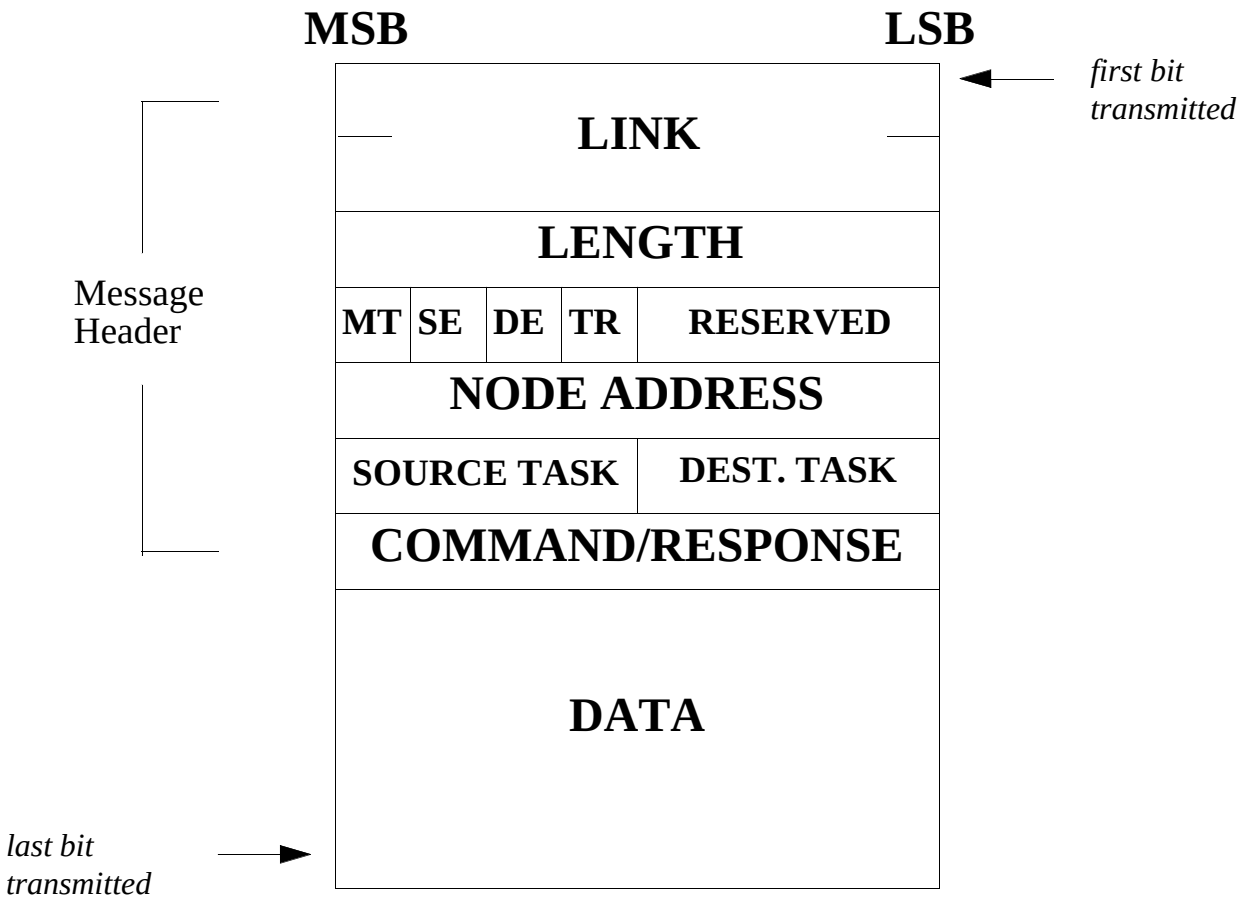


Figure 3. Bitbus Message Structure

The contents of the Bitbus message fields are explained below:

Link:	Reserved 2 byte field, required for compatibility with the Bitbus specification. Normally set to 0000H.
Length:	Length of the Bitbus message, number of data bytes in the message plus the 7 bytes in the header (range 7 to 255). Maximum length is FFH
Message Type (MT):	Set to 0 (zero) for an order message and 1 (one) for a reply. Set to 0 for orders from Master. Set to 1 by Slave on reply.
Source/Dest Extension: (SE/DE)	Indicates whether or not the message has come from, or is destined for, an extension module connected to the source or destination node. Set to 1 for an extension or 0 (zero) otherwise. Set to 1 for use with Quin Master cards.

Track (TR):	Used by some implementations for message control. If used it is set to 0 (zero) on sending and 1 on receiving. Set to 0 for use with Quin Master cards.
Reserved:	Reserved for possible future enhancements. Cleared on send, unknown on receive. Set to 0 for use with Quin Master cards.
Node Address:	Destination node for orders, source node for replies, i.e. it doesn't change during the message transaction. In the range 1-250. Set to zero for intertask messages on BMS232C. Set to 01H...FAH to access slave nodes on network or set to FFH to access a Quin master card from a PC resident task.
Source Task:	The task that initiated the message transaction or that will receive the reply (depending on the value set in the Message Type field). Set to 1 for use with Quin Master cards.
Destination Task:	The task that will receive the original order message or the source of the reply message. Set to 0 to access RAC functions or 1..7 to access other module-resident tasks.
Command / Response:	This field is normally used by user tasks to identify the message within the overall user application, i.e. read, write, update command message. It is also used by the Bitbus message protocol in message responses to indicate errors that have occurred. See following section for detailed information on the contents of this field.
Data:	The Data field contains any data specific to the Bitbus message and is defined by the command field. This field is the only optional field in the message and can be in the range from 0 bytes to 248 bytes long, dependent on implementation. The Bitbus specification requires that every Bitbus node be able to accept a message length of 20 bytes, ie. 13 bytes of data, however, the maximum permitted message length is 255 bytes giving a maximum data field size of 248 bytes. See following section for detailed information on the contents of this field.

3.1.5 Remote Access and Control (RAC) Commands

The Bitbus specification defines a set of high level commands (and associated responses) to carry out various general purpose operations on remote slave nodes. This interface, the Remote Access and Control (RAC) command interface, provides a standard means of interrogating slave nodes across all Bitbus network implementations.

The remote access interface provides the user with general I/O read & write facilities, together with memory upload and download capability. The remote control functions allow control and monitoring of tasks at remote slaves.

When supported, the RAC interface is always implemented as task 0 (zero) on the slave node and makes use of the standard Bitbus message protocol. This enables RAC commands to be implemented as standard on all Bitbus nodes. The functions specified and supported by RAC are shown in the following table:

Table 5: Bitbus RAC Commands

Command	*Access*	*Control*	*Command /Response*	*Action Taken by Task 0*
Reset		*	00H	Perform software reset
Create Task		*	01H	Start a module-resident task
Delete Task		*	02H	Delete a module-resident task
Get Function IDs		*	03H	Return IDs of all running tasks
RAC Protect		*	04H	Suspend/Resume Access services
Read I/O	*		05H	Return data from specified I/O ports
Write I/O	*		06H	Write to specified I/O ports
Update I/O	*		07H	Update specified I/O ports
Upload Memory	*		08H	Return values from specified memory area
Download Memory	*		09H	Write values to specified memory area.
OR I/O	*		0AH	OR values into specified I/O ports
AND I/O	*		0BH	AND values into specified I/O ports
XOR I/O	*		0CH	XOR values into specified I/O ports
Read Internal	*		0DH	Read values from specified ‘internal’ memory
Write Internal	*		0EH	Write values to specified ‘internal’ memory
Node Information	*		0FH	Return module related information.
Offline		*	10H	Set remote node offline
Upload Code	*		11H	Read values from code memory space
Download Code	*		12H	Write values to specified code space

3.2 iDCM44 Remote Access and Control (RAC) Task Summary

The Remote Access and Control (RAC) service, contained in task 0, allows resources on Bitbus nodes interconnected via a Bitbus network, to be accessed via the standard Bitbus message interface. A RAC service request is made by sending an order message to task 0 on the node at which the service is to be performed. RAC order messages are standard Bitbus messages that have the COMMAND/RESPONSE field set to indicate which RAC service is to be performed. Any additional required information is sent and received in the DATA field.

NOTE: The qDCX302 RAC task, which is provided on the newer MC68302 based BMC302B, provides exactly the same functionality as the iDCM44 RAC commands contained within task 0 included on the 8044BEM processor.

3.3 Quin Bitbus Libraries

The simplest way of utilising the BMS232C on a Bitbus network is to connect the network to a PC resident master card. If this master card is compatible with the Intel standard, eg the Quin SYK-PC-CON or IBC-PC-CON master card, the network can be controlled using the high-level functions provided within the Quin Bitbus Libraries. These functions support all of the standard RAC commands available under iDCX51/ iDCM44. Use of the Quin libraries relieves the programmer of most of the protocol and error handling thus ensuring maximum productivity and a robust system design. A user-friendly monitor program is available which allows the system designer to manually exercise the RAC functions on the BMS232C and all other makes of Bitbus node.

3.4 BMS232C Memory Model

The address map for the 8044 processor is shown in the diagram below. The local EPROM and RAM memories are mapped into the code and data spaces of the 8044 address map. This allows either code or data to be stored in either type of memory for maximum flexibility.

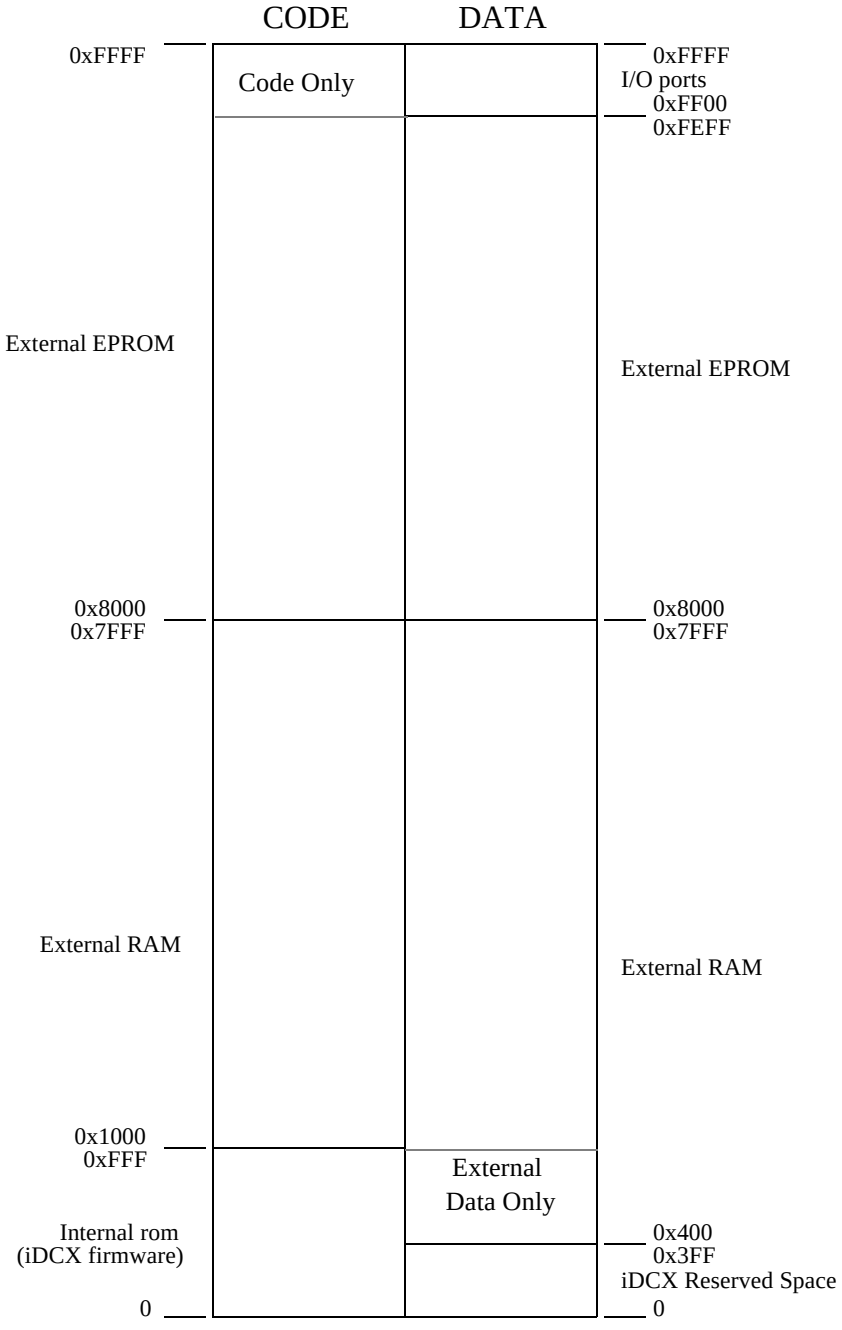


Figure 4. 8044 Address Map

3.5 I/O Address Space

The processor I/O space is shown below.

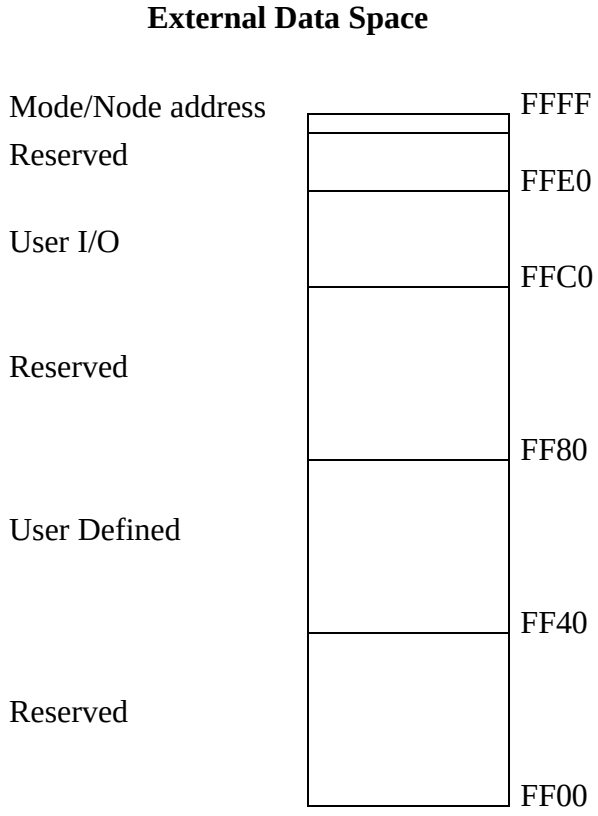


Figure 5. Processor I/O Address Space

This I/O address space can be easily accessed by reading and writing to the equivalent external data memory address allocated to this top page which is $FF00 and above.

The iDCX operating system together with iDCM44 require the reserved addresses for such things as SBX port addresses (SBX is the Intel standard method of host computers talking to Bitbus master nodes.)

The Mode register and node address register are located at the top of the external data memory, and in reality these are hardware latches and switches which the processor reads by accessing these locations. Due the nature of these hardware latches, a write to these locations has no effect.

3.6 Remote Access and Control (RAC) Command Reference

This section explains the functions performed by each of the iDCM44 standard RAC commands which are identical to those in the Quin qDCX302 operating system. All memory mapping required to translate I/O and memory locations to the 8044BEM memory map (figures 4 & 5) are carried out within the RAC application and are transparent to the user.

3.6.1 RESET STATION (00H)

Description

This RAC service performs a software reset (causes jump to restart vector) on the node at the specified address. No reply is returned, unless there is a general error with the operation.
NOTE: An Offline command should be issued following a Reset Station to ensure that the Master resynchronises with the reset slave node.

Input Parameters

Command/Response = 00H

Output Parameters

None. If the operation was carried out successfully.
Command/Response = 91H. If node didn't exist.

3.6.2 CREATE TASK (01H)

Description

This RAC service creates (runs) the task whose Initial Task Descriptor (ITD) resides at the address given in the DATA field. Status is returned in the reply message Command/Response field.

Input Parameters

Command/Response = 01H
Data(0) = high byte ITD address
Data(1) = low byte ITD address

Output Parameters

Command/Response = valid response from create task system call,
or
= 81H-86H If create task failed.
= 91H Protocol error during message transfer

An error is reported if no ITD is found at the specified address.

3.6.3 DELETE TASK (02H)

Description

This RAC service deletes (kills) a running task which has the task ID given in the Data field. Status is returned in the reply message Command/Response field.

Input Parameters

Command/Response = 02H
Data(0) = task ID (0..7)

Output Parameters

Command/Response = 0 The task was successfully deleted
= 80H the specified task did not exist
= 91H Protocol error during message transfer

3.6.4 GET FUNCTION ID (03H)

Description

This RAC service returns the Function IDs for all currently running tasks. The FID table is returned in the reply message Data field.

Input Parameters

Command/Response = 03H
Data (0 to 7) = An 8 byte array to allow buffer space for reply. The data values contained in the array are of no consequence.

Output Parameters

Command/Response = 0 Successful command execution
= 91H Protocol error during message transfer
Data (0 to 7) = FID table for currently running tasks. Any unused element is set to 0 by this RAC service.

3.6.5 RAC_PROTECT (04H)

Description

This RAC service disables or enables certain RAC services. If a disable RAC Protect message is sent, further RAC services at this node are suspended until a re-enable RAC Protect message is sent. If an attempt is made to invoke a disabled RAC service a value of 95H is returned in the Command/Response indicating that it is currently RAC protected.

Input Parameters

Command/Response = 04H
Data(0) = 0 to disable RAC services or
= 1 to re-enable RAC services

Output Parameters

Command/Response = 0 Successful command execution
= 91H Protocol error during message transfer

3.6.6 READ IO (05H)

Description

This RAC service reads the specified I/O ports and returns their values. I/O port references and returned I/O bytes are stored in a structured array in the Data field.

Input Parameters

Command/Response = 05H

Data(0) = a byte offset value, relative to the base I/O space = FF00H, that provides a reference to the I/O location to be read.
Data(1) = an 8-bit place holder where the returned I/O byte will be placed.
Data (len - 8) = Last I/O reference in Data field.
Data (len - 7) = Last data byte placeholder.
NOTE: The maximum number of I/O ports that can be read at one request is determined by the size of the data field which is set by the message length less the 7 bytes for the header.

Output Parameters

Command/Response = 0 Successful command execution
= 91H Protocol error during message transfer
= 95H Command presently RAC protected
Data() = same as shown in "Input Parameters" above but with the read I/O data contained in the relevant 8-bit databyte place holder.

3.6.7 WRITE IO (06H)

Description

This RAC service writes to the specified I/O ports. I/O port reference and write byte are stored in a structured array in the Data field.

Input Parameters

Command/Response = 06H
Data(0) = a byte offset value, relative to the base I/O space = FF00H, that provides a reference to the I/O location to be written.
Data(1) = an 8-bit value to be written to the preceding I/O port.
Data(len - 8) = Last I/O reference in Data field.
Data(len - 7) = Last byte to be written.
NOTE: The maximum number of I/O ports that can be written at one request is determined by the size of the data field which is set by the message length less the 7 bytes for the header.

Output Parameters

Command/Response = 0 Successful command execution
= 91H Protocol error during message transfer
= 95H Command presently RAC protected

3.6.8 UPDATE IO (07H)

Description

This RAC service writes to the specified I/O ports and then rereads and returns the read values. I/O port reference and write byte are stored in a structured array in the Data field. The read I/O byte is returned in place of the write byte in the reply message.

Input Parameters

Command/Response = 07H
Data(0) = a byte offset value, relative to the base I/O space = FF00H, that provides a reference to the I/O location to be read/written.
Data(1) = an 8-bit value to be written to the preceding I/O port. Upon return, this field will contain the value read from the I/O port.
Data(len - 8) = Last I/O reference in data field.
Data(len - 7) = Last byte to be written.

NOTE: The maximum number of I/O ports that can be updated at one request is determined by the size of the data field which is set by the message length less the 7 bytes for the header.

Output Parameters

Command/Response = 0 Successful command execution
= 91H Protocol error during message transfer
= 95H Command presently RAC protected
Data() = same as shown in "Input Parameters" above but with the write bytes replaced by the returned I/O bytes.

3.6.9 UPLOAD MEMORY (08H)

Description

This RAC service reads a specified number of bytes from external memory starting at a specified address. The message length field minus 9 bytes for the message header and starting address determines the number of bytes read. Values read are returned in the reply message.

Input Parameters

Command/Response = 08H
Data(0) = High byte of starting address.
Data(1) = Low byte of starting address.

Output Parameters

Command/Response = 0 Successful command execution
= 91H Protocol error during message transfer
= 95H Command presently RAC protected
Data(2 to (length - 9))= An array of data bytes read. The number of bytes returned depends upon the message length specified minus 9 bytes for the message header and starting address.

NOTE: The maximum number of memory locations that can be uploaded at one request is determined by the size of the available data field which is set by the message length less the 7 bytes for the header and the 2 bytes for the starting address.

3.6.10 DOWNLOAD MEMORY (09H)

Description

This RAC service writes a specified number of bytes to external memory starting at a specified address. The value of message length minus 9 bytes for the message header and starting address determines the maximum number of bytes that can be downloaded.

Input Parameters

Command/Response = 09H
Data(0) = High byte of starting address.
Data(1) = Low byte of starting address.
Data(2 to (length - 9))= An array of data bytes to be written.

NOTE: The maximum number of memory locations that can be downloaded at one request is determined by the size of the available data field which is set by the message length less the 7 bytes for the header and the 2 bytes for the starting address.

Output Parameters

Command/Response = 0 Successful command execution

= 91H Protocol error during message transfer
= 95H Command presently RAC protected

3.6.11 OR IO (0AH)

Description

This RAC service performs one or more OR operations on the specified I/O ports, with specified OR masks, then rereads the ports, and returns the values read. I/O port reference and OR mask byte are stored in a structured array in the Data field. The byte read is returned in place of the OR mask in the reply message.

Input Parameters

Command/Response = 0AH
Data(0) = a byte offset value, relative to the base I/O space = FF00H, that provides a reference to the location to perform the OR operation.
Data(1) = an 8-bit value to be ORed with the specified I/O port. Upon return, this field will contain the value read from the I/O port.
Data(len - 8) = Last I/O reference in Data field.
Data(len- 7) = Last OR mask.

NOTE: The maximum number of I/O ports that can be ORed at one request is determined by the size of the data field which is set by the message length less the 7 bytes for the header.

Output Parameters

Command/Response = 0 Successful command execution
= 91H Protocol error during message transfer
= 95H Command presently RAC protected
Data() = same as "Input Parameters" above but with the ORed data returned in place of the OR masks.

3.6.12 AND IO (0BH)

Description

This RAC service performs one or more AND operations on the specified I/O ports, with specified AND masks, then rereads the ports, and returns the read values. I/O port reference and AND mask byte are stored in a structured array in the Data field. Read byte is returned in place of the AND mask in the reply message.

Input Parameters

Command/Response = 0BH
Data(0) = a byte offset value, relative to the base I/O space = FF00H, that provides a reference to the I/O to perform the AND operation.
Data(1) = an 8-bit value to be ANDed with the specified I/O port. Upon return, this field will contain the value read from the I/O port.
Data(len - 8) = Last I/O reference in Data field.
Data(len - 7) = Last AND mask.

NOTE: The maximum number of I/O ports that can be ANDed at one request is determined by the size of the data field which is set by the message length less the 7 bytes for the header.

Output Parameters

Command/Response	=	0 Successful command execution
	=	91H Protocol error during message transfer
	=	95H Command presently RAC protected
Data()	=	same as shown in "Input Parameters" above but with the ANDed data returned in place of the AND masks.

3.6.13 XOR IO (0CH)

Description

This RAC service performs one or more XOR operations on the specified I/O ports, with specified XOR masks, then rereads the ports, and returns the read values. I/O port reference and XOR mask byte are stored in a structured array in the Data field. The read byte is returned in place of the XOR mask in the reply message.

Input Parameters

Command/Response	=	0CH
Data(0)	=	a byte offset value, relative to the base I/O space = FF00H, that provides a reference to the I/O to perform the XOR operation.
Data(1)	=	an 8-bit value to be XORed with the specified I/O port. Upon return, this field will contain the value read from the I/O port.
Data(len - 8)	=	Last I/O reference in Data field.
Data(len - 7)	=	Last XOR mask.

NOTE: The maximum number of I/O ports that can be XORed at one request is determined by the size of the data field which is set by the message length less the 7 bytes for the header.

Output Parameters

Command/Response	=	0 Successful command execution
	=	91H Protocol error during message transfer
	=	95H Command presently RAC protected
Data()	=	same as shown in "Input Parameters" above but with the XORed data returned in place of the XOR masks.

3.6.14 READ INTERNAL (0DH)

Description

This RAC service reads the specified internal memory locations and returns their values. Internal memory location and the byte read are stored in a structured array in the Data field.

Input Parameters

Command/Response	=	0DH
Data(0)	=	a byte value that specifies the internal memory location to be read.
Data(1)	=	a byte value place holder where the read byte is placed.
Data(len - 8)	=	Last internal memory location.
Data(len - 7)	=	Last data byte placeholder.

NOTE: The maximum number of internal memory locations that can be read at one request is determined by the size of the data field which is set by the message length less the 7 bytes for the header.

Output Parameters

Command/Response	=	0 Successful command execution
	=	91H Protocol error during message transfer
	=	95H Command presently RAC protected
Data()	=	same as shown in "Input Parameters" above but with the read data returned in the place holders.

.

3.6.15 WRITE INTERNAL (0EH)

Description

This RAC service writes to the specified internal memory locations. Internal memory location and write byte are stored in a structured array in the Data field.

Input Parameters

Command/Response	=	0EH
Data(0)	=	a byte value that specifies the internal memory location to be written.
Data(1)	=	a byte value to be written to the specified internal memory location.
Data(len - 8)	=	Last address in Data field.
Data (len - 7)	=	Last byte to be written.

NOTE: The maximum number of internal memory locations that can be written at one request is determined by the size of the data field which is set by the message length less the 7 bytes for the header.

Output Parameters

command/response	=	0 Successful command execution
	=	91H Protocol error during message transfer
	=	95H Command presently RAC protected

3.6.16 NODE INFO (0FH)

Description

This RAC service retrieves device specific information about the specified node. The message length must be set to 20 to allow buffer space for this information.

Input Parameters

Command/Response = 0FH

Output Parameters

Command/Response	=	0 Successful command execution
	=	91H Protocol error during message transfer
Data(0 - 5)	=	A six byte array in which an ASCII identifier string is placed. For Quin Bitbus products, the string will be 'QSL302'.
Data(6 - 7)	=	A two byte array specifying the ASCII firmware version number. Production versions of the firmware will be from '10' to '99', indicating versions 'V1.0' through 'V9.9'.
Data(8)	=	Byte value that is read from the node configuration jumpers. A BMS232C node will have one of the following values:- 0 = synchronous Bitbus mode 1 = 375Kbit asynchronous Bitbus mode (default)

3 = 62.5Kbit asynchronous Bitbus mode

Data(9) = A byte value that indicates the maximum message buffer size supported by this node. This is equal to rqbufsize (20 bytes for a standard Bitbus implementation).

3.6.17 OFFLINE (10H)

Description

This RAC function sets the designated node offline, meaning that the next command message sent to that node will cause resynchronisation with that node. This command is only required when the master node needs to resynchronise with a slave.

Input Parameters

Command/Response = 0H
Data(0) = Node to be set offline.

Output Parameters

None

3.6.18 UPLOAD CODE (11H)

Description

This RAC service reads a specified number of bytes from code memory space starting at a specified address. The message length field minus 9 bytes for the message header and starting address determines the number of bytes read. Values read are returned in the reply message.

Input Parameters

Command/Response = 11H
Data(0) = High byte of starting address.
Data(1) = Low byte of starting address.

Output Parameters

Command/Response = 0 Successful command execution
= 91H Protocol error during message transfer
= 95H Command presently RAC protected
Data(2 to (length - 9))= An array of data bytes read.

NOTE: The maximum number of code locations that can be uploaded at one request is determined by the size of the available data field which is set by the message length less the 7 bytes for the header and the 2 bytes for the starting address.

3.6.19 DOWNLOAD CODE (12H)

Description

This RAC service writes a specified number of bytes to code memory space starting at a specified address. The value of message length minus 9 bytes for the message header determines the maximum number of bytes written.

Input Parameters

Command/Response = 12H
Data(0) = High byte of starting address.
Data(1) = Low byte of starting address.

Data(2 to (length - 9))= An array of data bytes to be written.
NOTE: The maximum number of code locations that can be downloaded at one request is determined by the size of the available data field which is set by the message length less the 7 bytes for the header and the 2 bytes for the starting address.

Output Parameters

Command/Response = 0 Successful command execution
= 91H Protocol error during message transfer
= 95H Command presently RAC protected

4. Hardware Configuration

4.1 General.

Please refer to Figure 12 for PCB layout diagram.

The Bitbus interface is capable of running at the two standard asynchronous data rates of 62.5 Kbaud and 375Kbaud, and the synchronous rate of 2Mbaud. The synchronous rate is obtained from the processors ALE clock signal.

The interface has been designed such that the Bitbus network will run as supplied with only one twisted pair cable and screen. This is assuming that the configuration register and Bitbus options jumpers are set to the default. That is running at 375Kbaud, asynchronous and not as a repeater.

When the above default settings are used, then a recommended cable could be a BICC-VERO cable part number H8082(screened twisted pair) or the “Twinax” standard (twisted pair co-ax) obtainable from Farnell Electronic Components.

4.2 Configuration Register: J4

Jumper J4 is the Bitbus configuration register. Its function is described more fully in the Intel data sheet for the 8044 Bitbus Enhanced Microcontroller, but it is described briefly here for completeness.

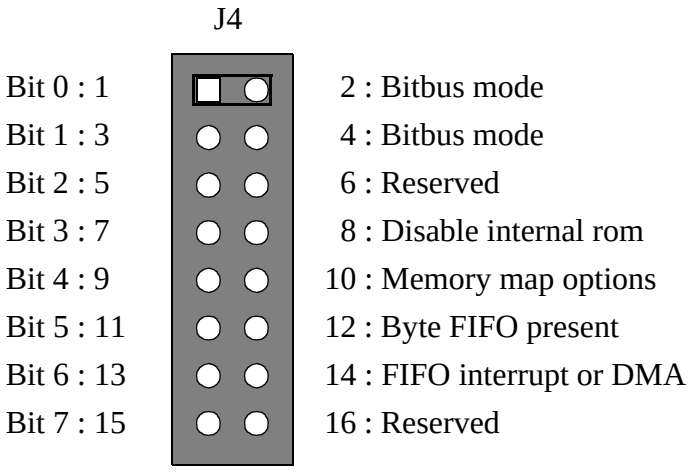


Figure 6. Configuration Register: J4

Note that only bits 0 and 1 should be used as the other facilities are not available. The selection of the Bitbus operating mode by the jumper links fitted in bit positions 0 and 1 is as follows.

Table 6: Bitbus Mode Configuration

Bit 0	Bit 1	Bitbus Mode
Open	Open	Synchronous
Linked	Open	Self Clocked 375 kb/s
Open	Linked	Reserved
Linked	Linked	Self Clocked 62.5 kb/s

4.3 Bitbus Interface Options: J2

Jumper J2 is used to configure the Bitbus interface for different options. A jumper link fitted between pins 1 and 2 enables the RTS output signals for use with repeaters, and a link fitted between pins 3 and 4 enables synchronous mode operation.

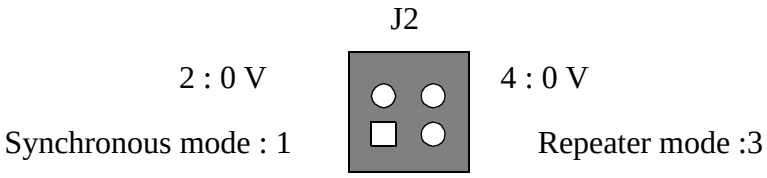


Figure 7. Bitbus Interface Options: J2

4.4 Watchdog enable: J5

Jumper J5 is used to enable or disable the watchdog facility. If J5 is removed, the watchdog is disabled.

Associated with this watchdog enable jumper, and labelled as D4 is the watchdog LED. This is directly connected to the signal line toggling the watchdog, albeit through a buffer. This gives an indication that the processor is at least toggling the watchdog. If the toggling is very fast, e.g. the processor is not doing much other than toggling the watchdog, the LED may appear to be continuously on or off; this is due to the mark/ space ratio of flashing and does not represent an error condition.

4.5 RS232/422/485 Configuration

The BMS232C is capable of sending and receiving characters in various different forms, being RS232 (+12V Single Ended), RS422 (Differential) and RS485 (Differential Multi Drop). In order to switch between these standards, in addition to the following jumper position changes, the relevant driver chips must be fitted. These are as follows:

Table 7: RS Transceiver Requirements

Standard	IC's Required		
	U28	U29	U30
RS232	N/F	14C88 RS232 Driver	34C86 RS422 Receiver
RS422	34C87 RS422 Driver	N/F	34C86 RS422 Receiver
RS485	75174 RS485 Driver	N/F	75175 RS485 Receiver

4.5.1 RS232 Pullup Jumper Block: J8

The J8 jumper block is used to pull the individual RS232 complement lines high allowing the use of the standard RS422 receiver for RS232 communications. These jumpers should be removed for operation with RS422 or RS485 in hardware handshake mode.

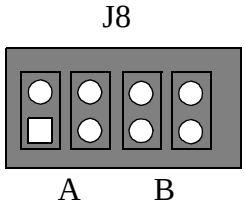


Figure 8. RS232 Pullup Jumper Block.

4.5.2 Transmitter line select J7

J7 is used to select which transmitter circuit is used for each channel. These settings are channel independent. The Tx and CTS lines are configurable, depending on wether RS232 or RS422/485 is being used.

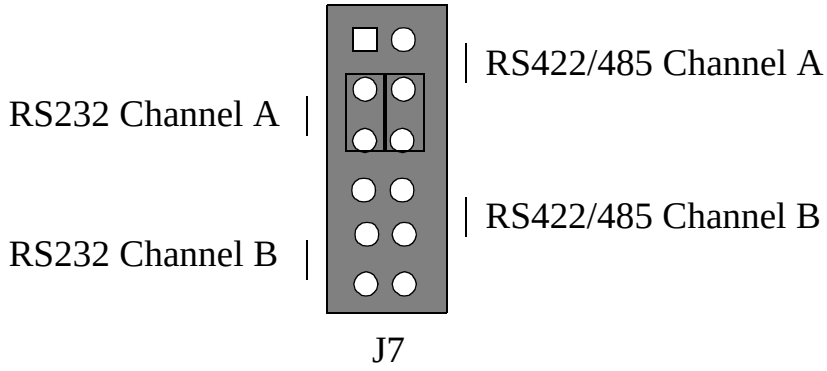


Figure 9. J7 Transmitter Line Selector

4.5.3 RTS/CTS Tristate Control Jumper J6

This jumper is used when it might be necessary to tristate (i.e. use multidrop) the RS422/485 Serial connections. This jumper when fitted prevents the transmitters being permanently enabled, as is the case in point to point serial communications.

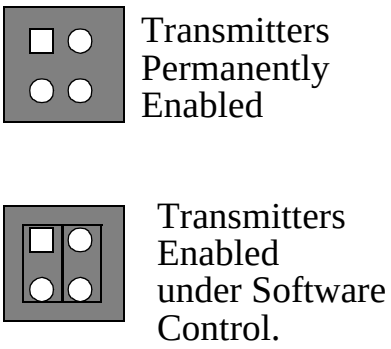


Figure 10. Transmitter Tristate Control Jumper J6

4.6 Node Address: SW1, 2

The Bitbus node address for the BMS232C is set by means of two rotary hexadecimal coded switches. This allows the node address to be set to the required value directly as a two digit hex number. The switches may be operated by using a small screwdriver or a trimmer adjustment tool. SW1 is the high nibble and SW2 the low. The hex address on the switches is set as the switches are read.

If the BMSG232C module is being used as a master to drive a Bitbus network then the node address of the module is unimportant, however it is recommended that it is set to node FF to prevent clashes with other possible slaves on the network.

4.7 Module Address Configuration SW1, SW2

The base address of the BMS232C is selectable within the allowable Bitbus address range of 1...250. The node address is set in hexadecimal by rotary switches SW1 (MSNibble) and SW2 (LSNibble). The permitted range of switch settings is therefore 01...FA. The following diagram shows SW1 and SW2 configured for node address 3F hex (63 decimal).

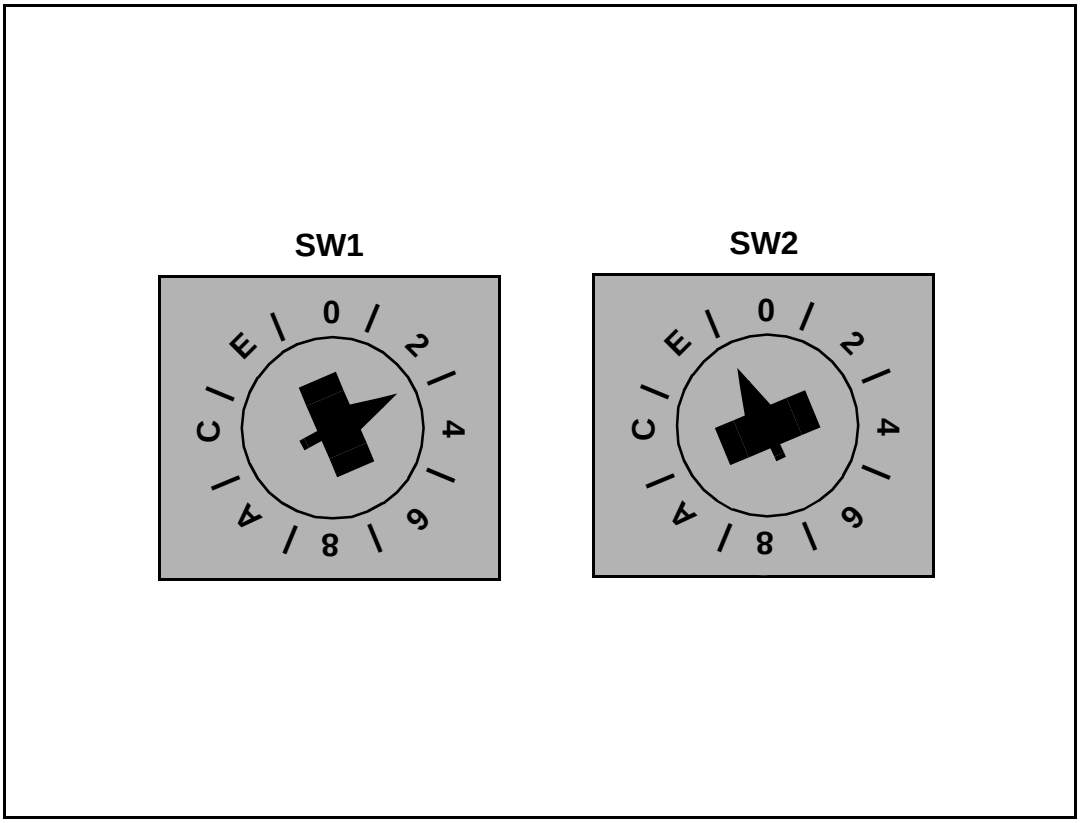
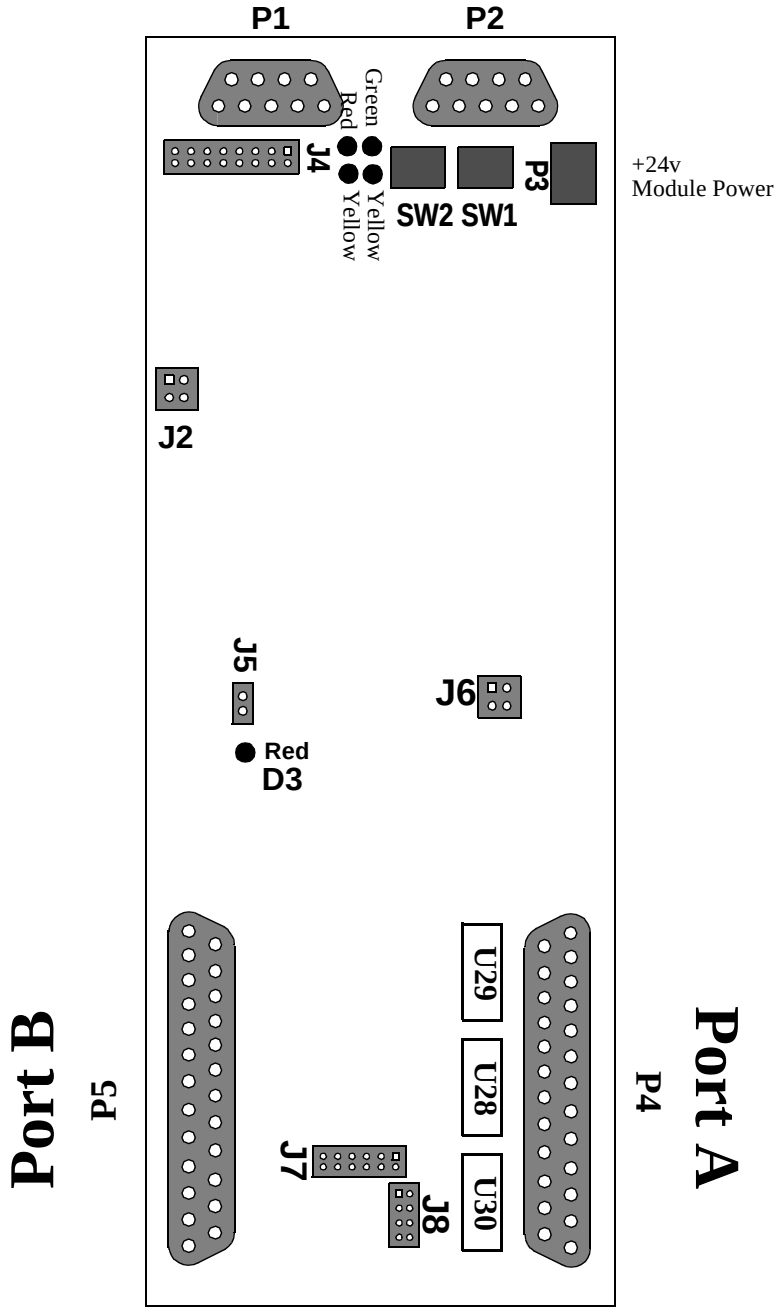


Figure 11. Node Address Selector Switches SW1 & SW2

4.8 Switch, Power and Jumper Locations



BMS232C module - Top View

Figure 12. Switch and jumper locations

5. Connections

5.1 General

This section gives the connections for the BMS232C module. The connector used at the front of the card for Bitbus (S1) is a 9 way D type socket. The two large connectors at the opposite end of the module are 25way D sub female sockets, compiling to the standard RS232C interface wiring as a DCE. That is to connect the Module to a terminal requires a straight through 5 pin connection.

5.2 Signal Names

On all signals, a '/' prefix is used to denote an inverted or active low signal. Thus the /DATA signal is the active low side of Bitbus Data signal from the 8044.

5.3 Power Supplies

The power supplies to the BMS232C module are connected via one screw terminal connector P1. The relevant pins are as follows:

With the Bitbus connectors on the left, board facing, the +24v goes to the left. The ground to the right.

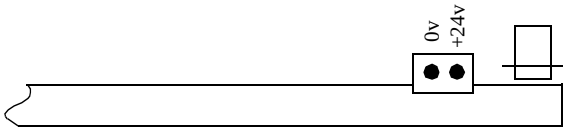


Figure 13. Power Connector Placement

5.4 Bitbus Connections

The pin functions for the Bitbus 9 way D sockets are shown below.

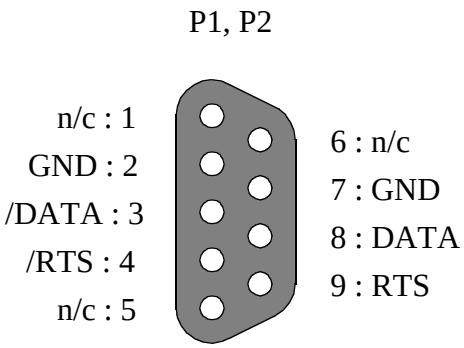


Figure 14. Bitbus Connections

5.5 Serial Connections.

The pin functions for the 25 way D connector are shown below

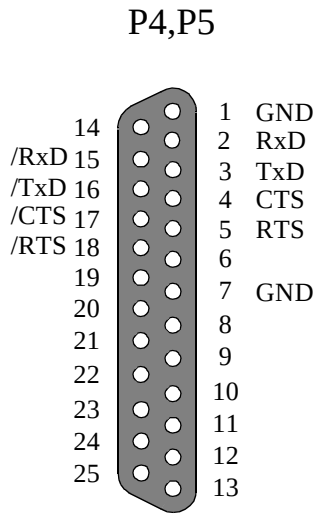


Figure 15. RS232C Connections

5.6 Expansion connector pinout.

In addition to the I/O connections described above, there is also the expansion bus connector which appears on all of the Quin Systems range of Bitbus Modules. This connector basically provides a copy of the system address/date bus and a full set of control and enable lines.

The enable lines are defined by PAL121 and PAL122. As the default settings, these define the /TOP_PAGE signal on the expansion bus as being active when the I/O page is being addressed (location $FF00 and above). There is also an optional additional line, /SEL which by default is not used, but by a change of PAL122, this can be programmed to address any page in memory.

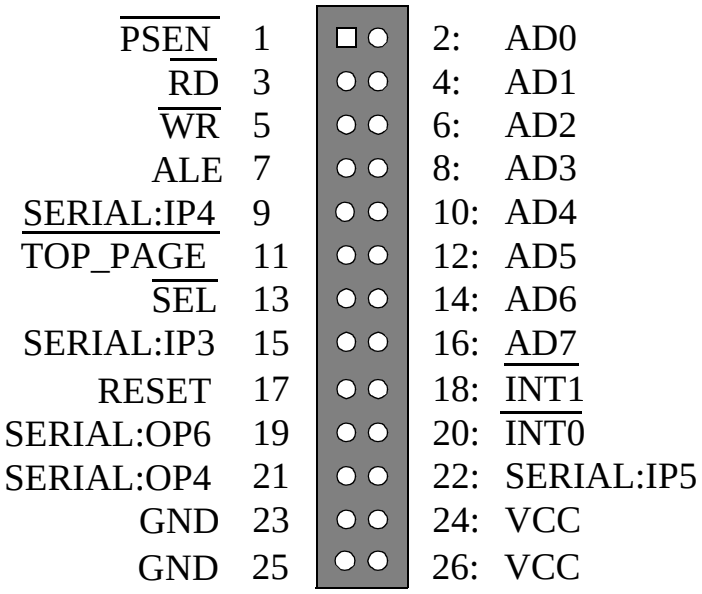


Figure 16. Expansion Bus connections

With the whole internal bus system of the microcontroller being brought out to a single connection, this gives the maximum flexibility in designing add on boards to the Bitbus range of modules. Providing any devices which required the expansion bus connector can exist within one page of memory, or preferably within the I/O address space of the processor, then any device could be attached.

In addition to the standard processor lines, there are above several digital I/O lines from the serial DUART device. These lines can be used as general digital I/O lines. However this expansion bus is **not isolated** and hence care must be taken to ensure that module isolation is not compromised. The digital I/O lines provided are such that further handshaking of the serial communications may be achieved, for example, generating DCE/DTR handshake lines, in addition to user digital I/O.

6. Serial Hardware description

6.1 Introduction

The serial hardware comprises the Exar XR88C681, a CMOS dual universal asynchronous receiver and transmitter(DUART) mapped into the 8044's I/O address space. The DUART device allows fully independent interrupt control of the serial communications. The device has the following features. For more details please refer to the Exar XR88C681 data sheet. This device is pin and function compatible with the industry standard 2681 DUART device.

6.2 Features

•Full Duplex, Dual Channel, Asynchronous Receiver and transmitter

•Quad-buffered receiver, Dual-buffered Transmitter

•Stop bits programmable in 1\over16-bit increments

•Internal Bit rate generator with 23 Bit rates.

•Independent Bit rate selection for each receiver and transmitter.

•Multifunction 16bit counter/timer

•Two digital inputs and two digital outputs in addition to handshake.

•Change of state detectors on inputs.

6.3 Serial Interface

All outgoing serial lines comply to the RS232C electrical standard and are fully isolated to 500VDC. The opto-couplers used ensure that the device is capable of transmitting and receiving up to the maximum data rate of 115.2Kbaud.

The DUART device may be used in a polling or interrupt driven environment. The hardware supports both software (Xon/Xoff) and hardware (RTS/CTS) handshaking. This is selectable either directly be accessing the memory mapped registers, or through the serial driver task.

When in the interrupt mode, the serial DUART can be made to generate interrupts to the host processor (the 8044) on various events happening within the serial devices' buffers. The DUART can hold up to four characters internally in the receiver and two in the transmitter and can generate interrupts on any combination of these buffers being empty or full. This is useful to increase the device's throughput.

6.4 Serial Operation

The serial device can be used in two ways. The standard way with the BMS232C Bitbus module is to use the BMS232C Serial Device driver. This provides a user-friendly front end running under the iDCX operating system. This task allows application programs to talk directly to a task via the message passing primitives, either locally or remotely over the Bitbus network. For further information please refer to the BMS232C Programmers Reference manual.[004]

In order to use the full features of the serial chip, including the timer/counters and spare digital I/O, it is necessary to read and write the hardware registers directly. The addresses of these registers are detailed below:

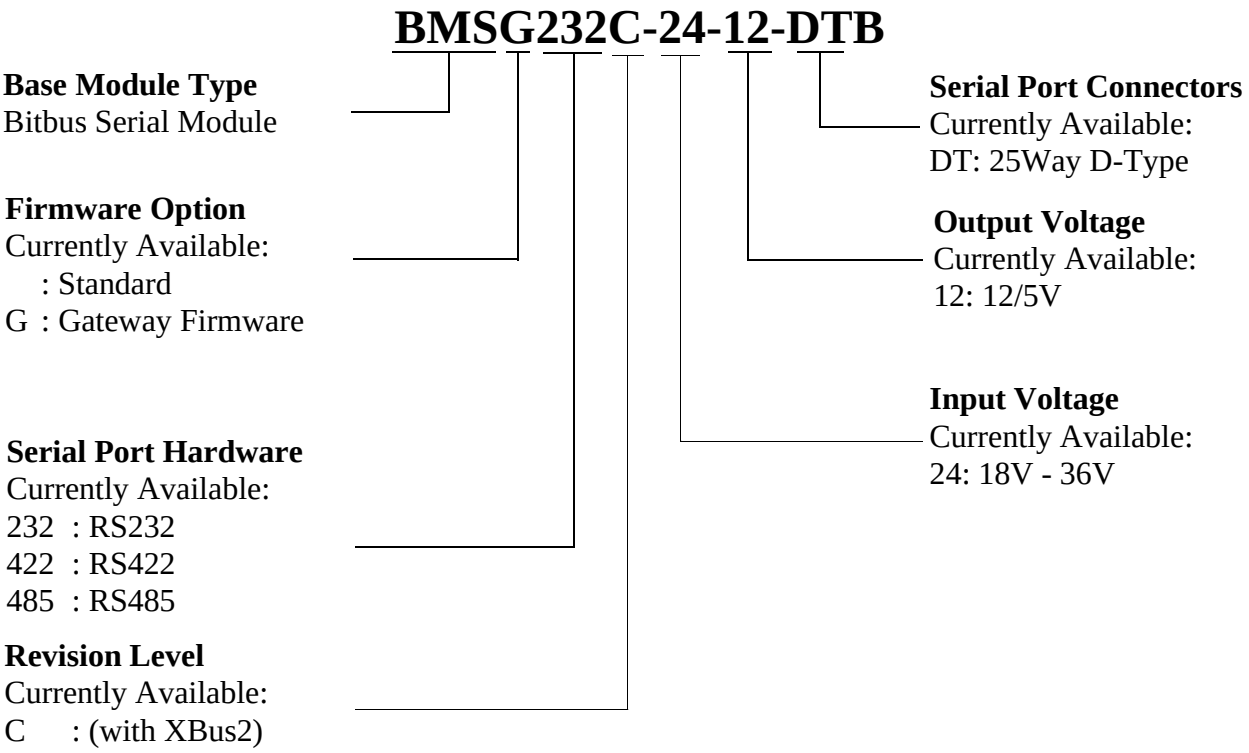
Table 8: 88C681 DUART Registers

Address	Read	Write
FFD0	Mode A	Mode A
FFD1	Status A	Clock Select A
FFD2	Interrupt Status Masked	Command A
FFD3	Rx Holding A	Tx Holding A
FFD4	Input Port Change	Auxiliary Control
FFD5	Interrupt Status Unmasked	Interrupt Mask
FFD6	Counter/timer Upper Byte	Counter/timer Upper Byte
FFD7	Counter/timer Lower Byte	Counter/timer Lower Byte
FFD8	Mode B	Mode B
FFD9	Status B	Clock Select B
FFDA	RESERVED	Command B
FFDB	Rx Holding B	Tx Holding B
FFDC	Interrupt Vector	Interrupt Vector
FFDD	Input Port	Output Port Configuration
FFDE	Start Counter	Set Output Port Bits Command
FFDF	Stop Counter	Reset Output Port Command

Figure 17. DUART Port and register addressing

7. Ordering Information

This module is designed to interface to the IEEE1118 Bitbus network and provides an XBus2 multiplexed 8 bit address/ 8 bit data bus for addition of XBus2 compatible expansion modules. Only one variant of the module is currently available, the following diagram shows how to specify the correct part number when ordering.



The above example shows a gateway configuration of the module with two high-speed RS232 serial ports fitted with standard 25 way D-Type connectors.

Numerics

A

B

C

D

E

H

I

J

L

M

N

O

P

R

S

T

U

W

X

