

CAN Peripherals

Hardware Guide

MN1255

Issue: 3.2

MN1255 1/2000





Copyright Baldor Optimised Control Ltd © 2000. All rights reserved.

This manual is copyrighted and all rights are reserved. This document may not, in whole or in part, be copied or reproduced in any form without the prior written consent of Baldor Optimised Control.

Baldor Optimised Control makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of fitness for any particular purpose. The information in this document is subject to change without notice. Baldor Optimised Control assumes no responsibility for any errors that may appear in this document.

Mint™ is a registered trademark of Baldor Optimised Control Ltd.

Limited Warranty

For a period of one (2) year from the date of original purchase, BALDOR will repair or replace without charge controls which our examination proves to be defective in material or workmanship. This warranty is valid if the unit has not been tampered with by unauthorized persons, misused, abused, or improperly installed and has been used in accordance with the instructions and/or ratings supplied. This warranty is in lieu of any other warranty or guarantee expressed or implied. BALDOR shall not be held responsible for any expense (including installation and removal), inconvenience, or consequential damage, including injury to any person or property caused by items of our manufacture or sale. (Some states do not allow exclusion or limitation of incidental or consequential damages, so the above exclusion may not apply.) In any event, BALDOR's total liability, under all circumstances, shall not exceed the full purchase price of the control. Claims for purchase price refunds, repairs, or replacements must be referred to BALDOR with all pertinent data as to the defect, the date purchased, the task performed by the control, and the problem encountered. No liability is assumed for expendable items such as fuses.

Goods may be returned only with written notification including a BALDOR Return Authorization Number and any return shipments must be prepaid.

Baldor Optimised Control Ltd
178-180 Hotwell Road
Bristol
BS8 4RP
U.K.
Telephone: +44 (0) 117 987 3100
Fax: +44 (0) 117 987 3101
email: sales@baldor.co.uk
Web site: www.baldor.co.uk

Baldor Electric Company
Telephone: +1 501 646 4711
Fax: +1 501 648 5792
email: sales@baldor.com
web site: www.baldor.com

Baldor ASR GmbH
Telephone: +49 (0) 89 90508-0
Fax: +49 (0) 89 90508-492

Baldor ASR AG
Telephone: +41 (0) 52 647 4700
Fax: +41 (0) 52 659 2394

Australian Baldor Pty Ltd
Telephone: +61 2 9674 5455
Fax: +61 2 9674 2495

Baldor Electric (F.E.) Pte Ltd
Telephone: +65 744 2572
Fax: +65 747 1708

Baldor Italia S.R.L.
Telephone: +39 (0) 11 56 24 440
Fax: +39 (0) 11 56 25 660





Safety Notice:


Only qualified personnel should attempt the start-up procedure or troubleshoot this equipment.


This equipment may be connected to other machines that have rotating parts or parts that are controlled by this equipment. Improper use can cause serious or fatal injury. Only qualified personnel should attempt to start-up, program or troubleshoot this equipment.


PRECAUTIONS:


 **Warning:** Do not touch any circuit board, power device or electrical connection before you first make sure that no high voltage present at this equipment or other equipment to which it is connected. Electrical shock can cause serious or fatal injury. Only qualified personnel should attempt to start-up, program or troubleshoot this equipment.


 **Warning:** Be sure that you are completely familiar with the safe operation of this equipment. This equipment may be connected to other machines that have rotating parts or parts that are controlled by this equipment. Improper use can cause serious or fatal injury. Only qualified personnel should attempt to program, start-up or troubleshoot this equipment.

-  **Warning:** Be sure that you are completely familiar with the safe programming of this equipment. This equipment may be connected to other machines that have rotating parts or parts that are controlled by this equipment. Improper programming of this equipment can cause serious or fatal injury. Only qualified personnel should attempt to program, start-up or troubleshoot this equipment.

-  **Warning:** Be sure all wiring complies with the National Electrical Code and all regional and local codes. Improper wiring may result in unsafe conditions.

-  **Caution:** To prevent equipment damage, be certain that the input power has correctly sized protective devices installed.

-  **Caution:** To prevent equipment damage, be certain that input and output signals are powered and referenced correctly.

-  **Caution:** To make sure reliable performance of this equipment be certain that all signals to/from the controller are shielded correctly.

Manual Revision History

Issue	Date	BOCL Reference	Comments
1.0	18/11/95	MN00200-000	1st draft
1.1	18/10/96	MN00200-001	Now includes 24I/O and keypad
2.0	13/03/98	MN00200-002	Re-formatted and updated
3.0	Sept 98	MN00200-003	<ul style="list-style-type: none">• Includes details for SmartMove• KeypadNode 4 included
3.1	Feb 99	UM00531-000	Draft changes to sections 9 and 10 with respect to the IoNode 24/24 CAN node.
3.2	Jan 2000	UM00531-001	Updated for NextMove PCI release.

Introduction	1
The CAN Peripherals	5
2.1 Common Features	6
2.1.1 RJ-45 CAN Connectors.....	6
2.1.2 CAN Jumpers.....	8
2.1.3 Network Termination.....	9
2.1.4 Power Supply	10
2.1.5 CAN Status LED.....	12
<i>InputNode 8: Eight Input Node</i>	13
3.1 J4: Input Connector	15
3.2 Input LEDs	16
<i>RelayNode 8: Eight Relay Output Module</i>	17
4.1 J4 and J5 : Output Connectors.....	19
4.2 Output LED Indication.....	19
<i>OutputNode 8: Eight PNP Output Module</i>	21
5.1 J4: Output Connector.....	24
5.2 Output LED Indication.....	25
<i>IoNode 24/24</i>	27
6.1 Differences from Other CAN Peripheral Products.....	28
6.2 Mechanical.....	28
6.3 Inputs	30
6.4 Outputs.....	31

KeypadNode	35
7.1 Mechanical.....	37
7.2 Electrical.....	39
7.2.1 Programming the Keypad and Display	39
Product History	45
8.1 <i>InputNode 8, OutputNode 8, RelayNode 8</i>	46
8.2 <i>IoNode 24/24</i>	46
8.3 <i>KeypadNode</i>	46
8.4 <i>KeypadNode 4</i>	46
Getting Started with SmartMove	47
9.1 Network Possibilities	48
9.2 Support for <i>IoNode 24/24</i>	49
9.3 Quick Start.....	50
9.3.1 Jumper settings	51
9.3.2 Connections and Configuration	52
9.4 SmartMove and CAN Peripherals.....	53
9.4.1 Selection of CAN Channel.....	53
9.4.2 Selection of CAN Baud Rate	53
9.4.3 Selection of Node ID.....	54
9.4.4 Network Termination.....	55
9.4.5 Static Configuration.....	56
9.4.6 Normal Operation.....	58
9.5 An Example Network.....	58
9.6 Using a <i>KeypadNode</i>	60

Mint Support for *SmartMove* 61

- 10.1 Errors And Error Handling 62
 - 10.1.1 Error Messages..... 62
- 10.2 Mint Keyword Summary..... 67
- 10.3 Mint Keyword Reference..... 68

Bibliography 81

Introduction

1

This chapter provides an introduction to the Baldor CAN peripherals and the Mint controllers, which support the devices.

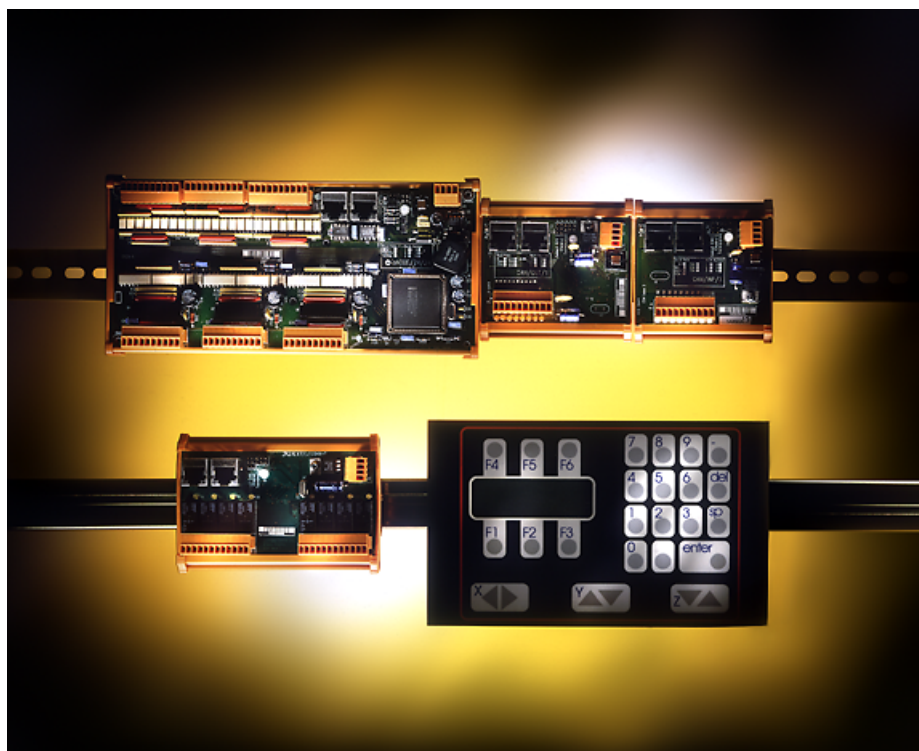


Figure 1: CAN Peripheral Product Family

Baldor CAN Peripherals are input and output expansion modules which communicate with the system controller via a CAN bus – a 1Mbit/s serial link. There are five devices in the family, each of which are discussed in this manual:

Order Code	Name	Description
ION001-503	<i>InputNode 8</i>	8 input node.
ION002-503	<i>RelayNode 8</i>	8 relay output node.
ION003-503	<i>OutputNode 8</i>	8 PNP Darlington driver output node.
ION004-503	<i>IoNode 24/24</i>	24 inputs and 24 outputs.
KPD002-502	<i>KeypadNode</i>	Operator keypad and display (3 axes)
KPD002-505	<i>KeypadNode 4</i>	Operator keypad and display (4 axes)

CAN is a high speed, noise-immune serial bus which allows multiple devices to communicate over a single twisted pair line (the bus or network). Several Baldor CAN Peripheral devices can be attached to the CAN network. A host device, such as a *MintDrive* or *NextMove* can communicate with the node by sending messages over the bus. The node will take this message and interpret its contents. For example, an *OutputNode 8* may receive a message to turn one of its outputs on. In order that a node can accept the correct message, it is identified by a unique address. In theory CAN supports up to a maximum of 63 devices on the network. *SmartMove* though, limits the maximum number of nodes in software. It only provides support for 7 devices in total on the network (see Section 9).

For more details on CAN configurations on v3 Mint controllers other than *SmartMove*, please refer to the Mint Programming Manual^[5].

For more details on CAN configurations on v4 Mint controllers, please refer to the Mint v4 CAN Programming Guide^[8].

The following table shows the controllers currently able to support Baldor CAN Peripherals:

Controller Product	Number of CAN Channels supported	v3 Mint	v4 Mint
		Supported on CAN channel	
<i>SmartMove 1, 2 and 3</i>	1	CAN 1	N/a
<i>ServoNode 50</i>	1	CAN 1	N/a
<i>ServoNode 51</i>	1	N/a	CAN 2
<i>MintDrive</i>	2	N/a	CAN 2
<i>NextMove BX</i>	2	CAN 2	CAN 2
<i>NextMove RK</i>	2	CAN 2	CAN 2
<i>NextMove PC</i>	1	CAN 1	CAN 2
<i>NextMove PCI</i>	2	N/a	CAN 2

For controllers running v4 Mint, the Baldor CAN Peripherals are always supported on CAN2 regardless of controller type.

The CAN Peripherals

2

This chapter provides basic physical details about CAN and the common features shared by the Baldor CAN peripherals.

- ◇ Common features.
- ◇ RJ45 CAN connector.
- ◇ CAN jumpers
- ◇ CAN network termination.

2.1 Common Features

The following features are common to the six Baldor CAN Peripherals, *InputNode 8*, *RelayNode 8*, *OutputNode 8*, *IoNode 24/24*, *KeypadNode* and *KeypadNode 4*:

- Remote operation over CAN.
- Operation over an ambient temperature range of 0°C – 40°C (32°F – 104°F).
- CAN status indication via a red-green LED.
- Power supplied, either via a four pin two part 3.5mm connector, or through the RJ-45 CAN connector. All nodes operate from 24V and some from as low as 12V.

All of the nodes except the *KeypadNodes* also feature:

- DIN rail mounting. The Modules are designed for mounting on either 35mm symmetric DIN rail (EN50 022, DIN 46277-3) or G-profile DIN rail (EN50 035, DIN 46277-1).

2.1.1 RJ-45 CAN Connectors

A pair of vertical, shielded RJ-45 connectors J1 and J2 are situated at the top left of each module. They provide connection to the CAN and a possible means of powering the CAN Peripheral.

The connections to both J1 and J2 are as follows:

Pin Number	Signal	Description
1	can1+	CAN channel 1 +
2	can1-	CAN channel 1 -
3	n/c	Not connected
4	0V	Signal Common
5	can-pwr+	Supply Voltage
6	n/c	Not connected
7	can2+	CAN channel 2 +
8	can2-	CAN channel 2 -

A very low error rate of CAN communication can only be achieved with a suitable wiring scheme.

Close attention should be paid to the following points:

1. CAN must be connected via twisted pair cabling e.g. CAT5 (or cable with equivalent specification). The connection arrangement is normally a simple multi-point drop. The CAN cables should have a characteristic impedance of 120Ω ; and a delay of 5ns/m (1.72ns/ft). Other characteristics depend upon the length of the cabling:

Cable length	Maximum bit rate	Specific resistance	Conductor area
0-40m (0-131ft)	1 Mbit/s	$70\text{m}\Omega$	$0.25\text{-}0.34\text{mm}^2$
40m-300m (131-984 ft)	200 kbit/s	$< 60\text{m}\Omega$	$0.34\text{-}0.60\text{mm}^2$
300m-600m (984-1968 ft)	100 kbit/s	$< 40\text{m}\Omega$	$0.50\text{-}0.60\text{mm}^2$
600m-1000m (1968-3280 ft)	50 kbit/s	$< 26\text{m}\Omega$	$0.75\text{-}0.80\text{mm}^2$

2. Terminators should be fitted at both ends of the network and nowhere else.
3. To reduce RF emissions and, more importantly, to provide immunity to conducted interference, shielded twisted pair cabling should be used. If two CAN channels are bundled in a cable then each needs a twisted pair.

NOTE: Cable screens/shields should not be connected to 0V on connectors J1 and J2 as this will inject conducted interference into the 0V plane on the processor board.

4. The 0V rails of all of the nodes on the network must be tied together. One way of achieving this is by arranging for the CAN cabling to provide continuity of the 0V on connector J1 or J2 to the 0V on connector J1 or J2 of the adjacent CAN Peripherals. This makes sure that the CAN signal levels transmitted by a CAN Peripheral, are within the common mode range of the receiver circuitry of other nodes on the network.

The following are examples of suitable connectors and cabling:

Connector	Description	Part Number
RJ-45 connector	Molex 95043.2891	(Farnell part No. 497-861)
Cable	Alcatel Cablenet 4 pair, foil screened/shielded, twisted pair	(Farnell part No. 296-788)
Crimp Tool	Molex 69008.1100	(Farnell part No. 473-250)

Standard, good quality cables should be purchased from Baldor. The order codes are:

Order Code	Description
CBL007-501	0.25m cable (0.8ft)
CBL007-502	0.5m cable (1.6ft)
CBL007-503	1m cable (3.3ft)
CBL007-504	2m cable (6.6ft)
CBL007-505	3m cable (9.9ft)
CBL007-506	5m cable (16.5ft)
CBL007-507	10m cable (33ft)
CBL007-508	25m cable (82.5ft)

2.1.2 CAN Jumpers

Setting up CAN communications, for all six modules is covered by jumpers JP1, JP2 and JP3.

The CAN communications can be connected to CAN channel 1 or CAN channel 2. CAN channel 1 is chosen by fitting the jumpers JP1 and JP2 to the lower positions (Position 1). CAN channel 2 is chosen by fitting JP1 and JP2 to the upper positions (Position 2).

JP3, when fitted, connects a 120Ω terminating resistor across the CAN lines. It should be omitted unless the CAN Peripheral is at the end of the system.

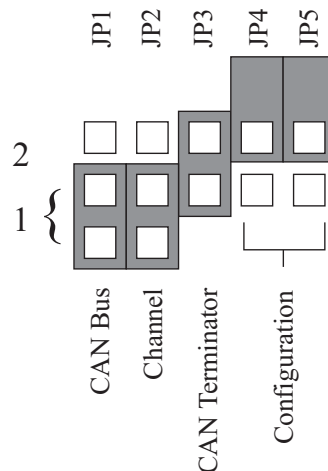


Figure 2: CAN Jumpers

NOTE : Jumpers JP4 and JP5 are used for configuration purposes.

2.1.3 Network Termination

Termination resistors must be fitted at the ends of the network to reduce signal reflection. The controllers and CAN Peripherals are fitted with termination resistors specifically for this purpose.

NOTE: If a node or a controller is at the end of the network, its terminator must be fitted.

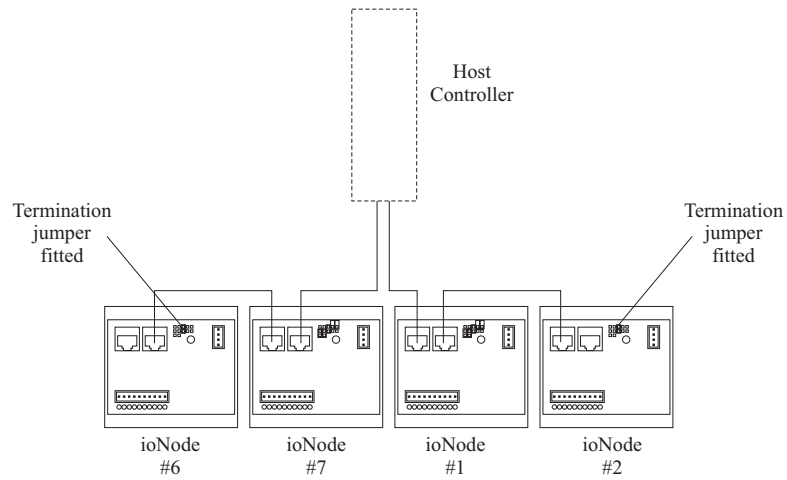


Figure 3: Multi-node network (a) showing termination

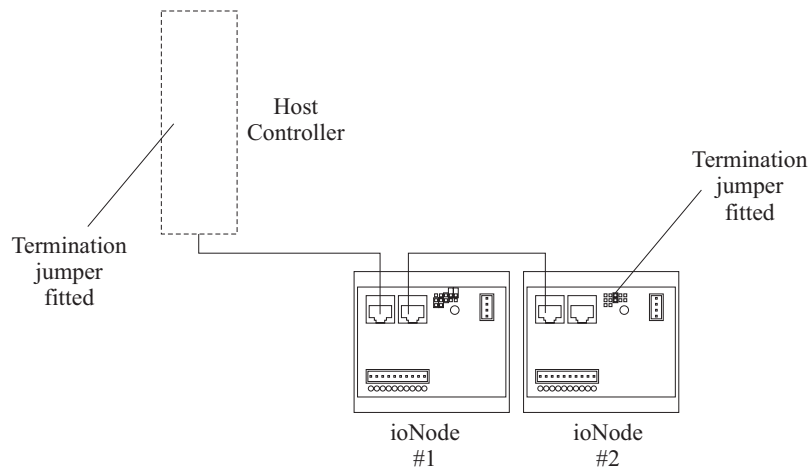


Figure 4: Multi-node network (b) showing termination

Node	Terminator selection
<i>SmartMove</i>	The terminator is selected by turning on switch 5 of the 5 pole DIP switch accessible at the front panel.
<i>ServoNode 50</i>	The terminator is selected by fitting a jumper to JP1 and JP2 headers.
<i>ServoNode 51</i>	The terminator is selected by turning on switch 4 (CAN Configuration - 120R) of the 4 pole DIP switch accessible at the top face of the product.
<i>MintDrive</i>	The terminator is selected by turning on switch 2 (CAN2) of the 3 pole DIP switch accessible at the front panel.
<i>NextMove RK</i>	The terminator is selected by fitting a link to CAN jumper 2, located next to the RJ-45 connectors.
<i>NextMove BX</i>	The terminator is selected by fitting a link to CAN jumper 2, located next to the RJ-45 connectors.
<i>NextMove PC</i>	The terminator is selected by fitting jumper link M on the PCB.
<i>NextMove PCI</i>	The terminator is selected by fitting a jumper to J17 (CO Term) on the Breakout board.

Please refer to the appropriate Installation manuals^{[1][2][3][4]} for more details on selecting the termination resistors.

2.1.4 Power Supply

The eight input module *InputNode 8* and the eight PNP output module *OutputNode 8* can be powered by +12V to +24V DC, whereas the *IoNode 24/24* operates with a supply voltage of +12 to +30V DC.

The *KeypadNode* supply voltage range is +15V to +24V DC; operation above +30V DC may cause damage and operation below 15V DC will cause the internal 12V rail to go out of tolerance, but the unit will still function only the buzzer will be quieter.

The relay output module *RelayNode 8* must be powered by +24V DC.

Relay Properties:

Load	Resistive load ($\cos\phi = 1$)
Rated Load	0.5 A at 125V AC; 2 A at 30 V DC
Max. switching voltage	125 V AC, 125 V DC
Max. switching current	2 A
Max. switching capacity	62.5 VA, 60 W
Min. permissible load	0.01mA at 10 mV DC

Contact resistance	50 mΩ max.
Operate time	7 ms max.
Release time	3 ms max.
Bounce time	Operate: approx. 0.3 ms Release: approx. 1.5 ms
Max. operating frequency	Mechanical: 36,000 operations/h Electrical : 1,800 operations/hr (under rated load)
Insulation resistance	1,000MΩ min. (at 500 V DC)
Life expectancy	Mechanical: 15,000,000 operations min. (at 36,000 ops/h) Electrical: 100,000 operations min. (at 1,800 ops/h)

The voltage supply can be distributed around a system of CAN nodes via pins 4 and 5 of the RJ-45 CAN connector. However, it must be connected to at least one node via the four position two-part 3.5mm connector J3.

NOTE: If there is one or more *RelayNode 8* module in the system the supply must be 24V dc.

The J3 power connectors are all reverse bias protected, so if different power supplies are connected to more than one CAN node, the system will be powered by the highest voltage with no damage to the other power supplies.

NOTE: The maximum current that can be passed through the RJ-45 connectors is 1A, if this is likely to be exceeded, power must be connected via the J3 connector of each CAN Peripheral .

The following table shows the power supply connections to the four pin connector J3 which is situated towards the top right of all the CAN nodes:

Terminal	Signal
1	can-pwr+
2	0V
3	can-pwr +
4	0V

Power does not have to be connected to both pairs of pins as the CAN nodes can be daisy-chained.

2.1.5 CAN Status LED

The red-green LED located near the top centre of each module is the CAN Status LED with the following operation:

Status	Description
Flashing Green	<ul style="list-style-type: none">• When involved in CAN communication The LEDs on the CAN Peripherals will flash green. The controllers operate a node guarding procedure in which all nodes are regularly sent a CAN message. This procedure is seen on the CAN Peripheral as a flash of the green LED, once every half-second.
Constant Green	<ul style="list-style-type: none">• Fitting JP4 forces the green LED on.• The LED shows constant green if there is a continuous stream of messages to the node.
Flashing Red	<ul style="list-style-type: none">• A fault condition is present, such as the node guarding message is lost.• The LED will flash red on initial power-up of the node.
Constant Red	<ul style="list-style-type: none">• Fitting JP5 forces the red LED on.• The LED will show constant red if the node experiences a problem in initializing its CAN controller or EPROM. In this case the node will not respond to any CAN messages.

InputNode 8: Eight Input Node

3

This chapter provides details on InputNode 8, the 8 digital input node.

- ◇ Technical Information.
- ◇ Connection Information.

InputNode 8 is an eight input expansion module.

Length	83mm (3.26")
Width	86mm (3.86")
Depth	60mm (2.36")
Weight	105g approx. (3 3/4oz)
Power consumption	45mA @ 12V 90mA @ 24V

InputNode 8 has eight inputs, which are optically isolated and can be positive or negative common (for use with NPN or PNP transistors). The inputs are guaranteed to be active in the range +/-12V to +/-24V and inactive between + and - 2V.

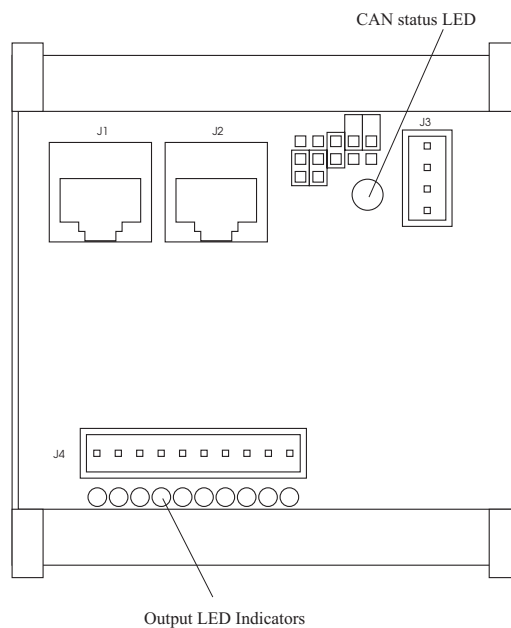


Figure 5: InputNode 8

3.1 J4: Input Connector

The input connector J4 is the 10 pin two part 3.5mm connector situated towards the bottom of the module. The connections are as follows:

Terminal	Signal
1	input 7
2	input 6
3	input 5
4	input 4
5	input 3
6	input 2
7	input 1
8	input 0
9	usr common
10	usr common

The input impedance is 2K2. Connections are made as follows:

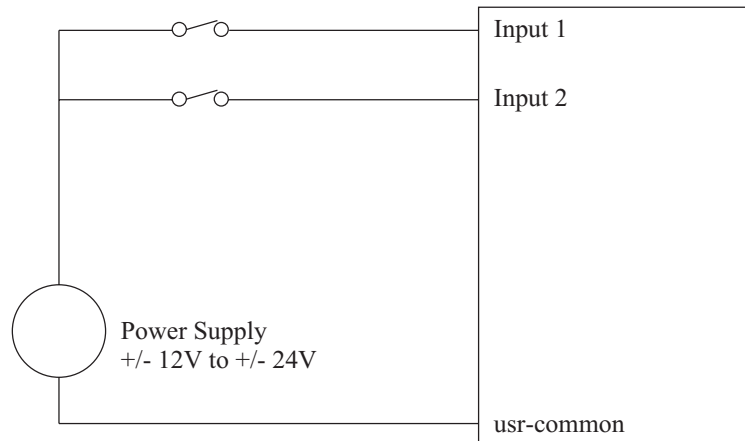


Figure 6: Input connection example

NOTE. Either PNP or NPN inputs may be used.

3.2 Input LEDs

The eight input LEDs found under the input connector J4, indicate when voltage is presented to an input. The LEDs will light green if a positive input (PNP) is applied and yellow if a negative input (NPN) is applied.

RelayNode 8 : Eight Relay Output Module

4

This chapter provides details on RelayNode 8, the 8 relay output node.

- ◇ Technical Information.
- ◇ Connection Information.

RelayNode 8 is an eight relay output expansion module.

Length	125mm (4.92")
Width	86mm (3.38")
Depth	60mm (2.36")
Weight	190g approx. (6 ³ / ₄ oz)
Power consumption	215 mA @ 24V approx.

RelayNode 8 requires a power supply of +24V dc to make sure of correct operation of the relay coils.

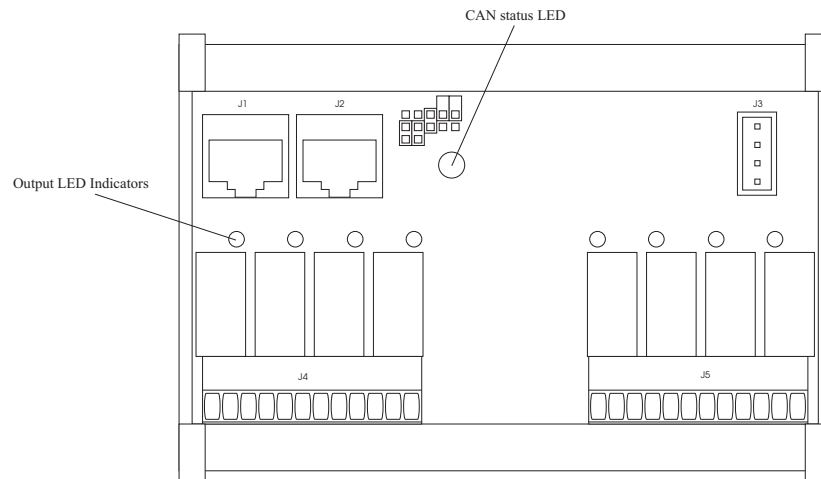


Figure 7: RelayNode 8

4.1 J4 and J5 : Output Connectors

The output relay connections are brought to the two, 12 pin, two part, 3.5mm connectors found at the bottom of the *RelayNode 8* module. They are arranged as follows:

Connector J4		Connector J5	
Terminal	Signal	Terminal	Signal
1	common 0	1	common 4
2	Normally Closed 0	2	Normally Closed 4
3	Normally Open 0	3	Normally Open 4
4	common 1	4	common 5
5	Normally Closed 1	5	Normally Closed 5
6	Normally Open 1	6	Normally Open 5
7	common 2	7	common 6
8	Normally Closed 2	8	Normally Closed 6
9	Normally Open 2	9	Normally Open 6
10	common 3	10	common 7
11	Normally Closed 3	11	Normally Closed 7
12	Normally Open 3	12	Normally Open 7

4.2 Output LED Indication

The *RelayNode 8* output module has a yellow LED for each output which is lit when the relay is energised.

OutputNode 8: Eight PNP Output Module

5

This chapter provides details on OutputNode 8, the 8 digital output node.

- ◇ Technical Information.
- ◇ Connection Information.

OutputNode 8 is an eight output expansion module. The outputs are optically isolated and are protected against *over current* and *over temperature*.

Length	90mm (3.54")
Width	86mm (3.38")
Depth	60mm (2.36")
Weight	110g approx. (3 3/4oz)
Power consumption	110mA @ 12V approx. 130mA @ 24V approx

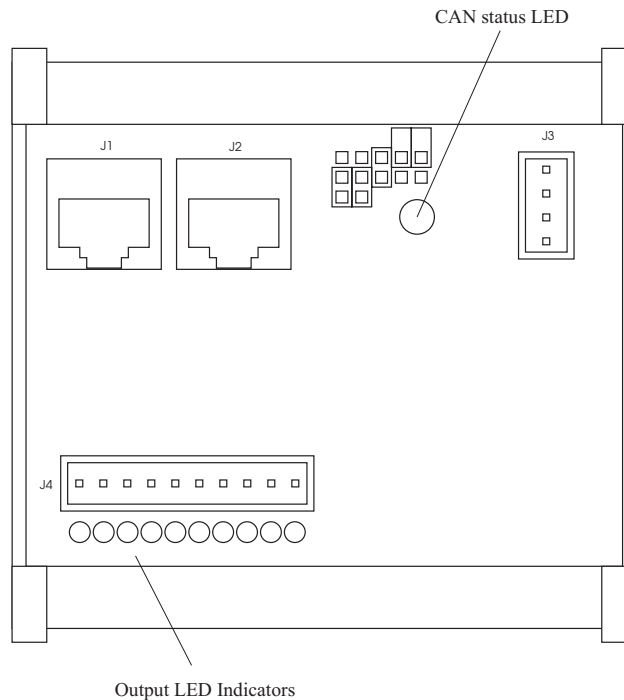


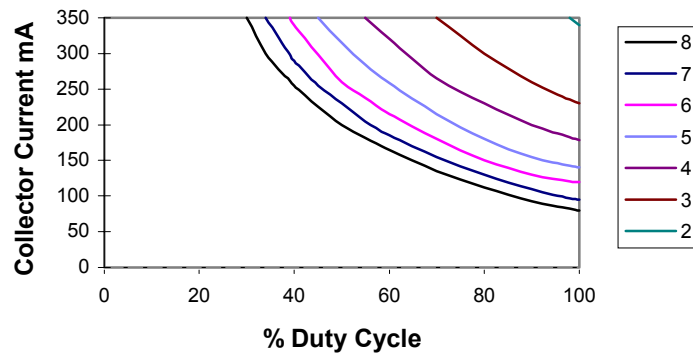
Figure 8: *OutputNode 8*

The outputs are driven by an octal PNP Darlington array (Allegro UDN2987A). Each output is capable of continually sourcing 50mA nominal, on all channels. A single channel can source up to 350mA. However, the total output for all channels in a bank of eight cannot exceed 500mA.

If the *over current* or *over temperature* protection is activated, one or more of the outputs will switch off. It will not be possible to switch that, or any other output, back on until the fault has been removed and the module reset.

Some loads, such as tungsten filament lamps, may draw only 50mA in the steady state but draw an inrush current at turn on which is large enough to overload the *over current* protection. In such cases the use of LED lamps should be considered. It is also possible to connect more than one output in parallel, although there is no guarantee of the outputs sharing the load equally.

Number Of Outputs On Simultaneously



The above diagram gives information about the current sourcing capabilities of the eight outputs.

5.1 J4: Output Connector

The output connections are brought out to the 10 pin, two part, 3.5mm connector situated at the bottom of the *OutputNode 8* module. The outputs are supplied via the **user V+** and **user ground** connections and should be in the range +12V to +24V. The output connections are arranged as follows:

Terminal	Signal
1	output 0
2	output 1
3	output 2
4	output 3
5	output 4
6	output 5
7	output 6
8	output 7
9	user V+
10	user ground

NOTE: There is a mis-print on the silkscreen of issue 2 boards. The silkscreen reads GND on pin 9 of J4, it should read V+.

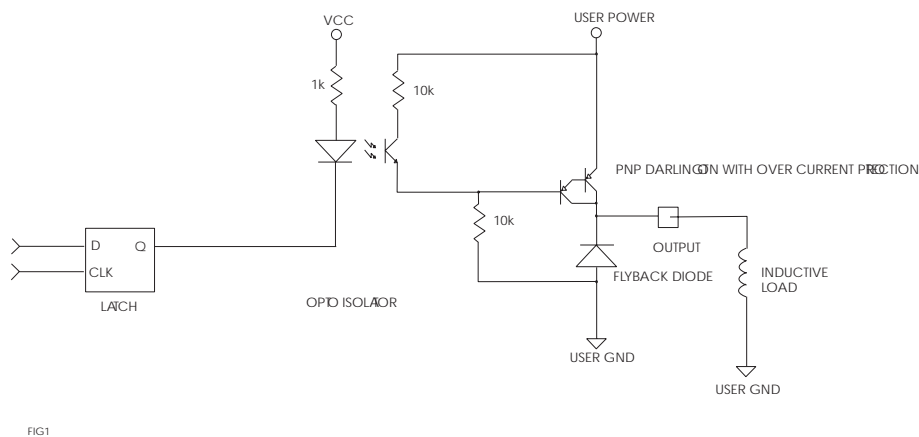


Figure 9: Output Circuit (simplified schematic).

5.2 Output LED Indication

Yellow LEDs located below the output connector J4 indicate the state of each output. They will be lit when an output is on.

IoNode 24/24

6

This chapter provides details on IoNode 24/24, the 24 digital input and output node.

- ◇ Technical Information.
- ◇ Connection Information.

IoNode 24/24 supports 24 input channels and 24 output channels. The inputs are optically-isolated and are organised into three banks of eight – each having its own **common**. The outputs are also opto-isolated and again are organised into three banks of eight. Thus *IoNode 24/24* is electrically equivalent to 3 x *InputNode 8* plus 3 x *OutputNode 8*. The pin numbering of connectors and the functions of the jumpers, the CAN LEDs and so on, have been made as similar as possible across these products.

6.1 Differences from Other CAN Peripheral Products

The differences can be summarised as follows:

- The PCB width is 100mm (3.9”) rather than 68mm (2.67”) for the other DIN-rail mounting units.
- Unlike the other nodes, *IoNode 24/24* uses a switching power supply. The operating range is 12 – 30V and the unit typically draws 79mA@12V and 59mA@24V with all inputs and outputs *off*. It draws 369mA@12V and 267mA@24V with all inputs and outputs *on*.
- For reasons of space, the inputs do not have individual LEDs indicating the presence and direction of input current. Instead there is one LED, D10, which flashes in response to a detected change of input.
- For reasons of space the outputs do not have individual LEDs. Instead there is one LED, D8, which flashes in response to a new output command.
- When JP7 is fitted (it is normally omitted) the 0V supply rail is tied to the potential of the chassis connection points and the screen on the RJ-45 connector. This offers the user some additional options when designing an earthing (grounding) scheme.
- *IoNode 24/24* operates differently on SmartMove. See section 9 for details.

6.2 Mechanical

The circuitry is housed on a 100mm x 220mm (3.9” x 8.66”) PCB. It weighs 265g (9.35oz) and, when housed on the DIN rail raft, weighs 425g (15oz) overall. The clearance above the PCB should be 50mm (2” approx.) – or 90mm (3 ½”) above the base of the DIN rail – to allow for adequate bending radius for the cabling. The operating temperature range is 0°C to 40°C (32°F to 104°F), operation at up to 50°C (122°F) is possible but subject to limitations on the current drawn from the output drivers. A view of the unit is shown in Figure 10.

The unit may be purchased as a stand-alone board, (not contained within a rail mountable raft/module). In this case, fixing holes for M3 diameter screws and an offset pillar to raise the board from a carrier PCB are provided. All four corner holes and at least one of the central holes must be used to make sure that the PCB is adequately supported. Any of the corner fixing holes, except that near the processor (U9), provides a means of making electrical contact to the chassis screen/shield on the PCB.

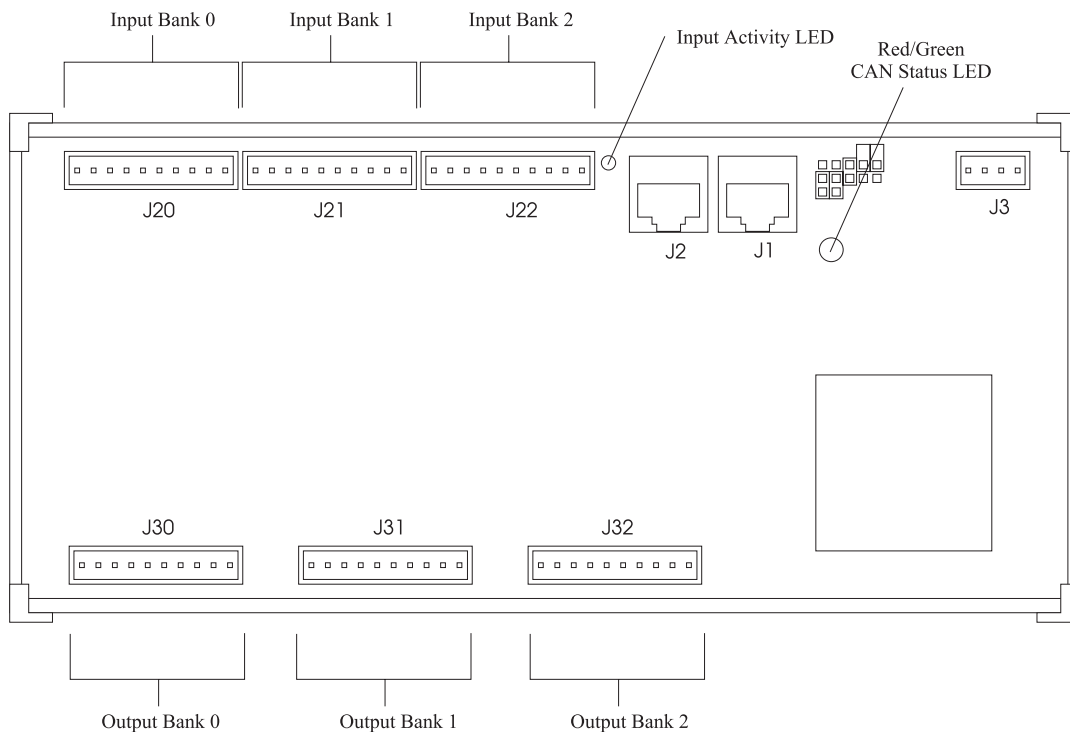


Figure 10: IoNode 24/24

For maximum noise immunity it is recommended to connect the chassis connection on the PCB to the system star point using a low impedance conductor. In many cases the most convenient way to achieve this is by direct connection to nearby metalwork. There are two methods:

- By fitting an un-insulated offset pillar in an appropriate corner fixing position and bolting the unit down.
- By fitting an earthing (ground) strap to the 1/8" spade terminal J6.

6.3 Inputs

The inputs are numbered **din0–din23**. They are grouped into three banks of eight, each has a distinct common; **com0–com2**. The use of AC opto-isolators allows a given bank to be connected for all PNP or all NPN use. It also provides reverse bias protection. To configure a bank’s inputs all for PNP use, tie **comx** to the more negative rail of the machine control supply. Conversely, to configure a bank’s inputs all for NPN use, tie **comx** to the more positive rail of the machine control supply. The circuit for a bank is:

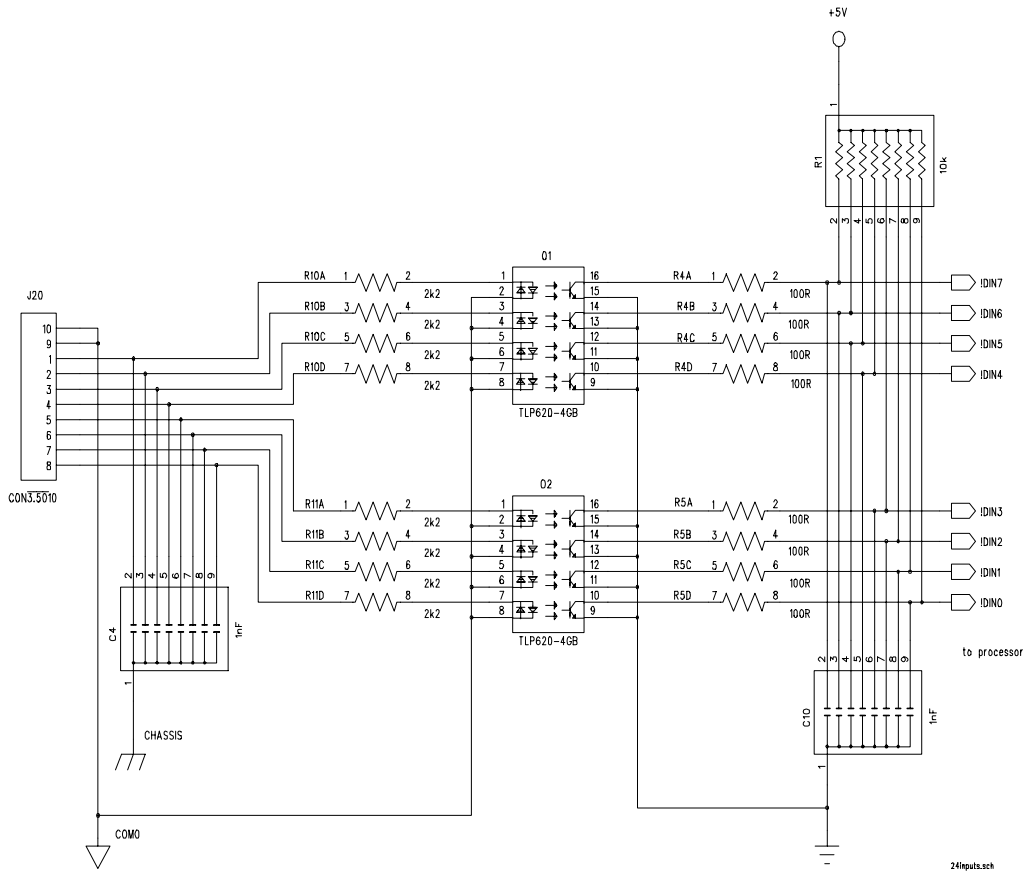


Figure 11: IoNode 24/24 input circuit

The input connections are summarised in the following table:

Pin	J20	J21	J22
1	din7	din15	din23
2	din6	din14	din22
3	din5	din13	din21
4	din4	din12	din20
5	din3	din11	din19
6	din2	din10	din18
7	din1	din9	din17
8	din0	din8	din16
9	com0	com1	com2
10	com0	com1	com2

A green LED, D10, flashes every time a change of input has been detected. The inputs may be operated from 12V to 24V. Operation above 30V may cause damage.

As with the other CAN Peripherals, the isolation provided is nominal, din0–23 and com0–2 must all be within 42.4V of machine’s safety earth (ground) potential.

6.4 Outputs

The outputs are numbered **Dout0** to **Dout23**. They are grouped into three banks of eight, each has a distinct pair of supply rails (**usrV+0, usrgnd0**), (**usrV+1, usrgnd1**) and (**usrV+2, usrgnd2**). A more detailed discussion of the outputs can be found in the section on *OutputNode 8*. The circuit for a bank is shown in Figure 12:

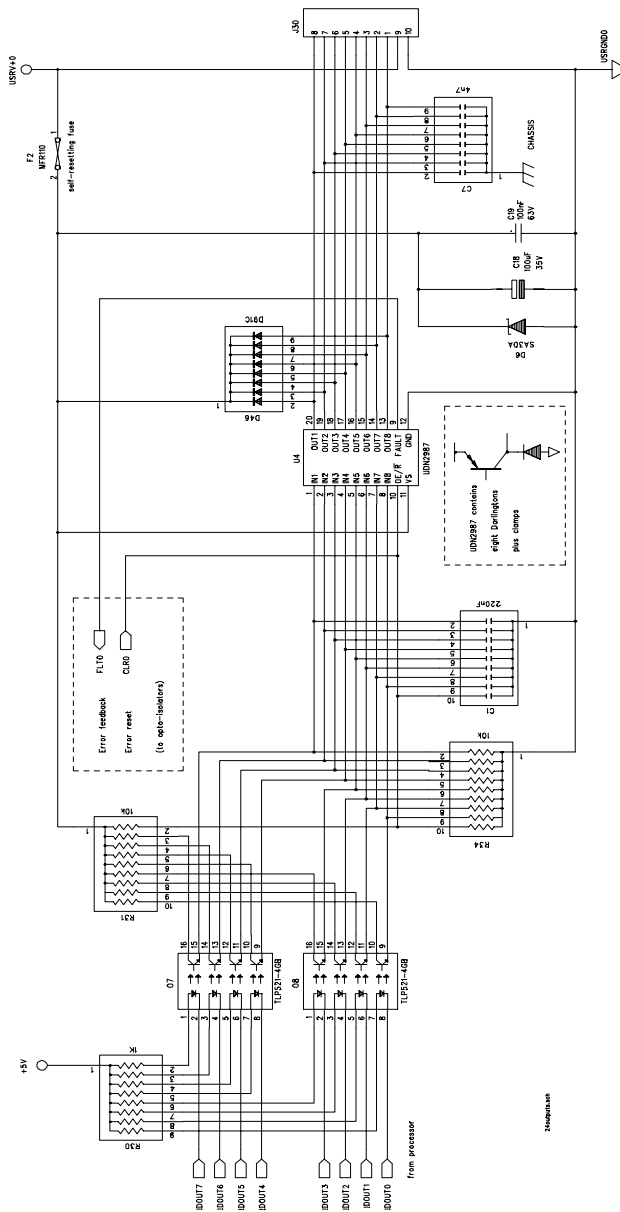


Figure 12: IoNode 24/24 output circuit

Note that each bank has its own feedback of a fault condition. Thus an overheated output will shut down all the outputs in the same bank but will not affect the other banks. An overloaded output can be traced to its bank of eight. Each bank has its own *error reset* line which may be used to clear the fault in any output on that bank, all of the outputs will be momentarily disabled while the fault is being cleared.

The output connections are summarised below:

pin	J30	J31	J33
1	Dout0	Dout8	Dout16
2	Dout1	Dout9	Dout17
3	Dout2	Dout10	Dout18
4	Dout3	Dout11	Dout19
5	Dout4	Dout12	Dout20
6	Dout5	Dout13	Dout21
7	Dout6	Dout14	Dout22
8	Dout7	Dout15	Dout23
9	usrV+0	usrV+1	usrV+2
10	usrgnd0	usrgnd1	usrgnd2

A green LED, D8, flashes every time a change of output command has been received. The outputs may be operated from 12V to 24V. Operation above 30V may cause damage.

As with the other CAN Peripherals, the isolation provided is nominal, Dout0–23, usrV+0–2 and usrgnd0–2 must all be within 42.4V of the machine’s safety earth (ground) potential.

KeypadNode

7

This chapter provides details on KeypadNode, the family of CAN based operator panels.

- ◇ Technical Information.
- ◇ Keyboard mapping for Mint.
- ◇ Connection Information.

KeypadNode provides a CAN based operator panel with the following features:

- 4 line by 20 character back-lit LCD display controlled by software.
- Audible buzzer which is controlled by software.
- Choice of 27-key or 41-key membrane, numerical keypad, function keys and jog keys. Tactile feedback is provided on all keys.
- It can be fixed to a plain metal surface such as a control cabinet, or alternatively into a 3U rack. Refer to Section 7.1.

In addition to these features, CAN has inherent EMC benefits and the option of placing the keypad a considerable distance away from the host controller.

The pin numbering of connectors and the functions of the jumpers and for example, the CAN LED have been made similar to other CAN Peripheral family products.

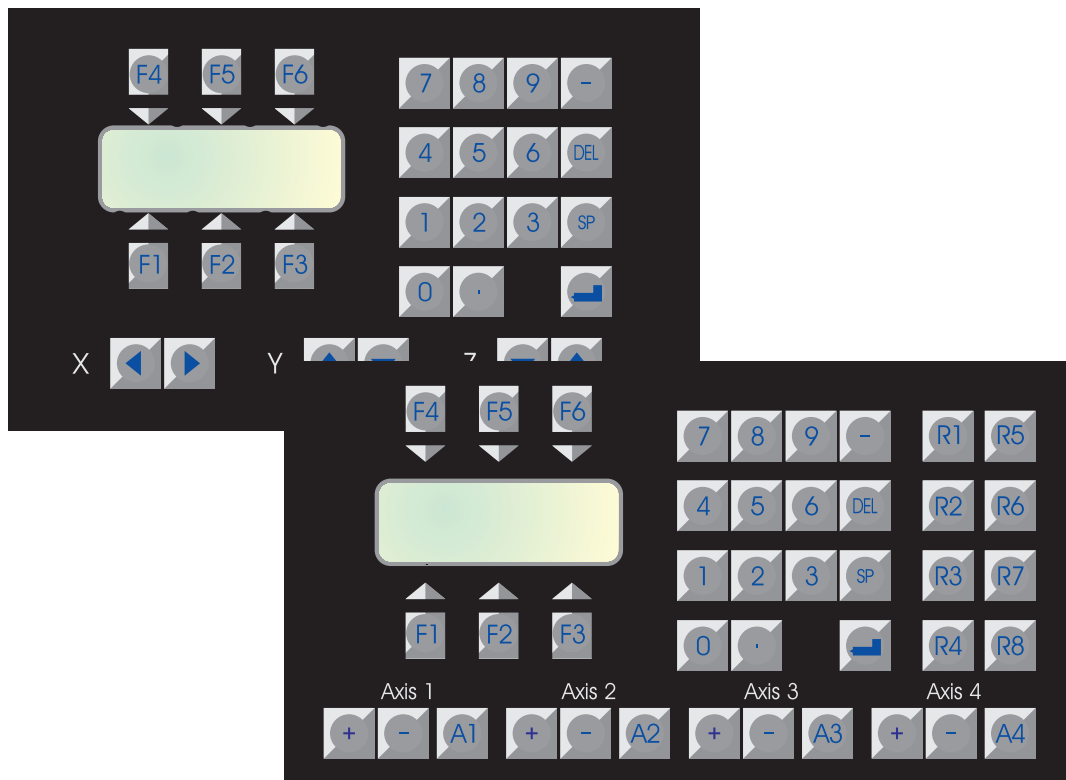


Figure 13: Keypad layout

7.1 Mechanical

The circuitry is housed on a 100mm x 93mm (3.9" x 3.7") PCB. It weighs 75g (2.65oz) and when fitted to the keypad unit itself weighs 405g (0.89lb) overall. The clearance behind the keypad front plate should be 50mm (1.95") to allow an adequate cable-bend radius. The operating temperature range is 0° – 40°C (32° – 104°F). A back view of the unit is shown in Figure 15.

The studs used for panel mounting are M3 and the studs, which hold the PCB are M2.5. When fitting to a 3U rack the oval holes in the membrane must be cut out to admit the fixing screws provided.

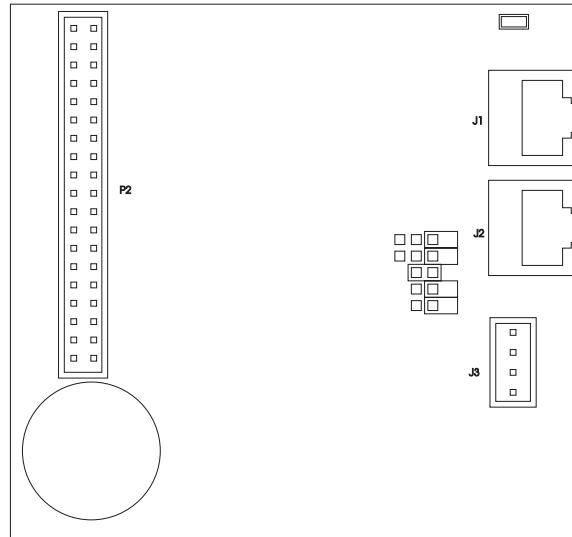


Figure 14: KeypadNode PCB

There is a rear cover plate, which is intended to provide a measure of protection to the PCB.

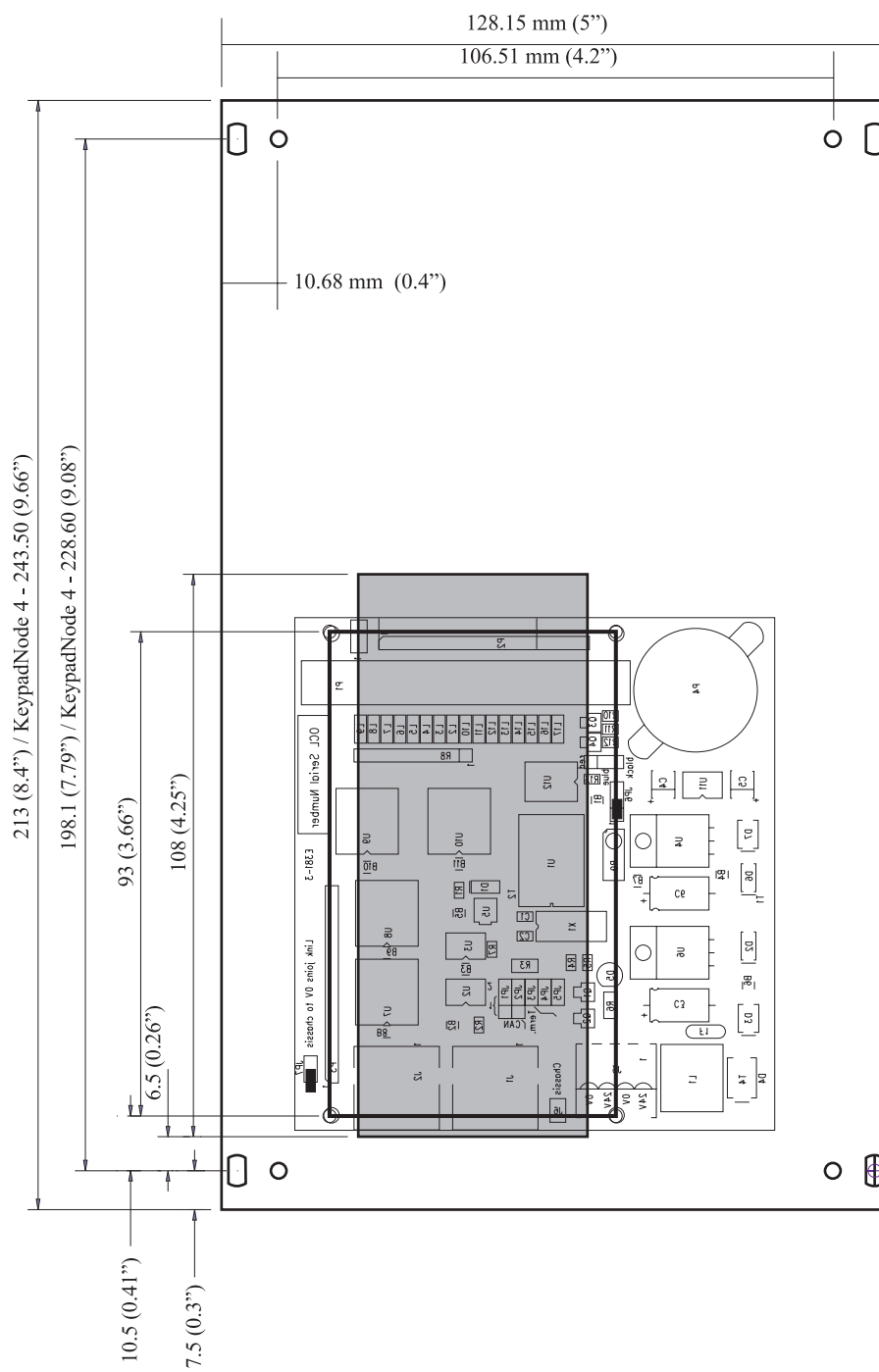


Figure 15: KeypadNode Mechanical Layout

7.2 Electrical

The unit draws typically 135mA, the supply voltage range is 15V–24Vdc. Operation above 30Vdc may cause damage. Operation below 15V will cause the internal 12V rail to drift out of tolerance. However, the unit will still function although the buzzer will be quieter. The PCB may be purchased without the keypad in which case fixing holes for M2.5 screws and pillar are provided. All four corner holes should be used to make sure that the PCB is adequately supported. Connection to the keypad is via the 34-pin header P1, which has an identical pin-out to *EuroSystem* products. The buzzer on the target keypad should be disconnected.

The header JP6 selects +5V to -5V or 0V to -5V bias voltage range for the display. Fitting a jumper to the right selects +5V to -5V and this is standard.

For maximum noise immunity it is recommended to connect the chassis connection on the PCB to the system star point using a low impedance conductor. In many cases the most convenient way to achieve this is by direct connection to nearby metalwork. There are two methods:

- By fitting an un-insulated offset pillar in an appropriate corner fixing position and bolting the unit down.
- By fitting an earthing (ground) strap to the 1/8" spade terminal J6.

When JP7 is fitted (it is normally omitted) the 0V supply rail is tied to the potential of the chassis connection points and the screen/shield on the RJ-45 connector. This offers the user some additional options when devising an earthing/grounding scheme.

7.2.1 Programming the Keypad and Display

Mint software supports the keypad and display as if it were a standard serial terminal. Mint statements **PRINT**, **INPUT**, **CLS**, **LOCATE** and so on can be used with the display. **BEEP** activates the buzzer. Key presses cause characters to be placed in the serial port buffer so that they can be read as normal by **INKEY** and **INPUT**. The **READKEY** function returns the value of the key that is currently pressed. This is an enhancement not normally available on serial terminals and can be used for jogging motors when the operator holds a finger on a key.

The keys on the keypads are mapped as shown:

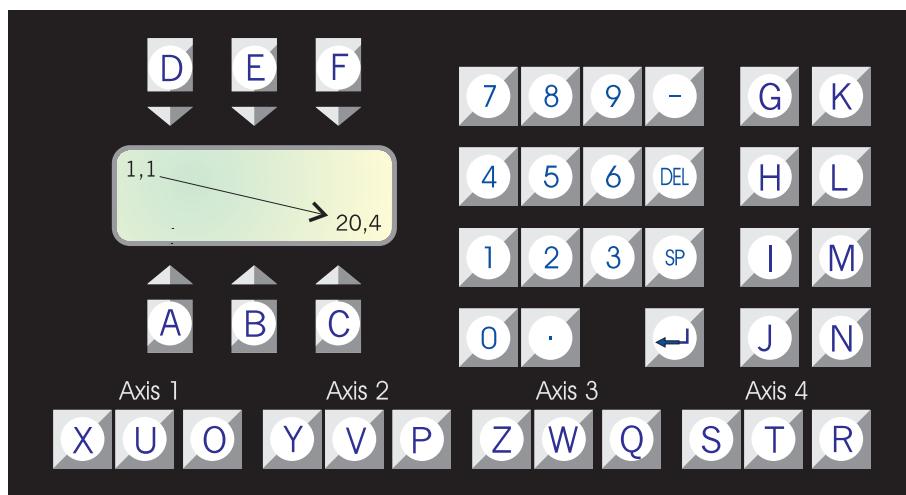
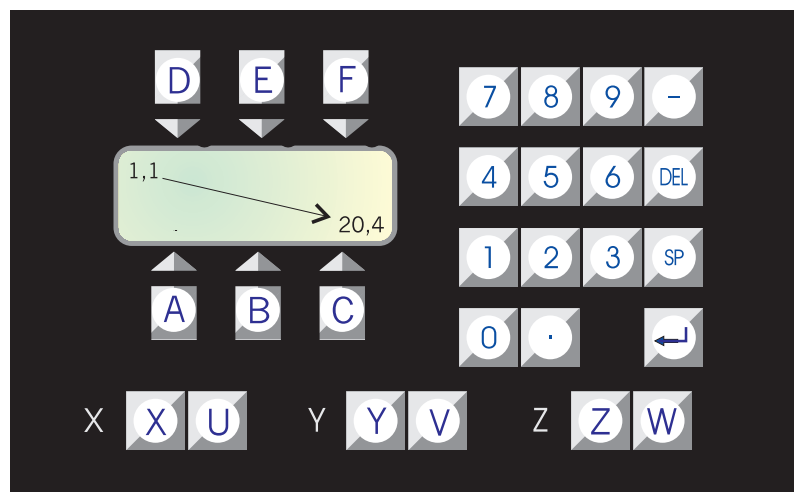


Figure 16: Keypad key mappings

All the keys return a code that corresponds to a character from a pre-defined string:

```
`.9.87FED~W4.Z56-X@0.VYU.AB1C.23 K.....OLN...P.IMTQ.....JHR.....SG"
```

In order to use the 4 axis keypad with SmartMove, the following command must be placed in the Mint program or configuration file:

```
KEYS ``.9.87FED~W4.Z56-X@0.VYU.AB1C.23 K.....OLN...P.IMTQ.....JHR.....SG"
```

All the letters are returned in upper case. For example, F1 will return the capital letter 'A' to Mint.

The keypad and display supports the following terminal I/O keywords:

Keyword	Description
BEEP	Sound the buzzer
BIN	Print a binary number
DEC	Print a decimal number
HEX	Print a hexadecimal number
BOL	Send cursor to beginning of line
CLS	Clear screen
INKEY	Read a key from the serial port buffer
INPUT	Basic formatted input
LINE	Formatted print to a specified line
LOCATE	Locate cursor at column, row
PRINT	Basic formatted print
READKEY	Read value of key currently pressed
TERM/TERMINAL	Direct output to LCD or serial port

For more details on the terminal I/O keywords in v3 Mint controllers, please refer to the Mint Programming Manual^[5].

For more details on the terminal I/O keywords in v4 Mint controllers, please refer to the Mint v4 CAN Programming Guide^[8].

The keypad interface makes it easy to build a customised operator panel, using push buttons and a display, which is exactly suited to a specific application.

The operator panel incorporates six function keys placed above and below the display, so that the function of each key can be indicated by printing a legend on the top and bottom lines of the display. This allows menu driven operator interfaces to be readily written, enabling the function of each key to change depending on the menu. The middle two lines are used for messages or operator prompts.

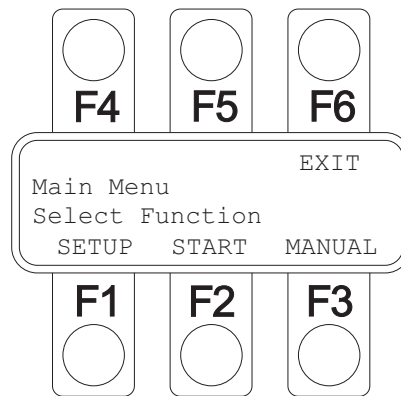


Figure 17: Operator Panel

NOTE: Care must be taken when writing to the LCD display since the display does not scroll. If the last character of the display is followed by a carriage return, the entire display will be cleared. Use a comma at the end of a print statement to suppress line feeding.

Mint Example:

```

LOOP
  LINE 1, "                EXIT"
  LINE 2, "Sample keypad"
  LINE 3, "Program"
  LINE 4, "SETUP  START  MANUAL",
  key = 0
  REPEAT
    key = INKEY
    IF key = 'A' THEN BEEP: GOSUB setup
    IF key = 'B' THEN BEEP: GOSUB start
    IF key = 'C' THEN BEEP: GOSUB manual
    IF key = 'F' THEN BEEP: END
  UNTIL key <> 0
ENDL

#setup
  REM Code here
RETURN

#start
  REM Code here
RETURN

#manual
  REM Code here
RETURN

```

Note the use of **BEEP** to provide audible feedback to the operator. Also note the comma at the end of **LINE 4** statement. This is used to suppress line feeding and clearing the screen.

Using the keypad interface, it is possible to read the currently pressed key using **READKEY**. This is unlike **INKEY**, which returns the next keypress in the serial input buffer. **READKEY** is useful for jogging motors while a key is pressed.

Mint Example:

```

LOOP
  IF READKEY = 'X' THEN JOG[0] = 10
  IF READKEY = 'U' THEN JOG[0] = -10
  IF READKEY = 0 THEN JOG[0] = 0
ENDL

```

Using the X left and right keys on the keypad, the motor will be jogged left or right while the key is pressed. When either of the keys are released (**READKEY = 0**) the motor will come to a stop.

For embedded 'C' applications on *NextMove* and *MintDrive*, a set of functions are provided (please refer to the Mint v4 Function Reference Guide^[10] for more details on the terminal I/O functions):

Function	Description
<code>get/setKeypadNode()</code>	Reads or sets the location and channel used by a <i>KeypadNode</i> .
<code>getReadKey()</code>	Reads the key currently being pressed on a <i>KeypadNode</i> .
<code>tCls()</code>	Clears the screen of the desired <i>KeypadNode</i> .
<code>tGetChar()</code>	Reads the next available character from the desired <i>KeypadNode</i> .
<code>tLocateCursor()</code>	Locates the cursor on the desired <i>KeypadNode</i> .
<code>tPrintf()</code>	Transmits a formatted string to the desired <i>KeypadNode</i> .
<code>tPutChar()</code>	Transmits a character to the desired <i>KeypadNode</i> .
<code>tPutString()</code>	Transmits an unformatted string to the desired <i>KeypadNode</i> .

Product History

8

This chapter provides a history of the different products.

- ◇ InputNode 8
- ◇ OutputNode 8
- ◇ RelayNode 8
- ◇ IoNode 24/24
- ◇ KeypadNode
- ◇ KeypadNode 4

8.1 *InputNode 8, OutputNode 8, RelayNode 8*

Revision	Date	Description of Changes
2A		Introduction of <i>InputNode 8</i> , <i>RelayNode 8</i> and <i>OutputNode 8</i>
3	June 96	Tidy-up and opportunity to improve immunity to conducted interference. Some early production used 4MHz crystals and issue 0FFB of the processor (denoted by a yellow label on the underside of the PCB); these devices can not be operated above 125kbit.

8.2 *IoNode 24/24*

Revision	Date	Description of Changes
2A	Feb 97	First production units

8.3 *KeypadNode*

Revision	Date	Description of Changes
3A	Feb 97	First production units

8.4 *KeypadNode 4*

Revision	Date	Description of Changes
1A	Jul 98	First production units

Getting Started with *SmartMove*

9

This chapter provides details on using the IoNodes with SmartMove, the 1 to 3 axis servo controller. Details for the other Mint controllers can be found in the respective Installation Manuals.

- ◇ Configuring a node
- ◇ Selecting CAN bus channels, node numbers and baud rates

Full details of the CAN Peripheral supported Mint keywords can be found in Section 10.

Before starting, it is important to make sure that *SmartMove* is CAN Peripheral ready. This can be found by typing **VER** at the Mint command line:

```
C>VER
```

The Mint version number will be displayed:

```
esMint v2.7e/SMM2/P/C/M  
Copyright ...
```

If **/C** is in the version number string, then *SmartMove* will support the CAN Peripheral products. If **/CK** is in the version number string, then *SmartMove* will also support *KeypadNode*, the CAN based operator panel. For example:

```
esMint v2.7e/SMM1/P/CK/M
```

If **/CK** is in the version number string and the version is v2.72 or higher, then *SmartMove* will also support *IoNode 24/24*, the extended I/O CAN based peripheral. For example:

```
esMint v2.72/SMM3/P/CK/M
```

9.1 Network Possibilities

CAN nodes are attached to *SmartMove* using a twisted pair cable, such as CAT5 (or cable with equivalent specification), which is daisy chained to each node, forming a network (or bus). CAN supports up to 63 nodes on the same network, but *SmartMove* limits the number of nodes it can support in software. The limitations are a maximum of 6 input nodes (*InputNode 8*), 6 output nodes (a mix of *RelayNodes* and *OutputNodes*), one *KeypadNode* and 4 *IoNode 24/24s*.

Each node on the network is identified by a unique address called the *Node ID*. The Node ID is set using software by linking the node to the *SmartMove*. Instructions for setting the Node ID are covered in later sections.

The table below covers the nodes and Node IDs supported on *SmartMove*:

Node ID	Node Type
1 - 6	<i>InputNode 8</i>
7 - 12	<i>OutputNode 8</i>
7 - 12	<i>RelayNode 8</i>
1 - 10	<i>IoNode 24/24</i>
13	<i>Reserved</i>
14	<i>KeypadNode</i>

9.2 Support for *IoNode 24/24*

SmartMove supports *IoNode 24/24*, but due to its lack of 24 bit numbers, this is supported differently to other Mint based controllers.

When the node number is set, the I/O will be mapped across 3 consecutive node numbers, as banks of 8 bits each. For this reason, the user must be careful not to add nodes to the network that will conflict with the 3 consecutive node numbers used by the *IoNode 24/24*, otherwise the relevant data may become corrupted.

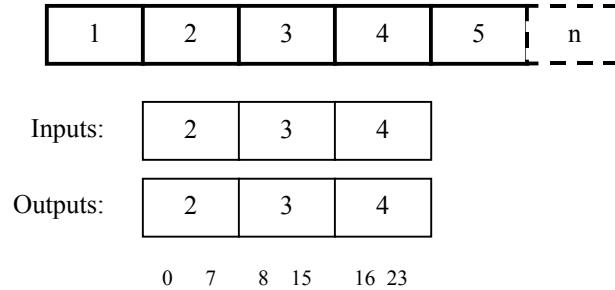


Figure 18: I/O Mapping for *IoNode 24/24*

***IoNode 24/24* is only supported with esMint v2.72 and above.**

9.3 Quick Start

It is possible to get configured in a matter of minutes. All CAN Peripherals have a default configuration, which is compatible with *SmartMove*:

CAN Peripheral Type	Default Values	
	CAN baud rate	Node ID
<i>InputNode 8</i>	125	1
<i>OutputNode 8</i>	125	7
<i>RelayNode 8</i>	125	7
<i>IoNode 24/24</i>	125	8
<i>KeypadNode and KeypadNode 4</i>	125	14

NOTE: If the CAN Peripheral's configuration has been changed, it will need to be statically reconfigured (refer to Section 9.4.5).

If multiple nodes of more than one type are to be added, then the Node IDs will require configuration so that no two CAN nodes have the same Node ID. This is covered in section 9.4.

All that is required is to power both the controller and CAN Peripheral, connect their CAN ports and start controlling remote I/O. *SmartMove* automatically adds to the network any CAN Peripherals which are present on the CAN Bus.

This following exercise, using *SmartMove 1* as the host controller, illustrates the ease of configuration. This example uses a *RelayNode 8* CAN Peripheral. Refer to the configuration diagram represented in Figure 19.

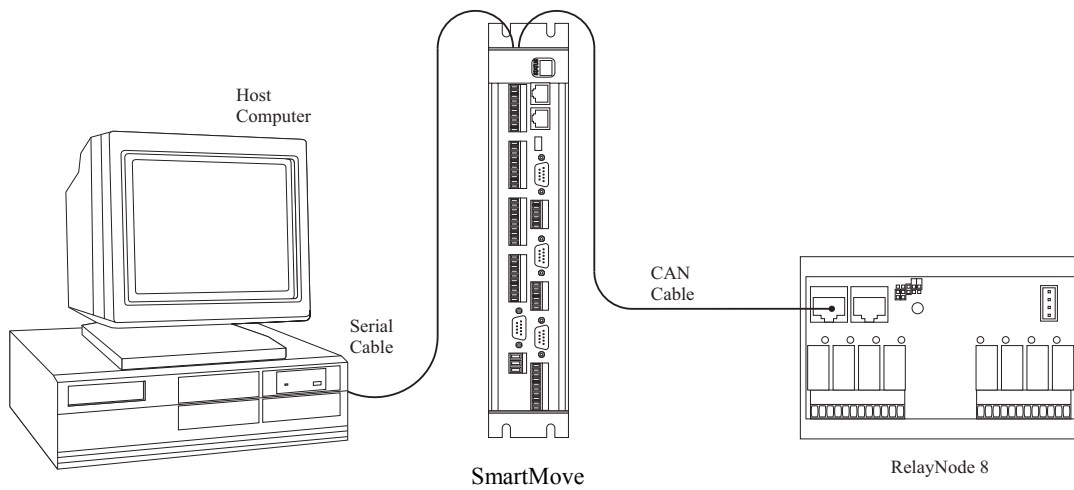


Figure 19: *SmartMove* connected to one *RelayNode 8*

9.3.1 Jumper settings

1. Make sure that jumpers JP1 and JP2 on the *RelayNode 8* are fitted in position 1, as shown in Figure 20.
2. Make sure that jumper JP3 on the *RelayNode 8* is fitted, as shown in Figure 20.
3. Make sure that jumpers JP4 and JP5 on the *RelayNode 8* are not fitted. Refer to Figure 20.

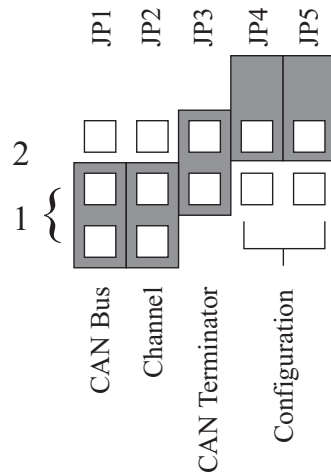


Figure 20: Jumper Settings

9.3.2 Connections and Configuration

1. Connect the *RelayNode 8* and *SmartMove* using a CAN cable.
2. Connect the *RelayNode 8* to a 24V dc supply.
3. Connect the *SmartMove* to a suitable supply. (It is possible to use the same supply being used to power the *RelayNode 8*).
4. Connect *SmartMove* to a PC via an RS232 cable and start *cTERM for Windows*. For details on using *cTERM for Windows* please refer to the *SmartMove Installation Manual*^[1].
5. Make sure that the DIP switch numbered '5' on the *SmartMove* front panel is in the *on* position.
6. Switch on the power supplies to both *SmartMove* and the *RelayNode 8*. (The order in which this is done is not important.)

cTERM displays the Mint sign on message, for example:

```
esMint v2.7e/SMM1/P/CK/M
Copyright ...
```

7. The LED on the *RelayNode 8* will start to flash green, approximately once every half-second. Each flash indicates that the *RelayNode 8* is participating in CAN activity.
8. Each time a CAN Peripheral is added, a *node live* event occurs. Type the following at the Mint command line:

```
? CANSTATUS
```

This will monitor the CAN Bus, reporting any events or errors. Reading the CAN event will clear the buffer. Refer to section 10.1.1.4.

9. To confirm that *SmartMove* is able to communicate with the *RelayNode 8*. At the Mint command line type:

```
? NODELIVE.7
```

10. *SmartMove* replies with the value **3**, which indicates that the node with ID 7 is a *RelayNode 8*. It is now possible to freely control the outputs. For example, to turn output 3 on enter the command:

```
REMOTEOUT.7.3=1
```

Refer to Section 10 of this manual for a full description of the *SmartMove* Mint keywords which support CAN Peripherals.

9.4 SmartMove and CAN Peripherals

9.4.1 Selection of CAN Channel

SmartMove supports CAN Peripherals on CAN Bus channel 1.

When connecting CAN Peripherals to a *SmartMove* controller, CAN Bus channel 1 must be selected on the CAN Peripheral by fitting jumpers JP1 and JP2 to position 1, as shown in Figure 21.

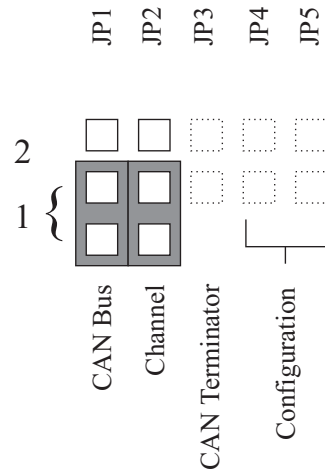


Figure 21: CAN Channel Jumper Selection

9.4.2 Selection of CAN Baud Rate

The CAN Baud rate is the rate at which data is transferred over the network. The controller and CAN Peripherals use a default CAN transmission rate of 125 kbit/s. Although it is possible to alter this, it is unlikely to be necessary. At this baud rate the network can be up to 500m (1640ft) long. Refer to Section 2.1.1 for maximum bit rates vs cable lengths.

If it is found necessary to change from the default baud rate, the CAN Peripherals must be statically configured.

9.4.3 Selection of Node ID

Each CAN Peripheral must be given a unique Node ID within the network. The Node ID is used to filter out CAN messages that are directed at other nodes and is simply designated by a number. The Node ID is assigned to the node using *SmartMove* by a means called *Static Configuration*. (Refer to section 9.4.5). The rules that govern how Node IDs should be assigned for a *SmartMove* host are:

- *InputNode 8* nodes may have a Node ID of between 1 and 6.
- *OutputNode 8* and *RelayNode 8* nodes may have a Node ID of between 7 and 12.
- *KeypadNode* must have a Node ID of 14 (default).
- *IoNode 24/24* nodes may have a Node ID of between 1 and 10.
- No two nodes may have the same Node ID in the same network as shown below. This will result in spurious activity on the bus resulting in the bus going down. Refer to Figure 22.

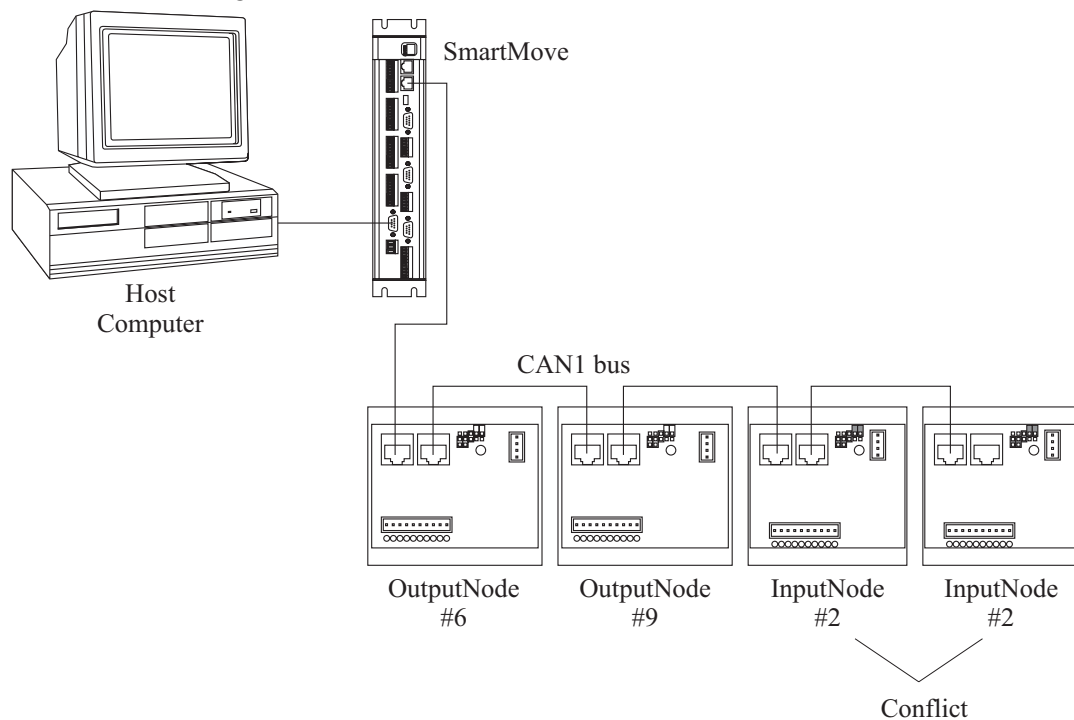


Figure 22: Network with conflicting Node IDs

The nodes do not have to have Node IDs that reflect the order in which they are connected on the network, as can be seen in Figure 23.

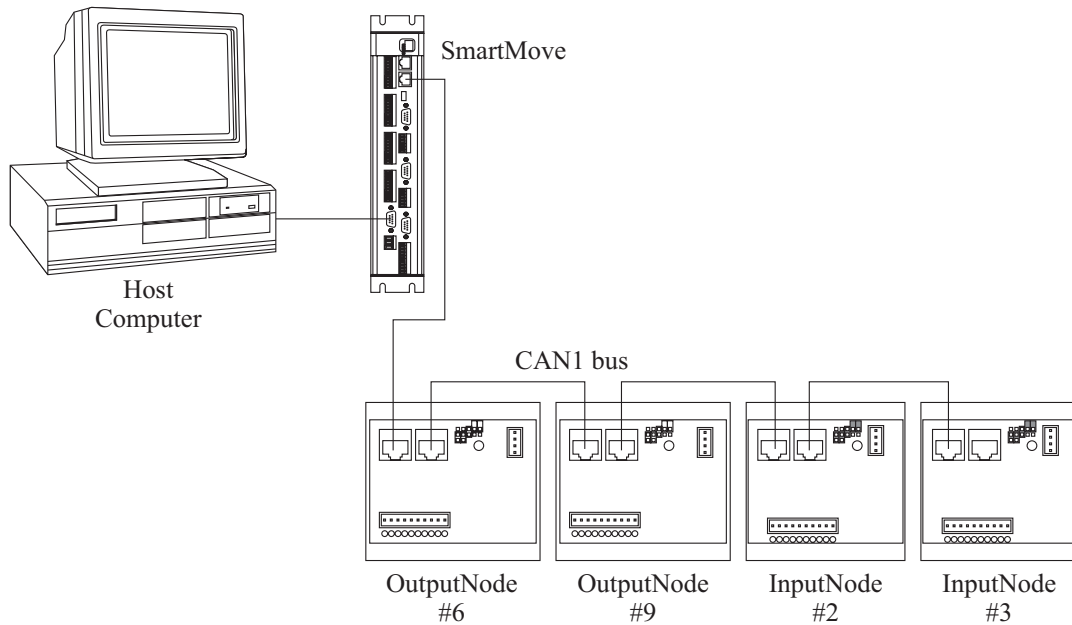


Figure 23: Typical network

To set a Node ID, the node must be statically configured as described in Section 9.4.5.

9.4.4 Network Termination

Termination resistors must be fitted at the ends of the network to reduce signal reflection. The controllers and CAN Peripherals are fitted with termination resistors specifically for this purpose. Refer to Section 2.1.3 for full details.

On *SmartMove*, the terminator is selected by turning on switch '5' of the 5 position DIP switch accessible at the front panel.

On the CAN Peripherals, the terminator is selected by fitting a jumper to JP3, as shown in Figure 24.

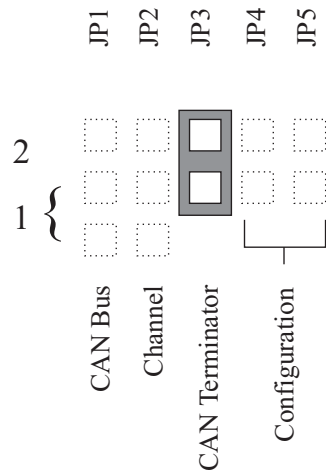


Figure 24: Network Termination Jumper settings

9.4.5 Static Configuration

A CAN Peripheral possesses two attributes that may be altered:

- CAN communication baud rate (default 125 Kbit/s)
- Node ID. The defaults are shown:

Node	Default Node ID
<i>InputNode 8</i>	1
<i>OutputNode 8</i>	7
<i>RelayNode 8</i>	7
<i>IoNode 24/24</i>	8
<i>KeypadNode and KeypadNode 4</i>	14

These attributes are altered via the CAN Bus. There must be only two devices in the network – the node to be configured and a CAN-based configuration tool (in this case *SmartMove*). Because this configuration has to take place with the node removed from the full network, it is called *Static Configuration*. Refer to Figure 25.

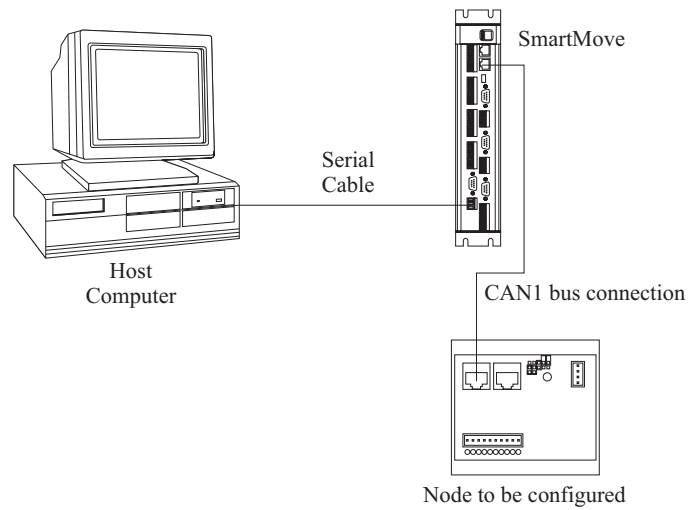


Figure 25: Node Configuration

To statically configure a CAN Peripheral, refer to Figure 26.

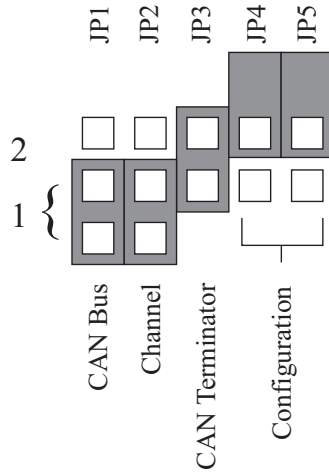


Figure 26: Static Configuration Jumper Position

1. Make sure that jumpers JP1 and JP2 (CAN Bus channel) on the CAN Peripheral are fitted in position 1 (CAN1), as shown in Figure 26.
2. Make sure that jumper JP3 (CAN terminator) on the CAN Peripheral is fitted, as shown in Figure 26.
3. Make sure that jumpers JP4 and JP5 (Configuration) on the CAN Peripheral are fitted, as shown in Figure 26.

4. Use a CAN cable to make a CAN connection between *SmartMove* and the CAN Peripheral.
5. Start *cTERM for Windows* and open the *Terminal Window* (refer to the *SmartMove Installation Manual*^[1]).
6. Power up *SmartMove* and 'break' any program, which may be running by typing **[ctrl]+[E]** from the terminal window.
7. Power up the CAN Peripheral. The LED on the node will flash red approximately every half a second.
8. At the Mint command line type:

```
CANBAUD = 125
REMOTENODE = <nodeID>
```

where **<nodeID>** is the number that will be used to reference the node in future.
9. If not using the default baud rate, at the Mint command line type:

```
REMOTEBAUD = <baud>
```

where **<baud>** is the CAN baud rate intended to be used.
10. Power down the CAN Peripheral.
11. Remove jumpers from JP3, JP4 and JP5 on the CAN Peripheral.

9.4.6 Normal Operation

When involved in CAN communication the status LEDs on the CAN Peripherals flash green. The controller operates a *node guarding* procedure in which all nodes are regularly sent a CAN message. This procedure is seen on the CAN Peripheral as a flash of the green LED approximately once every half-second.

For further details on the operation of the CAN status LED, refer to Section 2.1.5.

9.5 An Example Network

To illustrate the steps required in putting a CAN network together, a network comprising the following elements can be considered:

- 1 *SmartMove 3* controller
- 2 *InputNode* 8 nodes
- 2 *OutputNode* 8 nodes

- 1 *RelayNode* 8 node
- 1 *KeypadNode* node

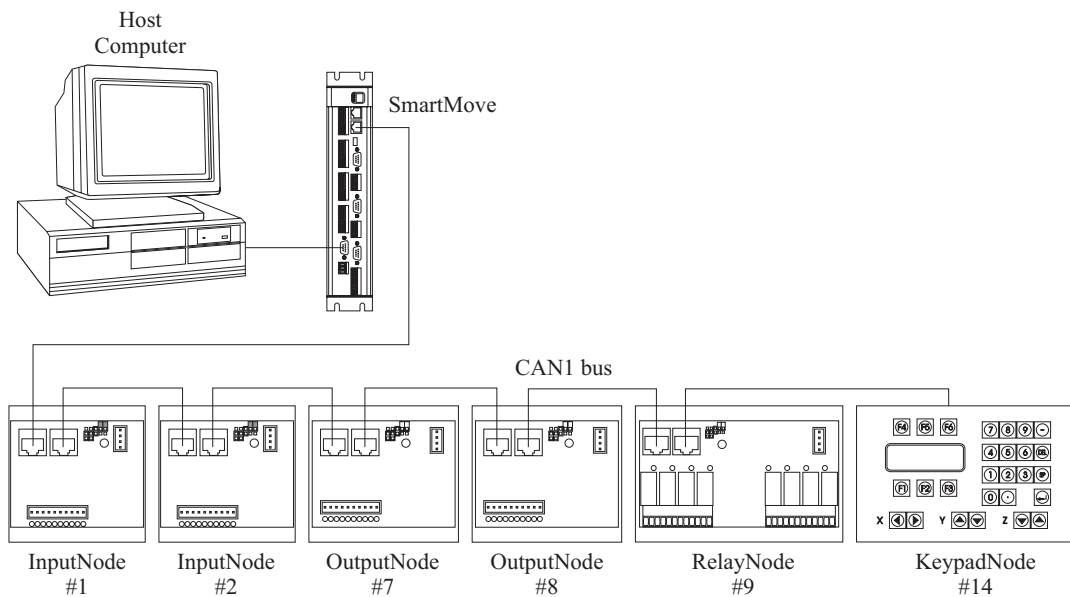


Figure 27: Example multi-node network

The nodes are daisy-chained together with the controller at one end of the network and the *KeypadNode* at the other end of the network. The controller is housed in a control cabinet located a short distance from the machine. The CAN Peripherals are distributed over the machine locally to their respective actuators and sensors. Each node is supplied with 24Vdc from a bus supplying the machine. The CAN baud rate used is the default 125 Kbit/s.

1. Statically configure the two *InputNode* 8 nodes with Node IDs 1 and 2 (refer to Section 9.4.5).
2. Statically configure the two *OutputNode* 8 nodes with Node IDs 7 and 8 (refer to Section 9.4.5).
3. Statically configure the *RelayNode* 8 node with Node ID 9 (refer to Section 9.4.5).
4. Statically configure the *KeypadNode* node with Node ID 14 (refer to Section 9.4.5).
5. Terminate the network:
 - ◇ On *SmartMove*, the terminator is selected by turning on switch '5' of the 5 pole DIP switch accessible on the front panel.

- ◇ As the *KeypadNode* is at the end of the network its terminator should be selected by fitting a jumper to JP3.
 - ◇ The remaining nodes should have their terminator switched out by removing the jumper on JP3.
6. Position the CAN Peripherals and controller on the machine.
 7. Power up the system.

9.6 Using a *KeypadNode*

Before connecting a *KeypadNode* to *SmartMove* it is important to check that Mint supports the *KeypadNode*. This can be done by checking the version number. From the command line type **VER**. The version will be printed to the screen:

```
esMint v2.7e/SMM1/P/CK
```

If **/CK** is in the version number, then *SmartMove* is ready to support *KeypadNode*. If not, then a local supplier should be contacted for a firmware upgrade.

When connecting a *KeypadNode* to the CAN1 bus, the node is added automatically to the network. Once the *KeypadNode* has been configured onto the bus, the Mint terminal keywords such as **PRINT**, **INKEY**, **LOCATE** and **CLS** can be used. For example, the following command sequence can be tried:

```
CLS : ? "Hello"
```

Hello appears at the top left hand corner of the LCD display.

The keyword **TERM** is used to set the default terminal input and output channels for *SmartMove*. By default, it is assigned to communicate with all supported terminal devices, therefore allowing all terminal keywords to be used with the *KeypadNode*. Also, any key presses on the *KeypadNode* will be echoed to the host display and vice-versa. The **TERM** keyword can be used to suppress communications with *KeypadNode*.

Refer to Section 7.2.1 for more details on *KeypadNode*.

Mint Support for *SmartMove*

10

This chapter provides details on the Mint keywords supporting the CAN Peripheral devices on SmartMove.

- ◇ Error handling.
- ◇ Keyword reference.

10.1 Errors And Error Handling

CAN support introduces two new categories of error:

- Synchronous CAN errors
- Asynchronous CAN events

Synchronous CAN errors occur when keywords associated with CAN Bus communication are executed. An example is when an attempt is made to set an output on an output node and the remote node fails to reply within a timeout period.

Asynchronous CAN events can occur at any time and are the result of, usually, unexpected conditions. An example is where a remote node becomes disconnected from the CAN Bus due to faulty wiring.

There are three new Mint keywords associated with CAN error detection and handling:

- **CANSTATUS**
- **REMOTEERROR**
- **STATUSNODE**

When asynchronous CAN errors and events occur, the keyword **CANSTATUS** can be used to determine which errors and events have occurred.

If the error or event relates to a particular node, the **STATUSNODE** keyword can be used to find the Node ID of that node.

If the event indicates that a remote node has experienced an error, the **REMOTEERROR** keyword may be used to determine the cause of the error on the remote node.

10.1.1 Error Messages

The following sections describe the error codes and associated error messages produced by the controllers.

10.1.1.1 Synchronous CAN Error Messages

These are given in the format of error code number followed by error message. Refer to the table in Section 10.1.1.3.

63: CAN timeout at node <nodeID>

A remote node did not respond to a CAN Bus message within a predefined timeout period.

- Faulty network wiring.

- Failure to change baud rate properly.

Check that the network is intact. CAN timeouts may be accompanied by spurious *Live/Dead* asynchronous CAN events.

67: Wrong type of node (nodeID)

The remote node does not support the action requested.

- Trying to set an output on an input node.
- Trying to access the node if it is not on the bus (not *Live*).

68: Not in config mode at node <nodeID>

The remote node is not able to accept configuration messages (**REMOTEBAUD**, **REMOTENODE**).

- There must be only one remote node on the bus.
- The remote node must be powered up with jumpers fitted to JP4 and JP5.
- An error occurred while saving data on the remote node.

69: CAN operation failed at node <nodeID>

- Failed to set remote baud rate or remote Node ID.

10.1.1.2 Asynchronous CAN Error Messages

70: Asynchronous CAN error/event

Asynchronous CAN bus errors and events can occur at any time, not only when a Mint command is executed. The errors and events are as follows:

- **Abnormal bus errors.** The CAN controller is experiencing an abnormal rate of failed transmissions and receptions¹.
- **Bus off.** The CAN controller has gone into the ‘bus-off’ state because of a high rate of failed transmissions and receptions¹.
- **Receive overrun.** CAN Bus messages are arriving more frequently than they can be serviced.
- **Remote node has died.** The node guarding services on the controller have detected that a remote node that was previously present on the network is no longer present.

¹ These states are defined in the CAN standard.

- **Remote node has become live.** The node guarding services on the controller have detected that a remote node has appeared on the network.
- **Remote node error.** A remote node has experienced an error condition.

More than one of the above errors and events can be simultaneously reported. The **CANSTATUS** keyword returns a bit map which indicates which of the above errors/events have occurred. In the case of the *remote node* bits the **STATUSNODE** keyword can be used to determine the Node ID of the node in question.

10.1.1.3 Categories Of Error

CAN errors can be trapped by the Mint **#ONERROR** routine. The types of errors that can be trapped are:

- CAN timeout
- Illegal operations on CAN Peripherals
- CAN Peripheral errors
- Network errors/events

Extensions to the **ERR** error codes are as follows:

Synchronous CAN errors: These errors may occur during a CAN read/write command	
<i>reserved</i>	55 . . 62
CAN timeout	63
<i>reserved</i>	64 . . 66
Invalid CAN device	67
CAN node configuration	68
CAN operation failed	69
Asynchronous CAN errors and events: These errors/events may occur at any time. The error condition may be further investigated by reading CANSTATUS and STATUSNODE.	
Asynchronous CAN error/event	70

Please consult Section 11 of the Mint Programming Manual^[5] for full details on error messages.

10.1.1.4 Example ONERROR routine:

```

REM Example ONERROR skeleton routine showing enhancements to the Mint
REM language.
#ONERROR
  REM Synchronous or system errors
  IF ERR>=0 AND ERR<=9 DO
    ?"System error"
    PAUSE INKEY=` `
    RUN
  ENDF

  REM Asynchronous or motion errors
  IF ERR>=10 AND ERR<=19 DO
    IF ERROR.0= limit OR ERROR.1=_limit DO
      DISLIMIT[0,1]
      SERVOFF[0,1]
      PAUSE !LIMIT.0 AND !LIMIT.1
      SERVON[0,1]
      ENLIMIT[0,1]
    ENDF
  ENDF

  REM Motion keyword interpretation errors
  IF ERR>=20 AND ERR<=24 DO
    ENDF

  REM Errors 25 to 34 are compiler errors which cannot be trapped

  REM Interpreter errors
  IF ERR>=35 AND ERR<=49 DO
    ENDF

  REM CAN Synchronous errors
  IF ERR>=55 AND ERR<=69 DO
    IF ERR=63 THEN ?"CAN timeout"
    IF ERR=67 THEN ?"Invalid CAN device"
    IF ERR=68 THEN ?"Node not in config mode / EEPROM failed"
    IF ERR=69 THEN ?"CAN operation failed"
  ENDF

  REM CAN Asynchronous errors/events
  IF ERR=70 DO
    status = CANSTATUS
    IF status AND _aberr THEN ?"ABERR  "
    IF status AND _busoff THEN ?"BUSOFF  "
    IF status AND _over THEN ?"OVERRUN  "
    IF status AND (_live+_dead+_error) DO
      node = STATUSNODE
      IF status AND _dead THEN ?"DEAD on node ",node
      IF status AND _live THEN ?"LIVE on node ",node
      IF status AND _error DO
        ?"ERROR on node ",node,
        rError = REMOTEERROR.node
        IF rError= 0 THEN ?" (No error)"
        IF rError= 1 THEN ?" (EEPROM write timed out)"
        IF rError= 2 THEN ?" (EEPROM didn't ACK)"
        IF rError= 3 THEN ?" (Bit timing index out of range)"
      ENDF
    ENDF
  ENDF

```

```
IF rError= 4 THEN ?" (Debounce samples out of range)"
IF rError= 5 THEN ?" (CAN stuck in reset)"
IF rError= 6 THEN ?" (Node guarding timed out)"
IF rError= 7 THEN ?" (Illegal operation)"
IF rError= 8 THEN ?" (Bus off)"
IF rError= 9 THEN ?" (Error count)"
IF rError=10 THEN ?" (Overrun)"
IF rError=11 THEN ?" (Output error)"
ENDIF
ENDIF
ENDIF
RETURN
```

NOTE: once set, ERR is not cleared down and retains a value equivalent to the last reported error.

10.2 Mint Keyword Summary

Keyword	Description
Configuration Keywords	
CANBAUD	set the controller CAN Bus baud rate
REMOTREAL	sets or returns active high I/O
REMOTEBAUD	set the remote node CAN Bus baud rate
REMOTERESET	set the <i>IoNode 24/24</i> into it's reset state
Error Keywords	
CANSTATUS	returns CAN Bus related error/event status
NODELIVE	indicate the type of node with a given Node ID
REMOTEERROR	returns error condition from remote node
STATUSNODE	Node ID of node that caused asynchronous CAN error
Input/Output Keywords	
REMOTEIN	return state of inputs from a remote node
REMOTEOUT	set or return the state of outputs on a remote node

10.3 Mint Keyword Reference

CANBAUD/CB

Purpose:

To set or return the CAN bit rate on the host controller

Format:

```
CANBAUD = <expression>
v = CANBAUD
```

Abbr.	Read	Write	Command	Scaled	Default	Range
CB	c	c			125	10, 20, 50, 125, 250, 500, 800, 1000

All devices communicating on the CAN network must be configured to use the same bit rate. The bit rate is specified in Kbit/s. The default bit rate is 125Kbit/s. The CAN bit rate is also called *CAN bus speed* or *CAN baud rate*.

Writing to **CANBAUD** configures the local CAN controller to operate at the bit rate specified in **<expression>**.

Reading **CANBAUD** returns the bit rate currently in use.

The bit rate used by other CAN devices should be the same as that used by the controller. The way in which individual CAN devices may be configured will vary. Consult the support documentation for the particular device being used. Baldor Optimised Control's CAN based I/O nodes may be remotely configured by the controller using the keywords **REMOTEBAUD** and **REMOTENODE**.

Errors:

Out of range - **<expression>** does not evaluate to one of the values in the list.

Example:

```
CANBAUD = 1000
```

will set the local CAN interface to operate at 1000 Kbit/s.

See Also:

REMOTEBAUD, REMOTENODE

CANSTATUS/CST

Purpose:

To read the cause(s) of an asynchronous CAN error or event on the host controller

Format:

v = CANSTATUS

Abbr.	Read	Write	Command	Scaled	Default	Range
CST	c				-	-

CANSTATUS is a bit map that indicates the cause of an asynchronous CAN error or event.

The meanings of the bits are as follows:

7	6	5	4	3	2	1	0
-	ERROR	GUARD	LIVE	DEAD	OVER	BUSOFF	ABERR

Error Bit	Value	Description
ABERR	1	Abnormal bus errors. The CAN controller is experiencing an abnormal rate of failed transmissions and receptions.
BUSOFF	2	Bus off. The CAN controller has gone into the <i>bus-off</i> state because of a high rate of failed transmissions and receptions.
OVER	4	Receive overrun. CAN-bus messages are arriving more frequently than they can be serviced.
DEAD	8	Node has died. This error can only occur on the CAN Bus Administrator. The node-guarding services on the CBA have detected that a remote node that was previously present on the CAN Bus is no longer present.
LIVE	16	Node has become live. This event can only occur on the CAN Bus Administrator. The node-guarding services on the CBA have detected that a remote node has appeared on the CAN Bus.
GUARD	32	Guarding failed. This error can only occur on the CAN Bus Administrator. The CBA has been unable to operate the node-guarding services.
ERROR	64	Node has experienced an error. This event can only occur on the CAN Bus Administrator. The CBA has detected that a remote node has experienced an error.

More than one of the above errors and events can be simultaneously reported. The **CANSTATUS** keyword returns a bit map which indicates which of the above errors/events have occurred. Furthermore, in the case of the **DEAD** error and the **LIVE** and **ERROR** events, the **STATUSNODE** keyword can be used to find out the ID of the node in question.

Reading **CANSTATUS** clears it. The **GUARD**, **OVER**, **BUSOFF**, and **ABERR** bits may become set again immediately if the associated error condition persists. The **LIVE**, **DEAD** and **ERROR** bits remain at 0 until **STATUSNODE** is read. **STATUSNODE** gives the ID of the node that became *live*, *died* or experienced an error. If a similar event occurs on a second node in the meantime, reading **STATUSNODE** will immediately cause the **DEAD**, **LIVE** or **ERROR** bit in **CANSTATUS** to become set again. Reading **STATUSNODE** for a second time will then give the ID of the second node.

Errors:

CANSTATUS does not return any errors.

Example:

Refer to the example **#ONERROR** routine.

See Also:

ERR, **STATUSNODE**

NODELIVE/NL

Purpose:

Returns the type of node detected at a given Node ID on the CAN Bus.

Format:

v = **NODELIVE.<nodeId>**

Abbr.	Read	Write	Command	Scaled	Default	Range
NL	C				-	-

<nodeID> is an expression that evaluates to the Node ID on the CAN Bus of the node whose type is to be returned.

When a remote node is detected on the CAN Bus the controller considers it to be *live*. If no node has been found, the controller considers the node to be *dead*. This function indicates that a node is *live* by returning its type number.

The type codes currently supported are:

Type value	Node type
0	Not present, dead
1	8 digital input node
2	8 digital output node
3	8 relay output node
8	24/24 IO Node
9	KeypadNode

When a node becomes live or dies, the event/error is reported by the **LIVE** or **DEAD** bits being set in **CANSTATUS**. Keeping track of which nodes are *live* or *dead* can result in unwieldy code and therefore the **NODELIVE** keyword provides a quick and easy means of checking that a node is *live* and the type of node that occupies that Node ID.

Errors:

CAN Node ID out of range (nodeID) - <nodeID> is out of range.

Example:

```
PAUSE NODELIVE.5
```

will wait for a remote CAN node with Node ID #5 to come onto the bus.

See Also:

CANSTATUS, STATUSNODE

REMOTEAL/RL

Purpose:

To set or return the state for remote active I/O channels.

CAN based I/O nodes that have digital inputs and/or outputs may have these I/O configured to be 'active' when *on* or 'active' when *off*. In this context, *on* means that the I/O channel is conducting, *off* means that the I/O channel is non-conducting. Normally, inputs and outputs are considered active when ON. In some circumstances it may be more appropriate that the sense of the I/O channel is inverted. This is more often applied to inputs than to outputs. Note that the power-off and error state of outputs is always *off*.

The format of the keyword is dependant on the type of CAN node that is being accessed.

Format for *InputNode 8*, *OutputNode 8* and *RelayNode 8*:

```
REMOTEAL.<nodeId> = <expression>
v = REMOTEAL.<nodeID>
```

Abbr.	Read	Write	Command	Scaled	Default	Range
RL	C	C			-	-

<nodeID> is an expression that identifies the node on the bus. This is the Node ID that was stored in the node during its static configuration.

Writing to **REMOTEAL** will selectively configure individual I/O channels to be active when *on* or *off*. If the operation is valid the remote node will save this configuration in non-volatile memory for use on subsequent power-ups.

Reading **REMOTEAL** will return the I/O configuration currently used by <nodeID>.

The data passed to and from the remote node is a bit-map in which each bit position corresponds to an I/O channel. A '1' in a bit position means that the I/O channel is active when *on*. A '0' in a bit position means that the I/O channel is active when *off*.

Format for *IoNode 24/24*:

```
REMOTEAL.<nodeId>.<bank> = <expression>
v = REMOTEAL.<nodeID>.<bank>
```

Abbr.	Read	Write	Command	Scaled	Default	Range
RL	C	C			-	0 to 255

<nodeID> is an expression that identifies the node on the bus. This is the Node ID that was stored in the node during its static configuration.

The *IoNode 24/24*'s I/O data is stored as 8-bit blocks over 3 consecutive node numbers.

<block> is an expression that identifies the block of I/O data that is required.

For example, if the node number for *IoNode 24/24* was set to 2, the active level (for both inputs and outputs) can be read using:

```
myActLev1 = REMOTEAL.2.0 : REM Read first 8 bits      (for I/O 0 to 7)
myActLev2 = REMOTEAL.2.1 : REM Read next 8 bits      (for I/O 8 to 15)
myActLev3 = REMOTEAL.2.2 : REM Read remaining 8 bits (for I/O 16 to 24)
```

Writing to **REMOTEAL** will selectively configure individual I/O channels to be active when *on* or *off*.

Reading **REMOTEAL** will return the I/O configuration currently used by <nodeID>.

The data passed to and from the remote node is a bit-map in which each bit position corresponds to an I/O channel. A '1' in a bit position means that the I/O channel is active when *on*. A '0' in a bit position means that the I/O channel is active when *off*.

Errors:

CAN Node ID out of range (nodeID) - <nodeID> is out of range for this type of node.

Wrong type of node <nodeID> - the node whose ID is <nodeID> is not a CAN Peripheral.

CAN operation failed at node <nodeID> - the controller was unable to send a message or the remote node reported an error.

CAN timeout at node <nodeID> - the remote node failed to respond in time.

Example:

```
REMOTEAL.4 = 011110000
```

will set inputs 0 to 3 to be reported active when *off* and inputs 4 to 7 to be reported active when *on*. This is configured on the input node whose Node ID is 4.

See Also:

REMOTEIN, REMOTEOUT

REMOTEBAUD/RB

Purpose:

To set the CAN bit rate for a remote CAN node.

Format:

```
REMOTEBAUD = <expression>
```

Abbr.	Read	Write	Command	Scaled	Default	Range
RB		C			125	10, 20, 50, 125, 250, 500, 800, 1000

All devices that communicate using the CAN network must be configured to use the same bit rate. The bit rate is specified in Kbit/s. The default bit rate is 125Kbit/s. The CAN bit rate is also called *CAN bus speed* or *CAN baud rate*.

REMOTEBAUD is used to configure a single remote CAN based node to operate at the bit rate specified in <expression>. If the remote node is successfully configured the local bit rate is changed to match and the remote node will store the new bit rate in non-volatile memory for use on subsequent power-ups. Note that the host controller will always power up with the default bit rate.

Because of the CAN messaging involved, there must be exactly one other node in the CAN network in addition to the controller. The remote node must have been powered up in *Configuration mode* at a known bit rate (refer to section 9.4.5).

Errors:

Out of range - `<expression>` does not evaluate to one of the values in the list.

CAN operation failed - the controller was unable to send a message or the remote node reported an error.

Failed to save config - the remote node failed to save the configuration to non-volatile memory.

CAN Timeout - the remote node is not in *Configuration mode*.

Example:

```
REMOTEBAUD = 1000
```

will set the remote CAN I/O node and controller to communicate at 1000Kbit/s. The remote I/O node must be in *Configuration mode* and must be the only node in the network.

See Also:

CANBAUD, **REMOTENODE**

REMOTERROR/RR

Purpose:

To read the error status from a remote CAN node.

Format:

```
v = REMOTERROR.<nodeID>
```

Abbr.	Read	Write	Command	Scaled	Default	Range
RR	c				-	-

The exact sequence of events that occurs as a result of a remote node experiencing an error will depend upon the type of remote node. Typically though, the node will take whatever action is appropriate to make its local system safe then report the error condition via CAN. Many remote nodes will report a change in error.

When a remote node reports a change in error the **ERROR** bit will be set in **CANSTATUS** and the ID of the node that reported the error may be read using **STATUSNODE**. The cause of the error event may be determined by reading **REMOTERROR**. The value returned is node specific.

`<nodeID>` is an expression that identifies the node on the bus. This will be the Node ID that was stored in the node during its static configuration.

Errors:

Wrong type of node <nodeID> - the node whose ID is <nodeID> is not a CAN Peripheral.

Example:

Refer to the #ONERROR example.

See Also:

CANSTATUS, STATUSNODE

REMOTEIN/RI

Purpose:

Read the state of digital inputs from a CAN based input node. The keyword format is dependant on the type of CAN node that is being accessed.

Format for *InputNode 8*, *OutputNode 8* and *RelayNode 8*:

`v = REMOTEIN.<nodeId>[.<channel>]`

Abbr.	Read	Write	Command	Scaled	Default	Range
RI	C				-	-

<nodeID> is an expression that identifies the node on the bus. This is the Node ID that was stored in the node during its static configuration.

If the optional specifier <channel> is omitted the value returned is a bit-map of the remote inputs that are currently active. An active input is reported as a '1' in the corresponding bit position.

If the optional specifier <channel> is given the value returned is the state of the individual input channel. '1' indicates that the input is active.

The active state of each input may be configured using the keyword **REMOTEA**.

Example:

`PAUSE REMOTEIN.4.3`

will wait for input #3 on node #4 to become active.

Format for *IoNode 24/24*:

`v = REMOTEIN.<nodeId>.<block>`

Abbr.	Read	Write	Command	Scaled	Default	Range
RI	C				-	0 to 255

<nodeID> is an expression that identifies the node on the bus. This is the Node ID that was stored in the node during its static configuration.

The *IoNode 24/24*'s I/O data is stored internally in *SmartMove* as 8-bit blocks over 3 consecutive node numbers, although only the first node number is used to access the data in Mint. <block> is an expression that identifies the block of I/O data that is required.

The active state of each input may be configured using the keyword **REMOTREAL**.

Example:

For example, if the node number for *IoNode 24/24* was set to 2, the inputs can be read using:

```
myIn1 = REMOTEIN.2.0 : REM Read first 8 bits      (inputs 0 to 7)
myIn2 = REMOTEIN.2.1 : REM Read next 8 bits      (inputs 8 to 15)
myIn3 = REMOTEIN.2.2 : REM Read remaining 8 bits (inputs 16 to 24)
```

Errors:

CAN Node ID out of range (nodeID) - <nodeID> is out of range for this type of node.

Wrong type of node <nodeID> - the node whose ID is <nodeID> is not an input node.

Invalid I/O channel (channel) - <channel> is not valid for the node referenced.

See Also:

REMOTREAL

REMOTENODE/RN

Purpose:

Set the Node ID of a remote CAN node.

Format:

```
REMOTENODE = <expression>
```

Abbr.	Read	Write	Command	Scaled	Default	Range
RN		C			-	1 to 12

Mint references individual nodes on the CAN Bus using a unique number called the *Node ID*. The node must be told what its Node ID is in order to receive messages sent to it. This is a once only configuration task.

REMOTENODE is used to set the Node ID of a single remote CAN based node to the number specified by **<expression>**. If the remote node is successfully configured it will store the new Node ID in non-volatile memory for use on subsequent power-ups.

Because of the CAN messaging involved, there must be exactly one other node in the CAN network in addition to the controller. The remote node must have been powered up in *Configuration mode* and at a known bit rate.

With IoNode 24/24, the I/O data is stored in 3 consecutive node numbers. For example, given a node number of 3, the I/O data is stored as 8 bit blocks to node numbers 3, 4 and 5.

Errors:

Out of range - **<expression>** does not evaluate to a valid Node ID.

CAN operation failed - the controller was unable to send a message or the remote node reported an error.

CAN timeout - the remote node is not in *Configuration mode*.

Failed to save config - the remote node failed to save the configuration to non-volatile memory.

Example:

```
REMOTENODE = 3
```

will set the remote input node's Node ID to '3'. The remote I/O node must be in *Configuration mode* and must be the only node in the network.

See Also:

CANBAUD, REMOTEBAUD

REMOTEOUT/RO

Purpose:

Read or set the state of digital outputs on a remote CAN I/O node. The keyword format is dependant on the type of CAN node that is being accessed.

Format for *InputNode 8, OutputNode 8* and *RelayNode 8*:

```
REMOTEOUT.<nodeId>[.<channel>] = <expression>
v = REMOTEOUT.<nodeId>[.<channel>]
```

Abbr.	Read	Write	Command	Scaled	Default	Range
RO	c	c			-	-

<nodeID> is an expression that identifies the node on the bus. This will be the Node ID that was stored in the node during its static configuration.

If the optional specifier <channel> is omitted the value returned/written is a bit-map of the remote outputs that are/should be currently active. An active output is reported as a '1' in the corresponding bit position.

If the optional specifier <channel> is given the value returned/written is the current / desired state of the individual output channel. '1' indicates that the output is active.

The active state of each output may be configured using the keyword **REMOTREAL**.

Example:

```
REMOTOUT.7.3 = 1
```

will turn on output #3 on node #7.

Format for *IoNode 24/24*:

```
REMOTOUT.<nodeId>.<block> = <expression>
v = REMOTOUT.<nodeId>.<block>
```

Abbr.	Read	Write	Command	Scaled	Default	Range
RO	c	c			-	0 to 255

<nodeID> is an expression that identifies the node on the bus. This will be the Node ID that was stored in the node during its static configuration.

The *IoNode 24/24*'s I/O data is stored internally in *SmartMove* as 8-bit blocks over 3 consecutive node numbers, although only the first node number is used to access the data in Mint. <block> is an expression that identifies the block of I/O data that is required.

The active state of each output may be configured using the keyword **REMOTREAL**.

Example:

For example, if the node number for *IoNode 24/24* was set to 2, the outputs can be set using:

```
REMOTOUT.2.0 = 0 : REM Clear first 8 bits      (outputs 0 to 7)
REMOTOUT.2.1 = 0 : REM Clear Next 8 bits      (outputs 8 to 15)
REMOTOUT.2.2 = 0 : REM Clear Remaining 8 bits (outputs 16 to 24)
```

Errors:

CAN Node ID out of range (nodeID) - <nodeID> is out of range for this type of node.

Wrong type of node <nodeID> - the node whose ID is <nodeID> is not an output node.

Invalid I/O channel (channel) - <channel> is not valid for the node referenced.

See Also:

REMOTREAL

REMOTERESET

Purpose:

To force an *IoNode 24/24* CAN node into it's reset state.

Keyword is only supported in esMint v2.72 and above.

Format:

REMOTERESET.<nodeID> = <expression>

Abbr.	Read	Write	Command	Scaled	Default	Range
-		c			-	1

<nodeID> is an expression that identifies the node on the bus. This will be the Node ID that was stored in the node during its static configuration.

The REMOTERESET keyword can be used to cause an *IoNode 24/24* to perform a complete software reset. This has the effect of turning off all digital outputs and clearing any errors.

Once reset, the node will generate two event messages with the `_error` bit set.

Errors:

CAN Node ID out of range (nodeID) - <nodeID> is out of range for this type of node.

Wrong type of node <nodeID> - the node whose ID is <nodeID> is not an *IoNode 24/24*.

Example:

REMOTERESET.1 = 1

will force the *IoNode 24/24* at node #1 into reset.

See Also:

CANSTATUS, STATUSNODE, REMOTEERROR

STATUSNODE/SN

Purpose:

To read the Node ID of the node that caused an asynchronous CAN error or event.

Format:

v = STATUSNODE

Abbr.	Read	Write	Command	Scaled	Default	Range
SN	C				-	-

The **CANSTATUS** keyword allows the cause of an asynchronous CAN error to be determined. If **CANSTATUS** has any of the **LIVE**, **DEAD** or **ERROR** bits set, **STATUSNODE** can be read to determine the ID of the node to whom the error or event applies. Reading **STATUSNODE** allows it to be updated along with the **LIVE**, **DEAD** and **ERROR** bits of **CANSTATUS**. Thus, if several nodes have experienced errors or events, successive reads of **CANSTATUS** and **STATUSNODE** will yield the IDs of all of the nodes involved.

Errors:

STATUSNODE does not return any errors.

Example:

Refer to the example **#ONERROR** routine.

See Also:

CANSTATUS

Bibliography

11

Bibliography.

1. *SmartMove Installation Manual*. Baldor document reference number: MN1250.
2. *MintDrive Installation Manual*. Baldor document reference number: MN1274.
3. *NextMove PC Installation Manual*. Baldor document reference number: MN1257.
4. *NextMove PCI Installation Manual*. Baldor document reference number: MN1277.

For v3 Mint the following manuals are applicable:

5. *Mint Programming Manual*. Baldor document reference number: MN1260.
6. *Mint for NextMove - Programming Manual*. Baldor document reference number: MN1261.
7. *Mint for ServoNode - Programming Manual*. Baldor Optimised Control.

For v4 Mint the following manuals are applicable:

8. *Mint v4 Programming Guide*. Baldor document reference number: MN1262.
9. *Mint v4 CAN Programming Guide*. Baldor document reference number: MN1262.
10. *Mint v4 Function Reference Guide*. Baldor document reference number: MN1280.

