3G8B3-DRM21 CompoBus/D VME Board

Operation Manual

Produced June 1997

Notice:

OMRON products are manufactured for use according to proper procedures by a qualified operator and only for the purposes described in this manual.

The following conventions are used to indicate and classify precautions in this manual. Always heed the information provided with them. Failure to heed precautions can result in injury to people or damage to the product.

⚠ **DANGER**	Indicates information that, if not heeded, is likely to result in loss of life or serious injury.
⚠ **WARNING**	Indicates information that, if not heeded, could possibly result in loss of life or serious injury.
⚠ **Caution**	Indicates information that, if not heeded, could result in relatively serious or minor injury, damage to the product, or faulty operation.

OMRON Product References

All OMRON products are capitalized in this manual. The word “Unit” is also capitalized when it refers to an OMRON product, regardless of whether or not it appears in the proper name of the product.

The abbreviation “PC” means Programmable Controller and is not used as an abbreviation for anything else.

Visual Aids

The following headings appear in the left column of the manual to help you locate different types of information.

Note Indicates information of particular interest for efficient and convenient operation of the product.

1, 2, 3... 1. Indicates lists of one sort or another, such as procedures, checklists, etc.

Trademarks and Copyrights

COMBICON is a registered trademark of Phoenix Contact K.K.

DeviceNet is a registered trademark of the Open DeviceNet Vendor Association, Inc.

Other product names and company names in this manual are trademarks or registered trademarks of their respective companies.

The copyright to the CompoBus/D VME Board belongs to S–S Technologies Inc.

TABLE OF CONTENTS

TABLE OF CONTENTS

About this Manual:

This manual describes the installation and operation of the CompoBus/D VME Board and includes the sections described below. The CompoBus/D VME Board connects to the VME bus and acts as a Master Unit which controls the I/O of Slaves in a CompoBus/D FA network.

Please read this manual carefully and be sure you understand the information provided before attempting to install and operate the CompoBus/D VME Board.

Section 1 provides an overview of the CompoBus/D VME Board's functions, specifications, and system configuration.

Section 2 describes the functions of the CompoBus/D VME Board's components and explains how to make switch settings.

Section 3 describes how to install the CompoBus/D VME Board and make the CompoBus/D connections.

Section 4 describes the structure and function of the CompoBus/D VME Board's memory areas.

Section 5 describes the commands that can be used with the CompoBus/D VME Board.

Section 6 provides examples of basic CompoBus/D VME Board operation

Section 7 provides a sample program with a variety of Board operations.

Section 8 provides information on errors that can occur during VME Board operation.

The **Appendix** provides a summary of memory sizes for both short I/O address space and standard address space.

⚠**WARNING** Failure to read and understand the information provided in this manual may result in personal injury or death, damage to the product, or product failure. Please read each section in its entirety and be sure you understand the information provided in the section and related sections before attempting any of the procedures or operations given.

PRECAUTIONS

This section provides general precautions for using the Programmable Controller (PC) and related devices.

The information contained in this section is important for the safe and reliable application of the PC. You must read this section and understand the information contained before attempting to set up or operate a PC system.

1 Intended Audience

This manual is intended for the following personnel, who must also have knowledge of electrical systems (an electrical engineer or the equivalent).

- Personnel in charge of installing FA systems.
- Personnel in charge of designing FA systems.
- Personnel in charge of managing FA systems and facilities.

2 General Precautions

The user must operate the product according to the performance specifications described in the operation manuals.

Before using the product under conditions which are not described in the manual or applying the product to nuclear control systems, railroad systems, aviation systems, vehicles, combustion systems, medical equipment, amusement machines, safety equipment, and other systems, machines, and equipment that may have a serious influence on lives and property if used improperly, consult your OMRON representative.

Make sure that the ratings and performance characteristics of the product are sufficient for the systems, machines, and equipment, and be sure to provide the systems, machines, and equipment with double safety mechanisms.

This manual provides information for programming and operating OMRON PCs. Be sure to read this manual before attempting to use the software and keep this manual close at hand for reference during operation.

⚠ **WARNING** It is extremely important that a PC and all PC Units be used for the specified purpose and under the specified conditions, especially in applications that can directly or indirectly affect human life. You must consult with your OMRON representative before applying a PC System to the abovementioned applications.

3 Safety Precautions

⚠ **WARNING** Never attempt to disassemble any Units while power is being supplied. Doing so may result in serious electrical shock or electrocution.

⚠ **WARNING** Never touch any of the terminals while power is being supplied. Doing so may result in serious electrical shock or electrocution.

4 Operating Environment Precautions

Do not operate the control system in the following places.

- Locations subject to direct sunlight.
- Locations subject to temperatures or humidity outside the range specified in the specifications.
- Locations subject to condensation as the result of severe changes in temperature.
- Locations subject to corrosive or flammable gases.
- Locations subject to dust (especially iron dust) or salts.
- Locations subject to shock or vibration.
- Locations subject to exposure to water, oil, or chemicals.

- Take appropriate and sufficient countermeasures when installing systems in the following locations.
    - Locations subject to static electricity or other forms of noise.
    - Locations subject to strong electric fields or magnetic fields.
    - Locations subject to possible exposure to radioactivity.
    - Locations close to power supplies.

⚠ Caution The operating environment of the PC System can have a large effect on the longevity and reliability of the system. Improper operating environments can lead to malfunction, failure, and other unforeseeable problems with the PC System. Be sure that the operating environment is within the specified conditions at installation and remains within the specified conditions during the life of the system.

5 Application Precautions

Observe the following precautions when using the PC.

⚠ WARNING Failure to abide by the following precautions could lead to serious or possibly fatal injury. Always heed these precautions.

- Always ground the system to 100 Ω or less when installing the system to protect against electrical shock.
- Always turn off the power supply to the PC, Slaves, and communications lines before attempting any of the following:
    - Assembling devices.
    - Setting DIP switches.
    - Connecting or laying cables.

⚠ Caution Failure to abide by the following precautions could lead to faulty operation of the PC or the system or could damage the PC or PC Units. Always heed these precautions.

- Fail-safe measures must be taken by the customer to ensure safety in the event of incorrect, missing, or abnormal signals caused by broken signal lines, momentary power interruptions, or other causes.
- Interlock circuits, limit circuits, and similar safety measures must be provided by the customer as external circuits.
- Always use the power supply voltage specified in this manual.
- Take appropriate measures to ensure that the specified power with the rated voltage and frequency is supplied. Be particularly careful in places where the power supply is unstable.
- Install external breakers and take other safety measures against short-circuiting in external wiring.
- Install the Unit properly as specified in this manual.
- Do not attempt to disassemble, repair, or modify any Units.
- Be sure that the Board mounting screws are tightened to the specified torque.
- Be sure that the connector screws are tightened to the torque specified in this manual.
- Mount the Unit only after checking the terminal block completely.
- Use crimp terminals for wiring. Do not connect bare stranded wires directly to terminals.
- Double-check all the wiring before turning on the power supply.

- Wire correctly.
- Observe the following precautions when laying communications cables.
  - Separate communications cables from the power lines or high-tension lines.
  - Do not bend communications cables.
  - Do not pull on communications cables with excessive force.
  - Do not place heavy objects on top of communications cables.
  - Wire communications cables inside ducts.
- Use appropriate communications cables. Do not use inappropriate cables or inappropriate crimp terminals.
- Be sure that communications cables and other items with locking devices are properly locked into place.
- Check the user program for proper execution before actually running it in the Unit.
- Cover the Board with conductive materials to prevent the LSIs, ICs, etc. from being affected by electrostatic destruction during transportation or storage and store the Board under the specified storage temperature range.
- Do not touch components either on the surface where they are mounted or on the back side of the board. There are sharp edges or points, such as components leads.

SECTION 1
Overview

This section provides an overview of the CompoBus/D VME Board's functions, specifications, and system configurations.

1-1 Functions

1-1-1 Features

The CompoBus/D VME Board connects to the VME bus and has the functions of a Master Unit which controls the I/O of Slaves in a CompoBus/D network.

Note A CompoBus/D network is an FA network. The network can transfer I/O data between remote Slaves through a single cable. Refer to the *CompoBus/D (DeviceNet) Operation Manual* for more details.

The Board has the following features:

Automatic Indication of Slave Data Status

The status of I/O points in CompoBus/D network Slaves is automatically reflected in the Board. Consequently, I/O data can be transferred to and from a Slave just by transferring the data to and from the Slave's data area.

Universal DeviceNet Standards

The CompoBus/D system conforms to the universal DeviceNet standards, so it is compatible with Slaves manufactured by companies all over the world.

Up to 63 Slaves and Up to 12,288 Bytes

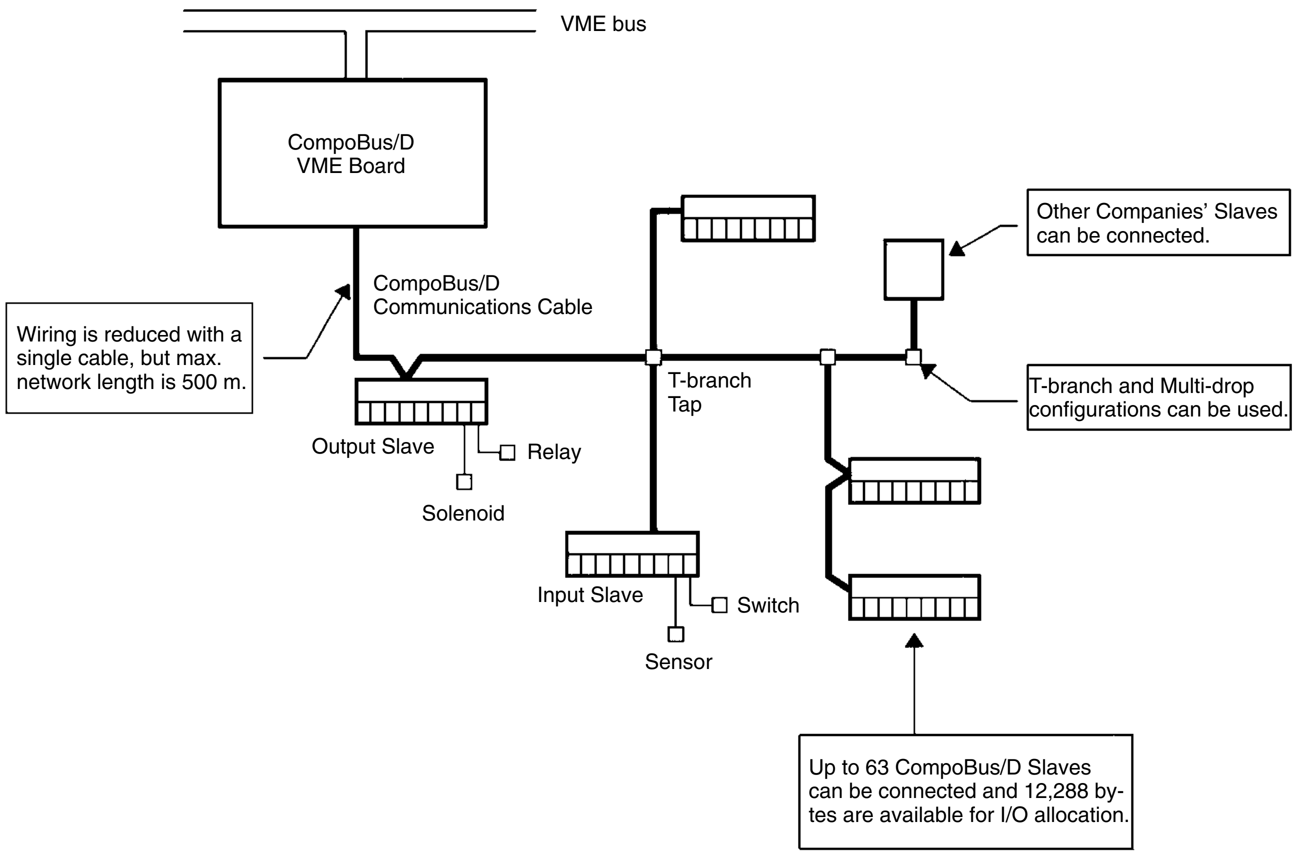
The Board can transfer I/O with up to 63 Slaves and has 12,288 bytes of memory available for I/O allocation.

Maximum Network Length of 500 m

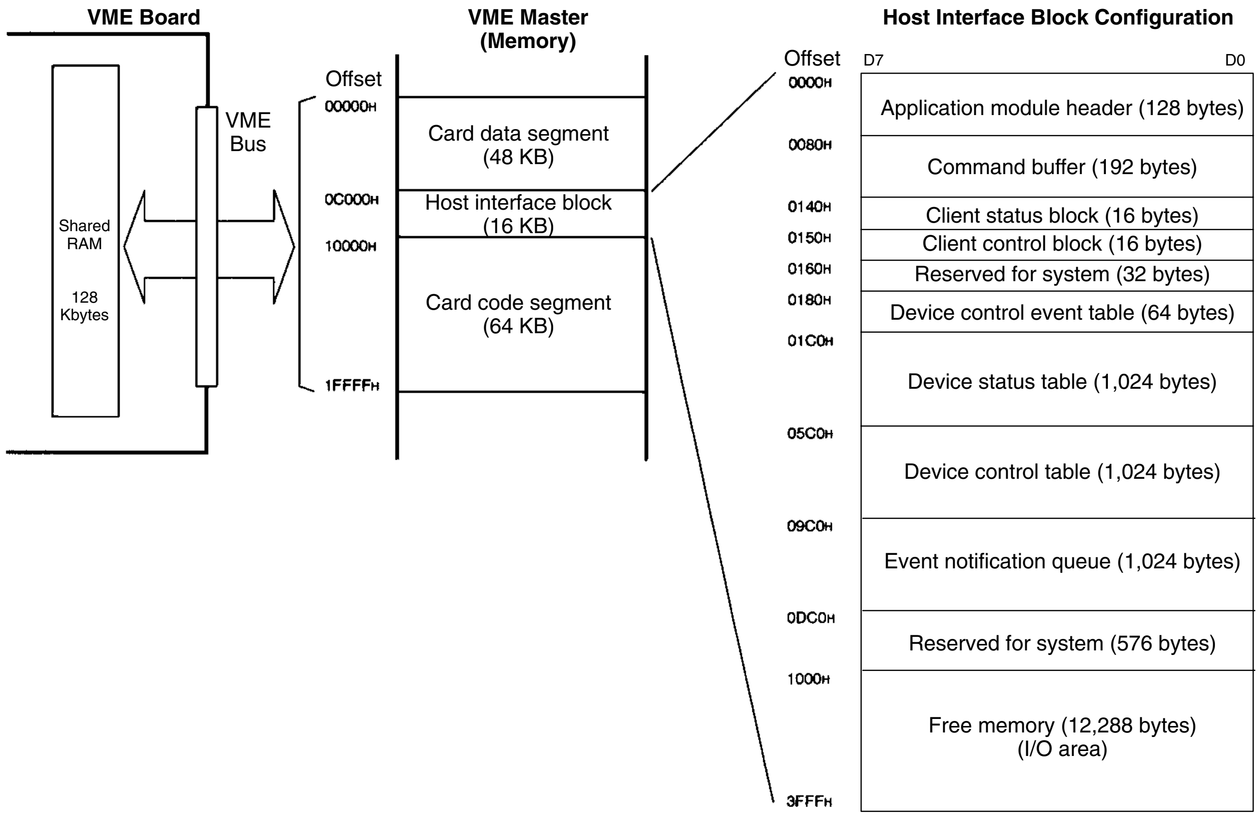
The length of the network can be extended to 500 m max. when a thick cable is used and the baud rate is set to 125 K baud. More distant Slaves can be controlled.

VME Bus

The I/O of CompoBus/D Slaves can be controlled from a VME Master.


VME bus
CompoBus/D VME Board
CompoBus/D Communications Cable
Wiring is reduced with a single cable, but max. network length is 500 m.
Output Slave
Relay
Solenoid
T-branch Tap
Input Slave
Switch
Sensor
Other Companies' Slaves can be connected.
T-branch and Multi-drop configurations can be used.
Up to 63 CompoBus/D Slaves can be connected and 12,288 bytes are available for I/O allocation.

1-1-2 Memory Configuration


VME Board
Shared RAM
128 Kbytes
VME Bus
VME Master (Memory)
Offset
00000H
0C000H
10000H
1FFFFH
Card data segment (48 KB)
Host interface block (16 KB)
Card code segment (64 KB)
Host Interface Block Configuration
Offset
D7
D0
0000H
Application module header (128 bytes)
0080H
Command buffer (192 bytes)
0140H
Client status block (16 bytes)
0150H
Client control block (16 bytes)
0160H
Reserved for system (32 bytes)
0180H
Device control event table (64 bytes)
01C0H
Device status table (1,024 bytes)
05C0H
Device control table (1,024 bytes)
09C0H
Event notification queue (1,024 bytes)
0DC0H
Reserved for system (576 bytes)
1000H
Free memory (12,288 bytes) (I/O area)
3FFFH

VME Master Memory Configuration

There is 128 KB of shared RAM in the VME Board and this memory is allocated to the VME Master's memory.

Card Data Segment
This is a working data area. This area it can't be used from the VME Master, so its operation isn't covered in this manual.

Host Interface Block
This area is used for I/O with the Host, such as controlling the Board and reading its status.

Card Code Segment
This is the Board's program area. This area it can't be used from the VME Master, so its operation isn't covered in this manual.

Host Interface Block Configuration

The various regions of the host interface block are described below.

Application Module Header
This area is used to indicate and control the Board's status and indicate the CompoBus/D network's status.

Command Buffer
This area is used to transfer commands between the Board and the Slaves.

Client Status Block
This area is used to indicate the status of functions related to remote I/O communications.

Client Control Block
This area is used to control functions related to remote I/O communications.

Device Control Event Table
This area is used to notify the VME Board that Slaves are being controlled. One byte is provided for each Slave.

Device Status Table
This area is used to indicate the status of the Slaves. Sixteen bytes are provided for each Slave.

Device Control Table
This area is used to control the Slaves. Sixteen bytes are provided for each Slave.

Event Notification Queue
This is the event notification queue from the Board to the VME Host. When the Host is notified of an event by an interrupt or other means, it can determine the source of the event by checking the event notification queue.

Free Memory (I/O Area)
This memory area doesn't have a fixed function. Use the free memory area to store I/O data for the Slaves.

1-1-3 Control Functions

Initial Processing

Basic Settings

Make basic settings such as the interrupt level and base address with the Board's registers that are allocated in the short I/O space. (See page 51.)

Connection to the Network by Command

When using the Board for remote I/O communications, you must connect to the network with the DN_ONLINE command. (See *Section 5* and page 54.)

Creating the Scan List with Commands

Before starting remote I/O communications, you must allocate Slave I/O in memory. The scan list is a record of this allocation data. Use the ADD_DEVICE command to register Slaves in the scan list and use the DELETE_DEVICE command to remove Slaves from the scan list when necessary.
It is alright to create the scan list before connecting to the network with the DN_ONLINE command. It is possible to add and remove Slaves during remote I/O communications. (See *Section 5* and page 54.)

Starting the Scan by Command

Use the START_SCAN command to start communications with a Slave. When this command is executed, remote I/O communications will start and I/O will be transferred to and from the Slave. (See *Section 5* and page 54.)

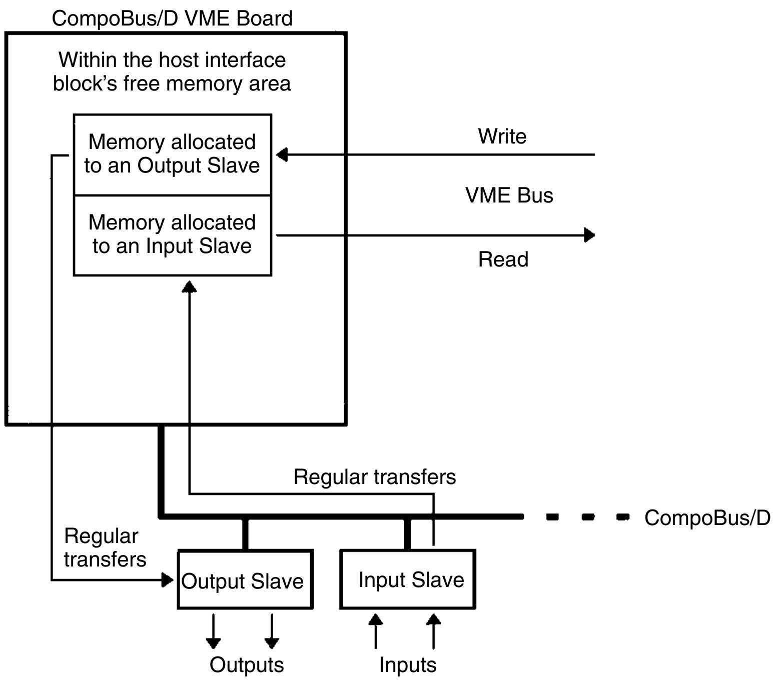
Data I/O

Writing Output Data to a Slave

Just write data to the memory allocated to the Output Slave. The status of the bits in memory will be reflected in the Slave's outputs.

Reading Input Data from a Slave

The status of the Input Slave's inputs will will be reflected in the memory allocated to the Input Slave, so just read the corresponding memory location.


CompoBus/D VME Board
Within the host interface block's free memory area
Memory allocated to an Output Slave
Memory allocated to an Input Slave
Write
VME Bus
Read
Regular transfers
CompoBus/D
Regular transfers
Output Slave
Input Slave
Outputs
Inputs

Note

1. Slaves can be allocated memory anywhere within the host interface block's free memory area. Inputs and outputs can be mixed as well, but the same memory location mustn't be allocated to more than one Slave.
2. To prevent the VME Host and the Board from accessing the same I/O memory simultaneously, I/O memory access from the Board is temporarily prohibited when data is being transferred. Refer to *6-6 Transferring I/O with Slave* for details.

Interrupt Processing

A hardware interrupt can be generated in the Board when the communications status changes, an event occurs, or a response to a command is received.

Interrupt Causes

A hardware interrupt can be generated in the Board in the following cases:

- A response to a command is received.
- There is a change in communications status due to an error or other cause.
- One of the following events has occurred:

  The start of each scan (remote I/O communications)

  A scan started or stopped.

  The status of a Slave changed.

  Input data was written from a Slave to the area allocated to an Input Slave.

It isn't necessary to use interrupts in the Board and the interrupt causes listed above can be enabled or disabled independently. The interrupts related to Slaves can be enabled or disabled independently for each Slave.

Interrupt Processing

Use the following procedure when an interrupt has occurred. The procedure is just outlined here. Refer to *6-9 Processing Interrupts Processing* for details.

1, 2, 3...

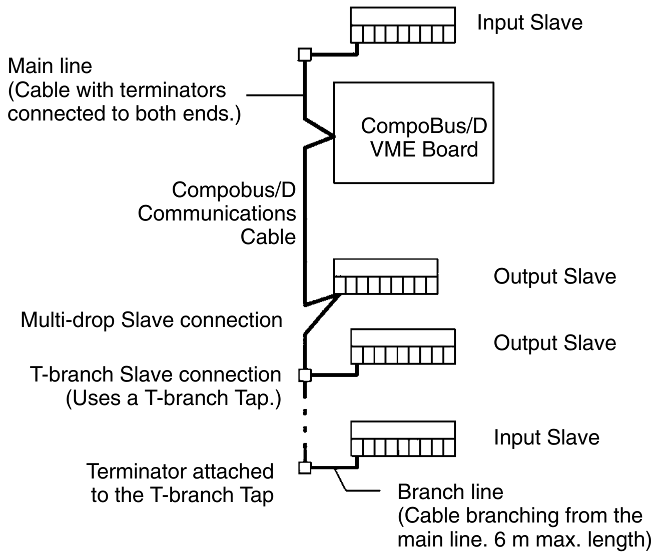
1. Check the application module or other location to determine the cause of the interrupt. The location to be checked depends on the cause of the interrupt.
2. Perform the processing required for the interrupt.
3. Clear the interrupt.

1-1-4 Maximum Number of Slaves and I/O Points

The following table shows the maximum number of I/O points and Slaves that can be connected to the Board.

Item	Maximum number
I/O points	12,288 bytes (6,144 words or 98,304 bits)
Slaves	63 Units (Node addresses 0 to 63)

1-2 Device Configuration


Input Slave
Main line
(Cable with terminators connected to both ends.)
CompoBus/D VME Board
Compobus/D Communications Cable
Output Slave
Multi-drop Slave connection
Output Slave
T-branch Slave connection
(Uses a T-branch Tap.)
Input Slave
Terminator attached to the T-branch Tap
Branch line
(Cable branching from the main line. 6 m max. length)

1-2-1 Baud Rate and Communications Distance

The maximum communications distance depends on the baud rate in the CompoBus/D system, as shown in the following table.

Baud rate	Max. network length		Branch line	Total branch line length
	Thick cable	Thin cable		
500,000 baud	100 m max.	100 m max.	6 m max.	39 m max.
250,000 baud	250 m max.			78 m max.
125,000 baud	500 m max.			156 m max.

1-2-2 Slave Connection Methods

Slave devices can be connected in the following two ways.

Method	Description
T-branch Tap	With this method, a Slave is connected to a branch line that branches from the main line or a branch line with a T-branch Tap.
Multi-drop	With this method, a Slave is connected directly to the main line or a branch line.

The T-branch Tap and multi-drop methods can be mixed freely in the system.

Note Refer to the *CompoBus/D (DeviceNet) Operation Manual* for details on connection methods and grounding.

1-2-3 Devices

CompoBus/D VME Board

Model	Specifications
3G8B3-DRM21	CompoBus/D (DeviceNet) Master

Cables

Cable	Type	Outer diameter	Primary use
5-conductor (2 signal wires, 2 power wires, and 1 shield)	Thick cable	11.2 to 12.1 mm	Main line
	Thin cable	6.9 mm	Main line (see note) or branch lines

Note When thin cable is used for the main line, the length of the main line is limited to 100 m max.

1-3 Preparatory Procedures

1-3-1 Settings

See *Section 2 Components and Switch Settings* for details.

The setting location refers to the pins on the Board's DIP switch.

Setting	Setting location	Function
I/O address	Pins 1 to 6	Sets the area which the Board uses in the short I/O space.
Privileged/non-privileged access	Pin 7	Sets whether only privileged access, or both privileged and non-privileged access are allowed.
Interrupt level	Pins 8 to 10	Sets the interrupt signal level output from the Board and the interrupt response level for interrupt responses.

DIP switch settings must be made before the Board is installed.

1-3-2 Installation and Wiring

See *Section 3 Installation and Connections* for details.

Install the Board in the Rack and connect the CompoBus/D Communications Cable. This manual explains the installation and wiring for the Board only; refer to the *CompoBus/D (DeviceNet) Operation Manual* for details on installing Slaves.

1-3-3 Creating the Program

See *Sections 4* through *7* for details.

Create the program that sets and controls the Board.

1-4 Specifications

1-4-1 General Specifications

Item	Specifications
Current consumption	5 VDC ± 5% 750 mA
Operating temperature	0 to 50°C
Operating humidity	5% to 90%RH (with no condensation)
Storage temperature	– 40°C to 85°C

1-4-2 CompoBus/D Specifications

Item	Specifications
Communications method	DeviceNet communications standards
Baud rate	500,000 baud, 250,000 baud, or 125,000 baud (Set with the DN_ONLINE command.) At 500,000 baud: Network length: 100 m max. Branch line length: 6 m max. Total branch line length: 39 m max. At 250,000 baud: Network length: 250 m max. with thick cable (100 m max. with thin cable) Branch line length: 6 m max. Total branch line length: 78 m max. At 125,000 baud: Network length: 500 m max. with thick cable (100 m max. with thin cable) Branch line length: 6 m max. Total branch line length: 156 m max.
Max. number of devices	64 Units max. including the Master (Up to 63 Slaves can be connected.)
Error control	CRC error correction, node address duplication check, and scan list verification
Cable	5-conductor (2 signal wires, 2 power wires, and 1 shield)
Communications power supply	11 to 24 VDC, 50 mA
Dielectric strength	500 V

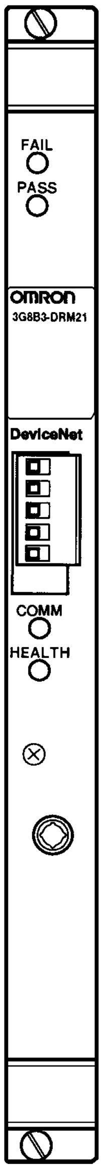
Note Refer to the *CompoBus/D (DeviceNet) Operation Manual* for details on topics such as mixing thick and thin cables and network configurations.

1-4-3 Board Specifications

VME Specifications

Item	Specifications
Model	3G8B3-DRM21
Board size	IEEE1014, 6U height, P1 Compatible
Required addresses	8 bytes in short I/O address space, 128 KB in standard address space
Interrupts	IRQ 1 to 7 set with the DIP switch (ROAK type) When these interrupts aren't used, monitor interrupt status A/B in the application module header.

1-4-4 Front Panel Appearance

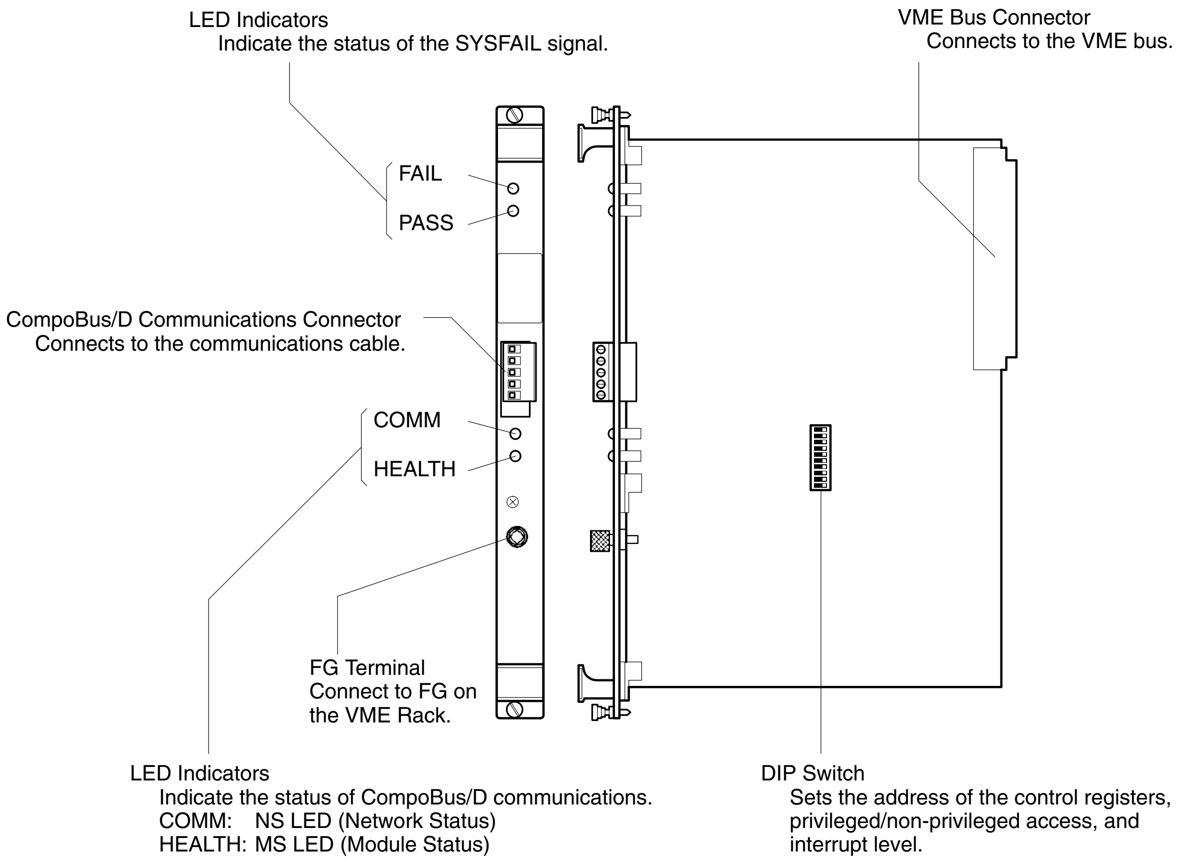


SECTION 2
Components and Switch Settings

This section describes the functions of the CompoBus/D VME Board's components and explains how to make switch settings.

2-1 Components

The following diagram shows the main components of the Board.


LED Indicators
Indicate the status of the SYSFAIL signal.
FAIL
PASS
VME Bus Connector
Connects to the VME bus.
CompoBus/D Communications Connector
Connects to the communications cable.
COMM
HEALTH
FG Terminal
Connect to FG on the VME Rack.
LED Indicators
Indicate the status of CompoBus/D communications.
COMM: NS LED (Network Status)
HEALTH: MS LED (Module Status)
DIP Switch
Sets the address of the control registers, privileged/non-privileged access, and interrupt level.

Note Contact is made between the VME Board and the Rack when the front panel of the VME Board is screwed to the VME Rack. To ensure proper connection to the FG, connect the FG terminal shown above to the FG terminal on the Rack.

2-2 LED Indicators

The following table shows the meaning of the LED indicators.

Indicator	Name	Color	Status	Meaning
FAIL	SYSFAIL	Red	ON	A SYSFAIL occurred.
			OFF	A SYSFAIL hasn't occurred or the SYSFAIL signal is disabled.
PASS	SYSFAIL disabled	Green	ON	The SYSFAIL signal is disabled.
			OFF	The SYSFAIL signal is enabled.
COMM (NS)	Network status	Green	ON	Communications established (Normal network status)
			Flashing	Communications not yet established (The network is normal, but communications haven't been established.)
		Red	ON	Fatal communications error (The Board is indicating one of the following errors that disable communications in the network.) • Node address duplication • Bus Off detected
			Flashing	Non-fatal communications error A communications error occurred in one of the Slaves.
		---	OFF	Network disconnected The DN_ONLINE command hasn't been executed or the DN_OFFLINE command has been executed.
HEALTH (MS)	Module status	Green	ON	Normal status (Normal Board status)
		Red	ON	Fatal malfunction A Board hardware error occurred or the HEALTH LED is disabled during startup.
		---	OFF	Power supply off Power isn't being supplied to the Board or the Board is being reset.

The status of the FAIL, PASS, and HEALTH indicators can be controlled with the Board's setting/status registers. (See *4-2-1 Setting/Status Register* for details.)

When the SYSFAIL signal is disabled with the setting/status registers, the FAIL indicator will be off and the PASS indicator will be on at all times. If the PASS indicator is off, the SYSFAIL signal is enabled and the status of the SYSFAIL signal is reflected by the FAIL indicator.

When the HEALTH indicator is disabled with the setting/status registers, the HEALTH indicator will be on (red) regardless of the Board's status. Of course the HEALTH indicator will be off when power isn't being supplied to the Board.

Note The Board's HEALTH indicator is the same as a Master's MS (module status) indicator and the Board's COMM indicator is the same as a Master's NS (network status) indicator.

2-3 DIP Switch Settings

The following settings are made with the Board's DIP switch.

- The control register address setting
- The privileged/non-privileged access setting
- The interrupt level setting

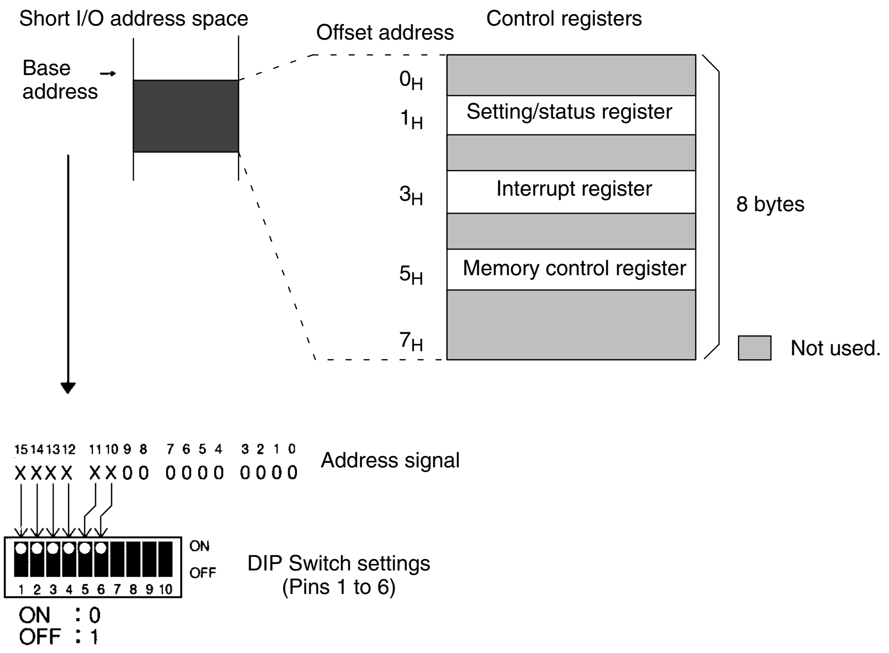
2-3-1 Control Register Address Setting (Pins 1 to 6)

Pins 1 to 6 are used to set the base address of the Board's control registers in the VME short I/O address space. This setting allocates 8 bytes of the short I/O address space to the control registers. (Actually, 3 bytes are used as control registers.)

Note Settings for basic operations and the address allocated for shared RAM can be made with control registers. Refer to *4-2 Control Register Functions* for details.

As shown in the following diagram, the base address (lowest address in the allocated range) is set in binary (address signal). The following table shows the address signals to set.

Address signal	Base address	
	Setting units	Valid setting range
A15 to A10	400_H units	0000_H to $FC00_H$



The factory default setting is 0000_H.

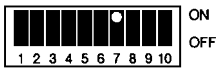
Base address	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6
0000H	ON	ON	ON	ON	ON	ON
0400H	ON	ON	ON	ON	ON	OFF
0800H	ON	ON	ON	ON	OFF	ON
0C00H	ON	ON	ON	ON	OFF	OFF
1000H	ON	ON	ON	OFF	ON	ON
1400H	ON	ON	ON	OFF	ON	OFF
1800H	ON	ON	ON	OFF	OFF	ON
1C00H	ON	ON	ON	OFF	OFF	OFF
2000H	ON	ON	OFF	ON	ON	ON
2400H	ON	ON	OFF	ON	ON	OFF
2800H	ON	ON	OFF	ON	OFF	ON
2C00H	ON	ON	OFF	ON	OFF	OFF
3000H	ON	ON	OFF	OFF	ON	ON
3400H	ON	ON	OFF	OFF	ON	OFF
3800H	ON	ON	OFF	OFF	OFF	ON
3C00H	ON	ON	OFF	OFF	OFF	OFF
4000H	ON	OFF	ON	ON	ON	ON
4400H	ON	OFF	ON	ON	ON	OFF
4800H	ON	OFF	ON	ON	OFF	ON
4C00H	ON	OFF	ON	ON	OFF	OFF
5000H	ON	OFF	ON	OFF	ON	ON
5400H	ON	OFF	ON	OFF	ON	OFF
5800H	ON	OFF	ON	OFF	OFF	ON
5C00H	ON	OFF	ON	OFF	OFF	OFF
6000H	ON	OFF	OFF	ON	ON	ON
6400H	ON	OFF	OFF	ON	ON	OFF
6800H	ON	OFF	OFF	ON	OFF	ON
6C00H	ON	OFF	OFF	ON	OFF	OFF
7000H	ON	OFF	OFF	OFF	ON	ON
7400H	ON	OFF	OFF	OFF	ON	OFF
7800H	ON	OFF	OFF	OFF	OFF	ON
7C00H	ON	OFF	OFF	OFF	OFF	OFF
8000H	OFF	ON	ON	ON	ON	ON
8400H	OFF	ON	ON	ON	ON	OFF
8800H	OFF	ON	ON	ON	OFF	ON
8C00H	OFF	ON	ON	ON	OFF	OFF
9000H	OFF	ON	ON	OFF	ON	ON
9400H	OFF	ON	ON	OFF	ON	OFF
9800H	OFF	ON	ON	OFF	OFF	ON
9C00H	OFF	ON	ON	OFF	OFF	OFF

Base address	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6
A000H	OFF	ON	OFF	ON	ON	ON
A400H	OFF	ON	OFF	ON	ON	OFF
A800H	OFF	ON	OFF	ON	OFF	ON
AC00H	OFF	ON	OFF	ON	OFF	OFF
B000H	OFF	ON	OFF	OFF	ON	ON
B400H	OFF	ON	OFF	OFF	ON	OFF
B800H	OFF	ON	OFF	OFF	OFF	ON
BC00H	OFF	ON	OFF	OFF	OFF	OFF
C000H	OFF	OFF	ON	ON	ON	ON
C400H	OFF	OFF	ON	ON	ON	OFF
C800H	OFF	OFF	ON	ON	OFF	ON
CC00H	OFF	OFF	ON	ON	OFF	OFF
D000H	OFF	OFF	ON	OFF	ON	ON
D400H	OFF	OFF	ON	OFF	ON	OFF
D800H	OFF	OFF	ON	OFF	OFF	ON
DC00H	OFF	OFF	ON	OFF	OFF	OFF
E000H	OFF	OFF	OFF	ON	ON	ON
E400H	OFF	OFF	OFF	ON	ON	OFF
E800H	OFF	OFF	OFF	ON	OFF	ON
EC00H	OFF	OFF	OFF	ON	OFF	OFF
F000H	OFF	OFF	OFF	OFF	ON	ON
F400H	OFF	OFF	OFF	OFF	ON	OFF
F800H	OFF	OFF	OFF	OFF	OFF	ON
FC00H	OFF	OFF	OFF	OFF	OFF	OFF

The factory default setting is 0000_H (pins 1 to 6 all ON).

2-3-2 Privileged/Non-privileged Access Setting

This setting determines whether only privileged access is possible or both privileged and non-privileged access are possible.



DIP switch setting (Pin 7)

Pin 7	Setting
ON (default)	Both privileged and non-privileged access are possible.
OFF	Only privileged access is possible.

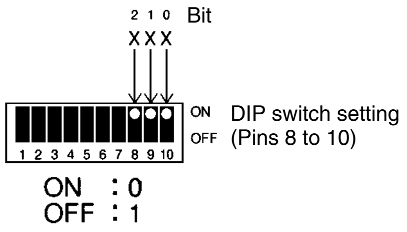
The factory default setting is ON. (Both privileged and non-privileged access are possible.)

2-3-3 Interrupt Level Setting

This setting determines the interrupt signal level output from the Board as well as the interrupt response level for responses from the Board. The Board's interrupt signal level and interrupt response level are normally the same.

The interrupt signal/response level is set in binary and the valid setting range is 0 to 7, shown in the following diagram. (A setting of 0 disables the interrupts.)

The interrupt signal level and interrupt response level are expressed in binary.
(0: No interrupts, 1 to 7: Interrupt signal/response level)



The factory default setting is 0 (no interrupts).

Note
1. When the interrupt level is set to 0 (no interrupts), be sure to monitor interrupt status A/B in the application module header.
2. The Board's interrupts are the ROAK (release on acknowledge) type. The interrupt signal from the Board is cleared when the interrupt response signal is received from the host, but the interrupt flag (IRQ) in the Board's setting/status register isn't cleared automatically. The interrupt flag must be cleared by writing 0 from the Host. See page 61 for details.

Pin settings			**Interrupt level**
Pin 8	**Pin 9**	**Pin 10**	
ON	ON	ON	No interrupts (factory default setting)
OFF	ON	ON	1
ON	OFF	ON	2
OFF	OFF	ON	3
ON	ON	OFF	4
OFF	ON	OFF	5
ON	OFF	OFF	6
OFF	OFF	OFF	7

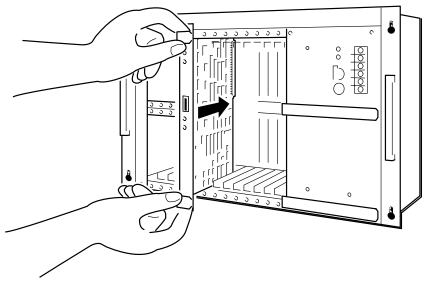
SECTION 3
Installation and Connections

This section explains how to install the CompoBus/D VME Board and make the CompoBus/D connections.

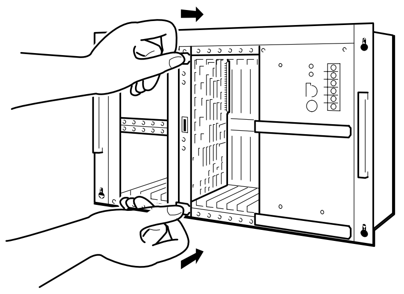
3-1 Board Installation

The following procedure shows how to install the Board. An SBC Rack is used in this example.

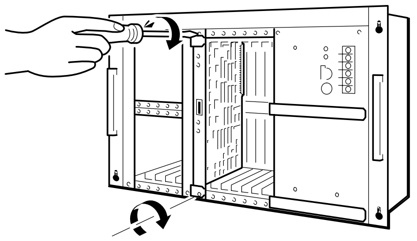
1, 2, 3...
1. Remove all cables from the Board.
2. Turn off the power in the Rack where the Board is being installed and disconnect the power supply cord.
3. Align the VME bus connectors on the Rack and Board. Align the Board in the guide grooves in the Rack's slot and carefully insert the Board.



4. When you feel the Board's connector contact the Racks connector, push the Board until it reaches the back of the Rack. Be sure not to apply too much force when inserting the Board.



5. Lightly pull the latches on the Board's panel and check that the Board can't come out.
6. Tighten the screws on the Board's panel to a torque of about 0.2 N-m to secure the Board.



3-2 CompoBus/D Connections

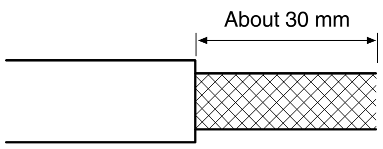
Once the Board is installed, connect the CompoBus/D Communications Cable. This manual describes the CompoBus/D wiring directly related to the Board. Refer to the *CompoBus/D (DeviceNet) Operation Manual* for details on topics such as wiring communications cables or connecting Slaves.

3-2-1 Preparing the Cable

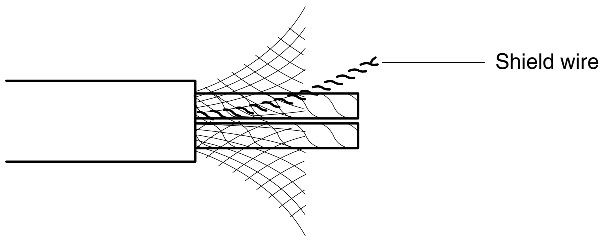
The following procedure shows how to prepare the network communications cable for attachment to the connector.

1, 2, 3...

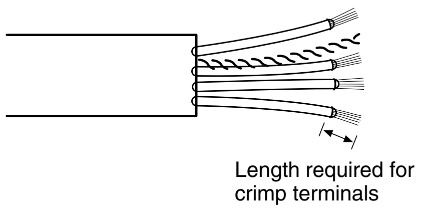
1. Remove about 30 mm of the cable covering, being careful not to damage the shield weaving underneath. Do not remove more than about 30 mm; removing too much of the covering can result in short-circuits.



2. Carefully peel back the weaving. You'll find the signal lines, power lines, and the shield line. The shield line will be loose on the outside of the other lines, but it is harder than the weaving and should be easily identified.

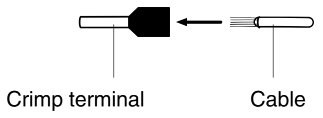


3. Remove the exposed weaving, remove the aluminum Tape from the signal and power lines, and strip the covering from the signal and power lines to the proper length for the crimp terminal connectors. Twist together the wires of each of the signal and power lines.



Note The following crimp terminals are recommended:

- Phoenix Contact K.K., AI-series Crimp Terminals



Insert the cable and crimp the terminal.

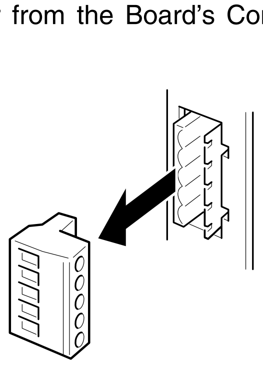
The following crimp tool is also available.

Phoenix Contact K.K., ZA3 Crimp Tool

3-2-2 Connecting the Communications Cable

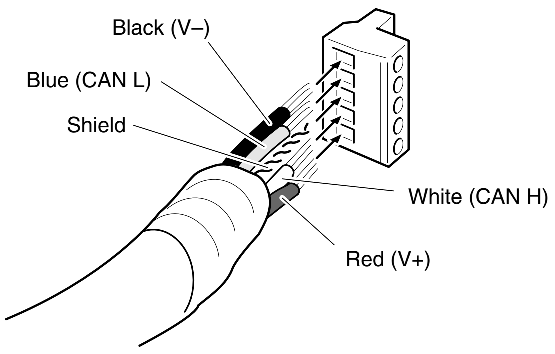
1, 2, 3...

1. Remove the connector from the Board's CompoBus/D communications connector.

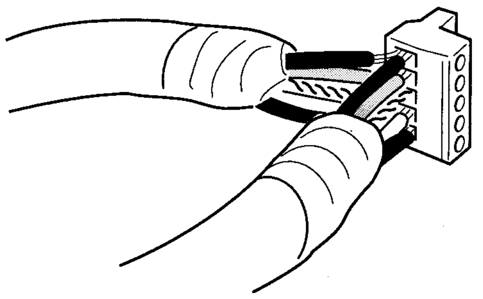


Note The procedure can be performed with the connector in the Board if the cable can be connected that way.

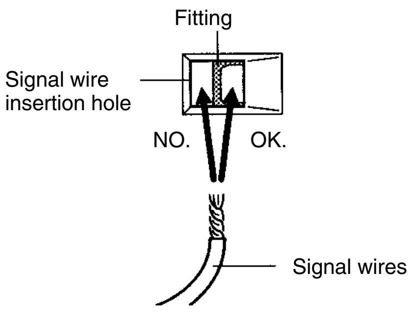
2. Attach the crimp terminals to the lines and then cover any exposed areas of the cable and lines with electricians Tape or heat-shrinking tubes.
   Orient the connector properly, loosen the line lock screws, and then insert the lines in order: Black, blue, shield, white, and then red.
   The following diagram shows the T-branch method.



When connecting with the multi-drop method, insert two of the same colored signal wires in the same hole as shown in the following diagram. The two wires that are inserted into the same hole should be twisted together before insertion.



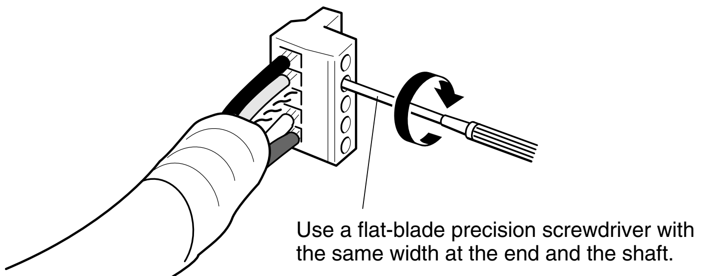
Note Be sure to loosen the screws that secure the signal wires to the connector before inserting the signal, power, and shield wires. If the screw isn't loosened, the wire might go into the wrong side of the clamp and it won't be possible to secure the wire.



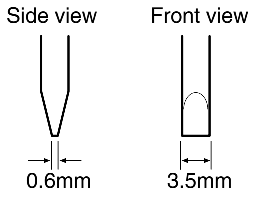
Note When connecting with the multi-drop method, you can use connectors (sold separately) made specifically for the multi-drop method, but this type of connector might block the adjacent slot and make it unusable. In this case, either connect two cables in a single connector or use the T-branch method. Refer to the *CompoBus/D (DeviceNet) Operation Manual* for details on the multi-drop connectors.

3. Tighten the screws that secure the signal wires in the connector.
   Don't use a screwdriver that is just narrow at the point because it won't fit all

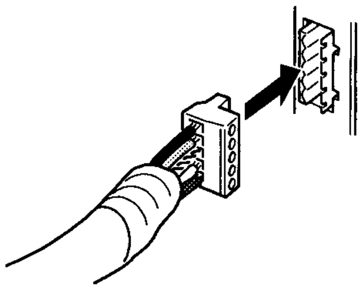
the way into the connector to fully tighten the screw. Use a screwdriver that has the same width along the shank.
Tighten the screws that secure the cable to a torque of about 0.5 N-m.


Use a flat-blade precision screwdriver with the same width at the end and the shaft.

Note The OMRON XW4Z-00C Screwdriver is ideal for tightening these screws.



4. Align the cable's connector with the Board's connector and insert the connector all the way in.



Note
1. Before connecting the communications cable, be sure to turn off the communications power supply as well as the power supplies to the VME bus system and all of the Slaves.
2. Always use crimp terminals when wiring. Just twisting the power wires together might cause malfunctions or even damage the Units if the wires are pulled out or there is a power interruption.
3. Use the appropriate crimping tool and proper technique when attaching the crimp terminals. Contact the supplier of the crimp terminals for details regarding the tools and techniques that should be used. Poorly attached terminals might cause the cable to disconnect.
4. Make sure that the signal, power, and shield wires are in the correct location in the connector.
5. Securely tighten the screws that hold the wires in the connector. (The proper torque is 0.5 N-m.)
6. Take precautions to prevent the signal, power, and shield wires from being pulled out during communications.
7. Do not pull on the communications cable excessively. Excessive force on the cable might pull out the connector or disconnect one of the wires.
8. Do not bend the communications cable too sharply. Bending the cable too sharply might pull out the connector or disconnect one of the wires.

9. Do not place heavy objects on the communications cable. Too much force on the cable might break the wires within it or cause short-circuits.
10. Check the wiring carefully before turning on the power.

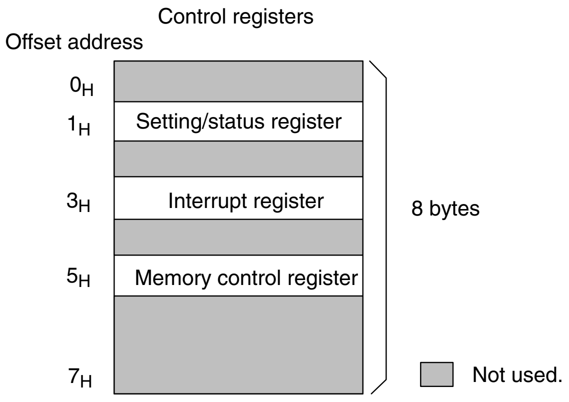
SECTION 4
Memory Area Functions

This section describes the structure and functions of the CompoBus/D VME Board's memory areas.

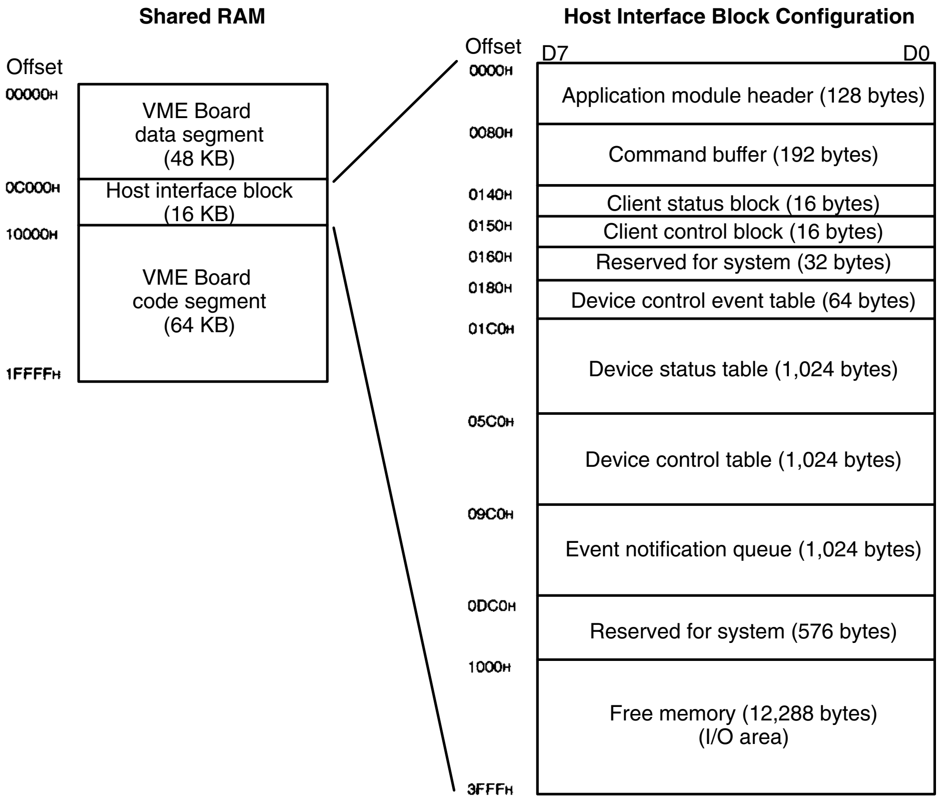
4-1 Memory Area Structure

The following diagrams show how memory is allocated in short I/O address space and standard address space.

Short I/O Address Space



Standard Address Space


Shared RAM
Offset
00000H
VME Board data segment (48 KB)
0C000H
Host interface block (16 KB)
10000H
VME Board code segment (64 KB)
1FFFFH
Host Interface Block Configuration
Offset
D7
D0
0000H
Application module header (128 bytes)
0080H
Command buffer (192 bytes)
0140H
Client status block (16 bytes)
0150H
Client control block (16 bytes)
0160H
Reserved for system (32 bytes)
0180H
Device control event table (64 bytes)
01C0H
Device status table (1,024 bytes)
05C0H
Device control table (1,024 bytes)
09C0H
Event notification queue (1,024 bytes)
0DC0H
Reserved for system (576 bytes)
1000H
Free memory (12,288 bytes) (I/O area)
3FFFH

The following table shows how each area's starting address (base address) is specified.

Area	Base address specification
Control registers	Set with the DIP switch. (See *2-3 DIP Switch Settings*.)
Shared RAM	Set with the memory control register in the control registers. (See *4-2-3 Memory Control Register*.)

About the Byte Order of the Board's Memory

The Board's memory uses Intel byte order, in which the LSB is the lower byte and the MSB is the upper byte. When the VME Host uses Motorola byte order (the LSB is the upper byte and the MSB is the lower byte), switch the order in the software.

The following definition statement switches the upper and lower bytes for 16-bit access:

```
#define SWAP_WORD (WordData) ((WordData<<8) | (WordData>>8))
```

When there are several bytes of data, the memory location with the higher address is the upper byte.

4-2 Control Register Functions

4-2-1 Setting/Status Register (Offset Address 1)

The following table shows the names of the bits in the setting/status register.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RUN	SYS	WDI	HLTH	IRQE	IRQ	RSV	CINT

CINT (Bit 0)

An interrupt occurs in the Board when this bit goes ON.

This bit is turned ON when an interrupt is sent from the VME Host to the Board in order to execute a command. The Board detects the interrupt when this bit goes ON and starts execution of the command set in the command buffer.

This bit must remain ON while the Board performs interrupt processing. An “Incomplete interrupt error” will occur if this bit is turned OFF during interrupt processing.

RSV (Bit 1)

This bit is reserved for the system. Leave it OFF.

IRQ (Bit 2)

When reading data, this bit shows the status of the interrupt from the Board. When writing data, this bit is turned ON by the VME Host to clear the interrupt signal.

This bit is turned ON when an interrupt is generated from the Board to the VME Host. This bit is valid only when the interrupt level is set to a non-zero value with DIP switch pins 8 to 10; it can’t be used when the interrupt level is set to 0 (no interrupts).

Turning this bit ON from the interrupt processing routine in the VME Host clears the hardware interrupt signal.

IRQE (Bit 3)

Interrupts are disabled when this bit is OFF and enabled when this bit is ON.

This bit can be turned OFF to disable interrupts even when the interrupt level is set to a non-zero value (interrupt level 1 to 7). Use this bit to disable interrupts temporarily.

HLTH (Bit 4)

This bit controls the HEALTH indicator. The HEALTH indicator is disabled when this bit is OFF and enabled when this bit is ON.

When disabled, the HEALTH indicator will be red even when it should be green based on the Board’s operating status.

WDI (Bit 5)

This bit controls the Board’s watchdog timer. The watchdog timer is disabled when this bit is OFF and enabled when this bit is ON.

SYS (Bit 6)

This bit controls the SYSFAIL signal. The SYSFAIL signal is disabled when this bit is OFF and enabled when this bit is ON.

When the SYSFAIL signal is disabled, the FAIL indicator will be OFF and the PASS indicator will be ON regardless of the status of the SYSFAIL signal.

RUN (Bit 7)

This bit is OFF while the Board is resetting and ON while the Board is operating.

The Board is in reset status while this bit is OFF. The Board’s firmware begins operation when this bit goes ON.

4-2-2 Interrupt Register (Offset Address 3)

Set the interrupt ID in this register. The interrupt ID is output from the Board in the interrupt response cycle. This register is composed of 8 bits; when reading 16 bits, the upper 8 bits will all be ON.

4-2-3 Memory Control Register (Offset Address 5)

The following table shows the names of the bits in the memory control register.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
A23	A22	A21	A20	A19	A18	A17	MEN

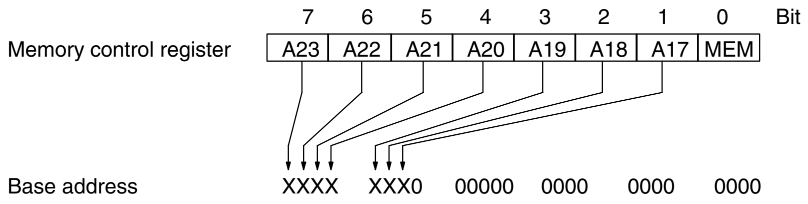
MEN (Bit 0)

Access to shared RAM is disabled when this bit is OFF and enabled when this bit is ON.

This bit is used to notify the Board that the VME Host will access shared RAM; it is ON when the VME Host will access shared RAM. When this bit is OFF, access is disabled. Normally, this bit is turned ON from the VME Host and remains ON from that point.

A17 to A23 (Bits 1 to 7)

These bits set the starting address (base address) of shared RAM. Bits A17 to A23 correspond to address signals A17 to A23.

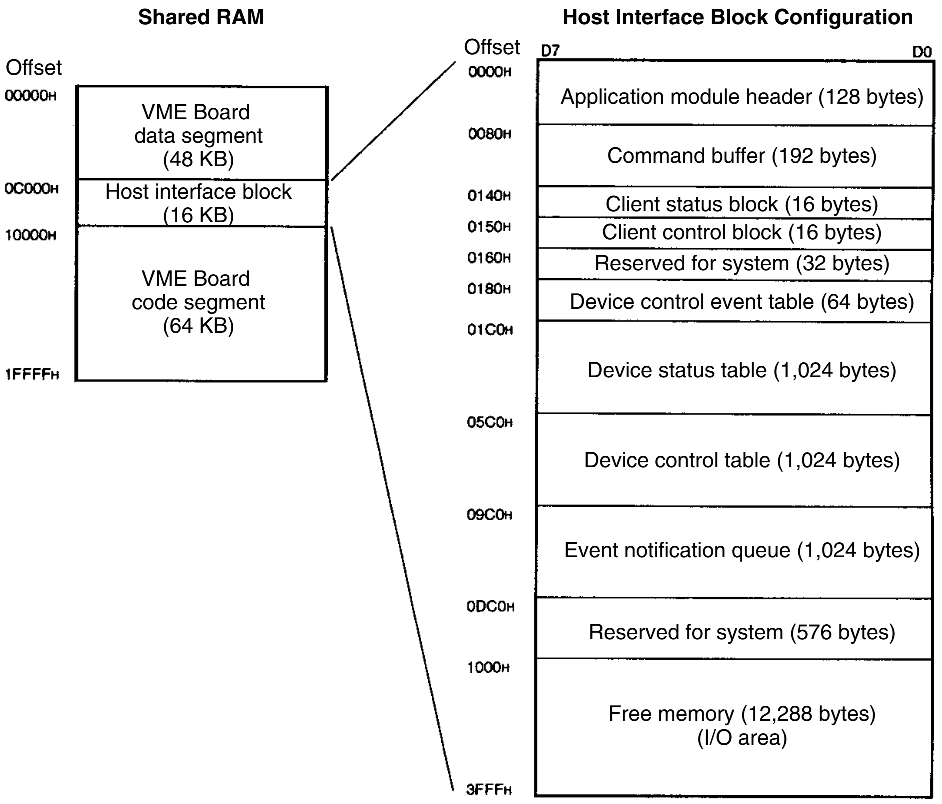

7 6 5 4 3 2 1 0 Bit
Memory control register
A23 A22 A21 A20 A19 A18 A17 MEM
Base address
XXXX XXX0 00000 0000 0000 0000

Address signals	Base address	
	Setting units	Valid setting range
A23 to A17	2000 units	000000 to FE0000

The Board's shared RAM occupies 128 KB in standard address space.

4-3 Shared RAM Functions

The following diagram shows the structure of the shared RAM.


Shared RAM
Offset
00000H
VME Board data segment (48 KB)
0C000H
Host interface block (16 KB)
10000H
VME Board code segment (64 KB)
1FFFFH
Host Interface Block Configuration
Offset
D7
D0
0000H
Application module header (128 bytes)
0080H
Command buffer (192 bytes)
0140H
Client status block (16 bytes)
0150H
Client control block (16 bytes)
0160H
Reserved for system (32 bytes)
0180H
Device control event table (64 bytes)
01C0H
Device status table (1,024 bytes)
05C0H
Device control table (1,024 bytes)
09C0H
Event notification queue (1,024 bytes)
0DC0H
Reserved for system (576 bytes)
1000H
Free memory (12,288 bytes) (I/O area)
3FFFH

The VME Board data segment is the Board's working memory area and the VME Board code segment is the area where the Board's software is stored. Don't access these two areas from the VME Host while the Board is operating (after the setting/status register's RUN bit goes ON).

This section describes the functions of the host interface block which can be used by the VME Host while the Board is operating.

Note Hereafter, the memory addresses are expressed as the offset from the host interface block's starting address. All addresses are in hexadecimal.

4-3-1 Application Module Header (Offset Addresses 0000 to 007F)

The following table shows the structure of the application module header, which is used to indicate and control the status of the Board and indicate the status of the CompoBus/D network.

Area			Name	Function
	Offset address	Bit		
Module type				
	0000 0001	0 to 7	Module type	Contains "DN" (444E in ASCII) when normal, "ER" (4552 in ASCII) when an error has occurred.
Shared RAM size				
	0002 0003	0 to 7	WinSize	Set to 0003 before Board operation (before the setting/status register's RUN bit goes ON).
Reserved				
	0004 to 0029	0 to 7	–	Reserved for the system.
Interrupt control				
	002A	0	CM	Controls the command completion interrupt. (0: Disabled, 1: Enabled)
		1	BS	Controls the network status (0030) change interrupt. (0: Disabled, 1: Enabled)
		2	QE	Controls the event notification queue interrupt. (0: Disabled, 1: Enabled)
		3 to 7	–	Reserved for the system.
	002B	0 to 7	–	Reserved for the system.
Interrupt status A (See page *30*.)				
	002C	0	CM	Indicates the status of the command completion interrupt. (0: Didn't occur, 1: Occurred)
		1	BS	Indicates the status of the network status (0030) change interrupt. (0: Didn't occur, 1: Occurred)
		2	QE	Indicates the status of the event notification queue interrupt. (0: Didn't occur, 1: Occurred)
		3 to 7	–	Reserved for the system.
Interrupt status B (See page *30*.)				
	002D	0	CM	Indicates the status of the command received interrupt. (0: Didn't occur, 1: Occurred)
		1	BS	Indicates the status of the network status (0030) change interrupt. (0: Didn't occur, 1: Occurred)
		2	QE	Indicates the status of the event notification queue interrupt. (0: Didn't occur, 1: Occurred)
		3 to 7	–	Reserved for the system.
Reserved				
	002E 002F	0 to 7	–	Reserved for the system.

Area			Name	Function
	Offset address	Bit		
Network status				
	0030	0	OL	ON (1) when communications interface initialization has been completed. (Online status: DN_ONLINE command completed successfully.)
		1	–	Reserved for the system.
		2	BO	ON (1) when a Bus Off error is detected.
		3	–	Reserved for the system.
		4	TA	ON (1) when there aren't any Slaves.
		5	TO	ON (1) when a communications timeout occurs.
		6	–	Reserved for the system.
		7	–	Reserved for the system.
	0031	0	–	Reserved for the system.
		1	BP	ON (1) when the network power supply is on.
		2	–	Reserved for the system.
		3	–	Reserved for the system.
		4	O1	ON (1) when the baud rate is 125,000 baud.
		5	O2	ON (1) when the baud rate is 250,000 baud.
		6	O5	ON (1) when the baud rate is 500,000 baud.
		7	SA	ON (1) when remote I/O communications are being executed.
CAN transmission counter				
	0032 0033	0 to 7	–	Shows the number of times that messages were transmitted to the network since the Board was initialized.
Reserved				
	0034 0035	0 to 7	–	Reserved for the system.
CAN reception counter				
	0036 0037	0 to 7	–	Shows the number of times that messages were received from the network since the Board was initialized.
CAN error counter				
	0038 0039	0 to 7	–	Shows the number of times that frame errors were detected since the Board was initialized.
Reserved				
	003A to 003F	0 to 7	–	Reserved for the system.
Board information				
	0040 to 007F	0 to 7	–	Contains the error message to be displayed when the module type area (0000) contains "ER".

Accessing Interrupt Status A/B

There are two interrupt status areas, area A and area B, to prevent access conflicts between the VME Host and Board. Use the following procedure when accessing the interrupt status, although it is fine just to write 0 to the desired bit when clearing a single bit.

1, 2, 3...

1. Read "interrupt status A" and store the content in variable A.
2. Clear the desired bit in "interrupt status A" by writing 0 to the bit.
3. Read "interrupt status B" and store the content in variable B.
4. Clear the desired bit in "interrupt status B" by writing 0 to the bit.
5. Use the logical OR of variable A and variable B as the interrupt status.

4-3-2 Command Buffer (Offset Addresses 0080 to 013F)

This buffer is used to exchange commands from the VME Host to the Board and responses from the Board to the VME Host. Refer to *Section 5 Command Usage* for more details.

4-3-3 Client Status Block (Offset Addresses 0140 to 014F)

The following table shows the structure of the client status block. This block is used to transfer information about the status of functions related to remote I/O communications. (Information is transferred from the Board to the VME Host.)

Area			Name	Function
	Offset address	Bit		
Status code				
	0140	0 to 7	–	Indicates the status of remote I/O communications, as follows: 00: Stopped 01: Operating
Reserved				
	0141	0 to 7	–	Reserved for the system.
Remote I/O communications event				
	0142	0	S	This bit is set to 1 at the start of each remote I/O communications scan. The VME Host can clear this bit only when it is set to 1. When bit 2 of the event queue control (015E) is 1, an event is generated and stored in the event queue each time that this bit is changed from 0 to 1. (An event isn't generated when the bit is already set to 1.)
		1 to 7	–	Reserved for the system.
Reserved				
	0143 to 014F	0 to 7	–	Reserved for the system.

4-3-4 Client Control Block (Offset Addresses 0150 to 015F)

The following table shows the structure of the client control block. This block is used to control functions related to remote I/O communications from the VME Host.

Area			Name	Function
	Offset address	Bit		
Reserved				
	0150 to 015D	0 to 7	–	Reserved for the system.
Event queue control				
	015E	0	C	This bit determines whether or not an event is generated and stored in the queue when there is a change in the client status block's status code (0140). 0: Don't store in queue. 1: Store in queue.
		1	–	Reserved for the system.
		2	S	This bit determines whether or not an event is stored in the queue when an event has occurred in the client status block's remote I/O communications event (0142). 0: Don't store in queue. 1: Store in queue.
		3 to 7	–	Reserved for the system.
Reserved				
	015F	0 to 7	–	Reserved for the system.

4-3-5 Device Control Event Table (Offset Addresses 0180 to 01BF)

This table is used to transfer information about Slave control from the VME Host to the Board. The 64 bytes from 0180 to 01BF correspond to node addresses 0 to 63.

When the Board is notified of the status of the I/O event flag in a device's device control table, 01 is written to the corresponding byte in this table. (Don't write anything when the value is non-zero already.)

The Board performs the processing indicated by the device control table's I/O event flag, clears the corresponding byte in this device control event table to 00, and then clears the device control table's I/O event flag.

Offset address	Corresponding Slave
0180	Node address 0
0181	Node address 1
:	:
01BE	Node address 62
01BF	Node address 63

4-3-6 Device Status Table (Offset Addresses 01C0 to 05BF)

This table indicates each Slave's status. There are 16 bytes allocated to each Slave, as shown in the following table.

Offset address	Corresponding Slave
01C0 to 01CF	Node address 0
01D0 to 01DF	Node address 1
:	:
05A0 to 05AF	Node address 62
05B0 to 05BF	Node address 63

The following table shows the structure of each Slave's device status table.

Area			Name	Function
	Offset address	Bit		
Status code				
	+0	0 to 7	–	Contains the status code that shows the Slave's status. (See the "Status Codes" table on page *33*.)
Status flags				
	+1	0	–	Reserved for the system.
		1	I1	Input interlock. This bit is set to 1 when the Board starts writing input data. To ensure simultaneity between the Slaves' data, do not read the corresponding input data from the VME Host while this bit is set to 1. (Refer to *6-6 Transferring I/O with Slaves* for details.)
		2	–	Reserved for the system.
		3	O1	Output interlock. This bit is set to 1 when the Board starts reading output data. To ensure simultaneity between the Slaves' data, do not write the corresponding output data from the VME Host while this bit is set to 1. (Refer to *6-6 Transferring I/O with Slaves* for details.)
		4 to 7	–	Reserved for the system.
I/O event flag				
	+2	0	U	This bit is set to 1 each time that input data is received from a Slave. The VME Host can clear this bit only when it is set to 1. When the device control table's "event queue control bit" (bit 2 of byte +14) is set to 1, an event is generated and stored in the event queue each time that this bit is set from 0 to 1. (An event isn't generated when the bit is already set to 1.)
		1 to 7	–	Reserved for the system.
Reserved				
	+3 to +15	0 to 7	–	Reserved for the system.

The settings of the device control table's "event queue control bits" determine whether an event is stored in the event queue when there is a change in the status code or an event occurs in the I/O event flag.

Status Codes

The following table shows the Slave's status codes.

Status code	Meanings
00	Device not in scan list
01	Device idle (not being scanned)
02	Device being scanned
03	Device timed-out
04	UCMM connection error
05	Master/Slave connection set is busy
06	Error allocating Master/Slave connection set
07	Invalid vendor id
08	Error reading vendor id
09	Invalid device type
0A	Error reading device type
0B	Invalid product code
0C	Error reading product code
0D	Invalid input size
0E	Error reading input size
0F	Invalid output size
10	Error reading output size
11	Reserved for the system.
12	Reserved for the system.
13	Reserved for the system.
14	Reserved for the system.
15	Remote I/O connection packet rate error
16	Reserved for the system.
17	Master/Slave connection set sync fault
18 to FF	Reserved for the system.

4-3-7 Device Control Table (Offset Addresses 05C0 to 09BF)

This table is used to control the Slaves. There are 16 bytes allocated to each Slave, as shown in the following table.

Offset address	Corresponding Slave
05C0 to 05CF	Node address 0
05D0 to 05DF	Node address 1
:	:
09A0 to 09AF	Node address 62
09B0 to 09BF	Node address 63

The following table shows the structure of each Slave's device control table.

Area			Name	Function
	Offset address	Bit		
Control bits				
	+0	0	–	Reserved for the system.
		1	I1	Input interlock. This bit is set to 1 to notify the Board when the VME Host starts reading input data. The Board waits to write the corresponding data while this bit is set to 1. This bit can be used to ensure input data simultaneity. (Refer to *6-6 Transferring I/O with Slaves* for details.)
		2	–	Reserved for the system.
		3	O1	Output interlock. This bit is set to 1 to notify the Board when the VME Host starts writing output data. The Board waits to read the corresponding data while this bit is set to 1. This bit can be used to ensure output data simultaneity. (Refer to *6-6 Transferring I/O with Slaves* for details.)
		4 to 7	–	Reserved for the system.
I/O event flag				
	+1	0	C	After this bit has been set to 1 from the VME Host, the Board is notified that there was a change in output data by the fact that 01 is written in the corresponding byte in the device control event table. Normally, the Slave's output data is refreshed once each communications cycle even if this bit isn't used. Use this bit when you want the change in output data to be reflected in the Slave immediately.
		1 to 7	–	Reserved for the system.
Reserved				
	+2 to +13	0 to 7	–	Reserved for the system.
Event queue control bits				
	+14	0	C	This bit determines whether or not an event is generated and stored in the queue when there is a change in the device status table's status code (byte +0). 0: Don't store in queue. 1: Store in queue.
		1	–	Reserved for the system.
		2	I1	This bit determines whether or not an event is stored in the queue when an event has occurred in the device status table's I/O event flag (byte +2). 0: Don't store in queue. 1: Store in queue.
		3 to 7	–	Reserved for the system.
Reserved				
	+15	0 to 7	–	Reserved for the system.

4-3-8 Event Notification Queue (Offset Addresses 09C0 to 0DBF)

Relevant information is stored as an event in the following cases. (The user can specify whether or not these events are stored in the event queue.)

- There is a change in the client status block's status code (0140).
- An event occurs in the client status block's remote I/O communications event flag (0142).
- There is a change in the device status block's status code (byte +0).
- An event occurs in the device status block's I/O event flag (byte +2).

If an event is stored in the queue, bit 2 of the application module header's interrupt status areas A (002C) and B (002D) will be set to 1. An interrupt will be generated if interrupts are enabled. Refer to *6-9 Processing Interrupts* for more details on the queue and interrupts.

The following table shows the structure of the event notification queue.

Area			Name	Function
	Offset address	Bit		
Queue status A				
	09C0	0	O	ON (1) when a queue overrun occurred.
		1 to 7	–	Reserved for the system.
Queue status B				
	09C1	0	O	ON (1) when a queue overrun occurred.
		1 to 7	–	Reserved for the system.
Queue IN				
	09C2 09C3	0 to 7	–	Contains the address (offset address from the beginning of the host interface area) where the next event will be stored. Only the Board can write to these bytes. The Board sets these bytes to 09C6 immediately after firmware startup.
Queue OUT				
	09C4 09C5	0 to 7	–	Contains the address (offset address from the beginning of the host interface area) where the next event will be read. Only the VME Host can write to these bytes. The Board sets these bytes to 09C6 immediately after firmware startup.
Event queue				
	09C6 to 0DBF	0 to 7	–	The event information is stored in this area. Each event is allocated 2 bytes and a maximum of 509 events can be stored. (See *Event Information* below for details in the structure of the 2 bytes.)

Accessing Queue Status A/B

Use the following procedure to check the status of bits in queue status A (09C0) and queue status B (09C1).

1, 2, 3...

1. Read “queue status A” and store the content in variable A.
2. Clear the desired bit in “queue status A” by writing 0 to the bit.
3. Read “queue status B” and store the content in variable B.
4. Clear the desired bit in “queue status B” by writing 0 to the bit.
5. Use the logical OR of variable A and variable B as the queue status.

Event Information

Area			Name	Function
	Offset address	Bit		
Event source code				
	+0	0 to 7	–	Contains a code that shows where the event occurred. (See table below.)
Event ID				
	+1	0 to 7	–	Contains an ID that shows what kind of event occurred. (See table below.)

Event Source Code

The event source code shows where the event occurred.

Code	
00	Device with node address 0 (device status table)
01	Device with node address 1 (device status table)
:	:
3E	Device with node address 62 (device status table)
3F	Device with node address 63 (device status table)
40	Client (client status block)
41 to FF	Reserved for the system.

Event ID

The event ID shows what kind of event occurred. The following table shows the function that generated the event and the offset address of the beginning of the device status table or client status block.

Code	Event information
00	The status code (+0 bytes) of a Slave in the device status table (01C0) changed.
02	An event occurred with the I/O event flag (+2 bytes) of a Slave in the device status table (01C0).
00	The status code of the client status block (0140) changed.
02	An event occurred with the remote I/O communications event flag of the client status block (0142).

Together the event source code and event ID show where to check for more detailed information on the location and nature of the event.

Example 1

The status of the Slave with node address 15 changed.

Event source code:	0F (15 decimal)
Event ID:	00
Address to check:	01C0 + 16 × 0F + 00 = 02B0

Example 2

An event occurred with the remote I/O communications event flag of the client status block.

Event source code:	40
Event ID:	02
Address to check:	0142

4-3-9 Free Memory (Offset Addresses 1000 to 3FFF)

The area from 1000 to 3FFF can be used freely for Board operations. Normally, the free memory is used as an I/O area for I/O transfers with the Slaves.

Refer to *5-3-3 ADD_DEVICE (Command Code 03)* for details on using the free memory area

SECTION 5
Command Usage

This section describes the commands that can be used with the CompoBus/D VME Board.

5-1 Command Summary

CompoBus/D VME Board processing can be controlled by exchanging commands and responses with the VME Host.

The following commands can be used with the Board. Commands are identified by their command codes.

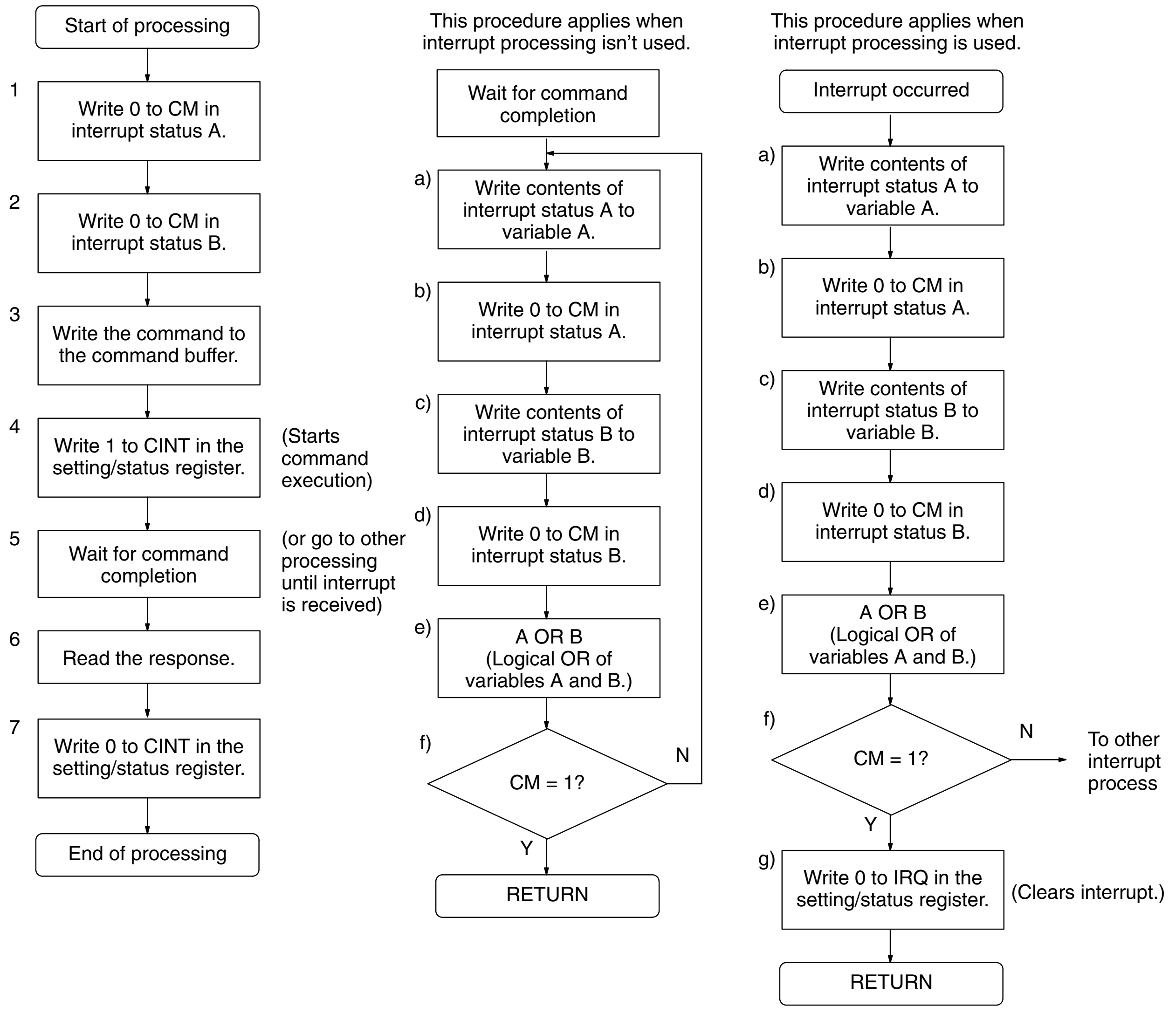
Command	Code	Function	Timeout detection time	Page
DN_ONLINE	01	Connects the Board to the network.	3 s	41
DN_OFFLINE	02	Removes the Board from the network.	1 s	42
ADD_DEVICE	03	Registers a Slave in the scan list, which specifies how I/O is transferred in remote I/O communications. Slaves cannot participate in remote I/O communications unless they are registered in the scan list.	1 s	42
GET_DEVICE	04	Reads the Slave information registered in the scan list.	1 s	45
DELETE_DEVICE	05	Removes a Slave from the scan list.	1 s	46
START_SCAN	06	Starts remote I/O communications.	1 s	46
STOP_SCAN	07	Stops remote I/O communications.	1 s	47

A timeout error will occur if a response is not received within the “timeout detection time” shown in the table.

5-2 Command Execution

Commands and responses are exchanged between the VME Host and the Board through the host interface area's command buffer. The data written to the command buffer or stored in the buffer as a response depends upon the command. Refer to the description of each command for details.

This section describes the procedures that are common to all of the commands. The following diagram shows the procedures for exchanging commands and responses through the command buffer.


Start of processing
1 Write 0 to CM in interrupt status A.
2 Write 0 to CM in interrupt status B.
3 Write the command to the command buffer.
4 Write 1 to CINT in the setting/status register.
(Starts command execution)
5 Wait for command completion
(or go to other processing until interrupt is received)
6 Read the response.
7 Write 0 to CINT in the setting/status register.
End of processing
This procedure applies when interrupt processing isn't used.
Wait for command completion
a) Write contents of interrupt status A to variable A.
b) Write 0 to CM in interrupt status A.
c) Write contents of interrupt status B to variable B.
d) Write 0 to CM in interrupt status B.
e) A OR B (Logical OR of variables A and B.)
f) CM = 1?
N
Y
RETURN
This procedure applies when interrupt processing is used.
Interrupt occurred
a) Write contents of interrupt status A to variable A.
b) Write 0 to CM in interrupt status A.
c) Write contents of interrupt status B to variable B.
d) Write 0 to CM in interrupt status B.
e) A OR B (Logical OR of variables A and B.)
f) CM = 1?
N
To other interrupt process
Y
g) Write 0 to IRQ in the setting/status register.
(Clears interrupt.)
RETURN

1, 2, 3...
1. Write 0 to CM in interrupt status A and clear it.
2. Write 0 to CM in interrupt status B and clear it.

   The interrupt status indicates whether command execution is completed.

Name	Offset address	Bit	Function
CM	002C (interrupt status A) 002D (interrupt status B) (in the host interface area)	Bit 0	Contains 0 when the command completion interrupt hasn't occurred, 1 when it has.

3. Write the command code and parameters in the command buffer (0080 to 013F). Refer to the description of the command in *5-3 Command Specifics* for details.

4. Write 1 to CINT in the setting/status register. When CINT goes from 0 to 1, an interrupt occurs in the Board and the command is executed.

Name	Offset address	Bit	Function
CINT	1 (in short I/O space)	Bit 0	On the rising edge of CINT, an interrupt occurs in the Board and command execution begins.

   CINT must remain set to 1 until execution of the command is completed.

5. Wait until CM in interrupt status A and interrupt status B are set to 1. Hardware interrupts can be used when the command completion interrupt is enabled with the interrupt control area's CM bit (bit 0 of 002A). (See *6-9 Processing Interrupts* for more details.)

   Use the following procedure when checking the bits in interrupt status A/B.

   a) Read interrupt status A and store the contents in variable A.
   b) Write 00 to interrupt status A and clear it.
   c) Read interrupt status A and store the contents in variable B.
   d) Write 00 to interrupt status B and clear it.
   e) Take the logical OR of variables A and B and use the result as the interrupt status.
   f) Check whether CM (bit 0) is 1 in the result from step e).
      When interrupts aren't being used, repeat the procedure from step a) if CM is still 0. Proceed to step 6. if CM is 1.
      When interrupts are being used, another type of interrupt has occurred if CM is still 0 so check the value of the result obtained in step e) and perform the appropriate interrupt processing. Proceed to step g) if CM is 1.
   g) When interrupts are being used, write 1 to IRQ in the setting/status register to clear the interrupt, and proceed to step 6.

Name	Offset address	Bit	Function
IRQ	1 (in short I/O space)	Bit 2	Reading: Shows interrupt status. Writing 1: Clears interrupt signal.

6. When the CM bits of interrupt status A and B are 1, execution of the command has been completed and the command buffer contains the response, so read the contents of the command buffer. Refer to *5-3 Command Specifics* or *5-4 Command Errors* for details on the response.

7. Once the response has been read, write 0 to CINT in the setting/status register. (This bit was set to 1 in step 4.)

5-3 Command Specifics

This section describes the commands and the information required to use them, as well as how to create the scan list that governs remote I/O communications.

5-3-1 DN_ONLINE (Command Code 01)

The DN_ONLINE command connects the Board to the CompoBus/D network. Always use this command to connect to the network before starting remote I/O communications with the START_SCAN command.

It is fine to leave the Board connected to the network after stopping remote I/O communications. After the Board is removed from the network, the DN_ONLINE command can be used to reconnect the Board to the network immediately as long as a hardware reset hasn't been performed on the Board.

Note A node address duplication check is performed when the DN_ONLINE command is executed. This check may take 2 s or more to complete.

Command Buffer Settings

The following table shows the command buffer settings required when using the DN_ONLINE command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Set to 0001.
0082	Node address	Word	Set the Board's node address from 0000 to 003F (0 to 63). Don't set a node address that is being used by another device (Master or Slave).
0084	Baud rate	Word	Set the baud rate necessary for the Board to connect to the CompoBus/D network. 0000: 125,000 baud 0001: 250,000 baud 0002: 500,000 baud Always set the same baud rate in the other devices (Masters and Slaves).
0086	Communications cycle time	Word	Sets a fixed cycle time for remote I/O communications. This setting can be used to avoid variations in the communications cycle caused by changing network conditions and prevent timeout errors caused by slow Slaves. The system will take as much time as necessary if the time required for remote I/O communications exceeds the cycle time setting. The default setting is for a variable cycle time which is the minimum time required for remote I/O communications. The cycle time can be set in 1-ms units from 5 to 1,000 ms, but the actual cycle time is processed in 5-ms units. For example, a setting of 21 results in an actual communications cycle time of 25 ms.
0088 to 013F	Reserved	---	Set to 0000.

Response

The following table shows the Board's response to the DN_ONLINE command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Normal completion: 0001 Error: 8001 Refer to *5-4 Command Errors* for details on errors.
0082 to 013F	Reserved	---	---

5-3-2 DN_OFFLINE (Command Code 02)

The DN_OFFLINE command removes the Board from the CompoBus/D network.

This command cannot be used while remote I/O communications are operating. Always stop remote I/O communications with the STOP_SCAN command before using this command to remove the Board from the network.

Command Buffer Settings

The following table shows the command buffer settings required when using the DN_OFFLINE command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Set to 0002.
0082 to 013F	Reserved	---	Set to 0000.

Response

The following table shows the Board's response to the DN_OFFLINE command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Normal completion: 0002 Error: 8002 Refer to *5-4 Command Errors* for details on errors.
0082 to 013F	Reserved	---	---

5-3-3 ADD_DEVICE (Command Code 03)

The ADD_DEVICE command registers a Slave in the Board's scan list. Slaves participating in remote I/O communications with the Board must be registered in the scan list.

This command can be executed at any time after firmware startup. If ADD_DEVICE is executed while remote I/O communications are operating, remote I/O communications will start with the new Slave as soon as it is registered.

Scan List Registration

The following Slave information and I/O information is registered in the scan list for each Slave.

- Node address (MacId)
- Vendor ID
- Device type
- Product code
- Connection type
- Output size (bytes)
- Output address offset (the starting address where output data to the Slave is written – expressed as an offset from the beginning of the host interface area)
- Input size (bytes)

- Input address offset (the starting address where input data from the Slave is stored – expressed as an offset from the beginning of the host interface area)

Set 0000 for the vendor ID, device type, and product code when these values are unknown. The Board will read these values from the Slave and register them automatically. (The registered values can be read from the scan list with the GET_DEVICE command.)

Set one of the following values for the "connection type."

Connection type	Supported connection
0002	Poll connection
0004	Strobe connection

When 0004 (strobe connection) is specified for the connection type, set the output size to 0001. An ERR_STROBE_BUFFER error (error code 0011) will occur if any other value is set for the output size.

Set the "output size" and "input size" supported by the Slave in hexadecimal.

Set the lower 4 digits of the "output address offset" and "input address offset" in hexadecimal. Specify these addresses as an offset from the beginning of the host interface area.
If free memory (1000 to 3FFF) is used, any addresses that aren't being used by another Slave can be specified. The addresses can be allocated in any order as long as the allocated areas don't exceed the boundaries of the free memory area.

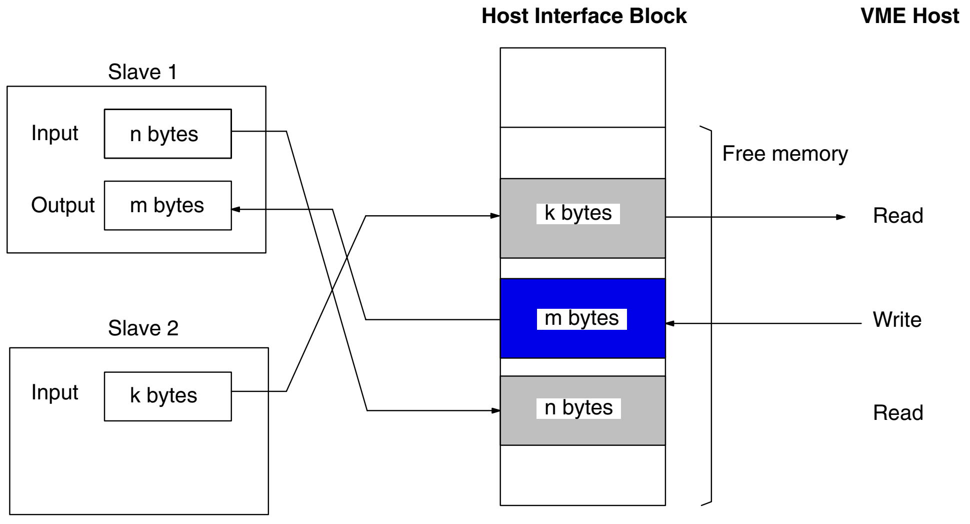
When an OMRON Slave is being used, set the "connection type", "output size", and "input size" shown in the following table.

Slave	Model	Connection type	Output size	Input size
I/O Link Unit	CQM1-DRT21	0002	2	2
I/O Terminal	DRT1-ID08	0004	1	1
	DRT1-OD08	0002	1	0
	DRT1-ID16	0004	1	2
	DRT1-OD16	0002	2	0
Remote Adapter	DRT1-ID16X	0004	1	2
	DRT1-OD16X	0002	2	0
Sensor Terminal	DRT1-HD16S	0004	1	2
	DRT1-ND16S	0002	1	1
Analog Input Terminal	DRT1-AD04 (See note 1.)	0004	1	4
			1	8
	DRT1-AD04H (See note 2.)	0004	1	8
Analog Output Terminal	DRT1-DA02	0002	4	0
Temperature Input Terminal	DRT1-TS04T (See note 2.)	0004	1	8
	DRT1-TS04P (See note 2.)	0004	1	8

Note 1. The DRT1-AD04's input size can be changed with the DIP switch.

2. These Units will be available soon.

The following example shows the elements of the number of input/output size and input/output address offset settings.



Slave 1 Settings (hexadecimal):

Input size:	n
Input address offset:	aaaa
Output size:	m
Output address offset:	bbbb

Slave 2 Settings (hexadecimal):

Input size:	k
Input address offset:	cccc

Command Buffer Settings

The following table shows the command buffer settings required when using the ADD_DEVICE command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Set to 0003.
0082	Node address	Word	See preceding explanation.
0084	Vendor ID	Word	
0086	Device type	Word	
0088	Product code	Word	
008A to 008F	Reserved	---	Set to 0000.
0090	Connection type	Word	Set to 0002 for poll connection. Set to 0004 for strobe connection.
0092 to 0097	Reserved	---	Set to 0000.
0098	Output size (bytes)	Word	See preceding explanation.
009A	Output address offset	Word	
009C to 009F	Reserved	---	Set to 0000.
00A0	Input size (bytes)	Word	See preceding explanation.
00A2	Input address offset	Word	
00A4 to 013F	Reserved	---	Set to 0000.

Response

The following table shows the Board's response to the ADD_DEVICE command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Normal completion: 0003 Error: 8003 Refer to *5-4 Command Errors* for details on errors.
0082 to 013F	Reserved	---	---

5-3-4 GET_DEVICE (Command Code 04)

The GET_DEVICE command reads the settings of a Slave registered in the scan list. This command can be executed at any time after firmware startup.

Command Buffer Settings

The following table shows the command buffer settings required when using the GET_DEVICE command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Set to 0004.
0082	Node address	Word	Set the node address of the desired Slave from 0000 to 003F (0 to 63 decimal).
0084 to 013F	Reserved	---	Set to 0000.

Response

The following table shows the Board's response to the GET_DEVICE command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Normal completion: 0004 Error: 8004 Refer to *5-4 Command Errors* for details on errors.
0082	Node address	Word	Contains the Slave's node address.
0084	Vendor ID	Word	Contains the Slave's vendor ID.
0086	Device type	Word	Contains the Slave's device type.
0088	Product code	Word	Contains the Slave's product code.
008A to 008F	Reserved	---	---
0090	Connection type	Word	0002: Poll connection. 0004: Strobe connection.
0092 to 0097	Reserved	---	---
0098	Output size (bytes)	Word	Contains the Slave's number of output bytes.
009A	Output address offset	Word	Contains the Slave's output address offset.
009C to 009D	Reserved	---	---
009E	Input size (bytes)	Word	Contains the Slave's number of input bytes.
00A0	Input address offset	Word	Contains the Slave's input address offset.
00A2 to 013F	Reserved	---	---

See *Scan List Registration* on page 42 for details on the parameters.

5-3-5 DELETE_DEVICE (Command Code 05)

The DELETE_DEVICE command deletes a Slave's information from the scan list.

This command can be executed at any time after firmware startup. If DELETE_DEVICE is executed while remote I/O communications are operating, remote I/O communications with the specified Slave will stop as soon as it is deleted from the scan list. (Remote I/O communications will continue with the other Slaves.)

Command Buffer Settings

The following table shows the command buffer settings required when using the DELETE_DEVICE command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Set to 0005.
0082	Node address	Word	Set the node address of the desired Slave from 0000 to 003F (0 to 63 decimal).
0084 to 013F	Reserved	---	Set to 0000.

Response

The following table shows the Board's response to the DELETE_DEVICE command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Normal completion: 0005 Error: 8005 Refer to *5-4 Command Errors* for details on errors.
0082 to 013F	Reserved	---	---

5-3-6 START_SCAN (Command Code 06)

The START_SCAN command starts remote I/O communications. Complete the following steps before issuing this command.

• Connect to the network with the DN_ONLINE command.

• Register the Slaves in the scan list with the ADD_DEVICE command.

This command can't be used (an error will occur) if remote I/O communications are already operating or the Board isn't connected to the network.

Command Buffer Settings

The following table shows the command buffer settings required when using the START_SCAN command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Set to 0006.
0082 to 013F	Reserved	---	Set to 0000.

Response

The following table shows the Board's response to the START_SCAN command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Normal completion: 0006 Error: 8006 Refer to *5-4 Command Errors* for details on errors.
0082 to 013F	Reserved	---	---

Note Since the processes required to stop remote I/O communications continue for some time even after STOP_SCAN has been completed normally, an error may occur when START_SCAN is used to restart communications shortly after they were stopped.

5-3-7 STOP_SCAN (Command Code 07)

The STOP_SCAN command stops remote I/O communications. This command can be used only when remote I/O communications are operating.

This command can't be used (an error will occur) if remote I/O communications are already stopped or the Board isn't connected to the network.

Command Buffer Settings

The following table shows the command buffer settings required when using the STOP_SCAN command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Set to 0007.
0082 to 013F	Reserved	---	Set to 0000.

Response

The following table shows the Board's response to the STOP_SCAN command.

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Normal completion: 0007 Error: 8007 Refer to *5-4 Command Errors* for details on errors.
0082 to 013F	Reserved	---	---

Note Since the processes required to stop remote I/O communications continue for some time even after STOP_SCAN has been completed normally, an error may occur when START_SCAN is used to restart communications shortly after they were stopped.

5-4 Command Errors

The following responses will be returned when an error has occurred during execution of a command.

Error Response

Offset address	Parameter	Data type	Explanation
0080	Command code	Word	Contains the command code plus 8000. (The command code with bit 15 ON.)
0082	Command error code	Word	See the following table.
0084 to 013F	Reserved	---	---

Command Error Codes

The following table lists the command error codes returned after an error.

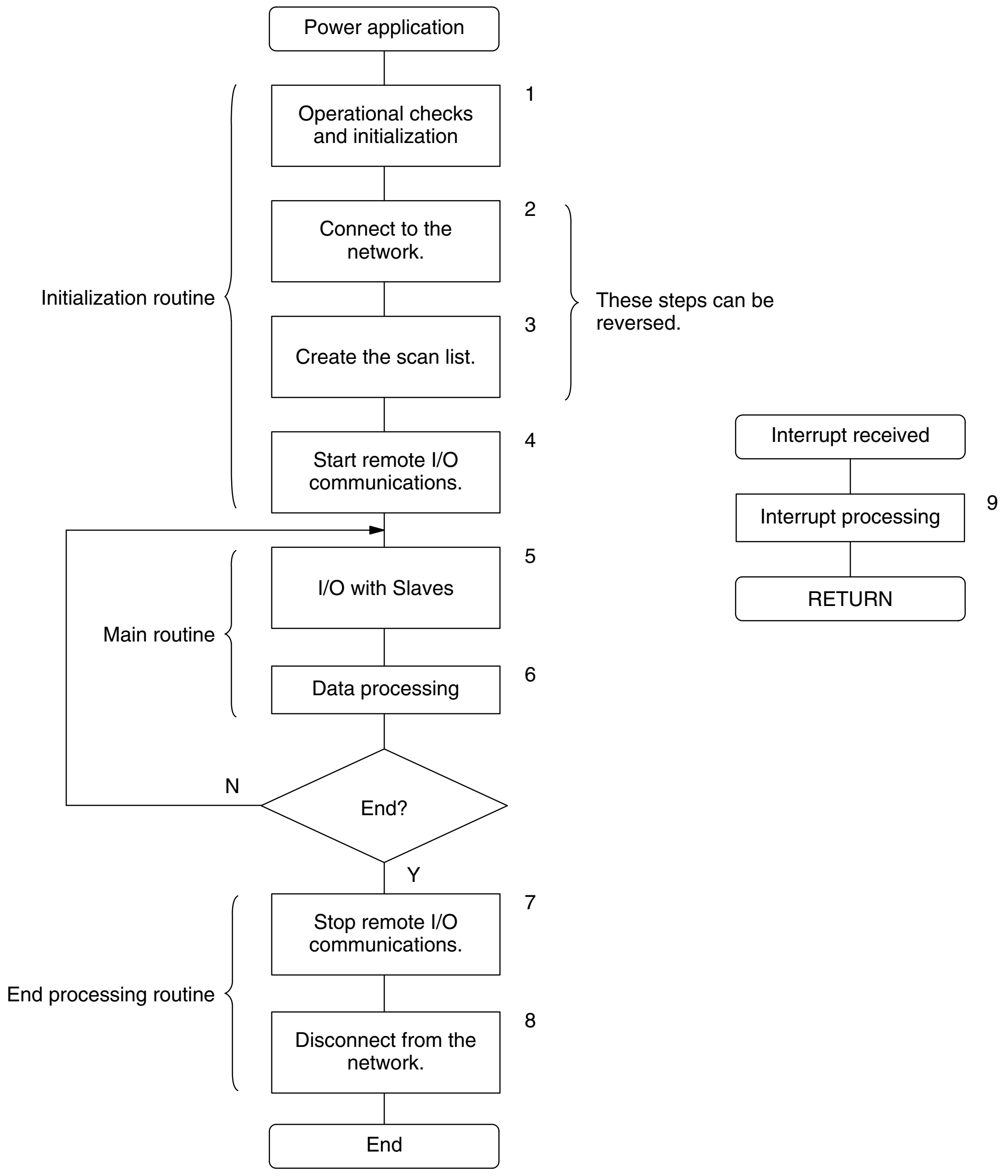
Error code	Name	Meaning	Corresponding command
0000	Reserved	---	---
0001	ERR_CMD	Invalid command	---
0002	ER_MAC	Invalid MAC ID	DN_ONLINE
0003	ERR_BAUD	Invalid baud rate	DN_ONLINE
0004	ERR_DUPMAC	Duplicate MAC ID	DN_ONLINE
0005	ERR_DUPDEV	Duplicate Device	ADD_DEVICE
0006	ERR_NODEV	Device not found	GET_DEVICE DELETE_DEVICE
0007	ERR_OFF	Bus offline	START_SCAN
0008	ERR_ACTIVE	Scanner is active	DN_OFFLINE
0009	ERR_NOTOFF	Bus is not offline	DN_ONLINE
000A	ERR_SCAN	Scanner is running	START_SCAN
000B	ERR_NOTSCAN	Scanner not running	STOP_SCAN
000C	ERR_SCANOFF	Scanner is stopping	START_SCAN STOP_SCAN
000D	ERR_OFFSET	Invalid free memory offset	ADD_DEVICE
000E	ERR_BUSOFF	A Bus Off error occurred.	DN_ONLINE
000F	ERR_CONNECTION _FLAGS	Invalid connection flags combination	ADD_DEVICE
0010	Reserved	---	---
0011	ERR_STROBE_ BUFFER	Invalid strobe buffer size (output must be 1)	ADD_DEVICE
0012 to FFFF	Reserved	---	---

SECTION 6
Example Operation

This section provides examples of basic CompoBus/D VME Board operations.

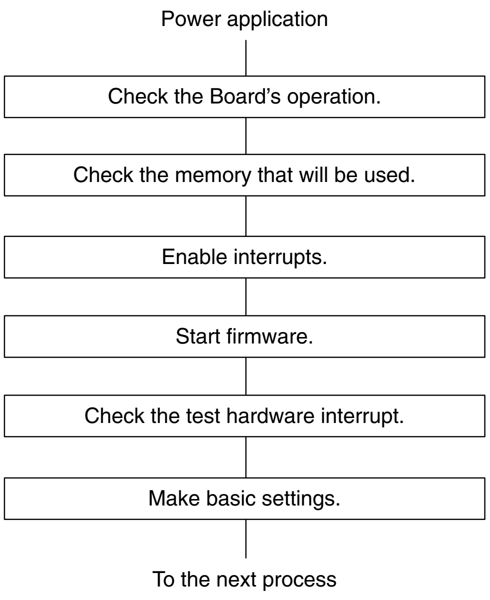
6-1 Operational Flowchart

The following flowchart shows the steps involved in the Board's operation.



Step	Process	Page
1	Check for normal operation and make basic settings such as the Board's memory settings.	51
2	Connect the Board to the CompoBus/D network with the DN_ONLINE command.	54
3	Create the scan list with the ADD_DEVICE command. Register the Slaves that will participate in remote I/O communications.	55
4	Start remote I/O communications with the START_SCAN command.	55
5	Read and write the Slave's I/O data as required.	55
6	Process the I/O data as required.	---
7	When you have finished using remote I/O communications, stop them with the STOP_SCAN command.	57
8	When you have finished using the Board, remove it from the network with the DN_OFFLINE command.	57
9	Read the cause of interrupts and process them as they occur. After completing interrupt processing, clear the interrupt so that other interrupts can be received.	58

6-2 Operational Checks and Initialization


Power application
Check the Board's operation.
Check the memory that will be used.
Enable interrupts.
Start firmware.
Check the test hardware interrupt.
Make basic settings.
To the next process

6-2-1 Checking the Board's Operation

Check that the Board is operating normally. This step isn't absolutely necessary when creating a program.

The Board might not be installed properly if the written data differs from the read data or a VME bus error occurs in the following operations. If one of these problems occurs, stop checking the Board's operation and check its installation.

1, 2, 3...

1. Write 20 in the setting/status register and then read the contents of the register to confirm that it is set to 20.
   The setting/status register is located in the second byte of short I/O address space (the base address + 1). The base address is set with pins 1 to 6 of the Board's DIP switch.
   The default startup setting of the setting/status register is 20.
2. Write 00 in the memory control register and then read the contents of the register to confirm that it is set to 00.
   The memory control register is located in the sixth byte of short I/O address space (the base address + 5).
3. Repeat step 2. while incrementing the write data from 01 to FE.
4. Write 00 in the interrupt register and then read the contents of the register to confirm that it is set to 00.
   The interrupt register is located in the fourth byte of short I/O address space (the base address + 3).
5. Repeat step 4. while incrementing the write data from 01 to FF.

6-2-2 Checking Memory in Standard Address Space

Check the status of the standard address space that the Board uses for shared RAM as shown below.

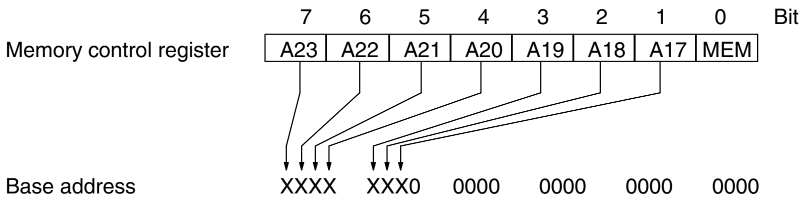
Conflict Checks

This step isn't absolutely necessary when creating a program.

Read the memory in standard address space that is being used. Read 128-KB at a time from the beginning and check for VME bus errors in all of the addresses. When a VME bus error doesn't occur, there is a possibility that the address is already allocated to another device, so change the memory addresses being used.

Base Address Setting

1, 2, 3...
1. Set the location of the shared RAM's starting address (base address) in standard address space with A23 to A17 (bits 7 to 1) of the memory control register.
   The memory control register is the sixth byte in short I/O address space (base address + 5) and the base address of short I/O address space is set with the DIP switch.



2. Write 1 to the memory control register's MEN bit. (The offset address is the offset from the beginning of short I/O address space.)

Name	Offset address	Bit	Function
MEN	5 (in short I/O space)	Bit 0	0: Shared RAM access disabled 1: Shared RAM access enabled

Shared RAM Check

This step isn't absolutely necessary when creating a program.

1, 2, 3...
1. Perform a writing check of all shared RAM (128 KB). We recommend writing the lower 4 digits of each offset address to the address. For example, write 0000 to address 00000, 0002 to address 00002, etc.
2. Read the contents of the addresses written in step 1. and check that the contents are correct.
3. There might be an error in the shared RAM if the written data doesn't match the read data. In this case, replace the Board with a new one.

6-2-3 Enabling Interrupts

Use the following procedure to enable interrupts when interrupts are being used.

1, 2, 3...
1. Set the interrupt ID in the interrupt register.

Name	Offset address	Bit	Function
Interrupt register	3 (in short I/O space)	---	Contains the interrupt ID.

2. Write 1 to IRQ in the setting/status register to clear the interrupt signal.

Name	Offset address	Bit	Function
IRQ	1 (in short I/O space)	Bit 2	Reading: Shows interrupt status. Writing 1: Clears interrupt signal.

3. The Board generates a test hardware interrupt at startup. When performing the hardware interrupt test, prepare a program to process the test interrupt and write 1 to IRQE in the setting/status register to enable hardware interrupts.

Name	Offset address	Bit	Function
IRQE	1 (in short I/O space)	Bit 3	0: Disables hardware interrupts. 1: Enables hardware interrupts.

6-2-4 Starting the Firmware

Check for firmware (applications within the Board) startup and normal startup.

1, 2, 3...
1. Clear all of the shared RAM (128 KB) to 0000.

2. Write 0003 to the application module header’s shared RAM size area.

Area	Offset address	Data type	Function
Shared RAM size	0002 (in host interface block)	Word	Always set to 0003.

3. Write 1 to WDI in the setting/status register to disable the watchdog timer. The self-diagnostic function will generate an error if the watchdog timer is enabled during the firmware startup processing, so the watchdog timer must be disabled.

Name	Offset address	Bit	Function
WDI	1 (in short I/O space)	Bit 5	0: Watchdog timer enabled 1: Watchdog timer disabled

4. Write 0 and then 1 to RUN in the setting/status register two times. The firmware starts up and the 2-s timer starts when RUN goes from 0 to 1.

Name	Offset address	Bit	Function
RUN	1 (in short I/O space)	Bit 7	When this bit is read, a 0 indicates that the Board is resetting and a 1 indicates that firmware is operating. When this bit is set from 0 to 1, the firmware starts up.

5. Check the contents of the application module header’s module type area after the timer times out.

Area	Offset address	Data type	Function
Module type	0000 (in host interface block)	Word	DN: Application started normally. ER: An error occurred. When an error occurred, the error information will be stored in the Board information (0040 to 007F).

6. Confirm that the module type area contains “DN” (444E in hexadecimal). The Board has started normally if the module type is “DN.”
   If the module type is “ER,” the error information will be stored in the Board information (0040 to 007F). Refer to *8-2-1 Errors Indicated in the Board Information Area* for details on troubleshooting.

   If the module type is neither “DN” nor “ER”, double check the Board’s register settings and check the module type again. Replace the Board if the problem recurs.

Note Be sure to wait at least 2 s after setting the setting/status register’s RUN bit from 0 to 1. The Board’s initialization processes might not be completed within 2 s even if the module type has been set to “DN.”

6-2-5 Checking the Test Hardware Interrupt

The Board generates a test hardware interrupt at startup. When 1 has been written to IRQE in the setting/status register to perform the hardware interrupt test as described in *6-2-3 Enabling Interrupts*, check the hardware interrupt with the following procedure.

If a hardware interrupt hasn’t been generated, double check the Board’s register settings and check the interrupt again. Replace the Board if the problem recurs.

1, 2, 3...

1. Read the content of the interrupt register and confirm that it matches the value set for the interrupt ID.

Area	Offset address	Bit	Function
Interrupt register	3 (in short I/O space)	---	Contains the interrupt ID.

2. Confirm that the IRQ bit in the setting/status register is set to 1.

Name	Offset address	Bit	Function
IRQ	1 (in short I/O space)	Bit 2	Reading: Shows interrupt status. Writing 1: Clears interrupt signal.

3. Run the previously prepared interrupt processing program to confirm that a hardware interrupt has been generated.
4. Write 1 to the IRQ bit in the setting/status register to clear the interrupt signal.

Name	Offset address	Bit	Function
IRQ	1 (in short I/O space)	Bit 2	Reading: Shows interrupt status. Writing 1: Clears interrupt signal.

6-2-6 Basic Settings

If the Board's firmware has started up normally, make the basic settings in the setting/status register.

1, 2, 3...

1. Write 0 to WDI in the setting/status register to enable the watchdog timer.

Name	Offset address	Bit	Function
WDI	1 (in short I/O space)	Bit 5	0: Watchdog timer enabled 1: Watchdog timer disabled

2. Write 1 to HLTH in the setting/status register to enable the HEALTH indicator. The indicator will be green when enabled.

Name	Offset address	Bit	Function
HLTH	1 (in short I/O space)	Bit 4	0: HEALTH indicator disabled 1: HEALTH indicator enabled

3. Write 0 or 1 to SYS in the setting/status register as required. The SYSFAIL signal is enabled when SYS is 1.

   When the SYSFAIL signal is disabled, the FAIL indicator will be off and the PASS indicator will be on regardless of the status of the SYSFAIL signal.

   When the SYSFAIL signal is enabled, the FAIL indicator will reflect the status of the SYSFAIL signal and the PASS indicator will be off.

Name	Offset address	Bit	Function
SYS	1 (in short I/O space)	Bit 6	0: SYSFAIL signal disabled 1: SYSFAIL signal enabled

4. Write 0 or 1 to IRQE in the setting/status register as required. Hardware interrupts are enabled when IRQE is 1, so set this bit to 1 when using hardware interrupts.

Name	Offset address	Bit	Function
IRQE	1 (in short I/O space)	Bit 3	0: Hardware interrupts disabled 1: Hardware interrupts enabled

6-3 Connecting to the Network

Use the DN_ONLINE command to connect the Board to the CompoBus/D network. Refer to *5-3-1 DN_ONLINE (Command Code 01)* for details.

The DN_ONLINE command can be used only when the Board is disconnected from the network.

6-4 Creating the Scan List

Use the ADD_DEVICE command to register in the scan list all of the Slaves that will participate in remote I/O communications with the Board. Refer to *5-3-3 ADD_DEVICE (Command Code 03)* for details on registering Slaves.

Only one Slave can be registered each time that ADD_DEVICE is executed. Repeat the ADD_DEVICE command until all of the Slaves have been registered. When necessary, Slaves can be removed from the scan list with the DELETE_DEVICE command.

Slaves can be added to or removed from the scan list any time after firmware startup. If a Slave is added to or removed from the scan list while remote I/O communications are operating, communications with the specified Slave will be started or stopped immediately.

6-5 Starting Remote I/O Communications

Use the START_SCAN command to start remote I/O communications. Refer to *5-3-6 START_SCAN (Command Code 06)* for more details.

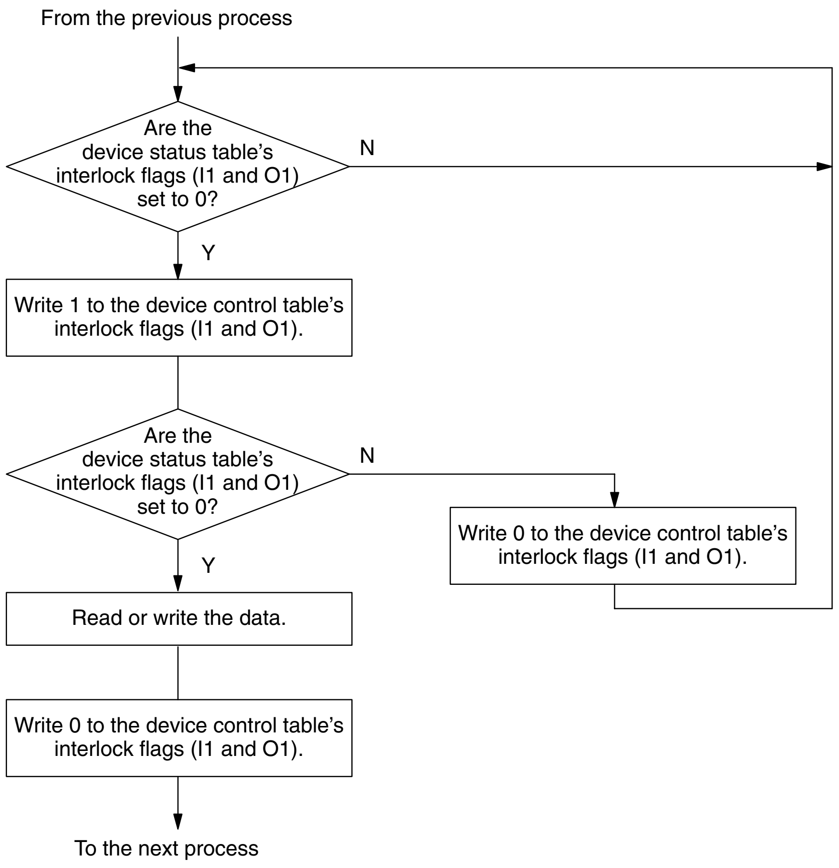
START_SCAN can be used only when remote I/O communications are stopped. Confirm that the client status block's status code contains 00 before executing the START_SCAN command.

Area	Offset address	Data type	Function
Status code	0140 (in host interface block)	Word	00: Remote I/O communications stopped 01: Remote I/O communications operating

6-6 Transferring I/O with Slaves

I/O with the Slaves is exchanged between the Board and the VME Host through free memory. The number of bytes and the memory addresses used for each Slave's I/O is registered in the scan list.

Use the following procedure when transferring I/O data to and from free memory, so that memory isn't accessed simultaneously from the VME Host and the Board.


From the previous process
Are the device status table's interlock flags (I1 and O1) set to 0?
N
Y
Write 1 to the device control table's interlock flags (I1 and O1).
Are the device status table's interlock flags (I1 and O1) set to 0?
N
Write 0 to the device control table's interlock flags (I1 and O1).
Y
Read or write the data.
Write 0 to the device control table's interlock flags (I1 and O1).
To the next process

1, 2, 3...

1. Check whether the desired Slave's interlock flag (I1 for input, O1 for output) is set to 0 (reading/writing from the VME Host enabled). The interlock flags are located in the Slave's device status table.

   Check I1 when inputting from the Slave, O1 when outputting to the Slave. If the flag is 1, wait until it is reset to 0.

Name	Offset address	Bit	Function
I1	01C0 + 10 × node address + 1 (in host interface block)	Bit 1	Input data access status 0: Reading from VME Host enabled 1: Reading from VME Host disabled (when the Board is writing data)
O1	01C0 + 10 × node address + 1 (in host interface block)	Bit 3	Output data access status 0: Writing from VME Host enabled 1: Writing from VME Host disabled (when the Board is reading data)

2. Write 1 to the desired Slave's interlock control bit (I1 for input, O1 for output). The interlock control bits are located in the Slave's device control table.

Write 1 to I1 when inputting from the Slave, write 1 to O1 when outputting to the Slave.

Name	Offset address	Bit	Function
I1	05C0 + 10 × node address + 1 (in the host interface block)	Bit 1	Controls input data writing. 0: Writing from Board enabled 1: Writing from Board disabled
O1	05C0 + 10 × node address + 1 (in the host interface block)	Bit 3	Controls output data reading. 0: Reading from Board enabled 1: Reading from Board disabled

3. Doublecheck the status of the Slave's interlock flag (I1 for input, O1 for output) that was checked in step 1. The flag might have been set to 1 while during step 2., so doublecheck the flag to make sure that the VME Host and Board don't access memory simultaneously.

   Check I1 when inputting from the Slave, O1 when outputting to the Slave. If the flag is 1, reset (to 0) the interlock control bit that was set in step 2. and return to step 1.
4. Read or write the desired input or output data.
5. Reset (to 0) the interlock control bit that was set in step 2.
   Write 0 to control bit I1 when inputting from the Slave, write 0 to control bit O1 when outputting to the Slave.

Note A fatal error will occur if the interlock control bit remains set to 1, because the bit disables reading or writing data from the Board. Set these control bits to 1 only when transferring data from the VME Host and always reset the bits to 0 when the data transfer is completed.

6-7 Stopping Remote I/O Communications

Use the STOP_SCAN command to stop remote I/O communications. Refer to *5-3-7 STOP_SCAN (Command Code 07)* for more details.

STOP_SCAN can be used only when remote I/O communications are operating. Confirm that the client status block's status code contains 01 before executing the STOP_SCAN command.

Area	Offset address	Data type	Function
Status code	0140 (in host interface block)	Word	00: Remote I/O communications stopped 01: Remote I/O communications operating

6-8 Disconnecting from the Network

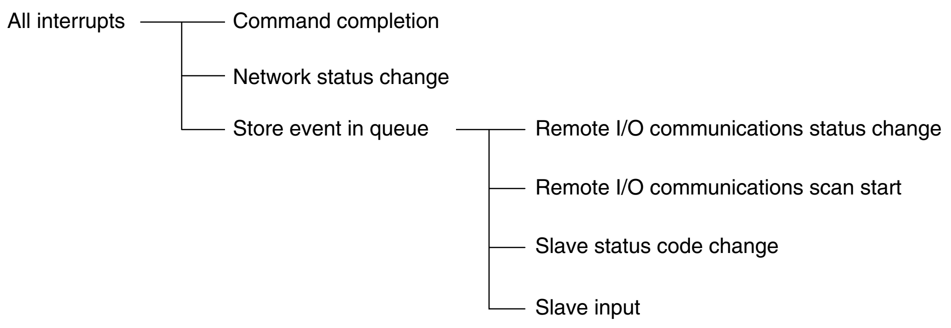
Use the DN_OFFLINE command to withdraw the Board from the CompoBus/D network. Refer to *5-3-2 DN_OFFLINE (Command Code 02)* for details.

The DN_OFFLINE command can be used only when the Board is connected to the network.

6-9 Processing Interrupts

6-9-1 Interrupt Sources

The following diagram shows the sources of interrupts that can be received by the Board.

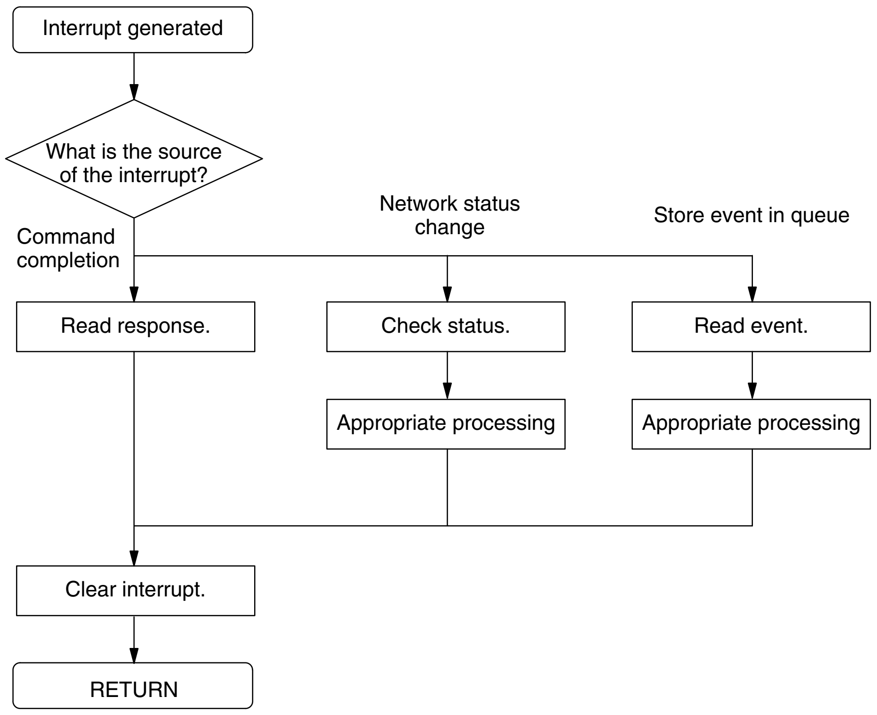


The status of these interrupt sources are reflected separately in the following locations. These interrupts can be enabled and disabled independently. Enable the interrupts beforehand.

Source	Status location	Control location
All interrupts	Hardware interrupt signal	The Board's DIP switch
		IRQE (bit 3 of the setting/status register)
	IRQ (bit 2 of the setting/status register)	None
Command completion	CM bits (bit 0) of the application module header's interrupt status A and B (002C and 002D)	Bit CM (bit 0) of the application module header's interrupt control area (002A)
Network status change	BS bits (bit 1) of the application module header's interrupt status A and B (002C and 002D)	Bit BS (bit 1) of the application module header's interrupt control area (002A)
Storing event in queue	QE bits (bit 2) of the application module header's interrupt status A and B (002C and 002D)	Bit QE (bit 2) of the application module header's interrupt control area (002A)
	The event notification queue's event queue (09C6 to 0DBF)	
Remote I/O communications status change	The client status block's status code (0140)	Bit C (bit 0) of the client control block's event queue control area (015E)
Remote I/O communications scan start	Bit S (bit 0) of the client status block's remote I/O communications event area (0142)	Bit S (bit 2) of the client control block's event queue control area (015E)
Slave status code change (separate for each Slave)	The Slave's status code area in the device status table (01C0 + 10 × node address)	Bit C (bit 0) of the event queue control bits in the device control table (05C0 + 10 × node address + 14)
Slave input (separate for each Slave)	Bit U (bit 0) of the Slave's I/O event flag area in the device status table (01C0 + 10 × node address + 2)	Bit I1 (bit 2) of the event queue control bits in the device control table (05C0 + 10 × node address + 14)

6-9-2 Summary of Interrupt Processing

The following diagram shows typical steps taken when an interrupt is generated from the Board.



6-9-3 Interrupt Processing Operations

Check Interrupt Source

When the interrupt has been generated from the Board, first find the source of the interrupt with the following procedure.

Even if hardware interrupts aren't being used, you can see when an interrupt event has occurred by checking interrupt status A and B in the application module header.

Read interrupt status A and interrupt status B and see which bit is set to 1. If more than one bit is set to 1, proceed to the next step and process the interrupts one at a time.

Area	Offset address	Data type	Function
Interrupt status A	002C (in host interface block)	Byte	Bit 0:CM (Command completion) Bit 1: BS (Network status change) Bit 2: QE (Store event in queue)
Interrupt status B	002D (in host interface block)	Byte	

Use the following procedure to get a single result when checking the status of bits in interrupt status A and B.

1, 2, 3...
1. Read "interrupt status A" and store the content in variable A.
2. Write 0 to the appropriate bits in "interrupt status A" to clear the interrupt(s).
3. Read "interrupt status B" and store the content in variable B.
4. Write 0 to the appropriate bits in "interrupt status B" to clear the interrupt(s).
5. Use the logical OR of variable A and variable B as the interrupt status.

Reading the Response

When the CM bit (bit 0) is set to 1 as the interrupt source, it indicates that a command executed from the VME Host has been completed and the response has been stored in the command buffer.

Read the response from the command buffer (0080 to 013F). The content of the response depends on the command that was executed and the results of that execution. Refer to *Section 5 Command Usage* for more details.

Checking the Network Status

When the BS bit (bit 1) is set to 1 as the interrupt source, it indicates a change in the network and the contents of the application module header's network status area.

1, 2, 3...

1. Read the contents of the application module header's network status area.

Area	Offset address	Data type	Function
Network status	0030 (in host interface block)	Word	Contains network status information.

   Refer to 29 for details on the network status area.

2. Perform the required processing.

Reading the Event Queue

When the QE bit (bit 2) is set to 1 as the interrupt source, it indicates that some kind of event has occurred and has been stored in the event queue. Read the event queue to determine the nature of the event.

1, 2, 3...

1. Compare the contents of the event notification queue's "queue IN" and "queue OUT" areas.

Area	Offset address	Data type	Function
Queue IN	09C2 (in host interface block)	Word	Contains the address where the next event will be stored. (Written by the Board.)
Queue OUT	09C4 (in host interface block)	Word	Contains the address where the next event will be read. (Written by the VME Host.)

   If queue IN and queue OUT are equal, an event hasn't been stored in the queue.

2. Read the event notification queue's "queue status A" and "queue status B" areas and see which bits have been set to 1.

Area	Offset address	Data type	Function
Queue status A	09C0 (in host interface block)	Byte	Bit 0: O (Queue overrun occurred.)
Queue status B	09C1 (in host interface block)	Byte	

   A queue overrun will occur if there are more than 509 unread events. In this case, events beyond 509 will be lost.

   Use the following procedure to get a single result when checking the status of bits in queue status A and B.

   a) Read "queue status A" and store the content in variable A.
   b) Write 0 to the appropriate bit in "queue status A" to clear it.
   c) Read "queue status B" and store the content in variable B.
   d) Write 0 to the appropriate bit in "queue status B" to clear it.
   e) Use the logical OR of variable A and variable B as the queue status.

3. Read the event notification queue's queue OUT area to determine the beginning address of the events which haven't been read.

Area	Offset address	Data type	Function
Queue OUT	09C4 (in host interface block)	Word	Contains the beginning address of events which haven't been read.

The address indicated in the queue OUT area contains the event source code and the next address contains the event ID.

4. Read the event source code and the event ID.

Area	Offset address	Data type	Function
Event source code	---	Byte	Contains a code that shows where the event occurred.
Event ID	---	Byte	Contains an ID that shows what kind of event occurred.

Refer to 35 for details on event source codes and event IDs.

5. Add 2 to the address indicated in the queue OUT area to get the beginning address for the next event.

If the result of the addition is 0DC0, which exceeds the upper limit of the event queue, store 09C6 in queue OUT rather than 0DC0.

Area	Offset address	Data type	Function
Queue OUT	09C4 (in host interface block)	Word	Contains the beginning address of events which haven't been read.

6. Check the following locations depending on the event that occurred.

Event source code	Event ID	Interrupt source	Location to check
40	00	Remote I/O communications status change	Check the client status block's status code (0140).
40	02	Remote I/O communications scan start	Check bit S (bit 0) of the client status block's remote I/O communications event area (0142) and write 0 to S.
00 to 3F	00	Slave status code change	Check the Slave's status code area in the device status table. (01C0 + 10 × node address)
00 to 3F	02	Slave input	Check bit U (bit 0) of the Slave's I/O event flag area in the device status table (01C0 + 10 × node address + 2) and write 0 to U.

7. Perform the processing required for the event that occurred. Perform the processing appropriate for the system being used.
8. Compare the contents of the event notification queue's "queue IN" and "queue OUT" areas. If queue IN and queue OUT aren't equal, there are still unread events. In this case, repeat the operation from step 3.

Area	Offset address	Data type	Function
Queue IN	09C2 (in host interface block)	Word	Contains the address where the next event will be stored. (Written by the Board.)
Queue OUT	09C4 (in host interface block)	Word	Contains the address where the next event will be read. (Written by the VME Host.)

If queue IN and queue OUT are equal, an event hasn't been stored in the queue.

Clearing the Interrupts

When another bit has been set to 1, return to the interrupt processing explanation beginning with *Reading the Response* on page 59 and process the next interrupt source.

When all of the interrupt sources have been processed, clear the interrupt signals so that interrupts can occur again.

Note The Board will make new interrupts wait if new interrupts occur during interrupt processing. When the original interrupts are cleared, the waiting interrupts will occur immediately.

Write 1 to IRQ in the setting/status register to clear the interrupt signal.

Name	Offset address	Bit	Function
IRQ	1 (in short I/O space)	Bit 2	Reading: Shows interrupt status. Writing 1: Clears interrupt signal.

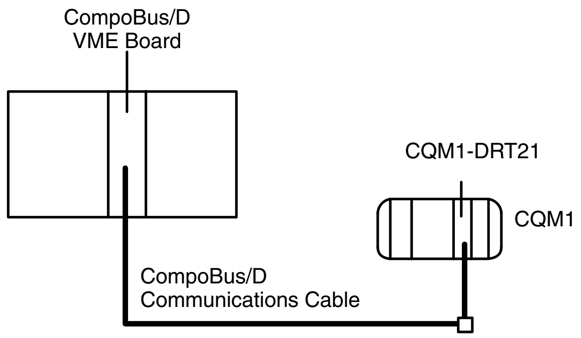
This completes interrupt processing and returns the Board to normal processing.

SECTION 7
Sample Program

This section provides a sample program with a variety of Board operations.

7-1 Operating Conditions

7-1-1 Example System Configuration



7-1-2 Required Settings

Board Settings

There are no special Board settings for the sample program. Default Board settings are used.

I/O Link Unit Settings

The CQM1-DRT21 I/O Link Unit's baud rate is set to 500 K baud and its node address is set to 1.

7-1-3 About the Sample Program

- The CQM1-DRT21 I/O Link Unit takes in input data and outputs output data.
- The sample program shown here is written for a CPU Unit that uses a Motorola MC68000, but only the parts required for Board operation are described. Programming such as the initial settings for the CPU board is omitted.

7-2 Sample Program

```
******************************************************************************************
*
*   CompoBus/D VME Board Sample Program (without interrupts)
*
******************************************************************************************
*   Baud rate:                              500 K baud
*   Master (VME Board) node address:        63 (0x3F)
*   Slave (CQM1-DRT21) node address:        1 (0x01)
******************************************************************************************


******************************************************************************************
*   VME Board Initial Settings and Startup
******************************************************************************************
Main:
      move.l   #STACK,a7                    ** Set stack pointer.
      move.b   #BaseAdrValue,MemoryControlReg ** Set base address (to memory control register).
      move.l   #HostIFBaseAddress,a6        ** Set base address of host interface block in A6.

                                            ** Clear shared RAM.
      move.l   #BaseAddress,a0              **     Set base address of shared RAM in A0.
      move.l   #0x20000,d0                  **     Set shared RAM size in D0.
Main01:                                     **     Loop until D0 becomes 0 and clear memory.
      move.b   #0x00,(a0)+                  **       Clear A0 and increment A0.
      subi.l   #1,d0                        **       Decrement D0.
      bne      main01                       **       Return to start of loop if D0 isn’t 0.

                                            ** Set shared RAM size.
      move.b   #0x03,(0x0002,a6)            **     Set application module header’s window size.

                                            ** Firmware startup (Toggle RUN bit twice.)
      move.b   #0x20,ConfigStatusReg        **     Set initial value. (RUN GOES ON: Starts Board.)
      move.b   #0xa4,ConfigStatusReg        **     RUN/WDI/IRQ ON     (WDI ON: Disables WDT.)
      move.b   #0x20,ConfigStatusReg        **     Set initial value. (IRQ ON: Clears interrupts.)
      move.b   #0xa4,ConfigStatusReg        **     RUN/WDI/IRQ ON

      move.l   #Initial_Wait_Timer,d1       ** Wait for firmware startup. (About 2 seconds)
Main02:
      subi.l   #1,d1
      bne      main02
                                            ** Check firmware startup.
      move.w   (a6),d0                      **     Read application module header’s module type.
      cmp.w    #0x444e,d0                   **     Check if module type is DN (0x444E).
      beq      main03                       **     If DN, go to startup completion process.
      cmp.w    #0x4552,d0                   **     Check if module type is ER (0x4552).
      beq      Initial_Error                **     If ER, go to startup error process.
      jmp      Initial_Error                **     If not ER or DN, go to startup error process.

Main03:
      move.b   #0x90,ConfigStatusReg        ** Complete firmware startup.
                                            ** Setting/status register’s WDI OFF, HEALTH ON
                                            **                  (WDI OFF: Enables WDT.)
                                            **                  (HEALTH ON: HEALTH LED green.)
```

```
*******************************************************************************************
*   DN_ONLINE Command Processing
*******************************************************************************************
      move.l   #Dn_Online_Table,a0             ** Set command table starting address in A0.
      jsr      Send_Command                    ** Call command execution subroutine.
      cmp.w    #0,d0                           **      When there is an error in command execution,
      bne      Send_Command_Error              **      go to command execution error process.
*******************************************************************************************
*   ADD_DEVICE Command Processing (Register CQM1-DRT21 only.)
*******************************************************************************************
      move.l   #Add_Device_Table_DRT21,a0      ** Set command table starting address in A0.
      jsr      Send_Command                    ** Call command execution subroutine.
      cmp.w    #0,d0                           **      When there is an error in command execution,
      bne      Send_Command_Error              **      go to command execution error process.
*******************************************************************************************
*   START_SCAN Command Processing
*******************************************************************************************
      move.l   #Start_Scan_Table,a0            ** Set command table starting address in A0.
      jsr      Send_Command                    ** Call command execution subroutine.
      cmp.w    #0,d0                           **      When there is an error in command execution,
      bne      Send_Command_Error              **      go to command execution error process.
*******************************************************************************************
*   CQM1-DRT21's I/O Data Processing
*******************************************************************************************
Main10:
                                               ** Output the output data to the CQM1-DRT21.
      move.l   #Output_Data,a0                 **      Set output data's transfer address in A0.
      move.l   #HostIFBaseAddress+0x1000,a1    **      Set free memory's transfer address in A1.
      move.w   2,d0                            **      Set number of output bytes (2) in D0.
      move.w   1,d1                            **      Set the CQM1-DRT21's node address in D1.
      jsr      Write_IO_Data                   **      Call output data writing subroutine.

                                               ** Input the input data from the CQM1-DRT21.
      move.l   #HostIFBaseAddress+0x2000,a0    **      Set free memory's transfer address in A0.
      move.l   #Input_Data,a1                  **      Set input data's transfer address in A1.
      move.w   2,d0                            **      Set number of input bytes (2) in D0.
      move.w   1,d1                            **      Set the CQM1-DRT21's node address in D1.
      jsr      Read_IO_Data                    **      Call input data reading subroutine.

      bra      main10

*******************************************************************************************
*   Command Execution Subroutine
*      Input :  a0: Command table address
*      Output:  d0: Result (word)
*                    0: Normal completion
*                   -1: Error completion
*                   -2: Timeout
*******************************************************************************************
Send_Command:
                                               ** Clear CM bits in interrupt status A/B.
      move.b   (0x002c,a6),d0                  **      Read interrupt status A.
      andi     #0xfe,d0                        **      Clear CM bit in status A.
      move.b   d0,(0x002c,a6)                  **      Write result to interrupt status A.
      move.b   (0x002d,a6),d0                  **      Read interrupt status B.
      andi      #0xfe,d0                         **      Clear CM bit in status B.
      move.b    d0,(0x002d,a6)                   **      Write result to interrupt status B.
                                                 **  Clear command buffer area.
      move.l    #HostIFBaseAddress+0x0080,a1     **      Set command buffer’s starting address in A1.
      move.w    #0x00c0,d0                       **      Set command buffer’s size in D0.
send_command01:                                  **      Loop until D0 becomes 0 and clear memory.
      move.b    #0,(a1)+                         **        Clear A1 and increment A1.
      subi.w    #1,d0                            **        Decrement D0.
      bne       send_command01                   **        Return to start of loop if D0 isn’t 0.

                                                 **  Set the command data.
      move.l    a0,a1                            **      Set command table’s starting address in A1.
      move.l    #HostIFBaseAddress+0x0080,a2     **      Set command buffer’s starting address in A2.
send_command02:                                  **      Copy command table to command buffer.
      move.w    (a1)+,d0                         **        Set command table data from A1 in D0.
      cmp.w     #0xffff,d0                       **        Check for end of table.
      beq       send_command03                   **        Complete copying if it is the end of table.
      move.w    d0,(a2)+                         **        If it isn’t the end of table, set D0 in the
                                                 **        command buffer indicated in A2.
      jmp       send_command02                   **        Loop to beginning.

send_command03:                                  **  Execute command. (Interrupt to VME Board)
      move.b    ConfigStatusReg,d0               **      Read the setting/status register to D0.
      ori       #0x01,d0                         **      Turn the CINT bit ON.
      move.b    d0,ConfigStatusReg               **      Write D0 to the setting/status register.

                                                 **  Wait for command completion. (Wait until the CM
                                                 **  bits in interrupt status A/B go ON.)
      move.l    #Command_Wait_Timer,d3           **      Set the waiting time (about 2 s).
send_Command04:                                  **      Check the CM bits in interrupt status A/B.
      move.b    (0x002c,a6),d0                   **        Read interrupt status A.
      move.b    d0,d1                            **        Store interrupt status A in D1.
      andi      #0xfe,d0                         **        Clear the CM bit in interrupt status A.
      move.b    d0,(0x002c,a6)                   **        Write data to interrupt status A.
      move.b    (0x002d,a6),d0                   **        Read interrupt status B.
      move.b    d0,d2                            **        Store interrupt status B in D2.
      andi      #0xfe,d0                         **        Clear the CM bit in interrupt status B.
      move.b    d0,(0x002d,a6)                   **        Write data to interrupt status B.
      or.b      d1,d2                            **        Take logical OR of interrupt status A and B.
      andi      #0x01,d2                         **        Check whether the CM bit is ON.
      bne       send_command05                   **        If CM is ON, go to response check process.
      subi.l    #1,d3                            **        If CM isn’t ON, decrement the waiting time.
      bne       send_command04                   **        Continue loop if waiting time hasn’t elapsed.
      move.w    #-2,d1                           **      When a waiting time timeout occurs, set -2 in
      jmp       send_command_end                 **      the return code register and go to end process.

send_command05:                                  **  Response check process
      move.w    #0,d1                            **      Clear the return code register.
      move.b    (0x0081,a6),d0                   **      Read the response (high byte).
      andi      #0x80,d0                         **      Check for error response. (Error if MSB is ON.)
      beq       send_command_end                 **      Go to end process (return code 0) if no error.
      move.w    #-1,d1                           **      Set -1 in return code register if error.

send_command_end:                                **  End process
                                                 **      Clear the setting/status register’s CINT bit.
      move.b   ConfigStatusReg,d0                **        Read the setting/status register to D0.
      andi     #0xfe,d0                          **        Turn the CINT bit OFF.
      move.b   d0,ConfigStatusReg                **        Write D0 to the setting/status register.

      move.w   d1,d0                             **       Set the return code to D0.
      rts                                        **       Return.

**********************************************************************************************
*   Input Data Read Subroutine
*      Input :  a0:  Read source address (address in memory pool)
*               a1:  Read destination address
*               d0:  Number of bytes to read (word)
*               d1:  Slave node address (word)
*      Output:  d0:  Result (word)
*                     0:  Normal completion
*                    -1:  VME Board accessing memory
**********************************************************************************************
Read_IO_Data:
      move.l   #HostIFBaseAddress+0x01c0,a2      **  Set starting address of device status table in A2.
      move.w   #16,d2                            **  Calculate address offset for the specified Slave.
      mulu     d1,d2                             **       The offset is 16 times the node address.
      adda.w   d2,a2                             **  Set the starting address for the Slave in A2.

      move.l   #HostIFBaseAddress+0x05c0,a3      **  Set starting address of device control table in A3
      move.w   #16,d2                            **  Calculate address offset for the specified Slave.
      mulu     d1,d2                             **       The offset is 16 times the node address.
      adda.w   d2,a3                             **  Set the starting address for the Slave in A3.

read_io_data_01:                                 **  Check the (status) input area's interlock flag.
      move.b   (0x01,a2),d2                      **       Read the I/O status flags.
      andi     #0x02,d2                          **       Check the input area's interlock flag.
      beq      read_io_data_02                   **       Go to next process if the flag = 0.
      move.w   #-1,d0                            **       If the flag = 1, set the return code to -1
                                                 **       because the VME Board is accessing memory.
      jmp      read_io_data_end                  **       Go to end process.

read_io_data_02:                                 **  Set the (control) input area's interlock bit.
      move.b   (a3),d2                           **       Read the interlock bit.
      ori      #0x02,d2                          **       Turn the input area's interlock bit ON.
      move.b   d2,(a3)                           **       Set the interlock bit.

read_io_data_03:                                 **  Check the (status) input area's interlock flag.
      move.b   (0x01,a2),d2                      **       Read the I/O status flags.
      andi     #0x02,d2                          **       Check the input area's interlock flag.
      beq      read_io_data_04                   **       Go to next process if flag = 0.
                                                 **       If flag = 1, the Board requested access too, so
      move.b   (a3),d2                           **       clear the (control) input area's interlock bit.
      andi     #0xfd,d2                          **       Turn the input area's interlock bit OFF.
      move.b   d2,(a3)                           **       Clear the interlock bit.
      jmp      read_io_data_01                   ** Start over from the beginning of the process.

read_io_data_04:                                 **  Read input data.
      move.b   (a0)+,d2                          **       Read data from read source (in memory pool).
      move.b   d2,(a1)+                          **       Write data to read destination (specified area)
      subi.l   #1,d0                             **       Decrement D0 (number of bytes to read).
      bne      read_io_data_04                 **      Repeat until the specified amount is read.

read_io_data_05:                               ** Clear the (control) input area's interlock bit.
      move.b   (a3),d2                         **      Read the interlock bit.
      andi     #0xfd,d2                        **      Turn the input area's interlock bit OFF.
      move.b   d2,(a3)                         **      Clear the interlock bit.

      move.w   #0,d0                           ** Normal completion (Return code set to 0.)

read_io_data_end:                              ** End process
      rts                                      **      Return

*****************************************************************************************************
*   Output Data write Subroutine
*      Input :  a0:  Write source address
*               a1:  Write destination address (in memory pool)
*               d0:  Number of bytes to write (word)
*               d1:  Slave node address (word)
*      Output:  d0:  Result (word)
*                     0:  Normal completion
*                    -1:  VME Board accessing memory
*****************************************************************************************************
Write_IO_Data:
      move.l   #HostIFBaseAddress+0x01c0,a2    ** Set starting address of device status table in A2.
      move.w   #16,d2                          ** Calculate address offset for the specified Slave.
      mulu     d1,d2                           **      The offset is 16 times the node address.
      adda.w   d2,a2                           ** Set the starting address for the Slave in A2.

      move.l   #HostIFBaseAddress+0x05c0,a3    ** Set starting address of device control table in A3
      move.w   #16,d2                          ** Calculate address offset for the specified Slave.
      mulu     d1,d2                           **      The offset is 16 times the node address.
      adda.w   d2,a3                           ** Set the starting address for the Slave in A3.

write_io_data_01:                              ** Check the (status) output area's interlock flag.
      move.b   (0x01,a2),d2                    **      Read the I/O status flags.
      andi     #0x08,d2                        **      Check the output area's interlock flag.
      beq      write_io_data_02                **      Go to next process if the flag = 0.
      move.w   #-1,d0                          **      If the flag = 1, set the return code to -1 and
      jmp      write_io_data_end               **      go to end process.

write_io_data_02:                              ** Set the (control) output area's interlock bit.
      move.b   (a3),d2                         **      Read the interlock bit.
      ori      #0x08,d2                        **      Turn the output area's interlock bit ON.
      move.b   d2,(a3)                         **      Set the interlock bit.

write_io_data_03:                              ** Check the (status) output area's interlock flag.
      move.b   (0x01,a2),d2                    **      Read the I/O status flags.
      andi     #0x08,d2                        **      Check the output area's interlock flag.
      beq      write_io_data_04                **      Go to next process if flag = 0.
                                               **      If flag = 1, the Board requested access too, so
      move.b   (a3),d2                         ** clear the (control) output area's interlock bit.
      andi     #0xf7,d2                        **      Turn the output area's interlock bit OFF.
      move.b   d2,(a3)                         **      Clear the interlock bit.
      jmp      write_io_data_01                ** Start over from the beginning of the process.
```

```
write_io_data_04:                                ** Write output data.
      move.b   (a0)+,d2                          **      Read data from write source (specified area).
      move.b   d2,(a1)+                          **      Write data to write destination (memory pool).
      subi.l   #1,d0                             **      Decrement D0 (number of bytes to write).
      bne      write_io_data_04                  **      Repeat until the specified amount is written.

write_io_data_05:                                ** Clear the (control) output area's interlock bit.
      move.b   (a3),d2                           **      Read the interlock bit.
      andi     #0xf7,d2                          **      Turn the output area's interlock bit OFF.
      move.b   d2,(a3)                           **      Clear the interlock bit.

      move.w   #0,d0                             ** Normal completion (Return code set to 0.)

write_io_data_end:                               ** End process
      rts                                        **      Return
*********************************************************************************************
* Error Processing
*********************************************************************************************

Initial_Error:                                   ** Firmware startup error
      bra      Initial_Error                     **      (Infinite loop)

Send_Command_Error:                              ** Command execution error
      bra      Send_Command_Error                **      (Infinite loop)


*********************************************************************************************
* Command Table
*********************************************************************************************
      .data
      .even

DN_Online_Table:                                 ** Command table for DN_ONLINE
      .byte    0x01                              **      DN_ONLINE Command Code              (Low Byte)
      .byte    0                                 **                                          (High Byte)
      .byte    63                                **      Master node address (63)            (Low Byte)
      .byte    0                                 **                                          (High Byte)
      .byte    0x02                              **      Baud rate (2=500 K baud)            (Low Byte)
      .byte    0                                 **                                          (High Byte)
      .byte    0                                 **      Communications cycle time (0=auto)  (Low Byte)
      .byte    0                                 **                                          (High Byte)
      .word    0xffff                            **      Table end identifier

Add_Device_Table_DRT21:                          ** Command table for ADD_DEVICE (CQM1-DRT21)
      .byte    0x03                              **      ADD_DEVICE Command Code             (Low Byte)
      .byte    0                                 **                                          (High Byte)
      .byte    1                                 **      Slave node address (1)              (Low Byte)
      .byte    0                                 **                                          (High Byte)
      .byte    47                                **      Vendor ID (47=OMRON)                (Low Byte)
      .byte    0                                 **                                          (High Byte)
      .byte    0                                 **      Device type (0=auto)                (Low Byte)
      .byte    0                                 **                                          (High Byte)
      .byte    0                                 **      Product code (0=auto)               (Low Byte)
      .byte    0                                 **                                          (High Byte)
      .word    0                                 **      Reserved area
      .word    0                              **     Reserved area
      .word    0                              **     Reserved area
      .byte    2                              **     Communication type (2=Poll)          (Low Byte)
      .byte    0                              **                                          (High Byte)
      .word    0                              **     Reserved area
      .word    0                              **     Reserved area
      .word    0                              **     Reserved area
      .byte    2                              **     Output size (2 bytes)                (Low Byte)
      .byte    0                              **                                          (High Byte)
      .byte    0x00                           **     Output area offset (0x1000)          (Low Byte)
      .byte    0x10                           **                                          (High Byte)
      .word    0                              **     Reserved area
      .word    0                              **     Reserved area
      .byte    2                              **     Input size (2 bytes)                 (Low Byte)
      .byte    0                              **                                          (High Byte)
      .byte    0x00                           **     Output area offset (0x2000)          (Low Byte)
      .byte    0x20                           **                                          (High Byte)
      .word    0xffff                         **     Table end identifier
Start_Scan_Table:                             ** Command table for START_SCAN
      .byte    0x06                           **     START_SCAN Command Code              (Low Byte)
      .byte    0                              **                                          (High Byte)
      .word    0xffff                         **     Table end identifier

      .end
```

SECTION 8
Error Processing

This section provides information on errors that can occur during VME Board operation.

8-1 LED Indicators

There are two LED indicators on the CompoBus/D VME Board that can indicate when an error has occurred and the general nature of the error. The HEALTH indicator shows the status of the Board and the COMM indicator shows the status of the network. The meaning of these indicators is described below.

Note The Board's HEALTH indicator is equivalent to a Master Unit's MS (module status) indicator and its COMM indicator is equivalent to a Master Unit's NS (network status) indicator. Refer to the *CompoBus/D (DeviceNet) Operation Manual* for more details on LED indicators.

The HEALTH and COMM indicators can be red or green. The following table shows the meaning of the possible indicator combinations.

Name	Color	Status		Meaning
HEALTH	Green	ON	Normal status	Communications operating normally.
	Red	ON	Fatal error	A non-recoverable fatal error such as a watchdog timer error, memory error, or system error occurred. Replace the Board if the error recurs. The HLTH bit in the setting/status register isn't set to 1.
	–	OFF	No power supply	Power isn't being supplied or the Board is resetting.
COMM	Green	ON	Online/ communications established	The Board is online and remote I/O communications have been established with the Slaves registered in the scan list.
		Flashing	Online/ communications not established	The Board is online, but remote I/O communications haven't been established (reading scan list or communications were stopped).
	Red	ON	Fatal communications error	Communications not possible (node address duplication or Bus Off error detected)
		Flashing	Non-fatal communications error	Communications timeout
	–	OFF	Offline/ Power supply off	The Board is not online.

8-2 Errors Indicated in Memory

8-2-1 Errors Indicated in the Board Information Area

When an error has occurred in the Board's firmware (application in the Board), the application module header's module type (address 0000) is set to "ER" (4552 hexadecimal) and details about the error are stored in the board information area (addresses 0040 to 007F).

Number	Error message	Error details and processing
Error 1	RAM data test failed	An error occurred while testing the RAM's data bus. Replace the Board.
Error 2	RAM address test failed	An error occurred while testing the RAM's address bus. Replace the Board.
Error 3	RAM A16 address test failed	An error occurred while testing RAM signal A16. Replace the Board.
Error 4	RAM A17 address test failed	An error occurred while testing RAM signal A17. Replace the Board.
Error 5	Module checksum is invalid	A checksum error occurred in the firmware. This may indicate memory damage. Replace the Board if the error recurs.
Error 6	CAN reset flag failed to clear	An error occurred while testing the CAN controller. Replace the Board.
Error 7	CAN data test failed	An error occurred while testing the CAN controller's data bus. Replace the Board.
Error 8	CAN address test failed	An error occurred while testing the CAN controller's address bus. Replace the Board.
Error 9	Invalid NVRAM data	There was an error in the Board's non-volatile memory data. Replace the Board.
Error 10	Execution permission denied	This is a hardware error. Replace the Board.
Error 11	Application initialization error	An error occurred during firmware initialization. Replace the Board.
Error 12	Unknown application initialization code	An error occurred during firmware initialization. Replace the Board.
Error 13	Application terminated	Firmware ended with an error termination. Replace the Board.
Error 14	Application fatal error	Firmware ended with an error termination. Replace the Board.
Errors 15 to 21	XXX interrupt	An unexpected interrupt was detected. Replace the Board.
Error 22	Event queue overflow	An error occurred in the firmware. Replace the Board.
Error 23	Nested user timer interrupt	An error occurred in the firmware. Replace the Board.
Error 24	Invalid CAN interrupt	An error occurred in the firmware. Replace the Board.
Error 25	Nested system timer interrupt	An error occurred in the firmware. Replace the Board.
Error 26	Imperfect interrupt	There was a problem in interrupt processing, such as CINT being cleared during the Board's interrupt processing. Check the way that interrupts are generated from the VME Host.
Error 99	Unexpected condition encountered	An unexpected error occurred. Replace the Board.

8-2-2 Errors Indicated in the Application Module Header

The following table shows the errors that are indicated in the application module header. In general, these are errors in the Board itself or errors during communications with the network. (Offset addresses are hexadecimal.)

Offset address	Bit	Name	Status	Error details and processing
0030	2	BO	1	A Bus Off error occurred (communications stopped because of multiple data errors). Check the nodes' baud rate settings, communications cables, and ground and then restart the Board.
	4	TA	1	There isn't even one Slave. Check the contents of the scan list and the status of the Slaves.
	5	TO	1	A communications timeout occurred. The load on the network is too high, so divide the network if another Master is available.
0031	1	BP	0	Communications power isn't being supplied from the communications connector. Inspect the communications power supply and power cable.

In addition to the error flags above, the Board has the following counters which show the number of messages that have been transferred and the number of frame errors that have occurred. These counters can be used for troubleshooting.

Offset address	Name	Data type	Contents
0032	CAN transmission counter	Word	Number of times that messages were transmitted to the network.
0036	CAN reception counter	Word	Number of times that messages were received from the network.
0038	CAN error counter	Word	Number of times that frame errors occurred.

8-2-3 Errors Indicated in the Device Status Table

The first byte allocated to each Slave in the device status table contains a hexadecimal code which provides information on that Slave's errors.

Status code	Meanings and processing
00	Device not in scan list. The Slave isn't registered in the scan list. Use the ADD_DEVICE command to register the Slave in the scan list if the Slave is going to participate in remote I/O communications.
01	Device idle (not being scanned). Remote I/O communications stopped.
02	Device being scanned. Remote I/O communications operating.
03	Device timed-out
04	UCMM connection error
05	Master/Slave connection set is busy
06	Error allocating Master/Slave connection set
07	Invalid vendor id. There is a mistake in the vendor ID setting. Get the correct vendor ID from the vendor and change the setting.
08	Error reading vendor id
09	Invalid device type. There is a mistake in the device type setting. Get the correct device type from the vendor and change the setting.
0A	Error reading device type
0B	Invalid product code. There is a mistake in the product code setting. Get the correct product code from the vendor and change the setting.
0C	Error reading product code
0D	Invalid input size. There is a mistake in the input size setting. Contact the vendor to get the number of input bytes supported by the Slave and change the setting.
0E	Error reading input size
0F	Invalid output size. There is a mistake in the output size setting. Contact the vendor to get the number of output bytes supported by the Slave and change the setting.
10	Error reading output size
11	Reserved for the system.
12	Reserved for the system.
13	Reserved for the system.
14	Reserved for the system.
15	Remote I/O connection packet rate error
16	Reserved for the system.
17	Master/Slave connection set sync fault
18 to FF	Reserved for the system.

8-2-4 Errors Indicated in the Event Notification Queue

A queue overrun error will occur if events are not read from the VME Host and there are more than 509 unread events in the queue.
The queue overrun errors are indicated in queue status A/B as follows:

Name	Offset address	Bit	Function
O	Queue status A: 09C0 Queue status B: 09C1 (in the host interface block)	Bit 0	0: A queue overrun error didn't occur. 1: A queue overrun error occurred.

When a queue overrun error has occurred, read and process the stored events in interrupt processing and then clear queue status A and queue status B. Refer to *6-9 Processing Interrupts* for details on reading events in the event queue and related processing.

Appendix A
Summary of Memory Functions

Short I/O Address Space

Offset address	Name	Bit	Name	Function
0	Setting/status register	0	CINT	Command execution begins when this bit goes ON.
		2	IRQ	Shows interrupt status from the Board, clears interrupts.
		3	IRQE	Enables hardware interrupts from the Board.
		4	HLTH	Controls the HEALTH indicator.
		5	WDI	Controls the watchdog timer.
		6	SYS	Controls the FAIL and PASS indicators.
		7	RUN	Shows operating status of the Board's firmware, starts the firmware.
3	Interrupt register	0 to 7	–	Sets the interrupt ID from the Board.
5	Memory control register	0	MEN	Enables access from the VME Host to shared RAM.
		1 to 7	A17 to A23	Sets the base address of shared RAM.

Standard Address Space

Application Module Header

Offset address	Name	Bit	Name	Function
0000 0001	Module type	0 to 7	Module Type	Contains the Board's status code.
0002 0003	Shared RAM size	0 to 7	WinSize	Sets the size of shared RAM. (Fixed at 0003.)
002A	Interrupt control	0	CM	Enables the command completion interrupt.
		1	BS	Enables the network status change interrupt.
		2	QE	Enables the event notification interrupt.
002C	Interrupt status A	0	CM	Shows the status of the command completion interrupt.
		1	BS	Shows the status of the network status change interrupt.
		2	QE	Shows the status of the command completion interrupt.
002D	Interrupt status B	0	CM	Shows the status of the command completion interrupt.
		1	BS	Shows the status of the network status change interrupt.
		2	QE	Shows the status of the command completion interrupt.
0030	Network status	0	OL	Communications interface initialization completed flag
		2	BO	Bus off detected flag
		4	TA	No Slaves flag
		5	TO	Communications timeout flag
0031		1	BP	Network power supply on flag
		4	O1	Baud rate 125,000 flag
		5	O2	Baud rate 250,000 flag
		6	O5	Baud rate 500,000 flag
		7	SA	Remote I/O communications operating flag
0032 0033	CAN transmission counter	0 to 7	–	Number of messages transmitted from the Board
0036 0037	CAN reception counter	0 to 7	–	Number of messages received by the Board

Offset address	Name	Bit	Name	Function
0038 0039	CAN error counter	0 to 7	–	Number of frame errors detected by the Board
0040 to 007F	Board information	0 to 7	–	Contains error message.

Command Buffer

Offset address	Name	Bit	Name	Function
0080 to 013F	Command buffer	0 to 7	–	Used to exchange commands from the VME Host to the Board and responses from the Board to the VME Host.

Client Status Block

Offset address	Name	Bit	Name	Function
0140	Status code	0 to 7	–	Operating status of remote I/O communications
0142	Remote I/O communications event	0	S	Remote I/O communications scan start flag

Client Control Block

Offset address	Name	Bit	Name	Function
015E	Event queue control	0	C	Enables storage of status code (0140) changes in the event queue.
		2	S	Enables storage of remote I/O communications events (0142) in the event queue.

Device Control Event Table

Offset address	Name	Bit	Name	Function
0180	–	0 to 7	–	Notification of control (output) to node address 0
:	:	:	:	:
01BF	–	0 to 7	–	Notification of control (output) to node address 63

Device Status Table

Offset address	Name	Bit	Name	Function
01C0 to 01CF	–	0 to 7	–	Device status information for node address 0 (See below.)
:	:	:	:	:
05B0 to 05BF	–	0 to 7	–	Device status information for node address 63 (See below.)

The following table shows the device status information provided for each Slave.

Offset address	Name	Bit	Name	Function
+0	Status code	0 to 7	–	Shows the Slave's status.
+1	Status flags	1	I1	Input interlock from the Board
		3	O1	Output interlock from the Board
+2	I/O event flag	0	U	Input data received from Slave flag

Device Control Table

Offset address	Name	Bit	Name	Function
05C0 to 05CF	–	0 to 7	–	Controls node address 0. (See below.)
:	:	:	:	:
09B0 to 09BF	–	0 to 7	–	Controls node address 63. (See below.)

The following table shows the control bits provided for each Slave.

Offset address	Name	Bit	Name	Function
+0	Control bits	1	I1	Input interlock from the VME Host
		3	O1	Output interlock from the VME Host
+1	I/O event flag	0	C	Write output data to Slave flag
+14	Event queue control bits	0	C	Enables storage of Slave status code (+0) changes in the queue.
		2	I1	Enables storage of input data reception from Slaves (+2) in the queue.

Event Notification Queue

Offset address	Name	Bit	Name	Function
09C0	Queue status A	0	O	Queue overrun occurred flag
09C1	Queue status B	0	O	Queue overrun occurred flag
09C2 09C3	Queue IN	0 to 7	–	Contains the address where the next event will be stored. (Written from the Board.)
09C4 09C5	Queue OUT	0 to 7	–	Contains the address where the next event will be read. (Written from the VME Host.)
09C6 to 0DBF	Event queue	0 to 7	–	The event information is stored in this area. (See below.)

Event Information

Offset address	Name	Bit	Name	Function
+0	Event source code	0 to 7	–	Shows where the event occurred.
+1	Event ID	0 to 7	–	Shows what kind of event occurred.

Free Memory

Offset address	Name	Bit	Name	Function
1000 to 3FFF	Free memory	0 to 7	–	Used as an I/O area for I/O transfers with the Slaves.

Index

Revision History

A manual revision code appears as a suffix to the catalog number on the front cover of the manual.

Cat. No. W327-E1-1

Revision code

The following table outlines the changes made to the manual during each revision. Page numbers refer to the previous version.

Revision code	Date	Revised content
1	June 1997	Original production