# Group3 Control
## Loop Controller
## Programmer's Manual

Device Interface Software V5.0i
CNA Module Software V5.0g
F Board Software V5.0c
Loop Controller Software V5.1

05 August 2008

Group3 Technology Ltd., 2 Charann Place, Avondale, Auckland 1026. P.O. Box 71-111, Rosebank, Auckland 1348, New Zealand.
Phone: +64 9 828 3358 Fax: +64 9 828 3357 email: info@group3technology.com web: www.group3technology.com

Thank you for purchasing and using Group3 Control equipment. We hope you will join the growing number of people who are enthusiastic about the features Group3 Control has to offer.

Group3 has been designing and building specialised control systems for over fourteen years. We are constantly upgrading and improving our products and the supporting documentation. We welcome input from our customers, so if there are aspects of the system you particularly like, or things you would like to see implemented, improved or developed in the way of hardware, software or documentation please contact your Group3 representative, or Group3 directly with your suggestions.

The Group3 web-site ( http://www.group3technology.com ) contains some pages of information on Group3 products. This site will be regularly updated, and you may choose to check it from time to time, to learn about recent developments.

Listed on the "What's New" page are the recent developments and enhancements to the product range, grouped by the year of introduction.

The page "User Technical Notes" lists current versions of software, and also details past versions, with a brief list of the features they introduced.

Group3 Technology Ltd.
2 Charann Place, Avondale, Auckland 1026
PO Box 71-111, Rosebank, Auckland 1348, New Zealand.
Phone: +64 9 828 3358
Fax: +64 9 828 3357
Email: info@group3technology.com
Web: http://www.group3technology.com

# CONTENTS

# 1 Introduction

This manual documents the specific requirements for creating a program to interface to and operate Group3 Loop Controllers from a computer. The Loop Controller is the computer's access point into a control system implemented with Group3 Control hardware. Three software interfaces have already been developed by Group3 for customers to use with their applications: for WonderWare InTouch (via a DDE server), LabVIEW and C++. Using this manual, a programmer can create their own interface using any other computer programming language. Several Group3 Control customers have done this.

The Loop Controller card fits into a computer and is an integral part of the Group3 Control product range. On each Loop Controller (LC) card can be one or more Loop Controllers, each with their own processor, dualport RAM and fiber optic connectors. The Loop Controller processor handles all communications with the remote hardware, which can be a loop of Group3 Device Interfaces (DIs), a loop of Group3 Teslameters (DTMs), another Loop Controller or any other devices that can communicate serially. Therefore the computer is not directly involved in the communication. With the exception of DIs operating back to back (DI to DI mode) all Group3 Control applications require the use of a Loop Controller.

Each Loop Controller has a block of dualport RAM (shared memory) on board. The dualport RAM is shared between the Loop Controller processor and the application program running on the computer. The computer and the Loop Controller processor exchange data through this memory. In fact the only access the computer has to the Loop Controller is via the this dualport RAM. For example, to send a new output (control) value to an I/O board in a DI, the computer simply writes the new value into the appropriate area of the dualport RAM. The processor on the LC card automatically uplifts this new value and sends it out on the fiber optic loop to the DI. The Loop Controller also continually interrogates the DIs on its loop and updates the dualport RAM with the new input (status) values. The computer just has to read the dualport RAM to get the latest input values read back from a DI.

Therefore to create an interface to a Group3 Loop Controller is to create an interface to its dualport RAM.

The Loop Controller has three main modes of operation: LC to DI (modes 0 and 7), LC to LC (mode 4) and Serial (mode 1). These are described in detail in chapters 4, 5 and 6 respectively. Mode 1 has the further configurable division of either General Serial mode or Group3 DTM Loop mode.

| **Communication Mode** | | **Value** |
|---|---|---|
| LC to DI Loop | SDLC at 1.152Mbaud | 0 |
| Serial comm's (General/DTM loop) | selectable baud rate | 1 |
| LC to LC comm's. | SDLC at 1.152Mbaud | 4 |
| Fast LC to DI Loop (multi-boards /message) | SDLC at 1.152Mbaud | 7 |

The missing values from this table (2, 3, 5 and 6) are either redundant or reserved modes of communication.

## The Jargon

The following list is to explain the meaning of various specific terminology used throughout this manual:

| | |
|---|---|
| Group3 Control | The name of the control system manufactured by Group3 Technology. This includes the various types of Loop Controllers, DI processor and I/O boards as well as control software and memory drivers. |
| Loop Controller card | The Group3 interface card that is mounted in a computer. Depending on the computer bus it may contain one or more Loop Controllers. |
| LC card | Abbreviation for Loop Controller card. |
| Loop Controller | One independent interface (processor, dualport RAM and fiber optic connectors) located on a Loop Controller card. Each Loop Controller handles all the communications for one loop of remote hardware. It is configured by the application program running on the computer. |
| LC | Abbreviation for Loop Controller. |
| Loop | One or more Device Interfaces connected in daisy-chain fashion to a Loop Controller with fiber optic cables. It can also be one or more Group3 DTMs connected similarly to either a Loop Controller or a port on a type F board. |
| Dualport RAM | A block of memory shared between the Loop Controller processor and an application program running on the computer. There is one dualport RAM with each Loop Controller processor on a Loop Controller card. |
| DP | Abbreviation for Dualport RAM. |
| Device Interface | A control unit containing a processor board and up to three I/O boards. |
| DI | Abbreviation for Device Interface. |
| SDLC | Acronym for “Synchronous Data Link Control”. This is a robust industry standard link-level communications protocol. |
| ISA bus | Old PC expansion bus. No longer incorporated in new PCs. |
| PCI bus | New PC standard expansion bus. Supports plug-and-play. |
| VME bus | An industry standard computer bus. Fast and flexible. |
| CAMAC bus | A specialised computer bus favoured for scientific use. |
| STD bus | Another industrial computer bus. |
| control data | Output data (analogs, digitals etc) sent to an I/O board. |
| status data | Input data (analogs, digitals etc) returned from an I/O board. |
| Teslameter | Group3 magnetic field strength measuring instrument. |
| DTM | Same as Teslameter (Digital Teslameter). |

# 2 Loop Controllers - Hardware Description

There are a number of LCs adapted to different computers. The dualport RAM structure makes it relatively easy for Group3 to adapt the card to other computer architectures. Consult your Group3 Control representative for the latest list of LCs available.

There are two Loop Controllers designed for a PCI bus slot on a PC computer:
**LC1-PCI** controls a single loop.
**LC3-PCI** controls three independent loops from a single bus slot.

Similarly there are two Loop Controllers that operate from an ISA computer bus slot:
**LC1-ISA** controls a single loop.
**LC3-ISA** controls three independent loops from a single bus slot.

**LC2-VME** is a VME card that can control two loops from a VME crate.

**LC-STD** controls a single loop from an STD bus crate.

**LC1-CAM** controls a single loop from a CAMAC crate.
**LC2-CAM** controls two loops from a single slot in a CAMAC crate.

Note that on the multi-loop LCs (LC3-PCI, LC3-ISA, LC2-VME, and LC2-CAM) the different loops are completely independent - one loop could be used for talking to another LC, and the others could be set to control Group3 DIs (the most common type of control installation).

Plastic fiber cable connectors are supplied as standard, but ST or SMA glass fiber connectors can be fitted to order. Maximum fiber optic cable length:

| | |
|---|---|
| plastic cable | 40m |
| glass cable | 3000m |

## 2.1 PCI Bus Loop Controllers: LC1-PCI & LC3-PCI

The LC1-PCI and LC3-PCI are Loop Controller cards designed to operate Group3 Control modules from a PCI bus slot. The LC1 drives a single fiber optic communication loop, while the LC3 allows three completely independent loops to be controlled from one slot. A maximum of 16 PCI LCs can be controlled from any one computer. For example 5 LC3's and 1 LC1 gives the maximum of 16 Loop Controllers. LC1 and LC3 cards can be mixed in a system.

The PCI Loop Controllers are Plug and Play devices. The computer BIOS allocates the memory addresses of their dualport RAMs on start up and the user does not need to be involved in that process. These cards require a system driver to be installed on the computer to provide application programs with an access address to the dualport RAM of each Loop Controller upon request. This driver software is supplied on a disk with each Loop Controller card and instructions for installing the driver are available as a text file, *install.txt*, on that disk.

The only hardware setting the user needs to be aware of is the rotary switch on the top edge of the circuit board. This is shipped from the factory set at position '0', and there is only need to alter this setting if two or more PCI Loop Controller cards are to be installed in the same computer. The switch is there to allow the user and the application software to distinguish between the different Loop Controllers in the same computer.

- The switch on the LC1-PCI sets the loop number for that card.
- The switch on the LC3-PCI sets the loop number (call it **n**) of the top transmitter/receiver pair - the one furthest from the gold edge fingers. The number of the middle loop is one more than the switch setting (**n**+1), and the number of the bottom loop is two more than the switch setting (**n**+2).

If using several Loop Controller cards in one computer remember that the switch settings must be different, and that the three loops of an LC3-PCI occupy three consecutive numbers.

### Specifications

**Physical**

LC1-PCI 100 x 120mm
LC3-PCI 100 x 290mm

Dimensions do not include the gold-plated edge connector.
A standard metal mounting plate is fitted to one edge of the card.

Fiber optic connectors (transmit and receive for each loop) for H-P plastic fiber cable are accessible through the mounting plate. The transmit connectors are gray, and the receive connectors are blue. ST or SMA glass fiber connectors can be factory fitted to special order.

On the LC3-PCI the port with the lowest address is furthest from the edge connector.

**PCI Bus Interface**

The whole of dualport RAM appears as memory mapped I/O - controlled outputs are set and inputs monitored solely by writing and reading from memory locations.

Dualport RAM capacity 2048 bytes per Loop Controller.

## Interrupts

PCI Loop Controllers can use interrupts to notify the host computer that new data has arrived. Interrupts can be enabled or disabled as required simply by setting or clearing bit-1 in the Communications Enabled byte of the dualport RAM system area. If enabled, an interrupt will be generated by the Loop Controller if:

**LC to DI Mode**

- New data from an I/O board has arrived in over the loop, and it has different values from the data already present in the dualport for that board.
- Data has been received for a type F board, type K board, or over-the-loop diagnostic port.
- Data for a type F board, type K board, or diagnostic port has been sent.

**LC to LC Mode**

- A block of data has come in and has been placed in a Receive Data Area.

**Serial: General Serial Mode**

- Data has been placed in the Receive Buffer.
- Data in the Send Buffer has all been sent.

**Serial: DTM Loop Mode**

- Data (field, temperature and error) has been stored for a DTM.
- An error code has been stored for a DTM.

See sections 4.7, 5.5 and 6.3 for specific information about how the interrupts are used on the PCI Loop Controller as well as the note in section 3.1.

## 2.2 ISA Bus Loop Controllers: LC1-ISA & LC3-ISA

The LC1-ISA and LC3-ISA are Loop Controller cards designed to operate Group3 Control modules from an ISA bus slot. The LC1-ISA drives a single fiber optic communication loop, while the LC3-ISA allows three completely independent loops to be controlled from one slot. The only limit to the number of ISA Loop Controllers used in a computer is the number of available ISA slots. LC1 and LC3 cards can be mixed in a system.

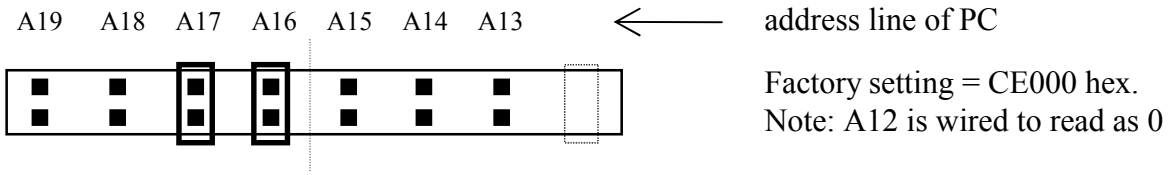
Communication between the computer and each LC takes place through dualport RAM installed on the LC card. This dualport RAM is accessed directly from the computer's address bus. Jumpers are provided on the LCs to place the dualport RAM at a suitable position within the computer's address range.

Operating these cards in a 32-bit environment requires a system driver to be installed on the computer to provide application programs with an access address to the dualport RAM of each Loop Controller upon request. This driver software is supplied on a disk with each Loop Controller card and instructions for installing the driver are available as a text file, *install.txt*, on that disk..

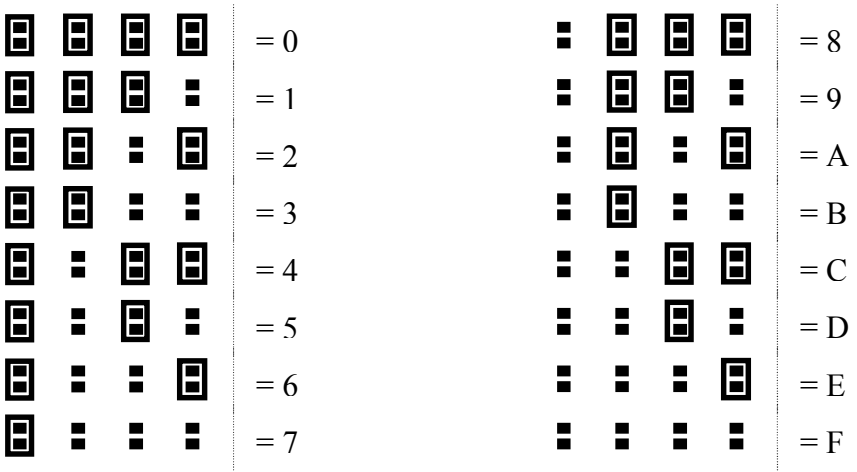
On an ISA Loop Controller the dualport RAM is intended to be placed in the area between 640k and 1M bytes, where free address space can usually be found. Conflict with other devices in the system (such as shadow RAM) must be avoided. The LC base address is determined by comparing PC address lines A13 to A19 with the jumper settings. The base address of the dualport RAM can be placed anywhere between 0 hex and FE000 hex in increments of 2000 hex (8192 bytes).

The absence of a jumper on a particular address line corresponds to a 1.
The presence of a jumper corresponds to a 0.

Dual row pin header P2 on Loop Controller card




Hexadecimal coding table: ▪ = no jumper = 1 ▣ = jumper in place = 0



- The address jumpers on an LC1-ISA set the base address for the dualport RAM on that card.
- The address jumpers on the LC3-ISA set the base address of the top transmitter/receiver pair - the one furthest from the gold card edge fingers. The base address of the middle loop is 2048 bytes (800 hex) more than the address jumpers, and the base address of the bottom loop is 4096 bytes (1000 hex) more than the address jumpers. With the jumpers set for CE, as per the factory setting, the base addresses of the three loops are CE000, CE800, and CF000.

**Note:** Care must be taken if mixing the current ISA bus LC's with the earlier versions (pre 1993). These LC cards have 1Kbyte dualport RAMs (boards labeled 16000032A for LC1s, 16000048 for LC3s). They also have an eighth jumper to set A12 as well. If mixing old and new versions check that the address jumpers are set so that the memory areas used by the boards are different - it is not sufficient that the jumpers just appear to be set to physically different pins.

## Specifications

### Physical

LC1-ISA 100 x 115mm
LC3-ISA 100 x 300mm

Dimensions do not include the gold-plated edge connector.
A standard metal mounting plate is fitted to the one edge of the card.

Fiber optic connectors (transmit and receive for each loop) for H-P plastic fiber cable are accessible through the mounting plate. The transmit connectors are gray, and the receive connectors are blue. ST or SMA glass fiber connectors can be factory fitted to special order.

On the LC3-ISA the port with the lowest address is furthest from the edge connector.

### ISA Bus Interface

The whole of dualport RAM appears as memory mapped I/O - controlled outputs are set and inputs monitored solely by writing and reading from memory locations.

| | |
|---|---|
| Dualport RAM capacity | 2048 bytes per Loop Controller. |
| Base address | Jumper selected from $00000_h$ to $FE000_h$ in increments of $2000_h$ (8192 bytes)<br>Factory setting $CE000_h$<br>LC3 loops are spaced at increments of $800_h$ |

### Interrupts

This board does not generate interrupt signals.

## 2.3 VME Bus Loop Controllers: LC2-VME

The LC2-VME is a Loop Controller card designed to operate Group3 Control modules from a VME bus crate. Two completely independent loops are controlled from the one LC2-VME card. Up to eight LC2-VME cards can be installed in one VME crate. Communication between the computer and each LC takes place through dualport RAM installed on the LC card.

### Specifications

**Physical**

Standard double-height VME board (160 x 233mm), with a 20mm wide front panel.

VME backplane connector J1/P1 only required.

Fiber optic connectors (transmit and receive for each loop) for H-P plastic fiber cable are accessible through the front panel. The transmit connectors are gray, and the receive connectors are blue. ST or SMA glass fiber connectors can be factory fitted to special order.

Indicator LEDs for transmit and receive data (each loop) are on the front panel.

**VME Bus Interface**

The whole of dualport RAM appears as memory mapped I/O - controlled outputs are set and inputs monitored solely by writing and reading from memory locations.

Dualport RAM capacity 2048 bytes per Loop Controller.

The LC2-VME uses the VME Short Addressing scheme: 8-bit data and 16-bit address. This means that sequential locations in the LC2-VME dualport RAMs are addressed as alternate (odd) bytes over the VME bus. For this reason, although each loop has 2k bytes of dualport RAM, it occupies 4k of VME address space, and each LC2-VME card with two loops occupies 8k of the 64k short address space. Jumpers are provided to position this 8k block within the 64k short address range.

The LC2-VME responds to Address Modifier codes of 29 and 2D, as standard. Other address modifiers can be implemented to special order.

The board uses the standard VME 16 MHz SYSCLK signal to generate the delays required before issuing DTACK. Four rising edges of SYSCLK are required to complete a data transfer cycle, so access cycle times are between 248ns and 310ns.

**Interrupts**

This board does not generate interrupt signals.

## 2.4 STD Bus Loop Controllers: LC-STD

The LC-STD is a Loop Controller card designed to operate Group3 Control modules from an STD bus crate. A single LC-STD card controls one loop. The number of LC-STD cards that can be installed in the STD crate is only limited by the number of available slots in the crate. Communication between the computer and the LC takes place through dualport RAM installed on the LC card.

### Specifications

**Physical**

Standard STD board (approx. 165 x 115mm), with gold plated edge connector to suit STD bus backplane.

Fiber optic connectors (transmit and receive) for H-P plastic fiber cable. The transmit connector is gray, and the receive connector is blue. ST or SMA glass fiber connectors can be factory fitted to special order.

**STD Bus Interface**

The whole of dualport RAM appears as memory mapped I/O - controlled outputs are set and inputs monitored solely by writing and reading from memory locations.

Dualport RAM capacity 2048 bytes per Loop Controller.

The standard LC-STD uses 8-bit data and 16-bit address lines.

The dualport RAM can be positioned in either I/O space or Memory space by placing a jumper to allow access by either the IORQ or the MEMRQ signal. The 2 Kbyte block of dualport RAM can be set at a base address in either of these 64K (16bit) address spaces by setting jumpers on the base address jumper pins.

The LC-STD can also support the 24-bit address scheme of the STD bus specification, by taking the upper 8-bits (A16-A23) from the multiplexed address/data bus. This option requires some additional components to be added to the 16-bit address board.

The board will issue a WAITRQ if both the STD processor and the LC-STD processor try to access the same dualport RAM location simultaneously.

STD signals SYSRESET and PBRESET will reset the processor on the LC-STD.

**Interrupts**

The hardware is in place to issue interrupt (INTRQ, INTRQ1, or INTRQ2, selectable by jumper) signals to the STD processor. The Loop Controller software now supports interrupts but a small hardware modification is required to LC-STD cards already in use before this feature can be used. In the meantime these jumpers should be left off. See the *Interrupts* section for the PCI Loop Controller in this chapter.

## 2.5 CAMAC Loop Controllers: LC1-CAM & LC2-CAM

The LC1-CAM and LC2-CAM are Loop Controller cards designed to operate Group3 Control modules from a CAMAC crate. The LC1-CAM drives a single fiber optic communication loop, while the LC2-CAM allows two completely independent loops to be controlled from one slot. The only limit to the number of CAMAC Loop Controllers that can be used in a CAMAC crate is the number of available slots. Communication between the computer and each LC takes place through dualport RAM installed on the LC card.

### Specifications

#### Physical

Standard single slot width CAMAC module (approx. 310 x 200 x 17mm), with gold plated edge connector to suit CAMAC backplane.

Fiber optic connectors (transmit and receive for each loop) for H-P plastic fiber cable are on the front panel of the module. The transmit connectors are gray, and the receive connectors are blue. ST or SMA glass fiber connectors can be factory fitted to special order.

LEDs on the front panel indicate data on the fiber optic cables and another LED lights to indicate **X** - "CAMAC command accepted".

#### CAMAC Interface

The whole of dualport RAM appears as memory mapped I/O - controlled outputs are set and inputs monitored solely by writing and reading from memory locations.

Dualport RAM capacity 2048 bytes per Loop Controller.

There is an address register on the CAMAC Loop Controller which can be written to from the CAMAC write lines. The address written into this register remains there until written over or power is removed. This address register holds the address lines used to access a particular location in either of the dualport RAMs.

To write an address to this register the user must perform a write operation - CAMAC function F(16).A(2) writes to the address register, from CAMAC lines W01-W11. Note that only the lower 11 bits are used.

All data written into or read out from the dualport RAMs is transferred eight bits (one byte) at a time.

Having used F(16).A(2).N( ) to write the address to the address register, the following CAMAC commands are available.

F(0).A(0).N( ) reads one byte from this address in the DP RAM of loop 0, onto R1-R8.
F(16).A(0).N( ) writes one byte to this address in the DP RAM of loop 0, from W1-W8.

F(0).A(1).N( ) reads one byte from this address in the DP RAM of loop 1, onto R1-R8.
F(16).A(1).N( ) writes one byte to this address in the DP RAM of loop 1, onto W1-W8.

The user must set up the data areas of the dualport RAM as described in sections 4, 5 and 6 of this manual. From then on:

- To set an output channel the user writes the address of that channel to the address register, then writes the desired value to that address.
- To read an input channel the user writes the address of that channel to the address register, then reads the data from that address.

**Interrupts**

This board does not generate interrupt signals.

# 3 Memory Access Drivers & Software Interfaces

## 3.1 Introduction

The Group3 Control system Loop Controller exchanges I/O data with DIs distributed around a fiber optic loop, another Loop Controller, a loop of Group3 Teslameters or other remote serial instruments. It sends output data to the remote hardware which it takes from dualport RAM onboard the Loop Controller card. It also places received data in dualport RAM. All of the data in dualport RAM is stored at defined locations (as detailed in chapters 4, 5 and 6 of this manual). This dualport RAM is also able to be accessed by the host computer bus, so that the data can be stored and read out by the application program, so direct memory accesses by the computer are required. However it is usual, particularly with the later PC operating systems, to require the use of system drivers to gain access to specified regions of physical memory (see section 3.2). Such a region might encompass the dualport RAM of one or more ISA Loop Controller. Accesses to memory outside of the system authorised regions will generate operating system errors.

Once access is gained to the dualport RAM of a Loop Controller, it is up to the application program to perform all the necessary handshaking and arbitration required to ensure that data is read and written without corruption. These protocols are described in detail in chapters 4, 5 and 6 of this manual. Group3 has developed three high level software interfaces which perform all these protocols, thus freeing the programmer to concentrate on the specifics of their application and graphical interface.

The PCI Loop Controllers allow the option of generating interrupts to the host computer, so that retrieving information from the dualport can become an interrupt driven process. If enabled, an interrupt will occur when new data from an I/O board or other remote hardware arrives from the loop and is stored in dualport RAM, or if all the serial data in a transmit buffer has been sent out over the loop. Interrupts are only of relevance to users writing their own software to run the Group3 Control system. This can be done using the Group3 C++ Library (see section 3.3.2) or by users writing their own low level interface. Either way the Group3 file **PCI_LIB.DLL** is required. See sections 4.7, 5.5 and 6.3 for specific information about how the interrupts are used on the PCI Loop Controller. Neither the Group3 LabVIEW driver nor the Group3 DDE server support interrupts.

## 3.2 Memory Access Drivers

In order to reduce the consequences of a program crashing, the 32-bit Windows operating systems restrict access to memory. Since the Group3 Loop Controllers require direct access to memory (dualport RAM) a system driver is required to gain this access.

Group3 offers drivers for using Loop Controllers under 32-bit Microsoft operating systems. These drivers are supplied on floppy disk with every Loop Controller card. They are also supplied with Group3's LabVIEW Driver, C++ Library and DDE Server. Comprehensive information on using these drivers is also supplied on the floppy disk.

The following table lists the drivers that Group3 supplies:

<table>
<tr><th></th><th>Win 3.X/95/98/ME<br>(16-Bit)</th><th>Win 95<br>(32-Bit)</th><th>Win 98/ME<br>(32-Bit)</th><th>Win NT/2000/XP<br>(32-Bit)</th></tr>
<tr><td>PCI<br>Driver Files</td><td>✘<br>Not supported</td><td>WINDRIVR.VXD<br>\WINDOWS\SYSTEM\VMM32<br>PCI_LIB.DLL<br>\WINDOWS\SYSTEM</td><td colspan="2">WINDRVR.SYS<br>\WINDOWS\SYSTEM32\DRIVERS<br>PCI_LIB.DLL<br>\WINDOWS\SYSTEM32</td></tr>
<tr><td>ISA<br>Driver Files</td><td>No driver required</td><td colspan="2">VSS_PP.VXD<br>PP_95.DLL<br>PP_32.DLL<br>\WINDOWS\SYSTEM</td><td>G3LCDRV.SYS<br>\WINNT\SYSTEM32\DRIVERS</td></tr>
</table>

The interface to each of the memory drivers used for Group3 Loop Controllers is via a chain consisting of a .DLL file, a .C file and a .C++ file, each with a corresponding published interface: .H file. All of these files are provided with the Group3 C++ Library. These file chains allow developers to interface to the Group3 drivers at either the C or C++ level or use them as an example and simply write their own in any other programming language. For example, the Group3 LabVIEW Driver interfaces to the memory drivers via a LabVIEW "VI" which contains a LabVIEW "CIN" which directly uses the .C files mentioned above.

## 3.3 Software Interfaces

Group3 has developed software interfaces to the Group3 Control system, described in the next three sub-sections. They were developed with the aid of this manual (Group3 Control Loop Controller Programmer's Manual). A programmer using one of these interfaces will still need to create their own specific application using either LabVIEW, C++ or some DDE aware package but all the details of the Loop Controller memory layout and protocols for data exchange have been taken care of by these software interfaces.

All of these products are shipped with a full set of drivers for gaining access to dualport RAM under 32-bit Microsoft operating systems.

The following table lists the drivers that Group3 supplies:

| | **Win 3.X/95/98/ME (16-Bit)** | **Win 95 (32-Bit)** | **Win 98/ME (32-Bit)** | **Win NT/2000/XP (32-Bit)** |
|---|---|---|---|---|
| **LabVIEW** Application Files | ✘ Not supported | **G3C.LLB** | | |
| **C++ Library** Application Files | **C++ Library** | | | |
| **DDE Server** Application Files | **G3DDE16.EXE** | **G3DDE32.EXE** | | |

### 3.3.1 LabVIEW Driver

LabVIEW from National Instruments is a high level graphical programming tool suitable for developing control applications. Group3 has developed a LabVIEW driver which facilitates running a Group3 Control system. This driver will handle all the set-up tasks and accessing of all the I/O data in a Loop Controller. It handles all the dualport arbitration and error handling. Some example applications are also included with the driver.

Group3 offers three versions of this driver, for use with LabVIEW versions 4, 5 and 6 (plus sub-versions). All of these versions will run under Microsoft Windows 95/98/ME/NT/2000/XP operating systems.

The Group3 LabVIEW driver is a set of VIs (Virtual Instruments) written in the LabVIEW graphical programming language. These VIs are intended for use in LabVIEW applications that a user develops.

The details of driver installation and how to use the VIs in a LabVIEW application are documented in the manual "Group3 Control LabVIEW Driver for Windows User's Manual" which is shipped with the Group3 LabVIEW driver software.

### 3.3.2 C++ Library

Group3 has developed a comprehensive collection of C++ code that greatly speeds up the writing of applications to run a Group3 Control system. The C++ Library contains routines to handle all the set-up tasks, and to access all the I/O data in a Loop Controller. It handles all the dualport arbitration and error handling. It also has several example programs.

The C++ Library was written completely without usage of any operating system specific commands or features. Because of this, it can in theory be used on absolutely any computer platform/operating system. Obviously therefore, all aspects of the Graphical User Interface are up to the user of the C++ Library to create.

The C++ Library is highly recommended for people who want to write their own software. For those using some language other than C++ this toolkit still contains excellent documentation and examples that will save considerable time. It was after all written by the people who created the Group3 Control product range.

### 3.3.3 DDE Server

Group3 offers two distinct versions of their DDE server - a 16-bit version for Windows 3.x/95/98/ME and a 32-bit version for Windows 95/98/ME/NT/2000/XP.

The Group3 DDE server forms the interface between the dualport RAM on the Group3 Loop Controller and general application programs that require access to the data it contains. The Group3 DDE server reads and writes the values of the I/O data in the dualport RAM, and converts them in and out of the format required for DDE data sharing. DDE is a standard for data interchange between programs. Because the I/O data from a Group3 Control system is available in the DDE format, any program that is DDE aware can gain access to that data. The DDE based program most commonly used with Group3 Control is InTouch, part of the Factory Suite offered by WonderWare Corp. However any DDE aware program can be used. For example the values of particular I/O points in a Group3 Control system can be linked to specific cells in an “Excel” spreadsheet.

The details of DDE server installation and the format needed to address the I/O data are documented in the manual “Group3 DDE Server Operation Manual” which is shipped with the Group3 DDE server software.

# 4 LC to DI Loop Communications (SDLC)

## 4.1 Introduction

The Group3 Loop Controller can be set to communicate directly with a loop of Group3 Device Interfaces (DIs), continually sending control information to them and gathering status information from them (analogs, digitals, motor positions etc). The status information is automatically placed in the dualport RAM on the Loop Controller card. The computer can then get the latest data simply by reading the defined memory areas. Likewise the computer can control remote outputs simply by placing the new desired values in their defined memory areas.

The communications on the fiber optic cables runs at 1.152Mbaud using SDLC. This is a fast, robust communications protocol which includes CRC checksums sent with each data packet. If the receiving DI or LC detects a corrupted message it will discard it.

## 4.2 Dualport RAM Space Allocation / Initialisation

The dualport RAM (shared memory) is divided into three areas:

- **System Data Area**: used for storing system dependent parameters.
- **I/O Definition Area**: used for defining the I/O areas used in the system.
- **I/O Data Area**: used for storing the data sent to, and received from, the DIs.

The computer must set up and completely initialise the dualport RAM for the System Data Area and the I/O Definition Area. An I/O Data Area of the appropriate size must be allocated somewhere in the remaining dualport RAM for each I/O definition and its address offset must be stored in that I/O definition.

The dualport RAM in the Loop Controller must be set up with an I/O definition and data area for each I/O board connected to the loop via a DI. Any boards on the loop without definitions in dualport will be out of the control of the computer.

Full initialisation of the I/O Data Areas is advisable. For example, the Port Number and Port Type for a type F board are the only two bytes that **have to be set** in its I/O Data Area but if its Send Count and Receive Count are not set to zero, the computer will initially send and receive garbage. The same applies to the inputs and outputs for all I/O boards. The recommended initial setting for each Send Data Flag is 1 (data ready to go) and for each Receive Data Flag is 0 (no received data yet).

# Memory Allocation Map: System Data and I/O Definitions

Typical data is given for a system with one DI containing a type ‘C’ and a ‘D’ I/O board.

| System Data | System Flag 01 to load new set-up | Comm’s Mode 00 = SDLC | Comm’s Enabled | No of I/O Definitions | System Error | Extended Error | Accumulated Error Count LSB | Accumulated Error Count MSB | Number of Messages Sent LSB | | | Number of Messages Sent MSB | Number of Messages Received LSB | | | Number of Messages Received MSB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Typical Data** | 00 | 00 | 01 | 02 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| Offset PC | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
| VME | 01 | 03 | 05 | 07 | 09 | 0B | 0D | 0F | 11 | 13 | 15 | 17 | 19 | 1B | 1D | 1F |

| System Data | Reserved | | | | | Timeout Flag | Timeout Count | Timeout Kick | Software Version | | | | Last I/O Def’ Updated | Comm’s Status | Loop Status | Reserved |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Typical Data** | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 35 | 2E | 30 | 64 | 00 | 00 | 00 | 00 |
| Offset PC | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F |
| VME | 21 | 23 | 25 | 27 | 29 | 2B | 2D | 2F | 31 | 33 | 35 | 37 | 39 | 3B | 3D | 3F |

| I/O Definitions | DI Address | Board Number | Board Type | Offline Flag | Offset to Data LSB | Offset to Data MSB | Board Sub-Type | Reserved | DI Address | Board Number | Board Type | Offline Flag | Offset to Data LSB | Offset to Data MSB | Board Sub-Type | Reserved |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Typical Data** | 00 | 01 | 03 | 00 | 30 | 00 | 00 | 00 | 00 | 02 | 04 | 00 | 42 | 00 | 00 | 00 |
| Offset PC | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 2A | 2B | 2C | 2D | 2E | 2F |
| VME | 41 | 43 | 45 | 47 | 49 | 4B | 4D | 4F | 51 | 53 | 55 | 57 | 59 | 5B | 5D | 5F |

If there were further I/O boards in the system then their definitions would follow.
After the I/O board definition area is the individual board I/O Data Area.

## 4.2.1 System Data Area

The first 32 bytes of the dualport RAM contain system configuration information. This is used as follows:

| | bytes | |
|---|---|---|
| System Flag | 1 | set to 1 to load new configuration |
| Communication Mode | 1 | 0 = LC to DI, 7 = Fast LC to DI |
| Communications Enabled | 1 | 1 = enabled, 0 = disabled,<br>3 = enabled with interrupts (for PCI LCs) |
| Number of I/O Definitions | 1 | |
| System Error | 1 | |
| Extended Error Information | 1 | (also actual comm's state flag) |
| Accumulated Error Count | 2 | |
| Number of Messages Sent | 4 | |
| Number of Messages Received | 4 | |
| Reserved | 5 | |
| Time Out Flag | 1 | set to 1 to enable timeout feature |
| Time Out Count | 1 | sets the timeout period |
| Time Out Kick | 1 | must write to this regularly |
| LC Software Version | 4 | ASCII string written by LC processor |
| Last I/O Def' Updated | 1 | the I/O def' number starts at 1 |
| Comm's Status | 1 | 1 = communicating over loop |
| Loop Status | 1 | 0 = loop OK |
| Reserved | 1 | |

**System Flag**

The System Flag has two valid states: clear = 0, and set = 1.

- The computer sets this flag to 1 tell the LC processor to load the new set-up that has been placed in dualport RAM.
- The LC processor reads the new set-up information then clears this flag to 0.

Before setting this flag, the computer must set up the dualport RAM so that the LC knows what I/O boards it is expected to communicate with on the loop. When this setting up process is complete the computer sets the System Flag. The LC then examines and processes the dualport RAM containing the I/O definitions, checking the validity of the configuration as it goes. Once complete, the LC clears the System Flag to indicate to the computer that the set-up stored in dualport RAM has been used.

**Communication Mode**

This byte tells the Loop Controller what communication mode to use.
The following values are acceptable for LC to DI Loop communications:

| Mode | Value |
|---|---|
| LC to DI Loop | 0 |
| Fast LC to DI Loop (multi-boards/message) | 7 |

Either LC to DI or Fast LC to DI mode can be used to communicate with a loop of DIs. Both communicate at 1.152Mbaud but Fast LC to DI groups messages to the same DI for type B, C, D, H and J I/O boards and sends them out as one message. Generally the Fast LC to DI mode operates at about twice the speed of regular LC to DI mode.

**Communications Enabled**

Three allowed states:

0 disabled
1 enabled
3 enabled, with interrupts (PCI LCs only).

This byte enables or disables communication between the LC and the DIs. If the control program is shut down it is recommended that it should first clear this location.

If the configuration in dualport RAM is to be changed, it is important that communications are shut down first. It is not enough just to clear this location to 0 however as the Loop Controller will continue to communicate for a short period until it checks this location. Once this location has been cleared, the computer should either just wait for a small period of time, say 200ms minimum or wait for the Comm's Status (location $1D_h$) to change to 0. See also note 11 in section 7 of this manual.

Only the PCI Loop Controller supports interrupts but enabling this feature for an ISA Loop Controller will do no harm.

A few milliseconds after communications have been enabled, the Loop Controller will begin sending data messages to the remote hardware on the loop. For most applications it is important that the data areas in dualport RAM are set-up with sensible values before allowing them to be sent to the I/O boards.

**Number of I/O Definitions**

The value here indicates the number of I/O definitions in the I/O Definition Area of the dualport RAM. Note that this number is not necessarily the same as the number of I/O boards in the DIs on the loop. For instance each port of a type F (serial comm's) board has an I/O definition of its own, and some of the I/O boards in the DIs may not be being used. Furthermore if the operators are using over-the-loop diagnostic ports then this number will be the number of I/O boards/ports in use plus the number of diagnostic port windows being operated over the loop. Additionally the Parameter Tool, if used, requires one I/O definition.

The maximum number of I/O definitions is 60.

**System Error**

The errors reported here fall into two main categories. The first are set-up errors, those that the LC will detect and report when processing the new set-up that has been placed in dualport RAM. The second category are communications errors, those which are detected while the loop is running.

If a communication error occurs, the LC will store the error number in the System Error location and then continue with communications. It is the responsibility of the computer to periodically check the System Error location for errors, and to clear the Error byte after examining it. Until the computer has done this, the LC will not report new errors.

For some errors, more information can be found in the Extended Error Information field.

The list of possible error numbers follows:

**Set-Up Errors**

| **Error Description** | **Error Number** Hex. | Decimal | |
|---|---|---|---|
| Invalid Communication Mode | 01 | 1 | |
| Too Many I/O Board Definitions | 02 | 2 | |
| Invalid DI Address | 03 | 3 | * |
| Invalid I/O Board Number | 04 | 4 | * |
| Duplicated I/O Board Address | 05 | 5 | * |
| Invalid Board Type | 06 | 6 | * |
| Overlapping Memory Allocation | 0C | 12 | * |
| Out of Dualport RAM | 0D | 13 | * |
| Invalid Fiber Optic Port Type | 1F | 31 | * |
| Invalid Fiber Optic Port Number | 20 | 32 | * |

**Communication Errors**

| **Error Description** | **Error Number** Hex. | Decimal | |
|---|---|---|---|
| Unnecessary Beacon | 14 | 20 | t |
| Break In Loop | 15 | 21 | t |
| Loop Inactive | 16 | 22 | |
| Loop Not Echoing | 17 | 23 | |
| DI Not Responding | 18 | 24 | t |
| Bad SDLC Packet | 19 | 25 | s |
| Invalid Board Command | 1A | 26 | * |
| Non-Existent Board | 1B | 27 | * |
| Invalid Command | 1C | 28 | * |
| Unrecognised Command | 1D | 29 | t |
| Invalid Fiber Optic Port Type | 1F | 31 | * |
| Invalid Fiber Optic Port Number | 20 | 32 | * |

***** Indicates errors which have the index number of the offending I/O definition stored in the Extended Error Information field. Definitions are assigned a sequential index number, starting with 1 for the first definition in the definition area.

**t** Indicates errors which have the offending DI address number stored in the Extended Error Information field.

**s** Indicates errors which have a status byte stored in the Extended Error Information field.

**System Error Notes:**

**$0D_h$ Out of Dualport RAM**

The indicated I/O definition refers to a data area which extends beyond the end of dualport RAM.

**$14_h$ Unnecessary Beacon**

This error is very unlikely to happen but probably indicates a faulty DI if it does.

**$15_h$ Break In Loop**

Wait for one or two seconds before reading out the address of the DI from the Extended Error Information field. If there is more than one DI between the break and the Loop Controller on the return fiber then it will take this amount of time before the address is reported correctly. If there are no DIs between the break and the Loop Controller on the return fiber then this error will not be reported.

**$17_h$ Loop Not Echoing**

This is the only error that is guaranteed to be reported and keep being reported while there is a break in the loop.

**$19_h$ Bad SDLC Packet**

The status information for this error will be reported in the Extended Error Information field:

Bit-2 set if CRC error detected
Bit-3 set if overrun error detected
Bit-4 set if frame with residue bits detected
Bit-5 set if frame with abort end detected
Bit-6 set if short frame detected
Bit-7 set if end of receive frame detected

This error typically occurs as a result of damaged or loose communication fibers and its exact cause as determined by the Status byte is usually random. Interpretation of this byte is therefore not generally helpful.

**$1A_h$ Invalid Board Command**

A wrong board type for a given DI and board address was set in dualport RAM. This error was reported back from the DI.

**$1C_h$ Invalid Command**

An invalid command was sent to board 0 (the processor board) for a DI. This error is very unlikely to happen, except as a parameter error (see section 4.4).

**$1D_h$ Unrecognised Command**

This error is very unlikely to happen and can generally be ignored.

**Extended Error Information**

Contains further error information as described before.

Also, if the system error is 0, this location is used as a flag to indicate the actual state of communications on the loop:

1 loop communicating

0 loop shut down

This function is now superseded by the comm's status location.

**Accumulated Error Count**

This location keeps a count of the number of errors which have occurred. It is stored as a 2-byte unsigned integer.

**Number of Messages Sent**

This is a count of the total number of messages which have been sent on the loop, stored as a 4-byte unsigned integer.

**Number of Messages Received**

This is a count of the total number of messages which have been received on the loop, stored as a 4-byte unsigned integer.

**Time Out Flag**

LC to DI mode features a time out facility. The Loop Controller can be programmed to set digital, analog, and motor outputs to zero on selected I/O board channels if the computer stops communicating with the Loop Controller. This time out feature is enabled by setting the Time Out Flag to 1.

**Time Out Count**

The Time Out Count byte stores the period that must elapse after the computer stops communicating with the Loop Controller before time out action is taken. The count may be in the range 1 to 255. The time out period is the time out count multiplied by 0.1 seconds.

**Time Out Kicker**

If the time out feature is enabled the computer must continually load the Time Out Kicker with a non-zero value at intervals less than the time out period. The Loop Controller checks the Time Out Kicker location at intervals equal to the time out period. If the Loop Controller finds that the value is non-zero, it clears it to zero and starts another timer period. If the Loop Controller finds that the value has remained at zero, time out action is taken.

**Software Version**

The processor on the LC card writes its software version into this location on startup. It is stored as a 4-byte ASCII string of the form "5.0d".

The computer can read this location to verify that an LC is present, and that the LC processor has started correctly. It can also be read out by the computer and used for controlling the use of Loop Controller features which were introduced at specific software versions (see Appendix B).

Note: the suffix for software versions can also be the ' ' (space character – 20 hex). As a version, this precedes 'a', 'b', 'c' ...

**Last I/O Def' Updated**

As the LC processor updates a block of I/O data from information brought in off the loop, it stores the number of the corresponding I/O definition in this location. Note that the numbering of I/O definitions starts at 1.

By monitoring this location the control software can easily be pointed to the newly arrived data, speeding up its retrieval. The user can clear this location at will, to assist in detecting changes.

**Comm's Status**

This byte indicates when the LC is communicating. It will be set to 1 a few ms after the Communications Enabled flag is set and will be cleared to 0 a few ms after the Communications Enabled flag is cleared. When shutting down the LC communications to change the dualport configuration, it is important to wait until this location is cleared. This location has been used for this purpose from LC software version 4.3a and onwards. See also note 11 in section 7 of this manual.

**Loop Status**

The Loop Status location replicates the functionality of LC errors 15, 16 and 17 (hex). These errors are actually status codes and so do not lend themselves to the event-type error mechanism offered by the System Error location. If this byte is non-zero then there is a broken loop. If it is zero then the loop is intact or the LC is not attempting to communicate. The best use of this byte is to test whether it is zero or non-zero. The make-up of this byte is as follows:

| | | | |
|---|---|---|---|
| bits | 0..3 | Address of DI closest to break if bit-4 is set | |
| bit | 4 | Set if break in loop | (equivalent to error 15 hex) |
| bit | 5 | Set if loop is inactive | (equivalent to error 16 hex) |
| bit | 6 | Set if no loop echo | (equivalent to error 17 hex) |

This location has been used for this purpose from LC software version 4.3a and onwards. See note 12 in section 7 of this manual.

Wait for one or two seconds before reading out the address of the DI associated with bit-4. If there is more than one DI between the break and the Loop Controller on the return fiber then it will take this amount of time before the address is reported correctly.

## 4.2.2 I/O Definition Area

This section of dualport RAM is used to define each active I/O board in each DI on the loop, as well as each active over-the-loop diagnostic port and Parameter Tool definition. The memory area allocated to I/O definitions forms a contiguous block starting immediately after the 32 bytes of the System Data Area, with 8 bytes allocated to each I/O definition.

Note that the number of I/O definitions is not necessarily the same as the number of I/O boards in the DIs on the loop. For instance each port of a type F (serial comm's) board has an I/O definition of its own, and some of the I/O boards in the DIs may not be being used. Furthermore if the operators are using over-the-loop diagnostic ports then the number of I/O definitions will be the number of I/O board definitions plus the number of diagnostic port windows being operated over the loop. Additionally the Parameter Tool, if used, occupies one I/O definition.

The format for each I/O definition is as follows:

| | **bytes** | |
|---|---|---|
| DI Address | 1 | as set with switch on DI (0-$F_h$) |
| Board Number | 1 | as set with jumpers on board (1-3) or 0 |
| Board Type | 1 | see selection on Board Type |
| Offline Flag | 1 | 0 = online, 1 = offline |
| Offset to Data | 2 | offset from start of dualport RAM |
| Reserved | 2 | |

Note that the type F (serial comm's) board requires two I/O definitions - one for each port (if both ports are to be used). The I/O definitions for the two ports have the same values for DI Address, Board Number and Board Type - they differ only in the Offset to Data value. The Port Number is contained in the I/O data, not the I/O definition.

**Device Interface Address**

The computer sets this location to the DI address. The address range is 0 to 15($F_h$), and must match the address (0 - F in hex.) set by the address switch on the DI.

The value of this byte for the Parameter Tool definition is fixed at FE hex

**Board Number**

The computer sets this location to the number of the I/O board being defined. A DI can contain 1, 2, or 3 I/O boards, and each I/O board must have a unique address of 1, 2 or 3 as selected by the jumpers on that board (see the Group3 Control Users Manual, page 2-9). These board number assignments are used both by the diagnostic port and the fiber optic communication. The relative physical position of the boards in the DI does not affect their address. Gaps may exist in the board numbering scheme within a DI. For instance, a DI may have only two boards with addresses of 1-and-2 or 2-and-3 or 1-and-3. In all cases the Board Number location must be set to the address of the board to be controlled as defined by its address jumpers.

The processor board in a DI is always defined as board 0. The over-the-loop diagnostic port is associated with the processor board so when configuring one of these, this location should be set to 0.

Note that for the CNA module, if addressing I/O data, the Board Number is 1. If talking to a CNA over-the-loop diagnostic port, the Board Number must be set to 0 to access the diagnostic port features.

When configuring a Parameter Tool, this location should be set to 0.

**Board Type**

The computer sets this location to the I/O board type. Allowed values are as follows:

| Board type | Value | |
|---|---|---|
| A - Fast Analog/Digital I/O Board | 1 | |
| B - Digital I/O Board | 2 | |
| C - 8-Analog Input Board | 3 | |
| D - 8-Analog Output Board | 4 | |
| E - 4-DC Motor Driver Board | 5 | |
| F - Serial Communication Board | 6 | |
| G - 4-Stepper Motor Driver Board | 7 | |
| H - 4-Encoder Input Board | 8 | |
| J - 2-Precision Analog Output Board | 10 | ($0A_h$) |
| K - GPIB / IEEE 488 Controller Board | 11 | ($0B_h$) |
| CNA - I/O module with PID | 101 | ($65_h$) |
| Diagnostic Port (over-the-loop) | 6 | (as per serial) |
| Parameter Tool | 13 | ($0D_h$) |

**Offline Flag**

This is a status location for each board and is used by the Loop Controller to indicate whether each board is online (0) or offline (1). This location should be initialised by the computer to 0. The Loop Controller determines that a board is offline if it doesn't reply to ten consecutive messages. The response time for the Loop Controller to notify that a board is offline is not fixed and is dependent on the number of I/O boards on the loop. With this qualification, a typical board-offline response time for a loop of say 20 I/O boards will be 200ms to 400ms - less in Fast LC to DI mode.

Typically the offline flag will be detecting boards that might simply not be present on the loop when comm's is started or that may have been addressed incorrectly rather than boards which cease to operate during operation. The status is in effect the accumulation of System Communication Errors $18_h$, $1A_h$, $1B_h$, $1C_h$, $1F_h$ and $20_h$ for a given board on the loop.

The offline flag will not work for a Parameter Tool as these are not associated with any one physical board.

This location has been used for this purpose from LC software version 5.0c and onwards.

**Offset to Data**

This contains the offset from the start of the dualport RAM to the start of the I/O Data Area for this I/O definition. This is a 2-byte unsigned integer, stored least significant byte first.

The computer must set up these offsets taking into account the number of bytes each I/O definition requires for its data area. The size of each data area is implicit in its Board Type (see Memory Allocation in section 4.2.3 I/O Data Areas). The data areas must not overlap. To conserve dualport RAM space the data areas are usually set to be following on from each other, but on a small system this is not essential.

Note that each I/O Data Area can be positioned anywhere within the remaining free space of dualport RAM. One convenient algorithm is to stack the data areas downwards from the top of dualport. The remaining free space is then the gap between the last I/O definition added and its corresponding data area.

**Board Sub-Type**

This location is used to distinguish between operational differences of IO boards and should not be confused with the various board categories, described in the "Group3 Control User's Manual" (e.g. A1, A2, A3). It is normally set to 0.

So far only the H board is affected by this location. If it is set to 0 for an H board, the storage of encoder data in dualport is 16-bits. If it is set to 1 for an H board, the storage of encoder data in dualport is 32-bits. Note: this affects the necessary size of the H board data area.

This location has been used for this purpose from LC software version 5.1 and onwards.

## 4.2.3 I/O Data Areas

Refer to the memory allocation diagram on page 4-14 when reading the following text.

### Send and Receive Data Flags

The first two locations of each boards I/O Data Area are used for the **Send Data Flag** and the **Receive Data Flag**. These two flags are used for handshaking between the computer and the Loop Controller during storage and retrieval of data blocks. These ensure that consistent blocks of data are read out, both by the computer and the Loop Controller. Problems could otherwise arise if, for instance the computer, had only half read out the required data from a block of the dualport RAM when an updated block of data was written in from the other side by the Loop Controller. The flags should be used in the following way:-

**Outputting Data**
To write data into an output data area:

1. change the Send Data Flag to make it even (this is easily done by clearing bit-0)
2. write the output data into the output data area
3. change the Send Data Flag to make it odd (and different from its starting value) (easily done by adding three)

In this manner if the Send Data Flag has an even value it indicates that data is being changed, and the Loop Controller will not attempt to read the output block until the Send Data Flag has been changed to an odd value. To make sure this does not introduce delays in outputting data the programmer must change the Send Data Flag back to an odd number as soon as the output data has been written.

Note the suggested method produces sequential flag values of 1, 0, 3, 2, 5, 4, .....

**Inputting Data – Method 1**
To read data from an input data area:

1. wait until the Receive Data Flag is odd
2. read the input data from the input data area
3. check that the Receive Data Flag has not changed (if it has repeat from step 1)
4. return the input data

In this manner if the Receive Data Flag still has an odd value, and has remained unchanged, it indicates that the data is consistent. If the Receive Data Flag has changed then it means the data was probably updated half way through the read operation. The data should be discarded, and the input data block read again.

The data unavailable times (as determined by the Receive Data Flags in dualport RAM) are highly optimised, but if a dualport collision occurs between the Loop Controller software and application software using this inputting method, access will be blocked in step 1 (via. the Receive Data Flag) for up to the maximum times listed next (times in micro-seconds):

| | | |
|---|---|---|
| A Board | 7.8 | |
| B Board | 5.2 | |
| C Board | 22.1 | |
| D Board | N/A | |
| E Board | N/A | |
| F Board (General Serial) | 40.4 | (same for K Board and diagnostic port) |
| F Board (DTM) | n x 22.8 - 5.5 | (LC version < 4.3) |
| F Board (DTM) | 13.3 | (LC version >= 4.3 (flag per DTM)) |
| G Board | 69.0 | (LC version < 4.3) |
| G Board | 8.1 | (LC version >= 4.3 (flag per Stepper Motor)) |
| H Board | 11.7 | (LC version < 5.1 or sub-type = 0 (16-bit encoders)) |
| H Board | 22.1 | (LC version >= 5.1 and sub-type = 1 (32-bit encoders)) |
| J Board | N/A | |
| CNA Board | 9.1 | |
| Parameter Tool | 18.9 | (read) |
| Parameter Tool | 4.2 | (write or error) |

**Inputting Data – Method 2**

The computer maintains a local copy of each input data field (local_data), and returns this value if the dualport RAM cannot be accessed at the time of the request. With this method the computer application never has to wait for data - it can always get a useable value from the locally held copy. Even though the data-unavailable times which can be experienced in Method 1 are quite short, the down-time effect of waiting for data becomes more significant with modern fast computers. With Method 2 the computer uses the following algorithm to fetch new data and update the locally held copy:

1. read the Receive Data Flag (old_rx_flag)
2. if old_rx_flag is odd:-
3. read the input data from the input data area (new_data)
4. read the Receive Data Flag (new_rx_flag)
5. if old_rx_flag is the same as new_rx_flag
6. store new_data into local_data
7. return local_data

This method can be augmented so that the computer application can retain knowledge as to whether the fetched data is new or old. Sometimes this is useful.

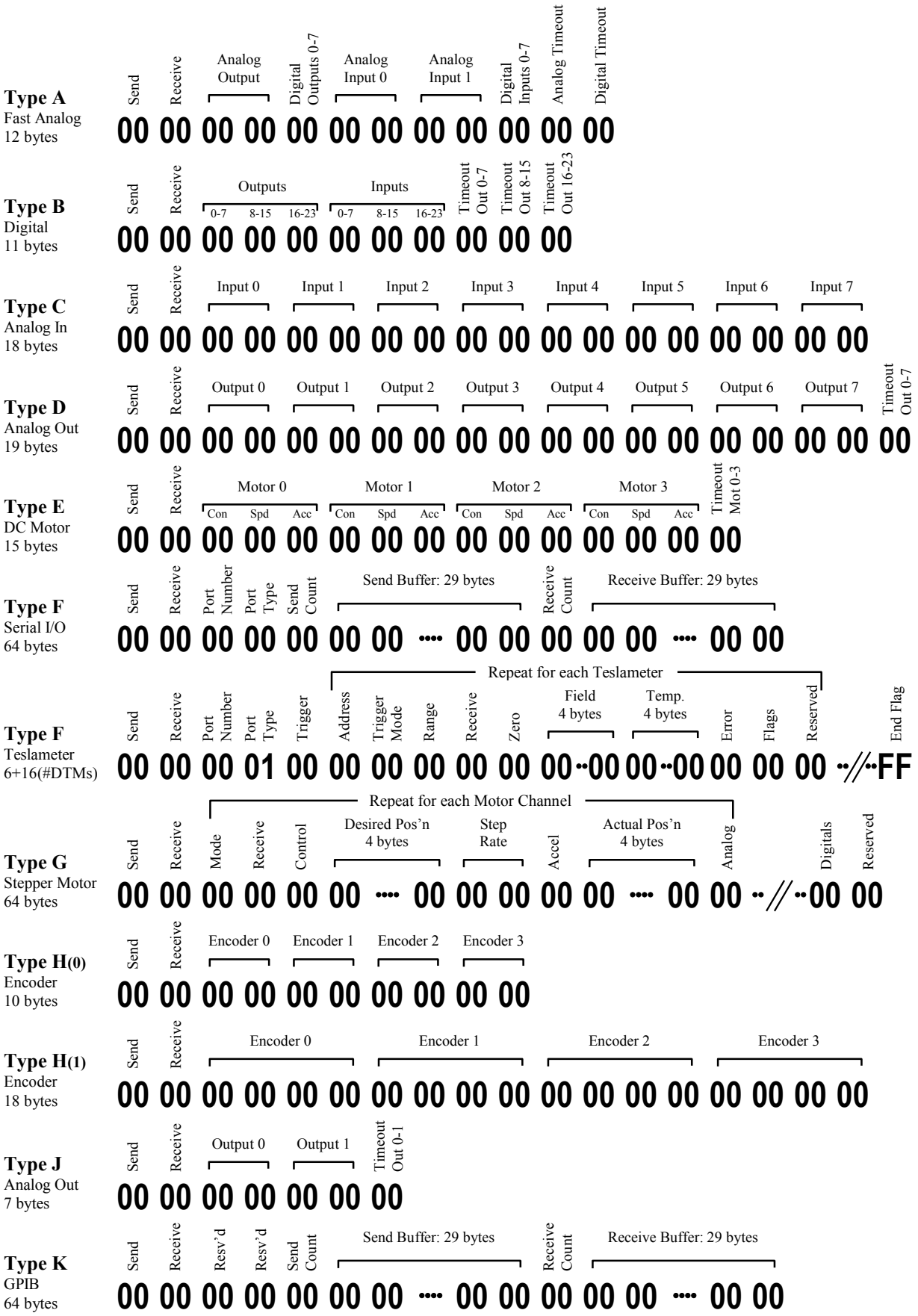
## Timeout Bytes

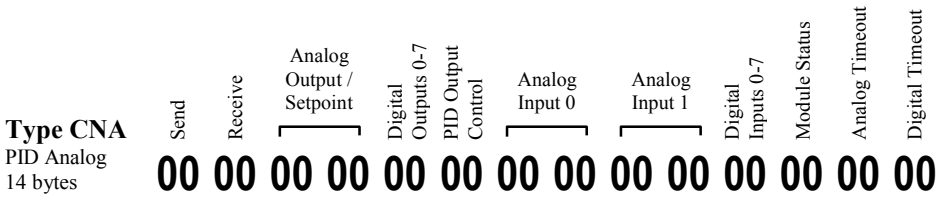
For I/O boards with control (output) values associated with them, there are locations set aside at the end of their I/O Data Areas to determine, in the event of a time out (a break in communications between the computer application and the LC), whether the control value will be left as it is or set to zero. A bit is used for each channel, whether it is analog, digital or motor. If the bit is 1 then in the event of a time out the control value is unchanged, but if the bit is 0 the control value is set to zero. In all cases, the bit number corresponds to the channel number, eg bit-0 corresponds to analog output 0 on a D board.

## I/O Data

After the Send Data Flag and Receive Data Flag is the actual I/O data. These take different forms for each I/O board, and are described in detail in the following pages.

# Memory Allocation Map: I/O Board Data Formats




Type CNA
PID Analog
14 bytes
Send
Receive
Analog Output / Setpoint
Digital Outputs 0-7
PID Output Control
Analog Input 0
Analog Input 1
Digital Inputs 0-7
Module Status
Analog Timeout
Digital Timeout
00 00 00 00 00 00 00 00 00 00 00 00 00 00

## 4.2.3.1 Type A - Fast Analog/Digital I/O Board

The 2-byte analog values are stored least significant byte first. They will be either 2's complement or unsigned integers, depending on the configuration of the A-board.

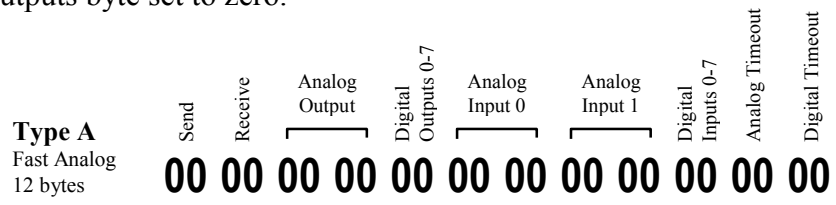
The analog output channel is 14-bits and can be set to a maximum of:

bipolar: -8000 to +8000 (full scale)
unipolar: 0 to +16000 (full scale)

The analog input channels are 16-bits and will be set to a maximum of:

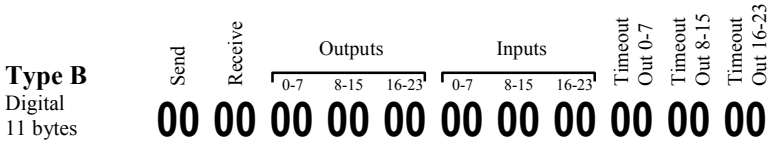
bipolar: -32000 to +32000 (full scale) (some over-range is possible)
unipolar: 0 to +64000 (full scale) (some over-range is possible)

Digital channels that are going to be used as inputs should have the corresponding bit in the outputs byte set to zero.




## 4.2.3.2 Type B - Digital I/O Board

Digital channels that are going to be used as inputs should have the corresponding bit in the outputs byte set to zero.

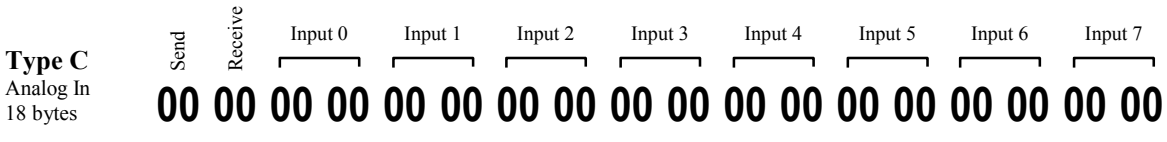



## 4.2.3.3 Type C - 8-Analog Input Board

The 2-byte analog values are stored least significant byte first. They will be either 2's complement or unsigned integers, depending on the configuration of the C-board.

The analog input channels are 16-bits and will be set to a maximum of:

bipolar: -32000 to +32000 (full scale) (some over-range is possible)
unipolar: 0 to +64000 (full scale) (some over-range is possible)

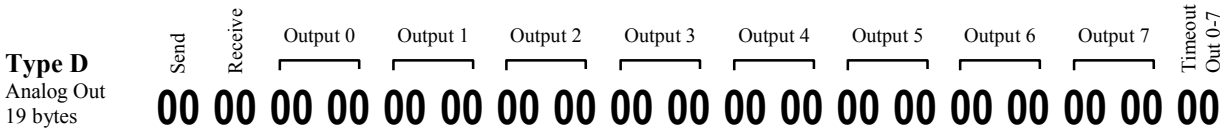


## 4.2.3.4 Type D - 8-Analog Output Board

The 2-byte analog values are stored least significant byte first. They will be either 2's complement or unsigned integers, depending on the configuration of the D-board.

The analog output channels are 14-bits and can be set to a maximum of:

bipolar: -8000 to +8000 (full scale)
unipolar: 0 to +16000 (full scale)

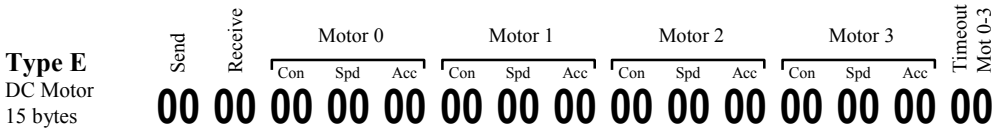



## 4.2.3.5 Type E - 4-DC Motor Driver Board

The Control byte can take the following values:

0 motor off (windings open)
1 motor forward run
2 motor reverse run
3 motor stop (windings shorted)

Only low values of acceleration (say 1 to 30) produce audible/visible rates of increase - anything greater appears to a person to produce instant full speed.




## 4.2.3.6 Type F - Serial Communication Board

Each port of the type F board is treated in the LC dualport as if it were a separate I/O board. The I/O definitions for each port have the same data except the Offset to Data.
The Data area for each port contains the port number identifier, for port 0 or port 1.

The type F board allows two selectable modes of operation:
- General Serial communication
- DTM Loop communication

## General Serial Communication (Occupies 64 bytes)

In this mode, a message placed by the computer in dualport RAM is sent out by the fiber optic transmitter on the type F board. Similarly, a message received by the type F board fiber optic port from an external device is placed by the Loop Controller in dualport RAM.

Protocols are used by both the application software and the Loop Controller to prevent old messages being overwritten until they have been completely read or new messages being read before they have been completely written. These protocols are described under Send Count and Receive Count and replace the need to use the Send and Receive Data Flags.

There are 29-byte send and receive buffers, so if a message is longer than 29 bytes the computer must re-assemble it from 29-byte segments as they come in.


**Port Number**

This location must be set to **0** or **1**, according to the port being used.

**Port Type**

This location specifies the communication protocol to be used on the port.
For General Serial mode this should be set to **0**.

**Send Count**

The computer sets this location to the number of bytes in the message to be sent. This must be done AFTER the Send Buffer has been written.

The Loop Controller sets this location to 0 after sending the message.

The computer must not place new data in the Send Buffer or change the Send Count until it has been cleared to 0 by the Loop Controller.

**Send Buffer**

The message to be sent is stored in this area, with the first byte to be sent stored at the lower address. See the description of the transmit protocol using the Send Count.

**Receive Count**

The Loop Controller sets this location to the number of bytes in the received message.

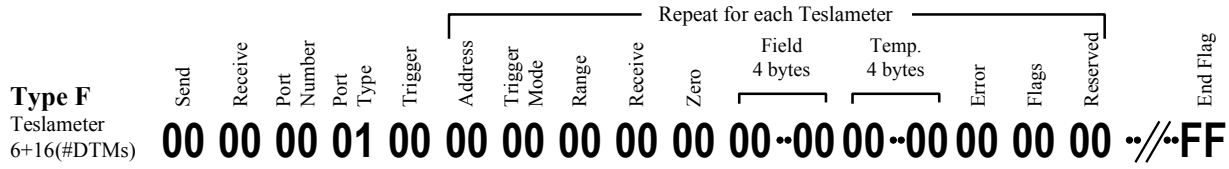
AFTER the computer has read the message out of the Receive Buffer, it should reset this location to zero to indicate to the Loop Controller that the message has been read. The Receive Buffer is then available for the next message.

**Receive Buffer**

The received message is stored in this area with the first received byte stored at the lower address.

## DTM Loop Communication (Occupies 6+16x(no. of DTMs) bytes)

The data area starts with the Send Data and Receive Data Flags, then has Port Number, Port Type and Trigger. After the Trigger location is a block of data repeated for each teslameter connected to the LC. After the last teslameter data block is an End Flag, value FF hex.




**Port Number**

This location is set to **0** or **1**, according to the port being used.

**Port Type**

This location specifies the communication protocol to be used on the port.
For communicating with a loop of Group3 Digital Teslameters, this should be set to **1**.

**Trigger**

Setting this location to 1 will trigger any teslameters on the loop that have been put in trigger mode. After the Loop Controller has sent the message to trigger the teslameters it will clear this location to indicate the triggering has been done.

**DTM Address**

This location holds the address of the teslameter on the loop. Each teslameter must be set to its own unique address using its internal switches, which allow addresses in the range 0 to 31. A maximum of 8 teslameters can be run from any one port/loop on a type F board.

**Trigger Mode**

This location specifies the trigger mode for this particular teslameter. It is set to either 'C' to invoke continuous mode or 'V' for triggered mode.

**Range**

This location specifies the desired range for the teslameter, and is 0 for the lowest range, through to 3 for the highest range.

If this location is set to FF(hex), the teslameter range will not be changed - the range setting of the teslameter will be determined by the buttons on the front panel of the instrument. (This feature requires the type F board to have software version 3.3 or later).

**Receive**

Used as a flag to indicate that the data for this teslameter is being updated. This flag can be used instead of the overall Receive Flag (at the beginning of the F-board data) – but only for this specific teslameter. Follow the protocols for inputting data outlined in section 4.2.3. (This feature requires the Loop Controller to have LC software 4.3 or later).

**Zero**

When this location is set to 1, a message is sent to the teslameter to zero the field reading on the current range. After the Loop Controller has sent the zeroing message, it clears this location to indicate that the requested function has been performed.

**Field**

This location holds the field value as read for this teslameter. It is a 4-byte value which is a floating point representation of the field using the IEEE 754 format.

**Temperature**

This location holds the temperature reading as read for this teslameter. It is a 4-byte value which is a floating point representation of the temperature in degrees Celsius using the IEEE 754 format.

**Error**

This location holds the code for any errors detected during communication with this teslameter.

**Teslameter Error Codes**

| | |
|---|---|
| 0 | no error |
| 1 | break in DTM communication loop |
| 2 | wrong echo |
| 3 | timeout |
| 4 | message too long |
| 5 | overflow (value does not fit DTM display) |
| 6 | over range (field too high for the selected range) |
| 7 | invalid DTM address |
| 8 | triggered wait (waiting for triggered DTMs to produce new data) |
| 9 | zeroed wait (waiting for zeroed DTM to zero its range(s)) |
| 10 | no temperature probe (probe has no temperature capability) |
| 11 | bad temperature reading (probe has faulty temperature sensor) |
| 12 | no mag' probe (faulty or missing field probe) |
| 13 | fixed range (attempted to set range for fixed range probe) |
| 14 | auto range (attempted to set range for autoranging DTM) |
| 15 | parity error in message from DTM |
| 16 | framing error in message from DTM |
| 17 | overrun error in message from DTM |
| 18 | bad calibration data in DTM (since F board s/w version 5.0a & DTM s/w 7.0) |

Note: Error codes 8 and 9 are actually status codes, not errors. Error code 10 is usually a status and is not necessarily an error.

**Flags**

The bits in this byte can be set to control aspects of the communication with the DTM, including what data is gathered. This feature requires the following software versions:

| | |
|---|---|
| LC | version 5.0f or later |
| DI | version 5.0g or later |
| F-board | version 5.0b or later |

**Allow Address Skipping – (bit-0)**

Bit-0, if set to 1, indicates to the F-board that if successive communications are to the same DTM then it may omit re-addressing that DTM. This can save considerable time if operating a single DTM in triggered mode and allows more data updates per second.

**No Field – (bit-1)**

Bit-1, if set to 1, will prevent the F-board from getting the field value from the DTM and will cause 0.0 to be stored in the dualport Field location.

**No Temperature – (bit-2)**

Bit-2, if set to 1, will prevent the F-board from getting the temperature value from the DTM and will cause 0.0 to be stored in the dualport Temperature location. Some DTM probes don't have a temperature sensor and sometimes the temperature is simply not required. Either way, this feature saves considerable time and allows more data updates per second.

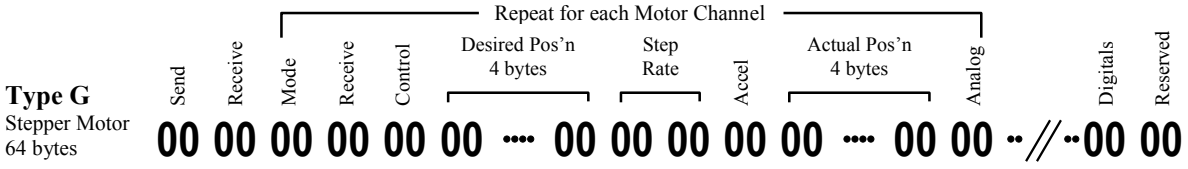
**End Flag**

The block of 16 bytes from DTM Address through Reserved is repeated for each DTM connected to the type F board. After the block for the last DTM, an End Flag byte MUST be set by the computer to FF hex to indicate that there are no more DTM blocks.

The maximum number of DTMs allowed on each port is 8.

## 4.2.3.7 Type G - 4-Stepper Motor Driver Board

After the Send Data and Receive Data Flag locations is a block of data repeated for each stepper motor starting with motor 0. These 4 blocks of 15 bytes are followed by a byte of Digital Inputs and a Reserved byte, making 64 bytes in total. The Desired and Actual Positions are stored as 2's complement 4-byte integers, least significant byte first.




**Mode**

| | | | |
|---|---|---|---|
| bit-0 | run mode | 0 = position control, | 1 = continuous run |
| bit-1 | stop mode | 0 = locked (energised), | 1 = free (windings off) |
| bit-2 | step increment | 0 = full step, | 1 = half step |
| bits 3 – 7 | not used | | |

**Receive**

Used as a flag to indicate that the data for this motor is being updated. This flag can be used instead of the overall Receive Data Flag (at the beginning of the G-board data) – but only for this specific motor. Follow the protocols for inputting data outlined in section 4.2.3. (This feature requires the Loop Controller to have LC software 4.3 or later).

**Control Byte**

bits 1 & 0 (these bits are relevant only in continuous run mode)

- 00 = stop free (all windings off)
- 01 = forward (position count incrementing)
- 10 = reverse (position count decrementing)
- 11 = stop locked (motor energised)

bit-2 (Zero)

- 0 = actual position count accumulates
- 1 = count set to zero

**Desired Position**

A 4-byte integer, stored as 2's complement. A new value placed here will be sent to the stepper motor I/O board and the motor will move to this new position at the preset Acceleration and Step Rate.

**Step Rate**

A 2-byte unsigned integer representing steps per second. The type G board can operate at up to approximately 5000 steps per second, depending on how many motors are operating.

**Acceleration**

A 1-byte unsigned integer, representing the acceleration of the motor.

**Actual Position**

A 4-byte integer, stored as 2's complement.

**Analog Input**

An 8-bit value read from that channel's analog input.

**Digital Inputs**

When these are set as limit switch inputs (through the DI's diagnostic port):

| | | |
|---|---|---|
| bit-0 | 0 = motor 0 lower limit reached, | 1 = motor enabled |
| bit-1 | 0 = motor 0 upper limit reached, | 1 = motor enabled |
| bit-2 | 0 = motor 1 lower limit reached, | 1 = motor enabled |
| bit-3 | 0 = motor 1 upper limit reached, | 1 = motor enabled |
| bit-4 | 0 = motor 2 lower limit reached, | 1 = motor enabled |
| bit-5 | 0 = motor 2 upper limit reached, | 1 = motor enabled |
| bit-6 | 0 = motor 3 lower limit reached, | 1 = motor enabled |
| bit-7 | 0 = motor 3 upper limit reached, | 1 = motor enabled |

Motor direction conventions:

- If a motor is rotating in the **forward** direction, its position count is **increasing** and it will stop if its **upper** limit input is activated.
- Conversely, if a motor is rotating in the **reverse** direction, its position count is **decreasing**, and it will stop if its **lower** limit input is activated.

Note: the following functions are selected through the diagnostic port only:

- Output mode: quadrature, clock and direction, or dual clock.
- Polarity of the MOTOR OFF output signal (applicable to clock modes only).
- Polarity of the HALF/FULL STEP output signal (applicable to clock modes only).
- Activation of digital inputs as limit stops, and polarity of limit inputs.

## 4.2.3.8 Type H - 4-Encoder Input Boards

The H(0) and H(1) boards are physically the same board, just with different dualport configuration. H(0) boards are represented by 16 bit encoders in dualport and H(1) boards are represented by 32 bit encoders in dualport

## 4.2.3.8.0 Type H(0) - 4-Encoder Input Board (16-bit encoders)

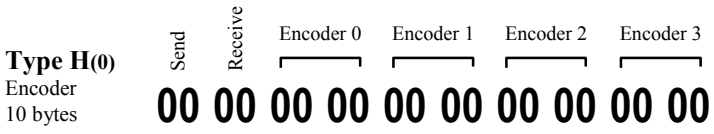
To enable an H board to use 16-bit encoders in dualport, the following conditions must be met:

- The Board Sub-Type byte in the IO definition area for the H board must be set to 0.
- 10 bytes must me allocated for the data area for the H board.

Each 2-byte encoder value is stored least significant byte first. They are 2's complement integers.

If data for an encoder overflows the storage capability for a 2-byte integer, it will be held at the most positive or negative value that can be represented as appropriate: -32768 or 32767.

There is no protocol in place, yet, to allow the computer to reset encoder values in dualport RAM. If this is required, the best procedure is to reset the location, read it back to check it and repeat if necessary.




## 4.2.3.8.1 Type H(1) - 4-Encoder Input Board (32-bit encoders)

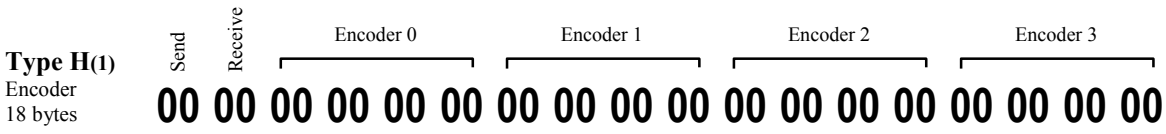
To enable an H board to use 32-bit encoders in dualport, the following conditions must be met:

- The Loop Controller must have software version 5.1 or later.
- The Board Sub-Type byte in the IO definition area for the H board must be set to 1.
- 18 bytes must me allocated for the data area for the H board.

Each 4-byte encoder value is stored least significant byte first. They are 2's complement integers.

If data for an encoder attempts to underflow below –2,000,000,000 or overflow beyond 2,000,000,000, it will be held at either –2,000,000,000 or 2,000,000,000 respectively.

There is no protocol in place, yet, to allow the computer to reset encoder values in dualport RAM. If this is required, the best procedure is to reset the location, read it back to check it and repeat if necessary.

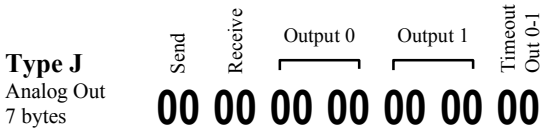


## 4.2.3.9 Type J - 2-Precision Analog Output Board

The 2-byte analog values are stored least significant byte first. They will be either 2's complement or unsigned integers, depending on the configuration of the J-board.

The analog output channels are 16-bits and can be set to a maximum of:

bipolar: -32000 to +32000 (full scale)
unipolar: 0 to +64000 (full scale)



## 4.2.3.10 Type K - GPIB / IEEE 488 Controller Board

A message placed by the computer in the Send Buffer is sent to the type K board. This message takes the form of <K board command><device><data>.

Similarly, a message received from the type K board is placed in the Receive Buffer by the Loop Controller.

There are 29-byte send and receive buffers, so if a message is longer than 29 bytes the computer must re-assemble it from 29-byte segments as they come in.

Details of all the commands applicable to the K board, and the required data formats are included in the documentation shipped with each board.


The two bytes marked Reserved should be set to zero.

**Send Count**

The computer sets this location to the number of bytes in the message to be sent. This must be done AFTER the Send Buffer has been written.

The Loop Controller sets this location to 0 after sending the message.

The computer must not place new data in the Send Buffer or change the Send Count until it has been cleared to 0 by the Loop Controller.

**Send Buffer**

The message to be sent is stored in this area, with the first byte to be sent stored at the lower address. See the description of the transmit protocol using the Send Count.

When placing data in the Send Buffer, the application program should treat the following five characters differently and encrypt them into the 2-character sequences as follows:

| Transmit Character Value In Hex | Encrypt To: 2-Byte Character Sequence |
|---|---|
| $00_h$ | "\0" |
| $03_h$ | "\c" |
| $0A_h$ | "\n" |
| $0D_h$ | "\r" |
| $5C_h$ | "\\" |

**Receive Count**

The Loop Controller sets this location to the number of bytes in the received message.

AFTER the computer has read the message out of the Receive Buffer, it should reset this location to zero to indicate to the Loop Controller that the message has been read. The Receive Buffer is then available for the next message.

**Receive Buffer**

The received message is stored in this area with the first received byte stored at the lower address.

When retrieving data from the Receive Buffer, the application program will have to decrypt the following 2-character sequences into the hex values as follows:

| **Received Character Sequence** | **Decrypt To: Character Value In Hex** |
|---|---|
| "\0" | $00_h$ |
| "\c" | $03_h$ |
| "\n" | $0A_h$ |
| "\r" | $0D_h$ |
| "\\" | $5C_h$ |

**Note 1:** The maximum number of characters that can be transferred with either a GPIB read or write command on the Group3 type K board is 65535, the range of a 2-byte unsigned integer.

**Note 2:** Data received with a GPIB read command can be larger than the 29-byte Receive Buffer and so will be received in 29-byte segments. The application has to concatenate these 29-byte segments to reconstruct the whole message.

**Note 3:** Data transmitted with a GPIB write command can be larger than the 29-byte Transmit Buffer and so will have to be sent in segments. The application has to break up the outgoing message into segments of 29 bytes (or less) and send them one at a time.

## 4.2.3.11 CNA Module - Analog/Digital I/O, with PID

The 2-byte analog values are stored least significant byte first. They will be either 2's complement or unsigned integers, depending on the configuration of the CNA-module.

The analog output channel is 16-bits and can be set to a maximum of:

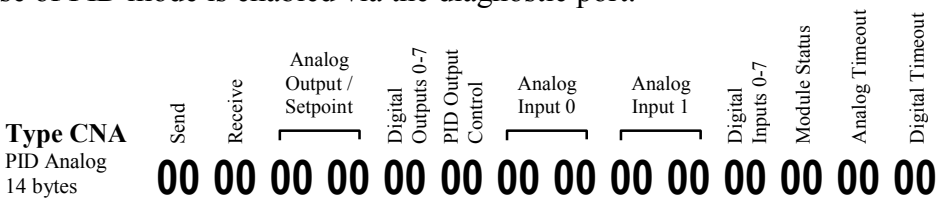
bipolar: -32000 to +32000 (full scale)
unipolar: 0 to +64000 (full scale)

The analog input channels are 16-bits and will be set to a maximum of:

bipolar: -32000 to +32000 (full scale) (some over-range is possible)
unipolar: 0 to +64000 (full scale) (some over-range is possible)

Digital input channels are independent of the digital output channels.

Use of PID mode is enabled via the diagnostic port.




**Analog Output / Setpoint for PID control**

This location in the Loop Controller dualport has two functions, depending on whether the module is running PID control or not.

**Analog Output**

If the module is used as a simple I/O module then this location is used to send the value to generate the output voltage or current directly.

**Setpoint for PID control**

If the module is set to operate as a PID controller then the output value is not sent directly. Although what is sent from this dualport location appears to be an output value the module actually uses it as a desired input value, and does everything possible (by altering its DAC output) to make its read-back (analog input-0) equal this setpoint value.

As an example, suppose a transducer probe (which returns a 0 to 10 volt signal) is attached to analog input channel 0. If the controlled variable is to be half full scale, a request for 5.0V is written into the setpoint location. The module then alters its actual DAC output in whatever way it can to achieve the value 5.0V at its analog input channel 0. The Loop Controller has no direct control over the DAC output in PID mode. Note that because analog values in dualport are defined in counts, not volts the 5.0V request is actually set in dualport as either 16,000 or 32,000 depending on whether analog input channel 0 is bipolar or unipolar respectively.

An important note here is that whether the setpoint data is considered to be bipolar or unipolar is determined by the configuration of analog input channel 0 inside the CNA module, not the analog output channel.

**Digital Outputs**

The eight digital outputs are controlled by one byte in the dualport. Each bit in the byte corresponds to a channel.

The polarity of the bits is programmable on a channel by channel basis by using the Group3 diagnostic port - under the **System / Digital / Output Polarities** selection. The default setting for polarities is LOW but can be changed to achieve an inversion. If the polarity is selected as LOW, a 1 in a bit location will cause the corresponding channel's relay contact to close.

**PID Control Byte**

**Reset - (bit-0)**

This is used to zero the analog output and clear all PID calculations.

If bit-0 is set to 1 then all of the following occur:
- the module's analog output is set to zero.
- the integral accumulation is cleared to zero.
- all PID calculations are stopped.

These actions can also be invoked by any one of the following:-

- Digital-input-5, if it is enabled to function as PID reset.
- Pressing [R] on the diagnostic port: I/O monitoring screen.
- Simulating Digital-input-5, and setting it to value 1, if it is enabled as above.

**Hold - (bit-1)**

This is used to hold the analog output and PID calculations.

If bit-1 is set to 1 then all of the following occur:
- the module's analog output is held at its last value.
- the integral accumulation is halted.

These actions can also be invoked by any one of the following:-

- Digital-input-6, if it is enabled to function as PID hold.
- Pressing [H] on the diagnostic port: I/O monitoring screen.
- Simulating Digital-input-6, and setting it to value 1, if it is enabled as above.

**Analog Inputs**

Channel 0 and channel 1 are identical.
If the module is set to operate as a PID controller then channel 0 is used for feedback within the controlled loop.

**Digital Inputs**

The eight digital inputs are read as one byte in the dualport. Each bit in the byte corresponds to a channel.

The polarity of the bits is programmable on a channel by channel basis by using the Group3 diagnostic port - under the **System / Digital / Input Polarities** selection. The default setting for polarities is LOW but can be changed to achieve an inversion. If the polarity is selected as LOW, and if current is flowing in an input opto-coupler, a 1 will be read in the bit location corresponding to that channel.

**Module Status Byte**

This byte is returned by the CNA. Various bits within the byte are used to indicate certain conditions that a controlling application should know about. All bits indicate their described status if set. This enables an application program to test the value of the entire byte, and proceed no further if it is zero.

**DAC overflow or underflow - (bit-0)**

Bit-0 is used to indicate to the application program that a number was passed to the module for analog output which, after application of the calibration coefficients, produced a number which could not be represented with 16-bits. The module adjusted the resulting value to the most +ve or -ve value possible (as appropriate) and set this overflow/underflow status flag.

Note that if the CNA module is operating in PID mode, this status flag is not necessarily indicative of a fault condition. If the required output range is within the capability of the module then momentary attempts to produce analog output values outside the boundaries described above are quite normal.

**Channel-0 and Channel-1 ADC overflow or underflow – (bits 1 &2)**

bit-1 overflow / underflow: analog input channel 0
bit-2 overflow / underflow: analog input channel 1

If one of these bits is read as a 1, it means that a number outside the range –32,000 to +32,000 (bipolar range) or 0 to 64,000 (unipolar range) was generated by the analog input componentry on the module. It doesn't necessarily mean that the reading is no good, but that it is outside the specified range of the product.

**PID calculation overflow – (bit-3)**

Bit-3 can be set for one of two conditions:

- It can indicate that in calculating the PID output an integral number was created that exceeded the specified limits. If the PID integral value exceeds the upper or lower limit, as configured via the diagnostic port terminal, the integral is clamped to either the upper or lower limit (as appropriate) and bit-3 is set.
- If the module calculates a PID overall analog output correction signal that can’t be represented with 16-bits then the signal is adjusted to the most +ve or -ve value possible (as appropriate) and again status bit-3 is set.

Note that if the module is operating in PID mode, this status flag is not necessarily indicative of a fault condition. If the required output range is within the capability of the

module then momentary attempts to produce analog output values outside the boundaries described above are quite normal.

**Simulations active – (bit-4)**
Bit-4 indicates that one or more simulations are active at the module. If this bit is set it means that someone, via the diagnostic port terminal, has caused the module to simulate one or more of its input or output values. This is important information to the control application, as any simulations could mislead an application and prevent normal control.

**Board needs calibrating – (bit-5)**
Bit-5 indicates that the board needs calibrating. This bit will be set if the calibration data is deliberately cleared within the module.

## 4.3 Diagnostic Port / Window (Over-The-Loop)

The over-the-loop diagnostic port facility provides all the features and options that are available when plugging a terminal into the RS232 "Diagnostic" port socket on the front of a Group3 DI. The same data stream that would otherwise be directed to/from a local physical terminal is transmitted over the loop to/from a data area in dualport RAM. The terminal then becomes a window on the computer screen and the connection to the remote DI is via the fiber optic loop.

To set up an over-the-loop diagnostic port to a DI, an **I/O definition** needs to be set as follows:-

| | |
|---|---|
| DI address | is set to the address of the DI to talk to |
| Board Number | is 0, for the processor board |
| Board type | is set to 6, as per a serial communications card |
| Offset to Data | is set to point to the data area higher up in the dualport RAM |

The **data area** takes the same format as that of a type F board in General Serial mode except that the port number and port type locations are now reserved and should be set to zero.


Multiple over-the-loop diagnostic ports can be set up for more than one DI at a time and can operate simultaneously with regular board I/O communications on a loop. The over-the-loop diagnostic port does need the loop to be complete and powered up. Once communication is established it is a very useful tool to allow diagnostic and configuration operations to be undertaken on a DI that is at high voltage, or is mounted in an inaccessible place.

Driving the over-the-loop diagnostic port facility is almost exactly the same as operating the serial (type F) board (see section 4.2.3.6). Text is placed in the Send Buffer then the Send Count location is set to the number of bytes to send. However outgoing messages usually correspond to keyboard input and are quite short, perhaps one keyboard character per message. On the other hand, the messages sent back from the DI for display in a window are usually a full screen's worth of text and these will be received in 29-byte segments. The application has to concatenate these 29-byte segments to reconstruct the whole message.

**Special character sequences**

For a diagnostic port (either over-the-loop or via the socket on the front of the DI), the serial data stream from the Device Interface will include two ANSI escape sequences. They are:

| | |
|---|---|
| **<Esc> [ H** | the screen home command |
| **<Esc> [ 2J** | the clear screen command |

where <Esc> is ASCII character 27 (1B hex)

If the programmer is writing their own terminal window to display the data stream they will need to interpret these character sequences appropriately. These character sequences may also be broken across 29-byte message segments.

**Effect on system performance**

One over-the-loop diagnostic port is required for each DI that is to be talked to remotely and each one of these consumes 8+64 bytes of dualport RAM for an **I/O definition** and **data area** as well as some message time on the loop. Having many over-the-loop diagnostic ports open simultaneously will impact on control system performance so it is better to shut them down when they are finished with. Some early Loop Controllers with 1024-byte dualports may run out of space if configured for many I/O boards plus over-the-loop diagnostic ports.

**Reconfiguring a Loop to change / delete a diagnostic port definition.**

One solution to the resource problem is to reconfigure dualport RAM to exclude over-the-loop diagnostic ports when they are not required. Another is to stack the diagnostic port I/O definitions at the top end of the I/O Definition Area, beyond the I/O board definitions. Then over-the-loop diagnostic ports can be disabled and enabled simply by reloading the number of I/O definitions (location $03_h$) in the dualport system area. Yet another method is to have only one over-the-loop diagnostic port and change its configuration so that it is used for one DI at a time on the loop. In all cases, changing the configuration requires the computer to:

1. shut down comm's
2. reload the dualport configuration with changes to the over-the-loop diagnostic port(s)
3. set the System Flag
4. enable comm's

This can happen relatively fast when automated. However note that all of these reconfiguring methods involve stopping and restarting communications on the loop. If control outputs have been set to "zero on a comm's timeout" then this might happen during the reconfiguring and control will be lost. This problem can be avoided by alternatively using the Parameter Tool to access the configuration and set-up data for DIs (see section 4.4 Parameter Tool).

**Switching between an over-the-loop port window and a local hand-held terminal**

A DI will accept keyboard commands from and send screen information to only one diagnostic port at a time. This can either be over-the-loop or via the socket on the front of the DI but not both at the same time. When a DI is first powered on it will communicate with a local physical terminal by default. When an over-the-loop diagnostic port is first enabled it will be necessary for it to send a specific character to the DI to establish that communications path. This character must be one of either:

**<CR>** ASCII character 13 (0D hex)
or **<Esc>** ASCII character 27 (1B hex)

To switch the communications path back to the local physical terminal it will be necessary for an operator to press either <CR> or <Esc> once on its keyboard.

**Resetting a DI processor over the loop.**

If an over-the-loop diagnostic port is defined for a DI then using it to send the text "***RESET***" will always cause the DI processor to reset. A reset can also be forced by sending the **ZP** command from within the System menu of the diagnostic port.

## 4.4 Parameter Tool

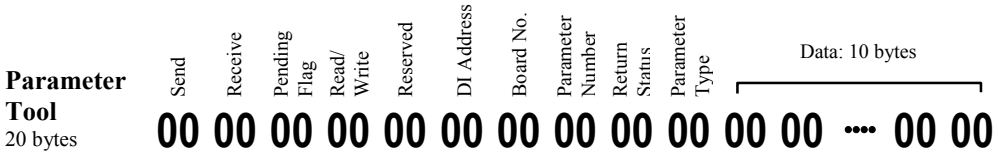
The Parameter Tool is a feature that can be used to access all the configuration and set-up information for DIs over the loop. This is available with software version 5.0 or later of LC, DI and CNA software. All the range settings, polarities and calibration data can be read over the loop and backed up for future reference. This data can then be used to greatly speed up the restoration or replacement of a faulty DI. It can also be used during program initialisation to check that all required DIs and I/O boards are present on the loop and are configured correctly.

The Parameter Tool requires only one **I/O definition** and one **data area** to access all the DIs on a loop. It has very little impact on the performance of a system, unlike multiple over-the-loop diagnostic ports which consume both loop time and memory space. When not being used to read or write parameters it consumes no loop time at all.

To set up a Parameter Tool, an **I/O definition** needs to be set as follows:

| | |
|---|---|
| DI address | is set to FE hex, to create a Parameter Tool |
| Board Number | is 0 |
| Board type | is set to 13 |
| Offset to Data | is set to point to the data area higher up in the dualport RAM |

The **data area** occupies 20 bytes, and takes the following format:




**Pending Flag**

This flag is used to ensure that only one Parameter Tool operation is going on at any one time. When the application wants to read or write a parameter, it loads the appropriate data to this I/O area, then sets this flag to 1. The Loop Controller will then send the message out on the loop. When the data or acknowledgment is returned from the DI, the Loop Controller will clear this flag.

This flag is set to 1 by the application to initiate a parameter operation. It is cleared to 0 by the LC to indicate operation complete, (no pending operation), or if the operation timed out before completion (see the note about **Timing** near the end of this section).

**Read / Write**

| value | meaning |
|---|---|
| 0 | read parameter from board |
| 1 | write parameter to board |

**Reserved**

Set to 0.

**DI Address**

Address of the DI that houses the board of interest.
Value 0..15 (0..F in hex), as set on the address switch on the DI.

**Board Number**

Number of the board within the DI that is of interest. Value 0..3:

| | |
|---|---|
| 0 | refers to the DI processor board. |
| 1,2,3 | are the I/O boards in a DI, as set by the board address jumpers. |

**Parameter Number**

Number of the parameter of interest.
Value 0..n, where n varies according to the type of I/O board.
See appendix A for details of the parameters of each I/O board.

**Return Status**

Used to return acknowledgment that the parameter operation was accepted, or to indicate the nature of the error if the parameter operation was not successful.

| value | meaning |
|---|---|
| 0 | OK, accepted |
| 3 | invalid DI address (outside range 0 to 15) |
| 4 | invalid I/O board number (outside range 0 to 3) |
| 24 | DI not responding (no DI at this address) |
| 27 | non existent board (this board number is not present in the addressed DI) |
| 28 | invalid command (probably old s/w in the DI - doesn't support parameters) |
| 101 | invalid parameter number (number exceeds maximum for this board) |
| 102 | no parameter at this number (reserved location) |
| 103 | parameter is not allowed to be read (returned if attempting to read) |
| 104 | parameter is not allowed to be written (returned if attempting to write) |
| 105 | invalid parameter data (when writing) |
| 106 | invalid parameter type for parameter number (when writing) |
| 107 | secure access required (password access levels are needed) |

**Parameter Type**

Used to validate the data transfer and indicate the type/size of the data returned.
Set by the application if writing a parameter to an I/O board.
Set by the Loop Controller if a parameter is returned from an I/O board.

| value | meaning |
|---|---|
| 0 | no type |
| 1 | 8-bit signed integer |
| 2 | 8-bit unsigned integer |
| 3 | 16-bit signed integer |
| 4 | 16-bit unsigned integer |
| 5 | 32-bit signed integer |
| 6 | 32-bit unsigned integer |
| 7 | 32-bit floating point (IEEE 754) |
| 8 | 64-bit double floating point (IEEE 754) |
| 9 | text |

**Data Value**

10-byte block for the data value of the parameter.
For the numerical data types listed before only the first 1, 2, 4, or 8 bytes are relevant. The remaining bytes in the block, that are not used by the particular data type, are not defined, and will probably contain random junk from previous parameter operations.

For the text data type there is a format for the data area:-
The first byte is the length of this message, and can take the value 1 to 9.
The subsequent bytes are the message itself, to a maximum of 9 characters.
As for the numerical data type, the unused portion of the text data area will probably contain random junk from previous parameter operations.

To access a particular parameter on a particular board of a DI, set up the data area with Read/Write, DI Address, Board Number, Parameter Number, and, if writing a parameter, Parameter Type and Data. Lastly, set the Pending Flag.

Parameter Numbers and Parameter Type are listed in Appendix A, for all boards.

**Secure Parameters**

Some parameters are 'read only' - such as the board type and serial number, and can not be set. Other parameters, such as factory entered calibration data, are specific to each individual board. These parameters can be read out from the board in the normal manner. However these are secure parameters and to write to them, the application must first write the correct serial number to the serial number parameter location for that I/O board. If the serial number written matches the serial number already in the board, then the next **one** write to a secure parameter for that board will be allowed. If the serial numbers do not match, the write will not be actioned.

So for example, to restore all the calibration data for a type C board the sequence would be:-
write s/no., write chan-0/range-1 zero, write s/no., write chan-0/range-1 cal, write s/no., .....etc

In this way it should be possible to avoid accidentally writing a set of calibration figures to the wrong board.

**Timing**

The Parameter Tool uses one **I/O definition** and one **I/O Data Area**, and operates in a similar manner to an I/O board. If it is the only I/O definition on the loop then parameter reads will take about 2ms. If there are I/O boards defined then the Parameter Tool will share loop time with them. Parameter writes generally take much longer than reads. If the written value is the same as the stored value then it will take only a few milliseconds, but if there are large changes then it could take a few hundred milliseconds per parameter write.

If a DI does not respond to a Parameter Tool operation the Loop Controller will re-send the request up to 10 times. If there is still no response the LC will store an error value in the Return Status byte of the Parameter Tool Data Area, and clear the Pending Flag. Parameter writes will in general be slower than reads, so extra time is allowed between retries. Ten retries of a write could take over a second before the Loop Controller gives up, posts an error and releases the Parameter Tool for another operation. Timeout for a read operation is much faster.

**Reading All Parameters**

One application of the Parameter Tool is for all the parameters for all the boards on a loop to be read out and stored in a file. Then, if there is a problem, the settings and calibrations etc can be restored to the original I/O boards. If a board needs to be replaced, then all the user settings (ranges, polarities etc) can be downloaded to a new substitute board, allowing quick field replacement.

One mechanism for reading out all the parameters for all boards is to address each board in turn, and read parameters from it in sequence until the LC stores a value of 101 in the Return Status byte of the Parameter Tool data area. A value of 101 indicates that the parameter number requested was greater than the maximum number for this board - ie. all parameters have been read out for this board. Then move on to the next board, and so on around the loop. The parameters should be logged to a file as they are read out.

It is even possible to scan a whole loop blindly, without knowing how many DIs or boards there are. Simply request parameters for all possible DIs (0 to 15) and all boards (0 to 3). Skip the DI number when status 24 is returned ( DI not responding), and skip the board number when status 27 is returned (non existent board).

## 4.5 Loop Controller Dualport RAM Capacity

Loop Controller (LC) cards have 2048 bytes of dualport RAM for each fiber optic communication loop. The preceding sections describe in detail how this memory is assigned. When a loop of Device Interfaces is being designed, it is necessary to check that the dualport RAM space required by the system does not exceed the capacity available.

The only time this is likely to happen is if many diagnostic port windows are opened up at the same time. It can also occur with early versions of the PC Loop Controllers which had only 1024 bytes of dualport RAM.

If the total exceeds the capacity of the LC in use, the loop can be split into two or more loops, as necessary. Multiple loops can be installed in all of the supported computer platforms either by using more than one Loop Controller card or one or more multi-loop LC cards.

## 4.6 Analog Channels

The analog input and output channels on the various I/O boards are either 14-bits or 16-bits. In all cases their data is contained in 16-bit integers. For the purposes of this discussion, the 8-bit analogs on the G-board are not considered here.

The 2-byte analog locations in dualport RAM hold values in binary bit-counts, not volts. A bipolar analog value will be stored as a 2's complement integer, unipolar as an unsigned integer.

Note that the numerical range used for 14-bit and 16-bit analog channels is somewhat less than the maximum allowed by 14 / 16-bits. Eg for a 16-bit unipolar analog output the range of possible data sent over the loop is 0 to 64,000 counts, not the full 0 to 65,535 where 64000 represents full scale output. The difference between 64000 and 65535 gives the module processor some headroom to apply calibration coefficients without causing a 16-bit overflow. For example +5V for a 16-bit bipolar channel which has a full scale of 10V will be stored as $16{,}000_{10}$ where full scale of –10V to +10V is represented as –32,000 to +32,000. If this example had been a 14-bit bipolar channel with a full scale of 10V, +5V would be stored as $4{,}000_{10}$ where full scale of –10V to +10V is represented as –8,000 to +8,000.

The Group3 diagnostic port can be made to display analog values in bit counts rather that volts. From the main menu select Display Options then choose Decimal to display the analog quantities as decimal number counts.

Prior knowledge of the analog channel configurations is required by the control program running on the computer. For each analog channel, the full scale input/output voltage, the full scale count and whether it is bipolar or unipolar must all be known before sensible output values can be stored in or read from dualport. This information is published in the “Group3 Control User’s Manual” for all board types/categories and range selections. In addition, the full scale input/output voltage and range selection can be uploaded for any board/channel using the Parameter Tool if the DI/CNA/LC software is version 5.0 or later.

**Calculating Analog Output Values**

$$\text{output count} = \frac{\text{desired output voltage}}{\text{full scale output voltage}} \times \text{full scale count}$$

for example: +7.5V for a 16-bit bipolar channel which has a full scale of 10V

$$\text{output count} = \frac{7.5}{10.0} \times 32{,}000 = 24{,}000$$

for example: -4.0V for a 14-bit bipolar channel which has a full scale of 10V

$$\text{output count} = \frac{-4.0}{10.0} \times 8{,}000 = -3{,}200$$

The 4mA to 20mA range on the CNA is a little different because of the 4mA offset. A count of 0 represents 4mA and a count of 64,000 represents 20mA.

$$\text{output count} = \frac{\text{desired output current} - 4.0}{16.0} \times 64{,}000$$

for example: 12.0mA

$$\text{output count} = \frac{12.0 - 4.0}{16.0} \times 64{,}000 = 32{,}000$$

**Calculating Analog Input Values**

$$\text{input voltage} = \frac{\text{input count}}{\text{full scale input count}} \times \text{full scale input voltage}$$

for example: 24,000 for a 16-bit unipolar channel which has a full scale of 50mV

$$\text{input voltage} = \frac{24{,}000}{64{,}000} \times 50 = 18.75\text{mV}$$

for example: -24,000 for a 16-bit bipolar channel which has a full scale of 10V

$$\text{input voltage} = \frac{-24{,}000}{32{,}000} \times 10 = -7.5\text{V}$$

The control software running on the computer will make short work of these sorts of calculations and it may even come prepackaged with options for sorting out this type of scaling.

## 4.7 Interrupts

Interrupts typically signal the arrival of new data in dualport RAM but are only available on PCI Loop Controllers. If interrupts are enabled, one will be generated immediately after new data has been placed in dualport RAM for any of the I/O boards, over-the-loop diagnostic ports or the Parameter Tool. Interrupts also occur when data in a 29-byte Send Buffer has all been sent and its Send Count has been cleared to zero. With the exception of the Parameter Tool and 29-byte Receive Buffers, the newly arrived data has to be different to the previous contents in dualport RAM before the Receive Data Flag changes state and an interrupt is generated etc. This greatly reduces the number of interrupts that the computer has to service. The interrupt indicates to the computer that it should examine the Last I/O Def' Updated location, generally to retrieve the new data that it refers to, but in the case of 29-byte Send Buffers to load in the next part of the message to be sent. The full sequence when new data arrives is:

1. The overall Receive Data Flag is driven even (bit-0 is cleared).
2. The new (different) data is placed in the I/O Data Area (or a Send Count is zeroed).
3. The overall Receive Data Flag is driven odd (+3).
4. The corresponding I/O Def' number is stored in the Last I/O Def' Updated location.
5. The interrupt signal occurs.

**Note 1:** For 29-byte Receive Buffers (either type F boards in General Serial mode, type K boards or over-the-loop diagnostic ports), if new data is placed in the Receive Buffer at the same time as the Send Count for the corresponding Send Buffer is cleared to zero, only one interrupt will occur.

**Note 2:** When data is placed in dualport RAM for an I/O board, all of the data (eg all eight analog values on a type C board) is stored before steps 3, 4 and 5 occur.

**Note 3:** For type F boards operating in DTM Loop mode, there is only one interrupt: at the end of data placement for all of the DTMs on the loop. However, type F boards operating in DTM Loop mode also have per-DTM Receive Flags which are driven around the data placement of individual DTMs.

**Note 4:** For type G boards (stepper motor), there is only one interrupt: at the end of data placement for all four stepper motors. However, type G boards also have per-motor Receive Flags which are driven around the data placement of individual stepper motors.

**Note 5:** A Loop Controller can generate interrupts quite rapidly and they can occur in bursts of two or three in quick succession if Fast LC to DI mode is in use. Because of this it is recommended that, if using interrupts, interrupt service routines should be kept small and the computer should be fast.

# 5 LC to LC Communications

## 5.1 Introduction

This use of the Group3 Loop Controller (LC) allows two computers to share information, using fiber optic cables. Two LCs (one in each computer) can continually exchange the data held in defined areas of their dualport RAM. If one computer writes some data into a transmit area in its LCs dualport, this data will automatically appear a few milliseconds later in the corresponding receive area of the LCs dualport in the other computer. Data will be simultaneously transferred in the other direction as well.

The LCs continuously send each entire transmit area to the corresponding receive area of the other LC in the other computer. Thus the computers are not directly involved in the communication with each other; they simply access their dualport RAM, while the LCs look after all communication overhead. In this mode the LCs are also completely independent of any other Group3 Control hardware.

LC to LC communication enables two computers to share information, even if they are at very different electrical potentials, because of the use of fiber optics.

The computers could be two of the same type (two PCs, or two VME crates talking to each other), or of different types (a PC talking to a VME crate, or a VME crate talking to an STD crate, for example). LC to LC communication can thus provide an isolated comm's channel between different computer platforms.

It is recommended that the I/O Data Area of each dualport RAM be partitioned into two areas: a transmit area and a receive area. These can be of equal or different sizes, but the receive area in one computer must be the same size as the corresponding transmit area in the other computer. In fact any number of transmit and receive areas are permissible, but defining more than one of each may slow down data exchange.

The communications on the fiber optic cables runs at 1.152Mbaud using SDLC. This is a fast, robust communications protocol which includes CRC checksums sent with each data packet. If the receiving LC detects a corrupted message it will discard it.

## 5.2 Dualport RAM Space Allocation / Initialisation

Refer to the memory allocation diagram on page 5-3 when reading the following text.

The dualport RAM (shared memory) is divided into three areas:

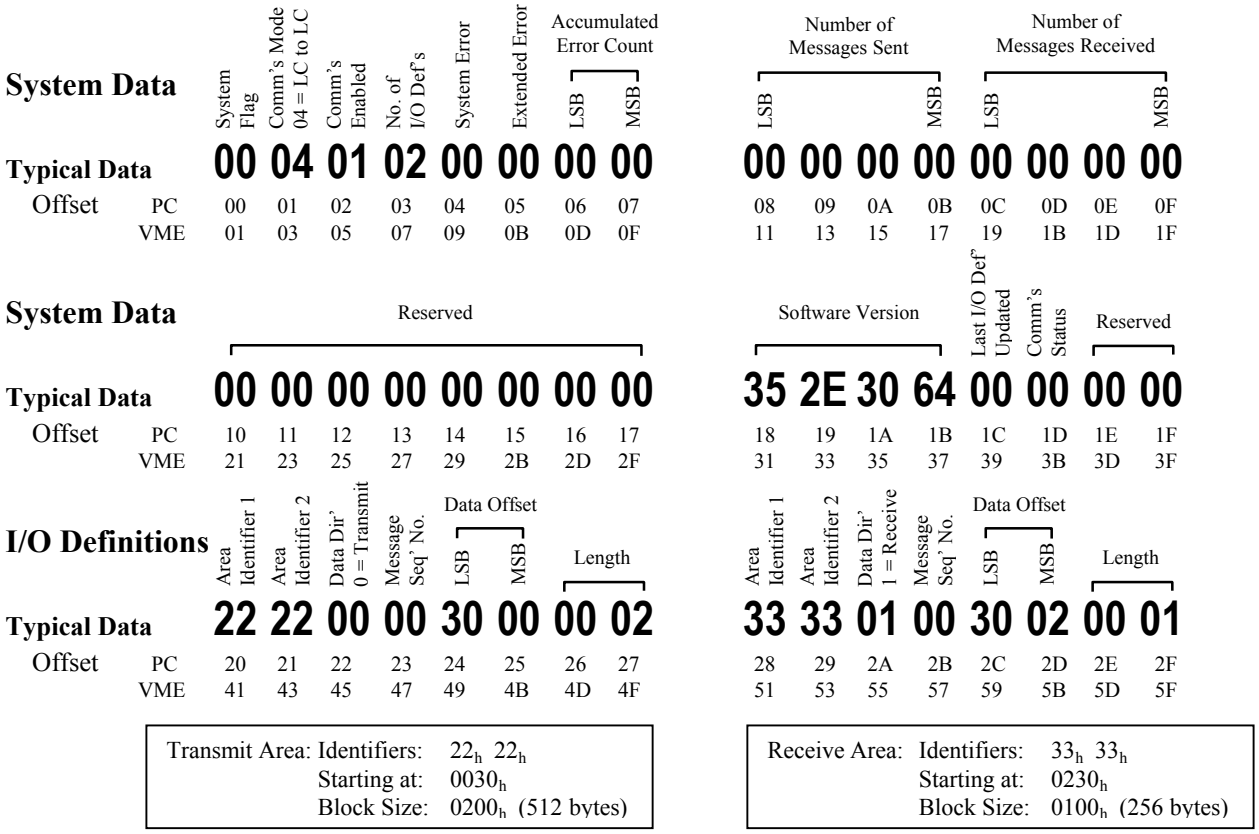
- **System Data Area**: used for storing system dependent parameters.
- **I/O Definition Area**: used for defining the I/O areas used in the system.
- **I/O Data Area**: used for storing the data sent to, and received from, the other LC.

Each computer must set up and completely initialise the dualport RAM for the System Data Area and the I/O definitions. An I/O Data Area of the appropriate size must be allocated somewhere in the remaining dualport RAM for each I/O definition and its address offset must be stored in that I/O definition. This is very similar to the set-up required for a standard loop of DIs (chapter 4).

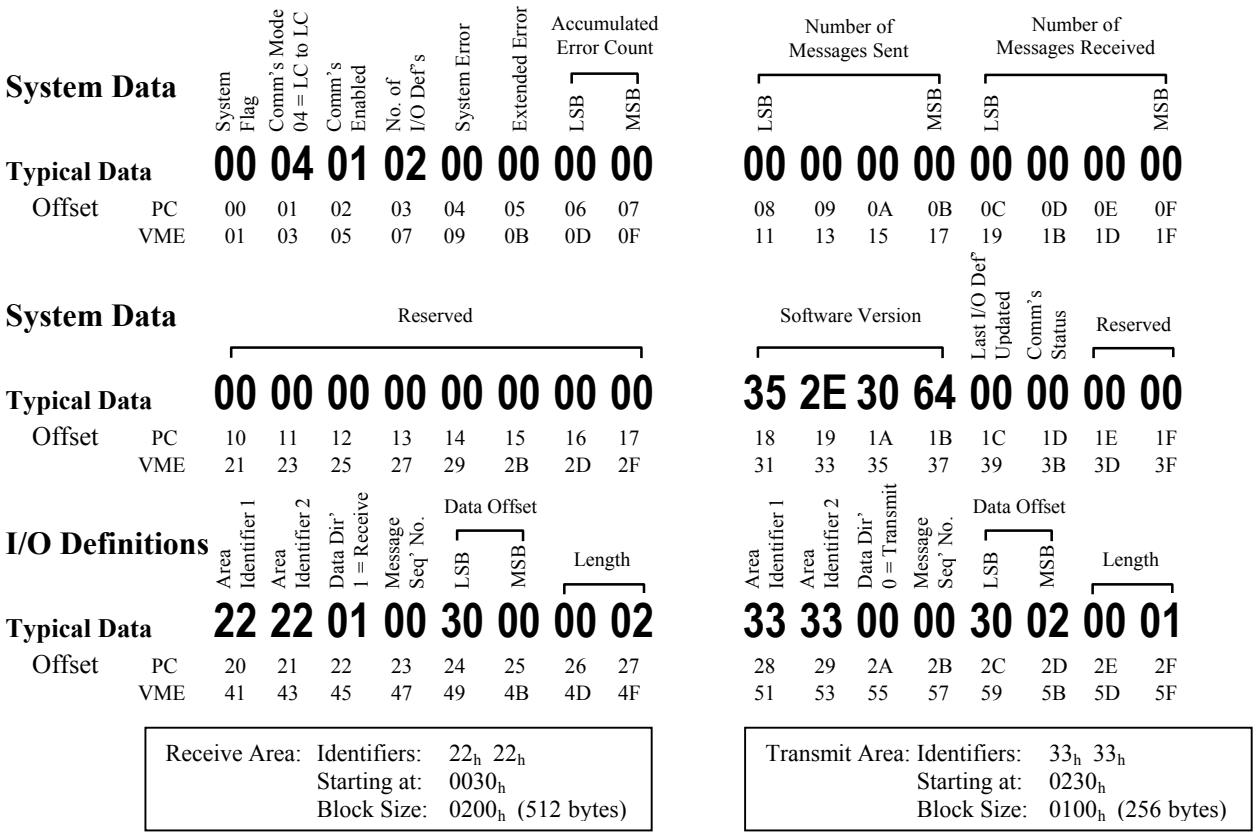
Full initialisation of the I/O Data Areas is optional and will depend on their format and expected use. However the sequence byte (and acknowledge byte if used) as well as local copies of sequence bytes (outside of dualport), should all be initialised to the same value, say 0, to avoid initial transmission/reception of garbage. If the data message areas contain a data length value then these should also be initialised to 0 at this time.

# Memory Allocation Map: Two LCs Exchanging Data

Machine A: sending 512 bytes to machine B, receiving 256 bytes from machine B




Machine B: receiving 512 bytes from machine A, sending 256 bytes to machine A



## 5.2.1 System Data Area

The first 32 bytes of the dualport RAM contain system configuration information. This is used as follows:

| | bytes | |
|---|---|---|
| System Flag | 1 | set to 1 to load new configuration |
| Communication Mode | 1 | set to value 4 for LC to LC comm's |
| Communications Enabled | 1 | 1 = enabled, 0 = disabled,<br>3 = enabled with interrupts (for PCI LCs) |
| Number of I/O Definitions | 1 | recommended 2: one Tx, one Rx |
| System Error | 1 | |
| Extended Error Information | 1 | (also actual comm's state flag) |
| Accumulated Error Count | 2 | |
| Number of Messages Sent | 4 | |
| Number of Messages Received | 4 | |
| Reserved | 8 | |
| LC Software Version | 4 | ASCII string written by LC processor |
| Last I/O Def' Updated | 1 | the I/O def' number starts at 1 |
| Comm's Status | 1 | 1 = communicating |
| Reserved | 2 | |

**System Flag**

The System Flag has two valid states: clear = 0, and set = 1.

- The computer sets this flag to 1 tell the LC processor to load the new set-up that has been placed in dualport RAM.
- The LC processor reads the new set-up information then clears this flag to 0.

Before setting this flag, the computer must set up the dualport RAM so that the LC knows the details of the data blocks it is expected to exchange data with on the remote LC. When this setting up process is complete the computer sets the System Flag. The LC then examines and processes the dualport RAM containing the I/O definitions, checking the validity of the configuration as it goes. Once complete, the LC clears the System Flag to indicate to the computer that the set-up stored in dualport RAM has been used.

**Communication Mode**

This byte tells the Loop Controller what communication mode to use.
For LC to LC communications, this byte should be set to 4.

**Communications Enabled**

Three allowed states:

0 disabled
1 enabled
3 enabled, with interrupts (PCI LCs only).

This byte enables or disables communication between this LC and the remote LC. If the control program is shut down it is recommended that it should first clear this location.

If the configuration in dualport RAM is to be changed, it is important that communications are shut down first. It is not enough just to clear this location to 0 however as the Loop Controller will continue to communicate for a short period until it checks this location. Once this location has been cleared, the computer should either just wait for a small period of time, say 200ms minimum or wait for the Comm's Status (location $1D_h$) to change to 0. See also note 11 in section 7 of this manual.

Only the PCI Loop Controller supports interrupts but enabling this feature for an ISA Loop Controller will do no harm.

A few milliseconds after communications have been enabled, the Loop Controller will begin exchanging data blocks with the other LC. For most applications it is important that the data areas in dualport RAM are set-up with sensible data before allowing them to be sent to the other LC.

**Number of I/O Definitions**

The value here indicates the number of I/O definitions in the I/O Definition Area of the dualport RAM. It is recommended that this be set to 2, one transmit area and one receive area, though as noted in the introduction the protocol does allow any number of data areas to be defined if necessary, up to a maximum of 60.

**System Error**

The errors reported here fall into two main categories. The first are set-up errors, those that the LC will detect and report when processing the new set-up that has been placed in dualport RAM. The second category are communications errors, those which are detected while the LC is exchanging data with the other LC.

If a communication error occurs, the LC will store the error number in the System Error location and then continue with communications. It is the responsibility of the computer to periodically check the System Error location for errors, and to clear the Error byte after examining it. Until the computer has done this, the LC will not report new errors.

For some errors, more information can be found in the Extended Error Information field.

The list of possible error numbers follows:

**Set-Up Errors**

| **Error Description** | **Error Number** Hex. | Decimal | | |
|---|---|---|---|---|
| Invalid Communication Mode | 01 | 1 | | |
| Too Many I/O Definitions | 02 | 2 | | |
| Invalid Area Identifier #1 | 03 | 3 | * | |
| Duplicated I/O Area Address | 05 | 5 | * | (see note following) |
| Invalid Data Direction | 06 | 6 | * | |
| Overlapping Memory Allocation | 0C | 12 | * | |
| Out of Dualport RAM | 0D | 13 | * | |

**Communication Errors**

| Error Description | Error Number Hex. | Decimal | |
|---|---|---|---|
| LC Not Responding | 18 | 24 | |
| Bad SDLC Packet | 19 | 25 | s |

***** Indicates errors which have the index number of the offending I/O definition stored in the Extended Error Information field. Definitions are assigned a sequential index number, starting with 1 for the first definition in the definition area.

**s** Indicates errors which have a status byte stored in the Extended Error Information field.

**System Error Notes:**

**$05_h$ Duplicated I/O Area Address**

Two definitions can have the same identifiers (ID1 and ID2) provided they have different data directions.

**$0D_h$ Out of Dualport RAM**

The indicated I/O definition refers to a data area which extends beyond the end of dualport RAM.

**$19_h$ Bad SDLC Packet**

The status information for this error will be reported in the Extended Error Information field:

Bit-2 set if CRC error detected
Bit-3 set if overrun error detected
Bit-4 set if frame with residue bits detected
Bit-5 set if frame with abort end detected
Bit-6 set if short frame detected
Bit-7 set if end of receive frame detected

This error typically occurs as a result of damaged or loose communication fibers and its exact cause as determined by the Status byte is usually random. Interpretation of this byte is therefore not generally helpful.

**Extended Error Information**

Contains further error information as described before.

Also, if the system error is 0, this location is used as a flag to indicate the actual state of communications:

1 communicating OK
0 communications shut down

This function is now superseded by the comm's status location.

**Accumulated Error Count**

This location keeps a count of the number of errors which have occurred. It is stored as a 2-byte unsigned integer.

**Number of Messages Sent**

This is a count of the total number of messages which have been sent to the other LC, stored as a 4-byte unsigned integer. Because the transmit data areas are being continuously transmitted, this counter will increment rapidly and has no relation to the number of *different* messages sent as determined by the message protocol.

**Number of Messages Received**

This is a count of the total number of messages which have been received from the other LC, stored as a 4-byte unsigned integer. Because the receive data areas are being continuously filled with data sent from the other Loop Controller, this counter will increment rapidly and has no relation to the number of *different* messages received as determined by the message protocol.

**Software Version**

The processor on the LC card writes its software version into this location on startup. It is stored as a 4-byte ASCII string of the form "5.0d".

The computer can read this location to verify that an LC is present, and that the LC processor has started correctly. It can also be read out by the computer and used for controlling the use of Loop Controller features which were introduced at specific software versions (see Appendix B).

Note: the suffix for software versions can also be the ' ' (space character – 20 hex). As a version, this precedes 'a', 'b', 'c' ...

**Last I/O Def' Updated**

As the LC processor updates a block of I/O data from information brought in from the other LC, it stores the number of the corresponding I/O definition in this location. Note that the numbering of I/O definitions starts at 1.

By monitoring this location the control software can easily be pointed to the newly arrived data, speeding up its retrieval. The user can clear this location at will, to assist in detecting changes.

**Comm's Status**

This byte indicates when the LC is communicating. It will be set to 1 a few ms after the Communications Enabled flag is set and will be cleared to 0 a few ms after the Communications Enabled flag is cleared. When shutting down the LC communications to change the dualport configuration, it is important to wait until this location is cleared. This location has been used for this purpose from LC software version 4.3a and onwards. See also note 11 in section 7 of this manual.

## 5.2.2 I/O Definition Area

This section of dualport RAM is used to define each transmit and receive data area used for LC to LC communications. The memory area allocated to I/O definitions forms a contiguous block starting immediately after the 32 bytes of the System Data Area, with 8 bytes allocated to each I/O definition.

The format for each I/O definition is as follows:

| | **bytes** | |
|---|---|---|
| Area Identifier #1 | 1 | any character to label the data area |
| Area Identifier #2 | 1 | any character to label the data area |
| Data Direction | 1 | 0 to transmit, 1 to receive |
| Message Sequence No. | 1 | must change with each message sent |
| Offset to Data | 2 | offset from start of dualport RAM |
| Length | 2 | number of bytes to be exchanged |

The two Area Identifiers are sent to the other LC, along with the data, so that the other LC knows where to store that data. Two I/O definitions can have the same Area Identifiers (ID1 and ID2) provided they have different Data Directions.

**Area Identifier #1**

The computer loads this location with an identifier for this data area. It can be any value less than 200 (C8 hex).

**Area Identifier #2**

The computer loads this location with a second identifier for this data area. This location can take any value.

**Data Direction**

The computer sets this location for the direction of data flow for this I/O Data Area.
Valid values are:

| | | |
|---|---|---|
| 0 | Transmit | this area is to be sent to the other LC |
| 1 | Receive | this area is to be used to receive data from the other LC |

**Message Sequence Number**

This location is used during communications. The computer places a sequence number in this location after placing new data in the corresponding I/O Data Area. The sequence number must be different to the previous one used.

The sequence number ensures uncorrupted reception of a data block, and if necessary, ensures that each received data block is read only once. To this end it can be used to form a sequence / acknowledge protocol. The recommended protocol is described in section 5.3 Message Protocol.

**Offset to Data**

This contains the offset from the start of the dualport RAM to the start of the I/O Data Area for this I/O definition. This is a 2-byte unsigned integer, stored least significant byte first.

The computer must set up these offsets taking into account the number of bytes each I/O definition requires for its data area. The data areas must not overlap. To conserve dualport RAM space the data areas are usually set to be following on from each other, but if memory requirements are modest, this is not essential.

Note that each data area can be positioned anywhere within the remaining free space of dualport RAM. One convenient algorithm is to stack the data areas downwards from the top of dualport. The remaining free space is then the gap between the last I/O definition added and its corresponding data area.

**Length**

The computer sets this location to the length of the data area - the number of bytes that are to be sent or received. It is a two byte unsigned integer, stored least significant byte first.

The data area must be set one byte longer than required for the actual data. The first byte in each data area is reserved for a message sequence number as part of the message transfer protocol. It is recommended that the second byte also be reserved for implementing protocol (see section 5.3 Message Protocol). For example if 500 bytes are to be transferred using the recommended protocol then the data area Length should be set to 502 bytes. In addition if variable data-length control is required then adding a length value to the message increases the data area Length to 504 bytes.

- For best efficiency, the data block size should be set between 100 and 980 bytes.
- The maximum block size is 1500 bytes (including the sequence byte).

Several of the I/O definition set-up values for the transmit data area in one LC must match the set-up values for its corresponding receive area in the other LC:

| | |
|---|---|
| Area Identifier #1 | **must match** |
| Area Identifier #2 | **must match** |
| Data Direction | **0** for transmit area **1** for receive area |
| Message Sequence No. | see section 5.3 Message Protocol |
| Offset to Data | can be different for different LCs |
| Length | **must match** |

### 5.2.3 I/O Data Areas

The use of the data areas is almost completely open to the programmer. The exception to this is that the first byte of each transmit and receive data area is reserved for use as a sequence byte by the Loop Controller. The use of this byte is described in section 5.3 Message Protocol.

The size of the data areas is also fairly open. Recommendations on this are made in section 5.2.2 I/O Definition Area - Length.

If the length of data to be sent between Loop Controllers is to vary from message to message then a useful technique is to make the pairs of data areas one or two byes longer and also store a data length (of one or two bytes) at the start of the data message (after the sequence and acknowledge bytes).

## 5.3 Message Protocol

For this protocol to work, transmit and receive data areas (plus corresponding I/O definitions) must be created in pairs in the same dualport. They don't have to be the same size but are used in conjunction, not only for passing data, but also for exchanging sequence and acknowledge bytes to implement the protocol.

The first byte of each data block (transmit and receive) is reserved for use by the Loop Controllers as a sequence byte.

It is recommended that the second byte of each data block (transmit and receive) also be reserved for use by the application to implement an acknowledge byte.

If data transmission is only required in one direction, pairs of data areas should still be created in each Loop Controllers dualport but the redundant transmit/receive pair of data areas could be made very small, say two bytes each: just big enough to hold a sequence and acknowledge byte.

The recommended transmit and receive protocols are described next.

**Transmitting**

- The application software compares the ack byte in the receive block area (second byte) against the transmit sequence number (a copy is in the transmit I/O definition: fourth byte). If they are not the same, the previous message has not been acknowledged by the other computer yet and so it does not process further.
- The application loads the data into the transmit data area.
- The application places a new sequence number in the sequence byte of the transmit I/O definition.
- *Now the LC takes the sequence byte out of the transmit I/O definition and places it in the first byte of the transmit block data area.*
- *Then the LC sends the whole data block to the other LC.*

**Receiving**

- *The LC receives a block of data and stores it in the receive block data area.*
- *Then the LC retrieves the sequence byte from the receive block data area and places it in the sequence byte of the receive I/O definition: fourth byte.*
- The application determines if the sequence byte in the receive I/O definition is new. If not, there is no new data and it does not process further.
- The application retrieves the message from the receive block data area.
- The application takes the sequence byte in the receive I/O definition and places it in the second byte of the transmit block area. This is the acknowledge returning.

## 5.4 Timing

The timing depends on the defined lengths of the I/O Data Areas. All of the transmit data areas are sent as one data packet from the source LC to the destination LC. Once communications are started the LCs exchange information continuously, as fast as they are able and in both directions simultaneously. Some typical timing values are:

| | |
|---|---|
| to transmit 2 kbyte in one direction | 16ms |
| to send 1 kbyte in each direction | 8ms |
| to send 2 bytes in each direction | 1ms |
| to send 256 bytes, and receive 2 bytes | 2ms |

## 5.5 Interrupts

Interrupts signal the arrival of new data in dualport RAM but are only available on PCI Loop Controllers. If interrupts are enabled, one will be generated immediately after a data block has been received and placed in dualport RAM. The newly arrived data does not have to be different to the previous contents in dualport RAM for an interrupt to be generated etc. The interrupt indicates to the computer that it should examine the Last I/O Def' Updated location to retrieve the new data that it refers to. The full sequence when new data arrives is:

1. The data block is placed in the I/O Data Area
2. The message sequence number is transferred to the I/O definition (by the LC)
3. The corresponding I/O Def' number is stored in the Last I/O Def' Updated location
4. The interrupt signal occurs

**Note:** A Loop Controller can generate interrupts quite rapidly and they can occur in bursts if there is more than one receive block per LC. Because of this it is recommended that, if using interrupts, interrupt service routines should be kept small and the computer should be fast.