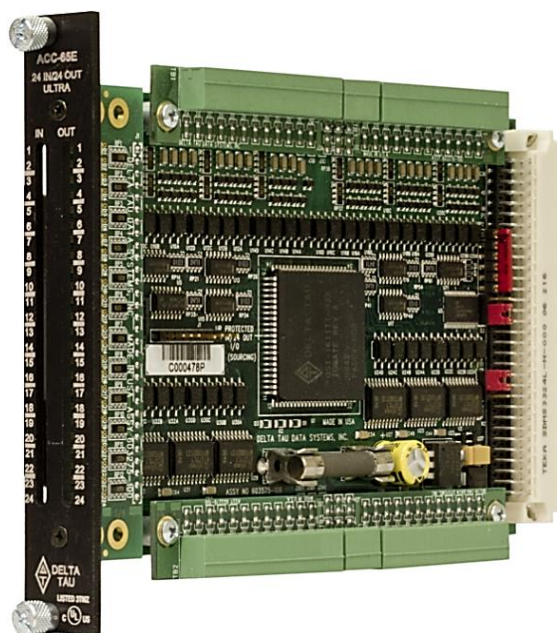


HARDWARE REFERENCE MANUAL

Accessory 65E



Digital I/O – Sinking Inputs, Sourcing Outputs

3A-603575-U

February 2, 2022

Document # MN-000208



Copyright Information

© 2022 Delta Tau Data Systems, Inc. All rights reserved.

This document is furnished for the customers of Delta Tau Data Systems, Inc. Other uses are unauthorized without written permission of Delta Tau Data Systems, Inc. Information contained in this manual may be updated from time-to-time due to product improvements, etc., and may not conform in every respect to former issues.

To report errors or inconsistencies, call or email your local Omron representative

Operating Conditions

All Delta Tau Data Systems, Inc. motion controller products, accessories, and amplifiers contain static sensitive components that can be damaged by incorrect handling. When installing or handling Delta Tau Data Systems, Inc. products, avoid contact with highly insulated materials. Only qualified personnel should be allowed to handle this equipment. Before powering, please ensure there is no visible damage to the product.

In the case of industrial applications, we expect our products to be protected from hazardous or conductive materials and/or environments that could cause harm to the controller by damaging components or causing electrical shorts. Our products should not be placed in locations that can accrue a lot of dust, salt, or conductive iron-like powder. When our products are used in an industrial environment, install them into an industrial electrical cabinet or industrial PC to protect them from excessive or corrosive moisture, abnormal ambient temperatures, and conductive materials. If Delta Tau Data Systems, Inc. products are directly exposed to hazardous or conductive materials and/or environments, we cannot guarantee their operation. For your own safety, please keep the product's environmental conditions within the range outlined by the Environment Specifications section that can be located from the table of contents in this manual.

Trademarks

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



EN Dispose in accordance with applicable regulations.



Safety Instructions

Qualified personnel must transport, assemble, install, and maintain this equipment. Properly qualified personnel are persons who are familiar with the transport, assembly, installation, and operation of equipment. The qualified personnel must know and observe the following standards and regulations:

IEC364resp.CENELEC HD 384 or DIN VDE 0100

IEC report 664 or DIN VDE 0110

National regulations for safety and accident prevention or VBG 4

Incorrect handling of products can result in injury and damage to persons and machinery. Strictly adhere to the installation instructions. Electrical safety is provided through a low-resistance earth connection. It is vital to ensure that all system components are connected to earth ground.

This product contains components that are sensitive to static electricity and can be damaged by incorrect handling. Avoid contact with high insulating materials (artificial fabrics, plastic film, etc.). Place the product on a conductive surface. Discharge any possible static electricity build-up by touching an unpainted, metal, grounded surface before touching the equipment.

Keep all covers and cabinet doors shut during operation. Be aware that during operation, the product has electrically charged components and hot surfaces. Control and power cables can carry a high voltage, even when the motor is not rotating. Never disconnect or connect the product while the power source is energized to avoid electric arcing.



Warning

A Warning identifies hazards that could result in personal injury or death. It precedes the discussion of interest.



Caution

A Caution identifies hazards that could result in equipment damage. It precedes the discussion of interest.



Note

A Note identifies information critical to the understanding or use of the equipment. It follows the discussion of interest.

REVISION HISTORY				
REV.	DESCRIPTION	DATE	CHG	APPVD
1	Added CE Declaration	06/07/06	CP	SF
2	Revs. To J1, J2, Pins 8, 15	05/11/07	CP	AO
3	Reformatted Schematics	01/18/08	CP	SF
4	Updated Max Current Output	01/29/08	CP	SM
5	Added UL seal, updated agency approval	10/01/09	CP	SF
6	Added Power PMAC/MACRO Reformatted manual	02/24/14	RN	RN
7	Corrected DIP switch settings	02/03/15	RN	RN
8	Updated “Operating temperature” in environmental specifications	09/14/20	SM	RN
9	Added Mounting and Installation section	12/15/20	SM	RN
A	Added UKCA Marking to front cover and added description in Agency of Approval section	09/15/21	AE	SM
B	Updated UKCA standard	02/02/22	AE	SF

This page intentionally left blank

Table of Contents

INTRODUCTION.....	8
SPECIFICATIONS.....	9
Environmental Specifications	9
Electrical Specifications.....	9
<i>Power Requirements</i>	9
<i>Fuse</i>	9
Input Drivers	9
Output Drivers	9
Physical Specifications	10
<i>Terminal Block Layout</i>	10
<i>D-Sub Layout</i>	10
Agency Approval and Safety	11
MOUNTING AND INSTALLATION	13
ADDRESSING THE ACC-65E	14
Address Select DIP Switch SW1	14
Legacy MACRO Dip Switch Settings	15
Hardware Address Limitations	15
USING THE ACC-65E WITH POWER UMAC	17
Declaring Pointers to I/O Structure Elements.....	17
USING THE ACC-65E WITH TURBO UMAC.....	18
Assigning M-Variables to I/O Memory Locations	18
Configuring the Control Word.....	19
USING THE ACC-65E WITH MACRO	20
MACRO16 I/O Node Addressing.....	21
Ring Controller I/O Node Addressing	23
<i>Turbo PMAC2 I/O Node Addressing</i>	23
<i>Power PMAC2 I/O Node Addressing</i>	24
<i>Power PMAC3 I/O Node Addressing</i>	26
Configuring MACRO I/O Transfers.....	29
<i>MS{anynode},MI160</i>	30
<i>MS{anynode},MI71</i>	34
<i>MS{anynode},MI69/MI70</i>	38
Accessing the Transferred Data	41
<i>Outputs Mirror Image Concept</i>	41
<i>Turbo PMAC2 MI160 Mapping Example</i>	42
<i>Turbo PMAC2 MI71 Mapping Example</i>	44

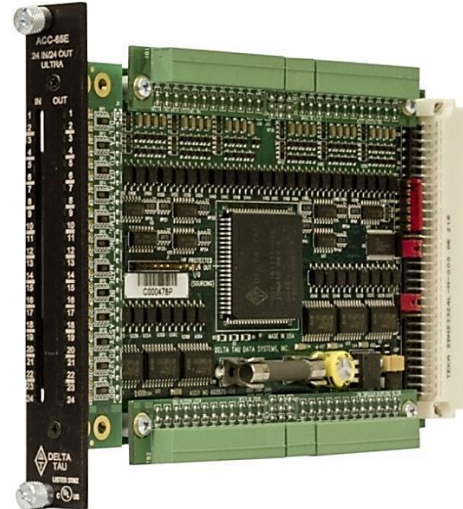
<i>Turbo PMAC2 MI69/70 Mapping Example</i>	46
<i>Power PMAC2 MI160 Mapping Example</i>	48
<i>Power PMAC2 MI71 Mapping Example</i>	50
<i>Power PMAC2 MI69/70 Mapping Example</i>	52
<i>Power PMAC3 MI160 Mapping Example</i>	54
<i>Power PMAC3 MI71 Mapping Example</i>	55
<i>Power PMAC3 MI69/70 Mapping Example</i>	56
Configuring the Control Word for MACRO	57
CONNECTOR PINOUTS AND WIRING	58
Terminal Block Connectors	58
<i>Top: Inputs</i>	58
<i>Bottom: Outputs</i>	59
<i>TB Wiring Diagram</i>	60
D-Sub Connectors	61
<i>Top: Inputs</i>	61
<i>Bottom: Outputs</i>	62
<i>D-Sub Wiring diagram</i>	63
P1: UMAC Bus (UBUS) Connector	64
APPENDIX A: E-POINT JUMPERS	66
APPENDIX B: SCHEMATICS	67
APPENDIX C: USING THE ACC-65E IN C	69
<i>ACC-65E C Library</i>	69
<i>Locations for the C Files</i>	70
<i>Function Descriptions</i>	72
<i>Example</i>	73

INTRODUCTION

The accessory 65E (ACC-65E) is a general purpose digital input and output card. It provides 24 inputs and 24 outputs of self-protected, optically isolated sinking inputs and sourcing outputs.

The ACC-65E is a 3U Euro style card intended to plug into the UMAC BUS backplane. It can be used with the:

- Turbo UMAC CPU
- Power UMAC CPU
- MACRO UMAC Station
(MACRO16 CPU, PMAC2 or PMAC3 Style)



SPECIFICATIONS

Environmental Specifications

Description	Specification	Notes
Operating Temperature	0°C to 55°C	
Storage Temperature	-25°C to 70°C	
Humidity	10% to 95 %	Non-Condensing

Electrical Specifications

Power Requirements

Whether providing the ACC-65E with power from the 3U backplane bus or externally (standalone mode) through TB1, the power requirements ($\pm 10\%$) are:

+5 V @ 0.38 A	(Board Logic Input)
+24 V @ 0.60 A	(Per Output Pin)
+24 V @ 8.00 A	(Maximum current when using all outputs simultaneously)



Note

TB1 is a 2-pin 5V connector on the base board (used for standalone operation), not to be confused with the 12-pin TB1 connector on either of the I/O mezzanine boards.

Fuse

Manufacturer	Specification	Delta Tau Part Number	
Bussmann	125 V @ 20 A	MDA 20A	

Input Drivers

The inputs to the ACC-65E have an activation range from 12V to 24V. Due to the self-protecting circuitry, the inputs can only be configured as sinking.

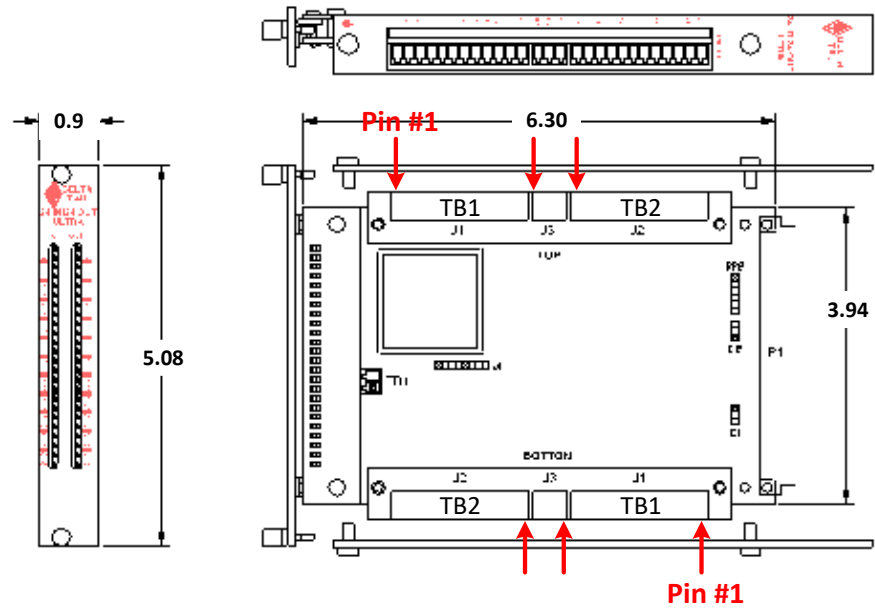
Although self-protecting, no more than 33VDC should be applied to any I/O pin. This is a limitation of the protective circuitry which includes [MMBZ33VALT1](#) Zener diodes.

Output Drivers

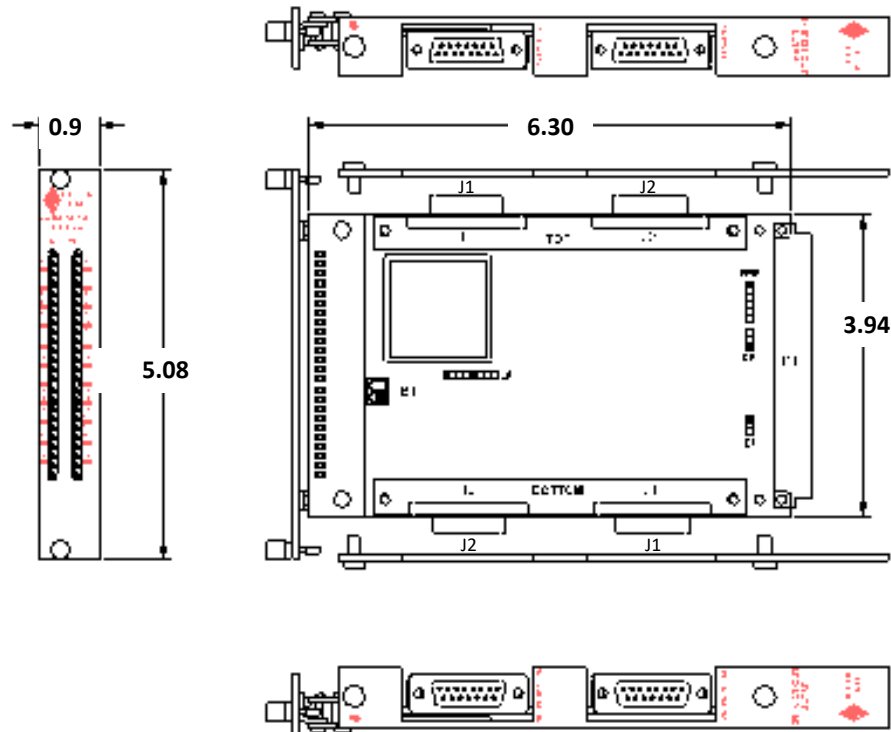
The output drivers use the Diodes Inc. Zetex ZXMS6006DG chip. The current drawn from each output line should be limited to 600 mA at voltage levels between 12 VDC and 24 VDC and no more than 8 Amps total for all outputs simultaneously.

Physical Specifications

Terminal Block Layout



D-Sub Layout



Description	Specification	Notes
Dimensions	Length: 16.256 cm (6.4 in.) Height: 10 cm (3.94 in.) Width: 2.03 cm (0.8 in.)	
Weight w/o Option 1A	180 g	Front Plate included
Terminal Block Connectors	FRONT-MC1,5/12-ST3,81 FRONT-MC1,5/5-ST3,81 FRONT-MC1,5/3-ST3,81	Terminal Blocks from Phoenix Contact. UL 94V-0
DB Option Connectors	DB15 Female	UL 94V-0

**Note**

The width is the width of the front plate. The length and height are the dimensions of the PCB. See Layout section for physical dimensions.

Agency Approval and Safety

Item	Description
CE Mark	Full Compliance
EMC	EN55011 Class A Group 1 EN61000-3-2 Class A EN61000-3-3 EN61000-4-2 EN61000-4-3 EN61000-4-4 EN61000-4-5 EN61000-4-6 EN61000-4-11
Safety	EN 61010-1
UL	UL 61010-1 File E314517
cUL	CAN/CSA C22.2 No. 1010.1-92 File E314517
UKCA	2016 No. 1091

Application of Council Directive: 89/336/EEC, 72/23/EEC

Manufacturers Name: Delta Tau Data Systems, Inc.

Manufacturers Address: 21314 Lassen Street
Chatsworth, CA 91311
USA

We, Delta Tau Data Systems, Inc. hereby declare that the product

Product Name: Accessory 65E

Model Number: 603575

And all of its options conforms to the following standards:

EN61326: 1997	Electrical equipment for measurement, control, and laboratory use- EMC requirements
EN55011: 1998	Limits and methods of measurements of radio disturbance characteristics of information technology equipment
EN61010-1	Electrical equipment for measurement, control, and laboratory use- Safety requirements
EN61000-3-2 :1995 A14:1998	Limits for harmonic current emissions. Criteria A
EN61000-3-3: 1995	Limitation of voltage fluctuations and flicker in low-voltage supply systems for equipment with rated current $\leq 16A$. Criteria B.
EN61000-4-2:1995 A1: 1998	Electro Static Discharge immunity test. Criteria B
EN61000-4-3: 1995 A1: 1998	Radiated, radio-frequency, electromagnetic field immunity test. Criteria A
EN61000-4-4: 1995	Electrical fast transients/burst immunity test. Criteria B
EN61000-4-5: 1995	Surge Test. Criteria B
EN61000-4-6: 1996	Conducted immunity test. Criteria A
EN61000-4-11: 1994	Voltage dips test. Criteria B and C

Date Issued: 11 May 2006

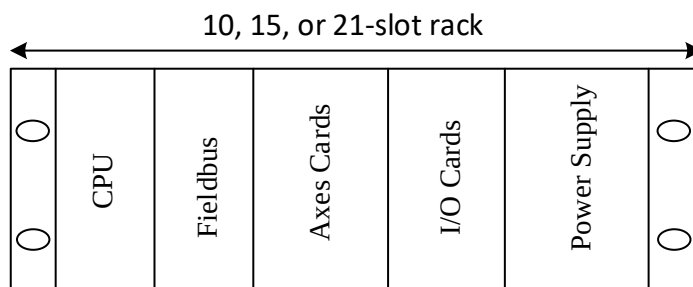
Place Issued: Chatsworth, California USA

Mark of Compliance



MOUNTING AND INSTALLATION

To connect a UMAC accessory, simply slide the board into any open slot of the UMAC rack. Customarily, accessories are installed from left to right as follows:



Prior to installation, make sure that you have set the jumpers and address settings to your desired requirements. Use the guide tracks that have been installed in the empty slots of your UMAC system when installing a board.

As you slide the board into the rack, use caution to ensure none of the components on the board make contact with the front plates of the boards on either side. Getting the front plate flush with the front of the rack and turning the front screws firmly will ensure a good connection with the backplane.

When removing a board from the system, the user must first pull out any wired connections from the top, bottom, and front panels then loosen the pem-nuts on the front of the rack. Next, the user can gently pull the board from the rack and use caution to ensure that none of the components on the board make contact with the boards on either side.



Caution

This product contains D-SUB style connectors. Do not overtighten any screws on mating connectors, and when possible, only tighten by hand. Overtightening, such as with manual or electric tools, may lead to the mating nut detaching when subsequently unscrewed, which may result in damage to the product, other electronics, and/or injury.



Screw

Hex nut
(Risk of falling)

ADDRESSING THE ACC-65E

Address Select DIP Switch SW1

The switch SW1 selects the starting address location for the first I/O gate on the ACC-65E. The following table shows the dip switch settings for the Turbo, Power, and MACRO Station settings:

Chip Select	Base Address				SW1 Positions					
	TURBO	MACRO	POWER		6	5	4	3	2	1
			I/O Base Offset	Index ACC-65E[n]						
CS10	Y:\$78C00	Y:\$8800	\$A00000	0	ON	ON	ON	ON	ON	ON
	Y:\$79C00	Y:\$9800	\$A08000	4	ON	ON	ON	OFF	ON	ON
	Y:\$7AC00	Y:\$A800	\$A10000	8	ON	ON	OFF	ON	ON	ON
	Y:\$7BC00	Y:\$B800	\$A18000	12	ON	ON	OFF	OFF	ON	ON
CS12	Y:\$78D00	Y:\$8840	\$B00000	1	ON	ON	ON	ON	ON	OFF
	Y:\$79D00	Y:\$9840	\$B08000	5	ON	ON	ON	OFF	ON	OFF
	Y:\$7AD00	Y:\$A840	\$B10000	9	ON	ON	OFF	ON	ON	OFF
	Y:\$7BD00	Y:\$B840	\$B18000	13	ON	ON	OFF	OFF	ON	OFF
CS14	Y:\$78E00	Y:\$8880	\$C00000	2	ON	ON	ON	ON	OFF	ON
	Y:\$79E00	Y:\$9880	\$C08000	6	ON	ON	ON	OFF	OFF	ON
	Y:\$7AE00	Y:\$A880	\$C10000	10	ON	ON	OFF	ON	OFF	ON
	Y:\$7BE00	Y:\$B880	\$C18000	14	ON	ON	OFF	OFF	OFF	ON
CS16	Y:\$78F00	Y:\$88C0	\$D00000	3	ON	ON	ON	ON	OFF	OFF
	Y:\$79F00	Y:\$98C0	\$D08000	7	ON	ON	ON	OFF	OFF	OFF
	Y:\$7AF00	Y:\$A8C0	\$D10000	11	ON	ON	OFF	ON	OFF	OFF
	Y:\$7BF00	Y:\$B8C0	\$D18000	15	ON	ON	OFF	OFF	OFF	OFF



Note

ON designates **Closed**. OFF designates **Open**. Factory default is all ON.

Legacy MACRO Dip Switch Settings

Chip Select	Base Address (Alternate)	SW1 Positions					
		6	5	4	3	2	1
10	Y:\$B800 (Y:\$FFE0)	ON (OFF)	ON (OFF)	OFF	OFF	ON	ON
12	Y:\$B840 (Y:\$FFE8)	ON (OFF)	ON (OFF)	OFF	OFF	ON	OFF
14	Y:\$B880 (Y:\$FFF0)	ON (OFF)	ON (OFF)	OFF	OFF	OFF	ON
16	Y:\$B8C0 (Y:\$FFF8)	ON (OFF)	ON (OFF)	ON	ON	OFF	OFF



Note

The Legacy Macro base addresses are double mapped. Set SW1 positions 5 & 6 to OFF if the alternate addressing is desired.

Hardware Address Limitations

Historically, two types of accessory cards have been designed for the UMAC 3U bus type rack: type A and type B cards. They can be sorted out as follows:

Name	Type	Category	Possible Number of Base Addresses	Maximum Number of cards in 1 rack
ACC-9E	A	General I/O	4	10
ACC-10E	A	General I/O	4	
ACC-11E	A	General I/O	4	
ACC-12E	A	General I/O	4	
ACC-53E	B	Feedback	12	12
ACC-57E	B	Feedback	12	
ACC-58E	B	Feedback	12	
ACC-59E	B	Analog I/O	12	
ACC-14E	B	General I/O	16	16
ACC-28E	B	Analog I/O	16	
ACC-36E	B	Analog I/O	16	
ACC-65E	B	General I/O	16	
ACC-66E	B	General I/O	16	
ACC-67E	B	General I/O	16	
ACC-68E	B	General I/O	16	
ACC-84E	B	Feedback	12	12

Addressing Type A and Type B accessory cards in a UMAC or MACRO station rack requires attention to the following set of rules:

Populating Rack with Type A Cards Only (no conflicts)

In this case, the card(s) can potentially use any available Address/Chip Select.



Note

The type A cards have only one base address per chip select (CS10, CS12, and CS14). Each card can be set up (jumper settings) to use the low, middle, or high byte of a specific base address. This makes it possible to populate a single rack with 9 (3 bases x 3 byte locations) Type A accessory cards. A fourth address is available at CS16 in which only the high byte can be used, thus making the maximum 10 Type A accessory cards.

Populating Rack with Type B Cards Only (no conflicts)

In this case, the card(s) can potentially use any available Address/Chip Select.

Populating Rack with Type A & Type B Cards (possible conflicts)

Typically, Type A and Type B cards should not share the same Chip Select. If this configuration is possible, then the following rules apply:

➤ **Type A and Type B Feedback Cards**

Type A cards cannot share the same base address as Type B Feedback cards.

➤ **Type A and Type B General I/O Cards**

Type A cards can share base addresses with Type B general I/O cards; however, in this case, Type B cards naturally use the lower byte (default), and Type A cards must be set to the middle/high byte of the selected base address.

➤ **Type A Cards and Type B Analog Cards**

Type A cards can share base addresses with Type B analog I/O cards; however, in this case, Type B cards naturally use the middle/high bytes (default), so Type A cards should be set to the low byte of the selected base address.



Note

The above conflicts only occur with the first base address in each Chip Select of the Type B cards (i.e. \$78C00, \$78D00, \$78E00, and \$78F00). Conflicts can more simply be avoided by using only the second to fourth addresses of each Chip Select.

USING THE ACC-65E WITH POWER UMAC

Using the ACC-65E with Power UMAC requires:

- Knowing and configuring the index (address offset) of the card
- Declaring pointers, with user given names, to the appropriate I/O structure elements

Declaring Pointers to I/O Structure Elements

Mapping an ACC-65E at index 0, for example, with user configurable pointer names:

Inputs	Outputs
<code>PTR Input1->ACC65E[0].DataReg[0].0.1;</code>	<code>PTR Output1->ACC65E[0].DataReg[3].0.1;</code>
<code>PTR Input2->ACC65E[0].DataReg[0].1.1;</code>	<code>PTR Output2->ACC65E[0].DataReg[3].1.1;</code>
<code>PTR Input3->ACC65E[0].DataReg[0].2.1;</code>	<code>PTR Output3->ACC65E[0].DataReg[3].2.1;</code>
<code>PTR Input4->ACC65E[0].DataReg[0].3.1;</code>	<code>PTR Output4->ACC65E[0].DataReg[3].3.1;</code>
<code>PTR Input5->ACC65E[0].DataReg[0].4.1;</code>	<code>PTR Output5->ACC65E[0].DataReg[3].4.1;</code>
<code>PTR Input6->ACC65E[0].DataReg[0].5.1;</code>	<code>PTR Output6->ACC65E[0].DataReg[3].5.1;</code>
<code>PTR Input7->ACC65E[0].DataReg[0].6.1;</code>	<code>PTR Output7->ACC65E[0].DataReg[3].6.1;</code>
<code>PTR Input8->ACC65E[0].DataReg[0].7.1;</code>	<code>PTR Output8->ACC65E[0].DataReg[3].7.1;</code>
<code>PTR Input9->ACC65E[0].DataReg[1].0.1;</code>	<code>PTR Output9->ACC65E[0].DataReg[4].0.1;</code>
<code>PTR Input10->ACC65E[0].DataReg[1].1.1;</code>	<code>PTR Output10->ACC65E[0].DataReg[4].1.1;</code>
<code>PTR Input11->ACC65E[0].DataReg[1].2.1;</code>	<code>PTR Output11->ACC65E[0].DataReg[4].2.1;</code>
<code>PTR Input12->ACC65E[0].DataReg[1].3.1;</code>	<code>PTR Output12->ACC65E[0].DataReg[4].3.1;</code>
<code>PTR Input13->ACC65E[0].DataReg[1].4.1;</code>	<code>PTR Output13->ACC65E[0].DataReg[4].4.1;</code>
<code>PTR Input14->ACC65E[0].DataReg[1].5.1;</code>	<code>PTR Output14->ACC65E[0].DataReg[4].5.1;</code>
<code>PTR Input15->ACC65E[0].DataReg[1].6.1;</code>	<code>PTR Output15->ACC65E[0].DataReg[4].6.1;</code>
<code>PTR Input16->ACC65E[0].DataReg[1].7.1;</code>	<code>PTR Output16->ACC65E[0].DataReg[4].7.1;</code>
<code>PTR Input17->ACC65E[0].DataReg[2].0.1;</code>	<code>PTR Output17->ACC65E[0].DataReg[5].0.1;</code>
<code>PTR Input18->ACC65E[0].DataReg[2].1.1;</code>	<code>PTR Output18->ACC65E[0].DataReg[5].1.1;</code>
<code>PTR Input19->ACC65E[0].DataReg[2].2.1;</code>	<code>PTR Output19->ACC65E[0].DataReg[5].2.1;</code>
<code>PTR Input20->ACC65E[0].DataReg[2].3.1;</code>	<code>PTR Output20->ACC65E[0].DataReg[5].3.1;</code>
<code>PTR Input21->ACC65E[0].DataReg[2].4.1;</code>	<code>PTR Output21->ACC65E[0].DataReg[5].4.1;</code>
<code>PTR Input22->ACC65E[0].DataReg[2].5.1;</code>	<code>PTR Output22->ACC65E[0].DataReg[5].5.1;</code>
<code>PTR Input23->ACC65E[0].DataReg[2].6.1;</code>	<code>PTR Output23->ACC65E[0].DataReg[5].6.1;</code>
<code>PTR Input24->ACC65E[0].DataReg[2].7.1;</code>	<code>PTR Output24->ACC65E[0].DataReg[5].7.1;</code>

Typically, these pointers would be put in a “Global Includes” file.



Note

To switch/add definitions to a different card, change the ACC65E[0] to ACC65E[n], where **n** is the card index set by the dip switch settings.



Note

With Power PMAC, the control word is set up automatically by the firmware with **ACC65E[n].CtrlReg = 7**.

USING THE ACC-65E WITH TURBO UMAC

Using the ACC-65E with Turbo UMAC requires:

- Knowing (configuring) the base address of the card
- Pointing M-Variables to the appropriate I/O memory locations
- Configuring the control word

Assigning M-Variables to I/O Memory Locations

Mapping an ACC-65E at \$78C00, for example, with user configurable M-Variable numbers and name substitutions:

Inputs		Outputs	
#define Input1 M7001	Input1->Y:\$78C00,0,1	#define Output1 M7025	Output1->Y:\$78C03,0,1
#define Input2 M7002	Input2->Y:\$78C00,1,1	#define Output2 M7026	Output2->Y:\$78C03,1,1
#define Input3 M7003	Input3->Y:\$78C00,2,1	#define Output3 M7027	Output3->Y:\$78C03,2,1
#define Input4 M7004	Input4->Y:\$78C00,3,1	#define Output4 M7028	Output4->Y:\$78C03,3,1
#define Input5 M7005	Input5->Y:\$78C00,4,1	#define Output5 M7029	Output5->Y:\$78C03,4,1
#define Input6 M7006	Input6->Y:\$78C00,5,1	#define Output6 M7030	Output6->Y:\$78C03,5,1
#define Input7 M7007	Input7->Y:\$78C00,6,1	#define Output7 M7031	Output7->Y:\$78C03,6,1
#define Input8 M7008	Input8->Y:\$78C00,7,1	#define Output8 M7032	Output8->Y:\$78C03,7,1
	Input9->Y:\$78C01,0,1		Output9->Y:\$78C04,0,1
#define Input9 M7009	Input10->Y:\$78C01,1,1	#define Output9 M7033	Output10->Y:\$78C04,1,1
#define Input10 M7010	Input11->Y:\$78C01,2,1	#define Output10 M7034	Output11->Y:\$78C04,2,1
#define Input11 M7011	Input12->Y:\$78C01,3,1	#define Output11 M7035	Output12->Y:\$78C04,3,1
#define Input12 M7012	Input13->Y:\$78C01,4,1	#define Output12 M7036	Output13->Y:\$78C04,4,1
#define Input13 M7013	Input14->Y:\$78C01,5,1	#define Output13 M7037	Output14->Y:\$78C04,5,1
#define Input14 M7014	Input15->Y:\$78C01,6,1	#define Output14 M7038	Output15->Y:\$78C04,6,1
#define Input15 M7015	Input16->Y:\$78C01,7,1	#define Output15 M7039	Output16->Y:\$78C04,7,1
#define Input16 M7016		#define Output16 M7040	
	Input17->Y:\$78C02,0,1	#define Output17 M7041	Output17->Y:\$78C05,0,1
#define Input17 M7017	Input18->Y:\$78C02,1,1	#define Output18 M7042	Output18->Y:\$78C05,1,1
#define Input18 M7018	Input19->Y:\$78C02,2,1	#define Output19 M7043	Output19->Y:\$78C05,2,1
#define Input19 M7019	Input20->Y:\$78C02,3,1	#define Output20 M7044	Output20->Y:\$78C05,3,1
#define Input20 M7020	Input21->Y:\$78C02,4,1	#define Output21 M7045	Output21->Y:\$78C05,4,1
#define Input21 M7021	Input22->Y:\$78C02,5,1	#define Output22 M7046	Output22->Y:\$78C05,5,1
#define Input22 M7022	Input23->Y:\$78C02,6,1	#define Output23 M7047	Output23->Y:\$78C05,6,1
#define Input23 M7023	Input24->Y:\$78C02,7,1	#define Output24 M7048	Output24->Y:\$78C05,7,1
#define Input24 M7024			



Note

To address a different card, replace the **xx** digits in the address locations \$07**xx**00 to \$07**xx**05 to correspond to the base address configured by the dip switch settings.

Configuring the Control Word

With Turbo PMAC, the control word must be set to 7. This is done by writing (once on power-up) to the control register which is at base address + 7, bits [7:0].

```
M7000->Y:$078C07,0,8 // For base address $78C00
```

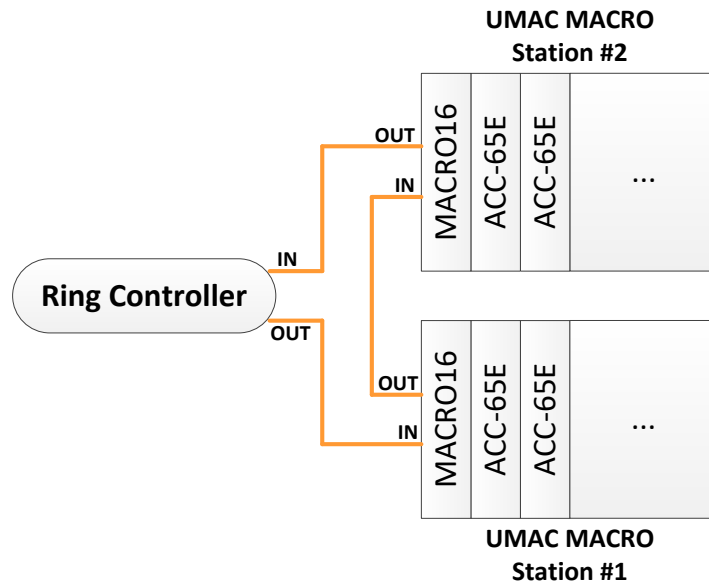
```
Open PLC 1 Clear  
M7000 = 7  
Disable PLC 1  
Close
```

USING THE ACC-65E WITH MACRO

In a MACRO configuration, the ACC-65E resides typically in a UMAC MACRO Station with a MACRO8 (legacy) or MACRO16 CPU.

The ring controller can be either a Turbo or a Power PMAC.

- **Turbo PMAC ring controllers:**
 - Turbo Brick family
 - Turbo UMAC with ACC-5E
 - Turbo UltraLite
- **Power PMAC ring controllers:**
 - Power Brick family
 - Power UMAC with ACC-5E
 - Power UMAC with ACC-5E3
 - Power EtherLite



Generally, the user's goal is to transfer the ACC-65E I/O data (48 bits per card) from/to the ring controller by reading the 24-bits of inputs and writing to the 24-bits of outputs.

This I/O data transfer is accomplished using I/O nodes; therefore, it is essential to know:

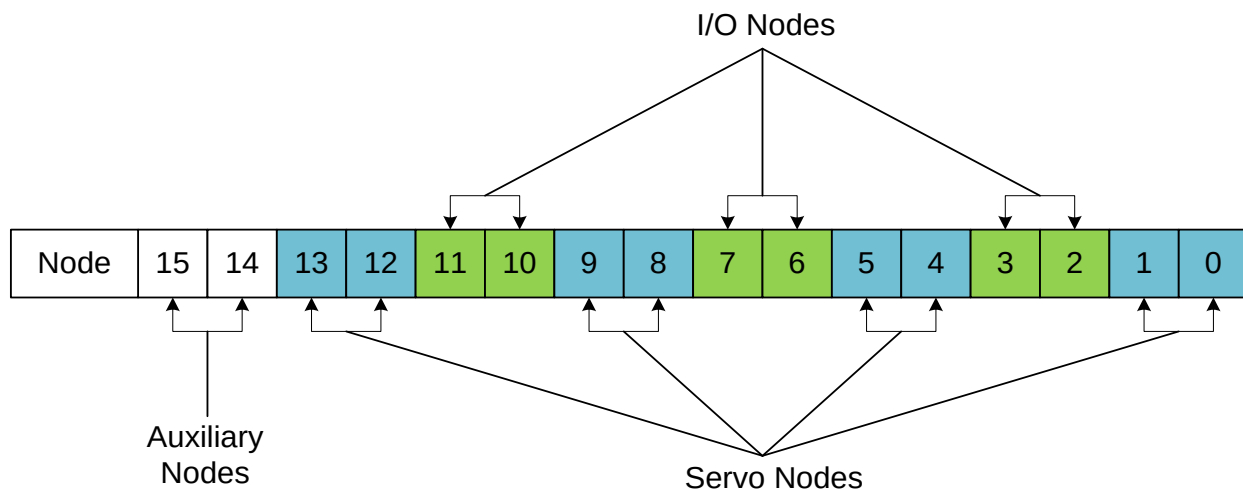
- Which I/O nodes are used
- Which MACRO station I/O node addresses correspond to which Ring Controller I/O node addresses

MACRO16 I/O Node Addressing

A MACRO IC consists of a number of auxiliary, servo, and I/O nodes:

- Auxiliary nodes are Master/Control registers and are for internal firmware use.
- Servo nodes carry information such as feedback, commands, and flags for motor control.
- I/O nodes are by default unoccupied and are configurable for transferring miscellaneous data.

The MACRO16 CPU is populated with two **PMAC2 style MACRO ICs**, each consisting of 16 nodes: 2 auxiliary, 8 servo, and 6 I/O nodes (the legacy MACRO8 has only one IC).



Each I/O node consists of one 24-bit and three 16-bit (upper) data registers:

13	12	11	10	9	8	7	6	5	4	3	2	1	0
		24-bit 1 st 16-bit 2 nd 16-bit 3 rd 16-bit	24-bit 1 st 16-bit 2 nd 16-bit 3 rd 16-bit			24-bit 1 st 16-bit 2 nd 16-bit 3 rd 16-bit	24-bit 1 st 16-bit 2 nd 16-bit 3 rd 16-bit			24-bit 1 st 16-bit 2 nd 16-bit 3 rd 16-bit	24-bit 1 st 16-bit 2 nd 16-bit 3 rd 16-bit		



Note

The MACRO8 CPU is populated with only 1 PMAC2 Style MACRO IC (**IC# 0**).

The I/O node data register addresses (\$C0XX) for the two MACRO ICs on a station are:

MACRO Station IC #0						
Node	2	3	6	7	10	11
24-bit	X:\$C0A0	X:\$C0A4	X:\$C0A8	X:\$C0AC	X:\$C0B0	X:\$C0B4
16-bit	X:\$C0A1	X:\$C0A5	X:\$C0A9	X:\$C0AD	X:\$C0B1	X:\$C0B5
16-bit	X:\$C0A2	X:\$C0A6	X:\$C0AA	X:\$C0AE	X:\$C0B2	X:\$C0B6
16-bit	X:\$C0A3	X:\$C0A7	X:\$C0AB	X:\$C0AF	X:\$C0B3	X:\$C0B7

MACRO Station IC #1						
Node	2	3	6	7	10	11
24-bit	X:\$C0E0	X:\$C0E4	X:\$C0E8	X:\$C0EC	X:\$C0F0	X:\$C0F4
16-bit	X:\$C0E1	X:\$C0E5	X:\$C0E9	X:\$C0ED	X:\$C0F1	X:\$C0F5
16-bit	X:\$C0E2	X:\$C0E6	X:\$C0EA	X:\$C0EE	X:\$C0F2	X:\$C0F6
16-bit	X:\$C0E3	X:\$C0E7	X:\$C0EB	X:\$C0EF	X:\$C0F3	X:\$C0F7



Note

I/O nodes which will be chosen to transfer the ACC-65E(s) data are configurable and chosen by the user depending on availability and I/O node management in the MACRO station.



Note

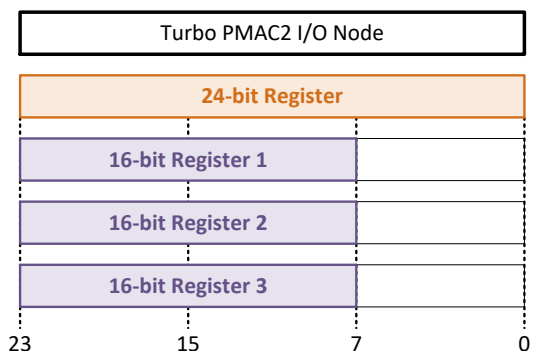
Non-Turbo PMAC2 Ultralite (legacy) I/O node addresses are the same as MACRO Station IC# 0 node registers. For instance, addresses on the ring controller start at \$C0A0, instead of at \$78420 as seen in the following section.

Ring Controller I/O Node Addressing

Turbo PMAC2 I/O Node Addressing

Turbo ring controllers interface with PMAC2 Style MACRO ICs. The data resides in the bit fields illustrated on the right:

Turbo ring controllers can be populated with up to 4 PMAC2 style MACRO ICs (reported by **I4902**). Below are the I/O node addresses (\$7XXXX) for each of the PMAC2 Style MACRO ICs:



Turbo Ring Controller MACRO IC #0 Node Registers						
Station I/O Node#	2	3	6	7	10	11
Ring Controller I/O Node#	2	3	6	7	10	11
24-bit	X:\$78420	X:\$78424	X:\$78428	X:\$7842C	X:\$78430	X:\$78434
16-bit	X:\$78421	X:\$78425	X:\$78429	X:\$7842D	X:\$78431	X:\$78435
16-bit	X:\$78422	X:\$78426	X:\$7842A	X:\$7842E	X:\$78432	X:\$78436
16-bit	X:\$78423	X:\$78427	X:\$7842B	X:\$7842F	X:\$78433	X:\$78437

Turbo Ring Controller MACRO IC #1 Node Registers						
Station I/O Node#	2	3	6	7	10	11
Ring Controller I/O Node#	18	19	22	23	26	27
24-bit	X:\$79420	X:\$79424	X:\$79428	X:\$7942C	X:\$79430	X:\$79434
16-bit	X:\$79421	X:\$79425	X:\$79429	X:\$7942D	X:\$79431	X:\$79435
16-bit	X:\$79422	X:\$79426	X:\$7942A	X:\$7942E	X:\$79432	X:\$79436
16-bit	X:\$79423	X:\$79427	X:\$7942B	X:\$7942F	X:\$79433	X:\$79437

Turbo Ring Controller MACRO IC #2 Node Registers						
Station I/O Node#	2	3	6	7	10	11
Ring Controller I/O Node#	34	35	38	39	42	43
24-bit	X:\$7A420	X:\$7A424	X:\$7A428	X:\$7A42C	X:\$7A430	X:\$7A434
16-bit	X:\$7A421	X:\$7A425	X:\$7A429	X:\$7A42D	X:\$7A431	X:\$7A435
16-bit	X:\$7A422	X:\$7A426	X:\$7A42A	X:\$7A42E	X:\$7A432	X:\$7A436
16-bit	X:\$7A423	X:\$7A427	X:\$7A42B	X:\$7A42F	X:\$7A433	X:\$7A437

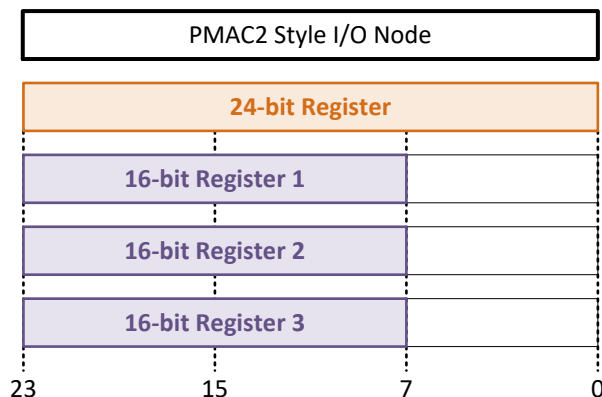
Turbo Ring Controller MACRO IC #3 Node Registers						
--	--	--	--	--	--	--

Station I/O Node#	2	3	6	7	10	11
Ring Controller I/O Node#	50	51	54	55	58	59
24-bit	X:\$7B420	X:\$7B424	X:\$7B428	X:\$7B42C	X:\$7B430	X:\$7B434
16-bit	X:\$7B421	X:\$7B425	X:\$7B429	X:\$7B42D	X:\$7B431	X:\$7B435
16-bit	X:\$7B422	X:\$7B426	X:\$7B42A	X:\$7B42E	X:\$7B432	X:\$7B436
16-bit	X:\$7B423	X:\$7B427	X:\$7B42B	X:\$7B42F	X:\$7B433	X:\$7B437

Power PMAC2 I/O Node Addressing

The Power PMAC can interface with the ACC-5E which carries PMAC2 Style MACRO ICs.

The PMAC2 Style MACRO IC I/O node data registers reside in the bit fields illustrated on the right:



And the corresponding structure elements:

Structure Element	Data Register
Gate2[<i>i</i>].Macro[<i>j</i>][0]	24 bits
Gate2[<i>i</i>].Macro[<i>j</i>][1]	16 bits
Gate2[<i>i</i>].Macro[<i>j</i>][2]	16 bits
Gate2[<i>i</i>].Macro[<i>j</i>][3]	16 bits

Where: ➤ *i* is the PMAC2 Style MACRO IC index
➤ *j* is the I/O node number.



Note

Bitwise mapping, into the PMAC2 Style MACRO structure elements requires Power PMAC firmware version **1.5.8.215** or newer.

The tables below show I/O Node numbers of the 4 PMAC2 Style MACRO ICs:

	Gate2[0]					
Station I/O Node#	2	3	6	7	10	11
Ring Controller I/O Node [j]	2	3	6	7	10	11

	Gate2[1]					
Station I/O Node#	2	3	6	7	11	12
Ring Controller I/O Node [j]	18	19	22	23	26	27

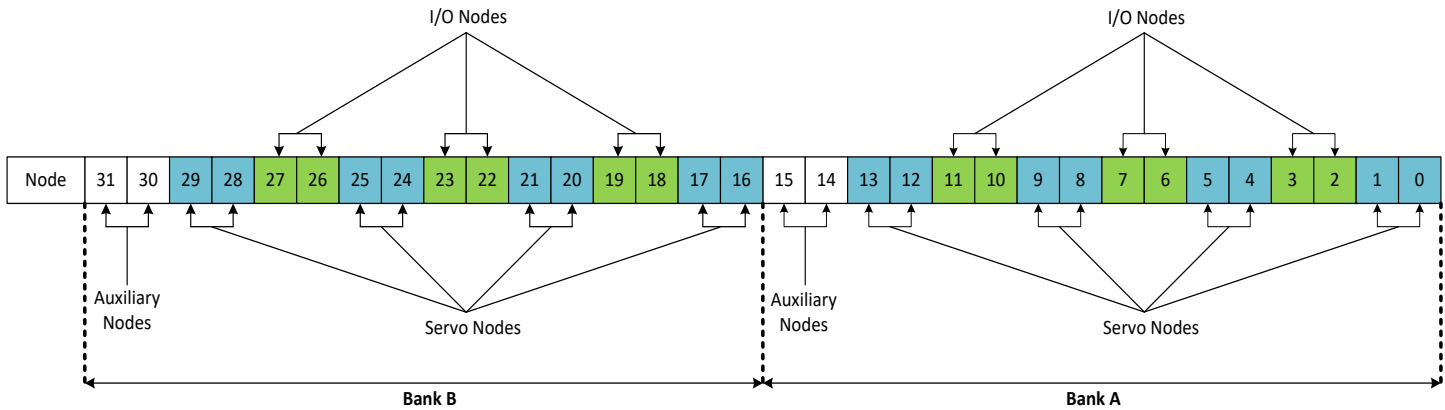
	Gate2[2]					
Station I/O Node#	2	3	6	7	10	11
Ring Controller I/O Node [j]	34	35	38	39	42	43

	Gate2[3]					
Station I/O Node#	2	3	6	7	11	12
Ring Controller I/O Node [j]	50	51	54	55	58	59

Power PMAC3 I/O Node Addressing

A **PMAC3 style MACRO IC** consists of 32 nodes: 4 auxiliary, 16 servo, and 12 I/O nodes. One or more of these ICs can be found in the following hardware:

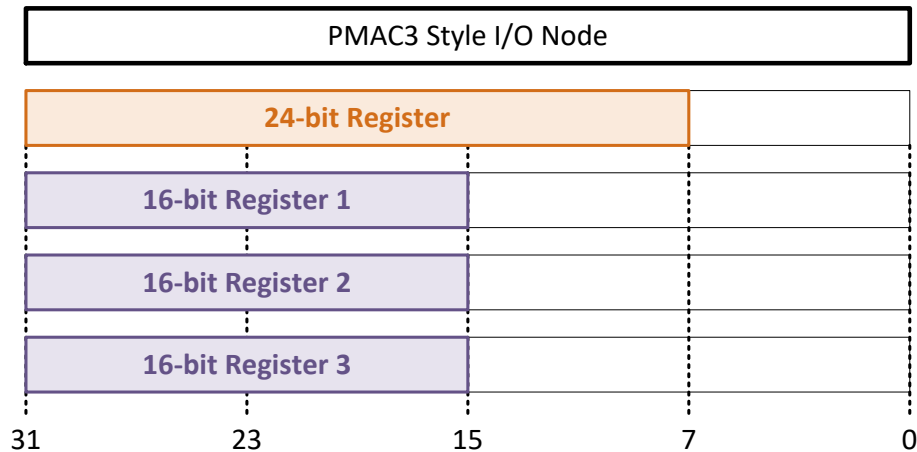
- Power Brick Family
- Power UMAC with ACC-5E3
- Power EtherLite



Note

The Power PMAC can have up to 16 PMAC3 Style MACRO ICs. ICs present are reported by the variable **Macro.IC3s**.

A **PMAC3 Style MACRO IC** I/O node consists of 4 data registers: 1 x 24-bit and 3 x 16-bit residing in the following bit fields:



With the **PMAC3 Style MACRO ICs**, the I/O node data registers are typically accessed using structure elements, which can be inputs or outputs for either bank:

Bank B		Bank A		Data Register
Inputs	Outputs	Inputs	Outputs	
Gate3[<i>i</i>].MacroInB[<i>j</i>][0]	Gate3[<i>i</i>].MacroOutB[<i>j</i>][0]	Gate3[<i>i</i>].MacroInA[<i>j</i>][0]	Gate3[<i>i</i>].MacroOutA[<i>j</i>][0]	24-bit
Gate3[<i>i</i>].MacroInB[<i>j</i>][1]	Gate3[<i>i</i>].MacroOutB[<i>j</i>][1]	Gate3[<i>i</i>].MacroInA[<i>j</i>][1]	Gate3[<i>i</i>].MacroOutA[<i>j</i>][1]	1 st 16-bit
Gate3[<i>i</i>].MacroInB[<i>j</i>][2]	Gate3[<i>i</i>].MacroOutB[<i>j</i>][2]	Gate3[<i>i</i>].MacroInA[<i>j</i>][2]	Gate3[<i>i</i>].MacroOutA[<i>j</i>][2]	2 nd 16-bit
Gate3[<i>i</i>].MacroInB[<i>j</i>][3]	Gate3[<i>i</i>].MacroOutB[<i>j</i>][3]	Gate3[<i>i</i>].MacroInA[<i>j</i>][3]	Gate3[<i>i</i>].MacroOutA[<i>j</i>][3]	3 rd 16-bit

Where:

- *i* is the PMAC3 Style MACRO IC index
- *j* is the I/O node number.



Note

Bitwise mapping into the PMAC3 Style MACRO structure elements requires Power PMAC firmware version **1.5.8.215** or newer.

Below are example tables showing I/O Node numbers of the first 4 PMAC3 Style MACRO ICs:

Gate3[0]													
Station Node#	Bank A						Bank B						
	2	3	6	7	10	11	2	3	6	7	11	12	
Ring Controller I/O Node [<i>j</i>]	2	3	6	7	10	11	18	19	22	23	26	27	

Gate3[1]													
Station Node#	Bank A						Bank B						
	2	3	6	7	10	11	2	3	6	7	11	12	
Ring Controller I/O Node [<i>j</i>]	34	35	38	39	42	43	50	51	54	55	58	59	

	Gate3[2]											
	Bank A						Bank B					
Station Node#	2	3	6	7	10	11	2	3	6	7	11	12
Ring Controller I/O Node [j]	66	67	70	71	74	75	82	83	86	87	90	91

	Gate3[3]											
	Bank A						Bank B					
Station Node#	2	3	6	7	10	11	2	3	6	7	11	12
Ring Controller I/O Node [j]	98	99	102	103	106	107	114	115	118	119	122	123

Configuring MACRO I/O Transfers

Having set up the following:

- Ring Controller MACRO communication for ASCII and MS commands
I6840 with Turbo PMAC2
Gate2[i].MacroMode with Power PMAC2
Gate3[i].MacroModeA and **Gate3[i].MacroModeB** with Power PMAC3
- MACRO Station MACRO communication for ASCII and MS commands
MI996
- Enabled the chosen I/O nodes on the ring controller
I6841 with Turbo PMAC2
Gate2[i].MacroEnable with Power PMAC2
Gate3[i].MacroEnableA and **Gate3[i].MacroEnableB** with Power PMAC3
- Enabled the corresponding I/O nodes on the MACRO station IC
MS{anynode},MI975
Typically, masked with the enabled I/O node(s) E.g. MS2,MI975=\$4 enables transfers using I/O node number 2.
- Set up the I/O Data Transfer Period
MS{anynode},MI19
Typically = 4

The ACC-65E(s) I/O data should now be available to transfer to/from the ring controller.

Each ACC-65E possesses 48 bits (24 in/24 out) of data to be transferred. Depending on the number of cards in the MACRO station and I/O nodes available, one or more of the following methods can be used, stated in the order of simplicity:

- **MS{anynode},MI160**
48-bit transfer into a 1 x 24-bit data register (uses 1 x I/O node)
- **MS{anynode},MI71**
48-bit transfer into 2 x 24-bit data registers (uses 2 x I/O nodes)
- **MS{anynode},MI69/MI70**
48-bit transfer into 3 x 16-bit data registers (uses 1 x I/O nodes)



Note

{anynode} refers to any activated node on a particular MACRO IC.



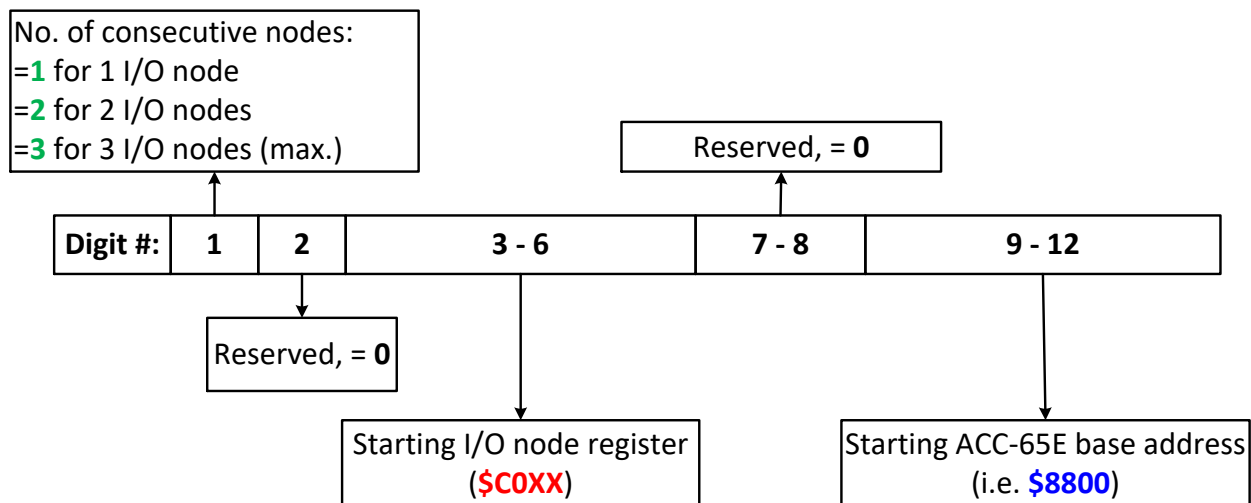
Note

MS{anynode},MI160 requires MACRO16 CPU firmware 1.204 or newer.

MS{anynode},MI160

MI160 transfers all 48 bits (24 in, 24 out) of an ACC-65E into a single read/write 24-bit data register. It handles up to 3 x ACC-65Es at consecutive base addresses (e.g. Y:\$8800, Y:\$9800, Y:\$A800) and places the data in 3 x consecutive 24-bit data registers.

MI160 is a 48-bit variable represented as 12 hexadecimal digits which are set up as follows (digit #1 is leftmost when constructing the word):

**Note**

MI160 requires MACRO16 CPU firmware ~1.204 or higher.

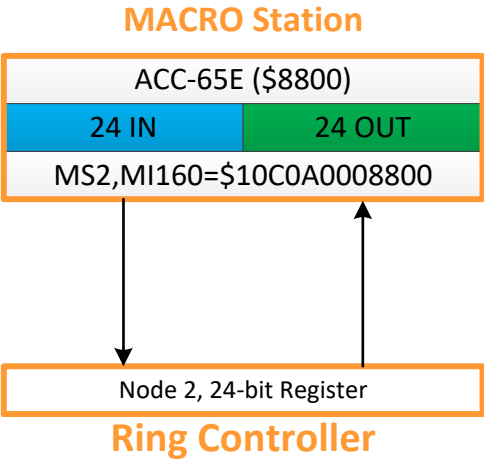
**Note**

For multiple ACC-65E transfers with MI160, consecutive cards must be under the same chip select.

MI160 example 1

Transferring I/O data for one ACC-65E at address \$8800 over node 2 of MACRO IC 0:

```
MS2,MI19=4           // Data transfer period [msec]
MS2,MI160=$10C0A0008800 // MI160 transfer
MS2,MI975=$4         // Enable Mask I/O node 2
```



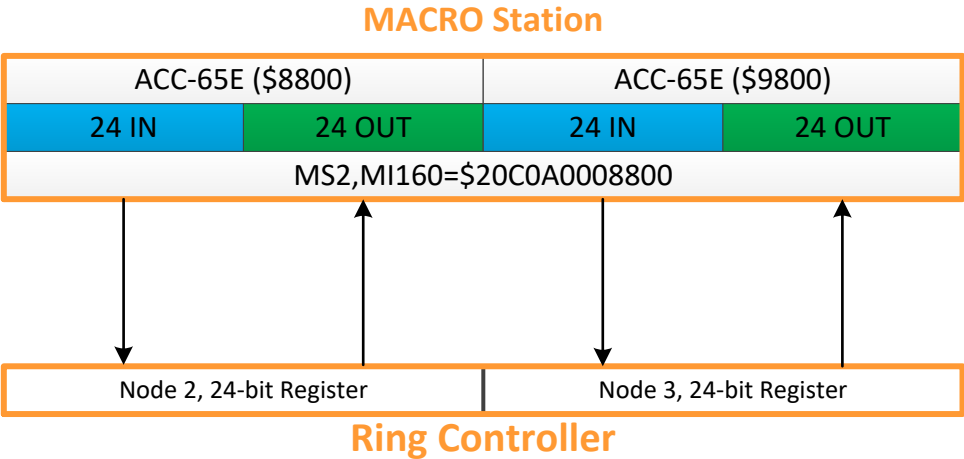
Having downloaded the above settings into the MACRO16 station, the inputs can be read and outputs can be transferred from the ring controller side (I/O node 2) in the following bit fields:

	Turbo		Power			
	I/O Node Address	Data Bits	Structure Element (PMAC2)	Data Bits	Structure Element (PMAC3)	Data Bits
Inputs	X:\$78420	23:00	Gate2[0].Macro[2][0]	23:00	Gate3[0].MacroInA[2][0]	31:08
Outputs					Gate3[0].MacroOutA[2][0]	

MI160 example 2

Transferring I/O data for two ACC-65Es at consecutive base addresses \$8800 and \$9800 over nodes 2 and 3 of MACRO IC 0:

MS2,MI19=4
MS2,MI160=\$20C0A0008800
MS2,MI975=\$C



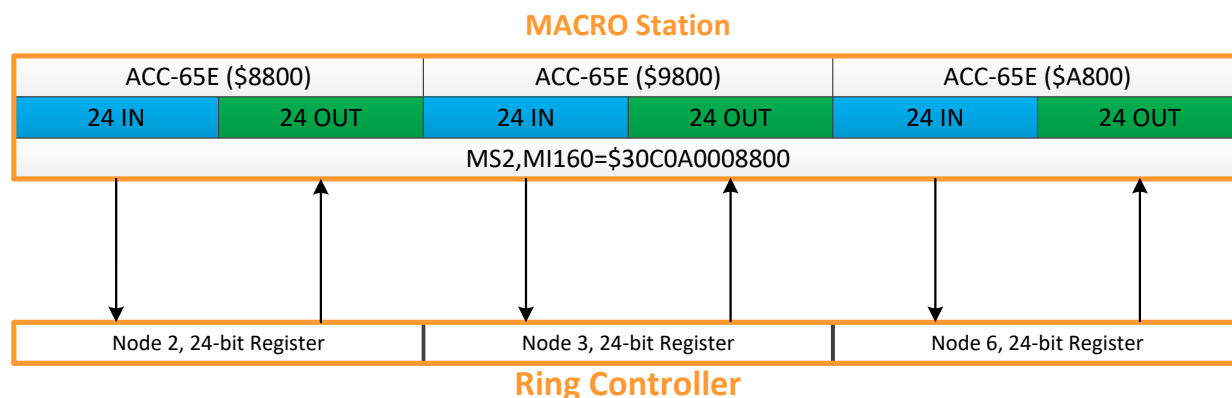
Having downloaded the above settings into the MACRO16 station, the inputs can be read and outputs can be transferred from the ring controller side (I/O nodes 2 and 3) in the following bit fields:

		Turbo		Power			
		I/O Node Address	Data Bits	Structure Element (PMAC2)	Data Bits	Structure Element (PMAC3)	Data Bits
1 st ACC-65E	Inputs	X:\$78420	23:00	Gate2[0].Macro[2][0]	23:00	Gate3[0].MacroInA[2][0]	31:08
	Outputs					Gate3[0].MacroOutA[2][0]	
2 nd ACC-65E	Inputs	X:\$78424	23:00	Gate2[0].Macro[3][0]	23:00	Gate3[0].MacroInA[3][0]	31:08
	Outputs					Gate3[0].MacroOutA[3][0]	

MI160 example 3

Transferring I/O data for three ACC-65Es at consecutive base addresses \$8800, \$9800, and \$A800 over nodes 2, 3, and 6:

```
MS2,MI19=4
MS2,MI160=$30C0A0008800
MS2,MI975=$4C
```



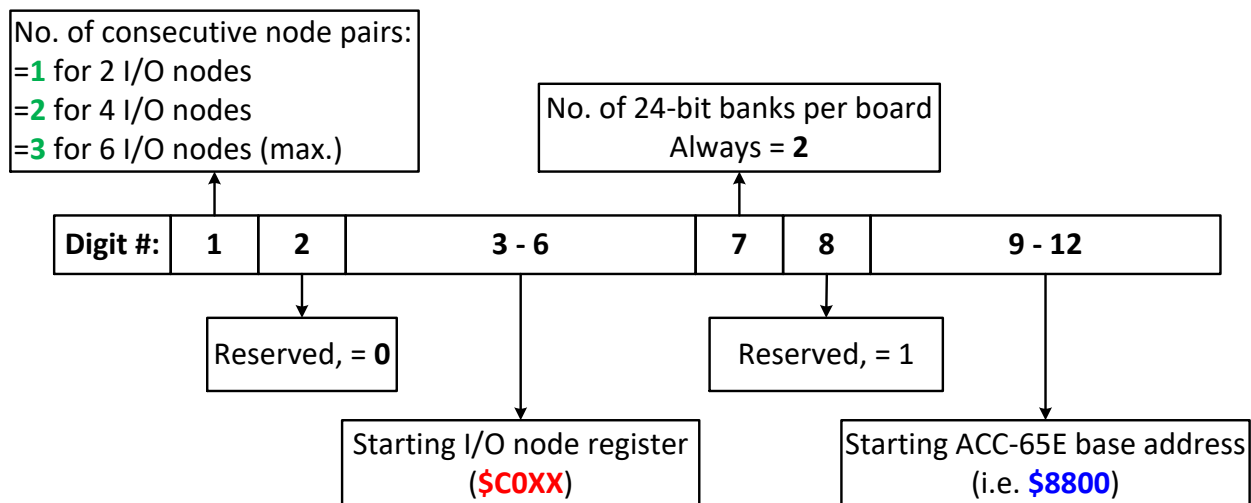
Having downloaded the above settings into the MACRO16 station, the inputs can be read and outputs can be transferred from the ring controller side (I/O nodes 2, 3, and 6) in the following bit fields:

		Turbo		Power			
		I/O Node Address	Data Bits	Structure Element (PMAC2)	Data Bits	Structure Element (PMAC3)	Data Bits
1 st ACC-65E	Inputs	X:\$78420	23:00	Gate2[0].Macro[2][0]	23:00	Gate3[0].MacroInA[2][0]	31:08
	Outputs					Gate3[0].MacroOutA[2][0]	
2 nd ACC-65E	Inputs	X:\$78424	23:00	Gate2[0].Macro[3][0]	23:00	Gate3[0].MacroInA[3][0]	31:08
	Outputs					Gate3[0].MacroOutA[3][0]	
3 rd ACC-65E	Inputs	X:\$78428	23:00	Gate2[0].Macro[6][0]	23:00	Gate3[0].MacroInA[6][0]	31:08
	Outputs					Gate3[0].MacroOutA[6][0]	

MS{anynode},MI71

MI71 transfers all 48 bits (24 in, 24 out) of an ACC-65E into 2 x consecutive I/O node 24-bit data registers. It handles up to 3 x ACC-65Es at consecutive base addresses (e.g. Y:\$8800, Y:\$8900, Y:\$8A00), and places the data in 6 x consecutive 24-bit I/O node data registers.

MI71 is a 48-bit variable represented as 12 hexadecimal digits which are set up as follows (digit #1 is leftmost when constructing the word):

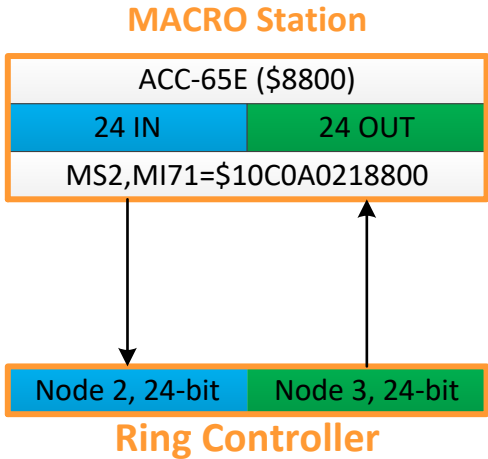
**Note**

For multiple ACC-65E transfers with MI71, consecutive cards must be under the same chip select.

MI71 example 1

Transferring I/O data for one ACC-65E at address \$8800 over nodes 2 and 3 of MACRO IC 0:

```
MS2,MI19=4
MS2,MI71=$10C0A0218800
MS2,MI975=$C
```



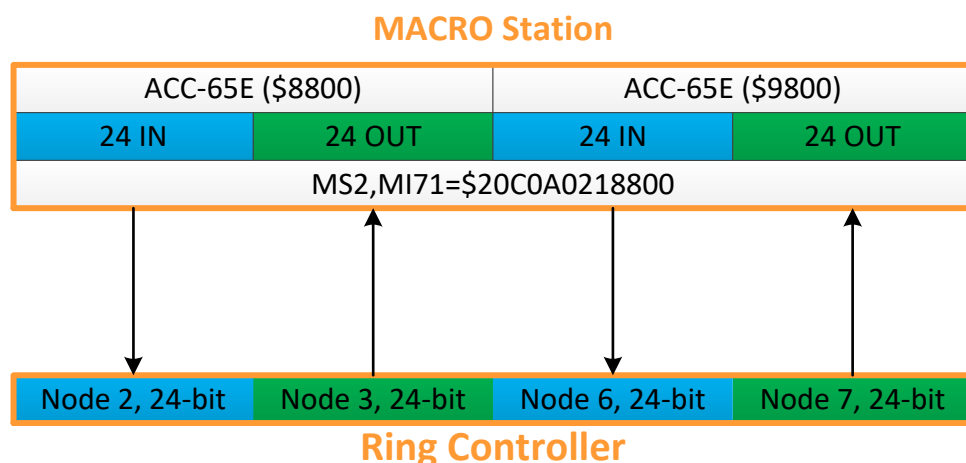
Having downloaded the above settings into the MACRO16 station, the inputs and outputs are now available to access from the ring controller side (e.g. I/O nodes 2, 3) in the following bit fields:

	Turbo		Power			
	I/O Node Address	Data Bits	Structure Element (PMAC2)	Data Bits	Structure Element (PMAC3)	Data Bits
Inputs	X:\$78420	23:00	Gate2[0].Macro[2][0]	23:00	Gate3[0].MacroInA[2][0]	31:08
Outputs	X:\$78424	23:00	Gate2[0].Macro[3][0]	23:00	Gate3[0].MacroOutA[3][0]	31:08

MI71 example 2

Transferring I/O data for two ACC-65Es at consecutive base addresses \$8800 and \$8900 over nodes 2, 3, 6, and 7 of MACRO IC 0:

```
MS2,MI19=4
MS2,MI71=$20C0A0218800
MS2,MI975=$CC
```



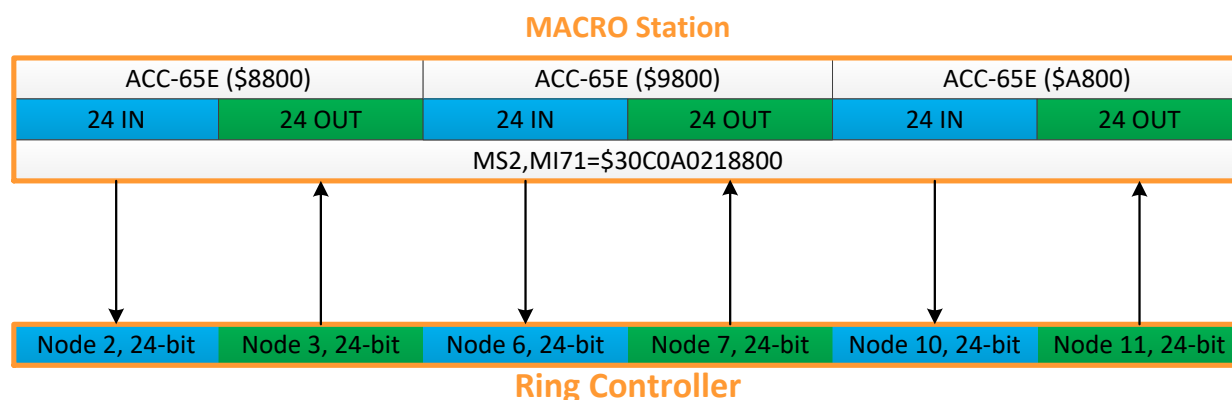
Having downloaded the above settings into the MACRO16 station, the inputs and outputs are now available to access from the ring controller side (e.g. I/O nodes 2, 3, 6, and 7) in the following bit fields:

		Turbo		Power			
		I/O Node Address	Data Bits	Structure Element (PMAC2)	Data Bits	Structure Element (PMAC3)	Data Bits
1 st ACC-65E	Inputs	X:\$78420	23:00	Gate2[0].Macro[2][0]	23:00	Gate3[0].MacroInA[2][0]	31:08
	Outputs	X:\$78424	23:00	Gate2[0].Macro[3][0]	23:00	Gate3[0].MacroOutA[3][0]	31:08
2 nd ACC-65E	Inputs	X:\$78428	23:00	Gate2[0].Macro[6][0]	23:00	Gate3[0].MacroInA[6][0]	31:08
	Outputs	X:\$7842C	23:00	Gate2[0].Macro[7][0]	23:00	Gate3[0].MacroOutA[7][0]	31:08

MI71 example 3

Transferring I/O data for three ACC-65Es at consecutive base addresses \$8800, \$9800, and \$A800 over nodes 2, 3, 6, 7, 10, and 11:

```
MS2,MI19=4
MS2,MI71=$30C0A0218800
MS2,MI975=$CCC
```



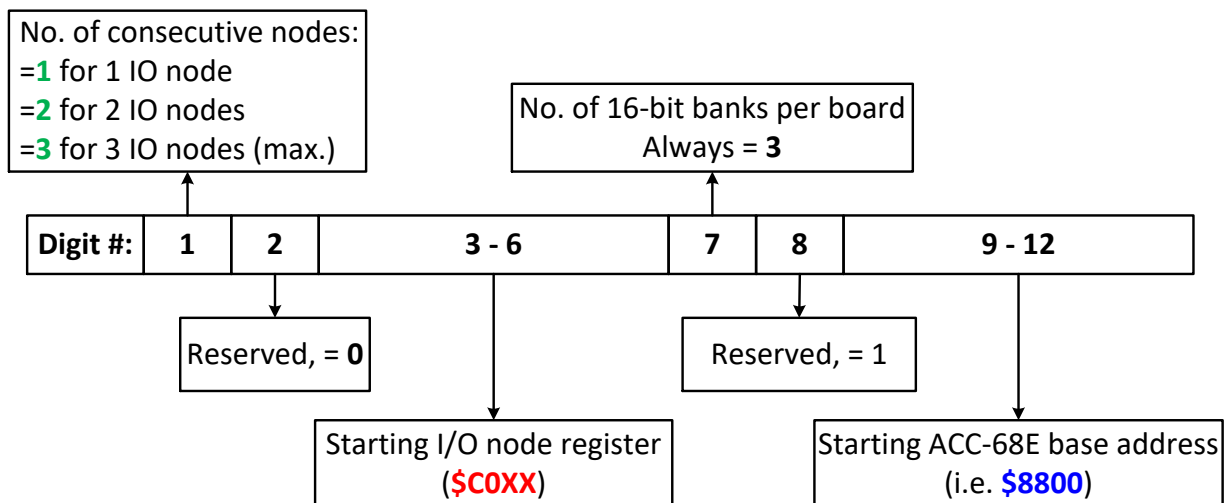
Having downloaded the above settings into the MACRO16 station, the inputs and outputs are now available to access on the ring controller side (I/O nodes 2, 3, 6, 7, 10, and 11) in the following bit fields:

		Turbo		Power			
		I/O Node Address	Data Bits	Structure Element (PMAC2)	Data Bits	Structure Element (PMAC3)	Data Bits
1 st ACC -65E	Inputs	X:\$78420	23:0 0	Gate2[0].Macro[2][0]	23:0 0	Gate3[0].MacroInA[2][0]	31:0 8
	Outputs	X:\$78424	23:0 0	Gate2[0].Macro[3][0]	23:0 0	Gate3[0].MacroOutA[3][0]	31:0 8
2 nd ACC -65E	Inputs	X:\$78428	23:0 0	Gate2[0].Macro[6][0]	23:0 0	Gate3[0].MacroInA[6][0]	31:0 8
	Outputs	X:\$7842C	23:0 0	Gate2[0].Macro[7][0]	23:0 0	Gate3[0].MacroOutA[7][0]	31:0 8
3 rd ACC -65E	Inputs	X:\$78430	23:0 0	Gate2[0].Macro[10][0]	23:0 0	Gate3[0].MacroInA[10][0]	31:0 8
	Outputs	X:\$78434	23:0 0	Gate2[0].Macro[11][0]	23:0 0	Gate3[0].MacroOutA[11][0]	31:0 8

MS{anynode},MI69/MI70

MI69/MI70 transfers all 48 bits (24 in, 24 out) of an ACC-65E into 3 x I/O node 16-bit data registers. It handles up to 3 x ACC-65Es at consecutive base addresses (e.g. Y:\$8800, Y:\$8900, Y:\$8A00), and places the data in 9 x consecutive 16-bit I/O node data registers.

MI69/70 are 48-bit variables represented as 12 hexadecimal digits which are set up as follows (digit #1 is leftmost when constructing the word):

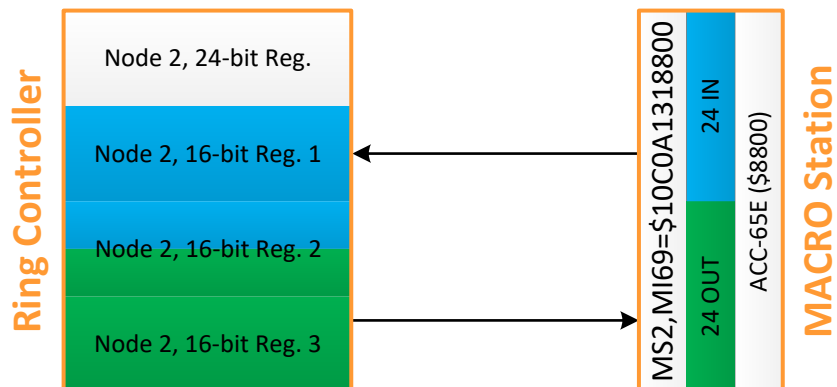
**Note**

For multiple ACC-65E transfers with MI69/MI70, consecutive cards must be under the same chip select.

MI69/70 example 1

Transferring I/O data for one ACC-65E at address \$8800 over node 2 of MACRO IC 0:

```
MS2,MI19=4
MS2,MI69=$10C0A1318800
MS2,MI975=$4
```



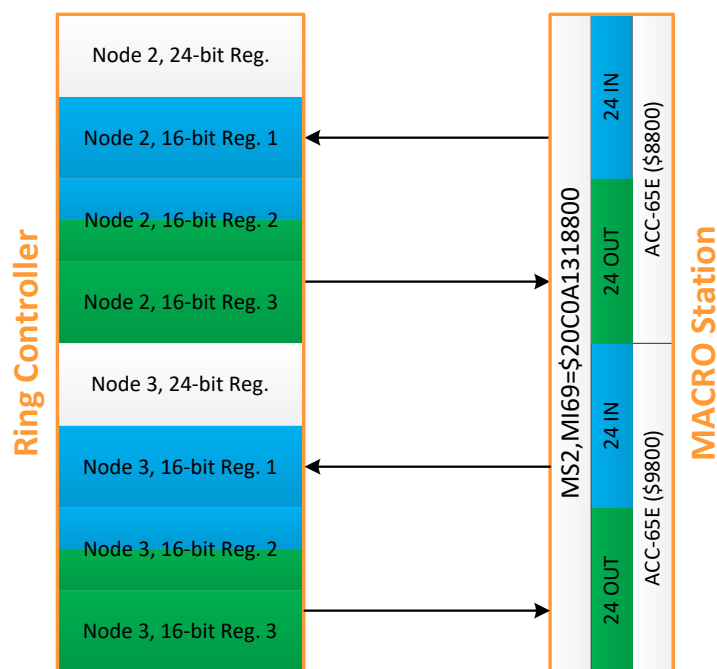
Having downloaded the above settings into the MACRO16 station, the inputs and outputs are now available to access on the ring controller side (I/O node 2) in the following bit fields:

	Turbo		Power			
	I/O Node Address	Data Bits	Structure Element (PMAC2)	Data Bits	Structure Element (PMAC3)	Data Bits
Inputs	X:\$78421	23:08	Gate2[0].Macro[2][1]	23:08	Gate3[0].MacroInA[2][1]	31:16
	X:\$78422	15:08	Gate2[0].Macro[2][2]	15:08	Gate3[0].MacroInA[2][2]	23:16
Outputs	X:\$78422	23:16	Gate2[0].Macro[2][2]	23:16	Gate3[0].MacroOutA[2][2]	31:24
	X:\$78423	23:08	Gate2[0].Macro[2][3]	23:08	Gate3[0].MacroOutA[2][3]	31:16

MI69/70 example 2

Transferring I/O data for two ACC-65Es at consecutive base addresses \$8800 and \$9800 over nodes 2 and 3 of MACRO IC 0:

```
MS2,MI19=4
MS2,MI69=$20C0A1318800
MS2,MI975=$C
```



Having downloaded the above settings into the MACRO16 station, the inputs and outputs are now available to access on the ring controller side (I/O nodes 2 and 3) in the following bit fields:

		Turbo		Power			
		I/O Node Address	Data Bits	Structure Element (PMAC2)	Data Bits	Structure Element (PMAC3)	Data Bits
1 st ACC-65E	Inputs	X:\$78421	23:08	Gate2[0].Macro[2][1]	23:08	Gate3[0].MacroInA[2][1]	31:16
		X:\$78422	16:08	Gate2[0].Macro[2][2]	16:08	Gate3[0].MacroInA[2][2]	23:16
	Outputs	X:\$78422	23:16	Gate2[0].Macro[2][2]	23:16	Gate3[0].MacroOutA[2][2]	31:24
		X:\$78423	23:08	Gate2[0].Macro[2][3]	23:08	Gate3[0].MacroOutA[2][3]	31:16

2 nd ACC-65E	Inputs	X:\$78425	23:08	Gate2[0].Macro[3][1]	23:08	Gate3[0].MacroInA[3][1]	31:16
		X:\$78426	15:08	Gate2[0].Macro[3][2]	15:08	Gate3[0].MacroInA[3][2]	23:16
	Outputs	X:\$78426	23:16	Gate2[0].Macro[3][2]	23:16	Gate3[0].MacroOutA[3][2]	31:24
		X:\$78427	23:08	Gate2[0].Macro[3][3]	23:08	Gate3[0].MacroOutA[3][3]	31:16

Accessing the Transferred Data

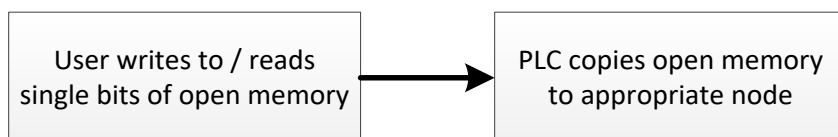
Having transferred the ACC-65E I/O data to and from the MACRO station and into the ring controller's registers mentioned above, bitwise mapping is the final step.

- With the PMAC2 style MACRO IC:
The inputs can be bitwise mapped and read.
The output state is not reported, and writing to two separate bits simultaneously (without using the full word) is not possible, thus an image word is required.
- With the PMAC3 style MACRO IC:
The inputs can be bitwise mapped and read.
The outputs can be directly bitwise mapped, written to, and reported.

Outputs Mirror Image Concept

A mirror image is a memory location on the ring controller, typically open or scratch memory that mimics the state of the ACC-65E outputs (residing in the MACRO station). This can be done in a PLC program. The following are recommended memory registers to use:

- Turbo PMAC Open Memory Registers (24 bits) **\$10F0-\$10FF** (X and Y)
- Power PMAC Open Memory Registers (32 bits), unsigned **Sys.Udata[i]**
Use Sys.Udata[1], Sys.Udata[2], Sys.Udata[3]...



Bitwise mapping to user memory structure elements in Power PMAC is not allowed. Explicit syntax with offset address must be used. The offset address is found by multiplying the element structure index by 4. To the right is an example of the first 4 unsigned open memory structure elements and their corresponding address offsets:

Structure Element	Address Offset
Sys.Udata[1]	U.USER:4
Sys.Udata[2]	U.USER: 8
Sys.Udata[3]	U.USER:12
Sys.Udata[4]	U.USER:16

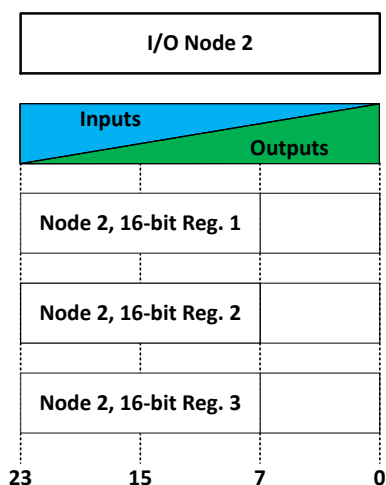


Note

A large number of self-addressed (default **Sys.pushm**) pointers in Power PMAC use **Sys.Udata[0]**; therefore, it is highly advised NOT to use it as a general purpose user memory.

Turbo PMAC2 MI160 Mapping Example

MI160 is used to transfer ACC-65E data through MACRO IC 0 node 2 of the ring controller, which places the 24 bits of inputs and 24 bits of outputs in the same 24-bit data register:



The inputs are mapped into the 24-bit register of node 2

```
#define Input1 M7001      Input1->X:$78420,0,1
#define Input2 M7002      Input2->X:$78420,1,1
#define Input3 M7003      Input3->X:$78420,2,1
#define Input4 M7004      Input4->X:$78420,3,1
#define Input5 M7005      Input5->X:$78420,4,1
#define Input6 M7006      Input6->X:$78420,5,1
#define Input7 M7007      Input7->X:$78420,6,1
#define Input8 M7008      Input8->X:$78420,7,1
#define Input9 M7009      Input9->X:$78420,8,1
#define Input10 M7010     Input10->X:$78420,9,1
#define Input11 M7011     Input11->X:$78420,10,1
#define Input12 M7012     Input12->X:$78420,11,1
#define Input13 M7013     Input13->X:$78420,12,1
#define Input14 M7014     Input14->X:$78420,13,1
#define Input15 M7015     Input15->X:$78420,14,1
#define Input16 M7016     Input16->X:$78420,15,1
#define Input17 M7017     Input17->X:$78420,16,1
#define Input18 M7018     Input18->X:$78420,17,1
#define Input19 M7019     Input19->X:$78420,18,1
#define Input20 M7020     Input20->X:$78420,19,1
#define Input21 M7021     Input21->X:$78420,20,1
#define Input22 M7022     Input22->X:$78420,21,1
#define Input23 M7023     Input23->X:$78420,22,1
#define Input24 M7024     Input24->X:$78420,23,1
```

The outputs require an image word.

We will use **Y:\$10FF** (24 bits) open memory register; therefore, the bitwise mapping of the outputs should point to the open memory.

#define Output1	M7025	Output1->Y:\$10FF,0,1
#define Output2	M7026	Output2->Y:\$10FF,1,1
#define Output3	M7027	Output3->Y:\$10FF,2,1
#define Output4	M7028	Output4->Y:\$10FF,3,1
#define Output5	M7029	Output5->Y:\$10FF,4,1
#define Output6	M7030	Output6->Y:\$10FF,5,1
#define Output7	M7031	Output7->Y:\$10FF,6,1
#define Output8	M7032	Output8->Y:\$10FF,7,1
#define Output9	M7033	Output9->Y:\$10FF,8,1
#define Output10	M7034	Output10->Y:\$10FF,9,1
#define Output11	M7035	Output11->Y:\$10FF,10,1
#define Output12	M7036	Output12->Y:\$10FF,11,1
#define Output13	M7037	Output13->Y:\$10FF,12,1
#define Output14	M7038	Output14->Y:\$10FF,13,1
#define Output15	M7039	Output15->Y:\$10FF,14,1
#define Output16	M7040	Output16->Y:\$10FF,15,1
#define Output17	M7041	Output17->Y:\$10FF,16,1
#define Output18	M7042	Output18->Y:\$10FF,17,1
#define Output19	M7043	Output19->Y:\$10FF,18,1
#define Output20	M7044	Output20->Y:\$10FF,19,1
#define Output21	M7045	Output21->Y:\$10FF,20,1
#define Output22	M7046	Output22->Y:\$10FF,21,1
#define Output23	M7047	Output23->Y:\$10FF,22,1
#define Output24	M7048	Output24->Y:\$10FF,23,1

The following mirror image PLC (which should be executing constantly) will copy the outputs into node 2:

```
#define N2Twenty4      M7049    ; Node 2, 24-bit data register
#define OutMirror      M7050    ; Mirror word, open memory

N2Twenty4->Y:$78420,0,24    ; Node 2, 24-bit data register
OutMirror->Y:$10FF,0,24     ; Open memory register
OutMirror = 0               ; Save/Initialize to zero or desired state

Open PLC 1 Clear
N2Twenty4 = OutMirror       ; Update data register
Close
```

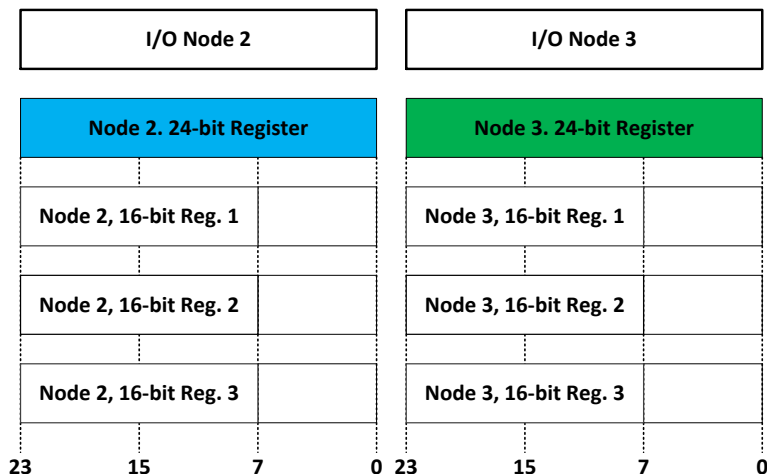


Note

The output states are now reported and can now be toggled individually using the bitwise assignments to the open memory register.

Turbo PMAC2 MI71 Mapping Example

MI71 is used to transfer ACC-65E data through MACRO IC 0 nodes 2 and 3 of the ring controller, which places the 24 bits of inputs in the 24-bit register of node 2 and the 24 bits of outputs in the 24-bit register of node 3:



The inputs are mapped into the 24-bit register of node 2

```
#define Input1 M7001 Input1->X:$78420,0,1
#define Input2 M7002 Input2->X:$78420,1,1
#define Input3 M7003 Input3->X:$78420,2,1
#define Input4 M7004 Input4->X:$78420,3,1
#define Input5 M7005 Input5->X:$78420,4,1
#define Input6 M7006 Input6->X:$78420,5,1
#define Input7 M7007 Input7->X:$78420,6,1
#define Input8 M7008 Input8->X:$78420,7,1
#define Input9 M7009 Input9->X:$78420,8,1
#define Input10 M7010 Input10->X:$78420,9,1
#define Input11 M7011 Input11->X:$78420,10,1
#define Input12 M7012 Input12->X:$78420,11,1
#define Input13 M7013 Input13->X:$78420,12,1
#define Input14 M7014 Input14->X:$78420,13,1
#define Input15 M7015 Input15->X:$78420,14,1
#define Input16 M7016 Input16->X:$78420,15,1
#define Input17 M7017 Input17->X:$78420,16,1
#define Input18 M7018 Input18->X:$78420,17,1
#define Input19 M7019 Input19->X:$78420,18,1
#define Input20 M7020 Input20->X:$78420,19,1
#define Input21 M7021 Input21->X:$78420,20,1
#define Input22 M7022 Input22->X:$78420,21,1
#define Input23 M7023 Input23->X:$78420,22,1
#define Input24 M7024 Input24->X:$78420,23,1
```

The outputs require an image word.

We will use **Y:\$10FF** (24 bits) open memory register (to copy the 24 bits of node 3); therefore, the bitwise mapping of the outputs should point to the open memory.

#define Output1	M7025	Output1->Y:\$10FF,0,1
#define Output2	M7026	Output2->Y:\$10FF,1,1
#define Output3	M7027	Output3->Y:\$10FF,2,1
#define Output4	M7028	Output4->Y:\$10FF,3,1
#define Output5	M7029	Output5->Y:\$10FF,4,1
#define Output6	M7030	Output6->Y:\$10FF,5,1
#define Output7	M7031	Output7->Y:\$10FF,6,1
#define Output8	M7032	Output8->Y:\$10FF,7,1
#define Output9	M7033	Output9->Y:\$10FF,8,1
#define Output10	M7034	Output10->Y:\$10FF,9,1
#define Output11	M7035	Output11->Y:\$10FF,10,1
#define Output12	M7036	Output12->Y:\$10FF,11,1
#define Output13	M7037	Output13->Y:\$10FF,12,1
#define Output14	M7038	Output14->Y:\$10FF,13,1
#define Output15	M7039	Output15->Y:\$10FF,14,1
#define Output16	M7040	Output16->Y:\$10FF,15,1
#define Output17	M7041	Output17->Y:\$10FF,16,1
#define Output18	M7042	Output18->Y:\$10FF,17,1
#define Output19	M7043	Output19->Y:\$10FF,18,1
#define Output20	M7044	Output20->Y:\$10FF,19,1
#define Output21	M7045	Output21->Y:\$10FF,20,1
#define Output22	M7046	Output22->Y:\$10FF,21,1
#define Output23	M7047	Output23->Y:\$10FF,22,1
#define Output24	M7048	Output24->Y:\$10FF,23,1

The following mirror image PLC (which should be executing constantly) will copy the outputs into node 3:

```
#define N3Twenty4      M7049      ; Node 3, 24-bit register
#define OutMirror      M7050      ; Mirror word, open memory

N3Twenty4->Y:$78424,0,24      ; 24-bit register, node 3
OutMirror->Y:$10FF,0,24      ; Open memory register
OutMirror = 0                  ; Save/Initialize to zero or desired state

Open PLC 1 Clear
N3Twenty4 = OutMirror          ; Update data register
Close
```



Note

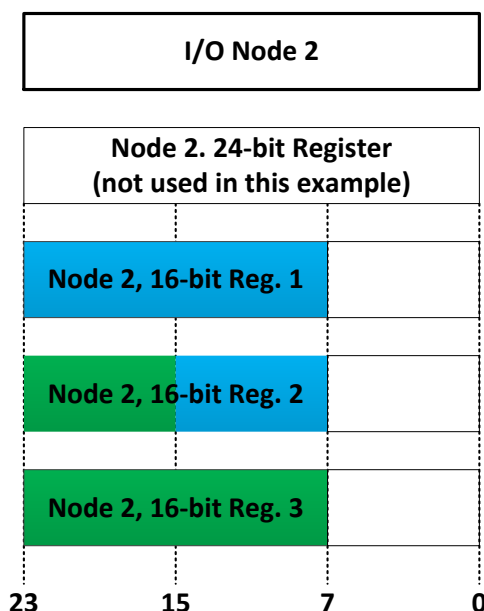
The output states are now reported and can be toggled individually using the bitwise assignments to the open memory register.

Turbo PMAC2 MI69/70 Mapping Example

MI69 / MI70 is used to transfer ACC-66E data through MACRO IC 0 node 2 of the ring controller, which places the data in registers as illustrated on the right:

The lower 16 bits of the 24 bits of inputs reside in the first 16-bit register, and the upper 8 bits of the 24 bits of inputs reside in the lower byte of the second 16-bit register.

The lower 8 bits of the 24 bits of outputs reside in the upper byte of the second 16-bit register, and the upper 16 bits of the 24 bits of outputs reside in the third 16-bit register.



The inputs are mapped into 16-bit registers of node 2:

#define Input1 M7001	Input1->X:\$78421,8,1
#define Input2 M7002	Input2->X:\$78421,9,1
#define Input3 M7003	Input3->X:\$78421,10,1
#define Input4 M7004	Input4->X:\$78421,11,1
#define Input5 M7005	Input5->X:\$78421,12,1
#define Input6 M7006	Input6->X:\$78421,13,1
#define Input7 M7007	Input7->X:\$78421,14,1
#define Input8 M7008	Input8->X:\$78421,15,1
#define Input9 M7009	Input9->X:\$78421,16,1
#define Input10 M7010	Input10->X:\$78421,17,1
#define Input11 M7011	Input11->X:\$78421,18,1
#define Input12 M7012	Input12->X:\$78421,19,1
#define Input13 M7013	Input13->X:\$78421,20,1
#define Input14 M7014	Input14->X:\$78421,21,1
#define Input15 M7015	Input15->X:\$78421,22,1
#define Input16 M7016	Input16->X:\$78421,23,1
#define Input17 M7017	Input17->X:\$78422,8,1
#define Input18 M7018	Input18->X:\$78422,9,1
#define Input19 M7019	Input19->X:\$78422,10,1
#define Input20 M7020	Input20->X:\$78422,11,1
#define Input21 M7021	Input21->X:\$78422,12,1
#define Input22 M7022	Input22->X:\$78422,13,1
#define Input23 M7023	Input23->X:\$78422,14,1
#define Input24 M7024	Input24->X:\$78422,15,1

The outputs require an image word

We will use **Y:\$10FF** open memory register to create an image word; therefore, the bitwise mapping of the outputs should point to the open memory.

#define Output1 M7025	Output1->Y:\$10FF,0,1
#define Output2 M7026	Output2->Y:\$10FF,1,1
#define Output3 M7027	Output3->Y:\$10FF,2,1
#define Output4 M7028	Output4->Y:\$10FF,3,1
#define Output5 M7029	Output5->Y:\$10FF,4,1
#define Output6 M7030	Output6->Y:\$10FF,5,1
#define Output7 M7031	Output7->Y:\$10FF,6,1
#define Output8 M7032	Output8->Y:\$10FF,7,1
#define Output9 M7033	Output9->Y:\$10FF,8,1
#define Output10 M7034	Output10->Y:\$10FF,9,1
#define Output11 M7035	Output11->Y:\$10FF,10,1
#define Output12 M7036	Output12->Y:\$10FF,11,1
#define Output13 M7037	Output13->Y:\$10FF,12,1
#define Output14 M7038	Output14->Y:\$10FF,13,1
#define Output15 M7039	Output15->Y:\$10FF,14,1
#define Output16 M7040	Output16->Y:\$10FF,15,1
#define Output17 M7041	Output17->Y:\$10FF,16,1
#define Output18 M7042	Output18->Y:\$10FF,17,1
#define Output19 M7043	Output19->Y:\$10FF,18,1
#define Output20 M7044	Output20->Y:\$10FF,19,1
#define Output21 M7045	Output21->Y:\$10FF,20,1
#define Output22 M7046	Output22->Y:\$10FF,21,1
#define Output23 M7047	Output23->Y:\$10FF,22,1
#define Output24 M7048	Output24->Y:\$10FF,23,1

The following mirror image PLC (which should be executing constantly) will copy the outputs into node 2:

```
#define N2Second16 M7049 ; Node 2, second 16-bit register
#define N2Third16 M7050 ; Node 2, third 16-bit register
#define OutMirror M7051 ; Mirror word, open memory

N2Second16->X:$78422,8,16 ; Second 16-bit register, node 2
N2Third16->X:$78423,8,16 ; Third 16-bit register, node 2
OutMirror->Y:$10FF,0,24 ; Open memory
OutMirror = 0 ; Save/Initialize to zero or desired state

Open PLC 1 Clear
N2Second16 = OutMirror & $0000FF ; Mask with lower 8 bits
N2Third16 = OutMirror & $FFFF00/$100 ; Mask OutMirror with upper 16 bits and shift to lower 16
Close
```

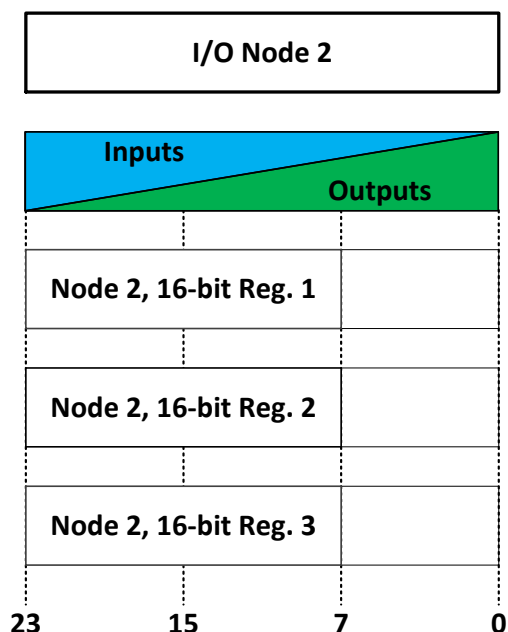


Note

The output states are now reported and can be toggled individually using the bitwise assignments to the open memory register.

Power PMAC2 MI160 Mapping Example

MI160 is used to transfer ACC-65E data through MACRO IC 0 node 2 of the ring controller, which places the 24 bits of inputs and 24 bits of outputs in the same 24-bit data register:



The inputs are mapped into the 24 bits of node 2:

```
PTR Input1->Gate2[0].Macro[2][0].0.1;
PTR Input2->Gate2[0].Macro[2][0].1.1;
PTR Input3->Gate2[0].Macro[2][0].2.1;
PTR Input4->Gate2[0].Macro[2][0].3.1;
PTR Input5->Gate2[0].Macro[2][0].4.1;
PTR Input6->Gate2[0].Macro[2][0].5.1;
PTR Input7->Gate2[0].Macro[2][0].6.1;
PTR Input8->Gate2[0].Macro[2][0].7.1;
PTR Input9->Gate2[0].Macro[2][0].8.1;
PTR Input10->Gate2[0].Macro[2][0].9.1;
PTR Input11->Gate2[0].Macro[2][0].10.1;
PTR Input12->Gate2[0].Macro[2][0].11.1;
PTR Input13->Gate2[0].Macro[2][0].12.1;
PTR Input14->Gate2[0].Macro[2][0].13.1;
PTR Input15->Gate2[0].Macro[2][0].14.1;
PTR Input16->Gate2[0].Macro[2][0].15.1;
PTR Input17->Gate2[0].Macro[2][0].16.1;
PTR Input18->Gate2[0].Macro[2][0].17.1;
PTR Input19->Gate2[0].Macro[2][0].18.1;
PTR Input20->Gate2[0].Macro[2][0].19.1;
PTR Input21->Gate2[0].Macro[2][0].20.1;
PTR Input22->Gate2[0].Macro[2][0].21.1;
PTR Input23->Gate2[0].Macro[2][0].22.1;
PTR Input24->Gate2[0].Macro[2][0].23.1;
```



Note

Bitwise mapping with PMAC2 style MACRO IC is supported starting with Power PMAC firmware version **1.5.8.215**.

The outputs require an image word.

We will use **Sys.Udata[1]** open memory register to create the image word; therefore, the bitwise mapping of the outputs:

```
PTR Output1->U.USER:4.0.1;
PTR Output2->U.USER:4.1.1;
PTR Output3->U.USER:4.2.1;
PTR Output4->U.USER:4.3.1;
PTR Output5->U.USER:4.4.1;
PTR Output6->U.USER:4.5.1;
PTR Output7->U.USER:4.6.1;
PTR Output8->U.USER:4.7.1;
PTR Output9->U.USER:4.8.1;
PTR Output10->U.USER:4.9.1;
PTR Output11->U.USER:4.10.1;
PTR Output12->U.USER:4.11.1;
PTR Output13->U.USER:4.12.1;
PTR Output14->U.USER:4.13.1;
PTR Output15->U.USER:4.14.1;
PTR Output16->U.USER:4.15.1;
PTR Output17->U.USER:4.16.1;
PTR Output18->U.USER:4.17.1;
PTR Output19->U.USER:4.18.1;
PTR Output20->U.USER:4.19.1;
PTR Output21->U.USER:4.20.1;
PTR Output22->U.USER:4.21.1;
PTR Output23->U.USER:4.22.1;
PTR Output24->U.USER:4.23.1;
```

Bitwise mapping to user memory structure elements is not allowed. Explicit syntax with offset address must be used. The offset address is found by multiplying the element structure index by 4. To the right is an example of the first 4 unsigned open memory structure elements and their corresponding address offsets:

Structure Element	Address Offset
Sys.Udata[1]	U.USER:4
Sys.Udata[2]	U.USER:8
Sys.Udata[3]	U.USER:12
Sys.Udata[4]	U.USER:16



Note

A large number of self-addressed (default **Sys.pushm**) pointers in Power PMAC use **Sys.Udata[0]**; therefore, it is highly advised NOT to use it as general purpose user memory.

The following mirror image PLC (which should be executing constantly) will copy the outputs into node 2:

```
PTR N2Twenty4->Gate2[0].Macro[2][0]; // node 2 (shared with inputs)
PTR OutMirror->U.USER:4.0.24; // Mirror Word, shared user memory
OutMirror = 0 // Save/initialize to zero or desired state

Open PLC 1
N2Twenty4 = OutMirror // Update data register
Close
```

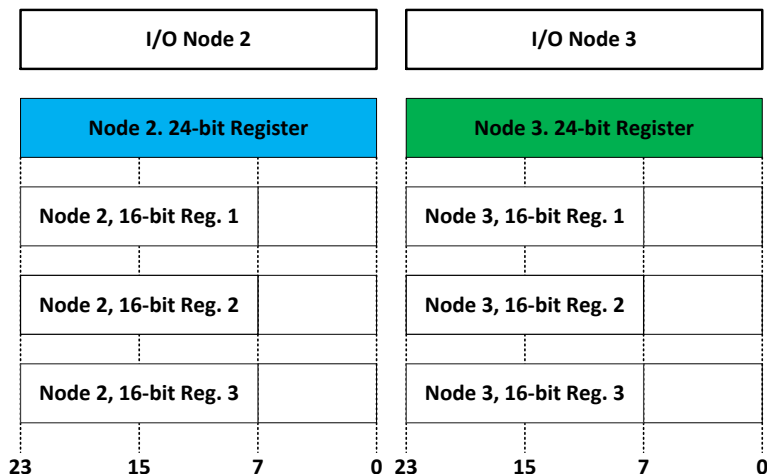


Note

The output states are now reported and can be toggled individually using the bitwise assignments to the open memory register.

Power PMAC2 MI71 Mapping Example

MI71 is used to transfer ACC-65E data to MACRO IC 0 nodes 2 and 3 of the ring controller, which places the 24 bits of inputs in the 24-bit register of node 2 and the 24 bits of outputs in the 24-bit register of node 3:



The inputs are mapped into the 24 bits of node 2:

```
PTR Input1->Gate2[0].Macro[2][0].0.1;
PTR Input2->Gate2[0].Macro[2][0].1.1;
PTR Input3->Gate2[0].Macro[2][0].2.1;
PTR Input4->Gate2[0].Macro[2][0].3.1;
PTR Input5->Gate2[0].Macro[2][0].4.1;
PTR Input6->Gate2[0].Macro[2][0].5.1;
PTR Input7->Gate2[0].Macro[2][0].6.1;
PTR Input8->Gate2[0].Macro[2][0].7.1;
PTR Input9->Gate2[0].Macro[2][0].8.1;
PTR Input10->Gate2[0].Macro[2][0].9.1;
PTR Input11->Gate2[0].Macro[2][0].10.1;
PTR Input12->Gate2[0].Macro[2][0].11.1;
PTR Input13->Gate2[0].Macro[2][0].12.1;
PTR Input14->Gate2[0].Macro[2][0].13.1;
PTR Input15->Gate2[0].Macro[2][0].14.1;
PTR Input16->Gate2[0].Macro[2][0].15.1;
PTR Input17->Gate2[0].Macro[2][0].16.1;
PTR Input18->Gate2[0].Macro[2][0].17.1;
PTR Input19->Gate2[0].Macro[2][0].18.1;
PTR Input20->Gate2[0].Macro[2][0].19.1;
PTR Input21->Gate2[0].Macro[2][0].20.1;
PTR Input22->Gate2[0].Macro[2][0].21.1;
PTR Input23->Gate2[0].Macro[2][0].22.1;
PTR Input24->Gate2[0].Macro[2][0].23.1;
```



Note

Bitwise mapping with PMAC2 style MACRO IC is supported starting with Power PMAC firmware version 1.5.8.215.

The outputs require an image word.

We will use **Sys.Udata[1]** open memory register (to copy the 24 bits of node 3) to create the image word. Therefore, the bitwise mapping of the outputs:

```
PTR Output1->U.USER:4.0.1;
PTR Output2->U.USER:4.1.1;
PTR Output3->U.USER:4.2.1;
PTR Output4->U.USER:4.3.1;
PTR Output5->U.USER:4.4.1;
PTR Output6->U.USER:4.5.1;
PTR Output7->U.USER:4.6.1;
PTR Output8->U.USER:4.7.1;
PTR Output9->U.USER:4.8.1;
PTR Output10->U.USER:4.9.1;
PTR Output11->U.USER:4.10.1;
PTR Output12->U.USER:4.11.1;
PTR Output13->U.USER:4.12.1;
PTR Output14->U.USER:4.13.1;
PTR Output15->U.USER:4.14.1;
PTR Output16->U.USER:4.15.1;
PTR Output17->U.USER:4.16.1;
PTR Output18->U.USER:4.17.1;
PTR Output19->U.USER:4.18.1;
PTR Output20->U.USER:4.19.1;
PTR Output21->U.USER:4.20.1;
PTR Output22->U.USER:4.21.1;
PTR Output23->U.USER:4.22.1;
PTR Output24->U.USER:4.23.1;
```

Bitwise mapping to user memory structure elements is not allowed. Explicit syntax with offset address must be used. The offset address is found by multiplying the element structure index by 4. To the right is an example of the first 4 unsigned open memory structure elements and their corresponding address offsets:

Structure Element	Address Offset
Sys.Udata[1]	U.USER:4
Sys.Udata[2]	U.USER:8
Sys.Udata[3]	U.USER:12
Sys.Udata[4]	U.USER:16



Note

A large number of self-addressed (default **Sys.pushm**) pointers in Power PMAC use **Sys.Udata[0]**; therefore, it is highly advised NOT to use it as general purpose user memory.

The following PLC (which should be executing constantly) will copy the outputs into node 3:

```
PTR N3Twenty4->Gate2[0].Macro[3][0]; // 24-bit register, node 3
PTR OutMirror->U.USER:4.0.24; // Mirror Word, shared user memory
OutMirror = 0 // Save/initialize to zero or desired state

Open PLC 1
N3Twenty4 = OutMirror // Update data register
Close
```



Note

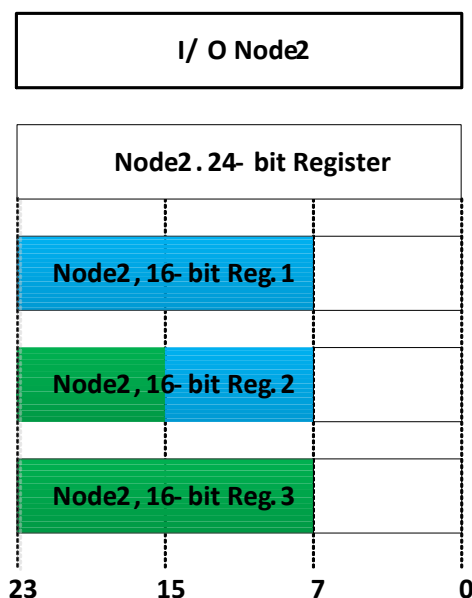
The output states are now reported and can be toggled individually using the bitwise assignments to the open memory register.

Power PMAC2 MI69/70 Mapping Example

MI69 / MI70 is used to transfer ACC-65E data through MACRO IC 0 node 2 of the ring controller, which places the data as illustrated on the right:

The lower 16 bits of the 24 bits of inputs reside in the first 16-bit register, and the upper 8 bits of the 24 bits of inputs reside in the lower byte of the second 16-bit register.

The lower 8 bits of the 24 bits of outputs reside in the upper byte of the second 16-bit register, and the upper 16 bits of the 24 bits of outputs reside in the third 16-bit register.



The inputs are mapped into 16-bit registers of node 2:

```
PTR Input1->Gate2[0].Macro[2][1].8.1
PTR Input2->Gate2[0].Macro[2][1].9.1
PTR Input3->Gate2[0].Macro[2][1].10.1
PTR Input4->Gate2[0].Macro[2][1].11.1
PTR Input5->Gate2[0].Macro[2][1].12.1
PTR Input6->Gate2[0].Macro[2][1].13.1
PTR Input7->Gate2[0].Macro[2][1].14.1
PTR Input8->Gate2[0].Macro[2][1].15.1
PTR Input9->Gate2[0].Macro[2][1].16.1
PTR Input10->Gate2[0].Macro[2][1].17.1
PTR Input11->Gate2[0].Macro[2][1].18.1
PTR Input12->Gate2[0].Macro[2][1].19.1
PTR Input13->Gate2[0].Macro[2][1].20.1
PTR Input14->Gate2[0].Macro[2][1].21.1
PTR Input15->Gate2[0].Macro[2][1].22.1
PTR Input16->Gate2[0].Macro[2][1].23.1
PTR Input17->Gate2[0].Macro[2][2].8.1
PTR Input18->Gate2[0].Macro[2][2].9.1
PTR Input19->Gate2[0].Macro[2][2].10.1
PTR Input20->Gate2[0].Macro[2][2].11.1
PTR Input21->Gate2[0].Macro[2][2].12.1
PTR Input22->Gate2[0].Macro[2][2].13.1
PTR Input23->Gate2[0].Macro[2][2].14.1
PTR Input24->Gate2[0].Macro[2][2].15.1
```



Note

Bitwise mapping with PMAC2 style MACRO IC is supported starting with Power PMAC firmware version **1.5.8.215**.

The outputs require an image word. We will use **Sys.Udata[1]** open memory register to create the image word. Therefore, the bitwise mapping of the outputs:

```
PTR Output1->U.USER:4.0.1;
PTR Output2->U.USER:4.1.1;
PTR Output3->U.USER:4.2.1;
PTR Output4->U.USER:4.3.1;
PTR Output5->U.USER:4.4.1;
PTR Output6->U.USER:4.5.1;
PTR Output7->U.USER:4.6.1;
PTR Output8->U.USER:4.7.1;
PTR Output9->U.USER:4.8.1;
PTR Output10->U.USER:4.9.1;
PTR Output11->U.USER:4.10.1;
PTR Output12->U.USER:4.11.1;
PTR Output13->U.USER:4.12.1;
PTR Output14->U.USER:4.13.1;
PTR Output15->U.USER:4.14.1;
PTR Output16->U.USER:4.15.1;
PTR Output17->U.USER:4.16.1;
PTR Output18->U.USER:4.17.1;
PTR Output19->U.USER:4.18.1;
PTR Output20->U.USER:4.19.1;
PTR Output21->U.USER:4.20.1;
PTR Output22->U.USER:4.21.1;
PTR Output23->U.USER:4.22.1;
PTR Output24->U.USER:4.23.1;
```

Bitwise mapping to user memory structure elements is not allowed. Explicit syntax with offset address must be used. The offset address is found by multiplying the element structure index by 4. To the right is an example of the first 4 unsigned open memory structure elements and their corresponding address offsets:

Structure Element	Address Offset
Sys.Udata[1]	U.USER:4
Sys.Udata[2]	U.USER:8
Sys.Udata[3]	U.USER:12
Sys.Udata[4]	U.USER:16



Note

A large number of self-addressed (default **Sys.pushm**) pointers in Power PMAC use **Sys.Udata[0]**; therefore, it is highly advised NOT to use it as general purpose user memory.

The following mirror image PLC (which should be executing constantly) will copy the outputs into node 2:

```
PTR N2Second16->Gate2[0].Macro[2][2]; // Node 2, Second 16-bit register (upper 8 bits)
PTR N2Third16->Gate2[0].Macro[2][3]; // Node 2, Third 16-bit register (upper 16 bits)
PTR OutMirror->U.USER:4.0.24; // Mirror Word, shared user memory
OutMirror = 0 // Save/initialize to zero or desired state

Open PLC 1
N2Second16 = OutMirror & $0000FF <<16 // Update 2nd 16-bit reg; shift to upper 16-bits
N2Third16 = OutMirror & $FFFF00 // Update node 2 third 16-bit register
Close
```

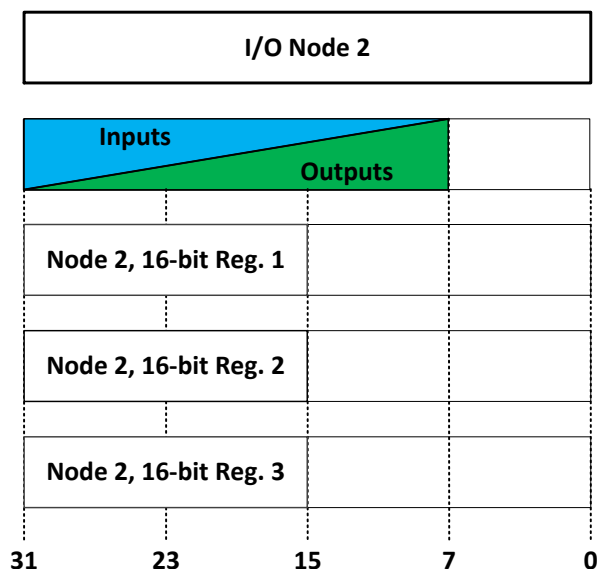


Note

The output states are now reported and can be toggled individually using the bitwise assignments to the open memory register.

Power PMAC3 MI160 Mapping Example

MI160 is used to transfer ACC-65E data through MACRO IC 0 Bank A node 2 of the ring controller, which places the 24 bits of inputs in the **IN** 24-bit data register of node 2, and the 24 bits of outputs in the **OUT** 24-bit data register of node 2:



Inputs

```
PTR Input1->Gate3[0].MacroInA[2][0].8.1;
PTR Input2->Gate3[0].MacroInA[2][0].9.1;
PTR Input3->Gate3[0].MacroInA[2][0].10.1;
PTR Input4->Gate3[0].MacroInA[2][0].11.1;
PTR Input5->Gate3[0].MacroInA[2][0].12.1;
PTR Input6->Gate3[0].MacroInA[2][0].13.1;
PTR Input7->Gate3[0].MacroInA[2][0].14.1;
PTR Input8->Gate3[0].MacroInA[2][0].15.1;
PTR Input9->Gate3[0].MacroInA[2][0].16.1;
PTR Input10->Gate3[0].MacroInA[2][0].17.1;
PTR Input11->Gate3[0].MacroInA[2][0].18.1;
PTR Input12->Gate3[0].MacroInA[2][0].19.1;
PTR Input13->Gate3[0].MacroInA[2][0].20.1;
PTR Input14->Gate3[0].MacroInA[2][0].21.1;
PTR Input15->Gate3[0].MacroInA[2][0].22.1;
PTR Input16->Gate3[0].MacroInA[2][0].23.1;
PTR Input17->Gate3[0].MacroInA[2][0].24.1;
PTR Input18->Gate3[0].MacroInA[2][0].25.1;
PTR Input19->Gate3[0].MacroInA[2][0].26.1;
PTR Input20->Gate3[0].MacroInA[2][0].27.1;
PTR Input21->Gate3[0].MacroInA[2][0].28.1;
PTR Input22->Gate3[0].MacroInA[2][0].29.1;
PTR Input23->Gate3[0].MacroInA[2][0].30.1;
PTR Input24->Gate3[0].MacroInA[2][0].31.1;
```

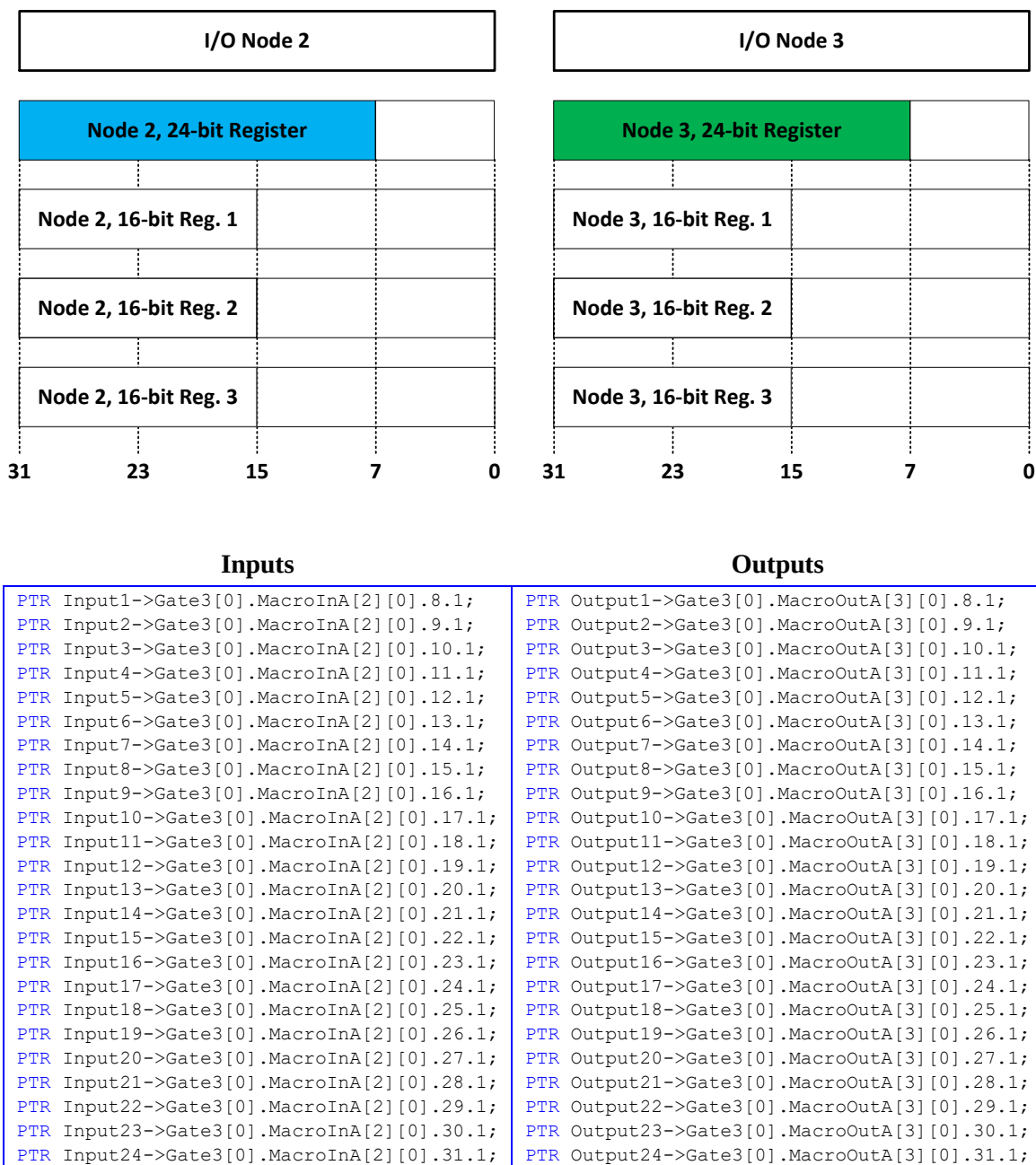
Outputs

```
PTR Output1->Gate3[0].MacroOutA[2][0].8.1;
PTR Output2->Gate3[0].MacroOutA[2][0].9.1;
PTR Output3->Gate3[0].MacroOutA[2][0].10.1;
PTR Output4->Gate3[0].MacroOutA[2][0].11.1;
PTR Output5->Gate3[0].MacroOutA[2][0].12.1;
PTR Output6->Gate3[0].MacroOutA[2][0].13.1;
PTR Output7->Gate3[0].MacroOutA[2][0].14.1;
PTR Output8->Gate3[0].MacroOutA[2][0].15.1;
PTR Output9->Gate3[0].MacroOutA[2][0].16.1;
PTR Output10->Gate3[0].MacroOutA[2][0].17.1;
PTR Output11->Gate3[0].MacroOutA[2][0].18.1;
PTR Output12->Gate3[0].MacroOutA[2][0].19.1;
PTR Output13->Gate3[0].MacroOutA[2][0].20.1;
PTR Output14->Gate3[0].MacroOutA[2][0].21.1;
PTR Output15->Gate3[0].MacroOutA[2][0].22.1;
PTR Output16->Gate3[0].MacroOutA[2][0].23.1;
PTR Output17->Gate3[0].MacroOutA[2][0].24.1;
PTR Output18->Gate3[0].MacroOutA[2][0].25.1;
PTR Output19->Gate3[0].MacroOutA[2][0].26.1;
PTR Output20->Gate3[0].MacroOutA[2][0].27.1;
PTR Output21->Gate3[0].MacroOutA[2][0].28.1;
PTR Output22->Gate3[0].MacroOutA[2][0].29.1;
PTR Output23->Gate3[0].MacroOutA[2][0].30.1;
PTR Output24->Gate3[0].MacroOutA[2][0].31.1;
```

With Power PMAC3, the outputs can be directly bitwise mapped, written to, and reported. Mirror word is not required.

Power PMAC3 MI71 Mapping Example

MI71 is used to transfer ACC-65E data through MACRO IC 0 Bank A nodes 2 and 3 of the ring controller, which places the 24 bits of inputs in the **In** 24-bit data register of node 2 and the 24 bits of outputs in the **Out** 24-bit register of node 3:



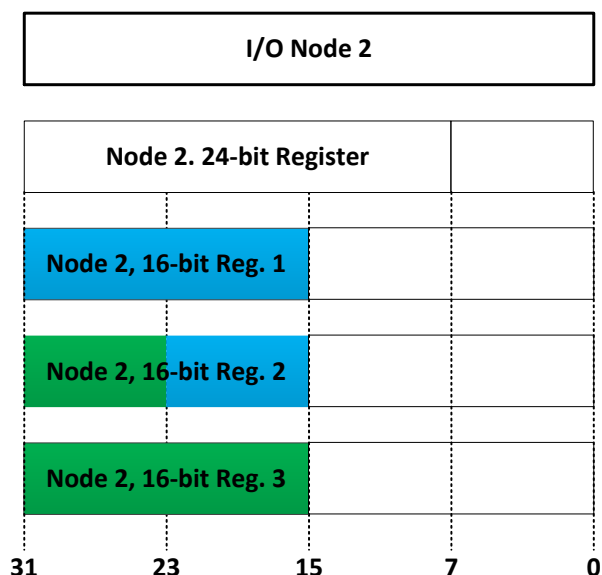
With Power PMAC3, the outputs can be directly bitwise mapped, written to, and reported. Mirror word is not required.

Power PMAC3 MI69/70 Mapping Example

MI69 / MI70 is used to transfer ACC-65E data through MACRO IC 0 Bank A node 2 of the ring controller, which places the data as illustrated on the right:

The lower 16 bits of the 24 bits of inputs reside in the first 16-bit register, and the upper 8 bits of the 24 bits of inputs reside in the lower byte of the second 16-bit register. All in **In** registers.

The lower 8 bits of the 24 bits of outputs reside in the upper byte of the second 16-bit register, and the upper 16 bits of the 24 bits of outputs reside in the third 16-bit register. All in **Out** registers.



Inputs

```
PTR Input1->Gate3[0].MacroInA[2][1].16.1;
PTR Input2->Gate3[0].MacroInA[2][1].17.1;
PTR Input3->Gate3[0].MacroInA[2][1].18.1;
PTR Input4->Gate3[0].MacroInA[2][1].19.1;
PTR Input5->Gate3[0].MacroInA[2][1].20.1;
PTR Input6->Gate3[0].MacroInA[2][1].21.1;
PTR Input7->Gate3[0].MacroInA[2][1].22.1;
PTR Input8->Gate3[0].MacroInA[2][1].23.1;
PTR Input9->Gate3[0].MacroInA[2][1].24.1;
PTR Input10->Gate3[0].MacroInA[2][1].25.1;
PTR Input11->Gate3[0].MacroInA[2][1].26.1;
PTR Input12->Gate3[0].MacroInA[2][1].27.1;
PTR Input13->Gate3[0].MacroInA[2][1].28.1;
PTR Input14->Gate3[0].MacroInA[2][1].29.1;
PTR Input15->Gate3[0].MacroInA[2][1].30.1;
PTR Input16->Gate3[0].MacroInA[2][1].31.1;
PTR Input17->Gate3[0].MacroInA[2][2].16.1;
PTR Input18->Gate3[0].MacroInA[2][2].17.1;
PTR Input19->Gate3[0].MacroInA[2][2].18.1;
PTR Input20->Gate3[0].MacroInA[2][2].19.1;
PTR Input21->Gate3[0].MacroInA[2][2].20.1;
PTR Input22->Gate3[0].MacroInA[2][2].21.1;
PTR Input23->Gate3[0].MacroInA[2][2].22.1;
PTR Input24->Gate3[0].MacroInA[2][2].23.1;
```

Outputs

```
PTR Output1->Gate3[0].MacroOutA[2][2].24.1;
PTR Output2->Gate3[0].MacroOutA[2][2].25.1;
PTR Output3->Gate3[0].MacroOutA[2][2].26.1;
PTR Output4->Gate3[0].MacroOutA[2][2].27.1;
PTR Output5->Gate3[0].MacroOutA[2][2].28.1;
PTR Output6->Gate3[0].MacroOutA[2][2].29.1;
PTR Output7->Gate3[0].MacroOutA[2][2].30.1;
PTR Output8->Gate3[0].MacroOutA[2][2].31.1;
PTR Output9->Gate3[0].MacroOutA[2][3].16.1;
PTR Output10->Gate3[0].MacroOutA[2][3].17.1;
PTR Output11->Gate3[0].MacroOutA[2][3].18.1;
PTR Output12->Gate3[0].MacroOutA[2][3].19.1;
PTR Output13->Gate3[0].MacroOutA[2][3].20.1;
PTR Output14->Gate3[0].MacroOutA[2][3].21.1;
PTR Output15->Gate3[0].MacroOutA[2][3].22.1;
PTR Output16->Gate3[0].MacroOutA[2][3].23.1;
PTR Output17->Gate3[0].MacroOutA[2][3].24.1;
PTR Output18->Gate3[0].MacroOutA[2][3].25.1;
PTR Output19->Gate3[0].MacroOutA[2][3].26.1;
PTR Output20->Gate3[0].MacroOutA[2][3].27.1;
PTR Output21->Gate3[0].MacroOutA[2][3].28.1;
PTR Output22->Gate3[0].MacroOutA[2][3].29.1;
PTR Output23->Gate3[0].MacroOutA[2][3].30.1;
PTR Output24->Gate3[0].MacroOutA[2][3].31.1;
```

With Power PMAC3, the outputs can be directly bitwise mapped, written to, and reported. Mirror word is not required.

Configuring the Control Word for MACRO

The control word is configured automatically with MACRO16 CPU firmware version 1.204 and newer. The setup below is for older firmware versions.

The control word can be configured with MACRO using MI198 and MI199:

- Set MI198 to \$40000 + {base address} + 7
- Set MI199 = 7

Configuring the Control Word must be done on every power-up/reset such as in the following example PLC which performs the configuration and then disables itself.

Example:

With ACC-65E at base addresses \$8800, and using node 2 for MACRO Station communication.

Turbo PMAC ring controllers:

```
Open PLC 1 Clear
I5111 = 250 * 8388608 / I10 WHILE (I5111 > 0) EndW

CMD"MS2,MI198=$408807"           // Point to control word of Card 1 at address $8800
CMD"MS2,MI199=7"                 // Write to control word
I5111 = 250 * 8388608 / I10 WHILE (I5111 > 0) EndW

Disable PLC 1
Close
```

Power PMAC ring controllers:

```
Open PLC 1
CALL Timer(0.250)                // 250 msec delay. Requires loading the Timer subprogram.

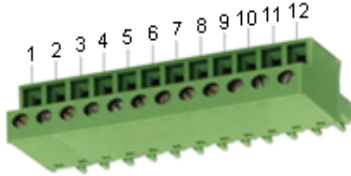
CMD"MACROSTATION2,MI198=$408807" // Point to control word of Card 1 at address $8800
CMD"MACROSTATION2,MI199=7"       // Write to control word
CALL Timer(0.250)

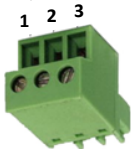
Disable PLC 1                    // Disable this PLC
Close
```

CONNECTOR PINOUTS AND WIRING

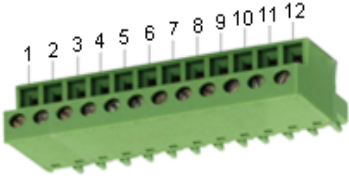
Terminal Block Connectors

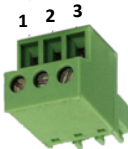
Top: Inputs

TB1 / TB2 Top (Inputs)		
Pin #	Function	Description
1	Input	Input #1 / 13
2	Input	Input #2 / 14
3	Input	Input #3 / 15
4	Input	Input #4 / 16
5	Input	Input #5 / 17
6	Input	Input #6 / 18
7	Input	Input #7 / 19
8	Input	Input #8 / 20
9	Input	Input #9 / 21
10	Input	Input #10 / 22
11	Input	Input #11 / 23
12	Input	Input #12 / 24

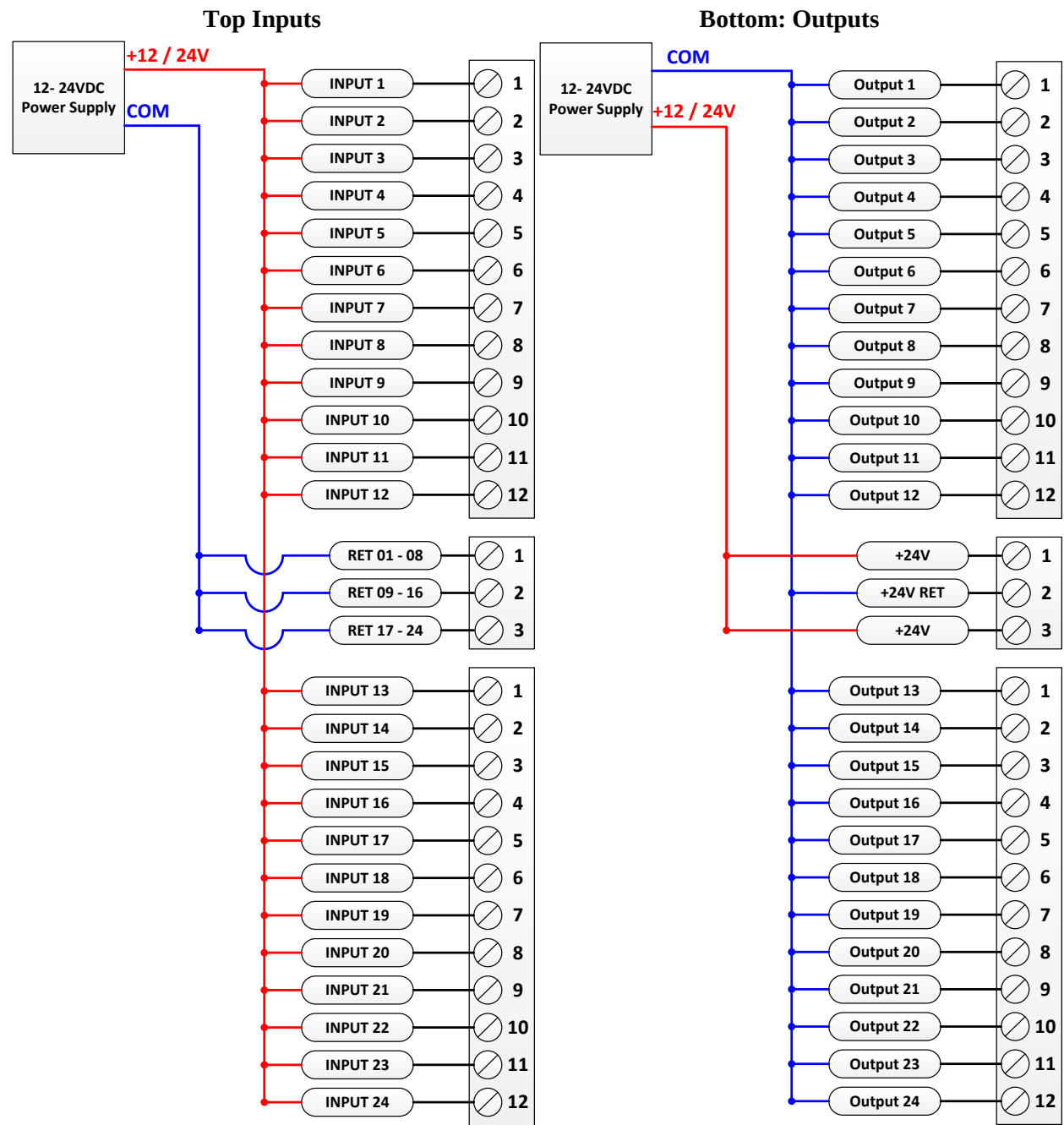
TB3 Top (Inputs Reference / Return)		
Pin #	Function / Description	
1	Inputs Return 01 – 08	
2	Inputs Return 09 – 16	
3	Inputs Return 17 – 24	

Bottom: Outputs

TB1 / TB2 Bottom (Outputs)		
Pin #	Function	Description
1	Output	Output #1 / 13
2	Output	Output #2 / 14
3	Output	Output #3 / 15
4	Output	Output #4 / 16
5	Output	Output #5 / 17
6	Output	Output #6 / 18
7	Output	Output #7 / 19
8	Output	Output #8 / 20
9	Output	Output #9 / 21
10	Output	Output #10 / 22
11	Output	Output #11 / 23
12	Output	Output #12 / 24

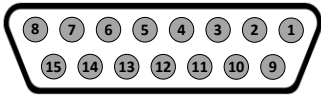
TB3 Bottom (Outputs Power)		
Pin #	Function / Description	
1	+ 24V	
2	+24V RETURN	
3	+ 24V	

TB Wiring Diagram

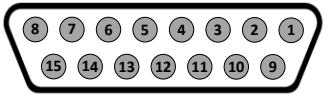


D-Sub Connectors

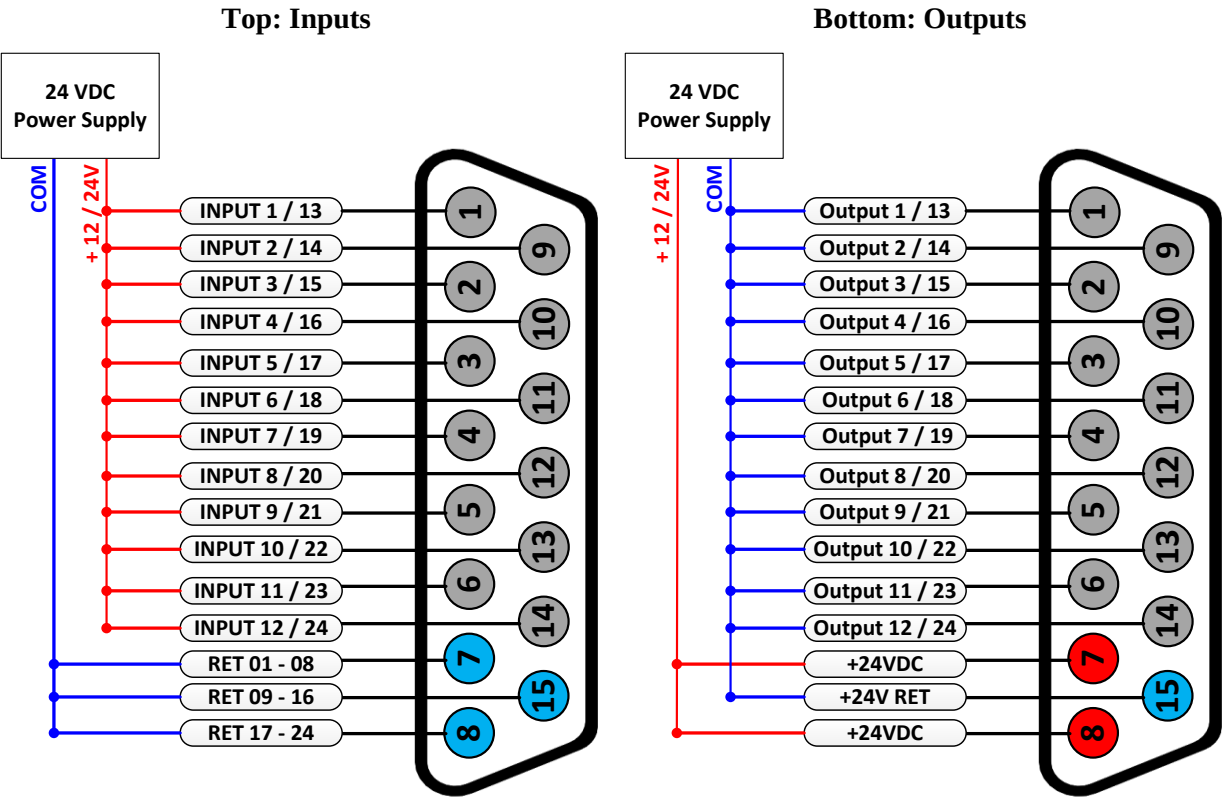
Top: Inputs

J1 / J2 Top: D-sub DA-15F Mating: D-sub DA-15M			
Pin #	Function	J1	J2
1	Input	Input #1	Input #13
2	Input	Input #3	Input #25
3	Input	Input #5	Input #17
4	Input	Input #7	Input #19
5	Input	Input #9	Input #21
6	Input	Input #11	Input #23
7	Return	Inputs Return 1–8	
8	Return	Inputs Return 17–24	
9	Input	Input #2	Input #14
10	Input	Input #4	Input #16
11	Input	Input #6	Input #18
12	Input	Input #8	Input #20
13	Input	Input #10	Input #22
14	Input	Input #12	Input #24
15	Return	Inputs Return 9–16	

Bottom: Outputs

J1 / J2 Top: D-sub DA-15F Mating: D-sub DA-15M			
Pin #	Function	J1	J2
1	Output	Output #1	Output #13
2	Output	Output #3	Output #15
3	Output	Output #5	Output #17
4	Output	Output #7	Output #19
5	Output	Output #9	Output #21
6	Output	Output #11	Output #23
7	Input	+ 24V	
8	Input	+ 24V	
9	Output	Output #2	Output #14
10	Output	Output #4	Output #16
11	Output	Output #6	Output #18
12	Output	Output #8	Output #20
13	Output	Output #10	Output #22
14	Output	Output #12	Output #24
15	Return	+24V RETURN	

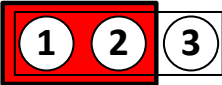
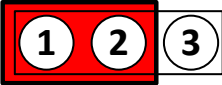
D-Sub Wiring diagram



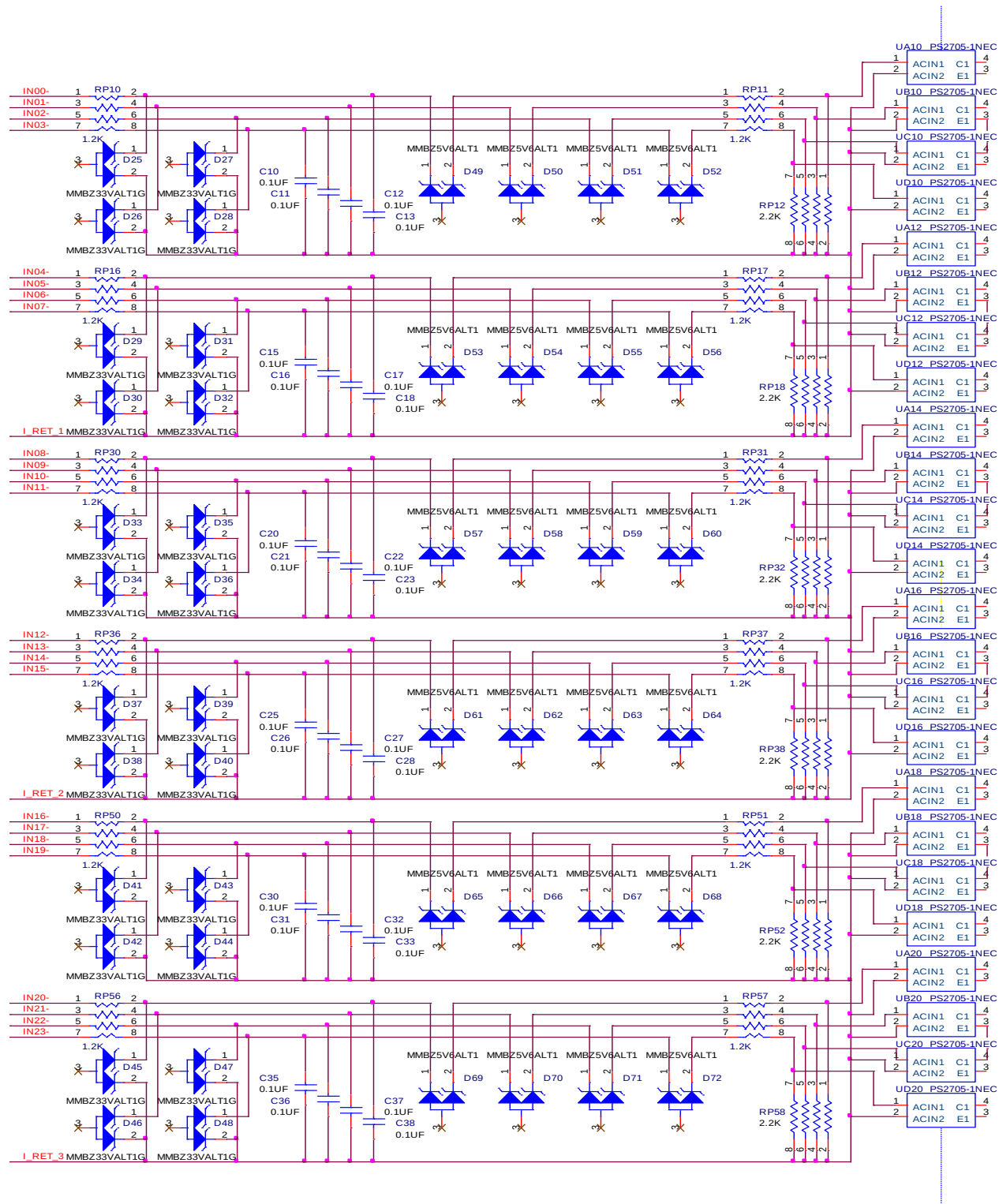
P1: UMAC Bus (UBUS) Connector

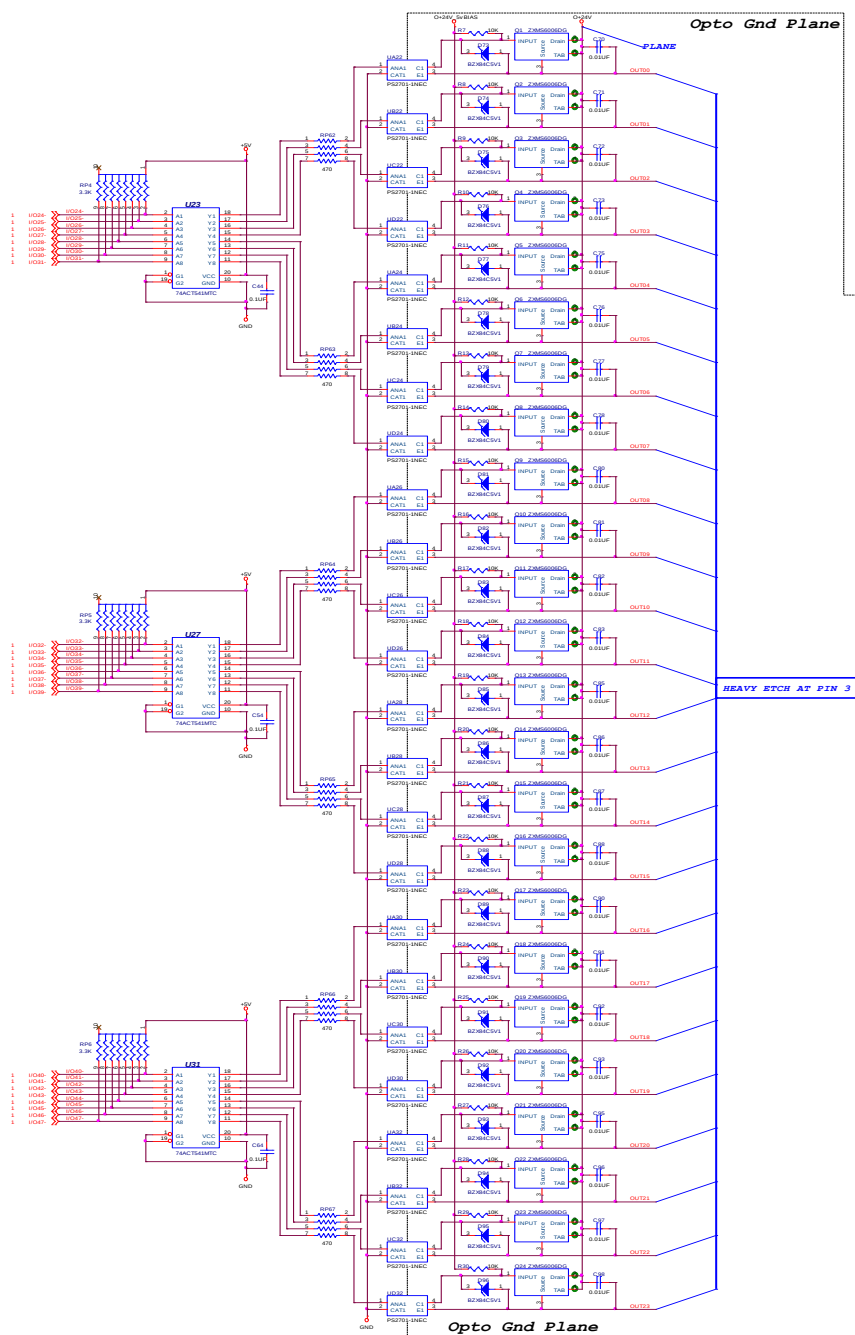
P1 UBUS (96-Pin Header)		C32 B32 A32	<table><tr><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td></tr></table>	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○				

APPENDIX A: E-POINT JUMPERS

Jumper	Configuration	Default
E1: 	➤ 1 to 2 Turbo/Power/MACRO CPUs Revision 104 or newer ➤ 2 to 3 for Legacy MACRO CPUs Rev. 103 or older	Factory Set
E2: 	➤ 1 to 2 to sample at Servo Rate ➤ 2 to 3 to sample at Phase Rate	1 – 2

APPENDIX B: SCHEMATICS





APPENDIX C: USING THE ACC-65E IN C

This section shows how the ACC-65E can be programmed and used via the C programming language in Power PMAC.

ACC-65E C Library

Include the following header file (*acc65e.h*) into the Project:

```
#include <gplib.h>
#include <RtGpShm.h> // Global Rt/Gp Shared memory pointers
//-----
// The following is a projpp created file from the User defines
//-----
#include "../../Include/pp_proj.h"

#define ON 1 // Assumes that (logic high)=true
#define OFF 0 // Assumes that (logic low)=false

void ACC65E_SetControlWord(unsigned int CardNumber);
unsigned int ACC65E_GetInputState(unsigned int CardNumber, unsigned int InputNumber);
unsigned int ACC65E_GetOutputState(unsigned int CardNumber, unsigned int OutputNumber);
void ACC65E_SetOutputState(unsigned int CardNumber, unsigned int OutputNumber, unsigned int State);
```

Then put the following source code file (*acc65e.c*) into the same folder as the header above:

```
#include <gplib.h>
#include <RtGpShm.h> // Global Rt/Gp Shared memory pointers
//-----
// The following is a projpp created file from the User defines
//-----
#include "../../Include/pp_proj.h"
#include "acc65e.h"

void ACC65E_SetControlWord(unsigned int CardNumber)
{
    // CardNumber: Power PMAC I/O Card Index of this card (0-15)
    volatile unsigned int *ioptr;
    ioptr = piom + pshm->OffsetCardIO[CardNumber]/4 + 7;
    *ioptr = 7 << 8; // Shift 7 up 8 and write the value
    return;
}

unsigned int ACC65E_GetInputState(unsigned int CardNumber, unsigned int InputNumber)
{
    // CardNumber: Power PMAC I/O Card Index of this card (0-15)
    // InputNumber: Input Pin Number (1-24)
    volatile unsigned int *ioptr;
    InputNumber--;
    ioptr = piom + pshm->OffsetCardIO[CardNumber]/4;
    ioptr += InputNumber/8;
    // Shift all bits above the desired bit out
    // Then shift down to bit 0 and return
    return (unsigned int)((*ioptr << (31 - (InputNumber%8 + 8))) >> 31);
}

unsigned int ACC65E_GetOutputState(unsigned int CardNumber, unsigned int OutputNumber)
{
    // CardNumber: Power PMAC I/O Card Index of this card (0-15)
    // OutputNumber: Input Pin Number (1-24)
    volatile unsigned int *ioptr;
    OutputNumber--;
    ioptr = piom + pshm->OffsetCardIO[CardNumber]/4;
    ioptr += OutputNumber/8 + 3;
    // Shift all bits above the desired bit out
    // Then shift down to bit 0 and return
    return (unsigned int)((*ioptr << (31 - (OutputNumber%8 + 8))) >> 31);
}
```

```

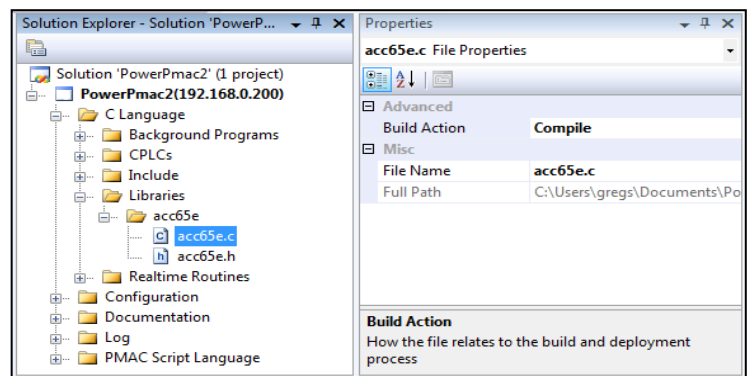
void ACC65E_SetOutputState(unsigned int CardNumber, unsigned int OutputNumber, unsigned int
State)
{
// CardNumber: Power PMAC I/O Card Index of this card (0-15)
// OutputNumber: Input Pin Number (1-24)
// State: 1=ON, 0=OFF
    volatile unsigned int *ioptr;
    unsigned int HighBitInCorrectLocation;
    OutputNumber--;
    HighBitInCorrectLocation = 1 << ((OutputNumber % 8) + 8);
    ioptr = piom + pshm->OffsetCardIO[CardNumber]/4;
    ioptr += OutputNumber/8 + 3;
    if(State == 1) // If the user wants the pin to be ON (high true)
    // Logical OR with the bit the user desires to activate
        *ioptr |= HighBitInCorrectLocation;
    else
    // Logical AND the register with a 0 in the desired location to bring the pin's state low
    // right shift to push out garbage in lowest 8 bits, then shift back up 8 bits to have
    // data in the proper location
        *ioptr &= (((~0)^HighBitInCorrectLocation) >> 8) << 8;
    return;
}

```

Locations for the C Files

Background C Programs

To use the above ACC-65E code in a Background C Program, make a new folder called “acc65e” in the C Language→Libraries folder under the Solution Explorer in the Power PMAC IDE. Then, put the code into this folder and make sure that the Build Action for acc65e.c under Properties is set to “Compile”. The Solution Explorer should then look something like the screenshot to the right.

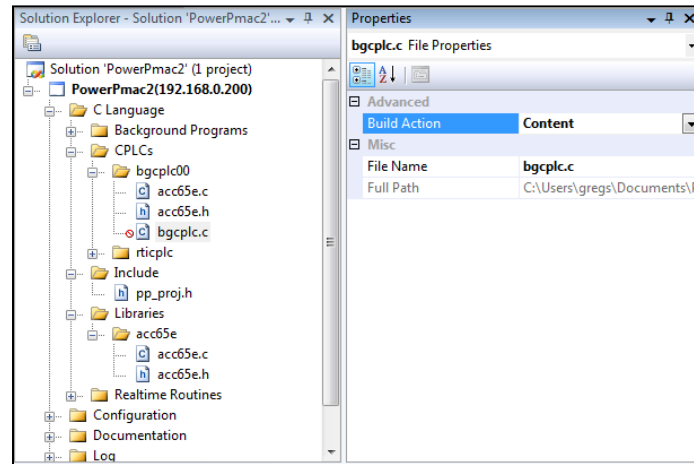


Then, in the Background Program, include the header file with the following preprocessor directive:

```
#include "../../Libraries/acc65e/acc65e.h"
```

Background CPLCs (BGCPLCs) and Real-Time CPLCs (RTICPLCs)

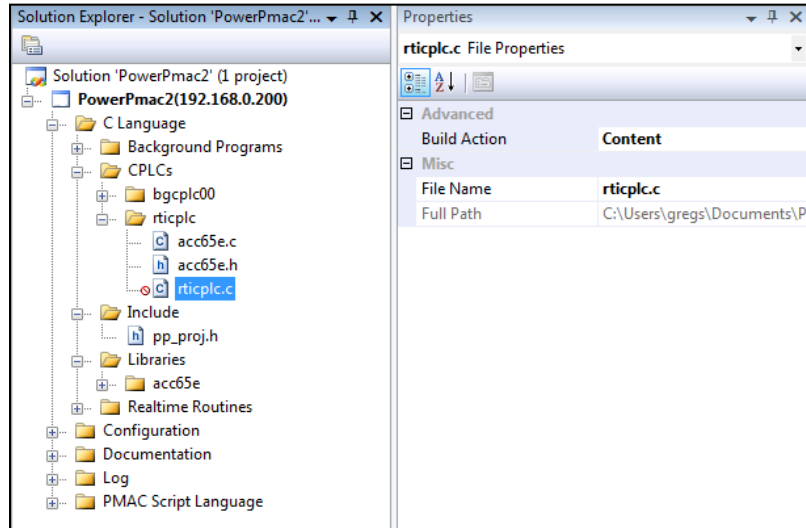
To use the above code in a Background CPLC (BGCPLC), put *acc65e.h* and *acc65e.c* into the same folder as the BGCPLC. For example, when using BGCPLC00, Solution Explorer might look like the following:



Then, include the header file with the following preprocessor directive:

```
#include "acc65e.h"
```

The same is true for RTICPLCs:



Function Descriptions

ACC65E_SetControlWord()

Description

This function sets the control word of ACC-65E.

Input Arguments

CardNumber: The Power PMAC I/O Card Index of this card, given in the table in the Addressing ACC-65E section of this manual, corresponding to this card's I/O base address offset. Range: 0–15.

Return Value

None; returns void.

ACC65E_GetInputState()

Description

This function returns the state of the input pin specified.

Input Arguments

CardNumber: The Power PMAC I/O Card Index of this card, given in the table in the Addressing ACC-65E section of this manual, corresponding to this card's I/O base address offset. Range: 0–15.

InputNumber: The pin number of the input pin whose state the user desires to read. Inputs on ACC-65E are numbered 1–24, which is the range for this argument.

Return Value

Returns the state of the input pin:

0 = False/Low/Off

1 = True/High/On

ACC65E_GetOutputState()

Description

This function returns the state of the output pin specified.

Input Arguments

CardNumber: The Power PMAC I/O Card Index of this card, given in the table in the Addressing ACC-65E section of this manual, corresponding to this card's I/O base address offset. Range: 0–15.

OutputNumber: The pin number of the output pin whose state the user desires to read. Outputs on ACC-65E are numbered 1–24, which is the range for this argument.

Return Value

Returns the state of the input pin:

0 = False/Low/Off

1 = True/High/On

ACC65E_SetOutputState()

Description

This function sets the state of the output pin specified.

Input Arguments

CardNumber: The Power PMAC I/O Card Index of this card, given in the table in the Addressing ACC-65E section of this manual, corresponding to this card's I/O base address offset. Range: 0–15.

OutputNumber: The pin number of the output pin whose state the user desires to set. Outputs on ACC-65E are numbered 1–24, which is the range for this argument.

Return Value

None; returns void.

Example

Getting Inputs and Setting Outputs in a BGCPLC

For a single ACC-65E at I/O base offset \$B00000, this BGCPLC reads inputs 1–24, storing the values in P100–P123, respectively, and writes to outputs 1–24 using values stored in P200–P223, respectively. The states of output pins 1-24 are read and then stored in P300–P323, respectively, for sake of comparison or demonstration purposes.

```
#include <gplib.h>
#include <stdio.h>
#include <dlfcn.h>

#include "../Include/pp_proj.h"
#include "acc65e.h"

#define ACC65E_CardNumber    1 // for base offset of $B00000

void user_plcc()
{
    unsigned int ChannelNumber, ArrayIndex;
    for(ChannelNumber = 1; ChannelNumber < 25; ChannelNumber++)
    { // Cycle through all 24 channels
        ArrayIndex = ChannelNumber - 1; // Convert channel number to array index
        // Get the states of input pins 1-24 and store the states in P100-P123
        pshm->P[100 +
ArrayIndex]=(double)ACC65E_GetInputState(ACC65E_CardNumber,ChannelNumber);
        // Set the states of output pins 1-24 to the values stored in P200-P223
        ACC65E_SetOutputState(ACC65E_CardNumber,ChannelNumber, (unsigned int)pshm->P[200 +
ArrayIndex]);
        // Get the states of output pins 1-24 and store the states in P300-P323
        pshm->P[300 +
ArrayIndex]=(double)ACC65E_GetOutputState(ACC65E_CardNumber,ChannelNumber);
    }
    return;
}
```