# USER MANUAL

# Accessory 28E

16-Bit Analog-To-Digital Converter Board

3Ax-603404-xUxx

February 14, 2015

DELTA TAU
Data Systems, Inc.

*NEW IDEAS IN MOTION …*

*Single Source Machine Control* *Power // Flexibility // Ease of Use*

***21314 Lassen Street Chatsworth, CA 91311 // Tel. (818) 998-2095 Fax. (818) 998-7807 // www.deltatau.com***

## Copyright Information



To report errors or inconsistencies, call or email:

**Delta Tau Data Systems, Inc. Technical Support**
Phone: (818) 717-5656
Fax: (818) 998-7807
Email: support@deltatau.com
Website: http://www.deltatau.com

## 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.

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. 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.

**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 user's 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 | ADDED ORDERING OPTIONS; NEW LAYOUTS AND SCHEMATIC | 11/14/07 | CP | SM |
| 3 | ADDED UL SEAL TO MANUAL COVER; UPDATED AGENCY APPROVAL/SAFETY SECTION | 09/30/09 | CP | SF |
| 4 | CORRECTED PORT I/O REGISTER INFO | 03/24/10 | CP | MY |
| 5 | ADDED POWER PMAC SECTION; REVISED MACRO SECTION; REORGANIZED MANUAL | 02/14/15 | DCDP | RN |

## Table of Contents

# INTRODUCTION

Delta Tau's Accessory 28E (ACC-28E) is a 2 or 4 (Option 1) channel analog to digital (A-D) converter interface board designed to provide a means for precision voltage measurement as an input to the Turbo UMAC Turbo, Power UMAC, or UMAC MACRO systems. This accessory uses 2 or 4 (Option 1) 16-bit analog to digital converters to provide voltage measurements accurate to ±2 bits.

Below are two images of the card from side views:

# SPECIFICATIONS

## Environmental Specifications

| Description | Specification | Notes |
|---|---|---|
| Operating Temperature | 0 °C to 45 °C | |
| Storage Temperature | -25 °C to 70 °C | |
| Humidity | 10% to 95% | Non-Condensing |

## Electrical Specifications

### Power Requirements

Whether providing the ACC-28E with power from the 3U backplane bus or externally (standalone mode) through TB1, the power requirements (± 10%) are:

+ 5 VDC @ 150 mA
+15 VDC @ 20 mA
- 15 VDC @ 20 mA

### Agency Approval and Safety

| Item | Description |
|---|---|
| CE Mark | Full Compliance |
| EMC | EN55011 Class A Group 1<br>EN61000-3-2 Class A<br>EN61000-3-3<br>EN61000-4-2<br>EN61000-4-3<br>EN61000-4-4<br>EN61000-4-5<br>EN61000-4-6<br>EN61000-4-11 |
| Safety | EN 61010-1 |
| UL | UL 61010-1 File E314517 |
| cUL | CAN/CSA C22.2 No. 1010.1-92 File E314517 |
| Flammability Class | UL 94V-0 |

### Input Offset Nulling

Input nulling is performed at Delta Tau with the A-D inputs shorted together using Bipolar conversion. If the user's equipment has output offsets, it is possible to adjust the VR1 thru VR4 to zero the inputs for ADC 1 thru ADC 4, respectively. Locations of VR1 to VR4 adjustment pots are shown in Layout and Pin-outs section. The input voltage adjustment swing is limited to approximately 60 mV.

When selected for Bipolar conversion, a 0 VDC input should read a number around 32,768 on the A-D input. When selected for Unipolar conversion, the input should be adjusted to 0.

## Physical Specifications

| Description | Specification | Notes |
|---|---|---|
| Dimensions | Length: 16.256 cm (6.4 in.)<br>Height: 10 cm (3.94 in.)<br>Width: 2.03 cm (0.8 in.) | |
| Weight w/o Option 1A | 194 g | Front , Top, and Bottom plates included |
| Terminal Block Connectors | FRONT-MC1,5/10-ST3,81 | Terminal Blocks from Phoenix Contact. UL-94V0 |
| DB Option Connectors | DB9 Female | UL-94V0 |
| The width is the width of the front plate.  The length and height are the dimensions of the PCB. | | |

# ADDRESSING ACC-28E

Dip switch (SW1) specifies the base address of the ACC-28E in a 3U TURBO / POWER UMAC, or MACRO Station rack.

## Turbo/Power UMAC, MACRO Station Dip Switch Settings

| **Chip Select** | **Base Address** | | | | **SW1 Positions** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **TURBO** | **MACRO** | **POWER Offset** | **POWER Index (*i*)** | **6** | **5** | **4** | **3** | **2** | **1** |
| 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
- The maximum addressable number of ACC-28Es (or similar type accessories) in a single rack is 16

## 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 Addresses | Maximum Number of cards in 1 rack |
|---|---|---|---|---|
| ACC-9E | A | General I/O | 4 | 12 |
| ACC-10E | A | General I/O | 4 | |
| ACC-11E | A | General I/O | 4 | |
| ACC-12E | A | General I/O | 4 | |
| ACC-14E | B | General I/O | 16 | 16 |
| ACC-28E | B | Analog I/O | 16 | |
| ACC-36E | B | Analog I/O | 16 | |
| ACC-53E | B | Feedback | 16 | |
| ACC-57E | B | Feedback | 16 | |
| ACC-58E | B | Feedback | 16 | |
| ACC-59E | B | Analog I/O | 12 | 12 |
| ACC-65E | B | General I/O | 16 | 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 the attention to the following set of rules (next page):

## Populating Rack with Type A Cards Only (no conflicts)

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

***Note***

Type A cards can have up to 4 different base addresses. And knowing that each card can be set up (jumper settings) to use the lower, middle or higher byte of a specific base address, it is then possible to populate a single rack with a maximum of 12 Type A accessory cards.

## Populating Rack with Type B Cards Only (no conflicts)

In this mode, 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 next couple of rules does not apply, and can be disregarded.
- Type A cards cannot share the same Chip Select as Type B Feedback cards.
- Type A cards can share the same Chip Select as Type B general I/O cards. However, in this mode, Type B cards naturally use the lower byte (default), and Type A cards must be set up (by means of jumper settings) to the middle/high byte of the selected base address.

## Type A Cards and Type B Feedback Cards

Type A cards cannot share the same Chip Select as Type B Feedback cards.

## Type A Cards and Type B General I/O Cards

Type A cards can share the same Chip Select as Type B general I/O cards; however, in this mode, Type B cards naturally use the lower byte (default), and Type A cards must be set up (jumper settings) to the middle/high byte of the selected base address.

## Type A Cards and Type B Analog Cards

Type A cards can share the same Chip Select as Type B analog I/O cards; however, in this mode, Type B cards naturally use the middle/high bytes (default), and Type A cards must be set up (jumper settings) to the lower byte of the selected base address.

# USING ACC-28E WITH TURBO UMAC

To ensure that ACC-28E or any other analog I/O cards work properly in a UMAC rack, it is crucial to have a clock source. The clock in a UMAC rack usually comes from ACC-24E2, ACC-24E2A, ACC-5E, ACC-51E, or any other card with a gate array on it. Before trying to use ACC-28E, ensure that the rack has a clock source.

## Setting Up Analog Inputs (ADCs)

Steps to set up ACC-28E with Turbo UMAC are as follows:

1. **Select the base address:**

   With SW1 dip switch setting, the base address offset can be determined for the card. Channel #1's data resides at this address, and the three sequential memory addresses (offset by 1 address per channel) are for channel #2, #3, and #4.

   Since ACC-28E is a 16-bit ADC card, only the highest 16 bits of these addresses are used.

2. **Select the appropriate conversion jumper:**

   Jumpers E1 thru E4 determine whether the conversion method of Channels #1 to #4 is unipolar or bipolar.

3. **M-Variable:**

   Use M-Variable as pointers to read ADC results.

### Reading the ADCs

#### ADC Read Example 1 (Unipolar)

Configuring ACC-28E card base address as Y:$78C00 with unipolar inputs.

1. **Base address select:** SW1 dip switches have pins 1 through 6 set to ON.
2. **Conversion jumper:** Jumpers E1 through E4 are in position 1–2.
3. **M-Variable:** Use M-Variables as pointers to read the ADC results.

```
M7101->Y:$78C00,8,16,U          // M7101 is assigned to read ADC 1
M7102->Y:$78C01,8,16,U          // M7102 is assigned to read ADC 2
M7103->Y:$78C02,8,16,U          // M7103 is assigned to read ADC 3
M7104->Y:$78C03,8,16,U          // M7104 is assigned to read ADC 4
```

The user can choose other M-Variables as pointers to read the ADC results if desired.

#### ADC Read Example 2 (Bipolar)

Configuring ACC-28E card base address as Y:$78C00 with bipolar inputs.

1. **Base address select:** SW1 dip switches have pins 1 through 6 set to ON.
2. **Conversion jumper:** Jumpers E1 through E4 are in position 2–3.
3. **M-Variable:** Use M-Variable as pointers to read ADC results.

```
M7101->Y:$78C00,8,16,U          // M7101 is assigned to read ADC 1
M7102->Y:$78C01,8,16,U          // M7102 is assigned to read ADC 2
M7103->Y:$78C02,8,16,U          // M7103 is assigned to read ADC 3
M7104->Y:$78C03,8,16,U          // M7104 is assigned to read ADC 4
```

The user can choose other M-Variables as pointers to read the ADC results if desired.

*Note*

Although in Example 2, E1 through E4's jumper settings are bipolar, the M-Variable definitions need to be **Unsigned** since the ADC result values are all in positive range. Refer to **Testing the Analog Inputs** section below to see how the positive ADC results relate to negative voltages.

## Testing Analog Inputs

The Analog Inputs can be brought into the ACC-28E as single ended (ADC+ & Ground) or differential (ADC+ & ADC-) signals.

***Note***

In single-ended mode, the ADC- for the channel (e.g. ADC1- for channel #1) should be tied to analog ground for full resolution and proper operation; do not leave the pin floating.

Reading the input signals in software counts using the M-Variables defined as above should show the following:

| ADC Input | Single-Ended [V] ADC+ ←→ AGND | Differential [V] ADC+ ←→ ADC- | Software Counts |
|---|---|---|---|
| Unipolar Mode | 0 | 0 | 0 |
| | 5 | 5 | 32768 |
| | 10 | 10 | 65535 |
| Bipolar Mode | -10 | -10 | 0 |
| | 0 | 0 | 32768 |
| | 10 | 10 | 65535 |

## Using Analog Input for Servo Feedback

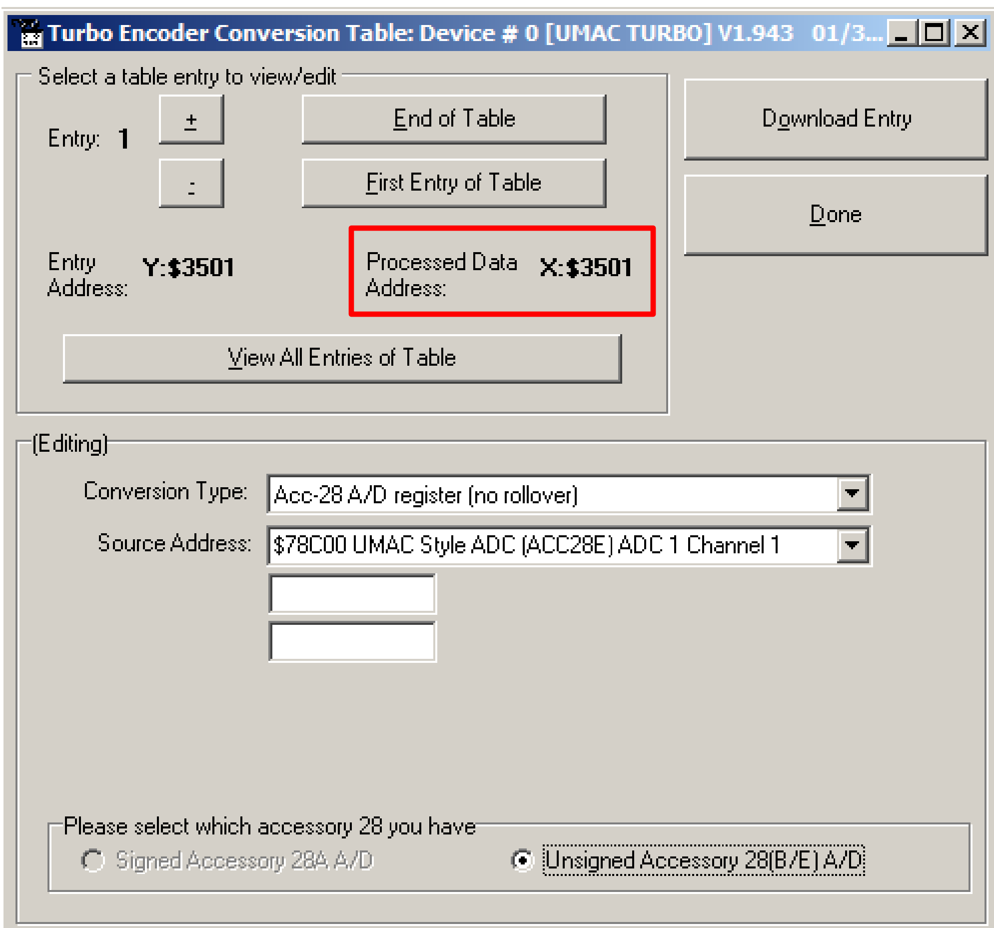
The ACC-28E analog inputs can be used as a feedback device for a servo motor

> ***Note***
>
> Refer to Delta Tau's released application notes or Turbo User Manual for cascaded-loop control (i.e. force, height control around position loop).

**Position Servo Feedback Example:**
This example shows how to use ADC Channel #1 for Motor #1's position feedback. The Encoder Conversion method digit is $1.

The analog input should be brought into the Encoder Conversion Table as the 1st line when being used as the position feedback for Motor #1 in this example. To access the Encoder Conversion Table Configurator, from within PeWin32Pro2, click Configure→Encoder Conversion Table. Then, adjust the following settings in the window that appears:

- Conversion Type: ACC-28 A/D register (no rollover)
- Source Address: Determined by the SW1 setting and the channel desired. In this case, select $78C000 as the base address and choose ADC 1 Channel 1.
- Signed or Unsigned: This is automatic selected according to the card type (i.e. ACC-28E's data is unsigned).



The equivalent code in Turbo PMAC I8000 Encoder Conversion Table parameters is as follows:

```
I8000=$1F8C00                    // $180000 + $078C00
                                 // $180000 defines ACC-28 A/D register, unsigned data
                                 // $078C00 is register address
```

The position and velocity pointers should then be set to the processed data address (i.e. $3501) as follows:

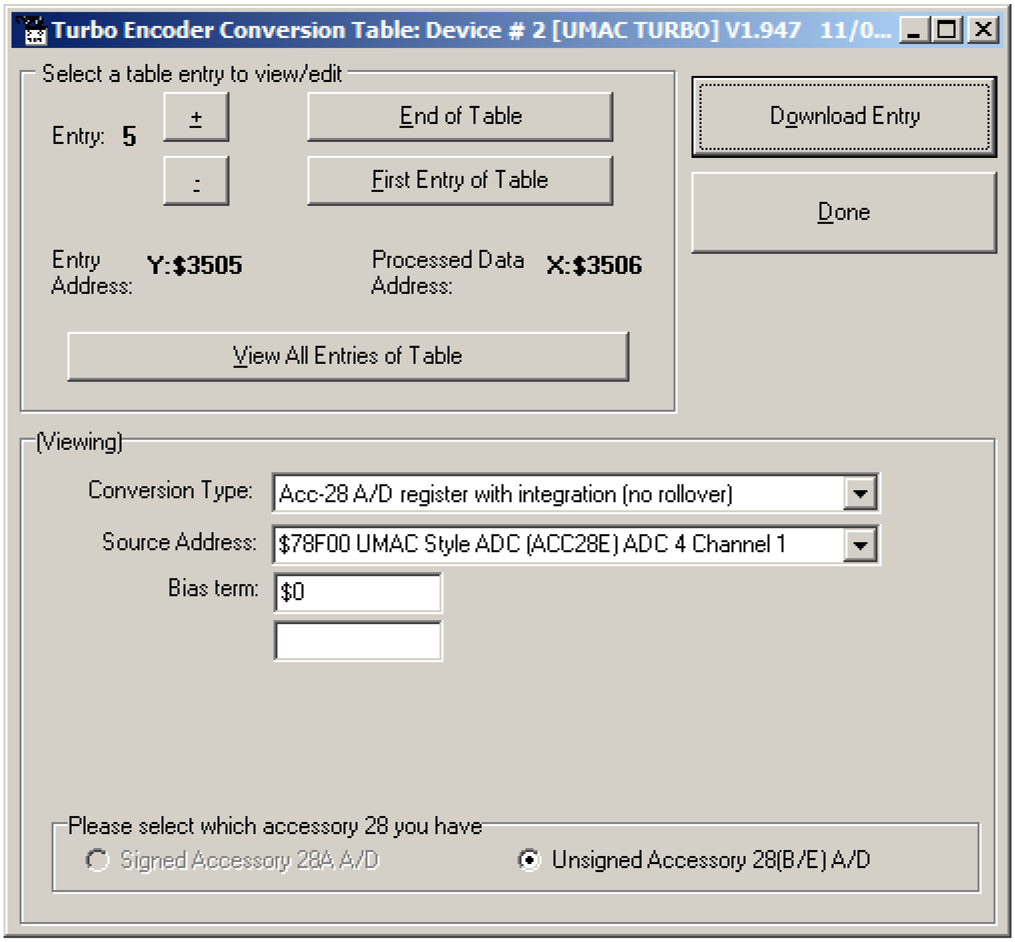
```
I103=$3501                       // Motor #1 position loop feedback address
I104=$3501                       // Motor #1 velocity loop feedback address
```

If you are using a different Encoder Conversion Table entry number, you can examine on the Configurator where the “Processed Data Address” is listed (as shown surrounded by red in the above screenshot).

**Velocity Servo Feedback Example:**
This example demonstrates how to set up ADC Channel #1 for Motor #5's velocity feedback. The Encoder Conversion method digit is $5.

The analog input should be brought into the Encoder Conversion Table as the 5th line for Motor #5's velocity feedback in this example. To access the Encoder Conversion Table Configurator, from within PeWin32Pro2, click Configure→Encoder Conversion Table. Then, adjust the following settings in the window that appears:

- Conversion Type: ACC-28 A/D register with integration (no rollover)
- Source Address: Determined by SW1 setting and the desired channel. In this case, select $78F00 as the base address, and select ADC 4 Channel 1.
- Bias term: A 24-bit number subtracted from the source A/D data (whose LSB is in bit 8) before numerical integration.
- Signed or Unsigned: This is automatically selected according to card type (i.e. ACC-28E's data is unsigned).




The equivalent code in Turbo PMAC I8000 Encoder Conversion Table parameters is as follows:

```
I8004=$5F8F00                   // $580000 + $078F00
I8005=$0                        // Bias term is 0; $580000 defines ACC-28 A/D register
                                // Unsigned data, $078F00 is register address
```

The position and velocity pointers should then be set to the processed data address (i.e. $3506) as follows:

```
I503=$3506                      // Motor #5 position loop feedback address
I504=$3506                      // Motor #5 velocity loop feedback address
```

## Using Analog Input for Power-On Position

ACC-28E can be used to establish an absolute position reference for power-on position. The following two I-Variables need to be set for this feature:

- **Ixx10**: Motor xx Power-On Position Address
- **Ixx95**: Motor xx Power-On Servo Position Format

**Ixx10** specifies what register is read for absolute position data. **Ixx95** specifies how the data in the register is interpreted. If ACC-28E is used to give absolute power-on position, **Ixx10** needs to be set to the address of the ADC channel whose input value will be used as the power-on absolute position. **Ixx95** needs to be $310000 since ACC28E only uses unsigned data.

**Power-On Absolute Position Example:**
This example demonstrates how to set up Motor #5's power-on absolute position using ACC-28E's ADC Channel #1. In this example, SW1's position 2 is OFF and the E1 jumper is set pins 1-2 for unipolar mode.

The software setup is as follows:

```
I510=$78E00                      // ADC 1 register of ACC-28E w/ SW1 position 2 OFF
I595=$310000                     // Unsigned data from register specified by Ixx10
```

With any of the following three commands, the power-on absolute position will be read after the setup:

- **$$$**: Power-on/reset
- **#n$***: Need to specify motor number **n**, such as **#5$***. Forces absolute position read for the specified motor.
- **&n$$***: Need to specify the coordinate system **n** containing the motor in interest, such as **&1$$***. Forces absolute position read for all motors in the specified coordinate system.

Depending on the mode selection (unipolar or bipolar), the absolute position reading (software counts) is as follows:

| ADC Input | Single-Ended [V] ADC+ <=> AGND | Differential [V] ADC+ <=> ADC- | Software Counts |
|---|---|---|---|
| Unipolar Mode | 0 | 0 | 0 |
| | 5 | 5 | 32768 |
| | 10 | 10 | 65535 |
| Bipolar Mode | -10 | -10 | 0 |
| | 0 | 0 | 32768 |
| | 10 | 10 | 65535 |

# USING ACC-28E WITH POWER UMAC

## Setting Up Analog Inputs (ADCs)

Power UMAC has structures to read ADC values as follows:

| Structure | Function |
|---|---|
| **ACC28E[*i*].ADCuData[*i*]** | Data of ADC channel (***j***+1), unipolar, for card index ***i*** |
| **ACC28E[*i*].ADCsData[*i*]** | Data of ADC channel (***j***+1), bipolar, for card index ***i*** |

where ***i*** ranges from 0 to 15 and is the card index given in Addressing ACC-28E section, and ***i*** is the channel index, which ranges from 0 to 3 for Channels 1 to 4, respectively.

### Manual ADC Read Using ACC-28E Structures

**Unipolar Script PLC Example**

This example demonstrates how to read ADC Channels #1 through 4 as unipolar and then convert them to voltages stored in **global** variables. SW1 setting has switches 1-6 all set to ON, which corresponds to index ***i*** = 0. Jumpers E1 through E4 are in position 1–2.

```
ptr ADC1u->ACC28E[0].ADCuData[0];            // Channel 1 ADC pointer variable
ptr ADC2u->ACC28E[0].ADCuData[1];            // Channel 2 ADC pointer variable
ptr ADC3u->ACC28E[0].ADCuData[2];            // Channel 3 ADC pointer variable
ptr ADC4u->ACC28E[0].ADCuData[3];            // Channel 4 ADC pointer variable

global Channel1_Voltage_Unipolar;
global Channel2_Voltage_Unipolar;
global Channel3_Voltage_Unipolar;
global Channel4_Voltage_Unipolar;

global UnipolarScaleFactor=10.0/65535.0; // 10 VDC per 65535 ADC counts

Open PLC 1
Channel1_Voltage_Unipolar=ADC1u*UnipolarScaleFactor;
Channel2_Voltage_Unipolar=ADC2u*UnipolarScaleFactor;
Channel3_Voltage_Unipolar=ADC3u*UnipolarScaleFactor;
Channel4_Voltage_Unipolar=ADC4u*UnipolarScaleFactor;
Close
```

## Bipolar Script PLC Example

This example demonstrates how to read ADC Channels #1 through 4 as bipolar and then convert them to voltages stored in **global** variables. SW1 setting has switch 1 set to OFF, switches 2–6 set to ON, corresponding to index ***i* = 1**. Jumpers E1 through E4 jumpers are in position 2–3.

```
ptr ADC1s->ACC28E[1].ADCsData[0];            // Channel 1 ADC pointer variable
ptr ADC2s->ACC28E[1].ADCsData[1];            // Channel 2 ADC pointer variable
ptr ADC3s->ACC28E[1].ADCsData[2];            // Channel 3 ADC pointer variable
ptr ADC4s->ACC28E[1].ADCsData[3];            // Channel 4 ADC pointer variable

global Channel1_Voltage_Bipolar;
global Channel2_Voltage_Bipolar;
global Channel3_Voltage_Bipolar;
global Channel4_Voltage_Bipolar;

global BipolarScaleFactor=10.0/32768.0; // 10 VDC per 32768 ADC counts

Open PLC 2
Channel1_Voltage_Bipolar=ADC1s*BipolarScaleFactor;
Channel2_Voltage_Bipolar=ADC2s*BipolarScaleFactor;
Channel3_Voltage_Bipolar=ADC3s*BipolarScaleFactor;
Channel4_Voltage_Bipolar=ADC4s*BipolarScaleFactor;
Close
```

## Testing Analog Inputs

The Analog Inputs can be brought into the ACC-28E as single ended (ADC+ & Ground) or differential (ADC+ & ADC-) signals.

***Note***

In single-ended mode, the ADC- for the channel (e.g. ADC1- for channel #1) should be tied to analog ground for full resolution and proper operation; do not leave the pin floating.

Reading the input signals in software counts using the predefined M-Variables should show the following:

| ADC Input | Single-Ended [V] ADC+ <=> AGND | Differential [V] ADC+ <=> ADC- | Software Counts |
|---|---|---|---|
| Unipolar Mode | 0 | 0 | 0 |
| | 5 | 5 | 32768 |
| | 10 | 10 | 65535 |
| Bipolar Mode | -10 | -10 | -32768 |
| | 0 | 0 | 0 |
| | 10 | 10 | 32768 |

***Note***

The bipolar mode software count range is different from that in Turbo PMAC because the structure **ACC28E[*i*].ADCsData[*j*]** automatically scales the result to ±32768 counts for more intuitive reading of bipolar signals.

## Using Analog Input for Servo Feedback

The ACC-28E analog inputs can be used as feedback for servo motors.

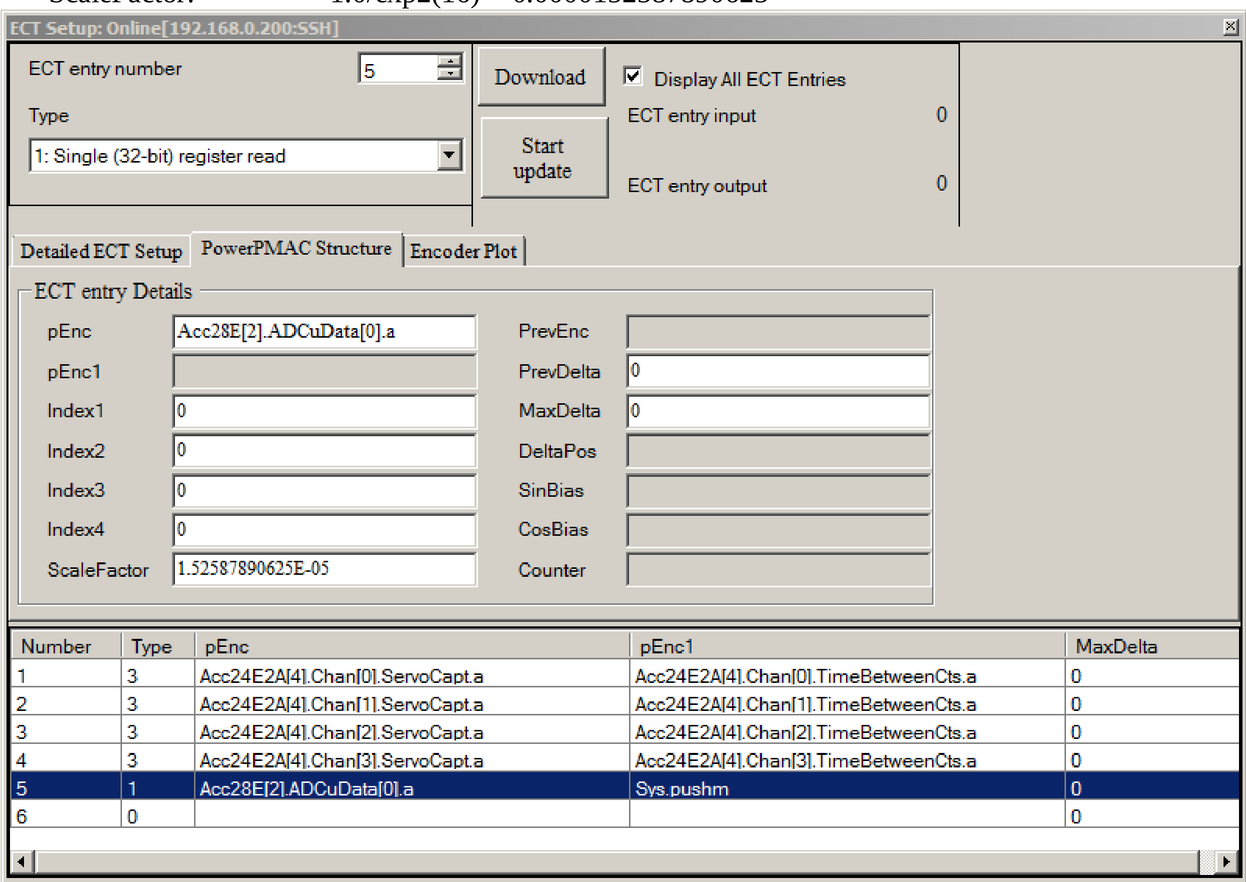
**Example:**

This example sets up Motor #5 with position and velocity feedback from ACC-28E with SW1's position 2 OFF. The card's index is ***i* = 2** and ADC Channel #1 is being used.

The analog input should be brought into the Encoder Conversion Table (ECT) with a Type 1 conversion method, which performs a single read of a 32-bit register. The 16-bit data coming from ACC-28E occupies only the upper 16 bits of this 32-bit register. Because of this, the Encoder Conversion Table Scale Factor (**EncTable[*n*].ScaleFactor**) needs to be set to $1/2^{16}$ in order to have the correct reading (i.e. to scale the position down to bit 0 of the position register).

The ECT setup window can be found in IDE by clicking on Delta Tau → Configure → Encoder Conversion Table. Change the following parameters within that window to the settings below:

- ECT entry number = 5, corresponding to the motor number
- Type = 1, Single (32-bit) register read
- pEnc: = **Acc28E[2].ADCuData[0].a**
- ScaleFactor: =1.0/exp2(16) = 0.0000152587890625



**Note**

- **EncTable[*n*].pEnc** should point to **Acc28E[*i*].ADCuData[*j*].a** for card index ***i***, ADC Channel (***j*+1)**, no matter if the signal is unipolar or bipolar.
- The Unipolar or Bipolar conversion setting is determined by jumpers E1 through E4 only. The Encoder Conversion Table will choose which setting to use automatically.

The equivalent code to set this up via Power PMAC Structures is as follows:

```
EncTable[5].type = 1;                       // Set entry type to 32-bit word read
EncTable[5].pEnc = Acc28E[2].ADCuData[0].a;  // Set encoder address to Card Index 2, Channel 1
EncTable[5].pEnc1 = Sys.pushm;              // Unused; set to Sys.pushm
EncTable[5].index1 = 0;                     // No Shift left
EncTable[5].index2 = 0;                     // No Shift right
EncTable[5].index3 = 0;                     // No limit on first derivative
EncTable[5].index4 = 0;                     // No limit on second derivative
EncTable[5].ScaleFactor = 1/exp2(16);       // Scale factor 1/(2^16)
```

The position and velocity pointers are then set to the processed data address:

```
Motor[5].pEnc = EncTable[5].a;              // Outer (position) loop source address
Motor[5].pEnc2 = EncTable[5].a;             // Inner (velocity) loop source address
```

One can then adjust **Motor[5].PosSf** and **Motor[5].Pos2Sf**, the position and velocity scale factors, respectively, if necessary.

## Using Analog Input for Power-On Position

Some analog devices are absolute along the travel range of the motor (e.g. in hydraulic piston applications). Generally, it is desirable to obtain the motor's position from the input voltage on power up or reset (**$$$**).

**Unipolar unsigned data:**

The following example code will configure Motor #5 to use ADC Channel #1's unipolar unsigned data for a power-on position read with a Scale Factor of 1:

```
Motor[5].PowerOnMode = 4;                   // Enable power on absolute position read
Motor[5].pAbsPos = Acc28E[2].ADCuData[0].a;  // Set absolute position register to ADC channel 1
Motor[5].AbsPosFormat = $00001010;          // Set format: unsigned 16 bits starting at bit 16
Motor[5].AbsPosSF = 1;                      // Set scale factor: 1
```

**Bipolar signed data:**

The following example code will configure Motor #5 to use ADC Channel #1's bipolar signed data for a power-on position read with a Scale Factor of 1:

```
Motor[5].PowerOnMode = 4;                   // Enable power on absolute position read
Motor[5].pAbsPos = Acc28E[2].ADCuData[0].a;  // Set absolute position register to ADC channel 1
Motor[5].AbsPosFormat = $00001010;          // Set format: unsigned 16 bits starting at bit 16
Motor[5].AbsPosSF = 1;                      // Set scale factor: 1
Motor[5].HomeOffset = 32768;                // Set HomeOffset to gaurantee correct data reading
```

**Note**

The ADC channels of ACC-28E always produce readings from 0 to 65535. The structure **ACC28E[*i*].ADCsData[*j*]** just offset the whole scale by 32768 to give a reading from -32768 to +32768. Since absolute power-on position is read directly from the register as an unsigned value, home offset needs to be added to have the correct signed value.

## Manual ADC Read Using Pointers (Optional)

An optional method of accessing the ADC data is through pointer (**ptr**) variables that directly map the memory address of each channel. The Base Address Offset of the card, as mentioned in the table below) is determined by the Index #***i*** as set by DIP switch SW1 as described in the Addressing ACC-28E section of this manual. Once the base address offset is determined, the channel addresses can be determined as below:

| ADC channel | Address Offset | 16-bit Location |
|---|---|---|
| 1 | Base Address Offset + $0 | [31:16] |
| 2 | Base Address Offset + $4 | [31:16] |
| 3 | Base Address Offset + $8 | [31:16] |
| 4 | Base Address Offset + $C | [31:16] |

### ADC Pointer Manual Read Example

SW1 switch setting has all 6 switches set to the ON position, which dictates a base address of $A00000, and jumpers E1 through E4 are set to position 1–2 settings for unipolar inputs.

```
ptr ACC28E_ADC1->u.io:$A00000.16.16;
ptr ACC28E_ADC2->u.io:$A00004.16.16;
ptr ACC28E_ADC3->u.io:$A00008.16.16;
ptr ACC28E_ADC4->u.io:$A0000C.16.16;
```

***Note***

- `u.io:$A00000.16.16` indicates **U**nsigned **I/O** address `$A00000`, starting at bit 16, with a 16-bit width.
- If the settings of jumpers E1 thru E4 are set for bipolar inputs, the pointer definitions still need to be **Unsigned** since the raw ADC result values are all still in the positive range. Scaling must be performed thereafter on these values.

## Voltage/ADC Value Conversion C Language Function (Optional)

The following C language function code provides an example of converting between the raw ADC values and input voltages. It assumes Card Index 0. It copies the ADC results to P5000–P5007 and the voltages in P6000–P6007.

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

// Definition(s)
#define ThisCardIndex         0       // For $A00000
#define Unipolar_Code         0       // Unipolar input signals
#define Bipolar_Code          1       // Bipolar input signals

// Prototype(s)
int ACC28E_ADC(unsigned int Card_Index, unsigned int ADC_Channel, int *ADC_Result);
int ConvertToVolts(unsigned int SoftwareCounts, unsigned int Polarity, double *Volts);

void user_plcc()
{
        unsigned int index,Channel_Number,ADC[8],tempu;
        int     ErrorCode = 0;
        double Voltages[8],tempd;
        for(index = 0; index < 4; index++)
        {
                Channel_Number = index + 1;            // Compute ADC channel number
                // Acquire ADC result
                ErrorCode = ACC28E_ADC(ThisCardIndex,Channel_Number,&tempu);
                ADC[index] = tempu;                    // Store result in array
                if(ErrorCode < 0)
                {
                        return; // error
                }
                ErrorCode = ConvertToVolts(ADC[index],Bipolar_Code,&tempd);
                Voltages[index] = tempd;
                if(ErrorCode < 0)
                {
                        return; // error
                }
                pshm->P[5000 + index] = ADC[index];   // Store result in P-Variable (optional)
                pshm->P[6000 + index] = Voltages[index];// Store voltages in P-Variable (optional)
        }
        return;
}

int ACC28E_ADC(unsigned int Card_Index, unsigned int ADC_Channel, int *ADC_Result)
{ // Obtains ADC result from a specified channel of the ACC28E with specified sign.
/* Inputs:
Card_Index: Index (k) of the card based on addressing from SW1 setting
ADC_Channel: The number of the channel that the user desires to sample (1, 2, 3, or 4)
Polarity: 0 = unipolar inputs, 1 = bipolar inputs
Outputs:
return 0 and put the ADC result of the desired channel into *ADC_Result if everything happened
correctly
return -1 if Card_Index is invalid
return -2 if ADC_Channel is invalid
*/

        unsigned int BaseOffset,Address;
        volatile unsigned int *pACC28Eu;
        if(Card_Index < 0 || Card_Index > 15)
        {
                return -1;
        }
```

```
        if(ADC_Channel < 1 || ADC_Channel > 4)
        {
                return -2;
        }

        BaseOffset = pshm->OffsetCardIO[Card_Index];         // Acquire base offset of card
        Address = piom + (BaseOffset + 4*(ADC_Channel - 1))/4;// Compute ADC channel address
        pACC28Eu = Address;                                  // Assign address to pointer
        *ADC_Result = ((unsigned int)(*pACC28Eu >> 16));     // Assign value to address
        return 0;
}

int ConvertToVolts(unsigned int SoftwareCounts, unsigned int Polarity, double *Volts)
{ // Converts the software count ADC result from ACC28E to volts in double precision
// Inputs:
// SoftwareCounts: ADC result from ACC28E in units of software counts
// Polarity: 0 = unipolar inputs, 1 = bipolar inputs

// Outputs:
// return 0 if everything went correctly, and store Volts in *Volts
// return -1 if SoftwareCounts invalid
// return -2 if Polarity invalid
        double ConversionFactor,MaxVolts = 10.0;
        unsigned int Offset,MaxCounts = 65535;

        if(SoftwareCounts < 0 || SoftwareCounts > 65535)
        {
                return -1;
        }
        if(Polarity != Unipolar_Code && Polarity != Bipolar_Code)
        {
                return -2;
        }
        if(Polarity == Unipolar_Code)
        {
                Offset = 32768;

        } else {
                Offset = 0;
        }
        ConversionFactor = (double)(MaxVolts/((double)(MaxCounts - Offset)); // Volts/Ct
        *Volts = (double)(((double)SoftwareCounts)*ConversionFactor);
        return 0;
}
```

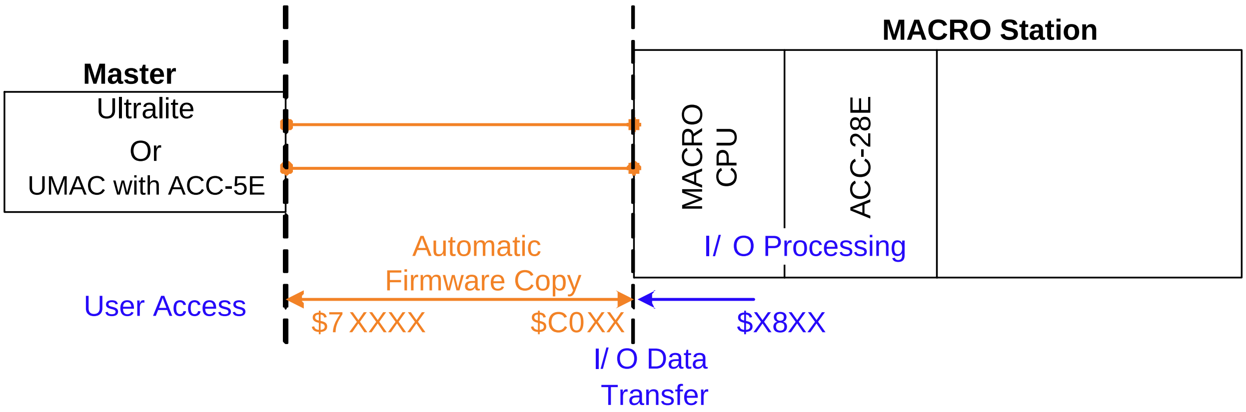
***Note***

This code could be written more efficiently in a final application, but is written here for the purpose of clearer understanding for the user.

# USING ACC-28E WITH UMAC MACRO

Setting up ACC-28E on a MACRO station requires the following steps:

- Establishing communication with the MACRO Station and enabling nodes
- Transferring Data over Nodes




The goal is to allow the user "software" access to the digital inputs and outputs brought into the MACRO Station(s) and transferred to the Master (i.e. Turbo PMAC2 Ultralite, or UMAC with ACC-5E). Note that the Master is sometimes referred to as the Ring Controller.

For all MACRO-Station I/O accessories, the information is transferred to or from the accessory I/O Gate to the MACRO-Station CPU Gate 2B. Information from the MACRO-Station Gate 2B is then read or written directly to the MACRO IC on the Master.

ACC-28E can also be used for power-on and/or servo position feedback. In this case, Servo Nodes can be used to transfer data to Master.

Once the information is at the Master, it can be used in application motion programs or PLC programs.

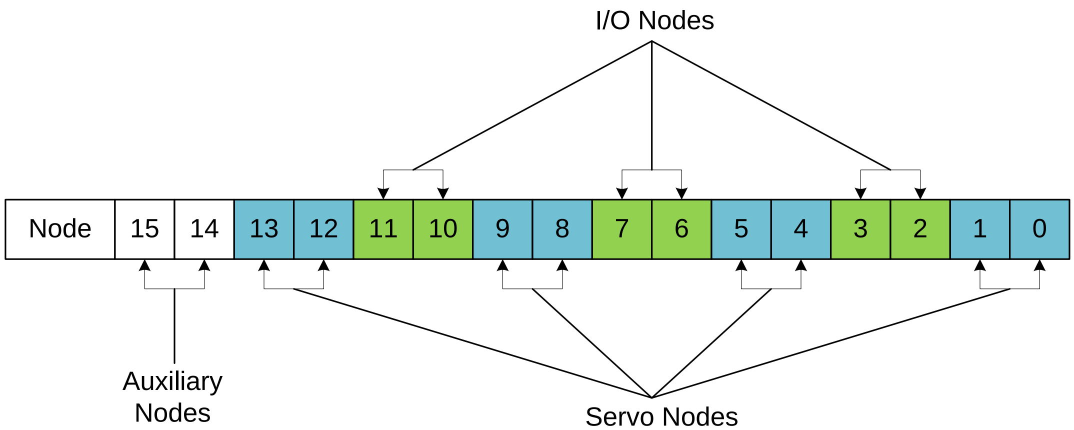
**Note**

Refer to the 16-Axis MACRO CPU manuals (SRM, USER, and HRM) f
for more information on MACRO

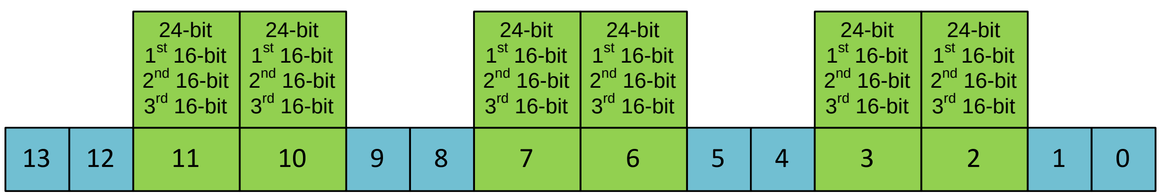
## Quick Review: Nodes and Addressing

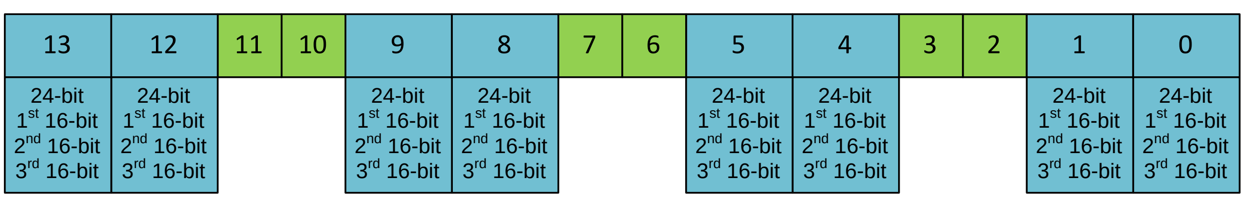
Each MACRO IC consists of 16 nodes: 2 auxiliary, 8 servo, and 6 I/O nodes.

- Auxiliary nodes are for Master/Control registers and internal firmware use.
- Servo nodes are used for motor control, carrying feedback, commands, and flag information.
- I/O nodes are by default unoccupied and are user configurable for transferring various data.




Each node consists of 4 registers; one 24-bit and three 16-bit registers for a total of 72 bits of data.
I/O nodes have X register addresses, and Servo nodes have Y register addresses.




Both Master and MACRO station CPU have an address corresponding to a node register. For example, Node 2, 24-bit register:

- Master address: X:$78420
- MACRO station address: X:$C0A0

The data is transferred between these two addresses automatically once the nodes are activated on both Master side and MACRO station side.

## MACRO Station Node Addresses

A given MACRO Station can be populated with either a MACRO8 or MACRO16 CPU:

- MACRO8 supports 1 MACRO IC (IC#0).
- MACRO16 supports 2 MACRO ICs (IC#0 and IC#1).

The following are the node register addresses on MARCO Station ICs.

### I/O Node Addresses on MACRO Station ICs

| | MACRO Station IC #0 I/O Node Registers | | | | | |
|---|---|---|---|---|---|---|
| **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 I/O Node Registers | | | | | |
|---|---|---|---|---|---|---|
| **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 |

### Servo Node Addresses on MACRO Station ICs

| | MACRO Station IC #0 Servo Node Registers | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Node** | **0** | **1** | **4** | **5** | **8** | **9** | **12** | **13** |
| **24-bit** | Y:$C0A0 | Y:$C0A4 | Y:$C0A8 | Y:$C0AC | Y:$C0B0 | Y:$C0B4 | Y:$C0B8 | Y:$C0BC |
| **16-bit** | Y:$C0A1 | Y:$C0A5 | Y:$C0A9 | Y:$C0AD | Y:$C0B1 | Y:$C0B5 | Y:$C0B9 | Y:$C0BD |
| **16-bit** | Y:$C0A2 | Y:$C0A6 | Y:$C0AA | Y:$C0AE | Y:$C0B2 | Y:$C0B6 | Y:$C0BA | Y:$C0BE |
| **16-bit** | Y:$C0A3 | Y:$C0A7 | Y:$C0AB | Y:$C0AF | Y:$C0B3 | Y:$C0B7 | Y:$C0BB | Y:$C0BF |

| | MACRO Station IC #1 Servo Node Registers | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Node** | **0** | **1** | **4** | **5** | **8** | **9** | **12** | **13** |
| **24-bit** | Y:$C0E0 | Y:$C0E4 | Y:$C0E8 | Y:$C0EC | Y:$C0F0 | Y:$C0F4 | Y:$C0F8 | Y:$C0FC |
| **16-bit** | Y:$C0E1 | Y:$C0E5 | Y:$C0E9 | Y:$C0ED | Y:$C0F1 | Y:$C0F5 | Y:$C0F9 | Y:$C0FD |
| **16-bit** | Y:$C0E2 | Y:$C0E6 | Y:$C0EA | Y:$C0EE | Y:$C0F2 | Y:$C0F6 | Y:$C0FA | Y:$C0FE |
| **16-bit** | Y:$C0E3 | Y:$C0E7 | Y:$C0EB | Y:$C0EF | Y:$C0F3 | Y:$C0F7 | Y:$C0FB | Y:$C0FF |

## Ring Master Node Addresses

Note

Non-Turbo PMAC2 Ultralite (legacy) node addresses are the same as MACRO Station IC#0 node registers.

A given Master (Turbo PMAC2 Ultralite or UMAC with ACC-5E) can be populated with up to 4 MACRO ICs (IC#0, IC#1, IC#2, and IC#3) which can be queried with global variable I4902:

| If I4902= | Populated MACRO IC #s |
|---|---|
| $0 | None |
| $1 | 0 |
| $3 | 0, 1 |
| $7 | 0, 1, 2 |
| $F | 0, 1, 2, 3 |

The following are node register addresses on Master MACRO ICs:

### I/O Node Addresses on Master MACRO ICs

| | Master MACRO IC #0 I/O Node Registers | | | | | |
|---|---|---|---|---|---|---|
| **Station I/O Node#** | **2** | **3** | **6** | **7** | **10** | **11** |
| **Master 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 |

| | Master MACRO IC #1 I/O Node Registers | | | | | |
|---|---|---|---|---|---|---|
| **Station I/O Node#** | **2** | **3** | **6** | **7** | **10** | **11** |
| **Master 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 |

| | Master MACRO IC #2 I/O Node Registers | | | | | |
|---|---|---|---|---|---|---|
| **Station I/O Node#** | **2** | **3** | **6** | **7** | **10** | **11** |
| **Master 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 |

| | Master MACRO IC #3 I/O Node Registers | | | | | |
|---|---|---|---|---|---|---|
| **Station I/O Node#** | **2** | **3** | **6** | **7** | **10** | **11** |
| **Master 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 |

## Servo Node Addresses on Master MACRO ICs

| | Master MACRO IC #0 Servo Node Registers | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Station Node#** | **0** | **1** | **4** | **5** | **8** | **9** | **12** | **13** |
| **Master Node#** | **0** | **1** | **4** | **5** | **8** | **9** | **12** | **13** |
| **24-bit** | Y:$78420 | Y:$78424 | Y:$78428 | Y:$7842C | Y:$78430 | Y:$78434 | Y:$78438 | Y:$7843C |
| **16-bit** | Y:$78421 | Y:$78425 | Y:$78429 | Y:$7842D | Y:$78431 | Y:$78435 | Y:$78439 | Y:$7843D |
| **16-bit** | Y:$78422 | Y:$78426 | Y:$7842A | Y:$7842E | Y:$78432 | Y:$78436 | Y:$7843A | Y:$7843E |
| **16-bit** | Y:$78423 | Y:$78427 | Y:$7842B | Y:$7842F | Y:$78433 | Y:$78437 | Y:$7843B | Y:$7843F |

| | Master MACRO IC #1 Servo Node Registers | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Station Node#** | **0** | **1** | **4** | **5** | **8** | **9** | **12** | **13** |
| **Master Node#** | **16** | **17** | **20** | **21** | **24** | **25** | **28** | **29** |
| **24-bit** | Y:$79420 | Y:$79424 | Y:$79428 | Y:$7942C | Y:$79430 | Y:$79434 | Y:$79438 | Y:$7943C |
| **16-bit** | Y:$79421 | Y:$79425 | Y:$79429 | Y:$7942D | Y:$79431 | Y:$79435 | Y:$79439 | Y:$7943D |
| **16-bit** | Y:$79422 | Y:$79426 | Y:$7942A | Y:$7942E | Y:$79432 | Y:$79436 | Y:$7943A | Y:$7943E |
| **16-bit** | Y:$79423 | Y:$79427 | Y:$7942B | Y:$7942F | Y:$79433 | Y:$79437 | Y:$7943B | Y:$7943F |

| | Master MACRO IC #2 Servo Node Registers | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Station Node#** | **0** | **1** | **4** | **5** | **8** | **9** | **12** | **13** |
| **Master Node#** | **32** | **33** | **36** | **37** | **40** | **41** | **44** | **45** |
| **24-bit** | Y:$7A420 | Y:$7A424 | Y:$7A428 | Y:$7A42C | Y:$7A430 | Y:$7A434 | Y:$7A438 | Y:$7A43C |
| **16-bit** | Y:$7A421 | Y:$7A425 | Y:$7A429 | Y:$7A42D | Y:$7A431 | Y:$7A435 | Y:$7A439 | Y:$7A43D |
| **16-bit** | Y:$7A422 | Y:$7A426 | Y:$7A42A | Y:$7A42E | Y:$7A432 | Y:$7A436 | Y:$7A43A | Y:$7A43E |
| **16-bit** | Y:$7A423 | Y:$7A427 | Y:$7A42B | Y:$7A42F | Y:$7A433 | Y:$7A437 | Y:$7A43B | Y:$7A43F |

| | Master MACRO IC #3 Servo Node Registers | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Station Node#** | **0** | **1** | **4** | **5** | **8** | **9** | **12** | **13** |
| **Master Node#** | **48** | **50** | **52** | **53** | **56** | **57** | **60** | **61** |
| **24-bit** | Y:$7B420 | Y:$7B424 | Y:$7B428 | Y:$7B42C | Y:$7B430 | Y:$7B434 | Y:$7B438 | Y:$7B43C |
| **16-bit** | Y:$7B421 | Y:$7B425 | Y:$7B429 | Y:$7B42D | Y:$7B431 | Y:$7B435 | Y:$7B439 | Y:$7B43D |
| **16-bit** | Y:$7B422 | Y:$7B426 | Y:$7B42A | Y:$7B42E | Y:$7B432 | Y:$7B436 | Y:$7B43A | Y:$7B43E |
| **16-bit** | Y:$7B423 | Y:$7B427 | Y:$7B42B | Y:$7B42F | Y:$7B433 | Y:$7B437 | Y:$7B43B | Y:$7B43F |

## MACRO Data Transfer

The principle of MACRO data transfer for an ACC-28E card is a two-step process.

1. On a MACRO station, ACC-28E has a base register address set by SW1. The data in this register needs to be transferred to a node register.
2. Between a MACRO station and a MACRO Ring Master, the data in a node register on the MACRO station is automatically transferred to the corresponding node register on the MACRO Ring Master. There is no need to set up this transfer, but M-Variables are needed to point to the registers on the MACRO Ring Master to retrieve the data transferred back from nodes on the MACRO station.

This procedure assumes that communication over the MACRO ring has already been established, and that the user is familiar with node activation on both the Master (also called Ring Controller) and MACRO Station. Thus, any node(s) used in the following examples have to be enabled previously.

For ACC-28E, there are 3 transfer methods as follows:

- MACRO data transfer via I/O nodes (MI20, MI21 – MI68).
- MACRO data transfer via Servo nodes (ECT and Power-On Position).
- Using MACRO I-variables MI198 and MI199 for quick hardware check.

***Note***

This section assumes that MACRO ring, I/O nodes (i.e. I6841), and ring check error settings have been configured properly.

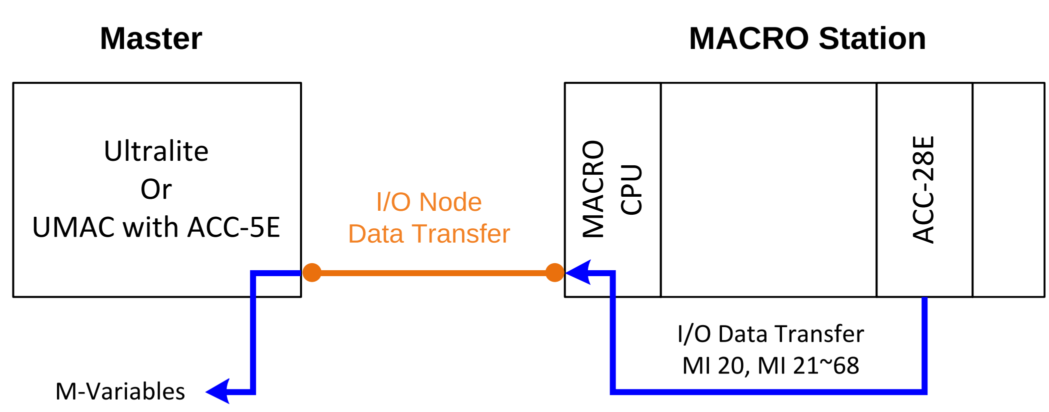
# MACRO Data Transfer via I/O Nodes

## Preparing MACRO16 for I/O Data Transfer

The following parameters should be configured properly for the I/O node transfer to work properly:

- **MS{anynode}, MI992** Max Phase frequency control
  Typically, set equal to ring controller's **I6800**
- **MS{anynode}, MI997** Phase clock frequency control
  Typically, set equal to ring controller's **I6801**
- **MS{anynode}, MI995** MACRO Ring configuration/status
  Typically, set to = $4080
- **MS{anynode}, MI996** MACRO IC#0 node activate control
- **MS{anynode}, MI975** MACRO IC#0 I/O Node enable
  **MI975** should match enabled I/O nodes in **MI996**
- **MS{anynode}, MI8** MACRO Ring Check Period
  Typically set = 8 (with default clock settings)
- **MS{anynode}, MI9** MACRO Ring error shutdown count
  Typically set = 4 (with default clock settings)
- **MS{anynode}, MI10** MACRO Sync packet shutdown count
  Typically set = 4 (with default clock settings)
- **MS{anynode},MI19** I/O node data transfer rate
  = 0, transfer disabled.
  > 0, transfer period in Phase clock cycles (typically set = 4).
- **MS{anynode}, MI1996** MACRO IC#1 node activate control (if IC#1 is used)
- **MS{anynode}, MI1975** MACRO IC#1 I/O Node enable (if IC#1 is used)
  **MI1975** should match enabled I/O nodes in MI1996

The I/O node data transfer scheme from MACRO station to Master is illustrated as following.



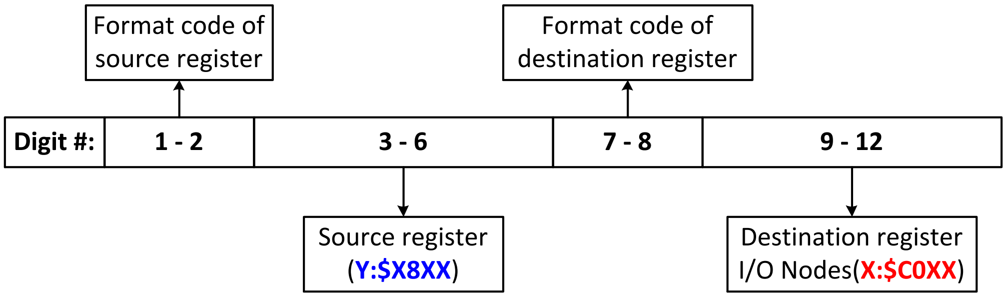

## MS{anynode},MI20: Data Transfer Enable Mask

**MI20** controls which of 48 possible data transfer operations are performed at the data transfer period set by **MI19**. **MI20** is a 48-bit value; and each bit controls whether the data transfer specified by one of the variables **MI21** through **MI68** is performed.

```
MI20=$1                       // bit 0 = 1, use transfer method MI21
MI20=$3                       // bit 0,1 = 1, use transfer methods MI21, MI22
MI20=$F                       // bit 0,1,2,3 = 1, use transfer methods MI21, MI22, MI23, MI24
```

## MS{anynode},MI21-MI68: Data Transfer Source and Destination Address

**MI21** through **MI68** specify 48 different possible data transfer methods. Each of these variables is a 48-bit word constructed in hex digits as follows:




Each method specifies the format of data transfer between source and destination registers. The data bit widths of source and destination registers needs to match. When using ACC-28E in a MACRO station, the source register is the register determined by Dip switch SW1's setting. The destination register is determined by which I/O node is used for data transfer. Since ACC-28E is a 2 or 4 channel 16-bit ADC card, the data transfer always uses 16-bit transfer methods.

The following table shows the 2-digit hex format (digit 1-2 or 7-8) and selected portions of the register that need to be used.

| Format Code | Register X or Y | Bit Width | Bit Range | Notes |
|---|---|---|---|---|
| $40 | Y | 8 | 0-7 | |
| $48 | Y | 8 | 8-15 | |
| $50 | Y | 8 | 16-23 | |
| $54 | Y | 12 | 0-11 | Lower 12-bit ADC Registers |
| $60 | Y | 12 | 12-23 | Upper 12-bit ADC Registers |
| $64 | Y | 16 | 0-15 | |
| $6C | Y | 16 | 8-23 | 16-bit MACRO Servo Node Registers |
| $78 | Y | 24 | 0-23 | 24-bit MACRO Servo Node Registers |
| $B0 | X | 8 | 0-7 | |
| $B8 | X | 8 | 8-15 | |
| $C0 | X | 8 | 16-23 | |
| $C4 | X | 12 | 0-11 | |
| $D0 | X | 12 | 12-23 | |
| $D4 | X | 16 | 0-15 | |
| $DC | X | 16 | 8-23 | 16-bit MACRO I/O Node Registers |
| $E8 | X | 24 | 0-23 | 24-bit MACRO I/O Node Registers |

For each ADC channel of ACC-28E, the 16-bit source data always resides in the upper 16 bits of that channel's register (Y:$X8XX). The data also resides in the upper 16 bits of the selected destination 16-bit register (X:$C0XX) of the selected I/O node.

*Note*

The following I/O data transfer method examples assume that MACRO communication, enabling I/O node, and other aforementioned necessary MACRO ring parameters have been configured properly on both the ring Controller and MACRO Station.

**Example:**

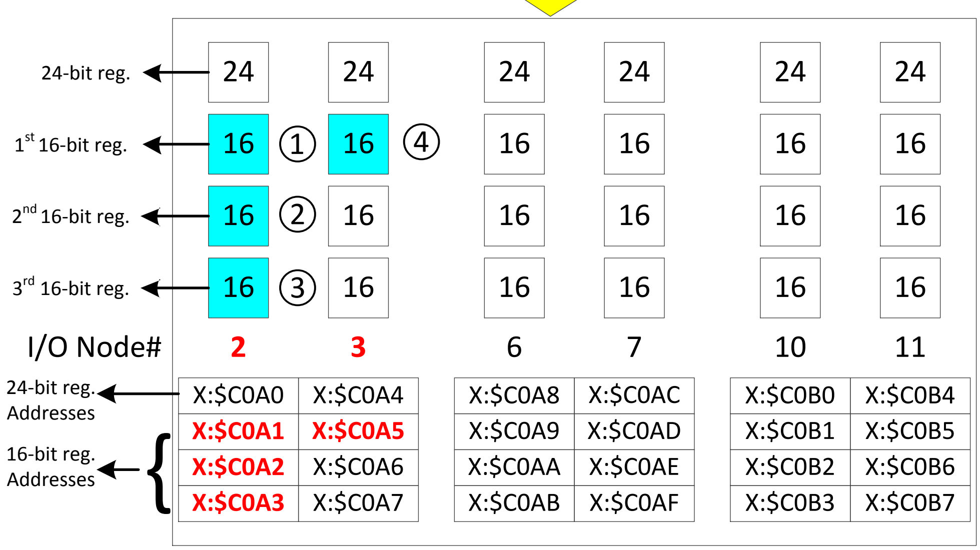
Transferring 4 ADC channels from ACC-28E in a MACRO UMAC Station rack back to the MACRO Ring Master (Turbo PMAC2 Ultralite or UMAC with ACC-5E).

- On MACRO station, transferring I/O data of one ACC-28E card with 4 ADC channels at base address **Y:$8800** to I/O nodes **2** and **3** using **4** of the 16-bit registers (**X:$C0A1-$C0A3, and X:$C0A5**).
- On MACRO Ring Master, use M-variables to read the data transferred back from MACRO station I/O nodes.

| ACC-28E (**Y:$8800**) | **4** ADC Channels (16 bits) |
|---|---|

| Base address offset | X-Register | Y-Register | | |
|---|---|---|---|---|
| | | High byte | Mid. byte | Low byte |
| +$0 | | ① Channel 1 | | |
| +$1 | | ② Channel 2 | | |
| +$2 | | ③ Channel 3 | | |
| +$3 | | ④ Channel 4 | | |
| +$4 | | | | |
| +$5 | | | | |
| +$6 | | | | |
| +$7 | | | | |

```
MS0,MI21=$6C8800DCC0A1
MS0,MI22=$6C8801DCC0A2
MS0,MI23=$6C8802DCC0A3
MS0,MI24=$6C8803DCC0A5
```




MACRO Station IC # 0

### 1. MACRO station setup

```
MS0,MI19=$4                    // Transfer data once every 4 phase clock (servo default)
MS0,MI975=$C                   // Activate I/O nodes 2 and 3 at MACRO station
MS0,MI20=$F                    // Activate transfer methods MI21, MI22, MI23, MI24

MS0,MI21=$6C8800DCC0A1         // Copy upper 16-bit data from Y:$8800 to X:$C0A1 (Node 2)
MS0,MI22=$6C8801DCC0A2         // Copy upper 16-bit data from Y:$8801 to X:$C0A2 (Node 2)
MS0,MI23=$6C8802DCC0A3         // Copy upper 16-bit data from Y:$8802 to X:$C0A3 (Node 2)
MS0,MI24=$6C8803DCC0A5         // Copy upper 16-bit data from Y:$8803 to X:$C0A5 (Node 3)

MSSAVE0                        // Save these changes to MACRO station
MS$$$0                         // Reset MACRO station
```

### 2. MACRO Ring Master setup

```
I6841=$0FC03F                  // Enable servo nodes 0,1,4,5 and I/O nodes 2,3

M980->X:$78421,8,16 // ADC1, Master register corresponding to 1st 16-bit word on Node 2
M981->X:$78422,8,16 // ADC2, Master register corresponding to 2nd 16-bit word on Node 2
M982->X:$78423,8,16 // ADC3, Master register corresponding to 3rd 16-bit word on Node 2
M983->X:$78425,8,16 // ADC4, Master register corresponding to 1st 16-bit word on Node 3
```

This example assumes that 4 other motors are controlled by Ring Master in other MACRO station such that I6841 is set to $0FC03F. Now the M-Variables can be used in PLCs or motion programs for data acquisition purposes.

***Note***

If a Non-Turbo PMAC2 Ultralite (legacy) is used as Master, the node enable variable should be I996, not I6841.The ADC reading registers then should be the same as I/O node addresses on a MACRO Station, like $C0XX, instead of $7XXXX.

## Testing Analog Inputs

The Analog Inputs can be brought into the ACC-28E as single ended (ADC+ & Ground) or differential (ADC+ & ADC-) signals.

***Note***

In single-ended mode, the ADC- for the channel (e.g. ADC1- for channel #1) should be tied to analog ground for full resolution and proper operation; do not leave the pin floating.

Reading the input signals in software counts using the predefined M-Variables should show the following:

| **ADC Input** | **Single-Ended [V]**<br>**ADC+ <=> AGND** | **Differential [V]**<br>**ADC+ <=> ADC-** | **Software Counts** |
|---|---|---|---|
| Unipolar Mode | 0 | 0 | 0 |
| | 5 | 5 | 32768 |
| | 10 | 10 | 65535 |
| Bipolar Mode | -10 | -10 | 0 |
| | 0 | 0 | 32768 |
| | 10 | 10 | 65535 |

# MACRO Data Transfer via Servo Nodes

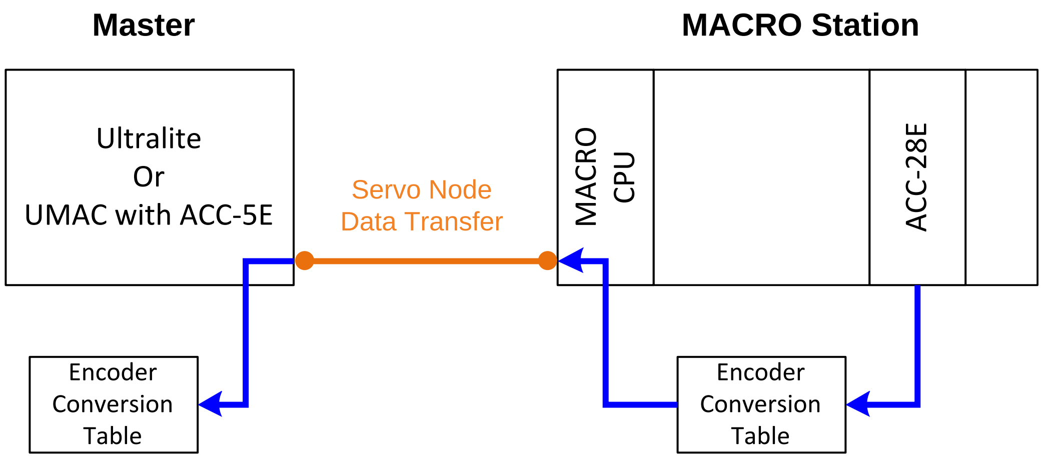
Similar to ACC-28E in a Turbo UMAC or a Power UMAC, the A/D feedback from ACC-28E on a MACRO station can also be used as servo feedback and/or absolute power-on position.

*Note*

Make sure that the servo nodes utilized in data transfer are enabled on both Master and MACRO Station sides.

## Using Analog Input for Servo Feedback

The servo node data transfer scheme from MARCO station to Master is illustrated as following.




The setup for servo feedback utilizes the Encoder Conversion Table (ECT) on the MACRO station, which is a different table from the ECT on Master. The setup procedure is as follows:

- On the MACRO station:
    1. Set up the ECT on the Station to bring data from the ACC-28E with the desired setting (position or velocity). This utilizes **MS{anynode},MI120–MI151, MI1120–MI1151**.
    2. Bring the result from the ECT to a servo node for data transfer to the Master. This utilizes **MS{anynode}, MI101–MI108, MI1101–1108**.
- On the Ring Master:
    1. Set up the ECT to process the data in servo node.
    2. Set **Ixx03** and **Ixx04** to the ECT entry's result register address to complete the feedback loop.

Usually, for a servo node on Master side, the already ECT has the default settings set by the firmware. In order to get the correct feedback from a desired channel, **Ixx03** and **Ixx04** need to be pointing to the correct output address of the ECT entry, which has the processed servo node data transferred back from the MACRO station.

## Encoder Conversion Table on MACRO Station

ECT entries are set up by the following MI-Variables:

| MACRO IC Number | MI-Variable | Corresponding Address |
|---|---|---|
| 0 | **MS{anynode},MI120–MI151** | $0010–$002F |
| 1 | **MS{anynode},MI1120–MI1151** | $0090–$00AF |

Each ECT MI-Variable is a 6-hex-digit number which contains the source address (digits 3–6) and the conversion method (digits 1–2).

| **Hex Digit #** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **Content** | Method | | Source Address | | | |
| **Example** | 1 | 8 | 8 | 8 | 0 | 0 |

The example above is utilizing:
Method: $18, ACC-28 style A/D converter, unsigned value.
Address: Y:$8800, ACC-28E 1st base address.

For ACC-28E, there are 4 different methods listed below to process the data from a base address:

| Method | Conversion Process | Polarity | # of lines |
|---|---|---|---|
| $10 | ACC-28 style A/D converter | Signed | 1 |
| $18 | ACC-28 style A/D converter | Unsigned | 1 |
| $50 | Integrated ACC-28 style A/D converter | Signed | 2 |
| $58 | Integrated ACC-28 style A/D converter | Unsigned | 2 |

Usually only the unsigned methods are used, and the source addresses for these methods are treated as Y-Registers since ACC-28E has its data in Y-Register addresses.

For method **$18**, the processed result is in the X-Register of the corresponding MI-Variable used for the conversion. For example, if **MI120** is used, then the result is in X:$0010.

For method **$58**, the processed result is in the last X-Register of the corresponding MI-Variables used for the conversion. For example, if **MI120** and **MI121** are used, then the result is in X:$0011. The 2nd line entry of integrated method contains a bias term, which is a 24-bit number subtracted from the source A/D data before numerical integration.

## Servo Data Transfer on MACRO Station

Ongoing position servo data is transferred to a node by setting the following variables to a data address:

| MACRO IC 0 | | | MACRO IC 1 | | |
|---|---|---|---|---|---|
| **MI-Variable** | **Node Number** | **Default Address** | **MI-Variable** | **Node Number** | **Default Address** |
| **MI101** | 0 | $0010 | **MI1101** | 0 | $0090 |
| **MI102** | 1 | $0011 | **MI1102** | 1 | $0091 |
| **MI103** | 4 | $0012 | **MI1103** | 4 | $0092 |
| **MI104** | 5 | $0013 | **MI1104** | 5 | $0093 |
| **MI105** | 8 | $0014 | **MI1105** | 8 | $0094 |
| **MI106** | 9 | $0015 | **MI1106** | 9 | $0095 |
| **MI107** | 12 | $0016 | **MI1107** | 12 | $0096 |
| **MI108** | 13 | $0017 | **MI1108** | 13 | $0097 |

The addresses here all refer to X-Registers since the servo position feedback data from the ECT will always be in X-Registers. The MI-Variables' default values correspond to the first 8 lines of the ECT addresses. **Example: MI101=$0010** means transfer servo data in X:$0010 to Servo Node 0 on MACRO Station.

**Servo Position Feedback Example**

On a MACRO station, configure 4 ADCs from ACC-28E for unsigned position feedback and transfer the servo feedback data to the Ring Master's servo nodes, assuming that these servo nodes are enabled on both the MACRO Station and the Ring Master sides.

### 1. MACRO Station Setup

Bring 4 ADCs' data from an ACC-28E (at base address $8800) to the Encoder Conversion Table, and then bring the ECT result to servo nodes on the MACRO Station side.

```
MS0,MI120=$188800              // Process ADC1 from Y:$8800 as unsigned pos., result in X:$0010
MS0,MI121=$188801              // Process ADC2 from Y:$8801 as unsigned pos., result in X:$0011
MS0,MI122=$188802              // Process ADC3 from Y:$8802 as unsigned pos., result in X:$0012
MS0,MI123=$188803              // Process ADC4 from Y:$8803 as unsigned pos., result in X:$0013

MS0,101=$0010                  // Transfer processed ADC1 result in X:$0010 to Node 0
MS0,102=$0011                  // Transfer processed ADC2 result in X:$0011 to Node 1
MS0,103=$0012                  // Transfer processed ADC3 result in X:$0012 to Node 4
MS0,104=$0013                  // Transfer processed ADC4 result in X:$0013 to Node 5
```

### 2. MACRO Ring Master setup

Bring servo node data to the ECT on the Master side, and then point **Ixx03** and **Ixx04** to the ECT result addresses for position and velocity loop feedback.

```
I8000=$2F8420                  // $280000+$078420, parallel Y word no filter, mode=1,from Node 0
I8001=018000                   // Use 24 bits data, starting at bit 0
I8002=$2F8424                  // $280000+$078424, parallel Y word no filter, mode=1,from Node 1
I8003=018000                   // Use 24 bits data, starting at bit 0
I8004=$2F8428                  // $280000+$078428, parallel Y word no filter, mode=1,from Node 4
I8005=018000                   // Use 24 bits data, starting at bit 0
I8006=$2F842C                  // $280000+$07842C, parallel Y word no filter, mode=1,from Node 5
I8007=018000                   // Use 24 bits data, starting at bit 0

I103=$3502 I104=$3502 // Position and velocity feedback address as 2nd line of ECT @I8001
I203=$3504 I204=$3504 // Position and velocity feedback address as 4th line of ECT @I8003
I303=$3506 I304=$3506 // Position and velocity feedback address as 6th line of ECT @I8005
I403=$3508 I404=$3508 // Position and velocity feedback address as 8th line of ECT @I8007
```

### Velocity Servo Feedback Example

On a MACRO station, configure 4 ADCs from ACC-28E for unsigned velocity feedback using the integrated data method, and then transfer the velocity feedback data to the Ring Master's servo nodes, assuming that the servo nodes are enabled on both the MACRO Station and the Ring Master sides.

#### 1. MACRO Station setup

Bring 4 ADCs' data from an ACC-28E (base address $8800) to the Encoder Conversion Table, and then bring the ECT result to the servo nodes on the MACRO Station.

```
MS0,MI120=$588800                // Process ADC1 from Y:$8800 as unsigned velocity,
MS0,MI121=$0                     // No bias term, result in X:$0011
MS0,MI122=$588801                // Process ADC2 from Y:$8801 as unsigned velocity,
MS0,MI123=$0                     // No bias term, result in X:$0013
MS0,MI124=$588802                // Process ADC3 from Y:$8802 as unsigned velocity,
MS0,MI125=$0                     // No bias term, result in X:$0015
MS0,MI126=$588803                // Process ADC4 from Y:$8803 as unsigned velocity,
MS0,MI127=$0                     // No bias term, result in X:$0017

MS0,101=$0011                    // Transfer processed ADC1 result in X:$0010 to Node 0
MS0,102=$0013                    // Transfer processed ADC2 result in X:$0011 to Node 1
MS0,103=$0015                    // Transfer processed ADC3 result in X:$0012 to Node 4
MS0,104=$0017                    // Transfer processed ADC4 result in X:$0013 to Node 5
```

#### 2. MACRO Ring Master setup

Bring servo node data to the ECT on the Master side, and then point **Ixx03** and **Ixx04** to the ECT result addresses for position and velocity loop feedback.

```
I8000=$2F8420                    // $280000+$078420, parallel Y word no filter, mode=1,from Node 0
I8001=018000                     // Use 24 bits data, starting at bit 0
I8002=$2F8424                    // $280000+$078424, parallel Y word no filter, mode=1,from Node 1
I8003=018000                     // Use 24 bits data, starting at bit 0
I8004=$2F8428                    // $280000+$078428, parallel Y word no filter, mode=1,from Node 4
I8005=018000                     // Use 24 bits data, starting at bit 0
I8006=$2F842C                    // $280000+$07842C, parallel Y word no filter, mode=1,from Node 5
I8007=018000                     // Use 24 bits data, starting at bit 0

I103=$3502 I104=$3502 // Position and velocity feedback address as 2nd line of ECT @I8001
I203=$3504 I204=$3504 // Position and velocity feedback address as 4th line of ECT @I8003
I303=$3506 I304=$3506 // Position and velocity feedback address as 6th line of ECT @I8005
I403=$3508 I404=$3508 // Position and velocity feedback address as 8th line of ECT @I8007
```

## Using an Analog Input for Power-On Positioning

The ACC-28E ADC results are usually read as unsigned values and can be used for absolute power-on positioning. An absolute power-on position can be obtained at power up or upon request (with the **#*n*$* for motor *n***) after proper setting.

The setup procedure for reading the absolute power-on position over MACRO is as follows:

- On the MACRO Station:

  Use **MS{anynode}, MI111–MI118** for MACRO IC 0, and **MI1111–MI1118** for MACRO IC 1, to set the absolute power-on position reading feature up on the MACRO station side, and to prepare to transfer the absolute position to the Master over MACRO.
- On the Ring Master:

  Configure **Ixx10** and/or **Ixx95** to specify the node number and reading format for receiving absolute positions from the node.

***Note***

The absolute power-on position reading feature is only provided with the firmware versions below:

MACRO Station: MACRO firmware version 1.114 or newer.
Ring Master: Non-Turbo Ultralite, firmware version 1.16H or newer; Turbo Ultralite or UMAC with ACC-5E, firmware version 1.936 or newer.

### On the MACRO Station

MI-Variables for the absolute power-on position reading feature of each node are as follows:

| **MACRO IC 0** | | **MACRO IC 1** | |
|---|---|---|---|
| **MI-Variable** | **Node Number** | **MI-Variable** | **Node Number** |
| **MI111** | 0 | **MI1111** | 0 |
| **MI112** | 1 | **MI1112** | 1 |
| **MI113** | 4 | **MI1113** | 4 |
| **MI114** | 5 | **MI1114** | 5 |
| **MI115** | 8 | **MI1115** | 8 |
| **MI116** | 9 | **MI1116** | 9 |
| **MI117** | 12 | **MI1117** | 12 |
| **MI118** | 13 | **MI1118** | 13 |

And the format of these MI-Variables is as follows:

| **Hex Digit #** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **Content** | Method | | Source Address | | | |
| **Example** | 3 | 1 | 8 | 8 | 0 | 0 |

For ACC-28E, the method digits (hex digit 1–2) can only be **$31** (unsigned) or **$B1** (signed). Usually, the unsigned method is used. For example, setting **MI111=$318800** will process data in Y:$8800 as unsigned and uses it as the absolute power-on position for node 0.

## On the Ring Master

- Non-Turbo Ultralite: **Ix10** (Motor x Power-On Servo Position Address)
- Turbo Ultralite / UMAC with ACC-5E: **Ixx10** (Motor xx Power-On Servo Position Address)
  **Ixx95** (Motor xx Power-On Servo Position Format)

**For Non-Turbo Ultralite**
**Ix10=$74000n**. **$74** is used for the MACRO Station Parallel Input method, and ***n*** is the servo node number, which can be 0, 1, 4, 5, 8, 9, $C (12 in decimal) or $D (13 in decimal).

**For Turbo Ultralite or UMAC with ACC-5E**
Format of **Ixx10**:

| **Hex Digit #** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **Content** | N/A | | | | IC Number | Node Number |
| **Example** | 0 | 0 | 0 | 0* | 0 | 1 |

**I110=$000001** means that Motor 1's power-on position is coming from Node 1 of MACRO IC 0 over MACRO.

***Note***

Digit 4 is usually 0, but if MACRO IC 0 Node 0 is used for power-on position, then digit 4 must = 1 since **Ixx10** = $0 will disable power-on position reading

The following table shows all the values for **Ixx10**:

| **MACRO Node Number** | **Ixx10 for MACRO IC 0** | **Ixx10 for MACRO IC 1** | **Ixx10 for MACRO IC 2** | **Ixx10 for MACRO IC 3** |
|---|---|---|---|---|
| 0 | $000100 | $000010 | $000020 | $000030 |
| 1 | $000001 | $000011 | $000021 | $000031 |
| 4 | $000004 | $000014 | $000024 | $000034 |
| 5 | $000005 | $000015 | $000025 | $000035 |
| 8 | $000008 | $000018 | $000028 | $000038 |
| 9 | $000009 | $000019 | $000029 | $000039 |
| 12 | $00000C | $00001C | $00002C | $00003C |
| 13 | $00000D | $00001D | $00002D | $00003D |

**Ixx95**, the absolute power-on position reading format, can be set to:

- **$740000** for unsigned MACRO Station Parallel Input, absolute power-on position.
- **$F40000** for signed MACRO Station Parallel Input, absolute power-on position.

Again, the unsigned value is usually used here.

### Example: Reading Absolute Power-On Position over MACRO

On a MACRO station, configure 4 ADCs from ACC-28E for unsigned absolute power-on position feedback and then transfer the absolute position data back to the Ring Master, assuming that the servo nodes are already enabled on both the MACRO Station and the Ring Master.

**1. MACRO Station setup**

Use 4 channels of ADC data from an ACC-28E (base address $8800) as power-on position for the motors on Servo Nodes 0, 1, 4, and 5.

```
MS0,MI111=$318800                // Set ADC1 from Y:$8800 as unsigned abs. power-on pos., Node 0
MS0,MI112=$318801                // Set ADC2 from Y:$8801 as unsigned abs. power-on pos., Node 1
MS0,MI113=$318802                // Set ADC3 from Y:$8802 as unsigned abs. power-on pos., Node 4
MS0,MI114=$318803                // Set ADC4 from Y:$8803 as unsigned abs. power-on pos., Node 5
```

**2. MACRO Ring Master setup**

On a Turbo Ultralite, set **Ixx10** and **Ixx95** up to receive absolute power-on positions over MACRO for Motors 1–4.

```
I110=$000100                     // Motor 1 absolute power-on position from MACRO IC 0, Node 0
I210=$000001                     // Motor 2 absolute power-on position from MACRO IC 0, Node 1
I310=$000004                     // Motor 3 absolute power-on position from MACRO IC 0, Node 4
I410=$000005                     // Motor 4 absolute power-on position from MACRO IC 0, Node 5

I195=$740000                     // Motor 1 abs. power-on pos. format as unsigned value over MACRO
I295=$740000                     // Motor 2 abs. power-on pos. format as unsigned value over MACRO
I395=$740000                     // Motor 3 abs. power-on pos. format as unsigned value over MACRO
I495=$740000                     // Motor 4 abs. power-on pos. format as unsigned value over MACRO
```

## Direct Verification of ACC-28E ADCs

MACRO MI-Variables **MS{anynode},MI198** and **MI199** can be used to read from or write to virtually any MACRO Station memory location. This can be useful especially when trying to test the hardware on the MACRO Station. **MI198** contains the format and the register address of the data, and **MI199** is used to read from or write to the register with the format specified by **MI198**.

The format of **MI198** is as below:

| **Hex Digit #** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **Content** | Method | | Source Address | | | |
| **Example** | 6 | C | 8 | 8 | 0 | 0 |

For ACC-28E, the recommended method is **$6C**: Y-Register, starting from bit 8, bit width 16 bits, unsigned value (high 16-bit value of a 24-bit register).

**Example:**

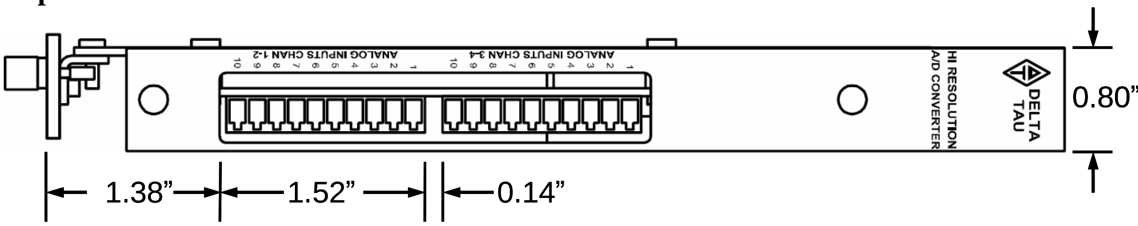
Using **MI198** and **MI199** to read ADC1 and ADC2 of ACC-28E with base address $9800.

```
MS0,MI198=$6C9800                ; Set ADC1 reading from Y:$9800, unsigned, High 16 bits
MS0,MI199                        ; Read ADC1

// $000000004B78                 ; Response: 19320 ADC bits

MS0,MI198=$6C9801                ; Set ADC2 reading from Y:$9801, unsigned, High 16 bits
MS0,MI199                        ; Read ADC2

// $00000000B4A6                 ; Response: 46186 ADC bits
```

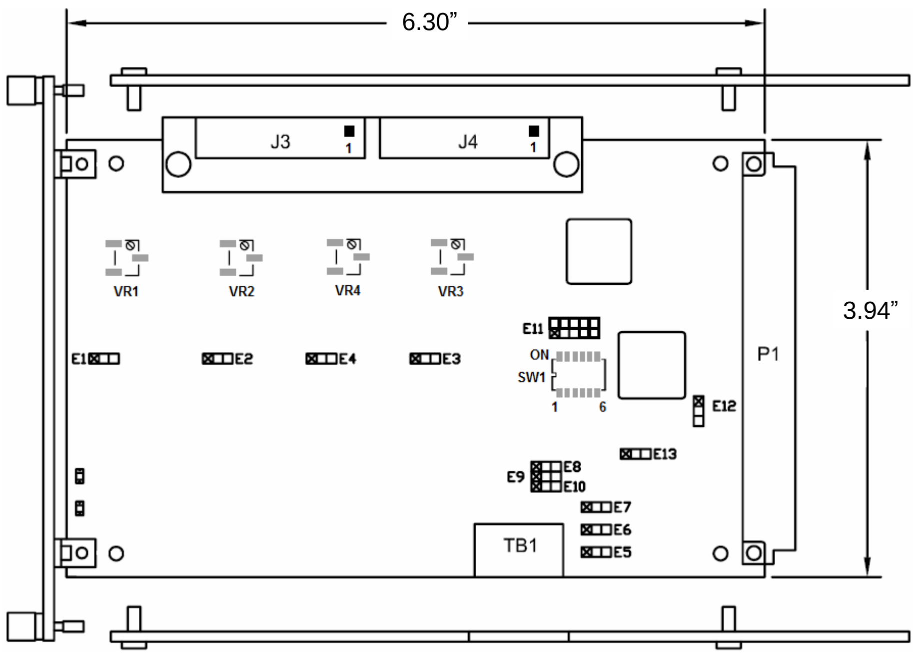
# LAYOUT & PINOUTS

## Terminal Block Option

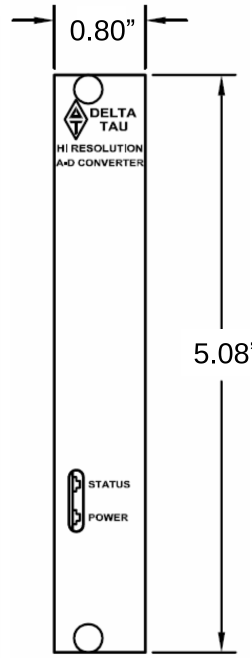
**Top View**



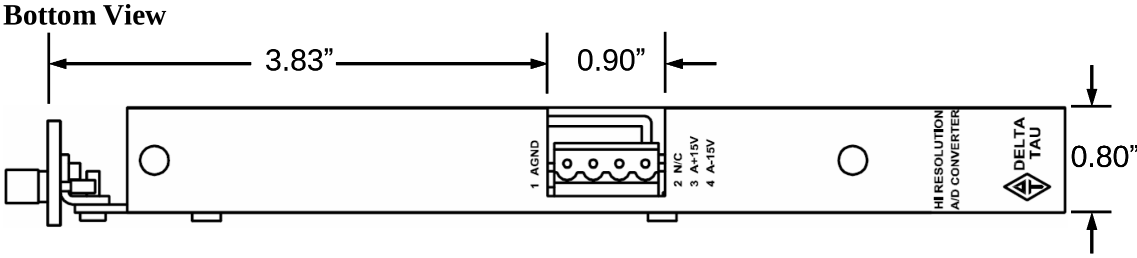

**Side View**




**Front View**



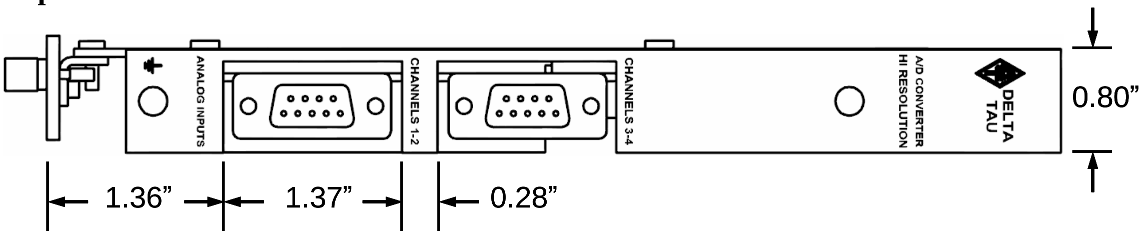

**Bottom View**



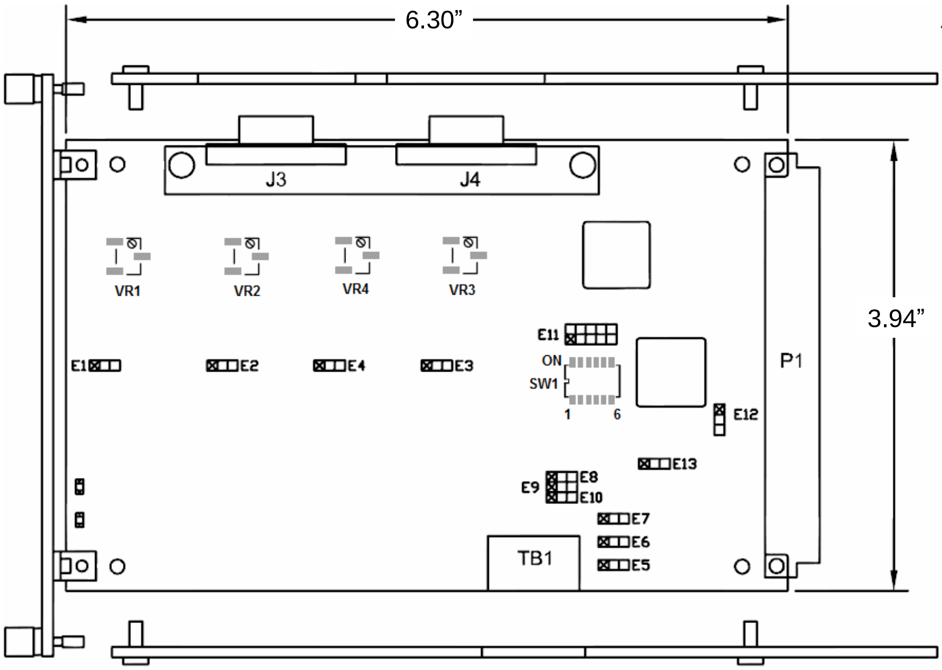

All dimensions are in inches.

## DB9 Option

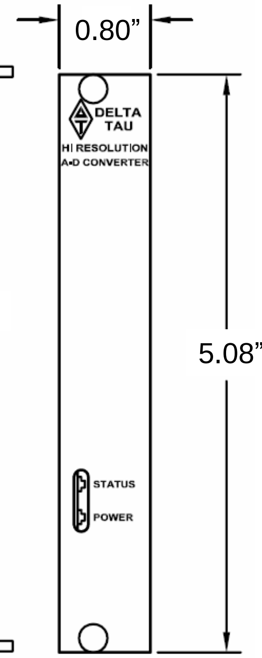
**Top View**



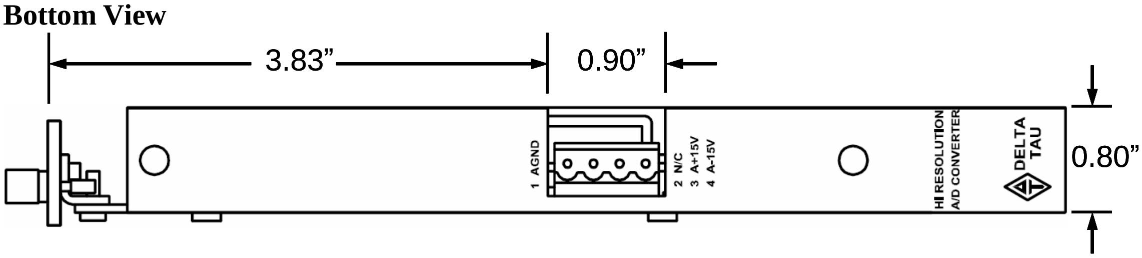

**Side View**




**Front View**

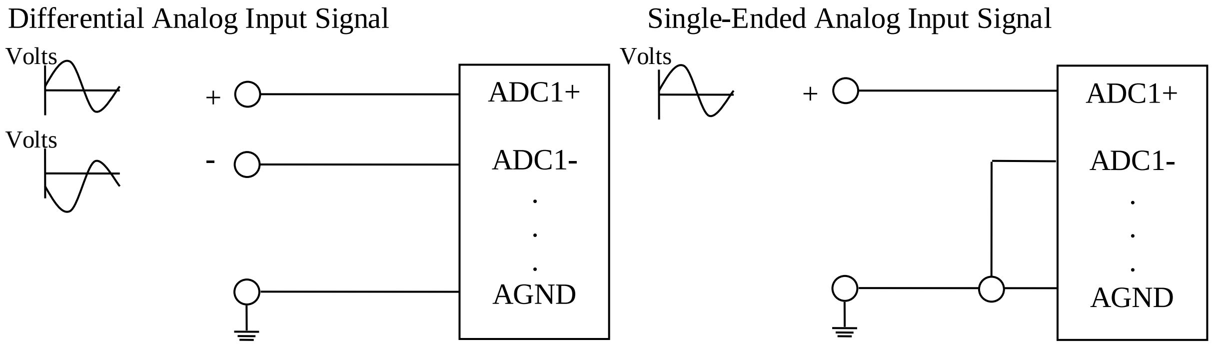



**Bottom View**




All dimensions are in inches. Drawings are not to scale.

## Sample Wiring Diagram




## P1: Backplane Bus

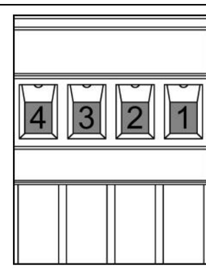
This connector is used for interface to UMAC's processor bus via the backplane of the 3U rack. The signals that are brought in through this connector are buffered on board.
This is a 20-pin header that is used for factory calibration.

## TB1 (4-Pin Terminal Block)

***Caution***

Do not use TB1 when the ACC-28E is plugged into the backplane.

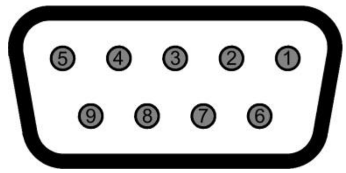
This 4-pin terminal block provides the connection for an external power supply (standalone mode).

<table>
<tr><td colspan="2">TB1: 4-Point Terminal Block</td><td colspan="3">
</td></tr>
<tr><th>Pin #</th><th>Symbol</th><th>Function</th><th>Description</th><th>Notes</th></tr>
<tr><td>1</td><td>AGND</td><td>Common</td><td>Digital ground</td><td></td></tr>
<tr><td>2</td><td>N.C.</td><td></td><td>Not connected</td><td></td></tr>
<tr><td>3</td><td>+15VDC</td><td>Input</td><td>+15 VDC power input</td><td>200 mA required</td></tr>
<tr><td>4</td><td>-15VDC</td><td>Input</td><td>-15 VDC power input</td><td>70 mA required</td></tr>
</table>

# DB9 Connector Option

## J3A – ADC Inputs 1 and 2

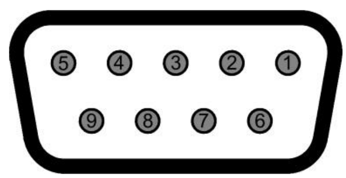
**J3A: D-sub DA-9F**
**Mating: D-sub DA-9M**




| Pin# | Symbol | Function | Description |
|---|---|---|---|
| 1 | -5 VDC | Output | -5 VDC reference output |
| 2 | VREF | Output | 4.096 VDC precision reference |
| 3 | ADC2- | Input | A-D Conv. Channel 2- |
| 4 | AGND | Gnd. | Shield |
| 5 | ADC1+ | Input | A-D Conv. Channel 1+ |
| 6 | +5 VDC | Output | +5 VDC reference output |
| 7 | AGND | Gnd. | Shield |
| 8 | ADC2+ | Input | A-D Conv. Channel 2+ |
| 9 | ADC1- | Input | A-D Conv. Channel 1- |

## J4A – ADC Inputs 3 and 4

**J3A: D-sub DA-9F**
**Mating: D-sub DA-9M**




| Pin# | Symbol | Function | Description |
|---|---|---|---|
| 1 | -5 VDC | Output | -5 VDC reference output |
| 2 | VREF | Output | 4.096 VDC precision reference |
| 3 | ADC4- | Input | A-D Conv. Channel 4- |
| 4 | AGND | Gnd. | Shield |
| 5 | ADC3+ | Input | A-D Conv. Channel 3+ |
| 6 | +5 VDC | Output | +5 VDC reference output |
| 7 | AGND | Gnd. | Shield |
| 8 | ADC4+ | Input | A-D Conv. Channel 4+ |
| 9 | ADC3- | Input | A-D Conv. Channel 3- |

.

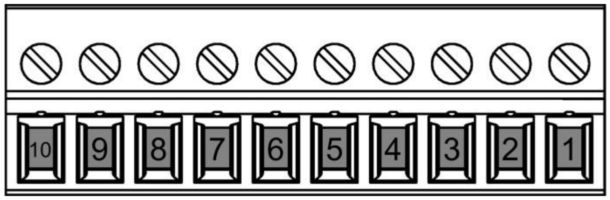
*Note*

- DB9 shell is connected to the UMAC chassis ground
- Shields are internally connected to the ground plane inside the ACC-28E. Shields are normally connected at only one end of the wire only (this eliminates possible system ground loops).
- VREF is a buffered tap from the A-D precision reference. External hardware that uses this signal reference will typically scale it for a full-scale A-D voltage input

# Terminal Block Option

## J3 – ADC Inputs 1 and 2

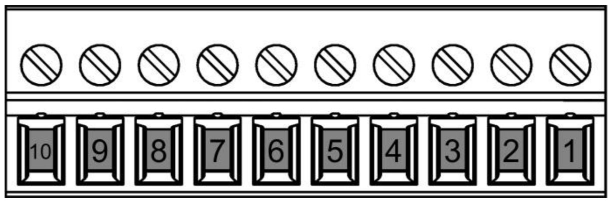
**J3 Terminal Block**




| Pin | Symbol | Function | Description |
|---|---|---|---|
| 1 | ADC1+ | Input | A-D Converter Channel 1+ |
| 2 | ADC1- | Input | A-D Converter Channel 1- |
| 3 | AGND | GND. | Shield |
| 4 | ADC2+ | Input | A-D Converter Channel 2+ |
| 5 | ADC2- | Input | A-D Converter Channel 2- |
| 6 | AGND | GND. | Shield |
| 7 | VREF | Output | 4.096 VDC precision reference |
| 8 | AGND | GND. | Shield |
| 9 | +5 VDC | Output | +5 VDC reference output |
| 10 | -5 VDC | Output | -5 VDC reference output |

## J4 – ADC Inputs 3 and 4

**J4 Terminal Block**




| Pin | Symbol | Function | Description |
|---|---|---|---|
| 1 | ADC3+ | Input | A-D Converter Channel 3+ |
| 2 | ADC3- | Input | A-D Converter Channel 3- |
| 3 | AGND | Gnd. | Shield |
| 4 | ADC4+ | Input | A-D Converter Channel 4+ |
| 5 | ADC4- | Input | A-D Converter Channel 4- |
| 6 | AGND | Gnd. | Shield |
| 7 | VREF | Output | 4.096 VDC precision reference |
| 8 | AGND | Gnd. | Shield |
| 9 | +5 VDC | Output | +5 VDC reference output |
| 10 | -5 VDC | Output | -5 VDC reference output |

***Note***

- Shields are internally connected to the ground plane inside the ACC-28E. Shields are normally connected at only one end of the wire only (this eliminates possible system ground loops).
- VREF is a buffered tap from the A-D precision reference. External hardware that uses this signal reference typically will scale it for a full-scale A-D voltage input

# CARD IDENTIFICATION

## Card ID Format

Card identification can be read from the following address:

| **Base Addr. + (Bank Sel. =0)** | **D4** | **D3** | **D2** | **D1** | **D0** |
|---|---|---|---|---|---|
| | Phase_Dir | Vendor ID ID:03 | Vendor ID ID:02 | Vendor ID ID:01 | Vendor ID ID:00 |
| | Bank Sel. =0 | Vendor ID ID:13 | Vendor ID ID:12 | Vendor ID ID:11 | Vendor ID ID:10 |
| | Card option CO:04 | Card Option CO:03 | Card Option CO:02 | Card Option CO:01 | Card Option CO:00 |
| | Card option CO:09 | Card Option CO:08 | Card Option CO:07 | Card Option CO:06 | Card Option CO:05 |

| **Base Addr. + (Bank Sel. =1)** | Phase_Dir | Revision # CR:03 | Revision # CR:02 | Revision # CR:01 | Revision # CR:00 |
|---|---|---|---|---|---|
| | Bank Sel. = 1 | Card ID CT: 03 | Card ID CT: 02 | Card ID CT: 01 | Card ID CT: 00 |
| | Card ID CT: 08 | Card ID CT: 07 | Card ID CT: 06 | Card ID CT: 05 | Card ID CT: 04 |
| | Card ID CT: 13 | Card ID CT: 12 | Card ID CT: 11 | Card ID CT: 10 | Card ID CT: 09 |

The card identification number of all Delta Tau cards is derived from the last four digits of the PCB assembly number. For example, the ACC-28E card assembly number is 603404. Convert the last four digits into a hex number (i.e. 3404 = $D4C).

- This will be the card identification for ACC-28E.
- Vender identification number = 1 for Delta Tau.
- Revision number for this card is 1.
- Option 1: Additional two axes (present if ACC-28E has 4 channels)

# DECLARATION OF CONFORMITY

**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 28E
**Model Number:** 603404
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 ≤ 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

Yolande Cano
Quality Assurance Manager

Mark of Compliance

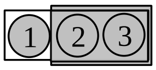
CE

# APPENDIX A: E-POINT JUMPERS

Refer to the layout diagram of ACC-28E for the location of the jumpers on the board.

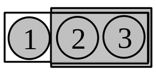
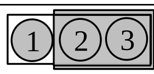
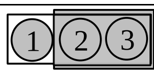
## E1, E2, E3, E4 — Unipolar/Bipolar Conversion Mode

Jumpers E1 through E4 are for selecting the conversion mode of ADC channels #1 through #4.

| E Point | Pin Layout | Description | Default |
|---|---|---|---|
| **E1, E2, E3, E4** |  | Jumper pin 1 to 2 for Unipolar conversion.<br>Jumper pin 2 to 3 for Bipolar conversion.<br>E1 - Channel #1; E2 - Channel #2<br>E4 - Channel #3; E3 - Channel #4 | 2 to 3 |

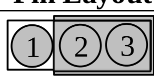
## E5, E6, E7 — Power Supply Selection

These jumpers allow the user to choose between the external DC power supply and its ground or the 3U rack's backplane DC power for the A/D converter.

| E Point | Pin Layout | Description | Default |
|---|---|---|---|
| **E5** |  | Jumper pin 1 to 2 to use external power supply common to the user supplied AGND.<br>Jumper pin 2 to 3 to use 3U backplane AGND. | 2 to 3 |
| **E6** |  | Jumper pin 1 to 2 to use external +15 VDC.<br>Jumper pin 2 to 3 to use 3U rack backplane +15 VDC. | 2 to 3 |
| **E7** |  | Jumper pin 1 to 2 to use external -15 VDC.<br>Jumper pin 2 to 3 to use 3U rack backplane -15 VDC. | 2 to 3 |

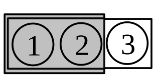
## E8, E9, E10 — Programming Jumpers

Used by the factory to download Xilinx programs. The jumpers will be installed in the right position when it is shipped and the user should not change or move them.

| E Point | Pin Layout | Description | Default |
|---|---|---|---|
| **E8, E9, E10** |  | Jumper pin 1 to 2 to connect the Xilinx loader cable.<br>Jumper pin 2 to 3 to use EEPROM. | 2 to 3 |

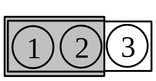
## E12 — Station Type Select

This jumper is used to choose either a Turbo/Power UMAC station, UMAC MACRO station, or a legacy Macro Station.

| E Point | Pin Layout | Description | Default |
|---|---|---|---|
| **E12** |  | Jumper pin 1 to 2 for Turbo 3U CPU, Power CPU, and MACRO CPU.<br>Jumper pin 2 to 3 for legacy MACRO CPU (before 6/00). | 1 to 2 |
| * For legacy MACRO Stations (part number 602804-100 thru 602804-104) | | | |

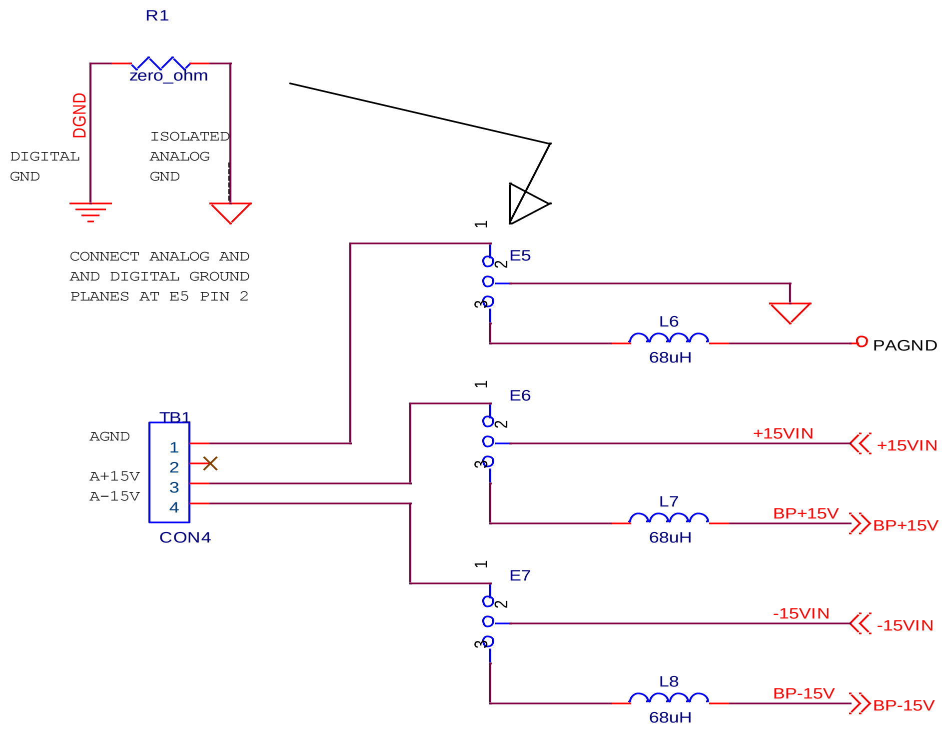
## E13 — Clock Select

This jumper is used to determine which clock to synchronize with for the A/D conversion.

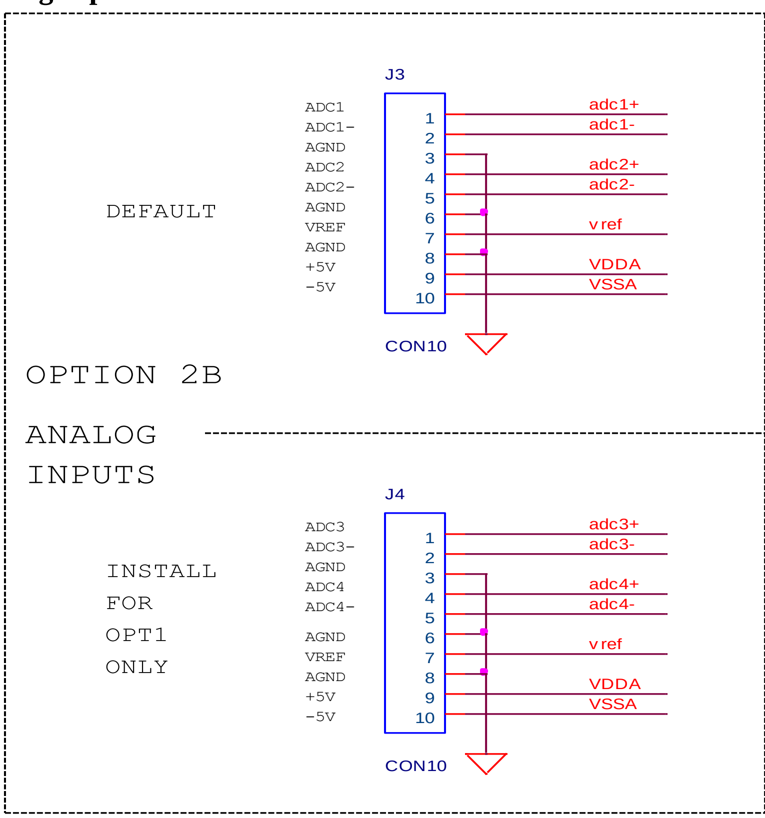
| E Point | Pin Layout | Description | Default |
|---|---|---|---|
| **E13** |  | Jumper pin 1 to 2 to use servo clock to start the A/D conversion.<br>Jumper pin 2 to 3 to use phase clock to start the A/D conversion. | 1 to 2 |

# APPENDIX B: SCHEMATICS

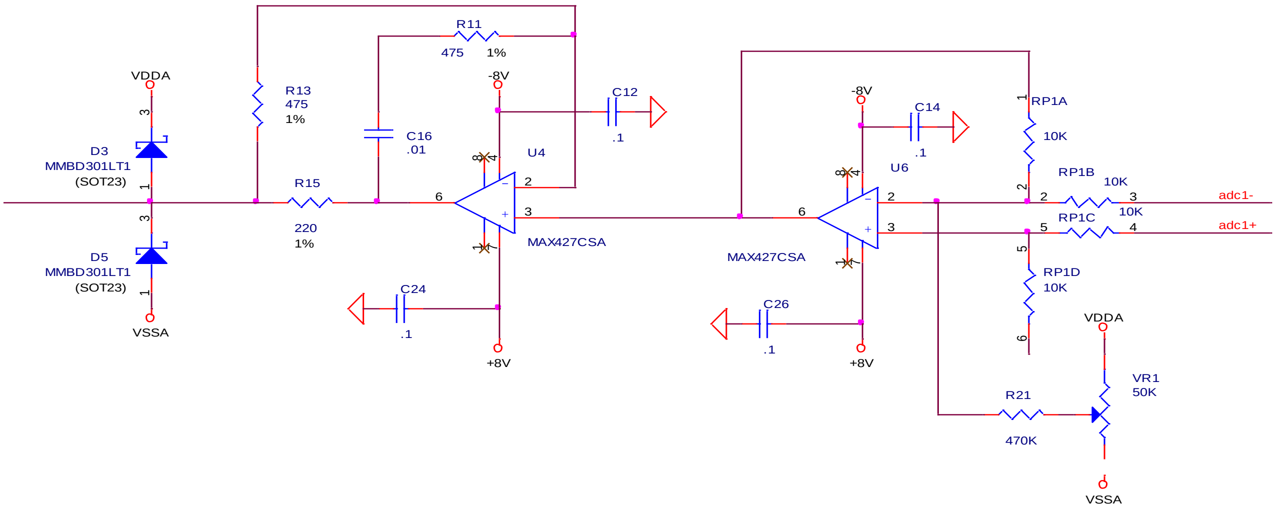
## TB1 Power Input Connector




## Terminal Block Analog Inputs



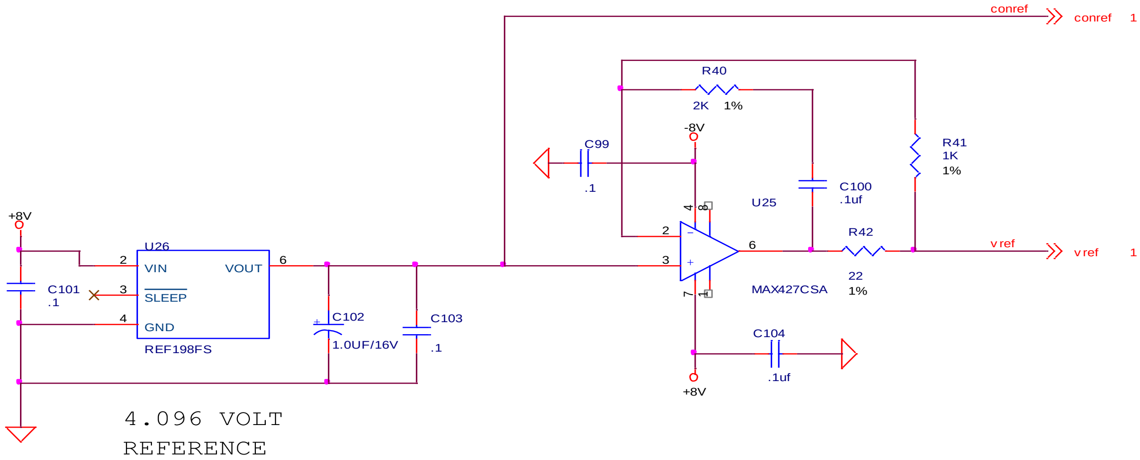
## ADC1+ Circuitry




**Note**

The circuitry for ADCs 2–4 is identical to ADC 1's, so it is not shown here.

## VREF Pin Circuitry on Analog Input Connector




## ± 15 VDC Pin Circuitry on Analog Input Connector