# MULTI-IO-ARD DAC I$^2$C Address Programming Procedure

SCIDYNE Corporation
Version: 1.00     Date: 8-15-23
Mark Durgin

## Description

The MULTI-IO-ARD Digital-to-Analog (DAC) chips do not use moveable jumpers to set their I2C addresses. Instead, the addresses are stored in EEPROM. Default addresses are pre-programmed at the factory. Use this procedure and accompanying software to change the addresses in the field.

## Intended Audience

This procedure assumes the user is familiar with the Arduino Integrated Development Environment (IDE) including the loading, modifying, compiling, and running Arduino sketches. Only instructions and information relevant to changing the MULTI-IO-ARD are described. If you are uncomfortable performing these tasks please seek assistance from a qualified engineer or programmer.

## Required Equipment

Along with the MULTI-IO-ARD board to be reprogrammed the following items will be needed:

- An Arduino board, e.g.; UNO, MEGA, DUE
- Computer with the Arduino IDE installed
- USB cable to connect computer to the Arduino board
- A Wire Jumper, about 4" long each
- MCP4728 programming sketch - **mcp4728_program_address.ino**
- I2C Scanner sketch - **Arduino_i2c_scanner.ino**
- Two Shorting Shunts

## Hardware Setup

Ensure the Arduino is disconnected from the computer's USB cable and is not powered by other means.

Set the MULTI-IO-ARD I/O VOLTAGE Jumper J9 to match the operating voltage of the Arduino being used. For example; The Arduino UNO and Mega require +5V, where a DUE requires 3.3V.

*WARNING: Incorrectly setting the J9 I/O VOLTAGE jumper can cause reprogramming to fail or can permanently damage the MULTI-IO-ARD and/or Arduino hardware.*

Enable the I2C Pull-Up resistors by placing shunts at the SDA and SCL positions of J2. If your version of the MULTI-IO-ARD is not equipped with J2 mounted, temporary wires should be installed. See figure 2. Plug the MULTI-IO-ARD on to the Arduino

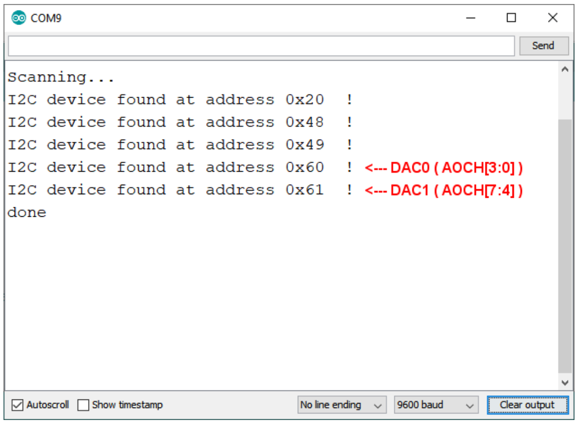
Apply power to the Arduino by connecting the USB cable.

## Determining existing DAC I2C addresses

It is necessary to know each DACs current I2C address to successfully complete this procedure. The factory default addresses are 0x60 for DAC0, and 0x61 for DAC1. DAC addresses range from 0x60 to 0x67.

If the addresses are known move on to the next section.

If the current DAC addresses are unknown, run the **Arduino_i2c_scanner.ino** sketch. Using the Serial Monitor, make note of the results.



```
Scanning...
I2C device found at address 0x20  !
I2C device found at address 0x48  !
I2C device found at address 0x49  !
I2C device found at address 0x60  !  <--- DAC0 ( AOCH[3:0] )
I2C device found at address 0x61  !  <--- DAC1 ( AOCH[7:4] )
done
```

*Figure 1 – Serial Monitor I2C Scanner Results*

As shown in Figure 1, the two DAC addresses are shown last in the list of I2C devices found. The DACs will have addresses within the range of 0x60 – 0x67.

## DAC Address Programming

Load the programming sketch **mcp4728_program_address.ino**.

Near the top of the sketch locate these lines of code and make any changes necessary to match your setup and reqirements:

```
/*****************************************************
 * IMPORTANT! - MUST SET THESE PARAMETERS BEFORE RUNNING PROGRAM
 */
uint8_t SDApin = 18;               // Sets the pin used for SDA signal:  MKR Zero=11;  UNO=18;  MEGA=20;  DUE=(70u);
uint8_t SCLpin = 19;               // Sets the pin used for SCL signal:  MKR Zero=12;  UNO=19;  MEGA=21;  DUE=(71u);
uint8_t LDACpin = 7;               // Sets the pin used to drive the LDAC of MCP4728, pulled LOW on PCB so drive High when needed
uint8_t I2C_OldAddress = 0x60;     // Existing Address of DAC being changed, valid addresses are 0x60-0x67
uint8_t I2C_NewAddress = 0x62;     // New Address of DAC being changed, valid addresses are 0x60-0x67
```

*Permitted DAC addresses are in the range of 0x60 to 0x67. The two DACs must have different addresses and the addresses cannot be used by any other device on the I2C bus. Typically, DAC0 is given a lower address than DAC1.*

If programming the address of DAC0 ( i.e., U2, AOCH[3:0] ), temporarily place a wire jumper between digital IO #7 and the MULTI-IO-ARD Pad labeled AP0 as shown in Figure 2A.

If programming the address of DAC1 ( i.e., U3, AOCH[7:4] ), temporarily place a wire jumper between digital IO #7 and the MULTI-IO-ARD Pad labeled AP1 as shown in Figure 2B.

Programming Wire Jumper Positions

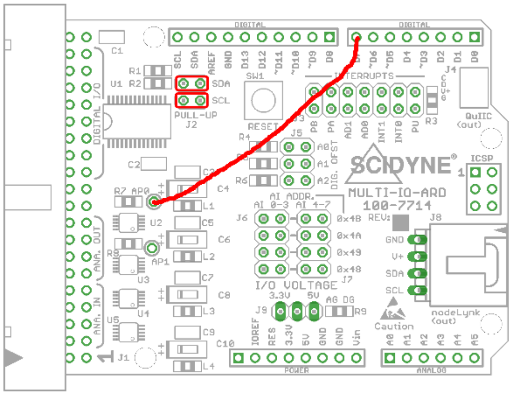


*Figure 2A - DAC0 Programming Jumper Position*

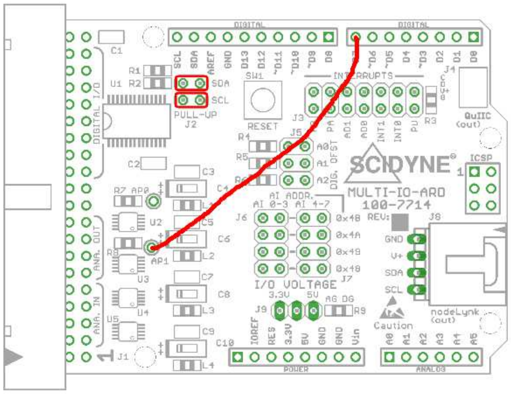
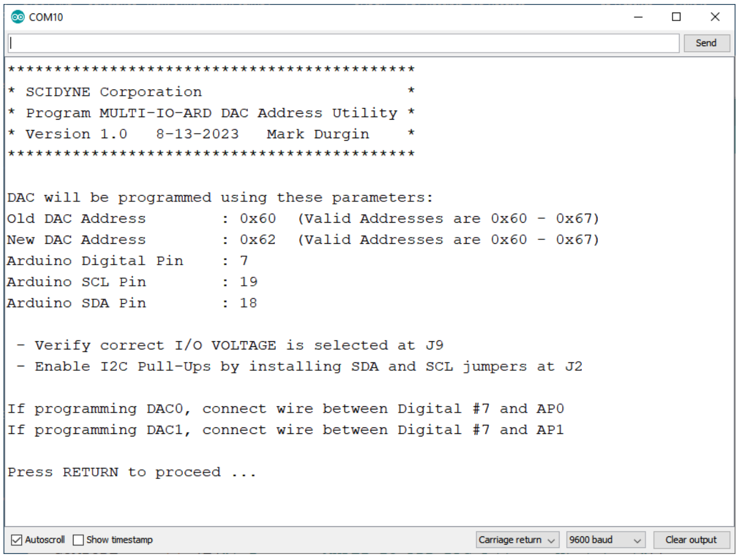


*Figure 2B - DAC1 Programming Jumper Position*

Compile and run the sketch. The Serial Monitor will display the screen shown in Figure 3.



```
*********************************************
* SCIDYNE Corporation                       *
* Program MULTI-IO-ARD DAC Address Utility *
* Version 1.0    8-13-2023    Mark Durgin    *
*********************************************

DAC will be programmed using these parameters:
Old DAC Address        : 0x60  (Valid Addresses are 0x60 - 0x67)
New DAC Address        : 0x62  (Valid Addresses are 0x60 - 0x67)
Arduino Digital Pin    : 7
Arduino SCL Pin        : 19
Arduino SDA Pin        : 18

 - Verify correct I/O VOLTAGE is selected at J9
 - Enable I2C Pull-Ups by installing SDA and SCL jumpers at J2

If programming DAC0, connect wire between Digital #7 and AP0
If programming DAC1, connect wire between Digital #7 and AP1

Press RETURN to proceed ...
```

*Figure 3 - Programmer Screen*

Press Carriage Return to start the programming process.

After programming completes one of two messages appear:

If Successful:
**"DAC Address successfully set to new value."**

If Not Successful:
**"DAC Address Set procedure FAILED! I2C ACK was missing at Byte number 2"**

To program a different DAC, move the wire jumper to the other APx pad. Make appropriate changes to the programming sketch, compile. download, and run again.

Results can be verified by loading and running the **Arduino_i2c_scanner.ino** sketch.

When finished, disconnect power from the Arduino, return the I/O Voltage jumper J9 to its original position, set SDA and SCL Pull-Ups jumpers at J2 to their original position.