MCP4725

MCP4725 I^{2}C DAC Arduino and chipKit library

Manual

Introduction:

This library has been made to easily interface and use the MCP4725 DAC with an Arduino or chipKit.

This library will default to I^2C Fast Mode (400 KHz) when using the hardware I^2C interface.

The library has not been tested in combination with the Wire library and I have no idea if they can share pins. Do not send me any questions about this. If you experience problems with pin-sharing you can move the MCP4725 SDA and SCL pins to any available pins on your development board. This library will in this case fall back to a software-based, TWI-/I^2C-like protocol which *will* require exclusive access to the pins used.

It should be noted that the output voltage from the MCP4725 is referenced to the operating voltage of the chip. If your operating voltage is below/above nominal the output will also be below/above what you expect from the set value.

If you are using a chipKit Uno32 or uC32 and you want to use the hardware I^2C interface you must remember to set the JP6 and JP8 jumpers to the I^2C position (closest to the analog pins).

From the MCP4725 datasheet:

> The MCP4725 is a low-power, high accuracy, single channel, 12-bit buffered voltage output Digital-to-Analog Convertor (DAC) with non-volatile memory (EEPROM). Its on-board precision output amplifier allows it to achieve rail-to-rail analog output swing.
>
> The DAC input and configuration data can be programmed to the non-volatile memory (EEPROM) by the user using I^2C interface command. The non-volatile memory feature enables the DAC device to hold the DAC input code during power-off time, and the DAC output is available immediately after power-up. This feature is very useful when the DAC device is used as a supporting device for other devices in the network.
>
> The device includes a Power-On-Reset (POR) circuit to ensure reliable power-up and an on-board charge pump for the EEPROM programming voltage. The DAC reference is driven from V_{DD} directly. In power-down mode, the output amplifier can be configured to present a low, medium, or high resistance output load.
>
> The MCP4725 has an external A0 address pin. This A0 pin can be tied to V_{DD} or V_{SS} of the user’s application board.

You can always find the latest version of the library at **http://www.RinkyDinkElectronics.com/**

For version information, please refer to **version.txt**.

Structures:

MCP4725_Status;

Structure to check the status of the DAC after calling getStatus().

Variables:

currentValue	The value of the DAC output
currentVoltage	The calculated output voltage based on the DAC value
currentPowerState	The current power down mode of the DAC
startupValue	The startup/power-on value of the DAC output
startupPowerState	The startup/power-on power down mode of the DAC

Usage:

```
MCP4725_Status s; // Define a structure named s of the MCP4725_Status-class
```

Notes: The values are updated only when calling getStatus().

Defined Literals:

Part number

For use with setDevice()

MCP4725A0:	0x60
MCP4725A1:	0x62
MCP4725A2:	0x64
MCP4725A3:	0x66

Power states

For use with setPowerDown() and storePowerDown()

MCP4725_POWERDOWN_OFF:	0x00
MCP4725_POWERDOWN_1K:	0x01
MCP4725_POWERDOWN_100K:	0x02
MCP4725_POWERDOWN_500K:	0x03
MCP4725_POWERDOWN_UNK:	0xFF

Voltage

Can be used with setVoltage() and storeVoltage()

MAX_VOLTAGE:	5.00f (AVR microcontrollers) 3.30f (ARM and PIC32 microcontrollers)

Functions:

MCP4725(SDA, SCL);	
The main class constructor.	
Parameters:	SDA: Pin connected to the SDA-pin of the MCP4725 SCL: Pin connected to the SCL-pin of the MCP4725
Usage:	MCP4725 dac(SDA, SCL); // Start an instance of the MCP4725 class using the hardware I^2C interface
Notes:	You can connect the MCP4725 to any available pin but if you use any other than hardware I^2C pin the library will fall back to a software-based, TWI-like protocol which will require exclusive access to the pins used, and you will also have to use appropriate, external pull-up resistors on the data and clock signals. External pull-up resistors are *always* needed on chipKit boards.

setDevice([device]);	
Select what device type you are using.	
Parameters:	Device: <Optional> MCP4725A0 (Default) MCP4725A1 MCP4725A2 MCP4725A3
Returns:	Nothing
Usage:	dac.setDevice(MCP4725A1); // Select the MCP4725A1 device type
Notes:	More information about the four different device types can be found in the MCP4725 datasheet. If you need to change the device type you *must* do so before calling begin().

begin([channel]);	
Get current time as a string.	
Parameters:	channel: **<Optional>** 0 (default) 1
Returns:	Nothing
Usage:	dac.begin(); // Initialize the library for use
Notes:	Which channel you need to select depends upon how the A0 pin of the MCP4725 is connected. ➔ A0 connected to V_{SS} = channel 0 ➔ A0 connected to V_{DD} = channel 1

getStatus();	
Get the current status from the MCP4725.	
Parameters:	None
Returns:	MCP4725_Status structure.
Usage:	s = dac.getStatus(); // Get the current DAC status and store it in s

setValue(value);

Write a 12-bit value to the DAC.

Parameters:	value: 0-4095
Returns:	Nothing
Usage:	dac.setValue(0); // Set the value to 0

setVoltage(voltage);

Write a specific voltage to the DAC.

Parameters:	voltage: 0.00 to 5.00 (AVR microcontrollers) 0.00 to 3.30 (ARM and PIC32 microcontrollers)
Returns:	Nothing
Usage:	dac.setVoltage(1.25f); // Set the DAC output to 1.25v
Notes:	In order for the output voltage to be exactly what you specify the operating voltage of the MCP4725 must be exactly 5.00v for AVR microcontrollers or 3.30v for ARM and PIC32 microcontrollers. *Any* deviation in the operating voltage of the MCP4725 will result in a subsequent deviation in the output voltage.

setPowerDown(value);

Set the power down mode of the MCP4725.

Parameters:	value: MCP4725_POWERDOWN_OFF, MCP4725_POWERDOWN_1K, MCP4725_POWERDOWN_100K or MCP4725_POWERDOWN_500K
Returns:	Nothing
Usage:	dac.setPowerDown(MCP4725_POWERDOWN_OFF); // Disable power down mode and restore normal output
Notes:	setPowerDown() will set the output of the MCP4725 to 0. setValue() and setVoltage() will automatically set the power down mode to MCP4725_POWERDOWN_OFF.

storeValue(value);

Write a 12-bit value to the DAC and store it in the internal EEPROM as the startup/power-on value.

Parameters:	value: 0-4095
Returns:	Nothing
Usage:	dac.storeValue(0); // Set the value to 0 and store that as the default power-on value as well

storeVoltage(voltage);

Write a specific voltage to the DAC and store it in the internal EEPROM as the startup/power-on value.

Parameters:	voltage: 0.00 to 5.00 (AVR microcontrollers) 0.00 to 3.30 (ARM and PIC32 microcontrollers)
Returns:	Nothing
Usage:	dac.storeVoltage(1.25f); // Set the DAC output to 1.25v and store that as the default power-on value
Notes:	In order for the output voltage to be exactly what you specify the operating voltage of the MCP4725 must be exactly 5.00v for AVR microcontrollers or 3.30v for ARM and PIC32 microcontrollers. *Any* deviation in the operating voltage of the MCP4725 will result in a subsequent deviation in the output voltage.

storePowerDown(value);

Set the power down mode of the MCP4725 and store it in the internal EEPROM as the startup/power-on value.

Parameters:	value: MCP4725_POWERDOWN_OFF, MCP4725_POWERDOWN_1K, MCP4725_POWERDOWN_100K or MCP4725_POWERDOWN_500K
Returns:	Nothing
Usage:	dac.storePowerDown(MCP4725_POWERDOWN_1K); // Enable power down and set it as the power-up default
Notes:	storePowerDown() will set the default output of the MCP4725 to 0. storeValue() and storeVoltage() will automatically set the default power down mode to MCP4725_POWERDOWN_OFF.