1300 Henley Court
Pullman, WA 99163
509.334.6306
www.digilentinc.com

chipKIT™ DACSPI1 Library Reference Manual

Revised April 8, 2014

Overview

The Digilent PmodDA1 has four simultaneous D/A conversion channels, each with an 8-bit converter that can process a separate digital signal. Digilent provides an Arduino driver library for this device that is able to access only one converter channel. This document provides an overview of the operation of this driver library and describes the functions that define its programming interface.

The PmodDA1 device contains two Analog Devices AD7303 8-bit D/A converter chips that convert up to one MSa per second, Serial Input, Dual Voltage Output 8-Bit D/A Converters (DACs), implementing four simultaneous D/A conversion channels. Due to the physical pin layout of the SPI hardware interfaces, this library implements access only to the converter channel IC1, corresponding to DAC A (pin 1) of the J2 connector of PmodDA1.

For more information about the hardware interface of the PmodDA1, refer to the PmodDA1 reference manual available for download from the Digilent website www.digilentinc.com.

The converter is a serial device that is accessed (written) via an SPI interface. The library allows the use of any of the two SPI hardware interfaces. The values written to the converter are on 12 bits (from 0 to 0x0FFF) which is considered "*IntegerValue*". The library also deals with "*PhysicalValue*", which is a floating point value that corresponds to the "*IntegerValue"* and to a reference value that defines the reference voltage (associated to the maximum integer value).

$$\text{PhysicalValue} = \text{IntegerValue} \cdot \left(\frac{\text{ReferenceValue}}{\left(2^{8-1}\right)} \right)$$

The physical value corresponds to the value that will be reconstructed (and can be measured) at the output of the D/A converter. By default, the reference value corresponds to a reference voltage of 3.3V.

Note: In order to use the "PhysicalValue" functionality you must have the following line placed at the top of the DACSPI1.h file:

- #define DA1_FLOATING_POINT

1 Library Operation

1.1 Library Interface

The header file DACSPI1.h defines the interfaces to the DACSPI1 driver. The library is accessed via the methods and constants defined for the DACSPI1 object class. In order to use this library, the user has to instantiate one library object.

1.2 SPI Initialization

In order to communicate with the PmodDA1, the SPI interface has to be initialized. Before calling any other library functions, the begin() function must be called. This function initializes the SPI port used by the library.

1.3 DA Converter Functions

In order to write values to the DA converter, the library provides two types of functions: WriteIntegerValue() and WritePhysicalValue() (see note below). Read the Overview section for a description of these values.

Note: In order to use the "PhysicalValue" functionality you must have the following line placed at the top of the DACSPI1.h file:

- #define DA1_FLOATING_POINT

2 DACSPI1 Library Functions

2.1 Errors

Table 1 shows the possible error messages returned by the DACSPI1 library functions:

Value	Name	Description
0	DACSPI1_ERR_SUCCESS	The action completed successfully
1	DACSPI1_ERR_VAL_OUT_OF_RANGE	The value is out of range

Table 1. List of errors.

void begin(uint8_t bAccessType)

Parameters:

- uint8_t bAccessType
  - The SPI interface where the PmodDA1 is connected. It can be one of the parameters from Table 2 below:

Value	Name	Connector on chipKIT Pro MX4
0	PAR_ACCESS_SPI0	JB
1	PAR_ACCESS_SPI1	J1

Table 2. List of values for bAccessType parameter of begin function.

This function initializes the selected SPI interface, setting the SPI frequency to 1 MHz.

uint8_t WriteIntegerValue(uint8_t bIntegerValue)

Parameters:

- uint8_t bIntegerValue
  - The 8-bit value to be written to DA converter.

Return value:

- DACSPI1_ERR_SUCCESS
  - The action completed successfully.
- DACSPI1_ERR_VAL_OUT_OF_RANGE
  - The value is not within the 0 - 0xFF range.

If the value is within the allowed range (0 - 0xFF) then this function writes the 8-bit value to the DA converter and returns the DACSPI1_ERR_SUCCESS status message. If the value is outside of the allowed range, then the function does nothing and returns the DACSPI1_ERR_VAL_OUT_OF_RANGE message.

uint8_t WritePhysicalValue(float dPhysicalValue, float dReference)

Parameters

- float dPhysicalValue
  - The physical value that must be reconstructed at the output of the DA converter.
- float dReference
  - The value corresponding to the maximum converted value (reference voltage). If this parameter is not provided, it has a default value of 3.3.

Return value:

- DACSPI1_ERR_SUCCESS
  - The action completed successfully.
- DACSPI1_ERR_VAL_OUT_OF_RANGE
  - The physical value is not within the accepted range.

The function computes the integer value corresponding to the physical value by considering the reference value as the one corresponding to the maximum integer value (0xFF). If the integer value is within the accepted range (0 - 0xFF), then this function writes the 8-bit value to the DA converter and returns the DACSPI1_ERR_SUCCESS message. If the integer value is outside of the allowed range, then the function does nothing and returns the DACSPI1_ERR_VAL_OUT_OF_RANGE message. If the dReference function argument is missing, a 3.3 value is used as the reference value.

Note: In order to use the "PhysicalValue" functionality, you must have the following line placed at the top of the DACSPI1.h file (this function doesn't even exist without it):

- #define DA1_FLOATING_POINT

3 Library Usage

This section of the document describes the way the library is used:

- The PmodDA1 should be plugged into one of the SPI connectors.

Connection	Connector on chipKIT Pro MX4
SPI0	JB
SPI1	J1

Table 3. List of possible connections for PmodAD1.

- The analog value is generated to pin 1 (DAC A) of the J2 connector of PmodDA1.
- Copy the library files according to the README.txt file.

- In the sketch, include the DACSPI1 library header file:
```
#include <DACSPI1.h>
```

- In the sketch, include the DSPI library header file. It is needed in order to access the SPI functionality:
```
#include <DSPI.h>
```

- In the sketch, instantiate one library object called, for example, myDACSPI1:
```
DACSPI1 myDACSPI1;
```

- In the sketch, use library functions by calls such as:
```
myDACSPI1.WriteIntegerValue(bValue);
```

4 Simple Demo

In order to run this demo, place the library and sketch and connect the wire corresponding to the analog value as explained in the Library usage section. This demo performs the following operations:

- In the setup() function:
  - Initializes the DACSPI1 library.
- In the loop() function:
  - Repeatedly increases and decreases the physical value to be reconstructed by the DA converter. (note: This demo does utilize floating point functionality via #define DA1_FLOATING_POINT)