MCP4725

1.0.0

Generated by Doxygen 1.9.1

Chapter 1

MCP4725_PICO

This documentation is for the application programming interface for project MCP4725_PICO. Library Driver for the MCP4725 DAC modules, for Raspberry pi PICO RP2040 C++ SDK.

The main project documentation is in a README.md file at the github repository URL :

`Github Project url`

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 MCP4725_PICO Class Reference

Class for MCP4725_PIC0 DAC.

```
#include <mcp4725.hpp>
```

Public Member Functions

- MCP4725_PICO (float refV=MCP4725_REFERENCE_VOLTAGE)

  Constructor for class MCP4725_PIC0.
- bool begin (MCP4725_I2C_Addr_e addr, i2c_inst_t ∗type, uint16_t speed, uint8_t SDA, uint8_t SCLK)

  Init & config i2c.
- bool isConnected ()

  Checks if DAC is connected.
- bool GeneralCall (MCP4725_GeneralCallType_e)

  General Call, name from datasheet section 7.3.
- void deinitI2C ()

  Switch off the I2C interface and return I2C GPIO to default state.
- void setReferenceVoltage (float value)

  Sets the reference voltage.
- float getReferenceVoltage (void)

  Gets the reference voltage.
- bool setInputCode (uint16_t inputCode, MCP4725_CmdType_e=MCP4725_FastMode, MCP4725_PowerDownType_e=MCP47

  Set voltage out based on DAC input code.
- uint16_t getInputCode (void)

  get current DAC InputCode from DAC register
- bool setVoltage (float voltage, MCP4725_CmdType_e=MCP4725_FastMode, MCP4725_PowerDownType_e=MCP4725_Power

  Set voltage out based on voltage input in volts.
- float getVoltage (void)

  get DAC inputCode from DAC register & convert to volts
- uint16_t getStoredInputCode (void)

  Read DAC inputCode from EEPROM.
- float getStoredVoltage (void)

  Read stored DAC InputCode from EEPROM & convert to voltage.

- uint16_t getPowerType (void)

    Get current power type from DAC register.
- uint16_t getStoredPowerType (void)

    Get stored power type from EEPROM.
- void setSerialDebugFlag (bool onOff)

    Setter for serial debug flag.
- bool getSerialDebugFlag (void)

    Gets the serial Debug flag value.
- void setSafetyCheckFlag (bool onOff)

    Setter for safety Check flag.
- bool getSafetyCheckFlag (void)

    Gets the safety Check flag value.

4.1.1 Detailed Description

Class for MCP4725_PIC0 DAC.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 MCP4725_PICO()

```
MCP4725_PICO::MCP4725_PICO (
             float refV = MCP4725_REFERENCE_VOLTAGE )
```

Constructor for class MCP4725_PIC0.

Parameters

refV	The the reference voltage to be set in Volts.

4.1.3 Member Function Documentation

4.1.3.1 begin()

```
bool MCP4725_PICO::begin (
             MCP4725_I2C_Addr_e addr,
             i2c_inst_t * i2c_type,
             uint16_t CLKspeed,
             uint8_t SDApin,
             uint8_t SCLKpin )
```

Init & config i2c.

Parameters

addr	I2C address 8 bit address 0x6?.
i2c_type	I2C instance of port, IC20 or I2C1.
CLKspeed	I2C Bus Clock speed in Kbit/s. see 7.1 datasheet
SDApin	I2C Data GPIO
SCLKpin	I2C Clock GPIO

Returns

true if success , false for failure

4.1.3.2 GeneralCall()

```
bool MCP4725_PICO::GeneralCall (
            MCP4725_GeneralCallType_e typeCall )
```

General Call, name from datasheet section 7.3.

Parameters

typeCall	Reset or wakeup see MCP4725_GeneralCallType_e.

Returns

True on success, false on I2c error OR wrong input(GeneralCallAddress)

Note

1. Reset MCP4725 & upload data from EEPROM to DAC register. Immediately after reset event, uploads contents of EEPROM into the DAC reg.
2. Wake up & upload value from DAC register, Current power-down bits are set to normal, EEPROM power-down bit are not affected

4.1.3.3 getInputCode()

```
uint16_t MCP4725_PICO::getInputCode (
            void )
```

get current DAC InputCode from DAC register

Returns

DAC InputCode :or 0xFFFF if I2C error

4.1.3.4 getPowerType()

```
uint16_t MCP4725_PICO::getPowerType (
             void  )
```

Get current power type from DAC register.

Returns

power type or 0xFFFF if I2C error

Note

Power type corresponds to enum MCP4725_PowerDownType_e

4.1.3.5 getReferenceVoltage()

```
float MCP4725_PICO::getReferenceVoltage (
             void  )
```

Gets the reference voltage.

Returns

The reference voltage in volts.

4.1.3.6 getSafetyCheckFlag()

```
bool MCP4725_PICO::getSafetyCheckFlag (
             void  )
```

Gets the safety Check flag value.

Returns

The safety Check flag value

4.1.3.7 getSerialDebugFlag()

```
bool MCP4725_PICO::getSerialDebugFlag (
             void  )
```

Gets the serial Debug flag value.

Returns

The serial Debug flag value

4.1.3.8 getStoredInputCode()

```
uint16_t MCP4725_PICO::getStoredInputCode (
            void )
```

Read DAC inputCode from EEPROM.

Returns

stored EEPROM inputcode value or 0xFFFF if I2C error

4.1.3.9 getStoredPowerType()

```
uint16_t MCP4725_PICO::getStoredPowerType (
            void )
```

Get stored power type from EEPROM.

Returns

EEPROM power type or 0xFFFF if I2C error

Note

Power type corresponds to enum MCP4725_PowerDownType_e

4.1.3.10 getStoredVoltage()

```
float MCP4725_PICO::getStoredVoltage (
            void )
```

Read stored DAC InputCode from EEPROM & convert to voltage.

Returns

stored EEPROM voltage or 0xFFFF if I2C error

4.1.3.11 getVoltage()

```
float MCP4725_PICO::getVoltage (
            void )
```

get DAC inputCode from DAC register & convert to volts

Returns

DAC voltage or 0xFFFF if I2C error

4.1.3.12 isConnected()

```
bool MCP4725_PICO::isConnected ( )
```

Checks if DAC is connected.

Returns

true if DAC is connected , false if not

4.1.3.13 setInputCode()

```
bool MCP4725_PICO::setInputCode (
            uint16_t InputCode,
            MCP4725_CmdType_e mode = MCP4725_FastMode,
            MCP4725_PowerDownType_e powerType = MCP4725_PowerDown_Off )
```

Set voltage out based on DAC input code.

Parameters

InputCode	0 to MCP4725_MAX_VALUE.
mode	MCP4725DAC mode, see enum MCP4725_CmdType_e.
powerType	MCP4725DAC power type, see enum MCP4725_PowerType_e

Returns

output of writeCommand method, true for success, false for failure.

4.1.3.14 setReferenceVoltage()

```
void MCP4725_PICO::setReferenceVoltage (
            float voltage )
```

Sets the reference voltage.

Parameters

voltage	the reference voltage to be set, called from constructor.

4.1.3.15 setSafetyCheckFlag()

```
void MCP4725_PICO::setSafetyCheckFlag (
            bool onOff )
```

Setter for safety Check flag.

Parameters

onOff	Turns or or off the safety check flag

4.1.3.16 setSerialDebugFlag()

```
void MCP4725_PICO::setSerialDebugFlag (
            bool onOff )
```

Setter for serial debug flag.

Parameters

onOff	Turns or or off the serial debug flag

4.1.3.17 setVoltage()

```
bool MCP4725_PICO::setVoltage (
            float voltage,
            MCP4725_CmdType_e mode = MCP4725_FastMode,
            MCP4725_PowerDownType_e powerType = MCP4725_PowerDown_Off )
```

Set voltage out based on voltage input in volts.

Parameters

voltage	0 to_MCP4725_REFERENCE_VOLTAGE, voltage out
mode	MCP4725DAC mode, see enum MCP4725_CmdType_e.
powerType	MCP4725DAC power type, see enum MCP4725_PowerType_e

Returns

output of writeCommand method, true for success, false for failure.

The documentation for this class was generated from the following files:

- mcp4725.hpp
- mcp4725.cpp

Chapter 5

File Documentation

5.1 main.cpp File Reference

Example cpp file for MCP4725 DAC library , demoTest.

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "mcp4725/mcp4725.hpp"
```

Macros

- #define **DAC_REF_VOLTAGE** 3.3

Functions

- void **printDACSettings** (void)
- int main ()

  The MAIN loop function, demoTest example file.

Variables

- MCP4725_PICO **myDAC** (DAC_REF_VOLTAGE)
- uint16_t **TestCount** = 0

5.1.1 Detailed Description

Example cpp file for MCP4725 DAC library , demoTest.

Author

Gavin Lyons

Note

This file carries our a series of tests to demonstrate features. Such as Power down reset wakeup and EEPROM data save. Output to Serial monitor and multimeter on Vout of DAC.

1. Test 0 print settings.
2. Test 1 set voltage to Vmax with power ON mode , Vout = 3.3 V.
3. Test 2 set voltage to 1.65 with power off mode , Vout = 0.0 V.
4. Test 3 set voltage to 900mV with power off mode , Vout = 0.0 V.
5. Test 4 wake up device, Vout = 900m V.
6. Test 5a set voltages to Vmax with power ON mode, Vout = 3.3 V.
7. Test 5b print settings.
8. Test 6a Set Voltage and power mode on in the EEPROM, Vout = 1.0 V.
9. Test 6b set voltage to Vmax with power ON mode Vout = 3.3 V.
10. Test 6c Reset device & Vout will revert EEPROM in 6a, Vout = 1.0 V.
11. Test 7 Clear the EEPROM, Vout = 0.0 V.
12. Test 7b print settings.

5.1.2 Function Documentation

5.1.2.1 main()

```
int main ( )
```

The MAIN loop function, demoTest example file.

Returns

Program Exit code

5.2 main.cpp File Reference

MCP4725 DAC library example file. IsConnected.

```
#include "mcp4725/mcp4725.hpp"
```

Macros

- #define **DAC_REF_VOLTAGE** 3.3

Functions

- int main ()

  The MAIN loop function, isConnected example file.

Variables

- MCP4725_PICO **myDAC** (DAC_REF_VOLTAGE)
- uint16_t **TestCount** = 0

5.2.1 Detailed Description

MCP4725 DAC library example file. IsConnected.

Author

Gavin Lyons

Note

This file carries out Connection test to see if DAC on the I2C bus. Output to Serial monitor 38400 baud rate

5.2.2 Function Documentation

5.2.2.1 main()

```
int main ( )
```

The MAIN loop function, isConnected example file.

Returns

Program Exit code

5.3 main.cpp File Reference

Example cpp file for MCP4725 DAC library, random.

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "mcp4725/mcp4725.hpp"
```

Macros

- #define **DAC_REF_VOLTAGE** 3.3

Functions

- int main ()

  The MAIN loop function, random example file.

Variables

- MCP4725_PICO **myDAC** (DAC_REF_VOLTAGE)

5.3.1 Detailed Description

Example cpp file for MCP4725 DAC library, random.

Author

Gavin Lyons

Note

Generate random data for MCP4725.

5.3.2 Function Documentation

5.3.2.1 main()

```
int main ( )
```

The MAIN loop function, random example file.

Returns

Program Exit code

5.4 main.cpp File Reference

Example cpp file for MCP4725 DAC library, sawToothWave.

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "mcp4725/mcp4725.hpp"
```

Macros

- #define **DAC_REF_VOLTAGE** 3.3

Functions

- int main ()
  The MAIN loop function, sawTooth example file.

Variables

- MCP4725_PICO **myDAC** (DAC_REF_VOLTAGE)
- int16_t **smoothing** = 50
- int16_t **counter** = 0

5.4.1 Detailed Description

Example cpp file for MCP4725 DAC library, sawToothWave.

Author

Gavin Lyons

Note

Generated a sawtooth waveform for MCP4725 146 Hz.

5.4.2 Function Documentation

5.4.2.1 main()

```
int main ( )
```

The MAIN loop function, sawTooth example file.

Returns

Program Exit code

5.5 main.cpp File Reference

Example cpp file for MCP4725 DAC library, SetVoltage.

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "mcp4725/mcp4725.hpp"
```

Macros

- #define **DAC_REF_VOLTAGE** 3.3

Functions

- int main ()

  The MAIN loop function, setVoltage example file.

Variables

- MCP4725_PICO **myDAC** (DAC_REF_VOLTAGE)

5.5.1 Detailed Description

Example cpp file for MCP4725 DAC library, SetVoltage.

Author

Gavin Lyons

Note

This file carries out some set voltage tests.

1. Test 1 setInputCode function 4096 vout = 3.3V.
2. Test 2 setInputCode function 2048 Vout = 1.65V.
3. Test 3 setVoltage function 2.0 , vout 2.0V.
4. Test 4 setVoltage function 0.800 , vout = 800mV.

Example input code calculation . PICO Vref = 3.3 , MCP4725A0 , resolution = 2^12 = 4096. Note Vout = (Vref X input code) / Resolution . Vout = (3.3 X inputcode)/ 4096. eg for input code 2048 -> Vout = (3.3 X 2048) /4096 = 1.65 V.

5.5.2 Function Documentation

5.5.2.1 main()

```
int main ( )
```

The MAIN loop function, setVoltage example file.

Returns

Program Exit code

5.6 main.cpp File Reference

Example cpp file for MCP4725 DAC library, SineWave.

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "mcp4725/mcp4725.hpp"
#include "mcp4725/mcp4725_Sinewave_Data.hpp"
```

Macros

- #define **DAC_REF_VOLTAGE** 3.3

Functions

- int main ()

    The MAIN loop function, sineWave example file.

Variables

- MCP4725_PICO **myDAC** (DAC_REF_VOLTAGE)

5.6.1 Detailed Description

Example cpp file for MCP4725 DAC library, SineWave.

Author

Gavin Lyons

Note

Generate a Sine waveform for MCP4725. The data to create Sine wave is the MCP4725 Sinewave_Data.↩
cpp file. In the MCP4725_Sinewave_Data.hpp user can pick Resolution "SINEWAVE_RES". Resolution of Sinewave in bits (2^bits). Select 512, 256, 128, 64 or 32 points table. i.e. 2^9 = 512 , 9 bit resolution etc. default 128 100Hz.

5.6.2 Function Documentation

5.6.2.1 main()

```
int main ( )
```

The MAIN loop function, sineWave example file.

Returns

Program Exit code

5.7 main.cpp File Reference

Example cpp file for MCP4725 DAC library, SquareWave.

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "mcp4725/mcp4725.hpp"
```

Macros

- #define **DAC_REF_VOLTAGE** 3.3

Functions

- int main ()

    The MAIN loop function, squareWave example file.

Variables

- MCP4725_PICO **myDAC** (DAC_REF_VOLTAGE)
- int **TestCount** = 0
- int **Period** = 10

5.7.1 Detailed Description

Example cpp file for MCP4725 DAC library, SquareWave.

Author

Gavin Lyons

Note

Generate a square wave 100Hz.

5.7.2 Function Documentation

5.7.2.1 main()

```
int main ( )
```

The MAIN loop function, squareWave example file.

Returns

Program Exit code

5.8 main.cpp File Reference

Example cpp file for MCP4725 DAC library, triangleWave.

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "mcp4725/mcp4725.hpp"
```

Macros

- #define **DAC_REF_VOLTAGE** 3.3

Functions

- int main ()

  The MAIN loop function, squareWave example file.

Variables

- MCP4725_PICO **myDAC** (DAC_REF_VOLTAGE)
- int16_t **smoothing** = 50
- int16_t **counter** = 0

5.8.1 Detailed Description

Example cpp file for MCP4725 DAC library, triangleWave.

Author

Gavin Lyons

Note

Generated a triangle waveform for MCP4725 75 Hz.

5.8.2 Function Documentation

5.8.2.1 main()

```
int main ( )
```

The MAIN loop function, squareWave example file.

Returns

Program Exit code

5.9 mcp4725.cpp File Reference

MCP4725 DAC library cpp file.

```
#include "../include/mcp4725/mcp4725.hpp"
```

5.9.1 Detailed Description

MCP4725 DAC library cpp file.

Author

Gavin Lyons

5.10 mcp4725.hpp File Reference

Library header file for MCP4725 PICO DAC library.

```
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/i2c.h"
#include <cmath>
```

Classes

- class MCP4725_PICO

  Class for MCP4725_PIC0 DAC.

Macros

- #define MCP4725_I2C_DELAY 50000
- #define MCP4725_ERROR 0xFFFF
- #define MCP4725_EEPROM_WRITE_TIME 25
- #define MCP4725_REFERENCE_VOLTAGE 3.3
- #define MCP4725_RESOLUTION 12
- #define MCP4725_STEPS pow(2, (MCP4725_RESOLUTION))
- #define MCP4725_MAX_VALUE ((MCP4725_STEPS) - 1)

Enumerations

- enum MCP4725_I2C_Addr_e : uint8_t {
  MCP4725A0_Addr_A00 = 0x60 , MCP4725A0_Addr_A01 = 0x61 , MCP4725A1_Addr_A00 = 0x62 , MCP4725A1_Addr_A01 = 0x63 ,
  MCP4725A2_Addr_A00 = 0x64 , MCP4725A2_Addr_A01 = 0x65 }
- enum MCP4725_CmdType_e : uint8_t { MCP4725_FastMode = 0x00 , MCP4725_RegisterMode = 0x40 , MCP4725_EEPROM_Mode = 0x60 }
- enum MCP4725_PowerDownType_e : uint8_t { MCP4725_PowerDown_Off = 0x00 , MCP4725_PowerDown_1kOhm = 0x01 , MCP4725_PowerDown_100kOhm = 0x02 , MCP4725_PowerDown_500kOhm = 0x03 }
- enum MCP4725_ReadType_e : uint8_t { MCP4725_ReadSettings = 1 , MCP4725_ReadDACReg = 3 , MCP4725_ReadEEPROM = 5 }
- enum MCP4725_GeneralCallType_e : uint8_t { MCP4725_GeneralCallAddress = 0x00 , MCP4725_GeneralCallReset = 0x06 , MCP4725_GeneralCallWakeUp = 0x09 }

5.10.1 Detailed Description

Library header file for MCP4725 PICO DAC library.

Author

Gavin Lyons

Note

See URL for full details. https://github.com/gavinlyonsrepo/MCP4725_PICO

5.10.2 Macro Definition Documentation

5.10.2.1 MCP4725_EEPROM_WRITE_TIME

```
#define MCP4725_EEPROM_WRITE_TIME 25
```

mSec Memory write time, maximum 50 mSec

5.10.2.2 MCP4725_ERROR

```
#define MCP4725_ERROR 0xFFFF
```

returns this value if I2C bus error from some methods

5.10.2.3 MCP4725_I2C_DELAY

```
#define MCP4725_I2C_DELAY 50000
```

uS delay , I2C timeout

5.10.2.4 MCP4725_MAX_VALUE

```
#define MCP4725_MAX_VALUE ((MCP4725_STEPS) - 1)
```

Max value = 4096 -1 , 0 to 4095

5.10.2.5 MCP4725_REFERENCE_VOLTAGE

```
#define MCP4725_REFERENCE_VOLTAGE 3.3
```

supply-reference Voltage in volts

5.10.2.6 MCP4725_RESOLUTION

```
#define MCP4725_RESOLUTION 12
```

resolution in bits , 12-bit

5.10.2.7 MCP4725_STEPS

```
#define MCP4725_STEPS pow(2, (MCP4725_RESOLUTION))
```

quantity of DAC steps 2^12-bits = 4096

5.10.3 Enumeration Type Documentation

5.10.3.1 MCP4725_CmdType_e

```
enum MCP4725_CmdType_e : uint8_t
```

DAC register, command bits C2C1C0

Enumerator

MCP4725_FastMode	Writes data to DAC register
MCP4725_RegisterMode	Writes data & config bits to DAC register
MCP4725_EEPROM_Mode	Writes data & config bits to DAC register & EEPROM

5.10.3.2 MCP4725_GeneralCallType_e

```
enum MCP4725_GeneralCallType_e : uint8_t
```

DAC general call command datasheet 7.3

Enumerator

MCP4725_GeneralCallAddress	General call address
MCP4725_GeneralCallReset	General call reset command
MCP4725_GeneralCallWakeUp	General call wake-up command

5.10.3.3 MCP4725_I2C_Addr_e

```
enum MCP4725_I2C_Addr_e : uint8_t
```

8-bit i2c address.

Enumerator

MCP4725A0_Addr_A00	MCP4725A0 with A0 = GND
MCP4725A0_Addr_A01	MCP4725A0 with A0 = VCC
MCP4725A1_Addr_A00	MCP4725A1 with A0 = GND
MCP4725A1_Addr_A01	MCP4725A1 with A0 = VCC
MCP4725A2_Addr_A00	MCP4725A2 with A0 = GND
MCP4725A2_Addr_A01	MCP4725A2 with A0 = VCC

5.10.3.4 MCP4725_PowerDownType_e

```
enum MCP4725_PowerDownType_e : uint8_t
```

DAC register, power down bits PD1 PD0 , BSY,POR,xx,xx,xx,PD1,PD0,xx

Enumerator

MCP4725_PowerDown_Off	Power down off draws 0.40mA no load & 0.29mA max load
MCP4725_PowerDown_1kOhm	Power down on, with 1.0 kOhm to GND, draws ∼60nA
MCP4725_PowerDown_100kOhm	Power down on, with 100 kOhm to GND
MCP4725_PowerDown_500kOhm	Power down on, with 500 kOhm to GND

5.10.3.5 MCP4725_ReadType_e

```
enum MCP4725_ReadType_e : uint8_t
```

DAC library read register type

Enumerator

MCP4725_ReadSettings	Read 1 byte, Settings data
MCP4725_ReadDACReg	Read 3 bytes, DAC register data
MCP4725_ReadEEPROM	Read 5 bytes, EEPROM data

5.11 mcp4725_Sinewave_Data.cpp File Reference

Data file for MCP4725 Sinewave generation.

```
#include "../include/mcp4725/mcp4725_Sinewave_Data.hpp"
```

Variables

- const uint16_t ∗ pDacLookupSineWave = DACLookupSineWave

5.11.1 Detailed Description

Data file for MCP4725 Sinewave generation.

Author

Gavin Lyons

5.11.2 Variable Documentation

5.11.2.1 pDacLookupSineWave

```
const uint16_t* pDacLookupSineWave = DACLookupSineWave
```

Pointer to data which is in cpp file

5.12 mcp4725_Sinewave_Data.hpp File Reference

Data header file for MCP4725 Sinewave generation.

```
#include <stdint.h>
```

Macros

- #define SINEWAVE_RES 128

Variables

- const uint16_t ∗ pDacLookupSineWave

5.12.1 Detailed Description

Data header file for MCP4725 Sinewave generation.

Author

Gavin Lyons

5.12.2 Macro Definition Documentation

5.12.2.1 SINEWAVE_RES

```
#define SINEWAVE_RES 128
```

Resolution of Sinewave in bits (2^bits) Select 512, 256, 128, 64 or 32 points table i.e. 2^9 = 512 , 9 bit resolution etc

5.12.3 Variable Documentation

5.12.3.1 pDacLookupSineWave

```
const uint16_t* pDacLookupSineWave  [extern]
```

Pointer to data which is in cpp file

Index