



PXDAC4800 MatLAB Library Reference

Version 1.21

Vitrek LLC
900 N. State Street
Lockport, IL 60441 USA
Tel: (800) 567 - 4243
Fax: (800) 780 - 8411

<http://www.signatec.com>

June 29, 2022

TABLE OF CONTENTS

1	GETTING STARTED	1
	Overview	1
	Installation of PXDAC4800 MatLAB Software.....	1
	First Run	1
	Setup a MatLAB compiler.....	2
2	PXDAC4800 MATLAB SDK	2
	Essential header	2
	API functions.....	3
	ConnectToDeviceXD48	4
	SetPowerupDefaultsXD48	4
	SetPlaybackClockSourceXD48	4
	SetClockDivider1	4
	SetClockDivider2	4
	SetTriggerModeXD48	4
	SetDacSampleFormatXD48	5
	SetDacSampleSizeXD48	5
	SetActiveChannelMaskXD48	5
	LoadRamBufXD48.....	5
	BeginRamPlaybackXD48	6
	SessionStreamCreateParmsAXD48	6
	SessionStreamEndNoStructXD48.....	6
	SessionStreamDeleteNoStructXD48	7
	DisconnectXD48	7
	GetErrorMessXD48.....	7
	GetUserDataXD48	7
	Sample programs.....	7
	PXDAC4800_Config.m	7
	PXDAC4800_Playback_Buffer.m.....	8
	PXDAC4800_Playback_File.m.....	8
	PXDAC4800_Streaming.m	9
	Notes on 64-bit MATLAB SDK	9
3	APPENDIX 1 – REVISION HISTORY	11

TABLE OF FIGURES

Figure 1: Setup a MatLAB compiler	2
Figure 2: Essential header	3

1 GETTING STARTED

Overview

This manual serves as an aid to engineers using the PXDAC4800, a high-performance data generation card, in the MATLAB 2010b+ for Windows environment.

The PXDAC4800 SDK for MATLAB for Windows supports the PXDAC4800 only.

This manual assumes that you are familiar with the MATLAB programming environment. If you do not feel comfortable with MATLAB, it is highly recommended that you go through the Getting Started with MATLAB® manual supplied by The MathWorks, Inc. before starting any program development for the Signatec PXDAC4800 card.

It is also assumed that you are familiar with PCs and Microsoft Windows and that you have correctly installed the Signatec PXDAC4800 driver.

Please note that this manual is not intended as a reference for any software other than the Signatec PXDAC4800 SDKs for MATLAB for Windows. If you did not receive the correct guide, please contact the factory for a replacement.

Please note that our 32-bit MATLAB SDK uses calllib to make calls directly to the PXDAC4800.dll, while our 64-bit MATLAB SDK makes the same calls to a wrapper library named pxdac4800_wrap.dll.

To maintain the accuracy of the information contained herein, we reserve the right to make changes to this manual from time to time.

Installation of PXDAC4800 MatLAB Software

Installation of the PXDAC4800 MATLAB software will be done with the regular PXDAC4800 installation.

By default, the PXDAC4800 MATLAB SDK will install itself in: O/S system drive:\Program Files\Signatec\ PXDAC4800 \Examples\MATLAB. It is recommended that you use the default installation location.

If you require more detailed installation instructions, please refer to the Signatec PXDAC4800 readme, which was shipped with your order.

First Run

The first thing you need to do, is to open MATLAB and select the current folder (InstallationPath/Examples/MATLAB). Then view the SDK (.m) files in the folder panel.

Then open “PXDAC4800_Config.m” and run it. If you don’t have any compiler setup in your MATLAB, it will give you an error. If it’s the case, follow the [procedure](#) “Setup a MATLAB compiler”.

Setup a MatLAB compiler

```
>> mex -setup
Please choose your compiler for building external interface (MEX) files:

Would you like mex to locate installed compilers [y]/n? y

Select a compiler:
[1] Lcc-win32 C 2.4.1 in D:\PROGRA~1\MATLAB\R2010B~1\sys\lcc
[2] Microsoft Visual C++ 2010 in D:\Program Files (x86)\Microsoft Visual Studio 10.0

[0] None

Compiler: 1

Please verify your choices:

Compiler: Lcc-win32 C 2.4.1
Location: D:\PROGRA~1\MATLAB\R2010B~1\sys\lcc

Are these correct [y]/n? y

Trying to update options file: C:\Users\GATI\AppData\Roaming\MathWorks\MATLAB\R2010b\mexopts.bat
From template:          D:\PROGRA~1\MATLAB\R2010B~1\bin\win32\mexopts\lccopts.bat

Done . . .

*****
Warning: The MATLAB C and Fortran API has changed to support MATLAB
variables with more than 2^32-1 elements. In the near future
you will be required to update your code to utilize the new
API. You can find more information about this at:
http://www.mathworks.com/support/solutions/en/data/1-5C27B9/?solution=1-5C27B9
Building with the -largeArrayDims option enables the new API.
*****

fx >>
```

Figure 1: Setup a MATLAB compiler

2 PXDAC4800 MATLAB SDK

Essential header

If you want to write your own code, the only thing you need before getting started is to add the proper paths and copy the “essential header” to the top of your .m file. This header loads the library, creates the pointer to the handle, connects you to the card and gives you the handle of the card. If you have more than one card, you need to do an “Init Handle”, “Connect to device” (don’t forget to increment the card number) and “Get the handle pointer” for each of them.

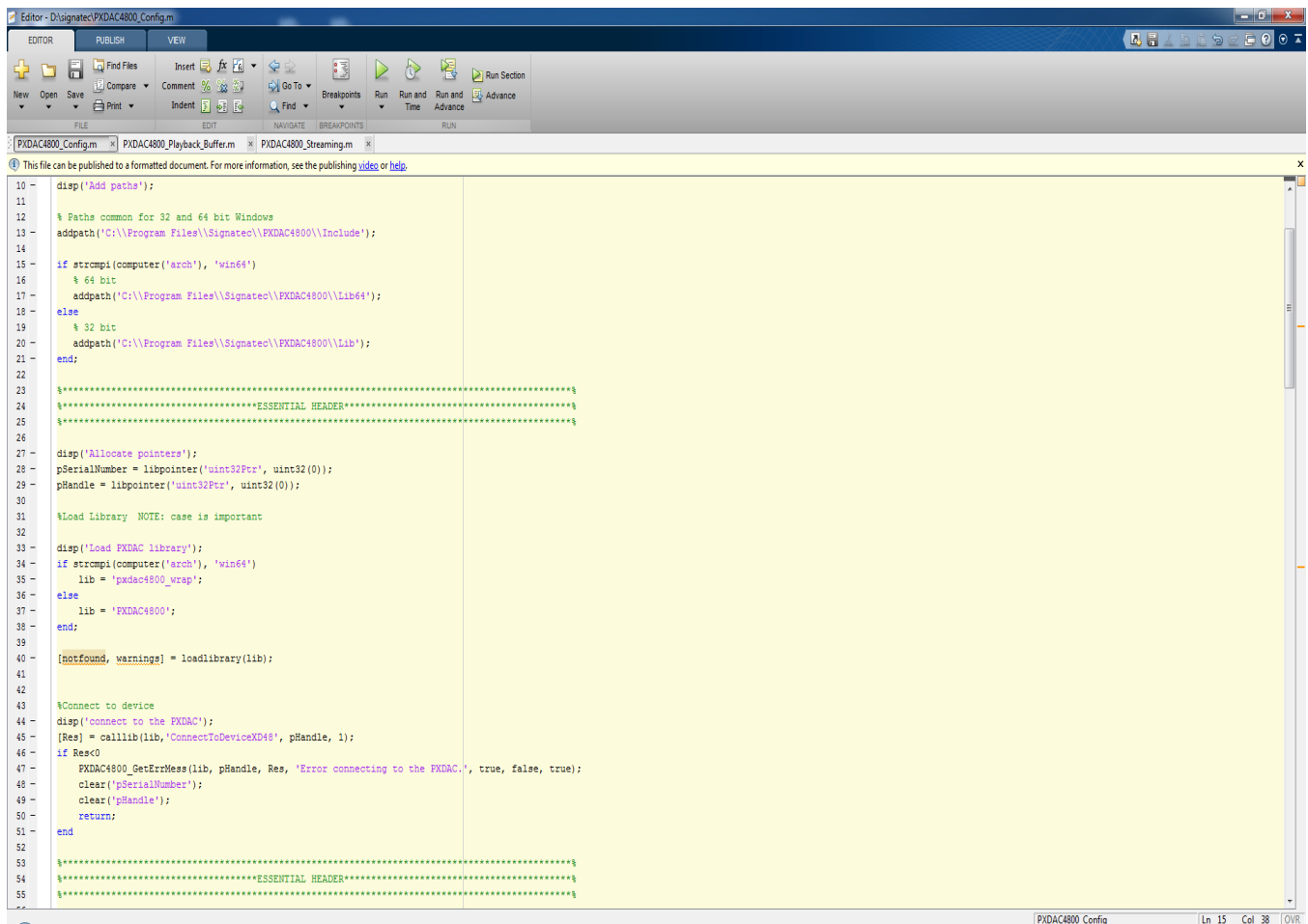


Figure 2: Essential header

Examples of doing this can be found in any of our sample m files. Then you can call the API functions just like in C. See the section below.

API functions

Using the MATLAB function `calllib` is similar to calling a function in C/C++.

Your first parameter is always the name of the library, (PXDAC4800) and the second is the name of the function you want. Then you put all the parameters of the function you want to call, in order.

To see all the functions available, you can use the MATLAB function `libfunctionsview` i.e. `libfunctionsview PXDAC4800` (or `libfunctionsview pxdac4800_wrap`)

If you have any questions about all the parameters a function needs, please refer to the PXDAC4800 Operators Manual, which was shipped with your order.

Here are some examples making an API call in MATLAB:

ConnectToDeviceXD48

C/C++:

```
outConnect = ConnectToDeviceXD48(&hBrd, 1);
```

MatLAB:

```
outConnect = calllib('PXDAC4800','ConnectToDeviceXD48', ppHandle, 1);
```

SetPowerupDefaultsXD48

C/C++:

```
outSetDefaults = SetPowerupDefaultsXD48(hBrd1);
```

MatLAB:

```
outSetDefaults = calllib('PXDAC4800','SetPowerupDefaultsXD48', pHandle, 1);
```

SetPlaybackClockSourceXD48

C/C++:

```
outSetClockSource = SetPlaybackClockSourceXD48(hBrd, 0);
```

MatLAB:

```
outSetClockSource = calllib(lib,'SetPlaybackClockSourceXD48', pHandle, 0);
```

SetClockDivider1

C/C++:

```
outSetClockDivider1 = SetClockDivider1XD48(hBrd, 1);
```

MatLAB:

```
outSetClockDivider1 = calllib(lib, 'SetClockDivider1XD48', pHandle, 1);
```

SetClockDivider2

C/C++:

```
outSetClockDivider2 = SetClockDivider2XD48(hBrd, 1);
```

MatLAB:

```
outSetClockDivider2, pHandle = calllib(lib,'SetClockDivider2XD48', pHandle, 1);
```

SetTriggerModeXD48

C/C++:

```
#define XD48TRIGMODE_CONTINUOUS      2
outSetTrigMode = SetTriggerModeXD48 (hBrd, 2);
```

MatLAB:

```
outSetTrigMode = calllib(lib,'SetTriggerModeXD48', pHandle, 2);
```

SetDacSampleFormatXD48

C/C++:

```
#define XD48SAMPFMT_UNSIGNED          0
outSetDacFormat = SetDacSampleFormatXD48(hBrd, XD48SAMPFMT_UNSIGNED);
```

MatLAB:

```
outSetDacFormat = calllib(lib, 'SetDacSampleFormatXD48', pHandle, 0);
```

SetDacSampleSizeXD48

C/C++:

```
#define XD48SAMPSIZE_14BIT_LSBPAD      2
outSetDacSize = SetDacSampleSizeXD48 (hBrd, XD48SAMPSIZE_14BIT_LSBPAD);
```

MatLAB:

```
outSetDacSize = calllib(lib, 'SetDacSampleSizeXD48', pHandle, 2);
```

SetActiveChannelMaskXD48

C/C++:

```
outSetActiveChannel = SetActiveChannelMaskXD48(hBrd, 1);
```

MatLAB:

```
[outSetActiveChannel, pHandle] = calllib(lib, 'SetActiveChannelMaskXD48', pHandle, 1);
```

LoadRamBufXD48

C/C++:

```
outLoadRamBuffer = LoadRamBufXD48(hBrd,
                                     0,
                                     upload_samples * sizeof(unsigned short),
                                     bufp,
                                     0);
```

MatLAB:

```
outLoadRamBuffer = calllib(lib, 'LoadRamBufXD48', pHandle, offset, pattern_size, pBuffer, 0);
```


BeginRamPlaybackXD48

C/C++:

```
outBeginRamPlayback = BeginRamPlaybackXD48(hBrd, 0, playback_bytes, 0);
```

MatLAB:

```
outBeginRamPlayback = calllib(lib, 'BeginRamPlaybackXD48', pHandle, offset, pattern_size, 0);
```

SessionStreamCreateParmsAXD48

C/C++:

```
outSessionStreamCreate = SessionStreamCreateParmsAXD48 ( hBrd
                                                         , FilePath
                                                         , Flags
                                                         , StreamBytes
                                                         , StreamSamples
                                                         , SrcOffsetBytes
                                                         , SrcLenBytes
                                                         , XferBytes
                                                         , SnapShotLenBytes
                                                         , SnapShotPeriodXfer
                                                         , SnapShotPeriodMs
                                                         , pSession);
```

MatLAB:

```
pSession = libpointer('uint32Ptr', uint32(0));
outSessionStreamCreate = calllib(lib, 'SessionStreamCreateParmsAXD48'
                                  , pHandle
                                  , FilePath
                                  , Flags
                                  , StreamBytes
                                  , StreamSamples
                                  , SrcOffsetBytes
                                  , SrcLenBytes
                                  , XferBytes
                                  , SnapShotLenBytes
                                  , SnapShotPeriodXfer
                                  , SnapShotPeriodMs
                                  , pSession);
```

SessionStreamEndNoStructXD48

C/C++:

```
outSessionStreamEnd = SessionStreamEndNoStructXD48 (pSession, 0);
```

MatLAB:

```
outSessionStreamEnd = calllib(lib, 'SessionStreamEndNoStructXD48', pSession, 0);
```

SessionStreamDeleteNoStructXD48

C/C++:

```
outSessionStreamDelete = SessionStreamDeleteNoStructXD48 (pSession);
```

MatLAB:

```
outSessionStreamDelete = calllib(lib,'SessionStreamDeleteNoStructXD48', pSession);
```

DisconnectXD48

C/C++:

```
outDisconnect = DisconnectFromDeviceXD48(hBrd);
```

MatLAB:

```
outDisconnect = calllib(lib,'DisconnectFromDeviceXD48', pHandle);
```

GetErrorMessXD48

C/C++:

```
TCHAR* bufp;  
res = GetErrorMessXD48(errCode, bufp, hBrd, 1);
```

MatLAB:

```
pErrMess = libpointer('cstring', char(zeros(1, 1024)));  
res = calllib(lib,'GetErrorMessXD48', ErrCode, pErrMess, pHandle, 1);
```

GetUserDataXD48

C/C++:

```
void** pUserData;  
res = GetErrorMessXD48(hBrd, pUserData);
```

MatLAB:

```
pUserData = libpointer('int32PtrPtr');  
res = calllib(lib,'GetUserDataXD48', pHandle, pUserData);  
userData = pUserData.Value;  
setdatatype(userData, 'int32Ptr', 1, 1);
```

Sample programs

PXDAC4800_Config.m

This sample program is an example of how to configure the card. This program will do the following:

- Load the library
- Init the handle
- Connect to device
- Get the handle pointer
- Get the serial number
- Set Trigger mode to Continuous
- Set the clock to 1.2 GHz
- Set the clock divider #1 to 1
- Set the clock divider #2 to 1
- Set the DAC format to unsigned
- Set the DAC sample to 14 bit LSB
- Set active channel to Single channel - channel 1
- Disconnect the device
- Unload the library

PXDAC4800_Playback_Buffer.m

This sample program is an example of how to create a buffer and playback it to the card. This program will do the following:

- Load the library
- Init the handle
- Connect to device
- Get the handle pointer
- Get the serial number
- Create a buffer (sinus)
- Setup a pointer to the buffer and put the buffer in a pointer
- Load the buffer
- Begin the playback
- Issue a software trigger
- End the playback
- Unload the library

PXDAC4800_Playback_File.m

This sample program is an example of how to playback from a file. This program will do the following:

- Load the library
- Init the handle
- Connect to device
- Get the handle pointer
- Get the serial number
- Set offset

- Set file size
- Set the file path
- Load the file to the PXDAC4800
- Begin playback
- Issue a software trigger
- End the playback
- Unload the library

PXDAC4800_Streaming.m

This sample program is an example of how to stream a data file. This program will do the following:

- Load the library
- Generate the data file to stream
- Initialize the handle
- Connect to device
- Get the serial number
- Set power up defaults
- Set operation mode to Stand-By
- Set trigger mode to Single-Shot
- Set clock source to 1200MHz
- Set clock divider #1 to 12
- Set clock divider #2 to 2
- Set DAC sample format to Signed
- Set DAC sample size to 16 bit LSB pad
- Set active channel 1
- Create streaming session
- Poll data file loading status until PXDAC RAM full
- Issue software trigger
- Poll file playback status until complete
- End streaming session
- Close streaming session
- Disconnect from the device
- Unload the library

Notes on 64-bit MATLAB SDK

The MATLAB 64-bit SDK cannot call the PXDAC4800 library directly, so it goes through a wrapper dll named pxdac4800_wrap. This DLL, along with it's associated lib file, should be located in the lib64 folder of your PXD4800 MATLAB SDK installation folder. The wrapper library is called the same way that 32-bit MATLAB samples call the PXDAC4800 library.

The wrapper library wraps most, but not all, of the PXDAX4800 API that is documented in the PXDAC4800 Operators Manual. The calls that are not supported are generally not suitable for use in MATLAB, such as calls that manipulate the DMA buffers directly, use complicated structures with pointers or use memory that's been allocated by the PXDAC4800 library. Specifically, the calls not included in the wrapper dll are:

GetErrorTextXD48
 DumpLibErrorXD48
 GetBoardNameXD48

GetVersionTextXD48
_SetActiveMemoryRegionXD48
AllocateDmaBufferXD48
EnsureUtilityDmaBufferXD48
FreeDmaBufferXD48
FreeMemoryXD48
FreeUtilityBufferXD48
GetUtilityBufferXD48

LoadStreamDataFastXD48
SessionStreamCreateXD48
_SessionStreamCreateStdXD48
SessionStreamDeleteXD48
SessionStreamEndXD48
SessionStreamProgressXD48
SessionStreamSnapshotXD48

As shown in the PXDAC4800_Streaming.m sample, streaming sessions can be performed using the 'NoStruct' equivalent of these calls.

Refer to the PXDAC4800 Operators Manual for information on the parameters and constants to use in any SDK call.

3 APPENDIX 1 – REVISION HISTORY

Revision 0.1 – Internal initial release

Revision 1.0 – Initial release

Revision 1.1 – Added Streaming

Revision 1.2 – Added 64-bit support

Revision 1.21 – Copyright Vitrek LLC