```
/*
 * ADDAC SYSTEM 2012
 *
 * Voltage Controlled Computer C++ LIBRARY
 * Version 0.22 November, 2012
 * Copyright 2012 André Gonçalves
 * For details, see http://
 *
 * AD5668 programming based on
 * DAC V11 by Robin Price 2009
 *
 * shiftIn Example 2.1 by Carlyn Maw
 */
```

ADDAC LIBRARY INDEX

ADDAC Library	Code Reference	VS.0004
ADDAC.h & ADDAC.cpp	Main Class	
ADDAC_Adsr.h & ADDAC_Adsr.cpp	ADSR Class	
ADDAC_Clock.h & ADDAC_Clock.cpp	Clock Class	
ADDAC_CvBuffer.h & ADDAC_CvBuffer.cpp	CV Buffer Class	
ADDAC_CvLooper.h & ADDAC_CvLooper.cpp	CV Looper Class	
ADDAC_Empty.h & ADDAC_Empty.cpp	Basic Class Construction Example	
ADDAC_LFO.h & ADDAC_LFO.cpp	LFO Class	
ADDAC_Liss.h & ADDAC_Liss.cpp	Liss Class	
ADDAC_Logic.h & ADDAC_Logic.cpp	Logic Class	
ADDAC_MIDI.h & ADDAC_MIDI.cpp	MIDI Class	
ADDAC_Physics.h & ADDAC_Physics.cpp	Marble Physics Class	
ADDAC_Quad.h & ADDAC_Quad.cpp	Quadraphonic Spatializer Class	
ADDAC_Quantizer.h & ADDAC_ Quantizer.cpp	Quantizer Class	

Main Class

ADDAC.h & ADDAC.cpp

Main Class

This Class configures all MCU pin definitions for ADDAC 00X Series

Provides control for the CV Outputs
Provides control for the ALL Expansions I/O
Features a Mixer Function

Implemented Functions:

setup();	ADDAC001 Setup
update();	Update MODE & SUBMODE Readings
readMODEswitch();	Updates MODE & SUBMODE Readings
readMODE();	Returns MODE Value
readSUBMODE();	Returns SUBMODE Value
WriteChannel(int _channel, unsigned int _voltage);	Writes to CV Outputs
ReadCvs(int _socket);	Reads from ADDAC002 in Socket A, B or C
ReadCv(int _socket, int _channel);	Reads from ADDAC002 in Socket A, B or C in defined Channel
ReadManuals(int _socket, int _channel);	Reads from ADDAC003 in Socket A, B or C
ReadManual(int _socket, int _channel);	Reads from ADDAC003 in Socket A
ReadGates(int _socket, bool _invert);	Reads from ADDAC004 in Socket A, B or C
ReadGate(int _socket, bool _invert, int _channel);	Reads from ADDAC004 in Socket A, B or C in defined Channel
WriteGatesA(int _channel, int _data)	Writes to ADDAC005 in Socket A
WriteGatesB(int _channel, int _data)	Writes to ADDAC005 in Socket B
mixerMode()	Channel Mixer

ADSR Class

ADDAC_Adsr.h & ADDAC_Adsr.cpp

Complex ADSR Class

This Class features a fully controllable ADSR or AD CV Generator

Independent Time and Amplitude for each stage
Independent Log/Exp curves for each stage
Free Mode or Sustain Mode for Note On commands
Trigger and Reset

Implemented Functions:

AD_trigger();	Triggers AD Mode
AD_trigger(float _A);	Triggers AD Mode at a certain amplitude
AD_release();	Release AD
AD_update(float _Atime, float _Dtime);	Updates AD
ADSR_update(	Updates ADSR
float _A, float _Atime,	Attack amplitude and time
float _D, float _Dtime,	Decay amplitude and time
float _S,float _Stime,	Sustain amplitude and time
float _Rtime);	Release time
adsrMode(	Standard Free Mode ADSR
int _channel, bool _trigger, bool _inverted,	
float _A, float _Atime,	Attack amplitude and time
float _D, float _Dtime,	Decay amplitude and time
float _S,float _Stime,	Sustain amplitude and time
float _Rtime);	Release time
adsrLogExpMode(	Standard Free Mode Log/Exp ADSR
int _channel, bool _trigger, bool _inverted,	
float _A, float _Atime, float _Ashape,	Attack amplitude, time and Log/Exp Shape
float _D, float _Dtime, float _Dshape,	Decay amplitude, time and Log/Exp Shape
float _S, float _Stime, float _Sshape,	Sustain amplitude, time and Log/Exp Shape
float _Rtime, float _Rshape);	Release time and Log/Exp Shape

Operational variables:

bool ADSRtrigger;	range: 0, 1	Triggers ADSR Mode
bool SUSTAIN;	range: 0, 1	Sustain for Gate On control

Output variable:

unsigned int CVstream;	range: 0, 65535	Variable to Acess ADSR Stream

ADSR Class

ADDAC_Adsr.h & ADDAC_Adsr.cpp

Complex ADSR Class

This Class features a fully controllable ADSR or AD CV Generator

Independent Time and Amplitude for each stage
Independent Log/Exp curves for each stage
Free Mode or Sustain Mode for Note On commands
Trigger and Reset

Implemented Functions:

AD_trigger();	Triggers AD Mode
AD_trigger(float _A);	Triggers AD Mode at a certain amplitude
AD_release();	Release AD
AD_update(float _Atime, float _Dtime);	Updates AD
ADSR_update(	Updates ADSR
float _A, float _Atime,	Attack amplitude and time
float _D, float _Dtime,	Decay amplitude and time
float _S,float _Stime,	Sustain amplitude and time
float _Rtime);	Release time
adsrMode(	Standard Free Mode ADSR
bool _trigger, bool _inverted,	
float _A, float _Atime,	Attack amplitude and time
float _D, float _Dtime,	Decay amplitude and time
float _S,float _Stime,	Sustain amplitude and time
float _Rtime);	Release time
adsrLogExpMode(	Standard Free Mode Log/Exp ADSR
int _channel, bool _trigger, bool _inverted,	
float _A, float _Atime, float _Ashape,	Attack amplitude, time and Log/Exp Shape
float _D, float _Dtime, float _Dshape,	Decay amplitude, time and Log/Exp Shape
float _S, float _Stime, float _Sshape,	Sustain amplitude, time and Log/Exp Shape
float _Rtime, float _Rshape);	Release time and Log/Exp Shape

Operational variables:

bool ADSRtrigger;	range: 0, 1	Triggers ADSR Mode
bool SUSTAIN;	range: 0, 1	Sustain for Gate On control

Output variable:

unsigned int CVstream;	range: 0, 65535	Variable to Acess ADSR Stream

```
/*
 * ADDAC SYSTEM 2012
 *
 * Necessary or Recommended External Libraries
 *
 */
```

External LIBRARIES

Library Name	Reference	Direct Link
StandardCplusplus	Needed to use ADDAC Library (important)	https://github.com/maniacbug/StandardCplusplus
MIDI	Needed for ADDAC009 MIDI communication	http://arduino.cc/playground/Main/MIDILibrary
Z_OSC	Needed for ADDAC007 OSC communication	https://github.com/djiamnot/Z_OSC
SimpleMessageSystem	Needed for MAX-MSP communication	http://www.arduino.cc/playground/Code/SimpleMes
Iannix	Needed for interfacing with IANNIX software	http://www.iannix.org/en/download.php
MemoryFree	Needed for debugging available memory	http://arduino.cc/playground/Code/AvailableMemor

ADDAC Main Class Library | **Code Reference** | **VS.0004**

ADDAC.h & ADDAC.cpp

ADDAC.h

Hard-Coded Definitions

ADDAC001 Version

```
// WHICH VCC VERSION?
//#define VS1
//#define VS2
//#define VS3
```

Uncomment correct line to define Your version
Set VS1
Set VS2
Set VS3

ADDAC004 Version

```
// WHICH 004 VERSION?
//#define ADDAC004VS1
//#define ADDAC004VS2
```

Uncomment correct line to define Your version
Set VS1
Set VS2

Constants:

```
#define addacMaxResolution 65535
```

Maximum CV out range: 16bit

ADDAC.cpp

`setup();`
Initializes ADDAC001

variables initialized:

```
long DACvolts[8];
unsigned long DACtimes[8];
unsigned int RNDdelays[8];
```

range: 1 to 8
, channel number = position in array

function type: `void`

`update();`
Updates MODE & SUBMODE Switches using:

`readMODEswitch();`
described below

function type: `void`

READ MODE & SUBMODE SWITCHES

`readMODEswitch();`
Updates MODE & SUBMODE Variables

variables updated:

```
int MODE
int SUBMODE
```

range: 0, 15
range: 0, 15

function type: `void`

`readMODE();`
Reads & returns the MODE button position:

```
int MODE
```

range: 0, 15

function type: `int`

`readSUBMODE();`
Reads & returns the SUBMODE button position:

```
int SUBMODE
```

range: 0, 15

function type: `int`

`readOnboardPot();`
Returns the ONBOARD pot value
(VS.I & II only)

(VS.I & II only)

```
int onboardVal
unsigned int onboardValMapped
```

range: 0, 1023
range: 0, 65535

function type: `int`

GENERATE CV OUTPUT

`WriteChannel(int _channel, unsigned int _voltage);`
Send a Voltage to a channel

```
int _channel
unsigned int _voltage
```

range: 0, 7
range: 0, 65535

function type: `void`

Channel to Set
Voltage range: 0 = 0v & 65535 = 5v

READ FROM ADDAC002

Connected to SOCKET A

`ReadCvsA();`
Read all 6 channels at once

updates variables:

```
unsigned int cvValuesA[6]
unsigned int cvValuesAMapped[6]
```

range: 0, 1023
range: 0, 65535

function type: `void`

`ReadCvsA(int _channel);`
Reads & Returns the value from requested channel

```
int _channel
```

range: 0, 5

function type: `int`

updates variable:

```
unsigned int cvValuesA[6]
```

range: 0, 1023

Connected to SOCKET B

`ReadCvsB(int _channel);`
Reads & Returns the value from requested channel

```
int _channel
```

range: 0, 5

function type: `int`

updates variable:

```
unsigned int cvValuesB[6]
```

range: 0, 1023

READ FROM ADDAC003

Connected to SOCKET A

`ReadAnalogsA();`
Read all 5 channels at once

updates variables:

```
unsigned int analogValuesA[6]
unsigned int analogValuesAMapped[6]
```

range: 0, 1023
range: 0, 65535

function type: `void`

`ReadAnalogsA(int _channel);`
Returns the value from each channel individually

```
int _channel
```

range: 0, 4

function type: `int`

updates variable:

```
unsigned int analogValuesA[_channel]
```

range: 0, 1023

Connected to SOCKET B

`ReadAnalogsB();`
Read all 5 channels at once

updates variables:

```
unsigned int analogValuesB[6]
unsigned int analogValuesBMapped[6]
```

range: 0, 1023
range: 0, 65535

function type: `void`

`ReadAnalogsB(int _channel);`
Returns the value from each channel individually

```
int _channel
```

range: 0, 4

function type: `int`

updates variable:

```
unsigned int analogValuesB[_channel]
```

range: 0, 1023

READ FROM ADDAC004

Connected to SOCKET A

`ReadGatesA(bool _invert);`
Read all 8 channels at once

```
bool _invert
```

range: 0, 1

(0= normal readings, 1= inverted readings)

updates variables:

Variable that holds all channel A values

```
byte gateValuesA[8]
```

range: 0, 1

0 = below 2.5v, 1 = above 2.5v

`MAXsendGatesA();`
Sends values to MAX-MSP when communication is enabled

function type: `void`

Connected to SOCKET B

`ReadGatesB(bool _invert);`
Read all 8 channels at once

```
bool _invert
```

range: 0, 1

(0= normal readings, 1= inverted readings)

updates variables:

Variable that holds all channel B values

```
byte gateValuesA[8]
```

range: 0, 1

0 = below 2.5v, 1 = above 2.5v

`MAXsendGatesA();`
Sends values to MAX-MSP when communication is enabled

function type: `void`

WRITE TO ADDAC005

Connected to SOCKET A

`WriteGatesA(int _channel, int _data)`
Change each channel independently

```
int _channel
int data
```

range: 0, 7
range: 0, 1

Channel to Set
Set State: 0= 0v, 1= 5v

function type: `void`

Connected to SOCKET B

`WriteGatesB(int _channel, int _data)`
Change each channel independently

```
int _channel
int data
```

range: 0, 7
range: 0, 1

Channel to Set
Set State: 0= 0v, 1= 5v

function type: `void`

MIXER FUNCTIONS

`void ADDAC::mixerMode(){`
Calculates an average of the first 7 channels and updates channel 8 with the result.

```
DACvolts[8]=
```