

Arduino MIDI Library
Version 3.0

Generated by Doxygen 1.5.8

Sun Mar 6 16:07:36 2011

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	MIDI_Class Class Reference	5
3.1.1	Detailed Description	6
3.1.2	Constructor & Destructor Documentation	6
3.1.2.1	MIDI_Class	6
3.1.2.2	~MIDI_Class	6
3.1.3	Member Function Documentation	6
3.1.3.1	begin	6
3.1.3.2	check	7
3.1.3.3	getChannel	7
3.1.3.4	getData1	7
3.1.3.5	getData2	8
3.1.3.6	getFilterMode	8
3.1.3.7	getInputChannel	8
3.1.3.8	getSysExArray	8
3.1.3.9	getThruState	8
3.1.3.10	getType	8
3.1.3.11	read	9
3.1.3.12	read	9
3.1.3.13	sendAfterTouch	9
3.1.3.14	sendControlChange	9
3.1.3.15	sendNoteOff	10
3.1.3.16	sendNoteOn	10

3.1.3.17	sendPitchBend	10
3.1.3.18	sendPitchBend	11
3.1.3.19	sendPolyPressure	11
3.1.3.20	sendProgramChange	11
3.1.3.21	sendRealTime	12
3.1.3.22	sendSongPosition	12
3.1.3.23	sendSongSelect	12
3.1.3.24	sendSysEx	13
3.1.3.25	sendTimeCodeQuarterFrame	13
3.1.3.26	sendTimeCodeQuarterFrame	13
3.1.3.27	sendTuneRequest	14
3.1.3.28	setInputChannel	14
3.1.3.29	setThruFilterMode	14
3.1.3.30	setThruFilterMode	14
3.1.3.31	turnThruOff	15
3.1.3.32	turnThruOn	15
3.2	midimsg Struct Reference	16
3.2.1	Detailed Description	16
3.2.2	Member Data Documentation	16
3.2.2.1	channel	16
3.2.2.2	data1	16
3.2.2.3	data2	16
3.2.2.4	sysex_array	16
3.2.2.5	type	17
3.2.2.6	valid	17
4	File Documentation	19
4.1	/Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILib/trunk/Arduino/Compatibility_v2.5.h File Reference	19
4.1.1	Detailed Description	19
4.1.2	Define Documentation	20
4.1.2.1	ATCanal	20
4.1.2.2	ATPoly	20
4.1.2.3	CC	20
4.1.2.4	MIDI_FILTER_ANTICANAL	20
4.1.2.5	MIDI_FILTER_CANAL	20
4.1.2.6	MIDI_FILTER_FULL	20

4.1.2.7	MIDI_FILTER_OFF	20
4.1.2.8	MIDI_rate	20
4.1.2.9	PC	20
4.1.2.10	SysEx	21
4.2	/Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILib/trunk/Arduino/MIDI.cpp File Reference	22
4.2.1	Detailed Description	22
4.2.2	Variable Documentation	22
4.2.2.1	MIDI	22
4.3	/Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILib/trunk/Arduino/MIDI.h File Reference	23
4.3.1	Detailed Description	24
4.3.2	Define Documentation	24
4.3.2.1	COMPATIBILITY_V25	24
4.3.2.2	COMPFLAG_MIDI_IN	24
4.3.2.3	COMPFLAG_MIDI_OUT	24
4.3.2.4	MIDI_BAUDRATE	24
4.3.2.5	MIDI_CHANNEL_OFF	24
4.3.2.6	MIDI_CHANNEL_OMNI	24
4.3.2.7	MIDI_SYSEX_ARRAY_SIZE	24
4.3.2.8	USE_RUNNING_STATUS	24
4.3.2.9	USE_SERIAL_PORT	25
4.3.3	Typedef Documentation	25
4.3.3.1	byte	25
4.3.4	Enumeration Type Documentation	25
4.3.4.1	kMIDIType	25
4.3.4.2	kThruFilterMode	26
4.3.5	Variable Documentation	26
4.3.5.1	MIDI	26

Chapter 1

Class Index

1.1 Class List

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

MIDI_Class	5
midimsg	16

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

/Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILib/trunk/Arduino/ Compatibility_v2.5.h (Compatibility file for MIDI Library v2.5 Version 3.0)	19
/Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILib/trunk/Arduino/ MIDI.cpp (MIDI Library for the Arduino)	22
/Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILib/trunk/Arduino/ MIDI.h (MIDI Library for the Arduino Version 3.0)	23

Chapter 3

Class Documentation

3.1 MIDI_Class Class Reference

```
#include <MIDI.h>
```

Public Member Functions

- [MIDI_Class](#) ()
- [~MIDI_Class](#) ()
- void [begin](#) (const [byte](#) inChannel=1)
- void [sendNoteOn](#) ([byte](#) NoteNumber, [byte](#) Velocity, [byte](#) Channel)
- void [sendNoteOff](#) ([byte](#) NoteNumber, [byte](#) Velocity, [byte](#) Channel)
- void [sendProgramChange](#) ([byte](#) ProgramNumber, [byte](#) Channel)
- void [sendControlChange](#) ([byte](#) ControlNumber, [byte](#) ControlValue, [byte](#) Channel)
- void [sendPitchBend](#) (unsigned int PitchValue, [byte](#) Channel)
- void [sendPitchBend](#) (double PitchValue, [byte](#) Channel)
- void [sendPolyPressure](#) ([byte](#) NoteNumber, [byte](#) Pressure, [byte](#) Channel)
- void [sendAfterTouch](#) ([byte](#) Pressure, [byte](#) Channel)
- void [sendSysEx](#) ([byte](#) length, [byte](#) *array, bool ArrayContainsBoundaries=false)
- void [sendTimeCodeQuarterFrame](#) ([byte](#) TypeNibble, [byte](#) ValuesNibble)
- void [sendTimeCodeQuarterFrame](#) ([byte](#) data)
- void [sendSongPosition](#) (unsigned int Beats)
- void [sendSongSelect](#) ([byte](#) SongNumber)
- void [sendTuneRequest](#) ()
- void [sendRealTime](#) ([kMIDIType](#) Type)
- bool [read](#) ()
- bool [read](#) (const [byte](#) Channel)
- [kMIDIType](#) [getType](#) ()
- [byte](#) [getChannel](#) ()
- [byte](#) [getData1](#) ()
- [byte](#) [getData2](#) ()
- [byte](#) * [getSysExArray](#) ()
- bool [check](#) ()
- [byte](#) [getInputChannel](#) ()
- void [setInputChannel](#) (const [byte](#) Channel)

- [kThruFilterMode](#) [getFilterMode](#) ()
- bool [getThruState](#) ()
- void [turnThruOn](#) ([kThruFilterMode](#) [inThruFilterMode=Full](#))
- void [turnThruOff](#) ()
- void [setThruFilterMode](#) (const [byte](#) [inThruFilterMode](#))
- void [setThruFilterMode](#) (const [kThruFilterMode](#) [inThruFilterMode](#))

3.1.1 Detailed Description

The main class for MIDI handling. See member descriptions to know how to use it, or check out the examples supplied with the library.

Definition at line 120 of file MIDI.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 MIDI_Class::MIDI_Class ()

Default constructor for [MIDI_Class](#).

Definition at line 22 of file MIDI.cpp.

```
22 { }
```

3.1.2.2 MIDI_Class::~~MIDI_Class ()

Default destructor for [MIDI_Class](#).

This is not really useful for the Arduino, as it is never called...

Definition at line 26 of file MIDI.cpp.

```
26 { }
```

3.1.3 Member Function Documentation

3.1.3.1 void MIDI_Class::begin (const byte *inChannel* = 1)

Call the begin method in the setup() function of the Arduino. All parameters are set to their default values:

- Input channel set to 1 if no value is specified
- Full thru mirroring

Definition at line 34 of file MIDI.cpp.

```
34                                     {
35
36     // Initialise the Serial port
37     USE_SERIAL_PORT.begin(MIDI_BAUDRATE);
38
39
```

```
40 #if COMPFLAG_MIDI_OUT
41
42 #if USE_RUNNING_STATUS
43     mRunningStatus_TX = InvalidType;
44 #endif // USE_RUNNING_STATUS
45
46 #endif // COMPFLAG_MIDI_OUT
47
48
49 #if COMPFLAG_MIDI_IN
50
51     mInputChannel = inChannel;
52     mRunningStatus_RX = InvalidType;
53     mPendingMessageIndex = 0;
54     mPendingMessageExpectedLenght = 0;
55
56     mMessage.valid = false;
57     mMessage.type = InvalidType;
58     mMessage.channel = 0;
59     mMessage.data1 = 0;
60     mMessage.data2 = 0;
61
62 #endif // COMPFLAG_MIDI_IN
63
64
65 #if (COMPFLAG_MIDI_IN && COMPFLAG_MIDI_OUT) // Thru
66
67     mThruFilterMode = Full;
68
69 #endif // Thru
70
71 }
```

3.1.3.2 bool MIDI_Class::check ()

Check if a valid message is stored in the structure.

Definition at line 599 of file MIDI.cpp.

```
599 { return mMessage.valid; }
```

3.1.3.3 byte MIDI_Class::getChannel ()

Getter method: access to the channel of the message stored in the structure.

Definition at line 591 of file MIDI.cpp.

```
591 { return mMessage.channel; }
```

3.1.3.4 byte MIDI_Class::getData1 ()

Getter method: access to the first data byte of the message stored in the structure.

If the message is SysEx, the length of the array is stocked there.

Definition at line 593 of file MIDI.cpp.

```
593 { return mMessage.data1; }
```

3.1.3.5 byte MIDI_Class::getData2 ()

Getter method: access to the second data byte of the message stored in the structure.

Definition at line 595 of file MIDI.cpp.

```
595 { return mMessage.data2; }
```

3.1.3.6 kThruFilterMode MIDI_Class::getFilterMode () [inline]

Definition at line 226 of file MIDI.h.

```
226 { return mThruFilterMode; }
```

3.1.3.7 byte MIDI_Class::getInputChannel () [inline]

Definition at line 186 of file MIDI.h.

```
186 { return mInputChannel; }
```

3.1.3.8 byte * MIDI_Class::getSysExArray ()

Getter method: access to the System Exclusive byte array. Array length is stocked in Data1.

Definition at line 597 of file MIDI.cpp.

```
597 { return mMessage.sysex_array; }
```

3.1.3.9 bool MIDI_Class::getThruState () [inline]

Definition at line 227 of file MIDI.h.

```
227 { return mThruActivated; }
```

3.1.3.10 kMIDIType MIDI_Class::getType ()

Getter method: access to the message type stored in the structure.

Returns an enumerated type.

Definition at line 589 of file MIDI.cpp.

```
589 { return mMessage.type; }
```

3.1.3.11 bool MIDI_Class::read (const byte *inChannel*)

Reading/thru-ing method, the same as [read\(\)](#) with a given input channel to read on.

Definition at line 306 of file MIDI.cpp.

```
306                                     {
307
308         if (inChannel >= MIDI_CHANNEL_OFF) return false; // MIDI Input disabled.
309
310         if (parse(inChannel)) return filter(inChannel);
311         else return false;
312
313 }
```

3.1.3.12 bool MIDI_Class::read ()

Read a MIDI message from the serial port using the main input channel (see [setInputChannel\(\)](#) for reference).

Returned value: true if any valid message has been stored in the structure, false if not. A valid message is a message that matches the input channel.

If the Thru is enabled and the messages matches the filter, it is sent back on the MIDI output.

Definition at line 301 of file MIDI.cpp.

```
301                                     {
302         return read(mInputChannel);
303 }
```

3.1.3.13 void MIDI_Class::sendAfterTouch (byte *Pressure*, byte *Channel*)

Send a MonoPhonic AfterTouch message (applies to all notes)

Parameters:

Pressure The amount of AfterTouch to apply to all notes.

Channel The channel on which the message will be sent (1 to 16).

Definition at line 192 of file MIDI.cpp.

```
192 { send(AfterTouchChannel,Pressure,0,Channel); }
```

3.1.3.14 void MIDI_Class::sendControlChange (byte *ControlNumber*, byte *ControlValue*, byte *Channel*)

Send a Control Change message

Parameters:

ControlNumber The controller number (0 to 127). See the detailed description here:
<http://www.somascape.org/midi/tech/spec.html#ctrlnums>

ControlValue The value for the specified controller (0 to 127).

Channel The channel on which the message will be sent (1 to 16).

Definition at line 179 of file MIDI.cpp.

```
179 { send(ControlChange,ControlNumber,ControlValue,Channel); }
```

3.1.3.15 void MIDI_Class::sendNoteOff (byte NoteNumber, byte Velocity, byte Channel)

Send a Note Off message (a real Note Off, not a Note On with null velocity)

Parameters:

NoteNumber Pitch value in the MIDI format (0 to 127). Take a look at the values, names and frequencies of notes here: <http://www.phys.unsw.edu.au/jw/notes.html>

Velocity Release velocity (0 to 127).

Channel The channel on which the message will be sent (1 to 16).

Definition at line 166 of file MIDI.cpp.

```
166 { send(NoteOff,NoteNumber,Velocity,Channel); }
```

3.1.3.16 void MIDI_Class::sendNoteOn (byte NoteNumber, byte Velocity, byte Channel)

Send a Note On message

Parameters:

NoteNumber Pitch value in the MIDI format (0 to 127). Take a look at the values, names and frequencies of notes here: <http://www.phys.unsw.edu.au/jw/notes.html>

Velocity Note attack velocity (0 to 127). A NoteOn with 0 velocity is considered as a NoteOff.

Channel The channel on which the message will be sent (1 to 16).

Definition at line 159 of file MIDI.cpp.

```
159 { send(NoteOn,NoteNumber,Velocity,Channel); }
```

3.1.3.17 void MIDI_Class::sendPitchBend (double PitchValue, byte Channel)

Send a Pitch Bend message using a floating point value.

Parameters:

PitchValue The amount of bend to send (in a floating point format), between -1 (maximum downwards bend) and +1 (max upwards bend), center value is 0.

Channel The channel on which the message will be sent (1 to 16).

Definition at line 207 of file MIDI.cpp.


```

207                                     {
208
209         unsigned int pitchval = (PitchValue+1.f)*8192;
210         if (pitchval > 16383) pitchval = 16383;           // overflow protection
211         sendPitchBend(pitchval,Channel);
212
213     }

```

3.1.3.18 void MIDI_Class::sendPitchBend (unsigned int *PitchValue*, byte *Channel*)

Send a Pitch Bend message using an integer value.

Parameters:

PitchValue The amount of bend to send (in an integer format), between 0 (maximum downwards bend) and 16383 (max upwards bend), center value is 8192.

Channel The channel on which the message will be sent (1 to 16).

Definition at line 198 of file MIDI.cpp.

```

198                                     {
199
200         send(PitchBend, (PitchValue & 0x7F), (PitchValue >> 7) & 0x7F,Channel);
201
202     }

```

3.1.3.19 void MIDI_Class::sendPolyPressure (byte *NoteNumber*, byte *Pressure*, byte *Channel*)

Send a Polyphonic AfterTouch message (applies to only one specified note)

Parameters:

NoteNumber The note to apply AfterTouch to (0 to 127).

Pressure The amount of AfterTouch to apply (0 to 127).

Channel The channel on which the message will be sent (1 to 16).

Definition at line 186 of file MIDI.cpp.

```

186 { send(AfterTouchPoly,NoteNumber,Pressure,Channel); }

```

3.1.3.20 void MIDI_Class::sendProgramChange (byte *ProgramNumber*, byte *Channel*)

Send a Program Change message

Parameters:

ProgramNumber The Program to select (0 to 127).

Channel The channel on which the message will be sent (1 to 16).

Definition at line 172 of file MIDI.cpp.

```

172 { send(ProgramChange,ProgramNumber,0,Channel); }

```

3.1.3.21 void MIDI_Class::sendRealTime (kMIDIType *Type*)

Send a Real Time (one byte) message.

You can also send a Tune Request with this method.

Parameters:

Type The available Real Time types are: Start, Stop, Continue, Clock, ActiveSensing and SystemReset.

Definition at line 273 of file MIDI.cpp.

```
273                                     {
274     switch (Type) {
275         case TuneRequest: // Not really real-time, but one byte anyway.
276         case Clock:
277         case Start:
278         case Stop:
279         case Continue:
280         case ActiveSensing:
281         case SystemReset:
282             USE_SERIAL_PORT.write((byte) Type);
283             break;
284         default:
285             // Invalid Real Time marker
286             break;
287     }
288 }
```

3.1.3.22 void MIDI_Class::sendSongPosition (unsigned int *Beats*)

Send a Song Position Pointer message.

Parameters:

Beats The number of beats since the start of the song.

Definition at line 254 of file MIDI.cpp.

```
254                                     {
255
256     USE_SERIAL_PORT.write(SongPosition);
257     USE_SERIAL_PORT.write(Beats & 0x7F);
258     USE_SERIAL_PORT.write((Beats >> 7) & 0x7F);
259
260 }
```

3.1.3.23 void MIDI_Class::sendSongSelect (byte *SongNumber*)

Send a Song Select message

Definition at line 263 of file MIDI.cpp.

```
263                                     {
264
265     USE_SERIAL_PORT.write(SongSelect);
266     USE_SERIAL_PORT.write(SongNumber & 0x7F);
267
268 }
```

3.1.3.24 void MIDI_Class::sendSysEx (byte *length*, byte * *array*, bool *ArrayContainsBoundaries* = false)

Generate and send a System Exclusive frame.

Parameters:

length The size of the array to send

array The byte array containing the data to send

ArrayContainsBoundaries When set to 'true', 0xF0 & 0xF7 bytes (start & stop SysEx) will NOT be sent (and therefore must be included in the array).\ default value is set to 'false' for compatibility with previous versions of the library.

Definition at line 221 of file MIDI.cpp.

```

221                                     {
222         if (!ArrayContainsBoundaries) USE_SERIAL_PORT.write(0xF0);
223         for (byte i=0;i<length;i++) USE_SERIAL_PORT.write(array[i]);
224         if (!ArrayContainsBoundaries) USE_SERIAL_PORT.write(0xF7);
225     }
```

3.1.3.25 void MIDI_Class::sendTimeCodeQuarterFrame (byte *data*)

Send a MIDI Time Code Quarter Frame. See MIDI Specification for more information.

Parameters:

data if you want to encode directly the nibbles in your program, you can send the byte here.

Definition at line 244 of file MIDI.cpp.

```

244                                     {
245
246         USE_SERIAL_PORT.write(TimeCodeQuarterFrame);
247         USE_SERIAL_PORT.write(data);
248
249     }
```

3.1.3.26 void MIDI_Class::sendTimeCodeQuarterFrame (byte *TypeNibble*, byte *ValuesNibble*)

Send a MIDI Time Code Quarter Frame. See MIDI Specification for more information.

Parameters:

TypeNibble MTC type

ValuesNibble MTC data

Definition at line 234 of file MIDI.cpp.

```

234                                     {
235
236         byte data = ( ((TypeNibble & 0x07) << 4) | (ValuesNibble & 0x0F) );
237         sendTimeCodeQuarterFrame(data);
238
239     }
```

3.1.3.27 void MIDI_Class::sendTuneRequest ()

Send a Tune Request message. When a MIDI unit receives this message, it should tune its oscillators (if equipped with any)

Definition at line 228 of file MIDI.cpp.

```
228 { sendRealTime(TuneRequest); }
```

3.1.3.28 void MIDI_Class::setInputChannel (const byte *Channel*)

Set the value for the input MIDI channel

Parameters:

Channel the channel value. Valid values are 1 to 16, MIDI_CHANNEL_OMNI if you want to listen to all channels, and MIDI_CHANNEL_OFF to disable MIDI input.

Definition at line 606 of file MIDI.cpp.

```
606 { mInputChannel = inChannel; }
```

3.1.3.29 void MIDI_Class::setThruFilterMode (const kThruFilterMode *inThruFilterMode*)

Set the filter for thru mirroring

Parameters:

inThruFilterMode a filter mode See kThruFilterMode for detailed description.

Definition at line 620 of file MIDI.cpp.

```
620 {
621     mThruFilterMode = inThruFilterMode;
622     if (mThruFilterMode != Off) mThruActivated = true;
623     else mThruActivated = false;
624 }
```

3.1.3.30 void MIDI_Class::setThruFilterMode (const byte *inThruFilterMode*)

Set the filter for thru mirroring

Parameters:

inThruFilterMode a filter mode See kThruFilterMode for detailed description.

This method uses a byte parameter and is for compatibility only, please use kThruFilterMode for future programs.

Definition at line 630 of file MIDI.cpp.

```
630 {
631     mThruFilterMode = (kThruFilterMode)inThruFilterMode;
632     if (mThruFilterMode != Off) mThruActivated = true;
633     else mThruActivated = false;
634 }
```

3.1.3.31 void MIDI_Class::turnThruOff ()

Setter method: turn message mirroring off.

Definition at line 643 of file MIDI.cpp.

```
643         {  
644             mThruActivated = false;  
645             mThruFilterMode = Off;  
646     }
```

3.1.3.32 void MIDI_Class::turnThruOn (kThruFilterMode *inThruFilterMode* = Full)

Setter method: turn message mirroring on.

Definition at line 638 of file MIDI.cpp.

```
638         {  
639             mThruActivated = true;  
640             mThruFilterMode = inThruFilterMode;  
641     }
```

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

- /Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILib/trunk/Arduino/[MIDI.h](#)
- /Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILib/trunk/Arduino/[MIDI.cpp](#)

3.2 midimsg Struct Reference

```
#include <MIDI.h>
```

Public Attributes

- [byte channel](#)
- [kMIDIType type](#)
- [byte data1](#)
- [byte data2](#)
- [byte sysex_array](#) [MIDI_SYSEX_ARRAY_SIZE]
- [bool valid](#)

3.2.1 Detailed Description

The [midimsg](#) structure contains decoded data of a MIDI message read from the serial port with `read()` or `thru()`.

Definition at line 100 of file MIDI.h.

3.2.2 Member Data Documentation

3.2.2.1 `byte midimsg::channel`

The MIDI channel on which the message was recieved.

Value goes from 1 to 16.

Definition at line 102 of file MIDI.h.

3.2.2.2 `byte midimsg::data1`

The first data byte.

Value goes from 0 to 127.

If the message is SysEx, this byte contains the array length.

Definition at line 106 of file MIDI.h.

3.2.2.3 `byte midimsg::data2`

The second data byte. If the message is only 2 bytes long, this one is null.

Value goes from 0 to 127.

Definition at line 108 of file MIDI.h.

3.2.2.4 `byte midimsg::sysex_array[MIDI_SYSEX_ARRAY_SIZE]`

System Exclusive dedicated byte array.

Array length is stocked in data1.

Definition at line 110 of file MIDI.h.

3.2.2.5 kMIDIType midimsg::type

The type of the message (see the define section for types reference)

Definition at line 104 of file MIDI.h.

3.2.2.6 bool midimsg::valid

This boolean indicates if the message is valid or not. There is no channel consideration here, validity means the message respects the MIDI norm.

Definition at line 112 of file MIDI.h.

The documentation for this struct was generated from the following file:

- /Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILib/trunk/Arduino/[MIDI.h](#)

Chapter 4

File Documentation

4.1 /Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILibrary_v2.5.h File Reference

Compatibility file for MIDI Library v2.5 Version 3.0.

Defines

- #define [MIDI_rate](#) MIDI_BAUDRATE
- #define [ATPoly](#) AfterTouchPoly
- #define [CC](#) ControlChange
- #define [PC](#) ProgramChange
- #define [ATCanal](#) AfterTouchChannel
- #define [SysEx](#) SystemExclusive
- #define [MIDI_FILTER_OFF](#) 0
- #define [MIDI_FILTER_FULL](#) 1
- #define [MIDI_FILTER_CANAL](#) 2
- #define [MIDI_FILTER_ANTICANAL](#) 3

4.1.1 Detailed Description

Compatibility file for MIDI Library v2.5 Version 3.0.

Project MIDI Library

Author:

François Best

Date:

24/02/11 License GPL Forty Seven Effects - 2011

Definition in file [Compatibility_v2.5.h](#).

4.1.2 Define Documentation

4.1.2.1 **#define ATCanal AfterTouchChannel**

Message type AfterTouch Channel

Definition at line 30 of file Compatibility_v2.5.h.

4.1.2.2 **#define ATPoly AfterTouchPoly**

Message type AfterTouch Poly

Definition at line 24 of file Compatibility_v2.5.h.

4.1.2.3 **#define CC ControlChange**

Message type Control Change

Definition at line 26 of file Compatibility_v2.5.h.

4.1.2.4 **#define MIDI_FILTER_ANTICANAL 3**

Definition at line 38 of file Compatibility_v2.5.h.

4.1.2.5 **#define MIDI_FILTER_CANAL 2**

Definition at line 37 of file Compatibility_v2.5.h.

4.1.2.6 **#define MIDI_FILTER_FULL 1**

Definition at line 36 of file Compatibility_v2.5.h.

4.1.2.7 **#define MIDI_FILTER_OFF 0**

Definition at line 35 of file Compatibility_v2.5.h.

4.1.2.8 **#define MIDI_rate MIDI_BAUDRATE**

The basic baudrate for MIDI communications.

Definition at line 22 of file Compatibility_v2.5.h.

4.1.2.9 **#define PC ProgramChange**

Message type Program Change

Definition at line 28 of file Compatibility_v2.5.h.

4.1

/Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILib/trunk/Arduino/Compatibility_v2.5.h File

Reference `#define SysEx SystemExclusive`

21

Message type System Exclusive

Definition at line 32 of file Compatibility_v2.5.h.

4.2 /Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILibrary File Reference

MIDI Library for the Arduino.

```
#include "MIDI.h"  
#include <stdlib.h>  
#include "WConstants.h"  
#include "HardwareSerial.h"
```

Variables

- [MIDI_Class MIDI](#)

4.2.1 Detailed Description

MIDI Library for the Arduino.

Project MIDI Library

Version:

3.0

Author:

François Best

Date:

24/02/11 GPL Forty Seven Effects - 2011

Definition in file [MIDI.cpp](#).

4.2.2 Variable Documentation

4.2.2.1 MIDI_Class MIDI

Main instance (the class comes pre-instantiated).

Definition at line 18 of file MIDI.cpp.

File Reference

MIDI Library for the Arduino Version 3.0.

```
#include <inttypes.h>
#include "Compatibility_v2.5.h"
```

Classes

- struct [midimsg](#)
- class [MIDI_Class](#)

Defines

- #define [COMPATIBILITY_V25](#) 1
- #define [COMPFLAG_MIDI_IN](#) 1
- #define [COMPFLAG_MIDI_OUT](#) 1
- #define [USE_SERIAL_PORT](#) Serial1
- #define [USE_RUNNING_STATUS](#) 1
- #define [MIDI_BAUDRATE](#) 31250
- #define [MIDI_CHANNEL_OMNI](#) 0
- #define [MIDI_CHANNEL_OFF](#) 17
- #define [MIDI_SYSEX_ARRAY_SIZE](#) 255

Typedefs

- typedef uint8_t [byte](#)

Enumerations

- enum [kMIDIType](#) {
[NoteOff](#) = 0x80, [NoteOn](#) = 0x90, [AfterTouchPoly](#) = 0xA0, [ControlChange](#) = 0xB0,
[ProgramChange](#) = 0xC0, [AfterTouchChannel](#) = 0xD0, [PitchBend](#) = 0xE0, [SystemExclusive](#) = 0xF0,
[TimeCodeQuarterFrame](#) = 0xF1, [SongPosition](#) = 0xF2, [SongSelect](#) = 0xF3, [TuneRequest](#) = 0xF6,
[Clock](#) = 0xF8, [Start](#) = 0xFA, [Continue](#) = 0xFB, [Stop](#) = 0xFC,
[ActiveSensing](#) = 0xFE, [SystemReset](#) = 0xFF, [InvalidType](#) = 0x00 }
- enum [kThruFilterMode](#) { [Off](#) = 0, [Full](#) = 1, [SameChannel](#) = 2, [DifferentChannel](#) = 3 }

Variables

- [MIDI_Class MIDI](#)

4.3.1 Detailed Description

MIDI Library for the Arduino Version 3.0.

Project MIDI Library

Author:

François Best

Date:

24/02/11 License GPL Forty Seven Effects - 2011

Definition in file [MIDI.h](#).

4.3.2 Define Documentation

4.3.2.1 `#define COMPATIBILITY_V25 1`

Definition at line 30 of file MIDI.h.

4.3.2.2 `#define COMPFLAG_MIDI_IN 1`

Definition at line 33 of file MIDI.h.

4.3.2.3 `#define COMPFLAG_MIDI_OUT 1`

Definition at line 34 of file MIDI.h.

4.3.2.4 `#define MIDI_BAUDRATE 31250`

Definition at line 56 of file MIDI.h.

4.3.2.5 `#define MIDI_CHANNEL_OFF 17`

Definition at line 60 of file MIDI.h.

4.3.2.6 `#define MIDI_CHANNEL_OMNI 0`

Definition at line 59 of file MIDI.h.

4.3.2.7 `#define MIDI_SYSEX_ARRAY_SIZE 255`

Definition at line 62 of file MIDI.h.

4.3.2.8 `#define USE_RUNNING_STATUS 1`

Definition at line 44 of file MIDI.h.

4.3

/Users/franky/Documents/Dropbox/SVN/embedded/toolbox/libraries/MIDILib/trunk/Arduino/MIDI.h
File Reference 25

4.3.2.9 #define USE_SERIAL_PORT Serial1

Definition at line 40 of file MIDI.h.

4.3.3 Typedef Documentation

4.3.3.1 typedef uint8_t byte

Type definition for practical use (because "unsigned char" is a bit long to write..)

Definition at line 65 of file MIDI.h.

4.3.4 Enumeration Type Documentation

4.3.4.1 enum kMIDIType

Enumeration of MIDI types

Enumerator:

NoteOff
NoteOn
AfterTouchPoly
ControlChange
ProgramChange
AfterTouchChannel
PitchBend
SystemExclusive
TimeCodeQuarterFrame
SongPosition
SongSelect
TuneRequest
Clock
Start
Continue
Stop
ActiveSensing
SystemReset
InvalidType

Definition at line 68 of file MIDI.h.

```
68      {  
69      NoteOff      = 0x80,    // Note Off  
70      NoteOn      = 0x90,    // Note On  
71      AfterTouchPoly = 0xA0,    // Polyphonic AfterTouch  
72      ControlChange = 0xB0,    // Control Change / Channel Mode  
73      ProgramChange = 0xC0,    // Program Change
```

```

74      AfterTouchChannel    = 0xD0,    // Channel (monophonic) AfterTouch
75      PitchBend            = 0xE0,    // Pitch Bend
76      SystemExclusive      = 0xF0,    // System Exclusive
77      TimeCodeQuarterFrame = 0xF1,    // System Common - MIDI Time Code Quarter Frame
78      SongPosition         = 0xF2,    // System Common - Song Position Pointer
79      SongSelect           = 0xF3,    // System Common - Song Select
80      TuneRequest          = 0xF6,    // System Common - Tune Request
81      Clock                = 0xF8,    // System Real Time - Timing Clock
82      Start                = 0xFA,    // System Real Time - Start
83      Continue             = 0xFB,    // System Real Time - Continue
84      Stop                 = 0xFC,    // System Real Time - Stop
85      ActiveSensing        = 0xFE,    // System Real Time - Active Sensing
86      SystemReset          = 0xFF,    // System Real Time - System Reset
87      InvalidType          = 0x00     // For notifying errors
88  };

```

4.3.4.2 enum kThruFilterMode

Enumeration of Thru filter modes

Enumerator:

Off

Full

SameChannel

DifferentChannel

Definition at line 91 of file MIDI.h.

```

91      {
92      Off                = 0,    // Thru disabled (nothing passes through).
93      Full               = 1,    // Fully enabled Thru (every incoming message is sent back).
94      SameChannel        = 2,    // Only the messages on the Input Channel will be sent back.
95      DifferentChannel    = 3    // All the messages but the ones on the Input Channel will be sent
96  };

```

4.3.5 Variable Documentation

4.3.5.1 MIDI_Class MIDI

Main instance (the class comes pre-instantiated).

Definition at line 18 of file MIDI.cpp.

Index

- ~MIDI_Class
 - MIDI_Class, [6](#)
- /Users/franky/Documents/Dropbox/SVN/embedded/toolbox/MIDI.h, [19](#)
- /Users/franky/Documents/Dropbox/SVN/embedded/toolbox/MIDI.h, [22](#)
- /Users/franky/Documents/Dropbox/SVN/embedded/toolbox/MIDI.h, [23](#)
- ActiveSensing
 - MIDI.h, [25](#)
- AfterTouchChannel
 - MIDI.h, [25](#)
- AfterTouchPoly
 - MIDI.h, [25](#)
- ATCanal
 - Compatibility_v2.5.h, [20](#)
- ATPoly
 - Compatibility_v2.5.h, [20](#)
- begin
 - MIDI_Class, [6](#)
- byte
 - MIDI.h, [25](#)
- CC
 - Compatibility_v2.5.h, [20](#)
- channel
 - midimsg, [16](#)
- check
 - MIDI_Class, [7](#)
- Clock
 - MIDI.h, [25](#)
- Compatibility_v2.5.h
 - ATCanal, [20](#)
 - ATPoly, [20](#)
 - CC, [20](#)
 - MIDI_FILTER_ANTICANAL, [20](#)
 - MIDI_FILTER_CANAL, [20](#)
 - MIDI_FILTER_FULL, [20](#)
 - MIDI_FILTER_OFF, [20](#)
 - MIDI_rate, [20](#)
 - PC, [20](#)
 - SysEx, [20](#)
- COMPATIBILITY_V25
 - MIDI.h, [24](#)
- COMPFLAG_MIDI_IN
 - MIDI.h, [34](#)
- COMPFLAG_MIDI_OUT
 - MIDI.h, [34](#)
- Continue
 - MIDI.h, [35](#)
- ControlChange
 - MIDI.h, [25](#)
- data1
 - midimsg, [16](#)
- data2
 - midimsg, [16](#)
- DifferentChannel
 - MIDI.h, [26](#)
- Full
 - MIDI.h, [26](#)
- getChannel
 - MIDI_Class, [7](#)
- getData1
 - MIDI_Class, [7](#)
- getData2
 - MIDI_Class, [7](#)
- getFilterMode
 - MIDI_Class, [8](#)
- getInputChannel
 - MIDI_Class, [8](#)
- getSysExArray
 - MIDI_Class, [8](#)
- getThruState
 - MIDI_Class, [8](#)
- getType
 - MIDI_Class, [8](#)
- InvalidType
 - MIDI.h, [25](#)
- kMIDIType
 - MIDI.h, [25](#)
- kThruFilterMode
 - MIDI.h, [26](#)
- MIDI

- MIDI.cpp, 22
- MIDI.h, 26
- MIDI.cpp
 - MIDI, 22
- MIDI.h
 - ActiveSensing, 25
 - AfterTouchChannel, 25
 - AfterTouchPoly, 25
 - byte, 25
 - Clock, 25
 - COMPATIBILITY_V25, 24
 - COMPFLAG_MIDI_IN, 24
 - COMPFLAG_MIDI_OUT, 24
 - Continue, 25
 - ControlChange, 25
 - DifferentChannel, 26
 - Full, 26
 - InvalidType, 25
 - kMIDIType, 25
 - kThruFilterMode, 26
 - MIDI, 26
 - MIDI_BAUDRATE, 24
 - MIDI_CHANNEL_OFF, 24
 - MIDI_CHANNEL_OMNI, 24
 - MIDI_SYSEX_ARRAY_SIZE, 24
 - NoteOff, 25
 - NoteOn, 25
 - Off, 26
 - PitchBend, 25
 - ProgramChange, 25
 - SameChannel, 26
 - SongPosition, 25
 - SongSelect, 25
 - Start, 25
 - Stop, 25
 - SystemExclusive, 25
 - SystemReset, 25
 - TimeCodeQuarterFrame, 25
 - TuneRequest, 25
 - USE_RUNNING_STATUS, 24
 - USE_SERIAL_PORT, 24
- MIDI_BAUDRATE
 - MIDI.h, 24
- MIDI_CHANNEL_OFF
 - MIDI.h, 24
- MIDI_CHANNEL_OMNI
 - MIDI.h, 24
- MIDI_Class, 5
 - ~MIDI_Class, 6
 - begin, 6
 - check, 7
 - getChannel, 7
 - getData1, 7
 - getData2, 7
 - getFilterMode, 8
 - getInputChannel, 8
 - getSysExArray, 8
 - getThruState, 8
 - getType, 8
 - MIDI_Class, 6
 - MIDI_Class, 6
 - read, 8, 9
 - sendAfterTouch, 9
 - sendControlChange, 9
 - sendNoteOff, 10
 - sendNoteOn, 10
 - sendPitchBend, 10, 11
 - sendPolyPressure, 11
 - sendProgramChange, 11
 - sendRealTime, 11
 - sendSongPosition, 12
 - sendSongSelect, 12
 - sendSysEx, 12
 - sendTimeCodeQuarterFrame, 13
 - sendTuneRequest, 13
 - setInputChannel, 14
 - setThruFilterMode, 14
 - turnThruOff, 14
 - turnThruOn, 15
- MIDI_FILTER_ANTICANAL
 - Compatibility_v2.5.h, 20
- MIDI_FILTER_CANAL
 - Compatibility_v2.5.h, 20
- MIDI_FILTER_FULL
 - Compatibility_v2.5.h, 20
- MIDI_FILTER_OFF
 - Compatibility_v2.5.h, 20
- MIDI_rate
 - Compatibility_v2.5.h, 20
- MIDI_SYSEX_ARRAY_SIZE
 - MIDI.h, 24
- midimsg, 16
 - channel, 16
 - data1, 16
 - data2, 16
 - sysex_array, 16
 - type, 17
 - valid, 17
- NoteOff
 - MIDI.h, 25
- NoteOn
 - MIDI.h, 25
- Off
 - MIDI.h, 26
- PC

Compatibility_v2.5.h, [20](#)

PitchBend
MIDI.h, [25](#)

ProgramChange
MIDI.h, [25](#)

read
MIDI_Class, [8](#), [9](#)

SameChannel
MIDI.h, [26](#)

sendAfterTouch
MIDI_Class, [9](#)

sendControlChange
MIDI_Class, [9](#)

sendNoteOff
MIDI_Class, [10](#)

sendNoteOn
MIDI_Class, [10](#)

sendPitchBend
MIDI_Class, [10](#), [11](#)

sendPolyPressure
MIDI_Class, [11](#)

sendProgramChange
MIDI_Class, [11](#)

sendRealTime
MIDI_Class, [11](#)

sendSongPosition
MIDI_Class, [12](#)

sendSongSelect
MIDI_Class, [12](#)

sendSysEx
MIDI_Class, [12](#)

sendTimeCodeQuarterFrame
MIDI_Class, [13](#)

sendTuneRequest
MIDI_Class, [13](#)

setInputChannel
MIDI_Class, [14](#)

setThruFilterMode
MIDI_Class, [14](#)

SongPosition
MIDI.h, [25](#)

SongSelect
MIDI.h, [25](#)

Start
MIDI.h, [25](#)

Stop
MIDI.h, [25](#)

SysEx
Compatibility_v2.5.h, [20](#)

sysex_array
midimsg, [16](#)

SystemExclusive
MIDI.h, [25](#)

SystemReset
MIDI.h, [25](#)

TimeCodeQuarterFrame
MIDI.h, [25](#)

TuneRequest
MIDI.h, [25](#)

turnThruOff
MIDI_Class, [14](#)

turnThruOn
MIDI_Class, [15](#)

type
midimsg, [17](#)

USE_RUNNING_STATUS
MIDI.h, [24](#)

USE_SERIAL_PORT
MIDI.h, [24](#)

valid
midimsg, [17](#)