

<b>DESCRIPTION</b>	This function is used to handle requests for alarm registered by the <b>MIDIRequestAlarm()</b> function. <i>replyPort</i> represents the port passed by the <b>MIDIRequestAlarm()</b> function. <i>requestedTime</i> represents the time passed by the <b>MIDIRequestAlarm()</b> function. <i>actualTime</i> represents the actual time at which the alarm is sent.
--------------------	---

[illegible]

**DESCRIPTION** This function is used to handle requests for data registered by the **MIDIRequestData()** function. *replyPort* represents the port passed by the **MIDIRequestData()** function. *events* represents an array of MIDIRawEvent data. *count* represents the number of elements in *events*.

```
(*MIDIExceptionReplyFunction)(port t replyPort, int exception);
```

<b>DESCRIPTION</b>	This function is used to handle requests for exceptions registered by the <b>MIDIRequestExceptions()</b> function. <i>replyPort</i> represents the port passed by the <b>MIDIRequestExceptions()</b> function. <i>exception</i> represents the exception sent by the driver.
--------------------	--

```
(*MIDIQueueReplyFunction)(port t replyPort, short unit);
```

**DESCRIPTION** This function is used to handle requests for queue information registered by the **MIDIRequestQueueNotification()** function. *replyPort* represents the port passed by the

**MIDIRequestQueueNotification()** function. *unit* represents the serial port with which the queue is associated.

## MIDIRawEvent

**DECLARED IN**      `mididriver/midi_driver.h`

```

SYNOPSIS
    int  time;
    unsigned char  byte;
} MIDIRawEvent;

```

DESCRIPTION	
<i>time</i> is the timestamp associated with the MIDI data.	
<i>byte</i> is the actual MIDI data.	

## MIDIReplyFunctions

**DECLARED IN**      `mididriver/midi_driver.h`

```

SYNOPSIS
typedef struct {
    MIDIDataReplyFunction  dataReply;
    MIDIArmReplyFunction  alarmReply;
    MIDIExceptionReplyFunction  exceptionReply;
    MIDIQueueReplyFunction  queueReply;
} MIDIReplyFunctions;

```

**DESCRIPTION** This structure is used as an argument to the **MIDIAwaitReply()** and **MIDIHandleReply()** functions to allow an application to handle replies to requests to the MIDI driver.

# Symbolic Constants

## Clock Modes

**DECLARED IN**      `mididriver/midi_driver.h`

**SYNOPSIS** MIDI\_CLOCK\_MODE\_INTERNAL  
MIDI\_CLOCK\_MODE\_MTC\_SYNC

## Controller Definitions

**DECLARED IN**      `mididriver/midi spec.h`

**SYNOPSIS** MIDI\_TREMELODEPTH MIDI\_CHORUSDEPTH MIDI\_DETUNEDEPTH MIDI\_PHASERDEPTH MIDI\_EXTERNALEFFECTSDEPTH

(from original 1.0 MIDI spec)  
MIDI\_EFFECTS1  
MIDI\_EFFECTS2  
MIDI\_EFFECTS3  
MIDI\_EFFECTS4  
MIDI\_EFFECTS5  
MIDI\_DATAINCREMENT  
MIDI\_DATADECREMENT  
(From June 1990 spec)

Error Codes

DECLARED IN      mididriver/midi\_driver.h

SYNOPSIS

MIDI\_ERROR\_NOT\_OWNER  
MIDI\_ERROR\_QUEUE\_FULL  
MIDI\_ERROR\_BAD\_MODE  
MIDI\_ERROR\_UNIT\_UNAVAILABLE  
MIDI\_ERROR\_ILLEGAL\_OPERATION  
MIDI\_ERROR\_UNKNOWN\_ERROR

MIDI\_ERROR\_BUSY

Event Count

DECLARED IN      mididriver/midi\_driver.h

SYNOPSIS

MIDI\_MAX\_EVENT      100

Event Size

DECLARED IN      mididriver/midi\_driver.h

SYNOPSIS

MIDI\_MAX\_MSG\_SIZE      1024

Exception Codes

DECLARED IN      mididriver/midi\_driver.h

SYNOPSIS

MIDI\_EXCEPTION\_MTC\_STOPPED  
MIDI\_EXCEPTION\_MTC\_STARTED\_FORWARD  
MIDI\_EXCEPTION\_MTC\_STARTED\_REVERSE

General MIDI Constants

DECLARED IN      mididriver/midi\_spec.h

SYNOPSIS

MIDI\_RESETCONTROLLERS  
MIDI\_LOCALCONTROL

MIDI\_ALLNOTESOFF  
MIDI\_OMNIOFF  
MIDI\_OMNION  
MIDI\_MONO  
MIDI\_POLY  
MIDI\_NOTEOFF  
MIDI\_NOTEON  
MIDI\_POLYPRES  
MIDI\_CONTROL  
MIDI\_PROGRAM  
MIDI\_CHANPRES  
MIDI\_PITCH  
MIDI\_CHANMODE  
MIDI\_CONTROL  
MIDI\_SYSTEM  
MIDI\_SYSEXCL  
MIDI\_TIMECODEQUARTER  
MIDI\_SONGPOS  
MIDI\_SONGSEL  
MIDI\_TUNEREQ  
MIDI\_EOX  
MIDI\_CLOCK  
MIDI\_START  
MIDI\_CONTINUE  
MIDI\_STOP  
MIDI\_ACTIVE  
MIDI\_RESET  
MIDI\_MAXDATA  
MIDI\_MAXCHAN  
MIDI\_NUMCHANS  
MIDI\_NUMKEYS  
MIDI\_ZEROBEND  
MIDI\_DEFAULTVELOCITY

**DESCRIPTION**      These constants represent various MIDI-specified messages.

**Ignores**

**DECLARED IN**      mididriver/midi\_driver.h

**SYNOPSIS**

MIDI\_IGNORE\_START  
MIDI\_IGNORE\_CONTINUE  
MIDI\_IGNORE\_STOP  
MIDI\_IGNORE\_ACTIVE  
MIDI\_IGNORE\_RESET  
MIDI\_IGNORE\_REAL\_TIME

MIDI\_IGNORE\_CLOCK

**DESCRIPTION**      Used with the **MIDISetSystemIgnores()** function.

**Least Significant Bit for Controller Numbers**

**DECLARED IN**      mididriver/midi\_spec.h

SYNOPSIS

MIDI\_BREATHLSB  
MIDI\_FOOTLSB  
MIDI\_PORTAMENTOTIMELSB  
MIDI\_DATAENTRYLSB  
MIDI\_MAINVOLUME  
MIDI\_BALANCELSB  
MIDI\_PANLSB  
MIDI\_EXPRESSIONLSB

MIDI\_MODWHEELLSB

Masks for MIDI Status Bytes

DECLARED IN      mididriver/midi\_spec.h

SYNOPSIS

MIDI\_STATUSMASK  
MIDI\_SYSRTBIT

MIDI\_STATUSBIT

Miscellaneous

DECLARED IN      mididriver/midi\_driver.h

SYNOPSIS

MIDI\_NO\_TIMEOUT

MIDI Controller Numbers

DECLARED IN      mididriver/midi\_spec.h

SYNOPSIS

MIDI\_BREATH  
MIDI\_FOOT  
MIDI\_PORTAMENTOTIME  
MIDI\_DATAENTRY  
MIDI\_MAINVOLUME  
MIDI\_BALANCE  
MIDI\_PAN  
MIDI\_EXPRESSION  
MIDI\_EFFECTCONTROL1  
MIDI\_EFFECTCONTROL2  
MIDI\_DAMPER  
MIDI\_PORTAMENTO  
MIDI\_SOSTENUTO  
MIDI\_SOFTPEDAL  
MIDI\_HOLD2

MIDI\_MODWHEEL

Port Constants

DECLARED IN      mididriver/midi\_driver.h

SYNOPSIS

MIDI\_PORT\_A\_UNIT

MIDI\_PORT\_B\_UNIT

**DESCRIPTION**      Used to identify the port claimed for the application.