The MIDI Batch File Utility

MBATCH

Version 2.22

September 10, 1991

Music Quest, Inc.

Software License

Warranty/Disclaimer

Trademarks and Acknowledgements

Contents

Introduction

Many operating systems have batch file capabilities like the DOS or OS/2 batch file feature. Some batch file products can be added to your system to give it more powerful script routine features. The Personal REXX product is an example of a batch file add-on, although it is much more powerful than your average batch file processor.

MBATCH is a MIDI oriented batch file processor, with simple yet useful functions. MBATCH was born when we realized we had accumulated one too many MIDI hardware products with no corresponding software. For example, that rack mount Mirage always forgets how it was set up when you remove power. We wanted some simple way to send system exclusive commands to anything in my configuration, without having to run 10 programs. The result was MBATCH.

Requirements

MBATCH is a DOS based program, and it should run on any 3.X version of DOS. In terms of memory, any machine with 256K or more RAM should be adequate. MBATCH will work with any of the Music Quest interfaces.

What's in a Batch File?

An MBATCH file isn't too much different from a DOS batch file. It contains one or more statements that are interpretively executed by MBATCH. Each line of the batch file is a statement. A statement consists of a command, function name, or verb followed by zero or more operands (arguments).

```
     functionname operand1 operand2....operandn
```

Commands and function names are not case sensitive, so you can use upper/lower case at will. Operands can be separated by blanks or commas. Operands can be of three types: variables, character strings or numbers. Variables provide a powerful way to substitute other values into a statement (a varaible can contain a character string or number). Character strings consist of 1 or more non-white space characters and are typically used to specify things like file names. Numbers can be entered in any of three forms: hex, decimal or octal. A hex number starts with the characters 0x, as in 0xFF. A decimal number starts with the digits 1-9, as in 24. An octal number starts with a 0 with the second digit being 1-7.

In summary:

```
     c:file.any = character string
     0xAC = hex
     39 = decimal
     017 = octal
```

You build a batch file by combining various statements into some useful sequence, such as uploading a patch bank from a synthesizer. The MBATCH package includes several useful batch files that you can use as models for creating more batch files.

Learning How to Use MBATCH

You will find that the fastest way to learn how to use MBATCH is to look at the example batch files. If you don't understand something in a batch file, then look up the particular statement in this document. More than likely, you will be able to create new batch files by starting from the ones that are included with the package.

Running MBATCH

The MBATCH program is run from the DOS prompt:

```
     mbatch batchfilename
```

Here, batchfilename is any full file name that may include a path and a file extension. For example,

```
     mbatch c:dx7.mbt
```

starts MBATCH and tells it to interpretively execute the batch file c:dx7.mbt

Variables and the SET Command

Variables allow you to manipulate data in various ways. For example, you can use variables to accomplish basic mathematic operations, modify MIDI data (such as System Exclusive data), or parameterize a batch file. With the SET command, you can define a variable name with up to 31 characters. The first character of the variable name must be non-numeric (anything other than a number). For example:

```
     SET midichannel 0
```

defines the variable 'midichannel' and sets its value to 0.

One of the advantages of variables is that you can use them just about any place where an operand is required. You can assign any number, character string, or quoted string to a variable. Note that the length of a string is limited to 32 characters.

The BUFFER Variable

There is one special variable that you will want to know about, namely the BUFFER variable. The BUFFER variable refers to MBATCH's System Exclusive data buffer. The BUFFER is typically more than 64K bytes long, so there must be some way to specify what byte of BUFFER is being referenced. An example will illustrate how this is handled.

```
     SET buffer[0] 0xF0
```

This statement sets the byte at offset 0 within BUFFER to the value 0xF0. It illustrates how the square brackets (ala C language syntax) are used to specify an index into the BUFFER array.

Commands, Functions, and Verbs

Here is a description of all of the commands currently supported by MBATCH. The commands are arranged according to category. Where appropriate, an example illustrating the use of each command is included.

Miscellaneous Commands

Comments

Syntax: ;anything

Any statement or line that starts with a semicolon is treated as a comment, and ignored by MBATCH.

```
     ;A comment statement
```

Controlling Batch File Listing

Syntax: echo {on | off}

The echo command works just like the DOS batch file echo command. Ordinarily, each statement of the batch file is displayed. The command echo off suspends the display of batch file lines, while echo on resumes the display.

Help Screen

Syntax: help

The help command prints a help screen with each possible command listed. This provides a handy way to remember possible command names when using MBatch interactively.

Commands that Assign Values to Variables

Set (Define) a Variable

Syntax: set varname value

The set command assigns a value to a variable. The value can be an integer number or character string of up to 31 characters. Numeric variables are maintained as 32-bit values.

Get Data from User

Syntax: input prompt varname

The input command displays the prompt and waits for the user to enter data. The prompt can be anything, including a quoted string. The data is assigned to the variable varname. This is similar to the BASIC input command.

```
     input "Enter MIDI channel number: " midichannel
```

Arithmetic and Logical Commands

These commands allow you to alter the value of a variable. Their syntax is somewhat similar to many assembler languages in that they consist of a command (mnemonic) and two operands. The first operand is also the destination of the result. Note that all arithmetic operations are performed using 32-bit arithmetic.

Add to Variable

Syntax: add varname value

The value is added to the variable varname.

Subtract from Variable

```
Syntax:   subtract varname value
          sub varname value
```

The value is subtracted from the variable varname.

And to Variable

```
Syntax:   and varname value
```

The value is anded to the variable varname.

Or to Variable

```
Syntax:   or varname value
```

The value is orred into the variable varname.

Exclusive Or to Variable

```
Syntax:   xor varname value
```

The value is exclusive orred into the variable varname.

Execution Control Commands

The execution commands provide loops and conditional execution capability.

Logical Expressions

Control statements use logical expressions to determine which statements are interpreted. A simple logical expression consists of two values and a logical operator:

```
     value1 logical-operator value2
```

Values can be character strings or numbers. The supported logical operators are:

Operator	Meaning
==	equal to
!=	not equal to
>	greater than

Operator	Meaning
<	less than
>=	greater than or equal to
<=	less than or equal to

The result of a logical expession is "true" if the condition is true; otherwise, the result is false. Two simple logical expressions can be combined to form a compound logical expression. The compound operators are:

Operator	Meaning
&	and
|	or

Here are some syntactically valid logical expressions:

Expression	Meaning
x > 6	x greater than 6
x > 6 & x < 12	x greater than 6 and x less than 12
string != abcd	variable string not equal to character abcd
string == abcd | string == xyz	variable string equal to character string abcd or xyz

While-end Loop

```
Syntax:   while logical-expression
          statement(s)
          end
```

The while-end construct implements the traditional while-loop. The three operands of the while command form an expression that controls the loop. If the expression is false, the loop terminates. Value1 and value2 can be variables, numbers, or character strings.

Example:

```
          set i 0
          while i < 16
            print i
            add i 1
          end
```

will print the numbers from 0 through 15.

If-else-endif

```
Syntax:   if logical-expression
          statement(s)
          else
          statement(s)
          endif
```

The if-else-endif construct implements conditional execution. If the logical expression is true, the statements up to the else are executed; otherwise, the statements between the else and endif are executed. If there is no else clause, the else command may be omitted, yielding an if-endif construct. The endif command MUST always be included.

Example:

```
          input "Enter a number: " n
          if n > 5
            print n "is greater than 5"
          else
            print n "is less than or equal to 5"
          endif
```

This example asks the user for a number, and responds by telling the user if the number is greater than or less than equal to 5.

Commands for Manipulating MIDI Data

Send Data Bytes

Syntax: data [byte1] [byte2]....[byten]

The data command causes all of the operand data bytes to be sent to MIDI-out. No error checking is performed.

Start System Exclusive

Syntax: sysex [byte1] [byte2]....[byten]

The sysex command is probably the most useful command in the MBATCH vocabulary. Effectively, MBATCH sends a 0xF0 (start of system exclusive) to MIDI-out followed by what ever data bytes appear on the rest of the statement. MBATCH doesn't validate the operands, it merely sends them to MIDI-out, so you must pay attention to the operands.

The following sends a bulk dump request to a DX-7:

```
     sysex 0x43 0x20 9 0xF7
```

Notice that an end of exclusive, 0xF7, was included to complete the system exclusive. You can explicitly write the 0xF7, or you can use the eox command to do it for you. For example, the following two statements are functionally the same as the single statement above:

```
     sysex 0x43 0x20 9
     eox
```

Long system exclusives can be continued over multiple statements by using the data command. Using the data statement, the above could be written:

```
     sysex
     data 0x43 0x20 9
     eox
```

In summary, you can use these three statements to send just about anything imaginable to your MIDI hardware.

Send End of Exclusive

Syntax: eox

This command causes a 0xF7 to be sent to MIDI-out, effectively terminating any open system exclusive.

Flush System Exclusive Receive BUFFER

Syntax: flushbuffer

MBATCH implements a 64K byte sysex BUFFER for storing incoming system exclusive data. The flushbuffer command clears the BUFFER and prepares the MIDI-interface to receive new system exclusive data.

Receive System Exclusive Data

Syntax: receive [n]

The receive command stores received system exclusive data in the BUFFER. The command completes after 'n' system exclusive units have been received. The single operand specifies the number of system exclusive units that are to be saved in the file. If 'n' is omitted, the value 1 is assumed. A system exclusive unit is one complete system exclusive message, starting with 0xF0 and ending with 0xF7. Many MIDI devices (e.g Emu Proteus) can transmit multiple system exclusive units in response to a single request. For example, the Emu Proteus sends 64 units in response to a "dump factory presets" request.

Example: receive 64

This statement causes MBATCH to wait until 64 system exclusive units have been received into the sysex BUFFER.

Save BUFFER to File

Syntax: save filename

The save command writes the contents of the sysex BUFFER to the designated file. The filename operand specifies the full name of the file. You can specify any path, name and extension.

Receive System Exclusive Data and Save

Syntax: receivefile filename [n]

The receivefile is the logical combination of the receive and save commands. After receiving 'n' system exclusive units, the contents of the BUFFER are saved in the specified file. The filename operand specifies the full name of the file. You can specify any path, name and extension. The second operand specifies the number of system exclusive units that are to be saved in the file. If n is omitted, the value 1 is assumed. A system exclusive unit is one complete system exclusive message, starting with 0xF0 and ending with 0xF7. Many MIDI devices can transmit multiple system exclusive units in response to a single request. For example, the Emu Proteus sends 64 units in response to a "dump factory presets" request.

Example: receivefile c:factory.pro 64

This statement causes MBATCH to wait until 64 system exclusive units have been received into the sysex BUFFER. When all 64 have arrived, they are written to the file c:factory.pro as a continuous stream of data.

Send System Exclusive Data from a File

Load a File into the BUFFER

Syntax: load filename

The specified file is loaded into the sysex BUFFER. The file must be in MIDIEX format (a MIDIEX file contains one complete Sysex, starting with a 0xF0 and ending with a 0xF7).

Send Contents of BUFFER

Syntax: send

The current contents of the sysex BUFFER are sent to MIDI-out.

Load and Send File

Syntax: sendfile filename

The sendfile command is the logical combiantion of the load and send commands, and the complement to the receivefile command. It reads the contents of the specified file into the sysex BUFFER and then sends the data to MIDI-out. It can read/send any file that was saved via receivefile. The file format employed is upward compatible with the popular MIDIEX format, so you can use sendfile to send anything that was captured using MIDIEX.

Receive Time Out Value

Syntax: timeout tttt

When you use the receivefile command, it will wait until the specified number of sysex units arrive. If you make a mistake with a preceding sysex request, the data may never arrive. To avoid program hangs, the receivefile command will give up if it does not receive any data within the time out value. You control the time out value through the timeout command. The value of the tttt operand specifies the time out value in terms of seconds.

Example: timeout 10

The receivefile command will terminate if no data is received for 10 seconds.

Receive Status Pacing

Syntax: pacing pppp

While the sendfile command is running, it reports status about how many sysex units and data bytes have been received. The pacing value, pppp, determines how frequent status is updated. The pppp value specifies that the status is to be updated every pppp bytes. If you never specify a pacing value, it defaults to 512.

The pacing value has a not-so-obvious side effect from which its name was derived. Updating the status consumes CPU cycles, which inherently slows down the effective of rate of data transfer to MIDI-out. Thus, you can use it to pace the rate of outgoing data for devices that cannot receive data at full MIDI speed. Setting a pacing value of 1 will cause the status to be updated every byte, and it will provide the slowest data transfer.

Delay Timer

Syntax: `delay mmmm`

The delay command simply pauses for mmmm milliseconds. Some MIDI hardware is timing sensitive, so you can use the delay command to slow down transactions. Be aware that the PC timer is used to measure delays. Since the PC timer is in units of approximately 55 milliseconds, the total delay time will be rounded to the nearest 55 millisecond unit.

Send Note On

Syntax: `noteon ch nn vv`

The noteon command sends a note on message for channel ch (0-15), note nn (0-127), at velocity vv (0-127).

Send Note Off

Syntax: `noteoff ch nn vv`

The noteoff command sends a note off message for channel ch (0-15), note nn (0-127), at velocity vv (0-127).

Send Program Change

Syntax: `programchange ch pp`

The programchange command sends a program change message for channel ch (0-15) to change to program pp (0-127).

Send Control Change

Syntax: `controlchange ch cc vv`

The controlchange command sends a control change message for channel ch (0-15) to change controller cc (0-127) to value vv (0-127). For example, controller 64 is the sustain controller. Thus,

```
     controlchange 0 64 127
```

would cause the sustain controller on channel 0 to be turned on.

Send Pitch Bend Message

Syntax: pitchbend ch bbbb

The pitchbend command sends a pitch bend message on channel ch (0-15) with bend value bbbb (0-0x3FFF). Note that 0x2000 is the center value (i.e no bend). Thus the value 0 is the maximum bend down and 0x3FFF is the maximum bend up.

Send General MIDI Message

Syntax: midimessage rs b1 [b2]

The midimessage command can be used to send any message. Other MBATCH commands allow you do cover most MIDI messages, but this command gives yo the most control. The rs byte specifies the running status byte, while b1 and b2 are the data bytes.

Example: midimessage 0x90 36 64

This sends a note on (0x90) for channel 0, note 36, at velocity 64. Note that the statement:

```
     noteon 0 36 64
```

accomplishes the same result, but is somewhat easier to read.

Displaying Data

MBATCH includes some basic functions for displaying data. These can be especially useful for dumping out the contents of system exclusives or other MIDI data.

Print (Display)

Syntax: print item1 [item2...itemn]

The print command displays one or more data items (just like the BASIC print statement). The data items can be numbers, character strings or quoted strings. A blank is inserted between each data item, and a new-line is added after the last data item.

Example:

```
          print buffer[0] buffer[1] buffer[3] buffer[4]
```

prints out the first 4 bytes of the system exclusive BUFFER.

Set Print Format

Syntax: printformat format

This command allows you to tell MBATCH how you want printed data to be formatted. The format string is a C language format string that would be typically used with the printf function. Note that when printing numeric values, the format string must include an 'l', because all numeric values are maintained as long integers (32-bit values).

For those not familiar with C, the here is a list of useful formats. Assume that your are printing the value 0xF7.

String	Results	Comments
0x%02lX	0xF7	this is the default format
%ld	247	print as decimal digits, no leading zeroes
%02lX	F7	an alternative hexadecimal format

Typical Uses of MBATCH

Probably the most frequent use of MBATCH will be to retrieve system exclusive data from a synth or to send system exclusive data to a synth.

Uploading Sysex Data

With most synths (Roland products being an exception), you can upload sysex data by sending a single request to the synth and recording what the synth sends back. This works well with most Yamaha equipment.

The general form of a batch file for uploading and saving sysex data is:

```
     flushbuffer
     timeout
     sysex
     receivefile
```

The flushbuffer command clears out any data that may be in the receive buffer. The timeout command establishes the time interval before a time out condition is declared. The sysex command sends the request to the synth, and the receivefile command writes the received data to a file.

For a specific example, look at the dx7.mbt file.

Downloading Sysex Data

The general form of a batch file for downloading sysex data that has been saved in a file is:

```
     pacing
     sendfile
```

The pacing command sets the status update frequency, which in turn controls the rate of data transfer. The sendfile command transmits the designated file.

The dx7.mbt file shows a specific example of this general form.

Using MBATCH Interactively

Even though MBATCH was specifically designed to interpretively execute batch files, it can be used interactively from the keyboard. You will notice that when you use it from the keyboard MBATCH always echoes each completely entered statement.

To start MBATCH for keyboard use, enter the following at the DOS prompt:

```
     mbatch con
```

Then, type an entire statement and press ENTER. MBATCH echoes the statement and executes it. When you are finished, type a control-C (which will echo as ^C) press ENTER and answer Y to the question 'Terminate MIDI Batch File Execution (y/n)?'