The Hybrid Music System for the BBC Microcomputer

USER GUIDE

First published 1988

Hybrid Technology Limited
Unit 3, Robert Davies Court
Nuffield Road
CAMBRIDGE
CB4 1TP

Issue 2

Written by Chris Jordan

Contents

1 Introduction

The Music 2000 Interface links MIDI-compatible instruments to the Hybrid Music System to expand the range and number of sounds available. It works alongside the Music 5000 synthesiser, and, if fitted, the Music 4000 keyboard, both of which continue to operate as before.

AMPLE is designed to be largely independent of the musical output device in use, so it's music control facilities, including AMPLE notation, Staff Editor, Recorder and all AMPLE Nucleus words, operate equally on the Music 5000 and through the Music 2000. The part of AMPLE that is specific to the Music 5000 - the module containing sound words for controlling voices and making instruments - has a direct counterpart for the Music 2000 on the Studio 5000-2 System disc, and the two operate simultaneously.

This user guide is in two parts. The tutorial section describes the use of the Music 2000 with any MIDI-compatible instrument, complementing the music 5000 user guide in content, and introduces the more advanced applications that are possible with instruments that have further MIDI facilities. The Reference section contains detailed descriptions of all new words, and is designed to give further explanation of the basic words, and as the primary source of information on those words only required for advanced applications. Its contains many programming examples that use words only described in the AMPLE Nucleus Programmer Guide, in addition to the Music 2000 word that is the subject of the example. You can use these example definitions, as given, as applications in your own projects without having to know the words used, but you will need the Programmer Guide for a full understanding and the ability to program your own applications.

Part 1 - General

2 Introduction to MIDI

Though MIDI stands for 'Musical Instrument Digital Interface', it was originally designed simply to send performance information from one music keyboard to another, and to allow drum machines and sequencers to synchronise with each other so they would play along in time. Indeed not much more was foreseen by its designers. Hence, MIDI provides effective remote control of the sound-generating elements of many electronic musical instruments, and hence allows them to be used as additional output devices in a computer music system. It is for this that Hybrid Music System, through the Music 2000 Interface, uses MIDI.

'Standard' is a word that is often misunderstood, and rarely more so than in its application to MIDI. MIDI is a specification of which a certain central part should be met by any instrument describing itself as MIDI-compatible. Two such instruments will be able to co-operate in the functions of this 'common MIDI', and may or may not be able to co-operate on higher levels depending on how much of the MIDI specification they adhere to.
Naturally, while one instrument may have more or less facilities than another, no 'standard' can make them 100% compatible. Rather, MIDI makes it more likely that similar facilities of different instruments can be used co-operatively, where it is useful to do so.

Since MIDI is designed for remote substitution of instruments' local physical controls, it is oriented not to general-purpose sound-making elements, but to specific input devices such as the music keyboard, that is, not to 'notes' but to 'keys'. Hence, the computer can play any MIDI instrument through the depression of the imaginary keys, reproducing anything that could be played on the keyboard, but not other things, such as two simultaneous notes of the same pitch. In the case of drum machines and sequencers - devices that can perform from internally-stored scores or 'sequences' - the computer can provide the start/stop/continue controls, and provide a 'timing clock' to drive the external part in synchronisation with the internal ones.

The Hybrid Music System applications of these 'common MIDI' functions are described in Chapter 3, 'Playing voices' and Chapter 4, 'Controlling sequencers'. Any instrument of the appropriate type is suitable, and no specialist knowledge of MIDI or the instrument's specification is required. At this level, AMPLE makes MIDI very easy to use through its integration of MIDI with the general system facilities, including music notation.

The contents of Chapter 5, 'Further applications' apply only to those instruments that meet further sections of the MIDI specification. You may need detailed documentation on the instrument, including its MIDI Implementation Chart, to find out whether certain facilities are available, and how to invoke them. AMPLE's facilities for modifying and extending the existing MIDI control options through user programming provides total access to any instrument's MIDI facilities, including ones outside the MIDI specification.

3 Playing voices

The **voice** is the basic note-playing unit of any musical instrument, and MIDI-compatible instruments allow their voices to be controlled by the Music 2000 Interface through a 'MIDI in' socket on the instrument. AMPLE then lets you direct any voice or part of a piece to perform on the instrument's voices.

starting the system

Make sure the Music 2000 Interface is installed as described in the Installation Guide, and the Studio 5000-2 disc is in the drive. To start the system, press BREAK while holding SHIFT. After a moment, the Main Menu will appear. Press TAB to enter command mode.

connecting the instrument

Any of the three MIDI output lines can be used for the instrument, but for the moment connect a MIDI lead between its 'MIDI in' socket and the Music 2000's 'OUT 1' socket.

Make sure the instrument is turned on and its sound is audible, and enter:

```
READY
1 VOICES MIDIV
C^
```

This should play a short middle C on the instrument. The top light on the Music 2000 should flash as the note starts, and flash again when it stops.

calling up MIDI voices

MIDIV is the instruction that makes the current AMPLE voice or voices be MIDI voices. Any complete piece can have Music 5000 voices on one of its parts replaced by MIDI voices using MIDIV in a simple command. As an example, load the example program 'manor' from the Studio 5000 System Disc, by inserting the disc in to the selected drive, pressing f9 to call up the jukebox, and selecting it as normal.

After the music starts to play, to change part 3 from the Music 5000 instrument 'elecorgan' to the MIDI instrument, enter

```
3 SHARE MIDIV
```

You should hear the change immediately. You may like to select an appropriate sound on the MIDI instrument, and adjust its volume to suit.

To return to 'elecorgan', enter:

```
elecorgan
```

To direct the bass part, part 4, to the MIDI instrument, enter:

```
4 SHARE MIDIV
```

more than one part

You can send more than one part to the same MIDI instrument provided they do not overlap, that is, they remain in separate pitch ranges. In this example, to 'merge' parts 3 and 4 on the MIDI instrument, enter:

```
3 SHARE MIDIV
4 SHARE MIDIV
```

If your instrument has a 'split keyboard' facility, you should be able to choose a separate sound for the two parts, after first having set the split point accordingly.

adding MIDI voices to a program

To permanently add MIDI voices to a program, so they are used automatically when the piece is run, we add a new 'mix' word to the program. For convenience, we use mix number 9, 'mix9'. This MIDI mix word specifies which parts are to have MIDI voices, and does not affect Music 5000 voices on other parts.

A ready-made definition of 'mix9' is provided as a text file on your Studio 5000-2 disc. To add it to the current program, for example 'manor', make sure the Studio 5000-2 disc is in the selected drive, and enter:

```
*EXEC mix9
```

Once the operation is complete, the word of 'mix9' is part of the program, and may be edited to your requirements like any other

word. Select 'Notepad' from the %in Menu, and call up mix9 for editing by entering:

```
  "mix9"GET
```

You will see the following on the screen:

```
%5 SHARE  8 VOICES MIDIV
% 1 MIDILINE  1 MIDICHANNEL
% 0 MIDIPROGRAM
%6 SHARE  8 VOICES MIDIV
% 2 MIDILINE  1 MIDICHANNEL
% 0 MIDIPROGRAM
%7 SHARE  8 VOICES MIDIV
% 3 MIDILINE  1 MIDICHANNEL
% 0 MIDIPROGRAM
%8 SHARE  7 VOICES MIDIV
% 1 MIDILINE  2 MIDICHANNEL
% 0 MIDIPROGRAM
%9 SHARE  1 VOICES MIDIRT
% 3 MIDILINE
PNUM SHARE
```

This set of instructions allows for a wide choice of MIDI voice arrangements. The instruction lines are currently treated as comments, since they are preceded by % signs on the left, but we activate any one of them by removing the % and MAKEing the word. This definition is rather like a partly-filled-out form which you complete to make the MIDI voice arrangement you require.

For this example, we need to add MIDI voices on part 3. To make the appropriate 'mix9', remove the first % by typing a space on it, and change the digit ’5' to '3'. (Don't remove any other % signs, since none of the other lines are required.) Now MAKE the word.

We can now play the ’MIDI organ' version of the piece by entering:

```
  "1234-9abcad”PLAY
```

The digit '9' invokes the mix9 we have created. To include mix9 in the piece's PLAY instruction, enter

```
  "RUN"GET
```

add the 9 after the '-' on the bottom line, and MAKE the word. Now when you enter RUN to play the piece, it will automatically use the MIDI instrument for part 3.

adding a MIDI mix to other programs

To add a MIDI mix to a program, the program must use a PLAY instruction with section letters. If the program uses 'part1', 'part2' etc., these can easily be changed as follows:

```
"part1" "part1a" RENAME
"part2" "part2a" RENAME
```

When you start a new program that may need to use MIDI voices, you use the EXEC command to add mix9 immediately when you start.

In our example, part 3's Music 5000 voices, assigned in the word 'mix', are replaced by the MIDI voices when mix9 is carried out. When you create a new piece, you would not normally include Music 5000 voices on a part to which 'mix9' assigned MIDI voices. Music 5000 voices remain independent of MIDI voices when the two kinds are used in a piece, so for example, you use the Mixing Desk as normal to adjust Music 5000 voices. MIDI voices do not appear on the Mixing Desk, and are not affected by its settings. Music 5000 sound words, such as those used in Music 5000 instruments, have no effect on MIDI voices, and the MIDI sound words discussed below have no effect on Music 5000 voices.

instrument sound selection

The stored instrument sounds of a MIDI instrument are called 'programs' in MIDI terms. They can be selected using the instrument's front panel controls, or using the word MIDIPROGRAM. To demonstrate this, set the piece playing with RUN and enter

```
3 SHARE 0 MIDIPROGRAM
```

You will see and hear the current sound program change to number 0, unless of course it was already 0. Try some other numbers (it is not necessary to type '3 SHARE' again, only the number followed by MIDIPROGRAM, unless you RUN again).

Note that you may have to set 'program change enable' or some such on your instrument before MIDIPROGRAM will take effect.

You may find that the MIDIPROGRAM number does not correspond directly to the number displayed on the instrument, for example, 0 MIDIPROGRAM may call-up sound program 1. This is inevitable with instruments that use a numbering scheme other than MIDI's own, so you may have to convert the numbers yourself. The Dictionary of Words gives examples of user-defined versions of MIDIPROGRAM that work in alternative numbering schemes.

stored instrument sound selections

At the moment, whatever sound or 'program' is selected on the instrument will be used when the piece plays. To make sure that the piece plays with the correct sound each time, you can add a MIDIPROGRAM instruction to select the sound automatically. This instruction is already provided in 'mix9'.

Using the existing example program with mix9 added, enter

```
"mix9"GET
```

to get mix9 into the Notepad. Now remove the % sign at the start of the third line from the top, so that the MIDIPROGRAM instruction becomes active. Replace the 0 with the sound program number of your choice, and MAKE the word.

Now when you type RUN to play the piece, the sound program number you specified will be automatically selected on the instrument.

music notation

All the elements of standard AMPLE notation, whether written directly in text form, created on the Staff, or recorded, are reproduced by MIDI voices, with two exceptions. Firstly, MIDI does not cater for slurred notes, so they are ignored. Secondly, dynamic level is expressed as key velocity, so only if the receiver is velocity-sensitive will the effects of changing level be heard.

Whereas the range of dynamic level (=L) on a Music 5000 voice is up to 64, a MIDI voice will respond up to 127. This means that the default level, 64, is in the middle of the range, and, for example, there is no need to reduce the =L value before using accents ('). When scoring music to take advantage of the additional range, remember that it will not have the same effect on a Music 5000 voice.

Many instruments allow the sound to be considerably different for soft (low velocity) and hard (high velocity) notes, not just in loudness. This gives the opportunity to use different sounds for two or more parts played on a single MIDI instrument, since each part may receive a contrasting dynamic level in the score.

limitations

MIDI itself does not allow for a unison (two simultaneous notes of the same pitch), since this is impossible to play on a keyboard. Some MIDI instruments do allow it, but give no control over which of the two notes, which may have different velocities, is released first. The remaining instruments do not allow unisons at all, and usually leave a 'hole' in the music when one arises.

This can happen unexpectedly in a chord sequence which has the same pitch in different voices on adjacent beats - a 'sequential unison' - since the problematic parallel unison can arise momentarily in passing. For example, in the sequence

```
C(EG) E(FB)
```

for an instant, the E plays on voices 1 and 2. One possible solution is to re-write the notes on different voices, for example:

```
C(EG) F(eB)
```

Another is to insert a separate rest, either manually, or through the use of a -ve Len setting, say, -2 Len.

A separate problem arises from the slowness of response to 'note-off' messages of some MIDI instruments, causing the next 'note-on' to be ignored if it follows immediately and is on the same key. From AMPLE, the result is the loss of repeated notes, in sequences such as:

```
CCCCC
```

Again, the cure is to add a -ve Len setting, simulating the shortest gap that would be produced on a physical keyboard.

using two instruments

A second instrument may be used alongside the first to play a second part of the music. This allows a greater number of voices, different sound programs on the two parts, and complete independence of the parts so that, for example, two notes of the same pitch could sound together.

Connect the second instrument to the Music 2000 Interface at 'OUT 2'. To test it, enter

```
READY
1 VOICES MIDIV
2 MIDILINE
C^
```

The middle light should flash at the start and end of a middle C note playing on the second instrument. The MIDILINE instruction directs the voice to output line 2.

To send one of the piece's parts to the second instrument, you include the second SHARE line (on the fourth line of the 'mix9' word) and the line below it, containing the MIDILINE instruction. Call 'mix9' into Notepad, remove the % signs from lines 4 and 5, and enter the number of the part before SHARE on line 4. Make the word.

Now, when you play the piece, the two MIDI parts will play independently on the two instruments. You can use MIDIPROGRAM as before to select different sound programs, and introduce line 6 of 'mix9' to add an automatic program selection for the second MIDI part.

A third instrument may be connected at 'OUT 3', and the third SHARE line added to 'mix9' to send a part to it. The MIDILINE below is used, as before, to direct the voices to the new output line.

more than three instruments

By the time you have added three MIDI parts to a piece which already uses most of the Music 5000 voices, you may well be nearing the limit of available memory and processing power.

However, some applications will require more than three instruments. The fourth and subsequent instruments can be accommodated on the same output lines as the first three by connecting each additional instrument to the 'MIDI through' socket of an existing instrument. When two instruments share the same line, the information is carried along separate MIDI channels within the line - notes directed to a particular channel will only be received by an instrument set to receive on that channel, so each instrument on the line can receive a separate sequence of notes through a unique channel.

To control separately two instruments on the same line, you set them to receive on channels 1 and 2 respectively. Initially after

power-on, each instrument receives on all channels - it is in the so-called 'omni on' mode – so both instruments on the same line must have 'omni' set to 'off' as well as having their receive channels set. See the instrument's manual for how to do this. (It is also possible to set 'omni off' from the program – see MIDICONTROL for details.)

In the AMPLE program, the word MIDICHANNEL is used to direct voices to a particular channel on the line set with MIDILINE. The fourth SHARE in 'mix9' already has the instructions to use channel 2 on line 1.

Once channels are set up in this way, they operate just like separate lines as far as voice control messages are concerned. Remember that a channel on one line is completely separate from the same-numbered channel on a different line. To make this clear, the term 'line channel' is used to describe a particular channel on a particular line.

alternative voice set-ups

The 'mix9' definition supplied on the disc is suitable for a wide range of pieces. However, you may want to design your own arrangement of voices, for example, one with more MIDI parts. You are quite free to do this, and to make multiple mixes and sub-mixes for MIDI voices in exactly the same way as for Music 5000 voices. Here are some guidelines:

* nine parts are available, when using the PLAY instruction
* each part can have up to twelve voices (1-12), each of which can be MIDI, music 5000 or unused
* 32 MIDI voices are available in total
* MIDIV uses values of 1 for MIDILINE and MIDICHANNEL unless these settings are included

Chapter 5, 'Further applications', suggests further ways to use MIDI voices and extra voice facilities of instruments.

4 Controlling sequencers

So far we have seen how to use ordinary MIDI instruments that play sound in response to their own music keyboard, if they have one, and to key-press information sent by MIDI. A different type of device is the sequencer - this has an internally stored score or 'sequence' which it plays, on command, on the voices of an ordinary instrument to which it is linked. (By this definition, the computer is also a sequencer, but the term is usually reserved for simple 'dedicated' sequencers with limited facilities.) A drum machine has a sequencer and instrument combined in a single unit.

The 'common MIDI' facilities allow all sequencer-type devices to be synchronised to the computer so that their own internal musical parts play along in time with those in the computer. Further facilities, not found on all devices, allow more flexible control of the playback sequence, and in the case of drum machines, direct access to the percussion voices. These are described in Chapter 5, 'Further applications'.

Most of the functions of a sequencer are usually available on the computer itself, so sequencers are not used with computers as often as are drum machines. The following descriptions refer to a drum machine, though they apply equally to any MIDI sequencer-type device.

connecting the drum machine

Any of the Music 2000's output lines may be used for a drum machine, but it is conventional to use 'OUT 3', leaving 'OUT 1' and 'OUT 2' free for ordinary instruments. Connect 'OUT 3' to the drum machine's MIDI 'in' socket.

selecting external synchronisation

Before your drum machine will synchronise with the computer, it must be set to receive timing information from MIDI instead of from its own tempo generator. This selection is usually labelled 'external sync', 'MIDI sync on' or similar – see the drum machine's own manual for details.

Once external sync is in use, the drum machine will not normally play except under the control of the computer. To use it

independently, you must return it to 'internal sync', 'MIDI sync off' etc.

running the drum machine

To set the drum machine playing under control of the computer, enter:

```
READY
1 VOICES MIDIRT
3 MIDILINE
X
```

The drum machine will start as soon as you press RETURN after the X. The lower light on the front of Music 2000 will be on continuously, since the timing clock is being sent continuously.

To stop the drum machine, you can either press ESCAPE, or enter:

```
^
```

On most drum machines, you can still use the start/stop switch on the front panel when using external sync.

As the drum machine plays, it will follow changes in tempo. Try entering the following examples:

```
150=T                  % set to 150 beats per minute

64 16 -T //// //// //// ////   % gradually slow to half tempo

64 4  +T ////              % rapidly speed-up to double tempo
```

(Note that +T and -T may be used as a pair with the same first number, to move a new tempo and return precisely to the original. See =T, +T and -T in the Music 5000 User Guide for details.)

adding the drum machine to a piece

MIDIRT is the instruction that provides control over the drum machine. It makes the current voice a 'real-time voice' - a special kind of voice which sends the clock and 'start'/'stop' messages instead of the usual notes.

The 'mix9' word already includes the MIDIV instruction as a comment, ready to be included fully to add the drum machine to the piece. If your piece does not include a 'mix9' word already, add

it by entering:

```
*EXEC mix9
```

Again, you can use 'manor' as an example.

Select Notepad from the Main Menu, and enter

```
"mix9"GET
```

to call the mix word into the editor. Locate the line containing 9 SHARE 1 VOICES MIDIRT, near the bottom of the screen, and remove the % sign at its start. MAKE the word.

Next, we add a very simple 'clock part' on player 9, containing just a single X to make the drum machine start at the start of the piece. This part is a word whose definition can be entered directly at the % prompt:

```
"part9a" [ X ]
```

Finally, the PLAY instruction should include part9 and mix9. As before, you can enter this as a command, or put it in the RUN word.

```
"12349-9abcad"PLAY
```

If you want to play the piece without the drum machine, remove the first '9' from the PLAY instruction, leaving the second one so that any other MIDI voices will be retained.

starting late

In the example above, the clock part had an X in section 'a', and since section 'a' was the first section in the PLAY instruction, the drum machine came in at the beginning of the piece.

You can make the drum machine start at any other section, so that, for example, it does not come in until after a quiet introduction. To arrange this you just add the appropriate section for part 9, not forgetting to remove the previous one, for example:

```
"part9a"DELETE
"part9b" [ X ]
```

This will cause the drum machine to start on section b.

(It doesn't matter if the X section is called up again later in the piece, because once the drum machine is started, X has no

effect.)

starting and stopping

You can also stop the drum machine in the music, by including a ^ (rest) in the clock part. For example, to make it stop at the beginning of section 'c', you would add:

```
"part9c" [ ^ ]
```

The next X on part 9 will make it continue playing from the point at which it stopped.

Your piece could include a full set of sections for part 9, each one containing an X or ^ to indicate whether the drum machine was required or not, for example

```
"part9a" [ ^ ]                  % rest in section a
"part9b" [ X ]                  % play in section b
"part9c" [ X ]                  % play in section c
"part9d" [ ^ ]                  % rest in section d
...
"129-9abcad"PLAY
```

With the clock part included in every section letter like this, you can freely re-order the sections. Don't forget, though, that the drum machine part will not be re-ordered but merely stopped and restarted. To control the order of the drum machine's part's sections, you need to use the more advanced facilities described in Chapter 5, 'Further applications'.

independent starting and stopping

You would normally program your drum machine with the correct number of bars so it stopped naturally at the end of the piece. An alternative is to add a section for part 9 only, to stop it when the last main part section has finished, for example:

```
"part9s" [ ^ ]
"12349-9abcads"PLAY
             |
              stop drum machine
```

This can be particularly useful if you are using a single rhythm repeated throughout as you try out different numbers of sections and different lengths.

As a further refinement, you can add a clock-part-only start

section, giving the ability to mark start and stop points in the PLAY sequence independently of the sequence of main parts.

```
"part9r" [ X ]
"part9s" [ ^ ]
...
"12349-9arbcasd"PLAY
         |   |
         |    stop drum machine
         |
          run drum machine
```

further information

X and ^ use ON GATE and OFF GATE which in turn generate the MIDI messages 'start', 'continue' and 'stop'. The first ON GATE after MIDIRT generates a 'start', which puts the drum machine to the beginning of its sequence before starting it going, and subsequent X's simply send 'continue', which just starts it going.

You can use X and ^ in any normal musical context, not just as single items in sections. For example, to stop the drum machine for a short pause in music, you would write a rest at the correct position in the clock part, just as you would for a note-playing part.

5 Further applications

The Music 2000 offers much more than the 'common MIDI' we have used so far, by taking advantage of additional AMPLE facilities and the more-advanced MIDI messages available on many devices. This chapter outlines some of these applications, and guides you to entries in Chapter 6, 'Dictionary of words' for examples. Remember that your MIDI equipment may not provide the facilities required – check its own instruction manual to see what it offers.

control from the music keyboard

Since Music 5000 and MIDI voices are interchangeable, the Music 4000 Keyboard can equally be used to play MIDI voices.

The 'Keyboard' option on the main menu calls the contents of the word KEYB into Notepad to provide a keyboard control panel. To convert the panel for MIDI, simply change the 'Simpleins' instrument to MIDIV, using Notepad's text mode. To activate the modified panel, return to panel mode and press f1 to execute it. The keyboard will play the MIDI instrument connected to 'OUT 1'.

The MIDI keyboard panel can be stored as normal using MAKE. If it is given the name KEYB, it will be recalled automatically when 'Keyboard' is selected. Once made as a word, the settings of any control panel can be activated by entering its name as a command.

The keyboard panel supplied by the KFX module (loaded by the example instrument sounds on the Studio 5000-4 disc) can also be converted to MIDI, but those of its controls that use Music 5000 sound words, such as 'TRANS', will not operate on MIDI voices.

adding to the keyboard panel

Any of the MIDI sound words can be added to the bottom of the MIDI version of the keyboard panel to form additional controls. For example, inserting MIDIPROGRAM on the line below MIDIV will allow you to change sound program in the same way as you would adjust 'TRANS' or 'Scale'. MIDILINE and MIDICHANNEL controls will let you select instrument. Make sure that there is at least one space before each item on a line.

Another possibility is a choice of MIDI instrument definitions, selected in the same way as Music 5000 instruments. Each

instrument definition must start with MIDIV, and MIDI sound instructions such as MIDIPROGRAM may follow, for example:

```
   "strings" [ MIDIV 3 MIDIPROGRAM ]
   "leadsynth" [ MIDIV 8 MIDIPROGRAM ]
   ... etc ...
or
   "junostrings" [ MIDIV 1 MIDILINE 3 MIDIPROGRAM )
   "czleadsynth" [ MIDIV 2 MIDILINE 8 MIDIPROGRAM ]
   ... etc ...
```

You now replace the panel's MIDIV by the name of any one of the instruments. When you use the panel, Notepad will automatically present the list of alternatives when you press SHIFT on it.

defining general control panels

Notepad's flexible 'control panel kit' is well suited to a wide range of MIDI applications, since it makes it easy for the user to build panels to suit specific requirements and MIDI instruments.

To make a simple panel, you just enter an instruction with a number value, such as 0 MIDIPROGRAM, in the middle of the first line of the clear Notepad screen. Press f2 to enter panel mode, and use the SHIFT cursor keys as normal to change the value. You will hear the effect of each change immediately if you first set a note playing by entering, in command mode,

```
   READY 1 VOICES MIDIV C
```

When using a panel, RETURN acts as a gate key, so you can use this to strike the notes on the current voices.

You can add further control instructions to the panel on the same and other lines. Where an instruction has more than one input value, each one becomes a separate control. Any word with no input values gives a pop-up menu of alternatives - all words that contain the 'key' word found at the start of the current word's definition, for example, the instrument definitions above. A simple no-choice 'button' can be defined as a word with a unique dummy instruction at its start, for example:

```
   "stopkeyword" []
   "stop" [ stopkeyword 0,^ ]
```

More advanced control panels include context indicator lines such as the 'n CHAN' lines of a Music 5000 instrument panel, distinguished by a non-space at the start of the line. Whenever a control is changed, the context line above it is executed first.

into separate areas for the voices of different parts.

The rules of Notepad panel design are given under 'modifying and designing panels' in Chapter 7, 'Making instruments' of the Music 5000 User Guide.

additional voice control

The AMPLE Nucleus music notation uses just the standard voice performance controls - pitch, gate and velocity - which are applicable to all types of musical voice. MIDI caters for additional performance controls that are specific to music keyboards, and these may be used for extra control over music via corresponding words provided in the Music 2000 software.

Some controls are dedicated, so have specific AMPLE words:

```
MIDIBEND  set pitch bend
  number MIDIBEND
MIDICHPRESSURE  set channel pressure
  number MIDICHPRESSURE
MIDIPRESSURE  set pressure
  number MIDIPRESSURE
```

Other controls are accessed by a general controller word:

```
MIDICONTROL  set control
  valuenumber controlnumber MIDICONTROL
```

MIDICONTROL can simulate an instrument's auxiliary controllers, including the sustain footswitch and continuous foot pedal.

MIDICONTROL also gives access to a range of direct voice parameters that do not correspond to particular physical controllers, including volume and (pan) stereo position. See your instrument's manual for details of general controllers available.

Note that these words, like all other MIDI sound words, operate on the current voice or voices, selected by VOICE, VOICES or RVOICES.

direct voice control

MIDIV voices may be played directly using the three basic performance controls - pitch, gate and key velocity (dynamic level) - rather than the higher-level music notation. The sound words are:

```
GATE  set gate
```

```
    flag GATE
  PITCH  set pitch
    number PITCH
  VEL  set velocity (dynamic level)
    number VEL
```

The words themselves are in the M5 module rather than the M2 module, but they may be freely used on any voice type. Any of the applications of direct voice control of music 5000 voices can be applied to MIDI voices, subject to some limitations of the MIDIV interpretation of the sound words, due to the fact that the information is converted into MIDI key-press form. In particular, music action definitions (using ACT(...)ACT) allow alternative performance modes to be created.

percussion voice control

Most drum machines allow their voices to be played directly from MIDI, as opposed to from their internal sequencers. This lets the computer bypass the sequencer and play the voices from its own parts in the same way as the note-playing voices on a keyboard instrument. The advantage is that the AMPLE's music control power can be used to score drum parts that the internal sequencer cannot represent. In particular, the standard AMPLE Nucleus music notation offers detailed control over timing and velocity, and a variety of scoring methods, with the option of user-defined percussion symbols.

Because 'common MIDI' provides no direct control over individual voices such as percussion voices, drum machines use a special non-MIDI-specification method that assigns the various voices to specific keys: sending a note with a certain pitch will 'hit' a certain instrument. Each drum voice is usually fixed to one particular instrument. Though normal pitch control and sound program change are not possible, and rests ('note off's) are generally ignored, this provides the control necessary for simple applications.

Despite this, drum voices can still appear on separate voices in the AMPLE program, by 'pre-setting' the pitch of each voice to correspond the required key number. This is done using

```
  PITCH  set pitch
    number PITCH
```

X is the 'hit' symbol in the AMPLE Nucleus music notation - see the Music 5000 User Guide or AMPLE Nucleus Programmer Guide for details.

Note that many drum machines allow their voices to played from the MIDI input and the internal sequencer at the same time. This means that you can, for example, add AMPLE-generated hits to a pattern played by the sequencer.

reception modes

MIDI provides four different reception modes, two of which (modes 1 and 2) are used for basic applications.

The best mode for control by AMPLE is mode 4. This allows each AMPLE voice to directly control a single voice in the instrument via its own MIDI channel, and the voices are usually completely independent. Some instruments can receive on more than one channel in other ways, with similar advantages. The instrument's own documentation will explain how to set up the different receive channels. Once this is done, each AMPLE voice is given a corresponding channel number with:

```
MIDICHANNEL  set channel
  number MIDICHANNEL
```

As an alternative to selecting the mode manually on the receiver, mode messages may be sent by the program, using:

```
MIDICONTROL  set control
  valuenumber controlnumber MIDICONTROL
```

sequence control

Many sequencer-type devices, including drum machines, allow a greater degree of control over the performance of the programmed sequence using 'song position pointers' to set the sequence to a particular point. Similarly, where the device can store more than one sequence, or 'song', simultaneously, a 'song select' message can select any one of them.

Song position and song select messages can be sent using:

```
MIDIOUT  send a byte
  bytenumber MIDIOUT

MIDIWOUT  send a word
  number MIDIWOUT
```

See these words for example definitions of complete pointer and select words.

Song position pointers are designed primarily to 'auto-locate' the sequencer when the controlling device is re-started in the middle of the piece, and the song select message is an alternative to selecting the song manually. Both these applications may be made from AMPLE.

In addition, AMPLE allows pointer and select messages to be used dynamically, to control the order of performance while the music plays. On devices that have a sufficiently fast response to these messages, this is a very powerful technique for combining the ease of programming simple patterns on a drum machine, with AMPLE's flexibility of building pieces from sections. See the words MIDIOUT and MIDIWOUT for details of this application.

Note that system messages such as song pointer and select are sent via a AMPLE voices, even though they are not directly affect the voices of the receiver.

manual real-time messages

MIDIRT provides adequate control of sequencer performance for most applications, but it is also possible to send the real-time messages directly for special modes of operation. For example, it may be necessary to send an explicit 'continue' message to start the sequence after setting the song position. You can even generate the clock manually by sending the 'timing clock' tick message repeatedly from one part, making unconventional timing control possible.

See MIDIOUT and MIDIWOUT for details of how to send real-time messages.

system exclusive

System exclusive messages are used by many instruments to control functions outside the MIDI specification. The instructions for your instrument should have details of the system exclusive messages it responds to. You can generate any system exclusive message using MIDIOUT and MIDIWOUT.

Part 2 - Reference

6 Dictionary of words

This chapter provides a detailed description of every word in the Music 2000 Interface's module, M2.

The general form of an entry is as follows:

```
word name  function
   form of use

description

example(s)

related words

further information
```

The form of use shows what input items the word requires. This may be a single number, or two numbers. If no form of use is shown, the word is used by itself, that is, with no input items.

The examples may use AMPLE Nucleus programming words. These are fully described in the AMPLE Nucleus Programmer Guide.

The list of related words includes those which are often used with the word being described, and others which have related or alternative functions in which you might be interested.

index of words

The following index shows the function and input item description of each word, and indicates the order in which the words appear in the dictionary of words.

GATE set gate
  flag GATE
PITCH set pitch
  number PITCH
VEL set velocity (dynamic level)
  number VEL
MIDIBEND set pitch bend
  number MIDIBEND
MIDICHANNEL set channel
  number MIDICHANNEL
MIDICHPRESSURE set channel pressure
  number MIDICHPRESSURE
MIDICONTROL set control
  valuenumber controlnumber MIDICONTROL
MIDILINE set output line
  number MIDILINE
MIDIOUT send a byte
  bytenumber MIDIOUT
MIDIPRESSURE set pressure
  number MIDIPRESSURE
MIDIPROGRAM select program
  number MIDIPROGRAM
MIDIRT assign a real-time control voice
MIDIV assign a voice
MIDIWOUT send a word
  number MIDIWOUT

dictionary of words

GATE set gate

flag GATE

GATE sets the gate of the current voice or voices to ON or OFF. It controls the sound output of a sound-playing voice – ON GATE makes the voice sound (to play a note or hit) and OFF GATE makes it silent (to play a rest).

GATE is used internally by AMPLE music words to carry out note, hit and rests music events. The user may employ GATE directly in definitions of music event actions, and for direct control of voices for special applications.

On a real-time control voice, as opposed to a normal sound-playing one, GATE behaves differently - ON GATE sends a 'start' or 'continue' message, and OFF GATE a 'stop' message - see MIDIRT for details.

examples

```
  ON GATE          % set the current voice's gate to ON

  READY
  1 VOICES MIDIV   % (normal voice)
  ON GATE          % sound the voice
  OFF GATE         % silence the voice

  READY
  1 VOICES MIDIRT  % real-time voice
  ON GATE          % send 'start'
  OFF GATE         % send 'stop'

  "tuneup" [       % play middle C on all instruments
  READY            %  for tuning
  1 VOICES MIDIV   % (middle C is initial pitch)
  "Press RETURN to end" $OUT
  REP(
    3 FOR( % for all lines...
    COUNT MIDILINE          % select line
    ON GATE       % start note
    )FOR
    200 DURATION  % wait awhile
    #IN 13 #= )UNTIL( )REP % repeat until RETURN pressed
  3 FOR(           % for all lines...
   COUNT MIDILINE  % select line
   OFF GATE )FOR ] % stop note
```

related words VOICE PITCH VEL ON OFF

further information

The precise effect of GATE on the sound depends on the instrument in use, since it takes effect through the voice's envelopes. Conventional envelope shapes have an 'on' and an 'off' section, and each GATE causes the appropriate section to be applied to the sound. A simple envelope like that of an organ has an 'on' section that plays the sound with constant loudness, and an 'off' section which silences it, so there is a direct correspondence with the GATE state. In contrast, the envelope of a piano-like instrument will have an 'on' section that starts the sound on ON GATE, but lets it decay to silence naturally, if not silenced by an OFF GATE. Another kind of envelope has an 'off' section with a slow decay instead of instant silence - the immediate effect of OFF GATE is diminished so short rests may be less audible. Drum-like instruments are usually created with piano-style envelopes so that ON GATE is used for normal hits, and OFF GATE provides an extra damping control where the decay is long.

In the MIDI interpretation of GATE, each MIDIV voice translates ON GATE and OFF GATE into key-on and key-off messages (called 'note-on' and 'note-off' in MIDI parlance), while taking account of the restriction that each key must receive strictly alternating 'on' and 'off' messages. Every ON GATE sends a key-on message using the last-set PITCH and VEL values, and a following OFF GATE sends a key-off message for the same key. If ON GATE is repeated without an intervening OFF GATE (as normal in a sequence of notes), the required key-off message is automatically sent first. If OFF GATE is repeated without an intervening ON GATE, no message is sent.

GATE uses the MIDI 'note on' message for key-on events, and this or 'note off' for key-off events, depending on the VEL setting.

The M2 module does not in fact include the word GATE, but it is described here since the definition provided by any other module in the installation may be used as described.

PITCH set pitch

number PITCH

PITCH sets the pitch of the current voice or voices in semitone units. The range is -60 to 67, with 0 being middle C.

PITCH is used internally by AMPLE music words to carry out note music events. The user may employ PITCH directly in definitions of music event actions, and for direct control of voices for special applications.

PITCH has no immediate effect on the destination instrument, but determines the key used by subsequent ON GATE and OFF GATE instructions.

PITCH is especially useful for directing an AMPLE voice to a particular percussion voice on a drum machine, identified by a key number. Once the pitch is set on a voice, all hits on that voice are transmitted on the corresponding key number, and therefore to the associated percussion voice.

On a real-time control voice, as opposed to a normal sound-playing one, PITCH has the effect of clearing the pending 'start' message – see MIDIRT for details.

examples

```
  12 PITCH         % set the current voice's pitch to 12 -
                   % the C above middle C

  READY
  1 VOICES MIDIV   % (normal voice)
  12 PITCH         % set the pitch
  ON GATE          % sound the voice

  % play random pitch sequence
  "bleeper" [
  READY
  1 VOICES MIDIV   % set up a voice
  REP(
   24 RANDL PITCH  % set random pitch
   ON GATE         % sound voice
   10 DURATION     % wait for 10 ticks
  )REP ]           % repeat until ESCAPE pressed

  % demonstration key addressing for percussion
  READY
  1 VOICES MIDIV   % set up a voice for percussion...
   -12 PITCH       % on the key below middle C (number 48)
  X//XX/X/X        % sent hits to that key/percussion voice

  % word to set midi key number for current voice
  "midikey" [ % number midikey
  60 #-            % subtract key number of 0 PITCH
  PITCH ]
  ...
  48 midikey       % equivalent to  -12 PITCH
```

```
% mix word for 10 percussion voices on drum machine
% (key numbers depend on particular drum machine)
"mix9" [
5 SHARE 10 VOICES MIDIV  % get 10 voices on part 5
 1 VOICE  35 midikey     % bass drum
 2 VOICE  38 midikey     % snare drum
 3 VOICE  39 midikey     % hand clap
 4 VOICE  42 midikey     % closed hi-hat
 5 VOICE  46 midikey     % open hi-hat
 6 VOICE  41 midikey     % low tom
 7 VOICE  45 midikey     % mid tom
 8 VOICE  48 midikey     % high tom
 9 VOICE  49 midikey     % crash cymbal
10 VOICE  51 midikey     % ride cymbal
PNUM SHARE ]             % end of mix

% midikey list word for more convenient kit definition
"mklist" [ % number1 number2 ... numbern quantitynumber mklist
FOR(             % for the spec1fied quantity of voices...
 INDEX VOICE     % count down through the voices
 60 #- PITCH     % using successive numbers to set pitch
)FOR ]
...
"mix9" [
5 SHARE 10 VOICES MIDIV
 35 38 39 42 46 41 45 48 49 51  10 mklist
PNUM SHARE ]

% percussion pattern scoring method
"patt1" [ 24,
%b shcolmhcr     % reminder of instruments
 X(/////////)
 /(//X//////)    % //////) has no effect – for guide only
 X(X//X/////)
 X(/////////)
 /(//X//////)
 X(X//X/////)
 /(/X///////)

% word to direct rests to next voice up, for scoring hits and
% rests on an instrument that the drum machine p rovides as
% separate hit and rest instruments, e.g. hi-hat
"restup" [
20 ACT(
5 FVAR #?
6 FVAR #? AND IF(
 5 FVAR #? 1 #+
 VOICE! )IF
ACT )ACT ]
```

```
...
 2 VOICE  42 midikey      % closed hi-hat  (in mix)
 1 VOICE  46 midikey      % open hi-hat
...
SCORE restup 24,
X^X^X^X^                 % alternate open-closed pattern
```

related words VOICE GATE VEL

further information

The M2 module does not in fact include the word PITCH, but it is described here since the definition provided by any other module in the installation may be used as described.

VEL set velocity (dynamic level)

number VEL

VEL sets the dynamic level of the current voice or voices. The range is 1 to 127, and the initial value is 64.

VEL is used internally by AMPLE music words to carry out note and hit music events, executing the dynamic level value provided by =L, +L and -L. The user may employ VEL directly for advanced alternative music event actions, and direct control of voices for special applications.

examples

```
80 VEL           % set the current voice's velocity to 80

READY
1 VOICES MIDIV   % (normal voice)
ON GATE          % sound the voice with normal velocity
80 VEL           % set higher velocity
ON GATE          % sound the voice

% using VEL to balance instruments in a kit
"mix9" [
5 SHARE 10 VOICES MIDIV     % get 10 voices on part 5
 1 VOICE  35 midikey 80 VEL % bass drum
 2 VOICE  38 midikey 60 VEL % snare drum
 3 VOICE  39 midikey 75 VEL % hand clap
 ...
PNUM SHARE ]
```

related words PITCH GATE

further information

VEL may also be used to control key release velocity - see Chapter 5 for information.

The M2 module does not in fact include the word VEL, but it is described here since the definition provided by any other module in the installation may be used as described.

MIDIBEND set pitch bend

number MIDIBEND

MIDIBEND sets the amount by which the note pitch is 'bent' above or below the normal pitch, by sending a pitch bend setting on the channel of the current voice. The MIDI pitch bend message is intended for communicating the position of the pitch bend wheel fitted to many keyboards, but it may be used from AMPLE scores to vary pitch both continuously (between semitone values) and within notes (between 'note-on ' and 'note-off').

The number is in the range -8192 (lowest) to 8191 (highest), with 0 being the normal, centre position. The MIDI pitch bend value is not designed to have a specific degree of effect, so the pitch difference resulting from a particular value depends on the particular receiver and the pitch bend 'range' or 'sensitivity' setting of the receiver, if it has one. Most receivers allow a maximum pitch bend range of approximately eight to twelve semitones above and below the main pitch, so AMPLE programs should assume a 'standard' total range of plus and minus seven semitones to be able to suit most receivers. This range gives 1024 pitch bend units per semitone.

Note that, independently, the pitch bend message is not designed for precise pitch intervals, such as between the note pitch and another pitch of the scale. However, since MIDI provides no other standard method of changing pitch within a note, it is useful to use pitch bend for this, and this can be successful if the receiver's pitch bend sensitivity is set precisely. Again, the program should use a 'standard' pitch bend scale such as 1024 units per semitone, and then the receiver's sensitivity be tuned for the correct pitch with the pitch bend set on the largest available chromatic interval in the range.

examples

```
1000 MIDIBEND         % set bend to 1000 units
...
0 MIDIBEND            % return bend to 0 – normal pitch
```

```
% semitone MIDIBEND for chromatic bends.
% receiver's pitch bend sensitivity to be manually set to match
% (wider range may be used if available - reduce 1024 to 512)
"sb" [  % number sb   % range of number is -8 to 7
1024 #*               % multiply by 1024
MIDIBEND ]            % send as normal
...
SCORE
C 2 sb /              % bend C to D (adjust bend sens. for D)
0 sb                  % return to C

% MIDIBEND exerciser for testing receiver's response
"bender" [ % press < to decrease, and > to increase
0 REP(                              % starting value is 0
 13 #OUT #11 $STR 5 $PAD $OUT       % print number neatly
 #IN "," ASC #=                     % get a key - is it , (<)?
 IF( 256 #- )ELSE( 256 #+ )IF       % go up or down
 #11 MIDIBEND                       % set bend
)REP ]
...
1 VOICES MIDIV SCORE C              % set up note specially
bender                              % bend it
...
RUN                   % play a complete piece
3 SHARE               % select one of the MIDI parts
bender                % bend it while playing

% simple glissando words: +gl & -gl  to go up & down
% (first step at normal pitch, last + 1 step at final pitch)
% range of number is -8 to 7 (using multiplier 1024 as shown)
"+gl" [ % number +gl
0 MIDIBEND            % start at normal pitch
FOR( /                % wait between steps
 COUNT                % count...
 1024 #* MIDIBEND     % ...up the pitch bend
)FOR ]
"-gl" [ % number -gl
0 MIDIBEND            % start at normal pitch
FOR ( /               % wait between steps
 COUNT                % count...
 -1024 #* MIDIBEND    % ...down the pitch bend
)FOR ]
...
7 +gl                 % glissando up by 7 semitones
48,C 6,/ 7 +gl 48,/   % glissando up to G on 3rd 48, beat
```

```
% slurred notes automatically played by pitch bend of prev note
% receiver's pitch bend sensitivity to be manually set to match
"lastpitch" [GVAR]    % variable holding last note pitch
"slurbend" [
30 ACT(               % definition of music event action
 1 FVAR #? IF(        % if pitched event...
  5 FVAR #? IF(       % then, if' gated...
   0 MIDIBEND         % cancel bend
   2 FVAR #? lastpitch #!    % record pitch
   ACT                % play as normal
  )ELSE(              % else, if not gated (slurred)...
   2 FVAR #?          % get pitch value
   lastpitch #? #-    % subtract last pitch
   1024 #* MIDIBEND   % bend by difference
   7 FVAR #? DURATION % execute duration
  )IF
 )ELSE(               % else (that is, not pitched),
  ACT                 % execute as normal
 )IF )ACT ]
...
SCORE slurbend
C ~D                  % slur C to D (adjust bend sens. for D)
~C                    % return to C
C~DD~G C~b~a~g
f~e~d~c ^

% glissando, stepping up on each beat (maximum 7 beats)
"glvar" [GVAR]
"glh" [ % flag glh
#11 IF( #2            % if turning on
 0 glvar #!           % start at 0
 0 MIDIBEND
 30 )IF               % ( leave ACT( value )
ACT( ACT              % do each music event as normal
 1024 glvar #+!       % move to next step
 ON VOICE             % select voices – ACT may do OFF VOICE
 glvar #? MIDIBEND    % set bend
) ACT ]
...
48,C 6, / ONglh /// //// OFFglh 48,/ % go up to G on 3rd beat
...
ONglh 6,CCCCCCC       % chromatic scale

% self-contained glissando word, which does not extend
% previous event, i.e. does not displace music time position.
"gl" [ % durnumber stepsnumber gl
#11 0 #<              % negative?
IF( 0 #12 #-          % make +ve
 #12 #212 -1024       % keep copy of num & leave -ve multiplier
)ELSE(
```

```
 #12 #212 1024 )IF  % keep copy of num & leave +ve multiplier
0 MIDIBEND          % start at normal pitch
#12 FOR(
 #212 DURATION      % wait between steps
 #11 COUNT #* MIDIBEND % set bend for next step
)FOR #2             % discard multiplier
#*                  % get total duration of all steps
0 #12 #-            % negate
DURATION ]          % execute to return time to start of gliss.
...
48,C          / A   % without gliss
...
48,  6 7 gl   / A   % up 7 step of 6 units long, inside the /
                    % can also be used on Staff

% continuous, wheel-style pitch bend up and back to normal
% bend plays where placed onwards, without extending prev event
"ud" [
 10 FOR( 3 DURATION         % wait between steps
  COUNT 300 #* MIDIBEND     % go up
 )FOR
 10 FOR( 3 DURATION         % wait between steps
  INDEX 300 #* MIDIBEND     % go down
 )FOR
 0 MIDIBEND                 % return to normal
 -60 DURATION ]             % return time to start of bend
...
48,C ud //F G ud //
```

related words MIDILINE

further information

The pitch bend setting applies to all notes playing on that channel.

A particular receiver may not use the full range of precision of the pitch bend message, ignoring some lower bits in the binary value. The effect is that the finest changes are ignored, for example, values 0 to 15 give a single pitch, and values 16 to 31 give a single, slightly higher, pitch. This particular case would give a total range of -512 to 511 distinct steps (spread evenly over the normal range of -8192 to 8191), which is adequate for most purposes. It may be useful to define a word that works in precise receiver pitch steps, as follows:

```
% coarse MIDIBEND, for a receiver that
% ignores the bottom 4 bits (uses each 16th step)
"cbend" [ % number cbend    % range of number is -511 to 512
16 #*                % multiply by 16
MIDIBEND ]           % send as normal
64 cbend             % equivalent to  1024 MIDIBEND
```

Because limited-precision pitch bend response always has a step size of an integral power of two units (8, 16, 32 etc.), it is a good idea to make any higher pitch bend multiplier (for semitones, for example) also a power of two, such as 512, 1024, etc., to ensure maximum accuracy.

The MIDI pitch bend message value is an unsigned 14-bit value with the range 0 (lowest) to &2000 (centre, normal pitch) to &2FFF (highest). MIDIBEND converts its input number into this form by adding &2000, and discarding unused higher bits.

MIDICHANNEL set channel

number MIDICHANNEL

MIDICHANNEL sets the number of the MIDI channel to be used by the current voice or voices. It is used to direct control to different instruments (or separate voices in the same instrument) when they are connected to the same MIDI output line and set to receive on different channels. The number has the range 1 to 16, and is initially l.

MIDICHANNEL is normally used after MIDIV (or MIDIRT) to select a particular output line to be used by that voice or group of voices throughout the piece. Each output line has its own 16 channels, so MIDICHANNEL may be used with MIDILINE to select the line and channel required.

examples

```
2 MIDICHANNEL    % set current voice to send on channel 2

2 SHARE 8 VOICES
 MIDIV 3 MIDILINE 2 MIDICHANNEL     % typical voice setting

% 'location' of named receiver on a particular line channel
"tr505" [ 3 MIDILINE 10 MIDICHANNEL ]

% complete 'mode 4' (OMNI OFF/MONO ON) word, including voice
% channel and mode, with selectable line and basic channel.
"mode4" [ % basicchannelnumber linenumber voicesnumber mode4
#11 VOICES MIDIV     % set up number of voices reqd.
MIDILINE             % select output line
```

```
  #212 MIDICHANNEL      % select basic channel for...
  0 124 MIDICONTROL     % 'omni off', and
  #11 126 MIDICONTROL   % 'mono on' with number of channels
  FOR(                  % For each voice,
   COUNT VOICE          % select it, and
   #11 COUNT #+ 1 #-
   MIDICHANNEL          % select voice ch. above basic ch.
  )FOR
  #2 ]                  % discard the basic channel number
  ...
  1 SHARE 3 1 8 mode4   % on basic chan 1, line 1, set up 8 voices
  2 SHARE 8 1 4 mode4   % on basic chan 1, line 1, set up 8 voices
```

related words MIDILINE

further information

MIDICHANNEL affects all channel messages sent by the voice on which it is used. The class 'channel messages' include those sent by the following words:

```
  notes, hits and rests (note messages sent by GATE)
  GATE
  MIDIBEND
  MIDICHPRESSURE
  MIDICONTROL
  MIDIPRESSURE
  MIDIPROGRAM
```

Any channel messages assembled manually and sent directly with MIDIOUT and MIDIWOUT contain explicit channel numbers, and are unaffected by the MIDICHANNEL setting.

Like MIDILINE, MIDICHANNEL is not normally used in the middle of a message sequence, such as a string of notes, since changing line or channel at this point may interrupt the sequence received by an instrument. For example, the instructions

```
  1 MIDICHANNEL C  2 MIDICHANNEL D
```

will leave the C playing indefinitely, since the message to silence it (sent by the D) is diverted to another instrument. For the same reason, neither executing READY nor pressing ESCAPE will silence a note that has been started on a line channel other than the voice's current one.

Note that each MIDI channel on each line can carry a musical part playable on a single keyboard, but no more. In particular, only one note per key is allowed. For accurate performance in all cases, each part should be directed via a separate channel to

either a separate instrument (or a separate channel of an instrument with multiple receive channels) - this also allows each part to have a different sound. Ideally, a separate channel should be used for each voice, also allowing access to individual voice effects provided by the receiver, such as true slurs between successive notes of different pitches.

MIDICHPRESSURE set channel pressure

number MIDICHPRESSURE

MIDICHPRESSURE sets the amount of key pressure effect to be applied to all notes on the current voice's channel. The number is in the range 0 (normal) to 127 (maximum pressure).

The corresponding MIDI message ('channel pressure') is intended for communicating the amount of pressure applied to the keys of a keyboard, and its effect on the sound is dependent solely on the receiver. Receivers that are equipped to respond to key pressure usually allow it to control a number of sound parameters, so MIDICHPRESSURE provides an additional means of sound control from AMPLE programs, for special applications.

MIDI's 'channel pressure' message is similar to the 'key pressure' message sent by MIDIPRESSURE, the difference being that 'channel pressure' affects all notes on that channel, whereas 'key pressure' affects just the note ('key') to which it is applied. In theory, MIDICHPRESSURE is less useful than MIDIPRESSURE since when applied to a voice, other voices will be affected if they are on the same channel, but receivers that respond to MIDICHPRESSURE are more common than those that respond to MIDIPRESSURE .

examples

```
80 MIDICHPRESSURE     % set pressure to 80 units
...
0 MIDICHPRESSURE      % set pressure to 0 - normal

% MIDICHPRESSURE exerciser for testing receiver's response
"presser" [ % press < to decrease, and > to increase
0 REP(                              % starting value is 0
 13 #OUT #11 $STR 3 $PAD $OUT       % print number neatly
 #IN "," ASC #=                     % get a key – is it , (<)?
 IF( 1 #- )ELSE( 1 #+ )IF           % go up or down
 #11 MIDICHPRESSURE                 % set pressure
)REP ]
...
1 VOICES MIDIV SCORE C              % set up note specially
presser                             % press it
...
```

```
RUN                         % play a complete piece
3 SHARE                     % select one of the MIDI parts
presser                     % change pressure while playing


% simple word to give fixed increasing pressure within a note.
"pr" [ % increasing pressure over period of 96 ticks
0 MIDICHPRESSURE            % start at 0
 48 FOR(
  2 DURATION                % wait between steps
  COUNT MIDICHPRESSURE      % increase pressure to 50 max.
 )FOR
-96 DURATI0N ]              % retract total duration (48x2)
"endpr" [ % return to normal instantly
0 MIDICHPRESSURE ]
...
48, pr C/                   % start pressure at start of note
48, C pr //                 % start pressure after 1 beat
48, pr C/ endpr G/ d/ pr // endpr

% word to give settable increasing pressure within a note.
% plays onwards from 'press', without extending prev event
"pr" [ % depthnumber durnumber pr
0 MIDICHPRESSURE            % start at 0
 #212 FOR(
  #11 DURATION              % wait between steps
  COUNT MIDICHPRESSURE      % increase pressure
 )FOR
#*                          % total duration
0 #12 #-                    % negate
DURATION ]                  % execute to move back to start time
...
48, 2 50 pr C/              % pressure up to 50 over 100 ticks
0 0 pr                      % set pressure back to 0
48, 2 50 pr C/ 0 0 pr G/ d/ 2 50 pr // 0 0 pr
                                 % can also be used on Staff.
```

related words MIDIPRESSURE MIDILINE MIDICHANNEL

MIDICONTROL set control

```
valuenumber controlnumber MIDICONTROL
```

MIDICONTROL sends a control change message to set the numbered control to the given value (position). It provides computer control over the effective positions of the instrument's panel and internal controls, and also the instrument's MIDI reception mode.

The control number describes which of the receiver's controls is to be set, and has a range of 0 to 127. The value can be a number in the range 0 to 127, or a flag – OFF and ON are translated in to

0 and 127 respectively. Normally you will use a number where the controller is continuous (as a knob, slider or wheel), and a flag where it is switch-type.

The control number range is divided as follows:

control number	function
0 - 31	continuous controls 0 - 31, bits 7-13
32 – 63	continuous controls 0 - 31, bits 0-6
64 - 95	switch-type controls (0 = OFF, 127 = ON)
96 - 121	undefined
122 - 127	channel mode controls

Continuous controls have a value range of 14 bits, sent as two separate 7-bit values under different control numbers. To set the complete value, two control messages must be sent, but if a change in value requires only one of the 7-bit groups to be changed, then only the single appropriate control message need be sent.

Switch controls should only be sent values of 0 and 127, representing OFF and ON respectively. MIDICONTROL converts ON and OFF flag values accordingly.

The function of the various control numbers depends entirely on the receiver. However, some controls have been assigned specific functions, though a given receiver may or may not provide each function. For example, continuous control number 1 has been assigned to the modulation wheel, and therefore controls the modulation depth on the receiver, and switch-type control number 64 has been assigned to the sustain pedal, and therefore holds the notes playing on the receiver when ON. The exact interpretation of the values of assigned controls depends on the receiver. Assigned control numbers are described in the Chapter 8, 'MIDI Specification summary'.

Undefined control numbers should not be used.

The channel mode controls determine the way in which the receiver responds to incoming MIDI messages - its 'MIDI reception mode'. They are in fact not considered as controls (they rarely exist as physical controls on receivers), but specifically as 'channel mode

messages'. They are as follows:

control number	function	control value	meaning
122	local control on/off	0	off
		127	on
123	all notes off	0	
124	omni off	0	
125	omni on	0	
126	mono mode	0	max qty of channels
		1-16	quantity of channels
127	poly mode	0	

Unlike other channel messages, channel mode messages are accepted only if sent on the receiver's basic channel whatever reception mode it is in. The omni on/off and poly/mono messages are used together to set the receiver to one of reception mode s 1, 2, 3 and 4, and for mode 4, to set the number of voices. Channel mode messages are described more fully in Chapter 8, 'MIDI Specification summary'.

examples

```
30 1 MIDICONTROL        % set control no. 1 (modulation) to 30

ON 64 MIDICONTROL       % set control no. 64 (sustain) ON

"mod" [ % number mod
1 MIDICONTROL ]         % define word 'mod' to set modulation
...
30 mod                  % use mod to set modulation to 30
                        %   on the current voice, for example
READY
1 VOICES MIDIV          % set up a MIDI voice
C                       % start a note playing
30 mod                  % add modulation
0 mod                   % remove modulation

"sus" [ 1 flag sus
64 MIDICONTROL ]        % define word 'sus' to set sustain state
...
ON sus                  % use sus to turn sustain on,
                        %   (on an existing voice).

OFF 122 MIDICONTROL     % mode message to turn local control OFF

"local" [ % flag local  % define word for 'local control'
122 MIDICONTROL ]
...
```

```
OFF local               % turn local control off
...
ON local                % turn local control on

"mode4" [               % define word to select mode 4, sending
0 124 MIDICONTROL       % 'omni off', and
0 126 MIDICONTROL       % 'mono on' (number of channels = 0)
]
...
3 SHARE 1 VOICES
    MIDIV mode4         % use it on a new voice

% word definition to test control operation by entered value
"contest" [ % controlnumber contest
REP(
 "Enter value: " $OUT $IN  % get value from user input
 VAL $2                    % convert string to number
 IF(                       % if number found,
  #212 MIDICONTBOL         % copy control number and send
 )IF
)REP ]                     % repeat until ESCAPE pressed
READY 1 VOICES MIDIV       % set up a voice
C                          % start a note playing
1 contest                  % test control 1, modulation

% scan control value up and down using '.' and ',' keys
"conscan" [ % controlnumber conscan
0 REP(                          % starting value is 0
 13 #OUT #11 $STR 3 $PAD $OUT   % print number neatly
 #IN "," ASC #=                 % get a key - is it , (<)?
 IF( 1 #- )ELSE( 1 #+ )IF       % go up or down
 #2121 #12                      % copy value and controlnum
 MIDICONTROL                    % set control
)REP ]                          % repeat until ESC pressed
READY 1 VOICES MIDIV    % set up a voice
C                       % start a note playing
1 conscan               % scan control 1, modulation
...
RUN                     % start the piece playing
3 SHARE                 % share a player with MIDI voices
1 conscan               % scan control 1 while playing

% set full 14-bit control value in two operations
&10  1 MIDICONTROL      % send modulation bits 7-l3
&04 33 MIDICONTROL      % plus bits 0-6, total value: &804

% word to set a given 14-bit control from a single value number
"midicontrolw" [ % valuenumber controlnumber midicontrolw
#2121                   % copy value number and control number
32 #+                   % add 32 to get 'LSB' control number
```

```
MIDICONTROL             % send bits 0-6 to 'LSB' control number
#12                     % get value number to stack top
2 #*                    % shift bits 7-13 to Upper byte
#B12                    % swap bytes, moving bits 7-13 to 0-6
#12                     % back to 'valuenumber controlnumber'
MIDICONTROL             % send bits 7-13 on 'MSB' controlnumber
]
...
&808 1 midicontrolw     % set modulation to &808

% word to route music's dynamic level (=L) to spec ified control
"levcont" [ % flag levcont
30 ACT(
3 FVAR #? VOICE         % select voice
4 FVAR #?               % get level value
1 MIDICONTROL           % send to control (no. 1 = modulation)
OFF 3 FVAR #!           % disable further level interpretation
ACT )ACT ]              % continue event interpretation
...
SCORE levcont 24,       % select control of modulation by level
0 =L                    % without modulation
0:C/G/c/G/c/deC/C^
30 =L                   % with modulation
0:C/G/c/G/c/deC/C^
O=L 30 15 +L            % with increasing modulation
0:C/G/c/G/c/deC/C^

% control from computer's joystick
"joyval" [ % axisnumber joyval -> valuenumber
128 &FFF4 CODE #2       % get value in range 0-65535
512 #/ #2               % reduce range to 0-127
#11 128 AND #+ ]        % correcting sign
"joymod" [
REP(
 1 joyval               % get joystick value
 1 MIDICONTROL          % set control
)REP ]                  % repeat until ESCAPE pressed
READY 1 VOICES MIDIV    % set up a voice
C joymod                % play note and try it

% improved version that sends changed values only
"joymodi" [
0                       % initial value
REP(
 1 joyval
 #2121 #-               % if changed...
 IF(
  #11 1 MIDICONTROL     % send new
  #12                   % and exchange old and new
 )IF #2                 % discard
```

```
  )REP ]

  % two-axis version, controlling control and pitch bend
  "dualjoy" [
  0 0                     % initial values
  REP(
   1 joyval
   #2121 #- IF(           % if changed...
    #11 1 MIDICONTROL     % send new
    #12                   % leave old on top
   )IF #2                 % discard
  #12                     % get old pitch bend value to top
   2 joyval
   #2121 #- IF(           % if changed...
    #11
    64 #-                 % adjust to range of -64 – 63
    128 #* MIDIBEND       % set pitch bend
    #12                   % leave old on top
   )IF #2                 % discard
  #12 )REP ]              % get old control value to top
  READY 1 VOICES MIDIV    % set up a voice
  C dualjoy               % and try it.
```

further information

MIDICONTROL ignores bits 7-15 of the control and value numbers.

MIDILINE set output line number MIDILINE

MIDILINE sets the number of the MIDI output line to be used by the current voice or voices. It is used to direct control to different instruments when they are connected at different MIDI output lines. The line number has the range 1 to 3, and is initially 1.

MIDILINE is normally used after MIDIV to select a particular output line to be used by that voice or group of voices throughout the piece, or after MIDIRT to direct real-time messages, including clock, to a drum machine.

examples

```
  2 SHARE                    % in 'mix9' word,
  8 VOICES MIDIV 2 MIDILINE  % set up 8 voices on output line 2

  8 SHARE                    % in 'mix9' word,
  1 VOICES MIDIRT 3 MIDILINE % set up clock on line 3, keeping
                             % lines 1 and 2 free for notes
```

```
  "juno"  [ MIDIV 1 MIDILINE ]    % words for voices of named
  "cz101" [ MIDIV 2 MIDILINE ]    % actual instruments on
  "oscar" [ MIDIV 3 MIDILINE ]    % particular output lines.
  "mix9" [                        % use in 'mix' word
  3 SHARE 8 VOICES juno
  4 SHARE 8 VOICES cz101
  5 SHARE 1 VOICES oscar
  PNUM SHARE ]

  % MIDI reset command, sending on all lines
  "midireset" [
  READY 1 VOICES MIDIV
  3 FOR(                      % for all three lines
   COUNT MIDILINE             % select line
   &FF MIDIOUT                % and send 'reset' code
  )FOR
  READY ]                     % free the voice when done
```

related words MIDICHANNEL

further information

MIDILINE affects all messages sent by the voice on which it is used, including all channel messages, clock messages and those sent directly with MIDIOUT and MIDIWOUT.

Like MIDICHANNEL, MIDILINE is not normally used in the middle of a message sequence, such as a string of notes, since changing line or channel may interrupt the sequence received by an instrument. For example, the instructions

```
  1 MIDILINE C  2 MIDILINE D
```

will leave the C playing indefinitely, since the message to silence it (sent by the D) is diverted to another instrument. For the same reason, neither executing READY nor pressing ESCAPE will silence a note that has been started on a line channel other than the voice's current one.

MIDIOUT send byte

bytenumber MIDIOUT

MIDIOUT sends a single byte directly to the MIDI output line of the current voice. Bits 0-7 (the low byte) of the supplied number are sent, and the rest ignored.

MIDIOUT is used for special applications including user definition

of additional control instructions, and transmission of system exclusive data.

examples

```
% send 'system reset' message on an existing voice
&FF MIDIOUT             % sends &FF

% complete MIDI 'system reset' command
"sysreset" [
READY                   % stop any music and free all voices
1 VOICES MIDIV          % get a voice
3 FOR(                  % for all output lines,
 COUNT MIDILINE         % select the line, and
 &FF MIDIOUT            % send 'system reset' message.
)FOR
READY ]                 % finally, free the voice

% instant 'music concrete' from random MIDI data
% (turn write-protect ON before using, if required, and
% select omni on/poly mode for maximum effect)
"concrete" [
READY 1 VOICES MIDIV    % set up a voice (output line 1)
REP( RAND MIDIOUT          % send a random byte
)REP ]                     % repeat until ESCAPE is pressed
concrete

"slowconcrete" [        % slower version - wait between bytes
READY 1 VOICES MIDIV    % set up a voice (output line 1)
REP( RAND MIDIOUT          % send a random byte
5 DURATION                 % adjust number for desired density
)REP ]                     % repeat until ESCAPE is pressed
slowconcrete

% word to send system exclusive message
% sends any short fixed sysex. message on the existing voice
"sysex" [ % numbern ... number2 number1 quantitynumber sysex
&F0 MIDIOUT             % start system exclusive
FOR( MIDIOUT            % send the next supplied number,
)FOR                    % 'quantitynumber' times.
&F7 MIDIOUT ]           % end system exclusive
...
1 VOICES MIDIV
0 &2B 1 &3A  4 sysex   % sends:  &FO &3A &01 &2B &00 &F7

% alternative left-to-right format system exclusive word
"sysex" [ % -1 number1 number2 ... numbern sysex
&FO MIDIOUT             % start system exclusive
FRAME                   % mark numbers as FVAR elements
1                       % initial element number
```

```
REP(
#11 FVAR #?             % get element
 -1 #= )UNTIL(          % repeat until -1 reached
 1 #+ )REP              % move to higher-numbered element
#11 1 #-                % move to element one before -1
FOR( INDEX              % in decreasing element order...
 FVAR #? MIDIOUT        % get value and send
)FOR
FOR( #2 )FOR   &F7 MIDIOUT ]   % discard all numbers
...
1 VOICES MIDIV
-1  &3A 1 &2B 0  sysex  % send in left-right order, from -1

% song position pointer word, converting to 2x7-bit data format
"songpos" [ % number songpos % 0 <= songpos <= 16384
&F2 MIDIOUT           % send 'song position pointer' status byte
&3FFF AND             % force to 14-bit range
#11
&7F AND MIDIOUT       % send bits 0-6 in data byte 1
128 #/ #2             % move bits 7-13 to bottom byte
MIDIOUT ]             % send bits 7-13 in data byte 2

% song position pointer word, working in 4/4 bar (4x48,) units
"songbar" [ % barnumber songbar
6 #/ #2               % divide by MIDI clocks per song pos unit
24 #*                 % multiply by MIDI clocks per quarter-note
4 #*                  % multiply by quarter-notes per bar
songpos ]
% alternative version – pre-calculated literal factor for speed
"songbar" [ % barnumber songbar
16 #* songpos ]

% example uses of song position pointer words

% locating of drum machine on score sections
"mix9" [
9 SHARE 1 VOICES MIDIRT     % set up for clock in mix
PNUM SHARE ]
"part9a" [ 0,^  0 songbar X ]
"part9b" [ 0,^ 16 songbar X ]   % (length of section a)
"part9c" [ 0,^ 32 songbar X ]   % (length of sections a&b)
...
"1239-9abca"PLAY      % play normally (note: 'a' repeated)
"1239-9b"PLAY         % play section b - starts at bar 16

% chaining drum machine patterns from AMPLE score -
% drum machine song is programmed to 'patt1 patt2 patt3 etc',
% and then patterns accessed by song position pointer, in play
"patt" [ % patternnumber patt
0,^ songbar X    % select pattern by sending song position
```

```
  48,////         % wait 1 bar (assuming patterns are in 4/4)
  "mix9" [
  9 SHARE 1 VOICES MIDIRT    % setup clock in mix
  PNUM SHARE ]
  "part9" [ X          % X starts drum machine playing
  0 patt  0 patt       % sequence of patterns to be played
  0 patt  1 patt       % (0 is first pattern in song)
  ... etc ...
  ]
  % alternative, more compact, pattern sequence format
  "patts" [ % patternnumstring patts
  $REV                       % reverse string (loop re-reverses)
  0                          % initial pattern number count
  REP( VAL                   % get next pattern number
   NOT )UNTIL(               % until no more
   #12 1 #+                  % store number, and increment count
  )REP $2                    % discard remaining string
  FOR( patt )FOR             % execute patterns in order
  % note: this word does not send pattern numbers directly from
  % the string, since players are allowed only temporary string
  % use (that is between, but not over, music events such as /
  ]
  "0 0 0 1 2 2 2 3" patts    % play specified pattern sequence
                             % (only single digits allowed)

  % note: some drum machines take a noticeable time to respond to
  % song pointer position messages, during which they cannot play
  % notes, and therefore will not be able to carry out these
  % operations reliably.
```

related words MIDIWOUT

further information

MIDIWOUT is an alternative to MIDIOUT which provides more-efficient transmission of pairs of bytes.

MIDIPRESSURE

set pressure

number MIDIPRESSURE

MIDIPRESSURE sets the amount of key pressure effect applied to the note on the current voice. The number is in the range 0 to 127, and is initially 0.

The corresponding MIDI message ('key pressure') is intended for communicating the amount of pressure applied to each key of a keyboard, and its effect on the sound is dependent solely on the receiver. Receivers that are equipped to respond to key pressure

usually allow it to control a number of sound parameters, so MIDIPRESSURE provides an additional means of sound control from AMPLE programs for special applications.

MIDI's 'key pressure' message is similar to the 'channel pressure' message sent by MIDICHPRESSURE, the difference being that 'key pressure' applies to an individual note ('key'), whereas 'channel pressure' applies to all notes on that channel. Therefore, in theory, MIDIPRESSURE is more useful, but receivers responding to 'key pressure' are less common than those responding to 'channel pressure'.

examples

```
80 MIDIPRESSURE     % set pressure to 80 units
...
0 MIDIPRESSURE      % set pressure to 0 – normal

% MIDICHPRESSURE examples work equally with MIDIPRESSURE –
% see MIDICHPRESSURE
```

related words MIDICHPRESSURE

MIDIPROGRAM select program number MIDIPROGRAM

MIDIPROGRAM selects the sound to be used for the current voice or voices by sending a 'program change' message to the instrument. It is used either after MIDIV to select a sound for use throughout the piece, or at some later point to change to a different sound. The range is 0 to 127. Initially, the sound used is the one selected on the instrument.

Though MIDIPROGRAM may be applied to a single voice, MIDI applies 'program' changes to all voices on the same channel, so they must all use the same sound. In other words, for a voice to have its own independent instrument sound, it must be on a channel of its own and hence on either its own instrument, or its own channel of an instrument with multiple receive channels. From the point of view of the AMPLE program, this simply means that only each group of voices with a unique MIDICHANNEL/MIDILINE selection can have an independent MIDIPROGRAM selection.

Often an instrument will use a different program numbering scheme for its front panel, for example, shifted up by one to give 1-128, or with separate bank and program numbers into which it converts MIDI's program numbers 0-127 when received. This means that the MIDIPROGAM number for a certain program may not be that shown on the front panel when it is selected manually. Also, some

receivers extend the number of programs by using alternative banks of 128 programs. MIDIPROGRAM selects programs from the current bank, but there is no standard message to set bank number.

examples

```
  3 SHARE 1 VOICES MIDIV     % set up voice and
    34 MIDIPROGRAM           % select sound number 34

  "leadsyn" [ 34 MIDIPROGRAM ]    % named selections for
  "softsyn" [ 32 MIDIPROGRAM ]    % convenient use in piece

  % word definition to test program change by entered value
  "progtest" [
  REP(
   "Enter value: "$OUT $IN        % get value from user input
   VAL $2                         % convert string to number
   IF(                            % if number found,
    MIDIPROGRAM                   % copy number and send
    C                             % play a note
   )IF
  )REP ]                          % repeat until ESCAPE pressed
  READY 1 VOICES MIDIV                % set up a voice
  progtest
  ...
  RUN                             % set piece playing
  2 SHARE                         % share a MIDI part
  progtest                        % send program changes to it

  % word to set program number in 1-128 range, to suit instrument
  "cprog" [ % number cprog
  1 #-                    % convert to 0-127
  MIDIPROGRAM ]           % send as normal
  1 cprog                 % equivalent to  0 MIDIPROGRAM

  % word to set program number using 3-digit 8/8/2 format
  "junoprog" [ % patchnum banknum groupnum junoprog
  1 #- 64 #*            % convert group
  #12                   % get banknum
  1 #- 8 #*             % convert banknum
  #213                  % get patchnum
  1 #-                  % convert patchnum
  #+ #+                 % add to produce MIDI 'program' number
  MIDIPROGRAM ]         % send
  2 3 1 junoprog        % select patch 2, bank 3, group 1
                        % equivalent to  18 MIDIPROGRAM

  % bank & program select
  % user selects bank manually - for use in initial mix only
  "bankprog" [ % programnumber banknumber bankprog
```

```
"Select bank number " $OUT NOUT NL   % command to user
MIDIPROGRAM ]                        % send 'program'
1 2 bankprog         % select prog 1 in bank 2

% program number mapper – calls number from stored list
"progmap" [ % myprognumber progstring progmap
FOR( VAL #2 #2 $STRIP )FOR           % discard leading numbers
VAL $2                 % get the required number
MIDIPROGRAM            % send
$2 ]                   % discard trailing program numbers
...
"fzprog" [ % myprognumber fzprog     % define my program map
"21 33 34 54 67 102" progmap ]       % program numbers nos. 0-5
...
2 fzprog               % call fzprog no. 2 - does  34 MIDIPROG.
...
"czprog" [ % herprognumber czprog    % define 2nd program map
"3  4  45 46 50" progmap ]           % for independent use
```

related words MIDILINE MIDICHANNEL

further information

Note that MIDIV does not make a default program setting.

Some instruments take a noticeable amount of time to carry out a program change, during which they cannot play notes. To overcome this, the program may require a pause after the MIDIPROGRAM instruction, or more conveniently, after the mix containing it, for example:

```
"part1z" [ SCORE / ]
...
"1234-19zabc "PLAY   % part1z delays start of music
```

MIDIRT

assign a real-time control voice

MIDIRT assigns a real-time control voice to the current voice position. Unlike a normal, note-playing voice assigned by MIDIV, this generates clock and control messages for synchronisation of external 'sequencer'-type devices, including drum machines and stand-alone sequencers. MIDILINE is used to select the output line as normal, and is initially 1.

The clock is a regular 'tick' that is sent continuously and automatically by the voice to make external sequencers follow the beat and tempo of the AMPLE performance, including instant and continuous tempo changes made with =T, +T, and -T.

The real-time voice also allows the user to pause or resume the external sequence using music words, or (for special applications), sound words:

music word	sound word	effect
^	OFF GATE	stop play
X	ON GATE	resume from last position

These may be added to note-playing parts or used in an additional part reserved for clock control. Often, just a single X is used to set the sequencer going at the start of the music. Alternatively, each section may be marked with X or ^, and short pauses marked on or in between notes.

The MIDICHANNEL setting has no effect on real-time messages.

Apart from the fact that music events control the sequence rather than play notes, the real-time voice operates like a normal voice, so for example it can be used to send additional real-time messages with MIDIOUT.

It is only useful to assign one real-time voice in a program. If clock control is required from more than one player, SHARE may be used to access the single real-time voice.

examples

```
1 VOICES MIDIRT      % set up real-time voice, sending clock

% to run synched device such as drum machine in a piece
"mix9" [
9 SHARE 1 VOICES MIDIRT    % add this line to mix for sync
PNUM SHARE ]               % (statutory 'end-of-mix' line)
"part9a" [ X ]  % 'clock part' simply starts the
...             %   synched device playing
"129-9a"PLAY    % include clock part and mix in performance
"12-9a"PLAY     % play without clock part

% alternative mix for drum machine connected to output line 3
"mix9" [
9 SHARE 1 VOICES MIDIRT    % as before,
3 MIDILINE                 % but use output line 3, so keeping
PNUM SHARE ]               % lines 1 & 2 free for notes

% mix word for keyboards plus drum machine
"mix9" [
```

```
  5 SHARE 8 VOICES MIDIV     % first MIDI part for notes
    1 MIDILINE
  6 SHARE 8 VOICES MIDIV     % second MIDI part for notes
    2 MIDILINE
  9 SHARE 1 VOICES MIDIRT    % third MIDI part for clock
    3 MIDILINE               % and real-time controls
  PNUM SHARE ]

  % part to play drums in some sections, but not others
  "part9a" [ ^ ]      % rest in section a
  "part9b" [ X ]      % play in section b
  "part9c" [ X ]      % play in section c
  "part9d" [ ^ ]      % rest in section d
  ...
  "129-9abcad"PLAY

  % general pause/fermata, pausing all parts inc. external sync
  "mix9" [                        % mix word
  1 SHARE                         % on the same part as
   8 VOICES MIDIV 1 MIDILINE      % note-playing voices,
   9 VOICE MIDIRT 2 MIDILINE      % put the real-time voice.
  PNUM SHARE ]
  "part1a" [ SCORE
  ... 12,
  eBagfedcb(9;^)/// /     % pause for five beats on d
  c(9;X)/D/c/b/           % resuming on c
  ... ]
  % alternative using 'fermata' word
  "fermata" [ % lengthnumber fermata
  (9;^)           % pause
  DURATION        % wait (DURATION doesn't add to bar length)
  (9;X)           % resume
  ]
  "part1a" [ SCORE
  ... 12,
  eBagfedcb 60 fermata    % pause for five beats on d
  c/D/c/b/                % resuming on c
  ... ]
```

related words VOICE VOICES UNUSED

further information

There is a total of 32 available MIDI voices, both note-playing and real-time. The Nucleus word UNUSED may be used to free a voice for re-use at another position. READY frees all voices.

The real-time voice's clock output ceases when the voice is freed by UNUSED or READY.

Normally, MIDIRT translates the first X or ON GATE into a MIDI 'start' message, but subsequent X's and ON GATE's are translated into MIDI 'continue' messages. However, some applications require the first message also to be 'continue', rather than 'start', for example, where a song position pointer is used to set the start position. The real-time voice provides for this through the use of note events or PITCH - any pitch setting on the voice clears the pending 'start' so that the next X or ON GATE sends 'continue'. So, to disable the 'start' before the first X, a slurred note, for example ~C, is included at some point before the X (because the note is slurred, it has no gate effect and therefore does not itself send a 'continue' message). Alternatively, a normal note, for example C, is used in place of the first X – its pitch effect clears the 'start' and then its gate effect acts exactly as X.

MIDI's timing clock uses 24 ticks per quarter note, and AMPLE uses 48, so the AMPLE clock is divided by two for transmission over MIDI. The first AMPLE tick after the MIDIRT is sent, followed by every second one after that.

MIDIV assign a voice

MIDIV assigns a voice at each of the current voice positions, and is used to direct the musical part's notes and other information to connected MIDI instruments.

The commonest use of MIDIV is in the piece's MIDI 'mix' word, where each part that is to be played on a MIDI instrument is given the required number of voices with an instruction line like this:

```
3 SHARE 8 VOICES MIDIV     % part 3 has 8 voices on MIDI
```

Where more than one MIDI instrument is in use, MIDIV is usually followed by instructions which select the instrument. MIDILINE selects the output line, and therefore the instrument connected to that line. Where there are two or more instruments on a single line, set to receive on different channels, MIDICHANNEL is used to set the channel number. MIDICHANNEL is also used to select between groups of voices in an instrument that can receive on more than one channel simultaneously.

With MIDIV voices set up on a part, the music played on that part will be directed to the MIDI instrument. All AMPLE music notation elements are reproduced except slurred notes which, being outside MIDI's key-based vocabulary, are ignored.

As well as playing the part's notes, hits and rests, the voice can provide additional sound control (where supported by the

instrument in use) through words that are included after MIDIV to apply throughout the music, or in the score to make changes between and within notes. The standard MIDI functions are provided by MIDIPROGRAM, MIDIBEND, MIDIPRESSURE and MIDICHPRESSURE, and more general sound control, and MIDI reception mode control, by MIDICONTROL. Specific and advanced applications may define words for new and non-standard MIDI functions with MIDIOUT AND MIDIWOUT.

examples

```
% simple test
READY                       % ready system
1 VOICES MIDIV              % assign a single voice
C                           % play middle C

% MIDI 'mix' to put voices in piece
"mix9" [
1 SHARE 8 VOICES MIDIV      % put 8 MIDI voices on part 1
PNUM SHARE ]
...
"123-19ababc "PLAY          % play piece, using MIDI voices

% voices for instrument on specific line and channel
"mix9" [
1 SHARE 8 VOICES MIDIV          % put 8 MIDI voices on part 1
 2 MIDILINE 10 MIDICHANNEL      % set line and channel for
                                % connected instrument
PNUM SHARE ]
...
"123-19ababc "PLAY              % play piece using MIDI mix

% MIDI voices on more than one part
"mix9" [
5 SHARE 8 VOICES MIDIV          % part 5 has 8 voices
 1 MIDILINE                     %   on instrument on line 1
6 SHARE 8 VOICES MIDIV          % part 6 has 8 voices
 2 MIDILINE                     %  on instrument on line 2
PNUM SHARE ]
```

related words VOICE VOICES MIDIRT UNUSED

further information

There are total of 32 available MIDI voices, including both note-playing and real-time. The Nucleus word UNUSED may be used to free a voice for re-use at another position. READY frees all voices.

If a voice is still playing when it is freed by UNUSED or READY,

it is silenced. Voices are also silenced by ESCAPE presses, STOP, errors in players 1-10, and commands like NEW, LOAD and DELETE which re-arrange user memory. The effect of silencing a voice is the same as playing a rest on it, so the sound may decay naturally rather than being cut off instantly.

PITCH, VEL, and GATE (the standard voice sound event words for music event interpretation) are not included in the M2 module itself, but where they are included in another module in the installation, they may be used as normal for direct control of MIDIV and MIDIRT voices. Initially, a MIDIV voice has the following settings:

```
0 PITCH  64 VEL  OFF GATE
```

On a MIDIV voice, the range of PITCH is -60 to 67 with 0 being middle C, though the MIDI instrument will not necessarily support this complete range. The range of VEL on a MIDIV voice is 0 to 127 (placing the SCORE initial setting of 64 in the centre of the range).

Each MIDIV voice translates PITCH, VEL and GATE events into MIDI's key-on and key-off ('note-on' and 'note-off') messages. PITCH and VEL events simply store their values, and these are then used by the next GATE on that voice to generate the appropriate key messages. Since it is not possible to send a change of pitch without a change of gate, the corresponding music event - a slurred note - will be ignored by a MIDIV voice. This is an inherent limitation of MIDI's keyboard-based structure.

Another possible problem arises from the fact that each MIDI channel can carry only a single keyboard-style part. Because each 'key' is either down or up, its pitch is only available to one voice at a time. If at any time two voices using the same channel try to play the same pitch, the instrument will give undefined results, so, for example, one cannot necessarily play a three-voice cannon on a single-channel three-voice MIDI instrument. The problem can also arise unexpectedly in transitions between chords, for example in:

```
C(GE) G(DA)
```

The second chord plays G on the first voice before the same G is released from the first chord's second voice. Transitory unisons can be avoided by the use of a small negative ’Len’ setting, ensuring that each chord is completely released before the next starts.

MIDIWOUT send a word

number MIDIWOUT

MIDIWOUT sends two bytes directly to the MIDI output line of the current voice. Bits 0-7 (the low byte) of the number are sent first, followed by bits 8-15.

MIDIWOUT is used for special applications including user definition of additional control instructions, and transmission of system exclusive data. It is a more-efficient alternative to MIDIOUT where pairs of bytes are sent.

examples

```
&10F3 MIDIWOUT       % send two bytes, &F3 then &10

% song select word (for use on existing voice)
% more efficient alternative to MIDIOUT version
"song" [ % number song % (1 <= number <= 128)
1 #-             % convert number to MLDI 0-127 format
#B12             % move to top byte
&F3 OR           % put 'song select' in bottom byte
MIDIWOUT ]       % send complete two-byte msg in one go

% song select word for use as dynamic section select
% in drum machine - sections are programmed as separate songs
% and called up during play using song select
"sect" [ % number sect % 1 <= number <= 127, 0 means stop
0,^              % stop sequence
#11 IF(
 1 #-            % convert number to MLDI 0-127 format
 #B12 &F3 OR     % move to top byte and add 'song select'
 MIDIWOUT        % send
 &FA MIDIOUT     % unconditional 'start' to begin new song
)ELSE( #2 )IF
]
READY 1 VOICES MIDIRT
3 MIDILINE
1 sect           % play song one
2 sect           % play sang two
0 sect           % stop
...
"mix9" [
...
9 SHARE 1 VOICES MIDIRT 2 MIDILINE
PNUM SHARE ]
"part9a" [ 1 song ] % assign songs to sections
"part9b" [ 2 song ]
... etc ...
"12349-9abcad"PLAY
```

related words MIDIOUT

further information

See MIDIOUT.

7 Implementation information

This chapter includes further information on the Music 2000's M2 module, including its implementations of the AMPLE module interface and the MIDI Specification.

MIDI message status

The M2 module uses the 'note on' message (&9n) for key-on events and standard key-off events. The 'note off' message (&8n) is used for key-off events which have velocity values.

The running status scheme is used to reduce the size of channel messages, as normal.

system 'silence' events

The M2 module responds directly to various system events that ask for silence, including serious errors, ESCAPE presses and execution of READY, STOP etc., by setting OFF GATE on all voices that are in the ON GATE state. On a MIDIV voice, this sends a 'note off' message, causing the note to decay naturally as if the key was lifted. On a MIDIRT voice, this sends a 'stop' message to stop the sequence, and thereby silences the voices controlled by the sequence. Running status is cleared before sending the messages, ensuring that a receiver that has lost track of the running status, due to a line break or similar, will have it restored.

M2 has no way of knowing whether there are notes playing on line channels other than those currently in use by the voices, so such notes cannot be silenced. M2 makes no attempt to silence voices by sending an 'all notes off' message.

module installation

The M2 module must be installed below address &3000 to function. This is guaranteed when using the supplied !BOOT file on a standard computer, but in alternative installations, be sure to include M2 immediately after INT in the !BOOT file.

MIDI implementation chart

[Computer system] Date: 13 Jan 1988
Model AMPLE/Music 2000 MIDI Implementation Chart Version: 1.0

Function		Transmitted	Remarks
Basic	default	1	
channel	changed	1-16	per line
	default	modes 3,4	per line
Mode	messages	omni on/off, mono, poly	
	altered	**********	
Note		0-127	
Number	true voice	**********	
Velocity	Note ON	0 &9n v = 1-127	
	Note OFF	0 &8n v = 1-127	also, &9n v = 0
After	key's	0	
Touch	ch's	0	
Pitch bender		0	14-bit resoln.
Control change		0	all controllers
Program		0 0 - 127	
Change	true #	**********	
System exclusive		0 any possible message	via MIDIOUT
System	Song pos	0	via MIDIOUT
Common	Song sel	0	via MIDIOUT
	Tune	0	via MIDIOUT
System	clock	0	
Real-time	commands	0	
Aux	Local on/off	0	via MIDIOUT
Messages	All notes off	0	via MIDIOUT
	Active sense	0	via MIDIOUT
	Reset	0	via MIDIOUT

Mode 1 : OMNI ON, POLY Mode 2 : OMNI ON, MONO 0 : Yes
Mode 3 : ONNI OFF, POLY Mode 4 : OMNI OFF, MONO X : No

8 MIDI Specification summary

This chapter gives a concise description of all MIDI messages defined in the MIDI 1.0 Specification, providing the programmer with information for generating messages directly from AMPLE, using MIDIOUT/MIDIWOUT and MIDICONTROL, for example.

introduction

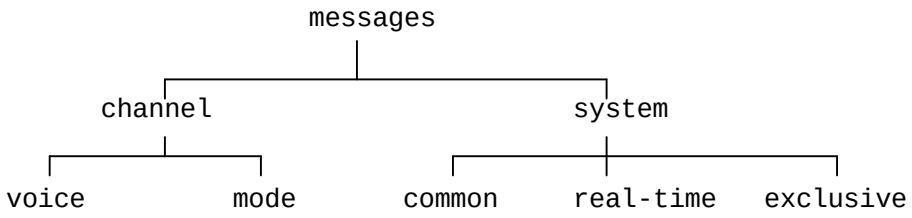
The MIDI line is the path connecting a single transmitter to a number of receivers, allowing the transmitter to send messages to any or all receivers.

bytes and messages

The byte is the smallest unit of MIDI communication. It consists of eight binary digits each of which can take the value 0 or 1. They are numbered 0 (least significant) to 7 (most significant) and give the byte 256 possible values, represented in hexadecimal as &00 to &FF.

The message is the fundamental functional unit of MIDI communication. Messages are made up of bytes. Each message consists of a status byte which describes the function of the message, plus zero, one, two or an indefinite number of data bytes. The status byte is always in the range &80-&FF, that is, bit 7 is always 1, and data bytes are in the range &00-&7F, that is, bit 7 is always 0.

There are a number of different messages with defined functions. They can be classed as follows:



channel messages

A **channel** is a parallel functional division of the MIDI line. **Channel messages** travel on particular channels, allowing independent streams of channel messages to travel along the same line. Each line has 16 channels, numbered 1 to 16. The status byte of a channel message determines the channel number of the message, as well as the function, as follows:

```
Bits 0-3:          channel number, 0-15
                    0 = channel 1
                    1 = channel 2
                    ...
                   15 = channel 16
Bits 4-6:          function
Bit 7:             1
```

voice messages

The voice is the fundamental sound-producing unit of an instrument which can play one note at a time. **Channel voice messages**, or **voice messages** for short, are those that control the receiver's voices. **Channel mode messages**, or mode messages for short, control the MIDI 'mode' of the receiver - the way it interprets messages received from MIDI and its own control panel.

The following voice messages are included:

function	status byte	data bytes	data description
Note Off	&8n	key number	0 - 127 (60 = middle C)
		velocity	1 – 127 (soft to hard)
Note On	&9n	key number	0 - 127 (60 = middle C)
		velocity	1 - 127 (soft to hard)
			0 equiv. to note off
Key pressure	&An	key number	0 - 127 (60 = middle C)
		pressure	0 - 127 (soft to hard)
Control change	&Bn	control number	0 - 12l (122-7 reserved)
		control value	0 - 127
Program change	&Cn	program number	0 - 127
Channel pressure	&Dn	pressure value	0 – 127 (soft to hard)
Pitch bend change	&En	value bits 0-6	0 – 127
		value bits 7-13	0 – i27

control numbers

The following control numbers correspond to specific physical controllers or control functions on most of those devices that provide them:

control number	control device/function
1	modulation wheel or lever
2	breath controller
4	foot controller (pedal)
5	portamento time
6	data entry
7	volume
10	pan (stereo position)

mode messages

The mode messages use the same status value as the control change message, with the first data byte indicating the function:

function	status byte	data bytes	data description
Local control	&Bn	122 on/off	0: off 127: on
All notes off	&Bn	123 0	
Omni off	&Bn	124 0	
Omni on	&Bn	125 0	
Mono on	&Bn	126 qty of channels	0: maximum available 1-16: specified number
Mono off	&Bn	127 0	

'Local control' determines whether the music keyboard and control panel of the receiving keyboard are operative or not.

'All notes off' is designed to silence all MIDI notes playing on the

receiver, but since there is no requirement for a receiver to recognise it and many do not, it is of limited use.

The remaining four mode messages set two variables, 'omni' and 'mono', each of which may be 'on' or 'off', and may also be set by physical controls on the receiver. Together they give a total of four possible reception modes, numbered 1, 2, 3 and 4:

mode number	omni state	mono state	
1	on	off	polyphonic use of instrument on its own line
2	off	on	monophonic use of instrument on its own line
3	on	off	polyphonic use if instrument on shared line, on its own channel
4	off	on	monophonic use of instruments voices on separate channels ('voice channels')

The 'omni' state determines whether messages will be received on all channels indiscriminately (omni on), or on only the basic channel of the receiver, set to one of 1 to 16 using its own controls. Initially when turned on, a receiver will have omni 'on', and this is suitable for operation when the receiver is connected to a MIDI line of its own. Where there are two or more receivers on the same line, they will normally have different basic channels selected so that, when omni is turned 'off' either by a panel control or the 'omni off' message, they receive independent streams of channel messages, to play different musical parts for example.

The 'mono' mode determines whether the receiver will assign received notes to all its voices (polyphony) or to just a single voice (monophony). The 'omni off, mono on' state, that is mode 4, is a special case which calls for the receiver to attach the specified number of its voices to individual channels, then called voice channels, from the basic channel upwards. Each voice on its own channel provides the direct and independent control of voices required by computer and other systems.

system messages

System messages are not channel-specific, so apply universally to all receivers on the line.

System real-time messages are provided to synchronise and control receivers that play their voices from internally stored scores or 'sequences', such as drum machines and sequencers. Each consists of a status byte only, with no data bytes. They include:

function	status byte	description
Timing clock	&F8	24 per quarter note
Start	&FA	play from start
Continue	&FB	play from current posn
Stop	&FC	stop playing
Active sensing	&FE	optional confirmation
System reset	&FF	reset to power-on state

The 'timing clock' message is a regular repeating 'tick' that marks the passage of musical time. It is followed by those receivers set to respond to it, so they play along synchronised to the transmitter. 'Start', 'continue' and 'stop' control the receiver's performance whether it is synchronised to the timing clock or not.

System common messages, are those system messages which do not fall in to either of the other classes of system messages. They include:

function	status byte	data bytes	data description
Song position pointer	&F2	beats bits 0-6 beats bits 7-13	1 beat = 1 semiquaver 1 sixteenth note
Song select	&F3	song number	0 – 127
Tune request	&F6		
end of system exclusive	&F7		

System exclusive messages are general messages of undefined function that may be used by manufacturers for specific purposes. Messages contain a manufacturer's identifier, followed by any number of data bytes, followed by the 'end system exclusive' message:

function	status byte	data bytes	data description
system exclusive	&FO	identifier	0 - 127
		any number of data bytes	0 - 127
	&F7		(end of system exclusive)