AKAI professional

Z4 Z8

MPC4000
Music Production Center

MIDI System Exclusive Protocol Specification

(OS Version 1·50)

(This Page has been left intentionally blank.)

List of Tables

(This Page has been left intentionally blank.)

Introduction

This document details the specification for the MIDI System Exclusive protocol for the Akai Z4 and Z8 samplers and the sampling engine of the Akai MPC4000 Music Production Centre. System Exclusive (or SysEx) is a feature of MIDI which allows custom information to be sent to an instrument, making it possible for a computer (or anything else which can send customised MIDI SysEx messages) to remotely control and configure the samplers.

On the samplers, SysEx messages are decoded on each port independently (*A* or *B*), so both ports can be used at the same time if desired[†]. If the SysEx Manufacturer ID[‡] is not AKAI <&47{71}>, the entire SysEx message is ignored. To allow for feedback from a SysEx message, the *out* port is used to send SysEx confirmation back to the controller (port A_{out} confirms data received by A_{in} and port B_{out} confirms data received by B_{in}).

The SysEx messages received by the samplers are buffered, so it is possible to send several messages without pauses. However, if this is done, it is possible that the internal buffers of the samplers will fill up, resulting in lost data. Therefore, it is recommended that the confirmation messages are used to ensure that data was received and processed correctly.

In this document, hexadecimal notation is used where appropriate and designated by the "&" symbol. Where this is used, decimal values are also given as follows: &HEX{DECIMAL}.

Note: Several functions provided by the System Exclusive specification can take a noticable amount of time to complete, which may interrupt normal (musical) MIDI processing. Therefore, it is recommended that only those functions specifically designed for real-time control (*e.g.,* adjustment of some Multi parameters) be used when normal MIDI information is being sent. Such real-time items are marked with the ($_{\text{RT}}$) symbol in this document.

[†] If ports *A* and *B* are used *simultaneously* for SysEx transmissions, care should be taken to ensure that the function performed on one port does not depend on the completion of a function on another port.

[‡] The standard format of a SysEx message is "<&F0> <Manufactured ID> … <&F7>", where the Manufacturer ID is assigned by the MIDI Manufacturers' Association. The assigned ID for AKAI is &47{71}. The use of a manufacturer ID ensures that instruments from other manufacturers will ignore SysEx messages not intended for them.

Modification History[†]

The first version of this specification was for use with Operating System Version 1·00.

Operating System Version 1·50

- Corrected error in *Disk Tools* section: incorrectly documented as Section=&10, should be &20.
- Added protocol for determining the state of, and cancelling of, asynchronous operations.
- Added support for Real-time Keygroup Zone Crossfade.
- New Sysex for Program, Keygroup and Zone Automation.
- Added support for MPC4000 pad editing.
- Added settings for CDR write speed and test enable.
- Program Automation Sections added [&68 {104} – &69 {105}].
- Several additional errors were corrected throughout the document.

[†]. AKAI professional M.I. Corp. reserve the right to change this SysEx specification without prior notice. However, such changes are likely to be minimal, only being implemented to improve the performance of the product. If you encounter problems, please ensure that you are using the latest SysEx document.

System Exclusive Protocol

The System Exclusive feature is used to remotely control and configure the sampler. To provide feedback to the host (*e.g.,* a PC or MAC) the MIDI out port is used to transmit SysEx confirmation messages upon reception of and after processing of viable SysEx messages. If desired, this confirmation can be turned off via SysEx commands.

Control Message Format

To allow several devices to coexist on the same MIDI bus used for system exclusive messages, several identifiers are used by the Z4/Z8/MPC4000 to ensure that it only responds to those messages which are intended for it. Thus, all system exclusive messages begin with the following bytes:

`<Start of SysEx> <AKAI ID> <Z4/Z8 ID> <User-selectable Device ID> <User-Refs..> …`

Where the values of the bytes are: `<&F0> <&47> <&5F> <0..&7F> …`
`{<240> <71> <95> <0..127> …}`

AKAI ID <&47{71}> and Z4/Z8/MPC4000 ID <&5F{95}>

The AKAI ID and the Z4/Z8/MPC4000 ID ensure that only AKAI Z4/Z8/MPC4000 samplers respond to these messages.

User-Selectable Device ID <0..&7F{0..127}>

The user-selectable DeviceID allows more than one AKAI Z4/Z8/MPC4000 sampler to coexist on the same MIDI bus, but be configured independently via SysEx. The default DeviceID is zero.

The DeviceID has been limited to the range: 0–31. This is because the top 2 bits (bits 5 and 6) are used to determine the number of User-Ref bytes being sent—bits 0–4 represent the DeviceID. The number of User-Refs expected is as shown in Table 1. This then provides a flexible means of sending User-Ref bytes; zero bytes to conserve bandwidth, more bytes if required by your application. Moreover, this method allows the number of User-Refs sent to vary on a per-message basis.

Table 1: Number of User-Refs Being Sent

Device ID bit 6	Device ID bit 5	Num User-Refs
0	0	0
0	1	1
1	0	2
1	1	3

Note: If the Z4/Z8/MPC4000 is set to have a user-selectable DeviceID of zero (0), then it will respond to *all* SysEx messages regardless of the DeviceID transmitted. Similarly, if a DeviceID of zero is transmitted by the controller (*i.e.,* bits 0–4 = 0), *all* Z4/Z8/MPC4000s will respond regardless of which DeviceID is set.

For non-zero DeviceIDs, the sampler will only respond to SysEx messages if its DeviceID matches that sent.

User-Refs <0..&7F{0..127}>

The User-Ref parameters can be set to any value. It is only useful when confirmation messages are enabled (or if a REPLY is requested) where the User-Ref parameter is echoed with every confirmation message. This is to allow flexibility in the design of control software where each SysEx message can be *stamped* with an ID so that the confirmation messages can be matched to the commands sent. This is especially useful if the control software sends buffered messages out-of-sequence.

Either 0, 1, 2, or 3 User-Refs can be sent with any message. To allow for this, the top 2 bits of the DeviceID contain a count of the number of User-Refs which will be sent. The default number of User-Refs is 0: see Table 1 for more details.

In this document, optional User-Refs will be shown as: <…>.

Checksums

The checksum provides a means of error-detection in the SysEx message. The checksum is a single data byte sent as the last item before the End-of-SysEx byte <&F7 {247}>. By default, checksums are disabled making sending SysEx messages as simple as possible. Enabling and disabling of checksums can only be done via SysEx messages.

Note that because the SysEx specification for the Z4/Z8/MPC4000 supports a variable number of parameters, if checksums are disabled, the calculated checksums may still be transmitted; although they will be ignored.

A checksum calculation begins at the first User-Ref byte (if any) — there is no point in calculating the checksum earlier than this because if an error occurs in the first bytes, the SysEx message will be ignored anyway — and the calculation stops before the End-of-SysEx byte.

To calculate the checksum, unsigned 8-bit addition is used, which wraps on overflow (*i.e.,* 255+1 = 0). To ensure compatibility with the MIDI data byte specification, the high-bit of the checksum is set to zero (logical AND with &7F {127}) once the checksum has been calculated.

For example, given the following SysEx message:

```
<&F0> <&47> <&5F> <&05> <&10> <&0C> <&1B> <&35> <&6D> <&F7>
  {<240> <71> <95> <5> <16> <12> <27> <53> <109> <247>}
```

The checksum would be calculated as:

```
(&10 + &0C + &1B + &35 + &6D) = &D9   ->   (&D9 AND &7F) = &59
 {(16 + 12 + 27 + 53 + 109) = 217   ->   (217 AND 127) = 89}
```

And the new checksummed message would be:

```
<&F0> <&47> <&5F> <&05> <&10> <&0C> <&1B> <&35> <&6D> <&59> <&F7>
   {<240> <71> <95> <5> <16> <12> <27> <53> <109> <89> <247>}
```

A useful tip — to turn off Checksums on all Z4/Z8/MPC4000s, regardless of DeviceID, send:

```
<&F0> <&47> <&5F> <&00> <&00> <&04> <&00> <&04> <&F7>
     {<240> <71> <95> <0> <0> <4> <0> <4> <247>}
```

A complete message

The format of a complete control message is as follows:

```
<&F0> <&47> <&5F> <0..&7F> <…> <Section> <Item> <Data1> … <DataN> <checksum> <&F7>
```

This allows the selection of various *Sections*, such as “Multi”, “Sample”, “Program”, “Config”, and the variable number of data parameters allows the efficient passing of strings (which, for example, are used to name programs, multis, etc.). The Section numbers are detailed in Table 4.

If extensive use is made of the System Exclusive protocol, it is recommended that checksums are enabled so that transmission errors can be detected and handled.

Confirmation Messages

The confirmation message is a 7 or more byte SysEx message:

```
<&F0> <AKAI ID> <Z4/Z8 ID> <Device ID> <User-Ref…> <Reply ID> <Section> <Item> <Data1> …
                              <DataN> <checksum> <&F7>
```

The first 6 bytes (or more if `<User-Refs>` are used) are the same as those transmitted in the original message, except that the `<Reply ID>` item has been inserted. This format ensures that confirmation messages from different devices can be distinguished, and the insertion of the `<Reply ID>` item ensures that confirmation messages are not confused with other SysEx messages. Moreover, the `<User-Ref>`, `<Section>` and `<Item>` can be used by a controlling computer to determine which SysEx message generated the confirmation message. The values of Reply ID and Data1 … DataN are explained in Table 2. Note that the `<checksum>` will only be sent if checksums are enabled (see Table 6).

Table 2: SysEx Confirmation Messages

Reply ID	Data1	Data2	Meaning
&4F {79} ‘O’	*NA*	*NA*	“OK” Valid SysEx has been received and is being processed.
&44 {68} ‘D’	*NA*	*NA*	“DONE” SysEx instruction has been completed successfully.
&52 {82} ‘R’	Reply Type	Reply 1…	“REPLY” A variable number of bytes is returned as a reply (data returned depends on SysEx message sent). Every piece of reply data is preceded with a single ID byte indicating the type of data following, making it easier to interpret the data received. The values of these ID bytes are shown in Table 5.
&45 {69} ‘E’	<LSB 0–127>	<MSB 0–127>	“ERROR” An error has occurred, Error Number = Data1 + 128×Data2

Note that because the user-selectable DeviceID is returned in these messages, a controller can establish the number of devices and their DeviceIDs connected in a chain by using the “Query” SysEx command with the DeviceID set to be zero (all devices), which will return an “OK” and a “DONE” message from each sampler in the chain. The DeviceID will also have it’s top 2 bits set to show how many `<User-Ref>` bytes are included in the message (see Table 1 for more information). The number of and the values of `<User-Ref>` bytes in the confirmation message will always be the same as those in the message which caused the confirmation message to be generated.

The normal flow of confirmation messages is that the “OK” message will be transmitted as soon as a valid Sysex message (i.e., manufacturer = &47 {71}, model = &5F {95}, DeviceID = set value) has been received. If there is an error in this message, or the message is unsupported, then the “ERROR” confirmation message will be returned—possible error numbers are explained in Table 3. If the message is supported, the function will be performed then, once processing is complete, the “DONE” confirmation message will be transmitted. Alternatively, if a request for information was issued, the “DONE” message will be replaced by a “REPLY” message with the appropriate data contained within it. Note that it is possible, but unlikely, that a “REPLY” message may be followed by an “ERROR” message if an error occurred during the generation of the reply.

Note: Although “OK”, confirmation messages can be turned on and off via SysEx, the “DONE” “REPLY” and “ERROR” messages cannot. This is to ensure that at least one message is returned for every SysEx message received; thus making synchronisation of a controller easier.

Table 3: Error Numbers Returned

Error Number	MSB	LSB	Description of Error
	General Errors		
&00 {0}	0	0	The <Section> <Item> supplied are not supported
&01 {1}	0	1	Checksum invalid
&02 {2}	0	2	Unknown error
&03 {3}	0	3	Invalid message format
&04 {4}	0	4	Parameter out of range
&05 {5}	0	5	Operation is pending
	System Errors		
&80 {128}	1	0	Unknown system error
&81 {129}	1	1	Operation had no effect
&82 {130}	1	2	Fatal error
&83 {131}	1	3	CPU memory is full
&84 {132}	1	4	WAVE memory is full
	Item Errors		
&100 {256}	2	0	Unknown item error
&101 {257}	2	1	Item not found
&102 {258}	2	2	Item in use
&103 {259}	2	3	Invalid item handle
&104 {260}	2	4	Invalid item name
&105 {261}	2	5	Maximum number of items of a particular type reached
&120 {288}	2	32	Keygroup not found
	Disk Errors		
&180 {384}	3	0	Unknown disk error
&181 {385}	3	1	No Disks
&182 {386}	3	2	Disk is invalid
&183 {387}	3	3	Load error
&184 {388}	3	4	Create error
&185 {389}	3	5	Directory not empty
&186 {390}	3	6	Delete error
&187 {391}	3	7	Disk is write-protected
&188 {392}	3	8	Disk is not writable
&189 {393}	3	9	Disk full
&18A {394}	3	10	Disk abort
	File Errors		
&200 {512}	4	0	Unknown file error
&201 {513}	4	1	File format is incorrect

Table 3: Error Numbers Returned

Error Number	MSB	LSB	Description of Error
&202 {514}	4	2	WAV format is incorrect
&203 {515}	4	3	File not found
&204 {516}	4	4	File already exists

Control Messages

The functions of the Z4/Z8/MPC4000 which can be controlled via SysEx are grouped into *Sections*. For example, there is a section to configure a Multi, and a section to change the MIDI configuration. These Sections then have several functions associated with them, called an *Item*. Thus each SysEx control message consists of the appropriate device header:

```
<&F0> <AKAI ID> <Z4/Z8 ID> <Device ID> <User-Ref…>
```

then the Section and Item number:

```
<Section> <Item>
```

followed by the appropriate data for that command. The defined Sections are shown in Table 4.

Table 4: Description of <Section> Parameter

<Section>	Description of Section
&00 {0}	SysEx Configuration
&04 {4}	System Setup
&0C {12}	Keygroup Zone Manipulation
&10{16}	Keygroup Manipulation
&14 {20}	Program Manipulation
&18 {24}	Multi Manipulation
&1C {28}	Sample Tools
&20 {32}	Disk Tools
&24 {36}	Multi FX Control
&28 {40}	MIDI song file tools
&2C {44}	Front Panel Control
&30 {48}	Recording
&44 {68}	<Reserved>
&45 {69}	<Reserved>
&4F {79}	<Reserved>
&52 {82}	<Reserved>
&60 {96}[a]	Alternative (by-HANDLE) Sample Tools
&61 {97}[a]	Alternative (by-HANDLE) Keygroup Zone and Keygroup
&62 {98}[a]	Alternative (by-HANDLE) Program
&63 {99}[a]	Alternative (by-HANDLE) Multi
&64 {100}[a]	Alternative (by-HANDLE) Multi FX Control

Table 4: Description of <Section> Parameter

<Section>	Description of Section
&65 {101}[a]	Alternative (by-HANDLE) Song File Control
&68 {104}[b]	Keygroup and Zone Automation
&69 {105}[b]	Program Automation

a. These represent an alternative means of control, where the operation is performed on the item (Multi, Program or Sample) specified by handle, rather than on the currently selected item.

b. These operations are intended for automation of Program parameters during playback of a sequencer. Any Program, Keygroup, or Keygroup Zone parameter can be modified for any part in the current Multi.

Format of Message Data

The Z4/8 SysEx protocol supports variable-length messages, making it both flexible and adaptable. Therefore, it is possible to send differently formatted data depending on the function to be performed and in some cases, more than one piece of data is required in a single message. The format of the data required in the messages should be strictly followed to avoid problems — although in most cases, an incorrect message will simply generate an ERROR confirmation message.

> Note: Data bytes sent within SysEx messages must not exceed a value of 127 (or &7F). This limitation is imposed by the MIDI specification. Failure to observe this limit may lead to undefined behaviour!

Numeric data is always sent as 7-bit bytes (*i.e.,* the top bit of each byte must be zero). Numbers which require more than one byte to represent them are always sent least-significant byte first. Character strings can be any length, but must always have a terminating character of value zero. A summary of all the data types is given in Table 5. Note that the data format ID is the value returned with every piece of reply data in confirmation messages, a feature which makes automatic decoding of replies easier.

Table 5: SysEx Data Formats[a]

Name	ID	Byte Format	Description (Range)
BYTE	1	<value>	byte (0–127)
SBYTE	2	<sign><value>	signed byte (±127)
WORD	3	<LSB><MSB>	word (0–16383)
SWORD	4	<sign><LSB><MSB>	signed word (±16383)
DWORD	5	<LSB><SB1><SB2><MSB>	double word (0–268,435,455)
SDWORD	6	<sign><LSB><SB1><SB2><MSB>	signed double word (±268,435,455)
QWORD	7	<B1><B2><B3><B4><B5><B6><B7><B8>	quad word (0–72,057,594,037,927,935)
SQWORD	8	<sign><B1><B2><B3><B4><B5><B6><B7><B8>	signed quad word (±72,057,594,037,927,935)
STRING	9	<char1><char2>…<charN><0>	Null-terminated string
2BYTES	10	<value1><value2>	2 data bytes (*command-specific*)
3BYTES	11	<value1><value2><value3>	3 data bytes (*command-specific*)
CUSTOM	32	specific to command	Custom data (*command-specific*)

[a] <sign>: 0 = positive, 1 = negative.

Arrays

If an array of values is to be sent, this is indicated by [n] following the data type. For example, an array of signed bytes with 10 entries would be shown as SBYTE[10].

Use of Ellipsis

The ellipsis, "…", is used to illustrate that more data may be transmitted than the explicit data values shown in the tables. For example, <Data1>…<DataeN> means that there may be additional bytes between Data1 and DataN. The number of additional bytes depends on the both format and the content of the data being sent.

Item List for SysEx Configuration section [&00{0}]

These options control how the sampler responds to SysEx messages. These options only apply to the MIDI port on which the SysEx command was sent. This allows different applications to coexist on different ports without interfering with the other's communications. For example, problems would arise if port A disabled checksums when port B required them.

> Note: To ensure that operations on one port do not interfere with those on another, LCD synchronisation should be turned off when it is not essential (*e.g.,* if editing a program) and only used when required, *e.g.,* when selecting the current multi to be played.

Table 6: Control Items for Section &00{0} — SysEx Configuration

<Item>	<Data1>	<Data2>	Description of Item
&00{0}$_{\text{RT}}$	*NA*	*NA*	"Query" — use with user-selectable DeviceID=0 to get an "OK!" and a "DONE" reply with DeviceID returned
&01{1}$_{\text{RT}}$	BYTE(0, 1)	*NA*	Enable/Disable received message notification ("OK!") <Data1> = (0=OFF, 1=ON)[a]
&03{3}$_{\text{RT}}$[b]	BYTE(0, 1)	*NA*	Enable/Disable synchronisation between the currently selected samples/programs/multis and those displayed on the front-panel <Data1> = (0=OFF, 1=ON)[a]
&04{4}$_{\text{RT}}$	BYTE(0, 1)	*NA*	Enable/Disable checksum verification <Data1> = (0=OFF, 1=ON)
&05{5}$_{\text{RT}}$	BYTE(0, 1)	*NA*	Enable/Disable automatic screen updating when a SysEx message is processed <Data1> = (0=OFF, 1=ON)[c]
&06{6}$_{\text{RT}}$	BYTE	BYTE, BYTE, BYTE	*Echo Message*: a special test function which will echo all 4 data bytes by returning them as a Reply. This is useful when debugging a controlling program.
&07{7}$_{\text{RT}}$[d]	BYTE(0, 1)	*NA*	Enable/Disable "*Still Alive"* monitor <Data1> = (0=OFF, 1=ON)[c]
&08{8}	BYTE(0, 1)	*NA*	Enable/Disable synchronisation between the current playback item and the playback item selected on the front-panel <Data1> = (0=OFF, 1=ON)[a]
&10{16}$_{\text{RT}}$	*NA*	*NA*	Get SysEx Buffer Size

a. The state of this option at power-on is ON.

b. Note that if synchronisation is enabled and the current muti, program or sample is changed by SysEx on a different port which also has synchronisation enabled, the currently selected item on the current port will also change because the item displayed on the LCD will have changed. To avoid this situation, synchronisation should be turned off, and enabled only when required.

c. The state of this option at power-on is OFF.

d. Some SysEx messages may require substantial time to execute. This can result in large delays between an OK and a DONE (or REPLY/ERROR) message which the host could interpret as "samper not responding". To avoid this, if the *Still Alive* monitor is enabled, a NULL message (`<&F0><&F7>`) will be transmitted to the host approximately every second, whilst the host is awaiting a response.

Table 7: Format of "REPLY" confirmation messages for Section &00{0} — SysEx Configuration

<Item> requested	<Reply>…	Description of Data Returned
&10{16}	WORD	Get SysEx Buffer Size

Item List for System Setup section [&04{4}]

This System section contains several general settings which control the behaviour of the sampler.

When information about the System Setup is requested with a *Get* message, the data is returned in a “REPLY” confirmation message (see *Confirmation Messages* on page 5). The format of these messages is summarised in Table 9 and Table 11.

Table 8: Control Items for Section &04{4} — System Main

<Item>	<Data1>	<Data2>	Description of Item
	General		
&00 {0}	*NA*	*NA*	Get Operating System Software Version
&01 {1}	*NA*	*NA*	Get the Sub-Version of the Operating System
&04 {4}	*NA*	*NA*	Get Sampler Model
&08 {8}	*NA*	*NA*	Get List of supported filetypes
	Memory Information		
&10 {16}	*NA*	*NA*	Get the percentage free Wave memory
&11 {17}	*NA*	*NA*	Get the percentage free CPU memory
&12 {18}	*NA*	*NA*	Get the total number of kilobytes of Wave memory
&13 {19}	*NA*	*NA*	Get the number of kilobytes of free Wave memory
&18 {24}	*NA*	*NA*	Clear Sampler Memory (delete all items from memory)
&19 {25}	BYTE	*NA*	Purge Unused Items <Data1> = (0=SAMPLE, 1=PROGRAM)
&1A {26}	BYTE	*NA*	Tag Unused Items <Data1> = (0=SAMPLE, 1=PROGRAM)
	Wave Memory Compacting Functions		
&20 {32}	*NA*	*NA*	Start Compact Wave Memory
&21 {33}	*NA*	*NA*	Cancel Compact Wave Memory
&22 {34}	*NA*	*NA*	Get Compact Wave Memory Progress (%)
	Asynchronous Operation Control		
&30 {48}	*NA*	*NA*	Get State of Asynchronous Operation
&31 {49}	*NA*	*NA*	Cancel Current Asynchronous Operation

Table 9: Format of “REPLY” confirmation messages for Section &04{4} — System Main

<Item> requested	<Reply>…	Description of Data Returned
&00 {0}	2BYTES	Operating System Version: <Value1>=major version number, <Value2>=minor version number
&01 {1}	BYTE	Operating System Sub-Version
&04 {4}	BYTE(0–2)	Sampler Model <Reply1> = (0=Z4, 1=Z8, 2=MPC4000)
&08 {8}	STRING[]	Get List of supported filetypes <Reply…> contains a list of file extensions which are supported
&10 {16}	BYTE	Get the percentage free Wave memory
&11 {17}	BYTE	Get the percentage free CPU memory
&12 {18}	DWORD	Get the total number of kilobytes of Wave memory

Table 9: Format of “REPLY” confirmation messages for Section &04{4} — System Main

<Item> requested	<Reply>…	Description of Data Returned
&13 {19}	DWORD	Get the number of kilobytes of free Wave memory
&22 {34}	BYTE	Get Compact Wave Memory Progress (%)
&30 {48}	*(see footnote[a])*	Get State of Asynchronous Operation

---

a. If the asynchronous operation is complete, a DONE confirmation message will be returned. If the operation is still pending, an ERROR confirmation message will be returned with the error “Operation is pending”.

Table 10: Control Items for Section &06{6} — System Parameter Set

<Item>	<Data1>	<Data2>…	Description of Item
	Global Options		
&01 {1}[a]	STRING	*NA*	Set Sampler Name
&02 {2}	BYTE(0−7)	*NA*	Set SCSI self ID
&03 {3}	SWORD (0−±3600)	*NA*	Set Master Tune
&04 {4}	BYTE(0−7)	*NA*	Set Master Level <Data1> = (–42 dB – 0 dB in 6 dB steps) (0=–42 dB, 1=–36 dB, …, 7=0 dB)
&05 {5}	BYTE(0, 1)	BYTE(0, 2)	Set MIDI OUT/THRU <Data1> = MIDI port (0=A, 1=B), <Data2> = (0=OUT, 1=THRUA, 2=THRUB)
&06 {6}	BYTE(0, 1)	*NA*	Set Qlink Local Control <Data1> = (0=OFF, 1=ON)
&07 {7}	BYTE(0, 1)	*NA*	Set Create Default Items at Startup <Data1> = (0=OFF, 1=ON)
&08 {8}	BYTE(0, 1)	*NA*	Set MIDI file save format
&09 {9}	BYTE(0−7)	*NA*	Set CD-R write speed <Data1> = (0=×1, 1=×2, 2=×4, 3=×6, 4=×8, 5=×12, 6=×16, 7=MAX)
&0A {10}	BYTE(0−2)	*NA*	Set CD-R write mode <Data1> = (0=TEST+WRITE, 1=TEST ONLY, 2=WRITE ONLY)
	Display Options		
&10 {16}	BYTE(0, 1)	*NA*	Set Front panel lock-out state <Data1> = (0=NORMAL; 1=LOCKED)
&11 {17}	BYTE(0−19)	*NA*	Set Display Contrast
&12 {18}	BYTE(0, 1)	*NA*	Set Note Display <Data1> = (0=NUMBER, 1=NAME)
&13 {19}	BYTE(0−3)	*NA*	Set Date Display Format <Data1> = (0=DDMMYY, 1=MMDDYY, 2=YYMMDD)
&14 {20}	BYTE(0, 1)	*NA*	Set Time Display Format <Data1> = (0=12 HOUR, 1=24 HOUR)
&18 {24}	BYTE(0, 1)	*NA*	Set Waveform View Scale <Data1> = (0=LINEAR, 1=LOG)
&19 {25}	BYTE(0, 1)	*NA*	Set Waveform View Type <Data1> = (0=RECTIFIED, 1=BIPOLAR)
&1A {26}	BYTE(0, 1)	*NA*	Set Waveform View Fill <Data1> = (0=OFF, 1=ON)
&1B {27}	BYTE(0, 1)	*NA*	Set Item Sort Mode <Data1> = (0=ALPHABETIC, 1=MEMORY)
	Time and Date		
&20 {32}	BYTE	*NA*	Set Year
&21 {33}	BYTE	*NA*	Set Month

Table 10: Control Items for Section &06{6} — System Parameter Set

<Item>	<Data1>	<Data2>…	Description of Item
&22 {34}	BYTE	*NA*	Set Day
&23 {35}	BYTE	*NA*	Set Day of Week (0=SUN)
&24 {36}	BYTE	*NA*	Set Hours
&25 {37}	BYTE	*NA*	Set Minutes
&26 {38}	BYTE	*NA*	Set Seconds
	Digital Options		
&30 {48}	BYTE(0, 1)	*NA*	Set System Clock <Data1> = (0=44·1 kHz, 1=48 kHz, 2=96 kHz)
&31 {49}	BYTE(0–3)	*NA*	Set Digital Out Sync <Data1> = (0=INTERNAL, 1=DIGITAL IN, 2=ADAT IN, 3=WORDCLOCK)
&32 {50}	BYTE(0, 1)	*NA*	Set Digital Format <Data1> = (0=PRO, 1=CONSUMER)
&33 {51}	BYTE(0, 1)	*NA*	Set ADAT Main Out <Data1> = (0=L/R, 1=1/2)
	Play Functions		
&40 {64}	BYTE(0–3)	DWORD	Set Play Mode <Data1> = (0=Multi, 1=Program; 2=Sample; 3=Muted) <Data2> = handle of item to become active *Play Item*
&41 {65}	BYTE(0, 1)	*NA*	Set Program Monitor Mode <Data1> = (0=Multi, 1=Program(OMNI))
&42 {66}	BYTE(0–2)	*NA*	Set Sample Monitor Mode <Data1> = (0=Multi, 1=Program; 2=Sample(OMNI))
&48 {72}	BYTE	*NA*	Set Play Key Note
&49 {73}	BYTE	*NA*	Set Play Key Velocity
&4A {74}	BYTE(0–31)	*NA*	Set Play Key Midi Channel <Data1> = (1A=0, 2A=1, …, 16B=31)
&4B {75}	BYTE(0, 1)	*NA*	Set Play Key Echo <Data1> = (0=OFF, 1=ON)
&4C {76}	BYTE(0, 1)	*NA*	Set Program Change Enable <Data1> = (0=OFF, 1=ON)
&4D {77}	BYTE(0, 1)	*NA*	Set Autoload Enable <Data1> = (0=OFF, 1=ON)
	MPC Pad Setup[b]		
&50 {80}	BYTE(0, 1)	*NA*	Set Global Pad Mode <Data1> = (0=DEFAULT, 1=CHROMATIC)
&51 {81}	BYTE(0–15)	*NA*	Set MIDI Channel
&52 {82}	BYTE(0–15)	BYTE(0–100)	Set Pad Sensitivity <Data1> = Pad <Data2> = Sensitivity (0–100 = 100%–200%)
&53 {83}	BYTE(0–95)	BYTE	Set Default Note Assignment <Data1> = Pad, <Data2> = Note
&54 {84}	BYTE	*NA*	Set Chromatic Start Note

[a] Unless this is changed by the user, the default value will be returned: either “Z4 Sampler”, “Z8 Sampler” or “MPC4000”. This is an alternative way of remotely identifying samplers.

[b] This is currently only supported by the MPC4000.

Table 11: Control Items for Section &07{7} — System Parameter Get

<Item>	<Data1>…	<Reply>…	Description of Item
	Global Options		
&01{1}	*NA*	STRING	Get Sampler Name
&02{2}	*NA*	BYTE(0−7)	Get SCSI self ID
&03{3}	*NA*	SWORD (0−±3600)	Get Master Tune
&04{4}	*NA*	BYTE(0−7)	Get Master Level <Reply> = (–42dB – 0dB in 6dB steps) (0=–42dB, 1=–36dB, …, 7=0dB)
&05{5}	BYTE(0, 1)	BYTE(0, 2)	Get MIDI OUT/THRU <Data1> = MIDI port (0=A, 1=B), <Reply> = (0=OUT, 1=THRUA, 2=THRUB)
&06{6}	*NA*	BYTE(0, 1)	Get Qlink Local Control <Reply> = (0=OFF, 1=ON)
&07{7}	*NA*	BYTE(0, 1)	Get Create Default Items at Startup <Reply> = (0=OFF, 1=ON)
&08{8}	*NA*	BYTE(0, 1)	Get MIDI file save format
&09{9}	*NA*	BYTE(0−7)	Set CD-R write speed <Reply> = (0=×1, 1=×2, 2=×4, 3=×6, 4=×8, 5=×12, 6=×16, 7=MAX)
&0A{10}	*NA*	BYTE(0−2)	Set CD-R write mode <Reply> = (0=TEST+WRITE, 1=TEST ONLY, 2=WRITE ONLY)
	Display Options		
&10{16}	*NA*	BYTE(0, 1)	Get Front panel lock-out state <Reply> = (0=NORMAL; 1=LOCKED)
&11{17}	*NA*	BYTE(0−19)	Get Display Contrast
&12{18}	*NA*	BYTE(0, 1)	Get Note Display <Reply> = (0=NUMBER, 1=NAME)
&13{19}	*NA*	BYTE(0−3)	Get Date Display Format <Reply> = (0=DDMMYY, 1=MMDDYY, 2=YYMMDD)
&14{20}	*NA*	BYTE(0, 1)	Get Time Display Format <Reply> = (0=12HOUR, 1=24HOUR)
&18{24}	*NA*	BYTE(0, 1)	Get Waveform View Scale <Reply> = (0=LINEAR, 1=LOG)
&19{25}	*NA*	BYTE(0, 1)	Get Waveform View Type <Reply> = (0=RECTIFIED, 1=BIPOLAR)
&1A{26}	*NA*	BYTE(0, 1)	Get Waveform View Fill <Reply> = (0=OFF, 1=ON)
&1B{27}	*NA*	BYTE(0, 1)	Get Item Sort Mode <Reply> = (0=ALPHABETIC, 1=MEMORY)
	Time and Date		
&20{32}	*NA*	BYTE	Get Year
&21{33}	*NA*	BYTE	Get Month
&22{34}	*NA*	BYTE	Get Day
&23{35}	*NA*	BYTE	Get Day of Week (0=SUN)
&24{36}	*NA*	BYTE	Get Hours
&25{37}	*NA*	BYTE	Get Minutes
&26{38}	*NA*	BYTE	Get Seconds
	Digital Options		
&30{48}	*NA*	BYTE(0, 1)	Get System Clock <Reply> = (0=44·1kHz, 1=48kHz, 2=96kHz)

Table 11: Control Items for Section &07{7} — System Parameter Get

<Item>	<Data1>…	<Reply>…	Description of Item
&31 {49}	*NA*	BYTE(0–3)	Get Digital Out Sync <Reply> = (0=INTERNAL, 1=DIGITAL IN, 2=ADAT IN, 3=WORDCLOCK)
&32 {50}	*NA*	BYTE(0, 1)	Get Digital Format <Reply> = (0=PRO, 1=CONSUMER)
&33 {51}	*NA*	BYTE(0, 1)	Get ADAT Main Out <Reply> = (0=L/R, 1=1/2)
	Play Functions		
&40 {64}	*NA*	BYTE(0–3) DWORD	Get Play Mode <Reply1> = (0=Multi, 1=Program; 2=Sample; 3=Muted) <Reply2> = handle of item which is the active *Play Item*
&41 {65}	*NA*	BYTE(0, 1)	Get Program Monitor Mode <Reply1> = (0=Multi, 1=Program(OMNI))
&42 {66}	*NA*	BYTE(0–2)	Get Sample Monitor Mode <Reply1> = (0=Multi, 1=Program; 2=Sample(OMNI))
&48 {72}	*NA*	BYTE	Get Play Key Note
&49 {73}	*NA*	BYTE	Get Play Key Velocity
&4A {74}	*NA*	BYTE(0–31)	Get Play Key Midi Channel <Reply> = (1A=0, 2A=1, …, 16B=31)
&4B {75}	*NA*	BYTE(0, 1)	Get Play Key Echo <Reply> = (0=OFF, 1=ON)
&4C {76}	*NA*	BYTE(0, 1)	Get Program Change Enable <Reply> = (0=OFF, 1=ON)
&4D {77}	*NA*	BYTE(0, 1)	Get Autoload Enable <Reply> = (0=OFF, 1=ON)
	MPC Pad Setup[a]		
&50 {80}	*NA*	BYTE(0, 1)	Get Global Pad Mode <Reply> = (0=DEFAULT, 1=CHROMATIC)
&51 {81}	*NA*	BYTE(0–15)	Get MIDI Channel
&52 {82}	BYTE(0–15)	BYTE(0–100)	Get Pad Sensitivity <Data1> = Pad <Reply> = Sensitivity (0–100 = 100%–200%)
&53 {83}	BYTE(0–95)	BYTE	Get Default Note Assignment <Data1> = Pad, <Reply> = Note
&54 {84}	*NA*	BYTE	Get Chromatic Start Note

[a] This is currently only supported by the MPC4000.

Item List for the Keygroup Zones section [&0C{12}]

For convenience, Keygroup Zones have their own dedicated section, rather than being manipulated as part of the Keygroup section. Unlike in many of the other sections, selection of a zone is not required before it can be manipulated. Instead each of the zone manipulation functions includes the zone number as the first parameter. The value of zone = 0 means ALL zones, in which case all 4 zones will be updated simultaneously.

Note that zones always refer to the *currently selected* keygroup in the *currently selected* program. So if another keygroup, or program, is to be adjusted the functions in the appropriate sections must be used to select the desired keygroup or program.

Table 12: Control Items for Section &0E {14} — Keygroup Zone Set Parameter

<Item>	<Data1> (Zone number)	<Data2>…	Description of Item
	Setting Zone Parameters		
&01 {1}	BYTE(0, 1–4)	STRING	Set Zone Sample <Data2…0> = name of sample to assign to zone.
&02 {2}$_{\text{RT}}$	BYTE(0, 1–4)	SWORD(−600–+60)	Set Zone Level <Data1> = level in 10×dB
&03 {3}$_{\text{RT}}$	BYTE(0, 1–4)	BYTE(0−100)	Set Zone Pan/Balance <Data2> = Pan/Bal, where (0–100 = L50–R50); centre=&32{50}
&04 {4}	BYTE(0, 1–4)	BYTE(0−15)	Set Zone Output <Data2> = output, where 0=MULTI, 1 = L/R; 2–5 = op1/2–op7/8; 6–15 = L, R, op1-op8
&05 {5}$_{\text{RT}}$	BYTE(0, 1–4)	SBYTE(0−±100)	Set Zone Filter
&06 {6}$_{\text{RT}}$	BYTE(0, 1–4)	SWORD(0−±3600)	Set Zone Cents Tune
&07 {7}	BYTE(0, 1–4)	BYTE(0, 1)	Set Zone Keyboard Track <Data2> = (0=OFF, 1=ON)
&08 {8}	BYTE(0, 1–4)	BYTE(0−6)	Set Zone Playback <Data2> = mode, where 0=NO LOOPING, 1=ONE SHOT 2=LOOP IN REL, 3=LOOP UNTIL REL, 4=LIR→RETRIG, 5=PLAY→RETRIG, 6=AS SAMPLE
&09 {9}	BYTE(0, 1–4)	SWORD (0−±9999)	Set Zone Mod→Start
&0A {10}	BYTE(0, 1–4)	BYTE	Set Zone Low Velocity
&0B {11}	BYTE(0, 1–4)	BYTE	Set Zone High Velocity
&0C {12}$_{\text{RT}}$	BYTE(0, 1–4)	BYTE(0, 1)	Set Zone Mute <Data2> = (0=OFF, 1=ON)
&0D {13}$_{\text{RT}}$	BYTE(0, 1–4)	BYTE(0, 1)	Set Zone Solo <Data2> = (0=OFF, 1=ON)

Table 13: Control Items for Section &0F {15} — Keygroup Zone Get Parameter

<Item>	<Data1> (Zone number)	<Reply>…	Description of Item
	Getting Zone Parameters		
&01 {1}	BYTE(0, 1–4)	STRING	Get Zone Sample
&02 {2}	BYTE(0, 1–4)	SWORD(−600–+60)	Get Zone Level <Reply> = level in 10×dB
&03 {3}	BYTE(0, 1–4)	BYTE(0−100)	Get Zone Pan/Balance
&04 {4}	BYTE(0, 1–4)	BYTE(0−15)	Get Zone Output
&05 {5}	BYTE(0, 1–4)	SBYTE(0−±100)	Get Zone Filter
&06 {6}	BYTE(0, 1–4)	SBYTE(0−±3600)	Get Zone Cents Tune

Table 13: Control Items for Section &0F {15} — Keygroup Zone Get Parameter

<Item>	<Data1> (Zone number)	<Reply>…	Description of Item
&07 {7}	BYTE(0, 1–4)	BYTE(0, 1)	Get Zone Keyboard Track
&08 {8}	BYTE(0, 1–4)	BYTE(0−6)	Get Zone Playback
&09 {9}	BYTE(0, 1–4)	SWORD (0−±9999)	Get Zone Mod→Start
&0A {10}	BYTE(0, 1–4)	BYTE	Get Zone Low Velocity
&0B {11}	BYTE(0, 1–4)	BYTE	Get Zone High Velocity
&0C {12}	BYTE(0, 1–4)	BYTE(0, 1)	Get Zone Mute <Reply> = (0=OFF, 1=ON)
&0D {13}	BYTE(0, 1–4)	BYTE(0, 1)	Get Zone Solo <Reply> = (0=OFF, 1=ON)

When information about a Keygroup Zone is requested with a *Get* message, the data is returned in a “REPLY” confirmation message (see *Confirmation Messages* on page 5). The format of the data content of these messages is summarised in Table 13.

If the selected Zone is set to be 0 (zero), then the REPLY message will contain additional data (one set for every Zone in the current Keygroup). The format of this extra data is the same as that shown in Table 13 where the set of data bytes is repeated four times (once for each Zone) in a single REPLY confirmation message.

Item List for the Keygroup section [&10{16}]

Although Keygroups are really part of a program, it is more convenient to edit them as separate entities. Before a Keygroup can be used, it must be selected to be *current*. This is done using the *Select Keygroup* function where the number of the keygroup is specified. Note that this keygroup always refers to the appropriate keygroup in the currently selected program, so care must be taken to ensure that the current keygroup remains valid when a different program is selected. All of the keygroups in a program can be edited simultaneously by selecting keygroup edit mode to be ALL or ADD. Note that, for convenience, Keygroup Zones are manipulated using functions in the Keygroup Zone section.

When information about a Keygroup is requested with a *Get* message, the data is returned in a “REPLY” confirmation message (see *Confirmation Messages* on page 5). The format of the data content of these messages is summarised in Table 17 and Table 15.

Table 14: Control Items for Section &10 {16} — Keygroup Main

<Item>	<Data1>	<Data2>…	Description of Item
&01 {1}	BYTE	*NA*	Select Keygroup to be *current* <Data1> = Keygroup number
&02 {2}	*NA*	*NA*	Get Current Keygroup

Table 15: Format of “REPLY” confirmation messages for Section &10{16} — Keygroup

<Item> requested	<Data1>	<Data2>	Description of Data Returned
&02 {2}	BYTE	*NA*	Current Keygroup <Data1> = Keygroup number

Table 16: Control Items for Section &12 {18} — Keygroup Set Parameter

<Item>	<Data1>	<Data2>…	Description of Item
	Setting General Options		
&01 {1}	BYTE	*NA*	Set Group ID
&02 {2}	BYTE(0–2)	*NA*	Set Keygroup Edit Mode <Data2> = (0=SINGLE, 1=ALL, 2=ADD)
&04 {4}	BYTE	*NA*	Set Low Note
&05 {5}	BYTE	*NA*	Set High Note
&06 {6}	BYTE(0, 1–64)	*NA*	Set Mute Group <Data1> = (0=OFF, 1–64=value)
&07 {7}	BYTE(0, 1–4)	*NA*	Set FX override <Data1> = (0=OFF, 1=A, 2=B, 3=C, 4=D, 5=AB, 6=CD, 7=MULTI)
&08 {8}$_{\mathrm{RT}}$	SWORD (−600–+60)	*NA*	Set FX Send Level <Data1> = level in 10×dB
&09 {9}	BYTE(0–2)	*NA*	Set Zone Crossfade <Data1> = (0=OFF, 1=VELOCITY, 2=REAL-TIME)
&0A {10}	BYTE(0–2)	*NA*	Crossfade type <Data1> = (0=LIN, 1=EXP, 2=LOG)
&0E {14}	BYTE(1–64)	*NA*	Set Polyphony
&0F {15}	BYTE	*NA*	Set Zone Crossfade Source Controller Number (only used when Zone Crossfade Source is MIDI CTRL)
	Set Keygroup Pitch/Amp		

Table 16: Control Items for Section &12 {18} — Keygroup Set Parameter

<Item>	<Data1>	<Data2>…	Description of Item
&10 {16}$_{RT}$	SBYTE(0 – ±3600)	*NA*	Set Cents Tune
&11 {17}$_{RT}$	SWORD	*NA*	Set Keygroup Level <Data1> = value in 10×dB
	Set Keygroup Trigger		
&18 {24}	BYTE(0, 1)	*NA*	Set Play Trigger <Data1> = (0=NOTE ON, 1=NOTE OFF)
&19 {25}	WORD(0–129)	*NA*	Set Play Trigger Velocity <Data1> = (0=ON VEL, 1=OFF VEL, 2–129=0–127)
&1A {26}	BYTE(0, 1)	*NA*	Set Play Toggle Note <Data1> = (0=OFF, 1=ON)
	Set Filter <Data1> = Filter block (0=NORMAL, 1=TRIPLE(1), 2=TRIPLE(2), 3=TRIPLE(3))		
&20 {32}	BYTE(0–3)	BYTE(0–35)	Set Filter Mode NORMAL mode, when <Data1> = 0; <Data2> = (0=OFF, 1=2-POLE LP, 2=2-POLE LP+, 3=4-POLE LP, 4=4-POLE LP+, 5=6-POLE LP, 6=2-POLE BP, 7=2-POLE BP+, 8=4-POLE BP, 9=4-POLE BP+, 10=6-POLE BP, 11=1-POLE HP, 12=1-POLE HP+, 13=2-POLE HP, 14=2-POLE HP+, 15=4-POLE HP, 16=4-POLE HP+, 17=6-POLE HP, 18=LO<>HI, 19=LO<>BAND, 20=BAND<>HI, 21=NOTCH 1, 22=NOTCH 2, 23=NOTCH 3, 24=WIDE NOTCH, 25=BI-NOTCH, 26=PEAK 1, 27=PEAK 2, 28=PEAK 3, 29=WIDE PEAK, 30=BI-PEAK, 31=PHASER 1, 32=PHASER 2, 33=BI-PHASE, 34=VOWELISER, 35=TRIPLE) TRIPLE mode, when <Data1> = 1–3; <Data2> = (0=OFF, 1=2-POLE LP, 2=1-POLE BP 3=1-POLE HP, 4=2-POLE HP, 5=NOTCH 1, 6=EQ, 7=EQ+)
&21 {33}$_{RT}$	BYTE(0–3)	BYTE(0–100)	Set Filter Cutoff Frequency
&22 {34}$_{RT}$	BYTE(0–3)	BYTE(0–64)	Set Filter Resonance
&23 {35}$_{RT}$	BYTE(0–5)	*NA*	Set Filter Attenuation (one setting for all filters) <Data1> = (0, 1, 2, 3, 4, 5 = 0dB, 6dB, 12dB, 18dB, 24dB, 30dB)
	Set Envelope <Data1> = Envelope (0=AMP, 1=FILTER, 2=AUX)		
&30 {48}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Set Envelope Rate 1 *(for AMP = Attack)*
&31 {49}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Set Envelope Level 1 *(FILTER and AUX only)*
&32 {50}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Set Envelope Rate 2 *(for AMP = Decay)*
&33 {51}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Set Envelope Level 2 *(for AMP = Sustain)*
&34 {52}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Set Envelope Rate 3 *(for AMP = Release)*
&35 {53}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Set Envelope Level 3 *(FILTER and AUX only)*
&36 {54}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Set Envelope Rate 4 *(FILTER and AUX only)*
&37 {55}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Set Envelope Level 4 *(FILTER and AUX only)*
&42 {66}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Set Envelope Reference *(FILTER and AUX only)*
&43 {67}$_{RT}$	BYTE(0–2)	BYTE(0, 1)	Set Attack Hold <Data1> = (0=OFF, 1=ON) *(AMP only)*
	LFOs — Set Values. <Data1> = LFO (0=LFO1, 1=LFO2)		
&50 {80}$_{RT}$[a]	BYTE(0, 1)	BYTE(0–100)	Set LFO Rate <Data2> = rate
&51 {81}$_{RT}$[a]	BYTE(0, 1)	BYTE(0–100)	Set LFO Delay <Data2> = delay
&52 {82}$_{RT}$	BYTE(0, 1)	BYTE(0–100)	Set LFO Depth <Data2> = depth
&53 {83}$_{RT}$	BYTE(0, 1)	BYTE(0–8)	Set LFO Waveform <Data2> = waveform, (see Table 23)
&54 {84}$_{RT}$	BYTE(0, 1)	WORD(0–360)	Set LFO Phase

Table 16: Control Items for Section &12 {18} — Keygroup Set Parameter

<Item>	<Data1>	<Data2>…	Description of Item
&55 {85}$_{\mathrm{RT}}$	BYTE(0, 1)	SBYTE(0−±50)	Set LFO Shift
&56 {86}$_{\mathrm{RT}}$	BYTE(0, 1)	BYTE(0, 1)	Set LFO MIDI Sync <Data2> = (0=OFF, 1=ON)
&57 {87}$_{\mathrm{RT}}$	BYTE(0, 1)	BYTE(0–68)	Set MIDI Clock Sync Division <Data2> = value, where 0=8 cy/bt, 1=6 cy/bt, 2=4 cy/bt, 3=3 cy/bt, 4=2 cy/bt, 5=1 cy/bt 6=2 bt/cy, 7=3 bt/cy, …, 68=64 bt/cy
&58 {88}$_{\mathrm{RT}}$	BYTE(0, 1)	BYTE(0, 1)	Set LFO Re-trigger <Data2> = (0=OFF, 1=ON)
&59 {89}$_{\mathrm{RT}}$	BYTE(0, 1)	BYTE(0, 1)	Set LFO sync (*i.e.,* all voices in program locked to same lfo)

a. The LFO Rate and Delay settings are only valid if the LFO is not synchronised to MIDI clock.

Table 17: Control Items for Section &13 {19} — Keygroup Get Parameter

<Item>	<Data1>	<Reply>…	Description of Item
	Getting General Options		
&01 {1}	*NA*	BYTE	Get Group ID
&02 {2}	*NA*	BYTE(0−2)	Get Keygroup Edit Mode <Reply1> = (0=SINGLE, 1=ALL, 2=ADD)
&04 {4}	*NA*	BYTE	Get Low Note
&05 {5}	*NA*	BYTE	Get High Note
&06 {6}	*NA*	BYTE(0, 1–64)	Get Mute Group <Reply1> = (0=OFF, 1–64=value)
&07 {7}	*NA*	BYTE(0, 1–4)	Get FX override <Reply1> = (0=OFF, 1=A, 2=B, 3=C, 4=D, 5=AB, 6=CD, 7=MULTI)
&08 {8}	*NA*	SWORD (−600–+60)	Get FX Send Level <Reply1> = level in 10×dB
&09 {9}	*NA*	BYTE(0−2)	Get Zone Crossfade <Reply1> = (0=OFF, 1=VELOCITY, 2=REAL-TIME)
&0A {10}	*NA*	BYTE(0−2)	Get Crossfade type <Reply1> = (0=LIN, 1=EXP, 2=LOG)
&0E {14}	*NA*	BYTE(1−64)	Get Polyphony
&0F {15}	*NA*	BYTE	Get Zone Crossfade Source Controller Number (only used when Zone Crossfade Source is MIDI CTRL)
	Get Keygroup Pitch/Amp		
&10 {16}$_{\mathrm{RT}}$	*NA*	SBYTE(0−±3600)	Get Cents Tune.
&11 {17}$_{\mathrm{RT}}$	*NA*	SWORD	Get Keygroup Level <Reply> = value in 10×dB
	Set Keygroup Trigger		
&18 {24}	*NA*	BYTE(0, 1)	Get Play Trigger <Reply> = (0=NOTE ON, 1=NOTE OFF)
&19 {25}	*NA*	WORD(0−129)	Get Play Trigger Velocity <Reply> = (0=ON VEL, 1=OFF VEL, 2–129=0–127)
&1A {26}	*NA*	BYTE(0, 1)	Get Play Toggle Note <Reply> = (0=OFF, 1=ON)
	Get Filter <Data1> = Filter block (0=NORMAL, 1=TRIPLE(1), 2=TRIPLE(2), 3=TRIPLE(3))		

Table 17: Control Items for Section &13 {19} — Keygroup Get Parameter

<Item>	<Data1>	<Reply>…	Description of Item
&20 {32}	BYTE(0–3)	BYTE(0–35)	Get Filter Mode NORMAL mode, when <Data1> = 0; <Reply> = (0=OFF, 1=2-POLE LP, 2=2-POLE LP+, 3=4-POLE LP, 4=4-POLE LP+, 5=6-POLE LP, 6=2-POLE BP, 7=2-POLE BP+, 8=4-POLE BP, 9=4-POLE BP+, 10=6-POLE BP, 11=1-POLE HP, 12=1-POLE HP+, 13=2-POLE HP, 14=2-POLE HP+, 15=4-POLE HP, 16=4-POLE HP+, 17=6-POLE HP, 18=LO<>HI, 19=LO<>BAND, 20=BAND<>HI, 21=NOTCH 1, 22=NOTCH 2, 23=NOTCH 3, 24=WIDE NOTCH, 25=BI-NOTCH, 26=PEAK 1, 27=PEAK 2, 28=PEAK 3, 29=WIDE PEAK, 30=BI-PEAK, 31=PHASER 1, 32=PHASER 2, 33=BI-PHASE, 34=VOWELISER, 35=TRIPLE) TRIPLE mode, when <Data1> = 1–3; <Reply> = (0=OFF, 1=2-POLE LP, 2=1-POLE BP 3=1-POLE HP, 4=2-POLE HP, 5=NOTCH 1, 6=EQ, 7=EQ+)
&21 {33}$_{RT}$	BYTE(0–3)	BYTE(0–100)	Get Filter Cutoff Frequency
&22 {34}$_{RT}$	BYTE(0–3)	BYTE(0–64)	Get Filter Resonance
&23 {35}$_{RT}$	*NA*	BYTE(0–5)	Get Filter Attenuation (one setting for all filters) <Reply> = (0, 1, 2, 3, 4, 5 = 0dB, 6dB, 12dB, 18dB, 24dB, 30dB)
	Get Envelope <Data1> = Envelope (0=AMP, 1=FILTER, 2=AUX)		
&30 {48}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Get Envelope Rate 1 *(for AMP = Attack)*
&31 {49}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Get Envelope Level 1 *(FILTER and AUX only)*
&32 {50}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Get Envelope Rate 2 *(for AMP = Decay)*
&33 {51}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Get Envelope Level 2 *(for AMP = Sustain)*
&34 {52}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Get Envelope Rate 3 *(for AMP = Release)*
&35 {53}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Get Envelope Level 3 *(FILTER and AUX only)*
&36 {54}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Get Envelope Rate 4 *(FILTER and AUX only)*
&37 {55}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Get Envelope Level 4 *(FILTER and AUX only)*
&42 {66}$_{RT}$	BYTE(0–2)	BYTE(0–100)	Get Envelope Reference *(FILTER and AUX only)*
&43 {67}$_{RT}$	BYTE(0–2)	BYTE(0, 1)	Get Attack Hold <Reply> = (0=OFF, 1=ON) *(AMP only)*
	LFOs — Get Values. <Data1> = LFO (0=LFO1, 1=LFO2)		
&50 {80}$_{RT}$[a]	BYTE(0, 1)	BYTE(0–100)	Get LFO Rate
&51 {81}$_{RT}$[a]	BYTE(0, 1)	BYTE(0–100)	Get LFO Delay
&52 {82}$_{RT}$	BYTE(0, 1)	BYTE(0–100)	Get LFO Depth
&53 {83}$_{RT}$	BYTE(0, 1)	BYTE(0–8)	Get LFO Waveform <Reply> = waveform, (see Table 23)
&54 {84}$_{RT}$	BYTE(0, 1)	WORD(0–360)	Get LFO Phase
&55 {85}$_{RT}$	BYTE(0, 1)	SBYTE(0–±50)	Get LFO Shift
&56 {86}$_{RT}$	BYTE(0, 1)	BYTE(0, 1)	Get LFO MIDI Sync <Reply> = (0=OFF, 1=ON)
&57 {87}$_{RT}$	BYTE(0, 1)	BYTE(0–68)	Get MIDI Clock Sync Division <Reply> = value, where 0=8cy/bt, 1=6cy/bt, 2=4cy/bt, 3=3cy/bt, 4=2cy/bt, 5=1cy/bt 6=2bt/cy, 7=3bt/cy, …, 68=64bt/cy
&58 {88}$_{RT}$	BYTE(0, 1)	BYTE(0, 1)	Get LFO Re-trigger <Reply> = (0=OFF, 1=ON)
&59 {89}$_{RT}$	BYTE(0, 1)	BYTE(0, 1)	Get LFO sync (*i.e.,* all voices in program locked to same LFO)

[a] The LFO Rate and Delay settings are only valid if the LFO is not synchronised to MIDI clock.

Item List for Program section [&14{20}]

When changing the parameters of a Program, it is not necessary to send the name with every parameter change SysEx message. Instead, the *currently selected* Program is always acted on with these messages. There are three ways to select a Program: (i) Create a new Program; and (ii) select an existing Program in memory to be the *current* one; and (iii) change the current Program via the sampler's front panel. The first method can only be used if a Program of the specified name does not already exist in memory. The last method will only work if the *synchronisation* option is on (this is the default option; see SysEx section on how to change this) and the current mode is "PROGRAM" mode.

Handles used to access Programs are special 28-bit values which uniquely identify Programs in memory.

All of the parameters of a Program can be edited using the functions in this section, with the exception of Keygroups which, for convenience, are edited using their own functions.

When information about a Program is requested with a *Get* message, the data is returned in a "REPLY" confirmation message (see *Confirmation Messages* on page 5). The format of the data content of these messages is summarised in Table 19. The parameters for the modulation sources used in the Programs are detailed in Table 24.

Table 18: Control Items for Section &14 {20} — Program Main

<Item>	<Data1>	<Data2>…	Description of Item
	Generic List Functions		
&01 {1}	*NA*	*NA*	Get number of items in memory
&02 {2}	BYTE(0–3)	*NA*	Get list of info for all items: <Data1>: 0=list of handles; 1=list of names; 2=list of handle+name; 3=list of handle+modified/tagged name
&03 {3}	DWORD	*NA*	Select *current* item by handle
&04 {4}	STRING	*NA*	Select *current* item by name
&05 {5}	*NA*	*NA*	Get handle of *current* item
&06 {6}	*NA*	*NA*	Get name of *current* item
&07 {7}	DWORD	*NA*	Get item name from handle
&08 {8}	STRING	*NA*	Get item handle from name
&09 {9}	*NA*	*NA*	Delete ALL items from memory
&0A {10}	*NA*	*NA*	Delete *current* item from memory
&0B {11}	DWORD	*NA*	Delete item represented by handle <Data1>
&0C {12}	STRING	*NA*	Rename *current* item
&0D {13}	DWORD	STRING	Rename item represented by handle <Data1>
&0E {14}	BYTE(0−7)	BYTE(0, 1) BYTE(0, 1)	Set Tag Bit <Data1> = bit to set, <Data2> = (0=OFF, 1=ON), <Data3> = (0=CURRENT, 1=ALL)
&0F {15}	*NA*	*NA*	Get Tag Bitmap <Reply> = WORD
&10 {16}	*NA*	*NA*	Get name of *current* item with modified/tagged info.
&11 {17}	*NA*	*NA*	Get modified state of *current* item.
&18 {24}	BYTE(0−7)	*NA*	Delete tagged items <Data1> = tag bit
	General Program Functions		
&40 {64}	WORD	STRING	Create New Program <Data1> = number of keygroups; <Data2> = name
&41 {65}	BYTE	*NA*	Add Keygroups to Program <Data1> = number of keygroups to add

Table 18: Control Items for Section &14 {20} — Program Main

<Item>	<Data1>	<Data2>…	Description of Item
&42 {66}	BYTE	*NA*	Delete Keygroup from Program: <Data1> = index of keygroup to delete (zero-based)
&43 {67}	*NA*	*NA*	Delete Blank Keygroups
&44 {68}	BYTE(0–2)	*NA*	Arrange Keygroups <Data1> = (0=ORIGINAL NOTE, 1=LOW NOTE, 2=HIGH NOTE)
&45 {69}	BYTE	*NA*	Copy Keygroup <Data1> = index of keygroup to copy
&48 {72}	STRING	*NA*	Copy Program <Data1> = name of new program
&49 {73}	DWORD	DWORD	Merge Program <Data1> = handle of Program1, <Data2> = handle of Program2
&4A {74}	BYTE	BYTE, BYTE(1–4), BYTE(0, 1), STRING	Add Keygroup Sample <Data1> = low note, <Data2> = high note, <Data3> = zone, <Data4> = keytrack (0=OFF, 1=ON), <Data5> = sample name
&50 {80}	*NA*	*NA*	Copy Program Temperament to User-Temperament Template
&54 {84}	*NA*	*NA*	Get Number of Modulation Connections
&55 {85}	*NA*	*NA*	Get Number of Modulation Sources
&56 {86}	*NA*	*NA*	Get Number of Modulation Destinations
&57 {87}	WORD	*NA*	Get Name of Modulation Source <Data1> = index of source
&58 {88}	WORD	*NA*	Get Name of Modulation Destination <Data1> = index of destination

Table 19: Format of "REPLY" confirmation messages for Section &14 {20} — Program Main

<Item> requested	<Reply>…	Description of Data Returned
&01 {1}	DWORD	Get number of items in memory
&02 {2}	DWORD[n] *or* STRING[n] *or* (DWORD, STRING)[n]	Get list of info for all items: <Data1>: 0=list of handles; 1=list of names; 2=list of handle+name; 3=list of handle+modified/tagged name An array of handles, names, or a combination of both (handle, name) will be returned in a single data stream. There will be *n* sets of data, where *n* = number of items in memory.
&05 {5}	DWORD	Get handle of *current* item
&06 {6}	STRING	Get name of *current* item
&07 {7}	STRING	Get item name from handle
&08 {8}	DWORD	Get item handle from name
&0F {15}	WORD	Get Tag Bitmap
&10 {16}	STRING	Get name of *current* item with modified/tagged info. The name is modified to indicate the current *modified* and *tagged* state of the item.
&11 {17}	BYTE	Get modified state of *current* item <Reply> = (0=NOT MODIFIED, 1=MODIFIED)
&40 {64}	DWORD	Create Item <Reply> = handle of new item
&4A {74}	BYTE, BYTE, BYTE(1–4), BYTE(0, 1), STRING	Add Keygroup Sample <Reply1> = low note, <Reply2> = high note, <Reply3> = zone, <Reply4> = keytrack (0=OFF, 1=ON), <Reply5> = sample name
&54 {84}	BYTE	Get Number of Modulation Connections

Table 19: Format of “REPLY” confirmation messages for Section &14 {20} — Program Main

<Item> requested	<Reply>…	Description of Data Returned
&55 {85}	WORD	Get Number of Modulation Sources
&56 {86}	WORD	Get Number of Modulation Destinations
&57 {87}	STRING	Get Name of Modulation Source
&58 {88}	STRING	Get Name of Modulation Destination

Table 20: Control Items for Section &16 {22} — Program Set Parameter

<Item>	<Data1>	<Data2>…	Description of Item
	Set General Options		
&01 {1}	BYTE	*NA*	Set Group ID
&03 {3}	BYTE(0, 1)	*NA*	Set Program Type <Data1> = (0=KEYGROUP, 1=DRUM)
&04 {4}	STRING	*NA*	Set Genre
&08 {8}	WORD(0−128)	*NA*	Set “Program Number” <Data1> = (0=OFF, 1–128)
&09 {9}	WORD	*NA*	Set Number of keygroups
&0A {10}	BYTE(0, 1)	*NA*	Set Keygroup Crossfade <Data1> = (0=OFF, 1=ON)
&0B {11}	BYTE(0−2)	*NA*	Set Keygroup Crossfade type <Data1> = (0=LIN, 1=EXP, 2=LOG)
&0C {12}	SWORD (−600−+60)	*NA*	Set Program Level <Data1> = level in 10×dB
	Midi Options — Set		
&10 {16}	BYTE(1−64)	*NA*	Set Polyphony
&11 {17}	BYTE(0, 1)	*NA*	Set Reassignment <Data1> = (0=QUIETEST, 1=OLDEST)
&12 {18}$_{\mathrm{RT}}$	BYTE(0–100)	*NA*	Soft Pedal Loudness Reduction
&13 {19}$_{\mathrm{RT}}$	BYTE(0–100)	*NA*	Soft Pedal Attack Stretch
&14 {20}$_{\mathrm{RT}}$	BYTE(0–100)	*NA*	Soft Pedal Filter Close
&15 {21}$_{\mathrm{RT}}$	SBYTE(−36−+36)	*NA*	Midi Transpose
&18 {24}	BYTE(0–95)	BYTE	MPC/MPD pad assignment <Data1> = pad, <Data2> = note
	Set Modulation Matrix (Data1 = pin number)		
&20 {32}	BYTE	WORD, WORD, WORD, SBYTE(0−±100)	Set Modulation Connection <Data1> = connection (pin) number; <Data2> = keygroup number (0=ALL, 1–128=KEYGROUP) <Data3> = source (see Table 24); <Data4> = destination (see Table 25); <Data5> = level. If Source or Destination is zero, the connection will be cleared.
&21 {33}	BYTE	WORD	Set Modulation Source (see Table 24)
&22 {34}	BYTE	WORD	Set Modulation Destination (see Table 25)
&23 {35}$_{\mathrm{RT}}$	BYTE	WORD SBYTE(0−±100)	Set Modulation Level <Data1> = pin number; <Data2> = (0=ALL, 1–128=KEYGROUP); <Data3> = level
&24 {36}	BYTE	BYTE	Set MIDI controller number (only used if Source = CTRL)
&25 {37}	BYTE	WORD	Set *Edit Keygroup* (used to edit level) <Data2> = *Edit Keygroup*
&26 {38}	BYTE	SBYTE	Set Modulation Level of *Edit Keygroup*

Table 20: Control Items for Section &16 {22} — Program Set Parameter

<Item>	<Data1>	<Data2>…	Description of Item
	Tune — Set Values		
&30 {48}$_{\text{RT}}$	SWORD(0–±3600)	*NA*	Cents Tune
&31 {49}	BYTE(0–7)	*NA*	Temperament Template, where <Data1> = template (see Table 22).
&32 {50}	SBYTE[12] (0–±50)	*NA*	Set Program Temperament All the values are sent one after the other starting at C (*i.e.,* 24 data bytes are transmitted, representing all 12 notes). These values always represent a Key of C. If the current Key is not C, the temperament will be adjusted accordingly.
&33 {51}	BYTE(0–11)	*NA*	Key = <Data1> where: 0=C, 1=C#, 2=D, 3=E♭, 4=E, 5=F, 6=F#, 7=G, 8=G#, 9=A, 10=B♭, 11=B
&34 {52}	BYTE(0–11)	SBYTE(0–±50)	Set User Temperament Note <Data1> = note, <Data2> = cents
	Pitch Bend — Set Values		
&40 {64}$_{\text{RT}}$	BYTE(0–24)	*NA*	Set Pitch Bend Up <Data1> = semitones
&41 {65}$_{\text{RT}}$	BYTE(0–24)	*NA*	Set Pitch Bend Down <Data1> = semitones
&42 {66}$_{\text{RT}}$	BYTE(0, 1)	*NA*	Set Bend Mode <Data1> = (0=NORMAL, 1=HELD)
&43 {67}$_{\text{RT}}$	SBYTE(0–±12)	*NA*	Set Aftertouch Value
&44 {68}$_{\text{RT}}$	BYTE(0–2)	*NA*	Set Legato Setting <Data1> = (0=OFF, 1=PITCH, 2=LOOP)
&45 {69}$_{\text{RT}}$	BYTE(0, 1)	*NA*	Set Portamento Enable <Data1> = (0=OFF, 1=ON)
&46 {70}$_{\text{RT}}$	BYTE(0, 1)	*NA*	Set Portamento Mode <Data1> = (0=TIME, 1=RATE)
&47 {71}$_{\text{RT}}$	BYTE(0–100)	*NA*	Set Portamento Time
&48 {72}$_{\text{RT}}$	BYTE(0, 1)	*NA*	Set Glissando Mode <Data1> = (0=PORTAMENTO, 1=GLISS)
&49 {73}$_{\text{RT}}$	BYTE(0, 1)	*NA*	Set Aftertouch Type <Data1> = (0=CHANNEL, 1=POLY)

Table 21: Control Items for Section &17 {23} — Program Get Parameter

<Item>	<Data1>…	<Reply>…	Description of Item
	Get General Options		
&01 {1}	*NA*	BYTE	Get Group ID
&03 {3}	*NA*	BYTE(0, 1)	Get Program Type <Data1> = (0=KEYGROUP, 1=DRUM)
&04 {4}	*NA*	STRING	Get Genre
&08 {8}	*NA*	WORD(0–128)	Get "Program Number" <Reply1> = (0=OFF, 1–128)
&09 {9}	*NA*	WORD	Get Number of keygroups
&0A {10}	*NA*	BYTE(0, 1)	Get Keygroup Crossfade <Reply1> = (0=OFF, 1=ON)
&0B {11}	*NA*	BYTE(0–2)	Get Keygroup Crossfade type <Reply1> = (0=LIN, 1=EXP, 2=LOG)
&0C {12}	*NA*	SWORD(–600–60)	Get Program Level <Reply1> = level in 10×dB
	Midi Options — Get		
&10 {16}	*NA*	BYTE(1–64)	Get Polyphony
&11 {17}	*NA*	BYTE(0, 1)	Get Reassignment <Reply1> = (0=QUIETEST, 1=OLDEST)
&12 {18}	*NA*	BYTE(0–100)	Soft Pedal Loudness Reduction
&13 {19}	*NA*	BYTE(0–100)	Soft Pedal Attack Stretch

Table 21: Control Items for Section &17 {23} — Program Get Parameter

<Item>	<Data1>…	<Reply>…	Description of Item
&14 {20}	*NA*	BYTE(0–100)	Soft Pedal Filter Close
&15 {21}	*NA*	SBYTE(−36−+36)	Midi Transpose
&18 {24}	BYTE(0–95)	BYTE	MPC/MPD pad assignment <Data1> = pad, <Reply> = note
	Get Modulation Matrix (Data1 = pin number)		

&20 {32}	BYTE,
	WORD
		WORD,
		WORD,
		SBYTE(0–±100)[n]
			Get Modulation Connection
			<Data1> = connection (pin) number;
			<Data2> = keygroup number for level (0=ALL, 1–128=KEYGROUP)
			<Reply1> = source (see Table 24);
			<Reply2> = destination (see Table 25);
			<Reply3>[n] = level. *(This is either a single value, or multiple values if <Data2> = 0 (ALL); one value per keygroup)*

&21 {33}	BYTE	WORD	Get Modulation Source (see Table 24)
&22 {34}	BYTE	WORD	Get Modulation Destination (see Table 25)
&23 {35}	BYTE WORD	SBYTE(0–±100)	Get Modulation Level <Data2> = keygroup number (0=ALL, 1–128=KEYGROUP) *(This is either a single value, or multiple values if <Data2> = ALL; one value per keygroup)*
&24 {36}	BYTE	BYTE	Get MIDI controller number (only used if Source = CTRL)
&25 {37}	BYTE	WORD	Get *Edit Keygroup* (used to edit level)
&26 {38}	BYTE	SBYTE	Get Modulation Level of *Edit Keygroup*
	Tune — Get Values		
&30 {48}	*NA*	SWORD(0–±3600)	Cents Tune
&31 {49}	*NA*	BYTE(0–7)	Temperament Template, where <Reply1> = template (see Table 22).
&32 {50}	*NA*	SBYTE[12] (0–±50)	Get Program Temperament. All the values are sent one after the other starting at C (*i.e.,* 24 data bytes are transmitted, representing all 12 notes). These values always represent a Key of C. If the current Key is not C, the temperament will be adjusted accordingly.
&33 {51}	*NA*	BYTE(0–11)	Key = <Reply1> where: 0=C, 1=C#, 2=D, 3=E♭, 4=E, 5=F, 6=F#, 7=G, 8=G#, 9=A, 10=B♭, 11=B
&34 {52}	BYTE(0–11)	SBYTE(0–±50)	Get User Temperament Note <Data1> = note, <Reply> = cents
	Pitch Bend — Get Values		
&40 {64}	*NA*	BYTE(0–24)	Get Pitch Bend Up
&41 {65}	*NA*	BYTE(0–24)	Get Pitch Bend Down
&42 {66}	*NA*	BYTE(0, 1)	Get Bend Mode <Reply> = (0=NORMAL, 1=HELD)
&43 {67}	*NA*	SBYTE(0–±12)	Get Aftertouch Value
&44 {68}	*NA*	BYTE(0–2)	Get Legato Setting <Reply> = (0=OFF, 1=PITCH, 2=LOOP)
&45 {69}	*NA*	BYTE(0, 1)	Get Portamento Enable <Reply> = (0=OFF, 1=ON)
&46 {70}	*NA*	BYTE(0, 1)	Get Portamento Mode <Reply> = (0=TIME, 1=RATE)
&47 {71}	*NA*	BYTE(0–100)	Get Portamento Time
&48 {72}	*NA*	BYTE(0, 1)	Get Glissando Mode <Reply> = (0=PORTAMENTO, 1=GLISS)
&49 {73}	*NA*	BYTE(0, 1)	Get Aftertouch Type <Reply> = (0=CHANNEL, 1=POLY)

Table 22: Program Temperament Templates

Template Number	Template
0	USER
1	EVEN-TEMPERED
2	ORCHESTRAL
3	WERKMEISTER
4	1/5 MEANTONE
5	1/4 MEANTONE
6	JUST
7	ARABIAN

Table 23: Program LFO Waveforms

Waveform Number	Waveform
0	TRIANGLE
1	SINE
2	SQUARE
3	SAW UP
4	SAW DOWN
5	RANDOM

Table 24: Modulation Matrix Sources

Index	Description	Index	Description	Index	Description	Index	Description
0	*NONE*	1	MODWHEEL	2	BEND UP	3	BEND DOWN
4	AFTERTOUCH	5	VELOCITY	6	BIPOLAR VELICITY	7	OFF VELICITY
8	KEYBOARD	9	LFO1	10	LFO2	11	AMP ENV
12	FILTER ENV	13	AUX ENV	14	MIDI CTRL		
15	Q-LINK 1	16	Q-LINK 2	17	Q-LINK 3	18	Q-LINK 4
19	Q-LINK 5	20	Q-LINK 6	21	Q-LINK 7	22	Q-LINK 8
23	𝅗𝅥MODWHEEL	24	𝅗𝅥BEND UP	25	𝅗𝅥BEND DOWN	26	𝅗Q-LINK 1
27	𝅗Q-LINK 2	28	𝅗𝅥Q-LINK 3	29	𝅗𝅥Q-LINK 4	30	𝅗Q-LINK 5
31	𝅗Q-LINK 6	32	𝅗𝅥Q-LINK 7	33	𝅗𝅥Q-LINK 8	34	𝅗LFO1
35	𝅗LFO2	36	𝅗𝅥MIDI CTRL				

Table 25: Modulation Matrix Destinations

Index	Description	Index	Description	Index	Description	Index	Description
0	NONE	1	AMPLITUDE	2	PAN	3	PITCH
4	LFO1 RATE	5	LFO1 DEPTH	6	LFO1 DELAY	7	LFO1 PHASE
8	LFO1 OFFSET	9	LFO2 RATE	10	LFO2 DEPTH	11	LFO2 DELAY
12	LFO2 PHASE	13	LFO2 OFFSET	14	FILTER CUTOFF	15	FILTER RESONANCE
16	TR. FILT 1 CUTOFF	17	TR. FILT 1 RESONANCE	18	TR. FILT 2 CUTOFF	19	TR. FILT 2 RESONANCE
20	TR. FILT 3 CUTOFF	21	TR. FILT 3 RESONANCE	22	AMP ENV ATTACK	23	AMP ENV DECAY
24	AMP ENV RELEASE	25	FILTER ENV RATE 1	26	FILTER ENV RATE 2	27	FILTER ENV RATE 4
28	AUX ENV RATE 1	29	AUX ENV RATE 2	30	AUX ENV RATE 4	31	ZONE CROSSFADE
32	ZONE 1 LEVEL	33	ZONE 1 PAN	34	ZONE 1 PITCH	35	ZONE 1 START
36	ZONE 1 FILTER	37	ZONE 2 LEVEL	38	ZONE 2 PAN	39	ZONE 2 PITCH
40	ZONE 2 START	41	ZONE 2 FILTER	42	ZONE 3 LEVEL	43	ZONE 3 PAN
44	ZONE 3 PITCH	45	ZONE 3 START	46	ZONE 3 FILTER	47	ZONE 4 LEVEL
48	ZONE 4 PAN	49	ZONE 4 PITCH	50	ZONE 4 START	51	ZONE 4 FILTER

Item List for Multi section [&18{24}]

The simplest functions of the Multi section allow control of all of the parameters of a Multi in real-time. For example, you can create a *mixermap* (or control *panel*) in a sequencer and use this to adjust the pan and level settings of each part while a sequence is playing. Moreover, if these adjustments are recorded by the sequencer, automated control of the part settings can be realised. In addition, “Get” options allow suitable software to determine the current settings of a Multi.

More advanced functions are also provided to facilitate control over the multis, including creation, deletion and selection of a part’s program. Note that modification of a Multi part’s program must be done through the Program Section [&14{20}]. This can be easily achieved by getting the name of the requested part’s program, and using this name to select the program in the Program Section, whereupon changes may be made.

When modifying a Multi, it is not necesary to send the Multi’s name with every SysEx message. Instead, the *currently selected* Multi is always acted on with these messages. There are three ways to select a Multi and make it *current*: (i) Create a new Multi; (ii) select an existing Multi in memory to be the *current* one; and (iii) change the current Multi via the sampler’s front panel. The first method can only be used if a program of the specified name does not already exist in memory, the last method will only work if the *synchronisation* option is on (this is the default option; see SysEx section on how to change this) and the current mode is “MULTI” mode.

Handles used to access Multis are special 28-bit values which uniquely identify Multis in memory.

When information about a Multi is requested with a *Get* message, the data is returned in a “REPLY” confirmation message (see *Confirmation Messages* on page 5). The format of the data content of these messages is summarised in Table 27 and Table 29.

Table 26: Control Items for Section &18{24} — Multi Main

<Item>	<Data1>	<Data2>…	Description of Item
	Generic List Functions		
&01{1}	*NA*	*NA*	Get number of items in memory
&02{2}	BYTE(0–3)	*NA*	Get list of info for all items: <Data1>: 0=list of handles; 1=list of names; 2=list of handle+name; 3=list of handle+modified/tagged name
&03{3}	DWORD	*NA*	Select *current* item by handle
&04{4}	STRING	*NA*	Select *current* item by name
&05{5}	*NA*	*NA*	Get handle of *current* item
&06{6}	*NA*	*NA*	Get name of *current* item
&07{7}	DWORD	*NA*	Get item name from handle
&08{8}	STRING	*NA*	Get item handle from name
&09{9}	*NA*	*NA*	Delete ALL items from memory
&0A{10}	*NA*	*NA*	Delete *current* item from memory
&0B{11}	DWORD	*NA*	Delete item represented by handle <Data1>
&0C{12}	STRING	*NA*	Rename *current* item
&0D{13}	DWORD	STRING	Rename item represented by handle <Data1>
&0E{14}	BYTE(0−7)	BYTE(0, 1) BYTE(0, 1)	Set Tag Bit <Data1> = bit to set, <Data2> = (0=OFF, 1=ON) <Data3> = (0=CURRENT, 1=ALL)
&0F{15}	*NA*	*NA*	Get Tag Bitmap
&10{16}	*NA*	*NA*	Get name of *current* item with modified/tagged info.

Table 26: Control Items for Section &18{24} — Multi Main

<Item>	<Data1>	<Data2>…	Description of Item
&11 {17}	*NA*	*NA*	Get modified state of *current* item.
&18 {24}	BYTE(0−7)	*NA*	Delete tagged items <Data1> = tag bit
	General Multi Functions		
&40 {64}	WORD	STRING	Create New Multi <Data1> = number of parts, <Data2> = name
&41 {65}	STRING	*NA*	Copy Multi <Data1> = name of new multi
&42 {66}	BYTE	*NA*	Delete Part <Data1> = index of part to delete
&43 {67}	*NA*	*NA*	Delete Unused Parts
&44 {68}	*NA*	*NA*	Arrange Parts (sort by MIDI channel)

Table 27: Format of "REPLY" confirmation messages for Section &18{24} — Multi Main

<Item> requested	<Reply>…	Description of Data Returned
&01 {1}	DWORD	Get number of items in memory
&02 {2}	DWORD[n] *or* STRING[n] *or* (DWORD, STRING)[n]	Get list of info for all items: <Data1>: 0=list of handles; 1=list of names; 2=list of handle+name; 3=list of handle+modified/tagged name An array of handles, names, or a combination of both (handle, name) will be returned in a single data stream. There will be *n* sets of data, where *n* = number of items in memory.
&05 {5}	DWORD	Get handle of *current* item
&06 {6}	STRING	Get name of *current* item
&07 {7}	STRING	Get item name from handle
&08 {8}	DWORD	Get item handle from name
&0F {15}	WORD	Get Tag Bitmap
&10 {16}	STRING	Get name of *current* item with modified/tagged info. The name is modified to indicate the current *modified* and *tagged* state of the item.
&11 {17}	BYTE	Get modified state of *current* item <Reply> = (0=NOT MODIFIED, 1=MODIFIED)
&40 {64}	DWORD	Create Item <Reply> = handle of new item

Table 28: Control Items for Section &1A{26} — Multi Set Parameter

<Item>	<Data1>	<Data2>…	Description of Item
	Setting General Multi Information		
&01 {1}	BYTE	*NA*	Set Group ID
&02 {2}	BYTE(0–2)	*NA*	Set Multi Select <Data1> = (0=OFF, 1=BANK, 2=PROG CHANGE)
&03 {3}	BYTE(0–31)	*NA*	Set Multi Select Channel <Data1> = (1A=0, 2A=1, …, 16B=31)
&04 {4}	WORD	*NA*	Set Multi Tempo <Data1> = 10×bpm
&08 {8}	WORD(0–128)	*NA*	Set Multi Program Number Data1: (0=OFF, 1–128)
&09 {9}	BYTE(0–127)[a]	DWORD	Set Multi Part by handle <Data1> = Part Number; <Data2> = Handle of program

Table 28: Control Items for Section &1A{26} — Multi Set Parameter

<Item>	<Data1>	<Data2>…	Description of Item
&0A {10}	BYTE(0–127)[a]	STRING	Set Multi Part by name <Data1> = Part Number; <Data2> = Name of part
&0F {15}	WORD	*NA*	Set Number of Parts (resize) <Data1> = new number of parts
	Setting Multi Part Parameters (Data1 = Part Number-1[a])		
&10 {16}$_{\text{RT}}$	BYTE(0–127)	BYTE(0–31)	Set Part MIDI Channel, <Data2> = (1A=0, 2A=1, …, 16B=31)
&11 {17}$_{\text{RT}}$	BYTE(0–127)	BYTE(0, 1)	Set Part Mute, <Data2> = (0=OFF, 1=ON)
&12 {18}$_{\text{RT}}$	BYTE(0–127)	BYTE(0, 1)	Set Part Solo, <Data2> = (0=OFF, 1=ON)
&13 {19}$_{\text{RT}}$	BYTE(0–127)	SWORD(−600–+60)	Set Part Level, <Data2> = PartLevel in 10×dB
&14 {20}$_{\text{RT}}$	BYTE(0–127)	BYTE(0–14)	Set Part Output, <Data2> = (Output: 0 = L/R; 1–4 = op1/2–op7/8; 5–14 = L, R, op1-op8)
&15 {21}$_{\text{RT}}$	BYTE(0–127)	BYTE(0–100)	Set Part Pan/Balance, <Data2> = (Pan/Bal (0–100 = L50–R50); centre=&32 {50})
&16 {22}$_{\text{RT}}$	BYTE(0–127)	BYTE(0–4)	Set Part Effects Channel: <Data2> = (0=OFF, 1=FX1, 2=FX2, 3=RV3, 4=RV4)
&17 {23}$_{\text{RT}}$	BYTE(0–127)	SWORD(−600–+60)	Set Part FX Send Level <Data2> = level in 10×dB
&18 {24}$_{\text{RT}}$	BYTE(0–127)	SWORD(0–±3600)	Set Part Cents Tune: <Data2> = cents tuning
&1A {26}$_{\text{RT}}$	BYTE(0–127)	BYTE	Set Part Low Note
&1B {27}$_{\text{RT}}$	BYTE(0–127)	BYTE	Set Part High Note
&1C {28}$_{\text{RT}}$	BYTE(0–127)	BYTE(0–3)	Set Part Priority <Data1> = (0=HOLD, 1=HIGH, 2=NORM, 3=LOW)
&1D {29}$_{\text{RT}}$	BYTE(0–127)	WORD(0–128)	Set Multi Part Program Number <Data2> = (0=OFF, 1–128)
&1F {31}	BYTE(0–127)	BYTE	Set Part Group ID
	Multi EQ		
&30 {48}	BYTE	*NA*	Set EQ Output Channel
&31 {49}	BYTE(0, 1)	*NA*	Set EQ Enable <Data2> = (0=OFF, 1=ON)
&32 {50}	SWORD	*NA*	Set EQ Low Gain
&33 {51}	WORD	*NA*	Set EQ Low Frequency
&34 {52}	SWORD	*NA*	Set EQ Mid Gain
&35 {53}	WORD	*NA*	Set EQ Mid Frequency
&36 {54}	SWORD	*NA*	Set EQ High Gain
&37 {55}	DWORD	*NA*	Set EQ High Frequency
	MIDI Filter		
&40 {64}	BYTE(0–127)	BYTE(0, 1)	Set MIDI filter enable <Data1> = part <Data2> = (0=OFF, 1=ON)

&41 {65}	BYTE(0–127)	BYTE(0–100)
		BYTE(0, 1)
			Set MIDI filter (by part) <Data1> = part
			<Data2> = filter type = (0=NOTE ON, 1=POLY ATCH, 2=CHAN ATCH, 3=PITCH BEND, 4=SYSEX, 5–36=CTL0–31, 37–100=CTL64–127)
			<Data3> = pass state (0=YES, 1=NO)

&42 {66}	BYTE(0–31)	BYTE(0–100)
		BYTE(0, 1)
			Set MIDI filter (by channel) <Data1> = MIDI channel
			<Data2> = filter type = (0=NOTE ON, 1=POLY ATCH, 2=CHAN ATCH, 3=PITCH BEND, 4=SYSEX, 5–36=CTL0–31, 37–100=CTL64–127)
			<Data3> = pass state (0=YES, 1=NO)

	QLink Controls (Data1 = Qlink control, 0–7 = QLink 1–8)		
&50 {80}	BYTE(0–7)	BYTE(0, 1)	Set Assign Type <Data2> = (0=PART, 1=FX)

Table 28: Control Items for Section &1A{26} — Multi Set Parameter

<Item>	<Data1>	<Data2>…	Description of Item
&51 {81}	BYTE(0–7)	WORD	Set Target Part <Data2> = (0=ALL, 1–128=PART) *(when Assign Type = PART)* Set Target FX channel <Data2> = (0–3 = FX 1–FX 4) *(when Assign Type = FX)*
&52 {82}	BYTE(0–7)	WORD	Set Destination
&53 {83}	BYTE(0–7)	BYTE(0, 1)	Set Change Type <Data2> = (0=REPLACE, 1=OFFSET)
&54 {84}	BYTE(0–7)	SBYTE	Set Scale Minimum
&55 {85}	BYTE(0–7)	SBYTE	Set Scale Maximum
&56 {86}	BYTE(0–7)	WORD(0–128)	Set MIDI controller output (0=OFF, 1–128=CTRL)
&57 {87}	BYTE(0–7)	BYTE	Set MIDI channel output *(when Assign Type = FX)*

[a] The number set is 1 smaller than that set via the front panel. *i.e.,* numbers 1–128 shall be transmitted as 0–127.

Table 29: Control Items for Section &1B{27} — Multi Get Parameter

<Item>	<Data1>…	<Reply>…	Description of Item
	Getting General Multi Information		
&01 {1}	*NA*	BYTE	Get Group ID
&02 {2}	*NA*	BYTE(0–2)	Get Multi Select <Reply1> = (0=OFF, 1=BANK, 2=PROG CHANGE)
&03 {3}	*NA*	BYTE(0–31)	Get Multi Select Channel <Reply1> = (1A=0, 2A=1, …, 16B=31)
&04 {4}	*NA*	WORD	Get Multi Tempo <Reply> = 10×bpm
&08 {8}	*NA*	WORD(0–128)	Get Multi Program Number
&09 {9}	BYTE(0–127)[a]	DWORD	Get Multi Part handle. <Data1> = Part Number; <Reply> = Handle of program
&0A {10}	BYTE(0–127)[a]	STRING	Get Multi Part name. <Data1> = Part Number; <Reply> = Name of part
&0F {15}	*NA*	WORD	Get Number of Parts. <Reply> = new number of parts
	Getting Multi Part Parameters (Data1 = Part Number-1[a])		
&10 {16}$_{RT}$	BYTE(0–127)	BYTE(0–31)	Get Part MIDI Channel, <Reply> = (1A=0, 2A=1, …, 16B=31)
&11 {17}$_{RT}$	BYTE(0–127)	BYTE(0, 1)	Get Part Mute, <Reply> = (0=OFF, 1=ON)
&12 {18}$_{RT}$	BYTE(0–127)	BYTE(0, 1)	Get Part Solo, <Reply> = (0=OFF, 1=ON)
&13 {19}$_{RT}$	BYTE(0–127)	SWORD(−600–+60)	Get Part Level, <Reply> = PartLevel in 10×dB
&14 {20}$_{RT}$	BYTE(0–127)	BYTE(0–14)	Get Part Output, <Reply> = (Output: 0 = L/R; 1–4 = op1/2–op7/8; 5–14 = L, R, op1-op8)
&15 {21}$_{RT}$	BYTE(0–127)	BYTE(0–100)	Get Part Pan/Balance, <Reply> = Pan/Bal (0–100 = L50–R50); centre=&32{50}
&16 {22}$_{RT}$	BYTE(0–127)	BYTE(0–4)	Get Part Effects Channel: Reply = (0=OFF, 1=FX1, 2=FX2, 3=RV3, 4=RV4)
&17 {23}$_{RT}$	BYTE(0–127)	SWORD(−600–+60)	Get Part FX Send Level <Reply> = level in 10×dB
&18 {24}$_{RT}$	BYTE(0–127)	SWORD(0–±3600)	Get Part Cents Tune
&1A {26}$_{RT}$	BYTE(0–127)	BYTE	Get Part Low Note
&1B {27}$_{RT}$	BYTE(0–127)	BYTE	Get Part High Note

Table 29: Control Items for Section &1B{27} — Multi Get Parameter

<Item>	<Data1>…	<Reply>…	Description of Item
&1C {28}$_{RT}$	BYTE(0–127)	BYTE(0–3)	Get Part Priority <Reply> = (0=HOLD, 1=HIGH, 2=NORM, 3=LOW)
&1D {29}$_{RT}$	BYTE(0–127)	WORD(0–128)	Get Multi Part Program Number
&1F {31}	BYTE(0–127)	BYTE	Get Part Group ID
	Multi EQ		
&30 {48}	*NA*	BYTE	Get EQ Output Channel
&31 {49}	*NA*	BYTE(0, 1)	Get EQ Enable <Reply> = (0=OFF, 1=ON)
&32 {50}	*NA*	SWORD	Get EQ Low Gain
&33 {51}	*NA*	WORD	Get EQ Low Frequency
&34 {52}	*NA*	SWORD	Get EQ Mid Gain
&35 {53}	*NA*	WORD	Get EQ Mid Frequency
&36 {54}	*NA*	SWORD	Get EQ High Gain
&37 {55}	*NA*	DWORD	Get EQ High Frequency
	MIDI Filter		
&40 {64}	BYTE(0–127)	BYTE(0, 1)	Get MIDI filter enable <Data1> = part, <Reply> = (0=OFF, 1=ON)

&41 {65}	BYTE(0–127)
	BYTE(0–100)
		BYTE(0, 1)	Get MIDI filter (by part) <Data1> = part <Data2> = filter type = (0=NOTE ON, 1=POLY ATCH, 2=CHAN ATCH, 3=PITCH BEND, 4=SYSEX, 5–36=CTL0–31, 37–100=CTL64–127) <Reply> = pass state (0=YES, 1=NO)

&42 {66}	BYTE(0–31)
	BYTE(0–100)
		BYTE(0, 1)	Get MIDI filter (by channel) <Data1> = MIDI channel <Data2> = filter type = (0=NOTE ON, 1=POLY ATCH, 2=CHAN ATCH, 3=PITCH BEND, 4=SYSEX, 5–36=CTL0–31, 37–100=CTL64–127) <Reply> = pass state (0=YES, 1=NO)

&48 {72}	*NA*	BYTE[n]	Get mute/solo state for all multi parts. An array of BYTE is returned: one BYTE per part in the multi. Value = (0=mute and solo off; 1=mute on; 2=solo on)
	QLink Controls (Data1 = Qlink control, 0–7 = QLink 1–8)		
&50 {80}	BYTE(0–7)	BYTE(0, 1)	Get Assign Type <Reply> = (0=PART, 1=FX)
&51 {81}	BYTE(0–7)	WORD	Get Target Part <Reply> = (0=ALL, 1–128=PART) *(when Assign Type = PART)* Get Target FX channel <Reply> = (0–3 = FX 1–FX 4) *(when Assign Type = FX)*
&52 {82}	BYTE(0–7)	WORD	Get Destination
&53 {83}	BYTE(0–7)	BYTE(0, 1)	Set Change Type <Reply> = (0=REPLACE, 1=OFFSET)
&54 {84}	BYTE(0–7)	SBYTE	Get Scale Minimum
&55 {85}	BYTE(0–7)	SBYTE	Get Scale Maximum
&56 {86}	BYTE(0–7)	WORD(0–128)	Get MIDI controller output <Reply> = (0=OFF, 1–128=CTRL)
&57 {87}	BYTE(0–7)	BYTE	Get MIDI channel output
&58 {88}	BYTE(0–7)	STRING	Get Name of Destination FX parameter *(when Assign Type = FX)* Get Name of Modulation Destination *(when Assign Type = PART)*

---

a. The number set is 1 smaller than that set via the front panel. *i.e.,* numbers 1–128 shall be transmitted as 0–127.

Item List for Sample section [&1C{28}]

The Sample section provides a means to determine which samples are currently in memory and to determine or modify their basic parameters — off-line sample processing functions are not supported. Once the names of samples are determined, they can be assigned to a keygroup zone using the functions of the Keygroup Zone section &0E{14}.

Note that the active sample displayed on the LCD screen will only correspond to the one selected via SysEx if the *synchronisation* option is on (this is the default option; see SysEx section on how to change this) and the current mode is “SAMPLE” mode.

Handles used to access Samples are special 28-bit values which uniquely identify Samples in memory.

When information about a Sample is requested with a *Get* message, the data is returned in a “REPLY” confirmation message (see *Confirmation Messages* on page 5). The format of the data content of these messages is summarised in Table 31 and Table 33.

Table 30: Control Items for Section &1C{28} — Sample Main

<Item>	<Data1>	<Data2>…	Description of Item
	Generic List Functions		
&01{1}	*NA*	*NA*	Get number of items in memory
&02{2}	BYTE(0–3)	*NA*	Get list of info for all items: <Data1>: 0=list of handles; 1=list of names; 2=list of handle+name; 3=list of handle+modified/tagged name
&03{3}	DWORD	*NA*	Select *current* item by handle
&04{4}	STRING	*NA*	Select *current* item by name
&05{5}	*NA*	*NA*	Get handle of *current* item
&06{6}	*NA*	*NA*	Get name of *current* item
&07{7}	DWORD	*NA*	Get item name from handle
&08{8}	STRING	*NA*	Get item handle from name
&09{9}	*NA*	*NA*	Delete ALL items from memory
&0A{10}	*NA*	*NA*	Delete *current* item from memory
&0B{11}	DWORD	*NA*	Delete item represented by handle <Data1>
&0C{12}	STRING	*NA*	Rename *current* item
&0D{13}	DWORD	STRING	Rename item represented by handle <Data1>
&0E{14}	BYTE(0–7)	BYTE(0, 1) BYTE(0, 1)	Set Tag Bit <Data1> = bit to set, <Data2> = (0=OFF, 1=ON) <Data3> = (0=CURRENT, 1=ALL)
&0F{15}	*NA*	*NA*	Get Tag Bitmap
&10{16}	*NA*	*NA*	Get name of *current* item with modified/tagged info
&11{17}	*NA*	*NA*	Get modified state of *current* item.
&18{24}	BYTE(0–7)	*NA*	Delete tagged items <Data1> = tag bit
	General Sample Functions		
&40{64}	BYTE	BYTE	Start auditioning the current sample <Data1> = velocity <Data2> = (NO LOOPING, 1=LOOPING)
&41{65}	*NA*	*NA*	Stop playback of the current sample
&42{66}	BYTE	QWORD	Play To <Data1> = velocity, <Data2> = sample position
&43{67}	BYTE	QWORD	Play From <Data1> = velocity, <Data2> = sample position

Table 30: Control Items for Section &1C{28} — Sample Main

<Item>	<Data1>	<Data2>…	Description of Item
&44 {68}	BYTE	QWORD	Play Over <Data1> = velocity, <Data2> = sample position
&45 {69}	BYTE	BYTE	Play Loop <Data1> = velocity, <Data2> = loop index
&46 {70}	BYTE	BYTE	Play Region <Data1> = velocity, <Data2> = region index
	Loop and Region Creation/Deletion		
&48 {72}	*NA*	*NA*	Create New Loop
&49 {73}	BYTE	*NA*	Delete Loop <Data1> = index
&4A {74}	*NA*	*NA*	Create Region
&4B {75}	BYTE	*NA*	Delete Region <Data1> = index

Table 31: Format of “REPLY” confirmation messages for Section &1C{28} — Sample Main

<Item> requested	<Reply>…	Description of Data Returned
&01 {1}	DWORD	Get number of items in memory
&02 {2}	DWORD[n] *or* STRING[n] *or* (DWORD, STRING)[n]	Get list of info for all items: <Data1>: 0=list of handles; 1=list of names; 2=list of handle+name; 3=list of handle+modified/tagged name An array of handles, names, or a combination of both (handle, name) will be returned in a single data stream. There will be *n* sets of data, where *n* = number of items in memory.
&05 {5}	DWORD	Get handle of *current* item
&06 {6}	STRING	Get name of *current* item
&07 {7}	STRING	Get item name from handle
&08 {8}	DWORD	Get item handle from name
&0F {15}	WORD	Get Tag Bitmap
&10 {16}	STRING	Get name of *current* item with modified/tagged info. The name is modified to indicate the current *modified* and *tagged* state of the item.
&11 {17}	BYTE	Get modified state of *current* item <Reply> = (0=NOT MODIFIED, 1=MODIFIED)

Table 32: Control Items for Section &1E{29} — Sample Parameter Set

<Item>	<Data1>	<Data2>…	Description of Item
	Setting General Options		
&01 {1}	BYTE	*NA*	Set Group ID
	Setting Parameters		
&20 {32}	QWORD	*NA*	Set Trim Start
&21 {33}	QWORD	*NA*	Set Trim End
&22 {34}	QWORD	*NA*	Set Trim Length
&24 {36}	BYTE	*NA*	Set Original Pitch

Table 32: Control Items for Section &1E{29} — Sample Parameter Set

<Item>	<Data1>	<Data2>…	Description of Item
&25 {37}	SWORD (0–±3600)	*NA*	Set Cents Tune
&26 {38}	BYTE(0–2)	*NA*	Set Playback Mode, where <Data1> = (0=NO LOOPING, 1=LOOPING, 2=ONE SHOT)
	Set Loop Parameters <Data1> = loop index		
&30 {48}	BYTE	QWORD	Set Loop Start
&31 {49}	BYTE	QWORD	Set Loop End
&32 {50}	BYTE	QWORD	Set Loop Length
&33 {51}	BYTE	BYTE(0, 1)	Set Loop Lock <Data1> = (0=OFF, 1=ON)
&34 {52}	BYTE	SBYTE (0–±50)	Set Loop Tune
&35 {53}	BYTE	BYTE(0, 1)	Set Loop Direction <Data1> = (0=FORWARDS, 1=ALTERNATING)
&36 {54}	BYTE	BYTE(0, 1)	Set Loop Type <Data1> = (0=LOOP IN REL, 1=LOOP UNTIL REL)
&37 {55}	BYTE	BYTE	Set Number of Loop Repetitions (0=INFINITE)
	Set Region Parameters		
&40 {64}	BYTE(0–31)	QWORD	Set Region Start <Data1> = Region Num, <Data2> = start
&41 {65}	BYTE(0–31)	QWORD	Set Region End <Data1> = Region Num <Data2> = end
&42 {66}	BYTE(0–31)	QWORD	Set Region Length <Data1> = Region Num <Data2> = length

Table 33: Control Items for Section &1F{30} — Sample Parameter Get

<Item>	<Data1>…	<Reply>…	Description of Item
	Getting General Options		
&01 {1}	*NA*	BYTE	Get Group ID
	Getting Parameters		
&20 {32}	*NA*	QWORD	Get Trim Start
&21 {33}	*NA*	QWORD	Get Trim End
&22 {34}	*NA*	QWORD	Get Trim Length
&24 {36}	*NA*	BYTE	Get Original Pitch
&25 {37}	*NA*	SWORD (0–±3600)	Get Cents Tune
&26 {38}	*NA*	BYTE(0–2)	Get Playback Mode, where <Data1> = (0=NO LOOPING, 1=LOOPING, 2=ONE SHOT)
	Get Loop Parameters <Data1> = loop index		
&30 {48}	BYTE	QWORD	Get Loop Start
&31 {49}	BYTE	QWORD	Get Loop End
&32 {50}	BYTE	QWORD	Get Loop Length
&33 {51}	BYTE	BYTE(0, 1)	Get Loop Lock <Data1> = (0=OFF, 1=ON)
&34 {52}	BYTE	SBYTE (0–±50)	Get Loop Tune
&35 {53}	BYTE	BYTE(0, 1)	Get Loop Direction <Data1> = (0=FORWARDS, 1=ALTERNATING)
&36 {54}	BYTE	BYTE(0, 1)	Get Loop Type <Data1> = (0=LOOP IN REL, 1=LOOP UNTIL REL)

Table 33: Control Items for Section &1F{30} — Sample Parameter Get

<Item>	<Data1>…	<Reply>…	Description of Item
&37{55}	BYTE	BYTE	Get Number of Loop Repetitions
&38{56}	*NA*	BYTE	Get Number of Loops
	Get Region Parameters		
&40{64}	BYTE(0–31)	QWORD	Get Region Start <Data1> = Region Num, <Reply1> = start
&41{65}	BYTE(0–31)	QWORD	Get Region End <Data1> = Region Num <Reply1> = end
&42{66}	BYTE(0–31)	QWORD	Get Region Length <Data1> = Region Num <Reply1> = length
&44{68}	BYTE	*NA*	Get Number of Regions
	Get Read-only Information		
&50{80}	*NA*	QWORD	Get Sample Length
&51{81}	*NA*	DWORD	Get Sample Rate [Hz]
&52{82}	*NA*	BYTE	Get Sample Bit-Depth [bits]
&54{83}	*NA*	BYTE(0, 1)	Get Sample Type <Reply> = (0=RAM, 1=VIRTUAL)
&55{84}	*NA*	BYTE	Get Number of Channels

Item List for Disk Tools section [&20{32}]

All disk operations are performed on the *currently selected* disk. Therefore, before the first disk access is attempted, one must select a disk. (Note that the selected disk is NOT the same as that selected via the front panel.) Disks are selected using a unique 14-bit handle, initially obtained when the list of connected disks is determined (Item &05{5}).

Before any disk operations are performed, the disk list must be updated. This can either be done via the sampler's front panel, or via SysEx. Indeed, the disk list is always updated when the sampler is first switched on. Note that if a disk is added to, or removed from, the sampler, the disk list must always be updated, or incorrect operation may result. A disk must be selected before any further disk operations will be valid. Once this is done, information about the files and folders on the disk can be obtained.

Table 34: Control Items for Section &20{32} — Disk Tools

<Item>	<Data1>	<Data2>	Description of Item
	General Disk Functions		
&01{1}	*NA*	*NA*	Update the list of disks connected
&02{2}	WORD	*NA*	Select Disk <Data1> = Disk Handle
&03{3}[a]	WORD	*NA*	Test if the disk is valid <Data1> = Disk Handle
&04{4}[b]	*NA*	*NA*	Get the number of disks connected
&05{5}	*NA*	*NA*	Get list of all connected disks
&09{9}	*NA*	*NA*	Get current path of current disk
&0D{13}	WORD	*NA*	Eject Disk <Data1> = Disk Handle
	Folder Functions		
&10{16}	*NA*	*NA*	Get number of sub-folders in the current folder.
&12{18}	*NA*	*NA*	Get the names of all of the sub-folders in the current folder.
&13{19}	STRING	*NA*	Open Folder. This sets the current folder to be the requested one. (Move down one level in the folder heirarchy.) <Data1> = name of folder to open.. (If <Data1> = 0, the root folder will be selected.)
&15{21}	STRING	*NA*	Load Folder: the selected folder, and all its contents (including sub-folders) are loaded into memory. <Data1> = name of folder to load.
&16{22}	STRING	*NA*	Create Folder: Creates a sub-folder in the currently selected folder. <Data1> = name of folder to create.
&17{23}	STRING	*NA*	Delete Sub-Folder. <Data1> = name of folder to delete.
&18{24}	STRING	STRING	Rename Folder: <Data1> = name of folder to rename, <Data2> = new name for folder.
	File Functions		
&20{32}	*NA*	*NA*	Get number of files in the current folder.
&22{34}	*NA*	*NA*	Get the names of all of the files in the current folder.
&28{40}	STRING	STRING	Rename File: <Data1> = name of file to rename, <Data2> = new name for file.
&29{41}	STRING	*NA*	Delete File. <Data1> = name of file to delete.
&2A{42}[c]	STRING	*NA*	Load File <Data1> = name of file to load. The filename must include the correct file extension.

Table 34: Control Items for Section &20{32} — Disk Tools

<Item>	<Data1>	<Data2>	Description of Item
&2B {43}[d]	STRING	*NA*	Load File including all dependent child files. <Data1> = name of file to load. The filename must include the correct file extension.

&2C {44}	DWORD	BYTE(1–4)
		BYTE(0, 1)
		BYTE(0, 1)
			Save Memory Item to Disk
			<Data1> = Handle of Memory Item
			<Data2> = Type = (1=Multi; 2=Program; 3=Sample; 4=SMF)
			<Data3> = (0=Skip if file exists; 1=Overwrite existing files)
			<Data4> = (0=Don’t save children; 1=Save Children)

&2D {45}	BYTE(0–4)	BYTE(0, 1) BYTE(0, 1)	Save All Memory Items to Disk <Data1> = Type = (0=All; 1=Multi; 2=Program; 3=Sample; 4=SMF) <Data2> = (0=Skip if file exists; 1=Overwrite existing files) <Data3> = (0=Don’t save children; 1=Save Children)

[a] If the currently selected disk is valid, a DONE message will be returned, otherwise an ERROR reply will be returned.

[b] Before the number of disks is determined, the list of disks connected must have been updated. If this is not the case, the value returned will be invlaid.

[c] Loading a file will load a multi, program, sample, *etc.*, depending on the type of file supplied. Note that if the requested file has child files which it depends upon (*e.g.,* a program’s children are the sample files it requires) then these child files will not be loaded automatically. To load child files automatically, another function must be used.

[d] This operation ensures that all child files (*e.g.,* the samples of a program, or the programs of a multi) are also loaded automatically.

When information is requested in the Disk Tools section, it is returned in a REPLY message — unless an error occurs, whereupon an ERROR confirmation message is returned instead. The format of these REPLY messages is detailed in Table 35.

Table 35: Format of “REPLY” confirmation messages for Section &20{32} — Disk Tools

<Item> requested	<Reply…>	Description of Data Returned
	General Disk Functions	
&04 {4}	BYTE	Number of logical disks connected to the sampler.
&05 {5}[a]	WORD BYTE BYTE BYTE BYTE(0, 1) STRING	<Reply1> = Disk handle <Reply2> = Disk type = (0=floppy; 1=hard drive; 2=cd-rom; 3=optical; 4=unknown) <Reply3> = Disk Format = (1=FAT; 2=FAT32; 3=ISO9660; 4=S1000; 5=S3000; 6=EMU; 7=Roland; 8=CD-AUDIO; 100=empty) <Reply4> = Physical Disk ID <Reply5> = (0=not-writable; 1=writable) <Reply6> = Volume name This sequence is repeated for each of the logical disks connected to the sampler.
&09 {9}	STRING	<Reply1> = current path on the current disk. If the root folder is selected, this will return a single byte = 0.
	Folder Functions	
&10 {16}	WORD	The number of sub-folders in the current folder
&12 {18}[a]	STRING[]	Returns a list of the names of all of the sub-folders
	File Functions	
&20 {32}	WORD	The number of files in the current folder
&22 {34}[a]	STRING[]	Returns a list of the names of all the files in the current folder

[a] It is recommended that the control computer determines the amount of data it expects before this <Item> is requested.

Item List for Multi FX Control section [&24{36}]

Effects settings belong to a Multi, and this section enables the adjustment of effects for the currently selected multi (see Section &18 for information on how to select a multi to be current). The SysEx protocol for controlling the effects has been designed to be as flexible and extensible as possible; rather than being tied to a particular hardware architecture. To achieve this, the hardware is presented as a series of effects channels, each with a certain number of modules. These modules can be set to include an effect (or no effect). Note that certain restrictions may apply to which combinations of effects may be used simultaneously depending on the hardware installed. To illustrate the layout, an example FX configuration is shown in Figure 1.

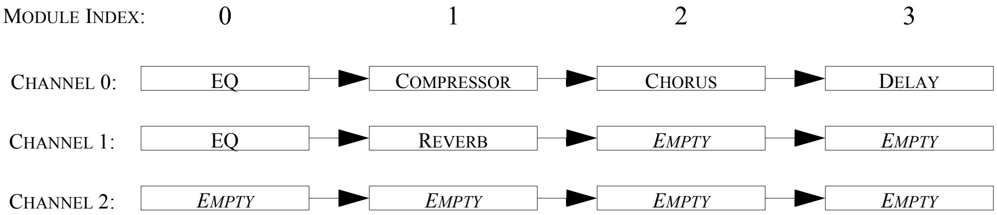

Figure 1: Example FX configuration

Prior to use, the configuration of all of the effects channels and modules must be determined so that the layout of the current hardware is known. Only then should parameters be modified or effects be selected into modules.

Note that all of the index values are zero-based. (*i.e.,* the first item = 0, the second item = 1.)

Parameter values are always passed as as SDWORD even although many parameters only require a single unsigned byte. This provides flexibility and means that different messages do not need to be used for different parameter types.

Table 36: Control Items for Section &24{36} — Multi FX Main

<Item>	<Data1>	<Data2>…	Description of Item
	Determine if there is an FX board installed or not		
&01 {1}	*NA*	*NA*	Get FX card installed
	Determine the characteristics of the installed FX hardware		
&10 {16}	*NA*	*NA*	Get Number of FX channels
&11 {17}	BYTE	*NA*	Get Maximum Number of FX modules on given channel <Data1> = channel
	Get Information about available FX		
&20 {32}	*NA*	*NA*	Get Number of effects available
&21 {33}	WORD	*NA*	Get Name of effect <Data1> = index of effect
&22 {34}[a]	WORD	*NA*	Get Unique ID of effect <Data1> = index of effect
&24 {36}[b]	WORD	*NA*	Get Parameter Index for Output Control <Data1> = index of effect

Table 36: Control Items for Section &24{36} — Multi FX Main

<Item>	<Data1>	<Data2>…	Description of Item
	Get Information about FX parameters: <Data1> = channel, <Data2> = module, <Data3> = parameter		
&30{48}	BYTE	BYTE	Get Number of Parameters
&31{49}	BYTE	BYTE BYTE	Get Parameter Minimum
&32{50}	BYTE	BYTE BYTE	Get Parameter Maximum
&33{51}	BYTE	BYTE BYTE	Get Parameter Name
&34{52}	BYTE	BYTE BYTE	Get Parameter Units
&35{53}	BYTE	BYTE BYTE	Get Parameter Type
&36{54}[c]	BYTE	BYTE BYTE	Get Display Template for parameter
&38{56}	BYTE	BYTE BYTE	Get Parameter Position ID
&40{64}[d]	BYTE	BYTE	Get Number of Parameter Groups
&41{65}	BYTE	BYTE BYTE	Get Group Name <Data3> = Index of Group
&42{66}	BYTE	BYTE BYTE	Get Group Index of Parameter

a. Each effect has a globally unique ID: the top 14 bits represent a manufacturer ID, the bottom 14 bits represent the effect ID.

b. The parameter index for the output control is a special index used to access the output control parameter. This parameter is not actually a part of the parameter list, and is not included in the number of parameter count. Using a parameter index provides a consistent interface to access the control. This index must not clash with any existing parameters.

c. Display templates are used to provide a hint as to how the parameter should be displayed and implicity defines any scaling required. For example, a level might have the template 0.00, which means that a value of 100 would be displayed as 1.00 (*i.e.,* the integer representation of the number is divided by 100 to yield the floating point representation). This makes it simple to transfer both integer and floating point values using an SDWORD.

d. To provide a guide to displaying parameters on a user-interface, parameters may be grouped into logical sections. Each parameter can have a group index associated with it, and each group can have a name. For example, a stereo delay could have 3 groups: *delay*, *damping* and *feedback*. An effect does not need to contain groups, in which case Number of Groups = 0.

Table 37: Format of “REPLY” confirmation messages for Section &24{36} — Multi FX Control

<Item> requested	<Reply>…	Description of Data Returned
&01{1}	BYTE	FX card installed: 0=none, 2=Z48
&10{16}	BYTE	Number of FX channels
&11{17}	BYTE	Maximum Number of FX modules on given channel
&20{32}	WORD	Get Number of effects available
&21{33}	STRING	Get Name of effect
&22{34}	DWORD	Get Unique ID of effect
&24{36}	BYTE	Get Parameter Index for Output Control
&30{48}	BYTE	Get Number of Parameters
&31{49}	SDWORD	Get Parameter Minimum
&32{50}	SDWORD	Get Parameter Maximum

Table 37: Format of “REPLY” confirmation messages for Section &24{36} — Multi FX Control

<Item> requested	<Reply>…	Description of Data Returned
&33 {51}	STRING	Get Parameter Name
&34 {52}	STRING	Get Parameter Units
&35 {53}	BYTE	Get Parameter Type <Reply> (0=NUMBER, 1=STRING-LIST)
&36 {54}	STRING	Get Display Template for parameter
&38 {56}	BYTE	Get Parameter Position ID
&40 {64}	BYTE	Get Number of Parameter Groups
&41 {65}	STRING	Get Group Name <Data3> = Index of Group
&42 {66}	BYTE	Get Group Index of Parameter

Table 38: Control Items for Section &26{38} — Multi FX Parameter Set

<Item>	<Data1>	<Data2>…	Description of Item
	Configuration of the FX channels <Data1> = channel		
&20 {32}	BYTE	BYTE(0, 1)	Set Mute Status of Channel <Data2> = (0=ON, 1=MUTE)
&21 {33}	BYTE	BYTE	Set Channel Input <Data2> = input
&22 {34}	BYTE	BYTE	Set Channel Output <Data2> = output
	Configuration of the FX modules		
&30 {48}	BYTE	BYTE STRING	Set effect in module on given channel (by name) <Data1> = channel; <Data2> = module; <Data3> = effect. If the effect name is NULL (*i.e.,* <Data3> = 0), then the specified module is set to have no effect
&31 {49}	BYTE	BYTE WORD	Set effect in module on given channel (by index) <Data1> = channel; <Data2> = module; <Data3> = effect
	Enabling/Disabling (bypassing) FX modules		
&40 {64}	BYTE	BYTE BYTE(0, 1)	Set Enabled/Disabled State of FX module. <Data1> = channel; <Data2> = module; <Data3> =0 (disable) or 1 (enable)
	FX parameter values		
&50 {80}	BYTE	BYTE BYTE SDWORD	Set parameter value of given module in given channel. <Data1> = channel; <Data2> = module; <Data3> = index of parameter to set <Data4> = parameter value
&52 {82}	BYTE	BYTE BYTE BYTE	Set Qlink control used to control the parameter <Data1> = channel; <Data2> = module; <Data3> = index of parameter to set <Data4> = Qlink control (0=NONE, 1–n = Qlink 1–n)

Table 39: Control Items for Section &27{39} — Multi FX Parameter Get

<Item>	<Data1>…	<Reply>…	Description of Item
	Configuration of the FX channels <Data1> = channel		
&20{32}	BYTE	BYTE(0, 1)	Get Mute Status of Channel <Reply> = (0=ON, 1=MUTE)
&21{33}	BYTE	BYTE	Get Channel Input <Reply> = input
&22{34}	BYTE	BYTE	Get Channel Output <Reply> = output
	Configuration of the FX modules		
&30{48}	BYTE BYTE	STRING	Get effect in module on given channel (by name) <Data1> = channel; <Data2> = module; <Reply> = effect
&31{49}	BYTE BYTE	WORD	Get effect in module on given channel (by index) <Data1> = channel; <Data2> = module; <Reply> = effect
	Enabling/Disabling FX modules		
&40{64}	BYTE BYTE	BYTE(0, 1)	Get Enabled/Disabled State of FX module <Data1> = channel; <Data2> = module; <Reply> = (0=disabled, 1=enabled)
	FX parameter values		
&50{80}	BYTE BYTE BYTE	SDWORD	Get parameter value of given module in given channel. <Data1> = channel; <Data2> = module; <Data3> = index of parameter <Reply> = parameter value
&51{81}	BYTE BYTE BYTE	STRING	Get parameter string of given module in given channel. <Data1> = channel; <Data2> = module; <Data3> = index of parameter <Reply> = parameter value (NOTE: This message is only used if the parameter type is STRING-LIST. For other parameter types, this command will return a zero-length string.)
&52{82}	BYTE BYTE BYTE	BYTE	Get Qlink control used to control the parameter <Data1> = channel; <Data2> = module; <Data3> = index of parameter to set <Reply> = Qlink control (0=NONE, 1–n = Qlink 1–n)

Item List for MIDI Song File Tools section [&28{40}]

The MIDI Song File section provides a means to determine which MIDI song files are currently in memory and to determine or modify their basic parameters.

Handles used to access Song Files are special 28-bit values which uniquely identify Song Files in memory.

When information about a Song Files is requested with a *Get* message, the data is returned in a "REPLY" confirmation message (see *Confirmation Messages* on page 5). The format of the data content of these messages is summarised in Table 41 and Table 43.

Table 40: Control Items for Section &28{40} — Song File Main

<Item>	<Data1>	<Data2>…	Description of Item
	Generic List Functions		
&01 {1}	*NA*	*NA*	Get number of items in memory
&02 {2}	BYTE(0–3)	*NA*	Get list of info for all items: <Data1>: 0=list of handles; 1=list of names; 2=list of handle+name; 3=list of handle+modified/tagged name
&03 {3}	DWORD	*NA*	Select *current* item by handle
&04 {4}	STRING	*NA*	Select *current* item by name
&05 {5}	*NA*	*NA*	Get handle of *current* item
&06 {6}	*NA*	*NA*	Get name of *current* item
&07 {7}	DWORD	*NA*	Get item name from handle
&08 {8}	STRING	*NA*	Get item handle from name
&09 {9}	*NA*	*NA*	Delete ALL items from memory
&0A {10}	*NA*	*NA*	Delete *current* item from memory
&0B {11}	DWORD	*NA*	Delete item represented by handle <Data1>
&0C {12}	STRING	*NA*	Rename *current* item
&0D {13}	DWORD	STRING	Rename item represented by handle <Data1>
&0E {14}	BYTE(0−7)	BYTE(0, 1) BYTE(0, 1)	Set Tag Bit <Data1> = bit to set, <Data2> = (0=OFF, 1=ON) <Data3> = (0=CURRENT, 1=ALL)
&0F {15}	*NA*	*NA*	Get Tag Bitmap
&10 {16}	*NA*	*NA*	Get name of *current* item with modified/tagged info.
&11 {17}	*NA*	*NA*	Get modified state of *current* item.
&18 {24}	BYTE(0−7)	*NA*	Delete tagged items <Data1> = tag bit
	General MIDI Song File Functions		
&40 {64}	*NA*	*NA*	Play Song
&41 {65}	*NA*	*NA*	Pause Song
&42 {66}	*NA*	*NA*	Stop Song

Table 41: Format of "REPLY" confirmation messages for Section &28{40} — Song File Main

<Item> requested	<Reply>…	Description of Data Returned
&01 {1}	DWORD	Get number of items in memory
&02 {2}	DWORD[n] *or* STRING[n] *or* (DWORD, STRING)[n]	Get list of info for all items: <Data1>: 0=list of handles; 1=list of names; 2=list of handle+name; 3=list of handle+modified/tagged name An array of handles, names, or a combination of both (handle, name) will be returned in a single data stream. There will be *n* sets of data, where *n* = number of items in memory.
&05 {5}	DWORD	Get handle of *current* item
&06 {6}	STRING	Get name of *current* item
&07 {7}	STRING	Get item name from handle
&08 {8}	DWORD	Get item handle from name
&0F {15}	WORD	Get Tag Bitmap
&10 {16}	STRING	Get name of *current* item with modified/tagged info. The name is modified to indicate the current *modified* and *tagged* state of the item.
&11 {17}	BYTE	Get modified state of *current* item <Reply> = (0=NOT MODIFIED, 1=MODIFIED)

Table 42: Control Items for Section &2A{42} — Song File Parameter Set

<Item>	<Data1>	<Data2>…	Description of Item
&01 {1}	Setting General Options		
	BYTE	*NA*	Set Group ID
	Setting Parameters		
&10 {16}	WORD	*NA*	Set From Bar
&11 {17}	WORD	*NA*	Set To Bar
&12 {18}	BYTE	*NA*	Set Tempo Mode <Data1> = (0=FILE, 1=MANUAL, 2=MULTI)
&13 {19}	WORD	*NA*	Set Manual Tempo <Data1> = (tempo×10) bpm
&18 {24}	BYTE(0–2)	*NA*	Set MIDI output port <Data1> = (0=NONE, 1=MIDI A, 2=MIDI B)

Table 43: Control Items for Section &2B{43} — Song File Parameter Get

<Item>	<Data1>…	<Reply>…	Description of Item
&01 {1}	Getting General Options		
	NA	BYTE	Get Group ID
	Getting Parameters		
&10 {16}	*NA*	WORD	Get From Bar
&11 {17}	*NA*	WORD	Get To Bar
&12 {18}	*NA*	BYTE	Get Tempo Mode <Reply> = (0=FILE, 1=MANUAL, 2=MULTI)
&13 {19}	*NA*	WORD	Get Manual Tempo <Reply> = (tempo×10) bpm

Table 43: Control Items for Section &2B{43} — Song File Parameter Get

<Item>	<Data1>…	<Reply>…	Description of Item
&18{24}	*NA*	BYTE(0–2)	Get MIDI output port <Reply> = (0=NONE, 1=MIDI A, 2=MIDI B)
	Getting Status Information		
&20{32}	*NA*	BYTE	Get (Time Signature) Beat Value
&21{33}	*NA*	BYTE	Get (Time Signature) Beats-per-Bar
&22{34}	*NA*	BYTE	Get Current Beat
&23{35}	*NA*	WORD	Get Current Bar
&24{36}	*NA*	WORD	Get Current Tempo <Reply> = (tempo×10)bpm

Item List for Front Panel Control section [&2C{44}]

To facilitate a *Remote Control* facility, this section allows the controls of the front panel to be manipulated directly via SysEx. The control codes which must be sent with some of these messages, are detailed in Table 45 – Table 47.

> Note that if a *"Key Hold"* message is sent, this must eventually be followed by a *"Key Release"* message, although there can be a delay between sending the release message. This is useful, for example, if the PLAY is pressed for a few seconds to hear a sample, before being released.

Table 44: Control Items for Section &2C{44} — Front Panel Control

<Item>	<Data1>	<Data2>…	Description of Item[a]
	Control Activation		
&01{1}	BYTE[b]	*NA*	"Key Hold": hold down panel key: <Data1> = keycode
&02{2}	BYTE[b]	*NA*	"Key Release": release panel key: <Data1> = keycode
&03{3}	SBYTE	*NA*	Move the data wheel forwards or backwards a certain amount
&04{4}	BYTE(0–7)	WORD(0–1023)	Set the value of a Q-link control. <Data1> = Qlink control, <Data2> = value
	ASCII keyboard		
&10{16}	WORD[c]	WORD[d]	ASCII Key Press: <Data1> = ASCII value, <Data2> = flags
&11{17}	WORD[c]	WORD[d]	ASCII Key Release: <Data1> = ASCII value, <Data2> = flags
	Mouse Operations[e]		
&20{32}	WORD	WORD	Mouse Click at Screen Coordinate (Select Parameter) <Data1> = x-coord, <Data2> = y-coord.
&21{33}	WORD	WORD	Mouse Double Click at Screen Coordinate (open Parameter Window) <Data1> = x-coord, <Data2> = y-coord.

[a] These functions cause the data to be queued within the sampler, it does not wait until the data has been processed. Thus the DONE confirmation message simply confirms that the data has been queued, not processed. If the data is not queued successfully, an ERROR will be returned.

[b] Only those keycodes defined in Table 45 should be used. Use of other values may lead to undefined behaviour.

[c] Standard ASCII codes can be used, as well as the virtual codes defined in Table 46.

[d] The flags contains a combination of the values detailed in Table 47. These indicate which combination of modifier keys are currently being held down.

[e] *Mouse* operations are used by remote displays, where a mouse and cursor system could be used.

Table 45: Keycodes for Front-Panel Control

Key	<Keycode>	Key	<Keycode>	Key	<Keycode>	Key	<Keycode>
Mode Keys		Function Keys		Other Keys		Cursor Keys	
MULTI	&17 {23}	F1	&01 {1}	WINDOW	&07 {7}	CURSOR ←	&0D {13}
FX	&13 {19}	F2	&02 {2}	Q-LINK SETUP	&08 {8}	CURSOR →	&0E {14}
EDIT SAMPLE	&16 {22}	F3	&03 {3}	CLIPBOARD	&09 {9}	CURSOR ↑	&0F {15}
EDIT PROGRAM	&12 {18}	F4	&04 {4}	PLAY	&0A {10}	CURSOR ↓	&0C {12}
RECORD	&15 {21}	F5	&05 {5}				
UTILITIES	&11 {17}	F6	&06 {6}	SHIFT	&40 {64}	+	&19 {25}
SAVE	&14 {20}					−	&1A {26}
LOAD	&10 {16}						

Table 46: Virtual ASCII codes for Front-Panel Control

Key	<Keycode>	Key	<Keycode>	Key	<Keycode>	Key	<Keycode>
Cursor Keys		Edit Keys		Function Keys		Control Keys	
CURSOR ←	&101 {257}	HOME	&120 {288}	F1	&201 {513}	SHIFT	&301 {769}
CURSOR →	&102 {258}	END	&121 {289}	F2	&202 {514}	CONTROL	&302 {770}
CURSOR ↑	&103 {259}	PAGE UP	&122 {290}	F3	&203 {515}	ALT	&303 {771}
CURSOR ↓	&104 {260}	PAGE DOWN	&123 {291}	F4	&204 {516}	MENU	&304 {772}
				F5	&205 {517}	PAUSE	&305 {773}
		INSERT	&110 {272}	F6	&206 {518}	GUI	&306 {774}
		DELETE	&7F {127}	F7	&207 {519}	ESCAPE	&1B {27}
				F8	&208 {520}		

Table 47: ASCII keyboard modifier flags for Front-Panel Control

Modifier Key	<Flag Value>
SHIFT	1
CONTROL	2
ALT	4
GUI	8

Item List for Recording section [&30{48}]

When information about the recording setup is requested with a *Get* message, the data is returned in a "REPLY" confirmation message (see *Confirmation Messages* on page 5). The format of the data content of these messages is summarised in Table 49 and Table 51.

Table 48: Control Items for Section &30{48} — Recording Main

<Item>	<Data1>	<Data2>…	Description of Item
	General Information		
&01 {1}	*NA*	*NA*	Get Record Status
&02 {2}	*NA*	*NA*	Get Record Progress
&03 {3}	*NA*	*NA*	Get Maximum Record Time
	Controlling the Recording Process		
&10 {16}	*NA*	*NA*	Arm Recording
&11 {17}	*NA*	*NA*	Start Recording
&12 {18}	*NA*	*NA*	Stop Recording
&13 {19}	*NA*	*NA*	Cancel Recording
	Post-Recording Operations		
&20 {32}	*NA*	*NA*	Play Recorded Sample Start
&21 {33}	*NA*	*NA*	Play Recorded Sample Stop
&22 {34}	*NA*	*NA*	Keep Recorded Sample. Sample with name assigned above, is added to the list of available samples.
&23 {35}	*NA*	*NA*	Delete Recorded Sample

Table 49: Format of "REPLY" confirmation messages for Section &30{48} — Recording Main

<Item> requested	<Reply>…	Description of Data Returned
&01 {1}	BYTE	Get Record Status
&02 {2}	DWORD	Get Record Progress
&03 {3}	DWORD	Get Maximum Record Time

Table 50: Control Items for Section &32{4A} — Recording Parameter Set

<Item>	<Data1>	<Data2>…	Description of Item
	Setting General Options		
&01 {1}	BYTE(0–6)	*NA*	Set Input <Data1> = (0=ANALOGUE, 1=DIGITAL, 2=MAIN OUT, 3=ADAT1/2, 4=ADAT3/4, 5=ADAT5/6, 6=ADAT7/8.
&02 {2}	BYTE(0–3)	*NA*	Set Record Mode <Data1> = (0=STEREO, 1=MONO L, 2=MONO R, 3=L/R MIX)
&03 {3}	BYTE(0, 1)	*NA*	Set Record Monitor <Data1> = (0=OFF, 1=ON)

Table 50: Control Items for Section &32{4A} — Recording Parameter Set

<Item>	<Data1>	<Data2>…	Description of Item
&04 {4}	DWORD	*NA*	Set Record Time <Data1> = time in seconds. If <Data1> = 0, MANUAL mode is enabled.
&05 {5}	BYTE	*NA*	Set Original Pitch
&06 {6}	SBYTE(−63–0)	*NA*	Set Threshold <Data1> = threshold in dB
&07 {7}	BYTE(0–2)	*NA*	Set Trigger Source <Data1> = (0=OFF, 1=AUDIO, 2=MIDI)
&08 {8}	BYTE(0, 1)	*NA*	Set Bit Depth <Data1> = (0=16-bit, 1=24-bit)
&09 {9}	WORD	*NA*	Set Pre-recording Time <Data1> = time in ms
&0A {10}	BYTE	*NA*	Set Recording Detination <Data1> = (0=RAM, 1=DISK)
&10 {16}	STRING	*NA*	Set Record Name
&11 {17}	STRING	*NA*	Set Record Name Seed
&12 {18}	BYTE(0, 1)	*NA*	Set Auto-Record Mode <Data1> = (0=OFF, 1=ON)
&13 {19}	BYTE(0, 1)	*NA*	Set Auto-Normalise Mode <Data1> = (0=OFF, 1=ON)

Table 51: Control Items for Section &33{4B} — Recording Parameter Get

<Item>	<Data1>…	<Reply>…	Description of Item
	Getting General Options		
&01 {1}	*NA*	BYTE(0–6)	Get Input <Reply> = (0=ANALOGUE, 1=DIGITAL, 2=MAIN OUT, 3=ADAT1/2, 4=ADAT3/4, 5=ADAT5/6, 6=ADAT7/8.
&02 {2}	*NA*	BYTE(0–3)	Get Record Mode <Reply> = (0=STEREO, 1=MONO L, 2=MONO R, 3=L/R MIX)
&03 {3}	*NA*	BYTE(0, 1)	Get Record Monitor <Reply> = (0=OFF, 1=ON)
&04 {4}	*NA*	DWORD	Get Record Time <Reply> = time in seconds. If <Reply> = 0, MANUAL mode is enabled.
&05 {5}	*NA*	BYTE	Get Original Pitch
&06 {6}	*NA*	SBYTE(−63–0)	Get Threshold <Reply> = threshold in dB
&07 {7}	*NA*	BYTE(0–2)	Get Trigger Source <Reply> = (0=OFF, 1=AUDIO, 2=MIDI)
&08 {8}	*NA*	BYTE(0, 1)	Get Bit Depth <Reply> = (0=16-bit, 1=24-bit)
&09 {9}	*NA*	WORD	Get Pre-recording Time <Reply> = time in ms
&0A {10}	*NA*	BYTE	Get Recording Detination <Reply> = (0=RAM, 1=DISK)
&10 {16}	*NA*	STRING	Get Record Name
&11 {17}	*NA*	STRING	Get Record Name Seed
&12 {18}	*NA*	BYTE(0, 1)	Get Auto-Record Mode <Reply> = (0=OFF, 1=ON)
&13 {19}	*NA*	BYTE(0, 1)	Get Auto-Normalise Mode <Reply> = (0=OFF, 1=ON)

Alternative Operations Sections [&60{96} – &64{100}]

When adjusting the parameters of a multi, program or sample, it is necessary to select which item is to be *current* before any operations can be performed. Whilst this provides a convenient means of operation, it is sometimes better if the item can be specified by handle for each operation — a typical example would be to obtain the lengths of each of the samples in memory, where it would be tedious to select the current sample before every operation.

Additional, alternative, Sections (&60 {96} – &64 {100}) simply provide a *back-door* way to access the usual Sections where the handle of the requested item is always represented in the first 4 bytes of the SysEx message. The subsequent bytes are exactly the same as those used for the usual Sections.

These messages also facilitate user-defined *blocking* of parameter requests into one message. For example, rather than using 13 different messages to determine the setup of an LFO in a program, one message containing all of these requests can be generated and the resultant data returned within a single reply message. Obviously, this can substantially reduce the communications overhead.

To use this feature to combine several calls to the same section into one message, one sends blocks of data of the form: `(<length=N> <Item> <Data`$_1$`> <…> <Data`$_{N-1}$`>)`. Where `<length=N>` is the number of raw bytes to follow and `<Item>` represents the item part of the protocol. Note that the number of bytes must be set correctly as it is used by the device to determine where the next block of parameter data begins.

Following a succesful Get operation, a single REPLY confirmation message will be generated containing all of the requested data; the data being a concatenation of the reply data found when a normal message is used. If an error results, 2 outcomes may result: (i) an ERROR confirmation message will be returned but without a REPLY confirmation message (in this case, the error number will explain the cause of the error) or (ii) the REPLY confirmation message will be incomplete, and will only contain data up to the point the error occurred.

A summary of the format of the required messages is shown in Table 52. Note that all of these functions change the *currently selected* item.

Table 52: Message Format for Alternative (By-Handle) Operations

<Section>	Section Offset[a]	<Handle>	<Index>	<*N*bytes>	<Item>	<Data…>	Description	Base Section
&60 {96}	BYTE(0–3)	DWORD	*NA*	BYTE	*see* Table 30–Table 33		Sample Manipulation by handle <Handle> = Sample Handle	&1C
&61 {97}[b]	BYTE(0–7)	DWORD	BYTE	BYTE	*see* Table 12–Table 17		Keygroup and Keygroup Zone Manipulation <Handle> = Program Handle <Index> = Keygroup Index	&0C
&62 {98}	BYTE(0–3)	DWORD	*NA*	BYTE	*see* Table 18–Table 21		Program Manipulation by handle <Handle> = Program Handle	&14
&63 {99}	BYTE(0–3)	DWORD	*NA*	BYTE	*see* Table 26–Table 29		Multi Manipulation by handle <Handle> = Multi Handle	&18
&64 {100}	BYTE(0–3)	DWORD	*NA*	BYTE	*see* Table 36–Table 39		Multi FX Manipulation by handle <Handle> = Multi Handle	&24
&65 {101}	BYTE(0–3)	DWORD	*NA*	BYTE	*see* Table 40–Table 43		Song File Manipulation by handle <Handle> = Song File Handle	&28

[a] The value of Section Offset is added to the destination section. For example, calling *Sample Manipulation by Handle* will cause <Item><Data…> to be redirected to Section &1C (*Sample Main*). To access Section &1E (*Sample Parameter Set*) Section Offset must be set to 2.

[b] Note that Keygroup Zone=0 (all Zones) must not be used with this message to *Get* data as it may yield unpredictable results.

Program Automation Sections [&68{104} – &69{105}]

While the SysEx protocol provides complete control of Multis and Programs, it requires that the item to be edited is *current,* or that the handle for the item is known. In many cases, this is not desirable, or even possible. In particular, it is difficult to configure a sequencer to control a Program's parameters in real-time, when playing back in MULTI mode.

To get around these problems, and allow Programs to be fully automated by a sequencer, additional SysEx has been provided. This works in a way similar to the Alternative Operations Section (page 50), but uses the current Multi's part number instead of the Program's handle. Thus, only programs assigned to the *current* Multi can be modified by this Section.

A summary of the format of the required messages is shown in Table 53. Note that none of these functions change the *currently selected* item, unlike the Alternative Operations Section.

Table 53: Message Format for Program Automation

<Section>	Section Offset[a]	<Part>	<KG>	<*N*bytes>	<Item>	<Data…>	Description	Base Section
&68 {104}[b]	BYTE (0–7)	BYTE	BYTE	BYTE	*see* Table 12–Table 17		Keygroup and Keygroup Zone Automation <Index> = Keygroup Index	&0C
&69 {105}	BYTE (0–3)	BYTE	*NA*	BYTE	*see* Table 18–Table 21		Program Automation	&14

a. The value of Section Offset is added to the destination section. For example, calling *Program Automation* will cause <Item><Data…> to be redirected to Section &14 (*Program Main*). To access Section &16 (*Program Parameter Set*) Section Offset must be set to 2.

b. Note that Keygroup Zone=0 (all Zones) must not be used with this message to *Get* data as it may yield unpredictable results.

To demonstrate how this protocol is used, the following example shows how to mute all Keygroup Zones within a Keygroup of the first Program in a Multi. The section which does this is "Keygroup Zone Set Parameter &0E {14}" (Table 12, page 16). Which means that we must use `<Section> = &68`, and `<Section Offset> = &02` (since the base section is &0C). We want to change the program in the first part, so `<Part> = &00`, since this is zero-based. `<KG>` is the zero-based index of the the keygroup we wish to mute.

The `<Item>`, `<Data…>` part is simply the same as in Table 12. We want to mute all the zones, so `<Item> = &0C`, `<Data1> = &00` (all zones), and `<Data2> = &01` (mute ON). Since this is 3 bytes, `<Nbytes> = &03`.

Combining this into a complete message, as previously discussed (page 4), we have:

```
<&F0> <&47> <&5F> <&00> <&68> <&02> <&00> <KG> <&03> <&0C> <&00> <&01> <&F7>
```

Note: If the same Program is assigned to multiple parts in a Multi, changes to that Program will affect *all* the parts which use it, not just the part specified in the SysEx message.

(This Page has been left intentionally blank.)