S5000V2

S6000V2

AKAI S5000/S6000
MIDI System Exclusive Protocol Specification

(OS Version 2·10)

(This Page has been left intentionally blank.)

Table of Contents

List of Tables

Introduction

This document details the specification for the MIDI System Exclusive protocol for the Akai S5000 and S6000 samplers. 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 S5000 and S6000 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 ($_{RT}$) 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 1·20. This section highlights the features added, or changes made, for different versions of the OS.

Version 1·20

This was intended to provide some basic real-time control of a Multi's parameters. For example, a mixermap can be created in a sequencer and used to adjust the level and pan position of a part while a sequence is playing.

Version 1·30

The use of the `<User Ref>` field has been extended to allow either 1, 2, 3 or 4 bytes to be used. To facilitate this, the top 2 bits (bits 5 and 6) of `<Device ID>` are now used to hold a count of the number of user-refs in the message. A consequence of this is that the `<Device ID>` is now limited to the range: 0–31.

The Confirmation Message format was changed to include the `<Section>` and `<Item>` portions of the incoming message, making it easier to tie up the responses with the sent messages. Checksums are also appended to confirmation messages (and replies) if the checksum facility is enabled. The SysEx specification was also significantly extended to include additional Multi, Program and Sample functionality.

Version 2·00

The workings of the SysEx decoder were modified to better separate the operations on separate MIDI channels, thus preventing unwanted interference from different applications. For example, MIDI port A may be driven by a multi editor mixermap, whereas MIDI port B could have a more general multi/program editor controlling it. In such situations, it is possible for certain operations (such as selecting the current multi) on port B to adversely affect those on port A and *vice-versa*. In the new scheme, it is possible to isolate both ports such that this situation does not arise — this is done by disabling the *synchronisation* option. This new architecture has also allowed the introduction of new functions to allow the modification of all multi, program and sample parameters by item index, rather than having to select the *current* item each time.

The SysEx specification was extended to provide additional information for the samples in memory, and to allow the manipulation of basic sample parameters. Further extensions include the editing of Multi Effects, and disk navigation.

Version 2·10

The SysEx decoder was further modified to provide improved resistance against SysEx data causing musical MIDI data (and hence playing music) to be interrupted. If only one SysEx message (per MIDI port) is sent at a time, using the DONE/REPLY/ERROR messages to synchronise transmission, it is now guaranteed that musical data will not be disturbed.

Further additions to the protocol have also been made. In particular a *Still Alive* message has been added to the SysEx Configuration Section, enabling the controller to know that the sampler is still busy processing a lengthy SysEx operation. A facility to lock and unlock the sampler's front-panel has also been added, which is useful if you want to ensure that the only control is via SysEx (for example, if the sampler is being remotely controlled from a distance). In addition, the functionality of the Disk Tools Section has been changed and enhanced.

Other minor corrections to this document, and to the SysEx protocol have been made.

[†.] 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.

To enable several S5000/6000s to coexist on the same MIDI chain and be configured independently via SysEx, a user-selectable DeviceID is implemented. Thus a MIDI chain could be setup as shown in Figure 1.

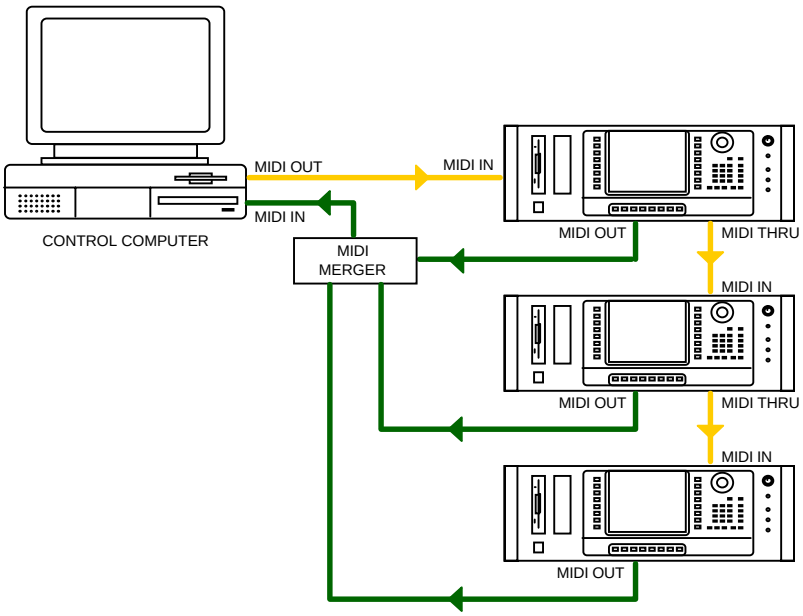

Figure 1: Illustration of a MIDI chain

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 S5000/6000 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> <S5000/6000 ID> <User-selectable Device ID> <User-Refs..> …
```

Where the values of the bytes are:

```
<&F0> <&47> <&5E> <0..&7F> <0..&F7> …
{<240> <71> <94> <0..127> <0..247> …}
```

AKAI ID <&47{71}> and S5000/6000 ID <&5E{94}>

The AKAI ID and the S5000/6000 ID ensure that only AKAI S5000/6000 samplers respond to these messages.

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

The user-selectable DeviceID allows more than one AKAI S5000/6000 sampler to coexist on the same MIDI bus, but be configured independently via SysEx. The default DeviceID is zero.

Since OS Version 1·30, the DeviceID has been limited to the range: 0–31. This is because the top 2 bits (bits 5 and 6) are now 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; one byte 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. (Note that this implementation is backwards compatible with previous OS Versions provided the DeviceID used is less than 31.)

Table 1: Number of User-Refs Being Sent

Device ID bit 6	Device ID bit 5	Num User-Refs
0	0	1
0	1	2
1	0	3
1	1	4

Note: that if the S5000/6000 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* S5000/6000s 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.

Since OS Version 1·30, either 1, 2, 3 or 4 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 1: 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 S5000/6000 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 — 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> <&5E> <&05> <&10> <&0C> <&1B> <&35> <&6D> <&F7>
   {<240> <71> <94> <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> <&5E> <&05> <&10> <&0C> <&1B> <&35> <&6D> <&59> <&F7>
   {<240> <71> <94> <5> <16> <12> <27> <53> <109> <89> <247>}
```

A useful tip — to turn off Checksums on all S5000/6000s, regardless of DeviceID, send:

```
<&F0> <&47> <&5E> <&00> <&00> <&00> <&04> <&00> <&04> <&F7>
      {<240> <71> <94> <0> <0> <0> <4> <0> <4> <247>}
```

A complete message

The format of a complete control message is as follows:

```
<&F0> <&47> <&5E> <0..&7F> <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> <S5000/6000 ID> <Device ID> <User-Ref> <…> <Reply ID> <Section> <Item> <Data1> …
                                <DataN> <checksum> <&F7>
```

The first 7 bytes (or more if more than one `<User-Ref>` is 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 5).

Table 2: SysEx Confirmation Messages

Reply ID	Data1	Data2	Meaning
&4F{79} ‘O’	N/A	N/A’	“OK” Valid SysEx has been received and is being processed.
&44{68} ‘D’	N/A	N/A	“DONE” SysEx instruction has been completed successfully.
&52{82} ‘R’	Reply 1	Reply 2…	“REPLY” A variable number of bytes is returned as a reply (data returned depends on SysEx message sent)
&45{69} ‘E’	<MSB 0–127>	<LSB 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 = &5E{94}, 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 that 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
&00{0}	0	0	Not Supported — the requested command is not supported.
&01{1}	0	1	Invalid Message Format — insufficient data supplied.
&02{2}	0	2	Parameter out of Range.
&03{3}	0	3	Unknown Error — could not complete command for some reason.
&04{4}	0	4	Requested program/multi/sample/etc., could not be found.
&05{5}	0	5	"new" element could not be created.
&06{6}	0	6	Deletion of requested item could not be completed.
&81{129}	1	1	Checksum Invalid.
&101{257}	2	1	Disk Error — Selected Disk is Invalid.
&102{258}	2	2	Disk Error — Error During Load.
&103{259}	2	3	Disk Error — Item Not Found.
&104{260}	2	4	Disk Error — Unable to Create.
&105{261}	2	5	Disk Error — Folder Not Empty.
&106{262}	2	6	Disk Error — Unable to Delete.
&107{263}	2	7	Disk Error — Unknown Error.
&108{264}	2	8	Disk Error — Error During Save.
&109{265}	2	9	Disk Error — Insufficient disk space.
&10A{266}	2	10	Disk Error — Media is write-protected.
&10B{267}	2	11	Disk Error — Cannot save because name is not unique.

Table 3: Error Numbers Returned

Error Number	MSB	LSB	Description of Error
&10C{268}	2	12	Disk Error — Invalid Disk Handle.
&10D{269}	2	13	Disk Error — Disk is Empty.
&10E{270}	2	14	Disk Error — Disk Operation was Aborted.
&10F{271}	2	15	Disk Error — Failed on Open.
&110{272}	2	16	Disk Error — Read Error.
&111{273}	2	17	Disk Error — Disk Not Ready.
&112{274}	2	18	Disk Error — SCSI Error.
&181{385}	3	1	Requested Keygroup Does not exist in Current Program

Control Messages

The functions of the S5000/S6000 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> <S5000/6000 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
&02{2}	System Setup
&04{4}	MIDI Configuration
&06{6}	Keygroup Zone Manipulation
&08{8}	Keygroup Manipulation
&0A{10}	Program Manipulation
&0C{12}	Multi Manipulation
&0E{14}	Sample Tools
&10{16}	Disk Tools
&12{18}	Multi FX Control
&14{20}	Scenelist Manipulation
&16{22}	MIDI song file tools
&20{32}	Front Panel Control
&2A{42}	Alt. Program Manipulation[a]
&2C{44}	Alt. Multi Manipulation[a]
&2E{46}	Alt. Sample Tools[a]
&32{50}	Alt Multi FX Control[a]

Table 4: Description of <Section> Parameter

<Section>	Description of Section
&38{56}	Alt. Keygroup Zone *Blocked*[b]
&3A{58}	Alt. Keygroup *Blocked*[b]
&3C{60}	Alt. Program *Blocked*[b]
&3E{62}	Alt. Multi *Blocked*[b]
&44{68}	<Reserved>
&45{69}	<Reserved>
&4F{79}	<Reserved>
&52{82}	<Reserved>

a. These represent an alternative means of control, where the operation is performed on the item (multi, program or sample) specified by index, rather than on the currently selected item.
b. These represent another alternative means of control, where user-defined *blocks* of parameters can be requested in one message. This allows the overhead of communication to be reduced.

Format of Message Data

The S5000/6000 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.

The format of the data which must be sent, and how this appears in the Tables, is now explained.

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, <byte1>…<byteN> means that there may be additional bytes between byte1 and byteN. The number of additional bytes depends on the both format and the content of the data being sent.

Discrete Byte Values

Discrete Bytes, which only have a few possible values are shown as "<value1>, <value2>, …, <valueN>". For example if only the boolean values "0" and "1" are allowed, this will be shown as "0, 1". Only one of the possible values can be transmitted in any one message.

Range of Byte Values

Where a range of data values are permitted, this will be shown as "<value1> – <valueN>". Only values between <value1> and <valueN> are permitted. For example, the range "0–127" means that any value between 0 and 127 may be transmitted including 0 and 127. Only one of the possible values can be transmitted in any one message.

Compound Word

Where a large number (*i.e.*, > 127) has to be transmitted, it must be split into 2 data bytes, and is shown by: "0–127(MSB)…0–127(LSB)". This means that 2 data bytes must be transmitted, with the most-significant byte (MSB) transmitted first, followed by the least-significant byte (LSB). The value of the compound word = LSB + 128×MSB. For example, if MSB = 5, and LSB = 65, the compound word = 65 + 128×5 = 705.

Compound Double Word

A double word is similar to a compound word, except there are 4 bytes used to represent the data. This is shown as: "0–127(MSB)…0–127(SB2)…0–127(SB1)…0–127(LSB)", where 4 data bytes are transmitted with the most-significant byte (MSB) transmitted first. The value which these bytes represent is calculated thus: double word = $LSB + 128 \times SB1 + 128^2 \times SB2 + 128^3 \times MSB$.

Compound Quad Word

A quad word is similar to a double word, except there are 8 bytes used to represent the data. This is shown as: "QWORD<8 bytes>", where 8 data bytes are transmitted with the most-significant byte (MSB) transmitted first. The value which these bytes represent is calculated thus: quad word = $BYTE8 + 128 \times BYTE7 + 128^2 \times BYTE6 + 128^3 \times BYTE5 + 128^4 \times BYTE4 + 128^5 \times BYTE3 + 128^6 \times BYTE2 + 128^7 \times BYTE1$.

Signed Values

Where signed values are required, the data value (byte, compound word, or compound double word) is preceded by a single byte representing the sign of the value, where 0 = positive, 1 = negative.

Null-terminated Character String

It is also necessary to send a string of characters, or more simply *text*, for many of the operations. This is achieved by sending the ASCII codes of the characters one after the other, and terminating the string with a zero. The zero is necessary so that the sampler can determine where the string has finished and it must always be sent. A null-terminated string is shown as: "char1…0".

> Note: Data values 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!

The format of data received via REPLY confirmation messages is the same as that for data sent (just discussed).

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

<Item>	<Data1>	<Data2>	Description of Item
&00{0}$_{RT}$	N/A	N/A	"Query" — use with user-selectable DeviceID=0 to get an "OK!" and a "DONE" reply with DeviceID returned.
&01{1}$_{RT}$	0, 1	N/A	Enable/Disable received message notification ("OK!"): 0=OFF, 1=ON
&03{3}$_{RT}$[a]	0, 1	N/A	Enable/Disable synchronisation between the currently selected samples/programs/multis and those displayed on the LCD. 0=OFF, 1=ON
&04{4}$_{RT}$	0, 1	N/A	Enable/Disable checksum verification: 0=OFF, 1=ON
&05{5}$_{RT}$	0, 1	N/A	Enable/Disable automatic screen updating when a SysEx message is processed: 0=OFF, 1=ON
&06{6}$_{RT}$	0–127	0–127, <Data3>=0–127, <Data4>=0–127	*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}$_{RT}$[b]	0, 1	N/A	Enable/Disable "*Still Alive"* monitor: 0=OFF, 1=ON

a. 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.

b. 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.

Item List for System Setup section [&02{2}]

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

Table 6: Control Items for Section &02{2} — System Setup

<Item>	<Data1>	<Data2>	Description of Item
&00{0}	N/A	N/A	Get Operating System Software Version.
&01{1}[a]	N/A	N/A	Get the Sub-Version of the Operating System
&02{2}$_{RT}$	char1	…0	Set sampler Name. Must end in a zero.
&03{3}$_{RT}$[b]	N/A	N/A	Get sampler Name. The name is returned as a character string in a REPLY message.
&04{4}$_{RT}$	N/A	N/A	Get Sampler Model: REPLY message: 0=S5000, 1=S6000.
&05{5}	N/A	N/A	Get Clock Time & Date

Table 6: Control Items for Section &02{2} — System Setup

<Item>	<Data1>	<Data2>	Description of Item
&06{6}	0–16 (MSB year)	0–127 (LSB year) <Data3>=1–12 … <Data8>=0–59	Set Clock Time & Date: <Data1>, <Data2> = Year(1980–2079), <Data3> = Month(1–12), <Data4> = DayOfMonth(1–31), <Data5> = DayOfWeek(1–7, 1=Sunday), <Data6> = Hours(0–23), <Data7> = Minutes(0–59), <Data8> = Seconds(0–59).
&10{16}	0, 1, 2	N/A	Set Play Mode: <Data1> = Play Mode to change to: 0=Multi, 1=Program; 2=Sample; 3=Muted.
&11{17}	0, 1	N/A	Set Front panel lock-out state. 0=normal; 1=locked.
&20{32}	N/A	N/A	Get Play Mode currently set.
&21{33}	N/A	N/A	Get Front panel lock-out state.
&30{48}	N/A	N/A	Get the percentage free Wave memory.
&31{49}	N/A	N/A	Get the percentage free MPKS (multis, programs, keygroups and samples) memory.
&32{50}	N/A	N/A	Clear Sampler Memory: all programs/multis/samples are deleted.
&33{51}	N/A	N/A	Get the Total number of bytes of Wave memory.
&34{52}	N/A	N/A	Get the number of bytes of free Wave memory.

a. This is for future expansion. Currently, the sub-version will always be returned as zero.
b. Unless this is changed by the user, the default value will be returned: either “AKAI S5000” or “AKAI S6000”. This is an alternative way of identifying samplers.

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 7.

Table 7: Format of “REPLY” confirmation messages for Section &02{2} — System Setup

<Item> requested	<Data1>	<Data2>	Description of Data Returned
&00{0}	0–127	0–127	Operating System Version: <Data1>=major version number, <Data2>=minor version number
&01{1}	0–127	N/A	Operating System Sub-Version. <Data1>=sub-version
&03{3}	char1	…0	Character string representing the sampler’s name (ends in a zero).
&04{4}	0, 1	N/A	Sampler Model. 0=S5000; 1=S6000.
&05{5}	0–16 (MSB year)	0–127 (LSB year) <Data3>=1–12 … <Data8>=0–59	Get Clock Time & Date: <Data1>, <Data2> = Year(1980–2079), <Data3> = Month(1–12), <Data4> = DayOfMonth(1–31), <Data5> = DayOfWeek(1–7, 1=Sunday), <Data6> = Hours(0–23), <Data7> = Minutes(0–59), <Data8> = Seconds(0–59).
&20{32}	0, 1, 2	N/A	Returns the current Play Mode: 0=Multi, 1=Program; 2=Sample; 3=Muted
&21{33}	0, 1	N/A	Returns the front-panel lock-out state: 0=normal; 1=locked.
&30{48}	0–100	N/A	Returns the % of Wave memory free (0–100%).
&31{49}	0–100	N/A	Returns the % of MPKS memory free (0–100%).
&33{51}	0–127(MSB)	…0–127(LSB)	A Compound Double Word (see page 9) representing the total bytes of wave memory installed.
&34{52}	0–127(MSB)	…0–127(LSB)	A Compound Double Word (see page 9) representing the free bytes of wave memory.

Item List for MIDI Configuration section [&04{4}]

This section allows you to change MIDI setup. Note that these options are the same as those in the "MIDI SETUP" section of the "UTILITIES" page.

Table 8: Control Items for Section &04{4} — MIDI Configuration

<Item>	<Data1>	<Data2>	Description of Item
&01{1}$_{RT}$	0, 1	N/A	Program Change Enable: 0=OFF, 1=ON
&02{2}$_{RT}$	0, 1, 2	N/A	Multi Select: 0=OFF, 1=PROG CHANGE, 2=BANK
&03{3}$_{RT}$	0–31	N/A	Multi Select Channel: 1A=0, 2A=1, …, 16B=31
&04{4}$_{RT}$	0–127	N/A	External APM Controller
&05{5}$_{RT}$	0, 1	N/A	Aftertouch: 0=Channel, 1=Polyphonic
&06{6}$_{RT}$	0–3	0–31	Enable MIDI filter for Port A & B. (allow messages) <Data1> : 0=NoteOn, 1=Aft'tch, 2=Wheels, 3=Volume <Data2> : Channel: 1A=0, 2A=1, …, 16B=31
&07{7}$_{RT}$	0–3	0–31	Disable MIDI filter for Port A & B. (ignore messages) <Data1> : 0=NoteOn, 1=Aft'tch, 2=Wheels, 3=Volume <Data2> : Channel: 1A=0, 2A=1, …, 16B=31

Item List for the Keygroup Zones section [&06{6}]

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 9: Control Items for Section &06 {6} — Keygroup Zone

<Item>	<Data1> (Zone number)	<Data2>…	Description of Item
	Setting Zone Parameters		
&01{1}	0, 1–4	char1…0	Set Zone Sample <Data2…0> = name of sample to assign to zone.
&02{2}$_{RT}$	0, 1–4	0, 1 <Data3>0–100	Set Zone Level <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
&03{3}$_{RT}$	0, 1–4	14–114	Set Zone Pan/Balance <Data2> = Pan/Bal, where (14–114 = L50–R50); centre=&40{64}
&04{4}	0, 1–4	0–24	Set Zone Output <Data2> = output, where 0=MULTI, 1–8 =op1/2–op15/16, 9–24=op1–op16
&05{5}$_{RT}$	0, 1–4	0, 1 <Data3>0–100	Set Zone Filter <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
&06{6}$_{RT}$	0, 1–4	0, 1 <Data3>0–50	Set Zone Fine Tune <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
&07{7}$_{RT}$	0, 1–4	0, 1 <Data3>0–36	Set Zone Semitone Tune <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
&08{8}	0, 1–4	0, 1	Set Zone Keyboard Track <Data2> = (0=OFF, 1=ON)

Table 9: Control Items for Section &06 {6} — Keygroup Zone

<Item>	<Data1> (Zone number)	<Data2>…	Description of Item
&09{9}	0, 1–4	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
&0A{10}	0, 1–4	0, 1 <Data3>(MSB) <Data4>(LSB) (range ±9999)	Set Zone Velocity→Start <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value (MSB), <Data4> = absolute value (LSB), (abs. value = LSB+128×MSB)
&0B{11}	0, 1–4	0–127	Set Zone High Velocity <Data2> = velocity
&0C{12}	0, 1–4	0–127	Set Zone Low Velocity <Data2> = velocity
&0D{13}	0, 1–4	0, 1	Set Zone Mute <Data2> = (0=OFF, 1=ON)
&0E{14}	0, 1–4	0, 1	Set Zone Solo <Data2> = (0=OFF, 1=ON)
	Getting Zone Parameters		
&21{33}	0, 1–4	N/A	Get Zone Sample
&22{34}	0, 1–4	N/A	Get Zone Level
&23{35}	0, 1–4	N/A	Get Zone Pan/Balance
&24{36}	0, 1–4	N/A	Get Zone Output
&25{37}	0, 1–4	N/A	Get Zone Filter
&26{38}	0, 1–4	N/A	Get Zone Fine Tune
&27{39}	0, 1–4	N/A	Get Zone Semitone Tune
&28{40}	0, 1–4	N/A	Get Zone Keyboard Track
&29{41}	0, 1–4	N/A	Get Zone Playback
&2A{42}	0, 1–4	N/A	Get Zone Velocity→Start
&2B{43}	0, 1–4	N/A	Get Zone High Velocity
&2C{44}	0, 1–4	N/A	Get Zone Low Velocity
&2D{45}	0, 1–4	N/A	Get Zone Mute
&2E{46}	0, 1–4	N/A	Get Zone Solo

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 10.

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 10 where the set of data bytes is repeated four times (once for each Zone) in a single REPLY confirmation message.

Moreover, if ALL keygroups are selected (*i.e.*, Current Keygroup = 0), then multiple data will be returned—one data set for every keygroup in the current program, with keygroup 1 transmitted first. This also means that if the Zone is set to ALL (*i.e.*, = 0) and the Keygroup is set to ALL (*i.e.*, = 0), the requested information for every Zone in every Keygroup for the current program will be returned in a single REPLY confirmation message. In this case, the data is transmitted in the following sequence:

```
Keygroup 1: <Zone1> <Zone2> <Zone3> <Zone4>
Keygroup 2: <Zone1> <Zone2> <Zone3> <Zone4>
                      …
Keygroup n: <Zone1> <Zone2> <Zone3> <Zone4>
```

Table 10: Format of "REPLY" confirmation messages for Section &06{6} — Keygroup Zone

<Item> requested	<Data1>	<Data2>	Description of Data Returned
	Getting Zone Parameters		
&21{33}	char1	…0	Get Zone Sample <Data1…0> = name of sample assigned to zone. If no sample is assigned, a single byte (<Data1>=0) will be returned.
&22{34}	0, 1	0–100	Get Zone Level <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&23{35}	14–114	N/A	Get Zone Pan/Balance <Data1> = Pan/Bal, where (14–114 = L50–R50); centre=&40{64}
&24{36}	0–24	N/A	Get Zone Output <Data1> = output, where 0=MULTI, 1–8 =op1/2–op15/16, 9–24=op1–op16
&25{37}	0, 1	0–100	Get Zone Filter <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&26{38}	0, 1	0–50	Get Zone Fine Tune <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&27{39}	0, 1	0–36	Set Zone Semitone Tune <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&28{40}	0, 1	N/A	Get Zone Keyboard Track <Data1> = (0=OFF, 1=ON)
&29{41}	0–6	N/A	Get Zone Playback <Data1> = 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
&2A{42}	0, 1	<Data2>(MSB) <Data3>(LSB) (range ±9999)	Get Zone Velocity→Start <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value (MSB), <Data3> = absolute value (LSB), (abs. value = LSB+128×MSB)
&2B{43}	0–127	N/A	Get Zone High Velocity <Data1> = velocity
&2C{44}	0–127	N/A	Get Zone Low Velocity <Data1> = velocity
&2D{45}	0, 1	N/A	Get Zone Mute <Data1> = (0=OFF, 1=ON)
&2E{46}	0, 1	N/A	Get Zone Solo <Data1> = (0=OFF, 1=ON)

Item List for the Keygroup section [&08{8}]

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 *all* keygroups to be active; this is done by selecting keygroup 0. Note that, for convenience, Keygroup Zones are manipulated using functions in the Keygroup Zone section.

Table 11: Control Items for Section &08 {8} — Keygroup

<Item>	<Data1>	<Data2>…	Description of Item
&01{1}	0, 1–99	N/A	Select Keygroup to be *current* <Data1> = Keygroup number. If <Data1> = 0, then ALL Keygroups will be selected.
&02{2}	N/A	N/A	Get which Keygroup is currently selected.
	Setting General Options		
&04{4}	21–127	N/A	Set Low Note. <Data1> = note, where: 21–127 = A-1–G8
&05{5}	21–127	N/A	Set High Note. <Data1> = note, where: 21–127 = A-1–G8
&06{6}	0, 1–32	N/A	Set Mute Group <Data1> = (0=OFF, 1–32=value)
&07{7}	0, 1–4	N/A	Set FX override <Data1> = (0=OFF, 1=FX1, 2=FX2, 3=RV3, 4=RV4)
&08{8}	0–100	N/A	Set FX Send Level <Data1> = level
&09{9}	0, 1	N/A	Set Zone Crossfade <Data1> = (0=OFF, 1=ON)
	Getting General Options		
&0A{10}	N/A	N/A	Get Low Note
&0B{11}	N/A	N/A	Get High Note
&0C{12}	N/A	N/A	Get Mute Group
&0D{13}	N/A	N/A	Get FX override
&0E{14}	N/A	N/A	Get FX Send Level
&0F{15}	N/A	N/A	Get Zone Crossfade
	Set Keygroup Pitch/Amp		
&10{16}$_{RT}$	0, 1	0–36	Set Semitone Tune. <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&11{17}$_{RT}$	0, 1	0–50	Set Fine Tune. <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&12{18}$_{RT}$	0–10	N/A	Set Keygroup Level <Data1> = value in dB, where: 0=–30dB, 1=–24dB, 2=–18dB, 3=–12dB, 4=–6dB, 5=0dB 6=6dB, 7=12dB, 8=18dB, 9=24dB, 10=30dB.
&13{19}$_{RT}$	1, 2	0, 1 <Data3>0–100	Set Pitch Mod Value. <Data1> = Pitch Mod (1 or 2) <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
&14{20}$_{RT}$	1	0, 1 <Data3>0–100	Set Amp Mod Value. <Data1> = Amp Mod (1 only) <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
	Get Keygroup Pitch/Amp		
&18{24}	N/A	N/A	Get Semitone Tune.
&19{25}	N/A	N/A	Get Fine Tune.
&1A{26}	N/A	N/A	Get Keygroup Level.
&1B{27}	1, 2	N/A	Get Pitch Mod Value. <Data1> = Pitch Mod (1 or 2)

Table 11: Control Items for Section &08 {8} — Keygroup

<Item>	<Data1>	<Data2>…	Description of Item
&1C{28}	1	N/A	Get Amp Mod Value. <Data1> = Amp Mod (1 only)
	Set Filter		
&20{32}	0–25	N/A	Set Filter Mode <Data1> = mode, where: 0=2-POLE LP, 1=4-POLE LP, 2=2-POLE LP+, 3=2-POLE BP, 4=4-POLE BP, 5=2-POLE BP+, 6=1-POLE HP, 7=2-POLE HP, 8=1-POLE HP+, 9=LO<>HI, 10=LO<>BAND, 11=BAND<>HI, 12=NOTCH 1, 13=NOTCH 2, 14=NOTCH 3, 15=WIDE NOTCH, 16=BI-NOTCH, 17=PEAK 1, 18=PEAK 2, 19=PEAK 3, 20=WIDE PEAK, 21=BI-PEAK, 22=PHASER 1, 23=PHASER 2, 24=BI-PHASE, 25=VOWELISER
&21{33}$_{RT}$	0–100	N/A	Set Filter Cutoff Frequency, <Data1> = value.
&22{34}$_{RT}$	0–15	N/A	Set Filter Resonance, <Data1> = value.
&23{35}$_{RT}$	0, 1	0–36	Set Filter Keyboard Track, <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&24{36}$_{RT}$	1, 2, 3	0, 1 <Data3>0–100	Set Filter Mod Input Value. <Data1> = Mod Input (1, 2 or 3) <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
&25{37}$_{RT}$	0–5	N/A	Set Filter Attenuation. <Data1> = value, where: 0=0dB, 1=6dB, 2=12dB, 3=18dB, 4=24dB, 5=30dB.
	Get Filter		
&28{40}	N/A	N/A	Get Filter Mode
&29{41}	N/A	N/A	Get Filter Cutoff Frequency
&2A{42}	N/A	N/A	Get Filter Resonance
&2B{43}	N/A	N/A	Get Filter Keyboard Track
&2C{44}	1, 2, 3	N/A	Get Filter Mod Input Value. <Data1> = Mod Input (1, 2 or 3)
&2D{45}	N/A	N/A	Get Filter Attenuation.
	Set Filter Envelope		
&30{48}$_{RT}$	0–100	N/A	Set Filter Envelope Attack. <Data1> = value.
&31{49}	0, 1	0–100	Set Filter Envelope Velocity→Attack <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&32{50}$_{RT}$	0–100	N/A	Set Filter Envelope Decay <Data1> = value
&33{51}$_{RT}$	0–100	N/A	Set Filter Envelope Sustain <Data1> = value
&34{52}$_{RT}$	0–100	N/A	Set Filter Envelope Release <Data1> = value
&35{53}	0, 1	0–100	Set Filter Envelope On Velocity→Release <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&36{54}	0, 1	0–100	Set Filter Envelope Keyscale <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&37{55}$_{RT}$	0, 1	0–100	Set Filter Envelope Depth <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&38{56}	0, 1	0–100	Set Filter Envelope Off Velocity→Release <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
	Get Filter Envelope		
&40{64}	N/A	N/A	Get Filter Envelope Attack.
&41{65}	N/A	N/A	Get Filter Envelope Velocity→Attack
&42{66}	N/A	N/A	Get Filter Envelope Decay
&43{67}	N/A	N/A	Get Filter Envelope Sustain

Table 11: Control Items for Section &08 {8} — Keygroup

<Item>	<Data1>	<Data2>…	Description of Item
&44{68}	N/A	N/A	Get Filter Envelope Release
&45{69}	N/A	N/A	Get Filter Envelope On Velocity→Release
&46{70}	N/A	N/A	Get Filter Envelope Keyscale
&47{71}	N/A	N/A	Get Filter Envelope Depth
&48{72}	N/A	N/A	Get Filter Envelope Off Velocity→Release
	Set Amplitude Envelope		
&50{80}$_{RT}$	0–100	N/A	Set Amplitude Envelope Attack. <Data1> = value.
&51{81}	0, 1	0–100	Set Amplitude Envelope Velocity→Attack <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&52{82}$_{RT}$	0–100	N/A	Set Amplitude Envelope Decay <Data1> = value
&53{83}$_{RT}$	0–100	N/A	Set Amplitude Envelope Sustain <Data1> = value
&54{84}$_{RT}$	0–100	N/A	Set Amplitude Envelope Release <Data1> = value
&55{85}	0, 1	0–100	Set Amplitude Envelope On Velocity→Release <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&56{86}	0, 1	0–100	Set Amplitude Envelope Keyscale <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&57{87}	0, 1	0–100	Set Amplitude Envelope Off Velocity→Release <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
	Get Amplitude Envelope		
&58{88}	N/A	N/A	Get Amplitude Envelope Attack
&59{89}	N/A	N/A	Get Amplitude Envelope Velocity→Attack
&5A{90}	N/A	N/A	Get Amplitude Envelope Decay
&5B{91}	N/A	N/A	Get Amplitude Envelope Sustain
&5C{92}	N/A	N/A	Get Amplitude Envelope Release
&5D{93}	N/A	N/A	Get Amplitude Envelope On Velocity→Release
&5E{94}	N/A	N/A	Get Amplitude Envelope Keyscale
&5F{95}	N/A	N/A	Get Amplitude Envelope Off Velocity→Release
	Set Aux Envelope		
&60{96}$_{RT}$	1, 2, 3, 4	0–100	Set Aux Env. Rate <Data1> = Aux. Rate (1, 2, 3 or 4) <Data2> = Rate value.
&61{97}	1, 4	0, 1 <Data3>0–100	Set Aux Env. Velocity→Rate <Data1> = Aux. Rate (1 or 4) <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
&62{98}	0, 1	0–100	Set Aux Env. Keyboard→R2/R4 <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&63{99}$_{RT}$	1, 2, 3, 4	0–100	Set Aux Env. Level <Data1> = Aux. Level (1, 2, 3 or 4) <Data2> = Level value.
&64{100}	4	0, 1 <Data3>0–100	Set Aux Env. Velocity→Rate <Data1> = Aux. Rate (4 only) <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
	Get Aux Envelope		
&68{104}	1, 2, 3, 4	N/A	Get Aux Env. Rate <Data1> = Aux. Rate (1, 2, 3 or 4)
&69{105}	1, 4	N/A	Get Aux Env. Velocity→Rate <Data1> = Aux. Rate (1 or 4)

Table 11: Control Items for Section &08 {8} — Keygroup

<Item>	<Data1>	<Data2>…	Description of Item
&6A{106}	N/A	N/A	Get Aux Env. Keyboard→R2/R4
&6B{107}	1, 2, 3, 4	N/A	Get Aux Env. Level <Data1> = Aux. Level (1, 2, 3 or 4)
&6C{107}	4	N/A	Get Aux Env. Off Velocity→Rate <Data1> = Aux. Rate (4 only)

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 12. Note that if ALL keygroups are selected (*i.e.*, Current Keygroup = 0) then each of the following REPLY messages may contain additional data (one for every keygroup in the current program). The format of this extra data is the same as that shown in Table 12 where the set of data bytes is repeated once for each additional keygroup in a single REPLY confirmation message.

Table 12: Format of "REPLY" confirmation messages for Section &08{8} — Keygroup

<Item> requested	<Data1>	<Data2>	Description of Data Returned
&02{2}	0, 1–99	N/A	<Data1> = Keygroup number which is currently selected. If <Data1> = 0, then ALL Keygroups are selected.
	Getting General Options		
&0A{10}	21–127	N/A	Get Low Note. <Data1> = note, where: 21–127 = A-1–G8
&0B{11}	21–127	N/A	Get High Note. <Data1> = note, where: 21–127 = A-1–G8
&0C{12}	0, 1–32	N/A	Get Mute Group <Data1> = (0=OFF, 1–32=value)
&0D{13}	0, 1–4	N/A	Get FX override <Data1> = (0=OFF, 1=FX1, 2=FX2, 3=RV3, 4=RV4)
&0E{14}	0–100	N/A	Get FX Send Level <Data1> = level
&0F{15}	0, 1	N/A	Get Zone Crossfade <Data1> = (0=OFF, 1=ON)
	Get Keygroup Pitch/Amp		
&18{24}	0, 1	0–36	Get Semitone Tune. <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&19{25}	0, 1	0–50	Get Fine Tune. <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&1A{26}	0–10	N/A	Get Keygroup Level <Data1> = value in dB, where: 0=–30dB, 1=–24dB, 2=–18dB, 3=–12dB, 4=–6dB, 5=0dB 6=6dB, 7=12dB, 8=18dB, 9=24dB, 10=30dB.
&1B{27}	0, 1	0–100	Get Pitch Mod Value. <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&1C{28}	0, 1	0–100	Get Amp Mod Value. <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
	Get Filter		
&28{40}	0–25	N/A	Get Filter Mode <Data1> = mode, where: 0=2-POLE LP, 1=4-POLE LP, 2=2-POLE LP+, 3=2-POLE BP, 4=4-POLE BP, 5=2-POLE BP+, 6=1-POLE HP, 7=2-POLE HP, 8=1-POLE HP+, 9=LO<>HI, 10=LO<>BAND, 11=BAND<>HI, 12=NOTCH 1, 13=NOTCH 2, 14=NOTCH 3, 15=WIDE NOTCH, 16=BI-NOTCH, 17=PEAK 1, 18=PEAK 2, 19=PEAK 3, 20=WIDE PEAK, 21=BI-PEAK, 22=PHASER 1, 23=PHASER 2, 24=BI-PHASE, 25=VOWELISER
&29{41}	0–100	N/A	Get Filter Cutoff Frequency, <Data1> = value.

Table 12: Format of "REPLY" confirmation messages for Section &08{8} — Keygroup

<Item> requested	<Data1>	<Data2>	Description of Data Returned
&2A{42}	0–15	N/A	Get Filter Resonance, <Data1> = value.
&2B{43}	0, 1	0–36	Get Filter Keyboard Track, <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&2C{44}	0, 1	0–100	Get Filter Mod Input Value. <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&2D{45}	0–5	N/A	Get Filter Attenuation. <Data1> = value, where: 0=0dB, 1=6dB, 2=12dB, 3=18dB, 4=24dB, 5=30dB.
	Get Filter Envelope		
&40{64}	0–100	N/A	Get Filter Envelope Attack. <Data1> = value.
&41{65}	0, 1	0–100	Get Filter Envelope Velocity→Attack <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&42{66}	0–100	N/A	Get Filter Envelope Decay <Data1> = value
&43{67}	0–100	N/A	Get Filter Envelope Sustain <Data1> = value
&44{68}	0–100	N/A	Get Filter Envelope Release <Data1> = value
&45{69}	0, 1	0–100	Get Filter Envelope On Velocity→Release <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&46{70}	0, 1	0–100	Get Filter Envelope Keyscale <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&47{71}	0, 1	0–100	Get Filter Envelope Depth <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&48{72}	0, 1	0–100	Get Filter Envelope Off Velocity→Rate <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
	Get Amplitude Envelope		
&58{88}	0–100	N/A	Get Amplitude Envelope Attack. <Data1> = value.
&59{89}	0, 1	0–100	Get Amplitude Envelope Velocity→Attack <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&5A{90}	0–100	N/A	Get Amplitude Envelope Decay <Data1> = value
&5B{91}	0–100	N/A	Get Amplitude Envelope Sustain <Data1> = value
&5C{92}	0–100	N/A	Get Amplitude Envelope Release <Data1> = value
&5D{93}	0, 1	0–100	Get Amplitude Envelope On Velocity→Release <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&5E{94}	0, 1	0–100	Get Amplitude Envelope Keyscale <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&5F{95}	0, 1	0–100	Get Amplitude Envelope Off Velocity→Rate <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
	Get Aux Envelope		
&68{104}	0–100	N/A	Get Aux Env. Rate <Data1> = Rate value.
&69{105}	0, 1	0–100	Get Aux Env. Velocity→Rate <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&6A{106}	0, 1	0–100	Get Aux Env. Keyboard→R2/R4 <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&6B{107}	0–100	N/A	Get Aux Env. Level <Data1> = Level value.
&6C{107}	0, 1	0–100	Get Aux Env. Off Velocity→Rate <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.

Item List for Program section [&0A{10}]

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 2 ways to select a program: (i) Create a new program; and (ii) select an existing program in memory to be the *current* one. The first method can only be used if a program of the specified name does not already exist in memory. Note that the active program 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 "PROGRAM" mode.

The *index* used to select a program by index, or returned when the current program is requested, is zero-based, *i.e.*, the first program in memory has *index*=0.

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.

Table 13: Control Items for Section &0A {10} — Program

<Item>	<Data1>	<Data2>…	Description of Item
	Program Selection, Creation and Deletion		
&02{2}	char1	…0	Create Program: <Data1…0> = Name
&03{3}	1–99	char1…0	Create Program with keygroups. <Data1> = number of keygroups, <Data2…0> = Name
&05{5}	char1	…0	Select Program (by name) to be *current*: <Data1…0> = Name
&06{6}	0–127(MSB)	0–127(LSB)	Select Program (by index) to be *current:* <Data1> = MSB; <Data2> = LSB (index = LSB+128×MSB).
&07{7}	N/A	N/A	Delete ALL programs from memory
&08{8}	N/A	N/A	Delete the currently selected Program from memory
&09{9}	char1	…0	Rename currently selected Program: <Data1…0> = Name
&0A{10}	0, 1	0–127[a]	Set "Program Number". <Data1>: 0=OFF; 1=ON. <Data2>=program number (ony required if <Data1>=1).
&0B{11}	1–98	N/A	Add Keygroups to Program <Data1> = Number of Keygroups to add.
&0C{12}	0–98	N/A	Delete Keygroup from Program: <Data1> = Number of the keygroup to delete. (zero-based)
&0D{13}	0, 1	N/A	Set Keygroup Crossfade <Data1>: 0=OFF; 1=ON
	Getting General Program Information for the *Current* Program. (Note: The format of the data returned in the "REPLY" confirmation messages are detailed in Table 14.)		
&10{16}	N/A	N/A	Get Number of Programs in memory
&11{17}	N/A	N/A	Get Current Program's "Program Number"
&12{18}	N/A	N/A	Get Current Program Index (*i.e.*, its position in memory)
&13{19}	N/A	N/A	Get Current Program Name
&14{20}	N/A	N/A	Get Number of Keygroups in Current Program
&15{21}	N/A	N/A	Get Keygroup Crossfade
	Getting General Program Information for All the Programs in Memory. (Note: The format of the data returned in the "REPLY" confirmation messages are detailed in Table 14.)		
&18{24}	N/A	N/A	Get the "Program Numbers" of all the Programs in memory
&19{25}	N/A	N/A	Get the names of all of the Programs in memory
	Output — Set Values		

Table 13: Control Items for Section &0A {10} — Program

<Item>	<Data1>	<Data2>…	Description of Item
&20{32}$_{RT}$	0–100	N/A	Set Loudness. <Data1> = loudness value.
&21{33}$_{RT}$	0, 1	0–100	Set Velocity Sensitivity. Values range from –100 to +100, where: <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&22{34}$_{RT}$	1, 2	0–14	Set Amp Mod Source. <Data1> = Amp Mod (1 or 2) <Data2> = Modulation Source. (see Table 15)
&23{35}$_{RT}$	1, 2	0, 1 <Data3>0–100	Set Amp Mod Value. <Data1> = Amp Mod (1 or 2) <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
&24{36}$_{RT}$	1, 2, 3	0–14	Set Pan Mod Source. <Data1> = Pan Mod (1, 2 or 3) <Data2> = Modulation Source. (see Table 15)
&25{37}$_{RT}$	1, 2, 3	0, 1 <Data3>0–100	Set Pan Mod Value. <Data1> = Pan Mod (1, 2 or 3) <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
	Output — Get Values		
&28{40}	N/A	N/A	Get Loudness.
&29{41}	N/A	N/A	Get Velocity Sensitivity.
&2A{42}	1, 2	N/A	Get Amp Mod Source. <Data1> = Amp Mod (1 or 2)
&2B{43}	1, 2	N/A	Get Amp Mod Value. <Data1> = Amp Mod (1 or 2)
&2C{44}	1, 2, 3	N/A	Get Pan Mod Source. <Data1> = Pan Mod (1, 2 or 3)
&2D{45}	1, 2, 3	N/A	Get Pan Mod Value. <Data1> = Pan Mod (1, 2 or 3)
	MIDI/Tune — Set Values		
&30{48}$_{RT}$	0, 1	0–36	Semitone Tune. <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&31{49}$_{RT}$	0, 1	0–50	Fine Tune. <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&32{50}	0–7	N/A	Tune Template, where <Data1> = template. 0=USER, 1=EVEN-TEMPERED, 2=ORCHESTRAL, 3=WERKMEISTER, 4=1/5 MEANTONE, 5=1/4 MEANTONE, 6=JUST, 7=ARABIAN.
&33{51}	0, 1	0–50…	Set User Tune Template. All the values are sent one after the other starting at C. The format of each value is the same as for Item &31{49}. (*i.e.*, 24 data bytes are transmitted, representing all 12 notes.)
&34{52}	0–11	N/A	Key = <Data1> where: 0=C, 1=C#, 2=D, 3=Eb, 4=E, 5=F, 6=F#, 7=G, 8=G#, 9=A, 10=Bb, 11=B
	MIDI/Tune — Get Values		
&38{56}	N/A	N/A	Get Semitone Tune.
&39{57}	N/A	N/A	Get Fine Tune.
&3A{58}	N/A	N/A	Get Tune Template.
&3B{59}	N/A	N/A	Get User Tune Template.
&3C{60}	N/A	N/A	Get Key.
	Pitch Bend — Set Values		
&40{64}	0–24	N/A	Set Pitch Bend Up. <Data1> = semitones
&41{65}	0–24	N/A	Set Pitch Bend Down. <Data1> = semitones
&42{66}	0, 1	N/A	Set Bend Mode. <Data1> = mode, where 0=NORMAL, 1=HELD
&43{67}	0, 1	0–12	Set Aftertouch Value. <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.

Table 13: Control Items for Section &0A {10} — Program

<Item>	<Data1>	<Data2>…	Description of Item
&44{68}	0, 1	N/A	Set Legato Setting <Data1> = mode, where 0=OFF, 1=ON
&45{69}	0, 1	N/A	Set Portamento Enable <Data1> = mode, where 0=OFF, 1=ON
&46{70}	0, 1	N/A	Set Portamento Mode <Data1> = mode, where 0=TIME, 1=RATE
&47{71}	0–100	N/A	Set Portamento Time
	Pitch Bend — Get Values		
&48{72}	N/A	N/A	Get Pitch Bend Up
&49{73}	N/A	N/A	Get Pitch Bend Down
&4A{74}	N/A	N/A	Get Bend Mode
&4B{75}	N/A	N/A	Get Aftertouch Value
&4C{76}	N/A	N/A	Get Legato Setting
&4D{77}	N/A	N/A	Get Portamento Enable
&4E{78}	N/A	N/A	Get Portamento Mode
&4F{79}	N/A	N/A	Get Portamento Time
	LFOs — Set Values. <Data1> = LFO (1=LFO1, 2=LFO2)		
&50{80}$_{RT}^{b}$	1, 2	0–100	Set LFO Rate. <Data2> = rate
&51{81}$_{RT}^{b}$	1, 2	0–100	Set LFO Delay. <Data2> = delay
&52{82}$_{RT}^{b}$	1, 2	0–100	Set LFO Depth. <Data2> = depth
&53{83}$_{RT}$	1, 2	0–8	Set LFO Waveform. <Data2> = waveform, where: 0=SINE, 1=TRIANGLE, 2=SQUARE, 3=SQUARE+, 4=SQUARE–, 5=SAW BI, 6=SAW UP, 7=SAW DOWN, 8=RANDOM
&54{84}$_{RT}$	1	0, 1	Set LFO Sync. <Data2> = (0=OFF, 1=ON). (LFO1 only)
&55{85}$_{RT}$	2	0, 1	Set LFO Re-trigger. <Data2> = (0=OFF, 1=ON). (LFO2 only)
&56{86}$_{RT}$	1, 2	0–14	Set Rate Mod Source <Data2> = Modulation Source. (see Table 15)
&57{87}$_{RT}$	1, 2	0, 1 <Data3>0–100	Set Rate Mod Value <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
&58{88}$_{RT}$	1, 2	0–14	Set Delay Mod Source <Data2> = Modulation Source. (see Table 15)
&59{89}$_{RT}$	1, 2	0, 1 <Data3>0–100	Set Delay Mod Value <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
&5A{90}$_{RT}$	1, 2	0–14	Set Depth Mod Source <Data2> = Modulation Source. (see Table 15)
&5B{91}$_{RT}$	1, 2	0, 1 <Data3>0–100	Set Depth Mod Value <Data2> = sign (0 = +ve, 1 = –ve), <Data3> = absolute value.
&5C{92}$_{RT}$	1	0–100	Set Modwheel <Data2> = value (LFO1 only)
&5D{93}$_{RT}$	1	0–100	Set Aftertouch <Data2> = value (LFO1 only)
&5E{94}$_{RT}$	2	0, 1	Set MIDI Clock Sync Enable <Data2> = (0=OFF, 1=ON) (LFO2 only)
&5F{95}$_{RT}$	2	0–68	Set MIDI Clock Sync Division (LFO2 only) <Data2> = 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
	LFOs — Get Values. <Data1> = LFO (1=LFO1, 2=LFO2)		
&60{96}^{b}	1, 2	N/A	Get LFO Rate
&61{97}^{b}	1, 2	N/A	Get LFO Delay

Table 13: Control Items for Section &0A {10} — Program

<Item>	<Data1>	<Data2>…	Description of Item
&62{98}[b]	1, 2	N/A	Get LFO Depth
&63{99}	1, 2	N/A	Get LFO Waveform
&64{100}	1	N/A	Get LFO Sync (LFO1 only)
&65{101}	2	N/A	Get LFO Re-trigger (LFO2 only)
&66{102}	1, 2	N/A	Get Rate Mod Source
&67{103}	1, 2	N/A	Get Rate Mod Value
&68{104}	1, 2	N/A	Get Delay Mod Source
&69{105}	1, 2	N/A	Get Delay Mod Value
&6A{106}	1, 2	N/A	Get Depth Mod Source
&6B{107}	1, 2	N/A	Get Depth Mod Value
&6C{108}	1	N/A	Get Modwheel (LFO1 only)
&6D{109}	1	N/A	Get Aftertouch (LFO1 only)
&6E{110}	2	N/A	Get MIDI Clock Sync Enable (LFO2 only)
&6F{111}	2	N/A	Get MIDI Clock Sync Division (LFO2 only)
	Keygroup Modulation Sources — Set Values		
&70{112}	1, 2	0–14	Set Pitch Mod Source. <Data1> = Pitch Mod (1 or 2) <Data2> = Modulation Source. (see Table 15)
&71{113}	1	0–14	Set Amp Mod Source. <Data1> = Amp Mod (1 only) <Data2> = Modulation Source. (see Table 15)
&72{114}	1, 2, 3	0–14	Set Filter Mod Input Source. <Data1> = Mod Input (1, 2 or 3) <Data2> = Modulation Source. (see Table 15)
	Keygroup Modulation Sources — Get Values		
&74{116}	1, 2	N/A	Get Pitch Mod Source. <Data1> = Pitch Mod (1 or 2)
&75{117}	1	N/A	Get Amp Mod Source. <Data1> = Amp Mod (1 only)
&76{118}	1, 2, 3	N/A	Get Filter Mod Input Source. <Data1> = Mod Input (1, 2 or 3)

a. The number sent is 1 smaller than that set via the S5000/6000 front panel. *i.e.*, numbers 1–128 shall be transmitted as 0–127.
b. The LFO Rate, Delay and Depth settings are only valid if the LFO is not synchronised to MIDI clock.

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 14. The parameters for the modulation sources used in the Programs are detailed in Table 15.

Table 14: Format of "REPLY" confirmation messages for Section &0A{10} — Program

<Item> requested	<Data1>	<Data2>	Description of Data Returned
	Getting General Program Information for the *Current* Program.		
&10{16}	0–127 (MSB)	0–127 (LSB)	The number of Programs in memory is returned, where: <Data1> = MSB; <Data2> = LSB (NumPrograms = LSB+128×MSB)
&11{17}	0, 1 (OFF,ON)	0–127[a]	The Current Program's "Program number" <Data1> = Program Number On or Off, 0=OFF, 1=ON <Data2> = Program Number (set to 0 if number is off).
&12{18}	0–127 (MSB)	0–127 (LSB)	Program Index (*i.e.*, position in memory), where: <Data1> = MSB; <Data2> = LSB (Index = LSB+128×MSB)

Table 14: Format of "REPLY" confirmation messages for Section &0A{10} — Program

<Item> requested	<Data1>	<Data2>	Description of Data Returned
&13{19}	char1	…0	The name of the Current Program, <Data1…0> is a character string.
&14{20}	0–99	N/A	<Data1> = Number of Keygroups in Current Program.
&15{21}	0, 1	N/A	Get Keygroup Crossfade <Data1>: 0=OFF; 1=ON
	Getting General Program Information for All the Programs in Memory.		
&18{24}[b]	0, 1 (OFF,ON)	0–127[a]	The "Program numbers" of all of the Programs in memory. Note that several <Data> fields may be transmitted (one set for every Program in memory). The message format is the same as for <Item> &11{17}.
&19{25}[b]	char1…	0…charN…0…	The names of all of the Programs in memory. *(See comments to <Item> &18{24} for more information.)*
	Output — Get Values		
&28{40}	0–100	N/A	Loudness = <Data1>
&29{41}	0, 1	0–100	Velocity Sensitivity values range from −100 to +100, where: <Data1> = sign (0 = +ve, 1 = −ve), <Data2> = absolute value.
&2A{42}	0–14	N/A	Amp Mod Source, where: <Data1> = Modulation Source. (see Table 15)
&2B{43}	0, 1	0–100	Amp Mod Value, where: <Data1> = sign (0 = +ve, 1 = −ve), <Data2> = absolute value.
&2C{44}	0–14	N/A	Amp Pan Source, where: <Data1> = Modulation Source. (see Table 15)
&2D{45}	0, 1	0–100	Amp Pan Value, where: <Data1> = sign (0 = +ve, 1 = −ve), <Data2> = absolute value.
	MIDI/Tune — Get Values		
&38{56}	0, 1	0–36	Semitone Tune. <Data1> = sign (0 = +ve, 1 = −ve), <Data2> = absolute value.
&39{57}	0, 1	0–50	Fine Tune. <Data1> = sign (0 = +ve, 1 = −ve), <Data2> = absolute value.
&3A{58}	0–7	N/A	Tune Template, where <Data1> = template. 0=USER, 1=EVEN-TEMPERED, 2=ORCHESTRAL, 3=WERKMEISTER, 4=1/5 MEANTONE, 5=1/4 MEANTONE, 6=JUST, 7=ARABIAN.
&3B{59}	0, 1	0–50…	Get User Tune Template Returns the fine tune settings of all of the 12 semitones one after another, starting at C. The format is the same as that in Item &39{57}
&3C{60}	0–11	N/A	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
	Pitch Bend — Get Values		
&48{72}	0–24	N/A	Get Pitch Bend Up <Data1> = semitones
&49{73}	0–24	N/A	Get Pitch Bend Down <Data1> = semitones
&4A{74}	0, 1	N/A	Get Bend Mode <Data1> = mode, where 0=NORMAL, 1=HELD
&4B{75}	0, 1	0–12	Get Aftertouch Value <Data1> = sign (0 = +ve, 1 = −ve), <Data2> = absolute value.
&4C{76}	0, 1	N/A	Get Legato Setting <Data1> = mode, where 0=OFF, 1=ON
&4D{77}	0, 1	N/A	Get Portamento Enable <Data1> = mode, where 0=OFF, 1=ON
&4E{78}	0, 1	N/A	Get Portamento Mode <Data1> = mode, where 0=TIME, 1=RATE
&4F{79}	0–100	N/A	Get Portamento Time <Data1> = time
	LFOs — Get Values.		

Table 14: Format of "REPLY" confirmation messages for Section &0A{10} — Program

<Item> requested	<Data1>	<Data2>	Description of Data Returned
&60{96}	0–100	N/A	LFO Rate = <Data1>
&61{97}	0–100	N/A	LFO Delay = <Data1>
&62{98}	0–100	N/A	LFO Depth = <Data1>
&63{99}	0–8	N/A	LFO Waveform = <Data1>, where: 0=SINE, 1=TRIANGLE, 2=SQUARE, 3=SQUARE+, 4=SQUARE–, 5=SAW BI, 6=SAW UP, 7=SAW DOWN, 8=RANDOM
&64{100}	0, 1	N/A	LFO Sync = <Data1>, where 0=OFF, 1=ON
&65{101}	0, 1	N/A	LFO Re-trigger = <Data1>, where 0=OFF, 1=ON
&66{102}	0–14	N/A	Rate Mod Source = <Data1>, where Modulation Source is returned as defined in Table 15.
&67{103}	0, 1	0–100	Rate Mod Value, where <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&68{104}	0–14	N/A	Delay Mod Source = <Data1>, where Modulation Source is returned as defined in Table 15.
&69{105}	0, 1	0–100	Delay Mod Value, where <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&6A{106}	0–14	N/A	Depth Mod Source = <Data1>, where Modulation Source is returned as defined in Table 15.
&6B{107}	0, 1	0–100	Depth Mod Value, where <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&6C{108}	0–100	N/A	Modwheel value = <Data1>
&6D{109}	0–100	N/A	Aftertouch value = <Data1>
&6E{110}	0, 1	N/A	MIDI Clock Sync Enable = <Data1>, where 0=OFF, 1=ON
&6F{111}	0–68	N/A	MIDI Clock Sync Division = <Data1>, 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
	Keygroup Modulation Sources — Get Values		
&74{116}	0–14	N/A	Get Pitch Mod Source. <Data1> = Modulation Source. (see Table 15)
&75{117}	0–14	N/A	Get Amp Mod Source. <Data1> = Modulation Source. (see Table 15)
&76{118}	0–14	N/A	Get Filter Mod Input Source <Data1> = Modulation Source. (see Table 15)

a. The number returned will always be 1 smaller than that set via the S5000/6000 front panel. *i.e.*, numbers 1–128 will be transmitted as 0–127.
b. It is recommended that the number of Programs in memory is determined before this <Item> is requested so that the controlling computer knows in advance how much data it is expecting. Information for the Programs will be returned in the order in which they are stored in memory.

Table 15: Modulation Sources as sent via SysEx.

SysEx Code	Mod Source	SysEx Code	Mod Source	SysEx Code	Mod Source
0	NO SOURCE	5	VELOCITY	10	FILT ENV
1	MODWHEEL	6	KEYBOARD	11	AUX ENV
2	BEND	7	LFO1	12	♩MODWHEEL
3	AFTERTOUCH	8	LFO2	13	♩BEND
4	EXTERNAL	9	AMP ENV	14	♩EXTERNAL

Item List for Multi section [&0C{12}]

The first implementation of the Multi section allowed control of all of the parameters of a Multi in real-time. For example, you could 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, the "Get" options allow suitable software to determine the current settings of a Multi.

This functionality has now been extended to provide 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 [&0A{10}]. 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.

The *index* used to select a multi by index, or returned when the current multi is requested, is zero-based, *i.e.*, the first multi in memory has *index*=0.

Table 16: Control Items for Section &0C{12} — Multi

<Item>	<Data1>	<Data2>…	Description of Item
	Multi Creation and Deletion		
&01{1}	0, 1, 2	N/A	Set the number of parts for new multis: 0=32, 1=64, 2=128.
&02{2}	char1	…0	Create Multi: Name is a string and must end with a zero.
&05{5}	char1	…0	Select Multi (by name) to be *current*: Name is a string and must end with a zero.
&06{6}	0–127(MSB)	0–127(LSB)	Select Multi (by index) to be *current*: <Data1> = MSB; <Data2> = LSB (index = LSB+128×MSB).
&07{7}	N/A	N/A	Delete ALL Multis from memory.
&08{8}	N/A	N/A	Delete the currently selected Multi from memory.
	Setting Multi Part Parameters (Data1 = Part Number)		
&10{16}$_{RT}$	0–127	&00–1F{0–31}	Set Part MIDI Channel, Data2: 1A=0, 2A=1, …, 16B=31
&11{17}$_{RT}$	0–127	0, 1	Set Part Mute, Data2: 0=OFF, 1=ON
&12{18}$_{RT}$	0–127	0, 1	Set Part Solo, Data2: 0=OFF, 1=ON
&13{19}$_{RT}$	0–127	&00–64{0–100}	Set Part Level, Data2=PartLevel
&14{20}$_{RT}$	0–127	&00–17{0–23}	Set Part Output, Data2=Output: 0–7 = op1/2–op15/16; 8–23 = op1-op16
&15{21}$_{RT}$	0–127	&0E–72{14–114}	Set Part Pan/Balance, Data2=Pan/Bal (14–114 = L50–R50); centre=&40{64}
&16{22}$_{RT}$	0–127	0–4	Set Part Effects Channel: Data2 = 0 (OFF), 1(FX1), 2(FX2), 3(RV3), 4(RV4)
&17{23}$_{RT}$	0–127	&00–64{0–100}	Set Part FX Send Level
&18{24}$_{RT}$	0–127	&00–64{0–100}	Set Part Fine Tune:Data2 = CentsTuning: (0 = –50cents; …; 100 = +50cents); centre=&32{50}

Table 16: Control Items for Section &0C{12} — Multi

<Item>	<Data1>	<Data2>…	Description of Item
&19{25}$_{RT}$	0–127	&00–48{0–72}	Set Part Transpose:Data2 = semitones: (0 = –36semitones; …; 72 = +36semitones); centre=&24{36}
&1A{26}$_{RT}$	0–127	&15–7F{21–127}	Set Part Low Note:Data2 = note: (21–127 = A-1 – G8)
&1B{27}$_{RT}$	0–127	&15–7F{21–127}	Set Part High Note:Data2 = note: (21–127 = A-1 – G8)
	Getting Multi Part Parameters (Data1 = Part Number) (Note: Parameters are returned as a single byte in a "REPLY" confirmation message.)		
&20{32}$_{RT}$	0–127	N/A	Get Part MIDI Channel *(See Set… above for parameter ranges)*
&21{33}$_{RT}$	0–127	N/A	Get Part Mute *(See Set… above for parameter ranges)*
&22{34}$_{RT}$	0–127	N/A	Get Part Solo *(See Set… above for parameter ranges)*
&23{35}$_{RT}$	0–127	N/A	Get Part Level *(See Set… above for parameter ranges)*
&24{36}$_{RT}$	0–127	N/A	Get Part Output *(See Set… above for parameter ranges)*
&25{37}$_{RT}$	0–127	N/A	Get Part Pan/Balance *(See Set… above for parameter ranges)*
&26{38}$_{RT}$	0–127	N/A	Get Part Effects Channel *(See Set… above for parameter ranges)*
&27{39}$_{RT}$	0–127	N/A	Get Part FX Send Level *(See Set… above for parameter ranges)*
&28{40}$_{RT}$	0–127	N/A	Get Part Fine Tune *(See Set… above for parameter ranges)*
&29{41}$_{RT}$	0–127	N/A	Get Part Transpose *(See Set… above for parameter ranges)*
&2A{42}$_{RT}$	0–127	N/A	Get Part Low Note *(See Set… above for parameter ranges)*
&2B{43}$_{RT}$	0–127	N/A	Get Part High Note *(See Set… above for parameter ranges)*
	Setting General Multi Information		
&30{48}	char1	…0	Rename Multi. <Data1…0> = New Name.
&31{49}	0, 1	0–127[a]	Set Multi Program Number Data1: 0=OFF, 1=ON, Data2: number (only required if Data1=1).
&32{50}	0–127	0–127(MSB)… 0–127(LSB)	Set Multi Part by index. <Data1> = Part Number; <Data2> = MSB of index; <Data3> = LSB of index (index = LSB+128×MSB)
&33{51}	0–127	char1…0	Set Multi Part by name. <Data1> = Part Number; <Data2…> = Name is a string and must end with a zero.
&34{52}	0–127	N/A	Delete Selected Part from Multi. <Data1> = Part Number
	Getting General Multi Information for the *Current* Multi (Note: The format of the data returned in the "REPLY" confirmation messages are detailed in Table 17.)		
&40{64}	N/A	N/A	Get Number of Multis in Memory
&41{65}	N/A	N/A	Get Current Multi's Program Number
&42{66}	N/A	N/A	Get Current Multi Index (*i.e.*, its position in memory)
&43{67}	N/A	N/A	Get Current Multi Name
&44{68}	N/A	N/A	Get the Number of Parts in the Current Multi
&45{69}	0–127	N/A	Get the Name of Part number <Data1> (returns 0 if none assigned)
&46{70}	N/A	N/A	Get the Names of all of the Parts in the multi.
&47{71}	0–127	N/A	Get all of the params (of <Items> &20–&2B) in a single message. <Data1> = Part Number to query.
&48{72}	N/A	N/A	Get the Mute and Solo status of all the parts in the current Multi.
	Getting General Multi Information for All the Multis in memory (Note: The format of the data returned in the "REPLY" confirmation messages are detailed in Table 17.)		

Table 16: Control Items for Section &0C{12} — Multi

<Item>	<Data1>	<Data2>…	Description of Item
&50{80}	N/A	N/A	Get the Multi Numbers of All Multis in memory
&51{81}	N/A	N/A	Get the Names of All Multis in memory
&52{82}	N/A	N/A	Get the Number of Parts in All the Multis in memory

a. The number set is 1 smaller than that set via the S5000/6000 front panel. *i.e.,* numbers 1–128 shall be transmitted as 0–127.

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 17.

Table 17: Format of "REPLY" confirmation messages for Section &0C{12} — Multi

<Item> requested	<Data1>	<Data2>	Description of Data Returned
&20–&2B {32–43}	0–127	N/A	Data is returned as a single byte in field <Data1>, corresponding to the requested parameter.
&40{64}	0–127 (MSB)	0–127 (LSB)	The number of Multis in memory, where: <Data1> = MSB; <Data2> = LSB (NumMultis = LSB+128×MSB).
&41{65}	0, 1 (OFF,ON)	0–127[a]	The Current Multi number <Data1> = Multi Number On or Off, 0=OFF, 1=ON <Data2> = Multi Number (set to 0 if number is off).
&42{66}	0–127 (MSB)	0–127 (LSB)	The Current Multi index (starting from 0). <Data1> = MSB; <Data2> = LSB (index = LSB+128×MSB).
&43{67}	char1	…0	The name of the Current Multi, <Data1…0> is a character string.
&44{68}	31, 63, 127	N/A	The number of parts in the Current Multi = <Data1> + 1
&45{69}	char1	…0	The name of the requested Part, <Data1…> is a character string. If there is no part assigned, char1 = 0.
&46{70}[b]	char1	…0…charN…0	The names of all of the parts in the current multi are returned in a single stream, punctuated by 0s. There will be 32, 64 or 128 names returned depending on the number of parts in the multi. If no part is assigned, the name will be returned as a single byte=0.
&47{71}	0–127	…0–127…	All of the parameters obtained using <Item>=&20–&2B are returned in a single message. 12 data bytes are returned.
&48{72}	0, 1, 2	0, 1, 2 …	One byte is returned for each part in the current Multi, where: 0=mute and solo off, 1=mute on, 2=solo on.
&50{80}[c]	0, 1 (OFF,ON)	0–127[a]…	The numbers of all of the Multis in memory. Note that several <Data> fields may be transmitted (one set for every multi in memory). The format of the message is the same as for <Item> &41{65}.
&51{81}[c]	char1	…0…	The names of all of the Multis in memory. *(See comments to <Item> &50{80} for more information.)*
&52{82}[c]	31, 63, 127	…	The number of parts in all the Multis in memory. *(See comments to <Item> &50{80} for more information.)*

a. The number returned will always be 1 smaller than that set via the S5000/6000 front panel. *i.e.,* numbers 1–128 will be transmitted as 0–127.
b. It is recommended that the number of parts in the current multi is determined before this <Item> is requested so that the controlling computer knows in advance how much data it is expecting. Information for the parts will be returned in order, starting from Part 1.
c. It is recommended that the number of multis in memory is determined before this <Item> is requested so that the controlling computer knows in advance how much data it is expecting. Information for the multis will be returned in the order in which they are stored in memory.

Item List for Sample section [&0E{14}]

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 &06{6}.

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.

The *index* used to select a sample by index, or returned when the current sample is requested, is zero-based, *i.e.*, the first sample in memory has *index*=0.

Table 18: Control Items for Section &0E{14} — Sample Tools

<Item>	<Data1>	<Data2>…	Description of Item
	Sample Selection		
&05{5}	char1	…0	Select Sample (by name) to be *current*: <Data1…0> = Name.
&06{6}	0–127(MSB)	0–127(LSB)	Select Sample (by index) to be *current*: <Data1>=MSB; <Data2>=LSB. (index = LSB+128×MSB)
&07{7}	N/A	N/A	Delete ALL samples from memory
&08{8}	N/A	N/A	Delete the currently selected Sample from memory.
&09{9}	char1	…0	Rename the currently selected Sample. <Data1…0> = Name.
&0A{10}	N/A	N/A	Start auditioning the current sample.
&0B{11}	N/A	N/A	Stop auditioning the current sample.
	Getting General Information of the Samples in Memory		
&10{16}	N/A	N/A	Get the Number of Samples loaded into memory.
&11{17}	0–127(MSB)	0–127(LSB)	Get the name of sample by index. <Data1>=MSB; <Data2>=LSB. (index = LSB+128×MSB)
&12{18}	N/A	N/A	Get the names of all of the samples in memory.
&13{19}	N/A	N/A	Get Current Sample's Index (*i.e.*, its position in memory)
&14{20}	N/A	N/A	Get Current Sample's Name
	Setting Sample Parameters		
&20{32}	0–127(MSB)	0–127(SB2) <Data3>=0–127(SB1) <Data4>=0–127(LSB)	Set Start Position A Compound Double Word (see page 9) representing the position from where the sample will start playback.
&21{33}	0–127(MSB)	0–127(SB2) <Data3>=0–127(SB1) <Data4>=0–127(LSB)	Set End Position A Compound Double Word (see page 9) representing the position where the sample will end playback.
&22{34}	&15–&7F {21–127}	N/A	Set Original Pitch <Data1> = note: (21–127 = A-1 – G8)
&23{35}	0, 1	0–36	Set Semitone Tune <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&24{36}	0, 1	0–50	Set Fine Tune <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&28{40}	0–5	N/A	Set Playback Mode, where <Data1> = mode. 0=NO LOOPING, 1=ONE SHOT, 2=LOOP IN REL, 3=LOOP UNTIL REL, 4=LIR→RETRIG, 5=PLAY→RETRIG.
&29{41}	0–127(MSB)	0–127(SB2) <Data3>=0–127(SB1) <Data4>=0–127(LSB)	Set Loop Start A Compound Double Word (see page 9) representing the position from where the sample loop will start.

Table 18: Control Items for Section &0E{14} — Sample Tools

<Item>	<Data1>	<Data2>…	Description of Item
&2A{42}	0–127(MSB)	0–127(SB2) <Data3>=0–127(SB1) <Data4>=0–127(LSB)	Set Loop End A Compound Double Word (see page 9) representing the position where the sample loop will end.
	Getting Sample Parameters		
&30{48}	N/A	N/A	Get Sample Type (VIRTUAL/RAM)
&31{49}	N/A	N/A	Get Number of Channels
&32{50}	N/A	N/A	Get Sample Length
&33{51}	N/A	N/A	Get Sample Rate
&34{52}	N/A	N/A	Get all of the params (of <Items> &30–&33) in a single message.
&40{64}	N/A	N/A	Get Start Position
&41{65}	N/A	N/A	Get End Position
&42{66}	N/A	N/A	Get Original Pitch
&43{67}	N/A	N/A	Get Semitone Tune
&44{68}	N/A	N/A	Get Fine Tune
&48{72}	N/A	N/A	Get Playback Mode
&49{73}	N/A	N/A	Get Loop Start
&4A{74}	N/A	N/A	Get Loop End
&4B{75}	N/A	N/A	Get all of the params (of <Items> &40–&4A) in a single message.

When Sample information is requested by a *Get* message, it is returned in a REPLY confirmation message (see *Confirmation Messages* on page 5) — unless an error occurs, whereupon an ERROR confirmation message is returned instead. The format of these REPLY messages is detailed in Table 19.

Table 19: Format of "REPLY" confirmation messages for Section &0E{14} — Sample Tools

<Item> requested	<Data1>	<Data2>	Description of Data Returned
&10{16}	0–127(MSB)	0–127(LSB)	The number of samples in memory is returned, where number = LSB+128×MSB
&11{17}	char1	…0	The name of the requested sample is returned as a character string.
&12{18}	char1…0…	charN…0	The names of all of the samples in memory are returned as a stream of character strings within a single REPLY message.
&13{19}	0–127(MSB)	0–127(LSB)	Current Sample Index, where: <Data1>=MSB; <Data2>=LSB. (index = LSB+128×MSB)
&14{20}	char1	…0	Current Sample's name*:* <Data1…0> = Name
	Getting Sample Parameters		
&30{48}	0, 1	N/A	Get Sample Type (VIRTUAL/RAM) <Data1>: 0=RAM, 1=VIRTUAL.
&31{49}	1, 2	N/A	Get Number of Channels <Data1>: 1=MONO, 2=STEREO
&32{50}	0–127(MSB)	0–127(SB2) <Data3>=0–127(SB1) <Data4>=0–127(LSB)	Get Sample Length A Compound Double Word (see page 9) representing the total length of the sample (*i.e.*, number of sample points).
&33{51}	0–127(MSB)	0–127(SB2) <Data3>=0–127(SB1) <Data4>=0–127(LSB)	Get Sample Rate A Compound Double Word (see page 9) representing the sample rate (in Hz) of the sample.

Table 19: Format of "REPLY" confirmation messages for Section &0E{14} — Sample Tools

<Item> requested	<Data1>	<Data2>	Description of Data Returned
&34{52}	0–127…	…0–127…	All of the parameters obtained using <Item>=&30–&33 are returned in a single message. 10 data bytes are returned.
&40{64}	0–127(MSB)	0–127(SB2) <Data3>=0–127(SB1) <Data4>=0–127(LSB)	Get Start Position A Compound Double Word (see page 9) representing the position from where the sample will start playback.
&41{65}	0–127(MSB)	0–127(SB2) <Data3>=0–127(SB1) <Data4>=0–127(LSB)	Get End Position A Compound Double Word (see page 9) representing the position where the sample will end playback.
&42{66}	&15–&7F {21–127}	N/A	Get Original Pitch <Data1> = note: (21–127 = A-1 – G8)
&43{67}	0, 1	0–36	Get Semitone Tune <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&44{68}	0, 1	0–50	Get Fine Tune <Data1> = sign (0 = +ve, 1 = –ve), <Data2> = absolute value.
&48{72}	0–5	N/A	Get Playback Mode, where <Data1> = mode. 0=NO LOOPING, 1=ONE SHOT, 2=LOOP IN REL, 3=LOOP UNTIL REL, 4=LIR→RETRIG, 5=PLAY→RETRIG.
&49{73}	0–127(MSB)	0–127(SB2) <Data3>=0–127(SB1) <Data4>=0–127(LSB)	Get Loop Start A Compound Double Word (see page 9) representing the position from where the sample loop will start.
&4A{74}	0–127(MSB)	0–127(SB2) <Data3>=0–127(SB1) <Data4>=0–127(LSB)	Get Loop End A Compound Double Word (see page 9) representing the position where the sample loop will end.
&4B{75}	0–127…	…0–127…	All of the parameters obtained using <Item>=&40–&4A are returned in a single message. 22 data bytes are returned.

Item List for Disk Tools section [&10{16}]

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.

> Note: Although the index of a file or folder can be used to obtain that item's name, caution must be used when using this method because the indices of these items are likely to change if the disk is modified in any way (deletion, addition, or renaming).

Table 20: Control Items for Section &10{16} — Disk Tools

<Item>	<Data1>	<Data2>	Description of Item
	General Disk Functions		
&01{1}	N/A	N/A	Update the list of disks connected.
&02{2}	0–127(MSB)	0–127(LSB)	Select Disk: Disk Handle = <Data2> + 128×<Data1>.
&03{3}[a]	0–127(MSB)	0–127(LSB)	Test if the disk is valid. Disk Handle = <Data2> + 128×<Data1>.
&04{4}[b]	N/A	N/A	Get the number of disks connected.
&05{5}	N/A	N/A	Get list of all connected disks.
&06{6}	N/A	N/A	Get current disk type.
&07{7}	0–127(MSB)	0–127(LSB)	Get type of specified drive. Disk Handle = <Data2> + 128×<Data1>
&08{8}	N/A	N/A	Get index of current disk.
&09{9}	N/A	N/A	Get current path of current disk.
&0A{10}	N/A	N/A	Get Format of current disk.
&0B{11}	N/A	N/A	Get Free Space on current disk.
&0D{13}	0–127(MSB)	0–127(LSB) <Data3> = 0, 1	Eject Disk. Disk Handle = <Data2> + 128×<Data1> <Data3>: 0=fail if drive in use, 1=discard virtual samples.
&0E{13}	0–127(MSB)	0–127(LSB)	Get the name of the specified disk Disk Handle = <Data2> + 128×<Data1>
	Folder Functions		
&10{16}	N/A	N/A	Get number of sub-folders in the current folder.
&11{17}	0–127 (MSB)	0–127 (LSB)	Get name of specified folder. Index of folder (starting from 0) = <Data2> + 128×<Data1>.
&12{18}	N/A	N/A	Get the names of all of the sub-folders in the current folder.
&13{19}	char1	…0	Open Folder. This sets the current folder to be the requested one. (Move down one level in the folder heirarchy.) <Data1…0> = name of folder to open.. *(If <Data1> = 0, the root folder will be selected.)*
&14{20}	N/A	N/A	Close Folder. (Move up one level in the folder heirarchy.) If this is used on a root folder, an ERROR confirmation message will be returned.

Table 20: Control Items for Section &10{16} — Disk Tools

<Item>	<Data1>	<Data2>	Description of Item
&15{21}	char1	…0	Load Folder: the selected folder, and all its contents (including sub-folders) are loaded into memory. <Data1…0> = name of folder to load.
&16{22}	char1	…0	Create Folder: Creates a sub-folder in the currently selected folder. <Data1…0> = name of folder to create.
&17{23}	char1	…0	Delete Sub-Folder. <Data1…0> = name of folder to delete.
&18{24}	char1…0	char1…0	Rename Folder: <Data1…0> = name of folder to rename, <DataN…0> = new name for folder.
	File Functions		
&20{32}	N/A	N/A	Get number of files in the current folder.
&21{33}	0–127 (MSB)	0–127 (LSB)	Get name of specified file. Index of file (starting from 0) = <Data2> + 128×<Data1>.
&22{34}	N/A	N/A	Get the names of all of the files in the current folder.
&23{35}	0–127 (MSB)	0–127 (LSB)	Get the size (in bytes) of the specified file. Index of file (starting from 0) = <Data2> + 128×<Data1>.
&24{36}	char1	…0	Get index of the file with the name: <Data1…0>
&28{40}[c]	char1…0	char1…0	Rename File: <Data1…0> = name of file to rename, <DataN…0> = new name for file.
&29{41}	char1	…0	Delete File. <Data1…0> = name of file to delete.
&2A{42}[d]	char1	…0 <DataN>=0, 1, 2	Load File <Data1…0> = name of file to load. The filename must include the correct file extension. <DataN>: 0=normal, 1=load samples as RAM, 2=load samples as VIRTUAL.
&2B{43}[e]	char1	…0	Load File including all dependent child files. <Data1…0> = name of file to load. The filename must include the correct file extension.
&2C{44}	0–127 (MSB)	0–127 (LSB) <Data3> 1–6 <Data4> 0, 1 <Data5> 0, 1	Save Memory Item to Disk Index of Memory Item = <Data2> + 128×<Data1>. <Data3> = Type.(1=Multi; 2=Program; 3=Sample; 4=SMF; 5=Setlist; 6=Scenelist) <Data4>: 0=skip if file exists; 1=Overwrite existing files <Data5>: 0=Don't save children; 1=Save Children.
&2D{45}	1–6	0, 1 <Data3> 0, 1	Save All Memory Items to Disk <Data1> = Type.(1=Multi; 2=Program; 3=Sample; 4=SMF; 5=Setlist; 6=Scenelist) <Data2>: 0=skip if file exists; 1=Overwrite existing files <Data3>: 0=Don't save children; 1=Save Children.
&30{48}	0–127 (MSB)	0–127 (LSB)	Start auditioning a sample from disk. Index of file (starting from 0) = <Data2> + 128×<Data1>
&31{49}	N/A	N/A	Stop auditioning a sample.

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. When renaming a file, the file extension must not be appended to the new name, as this is automatically added.
d. Loading a file will load a multi, program, sample, or scenelist 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.
e. 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 21.

Table 21: Format of "REPLY" confirmation messages for Section &10{16} — Disk Tools

<Item> requested	<Data1>	<Data2>	Description of Data Returned
	General Disk Functions		
&04{4}	0–127	N/A	<Data1> = number of logical disks connected to the sampler.
&05{5}[a]	0–127 (handle MSB)	0–127 (handle LSB) <Data3>=0, 1, 2, 3 <Data4>=0–7 <Data5>=0–7 <Data6>=0,1 <Data7>=char1…0…	Disk Handle = <Data2> + 128×<Data1>. <Data3> = the type of the disk: (See Item &07) <Data4>=Disk Format (see Item &0A) <Data5>=Physical (SCSI) disk ID <Data6>=Disk is writable (0=no, 1=yes) <Data7…0> = string containing the name of the disk. This sequence (<handle><type><format><id><name…0>) is repeated for each of the logical disks connected to the sampler.
&06{6}	0, 1, 2, 3	N/A	<Data1> = the type of the currently selected disk: 0=floppy; 1=hard disk; 2=CD ROM; 3=removable disk.
&07{7}	0, 1, 2, 3	N/A	<Data1> = the type of the specified disk: 0=floppy; 1=hard disk; 2=CD ROM; 3=removable disk. (Any other return value represents an unknown disk type.)
&08{8}	0–127 (handle MSB)	0–127 (handle LSB)	<Data1>, <Data2> = the handle of the currently selected disk (*i.e.*, the handle which was used with <Item>&02 to select the disk).
&09{9}	char1…0	N/A	<Data1…0> = string containing the current path on the current disk. If the root folder is selected, this will return a single byte = 0.
&0A{10}	0–7	N/A	Get Format of current disk. <Data1> = format, where: 0=other; 1=MSDOS; 2=FAT32; 3=ISO9660; 4=S1000; 5=S3000; 6=EMU; 7=ROLAND.
&0B{11}	QWORD<8bytes>	…	Get Free Space on current disk. 8 bytes are returned as a Compound Quad Word (see page 9).
	Folder Functions		
&10{16}	0–127 (MSB)	0–127 (LSB)	The number of sub-folders in the current folder = <Data2> + 128×<Data1>
&11{17}	char1…0	N/A	<Data1…0> = string containing the name of the requested folder.
&12{18}[b]	char1…0…	charN…0	Returns a list of the names of all of the sub-folders. <Data1…0> = name of first folder. This is followed by the names of all of the other folders.
	File Functions		
&20{32}	0–127 (MSB)	0–127 (LSB)	The number of files in the current folder = <Data2> + 128×<Data1>
&21{33}	char1…0	N/A	<Data1…0> = string containing the name of the requested file.
&22{34}[c]	char1…0…	charN…0	Returns a list of the names of all the files in the current folder. <Data1…0> = name of first file. This is followed by the names of all of the other files.
&23{35}	0–127(MSB)	0–127(SB2) <Data3>=0–127(SB1) <Data4>=0–127(LSB)	Get file size (in bytes) A Compound Double Word (see page 9) representing the size of the file.
&24{36}	0–127 (MSB)	0–127 (LSB)	The index of the file is returned if found.

a. It is recommended that the number of disks is determined before this <Item> is requested so that the controlling computer knows in advance how much data it is expecting.
b. It is recommended that the number of sub-folders is determined before this <Item> is requested so that the controlling computer knows in advance how many character strings it is expecting.
c. It is recommended that the number of files is determined before this <Item> is requested so that the controlling computer knows in advance how many character strings it is expecting.

Item List for Multi FX Control section [&12{18}]

Effects settings belong to a Multi, and this section enables the adjustment of effects for the currently selected multi (see Section &0C 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 then disabled (bypassed) or even changed for other modules (depending on the selected module and the effects hardware being used).

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 alternative effects modules selected.

The arrangement of the EB20 effects board, described by the above scheme is illustrated in Figure 2. Note that with this effects board, only modules 2 and 3, of channels 0 and 1 may be changed—these represent the modulation and delay effects sections, respectively.

> Note that all of the index bytes are zero-based. (*i.e.*, the first item = 0, the second item = 1.)
>
> Parameter values are always passed as a Signed Compound Word (three data bytes: <sign><MSB><LSB>) 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.

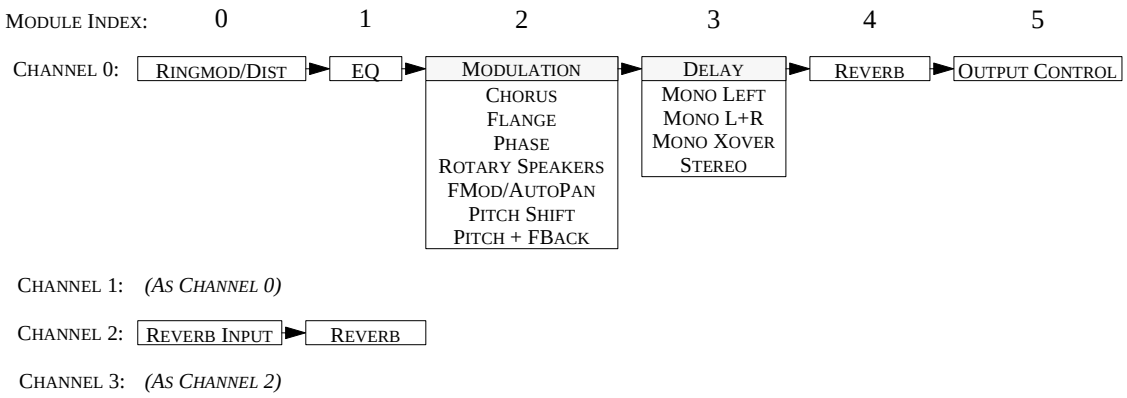

Figure 2: Organisation of the EB20 effects board.

Table 22: Control Items for Section &12{18} — Multi FX Control

<Item>	<Data1>	<Data2>…	Description of Item
&01{1}	Determine if there is an FX board installed or not.		
	N/A	N/A	Get FX card installed.
&10{16}	Determine the characteristics of the installed FX.		
	N/A	N/A	Get Number of FX channels.
&11{17}	0–127	N/A	Get Number of FX modules on given channel. <Data1> = channel.

Table 22: Control Items for Section &12{18} — Multi FX Control

<Item>	<Data1>	<Data2>…	Description of Item
	Setting/Getting the configuration of the FX channels.		
&20{32}	0–127	0, 1	Set Mute Status of Channel 0 = ON, 1 = MUTE.
&21{33}	0–127	N/A	Get Mute Status of Channel
	Setting/Getting the configuration of the FX modules.		
&30{48}	0–127	0–127 <Data3>=0–127	Set Type of FX module on given channel. <Data1> = channel; <Data2> = module; <Data3> = module type. (see Table 24 for decription of module types)
&31{49}	0–127	0–127	Get Type of FX module on given channel. <Data1> = channel; <Data2> = module.
	Enabling/Disabling FX modules.		
&40{64}	0–127	0–127 <Data3>=0, 1	Set Enabled/Disabled State of FX module. <Data1> = channel; <Data2> = module; <Data3> =0 (disable) or 1 (enable)
&41{65}	0–127	0–127	Get Enabled/Disabled State of FX module. <Data1> = channel; <Data2> = module
	Setting/Getting FX parameter values.		
&50{80}	0–127	0–127 <Data3>=0–127 <Data4>=0, 1 <Data5>=0–127 <Data6>=0–127	Set parameter value of given module in given channel. <Data1> = channel; <Data2> = module; <Data3> = index of parameter to set (see Table 25). <Data4> = sign (0 = +ve, 1 = –ve) <Data5> = Parameter Value (MSB); <Data6> = Parameter Value (LSB); (absolute value = LSB + 128× MSB)
&51{81}	0–127	0–127 <Data3>=0–127	Get parameter value of given module in given channel. <Data1> = channel; <Data2> = module; <Data3> = index of parameter to set (see Table 25) (Reply Value is a compound word, as in Item &50, above.

Table 23: Format of "REPLY" confirmation messages for Section &12{18} — Multi FX Control

<Item> requested	<Data1>	<Data2>	Description of Data Returned
&01{1}	0, 1	N/A	<Data1> = FX card installed: 0=none, 1=EB20.
&10{16}	0–127	N/A	<Data1> = Number of FX channels.
&11{17}	0–127	N/A	<Data1> = Number of FX modules on given channel.
&21{33}	0, 1	N/A	<Data1> = Mute Status of FX Channel 0=ON; 1=MUTE
&31{49}	0–127	N/A	<Data1> = Type of FX module on given channel (see Table 24 for decription of module types)
&41{65}	0, 1	N/A	<Data1> = 0 (Module disabled); 1(Module enabled)
&51{81}	0, 1	0–127 <Data3> =0–127	Get parameter value of given module in given channel. <Data1> = sign (0 = +ve, 1 = –ve) <Data2> = Parameter Value (MSB); <Data3> = Parameter Value (LSB); (absolute value = LSB + 128× MSB)

Table 24: Coding of FX Modules[a]

Value	Module	Value	Module	Value	Module
&00{00}	none	&01{01}	EB20 RingMod/Distortion	&02{02}	EB20 Chorus
&03{03}	EB20 Flange	&04{04}	EB20 Phase	&05{05}	EB20 Rotary Speaker
&06{06}	EB20 FMod Autopan	&07{07}	EB20 PitchShift	&08{08}	EB20 PitchShift + Feedback
&09{09}	EB20 EQ	&0A{10}	EB20 Mono Delay	&0B{11}	EB20 Mono L+R
&0C{12}	EB20 Mono Xover	&0D{13}	EB20 Stereo Delay	&0E{14}	EB20 Reverb
&0F{15}	EB20 Output Mix	&10{16}	EB20 Reverb Input		

a. See Table 25 for a description of the parameters associated with these modules.

Table 25: Description of Parameters of FX Modules

Param.	Function	Range
EB20 RingMod/Distortion		
0	Distortion	0–100
1	Output Level	0–100
2	RMod Frequency	1–5000Hz
3	RMod Depth	0–100
EB20 Chorus/Flange/Phase		
0	Rate	0–99 = 0.0–9.9
1	Depth	0–100
2	Feedback	–50–+50
EB20 Rotary Speaker		
0	Speed 1	0–99 = 0.0–9.9
1	Speed 2	0–99 = 0.0–9.9
2	Acceleration	0–99 = 0.0–9.9
3	Depth/Width	0–100
4	Init Speed	0, 1 (Speed1/2)
5	MIDI Control	1–127
6	MIDI Mode	0, 1 (level/toggle)
7	MIDI Channel	0–31 = (1A–16B)
EB20 PitchShift		
0	Left Semitone	–50–+50
1	Left Fine	–50–+50
2	Right Semitone	–50–+50
3	Right Fine	–50–+50
EB20 PitchShift + Feedback		
0	Left Semitone	–50–+50

Param.	Function	Range
EB20 EQ		
0	Low Freq	0–30 = 16–500Hz
1	Low Gain	–37–+12dB
2	Low Mid Freq	0–44 = 40–6k3Hz
3	Low Mid Gain	–37–+12dB
4	Low Mid Width	0–100
5	High Mid Freq	0–44 = 40–6k3Hz
6	High Mid Gain	–37–+12dB
7	High Mid Width	0–100
8	High Freq	0–30 = 500–16kHz
9	High Gain	–37–+12dB
10	Low Mid Sweep Rate	0–99 = 0.0–9.9
11	Low Mid Sweep Depth	0–100
12	High Mid Sweep Rate	0–99 = 0.0–9.9
13	High Mid Sweep Depth	0–100
EB20 FMod Autopan		
0	FMod Rate	0–99 = 0.0–9.9
1	FMod Depth	0–100
2	FMod Feedback	0–100
3	AMod Rate	0–99 = 0.0–9.9
4	AMod Depth	0–100
5	AMod Mode	0–3=PAN–TREMOLO
EB20 Reverb		
0	Reverb Type	0–6
1	Pre-Delay	0–90

Table 25: Description of Parameters of FX Modules

Param.	Function	Range
1	Left Fine	–50–+50
2	Left Delay	0–275ms
3	Left Feedback	0–100
4	Right Semitone	–50–+50
5	Right Fine	–50–+50
6	Right Delay	0–275ms
7	Right Feedback	0–100
EB20 Mono Delay / L+R / Xover		
0	Delay Time	0–670ms
1	Feedback	0–100%
2	HF Damping	0–46 = 100–20kHz
3	Ping Pong	–50–+50
4	Feedback Monitor	0, 1 = PRE, POST
EB20 Stereo Delay		
0	Left Delay Time	0–335ms
1	Left Feedback	0–100%
2	Left HF Damping	0–46 = 100–20kHz
3	Right Delay Time	0–335ms
4	Right Feedback	0–100%
5	Right HF Damping	0–46 = 100–20kHz
6	Feedback Monitor	0, 1 = PRE, POST

Param.	Function	Range
2	Reverb Decay / Time	0–100
3	Diffusion	0–100
4	Near	0–100
5	Reverb Level	0–100
6	Reverb Pan	–50–+50=L50–R50
7	LF Damping	0–40 = 10–1k0Hz
8	HF Damping	0–25 = 1k0–20kHz
EB20 Output Mix		
0	Mod/Delay Level	0–100
1	Mod/Delay Pan	–50–+50=L50–R50
2	Mod/Delay Width	0–100
3	Direct Signal	0, 1 = OFF, ON
4	Path Control	–50–+50
5	Dist/EQ Level	0–100
6	Dist/EQ Pan	–50–+50L50–R50
EB20 Reverb Input		
0	RV Input	0–4

Item List for Scenelist Manipulation section [&14{20}]

The protocol provided for scenelist management only permits the manipulation of existing scenelists; it does not currently allow the building and adjustment of scenelists. Table 26 details this protocol.

Table 26: Control Items for Section &14{20} — Scenelist Manipulation

<Item>	<Data1>	<Data2>…	Description of Item
	Scenelist Selection		
&05{5}	char1	…0	Select Scenelist (by name) to be *current*: <Data1…0> = Name.
&06{6}	0–127(MSB)	0–127(LSB)	Select Scenelist (by index) to be *current*: <Data1>=MSB; <Data2>=LSB. (index = LSB+128×MSB)
&08{8}	N/A	N/A	Delete the currently selected Scenelist from memory.
&09{9}	char1	…0	Rename the currently selected Scenelist. <Data1…0> = Name.
	Getting General Information of the Scenelist in Memory		
&10{16}	N/A	N/A	Get the Number of Scenelists loaded into memory.
&11{17}	0–127(MSB)	0–127(LSB)	Get the name of Scenelist by index. <Data1>=MSB; <Data2>=LSB. (index = LSB+128×MSB)
&13{19}	N/A	N/A	Get Current Scenelist's Index (*i.e.*, its position in memory)
&14{20}	N/A	N/A	Get Current Scenelist's Name

When Scenelist information is requested by a *Get* message, it is returned in a REPLY confirmation message (see *Confirmation Messages* on page 5) — unless an error occurs, whereupon an ERROR confirmation message is returned instead. The format of these REPLY messages is detailed in Table 27.

Table 27: Format of "REPLY" confirmation messages for Section &14{20} — Scenelist

<Item> requested	<Data1>	<Data2>	Description of Data Returned
&10{16}	0–127(MSB)	0–127(LSB)	The number of Scenelists in memory is returned, where number = LSB+128×MSB
&11{17}	char1	…0	The name of the requested Scenelist is returned as a character string.
&13{19}	0–127(MSB)	0–127(LSB)	Current Scenelists Index, where: <Data1>=MSB; <Data2>=LSB. (index = LSB+128×MSB)
&14{20}	char1	…0	Current Scenelist's name: <Data1…0> = Name

Item List for MIDI Song File Tools section [&16{22}]

The MIDI song file tools provide a simple means to determine which song files are currently loaded. Control of playback of these files should be done by standard MIDI messages, not by SysEx.

Table 28: Control Items for Section &16{22} — *MIDI Song File* Tools

<Item>	<Data1>	<Data2>…	Description of Item
	Scenelist Songfile		
&05{5}	char1	…0	Select Songfile (by name) to be *current*: <Data1…0> = Name.
&06{6}	0–127(MSB)	0–127(LSB)	Select Songfile (by index) to be *current*: <Data1>=MSB; <Data2>=LSB. (index = LSB+128×MSB)
&08{8}	N/A	N/A	Delete the currently selected Songfile from memory.
&09{9}	char1	…0	Rename the currently selected Songfile. <Data1…0> = Name.
	Getting General Information of the Songfile in Memory		
&10{16}	N/A	N/A	Get the Number of Songfiles loaded into memory.
&11{17}	0–127(MSB)	0–127(LSB)	Get the name of Songfile by index. <Data1>=MSB; <Data2>=LSB. (index = LSB+128×MSB)
&13{19}	N/A	N/A	Get Current Songfile's Index (*i.e.*, its position in memory)
&14{20}	N/A	N/A	Get Current Songfile's Name
&20{32}	N/A	N/A	Get the Number of Set Lists loaded into memory.
&21{33}	0–127(MSB)	0–127(LSB)	Get the name of Set List by index. <Data1>=MSB; <Data2>=LSB. (index = LSB+128×MSB)
&22{34}	0–127(MSB)	0–127(LSB)	Delete the Set List by index. <Data1>=MSB; <Data2>=LSB. (index = LSB+128×MSB)
&23{35}	0–127(MSB)	0–127(LSB) <Data3…0> = char1…0	Rename the Set List by index. <Data1>=MSB; <Data2>=LSB. (index = LSB+128×MSB) <Data3…0> = New name.

When MIDI songfile information is requested by a *Get* message, it is returned in a REPLY confirmation message (see *Confirmation Messages* on page 5) — unless an error occurs, whereupon an ERROR confirmation message is returned instead. The format of these REPLY messages is detailed in Table 27.

Table 29: Format of "REPLY" confirmation messages for Section &14{20} — *MIDI Song File*s

<Item> requested	<Data1>	<Data2>	Description of Data Returned
&10{16}	0–127(MSB)	0–127(LSB)	The number of Songfiles in memory is returned, where number = LSB+128×MSB
&11{17}	char1	…0	The name of the requested Songfile is returned as a character string.
&13{19}	0–127(MSB)	0–127(LSB)	Current Songfiles Index, where: <Data1>=MSB; <Data2>=LSB. (index = LSB+128×MSB)
&14{20}	char1	…0	Current Songfile's name: <Data1…0> = Name
&20{32}	0–127(MSB)	0–127(LSB)	The number of Set Lists in memory is returned, where number = LSB+128×MSB
&21{33}	char1	…0	The name of the requested Set List is returned as a character string.

Item List for Front Panel Control section [&20{32}]

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 31.

> 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 ENT/PLAY is pressed for a few seconds to hear a sample, before being released.

Table 30: Control Items for Section &20{32} — Front Panel Control

<Item>	<Data1>	<Data2>…	Description of Item[a]
&01{1}	64–107[b]	N/A	"Key Hold": hold down panel key: <Data1> = keycode.
&02{2}	64–107[b]	N/A	"Key Release": release panel key: <Data1> = keycode.
&03{3}	0, 1	1–8	"Data Wheel". Move the data wheel forwards or backwards a certain amount, where: <Data1>=direction (0=forwards, 1=backwards), <Data2>=number of *clicks* to move wheel.
&04{4}	0–127	N/A	ASCII keyboard data: <Data1> = ASCII value.

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 31 should be used. Use of other values may lead to undefined behaviour.

Table 31: Keycodes for use with Section &20{32}

Key	<Keycode>	Key	<Keycode>	Key	<Keycode>	Key	<Keycode>	Key	<Keycode>
Mode Keys		Function Keys		Function Keys		Numeric Keys		Other Keys	
MULTI	&44{68}	F1	&48{72}	F9	&50{80}	0	&58{88}	MARK	&68{104}
FX	&40{64}	F2	&49{73}	F10	&51{81}	1	&59{89}	JUMP	&69{105}
EDIT SAMPLE	&42{66}	F3	&4A{74}	F11	&52{82}	2	&5A{90}	WINDOW	&67{103}
EDIT PROGRAM	&43{67}	F4	&4B{75}	F12	&53{83}	3	&5B{91}	EXIT	&6A{106}
RECORD	&41{65}	F5	&4C{76}	F13	&54{84}	4	&5C{92}	ENT/PLAY	&6B{107}
UTILITIES	&45{69}	F6	&4D{77}	F14	&55{85}	5	&5D{93}	CURSOR <	&64{100}
SAVE	&46{70}	F7	&4E{78}	F15	&56{86}	6	&5E{94}	CURSOR >	&65{101}
LOAD	&47{71}	F8	&4F{79}	F16	&57{87}	7	&5F{95}	+	&63{99}
						8	&60{96}	–	&62{98}
						9	&61{97}		

Alternative Operations Sections [&2A{42} – &3E{62}]

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 index 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 (&2A{42} – &32{50}) simply provide a *back-door* way to access the usual Sections where the index of the requested item is always represented in the first 2 bytes of the SysEx message. The subsequent bytes are exactly the same as those used for the usual Sections.

An extension to this concept is realised with Sections &38{56} – &3E{62} which facilitate user-defined *blocking* of parameter requests into one message. For example, rather than using 16 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, 3 data bytes must be sent for each parameter required: `<Item> <Data1> <Data2> <Data3>`. Even if the requested parameter does not require any data, 3 data bytes must always be sent; in this case the unused bytes should be set to zero. Upon success, 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 32. Note that these functions do not change the c*urrently selected* item.

Table 32: Message Format for Alternative Operations.

<Section>	<Data1>	<Data2>	<Item>	<Data…>	Description
&2A{42}	0–127(MSB)	0–127(LSB)	see Table 13	see Table 13	Program Manipulation by index. Index = LSB+128×MSB
&2C{44}	0–127(MSB)	0–127(LSB)	see Table 16	see Table 16	Multi Manipulation by index. Index = LSB+128×MSB
&2E{46}	0–127(MSB)	0–127(LSB)	see Table 18	see Table 18	Sample Tools by index. Index = LSB+128×MSB
&32{50}	0–127(MSB)	0–127(LSB)	see Table 22	see Table 22	Multi FX Control by index. Multi Index = LSB+128×MSB
&38{56}[a,b]	0–127(MSB)	0–127(LSB) <Data3>=1–99	see Table 9	see Table 9	*Blocked* Keygroup Zone By Index. (Program Index = LSB+128×MSB, Keygroup Index = <Data3>.)
&3A{58}[b]	0–127(MSB)	0–127(LSB) <Data3>=1–99	see Table 11	see Table 11	*Blocked* Keygroup By Index. (Program Index = LSB+128×MSB, Keygroup Index = <Data3>.)
&3C{60}	0–127(MSB)	0–127(LSB)	see Table 13	see Table 13	*Blocked* Program By Index. Index = LSB+128×MSB
&3E{62}	0–127(MSB)	0–127(LSB)	see Table 16	see Table 16	*Blocked* Multi By Index. Index = LSB+128×MSB

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