

SoundDiver

Universal Module

Version 2.0.2 – State: August 4, 1997

Adaptation Programming Manual

Also applicable for SoundSurfer

Universal MIDI Librarian Management
and Editor System

Windows 95 and NT 4.0

Mac OS

Atari ST, STE, TT and Falcon



Distributor



Emagic Soft- und Hardware GmbH
Halstenbeker Weg 96
D-25462 Rellingen
Germany

<http://www.emagic.de>

All trademarks are property of their individual owners. »SoundSurfer« and »SoundDiver« are registered trademarks of EMAGIC Soft- und Hardware GmbH.

Credits

The contributors to the Universal Module and its programming manual are:

Concept, programming, manual, layout, and editing	Michael Haydn
Coordination	Andreas Dedring
Beta-testing and suggestions	Luca Anzilotti, Dietmar Belloff, MaBu, AJ Command, Michael Cretu, Andreas Heggendorf, Kurt Hofmann, Ueli Karlen, Dirk Karsten, Thomas Kern, Thomas Ker- schbaum, Pit Löw, Gerhard Mannal, Siggi Müller, Heinz Naleppa, Malte Rogacki, Christian Roth, Marc Schlaile, Wolfgang Schmid, Kristian Schultze, Thomas Siebert, and many others
Support gratefully acknowledged	CML, Dynacord, E-mu Systems, Ensoniq, Kawai, Klemm, Korg, MIDITEMP, music shop, Musik Meyer, Roland Deutschland, Kristian Schultze, Soundware Audio Team, Waldorf, Yamaha Europa

Contents

Chapter 1

Introduction	17
About this manual	17
Conventions	18
Terms	18
Keyboard shortcuts	18
The SoundDiverBox BBS	18
How to contact the SoundDiver programmers and Adaptation authors	19
Want to have your Adaptation included with SoundDiver?	19
Want to become a SoundDiver programmer?	20

Chapter 2

Basics	23
Computer technology basics	23
Numbering systems and their display	23
Hexadecimal display	23
Binary display	24
Octal display	25
Bits, Bytes, Words, and Longs	25
ASCII	25
SysEx messages	26
Global structure, manufacturer ID	26
Model ID, device ID	27
Command ID	28
SysEx limitations	28
Using SysEx	29
Structure of a device	29
SysEx message types	29
Transmission formats	30
7 Bit	30
Nibbles	30
ASCII Hex	31
8x7 Bit packed	31
Alesis Quadraverb	31

7+1 Bit and 1+7 Bit	31
7 Bit and 1+7 Bit mixed	32
Devices with word-oriented organization	32
2 times 7 Bit	32
Sample Dump Standard	32
Checksum formats	33
Roland SysEx	34
Handshake protocols	35
Parameters	36
Dumps and parameters	36
Transmission formats and parameter access	37
»big endian« vs. »little endian«	37
Encoding of negative numbers	38
Encoding of text	39
Parameter Change messages	39
SysEx implementations - Poetry and Truth	40

Chapter 3

Tutorials	43
------------------------	-----------

Bank manager Kawai K1	43
Creating a new Adaptation	44
Global parameters	44
Initialization message	46
Defining the Scan function	46
Defining the data types	47
Defining the bank drivers	48
Single Dump	49
Single Request	51
Bank Dump	52
Bank Request	53
Further Single banks	54
Cartridge banks	54
Edit buffer	55
Multi banks	56
Multi edit buffer	57
Bank manager Roland D-110	57
Creating a new Adaptation	58
Global parameters	58
Defining the Scan function	59
Defining the data type	60
Defining the bank driver (Tone Temporary)	60
Defining the bank driver (Internal Tones)	62

Defining a Conversion table	63
Generic mixer	64
Designing the MIDI mixer	65
Creating a new Adaptation	65
Making adjustments in the Adaptation editor	65
Global data	66
»Data Type« block	67
»Bank Driver« block	68
Creating the controls	69
Sliders	69
Assigning a MIDI message	70
Input with a MIDI keyboard	70
Manual input	71
Switches	71
Rotary knobs	73
Copying objects (»Copy and Paste«)	74
Graphical appearance	76
Saving the Adaptation	77
DX7 bank loader	77
Installing a new Adaptation	78
Global settings in the Adaptation editor	78
Defining the data type for the Edit Voice	79
The Bank Driver for Edit Voice	80
The Dump string for Edit Voice	81
The Request string for Edit Voice	83
Defining a 32-Voice Bank	84
The MIDI Strings for Internal Voices	85
DX7 editor	86
Basic requirements	86
Numerical Value Objects	87
Envelopes	89
Global settings for envelopes	90
Defining the envelope points	91

Chapter 4

Reference – Memory Managers

Structure of an Adaptation	95
MIDI strings and pseudo bytes	95
Status bytes	95
Data bytes	96
Pseudo bytes	96
VAL – Parameter value	96

MEM – Memory occupied by parameter	97
SIN – Single dump data	97
BNK – Bank dump data	97
EN# – entry number in a bank	98
CHK – Checksum	98
SUM – Sum up from here	99
PAU – Pause	99
[– Loop start,] – Loop end	100
TRA – data size (transmitted bytes)	101
STO – data size (stored bytes)	101
FRA – Fractional dump data	102
SKI – Skip dump data	102
RES – Reset dump data pointer	102
Creating a new Adaptation	103
Adaptation editor	103
Window title	103
Fade in/out gadget	103
The cursor	104
Selection and insertion point	104
Selecting a definition block	105
Setting the insertion point	105
Global Edit menu	106
Undo	106
Redo	106
Cut	106
Copy	106
Paste	106
Clear	106
Select All	107
Local menu	107
New data type	107
New address mapping table	107
New bank driver	107
New conversion table	108
Save	108
Export names	108
Info line	109
Editing MIDI strings	109
Global parameters	110
Manufacturer	110
Model name	111
Device ID offset	113
Device ID Min/Max	113

Device ID +1	113
Thru Channel = Device ID	114
Icon	114
Input status enable	114
Default Timeout, Default Send Pause, and Default Play Delay	114
Roland SysEx	116
Roland Model	116
Author	117
Card names	117
Global MIDI Strings	117
Initialization	118
Failure Response	118
Device Scan	118
Universal SysEx Device Inquiry	118
Request / Answer message pairs	120
Use for Scan	120
Special cases	120
Mode of operation	121
Data type	122
Type name	122
Data size	123
Entries with variable size	123
Name size	125
Name offset	125
Name format	125
Parameter Send Pause (SoundDiver only)	126
Init values	127
Name bytes contain data	128
Address Mapping	128
Bank driver	132
Common bank parameters	133
Bank name	133
Data type	133
X	133
Y	134
# of entries	135
# of rows	135
Bank numbering	136
H/V title	138
Transmission format	138
Card switches	142
Editable	142

Memory location	143
ROM location	144
Use for Scan	144
Request regularly	144
Default Names	145
Program Change detection	146
CHAN – MIDI channel and master switch	147
BANK-MSB	148
BANK-LSB	148
FIXED	148
OFFSET	149
Standard parameters	149
Checksum type	149
EN# Offset	150
EN# Format	150
Roland parameters	151
Packet Size	151
Mode	152
Address Base	152
Distance	153
No Request	153
Aligned	154
Bank driver MIDI strings	154
Single Request	154
Single Dump	154
Bank Request	155
Bank Dump	155
Before Request/Dump	155
After Dump	156
Global advice on requests and dumps	158
Conversion tables	160
Structure of a Conversion table	160
Creating a new conversion table	160
Conversion steps	161
Transfer	161
Initialize	162
Loop start	162
Loop end	163
Jump	163
Notes on the mode of operation of conversions	163
Examples	164
File conversion	165
Using SoundSurfer Adaptations with SoundDiver and vice versa	165

Converting Windows or Atari Adaptation files to Macintosh and vice versa	166
Windows/Atari to Macintosh	166
Macintosh to Windows/Atari	167
Atari to Windows	167
Windows to Atari	167
Converting Polyframe Adaptation files	168
Windows/Atari	168
Mac	169
Help files	169
Polyframe help files	169
Atari to Macintosh	170

Chapter 5

Reference – Editors 171

The concept	171
-------------------	-----

Creating a new Editor	172
-----------------------------	-----

Object operations	172
-------------------------	-----

Creating new objects	173
----------------------------	-----

Selecting objects	174
-------------------------	-----

Selecting a single object (deselecting all other objects) ..	174
--	-----

Selecting additional objects	174
------------------------------------	-----

Deselecting a single object	174
-----------------------------------	-----

Selecting objects with the »rubber-band«	174
--	-----

Deselecting all objects	175
-------------------------------	-----

Moving objects	175
----------------------	-----

Changing the size of objects	175
------------------------------------	-----

Copying objects	176
-----------------------	-----

Opening the object editor window	176
--	-----

»Edit« menu functions	176
-----------------------------	-----

Undo	177
------------	-----

Redo	177
------------	-----

Cut	177
-----------	-----

Copy	177
------------	-----

Paste	178
-------------	-----

Clear	178
-------------	-----

Select All	178
------------------	-----

Toggle selection	178
------------------------	-----

»Adaptation« menu functions	179
-----------------------------------	-----

Binary View	179
-------------------	-----

Layout Mode	180
-------------------	-----

Grid Snap	180
Object Snap	180
»New object« submenu	181
Open Object Editor	181
Snap to Grid	181
Flatten	181
Edit Adaptation	181
Save Adaptation	182
Object editors	182
Editing several objects simultaneously	182
Parameters common to several object types	183
Border (switch)	183
Fill (switch)	183
Border (flip menu)	183
Fill (flip menu)	184
Color (flip menu)	184
Large/Medium/Small	184
Flip Menu (switch)	185
Shadow	185
Inverted (display)	185
Minimum/Maximum	185
X, Y, W, H	186
Format	186
0 Offset	188
Bit field definition	188
LS Bit / # of bits	188
Expanded bit field definition	189
# of skipped Bits	189
Skipped LS Bit	189
2's complement / Sign magnitude	190
Order	191
Mem.Offset	191
Message	191
Name	192
Transmission Format	192
.....	193
Transmission formats »Controller«	194
Transmission formats »Controller, integer steps«	194
Transmission formats »ASCII Hex HL« and »ASCII Hex LH«	195
Transmission format »ASCII Decimal«	195
Transmission format »ASCII Dec., 0-terminated«	196
Transmission formats »1+7 Bit HL« and »7+1 Bit LH«	196
Inverted (transmission format)	197
Text/Box	197
Inverse	197

centered/leftalign/rightalign	197
Text	198
Image objects	198
User	198
Zoom	198
Grid	199
Functions	199
All White	199
All Black	199
Shift Left/Right/Up/Down	199
X, Y, W, H	199
Arrow	199
Direction	199
Thickness	200
Numerical values	200
Format	200
Text values	201
Minimum/Maximum	201
Text Length	201
Text input field	201
Fill menu	202
Sliders	202
Handling	202
Format	202
Type	202
Rotary knobs	203
Type	203
Switches	203
Style	204
Send immediately	204
Text	204
Function	204
Using switches to jump to screensets	204
Envelopes	205
Global parameters	206
Range	206
Help Lines	206
Envelope points	206
Selection column	206
Help Lines	206
Object Link	206
Reciprocal	207
OP (Operator) and Const	207
Positioning	208



Keyboard and Key/Velocity windows	208
Global parameters	209
Mode	209
Keyb H	209
Range	210
Direction	210
Labels	210
Keyboard	210
Value limitation	210
Keyboard range, Velocity range	211
Object Link	211
Reciprocal	211
OP (Operator) and Const	211
Positioning	211
Parameter Changes	212
Principle	212
Pseudo bytes	212
VAL – Parameter value, MEM – memory used for	
Parameter	212
EN# – Entry number in bank	213
SUM – Sum up from here	213
CHK – Checksum	214
PAU – Pause	214
[and] – »Repeat« feature	214
Transmission details	214
Roland mode specific notes	215
Automatic analysis of MIDI messages (Analyze)	216
How Analyze works	216
How to create good editors	217

Chapter 6

The SSHC help compiler

Requirements	219
Command shell	219
Text editor	219
Additional tools	220
Installing SSHC	220
Windows 95	220
Macintosh	221
Atari	221
Tutorials	221
Creating an SSHC help source file	221
Starting SSHC directly	222

Starting SSHC from a command shell	222
Format of source files	223
Platform indicator	224
Control characters	225
\f (Form Feed)	225
\n (Newline)	226
\r (Carriage Return)	227
\! (Exclamation Icon)	227
\e (eg Icon)	228
\i (Info Icon)	228
\m (Mouse Icon)	228
How to use icons correctly	228
\p (Product name)	228
\l (Listen to MIDI Icon)	229
\w (Word Wrap)	229
\ (Backslash, Space)	229
\\ (Backslash, Backslash)	230
\/ (Backslash, Slash)	230
\(and \) (conditional compiling)	231
Example for correct usage	232
Enumerations	232
Tables	232
Icons	232
Conventions for help files	233
Standard keywords used by SoundDiver/SoundSurfer ...	234
<model name>	234
»Installation«	234
»Scan«	234
»MIDI«	235
»SysEx Communication Error«	235
»Memory Manager«	235
<data type>	235
<bank name>	235
»Device Parameter box«	236
<Data type> »Editor«	236
<parameter name>	236
<parameter group name>	237
»Conversion«	237
»Credits«	237
»Copyright«	238
Notes on how to write good help files	238
Transcribing printed manuals	238
Listen to MIDI	238

Kopiermöglichkeiten	238
Cross-references	239
Repeated characters	239
Indentions	239
Parameter descriptions	240
Running SSHC	241
Windows	241
Macintosh	241
Menu bar	241
 > About SSHC	241
 > Help	241
File > Compile	241
File > Remove help	241
File > Preferences	242
Options file	242
Apple Events	243
Atari	243
SSHC options	243
-h (+help)	243
-v (+verbose)	243
-l (+light)	244
-m (+make)	244
-n (+no_compression)	244
-s (+standard)	245
-r<offset>	245
-o<output file>	246
<file name>	247
SSHC error and warning messages	248
SSHC's mode of operation	248
Mode of operation of SoundDiver's help system	251

Chapter 7

Menu overview	253
Local menus Memory Manager	253
Adaptation	253
Local menus Adaptation editor	253
Adaptation	253
Local menus Editor window	254
Adaptation	254
Global menus Editor window	254
Edit	254

Chapter 8

Key commands 255

Key command symbols 255

Key commands 256

Chapter 9

Mouse operation 257

Adaptation editor 257

Editor window 257

Chapter 10

SSHC options 259

Chapter 11 Trouble Shooting 261

Error messages 261

Problems in use 264

 Using existing Adaptations 264

 MIDI communication, driver definition 265

 Editor definition 266

SSHC error messages 267

SSHC warning messages 268

Problems when testing help files 269

Chapter 12

Bibliography 271

English 271

German 272

Chapter 13	
Manufacturer IDs	275
Chapter 14	
Conversion table	283
Chapter 15	
Glossary	287
Chapter 16	
Index	293

Chapter 1

Introduction

1.1 About this manual

SoundDiver 2.0 already provides Modules and Adaptations supporting more than 330 models. However, if one of your devices is not supported, you can create an own adaptation with the Universal Module and this manual.

Since some technical know-how is needed, and not every SoundDiver user has time or feels like doing this, this manual is separated from the SoundDiver manual. Thus, the SoundDiver manual stays pleasantly thin, and we save paper.

This manual is divided into four sections:

- the basics section describes how to use System Exclusive
- several tutorials make the basic way of working accessible to you
- the reference section has an answer to any question you might have about the Universal Module
- the appendix has several sections for reference

If you are not yet an experienced user, you find bibliography recommendations.

Notes:

- This manual exists as a PDF file only. You need Adobe Acrobat Reader which you can download for free from **www.adobe.com**. Version 3.0 is recommended.
- This manual is applicable for SoundSurfer and SoundDiver. The sections on creating editors however can be skipped by SoundSurfer owners.
- Every time the manual refers to SoundDiver, this includes SoundSurfer as well, except in the editors section (see above).

1.2 Conventions

Terms

There is only an English version of this manual. Your copy of the Universal Module might be localized to another language. Please translate the native terms to English.

If a term might not be clear to you, you can look it up in the glossar in the appendix of this manual. All terms used in the Universal Module (e.g. request, bank driver) are explained there.

Keyboard shortcuts

This manual shows the keyboard shortcuts of the Macintosh version of SoundDiver. A comparison with the appropriate shortcuts of the Windows and Atari versions can be found in the SoundDiver manual.

1.3 SoundDiver support on the Internet

We are currently setting up a download area for SoundDiver on our WWW server. Please have a look at www.emagic.de/english/updates/index.html.

1.4 The SoundDiverBox BBS

For those who did not yet realize it: there is a dedicated SoundDiver bulletin board system (BBS). The phone number is

+49-4101-495-190 (analog modem)

+49-4101-495-192 (ISDN X.75)

You can find detailed information on how to logon and how to use it in the file »SoundDiver 2.0 update info«.

In the SoundDiverBox, you can:

- download the newest versions of SoundSurfer/SoundDiver, the Universal Module, all other Modules and Adaptations as well as Libraries and new Adaptations
- join the numerous SoundDiver forums. You can ask questions as well as read other users' questions and the corresponding answers. Use BulkRate (a shareware version is downloadable for free) to read and reply to these forums offline.
- upload your own Adaptation or Library files.

1.5 How to contact the SoundDiver programmers and Adaptation authors

You can contact me, Michael Haydn

- in the Internet: **mhaydn@emagic.de**
- in the SoundDiverBox: by writing me mail.

Note:

- I would prefer if you would write messages/questions of common interest into the »SoundDiver« forum. The reason is that many users often ask the same questions, and I'm tired giving the same answers all the time.
- in CompuServe: **100520,1532** or **Michael_Haydn**

Note:

- please understand that I have a lot of work to do and thus try to minimize redundant user communication, so please only ask me questions on Adaptation programming. Questions on common SoundDiver operation or planned features/Modules/Adaptations are answered by your local Emagic distributor or Emagic Germany:

Hotline: **+49-4101-495-110**

Fax: **+49-4101-495-199**

SoundDiver Module programmers as well many of the Adaptation authors can be contacted in the SoundDiverBox. Use FirstClass's function »Directory«.

1.6 Want to have your Adaptation included with SoundDiver?

We at Emagic are always interested in new Adaptations. Adaptations which

- work without problems
 - have a user interface and editor layout which goes conform with the standards mentioned in this manual and
 - come with English and German Help source files
- will be added to the SoundDiver and SoundSurfer packages, and you will get some money (between DM 50 and DM 400, depends on complexity) for it. Please ask Andreas Dedring at Emagic (adedring@emagic.de) for details.

1.7 Want to become a SoundDiver programmer?

If you want to get more serious with SoundDiver, you can become a freelance SoundDiver Module developer. However, there are some conditions:

- You must have sound programming knowledge in ANSI C.
- You should be familiar with one of the following development environments:
 - PureC 2.0 (Atari)
 - Think C 7.0 or higher (Macintosh)
 - Metrowerks CodeWarrior (Macintosh)
 - Microsoft Visual C++ 4.0 or higher (Windows 95 or NT 4.0)
- You must have Internet email access with binary transfer or at least a modem for accessing the SoundDiverBox.

- You should already be familiar with operating SoundDiver and programming Adaptations, as many techniques are quite similar in Adaptations and Modules.
- You should have some experience with MIDI programming and perhaps already have created a bank loader or editor software.
- You should have at least 10 hours a week free for programming SoundDiver Modules.
- You should be reliable concerning deadlines, quality assurance and beta testing.
- German language is needed at this time, since programming documentation is in German. However, there might be English documentation in the future.
- You will have to supply your Module's source code, since we will port it from your platform to the others. Additionally, you will have to write the German and English help files yourself (translation by a professional is too expensive).

The pay you can earn for your programming efforts bases on the code size. Please ask Chris Adam at Emagic (cadam@emagic.de) for details.

-

Chapter 2

Basics

2

2.1 Computer technology basics

So as to be able to work with MIDI implementations and to create powerful Adaptations, you need a basic knowledge of computers and of MIDI technical matters. This section of the manual is designed as a glossary for users having little or no experience with programming computers.

Numbering systems and their display

We work with decimal numbers every day and are familiar with them. They are used with computers as well. Decimal numbers work with a *place* system with the *basis* 10, i.e. after the *digits* 0 to 9 (thus 10 digits), a second digit is used. To separate them from numbers displayed in other systems, decimal numbers are sometimes shown with a »d« placed after (e.g. 63d). More seldom is an index 10 (e.g. 63₁₀) which is useful for preventing confusion with hexadecimal numbers (see below).

Besides decimal numbers, other numbering systems are used.

Tips:

- The Universal Module provides a conversion display in the local menu bar of the Adaptation and Object editor windows.
- You can find a conversion table in Appendix H *Conversion table* from page 283.
- There are reasonable pocket calculators available which are able to convert between different numbering systems (e.g. Casio fx-115D)

Hexadecimal display

Here, the basis is not 10, but 16. Each digit has 16 different values. Since the arabic alphabet knows only ten digits, the values 10 to 15 are denoted with A to F (sometimes also a to f). Then, the next digit is used: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, ... and so on.

Hexadecimal numbers are marked with a \$ placed first (e.g. \$3F) or H or h placed after (e.g. 3Fh respectively 3FH). An index 16 placed after (e.g. 3F₁₆) or a »0x« placed first (e.g. 0x3F) are more rarely used. The latter format is used in the programming language »C«.

This manual, as well as the Universal Module uses exclusively the format \$3F.

Hexadecimal numbers are often denoted as hex numbers, sometimes sedecimal numbers (because this is the correct name for 16 used in Greek; »hexadecimal« is a wrong term which is however most frequently used).

Advantages:

- A byte (see below) can be displays with only two digits
- Each digits displays a nibble (see below)
- Single bits (see below) can be extracted more easily by experienced.

Drawbacks:

- Hexadecimal numbers can be confused with decimal numbers if they are not marked and contain no digits between A and F.

How to convert hexadecimal to decimal display:

- Take the first (most significant) digit. If it is A to F, take the corresponding value 10 to 15 instead.
- If there are further digits, multiply the present result by 16 and add the next hexadecimal digit according to step 1.
- Repeat step 2 until all digits are done.

Binary display

This display is a digit system as well, but with a basis of 2. For each digit, there are only two possible values (0 and 1). Binary digits are not marked in most cases, but sometimes with a % placed first (e.g. %00101100) or an index 2 placed after (e.g. 00101100₂). As you can see, leading zeros are normally not omitted, but filled up to eight digits.

Advantage:

- Binary numbers show the single digits directly. Computers work internally with the binary display directly, since digital memory units can store only two states (precisely 0 and 1).

Drawbacks:

- difficult to read and enter
- needs a lot of space

How to convert binary to decimal display:

- Take the first (most significant) digit.

- If there are further digits, multiply the present result by 2 and add the next binary digit according to step 1.
- Repeat step 2 until all digits are done.

Octal display

Octal numbers are not common to the MIDI world and are mentioned only for the sake of completeness. This display is a digit system with the basis 8, so there are the possible values of 0 to 7 for each digit. Octal numbers are marked with a `\` placed first in the C programming language.

Bits, Bytes, Words, and Longs

A bit (acronym for »binary digit«) is the smallest possible information unit. There are only two possible conditions, 0 and 1.

A byte is the combination of eight bits. So there are $2^8 = 256$ different possible values. The bits of a byte are numbered from 0 to 7. The most significant bit has the significance of $2^7 = 128$ and is also called bit 7.

A word is the combination of 2 bytes in most cases, thus 16 Bits. So there are $2^{16} = 65536$ different values. However, some computer systems define a word as 32 or more Bits. The bits of a 16-bit word are numbered from 0 to 15. The most significant bit of a word (bit 15) has the significance of $2^{15} = 32768$.

A long word is the combination of 4 bytes. Long words (also called simply longs) are very rare for most MIDI devices, but usual in samplers, e.g. for defining loop points or sample lengths. The bits of a long word are numbered from 0 to 31. The most significant bit of a long word (bit 31) has the significance of 2^{31} .

The following applies to bytes, words, and long words: the least significant bit (having the significance of 1) is called bit 0 or LS Bit. The most significant bit is often abbreviated with MSBit.

ASCII

ASCII is the abbreviation of »American Standard Code for Information Interchange«. This code is used all over the world to encode lower case and upper case letters, digits and punctuation marks. You can find a table in Appendix H *Conversion table* from page 283.

2.2 SysEx messages

The MIDI commando which is least paid attention to - or maybe most unpopular - is »System Exclusive«, also abbreviated with »SysEx«. However, if you occupy yourself with it, you will explore a new land of unthought-of possibilities. We will explain how to use SysEx with the Universal Module with some practical examples.

Each MIDI message contains of a so-called status byte and a certain number of so-called data bytes. So the message »volume of channel 5 to 100« contains of the bytes 180, 7, 100 (hexadecimal \$B4, \$07, \$64). In a status byte, the most significant bit (bit 7) is always set (binary 1xxxxxx), whereas it is always cleared in a data byte (binary 0xxxxxx). That is, status bytes are always greater or equal 128, while data bytes are always smaller than 128.

»System Exclusive« first means that every MIDI manufacturer may extend the MIDI standard to his own needs. The status byte »SysEx« (240 decimal, \$F0 hexadecimal) and its counterpart »End of Exclusive« (EOX, 247 decimal, \$F7 hex) build a formal »wrapping«, so that the beginning and end of a SysEx message can be detected easily in the MIDI data stream. This is necessary, since System Exclusive messages may be of any length in contrary to all other MIDI messages.

However, EOX is not necessarily needed - some older devices (e.g. Sequential Prophet V) don't send it. This is possible because the end of a SysEx message is defined not only by EOX, but also by any other status byte (except Real Time status bytes).

Global structure, manufacturer ID

To prevent that a MIDI device processes by mistake a SysEx message that is intended for a device of a different manufacturer, the SysEx status byte is followed by a manufacturer identification (manufacturer ID). It is normally made up of one, sometimes of three bytes. The manufacturer ID is assigned by the IMA (International MIDI Association). So, an imaginary message now says »To everyone made by Sukiyaki!«. The meaning of the remaining bytes, as well as their number, is proprietary to the manufacturer.

Besides manufacturer-proprietary messages, there are some standardized SysEx messages, having the manufacturer IDs 125 (\$7D) to 127 (\$7F):

Table 1 Standardized SysEx messages

ID	Sub ID 1 / description
\$7D	Non-Commercial SysEx: for research purposes
\$7E	Non-Real-Time SysEx
\$01	Sample Dump Header
\$02	Sample Dump Data Packet
\$03	Sample Dump Request
\$04	MIDI Time Code (MTC) Set-Up
\$05	Sample Dump Extensions
\$06	Inquiry Message
\$7C	Wait
\$7D	Cancel
\$7E	NAK (No Acknowledge)
\$7F	ACK (Acknowledge)
\$7F	Real-Time SysEx
\$01	MIDI Time Code (Full Message / User Bits)

Another special meaning has the manufacturer ID \$00. In this case, the manufacturer ID actually consists of three bytes, namely this zero and the following two bytes. This was necessary, because the possible 127 manufacturer IDs were not enough. Together with the 3-byte option, $127 + 128 * 128 = 16511$ can be defined, which is pretty much.

You can find a list of all known manufacturer IDs in Appendix G *Manufacturer IDs* from page 275.

Model ID, device ID

As virtually every MIDI manufacturer sells more than one model which are incompatible concerning SysEx, their SysEx messages must be separable from each other. For this purpose the »model ID« is used. It consists depending on the manufacturer of one up to eight bytes (e.g. Yamaha SY77: ASCII characters »LM 8101«). Now our imaginary messages says »To all DQ-71EX made by Sukiyaki!«.

On the other hand, to separate devices in a MIDI setup which are compatible concerning SysEx, most manufacturers use a »device ID«, also called »Device Number«, »Basic MIDI Channel« or »Global MIDI Channel«. In order to work, all compatible devices must have set device ID different by pair. Our message: »To the Sukiyaki DQ-71EX with ID 5!«.

Command ID

And finally, most manufacturers use the so-called »command ID« which determines what the message finally is about to do. Often, the command ID is combined with the global MIDI channel in one byte (e.g. DX7). Now, our message gets a meaning: »To the Sukiyaqi DQ-71EX with ID 5: here comes the Patchpresetmulti No. 1B-122!«.

This results in a step-by-step hierarchy:

Table 2 Hierarchy of SysEx messages

Hierarchy level	Example: Kawai K1 All Multi Data Dump (ext)		
SysEx	11110000	FOH	
Manufacturer	01000000	40H	Kawai
Model ID	00000000	00H	Synthesizer group
	00000011	03H	K1/K1m ID no.
Device ID	0000nnnn	0nH	Channel no.
Command ID	00000001	01H	1=ext
	01000000	40H	Multi

Model ID, device ID, and command ID need not necessarily be transmitted in this order, for example the device ID is the third byte in most cases (and not the fifth as in the above example).

SysEx limitations

There are virtually no limitations to the use of SysEx messages. However, there are two restrictions:

- Within a SysEx message, the most significant bit (bit 7) must not be used. Bit 7 separates - as described above - status bytes from data bytes. If there is data to be transmitted which uses bit 7, a special conversion must take place. See below for further details.
- MIDI is too slow for certain applications. As SysEx messages are many times longer than all other MIDI messages, this restriction has a particular effect in Real Time situations (e.g. when playing back a sequencer song). To transfer many hundreds of kilobytes (e.g. Samples), you should use SCSI instead.

2.3 Using SysEx

Structure of a device

Most of the MIDI compatible devices have a certain number of memory locations. Combined together, they are usually denoted as a »bank«. Many devices have more than one bank: internal, external (on a card or cartridge) and sometimes ROM banks (the latter cannot be changed).

When you recall a memory location, either by pressing more or less numerous buttons or by sending a Program Change message via MIDI, its contents is copied into an »Edit buffer«. The sound generation system accesses this edit buffer directly or indirectly so that each parameter change is immediately audible.

This structure bank – edit buffer is eventually repeated for several data types, e.g. for Programs, Multis, Micro Tunings and so on. Sometimes, the memory locations of a bank are identical with the edit buffers, i.e. the sound generation system accesses the memory locations directly (e.g. SY77 Pans and Micro Tunings, Akai S1000). In other cases there are multiple edit buffers which can be accessed simultaneously (e.g. Roland D-110, Waldorf microWave).

SysEx message types

Over the years, certain kinds of SysEx messages have crystallized because of this structure:

- Dumps: transmit the contents of certain memory locations (single dump), a whole bank (bank dump) or the whole memory (all data dump or bulk dump) of a device.
- Requests: request a dump. A device which receives a request normally answers with the corresponding dump.
- Parameter Change: changes a parameter in an edit buffer of a device. Although in most cases there is also a dump message for the edit buffer, a parameter change is much faster to transmit and process and therefore suitable as a musical expression tool - like Controller messages, especially when working with a sequencer.

Besides there are other messages, e.g. which transmit information about configuration and software version of a device or change the current mode.

Transmission formats

When transmitting dumps and parameter changes, you often get the problem that MIDI data bytes must not use the most significant bit (also called MSBit), this bit must always be 0, i.e. only seven of eight bits of a byte may be used freely. However, most of the MIDI devices have 8 bit memory, because this has been usual in computer technology for a long time. The problem arising from this fact called »how can I transmit 8 bit data using only 7 bits?« is unfortunately solved differently by the manufacturers.

7 Bit

The simplest method is to simply not use the MSBit, so that it can be omitted in the transmission (let's call this method »7 Bit«). However, this means that the parameters of a sound must have a maximum value range of 0 to 127 respectively -64 to 63. This is not sufficient for some parameters (e.g. sample number in Roland D-110), so that the range is expanded »artificially« by a second parameter (»Bank Select«).

In all other methods, the bytes of a data block are transmitted in several fractions, i.e. each byte is divided up in a certain way.

Nibbles

The easiest variant is to divide each byte »in the middle« to get two 4-bit halves. Those are called »nibbles« (or »nybbles«). The »upper half«, i.e. bits 4 to 7, is called »hi-nibble«, the bits 0 to 3 »lo-nibble«. Unfortunately, one could not reach an agreement which nibble is to be transmitted first, therefore there are two variants, called HL-nibbles and LH-nibbles upon the order.

As an example, let's consider the LH-nibbles transmission of the byte sequence 245, 137. These two bytes are displays as 11110101 10001001 binary (for binary code novices: $245 = 1 * 128 + 1 * 64 + 1 * 32 + 1 * 16 + 0 * 8 + 1 * 4 + 0 * 2 + 1 * 1$; $137 = 1 * 128 + 0 * 64 + 0 * 32 + 0 * 16 + 1 * 8 + 0 * 4 + 0 * 2 + 1 * 1$). Thus, the nibbles are 1111, 0101, 1000, and 1001. Because we use LH instead of HL transmission, each byte's nibbles are transmitted in reverse order. So the transmitted bytes are 00000101 00001111 00001001 00001000 (hexadecimal \$05 \$0F \$09 \$08, decimal 5 15 9 8).

As you can see, each byte in the device's memory must be transmitted as two bytes. Since in each transmitted byte three bits are unused (as only four of seven are used), the transmission time is unnecessarily long. Unfortunately, the MIDI manufacturers don't consider this as a major problem, so that even some of the newest models still use nibble transmission.

ASCII Hex

»ASCII Hex« is a variant of nibble transmission. It is used in some older Yamaha devices. The only difference is that the nibbles (in HL order) are not transmitted binary, but in ASCII characters of the 16 hexadecimal digit values. That means the values 0 to 9 are replaced by the ASCII digits »0« to »9« (ASCII code 48 to 57), the values 10 to 15 by the ASCII characters »A« to »F« (ASCII code 65 to 70). The only advantage is that the values can be read directly in hex code if the recording device (or the sequencer) shows the data in ASCII code. Our example would be 01000110 (»F«), 00110101 (»5«), 00111000 (»8«), 00111001 (»9«).

8x7 Bit packed

The high cost of time when using nibble transmission was recognized from Korg and Lexicon, and some of their devices work with the »8x7 Bit packed« method. It is quite complicated, but also the fastest possible. Here, the encoding is not done for each byte separately, but for a block of maximum seven bytes. From each of the seven bytes, our famous MSBit is »cut off«, and are combined in an extra byte (where the MSBit of the n-th byte is stored in the n-th bit of the extra byte) and transmitted. This extra byte is transmitted, followed by the »castrated« seven bytes. Thus, for seven internal bytes, only eight MIDI bytes have to be transmitted. Not a single bit of MIDI capacity is wasted.

Does this mean that this method allows only multiples of seven bytes to be transmitted? No, if there are less than seven bytes, only 1+n bytes are transmitted. The following example illustrates this: the two bytes 11110101 10001001 are transmitted as 00000011 01110101 00001001.

Alesis Quadraverb

The Quadraverb from Alesis uses a similar method. Here, seven 8-bit bytes are packed into a huge bit field and transmitted in 7-bit portions.

7+1 Bit and 1+7 Bit

Besides the method to split an 8-bit byte into two equal halves, as it is done with nibbles, there are of course other methods. One is to transmit the MSBit in bit 0 of an extra byte. Depending on which of the two

transmitted bytes is sent first, these methods are called 7+1 Bit or 1+7 Bit. The efficiency is by the way as bad as with nibbles.

7 Bit and 1+7 Bit mixed

And then there is a method that is used by Yamaha in newer models. Here, parameters which fit into seven bits, are transmitted in »7 Bit« format either, and 8-bit bytes are transmitted in two bytes (the MSBit in bit 0 of the first MIDI byte, then the remaining seven bits). This method is almost as fast as »8x7 Bit packed«, given there are only few 8-bit bytes in the data block. However, the big disadvantage is that the method depends on the parameters' value ranges and thus cannot be chosen »on the fly« in a Universal Editor program.

Devices with word-oriented organization

The progress in computer technology influenced the MIDI world in the last years. Now, some devices work with 16-bit architecture and 16-bit memory. The difference to conventional 8-bit devices is that a 16-bit value (having a maximum value range of 0 to 65535 respectively – 32768 to 32767) can be processed with a single memory access.

Some devices (i.e. those which don't allow a byte-oriented access to the memory) store all their parameters in 16-bit blocks (so-called words, e.g. E-mu Proteus, Roland U-20). Here, the same problem »how can I transmit 16-bit data with only 7-bit?« arises, and several solutions exist.

2 times 7 Bit

The first method is similar to the »7 Bit« method. Of the 16 bits in a word, only 14 are used. Each of the 14-bit words is transmitted in two 7-bit halves. Of course there are two possibilities again. However, currently only the LH order is used (E-mu Proteus).

The second possibility is the nibble method, again with two possible orders:

- Word HL nibbles: bits 15..12, 11..8, 7..4, 3..0
- Word LH nibbles: bits 3..0, 7..4, 11..8, 15..12

The latter is used by Roland U-20.

Sample Dump Standard

In Samplers, there are not only 8-bit and 16-bit value ranges, but also others. To transmit those values, there are manufacturer-proprietary formats (e.g. Ensoniq EPS 12-bit and 16-bit formats) as well as the stan-

standardized Sample Dump Standard (SDS) format. This format depends on the length of a sample word which may be between 8 and 28 bits. Each sample word is split up into several 7-bit fractions, and the seven most significant bits are transmitted first, and the bits in the last bytes are arranged »left-aligned«.

Table 3 Examples for Sample Dump Standard

<i>Format</i>	<i>in memory</i>	<i>MIDI transmission</i>
8 Bit	76543210	-7654321 -0-----
12 Bit	-----BA98 76543210	-BA98765 -43210--
14 Bit	--DCBA98 76543210	-DCBA987 -6543210
16 Bit	FEDCBA98 76543210	-FEDCBA9 -8765432 -10-----

Note:

- This format is not yet supported by the Universal Module.

Checksum formats

To recognize errors in the transmission of larger data amounts, many manufacturers use so-called checksums. This is a value which is calculated from the bytes to be transmitted by building a sum of these bytes.

The transmitting MIDI device (the sender) transmits this checksum immediately after the data within the MIDI dump message. While the transmission, the receiver calculates the checksum by the same way and compares it with the one transmitted by the sender. If there was a difference, the receiver »raises the alarm« and ignores the just received data. Some SysEx definitions (e.g. Roland handshake) lay down that the transmission is automatically repeated, so that the error isn't even recognized by the user.

Please note: even a correct checksum isn't a 100% guarantee for an error-free transmission. None of the following formats recognizes a difference between the number sequences 1,2,3 and 1,3,2, because always the same checksum comes out, because 1+2+3 is »unfortunately« the same as 1+3+2.

All methods have the following in common: after sending the dump message's header, a variable is deleted. Each transmitted byte is then

added to the variable. (An exception are the formats »LH Nibbles → LH« and »LH Nibbles → 7 Bit«: here, not the transmitted, but the bytes in memory are summed up. The Kawai K5 even sums up words, not bytes.) The resulting value is then encoded differently depending on the checksum format:

- 2's Complement: from the negative value of the sum, only the seven least significant bits are transmitted. The effect is that the sum of all data bytes and the transmitted checksum byte are zero in the seven least significant bits. This method is the most common used.
- 1's Complement: The one's complement means that all bits of the value changed (0 becomes 1 and vice versa). From the result, again the seven least significant bits are transmitted.
- Regular Checksum, LH Nibbles → 7 Bit: the seven least significant bits are transmitted directly.
- LH Nibbles → LH: the eight least significant bits of the sum are transmitted as LH nibbles, i.e. the checksum consists of two bytes.
- Kawai K1/K4: \$A5 (decimal 165) is added to the sum, the result negated (i.e. the negative value taken), and the seven least significant bits are transmitted.
- Kawai K5: the sum is subtracted from \$5A3C and transmitted in the format »LH Nibbles«. So the checksum consists of four bytes.

If you know other formats, please write us.

Roland SysEx

Roland has used a uniform format for SysEx transmission for some time. All Roland models since the S-10 as well as all devices from Boss and Rhodes use this format. It differs from the formats used by other manufacturers by the so-called »Address Map«. This means that the data accessible by SysEx isn't jump numbered consecutively, but placed in certain address ranges. For example, the D-110's Tone I-a11 is located in the address range 08 00 00 to 08 01 75. To send data, after the command ID »DT1«, the start address is given before the transmitted data. A request contains this address as well, then the length of the data to be requested. The advantages are obvious:

- There is no need to define separate Parameter Change messages. Instead, just a »mini dump« (normally containing just one byte) is

transmitted. This way, no parameter numbers have to be defined (which often cause lavish encoding and decoding functions).

Example: The name of a D-110 Tones is placed at the offsets 0 to 9. Thus, a »dump« of a single byte in address 08 00 09 changes the name's last character.

- Single dumps and bank dumps are just the same. Banks are arranged in the address map in a way that the single memory locations are located in consecutive address ranges.

Nevertheless, there is a maximum data size for dump message (usually 128 or 256 bytes). This results in the fact that long single or bank dumps are transmitted in several parts. There are two reasons:

- The denser the checksums are spread in a dump process, the safer they are.
- The receiver needs an input buffer where the incoming dump is first stored to validate the address and checksum before the incoming data is processed. If dump messages had random size, this buffer might not be large enough.

Notes:

- The length of a Roland address can be one to four bytes, depending on the device's complexity
- All addresses are given by Roland in the so-called »7 Bit Hex« format. It is similar to a normal hexadecimal number display, but with the difference that every second hexadecimal digit from the right may have a value range from 0 to 7 only (opposed to 0 to F). This results from the necessity that address statements are just normal SysEx data and thus bit 7 must not be used.

Table 4 Example: conversion from 7 Bit Hex to Hex

7 bit hex	0			8			0			1			7			F		
Binary	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1
Hex	0			2			0			0			F			F		

- This conversion is displayed by the Universal Module in the info line of the adaptation and object editors.

Handshake protocols

Some manufacturers use for some data transmittals a so-called handshake protocol. This means that sender and receiver communicate in

both directions similar to a phone call. The »conversation« typically looks like this:

Table 5 Example Handshake

Sender	Receiver
»Hi, I'd like to transmit you my sequencer memory. You will need 31548 bytes to store it.«	»OK, go ahead.«
»Here is the first data set: wow wow wow yippiyo yippiexxx«	»OK, got it alright, please continue.«
»Here is the second data set: hollaraxliööö«	»Oops, there was an error in the data.«
»Well then, I'll try again: wow wow wow yippiyo yippiyeah«	»OK, now it was correct.«
»Excellent, that's it. See ya.«	

As mentioned, Handshake protocols are a bit more lavish and thus are suitable for large data transfer sizes.

Besides a Handshake variant of the Sample Dump Standard (called »Closed loop«), there is a Handshake version of Roland SysEx communication.

Casio made with their CZ series a »Handshake« format which does not go conform with the MIDI standard. As soon as the »caller« started the SysEx communication with the SysEx status byte, the Handshake process is done by simply sending data bytes (without leading SysEx and trailing EOX status bytes) from both sides. This gets many MIDI programs into trouble, since the MIDI standard defines data bytes without a status as invalid and to be ignored.

Note:

- The Universal Module currently supports Handshake communication only in the Roland mode.

2.4 Parameters

Dumps and parameters

Until now, we talked about dumps without taking care of their meaning, their contents. This is ok as long as you just want to create a simple bank manager (except the display of names). However, as soon as you want to write an editor for a Sound (or Multi, Map, Table...), you have

to know in which order the single parameters are arranged in the dump.

Here are several possibilities which mainly depend on the value range and thus the number of bits the parameter uses. The simplest form is to store each parameter in a single byte (e.g. Roland D-50, D-110) or in a word (e.g. E-mu Proteus). When there are many parameters with very small value ranges and thus need only very few bits, a lot of memory is wasted with this method. In view of today's cost for memory chips, this shouldn't be so important anymore. However you have to consider that a dump then transmits a lot of bits which are unused. So the transmission time is increased without need.

Especially with switches which have only two states and thus need only one bit, this method is very inefficient. This is why often several parameters are combined to one byte (or word in word-oriented devices), e.g. parameter 1 in bit 0, parameter 2 in bits 1 to 2, parameter 3 in bits 3 to 6. This is called a »bit field«. To define a bit field element, there are only two additional pieces of information: the position of the parameter's least significant bit and the number of bits the parameter occupies.

Transmission formats and parameter access

Why is it necessary after all that one has to select a transmission format when creating an Adaptation? Of course you could save the incoming data as it is incoming via MIDI. This is ok for a plain dump manager and is common practice when using a sequencer as a SysEx librarian. However if you want to access single parameters, it is necessary that the single bits of the parameters are arranged in an order so that the parameter access can be defined quite easily. Supposed a parameter is placed in bits 0 to 4 of a byte, and the dump is transmitted in HL nibbles. Then you would have to define »bit 4 of the parameter is placed in bit 0 of byte n, and bits 3 to 0 of the parameter are in bits 3 to 0 of byte n+1«. However, if you decode the incoming HL nibbles, you just have to give »the parameter's least significant bit is at bit 0 of byte n, and the parameter uses 5 bits.«

»big endian« vs. »little endian«

Besides a correct transmission format, there might be another hurdle to clear over for an easy parameter access in a dump. It stems from the fact that there are different view of how to arrange the bits of a 16-bit

word in computer memory. As is well known, a word needs two bytes in memory. Each of the bytes is accessible by an address.

In so-called »big endian« machines (including the Motorola 680x0 CPU's and thus Atari ST/TT, Apple Macintosh, and Amiga), the bits 15 to 8 of a word are arranged in address n , and bits 7 to 0 are at address $n+1$ (therefore the name: the number's »end« is located at the »bigger« address).

In »little endian« machines, it's the other way around. Little endian machines are all Intel 80x86 processors, and thus all »industry standard« PCs.

When a dump is transmitted via MIDI, and sender and receiver belong to one of those »endian« groups each, a direct 16-bit access to a parameter will fail. The receiver would instead have to read the bytes separately and exchange their significance in order to achieve the correct bit order.

Examples for the few MIDI devices with »little endian« order are Lexicon LXP-1 and PCM-70.

Encoding of negative numbers

Parameter may often have negative values. In this case, the minus sign must be encoded into the binary display of the number in a certain way. There are several methods which might even be used by the same devices alternately:

- 2's complement: a negative number can be recognized by the fact that the MSBit is set (thus 1). To get the absolute value, you have to »toggle« all bits (0 becomes 1 and vice versa) and add 1. The maximum value range of an 8-bit byte is -128 to +127.

Example: the number -5 is encoded as 11111011 in a byte.
»Toggling« results in 00000100 (decimal 4). 1 added results in 5.

- Sign Magnitude: here, the MSBit is set as well if the number is negative. However, the absolute value is directly contained in the resulting bits. The maximum value range in an 8-bit byte: -127 to +127.

Example: -5 is encoded as 10000101.

- Binary Offset: the binary value range is shifted by a constant value so that no negative numbers can occur. To display the »real« value correctly, simply this constant has to be added.

Example: a parameter with the value range of -64 to 63 is en-

coded as 00000000 to 01111111 (0 to 127). The »offset« is -64 and thus corresponds the minimum value.

Encoding of text

Many synthesizers allow to assign names to their sounds. The ASCII code (see section *ASCII* on page 25) is used by the most MIDI devices. However, some devices use own codes, like Roland D-50, Alpha Juno 1/2 and the Oberheim Matrix series. In order to show and edit their names, a special conversion table is needed.

Parameter Change messages

Besides dump messages, many MIDI devices provide the possibility to transmit single parameter changes. The main difficulty arising here is how to state the parameter which is to be changed. It is related with the above mentioned two possibilities how to arrange parameters in a data block. If there is only one parameter per byte, each parameter can be addressed by the byte's offset in the block. If this is not the case, there are several other possibilities:

- always the whole byte is transmitted. The drawback is that the current value of the other parameters arranged in the byte must be known in order not to change them by mistake. You will have this problem in simple universal editors like Notator RMG or Cubase DMM. In the Yamaha DX7, this problem has been »solved« by defining two different formats for the same data type: one for the edit buffer (VCED, here each parameter is arranged in a separate byte) and one for memory locations (VMEM). However, a new problem results with this technique: the different formats must be converted.
- a code is defined which allows to access each parameter individually (Roland calls this method »Individual Parameter Changes«. The code consists of special addresses). However, this increases cost considerably, since extensive tables must be defined or even the message for each parameter must be defined separately.
- besides the offset, a mask is transmitted which defines which bits must not be changed. This method was used the first time in Yamaha SY22 and TG33. The only disadvantage is that the message gets slightly longer.

2.5 SysEx implementations - Poetry and Truth

If even the reading of a synthesizers owner's manual sometimes puzzles you, you wonder if it wouldn't had been better to buy a piano instead when you read the MIDI documentation at the latest. Although the SysEx implementations of today's MIDI devices get larger and larger, there are still very few devices which optimally support creating editor software. I want to give you an overview so you can better judge devices in this respect.

When you want to write an editor or librarian program or an adaptation for a universal editor, the question remains whether all needed messages are implemented in the device. The second question is what to do if a certain message does not exist. Unfortunately, it has not got around at certain MIDI manufacturers that the existence of a bank dump and an edit buffer dump does not allow to copy memory locations via MIDI. Often the user of a program which does this task even has to press certain buttons at the device - what technology anachronism!

So there are certain types of »workarounds« which make programming and/or operating more difficult, but at least solve the problem. A typical solution shall be described in some examples. Let's take the Waldorf microWave, version 1.25¹.

As described above, the microWave provides a message to transmit all 64 internal Sound programs and one to transmit the edit buffer, plus the requests belonging to them. However, there is no message to directly transmit or request a certain memory location in the bank. However, this function can be simulated by certain combinations of MIDI messages and button hits.

Requesting a memory location (let's use A03) is as follows: first the appropriate Program Change message (\$Cx \$03), then the »Basic Program Request« (BPRR) is transmitted. The »Basic Program Dump« (BPRD) the microWave will react with is not regarded as like that, but as a single dump for the desired memory location. (Note that it is problematic that the edit buffer contents are overwritten. The ideal case

¹. *This does not mean in any way that we think Waldorf products are bad - to the contrary: the microWave is a superb product and extremely stable. Merely the MIDI implementation's extent partly leaves a great deal to be desired, as we will see. Version 2.0 includes some new features, but still lacks the possibility to directly store into memory locations. The new microWave II however has an ideal implementation.*

would be to re-establish the edit buffer after the dump transfer.) However, selecting a certain memory location in the microWave is not that easy:

- the microWave must be in Single mode because otherwise, an Arrangement is selected instead of a Sound program. In versions 1.00 and 1.10, choosing a certain mode is only possible by pressing certain buttons at the microWave itself.
- the »Global MIDI Channel« must be known - this is the MIDI channel on which the microWave receives channel data in Single mode. It is independent from the SysEx device ID - contrary to the DX7 for example. This channel must be used in the Program Change message. To get this channel is only possible in version 1.20 or higher - from this version on, there are dump and request messages for the »Global Parameters« where this channel parameter is memorized.
- Program Change messages must be recognized by the microWave at all. In the »Global Parameters«, a parameter exists which allows to filter out incoming Program Change messages. To set this parameter via MIDI is possible only from version 1.20 - in earlier versions, no Parameter Change messages existed for the »Global Parameters«.
- the »Sound Program Change Map« must be deactivated. This table can assign any memory location to each incoming Program Change message, which is not desired in our case. This table can be deactivated by a Parameter Change message.

Not until now, the Program Change can be transmitted. Some devices might need a »rest time« between the reception of the Program Change message and the next dump request in order to activate the new Program. Not to keep this pause can result in very strange things: either the device ignores the request completely, it crashes, or you receive a dump which still contains old data. As the microWave is »clean« in this respect, the pause time can be omitted in this case.

Note that the Program Change message could have effects on other devices in your MIDI setup: the Program Change message is not proprietary to the microWave. Another device might react on the same MIDI channel - in the worst case, it's a MIDI patchbay which is connected between the computer and the microWave, resulting in disconnecting them.

You can see: the fact that a certain message type does not exist, many new problems arise which sometimes can't be solved at all or at least in an »ill-mannered« way.

A further example is storing a Sound program in a certain memory location of the microWave: after sending the Program Changes message preceded by the above mentioned measures, a »Basic Program Dump« message is transmitted to the microWave. Now, the Sound program to be stored is located in the edit buffer, and the recently selected memory location is the one where we want to store the Sound program. Unfortunately, the microWave V1.25 does not provide a »Write Request« message which would deal with this task. So there's nothing left to the programmer than to ask the user with a message to press the buttons [Shift] and [Store]. On the one hand, multiple copy operations become a patience puzzle, on the other hand, the programmer has to depend on the user's (human) reliability.

In order not to throw a unfavorable light on Waldorf, it is pointed to the fact that the above described problems applies to many devices of other manufacturers, e.g. all devices from Ensoniq, most from Korg, older ones from Roland, and all DX/TX devices and even some newer ones from Yamaha.

Of course the problems in SysEx communication are not limited to copy operations. There are many other limitations which make a programmer's life a misery and thus may delay or even prevent editor/librarian programs from being published.

Therefore, in the Technical Standards Bulletin Board (TSBB) of the MIDI Manufacturers Association (MMA), a 25 page article written by me and Robert Melvin (formerly Dr. T's, now Mark of the Unicorn) has been published. It contains notes on

- how to arrange parameters in a dump
- the display driver software
- global MIDI behavior
- dumps and dump requests
- checksums
- transmission format
- parameter changes
- and documentation

Let us hope that there are be devices in future that have no more weaknesses concerning SysEx communication. For example, it would be desirable that a device automatically transmits a message when a cartridge is inserted which will cause a librarian program to automatically request its contents.

Chapter 3

Tutorials

3

The main goal of SoundDiver was to manage and edit the data of all devices in a MIDI setup as comfortably as possible. This led to SoundDiver's modular concept: a global »frame« program builds the basis for several dedicated Modules which provide the quality of single editor programs. However, the Modules available for SoundDiver first have to be programmed. In face of the large number of different models on the market it may happen that a Module for a certain device does not yet exist, or the device is so exotic that it's not worth to write a Module for. This is the situation where the Universal Module can step in and help out.

The Universal Module is a SoundDiver Module which allows defining memory managers and editors for almost any MIDI device. These so-called Adaptations are on the same level as the other SoundDiver Modules (although they do not reach their quality). Both can be used simultaneously.

The Universal Module has been designed to have as few operational differences to the »normal« Modules and for an easiest creation of Adaptations. To create an Adaptation, you don't need to learn a programming language. You just have to give a formal description of the MIDI messages.

3.1 Bank manager Kawai K1

Now let's create a bank manager for the Kawai K1 with the Universal Module as an example. Besides some global statements we will have to define several so-called »bank drivers« which allow a MIDI data transfer of the K1's internal and external Single and Multi banks.










As you might have noticed, SoundDiver already provides a K1 Adaptation. However, we chose this example nevertheless, because

- the K1 is quite popular
- the K1 can be adapted quite easily

- you will be able to compare your try with the ready-made Adaptation.






If you might not have a K1, that doesn't matter. We have reprinted the relevant information from the K1 SysEx documentation, so you can follow the conversion into an Adaptation. The appropriate item of the K1 documentation is stated in brackets in the table titles.

Creating a new Adaptation

Open the Setup window with , and then the »Install« window with . Choose the model name »### New Adaptation ###« (to be found in the middle of the list in the »(other)« section) by entering    (German version: enter   ), and add this »device« with .

Note:

- you cannot »scan« a new Adaptation, since you have not yet defined how to do this.

In the Setup window, a new icon appears, named »NONAME00«. Open its (still empty) Memory Manager with   or , and from there the Adaptation editor with  . Tile the Memory Manager and Adaptation editor windows so that they don't overlap. This way, you will see immediately what results from the things you enter.

Global parameters

Now we will have to give some global parameters. To do so, take the K1's MIDI implementation and look up item 3 »Exclusive Data Format«. Here, the global format of a SysEx message is listed.

Table 6 K1/K1m MIDI Exclusive Format (3.)

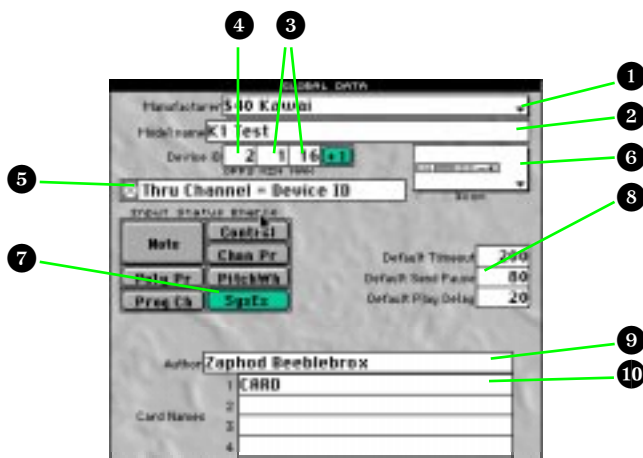
Status	11110000	FOH	System exclusive
Kawai ID no.	01000000	40H	
Channel no.	0000nnnn	0nH	
Function no.	0ffffff		
Group no.	00000000	00H	Synthesizer group
Machine ID no.	00000011¹	03H	K1/K1m ID. no.
Sub 1	0sssssss		Sub command 1
Sub 2	0sssssss		Sub command 2
Data	0xxxxxxx		
Data	0xxxxxxx		

...

Table 6 K1/K1m MIDI Exclusive Format (3.)

Data	0xxxxxxx
Data	0xxxxxxx
EOX	11110111 F7H

1. The K1 documentation reads 00000010 here. This is a typo. Unfortunately, you will often have to fight with such bugs as an Adaptation author.



The »Kawai ID no.« is »01000000 40H«. The H placed after means hexadecimal display. However, the Universal Module uses a \$ placed before to mark hexadecimal numbers. So you have to enter the value \$40 in the »Manufacturer« parameter ① in the Adaptation editor's global part. The »Model name« ② would normally be »K1«; as we already have an Adaptation with this name, you must choose a different name like »K1 Test«. Most of the other parameters are already preset correctly:

- The »Channel No.« corresponds the »Global MIDI Channel«, thus the device ID, and has a value range of 1 to 16 ③. It is the third byte; so we have an offset of 2 ④ (offsets are counted from 0). The Universal Module now automatically adds the device ID to the third byte of every SysEx message.
- The K1 may receive MIDI data on multiple MIDI channels in Multi mode. Thus, the actual Thru channel of SoundDiver's »MIDI Thru« function and the Global MIDI Channel are independent. Therefore, the switch »Thru Channel = Device ID« ⑤ remains unchecked.

- The selected icon **6** is only used to display it in the Setup window. However, it also determines that the K1 has a keyboard and can be defined as a Master Keyboard (i.e. the parameter »Master Keyboard« appears in the Device Parameter box and can be checked).
- Only SysEx messages must be processed. So all other »Input Status Enable« switches are switched off **7**.
- The default parameters **8** need not be changed either.
- You can give your name in the »Author« field **9**. This name will appear in the lower left corner of the Memory Manager window and all Editor windows.
- The K1 has one Card slot, so we enter the text »Card« in the first line **10**. We will need this later for defining the Card banks.

Initialization message

The K1 does not need an Initialization message so far, so leave this field empty.

Defining the Scan function

The definition block »Device Scan« is relevant when SoundDiver's Scan function searches for the connected devices.

The field »Universal Device Inquiry« stays empty, since the K1 does not support this message type. Instead, it uses an own pair of request and answer messages which are called »Machine ID Request« and »Machine ID Acknowledge« here.

These MIDI messages can be found in the items 5-10 and 4-7 of the K1 documentation.

The screenshot shows a MIDI message editor with two sections. The top section is titled 'Universal Device Inquiry' and contains a list of MIDI messages with their hex values and descriptions. The bottom section is titled 'Machine ID Request' and 'Machine ID Acknowledge' and also contains a list of MIDI messages with their hex values and descriptions.

OP#	HEX	DEC	ABC	REQUEST 1
0	5F 02 40			System Exclusive (SysEx)
1	54 00 64 00			Manufacturer: Kawai
2	50 00 00			Binary: 00000000 (Device ID)
3	50 00 00			Binary: 01100000
4	5F 72 47			End of Exclusive (EOX)

OP#	HEX	DEC	ABC	ANSWER 1
0	5F 02 40			System Exclusive (SysEx)
1	54 00 64 00			Manufacturer: Kawai
2	50 00 00			Binary: 00000000 (Device ID)
3	56 01 97 00			Binary: 01100001
4	50 00 00			Binary: 00000000
5	50 00 00			Binary: 00000011
6	5F 72 47			End of Exclusive (EOX)

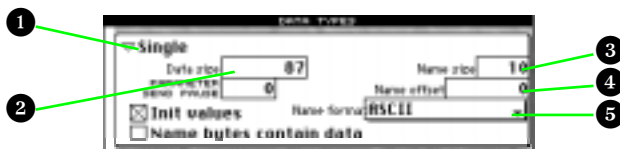
OP#	HEX	DEC	ABC	REQUEST 2
0				

OP#	HEX	DEC	ABC	ANSWER 2
0				

Defining the data types

The Universal Module separates the terms »data type« and »bank«; therefore, separate definition areas are available. Each bank refers to one data type which consequently needs to be defined only once. As an advantage, different banks using the same data type (e.g. internal and external Singles) are automatically recognized to be compatible in copy operations (even from and to Libraries).

Now you will have to define the data types »Single« and »Multi« exactly. This is accomplished by filling out the area »Data Types« below the global area.



The type name **1** of the first data type should be »Single«. The »Data Size« **2** defines the memory allocation of one Single in bytes. You can find this out from item 4-1 of the MIDI implementation. It is being said that there is »Patch Data s0 .. s87«. This would mean a size of 88 bytes. But now pay attention to item 6 »Single Data List«. In line »s87«, the checksum is stated. This means that Kawai (strangely) considers the checksum to be a part of the data. It would work if you'd define it this way. However, as soon as you change something in the Single's data (e.g. by renaming it), the Universal Module must be able to automatically recompute the checksum. For this reason, the checksum must always be considered to be separate from the data, and its calculation must be defined in the bank driver (see section *Single Dump* on page 49). Thus, the Single data size is 87 bytes.

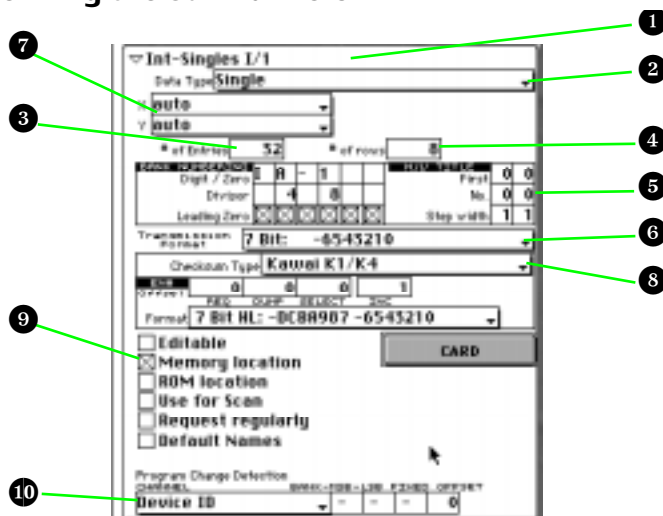
Now, it's essential to specify position, length, and format of the Single's name in the data block. This information is needed by the Universal Module to automatically display the names in the Memory Manager.

All settings are already correct by coincidence. Nevertheless, here are some hints how you can find this out. In the »Single Data List«, lines »s0« to »s9« contain the parameters »Name 1« to »Name 10«. You conclude that the name consists of ten characters **3** and begins at offset 0 **4**. The format **5** is given as »Ascii« in »s0«.

The same procedure is repeated for the data type »Multi«. Here, you first have to create a new definition block. Be sure that the small arrow at the window's left border (i.e. the insertion point) is below the »Sin-

gle« definition block, and choose »New Data Type« from the local »Adaptation« menu. A new data type definition block appears, and you can immediately enter the name »Multi«. The only additional value you have to enter is the data size 75 (how you can find out this value is similar to the method for the Single data type).

Defining the bank drivers



Now we will define the MIDI communication for the Singles I bank. To do this, the area »bank driver« is filled out. The bank's name is »Int-Singles I/1«, this is entered in the field »bank name« ¹. The »1« is used to separate this bank from the second internal bank which will be called »i/2« (the upper case I and lower case i which are used by Kawai would both be displayed as I in the Memory Manager, since SoundDiver always shows bank title bars in upper case). The data type »Single« is already correctly set.

The field »# of Entries« ³ must be set to 32, because the bank consists of 32 Singles. Now, in the Memory Manager window, a line of fields should appear. »# of rows« should be set to 8 (or 16 if preferred) so that you have a better overview. The »bank numbering« parameters ⁵ define how the single entries of the bank are numbered. The upper line »IA-1« defines the first entry in the bank. The numbers 4 and 8 in the lower line define that the bank is subdivided into four groups A, B, C, D with each 8 Singles.

When you examine the »Single Data List« of the MIDI implementation, you can recognize that the binary display of the bytes have always the MSBit (i.e. the leftmost bit 7) cleared. You can conclude from this, that the K1 uses the technique to not use bit 7. Thus, the K1's data can be transmitted via MIDI just as they are in memory. Therefore, the Transmission Format »7 Bit« is preselected correctly ⁶.

The setting »auto« in the x and y ⁷ parameters let the bank be automatically placed at a suitable position in the Memory Manager window.

The »Checksum type« ⁸ is set to »Kawai K1/K4« (what else?). The other parameters need not be changed.

Now let's consider the six switches below. We're defining an internal memory bank, so turn on the switch »Memory location« ⁹. This switch tells SoundDiver that the Entries of this bank are valuable, so that there will be a safety message before they are overwritten. A second function of this switch is that this bank will be considered in the functions »AutoRequest« and »Build Library«.

Note:

- For each data type in an Adaptation, there should be at least one editable entry. This is not the case yet in our example. The effect is that the functions »Surf!« and »AutoSurf« will not work, nor »Dive!« does.

The »Program Change Detection« parameters are used for the AutoLink Name Provider feature. The »Channel« parameter defines the MIDI channel on which the K1 reacts on Program Changes for this bank. Unfortunately, the K1 MIDI implementation does not tell this channel, so we set it to the device ID which is identical to the Global MIDI Channel in our case. Another option would be to set the parameter to »multiple«. Then the user could switch each channel on and off individually. Note that the »Offset« parameter stays 0, since the K1 switches to Singles for the Program Change numbers 1 to 64 (which is transmitted as 0 to 63).

Single Dump

Now we will do something more difficult: the definition of the MIDI strings. First we will define the string »Single Dump«.

Table 7 One Single Data Dump (4-1)

Status	11110000	FOH	System exclusive
Kawai ID no.	01000000	40H	
Channel no.	0000nnnn¹	OnH	

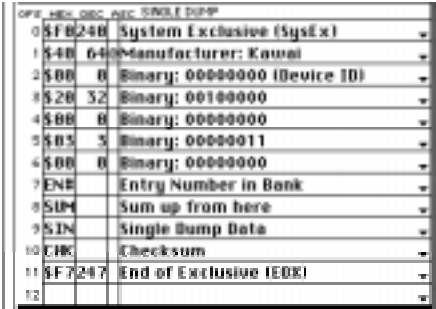
Table 7 One Single Data Dump (4-1)

Function no.	00100000	20H	One block data dump
Group no.	00000000	00H	Synthesizer group
Machine ID no.	00000011 ²	03H	K1/K1m ID. no.
Sub command 1	0000000x	00H 01H	Internal External
Sub command 2	0xxxxxxx		0..63 SINGLE A-1..d-8
Data	0xxxxxxx		Patch data s0
Data	0xxxxxxx		Patch data s1
Data	0xxxxxxx		Patch data s2
	...		
Data	0xxxxxxx		Patch data s85
Data	0xxxxxxx		Patch data s86
Data	0xxxxxxx		Patch data s87
EOX	11110111	F7H	

- 1. This byte contains the device ID. This was defined in the Adaptation's global data and is marked in the Adaptation editor with the comment »(Device ID)«.
- 2. The K1 documentation sais 00000010 here. This is a misprint. Unfortunately, you often have to fight with such errors as an Adaptation author.

Its corresponding form in the Universal Module looks like

\$F0 \$40 \$00 \$20 \$00 \$03 \$00 EN# SUM SIN CHK \$F7



The so-called »pseudo byte« **EN#** means »entry number« and is replaced by the number of the entry in the bank when the message is transmitted to the K1 respectively recognized like that when the message is received from the K1. In the above parameters, you can define a fixed offset for this entry number (separately for Request, Dump, and »Select« messages). This offset is already set correctly to 0.

The pseudo byte **SUM** starts summing up the checksum (it doesn't transmit anything), **SIN** (for »single entry data«) is a placeholder for the data to be transmitted, and **CHK** a placeholder for the calculated checksum.

Important:

- Be sure that you enter this message in the field »Single Dump«, not »Single Request«.

Hints on entering the MIDI strings:

- Click into the empty field in the »Hex« column in row 0 of the Single Dump message.
- Enter the hexadecimal codes directly (without the leading \$)
- As soon you have entered a value in an empty field, the MIDI string is automatically extended by one row.
- You can enter the pseudo byte by multiply pressing ☐. Alternatively, there are abbreviations: ☐ for **EN#**, ☐ for **SUM**, ☐ for **SIN**, and ☐ for **CHK**.

The Universal Module uses the MIDI string »Single Dump« for transmitting single entries, as well as for recognizing incoming single data dump message. The same applies for bank dumps, by the way.

Single Request

Table 8 One Block Data Request (5-1)

Status	11110000	F0H	System exclusive
Kawai ID no.	01000000	40H	
Channel no.	0000nnnn	0nH	
Function no.	00000000	00H	One single or multi request
Group no.	00000000	00H	Synthesizer group
Machine ID no.	00000011	03H	K1/K1m ID. no.
Sub command 1	0000000a	00H	a=0 Int, a=1 Ext
Sub command 2	0bbbbbbb		Single or multi patch no.
EOX	11110111	F7H	

To be able to comfortable request the bank or single entries, the accompanying Request string must be entered. It can be retrieved from item 5-1:

\$F0 \$40 \$00 \$00 \$00 \$03 \$00 EN# \$F7

Now try to select some entries in the Memory Manager and choose the item »Request« in the MIDI menu. When everything has gone well, the selected entries now show names - the data has been received.

Bank Dump

Table 9 All Single Data Dump (4-3)

Status	11110000	F0H	System exclusive
Kawai ID no.	01000000	40H	
Channel no.	0000nnnn ¹	0nH	
Function no.	00100001	21H	All block data dump
Group no.	00000000	00H	Synthesizer group
Machine ID no.	00000011 ²	03H	K1/K1m ID. no.
Sub command 1	0000000x	00H 01H	Internal External
Sub command 2	00xx0000		0=i or E, 20H=i or e
Data	0xxxxxxx		A-1 s0 data
Data	0xxxxxxx		A-1 s1 data
Data	0xxxxxxx		A-1 s2 data
	...		
Data	0xxxxxxx		A-1 s85 data
Data	0xxxxxxx		A-1 s86 data
Data	0xxxxxxx		A-1 s87 data
Data	0xxxxxxx		A-2 s0 data
Data	0xxxxxxx		A-2 s1 data
Data	0xxxxxxx		A-2 s2 data
	...		
Data	0xxxxxxx		A-2 s85 data
Data	0xxxxxxx		A-2 s86 data
Data	0xxxxxxx		A-2 s87 data
	...		
Data	0xxxxxxx		D-8 s0 data
Data	0xxxxxxx		D-8 s1 data
Data	0xxxxxxx		D-8 s2 data
	...		
Data	0xxxxxxx		D-8 s85 data
Data	0xxxxxxx		D-8 s86 data
Data	0xxxxxxx		D-8 s87 data
EOX	11110111	F7H	

1. This byte contains the device ID. This was defined in the Adaptation's global data and is marked in the Adaptation editor with the comment »(Device ID)«.

2. The K1 documentation sais 00000010 here. This is a typo. Unfortunately, you often have to fight with such errors as an Adaptation author.

Kawai included the possibility to transmit 32 Singles in a single mes-

sage. This is called a Bank Dump. You don't need to define it, because the same transmission can also be achieved by Single Dumps. However, if the Bank Dump is defined, and the whole bank is to be transmitted, it is used instead of 32 Single Dump messages.

Defining an additional Bank Dump message brings you the following advantages:

- The transmission of a whole bank is faster, because many requests and dump headers are omitted.
- Active Bank Dumps (i.e. initiated at the device itself) are recognized.

The corresponding MIDI string for transmitting the whole bank looks like this:

\$F0 \$40 \$00 \$21 \$00 \$03 \$00 \$00 [SUM SIN CHK] \$F7

The square brackets define a loop, so that after transmitting each Single's data, the Single's checksum is transmitted, until the whole bank is transmitted. This is necessary here, because Kawai regards the checksum belonging to the data, and so the checksum is calculated for each Single separately.

Many other manufacturers use only one checksum for the whole bank. In this case, [**SUM SIN CHK**] would be replaced by **SUM [SIN] CHK** or better **SUM BNK CHK**.

Bank Request

To the Bank Dump, a Bank Request belongs as well.

Table 10 One Block Data Request

Status	11110000	F0H	System exclusive
Kawai ID no.	01000000	40H	
Channel no.	0000nnnn	0nH	
Function no.	00000001	01H	All single or multi request
Group no.	00000000	00H	Synthesizer group
Machine ID no.	00000011	03H	K1/K1m ID. no.
Sub command 1	0000000a	00H	a=0 Int, a=1 Ext
Sub command 2	0xxx0000		0=single I or E 20H=single i or e 40H=multi
EOX	11110111	F7H	



It corresponds to

\$F0 \$40 \$00 \$01 \$00 \$03 \$00 \$00 \$F7

The Universal Module automatically uses the Single Request or the Bank Request, depending on what is to be requested.

Further Single banks

The bank driver for the second bank »Int-Singles i/2« is very similar to the one we just created. Therefore we will use a copy:

- Select the created bank driver by clicking in the white selection column to the very left in the window. The bank selection column as well as all MIDI strings' selection columns are highlighted.
- Copy the bank driver to the clipboard by choosing »Copy« from the »Edit« menu or using .
- Set the insertion point below the bank driver by clicking at the end of the selection column. A small arrow appears, symbolizing that a new bank driver can be pasted at this position.
- Paste the clipboard contents with menu item »Paste« or .
- Now the same bank appears a second time below the first - in the Memory Manager window as well as in the Adaptation editor window.

Now we just have to enter the different data:

- The bank name should be »Int-Singles i/2«
- The bank numbering must be **ia-1** instead of **IA-1**
- The second internal Single bank is counted from 32. Therefore, change all three **EN#** offsets to 32, as well as the Program Change Detection's offset parameter.
- In the Bank Dump/Request messages: **\$20** instead of **\$00** in offset 6

Cartridge banks

The Universal-Module provides the possibility to manage Cartridge data, given the device has SysEx messages to transmit and receive it. Fortunately, this is the case for the K1. So let's duplicate the two bank drivers we created again (duplicate of bank »Int-Singles i/1« becomes »Ext-Singles E/1«, and duplicate of bank »Int-Singles i/2« becomes »Ext-Singles e/2«, and enter the following changes.

In the third bank:

- Name: »Ext-Singles E/1«
- Bank numbering: **EA-1**

In the fourth bank:

- Name: »Ext-Singles e/2«
- Bank numbering: **ea-1**

In both new Banks:

- Activate the »CARD« switch. Then the bank only appears in the Memory Manager if the CARD switch is activated in the Special Device parameters. This feature prevents confusion for users who have a K1 without a card.
- The Program Change Detection Channel must be set to »off«. This is necessary, since there is no definite ways to select an external Single by a Program Change message. Thus, external Singles cannot be access from within Logic.
- In all MIDI strings, the byte at offset 6 must be changed from **\$00** to **\$01**.

Edit buffer


As mentioned before, we need at least one Single with the attribute »editable«. However, this is not quite easily accomplished in our example. The reason is that the K1 does not support to receive or transmit its Single edit buffer. So we need a workaround. The solution is to »abuse« one of the memory locations. It should be one which is not likely to be already used, so let's take i-d8.

We will define a bank driver which simulates the Single edit buffer. Here's how to do this:

- copy the bank driver for bank »Int-Singles I/1« to the clipboard.
- set the insertion point before this bank and paste the clipboard.

Note:

- the reason to paste the edit buffer bank driver *before* and not *after* the first internal bank is the convention how banks are arranged in SoundDiver Modules and Adaptations: banks of the same data type are arranged vertically, with the edit buffer at the top.
- change the new bank's name to »Edit Single«. You can add a hint »(i/2 d-8)« to remind the user not to use this memory location.
- set the »# of entries« and »# of rows« to 1
- clear the settings for the bank numbering completely. There is only one Single edit buffer, so we don't need to numerate it.

- turn the switch »Editable« on and the switch »Memory location« off. The first switch is essential. Editable entries get the small »E« symbol when clicked.
- set the »Program Change detection channel« to »off«
- change the value **EN#** to **\$3F** in both the single request and single dump message. This requests from respectively transmits to Single i-d8
- clear the bank request and bank dump messages. This is done by double-clicking the MIDI string's selection column (the column containing the offset numbers) and hitting .

Note:

- Since the bank has only one entry, there is no difference between single and bank messages. So we only need the single request/dump messages.
- we need a message which selects Single i-d8 after it has been dumped to the K1. This can be accomplished by defining the following MIDI string in the »After Dump« section:

PAU \$CO \$00 PAU \$CO \$3F

Notes:

- You might wonder what the **PAU** pseudo bytes are good for. They cause a short pause which the K1 needs to process the incoming MIDI data.
- You might also wonder why there are two Program Change messages. The reason is that the K1 seems to do an unwanted optimization: when it receives a Program Change which already had been selected before, the current Single is not recalled again. However, this is exactly what we want to happen, so we simply send a different Program Change before the »real« Program Change \$3F which is 63 decimal and thus selects Single i-d8.

Multi banks

Now we want to define the two Multi banks and the Multi edit buffer. Again we can use the »Int-Single I/1« bank driver as a guideline. Copy this bank driver and paste it at the very end. The following changes have to be carried out:

- Name: »Int-Multis«
- The data type must be changed from »Single« to »Multi«. To do this, choose this name from the »Data Type« flip menu. Note how the newly created bank is automatically arranged to the right of the

Single banks in the Memory Manager. This is a part of the »auto« positioning feature.

- Change all three **EN#** offsets to 64, as well as the Program Change Detection Offset. Multis are accessed with 64 to 95 in SysEx messages as well as in Program Change.
- The byte at offset 7 in the Bank Request and Bank Dump messages must be changed from **\$00** to **\$40**.

To create the Card Multis bank driver, we duplicate the new Internal Multis bank driver. Change the following parameters:

- Name: »Ext-Multis«
- Bank numbering: **EA-1**
- Turn on the »Card« switch
- In all MIDI strings, change the byte at offset 6 from **\$00** to **\$01**

Multi edit buffer

For the Multi edit buffer, we again have to use the above mentioned workaround. Copy the Edit Sound bank driver and paste it before the »Int-Multis« bank driver. Change the following:

- Change the name to »Edit Multi (=Int D-8)«
- Change the data type from Single to Multi.
- Change byte at offset 6 from **\$00** to **\$01**
- Change byte at offset 7 from **\$3F** to **\$5F**

Yeah! You just created your first bank manager Adaptation. Now you can try out sounds, configure your K1's memory or build Libraries at your heart's content.

3.2 Bank manager Roland D-110

Now let's create a bank manager for a Roland device, the D-110. With all Roland devices since the S-10, the definition of a bank manager is particularly easy, since no MIDI strings have to be defined at all.

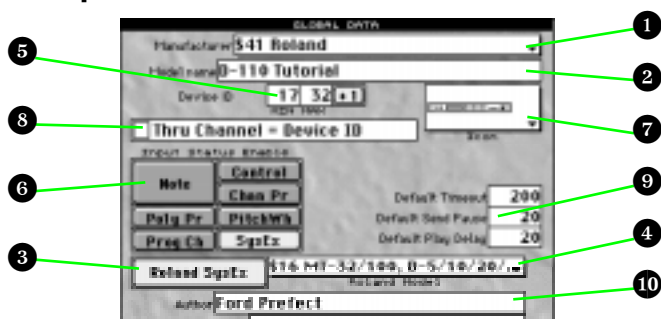
The Adaptation outlined here is to be limited to the management of Tones. However, it could be easily extended to support for Timbres, Rhythm Setups, System Setups and Patches.

The extent of the Adaptation confines itself this time to global data, a data type definition and two bank drivers.

Creating a new Adaptation

Similar to the last tutorial, you first have to create a new virtual device together with a new Adaptation with »### New Adaptation ###« in the »Install« window. Open the (still empty) Memory Manager window. With the local menu item »Adaptation → Edit ...«, the Adaptation editor window opens.

Global parameters



The first parameter ¹ is pretty clear: manufacturer Roland (\$41 hexadecimal). Below ² the name »D-110«. Probably, the Universal Module now complains that there is already an Adaptation with this name. In this case, enter a different name (like »D-110 Tutorial«).

By selecting »Roland« as the manufacturer, a new switch »Roland SysEx« ³ appeared. It is already selected so that the Universal Module's special »Roland SysEx« support is activated. Since the D-110 works with »Roland SysEx«, this is correct. You can find out that a Roland device works with Roland SysEx when a section about this »Roland System Exclusive« standard can be found in its manual.

To the right of the »Roland SysEx« switch, there is a new field which is invisible otherwise. The »Roland Model« ⁴ is for Roland devices what is known as »Model ID« at other manufacturers. Each Roland model has such an ID. But how can you tickle it out of the manual?

On page 114 (english edition) the MIDI implementation begins. In section 1, there is a reference to sections 4 and 5 concerning SysEx communication. In section 4, we make progress: »Model-ID# of D-110 is 16H« which is the same as \$16. That's what you've got to enter in this ominous field. You can opt to choose the value in a flip menu. This menu is pretty up to date. If you know IDs which are yet unassigned in the menu, please let us know.

Note:

- The value \$16 is also used by MT-32, MT-100, D-5, D-10, D-20, and GR-50. This is for compatibility reasons.

Why do I have to enter this model ID only at Roland devices? Because at other devices, you do this implicitly by defining the several MIDI strings. However in the Roland mode, the Universal Module transmits and recognizes the corresponding SysEx messages automatically. Another side effect of the Roland mode is that also the position of the device ID need not be given; it is always at offset 2.

In the same section you can find the minimum and maximum values **5** of the device ID to entered: 17 to 32. Theoretically, each D-110 part can be accessed separately by their MIDI channel as the device ID. However, this would require to install a virtual device for each part, what is not quite practical of course. Therefore, the Adaptation may use only address ranges which refer to »UNIT#«.

Note:

- when you try to enter the minimum 17, you will get a 16, because the maximum is still 16, and the minimum must not be greater than the maximum. So first enter the maximum 32, then you will be able to enter the minimum 17.

The »Input Status Enable« switches **6** are correctly again by default, and as an icon **7** we choose the 19 inch device with one 1 unit (what a coincidence, it looks exactly like a D-110...).

Since the D-110's device ID value range is different from that of a MIDI channel, the switch »Thru channel = Device ID« **8** must be switched off.

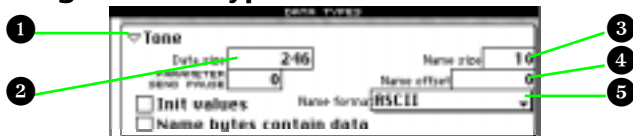
In the field »Author« **10** you can leave your mark.

Defining the Scan function

Older Roland device supports neither the Universal Device Inquiry nor a manufacturer-proprietary inquiry function; therefore you don't have to enter anything here. Instead, the switch »Use for Scan« is activated in a bank which exists only in the D-110, but not in the above men-

tioned models compatible with the D-110. This defines that SoundDiver's Scan function transmits a request for the first entry of that bank and waits for an appropriate answer from a possibly connected D-110.

Defining the data type



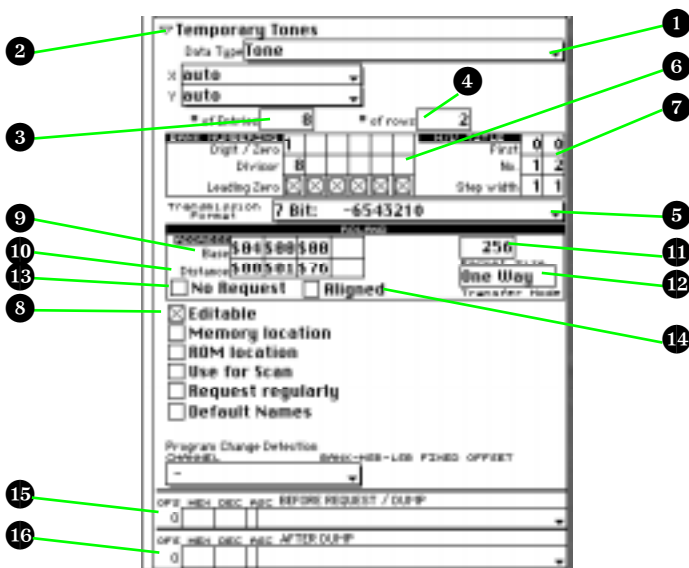
Now we're going to define the data type »Tone«. Exactly this name is entered in the name field **1** right to the triangle. The »Data size« **2**, that is the number of bytes allocated for a Tone, is not that easy to find out. In the table »Parameter Base Address« on page 117 in the area »Tone Temporary Area«, we find a reference to table 5-1. This table describes the global structure of a Tone. Unfortunately, no data information can be found; therefore we have to calculate it from the Tone's components.

A Tone consists of a »Common parameter« area and four »Partial parameter« areas. According to table 5-1-1, the first has the size »00 00 0E« which is decimal as much as 14 (see Appendix H *Conversion table* from page 283 onwards), and according to table 5-1-2, a partial needs »00 00 3A« = 58 bytes ($3 * 16 + 10 * 1$), together $14 + 4 * 58 = 246$ bytes. This is the value we finally enter in the »Data Size« field. Now watch the Adaptation editor window's local menu bar. It shows »246d | \$F6 | 01 76H 7bit Hex«. And what a miracle: the address distance between the Tone Temporary Area of Part 1 and the one of Part 1 is exactly these 01 76 bytes.

The other information is - by chance - already correct. You can check this quickly from table 5-1-1: at the address offsets 00 to 09, the name in ASCII format is located, thus: Name Size **3** = 10, Name Offset **4** = 0, Name Format **5** = ASCII.

Defining the bank driver (Tone Temporary)

And now we're about to define the bank driver for the eight parts (the Tone edit buffers) of the D-110. Since we have only one data type, this parameter **1** is already correctly set to »Tone«. The bank's name **2** should be »Temporary Tones« (of course you could also the name »Tone Parts« - this has no influence on the Adaptations's functionality). Since



we are dealing with D-110 with even eight edit buffers (quite abnormally), »# of Entries« **3** must be set to 8. »# of rows« **4** is an information which is just relevant for the bank's display in the Memory Manager window and should be set to 2.

The »Transmission format« **5** is »7 Bit« at the D-110. You can tell by the fact that no parameter has a value range greater than 0 to 127, so the MSBit of all parameters is always zero. This is obvious from tables 5-1-1 and 5-1-2.

In the bank numbering area **6**, you should simply count from 1 to 8, thus the 1 in the upper line and the 8 in the lower. The »H/V title« area **7** defines how the horizontal and vertical numbering bars to the left and at the top of the bank look like. The left column defines the horizontal title bar: it displays one digit (No. = 1) with a step width of 1. The second column defines the title column to the left. It shows one digit as well (however, No. is 2 to make the column wider).

So that the Parts become edit buffers, the »Editable« **8** switch must be activated; all others stay off.

Now we proceed to the bank driver's most important part: the »Roland« definition area. It replaces, as mentioned above, the definition of MIDI strings. In the driver for the Parts, it's pretty easy. The most important information is the »Base Address« **9**. It defines the start

address of the Parts in the D-110's »Address Map«. It is **04 00 00**. You can read this from the table »Whole Part« on page 117.

The »Distance« ¹⁰ below denotes the address distance between two Parts. It is, as we already realized, identical with a Tone's data size. Therefore, we can omit filling out the »Distance« (i.e. leave the four fields empty) - the Universal Module then automatically uses the data size as address distance. The fact that this information matches in this case however must not lead you to suppose that the address distance of a data block to the next is always its data size, as we will see with the Internal Tones.

The remaining parameter are already correct. Nevertheless a short description. The correct »Packet Size« ¹¹ cannot unfortunately be found in the manual. However, the default value »256« works perfectly. The setting »One Way« ¹² is useful here, since otherwise (with »Handshake«) a MIDI cabling to the D-110 in both directions would be necessary, which however gets in your way when auditioning Libraries. The switch »No Request« ¹³ stays unchecked, since the Temporary Tones can be requested (which is however not the case with the »Display« as an example). The switch »Aligned« ¹⁴ needs to be activated only at devices which work word-oriented (e.g. U-20), which is not the case for the D-110 however. You can see this at the fact that the address offsets in tables 5-1-1 and 5-1-2 have also odd values and denote bytes instead of words.

A MIDI string »Before Request/Dump« ¹⁵ is not needed for the D-110, since it is permanently in Multi Mode. It's just the same for the MIDI string »After Dump« ¹⁶: it's not necessary, since the dump is immediately processed by the D-110, so there's no further message necessary.

Now, transferring of Temporary Tones should already work. Try it out!

Defining the bank driver (Internal Tones)

The bank driver for the Internal Tones differs from the one we just created only slightly. Paste a new bank driver with the menu item »New bank driver« and enter:

- Bank name: »Internal Tones«
- # of Entries: 64

- # of rows: 16
- Bank numbering:

i	0	0			
	10	10	1		

This causes a numbering from **i01** to **i64**.

- Switch »Editable«: off – since in order to audition an Internal Tone, you have to copy it to a Part. When »surfing« a Tone memory location, SoundDiver automatically copies the Tone to the recently selected Temporary Tone (this is the one with the little »E« symbol).
- Roland Base Address: »**\$08 \$00 \$00**« - you can find this out from the table on page 117.
- Roland Address Distance: »**\$00 \$02 \$00**«. Here we have a special case in the D-110: the address distance between two successive Tones is greater than a Tone's data size. To let the Universal Module know, the distance must be entered manually. You can calculate the needed address distance by subtracting the address of »Tone Memory #1« (**08 00 00**) from the address of »Tone Memory #2« (**08 02 00**).

Unfortunately, the D-110 does not support to receive or transmit the ROM or Card Tones. Therefore, we cannot create bank drivers for these banks.

Defining a Conversion table

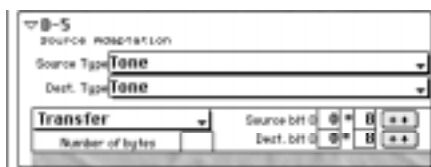
Roland have several different models of their D series. Some of them (D-5, D-10, D-20, D-110, and GR-50) are partially data compatible. On the other hand, they have major differences. Now the question arises how to take these differences into account.


The solution is to create a separate Adaptation for each model which has some unique properties. This is why SoundDiver comes with different Adaptations for MT-32, D-5, D-10/20, D-110, and GR-50. To be able to exchange the compatible data between each other of the various devices, you have to let the Universal Module know about these compatibilities. You use so-called »conversion tables« to do so.

Note:

- Conversion tables can also be used to convert data types inside a single Adaptation and even data types which are not 100% compatible.

Let's define a conversion table between D-5 Tones and D-110 Tones as an example. Create a new conversion table with the menu item of the same name. A new definition block appears, and you can immediately



enter the source Adaptation's name. This name must be 100% exact. If this is the case, you can see by the fact that after pressing , the source Adaptation's first data type is shown in the field »Source Type«. In our case »Tone« which is already what we need. Just as with the destination data type which is already set to »Tone«.

The following partial data block is a so-called »Conversion step«. This is already preset in a way that all bytes of a D-5 Tone is copied without change to the D-110 Tone.

Now you can easily copy D-5 (e.g. from a Library) to the D-110 Memory Manager. Note that the opposite direction is possible as well (given the D-110 Adaptation is in the Diver folder).

3.3 Generic mixer

The Universal Module was first and foremost developed to allow the processing of System Exclusive data (abbr.: SysEx data).

The meaning and function of all other MIDI messages were laid down by the manufacturers in the form of an obligatory protocol. This includes for example events like »Note On/Off«, »Program Change«, »Controllers« amongst others.

These messages can also be received, manipulated and re-sent by a Universal Module Adaptation. In order to show you the operation of this Module, without the need for you to struggle with the more complex blocks of SysEx, the following section will describe the construction of a simple Adaptation.

Designing the MIDI mixer


Our goal is to create a MIDI mixer, with which it is possible to control the volume and pan of timbres of connected sound modules across 16 Channels in real time as well as the muting of individual Channels.

For each mixer Channel we will need a slider, a rotary control and a switch, which gives 48 user elements in total.

- 16 Sliders for Volume Controls (MIDI message: Controller No.7)
- 16 Switches for Mute Buttons (MIDI message: also Controller No.7 set to 0)
- 16 Rotary Knobs for Pan Controls (MIDI message: Controller No.10)

Creating a new Adaptation

First of all we must create a new Adaptation.

- Change to the Setup window and press  to open the »Install« box.
- Click on the entry »### New Adaptation ###«. The box will close: on the screen there is a new device icon with the name »NONAME00« visible.
- Double-click on the device icon. This opens the Memory Manager window.
- Select »Edit ...« in the local »Adaptation« menu.

This will open the »Adaptation Editor« window.

For the following tasks it is worthwhile having both the Memory Manager and the Adaptation editor windows visible simultaneously. Use the »Tile« function in the »Windows« main-menu.

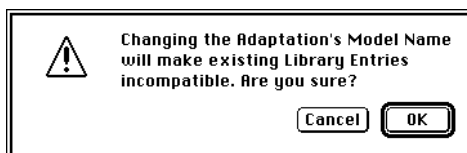
Making adjustments in the Adaptation editor

In order to construct an editor, a few preliminary default settings are necessary.

Global data

As we will not be controlling any specific instrument from any particular manufacturer with our MIDI mixer, we will leave the value in the input line next to »Manufacturer« as »00 (other)«.

- Click on the input line next to the »Model Name« where it shows »noname00«. This opens a warning box.



This warning advises you that a name change will mean that previously saved Library Entries can no longer be loaded into this Adaptation.

Don't let this warning worry you, because at this stage there are no such entries or files produced with this Adaptation.

- »OK« the warning box and then the »Edit String« text entry box will open.
- Type in the Adaptation name »MIDI mixer«.

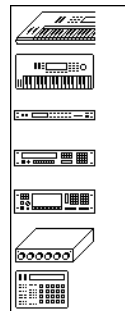
The field next to »Device ID« should be set to 0 (a blank field appears), since we do not deal with SysEx in this example.

With »Input Status Enable«, the recognition of unnecessary MIDI message types can be selectively prevented.

- In the »Input Status Enable« block, activate the »Control« switch and deactivate the »SysEx« switch.
- Click on the »Icon« graphic to the right of »Input Status Enable«.



In the flip-menu which opens you can choose which icon symbol should be used as the representation on the main screen. In this case, let's use the small Drum Machine icon at the bottom.



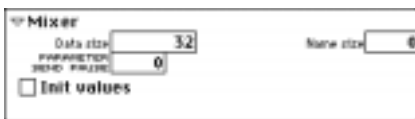
»Data Type« block

When creating an Adaptation, the next thing is to indicate the size, names etc. of a Data type.

In your later work you will for the most part create Adaptations for the different data types (e.g. Voices, Multis etc.) of a particular machine. In this case however it is only a question of making an Adaptation for just one data type – namely a combination of different Controller messages taking the form of a MIDI mixer.

In order to establish the necessary value for »Data Size« – this refers to the size and amount of the MIDI data to be generated – we must return once more to the »ingredients« of our mixer. For each of the 16 mixer Channels (referring to MIDI Channels 1–16) we need three control elements, of which two – Volume Control and Mute Switch – are linked to the same MIDI Controller (Control 7). Therefore for each Channel only two controllers are required. Altogether then for 16 MIDI Channels, 32 MIDI messages will be processed and sent. Each of the 32 MIDI messages must have a byte reserved for it in the editor.

- Increase the value of »Data Size« to »32«.



- Click on the empty field to the right of the triangle and type in »Mixer« into the text input box which appears. The text typed in here appears later in the Memory Manager and in its title line as the bank name.
- Reduce the »Name Size« value to »0«. Now the two fields »Name Offset« and »Name Format« are no longer shown; in this case no further inputs are required.

»Bank Driver« block

The Adaptation concept is based on the idea of joining together several Entries in the form of banks. Although in the case of our MIDI mixer it is only a question of one Entry, a bank must still be created for it. This corresponds to a synth sound bank with just one sound.

To define the bank we need the appropriately-named »Bank Driver«.

Use the vertical scroll bar to make the entire »Bank Driver« block visible and accessible.

In our example, only a few of the fields in the »Bank Driver« block are of any importance.

Description of the fields from top to bottom:

- »Bank Name«: type in »Mixer« as the bank name.
- »Data Type«: the name of the associated Entry (data) type has just been entered, is visible in the name line and can be left as it is.
- »# of Entries«: this is where the number of required Entries per bank is entered. As we saw above, we only need a single Item; so enter the value of »1« here.
- Now the »Mixer« bank will appear in the Memory Manager window with an empty input field.
- Activate the »Editable« switch in the »Bank Driver« Block, so that an editor window can be called up for this Item.

The remaining parameters are irrelevant for our mixer.

The screenshot shows the 'Bank Driver' configuration window for a 'Mixer' bank. The 'Data Type' is set to 'Mixer'. The '# of Entries' is set to 1, and the '# of rows' is set to 1. The 'Leading Zero' is set to 0, and the 'Gap width' is set to 1. The 'Transposition' is set to 7 Bit: -6543210. The 'Format' is set to 7 Bit HL: -BCD987-6543210. The 'Checksum Type' is set to No Checksum. The 'Editable' checkbox is checked. The 'Memory location' and 'ROM location' checkboxes are unchecked. The 'Use for Scan' checkbox is unchecked. The 'Request regularly' checkbox is unchecked. The 'Default Names' checkbox is unchecked. The 'Program Change Detection' is set to OFF. The 'DPF' fields are set to OFF.

Creating the controls

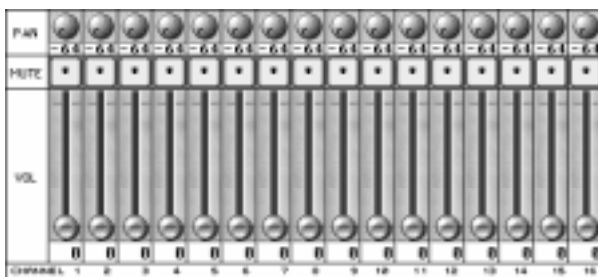
In order to create the graphic controls of the mixer, the editor must be open. To do this, double-click on the empty Entry in the Memory Manager.

Since we have not defined any dump or request MIDI strings, SoundDiver assumes that the Entry cannot be requested nor transmitted by an active dump, and thus automatically initializes it.


Now close the Memory Manager and Adaptation editor windows and enlarge the editor window.

Sliders

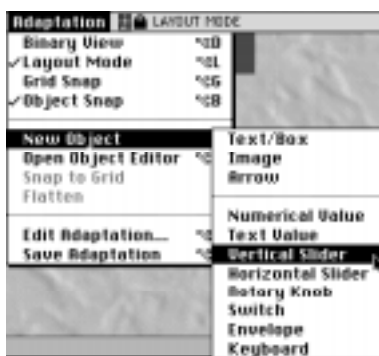
So that you get an idea of the intended object layout of the MIDI mixer, the following illustration shows the projected result.



Now a slider has to be created for the level (MIDI Volume) of MIDI Channel 1.

- Enter . The words »LAYOUT MODE« begin flashing in the info line (Windows: in the status line) to indicate that editing operation now affects objects, not their values. When you click the »Adaptation« menu title, you will see that some menu items have become available.

- Choose the menu item »Vertical Slider« from the (now available) »New Object« submenu in the »Adaptation« menu, and release the mouse button. Now you can place the slider object which appears below the mouse pointer. Leave enough room above it for the switch and the knob. To »let the slider off«, click once.



Another window opens. This is the »object editor« window, in which all the necessary changes to the graphical appearance and the associated MIDI communication are set up.

- Arrange the screen so that the two windows do not overlap, which makes them simultaneously active. The editor window must be sufficiently large to accommodate the addition of the remaining objects.

The first object is to be called »Volume Ch. 1«:

- Click on the empty field in the wide blank field of the object editor and enter this name into the text box which appears. This name appears in the editor window's menu bar whenever the object is clicked and at a later stage acts as a very useful aid in finding your way around. Object names are also used for the context-sensitive, interactive help system.

Assigning a MIDI message

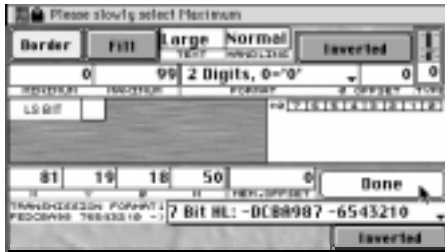
The Universal Module is able to analyze incoming MIDI messages and to assign them to newly-created objects. When the object is used, this message is actively sent with the appropriate current parameter value.

A prerequisite for the successful accomplishment of the next step is a MIDI keyboard (or other device) which can send MIDI Volume messages (Controller 7). If this is not possible, the required assignment must be accomplished manually (see later).

Input with a MIDI keyboard

- Click on the slider. Set the Send Channel of your keyboard to »1«.

- Click on the »Analyze« field in the object editor and then slowly move the slider, pedal or whatever is responsible for the MIDI Volume messages to its lowest possible position.
- Click once more on the same field (now labelled »Continue«) and move the controller to the upper end of its range. Notice that the »Maximum« value is automatically changed (it should go all the way to 127).



The Analyze function: assigning the Maximum value

- Click on »Done« again, so that its name becomes »Analyze« again.

With this simple procedure you have completely specified the MIDI message.

Manual input

(if no device with the ability to send MIDI Volume messages is available):

- Click in the first line of the »Message« area. The following values must be entered with the mouse (click in the fields and hold the mouse button: scrolling values will appear):

\$B0 \$07 VAL

Tip:

- these values can be entered very quickly with the keyboard (»**VAL**« =). Don't forget to turn on »Num Lock« before.
- The value in the »Maximum« field has to be changed from 99 to 127

You have now created one of the mixer's user elements. This slider is now able to send MIDI messages – experiment with it!

Switches

Before creating further objects, you should pull down the »Adaptation« menu and switch »Grid Snap« off and »Object Snap« on.

Now the slider must have a switch assigned to it, with which the volume of the channel can be switched back and forth between its minimum and maximum values.

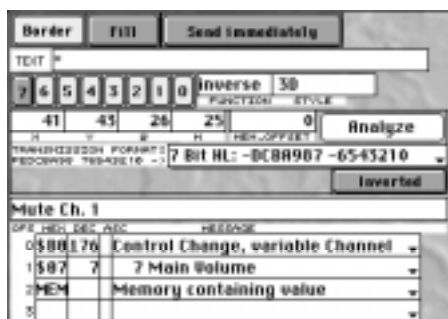
- Select a »Switch« from the »New« menu and place it above slider 1. The previously activated »Object Snap« function causes the switch to be placed flush with the top of the slider.

Remember:

- objects can be moved around only while Layout Mode is active.

All the installation options for the new object type are now available in the »object editor« window.

- Use all the values which have been entered in the following illustration.



Tip:

- by clicking on an object in the editor window, you can see all its settings in the object editor. You can use this facility to quickly compare the settings of different objects.

Make sure that switches 0 to 6 are active in the object editor; switch 7 should be off.

Controller No. 7 is once again used as the MIDI message; the »Message« entry is therefore identical to that of the slider, except that »MEM« has been entered as the third word instead of »VAL«. (click in any field of the appropriate line and type M).

»Memory Offset« is set to »0«. Switch 1 and Slider 1 use the same storage position: if the switch is turned on and off, the slider jumps to the maximum or minimum value accordingly.

Note:

- Slider settings are lost when the associated switch is used.
- Give the new object the name »Mute Ch. 1«.

Rotary knobs

Now we need a rotary knob as a pan-pot for the first channel of our mixer.

- Choose a »Knob« from the »New« menu and position it above the switch.
- Input all the settings in the illustration below.

Controller No. 10 is selected as the MIDI message. If your keyboard is unable to send this data, then you will have to enter it manually in the »Edit MIDI String« input box. The correct message reads:

\$B0 \$0A VAL**Tip:**

- Instead of inputting the values on the numeric keypad, you can click on the text field in the right column. This opens a flip-menu from which you can select the required value.

Minimum and maximum values range between »0« and »127« respectively. The slider sends all values in the range, the switch only the largest and smallest. The former is also true of our new knob.

Pan-pots are nominally set to the »zero« or center position, i.e. the signal is at the same volume on both audio channels. For this to be the case here, i.e. so that the central position shows as »0« value, »-64« must be entered in the »0 Offset« field.

The »Memory Offset« parameter for the knob is automatically set to a value of »1«, as the Universal Module always tries to set a so far unused Memory Offset when creating a new Object. This is exactly what is needed for the new Pan object.

Copying objects (»Copy and Paste«)

In order to create the other 15 channels' objects, those you have just created can simply be copied.

- Use the mouse to drag a rubber box around all three objects. They will now be shown with an animated border and small »handles« at the edges, marking them to be selected.
- In the »Edit« menu, click first on »Copy« and then on »Paste«. By doing this, you will create copies of the objects, which for the time being are covering the originals.

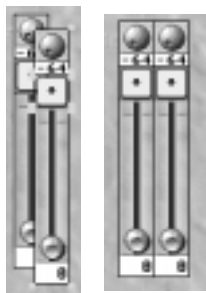
Note:

- in Layout Mode, the Edit menu has a different meaning than normally: it manipulates objects, not parameter group values.

»Copy« puts a copy of the selected objects into the »Clipboard« and then »Paste« takes the contents of the Clipboard and places them back in the active window.

- Move the mouse pointer to within the area of the selected (i.e. newly-copied) objects. The mouse pointer becomes a hand. Now

you can drag the selected objects and place them to the right of, and flush with, the originals.



Copying groups of objects

Note:

- If the mouse pointer unexpectedly changes into another symbol, this can result in a size change to the object. You can use the keyboard command **⌘Z** to return to the original status.

Except for their position, the copies are identical to the originals and are connected to the same parameters. You can tell that this is the case by the fact that moving one control causes all the others to move simultaneously.

You will find these settings:

Channel		1	2
Memory Offset	Pan	1	1
	Mute	0	0
	Volume	0	0

The copies must have separate »Memory Offset« values.

- Make sure that all objects of the second column are still selected, and choose »Open Object Editor« from the local »Edit« menu. Alternatively, you can doubleclick one of the selected objects.
- The parameter »Mem. Offset« in the Object editor should now read »0«. Change this value to »2« by click-dragging up. All selected objects' »Mem. Offset« will be changed by the same amount.

The altered settings should now be:

Channel		1	2
Memory Offset	Pan	1	3

<i>Channel</i>	<i>1</i>	<i>2</i>
Mute	0	2
Volume	0	2

Channels 1 and 2 are now independent of one another.

The objects of mixer Channel 2 must now be changed to address the corresponding MIDI Channels. Change the first value to »**\$B1**« of all selected objects:

<i>Channel</i>		<i>1</i>	<i>2</i>
Messages	Pan	\$B0 \$0A VAL	\$B1 \$0A VAL
	Mute	\$B0 \$07 MEM	\$B1 \$07 MEM
	Volume	\$B0 \$07 VAL	\$B1 \$07 VAL

The names of the objects must also be altered (Channel 2).

There follow all the necessary settings for Channel 2. Channels 3 to 16 can also be likewise duplicated.

Tip:

- do not copy single Channels only, but 2, 4 and 8 at a time. For channels 3 to 4, you will have to increase the memory offset by 4, for channels 5 to 8 by 8, and for channels 9 to 16 by 16.

The finished mixer:

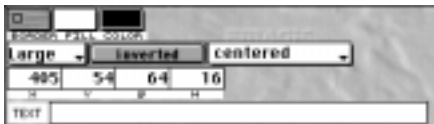
<i>Name</i>	<i>Message</i>	<i>Memory Offset</i>
Volume Ch. 1:	\$B0 \$07 VAL	0
Mute Ch. 1:	\$B0 \$07 MEM	0
Pan Ch. 1:	\$B0 \$0A VAL	1
Volume Ch. 2:	\$B1 \$07 VAL	2
Mute Ch. 2:	\$B1 \$07 MEM	2
Pan Ch. 2:	\$B1 \$0A VAL	3
... etc. until		
Volume Ch. 16:	\$BF \$07 VAL	30
Mute Ch. 16:	\$BF \$07 MEM	30
Pan Ch. 16:	\$BF \$0A VAL	31

Graphical appearance

The mixer is now fully-functional. All »technical« controls have been installed. To retain a clearer overview, the individual function areas can

be separated by additional descriptions. To this end, more object types are available.

- Look again at the diagram of our mixer example.
- Using the »New« menu, insert a »Text/Box« object.



Clicking »Border« or »Fill« in the object editor opens flip-menus. Here you may choose various borders and backgrounds. Below, you may enter textual descriptions in a text box in one of three sizes (here: »Large«), in various formats (here: »centered«), and in reverse shading.

Do use our mixer diagram as an example, but feel free to experiment with other layouts.

Saving the Adaptation

The completed Adaptation must be saved. Open the Adaptation editor window and click »Save Adaptation« in the »Adaptation« menu.

You will be warned if you try to quit SoundDiver without having saved an edited Adaptation.

If you work with several MIDI outputs, it is advisable to install one mixer per MIDI output. Follow this by simply changing the MIDI output in the Device Parameter box. By the appropriate positioning of the mixer windows you can change all channels on all outputs simultaneously.

3.4 DX7 bank loader

The following example illustrates the creation of a customized utility for a synth using the Yamaha DX7(I) as an example. If you have one of these synths or a compatible machine (TX7, TF1, DX7II, TX802), have the user manual to hand, as we will need the information about the MIDI data format.

The following instructions and underlying principles can be applied to other synths and MIDI devices.

Note:

- If you have already installed your DX7 compatible machine with

the provided DX7 Module, delete this device from your Setup window. Otherwise, incoming MIDI would not be processed as desired.

Installing a new Adaptation

- Choose »Install...« in the Setup window's local »New« menu. All available models are listed in the appearing window.
- Click the line »### New Adaptation ###«, then on the »Add« button, and close the window. A device icon with the label »NONAME00« will appear in the Setup window.
- Open the Memory Manager by double-clicking on this icon.
- Now click »Adaptation« in the Memory Manager menu bar and then on »Edit Adaptation...«. The »Adaptation Editor« will now appear. Make sure that both windows are visible.

Global settings in the Adaptation editor

The »Global Data« block is at the top of the Adaptation editor. This is where the overall settings are entered.

Click on the »Manufacturer« field and select »Yamaha« from the flip-menu which appears. You can also enter Yamaha's Manufacturer ID number (\$43 hexadecimal) via the numeric keypad.

The Adaptation must be named:

Click on the »Name« field and after »OK«-ing the warning box, enter in »DX7«.

Note:

- You can separate the DX7 Adaptation from the DX7 Module in the Install window's third column (increase the window's width if necessary): the Adaptation's line will show »UNI«, while the Module's line will read »DX7«.

Whatever name you type in will appear in future in the »Install« window and also serves as the Adaptation filename.

Now you will need your synth's System Exclusive Implementation handbook. The following information is taken from the TX802 documentation, so there may well be some slight variations in it. The data format of »Item Dumps« and »Bank Dumps« is listed in the »Bulk Data« section.

In order to be able to input the next three Adaptation parameters, you will need to find the position and value range of the »Device ID« or »Device No.«.

Status	11110000	(F0)
ID No.	01000011	(43)
Substatus / device No.	0000nnnn	(0n)
...		

As on all Yamaha instruments, the Device No. is located in the third byte of the message. So a value of »2« must be entered in »OFFS« (= offset) field in the »Device ID« line of the »Global Data« block.

In the manual, the last four bits are each represented by the variable »n«. This is the available value range for the Device No.:

4 bits allows a value range of 16 (0 to 15).

- Enter »1« in the »MIN« field and »16« in the »MAX«.
- In the »Input Status Enable« area, the SysEx switch must be depressed, so that incoming SysEx data is allowed through.
- By clicking on the »Icon« symbol, a suitable icon can be chosen from the flip-menu.

Defining the data type for the Edit Voice

In defining the data type in the »Data Type« block of the Adaptation editor we are dealing with a special case with the DX7. The DX7 uses two different formats for the »Voice« data type: »VCED« for the Edit Buffer and »VMEM« for saved Voices. In practice this means that saved Voices cannot be directly auditioned, as to do this the format must be converted.

- First we will define the »VCED« format.
- Type the name »Edit Voice« into the »Type Name« field (next to the triangle).

Now we must find the size of Edit Voice data. In the DX7 manual, you will find a table for the data formats.

Format No	Data	Byte Count
0	Voice edit buffer	155

- The value under »Byte Count« should be entered in the »Data Size« field in the Data Type block.

For the following parameters you will need the »Voice Parameter (VCED format)« table. The »P.NO« column gives the offset of the individual parameters within the Edit Voice data.

There are also 10 lines listed with the label »VOICE NAME«. The name is contained in bytes 145 to 154 and is 10 bytes long.

- –Enter the following values:
 Name Size = 10
 Name Offset = 145
 Name Format = ASCII

The Voice Name Format is given in the above table in the »DATA« column: there you will find the designation »ASC«.

The »Data Type« block should now look like this:

▼ Edit Voice	
Data size	155
PARAMETER	
SEND PAUSE	0
<input type="checkbox"/> Init values	
Name size	10
Name offset	145
Name format	ASCII ▼

The Bank Driver for Edit Voice

To make the Edit Voice appear in the Memory Manager, a separate Bank must be defined, containing just a single Item. Shift the Adaptation editor's window section so that the entire »Bank Driver« block is visible.

As so far only one data type has been defined, its name is already correctly set in the Bank Driver.

- For the »Bank Name« (to the right of the triangle), enter »Edit Voice«.
- A value of »1« must be entered into the »# of Entries« field, as the DX7 has only one Voice Edit Buffer.

For this application »Bank Numbering« is unnecessary, so leave these fields empty.

The Edit Voice bank name will now be shown in the Memory Manager in the black selection bar, under which there is a white input field.

There are still two important bits of information: »Checksum Type« and »Transmission Format«:

Yamaha uses the Checksum type »2's Complement« (for which no documentation exists unfortunately) and the »7 Bit« transmission format. With 7 bits, the maximum data value is »127«, therefore this is the upper limit on all Voice parameter values.

- Click on the »Transmission Format« field and select »7 Bit: .6543210« from the flip-menu which appears.
- Click on the »Checksum Type« field and select »2's Complement« from the flip-menu which appears.

The Dump string for Edit Voice

Now for the most important part of this tutorial, defining the »MIDI strings«. This is the data-chain of MIDI messages which control (for example) the individual operator parameters of a DX7 Voice.

- Leave the »Single Request« line empty for now and click on one of the empty fields below labelled »Single Dump«.

A string is made up of a chain of bytes, some being of defined values and others special »pseudo bytes«.

At the beginning of a data transmission, there must always be a »Header«. This serves to specify which type of data follows and to which device it is addressed.

The required header is described in its general form in the DX7 manual under the heading »bulk data«. A few more values must be added to those listed.

The first three bytes (0–2) are easily defined:

- »\$F0«, which stands for »SysEx«. Be sure »Num Lock« is on (the small key pad symbol together with a lock symbol right to the local menu bar). Enter »F0« in the first column or »240« in the second (use the numeric keypad) and move the cursor one line down);
- »\$43« for »Yamaha« (»43« in the first column or »67« in the second, and down again);
- »\$00« for the Substatus (»0« and down).

Note:

- The Device No. is packaged in the third byte of the message, the so-called »substatus byte«. Here the »High Nibble« (the first half of the byte) shows the required action (0 = Dump), and the »Low Nibble« (the second half of the byte) gives the Device Number. This can be seen from the »0000nnnn« form in the manual. »nnnn« stands for the 4 bits which specify the value range of the

Device Number (0–15). The range limits in the Adaptation editor are automatically increased by 1 to give the values »1 to 16«.

SoundDiver automatically adds the Device ID (as entered in the Device Parameter box) to the Substatus. We have already entered an offset of »2« for the Device ID in the Global Data block.

- Enter »0« for the value of the fourth byte. This is the »Format No.«. The »Voice Edit Buffer« has the »0« format.

The next two bytes are a bit more difficult to define. They give the lengths of the following data packets in the »7 Bit HL« format, which you can recognize from the »0bbbbbbb« form of both bytes. From the »MSB (= Most Significant Byte), LSB (= Least Significant Byte)« order, you can tell that the highest value byte is sent first and then the lowest.

You don't have to calculate the bytes' values yourself – the Universal Module can do this for you: simply append two **TRA** pseudo bytes to the MIDI string (do this by pressing ☐ until »TRA« is shown. As a result, two lines with the comment »Transmitted Data Size« should appear.

Notes:

- in this Adaptation, you could also use constant values (**\$01** and **\$1B**), since the Edit Voice data size is constant.
- The above mentioned Data Size encoding format »7 Bit HL« is already set correctly in the »Format« parameter above.
- A checksum is expected at the end of this SysEx message. The setting of the »**SUM**« pseudo byte in the seventh field will cause a checksum to be carried out (simply type ☐). This byte does not actually represent a value as such, is not transmitted and is only there to bring about the aforementioned checking operation.


Now comes the actual Edit Voice data. Of course it is not input individually, but rather replaced by a substitute pseudo-byte.

- Enter the pseudo-byte »**SIN**« in the eighth field (by typing ☐). This serves to represent the 155 data bytes which make up an Edit Voice.
- Enter a checksum byte »**CHK**« (enter ☐) in the ninth field.

- The »End of Exclusive« byte (abbreviated »EOX«) should appear in the tenth field (type in »F7« respectively »247«).

ofs	HEX	DEC	ASC	SINGLE DUMP
0	\$F0	240		System Exclusive (SysEx) ▼
1	\$43	67	C	Manufacturer: Yamaha ▼
2	\$00	0		Binary: 00000000 (Device ID) ▼
3	\$00	0		Binary: 00000000 ▼
4	TBA			Transmitted Data Size ▼
5	TBA			Transmitted Data Size ▼
6	SUM			Sum up from here ▼
7	SIN			Single Dump Data ▼
8	CHK			Checksum ▼
9	\$F7	247		End of Exclusive (EOX) ▼
10				▼

Now the first MIDI string is fully defined.

- Close the dialog box. Now send the contents of the DX7's Voice Edit Buffer to SoundDiver (press the corresponding buttons at your DX7). If the transfer is successful, the name of the voice selected on the DX7 should appear in the Memory Manager. If this is not the case, try comparing your input values with those shown in the MIDI Monitor window, which you can open with .

The Request string for Edit Voice

So that SoundDiver can request the DX7 data by »remote control«, we need to define a »Single Request« string.

The required information for this is listed in the manual in the »Reception Data« section under »Dump request«. A Request String can be differentiated from a »Bulk Dump« String by means of its differing Substatus byte and the lack of »Byte Count« and data bytes. Click on the »Item« field under »Request« and input the following values:

Offset	Hex	Dec	Bin	Meaning
0	\$F0	240	11110000	SysEx
1	\$43	67	01000011	Manufacturer: Yamaha
2	\$20	32	00100000	Substatus: Dump Request
3	\$00	0	00000000	Format: Edit Voice
4	\$F7	247	11110111	EOX

Now selecting the edit buffer and clicking on the »Request« button should cause the DX7 to send the current Edit Voice.

The other MIDI Strings are irrelevant for our example. You can leave these fields empty.

This completes the entries for Edit Voice in the »Item Type« and »Bank Driver« blocks.

Defining a 32-Voice Bank

As a bank with 32 »Internal Voices« uses a different format to that for the »Edit Voice« (see section *Defining the data type for the Edit Voice* on page 79), a separate definition is required for this data type.

- In the Adaptation editor window, choose »New Data Type« from the local »Adaptation« menu. An additional initialized »Data Type« block will appear. The data type to be defined should have the name »Voice«. Enter this name as usual in the »Type Name« field.

You cannot get the length information directly from the Yamaha manual. In the data format tables it says:

<i>Format No</i>	<i>Data</i>	<i>Byte Count</i>
9	Packed 32 voice	4096

Therefore a single Voice is 128 bytes long ($4096/32 = 128$). Enter »128« as the data size.

The byte length of the tone name corresponds to that of the Edit Voice (Name Length = 10). The position of the name is given in the »Voice Data (VMEM format)« table: offsets 118 to 127 contain the bytes »VNAM1« to »VNAM10«, which also represent the name in the »VCED« format. So the »Name Offset« you should enter is »118«. »Name Format« should also be »ASCII« in this case.

A Bank Driver must also be defined for this data type:

- Choose »New Bank Driver« from the local »Adaptation« menu. Arrange the window so the whole of the new empty »Bank Driver« block which appears is visible.
- Enter »Internal Voices« as the »Bank Name«.
- Click on »Data Type« and select the previously defined »Voice« from the flip-menu which appears.

Now the bank's position should have changed in the Memory Manager. This is due to the automatic bank arrangement. You can »override« it by choosing x = »left-aligned with« and y = »under« the »Edit Voice« bank.

The bank should have 32 storage locations. Type in:

of Entries = 32

of rows = 16.

Now the Bank will have 32 entries shown as two columns of 16.

As you input this data, you can see the structure of the new bank taking shape in the Memory Manager.

To make overall work in the Memory Manager possible, each entry should be assigned a number which will be shown in the info line whenever you click on a storage location. This number allocation can be separately set up for each bank in the »Bank Numbering« field of the respective Bank Driver. Here you will find two rows, each with room for 6 parameters.

On Yamaha devices, entries are numbered in the decimal system starting from 1. In order to number the entries like this in the bank, enter the following values:

Upper Row:	0	0	(leave the rest empty)
Lower Row:	10	10	1 (leave the rest empty)

The location number now has two places. Each place can have one of 10 values (0 to 9). A value of 1 should be added to each location number (normally counted from 0 up).

In order to show this, enter the following values in the »H/V Title« area to the right:

First	0	0
No.	2	2
Step width	1	1

The settings for »Checksum Type« and »Transmission Format« are identical to those for the Edit Voice (2's Complement/7 Bit).

The MIDI Strings for Internal Voices

Unfortunately, the DX7 doesn't have any way to send or receive individual entries in a 32-voice bank. Therefore MIDI Strings can only be defined for »Bank Dump« and »Bank Request«. These are very similar to the Strings defined in section *The Dump string for Edit Voice* on page 81 and section *The Request string for Edit Voice* on page 83:

Bank Dump:

Offset	Hex	Dec	Bin	Meaning
0	\$F0	240	11110000	SysEx
1	\$43	67	01000011	Manufacturer: Yamaha
2	\$00	0	00000000	Substatus: Dump
3	\$09	9	00001001	Format: Packed 32 voice
4	\$20	32	00100000	Byte Count MSB: 32 * 128
5	\$00	0	00000000	Byte Count LSB: 0 * 1
6	SUM	SUM		Start of summing
7	BNK	BNK		Bank data
8	CHK	CHK		checksum
9	\$F7	247	11110111	EOX



The »Byte Count« is worked out as above:

$4096/128 = 32$, remainder 0.

The **BNK** pseudo-byte represents all the 32 Voices which will be transmitted one after the other in the »7 Bit« format.

Bank request:

Offset	Hex	Dec	Bin	Meaning
0	\$F0	240	11110000	SysEx
1	\$43	67	01000011	Manufacturer: Yamaha
2	\$20	32	00100000	Substatus: Dump Request
3	\$09	9	00001001	Format: Packed 32 Voices
4	\$F7	247	11110111	EOX

This completes the Memory Manager for the DX7. Now save this Adaptation (type  .

3.5 DX7 editor

Basic requirements

In this tutorial, we will create parts of a Yamaha DX7 Voice editor. If you have a DX7, TX7, TF1, DX7II or a TX802, then you can follow the examples on the machine. In any case, the operation steps and the advice also apply to the procedure with other MIDI devices.

Warning:

- This tutorial is not aimed at MIDI novices or Universal Module beginners. Detailed knowledge of FM-synthesis and the DX7 is required.

Note:

- If you have already installed your DX7 compatible machine with the provided DX7 Module, delete this device from your Setup window. Otherwise, incoming MIDI would not be processed as desired.

Before creating a complete editor you should first plan out the graphic structure and appearance. Arrange all the components in such a way that the signal flow can be followed. A good example of this to look at is the supplied Matrix-6/6R editor.

In the case of a DX synthesizer, this suggestion can only be partially adhered to, as there is no signal flow in the traditional sense of the word, or rather it varies according to the algorithm used. In this case you can use a different guideline: arrange the objects as much as possible so that the re-occurring components lie one above the other (or next to one another if necessary), so that you can compare parameter groups.

The six operators should be arranged in lines, one under the other in such a way that they are all simultaneously visible on the screen. Refer to the supplied DX7 Module's Voice editor.

Numerical Value Objects

Here we are using the Memory Manager created in section *DX7 bank loader* on page 77 as a starting point.

Double-click on the »Edit Voice«. If a DX7 is connected, you should request its Edit Buffer data. If not, then clicking on »Initialize« should suffice. Now a further window will open containing the »empty« editor.

Note:

- If there is a message »No suitable editor found« instead, you have forgotten to activate the »Editable« switch in the Edit Voice bank driver.

We will begin at the top left of the editor to arrange the global parameters in a line. The Algorithm will be displayed numerically here, as this version of the Universal Module does not offer any graphic representation of parameter values.

Create a new »Numerical Value« object (choose »Numerical Value« from the »Adaptation« menu, submenu »New Object«) and place it in

the upper left corner. Make sure that both windows – object editor and Voice editor – are visible.

Have the »Voice Parameter (VCED Format)« table in the DX7 manual to hand. In it you will find the »ALGORITHM SELECTOR« parameter under P.NO = 134. This is the Memory Offset value.

The Parameter Change message can be input in two ways. The simplest method is to click on »Analyze« and change the Algorithm on the DX7:

Move the Data Entry Slider on the machine slowly to the bottom, click on »Continue«, move the slider to the top and click again on the button now called »Done«. That's it! The value range will be automatically set to 0...31.

The TX802 on the other hand cannot send Parameter Changes. We must therefore describe the somewhat more roundabout way of setting them. The SysEx message for Parameter Changes is described in the DX7 manual. The first three bytes are easily specified. The next two bytes define which parameter should be changed.

**0ggggghh
0ppppppp**

Here »**ggggg**« stands for »Parameter Group« (in our case »0«), »**hh**« for the two most significant bits, and »**ppppppp**« for the seven least significant bits of the parameter number, which – thanks to the VCED format – is identical in the case of the DX7 with the Memory Offset. In the DX manual these three pieces of information are available for every value.

However, care is advised: in the »P.NO« column of the »1 Voice Bulk Data« table, the resulting parameter number is given instead of the value for »**ppppppp**«. To get the correct value for »**ppppppp**«, the value »128« must be subtracted from parameter number 128 onwards. The »Algorithm Select« parameter is to be found under parameter number 134, so the two bytes contain the values:

**\$01 1 00000001
\$06 6 00000110**

The next byte contains the parameter value, for which the »**VAL**« pseudo-byte is required (type in ☒). The last byte carries the »EOX« status. Here is the complete message:

Hex	Dec	Bin	Meaning
\$F0	240	11110000	SysEx
\$43	67	01000011	Manufacturer: Yamaha

Hex	Dec	Bin	Meaning
\$10	16	00010000	Substatus: Parameter Change
\$01	1	00000001	Parameter Group: 0, Parameter Number 1*128 + 6*1
\$06	6	00000110	
VAL	VAL		
\$F7	247	11110111	

The value range of the parameter, i.e. the »Minimum/Maximum« values, must be entered manually. They can be taken straight from the table. The range 0...31 should be shown as »1...32« in this case, so enter the value »1« in the »0 Offset« field.

In order to label this parameter, a Text/Box object with »Algorithm« in it should be created.

The »Feedback« parameter should also appear as a numerical value: Memory Offset = 135.

»Analyze« automatically finds the Parameter Change message:

\$F0 \$43 \$10 \$01 \$07 VAL \$F7
Min = 0, Max = 7

The Voice Name consists of ten characters, each of which must have an object defined for it. Although letters are shown, it is not a case of a »Text Value« object, but a »Numerical Value« object. Create just such an object for the first character (Width = 8, »Border« and »Fill« switched off!). For »Format«, you should enter »ASCII«. At this point, the parameter value is not shown directly, but interpreted as a code for the respective ASCII character. The first character has the Memory Offset »145« and the message:

\$F0 \$43 \$10 \$01 \$11 VAL \$F7

The remaining 9 characters should be copied with »Object Snap« switched on, to place them flush with one another. For each character, the »Memory Offset« and the fifth byte of the message must be increased successively by »1«. If you have correctly input everything, the name of the Voice which is in the DX7 or »Edit Voice« should be in the display.

Envelopes

Assemble the editor for one Operator. Use the DX7 Module provided as a model. We will now move on directly to the envelopes.

Note:

- In the DX7 manual, it is unfortunately not documented that the data for the 6 Operators is arranged in reverse order. So parameter numbers from 0 to 20 are assigned to Operator 6's »EG RATE 1« through to »DETUNE«. The corresponding parameter numbers or Offsets for Operators 5, 4, 3, 2, 1 are shown down the right-hand side of the table, next to one another.

For Operator 1's Rate 1...4 and Level 1...4 we will place 8 vertical slider next to each other. The value range goes from 0 to 99, »0 Offset« is at »0«, the display Format »2 Digits, 0 = "0"«. Use the following table to continue:

Name	Offset	Message						
Rate 1	105	\$F0	\$43	\$10	\$00	\$69	VAL	\$F7
Rate 2	106	"	"	"	"	\$6A	"	"
Rate 3	107	"	"	"	"	\$6B	"	"
Rate 4	108	"	"	"	"	\$6C	"	"
Level 1	109	"	"	"	"	\$6D	"	"
Level 2	110	"	"	"	"	\$6E	"	"
Level 3	111	"	"	"	"	\$6F	"	"
Level 4	112	"	"	"	"	\$70	"	"

Don't forget to name the sliders. This will be helpful in the later stages.

The envelope graphics go next to the sliders. Select an »Envelope« object and place it as in the illustration. You can switch off the border.

The envelope object editor is very similar to the structure of the Adaptation editor, although more powerful. In the upper section you input the global settings and below, each envelope point has its own parameter block.

Global settings for envelopes

In the top right-hand corner of the envelope object editor you will find the »RANGE« input area. The four parameters arranged in the shape of a cross set the value range of the displayed x/y coordinate system. The value range for Envelope Levels is from 0 to 99. This should also be the range for the vertical axis in the envelope's graphic representation. Enter this range in the two parameter fields below and above »RANGE«. The right-hand value sets the maximum x coordinate, and as a result also indirectly determines the envelope display's time scaling. Later, it can be set as required; for the time being leave it at »200«.

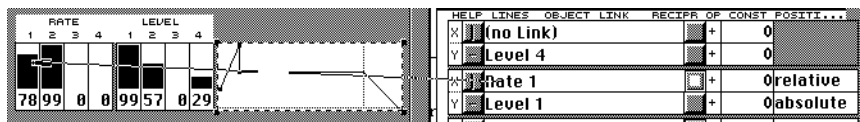
To disable the display of the Help Lines (which we do not need in this case), you should enter the value »32767« in the two »HELP LINES« fields. A »-« appears which indicates that there is no help line defined.

Defining the envelope points

The settings for each envelope point must be dealt with separately. Look at the envelope displays printed on the front panel of your DX7 or in the »Operation Guide«.

The Key On Point corresponds to our first »ENVELOPE POINT«. The x coordinate is currently set to »0« and this can be left as it is.

The DX7 envelopes have an unusual property: the amplitude of the Key On Point is controllable by Level 4. This link between slider and envelope point should be set first: click on the »(no Link)« field in the line next to »Y«. Keep the mouse button down and drag the tip of the patchcord to the Level 4 slider. On letting go of the mouse button, you will see that the connection has been successfully made by the replacement of »(no Link)« with the slider name »Level 4«.



The connection also applies any changes made in the slider value to the y coordinate of the envelope point. This latter is represented by a small black square. If this is clicked on and dragged around, the slider value will also be changed.

The second envelope point represents the end of the Attack phase. The y coordinate is directly set by Level 1: the x coordinate by the reciprocal value of Rate 1. The settings:

- for the x coordinate:
object Link: Rate 1, Reciprocal: on, OP: +, CONST: 0, Positioning: relative.
- for the y coordinate:
Object Link: Level 1, Reciprocal: off, OP: +, CONST: 0, Positioning: absolute.

The same applies to the following two envelope points, but these should be linked to Rate/Level 2 and Rate/Level 3 respectively.

Tip:

- Envelope point definitions can be selected by clicking or drag-

clicking in the white selection column at the very left, and may be copied with the usual clipboard operations Copy and Paste. To insert a new envelope point, just duplicate an existing one. There is no »New Envelope Point« function.

The next envelope point represents the Key Off Point. The amplitude is identical with that of the previous point. So these are the settings for the y coordinate:

Object Link: Level 3, Reciprocal: off, OP: +, CONST: 0, Positioning: absolute.

The moment at which the actual amplitude passes through the Key Off Point is solely dependent on when the musician releases the note. So no link should be defined for the x coordinate but a fixed absolute value of 160 should be set instead. The special nature of this point will be shown by a vertical Help Line.

It may be that the envelope displays thus far are stretched out horizontally in such a way that they overtake the 160 coordinate. In order to avoid the envelope going »backwards«, use the special positioning option of »*Key Off«. This differs from »absolute« in that the new coordinate can never become smaller than that of the previous point. In this particular case the Key Off Point will be moved to the right.

The next point should be assigned to Rate 4/Level 4.

Now an additional point should be created (by duplicating the last one), because Level 4 is also maintained after the previously defined point is reached. With a setting of say »2000« for »Const«, this point will be extended »to infinity«. Set the y coordinate thus: positioning = »relative«, »Const« = 0.

The complete definition of all envelope points is as follows:

No		Obj. Link	Recipr	OP	Const	Positioning	Help Line
0	x	(no Link)		+	0		
	y	Level 4		+	0		
1	x	Rate 1	*	+	0	relative	
	y	Level 1		+	0	absolute	
2	x	Rate 2	*	+	0	relative	
	y	Level 2		+	0	absolute	
3	x	Rate 3	*	+	0	relative	
	y	Level 3		+	0	absolute	
4	x	(no Link)		+	160	*Key Off	
	y	Level 3		+	0	absolute	
5	x	Rate 4	*	+	0	relative	

No		Obj. Link	Recipr	OP	Const	Positioning	Help Line
	y	Level 4		+	0	absolute	
6	x	(no Link)		+	2000	relative	
	y	(no Link)		+	0	relative	

That concludes the necessary settings for the envelope displays. Please use the supplied DX11 Adaptation as a guide to installing the remaining controls for the first Operator.

The finished Operator section can then be copied five times and the Memory Offset values can be easily adjusted using the Object editor. Altering the parameter numbers of the messages, on the other hand, is a bit more time-consuming (see also section *Copying objects* («Copy and Paste») on page 74).

Chapter 4

Reference – Memory Managers

4.1 Structure of an Adaptation

4

A SoundDiver Adaptation defines the structure of a whole MIDI device (not only a single bank, in contrary to other programs).

An Adaptation is divided up into four areas:

- global parameters: parameters which apply to the whole Adaptation;
- Data types (1 up to 255). In Roland mode, data types may contain an address mapping table;
- Bank drivers (having up to 32768 entries together);
- Data type conversion tables (as many as you like)

4.2 MIDI strings and pseudo bytes

MIDI strings are used to define certain parts of the MIDI communication between SoundDiver and the device. They are described in general terms in the following sections, since they are used for different purposes in an Adaptation.

Each MIDI string consists of constant values (status and data bytes) and possibly placeholders for variable data, the so-called »pseudo bytes«. MIDI strings are displayed as a list in the Adaptation and Object editors, each byte displayed in several different formats.

Status bytes

All status bytes (**\$80** to **\$F7**) are displayed in plain language.

Notes:

- The bytes **\$80**, **\$90**, **\$A0**, **\$B0**, **\$C0**, **\$D0**, and **\$E0** (i.e. channel status bytes with MIDI channel 1) are a special case: their MIDI channel is not 1 (as you might expect), but determined by the device's current MIDI Thru channel, marked by the text »(variable channel)«. This is important for Adaptations which work with channel messages. The MIDI channel of the message is thus identical with the device's Global MIDI channel or device ID. You should use these special status bytes together with the option »Thru Channel = Device ID«.
- To be able to use an Adaptation which uses channel messages, don't use channel status bytes with fixed channel (which might work with your device, but not with others), but instead with the above special case status bytes. This makes the Adaptation work with any MIDI channel.
- The status bytes **\$F1** to **\$F6** and **\$F8** to **\$FF** are not available.

Data bytes

All data bytes (**\$00** to **\$7F**) are shown in binary display to the right.

In SysEx messages, data bytes which contain the manufacturer ID, show the manufacturer name in plain text. Furthermore, the byte which contains the device ID is marked with the text »(Device ID)«.

In controller messages, the controller's name (i.e. the first data byte) is shown in plain text.

Pseudo bytes

Pseudo bytes are placeholders for variable data. Some pseudo bytes are for internal control only and do not refer to a certain MIDI byte, others represent a single MIDI byte, others more than one.

VAL – Parameter value

The current value of the parameter, encoded in the parameter transmission format. Shortcut: ☒

Notes:

- **VAL** is only available in Parameter Changes.
- each **VAL** pseudo bytes represents one MIDI byte. Coherent **VAL** bytes are treated together and transmitted/recognized using the parameter transmission format.
- the value is transmitted as it is in the dump. There is currently no possibility to add an offset or scale the transmitted value (except



»MIDI«

(English and German) gives the MIDI message types which are supported by the Adaptation (or, if easier to described, the ones which are not supported).

Here you can also describe peculiarities concerning the MIDI communication (Handshake, does the machine have to be set to a certain mode, device isn't capable doing something etc.)

SoundDiver only (not in SoundSurfer):

- Here also global information on »MIDI Monitoring« and »Listen to MIDI« can be given.

If necessary, a cross-reference to »Installation« should exist.

»SysEx Communication Error«

(German: »SysEx Kommunikationsfehler«) Here you should give hints on what to pay attention to so that the MIDI communication works (»activate SysEx reception«, special features concerning the device ID or MIDI channel).

This page is recalled in:

- the dialog »SysEx Communication Error«

»Memory Manager«

(English and German) General description of the Memory Manager. You should give notes on certain configurations here (e.g. Card banks).

A schematic overview of the banks (as they are arranged in the window) is desirable.

This page is recalled in:

- the Memory Manager, if no entry is selected
- the Memory Manager, if the keyword <Bankname> (see below) could not be found

<data type>

(e.g. »Multi«, »Program«) General description of the data type

SoundDiver only:

- this page does not contain information on the editor. Instead, it should have a cross-reference to the page <data type> »Editor«.

<bank name>

(e.g. »Internal Voices«) General description of the bank.

If necessary, you should deal with in which way single entries or the whole bank can be requested or transmitted. The data type belonging to it should be mentioned at least once so that there is a hyper link to it. If there is only one bank with one entry of a data type (e.g. »Global Data«), <data type> and <bank name> is the same of course. Then, this page should contain a description of the data type as well as of the entry.

This page is recalled:

- in the Memory Manager window (cursor entry)
- when changing the cursor entry with open help window

»Device Parameter box«

(German: »Geräte-Parameterbox«) Description of the special features of the Device Parameter box as well as the Special Parameters box. The latter are the up to four Card switches (see section *Card switches* on page 142) as well as the parameter »Regular requests« (see section *Request regularly* on page 144), if available.

This page is recalled:

- when clicking the »Help« button in the Special Parameters box

If there is a »Card«, »Cartridge« or similar, there should be a such keyword so that hyper links to it are created.

<Data type> »Editor«

(e.g. »Multi Editor«) (English and German) (not in SoundSurfer, thus embrace with \ (and \)) General description of the editor. Also peculiarities, like limited MIDI communication, should be mentioned.

This page is recalled:

- in the editor, if no cursor is set.
- in the editor, if no keyword could be found for the cursor object.
- when moving the cursor in the Editor, if the help window is open and no keyword could be found for the cursor object.

<parameter name>

(e.g. »Pitch Coarse«) (English and German) (not in SoundSurfer, thus embrace with \ (and \)) Description of the parameter, including special features and description of the available values, if necessary.

If the same parameter name exists in several data types, you can create several help pages. In the latter case, the pages' keywords have the data type placed after the parameter name in round brackets, e.g. »Detune (Multi)«. This is not necessary if the parameter has the same meaning; then you only need one page.

This page is recalled:

- in the editor, if a cursor is set
- when moving the cursor in the editor, if the help window is open.

Notes:

- It is recommended to use the »Export names ...« function (see section *Export names ...* on page 108) instead of typing the parameter names manually. This prevents you from problems due to wrong spelling.
- If there are several parameters which belong together (e.g. »Filter EG Time 1..5«, »Filter EG Level 1..5«), one help page with the parameter group's name is sufficient (in our examples »Filter EG Time« or »Filter EG Level« or even just »Filter EG«). If SoundDiver does not find a help page for a certain parameter name, it cuts the last word in the search template until it is found or only one word is left.
- These pages should have a cross-reference »Parameter group: xxx« (xxx = name of the superordinate parameter group) at the end.

<parameter group name>

(e.g. »TVA ENV«, »Effect Section«) (not in SoundSurfer, thus embrace with \ (and \)) General notes on the parameter group, e.g. what it is good for.

If sensible, add a cross-reference »Part of: xxx«

»Conversion«

(German: »Konvertierung«) Describes the Adaptation's conversion capabilities, e.g. D-10 Tones to D-5 Tones.

Note:

- It's a good idea to insert additional keywords like »Convert D-10 Tones«, so that the user can better cope with the index.

»Credits«

(English and German) Here, the Adaptation's and help file's author (and eventually translator) leave his trail. You are suggested to offer your address so that interested users can contact you.

»Copyright«

(English and German) is automatically created by SSHC. This page contains the help file's creation date and a copyright notice.

Additional help pages are possible to your heart's contents. However, they are only sensible if their keyword is used in other help pages, so that hyper links to the additional pages exist.

Notes on how to write good help files

Transcribing printed manuals

Converting a device's printed user manual presents certain difficulties, because the didactical structure is completely different.

As a matter of principle, a »flat« structure (all pages are in the same hierarchy level) is often didactically more favorable than a strictly hierarchical structure.

The latter is needed in printed manuals only because there is no computer-aided search function. However, in HyperText documents, a hierarchy is superfluous.

This becomes especially evident in description of parameter groups in editors: here, an »inverse« hierarchy is even better (i.e. there are pages for every parameter and a cross-reference to common properties each, e.g. »Envelopes«).

Commonly spoken: the hierarchy should always have a structure so that the highest level can be reached by only tracing hyper links. Searching in the index should be an exception only.

Cross-references

Notes like »An overview table of the software versions available up to now can be found in the appendix, section x.« can be simply replaced by »software versions« (as a cross-reference) at the end of a page. Even this cross reference can often be omitted, since the corresponding keyword already occurs in the text above.

Notes:

- Explicit cross-references at the end of a help page should only be made if they don't yet occur in the text itself.
- Explicit cross-references should always be located at the end of a page.
- Whether you mark explicit cross-references with »See« or »See al-

so:« is a matter of taste and is left to you.

Repeated characters

Single repeated characters may be used excessively, because they are compressed very efficiently:

```
*****
****
```

needs exactly 3 bytes in the compiled help file!

Indentations

Enumerations with a dash at the beginning or similar: the last space before the text should be marked as an indentation (i.e. you must insert a backslash before it, see section \ (Backslash, Space) on page 229), even if the text is only one line. Only in this case, the formatting is correct even with a narrow help window.

Example:

```
-\ This is a good example.
- This is a bad example.
```

causes different results with a narrow help window:



Parameter descriptions

SoundDiver only:

- Notes like »slider«, »rotary knob« etc. can be omitted for single parameters - this is obvious in an interactive help system.
- If you have several parameters which are quite similar and therefore can be explained in one page, use one of the two following techniques:
 - place all the parameters' keyword lines before the page. Then, the same page is called for all those parameters.

Example:

```
\f
TVF Envelope
\f
TVA Envelope
An Envelope consists of several Time and Level
parameters [...]
```

- if the parameter names only differ in the last word or words (e.g. »TVF Env Time 1«, »TVF Env Time 2«), use only one title line containing the commonly contained word or words (e.g. »TVF« or »TVF Time«). If SoundDiver does not find the parameter name in the help file, it omits one word at the end unless the name is found or nothing is left (in this case, the »<data

type name> Editor« page is shown).

- Text values should not only be described in general terms, but the single values as well, if not too lavish. This should be done as a table rather than in plain text.

Example:

```
\f
Wave Mode

stepped \ The wave scan process steps between the
Wavesteps.

smooth \ Uses a soft interpolation algorithm for smooth
changes between the Wavesteps.
```

6.6 Running SSHC

Note:

- Running SSHC without options automatically shows online help as with option **-h**.

Windows

```
sshc [-h] [-v] [-l] [-m] [-n] [-s] [-ooutput] [file name]
[[-ooutput] [file name] ...]
```

SSHC offers different options (given in square brackets here) which may be given before the source file name.

Note:

- These options are not available when dragging a source file onto the SSHC icon or double-clicking it with SSHC installed for **.ADT** files.

Macintosh

Since the Mac does not have a command line interpreter, you have to drag help file sources onto the SSHC icon. The compiled help is saved in the resource fork (resource type **HELP**, resource ID 128) of the file with the same name without the extension (**xxx.ADT** is saved in **xxx**)

Menu bar

The menu bar has the menu items

🍏 > About SSHC

shows an info about SSHC.

🍏 > Help

displays the options as if invoked by option **-h**

File > Compile ...

opens a file selector to select a source file manually

File > Remove help ...

Instead of adding help to the destination file, the help resources are removed.

File > Preferences ...

opens a dialog where you can set the default options described in section *SSHC options* on page 243. These preferences are stored directly in the SSHC application file when clicking on OK.

Note:

- Compile newer files only (Make): since the compiled help is stored in the Adaptation file directly, not the file modification date is compared, but a second resource (resource type **HDAT**, resource ID 128) is searched where the creation date of the **HELP** resource is memorized. This way, the help is recompiled although the adaptation file may be changed in the meantime.

Options file

If you have several help files or want to automate the SSHC launch, you can call SSHC with options by dragging a text file onto the SSHC icon:

- The file name must have the ending **».BAT«** (for »batch«).
- Each option must be placed in a separate line (since file and folder names may contain spaces - forbidding the space character to be used as a delimiter).
- Empty lines are allowed, so you can structure the files.
- Pathes must be full pathes including the Volume name such as **Cirrus 200-Q:Development:Surfer/Diver:Help Source:*.ADT**
- As you see, the asterisk may be the first character of a file name. If so, all files with the matching rest are compiled. Notice that SSHC does the »wildcard expansion« itself in this case.
- Option **-o:** be sure you have a colon at the end of the path.

- If SSHC is used with such a command file, all preferences settings default to »off« so that the command file options can enable single options or leave them disabled.

Example: This is the file I use to create all SoundSurfer and SoundDiver help files (English and German):

```
-m
-oConner:Dev:S/D:Release f:Diver:
SERVER.DPES:Dev:HELP_E:*.MOT
SERVER.DPES:Dev:HELP_E:*.ADT
-r2000
-oConner:Dev:S/D:Release f:Diver:
SERVER.DPES:Dev:HELP_D:*.MOT
SERVER.DPES:Dev:HELP_D:*.ADT
```

Apple Events

On a Macintosh with System 7, SSHC understands the Apple Events to open a document and to quit. Thus, you can use AppleScript or other scripting systems like UserLand Frontier to automate calling SSHC. Of course, this also works with option files (see above).

Atari

```
sshc [-h] [-v] [-l] [-m] [-n] [-s] [-ooutput] [file name]
[[-ooutput] [file name] ...]
```

SSHC offers different options (given in square brackets here) which may be given before the source file name.

Note:

- These options are not available when dragging a source file onto the SSHC icon or double-clicking it with SSHC installed for **.ADT** files.

SSHC options

Each option starts with a minus sign (-) or plus sign (+). Only the option's first character is taken into account. This way, you can opt to enter only the first character or the whole word.

All options must be separated from each other by one or more spaces.

-h (+help)

shows a short help. All following options and parameters are ignored.

Example: `sshc -h`

-v (+verbose)

If this option is given, more information is shown on screen while compiling. Each compiling phase is shown in a separate line.

Additionally, there are more tests on the completeness of the source file: if one of the pages »Installation«, »Scan«, »MIDI«, or »Device Parameter box« (or German »Geräte-Parameterbox«) is missing, a warning appears.

Note:

- If one of the pages »SysEx Communication Error« (or German »SysEx Kommunikationsfehler«) or »Memory Manager« is missing, a warning appears even if option -v is not given.
- Please recompile all your help files using the newest SSHC version, using option -v. This option returns more warnings on missing help pages. Add those pages if necessary.

Example: `sshc -v buggy.adt`

-l (+light)

SSHC ignores text which is embraced by \ (and \). This way, you can create two help file versions from one source file. This option is useful for creating Adaptations for SoundSurfer. Here, help for editors, parameters and parameter groups is not needed. By embracing all those help pages (and all cross-references to them) by \ (and \) in the source file, you won't have to delete them and thus don't have to deal with two versions of the same help.

-m (+make)

The given source files are only compiled if the destination file (or resource on the Mac) does not yet exist or is older than the source file. This option is useful to speed up the compilation process if you want to update all help files you have in one go.

Example: `sshc -m *.adt`

Only those source files will be compiled whose help files / resources belonging to them don't yet exist or are older.

Note:

- On the Mac, besides the help resource (type **HELP**, ID 128), a second help data resource (type **HDAT**, ID 128) is created which contains the creation date of the help resource. This way, SSHC can determine the help resource's creation date independent from the Adaptation file's modification date.

Important:

- To use this option, it is essentially important that your computer's system clock is set correctly. This might not be the case in

older Ataris which don't have a battery-buffered clock yet).

-n (+no_compression)

The destination file is not compressed.

SSHC normally compresses the help file / resource with a special method in order to save memory. If you just want to check the syntax of the source file, you can use this option to speed up the compilation process.

Example: `sshc -n synclavi.adt`

-s (+standard)

»Standard Macros applied first«: sometimes creates shorter help files / resources. Try it!

Note:

- SSHC's compression method works in two passes: first, a list of all words occurring in the source file is collected. Those words which allocate the most memory space (length multiplied with the number of occurrence), are replaced by a so-called macro code which makes up only two or less bytes each. This cannot be done for all words, since there is only a limited number of macro codes. Therefore, certain terms which often occur in SoundDiver help files like »Memory Manager«, »Program«, »MIDI Channel« etc. are replaced by so-called standard macros. Option -s reverses the order of macro replacement. Sometimes, this results in a better compression.

-r<offset>

Macintosh only: <offset> denotes an offset which is added to the resource ID used for the HELP and HDAT resources. This allows you to add online help in several languages to the same Adaptation file. For English help, the offset is 0. For other languages, the offset required for your language depends on a LANG resource found in the SoundDiver application. The LANG resource's ID depicts the language whose <offset> value it holds:

```
langEnglish      = 0, /* smRoman script */
langFrench       = 1, /* smRoman script */
langGerman       = 2, /* smRoman script */
langItalian      = 3, /* smRoman script */
langDutch        = 4, /* smRoman script */
langSwedish      = 5, /* smRoman script */
langSpanish      = 6, /* smRoman script */
langDanish       = 7, /* smRoman script */
langPortuguese   = 8, /* smRoman script */
langNorwegian    = 9, /* smRoman script */
```

```

langHebrew      = 10,/* smHebrew script */
langJapanese    = 11,/* smJapanese script */
langArabic      = 12,/* smArabic script */
langFinnish     = 13,/* smRoman script */
langGreek       = 14,/* smGreek script */
langIcelandic   = 15,/* extended Roman script */
langMaltese     = 16,/* extended Roman script */
langTurkish     = 17,/* extended Roman script */
langCroatian    = 18,/* Serbo-Croatian in extended Roman
    script */
langTradChinese = 19/* Chinese in traditional characters */

```

Currently, SoundDiver holds only the LANG resources for German-speaking countries. For German, the resource ID offset is 2000.

-o<output file>

Normally SSHC writes the help file or resource in the current folder (Windows/Atari only), and the file name is taken from the source file: on the Atari, the extension's last character **T** is replaced by **H**; on the Mac, **.ADT** is cut off. On Windows, an **.RTF** file is created which is the input file for the Microsoft Help Workshop. With option **-o**, the folder as well as the file name may be redefined. This allows you to place the source files in a different folder than the »Diver« or »Surfer« folder, as well as to comfortably manage English and German versions.

Directly after -o (i.e. without a separating space), the destination information must be given. To define a folder only (leaving the used file name to SSHC), this path information must end with a backslash (\, Atari) or a colon (:, Mac).

Example: `sshc -of:\surfer\ *.adt`
 writes all help files to folder **F:\SURFER**.

If you want to redefine the folder as well the file name, the file name to be used must be appended to the path information.

Example: `sshc -o\english\surfer\wave.adh`
`wave_e.adt`
 compiles the file **WAVE_E.ADT** to a help file **WAVE.ADH**
 in folder **\ENGLISH\SURFER**.

Option -o may be set to a new value before each compilation process.

Example: `sshc -o\surfer\ minimoog.adt`
`-o\surfer\english\ english\minimoog.adt`
 compiles the file **MINIMOOG.ADT** to the folder **\SURFER**
 and the file **ENGLISH\MINIMOOG.ADT** to the folder
\SURFER\ENGLISH.

Windows only:

- If you want to create several (e.g. English and German) versions

of your help files, you don't need to overwrite the same files in the »Diver« or »Surfer« folder in order to test them. SoundDiver/SoundSurfer has an undocumented feature which allows you to define a different folder where resource and help files are searched before they are searched in the »Diver« or »Surfer« folder.

SoundDiver first searches in a subfolder of the »Diver« folder which has the name of your local settings (e.g. »Deutsch« if you have set Windows to German local settings).

Macintosh only:

- If you want to create several (e.g. English and German) versions of your help files, you can write to the same files in the »Diver« or »Surfer« folder in order to test them. Simply add the **-r** option suitable for all the source files of a certain language.

Atari only:

- If you want to create several (e.g. English and German) versions of your help files, you don't need to overwrite the same files in the »Diver« or »Surfer« folder in order to test them. SoundDiver/SoundSurfer has an undocumented feature which allows you to define a different folder where resource and help files are searched before they are searched in the »Diver« or »Surfer« folder.

You need a command shell which is able to set so-called environment variables to define this setting (e.g. Gemini/Mupfel, Gulam or COMMAND.PRG).

Example: With SoundDiver, enter

```
LANG=deutsch\
```

From now on, help (*.MOH, *.ADH, and *.__H) and resource (*.RSC) files are first searched in the folder **DIVER\GERMAN**. Only if the desired file is not found in this folder, it is searched in folder **DIVER**.

Example: A useful structure would be:

- **DIVER**: normal Diver folder: Modules, Adaptations, English help and resource files
- **DIVER\DEUTSCH**: German help and resource files
- **DIVER\FRANCAIS**: French help and resource files, etc.
- **HELP_SRC**: English help source files
- **HELP_SRC\DEUTSCH**: German help source files
- **HELP_SRC\FRANCAIS**: French help source files

You can compile all help files of the above structure with a single SSHC call (write all parameters in one line):

```
sshc -m -o\diver\ \help_src\*.adt
-o\diver\deutsch\ \help_src\deutsch\*.adt
-o\diver\francais\ \help_src\francais\*.adt
```

<file name>

Name of the source file. The ending must be **.ADT** (all platforms; source files which don't have an extension or whose extension's third character is not **T**, are compiled in a different SSHC mode which is only relevant for EMAGIC programmers).

Atari only:

- The extension's **T** is replaced by a **H** (for »Help«) in the destination file name, as long the destination file name is not given explicitly (see section *–o<output file>* on page 246): ***.??T → *.??H**
- You can give any number of source files.
- Wildcards (e.g. **D–*.ADT**) must have been expanded (i.e. replaced by all file names that match the wildcard) by the command shell (which is done by Gemini and others). SSHC does not perform a wildcard expansion.
- SSHC supports parameter passing with ARGV, which is used by Gemini and others. This allows command lines of any length. Without ARGV, a command line can be only 126 characters long.

Mac only:

- For definition of the destination file name, the extension is cut off, as long as the destination file name is not given explicitly (see section *–o<output file>* on page 246): ***.??T → ***.
- The destination file (i.e. the Adaptation file) must already exist. If not, a warning appears, and the compilation process is canceled.
- SSHC does not overwrite the whole destination file, but only replaces the resources **HELP** 128 and **HDAT** 128.

6.7 SSHC error and warning messages

All error and warning messages are shown with the last paragraph's text as a hint where to search for the problem.

For details on the error messages, see section *SSHC error messages* on page 266 and section *SSHC warning messages* on page 267.

6.8 SSHC's mode of operation

If you are interested how SSHC works and what the various screen outputs mean, here the compilation phases of a help file are described in detail.

Notes:

- You can read all messages by using option -v (see section -v (+verbose) on page 243).
- The first number to the right shows the paragraphs still to process.
- The numbers in brackets show the resulting file size, as a percentage compared to the uncompressed text, and in the absolute number of bytes.
- »Loading Source File«
A so-called »parser« which is programmed as a »finite automata« reads the source file. Coherent lines are combined to paragraphs, and control characters are partially converted to a shorter format. Syntax errors are checked and warning or error messages given if necessary. Depending on the severeness of the error, the compilation might be canceled.
- »Creating Index Table«
A table of the keywords is built. If a keyword is multiply defined or too long (the maximum is 80 characters), an error message appears, and the compilation is canceled. Note that this step inserts additional characters, increasing the text size so that the percentage gets greater than 100%.
- »Creating references«
All occurrences of the keywords are marked in the text, i.e. the author does not have to define cross-references.
- Windows version: »Writing RTF and HPJ file«
SSHC now saves source files for Microsoft Help Workshop (HCW) and launches it, which creates the HLP file out of the source files. Thus, the following sections only apply to the Macintosh and Atari versions of SSHC:
- »Compressing repeated characters«
Characters which are repeated multiply (3 to 255 times) are compressed.

SSHC uses a special data compression method which compresses text by up to 50%, but still allows a decompression beginning from almost any position (which is not the case with common compression methods like Lempel-Ziv-Welch = LZW and similar).

The most commonly used words (to be precise, the words which allocate the most memory) are replaced by »macros« (sequences of 2 to 5 bytes). Spaces following a word are implicitly encoded in the macros. However, if no space follows a word to be replaced, a special character (»BackSpace«) is inserted in order to reproduce an identical result in decompression.

- »Creating Word Table«
In the first pass, all occurring words are cross-referenced in a table, together with their number of occurrence.
- »Deleting entries which are used only once«
Words which occur only once are deleted from the table, since replacing them by macros won't save memory.
- »Calculating memory usage«
The effective memory allocation of each word is calculated (occurrences times (length+1)).
- »Sorting word table«
The table is sorted by memory usage.
- »Creating Custom Macro Table«
The 239 words in the table which use up the most memory are saved in the so-called »Custom Macro Table«.
- »Sorting Custom Macro table«
This table is temporarily sorted alphabetically, in order to find its entries faster (by bisection).
- »Applying Custom Macros«
The custom macros are applied, i.e. the above mentioned 239 words are replaced by their macros. Up to four consecutive words to be replaced can be combined in one macro: n words need n+1 bytes in a macro.
- »Compressing Custom Macros«
The created table itself is compressed, since the compression method can also be recursively applied to word fractions.
 - »(1)«: The Custom Macro table is applied to itself.
 - »(2)«: The standard macros are applied to the Custom Macro table (only relevant for option -f)

- »Applying Standard Macros«
A second run replaces predefined words which often occur in the MIDI world by so-called »standard macros«.
- »Writing output file.«
Atari: SSHC creates a file in the so-called IFF format, consisting of three parts:
 - an index table which consists of offsets in the text which point to the keywords' positions. Depending on the text's size, the table is stored as words (2 bytes per entry) or longs (4 bytes per entry).
 - the list of words which have been replaced by the custom macros
 - the compressed text itselfMac: SSHC writes the same data to the **HELP** resource #128, however the three parts are stored in a slightly different format than that created by the Atari version. Additionally, a **HDAT** resource #128 is created which only contains the current date and time. Resource numbers may vary, depending on the resource number offset you used.

6.9 Mode of operation of SoundDiver's help system

Windows: SoundDiver uses the Windows Help system.

Macintosh and Atari:

When the help function is invoked, a »handle« to a help file and a text string is passed. First, the appropriate help file is loaded if not yet done. Then, the string is searched in the help file's index table.

As this table is sorted alphabetically, the bisection searching method can be used, i.e. with 2^n entries, only maximum n comparisons are necessary until the desired entry is found.

Since the keywords must be decompressed first before they can be compared with other text, the keywords may not be longer than 80 characters.

The found help pages is decompressed in a buffer. Since this is done paragraph by paragraph, paragraphs must not be longer than 4000 characters.

The decompressed buffer finally holds one page which is shown in the help window. The word wrapping is done in real-time depending on the help window width.

Appendix A

Menu overview

A.1 Local menus Memory Manager

Adaptation

Edit Adaptation... opens the Adaptation editor

Save Adaptation saves the Adaptation file

A.2 Local menus Adaptation editor

Adaptation

New Data Type inserts a new data type at the insertion point

Add Address Mapping Table adds an address mapping table to the selected data type (Roland mode only)

New Bank Driver inserts a new bank driver at the insertion point

New conversion table inserts a new conversion table at the insertion point

Save saves the Adaptation file

Export names... saves a template file as a default for a help source file. A file select box opens to specify the destination file name.

A.3 Local menus Editor window

Adaptation

Binary View shows Entry data as a list with hexadecimal and ASCII values

Layout Mode switches layout mode on and off

Grid Snap enables automatic grid alignment (in moving and sizing operations)

Object Snap enables automatic flattening alignment (in moving and sizing operations)

New Object creates an appropriate object

Open Object Editor opens the Object editor, showing one of the selected objects

Snap to Grid aligns selected objects's position to a 8x8 grid

Flatten aligns selected objects to not layer other objects

Edit Adaptation... opens the Adaptation editor

Save Adaptation saves the Adaptation file

A.4 Global menus Editor window

(only in Layout mode)

Edit

Undo, Redo cancels last operation

Cut moves selected objects to clipboard

Copy copies selected objects to clipboard

Paste pastes clipboard objects to editor

Clear deletes selected objects

Select All selects all objects

Appendix B

Key commands

The following tables show the key commands for the Windows, Macintosh and Atari versions concerning the Adaptation and Object editors. The tables are sorted alphabetically by key commands, so that you can find out the meaning of a key command as fast as possible.

B.1 Key command symbols

If you might operate SoundDiver on a different platform, here you find a list of the differences of the key commands.







Table 26 Differences of key commands

Operation	Windows	Macintosh	Atari
key command	-character	-character	-character
modified key command	-character	-character	-character
command for designing Adaptations	-character	-character	-character
scroll by page	Page, Page	,	-
upper left resp. lower right corner	,	,	,




NOTE: on Windows or Atari corresponds to on the Macintosh. This corresponds to the Macintosh convention where , not is used to modify another key. However, Windows and Atari do not recommend the use of so as to avoid combinations such as that would reset the computer.

B.2 Key commands












































Memory Manager

Atari	Macintosh	Atari	Operation
 E	 E	 E	Open Adaptation editor
 S	 S	 S	Save Adaptation

Adaptation editor

Atari	Macintosh	Atari	Operation
 S	 S	 S	Save Adaptation

Editor window

Atari	Macintosh	Atari	Operation
			Clear objects (in layout mode)
 A	 A	 A	Select all objects (in layout mode)
 B	 B	 B	Toggle object snap
 C	 C	 C	Copy objects (in layout mode)
 D	 D	 D	Toggle binary view
 E	 E	 E	Edit Adaptation
 G	 G	 G	Toggle grid snap
 L	 L	 L	Toggle layout mode
 O	 O	 O	Open object editor
 S	 S	 S	Save Adaptation
 V	 V	 V	Paste objects (in layout mode)
 X	 X	 X	Cut objects (in layout mode)
 Y	 Y	 Y	Redo object operation
 Z	 Z	 Z, 	Undo object operation

Appendix C


Mouse operation

C.1 Adaptation editor

<i>Windows</i>	<i>Macintosh</i>	<i>Atari</i>	<i>Operation</i>
	Click a parameter		sets cursor to parameter
	Click on selection column		selects block
	Click at top or bottom of selection column		sets insertion point
	Double click on selection column of a MIDI string		selects whole MIDI string
	Click and drag on a MIDI string selection column		selects a range

C.2 Editor window

(Layout mode assumed on)

<i>Windows</i>	<i>Macintosh</i>	<i>Atari</i>	<i>Operation</i>
	click an object		selects the object
	 -click an object		toggles the object's selection
	click in an empty region		deselects all objects
	drag objects (in the middle)		moves the selected objects
	drag objects at an edge or corner		resizes the selected objects

Appendix D

SSHC options

```
sshc [-h] [-v] [-l] [-m] [-n] [-s] [-roffset]
      [-ttemp-path] [-ooutput]
      [file name] [ [-ooutput] [file name] ... ]
```

Table 27

Option	Name	Description
-h	help	show options description
-v	verbose	verbose screen output, extended completeness checking
-l	light	skip Sections embraced with {(and)}
-m	make	compile file only if source is newer
-n	no compression	don't compress help file
-r	resource	(Mac only) set offset for resource ID. Default: 0 = English. German is 2000.
-s	standard	use standard macros first
-t	temporary path	set path for temporary files. Default: source path
-o	output	save help file in the given output file or folder
<file name>		source file(s) name



Appendix E

Trouble Shooting



E.1 Error messages

The following is a list of all error messages the Universal Module could show. They are shown in alphabetical order so that you can find them easier.

»Adaptation is unknown!«

- In your preferences' Setup, a device is installed which uses an Adaptation which could not be found in the »Diver« or »Surfer« folder.

Adaptation x, Data Type y: Inconsistent object list structure. At least one object has been deleted to keep structure intact.

- The Adaptation file is corrupted. SoundDiver tried to load the Adaptation and deleted objects with nonsense data in order not to get confused by them. To continue working with the mutilated Adaptation, save it with  .

»A Roland address was processed which could not be found in the address mapping of data type "xxx".«

- You have defined an undefined area within the address mapping, however a dump has arrived which fits in this undefined area. Check the address mapping again.

»At least one Driver must remain.«

- The last bank driver of an Adaptation cannot be deleted.

»Changing the Adaptation's Model Name will make existing Library Entries incompatible. Are you sure?«

- You want to rename your Adaptation. However, the Adaptation's name is saved with each Library entry in order to be able to find out the entry's source. If the Adaptation's name is changed, this assignment cannot be made anymore; the created entry cannot be used anymore.

»Error in Conversion Table: Destination offset outside valid limits.«

- The conversion table has a structure that would lead to data being written outside the destination entry. This can have many reasons. Check the length of the transferred data blocks, the ++ switches and the number of loop repetitions.

»Error in Conversion Table: Loops can be nested only in 32 levels.«

- You have loops nested in more than 32 levels in a conversion table. Try to use explicit copies of »Transfer« conversion steps.

»File exists.«

- The function »Export names ...« reports that the file name you chose already exists. By choosing »Replace« or »Overwrite« you can confirm this name.

»File UNI.MOD (or Uni) has a wrong Module version. Use INSTALL to update it.«

- You try to work with an Adaptation which needs a newer version of the Universal Module. Please contact EMAGIC or your country's distributor.

»Found an Address Mapping Table entry in data type "xxx" with Repeat > 1, but the Distance parameter is undefined.«

- To use an Address Mapping Table entry with a repetition, you must define the Distance parameter.

»Found an Address Mapping Table entry in data type "xxx" with size 0.«

- Address Mapping Table entries with size 0 don't make sense. Please check the Address Mapping Table for mistakes.

»If you want to edit Adaptations or create new ones, you can order a Programming Manual from EMAGIC or your local distributor for a nominal fee.«

- You have pressed help or F1. You are currently reading the Programming Manual, so you already have what is mentioned in this message.

»Lookahead too large.«

- A bank driver uses a data type with variable data size and has a single dump MIDI string where more than 256 bytes follow the **SIN** pseudo byte. The Universal Module then can't recognize the end of the data bytes.

»Module not loaded or Adaptation not found.«

- Your preferences file contains a device which refers to an Adaptation which cannot be found in the »Diver« or »Surfer« folder.

»New String / MIDI String does not fit in Object (Maximum Size: 64 KB)«

- Editor objects have a maximum size of 64 KB each. This limit is usually reached only by Image objects.

»No suitable Edit Entry found.«

- Maybe you have forgotten to activate the »Editable« switch in the bank which defines the edit buffer. However, this error message might also indicate that there are two different data formats for the same data type for memory locations and the edit buffer. In this case, you must define a conversion table (see section *Conversion table* on page 283).

»Object's Memory Offset exceeds Entry Size. Please correct it.«

- A value object exists in the editor whose »Memory Offset« is greater or equal the data type's »Data Size«.

»Only 256 Entry Types are possible.«

- You cannot define more than 256 data types in one Adaptation.

»Only 32768 Items are possible for one Adaptation.«

- You cannot define banks with together more than 32768 entries in one Adaptation.

»Quit Universal Module: Save changed Adaptation xxx«

- You want to quit SoundDiver, but the changed Adaptation xxx has not yet been saved.
 - Don't Save: you quit SoundDiver, but the Adaptation is not saved. Use this option if you have changed something by mistake.
 - Cancel: nothing happens, the quit command is cancelled.
 - OK: the Adaptation is saved, and you quit SoundDiver.

»Sorry, too many points!«

- You are trying to define an envelope with more than 512 points.

»Roland Handshake Communication Error: Checksum Error for xxx«

- This message only appears together with Roland SysEx. It indicates a wrong address size or a wrong transmission format.

»Roland Handshake Communication Error: xxx rejects.«

- In Handshake communication, the device reports an error. This error might be caused by a wrong address in the bank driver or by an active write protect at the device.

»Roland Handshake Communication Error: xxx did not reply.«

- The Universal Module has sent a WSD or RQD message, but did not receive an answer. Try again with a longer »Default Timeout« and check if the device must be set to a special »Data Transfer Mode«.

»The data size defined in data type "xxx" is too small for the address mapping table.«

- You have defined an address mapping which occupies more bytes than you have defined for the data type.

»There is already an Adaptation with this Name (at least File name).«

- You try to rename an Adaptation with a name which is already used by another Adaptation. Atari: if this is apparently not the case, check if the resulting file name of another Adaptation might be identical (e.g. »Prophet 5« and »Prophet 10« would both have the file name »PROPHET_ .ADA«, because only 8 characters are allowed). In this case, you must abbreviate a portion of the Adaptation's name.

»This Entry Type is used by a Driver. So it cannot be deleted.«

- You try to clear or cut a data type (or replace it by another) which is used by a bank driver or a conversion table. You can do this only after having changed the references in the bank driver or conversion table or delete these completely.

»This Object is already linked to a different Envelope.«

- You are trying to create a link with an object that is already linked with another envelope.

»This option cannot be used if "Thru Channel = Device ID" is switched on.«

- A user-defined or multiple Program Change channel cannot be defined together with this option.

»Tried to map an entry offset to a Roland address, but the address mapping table of data type "xxx" is too short.«

- An object has been edited which has a memory offset which is no more covered by the address mapping. Check the data type's address mapping. Either you have forgotten a data block, or a repetition counter is too small.

»Version Conflict. Adaptation File xxx has a wrong Version! Update your Universal Module!«

- You have tried to load an Adaptation which has been created with a newer version of the Universal Module than what is currently in use. The Adaptation uses features not present in this version.

E.2 Problems in use

Using existing Adaptations

The device does not react.

- Are the right MIDI Out port and device ID set? Check the settings in the Device Parameter box!
- Is the device set to receive SysEx messages? Look for the corresponding setting on the device itself (e.g. »SysEx enable« or »Input Filter«).

The device will not accept data.

- Is »Write Protect« switched off?

The »Generic« or »Generic SysEx« Adaptation: recorded data is not accepted by the device.

- Has the device ID been changed since the recording?
- The »Generic« and »Generic SysEx« Adaptations will not work with devices which are designed to work with Handshake com-

munication.

- The device might need delays between the individual dumps. The try to use the »Generic SysEx« Adaptation and experiment with the »Send Pause« parameter.

SoundDiver only: If one parameter is edited, another is changed as well.

- Both objects either have the same Memory Offset or overlap at least partially. This might be intention, or the two parameters are part of different parameter groups which are valid alternatively (e.g. the parameters of different effects algorithms).

MIDI communication, driver definition

No incoming data.

- First try to initiate an active dump at the device. If data is received now, the Request MIDI string is wrong.
- Is the manufacturer's ID correct?
- Is »Input Status Enable« Correct (SysEx switched on)?
- Is the parameter »Device ID Offset« correct? Does the device ID match?
- It is possible that the device can only send the data in a certain mode (as with the Korg M1). In this case, you must place the appropriate Mode Change in the »Before Request/Dump« message and maybe even add a **PAU** pseudo byte.
- With Roland devices: try using »Handshake Mode«.

The device warns of errors on receiving data.

- Is the checksum format correct?
- Are the **SUM** and **CHK** pseudo bytes in the right place?
- Is the data size right?
- Is the correct transmission format selected?
- On Roland devices: try using a smaller »Packet Size« value. Many devices which transmit nibbles use 128-byte packets.

The device reacts only occasionally or sometimes not at all, or crashes.

- Try placing **PAU** pseudo bytes at the beginning, in the middle or at the end of the dump MIDI string and experiment with different »Default Send Pause« values. **PAU** bytes are required especially with Ensoniq and some Korg devices.

There are characters missing at the beginning of the names.

There are strange characters at the end.

- Then the »Name Offset« parameter is too large.

There are strange characters at the beginning of the names. Some names are blank.

- Then the »Name Offset« parameter is too small.

Not all name characters are shown.

- Then the »Name Size« parameter is too small.

There are strange characters at the end of the names.

- Then the »Name Size« parameter is too large.

Only the first character of the name is shown.

- Try out the »Packed ASCII« name format.

Only the first entry's name of a bank is correct.

- Then a bank dump MIDI string is defined, but the data type's »Data Size« parameter is wrong. If the beginning of the second entry's name is missing, the data size is too large; if the second entry's name is missing completely of strange characters are at the beginning, the data size is too small.

Only the first entry in a bank is correctly transmitted / received.

- You have forgotten to properly set the **EN#** pseudo byte in the Single Dump or Single Request MIDI string. Be careful with the »EN# Offset« values as well.

Editor definition

A parameter reacts sluggishly to editing.

- You have forgotten to input the Parameter Change message. If this MIDI string field is empty, then the entire entry will be sent.

After editing and a request, completely different data arrives.

- Look very carefully at the order of parameters in the dump. Look it up in the SysEx documentation.

If a parameter is edited, other parameters in the device are also overwritten.

- There are several parameters collected together in the same byte or word. Look very carefully at the order of parameters in the dump and use **MEM** instead of **VAL**.

The parameter value is incorrectly transmitted.

- It may be that the several parameters are contained in one byte or word, with the parameter in question not beginning with bit 0. Or perhaps the »0 Offset« is set wrong.

The device displays an alert message when editing a parameter.

- With important parameters, safety questions have to be answered sometimes before major changes can be made (e.g. »Are you sure?«). The keyboard input for the necessary confirmation can be simulated with the »Key Remote Messages«, i.e. the required key commands can be remotely sent via MIDI. If necessary, insert **PAU** pseudo bytes after each command.

The device crashes when editing a parameter very quickly.

- Append a **PAU** pseudo byte to the Parameter Change message.

E.3 SSHC error messages

»|) without |(«

- These escape codes must be balanced. A \) without a \ (does not make sense.

»|f missing at file start.«

- Even for the first page, a \£ must lead the keyword. This error message often indicates that the source file is not an SSHC source file, but something completely different.

»Doubly defined Keyword 'XXX'«

- A keyword (line after \£) is used more than once. In this case, either the same parameter exists multiply (then simply delete the second keyword), or there are two different parameters with the same name (probably in different data types). In the latter case, append the data type in round brackets each.

»Fatal: could not open source file«

- SSHC could not find the source file. Is the used path correct?

»Fatal: too many index entries«

- The source file contains more than 1,000 keywords. Please report to Emagic.

»Fatal: too many word entries«

- The source file contains more than 10.000 different words. Please report to Emagic.

»Illegal character 'X' (0xXX)«

- A control character different from <Return> was found (tabs are not allowed!)

»Illegal character 'X' (0xXX) after Backslash«

- You entered an unknown escape code. If you want to display a backslash in the text, you have to enter two backslashes (\\).

Note:

- Escapes for octal and hexadecimal ASCII code (\0... and \x...) known from Polyframe don't exist anymore – due to incompatibilities of Atari and Mac ASCII code. Please use only ASCII Codes 32..127 and Umlaut characters.

»Keyword 'xxx': No Space (' ') at end allowed.«

- Keywords must not end with a space.

»Keyword 'xxx' too long.«

- Keywords have a maximum length of 80 characters.

»Line too long«

- You have written a paragraph with more than 20,000 characters. You should split it up into smaller fractions.

»No Keyword. Next paragraph: xxx«

- You entered a \f without a following keyword.

»No source file«

- Atari: You did not give a source file. Source files must not begin with a minus sign.

»Out of memory«

- SSHC is out of memory. Mac: increase the memory size with the Finder's »Information« window. Note: if SSHC might have created a file, this file is corrupted.

»Unexpected End of File - \ (without |)«

- These escape codes must be balanced. If you have a SoundDiver-only section, embrace them with \ (at the beginning and with \ (at the end.

»Unknown platform indicator 'x' (0xXX) at beginning of file«

- The first word in a source file must be »Windows«, »Mac« or »Atari«. Actually the first character only is checked, so actually »W«, »M« or »A« is sufficient.

»Warning: Keyword xxx not defined.«

- You forgot to create a standard help page with keyword xxx. The help file is created, but incomplete. You can add extended checking on standard pages with option -v.

E.4 SSHC warning messages

»8 bit ASCII characters found, but no platform indicator. Select platform which shows text correctly:«

- You forgot to begin the source file with the platform indicator »Windows«, »Mac« or »Atari«. However you can insert it now afterwards.
- Hit **[7]**, **[2]** or **[3]** to select the correct platform (which you can see from the text example – only one shows umlaut characters correctly).
- Hit **[4]** to display the next text example – possibly the first example does not show you anything where you can make out the correct platform.
- Hit **[5]** to abort compilation.

»(error log from file xxx)«

- The following text is the error log file created by HCW. It shows you any problems HCW had during compilation.

»Keyword xxx not defined.«

- You have forgotten to define a required keyword. Please refer to

section *Standard keywords used by SoundDiver/SoundSurfer* on page 234.

E.5 Problems when testing help files

A hyper link is not shown as desired.

- Double-check the spelling (upper/lower case, number of spaces between the words)
- SSHC skips hyper links which would make up the beginning of the current page's keyword.

Example:

\f
Wave

blabla

\f
Wavetable

A Wavetable blabla...

Here you won't get a hyper link to »Wave« as a portion of »Wavetable«

Appendix F

Bibliography

For those wishing to pursue the intricacies of MIDI, there are a few books and articles available. Some articles refer to PM-UNI, the predecessor of SoundDiver's Universal Module. Those are only partly suitable.

F.1 English

Jim Conger: MIDI Sequencing in C. M&T Books, 1989. ISBN 1-55851-046-X

Detailed description of a semi-professional 8 track sequencer for MS-DOS. Diskette with source code.

Steve De Furia, Joe Scacciaferro: The MIDI System Exclusive Book, Hal Leonard Books, ISBN 0-88188-586-x

Steve De Furia, Joe Scacciaferro: The MIDI Resource Book. Hal Leonard Books, ISBN 0-88188-587-8

Steve De Furia, Joe Scacciaferro: MIDI Programmer's Handbook. M&T Books, 1989. ISBN 1-55851-068-0 (out of print)

Complete reference of the MIDI standard. Detailed descriptions, very clear. Programming examples in Pascal.

Michael Haydn, Robert Melvin: Item #116 - Remote Control Sound Editing, from: TSBB (Technical Standards Bulletin Board) #18. MMA (MIDI Manufacturers Association), 1992

Detailed notes on an optimal SysEx implementation of MIDI devices and design of user documentation.

Chris Meyer (editor): TSBB (Technical Standards Bulletin Board) #18; MMA (MIDI Manufacturers Association), Los Angeles, USA, 1992.

Summary of the discussion inside the MMA concerning new standards and recommendations for the MIDI standard. New issues are published irregularly.

F. Richard Moore: Elements of Computer Music; Prentice-Hall, Englewood Cliffs, USA, 1990.

Substantial work on computer music from a scientific point of view. Much theory, many formulas, though still clear.

Christopher Yavelow: MacWorld Music & Sound Bible. IDG Books, 1992. ISBN 1-878058-18-5

Extremely substantial work (1,400 pages), which covers all MIDI applications on the Macintosh. All available products (even Shareware and Public Domain) are presented thoroughly and compared.

MIDI 1.0 Detailed Specification, Interim Version 4.1.1; International MIDI Association (IMA), Los Angeles, California, USA, 1991.

Official specification of MIDI. Some notes on implementations.

MIDI Show Control (MSC) 1.0; International MIDI Association (IMA), Los Angeles, USA, 1991.

Official specification of MSC

Standard MIDI Files 1.0; International MIDI Association (IMA), Los Angeles, USA, 1991.

Official specification of SMF

MIDI Machine Control 1.0; International MIDI Association (IMA), Los Angeles, USA, 1992.

Official specification of MMC

Computer Music Journal, MIT Press, Massachusetts, USA

A quarterly published magazine on newest research in computer music.

Proceedings of the 19xx International Computer Music Conference; Computer Music Association

Collection of abstracts of the lectures held at the ICMC.

F.2 German

Philipp Ackermann: Computer und Musik; Springer Verlag, Wien, New York, 1991

Michael Haydn: Was Sie schon immer über systemexklusive Daten wissen wollten ...; in: KEYS 6/91, p. 92, PPV Presse Project Verlag, Bergkirchen, Germany, 1991

Michael Haydn: Wir basteln uns einen Bank Manager; in: KEYS 6/91, p. 96, PPV Presse Project Verlag, Bergkirchen, Germany, 1991

Michael Haydn: Jäger des verlorenen Dumps; in: KEYS 1/92, p. 62, PPV Presse Project Verlag, Bergkirchen, Germany, 1992

Michael Haydn: Wir basteln uns noch einen Bank Manager; in: KEYS 1/92, p. 67, PPV Presse Project Verlag, Bergkirchen, Germany, 1992

Michael Haydn: Du sprechen SysEx?; in: KEYS 2/92, p. 76, PPV Presse Project Verlag, Bergkirchen, Germany, 1992

Michael Haydn: Ins Eingemachte, in: KEYS 2/92, p. 82, PPV Presse Project Verlag, Bergkirchen, Germany, 1992

Bert Marx, Cord Brandis: The Roland MIDI Guide; Roland Corp., 1993.

Ekki Schädel: Tips und Tricks zum Polyframe-Universalmodul PM-UNI. Keyboards 3/92, p. 72, Musik Media Verlag, Augsburg, Germany, 1992

H.J. Schäfer, L.Wagner: Key Report. Verlag M. Baumgardt, Munich, Germany, 1992. ISBN 3-9803008-0-3

Reference of all synthesizers and samplers from 1975 to 1992. Somewhat superficial.

Martin Thelen: Workshop: Polyframe Universalmodul: Wir erstellen eine Anpassung für den Kawai KC10 Spectra. Fachblatt Musik Magazin 7/92, p. 136, SZV Spezial-Zeitschriftengesellschaft, Munich, Germany, 1992

Describes the design of an editor.

Appendix G

Manufacturer IDs

If the manufacturer name of your device might not appear in the Adaptation editor's flip menu, first try to find it using the numerical value. If there is a totally different name, use it nevertheless (in this case, the manufacturer does not keep to the assignment list of the International MIDI Association, or the ID has been reassigned). If the entry should be empty, SoundDiver does not yet know this manufacturer ID. You can add the manufacturer name by editing the **TEXT** resource 129 of the SoundDiver/SoundSurfer application file (or file »**DIVE_TXT.RSC**« or »**SURF_TXT.RSC**« on the Atari). Use ResEdit (or an ASCII editor on the Atari) and insert the new line. The ID is given in hex without \$; 3-byte IDs as two hex numbers with a separating space. There must be a space as well between the ID and the manufacturer name.

This the list of the yet known manufacturer IDs (no guarantee that this list is complete!)

Table 28 American Group

<i>ID</i>	<i>Manufacturer</i>
\$00	(see 3-byte manufacturers)
\$01	Sequential
\$02	IDP
\$03	Voyetra/Octave-Plateau
\$04	Moog
\$05	Passport Design
\$06	Lexicon
\$07	Kurzweil
\$08	Fender
\$09	Gulbransen
\$0A	AKG Acoustics
\$0B	Voyce Music
\$0C	Waveframe Corp
\$0D	ADA Signal Processors
\$0E	Garfield Electronics
\$0F	Ensoniq
\$10	Oberheim
\$11	Apple Computer
\$12	Grey Matter Response

Table 28 American Group (cont.)

<i>ID</i>	<i>Manufacturer</i>
\$13	Digidesign
\$14	Palm Tree Instruments
\$15	JL Cooper
\$16	Lowrey
\$17	Adams-Smith
\$18	E-mu Systems
\$19	Harmony Systems
\$1A	ART
\$1B	Baldwin
\$1C	Eventide
\$1D	Inventronics
\$1E	Key Concepts
\$1F	Clarity

Table 29 European Group

<i>ID</i>	<i>Manufacturer</i>
\$20	Passac
\$21	Siel
\$22	Synthaxe
\$23	Stepp
\$24	Hohner
\$25	Crumar / Twister
\$26	Solton
\$27	Jellinghaus MS
\$28	Southworth
\$29	PPG
\$2A	JEN
\$2B	SSL Limited
\$2C	Audio Vertrieb P. Strueven
\$2D	Neve
\$2E	Soundtracs Ltd.
\$2F	Elka / General Music
\$30	Dynacord
\$31	Intercontinental Electronics
\$32	Drawmer
\$33	t.c. electronic / Clavia / ddrum
\$34	Audio Architecture
\$35	General Music (GEM)
\$36	Cheetah Marketing
\$37	C.T.M.

Table 29 European Group (cont.)

<i>ID</i>	<i>Manufacturer</i>
\$38	Simmons
\$39	Soundcraft Electronics
\$3A	Steinberg Digital Audio
\$3B	Wersi
\$3C	Avab Electronik Ab
\$3D	Digigram
\$3E	Waldorf
\$3F	Quasimidi

Table 30 Japanese Group

<i>ID</i>	<i>Manufacturer</i>
\$40	Kawai
\$41	Roland
\$42	Korg
\$43	Yamaha
\$44	Casio
\$45	Moridaira
\$46	Kamiya Studio
\$47	Akai
\$48	Japan Victor (JVC)
\$49	Meisoshia
\$4A	Hoshino Gakki
\$4B	Fujitsu Electronics
\$4C	Sony
\$4D	Nisshin Onpa
\$4E	TEAC Corp
\$4F	System Product
\$50	Matsushita Electric
\$51	Fostex
\$52	Zoom
\$53	Midori
\$54	Matsushita
\$55	Suzuki
\$56	
\$57	
\$58	
\$59	
\$5A	
\$5B	
\$5C	

Table 30 Japanese Group (cont.)

<i>ID</i>	<i>Manufacturer</i>
\$5D	
\$5E	
\$5F	
\$60	
\$61	Syntecno

Table 31 Global IDs

<i>ID</i>	<i>Manufacturer</i>
\$7D	Non-Commercial SysEx
\$7E	Non-Real-Time SysEx
\$7F	Real-Time SysEx

Table 32 3-byte Manufacturer IDs

<i>ID</i>	<i>Manufacturer</i>
\$00 \$00 \$00	
\$00 \$00 \$01	Warner New Media
\$00 \$00 \$02	Music Logic Systems
\$00 \$00 \$03	PAIA
\$00 \$00 \$04	Othertech
\$00 \$00 \$05	K-Muse
\$00 \$00 \$06	Stypher
\$00 \$00 \$07	Digital Music Corp
\$00 \$00 \$08	IOTA Systems
\$00 \$00 \$09	New England Digital (NED)
\$00 \$00 \$0A	Artisyn
\$00 \$00 \$0B	IVL
\$00 \$00 \$0C	Southern Music Systems
\$00 \$00 \$0D	Lake Butler Sound
\$00 \$00 \$0E	Alesis
\$00 \$00 \$0F	Sound Creation
\$00 \$00 \$10	DOD/Digitech
\$00 \$00 \$11	Studer-Editech
\$00 \$00 \$12	Sonus
\$00 \$00 \$13	Temporal Acuity Prod.
\$00 \$00 \$14	Jeff Tripp/Perfect Fretwks
\$00 \$00 \$15	KAT
\$00 \$00 \$16	Opcode
\$00 \$00 \$17	Rane Corp.
\$00 \$00 \$18	Spatial Sound/Anati Inc.
\$00 \$00 \$19	KMX

Table 32 3-byte Manufacturer IDs (cont.)

<i>ID</i>	<i>Manufacturer</i>
\$00 \$00 \$1A	Allen & Heath Brenell
\$00 \$00 \$1B	Peavey Electronics
\$00 \$00 \$1C	360 Systems
\$00 \$00 \$1D	Spectrum Design and Dev.
\$00 \$00 \$1E	Marquis Music
\$00 \$00 \$1F	Zeta Systems
\$00 \$00 \$20	Axxes
\$00 \$00 \$21	Orban
\$00 \$00 \$22	Indian Valley Mnfg.
\$00 \$00 \$23	Triton
\$00 \$00 \$24	KTI
\$00 \$00 \$25	Breakaway Technologies
\$00 \$00 \$26	CAE
\$00 \$00 \$27	Harrison
\$00 \$00 \$28	Future Lab
\$00 \$00 \$29	Rocktron
\$00 \$00 \$2A	PianoDisc
\$00 \$00 \$2B	Cannon Research Group
\$00 \$00 \$2C	
\$00 \$00 \$2D	Rogers Instrument Corp.
\$00 \$00 \$2E	Blue Sky Logic
\$00 \$00 \$2F	Encore Electronics
\$00 \$00 \$30	Uptown
\$00 \$00 \$31	Voce
\$00 \$00 \$32	CTI Audio / Intel
\$00 \$00 \$33	S&S Research
\$00 \$00 \$34	Broderbund
\$00 \$00 \$35	Allen Organ Co.
\$00 \$00 \$36	
\$00 \$00 \$37	Music Quest
\$00 \$00 \$38	Aphex
\$00 \$00 \$39	Gallien Krueger
\$00 \$00 \$3A	IBM
\$00 \$00 \$3B	
\$00 \$00 \$3C	Hotz
\$00 \$00 \$3D	ETA Lighting
\$00 \$00 \$3E	NSI
\$00 \$00 \$3F	Ad Lib
\$00 \$00 \$40	Richmond Sound Design
\$00 \$00 \$41	Microsoft
\$00 \$00 \$42	The Software Toolworks

Table 32 3-byte Manufacturer IDs (cont.)

<i>ID</i>	<i>Manufacturer</i>
\$00 \$00 \$43	RJMG/Niche
\$00 \$00 \$44	Intone
\$00 \$00 \$45	
\$00 \$00 \$46	
\$00 \$00 \$47	Groove Tubes / GT Elec.
\$00 \$00 \$4F	InterMIDI
\$00 \$00 \$55	Lone Wolf
\$00 \$00 \$59	Marion Systems
\$00 \$00 \$64	Musonix
\$00 \$00 \$65	Turtle Beach Systems
\$00 \$00 \$74	Ta Horng Musical Inst.
\$00 \$00 \$75	eTek (formerly Forte)
\$00 \$00 \$76	Electrovoice
\$00 \$00 \$77	Midisoft
\$00 \$00 \$78	Q-Sound Labs
\$00 \$00 \$79	Westrex
\$00 \$00 \$7A	NVidia
\$00 \$00 \$7B	ESS Technology
\$00 \$00 \$7C	MediaTrix Peripherals
\$00 \$00 \$7D	Brooktree
\$00 \$00 \$7E	Otari
\$00 \$00 \$7F	Key Electronics
\$00 \$01 \$01	Crystalake Multimedia
\$00 \$01 \$02	Crystal Semiconductor
\$00 \$01 \$03	Rockwell Semiconductur
\$00 \$20 \$00	Dream
\$00 \$20 \$01	Strand Lighting
\$00 \$20 \$02	Amek Systems, Ltd.
\$00 \$20 \$03	
\$00 \$20 \$04	Dr. Böhm/Musican Intl.
\$00 \$20 \$05	
\$00 \$20 \$06	Trident Audio
\$00 \$20 \$07	Real World Studio
\$00 \$20 \$08	Evolution Synthesis
\$00 \$20 \$09	Yes Technology
\$00 \$20 \$0A	Audiomatica
\$00 \$20 \$0B	Bontempi/Farfisa
\$00 \$20 \$0C	F.B.T. Elettronica
\$00 \$20 \$0E	LA Audio (Larking Audio)
\$00 \$20 \$0F	Zero 88 Lighting
\$00 \$20 \$10	Micon Audio

Table 32 3-byte Manufacturer IDs (cont.)

<i>ID</i>			<i>Manufacturer</i>
\$00	\$20	\$11	Forefront Technology
\$00	\$20	\$13	Kenton Electronics
\$00	\$20	\$15	ADB
\$00	\$20	\$16	Marshall
\$00	\$20	\$17	DDA
\$00	\$20	\$1F	t.c. electronic
\$00	\$20	\$20	Doepfer
\$00	\$20	\$29	novation
\$00	\$20	\$2B	Medeli Electronics Co
\$00	\$20	\$2C	Charlie Lab SRL
\$00	\$20	\$2D	Blue Chip Music Technology
\$00	\$20	\$2E	BEE OH
\$00	\$20	\$2F	LG Semiconductor
\$00	\$20	\$30	TESI
\$00	\$20	\$31	Emagic
\$00	\$20	\$32	Behringer

Appendix H

Conversion table

<i>Dec</i>	<i>Hex</i>	<i>Oct</i>	<i>Binary</i>	<i>ASCII</i>	<i>Dec</i>	<i>Hex</i>	<i>Oct</i>	<i>Binary</i>	<i>ASCII</i>
000	00	000	00000000		032	20	040	00100000	
001	01	001	00000001		033	21	041	00100001	!
002	02	002	00000010		034	22	042	00100010	"
003	03	003	00000011		035	23	043	00100011	#
004	04	004	00000100		036	24	044	00100100	\$
005	05	005	00000101		037	25	045	00100101	%
006	06	006	00000110		038	26	046	00100110	&
007	07	007	00000111		039	27	047	00100111	'
008	08	010	00001000		040	28	050	00101000	(
009	09	011	00001001		041	29	051	00101001)
010	0A	012	00001010		042	2A	052	00101010	*
011	0B	013	00001011		043	2B	053	00101011	+
012	0C	014	00001100		044	2C	054	00101100	,
013	0D	015	00001101		045	2D	055	00101101	-
014	0E	016	00001110		046	2E	056	00101110	.
015	0F	017	00001111		047	2F	057	00101111	/
016	10	020	00010000		048	30	060	00110000	0
017	11	021	00010001		049	31	061	00110001	1
018	12	022	00010010		050	32	062	00110010	2
019	13	023	00010011		051	33	063	00110011	3
020	14	024	00010100		052	34	064	00110100	4
021	15	025	00010101		053	35	065	00110101	5
022	16	026	00010110		054	36	066	00110110	6
023	17	027	00010111		055	37	067	00110111	7
024	18	030	00011000		056	38	070	00111000	8
025	19	031	00011001		057	39	071	00111001	9
026	1A	032	00011010		058	3A	072	00111010	:
027	1B	033	00011011		059	3B	073	00111011	;
028	1C	034	00011100		060	3C	074	00111100	<
029	1D	035	00011101		061	3D	075	00111101	=
030	1E	036	00011110		062	3E	076	00111110	>
031	1F	037	00011111		063	3F	077	00111111	?

<i>Dec</i>	<i>Hex</i>	<i>Oct</i>	<i>Binary</i>	<i>ASCII</i>	<i>Dec</i>	<i>Hex</i>	<i>Oct</i>	<i>Binary</i>	<i>ASCII</i>
064	40	100	01000000	@	096	60	140	01100000	`
065	41	101	01000001	A	097	61	141	01100001	a
066	42	102	01000010	B	098	62	142	01100010	b
067	43	103	01000011	C	099	63	143	01100011	c
068	44	104	01000100	D	100	64	144	01100100	d
069	45	105	01000101	E	101	65	145	01100101	e
070	46	106	01000110	F	102	66	146	01100110	f
071	47	107	01000111	G	103	67	147	01100111	g
072	48	110	01001000	H	104	68	150	01101000	h
073	49	111	01001001	I	105	69	151	01101001	i
074	4A	112	01001010	J	106	6A	152	01101010	j
075	4B	113	01001011	K	107	6B	153	01101011	k
076	4C	114	01001100	L	108	6C	154	01101100	l
077	4D	115	01001101	M	109	6D	155	01101101	m
078	4E	116	01001110	N	110	6E	156	01101110	n
079	4F	117	01001111	O	111	6F	157	01101111	o
080	50	120	01010000	P	112	70	160	01110000	p
081	51	121	01010001	Q	113	71	161	01110001	q
082	52	122	01010010	R	114	72	162	01110010	r
083	53	123	01010011	S	115	73	163	01110011	s
084	54	124	01010100	T	116	74	164	01110100	t
085	55	125	01010101	U	117	75	165	01110101	u
086	56	126	01010110	V	118	76	166	01110110	v
087	57	127	01010111	W	119	77	167	01110111	w
088	58	130	01011000	X	120	78	170	01111000	x
089	59	131	01011001	Y	121	79	171	01111001	y
090	5A	132	01011010	Z	122	7A	172	01111010	z
091	5B	133	01011011	[123	7B	173	01111011	{
092	5C	134	01011100	\	124	7C	174	01111100	
093	5D	135	01011101]	125	7D	175	01111101	}
094	5E	136	01011110	^	126	7E	176	01111110	~
095	5F	137	01011111	_	127	7F	177	01111111	

<i>Dec</i>	<i>Hex</i>	<i>Oct</i>	<i>Binary</i>	<i>ASCII</i>	<i>Dec</i>	<i>Hex</i>	<i>Oct</i>	<i>Binary</i>	<i>ASCII</i>
128	80	200	10000000		160	A0	240	10100000	
129	81	201	10000001		161	A1	241	10100001	
130	82	202	10000010		162	A2	242	10100010	
131	83	203	10000011		163	A3	243	10100011	
132	84	204	10000100		164	A4	244	10100100	
133	85	205	10000101		165	A5	245	10100101	
134	86	206	10000110		166	A6	246	10100110	
135	87	207	10000111		167	A7	247	10100111	
136	88	210	10001000		168	A8	250	10101000	
137	89	211	10001001		169	A9	251	10101001	
138	8A	212	10001010		170	AA	252	10101010	
139	8B	213	10001011		171	AB	253	10101011	
140	8C	214	10001100		172	AC	254	10101100	
141	8D	215	10001101		173	AD	255	10101101	
142	8E	216	10001110		174	AE	256	10101110	
143	8F	217	10001111		175	AF	257	10101111	
144	90	220	10010000		176	B0	260	10110000	
145	91	221	10010001		177	B1	261	10110001	
146	92	222	10010010		178	B2	262	10110010	
147	93	223	10010011		179	B3	263	10110011	
148	94	224	10010100		180	B4	264	10110100	
149	95	225	10010101		181	B5	265	10110101	
150	96	226	10010110		182	B6	266	10110110	
151	97	227	10010111		183	B7	267	10110111	
152	98	230	10011000		184	B8	270	10111000	
153	99	231	10011001		185	B9	271	10111001	
154	9A	232	10011010		186	BA	272	10111010	
155	9B	233	10011011		187	BB	273	10111011	
156	9C	234	10011100		188	BC	274	10111100	
157	9D	235	10011101		189	BD	275	10111101	
158	9E	236	10011110		190	BE	276	10111110	
159	9F	237	10011111		191	BF	277	10111111	

<i>Dec</i>	<i>Hex</i>	<i>Oct</i>	<i>Binary</i>	<i>ASCII</i>	<i>Dec</i>	<i>Hex</i>	<i>Oct</i>	<i>Binary</i>	<i>ASCII</i>
192	C0	300	11000000		224	E0	340	11100000	
193	C1	301	11000001		225	E1	341	11100001	
194	C2	302	11000010		226	E2	342	11100010	
195	C3	303	11000011		227	E3	343	11100011	
196	C4	304	11000100		228	E4	344	11100100	
197	C5	305	11000101		229	E5	345	11100101	
198	C6	306	11000110		230	E6	346	11100110	
199	C7	307	11000111		231	E7	347	11100111	
200	C8	310	11001000		232	E8	350	11101000	
201	C9	311	11001001		233	E9	351	11101001	
202	CA	312	11001010		234	EA	352	11101010	
203	CB	313	11001011		235	EB	353	11101011	
204	CC	314	11001100		236	EC	354	11101100	
205	CD	315	11001101		237	ED	355	11101101	
206	CE	316	11001110		238	EE	356	11101110	
207	CF	317	11001111		239	EF	357	11101111	
208	D0	320	11010000		240	F0	360	11110000	
209	D1	321	11010001		241	F1	361	11110001	
210	D2	322	11010010		242	F2	362	11110010	
211	D3	323	11010011		243	F3	363	11110011	
212	D4	324	11010100		244	F4	364	11110100	
213	D5	325	11010101		245	F5	365	11110101	
214	D6	326	11010110		246	F6	366	11110110	
215	D7	327	11010111		247	F7	367	11110111	
216	D8	330	11011000		248	F8	370	11111000	
217	D9	331	11011001		249	F9	371	11111001	
218	DA	332	11011010		250	FA	372	11111010	
219	DB	333	11011011		251	FB	373	11111011	
220	DC	334	11011100		252	FC	374	11111100	
221	DD	335	11011101		253	FD	375	11111101	
222	DE	336	11011110		254	FE	376	11111110	
223	DF	337	11011111		255	FF	377	11111111	

Appendix I

Glossary

This glossary describes the most important technical terms which occur while defining an Adaptation.

1's complement method for calculating checksums

14 Bit HL a →transmission format

14 Bit LH a →transmission format

2's complement method for calculating checksums

4 Bit HL a →transmission format

4 Bit LH a →transmission format

7 Bit HL a →transmission format

7 Bit LH a →transmission format

7 Byte Bitfield a →transmission format

8x7 Bit packed a →transmission format

Adaptation a file which adapts the Universal Module for a certain model

Adaptation editor a window which is used for editing an →Adaptation

ASCII abbreviation for »American Standard Code for Information Interchange«: internationally used assignment of characters (letters, digits, punctuation marks etc.) to binary codes

ASCII Hex a →transmission format used by Yamaha

Bank a block of entries in a device

Bank Dump a SysEx message which transfers the data of a bank

Bank Request a SysEx message which requests the data of a bank

Bank Select a standardized MIDI message (Control Changes 0 and 32) which is used to preselect a bank before a Program Change message

Bank driver component of an →Adaptation which defines the display and MIDI communication of a →bank.

Bank numbering component of a →bank driver which defines how the entries of a →bank are numbered.

big endian (vs. little endian) method for arranging several bytes in order to display numbers with large value ranges. The least significant byte has the highest address.

Bit smallest information unit. May have the value 0 and 1

BNK a →pseudo byte symbolizing the data of a bank

Bulk Dump SysEx message which transfers the whole memory contents of a device

Byte information unit, consisting of eight →bits. A byte can display 256 different values

Card →cartridge

Cartridge external memory media which can be inserted in a slot

Checksum component of →dump messages (more seldom of →Parameter Change messages and →Request messages) which serves to find out if a data transmission was correct

Checksum type one of the several methods of how to calculate and transmit a →checksum

CHK a →pseudo byte, symbolizing the →checksum

Command ID component of a SysEx message which determines the message's effect

Conversion transfer of data formatted in one →data type to another data type

Conversion step component of a →conversion table

Conversion table component of an →Adaptation which defines the →conversion between two →data types

Data Transfer Mode special mode in some older MIDI devices which allows the transmission and reception of the devices' memory contents

Data byte byte in a MIDI message which has the →MSBit cleared

Data type component of an →Adaptation which defines the data type of a device (e.g. Program, Patch, Multi, Combi etc.)

Device ID component of a SysEx message which serves to separate several devices of the same model

Device Number →device ID

Device Scan component of an →Adaptation which allows SoundDiver's →Scan function to automatically recognized a connected device

Dump SysEx message which transmits the contents of an entry, a bank, or the whole memory of a device

Edit buffer temporary buffer used for editing an entry. The editing effect is usually immediately audible

Entry a data block of a certain →data type in a device

EN# a →pseudo byte symbolizing the number of the entry in the bank

EN# Offset Format format which is used for transmitting the number of the entry in the bank

End of Exclusive Status byte which terminates a SysEx message

EOX →End of Exclusive

Handshake method for SysEx transmission where the receiver confirms the correct transmission or reports errors

Hexadecimal (vs. decimal, binary, octal) form of displaying numbers, using a base of 16

Help file a file (Atari file ending **.ADH**) which is created from a →source file by →SSHC and is read by SoundDiver to display help pages.

Hi nibble bits 4 to 7 of a →byte

HyperLink →reference

Icon graphic symbol in a →help file for commonly used note types

IMA →International MIDI Association

Index alphabetically sorted list of all →keywords in a →help file. Thanks to the special file structure, each help file automatically contains an index.

Individual Parameter Changes special address area in newer Roland devices which allows individual remote editing of single parameters even if they are arranged in bit fields in the dump

Industry standard name for computer systems which work with the MS-DOS operating system and Intel processors

International MIDI Association public head organization of the MIDI manufacturers

Japanese MIDI Standards Committee (vs. MMA) non-commercial organization of Japanese MIDI manufacturers which has the enhancement of the MIDI standard as its goal

JMSC →Japanese MIDI Standards Committee

Keyword The first paragraph of a →page in a →help file. May consist of several words. Each occurrence of the keyword in the help file (but not on the same page) is automatically marked by →SSHC (→reference). The keyword is displayed with a frame around it.

LH Nibbles → 7 Bit a →checksum type

LH Nibbles → LH a →checksum type

Little endian (vs. big endian) method for arranging several bytes in order to display numbers with large value ranges. The least significant byte has the lowest address.

Lo nibble the bits 0 to 3 of a →byte

Loop here: definition of a repeated execution of a partial MIDI string

Manufacturer ID component of a SysEx message which defines that the following data must be interpreted by the manufacturer's definition

Memory location component of a →bank in a device

MIDI abbreviation of »Musical Instrument Digital Interface«

MIDI Manufacturers Association (vs. JMSC) non-commercial organization of American, European and Australian MIDI manufacturers which has the enhancement of the MIDI standard as its goal

MIDI string component of a →bank driver or →object, consisting of one or more MIDI messages

MMA →MIDI Manufacturers Association

Mode Change MIDI message which changes the current mode of a device

Model ID component of a SysEx message which specifies the model of a manufacturer

MSBit the most significant bit of a →byte or →word

Nibble four consecutive bits

Nybble →nibble

Object a component of an →editor, e.g. text, box, slider, flip menu etc.

Offset a) constant value which is added to a variable

b) Position of a byte in a data block, counted from 0

One Way (vs. Handshake) method of SysEx transfer where the receiver sends no acknowledge messages

Page a text of random length in a →help file. Each page has a →keyword

Paragraph sequence of characters in a →source file, which is terminated by a blank line or \n

Parameter component of an →entry's data which has a certain effect on the entry's sound

Parameter Change SysEx message which individually changes a →parameter

PAU a →pseudo byte, symbolizing a pause or delay of certain length

Program Change MIDI message which selects a memory location

Reference occurrence of a →keyword in another →page of a →help file. In SoundDiver, references are printed in bold face and underlined. Clicking it leads to the appropriate page.

Request SysEx message which causes a device to send an appropriate →dump

Roland SysEx standardized SysEx format of Roland, specially supported by the Universal Module

ROM abbreviation of »Read Only Memory«. Here: name for memory locations which cannot be changed

Sample Dump Standard a Method standardized by the →MMA for manufacturer-independent transmission of samples

Scan function a special feature of SoundDiver for automatic installation of all connected devices

Selection column a column in the →Adaptation editor and →Object editor used for selection blocks

SDS →Sample Dump Standard

SIN a →pseudo byte, symbolizing the transmission of the data of a single entry

Single Dump SysEx message which transmits a single →entry

Single Request SysEx message which requests a Single Dump

Source file a file which is converted into a →help file by →SSHC. Source files are stored in readable ASCII format and can be created by any ASCII text editor.

Status byte first byte in a MIDI message. The →MSBit is always set.

SSHC Abbreviation of »SoundSurfer Help Compiler«

STO a →pseudo byte symbolizing the entry's data size

SUM a →pseudo byte symbolizing the beginning of checksum calculation

SysEx →System Exclusive

System Exclusive part of the MIDI standard which allows manufacturers to extend it individually to their needs

Technical Standards Bulletin Board publication by the →MMA where future extensions of the MIDI standard and »recommended Practices« are discussed and passed

Time-out name for the fact that a device did not answer on a →Request within a certain period

TRA a →pseudo byte, symbolizing the number of bytes which are necessary to transmit the entry's data

TSBB →Technical Standards Bulletin Board

Transmission format one of the several methods to transfer 8-bit data using only 7 bits per byte. See also →data byte

Universal Device Inquiry standardized method to recognize connected devices. See also →Device Scan and →Scan function

Universal SysEx Device Inquiry →Universal Device Inquiry

Word (vs. byte) information unit, usually consisting of 16 →bits

Write Request SysEx message which causes the device to store an →edit buffer into a certain →memory location

Index

Symbols

162
 # of bits 188
 # of entries 135, 145
 # of rows 187
 ++ 162
 .ADA 166, 167, 168, 221
 .ADT 221, 247
 .BAT 242
 .PA 168, 169
 \ (Backslash, Space) 229
 \! 227
 \{ 231
 \} 231
 \| 230
 \\\ 230
 \e 228
 \f 225
 \i 228
 \m 228
 \n 226
 \r 227
 \w 229

Numerics

0 Offset 188
 1's Complement 34, 149
 14 Bit Contr., Integer steps 194
 14 Bit Controller 194
 14 Bit HL 139
 14 Bit LH 139, 141
 2's complement 38
 2's Complement 34, 149
 2's complement 190
 300 101
 4 Bit HL 151
 4 Bit LH 151
 7 Bit 30, 139, 141
 7 Bit and 1+7 Bit mixed 32
 7 Bit Contr., Integer steps 194
 7 Bit Controller 194
 7 Bit Hex 35, 152
 7 Bit HL 151

7 Bit LH 151
 7 Byte Bitfield 140, 141
 7+1 Bit 141
 7up 220
 8x7 Bit packed 31, 139, 140, 141
 9010 97

A

absolute 163
 Access PC 166
 ACED 160
 Active Setup 160
 ADA 166, 167, 168, 221
 Adaptation 95
 • edit 181
 • new 44, 58, 65, 103
 • save 182
 Adaptation editor 103
 • Window title 103
 Add Address Mapping Table 128
 Add Memory Offset 183
 Address Base 152
 Address Map 34, 153
 Address Mapping 128
 Address mapping table
 • new 107
 ADT 221, 247
 After Dump 150, 154, 155, 156
 Akai 29, 113, 143
 Alesis 141, 150, 151
 Alesis Quadverb 151
 Aligned 154
 All Black 199
 all data dump 29
 All White 199
 Alpha Juno 1/2 39, 125, 141
 AMEM 160
 Amiga 38
 Analyze 216
 ANSI C 20
 Answer 1 120
 Apple Events 243
 Apple Macintosh 38
 AppleScript 243
 ARGV 248
 Arrow 199, 218
 ART 141
 ASCII 25, 109, 125
 ASCII Hex 31, 139, 141

Atari 166
Atari ST/TT 38
author 117, 237
auto 133, 134
AutoLink 146
AutoRequest 144
AutoSurf 143, 157

B

Background object 171
backslash 225
bank 29

- copy 54

Bank driver 60, 62, 132

- define 48, 60, 62
- new 107

bank dump 29, 52, 155
Bank dump data 97
bank name 235
bank numbering 61, 136
Bank Request 53, 155
Bank Select 30, 146
BANK-LSB 148
Bankmanager 43, 57
BANK-MSB 148
Base Address 152
Basic MIDI Channel 27
Before Request/Dump 150, 154, 155, 157
big endian 37, 191
Binary display 24
Binary Offset 38
Binary View 179
bisection 250, 251
bit 25
bit 0 25, 162
bit 15 25
bit 31 25
bit 7 25
bit field 37, 172, 189

- define 188

block alignment 224
BNK 97, 138, 212
Border 183
Boss 128, 141
bottom-aligned with 134
BPRD 40
BPRR 40
bulk dump 29
Bulk Load 152
byte 25

C

cabling 234
Card 29, 142, 167, 236
Card names 117
Card switch 236
Cartridge 29, 54, 142, 236
centered 197
CHAN 147
Checksum 33, 98
Checksum format 33
Checksum type 123
CHK 51, 98, 123, 149, 214
Clear 106, 178
clipboard 104
Color 184
Command byte 150
command ID 28
command lines 248
command shell 247
COMMAND.PRg 219, 247
comment 230
Compile 241
compression 244, 249
Const 207, 211
Control Channel 147
Controller 194
Controller, integer steps 194
Conversion 237

- mode of operation 163

Conversion step 64, 161
Conversion table 160

- define 63
- new 108, 160
- structure 160

Convert Files 170
Copy 106, 177
Copyright 238
Credits 237
Cubase DMM 39
cursor 104
Cut 106, 177
Cypress 220

D

D-10 160
D-110 29, 30, 34, 37, 57, 113, 132, 153, 160
D4 150
D-5 141, 160
D-50 37, 39, 114, 126, 151
D-550 126
D-70 141

DAT 116
 data byte 26, 96
 Data Decrement 217
 Data Increment 217
 Data size 60, 123

- stored bytes 101
- transmitted bytes 101
- variable 123

 data transfer mode 152
 Data type 122, 133, 234, 236

- define 47
- new 107

 data type 235
 Default Names 145, 218
 Default Play Delay 114
 Default Send Pause 99, 114
 Default Timeout 114
 definition blocks 105
 Dest. Type 161
 device family code 119
 Device ID 27, 96, 234

- +1 113
- Min/Max 113
- offset 113

 device ID 235
 Device Inquiry 118
 Device Number 27
 Device Parameter box 114, 115, 117, 144, 236
 Device Scan 118
 Digit / Zero 136
 Digitech 196
 Direction 210
 Distance 153
 DosMounter Plus 166
 DR-X 141
 D-Series 141
 DT1 34, 116
 Dump 29, 158

- request 29

 DX/TX 42
 DX7 39, 41, 114, 141, 160
 DX7II 141, 156, 159

E

E-5, E-10, E-20, E-30 141
 Edison 220
 Edit buffer 29, 55
 Edit menu 106
 Editable 138, 142, 172
 Editor 171

- new 172

 Editors 171

EM7F 166
 EMA5 166
 EMA6 166
 E-mu 32, 37, 98, 113, 126, 135, 141, 149
 EN# 50, 98, 150, 158, 213
 EN# Format 150
 EN# Offset 150
 EN# offset 168
 EN# Offset Format 98
 encoding 38
 End of Exclusive 26
 Ensoniq 32, 42, 100, 126, 141, 148, 149, 151, 190, 212
 Ensoniq EPS 151
 Ensoniq EPS 12 Bit 140, 141
 Ensoniq EPS 16 Bit 140, 141
 Entry 186, 218
 Entry Dependency Management 166
 entry number in a bank 98
 enumeration 227, 232
 Envelope 205
 environment variables 247
 EOX 26
 EPS 32, 141, 151
 ESQ-1 100, 112, 141, 190
 ESQ-M 190
 Everest 220
 Export Names ... 234
 Export Names... 221
 EXT 141

F

Fade in/out gadget 103
 family member code 119
 fat binary 221
 FB-01 100
 file name 111, 246
 Fill 183, 184
 finite automata 249
 Flatten 181
 Flip Menu 185
 folder 246
 Format 186, 200, 202, 218
 Frontier 243
 Function 204

G

Gemini 219, 247
 Generic 124
 Generic SysEx 124
 Geräte-Parameterbox 236

Global MIDI Channel 27
 Global parameters 44, 58, 110, 118
 Glossary 287
 GR-50 141, 160
 Grid 199
 Grid Snap 180
 GS-Standard 141
 Gulam 219, 247

H

H 186, 199
 H/V title 61, 138
 Handling 202
 Handshake 116, 152, 235
 HDAT 242, 248
 HELP 248, 251
 Help compiler 219
 Help files
 • conventions 233
 Help Lines 206
 help system
 • mode of operation 251
 Hexadezimal 23
 hi-nibble 30
 HL-nibbles 30
 Horizontal Slider 202
 hyper link 223
 Hypertext 223

I

Icon 114
 icon 227, 228, 232
 IFF format 250
 IMA 26
 Image 198, 218
 Indentions 239
 index 98, 223, 226, 237
 Individual Parameter Changes 39
 industry standard 38
 Info line 109
 Initialization message 46
 Initialize 162
 Input status enable 114, 124
 insertion point 104
 • set 105
 Install 110
 Install Application 221
 Installation 234
 Instrument 148
 Intel 80x86 38
 Internal Tones 62

International MIDI Association 26
 Inverse 197
 Inverted 185, 197
 Item Spec 187

J

Jump 163

K

K1 28, 43, 97, 141, 149, 150
 K1200 150
 K4 137, 141, 149
 K5 141, 150
 Kawai 28, 43, 97, 137, 141, 149, 150
 Kawai K1/K4 34, 149
 Kawai K5 34, 150
 Key Remote command 157
 Key/Velocity window 208
 Keyboard 208, 218
 Keyboard range 211
 keyword 223, 225, 234
 Knob 203, 218
 Konvertierung 237
 Korg 31, 42, 137, 141, 149, 156, 217
 KS-32 126
 Kurzweil 150

L

LANG 247
 Layout Mode 180
 Leading Zero 136
 least significant bit 25
 leftalign 197
 left-aligned with 133, 134
 Lexicon 31, 38, 98, 100, 101, 115, 137, 141, 149, 159, 191
 LH Nibbles -> 7 Bit 149
 LH Nibbles -> LH 34, 149
 LH-nibbles 30
 Library 111
 little endian 37
 Local menu 107
 LOGIC 146
 long word 25
 lo-nibble 30
 loop 100
 Loop end 100, 163
 Loop start 100, 162
 LS Bit 25, 188
 LXP-1 38, 98, 141, 159, 191

LXP-15 160
LXP-5 100, 137, 141, 159, 160

M

-m 244
M1 141, 156, 217
M1R 141
M3R 141
Machine ID Acknowledge 46
Machine ID Request 46
Macintosh 166
MacLink Plus Translators 170
macro 249
mantissa 191
manufacturer ID 26, 96, 110, 275
Map 202
Masterkeyboard 114
Matrix 126, 149
Matrix series 39
Matrix-1000 98, 120, 156, 159, 160
Matrix-1000/6/6R 141
Matrix-6 120, 160
Maximum 185, 201
MEM 97, 192, 212
Memory location 143, 187
Memory Manager 235, 243
Memory occupied by parameter 97
Michael Haydn 19
microWave 29, 40, 113, 132, 137, 141
MIDI 235
MIDI communication 95
MIDI Manufacturers Association 42
MIDI Monitoring 205, 209
MIDI string 95, 109, 154
• enter 51
Minimum 185, 201
minus sign 38
MK-80 190
MKS-100 141
MKS-50 125, 141, 190
MKS-70 141
MKS-80 141
MMA 42
mode 152, 156
Mode Change 156
Model ID 27
model name 234
most significant bit 25
Motorola 680x0 38
MotU 116
MSBit 25, 30
MT-32 113, 141

Multi 56, 213
Multi Instrument 148
multi-timbral 147
Multi-value object 171
MultiVerb 141
Mupfel 219, 247

N

-n 244
Name bytes contain data 128
Name format 125, 145
Name offset 125
Name size 125, 145
negative numbers 38
New
• address mapping table 107
• bank driver 107
• conversion table 108
• data type 107
New data type 122, 132
Nibble 30
Nibble HL 139, 141
Nibble LH 139, 141
Nibble LH (Word) 140, 141
nibbles 30
No Checksum 149
No Request 153
Normal 202
Notator RMG 39
Number of bytes 161, 162
Numerical value 146, 200, 218
nybbles 30

O

Oberheim 39, 98, 120, 126, 141, 149, 156
object 171
• change size 173
• move 173
Object Editor
• open 181
Object editor 182
Object Link 209, 211
Object Snap 180
Octal display 25
offset 39, 163, 164
One Way 116, 152
OP 207, 211
Options file 242
Overwriting memory locations 143

P

PA 168, 169
 Packed ASCII 125, 126
 packet 138
 Packet Size 151
 page 223
 Page Jump 158
 paragraph 223
 parameter 36, 171
 Parameter Change 29, 39, 212
 Parameter descriptions 240
 Parameter group 237
 parameter name 236
 Parameter value 96
 parser 249
 Paste 106, 178
 PAU 99, 115, 157, 158, 214, 215
 Pause 99
 PC Exchange 166
 PCM-70 38, 191
 Play Delay 114
 Polyframe 168, 183, 187
 position 133, 134
 Positioning 208, 211
 Power Macintosh 221
 preferences 143
 printed manuals 238
 PRO/CUSSION 135
 PRO-E 141
 Program Change 40, 146, 156
 Program Change detection 146
 programmer 19, 20
 Prophet V 26
 Proteus 32, 37, 113, 125, 126, 141
 Pseudo byte 212
 pseudo byte 95, 96
 PureC 20, 220
 PurePascal 220

Q

QED 220
 Quadraverb 140, 141, 151

R

R-5 141, 191
 R-8 141, 153
 RA-50 141
 Radio button 185
 Range 206, 210
 Reciprocal 207, 211

Redo 106, 177
 Register 160
 Regular Checksum 34, 149
 relative 163
 remote control 214
 Remove help 241
 Repeat 214
 Repeated characters 239
 Request 29, 158
 Request 1 120
 Request Retries 115
 Rhodes 190
 rightalign 197
 right-aligned with 133
 Roland 29, 32, 37, 39, 42, 57, 113, 114, 115, 128, 132, 137, 138, 141, 149, 189, 190, 191
 Roland Mode 124
 Roland mode 116, 128, 215
 Roland Model 58, 116
 Roland SysEx 34, 58, 116, 149, 215
 ROM 29
 ROM location 187
 Rotary knob 203
 RQ1 116
 RQD 116

S

-s 245
 S-10 34, 116
 S1000 29, 143
 Sample Dump Standard 32
 Save
 • ~ Adaptation

Adaptation

Save ~ 108

Save device entries when quitting 144
 Scan 234
 Scan function
 • define 46, 59
 screenset 204
 Script 220
 scripting 243
 SD-1 141
 SDS 33
 SE-50 141
 Select
 • definition block 105
 Select All 107, 178
 Selection 104
 selection column 105, 206
 Send immediately 204

Send Pause 99, 114
 Sequential 26
 Setup window 144
 SGE 141
 Shadow 185
 sign bit 191
 Sign Magnitude 38
 Sign magnitude 190
 Signum 220
 SIN 51, 97, 123, 138, 212
 single dump 29, 154
 Single dump data 97
 Single Request 51, 154
 Slider 202, 217
 Snap to Grid 181
 Sound Canvas 191
 SoundDiverBox 18
 source file 223, 247
 Source Type 161
 Special Parameters box 167, 168, 236
 Special parameters box 117
 SQ-1 141
 SQ-1/2/R 126
 SQ-80 112, 190
 SSHC 192, 219

- mode of operation 248
- options 243
- preferences 242
- run 241

 Standard keywords 234
 Standard parameters 149
 Status byte 26, 95
 STO 101, 124
 Studio 400 197
 Studio Quad 196
 Style 204
 SUM 51, 99, 149, 158, 213
 Sum up from here 99
 Switch 203
 SY/TG series 123, 124
 SY/TG55 101
 SY/TG-Reihe 141
 SY22 39
 SY77 27, 29, 143, 156, 189, 213, 217
 SysEx 26

- message types 29

 SysEx Communication Error 235, 243
 SysEx Kommunikationsfehler 235, 243
 SysEx reception 234, 235
 System 7 108, 243
 system clock 244
 System Exclusive 26

T

T1, T2, T3 141
 table 227, 232
 Technical Standards Bulletin Board 42
 Tempus 220
 text 39, 198
 Text Length 201
 text size 184
 Text value 146, 201, 218
 Text/Box 197, 218
 TG33 39, 100
 TG77 148
 Think C 20
 Thru channel 114
 Thru Channel = Device ID 114, 147
 Timeout 114
 to the right of 133, 134
 Toggle selection 178
 Tone Temporary 60
 top-aligned with 134
 TRA 101, 124
 Transfer 161
 Transmission format 30, 61, 138, 216
 TSBB 42
 Turbo C 220
 TX802 100, 141, 159
 Type 202

U

U-110 141
 U-20 32, 62, 154, 191
 U-20/220 141
 U-220 154
 under 134
 Undo 106, 177
 Universal Device Inquiry 46, 59, 121
 Universal SysEx Device Inquiry 118
 Universal SysEx Device Inquiry Message 118
 Use for Scan 59, 120

V

VAL 96, 192, 212, 216
 Value limitation 210
 Value object 171
 variable 123
 variable channel 96
 VCED 39, 160
 Velocity range 211
 VFX 141, 148, 212
 Vintage Keys 98, 112, 126

VMEM 39, 160
Voice 212

W

W 186, 199
Waldorf 29, 40, 113, 132, 137, 141, 149
Wavestation 141
Wildcard 248
Windows 166
word 25, 32
Word Plus 220
word wrapping 223
word-wrap 197
WP Mode 220
Write Request 157

X

X 133, 186, 199
Xpander 141, 191

Y

Y 133, 134, 186, 199
Yamaha 27, 39, 42, 100, 101, 109, 114,
123, 124, 141, 143, 148, 149,
156, 189, 213, 217

Z

Zoom 97, 198