INTRODUCING THE MIDI

In less than two decades the music synthesiser has developed from its monophonic origins into a fully integrated microprocessor controlled instrument. During this time manufacturers have all had their own ideas of how control voltage and/or trigger signals should be implemented. This has made it almost impossible to interface directly between different machines, without some form of conversion circuitry to cope with incompatible signal levels or polarities.

The fabrication of integrated circuits dedicated to electronic music production by Solid State Music (SSM) and Curtis Electro-Music (CEM) has helped this situation. Control voltages of 1V/Octave and positive going triggers between 5 and 15V have become the standard for any manufacturer using these devices. However, this does not help in the case of the current microprocessor controlled polyphonic machines. The complex algorithms used prohibit direct connection and each manufacturer again tends to produce dedicated interfaces which connect to his own specific controllers.

The problems of compatibility along with the advent of the home computer, providing its musically creative possibilities, has forced instrument manufacturers to finally make their products comply to an industry standard specification and thereby protect their equipment from obsolescence.

The Musical Instrument Digital Interface (MIDI) is such a specification, which has been developed by leading manufacturers in the last few years. It does not dictate instrument design but merely specifies a language which carries meaningful information between instruments.

What does this all mean to the musician? The purpose of the specification is to allow synthesisers, other electronic keyboards, sequencers, drum machines and home computers to be linked in one programmable system. Useful lifetime of equipment is therefore also multiplied. Some of the exciting possibilities are as follows:

Synthesisers can be configured 'in parallel' with instruments played simultaneously or remotely.

Entire compositions, consisting of monophonic and polyphonic sequences and rhythm can be played at a touch.

Computer terminals can be used for composing, sequence creation and editing.

Graphic quality printers can produce the 'hardcopy' manuscript of an improvisation or composition.

Video synthesis can be integrated with music synthesis.

Musical education such as reading music, scale recognition, and ear training can be automated.

History

SCI Digital Interface

Sequential Circuits Inc first became interested in microcomputer interfacing in conjunction with the design of the Prophet-10 polyphonic and its internal polyphonic sequencer. The Prophet and its sequencer each were based on Z-80 microcomputers. To record, as notes were played, every few milliseconds (at a rate set by the sequencer clock), the Prophet would send its complete keyboard 'status' to the sequencer. The sequencer had to figure out which notes were going on and off, and record these events in reference to the clock count. On playback, the sequencer computer also sent the complete keyboard status every clock pulse, with events as counted out by the clock. The Prophet would play these notes just as if they came from its own keyboard. Later, this sequencer was made available as an accessory for the Prophet-5. The Prophet-5 Remote Keyboard was also developed which used this interface. SCI Published the data protocol upon which this interface was based, in the hopes that the programming public would be encouraged to develop their own interfaces for the Prophet-5.

This did not occur, apparently because in being conceived for a specific application, the interface was very fast but too clumsy for general-purpose use. It was criticised as requiring too much programming 'overhead,' in the constant transmission of meaningless keyboard information. As a result of this experience, SCI resolved to pursue a more streamlined interface that would be easier for programmers to work with.

Universal Synthesiser Interface

In the meantime, occasional discussions between the presidents of Sequential Circuits, Oberheim Electronics and Roland (Dave Smith, Tom Oberheim and Ikutaroo Kakehashi) also revealed a shared interest in the interface problem and development of an interface widely acceptable to the industry.

Smith then outlined a specification for a 'Universal Synthesiser Interface' (USI). It was developed with the assistance of SCI's Chet Wood and presented at the Autumn, 1981 convention of the Audio Engineering Society (AES).

The USI differed markedly from the earlier SCI Digital interface in that rather than being polled at the sequencer clock rate, information was only sent when an event actually occurred - for example, a note going on or off. The USI was proposed to be serial, operating at 19.2 kBaud, with TTL levels, and connected through phone jacks.

After incorporating changes in response to comments from AES, Smith sent a questionnaire to all manufacturers and industry consultants he could find, asking for their suggestions and any special requirements. There was a strong response to this initiative; some saying, for example, that it would not be possible to do it serially, that a parallel interface was necessary. Others thought the proposed serial speed too fast for operation with home computers. Many other issues were raised.

All respondents were invited to a conference in coincidence with the January, 1982, Western National Association of Music Merchants (NAMM) convention in Anaheim. This meeting was attended by representatives from SCI, Roland, Oberheim, CBS/Rhodes, Yamaha, E-mu, Unicord (Korg), Music Technology Inc., Kawai, Octave Plateau, Passport Designs and Syntauri. Other manufacturers seemed to be maintaining a 'wait-and-see' policy.

At this meeting the chief changes which occurred to the USI were to add optoisolation to prevent audio ground loops, and to increase the speed to 31.25 kBaud.

Japanese Interface Proposal

Following the USI discussion at Anaheim, an alternative specification was presented by some of the Japanese companies which had grown out of their own research. Whereas the USI was basically content to specify note on/off codes, this new proposal went on to define many more complex operations. It also offered a different data structure, with status and data bytes being flagged by bit 7 (1=status, 0=data). This greatly simplified the protocol by eliminating all the checks which were other-

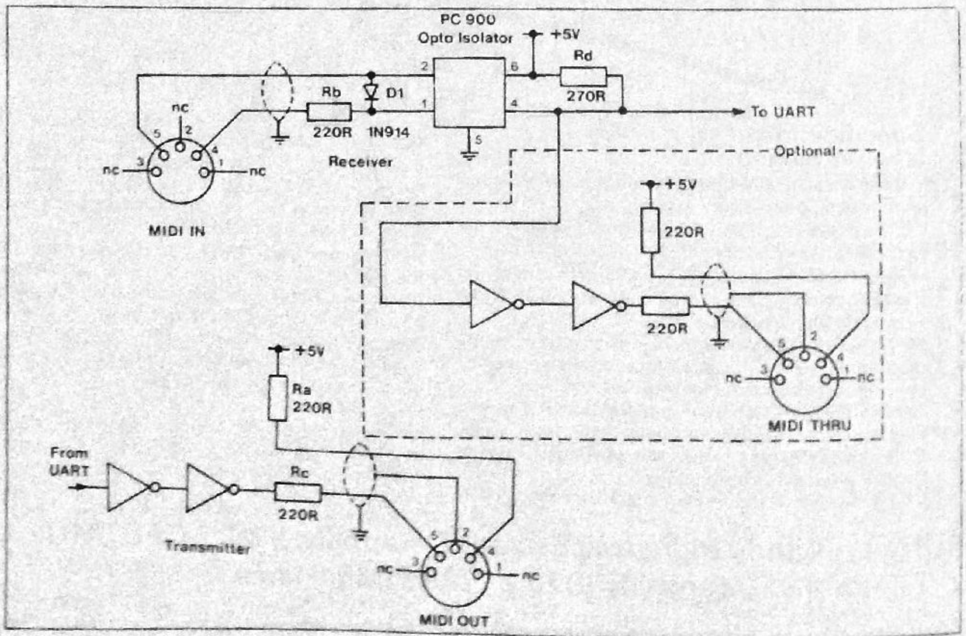

Figure 1. MIDI Hardware Schematic.

wise needed to distinguish the data category. With the most significant bit now defined as a 'flag', data is thereby limited to 7 bits, but this is sufficient for most synth data, and when not, can simply be sent as multiple 4-bit nibbles.

MIDI

After the Anaheim meeting, Smith and Wood integrated the USI and Japanese proposals, forming the first MIDI specification. This was sent to all of the meeting participants but, curiously, provoked no further comment from this continent. The final document was therefore arrived at after several exchanges between SCI and Roland, which is serving as liaison with Yamaha, Korg and Kawai.

Hardware

To simplify cabling between instruments, the interface is serial. It operates at 31.25 kBaud (thousand-bits-per-second), asynchronous. This is considered a high speed for serial operation - in comparison to the typical RS-232 maximum of 19.2 kBaud - but' was chosen to prevent objectionable delays between equipment. The 31.25 kHz clock can also be easily obtained from hardware, for example, by dividing 1 Mhz by 32. One serial data byte consists of a start bit, 8 data bits (D0 to D7), and a stop bit - for a total of 10 bits transferred in 320 microseconds (us).

Physically, MIDI appears as two or three jacks on the instrument. See Figure 1, the hardware schematic. The connectors are DIN 5-pin (180 degree) female panel mount receptacles. DIN connectors were agreed to by US manufacturers because it was felt that DIN connectors are now widely available here. However, the specification *does* provide that a manufacturer can use XLR connectors, if the firm makes available all necessary conversion cables.

The two required jacks are MIDI OUT and MIDI IN. The transmitter data typically originates in the instrument's UART. The interface circuit is a 5-mA current loop, designed especially to prevent the formation of audio ground loops which often develop in complex systems. The output is normally meant to drive only one input. If transmit data is low (0), current flows from Vcc (+5V) through Ra, over pin 4 of both connectors, through the opto-isolator, returns over pin 5, then through Rc. The opto-isolator output is normally pulled high by Rd. However, when current flows through the internal LED, the isolator output switch turns on, grounding Vo, thus sending a low to the receiver UART. When data is high, the LED does not light. The receiver UART therefore sees a high. D1 protects the opto-isolator from reverse-polarity currents which may result from transmitter anomalies.

Interconnecting cables should not exceed fifty feet (15 meters), and must have a corresponding 5-pin DIN male plug. The cable should be shielded twisted pair, with the shield connected to pin 2 at both ends. Notice that while the MIDI OUT jack is grounded to the instrument chassis, MIDI IN is not. This allows the cables to provide their shielding services without creating ground loops.

The optional third jack, MIDI THRU, provides a direct copy of data coming in MIDI IN. It is included when the manufacturer intends the instrument to operate in a 'chain' or 'loop' network, as opposed to a 'star' network.

Modes and Channels

The first thing to realise about MIDI is that the total control features available still depend on the design of each specific piece of equipment. MIDI does not magically transcend equipment limitations or differences. Rather it merely enables them to 'communicate' at their 'least common' level. For example, specific programmed sounds can't be transferred directly between different models of synthesisers because of inherent differences, but keyboard information and program *selections* can be communicated.

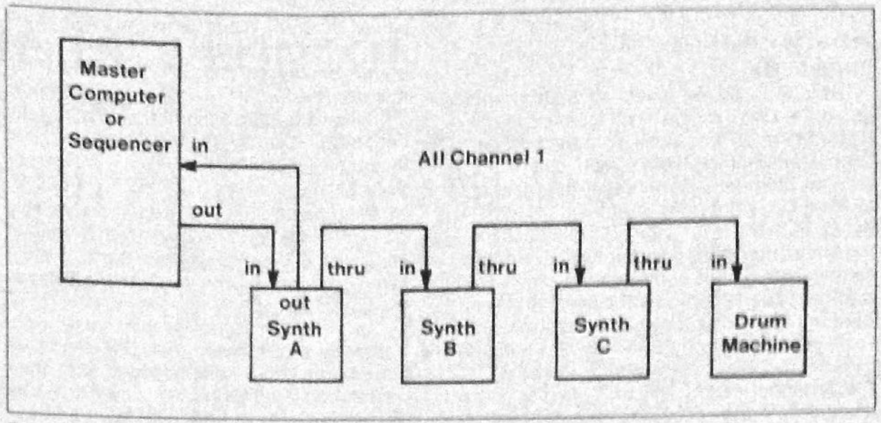

Figure 2. Omni-mode chain network.

One of MIDI's design goals was to be simple enough so that you could connect any polyphonic synthesiser to any other, or to a sequencer, and at the very least the notes would be correctly played or stored. This would be possible with virtually no other action on the part of the user. Above this minimum, each instrument may or may not include further facilities for complex control options.

Each type of equipment has different minimum requirements. For synthesisers, minimal usefulness seems to include remote control and program switching. While polyphonic sequencers send and receive keyboard data, they may or may not be interested in program changes. Monophonic sequencers can only deal with individual lines, so keyboard data must somehow be different for them. Drum units don't usually care about specific keyboard notes, but may need to synchronise to their timing, or to the sequencer, and perhaps react to program changes as well.

While most of these requirements and useful control options can be foreseen, the number of possible interconnections cannot. Therefore, while the specification says that each transmitter will drive one and only one receiver, provision has been made so that any specific instrument or synthesiser voice on the MIDI bus can be addressed, regardless of the interconnection scheme. This is accomplished by assigning up to 16 channels under increasingly powerful (and complex) modes.

Each unit connected to the MIDI bus has separate transmit and receive ports. There are three modes of operation for transmitters and receivers: Omni, Poly and Mono.

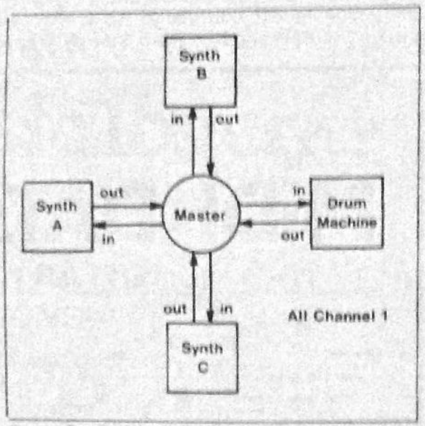

Figure 3. Omni-mode star network.

Omni mode is the most general level of operation, interfacing all units. Poly mode allows each unit (synth, sequencer, or drum box) to be addressed separately. Mono mode is the most specialised, allowing individual addressing of (for example) each synthesiser voice.

Normally, transmitters will periodically send out a Mode Select command for the most powerful mode to which they can be configured. However, the actual data transmitted will be in the mode to which a second transmitter may have switched the receiver. For example, Synth A by default transmits in Omni mode to Synth B. Synth B, being capable of Poly mode operation, periodically transmits Poly Mode Select codes to Synth C. But the *data* sent from Synth B to C will be in Omni format (because Synth B's receiver is constantly getting Omni Mode Select commands from Synth A). Synth C may or may not respond to the Poly Mode Select commands from Synth B, because if a receiver is capable of operating in the requested mode, it switches to that mode. Otherwise, it ignores the Mode Select command. (Note, the Mode Select commands double as 'All Notes Off' commands, therefore can only be

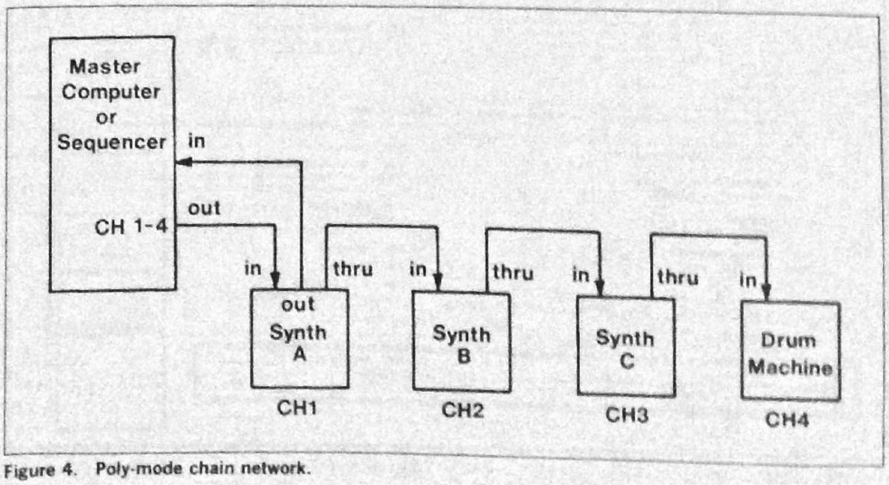

Figure 4. Poly-mode chain network.

sent while all notes are off, or when it is desired to turn all notes off).

Omni Mode

At power up or reset, all instruments default to Omni mode. See Figures 2 and 3. Regardless of the system configuration, Omni transmitters always send polyphonic data on Channel 1. Omni receivers respond to Note On/Off Events sent over *any* channel (1-16). These notes are handled according to the internal assignment scheme of the synthesiser. So this configuration allows any number of polyphonic synthesisers to play in parallel, as soon as they are interconnected.

A receiver's mode can only be changed by a Mode Select command transmitted in the channel(s) to which it is currently assigned. If the receiver is not capable of operating in the requested mode, it ignores the Mode Select command. No unit may switch its own modes. Even though a receiver in Omni mode receives in all channels, it will respond to Mode Select commands in only one channel: the one to which it is assigned.

Receivers and transmitters without channel selection capability are always assigned by default to Channel 1.

Poly Mode

Poly mode allows individual addressing of each unit. In other words, the master controller can send separate parts to each synth, whereas in Omni mode they all played the same part.

As shown in Figure 4, the master controller in the chained network sends all commands, which are encoded with their destination channel number, over one line. This requires each unit include an address selector switch to define its channel of operation.

The channel definitions having been made, the master controller must issue the command to the receiver on that channel to switch to Poly mode. Thereafter, the receiver listens for keyboard data encoded with its channel number. Any number of notes can be sent, to which, again, the polyphonic synth will respond according to its own priorities.

Poly mode will be useful for sequencing multi-part arrangements of standard synths, for example, which can't be done in Omni mode.

Mono Mode

When a synthesiser has Mono capability, and it receives a Mono Mode Select command, it configures itself to receive on the channel it is assigned to and above, up to the number of voices it has. For example, the Prophet-T8 in Mono mode will transmit and receive on Channels 1-8. (Future synthesisers could contain more elaborate channel selection capability).

Channeling each voice provides fast transfer of individual pressure (also called 'after touch') data for each key. It also makes true *legato* possible, because the note value (=voice pitch) can be changed without having to first turn the note off (as in Poly mode).

Data Format

There are five categories of MIDI data: Channel, System Common, System Real Time, System Exclusive and System Reset.

Each data category encompasses a number of 'status bytes' which define specific commands under that category, and which precede data bytes which specify the exact operation. Status bytes are distinguished from data bytes according to whether the most significant (MS) bit is set (1=status) or reset (0=data). The status bytes under each category are defined below. Note that any data sets (eg. Note On event data) which are sent successively under the same status, can be sent without a status byte until a different status byte is needed.

Channel information performs most of the routine work. Commands are addressed to specific channels by a 4-bit number which is encoded into the status byte. The associated data bytes can identify keys going down (on) and up (off), their on or off velocities, and pressure or 'after-touch' (on keyboards so equipped).

System Common, Real Time, and Reset information is intended for all channels in a system. System Common information identifies song selections and measure numbers for all units. Real Time information is used for synchronizing everything (perhaps to a master sequencer). Therefore, Channel and System Common information is interruptible by System Real Time information.

System Exclusive information allows the exchange of data which can be formatted as the manufacturer wishes. Only devices which recognise the manufacturer's format will attend the exchange.

Reset simply initialises all equipment to power-on condition.

The five categories are ordered in Table 2 according to their utility.

Conclusions

The MIDI is one of the most important and powerful developments in electronic music technology. Not only does it allow machines to communicate with each other but allows the instrument to become a peripheral for a computer system, unleashing the tremendous power that such a set-up can offer. **E&MM**

Our thanks to Sequential Circuits Inc., for allowing us to reproduce text from their MIDI specification.

MIDI DATA FORMAT

Channel

The most significant four bits of each Channel status byte define the command, while the least significant four bits identify the affective channel.

9xH NOTE ON EVENT
3 bytes: 1001 nnnn + 0kkk kkkk + 0vvv vvvv

nnnn
Channel code: 0-15. Corresponds to channel numbers 1-16.

kkk kkkk
Key number: 0-127.
For all keyboards: middle C=60. All C key numbers are multiples of 12.
The standard five-octave synth keyboard ranges 36-96.
The 88-note piano keyboard ranges 21-108.

vvv vvvv
Key On velocity: 0-127.
With no velocity sensors, default to 64.
With velocity: 1=ppp (softest), 127=fff (loudest).
Key On velocity=0, turns note off.

8xH NOTE OFF EVENT
3 bytes: 1000 nnnn + 0kkk kkkk + 0vvv vvvv

vvv vvvv
Key Off (release) velocity.
Implemented on Prophet-T8.

AxH POLYPHONIC KEY PRESSURE
3 bytes: 1010 nnnn + 0kkk kkkk + 0vvv vvvv

vvv vvvv
Pressure/After touch value: 0-127.
Used in Omni mode. (Compare code DxH, Mono mode).

BxH CONTROL CHANGE
3 bytes: 1011 nnnn + 0ccc cccc + 0vvv vvvv

ccc cccc
Control address: 0-127.
Except for the Pitch Bender (0), the controllers are not specifically defined. A manufacturer can assign the logical controllers to physical ones as necessary. The controller allocation table must be provided in the user's operation manual. Continuous controllers (including the Pitch bender) are divided into Most and Least Significant Bytes. If only 7 bits of resolution are needed for a specific controller, only the MSB is sent. It is not necessary to send the LSB. If more resolution is needed, then both are sent, first the MSB, then the LSB. If only the LSB has changed in value, the LSB may be sent without re-sending the MSB.

0	Pitch bender MSB
1	Controller 1 MSB
2	Controller 2 MSB
3	Controller 3 MSB
4-31	Continuous controllers 4-31 MSB
32	Pitch bender LSB
33	Controller 1 LSB
34	Controller 2 LSB
35	Controller 3 LSB
36-63	Continuous controllers 4-31 LSB
64-95	Switches (on/off)
96-123	Undefined
124	Local/Remote Keyboard Control (toggle)
125	Omni Mode Select/All notes off
126	Mono Mode Select/All notes off
127	Poly Mode Select/All notes off

If c=125, 126, or 127, v (see below) must be 0

vvv vvvv
Control value: 0-127.
For mode selections (c=125, 126, or 127), vvv vvvv must be 0.
Pitch benders should range from 0-127, with 64 being centre (no pitch bend).
Other controllers will range from 0=minimum to 127=maximum.
Switches are defined 0=off, 127=on.

CxH PROGRAM CHANGE
2 bytes: 1100nnnn + 0ppp pppp

ppp pppp
Program number: 0-127.

DxH CHANNEL PRESSURE
2 bytes: 1101nnnn + 0vvv vvvv

vvv vvvv
Channel pressure/after touch amount: 0-127.
For Mono mode: channel (rather than key) is identified.

ExH UNDEFINED
(SCI uses this status for Pitch Wheel change in the Prophet-600).

System Exclusive

A format has been defined for System Exclusive information, consisting of a two byte preamble, the data itself, and a one-byte end code. The purpose of this format is to provide for the transmission of data which may be useful to any two instruments from one manufacturer but uninterpretable to other MIDI-bussed devices. For example, SCI uses this protocol for loading and dumping program data. System Exclusive information can only be interrupted by a System Reset command.

Format: F0H + 0iii iiii + data + F7H

F0H
Status byte. Must be followed by manufacturers ID#.

0iii iiii
Manufacturers ID#.
iii iiii can be 0-127.
Current ID numbers are:

Sequential Circuits	01H
Kawai	40H
Roland	41H
Korg	42H
Yamaha	43H

Receivers which do not recognise the ID# ignore the ensuing system exclusive data.

data
Any number of bytes.
MSB must be reset. (Otherwise will signal a new status byte). Data can range 0-127.
Data is intended for all channels.

F7H
An END-OF-BLOCK code which terminates System Exclusive status.
SYSTEM RESET will also terminate System Exclusive status.

In no case should other data or status codes be interleaved with System Exclusive data, regardless of whether or not the ID code is recognised.

System Real Time

The System Real Time codes control the entire system in real time. They are used for synchronising sequencers and rhythm units.

To maintain timing precision, these codes can be sent between any System Common or Channel data sets which consist of two or more bytes. However, the codes may not be interleaved with System Exclusive data.

System Real Time statuses are intended for all channels and recognised by all units using the interface. If the functions specified are not implemented, they are simply ignored.

F8H TIMING-CLOCK-IN-PLAY
This clock is sent while the transmitter is in Play mode. The system is synchronised with this clock which is sent at a rate of 24 clocks/quarter note.

F9H MEASURE-END
The MEASURE-END is transmitted instead of the TIMING-CLOCK-IN-PLAY at the end of each measure.

FAH START-FROM-1st-MEASURE
This code is immediately sent when the PLAY button on the master (e.g. sequencer or rhythm unit) is hit. The first TIMING-CLOCK-IN-PLAY must follow within 5 ms after this code.

FBH CONTINUE START
This is sent when the CONTINUE button (on the master) is hit. A sequence will restart from the point where the sequence stopped on the last TIMING-CLOCK-IN-PLAY. The next TIMING-CLOCK-IN-PLAY must be sent within 5 ms after this code.

FCH TIMING-CLOCK-IN-STOP
This code is clocked in Stop mode to synchronise a Phase Locked Loop (PLL) which is used (during Stop) for interpolating the timing clock.

System Common

System Common information is intended for all channels in a system.

F1H Undefined

F2H MEASURE INFORMATION
3 bytes: F2H + 0mmm mmmm (MS) + 0mmm mmmm (LS)
The two data bytes code the 14 bit measure number.

F3H SONG SELECT
2 bytes: F3H + 0sss ssss
The data byte codes the 7 bit song number.

F4H Undefined

F5H Undefined

F6H TUNE REQUEST
Initiates synthesiser tune routines.

System Reset

There is one system reset code. It initialises the entire system to the condition of just having power switched on.

FFH SYSTEM RESET

System Reset should be used sparingly, preferably under manual command only in particular. It should not be sent automatically on power up. This could cause two units connected together to endlessly reset each other.

Table 1. Data Format.