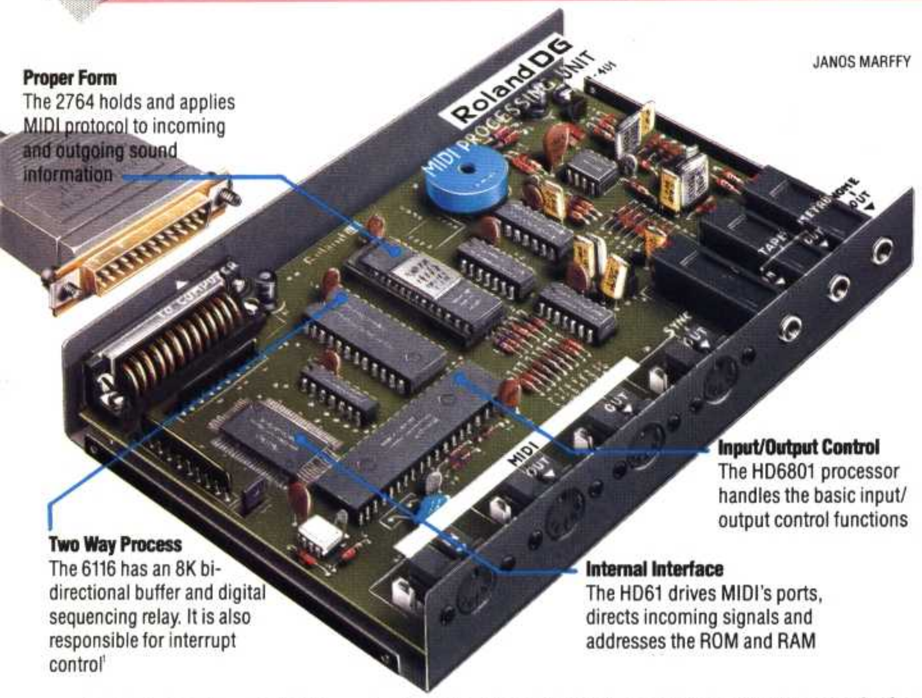

Proper Form
The 2764 holds and applies MIDI protocol to incoming and outgoing sound information

Input/Output Control
The HD6801 processor handles the basic input/output control functions

Two Way Process
The 6116 has an 8K bi-directional buffer and digital sequencing relay. It is also responsible for interrupt control'

Internal Interface
The HD61 drives MIDI's ports, directs incoming signals and addresses the ROM and RAM

The Musical Instrument Digital Interface
The MIDI interface reconciles the input/output protocols of the computer and musical instruments connected to it — just like any other interface — thus allowing the instruments to use the computer's memory. It also processes the digitised sound passed through it, adding control, sync and timing information to the synthesiser input

instruction is not given, and the rest of the melody, E, G, etc, is entered without duration parameters, all of the notes of the melody will continue to sound. The result will be a sustained chord made up of the notes of the melody.

Fortunately, even a slight acquaintance with the stave notation on the VDU will be enough to indicate that what was intended to be a melody seems likely to become a chord. And an MCL user who had made such an error could simply run the sequence, but this time send a MONOPHONIC instruction to the synthesiser. *Monophony* simply means one sound as opposed to many (*polyphony*). If the receiving synthesiser can produce only one sound at a time when set to MONOPHONIC, then the poorly-entered sequence can be performed as a succession of single notes only — a tune, rather than a chord.

Let us suppose that, after some trial and error, the first-time user has entered his tune accurately, and the interfaced synthesiser is now playing. The melody keeps time, and the rhythm — defined by the sequence of durations — is correct. It should be noted that so far in our discussion the types of instruction have been fairly limited. Only two musical parameters or characteristics have been called upon — pitch (middle C, E, G, etc.) and duration.

The composer of the melody listens to it a few times, then decides it sounds rather 'stiff' — as it is likely to do while it has only a minimum of definition. He decides that, instead of the first C and E occurring one after the other, the pitch of the C should glide upwards to the start of the E. This sort of movement is called a *glissando* or pitch bend, and would be characteristic of the way a person might whistle the tune. In this context, it might add a touch of jauntiness to the synthesiser's performance. So this instruction now replaces the original instruction for middle C, adding an extra byte.

This brings us to a simple point concerning MIDI interfacing. If the receiving synthesiser has no facility for producing *glissandi* (bending pitch), it cannot carry out this last instruction. It may perform the middle C as if it were receiving the *original* instruction, or it may do something else entirely. If a MIDI user's instructions are to produce a section of polyphonic music, and the receiving synthesiser is only a monophonic instrument, it will probably make an unpredictable selection from the polyphony, and then perform monophonically. In short, using MIDI to link a microcomputer to a very basic synthesiser will not turn it into an expensive synthesiser like the Fairlight.

These restrictions also apply in reverse. The receiving instrument may be a superb £10,000 synthesiser, but unless sufficient musical parameters have been defined, and unless the synthesiser's own controls have been set up as desired, the result may well be performed with the musicality of a pocket calculator.

In practice, the second of these two situations is easily improved. As many parameters as possible should be set as *constants* using the synthesiser controls, and the MIDI instructions should work within those parameters. This approach is the one most likely to be adopted by the synthesiser player whose problems we considered earlier.

So far, we have looked at pitch and duration characteristics, but MIDI provides for 128 theoretical controls, covering filtering, distortion, 'white noise' (all possible frequencies) and 'pink noise' (mid-range frequencies), each with values ranging from 0 to 128. This is more than adequate to deal with the parameters available on most synthesisers, and it is these controls that will probably interest microcomputer owners.

This is where the MIDI transmission rate becomes an issue. We have seen that a very straightforward command, concerning a single note and defining only two parameters, used three serial words. With the 31.25 Kbaud rate, this takes almost one millisecond. Six-note chords are common in many types of music: such a chord would take 5.76 milliseconds to transmit. If we now started to define this chord further using MIDI controls, the transmission time becomes slow enough for the human ear to begin to detect changes in the sound's characteristics caused by delay. These changes are apparent only when sounds, especially similar sounds, occur together — but as an audio interface MIDI was designed to handle simultaneous sounds. Music, unfortunately, is a 'parallel' medium: as listeners, we are used to hearing things happen simultaneously.

It is therefore not surprising that MIDI's serial transmission has been criticised; parallel transmission would have done the job better. It remains to be seen whether MIDI users are troubled by this failing. At present, the design of the interface appears to be a compromise between cost and efficiency, so it is worth remembering that the present specification may only be the first.