A DEMONSTRATION OF THE aXlO CONTROLLER

Brend Carion
The University of Calgary
Calgary, Alberta, Canada
carionb@facultycalgary.ca

ABSTRACT: The aXlO (alternative, eXpressive Input Object) is a new type of MIDI controller that provides unique physical interface to a variety of continuous and discrete dimensions of control. Each dimension of control can be configured to generate a MIDI message defined by the user.

The physical aspects of the aXlO controller have been described in detail in previous papers. (Carion, B. 1992, 1994) Three varieties of the aXlO controller are now in existence. These are identical, except for the physical interface for pitch selection. One employs a velocity sensitive chording keyboard as described in the preceding papers, a second has a two octave piano-type keyboard, which combined with the thumb activated octave switches gives a pitch range of six octaves, and the third controller has a number of touchpads that can be programmed in a variety of ways.

The electronics of the controller have undergone significant change and will be described in some detail here. Separate microprocessors are used for the pitch data generation tasks and for the switch and continuous control data generation. The piano-type keyboard uses electronics adapted from a commercial product and so will not be discussed. The chord keyboard uses an 8031 microprocessor and custom written software. Note and velocity MIDI data are generated, and keydown timing window (the period used to determine whether keypress are simultaneous or sequential) can be adjusted. The touchpads use a second unit of the 8031 board developed for the controller.

The 8011 board has five MIDI connections: two outputs, a programming or "boot" input, a thru connection for the programming input, and a MIDI merging input. Data generated by the analog inputs and the switches can be routed to either output. The programming input is exclusively used for sysEx messages. The MIDI thru output passes the sysEx messages and can be used to chain up a number of boards together. The boards have a set of jumpers that are used to determine the ID number (from 0 to 3). The merging input simply merges incoming MIDI data with the data generated by the board.

The board has eight 8-bit analog inputs and thirty-two switch inputs. Each analog input has an associated sample rate timer that can be set from 4 milliseconds to just over 1 second in 4 millisecond increments. The analog data is converted to digital data and reduced to 7 bits to reduce input jitter (very small fluctuations in the analog input signal causing undesired generation of MIDI data). The 7 bits of data (values 0 through 127) are processed using one of 8 user selectable tables (linear, reverse, logarithmic, exponential, etc.) to set the basic response or "feel" of the analog input. The data is then processed using a user defined table of 128 bytes. Each analog input can be translated into a one, two or three byte MIDI message. The primary use of the analog inputs is the generation of continuous control data. The MIDI channel and controller number of each output is also set by the user when used for this purpose. Data generated by the analog inputs is not confined to continuous control data; it can be transformed into any kind of MIDI data. As well, the analog inputs do not have to generate MIDI data directly. The values generated at one analog input can be used as part of the data portion of a message associated with another analog input. For example, one input could be used to set the controller number of a MIDI message while a second input is used for the controller value of the message. Multiple messages can also be generated by a single input: one input could generate volume, pitch-bend, and aftertouch data, for example.

ICMC PROCEEDINGS 1993 183
The switch inputs can also be programmed to send one, two or three byte MIDI messages. The closing and opening (or transition) of each switch can be programmed to send different messages. The switches can also be set to operate as momentary or push-on, push-off switches. In the momentary mode of operation, pressing a switch immediately sends one message and releasing the switch sends a second (possibly different) message. In the push-on, push-off mode pressing a switch toggles back and forth between two messages. The first press sends one message and releasing the switch sends nothing. Pressing the switch a second time sends the second message, pressing a third time sends the first message again, and so on. For example, a switch could send a sustain on message (B0 40h 00h) on the first press and a sustain off message (B0 40h 00h) on the second press. Pressing and releasing the switch would turn on the sustain and subsequently played notes would then sustain until the switch was pressed a second time.

The usefulness of the switches is greatly increased by having the ability to execute lists. Lists can be a set of instructions that act internally to re-program the functions of the electronics. As a simple example, imagine that four switches are programmed to send note-on messages when pressed. A fifth switch can execute a list of internal messages that will change the action of the other switches to send different note-on messages. Lists can also generate MIDI messages directly, such as an arpeggio triggered by a single switch. Lists can be executed successively (from the first entry to the last in sequence) or step-wise, one entry being executed for each switch press or transition.

The sysx programming messages take the form of [<F0h><manufacturer ID><device number>:<command><address><data><checksum><F7h>]. Data transfer to the electronics can handle records up to 256 bytes long. Data records of any length can also be downloaded from the electronics for examination. Both nibbled and non-nibbled data transfer is supported. The nibbled format is necessary to allow values greater than 127 (7Fh) for purposes such as setting the status byte of messages (for example, program change messages use status byte with values from 122 (COh) to 207 (CFh)). A handshake data transfer method using 8-into-7 data compression is also supported. Several means of long term data storage are provided, and up to 8 banks containing a complete programming setup are available.

The controller electronics are designed to be fast and responsive, and to provide maximum flexibility in terms of generating MIDI data. Each input can produce virtually any type of MIDI message desired. An editor to program the ax/60 has been developed using Opcode's MAX programming environment. The editor provides easy and convenient ways of manipulating individual or multiple bytes, tables, and lists of data in the electronics. Generated MIDI data can also be sent to the controlling computer for further processing, or for triggering other types of events such as MIDI sequences.

Musical applications of the ax/60 will be demonstrated and the flexibility of the electronics used in this project will be highlighted by a demonstration of a glove interface using the same electronics by Steve Miller, the designer and developer of the electronics.

ACKNOWLEDGEMENTS: This project was made possible by a grant from The Canada Council. I am indebted to David Eagle for his continuing enthusiasm and his use of the ax/60 controller. Very special thanks to Steve Miller, a very talented electronics designer and machine code programmer. The electronics originated in a project of Steve's, and he has devoted a great deal of time and energy to designing and programming the electronics used in the ax/60.