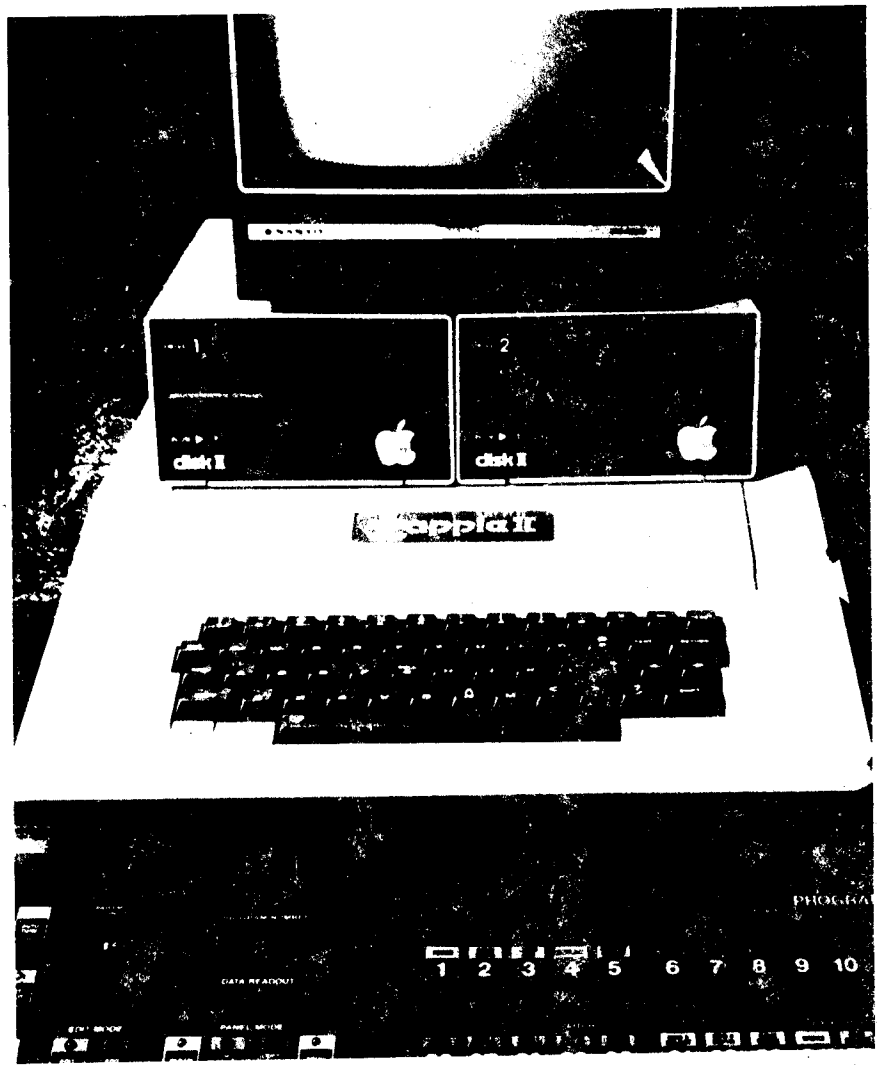
Rhodes

CHROMA
Interface Manual



CHROMA COMPUTER INTERFACE MANUAL

Table of Contents

INTRODUCTION

This section should serve as an overview of the Chroma computer interface and of the topics that will be covered in detail in the subsequent sections of this manual.

Purpose of the interface -- The Chroma computer interface is to allow the Chroma to be controlled by a computer. A Chroma can be controlled by another Chroma, but this is not the main intent of the interface.

Capabilities of the interface -- The interface allows a number of different functions to be performed.

A computer can "play" the Chroma's keyboard and performance controls.

A computer can record what a human plays on the keyboard and performance controls.

A computer can change sounds, or modify parameters in existing sounds.

A computer can record the changes made to a sound by a human manipulating the panel controls of the Chroma.

A computer can load or save packets of information using the cassette interface.

A computer can temporarily alter the workings of the Chroma by changing the firmware of the Chroma's internal computer. This facility is not likely to be very useful to anyone outside Rhodes, but may be used in future products designed to enhance the Chroma.

Physical nature of the interface -- The computer interface consists of two identical 8-bit parallel ports, one for each direction. Each port consists of 8 latched data bits, a status line that tells whether there is information on the port, and an acknowledge line which is pulsed whenever a data byte is read from the port. All signals are TTL compatible, which means that interface hardware is simple and cheap. However, this also means that the interface is not designed to work over great distances or in the presence of ground differences or large amounts of noise. Each device must have a set/reset flip-flop driving the status line for its output port. This flip-flop must be set when writing a byte to the port and cleared by the acknowledge pulse received from the other end. The acknowledge pulse will normally be the read pulse used to read the data from the port.

Interface protocol -- Communication in each direction is independent of communication in the other direction. The interface is best handled in an interrupt driven environment with a queue (first-in first-out list) for buffering the information in each direction (or at least in the input direction). There are two levels to the protocol, the physical and the logical. The physical level is kept simple by the fact that each byte transferred is acknowleged by the receiving end before another byte may be transferred. The logical level can be more complex because a communication that requests a response from the

other end doesn't necessarily receive the response immediately. In fact, several different requests may be transmitted before the responses start coming back.

Command language -- All communication issued by a computer to control a Chroma is in the form of commands. A command is a sequence of one or more bytes, where the first byte is the command code (similar to a machine language op-code) and the remaining bytes are operands. For instance, the command to set parameter 5 in program 9 to a value of 3 consists of the sequence 7, 9, 5 and 3. (7 is the command code for Write Parameter.) Certain commands require a response from the Chroma. In this case, the Chroma will respond by sending back what looks like a command that begins with the same command code. It isn't really a command, as the Chroma doesn't control the computer, but it takes a similar form. There are also commands to the Chroma that enable recording from the Chroma. When these commands are issued, the Chroma enters a mode in which it will send "commands" to the computer whenever a note is played or a control is moved. In a simple recording system, these commands may be played back verbatim at a later time. Thus, even though the Chroma doesn't actually tell the computer what to do, the communication in both directions is organized as commands.

The interface view of the Chroma -- The performer sitting at the Chroma "sees" a system that is capable of generating two sounds at a time using the Link capability. The computer connected to the Chroma "sees" a system that is capable of generating up to eight different sounds at a time using the "multiple instrument" capability. The reason that the performer isn't given this capability from the keyboard is that there is no easy and reliable way for a performer to "tell" the Chroma which notes go with which sounds, beyond a simple keyboard split. The computer interface doesn't have this limitation, as it can instruct the Chroma which sound to use with each note. The interface is designed to allow the performer to play one "track" at a time (or two, using a link) into a computer, and then the computer to play multiple "tracks" back as if there were multiple instruments being played.

The interface view of the Expander -- A Chroma Expander looks just like a Chroma through the computer interface, except that it is not capable of generating keyboard information. It should be used for playback only.

CHROMA STRUCTURE

The intent of this section is to describe the way the Chroma appears through the computer interface, and to relate the way a computer deals with the Chroma to the way a human performer or programmer deals with the Chroma. A computer has a much greater range of control over the instrument than a human, as it is capable of communicating with it much faster than a human could ever punch buttons. What a human sees of the Chroma's inner workings is a subset of what the computer sees, and it is important to understand both in order to fully exploit the capabilities of the interface.

The Human's View -- A person sitting at the Chroma and manipulating the controls on the Chroma's panel has a certain view of the inner workings of the Chroma. This view has the following characteristics:

There are fifty programs in the Chroma's memory that can be called upon at any time, numbered 1 through 50.

There is a fifty-first program (program 0) that controls the sound of the instrument.

Selecting a program causes one of the fifty stored programs to be copied into program 0, and storing a program causes program 0 to be copied into one of the fifty storage locations.

All creating and editing of programs is done to program 0, and then stored someplace else.

A second sound, called a link, can be added to the main sound. The link sound is controlled by the link program, which is one of the fifty stored programs. A parameter in program 0 "points to" the link program.

Certain other parameters in program 0 affect the link. These include the Keyboard Split, Link Mode (lower, unison or upper), Transposes and the Link Balance.

Since the link is one of the fifty stored sounds, it cannot be directly edited.

Changing a parameter from the panel causes program 0 to be modified, and the sound reflects this change. The original program from which program 0 was copied is not affected.

The performance controls affect both the main and link sounds. The notes played on the keyboard are assigned to the main and/or link sounds according to the link mode and keyboard split.

The Computer's View -- A computer "looking into" the computer interface port on the Chroma has a much more detailed view of the inner workings of the Chroma than a person sitting at the controls. The most important concept that must be understood when working with the computer interface is the concept of an "instrument". In a physical sense, the Chroma itself is obviously an instrument. Here, though, we are concerned with a more symbolic interpretation of the word. In this sense, the Chroma is capable of containing more than one "instrument" as it is capable of generating more than one sound at a time. To understand this, study the following definitions:

Channel -- A channel is the basic unit of sound generation, consisting of an oscillator, wave shaper, filter, amplifier, glide, sweep and two envelopes (if patch 0 is selected), or consisting of two oscillators, wave shapers, filters, amplifiers, glides and sweeps, and four envelopes (if patch 1-15 is selected).

Board -- A board is the physical entity containing the synthesizer circuitry. A board contains two channels if patch 0 is selected, or one channel if patch 1-15 is selected. If patch 0 is selected, the two halves of the board will always have the same sound characteristics. The Chroma has eight boards.

Program -- A program is a set of parameters that completely defines a sound, plus a few parameters that represent settings of certain panel controls (such as the edit mode and link information). The battery backup memory in the Chroma contains 51 programs.

Instrument -- An instrument is a group of boards that is treated as a logical entity, and whose sound is defined by a particular program. There can be as many as eight instruments defined in the Chroma at any one time, numbered 0 through 7. Each instrument has its own set of performance control and keyboard "inputs" that function independently of the inputs to other instruments.

The notion of an instrument makes sense if you consider the Chroma as a small ensemble. Through the computer interface one might play a piece of music involving a guitar, a bass, and two trumpets. This would be done by defining four instruments by sending the appropriate Define commands to the Chroma, and then sending commands to attack and release notes to the individual instruments within the Chroma. The two trumpet instruments would presumably be defined by the same program, emphasizing the fact that a program and an instrument are two different animals.

Translating The Human's View To The Computer's View -- When controlling the Chroma from its control panel and keyboard, two sounds can be played at a time. Internally this means that, you guessed it, two instruments can be defined. The various actions that can be performed from the panel translate into internal manipulations of programs and instruments as follows:

Program Select Without Link -- The selected program is copied into program 0. Then instrument 0 is defined according to program 0 and instrument 1 is undefined.

Program Select With Link -- The selected program is copied into program 0. Then instrument 0 is defined according to program 0 and instrument 1 is defined according to the link program.

Link -- Instrument 0 is left unchanged and instrument 1 is defined according to the specified link program. The Link Mode and Link Program parameters in program 0 are set accordingly.

Unlink -- Pressing [NO LINK] twice causes the link to be cleared by undefining instrument 1. The Link Mode parameter in program 0 is set to "No Link".

"No Link" Program Select -- Pressing [NO LINK] followed by a numbered switch causes the selected program to be copied into program 0, except that the link related parameters are not copied. Instrument 0 is then defined according to program 0 and instrument 1 is left alone.

Keyboard Information -- When a key is pressed and no link is in effect, the key number is transposed according to the Main Transpose parameter in program 0 and sent to instrument 0. If a link is in effect, the key number is sent to instrument 0 and/or 1, depending upon the Link Mode and Keyboard Split parameters in program 0. Information sent to instrument 1 is transposed according to the Link Transpose parameter in program 0. All key press information that is sent to either instrument is recorded, along with its transposition, in a key list. When a key is released, it is looked up in the key list and the transposition recorded there is sent to the appropriate instrument(s). Thus, the Chroma won't be confused by changing transpositions or keyboard splits while keys are held down.

Pressure Sensor Information -- If the Pressure Sensor option is installed, varying the pressure on any key causes the appropriate information to be sent to instrument 0 and/or instrument 1, depending upon the key's entry in the key list. That is, if the note's attack was sent to an instrument, subsequent pressure information will also be sent there.

Performance Control Information -- When a lever, pedal or footswitch is moved, the appropriate information is sent to instrument 0 and, if a link is in effect, to instrument 1.

Parameter Changes -- Changing any of the numbered parameters causes the appropriate values in program 0 to be altered. Whenever a program is altered in any way, any instruments defined by that program are automatically affected. Changing any of the panel parameters does not

directly affect any instruments, although it may, as in the case of the Link Mode parameter, indirectly affect either instrument 0 or 1.

Thus, the panel controls manipulate instruments 0 and 1, using instrument 0 for the main sound and instrument 1 for the link sound. Instruments 2 through 7 are not affected, and are always undefined when the instrument is first turned on.

Board Allocation and Channel Assignment -- Board allocation is the process of deciding which boards are assigned to which instruments. This process is repeated any time an instrument is defined or undefined. Channel assignment is the process of deciding which channels are assigned to which notes. This process occurs dynamically as notes are played and released. These two processes are separate, yet they do affect each other:

The channel assignment is controlled by the Keyboard Algorithm parameter, which determines whether the instrument is polyphonic (using multiple channels) or monophonic (using a single channel). This choice affects board allocation in that a monophonic instrument is never assigned more than one board.

The channel assignment is independent for each instrument and must use only those channels assigned to that instrument by the board allocation process. Defining or undefining an instrument (or changing between a mono and a poly keyboard algorithm) affects the number of boards available to other instruments and, as such, impacts the channel assignment of other instruments. Undefining an instrument generally makes channels available to other instruments, while defining an instrument usually robs channels from other instruments.

The board allocation process operates by first calculating how many boards should be allocated to each instrument and then boards are taken from instruments that have too many and given to instruments that have too few. Unaffected boards can continue to generate sound during this process. In addition, the board robbing is intelligent enough to favor boards that are not currently sounding. Calculating the number of boards each instrument should have is done in a round-robin manner, like dealing cards. The rules are:

The number of boards in the system is usually eight, but may be smaller if the autotune detected a bad board.

An undefined instrument doesn't get any boards.

An instrument whose Keyboard Algorithm parameter is 5 or more is considered monophonic, and is assigned a maximum of one board. (If patch 0 is selected, only one of the two channels will be used.)

An instrument whose Keyboard Algorithm parameter is 4 or less is considered polyphonic, and may be assigned any number of boards. (If patch 0 is selected, there are twice as many channels available as there are boards.)

The dealing of boards to instruments continues until all boards are used up, or until all instruments have been checked and none of them are polyphonic. In the latter case, there may be boards left over.

Temporary Parameter Changes -- When a performer selects a program and then alters a parameter, this change is recorded in program 0 but not in the original stored program. Although program 0 is in fact stored in the same battery-backup memory as the other fifty programs, it is, by conventional usage, a temporary program. During performance, one may modify a parameter in a program, but this change is usually not stored. Therefore, every time the same program is selected, the changes made last time it was selected are usually not still there.

The intent of the computer interface is to allow several "performances" to be recorded separately and then played back simultaneously. Since there is only one "program 0", another means is provided for making temporary changes to the several sounds that may be selected at once. Whenever an instrument is defined according to a particular program, a translation occurs. The information that makes up a program is very different from the information recorded for each instrument. A program consists of 59 bytes in which all the parameters are packed as tightly as possible to conserve memory. An instrument is represented by a couple hundred bytes of information in which the parameters are expanded into a form that allows for fast processing. There are separate commands for changing parameters in a program and changing parameters in an instrument. The Write Parameter command is used to alter the value of any parameter in any program, and its effect is permanent. If any instrument happens to be defined by that program, it will be affected too. The Set Parameter command, however, only alters the translation of the parameter in the instrument and doesn't affect the program that defines the instrument. If the Set Parameter command is used to alter parameters, these changes will be temporary and will not show up again the next time an instrument is defined by the same program. Note also that, while the Write Parameter command has a complementary Read Parameter command, the Set Parameter command has no complement.

Instrument Definition Parameters -- Whenever an instrument is defined through the computer interface, the performance control inputs must be initialized. To this end, the Define command includes operands that represent the positions of the levers, pedals and footswitches. There is also a volume operand that acts as a master volume control. The volume is also controllable separately through the use of the Volume command. An instrument's volume is only accessible from the panel through the use of the link balance control.

When recording from the Chroma, the computer will receive Define, Undefine and Volume commands from the Chroma whenever programs are selected, linked or unlinked. The performance control operands in the Define commands will reflect the true physical position of the performance controls at that time. The Link Balance and Link Mode parameter determines the volumes of the instruments as follows:

If no link is in effect, instrument 0 will be defined with a volume of 255 (maximum).

If a link is in effect, instruments 0 and 1 will be defined with volumes determined by the Link Balance parameter.

When a link is cleared, in addition to sending an Undefine command for instrument 1, the Chroma will send a Volume command for instrument 0 reflecting the fact that its volume is now 255.

When a link is set up, in addition to sending a Define command for instrument 1, the Chroma will send a Volume command for instrument 0 reflecting the setting of the Link Balance.

When the Link Balance parameter is changed, volume commands will be sent by the Chroma for both instruments 0 and 1 if a link is in effect. If no link exists, no volume commands will be sent.

It should be clear by now that most of the communication with the Chroma is actually communication with individual instruments within the Chroma. The command set, listed in the next section, shows this. Some commands (like Read and Write Parameter, mentioned earlier) aren't associated with an instrument, but most of the commands involved in recording and playing back music are addressed to individual instruments within the Chroma.

COMMAND DESCRIPTIONS

The computer that is connected to the Chroma via the interface cable communicates with the Chroma by sending and receiving commands. A command consists of:

A byte that specifies the command. If the command applies to one of the eight "instruments" within the Chroma, the instrument number will be encoded in this byte, too.

Zero or more bytes that specify parameters of the command. Although most commands require specific numbers of parameters, a few commands are variable in length.

Certain conventions are adhered to in the command language:

Undefined commands are considered to be No Operation commands; that is, undefined commands are ignored. All No Operation command have no parameters.

Command code zero and command code FF (hex) will always be No Operation commands, even for future instruments that utilize this interface.

Command code 1 will allways be an Identification command, for this and any other instrument utilizing this interface.

If a two-byte quantity (such as a memory address) is to be transferred, it will be sent most significant byte first, just the way you would write it on paper.

If a command is variable in length, the second byte of the command will specify the variable number of data bytes. This is not the same as the length of the command, as the count does not include the command code, the length byte, or any other fixed parameters for the command. The Peek command is a good example of this.

If a command is variable in length, the second byte of the command will specify the length as follows: values 1 to 255 represent byte counts of 1 to 255, and a value of zero represents a byte count of 256.

Any command that could conceivably "crash" the Chroma through misuse will not be allowed until a special "unlock" command is first issued. This minimizes the chance of a crash if the Chroma should receive garbage from a malfunctioning computer.

The commands fall roughly into three categories, according to protocol:

There are those commands that are issued by the controlling device and processed by the Chroma with no response.

There are those commands that are issued by the controlling device and require a specific response from the Chroma. The response will always be a "command" starting with the same code that was received from the controlling device.

There are those commands that establish modes within the Chroma that allow the Chroma to subsequently transmit unsolicited "commands" when certain events occur. The unsolicited commands will generally look like commands from the first group above.

The command set can also be split into two categories, according to destination:

There are those commands that are addressed to the Chroma as a whole. The lower command codes are assigned to these commands.

There are those commands that are addressed to individual instruments within the Chroma. The higher command codes are assigned to these commands. The three least significant bits of these command codes hold the instrument number.

What follows is a complete description of each command, along with the numerical code (in hexadecimal) for each command byte.

No Operation 00

The only significance of this particular No Operation (as opposed to any of the undefined command codes) is that the Chroma sends this code upon power-up or reset.

Identification 01

The Chroma (or any future instrument) will respond with three bytes, an Identification command, a device code (1 for a Chroma, 2 for a Chroma Expander), and a software revision level code (see Appendix A).

Read Program 02 pp

The Chroma will respond by transmitting program number pp. The information is transmitted as a Read Program command and 59 data bytes. (If pp is not between 0 and 50, the data bytes will be undefined.)

Write Program 03 pp dd ... dd

The 59 data bytes dd ... dd are written into program number pp in the Chroma. (If pp is not between 0 and 50, the data will be accepted and ignored.)

Load Packet 04

One packet of information is read from the cassette interface, its error detection codes are checked, and the result will be returned via the interface in the form:

04 nn dd ... dd

nn specifies the number of data bytes in the packet, and the dd bytes are the contents of the packet. The first byte of the packet (the first dd byte) is always the packet ID, which identifies the type of packet. The packet ID for valid data is always non-zero. If an error occurs in the reading of the cassette, a special error packet with an ID of 0 is returned.

This command starts reading from the cassette immediately. This can cause a problem if the cassette was previously idle. See the Tape Space command below.

The types of packets that are currently defined, and the forms the Chroma return them in, include:

Error Packet 04 02 00 nn

The length is 2, the ID is 0, and nn will be 0 if a read error is detected or FF hex if the cassette was not running (or was shut off in mid-operation).

Program Packet 04 3C 01 dd ... dd

The length is 60 (3C hex), the ID is 1, and the 59 bytes of data represent a Chroma program.

Program Number Packet 04 02 02 nn

The length is 2, the ID is 2, and the single byte of data consists of a valid program number (0 to 50). This type of packet appears, with a program number of 1, at the beginning of a tape recorded with SAVE ALL.

Stop Packet 04 01 03

The length is 1, the ID is 3, and there is no data in the packet. This type of packet appears at the end of a tape recorded with SAVE ALL.

Save Packet 05 nn dd ... dd

The packet dd ... dd containing nn bytes is written to the cassette. The first dd byte, which is the packet ID must be non-zero. The Chroma will respond when the operation is complete with 05 00 if the operation completes normally or 05 FF if the cassette isn't running.

Read Parameter 06 pp nn

Parameter number nn in program number pp is read and returned in the form 06 vv, where vv is the parameter value. If pp is not between 0 and 50, or if nn is not between 0 and 100, the vv value will be undefined.

Write Parameter 07 pp nn vv

Parameter number nn in program number pp is set to value vv. If pp is not between 0 and 50, or if nn is not between 0 and 100, the vv value will be ignored. If the vv value is not within the range defined for the parameter, the result is undefined, except that the parameter will never be set to an illegal value.

Panel Switch Off 08

The "panel switch" referred to is the software switch which "connects" the Chroma panel to the interface. When the Chroma receives this, it will echo it and disconnect the panel from the interface.

Panel Switch On 09

When the Chroma receives this, it will echo it and connect the panel to the interface. While this mode is in effect, the Chroma will transmit certain commands when the following events occur:

Whenever a program is selected, a Define command will be transmitted for instrument 0 and either a Define or an Undefine command will be transmitted for instrument 1, depending upon the existence of a link.

Whenever a parameter is changed, a Set Parameter command will be transmitted for instrument 0.

Whenever the link balance is varied, Volume commands will be transmitted for instruments 0 and 1.

Performance Switch Off 0A

The "performance switch" referred to is the software switch that "connects" the various performance controls to the interface. When the Chroma receives this command, it echoes it and disconnects the performance controls from the interface.

Performance Switch On 0B

When the Chroma receives this, it will echo it and connect the performance controls to the interface. While this mode is in effect, the Chroma will transmit certain commands when the following events occur:

Whenever a key is pressed on the keyboard, an Attack command will be transmitted for instrument 0, 1 or both, depending upon the link mode and keyboard split.

Whenever a key is released on the keyboard, a Release command will be transmitted for instrument 0, 1 or both, depending upon the link mode and whether or not an attack had already been sent for the note.

Whenever a lever, pedal or footswitch moves, the appropriate command is transmitted for instrument 0, and for instrument 1 if a link is in effect.

Peek 0C aa aa nn

The Chroma responds by transmitting nn bytes from its internal memory starting at location aaaa. The response is in the form:

0C nn dd ... dd

where the dd bytes are data bytes from ascending addresses.

Peek Two Bytes 0D aa aa

The Chroma responds by transmitting two bytes from its internal memory at locations aaaa and aaaa+1. The response is in the form:

0D dd dd

This command is guaranteed to extract the two bytes concurrently, with no chance that the memory locations could be altered between the transmittal of each byte.

Poke 0E aa aa nn dd ... dd

The nn data bytes dd ... dd are poked into the computer's address space starting at location aaaa. If an Unlock command has not been issued since the Chroma was powered up (or reset), the entire command will be read in and ignored.

Poke Two Bytes 0F aa aa dd dd

The two data bytes dd dd are poked into the computer's address space in locations aaaa and aaaa+1, respectively. If an Unlock command has not been issued since the Chroma was powered up (or reset), the entire command will be read in and ignored. This command is guaranteed to poke the two bytes concurrently, without danger of the computer utilizing half of the old contents and half of the new contents.

Tap Panel 10

The panel tapper is triggered, unless it has been disabled.

Unlock 11 00 FF

This sequence must be transmitted in order to enable the Poke and Poke Two Bytes commands.

Tape Space 12

The cassette motor will be run for two seconds. Upon completion, the Chroma will respond with 12 00 if the cassette was running, or 12 FF if it was shut off.

The purpose of this command is to allow startup time before other cassette operations. If a sequence of Save Packet commands are to be issued, they should be preceded by two Tape Space commands. In addition, if the packets are to be individually readable, they should be separated by two Tape Space commands. A single Tape Space command should be issued prior to a sequence of Load Packet commands.

Restore 13

The Chroma is restored to the state reflected by its panel settings. All instruments are undefined except instrument 0 and possibly 1, which are set up according to the currently selected program. The panel switch, performance switch and pressure switch are turned off, and a Panel Switch Off, Performance Switch Off and Pressure Switch Off command are echoed, in that order. (See Appendix A for information on early revisions.)

Pressure Switch Off 14

The "pressure switch" referred to is the software switch that "connects" the keyboard pressure sensors to the interface. When the Chroma receives this command, it echoes it and disconnects the pressure sensors from the interface. This command is not implemented in early Chromas. See Appendix A.

Pressure Switch On 15

When the Chroma receives this, it will echo it and connect the pressure sensors to the interface. While this mode is in effect, the Chroma will send Pressure commands for instrument 0 and/or 1 whenever the pressure on a key is varied. See Appendix A.

Pressure 68+i kk pp

Instrument i is told to set the key pressure input for note kk to value pp. The pressure is an unsigned number from 0 to 63.

This command will be transmitted for instrument 0 and/or 1 by the Chroma if the pressure switch is on and the measured pressure on a depressed key changes. Pressure commands only occur between the corresponding Attack and Release commands for the same note.

This command is not implemented in early Chromas, and must not be sent to them. Current Chromas will respond correctly to this command even if the Pressure Sensor option is not installed. See Appendix A.

Information 70+i

The Chroma responds by echoing the command and sending four information bytes. Currently, only the first byte is utilized, and contains the number of channel boards assigned to instrument i. The other three bytes are zero.

Volume 78+i vv

The Chroma sets the volume of instrument i to vv. The value vv is a linear control from 0 to 255, and is nominally 255. Thus, to reduce the volume of an instrument 6db, the correct vv value would be 128.

This command will be transmitted (for instruments 0 and 1) by the Chroma if the panel switch is on and the Link Balance parameter is varied.

Lever 1 80+i vv
Lever 2 88+i vv

The Chroma sets the value of the appropriate lever input on instrument i to vv, where vv is a signed 2's complement byte in the range -128 to +127. This range corresponds to the mechanical range from "pull" to "push", with 0 corresponding to "at rest".

These commands will be transmitted (for instruments 0 and possibly 1) by the Chroma if the performance switch is on and the performer moves a lever.

Pedal 1 90+i vv
Pedal 2 98+i vv

The Chroma sets the value of the appropriate pedal input on instrument i to vv, where vv is a number in the range 0 to 255. This range corresponds to the mechanical range from "heel" to "toe".

These commands will be transmitted (for instruments 0 and possibly 1) by the Chroma if the performance switch is on and the performer moves a pedal.

Footswitch 1 Down	A0+i
Footswitch 1 Up	A8+i
Footswitch 2 Down	B0+i
Footswitch 2 Up	B8+i

These commands activate or deactivate the footswitch functions on instrument i. Footswitch 1 is the right, or sustain, footswitch. Footswitch 2 is the left, or effects, footswitch.

These commands will be transmitted (for instruments 0 and possibly 1) by the Chroma if the performance switch is on and the performer presses or releases either footswitch.

Define C0+i pp aa bb cc dd ee ff

Instrument i is defined according to program pp (which must be in the range 0 to 50). The remaining bytes specify initial values for the performance inputs:

aa: lever 1	bb: lever 2
cc: pedal 1	dd: pedal 2
ee: volume	ff: footswitches

The footswitch byte uses the most significant bit to represent footswitch 1 and the next most significant bit to represent footswitch 2. A 0 means up, 1 means down. If pp is not between 0 and 50, the Chroma will not define the instrument according to garbage data, but nothing more is promised.

This command causes channel boards to be reallocated as fairly as possible among defined instruments. If this command requires that one or more channel boards be robbed from another instrument, the computer will be kind enough to try and pick boards that aren't currently sounding.

This command will be transmitted (for instrument 0 and possibly 1) by the Chroma if the panel switch is on and the performer selects a program or a link. Although instrument 0 is internally defined by program 0, the Define command that is transmitted whenever a program is selected includes the current program number as shown in the 2-digit display.

Undefine C8+i

Instrument i is removed from operation, and any channel boards assigned to it are redistributed among any other instruments.

This command will be transmitted for instrument 1 by the Chroma if the panel switch is on and an unlinked program is selected or a link is cleared.

Attack D0+i kk vv pp

Instrument i is told to attack note kk with a velocity vv and an initial pressure pp. The key number is a signed, 2's complement byte that must be in the range -64 to +63. The Chroma's keyboard has a range from -32 to +32, with 0 being middle C. The velocity must be a number from 0 (softest strike) to 31 (hardest strike), and the pressure must be a number from 0 (no pressure) to 63 (full pressure).

The result of this command depends upon the keyboard algorithm parameter in the program that the instrument is defined by.

This command will be transmitted for instrument 0 and/or 1 by the Chroma if the performance switch is on and the performer presses a key.

Early Chromas transmit a pressure byte that is always zero, and ignore the received pressure byte. Current Chromas respond to pressure information received even if the Pressure Sensor option is not installed.

Release D8+i kk vv

Instrument i is told to release note kk with a velocity vv. The result of this command depends upon the keyboard algorithm parameter in the program that the instrument is defined by.

This command will be transmitted for instrument 0 and/or 1 by the Chroma if the performance switch is on and the performer releases a key.

Set Parameter E0+i nn vv

Instrument i temporarily sets parameter nn to value vv. This does not affect the setting stored in non-volatile memory, which means that it won't affect other instruments defined according to the same program and it won't affect this instrument if it is redefined according to the same program. Only those parameters that pertain to the tone generation may be set with this command. These include:

1 through 5: control parameters
6 through 50: A parameters
55 through 100: B parameters

Any other parameter number will cause the command to be ignored. If vv is not within the valid range for the selected parameter, the only guarantee is that the parameter will not be set to an illegal value.

This command will be transmitted for instrument 0 by the Chroma if the panel switch is on and the performer varies one of the parameters.

Status E8+i

This command causes the Chroma to respond with:

E8+i pp aa bb cc dd ee ff

where the seven parameters represent the same quantities as the parameters of the Define command. If the instrument is undefined, the program number returned will be FF and the remaining bytes will be undefined. If the program number is 0, the program number in the display will be used instead.

Squelch F0+i kk

Any channels in instrument i that are assigned to key k are squelched by setting their envelopes to 0. This doesn't affect the channel assignment tables. Even latched channels may be squelched. If kk is -128 (80 hex) all channels will be squelched.

SOFTWARE REQUIREMENTS

Levels of Complexity -- The complexity of the software needed to communicate with the Chroma is dependent upon the kind of communication desired. The most important factor is whether or not the software can wait for input from the Chroma. A simple system can be designed in which all communication is essentially half-duplex, meaning that when the computer is expecting information from the Chroma it is doing nothing else. This precludes recording and playing concurrently. In fact, it precludes doing much more with information arriving from the Chroma than storing it for later processing.

In order to allow more processing to occur in response to information arriving from the Chroma (as opposed to processing that is totally independent of what the Chroma might be sending), it is advisable to make the input system interrupt driven. This would allow information to be taken into the computer as soon as the Chroma sends it, where it would be queued until it could be processed.

If it is desired to record and process information from the Chroma while doing a large amount of unrelated processing (such as communicating with other instruments or a display terminal), some form of multi-tasking is necessary. A generalized multi-tasking operating system would be nice, but hardly necessary. The Chroma firmware is itself structured as two concurrent tasks, as the Chroma has plenty of stuff to do besides wait around for commands to arrive on the interface.

If it is important that outputting information to the Chroma be fast, a queue can be provided for outgoing information, and an interrupt can be used to move bytes from the queue onto the port. This is also done in the Chroma.

A Simple System -- The simplest form of communication doesn't require any fancy software support. It is only necessary to wait for the output port to be empty before outputting each byte, and wait for the input port to be full before inputting each byte. BASIC peeks and pokes are sufficient for handling programming information, although most interpreted versions of BASIC aren't really fast enough to implement a decent sequencer. Note also that it would be advisable to include some method of getting out of the loop that waits for input from the Chroma (such as pressing a key on the computer terminal) to prevent communications problems from hanging the computer. A loop with a timeout might be appropriate when the computer requests specific information from the Chroma. Unless the Chroma is doing an autotune or cassette operation, it should respond to any command within a couple milliseconds.

A Sytem With Interrupt Driven Input -- This kind of system is what most people will probably be interested in playing with. The Hardware Requirements section of this manual shows an interface circuit that includes provisions for interrupting the processor when either the input port is full or the output port is empty. The output interrupt is less important, so the gates needed for this can be left out if not desired. The purpose behind making the input interrupt driven is that it keeps the real-time constraints of the Chroma from extending into the bulk of the computer software. This is because it allows

rapid bursts of information from the Chroma to be handled as long as the computer can keep up with the average rate of information flow. The software necessary to do interrupt driven input consists of three procedures. The initialization procedure sets up the queue pointers and enables the interrupt. The interrupt handler pulls bytes off the port and stuffs them in the queue. The input procedure pulls bytes out of the queue for processing. These algorithms are presented below. Note that the interrupt handler must be written so that it returns with the interrupt masked in the event that the queue is full, and the input routine must, upon removing a byte from the queue, reenable the interrupt.

INTERRUPT DRIVEN INPUT ALGORITHM

PROCEDURE TO INITIALIZE INTERFACE -- called upon start-up

```
set head and tail queue pointers to zero
enable input interrupt
```

INTERRUPT HANDLER

```
input byte into tail of queue
advance queue tail pointer
if input queue full
     disable interrupt
```

INPUT PROCEDURE

```
wait for queue to be not empty
remove byte from head of queue
disable interrupt
advance queue head pointer
enable interrupt
return byte
```

A fourth procedure might be provided to check to see if anything is in the queue without actually waiting in a loop.

A Fully Interrupt Driven Dual Task System -- This is really a description of the way the Chroma handles its end of the interface. The algorithms described below show how the interface software might be written to allow inputting information to be handled as a separate, parallel process, without the use of a multi-tasking operating system. The purpose of multi-tasking is to allow a computer to take turns doing more than one thing, giving the appearance that it is doing them simultaneously.

When a computer has more than one task to perform, some mechanism must be provided for deciding which task should be handled at any given instant. In the Chroma, there are two tasks, one controlling the synthesizer and one responding to commands from the interface. Deciding which task should be performed is simple. If a byte is available from the interface, it is processed. If no byte is available, one complete cycle of the synthesizer firmware is performed (lasting about 1.25msec). This "multi-tasking" is characterized by the following parameters:

The synthesizer task is only suspendable at one particular point in its loop, when it isn't in the middle of anything.

The external input task is only suspendable when it must wait for input.

The external input task has the higher priority.

The first rule makes the interface between the two tasks clean. There is no danger of the external interface task manipulating something that the synthesizer task was in the middle of manipulating when it was suspended.

In order to implement two parallel tasks, it is necessary to save all the information representing the state of one task while running the other task. If a task is suspendable at only one point, and under the same conditions every time, no state information is required. If the task is to be suspendable in more than one place under different conditions, this information must be saved. Also, tasks that don't always leave their stack pointers in the same place each time they are suspended cannot share the same stack. In the Chroma, the external input task is carefully written so as not to use the stack when calling for a byte of input for anything other than the return address. This allows it to function without a separate stack.

The algorithms presented below treat the synthesizer task as the "background" task and the external input task as a "priority" task. The synthesizer task checks the input queue every now and then and, if a byte is found, causes the external input task to be resumed. The external input task consists of a command interpreter that calls the input routine from many different points, meaning that the task state image must include a return address. The state image is created during initialization with a PC value that points into the beginning of the command interpreter. Thus, the first byte that arrives will cause the input procedure to resume, and "return" to the command interpreter, which will handle the received byte and ultimately call the input procedure again. The nice thing about this type of system is that it allows you to write the external input handling software as though it were the only thing going on. The details of switching between the two tasks are hidden inside the input procedure.

INTERRUPT DRIVEN DUAL TASK ALGORITHM

```
PROCEDURE TO INITIALIZE INTERFACE
     -- called upon start-up

     set input and output head and tail pointers to zero
     set non-responding flag -- this gets reset when first byte arrives
     create external input process state image
          -- PC (program counter) cell must point into command
          -- interpreter, as if command interpreter had called
          -- input procedure for first command code byte
     enable input interrupt -- output interrupts remain disabled

INPUT AND OUTPUT INTERRUPT HANDLER
     -- handles both interrupts arriving on one line, with a round-robin
     -- priority scheme for concurrent input and output interrupts

     go to ENTRY -- start by checking for output interrupt

   INPUT:
     clear non-responding flag
     input byte and put into tail of input queue
     advance input queue tail pointer
     if input queue full
          disable input interrupt

   ENTRY:
     if output interrupt pending
          go to OUTPUT
     else if input interrupt pending
          go to INPUT
     else return

   OUTPUT:
     output byte from head of output queue
     advance output queue head pointer
     if output queue empty
          disable output interrupt

     if input interrupt pending
          go to INPUT
     else if output interrupt pending
          go to OUTPUT
     else return

PROCEDURE TO DISPATCH EXTERNAL INPUT PROCESS
     -- called every 1msec or so by the main process

     if input queue not empty
          remove byte from head of input queue
          if input interrupt disabled -- meaning queue was full
               enable input interrupt
          save machine state
          restore external input process machine state
          return byte -- to external input process
```

```
PROCEDURE TO INPUT ONE BYTE
     -- called by external input process

     if input queue not empty within 100usec
          remove byte from head of input queue
          advance input queue head pointer
          if input interrupt disabled -- meaning queue was full
               enable input interrupt
          return byte
     if input queue still empty after 100usec
          save machine state
          restore main process machine state
          return -- to main process

PROCEDURE TO OUTPUT ONE BYTE
     -- called by either main or external input process

     if non-responding flag set
          return
     if output interrupt disabled -- meaning queue not in use
          if output port empty within 100usec
               output byte
               return
     -- otherwise, port was full or queue was already in use
     if room in output queue within 1.5msec
          put byte in tail of output queue
          advance output queue tail pointer
          enable output interrupt
     if output queue still full after 1.5msec
          set non-responding flag
          disable output interrupts
          set input and output head and tail pointers to zero
          reinitialize external input process state
```

With two hardware-prioritized interrupt lines, the interrupt handler could be split in two. Note also the 100usec timed loops that actually speed up data transfers by increasing the liklihood that a multi-byte data transfer can be handled without re-interrupting for each byte. The non-responding flag is a mechanism used in the Chroma to handle the case of a crashed computer at the other end of the interface. It is cleared by incoming bytes and set if the output doesn't respond within a reasonable time.

Time Measurement -- Computer software to aid in programming the Chroma does not require any timing circuitry. However, if you intend to record and play back music with the Chroma, time measurement becomes very important. The Chroma is pretty good about playing music that arrives over the interface without any noticeable time lag. However, if you intend to use an interrupt driven input system, you must make sure that the delay between a byte's acceptance by the interrupt handler and its ultimate processing doesn't cause timing errors in the music. The easiest way to assure this is to let the interrupt handler record the time each byte arrives. Each byte in the input queue will therefore be accompanied by time information. The timing resolution should be better than 10msec, yet anything faster than 1msec is probably

extraneous. A sixteen bit timer will provide enough time information, as long as commands don't arrive from the Chroma further apart than half the cycle time of the timer (more than thirty seconds at 1msec resolution). Thus, the input queue should be able to hold two time bytes for every command byte.

Once the information is pulled from the queue for processing, the time bytes associated with command operands can be eliminated, leaving only the time bytes associated with each command code byte. The absolute time measurements might also be converted to relative time between events, if that is more appropriate to the processing that is to be performed.

Utilizing The Command Language Of The Chroma -- The structure of a music recording and playback system is further impacted by the fact that the communication in each direction upon the interface is independent, yet the information flowing in each direction is not. To clarify, consider the case of the Performance Switch Off command. This command is normally sent to the Chroma as a signal that you are finished recording and no more information is to be accepted. But it is entirely possible, given the amount of buffering that the information must suffer, that further performance information will be transmitted in the milliseconds after this command has been issued. Even though the Chroma does in fact stop transmitting when it sees the Performance Switch Off command, there is no guarantee as to how long this will take. The difficulty is handled by the fact that all such mode change commands are echoed by the Chroma. Thus, the stream of data coming back from the Chroma will include "flags" that frame the information so that the computer knows when the Chroma is done transmitting. The correct way to start and stop recording from the Chroma is to keep a status flag that is set by receipt of a Performance Switch On command and cleared by receipt of a Performance Switch Off command. When recording is to commence or terminate, the computer should send the appropriate command, but the state of the status flag, controlled by the echoed commands, should start and stop the actual recording process.

The Restore command is actually the command most likely to be used to terminate recording. This command is provided as a convenience, making it unnecessary to explicitly restore the instrument definitions that were in effect when the recording started. Note, though, that the first few Chromas built do not echo the Performance, Panel and Pressure Switch Off commands when the Restore command is received, but current Chromas do. Refer to Appendix A.

When recording, with the panel, performance and/or pressure switches on, the Chroma will transmit "commands" with instrument codes 0 and 1. In order to allow playing and recording at the same time, instruments that are used for playback should be assigned higher numbers. If the interface is sending commands to instruments 0 and 1, there is nothing to prevent the performance controls and panel controls to send commands to these instruments at the same time. This shouldn't cause a problem, but it won't sound very good either. Note that the commands that are sent by the Chroma during recording are all in exactly the form (except for instrument number) that they should be transmitted back to the Chroma during playback. No other information will be sent by the Chroma unless it is explicitly requested.

HARDWARE REQUIREMENTS

This section describes the circuits needed to interface a computer to a Chroma. It is assumed that the reader is reasonably familiar with the bus structure of his own computer, and what is shown here will have to be modified accordingly. In particular, details of address decoding, bus acknowledge and interrupt control are not shown.

Minimal Interface -- At the very least, one must have an 8-bit latch driving the output lines that can be written into, an 8-bit tri-state driver sensing the input lines that can be read from, and a set/reset flip-flop associated with the output lines that maintains the status of the output port. (The other end of the interface maintains the status of the input port.) In addition, there must be a way of sensing the status of the two ports. This is shown in Fig. 5-1. It is assumed here that the bus cycle strobe, read and write signals and address are all decoded to provide individual active-low strobes to all circuits that require them, that acknowledgement is taken care of elsewhere (no wait states should be needed), and that the data bus is an 8-bit positive logic bus. Note that the XOACK (External Output Acknowledge) and XIFULL (External Input Full) lines are resistor terminated. This is because the Chroma drives these lines with open collectors.

Modified Minimal Interface -- The bus drivers shown here are LS TTL devices, as their low power consumption is convenient in a microprocessor system. For slightly better noise immunity, the Shottky outputs can be buffered by regular TTL parts which pull down closer to ground, although this is not done in the Chroma. The Chroma uses RC networks on all its outputs to avoid RFI problems with the FCC, but you can do what you like at your end of the interface. You might also put a couple of transistors on the Output Full and Input Ack lines to make sure that these outputs are not activated except when the computer is turned on. Otherwise, powering down the computer would continuously interrupt the Chroma. These modifications are shown in Fig. 5-2.

Interrupt Driven System -- The status of each port can be used to generate an interrupt. The input port should be capable of generating an interrupt when it is full, and the output port should be capable of generating an interrupt when it is empty. In addition, each interrupt should be independently maskable. If a multi-level interrupt structure already exists in your system, you will only need to connect the status lines (inverting one of them) to two interrupt lines. Otherwise, a 2-bit output port must be provided to hold the interrupt mask bits, and some gates must be provided to combine everything into one interrupt line. This is shown in Fig. 5-3. The circuit shown maintains the interrupt until the condition causing it is removed. Some systems will require an open collector interrupt signal.

A Simple Free-running Timer -- The circuit shown in Fig. 5-4 consists of a prescaler to convert a high-frequency clock signal into a more usable frequency of perhaps a few hundred hertz. This drives the counters inside a pair of 74LS590 chips[1]. Each of these chips has an 8-bit synchronous counter, an 8-bit latch, and an 8-bit tri-state driver. Note that the register clock inputs (RCK) are tied together to a separate strobe line, which must be strobed before reading the counter. Although a programmable prescaler might be nice, its function can be handled in software if you are reasonably clever.

Connecting to the Chroma -- The physical interconnection to the Chroma is through the 25-pin D-type connector on its rear panel. Figure 5-6 shows the pin-out of the connector. Note, however that the names of the connections shown in this diagram are from the Chroma's point of view. That is, the lines that are associated with the "output" port deal with information flowing out of the Chroma, and the lines that are associated with the "input" port deal with information flowing into the Chroma.

Obviously, the Chroma's output lines must connect to the computer's input lines and vice versa. There are two ways to do this. If you wish to use an inexpensive off-the-shelf ribbon cable, you must assign pin numbers at the computer's end of the interface according to the scheme in figure 5-7. This will connect the XO lines at one end to the XI lines at the other. Alternatively, you may assign the pin numbers at the computer's end the same as the Chroma, as shown in figure 5-6, and use a "crossover" cable. Rhodes sells a sturdy <u>shielded</u> crossover cable, with molded plugs at each end, that was designed to interface a Chroma to an Expander. The use of this cable (and the pin-out of figure 5-6) is recommended, as it simplifies larger systems: anything that can transmit can be connected to anything that can receive without concern about whether the pin-out is correct.

[1] See the 1981 Supplement to The TTL Data Book, Texas Instruments.

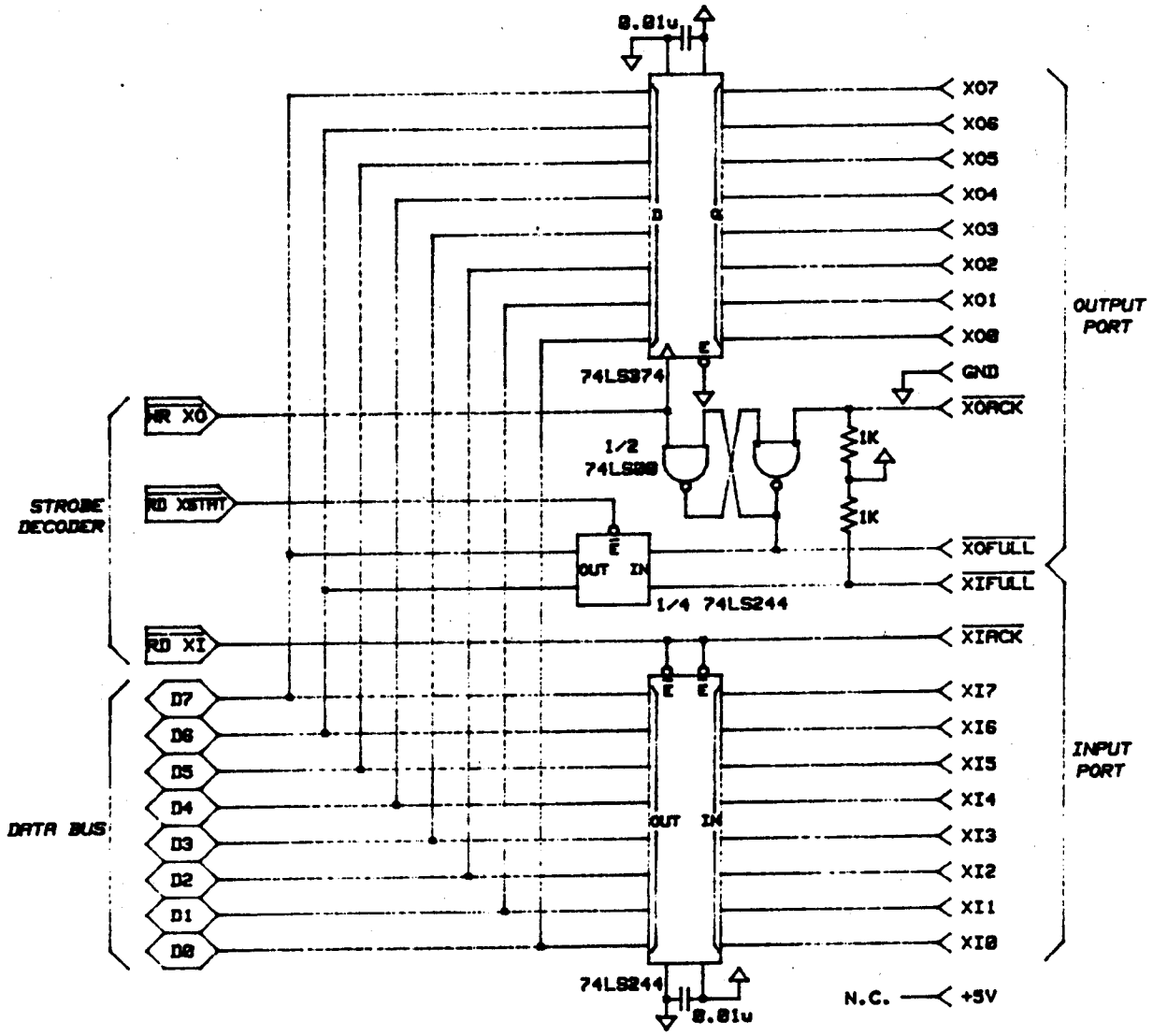


Fig. 5-1 Minimal Interface

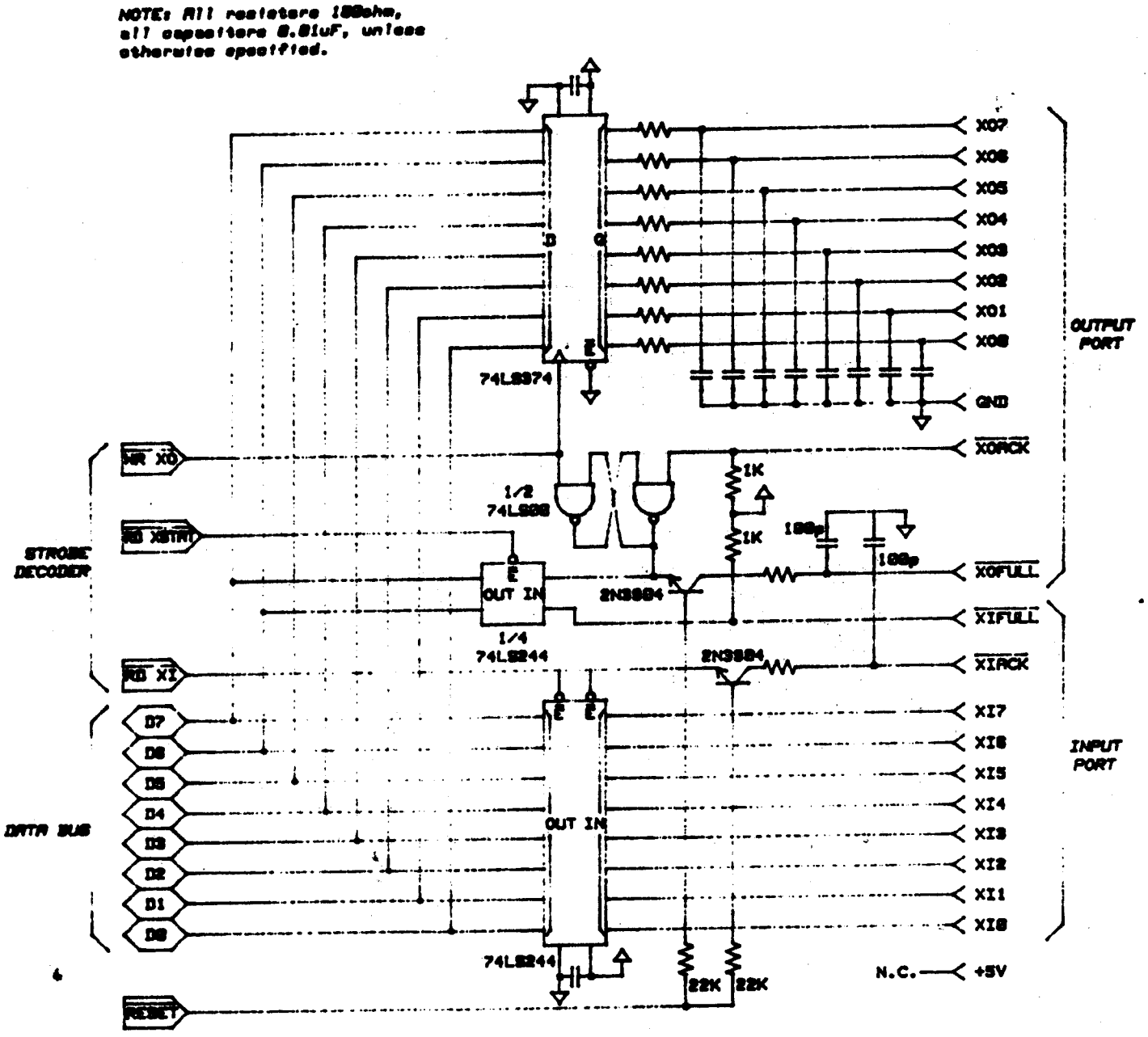


Fig. 5-2 Modified Minimal Interface

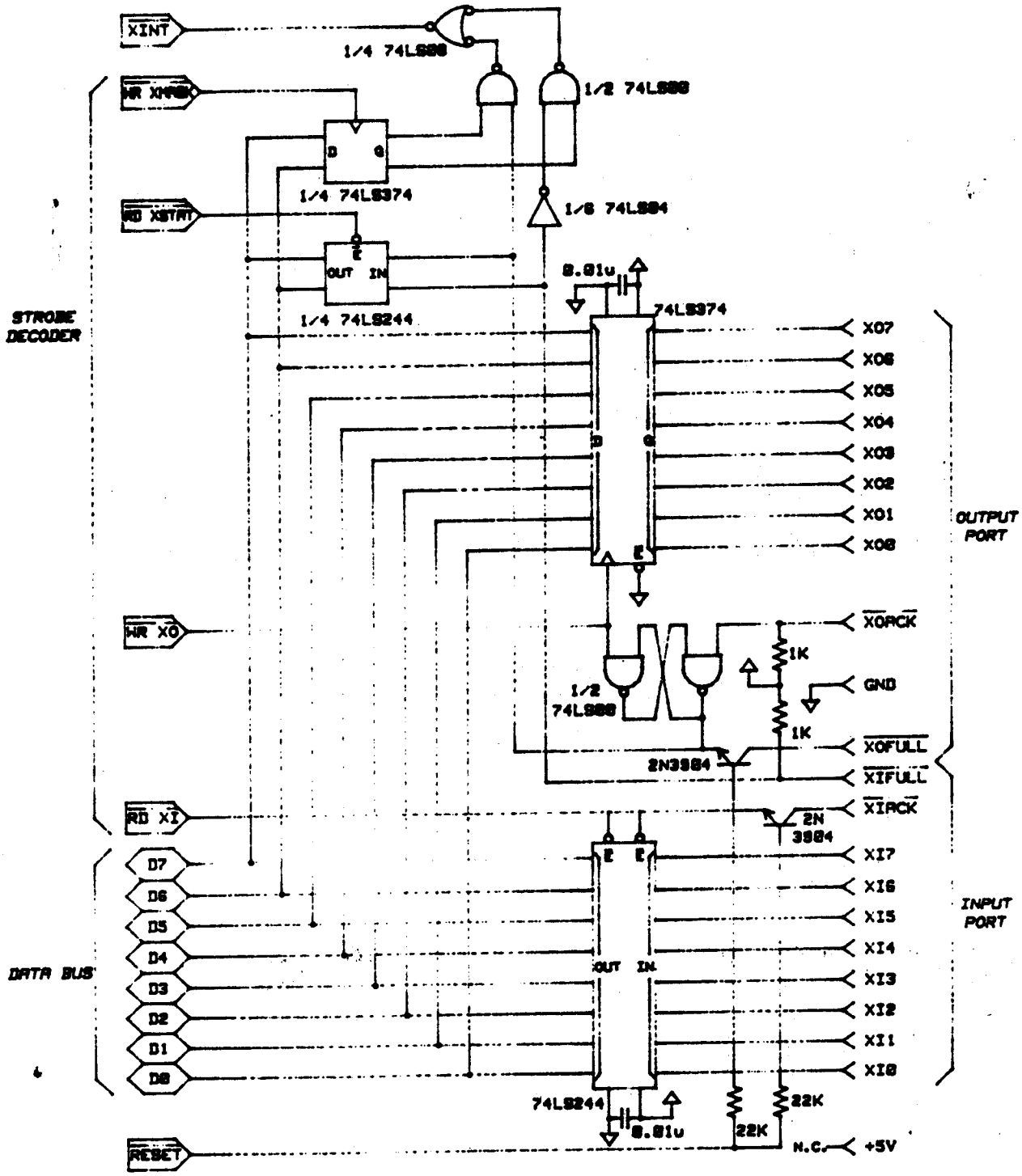


Fig. 5-3 Interrupt Driven Interface

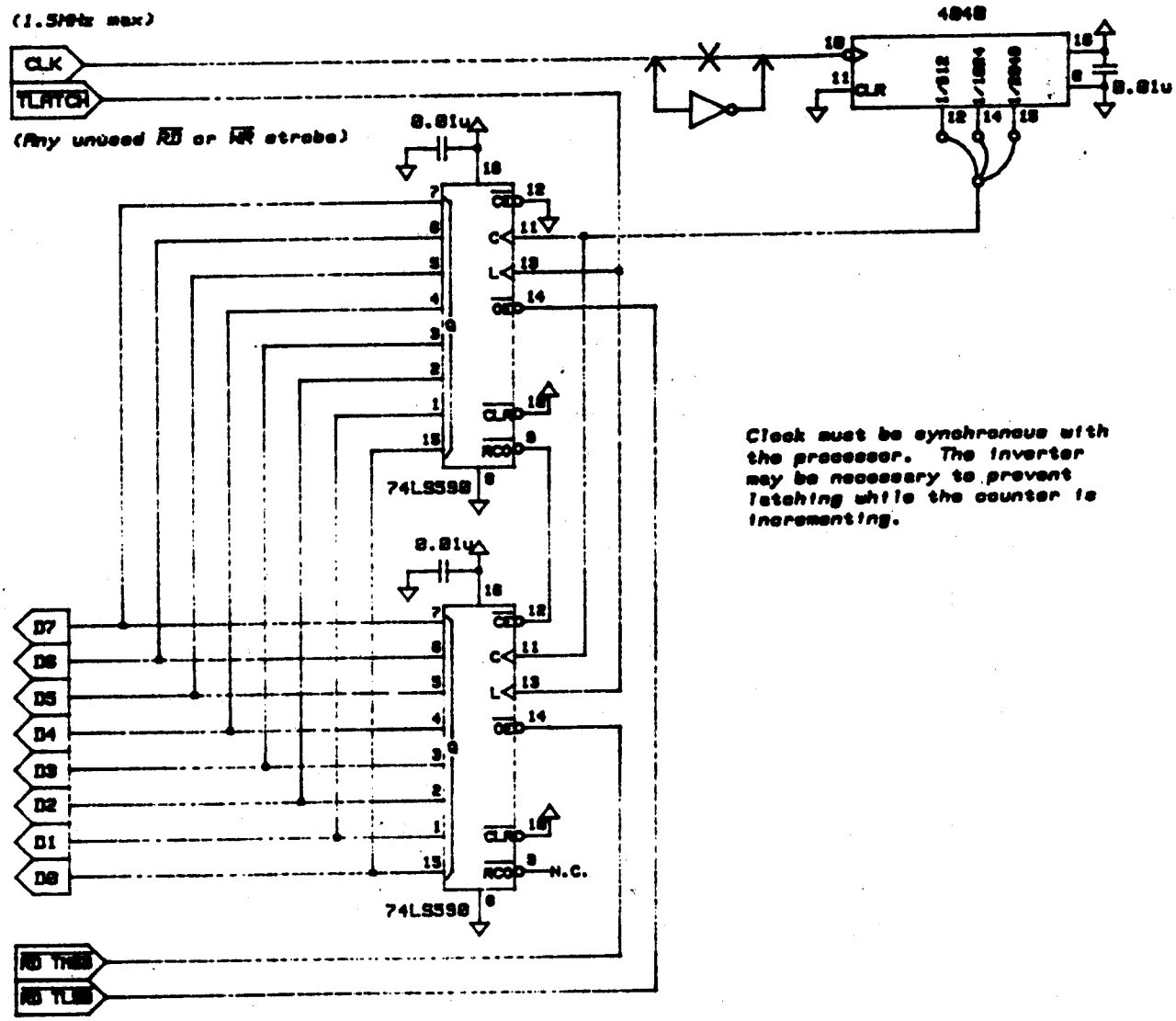
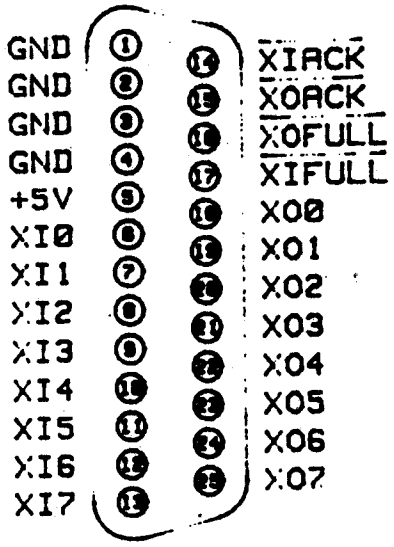
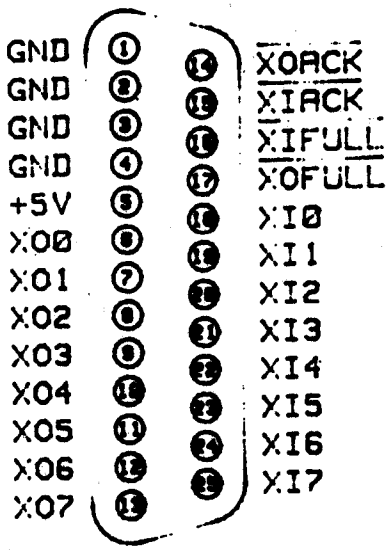


Fig. 5-4 Free-running Timer



Note: This pinout should be used at the computer's end of the interface, along with the "crossover" type cable.

Fig. 5-6 Chroma D-connector Pin-out



Note: This pinout may be used at the computer's end if an off-the-shelf ribbon cable is to be used. See text.

Fig. 5-7 Complementary Pin-out

APPENDIX A
REVISION INFORMATION

This section outlines the differences in behavior between Chromas and Expanders of different revision levels. The Interface Revision Number is returned in response to the Identification command. Please note that this number is not the same as the Software Revision Number as imprinted on the EPROMs inside the unit.

The revision levels are described in reverse order, starting with the current revision. Each revision description outlines the differences between that revision and the subsequent revision above it.

REV 3 (software REV 13) -- This is the current revision, as described in this manual.

REV 2 (software REV 12) -- This revision did not include the pressure sensor commands. This results in the following restrictions:

The Pressure Switch Off and Pressure Switch On commands are treated as No Operations. They are not echoed.

The Restore Command does not echo a Pressure Switch Off command.

The pressure byte in all Attack commands sent by the Chroma is 0. The pressure byte in all Attack commands received by the Chroma is ignored (although it must be present).

A bug was found in this revision:

If a link is in effect and a lever or pedal is moved, the Chroma will not transmit an instrument 0 and an instrument 1 command. Instead, the Chroma will send two identical instrument 0 commands. This only applies to the Lever 0, Lever 1, Pedal 0 and Pedal 1 commands.

REV 1 (Software REV 10) -- A number of bugs were found in this revision:

If a Footswitch command is sent to any instrument that has never been defined since power-up, it will crash the Chroma.

The Restore command does not do anything to instrument 1, regardless of the link.

Bytes coming from the Chroma occasionally get rearranged and are transmitted out of sequence. This only occurrs if the computer makes the Chroma wait more than 100usec or so, and the Chroma starts to use its output queue. If you experience problems at high data rates, suspect this.

Although the Restore command turns off the panel and performance switch, it does not echo the Panel and Performance Switch Off commands.

Upgrading A Chroma -- All it takes to bring a Chroma up to the current revision is to unplug the EPROMs and plug in new ones. This can be done by any authorized Rhodes Chroma service center, and is free if the instrument is under warranty. Upgrading is strongly recommended, as old software is only old because there was something wrong with it.

To request an upgrade from a service center, always refer to the Software Revision Number which is printed on the EPROMs, not the Interface Revision Number. Most service centers are not aware of Interface Revision Numbers.

The current revision includes provision for the Pressure Sensor option. This does not mean that the Pressure Sensor must be installed. The Chroma will respond to Pressure commands whether or not the option is installed. It just won't generate correct Pressure commands.

APPENDIX B
PARAMETER INFORMATION

A Chroma program consists of 101 parameters divided into four categories:

Panel Parameters -- These are parameters 0 and 51 through 55, and include the parameters that represent the states of various panel controls. These are described in detail below.

Control Parameters -- These are parameters 1 through 5, and are accessible from the panel using switches 1 through 5.

A Channel Parameters -- These are parameters 6 through 50, and are accessible from the panel using switches 6 through 50 in Edit A mode.

B Channel Parameters -- These are parameters 56 through 100, and are accessible from the panel using switches 6 through 50 in Edit B mode.

The Control, A Channel and B Channel Parameters are fully described in the Chroma Programming Manual. Their values appear in signed two's complement form over the interface.

There are six panel parameters. The panel parameters stored in programs 1 through 50 have no effect, even if an instrument is defined by the program. The only panel parameters that have any effect are those in program 0, and they represent the current states of the "programmable" panel controls:

0. Link Balance -- This is what appears in the Data Readout when a link is set up, although the values are different. The value shown in the display ranges from -14 to +14 in steps of 2, while the value accessible through the interface ranges from -7 to +7 in steps of 1.

51. Link -- This parameter includes the Link Mode and the Link Program Number in a single byte value. The six lsbs represent the Link Program Number, which must be between 1 and 50. The two msbs represent the Link Mode as follows:

00 = No Link, 01 = Link Upper, 10 = Link Lower, 11 = Link Unison

52. Edit -- This parameter includes the Edit Mode and the currently selected parameter number. The six lsbs represent the parameter number, which must be 0 if the Link Balance parameter is selected, or a number from 1 to 50 if a Control or Channel parameter is selected. The two msbs represent the Edit Mode as follows:

01 = Edit B, 10 = Edit A, 11 = Edit A&B

53. Keyboard Split -- This is a number from -32 to +31.

54. 55. Main. Link Transposes -- These represent the settings of the transpose switches as follows:

00 = Normal, 01 = Up 1 Oct, 10 = Down 1 Oct

PROGRAM TABLE LAYOUT

The Read Program and Write Program commands deal with complete sets of parameters as they appear in the Chroma's non-volatile memory, packed into 59 bytes. The following table shows the location of each parameter:

No(s)	Group	Name	Byte(s)	7 6 5 4 3 2 1 0
0	Panel	Link Balance	31	[- - - - N N N N]
1	Control	Patch	1	[- - - - N N N N]
2	Control	Fsw Mode	5	[- - - - - N N N]
3	Control	Kybd Alg	31	[N N N N - - - -]
4	Control	Detune	2	[N N N N N - - -]
5	Control	Output Select	2	[- - - - - N N -]
6,56	Glide	Rate	28,58	[N N N N N - - -]
7,57	Glide	Shape	14,44	[- N - - - - - -]
8,58	Sweep	Mode	4,34	[- - - - - - N N]
9,59	Sweep	Rate	4,34	[N N N N N N - -]
10,60	Sweep	Rate Mod	3,33	[- - - - N N N N]
11,61	Sweep	Wave Shape	6,36	[N N N N - - - -]
12,62	Sweep	Ampl Mod	6,36	[- - - - N N N N]
13,63	Env 1	Ampl Touch	9,39	[- - - - - N N N]
14,64	Env 1	Attack	7,37	[N N N N N - - -]
15,65	Env 1	Attack Mod	7,37	[- - - - - N N N]
16,66	Env 1	Decay	8,38	[N N N N N - - -]
17,67	Env 1	Decay Mod	8,38	[- - - - - N N N]
18,68	Env 1	Release	9,39	[N N N N N - - -]
19,69	Env 2	Delay	10,40	[N N N N N - - -]
20,70	Env 2	Ampl Touch	13,43	[- - - - - N N N]
21,71	Env 2	Attack	11,41	[N N N N N - - -]
22,72	Env 2	Attack Mod	11,41	[- - - - - N N N]
23,73	Env 2	Decay	12,42	[N N N N N - - -]
24,74	Env 2	Decay Mod	12,42	[- - - - - N N N]
25,75	Env 2	Release	13,43	[N N N N N - - -]
26,76	Pitch	Tune	14,44	[- - N N N N N N]
27,77	Pitch	Mod 1 Select	18,48	[N N N N - - - -]
28,78	Pitch	Mod 1 Depth	15,45	[- N N N N N N N]
29,79	Pitch	Mod 2 Select	18,48	[- - - - N N N N]
30,80	Pitch	Mod 2 Depth	16,46	[- N N N N N N N]
31,81	Pitch	Mod 3 Select	19,49	[N N N N - - - -]
32,82	Pitch	Mod 3 Depth	17,47	[- N N N N N N N]
33,83	Width	Wave Shape	20,50	[- - - - - - N N]
34,84	Width	Width	20,50	[N N N N N N - -]
35,85	Width	Mod Select	19,49	[- - - - N N N N]
36,86	Width	Mod Depth	21,51	[- N N N N N N N]

No(s)	Group	Name	Byte(s)	7	6	5	4	3	2	1	0
37,87	Cutoff	LP/HP	15,45	N	-	-	-	-	-	-	-
38,88	Cutoff	Resonance	10,40	-	-	-	-	-	N	N	N
39,89	Cutoff	Tune	22,52	-	-	N	N	N	N	N	N
40,90	Cutoff	Mod 1 Select	26,56	N	N	N	N	-	-	-	-
41,91	Cutoff	Mod 1 Depth	23,53	-	N	N	N	N	N	N	N
42,92	Cutoff	Mod 2 Select	26,56	-	-	-	-	N	N	N	N
43,93	Cutoff	Mod 2 Depth	24,54	-	N	N	N	N	N	N	N
44,94	Cutoff	Mod 3 Select	27,57	N	N	N	N	-	-	-	-
45,95	Cutoff	Mod 3 Depth	25,55	-	N	N	N	N	N	N	N
46,96	Volume	Mod 1 Select	27,57	-	-	-	-	N	N	-	-
47,97	Volume	Mod 1 Depth	3,33	N	N	N	N	-	-	-	-
48,98	Volume	Mod 2 Select	27,57	-	-	-	-	-	-	N	N
49,99	Volume	Mod 2 Depth	5,35	N	N	N	N	-	-	-	-
50,100	Volume	Mod 3 Select	28,58	-	-	-	-	-	N	N	N
51	Panel	Link	0	N	N	N	N	N	N	N	N
52	Panel	Edit	30	N	N	N	N	N	N	N	N
53	Panel	Keyboard Split	32	N	N	N	N	N	N	N	N
54	Panel	Main Transpose	1	N	N	-	-	-	-	-	-
55	Panel	Link Transpose	1	-	-	N	N	-	-	-	-
		Sequence Program Footswitch	29	N	N	N	N	N	N	N	N
		Free bits	2	-	-	-	-	-	-	-	N
			5	-	-	-	-	N	-	-	-
			35	-	-	-	-	N	N	N	N
			14,44	N	-	-	-	-	-	-	-
			16,46	N	-	-	-	-	-	-	-
			17,47	N	-	-	-	-	-	-	-
			21,51	N	-	-	-	-	-	-	-
			22,52	N	N	-	-	-	-	-	-
			23,53	N	-	-	-	-	-	-	-
			24,54	N	-	-	-	-	-	-	-
			25,55	N	-	-	-	-	-	-	-

Note -- The Sequence Program Footswitch byte contains the number of the program that will be selected when the Sequence Program Footswitch is depressed and released. This number is not accessible as a regular parameter. Even though this only requires six bits to represent, the full eight bits are reserved.

Note -- The Keyboard Split parameter is a signed number from -32 to +31. Even though this only requires six bits to represent, the full eight bits are reserved.

Note -- Signed parameter values are represented in two's complement format with the leftmost bit assigned to the parameter being the sign bit. Thus, a mod depth of -10 would appear in seven bits as 1110110.

APPENDIX C
USEFUL LOCATIONS WITHIN THE CHROMA

The memory address space within the Chroma is accessible through the use of the Peek, Peek Two Bytes, Poke and Poke Two Bytes commands. Although the usefulness of these commands depends upon an understanding of the internal structure of the Chroma firmware, there are some locations within the Chroma's address space that can be manipulated without really knowing what goes on inside the instrument. A few of these are documented here as an aid to anyone who wishes to experiment with them.

Revisions often involve rearranging the locations of things inside the computer's address space. The only thing that are likely to move, though, are those items contained in volatile RAM between locations 0100 and 0FFF.

Once you issue the Unlock command and start Poking into the Chroma, you have the capability of crashing the Chroma's computer. A crashed Chroma must be powered down and up again to restart it. It is possible that a crash may corrupt the programs stored in non-volatile RAM (possibly only a byte here and there). Therefore, the 50 programs should be reloaded via cassette (or via the interface) in the event of a crash.

Be sure to use the Peek Two Bytes and Poke Two Bytes commands whenever you are accessing a two byte value. This insures that the interface's access to the locations does not get interleaved with the Chroma's access to the same locations.

0020-0029: Display image -- This contains the image of the ten display digits. Changes made to the first eight locations (the Data Readout) only last until some action occurs on the panel that involves the display. Changes made to the last two locations (the Program Number) last until a program is selected or stored.

002A: LED image byte 1 -- This contains the image of those eight LEDs that are capable of being flashed. Writing to this location will cause the appropriate LEDs to change state, as this location is copied to the LED port byte 1 at regular intervals.

002B: LED image byte 2 -- This contains the image of those eight LEDs that never flash. Writing to this location does not make the LEDs change state, so the LED port byte 2 should be written as well.

002C: LED blink image -- This byte is XORed with the LED image byte 1 at regular intervals to cause LEDs to flash. This should normally be set whenever the image byte is set.

0048, 0049: Master tune -- This two-byte number contains the master tune setting for the instrument. Whenever the tune slider is moved, this will be set to an even number between -256 and +246, which represents a range from -1 to almost +1 semitones. As long as the master tune slider is not moved, this location can be played with through the interface.

005A, 005B: Safe buffer program number, modified flag -- This contains information about the contents of the safe buffer. Whenever a program is selected or stored, the safe buffer is used to store a backup copy of whatever is written over.

006A, 006B: Major loop hook -- Every 20 milliseconds or so, an indirect subroutine call is made through this location. Normally, this contains C100, the address of a Return instruction.

006C, 006D: Minor loop hook -- Every 1.25 milliseconds or so, an indirect subroutine call is made through this location. Normally, this contains C100, the address of a Return instruction.

0140-017A: Safe buffer -- Whenever a program is selected or stored, whatever is written over is first copied here.

017B-01BE: Stack -- The stack grows from high memory to low, and we have never seen it go below 0190, so that leaves twenty locations that are probably free. Assign from 017B up, to be safe.

01BF-02BF: Cassette buffer -- These 257 bytes are where each packet is stored as it is read from or written to the tape. Although it is possible to have a packet that fills the buffer, the packets used in ordinary cassette operations never exceed sixty bytes, so locations 01FB to 02BF are generally free. Assign from 02BF down, to be safe. Note: In REV 1 and 2, the cassette buffer is located in 0BFF to 0CFF.

1000 to 13FF: Empty -- These locations correspond to the two empty chip locations on the computer board.

1400 to 1FC0: Programs -- The 51 programs (0 to 50) are packed into this area. Each program occupies 59 bytes.

1FC1 to 1FF0: Free non-volatile memory -- These 48 locations are not currently used. Assign from 1FC1 up, to be safe.

1FF1: Cassette type -- If bit 2 of this byte is set, normal cassette motor sense/control functions are enabled. If bit 2 is clear, the cassette motor is ignored.

1FF2: Program number -- The current program number, as shown in the display, is kept here.

1FF3: Modified flag -- The modified flag appears in bit 7 of this byte. The other bits must be zero.

1FFC: Attack threshold -- General modulation selection 13 (Threshold Velocity) and Ampl Touch settings 6 and 7 compare the velocity of each attack to this number, which must be between 0 and 31.

1FFD: Release threshold -- Envelope Release setting 31 causes each release velocity to be compared to this number, which must be between 0 and 31.

1FFE: Release slow value -- Envelope Release setting 31 causes this number to be used as the release parameter for releases slow than the above threshold. It should be between 0 and 31.

1FFF: Release fast value -- Envelope Release setting 31 causes this number to be used as the release parameter for releases faster than the above threshold. It should be between 0 and 31.

2006: LED port byte 1 -- This write-only location directly controls those eight LEDs that are capable of flashing. This location should not be directly referenced, as it is automatically rewritten from the **LED image byte 1** at regular intervals.

2007: LED port byte 2 -- This write-only location directly controls those eight LEDs that never flash. When this location is written, the **LED image byte 2** should also be written.

C100: Return -- This location contains a Return instruction. Software hook cells should always contain C100 when not in use.

APPENDIX D
COMMAND LISTING

This section provides a brief listing of the form of each command. A superscript denotes a byte count. For greater detail, refer to section 3.

Name	Command bytes	Response bytes
No Operation	00	
Identification	01	01 device revision
Read Program	02 prog	02 data^{59}
Write Program	03 prog data^{59}	
Load Packet	04	04 n data^{n}
Save Packet	05 n data^{n}	05 result
Read Parameter	06 prog param	06 value
Write Parameter	07 prog param value	
Panel Switch Off	08	08
Panel Switch On	09	09
Performance Switch Off	0A	0A
Performance Switch On	0B	0B
Peek	0C addr^{2}	0C n data^{n}
Peek Two Bytes	0D addr^{2}	0D data^{2}
Poke	0E addr^{2} n data^{n}	
Poke Two Bytes	0F addr^{2} data^{2}	
Tap Panel	10	
Unlock	11 00 FF	
Tape Space	12	12 result
Restore	13	08 0A 14
Pressure Switch Off	14	14
Pressure Switch On	15	15
Pressure	68+i key pressure	
Information	70+i	70+i boards 0 0 0
Volume	78+i volume	
Lever 1	80+i position	
Lever 2	88+i position	
Pedal 1	90+i position	
Pedal 2	98+i position	
Footswitch 1 Down	A0+i	
Footswitch 1 Up	A8+i	
Footswitch 2 Down	B0+i	
Footswitch 2 Up	B8+i	
Define	C0+i prog l1 l2 p1 p2 vol fsw	
Undefine	C8+i	
Attack	D0+i key velocity pressure	
Release	D8+i key velocity	
Set Parameter	E0+i param value	
Status	E8+i	E8+i prog l1 l2 p1 p2 vol fsw
Squelch	F0+i key	