FAIRLIGHT

MUSIC COMPOSITION LANGUAGE

REFERENCE MANUAL

FAIRLIGHT MUSIC COMPOSITION LANGUAGE

M . C . L .

REFERENCE MANUAL

FEBRUARY 1983

Fairlight Instruments Pty. Ltd.
15 Boundary Street
Rushcutters Bay
SYDNEY AUSTRALIA 2011
Telephone (02) 331 6333
Telex AA27998

TABLE OF CONTENTS Page

SECTION 1 - THE MCL CONCEPT..... 1

STRUCTURE 1
MCL CAPACITY 3
ACCURACY 3
SYNCHRONIZATION 3
PRINCIPLE OF OPERATION 4
FILE STRUCTURE 6
FILE MANAGEMENT 6

SECTION 2A - SEQUENCE FILES 7

DEFAULTS 7
REPEATS 9
RELATIVE SPECIFICATION 10
CONTROLS and SWITCHES 11
WAIT FUNCTION 13
PROMPT 14

SECTION 2B - PART and PIECE FILES 16

PART FILE 16
PIECE FILE 17

SECTION 3 - MCL COMMANDS 18

NEW 19
EDIT 19
LOAD 20
SAVE 20
PLAY 21
NAME 22
COPY 22
CLEAR 23
DIR 23
PRINT 23
COM 24
DIV 25
SYSTEM VARIABLES 26
Q 27
RESET 27
SPEED 28
SYNC 28
CLICK 29
X 30
TIME 30

SECTION 4 - DEBUGGING AIDS 33

RUN TIME ERRORS AND WARNINGS 33

SECTION 5 - MCL EDITOR 35

STATUS LINE 36
COMMAND LINE 36
CURSOR 36
MOVING IN THE EDITOR 38
INFINITY 38
LOCATE LINE 39
FIND STRING 39
CHANGE STRING 40
CHANGE PREVIOUS STRING 40
MUSIC KEYBOARD INPUT 41
DELETE CHARACTERS 42
DELETE LINES 42
RESEQUENCE LINES 43
DISPLAY TEXT 45
PRINT TEXT ON LINE PRINTER 46
READ TEXT 47
WRITE TEXT 47
COPY BUFFER 48
OPEN COPY BUFFER 48
OPEN FILE 49

APPENDIX A - AN EXAMPLE - Three Blind Mice 50

APPENDIX B - EXTERNAL SYNCHRONIZATION 53

CLICK OUTPUT 53
SYNCHRONIZATION TO MULTI-TRACK TAPE 54

APPENDIX C - ERRORS AND WARNING MESSAGES 56

APPENDIX D - SHORTCUTS 58

INDEX 59

INTRODUCTION) AN EXAMPLE

This following simple example will show basic concepts in understanding the Music Composition Language MCL. For a complete description of commands and music implementation see the relevant areas elsewhere in this manual. A contents list is at the beginning of the manual and an alphabetic index is at the back.

****

To begin with, load a voice on Page 2 - DISK CONTROL with NPHONY 8. A short pitched sound like a piano is best for monitoring the effect of MCL.

Go to Page 7 - CONTROL PARAMETERS and patch KEYVEL to LEVEL. This now means that the volume of each note typed in can be varied.

SET the DAMPING on Page 7 to a value of around 200.

Now go to Page C by typing . . . . . PC<return>

The display will now change and you will see the MCL prompt appear. The display should look like...

FAIRLIGHT COMPOSER LEVEL 5.5

>

Music information is entered and stored in SEQUENCE files. These files are similar to voice files except that the SEQUENCE files have a suffix .SS and vary in length depending on how much music information is contained in the sequence.

To begin a new sequence called TUNE1,

type . . . . . NEW TUNE1.SS<return>

The display will change and the CMI will be in the MCL editing mode with the current filename in the top right-hand corner.

The minimum information required is the pitch of the note. You don't have to specify each and every parameter of a note. The CMI will assume DEFAULT values if none are provided.

Play the four notes A B C D, two octaves from the bottom of the keyboard...

Type . . . . . O=2 A B C D<return>

You will see the string O=2 A B C D move from the top of the screen to the centre of the screen and become part of a sequence line with a line number...

```
0001 O=2 A B C D
```

The ARROW keys and the HOME key behave in much the same way as on other CMI Display Pages.
To hear these four notes A B C D, we need to leave the editor and get back to the original Page C display.

Type . . . . . E<escape> (not <return> this time)

The display will now show:

TUNE1.SS END OF EDIT

So type . . . . . P TUNE1.SS<return>

You should now hear four notes play.

Each time you want to add or change music information to TUNE1.SS

type . . . . . E<return>

The CMI remembers the name of the last file edited.
The CMI also remembers the name of the last file it played.

THE FOLLOWING EXAMPLES WILL ASSUME THAT YOU ARE TYPING

E<return>	to EDIT the sequence,
E<escape>	to leave the Editor and
P<return>	to PLAY the sequence.

N O T E I N F O R M A T I O N

PITCH

PITCH is the only information needed. All other information such as note length and note level (volume) are optional and may be left out. The CMI will supply a default value.

Enter the Editor again. (Type E<return>.)

The current line should look like...

```
0001| O=2 A B C D
```

meaning the notes A B C D played in the second octave of the keyboard.

Press the RIGHT ARROW key a number of times.
Your display may now look like this...

```
0001| O=2 A B C D
```

The gap in the lines around the 2 is the CURSOR or point where information can be added or deleted. Position the CURSOR over the 2 as above.

Type . . . . . 3<set>.

The result is A B C D played in the third octave of the music keyboard...

```
0001| O=3 A B C D
```

To make note D an octave lower than the other notes, move CURSOR to the right of D...

```
0001| O=3 A B C D
```

Now type . . . . . -<add>
and so note D is lowered an octave...

```
0001| O=3 A B C D-
```

The following would put the note D in the fifth octave...

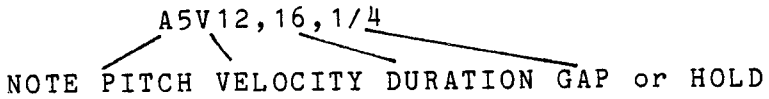
```
0001| O=3 A B C D+2
```

Another way to have note A played in the fifth octave would be to put 5 immediately after the D. Note D in the fifth octave...

```
0001| O=3 A B C D5
```

The sharp sign # is produced by pressing <control>S.
The flat sign ♭ is produced by pressing <control>F.
The natural sign ♮ is produced by pressing <control>D.

Additional note information is not essential in all cases but will be required for certain effects. In other words, the only thing you always have to type in is the PITCH of a note.
Additional note information occurs in the following order:



The following parameters are optional...

KEY VELOCITY Can control LEVEL and ATTACK if patched to those effects on Page 7. Range 0 to 15.
Individual note KEYVEL immediately follows pitch.

```
0001| V=8 A B C DV15
```
↑
This note will be the loudest.

DURATION Immediately follows pitch and KEYVEL with a comma and a number. If no number then assumes 1.

```
0001| O=3 A B C,2 D5,1/16
```

Note C is twice as long as notes A and B.
Note D is 1/16 as long as notes A and B.

GAP Immediately follows DURATION. It is a proportion of the total time that a note is OFF.
It must be less than total note length. Individual note GAP must be preceded by a comma.

A short GAP gives smooth legato notes...

```
0001| O=3 G=1/8 A B C D2,3,1/4
```
↑
Individual note gap

A long GAP gives short staccato notes...

```
0001| O=3 G=7/8 A B C D
```

HOLD can be used in place of GAP. It is the proportion of the total time that a note is ON.
It must be less than total note length. Individual note hold must be preceded by a full-stop.

A short HOLD gives short staccato notes...

```
0001| O=3 H=1/8 A B C D2,3.1/4
```
↑
Individual note HOLD

A long HOLD gives smooth legato notes...

```
0001| O=3 H=7/8 A B C D
```

C H O R D S

A chord, that is several notes played together, may be formed by putting the round left bracket symbol (at the start of the chord and the round right-hand bracket symbol) at the end of the chord. In all other respects (duration, level) the chord is treated as a single note.

Here is an A chord with a KEYVEL of 12 and a duration of 5.

```
0001| O=3 (A C# E)V12,5
```

R E P E A T S

Any amount of music may be repeated by putting the left bracket symbol < at the start of the repeat and the right bracket symbol > at the end of the repeat with the number of times to repeat immediately after the right-hand bracket >. Maximum number of repeats is 60,000.

Here the notes A B C and D in the third octave are repeated 1,000 times...

```
0001| O=3 < A B C D >1000
```

Repeats can be nested up to six times.

This extreme example is permissable...

```
0001| <<<<<<A B C>60000>60000>60000>60000>60000>60000
```

C O M M E N T S

Comments are for the assistance of the user and may be put anywhere in a sequence on their own line. The CMI will ignore comments.

A comment line always starts with an asterisk * .

Here is a typical comment line followed by a sequence line of four notes...

```
0001| * THIS IS THE FIRST MEASURE
0002| O=3 A B C D
```

Rest

R 1 = one unit rest

S A V I N G

To SAVE your work permanently onto floppy disk ...

type E<escape> ...exit from the Editor
type SAVE (name of sequence)<return>

The CMI will now save to disk. The saved sequence can now be seen on Page 2 - DISK CONTROL by typing +<return> or by pointing to the three letters MCL in the bottom right-hand corner.

L O A D I N G

LOADING copies a previously created (NEW) sequence file from disk into CMI hot memory for playing, or changing. Nothing is changed on disk until a subsequent SAVE command is issued.

MCL files cannot be loaded on Page 2 - DISK CONTROL.
MCL files can only be loaded on Page C - MCL.

To LOAD a previously created sequence file ...

type LOAD (sequence filename)<return>

****

This brief introductory example is meant to show just the very basic ways in which music may be implemented with MCL. There are many more commands and features available. For a more detailed discussion of these, see the relevant areas of this manual.

SECTION 1) - THE MCL CONCEPT

-FAIRLIGHT M.C.L. (MUSIC COMPOSITION LANGUAGE)-

STRUCTURE

The Composer is a TREE-STRUCTURED language. This means that a composition is treated as a "tree" containing three levels of hierarchy.

These levels are called PIECE (.PC), PART (.PT) and SEQUENCE (.SS). They are the three types of files that can exist on Page C.

The tree structure means that a PIECE consists of one to eight PARTS to be played simultaneously, corresponding to the eight KEYBOARDS found on Page 3.

Each PART consists of one or more SEQUENCES which are played sequentially. It is convenient to think of each PART as a different "musician"; each "musician" plays a series of SEQUENCES on his own keyboard. One musician may play another musician's SEQUENCE.

The playing of each PART is independent of other PARTS which may be playing at the same time. PART files may also specify which conceptual keyboard is to be played by that PART, allowing switching from one register of the Fairlight to another between sequences. A maximum of nine PART files can exist in a PIECE, eight for Page 3 Keyboards and one for Page 7 Controls and Switches, if required. See Section 3-MCL FILES under CONTROLS and SWITCHES.

The PIECE (.PC) behaves like a "conductor", instructing which PARTS are required to play. All named PARTS play simultaneously.

The PARTS (.PT) may call for any number of SEQUENCES, with the restriction that a maximum of 64 files may be resident in the system at the same time, irrespective of their size. In other words, a maximum of 64 .PC, .PT, and .SS files may be held in live memory on Page C simultaneously.

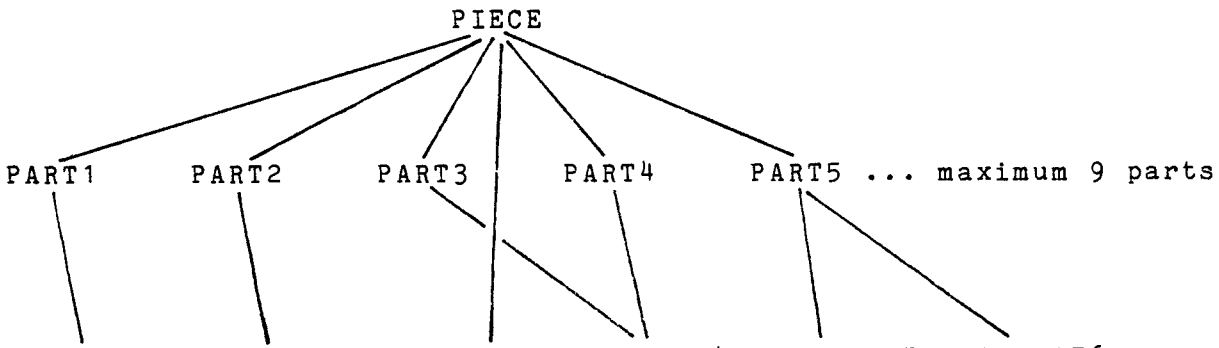
SEQUENCES (.SS) may be any length from 1 note to total memory workspace, although there is of course a limit to the total number of notes of sequence that can be accommodated at once. Note that more than one PART may share one or more common SEQUENCES. The one SEQUENCE can be played by several PARTS simultaneously without any interaction between them.

CHORDS are permitted in SEQUENCES (.SS) as long as the notes of the chord are all of the same duration and sufficient NPHONY (voices) has been allocated on Page 3.

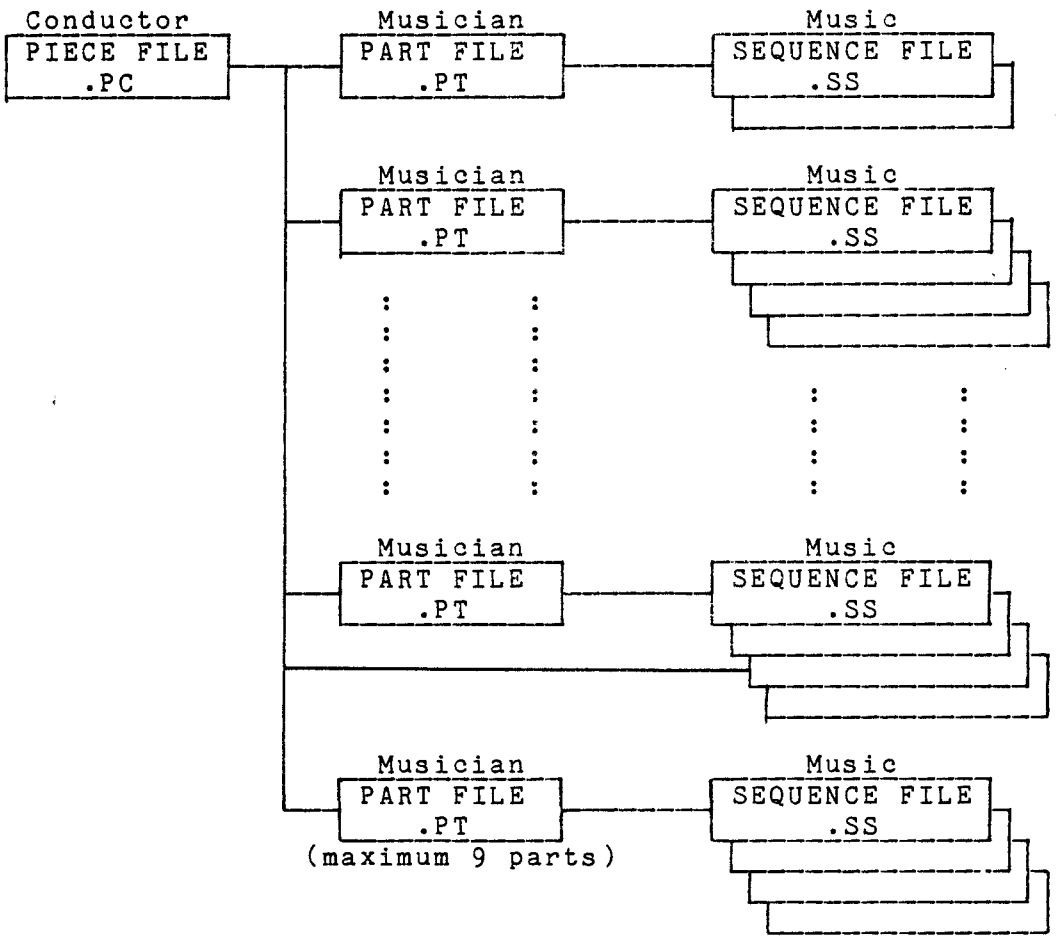
Sophisticated REPEAT directives are provided, so that redundant information is kept to a minimum. It is also possible to change the parameters of a sequence in certain ways each time it is repeated.

SECTION 1) - THE MCL CONCEPT (continued)

TREE STRUCTURE OF MUSIC COMPOSITION LANGUAGE - MCL



or



MCL CAPACITY

Due to the enormous number of different possibilities available to the musician using the Composer, it is meaningless to try to specify performance figures such as the maximum length of pieces possible or the number of keystrokes required to define a given "amount of music". The answer to most such questions will generally be "that depends...". Suffice it to say that under WORST CASE conditions, that is without using any of the "tree structure" or other short-cuts, several minutes of complex polyphonic music can be held in the Fairlight's main memory. Using the full facilities it is a simple matter to specify hundreds of hours of music with just a few minutes typing.
The MCL memory will hold around 15000 characters, and the COPY BUFFER will hold around 4000 characters.

ACCURACY

The time resolution of the Composer playing software is about 1 millisecond, so that very precise control of relative timing of events is possible. For example, rests of 1/100 of a beat can be used to generate subtle timing offsets to overcome the "mechanical" sound of a mathematically perfect performance. In fact, the timing control is sufficiently fine to achieve "flanging" effects by playing two identical voices slightly out of step.

SYNCHRONIZATION

The Fairlight can accept an external SYNC input from which the Composer replay system derives its timing reference. This can be used to synchronize to multi-track tape, motion-picture frames or any other source of pulses. By laying down a pilot track on tape before recording multi-part works, overall tempo variations common to all subsequent recording passes can easily be achieved.The CMI also has a fully variable digital metronome CLICK output. See APPENDIX B - SYNCRONIZATION.

It is also possible to synchronize a piece with any external event by use of the W (WAIT) function, which makes the composer halt at a predetermined place in the music until a key on the alphanumeric keyboard is struck.

PRINCIPLE OF OPERATION

The Composer system is supplied as part of a special Fairlight System Disk.
The middle two digits of the version number displayed on Page 1 will be C followed by the version of Composer on that disk.

V3.C5.10
↑
MCL Version Number

Like the Page 9 keyboard sequencer, the MCL system plays music via Keyboards 1 to 8 on Page 3 of the Fairlight. After the required selection of voices and register allocations have been made on Page 3, the MCL system is entered by typing

PC <return>

on any diplay page. The system will respond with the Composer sign-on message, followed by the MCL prompt. The prompt is the

>

symbol and when it appears at the bottom left of the screen, it indicates that the MCL is ready to accept a command. Various commands are available for creating, editing, saving, loading and playing Composer files. They are described in detail in SECTION 3 - MCL COMMANDS.

To enter a new composition, the MCL Editor is used to create a Sequence File.

This is done by typing

NEW <filename.SS><return>

(described in SECTION 3 - MCL COMMANDS) to create the new file. The EDITOR commands (see SECTION 5 - MCL EDITOR) are then used to enter the notes, durations and other musical data to create the Sequence file.

The information entered is not validated at this stage, that is you do not find out about errors until the MCL tries to play what you have specified. When all the data for the sequence has been entered, the Edit is ended by typing

E<esc>

The text disappears from the screen and the MCL prompt appears again. The Sequence can then be played immediately using the PLAY command (see SECTION 3). Any errors will be detected at run-time, that is, as the MCL tries to play what has been entered.

Invalid information in the sequence file will cause a WARNING message or an ERROR message to be displayed.

A WARNING results when certain limits are exceeded, but the music goes on regardless.

An ERROR results from feeding the MCL data which it cannot understand at all, in which case it stops playing. Both types of message will display the offending line along with an arrow pointing to the problem.

Various debugging aids are available to assist you in making corrections.

Further sequences can be subsequently entered and debugged in this way.

Which sequence is played is determined by the sequence file-name which is entered as part of the PLAY command.

To make a number of sequence file play sequentially, one after the other, their names are entered into a PART file (see SECTION 2B - PART and PIECE FILES), then the MCL is instructed to play that PART.

To play a number of sequences simultaneously, their names are entered into a PIECE file (see SECTION 2B - PART and PIECE FILES) and the PIECE file is played.

The various files created are NOT automatically saved on disc. They are saved using the SAVE command (see SECTION 3 - MCL COMMANDS) for later recall and editing.

MCL FILES

FILE STRUCTURE

When creating or editing a Composer file, line numbers are automatically inserted for the convenience of the musician. They are ignored by the system. Line numbering is useful for locating a particular point on a score if reference numbers are written down as the music is typed in. Run-time error messages print the line of text causing the error, along with the line number of the offending line.

Comment lines may be included in any file. These must have an

*

in the first non-blank character position following the line number. Comments are for the musician's reference only and are ignored by the Composer.

Three different types of file are used by the Composer.
These are PIECE (.PC), PART (.PT) and SEQUENCE (.SQ) files.

Extensive use is made of the concept of DEFAULTS. This is a time-saving technique whereby it is not necessary to specify every parameter controlling each event explicitly.
For example, if a whole series of notes are to be played in the same octave, it is possible to set a DEFAULT octave which the MCL will use whenever an octave number is not stated.

FILE MANAGEMENT

Composer file management is performed by the Fairlight on Page 2.
By typing

P2<return>

and then typing

+<return>

or pointing the lightpen to the letters

MCL

The displayed files will be those with .SS, .PT, .PC suffixes.
It is then possible to change the name of a file, delete or backup a file in the same way as voice files.

SECTION 2A) SEQUENCE FILES

Sequence files are a list of the items to be played. These include default settings, note specifications, repeat directives, control parameter changes, relative operations and comments.

To CREATE, EDIT, SAVE and LOAD SEQUENCE (.SS) files see Section 3 - MCL commands.

The first non-blank character of each line in a sequence file has a special meaning: * for comment lines (ignored by MCL).
Anything else means a line of music. Comment lines are especially useful to the user as a development aid. Sequence files and in fact all MCL files should be well planned and laid out.

DEFAULTS

DEFAULTS save typing effort allowing you to give a value to notes as a group rather than for each and every note.
DEFAULTS may be inserted anywhere in the music.
The following parameters may be specified as defaults:

OCTAVE specifies in which keyboard octave the specified note falls. Octave 1 is the lowest. Octave 6 is the highest.

EXAMPLE

All notes in octave 2 of the keyboard.

```
0001| O=2 A B C D
```

BEAT is the number of sub-divisions or more correctly the number of clock ticks within each time beat unit. The number can be from 2-255.

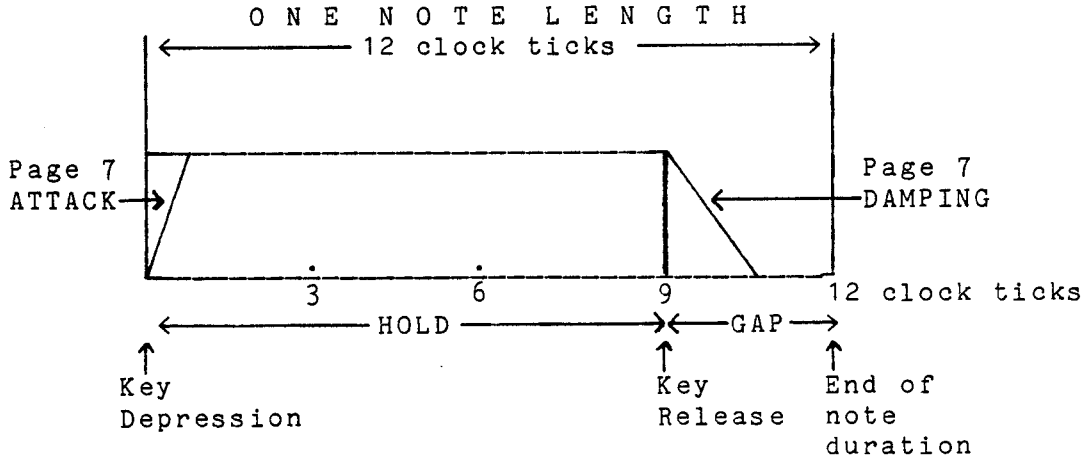
EXAMPLE B=16 means use default of 16 subdivisions per beat.

```
0001| B=16 A B C D
```

would play twice as fast as

```
0001| B=32 A B C D
```

EXAMPLE B=12 G=1/4



ACCIDENTALS

The sharp sign # is produced by pressing <control>S.
The flat sign ♭ is produced by pressing <control>F.
The natural sign ♮ is produced by pressing <control>D.

The accidental will affect the note immediately preceding it, if there is one. Otherwise it will affect the note following it.
If the accidental is by itself, an error message will result.

EXAMPLE AC#E The notes are A, C# and E.
AC #E The notes are A, C and E#.

KEY SIGNATURE allows defaulting certain notes to be assumed sharp or flat to achieve a certain key signature.
<control S>, <control F> and <control D> are used on the alphanumeric keyboard to specify sharp, flat or natural respectively. Don't confuse the <control S> character with the character on the "3" key of the typewriter keyboard.The key signature must be on a line by itself and preceded by an exclamation mark !.

EXAMPLE !<control S>F selects all F notes to be sharp unless otherwise directed.
EXAMPLE !<control S>F <control S>C <control S>G sets key signature of A major.

VELOCITY specifies the effective "key velocity" used when playing a note. The key velocity data is used exactly as if it had originated from the music keyboard. It can affect ATTACK or LEVEL (volume) if patched to those effects on PAGE 7. Velocity data must be in the range 0 to 15.

EXAMPLE V=15 is the fastest velocity. Maximum LEVEL. Fastest attack.
V=0 is the slowest velocity. No LEVEL. Slowest attack.

```
0001| V=4 A B C D     V=15 A B C D
       Quiet              Loud
```

See the description in SECTION 2A - SEQUENCE FILES
under CONTROLS and SWITCHES.

GAP is the time between the end of the current note and the start of the next note. It is in BEAT units. The GAP time must obviously not exceed the note time, and it must also be greater than zero. If GAP is fractional, then its denominator must divide equally into the BEAT value to avoid truncation errors.See diagram accompanying BEAT.
EXAMPLE

```
0001| B=12 G=1/4 A
```

This means that a note A will play for 9 clock ticks and release for 3 clock ticks making a total of twelve.

```
0001| B=12 G=1/5 A
```

This would introduce rounding errors as 5 does not divide equally into 12. Rounding errors may not really present an audible problem.
A long GAP, G=7/8, results in a staccato effect.
A short GAP, G=1/8, results in a smoother legato effect.

HOLD is used as an alternative to GAP, specifies the duration of the "key depression". It is in BEAT units. The HOLD time must obviously not exceed the note time, and it must also be greater than zero. If HOLD is fractional, then its denominator must divide equally into the BEAT value to avoid truncation errors as with GAP. See diagram accompanying BEAT.

EXAMPLE

```
0001| B=12 H=3/4 A
```

This means that a note A will play for 9 clock ticks and release for 3 clock ticks making a total of twelve.

Individual note HOLD must be preceded by a fullstop.

EXAMPLE A2,3.3/4 (is the same as A2,3,1/4)

Several different default parameters may be specified simultaneously for convenience.

EXAMPLE

```
0001| O=3 B=8 G=3/8 V=12 A B C D
```

TRANSPOSITION adds an offset to the note requested. It is specified as the number of keys up or down the keyboard the note is to be moved.

EXAMPLE Notes A B C D are two semitones higher.

```
0001| T=2 A B C D
```

Transposition is cumulative and may be positive or negative.

REPEATS

REPEATS may be put into sequences by enclosing the repeated section inside the brackets < and > immediately followed by the number of times to repeat. ANYTHING inside the left bracket <, and the right bracket >, will be repeated.

<A:B:C>3 is the same as A:B:C:A:B:C:A:B:C
REPEATS may be nested up to six levels.
<<A:B:C>2 D>2 is the same as A:B:C:A:B:C:D:A:B:C:A:B:C:D
REPEATS may be spread over more than one line, however there must be an equal number of left brackets and right brackets.

RELATIVE SPECIFICATION

The default values OCTAVE, TRANSPOSITION, VELOCITY and BEAT can be dynamically modified by specifying a RELATIVE value, that is to add or subtract some given number from the current value of the default. This allows major changes to be made to a sequence with just a few key presses.

The format is similar to setting a default, except that

a + or - is specified after the = sign

to indicate a RELATIVE change, rather than an ABSOLUTE change.

EXAMPLE O=+1 will increment the default octave number from whatever value it happens to be at that time. The effect of this is cumulative, so care is required to keep track of the current values, otherwise the permissible limits may be exceeded. If a value does exceed the limit, the Composer will generate a warning message when it attempts to play the sequence. Remember that the conventional default settings such as O=3 are absolute and will therefore always reset the default to the specified value.

EXAMPLE	O=3	O=+2	O=-3	O=4
	default octave is 3	default octave is 5	default octave is 2	default octave is 4

It is also permissible to specify a relative ratio (ie multiply or divide by a constant), where

* means multiplication
/ means division

EXAMPLE				
	B=20	B=*6	is the same as	B=120
	B=48	B=/8	is the same as	B=6

This method of changing speed enables different sequence files to have different BEAT values and still keep in step when a speed change is desired.
However the remarks made earlier about ensuring that the numbers are compatible to avoid rounding errors also apply here.

EXAMPLE				
	V=12	V=/4	is the same as	V=3
but	V=12	V=/5	is the same as	V=2

resulting in a rounding error.

CONTROLS and SWITCHES

Page 7 Controls, Switches and key velocity settings can be specified by MCL. There are six CONTROLS and five SWITCHES available as well as KEY VELOCITY.

EXAMPLE Go to Page 7. Observe the effects available.

Page 7 Effects

MODE	=	GLISSANDO	=SWTCH	LOOP CONTROL	=SWTCH
EXP	=	PORTAMENTO	=SWTCH	LOOP START	=CNTRL
LEVEL	=CNTRL	SPEED	=CNTRL	LOOP LENGTH	=CNTRL
FILTER	=	CONST. TIME	=SWTCH	START SEG	=CNTRL
DAMPING	=CNTRL	VIB. DEPTH	=CNTRL	SLUR	=SWTCH
ATTACK	=CNTRL	VIB SPEED	=CNTRL	SUSTAIN	=SWTCH

These are the Page 7 effects which can be either CONTROLLED or SWITCHED by the MCL. Six programmable Controls and five programmable Switches are available.

MODE	=	GLISSANDO	=	LOOP CONTROL	=
EXP	=	PORTAMENTO	=	LOOP START	=
LEVEL	=KEYVEL	SPEED	=	LOOP LENGTH	=
FILTER	=	CONST. TIME	=	START SEG	=
DAMPING	=	VIB. DEPTH	=	SLUR	=
ATTACK	=KEYVEL	VIB SPEED	=	SUSTAIN	=

These are the Page 7 effects which can be programmed by key velocity via the MCL.

Page 7 Controls and Switches can be programmed anywhere in a sequence (.SS) file by typing

for a CONTROL: Cn=<number or relative specification>

number between 1 and 6

number between 0 and 127

for a SWITCH: Sn=0 or 1

OFF or ON

number between 1 and 5

EXAMPLES

C4=0		Control 4 set to minimum value
C1=127		Control 1 set to maximum value
C2=10	C2=+15	is the same as C2=25
S1=0		Switch 1 is OFF
S1=OFF		Switch 1 is OFF
S5=1		Switch 5 is ON
S5=ON		Switch 5 is ON
V=15		KEYVEL at maximum
V=0		KEYVEL at minimum

Note Do not confuse C1, the Control with C1, the note.

EXAMPLE

MODE	=	GLISSANDO	=	LOOP CONTROL	=
EXP	=	PORTAMENTO	=SWTCH5	LOOP START	=
LEVEL	=KEYVEL	SPEED	=CNTRL5	LOOP LENGTH	=
FILTER	=	CONST. TIME	=	START SEG	=
DAMPING	=CNTRL2	VIB. DEPTH	=	SLUR	=SWTCH5
ATTACK	=CNTRL1	VIB SPEED	=	SUSTAIN	=

In the above example,

LEVEL (volume)	will be controlled by key velocity
ATTACK	will be controlled by Control 1
DAMPING	will be controlled by Control 2
PORTAMENTO) and SLUR)	will be controlled by Switch 5
PORTAMENTO SPEED	will be controlled by Control 5

The MCL keeps track of control settings for as long as a piece is in progress. The MCL Controls and Switches will override the physical Controls, Switches and footpedals on the music keyboard as long as you do not manually change them.

EXAMPLE Control 1 has previously been patched to LEVEL on Page 7. The following will slowly increase LEVEL from zero to maximum.

```
C1=0    < R   C1=+1 >127        Total time
                                equivalent
                                to R,127
Zero
LEVEL
        Time    Control 1
        between increased
        each    by 1        Repeat
        increment           127 times
```

EXAMPLES

C1=0 < R,1/10 C1=+1 >127	Total time equivalent to R,127/10 or R,12 R,7/10
LEVEL increases quickly from zero to maximum	

Note C1=0 < C1=+1 >127 Not valid as there is no "time" between each increment

Note < C1=0 R C1=+1 >127 LEVEL never increases

If more than one sequence (.SS) file is assigning values to the same CONTROL care must be taken not to exceed permitted values.
It may be convenient for the user to group all the CONTROL and SWITCH settings in one separate sequence file.

Controls and Switches may be used in PART (.PT) files. See the Section on PART files.

WAIT FUNCTION

By inserting the character

W

anywhere in a sequence file the music can be synchronized with any external event. The WAIT function however, must be enabled. See SECTION 3 - MCL COMMANDS under WAIT.

If the WAIT function is enabled (by WAIT=ON), then when the PLAY command encounters the letter W, the playing of music will be suspended until any key on the alphanumeric keyboard or music keyboard keypad is struck. The key may be struck in advance so that there is no wait at all. The music will then resume as if nothing had happened.

SINGLE STEPPING

It is possible to halt a piece of music at any time by typing

<control W>

that is holding down the <control> key and pressing "W".
This will have exactly the same effect as if a WAIT command (W) had been inserted in the piece being played, except that it will function regardless of whether the WAIT feature is enabled or not.

The single step feature is an extension of this. If the music is halted (by the <control W> combination or the WAIT function), then pressing

<control W>

will advance the music to the next note, skipping rests. It will then stop again. This process can be continued indefinitely by continually pressing <control W>.

Note that since, in general each note that you hear is two time units, (key depression and key release), the CMI will play one note or chord for each two <control W>'s.

PROMPT

This feature allows a piece of music being played by MCL to give cues to the musician by displaying pre-programmed messages on the graphics screen or the music keyboard display. Thus you may have for example, song lyrics appearing on the screen as the music is being played.

It is also possible to send data to another computer or synthesizer which can be connected to the CMI's PRINTER output on the rear panel or parallel interface output.

Even though messages may be embedded in sequence files, they will not be displayed unless

PROMPT=ON

on Page C. See SECTION 3 - MCL COMMANDS.

An output message (or control data) can be inserted in any sequence file (or Part file - see PART FILE), by enclosing the message, or string of characters, in square brackets [ ].

These brackets are obtained by typing

<shift K> for [and <shift M> for]

The message can be anywhere in the sequence.

Any printable character may be sent. By using the special character : (colon) any unprintable character such as <return>, <rubout>, may be sent. The colon indicates either a hexadecimal value or a control character definition.

If the character following the colon is a valid hex digit (in the range 0-9 or A-F) then it and the following character are converted to a hexadecimal number. If the second character is not a valid hex digit then the prompt string will be terminated. The first character after the colon may also be one of the following special control characters:

R, L, P, N, S, X, H which will cause the corresponding control character below to be sent:

:R	will send	<carriage return>	hex 0D
:L	will send	<line feed>	hex 0A
:P	will send	<form feed>	hex 0C
:N	will send	<null>	hex 00
:S	will send	<escape>	hex 1B
:X	will send	<backspace>	hex 08
:H	will send	<cancel>	hex 18

In these cases only the first character after the colon is treated as a special character. It is not necessary to use any of these control characters when sending messages to the music keyboard display. The other special character, " (double quote), forces the following character to be transmitted exactly as is.

Thus to send a colon use ":
To send a right square bracket use "]
To send a double quote use ""

EXAMPLES

[ABCDEFG]	sends	ABCDEFG
[ABCDEFG:R:L]	sends	ABCDEFG<return><line feed>
[:7f"]":"":0A]	sends	<rubout>]:"<line feed>

The timing is arranged so that the transmission of the string will begin at exactly the time specified in the sequence file (provided the output device is ready), and will continue as a background task to the playing of music. Thus the time taken to output the string will not affect the timing of the music in any way. If a second literal output request is encountered before the previous one is completed then the first one will be truncated and the next one will start immediately.

SECTION 2B - PART and PIECE FILES

PART FILE

A PART (.PT) file is a list of SEQUENCE files (.SS) to be played sequentially by that PART. The analogy may be made of the musician (PART file) playing music (SEQUENCE file).

To CREATE, EDIT, SAVE and LOAD PART files
see SECTION 3 - MCL commands.

Keyboard number played may also be set here,
by typing

!K=number between 1 and 8

These are the same eight keyboards as those on
Page 3 - KEYBOARD CONTROL.

EXAMPLE

```
0001 !K=1         Play the following on Keyboard 1.
0002 SEQ1.SS      SEQ1.SS is played on Keyboard 1.
0003 SEQ2.SS      SEQ2.SS is played on Keyboard 1 after SEQ1.SS.
0004 !K=8         Now switch to Keyboard 8.
0005 SEQ3         SEQ3.SS is played on Keyboard 8. The ".SS"
                  suffix is optional in a PART file. The CMI
                  assumes SEQ3 is a sequence.
```

It is also possible to insert lines of SEQUENCE file information (notes, rests, chords and default information) into a PART file by placing a double quote character " in the first position in the line.

Whatever follows this character will be treated exactly as though it was within a sequence file. Thus, if it is necessary for a PART to play a SEQUENCE, then wait a while before playing the next SEQUENCE, the required number of rests can be inserted in the PART file.

EXAMPLE

```
0001 !K=1         Play the following on Keyboard 1.
0002 SEQ1         SEQ1.SS is played on Keyboard 1.
0003 "R,4         Wait for four beats before playing SEQ1.SS.
0004 SEQ1         SEQ1.SS is played again on Keyboard 1 after
                  4 beats have elapsed.
```

PIECE FILE

PIECE files are a list of PART files which are to be played simultaneously. The analogy may be made that the PIECE (.PC) file is the conductor of the orchestra and the PART (.PT) files are the musicians.

To CREATE, EDIT, SAVE and LOAD PIECE files see SECTION 3 - MCL COMMANDS.
If the Keyboard number is not included in the PART file, then the order in which the PART files are listed in the PIECE file determines the keyboard allocated to that PART.

EXAMPLE

```
0001 TUNE1.PT}                      play TUNE1.PT on Keyboard 1.
0002 TUNE2.PT}                      play TUNE2.PT on Keyboard 2.
0003 TUNE3    }   simultaneously    play TUNE3.PT on Keyboard 3.
0004 SEQ7.SS  }                     play SEQ7.SS on Keyboard 4.
```

In general PIECE files will contain only PART files, so the .PT suffix on filenames is optional and may be omitted as with TUNE3 above. However PIECE files may contain SEQUENCE filenames provided that the suffix .SS is explicitly stated.

Defaults can be included in a PIECE file. They must be preceeded by an exclamation mark !.

SPEED, PROMPT, WAIT, CLICK are the permissable defaults.

```
To default      SPEED   type    !S=number between 1000 and 65535
                CLICK   type    !C=number between 0 and 255
                Note    You still have to set CLICK=ON on
                        Page C SYSTEM VARIABLES.

                WAIT    type    !W=ON or !W=1
                PROMPT  type    !P=ON or !P=1
```

All defaults can be on one line in a PIECE file for convenience.
EXAMPLE

```
0001| !S=3500 P=ON W=1 C=200,16
```

Spaces have been inserted for clarity.

SECTION 3) - MCL COMMANDS

MCL COMMANDS

A number of commands are available for selecting various MCL functions.

These include

NEW	files being created in CMI memory.
LOADING	files from disk to CMI memory for subsequent editing or playing.
EDITING	files to make changes or add extra information.
PLAYING	files.
SAVING	files from CMI memory to disk for later recall.

The three permitted file suffixes in MCL are

.SS for Sequence files
.PT for Part files
.PC for Piece files

MCL filenames may be viewed on Page 2 either by typing

+<return>

or pointing the lightpen at the area that reads MCL.

Filenames may be any valid Fairlight-type name, eight characters maximum, first character alphabetic. Identical filenames may co-exist as long as the suffixes are different.

EXAMPLES

FRED1A.SS	valid.
Q137BAT0.SS	valid.
1PATTERN.PC	invalid - first character not alphabetic.
SARABANDE.PT	invalid - more than eight characters.
JAMES$1.PT	invalid - $ not a number or letter.
OPUS.PC)	valid - different suffixes.
OPUS.PT)	
OPUS.SS)	

The following commands are for use outside the "Editor". That is, they are not used to directly input music information into the CMI. Rather the commands move blocks of information around in convenient forms.

NEW

```
NEW,<filename1.XX><,filename2.XX><return>
```

Create a new file	name of new file	suffix either .PC, .PT or .SS	optional	terminate the command

Create a new MCL file. The file will be created in memory but no disk allocation will be made.

Spaces may be used in place of commas.

<filenames> must not already exist in CMI memory.
If <,filename2.XX> is specified then it will be created to be a copy of <filename1.XX>, which must exist in memory.

EXAMPLE

NEW,TUNE1.SS<return>	A new file is created called TUNE1.SS. At this point the display will change and the "Editor" will be seen. To "escape" from the Editor back to Page C, type E<escape>.
NEW,TUNE1.SS,TUNE2.SS<return>	A new file is created called TUNE2.SS and it is a copy of TUNE1.SS which must already exist in main memory. This is similar to the COPY command.

EDIT

```
EDIT,<filename.XX><return>
  or E,<filename.XX><return>
```

Edit a file	name of existing file	suffix either .PC,.PT or .SS	terminate the command

Edit a file. At this point the display will change and the "Editor" will be seen. To "escape" from the editor back to Page C, type

E<escape>

<filename> must previously have been created with the NEW command or have been LOADED from disk. The letter E can be used instead of EDIT.

The CMI will automatically remember the last file edited so if you are constantly editing the same file it is only necessary to type

E<return>

after the initial edit.

EXAMPLE

E TUNE.SS<return>	Edit MCL file TUNE.SS. Comma may be replaced by space.

Refer to SECTION 5 - MCL EDITOR for details of Editor commands.

LOAD

LOAD,<filename.XX><return>
or L,<filename.XX><return>

Load a previously SAVED file from disk into memory.

EXAMPLE

L TUNE.SS<return>	MCL file TUNE.SS is loaded. TUNE.SS must exist on disk or message "TUNE.SS" not found will be displayed. Comma may be replaced by space.

A whole PIECE with its associated PARTS and SEQUENCES can be loaded by specifying brackets "((" before the filename.

EXAMPLE

L ((TUNE.PC<return>	Load TUNE.PC and all its associated PARTS and SEQUENCES.
L (TUNE.PC<return>	Load TUNE.PC and all its associated PARTS.
L TUNE.PC<return>	Load TUNE.PC only.
L (TUNE.PT<return>	Load TUNE.PT and all associated SEQUENCES.
L TUNE.PT<return>	Load TUNE.PT only.
L ((TUNE.PT<return>	Invalid use of brackets.
L (TUNE.SS<return>	Invalid use of bracket.

SAVE

SAVE,<filename.XX><return>

Save a file from memory to disk.

EXAMPLE

SAVE TUNE.SS<return>	MCL file TUNE.SS is saved. Comma may be replaced by space.

A whole PIECE with its associated PARTS and SEQUENCES can be saved by specifying brackets "((" before the filename.

EXAMPLE

SAVE ((TUNE.PC<return>	Save TUNE.PC and all its associated PARTS and SEQUENCES.
SAVE (TUNE.PC<return>	Save TUNE.PC and all its associated PARTS.
SAVE TUNE.PC<return>	Save TUNE.PC only.
SAVE (TUNE.PT<return>	Save TUNE.PT and all associated SEQUENCES.
SAVE TUNE.PT<return>	Save TUNE.PT only.
SAVE ((TUNE.PT<return>	Invalid use of brackets.
SAVE (TUNE.SS<return>	Invalid use of bracket.

Once a new file has been saved, the filename can be seen on Page 2.

The ability to save files is most important and fundamental.

Note The letter "S" cannot be used as an abbreviation of SAVE. It is reserved for use with SPEED control that is S=5000 means set SPEED to 5000.

PLAY

PLAY<,k><return>
or P<,k>,<filename1,filename2,filename3,..... ><return>

Play all named files simultaneously. The files may be a mixture of PART and SEQUENCE files.

<k> is an optional Keyboard number between 1 and 8. The files specified will be assigned to successive keyboards starting with keyboard "k" if specified.

If "k" is not specified, the first file will be played on Keyboard 1, the second to Keyboard 2, and so on.
If the Keyboard number is specified in the PART file, then that will take precedence.

EXAMPLE

```
P,5,TUNE.SS      Play TUNE.SS on Keyboard 5
```

Alternatively, and more commonly a single PIECE filename may be specified.

EXAMPLE

```
P MOVE1.PC<return>  Play MOVE1.PC and all associated
                    PARTS and SEQUENCES
                    Comma may be replaced by space.
```

The letter P may be used instead of the word PLAY. The CMI will automatically remember the last file played so if you are constantly playing the same file it is only necessary to type

P<return>

after the initial PLAY command.

TO abort a PLAY, type <control ESCAPE>
That is hold down the <control>key and press the <escape> key.

NAME

```
NAME <oldname.XX>,<newname.XX><return>
     |           |          |
  Command      Old        New name
               name
```

The NAME command allows renaming of a file or changing a file suffix in CMI live memory.

EXAMPLE

```
NAME A1.PC,BLUES.PC<return>     File called A1.PC is now
                                called BLUES.PC
NAME ZONT.PT,ZONT.SS<return>    File called ZONT.PT is
                                now called ZONT.SS
```

COPY

```
COPY <oldname.XX>,<name1.XX><,name2.XX><,.......><return>
       original     copies
       name
```

The COPY command allows copying of a file already loaded or created in CMI memory any number of times up to the maximum number of files permitted in MCL memory, that is 64.

EXAMPLE

```
COPY A1.SS,A2.SS,A3.SS ..... <return>
```

CLEAR,<filename.XX><return>

Remove the file specified from memory. This makes the memory and directory space occupied by that file free but it does NOT save the file to disk before clearing it.

A special use of the CLEAR instruction is

CLEAR,*<return>

which will clear all MCL from the CMI effectively resetting memory.

EXAMPLE

CLEAR TUNE.PT<return>	Clear the file called TUNE.PT from CMI memory. Comma may be replaced by space.

DIR<return>

Display a directory or list of filenames currently loaded into the CMI's memory.

DIRA<return>
or DIR A<return>

Display a directory or list of filenames currently existing in the CMI's memory as well as the size and addresses of each file beside each name.

PRINT <filename.XX><return>

Print the file specified on the line printer. The printer is optional and can connected via the PRINTER output on the rear panel of the CMI. Both daisy-wheel and dot matrix (for screen dump graphics and future music scoring) can be driven by the CMI.

The SHARP, FLAT and NATURAL symbols will be printed as s, f, and n.

The same format as the LOAD command can be used to print parts and sequences automatically.

EXAMPLE

PRINT ((TUNE.PC<return>	will print the PIECE and associated PARTS and SEQUENCES.
PRINT (TUNE.PC<return>	will print the PIECE and associated PARTS only.

COM

The COM (COMPILE) command will convert MCL pieces to Page 9 sequencer (.SQ) files so that they can be replayed or merged on Page 9 KEYBOARD SEQUENCER. A typical use would be to have the rhythm tracks done in MCL, converted to a Page 9 .SQ file and merged with keyboard solos and improvisations.

The COM format is as follows:

```
COM<,output filename><,keyboard><input filename(s)><return>
 |          |                  |                |               |
 |          |                  |                |               |
command     |              optional             |           terminate
            |                                   |           the command
        Page 9                               Page C
        .SQ file                             MCL files
```

All the parameters are identical to the PLAY command except for <output filename>. This is the Page 9 sequence file which is to be created and must have either the suffix .SQ or no suffix at all.

If the output file already exists then you will be asked whether you want to overwrite the existing file. Any response other than

Y<return>

will terminate the compile command.

If a COMpile has already been done then the previous output filename will be assumed if none is specified. That is, to repeat the same COMpile it is only necessary to type

COM<return>

The input filename can be any MCL file or list of up to eight sequence files and part files. If no input filename(s) are specified then the same default name as would have been used by the PLAY command will be taken. If an input filename is specified then the default PLAY filename will be changed to the name specified in the COM command. If the keyboard number is to be specified but no output filename then only one comma should be used between the COM and the number.

EXAMPLE

```
COM,TUNE.SQ,TUNE.PC<return>
    |       |
  Page 9   Page C
  file     file
```

The above will make a new Page 9 sequence called TUNE.SQ, exactly the same as TUNE.PC leaving TUNE.PC unchanged. If TUNE.PC is very long or has many repeats then TUNE.SQ will use much disk space. MCL files are very compressed compared to Page 9 files.
Note that if you have any of the system functions enabled that affect the MCL music, such as WAIT, CLICK, X then these will have an effect on the final COMpile.

EXAMPLE

The X function can selectively omit certain parts of sequences. Its main purpose is to enable the user to hear the end of a long piece without having to listen to everything up to that point.

If the X function is disabled, that is

X=OFF

then no music is omitted and the COM command converts everything.

If the X function is enabled, that is

X=ON

then COM only converts those parts not omitted, which may not be desireable.

When the compilation is over, you will then be able to merge TUNE.SQ with live keyboard playing.

DIV

Calculate the number of beats corresponding to a certain number of clock ticks at a given B value. A BEAT (B) is just a number of clock ticks.

The command format is:

```
DIV,<ticks>,<B value><return>
 /      |        |       |
command number  BEAT   terminate
        of      value  the
        clock          command
        ticks
```

The output produced is as follows:

<num>/<denom> OR <quot> + <rem>/<denom>

Where <num> and <denom> are the improper fraction corresponding to the required number of beats (i.e., same as <ticks>/<B value> but reduced to lowest terms), <quot> is the integer quotient of <num> and <denom>, and <rem> is the remainder.

This command is intended for converting the numbers of clock ticks output by the TIME (?) function.

By putting a question mark ? anywhere in a sequence file and setting the system variable

TIME=ON

on Page C, timimg messages will appear on the screen when the sequence is played revealing the number of clock ticks elapsed up to the "?".

EXAMPLE

Assume that a ? request informs you that at a certain point in a sequence 1000 clock ticks have elapsed.

You want to synchronize a second sequence so that it starts playing at this point. The second sequence has a BEAT value of 36.

How many rests are required?

Use the DIV command as follows:

```
DIV 1000 36<return>     Divide 1000 by 36.
                        Commas can be replaced by spaces.
```

The following output will appear on the screen

250/9 OR 27 + 7/9

Thus you can either use

```
    B=36   R,250/9          to produce the required delay
or  B=36   R,27 R,7/9       of 1000 clock ticks.
```

It is generally preferable to use the second format as the first method stands a greater chance of producing an arithmetic overflow.

SYSTEM VARIABLES

System variables are programmable by the user. Thet do not change the music but rather assist the user in developing extended works.

These variables are:

SPEED SYNC CLICK X TIME WAIT WARN PROMPT and FREE MEMORY (unused memory)

Q

This is the QUERY command which displays the state of the various system variables.

EXAMPLE

```
       Q<return>              SPEED=10000
                              SYNC=INT
                              CLICK=OFF : 48,4
                              X=OFF
                              TIME=OFF
                              WAIT=OFF
                              WARN=OFF
                              PROMPT=OFF
                              FREE MEMORY = $3C00/15360
       ↑                          ↑
       User                   CMI response
```

SYSTEM QUERIES

To find out the current individual settings for

SPEED, SYNC, CLICK, TIME, WAIT PROMPT and X

type in the name of the variable followed immediately by a carriage return. The system will respond by displaying the current value on the graphics screen.

EXAMPLE

```
       PROMPT<return>                 PROMPT=OFF
         ↑                               ↑
       User                           CMI response
```

It is quicker and probably better to just type Q<return> and get the status of all the system variables at once.

RESET

Reset the values of all CMI system variables (see above under Q command) to their defaults.

EXAMPLE

Before RESET<return>	After RESET<return>
SPEED=5236	SPEED=10000
SYNC=EXT/2	SYNC=INT (EXT division set to 1)
CLICK=ON : 200,8	CLICK=OFF : 48,4
X=ON	X=OFF
TIME=ON	TIME=OFF
WAIT=ON	WAIT=OFF
WARN=ON	WARN=OFF
PROMPT=ON	PROMPT=OFF
FREE MEMORY = 3C00/15360	FREE MEMORY = 3C00/15360

FREE MEMORY is not affected by RESET<return>.
FREE MEMORY is effectively reset by CLEAR *<return>.

SPEED

SPEED=number between 1000 and 65000
or S=number between 1000 and 65000

Set replay clock speed in microseconds per tick.
SPEED=10000 means 10 ms. / tick at which speed using B=24, a note of time-value 1 will last 240 ms.

Therefore when S=1000 means 1 ms. / clock tick.
S=60000 means 60ms. / clock tick.

The default speed on CMI system start-up is 10000.
The letter S on its own can be used instead of the word SPEED.

The SPEED setting can be put in PIECE (.PC) files as a default.

SYNC

SYNC=INT or EXT or number

Select the M.C.L. replay synchronization mode.

The external SYNC facility can be used to co-ordinate multi-track overdubbing of sequences by taping the SYNC tone and using the "sync-head" replay from the recorder to feed the external SYNC input of the CMI. This allows CMI replay tempo to be controlled by varying the external SYNC frequency.

When SYNC=INT (default) the SPEED control defines the microbeat duration in cycles of the CMI system clock (1.00525 MHz).

When SYNC=EXT the SPEED is ignored, and a microbeat becomes ONE CYCLE of the EXTERNAL SYNC INPUT which must be connected at the back panel.
SYNC input connection is Pin 2. Pulses or tone of 1 to 20 volts P-P. Waveform unimportant. Frequency range 2Hz to 5KHz. Impedance 10K ohms.
For accurate synchronization it is best to have a clean start to the tone and free from reverb or any effects.
Record the tone by itself (or while monitoring the CMI) rather than while taping the music.
It is not a good idea to synchronize the CMI to an external click track such as those produced by drum machines or analog sequencers. The CMI has a timing resolution of the order of milliseconds, whereas click tracks typically resolve to tenths of seconds or more, severely restricting the resolution capabilities of the CMI.
If a synchronized click track is required, the CMI can produce it at any required rate. See CLICK command.

For equal tempo between external and internal sync:

$$\text{SPEED} = \frac{2010.5}{\text{ext SYNC in KHz}} \qquad \text{EXT SYNC in KHZ} = \frac{2010.5}{\text{SPEED}}$$

EXAMPLE

An EXTernal SYNC tone of 1000HZ is equivalent to an INTernal speed of 2010.

SYNC="number" means use external SYNC divided by the number.

The permissible range of values for external SYNC is 1 to 65 536.

EXAMPLE

SYNC=5<return> means that if an external tone of frequency 2000Hz was used the CMI would divide that tone by 5 to 400Hz and use 400Hz as its reference.

CLICK

CLICK=<rate><,intro><return>

means generate a click track from the internal digital metronome via the SYNC output and/or monitor amplifier.

"Rate" is specified in BEAT units.

"Intro" is the number of clicks to be issued before the playing starts.

NOTE: The CLICK is found on three outputs:

1) SYNC socket (Pin 3)
2) Monitor Speaker Output
3) Phones Output

The CLICK volume is variable from the SYNC VOL MON control on the rear of the CMI.

The range of values allowed for rate and intro is 0 to 255. If no value is given for intro then the number 4 will be assumed.

EXAMPLE

CLICK=200,8<return> One click every 200 clock ticks
8 clicks introduction

CLICK=ON and CLICK=OFF can be used to turn the CLICK function on and off.

CLICK=<number>,<number> can be put into PIECE (.PC) files as a default however CLICK=ON or OFF is not permitted as a default.

X

This function enables the user to selectively omit certain parts of a piece when the PLAY command is used, while still keeping all parts in correct synchronization with each other.

By inserting the character X in a PART or SEQUENCE file, followed by a plus (+) or minus (-) sign the user can turn the actual playing of the piece on and off if X=ON on Page C.

```
X-    will turn the music off in the sequence file
X+    will turn the music on  in the sequence file
```

Note that the COMPOSER still has to read all of the music, in order to keep track of default values, timing etc., but no music will be played. Because of this, there may be a delay of a few seconds (or even a minute if it is a very long piece) before the music resumes, while the Composer skips through the files looking for the next X+.

Redundant X+ and X-'s (such as X- when the music is already off) will be ignored. This feature is especially useful when the musician is working on the end of a long piece, as it enables him to miss out the parts which have already been corrected, and just listen to the part he is working on.

Turn the X function on or off. This is actually outside the Editor.

Before X=ON<return>

```
SPEED=10000
SYNC=INT
CLICK=OFF : 48,4
=>X=OFF
TIME=OFF
WAIT=OFF
WARN=OFF
PROMPT=OFF
FREE MEMORY = 3C00/15360
```

After X=ON<return>

```
SPEED=10000
SYNC=INT
CLICK=OFF : 48,4
=>X=ON
TIME=OFF
WAIT=OFF
WARN=OFF
PROMPT=OFF
FREE MEMORY = 3C00/15360
```

TIME

When creating a piece of music using the composer it is often necessary to calculate the number of beats up to a certain point in the music, or between two points, in order to synchronize with another part. The TIME function (in conjunction with the X function described above) can be used to ease this problem.

By inserting a question mark at any point in any sequence file, the computer will, when it reaches that question mark during a PLAY command, display the following data:

```
   FILE:<seq>/<part>/<kbd> TIME = <time1>/<time2>
            /<line of sequence containing the ? >

            name of              time      time
            part       keyboard  since     since
   name of  playing    being     last      the
   sequence the        played    timing    start
   (.SS)    sequence             message
```

EXAMPLE

A typical display after the PLAY command has been given. TUNE.PT is playing FRED.SS on Keyboard 1.

```
>P
P TUNE.PT
  FILE:FRED     .SS/TUNE     .PT/1-TIME=100/100
0001 B=100  A4V15 ? R,1/100 ?
                  ↑
  FILE:FRED     .SS/TUNE     .PT/1-TIME=1/101
0001 B=100  A4V15 ? R,1/100 ?
                            ↑
>
```

Time elapsed up to the first ? in line 0001
of FRED.SS is 100 clock ticks.
Time elapsed from the first ? to the second ?
in line 0001 of FRED.SS is 1 clock tick.
Total time elapsed from the beginning of FRED.SS
to the second ? in line 0001 of FRED.SS is 101 clock ticks.

When the MCL is finished the CMI ready sign > is displayed.

Both the X and the ? features can be used in PART files by making use of
the QUOTE feature (").

See under PART files. Thus to listen to the last sequence only in a part, and to find out how many ticks into the piece it starts, the following construct may be used in the PART file:

```
0001 * TURN THE MUSIC OFF
0002 "X-
0003 SEQ1.SS
0004 SEQ2.SS
0005 SEQ3.SS
0006 * NOW TURN IT BACK ON AND FIND OUT
0007 * HOW MANY TICKS HAVE ELAPSED
0008 "X+ ?
0009 SEQ4.SS
```

SECTION 4) - DEBUGGING AIDS

```
**************************
**                      **
**   DEBUGGING AIDS     **
**                      **
**************************
```

RUN-TIME ERRORS AND WARNINGS

When you get the CMI to play MCL music (that is when you use the PLAY command) the CMI starts looking at the PIECES, PARTS or SEQUENCES and performing according to what it encounters within the files.

At this time various typing mistakes and incorrect ways of expressing ideas may be detected. Appropriate messages will appear on the screen.

The MCL differentiates between two levels of severity of such problems by classifying them as WARNINGS or ERRORS.

```
****************
**              **
**   WARNING    **
**              **
****************
```

WARNINGS will not stop music from playing. The CMI will cope as best it can.

Warning messages are given for some conditions reaching a limit. These are:

a) OCTAVE value out of range (not in range 1-7)
b) VELOCITY value out of range (not in range 0 to 15)
c) GAP time (or HOLD time) of zero
d) GAP time (or HOLD time) longer than the note
e) Control value (from Page 7) out of range (not in range 0-127)

When an out-of-range condition occurs the value will be forced to the limiting value which was exceeded. In the case of GAP and HOLD time, if the value calculated is zero then a value of 1 clock tick will be used. If the value is greater than or equal to the note time then a value of note time minus 1 clock tick will be used.
Warning messages are displayed in the same format as error messages but the associated part will not be terminated and playing will continue as normal. However if the warning appears very near the start of a "play" there may be a delay (and some lost clock ticks) because the message requires a disk access to be made.

When the CMI is first loaded the WARN function is OFF and the limiting condition is handled as above, but no message appears and no time will be lost by disk access.

To see messages for debugging use the command

WARN ON<return> (or WARN=ON<return>)

After a piece is debugged then the warning messages can be disabled by the command

WARN OFF<return> (or WARN=OFF<return>)

ERRORS result from :

a) PARTS or PIECES requesting files that do not exist or are not loaded.
b) Syntax errors in the files.
c) Any characters in a file which MCL does not recognize.

The user should consult the relevant areas of this manual to ascertain the nature of the error.

When an ERROR is detected, the PART or SEQUENCE concerned will stop playing, whereas after a WARNING the PART or SEQUENCE concerned will continue to play.

All messages are accompanied by the name of the file in which the error occurred. The offending line and its line number will be displayed, and an upward arrow will be displayed underneath the first character in the line which seemed to be responsible for the error. In some cases this will point exactly to the error, in other cases it will only provide a clue.

Note that the interpretation of composer files is always some time ahead of the music you hear, so that error messages will usually appear on the screen before the music being played reaches that point.

If, after a piece has finished playing, the message

WARNING: CLOCK TICKS LOST = nnnnn

appears, this indicates that the speed at which the music was played was so high that the composer system was not able to interpret the files fast enough to keep up with the rate at which the music was being played. "nnnnn" indicates how many B units were spent waiting for the composer to catch up.

The CLOCK TICKS LOST warning means that the CMI has skipped some notes rather than fall behind. Usually by this stage however, the music is playing so fast that it is unintelligible anyway.

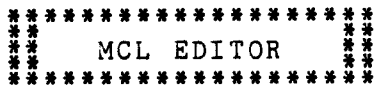
This warning should never occur in normal operation, but may appear if the X function is used to turn off the music in the middle of a piece or if a lot of WARNING messages were being generated.

Other aids to debugging your music program are:

HALT and SINGLE STEP
TIME FUNCTION
X FUNCTION
PROMPT FUNCTION
DIV COMMAND

The user should become familiar with these facilities to enable rapid music generation.

SECTION 5 - MCL EDITOR



All music and comments typed in by you will be set out in files consisting of consecutive lines inside the EDITOR. The EDITOR allows you to edit the MCL files

PIECE (.PC)
PART (.PT)
SEQUENCE (.SS)

For details of exactly what goes into each type of file, see the relevant parts of SECTION 2A - SEQUENCE FILES and
SECTION 2B - PART and PIECE FILES.

When you are in the EDITOR the display is headed

FAIRLIGHT COMPOSER EDITOR - LEVEL 1.5 - READY FILENAME=

EXAMPLE
Type NEW TUNE.SS<return>

The CMI will respond by changing the display to the following

FAIRLIGHT COMPOSER EDITOR - LEVEL 1.5 - READY FILENAME=TUNE.SS

This is in the EDIT mode. You will notice no line numbers appear yet.

Press the <return> key a few times. Consecutive line numbers should now start appearing. The <return> key effectively inserts lines and line numbers.

All lines are allocated automatically by the EDITOR and have a number for identification between 0000 and 9999. Consecutive line numbers generally increase although they may have the same line number (not a good idea - see the RESEQUENCE instruction).

If a line is created which has the same number as the one before or after then the warning message

LINE NUMBER CONFLICT

will be given and it is recommended that the RESEQUENCE command be used to avoid confusion later. Line numbers never decrease.

To leave the EDITOR and return to Page C type

E<escape>

STATUS LINE

Consistent with other pages, the EDITOR has a STATUS line which is a horizontal band at the top of the screen. Its function is to display messages from the CMI to you. If you make an incorrect entry from the typewriter keyboard an ERROR MESSAGE will be printed to notify you. Whenever your input does not result in the expected action, check the STATUS LINE, and you may find a message waiting.

COMMAND LINE

The line directly below the status line is known as the COMMAND LINE. Anything typed by you will appear on this line so that you may see what you have typed before putting it in the file.

A maximum of seventy nine characters per line is permitted, not including line numbers.

The RUBOUT key will successively erase characters from the command line.

The CLEAR key will clear the whole command line.

The <control-B> combination will "restore" the previous contents of the command line. Every time a command is executed the command line is saved. If the <control-B> combination is typed, characters which have been typed in since the last command will overwrite the previous contents of the command line, but the rest of the command is not changed. This feature is very useful if, for example, an error was made right at the start of a command. The first part of the line can be retyped, and then <control-B> is typed to RESTORE the rest of the line, thus saving a lot of typing.

CURSOR

The CURSOR indicates the current character position. It is the focus point where text can be inserted and deleted.

Initially the CURSOR will be at the home position (beginning) of the first line of the file.

When at the home position the CURSOR is pointing at the line number and this is indicated by the line number being displayed as black-on-white (inverted) on the screen. In this position lines of text can be inserted sequentially into the file in front of the current line.

When the CURSOR is moved out of the home position by use of the right or left arrow commands, or other cursor moving commands, then the character being referenced is indicated by a break in the window or rectangle surrounding the current line.

EXAMPLE

```
0001| A2V12,5 R,2 B#3
```

In this example, the CURSOR has been positioned to line number 0001 and around the "5" of A2V15,5.

Three keys will move text from the command line to the file. The position of the CURSOR will affect where in the line text is inserted.

The three keys are:

1) <return> Text is moved from the command line to the file. A new line number is generated. If the CURSOR is not at home, that is in the left-most position, then the rest of the line after the CURSOR will be allocated to a new line.

EXAMPLE

```
0001| A2V12,5 R,2 B#3
```

If <return> is now typed the result is

```
0001| A2V12,5
0002|R,2 B#3
```

2) <add> Text is moved from the command line and added to the current line, providing that this does not make the line longer than 79 characters. If the CURSOR is not at home then text is added to the immediate left of existing text.

EXAMPLE

```
0001| A2V12,5 R,2 B#3
```

If (C2 E2),3<add> is typed the result is

```
0001| A2V12,5(C2 E2),3 R,2 B#3
```

The <sub> key subtracts characters from the right of the line.

3) <set> Text is moved from the command line, overwriting any thing already there. If the line is empty (cursor at home) then <set> behaves the same as <add>.

EXAMPLE

```
0001| A2V12,5 R,2 B#3
```

If 7<set> is typed the result is

```
0001| A2V12,7 R,2 B#3
```

MOVING IN THE EDITOR

In order to insert typing into MCL files and to make changes you have to be able to navigate through the file. This is done by moving the CURSOR around much like on other CMI pages, thereby altering the place at where information is inserted and deleted.

At any time, the CURSOR is positioned on a line, either "at home" at the very left-hand side of a line or somewhere along a line, signified by a gap in the rectangle surrounding a line.

There are various ways to move the CURSOR around in the editor.

ARROWS

The four arrow keys, left, right, up, down and the HOME key are located on the right-hand side of the alpha-numeric keyboard. Numbers can be typed before hitting the arrow keys thus...

EXAMPLE	3↓	Move the CURSOR three lines down.
EXAMPLE	2→	Move the CURSOR two positions right.

However if there is more than a single number, or extra characters in the input line then the arrow commands will always take a default value of 1.

EXAMPLE	3A↓	Move the CURSOR one line down.

INFINITY

The "&" character is used to signify infinity or the maximum number possible. In this context, "&" is equivalent to the number "9999" as that is the maximum number of lines possible.

Thus...

&↑	Move the CURSOR 9999 lines up that is move to the BEGINNING of the file (line 0001).
&↓	Move the CURSOR 9999 lines down that is move to the END of the file.
&→	Move the CURSOR to the end of the current line.
&←	Move the CURSOR to the start of the current line. This is equivalent to pressing the HOME key.
HOME	Move the CURSOR to the start of the current line. (Reset the CURSOR).

LOCATE LINE

Instruction.....nL<escape>

between 1 and 9999, Default 1 — Locate

This command will allow you to go straight to a specified line number.

EXAMPLE 113L<escape>
Move the CURSOR to line number 113. If there is no line 113 then the CURSOR will be moved to the next higher line number.

EXAMPLE L<escape>
Move the CURSOR to the BEGINNING of the file (line 0000).
So L<escape> has the same effect as &↑.

EXAMPLE &L<escape>
Move the CURSOR to the END of the file (line 9999).
So L<escape> has the same effect as 9999L<escape> and &↓.

FIND STRING

Instruction.....<n>F<string><escape>

between 1 and 9999, Default 1 — FIND

NB: From cursor position onwards.

This command will search right for a string of characters.

A STRING means any group of alpha-numeric characters from A to Z and 1 to 9, as well as spaces.

Each time the string is found the cursor is repositioned to the START of the string and it will not be moved until another instance is found.

EXAMPLE FB=100<escape>

Find B=100. Note that this command is terminated with <escape> rather than <return>

EXAMPLE 22FA2V15,4<escape>

Find the 22nd occurrence of A2V15,4.

CHANGE STRING

Associated with the FIND command is the CHANGE command

```
Instruction..................nC<string1>\<string2><escape>
                            /  \        |    |         \
              between     CHANGE        |  <shift L>    \
             1 and 9999        original       new
              Default 1        string         string
```

This command will search the rest of file from current position and change <n> occurrences of <string 1> to <string 2>. Each time a string is changed the CURSOR is repositioned to the start of the string.

<n> defaults to 1
<string 1> must have at least 1 character.
<string 2> may be null, that is nothing.
The delimiter "\" MUST occur exactly once in the command line.
The backslash character \ is obtained by typing <shift L> on the typewriter keyboard.

This is very useful in changing many instances of the same thing.

EXAMPLE 10CA#3,2\A3,2<escape>
Change the following 10 occurrences of A#3,2 to A3,2.

EXAMPLE &C \<escape>
Change 9999 blanks to nulls. In other words take out 9999 blank spaces and replace them with nothing (COMPRESS the file).

CHANGE PREVIOUS STRING

Instruction..................\<string><escape>

This command enables a previously found or changed string to be changed again without retyping it. Whenever a string is found by the FIND or CHANGE commands, its position and length are saved until the CURSOR is moved again. This is referred to as the "current string" and it may be of any length including zero after a CHANGE with null <string 2>.

If there is no current string an error message will result.

EXAMPLE

```
CF A C\C E G<escape>   Change  F A C  to  C E G.
\C E A<escape>         Change C E G to C E A.
```

MUSIC KEYBOARD INPUT

<+ or ->M<sharp or flat><default octave><esc>

This command inputs notes from the music keyboard.
It does not input timing values or Key Velocity.

When you first edit a file the music keyboard is disabled.
After the M command is given, SWITCH 1 on the music keyboard will turn the music keyboard note inputting ON and OFF.
When turned on, key depressions on the music keyboard will cause the MCL representation of the note to be inserted into the file being edited at the current CURSOR position.
The exact format of the note inputted is determined by options specified in the M command, but is in the general format

:<note name><accidental (if any)><octave (if any)>

Note that no timing is taken from the notes entered on the music keyboard.

EXAMPLE

+M#0=4<escape> Turn the music keyboard ON. All black notes on the music keyboard will appear as sharps. All octaves will be relative to octave 4 on the music keyboard.

A typical display after playing the music keyboard might be:

A-:B:C#-3

EXAMPLE

+M<escape> Turn the music keyboard note inputting ON. Attach octave numbers to each note.

A typical display after playing the music keyboard might be:

A3:B4:C1

EXAMPLE

-M<escape> Turn the music keyboard note inputting OFF. The keyboard can be temporarily turned OFF by pressing and releasing SWITCH 1. Music keyboard is automatically turned OFF when you leave the Editor.

CHORDS

As well as the above, SWITCH 2 on the music keyboard is designated as the "CHORD" switch, to make entry of chords easier. When SWITCH 2 is pressed an open bracket (is written into the input line, and when it is released a close bracket) is put into the file.
Therefore all keystrokes entered while SWITCH 2 is depressed will be considered by the MCL Editor as a chord.

DELETE CHARACTERS

<n><sub>

Delete <n> characters starting at the current CURSOR position. <n> defaults to 1. The CURSOR remains at the same position. This is a character command and will not delete lines. Caution must be exercised of course with deleting.

This command differs from the <rubout> key which deletes characters from the COMMAND line.

If <n> is negative then characters will be deleted backwards (left) starting with the previous character. The line number will not be deleted unless CURSOR is at home.

EXAMPLE

<sub>	delete one character at the current CURSOR position.
6<sub>	delete six characters starting from the current CURSOR position.
-3<sub>	delete three characters left of the current CURSOR position.
&<sub>	delete the rest of the current line. & is infinity.
-&<sub>	delete the line left of the current CURSOR position.

DELETE LINES

<n>,L<sub>

Delete <n> lines starting from the current CURSOR position. If <n> is omitted then the comma must also be omitted and a value of 1 is assumed.

If CURSOR is at home then the current line is deleted and the following line becomes the new current line. This process is repeated <n> times or until the end of the file is reached.

If the CURSOR is not at home then the right part of the current line is deleted, the following line number is removed, and the following line joined to the left part of the current line, provided that this does not result in too long a line, that is more than 79 characters. If this is the case then an error message is given and nothing is deleted. The next line becomes the current line.

It will be seen that the L<sub> command will reverse a <return> (insert text) command if the CURSOR is repositioned to the start of the text that was inserted.

The not-at-home <n>,L<sub> command will repeat <n> times or until the end of file, or until a deletion cannot be done due to a line being too long, that is longer than 79 characters.

If the number <n> specified is preceded by a negative "-" sign then lines will be deleted backwards from the current CURSOR position. Thus if the CURSOR is at home then -L<sub> will delete the previous line.

If the CURSOR is not at home then the left-hand part of the current line and the current line will be deleted, and the right part of the current line will be concatenated (joined) to the previous line, provided that the resulting line is not too long, that is longer than 79 characters.

The CURSOR remains at the same position after a delete line operation.

EXAMPLES

<sub>	delete the current line (same as 1,L<sub>).
6,L<sub>	delete six lines starting from the current CURSOR position.
-3,L<sub>	delete three lines left of the CURSOR position.
&,L<sub>	delete the rest of the file starting from the current CURSOR position. & is infinity. A warning that the end of the file has been reached will be given.
-&,L<sub>	delete the file left of the current CURSOR position.

RESEQUENCE LINE NUMBERS

<n1>,<n2>S<escape>

Resequence line numbers. This command is used to make room in the line numbers for more lines to be inserted.

It is most commonly used after the warning message

WARNING LINE NUMBER CONFLICT

After this warning message appears in the STATUS line (the top most line) you will notice that more than one line will have the same number. You can still continue inserting text after the conflicting lines but not between conflicting lines. It is good practice though, to RESEQUENCE line numbers.

Resequence line numbers:

<n1>,<n2>S<escape>

<n1> is the increment to be applied between sucessive line numbers. Default is 10.

<n2> is the starting value, which must not conflict with the previous line number or current line number if at home. That is to say <n2> must be greater than the number of the line at which the CURSOR is currently positioned. If <n2> is not specified, then it defaults to the previous line number (or current line number) plus the increment.

The maximum line number is 9999 and a warning will be given if this value is reached before the end of file.

If <n1> is omitted, then the comma may also be omitted.

EXAMPLE
S<escape> is the same as 10,10S<escape>

EXAMPLE

Line numbering Before		Instruction	Line numbering After
0001	)	S<escape>	0010
0001	) line		0020
0001	) number		0030
0001	) conflict		0040
0002			0050
0003			0060
0004			0070

EXAMPLE

Line numbering Before		Resequence Instruction	Line numbering After
0001	)	1,1S<escape>	0001
0001	) line		0002
0001	) number		0003
0001	) conflict		0004
0002			0005
0003			0006

EXAMPLE

Line numbering Before	Resequence Instruction	Line numbering After
0001	100,5S<escape>	0005
0002		0105
0003		0205
0004		0305
0005		0405
0006		0505

EXAMPLE

Line numbering Before	Resequence Instruction	Line numbering After
0001	2,1111S<escape>	1111
0002		1113
0003		1115
0004		1117
0005		1119
0006		1121

EXAMPLE

The quickest way to fix a line number conflict, if you are not concerned about the actual line numbering, is to do the following:

L<escape>	go to the beginning of the file (locate line 0000).
S<escape>	resequence all lines in increments of 10. (same as 10,10S<escape>)

DISPLAY TEXT

<n>D<escape>

This command is purely for display purposes only.

Display <n> lines on the screen starting at current line. After a display the screen will not be showing the current CURSOR position, and if any CURSOR dependant commands are given, the Editor will display the message

NO LINE OPEN

To re-open the current line, the command character <escape> must be typed on its own, after which the screen will be rewritten at the correct position.

If a negative number is specified before the DISPLAY command then the display will scroll backwards towards the start of the file.

The DISPLAY command can be temporarily halted by pressing the <control W> combination, (for WAIT), and resumed by pressing <control W> again. The DISPLAY command can be stopped altogether by pressing the

<control><escape>

combination known as BREAK.

EXAMPLE

25D<escape>	display the next 25 lines.
D<escape>	display all lines. (same as &D<escape>
-100D<escape>	display the previous 100 lines.
-D<escape>	display the file from the current CURSOR position to the beginning of the file. (same as -&D<escape>)

PRINT TEXT ON LINE PRINTER

<n>P<escape>

Print <n> lines on the line printer. The name of the currently open file is printed at the start of a listing and at the top of each new page. <n> defaults to 1.

Other forms of the print command are:

P/<escape> causing the printer to eject one page.

P:<text><escape> will print characters on the line printer at the current page position then advance to the next line. Thus text can printed manually by the operator from inside the Editor.

P:<escape> will skip one line.
For other printable and control characters
see SECTION 2A - SEQUENCER FILES under PROMPT.

The <n>P<escape> command is essentially the same as the PRINT <filename> command (See SECTION 3 - MCL COMMANDS) except that <n>P<escape> prints from within the Editor.

If the printer is not connected or not ready (power off) when the P command is given, the error message

PRINTER NOT READY

will be displayed and the command halted.

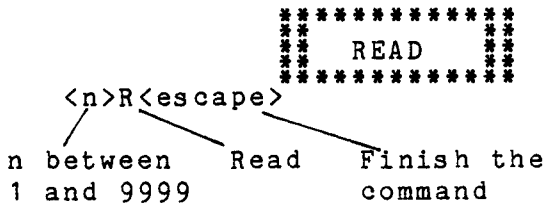
The PRINT command can be temporarily halted by pressing the <control W> combination, (for WAIT), and resumed by pressing <control W> again.

The PRINT command can be stopped altogether by pressing the

<control><escape>

combination known as BREAK.

Whenever editing music in MCL, use comments and spacing liberally. Any music score is quicker to implement or modify if well planned and well laid out. Set out your MCL score this way. It is hard to use up all the memory.



A block of music, or text, is READ, or copied into a temporary part of memory (see COPY BUFFER). The copying starts at the current cursor position. The length of the copy depends on the number inserted in front of the "R". The display will not change after this command.

EXAMPLE

5R<escape>

Five lines downwards will be read into the COPY BUFFER.

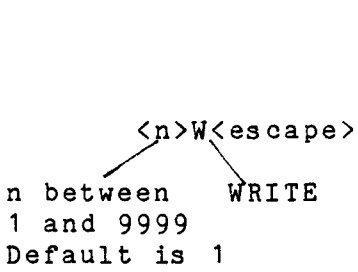
EXAMPLE

R<escape>

Read the current line into the COPY BUFFER. "n" defaults to 1.

The <n>R<escape> command has a complementary WRITE command: <n>W<escape>.

With these two commands lines, blocks or even whole files can be joined together in a "cut and paste" manner.



A block of music is WRITTEN, or copied from a temporary part of memory a number of times. The block is inserted immediately before the current cursor position. The length of the copy depends on the READ (nR<escape>) command inserted in front of the "R". The display will change showing the effect of the nW<escape> command.

EXAMPLE

5R<escape> 8W<escape>

Get the block of music, or text, previously "read" (5 lines) and "write" it 8 times into the position immediately before the cursor. This will result in 40 (5x8) additional lines.

EXAMPLE

R<escape> W<escape>

Copy the current line and insert in front of it.

If there has been no previous READ instruction, then, of course, the WRITE instruction will do nothing as there is nothing in the COPY BUFFER.
The READ/WRITE facility enables large amounts of information to be moved, concatenated, re-written slightly differently, files to be joined together and so on.
The lines of music that are READ are actually stored in a temporary area called the COPY BUFFER.

COPY BUFFER

The COPY BUFFER temporarily stores music, or text, after the READ command, <n>R<escape> has been used. Subsequent READ commands will overwrite. Temporary storage means that if you were to EXIT from the editor and play some music, or, went to another page then the COPY BUFFER would be emptied.
It is possible to open up the COPY BUFFER, change the contents, and exit from the COPY BUFFER, before actually WRITING (nW<escape>) the contents.
In this way for example, some music can be READ from one sequence, modified, and put into another sequence eliminating repetitive typing.
To see inside the copy buffer...

OPEN COPY BUFFER

B<escape>

B for buffer

If there has been no READ (nR<escape>) used to put information into the COPY BUFFER, or if you have changed pages since READING then the message

COPY BUFFER EMPTY

will be displayed on the top line of the screen (command line).
If there is something in the COPY BUFFER then the display will change and you will then see the COPY BUFFER. The filename COPY BUFFER is displayed in the top right-hand corner of the display.

The COPY BUFFER file is very similar to MCL files except of course you cannot save it, or play it. The capacity of the COPY BUFFER is around 4,000 characters. Obviously, you also cannot use the

<n>R<escape> or
<n>W<escape>

commands whilst the COPY BUFFER is displayed. That is, you cannot READ or WRITE the COPY BUFFER into itself.

OPEN or RE-OPEN

O<escape>

OPEN
original file

The display will revert back to the previous file being edited at the previous CURSOR position. Alternatively, type E<escape> as in any MCL file, to get straight back to Page C.

A variation of the OPEN command is

O<filename><escape>

which closes the current file and opens <filename>, which must exist in CMI memory eliminating the E<escape> step.

A very useful feature of the OPEN command is

O-<escape>

Open previous file, provided there has been one. The current file becomes the previous file. This allows you to skip between two files quickly by just typing O-<escape> each time as the current file becomes the previous file

NOTE

R<escape> will copy one line of music into the COPY BUFFER.
W<escape> will write the entire contents of the COPY BUFFER immediately in front of the cursor.
B<escape> will open the COPY BUFFER for inspection or modification.
O<escape> will get you back to the current MCL file being edited.
E<escape> will get you back to Page C.

A SIMPLE EXAMPLE

Reproduced below is an example of the MCL version of "Three Blind Mice". In this instance, three parts are specified which play the melody in canon using three different keyboards, and hence three different voices may be assigned to the parts.

The steps involved in generating this MCL piece would be:

1) Set up PAGE 3 with three different registers, each with a different voice, and assign keyboards 1 to 3 to registers A to C respectively.

2) Select PAGE C to enter the MCL sub-system, and use the EDITOR to create a sequence file.
The command would be: NEW MOUSE.SS<return>.

3) Once the edit has been ended,
using the E<escape> command), the message

END OF EDIT

will be displayed. The command PLAY MOUSE.SS will then play the sequence just created, and as no PART has been specified so far, it will default to keyboard 1.

4) Create the three parts desired, in this case

MOUSEA.PT
MOUSEB.PT
and MOUSEC.PT.

In order to make the MOUSEB and MOUSEC parts wait the correct time before starting, a sequence consisting of rests is introduced into their PART files.
Remember to start the line of sequence with a "
or the MCL will look for a sequence called B=24 .

5) Now create the REST8.SS sequence.

6) Create the PIECE, MICE.PC which is simply a list of the PARTS to be played. The order in which the parts are specified is of no consequence in this case as the keyboard number has been specified for each PART. Otherwise the order would dictate which part plays which keyboard.

7) Play the PIECE, using the command:

```
    PLAY MOUSE.PC
or      P MOUSE.PC
```

Note any mistakes in the music or error messages which may be displayed.

8) If your CMI is equipped with a Line Printer, you can list the piece for reference and as an aid to debugging. The command: PRINT ((MICE.PC

will give you a listing of the PIECE, PARTS and SEQUENCES like the one on the next page.

9) Use the editor to correct any mistakes.

10) Don't forget to SAVE your work on disk once you are satisfied with it. Use the command:

SAVE ((MOUSE.PC

to save the piece, parts, and sequences.

The following is a listing of the piece "THREE BLIND MICE".
Lines containing an asterisk * are comment lines only.

```
FAIRLIGHT COMPOSER - LEVEL 5.3     MICE.PC      PAGE 1

FILENAME = MICE    .PC

   0001  * THIS IS THE "THREE BLIND MICE" PIECE.
   0002  * MOUSEA.PT, MOUSEB.PT, MOUSEC.PT
   0003  * ARE PLAYED SIMULTANEOUSLY
   0004  * SPEED SET TO 8000, PROMPT IS ON
   0005  !S=8000 P=ON
   0006  MOUSEA
   0007  MOUSEB
   0008  MOUSEC

FILENAME = MOUSEA    .PT

   0001  !K=1
   0002  MOUSE
   0003  * MOUSE.SS IS PLAYED ON KEYBOARD 1

FILENAME = MOUSE    .SS

   0001  * THIS IS THE CENTRAL SEQUENCE TO BE
   0002  * PLAYED BY ALL THREE PARTS IN CANON.
   0003  <
   0004  B=24 O=3
   0005  <E D C R>2
   0006  G+ F+,1/2 F+,1/2 E R G+ F+,1/2 F+,1/2 E R,2/3
   0025  * NOW FOR THE MIDDLE BIT UP ONE OCTAVE
   0035  O=+ B=8
   0045  <G C,2 C B A B C,2 G G R>3 F
   0055  * THE LAST BAR DOWN ONE OCTAVE
   0065  O=- B=12
   0075  E D C R
   0085  * NOW REPEAT THE WHOLE LOT THREE TIMES
   0095  >3

FILENAME = MOUSEB    .PT

   0001  * SECOND PART OF THE ROUND.
   0002  * PLAYED ON KEYBOARD 2
   0003  !K=2
   0004  * REST FOR 8 BEATS
   0005  "B=24 R,8
   0006  MOUSE

FILENAME = MOUSEC    .PT

   0010  * THIRD PART OF THE ROUND.
   0020  * PLAYED ON KEYBOARD 3
   0040  !K=3
   0050  * NOW REST FOR 16 BEATS
   0060  "B=24 R,16
   0070  MOUSE
```

APPENDIX B - EXTERNAL SYNCHRONIZATION

EXTERNAL SYNCHRONIZATION

The C.M.I. is provided with a three pin CANNON connector labelled SYNC.

The function of each pin is:

PIN 1 SIGNAL GROUND
PIN 2 EXTERNAL SYNC INPUT
PIN 3 CLICK OUTPUT

CLICK OUTPUT

Using CLICK output

This is a pulse-type output which is used primarily for a beat reference for live musician accompaniment. It is also useful for debugging purposes.
The CLICK output can be connected to an audio amplifier via any single-ended line-level input. The amplitude of the CLICK is about 1 volt P-P. The CLICK can also be heard via the CMI monitor speaker output. The volume of the CLICK is controlled by the SYNC MONITOR control on the rear panel of the CMI.
The relationship between the CLICK output and the BEAT rate of the MCL piece is set by the command:

CLICK=<rate>,<intro>

where <rate> is the number of clock ticks per CLICK.
<intro> is the number of CLICKS introduction that will be heard before the piece starts playing.

Because the relationship between clock ticks and the BEAT rate of the MCL is set by the B=n statement in a sequence or part file, using the same number for B=n and CLICK=n will result in one CLICK per beat.

EXAMPLE
If the M.C.L. uses B=24, then CLICK=96,4 will result in one CLICK per 4 BEATS and 4 CLICKS (=4 bars) intro.

SYNCHRONIZING TO MULTI-TRACK TAPE

If the complete MCL piece requires more than the allowable maximum of eight parts, multi-track tape can be used to allow further parts to be laid down. In this case it is necessary to record a SYNC tone on an unused track of the tape so that the MCL can follow variations in the tape speed for perfect synchronization.
The SYNC tone should be a constant pitch derived from:

a tone generator
an audio oscillator
a synthesizer
a CMI channel

The SYNC tone should NOT be a CLICK. The CMI plays music with a resolution in the order of milliseconds and needs an AUDIO tone between 100Hz and 5000Hz for such high resolution, rather than a CLICK which is between .2Hz and 10Hz. If necessary, the CMI can provide a synchronized CLICK of any speed for external sequencers or drum machines.
The CMI detects the very beginning of the SYNC tone and uses that as its starting point.

If the pitch of the SYNC tone rises the CMI will play faster.
If the pitch of the SYNC tone falls the CMI will play slower.

The procedure is as follows:

1) The MCL will play at the speed determined by the precise frequency of the SYNC tone. To find out the correct frequency, connect the oscillator directly to Pin 2 of the SYNC connector.
Any oscillator with a variable frequency output in the range 100Hz to 5000 Hz can be used for a SYNC tone.
The shape of the waveform is irrelevant, however a smooth waveform such as a SINEWAVE or TRIANGLE wave is to be preferred over, say a SQUAREWAVE, which tends to "spill" somewhat onto other tracks of a multi-track.

2) Select EXTERNAL SYNC on Page C using the command:

SYNC=EXT

Play the MCL piece. Varying the PITCH of the oscillator will vary the replay speed of the MCL.
Make sure the CMI is getting enough level from the oscillator.
Select a suitable speed or range of speeds.

3) Connect the audio oscillator to the tape recorder input associated with the track which is to carry the SYNC tone. SYNC tracks are usually physically positioned at the other end of the record head to minimise "spill".

EXAMPLE If music is to be recorded onto tracks 1-5 of an 8-track machine, then the SYNC track should be track 8.

4) Connect the appropriate output of the tape recorder to the SYNC input of the CMI (Pin 2 of the SYNC connector). This is a single-ended (unbalanced) input, requiring a minimum level of 1 volt P-P for reliable operation.

5) Record the SYNC track BY ITSELF while monitoring the MCL. Differences in the position of the RECORD head and the PLAYBACK head on the multi-track means that if the SYNC tone and the MCL sequences are recorded simultaneously then subsequent synchronized recordings will be out of synchronization by the amount of time it takes for the tape to move from the RECORD head to the REPLAY head.
Alternatively take the SYNC tone directly from the RECORD head.
Make sure that the start of the SYNC tone is clean and is preceded by a few seconds of silence. The MCL will start playing as soon as the tone starts. It is possible to vary the speed of the piece dynamically by varying the oscillator frequency while laying down the SYNC track. For this purpose it is necessary to be MONITORING the MCL piece in EXTERNAL SYNC mode while laying down the sync track.
Let the SYNC track run for a few seconds longer than the MCL piece.

6) From now on, all MCL playing will faithfully follow this SYNC track (unless SYNC=INT is re-selected). Lay down each group of M.C.L. parts on a seperate tape track, making sure that the PLAY is executed with SYNC=EXT.

7) Remember that with CLICK=ON the number of INTRO CLICKS will make the total time of the piece longer. So either have CLICK=ON all the time or CLICK=OFF all the time whilst multi-tracking.

For equal tempo between EXTernal and INTernal SYNC:

$$\text{SPEED} = \frac{2010.5}{\text{EXT SYNC in KHZ}} \qquad \text{EXT SYNC in KHz} = \frac{2010.5}{\text{SPEED}}$$

EXAMPLE

An EXTernal SYNC tone of 1 KHz is equivalent to an INTernal SPEED of 2010.

For video synchronization the following table applies.

FOR	SET	24 frames	25 frames
microbeat calibration :	SPEED	5236	5026
1 BEAT = 1 sec (60mm) :	CLICK RATE	192	200
1 BEAT = .5 sec (120mm) :	.	96	100
1 BEAT = S seconds :	.	S x 192	S x 200
Tempo = F frames per beat :	.	F x 8	F x 8
Tempo = T BEATS per minute :	.	$\frac{T}{60}$ x 192	$\frac{T}{60}$ x 192

APPENDIX C - ERROR MESSAGES AND WARNINGS

MCL INTERPRETER MESSAGES

These messages may be generated while playing an MCL file.

CHORD MUST END WITH A RIGHT PARENTHESIS -
Chords use the (and) brackets.

DEFAULT TYPE NOT RECOGNIZED -
Default type not recognized.

GAP/HOLD TIME IS LONGER THAN NOTE -
The CMI will limit the GAP/HOLD time to the maximum possible.

GAP/HOLD TIME IS ZERO -
Usually happens when BEAT value is reduced. e.g., B=48 G=1/48 B=24 - the GAP time is now 1/48th of the B value of 24 and truncates or rounds to zero. The CMI will not stop playing. The above should correctly read B=48 G=1/48 B=24 G=1/24.

NESTING TOO DEEP IN LOOPS -
Repeats, that is anything between < and > can be nested up to six times like <<<<<< >>>>>>.

NO LOOP START FOR THIS LOOP END -
A right repeat bracket > has been found without a corresponding left repeat bracket <.

NOTE EXPECTED WITH AN ACCIDENTAL -
An accidental is by itself with no associated note.

NUMBER EXPECTED -
Number is expected usually after an "=" sign.

NUMBER MUST APPEAR AFTER AN "=" -
Usually refers to default settings.

NUMBER MUST APPEAR AFTER A "/" -
eg., B=/2 divides the note resolution by 2.

NUMBER MUST BE BETWEEN 1 AND 7 -
Octave number corresponding to music keyboard.

NUMBER MUST BE BETWEEN 1 AND 8 -
Keyboard 1 to 8 in a Part file.

NUMBER MUST BE BETWEEN 1 AND 255 -
Associated with many values.
e.g., G=1/260, R,256 are out of range.

NUMBER OF TIMES TO REPEAT MISSING -
A repeat number between 1 and 60,000 must immediately follow the right repeat bracket >.

NUMBER OUT OF RANGE -
A Control has been set to a value less than 0 or more than 127. The CMI will limit to the maximum or minimum possible.

UNRECOGNIZED ITEM -
The character is not valid. The sequence containing the unrecognized item will stop playing.

Messages encountered inside the Editor.

COPY BUFFER ALREADY OPEN -
The B<ESCAPE> command has already been used.

COPY BUFFER EMPTY -
Nothing in the COPY BUFFER.

COPY BUFFER FULL - READ ABORTED -
Trying to READ too much data into the COPY BUFFER.

END OF TEXT REACHED -
Just a reminder.

<filename> ALREADY EXISTS -
Occurs if you create, rename, or copy a file while a file with the same name exists in memory.

<filename> - END OF EDIT -
This message occurs when you actually leave the Editor.

INVALID COMMAND -
Incorrect use of a command terminated with <ESCAPE>.

LINE MAY NOT EXCEED 79 CHARACTERS -
The current line exceeds or will exceed 79 characters.

LINE NUMBER CONFLICT -
More than one line has the same number.
Fix with the RESEQUENCE command.

NO CURRENT STRING -
Occurs with the CHANGE PREVIOUSLY FOUND STRING.

NO LINE OPEN - HIT <ESC> TO RE-OPEN -
Occurs with the DISPLAY command.

NO PREVIOUS FILENAME -
Occurs with the OPEN command.

NUMBER OUT OF RANGE -
Number is probably less than zero or greater than 255.

PRINTER NOT READY -
Printer is either
1) not plugged into the CMI.
2) not connected to the power point.
3) or has filled up its own buffer and cannot take any more data from the CMI for the moment. This condition only lasts while the Printer buffer is full.

R/W INVALID WHILE COPY BUFFER OPEN -
Cannot copy the COPY BUFFER into itself.

STRING NOT FOUND -
A warning resulting from using the FIND or CHANGE commands.

WORKSPACE FULL -
All of the MCL memory (15,360 bytes) has been used.

APPENDIX D - SHORTCUTS

Use comments and spacing to make sequence files easier to follow.
Establish your own conventions e.g., one equal bar per line.
Make the important things stand out e.g., on a separate line.

Use the MUSIC KEYBOARD INPUT facility for putting notes and

chords directly into sequence files.
See SECTION 5 - MCL Editor Page 41.

Use DEFAULTS wherever possible to reduce typing effort, and conserve memory.
See SECTION 2A - SEQUENCER Files Page 6.

Use repeats wherever possible.
See SECTION 2A - SEQUENCER Files Page 9.

Use transposition. See SECTION 2A - SEQUENCER Files Page 9.

Use the READ/WRITE command combination for moving bulk text around.
See SECTION 5 - MCL Editor Page 47.

Develop one sequence at a time correctly before starting another.

Use the debugging aids extensively especially the TIME function (see Page 30) and single stepping (see Page 13).

Do experiments to clarify functions or commands. Understand why something performs a certain way. Make deliberate mistakes to see what happens. You cannot harm the CMI by typing anything in.

Become familiar with all CMI facilities and abilities. You may discover a new way of doing something or an instruction which saves you much time.

INDEX Page

accidentals 8,24
accuracy 3
add key 37
appendix A - simple example 50
listing 52
appendix B - external synchronization 53
appendix C - error messages and warnings 56
appendix D - shortcuts 58
attack 10
arrow keys 38

backslash (shift L) 40
beat 7
break 45,46

change string 40
chords 1,41
clear command 23,32
clear key 36
click 17,29,53,54
clock ticks 26,30,34
command line 36
commands 18
comments 6,7
com(pile) command 24
compress a file 40
conductor 1
controls and switches 11
copy buffer 47,48
copy command 22
current string 40
cursor 36,38,42
cut and paste 47

damping 11
debugging aids 33
defaults 6,7
delete characters 42
delete lines 42
dir(ectory) command 23
display text 45
div(ide) command 25
double quote character 14,15,31

edit command 19
editor 35
effects 11
errors 33,34,56
external synchronization 3,28,53,54

file management 6
file structure 6
find string 39,40
flanging 3
flat 8,24
free memory 3,32

gap 8
glissando 11

halt (wait) 13
hold 9
home key 38

infinity 38
internal synchronization 3,28,53,54

keyboard control 16
key signature 8,24
key velocity 11

level 11
line numbering 35,43,44
listing (of three blind mice)..... 52
load command 20
locate a line 39

maximum characters per line 36
maximum number of files 1
MCL capacity 3
MCL commands 18
MCL editor 35
MCL memory 3
MCL version number 4

memory 3,32
metronome 3,29,53,54
moving in the editor 38
multi-tracking 53,54,55
music keyboard input 41

name command 22
natural 8,24
new command 19
new composition 5
no current string 40

octave 7,33
open command 49
open copy buffer 48

part file 16
pause (wait) 13
piece file 17,35
play command 21
portamento 11
previous string 40
principle of operation 4
print command 23,46
printable character 14
prompt function 14,17,32

Q (query) command 27
quote (double quote) feature 14,15,16,31

read command 47
relative specification 10
rename (name) 22
re-open 49
repeats 1,9
resequence line numbers 44
reset 27
restore 36
return key 37
rub-out key 36
run-time errors 33,56

save command 21
section 1 - the MCL concept 1
section 2A - SEQUENCER files 7
section 2B- PART and PIECE files 16
section 3 - MCL commands 18
section 4 - debugging aids 33
section 5 - MCL editor 35
sequence files 7
set key 37
sharp 8,24
short cuts 58
single stepping 13
slur 10
speed 17,28,29
status line 36
string 39,40
structure 1
stop play 22,45,46
sub key 42
switches 11
sync tone 53,54
synchronization 3,28,53,54
system queries 27
system variables 26

three blind mice 50
time 26,30
time resolution 3
tone 53
transposition 9
tree (structure) 1,2

unprintable character 14

velocity 11,33
version number 4
vibrato 11
video synchronization 55

wait function 13,17,32,45
warn function 32,33
warning messages 32,33,56
write command 47

X function 25,30,32