Documentation Développeurs
Apple Computer France 1987

Vol. 2
Fas. 13

Document développeur numéro 91

Note Sequencer

type d'upgrade de ce ducument : 7
1 Documentation de première catégorie inchangée
2 Documentation de deuxième catégorie mise à jour
3 Documentation de deuxième catégorie inchangée
4 Mise à jour payante de la documentation de première catégorie
5 Mise à jour gratuite de la documentation de première catégorie
6 Nouveautés payantes non vitales
7 Nouveautés gratuites et vitales

Taille : 15 page(s) environ

Domaine : Tool 26

VERSION : Second Release
DATE : 2.02.87

2 February, 1987

The First Whole Sequencer ERS

Composed and Performed by John Worthington
based on an theme created by Geoff Brown

Obligatory Quote:

"Music is feeling then, not sound."
Wallace Stevens, *Peter Quince at the Clavier*

Modification History

30 Jan 87	First Release
2 Feb 87	Information on timing and rests added.

Note: This is a DRAFT representing work in progress. No promise is made that the actual tool, when completed will contain all of the calls described herein or that the function of the calls will remain as documented.

You are invited to send comments to:

John Worthington
Apple Computer, Inc.
20525 Mariani Ave, MS 27/AL
Cupertino, CA. 95014

AppleLink: WORTHINGTO1

Who Should Use This Tool?

The Sequencer is designed to be used by programmers who want an easy way of including music in their applications. The Sequencer allows a piece of music to be played asynchronously in the background of an application. Additionally, the Sequencer allows music to be represented in a fairly compact form.

What's a Sequencer?

The Sequencer is a cross between a "player piano" and a language interpreter. Sequences consist of a series of commands that tell the computer which notes to play and when. In addition the Sequencer provides some control structures for conditional branching and looping within a sequence.

The Sequencer toolkit only supports playback. Creating a new sequence will require an external utility of some type. Utilities for creating a new sequence from MIDI data will be provided.

The Sequencer is designed to run on an interrupt basis in the background of an application. For applications where critical timing is required, there is a mode of operation that allows the application to control sequence playing. This might be necessary if precise synchronization with graphics is required.

Interrupts must be enabled when using the Note Synthesizer and/or the Sequencer. Anything that disables interrupts such as reading the disk drives will disrupt the sound being played.

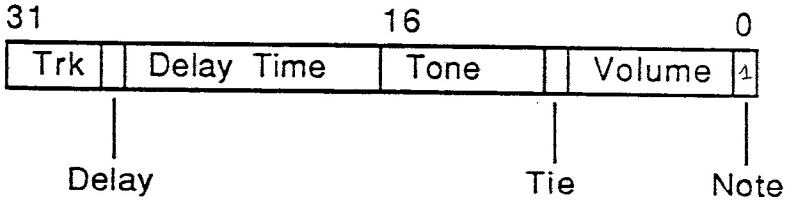
Patterns:

A pattern is a series of sequence items and the timing relationships between the items. A sequence item is a 32-bit command that describes an action to be performed by the Sequencer. These commands directly affect the sound that is being played. In general there are two types of commands: Note Commands and Control Commands. Note Commands turn notes off and on. Control Commands

are used to specify such things as pitch bend, program change, and sequence looping.

Note Command:

Note Commands are used to start notes playing or to stop notes that are currently playing. The general format of a note command is displayed below.



Bit 0 is always set for note commands. It is always cleared for control commands.

Bits 1-7 specify the note volume. This corresponds to MIDI velocity. If the volume is 0 then the player will release any note currently playing with the tone and track specified in the command (Note Off command).

Bit 8 is the tie bit. It is set if the note described is an extension to the duration of a previously played note. This allows durations longer than 2047 ticks to be used.

Bits 9-15 specify the semitone to be played. The range is 0 to 127 with a value of 60 corresponding to middle C_3 (261.6 Hz). [$f_{a_3}=440$]

Bits 16-26 specify the duration in timer ticks (0-2047). If the duration is 0 then the note will continue sounding until the player finds a matching "note off" command. That is, until the player encounters a command to play the same note with the volume = 0.

Bit 27 instructs the player to wait the duration of this command before executing the next command. If this bit is cleared then the note will continue to sound until a matching "note off" is found for it.

Bits 28-31 Specify the track number. The track number is used as an index to assign instruments and handlers for the note.

Note commands can be used in two ways. Matching pairs of "note on" and "note off" commands can be used in a MIDI-like manner to simulate pressing and releasing keys on a keyboard. The values for the semitone and volume fields of the note command are as specified by MIDI with one exception. A note command with a semitone of 0 is interpreted as a "rest".

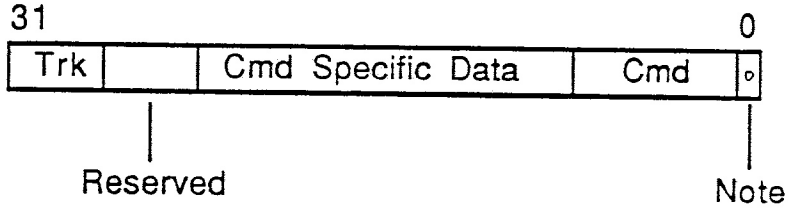
Rests are used primarily to provide timing information to the sequencer. Traditionally rests are used as a "note" of silence. The sequencer supports their use in this way. Additionally a rest can be used to make the sequencer pause for a specified amount of time. This would occur when playing performance data (ie. note on/note off pairs). Since the sequencer always plays sequence items at the specified rate, there needs to be something taking up the amount of time between the note on and note off commands. This is done with a rest. Rests almost always have a duration and the delay bit set. If the sequence is create using the SeqMaker (discussed at the end) then this is handled automatically for you.

If the duration bit is set then "note off" commands are not required. This results in a space savings of about 45% in typical sequences. Although this representation is more compact, it doesn't lend itself to real time data as does the MIDI-style format. SeqMaker will compact a MIDI sequence into duration format.

If the delay bit is set then the sequencer will not perform any other action until that event is completed. An example would be making the sequencer hold a chord using the duration format. The first and second notes are played with the delay bit cleared and a certain duration. The third note is played with the a duration and the delay bit set. The sequencer will not act on any additional sequence items until the third note has completed playing. If the duration for the first two notes is up at this time will also be silenced.

Control Commands:

In general, control commands look like the example below. The actual content of many of the fields is command specific.



Bit 0 is always cleared for non-note commands.

Bits 1-7 specify a command number.

Bits 8-24 contain data specific to the command

Bits 24-27 are reserved for future use by Apple. Unless otherwise stated, they should be cleared.

Bits 28-31 Specify the track number. The track number is used as an index to assign instruments and handlers for the note.

PatternEnd Command:

All patterns must end with this command. It tells the Sequencer that the pattern is complete and it is ok to go on to the next item in the phrase.

Bit 0 is cleared.

Bits 1-7 Command number = 1111111

Bits 8-24 are ignored.

Bits 24-27 are reserved for future use by Apple. Unless otherwise stated, they should be cleared.

Bits 28-31 Track number is ignored.

PitchBend Command:

PitchBend is used to raise or lower the pitch of a note. As part of each instrument definition is a variable that tells the note synthesizer the maximum number of semitones that the pitch may be altered. The pitch maybe raised the maximum amount by sending a PitchBend command with a value of 127. The pitch maybe lowered the maximum amount using 0 as the command value. A value of 64 results in no pitch bend.

Bit 0 is cleared.

Bits 1-7 Command Number = 0000001

Bits 8-15 contain the amount of pitch bend (64 = no pitch bend)

Bits 16-27 are ignored.

Bits 28-31 specify the track number.

Program Change Command:

This command allows the instrument being used by a particular track to be changed from within the sequence. The instrument selected must already be in the instrument table. New instruments cannot be added to the table from the sequencer.

Bit 0 is cleared.

Bits 1-7 Command number = 0000010

Bits 8-15 contains an index to the new instrument

Bits 16-27 are ignored.

Bits 28-31 specify the track number.

All Notes Off Command:

This command will turn off all notes being played by a particular track or all tracks depending on the value of the data field. It is more compact than sending individual note off commands when a large number of notes must be turned off. If the note on command specified a duration the notes will not be shut off by a note off command, but will be turned off by this command.

Bit 0 is cleared.

Bits 1-7 Command number = 0000011

Bits 8-15 contains $FF for all track, 0 for specific track.

Bits 16-27 are ignored.

Bits 28-31 specify the track number.

If-Then Command:

The If-Then command allows a pattern to skip ahead (or behind) depending on the number of times the If-Then command has been played.

Bit 0 is cleared.

Bits 1-7 Command number = 0000100

Bits 8-15 contain the number of times before the branch is taken.

Bits 16-23 contain the number of sequence items to skip forward (backward if negative).

Bits 24-27 are ignored.

Bits 28-31 specify the track number.

Jump Command:

The Jump command allows control to be unconditionally transferred to another part of the pattern. Control can be passed forward or backward.

Bit 0 is cleared.

Bits 1-7 Command number = 0000101

Bits 8-23 contain the number of sequence items to jump forwards (backwards if negative).

Bits 24-27 are ignored.

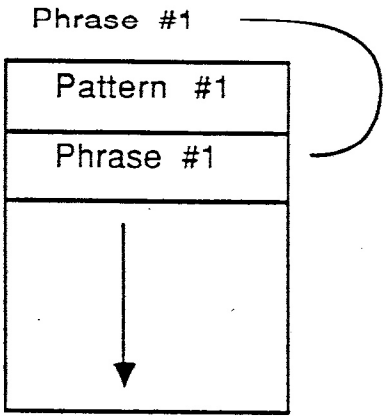
Bits 28-31 Specify the track number.

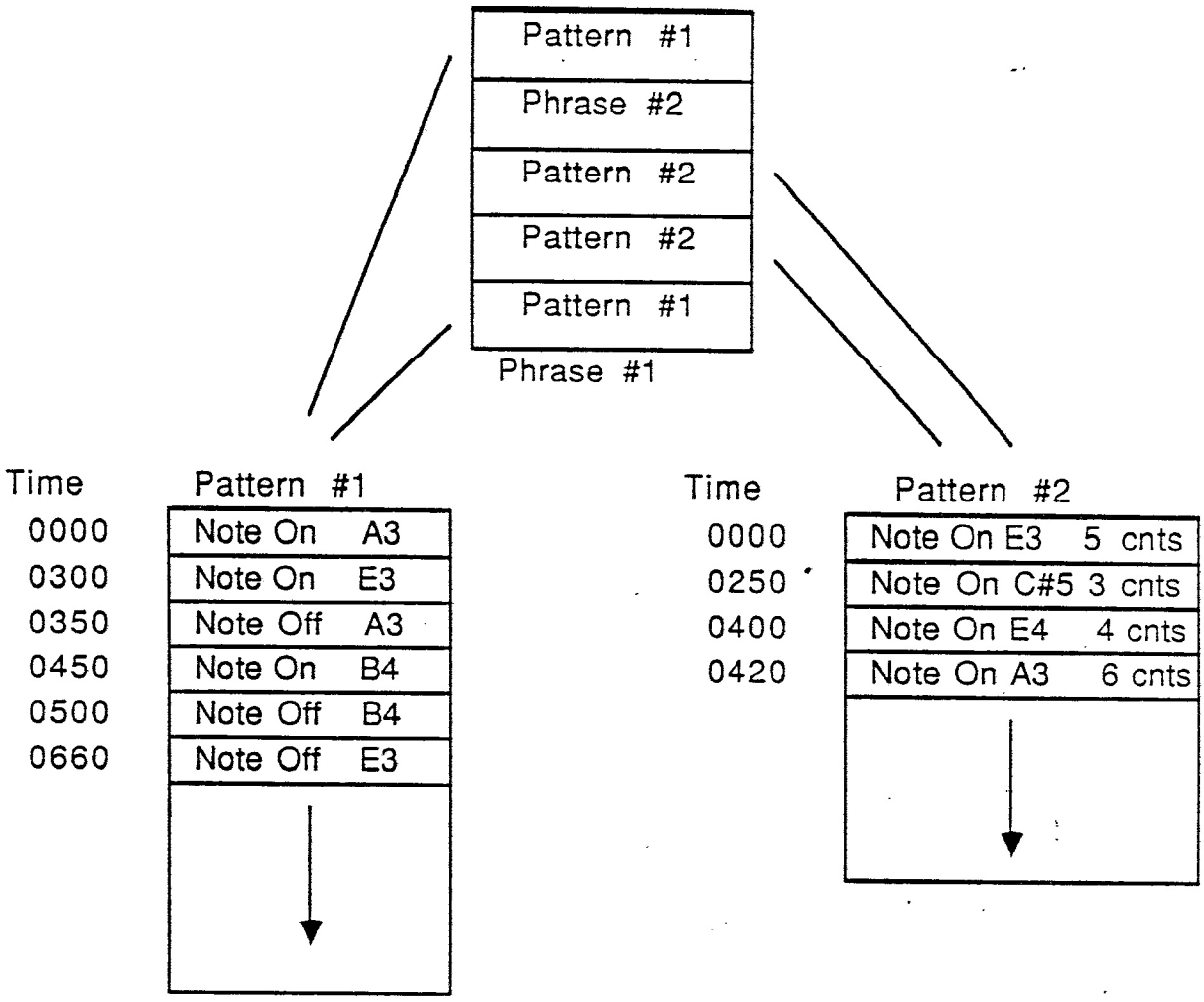
Phrases:

A phrase is composed of handles to patterns and other phrases. The term **sequence** is usually used to refer to the top level phrase. New phrases are created using the memory manager to allocate space, and then using the Sequencer call to add items to the phrase. The last item in a phrase should be the tag $FFFFFFFF (4 bytes). This tells the player that the end of a phrase has been reached.

A sequence can also be recursive. This occurs when a phrase contains itself and is very useful for playing a repeating pattern using a minimum amount of memory. If you want the sequence to

terminate eventually, you should use the if-then command to specify the number of times you want to recurse.




Pattern #1
Phrase #2
Pattern #2
Pattern #2
Pattern #1
Phrase #1
Time
Pattern #1
0000
0300
0350
0450
0500
0660
Note On A3
Note On E3
Note Off A3
Note On B4
Note Off B4
Note Off E3
Time
Pattern #2
0000
0250
0400
0420
Note On E3 5 cnts
Note On C#5 3 cnts
Note On E4 4 cnts
Note On A3 6 cnts

The example above shows a phrase that contains 2 patterns and another phrase. Pattern 1 uses MIDI style note commands. Pattern 2 uses durations. Phrase 2 is not shown. As is shown in the example, each pattern needs only be stored only once. Tool calls are provided for creating phrases from component patterns and phrases.

The Tool Calls:

SeqBootInit **Call Number 1**

Parameters: *None*

Does nothing. All variable initialization, memory allocation, etc. is performed in the startup call.

SeqStartup **Call Number 2**

Parameters:

input	*ZeroPageLoc*	WORD
input	*ProgramID*	WORD
input	*Update Rate*	WORD
input	*Increment*	WORD

The Sequencer uses 1 page of bank zero for its direct page starting at the specified address. *ProgramID* is the ID Sequencer will use when getting memory from the Memory Manager. All memory is reserved using this ID. *Update_rate* is the update rate to be used by the Note Synthesizer. The value is specified in units of .4hz. Typical values would be:

Rate: 60 Hz use 60/.4 = 150
Rate: 100Hz use 100/.4 = 250
Rate: 200Hz use 200/.4 = 500 (default)

Low rates should be used for low overhead. Higher rates will given better timing resolution and smoother sounding envelopes.

Increment is the number of ticks to skip before incrementing the sequencer clock. This value controls the tempo of the piece being played. Small values will play the piece faster than large values. The *Increment* relates to the number of beats per minute (BPM) as follows:

Increment = (update_rate* 24)/beats per minute

SeqShutdown **Call Number 3**
Parameters: *None*

This routine frees any buffers that might have been allocated by the Sequencer. It should be called by an application before quitting.

SeqVersion **Call Number 4**
Parameters:

output	*VersionInfo*	WORD

Returns the version of the Sequencer.

SeqReset **Call Number 5**
Parameters: *None*

This routine is called by the system when a reset is performed. All notes are turned off. The song pointer is returned to the beginning and all allocated generators are released.

SeqActive **Call Number 6**
Parameters:

output	*ActiveFlag*	WORD

ActiveFlag is TRUE if the Sequencer is active and FALSE otherwise.

(reserved) **Call Number 7**

This call number is reserved for future system enhancements.

(reserved) **Call Number 8**

This call number is reserved for future system enhancements.

StartTimer **Call Number 9**
Parameters

input	*Increment*	WORD

Starts the Sequence timer. *Increment* indicates the playback tempo as defined in SeqStartup.

StopTimer **Call Number 10**
Parameters: *None*

Stops the sequence timer.

GetSeqTimer **Call Number 11**
Parameters:

output	*CurrentTime*	LONG

Returns the current value of the sequence timer. Note that if the timer has not been stopped, the result is probably wrong.

AddItem **Call Number 12**
Parameters:

input	*Pattern*	HANDLE
input	*Phrase*	HANDLE

Pattern is a handle to a series of sequence items. The pattern is added to the end of the phrase pointed to by *Phrase*.

PopItem **Call Number** 13

Parameters:

input	*Phrase*	HANDLE
output	*Item*	HANDLE

Removes the first item from the phrase supplied and returns a handle to it. *Item* may be a handle to an item or phrase.

ClearItem **Call Number** 14

Parameters:

input	*Phrase*	HANDLE

Removes all elements from the phrase supplied. Note that this merely resets the internal variables that defined the phrase. The amount of memory it occupies is not affected by this call.

PlaySequence **Call Number** 15

Parameters:

input	*Sequence*	HANDLE
input	*CompRoutine*	LONG
input	*AddItem*	BOOLEAN
input	*StepMode*	BOOLEAN
input	*AlwaysCall*	BOOLEAN

Will begin playing *Sequence*. The sequence will play until complete or until . At that time the address specified by *CompRoutine* (if non-zero) will be called. If another sequence is currently playing and *AddItem* is TRUE then the sequence is added to the end of the sequence currently being played. If *AlwaysCall* is TRUE then the completion routine will be called at the completion of each phrase in the sequence.

If the Sequencer is in step-mode (ie. *StepMode* is TRUE) then StepSequence must be called repeatedly to play through the sequence.

StepSequence **Call Number 16**
Parameters: *None*

Plays the next item in the current sequence. The current sequence is setup by using the PlaySequence call. Control is returned to the application when the current item has been played. If the note being played has a duration associated with it, the note is started playing and will complete asynchronously.

StopSequence **Call Number 17**
Parameters: *None*

Stops the sequence that is currently being played. If the timer is running and there are additional phrases in the sequence, the next phrase will begin playing. To stop all playback, call ClearItem first.

About SeqMaker

SeqMaker is a Macintosh utility that will take a MIDI sequence stored in Opcode's MIDI file format and convert it to a GS sequence. It performs no quantization. In the process of converting the sequence for the GS, note commands are converted to duration format.

The main advantage to SeqMaker is that it will allow a sequence to be created by playing on a MIDI keyboard, using one of the "Intelligent Instruments" written for the Mac, or using one of the composition systems such as Professional Composer or Deluxe Music Construction Set. In general the sequences created with SeqMaker will not be as small as a hand optimized sequence.

SeqMaker will be documented in a separate set of documentation. It is mentioned here to make people aware of its existence.