Bach's Endlessly Rising Canon

by Ray Cannon (ray_cannon@compuserve.com)

Sending MIDI Messages to a Sound-Card with J

This article by Martin Neitzel appeared in Vector Vol.16 No3, fascinated me, so I downloaded the latest version of J and tried it out. It all worked, but being a long-sighted APL bigot, (I do not get on well with J, as I can't tell a ":" from a ";" without a magnifying glass <grin>) I decided to convert Martin's code into Dyalog APL/W. For the remainder of this article, I will assume you have read Martin's article.

Sending MIDI Messages to a Sound-Card with Dyalog APL/W

(All the code is available on-line in a Dyalog APL/W version 9 workspace from the Vector Web site http:\www.vector.org.uk\resource.)

First the WIN32 API calls used need to be "mapped" into ⎕*NA* calls. All the MIDI related calls are in the WINMM.DLL (Windows Multi-Media), which should be located in the Windows System directory.

```
'gnd'⎕NA'I winmm.C32|midiOutGetNumDevs '
'gdc'⎕NA'I winmm.C32|midiOutGetDevCapsA
      U2 >{U2 U2 U4 T[',(▼MAXPNAMELEN),'] U2 U2 U2 U2 U4 } U2'
'open'⎕NA'U winmm.C32|midiOutOpen =U U U U U'
'close'⎕NA'U winmm.C32|midiOutClose U'
'sm'⎕NA'U winmm.C32|midiOutShortMsg U U'
```

There, all the hard work is done, now you can write your great OPUS in A flat.

How many MIDI devices are on my PC and what can they do?

To find out the number of Midi devices , use midiOutGetNumDevs, this is a niladic function and simply returns the total number of Midi devices installed on your PC.

To find out the capabilities of each device use midiOutGetDevCapsA. This takes 3 arguments, the device number (origin zero), a pointer to the MIDIOUTCAPS structure, and the size of this structure. (See MIDIOUTCAPS in appendix 1)

The function *MidiCapabilities*, does this all for you.

```
     ∇ r←MidiCapabilities;nos;⎕IO;list;gnd;chnls;chnl;qname;dnos;names;rc;ca
p1;gdc;no;name;types;MAXPNAMELEN;sz;wMid;wPid;vDriverVersion;szPname;wTechno
logy;wVoices;wNotes;wChannelMask;dwSupportwMid;wPid;vDriverVersion;szPname;w
Technology;wVoices;wNotes;wChannelMask;dwSupport
[1]     ⍝ Describe the capabilities of the Midi devices on this machine.
[2]     ⍝ Returns a 3 element nested array of <nv device number> <vtv of Midi
 device names> <nv of Midi device types>
[3]     ⎕IO←0           ⍝ I hate origin 0 but it make life easier here, just
[4]     MAXPNAMELEN←32 ⍝  max length of product name  (including NULL)
[5]     ⍝ Define ⎕NA calls
[6]     'gnd'⎕NA'I winmm.C32|midiOutGetNumDevs '
[7]     'gdc'⎕NA'I winmm.C32|midiOutGetDevCapsA U2 >{U2 U2 U4 T[',(⍕MAXPNAMEL
EN),'] U2 U2 U2 U2 U4 } U2'
[8]     ⍝Find out how many Midi output devices are connected
[9]     nos←gnd
[10]    'This Machine has ',(⍕nos),' Midi output devices connected to it. The
y are:'
[11]    sz←+/2 2 4 MAXPNAMELEN 2 2 2 2 4  ⍝ Calculate the size of the MIDIOUT
CAP structure
[12]    names←0⍴c''
[13]    types←⍳0
[14]    dnos←⍳0
[15]    chnls←0⍴c⍳0
[16]    ⍝ list of synthesizer by type
[17]    list←'' 'output port' 'generic internal synth' 'square wave internal
synth' 'FM internal synth' 'MIDI mapper'
[18]    list,←'hardware wavetable synthesizer (I think)' 'software synthesize
r (I think)'
[19]    :For no :In ⍳nos                        ⍝ Loop for each device found
[20]        ''
[21]        rc cap1←gdc no sz sz                ⍝ Read the capabilities
[22]        :If rc=0                            ⍝ Check that it worked ok
[23]                                            ⍝ Split up MIDIOUTCAP str
[24]            wMid wPid vDriverVersion szPname wTechnology wVoices wNotes
                 wChannelMask dwSupport←cap1
[25]            name←(szPname⍳>⎕AV)↑szPname     ⍝ Extract the Product name
[26]            qname←'"',name,'"'              ⍝ and put quotes round it
[27]            :Select wTechnology             ⍝ Process info by device type
[28]            :Case 1
[29]                'Midi Device:',qname
[30]                'A midi output port which requiores a MIDI instrument
                      to be connected to it. (Ignored by this WS).'
[31]            :CaseList 2 3 4 6 7
[32]                'Midi device:',qname
[33]                'Is a ',(wTechnology>list),' (type ',(⍕wTechnology),
                      ' device).'
[34]                'It has ',(⍕wVoices),' voices.'
[35]                'Maximum number of simultaneous notes it can play is:',
                      (⍕wNotes),'.'
[36]                'It support Midi chanel numbers ',
                      (chnl←((16⍴2)⊤wChannelMask)/⍳16),'.'
[37]                :Select dwSupport
[38]                :Case 1
[39]                    'It supports a mono volume control'
[40]                :Case 3
```

```
[41]                    'It supports a Stero volume control'
[42]                :Case 4
[43]                    'It supports patch caching.'
[44]                :Case 5
[45]                    'It supports mono volume control and patch caching.'
[46]                :Case 7
[47]                    'It supports Stereo volume and patch caching.'
[48]                :End
[49]               ⍝ Remember name and type of any synthesizer
[50]                names,←⊂name
[51]                types,←wTechnology
[52]                dnos,←no
[53]                chnls,←⊂chnl
[54]            :Case 5
[55]                'Midi device:',qname,' is a "MidiMapper",
                      used to remap "non-standard Midi Channels", forget it.'
[56]            :Else
[57]                'Midi device of unknown type found:',qname
[58]            :End
[59]        :End
[60]    :End
[61]    r←dnos names types chnls
```

To use a Midi Device

Midi devices need to be "opened" (midiOutOpen), "sent messages" (midiOutShortMsg) and "closed"(midiOutClose).

Opening is easy, just give midiOutOpen the (origin zero) number of the Midi device you want to use. It will return you a handle to that device. You will need this handle to send messages to it and most importantly CLOSE the device. I emphasise this point, as if you do not close it properly, you may not be able to use it again without closing down Windows. (I will cover sending messages in more detail in the next section.)

The function *Midi* demonstrates all of this.

```
     ∇ Midi;gnd;gdc;sm;cap;no;nos;rc;cap1;c;open;hand1;hand;close;bin;err;et
xt;names;types;select;channels;dur;data
[1]    ⍝ Demo of Midi ⎕NA calls
[2]    ⍝ Display Midi output devices, and get info about any synthesizers
[3]     nos names types channels←MidiCapabilities
[4]     :If 0=⍴nos
[5]         'Sorry no synthesizers found on this machine.'
[6]     :Else ⍝ Set up some cals to the Midi APIs
[7]         'open'⎕NA'U winmm.C32|midiOutOpen =U U U U U'
[8]         'close'⎕NA'U winmm.C32|midiOutClose U'
[9]         'sm'⎕NA'U winmm.C32|midiOutShortMsg U U'
[10]        select←?⍴nos                ⍝ Select a synthesizers at random
[11]        no←select⊃nos               ⍝ Get the internal number of this device
[12]        rc hand1←open 256 no 0 0 0                  ⍝ Open this device
```

```
[13]        :If rc=0                                          ⍝ If the open worked
[14]            data←BERC select⊃channels    ⍝ "Create" some Music to plsy
[15]            'Using Midi device:',(select⊃names),' to play Canon'
[16]            hand1 Play data                               ⍝ Play the "Music"
[17]            bin←close hand1                               ⍝ Close this device
[18]        :Else                                             ⍝ Else the open failed
[19]            'An error occured trying to use ',select⊃names
[20]            'err'⎕NA'U winmm.C32|midiOutGetErrorTextA U =0T U'
[21]            c←150                  ⍝ buffer for error message
[22]            rc etxt←err rc(c⍴'')c
[23]            etxt
[24]        :End
[25]    :End
```

Sending Messages to a MIDI Device

To play a sound, what do we need to tell the device? I have written a function Play, to play some sound that requires knowing:

When to action request - from 0 to n (where n is the length of a piece of string/music).

What request to action "set Instrument" "Start-note" "End-note" "Duration of note" "Other" and "QUIT"

Which channel (instrument) is it played on (See INSTRUMENTS in appendix 2)

How high a pitch - 1-127 (60=Middle C)

How long a note

How loud a volume (velocity) – 0 to 127

I pass this information to Play as a 6-column matrix, 1 row per action.

```
     ∇ hand1 Play data;show;time;bo;more;now;action;bin;tick;channel;volume;
pitch;length;fix;instr;⎕IO
[1]    ⍝ Play sound defined in <data> on midi device with handle <hand1>
[2]    ⍝ data[;1]↔ When to action request
[3]    ⍝ data[;2]↔ What request to action 'I' 'S' 'E' 'D' 'Q'
                   Instrument/Start-note/End-note/Duration QUIT
[4]    ⍝ data[;3]↔ Which channel (instrument) does it apply to
[5]    ⍝ data[;4]↔ How hight a pitch
[6]    ⍝ data[;5]↔ How long a note
[7]    ⍝ data[;6]↔ How loud a volume (verlocity)
[8]     ⎕IO←1
[9]    ⍝ Magic numbers used in midiOutShortMsg (128 144 160 176 192)
[10]   ⍝ 128-143 ⍝ stop  note on channel 0-15
[11]   ⍝ 144-159 ⍝ start note on channel 0-15
[12]   ⍝ 160-175 ⍝ ?          on channel 0-15
[13]   ⍝ 176-191 ⍝ ?          on channel 0-15
```

```
[14]  ⍝ 192-207 ⍝ Set up an instrument on channel 0-15
[15]   fix←{⎕IO←0 ⋄ 256⊥⍵} ⍝ Encode data into midiOutShortMsg required form
[16]   time←¯1          ⍝ Initialise starting point
[17]   tick←0.01        ⍝ Define a default tick
[18]   more←1           ⍝ Control flag. Is there any more data?
[19]   :While more              ⍝ Loop for next slice of time
[20]       time+←1              ⍝ Increment timer
[21]       bo←data[;1]∈time     ⍝ what to action are due NOW
[22]       :If ∨/bo             ⍝ if anything to be actioned
[23]           now←bo⌿data      ⍝ select items to actioned now
[24]           show←⍳0
[25]           now←now['EDISQ'⍳now[;2];]     ⍝ Sort into order, end a note
                                               before starting a new one.
[26]           :For action :In ↓now          ⍝ Loop for each action required
[27]               :Select action[2]         ⍝ Process for this action
[28]               :Case 'E'                 ⍝ End of a note requested
[29]                   channel pitch length volume←action[3 4 5 6]
[30]                   bin←sm hand1(fix 0 volume pitch(128+channel))
[31]               :Case 'I'                 ⍝ Set a new instrument up
[32]                   channel instr←action[3 4]
[33]     ⍝'Ch:',(⍕channel),'Instr:',⍕((⊃¨Instruments)⍳instr)⊃Instruments
[34]                   bin←sm hand1(fix 0 0 instr(192+channel))
[35]               :Case 'D'                 ⍝ Duration of "gap"
[36]                   tick←action[3]
[37]               :Case 'O'                 ⍝ Other action
[38]                   channel pitch length volume←action[3 4 5 6]
[39]                   bin←sm hand1(fix 0 volume pitch(160+channel))
[40]               :Case 'S'                 ⍝ Start a new note
[41]                   channel pitch length volume←action[3 4 5 6]
[42]                   show,←pitch volume
[43]                   bin←sm hand1(fix 0 volume pitch(144+channel))
[44]   ⍝ Now add an action to data to end this note!
[45]                   action[1 2]←(time+length)'E'  ⍝ End time, END action
[46]                   data,[1]←action       ⍝ Append to list of actions
[47]               :Else                     ⍝ Any other request...eg Quit
[48]                   more←0                ⍝ Make sure we stop looping
[49]               :EndSelect
[50]           :EndFor
[51]       :EndIf
[52]       ⎕DL tick    ⍝ Another slice of time flys by, flap flap.
[53]   :EndWhile
```

What to Send?

Since first hearing Pink Floyd's "Echoes" from Meddle and then reading about Bach's "Endlessly Rising Canon" in Douglas R. Hofstadter's "Gödel, Escher, Bach: an Eternal Golden Braid", I have always wanted to "implement" an endlessly rising sound. So here is my chance.

```
     ∇ r←BERC channels;⎕IO;notes;rot;voices;starts;i;vols;last;off;channels;
inst;voice;length;offset;when;note;volume;instruments;dur
[1]   ⍝ Returns some  data to play on supplied <channels>
```

```
          Bach Endless Rising Canon
[2]     ⍝ r[;0]←→ When to action request - from 0 to n
                  (where n is the length of a piece of string/music).
[3]     ⍝ r[;1]←→ What request to action 'I' 'S' 'E' 'D' 'O' 'Q'
                   Instrument/Start-note/End-note/Duration/Other/QUIT
[4]     ⍝ r[;2]←→ Which channel (instrument) is it played on
[5]     ⍝ r[;3]←→ How hight a pitch - 1-127   (60=Middle C)
[6]     ⍝ r[;4]←→ How long a note
[7]     ⍝ r[;5]←→ How loud a volume (verlocity)  - 0 to 127
[8]      ⎕IO←0              ⍝ I hate origin 0 but it make life easier here, just
[9]      channels←64⍴(channels≠9)/channels ⍝ Exclude precussion
[10]     dur←0.5            ⍝ set a  duration for each "tick"
[11]     length←12          ⍝ number of "tick" units to play note for
[12]    ⍝ Set up 48 different volumes from very soft to very loud
[13]     vols←⌊0.5+127×10⍟10×(1+(1+⍳24)÷2.4)÷11 ⍝ Log scale
[14]     vols←vols,⌽vols             ⍝ soft(>0) loud(=127) soft(>0) over 48 steps
[15]    ⍝ Set up some pitchs - range of notes (Middle C=60),
[16]    ⍝ one per volume   C D E F# G# A# c d e f# g# a# etc
[17]     notes←22++\(⍴vols)⍴2
[18]    ⍝ Play a lot of notes on different channels at different volumes
[19]     r←0⍴⊂6⍴0           ⍝ prototype data
[20]     r,←⊂0 'D'dur       ⍝ First set up duration
[21]     :For when :In ⍳⍴notes
[22]         r,←⊂0,'I',when,14⍝ Set a instrument on each channel
[23]         :For rot :In 6×⍳8 ⍝ 8 voices 6 notes apart
[24]             r,←⊂+when'S'((rot⌽channels)[when])((rot⌽notes)[when])1
                                ((rot⌽vols)[when])
[25]         :End
[26]     :End
[27]     last←⌈/⊃¨r
[28]     r,←⊂last,'I',0,126  ⍝ Set a instrument
[29]     r,←⊂last'D' 0.1     ⍝ Set up duration
[30]     r,←⊂(last+2)'S' 0 50 30 127 ⍝ Applause
[31]     last←40+⌈/⊃¨r       ⍝ Make sure to stop it at the end
[32]     r,←⊂last'Q'         ⍝ Quit
[33]     r←↑6↑¨r
      ∇
```

What Next

That's up to you, but I want to know how to set up my own instrument. I would like to be able to play a dog's bark on my computer. (Listen to "Seamus" on Pink Floyd's Meddle) and of course what I really want to do is produce "Cannon's Endlessly Rising Bark".

Appendix 1

MIDIOUTCAPS

The MIDIOUTCAPS structure describes the capabilities of a MIDI output device.

```
typedef struct {
    WORD      wMid;
    WORD      wPid;
    MMVERSION vDriverVersion;
    CHAR      szPname[MAXPNAMELEN];
    WORD      wTechnology;
    WORD      wVoices;
    WORD      wNotes;
    WORD      wChannelMask;
    DWORD     dwSupport;
} MIDIOUTCAPS;
```

wMid

Manufacturer identifier of the device driver for the MIDI output device.
Manufacturer identifiers are defined in Manufacturer and Product Identifiers.

wPid

Product identifier of the MIDI output device.
Product identifiers are defined in Manufacturer and Product Identifiers.

vDriverVersion

Version number of the device driver for the MIDI output device.
The high-order byte is the major version number,
and the low-order byte is the minor version number.

szPname

Product name in a null-terminated string.

wTechnology

Flags describing the type of the MIDI output device.
It can be one of the following:
MOD_MIDIPORT = 1 ; output port
The device is a MIDI hardware port.
MOD_SYNTH = 2 ; generic internal synth
The device is a synthesizer.
MOD_SQSYNTH = 3 ; square wave internal synth
The device is a square wave synthesizer.

MOD_FMSYNTH = 4 ; FM internal synth
The device is an FM synthesizer.
MOD_MAPPER = 5 ; MIDI mapper
The device is the Microsoft MIDI mapper.
MOD_WAVETABLE =??? 6
The device is a hardware Wavetable synthesizer.
MOD_SWSYNTH =??? 7
The device is a software synthesizer.

wVoices

Number of voices supported by an internal synthesizer device.
If the device is a port, this member is not meaningful and is set to 0.

wNotes

Maximum number of simultaneous notes that can be played by an internal synthesizer device.
If the device is a port, this member is not meaningful and is set to 0.

wChannelMask

Channels that an internal synthesizer device responds to,
where the least significant bit refers to channel 0 and the most significant bit to channel 15.
Port devices that transmit on all channels set this member to 0xFFFF.

dwSupport

Optional functionality supported by the device.
It can be one or more of the following:
MIDICAPS_CACHE = 0004h
Supports patch caching.
MIDICAPS_LRVOLUME = 0002h ; separate left-right volume control
Supports separate left and right volume control.
MIDICAPS_STREAM
Provides direct support for the midiStreamOut function.
MIDICAPS_VOLUME = 0001h ; supports volume control
Supports volume control.
If a device supports volume changes,
the MIDICAPS_VOLUME flag will be set for the dwSupport member.
If a device supports separate volume changes on the left and right channels,
both the MIDICAPS_VOLUME and the MIDICAPS_LRVOLUME flags will be set for this member.

Appendix 2

INSTRUMENTS

```
      ∇ r←Instruments
[1]     ⍝ Return the "Standard MIDI Patch Assignments" (eg Instrument list)
[2]     ⍝ r←→Vector of 2 element vectors of(<patch no><instrument name>)
[3]
[4]    ⍝ Infomation from:-
[5]    ⍝Platform SDK: Windows Multimedia
[6]    ⍝Standard MIDI Patch Assignments
[7]    ⍝The standard MIDI patch assignments for authoring MIDI files
[8]    ⍝are based on the MIDI Manufacturers Association specification.
[9]    ⍝Following are the standard MIDI patch assignments.
[10]
[11]   ⍝Piano
[12]    r←⊂0 'Acoustic grand piano'
[13]    r,←⊂1 'Bright acoustic piano'
[14]    r,←⊂2 'Electric grand piano'
[15]    r,←⊂3 'Honky-tonk piano'
[16]    r,←⊂4 'Rhodes piano'
[17]    r,←⊂5 'Chorused piano'
[18]    r,←⊂6 'Harpsichord '
[19]    r,←⊂7 'Clavinet'
[20]   ⍝Chromatic Percussion
[21]    r,←⊂8 'Celesta '
[22]    r,←⊂9 'Glockenspiel '
[23]    r,←⊂10 'Music box'
[24]    r,←⊂11 'Vibraphone'
[25]    r,←⊂12 'Marimba'
[26]    r,←⊂13 'Xylophone '
[27]    r,←⊂14 'Tubular bells'
[28]    r,←⊂15 'Dulcimer'
[29]   ⍝Chromatic Percussion
[30]    r,←⊂16 'Hammond organ '
[31]    r,←⊂17 'Percussive organ '
[32]    r,←⊂18 'Rock organ'
[33]    r,←⊂19 'Church organ'
[34]    r,←⊂20 'Reed organ'
[35]    r,←⊂21 'Accordion '
[36]    r,←⊂22 'Harmonica '
[37]    r,←⊂23 'Tango accordion '
[38]   ⍝Guitar
[39]    r,←⊂24 'Acoustic guitar (nylon)'
[40]    r,←⊂25 'Acoustic guitar (steel)'
[41]    r,←⊂26 'Electric guitar (jazz) '
[42]    r,←⊂27 'Electric guitar (clean)'
[43]    r,←⊂28 'Electric guitar (muted)'
[44]    r,←⊂29 'Overdriven guitar'
[45]    r,←⊂30 'Distortion guitar'
[46]    r,←⊂31 'Guitar harmonics '
[47]   ⍝Bass
[48]    r,←⊂32 'Acoustic bass '
[49]    r,←⊂33 'Electric bass (finger)'
```

```
[50]    r,←⊂34 'Electric bass (pick)'
[51]    r,←⊂35 'Fretless bass '
[52]    r,←⊂36 'Slap bass 1 '
[53]    r,←⊂37 'Slap bass 2 '
[54]    r,←⊂38 'Synth bass 1'
[55]    r,←⊂39 'Synth bass 2'
[56]   ⍝Strings
[57]    r,←⊂40 'Violin'
[58]    r,←⊂41 'Viola '
[59]    r,←⊂42 'Cello (The answer to the ultimate question)'
[60]    r,←⊂43 'Contrabass'
[61]    r,←⊂44 'Tremolo strings '
[62]    r,←⊂45 'Pizzicato strings '
[63]    r,←⊂46 'Orchestral harp '
[64]    r,←⊂47 'Timpani '
[65]   ⍝Ensemble
[66]    r,←⊂48 'String ensemble 1'
[67]    r,←⊂49 'String ensemble 2'
[68]    r,←⊂50 'Synth. strings 1 '
[69]    r,←⊂51 'Synth. strings 2 '
[70]    r,←⊂52 'Choir Aahs '
[71]    r,←⊂53 'Voice Oohs'
[72]    r,←⊂54 'Synth voice '
[73]    r,←⊂55 'Orchestra hit'
[74]   ⍝Brass '
[75]    r,←⊂56 'Trumpet'
[76]    r,←⊂57 'Trombone'
[77]    r,←⊂58 'Tuba'
[78]    r,←⊂59 'Muted trumpet '
[79]    r,←⊂60 'French horn '
[80]    r,←⊂61 'Brass section '
[81]    r,←⊂62 'Synth. brass 1'
[82]    r,←⊂63 'Synth. brass 2'
[83]   ⍝Reed '
[84]    r,←⊂64 'Soprano sax '
[85]    r,←⊂65 'Alto sax'
[86]    r,←⊂66 'Tenor sax '
[87]    r,←⊂67 'Baritone sax'
[88]    r,←⊂68 'Oboe'
[89]    r,←⊂69 'English horn'
[90]    r,←⊂70 'Bassoon '
[91]    r,←⊂71 'Clarinet'
[92]   ⍝Pipe Synth '
[93]    r,←⊂72 'Piccolo '
[94]    r,←⊂73 'Flute'
[95]    r,←⊂74 'Recorder '
[96]    r,←⊂75 'Pan flute'
[97]    r,←⊂76 'Bottle blow'
[98]    r,←⊂77 'Shakuhachi '
[99]    r,←⊂78 'Whistle'
[100]   r,←⊂79 'Ocarina'
[101]  ⍝Lead Synth'
[102]   r,←⊂80 'Lead 1 (square) '
[103]   r,←⊂81 'Lead 2 (sawtooth)'
[104]   r,←⊂82 'Lead 3 (calliope lead)'
```

```
[105]  r,←⊂83 'Lead 4 (chiff lead) '
[106]  r,←⊂84 'Lead 5 (charang)'
[107]  r,←⊂85 'Lead 6 (voice)'
[108]  r,←⊂86 'Lead 7 (fifths) '
[109]  r,←⊂87 'Lead 8 (brass + lead) '
[110] ⍝Pad '
[111]  r,←⊂88 'Pad 1 (new age)'
[112]  r,←⊂89 'Pad 2 (warm)'
[113]  r,←⊂90 'Pad 3 (polysynth) '
[114]  r,←⊂91 'Pad 4 (choir) '
[115]  r,←⊂92 'Pad 5 (bowed) '
[116]  r,←⊂93 'Pad 6 (metallic)'
[117]  r,←⊂94 'Pad 7 (halo)'
[118]  r,←⊂95 'Pad 8 (sweep) '
[119] ⍝Sound Effects '
[120]  r,←⊂120 'Guitar fret noise '
[121]  r,←⊂121 'Breath noise'
[122]  r,←⊂122 'Seashore'
[123]  r,←⊂123 'Bird tweet'
[124]  r,←⊂124 'Telephone ring'
[125]  r,←⊂125 'Helicopter'
[126]  r,←⊂126 'Applause'
[127]  r,←⊂127 'Gunshot '
```