AN INTRODUCTION TO THE NOTE SYNTHESIZER

For the Cortland Computer

Introduction

The Sound Manager Tools provide several different ways of making sound on the Cortland. The Note Synthesizer is a set of functions which together create a flexible polyphonic digital synthesizer. It can be used at the same time as the other sound synthesizers (free-form and speech) which are part of the sound tools.

The note synthesizer is designed for real-time control. There are no note durations per se; a note begins when a NoteOn command is given, and it starts ending when a NoteOff command is given (it takes time for some sounds to end). The interface is, in this way, very similar to MIDI (Musical Instrument Digital Interface), which is a standard music communication protocol. It will be very easy to create music application programs which listen to external music keyboards and play notes on the Cortland. Likewise, programs which play music on the Cortland will be able to talk to external synthesizers in a very similar language. Other programs with less demanding sound requirements will find that the note synthesizer is convenient to use and can be fairly economical with processor time. At all times the sound quality will be better than on any other computer available today.

Synthesizer Features

The note synthesizer is actually a set of software control functions for the Digital Oscillator Chip (DOC). Together they create a music synthesizer with the following characteristics:

- up to 15 voices
- 1 or 2 oscillators per voice
- capable of playing back sampled sounds with loops
- seperate waveforms for each oscillator

1 eight stage amplitude envelope generator per voice
vibrato
modulation inputs for pitchbend, volume, and vibrato depth
multi-timbral capability

Instruments

The heart of the synthesizer is in the instrument description. When a note request is made, the note synthesizer is passed a pointer to an Instrument. An instrument is a data structure which describes a sound. It gives the shape of the envelope, and says which waveforms to play and at what frequencies. The relevant data is copied out of the instrument so that the instrument structure need not be locked in memory.

The instrument contains pointers to DOC RAM; several different instruments may point to the same waveform. The waveform data must be already resident in the DOC RAM when a NoteOn call is made (unlike the free-form synthesizer, which copies data into DOC RAM on the fly). Of course, the wave data must stay in DOC RAM while a note is sounding.

Modulation

Modulation is variation in a sound's pitch, loudness or timbre during the course of a single note. The note synthesizer provides two types of modulation: pre-programmed and real-time. The pre-programmed modulations are the envelope controlling the amplitude and the vibrato. These are set up in the instrument description and happen every time that instrument is played. The other modulators are set up like the controls on a physical keyboard synthesizer. Instead of wheels and pedals, there are fixed memory locations which represent the position of the controllers for each generator. These modulate pitch, vibrato amount (like the two wheels on a standard keyboard synthesizer) and volume. The Generator Control Block (GCB) is the area which holds these variables. Sophisticated music programs can create their own real-time functions, like pitch-envelopes, and drive these inputs into the note synthesizer.

Using the Note Synthesizer

The sequence for using the note synthesizer is as follows:

Call Tool Startup
Copy your waveforms into DOC RAM
Set up you Instrument description structure
Then, for each note to be played:
  Allocate a Generator
  Call NoteOn, with a pointer to the instrument
  And later, to end that note: Call NoteOff
When finished making music call AllNotesOff to make sure.

Resource Allocation

The note synthesizer shares hardware resources with the other sound synthesizers. There are 32 oscillators in the DOC. Of these, two are reserved by the sound manager, to be used as programmable timers. The remaining 30 are allocated in pairs, called generators. The 15 generators are dynamically allocated on a priority basis. In general, a generator is allocated to play a single note, and then it is returned to the pool. This technique frees the programmer from worrying about whether one sound is done before starting another.

The other resource is DOC memory. In the current Sound Tool implementation, the 64K bytes of waveform RAM must be managed by the application program. The note synthesizer assumes that there are waveforms already in DOC RAM.

System Overhead

Since the envelopes and the low-frequency oscillator for vibrato are actually software constructs, the synthesizer will use greater amounts of the processor's time when many notes are sounding. The update rate of

these functions can be selected to strike a compromise between processor overhead and sound quality. In general each voice must be updated every 5 to 10 milliseconds to create smooth sounding envelopes. The updates will also use less processor time if either the envelope or the vibrato is turned off on a given voice.

NOTE SYNTHESIZER
FUNCTIONAL SPECIFICATION

The Note Synthesizer is a group of functions which are part of the Sound Manager Tool Set for the Cortland Computer. As such, these functions conform to the Tool Locator External Reference Specification (ERS). They provide a way of making complex musical sounds on the computer when it is equipped with the ENSONIQ Digital Oscillator Chip (DOC).

An application program will use the note synthesizer by making tool calls at the beginning and at the end of each note to be played. The first call is to allocate one of the sound generators. Then the specified DOC generator will be set up to produce sound with a NoteOn call. During the course of the note the DOC registers for that generator will be automatically updated on a regular basis to create the shape of the sound. This happens from a timer interrupt routine which is part of the note synthesizer. The end of a note, called the release, starts when a NoteOff call is made.

Additional functions provided by the note synthesizer are NSStartup, NSShutdown, AllNotesOff and DeallocGen.

Generators and GCB's

There are 32 oscillators in the DOC. Of these, 2 are reserved for use as timers. The remaining 30 are grouped into pairs, called generators. These 15 generators are allocated on a priority basis as needed.

One page of bank zero memory must be assigned to the Sound Manager Tool Set when it is started up. This area is divided into 15 blocks of 16 bytes each, which are called Generator Control Blocks (GCB). The first byte of a GCB indicates which synthesizer is using that generator (if any). The definition of the other 15 bytes depends on which synthesizer is using it. The GCB is used as a mailbox by the note synthesizer. It contains the current values of three "knobs" or controllers which may be changed by the application program. These are for pitchbend, vibrato depth, and volume. All three controls have a range of 0 to 127. After a call to NoteOn, the GCB will be set up as follows:

GCB:	SynthID	byte	= 2
	GenNum	byte	= [0..14]
	Semitone	byte	as specified in call
	Volume	byte	as specified in call
	Pitchbend	byte	64 = no bend
	VibratoDepth	byte	as specified in instrument

Priority Allocation

The use of generators and generator priority is shared by all parts of the Sound Manager Tool Set. Because the note synthesizer will, by far, use the most generators, and allocate them more often, the generator allocation is now one of the note synthesizer functions. (In the final documentation this may not be so).

A generator's priority may range from 0 to 128. When priority is zero, that means that that generator is not being used, and therefore free to be used. When it is 128, then that generator is locked and may not be stolen. Priorities between 0 and 127 are used by the note synthesizer to control

the stealing of notes.

When a generator is allocated to be used by one of the synthesizers, the generator is given a new priority. The generator allocation function will return with the lowest priority generator. When the note synthesizer uses a generator, it automatically lowers its priority when the envelope hits the sustain portion, and again when it hits the release portion. When the note stops, it returns the generator to zero priority. Other synthesizers in the system (the free-form synth) should always get a generator with a priority of 128.

Interrupt Timer

The note synthesizer will use one oscillator as a free running timer to provide the update rate for the envelopes. The default rate for this update is 10 ms (100Hz). The oscillator will be initialized at tool startup time. Whenever a note is playing, the update timer interrupt will be enabled. To minimize overhead, when no notes are playing, the timer interrupt will be turned off.

A particular application may desire to change the update rate. This can be accomplished by changing the frequency of the oscillator which is used as a timer, using the SetSoundIRQ Rate function call.

Generators used for sound production by the note synthesizer will have their interrupts disabled.

Since MIDI can generate interrupts as often as every 333 usec, the note synthesizer routines will never have interrupts disabled for longer than 250 usec. This should prevent loss of incoming MIDI data.

DOC Memory

In the current definition, there is no system function to allocate DOC memory. It is up to the application to get the needed waveforms into DOC MEM, using the WriteRamBlock function.

FUNCTION CALLS

AllocGen

Input	*RequestPriority*	WORD
Output	*GenNum*	WORD

AllocGen is a request for a sound generator. If successful, it returns a generator number, from 0 to 14. Which generator is returned is determined by the current priorities of the 15 generators. If one of the generators is free, that is, it has a priority of zero, then the first free one is returned. If none are free, then it looks for one to 'steal'. It finds the lowest priority generator. If that generator's priority is lower than, or equal to, the *RequestPriority*, then that generator is 'stolen'. If all generators are already of a higher priority, then the request fails. There is one exception; a generator with a priority of 128 is never stolen.

A fail is indicated by carry set on return, and *GenNum* is returned as -1. When successful, the generator is assigned a priority equal to *RequestPriority*.

DeallocGen

Input	*GenNum*	WORD
Output	none	

This function sets the named generator's priority to zero. It also makes sure that the oscillators are halted.

AllocGen and DeallocGen can be used to gain control of generators in the DOC for any of the synthesizer functions in the Cortland Tools, or even a user defined synthesizer function. The programmer can guarantee success in this allocation by always requesting the same priority. This means that there will never be a situation where a note will not sound because all the generators are busy. That is the simplest way to use the dynamic allocation. A more advanced use of priority would be to put higher priorities on bass notes and melody lines.

NoteOn

Input	*GenNum*	WORD
Input	*Semitone*	BYTE
Input	*Volume*	BYTE
Input	*InstrumentPtr*	LONG
Output	None	

This function initiates the sounding of a note on the specified instrument.

GenNum is a generator number from 0 to 14. The *GenNum* used in the call should usually be obtained immediately prior to the call from a call to AllocGen.
The *Semitone* is specified in MIDI standard format: a value from 0 to 127, where middle C is 60.
The *Volume* parameter is also in the range of 0 to 127, and can be treated like MIDI velocity. This volume parameter is copied into the GCB. It is used as a linear scaler of the amplitude envelope.
The *InstrumentPtr* is a pointer to an Instrument structure, which is defined below.

NoteOff

Input	*GenNum*	BYTE
Input	*Semitone*	BYTE
Output	None	

This function causes the envelope generator of the given note to go to the release stage. The release usually causes the volume to drop to zero in a short time. When the envelope reaches zero, the note will no longer be heard and the note is considered off. The generator's priority will then be set to zero, indicating that it is free.

The *GenNum* and *Semitone* should be the same ones that were specified in the corresponding NoteOn call. There are cases where the note is no longer sounding; for example, if the envelope had already dropped to zero or if the generator had been stolen to play another note. NoteOff checks to make sure that the named generator is indeed playing the named semitone.

AllNotesOff

No Inputs or Outputs

This function turns off all the notes that the note synthesizer is playing and returns them to zero priority. It will not shut down other generators, such as those used by the free-form synthesizer.

INSTRUMENT DEFINITION

An Instrument is a data structure which resides somewhere in Cortland memory. A NoteOn call must pass a pointer to an instrument.

Instrument:

PriorityIncrement	BYTE
PitchBendRange	BYTE
VibratoDepth	BYTE
VibratoSpeed	BYTE
Envelope	24 BYTES
ReleaseSegment	BYTE
Spare	3 BYTES
AWavePtr	LONG
BWavePtr	LONG
Total:	40 BYTES

PriorityIncrement is a number which will be subtacted form the generator priority when the envelope reaches the sustain segment. Whe it reaches the release segment the priority will be cut in half.

PitchbendRange is the number of semitones that the pitch will be raised when the 'pitchwheel' reaches 127 (the center value is 64). The valid values for PitchbendRange are 1, 2, and 4.

VibratoDepth is the (initial) fixed depth of vibrato, ranging from 0 to 127. Vibrato is a triangle shaped LFO modulating the pitch of both oscillators in a generator. 8 is a tasteful amount and 64 is excessive.

VibratoSpeed controls the rate of the vibrato LFO. It can be any byte value, although the range from 5 to 20 is most useful. The frequency range is linear, in .4 Hz steps. A vibrato speed of zero will turn the vibrato mechanism OFF, which saves some CPU time.

The *Envelope* is composed of up to eight linear segments. Each segment is described by a level and a slope. The level is called the breakpoint and represents linear amplitude of the sound. The slope is described by an increment which will be added or subtracted from the current level at the update rate (100 times a second). The increment is a two byte fixed point number, that is, the lower 8 bits represent a fraction. Thus when the increment is 1 it represents 1/256. In this case, the increment would have to be added 256 times (2.56 seconds) to cause the envelope level to go up by 1.

The envelope is a list of breakpoints and increments:

stage 1:	breakpoint	increment
stage 2:	breakpoint	increment
stage 3:	breakpoint	increment
stage 4:	breakpoint	increment
stage 5:	breakpoint	increment
stage 6:	breakpoint	increment
stage 7:	breakpoint	increment
stage 8:	breakpoint	increment

Increment 1 is used to go from the initial level of 0 up to the level of breakpoint 1. Increment 2 is used to go from breakpoint 1 to breakpoint 2, and so on. The sustain level of the envelope, if there is one, is created by setting the increment to 0, causing the envelope to get "stuck" on that level. The release segment of the envelope is specified by the *ReleaseSegment* parameter, which must be a number from 0 to 7. The release may take several segments to get to zero. The last breakpoint should always be zero.

AWavePtr and *BWavePtr* both point to Wavelist structures. These control the waveforms which will be used for the two oscillators, A and B, which make up one generator.

The **Wavelist** structure is a variable length array where each entry is 6 bytes long. The information is particular to the DOC; the user should refer to the DOC specification when creating instruments. Each 6 byte entry represents a waveform and contains information about the allowable pitch range of the waveform. This means that the waves can be "multi-sampled" across (an imaginary) keyboard. When a note is played, the WaveList A and B will be examined and ONE waveform will be picked out and assigned to each oscillator.

Each wave in a Wavelist has the following 6 byte format:

TopKey	byte
WaveAddress	byte
WaveSize	byte
DOCMode	byte
RelPitch	word

TopKey is the highest MIDI semitone that will be played by this waveform. The synth will examine the topkey field of each waveform until it finds one greater than or equal to the note it is trying to play. The items in the Wavelist should be in order of increasing TopKey values. The last wave in a Wavelist should have a TopKey of 127. The TopKey value is, then, the split point between waveforms.

The next three bytes will be picked up and stuffed directly into the DOC registers. The WaveAddress is the high byte of the waveform address. The WaveSize sets both the size of the wavetable and the frequency resolution. The DOC mode goes into the mode register. The interrupt enable bit will be ignored.

Briefly, some of the ways that DOCMode may be used:

Synthesizer: both oscillators, A and B, in free run mode.

Sampled, no loop: Osc A in single cycle and trigger peer mode; Osc B in single cycle and halt mode, with halt set. Osc A will complete and start Osc B, which will play to the end and stop.

Sampled, with loop: Osc A in single cycle and trigger peer mode; Osc B in free run mode, with halt set. Osc A will complete and start Osc B, which will play continuously until the note ends.

RelPitch is a two-byte word which is used to tune the waveform. This will compensate for different sample rates and waveform sizes. The high byte is in semitones, but can be a signed number. The low byte is in 1/256 semitone increments.