MTU-130

SIMPLIFIED MUSIC COMPILER/PLAYER

SIMPLIFIED NOTRAN SCORE COMPILER

SIMPLIFIED REAL-TIME MUSIC PLAYER

February, 1982 REV. A

TABLE OF CONTENTS

1. INTRODUCTION

This manual describes the the music notation compiler and sound synthesis program that is standard on every MTU-130 computer system. These two programs allow the entry and performance of music with up to four parts or "voices". The tones generated have an organ-like timbre but with an infinite variety of possible "stops" available. The waveform definitions for four different stops are provided by default plus you may specify your own in terms of harmonic amplitudes and phases. These programs are a simplified implementation of the musical capabilities of the MTU-130 sound generation hardware. Other more sophisticated music synthesis and composition programs are available for the MTU-130. Contact MTU for details.

1.1 MUSIC SYNTHESIS ON THE MTU-130

Music (and other sound) is synthesized on the MTU-130 with a hardware device called a "digital-to-analog converter" or DAC for short. Quite simply put, a DAC is capable of transforming a string of numbers representing the actual shape of sound waves into an electrical signal that will indeed create that sound wave shape when played through a loudspeaker. This is accomplished much like one would plot a smoothly varying function on a graph -- with a lot of closely spaced points. For sound of reasonable quality, at least 8,000 of these points or samples must be fed to the DAC every second. Accomplishing this requires a good deal of highly efficient machine language programming but the Synthesizer-Player program supplied with the MTU-130 includes this programming. The NOTRAN Compiler translates your instructions written in a music-oriented language into sound synthesis instructions for the Synthesizer/Player program.

Most other computer systems use a hardware "synthesizer chip" to create musical sounds. When using such chips, the tonal variety and number of simultaneous sounds that may be synthesized is fixed in hardware. Typically the only tone color available is a square wave which sounds much like a kazoo and the maximum number of simultaneous notes is 3 or 4. The digital-to-analog converter on the MTU-130 on the other hand has no such hardware imposed limitation since software actually computes the sound waveforms. In essence, the Synthesizer-Player program simulates what a synthesizer chip would do. There are software limitations to be sure but even the simplified software system described in this manual has more capability than the majority of hardware synthesizer based computers available. The more sophisticated music software packages available for the MTU-130 can do some truly amazing things.

1.2 SIMPLIFIED NOTRAN COMPILER

Communicating a musical score to a computer presents a difficult problem to the system designer because of music notation's pictorial and often ambiguous nature. One effective way to input music is through use of a music language in which the note pitches and durations are described with letters and numbers. When encoded in this way, the score can then be entered and edited with a standard text editor rather than a specialized music editor.

Over the years, many "alphanumeric" music languages have been developed for different purposes. NOTRAN (NOte TRANslation) is one such language developed by Hal Chamberlin in 1970 which is exceptionally easy to learn and read and is suitable for encoding "conventional" musical scores. The Simplified NOTRAN compiler described here accepts a subset of the NOTRAN language and translates an alphanumeric score into instructions for a companion sound synthesis program.

In operation, the user first prepares a NOTRAN score file using the MTU text editor or other text editor or perhaps a composition program. The score exists in this form as a standard text file on disk that may be displayed, printed, or edited further at will. Next, the NOTRAN Compiler program is run which asks for the name of the score file to be compiled. It will also ask for an "object file" name which will contain the detail coded sound synthesis instructions. The compiler then reads the score file, interprets the meaning of the NOTRAN statements, and writes the sound synthesis instructions onto the object file. At the same time, a printed listing showing the NOTRAN score, corresponding object data, and any detected errors may be printed. To hear the actual music, the object file just created is loaded into memory and the Synthesizer-Player program is run. This program follows the compiled synthesis instructions to produce the sounds specified by the original NOTRAN score.

1.3 SIMPLIFIED SYNTHESIZER-PLAYER

The Simplified Synthesizer-Player is a program that interprets compactly coded synthesis instructions stored in memory and efficiently computes the corresponding sound waveform points. As each point is computed, it is written into a register in the 8-bit audio DAC that is standard on the MTU-130. These points are computed and written at the rate of 8,700 points per second giving a frequency response from 20 to about 3,500Hz. This is certainly not hi-fi but is comparable to A-M radio reception through a typical receiver. Note that an external speaker or amplifier may be plugged into the MTU-130 to give much better sound quality (particularly the bass) than what the built-in speaker provides.

The Simplified Synthesizer-Player program simulates a 4 voice sound synthesizer. Each voice may have a different waveform (tone color) and there may be up to 16 waveforms "on-line" in memory to choose from. If the desired waveform is not in memory when needed, it may be computed from harmonic specifications in less than a second. Tone envelopes are always rectangular in shape (off-on-off) so as a result, the tones are generally "organ-like". The more sophisticated synthesizer-player programs used by other music products overcome many of the limitations of the Simplified Synthesizer-Player described here but also require more care to use properly.

1.4 COMPILING AND PLAYING THE SAMPLE SCORE

The Distribution Disk has three files that are related to the simplified NOTRAN music system. These are:

1. SNOTRAN.C
2. SPLAY.C
3. SDEMOSCORE.T

Don't confuse any of these files with the music files on the Demonstration Disk. Those on the demo disk are slightly different because they are set up for playing pre-coded music in one step.

To compile and play the sample score, do the following (some experience with the CODOS operating system is assumed):

1. Insert a copy of the Distribution Disk into disk drive 0 and OPEN it. If you intend to use the music system extensively, you may wish to make a copy disk expressly for music work.

2. Start the NOTRAN compiler by typing in: SNOTRAN and a carriage return. The program called SNOTRAN will be loaded and run.

3. NOTRAN will now print on the screen:

```
MTU-130 SIMPLIFIED NOTRAN MUSIC COMPILER
ENTER NAME OF SOURCE FILE -
```

You should type in: SDEMOSCORE.T and a carriage return. This is the name of a file of NOTRAN music language statements for the demonstration piece.

4. NOTRAN will now print on the screen:

```
ENTER LISTING FILE NAME OR DEVICE NAME (N=NONE) -
```

If you have a printer attached to the system you may wish to respond with a P and a carriage return. Otherwise, respond with a C which will show the listing on the display screen. You can also enter a file name, such as SDEMOSCORE.L, and the listing will be written onto that file so you can look at it carefully later with the BROWSE or EDIT programs.

5. NOTRAN will finally print:

```
PLEASE ENTER OBJECT FILE NAME -
```

You may enter any file name you want but it is recommended that you enter SDEMOSCORE.M and a carriage return. The .M extension reminds you that it is a file of music object code that corresponds to the source file called SDEMOSCORE.T.

6. The NOTRAN compiler will now read the source file, display the listing, and write the object file to disk. Although this demonstration score won't have any, any errors detected will appear on the listing preceeded by **.

7. When the compilation is finished, NOTRAN will print:

```
COMPILATION COMPLETE
```

and return to CODOS with the music object file ready for performance.

8. Type in the following CODOS command to load the music object file into memory:

```
GET SDEMOSCORE.M
```

9. To play the music, type in this CODOS command:

```
SPLAY
```

10. The demonstration score will immediately start to play. It will return to CODOS when finished. If you wish to terminate the music prematurely, press the BRK keyboard key.

11. After the score has been compiled, you need only do steps 8 and 9 to play it any time you want.

2. TRANSCRIBING WRITTEN MUSIC INTO NOTRAN

Translating a written music score into NOTRAN statements is a simple, almost mechanical process that is easy to master. There is much room for creativity however in the orchestration and arrangement of the piece. Or if you have musical experience, the notes themselves can be altered to suit your taste.

A Simplified NOTRAN score consists of two sections, called the Commands Section, and the Notes Section. Statements in the Commands Section specify such things as the tempo, number of voices, tone colors, assignment of voices to tone colors, and the playing sequence of segments of the score including repeats. Statements in the Notes Section specify the actual notes to be played.

Typically the score is broken up into a number of "segments" where each segment is a group of related notes. By means of commands in the Commands Section, the tempo, voice assignments, and other parameters may be changed between segments. Since most music contains a great deal of repetition, only those segments that are unique need be coded in the Notes Section and commands in the Commands Section can be used to play the segments in the proper order.

The structure of a typical NOTRAN score can be seen by examining the skeleton below:

```
                *-comments-
                NVOICES 4
                WAVE 5 .....
Commands        ASSIGN 1 1 3 5
Section         TEMPO 1/4=500
                PLAY 100
                PLAY 200
                TEMPO 1/4=400
                PLAY 100
                ENDCMD
                *-comments-
                SEGMENT 100
                 -notes-
                 -notes-
Notes           ENDSEG
Section         *-comments-
                SEGMENT 200
                 -notes-
                 -notes-
                ENDSEG
                END
```

The initial -comments- typically give the song's title, the coder's name, and other identifying information. The NVOICES, WAVE, and ASSIGN statements define characteristics of the "orchestra" to be used and define a new "instrument". The TEMPO statement establishes the speed at which the music is to be played. The PLAY 100 statement plays the first segment of notes and the PLAY 200 statement plays the second segment. Then the tempo is increased slightly and finally the first segment of notes is played again. The ENDCMD command marks the end of the Commands Section and thus ends the performance.

In the Notes section is coding for the first segment (arbitrarily called segment number 100), and then coding for the second segment. Comments preface each segment identifying to a human reader what part of the score is included in the segment. Each segment starts with a SEGMENT statement, contains the notes associated with the segment, and ends with an ENDSEG statement. An END statement after the last segment marks the end of the NOTRAN score.

2.1 FORMAT OF NOTRAN STATEMENTS

A score written in NOTRAN is composed of lines of text where each line is generally an individual statement, much like any other programming language. The length of a NOTRAN line can be anything up to 254 characters but you will probably want to make lines 80 characters or less to simplify editing with the MTU Editor. In fact, there is an incentive to keep them down to 53 characters or less because then the listing lines produced when the score is compiled will fit on the display or 80 column printer without being broken. All statements are one line long except possibly the WAVE statement which can be continued to several lines.

Lines that begin with an * in column 1 are regarded as comment lines and are ignored by the compiler. Comment lines will however be printed in the listing. Completely blank lines are not allowed; if you want to space out the score for improved readability, put an * at the beginning of any blank lines. Lines beginning with anything other than an * are examined by the compiler and expected to be a valid command or note statement.

NOTRAN statements are generally made up of keywords, keyletters, numbers, and punctuation. Keywords are command names such as TEMPO or WAVE or END. Keywords must be spelled out in all capitals and must be all one word, that is, not broken up by blanks. Keyletters such as C, H, S, or # are used for various purposes in NOTRAN statements depending on the statement being discussed. For example, in a note statement, C would refer to a musical pitch or in a WAVE statement H would refer to a harmonic. When a keyletter is called for, the exact upper case letter required should be used.

Numbers are used extensively to define quantities such as tempo, octaves, percentages, and identifiers. In NOTRAN only positive whole numbers are used; a decimal point is flagged as an error. The largest number recognized is 65535 but most statements limit numbers to values much smaller than this. Commas must not be used in large numbers, instead the digits must be strung together and there must be no intervening blanks. You may place blanks immediately before a number however for improved readability.

Punctuation is used to separate the various elements of a statement from each other such as numbers in a list. In most NOTRAN statements blanks perform this function just as in CODOS commands. In a couple of cases however commas are used to "tie together" a group of related parameters while the groups are separated by blanks. When blanks are used like this for punctuation, you may use any number you wish as long as there is at least one.

Comments may also be added at the end of a statement since the NOTRAN compiler never looks beyond the end of the meaningful part of the statement. Note however that some kinds of statement errors may indeed cause NOTRAN to look beyond the intended end of the statement and signal additional errors when it tries to interpret the comments.

Let's examine the anatomy of the sample NOTRAN statement below:

```
WAVE 5   75  H1,25,0; H2,50;  H3,25  DEFINE WAVEFORM FOR HIGH NOTES.
```

It is not important to understand what the statement does right now, just the rationale behind its syntax. The word WAVE at the beginning is a keyword and is also the name of the statement thus this is called a WAVE statement. When the compiler sees the keyword WAVE, it then knows how to interpret the remaining information. Notice that WAVE is all the way to the left margin as required. Following the keyword is a separating blank and then the first parameter, a 5, which is the wave ID number. This is separated from the amplitude parameter, a 75, by several blanks although only 1 was necessary.

Following the first 2 parameters are 3 parameter groups. The keyletter H immediately followed by a 1 specifies harmonic number 1. Associated with harmonic 1 is an amplitude of 25 and a phase of zero. These parameters are separated from each other yet held together by commas. A semicolon (;) appears at the end of the group signifying in this case that there is another group to come and then a blank separates this group from the next group. Notice that the second and third groups have only the harmonic number and amplitude specified. The score writer in this case didn't care about the phase and allowed the compiler to make a random choice by omitting it. Since there is no semicolon after the third group, the compiler assumes that all of the harmonics of interest have been specified and goes on to the next statement, ignoring the annotation at the end of the statement.

2.2 THE COMMANDS SECTION

Statements in the Commands Section always begin with the command name in column 1, that is, starting at the left margin. Each of the commands recognized by Simplified NOTRAN is described in the sections below:

2.2.1 ASSIGN

PURPOSE: To assign waveforms (tone colors) to specific voices.

SYNTAX: ASSIGN <waveV1><waveV2><waveV3><waveV4>

ARGUMENTS:
<waveV1> = Waveform ID number to be sounded by voice 1
<waveV2> = Waveform ID number to be sounded by voice 2
<waveV3> = Waveform ID number to be sounded by voice 3
<waveV4> = Waveform ID number to be sounded by voice 4

EXAMPLES:

```
ASSIGN 2 2 7 8
```

Assigns waveform number 2 to voices 1 and 2 and assigns waveform 7 to voice 3 and waveform 8 to voice 4. Since waveform 2 is predefined by the system, it need not be defined in a WAVE statement. Waveforms 7 and 8 however must be defined by a WAVE statement before any notes are actually played with them.

```
ASSIGN 4 4 0 0
```

Assigns waveform number 4 to voices 1 and 2. Waveform 0, which is silence, is assigned to voices 3 and 4. This statement would typically be used when only two musical parts are playing.

NOTES

1. All 4 voices must be assigned to waveforms even if they are not used. Unused voices should be assigned to waveform 0 (silence) to avoid extraneous sounds.

2. 16 is the maximum allowable waveform number.

3. Waveforms 1-4 are predefined and need not be defined with a WAVE statement. Waveforms 5-16 must be defined before use. If they are not defined there is no error message but the sound generated will invariably be unpleasant.

2.2.2 ENDCMD

PURPOSE: To designate the end of the Commands Section.

SYNTAX: ENDCMD

ARGUMENTS: None.

EXAMPLES: ENDCMD

Defines the end of the Commands Section. The next statement will begin the Notes Section.

NOTES:

1. The Commands Section must be terminated with an ENDCMD statement. If it is not, the compiler will try to interpret the Notes Section as commands and give a multitude of errors.

2.2.3 NVOICES

Purpose: To specify the maximum number of simultaneous voices that will sound.

SYNTAX: NVOICES <#ofvoices>

ARGUMENTS: <#ofvoices> = the maximum number of voices that will be sounding in the succeeding music. Minimum value is 1, maximum is 4.

EXAMPLES:

```
NVOICES 4
```

Specifies that up to 4 voices may play at once in the music to be played following this statement up until another NVOICES statement.

```
NVOICES 2
```

Specifies that only 2 voices will be playing in the following music. An ASSIGN statement should follow this statement to assign voices 3 and 4 to waveform 0 (silence).

NOTES:

1. Four voices are specified by default when SPLAY is loaded thus only a change from 4 requires an NVOICES statement.

2. Specifying fewer than 4 voices is typically used only to conserve memory in the compiled score since each musical event requires NVOICES+1 bytes of memory to encode. Unless your score becomes very large, just use NVOICES 4.

3. The NVOICES statement in the Commands Section must agree with the MAXVOICE statement in the Notes Section in effect when the notes were compiled.

2.2.4 PLAY

PURPOSE: To designate the score segment in the Notes Section to be played next.

SYNTAX: PLAY <segmentID>

ARGUMENTS: <segmentID> = A number matching the ID of the segment to play next.

EXAMPLES:

```
PLAY 10
```

will play the group of notes given an ID of 10 in the Notes Section and then return to the Commands Section for the next command.

```
PLAY 2050
```

will play the group of notes given an ID of 2050 in the Notes Section and then return to the Commands Section for the next command.

NOTES:

1. The segment ID given must match the ID of one of the segments in the Notes Section. If it does not, an error will be given at the end of the listing.

2. All of the notes between the corresponding SEGMENT and ENDSEG statements in the Notes Section will be played with the current tempo, waveform assignments, and number of voices.

3. Segments may be played as many times as desired and in any order.

2.2.5 TEMPO

PURPOSE: To specify the tempo (speed) at which the music is to be played.

SYNTAX: TEMPO <num>/<den>=<duration>

ARGUMENTS: <num>/<den> = any reasonable fraction of a measure such as 1/4
<duration> = number of milliseconds (thousandths of a second) corresponding to the designated fraction of a measure.

EXAMPLES:

```
TEMPO 1/4=500
```

Specifies that a quarter note will last for 500 milliseconds (.5 second). This corresponds to 120 beats per minute if a quarter note is one beat.

```
TEMPO 3/8=700
```

Specifies that three eighth-notes together will take 700 milliseconds. Therefore, a single eighth-note will take 700/3 or 233 milliseconds.

NOTES:

1. The specified tempo applies to all segments played until another TEMPO statement is seen. Also, the tempo may be changed only between segments.

2. The tempo may be re-defined as often as desired.

3. The slowest tempo available is 1/1=7600 or equivalent. Tempos faster than 1/1=500 or equivalent will become increasingly inaccurate.

PURPOSE: To specify a new tone color in terms of its harmonics.

SYNTAX: WAVE <ID> <waveamp> H<harm#>,<amp> [,<phase>] [;H<harm#>,<amp>[,<phase>]...]

ARGUMENTS: <ID> = waveform ID number between 1 and 16 inclusive.
<waveamp> = overall waveform amplitude value between 0 and 100 inclusive
<harm#> = desired harmonic number between 1 and 127 inclusive.
<amp> = harmonic amplitude between 0 and 100 inclusive.
<phase> = harmonic phase in percent of a cycle. 0=sine wave.

EXAMPLES:

```
WAVE 10 75  H1,50,0;  H2,25,75;  H3,15,50;  H4,9,25
```

Generates waveform number 10 with an overall amplitude of 75% of maximum with fundamental (first harmonic) 50% of the total, second harmonic 25%, third harmonic 15% and fourth harmonic 10% of the total. The fundamental is a cosine wave (0 phase), the second harmonic is a sine wave (3/4 cycle or 270 degrees leading), the third harmonic is a negative cosine wave, and the fourth harmonic is a negative sine wave.

```
WAVE 2 127 H1,50; H3,50; H5,25; H7,25
```

Generates waveform number 2 with an overall amplitude of 127 with a fundamental amplitude 1/3 the total, third harmonic 1/3, fifth harmonic 1/6 and seventh harmonic 1/6 the total. All other harmonics are zero. The phases are chosen randomly. Note that this will re-define one of the 4 default waveforms for this piece.

```
WAVE 7 63 H1,15; H2,10; H3,6; H4,4; H5,3; H6,5; H7,12;
          H8,25; H9,10; H10,6; H11,4
```

This illustrates the use of a continued WAVE statement. Since a ; follows the H7 specification, the next line is assumed to be a continuation of the WAVE statement.

NOTES:

1. For most uses, the <waveamp> parameter should be set to 100. A small number may be used it if is desired that notes played with this waveform sound more softly than the standard default waveforms. If the amplitude is too low however, the background noise level will be excessive. See section 3.3 for more about this.

2. The harmonic amplitude parameters merely give relative amplitudes among the harmonics. There is no requirement that they add up to exactly 100%.

3. Waveforms 1-4 are predefined when SPLAY is loaded. If a WAVE statement redefines one of these, it will remain redefined until SPLAY is reloaded.

4. In most cases the phase parameters will have little effect on the timbre of the tone. They can have an effect on the apparent volume independent of the <waveamp> parameter however. Unless you desire a specific set of phases, it is best to leave them unspecified so they will be randomly set. Do not set them all to zero unless specifically desired, otherwise the tone volume may be less than expected.

5. You should not specify high harmonic numbers for waveforms that will be played with high pitches. See section 3.3 for details about this.

2.3 THE NOTES SECTION

Statements in the Notes Section may either be notes section commands or note statements. Commands must start at the left margin (column 1). If column 1 is blank or contains a digit, then it and columns 2, 3, and 4 are ignored and a note statement is assumed to start in column 5. Don't confuse commands in the Notes Section with commands in the Commands Section because they have different names and do different things. The following discussion defines the commands recognized in the Notes Section:

2.3.1 END

PURPOSE: To end the Notes Section and thus the entire score.

SYNTAX: END

ARGUEMENTS: None.

EXAMPLES: END

Defines the end of the score and terminates the compilation.

2.3.2 ENDSEG

PURPOSE: To end a segment of notes statements.

SYNTAX: ENDSEG

ARGUEMENTS: None.

EXAMPLES: ENDSEG

Defines the end of the previous segment of music and will terminate the PLAY command that initiated playing of this segment.

NOTES:

1. Every segment of music in the Notes Section must be terminated by an ENDSEG statement. Even if the whole piece is coded as one segment, it must be terminated by an ENDSEG statement before being terminated by an END statement.

2.3.3 MAXVOICE

PURPOSE: To specify the maximum voice number that will be used.

SYNTAX: MAXVOICE <maxvoice#>

ARGUMENTS: <maxvoice#> = the maximum voice number that will be used in the following notes coding. Minimum value is 1, maximum is 4.

EXAMPLES:

```
MAXVOICE 4
```

Specifies that voice numbers up to 4 will be used in the following notes coding.

```
MAXVOICE 2
```

Specifies that only 2 voices will be playing in the following music.

NOTES:

1. The maximum voice number specified must match the NVOICES command in effect when this segment is PLAYed.
2. The effect of a MAXVOICE command continues through succeeding segments until another MAXVOICE statement is seen.
3. MAXVOICE must not be used within a segment, only between segments.
4. The default maximum voice number is 4.

2.3.4 SEGMENT

PURPOSE: To identify the following segment of notes so that it can be PLAYed.

SYNTAX: SEGMENT <segmentID>

ARGUMENTS: <segmentID> = A number between 1 and 65535 used to uniquely identify this segment.

EXAMPLES: SEGMENT 100

Identifies the following segment of music as segment number 100.

NOTES:

1. Any number may be used for a segment ID. The value has no significance and need not be in ascending sequence. The only requirement is that no other segment have the same ID number.
2. You may have several SEGMENT statements representing different "entry points" in the same group of notes terminated by a single ENDSEG statement.

2.3.5 Note Statements

When the first character of a statement in the Notes Section is not a letter or asterisk, then the statement is assumed to be a note statement. When a note statement is so recognized, the first 4 columns are skipped so you can actually put whatever you like in them (except that column 1 has to be blank or have a digit in it in order to be recognized as a note statement). Thus, starting in column 5, the NOTRAN compiler expects to find the first note or rest specification of the statement.

A note specification consists of several digits, letters, and symbols all written together without any blanks between them. The first item in a note specification is the voice number that will play the note. In simplified NOTRAN this must be a digit in the range of 1 to MAXVOICE which is usually 4.

Immediately after the voice number is the pitch specification. The musical pitch is specififed by a keyletter which may be from A to G inclusive and must be upper case. Following the pitch letter may optionally be a sharp (#) or flat (@) modifier. The sharp simply raises the specified pitch by a half-step while a flat lowers it by a half-step. Double-sharps and double-flats (two half-step raise and lower) are also permitted simply by writing the # or @ twice. After the pitch letter or modifier is an octave number which indicates in which octave range the note should sound. Middle C (261.6Hz) is represented as C4 and concert A (440Hz) is represented as A4. Higher numbers represent higher octaves while lower numbers represent lower octaves. Octaves "turn over" at the Cs so the B just below middle C is B3. The lowest pitch available is C1 (32.7Hz) and the highest is C7 (2093Hz).

A comma separates the pitch specification just described from the duration specification. The note duration is represented by a fraction such as 1/4 for a quarter note or 1/16 for a sixteenth note, etc. You may specify any reasonable fraction you like (numerator and denominator less than 256), that evaluates to unity (1/1) or less. NOTRAN doesn't check for weird fractions such as 2/5 or 3/11 so you can set up some strange rhythms or mimic imprecise durations if you like. Following the fraction you may optionally place a period which simply multiplies the previous fraction by 3/2 (a dotted note).

The following are some legal note specifications along with verbal descriptions:

1C4,1/4	Middle C, a quarter note played by voice 1.
3B@4,1/1	B-flat above middle C, a whole note played by voice 3.
2F#2,1/8.	F-sharp two octaves below middle C, a dotted eighth note, voice 2.
4C@5,1/4	C-flat (equivalent to B4) above middle C, a quarter note.
1F##4,1/4..	F double-sharp (equivalent to G) double-dotted quarter note (equivalent to 1/4 x 3/2 x 3/2 = 9/16).
3G5,1/24	G an octave above middle C, a sixteenth note triplet (i.e., three of these take the same time as two sixteenth notes).
2E5,1/5	E an octave above middle C, a fifth note (drunken music anyone?).

The following specifications are illegal for the reasons noted:

5C4,1/4	Voice number not 1, 2, 3, or 4.
3B4@,1/1	Sharp or flat must come before the octave number.
2F#2,1/1.	The final evaluated duration must not exceed a whole note (1/1).
3H5,1/7	The pitch letter must be A-G.
1C4,1/72	Not really illegal but the actual duration of such a short note may be substantially in error due to roundoff.

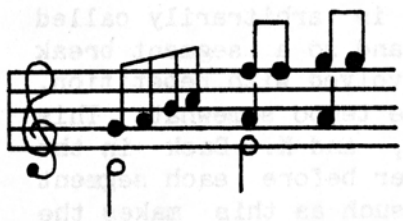
A rest specification consists of the keyletter R followed by a comma followed by the same kind of duration specification as is used for notes. In NOTRAN, rest specifications are needed only to specify complete silence (no notes playing), or to alter the "shortest duration" decision described below. Rests in individual voices occur automatically when no note is specified for the voice.

In NOTRAN, multi-part music (several notes playing at once) is entered a chord-at-a-time. Consequently most note statements will consist of several note specifications, one for each note in the chord. When there are several note specifications on a line, you must put a semicolon (;) after each specification except the last one. This tells the compiler that another note specification follows and should be processed. Note that each note specification on the line must have a different voice specified since a voice may only play one note at a time in the simplified NOTRAN system. The musical example below shows how a short series of chords would be represented in NOTRAN:

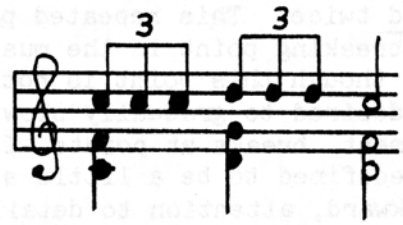
```
001 1C4,1/4; 2E4,1/4; 3G4,1/4; 4C5,1/4
    1C4,1/4; 2F4,1/4; 3A4,1/4; 4C5,1/4
    1C4,1/4; 2E4,1/4; 3G4,1/4; 4C5,1/4
    1B3,1/4; 2D4,1/4; 3G4,1/4; 4B4,1/4
    1C4,1/4; 2E4,1/4; 3G4,1/4; 4C5,1/4
```

When starting to play a note statement, all of the notes specified will <u>start</u> sounding their designated pitches at the same time. If all of the durations are the same as in the example above, then they will all stop sounding simultaneously as well. However if one of the durations is shorter than the others, it will finish sounding first. When this occurs, the computer will immediately move to the next line and start sounding the notes it specifies even though some notes are still sounding from the previous line. If there are 3 or 4 different durations on a line, the computer continues after the <u>shortest</u> one has expired. If a rest specification is mixed in with note specifications on the line, the duration of the rest is included in the shortest duration search. If the rest is indeed the shortest, then the next line is played even though <u>none</u> of the notes on the previous line have finished.

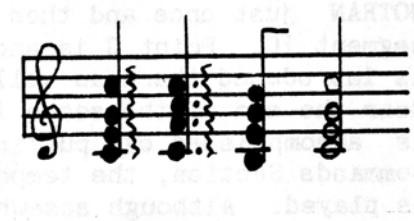
This one rule allows you to easily code things such as eighth notes against quarter notes, triplets, and other musical constructions. You must be careful not to start a new note with a voice before the note it was previously sounding expires, otherwise you will get an error message and the previously sounding note will be cut short. Below are some musical examples and how they would be coded:



```
1C4,1/2; 2G4,1/8
         2A4,1/8
         2B4,1/8
         2C5,1/8
1E4,1/2; 2A4,1/4; 3E5,1/8
                  3E5,1/8
         2A4,1/4; 3F5,1/8
                  3F5,1/8
```




```
1C4,1/4; 2F4,1/4; 3C5,1/12
                  3C5,1/12
                  3C5,1/12
1D4,1/4; 2G4,1/4; 3D5,1/12
                  3D5,1/12
                  3D5,1/12
1C4,1/2; 2F4,1/2; 3C5,1/2
```



```
1C4,1/4;  R,1/24
2E4,5/24; R,1/24
3G4,4/24; R,1/24
4C5,3/24
1C4,9/24; R,1/24
2F4,8/24; R,1/24
3A4,7/24; R,1/24
4C5,9/24
1B3,1/8; 2E4,1/8; 3G4,1/8; 4B4,1/8
1C4,1/2; 2E4,1/2; 3G4,1/2; 4C5,1/2
```

2.4 CODING A SAMPLE SCORE

In this section you will be led through the process of transcribing a sample score into NOTRAN. It is recommended that you read the discussion below in its entirety while referring to the musical score and its NOTRAN equivalent listed on the following pages. Points of interest are indicated by a circled letter and pointer. Then type in the score using the MTU Editor, compile it (see section 1.4), correct any typing errors caught by the compiler, and play it to find out what well known song it actually is.

2.4.1 Preliminaries

The first two lines of the score are simply comment lines for identification. Next, the number of voices is set to 4 since the vast majority of the music will use 4 (the maximum number) voices. Theoretically this is not necessary since 4 is the default but it doesn't hurt to be specific.

Next, a new waveform is defined. One of the voices in this score will always be playing very low notes, in fact never above A3 (220Hz). Each of the 4 default waveforms, however, is designed to be useable with high notes as well as low. Consequently only a few harmonics were used to create them and as a result they sound dull when played at low pitches. The waveform defined here has a large number of harmonics (up to the 16th) which will give a rich sounding bass voice not unlike a bassoon. Before any notes are actually played, it is necessary to assign waveforms to the voices. Here we will assign waveform 1 (the church organ waveform) to voice 1 which will carry the melody. Waveform 2 (a nice mellow sound) will be assigned to the accompanyment voices 2 and 3 and then the newly defined waveform will be assigned to the bass waveform 4. The assignment of voice numbers to specific musical parts is up to the user but the assignment just described works well for most music.

The last preliminary action is to specify a tempo. A look at the score (point A) shows that a quarter note is one beat and that the tempo should be 100 beats per minute. This implies that a quarter note is 1/100 of a minute or .6 second. Thus the appropriate tempo statement in TEMPO 1/4=600.

2.4.2 Segmenting the Score

Examination of the score reveals two rather long sections that are identical (points B-D is the same as points E-F). Therefore this part needs to be written in NOTRAN just once and then played twice. This repeated part is arbitrarily called segment 10. Point G is another breaking point in the music and so a segment break is introduced there as well even though this point is not involved with repetition. Near the end of the score it is desired to gradually slow the tempo somewhat. This is accomplished by putting segment breaks at points I, J, and K. Back in the Commands Section, the tempo is redefined to be a little slower before each segment is played. Although somewhat awkward, attention to details such as this makes the music sound much more natural.

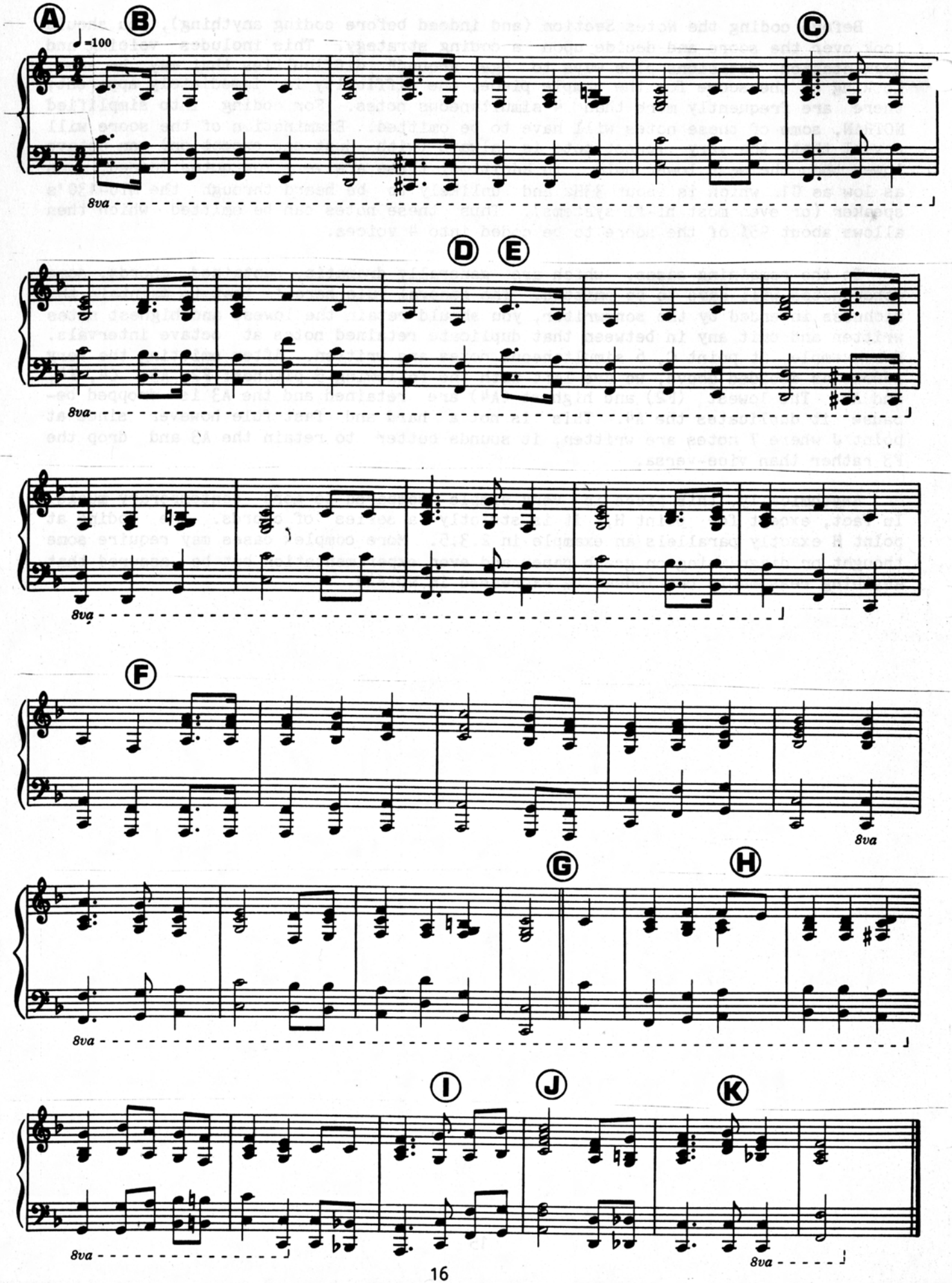
2.4.3 Coding the Notes

Before coding the Notes Section (and indeed before coding anything), you should look over the score and decide upon a coding strategy. This includes voicing and segmentation decisions plus ways to "get around" difficulties that may be seen. Looking at the score for the sample piece, one difficulty is immediately apparent; there are frequently more than 4 simultaneous notes. For coding into simplified NOTRAN, some of these notes will have to be omitted. Examination of the score will reveal that the very lowest note is always (with just one exception) an octave lower than the next lower note. In addition, these are very low notes indeed, some as low as C1 which is about 33Hz and unlikely to be heard through the MTU-130's speaker (or even most hi-fi systems). Thus these notes can be omitted which then allows about 95% of the score to be coded into 4 voices.

In the remaining cases, which are generally dramatic, sustained chords, some other notes will have to be omitted. The general rule here is that to maintain the richness intended by the songwriter, you should retain the lowest and highest notes written and omit any in between that duplicate retained notes at octave intervals. For example, at point C, 6 simultaneous notes are written. After omitting the very lowest as decided above, we are left with the following 5 pitches: F2, A3, C4, F4, and A4. The lowest (F2) and highest (A4) are retained and the A3 is dropped because it duplicates the A4. This is not a hard and fast fule however since at point J where 7 notes are written, it sounds better to retain the A3 and drop the F3 rather than vice-versa.

Rhythmically, this piece is very simple compared to more contemporary music. In fact, except for point H, it is strictly a series of chords. The coding at point H exactly parallels an example in 2.3.5. More complex cases may require some thought or diagramming on graph paper and even experimentation but be assured that anything reasonable can indeed be expressed in NOTRAN.

Sample Musical Score



2.4.5

```
* Sample Score
* Coded 1/13/82 CHAMBERLIN
*
NVOICES 4
* DEFINE A RICH WAVEFORM FOR THE LOW BASS
WAVE 5 100 H1,75;  H2,50;  H3,30;  H4,20;  H5,15;  H6,10;
           H7,10;  H8,10;  H9,10;  H10,10; H11,15; H12,25;
           H13,15; H14,10; H15,5;  H16,3
* VOICE 1 = TREBLE THROUGH VOICE 4 = BASS
ASSIGN 1 1 1 5
TEMPO 1/4=600    * POINT A
PLAY 10
PLAY 10
PLAY 20
PLAY 30
TEMPO 3/12=650
PLAY 40
TEMPO 1/4=725
PLAY 50
TEMPO 1/4=800
PLAY 60
TEMPO 1/4=1100
PLAY 70
ENDCMD
*
*
MAXVOICE 4
SEGMENT 10
 BE 1C4,1/8.;                                    4C3,1/8.
    1A3,1/16;                                    4A2,1/16
    1F3,1/4;                                     4F2,1/4
    1A3,1/4;                                     4F2,1/4
    1C4,1/4;                                     4F2,1/4
    1F4,1/2;    2D4,1/2;    3A3,1/2;             4D2,1/2
    1A4,1/8.;   2E4,1/8.;   3A3,1/8.;            4C#2,1/8.
    1G4,1/16;               3G3,1/16;            4C#2,1/16
    1F4,1/4;                3F3,1/4;             4D2,1/4
    1A3,1/4;                3F3,1/4;             4D2,1/4
    1B3,1/4;    2G3,1/4;    3E3,1/4;             4G2,1/4
    1C4,1/2;    2G3,1/2;    3E3,1/2;             4C3,1/2
    1C4,1/8;                                     4C3,1/8
    1C4,1/8;                                     4C3,1/8
 C  1A4,1/4.;   2F4,1/4.;   3C4,1/4.;            4F2,1/4.
    1G4,1/8;                3G3,1/8;             4G2,1/8
    1F4,1/4;                3F3,1/4;             4A2,1/4
    1E4,1/2;    2C4,1/2;    3G3,1/2;             4C3,1/2
    1D4,1/8.;   2B@3,1/8.;                       4B@2,1/8.
    1E4,1/16;   2C4,1/16;   3G3,1/16;            4B@2,1/16
    1F4,1/4;    2C4,1/4;    3A3,1/4;             4A2,1/4
    1F4,1/4;                                     4F3,1/4
    1C4,1/4;                                     4C3,1/4
    1A3,1/4;                                     4A2,1/4
 DF 1F3,1/4;                                     4F2,1/4
ENDSEG
SEGMENT 20
    1A4,1/8.;   2F4,1/8.;   3A3,1/8.;            4F2,1/8.
    1A4,1/16;   2F4,1/16;   3A3,1/16;            4F2,1/16
    1A4,1/4;    2F4,1/4;    3A3,1/4;             4F2,1/4
    1B@4,1/4;   2F4,1/4;    3B@3,1/4;            4G2,1/4
    1C5,1/4;    2F4,1/4;    3C4,1/4;             4A2,1/4
```

```
    1C5,1/2;    2F4,1/2;    3C4,1/2;    4A2,1/2
    1B@4,1/8;   2F4,1/8;    3B@3,1/8;   4G2,1/8
    1A4,1/8;    2F4,1/8;    3A3,1/8;    4F2,1/8
    1G4,1/4;    2E4,1/4;    3G3,1/4;    4C3,1/4
    1A4,1/4;    2F4,1/4;    3A3,1/4;    4F3,1/4
    1B@4,1/4;   2E4,1/4;    3B@3,1/4;   4G3,1/4
    1B@4,1/2;   2G4,1/2;    3E4,1/2;    4C3,1/2
    1B@4,1/4;   2E4,1/4;    3B@3,1/4;   4C2,1/4
    1A4,1/4.;   2C4,1/4.;   3A3,1/4.;   4F2,1/4.
    1G4,1/8;    2C4,1/8;    3G3,1/8;    4G2,1/8
    1F4,1/4;    2C4,1/4;    3F3,1/4;    4A2,1/4
    1E4,1/2;    2C4,1/2;    3G3,1/2;    4C3,1/2
    1D4,1/8;                3D3,1/8;    4B@2,1/8
    1E4,1/8;    2C4,1/8;    3E3,1/8;    4B@2,1/8
    1F4,1/4;    2C4,1/4;    3F3,1/4;    4A2,1/4
    1A3,1/4;    2F3,1/4;                4D3,1/4
    1B3,1/4;    2G3,1/4;    3F3,1/4;    4F2,1/4
    1C4,1/2;    2G3,1/2;    3E3,1/2;    4C2,1/2
ENDSEG
* G
SEGMENT 30
    1C4,1/4;                            4C3,1/4
    1F4,1/4;    2C4,1/4;    3A3,1/4;    4F2,1/4
    1F4,1/4;    2B@3,1/4;   3G3,1/4;    4G2,1/4
  H 1F4,1/8;    2C4,1/4;    3A3,1/4;    4A2,1/4
    1E4,1/8
    1D4,1/4;    2B@3,1/4;   3F3,1/4;    4B@2,1/4
    1D4,1/4;    2B@3,1/4;   3F3,1/4;    4B@2,1/4
    1D4,1/4;    2C4,1/4;    3F#3,1/4;   4A2,1/4
    1G4,1/4;    2B@3,1/4;   3G3,1/4;    4G2,1/4
    1B@4,1/8;   2B@3,1/8;               4G2,1/8
    1A4,1/8;    2A3,1/8;                4A2,1/8
    1G4,1/8;    2G3,1/8;                4B@2,1/8
    1F4,1/8;    2F3,1/8;                4B2,1/8
    1F4,1/4;    2C4,1/4;    3A3,1/4;    4C3,1/4
    1E4,1/4;    2C4,1/4;    3G3,1/4;    4C2,1/4
    1C4,1/8;                            4C3,1/8
    1C4,1/8;                            4B@2,1/8
    1F4,1/4.;   2C4,1/4.;   3A3,1/4.;   4A2,1/4.
ENDSEG
SEGMENT 40
 I  1G4,1/8;                3G3,1/8;    4C3,1/8
    1A4,1/8;                3A3,1/8;    4F3,1/8
    1B@4,1/8;               3B@3,1/8;   4G3,1/8
ENDSEG
SEGMENT 50
 J  1C5,1/2;    2A4,1/2;    3F4,1/2;    4A3,1/2
    1F4,1/8;    2D4,1/8;    3A3,1/8;    4D3,1/8
    1G4,1/8;    2B3,1/8;    3F3,1/8;    4D@3,1/8
    1A4,1/4.;   2C4,1/4.;   3A3,1/4.;   4C3,1/4.
ENDSEG
SEGMENT 60
 K  1B@4,1/8;   2F4,1/8;    3D4,1/8;    4C3,1/8
    1G4,1/4;    2E4,1/4;    3B@3,1/4;   4C3,1/4
ENDSEG
SEGMENT 70
    1F4,1/2;    2C4,1/2;    3A3,1/2;    4F2,1/2
ENDSEG
*
END
```

3. DEFINING YOUR OWN TIMBRES

One of the unusual features of simplified NOTRAN and indeed the sound synthesis technique utilized by the MTU-130 is the ability of the user to define new timbres (tone colors). This definition is done in the most flexible way possible: direct specification of the fundamental building blocks of sound, the harmonic partials. In the simplified NOTRAN system you may create literally an infinite variety of organ-like timbres.

In the simplified NOTRAN system, 4 timbres or waveforms have been predefined as described below along with their upper useful pitch range:

Waveform 1 - Bright church organ timbre. Uesful from C1 to C5.
Waveform 2 - A mellow, flutey timbre. Useful from C1 to G5.
Waveform 3 - A thin, reedy timbre. Useful from C1 to C5.
Waveform 4 - A full-bodied, robust sounding timbre. Useful from C1 to C6.

Waveforms 5 through 16 are available for definition by the user. When used they become part of the song score, not part of the NOTRAN system so overall you have an unlimited total number available. You may also redefine waveforms in the middle of a song if you like the only drawback being a fraction of a second pause while the new waveform is being computed.

3.1 TONES AND HARMONICS

Any steady musical tone is actually a mixture of simple tones called sine waves. A single sine wave tone, such as produced by a tuning fork, itself is essentially colorless. A combination of these simple tones however can represent any steady tone at all. Varying musical tones (such as the "wah" of a muted trombone) are composed of a changing mixture of simple tones. Although such effects cannot be synthesized by the Simplified NOTRAN system, more advanced music software for the MTU-130 can indeed do so.

Any of these simple sine tones can be completely described with just three measurements or parameters. Frequency is perhaps the most important and is directly related to the pitch of the simple tone. Amplitude is also important and is directly related to the loudness of the tone. The phase is much less important when humans are listening but as we shall see later, it can be significant to the computer. You may specify all three of these parameters for each component of user defined timbres.

A real musical tone then is a mixture of several of these simple sine wave tones, each with its own frequency, amplitude, and phase. Theoretically there need not be any relationship among these parameters and indeed for some tones, such as bells, there isn't. But for organ pipes (and other wind instruments) there is a very definite relationship among the frequencies of the component tones. This is called a harmonic relationship because the frequency of each of the component tones is an integral multiple of the tone's own frequency in cycles per second. Thus if we have a tone frequency of 440Hz (cycles per second) which is concert A, then the frequencies of each of its components will be 1, 2, 3, 4, 5, etc. times 440Hz or 880, 1320, 1760Hz, etc. If the same instrument played a lower note, say 370Hz, then its component tones would also be proportionally lower in frequency as well. What this means is that the actual frequency of each component does not have to be given at all, just its multiple. Thus the term "second harmonic" will refer to that component tone whose frequency is 2 times the overall tone's frequency, "third harmonic" to the 3X frequency tone, etc. A special name is given to the component whose frequency is the same as the tone's frequency; it is called the fundamental component.

Although the frequencies of the component tones are pretty well tied down by a harmonic relationship, the amplitudes are not. The amplitude of each harmonic actually tells how much contribution that harmonic makes to the overall tone color. A harmonic with a small amplitude and thus a low volume level would not be expected to contribute very much whereas a harmonic with a large amplitude would influence the mixture considerably. The effect is much like mixing paints together where color is analogous to frequency and quantity is analogous to amplitude.

The phases of the harmonic components affect the way in which the component sine wave shapes mesh together to produce the shape of the resultant tone's waveform. The phases generally have a very small effect on the audible properties of the mixture however. They do have a large effect on the wave shape of the resulting tone and since this shape must be stored in a numerical array with limited precision, the phases can affect the audible result indirectly. This is explained in more detail in section 3.3. For now we will ignore the phases and assume that they have been assigned randomly.

Given this background then, we can invent our own mixtures of harmonics to produce a variety of timbres that can in turn be played by a NOTRAN voice. Knowing which harmonics to include and what their amplitudes should be to give a particular imagined timbre is mostly a matter of experience and experimentation. For a beginner it is recommended that you start with one of the mixtures given in the next section and then make minor modifications to it to discover the effect of different harmonics and amplitudes. A good way to do this would be to code a short notes segment and then play it with each of several different waveforms by interleaving ASSIGN and PLAY statements in the Commands Section. This makes it much easier to compare the effects of changes to the harmonic composition of the tones.

3.2 THE WAVE STATEMENT

Section 2.2.6 gives a formal description of the WAVE statement so here we will just summarize its use and give some examples. Normally you will want to define all of the waveforms you will use before playing any notes. The reason for this is that it takes time to compute the waveform when WAVE statements are encountered during performance. However if the limit of 16 simultaneously defined waveforms is not enough, you can redefine a waveform between notes segments if you wish. The computation time is less than a second unless you have a large number of harmonics so this can usually be slipped in during natural pauses in the music. Although you can redefine waveforms 1-4 as easily as the others, it is recommended that you do not unless the 12 free ones are insufficient.

For most uses, the overall amplitude parameter should be set to 255. You may use smaller numbers to make the waveform sound more softly if you wish. Don't go overboard however since the dynamic range of the system is not very large thus very quiet tones may become immersed in background noise.

The remainder of the WAVE statement consists of a string of harmonic specifications. These don't have to be in any particular order but having them in order of ascending harmonic numbers makes sense. Only harmonics with non-zero amplitudes need be specified; all the others will automatically have a zero amplitude. Generally amplitudes less than 2 or 3 percent of the total will have an imperceptible effect on the timbre so they might as well be omitted.

Note that the harmonic amplitudes are given asnumbers between 0 and 100. You may consider these numbers to be percentages or just relative values. Any individual harmonic must be 100 or less but it is permissible for the sum of all of the harmonic amplitudes to exceed 100% with no chance of overflow. This allows adjustment room for individual harmonics without having to change all of them to sum up to exactly 100%. Conversely, the sum should not be much less than 100 either, otherwise the contribution of round-off error in the calculations will increase.

If the harmonic phase is not specified, it is chosen at random. On the average this will give a waveform with maximum audible loudness and minimum round-off error (noise) when stored in the waveform table. The phases are random however so there is no guarentee that this will be true. In fact, the unspecified phases will be different each time the song is compiled! If this uncertainty is undesirable, you may specify the phases randomly yourself. The phase parameter is given in 1/100ths of a full sine wave cycle (units of PI/50) although you can specify anything up to 65535 and it will be divided by 100 and the remainder used. Of course if you want a specific waveshape, such as for a sound physics demonstration, you can specify the exact phases desired.

Since WAVE statements are likely to require more than one line, you may continue them from line-to-line. Simply use as many lines as needed with at least one harmonic specification per line. A harmonic specification must not be split between two lines; put the split between the ; of the previous specification and the H of the next one. Remember that there must be a semicolon after the amplitude or phase number of each harmonic specification except the last whether you use continuations or not.

Below are the 4 WAVE statements that were used to create the 4 default waveforms built into the SPLAY program. Study them and use them as a starting point for experimenting with your own timbres.

```
WAVE 1 100 H1,25; H2,25; H4,25; H8,25                          (Bright church organ tone)

WAVE 2 100 H1,70; H3,20; H5,10                                 (Mellow, flutey)

WAVE 3 100 H1,15; H2,10; H3,8; H4,8; H5,10; H6,20; H7,15; H8,10    (Thin, reedy)

WAVE 4 100 H1,40; H2,25; H3,20; H4,15                          (Full-bodied, robust)
```

3.3 OPTIMIZING TONE QUALITY

Like everything else in this world, the digital synthesis technique used by the MTU-130 and Simplified NOTRAN is not perfect. For example, only 9,700 sound wave points are computed every second and sent to the D-to-A converter which means that frequencies only up to about 4.0KHz may be synthesized. Also, round-off error is introduced into the waveform points due to shortcuts in the computation (needed for speed) and the limited resolution of 8 bit D-to-A conversion. This error gives rise to some residual background noise and distortion in the reproduced sound. For best results, it is necessary to be cognizant of these limitations and utilize techniques to circumvent them or minimize their effect.

3.3.1 Avoiding Alias Distortion

The easiest trap to fall into when using digital sound synthesis is trying to generate frequencies that are too high. There is no lower limit so you can shake the room with bass chords but if one of the harmonic components of a tone is too high in frequency, it is "transformed" into an incorrect frequency rather than simply being lost. At worst the result is severe distortion while at best, strange sounds are produced.

Whenever a simple sine wave tone is digitally synthesized, there are actually two tones produced. One of these is at the correct frequency of F Hz while the other, which is called an alias, is 9700-F Hz. Remember that this action applies to each harmonic of each tone individually. When F is fairly small, such as 1000, its alias at 8700 is widely separated from it and the low-pass filter in the MTU-130's D-to-A converter can completely separate them and reject the alias tone. Tones as high as 4000Hz can be separated adequately from their aliases (5700Hz) by this filter. If you try to synthesize a very high frequency, such as 7000Hz, then the correct frequency will actually be blocked by the filter but its alias at 9700-7000=2700Hz will pass through and be heard as the wrong frequency. The only way to avoid the resulting alias distortion is to avoid generating frequencies higher than 4000.

As an example of how it is very easy to get into trouble with alias distortion, consider the case of playing D5, which is about 587Hz, with a voice that has been assigned to waveform 1 as defined on the previous page. The tone will consist of the fundamental frequency, 587Hz, the second harmonic at 1174Hz, the fourth harmonic at 2348Hz, and the eighth at 4696Hz. Thus the eighth harmonic is too high and will be inadequately separated from its alias at 5004Hz. As a result, the highest note you should attempt to play with this waveform is approximately C5.

While the preceeding example was a borderline case that may sometimes be acceptable, a more flagarent violation would be playing C5 (1044Hz) with default waveform 3. Here not just one but four of its harmonics exceed 4000Hz, the worst by so much that its alias is 1348Hz! This tone can be expected to be severely distorted. The rule then is that the highest harmonic frequency of the highest note in the score that plays it must not exceed 3500 to 4000Hz if alias distortion is to be avoided.

On the other hand, playing very low notes with waveforms designed for high notes (and consequently having only a few harmonics) will give an excessively muffled sound. It is desirable is such cases to define new waveforms having more harmonics for use on low notes. Likewise, when very high notes are needed such as C5 and beyond, you need to define new waveforms having fewer harmonics to avoid alias distortion.

3.3.2 Minimizing Background Noise

Unlike ailas distortion, background noise is an inescapable result of round-off error in digital synthesis and can only be minimized. In the Simplified NOTRAN system, the most effective way to minimize background noise is to make sure that the maximum allowable amplitude is used in all waveforms. It is also helpful if the waveshapes themselves have a high ratio of peak amplitude to average (rms) amplitude. The latter condition is achieved if the phases of the harmonics are scrambled up (randomized) so that the crests of the individual simple sine waves are not likely to line up at one point. The reason this is important is that the WAVE statement adjusts the amplitude of the entire waveshape so that no peak exceeds the specified overall amplitude. Thus if there is one very high peak where the crests line up, the amplitude of the entire waveshape is reduced to bring the peak in range.

4. APPENDICES

4.1 MEMORY MAP

This is the memory map of the Simplified NOTRAN system at execution time, i.e., when a compiled song is actually playing.

$0700-$0EFF	Synthesizer-Player program, entry point = $0700
$0F00-$0FFF	Silent waveform
$1000-$13FF	4 default waveform tables, waveforms 1 - 4.
$1400-$1FFF	12 User defined waveforms, waveforms 5 - 16.
$2000-	The compiled score with the Commands Section starting at $2000. The highest usable address is $BDFF.

4.2 OBJECT CODE FORMAT

The object code generated by the Simplified NOTRAN compiler is subsequently interpreted by the Synthesizer-Player program to generate the actual sound. The object code is generated as a standard CODOS "loadable" file which can be loaded into memory with a GET command. Note that this is not executable machine code, it is merely data for the Synthesizer-Player program. The object bytes generated are also shown at the left side of the listing generated by the compiler. The entry point address associated with the object file is in fact the entry point of the Synthesizer-Player. Thus if the player program has already been loaded in memory, a previously compiled song may be played by merely typing its name.

4.2.1 Format of Commands Section Object Code

The object code is always loaded into memory starting at location $1C00. The Commands Section code comes first immediately followed by the Notes Section code. In general, each command compiles into a single "instruction" for the Synthesizer-Player. All except one of these instructions start with a "flag" byte of $FF followed by an "operation code" byte followed in turn by one or more "operand" bytes. Each OP code is summarized below:

FF 00 NN Set number of voices. NN is the number of voices, with $0E=1, $10=2, $12=3 or $14=4.

FF 01 V1 V2 V3 V4 Assign waveforms to voices. In each of the operand bytes is the memory page number that contains the waveform table to be played by the corresponding voice. Page $0B is silence and $0C through $1B are waveforms 1-16 respectively.

FF 02 TT Set tempo to TT. TT is in units of 103 microseconds. Note durations are in units of 1/256 note. The overall duration of a note is the product of its duration value and TT.

FF 03 WW HH AA PP Create a waveform table. WW is the memory page address to receive the table. Following may be any number of triples of bytes (HH AA PP) where each triple is a harmonic specification. HH is the harmonic number, AA is amplitude as a fraction between 0 and .996, and PP is phase in units of PI/128. The end of the instruction is designated by HH=0.

FF 06 AA Set peak amplitude for subsequent FF 03 instruction. AA/2 gives the absolute peak value the waveform will be normalized to. This may either be the positive peak or the negative peak depending on which is greater in magnitude

FF FF End of Commands Section, return to the operating system

HH LL Play a segment starting at address HH,LL where HH is the high address byte and LL is the low address byte. Note that natural order is used for the address rather than 6502 reversed order. Confusion with other opcodes is avoided since the memory at FFxx is used by the operating system and cannot hold song data. Note that the listing will show the segment ID number instead of the segment memory address. The correct address is automatically inserted into the object file after compilation however.

4.2.2 Format of Notes Section Object Code

The Notes Section object code is loaded into memory immediately following the Commands Section object code. Each segment in the notes section consists of a series of events. The NOTRAN compiler generates an event whenever one or more voices starts a note, ends a note, or changes pitch. Every note statement will produce at least one event and may produce several when note durations in the previous statement are different.

The format of an event is as follows:

DD V1 V2 V3 V4

where DD is the duration byte and V1 - V4 are pitch bytes for each of the voices respectively. If MAXVOICE is less than 4, there will only be that many pitch bytes. The synthesizer-player program knows how many pitch bytes to expect through execution of an FF 00 instruction in the Commands Section. The duration byte is a binary fraction representing the fraction of a whole note to be played. The pitch bytes hold a pitch code which is later looked up in a table to determine the actual note frequencies.

The end of a segment is designated by a duration byte of zero. A duration of 01 is also not allowed because it will trigger an action in the player program that is not supported by the compiler (or needed in a system with ample main memory such as the MTU-130). A pitch byte of zero represents silence. Voices that are not playing will automatically be given pitch bytes of zero.

Note that there is no difference in the object code generated by note statements A below and B below in simplified NOTRAN:

A		B	
1C4,1/2;	2E4,1/4	1C4,1/4;	2E4,1/4
2G4,1/4		1C4,1/4;	2G4,1/4

The more advanced music systems available for the MTU-130 do distinguish between these cases by giving the user control over the envelope of the tones as well as the waveform.

4.3 ERROR MESSAGES

A number of error conditions are possible while using the Simplified NOTRAN compiler. Listed below is each error that may be detected, system action in response to the error condition, and possible corrective action the user may take.

4.3.1 Initial User Dialog Errors

These errors may occur when giving the file names at the beginning of a NOTRAN run. Each allows re-entry of the erroneous information. If you wish to abort the compilation instead, hold the CTRL key down and type a C which will give the CODOS prompt.

DEVICE NAME NOT ALLOWED, TRY AGAIN. - When specifying the source or object file name you must specify a disk file because the compiler may have to "backup" in the file during the compilation process. Use a system utility to transfer the source statements from the device to a disk file and then run the compiler again.

ILLEGAL FILE OR DEVICE NAME, TRY AGAIN. - The file or device name given does not conform to the CODOS rules for such names. Check the spelling or refer to section 2 of the CODOS manual.

SPECIFIED DISK DRIVE NOT OPEN, TRY AGAIN. - The file name specified is on a disk drive which does not have a diskette inserted into it or which has not been opened with an OPEN command. Check the file name spelling or insert the needed diskette and OPEN the drive.

SPECIFIED FILE DOES NOT EXIST, TRY AGAIN. - The name given for the source file name cannot be found. Check the spelling and drive number.

SPECIFIED DISK DRIVE IS WRITE PROTECTED, TRY AGAIN. - The disk drive specified in the listing or object file name is write protected. Specify a file on a different drive or exit the compiler, affix a write permit sticker on the disk, and rerun the compiler.

SPECIFIED FILE IS LOCKED, TRY AGAIN. - An existing file name has been given for either the listing or the object file but it is write protected. Either give another file name or exit the compiler, UNLOCK the file name and then re-run the compiler.

SPECIFIED FILE ALREADY EXISTS, OK TO OVERWRITE? - An existing file name has been given for either the listing or the object file. Type a Y if you wish this file to be replaced with the results of this run. Type an N if you would like to specify a different file name.

4.3.2 Compilation Errors Printed on the Listing

These errors relate directly to the statements in the NOTRAN source file. If there is more than one error indication on a single line, pay special attention to the first one printed since the condition that caused it may have spilled over to cause the others.

NO END STATEMENT - The end of the source file has been reached without encountering an END statement. Use the editor to added the needed END statement and compile again.

ER 1 - INVALID KEYWORD - A keyword has been misspelled. The statement is ignored. Use the editor to correct the spelling and compile again.

ER 2 - INVALID NUMBER' - A digit or number could not be found where expected. The remainder of the statement (except WAVE) is ignored. Check the statement syntax or the keyword spelling.

ER 3 - INVALID DELIMITER - The character immediately following a digit or number is not legal. The remainder of the statement (except WAVE and note) is ignored. Remember that blanks may not be embedded within harmonic or note specifications.

ER 4 - NUMBER IS OUT OF RANGE - A number in the statement is either too small or too large. The statement is compiled with a default value substituted for the out of range value. Check the limitations that apply to the statement and correct.

ER 5 - INVALID TEMPO FRACTION - Either the numerator or denominator is greater than 255 or there are embedded blanks or the / is omitted. The statement is ignored.

ER 6 - INVALID TEMPO DURATION - The number following the equals has an illegal character or the = after the tempo fraction was not found. The statement is ignored. Remember that commas must not be embedded in the duration number.

ER 7 - TEMPO TOO SLOW - The tempo specified is slower than 6.6 seconds for a whole note. Use a faster tempo and double the durations of the notes. This can also be caused by overflow in the tempo calculation. Reduce the tempo fraction to lowest terms and try again.

ER 8 - TEMPO TOO FAST - The tempo specified is so fast that all significance is lost in the internally calculated value. This is usually caused by overflow in the tempo calculation. Reduce the tempo fraction to lowest terms and try again.

ER 9 - ILLEGAL WAVE ID - The ID number in the WAVE statement is zero or greater than 16 or has some other kind of error. The WAVE statement is ignored.

ER 10 - ILLEGAL OVERALL AMPLITUDE - The overall amplitude in the WAVE statement is greater than 255 or has some other kind of error. The statement is ignored.

ER 11 - ILLEGAL HARMONIC NUMBER - The harmonic number is zero or greater than 127 or has some other kind of error. The harmonic specification is ignored.

ER 12 - ILLEGAL HARMONIC AMPLITUDE - The harmonic amplitude is greater than 100 or has some other kind of error. An amplitude of zero is substituted.

ER 13 - ILLEGAL HARMONIC PHASE - The harmonic phase parameter has a syntax error or some kind. A zero phase angle is substituted.

ER 14 - ILLEGAL NOTES SEGMENT ID - The ID number of the SEGMENT statement has a syntax error such as an embedded comma. The statement is ignored.

ER 15 - NO NOTES SECTION BEFORE END - You have probably forgotten to place an ENDCMD statement at the end of the Commands Section. The compiler terminates normally but the object file will not be usable.

ER 16 - INVALID SEGMENT ID - Identical to ER 14.

ER 17 - DUPLICATE SEGMENT ID - The ID number specified in the SEGMENT statement has already been used in a prior SEGMENT statement. The statement is ignores.

ER 18 - WARNING - NOTES STILL SOUNDING AT END OF SEGMENT - When an ENDSEG statement is encountered, there were still some notes sounding from previous statements. The most likely cause is an error in the duration field of some prior note statement. The condition is ignored. When the piece is played, the offending notes will be truncated at the segment boundary.

ER 19 - INVALID KEYLETTER IN NOTE STATEMENT - A line with a blank or digit in column 1 was encountered but a note or a rest specification could be not be found. Most likely cause is a forgotten * in a comment statement, or not starting the note or rest specifications in column 5 or beyond. The statement is ignored.

ER 20 - INVALID CHARACTER IN REST SPECIFICATION - Most likely a comma missing between the R and the duration fraction. The rest specification is skipped.

ER 21 - INVALID DURATION SPECIFICATION - Any kind of error in the duration fraction and following dots if any. Remember that the final evaluated duration including dots must not be greater than a whole note.

ER 22 - VOICE NUMBER OUT OF RANGE - In a note specification, the specified voice number is zero or greater than 4 or has a syntax error. The note specification is skipped.

ER 23 - ILLEGAL PITCH SPECIFICATION - The pitch letter is not A-G or is not upper case or the final evaluated pitch including sharps and flats is below C1 or above C7 or there is some other kind of syntax error. The note specification is ignored but any others on the line are processed.

ER 24 - INVALID CHARACTER IN NOTE SPECIFICATION - An illegal modifier character was seen after the duration specification. Simplified NOTRAN does not have modifier characters so there must be either a blank or a semicolon following the duration. The note specification is ignored.

ER 25 - VOICE NUMBER GREATER THAN CURRENT MAXVOICE - Self explanatory.

ER 26 - VOICE STILL SOUNDING FROM PREVIOUS LINE(S) - A voice has been instructed to play a note before one started in a previous statement has finished. Most likely casue is an error in the duration in a previous note statement. The previous note is truncated and the specified one played.

ER 27 - ENDSEG WITHOUT MATCHING SEGMENT - An ENDSEG statement was encountered without previously having seen a SEGMENT statement. The statement is ignored.

ER 28 - MAXVOICE CHANGE INSIDE A SEGMENT - The maximum number of voices may only be changed between segments. The statement is ignored.

ER 29 - NOTES ENCOUNTERED OUTSIDE OF A SEGMENT - There was no SEGMENT statement before the note statements began. Can also be caused by not starting a command in column 1 in the Notes Section. The notes are compiled but will not be played since they cannot be addressed by a PLAY statement.

ER 30 - ONE OR MORE PARAMETERS MISSING - In an ASSIGN statement, not all 4 voices were assigned. The missing parameters are assumed to be zero.

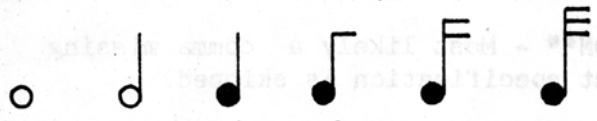
ER 31 - MORE THAN 1 NOTE PER VOICE - In a note statement, a voice is playing more than 1 note. The first note encountered for that voice is played.

UNDEFINED SEGMENT ID - XXXX - A PLAY statement in the Commands Section called for a segment ID (XXXX) that was never seen in a SEGMENT statement in the Notes Section. The address field of the compiled PLAY statement is <u>not</u> updated therefore the music will go wild when it reaches this PLAY statement.

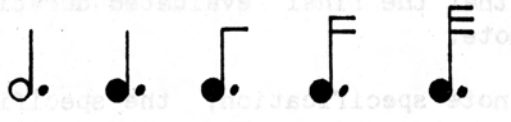
MEANINGS OF COMMON MUSICAL SYMBOLS

A musical score typically consists of notes and other symbols. Some scores use very few symbols and modifiers while others use a great deal of them. While it is impossible to fully explain music notation in a page, some of the more commonly used symbols are defined below. If the reader has no experience at all in reading music, a beginner's book or a friend should be consulted for background information.

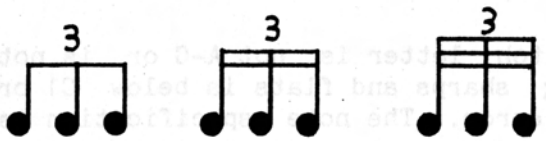
NOTE DURATIONS



1/1 1/2 1/4 1/8 1/16 1/32

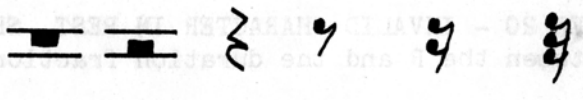


1/2. 1/4. 1/8. 1/16. 1/32.



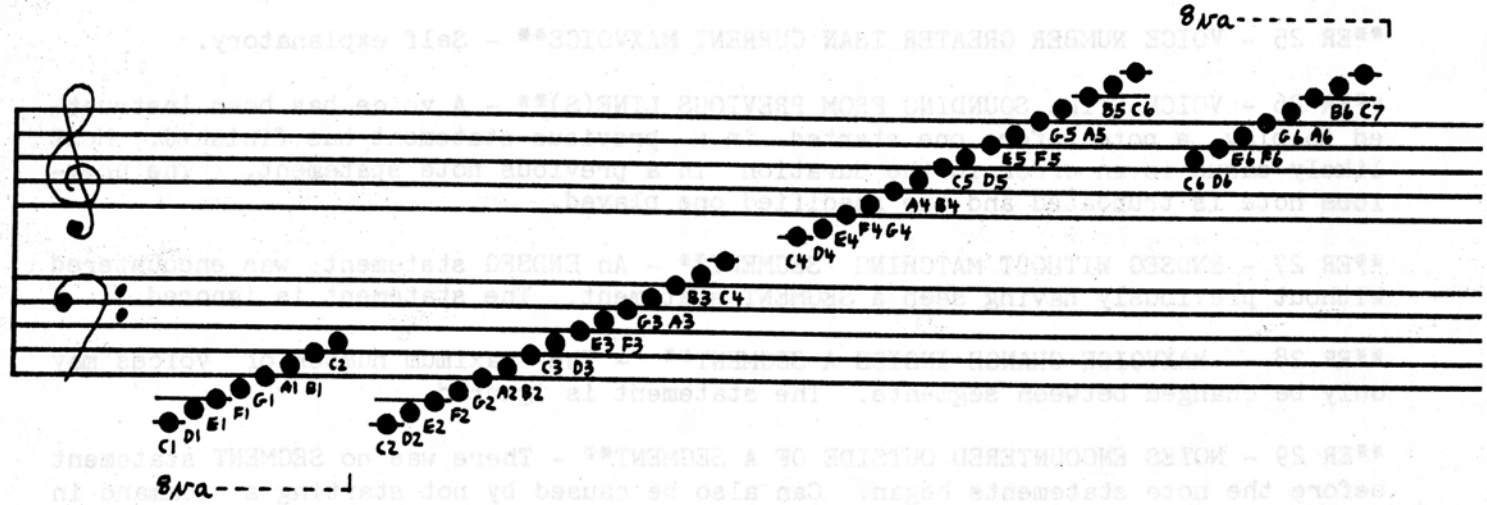
1/12 1/24 1/48

REST DURATIONS

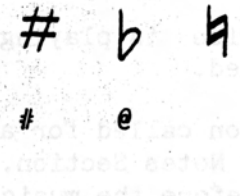


1/1 1/2 1/4 1/8 1/16 1/32

NOTE PITCHES



OTHER SYMBOLS

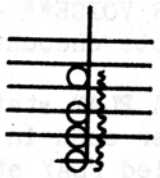


\# @

Sharp Flat Natural

1G4,1/16
R,1/16
1A4,1/16
R,1/16

Staccato



1C4,1/2; R,1/16
2E4,7/16; R,1/16
3G4,6/16; R,1/16
4C5,5/16

Arppegio

4.5 HIGHEST USABLE NOTE VS HARMONIC NUMBER

HIGHEST HARMONIC	HIGHEST NOTE (3500HZ LIMIT)	HIGHEST NOTE (4000HZ LIMIT)	HIGHEST HARMONIC	HIGHEST NOTE (3500HZ LIMIT)	HIGHEST NOTE (4000HZ LIMIT)
1	C7	C7	16	G#3	B3
2	G#6	B6	17	G3	A#3
3	C#6	E6	18	F#3	A3
4	G#5	B5	19	F3	G#3
5	F5	G5	20	F3	G3
6	C#5	E5	21	E3	F#3
7	B4	C#5	22	D#4	F3
8	G#4	B4	23	D3	F3
9	F#4	A4	24	C#3	E3
10	F4	G4	25	C#3	D#3
11	D#4	F4	26	C3	D3
12	C#4	E4	27	B2	D3
13	C4	D4	28	B2	C#3
14	B3	C#4	29	A#2	C3
15	A#3	C4	30	A#2	C3

4.6 ADDITIONAL REFERENCE MATERIAL

The field of computer syntheiszed music is an exciting and dynamic one with almost unlimited oppertunities for individuals to make significant contributions using personal computers such as the MTU-130. The following publications are suggested for further reading and study:

Books

1. Chamberlin, H. A. Musical Applications of Microprocessors. Rochelle Park, New Jersey: Hayden Book Co., 1980. (Available from MTU, K-1002-BOOK)

2. Mathews, Max V. The Technology of Computer Music. Cambridge, Massachusetts: MIT Press, 1969.

3. Morgan, C. P. The Byte Book of Computer Music. Peterborough, New Hampshire: Byte Publications, 1979.

4. Proceedings, Symposium on Small Computers in the Arts. IEEE Catalog No. 81CH1721-0, 1981.

Periodicals

1. Electronotes Newsletter, 1 Pheasant Lane, Ithaca, New York 14850.

2. Computer Music Journal, MIT Press, Cambridge, Massachusetts.

Articles

1. Chamberlin, H. A. "A Sampling of Techniques for Computer Performance of Music" Byte Magazine, September, 1977. (Available from MTU, K-1002-1ART)

2. Chamberlin, H. A. "Advanced Real-Time Music Synthesis Techniques." Byte Magazine, April, 1980. (Available from MTU, K-1002-6ART)