SENIOR PROJECT

HARMONIC SYNTHESIZER

by
Dale Timothy Neumann

Advisors:
Dr. Graham
Dr. Agbo

December 17, 1987

ACKNOWLEDGEMENTS

I would like to express my sincerest appreciation to Dr. Roger Messenger for allocating me enough time as necessary to do a thorough job on the project. I would also like to offer my gratitude to Dr. Nurgen Erdol, Dr. Peter Graham, and Dr. Samuel Agbo for their time and assistance throughout the course of this project. I would also like to thank Pauline Kartrude at Academic Computing for introducing me to PROCOMM and VAX Kermit with which I was able to establish the critical time-saving link between the VAX system and the XDS/320 Emulator. Finally, I wish to thank Robert Campbell for introducing me to the TMS32010 Development System and convincing me that it would be perfect for this type of project. Thanks, Bob, it was!

TABLE OF CONTENTS

INTRODUCTION . . . . p. 1

THEORY . . . . p. 2

HARDWARE . . . . p. 5

SOFTWARE . . . . p. 8

SYSTEM . . . . p. 13

COMMUNICATIONS . . . . P. 15

APPLICATION . . . . p. 18

CONCLUSION . . . . p. 25

LIST OF FIGURES . . . . P. 30

REFERENCES . . . . P. 31

APPENDIX . . . . Separate

INTRODUCTION

The purpose of my project was to design a harmonic synthesizer. By setting the amplitudes and phases of the harmonics the synthesizer was able to produce the desired real-time waveform.

The above was accomplished by selectively reading points from a high resolution sine table to reproduce the various harmonic frequencies at selected phases. Then, each harmonic data points could be scaled approximately, summed with the other harmonic data points of that time interval, and stored in a final waveform location. When all calculations are complete, the locations containing the final waveform could be outputted as a playback loop. These digital data values would then be converted to audible sound by passing them through a digital-to-analog converter.

THEORY

The theory behind the design of the harmonic synthesizer is the fact that time and frequency are interchangeable and carry complete information about the waveform. That is to say that a waveform can be studied in terms of its frequency components. Thus, a periodic waveshape can be created by setting its frequency components (1).

$$\mathcal{F}\{f(t)\} \Leftrightarrow F(w)$$

To have complete control of the harmonic components of the waveshape, it is necessary to create the waveform from specific frequency components. the purest and simplest waveshape is that of the sine wave (or cosine wave, a 90 degree phase shifted sine wave) which has only one frequency component as shown by the below graphs.

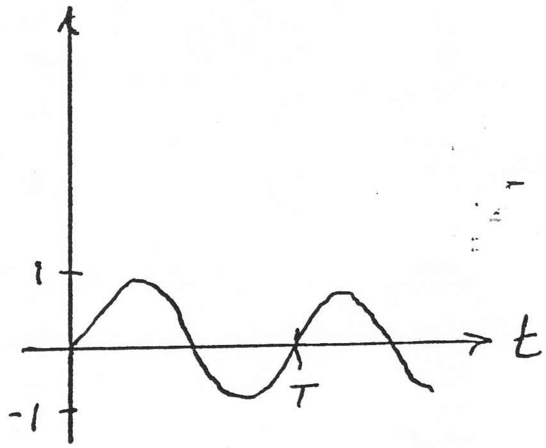


Fig. 1. Sinusoid in time domain

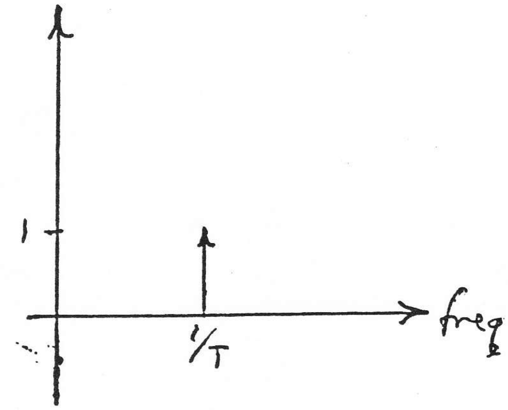


Fig. 2. Sine in freq. domain

It is possible to produce more complex periodic waveforms by adding sine waves of various amplitudes, periods, and phases. If frequencies are limited to harmonics of the fundamental and including the fundamental frequency, the wave form may be described by the below equation.

$$f(t) = A\cos(wt + \theta_1) + B\cos(2wt + \theta_2) + C\cos(3wt + \theta_3) + \ldots$$

To be able to create every conceivable periodic waveshape would require an infinite number of harmonics. This would obviously not be practical. Fortunately, the audible range for humans extends to only 20 KHz. In addition, harmonics higher than 11 above the fundamental have very little impact musically on a tone because of their relatively high frequency and their usually small amplitudes compared to the fundamental. Also many higher harmonics are not musical tones causing a detuning effect. Although these subtle harmonics can be used to fatten and color the sound, they will not be considered as major factors in the design of this synthesizer.

OVERTONE SERIES

Harmonics Generated by a Fundamental
(Great C*)

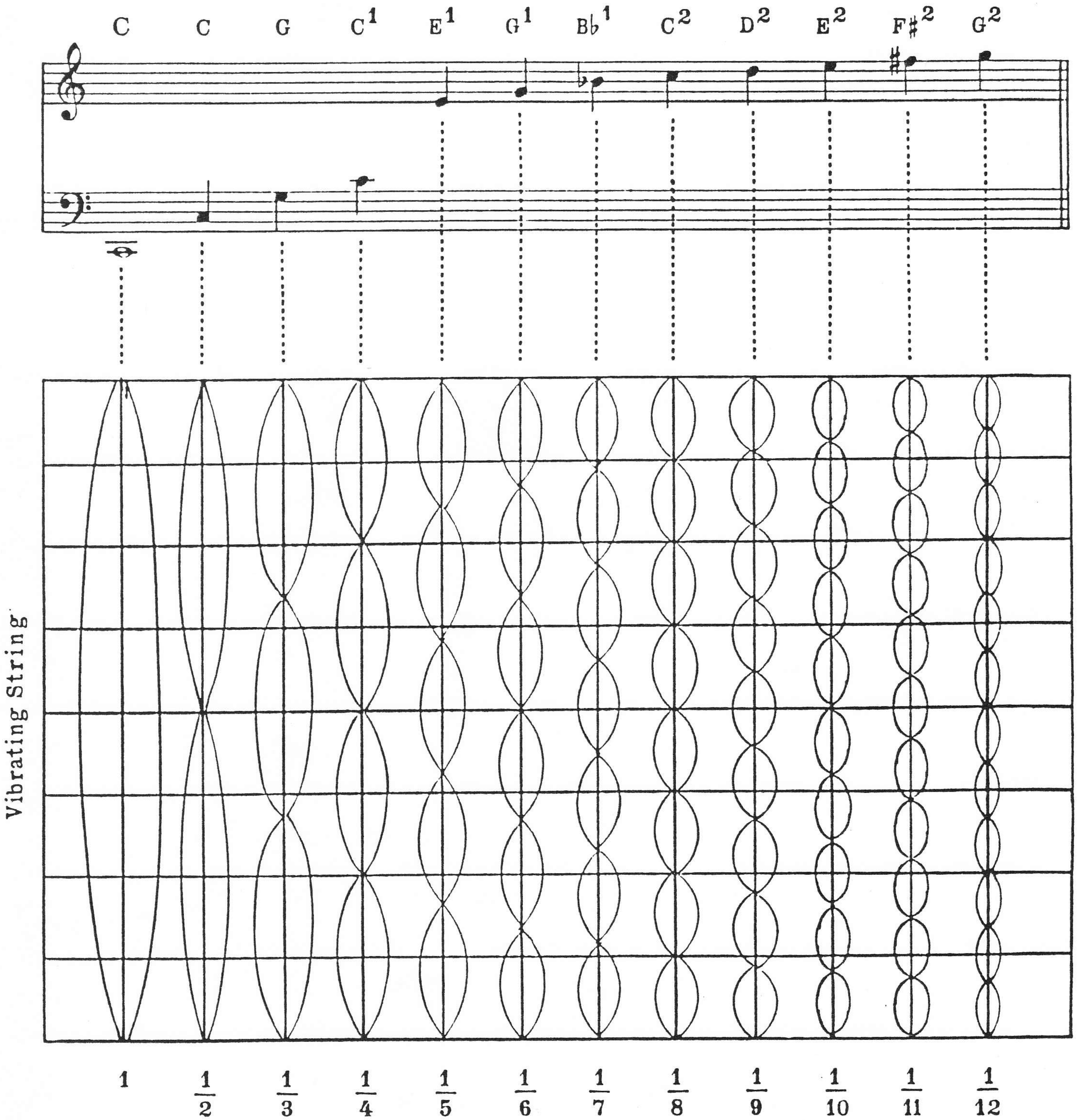


Fig. 3. Harmonics generated by fundamental (Lee, 1966).

HARDWARE

The TMS32010 Microprocessor is much different than typical microprocessors. There are several very notable traits that generally make this particular processor faster, more efficient, more flexible, more accurate, and more user friendly (3).

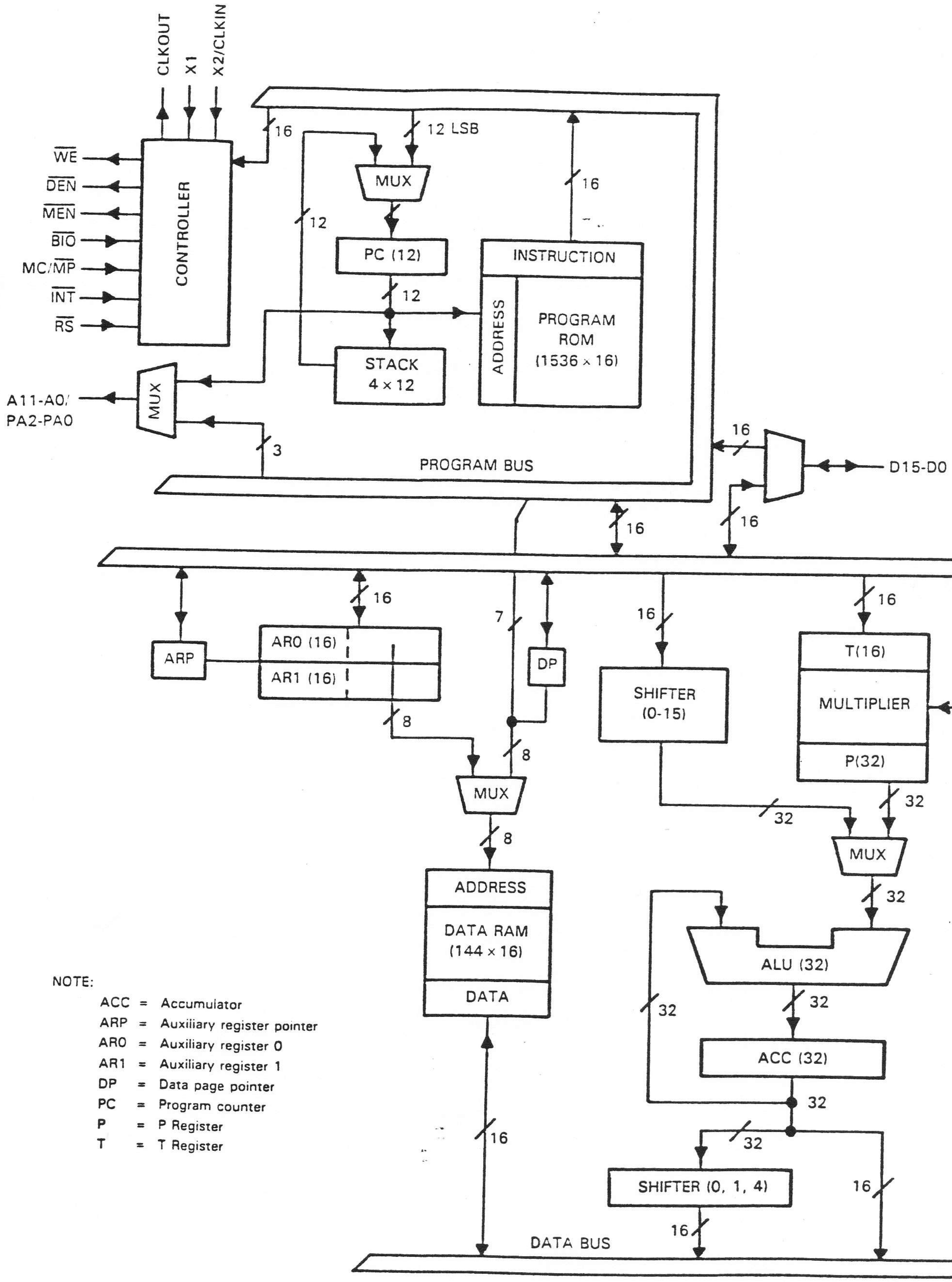
1. It is faster, having 200ns instruction cycles. And because most instructions require only one cycle to complete (Branch, Stack, and I/O require two {2} cycles; Table instructions require three {3}), the TMS32010 effectively is capable of executing five million instructions per second.

2. This speediness is largely due to the efficiency inherent in the architecture. The TMS32010 utilizes a Harvard architecture in which program memory and data memory lie in two separate spaces. Partitioning of memory allows an instruction to be prefetched while the previous instruction is executing.

3. In addition, the Harvard style was modified to allow transfer of data between program and data spaces (using Table instructions) increasing the flexibility of the device.

4. The TMS32010 is a 16-bit microprocessor with a double-precision 32-bit ALU/accumulator affording greater accuracy than typical microprocessors.

5. Finally, the TMS32010 is simpler to program, employing autoincrementing/decrementing registers for indirect addressing and loop counting and utilizing a single 200ns cycle multiplier which replaces slow and lengthy multiplication.



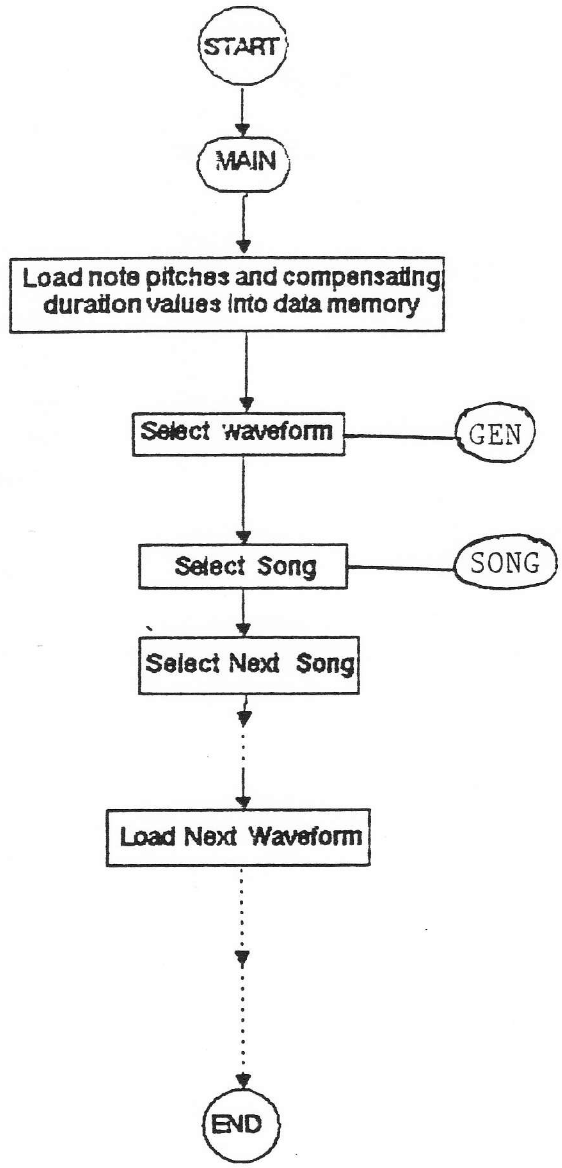
NOTE:

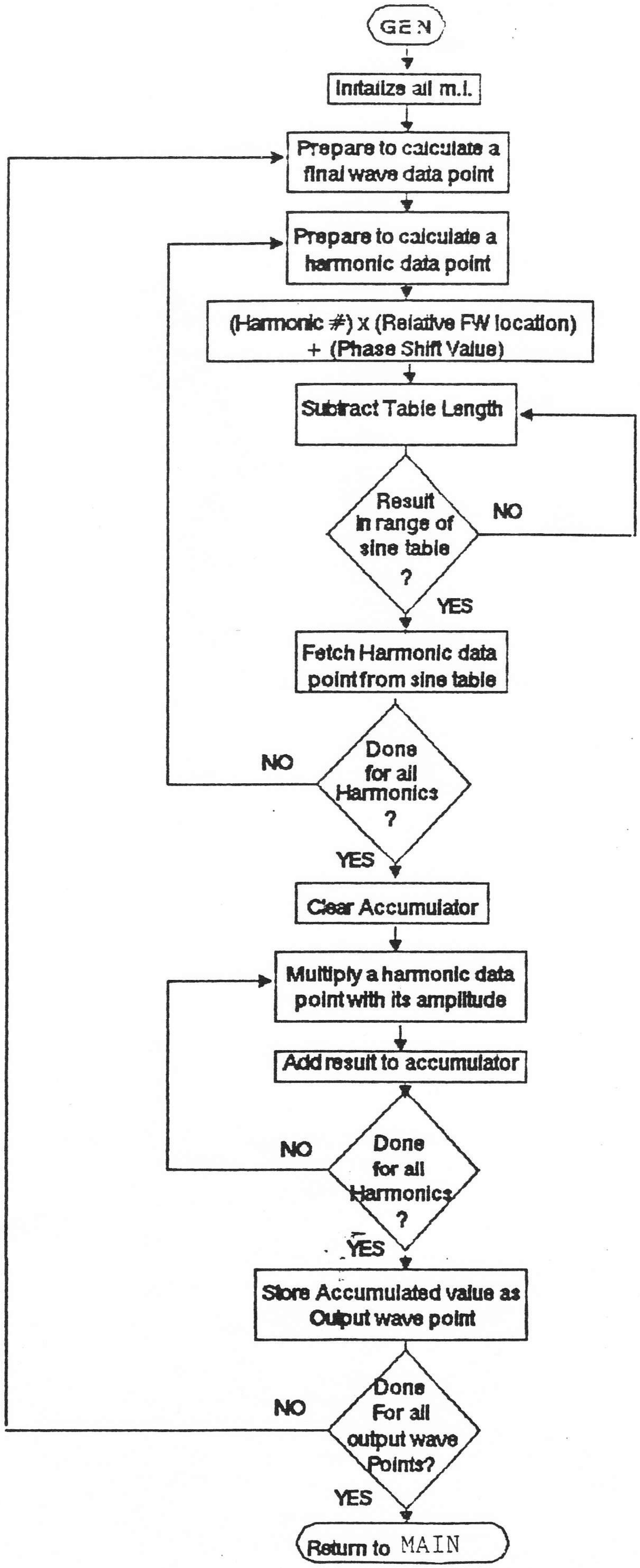
ACC = Accumulator
ARP = Auxiliary register pointer
AR0 = Auxiliary register 0
AR1 = Auxiliary register 1
DP = Data page pointer
PC = Program counter
P = P Register
T = T Register

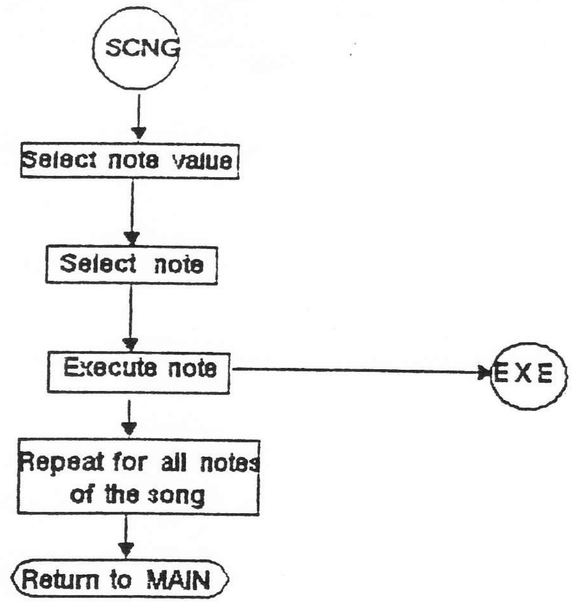
FIGURE 4 – BLOCK DIAGRAM OF THE TMS320M10

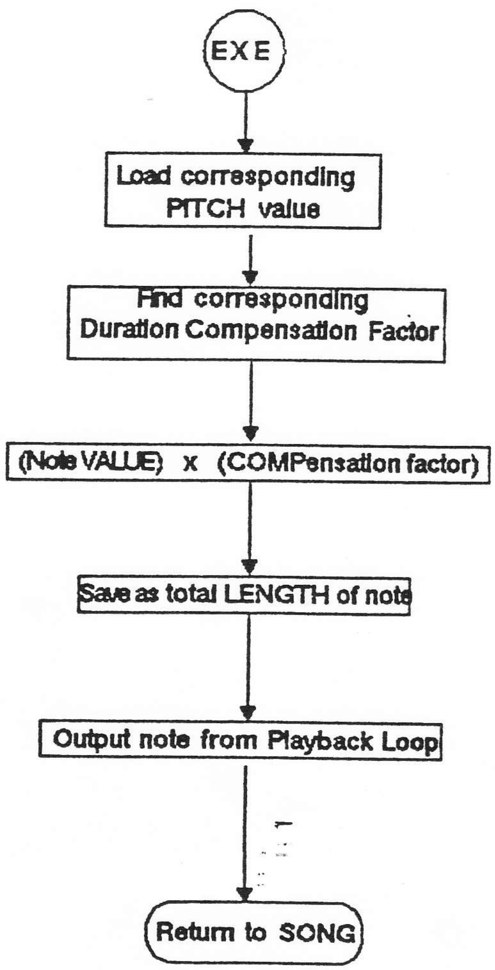
SOFTWARE

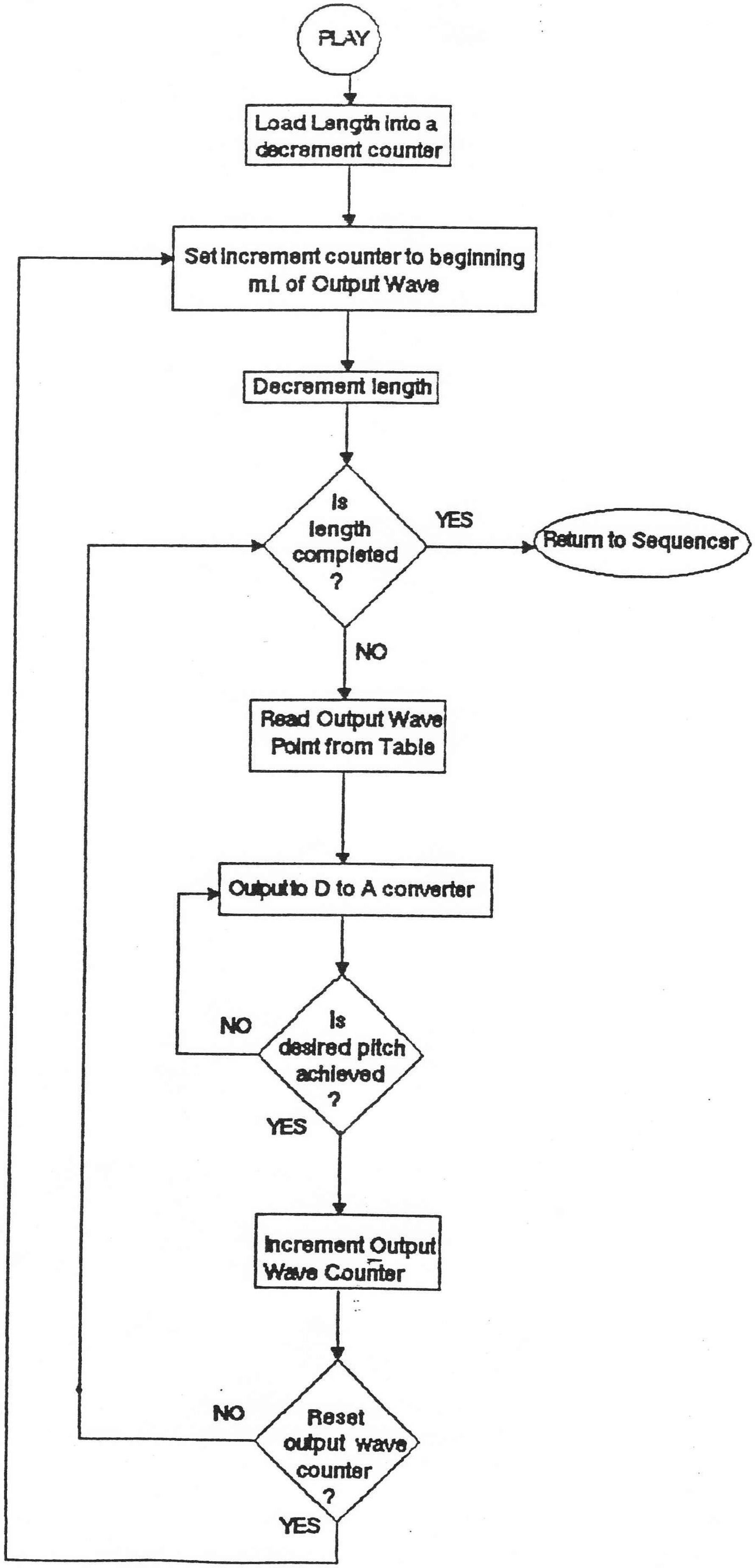
The following pages are the flow charts of the assembly language programming. The basic harmonic synthesizer is represented in the GENeration section and the PLAYback loop section. This rest of the flow charts show that the basic synthesizer can be incorporated into a sequencer program that can play various pitches and sounds with varied duration.


START
MAIN
Load note pitches and compensating duration values into data memory
Select waveform
GEN
Select Song
SONG
Select Next Song
Load Next Waveform
END


GEN
Initailze all m.I.
Prepare to calculate a final wave data point
Prepare to calculate a harmonic data point
(Harmonic #) x (Relative FW location) + (Phase Shift Value)
Subtract Table Length
Result in range of sine table ?
NO
YES
Fetch Harmonic data point from sine table
Done for all Harmonics ?
NO
YES
Clear Accumulator
Multiply a harmonic data point with its amplitude
Add result to accumulator
Done for all Harmonics ?
NO
YES
Store Accumulated value as Output wave point
Done For all output wave Points?
NO
YES
Return to MAIN


SCNG
Select note value
Select note
Execute note
EXE
Repeat for all notes of the song
Return to MAIN


EXE
Load corresponding PITCH value
Find corresponding Duration Compensation Factor
(Note VALUE) x (COMPensation factor)
Save as total LENGTH of note
Output note from Playback Loop
Return to SONG


PLAY
Load Length into a decrement counter
Set increment counter to beginning m.l. of Output Wave
Decrement length
Is length completed ?
YES
Return to Sequencer
NO
Read Output Wave Point from Table
Output to D to A converter
Is desired pitch achieved ?
NO
YES
Increment Output Wave Counter
Reset output wave counter ?
NO
YES

SYSTEM

The principles of this project were implemented using the TMS32010 Digital Signal Processor Development System by Texas Instruments. The system used included the VAX(VMS) XDS/320 Macro Assembler, Linker, and Simulator for software development, and included in the XDS/320 Emulator with Analog Interface Board (AIB) to integrate the hardware and software. The VAX(VMS) Simulator System allows the user to easily create and debug his software. The Simulator fully responds as the TMS32010 Microprocessor as the XDS/320 Emulator would. First the program is written to the Source File ({file name}.FIL) with the aid of VAX full screen editing. The Assembler then takes the Source File Program ({file name}.FIL) and produces both an assembled list ({file name}.FIL) complete with commands, code, error messages, and warning messages, and Mapped Object File ({file name}.MPO)

The assembled list file contains the programs, code, error, and warning messages. The Mapped Object File is loaded into the Simulator and may be loaded to the XDS Emulator with little modification. The Assembler, Linker, and Simulator commands are as follows:

To Assemble (Example file name - HS)

Type . . .

X320 <RETURN>

At prompt type . . .

HS.FIL <RETURN>

Hit <RETURN> for defaults.

Following Assembly, if there are errors in the module, type . . .

TYPE HS.LIS

The error free input file may now be linked using "LINK" command. In this particular project no other files needed to be linked, consequently the Mapped Object File, HS.MPO, is ready the be loaded into the Simulator.

COMMUNICATIONS

In the course of this project it was necessary to establish a link between the VAX(VMS) Simulator and the XDS/320 Emulator in order to facilitate and expedite movement of programming material. To accomplish this, a terminal was needed that would interface the two systems. In addition, because the emulator is not able to retain the program when shut off,it was desireable to be able to save the program in some manner.

The IBM Personal Computer was chosen as the communications link. Florida Atlantic University has IBM-PC's connected directly to VAX. With a commuications software called PROCOMM the Mapped Object File created by the Assembler can be downloaded to the IBM using a rapid file transferring tool of VAX called Kermit. The transferred file is written directly to disk, and the resulting file requires little modification of the object code to become compatible with the XDS.

To make the object code file compatible three steps must be taken in line edit (EDLIN) mode of IBM DOS:

1. Delete Kermit control character (^J) at very beginning of file.
2. Restore Mapped Object File Identification Characters to original state and insure that there are alotted eight characters for this identification. If some

are needed fill them with the <space bar> character.

3. Replace the check sum character number seven (7) (located as the sixth character from the right of the first line) with the ignore check sum character number eight (8).

Sample first line of code:

Before editing...

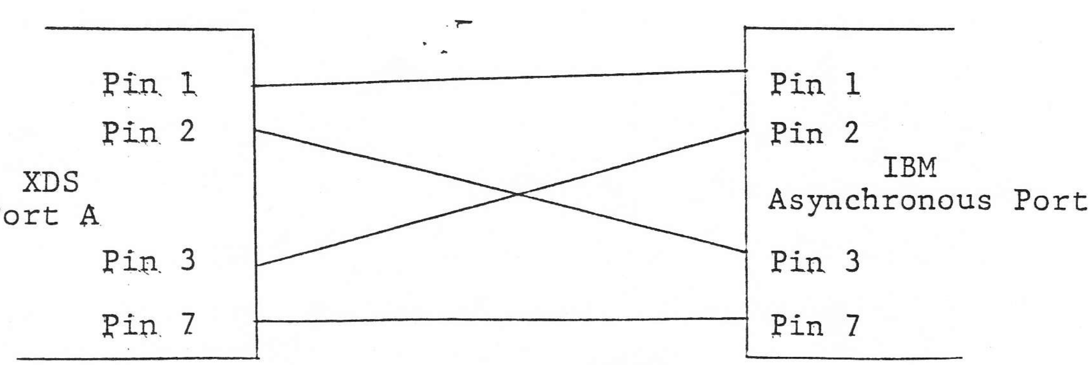
```
^JK0000HARM0A 90000B7001B3065B..............7F185FF   [Ignore
                                                        ident.]
```

After editing...

```
K0000HARMON  90000B7001B3065B..............8F185FF   [Ignore
                                                       ident.]
```

two spaces here

At this point an IBM must be connected to the XDS and operated as a terminal. Again, PROCOMM was used as the communications software. A cable was then constructed to connect the male RS232 Port connector of the IBM to the female connector on the XDS. The cable required only for lines. Earth and ground pins 1 and 7 were connected across in parallel fashion between the IBM and the XDS; however, the transmit and receive pins 2 and 3 are cross-connected (4).



Next, PROCOMM was initialized to match the line parameters required by the XDS. Optimum settings are as follows:

1. Bit Rate : 4800 bps
2. Parity : Even
3. Character Length : 7 bits
4. # of Stop Bits : 2 bits
5. Duplex : Full

With the IBM connected to Port A of the XDS, and the XDS operating in user mode, programs on disk were able to be sent from the IBM to the XDS by the below procedure:

1. Enter the XDS download (DL) command.
2. Default all but final prompt.
3. Enter "1" for user mode at final prompt.
4. Enter upload command <PgUp> from IBM PROCOMM
5. Enter name of mapped object file to be sent and <RETURN>

APPLICATION

The gap between the digital world and the analog world was bridged by the use of the TMS32010 Analog Interface Board (AIB). Of the many features offered by the board, the following were utilized for this project «4»:

1. 12-bit digital-to-analog converter
2. Two low pass filters: (1) an anti-aliasing filter and (2) a reconstruction filter, both factory set with a cutoff frequency of 4.7KHz

With the AIB connected to the XDS by a target cable, harmonic amplitudes and phases of several basic waveforms were loaded into their respective data memory addresses, the program was run and the results noted. The hexadecimal values of the harmonics of these waveforms were calculated from Fourier series expansion formulas «5» using programs self-written in IBM BASIC.

With the basic program working properly, options were added successfully: (1) control of fundamental frequency and (2) control of the duration of the note. An attempt was also made to combine the synthesizer with a sequencer to allow different sounds to be loaded automatically and also pitches and their durations to be output in succession without stopping the program, but this was not completed at the time of this report.

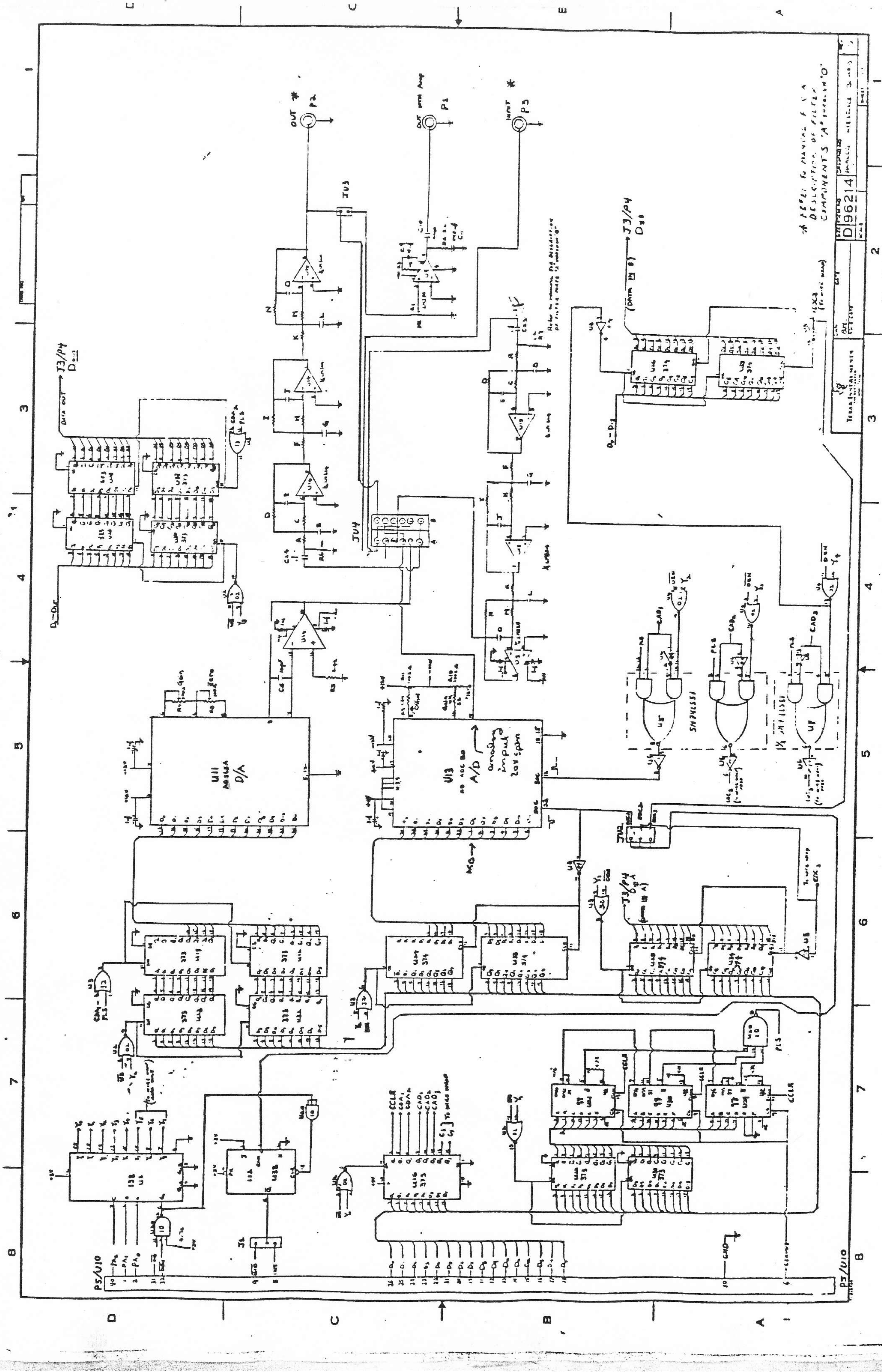
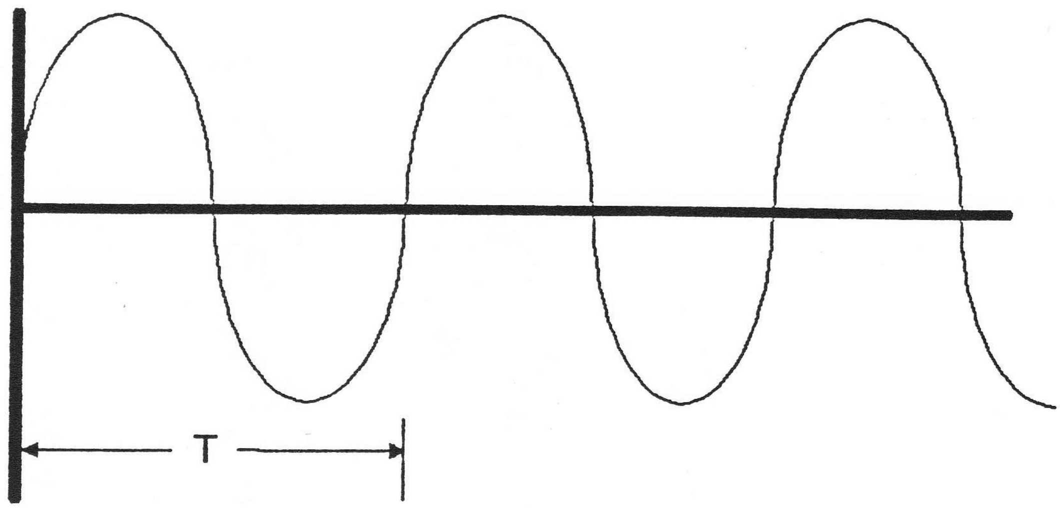


Figure. 6. Schematic of the TMS32010 Analog Interface Board.

SINE WAVE

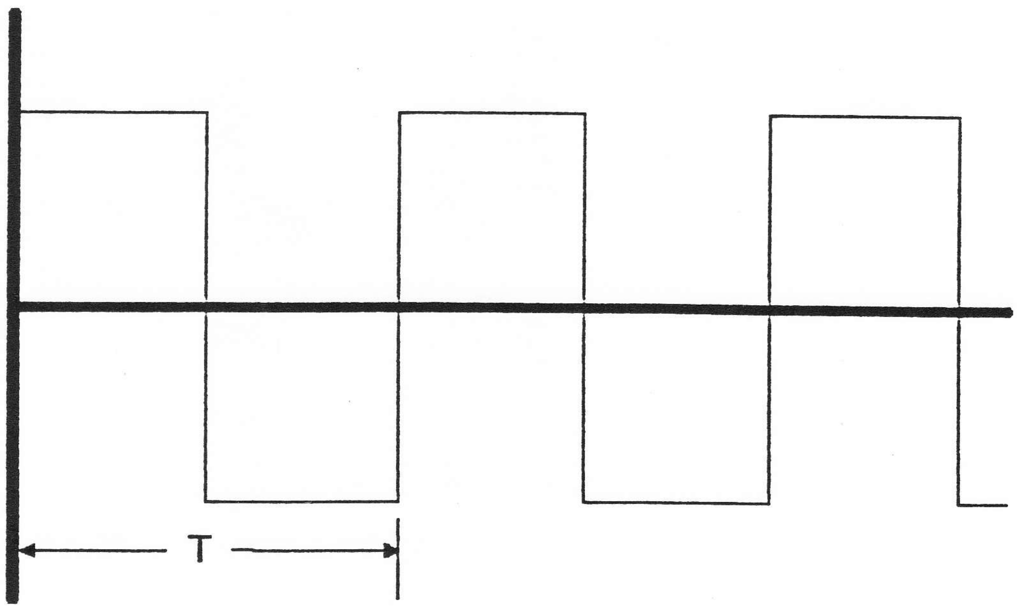


For Harmonic (n = 1) only

Amplitude = 1
Phase = Not Critical

Fig. 7. General equations for harmonics of a sine wave.

SQUARE WAVE

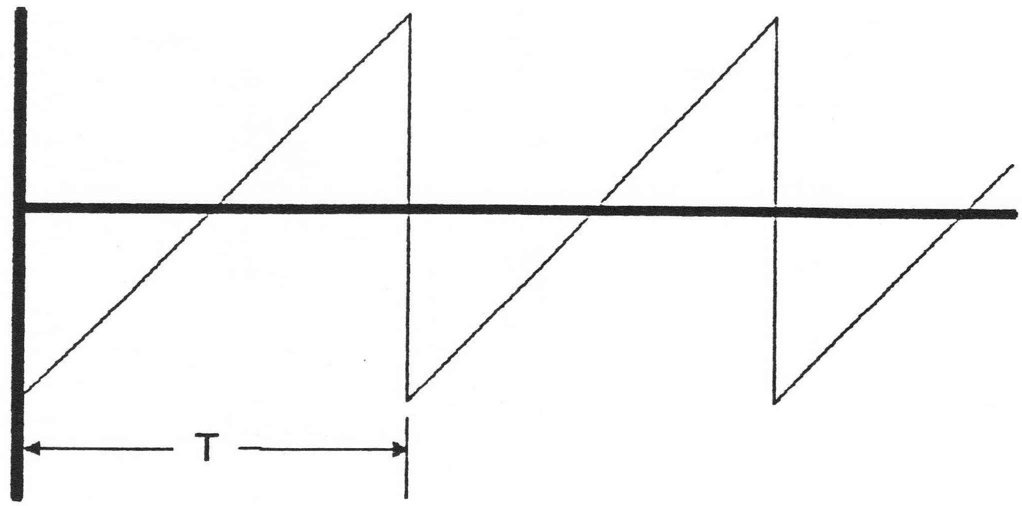


For Harmonic n (odd only)

Amplitude = 1/n
Phase = 0^0

Fig. 8. General equations for harmonics of a square wave.

SAWTOOTH WAVE

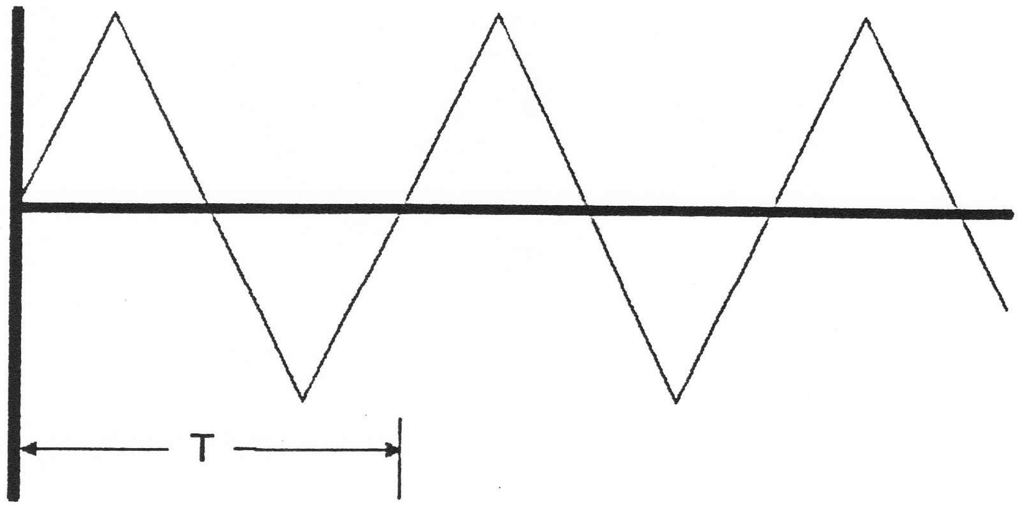


For Harmonic n

Amplitude = 1/n
Phase = 180° (n = even only)

Fig. 9. General equations for harmonics of a sawtooth wave.

TRIANGULAR WAVE

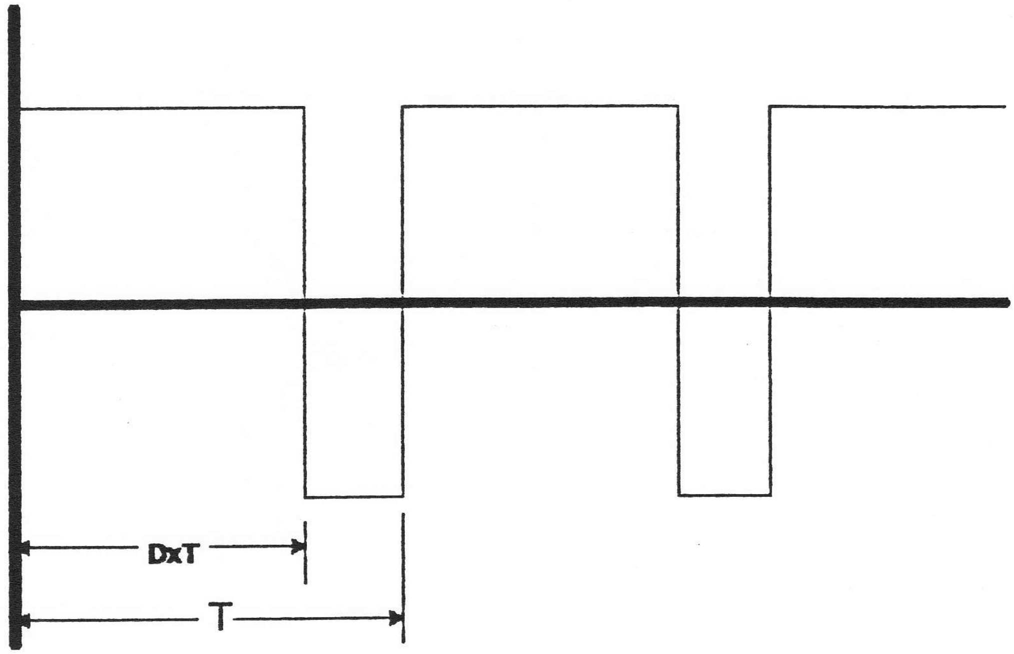


For Harmonic n (odd only)

Amplitude = $1/n^2$
Phase = 0°

Fig. 10. General equations for harmonics of a triangular wave.

PULSE WAVE
(Duty Cycle = D x 100%)



For Harmonic n

Amplitude = sin(D x pi x n) / n
Phase = 0°

Fig.11. General equations for harmonics of a pulse wave.

CONCLUSION

The TMS32010 microcomputer is an exceptionally fast and accurate device because of the need to precisely produce and output signals in real time. Assembly language programming was fairly straight forward. Assembly commands were easy to learn and use. Debugging brought some occasional difficulties when there were errors due to improper indirect addressing or when there was confusion about which auxiliary register is being incremented or decremented at a given point in the program.

A great deal of time was spent trying to avoid spending even greater time moving the simulation program to the XDS Emulator. Using a stand-alone terminal would have required numerous retyping of the program into the XDS as the terminal is unable to save the program at the end of the day's session. Consequently, the XDS was accessed using the IBM-PC with the communicatons software PROCOMM allowing safe storage of the program onto disk to be reloaded at the next session and allowing simulation versions from VAX to be transferred to disk using VAX Kermit and, again, PROCOMM.

Connecting the IBM-PC to the XDS posed some confusion as it was unclear in the XDS manual as to whether it was necessary to retain the stand-alone Espirit terminal cross-connected to Port A of the XDS and use the PC as the host computer at Port D or to use the IBM alone. Then with the IBM alone, it was unclear as to whether a "smart" terminal should be also cross-connected and as to which port it should be accessing, Port A or D. In addition, it was unclear as to how the program should be loaded into the XDS with the appropriate communications configuration. Finally, there was a problem uploading the program to the XDS because the check sum byte at the end of the first line of the mapped object code was detecting an error in the number of bits in that line because the Kermit transfer from VAX had reinterpreted a few nonessential characters. This problem was resolved by restoring the first line to its original condition and by changing the check sum character to one that ignores the sum.

Specifications for the D/A converter were inaccurate in the version of the AIB user's guide that I ordered from Texas Instruments. The D/A requires that the most significant bit be reversed (effectively removing the effect of the sign bit) to be interpreted properly by the D/A. The D/A produced two glitches in the output wave per period at the same absolute voltage level (one at +V and one at -V. Upon recheck of the sine table and playback loop determining, this cannot be a problem due to the playback loop or a mistyped sine table;therefore, it must be concluded that there is a glitch in the digital to analog converter.

Calculation of PITCH values and COMPensating duration values shows a trade off in accuracy: As frequency decreases the PITCH values become more accurate but COMP values decrease in accuracy, likewise as frequency increases notes move far from their proper respective frequencies but a steady beat becomes more easily attainable. This trade off can be attributed to the fact that at high frequencies PITCH values become very low rounded off integers and at low frequencies COMP values are very low rounded off integers. Consequenty, there is a bandwidth for which there is reasonable accuracy fo both note pitch and length. This bandwidth is dependent upon the speed limitation of the microprocessor and the efficiency at which the final wave points are outputted. A faster speed allows for larger more accurate integer values, hence greater bandwidth.

Faster microprocessor speed would also allow for the following:

1. A larger sine table making it possible to add further harmonics without sacrificing accuracy of the output and to decrease at very low fundamental frequencies the harmonic distortion caused from a slower sampling rate.
2. An increase in the high frequency range of the fundamental frequency without sacrificing the number of harmonics.
3. Real-time output of the final waveform giving rise to the ability to instantaneously change the waveform as as any harmonic is altered rather than reloading the harmonic data and recalculating the new waveform.

The disadvantages of additive harmonic synthesis are as follows:

1. Several harmonics are required to accurately reproduce any waveshape. Waveforms with particularly sharp edges are of particular notice because a discontinuous function cannot be exactly duplicated by summing

continuous functions.

2. Numerous harmonics require heavy repetitive calculation demanding rapid signal processing.

The advantages of additive harmonic synthesis are the following:

1. Fast, efficient, specially designed digital signal processors are making additive harmonic synthesis a practical, feasable, and refined method of creating waveforms.
2. There is virtual total control over tonal quality of sound because every frequency component of the waveform is adjustable.
3. Numerous waveshapes that would normally require several analog generators can be produced by one single "generator", the harmonic synthesizer.
4. It facilitates creating a particular sound because harmonics build a "picture" of what is heard. It is the "language" of the ear more than the eye. With harmonics one need not be familiar with what types of basic waves produce certain sounds.

This particular synthesizer designed with the TMS32010 Digital Signal Processor has the potential to be additionally programmed to load its own waveform harmonic data, change the frequency of the fundamental, and set the duration of a particular tone to be outputted. These possibilities allow for a sequence of notes to be played with any sound generated from the values of the harmonics. In addition, the fundamental frequency could be determined by a musical keyboard. Polyphony could be introduced with paralleled coprocessors each reading a section of memory containing the generated final waveform. Some voices could be partitioned to output a different final waveform allowing for a a split keyboard. This project is the foundation for a wide range of possibilities.

LIST OF FIGURES

Figure 1. Sinusoid in time domain . . . . p. 2

Figure 2. Sinusoid in frequency domain. . . . . p. 2

Figure 3. Harmonics generated by a fundamental. . . . . p. 4

Figure 4. Block Program of the TMS32010 M/O . . . . p. 7

Figure 5 Communication cable connection from
IBM-PC to XDS Emulator. . . . . p. 16

Figure 6 Schematic diagram of Analog Interface Board . . . p. 19

Figure 7. General equations for harmonics of a
sine wave . . . . p. 20

Figure 8. General equations for harmonics of a
square wave . . . . p. 21

Figure 9. General equations for harmonics of a
sawtooth wave . . . . p. 22

Figure 10. General equations for harmonics of a
triangular wave . . . . p. 23

Figure 11. General equations for harmonics of a
pulse wave . . . . p. 24

REFERENCES

(1) Hiat, William and Jack Kemmerly. Engineering Circuit Analysis. New York: McGraw-Hill, 1978

(2) Lee, Dr. William F., Music Theory Dictionary. New York: Charles Hanson, 1966

(3) Texas Instruments. TMS32010 User's Guide. Texas Instruments, 1983

(4) Texas Instruments. TMS32010 Analog Interface Board User's Guide. Texas Instruments, 1984

(5) Fundamental Formulas of Physics, Vol. 1. Ed. Donald H. Menzel. New York: Dover Publications, 1960

JOURNAL

JUNE	Use TMS32010 Simulator and Emulator Development System. Output emulator to a D/A and an audio amplifier. The rest is programming! Studying TMS32010 Assembly Language.
JULY	Programming TMS32010 using VAX Assembler, Linker, Simulator.
AUGUST	Debugging source file. Problem traced to Multiply Loop Sign bit not being suppressed in Low Accumulator. Play loop not usable in simulation. Decide to output simulation in "real time". Harmonic not being properly calculated because Sine Table was not calculated to consider the highest bit being a sign bit and it was D.C. biased because I was anticipating the effect of the D to A converter. Hence, the table is adjusted to remove D.C. Bias and utilizes highest bit as sign bit. In addition, it is scaled down to prevent an overflow should too many harmonics have very high amplitudes.

SEPTEMBER Searching for program Cross-Talk to enable IBM to communicate with XDS. If this is accomplished I will be able to save my programs on disk. Because of the large amount of time I would save with this feature I am determined to make this link work. Program Cross-Talk is unavailable. Engineering Lab has Asynch. Comm. Program. Esprit Terminal works fine, but there is no means to save my work. IBM Asynchromous Port is hooked up in Host Port of XDS standard with uncrossed RS232 cable. No success.

Terminal cable is used at the host connection to see if the host can communicate alone through Port D and still nothing. I am confused as to whether it is the fault of the communications program, the cable configuration or the port configuration. I seek help from TI-TMS32010 Hotline. I am informed that the XDS has definitely been used with IBM and a communications program called PROCOMM. At this time I have become familiar with this program for its usefulness as a means of downloading files from VAX to an IBM-PC. This would be how I get my program from VAX to an IBM-PC. The problem remains in the IBM-XDS link.

OCTOBER I try PROCOMM with every reasonable combination to no avail. I check the configuration of the IBM port to see if it is voltage or current loop. It checks out as voltage. I create a cable for the host (IBM) that matches the terminal cable: Cross-connected for pins two (2) and three (3). I set the terminal for Host Mode and this enables the IBM to type characters to the Esprit display. Attempts to transfer program to the XDS are unsuccessful. The data is transferred to the screen of the Esprit but not to the program memory.

NOVEMBER I call TI-320 Hotline, again. This operator says that the IBM need not be used as a host but can be used directly as a user terminal at Port A. With the latter set up, the program can be tranferred using the down load command set for user transfer. This is successful except for a check sum error in the first line of code. Additionally, the first few program commands were not transferred. A follow up call to TI lead to the suffestion that I should check the mapped object file to see that the first line starts with correct characters and that I might change the "check sum" character to an "ignore check sum" character. Following his advice, the program is transferred successfully.

The next step is to successfully hook up the Analog Interface Board. Having not been able to obtain a copy of an AIB user's manual, I call TI locally to see if there are any available. I am Informed that there are none available at that location and that they would check with other TI locations. I am still able to hook up the board based on the general information in other TMS32010 Manuals and based on Robert Campbell's autocorrelation project. Eventually, I obtain a copy of the AIB user's guide from TI. To power the board, I attach female banana plug connectors to the power leads of the board and mount and wire banana plug terminals on an incompleted +12 V to ±12 V power supply that shall be used to power the AIB.

With the entire system in operation, I run the software to discover that the output resembles a square wave not the test sine wave. Certain of the validity of the software, I write a short program to test the D/A. This "ramp" function program leads me to conclude that the D/A is operating properly. The sine wave is output again with higher amplitude and yields "humps" in the square wave output. I realize that the D/A is not accepting the sign bit properly.

Negative values are being read as positive voltage values while positive values are being interpreted as negative voltage values. Solution: invert sign bit raising positive values to the positive voltage range and lowering negative values to the negative voltage range.

DECEMBER The project is now working acceptably. An attempt is made to yse the audio amplifier on the AIB to boost the output signal. It is determined that signal is entering the op-amp, but is distorted. This distortion varies greatly with the potentiometer setting at the input to the op-amp. There is no output from the amplifier at all. Fortunately, the filtered D/A output is sufficient to drive my home stereo with no undesireable effects. Finally, I thought that it would be nice to have the synthesizer play more than one single note at a time. And since Christmas is upon us, I wished to have the synthesizer play a Christmas carol. I begin to write a sequencing program in the hopes that it will be ready in time for the project presentation or at least by Christmas.