SOFTWARE FOR A REAL-TIME DIGITAL SIGNAL PROCESSOR

Lawrence M.L.Casserley

Royal College of Music, London

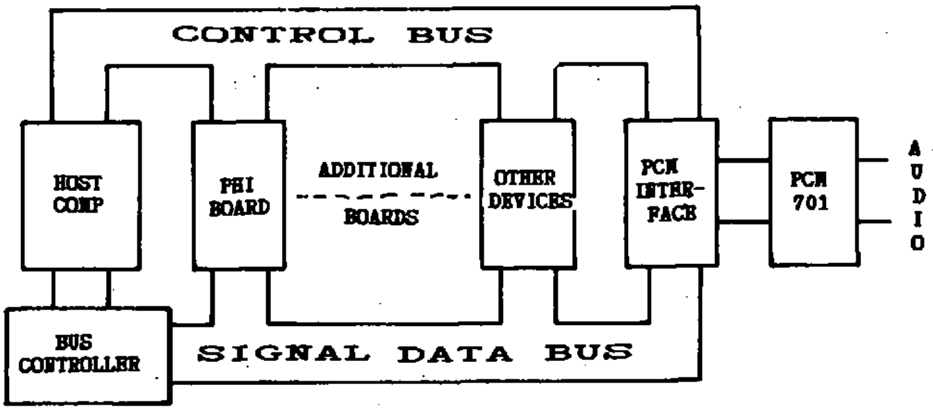
INTRODUCTION

A low-cost programmable digital signal processor has been developed, which enables real-time audio transformations to be effected. The board can be interfaced to a PCM recorder and a microcomputer to provide an economical signal processing package with applications in sound recording, electro-acoustic music, acoustic research, etc. A brief description of the system is given, followed by a discussion of some practical software for the processor. The paper also illustrates how these program modules may be combined to form practical devices.

PHI HARDWARE OVERVIEW

A block diagram of the system, known as SERIES PHI, is given in Figure 1. One or more PHI processor boards are connected to two independent busses. The first, the Control Bus, is used to download programs and control data from the host computer. The second, the Signal Data Bus carries audio samples between PHI processors, AtoD and DtoA converters and other devices, eg long delay memory. At present an interface card to a Sony PCM-701 Digital Audio Processor provides two channels of 16 bit AtoD and DtoA conversion as well as direct access to and from digital tape. Other types of conversion could, however, be used. With this interface the system is locked to the 44.1KHz sampling rate of the PCM.

Figure 1. Block Diagram of PHI system.



A block diagram of the PHI processor board is shown in figure 2. The CPU is a Texas TMS32010 16/32 bit Signal Processor chip. Two separate memories are used. The first appears as ROM to the 32010, but can be written into by the host computer. This is used for storing programs and wave tables and occupies the normal address space of the 32010. The second memory is accessed via two of the 32010´s I/O ports and is used for passing dynamic parameters and for storing delayed samples beyond what can be stored in the 32010´s internal memory. On the prototype approximately 100ms of delay is available. Longer delays will be handled off board by a separate delay memory system being developed. The use of both these memories and a number of other features of the PHI Processor Board will be demonstrated in the following discussion.

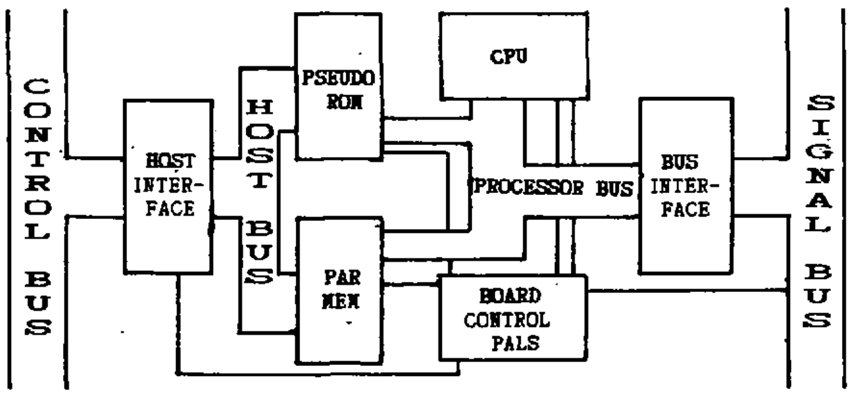
PHI SOFTWARE EXAMPLE

Figure 3 gives the assembler listing for an example program, MODUL1. This illustrates a typical electro-acoustic music application. The signal from an AtoD converter is low-pass filtered and ring-modulated with a sine wave, then reverberation is added before sending the result to a DtoA converter.

The first two lines of the listing define the function as a series of linked macros, which are expanded below. The figures in the left-hand column are instruction cycles. The 5 MHz instruction rate of the 32010 allows 112 instruction cycles per sample, and the assembler automatically keeps a running count of instruction cycles used.

Instruction 003 loads the input from Port 0; the assembler will generate a link map for the Signal Data Bus which will route ADC0 to Port 0. The symbols ´OP´ and ´IP´ are reserved in the PHI assembler to indicate the outputs and inputs of macro ´devices´ which are linked according to the function definition at the beginning of the code.

Figure 2. Block Diagram of PHI Processor Board



Low Pass Filter

Instruction 005 begins the code for a third order Chebyshev low pass filter[1]. First, a block of data is defined to hold the delayed samples. This ensures that they are in the correct order for most efficient implementation by the 32010. Next the coefficients are defined as parameters. The code illustrates use of the 32010´s pipelining facilities. The instruction ´LTD´ loads a new data word into the multiplier, accumulates the previous result and shifts the data to the next address in memory (hence the necessity to specify a precise data structure). Thus a filter of this kind can be implemented in two instruction periods per term plus three (zero accumulator to start, accumulate and store the final result to finish), a total of 17 in this case. Six such filter segments could be implemented on one PHI board.

Sine Wave Generator

The next segment of code beginning with the label SGEN implements a basic generator. The version shown assumes a half cycle sine (or other) table is stored in main memory. This is adressed by the lower twelve bits of the the accumulator during the ´TBLR´ (table read) instruction. The current phase angle and phase increment are stored as double word variables allowing 28 bit resolution in the phase calculations. The code at 035 determines which half of the cycle is current and the final four instructions allow amplitude control. This generator is somewhat more demanding of processor time, requiring 24 instruction periods, but could be simplified for less demanding applications by using only 16 bit precision and a whole cycle wave table.

Ring Modulator

By contrast this is extremely simple, in fact identical with the level controller at the end of the generator segment. In the former case it acts as an amplitude modulator as long as the ´LEVEL´ parameter is constrained to be a positive number. It also depends on the assumption that both inputs have no offset component.

Reverberation

The reverberation algorithm utilises a tapped delay line in the upper part of parameter memory. The number of taps is selected to fill the remaining processor time, in this case three are possible. The quality of reverberation is therefore somewhat dependent on what else is being done, but, in a programmable system, these options, like the choice of generator complexity, are made available to the user.

The function is implemented in four sections. First the new table addresses are calculated. Note that program iterations are used in preference to a subroutine which only wastes time in this context. Next the data transfers occur. Note that the board architecture uses two I/O ports named PARCAPT and PARREL writing to PARCAPT loads a new address and "captures" the memory, preventing a host write from interfering with the transfer. Reading from or writing to PARREL transfers data and "releases" the memory into control of the host once again. Reading from PARCAPT, however, also transfers data but keeps the memory "captured" so that a write to the same address can follow immediately. In this way the new sample is written in the same position as the final tap. Next the delayed samples are weighted and accumulated in the same manner as in the low pass filter. Finally the reverberation product and the direct signal are mixed and sent to the DAC.

The last part of the code implements a counter in one of the 32010´s Auxiliary registers, which is used to control the transfer of parameters at a rate of one per sample. This code is included in all programs automatically by the PHI assembler.

HOST SOFTWARE

PHIASM is an interactive editor/assembler which is designed to make the development of programs for the PHI Processor relatively straightforward. Since the code is necessarily short, it keeps both source and object files in memory and flags potential errors as they are entered. It deals with all memory allocations and automatically generates link maps for the Signal Data Bus and Parameter Memory for use by the run-time controller. It also keeps track of the number of instruction periods in use and pads unused time with NOPs.

PHIPER is a run-time system which sets up the performance environment, downloads programs and parameter data and sets up the Signal Data Bus link map. It is designed to allow the interfacing of a variety of control devices suitable to different applications. Other program generation and run-time control software is planned.

CONCLUSION

The PHI system has been developed over a number of years for the author´s own use as a performing musician. While the 112 instructions/sample rate of the PHI system presents some restrictions, it can be seen from the above discussion that such a device can implement viable and useful functions. The use of more than one board in a system quickly increases the sophistication of the possible sound transformations. Above all, its programmability makes the system capable of implementing many different functions, so that one unit can be equivalent to many individual devices.

[1]. I am indebted to Per Hartmann for assistance with the mathematics of the Chebyshev filter.

SOFTWARE FOR A REAL-TIME DIGITAL SIGNAL PROCESSOR

Figure 3. Example Program Listing - MODUL1

```
000     &FMODUL1        ADC0>CHLPF>RMOD(1)>REV4>DAC0
                        SGEN>RMOD(2),

        &X

000     PARLOAD:        IN      *,PREL          ;input parameter
002                     EINT

003     ADC0:           IN      OP,PORT0        ;get sample

005     CHLPF:
        &D              IPM3                    ;define data block
                        IPM2
                        IPM1
                        IP
                        OPM2
                        OPM1
                        OP
        &P              COEF1                   ;define parameters
                        COEF2
                        COEF3
                        COEF4
                        COEF5
                        COEF6
                        COEF7

005                     ZAC                     ;filter algorithm
006                     LT      IPM3
007                     MPY     COEF4
008                     LTD     IPM2
009                     MPY     COEF3
010                     LTD     IPM1
011                     MPY     COEF2
012                     LTD     IP
013                     MPY     COEF1
014                     LTA     OPM2
015                     MPY     COEF7
016                     LTD     OPM1
017                     MPY     COEF6
018                     LTD     OP
019                     MPY     COEF5
020                     APAC
021                     SACH    OP,1

022     SGEN:
        &P              &DPHINC                 ;define parameters
                        LIMIT
                        BASE
                        LEVEL
```

SOFTWARE FOR A REAL-TIME DIGITAL SIGNAL PROCESSOR

```
022                 ZALH    PHANG           ;update phase angle
023                 ADDS    PHANG + 1
024                 ADDH    PHINC
025                 ADDS    PHINC + 1
026                 SACH    PHANG,0
027                 SACL    PHANG + 1
028                 LAC     PHANG
029                 AND     LIMIT           ;prevent overflow
030                 SACL    PHANG
031                 OR      BASE            ;must be pwr of 2
032                 TBLR    SAMPLE
035                 LAC     PHANG           ;which half cycle?
036                 SUB     BASE
037                 BLZ     SCALE
039                 ZAC                     ;negate sample
040                 SUB     SAMPLE
041                 SACL    SAMPLE
042     SCALE:      LT      SAMPLE          ;level control
043                 MPY     LEVEL           ;must be positive
044                 PAC
045                 SACH    OP,1

046     RMOD:       LT      IP(1)           ;4 quadrant mult
047                 MPY     IP(2)
048                 PAC
049                 SACH    OP,1

050     REV4:
        &P          DEL1                    ;define parameters
                    DEL2
                    DELEND
                    TABLEND                 ;end of storage
                    OFFSET                  ;start of storage
                    TESTER                  ;equals offset + 1
                    DIRLEVEL
                    COEF1
                    COEF2
                    COEF3
                    COEF4

050                 LAC     DEL1,0          ;advance pointer
051                 SUB     TESTER,0
052                 BGEZ    RESULT1
054                 ADD     TABLEND,0
055     RESULT1:    ADD     OFFSET,0
056                 SACL    DEL1
057                 LAC     DEL2,0          ;rpt for each tap
058                 SUB     TESTER,0
059                 BGEZ    RESULT2
061                 ADD     TABLEND,0
062     RESULT2:    ADD     OFFSET,0
063                 SACL    DEL2
```

SOFTWARE FOR A REAL-TIME DIGITAL SIGNAL PROCESSOR

```
064                    LAC     DELEND,0
065                    SUB     TESTER,0
064                    BGEZ    RESULT3
066                    ADD     TABLEND,0
067     RESULT3:       ADD     OFFSET,0
068                    SACL    DELEND
069                    OUT     DEL1, PARCAPT   ;get data
071                    IN      DDAT1, PARREL
073                    OUT     DEL2, PARCAPT
075                    IN      DDAT2, PARREL
077                    OUT     DELEND, PARCAPT
079                    IN      DDAT3, PARCAPT
081                    OUT     IP, PARREL
083                    ZAC                     ;accumulate delays
084                    LT      DDAT1
085                    MPY     COEF1
086                    LTA     DDAT2
087                    MPY     COEF2
088                    LTA     DDAT3
089                    MPY     COEF3
090                    LTA     DDAT4
091                    MPY     COEF4
092                    LTA     IP              ;mix input
093                    MPY     DIRLEVEL
094                    APAC
095                    SACH    OP,1

096     DAC0:          OUT     IP,PORT0        ;result to DAC

097     PADNOP:                                ;pad with NOPs

109     PARADDR:               SAR     AR0, PAD;AR0 points to
110                    OUT     PARCAPT, PAD    ;next parameter
111                    BANZ    END
112                    LAR     AR0, NOPARS
113     END:           NOP
```