Computer Music on the NeXT Computer™

Julius Smith

IEEE WASPAA
October, 1989

DSP Hardware:

- Motorola DSP56001 clocked at 25MHz
- Memory-mapped Host Interface
- DMA to/from Host Interface (2-5MBytes/sec)
- 8K 24-bit words of zero-wait-state private static RAM
- D-15 connector: SSI and SCI serial ports of the DSP

DSP Software

Computer Music

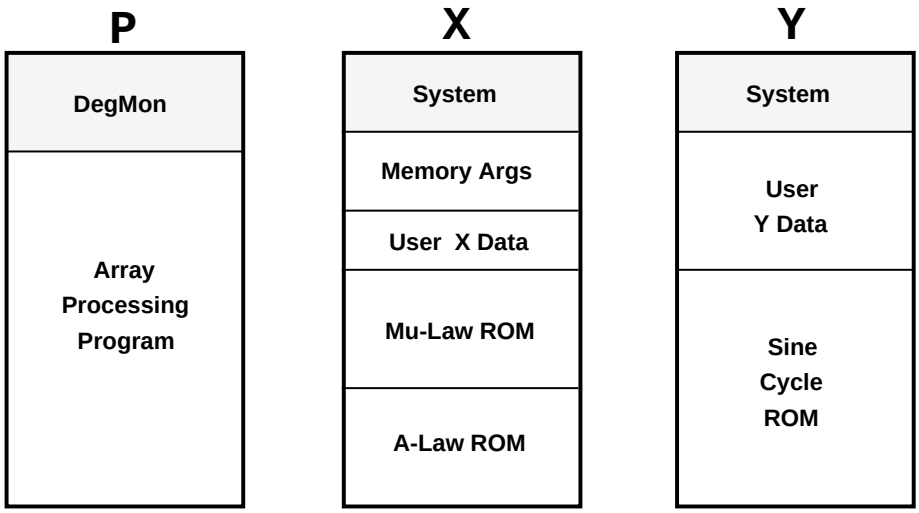
- Real Time Signal Processing
- Short Fixed Vector Size
- Real Sampled Data Streams (needing DMA buffers)
- Contiguous Signal Vectors
- Timed Message Support (needing message buffer)
- Untimed and ‘‘Timed 0’’ Messages

Array Processing

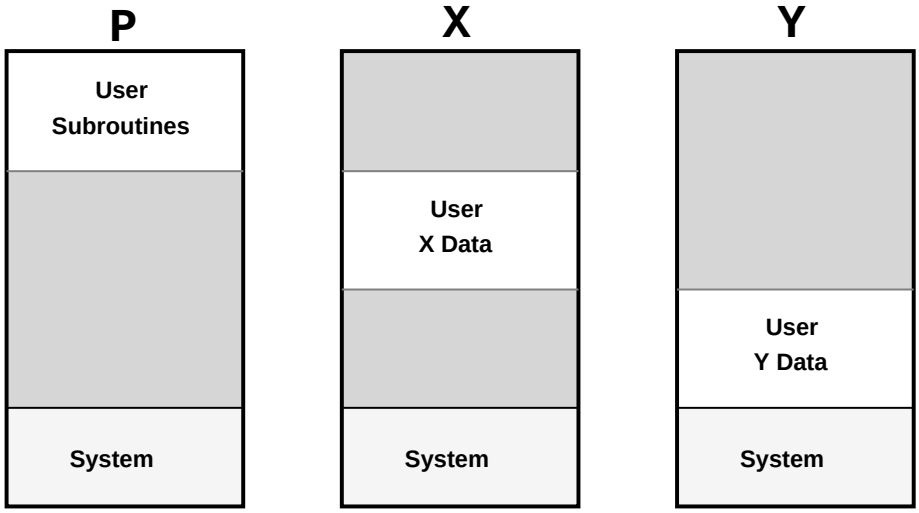
- Non-Real Time
- Arbitrary Vector (Array) Size

- Real or Complex Vectors or Matrices
- Matrix Access = Vector Access using ‘‘Skip Factor’’
- Complex Access = Real Access using ‘‘Skip Factor’’
- Simple Load and Go Protocol

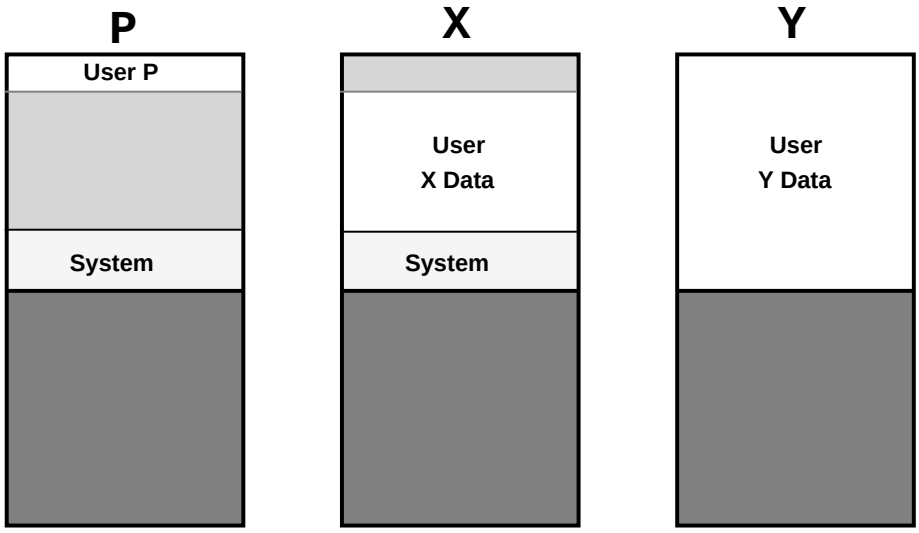
Array Processing Internal DSP Memory Map



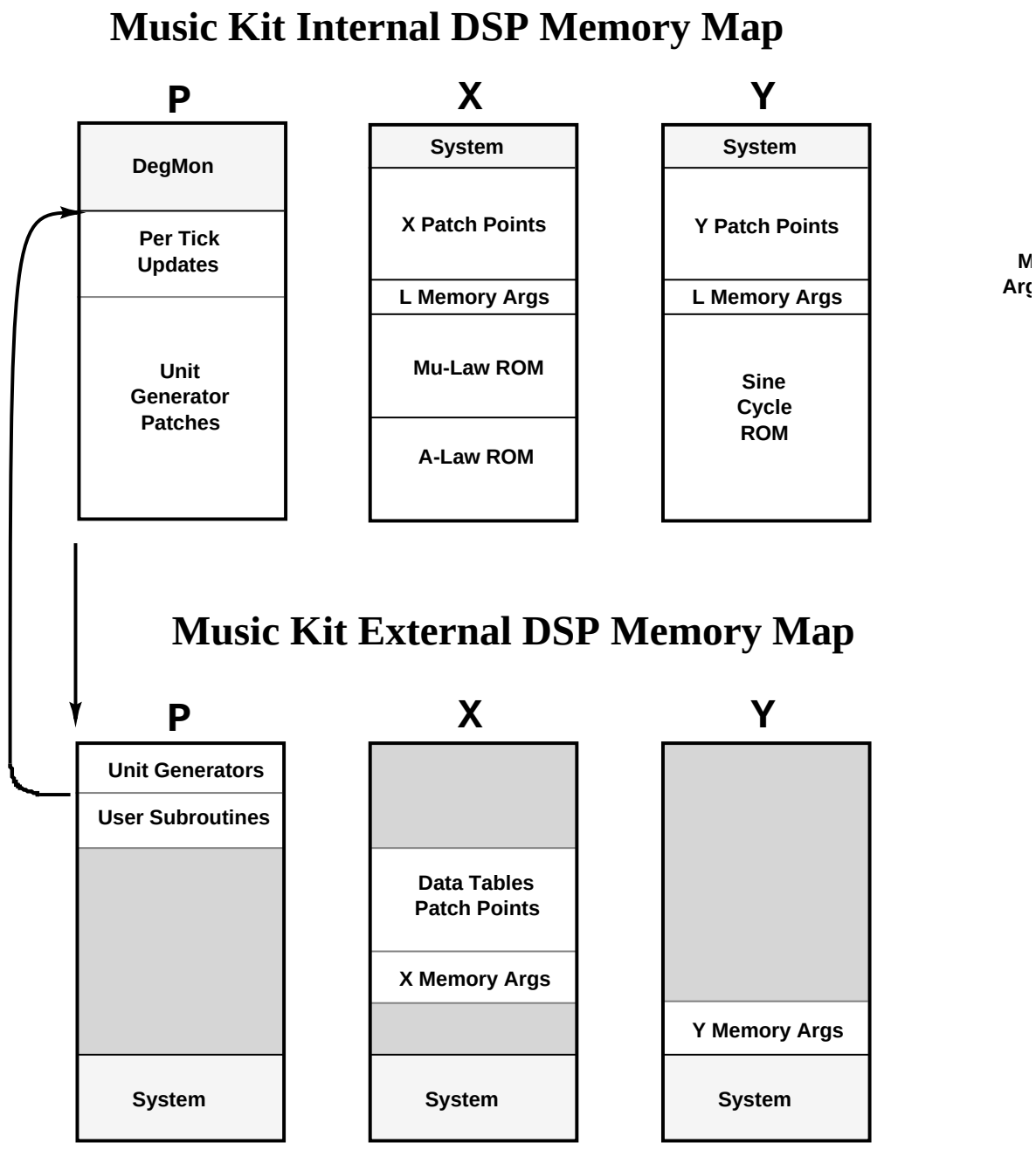
Array Processing External DSP Memory Map 1



Array Processing External Memory Map 2



- **X and Y physically separate**
- **P overlays X**
- **Memory halved in size**



Orchestra Loop Structure

The Orchestra Loop is a DSP sound synthesis program.
The loop executes once for each "tick" of output sound.
A "tick" is currently 16 samples.
All timed messages occur on a tick boundary.
The Orchestra Loop has the following structure:

System Updates

Switch DMA Buffers & issue DMA request, if necessary
Execute *all timed messages* having time-stamp ≤ "now"
Add 16 to 48-bit sample counter ("now")
Reset the 3 memory argument pointers (**x**, **y**, and **l**)

Sound Synthesis

Unit Generator 1
• • •
Unit Generator *n*

Possibly Leap to Off-Chip Memory
Unit Generator *n*+1
• • •
Unit Generator *m*

Go To System Updates

Example Orchestra Program

```
;; DSP56001 Assembly Language for DSP Orchestra Loop

  include 'music_macros'
  beg_orch 'test'
    new_yib yvec,16,0              ; Allocate y patch point
    beg_orcl                       ; Begin orchestra loop
      unoise p,1,y,yvec,0          ; Noise unit generator
      out2sum p,1,y,yvec,0.5,0.5   ; Panning stereo output
    end_orcl                       ; End orchestra loop
  end_orch 'test'                  ; End test program
```

- Actual stand-alone test program
- Used only for unit generator development

- Music Kit manages the equivalent program *dynamicall*y, driven by score performance needs.

Unit Generator Structure

- Each Unit Generator is a section of DSP synthesis code.
- Unit-Generator *macros* expanded *in line* (no subroutines).
- Signals are passed as pointers to 16-sample "patch points."
- Tables and delay-lines typically a multiple of 16 samples.
- Parameter update rate is sampling-rate / 16.

Each Unit Generator operates as follows:

Load ALU Registers from Memory Arguments

Move input/output signal pointers to R*n* address regs.
Move coefficients and state (e.g. filters) to Data ALU.
Access arguments using auto-increment addressing.

Execute Tick Loop

Perform **DO** loop for 16 samples (no loop overhead).
Access signals with auto-increment addressing.

Save State to Memory Arguments

Save run-time state (e.g. oscillator phase, filter delays).
Leave memory arg pointers set for next unit generator.

DSP56001 Index Register Assignments

R_X = **R0** = *x memory argument pointer*
R_Y = **R4** = *y memory argument pointer*
R_L = **R2** = *l memory argument pointer*

R_I1 = **R1** = *input 1 pointer* (unrestricted)
R_I2 = **R5** = *input 2 pointer* (unrestricted)
R_O = **R6** = *output pointer* (unrestricted)

R_HMS = **R3** = *host message stack pointer*
R_DMA = **R7** = *DMA transfer pointer*

Analogous names are used for the **N** and **M** registers, e.g., **N_X** = **N0** and **M_X** = **M0**.

Unit Generator Memory Argument Access

```
move x:(R_X)+,R_I1      ; Input 1 address to R_I1
move y:(R_Y)+,R_I2      ; Input 2 address to R_I2
move y:(R_Y)+,R_O       ; Output address to R_O

move x:(R_L)+,A         ; Long datum to A

move x:(R_X)+,X0 y:(R_Y)+,Y0  ; Load data to X0 and Y0
```

Notes

- Pointer loads can only happen one at a time.
- Two data loads may be possible in parallel.
- Long loads are always done using two parallel data moves.

A DSP Unit Generator Example - Add2

```
;; DSP56001 assembly language for the add2 unit generator.
;; The output signal is the sum of the input signals 1 and 2.
;; The macro is invoked with args i1spc, i2spc, and ospc,
;; which expand to 'x' or 'y' to specify the memory space of
;;  input 1, input 2, and the output signals.  I_NTICK is 16,
;; and add2_pfx expands to a globally unique string.

move y:(R_Y)+,R_I2          ; input 2 address to R_I2
move y:(R_Y)+,R_O           ; output address to R_O
move x:(R_X)+,R_I1          ; input 1 address to R_I1
move i2spc:(R_I2)+,A        ; load input 2 to A
move i1spc:(R_I1)+,X0       ; load input 1 to X0
```

```
   do #I_NTICK,add2_pfx\ tickloop     ; enter do loop
      add X0,A  i1spc:(R_I1)+,X0      ; add and fetch input 1
      if "i1spc"=='x'&&"i2spc"=='y'                ; xy{x,y}
         if "ospc"=='x'                          ; xyx
            move A,sout:(R_O)+ i2spc:(R_I2)+,A ; fast!
         else                                    ; xyy
            move A,sout:(R_O)+                   ; slower
            move i2spc:(R_I2)+,A
         endif
      else
         move A,sout:(R_O)+
         move i2spc:(R_I2)+,A
      endif
add2_pfx\ tickloop
```

Examples of Existing Unit Generator Macros

Asymp
One segment of a piecewise exponential envelope.

SlpDur
Piecewise linear envelope from specification of the slope and duration of each segment.

Biquad
Two-pole, two-zero filter section.

Delay
Digital delay line.

Dswitch
Delayed switch (for topology switch on a sample).

Oscgafi
Oscillator with **g**eneral address mask, and envelopes on **a**mplitude and **f**requency. The wave table is **i**nterpolated.

Scl1Add2
Add scaler times first input signal to the second.

There are many additional oscillator, mixer, and filter variations, and various signal generators (noise, im-pulses, etc.). Some have tick-based versions computing 1 rather than 16 values.

Use of DSP56001 Host Flags

- **HF0** – Tell DSP to abort current program
- **HF1** – Tell DSP requested DMA transfer is pending
- **HF2** – Tell host DSP is busy executing last message
- **HF3** – Tell host Timed Message Queue is full
  or array-processing function still running
- **HF2 & HF3** – Tell host DSP is in debugger

Interrupt Priority Levels Used in the DSP

0 – Default (used by all unit generators and most updates)

1 – Host (DMA word, message interrupts, host commands)

2 – Serial port interrupts (SSI serial port sound in or out)

3 – Critical sections and non-maskable interrupt service

Automatic C and Documentation Generation

Each array-processing or unit-generator macro source file has leading comments resembling a UNIX "man page." These comments contain information for automatically generating
three types of on-line documentation, and C software which which "wraps" each macro, marrying it to the host software environment.

The comment fields are used by program **dspwrap** to generate:

- A file containing a *one-line summary* for each macro
- A file containing *calling-sequences* for each macro
- A file containing a "*man page*" for each macro
- An invoking *C function* for each array processing macro
- A *prototype master* Objective C class for each unit generator
- *Leaf classes* in Objective C for each unit-generator variation
  (There's a variation for each input/output space combination)

Source-Code Comment Fields

Documentation Generation:

NAME
One-line description of the macro's function.

SYNOPSIS
One-line summary of the macro invocation syntax.

MACRO ARGUMENTS
One-line summary of the nature of each *macro* argument.

DSP MEMORY ARGUMENTS
One-line summary of the nature of each *memory* argument.

DESCRIPTION
A complete description of the macro's purpose and function.

PSEUDO-C NOTATION
Describes the macro's function in a DSP-extended C language.

DSPWRAP C SYNTAX
Calling-sequence summary for array processing C function.

Source-Code Comment Fields (continued)

C Code Generation:

DSPWRAP ARGUMENT INFO
Declares the type of each macro argument.

General Information:

MINIMUM EXECUTION TIME
MAXIMUM EXECUTION TIME
Execution time per element (AP) or for one tick (UG).

CALLING DSP PROGRAM TEMPLATE
Test main program in DSP56001 assembly language.

SOURCE
Pointer to on-line macro source file.

MEMORY USE
Words of DSP program memory used by macro expansion.

REGISTER USE
Which ALU registers are used and what they contain.