Phasor Programming: Supplemental Material:

Let's examine the following commands:

```
C411:0  N C410:F N C410:C
C411:F4 N C410:E N C410:C
```

There are 2 logical registers which are used to set the 14 sound parameters for each chip.
In this example, $C411 is the Data Register, and $C410 is the Control Register.

We first poke the Parameter Number (0) in the Data Register, and then set the bits of the Control Register to tell the chip to LATCH the data register value as the Parameter Number we want to set.

Then we poke the Data Value ($F4 = 244 = the note C one octave above middle C) for the current parameter number (0 = Tone Period Fine for Sound Channel A). Please note that the value 244 is only a recommended value which generates a frequency that sounds close to C using a tuning of A=440Hz. Additional values for recommended frequencies can be found in the Mockingboard manual. However, I encourage you to experiment with alternate and Non-Western tunings, as well as microtonal music. Give your music or physics teacher an Apple and encourage students to do something interesting with it. We then repeat these two steps for each of the 14 sound parameters. The next command sets Parameter Number 1 (Tone Period Coarse for Sound Channel A) to the value 0.

```
C411:1  N C410:F N C410:C
C411:0  N C410:E N C410:C
```

In our testing, we did not set all 14 parameters. The initialization routine zeroed out all parameter values, so we only set the ones we needed, and the card produced sound when the volume was set to a nonzero value, or an envelope shape or period was changed. While machine language programmers may want to take advantage of this feature and quickly change parameter values to create effects, it is a best practice to create a subroutine that sets all 14 parameter values at once to avoid unintended side effects. But for now, let's look at the remaining commands we issued to play the first tone:

```
C411:7  N C410:F N C410:C
C411:38 N C410:E N C410:C
C411:8  N C410:F N C410:C
C411:A  N C410:E N C410:C
```

We poked parameter 7 with the hex value $38 (56 decimal), which is a bit pattern that tells the chip which voices should be enabled for tones and which should be enabled for noise. When we changed the tone to noise, we changed the enable value to 7, which is 00000111 in binary. The 3 bits which are set to 1 disable tones on all 3 sound channels, so only noise is played. The value $38 equals 00111000 in binary, so the 3 bits which are set to 1 disable noise on all 3 sound channels. Noise and tones can be mixed to create interesting effects. The value 0 enables both noise and tone on all channels.

When we poked parameter 8 with the hex value $0A (10 decimal), it set the Volume of Sound Channel A for the chip. Fixed Volume ranges from $00 to $0F (0=silence, 1-15=soft to loud). A value of $10 (16 decimal) for Volume indicates that the volume should be varied by the chip based on the Envelope Shape and Envelope Period.

First notice that we stopped the tone from playing by entering:

```
C411:0 N C410:E N C410:C
```

We took a shortcut here and did not set the Parameter number. We left it at 8 and the chip remembered it. When we set the Control register values to $E and $C, the chip recognized that as the bit pattern for a WRITE command, so it wrote the value 0 in the current Parameter number (8), which resulted in silencing the tone that was playing on Tone Channel A by setting its volume to 0. This shortcut should only be used by Machine Language programmers for special effects. The best practice is to always issue 2 commands: a LATCH command to set the Parameter Number followed by a WRITE command to set the Value for the parameter.

Before we continue analyzing the commands we issued, let's discuss the available sound parameters, and how to access the other sound chips.

The commands that we entered into the monitor can be entered into Basic as a subroutine:

```
POKE DR(CH-1), RG
POKE CR(CH-1), LA(0,CH-1)
POKE CR(CH-1), LA(1,CH-1)
POKE DR(CH-1), DA
POKE CR(CH-1), WR(0, CH-1)
POKE CR(CH-1), WR(1, CH-1)
RETURN
```

The variables are defined as follows:

CH = Chip number. You pass in 1-4 and the subroutine converts it to the range 0-3. Error handling should be added to prevent invalid values from poking memory locations out of range.

DR array = the address of the Data Register for each Chip. Array Index = 0 to 3

CR array = the address of the Control Register for each Chip. Array Index = 0 to 3

RG = Register number (same as Parameter Number. Range = 0-13)

DA = Data value for the current Parameter Number. Range= 0-255.

LA array = The two byte values to be sent to each chip for the LATCH command. Range: [0-1, 0-3]

WR array = The two byte values to be sent to each chip for the WRITE command. Range: [0-1, 0-3]

The arrays should be initialized as follows:

```
SL = 4: REM SLOT
DIM CR(4),DR(4),LA(2,4),WR(2,4)
CR(0) = 12 * 4096 + SL * 256 + 16:CR(1) = CR(0)
DR(0) = CR(0) + 1:DR(1) = CR(1) + 1
CR(2) = 12 * 4096 + SL * 256 + 8 * 16:CR(3) = CR(2)
DR(2) = CR(2) + 1:DR(3) = CR(3) + 1
LA(0,0) = 15:LA(1,0) = 12:WR(0,0) = 14:WR(1,0) = 12
LA(0,1) = 23:LA(1,1) = 20:WR(0,1) = 22:WR(1,1) = 20
LA(0,2) = 15:LA(1,2) = 12:WR(0,2) = 14:WR(1,2) = 12
LA(0,3) = 23:LA(1,3) = 20:WR(0,3) = 22:WR(1,3) = 20
```

In hex:

CR(0) and CR(1) = $C410	DR(0) and DR(1) = $C411
CR(2) and CR(3) = $C480	DR(2) and DR(3) = $C481

The following chart summarizes the addresses and commands for each chip: (s=slot number)

CHIP Number	Control Register	Data Register	LATCH	WRITE
1	$Cs10	$Cs11	$0F, $0C	$0E, $0C
2	$Cs10	$Cs11	$17, $14	$16, $14
3	$Cs80	$Cs81	$0F, $0C	$0E, $0C
4	$Cs80	$Cs81	$17, $14	$16, $14

As an exercise, practice issuing the same experimental commands from the beginning of this article for chips 2 thru 4. Notice which sounds play through the left speaker and which play through the right speaker.

<u>Sound Parameters</u>

The parameters that can be set for each chip are as follows:

Parameter #	**Description**	**Range**
0	Channel A Fine Tone Period	0-255
1	Channel A Coarse Tone Period	0-15
2	Channel B Fine Tone Period	0-255
3	Channel B Coarse Tone Period	0-15
4	Channel C Fine Tone Period	0-255
5	Channel C Coarse Tone Period	0-15
6	Noise Period	0-31
7	Enable Bits (Noise/Tone for each Channel)	0-63
8	Channel A Amplitude	0-15, 16=variable
9	Channel B Amplitude	0-15, 16=variable
10 ($0A)	Channel C Amplitude	0-15, 16=variable
11 ($0B)	Envelope Period Fine	0-255
12 ($0C)	Envelope Period Coarse	0-255
13 ($0D)	Envelope Shape: 8 = \\|\\|\\|\\|\\|\\|\\|\\| (sawtooth, ramping down) 9 = \______________ (one-shot decay to silence) 10 = \/\/\/\/\/\/\/\/ (triangle, starting with down ramp) ______________ 11 = \\| (one-shot decay, then full volume) 12 = /|/|/|/|/|/|/|/| (sawtooth, ramping up) _______________ 13 = / (ramp up to full volume) 14 = /\/\/\/\/\/\/\/\ (triangle, starting with up ramp) 15 = /|_______________ (one-shot ramping up)	8-15

Using this chart, you can continue experimenting by playing multiple tones on all 4 chips. The following commands will play a C major chord on Chip 1 using envelope shape 13 to ramp up and 9 to ramp down.

```
C0CD
C493:FF
C492:FF
C411:0  N C410:F N C410:C
C411:F4 N C410:E N C410:C
C411:1  N C410:F N C410:C
C411:0  N C410:E N C410:C
C411:2  N C410:F N C410:C
C411:C1 N C410:E N C410:C
C411:3  N C410:F N C410:C
C411:0  N C410:E N C410:C
C411:4  N C410:F N C410:C
C411:A3 N C410:E N C410:C
C411:5  N C410:F N C410:C
C411:0  N C410:E N C410:C
C411:7  N C410:F N C410:C
C411:38 N C410:E N C410:C
C411:8  N C410:F N C410:C
C411:10 N C410:E N C410:C
C411:9  N C410:F N C410:C
C411:10 N C410:E N C410:C
C411:A  N C410:F N C410:C
C411:10 N C410:E N C410:C
C411:B  N C410:F N C410:C
C411:0  N C410:E N C410:C
C411:C  N C410:F N C410:C
C411:20 N C410:E N C410:C
C411:D  N C410:F N C410:C
C411:D  N C410:E N C410:C

To stop the chord:
C411:D  N C410:F N C410:C
C411:9  N C410:E N C410:C
```