

Yamaha YM2610 Application Manual II

or something like that. text by freem.
(v0.25, 2015/12/01)

Warning: This document takes advantage of modern crap like inline MathML rendering and SVG rendering using the tag. If you're using a really old browser, or a browser which refuses to implement MathML/do so properly, this will probably not work out for you. There's also some Javascript used for lazy people who don't want to break out a calculator for finding Delta-N values.

You're most likely reading this because you want to program a sound driver for the Neo-Geo (or another YM2610-using system/arcade board/whatever). If that's not why you're reading this, or your reason is not tangentially related, then you probably want to find a much better source of information. This information is primarily written from the perspective of the Neo-Geo, though. Reader beware. (For example, the system clock of the YM2610 in the Neo-Geo has a value of 8000000, and all equations in this document use that number directly instead of saying "system clock".)

The bulk of this page is based on a translation of the [Yamaha YM2610 Application Manual](#) (hence the page name). As the YM2610 (OPNB) is in the same family as the YM2608 (OPNA), the Nemesis-transcribed YM2608 Application Manual is also a great source of information. (You will need to watch out for differences between the 2608 and 2610, however.) I don't want to direct link it myself, but do a search for "YM2608 Application Manual" and you should find the PDF on Hacking Cult.

Table of Contents

1. YM2610 Information

1. [Functional Overview](#)
 1. [Register Writing](#)
 2. [ADPCM Register Initial Values](#)
 3. [Register Address Allocation](#)
2. [Status Registers](#)
 1. [Status 0](#)
 2. [Status 1](#)
3. [Timers](#)
 1. [Calculating Timer A](#)
 2. [Calculating Timer B](#)
 3. [Timer Control](#)
 4. [2CH Mode](#)
4. [SSG](#)
 1. [Register Overview](#)
 2. [Tone Period](#)
 3. [Noise Period](#)
 4. [Amplitude/Mode](#)
 5. [SSG Envelope Period](#)
 6. [SSG Envelope Shape/Cycle](#)
5. [ADPCM-B](#)
 1. [Register Overview](#)
 2. [ADPCM-B Sample Addresses](#)
 3. [Delta-N/Sampling Rate Calculation](#)
 4. [ADPCM Flag Control](#)
6. [ADPCM-A](#)
 1. [Register Overview](#)
 2. [Dump/ADPCM-A On](#)
 3. [ADPCM-A Total Level](#)
 4. [Output Select/Channel Level](#)
 5. [ADPCM-A Sample Addresses](#)
7. [FM](#)
 1. [Common Registers](#)
 2. [Channel Registers](#)
 3. [Algorithm](#)
 4. [Self Feedback](#)
 5. [Key On/Off](#)
 6. [Key Scale](#)
 7. [Attack Rate](#)
 8. [Decay Rate](#)
 9. [Sustain Rate](#)
 10. [Sustain Level](#)
 11. [Release Rate](#)
 12. [F-Numbers and Block](#)
 13. [Detune](#)
 14. [Multiple](#)
 15. [LFO Frequency](#)
 16. [PMS and AMS](#)

2. [ADPCM Sample Data](#)

1. [ADPCM-A](#)
2. [ADPCM-B](#)

3. [Neo-Geo Sound Driver](#)

1. [Introduction](#)
2. [Overall Z80 Memory Layout](#)
3. [Entry Point](#)
4. [Interrupts](#)
 1. [IRQ](#)
 2. [NMI](#)
 3. [Others](#)
5. [Main Loop](#)
6. [Command Handling](#)
 1. [Command \\$01 \(Slot Switch\)](#)
 2. [Command \\$02 \(Logo Music\)](#)
 3. [Command \\$03 \(Soft Reset\)](#)
 4. [Command Handler \(Other Commands\)](#)
7. [ADPCM Samples](#)
 1. [ADPCM-A](#)
 2. [ADPCM-B](#)
8. [Using the SSG Channels](#)
9. [Using the FM Channels](#)
10. [Z80 Banking and Modification](#)
 1. [Bankswitching](#)
 2. [Neo-Geo CD Techniques](#)
11. [CDDA Playback \(Neo-Geo CD only\)](#)

Functional Overview

Register Writing

When writing to the registers, first send the address, then send the data. (The order must always be address followed by data.) However, if you're accessing the same address multiple times, you may write the address first and proceed to write the data register multiple times.

After writing to the address or data registers, you must wait before accessing them again. The wait time differs between address and data writes.

| Wait Time | |
|---------------------|-----------|
| after Address write | 17 cycles |
| after Data write | 83 cycles |

Some register pairs must be written in a certain order for proper operation. These will be noted in the relevant sections.

ADPCM Register Initial Values

Cleared at initialization time (when IC = "0"), register initial values are as follows:

| ADPCM-A | | | ADPCM-B | | |
|---------|-----------------------------|-------------|---------|---------------------|-------------|
| Address | Register | Init. Value | Address | Register | Init. Value |
| 00 | Dump/ADPCM-A On | 0 | 10 | Control 1 | 0 |
| 01 | Total Level | 0 | 11 | Control 2 | 0 |
| 02 | Test | 0 | 12 | Start Address (LSB) | XX |
| 08-0D | Output Select Channel Level | 0 | 13 | Start Address (MSB) | XX |
| 10-15 | Start Address (LSB) | 0 | 14 | End Address (LSB) | XX |
| 18-1D | Start Address (MSB) | 0 | 15 | End Address (MSB) | XX |
| 20-25 | End Address (LSB) | 0 | 19 | Delta-N (LSB) | XX |
| 28-2D | End Address (MSB) | 0 | 1A | Delta-N (MSB) | XX |
| | | | 1B | EG Control | XX |
| | | | 1C | Flag Control | 0 |

A value of XX means "undefined".

Register Address Allocation

This controls whether or not you write to ports 4/5 or ports 6/7.

| A1=0 (4/5) | | A1=1 (6/7) | |
|------------|---------------|------------|---------------|
| \$00 | SSG | \$00 | ADPCM-A |
| ... | | ... | |
| \$0F | | ... | |
| \$10 | ADPCM-B | ... | |
| ... | | ... | |
| \$1F | | ... | |
| \$20 | FM (Common) | ... | |
| ... | | ... | |
| \$2F | | \$2F | |
| \$30 | FM (Ch1, Ch2) | \$30 | FM (Ch3, Ch4) |
| ... | | ... | |
| \$B6 | | \$B6 | |

Note: On other systems (e.g. not the Neo-Geo), you might need to write the values elsewhere. For example, on the Sega Genesis/Mega Drive (which uses the similar YM2612 chip), Z80 programs write addresses and data to the pairs 0x4000/0x4001 (A1=0) and 0x4002/0x4003 (A1=1).

Status Registers

Status registers allow you to access important information about the current status of the YM2610.

Status 0

To get the values in Status 0, read port 4.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|------|----|----|----|----|----|---------|---------|
| Busy | | | | | | Timer B | Timer A |

Busy

When this bit is set to 1, the YM2610 is busy with the registers. As mentioned in "[Register Writing](#)", there has to be a wait after address and data writes. There are two ways of doing this: manually (running some code to burn clock cycles), and by watching this bit. When set to 0, it's safe to write to the address/data registers again.

Timer A, Timer B

If one of the timers has expired, the relevant bit will be set to 1. (todo: explain resetting timers)

Status 1

To get the values in Status 1, read port 6.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|----|----|----|----|----|----|----|----|
| B | | A5 | A4 | A3 | A2 | A1 | A0 |

Flags

The flags here are used to determine if a channel has finished playing its sample. This is an especially useful register for managing both ADPCM-A looping and custom loop points.

The values are set as follows:

- Flags A0-A5 are set to 1 when each ADPCM-A channel (1-6) reaches its end address.
- Flag B is set when ADPCM-B has reached its end address.

Timers

Though the timers live in the Common FM section ([see below](#)), they're worth talking about separately.

| Addr. | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Comment |
|-------|---------|---------|---------|----------|----------|--------|--------|--|--------------------------------------|
| 24 | Timer A | | | | | | | | Timer A upper 8 bits |
| 25 | | | | | | | TA | | Timer A lower 2 bits |
| 26 | Timer B | | | | | | | | Timer B value |
| 27 | Mode | Reset B | Reset A | Enable B | Enable A | Load B | Load A | Timer A/B Control and 2CH Mode | |

Calculating Timer A

Timer A is a 10-bit timer with a minimum resolution of 9 microseconds (when Timer A = 1023). It can either be used as a normal timer or a CSM timer. The CSM timer handles the Key On/Off actions.

In theory, the ports should be written in \$24,\$25 order. In practice, (meaning in existing Neo-Geo M1 drivers) the timer is written in \$25,\$24 order.

$$tA = 72 \times (1024 - NA) \div 8000000$$

Where *NA* is the Timer A value (0-1023). *tA* is in microseconds.

Calculating Timer B

Timer B is an 8-bit timer with a minimum resolution of 144 microseconds (when Timer B = 255).

$$tB = 1152 \times (256 - NB) \div 8000000$$

Where *NB* is the Timer B value (0-255). *tB* is in microseconds.

Timer Control

(todo, as this is important)

Timer Load

Setting a **0** in the respective bit will stop that timer. If a **1** is written instead, the timer will begin.

Timer Enable

Setting a **0** in the respective bit will disable the timer from setting flags or generating IRQs. If a **1** is written instead, the timer will set the proper status flag once the counter overflows, and an IRQ will be generated.

Timer Reset

Setting a **1** in the respective bit will reset the status flags for that timer.

2CH Mode

Channel 2's output mode can be changed with these bytes. This has not been fully explored yet, and is partially based off of YM2608 documentation.

| D7 | D6 | Mode | Function |
|----|----|--------|--|
| 0 | 0 | Normal | Normal output, like other channels. |
| 0 | 1 | CSM | CSM voice synthesis mode; each slot can use a separate F-Number. In CSM mode, key on/off is done with Timer A. |

| D7 | D6 | Mode | Function |
|----|----|------|---|
| 1 | 0 | ? | ? (YM2608: "Effect sound"; Each slot can have a separate F-Number and CSM.) |
| 1 | 1 | ? | ? (YM2608: "Effect sound"; Each slot can have a separate F-Number and CSM.) |

SSG

The Simple Sound Generator, originally found in the YM2149, is compatible with the AY-3-8910/PSG. On the YM2610, the I/O ports were removed. (The missing bits in register \$07 and missing addresses \$0E and \$0F correspond to the I/O functions.)

Register Overview

All SSG addresses and data are written to ports 4/5.

| Addr. | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Comment |
|-------|-------------|----|-----|----------------|-------------|-----|-----|-----|--|
| 00 | Fine Tune | | | | | | | | Channel A Tone Period |
| 01 | | | | | Coarse Tune | | | | |
| 02 | Fine Tune | | | | | | | | Channel B Tone Period |
| 03 | | | | | Coarse Tune | | | | |
| 04 | Fine Tune | | | | | | | | Channel C Tone Period |
| 05 | | | | | Coarse Tune | | | | |
| 06 | | | | Period Control | | | | | Noise Period |
| 07 | | | /nC | /nB | /nA | /tC | /tB | /tA | /Enable (Output when 0; n=noise, t=tone) |
| 08 | | | | M | Level | | | | Channel A Amplitude/Mode |
| 09 | | | | M | Level | | | | Channel B Amplitude/Mode |
| 0A | | | | M | Level | | | | Channel C Amplitude/Mode |
| 0B | Fine Tune | | | | | | | | Envelope Period |
| 0C | Coarse Tune | | | | | | | | |
| 0D | | | | | CON | ATT | ALT | HLD | Envelope Shape Cycle |

Tone Period

The SSG tone period has a resolution of 12 bits, allowing for values from 1-4095. The "Coarse Tune" registers should be written before the "Fine Tune" registers.

To calculate the frequency from a given tone period, use this equation:

$$f_{\text{tone}} = 8000000 \div (64 \times TP)$$

Where TP is the Tone Period in decimal (base 10).

Noise Period

The SSG noise period has a resolution of 5 bits, allowing for values from 1-31. It is shared between all three channels. To calculate the frequency from a given noise period, use this equation:

$$f_{\text{tone}} = 8000000 \div (64 \times NP)$$

Where NP is the Noise Period in decimal (base 10).

Amplitude/Mode

The "M" bit in the Amplitude/Mode controls determines how the volume is modified.

- When **0**: Use the values in D3-D0.
- When **1**: Use the envelope generator. (See below.)

Amplitude Table

Values for the "Level" portion of the register:

| D3 | D2 | D1 | D0 |
|----|----|----|----|
| 16 | 8 | 4 | 2 |

It should be noted that the default level is 1, meaning if you set 0x00 here, the level will still be 1.

SSG Envelope Period

The SSG envelope period controls the repetition period of the envelope generator. It only matters if the "M" bit in Amplitude/Mode is set to 1 for at least one channel. The envelope is shared between all SSG channels.

Write to address \$0C first, then \$0B.

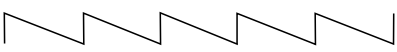
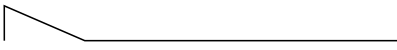
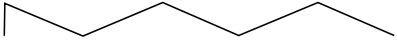

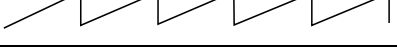

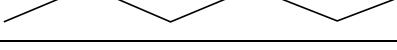
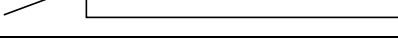
The SSG envelope period has a resolution of 16 bits, allowing for values from 1-65535. To calculate the frequency from a given envelope period, use this equation:

$$\text{EG freq.} = 8000000 \div (1024 \times \text{EP})$$

Where *EP* is the Envelope Period in decimal (base 10). Cycle tEG (?) is generated by 1/EG freq.

SSG Envelope Shape Cycle

(todo: point out the loop region in each shape)

| No. | CON | ATT | ALT | HLD | Envelope Shape |
|-----|-----|-----|-----|-----|---|
| 0 | 1 | 0 | 0 | 0 |  |
| 1 | 1 | 0 | 0 | 1 |  |
| 2 | 1 | 0 | 1 | 0 |  |
| 3 | 1 | 0 | 1 | 1 |  |
| 4 | 1 | 1 | 0 | 0 |  |
| 5 | 1 | 1 | 0 | 1 |  |
| 6 | 1 | 1 | 1 | 0 |  |
| 7 | 1 | 1 | 1 | 1 |  |

ADPCM-B

There is only one ADPCM-B channel, but it allows for varying sample rates (1.8KHz to 55.5KHz), unlike the fixed sample rate of ADPCM-A.

It should be noted that the Neo-Geo CD does not have ADPCM-B, as you're expected to use the CD-DA instead. Go figure.

Register Overview

All ADPCM-B addresses and data are written to ports 4/5.

| Addr. | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Comment |
|-------|-------|----|----|--------|----|----|----|-------|-----------|
| 10 | Start | | | Repeat | | | | Reset | Control 1 |

| Addr. | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Comment |
|-------|----------------------|----|----|----|----|----|----|----|---|
| 11 | L | R | | | | | | | Left/Right Control |
| 12 | Address lower 8 bits | | | | | | | | Start Address/256 (LSB) |
| 13 | Address upper 8 bits | | | | | | | | Start Address/256 (MSB) |
| 14 | Address lower 8 bits | | | | | | | | End Address/256 (LSB) |
| 15 | Address upper 8 bits | | | | | | | | End Address/256 (MSB) |
| 19 | Delta-N lower 8 bits | | | | | | | | Delta-N (LSB) |
| 1A | Delta-N upper 8 bits | | | | | | | | Delta-N (MSB) |
| 1B | Output Level | | | | | | | | EG Control (Volume) |
| 1C | B | | A5 | A4 | A3 | A2 | A1 | A0 | ADPCM Flag Control |

ADPCM-B Sample Addresses

The Start and End address registers only use two bytes, so the third byte of the sample address is forced, dependent on Start (all 0s) or End (all 1s).

For example, if a sound starts at physical address 0x000400 and ends at 0x000800 (non-inclusive), the values to be written would be 0x0004 and 0x0007 (translating to an end address of 0x0007FF).

Start

| D23 | D22 | D21 | D20 | D19 | D18 | D17 | D16 | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|-----------|-----|-----|-----|-----|-----|-----|-----|-----------|-----|-----|-----|-----|-----|----|----|---------|----|----|----|----|----|----|----|
| Start MSB | | | | | | | | Start LSB | | | | | | | | Fixed 0 | | | | | | | |

End

| D23 | D22 | D21 | D20 | D19 | D18 | D17 | D16 | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---------|-----|-----|-----|-----|-----|-----|-----|---------|-----|-----|-----|-----|-----|----|----|---------|----|----|----|----|----|----|----|
| End MSB | | | | | | | | End LSB | | | | | | | | Fixed 1 | | | | | | | |

Delta-N/Sampling Rate Calculation

The translated YM2610 datasheet going around gives the following formula for calculating the frequency from a Delta-N value, where ΔN is a 16-bit unsigned value:

$$f = \Delta N \div 256 \times 55.5$$

[KHz]

However, the YM2610 datasheet's algorithm doesn't seem to give proper values, so it's easier to follow the equation from the YM2608 datasheet:

$$\Delta N = (f \div 55.5 \text{ KHz}) \times 65536$$

If you would like to use the sampling rate in Hertz, expand 55.5 to 55000 and make sure you fill in f accordingly.

Sampling Rate to Delta-N Converter

Sampling Rate: ☒ KHz ☐ Hz

ADPCM Flag Control

Setting any bit to 1 in this register will set the corresponding flag in [Status 1](#) to 0.

ADPCM-A

Register Overview

All SSG addresses and data are written to ports 6/7.

| Addr. | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Comment |
|-------|----------------------|----|-------------|---------------|-----|-----|-----|-----|---|
| 00 | DM | | Ch5 | Ch4 | Ch3 | Ch2 | Ch1 | Ch0 | Dump/ADPCM-A On |
| 01 | | | Total Level | | | | | | ADPCM-A Total Level |
| 02 | Test | | | | | | | | LSI Test Data |
| 08-0D | L | R | | Channel Level | | | | | Output Select/Channel Level |
| 10-15 | Address lower 8 bits | | | | | | | | Start Address/256 (LSB) |
| 18-1D | Address upper 8 bits | | | | | | | | Start Address/256 (MSB) |
| 20-25 | Address lower 8 bits | | | | | | | | End Address/256 (LSB) |
| 28-2D | Address upper 8 bits | | | | | | | | End Address/256 (MSB) |

Dump/ADPCM-A On (\$00)

The "Dump" bit is more like a playback toggle bit. If it's 0, the sound will play. If it's 1, the sound will stop.

ADPCM-A Total Level (\$01)

ADPCM-A total volume is between 0dB and -47.25dB in steps of 0.75dB. When all values are set to 1, the volume is 0dB.

Output Select/Channel Level (\$08-\$0D)

ADPCM-A channel volumes are between 0dB and -23.25dB in steps of 0.75dB. When all values are set to 1, the volume is 0dB.

ADPCM-A Sample Addresses

ADPCM-A Sample Addresses operate under a few specific rules...

- A sample can't span multiple ROMs.
- An ADPCM-A sample can't be larger than 1MiB.
- ADPCM-A and ADPCM-B samples can't be mixed together; PCM-B samples should begin after the last PCM-A sample.
- Each sample is expected to start at 0x____00 and end at 0x____FF.

Start

| D23 | D22 | D21 | D20 | D19 | D18 | D17 | D16 | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|------------|-----|-----|-----|-----------|-----|-----|-----|-----------|-----|-----|-----|-----|-----|----|----|---------|----|----|----|----|----|----|----|
| Start MSB* | | | | Start MSB | | | | Start LSB | | | | | | | | Fixed 0 | | | | | | | |

End

| D23 | D22 | D21 | D20 | D19 | D18 | D17 | D16 | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|----------|-----|-----|-----|---------|-----|-----|-----|---------|-----|-----|-----|-----|-----|----|----|---------|----|----|----|----|----|----|----|
| End MSB* | | | | End MSB | | | | End LSB | | | | | | | | Fixed 1 | | | | | | | |

* The topmost four bits of the Start and End MSB values must be the same.

FM

Common Registers

The common FM registers are written to ports 4/5.

| Addr. | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Comment |
|-------|---------|----|----|----|----|-----------|----|----|---------------------------------------|
| 21 | Test | | | | | | | | LSI Test Data |
| 22 | | | | | On | LFO Freq. | | | LFO Frequency Control |
| 24 | Timer A | | | | | | | | Timer A upper 8 bits |

| Addr. | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Comment |
|-------|---------|----|-------|----|--------|----|------|----|--------------------------------|
| 25 | | | | | | | TA | | Timer A lower 2 bits |
| 26 | Timer B | | | | | | | | Timer B data |
| 27 | Mode | | Reset | | Enable | | Load | | Timer A/B Control and 2CH Mode |
| 28 | Slot | | | | | | Chan | | Key On/Off |

Channel Registers

Addresses and data for FM channels 1 and 2 are written to ports 4/5. Channels 3 and 4 use ports 6/7.

| Addr. | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Comment |
|-------|---------------|----|-------------|-------|--------|---------------|----|----|---|
| 31-3E | | DT | | | MULTI | | | | Detune (DT)/ Multiple |
| 41-4E | | TL | | | | | | | Total Level (TL) |
| 51-5E | KS | | | AR | | | | | Key Scale (KS)/ Attack Rate (AR) |
| 61-6E | AM | | | | DR | | | | AM On/ Decay Rate (DR) |
| 71-7E | | | | SR | | | | | Sustain Rate (SR) |
| 81-8E | SL | | | | RR | | | | Sustain Level (SL)/ Release Rate (RR) |
| 91-9E | | | | | SSG-EG | | | | SSG-type Envelope Generator |
| A1,A2 | F-Num 1 | | | | | | | | F-Numbers/Block |
| A5,A6 | | | | Block | | F-Num 2 | | | |
| A9,AA | 2CH * F-Num 1 | | | | | | | | 2CH - 2 Slot F-Numbers/Block |
| AD,AE | | | 2CH * Block | | | 2CH * F-Num 2 | | | |
| B1,B2 | | | FB | | | Algorithm | | | Self Feedback (FB)/ Algorithm |
| B5,B6 | L | R | AMS | | | PMS | | | LR Sel./ AM Sense (AMS)/ PM Sense (PMS) |

(todo: does the YM2610 use a similar setup to the YM2608 for CSM mode?)

Channel Operators

As the YM2610 has four operators per channel, the above register ranges need to be clarified.

| Operator | 1 | 2 | 3 | 4 |
|------------|----|----|----|----|
| Ch.1, Ch.3 | *1 | *5 | *9 | *D |
| Ch.2, Ch.4 | *2 | *6 | *A | *E |

Algorithm

(todo: diagrams)

The Algorithm, also known as the Connection, defines how each of the operators interacts with each other. There are eight algorithms. The algorithm can be thought of as the base element of the sound, while the parameters define the rest of the sound's character.

Self Feedback

The first slot of each channel provides a self-feedback function. If " π " looks odd on your system, it's meant to be the value of pi.

| Feedback | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|------------------|-----|----------|---------|---------|---------|-------|--------|--------|
| Modulation Level | Off | $\pi/16$ | $\pi/8$ | $\pi/4$ | $\pi/2$ | π | 2π | 4π |

Key On/Off

This is the primary way of starting/stopping notes. The on/off status of the four operators can be set.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|-----|-----|-----|-----|----|----------|----|----|
| Op4 | Op3 | Op2 | Op1 | | Channels | | |

Channels

The regular YM2610 only has four channels, but the YM2610B can access all six channels. (Setting all channel bits to 1 is unknown behavior??)

YM2610/Neo-Geo channels are in parentheses. Hard channels 3/4 are unused on the Neo-Geo, with the real channels 3/4 being mapped to hard channels 5/6. I wonder if anyone has the guts to put a YM2610B in their machine and perform some tests...

| Channel | D2 | D1 | D0 |
|---------|----|----|----|
| 1 (1) | 0 | 0 | 0 |
| 2 (2) | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 |
| 4 | 1 | 0 | 0 |
| 5 (3) | 1 | 0 | 1 |
| 6 (4) | 1 | 1 | 0 |

Key Scale

(todo)

Attack Rate

The Attack Rate defines how fast the sound reaches its maximum volume after Key On.

Valid values are 0-31. Larger rates mean the sound reaches maximum volume faster. A value of 0 will cause the envelope to not start.

Decay Rate

The Decay Rate defines how quickly the sound will drop to the Sustain Level after the initial peak.

Valid values are 0-31. Larger rates mean the sound decays faster. A value of 0 will cause the sound to continue without decaying.

Sustain Rate

The Sustain Rate defines the speed at which the volume of the sustained note increases or decreases.

Sustain Level

The Sustain Level defines the volume of the sound after the decay, but before the note is released (Key off).

Valid values are 0-15.

| | D7 | D6 | D5 | D4 |
|-------------|----|----|----|----|
| Volume (db) | 24 | 12 | 6 | 3 |

When all bits are set to 1, the value is 93dB.

Release Rate

The Release Rate defines how quickly the sound ends when the key is released (Key off).

Valid values are 0-15. Larger rates provide a shorter wait. A value of 0 will ??

F-Numbers and Block

The F-Numbers and Block value set the note frequency. The ideal way to set up your values involves creating a single octave's worth of F-Numbers, and then use that set of values while changing the Block.

To calculate the F-Number from a given frequency, use this equation:

F-Number = (144 ×
freq × 1048576 ÷
8000000) ÷ 2 Block –
1

| \$A5,\$A6 (2CH mode: \$AD,\$AE) | | | | | | | | | \$A1,\$A2 (2CH mode: \$A9,\$AA) | | | | | | | | |
|---------------------------------|----|----|-------|----|----|---------|----|--|---------------------------------|---------|----|----|----|----|----|----|--|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |
| | | | Block | | | F-Num 2 | | | | F-Num 1 | | | | | | | |

Block and F-Num 2 must be written first!

Detune

(todo)

D6 is the sign bit.

| Value | D6 | D5 | D4 |
|-------|----|----|----|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| -0 | 1 | 0 | 0 |
| -1 | 1 | 0 | 1 |
| -2 | 1 | 1 | 0 |
| -3 | 1 | 1 | 1 |

Mutliple

(todo)

0 is equal to 1/2, while 1-F (hex) map to 1-15.

LFO Frequency

Values for the LFO Frequency Control register (\$22) are as follows:

| Freq. Cont. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-------------|------|------|------|------|------|------|------|------|
| freq. (Hz) | 3.98 | 5.56 | 6.02 | 6.37 | 6.88 | 9.63 | 48.1 | 72.2 |

PMS and AMS

PM Sense

(todo)

| PMS Value | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---------------|---|------|------|-----|-----|-----|-----|-----|
| Diff. (Cents) | 0 | ±3.4 | ±6.7 | ±10 | ±14 | ±20 | ±40 | ±80 |

AM Sense

(todo)

| AMS Value | 0 | 1 | 2 | 3 |
|------------|---|------|-----|------|
| Diff. (dB) | 0 | 1.44 | 5.9 | 11.8 |

ADPCM Sample Data

The YM2610 is capable of playing back two different ADPCM types, each with a different task (and encoding).

ADPCM-A

ADPCM-A has a fixed sample rate of 18.5KHz. It's the primary method for sample playback, as there are 6 channels. It is available on all versions of the Neo-Geo.

Sounds were compressed using SNK's tools; To my knowledge, these tools aren't available to the general public. The first capable ADPCM-A sample compressor available to the public was found in MVSTracker, though it was a bit buggy. This code was later used by HPMAN as part of the base for a new compressor (which is not available as a binary).

There is [a command line tool](#) capable of producing working ADPCM-A samples.

ADPCM-B

ADPCM-B has a variable sample rate, which can be anywhere from 5.55KHz to 55.5KHz. This allows for a single sample to imitate an instrument, or can be exploited for space gains. There is only one ADPCM-B channel available. However, ADPCM-B is not available on the Neo-Geo CD.

There was likely a tool from Yamaha or SNK to encode ADPCM-B, though no records have been found to suggest so. A [third-party ADPCM-B encoder](#) by ValleyBell and Fred/Front was made available in October 2011.

Neo-Geo Sound Driver

At this point, we can attempt to write a sound driver for the Neo-Geo. This involves a Z80 driving the YM2610, and the 68K and Z80 exchanging command bytes in order for sounds and music to be played.

A successful driver will make use of at least one of the Timers*, and will allow playback on the FM channels, the SSG channels, and all ADPCM channels. I lack the knowledge to get CDDA working right now, so this primarily focuses on the cart systems.

* It is possible to keep the sound engine running via other means, but timers are the most efficient way.

Introduction

On the Neo-Geo, the sound CPU and the main CPU are not directly connected; communication happens via writing to/reading from hardware ports. The 68K (main) CPU sends commands to the Z80 (sound) CPU via a 68K memory mapped register at `$320000`. On the Z80 side, port `$00` is used for reading the code sent from the 68K, and port `$0C` is used to reply to the 68K.

Overall Z80 Memory Layout

On the Neo-Geo, the Z80 has 65535 bytes available to it. About 2048 bytes are RAM, while the rest is ROM (banked or not).

| Start Address | End Address | Size | Description |
|---------------|-------------|-------|-------------------|
| 0x0000 | 0x7FFF | 32KiB | Main code bank |
| 0x8000 | 0xBFFF | 16KiB | Switchable bank 3 |
| 0xC000 | 0xDFFF | 8KiB | Switchable bank 2 |
| 0xE000 | 0xFFFF | 4KiB | Switchable bank 1 |
| 0xF000 | 0xF7FF | 2KiB | Switchable bank 0 |
| 0xF800 | 0xFFFF | 2KiB | Work RAM |

Entry Point

Z80 code starts at position `0x0000`. The first two instructions are typically `di` (disable interrupts) and `jp` (jump) to the real startup code. Jumping to startup code held elsewhere allows you to use some of the other top addresses for interrupts (see below).

In your startup code, you'll want to do the following:

- Set the stack pointer (Typically to `0xFFFFC`; see below for why).
- Set Interrupt Mode 1
- Clear the RAM (`0xF800-0xFFFF`)
- Initialize any variables you're going to use.
- Silence/stop the SSG, FM, and ADPCM channels.
- Set the default program banks (mainly for CD systems?).

- Set initial timer values.
- Set initial volume and output channels for ADPCM-A and ADPCM-B.
- Enable NMIs by writing 1 to port 0x8.

Interrupts

The Z80 features a number of interrupts, as well as a variety of ways to handle them. Most drivers typically use Interrupt Mode 1 (im 1), which puts the IRQ at 0x0038. This is the recommended course of action, unless you are a Z80 guru. Everything in this section is written assuming Interrupt Mode 1 is being used.

IRQ

The IRQs are generated by the YM2610 when one of the [timers](#) expires. The two least significant bits of Status 0 contain the timer status, allowing you to see which timer fired off the IRQ.

Example IRQ Handler

```

IRQ:
    ; save registers
    push af
    push bc
    push de
    push hl
    push ix
    push iy

    ; update internal status 1 register
    in a,(6)
    ld (intStatus1),a

    ; check if any ADPCM channel has ended playback
    ; %10000000: ADPCM-B
    ld a,(intStatus1)
    bit 7,a
    jp Z,IRQ_CheckPCMA6

    ; (handle ADPCM-B channel ending)

IRQ_CheckPCMA6:
    ; %00100000: ADPCM-A Ch. 6
    ld a,(intStatus1)
    bit 5,a
    jp Z,IRQ_CheckPCMA5

    ; (handle ADPCM-A channel 6 ending)

IRQ_CheckPCMA5:
    ; %00010000: ADPCM-A Ch. 5
    ld a,(intStatus1)
    bit 4,a
    jp Z,IRQ_CheckPCMA4

    ; (handle ADPCM-A channel 5 ending)

IRQ_CheckPCMA4:
    ; %00001000: ADPCM-A Ch. 4
    ld a,(intStatus1)
    bit 3,a
    jp Z,IRQ_CheckPCMA3

    ; (handle ADPCM-A channel 4 ending)

IRQ_CheckPCMA3:
    ; %00000100: ADPCM-A Ch. 3
    ld a,(intStatus1)
    bit 2,a
    jp Z,IRQ_CheckPCMA2

    ; (handle ADPCM-A channel 3 ending)

```

```

IRQ_CheckPCMA2:
    ; %00000010: ADPCM-A Ch. 2
    ld a,(intStatus1)
    bit 1,a
    jp Z,IRQ_CheckPCMA1

    ; (handle ADPCM-A channel 2 ending)

IRQ_CheckPCMA1:
    ; %00000001: ADPCM-A Ch. 1
    ld a,(intStatus1)
    bit 0,a
    jp NZ,IRQ_UpdateStatus0

    ; (handle ADPCM-A channel 1 ending)

IRQ_UpdateStatus0:
    ; update internal status 0 register
    in a,(4)
    ld (intStatus0),a

    ; Check Timer B
    ld a,(intStatus0)
    bit 1,a
    jp Z,IRQ_CheckTimerA

    ; Timer B expired

IRQ_CheckTimerA:
    ld a,(intStatus0)
    bit 0,a
    jp Z,IRQ_end

    ; Timer A expired

IRQ_end:
    ; restore registers
    pop iy
    pop ix
    pop hl
    pop de
    pop bc
    pop af
    ret

```

NMI

The NMI (at address 0x0066) handles communication between the 68000 and Z80. This is one of the more important parts of your sound driver.

Example NMI Handler

```

NMI:
    ; save registers
    push af
    push bc
    push de
    push hl
    push ix
    push iy

    ; acknowledge NMI and get command from 68K
    in a,(0)
    ; save current command into a variable for later use
    ld (curCommand),a

    ; check for required commands
    cp 1          ; command 1 (slot switch)
    jp Z,doCommand1

```

```

    cp 3          ; command 3 (soft reset)
    jp Z,doCommand3
    or a          ; do nothing if command 0
    jp Z,NMI_end

    ; handle command
    call HandleCommand

    ; reply to 68K and clear sound code
    xor a
    out (0xC),a   ; reply to 68k
    out (0),a     ; clear sound code

NMI_end:
    ; restore registers
    pop iy
    pop ix
    pop hl
    pop de
    pop bc
    pop af
    retn

```

Others

In addition to the NMI and IRQ, interrupt vectors can exist at these locations: 0x0000, 0x0008, 0x0010, 0x0018, 0x0020, 0x0028, 0x0030, 0x0038. These vectors are easily accessed in software via the `rst` instruction (e.g. `rst $18`). You should take advantage of them for any code that's called a lot (e.g. writes to ports 4/5 and 6/7). In Interrupt Mode 1, you probably won't be calling `rst $38`.

Main Loop

This part is something I still don't understand how to write. It involves using a buffer for commands.

Command Handling

The Neo-Geo System ROM (BIOS) expects the Z80 to reply to certain commands in specific ways. Your sound driver might also want to handle its own commands as well.

Command \$01 (Slot Switch)

"sent by the system ROM when the slot is switched. As the Z80 rom will be swapped, all sounds need to be stopped, NMI needs to be activated, \$01 needs to be sent back to the 68k and the Z80 code has to sit in a loop in RAM. After receiving that \$01 reply, the system ROM can then switch slot without crashing the Z80." - NeoGeo Dev Wiki

(todo)

Example Command \$01 Handler

```

doCommand1:
    di
    xor a
    out (0xC),a   ; respond to 68K
    out (0),a     ; clear sound code
    ld sp,0xFFFC  ; set stack pointer

    ; call real command $01 handler
    ld hl,command_01
    push hl
    retn

command_01:
    di
    xor a
    out (0xC),a   ; respond to 68K
    out (0),a     ; clear sound code

    ; reset default banks
    call SetDefaultBanks

```

```

; (FM) turn off Left/Right, AM Sense and PM Sense
ld de,0xB500 ; $B500: turn off for channels 1/3
write45
write67
ld de,0xB600 ; $B600: turn off for channels 2/4
write45
write67

; (ADPCM-A, ADPCM-B) Reset ADPCM channels
ld de,0x00BF ; $00BF: ADPCM-A Dump=1, all channels=1
write67
ld de,0x1001 ; $1001: ADPCM-B Reset=1
write45

; (ADPCM-A, ADPCM-B) Poke ADPCM channel flags (write 1, then 0)
ld de,0x1CBF ; $1CBF: Reset flags for ADPCM-A 1-6 and ADPCM-B
write45
ld de,0x1C00 ; $1C00: Enable flags for ADPCM-A 1-6 and ADPCM-B
write45

; silence FM channels
ld de,0x2801 ; FM channel 1 (1/4)
write45
ld de,0x2802 ; FM channel 2 (2/4)
write45
ld de,0x2805 ; FM channel 5 (3/4)
write45
ld de,0x2806 ; FM channel 6 (4/4)
write45

; silence SSG channels
ld de,0x800 ; SSG Channel A
write45
ld de,0x900 ; SSG Channel B
write45
ld de,0xA00 ; SSG Channel C
write45

; set up infinite loop in RAM
ld hl,0xFFFF
ld (hl),0xC3 ; Set 0xFFFF = 0xC3 ($C3 is opcode for "jp")
ld (0xFFFE),hl ; Set 0xFFFE = 0xFFFF (making "jp $FFFD")
ld a,1
out (0xC),a ; Write 1 to port 0xC (Reply to 68K)
jp 0xFFFD ; jump to infinite loop in RAM

```

Command \$02 (Logo Music)

(todo)

Command \$03 (Soft Reset)

Command \$03 tells the Z80 to reset itself; this needs to be done in under 100ms, according to the Neo-Geo Dev Wiki. No reply to the 68K is expected, but I give one here anyways. Feel free to nuke it in your code.

Example Command \$03 Handler

```

doCommand3:
di
xor a
out (0xC),a ; respond to 68K
out (0),a ; clear sound code
ld sp,0xFFFC ; set stack pointer

; call real command $03 handler
ld hl,command_03
push hl
retn

```



```
command_03:
    ; just in case it wasn't handled the first time around, or there was
    ; another command waiting before...
    di
    ld a,0
    out (0xC),a
    out (0),a

    ld sp,0xFFFF
    rst 0          ; go back to the beginning
```

Command Handler (Other Commands)

While the above three commands are expected by the BIOS, the remaining 252 (since 0 is "no command") can be handled in any way you wish. This can be a blessing and a curse.

Typical Command Mapping

A typical sound driver will split the available command range into multiple parts:

| Command Range | Usage |
|---------------------|-------------------|
| \$01-\$1F (32 cmds) | System Commands |
| \$20-\$5F (64 cmds) | FM music tracks |
| \$60-\$7F (32 cmds) | SSG sound effects |
| \$80-\$BF (64 cmds) | ADPCM-A samples |
| \$C0-\$FF (64 cmds) | ADPCM-B samples |

Example Command Handler

```
HandleCommand:
    ld a,(curCommand)
    ; todo: handle command
    ret
```

ADPCM Samples

Sampled audio playback is handled via two different types of ADPCM.

ADPCM-A

There are six [ADPCM-A](#) channels, each one with a fixed sampling rate of 18.5KHz.

Play Sample (Single Channel)

(todo)

Stop Sample (Single Channel)

```
pcma_StopCh1:
    ; stop ADPCM-A channel 1
    ld de,0x0081    ; $0081 = Dump ADPCM-A channel 1 (stop sound)
    write67         ; macro to write value in de to ports 6 and 7
    ret
```

Stop Sample (All Channels)

```
pcma_StopAll:
    ; stop all ADPCM-A channels
    ld de,0x009F    ; $009F = Dump all ADPCM-A channels (stop sound)
```

28/3/2019

Yamaha YM2610 Application Manual II

```
        write67      ; macro to write value in de to ports 6 and 7
        ret
```

ADPCM-B

There is only one [ADPCM-B](#) channel, and it only appears on Neo-Geo cart systems. Unlike the ADPCM-A channels, ADPCM-B's sampling rate is variable (5.55KHz to 55.5KHz).

Play Sample

(todo)

Stop Sample

```
pcmb_Stop:
    ; stop ADPCM-B sample
    ld de,0x1001    ; $1001 = force stop synthesis for ADPCM-B
    write45         ; macro to write value in de to ports 4 and 5
    dec e           ; de is now $1000; $1000 = stop ADPCM-B output
    write45
    ret
```

Using the SSG Channels

The [SSG](#) channels are a bit tougher to use than the ADPCM channels, but they're easier than the FM channels.

Play Note (Single Channel)

(todo)

Stop Note (Single Channel)

(todo)

Stop Note (All Channels)

(todo)

Using the FM Channels

The hardest part of the YM2610 is the [FM](#). This is typical of FM, with half of the battle being spent tweaking the parameters to make good instruments.

(todo)

Z80 Banking and Modification

Sometimes, 64KiB just isn't enough for all of your sound data. There are a few ways around this, depending on what system you're using. The cart systems allow you to swap out different banks; the CD systems give you a few more options, considering the Z80 section is all RAM.

Bankswitching

(todo: how the banks are laid out in a .M1 file based on which bank you're loading into)

Bank Map

The following table shows which Z80 ports control the banking, as well as the bank size for each port.

| Z80 port | Bank range | Bank size |
|----------|---------------|-----------|
| \$0B | \$8000-\$BFFF | 16KiB |
| \$0A | \$C000-\$DFFF | 8KiB |
| \$09 | \$E000-\$EFFF | 4KiB |
| \$08 | \$F000-\$F7FF | 2KiB |

Setting default banks

```
SetDefaultBanks:
    ; Set $F000-$F7FF bank to bank $1E (30 * 2K)
    ld a,0x1E
    in a,(8)

    ; Set $E000-$EFFF bank to bank $0E (14 * 4K)
    ld a,0x0E
    in a,(9)

    ; Set $C000-$DFFF bank to bank $06 ( 6 * 8K)
    ld a,6
    in a,(0xA)

    ; Set $8000-$BFFF bank to bank $02 ( 2 * 16K)
    ld a,2
    in a,(0xB)
    ret
```

Neo-Geo CD Techniques

(todo: Neo-Geo CD specific stuff (e.g. PAT files, editing the RAM directly))

CDDA Playback (Neo-Geo CD only)

(todo)

(other parts todo)

This would cover topics such as:

- Looping samples
- ?????

This guide is written by freem, and is not endorsed, authorized, or otherwise supported by Yamaha, SNK Playmore, Sega, or anyone else.