



特集1: FM音源の基礎とYM2203

特集2: GR-SAKURA用シールド



NT京都2014

特集 1 : FM 音源の基礎と YM2203	3
FM 音源と PSG 音源の基礎	3
音のパラメータ	3
エンベロープ	5
音色	6
オペレータとアルゴリズム	7
YM2203 のピン機能	8
YM2203 のレジスタ設定	10
PSG 音源のレジスタ設定	10
FM 音源のレジスタ設定	13
特集 2 : GR-SAKURA 用 FM 音源シールド	22
回路設計	23
ライブラリ設計	25
ライブラリ・リファレンス	25

特集 1 : FM 音源の基礎と YM2203

FM 音源と PSG 音源の基礎

FM 音源と PSG 音源は、どちらも 80 年代ごろにパソコンやゲーム機などに使用された音源です。ファミコンや MSX や PC-88 の音に郷愁を感じるオッサンも多いでしょう(笑) FM 音源は 90 年代～00 年代ごろには携帯電話の着メロにも使われました。FM 音源も PSG 音源も、音量・音程・音色などのパラメータを設定して音を鳴らすタイプの音源です。実際の音を録音して再生するのに比べてデータ量は非常に小さいので、記憶容量の貧弱だった頃のパソコンやゲーム機や携帯電話で用いられました。音色の豊かさにおいて、FM 音源のほうが PSG 音源より優れています。もちろんそのぶん、設定するパラメータが多くて複雑なわけです。なお、ヤマハでは PSG 音源のことを SSG 音源と呼称するのですが、本書では呼称を PSG 音源に統一します。

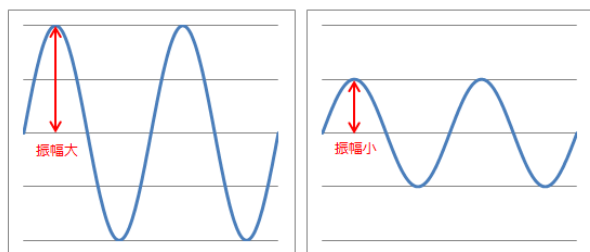
僕が今回製作した GR・SAKURA 用 FM 音源シールドは、ヤマハの YM2203 というチップを搭載しており、FM 音源 3 チャンネルと PSG 音源 3 チャンネルを持っています。ここでいうチャンネル数とは同時に発声できる音の数です。あまり正確な言い方ではありませんが、和音とも呼ぶこともあります。音楽用語として、単に同時に音になることを和音とは言わないはずですが、音源の世界ではなぜか昔からこの間違った用語が横行してます。ひところは、携帯の着メロで「16 和音」とか言ってましたね。その言い方なら YM2203 は FM 音源 3 和音+PSG 音源 3 和音ということになります。

音のパラメータ

では、音のパラメータとはどんなものでしょうか？ それは、音量・音程・エンベロープ・音色の 4 つです。このうちエンベロープを音色に含めて、音量・音程・音色を音の 3 要素と言ったりもします。

■ 音量

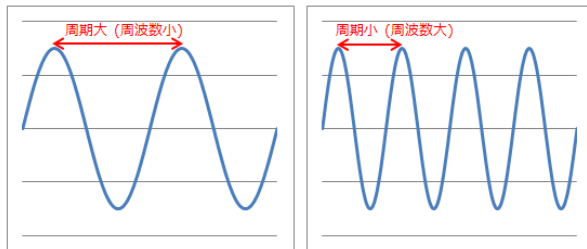
音量とは音の大きさです。音波の振幅です。まあ、これは簡単ですね。



■ 音程

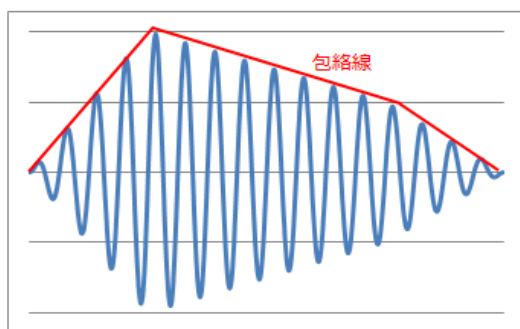
音程とは音の高さです。音波の基本周波数です。これも簡単ですね。五線譜の第二間のラ(A)の音が 440Hz です。1 オクターブ上がると周波数は 2 倍になります。そして 1 オクターブの中にある 12 の音(C, C#, D, D#, E, F, F#, G, G#, A,

A#, B)は等比数列で周波数が上がります。つまり公比は2の12乗根となります。(これを12平均律といいます。現代では12平均律が一般的ですが、2の12乗根などというワケ分からん無理数が使えなかった頃は、12平均律とは異なるピタゴラス音律などで調律されていたそうです。)



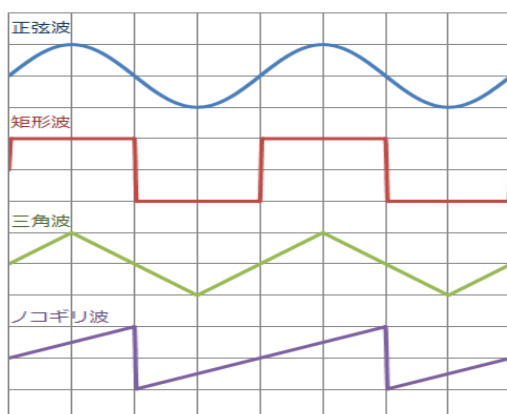
■ エンベロープ

エンベロープとは音量の時間的な変化です。音波の包絡線です。というか、包絡線を英語で envelope というのです。音量がどんなふうに立ちあがって、どんなふうに持続して、どんなふうに消えていくかです。広義には音色の一部とも考えられますが、ここでは分けるて考えることにします。詳しくは後述します。



■ 音色

音色とは、音の「感じ」です。音波の「波の形」です。いちばん記述の難しい要素です。簡単どころでは正弦波、矩形波、三角波、ノコギリ波。これらは同じ音量・同じ音程でも違う音に聞こえます。違うのは波の形、すなわち音色です。現実の楽器の音はもっと複雑な波の形をしており、それが豊かな音色となります。詳しくは後述します。



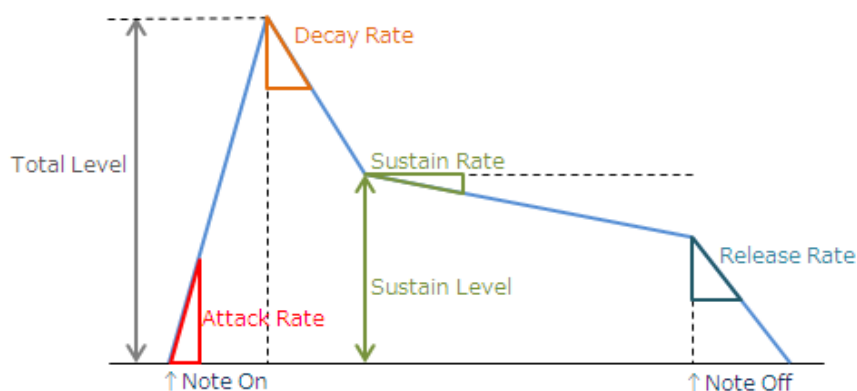
音量と音程は、FM 音源でも PSG 音源でも設定できます。そもそも、最低この 2 つが設定できないことには音楽を奏でることはできないでしょう。PSG 音源は、簡単なエンベロープの設定はできますが、音色の設定はできません。PSG 音源の音色は矩形波のみです。これに対して FM 音源は、エンベロープを PSG 音源より複雑に設定できるうえ、音色も正弦波を変調することでさまざまに設定できます。

	PSG 音源	FM 音源
音量の設定	○	○
音程の設定	○	○
エンベロープの設定	△	○
音色の設定	×	○

エンベロープ

YM2203 の FM 音源は次の 5 つのパラメータでエンベロープを設定できます。

1. Attack Rate = ノートオン(鍵盤を押したとき)からの音量の立ち上がりの傾き
2. Decay Rate = 最大音量から持続部分の音量に達するまでの減衰の傾き
3. Sustain Level = 音が持続する部分の音量
4. Sustain Rate = 音が持続する部分の音量の減衰の傾き
5. Release Rate = ノートオフ(鍵盤を離したとき)から音が消えるまでの減衰の傾き



これに対して、YM2203 の PSG 音源は、予め決められた 8 種類の簡単なエンベロープから選べるのみです。しかも、3 チャンネルの音源であるにも関わらずエンベロープ発生器は 1 系統しかありません。つまり 3 つの音を同時にさせるのに、エンベロープは別々にはかけられないのです。これはかなり残念な仕様だと思います。

音色

もっとも基本的な音の波形は正弦波です。次の式において、 A が振幅すなわち音量であり、 f が周波数すなわち音程です。これは高校の物理かなんかで習うから誰でも知ってますね。

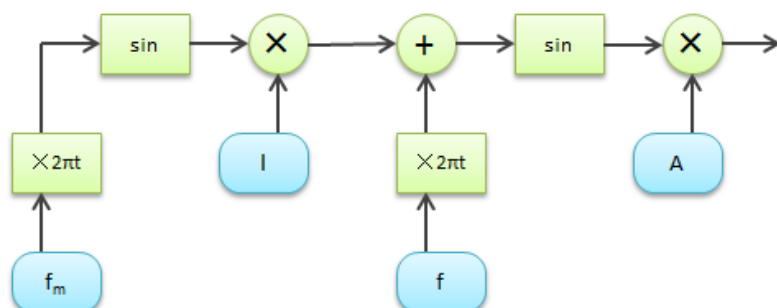
$$x(t) = A \sin(2\pi f t)$$

FM 音源では、これに次の式のような操作を加えます。 $2\pi f t$ の後に加えられた項によって周波数にゆさぶりがかけられます。これを周波数変調と呼びます。FM 音源の FM とは周波数変調(Frequency Modulation)を意味します。FM ラジオの FM と同じです。FM ラジオの場合、電波の周波数に対して、伝送したい音声信号でゆさぶりをかけます。FM 音源とはまったく異なる応用のしかたですが、数学的には同じような数式で表現できる操作です。

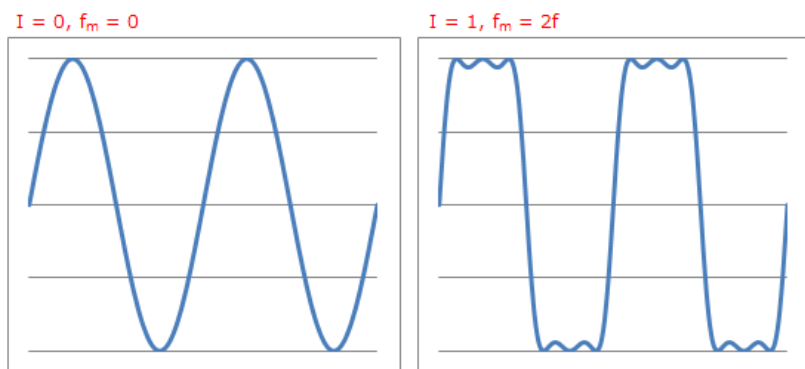
ここで、 I を変調指数、 f_m を変調周波数といいます。基本周波数に対してどれくらい大きく・どのような周期でゆさぶりをかけるかというパラメータです。

$$x(t) = A \sin(2\pi f t + I \sin(2\pi f_m t))$$

上の式をブロック図で表すと次のようになります。

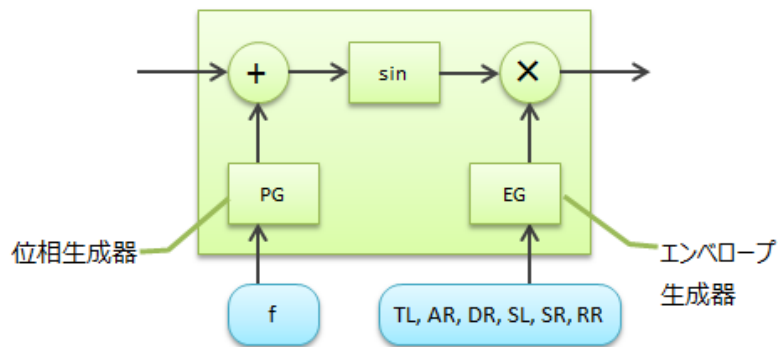


FM 音源はこのような操作によって音色を作ります。たとえば上の式において、 $I=1$, $f_m=2f$ とすると下図のような矩形波に近い波形になります。また当然ながら、 $I=0$ とすれば正弦波となります。

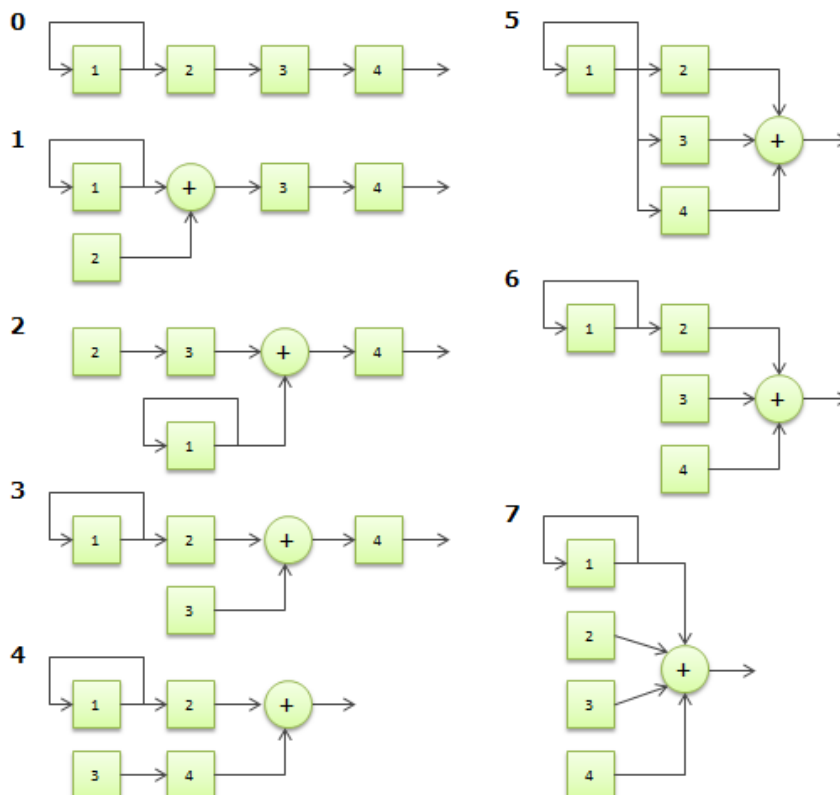


オペレータとアルゴリズム

YM2203 の FM 音源は 1 チャンネルにつき 4 個の「オペレータ」とよばれるユニットを持っています。オペレータとは、エンベロープと周波数変調の機能を持ったユニットです。これを 2 個組み合わせれば前述の基本的な周波数変調のブロック図を実現できることが分かります。



そして YM2203 では、4 個のオペレータを接続するパターンを 8 種類の中から選ぶことができます。このオペレータの接続パターンを「アルゴリズム」と呼びます。4 個のオペレータを組み合わせることによってより多彩な音色を表現できるようになります。



でもこれ、どのアルゴリズムがどのような音色を生み出すのでしょうか？ 要点は次の 3 点です。

- キャリア(最終段のオペレータ)が多いほど、つまり並列度が高いほど、音が厚くなる
- モジュレータ(前段のオペレータ)の段数が多いほど、つまり直列度が高いほど、倍音が複雑になる
- フィードバックが前段にあるほど倍音が複雑になり、後段にあるほど音にパワーが出る。

だいたい番号の若いアルゴリズムほど直列度が高く、番号の大きいアルゴリズムほど並列度が高いですね。もともと直列度の高い 0 のアルゴリズムはギターなどの撥弦楽器むきで、もともと並列度の高い 7 のアルゴリズムはオルガンやチャイムむきとことですが... う～ん、やはり経験が無いと実感が湧きませんねえ。このあたりは実際に鳴らしてみないことにはな

YM2203 のピン機能

YM2203 のピン機能についてまとめます。

GND	1	40	D0
D1	2	39	ϕS
D2	3	38	ϕM
D3	4	37	A0
D4	5	36	#RD
D5	6	35	#WR
D6	7	34	#CS
D7	8	33	IOB7
IOA7	9	32	IOB6
IOA6	10	31	IOB5
IOA5	11	30	IOB4
IOA4	12	29	IOB3
IOA3	13	28	IOB2
IOA2	14	27	IOB1
IOA1	15	26	IOB0
IOA0	16	25	#IRQ
AGND	17	24	#IC
Analog Channel C	18	23	OP-O
Analog Channel B	19	22	SH
Analog Channel A	20	21	Vdd

■ ϕM

マスタークロック入力。このクロックをもとに FM 音源と PSG 音源は動作します。最大入力周波数は 4.2MHz です。

※ 僕の FM 音源シールドでは GR-SAKURA の MTU で生成した 4MHz のパルスを入力しています。

■ ϕS , SH, OP-O

FM 音源のシリアル DAC 出力。専用の DAC YM3014 に接続します。 ϕS がクロック, SH が同期信号, OP-O が 13 ビ

ット浮動小数点シリアルデータです。

■ D0～D7, A0, #CS, #RD, #WR

プロセッサとのバス I/F。データバス、アドレスバス、チップセレクト、リード、ライトからなる、古典的な外部メモリバスです。データバス幅は 8 ビットです。しかしアドレスバスは 1 ビットしかありません。YM2203 の内部レジスタにアクセスするときはまず 0 番地に内部レジスタのアドレスを書き込んでから、1 番地を読み書きします。これによりアドレスバスのピン数を節約しています。

#CS	#RD	#WR	A0	動作
0	1	0	0	レジスタアドレスを書き込む
0	1	0	1	レジスタの内容を書き込む
0	0	1	0	ステータスを読み出す
0	0	1	1	レジスタの内容を読みだす
1	X	X	X	データバスはハイインピーダンス

※ 僕の FM 音源シールドでは、素直に GR-SAKURA の外部バスに接続しています。それほど高速なアクセスは必要ない(というか YM2203 のほうが遅い)ので、外部バス I/F を持たないマイコンでも、GPIO を 12 本使えばたぶん代用できるでしょう。

■ #IRQ

タイマー割り込み出力。YM2203 は 2 個のタイマーを持ち、一定周期でプロセッサに割り込みをかけることができます。プロセッサに YM2203 と同期して音楽演奏の処理をおこなわせるために使用します。

※ 僕の FM 音源シールドでは、そもそもプロセッサである GR-SAKURA から YM2203 にクロックを供給しているうえ、GR-SAKURA 自身が高精度のタイマ割り込みを持っているので、このピンの機能は使用しません。

■ #IC

リセット入力。LOW にすると YM2203 はリセットされ、すべての内部レジスタの内容は 0 になります。

■ Analog Channel A, B, C

PSG 音源のアナログ出力です。ソースフォロワーになっているので、抵抗を用いてミキシングできます。

■ IOA0～IOA7, IOB0～IOB7

汎用入出力ポートです。各端子ともプルアップ抵抗を内蔵しています。音源の機能とは直接関係ありません。

※ 僕の FM 音源シールドでは、使用していません。そもそも汎用入出力なら YM2203 の手を借りなくても GR-SAKURA 自身にたくさんありますし。ここらはやはり 30 年前のチップですね...

■ AGND

PSG 音源の内蔵 DAC のためのアナロググランド端子です。

■ Vdd, GND

+5V 電源端子とグランド端子です。

YM2203 のレジスタ設定

YM2203 のレジスタ設定についてまとめます。ググれば YM2203 のデータシート(英語版)は入手できますが、はっきり言ってデータシートだけではレジスタ設定についてはさっぱり分かりません。いろいろ資料を漁って照らし合わせることで情報を補完しています。まるで古文書研究ですね...

■ 表記について

表記を簡単にするため、レジスタ値については、@アドレス でそのアドレスの値とします。またレジスタ値の表記では、[ビット番号]で特定の 1 ビットを表し、[MSB:LSB] の形でビットの範囲を表すことにします。たとえば、@06h[4:0] は 06h 番地のビット 0~ビット 4 の 5 ビット値を示します。

PSG 音源のレジスタ設定

■ レジスタマップ

アドレス	値								概説
	D7	D6	D5	D4	D3	D2	D1	D0	
00h	Fine Tune								チャンネルA トーン周波数 下位
01h					Coarse Tune				チャンネルA トーン周波数 上位
02h	Fine Tune								チャンネルB トーン周波数 下位
03h					Coarse Tune				チャンネルB トーン周波数 上位
04h	Fine Tune								チャンネルC トーン周波数 下位
05h					Coarse Tune				チャンネルC トーン周波数 上位
06h				Noise Period					ノイズ周波数
07h	In/Out		#Noise			#Tone			IOA/Bの方向設定 / トーンとノイズのミキシング設定
08h				M	Level				チャンネルA エンベロープ有効 or 固定音量
09h				M	Level				チャンネルB エンベロープ有効 or 固定音量
0Ah				M	Level				チャンネルC エンベロープ有効 or 固定音量
0Bh	Fine Tune								エンベロープ周期 下位
0Ch	Coarse Tune								エンベロープ周期 上位
0Dh					C	Att	Alt	Hld	エンベロープ形状
0Eh	I/O Port A								IOAの入出力データ
0Fh	I/O Port B								IOBの入出力データ

■ トーン周波数(音程)

チャンネル A,B,C の矩形波トーンの周波数を設定します。要は音程の設定です。

```
f_tone = φM / (64 * TP)
f_tone: トーン周波数
φM: マスタークロック周波数
TP[チャンネル A] = ( @01h[3:0] << 8 ) + @00h[7:0]
TP[チャンネル B] = ( @03h[3:0] << 8 ) + @02h[7:0]
TP[チャンネル C] = ( @05h[3:0] << 8 ) + @04h[7:0]
```

■ ノイズ周波数

PSG 音源は、矩形波トーン以外に疑似乱数によるランダムノイズを発声することもできます。ランダムノイズは効果音などに用います。その周波数を設定します。

```
f_noise = φM / (64 * NP)
f_noise: ノイズ周波数
φM: マスタークロック周波数
NP = @06h[4:0]
```

■ トーンとノイズのミキサ設定

PSG 音源は、チャンネル A,B,C それぞれに対し、トーンとノイズのミキシングができます。トーン発生器は 3 系統ありますが、ノイズ発生器は 1 系統しかないので、チャンネル A,B,C で共用です。またミキシングといっても加算されているわけではなく、高速にチョップされているそうです。

```
@07h[0]: チャンネル A のトーン出力制御 (0:ON / 1:OFF)
@07h[1]: チャンネル B のトーン出力制御 (0:ON / 1:OFF)
@07h[2]: チャンネル C のトーン出力制御 (0:ON / 1:OFF)
@07h[3]: チャンネル A のノイズ出力制御 (0:ON / 1:OFF)
@07h[4]: チャンネル B のノイズ出力制御 (0:ON / 1:OFF)
@07h[5]: チャンネル C のノイズ出力制御 (0:ON / 1:OFF)
```

■ エンベロープ or 固定音量

チャンネル A,B,C の音量をエンベロープで変化させるか、固定音量にするかを設定します。

```
@08h[4]: チャンネル A の音量モード (0:固定音量 / 1:エンベロープ)
@09h[4]: チャンネル B の音量モード (0:固定音量 / 1:エンベロープ)
@0Ah[4]: チャンネル C の音量モード (0:固定音量 / 1:エンベロープ)
```

固定音量は 1,3,5,...,31 の 16 段階で設定できます。

```
音量[チャンネル A] = (@08h[3:0] << 1) + 1
音量[チャンネル B] = (@09h[3:0] << 1) + 1
音量[チャンネル C] = (@0Ah[3:0] << 1) + 1
```

■ エンベロープ周期

エンベロープの周期を設定します。なお、エンベロープ発生器は 1 系統しかないので、チャンネル A,B,C で共用です。

$$f_envelope = \phi M / (1024 * EP)$$

f_envelope: エンベロープ周波数


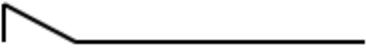


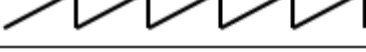

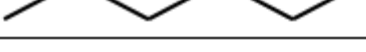
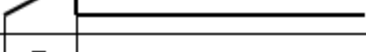
ϕM : マスタークロック周波数


$$EP = (@0Ch[7:0] \ll 8) + @0Bh[7:0]$$

■ エンベロープ形状

エンベロープの形状を 8 種類の中から選択します。

@0Dh[3:0]によって次の表のように設定します。表中の波形図の T が、前述のエンベロープ周期です。

D3	D2	D1	D0	エンベロープの形状
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	



■ IOA と IOB の方向とデータ

汎用入出力ポート機能です。これは音源とは直接関係ない機能です。アタリ規格のジョイスティックの接続によく活用されていました。

@07h[6]: IOA[7:0]の方向 (0:入力 / 1:出力)

@07h[7]: IOB[7:0]の方向 (0:入力 / 1:出力)

@0Eh[7:0]: IOA[7:0]の値

@0Fh[7:0]: IOB[7:0]の値

FM 音源のレジスタ設定

レジスタマップ概要

アドレス	値									概説	
	D7	D6	D5	D4	D3	D2	D1	D0			
21h	Test									テスト用レジスタ (アクセス禁止)	
24h	Timer A									タイマA 周期 (上位)	
25h										タイマA 周期 (下位)	
26h	Timer B									タイマB 周期	
27h	Mode	Reset	Enable	Load						チャンネル3のモード設定 / タイマA・Bの制御	
28h	Slot					Ch					キーオン/キーオフ
2Dh	この3つはアドレス 値をライトする だけで設定されるので、 データのライトは不要									クロックのプリスケール (FM:1/6, PSG:1/4)	
2Eh										クロックのプリスケール (FM:1/3, PSG:1/2)	
2Fh										クロックのプリスケール (FM:1/2, PSG:1/1)	
30h-32h		Detune	Multiple	チャンネル1,2,3 オペレータ1 Detune と Multiple							
34h-36h				チャンネル1,2,3 オペレータ3 Detune と Multiple							
38h-3Ah				チャンネル1,2,3 オペレータ2 Detune と Multiple							
3Ch-3Eh				チャンネル1,2,3 オペレータ4 Detune と Multiple							
40h-42h		Total Level		チャンネル1,2,3 オペレータ1 Total Level							
44h-46h				チャンネル1,2,3 オペレータ3 Total Level							
48h-4Ah				チャンネル1,2,3 オペレータ2 Total Level							
4Ch-4Eh				チャンネル1,2,3 オペレータ4 Total Level							
50h-52h	Key Scale		Attack Rate	チャンネル1,2,3 オペレータ1 Key Scale と Attack Rate							
54h-56h				チャンネル1,2,3 オペレータ3 Key Scale と Attack Rate							
58h-5Ah				チャンネル1,2,3 オペレータ2 Key Scale と Attack Rate							
5Ch-5Eh				チャンネル1,2,3 オペレータ4 Key Scale と Attack Rate							
60h-62h		Decay Rate		チャンネル1,2,3 オペレータ1 Decay Rate							
64h-66h				チャンネル1,2,3 オペレータ3 Decay Rate							
68h-6Ah				チャンネル1,2,3 オペレータ2 Decay Rate							
6Ch-6Eh				チャンネル1,2,3 オペレータ4 Decay Rate							
70h-72h		Sustain Rate		チャンネル1,2,3 オペレータ1 Sustain Rate							
74h-76h				チャンネル1,2,3 オペレータ3 Sustain Rate							
78h-7Ah				チャンネル1,2,3 オペレータ2 Sustain Rate							
7Ch-7Eh				チャンネル1,2,3 オペレータ4 Sustain Rate							
80h-82h	Sustain Level		Release Rate	チャンネル1,2,3 オペレータ1 Sustain Level と Release Rate							
84h-86h				チャンネル1,2,3 オペレータ3 Sustain Level と Release Rate							
88h-8Ah				チャンネル1,2,3 オペレータ2 Sustain Level と Release Rate							
8Ch-8Eh				チャンネル1,2,3 オペレータ4 Sustain Level と Release Rate							
90h-92h		SSG-EG		チャンネル1,2,3 オペレータ1 SSG Envelope							
94h-96h				チャンネル1,2,3 オペレータ3 SSG Envelope							
98h-9Ah				チャンネル1,2,3 オペレータ2 SSG Envelope							
9Ch-9Eh				チャンネル1,2,3 オペレータ4 SSG Envelope							
A0h-A2h	F-Number									チャンネル1,2,3 周波数 音名に応じた 値(下位)	
A4h-A6h			Block						チャンネル1,2,3 周波数 オクターブと 音名に応じた 値(上位)		
A8h	3ch-F-Number									チャンネル3 オペレータ4 周波数 (音声合成or効果音モード時)	
A9h										チャンネル3 オペレータ2 周波数 (音声合成or効果音モード時)	
AAh										チャンネル3 オペレータ3 周波数 (音声合成or効果音モード時)	
ACh										チャンネル3 オペレータ4 周波数 (音声合成or効果音モード時)	
ADh		3ch-Block		チャンネル3 オペレータ2 周波数 (音声合成or効果音モード時)							
A Eh				チャンネル3 オペレータ3 周波数 (音声合成or効果音モード時)							
B0h-B2h				Feedback	Algorithm	チャンネル1,2,3 フィードバックとアルゴリズム					

タイマ周期と制御

プロセッサへの割り込みを生成するタイマの設定です。タイマ A は短い周期の割り込みに、タイマ B は長い周期の割り込みに適しています。

■ タイマ A の周期

$$TA = 72 * (1024 - NA) / \phi M$$

TA: タイマ A の周期

ϕM : マスタークロック

$$NA = (@24h[7:0] \ll 2) + @25h[1:0]$$

■ タイマ B の周期

$$TB = 1152 * (256 - NB) / \phi M$$

TB: タイマ B の周期

ϕM : マスタークロック

$$NB = @26h[7:0]$$

■ タイマの制御

@27h[0]: タイマ A のスタート(1) / ストップ(0)

@27h[1]: タイマ B のスタート(1) / ストップ(0)

@27h[2]: タイマ A のオーバーフローでの割り込みを 有効(1) / 無効(0) にする。

@27h[3]: タイマ B のオーバーフローでの割り込みを 有効(1) / 無効(0) にする。

@27h[4]: 1 を書くとタイマ A の割り込みフラグがクリアされる。(このビットは即座に 0 に戻る。)

@27h[5]: 1 を書くとタイマ B の割り込みフラグがクリアされる。(このビットは即座に 0 に戻る。)

■ チャンネル 3 のモード

チャンネル 3 のみ、特別なモードを持ちます。そのモードを設定します。

@27h[7:6]	モード	動作
00	通常モード	他のチャンネルと同様の発声。
01	音声合成モード	CSM 音声合成のモード。 4 個のオペレータの周波数を独立に設定できる。 また、キーオン/キーオフはタイマ A を用いておこなう。
1X	効果音モード	4 個のオペレータの周波数を独立に設定できる。

■ キーオン/キーオフ

指定したチャンネルのキーオン/キーオフ、つまり発声のオン/オフをおこないます。またこのとき、4 つのオペレータを独立にオン/オフできます。

■ チャンネルの指定

@28h[1:0] == 00: チャンネル 1

@28h[1:0] == 01: チャンネル 2

@28h[1:0] == 10: チャンネル 3

■ オペレータごとのオン/オフ

@28h[4]: オペレータ 1 の オン(1) / オフ(0)

@28h[5]: オペレータ 2 の オン(1) / オフ(0)

@28h[6]: オペレータ 3 の オン(1) / オフ(0)

@28h[7]: オペレータ 4 の オン(1) / オフ(0)

■ マスタークロックのプリスケアラ

マスタークロックのプリスケアラの分周比を設定します。ここに関してだけは、特定のアドレス値を書き込むだけで設定されます。したがってデータの書き込みはありません。リセット時は FM 音源の分周比は 1/6、PSG 音源の分周比は 1/4 です。

アドレス	FM 音源の分周比	PSG 音源の分周比
2Dh	1/6	1/4
2Dh, 2Eh	1/3	1/2
2Fh	1/2	1/1

※ ただし、一度 FM 音源の分周比を 1/3 に設定すると、リセットしない限り 1/6 には戻せないようです。(未検証)

■ チャンネル・オペレータごとの設定

以下に述べる、Detune, Multiple, Total Level, Attack Rate, Decay Rate, Sustain Rate, Sustain Level, Release Rate, Key Scale, SSG Envelope の各パラメータは、各チャンネル・各オペレータごとに独立して設定できます。チャンネルは 3 個、オペレータは各チャンネルに 4 個ずつあるので、各パラメータには 3×4=12 個の設定値があり、それぞれに別個のアドレスのレジスタに置かれています。

各設定値のベースアドレスと、各チャンネル・各オペレータのオフセットアドレスは次の通りです。この記事の最初に載せたレジスタマップも合わせて参照してください。

ベースアドレス

Detune / Multiple	30h
Total Level	40h
Key Scale / Attack Rate	50h
Decay Rate	60h
Sustain Rate	70h
Sustain Level / Release Rate	80h
SSG Envelope	90h

オフセットアドレス

	オペレータ1	オペレータ2	オペレータ3	オペレータ4
チャンネル1	+00h	+08h	+04h	+0Ch
チャンネル2	+01h	+09h	+05h	+0Dh
チャンネル3	+02h	+0Ah	+06h	+0Eh

■ Detune(周波数のデチューン)

後述する周波数設定からのわずかなズレを設定します。

Detune = @(30h+オフセット)[6:4]

Block ※1	Note ※2	Detuneの値						
		7	6	5	4, 0	1	2	3
0	0	-0.106	-0.053	0.000	0.000	0.000	0.053	0.106
	1							
	2							
	3							
1	0	-0.159	-0.106	-0.053		0.053	0.106	0.159
	1							
	2							
	3							
2	0	-0.212	-0.159	-0.106		0.106	0.159	0.212
	1							
	2							
	3							
3	0	-0.264	-0.212	-0.159		0.159	0.212	0.264
	1							
	2							
	3							
4	0	-0.317	-0.264	-0.212		0.212	0.264	0.317
	1							
	2							
	3							
5	0	-0.370	-0.317	-0.264		0.264	0.317	0.370
	1							
	2							
	3							
6	0	-0.423	-0.370	-0.317		0.317	0.370	0.423
	1							
	2							
	3							
7	0	-0.476	-0.423	-0.370		0.370	0.423	0.476
	1							
	2							
	3							
8	0	-0.529	-0.476	-0.423		0.423	0.476	0.529
	1							
	2							
	3							
9	0	-0.582	-0.529	-0.476		0.476	0.529	0.582
	1							
	2							
	3							
10	0	-0.635	-0.582	-0.529		0.529	0.582	0.635
	1							
	2							
	3							
11	0	-0.688	-0.635	-0.582		0.582	0.635	0.688
	1							
	2							
	3							
12	0	-0.741	-0.688	-0.635		0.635	0.688	0.741
	1							
	2							
	3							
13	0	-0.794	-0.741	-0.688		0.688	0.741	0.794
	1							
	2							
	3							
14	0	-0.846	-0.794	-0.741		0.741	0.794	0.846
	1							
	2							
	3							
15	0	-0.899	-0.846	-0.794		0.794	0.846	0.899
	1							
	2							
	3							
16	0	-0.952	-0.899	-0.846		0.846	0.899	0.952
	1							
	2							
	3							
17	0	-1.005	-0.952	-0.899		0.899	0.952	1.005
	1							
	2							
	3							
18	0	-1.058	-1.005	-0.952		0.952	1.005	1.058
	1							
	2							
	3							
19	0	-1.111	-1.058	-1.005		1.005	1.058	1.111
	1							
	2							
	3							
20	0	-1.164	-1.111	-1.058		1.058	1.111	1.164
	1							
	2							
	3							

※1 音階のオクターブに相当します。(後述)

※2 オクターブ内の音程の上位 2 ビットに相当します。

■ Multiple(周波数の倍率)

後述する周波数設定からの倍率を設定します。

Multiple = @(30h+オフセット) [3 : 0]

Multipleの値	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
倍率	1/2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

■ Total Level(音量)

出力レベルすなわち音量を設定します。音量は、7ビットの設定値に比例した減衰量(デシベル表記)で設定されます。つ

まり、設定値=0 のとき音量は最大値の 0dB となり、設定値=127 のとき音量は最小の-95.25dB となります。

$$TL = @(40h+オフセット) [6 : 0]$$

TL 値のビット	D6	D5	D4	D3	D2	D1	D0
減衰量[dB]	48	24	12	6	3	1.5	0.75

■ Attack Rate

エンベロープパラメータの Attack Rate(AR)を設定します。ARは5ビットの値でキーオンからの立ち上がりの傾きを表します。AR=31 のとき傾きは最大となり、AR=0 のとき傾きは 0 となってエンベロープは立ち上がりません。

$$AR = @(50h+オフセット) [4 : 0]$$

■ Decay Rate

エンベロープパラメータの Decay Rate(DR)を設定します。DR は 5 ビットの値で最大音量からの減衰の傾きを表します。DR=31 のとき傾きは最大となり、DR=0 のとき傾きは 0(水平)となります。

$$Decay Rate = @(60h+オフセット) [4 : 0]$$

■ Sustain Rate

エンベロープパラメータの SustainRate(SR)を設定します。SR は 5 ビットの値で持続音の減衰の傾きを表します。SR=31 のとき傾きは最大となり、SR=0 のとき傾きは 0(水平)となって持続音を減衰させずに維持します。

$$Sustain Rate = @(70h+オフセット) [4 : 0]$$

■ Sustain Level

エンベロープパラメータの Sustain Level(SL) を設定します。SL は持続音のレベルを表します。持続音のレベルは、Total Level からの減衰量で設定されます。設定値=0 のとき持続音のレベルは最大値の Total Level - 0dB となり、設定値=15 のとき音量は最小値の Total Level - 93dB となります。

$$Sustain Level = @(80h+オフセット) [7 : 4]$$

SLの値	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
減衰量[db]	0	-3	-6	-9	-12	-15	-18	-21	-24	-27	-30	-33	-36	-39	-42	-93

■ Release Rate

エンベロープパラメータの Release Rate(RR) を設定します。RR は 5 ビットの値でキーオフ後の立ち下がりの傾きを表します。RR=31 のとき傾きは最大となり、RR=1 のとき傾きは最小となります。下の式のように、レジスタ設定じたいは 4 ビット値であることに注意してください。RR=0 だと音が永遠に消えないので、そうならない仕様になっています。

$$Release Rate = (@(80h+オフセット) [3 : 0] \ll 1) + 1$$

■ Key Scale

現実の楽器では、高音ほどエンベロープが短くなります。(たとえばピアノの「キンツ」という高音と「ダ～ン」という低音みた

いな?) これを再現するパラメータが Key Scale です。Key Scale は 2 ビットの値で、音程に応じて下の表のように AR, DR, SR, RR に対する補正値が決まります。

Key Scale = @(50h+オフセット) [7 : 6]

Block ※1	Note ※2	Key Scale の値			
		0	1	2	3
0	0	0	0	0	0
	1				1
	2			1	2
	3				3
1	0	1	1	2	4
	1				5
	2			3	6
	3				7
2	0	1	2	4	8
	1				9
	2			5	10
	3				11
3	0	3	3	6	12
	1				13
	2			7	14
	3				15
4	0	2	4	8	16
	1				17
	2			9	18
	3				19
5	0	2	5	10	20
	1				21
	2			11	22
	3				23
6	0	3	6	12	24
	1				25
	2			13	26
	3				27
7	0	3	7	14	28
	1				29
	2			15	30
	3				31

※1 音階のオクターブに相当します。(後述)

※2 オクターブ内の音程の上位 2 ビットに相当します。

Rate' = 2 * Rate + Rks

Rate: AR, DR, SR, RR の各設定値(0~31)

Rate': 補正後の AR, DR, SR, RR (0~63, 63 で飽和)

Rks: Key Scale による補正値

■ SSG Envelope

FM 音源のエンベロープ発生器は、これまで述べたような ARSR 式のエンベロープ以外に、SSG 音源(PSG 音源)相当のエンベロープを発生することもできます。このとき、ARSR パラメータは次のような振る舞いになります。

1. AR: かならず最大値の 31 に設定してください。

2. DR, SR, SL: キーオン中の SSG 型エンベロープのレベルと傾きを決定します。← ?
3. RR: 通常の場合と同じように、キーオフ後の立ち下りの傾きを決定します。

SSG Envelope = @(90h+オフセット)[3:0]

D3	D2	D1	D0	エンベロープの形状
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	



■ 周波数(音程)

周波数すなわち音程を設定します。設定値 Block および F-Number は、発声したい周波数から次の式で求めます。F-Number の計算で、f_note を 2^{Block} で割っていることに注目してください。1 オクターブ上がると周波数は 2 倍になりますので、同じ音名の音(たとえば C)であればオクターブが異なっても F- Number は同じになります。つまり、オクターブ内に 12 の音がある通常の音階であれば、F-Number は 12 個の値をテーブルで持てばよいことになります。

$$\text{F-Number} = 144 * f_{\text{note}} * 2^{(21-\text{Block})} / \phi M$$

f_note: 発声したい周波数

ϕM : マスタークロック

Block: オクターブ指定

$$\text{Block}[\text{チャンネル 1}] = @A4h[5:3] \gg 3$$

$$\text{Block}[\text{チャンネル 2}] = @A5h[5:3] \gg 3$$

$$\text{Block}[\text{チャンネル 3}] = @A6h[5:3] \gg 3$$

$$\text{F-Number}[\text{チャンネル 1}] = (@A4h[2:0] \ll 8) + @A0h[7:0]$$

$$\text{F-Number}[\text{チャンネル 2}] = (@A5h[2:0] \ll 8) + @A1h[7:0]$$

$$\text{F-Number}[\text{チャンネル 3}] = (@A6h[2:0] \ll 8) + @A2h[7:0]$$

■ 音声合成/効果音モード時のチャンネル 3 の周波数

チャンネル 3 を音声合成モードまたは効果音モードに設定した場合、4 個のオペレータに独立した周波数を設定できます。このとき前項の周波数設定はオペレータ 1 に適用され、オペレータ 2~4 の周波数は本項で設定します。設定値 Block および F-Number の意味は前項の周波数設定と同じです。

Block[オペレータ 4] = @ACh[5:3] >> 3
Block[オペレータ 2] = @ADh[5:3] >> 3
Block[オペレータ 3] = @AEh[5:3] >> 3
F-Number[オペレータ 4] = (@ACh[2:0] << 8) + @A8h[7:0]
F-Number[オペレータ 2] = (@ADh[2:0] << 8) + @A9h[7:0]
F-Number[オペレータ 3] = (@AEh[2:0] << 8) + @AAh[7:0]

■ フィードバック量

オペレータ 1 のセルフ・フィードバック量を設定します。各チャンネルでセルフ・フィードバックができるのはオペレータ 1 のみです。

Feedback[チャンネル 1] = @B0h[5:3] >> 3
Feedback[チャンネル 2] = @B1h[5:3] >> 3
Feedback[チャンネル 3] = @B2h[5:3] >> 3

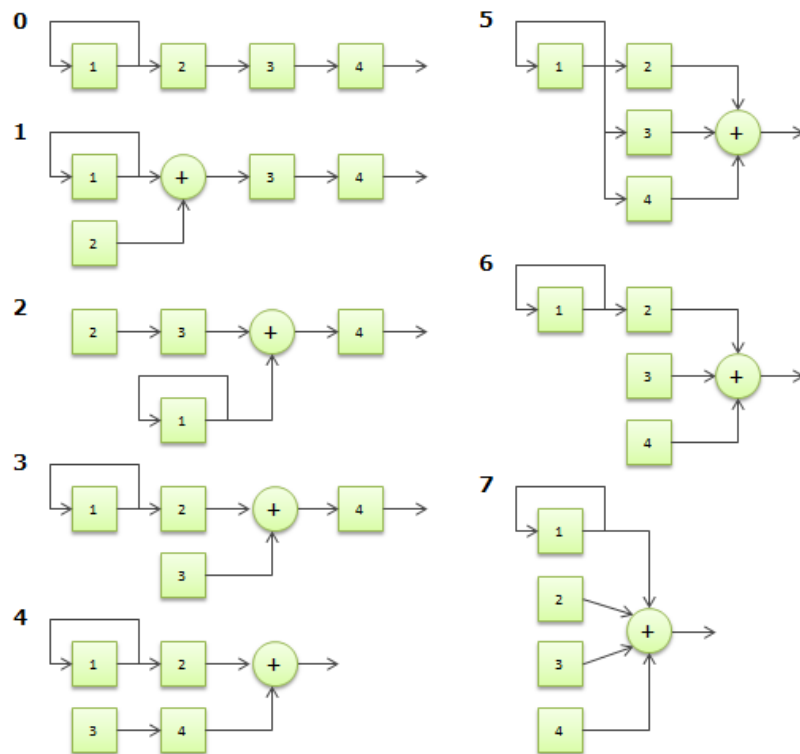
Feedbackの値	0	1	2	3	4	5	6	7
変調度	OFF	$\pi/16$	$\pi/8$	$\pi/4$	$\pi/2$	π	2π	4π

■ アルゴリズム

アルゴリズムを設定します。ここでいうアルゴリズムとは4個のオペレータの接続パターンのことです。

Algorithm[チャンネル 1] = @B0h[2:0]
Algorithm[チャンネル 2] = @B1h[2:0]
Algorithm[チャンネル 3] = @B2h[2:0]

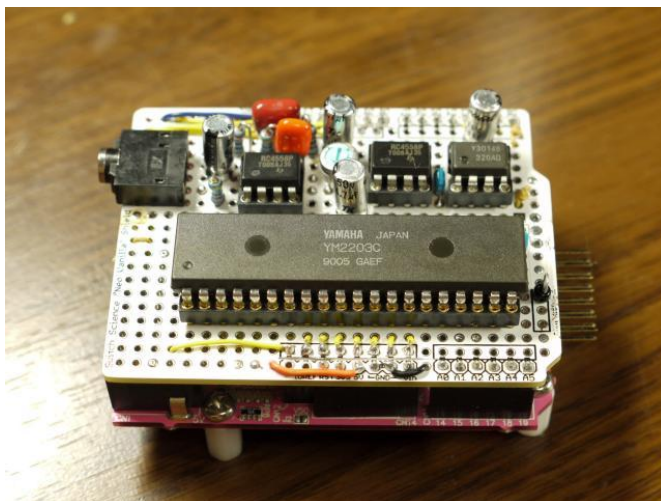
Algorithm の値とオペレータの接続パターンの対応は下図のようになります。



■ 参考文献

- YM2203 データシート (日本語版, 入手困難ですが, 下記の英語版は容易に入手できます.)
- YM2203 データシート (英語, ググればデータシート検索サイトなどで PDF が入手できます.)
- YM2608 アプリケーションマニュアル
(YM2608 は YM2203 の上位互換. 日本語版は入手できず.
これは非公式の英訳版で、自動翻訳のためかなり怪しい英語)
- YM3438 アプリケーションマニュアル (YM3438 は YM2608 の下位互換.)
- PC-9801-86 サウンドボード ユーザーズマニュアル
(PC-98 シリーズの純正音源ボード(YM2608 搭載)のマニュアル.)
- Oh!MZ 1985 年 12 月号, 1986 年 1 月号 多画正数「FM 音源ボードの製作」

特集 2 : GR-SAKURA 用 FM 音源シールド



YM2203を GR-SAKURA で制御するシールドを製作しました。回路図とソースコード(ソフトウェアライブラリおよびサンプルスケッチ)を公開しています。

ソースコード	http://licheng.sakura.ne.jp/fm_shield/FM_Shield_src.zip
回路図(PDF)	http://licheng.sakura.ne.jp/fm_shield/FM_Shield.pdf
回路図(BSch 形式)	http://licheng.sakura.ne.jp/fm_shield/FM_Shield_BSch.zip
動画(YouTube)	https://www.youtube.com/watch?v=W_pTmw4CA8c

本稿では、回路設計とソフトウェアライブラリの仕様について概説します。

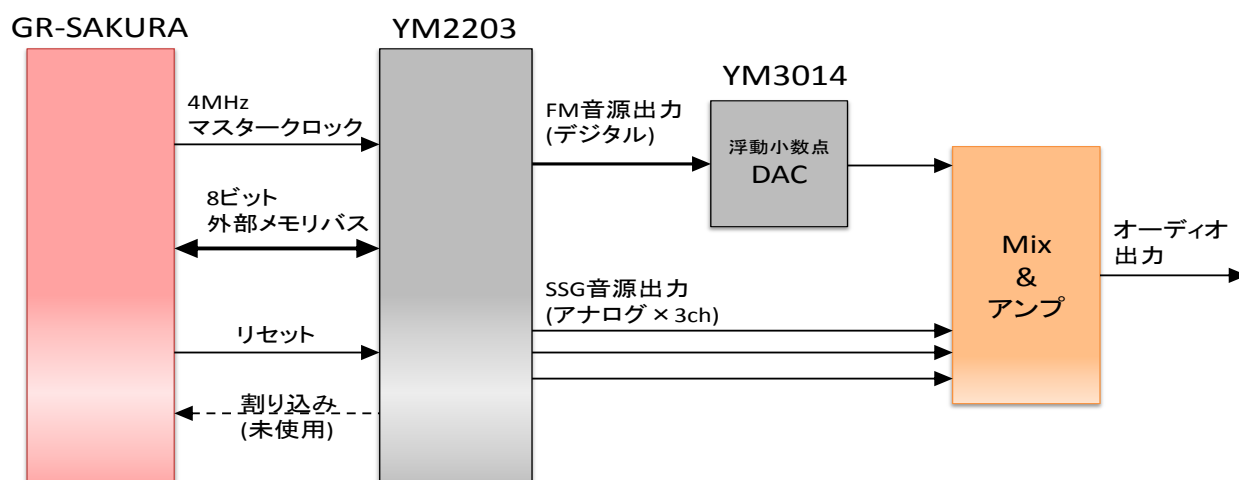
回路設計

Oh!MZ 1985 年 12 月号という 30 年近く昔の古文書に載っていた、多画正数(たが・まさかず)さんの「FM 音源ボードの製作」という記事を参考に設計しました。

概要を下図に示します。GR-SAKURA と YM2203 とのインターフェースは 8 ビットの外部メモリバスです。また GR-SAKURA は YM2203 に 4MHz のマスタークロックを供給します。GR-SAKURA はまた、YM2203 に対してリセットを出力できます。

YM2203 の出力は、SSG 音源がアナログ出力で、FM 音源がデジタル出力です。SSG のアナログ出力は各チャンネル独立の出力になっています。いっぽう、FM 音源のデジタル出力は 3 チャンネルが合成されたもので、13 ビットの浮動小数点実数のシリアル出力というかなり特殊な形式です。事実上の専用 DAC である YM3014 に接続してアナログ出力に変換します。よくわからない設計ですね。最初から YM2203 に DAC 内蔵したらよかったのにとします。きっと何か意図やいきさつがあったのでしょうか。

これら SSG 音源および FM 音源のすべてのアナログ出力を OP アンプ回路で Mix してオーディオ出力とします。



【備考】

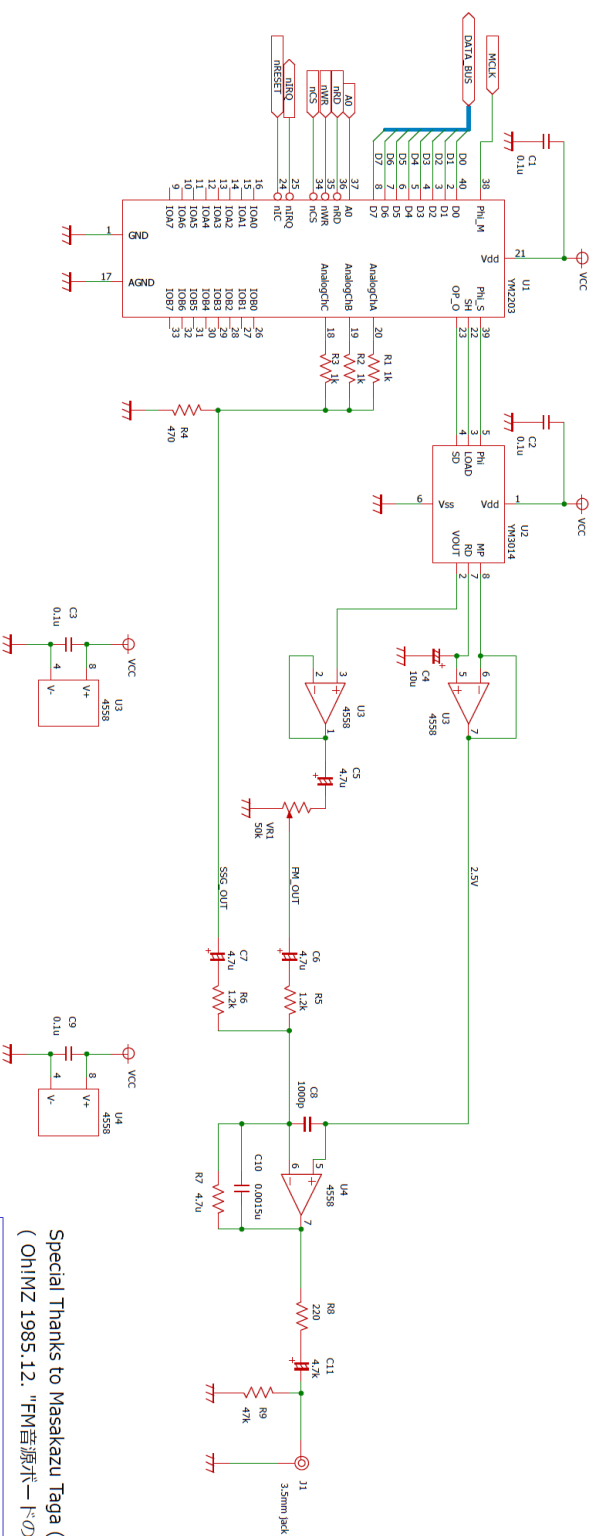
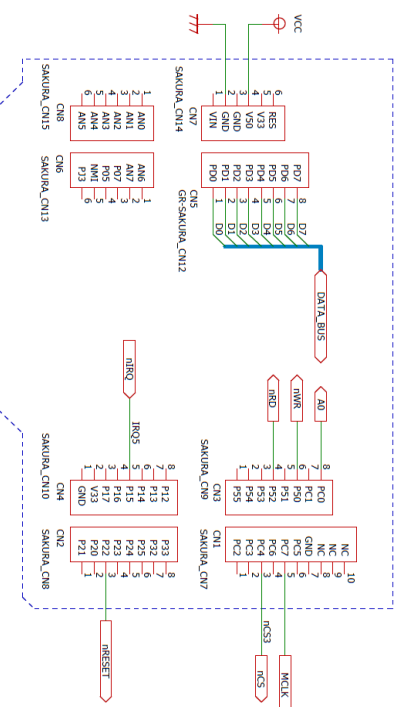
- 抵抗は金属皮膜抵抗、許容差±1%を使用。
- 電解コンデンサはオーディオ用を使用。
- 5V を使用するので、GR-SAKURA 本体に外部電源(CN1)から給電するときは J2 をショートし、5V を超える電圧は加えないこと。(USB からの給電の場合は不要。)

回路図を次ページに示します。

FM Sound Shield for GR-SAKURA with YAMAHA YM2203 (OPN)

YM2203 = Synthesizer (FM:3ch, SSG:3ch)
YM3014 = External DAC

GR-SAKURA Shield Connectors



This design is licensed under the Creative Commons Attribution-ShareAlike license.
This license allows for both personal and commercial derivative works.
as long as proper credit given to Bizan Nishimura and the design is released under the same license.

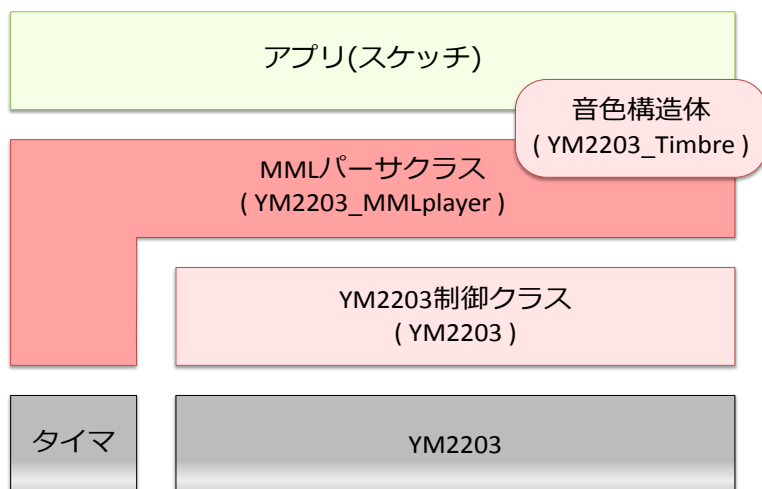
Special Thanks to Masakazu Taga (多田正数).
(OhIMZ 1985.12. "FM音源ボードの製作")

FM Sound Shield for GR-SAKURA	
with YAMAHA YM2203 (OPN)	Rev
Date: 2013/12/02	3
by Bizan Nishimura	Sheet 1 of 1

ライブラリ設計

ソフトの開発環境は HEW+GCC+E1 です。Web コンパイラでもたぶんビルドできると思いますが、未確認です。

ライブラリは、C++のクラスで設計されています。概念図を下記に示します。YM2203クラスがYM2203デバイスを直接制御します。YM2203_MMLplayer クラスは MML データを演奏クラスであり、YM2203 クラスを介して YM2203 デバイスを制御します。YM2203_MMLplayer クラスはまた、音楽のテンポ生成のためにマイコンのタイマを操作します。また、FM 音源の音色定義のために YM2203_Timbre 構造体を用意しました。



ライブラリ・リファレンス

■ ファイル

YM2203_MMLplayer.h/cpp	MML を解釈して YM2203 で演奏するプレイヤーのクラス 下記の YM2203 クラスをラップします。
YM2203.h/cpp	YM2203 を直接制御するクラス
YM2203_Timbre.h/cpp	YM2203 の音色構造体
gr_sketch.cpp	サンプルスケッチ

なお、YM2203_MMLplayer クラスではタイミング制御のために TMR0 によるタイマ割り込みを利用しています。このため、割り込みベクタとハンドラを定義している `intvect.c` に下記の修正を施す必要があります。

```
// TMR0_CMIA0
// void Excep_TMR0_CMIA0(void) { } ※1500 行めあたりのこの行をコメントアウトします。
```

■ YM2203_MMLplayer クラスの使用方法

下記ヘッダファイルをインクルードし、グローバルインスタンス MMLplayer が提供するメソッドをコールします。

```
#include "YM2203_MMLplayer.h"
```

以下に、メソッドの仕様を記します。

■ void MMLplayer.begin()

YM2203 デバイスおよび MML プレイヤーを初期化します。setup()でコールします。

```
void setup()
{
    MMLplayer.begin();
}
```

■ void MMLplayer.setTempo(int bpm)

テンポを設定します。全チャンネル共通です。

- bpm テンポ。1 分間の拍数を指定します。

```
MMLplayer.setTempo(58); // アダージョ
MMLplayer.setTempo(72); // アンダンテ
MMLplayer.setTempo(92); // モデラート
MMLplayer.setTempo(132); // アレグロ
```

■ void MMLplayer.setVolume(int ch, int volume)

チャンネルごとに音量を設定します。

- **ch** チャンネル番号。0～2 が FM 音源、3～5 が PSG 音源です。
- **volume** 音量。0～15 で指定します。0 が最小、15 が最大の音量です。
- チャンネル番号は、0～5 の数字の他、下記の定数でも指定できます。

定数名	値	意味
FM_CH1	0	FM 音源チャンネル 1
FM_CH2	1	FM 音源チャンネル 2
FM_CH3	2	FM 音源チャンネル 3
SSG_CH_A	3	SSG 音源チャンネル A
SSG_CH_B	4	SSG 音源チャンネル B
SSG_CH_C	5	SSG 音源チャンネル C


```

MMLplayer.setVolume(FM_CH1, 14);
MMLplayer.setVolume(FM_CH2, 11);
MMLplayer.setVolume(FM_CH3, 8);
MMLplayer.setVolume(SSG_CH_A, 11);
MMLplayer.setVolume(SSG_CH_B, 8);
MMLplayer.setVolume(SSG_CH_C, 8);

```

■ void MMLplayer.setEnvelope(int ch, int type, int interval)

チャンネルごとにエンベロープを設定します。(PSG 音源のみ)

- **ch** チャンネル番号。3～5 の PSG 音源のみ有効。
- **type** エンベロープの形状。(下記)
- **interval** エンベロープの周期 (0～65535)。周期は $\text{interval} * 256 \text{ usec}$ になります。

エンベロープの形状は、8～15 の値によって下図のように設定されます。表中の波形図の T が、エンベロープの周期です。エンベロープについての詳細は、FM 音源と PSG 音源の基礎と YM2203 のレジスタ設定(PSG 音源編)を参照してください。

D3	D2	D1	D0	エンベロープの形状
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

T

※ エンベロープと音量を同時に設定することはできません。MMLplayer.setVolume 関数を呼ぶとエンベロープの設定は無効になり、MMLplayer.setEnvelope 関数を呼ぶと音量の設定は無効になります。

■ MMLplayer.setToneNoise(int ch, int mode)

チャンネルごとにトーン/ノイズのミックスを設定します。(PSG 音源のみ)

- **ch** チャンネル番号。3～5 の PSG 音源のみ有効。
- **mode** トーン/ノイズ モードの指定。(下記)

PSG チャンネルの出力は、mode に下記の定数を与えることにより、(1)トーンのみ (2)ノイズのみ (3)トーン+ノイズ のい

ずれかを設定できます。

TONE_MODE	トーンのみ出力
NOISE_MODE	ノイズのみ出力
TONE_NOISE_MODE	トーン+ノイズを出力

■ void MMLplayer.setTimbre(int ch, YM2203_Timbre *timbre)

チャンネルごとに音色を設定します。(FM 音源のみ)

- ch チャンネル番号。0～2 の FM 音源のみ有効。
- timbre 音色データ構造体をポインタで渡します。

音色の設定には、YM2203_Timbre 構造体を用います。AR,DR,SR,RR,SL,TL,KS,ML,DT については、4 つのオペレータに対するパラメータをまとめて設定するメソッドを用意しました。

音色の各パラメータについての詳細は、FM 音源と PSG 音源の基礎と YM2203 のレジスタ設定(FM 音源編)を参照してください。

```
// エレキベースの音色定義
YM2203_Timbre tmbEBass;
tmbEBass.algorithm = 2;
tmbEBass.feedback = 5;
tmbEBass.opMask = MASK_ALL;
tmbEBass.setAR(31, 31, 31, 31);
tmbEBass.setDR( 8, 14, 16, 12);
tmbEBass.setSR( 0,  6,  3,  5);
tmbEBass.setRR( 0,  9,  0,  8);
tmbEBass.setSL( 3,  2,  2,  2);
tmbEBass.setTL(34, 42, 20,  0);
tmbEBass.setKS( 0,  0,  0,  0);
tmbEBass.setML( 0,  8,  0,  1);
tmbEBass.setDT( 3,  0, -3,  0);
// 音色の設定
MMLplayer.setTimbre(FM_CH1, &tmbEBass);
```

● algorithm

アルゴリズム。0～7。4 つのオペレータのつながり方を指定します。

● feedback

フィードバック。0～7。オペレータ 1 のセルフ・フィードバック量を指定します。

● opMask

オペレータマスク。0～15。4 つの各オペレータの有効/無効を指定します。

● setAR(op1, op2, op3, op4)

4つのオペレータに対して、アタック・レイトを指定します。各 0～31。

● setDR(op1, op2, op3, op4)

4つのオペレータに対して、ディケイ・レイトを指定します。各 0～31。

● setSR(op1, op2, op3, op4)

4つのオペレータに対して、サステイン・レイトを指定します。各 0～31。

● setRR(op1, op2, op3, op4)

4つのオペレータに対して、リリース・レイトを指定します。各 0～15。

● setSL(op1, op2, op3, op4)

4つのオペレータに対して、サステイン・レベルを指定します。各 0～15。

● setTL(op1, op2, op3, op4)

4つのオペレータに対して、トータル・レベルを指定します。各 0～127。

● setKS(op1, op2, op3, op4)

4つのオペレータに対して、キース・ケールを指定します。各 0～3。

● setML(op1, op2, op3, op4)

4つのオペレータに対して、マルチプルを指定します。各 0～15。

● setDT(op1, op2, op3, op4)

4つのオペレータに対して、デチューンを指定します。各 0～7

■ void MMLplayer.setGateTime(int ch, int gateTime)

チャンネルごとにゲートタイムの割合を設定します。

- **ch** チャンネル番号。0～2 が FM 音源、3～5 が PSG 音源です。
- **gateTime** ゲートタイムの長さ。1～8 で設定します。

ステップタイムに対するゲートタイムの長さの比を設定します。ステップタイムとは、音符の長さで指定される時間です。ゲートタイムとは、実際に音が出ている時間です。ステップタイム 8 に対するゲートタイムの長さで設定します。ステップタイムはつまり 1 が最も短く、8 が最も長くなります。

■ void MMLplayer.setNote(int ch, char* note)

チャンネルごとに楽譜データを MML で設定します。

- **ch** チャンネル番号。0～2 が FM 音源、3～5 が PSG 音源です。
- **note** MML 文字列をポインタで渡します。

このライブラリで利用できる MML は N-88BASIC の MML の下位互換であり、下記のコマンドに対応しています。

CDEFGAB	ドレミファソラシを表します。後に音符の長さを指定できます。たとえば 8 分音符のドなら C8。
R	休符。これも長さを指定できます。
1, 2, 4, 8, 16, 32, 3, 6, 12, 24	音符の長さ。たとえば 8 は 8 分音符。
.	付点。たとえば C4. は付点 4 分音符のドを表します。
+(#)	シャープ。たとえば C+ または C# で D# を表します。
-	フラット。たとえば C- で D \flat を表します。
O	オクターブを指定します。1~8。
L	デフォルトの音長を指定します。
Q	ゲートタイムの割合を指定します。音符の長さの n/8 のあいだ音が出力されます。
V	音量を指定します。0~15。
>	オクターブを上げます。
<	オクターブを下げます。
&	タイまたはスラー。
@	音色を選びます。 ※データは作成中
T	※未対応 テンポを指定します。BPM。
{ }	※未対応 連符。

■ void MMLplayer.play(void)

設定された楽譜データを演奏します。

このメソッドは非同期型です。演奏を開始すると関数から戻ってきます。以降はバックグラウンドで演奏を継続します。

■ void MMLplayer.stop(void)

演奏を強制停止します。

■ bool MMLplayer.isPlaying(void)

演奏中か否かを返します。

このメソッドは、MMLplayer.play() で演奏を開始した場合に、演奏の終了をチェックするのに用います。

- 戻り値 true: 演奏中 / false: 停止中

```
// 楽譜の設定
char *note1, *note2, *note3;
note1 = (char*)"L8Q704V14"
        "DBAGD4RddbAGE4REE>C<BAF+4R>DDDC<AB4RD"
        "DBAGD4RddbADE4REE>C<BA>DDDEDc<AGR>D4";
note2 = (char*)"L8Q405V13"
```

```

        "RRRRRD16C+16D16C+16DRRRRRE16D+16E16D+16E"
        "RRRRRF+16F16F+16F16F+RRRRRD16D+16E16D+16D"
        "RRRRRD16C+16D16C+16DRRRRRE16D+16E16D+16E"
        "RRRRRF+16F16F+16F16F+RRRRRD4";
note3 = (char*)"L8Q804V14"
        "G4D4G4D4G4AB>C4<G4>C4<G4A4D4A4D4GDEF+"
        "G4D4G4D4G4AB>C4<G4>C4<G4A4D4ADEF+G4D4";
MMLplayer.setNote(FM_CH1, note1);
MMLplayer.setNote(FM_CH2, note2);
MMLplayer.setNote(FM_CH3, note3);
MMLplayer.setNote(SSG_CH_A, ""); // 使用しない場合は空文字列
MMLplayer.setNote(SSG_CH_B, ""); // 使用しない場合は空文字列
MMLplayer.setNote(SSG_CH_C, ""); // 使用しない場合は空文字列
// 演奏を開始する
MMLplayer.play();
// 演奏が終わるまで LED を点滅
int blink = 1;
while( MMLplayer.isPlaying() ){
    digitalWrite(PIN_LED0, blink);
    delay(500);
    blink = 1-blink;
    // スイッチ押したら停止
    if (digitalRead(PIN_SW) == 0) {
        MMLplayer.stop();
    }
}
}

```

■ void MMLplayer.playAndWait(void)

設定された楽譜データを演奏します。

このメソッドは同期型です。演奏が終了するまで関数から戻ってきません。

```

// 演奏する
MMLplayer.playAndWait();
// 演奏が終わってから LED 点灯
digitalWrite(PIN_LED0, 1);

```



西村備山 (にしむら・びざん)

本業は組込み系のマイコン屋です。

趣味でロボットとか作ってます。

ロジカルシンキングとしての仏法に帰依してます。

ツンデレ萌え。

twitter : @lipoyang

URL : <http://d.hatena.ne.jp/licheng/>

Oh! FM音源 NT京都2014

2014年3月21日 初版発行

著者 西村備山(@lipoyang)

電気・メカ・手芸・工芸・一発ギャグ

ネタもの系ものづくり 大集合!!

NT京都2014

3.22(sat)9:30-17:00

西院春日幼稚園・春日神社境内

URL:<http://j.nicotech.jp/ntkyoto2014/>

入場無料 チップ制