ISGK Instruments アイ・エス・ジー・ケー・インストゥルメンツ

Digital Synth VRA8-P for Arduino Uno

2016.08.06-07 Maker Faire Tokyo 2016

Arduinoの限界に挑戦する「Digital Synth VRA8」シリーズ第４弾。Arduino Uno（Atmel AVR、８ビットCPU）で動く、３音擬似ポリフォニックシンセサイザー（MIDI音源）です。スケッチはフリーで公開しており、Arduino Unoと抵抗、コンデンサ、オーディオジャックだけで作ることができます。

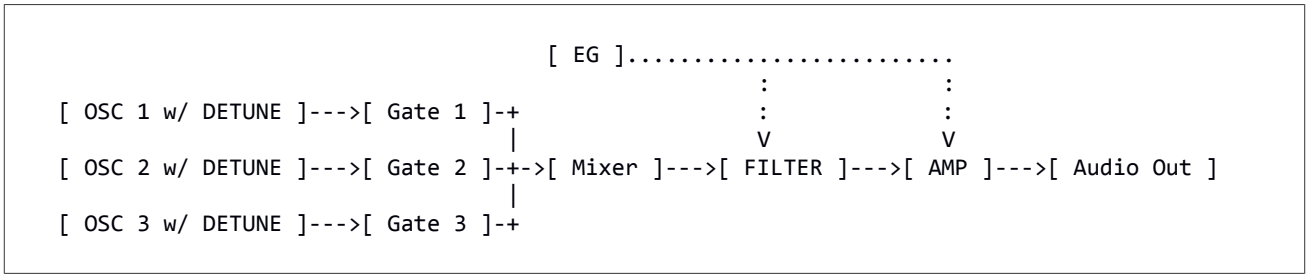
1　Digital Synth VRA8-P仕様

- Arduino Uno（8ビットCPU）で動作する，3音擬似ポリフォニックシンセサイザー（MIDI音源）
- MIDI入力：USBシリアル（38400 bps）　※USB MIDIやレガシーMIDI（31250 bps）対応改造も可能
- オーディオ出力：PWM（62500 Hz）＋RC回路（カットオフ周波数：10.6 kHz，R：150 Ω，C：100 nF）
- サンプリング周波数/ビット深度：15625 Hz/8 bit
- 音域：C1～C6（5オクターブ）　※ただし，中央のドをC4とする

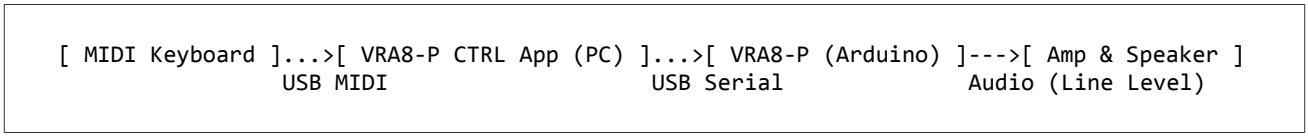
コントローラー仕様

UNISON CC16 3音ユニゾン (OFF/ON)	**OSC WAVEFORM** CC17 オシレーター波形 (SAW/ORGAN/SQ)	**OSC DETUNE** CC18 デチューン量	**AMP EG** CC19 音量の変化 (OFF/ON)
FILTER CUTOFF CC20 音の明るさ (CUTOFF FREQUENCY)	**FILTER RESO** CC21 音のクセ (RESONANCE)	**FILTER EG AMT** CC22 音の明るさの変化量 (MINUS/ZERO/PLUS)	**EG DECAY** CC23 音量・音の明るさの変化量の減衰時間

シグナルフロー図



システム構成



詳しい作り方（英語）　抵抗，コンデンサ，3.5 mmオーディオジャックだけで製作可能．ソースコードはCC0.

- http://www.instructables.com/id/Digital-Synth-VRA8-P-for-Arduino-Uno/

2　Digital Synth VRA8 シリーズ開発の経緯

作者は，Maker Faire Tokyo 2013 を見学後，2014 年に当時の勤務先で「ものづくり」の部活動に参加したことを契機に Arduino に触れるようになった．当初は明確に作りたい物が決まっていた訳ではなかったが，過去の DTM 経験や「音楽のまち」浜松市に住んでいること等から，シンセサイザー（音源モジュール）を作ることにした．

開発は，実験が行いやすいように Ruby 言語を用いて PC 上でプロトタイプを開発し，それを Arduino に移植するという手順で進めた．同時に，電子楽器に関する技術や文化的背景についての学習を進めた．そして，完成したのが最初の作品「Digital Synth VRA8」である．（詳細については，浜松 Ruby 会議 01 の発表資料「Ruby x Arduino でシンセサイザーを作ってみた」 https://gist.github.com/risgk/0db52ea683530652d933 を参照）はじめはファミコンやゲームボーイのような「8 ビット（チップチューン）」サウンドの音源をイメージしていたのだが，試行錯誤の末，出来あがったのは何故か「バーチャルアナログシンセサイザー」だった．

その後，Web Audio API による作品制作を挟みつつ，Arduino を用いた作品制作を続けてきた．技術的工夫により，シリーズ第 2 作「Digital Synth VRA8-M」では（主観的に）音質を改善することができたが，シリーズとしては必ずしも機能追加や改善を続けている訳ではなく，作品毎に違ったテーマを設定している．本日展示する「Digital Synth VRA8-P」は，シリーズ 4 番目の作品であると同時に，初のポリフォニック（和音対応）シンセサイザーである．

3　8 ビット CPU を用いたシンセサイザーの紹介

Arduino や Atmel AVR，その他の 8 ビット CPU を用いた楽器やシンセサイザー作品は数多く存在する．Arduino 環境での楽器の改造や自作，8 ビット CPU の処理能力，RAM，ROM 等の制約，PWM 出力のサウンドに対して，技術的あるいは音楽的なおもしろさを見出す制作者が多いのではないかと考える．一部の作品を以下に紹介する．

- **Auduino**　Tinker.it の Peter Knight さんの作品．Arduino 用ローファイグラニュラーシンセサイザー．
  - https://code.google.com/archive/p/tinkerit/wikis/Auduino.wiki
- **CAmiDion**　@きよし(Akiyoshi)さんの作品．PC 用 Java アプリ MIDI Chord Helper を楽器ガジェットとして「実体化」するという発想から生まれた楽器．Arduino ベース，CPU に ATmega328P を使用．
  - http://www.yk.rim.or.jp/~kamide/music/chordhelper/hardware/
- **Mozzi**　Tim Barrass さんによる Arduino 用サウンド合成ライブラリ．
  - http://sensorium.github.io/Mozzi/
- **fraAngelico**　STANDUINO（現 Bastl Instruments）社の製品．Arduino ベースのシンセサイザー．関連製品にドラムシンセサイザー frauAngelico やグラニュラーサンプラー microGranny がある．
  - http://umbrella-company.jp/standuino-fraangelico.html
- **Atmegatron**　Soulsby Synthesizers 社の製品．チップチューンサウンドを中心とした，Arduino ベースのシンセサイザー．様々なファームウェアに変更可能．関連製品に Arduino 用シールド miniAtmegatron がある．
  - http://soulsbysynths.com/
- **8bit CPU Synth**　akira matsui さんの作品．8051 系 CPU でアナログシンセの音源要素を再現．2016 年には，Roland Boutique A-01「8bit CPU Synth」として製品化された．
  - http://makezine.jp/event/maker2014/akira_matsui/
  - http://www.roland.co.jp/promos/roland_boutique/interview_3/
- **ARM Theremin**　Haruo Yamashita さんの作品．鍵盤状のトーンサークルの上にアームを回転させ，演奏する新しい電子楽器．CPU に Atmel AVR（ATmega328P）を使用．
  - http://hyamasynth.web.fc2.com/ArmThermin/ArmTerminRoot.html

4　Digital Synth VRA8-P 設計ノート

設計のポイントや残されている課題について，以下に簡単に列挙，説明する．

- **CC0 ライセンス**　ソースコードは実質的にパブリックドメイン扱いの CC0 ライセンスで公開している．そうするために，シリーズ作品はフルスクラッチ開発した．GPL や MIT ライセンスを否定する気は全くないが，完全に自由なソースコードを公開するのも，ひとつの作品発表方法だと考える．
- **8 個のコントローラー**　市販の MIDI キーボード（フィジカルコントローラー）にはノブが付いたものが色々とあるが，8 個のものが多いようだ．この認識を前提に，コントローラー（コントロールチェンジの種類）は 8 個に抑えている．少ないコントローラーによって幅広い音色の変化を実現しようとすることが，ある意味，設計で一番工夫したところかもしれない．
- **オルガン波形**　開発途中は「鋸波と矩形波のデチューン」というオシレーター波形を用意していたのだが，最後のほうでオルガン波形に入れ替えた．（ちょっと変わった仕様かもしれない）
- **アセンブラ不使用**　基本的にアセンブラを使用せず，C++（Arduino 言語）でコーディングしている．このため，ほぼ共通のコードで WAV ファイルを吐き出す PC アプリを動作させられている．
- **割り込み不使用**　通常，サウンド処理は割り込みで行うものだと思うが，この作品では敢えて割り込みを使用していない．Atmel AVR は（全レジスタを対象にする場合）レジスタ退避や復元のコストが大きいため，CPU パワーがもったいない，という判断による．アセンブラを活用すれば，効率的な割り込み処理を実現できるかもしれない．PWM 出力の切り替えタイミングは，タイマーをビジーループで監視することで判断している．
- **__attribute__((always_inline))**　基本的に Arduino のスケッチはバイナリサイズ重視で最適化されるので，この属性を指定しないと関数はインライン展開されない．（作者は，中間ファイルを確認してみるまでこの仕様に気付かなかった……）
- **ソースコードが汚い……**　最初の作品からはだいぶ改善したものの，ソースコードがあまり綺麗でない．設計ドキュメントも作っていない．新機能や新作を作る機会があれば，徐々にでも改善していきたい．
- **オシロスコープを持っていない**　波形については PC アプリで確認したり，録音すればよいのだが，GPIO を使用した CPU 負荷確認ができないのが不便である．現在は，「デッドラインを超えたら警告 LED を点灯させる」という工夫で CPU 負荷を確認している．（タイマーのカウント値をシリアル出力させるのもよいかも）
- **オーディオ出力がユニポーラ－（単極）**　この作品のように，Arduino の PWM 出力 0 から 5 V をそのまま使っているサウンドデバイスは少なくないようだが，ヘッドホンやスピーカーを直結すると当然ながら性能が活かせない．本来は，DC 成分をカットするためのコンデンサを回路に入れるか，減算回路を組んでバイポーラー（双極）出力に変換すべきだろうが，パワーアンプやヘッドホンアンプが DC 成分を（ある程度）カットしてくれることを期待して割愛している．実際に，PCM レコーダーの録音結果を確認したところ，ゼロレベルに問題はないようだった．（なお，PC での録音時にはグラウンドループに注意が必要である）
- **ソフトリアルタイム？**　割り込みを使用していない関係で，MIDI 受信をトリガーにした発音開始等のタイミングで，サウンド合成の計算が追いつかなくなる場合がある．この時，上述のように警告 LED が点灯するが，すぐに消灯するケースは基本的に，聴感上問題がないと判断している．（これをソフトリアルタイムと言っては怒られるかもしれないが……．なお，作者はこれを「8 ビット CPU の限界を超えた」とも表現している．ある種の「ゆらぎ」をもたらす効果もあるかもしれない？）
- **16 ビット乗算の近似**　Atmel AVR には 8 ビット乗算の機能しかないので，16 ビット乗算は 8 ビット乗算命令を組み合わせて実現する必要がある．16 ビット同士の乗算の場合，本来は 8 ビット乗算が 4 回必要なのだが，近似（計算誤差を許容）することで，3 回の乗算で済ませている．（なかなかきわどい設計である）

- **サンプリング周波数 15625 Hz**　最初の作品ではサンプリング周波数 31250 Hz を目指していたのだが，CPU パワーの制約により諦め，15625 Hz に変更したという経緯がある．別にローファイサウンドを目指しているわけではないのだが，CPU パワーの制約もあり，ハイファイなら良いというものでもないような気もする．
- **オシレーターでの帯域制限**　エイリアスノイズ対策のため，鋸波等の波形は指定の倍音までサイン波（倍音成分）を加算して合成している．Ruby スクリプトで計算し，プログラムメモリーのテーブルに格納．音高からどの倍音まで有効か判断．「サンプリング定理」は知っていたが，波形生成時にも問題になるとは知らなかった．
- **256 バイトの波形テーブル**　波形テーブル 1 周期分は，8 ビット×256 サンプル．低音域では量子化ノイズが目立つ．このサンプル数では理論上は 127 倍音まで記録できるが，再生時には（当然ながら Sinc 補間ではなく）線形補間しかしていないので，本当は 512 サンプル以上あったほうが良いのだろう．
- **デチューン**　各オシレーターの出力に，一定の周波数（0.24 Hz から 3.8 Hz）だけ音高をズラした出力（デチューンされた出力）をミックスすることで，コーラス効果をもたらしている．周波数を掛け算（割合）でなく足し算で計算しているので，低い音域ではデチューン量を小さく，高い音域ではデチューン量を大きく調整しないと音色が変化してしまう．（この点については，LFO を用いたパルス波の PWM と同様）
- **LPF（ローパスフィルター）**　PWM 出力を入れる RC 回路でなく，音作りのキモとなる倍音成分をコントロールするモジュールについて．いわゆる「RBJ cookbook」 http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt を参考に，16 ビット固定小数点双二次フィルター（2 次の IIR フィルター）を実装した．飽和演算は条件分岐で実現している．上述の乗算の近似の影響か，出力が振動してしまう傾向がある．
- **後着優先のノートアサイン**　3 音ポリなので，Cmaj7 コードを C，E，G，B の順に抑えると 5 度の G がオミットされる．C，E，G，B を同時に抑えると，どれがオミットされるかわからない．
- **VRA8-P CTRL**　Web MIDI API を用いるアプリケーションである．JavaScript で実装．現在のところ，Web MIDI API が使用できるのは Google Chrome のみ．
- **MIDI 規格違反 1（シリアル MIDI）**　38400 bps のシリアル通信は MIDI 規格準拠とは言えないが，DTM ブームの頃の GM 音源の多くは，このモードを備えていたように思う．
- **MIDI 規格違反 2（コントロール番号）**　本来は，予約された（Reserved）コントロール番号を勝手な目的で使用してはならない．しかし，シンセサイザー製品でも，このルールが破られることは当たり前になっている．（NRPN を扱うのが面倒だからだろうか……）
- **Arduino IDE バージョン**　この作品は Arduino IDE 1.6.8 で動作テストしているが，別の IDE バージョン（別の GCC バージョン）では CPU パワーが足りなくなって出音が変わる可能性がある．CPU パワーを限界まで使用しているため，少しのソースやバイナリの違いが動作に影響するかもしれない．微妙なところだが，作者は AVR-GCC が（ある程度）安定していることを期待している．

参考文献（一部）

- 相原 耕治『シンセサイザーがわかる本』スタイルノート（2011 年）
- 青木 直史『サウンドプログラミング入門』技術評論社（2013 年）
- 大須賀 淳『21 世紀のアナログシンセサイザー入門』秀和システム（2015 年）
- 音楽電子事業協会『MIDI 1.0 規格書』リットーミュージック（1998 年）
- 梯 郁太郎『サンプルのない時代 <ライフワークは音楽> 大幅増補改訂版』音楽之友社（2014 年）
- 小坂 直敏『サウンドエフェクトのプログラミング』オーム社（2012 年）
- 長嶋 洋一『コンピュータサウンドの世界』CQ 出版（1999 年）