MIDIシーケンサとDDSによる自動演奏システムの設計と試作

Design and Trial Manufacture of a MIDI Music Player with a MIDI Sequencer and a DDS

袴田 吉朗* 松永 康寛** 栗田 貴史**

Yoshiro HAKAMATA Yasuhiro MATSUNAGA Takafumi KURITA

Abstract: The paper describes a MIDI sequencer program using a PIC 16F877A micro computer. The sequencer reads and analyzes MIDI data stored in an I²C-EEPROM, and it sends MIDI messages to a Direct Digital Synthesizer (DDS). The DDS was designed with VHDL and fabricated using an FPGA or a CPLD, and design concept of the DDS is also given. A trial MIDI music player system was made, and it confirmed to work well.

1. はじめに

電気電子回路は,多くの機能が集約化されたICが主となって構成されている.昔と違って電子装置や回路を分解してもICが入っているだけで中身が良く分からないことが多く,恐らく一般の人には興味が湧かないし,理解が難しいと思われているようである.これが高校生に対しても電気電子が人気薄になっている一つの要因になっていると言われている.しかし逆に考えれば“動く”,“光る”,“音が出る”などの直感的な要素は,電気電子に対する拒否反応を解決する一手段ではないかとも考えられる.

この観点からH20年度には文字が "光り"ながら"動く",という2つの要素を取り込んだ16x16 LED漢字ディスプレイを設計,試作し[1],いろいろな所で使用してみて効果がありそうな感触を得た.またH21年度には,この試作した16x16 LED漢字ディスプレイを表示器に適用した赤/緑色LEDを使用するWDM光通信システムを試作してオープンキャンパスなどにおいて使用してきた[2].このシステムはプラスチック光ファイバの中を赤および緑色の“光”が点滅しながら“動き”,来学した高校生の「高校名」を伝送してLEDを用いた前述の漢字ディスプレイに表示するものである.

一方“音が出る”などの要素を含む回路,装置の検討に関しては,数年前からMIDI (Musical Instrument Data Interface)に興味を持ちサーベイを行ってきたところである.その先駆けとして昨年度には卒業研究においてDDS (Direct Digital Synthesizer)をとりあげ,VHDLを用いて設計しCPLDにより試作してその基本的な特性を評価した.また必要になる周辺回路についても試作し,特性の評価を行った.

今年度も同一のテーマを掲げたところ,幸いにもテーマ担当を希望する学生が研究室に加わった.そこで昨年度には不備であったベロシティの処理を加えると共に,従来から検討を進めてきた「PICマイコンを用いたMIDIシーケンサ」を完成させ自動演奏システムの構築を目指すことにした.

卒業研究の発表時点では一応音の出るシステムを実現できてはいたが,「デルタタイムの処理」の不具合に起因する「音飛び」があり満足できる状況になかった.しかしその後鋭意デバッグを進め,バグをフィックスすることができた.

本論文における構成は以下のようになっている.まずMIDIについてシステムを試作する上で必要不可欠となる部分に絞ってその概要を述べる.次にPICマイコンを使用したMIDIシーケンサのプログラム構成,処理について説明する.最後にVHDLを用いて設計し,FPGAを用いて試作したDDSおよびベロシティを処理するためのディジタル・アッテネータについて述べる.

なお,本研究は卒業研究の一環として行ったものであり袴田がシステム全体の取りまとめ及びPICマイコンのプログラムを,松永がDDSおよびディジタル・アッテネータの設計及び製作を,栗田がPICマイコン回路の製作および周辺回路の試作評価を担当した.システムのデバッグは全員で行った.

2. MIDIの概要[3][4][5]

SMF (Standard MIDI Format)の全体構造を図2.1に示す.ヘッダチャンクと複数のトラックチャンクからなる.本検討ではフォーマット0 (ヘッダチャンクと1つのトラックチャンクからなる)を取り扱っている.

図2.2にヘッダチャンクのフレーム構成(14バイト)を示す.フォーマット0の場合にはF=0であり,またトラック数Tr=1である.時間単位deltaは4分音符当たりのデルタタイム・チック数を表す値である.本検討で用いた「G線上のアリア」のMIDIデータではdelta=0x1E0 (480)になっている.

4D	54	68	64	00	00	00	06	00	F	Tr	delta
チャンクタイプ "MThd"				データ長 常に6				フォーマット	トラック数		時間単位

図2.2 ヘッダチャンクのフレーム構成

図2.3にトラックチャンクのフレーム構成を示す.演奏データは,MIDIメッセージにデルタタイムが前置された構成になっている.デルタタイムは図2.4に示すように可変長数値によ

4D	54	72	6B	00	00	0A	54	data
チャンクタイプ "MTrk"				データ長				演奏データ

図2.3 トラックチャンクのフレーム構成

2012年3月9日受理

*理工学部 電気電子工学科

**理工学部 電気電子工学科4年生

って表現されており，各バイトにおけるビット7が1のときは後続する下位バイトがあることを示し，0 の場合には最下位バイトを表す．

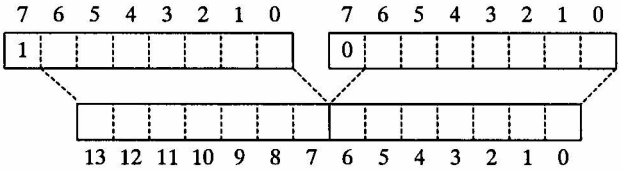

図2.4　デルタタイムやデータ長を表す可変長数値

MIDI メッセージの構成を図 2.5 に示す．第 1 バイトはステータスバイト，第 2 および第 3 バイトはデータバイトである．両者はビット 7 により区別され，ビット 7 が 1 であるバイトがステータスバイト，0 であるバイトがデータバイトである．

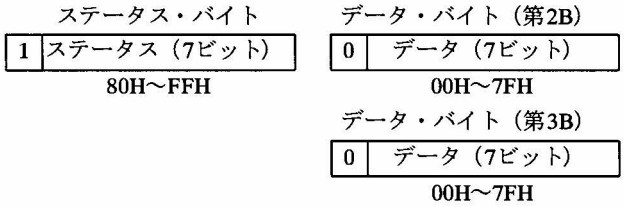

図2.5　3バイト構成のMIDIメッセージ

ステータスには複数の種類があるが，本検討ではノートオンメッセージだけを使用している．この場合のフレーム構成を図 2.6 に示す．第 2 バイトは「ノート番号」，第 3 バイトは音量に関連する「ベロシティ」である．

90H	ノート番号	ベロシティ

図2.6　ノートオンメッセージの構成

なおトラックチャンクにおける演奏データを挟み込むように制御用のメタメッセージが使用されている．メタメッセージのフレーム構成を図 2.7 に示す．「G 線上のアリア」の楽譜で使用されているメタメッセージを考慮して，以下のメタメッセージを読み出すようにプログラムを作成した．

- タイムシグナチャー　(0x58)
- キーシグナチャー　(0x59)
- シーケンス名／トラック名　(0x03)
- テンポ　(0x51)
- トラックマーカー　(0x2F)

なお実際に処理を行っているのはトラックマーカーだけであり，他は単に読み出しているだけである．

Delta Time	FF	イベントタイプ	データ長	データ

図2.7　メタイベントのフレーム構成

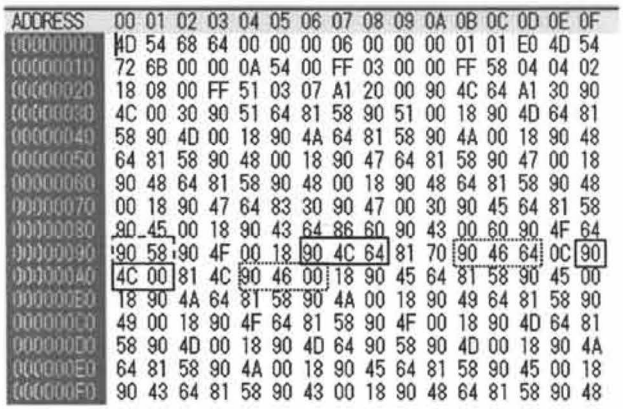

図2.8　G線上のアリアのバイナリモニタによる表示

図 2.8 にバイナリモニタにより「G 線上のアリア」の MIDI データを表示した結果を示す．先頭部分のヘッダチャンク(14B)から以下が分かる．

- フォーマット：　フォーマット 0
- トラック数　：　1
- 4 分音符当たりのデルタタイム・チック数：0x1E0(480)

また後続するトラックチャンクのヘッダからデータ数が 0xA54＝2644 バイトであることが分かる．FF で始まる 3 つのメタメッセージがあり，図 2.8 には表示されていないが最後尾にトラックマーカーを示すメタメッセージがある．

演奏データはアドレス 0x29 におけるデルタタイム=0 から始まっている．ノートオンメッセージ 0x90 に引き続きノート番号，ベロシティを出力し，デルタタイムに相当する時間が経過してからベロシティを 0 にすることによりノートオフとする構成が基本になっている．

しかしこの構成が乱れている箇所が□で囲ったアドレス 0x96 以降の部分に見られる．ここではノート番号 4C をノートオンした後，デルタタイム 240 (8 分音符) だけ経過した後，まだノート番号 4C をノートオフする以前にノート番号 46 をノートオンしている．その後ノート番号 4C，ノート番号 46 の順番にノートオフしている．

今回試作した DDS は PIC から出力されるノート番号を SET 信号の立ち上がりにおいてフリップフロップに読み込んで，次に SET 信号が来るまでその対応する周波数の信号を保持して出力するようになっている．したがってノートオフ信号の有無には関係なく，SET 信号が立ち上がったときに入力されている信号が出力される．このため両者が同時に発音する和音にはならない筈であるが，まだ十分には評価できていない．しかし両者が同時にノートオンになっている時間は 0x0C=12 チックであり，例えばアドレス 0xA7 におけるデルタタイム 0x18＝24 チ

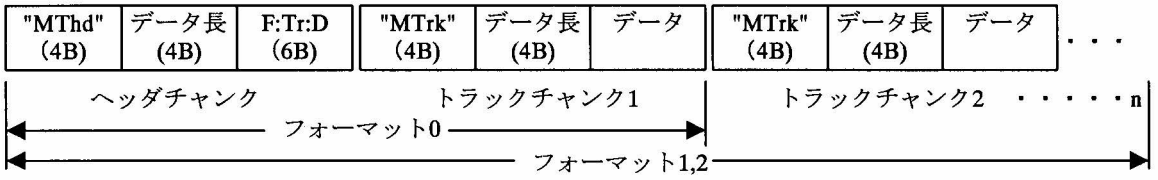

図2.1　SMFの全体構造

ックに比べても小さいので，実際には問題にならないと思われる．

またアドレス 0x90～0x91 はデルタタイムであるが，ノートオンメッセージのステータスと同じ値 0x90 から始まっている．このためプログラムではフラグ（note）を用いて両者を区別できるようにした．

図 2.8 より抜き出したデルタタイム（可変長数値）と音符の関係を表 2.1 にまとめて示した．

表 2.1 デルタタイムの表

4 分音符当たりのチック数		480	
可変長数値	16 進数	10 進数	4 分音符換算
A130	10B0	4272	8.9
30	30	48	0.1
8158	D8	216	0.45
18	18	24	0.05
8330	1B0	432	0.9
8660	360	864	1.8
60	60	96	0.2
9058	858	2136	4.45
8170	F0	240	0.5
0C	C	12	0.025
814C	CC	204	0.425

3． MIDI シーケンサ

3.1 PIC16F877A を用いた MIDI シーケンサの構成

MIDI シーケンサを PIC16F877A と I²C-EEPROM（24LC256）を用いて構成した．以下に諸元を示す．

- クロック周波数　10MHz
- ポート D　ノート番号およびベロシティ出力
- RE0　ノート番号用 SET 信号
- RE1　ベロシティ用 SET 信号
- RC4　SDA（I²C のデータ）
- RC3　SCL（I²C のクロック）
- ポート B　ICSP（インサーキット・シリアル・プログラミング用に使用）および演奏速度を調整するスイッチ用
- ポート A　LED 接続用

3.2 MIDI シーケンサにおける概略フローチャート

図3.1および図3.2にメインプログラムのフローチャートを示す．メインプログラムでは，MIDI データの EEPROM からの読み出しと，ノート番号，ベロシティの解析および DDS への出力，図 3.2 に示すデルタタイムの初期設定を主に行っている．

図 3.3 は割り込みサービスルーチン（ISR）のフローチャートである．ISR では 32μs 毎に生起するタイマー0 割り込みを用いてメインプログラムにおいて設定したデルタタイムの初期値が 0 になるまで待つ処理を行っている．以下各部分毎の処理の詳細について説明する．なお当初はプログラムの作成に当たり文献［6］を参考にさせて頂いた．

3.3 MIDI データの I²C-EEPROM への保存と読み出し

シリアル I²C-EEPROM に書き込んだ MIDI の楽譜データを，PIC マイコンを用いて読み出して解析し，DDS に対してノート番号およびベロシティを出力するのが MIDI シーケンサの基本動作である． 256K ビット（32K バイト）の I²C-EEPROM である 24LC256 を EEPROM として採用した．書き込みは，文献[7]で検討した I²C シリアル EEPROM ライターを使用して行った．後から ROM の内容を見たときの便宜を考慮して，先頭の 32B にコメントを付け加えた．このため PIC マイコンで読み出すヘッダチャンクの開始アドレスは 0x20 になる．

PIC マイコン 16F877A による EEPROM の読み出しは，マイコンを I²C ハードウェアマスターに設定して行っている．読み出し速度は 10kbaud である．なお，読み出しプログラムは，文献［8］に掲載されているプログラムを使用した．このプログラムがリロケータブル形式になっていたので，MIDI シーケンサ本体のプログラムもリロケータブル形式で作成した．

EEPROM から読み出した 16B のデータを，PIC マイコンの 0x50～0x6F 番地に設定した 32B のリングバッファに書き込み，このバッファから 1B ずつ読み出して処理を行っている．リングバッファの構成を図 3.4 に示す．

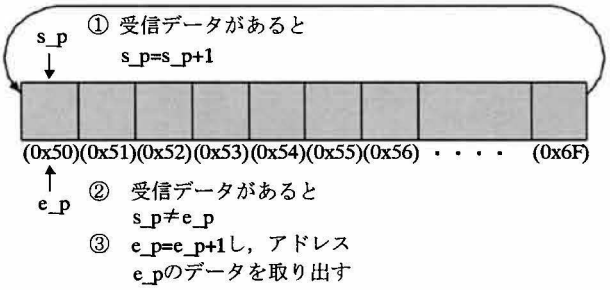

図3.4 受信リングバッファ

3.4 flags の処理

図 3.1～図 3.3 のフローチャートでは，図 3.5 に示す 3 ビットのフラグを用いて流れを制御している．

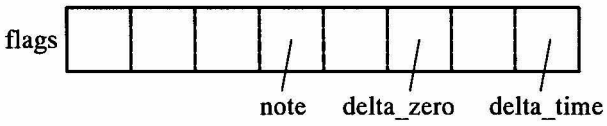

図 3.5 変数 flags によるフロー制御

各ビットの意味は以下に示すとおりである．

delta_time =1：デルタタイムを検出し，ISR において時間消費に入る

delta_zero =1：ゼロのデルタタイムを検出した

note =1：デルタタイムの処理が終了したので，次はデルタタイムではない（ノート番号，その他）

3.5 デルタタイムの処理

(1) デルタタイムにおける初期値の設定

デルタタイムの初期値の設定は図 3.1 におけるメインプログラムのフローチャートにおいて，サブルーチン「デルタタイムの設定」（図 3.2）において行っている．

デルタタイムの長さが 2 バイトであることを想定して，以下のように処理している．

① 読み出したデータ（BYT_1st）のビット 7 を検査する
② 1 であれば，2 バイト目を読み出しこれを BYT_2nd に格納
③ キャリービット C=0 の後 BYT_1st を右に 1 ビットシフトし，上位 2 ビットを 0 にマスクして結果を変数 DT0 に格納
④ キャリービット C の値を考慮して BYT_2nd を処理し，結果を変数 DT0+1 に格納
⑤ ①の結果ビット 7 が 0 の場合には DT0=0 とし DT0+1 に BYT_1st を格納する
⑥ flags ビットの処理はフローチャートに示すとおりである．

(2) デルタタイムに相当する時間の消費

デルタタイムに相当する時間の消費は，ISR において以下のように行っている．

① タイマー割り込みの周期は MIDI 規格に合わせて 32μs にしている．
② 図 3.3 のフローチャートに示したように，dt_cnt→DT0+1→DT0 の全変数が 0 になるまでの時間を待つ．変数 dt_cnt の初期値（play_dt）は現時点で 20 であり，スイッチ制御により最小 10 から最大 40 の間で変えられるようにしている．
③ デルタタイムに相当する待ち時間は式(1)のようになる．

$$32 \times play\_dt \times デルタタイム\ (\mu s) \qquad (1)$$

(3) メタイベントの処理

「G 線上のアリア」の楽譜における最後尾のデータを図 3.6 に示す．FF 2F 00 がトラックマーカーである．演奏をエンドレスで行うためにはトラックマーカー検出時に，先頭のノート番号に戻る必要があるが現時点ではまだこの処理に関するデバグができていない．

ADDRESS	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000A80	64	8D	40	90	48	00	00	FF	2F	00						

図 3.6 トラックマーカーの構成

トラックマーカー以外のメタイベントは，単に読み出しているだけである．

(4) ノート番号およびベロシティの処理

図 3.1 におけるメインプログラム（その 1）においてノート番号を検出するとこの値を変数 NOTE_NO に格納する．さらにすぐベロシティを読みに行き，これを変数 VELOCITY に格納する．その後 NOTE_NO，LE_NOTE，VELOCITY および LE_VELOCITY を図 3.7 に示すタイミングチャートに従って順次 DDS に出力する処理を繰り返すようにしている．

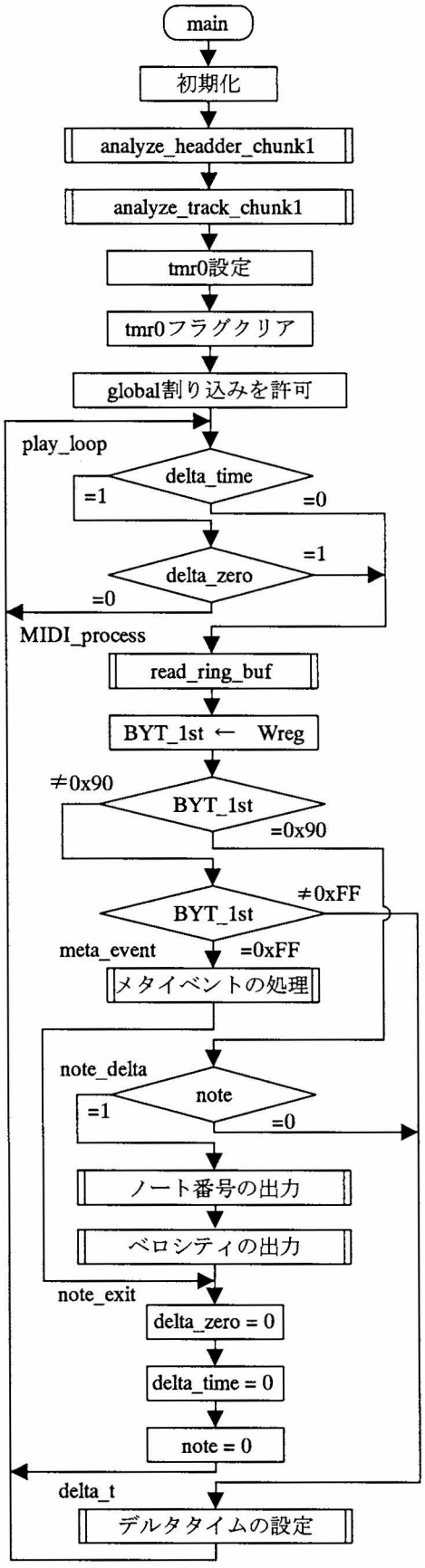

図 3.1 メインプログラムのフローチャート（その 1）

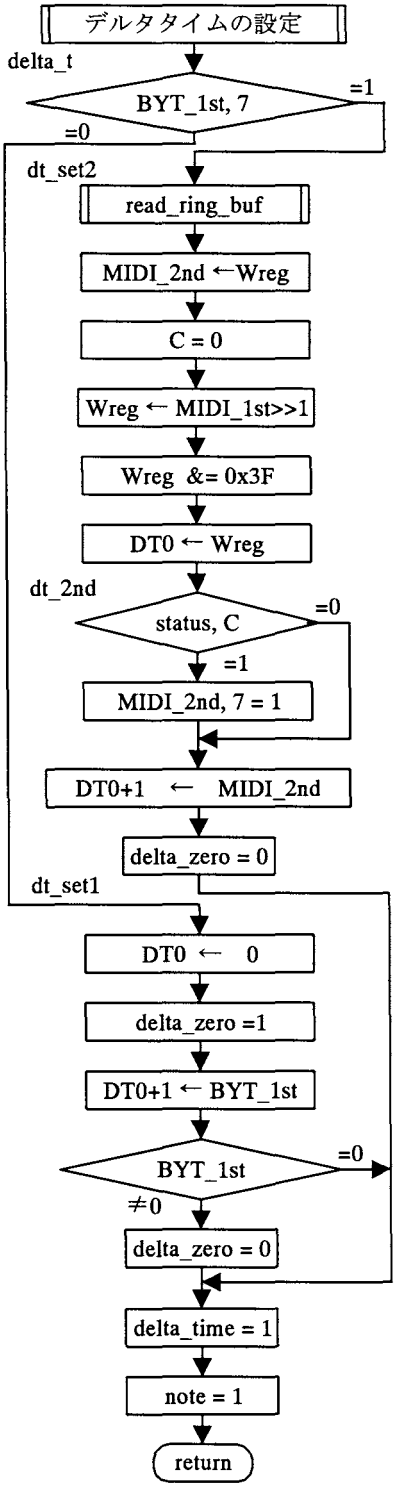

図3.2 メインプログラムのフローチャート（その2）

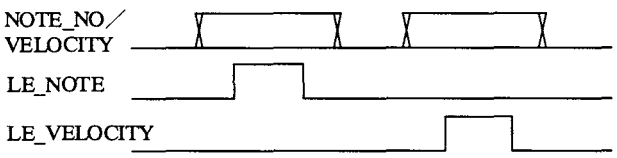

図3.7 ノート番号／ベロシティの処理

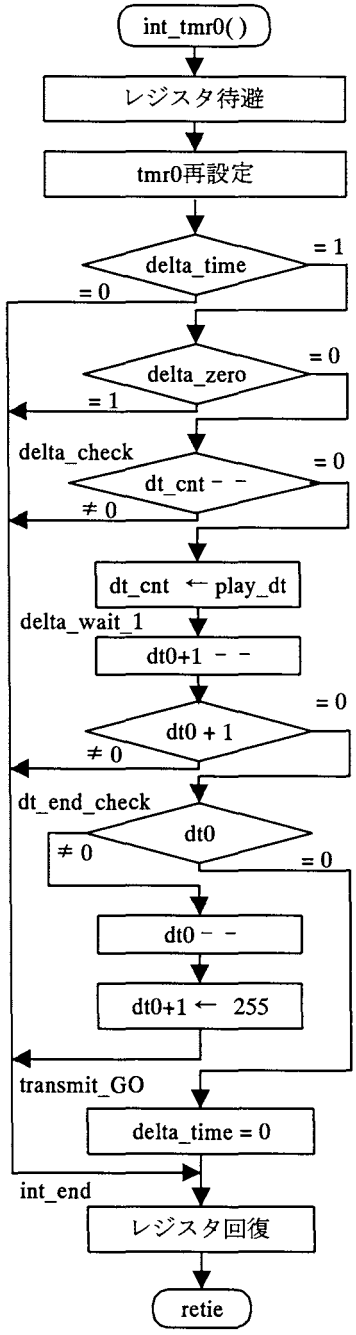

図3.3 割り込みサービスプログラムのフローチャート

ノート番号／ベロシティの立ち上がりと，SET信号の立ち上がり時間の差はPIC16F877Aの動作クロックの1クロック分（0.4μs）である．この値は検討に用いたCPLDあるいはFLEX10Kボードにおける FPGAのセットアップ時間20nsに対して十分に大きい．この値を100μsにまで大きくして動作に変化が見られるか評価して見たが，特段の変化は見られなかった．

４．DDSの設計と試作

4.1 DDSのプログラム構成

当初DDSの設計は，サイプレス社のCPLDであるCYC374i（64マクロセル）を用いて行ってきた．しかしこのCPLDはリソースが十分でなく，DDSシステムを構成するには全3チップを使用する必要があった．そこで今年度は，より大容量のCQ出版社製FLEX10Kボード（アルテラ社のEPF10K30EQC208-3使用）を用いてDDSの検討を進めてきた．このDDSでは1波形を64分割し，DDSの出力を6ビットで表示するようにした．

一方新たにアルテラ社のMAXIIMicro Board（EPM2210F324 CPLDデバイス）を用いてDDSシステムを1チップで構成して評価する検討に着手した．このシステムでは1波形を256分割し，8ビット出力とする検討を行っている．

DDSのVHDLプログラムは，①DDS本体プログラム，②ノート番号を周波数に変換するプログラム，③ベロシティを処理するディジタル・アッテネータのプログラムからなっている．

4.2 DDSにおける周波数可変の動作原理［9］

図4.1にDDSのブロック構成を示す．周波数データをアダーとラッチ回路を用いて累積加算することによりROMのアドレスを生成する．周波数データの大きさに対応してROMのアドレスの変化する周期が増減し，その結果ROMに蓄えられている基本波形と相似な波形でかつ周波数が変化した波形が得られる．この波形はディジタル信号であり，これをDA変換してローパスフィルタ（LPF）に通過させることによりアナログ信号に変換する．

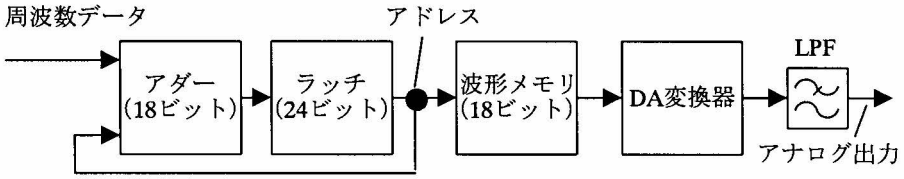

図4.1 DDSのブロック構成

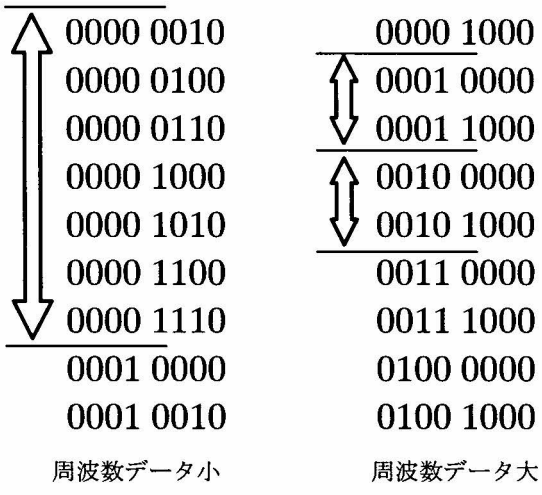

図4.2 DDSにおける周波数変更の原理

図4.2にビット数を縮減させた場合を例にして周波数可変の原理を示す．この例は1波形を16分割（4ビット出力）して取り出す例であり，周波数データが大きいほど（右側の処理）短時間に最上位から4ビット目にビットの変化が現れるので，この結果として上位の4ビットをROMのアドレスとして取り出せば周波数を可変できることが分かる．

DDSの出力波形はスイッチ切り替えにより正弦波／三角波を切り替えられるようにした．当初は1波形を64分割して検討を行ってきたが，最終的には1波形を256分割してROMに格納するようにした．したがって電圧値の量子化ビット数は8ビットになる．

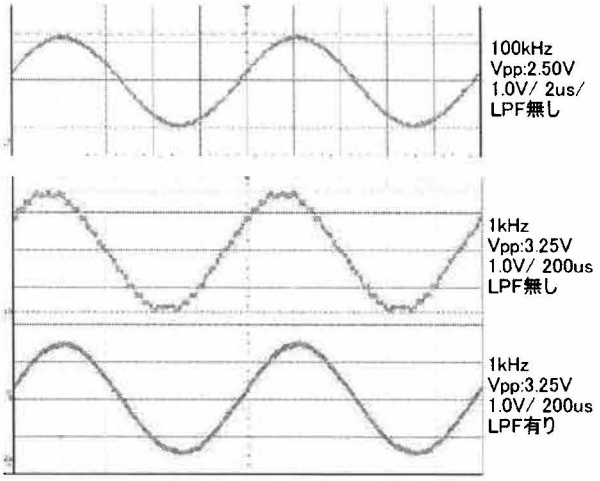

図4.3 DDSの出力波形の測定結果

図4.3にDDS出力を抵抗ラダー回路によるDA変換器を通して観測した波形を示す．使用したDDSは昨年度に検討したCPLD CYC374i上に実現したものであり，電源電圧は5Vである．

4.3 ノート番号から周波数への変換

ノート番号から周波数への変換は，ノート番号69番における周波数を440Hzとする平均律を式(1)により計算して求めた．この周波数をVHDLプログラムにおいてROMにテーブル化し，CONV_INTEGER関数を用いてstd_logi_vectorをINTEGERに変換して処理に使用している．

$$周波数=440\times 2^{\frac{NOTE\_NO-69}{12}}\ (Hz) \qquad (2)$$

ノート番号は0～127，周波数に直すと0Hz～12543Hzである．

4.4 クロック周波数の選択

nビットのアダーを用いて1Hzの周波数データを累積加算していくことを考えると，クロックを2^n個カウントしたときに1sになる．すなわちクロック周波数を2^nHzとする必要がある．

MIDI の場合には周波数データの最大値は 12543Hz であり，これを Hz 単位の 16 進数で表すと 0x30FF となる．すなわち n は少なくとも 14 ビット必要である．さらに 1 波形を 256 分割するためには 8 ビット必要であり，アダーのビット数 n を 22 ビットとした．したがってクロック周波数は 4.194304MHz になる．

後述するように，本検討ではこのクロック信号を，CMOS インバータを用いたコルピッツ型発振器を構成して CPLD に供給している．

4.5 PIC マイコンとのインタフェース

図 3.7 で示したように PIC マイコンからはノート番号／ベロシティが同じポート D から出力され，これを 2 つの SET 信号 LE_NOTE(RE0) および LE_VELOCITY(RE1) で区別している．

DDS 内部ではノート番号／ベロシティ（sdata）を受信する 2 バイトのフリップフロップを用意し，入力される sdata を SET 信号の立ち上がり時点において読み込んでいる．したがってこれらのノート番号／ベロシティのデータは，新たに SET 信号が入力されるまで保持される．

4.6 ベロシティの処理

MIDI メッセージの 3 バイト目に当たるベロシティは図 2.5 に示すように 7 ビットのデータ (0~127 の値) として出力される．これを小数点以下の値が 7 桁である 8 ビット固定小数点数（Q7 フォーマット数）と考えて処理を行うことにした．

DDS の出力は 8 ビットのオフセットバイナリ形式 (図 4.3(a)) で出力される．このため DDS 出力を図 4.3(b)に示す 2 の補数表示に変換し，Q7 フォーマットのベロシティとの積を作り，その結果を再度オフセットバイナリに変換して出力している．

これを実現するためのVHDLプログラムにおけるアーキテクチャの一部を抜粋して図4. 4に示す．まず入力データaの符号ビットを除く下位 7 ビットと，ベロシティで表される減衰比b（7 ビット）との積を求める．次に減衰比bの2の補数を計算し，入力データにおけるビット7（ a(7) ）に対応して処理を行っている．なお乗算あるいは加算とも演算子＊，＋を使用している．

```
process ( a, b) begin
    mul <= a(6 downto 0) * b(6 downto 0);
end  process;
process (a, mul, not_b, ONE ) begin
    if ( a(7) = '0' )    then
        xin <=  '0' & mul(13 downto 7) + not_b + "00000001";
    else
        xin <=  '0' & mul(13 downto 7);
    end if;
end process;
x <= (not xin(7) ) & xin(6 downto 0);
```

図4.4　ディジタル・アッテネータのプログラム（抜粋）

図4.5はシミュレーション結果である．時間の単位はnsである．1500nsまでは入力データaが正の値であり，出力xに正しく減衰比倍の値が出力されている．また1500ns以降は入力データaが負の場合である．こちらもオフセットバイナリ形式の正しい出力が得られている．なお組み合わせ論理回路で構成しており，データが変化したときに値が確定するまでに数10ns程度の時間遅れがあるが実用上特に問題になる値ではない．

5．　周辺回路

5.1 クロック発振回路

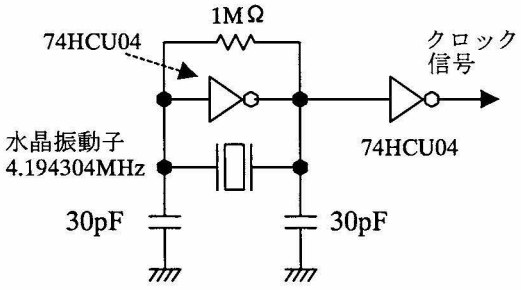

図5.1 コルピッツ型水晶発振回路

図5.1に示すようにCMOSインバータIC 74HCU04APを用いてコルピッツ型水晶発振回路を製作し，CPLD にクロックを供給するようにした．クロック周波数は前述したように4.194304MHz である．出力バッファなしのU型のインバータを使用したのは，バッファによる波形の鈍りを軽減するためである［10］．

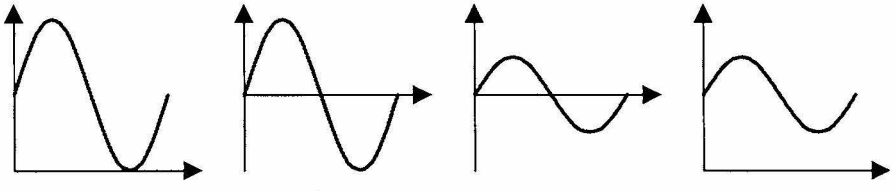

図4.3　ベロシティの処理

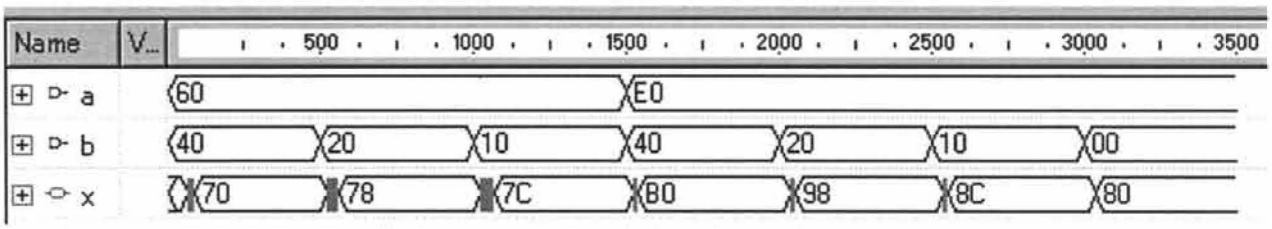

図4.5　ディジタル・アッテネータのシミュレーション結果

5.2 R−2Rラダー回路によるDA変換

DA変換器を原理的なR−2R抵抗ラダー回路網により構成した．図5.2に回路構成を示す．出力はハイインピーダンスで受ける必要があり，オペアンプNJM4580を用いたボルテージフォロワを介してローパスフィルタと接続している．入力信号のビット列をa0〜a7とし，下位桁より鳳テブナンの定理を繰り返し適用すると出力 v_o が式(3)により表される．したがってこの回路がDA変換器になっていることが分かる．

$$v_o = \frac{a7}{2} + \frac{a6}{4} + \frac{a5}{8} + \frac{a4}{16} + \frac{a3}{32} + \frac{a2}{64} + \frac{a1}{128} + \frac{a0}{256} \quad (3)$$

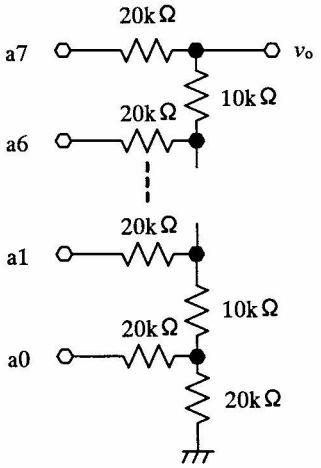

図5.2 R−2R抵抗ラダー回路によるDA変換器

5.3 2次アクティブローパスフィルタ

DA変換器の出力はまだ滑らかになっておらず，種々の高調波成分を含んでいる．したがってこれを取り除くためにオペアンプNJM4580を用いて図5.3に示す2次のアクティブローパスフィルタを構成してボルテージフォロワに後置した．この回路の高域遮断周波数 f_c および尖鋭度Qは式(4)により表される．

$$\begin{cases} fc = \dfrac{1}{2\pi R\sqrt{C_1 C_2}} \\ Q = \sqrt{\dfrac{C_1}{C_2}} \end{cases} \quad (4)$$

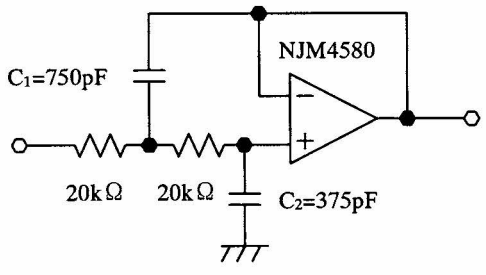

図5.3 2次アクティブフィルタ（LPF）の回路

高域遮断周波数 f_c はDDS出力の最高周波数が12.543 kHzであり，これを考慮して15kHzに設定した．また尖鋭度は経験に基づいてQ=0.7とした．

図5.4に実測したローパスフィルタの周波数特性を示す．3dB遮断周波数 f_c の実測値は13.9kHzであり，設計値との誤差率は−7%とほぼ妥当な値が得られた．

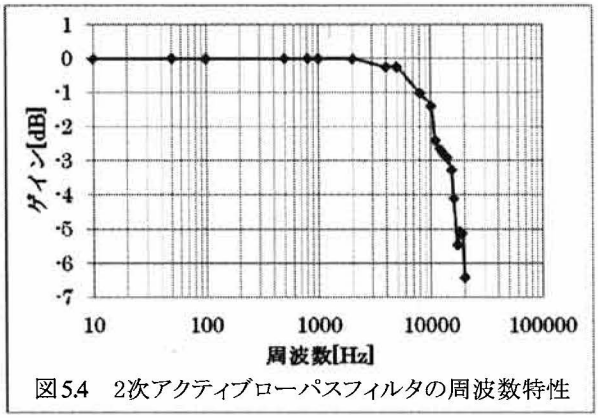

図5.4 2次アクティブローパスフィルタの周波数特性

5.4 その他の回路

上記の他に，電力増幅用のICとしてTA7275APを使用した．またCMOSインバータ74HC04を用いて負電源生成回路を製作してオペアンプの負電源を生成した．インバータ2段によりマルチバイブレータ（発振周波数約5kHz）を作り，この出力の波高値をダイオードにより約0.7Vにクランプした．さらにその出力をダイオードで半波整流して平滑し負電源を生成している．

6． 自動演奏システム

6.1 「G線上のアリア」の演奏結果

PICマイコンとDDSおよび周辺回路を接続して自動演奏システムを構成し，当初の目的である「G線上のアリア」の自動演奏ができるか否かを調べた．その結果音は出るものの，音の続く間隔がデルタタイムの値と著しく異なる箇所があり，まだ検討の余地のあることが分かった．

ADDRESS	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	00	4D	54	68	64	00	00	00	06	00	00	00	01	01	E0	4D
00000010	54	72	6B	00	00	0A	54	00	FF	03	00	00	FF	58	04	04
00000020	02	18	08	00	FF	51	03	07	A1	20	00	90	3C	64	8B	5C
00000030	90	3C	00	18	90	3D	64	8B	5C	90	3D	00	18	90	40	64
00000040	8B	5C	90	40	00	18	90	41	64	8B	5C	90	41	00	18	90
00000050	40	64	8B	5C	90	40	00	18	90	3D	64	8B	5C	90	3D	00
00000060	18	90	3C	64	8B	5C	90	3C	00	18	90	40	64	8B	5C	90
00000070	40	00	18	90	41	64	8B	5C	90	41	00	18	90	43	64	8B
00000080	5C	90	43	00	18	90	45	64	8B	5C	90	45	00	18	90	43
00000090	64	8B	5C	90	43	00	18	90	41	64	8B	5C	90	41	00	18
000000A0	90	40	64	8B	5C	90	40	00	18	90	3C	24	8E	08	90	3C
000000B0	00	18	90	3C	40	8E	08	90	3C	00	18	90	3C	50	8E	08
000000C0	90	3C	00	18	90	3C	64	8E	08	90	3C	00	18	90	3C	60
000000D0	8E	08	90	3C	00	18	90	3C	64	8B	5C	90	3C	00	18	90
000000E0	3D	64	8B	5C	90	3D	00	18	90	3D	24	8B	5C	90	3D	00
000000F0	18	90	40	64	8B	5C	90	40	00	18	90	40	24	8B	5C	90
00000100	40	00	18	90	41	64	8B	5C	90	41	00	18	90	41	24	8B
00000110	5C	90	41	00	18	90	40	64	8B	5C	90	40	00	18	90	3D
00000120	64	8B	5C	90	3D	00	18	90	3C	64	8B	5C	90	00	00	00
00000130	FF	2F	00	00	00	00	00	00	00	00	00	00	00	00	00	00

図6.1 「蛙の唄」のバイナリモニタによる表示

6.2 「蛙の唄」の演奏結果

そこで見通しを良くするためにより演奏時間が短い「蛙の唄」のMIDIデータを作り，演奏を行ってみた．状況に特段の変化は見られなかった．図6.1にバイナリモニタを用いて表示した「蛙の唄」のMIDIデータを示す．特にデータに異常な箇所は見あたらなかった．そこでロジックアナライザを用いてPICマイコンにおけるノート番号／ベロシティの出力であるポートDを観測してみた．しかしながら，元来ノート番号／ベロシティは近接して出力されるため，様子が十分に把握できなかった．

そのためプログラムを変更してノート番号のみがポートDに出力されるようにして，再度ロジックアナライザを用いて出力を観測してみた．その結果を図6.2に示す．D_0～D_7はノート番号，D_8はSET信号（LE_NOTE）である．

図6.2において①は0x3Cであり，これは図6.1のデータにおける最初のノート番号である0x3Cに対応している（アドレス0x2C番地）．

次の②は0x3Dであり，これは2番目のノート番号であるアドレス0x35番地のノート番号に対応している．但しこの①―②間のデルタタイムは，0x2E～0x2F間の値から本来0.96sとなる筈であるが，実際には5.95sになっている．さらに③のノート番号は0x3Cであり，MIDIデータと符合している．しかし③―④間の時間はデルタタイムから1.3sになるところが，約7.5sになっている．

また⑤の音は，図6.1から本来0x3Cであるべき筈であるが，ノーオフされていない．したがってアドレス0x124番地の0x3Dは読み出されていず，0x130番地におけるトラックマーカーも検出できていない．

以上の結果を整理して以下の知見が得られた．

◎ 異常な箇所は2ヶ所ともデルタタイムである．

◎ しかも図6.1においてアドレスの0E番地に位置している

図6.1はMIDIの楽譜データである．これがEEPROMに書き込まれて，再度読み出されてリングバッファに書き込まれるときに，ポインタs_pの処理の関係からアドレスが1だけ大きくなる．このため図6.1のアドレス0E番地のタイミングは，次の16バイトのデータをI^2C-EEPROMから読み出しに行くタイミングに相当する．この観点から図3.1および図3.2におけるサブルーチンread_ring_bufの動作を再度詳細に検討してみた．

6.3 サブルーチンread_ring_bufの詳細検討

(1) MIDIデータをプログラムメモリから読み出しデバッグ

図6.3がI^2C-EEPROMのデータを読出すフローチャートである．特段の不具合はないが，サブルーチンread_I2CEEPROM_write_ring_bufの後ろに，プログラム作成における初期の段階でデバッグに用いていたPICマイコンの内部EEPROMへの書き込みルーチンが余分についていることが分かった（図示していない）．この部分は最終的に削除した．

なお今までに行ってきたデバッグでは，サブルーチンread_I2CEEPROM_write_ring_bufの部分に来ると処理をスキップして，その代わりにデータを手入力でデバッガに入力してデバッグを行っていた．このやり方では処理を完全にデバッグしているとは言えず，また手間もかかるので，図6.3に示したフローチャートにおいて，サブルーチンread_I2CEEPROM_write_ring_bufを変更して，MIDIデータをプログラムメモリに保存しておきこれを読み出すようにしてデバッグを行ってみた．

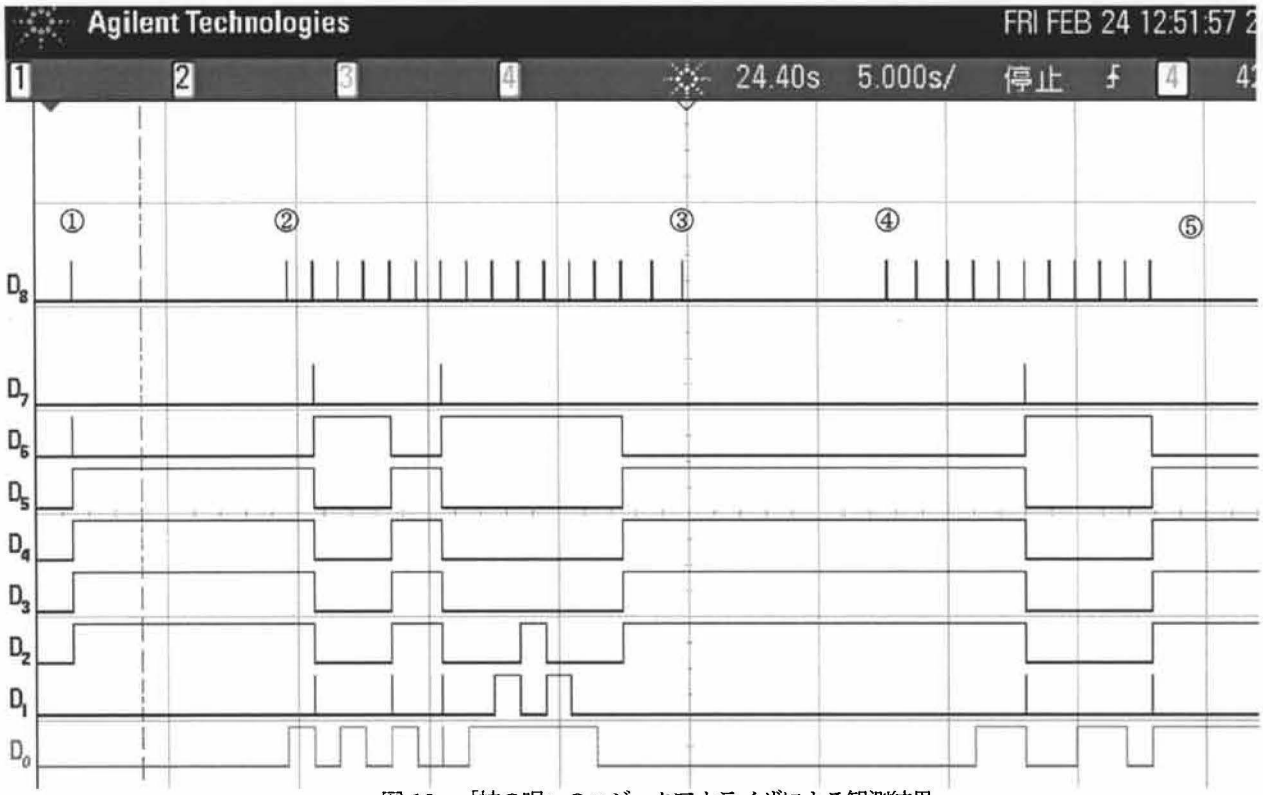

図6.2 「蛙の唄」のロジックアナライザによる観測結果

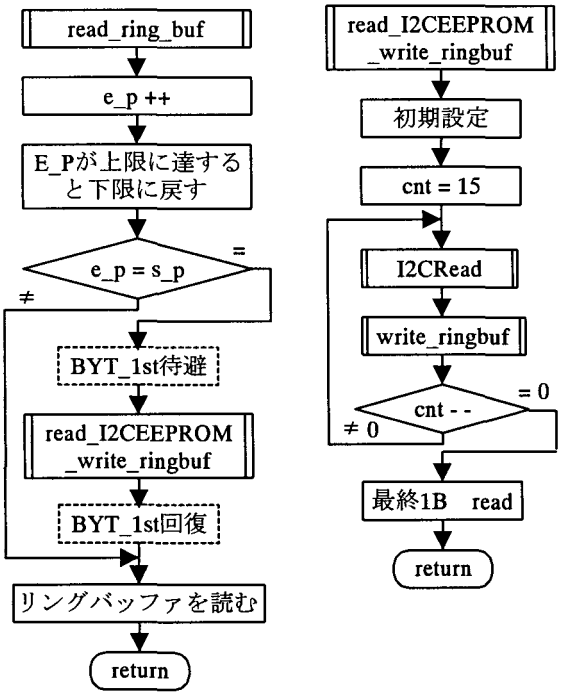

図6.3 read_ring_buf のフローチャート

(2) デバッグ結果

図6.3 におけるフローチャートの動作を以下に示す.

① 図6.3 (左側) においてポインタが e_p = s_p になると次の16バイトのデータをプログラムメモリから読み出し, リングバッファに格納する.

② ただし①の時点ではリングバッファから読み出した15バイト目のデータの処理はまだ行われていない.

③ ①が終了すると, リングバッファからまだ読み出していない16バイト目のデータを読み出す.

④ 15バイト目のデータがデルタタイムであるとき, ②の終了後デルタタイムの時間消費に入るが, このデルタタイムの処理に使われるデータが, ①で読み込んだデータによって上書きされている. これが不具合の原因であった.

そこで図6.3 において破線で示したように命令を一部追加して, サブルーチン read_I2CEEPROM_write_ring_buf の実行前に変数 BYT_1st を待避し, 実行後に回復するようにしてみた. これによりプログラムが期待通りに動作することを確認した.

しかしノートオフ時にスピーカから「ボコ」と言うような音が出ており, 耳障りであるのでこの音を軽減する方法を検討する必要がある.

7. むすび

高校生に電気電子に対する関心を持って貰う手段として "動く", "光る", "音が出る"などの直感的な要素を含む装置を試作して, デモを行うことが考えられる. この観点から"音が出る"要素を含む回路として MIDI 自動演奏システムについて検討した. すなわち PIC マイコン 16F877A をコントローラとする MIDI (Musical Instrument Data Interface) シーケンサを試作し, プログラムの詳細を示した. また音源を DDS (Direct Digital Synthesizer) により実現するために, その動作原理と VHDL により設計し FPGA あるいは CPLD により DDS を試作した結果を示した. 必要となる周辺回路の設計, 試作結果も合わせて示した.

これらの要素を組み合わせて MIDI 自動演奏システムのプロトタイプモデルを構成し, I^2C-EEPROM に書き込んだ「G線上のアリア」あるいは「蛙の唄」を読み出して自動演奏ができることを確認した.

なお以下のような検討課題が残った.

1) より複雑な MIDI データの演奏ができるようにする
2) ノートオフ時にスピーカから「ボコ」と言うような音が出ているが, この音を軽減する方法を検討する
3) 今回試作した DDS は, 周波数を 1Hz～12543Hz に変化させた正弦波の発音ができる. これを実際の楽器のデータに近づける方法を検討する
4) 演奏速度の調整法を検討する
5) 演奏する曲目数の増加を検討する

以上のような検討課題について今後検討を進めていく予定である.

[参考文献]

1) 袴田吉朗, "PIC マイコンと 16×16LED を用いた漢字表示電光掲示板の設計と試作", 静岡理工科大学紀要, Vol.17, pp.133-142, (2009)
2) 袴田吉朗, "展示用波長多重光通信システムの設計と構築", 静岡理工科大学紀要, Vol.18, pp.21-30, (2010)
3) 中島安貴彦, "MIDI バイブル I MIDI1.0 規格基礎編",
4) "詳説 MIDI 規格", http://www.pluto.dti.ne.jp/~daiki/Midi/Midi.html
5) "SMF の基礎知識", http://www.hikari-ongaku.com/study/smf.html
6) "MIDI シーケンサー (レコーダー/プレイヤー) の作成", http://www.mars.dti.ne.jp/~ogura/e_hobby/midirec.html
7) 袴田吉朗, "I^2C EEPROM ライターの設計と漢字ディスプレイへの応用", 静岡理工科大学紀要, Vol.18, pp.21-30, (2010)
8) "マイコンの1線, 2線, 3線インタフェース", CQ 出版社 (2008)
9) 後閑哲也, "電子制御のための PIC 応用ガイドブック", 技術評論社 (2002)
10) 稲葉 保, "定本発振回路の設計と応用", CQ 出版社 (2000)

[付録] 最終的なプログラムの所在

本資料における PIC プログラムは 2012.3.5 にデバグを完了したプロジェクト midirec20120305 に基づいて作成した. またディジタル・アッテネータは 2012.2.29 作成の digi_att_adder_new に基づいている.