MIDIシーケンサおよび3音源に対応した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 Playing System with a MIDI Sequencer and a DDS

袴田 吉朗* 萩山 友皓** 大橋 龍馬**

Yoshiro HAKAMATA Tomohiro HAGIYAMA Ryoma OHASHI

Abstract: The paper describes a MIDI sequencer program using a PIC 16F877 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 includes three sonic sources and was designed with VHDL. The operation of the DDS was simulated with the ModelSim simulator. The DDS was fabricated using a Cyclone Ⅳ FPGA. A trial MIDI music player system was made, and it confirmed to perform "Entertainer" well..

1. はじめに

高校生に電気電子に興味を持って貰うことを目的にしてこれまでに"動く", "光る", "音が出る"などの直感的な要素を取り込んだ電子回路を作り，報告してきた [1][2]. その一貫として"音が出る"要素を含む回路，装置に関しては，H23年度の卒業研究において「PICマイコンを用いたMIDIシーケンサ」を完成させ単音源のDDSと組み合わせて「G線上のアリア」の演奏ができる自動演奏システムの構築ついて報告した[3].

本報告は[3]の報告を発展させ多音源化を実現することによって自動演奏システムの汎用性を高めることを目的に行った検討結果をまとめたものである. MIDIの楽譜には3音源から構成されている「エンターテイナー」を選んだ. これは音源数として手頃な数であること，軽快なリズムで構成されており聞いていて楽しい楽曲であると独善的に判断したことがこの楽曲を選定した理由である.

MIDIシーケンサはPIC16F877を，DDSはCycloneⅣを搭載したFPGAボードを使用して開発を行った. 卒業研究発表会における時点で「エンターテイナー」を自動演奏できるシステムに出来上がっていることを確認できたので，一定の区切りを付けるために資料にまとめておくことにした. 本資料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3音源DDS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，萩山がPICマイコンのプログラムのデバッグおよびPICマイコンに関する回路の製作を，大橋が3音源DDSの設計，シミュレーションおよびFPGAによる実現を担当した. システムのデバッグは全員で行った.

2014年2月26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

2. MIDIの概要[4][5][6]

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「エンターテイナー」の楽譜ではdelta=0x1E0（480）になっている.

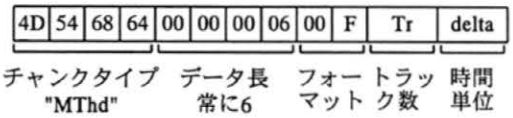

図2.2 ヘッダチャンクのフレーム構成

図2.3にトラックチャンクのフレーム構成を示す.

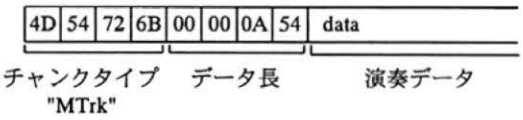

図2.3 トラックチャンクのフレーム構成

演奏データは，MIDIメッセージにデルタタイムが後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.

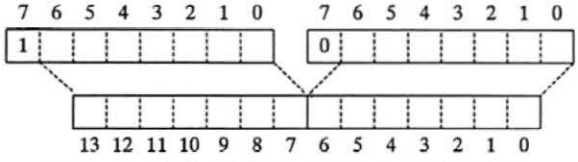

図2.4 デルタタイムやデータ長を表す可変長数値

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．

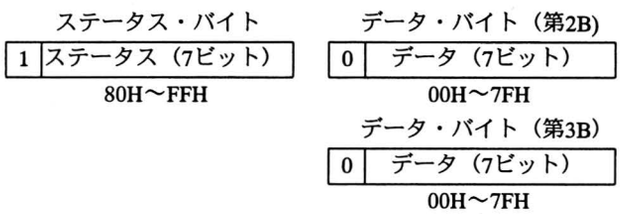

図2.5　3バイト構成のMIDIメッセージ

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およびノートオフ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．nはチャネルであり，「エンターテイナー」ではn=0，3およびA（16進数）が用いられている．

9nH	ノート番号	ベロシティ

8nH	ノート番号	ベロシティ

図2.6　ノートオンおよびノートオフ
メッセージの構成

```
ADDRESS   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  4D 54 68 64 00 00 00 06 00 00 00 01 01 E0 4D 54
00000010  72 6B 00 00 45 DF 00 F0 07 7F 7F 04 01 00 7F F7
00000020  00 BA 00 00 00 B0 00 00 00 B3 00 00 00 F0 05 7E
00000030  7F 09 01 F7 00 F0 07 7F 7F 04 01 00 7F F7 00 CA
00000040  01 00 C0 01 00 C3 01 00 FF 58 04 04 02 12 08 00
00000050  BA 07 73 00 B0 07 73 00 B3 07 5F 00 FF 59 02 00
00000060  00 00 BA 5B 64 00 B0 5B 64 00 B3 5B 6E 00 FF 51
00000070  03 09 3F 30 00 BA 5D 3C 00 B0 5D 3C 00 B3 5D 3C
00000080  00 BA 4A 78 00 B0 4A 78 00 B3 4A 64 00 BA 65 00
00000090  00 B0 65 00 00 B3 65 00 00 BA 64 00 00 B0 64 00
000000A0  00 B3 64 00 00 BA 06 0C 00 B0 06 0C 00 B3 06 0C
000000B0  00 BA 0A 18 00 B0 0A 68 00 B3 0A 40 00 BA 65 00
000000C0  00 90 56 40 00 93 24 40 00 BA 64 01 00 BA 06 38
000000D0  00 BA 26 00 3C 9A 56 40 2D 80 56 40 0F 90 58 40
000000E0  2D 8A 56 40 0F 9A 58 40 2D 80 58 40 0F 90 54 40
000000F0  2D 8A 58 40 0F 9A 54 40 2D 80 54 40 0F 90 51 40
```

図2.7　エンターテイナーのバイナリモニタによる表示

図2.7にバイナリモニタにより「エンターテイナー」の楽譜の冒頭部分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0x45DF＝17887バイトであることが分かる．

演奏データはアドレス0xC1におけるノートオンメッセージ90Hから始まっている．引き続きノート番号，ベロシティを出力している．続いてデルタタイム=0でノートオンメッセージ93Hが発音している．最初のノートオンメッセージ90Hが消音するのはアドレス0xD9における80Hにおいてである．2番目の93Hに対するノートオフは，図2.7の中には表示されていない．このように個々の音のノートオン－ノートオフが「入れ子構造」になった複雑な楽譜データになっており，これは「G線上のアリア」の楽譜とは著しく異なるところである．

アドレス0x17～0xD7においてFFで始まる3個のメタメッセージがあり，さらに図2.7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．メタメッセージのフレーム構成を図2.8に示す．

FF	イベントタイプ	データ長	データ	Delta Time

図2.8　メタメッセージのフレーム構成

「エンターテイナー」の楽譜では，以下のメタメッセージが使用されているので，これ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)・・2／4拍子
- キーシグナチャー　(0x59)・・ハ長調
- テンポ　(0x51)・・606μs／四分音符
- 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．

F0で始まる3個のシステム・エクスクルーシブも見られるが，これも単に読み出す処理を行っているだけである．

アドレス0x3Fから0x47まではプログラム・チェンジの指定である．この内容は各音源とも「ピアノ」とする指定である．

以上のメッセージ以外にBnH（n=0,3,A）で示されるコントロールチェンジが多数見られる．これらは3個の音源に対して同一の値が設定されており，以下に示す内容が指定されている．

- バンクセレクト　音色の選択
- 音量の指定　音量の設定
- エフェクトデプス1　リバーブセンドレベル設定
- エフェクトデプス3　コーラスセンドレベル設定
- ブライトネス　音の明るさの設定
- データエントリ

最後のコントロールチェンジは，最初の発音後に食い込んでいる．これらのメッセージも今回の検討では単に読み出しているだけである．

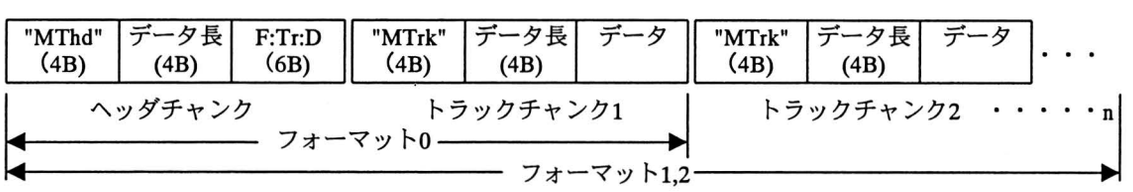

図2.1　SMFの全体構造

3．MIDI シーケンサ

3.1 PIC16F877 を用いた MIDI シーケンサの構成

MIDI シーケンサを PIC16F877 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
- RD0～RD7　ノート番号／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
- RC3　SCL（I²C のクロック）
- RC4　SDA（I²C のデータ）
- RC5　DDS0（DDS0 選択信号）
- RC6　DDS1（DDS1 選択信号）
- RC7　DDS2（DDS2 選択信号）
- RB0　演奏速度低下スイッチ
- RB1　演奏速度増大スイッチ
- RB3　演奏停止スイッチ
- RB4　I²C-EEPROM／Prog メモリ切り替え用
- RB7，RB6　ICSP（インサーキット・シリアル・プログラミング用に使用）
- RA1～RA3　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 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 への出力を主に行っている．

図 3.2 はデルタタイムを処理するサブルーチンのフローチャートである．ノートオンメッセージ毎に可変長数値で表されるデルタタイムを解析し，通常の数値に変換して初期値を設定し，ISR において消費されるデルタタイムが 0 になるまで待つようになっている．

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 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を消費する処理を行っている．

以下各部分の処理の詳細について説明する．

3.3 MIDI データの I²C-EEPROM への格納と読み出し

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．EEPROM には 256K ビット（32K バイト）の I²C-EEPROM である 24LC256 を用いた．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ている．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を 0x20 に設定した．

PIC マイコン 16F877 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るので，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．

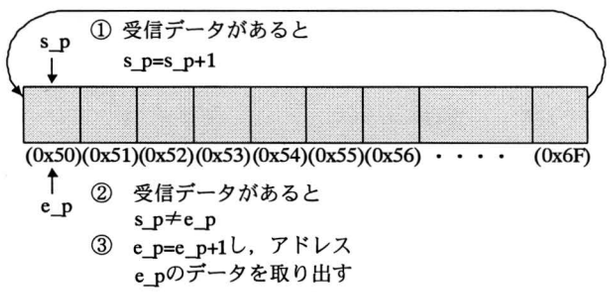

図3.4 受信リングバッファ

3.4 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「デルタタイムの処理」において行っている．その詳細を図 3.2 に示す．なおこのフローチャートでは，各デルタタイムは時間的に重複することなく時間消費が行われることを前提にしている．

具体的にはデルタタイムの長さが 2 バイトであり，可変長数値であることを考慮して，以下のように処理している．

① 読み出したデータ（MIDI_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 MIDI_2nd に格納
③ MIDI_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 DT に格納
④ キャリービット C の値を考慮して MIDI_2nd を処理し，結果を変数 DT+1 に格納
⑤ ①の結果ビット 7 が 0 の場合には DT=0 とし DT+1 に MIDI_1st を格納する．
⑥ flags_delta ビットの処理はフローチャートに示すとおりである．
⑦ 3 音源を制御する変数 DDS0～DDS2 の初期化を行う．

(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 タイマ割り込みの周期は MIDI 規格に合わせて 32μs にしている．

図 3.3 のフローチャートに示したように，dtcnt→DT+1→DT の全変数が 0 になるまでの時間を待つ．変数 dtcnt 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

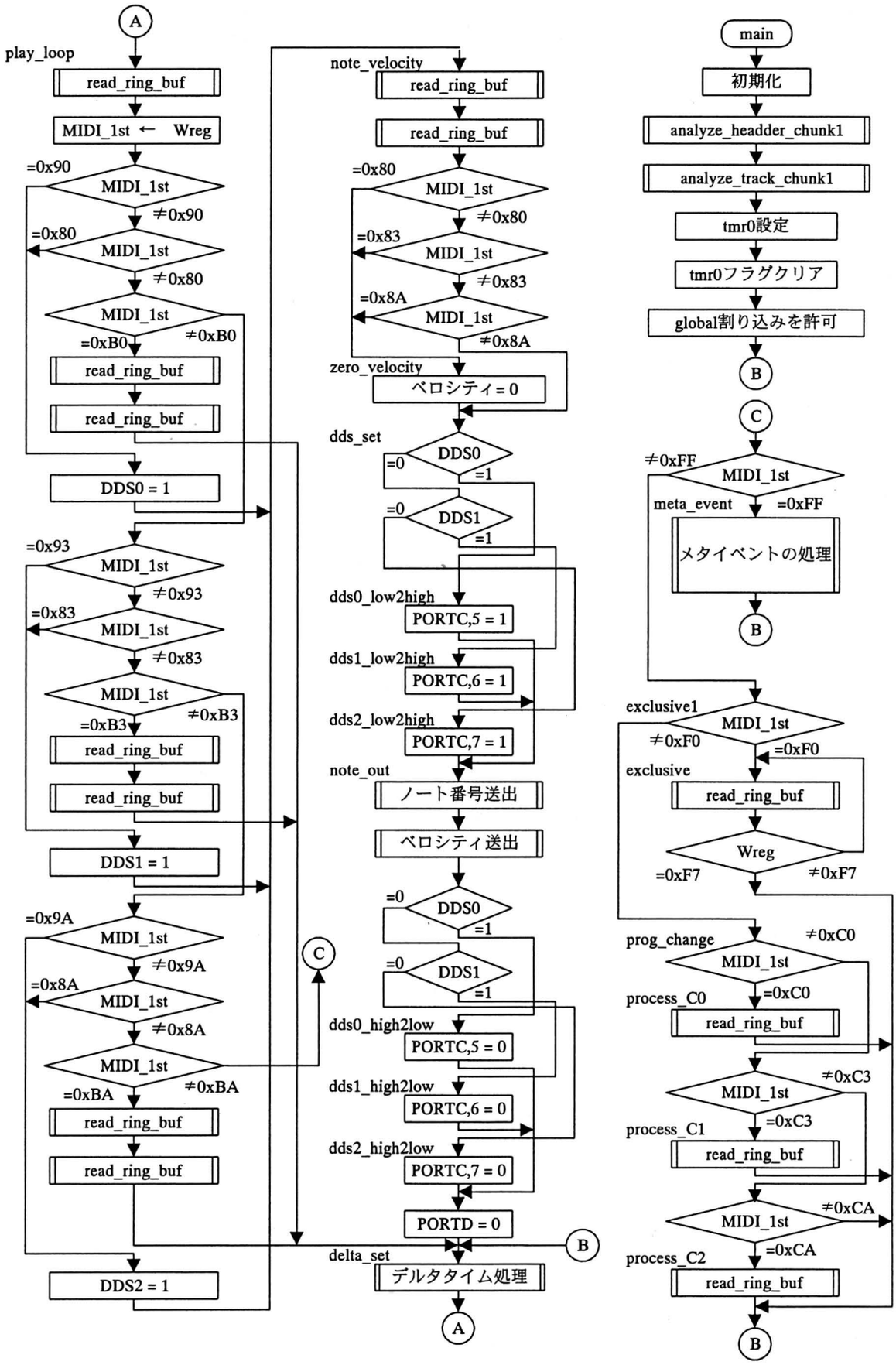

図3.1 メインプログラムのフローチャート

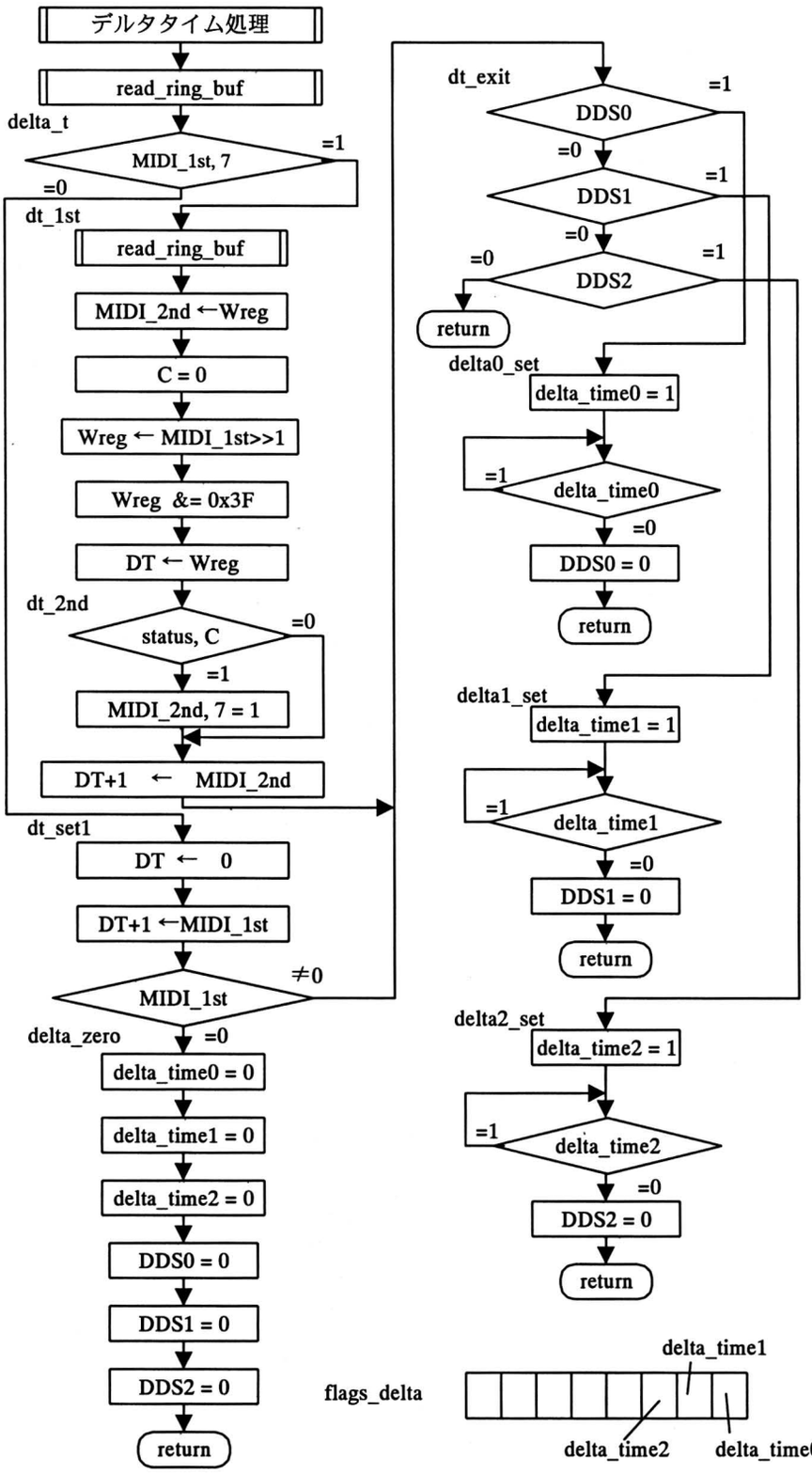

図3.2 デルタタイムの処理フローチャート

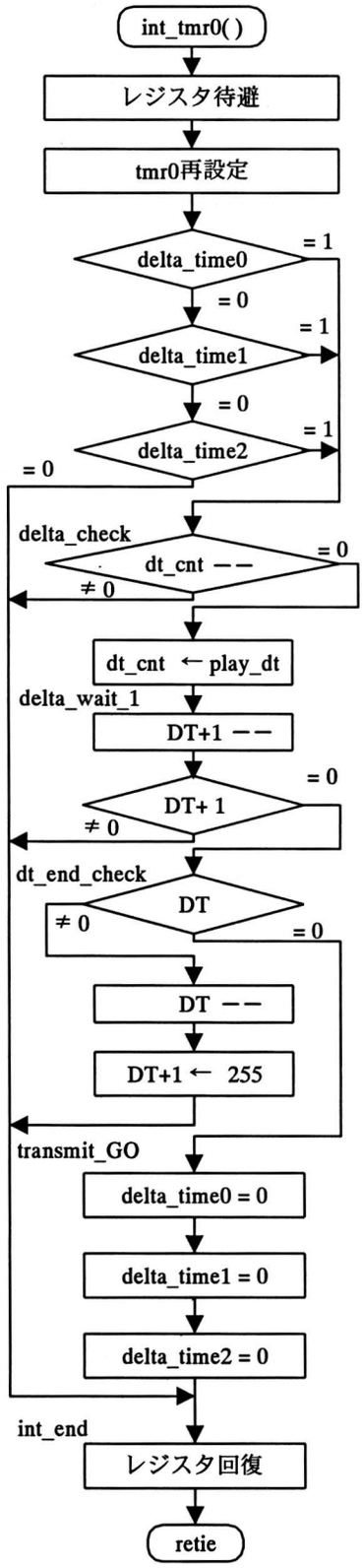

図3.3 割り込みサービスプログラムのフローチャート

から最大40の間で変えられるようにしている.

② デルタタイムに相当する待ち時間は以下のようになる.

32×play_dt×デルタタイム（μs）

(3) flags_delta の処理

図3.2～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デルタタイムに関する流れを制御している.

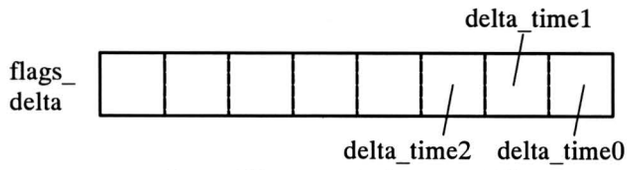

図3.5 変数flags_deltaによるフロー制御

デルタタイムを検出し，ISRにおいて時間消費に入るときに各ビットdelta_time0～delta_time2を1にする．これらのビットが0になると，次の処理に移行する.

3.5 メタイベントの処理

「エンターテイナー」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.

ADDRESS	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00004610	00 00 06 88 F4 00 FF 2F 00

図3.6 トラックマーカーの構成

演奏をエンドレスで行うためにトラックマーカー検出時に，メインプログラムの先頭に戻るようにしている.

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.

3.6 ノート番号およびベロシティの処理

図3.1および図3.2に示したフローチャートにおいて「dds_set～デルタタイムの処理」におけるタイミングチャートを図3.7に示す．簡略化するために音源が2チャネルの場合を例示している.

ノートオンの手順は以下のようになっている.

(1) MIDIデータはMIDIイベント＋デルタタイムの構成になっており，デルタタイムの値がその都度変わるので到着時刻は不規則になる．プログラムではノートオンメッセージを検出すると該当のDDSを選択するための信号（図ではDDS0あるいはDDS1）をHにする.

(2) その後，PICのポートD端子にノートオンメッセージ，ベロシティの順番に8ビット並列に出力する（note_velocity信号）.

(3) さらにDDS側でnote_velocity信号を分離できるようにラッチイネーブル信号LE_noteおよびLE_velocityを出力する．この信号をDDS側においてクロック（0.24μs幅）の立ち上がりでタイミングをとってフリップ／フロップに読み込んでいる．パルス幅は0.4μsである.

(4) DDS選択信号はデルタタイムに相当する時間消費が終了してからLとしている．しかしこの処理はnote_velocitys信号送出直後に行うことも可能である.

(5) 以上の結果として発音が行われる.

ノートオフの処理は，殆どノートオンの処理と同じである．しかし「G線上のアリア」の楽譜のようにベロシティ0によってノートオフを行うものもあるため，互換性を考慮して以下のように処理している.

(6) 8nH（nはチャネル）で始まるノートオフメッセージを検出すると，ベロシティを0にする．DDSでは0のベロシティを検出するとノートオフの処理を行うようにする.

ロジックアナライザを用いて測定した出力波形を図3.8に示す．この波形は図2.7の楽譜データにおいて，先頭のノート番号であるアドレス0xC1～0xC3の部分である．①で示す0x56はノート番号であり，音の高さを表す．②で示す0x40はベロシティであり，音量データを表している．③で示す信号は，DDS選択信号DDS0～2である．④で示す信号は，LE_NOTEおよびLE_VELOCITYである．いずれの信号も図3.7に示した通りのタイミングチャートになっていることが確認できる.

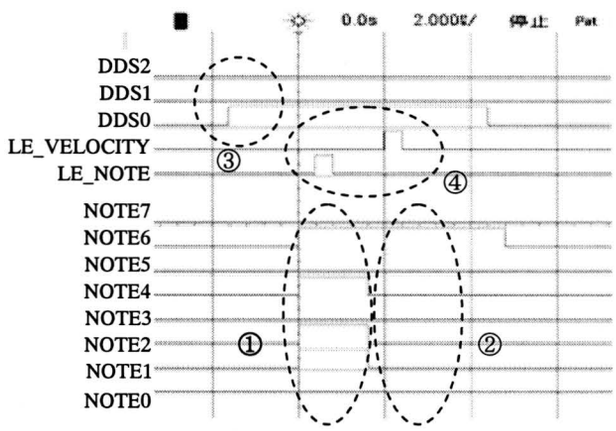

図3.8 タイミングチャートの実測結果

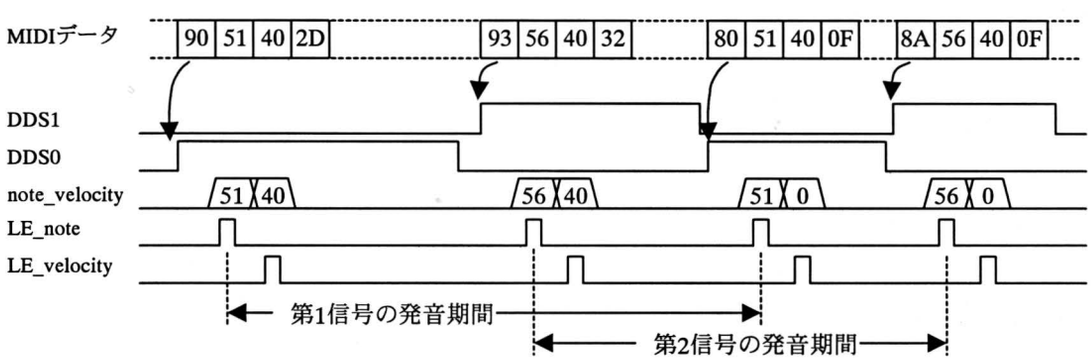

図3.7 ノート番号およびベロシティに関するタイミングチャート

4．G線上のアリアの演奏に用いた単音源DDSの回路構成

4.1 単音源DDSの回路構成

図4.1にG線上のアリアの演奏に使用した単音源DDSの構成を示す．DA変換器およびLPFは外付けの回路である．それ以外はアルテラ社のMax II Micro Board (EPM2210F324 CPLDデバイス）に実装した回路である．

図示されていないPICマイコンからノート番号／ベロシティ（note_velocity）が8ビット並列データとしてフリップ／フロップ（実際には各々8個ずつあるが，略記している）に入力され，これを選択するLE_NOTEおよびLE_VELOCITY信号によってラッチされてノート番号（7ビット）およびベロシティ（7ビット）に分離され保持される．クロック信号とのタイミングは取っていない．

ノート番号は後述する式(1)を用いて14ビットの周波数データに変換している．ベロシティは音の強弱を表す量であり，DDS出力との積を計算し音量を調整している．ベロシティ=0を検出したときノートオフの処理を行うようにしている．

4.2 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．

$$周波数=440\times 2^{\frac{NOTE\_NO-69}{12}}\ (Hz) \quad (1)$$

ノート番号は0～127，周波数に直すと0Hz～12543Hzである．

4.3 DDSの原理およびクロック周波数の選択

DDSは1波形分を記憶している波形メモリのデータを読み出すとき，発生しようとする周波数に応じて読み出し周期を変え，元の波形と相似でかつ周波数の異なる信号を生成する回路である．波形メモリの読み出し周期は，周波数データを累積加算することにより決定する．

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することを考えると，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は2^nHzとなる．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．1波形を256分割するものとすればさらに8ビットが必要であり，これよ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．

以上よりビット数を以下のように設定した．

- 周波数データビット数 14ビット
- DDS出力（波形メモリ）の分割数 8ビット
- アダーのビット数 22ビット

4.4 アッテネータによるベロシティの処理

MIDIメッセージの3バイト目であるベロシティは音の強弱を表す量であり，図2.5に示すように7ビットのデータ（0～127の値）である．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DDS出力（8ビット）との積を計算し，得られた結果（14ビット）における上位8ビットを取り出すことによってディジタル・アッテネータを実現している．

アッテネータ入力であるDDS出力は，8ビットのオフセットバイナリ形式で出力される．このためDDS出力の最上位ビットを反転して2の補数表示に変換し，Q7フォーマットである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た．なお乗算あるいは加算は，ともに演算子＊あるいは＋を使用して記述した．

4.5 その他の回路

外付けの回路としてR-2Rラダー型DA変換回路，2次アクティブローパスフィルタ（入力にボルテージフォロワを前置），コルピッツ型クロック発振回路，負電源生成回路などを使用している．

5．3音源DDSの設計

5.1 3音源DDSの回路構成

4で述べた単音源DDSを3個使用し，その出力を加算して3音源DDSを構成した．図5.1にブロック構成を示す．

点線の四角形で囲った部分が単音源DDSであるが，5.2および5.3に示す変更を加えている．

5.2 入力インタフェースの変更

入力インタフェース回路のブロック構成を図5.2に示す．

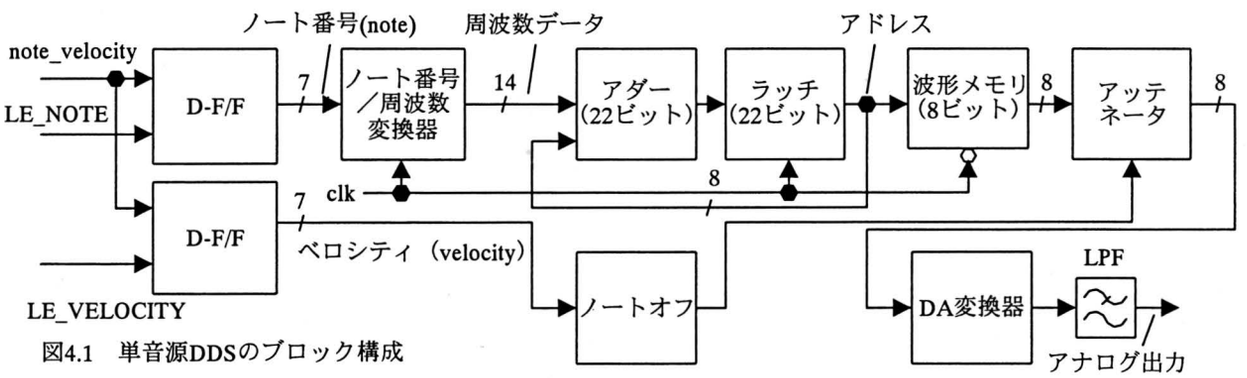

図4.1 単音源DDSのブロック構成

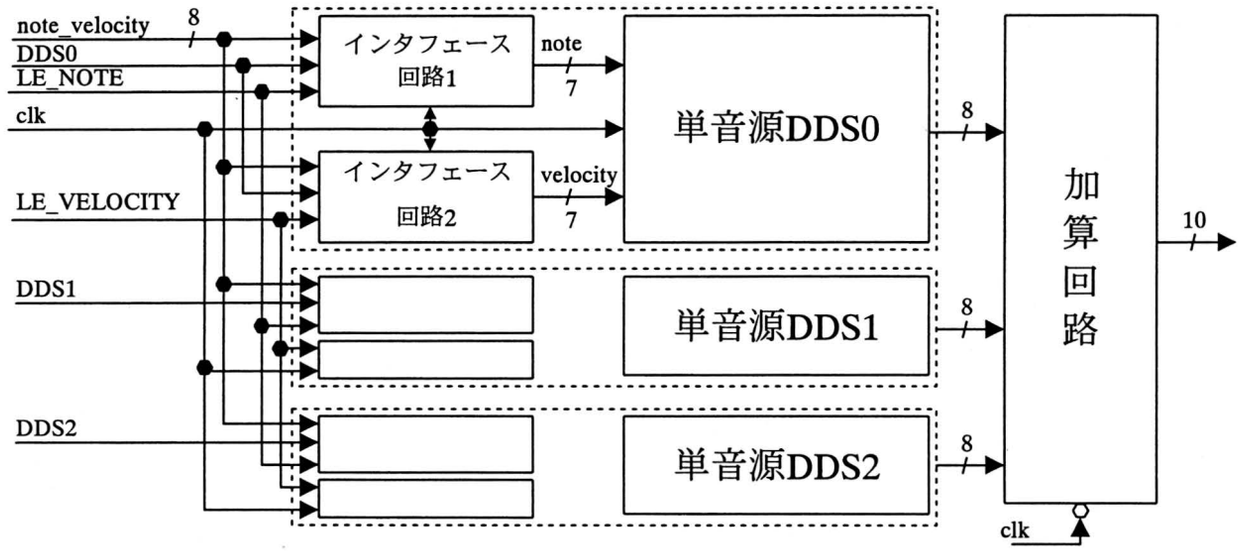

図5.1 3音源DDSのブロック構成

1個のDDSに対する回路である．図4.1からの変更点は

・ ノート番号／ベロシティ（note velocity）をクロックとタイミングをとってフリップ／フロップに入力するようにした

・ DDS選択信号を追加した

ことである．

当初は図 4.1 に示すようにノート番号／ベロシティをLE_NOTEおよびLE_VELOCITY信号の立ち上がりにおいてラッチし，フリップフロップに入力していた．クロックとのタイミングはとっていなかった．しかし開発段階においてエンターテイナーの演奏を聴取してみたところ異音が発生することがあった．このためDDS選択信号とLE_NOTEあるいはLE_VELOCITY信号とのANDをとり，この信号をイネーブル信号としてクロックの立ち上がりにおいてタイミングを取り，フリップフロップに入力するように変更した．これにより異音が減少する効果が得られた．

5.3 入力インタフェース及びマニュアル入力用スイッチ回路

DDS単体の試験を行うためにディップスイッチ入力によってノート番号／ベロシティを設定できるようにした．2:1セレクタにおいて，MIDI出力（フリップ／フロップ出力）とディップスイッチからの入力信号であるnote_manあるいはvelocity_manを切り替えて，各々7ビットの出力であるnoteあるいはvelocityを出力している．

MIDI出力あるいはマニュアル出力の切り替えは，未使用であるnote_man信号のビット7にスイッチを接続して行っている．ノート番号およびベロシティは，同時にMIDIかあるいはマニュアルに切り替わるようになる．

・ note_man(7)=1のとき・・・・MIDIからのデータを選択

・ note_man(7)=0のとき・・・・マニュアルデータを選択

マニュアルデータ選択時には，以下の信号によりDDS出力を切り替えている．

・ dds_set0_man=1・・・・DDS0の設定変更

・ dds_set1_man=1・・・・DDS1の設定変更

・ dds_set2_man=1・・・・DDS2の設定変更

ノート番号およびベロシティは専用のディップスイッチにより独立に設定できる．上記の各信号が0の場合にはそのDDSの出力を停止する．

5.4 出力インタフェースの変更とDDS出力の加算回路

4.1で示した単音源DDSにおけるアッテネータは，内部の計算は2の補数表示に基づいて行っているが，出力時にはオフセットバイナリ形式に変換している．このためオフセットバイナリ形式への変換をコメントアウトして2の補数表示に戻し，8ビットのDDS出力を10ビットに符号拡張してから3個加算し，その後改めてオフセットバイナリ形式に変換するようにした．3個のDDS出力を加算するために加算回路を使用したが，VHDLでの記述は単に+演算子を使用しているだけである．

図5.3にシミュレーション結果を示す．中程にある*a*，*b*，*c*がDDS出力である．DDSが動作した直後の波形であり，2の補数表示の0を表示している．また*x*が加算回路の出力であり，オフセットバイナリ形式に正しく変換されて0x200になっている．

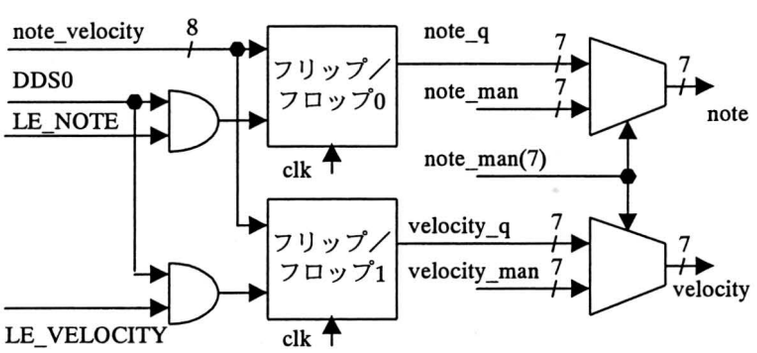

図5.2 入力インタフェース回路のブロック構成

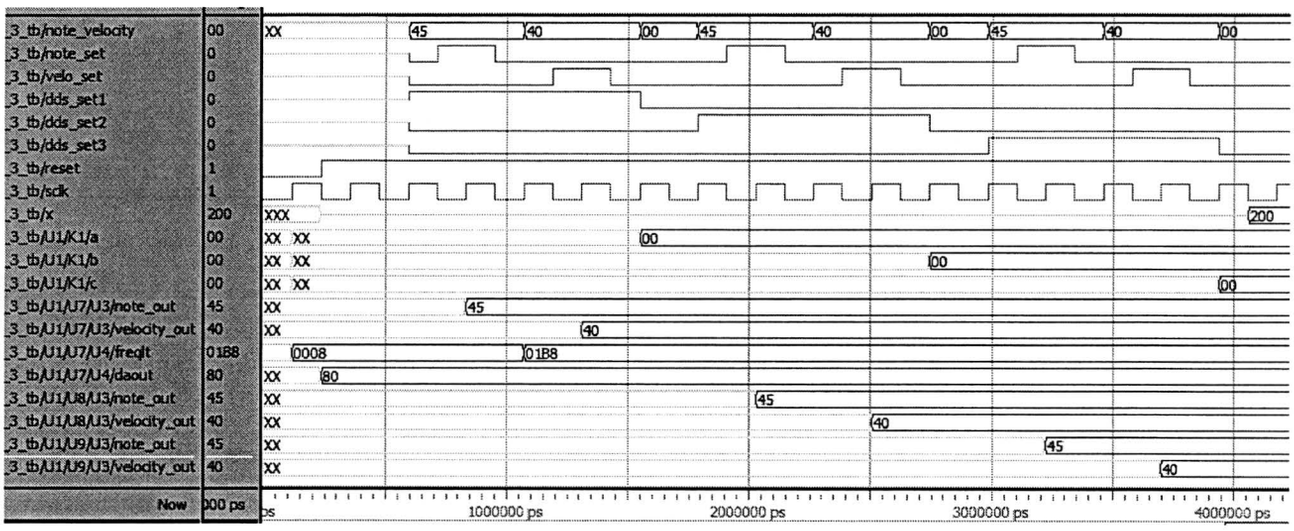

図 5.3 ModelSim によるシミュレーション結果

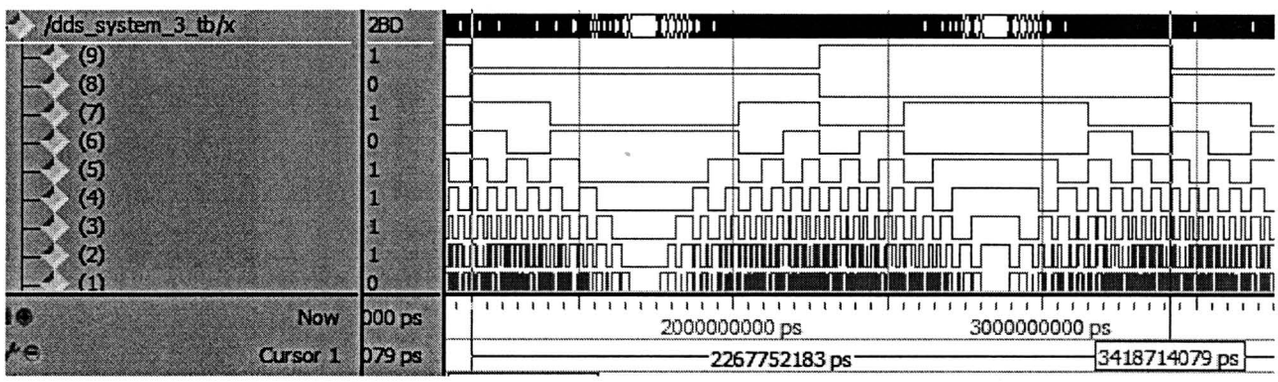

図 5.4 ModelSim によるシミュレーション結果（その 2）

5.5 シミュレーション結果とクロックエッジの確認

各所でクロックとのタイミングをとっており，エッジの確認を行っておく．

① MIDI シーケンサ出力では，図 3.8 に示したようにパルス幅 0.4 μs の LE_NOTE, LE_VELOCITY を出力している．この信号をイネーブル信号として，図 5.2 に示すフリップ／フロップの立ち上がりにおいてノート番号／ベロシティを読み込み出力している（図 5.3 における note_out, velocity_out）．

② ノート番号→周波数変換器では，ノート番号を入力し，クロックの立ち上がりエッジにおいて周波数データを出力している（図 5.3 における freqlt, 0x1B8 = 440Hz になっている）．

③ アダー（図 4.1）は組み合わせ論理回路である．

④ ラッチはクロックの立ち上がりエッジにおいてアダー出力を読み込み，出力している．

⑤ したがって波形メモリの読み出しは，クロックの立ち下がりエッジにおいて行っている（図 5.3 の daout が出力）．

⑥ アッテネータはクロックの立ち上がりでタイミングを取っている（図 5.3 における a, b および c がアッテネータ出力）．

⑦ したがって加算回路はクロックの立ち下がりで読み込んでいる（図 5.3 における x が加算回路出力）．

図 5.3 からデータとクロックのエッジに関しては問題のないことが確認できた．

図 5.4 は，図 5.3 に続く時間の加算回路出力 x のシミュレーション結果であり時間を拡大して示している．ノート番号は 0x45 であり，これは周波数に直すと前述したように 440Hz になる．x(9)の周期から発生周波数は 440.965Hz であり，ほぼ正しい値を出力できている．

6．CycloneⅣ FPGA を使用した 3 音源 DDS の試作結果

アルテラ社の VHDL シミュレータ Quartus Ⅱ を用いて設計を行い，ModelSim を用いてシミュレーションを行った．回路の実現は CycloneⅣ FPGA を搭載したアルテラ社の FPGA ボード（EP4CE22F17C6）を使用した．

6.1 DDS の出力波形

3 個の DDS から単独で信号を発生させた場合の出力波形を図 6.1(a)～(c)に示す．横軸は 2ms/div であり，縦軸は 500mV/div である．また(a)は C4（ド，262Hz），(b)は E4（ミ，330Hz），(c)は G4（ソ，390Hz）である．また図 6.1(d)に合成波形を示す．

6.2 ノートオフに対する処理

ノートオフメッセージとして 8XXXXXH のようなメッセー

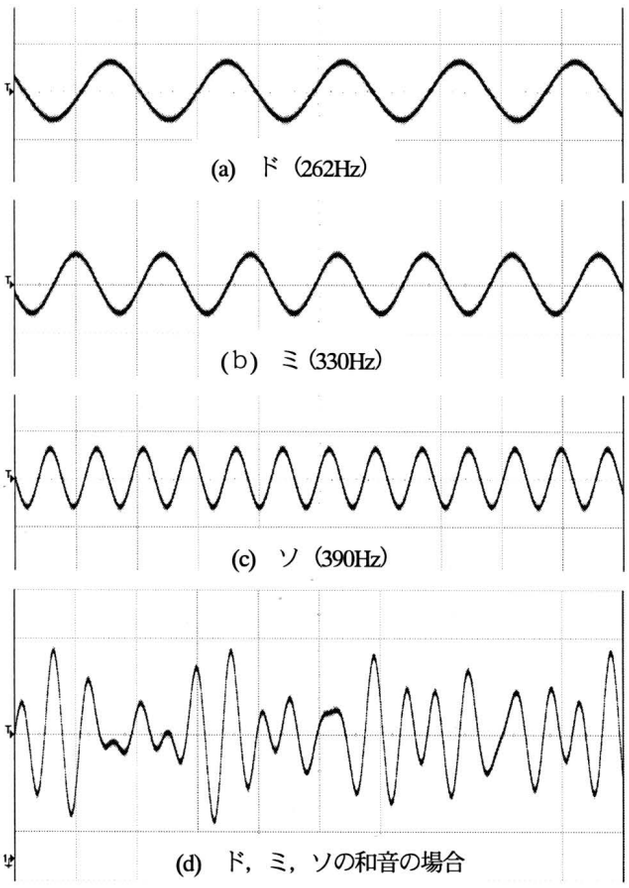

図 6.1 DDS の出力波形

ジと，9XXX00H の 2 種類を想定している．このため PIC において 80H から始まるノートオフメッセージを受信すると，図 3.1 の真ん中，上側の部分に示したようにベロシティを 0 に設定し直して DDS に出力するようにしている．これにより後者のメッセージとの互換性を保っている．

一昨年度に試作した G 線上のアリアを演奏する自動演奏システムでは，ノートオフ時にスピーカから耳障りな「ボコ」と言うような音が出ており非常に気になった．そこで今回はノートオフ時に完全に音を止めるのではなく，ノートオフ検出時にはベロシティを 25 に低下させるようなプロセスを組み込んでみた．これは DDS の出力電圧振幅を最大振幅の約 0.2 倍に相当する値に低下させる値である．比較的良好な結果が得られたものと考えているが，今後更に検討していく必要がある．なお周波数は変えていない．

6.3 聴取結果

最終的なプログラムが完成した段階で研究室のメンバーに協力して貰い「エンターテイナー」の聴取試験を行った．被験者は 8 名であり，以下の 2 点を評価項目とした．

① 1 回の演奏中に異音が何回聞こえたか．

② スピーカに接近したときに，気になる雑音が聞こえるか．

またノートオフ検出時に低下させるベロシティの値を 25，8 および 0 の 3 水準に変えて試験を行ってみた．その結果上記ベロシティの値を小さくするほど，音が切り変わるときに発生する異音がより気になることが分かった．異音の聞こえた回数は，どの被験者でも概ね 10 回／曲であった．またスピーカに接近して音を聞いた場合には，始終雑音が発生している状況があり，改善の余地が十分にあることが分かった．

7．むすび

高校生に電気電子に関心を持って貰う手段としての“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．演奏する MIDI の曲については 3 音源からなる「エンターテイナー」を選んだ．この曲を演奏するために PIC マイコン 16F877 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 VHDL により設計し，FPGA を用いて 3 音源 DDS を試作しその結果を示した．

これらの要素を組み合わせて MIDI 自動演奏システムを構成し，I^2C-EEPROM に書き込んだ「エンターテイナー」を読み出して自動演奏ができることを確認した．しかしながら音質の点でまだまだ改善の余地が十分にあることが分かった．

今後音質の改善についての検討を行っていく．また今回は MIDI シーケンサを 3 チャネルに特化して作成したが，フローの見直しにより 16 チャネルまで増大させることが可能であり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) 袴田吉朗，松永康寛，栗田貴史，“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”，静岡理工科大学紀要，Vol.20，pp.11-20，(2012)
4) 中島安貴彦，“MIDI バイブル I MIDI1.0 規格基礎編”
5) “詳説 MIDI 規格”，http://www.pluto.dti.ne.jp/~daiki/midi/midi.html
6) “SMF の基礎知識”，http://www.hikari-ongaku.com/study/smf.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)

［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 2014.2.16 にデバグを完了したプロジェクト midirec20140216 に基づいている．また DDS に関する VHDL プログラムは 2014.2.15 作成の dds_system_33 プロジェクトに基づいている．