サクソフォンの自動演奏ロボットの研究：演奏制御システムのコンパクト化と演奏表現の高度化

MORI, Takuya / 森, 琢也

(開始ページ / Start Page)
1
(終了ページ / End Page)
186
(発行年 / Year)
2016-03-24
(学位授与年月日 / Date of Granted)
2016-03-24
(学位名 / Degree Name)
修士(工学)
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)

2015 年度修士論文

サクソフォン自動演奏ロボットの研究

―演奏制御システムのコンパクト化と演奏表現の高度化―

STUDY OF THE AUTOMATIC PLAYING ROBOT OF SAXOPHONE
- COMPACTIFICATION OF THE PLAYING CONTROL SYSTEM AND SOPHISTICATION OF MUSICAL EXPRESSIONS -

指導教授　　高島　俊

法政大学大学院理工学研究科

機械工学専攻　修士課程

14R1135　森（もり）　琢也（たくや）

Abstract

There are several amusement robots in our laboratory and the automatic music playing robot of saxophone is the most long life one of them. In this study, the control problems of the robot are discussed. The robot controls reed pressing force by an artificial mouth system, controls blowing air pressure by a special pressure control valve and operates fingering system. Under these controls, it can perform any sheet music with original sheet music data or MIDI data just like a human player. Wind instruments usually require hard practice to control embouchure, breath, fingering and musical expressions. Especially the most difficult task for playing the saxophone is to keep proper embouchure. It is because human mouth is a very complex muscle and it can move very skillfully. It will be very useful to acquire knowledge, knowhow, engineering techniques and theoretical hints as the results of creating music performance robots aiming good performance like a human professional player.

The control system of the existing saxophone robot consists of a desktop PC and PCI bass interface boards, a power source, and other electrical devices. The control commands are sent to each device from PC through interface boards. Because the whole system is pretty large, it has a problem of portability and mobility considering the situation of attending the concert and playing with human players. To solve the problem, compactification of the system is considered. The compactification is attained by using a note-PC as a control device and using USB interface of D/A and DIO. Furthermore, the power supply and the actuator driving circuits are also compactified.

The saxophone robot can play any musical sheet of slow or fast, short or long, easy or difficult. The technical ability is enough to play music but the artistic ability is beyond the reach of human professional player. It cannot use more sophisticated musical expression techniques of Jazz players such as some kinds of attacks, glissando, sub-tone, over-tone , high level vibration control and other Jazz-like rhythm and tonguing although envelope control, vibration control can be realized by including them in MIDI data. In this paper, improvements for some of these more sophisticated performance abilities are attained.

目次

第１章　緒論

当研究室には様々なロボットがあり，その中には様々な楽器を自動演奏するロボットの研究がある．その一つにサクソフォン自動演奏ロボットがある．本研究はサクソフォン自動演奏ロボットの制御についての研究である．

サクソフォン自動演奏ロボットは，人間が演奏するのと同様に人工唇と人工口顎部を用いてリードを押さえる力を制御し，呼気の圧力を制御し，運指機構を制御することで，楽譜データに基づいて自動演奏を行うことができるロボットである．楽譜データはオリジナルなデータ形式のものか，または MIDI ファイルとしてのデータを用いることができる．

サクソフォンを含む管楽器の演奏の動作は，指の操作，息の吹き込み，口の動き，さらに楽曲の解釈から表現方法に至るまで，多くの複雑な制御を必要とする．これを機械システムに置き換えることは，機構学的にも人間工学的にも得るものは大きいと考えられる．

これまでの演奏制御システムでは，各制御システムのコントローラとしてデスクトップ PC を用いており，各種インターフェースも PCI バスを用いたオプションボードを装備し，各装置に制御指令を送出することで自動演奏を行っている．そのため，今後，人間の演奏者と合奏をしようとした場合，演奏会場への移動や携帯性に問題がある．したがって，ここでは，USB インターフェースを用いて，ノート PC をコントローラとすることを考える．さらに電源や駆動回路などのデバイスもコンパクト化し，より移動や携帯が可能となるような改造を施すこととする．

現在，木管楽器であるテナーサクソフォンの自動演奏ロボットは MIDI ファイルを使った既存の制御プログラムでは制御可能となっている．しかし，既存のサクソフォン自動演奏ロボットシステムの演奏技能は，まだまだ人間には及ばない．MIDI のデータにエンベロープ制御やバイブレーションなどは組み込めるが，特に JAZZ の演奏に用いられる数種類のアタックやグリッサンド，サブトーン，オーバートーンなどの特殊な技能や，JAZZ 独特のリズムとタンギングのとり方などには対応できていない．さらに，ビッグバンドなどでの人間との合奏では人間並みの演奏表現が求められるため，本研究では，演奏表現を豊かにするための種々の改良を考え，より汎用的な応用を可能にすることも目的とする．

第２章　MIDI

2.1　MIDI とは

MIDI(Musical Instrument Digital Interface)[4)5)]とは電子楽器やコンピュータなどと接続し演奏や音色などの情報伝達に用いる通信方法の世界共通規格のものである．

MIDI に対応している電子機器と MIDI ケーブルで接続し演奏情報を伝える．

MIDI データは演奏の情報を指し，音そのもののことではない．つまり MIDI データだけでは音は鳴らないので演奏を行う音源がないと再生ができない．また，MIDI データは音声データと比べ，データサイズが小さいのでインターネットを通してデータの受け渡しや，リモート演奏（遠隔で演奏ができること）を行う上でメリットがあると考えられる．さらに，無料の MIDI 作成アプリケーションも多く存在していることから作成や入手することが容易であることもメリットである．

MIDI 端子に使用されているコネクターは 5 ピンの DIN コネクター(オス)であり，接続する際の一般的なものである．DIN(Deutsche Industrie Normen)とは，ドイツの工業品標準規格のことであり, 同じコネクタが MIDI 以外にも使用されている．

コネクタには MIDI 信号を入力する(受信する)MIDI IN 端子, MIDI 信号を出力する(送信する)MIDI OUT 端子がある．さらにオプションとして MIDI IN に入力された信号をそのまま出力する MIDI THRU 端子がある．この MIDI THRU 端子は MIDI 楽器をカスケード接続(複数のハブ（HUB）をつなぎ合わせて、より多くの端末を接続できるようにすること)することにより 2 台以上の MIDI 楽器を接続できるように用意されたものである．しかし，いくらでも接続ができるわけではなく, 接続する楽器が多いほど MIDI 信号の遅延が起きるので限度はある．

2.2 MIDI メッセージ

情報を伝達することを MIDI メッセージ [4)5)]と言い，演奏動作の制御信号（鍵盤を弾く，ペダル操作など）を伝えるものである．このメッセージを伝達することで音色選択，編集にいたる操作を遠隔操作できるようになっている．MIDI は直列のインターフェイスであるので，見かけ上複数のメッセージを同時に送る場合も 1 ビットずつ順番に送信される．MIDI メッセージは 8 ビット(ビット 0～7)の組み合わせ(1 バイト)を最小単位として構成されている．ただし，複数のバイトを途切れなく流すと，どこが 8 ビットの先頭か末尾かわからないことから先頭にスタート・ビット，末尾にストップ・ビットと呼ばれるビットを 1 ビットずつ追加し，区別する(この時スタート・ビットは 0，ストップ・ビットは 1 に相当)．つまり，コンピュータの世界では 8 ビットを 1 バイトとするが，MIDI ではスタート・ビットとストップ・ビットを含む 10 ビットで 1 バイトとするのである．

MIDI メッセージの種類は大きく 2 つに分類できる．

①MIDI チャンネルごとに独立した演奏情報を伝えるために用いられるチャンネル・メッセージ

②MIDI に接続されたシステム全体を制御するために用いられるシステム・メッセージ

2.2.1 ステータス・バイト，データ・バイト

MIDI メッセージは，メッセージの種類を示すステータス・バイト×1 バイトとその値，データを示すデータ・バイト×1～数バイトで構成されている [1)]．ステータス・バイトの上位 1 ビットが 1 で，データ・バイトの上位 1 ビットが 0 である．これにより，ステータス・バイトかデータ・バイトかを区別する．メッセージを 0 と 1 だけの 2 進数で表す方法では分かりにくいことから 0～F の 16 進数で表される．それは，2 進数は 4 つを 1 組として 1 桁の 16 進数で書き表すことができるからである．よって，MIDI メッセージの内容を表すときは MIDI の 1 バイト(スタート, ストップ・ビットを除く 8 ビット)のうち, 最初の 4 ビット(上位 4 ビット)と次の 4 ビット(下位 4 ビット)に分け，2 桁の 16 進数を用いて表記できる．

2.2.2 ノート・オン・メッセージ，ノート・オフ・メッセージ

ノート・オン・メッセージとノート・オフ・メッセージは MIDI メッセージの演奏情報の中では基本となるもので，これらの情報は楽器のキーを押す(ノート・オン)，キーを離す(ノート・オフ)という動作に応じており，MIDI ではそれぞれ専用のステータスで割り当てられている [1)2)3)5)].

ステータス・バイト		データ・バイト1	データ・バイト2
メッセージ	チャンネル	ノート・ナンバー	ベロシティ
1XXX	nnnn	0kkk kkkk	0vvv vvvv

図 1. メッセージ概略図

ステータス・バイト										データ・バイト1										データ・バイト2									
0	1	0	0	1	n	n	n	n	1	0	0	k	k	k	k	k	k	k	1	0	0	v	v	v	v	v	v	v	1
↑スタート・ビット					ストップ・ビット↑					↑スタート・ビット					ストップ・ビット↑					↑スタート・ビット					ストップ・ビット↑			

図 2. ノート・オン・メッセージ表記

ステータス・バイト										データ・バイト1										データ・バイト2									
0	1	0	0	0	n	n	n	n	1	0	0	k	k	k	k	k	k	k	1	0	0	v	v	v	v	v	v	v	1
↑スタート・ビット					ストップ・ビット↑					↑スタート・ビット					ストップ・ビット↑					↑スタート・ビット					ストップ・ビット↑			

図 3. ノート・オフ・メッセージ表記

以上よりノート・オン，オフ・メッセージを 16 進数で表すと
ノート・オン・メッセージ：9n　kk　vv(H)
ノート・オフ・メッセージ：8n　kk　vv(H)
となる.

ステータス・バイトは上位 4 ビットがそれぞれノート・オン・メッセージを表す 9(H)とノート・オフ・メッセージを表す 8(H)で表記され，下位 4 ビットの n は 1～16 の MIDI チャンネルを表す.

データ・バイトは最初の kk はノート・ナンバー(鍵盤の音程のことで範囲は 0～127)，後の vv は，ベロシティ(鍵盤を押す，離す時の速さのことで範囲は 0～127)にそれぞれ相当する.

なお, 16 進数表記の後ろにある(H)の記号は Hexadecimal(=16 進数)の略であり，直前の表記が 16 進数であることを示す(00H～07H のように数値に加える場合もある).

2.2.3 MIDI チャンネル

MIDI にはチャンネル[4)5)]という概念が導入されている. 複数の楽器とカスケード接続し演奏する際に各音源に別々に演奏情報を送るためには，演奏情報の識別が必要である．演奏情報に識別子を持たせると MIDI システム上の各楽器は，識別子によりメッセージを選択して取り込める．そこで，演奏情報の識別をするために MIDI チャンネルが導入されている．チャンネルは 1～16 まであり，それぞれで演奏情報，音色情報をテレビ電波のように特定のチャンネルで送受信ができる．これにより，各々の楽器をそれぞれチャンネルに振り分けることで演奏情報と音色が混同されずにやりとりができる.

2.3 SMF

SMF(Standard MIDI-File Format)[1)2)5)]は MIDI 音楽編集ソフト(シーケンサー)で演奏情報を保存する場合は独自フォーマットで記録するのが普通であるが，それだと異なるシーケンサー同士で，ファイル単位でシーケンス・データ(演奏情報，テンポ情報，拍子情報など)のやり取りができないことから MIDI 演奏データの標準ファイルフォーマットとして SMF がある. SMF に対応しているシーケンサー同士であれば，CD-ROM などのメディアや通信を通し，シーケンス・データのやり取りが可能である．なお，現在はほとんどのシーケンサーが対応している.

SMF は形式の違いから 3 種類のフォーマット(0～2)が定義されており, どのフォーマットでも最大 16 チャンネルの利用が可能である．次にフォーマットの種類について記す.

表 1. SMF の種類

フォーマットの種類	説明
フォーマット0	単一トラックにすべてのチャンネル・メッセージを含むフォーマットである. SMFに対応するシーケンサーのすべてがこの形式に対応している.
フォーマット1	MIDIチャンネルごとにトラックを分け，複数のトラックを利用できるフォーマットである. トラック単位での編集が容易である.
フォーマット2	複数のトラックに加え，複数のソング(シーケンス)を含むことができるフォーマットである. ただ現在対応しているものがないので，シーケンス・データの受け渡しには使われない

SMF の構造はチャンクという一連のデータのかたまりから構成されており，SMF 全体の情報(SMF のフォーマット，トラック数，タイミングの分解能など)を含むヘッダー・チャンクと，実際の演奏情報を含むトラック・チャンクの 2 種類がある．フォーマット 0 の SMF はヘッダー・チャンク×1 とトラック・チャンク×1 から構成され，トラック・チャンクに MIDI チャンネル 1～16 の情報を格納する．フォーマット 1 の SMF はヘッダー・チャンク×1 と複数のトラック・チャンクから構成されており，MIDI チャンネルごとにトラック・チャンクを分けて格納する．以下に図として記す.

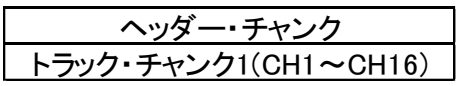


図 4. フォーマット 0 の構造

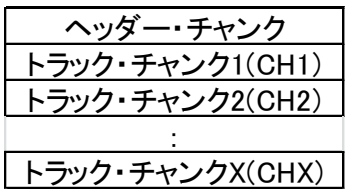


図 5. フォーマット 1 の構造

トラック・チャンクには
①ノート・データなどの MIDI 演奏情報に当たる MIDI イベント
②調/拍子，テンポ，歌詞，シーケンス名/トラック名などの演奏に影響しないメタ・イベント
などがある.

ヘッダー・チャンクは SMF の先頭 14 バイトのことで以下の情報が格納されている.

表 2. ヘッダー・チャンク構造

データ内容	バイト数	データ値
チャンクタイプ	4	MThd
データ長	4	6
フォーマット	2	0,1,2
トラック数	2	0～n
時間単位	2	Time

ヘッダー・チャンクのチャンクタイプは MThd で表し，データ長は，フォーマット，トラック数，時間単位の合計バイトで常に 6 である．フォーマットは MIDI フォーマットの種類(0～2)を表し，フォーマット 0 の場合はトラック数が 1 つだが，フォーマット 1 の場合は複数のトラック数が存在する．トラック数はそのトラックの数が格納される．時間単位はイベントを MIDI デバイスに送るタイミングを算出する場合に用いられる．

トラック・チャンクは以下の情報が格納されている．

表 3. トラック・チャンク

データ内容	バイト数	データ値
チャンクタイプ	4	MTrk
トラックのサイズ	4	n
一連のイベント	n	

トラック・チャンクのチャンクタイプは MTrk で表し，トラックのサイズは一連のイベントのバイト数である．プログラムにおいては，トラックからイベントを取得し，それを MIDI デバイスに送信することになる．

第３章　サクソフォン自動演奏ロボット

3.1　サクソフォン

サクソフォン[1)2)3)]は金属製の木管楽器の一種であり,種類はソプラノ,アルト,テナーなどの種類がある．サクソフォンの形はソプラノなどではまっすぐに作るのが普通であり，その他は円錐管の両端を折り曲げた形となっている．細い方の湾曲部の先にマウスピース，リードを取り付ける．管は下に向かうにつれ太くなり最後に湾曲し上方を向く，末端はやや外に反っていてベル型の開口部を成している．このような形成のため，リードにより起こる音波は内壁の一方から他方へ渡り合い，お互いに交錯しながら出口へと抜けていく．これが木管にも金管にも属さない特有の音色を出す.

管体には 22 あるいは 23 のホールと 2 つのオクターヴホールがあり，いずれも指でふさぐには大きいので，22 または 23 のキーで開閉を行う．原則として 1 つのホールの開閉で音程が半音上下するように作られている．オクターヴホールは倍音の吹鳴を容易にするために設けられているので，ホールとオクターヴホールの開閉により，2 オクターヴ半の吹鳴が可能である.

図 6. テナーサクソフォン

3.2　サクソフォン自動演奏ロボット

当研究室のサクソフォン自動演奏ロボット [1)2)3)]は送気システムを用いてサクソフォンを吹鳴させる．そして人間の口を人工口顎部装置，呼吸器官を送気システム，指を運指機構に置き換えることにより構成され，これらの制御システムと MIDI データを一台のコンピュータにより制御する．よって，人間と同様の演奏を行うことができるようにするための構造となっている．なお，サクソフォンの種類はテナーサクソフォンである．

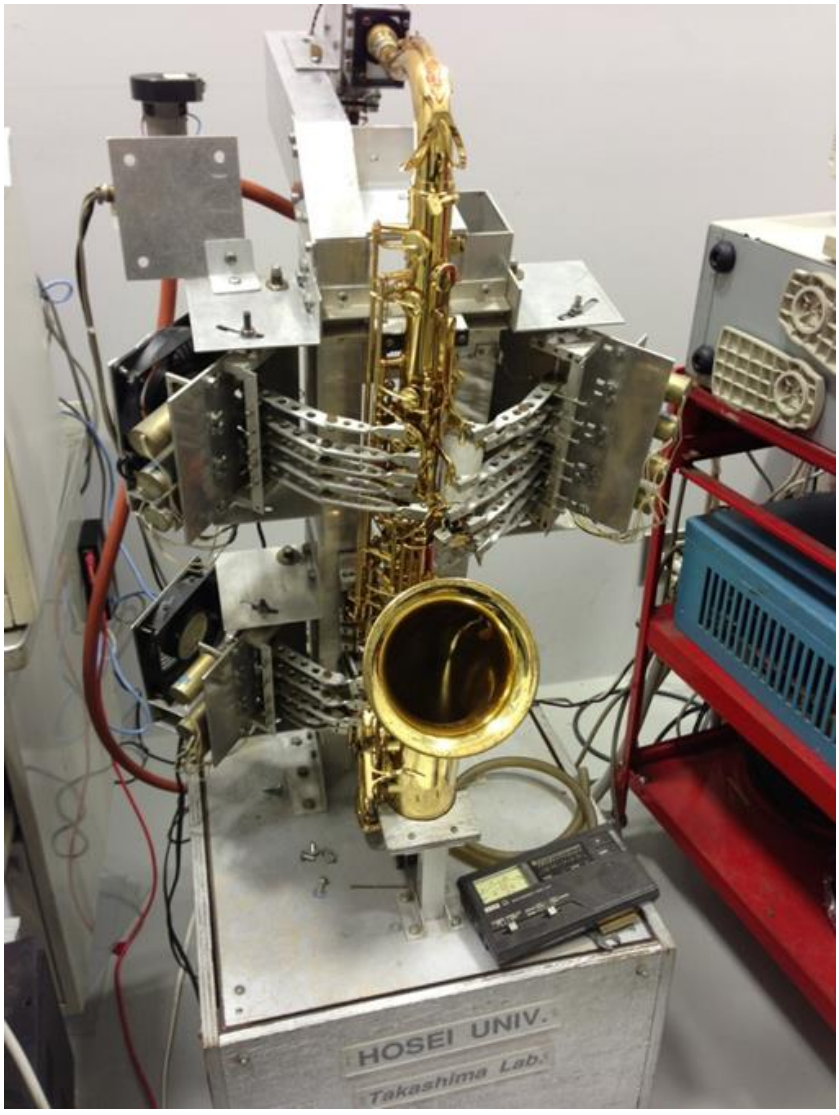


図 7. サクソフォン自動演奏ロボット

3.3　SMF を用いた自動演奏システム

このシステムの主な目的としては，演奏表現の幅を広げるとともに，独奏だけではなく合奏も可能な汎用インターフェイスおよび演奏制御を行うために作られたものである [1)2)3)].

このシステムは SMF ファイル，つまり MIDI を用いることで演奏や音色などの情報伝達に用いる通信方法の世界共通規格のものであることから汎用性が高いと考えられる．また，SMF は入力や編集等が容易に行えることや MIDI メッセージの種類なども多いことから，演奏表現が細かく設定できるというのもこのシステムの特徴であると考えられる．以下にシステムの図を記す．

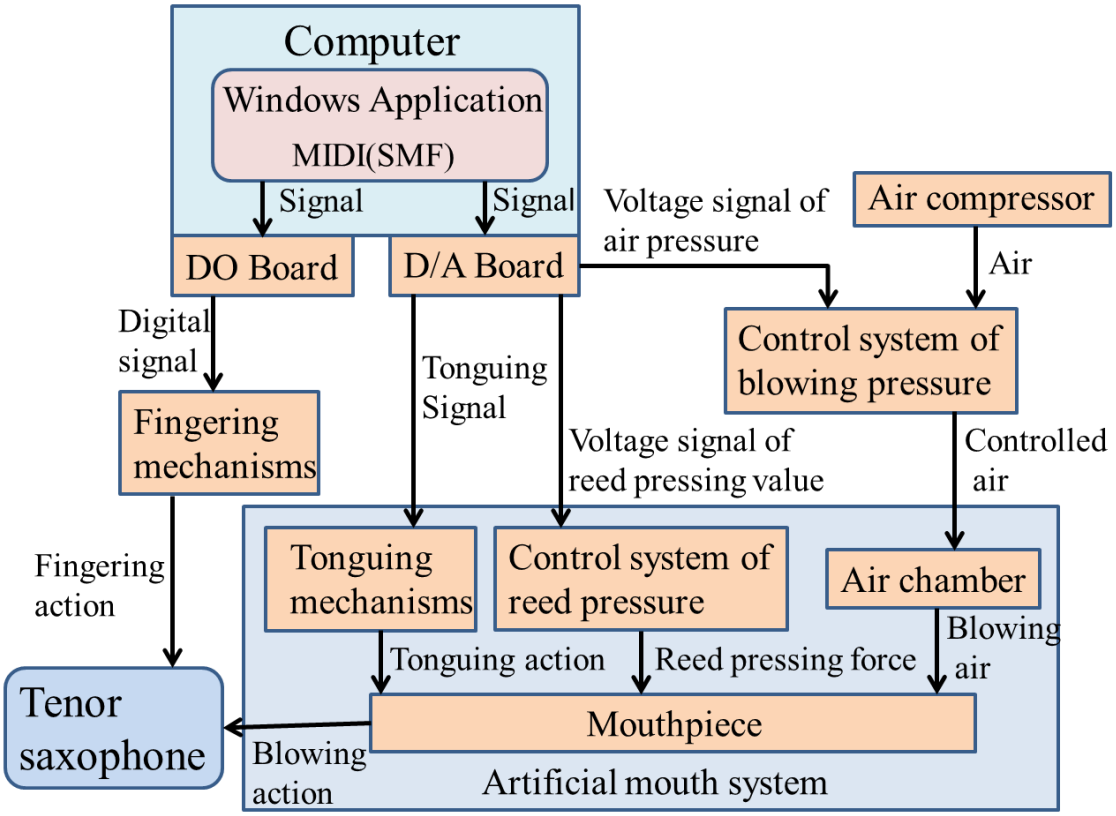


図 8. システム概略図

3.3.1　人工口顎部装置

人工口顎部装置[1)2)3)]は，適切なアンブシェアを再現した状態で，マウスピースに空気流を送り込むことができるような機構として製作されたものである．

自動演奏を行う場合，リードを押さえる力を制御する機構と，押さえる位置を移動する機構を持ち，なおかつマウスピースに空気を送り込むことが出来る構造でなければならない．これらの機構とマウスピースを取り付けることが可能な状態を作り出すために，マウスピースをアルミ角パイプとアクリル板を加工した気密室内部に固定し，リードを押さえる力を制御する機構と，押さえる位置を移動する機構，再び舌でリードを押さえることにより，空気を止めるタンギング機構を気密室内部に取り付けた構造となっている．人工口顎部装置の図を図 9 に記す.

3.3.2　人工唇

人間がサクソフォンを演奏する場合，マウスピースをくわえ，下唇を下前歯に巻き込むようにしてリードを適当な力で押さえる．このとき唇は，演奏に適した形と硬さになっており，適切なアンブシェアと空気流により正しい音を吹鳴することが出来る[1)2)3)].

この機構は，人間が演奏するのと同様な音質となるような唇として，様々な材料を検討し，結果，唇の柔らかく振動する部分をゴム風船にシリコンオイルを充鎮したもの，下前歯にかぶせた堅い部分をアメゴムチューブとし，これら 2 つを組み合わせたものを人工唇として用いている．また，歯の役割として唇部に突起物を取り付けている．

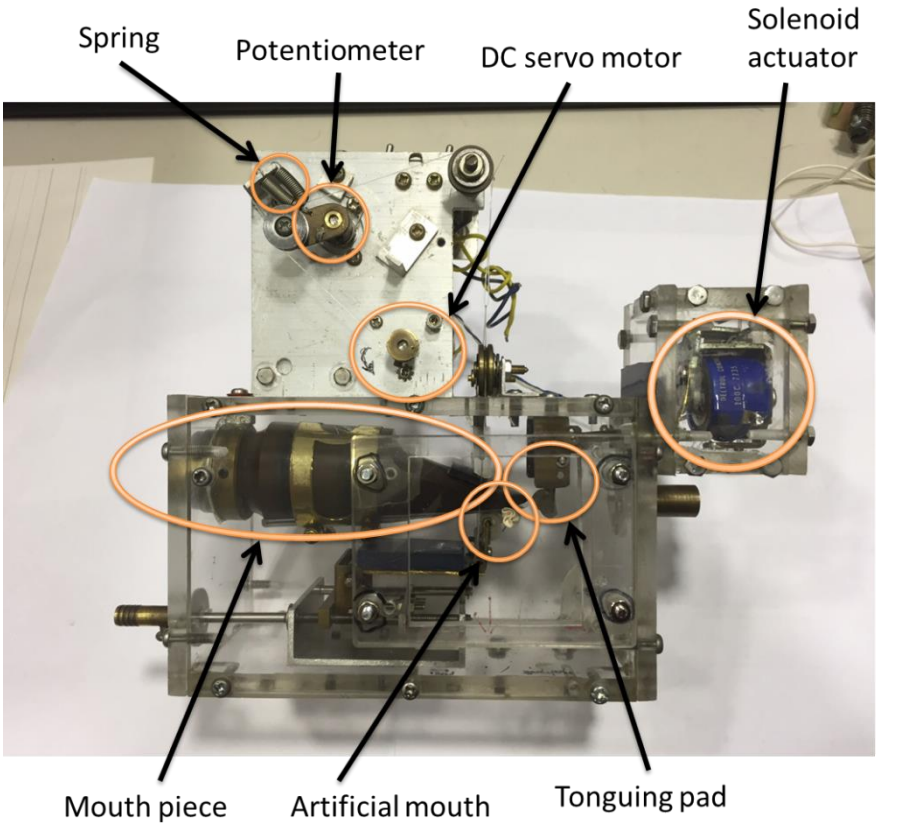
3.3.3　押さえ圧制御機構

サクソフォンを正確なピッチで吹鳴するには，音ごとにリードを押さえる力を微妙に制御する必要がある．

リードを押さえる機構は回転レバーが用いられ，釣り糸を介して張力制御機構で制御することで，下前歯部と人工唇部を適切な強さで押さえる．張力制御機構は，オリンパス製 DC サーボモータ（CL3A12D-G300）と，日本電産サーボ製ポテンショメータ（N28B）で構成されている．これは DC サーボモータに目標電圧を与えて釣り糸を巻き取っていくと，ポテンショメータに取り付けられたバネ付きのアーム（バネの力でもとの角度に戻ろうとする）が回転し，目標電圧とポテンショメータの出力電圧の偏差がなくなったところで停止する構造(図 9)となっている [1)2)3)].

3.3.4　タンギング機構

タンギングは，舌を用いて楽器の音を止める動作である．タンギング機構は，シリコンゲルのタンギングパッドをソレノイドアクチュエータで駆動することで，マウスピースの先端を押さえ，音を止める機構となっている [1)2)].


Spring
Potentiometer
DC servo motor
Solenoid actuator
Mouth piece
Artificial mouth
Tonguing pad

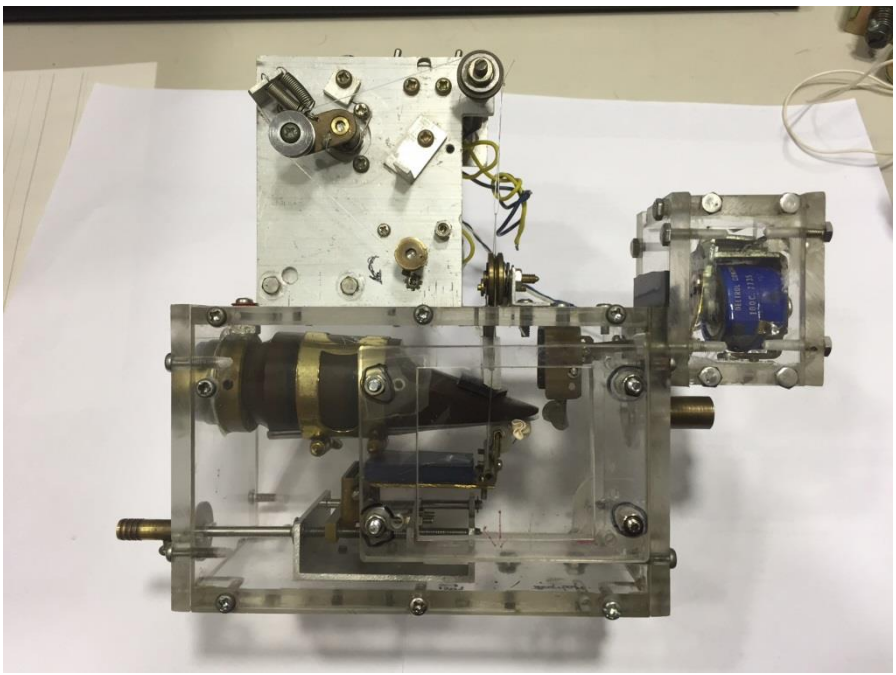


図 9. 人工口顎部装置

3.3.5　送気システム

送気システムは，人間の肺に相当するエアーコンプレッサ，その空気の圧力変動を抑制する役割を持つエアータンク，送気の圧力を変えるための送気圧制御装置によって構成されている[1)2)].

送気にはエアーコンプレッサ（日東工器製 AC0902）が用いられている．このコンプレッサの送気圧力は 20kPa であり，人間がサクソフォンを演奏するときの口腔内の圧力 10kPa よりも高くしている．そこで，エアータンクにて空気流を整流し，送気圧制御装置のリリーフバルブにて 0kPa～10kPa に減圧している構造となっている．

この装置は，自作の黄銅製リリーフバルブと多摩川精機製 DC サーボモータ（TS3717N11 E3）とコパル電子製ポテンショメータ（JT30-340-500）により構成されている．

DC サーボモータに目標電圧を与え，モータ軸に取り付けられたカムでバルブを押さえる．この力と，バルブから逃げる空気圧がつり合ったところでバルブが停止し送気圧を決定する構造となっている．また，既存の装置同様，モータ軸にポテンショメータが取り付けられており，モータの回転角度をフィードバックし制御を行っている．送気圧制御装置を図 10 に示す．

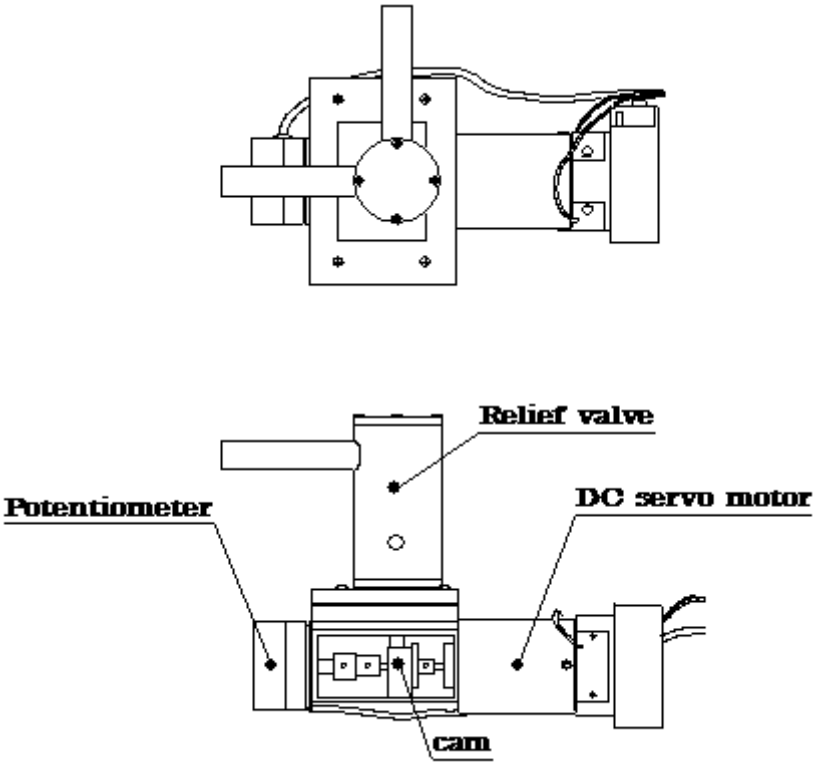


図 10. 送気圧制御装置

3.3.6　運指機構

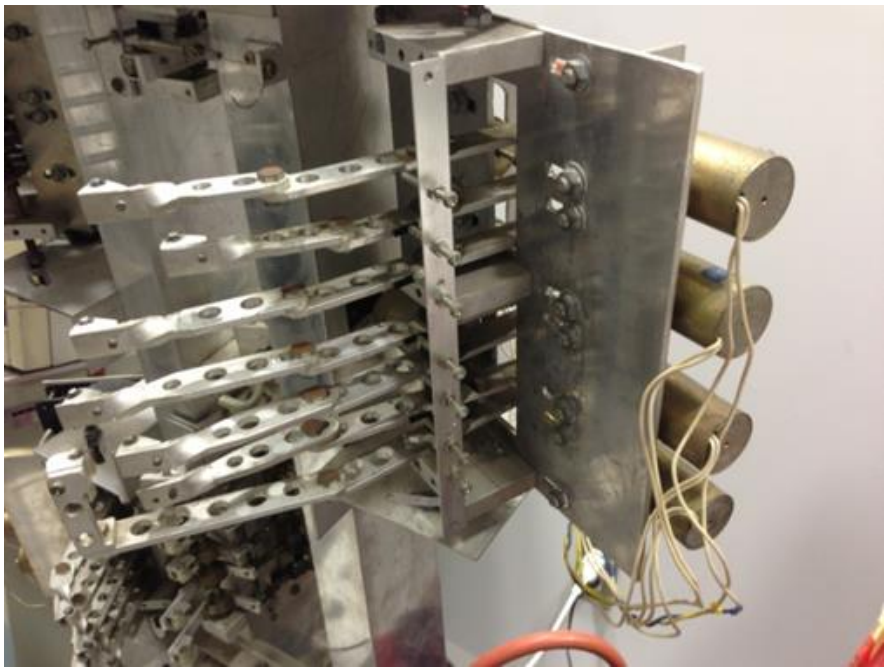
サクソフォンを吹鳴する際，管体にある 22 個のホールと 2 個のオクターヴホールをキーによって操作することで音階をつくる．人間の場合，両手で楽器を保持しながらこれらのキーを 9 本の指で操作している．しかしながら，人間同様に操作する装置を製作することは困難であるため, 1 つのキーに 1 本の指を対応させた運指機構となっている [1].

この機構は，サクソフォンの老朽化，楽器自体の特性を知りたいときなどに交換の必要があるため，メーカーによりキーの位置と角度の異なったテナーサクソフォンのすべてに対応できることを目的とし，ソレノイドアクチュエータと指部の取り付け位置及び，指部の角度，ストロークの変更が可能な構造となっている．

その構造は，2 本のアルミ角パイプとアルミのアングルで作られた本体に，5 つのユニットを取り付けたものである．ユニットには指の駆動源となるソレノイドアクチュエータが取り付けられ，伸縮可能なロッドを介し，てこの原理でキーを押さえている．

また，キーによっては指を常に押した状態になってしまうことがある．そのためソレノイドアクチュエータに電流が流れ続け，その結果として熱を持ってしまう．このことを防ぐために，2 つのファンが取り付けられ冷却している構造となっている．図 11 に運指機構とユニットを記す.

なお，動かす運指(ソレノイドアクチュエータ)は全部で 23 である．



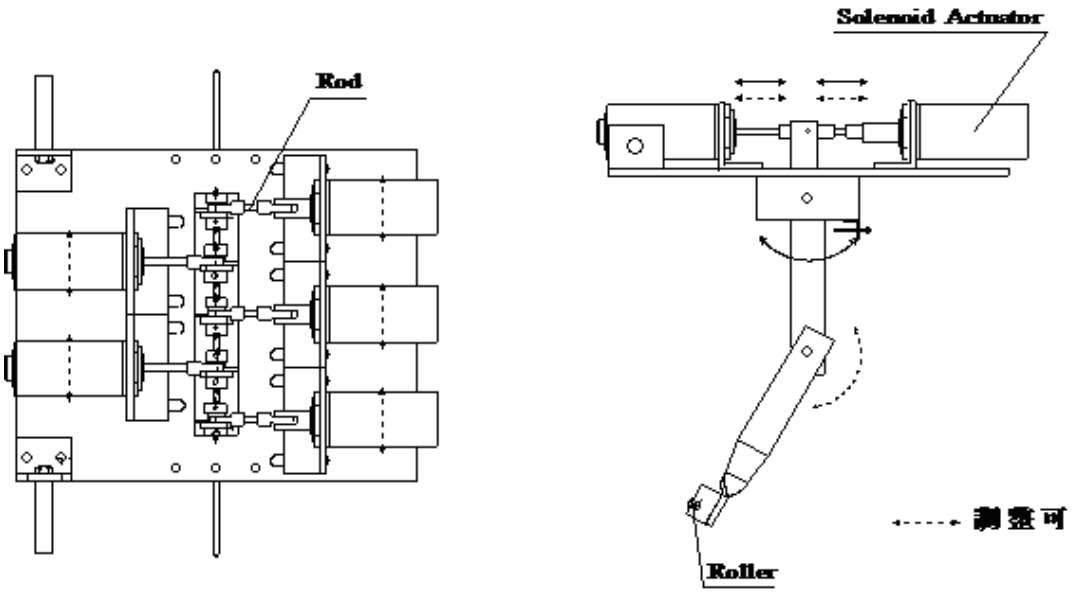


図 11　運指機構

3.3.7　制御用コンピュータ

制御用コンピュータの概要は，以下に記す.

表 4　制御コンピュータの仕様

CPU	Intel Corei 5　2.67GHz
メモリ	4GB
OS	Windows 7 Enterprise
システム	32 ビット

3.3.8 D/A 変換ボード

人工口顎部装置の唇押さえ圧およびタンギング，送気システムの送気圧を制御するために用いられている D/A 変換ボードの仕様を以下に記す.

出力レンジは±5V，±10V，0～10V の 3 つの設定があり，送気圧やタンギングに関してはチャンネルごとに出力レンジを設定することが可能である．当研究室のサクソフォン演奏ロボットシステムでは，唇押さえ圧は 0～－10V，送気圧は 0～10V の範囲を使用するため，バイポーラ±10V に各チャンネルを設定している．この設定では約 1／1000V 刻みの分解能を得ることができる.

表 5　D/A 変換ボードの仕様

メーカー	Interface 社
型番	PCI-3343A
出力数	12Bit 4 チャンネル
出力レンジ	DC0～±10V バイポーラ，ユニポーラの設定が可能

また，ボード側のコネクタピンアサインメントの各ピンに対するチャンネルとその対象を以下に記す．ただし，チャンネル 1 に関しては，実験中にマイナスの電圧が出力されなくなるという症状が起きたため使用していない.

表 6　D/A 変換ボードの各ピンの詳細

ピン番号	チャンネル	対象	
1	1	VOUT	未使用
2	2		唇押さえ圧
3	3		タンギング
4	4		送気圧
5	1	VCOM	未使用
6	2		唇押さえ圧
7	3		タンギング
8	4		送気圧
9～34	―	―	未使用

3.3.9 DIO ボード

運指機構を制御するためのデジタル入出力ボード（DIO ボード）の仕様を以下に記す.

表 7　DIO ボードの仕様

メーカー	Interface 社
型番	PCI-2403A
出力数	32Bit
出力レンジ	DC 0～5[V]

サクソフォン自動演奏ロボットの指の数は 23 であることから，指一本に対し出力を 1 つ当てている．出力データは int 型配列へのポインタとして指定されている．int 型配列は出力する点を 1，そうでない点は 0 と指定されている．

ボード側のコネクタピンアサインメントの各ピンに対応する対象を以下に記す.

表 8　DIO ボードの各ピンの詳細

ピン番号	対象
1，2	未使用
3～10	OUT 接点 1～8
11	－COM 接点 1～8
12～14	未使用
15～22	OUT 接点 9～16
23	－COM 接点 9～16
24～26	未使用
27～33	OUT 接点 17～23
34	未使用
35	－COM 接点 17～23
36～50	未使用

第４章　演奏制御アプリケーション

4.1　sax.exe，sax.cpp

SMF を用いた自動演奏システムを制御するアプリケーションとして sax.exe[2)3)] を用いている．この sax.exe のメイン制御プログラムは C 言語 [6)]で作成されている sax.cpp である．つまり sax.cpp のメインプログラムに新たにプログラムの変更と書き加えを行うことで新たに機能追加，変更ができ，アプリケーションの機能向上に繋がる．以下にアプリケーション画面を記す.

図 12. アプリケーション画面

4.2　アプリケーション使用方法

SMF を用いた自動演奏システムを制御するアプリケーション sax.exe の使用方法は

①チューニングボタンよりチューニングダイアログを開き，サクソフォンのチューニングを行う

(この時リード・送気圧の圧力値をダイアログが開いた時に自動で読み込む)

リード圧，送気圧ともに 0～999 の範囲で設定できる．

図 13. チューニングダイアログ画面

②ファイルボタンを押し，自動演奏に使用する SMF(拡張子が.mid)を選択しバイナリデータとして読み込む
(この時，各トラックの楽器，テンポ，調，拍子などの曲情報を検出する)

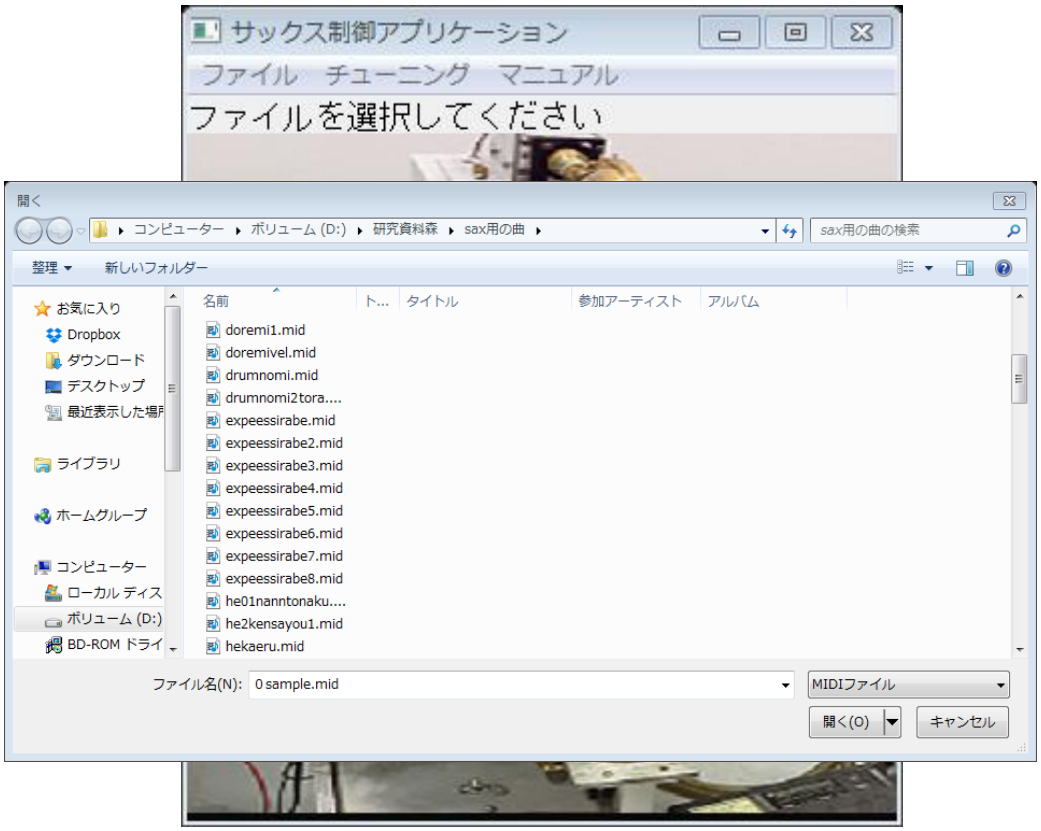


図 14. ファイル選択画面

③②の過程で読み込んだ各トラックの楽器情報がコンボボックスに表示される
(この時，同時にチューニングデータを読み込む)
自動演奏に使用したい楽器を選択して「実行」ボタンを押すと選択した楽器を除いた SMF を作成する

図 15. 楽器選択画面

④③の操作後実行画面に「全再生」ボタン,「ロボットのみ演奏」ボタンができ，押すことによりサクソフォン自動演奏ロボットの各機構にインターフェースボードを介して作動指令を送出し自動演奏させる
(この時,「全再生」ボタンを押した場合には演奏と同時に③で作成した SMF の音源を再生する)
なお，Esc(エスケープ)キーを押すことで自動演奏ロボットの演奏と音源の再生を途中で停止，Tab キーを押すことで SMF の音源のみ停止することができる．

図 16. 再生ボタンが出てきた時の画面

4.3　アプリケーションの問題点

以上のようにアプリケーションを使用する.

　しかし，SMF でサクソフォン自動演奏ロボットの
・演奏，音源の再生が問題なくできるもの
・再生しかできないもの
・演奏，再生ができないもの
など SMF の中の状況の様々な状態に対応できておらず，問題，不具合が生じてしまう SMF が存在していた.
また，演奏開始時にロボットの演奏と同時再生用の MIDI の音源のズレが生じ，ロボットの演奏の方が先に始まるという問題が起きていた.
なので，演奏表現の拡大の目的の前に以上の問題点を解決し，どんな SMF を用いても演奏ができ，音源とのズレをなくせるように sax.cpp に変更を行った.

第５章　SMF の処理に関しての問題点

5.1　演奏の問題点

4.2 で述べた問題でサクソフォン自動演奏ロボットが演奏をできる SMF と演奏をできない SMF がどのように違うのか，MIDI 音楽編集ソフトを用いて SMF を作成し, バイナリエディタで中の状況を見て, SMF の中の状況, および sax.cpp のプログラムを調べた.

5.2　フォーマット形式の違い

問題点においてまず，考えられたのが SMF のフォーマットの形式の違いである. SMF を用いた自動演奏システムはフォーマット 1 でのみ演奏が可能である. このことからフォーマット 0 では演奏はできない．しかし，調べた結果 Domino などの MIDI 音楽編集ソフトは基本 SMF の書き出し(保存)形式をフォーマット 1 で行われているのである．なので，この形式に関しては問題に関係なかった.

5.3 MIDI メッセージの処理の違い

通常楽器の音を鳴らす，音を止めるという動作をそれぞれノート・オン，ノート・オフの MIDI メッセージで扱っている．しかし，調べた結果，既存の自動演奏システムのメイン制御プログラム sax.cpp ではノート・オフを MIDI メッセージの特殊なルールの「ベロシティの値が 0 のノート・オンはノート・オフとみなす」を用いており，通常のノート・オフには対応しておらず，用いることができていなかった．なので，サクソフォン自動演奏ロボットが演奏できない SMF が存在していた．以下に処理に関しての図を記す．

ド(ノート・オフ)		
80	3C	40
ステータス・バイト	データ・バイト1	データ・バイト2

図 17. 通常のノート・オフ処理例

ド(ノート・オフ)		
90	3C	00
ステータス・バイト	データ・バイト1	データ・バイト2

図 18. 特殊なルールのノート・オフ処理例

5.3.1　ランニング・ステータス

この特殊なルールのノート・オフを用いる理由としてはランニング・ステータスという MIDI メッセージのデータ量を圧縮するテクニックを行うために用いる．以下に例を記す.

ド(ノート・オン)		
90	3C	7F
ステータス・バイト	データ・バイト1	データ・バイト2

ド(ノート・オフ)		
80	3C	40
ステータス・バイト	データ・バイト1	データ・バイト2

図 19. 通常のメッセージ処理例

ド(ノート・オン)		
90	3C	7F
ステータス・バイト	データ・バイト1	データ・バイト2

ド(ノート・オフ)		
~~90~~	3C	00
~~ステータス・バイト~~	データ・バイト1	データ・バイト2

図 20. ランニング・ステータスを用いた処理例

ランニング・ステータスとは「一つ前のステータス・バイトと同じステータス・バイトの時は省略できる」というものである．なので，図 16 のように特殊なルールのノート・オフを使った場合はランニング・ステータスを用いることができるので, データ量が通常のメッセージ処理(図 19)の場合が 6 バイトであるのに対して，ランニング・ステータスを用いた処理(図 20)では 5 バイトとなりデータ量の圧縮ができるのである．なお，このランニング・ステータスを用いるかどうかは各々のメーカーの裁量に任されている.

5.4　SMF 読み取り処理の拡大

この特殊なルールのノート・オフを用いず, 通常のノート・オフを用いる SMF は多々存在する. また, Domino などの MIDI 音楽編集ソフトでは通常のノート・オフを用いて SMF が作成されているものも多い.

そこで, 既存のメイン制御プログラム sax.cpp に通常のノート・オフ(ステータス・バイトが 8n のもの)でも対応できるように書き加えた.

5.4.1　演奏のプログラムの変更点

通常のノート・オフの処理が行えるように変更を行った. 付録 2 に変更を行った sax.cpp のプログラムを記載する. プログラムを書き換えることで通常のノート・オフでも特殊なルールのノート・オフでも演奏が行えるようになった.

5.5　変更結果

sax.cpp のプログラムによりどのようにサクソフォン自動演奏ロボットが演奏するのかテキストファイル形式で書き出す機能を備えており，その機能を用いて演奏できているのか確かめることができる．図 21 にプログラム変更前の状態を記す.

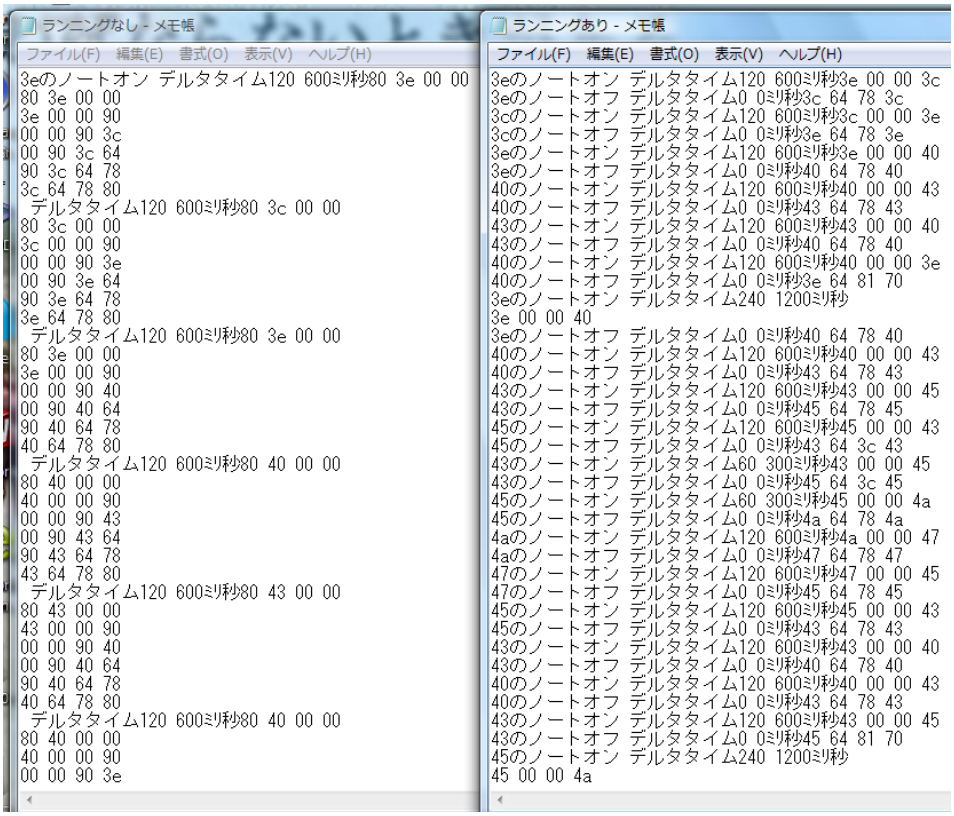


図 21. プログラム変更前の演奏状態

この図 17 は同じ楽譜の通常のノート・オフの処理をしている SMF と特殊なルールのノート・オフの処理をしている SMF である．左が通常のノート・オフ処理(ランニング・ステータスを用いていない)の SMF のテキストファイル，右が特殊なルールのノート・オフ処理(ランニング・ステータスを用いた)の SMF のテキストファイルである. そして, サクソフォン自動演奏ロボットは左の SMF は演奏ができないが，右の SMF は演奏ができる．左のテキストでは最初のノート・オン以降ノート・オフができておらず，表示がされていない．なので，演奏ができない．右のテキストではノート・オフが表示されている．なので，演奏ができる．次にプログラム変更後を図 22 に記す.

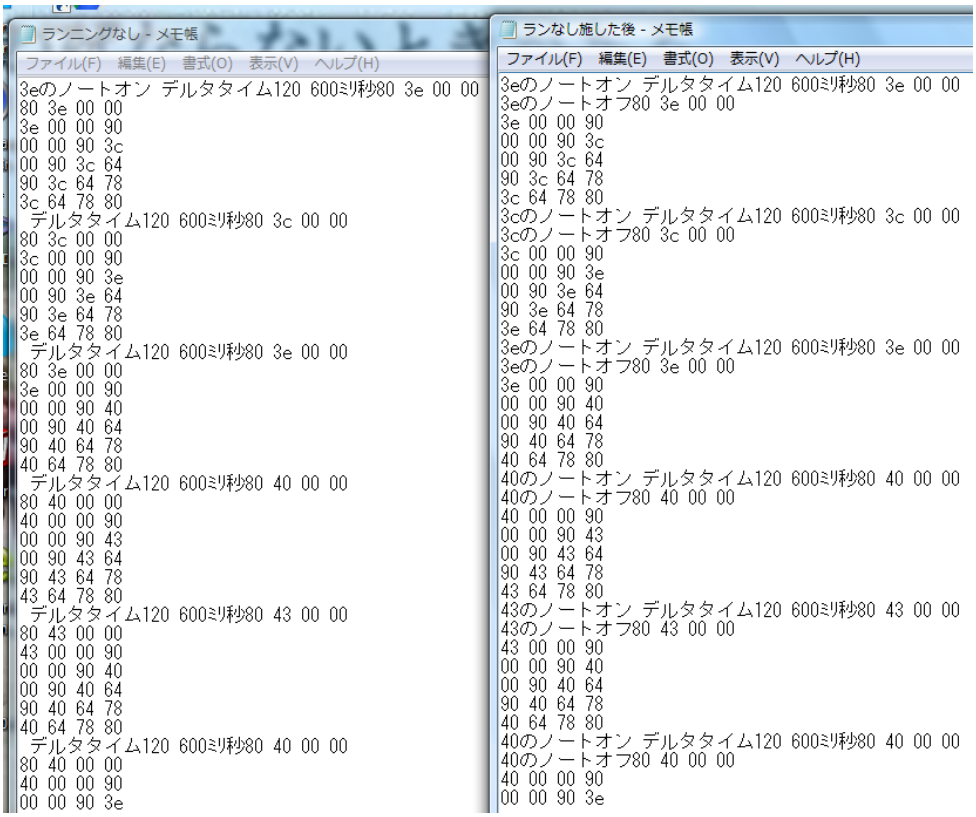


図 22. プログラム変更後の演奏状態

右がプログラムを変更した後の通常のノート・オフの処理の SMF のテキストで, 左が先ほどの変更前の SMF のテキストである. プログラム変更後はノート・オフも表示されて，演奏が可能となった.

以上より変更が上手くいき，様々な SMF に対応し，サクソフォン自動演奏ロボットが演奏できるようになった.

第６章　音源の再生について

6.1　音源の再生の問題点

4.2 で述べた問題で音源の再生ができる SMF と再生ができない SMF がどのように違うのか，こちらも第 5 章同様に MIDI 音楽編集ソフトを用いて SMF を作成し，バイナリエディタで中の状況を見て，SMF の中の状況，および sax.cpp のプログラムを調べた.

6.2　MCI

音源の再生に関しては MCI を用いて行っている.

MCI(Media Control Interface)とは音源や動画などのファイルや機器を制御するもののことである.

6.3　MCI エラー

この MCI エラーが起きることにより，音源の再生ができなくなっていた．以下にエラー時の画面を記す.

図 23. MCI エラー画面

この MCI エラーはコンボボックスから自動演奏に用いる楽器を選択し,「実行」ボタンを押すと図 23 のようになり，エラーが起き音源が再生されない．なお，演奏の方は問題なく行えるので音源がなく，サクソフォン自動演奏ロボットのみが演奏している状態になる.

6.3.1　エラーが起きる原因

　この MCI エラーが起きる原因として
①MIDI ファイルそのものが破損している

②既存のメイン制御プログラム sax.cpp がいくつかの SMF に対応していない

③それぞれの MIDI 音楽編集ソフトで独自の機能があり，それが含まれた SMF なので対応できていない

ということが考えられた.

　以上から②③よりそれぞれの MIDI 音楽編集ソフトの独自の機能，SMF の構造，制御プログラムの中身等を調べ，対応できるようにメイン制御プログラム sax.cpp に新たに書き加えた．なお，本研究で調べた主な MIDI 音楽編集ソフトは有名なソフトとして，Domino[7)]とランニング・ステータスで SMF が作成されるソフトで Musescore[8)]を調べた.

6.4　エラーを調べた結果(Domino)

エラーが起きる原因を調べた結果チャンネル(トラック・チャンク)数とそれぞれの MIDI 音楽編集ソフトの独自機能が影響していた. 以下に Domino[7]のチャンネル(トラック・チャンク)構成を記す.

A1	System Setup
A1	楽器
A2	楽器
A3	楽器
A4	楽器
A5	楽器
A6	楽器
A7	楽器
A8	楽器
A9	楽器
A10	楽器
A11	楽器
A12	楽器
A13	楽器
A14	楽器
A15	楽器
A16	楽器

図 24. Domino 構成

通常, MIDI チャンネルごとにトラック・チャンクに分けて格納する. よって, MIDI チャンネルは最高 16 チャンネルなのでトラック・チャンク数も 16 であるはずである．しかし，例外もあり，同一の MIDI チャンネルを複数のトラック・チャンクに分けて格納する SMF もある．Domino は例外が当てはまっており，A1 チャンネルが二つに分かれて独自の A1System Setup のトラック・チャンクを持っている．なので，トラック数が 1 つ多く通常は最高で 16 のところ，Domino では最高 17 になってしまっている．sax.cpp を調べると 16 までの対応になっていた.

これより，メイン制御プログラム sax.cpp に独自の A1 System Setup のトラック・チャンクを回避させて，最高 16 にするように書き加えた.

6.4.1　エラーを調べた結果(Musescore)

Musescore は音源の再生，演奏について問題は起きなかったがコンボボックスに最初の楽器だけ表示されない問題が発生していた．これは通常最初のトラック・チャンク内には演奏情報が入ることはないが，Musescore[8)]では入ってしまっているからであった．sax.cpp でも二つ目以降から演奏情報を抜きだすようにプログラムが書かれていた．なので，最初のトラック・チャンクに演奏情報が入った場合にもコンボボックスに表示できるように書き加えた．

6.5　音源の再生のプログラム変更と結果

以上より付録 2 に変更を行った sax.cpp のプログラムを記載する．

変更を行ったことで問題なく認識されるようになった．よって，様々な SMF に対応し音源の再生ができるようになった．

6.6　自動演奏と音源とのズレが起きる原因の解消

ロボットの演奏開始時に同時再生用の MIDI の音源とズレが生じてしまう原因として, 同時再生 MIDI 音源の方は音源デバイスを起動してから鳴らすため少し遅れるのではないかと考えられた.

同時再生 MIDI 音源の方は MCI を使用していることから MCI コマンドメッセージを使用することで解決する方法を考えた．MCI コマンドメッセージの一覧を表 9 に記す.

表 9. MCI コマンドメッセージ一覧

Operations	Command message
Open the device	MCI_OPEN
Play	MCI_PLAY
Stop	MCI_STOP
Close the device	MCI_CLOSE
Pause	MCI_PAUSE
Resume	MCI_RESUME
Scrape the playing time	MCI_STATUS_LENGTH
Scrape the current position	MCI_STATUS_POSITION
Scrape the current mode	MCI_STATUS_MODE

MCI の起動を早くすることは困難であるので自動演奏ロボットの演奏開始を遅らせるため MCI が起動されるまで自動演奏ロボットの演奏を行わないようにした．このために，表 1 にある現在の状態を取得する MCI_STATUS_MODE を使用した．このコマンドメッセージを送ることで現在の状態を取得できるので現在の状態が再生中(MCI_PLAY)となるまでは自動演奏ロボットを待機させて，再生中となった時に自動演奏ロボットが演奏を開始するようにプログラムを書き加えた．結果，演奏開始時のズレはなくなり，この問題は解決できたと考えられる．図 25 に結果の例を記す.

図 25. 演奏待機状態表示例

第７章　新たな演奏技法の追加

7.1　新たな機能について

以上によりアプリケーションの問題点の修正を行ったので，演奏技法を増やすために新たにメイン制御プログラム sax.cpp に書き加えを行い，認識できる MIDI メッセージを増やし，認識した際の処理についてアプリケーションに演奏技法の追加をした.

追加した MIDI メッセージに関しては

①一つ一つの音の音量を決めるベロシティの導入

②演奏音の音量をだんだん小さくしたり，大きくしたりするエクスプレッションの導入

③ピッチを変更するピッチベンドの導入

④音にビブラート(音の揺れ)を加えるモジュレーションの導入

⑤これらの導入にともない，チューニングダイアログの変更

⑥これらの導入にともない，演奏中に新たなる機能の追加

を sax.cpp に書き加えをした(付録 2). 音を取る実験風景を図 26 に記す.

図 26. 演奏実験風景

7.2　演奏精度の向上

チューニングダイアログで設定したリード圧，送気圧が演奏時のリード圧，送気圧にうまく反映されず，音が出ない，出にくいという問題が起きていた．この問題を解決するために原因を調べた．

調べた結果，演奏中に使用している値はチューニングダイアログで設定した値の小数点以下は切り捨てて用いていた．つまり，4.81 と値を設定したとしても実際は4.00の値になってしまっていた．これは今までのプログラムではdoubleや flot の関数を用いず，int を使用していたことが問題であった．これではチューニング中には音が鳴っていたとしても演奏中の演奏の表現がとても狭くなるので音が出にくくなってしまうことがわかる．チューニングをいかに細かく設定したとしても演奏中には反映されない．現在用いている D/A 変換ボードは12bit の分解能であるので 2^{12}=4096 までの数の表現ができ，用いる電圧の範囲は±10V の 20V で-10V～0V が 0～2048，0～10V が 2048～4096 となっている．また，これより 20÷4096≒0.005V(5mV)まで読み取ることのできる精度を持っている．

以上よりチューニングの値と同じ値を演奏中に用いることができるようにプログラムの書き換えを行った．その結果，チューニング中に設定した値を読み込むことができるようになり，演奏表現の幅が広くなり，演奏精度の向上にもなった．

7.3　ベロシティの導入

ベロシティ[5)]とは鳴らした際の音の強さ(音量)を表す. 範囲は 1～127 であり, 127 が最大で 1 が最小の音量である．2.2.2 で記した通りノート・オン・メッセージ, ノート・オフ・メッセージの 2 つ目のデータ・バイトに出てくる(ノート・オフ・メッセージでは基本 64 固定). なお, ベロシティには音量範囲が定められている(表 10).

表 10. ベロシティ範囲

表記	読み	ベロシティ値
ppp	ピアニッシッシモ	16
pp	ピアニッシモ	32
p	ピアノ	48
mp	メゾピアノ	64
mf	メゾフォルテ	80
f	フォルテ	96
ff	フォルティッシモ	112
fff	フォルティッシッシモ	127

これよりノート・オン・メッセージの 2 つ目のデータ・バイトを読み取れるようにプログラムを書き加えることで認識はできるようになる.

認識した際の処理に関しては表 10 より少なくとも 8 通りの音量変化があるということがわかる．しかし，いきなり音量変化を 8 通りに変えられるようにするのは音量変化が分かりにくく，区別できなくなる可能性がある.

なので，今回はベロシティの値が

・81～127 を音量強

・48～80 を音量中,

・1～47 を音量弱

として 3 種類の音量変化を出せるように試みた.

7.3.1　ベロシティの処理

7.3 よりベロシティの範囲を 3 種類に分けて音量変化を出せるようにするためにチューニングダイアログの変更を行った．変更点としてはチューニングダイアログに強，中，弱のボタンを新たに増やし，チューニングデータを強，中，弱と 3 種類保存できるように変更した．そして，7.3 よりベロシティの値によって各々対応したチューニングデータを呼び出し，演奏するという方法である．

以上よりプログラムを書き換えて新たに作成したチューニングダイアログを図 27 に記す．

図 27. 音量変化用チューニングダイアログ

図 27 により，3 種類のチューニングデータを保存できるようになった．

7.3.2　ベロシティ実験結果

実際にベロシティができるか実験を行った.

```
3aのノートオン1i=1,音階=58ベロシティ範囲ppp=  2 デルタタイム1920 fμ秒
81 3a 00 00
3a 00 00 91
3aのノートオフ3 デルタタイム0 fμ秒91 3a 3f 8f
3a 3f 8f 00
3aのノートオン1i=3,音階=58ベロシティ範囲mp= 63 デルタタイム1920 fμ秒
81 3a 00 00
3a 00 00 91
3aのノートオフ3 デルタタイム0 fμ秒91 3a 78 8f
3a 78 8f 00
3aのノートオン1i=5,音階=58ベロシティ範囲fff=120 デルタタイム1920 fμ秒
81 3a 00 81
3a 00 81 70
3aのノートオフ3 デルタタイム240 fμ秒
```

図 28. ベロシティによる演奏状態

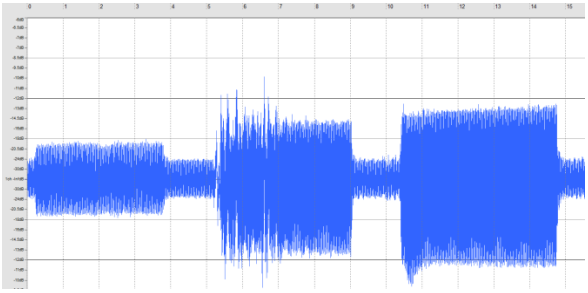


図 29. ベロシティによる音量変化実験

・実験状況
縦軸：音量(デシベル)
横軸：時間(秒)
テナーサックスのベル部先端から 300mm 離したところにベル部の先端に高さを合わせた AKG 製マイク(C3000)を設置し，Cubase[9]というアプリで音を取った.
設定値は
・1 つ目の音がベロシティの値 2 で音量範囲弱：リード圧(828)，送気圧(0)
・2 つ目の音がベロシティの値 63 で音量範囲中：リード圧(816)，送気圧(66)
・3 つ目の音がベロシティの値 120 で音量範囲強：リード圧(805)，送気圧(999)
の MIDI データを使用し，実験したものである.

図 28，図 29 より 3 つの音に対してそれぞれ強，中，弱のベロシティの認識，認識後の処理が行えており，音量に変化が表れていることがわかる.

以上で現状の音量範囲の設定内ではベロシティによる音量変化が行えるようになった.

7.4　エクスプレッションの導入，処理

エクスプレッション[5)]とは MIDI チャンネルごとに 128 段階(0～127)の音量変化(抑揚)を付ける MIDI メッセージであり，16 進数表記では

Bn 0B dd

n：MIDI チャンネル(1～16)

0B：コントロールナンバー

dd：設定値(0～127)

と記す．127 が最大で 1 が最小の音量である(0 は無音).

これより上記の MIDI メッセージを読み取れるようにプログラムを書き加えることで認識はできるようになる.

エクスプレッションも音量に関わることから 7.3 より同じ範囲を使用し, エクスプレッションの値が

・81～127 を音量強

・48～80 を音量中,

・1～47 を音量弱

として 3 種類の音量変化を出せるように試みた.

処理もベロシティ同様エクスプレッションの値によって各々対応したチューニングデータを呼び出し，演奏するという方法である.

7.4.1　導入する際の問題点の解消

エクスプレッションはノート・オン・メッセージとノート・オフ・メッセージの間に入る MIDI メッセージである．しかし，今までのプログラムでは間に MIDI メッセージを入れることを想定されていなかった．なので，ただエクスプレッションを認識できるようにしただけでは読み取ることができず，誤作動を起こしてしまい，演奏ができなくなってしまった(図 30 左)．以上から MIDI メッセージを間に入れられるようにさらに新たにプログラムを書き換えることで問題の解決を行った(図 30 右)．

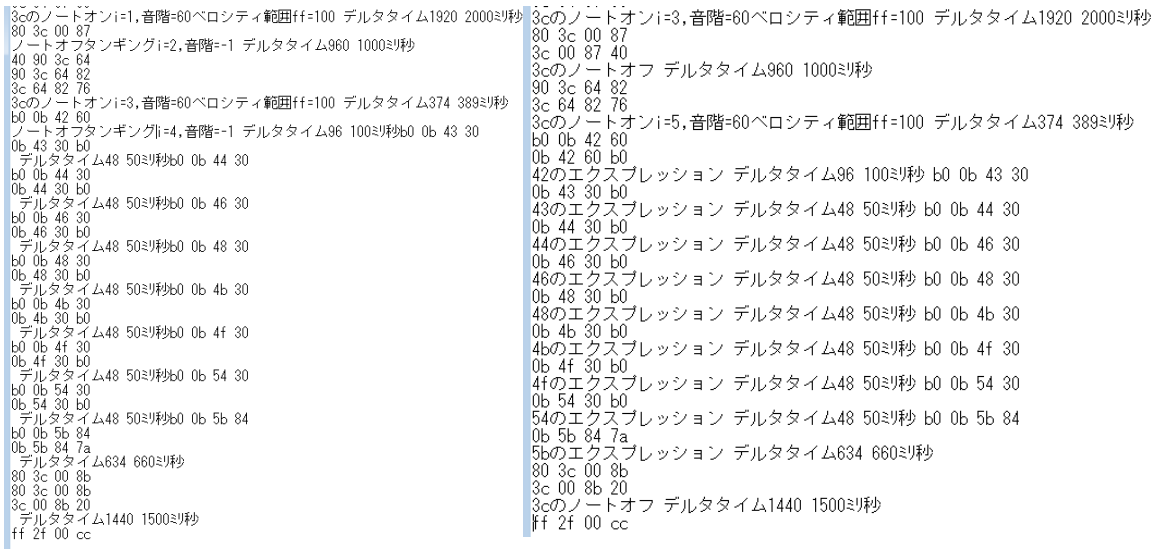


図 30. 読み取れるためのプログラム変更前(左)変更後(右)

7.4.2　エクスプレッション実験結果

実際にエクスプレッションができるか実験を行った.

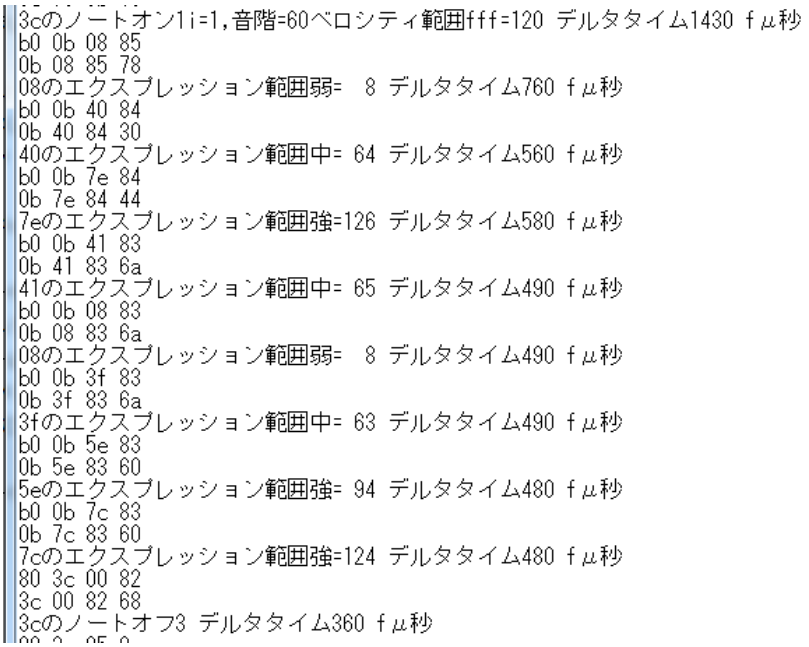


図 31. エクスプレッションによる演奏状態

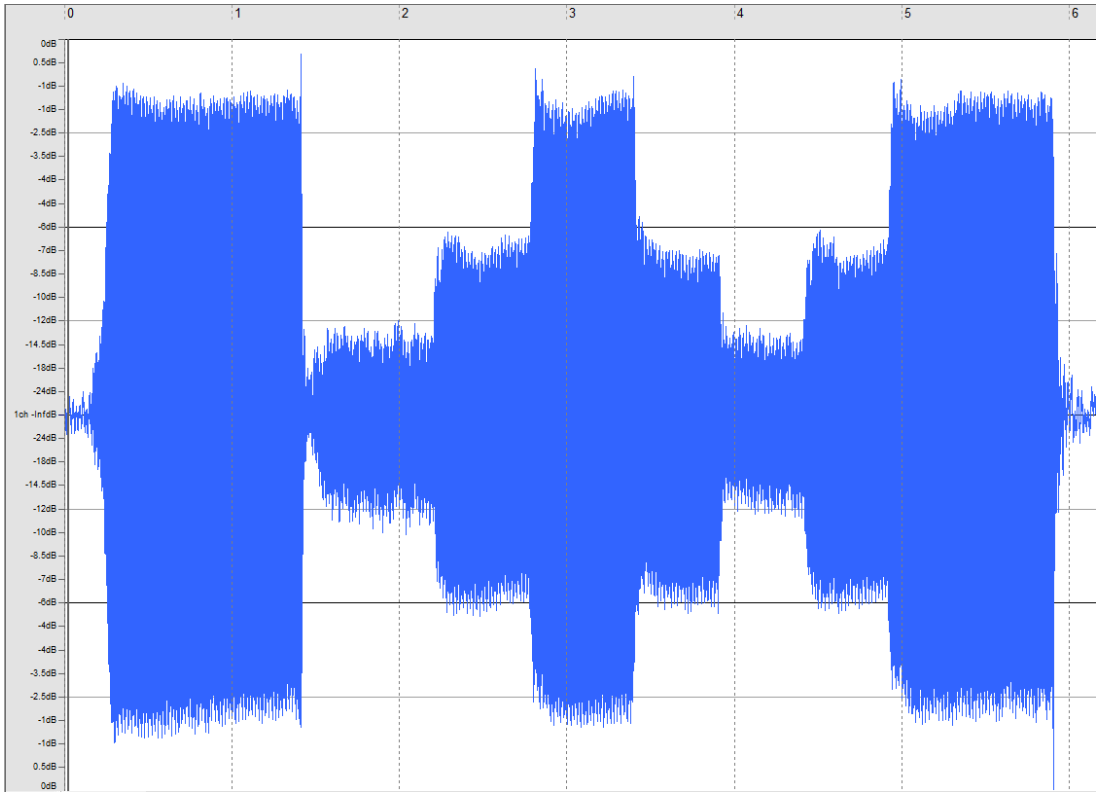


図 32. エクスプレッションによる音量変化実験

・実験状況
縦軸：音量(デシベル)
横軸：時間(秒)
テナーサックスのベル部先端から 300mm 離したところにベル部の先端に高さを合わせた AKG 製マイク(C3000)を設置し，Cubase[9]というアプリで音を取った．

図 31，図 32 は 1 つの音に対してエクスプレッションの値を 8，64，126，65，8，63，94，124 で設定した MIDI データを使用し，実験したものである．

図 31，図 32 より 1 つの音に対してそれぞれ強，中，弱のエクスプレッションの認識，認識後の処理が行えており，音量に変化が表れていることがわかる．

以上で現状の音量範囲の設定内ではエクスプレッションによる音量変化が行えるようになった．

7.5　ピッチベンドの導入，処理

ピッチベンド[5)]とは音色のピッチを調節する MIDI メッセージであり, 16 進数表記では

En dl dm

n：MIDI チャンネル(1～16)

dl：ピッチベンドの下位バイト(0～127)

dm：ピッチベンドの上位バイト(0～127)

と記す．dl と dm より最大 16384 段階(128×128)の解像度でピッチ変化のない基準値(0)を中心にプラス，マイナスの両方向に値が変化する．最大を 8191，基準値 0，最小値-8192 で表し変化する．エクスプレッション同様ノート・オン・メッセージとノート・オフ・メッセージの間に入る MIDI メッセージである．

7.4.1 の問題を解消することで上記の MIDI メッセージを読み取れるようにプログラムに書き加えることで認識し，上記の方法をプログラムに書き加えることで処理もできるようになる．

7.5.1　ピッチベンド実験結果

実際にピッチベンドができるか実験を行った.

```
3cのノートオン1i=2,音階=60ベロシティ範囲fff=120 デルタタイム1920 fμ秒
e0 7f 7f 3c
7f 7f 3c e0
 8191のビッチベンド デルタタイム60 fμ秒e0 00 00 3c
00 00 3c e0
-8192のビッチベンド デルタタイム60 fμ秒e0 7f 7f 3c
7f 7f 3c e0
 8191のビッチベンド デルタタイム60 fμ秒e0 00 00 3c
00 00 3c e0
-8192のビッチベンド デルタタイム60 fμ秒e0 7f 7f 3c
7f 7f 3c e0
 8191のビッチベンド デルタタイム60 fμ秒e0 00 00 3c
00 00 3c e0
-8192のビッチベンド デルタタイム60 fμ秒e0 7f 7f 3c
7f 7f 3c e0
 8191のビッチベンド デルタタイム60 fμ秒e0 00 00 3c
00 00 3c e0
-8192のビッチベンド デルタタイム60 fμ秒e0 7f 7f 3c
7f 7f 3c e0
 8191のビッチベンド デルタタイム60 fμ秒e0 00 00 3c
00 00 3c e0
-8192のビッチベンド デルタタイム60 fμ秒e0 7f 7f 3c
7f 7f 3c e0
 8191のビッチベンド デルタタイム60 fμ秒e0 00 00 3c
00 00 3c e0
-8192のビッチベンド デルタタイム60 fμ秒e0 7f 7f 3c
7f 7f 3c e0
 8191のビッチベンド デルタタイム60 fμ秒e0 00 00 3c
00 00 3c e0
-8192のビッチベンド デルタタイム60 fμ秒e0 7f 7f 3c
7f 7f 3c e0
 8191のビッチベンド デルタタイム60 fμ秒e0 00 00 3c
00 00 3c e0
-8192のビッチベンド デルタタイム60 fμ秒e0 7f 7f 3c
7f 7f 3c e0
 8191のビッチベンド デルタタイム60 fμ秒e0 00 00 3c
00 00 3c e0
-8192のビッチベンド デルタタイム60 fμ秒e0 00 40 95
00 40 95 48
    0のビッチベンド デルタタイム2760 fμ秒
80 3c 00 81
3c 00 81 96
3cのノートオフ3 デルタタイム19200 fμ秒
```

図 33. ピッチベンドによる演奏状態

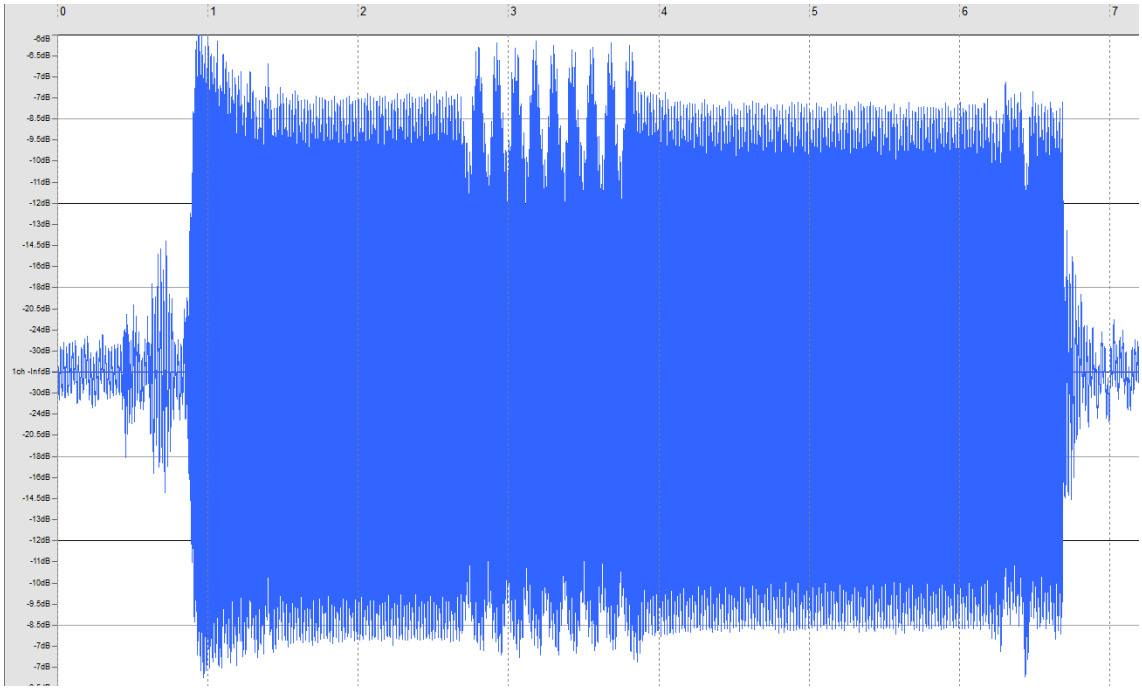


図 34. ピッチベンドによるピッチ変化実験

・実験状況
縦軸：音量(デシベル)
横軸：時間(秒)
テナーサックスのベル部先端から 300mm 離したところにベル部の先端に高さを合わせた AKG 製マイク(C3000)を設置し，Cubase[9]というアプリで音を取った．

図 33，図 34 は 1 つの音に対してピッチベンドの値を最大，最小の値を繰り返し設定した MIDI データを使用し，実験したものである．

図 33，図 34 より 1 つの音に対してピッチベンドの認識，認識後の処理が行えており，途中からピッチの変化が表れていることがわかる．なお，図 34 のような状態はビブラートのようになっている．つまり，ピッチベンドの MIDI メッセージを多様化することで疑似的にビブラートを作ることも可能である．

以上でピッチベンドによるピッチ変化が行えるようになった．

7.6　モジュレーションの導入，処理

モジュレーション[5)]とは鳴っている音に対して，音の揺れ(ビブラート)などの変調を加える時に利用する MIDI メッセージであり，16 進数表記では

Bn 01 dd

n：MIDI チャンネル(1～16)

01：コントロールナンバー

dd：設定値(0～127)

と記す．127 が最大で 0 が最小の変調である．エクスプレッション同様ノート・オン・メッセージとノート・オフ・メッセージの間に入る MIDI メッセージである．

7.4.1 の問題を解消することで上記の MIDI メッセージを読み取れるようにプログラムに書き加えることで認識する．

処理はビブラートを起こすために設定したリード圧に対し，ビブラートを起こす時間の間に正弦波を加えることで行えるようにした．

$$A \cdot (dd/127) \cdot \sin(2\pi \cdot f \cdot t)$$

A：振幅

dd：モジュレーションの設定値(0～127)

f：周波数(周期の値から算出)

t：時間(秒)

7.6.1　モジュレーション実験結果

実際にモジュレーションができるか実験を行った.

```
3cのノートオン1i=2,音階=60ベロシティ範囲mp= 59 デルタタイム2270 fμ秒
b0 01 7f 9b
01 7f 9b 22
7fのモジュレーション デルタタイム3490 fμ秒
80 3c 00 00
3c 00 00 90
3cのノートオフ3 デルタタイム0 fμ秒90 3e 3a 86
3e 3a 86 70
3eのノートオン1i=5,音階=62ベロシティ範囲mp= 58 デルタタイム880 fμ秒
b0 01 7f 88
01 7f 88 10
7fのモジュレーション デルタタイム1040 fμ秒
80 3e 00 81
3e 00 81 87
3eのノートオフ3 デルタタイム17280 fμ秒
```

図 35. モジュレーションによる演奏状態

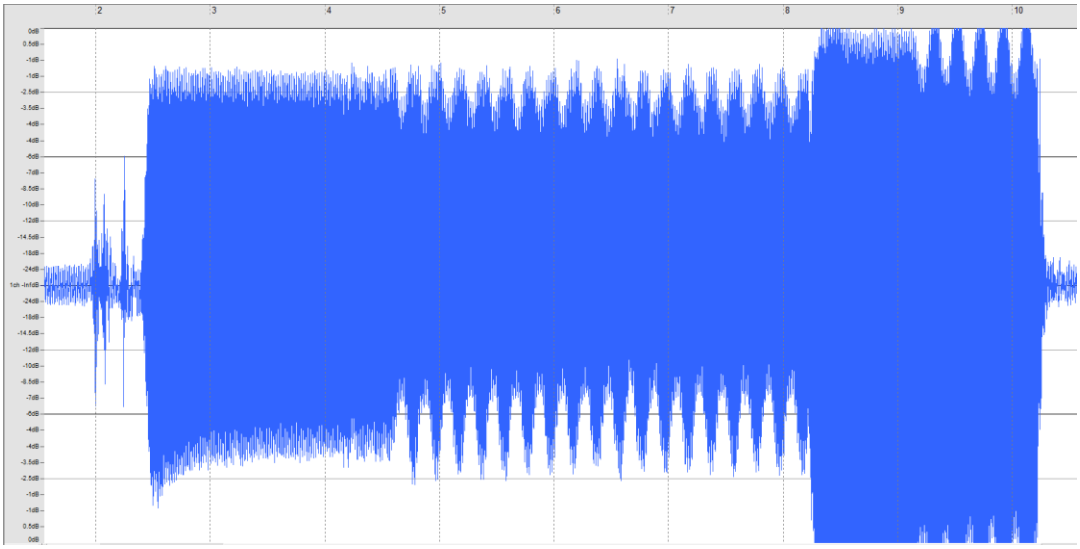


図 36. モジュレーションによるビブラート実験

・実験状況
縦軸：音量(デシベル)
横軸：時間(秒)
テナーサックスのベル部先端から 300mm 離したところにベル部の先端に高さを合わせた AKG 製マイク(C3000)を設置し，Cubase[9]というアプリで音を取った．

図 35，図 36 は 1 つ目，2 つ目の音にモジュレーションの値 127 で設定した MIDI データを使用し，実験したものである．

図 35，図 36 より 1 つ 1 つの音に対してモジュレーションの認識，認識後の処理が行えており，途中からビブラートが表れていることがわかる．

なお，図 36 のようなビブラートはピッチベンドのように MIDI メッセージを多様化せずに 1 つモジュレーションの MIDI メッセージをするだけでその音にビブラートがつくことが分かる．なので，ビブラートを行う際にはモジュレーションを使用した方が良いと考えられる．

以上でモジュレーションによるビブラートが行えるようになった．

7.6.2　ビブラート詳細設定の追加

ビブラートに使用している正弦波の振幅と周期は決めた値から変更することができずにいた．したがって，音量強の時でも中の時でも弱の時でも同じ値を使用していた．演奏する曲によって振幅や周期を変更できないのは演奏表現を狭めてしまい良くない．なので，チューニングダイアログに新たに振幅と周期の値を変更し設定できる項目を増やし，それぞれの音量に対し，値を設定できるようにした．また，実際に値を変更したことによってどんなビブラートになるか確認できるようにチューニングダイアログにビブラートボタンを増やし，チューニング中にビブラートをできるようにした．以上の変更を加えたチューニングダイアログを図 37 に記す．

図 37. ビブラートができるチューニングダイアログ

7.6.3　ビブラート詳細設定の実験結果

実際にチューニングダイアログで振幅，周期の値を変更しつつビブラートボタンを使用し，実験した結果を記す．なお，縦軸音量(デシベル)，横軸(秒)で振幅の単位は V(電圧), 周期の単位は cs(センチ秒), ビブラート時間は 3 秒である．

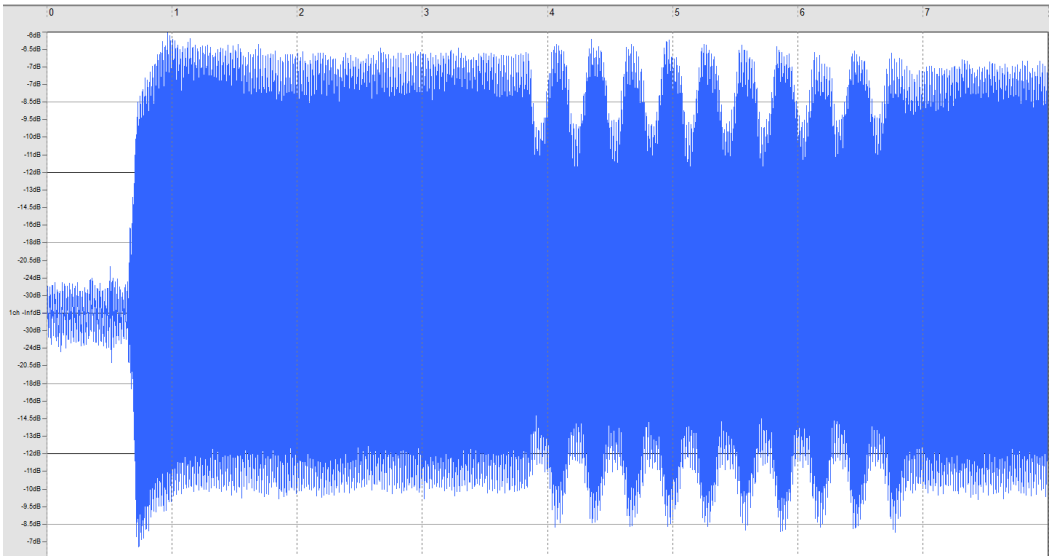


図 38. ビブラート実験(振幅 30，周期 30)

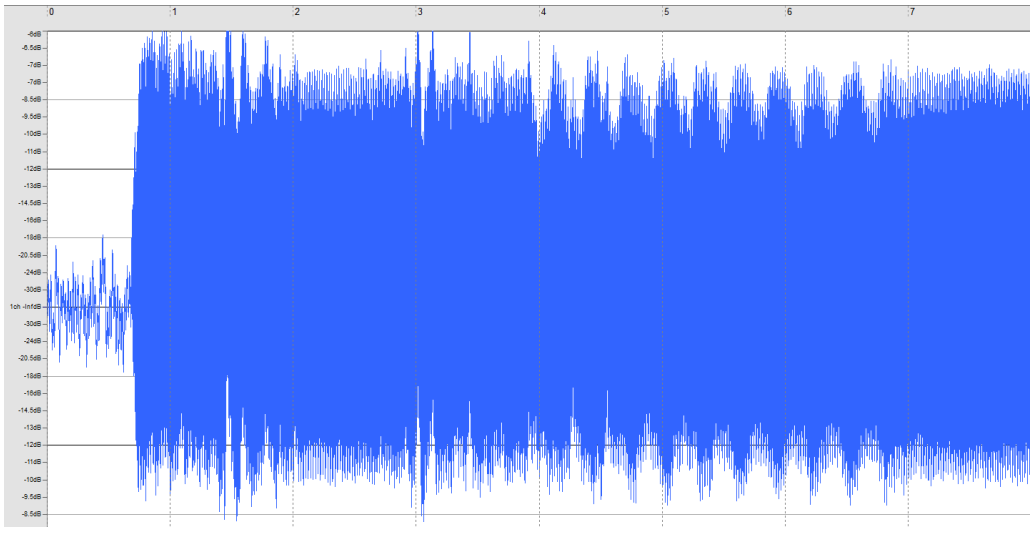


図 39. ビブラート実験(振幅 20，周期 30)

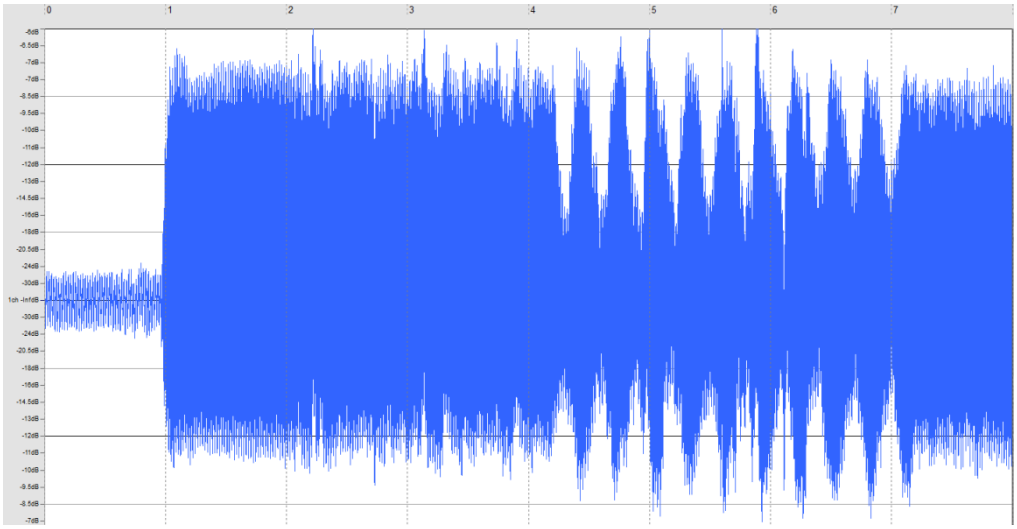


図 40. ビブラート実験(振幅 50，周期 30)

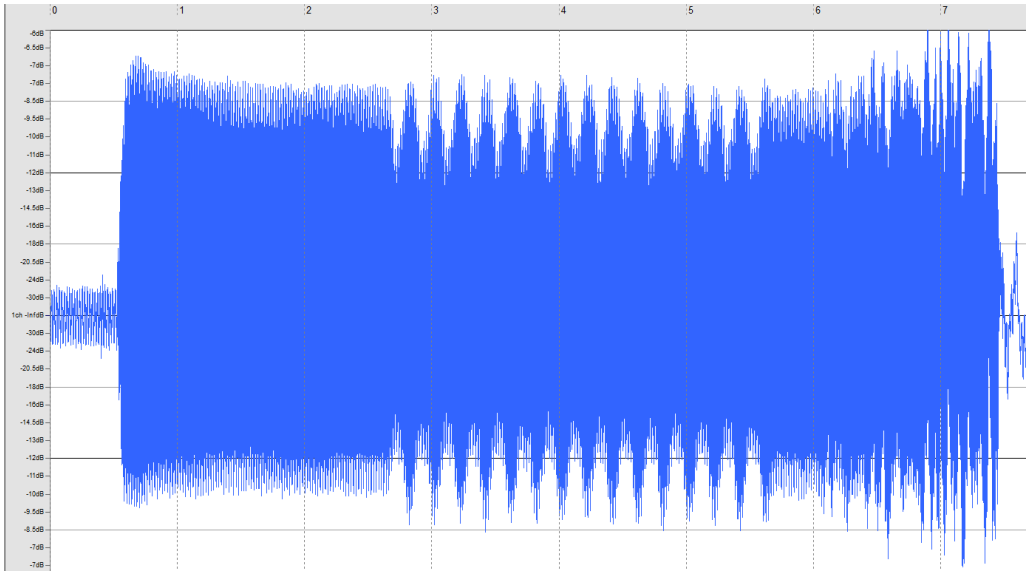


図 41. ビブラート実験(振幅 30，周期 20)

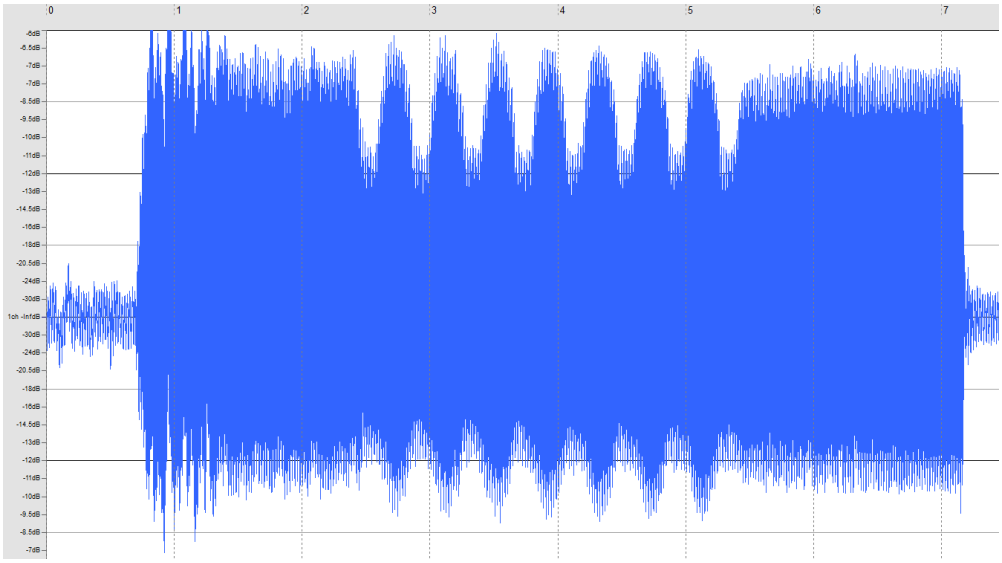


図 42. ビブラート実験(振幅 30，周期 40)

以上より実際に振幅，周期の値を変更し，音を鳴らした結果，問題なく反映された．また，ビブラートボタンも問題なく動作し，チューニングダイアログでビブラートを確認できるようになった．演奏中にも設定された値が反映できていた．

しかし，実際のビブラートと違い，本研究でのビブラートはリード圧を変更することによるピッチの変化でビブラートを起こしている．なので，振幅の設定値が大きすぎるとピッチがズレてしまう．このことから送気圧の変化により，ビブラートを起こせることが望ましいと考えられる．

7.7　演奏中に新たなる機能追加

今までに演奏中にEscキーを押すと演奏を全停止,Tabキーを押すことでMIDI音源のみ停止などの機能があった.

新たにできるようになった音量変化，ビブラートも任意に行えるようにプログラムの書き加えを行った.

演奏中に

・Sキー：演奏中の演奏音を弱にする.
・Mキー：演奏中の演奏音を中にする.
・Lキー：演奏中の演奏音を強にする.
・Vキー：演奏中の演奏音をビブラートする.

を書き加えた.

これにより，演奏中に手動で任意に行えることで演奏中に修正を加えることができ，演奏表現をさらに増やすことができると考えられる.

7.8　新たなる演奏表現，機能の追加を行った後の演奏結果

今までの結果をもとにベロシティ，エクスプレッション，ピッチベンド，モジュレーション等のMIDIメッセージを入れた曲(SMF)をdominoにて作成し，サクソフォン自動演奏ロボットで自動演奏を行った.

結果，自動演奏中に音量変化やビブラートなどを行うことができ，曲の一番盛り上がる部分では音量を大きくすることや，音の伸びている部分ではビブラートを加えることができた．これにより基本的な演奏技法ができ，演奏表現の幅が広がったのでサクソフォンの演奏技法を取り入れる準備ができたと考えられる.

第 8 章　演奏制御システムのコンパクト化

8.1　新たな演奏制御システム製作について

持ち運びを容易にするためのコンパクト化への第一歩としてデスクトップパソコンからノートパソコンに移行する．本研究では運指機構の移行を行うことを目標として新たな演奏制御システムの製作をした．

研究で使用している運指機構の状態は 3.3, 3.3.9 よりデスクトップパソコンから出した指令を DO ボードに通して実機に送り，運指の制御を行っている．これをデスクトップパソコンの代わりにノートパソコンで行うのにボードを挿すところがなく，既存の DO ボードは使用できない．なので，代わりに USB インターフェース付き PPI ユニットタートル工業製 TUSB-PIO[10)]を使用することを考えた．また，電流の増減幅を制御するために東芝製 MP4411 電界効果トランジスタ(POWER MOS FET)[11)]を使用することを考えた．

8.2 PPI ユニット

USB インターフェース付き PPI ユニット(株)タートル工業の TUSB-PIO[10]はノートパソコンの USB 端子と機器を USB コードでつなげて使う．出力端子は 2 つあり，フラットケーブル用 34 ピンコネクタを用いる．この機器を用いることでノートパソコンと運指機構の機器間の制御とデータ転送をインターフェース化する．図 43 に PPI ユニットの外観を記す.

図 43. PPI ユニット

入出力コネクタの仕様を表 11 に記す.

表 11. PPI ユニット各ピン仕様

ピン番号	入出力コネクタ	ポート
1		
2		
3		
4		
5	PortA0	PortA
6	PortA1	
7	PortA2	
8	PortA3	
9	PortA4	
10	PortA5	
11	PortA6	
12	PortA7	
13	PortC0	PortC 下位
14	PortC1	
15	PortC2	
16	PortC3	
17	GND	
18	GND	
19	PortC4	PortC 上位
20	PortC5	
21	PortC6	
22	PortC7	
23	PortB0	PortB
24	PortB1	
25	PortB2	
26	PortB3	
27	PortB4	
28	PortB5	
29	PortB6	
30	PortB7	
31		
32		
33		
34		

運指を制御するには 23 のコネクタが必要である．PPI ユニットには 24 の入出力コネクタがあるので問題ないと考えられた．

8.3 MP4411

東芝製 MP4411 電界効果トランジスタ(POWER MOS FET)[11]は 12 端子あり，GATE 端子，DRAIN 端子，SOURCE 端子がある．MP4411 の外観を図 44 に，仕様について表 12 に記す.

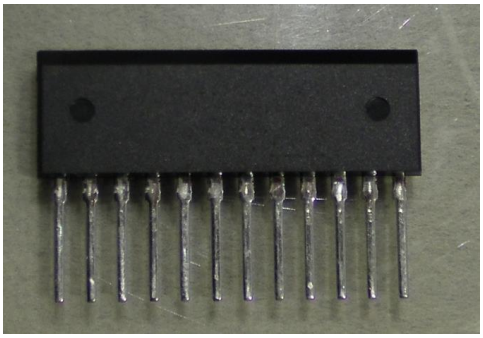


図 44. MP4411

表 12. MP4411 各端子仕様

端子(ピン)番号	種類
1,5,8,12	GATE
2,4,9,11	DRAIN
6,7	SOURCE

GATE に電圧をかけることで電子の流れに関門を設ける原理であり, DRAIN, SOURCE 間の電流を制御するトランジスタである．これにより，GATE に電圧をかけることで増減幅するということである．POWER MOS FET(Metal Oxide Semiconductor Field Effect Transistor)の仕組みについて図 45，図 46 に記す.

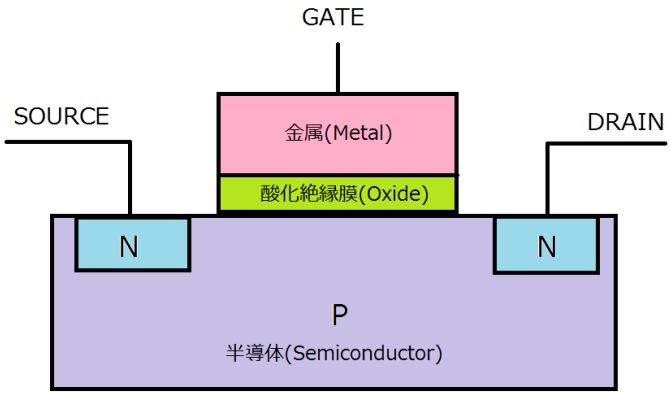


図 45. FET 動作前

図 40 は何もしていない状態で DRAIN 電極, SOURCE 電極間に電流は流れない.

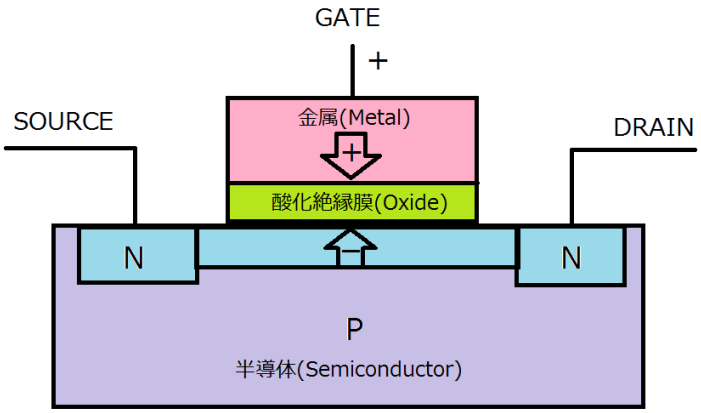


図 46. FET 動作時

図 41 は＋電圧がかかる状態で N チャネル半導体の電子が GATE に引き寄せられ、両電極間に橋ができ，電流が流れるようになる．この橋の太さは GATE に加える電圧により制御できる.

8.4　回路図

以上の PPI ユニット，MP4411 から運指を制御するための回路図を作成した．図 47 に記す．

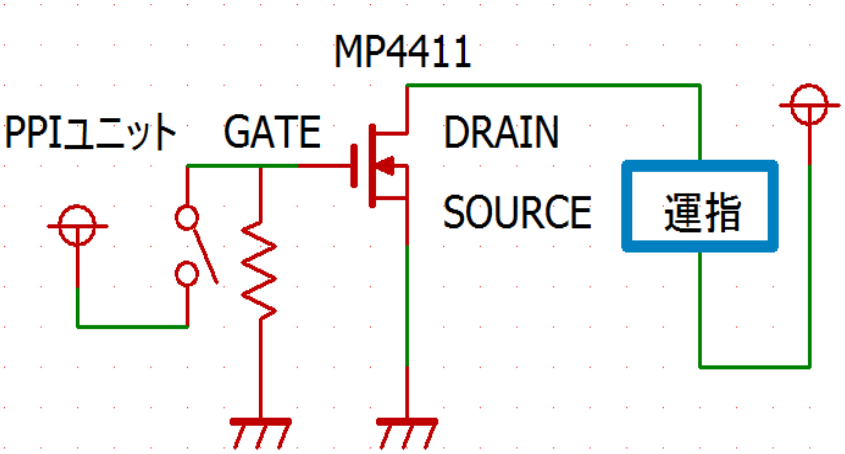


図 47. 回路図

運指を動かしたい時に PPI ユニットの対応させたピンに指令を出す．その指令が GATE に電圧をかけることになり，DRAIN と SOURCE が繋がり，電流が流れることで運指を動かすという仕組みである．

8.5　基板上の配線

回路図をもとに実際に基板上の配線をどのように張り巡らせるか，何を設置するか検討した．配線案を図 48 に記す．

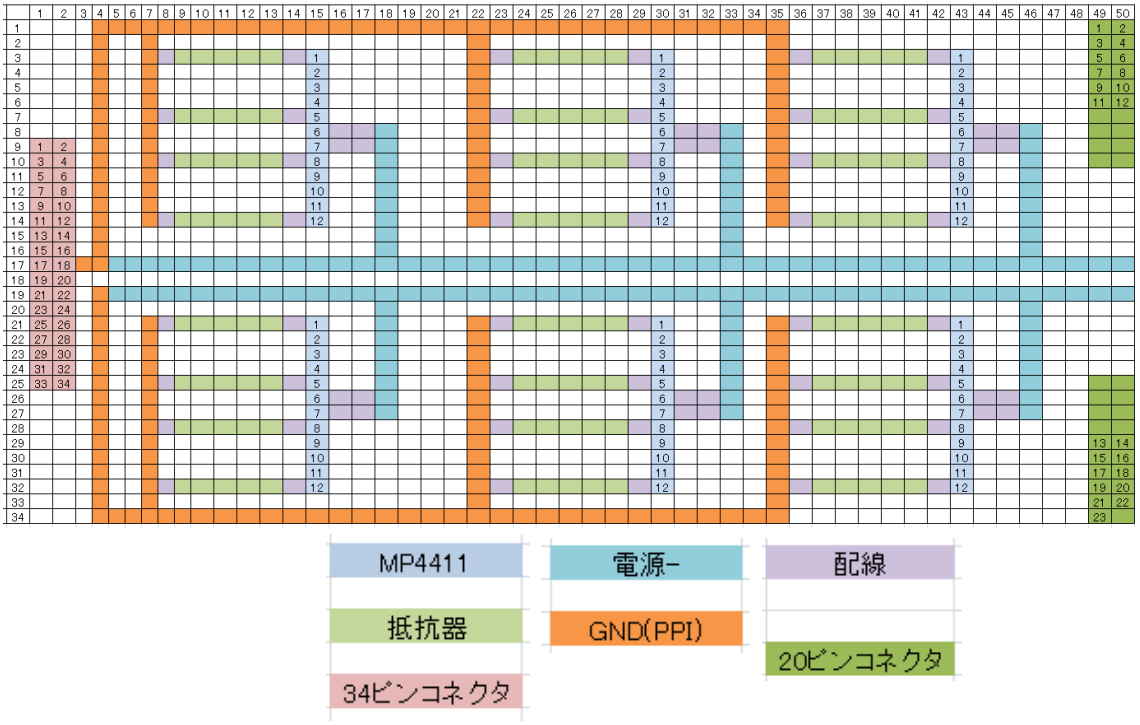


図 48. 基板上の配線案

34 ピンコネクタに PPI ユニットのピンからのフラットケーブルが接続される．抵抗器は 5KΩ を使用．実際に製作した途中のものを図 49 に記す．

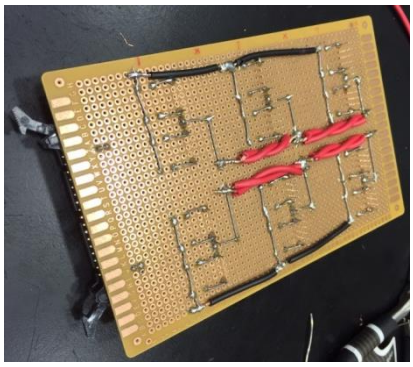


図 49. 基板上の配線製作途中

さらに基板上に設置した各コネクタやピン，MP4411 との配線を行うために全体の配線の組み合わせを検討した．図 50 に基板上の MP4411 に番号を割り振った図，表 13 に配線組み合わせを記す．

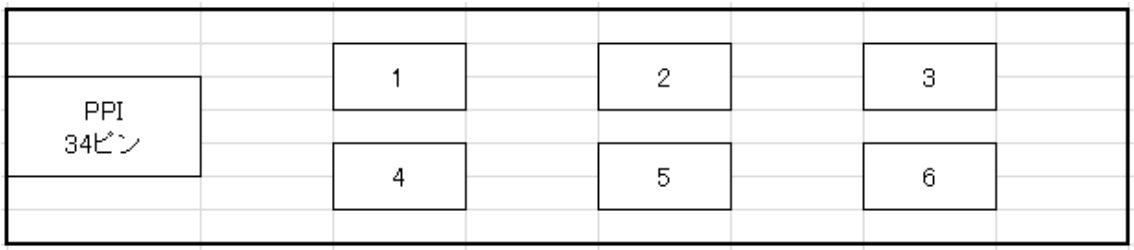


図 50. 基板上の番号振り分け

表 13. 配線組み合わせ

配線					
項目/配置	GATE/DRAIN/SOURCE	PPIピン	Port	運指	20ピンコネクタ
1	1/2/6	5	A0	1	1
	5/4/6	6	A1	2	2
	8/9/7	7	A2	3	3
	12/11/7	8	A3	4	4
2	1/2/6	9	A4	5	5
	5/4/6	10	A5	6	6
	8/9/7	11	A6	7	7
	12/11/7	12	A7	8	8
3	1/2/6	13	C0	9	9
	5/4/6	14	C1	10	10
	8/9/7	15	C2	11	11
	12/11/7	16	C3	12	12
4	1/2/6	19	C4	13	13
	5/4/6	20	C5	14	14
	8/9/7	21	C6	15	15
	12/11/7	22	C7	16	16
5	1/2/6	23	B0	17	17
	5/4/6	24	B1	18	18
	8/9/7	25	B2	19	19
	12/11/7	26	B3	20	20
6	1/2/6	27	B4	21	21
	5/4/6	28	B5	22	22
	8/9/7	29	B6	23	23
	12/11/7	30	B7	×	×

なお，運指(ソレノイドアクチュエータ)には 1 から 23 の番号が割り振りされている．

表 13 より実際に製作したものを図 51，図 52 に記す.

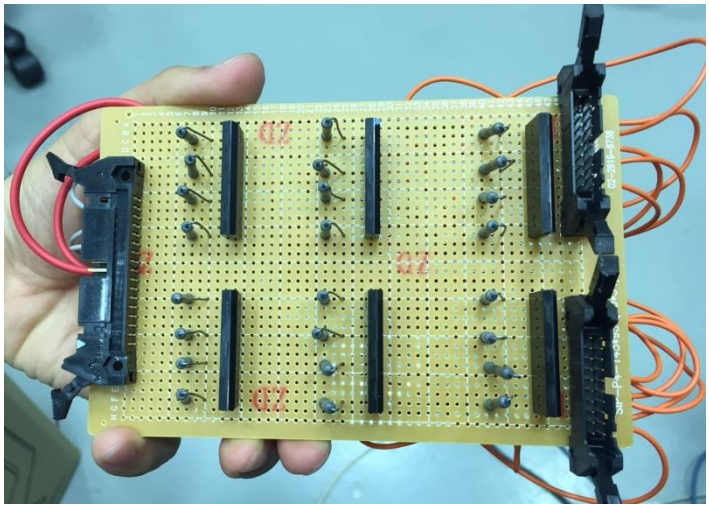


図 51. 製作した結果(表)

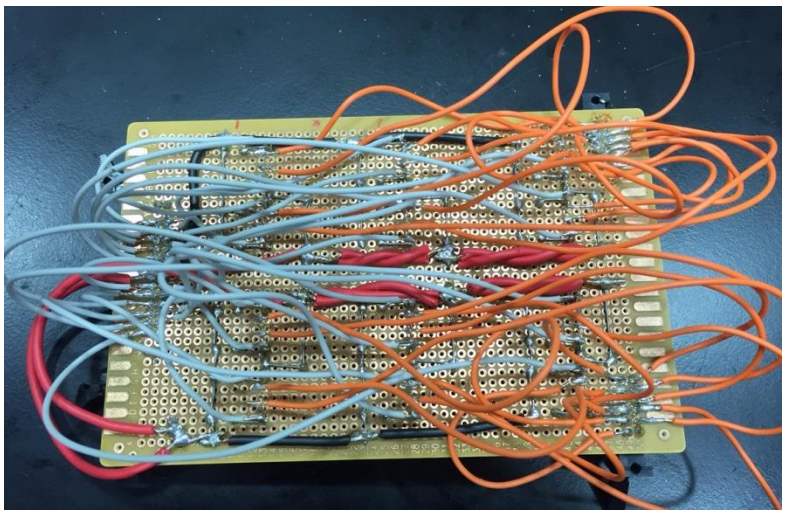


図 52. 製作した結果(裏)

8.5.1　全体の配線

8.5 より 20 ピンコネクタの先と各運指(ソレノイドアクチュエータ)を繋がなければならない．サクソフォン自動演奏ロボットの 23 個の運指の配線は 4 個の 7 ピンパネルコネクタ(それぞれ 1,2,3,4 と番号が振られている)と接続されている(図 53). 既存の演奏制御システムはその 4 個のパネルコネクタに 4 個の 7 ピンプラグを挿すことで繋ぎ，制御している．なので，新たに製作している演奏制御システムもこの 4 個の 7 ピンパネルコネクタと繋げられるようにすることで，ここの部分の切り替えで既存の演奏制御システムを今後も使用できる．

以上の考えを実現するためにロボット側の配線を理解し，その配線に合わせた演奏制御システムの設計考案，製作や新たなプラグ付きケーブルの製作などを行った．

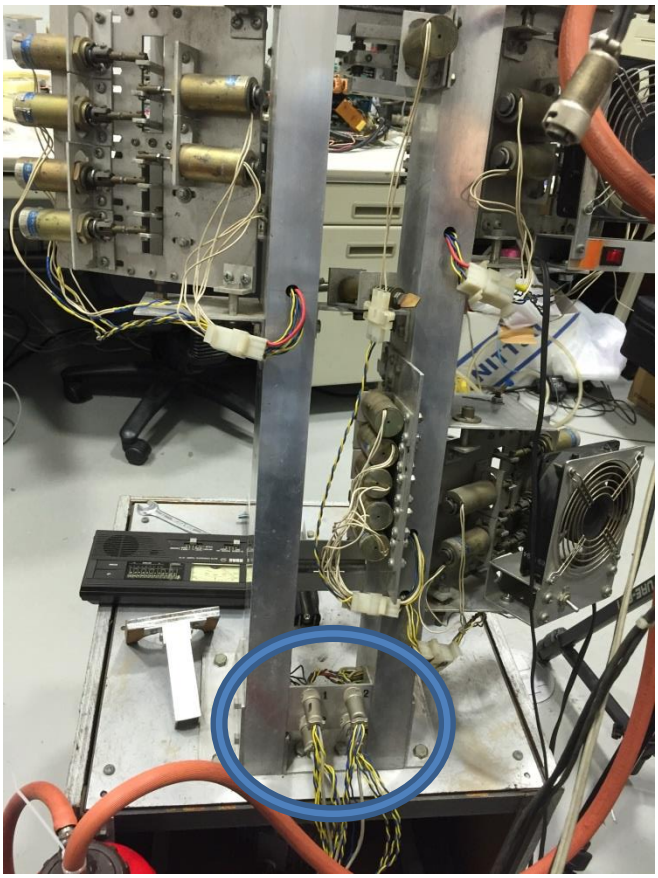


図 53. ロボット側運指パネルコネクタ間

8.5.2　自動演奏ロボット側の配線理解

新たな演奏制御システムを繋げられるようにするために 4 個のパネルコネクタから 23 個の運指への配線を調査した．調査結果を表 14 に記す．

表 14. ロボット側配線調査結果

7ピンから各運指(ソレノイドアクチュエータ)				
コネクタ番号 / ピン番号	1	2	3	4
1	1	6	16	23
2	2	7	15	22
3	3	8	14	21
4	4	9	13	20
5	5	10	12	19
6	17	×	11	18
7	1～5	6～10	11～17	18～23

1 から 23 の番号は運指に割り振られている番号である．表 14 より配線が分かったので，この配線に合わせたプラグ付きケーブルの製作を行った．なお，表 14，図 47 の回路図より各コネクタのピン番号 7 が電源の＋側に繋がっているということが分かる．

8.5.3　新たな演奏制御システムと各運指との配線考案

今までの製作や調査結果から配線の検討，演奏制御システムの設計を行った．図 54，図 55 に記す．

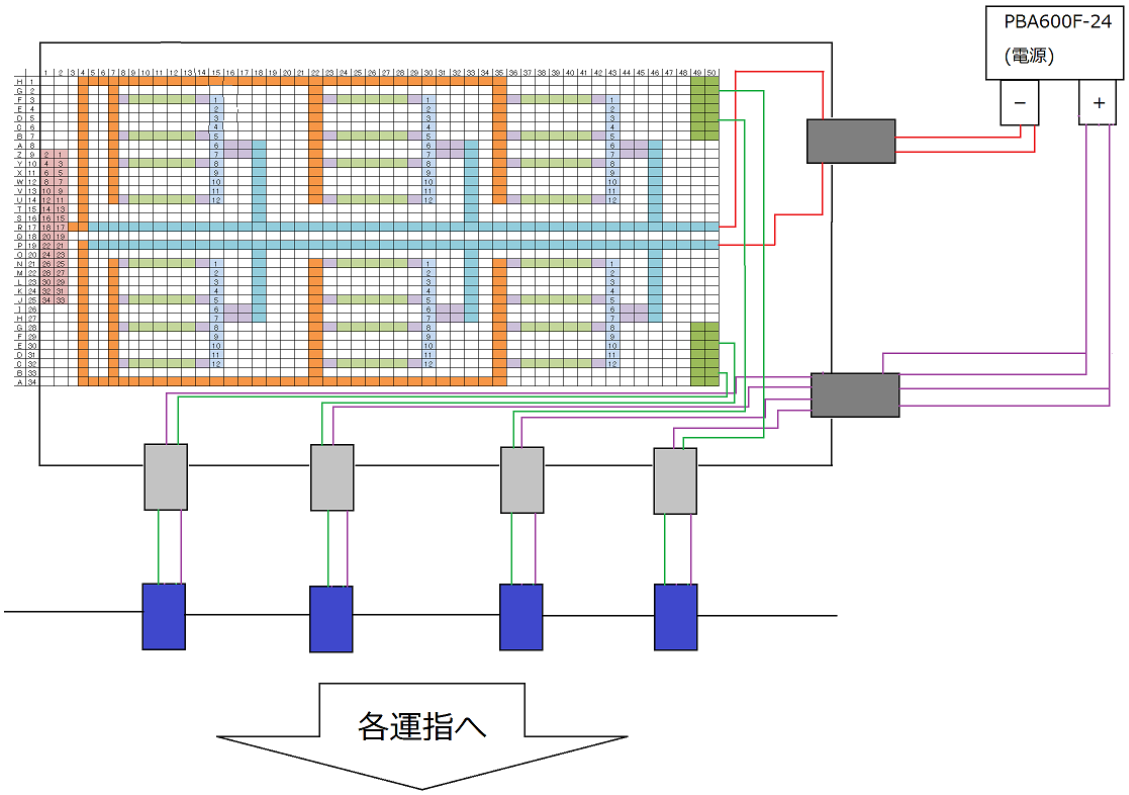


図 54. 配線考案概要

図 54 より青色の部分がロボット側の 7 ピンパネルコネクタの部分となる．薄い灰色の部分はロボット側と同じ 7 ピンパネルコネクタ，7 ピンプラグにすることを考えた．濃い灰色の部分は電源用の 4 ピンパネルコネクタ．4 ピンプラグを使用することを考えた．

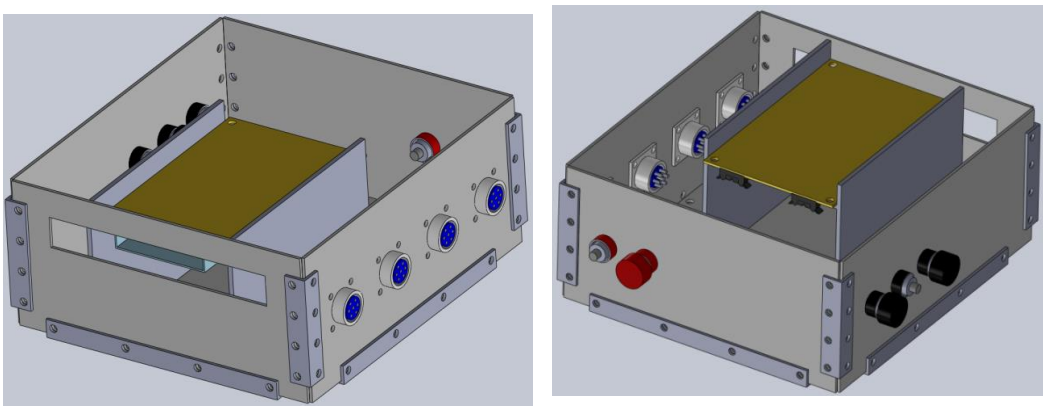


図 55. 演奏制御システム設計案

図 54 の配線考案を実現できるようなシステムの設計案を行った．なお，電源はコーセル株式会社製 PBA600F-24[12)]を使用する．

図 55 の黄色の部分が 8.5 の製作した基板であり，基板台を製作し，それに載せる方法を考えた．図 54 の薄い灰色の部分が図 55 では左の図の中が青のコネクタの部分になり，図 54 の濃い灰色の部分が図 55 では右の図の黒と赤のコネクタの部分で表わされている．

8.5.4　新たな演奏制御システムの製作結果

8.5.3 をもとに実際に製作したシステムを図 56 に記す.

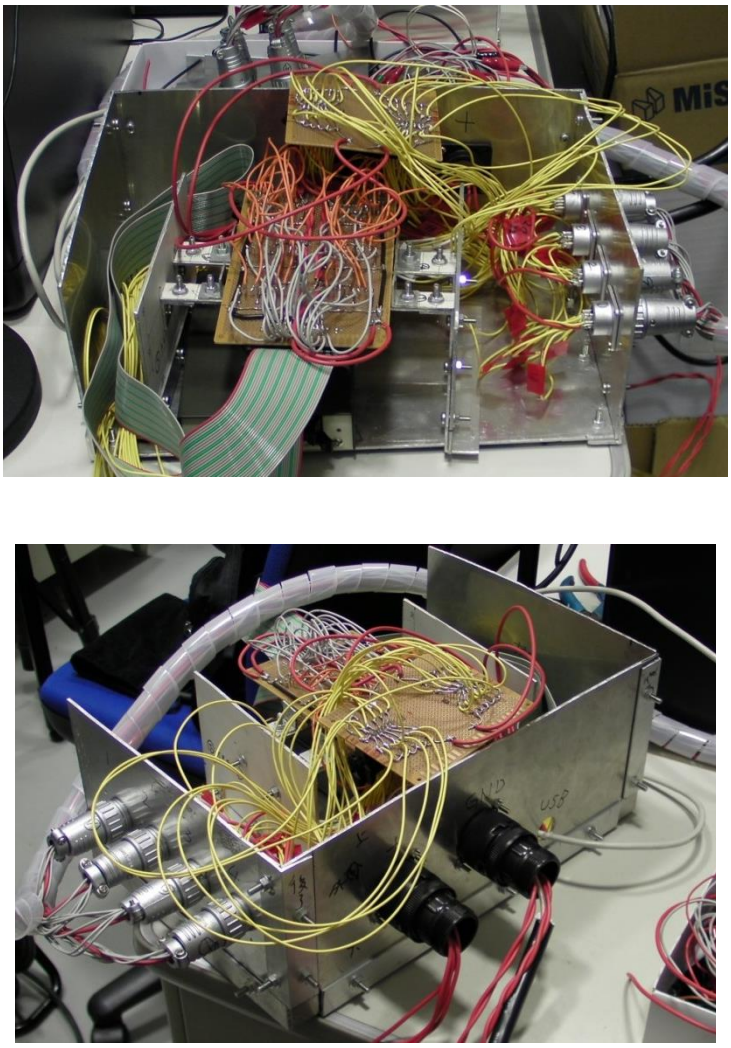


図 56. 新たな演奏制御システム完成形

製作した演奏制御システムを自動演奏ロボットに接続し，PPI ユニットに付属していたサンプルプログラムで運指を制御できるか試した結果，問題なく運指が動き，制御できたので新たな運指の演奏制御システムが完成した.

8.6　新たな演奏制御システムの制御アプリケーション作成

8.5.4 の完成した演奏制御システムを制御するためのアプリケーションを MFC(Microsoft Foundation Class)で作成した．作成したアプリケーションを記す．

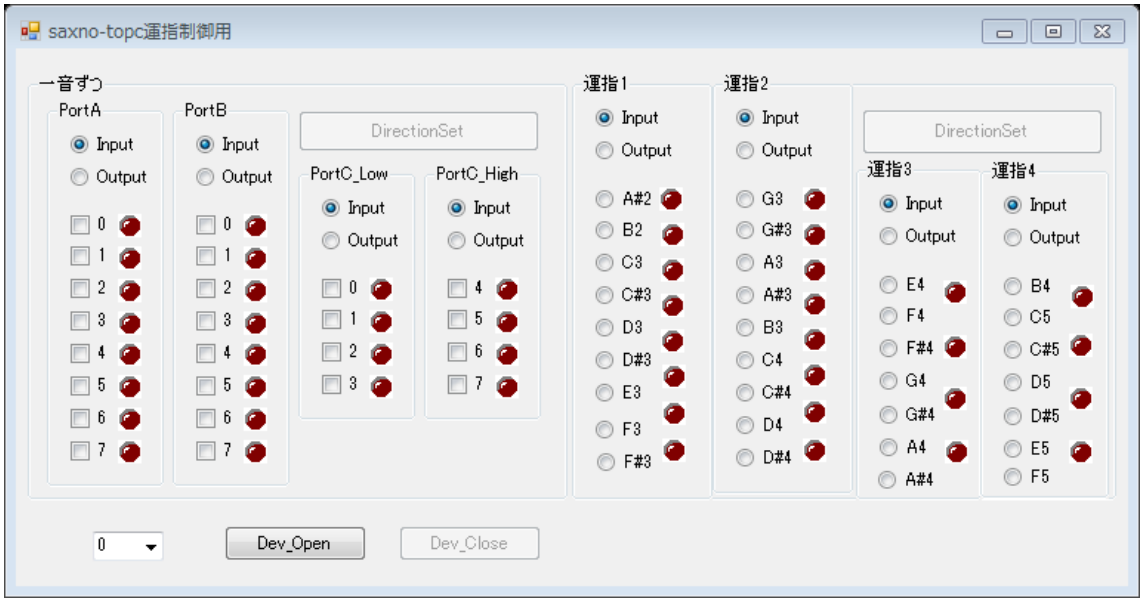


図 57. 演奏制御用アプリケーション

図 57 の左側は表 13 の配線組み合わせより Port 番号に対応した運指 1 つずつを動かせるようにした．右側はそれぞれの音階の運指を動かせるようにした．

使用方法は

①Dev_Open ボタンを押してアプリケーションと PPI ユニットを待機状態にする．この時 PPI ユニットが接続されていない場合はエラーを返し，アプリケーションを使えない．

②それぞれ Output ボタンを選択する．

③DirectionSet ボタンを押す．

④動かしたい運指のボタンを押すことでロボットの運指が動く．

⑤終わらせる時は Dev_Close ボタンを押して終了する．

なお，8.2 より PPI ユニットには 2 つの出力端子があることから図 52 の左側は PPI ユニットの上の出力端子，右側は PPI ユニットの下の出力端子で行えるようにした．

以上より実際にアプリケーションを使用した結果，問題なく運指の制御を行えた．

第 9 章　サブトーン導入に向けて

9.1　サブトーンとは

サブトーンとはリードの振動幅をそのままに、強制的に厚く、重くする奏法で，サウンドの中にエアノイズが多く含まれた柔らかで、輪郭のゆるいサウンドのことで，ジャズ演奏などにはなくてはならない演奏奏法である．

通常の演奏時は下の歯で下唇を巻き込み，リードが良く振動するように，下唇を下へ引っ張ってリードに下唇を極力触れないようにしている．サブトーンの演奏時の場合は下の歯，つまり顎の力を抜き下唇でリードを支える．今までは下唇の下から歯でリードへ直接圧力を加えていたが，その歯の圧力を弱くしそのぶん下唇でリードを支えるというイメージである．そうするとリードに触れている下唇の部分が多くなり、結果、下唇とリードが一体化すると言う方式になる．これでサブトーンを鳴らす．

9.2　人工口顎部装置の問題点

サブトーンを鳴らすには 9.1 よりリードを押さえている人工唇の移動ができないといけない．本研究で使用している人工口顎部装置では手動で人工唇の移動を行うことはできる．しかし，自動では移動できないことから演奏中には移動ができない機構となっている．なので，人工口顎部装置に新たに装置を取り付ける必要があると考えられる．

9.3　新たなる装置の取り付け

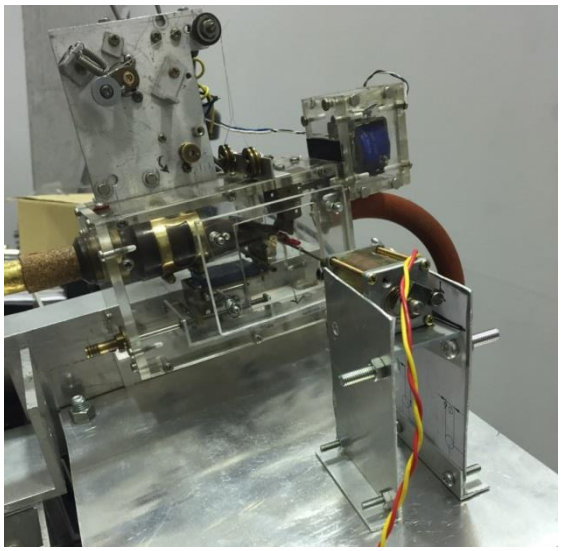
既存の人工口顎部装置に新たに装置を取り付けることでサブトーンを鳴らせるのか実験をする必要がある．なので，本研究では回転ソレノイドを使用する方法で実験を試みる．方法としてはサブトーンを行うにはリードの先端部分を押さえる必要がある．このことからアクト技研製回転ソレノイド(SL30A-12V)に取り付けた人工唇でサブトーンを使用する時だけ回転ソレノイドが動き，リードの先端を押さえるという案を考えた．以上の実験を行うために装置を取り付ける製作を行った．

・回転ソレノイド

人工口顎部装置の中に回転ソレノイドをいれるスペースがないことから回転ソレノイドには直径 3mm の円柱の棒を先に取り付けて円柱の棒の先端に人工唇を取り付けた(図 58)．この先端部分のみ人工口顎部装置の中に入れる．回転ソレノイドは 12V 駆動であるから，今回は 12V 出せる電源を使用して動かすことにした．

図 58. 回転ソレノイド

・人工口顎部装置，回転ソレノイドを載せる台
人工口顎部装置の中へ回転ソレノイドの先端部分を入れるためにアクリルパネルに直径 3mm の穴をあけたものを製作した．そして，それを人工口顎部装置に取り付けた．回転ソレノイドが人工口顎部装置の人工唇の高さと合わせるために台も製作した．以上のものを合体させた図を記す．



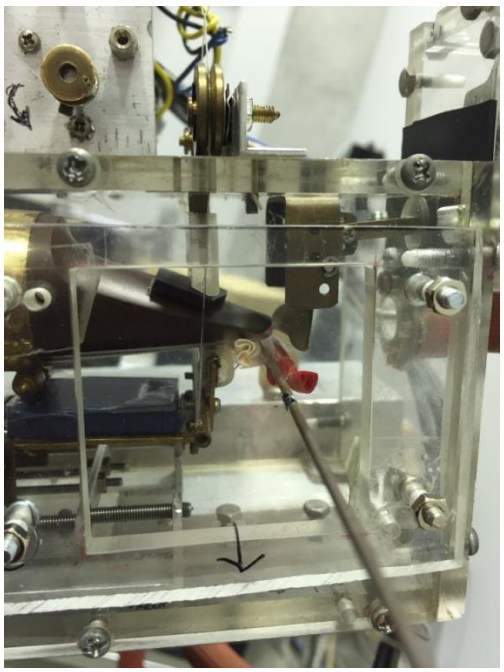


図 59. 回転ソレノイド取り付け

9.4　サブトーン実験結果

9.3 より新たなる装置を取り付けてサブトーンが鳴らせるか実験を行った．以下に結果を記す．

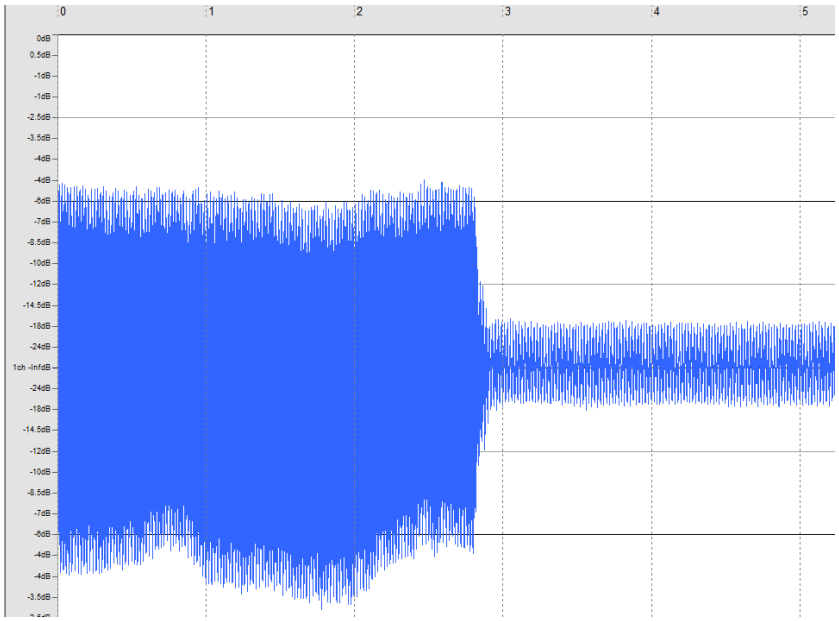


図 60．実験結果

・実験状況
縦軸：音量(デシベル)
横軸：時間(秒)
テナーサックスのベル部先端から 300mm 離したところにベル部の先端に高さを合わせたマイクを設置し，Cubase[9)]というアプリで音を取った．

図 60 より回転ソレノイドを動かした後は音量が下がった．ただ音量が下がっただけでなくサブトーンに聞こえる時もある．実験を通してサブトーンを常に出すには人工唇の選定によって音色が全然違うので様々な人工唇で実験，調節を行い，押さえる位置などもしっかり調整できる機構にする必要があると考えられた．

第 10 章　結論

本研究を行ったことでサクソフォン自動演奏ロボットの演奏表現に音に抑揚をつけることやビブラートをつけることができるようになった．これにより今までに鳴らすことのできなかった音で演奏ができるようになり，演奏表現の幅が広がったと考えられる．しかし，7.6.3 で述べたようにビブラートは送気圧で起こせなくてはならない．本研究により基本の演奏技法を取り入れることができたので，今後はアタックやオーバートーンやグリッサンドなどといったサクソフォンの演奏になくてはならに演奏技法を行えるようにしていかなければならない．これらの演奏技法を自動演奏ロボットで行えるようになることで人間との演奏にも支障なくできると考えられる．しかし，サブトーンの実験からも現在のシステムだと新たに演奏技法を取り入れるには限界がある．現在の人工口顎部装置は手動でないとリードを人工唇で押させる位置を変更できない．なので，自動的に演奏中にリードを押さえる位置を変更できるようになることや，新たに外部から装置を取り付けて対処するかしていかないと既存のリード圧の設定だけでは限度があると考えられる．また，既存の送気圧では出せる音の限界があることやビブラートを起こせるような機構がない．空気流そのものを一気に減らしたり増やしたりする機構を取り付けたりする必要があるのではないかと考える．そうすることでサブトーンなどの演奏技法もやりやすくなる．音量変化も 3 通りだけでなくそれ以上の音量変化を行えると考えられる．今後は新たなる装置，機構の取り付けが大事になっていくのではないかと考える．

製作した新たな演奏制御システムは問題なく制御でき，制御アプリケーションの作成もできた．しかし，本研究では運指の制御システムしか製作できなかった．なので，今後リード圧制御など他の部分の新たな演奏制御システムを製作していく必要がある．また，本研究で作成した制御アプリケーションでは既存の自動演奏制御プログラムなどに比べて機能に乏しい．今後作成したアプリケーションには既存の自動演奏制御プログラムのような機能を加えていく必要がある．以上より今後さらに研究を進めることで，デスクトップパソコンではなく，ノートパソコンで自動演奏ロボットの制御を行っていくことができ，持ち運びの利便性の向上が期待できる．さらに，新たな演奏制御システムでは USB ポートのついているノートパソコンであればすぐに使用ができる．このことから汎用性の向上も狙えると考える．

謝辞

本研究を行うにあたり，ご指導して頂きました法政大学理工学部機械工学科，高島俊教授に深く感謝の意を表すとともに，厚くお礼申し上げます.

また，本研究に協力して頂いた方々に深く感謝いたします.

2016 年 2 月 20 日

森　琢也

発表論文

森琢也，若林直哉，高島俊，“サクソフォン自動演奏ロボットの制御　- 他の奏者と協調演奏を行うための MIDI ファイル(SMF)を用いた演奏制御 - ”, ロボティクス・メカトロニクス講演会 2014，ROBOMECH2014 in TOYAMA

森琢也，高島俊，“サクソフォン自動演奏ロボットの開発　- 演奏制御システムのコンパクト化と高度な演奏表現 - ” ロボティクス・メカトロニクス講演会 2015，ROBOMECH2015 in KYOTO

参考文献

1) 竹内慧“自動演奏ロボットの高度な演奏制御-MIDI 入力装置による制御と自律合奏のためのシステム-”(2012),法政大学院工学研究科機械工学専攻修士論文

2) 宮脇毅“サクソフォン自動演奏ロボットの制御-スタンダード MIDI ファイルを用いた演奏制御-”(2003), 修士論文, 法政大学大学院工学研究科機械工学専攻

3) 細野佳明　本間義晃“サクソフォン自動演奏ロボットの制御-MIDI によるロボットの演奏制御-”, (2002), 法政大学工学部機械工学科卒業論文

4) 中島安貴彦“MIDI バイブル Ⅰ　MIDI 1.0 規格 基礎編”(1997),
株式会社リットーミュージック

5) 高橋信之“コンプリート MIDI ブック”(2005),
株式会社リットーミュージック

6) 菅原朋子“速習 C 言語入門”(2006),
株式会社毎日コミュニケーションズ

7) Domino
(http://takabosoft.com/domino)

8) MuseScore
(http://musescore.org/ja)

9) Cubase
(http://japan.steinberg.net/jp/products/cubase/start.html)

10) USBインターフェース付きPPIユニット 株式会社タートル工業
株式会社タートル工業TUSB-PIO 説明書

11) MP4411　TOSHIBA　POWER MOS FET 株式会社東芝
(http://www.ic72.com/pdf_file/2pdf/ea09380.pdf)

12) PBA600F-24 コーセル株式会社
(https://www.cosel.co.jp/product/powersupply/PBA/PBA600F/PBA600F-24/)

付録１　プログラム関数一覧

sax.cpp内で用いられているプログラム関数について記載する．

1-1　SMF の処理

・MIDIfileopen

説明	ファイルを開く際のメイン関数であり，ファイルのバッファへの読み込み，含まれる楽器や分解能の検出，調や拍子を検出する関数の呼び出しを行う．
宣言	void MIDIfileopen(HWND, HWND, HWND, HWND**)**
パラメータ	第 1 引数 … メインウィンドウのハンドル 第 2 引数 … コンボボックスのハンドル 第 3 引数 … 再生ボタンのハンドル 第 4 引数 … コメント表示用スタティックテキストのハンドル
戻り値	関数が成功 → 0 関数が失敗 → エラーコード（0 以外の値）

・checktrack

説明	各トラックの最初に呼び出され，トラック長を検出，変数に保存する．この変数は演奏用 SMF 作成に用いられる．
宣言	int checktrack(unsigned char＊)
パラメータ	第 1 引数 … SMF 用バッファのアドレス
戻り値	トラックの開始　　→トラック長 トラックの開始以外 → 0

・searchmeta

説明	メタ・イベントの検出を行うための関数で，テンポ，調，拍子の検出を行う．誤った検出を防止するため，検出を行う位置をそれぞれ対応するバイト数飛ばしている．
宣言	bool searchmeta(unsigned char＊, int＊)
パラメータ	第 1 引数 … SMF 用バッファのアドレス 第 2 引数 … バッファ検索用繰り返し変数のアドレス
戻り値	メタ・イベント　　→ TRUE メタ・イベント以外 → FALSE

・**checkevent**

説明	SMF の各信号の初め，かつ信号の種類が変わる，すなわちランニング・ステータスの途中ではない時に呼び出し，その信号が何の信号かを特定するための関数である.
宣言	unsigned char checkevent(unsigned char＊, unsigned char)
パラメータ	第 1 引数 … SMF 用バッファのアドレス 第 2 引数 … MIDI イベント判別用変数
戻り値	イベントを判別するための変数

・**checkonoff**

説明	SMF 内のノート・オン，ノート・オフの検出を行う．この際，音が重複する場合は，後から始まる音は検出しない．また，タンギングの判別も行う．検出した値は，テキストデータに保存すると同時に，演奏用データとして配列に保存する．但し，ノート・オフの音階は－ 1 としている.
宣言	unsigned char checkonoff(unsigned char＊, unsigned char＊, HANDLE)
パラメータ	第 1 引数 … SMF 用バッファのアドレス 第 2 引数 … ランニング・ステータス終了判別用変数 第 3 引数 … トラックデータ出力用テキストのハンドル
戻り値	ランニング・ステータス終了 →　0 新たなイベントが始まる　→ ‐1

・**delta**

説明	デルタタイムごと，すなわち各イベント間に毎回呼び出され，データから実時間(ミリ秒)に変換する. デルタタイムは可変長表示であるため理論上は何バイトでも続けることが出来るが，一般的な分解能 480，テンポ 120 の場合において 2 バイトでおよそ 16 秒，3 バイトで 30 分以上まで対応可能なため，この関数では 3 バイトまでとしている.
宣言	void delta(unsigned char＊, int＊, HANDLE)
パラメータ	第 1 引数 … SMF 用バッファのアドレス 第 2 引数 … デルタタイムのバイト数判別用変数 第 3 引数 … トラックデータ出力用テキストのハンドル
戻り値	n‐1（n = デルタタイムとして使用したバイト数）

・**checkcontrol**

説明	SMF 内のコントロール･チェンジ･メッセージの検出を行う．任意のタンギング，ビブラートの種類，モジュレーション･ディプス，ビブラート･コントローラを判別し，テキストデータに保存すると同時に，演奏用データとして配列に保存する．
宣言	unsigned char checkcontrol (unsigned char＊, unsigned char＊, HANDLE)
パラメータ	第 1 引数 … SMF 用バッファのアドレス 第 2 引数 … ランニング・ステータス終了判別用変数 第 3 引数 … トラックデータ出力用テキストのハンドル
戻り値	ランニング・ステータス終了 →　0 新たなイベントが始まる　→ －1

・**checkpitchbend**

説明	SMF 内のピッチ･ベンドの検出を行う. LSB と MSB のデータ並びを変換，ピッチ･ベンドを判別し，テキストデータに保存すると同時に，演奏用データとして配列に保存する．
宣言	unsigned char checkpitchbend (unsigned char＊, unsigned char＊, HANDLE)
パラメータ	第 1 引数 … SMF 用バッファのアドレス 第 2 引数 … ランニング・ステータス終了判別用変数 第 3 引数 … トラックデータ出力用テキストのハンドル
戻り値	ランニング・ステータス終了 →　0 新たなイベントが始まる　→ －1

・**MIDIBreak**

説明	ファイルを開いたときに呼び出される関数である．まずバッファの初期化を行い，ヘッダを読み込む．その際ヘッダに含まれる総トラック数を 1 つ減らしておく．続いてトラックごとに多次元配列へ書き込む．すなわち多次元配列の 1 つ目の変数が 0 にヘッダを，1 に 1 つ目のトラックを，2 に 2 つ目のトラックを書き込んでいる．
宣言	void MIDIBreak(int)
パラメータ	第 1 引数 … SMF のハンドル

・**MakeMIDIFile**

説明	プログラム実行時，コンボボックスで選択された楽器のトラックが削除された SMF を作成する関数である．ここで作られる SMF は通常 test.mid であるが，そのファイルが既に存在する場合 test1.mid，test2.mid，…，と後に付属する数字を増やしたファイル名で作成する. また, これらの SMF はアプリケーション終了時にすべて削除される．そして，この関数から演奏用データを作成する関数を呼び出す.
宣言	void MakeMIDIFile(HWND)
パラメータ	第 1 引数 … コンボボックスのハンドル

・**ReadPlayTrack**

説明	コンボボックスで選択されたトラック，すなわちロボットに演奏させるデータを作成する関数である．ここで検出を行う際，重複する音を無視し，サックスで表現不可能なデータの場合，間のデルタタイムのみを検出する.
宣言	void ReadPlayTrack(int)
パラメータ	第 1 引数 … ロボットに演奏を行わせるトラックの番号

1-2 MCI の処理

・OnActionButton

説明	実行ボタンを押した時の処理を行う．この処理を関数内で行うことによってプロシージャ内を整理し，処理の追加等も行いやすくなっている．まず実行ボタンが複数回押された時を考慮し，MCI デバイスを閉じるという関数を呼び出す．次に MCI デバイスを開く関数を呼び出し，以後のデバイスへのアクセスを可能としている．
宣言	void OnActionButton(HWND)
パラメータ	第 1 引数 … メインウィンドウのハンドル

・OnPlayButtonDown　OnStopButtonDown

説明	OnActionButton と同様，再生ボタン，ロボットのみ演奏ボタンを押した時の処理を行う．実行ボタンが押された時に作成された SMF の再生，停止を行う関数と，演奏用データを実機に送信する関数を呼び出す．
宣言	void OnPlayButtonDown(HWND)　void OnStopButtonDown(HWND)
パラメータ	第 1 引数 … メインウィンドウのハンドル

・InitMciDev

説明	デバイスの種類を MIDI に, ファイル名は実行ボタンを押した際に作成された SMF 名に指定し，MCI デバイスを開き ID を取得する．
宣言	BOOL InitMciDev()
戻り値	オープン成功 → TRUE オープン失敗 → FALSE

・TermMciDev

説明	MCI デバイスが開いている状態で呼び出された場合のみ，InitMciDev()で取得した ID の MCI デバイスを閉じる．
宣言	void TermMciDev()

・PlayMciDev

説明	InitMciDev で指定したファイルの再生を行う．また，再生中に何らかのエラーが発生した場合デバイスを閉じる．
宣言	BOOL PlayMciDev()
戻り値	正常の場合→ TRUE エラー発生→ FALSE

1-3　自動演奏ロボットとのインターフェイス

・SendPerformanceData

説明	ReadPlayTrack で作成された演奏用データをロボットに送信する関数である．演奏用データから音階を調べ，それに対応したチューニングにより決定された電圧を各チャンネルに送信する．そして，指令値を送信するたびに時間を決定する関数を呼び出す.
宣言	void SendPerformanceData()

・wait

説明	演奏の時間を決定する関数であり，アプリケーションが実行してからの時間をミリ秒単位で取得しループに入る．ループ内で時間を取得し続け，時間の差が目標時間に達した段階でループを抜けるということを行っている．このループ内で，Esc キーが押された場合，自動演奏を中止する．また，時間の取得は 1 [m sec]単位で行っているので，SMF の再生との誤差が生じることは極めて少ない.
宣言	void wait(int, bool＊)
パラメータ	第 1 引数 … 待ち時間をミリ秒単位で指定 第 2 引数 … Esc キーの演奏強制終了確認用変数

・cyutime

説明	チューニング画面でビブラートボタンを押した際に使用する関数である．ビブラート時間の間はリード圧に正弦波の値を足すという処理がされている．取得は 1 [m sec]単位で行っている.
宣言	void cyutime(double, int)
パラメータ	第 1 引数 … ビブラート時間(ミリ秒単位) 第 2 引数 … 判別用

1-4　ウインドウ

・**WinMain**

説明	プログラム実行時，最初に実行される関数であり，コンソールアプリケーションの main にあたる関数である．ウインドウの作成とメッセージの取得，送信を行う．
宣言	int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)

・**WinProc**

説明	WinMain 関数からメインウインドウ上のメッセージが送られてきて，その処理を行うさまざまな関数を呼び出すプロシージャである．ダイアログボックスの作成も行う．
宣言	LRESULT CALLBACK WinProc(HWND, UINT, WPARAM, LPARAM)

・**DlgProc**

説明	曲情報ダイアログ上のメッセージの処理を行うプロシージャである．分解能，テンポ，調，拍子の 4 つの情報を表示する．
宣言	BOOL CALLBACK DlgProc(HWND, UINT, WPARAM, LPARAM)

・**TuningDlgProc**

説明	チューニングダイアログ上のメッセージの処理を行うプロシージャである．値を変更した際，表示を変えると同時に，ロボットにデータの送信を行う．また，OK ボタンを押した際には，その値をテキストファイルとして保存を行う．
宣言	BOOL CALLBACK TuningDlgProc(HWND, UINT, WPARAM, LPARAM)

・**OnDestroy**

説明	アプリケーション終了時に呼び出される関数であり，DA，DO ボードのクローズと MCI デバイスのクローズ，そして演奏用として作成された SMF の削除を行う．
宣言	void OnDestroy(HWND)
パラメータ	第 1 引数 … メインウィンドウのハンドル

1-5　ライブラリ

① Fbida.lib（DA ボード）

・DaOpen

説明	指定されたデバイス名のボードをオープンし，以後ボードへのアクセスを行えるようにする.
宣言	HANDLE WINAPI DaOpen(LPCTSTR)
パラメータ	第 1 引数 … 任意のデバイス名
戻り値	関数が成功　→　有効なデバイスハンドル 関数が失敗　→　INVALID_HANDLE_VALUE（＝－1）

・DaClose

説明	ボードをクローズし，ボードアクセスのために使用されていた各種リソースを開放，以後ボードへのアクセスを禁止する.
宣言	int WINAPI DaClose(HANDLE)
パラメータ	第 1 引数 … 有効なデバイスハンドル
戻り値	関数が成功 →0 関数が失敗 → エラーコード（0 以外の値）

・DaOutputDA

説明	DASAMPLCHREG 構造体に格納された番号を指定したチャンネルから 1 件のアナログ出力を行う.
宣言	int WINAPI DaOutputDA(HANDLE, ULONG, PDASMPLCHREQ, LPVOID)
パラメータ	第 1 引数 … 有効なデバイスハンドル 第 2 引数 … チャンネル数 第 3 引数 … チャンネル番号と出力レンジのポインタ 第 4 引数 … 出力するデータのポインタ
戻り値	関数が成功 →0 関数が失敗 → エラーコード（0 以外の値）

② Fbidio.lib（DO ボード）

・**DioOpen**

説明	指定されたデバイス名のボードをオープンし，以後ボードへのアクセスを行えるようにする.
宣言	HANDLE WINAPI DioOpen(LPCTSTR,　DWORD)
パラメータ	第 1 引数 … 任意のデバイス名 第 2 引数 … オープン時のフラグ 0　＝重複オープン禁止 FBIDIO_FLAG_SHARE　＝重複オープン可
戻り値	関数が成功 → 有効なデバイスハンドル 関数が失敗 → INVALID_HANDLE_VALUE（＝－1）

・**DioClose**

説明	ボードをクローズし，ボードアクセスのために使用されていた各種リソースを開放，以後ボードへのアクセスを禁止する.
宣言	int WINAPI DioClose(HANDLE)
パラメータ	第 1 引数 … 有効なデバイスハンドル
戻り値	関数が成功 → 0 関数が失敗 → エラーコード（0 以外の値）

・**DioOutputPoint**

説明	指定した開始番号接点から 1 接点ずつ int 型変数に格納されたデータで接点を制御する.
宣言	int WINAPI DioOutputPoint(HANDLE, PINT, DWORD, DWORD)
パラメータ	第 1 引数 … 有効なデバイスハンドル 第 2 引数 … 出力するデータのバッファへのポインタ 第 3 引数 … 出力接点の開始番号 第 4 引数 … 出力接点数
戻り値	関数が成功 → 0 関数が失敗 → エラーコード（0 以外の値）

付録 2　sax. cpp

sax.cppについて記載する.

```
/*******************************************************
*　ＭＩＤＩを用いたＳＡＸ自動演奏制御プログラム　　　*
*******************************************************/

#include <windows.h>
#include "FbiDio.h"
#include "FbiDa.h"
#include "userlib.h"

#define ID_ACTION 1000
#define ID_PLAY 1001
#define ID_STOP 1002
#define ID_COMBO 1003
#define ID_MCISTOP 1004//MCIのみSTOP用
#define CHOOSE "選択してください"
#define SCont 10　　//音階の高さを調整
#define tongingtime 10　//ﾀﾝｷﾞﾝｸﾞする時間（ｿﾚﾉｲﾄﾞのｵﾝからｵﾌまでの時間）
#define BS_PUSHBUTTON (0)
#define BS_PUSHBUTTON1 (0)
#define BS_PUSHBUTTON2 (0)
#define PI 3.1415　　　　　　　　　　//円周率

int TrackSize[21];
double bunkai2,tempo1,tempodef,tempobuf;　//tempodef(変更前の元テンポ)追加
int tkcount=1;
int TotalTime=0;
int TrueTotalTime;　//wait抜ける毎にカウンタの値を追加していく
int TmpTime=0;　//リセットせずに合計時間を保存する変数
double taptime[6];　//テンポ変更用　カウンタの差を3回保存、[0]には平均値を保存 2枠分は補助用
double tap[6];　　//キー押したタイミングでのカウンタの値を4回保存 2枠分は補助用
int tapCount=0;　//タップ回数
```

```
int tongingflag,vibflag;//ﾀﾝｷﾞﾝｸﾞのON・OFFの判定値を入れておく変数
//ﾋﾞﾌﾞﾗｰﾄの種類の判別値を入れておく変数
int nowonkai;//その時のﾉｰﾄｵﾝの音階を入れておく変数(同音が続くかの判別に使用)
int nowi;       //その時のﾉｰﾄｵﾝが何番目か入れておく変数(同音が続くかの判別に使用)
int nowmodul,nowrate,nowdepth;//その時のﾋﾞﾌﾞﾗｰﾄに関する値を入れておく変数
int nowmodul2=127;

int LipP[48][3], BlowP[48][3];
int dina; //チューニング時の強弱の判定　0:強　1:中　2:弱
int dina2;//演奏中にボタンでチューニングデータ替える用
int expre[4];//エクスプレッションにより音量変化用
int exnoto;//エクスプレッションとノートオンとの関係判別用
int vel[8];//ベロシティ判別用
int nowexp;//現在のエクスプレッションの値保存用
int bib;//ビブラート判別用
int bib2;//ビブラートボタン用
int bib3=0;//ビブラートチューニング中用
int cyulip=0;//ビブラート時のリード圧用
int cyulip2=0;
double sec_pc;//ビブラート用
double clipVBase[3];//ビブラート用
double dlipVBase[3];//ビブラート用
int clip;
double cyulipVBase;
double shinpuku[51][3];
double syuki[51][3];
double shinpuku2[3];
double syuki2[3];

int tuningflag = 0;

int PerformanceData[10000][10];   //配列の内容は、
// [n][0] ﾉｰﾄｵﾌ＝-1, ﾉｰﾄｵﾝ＝音階
// [n][1] 次のｽﾃｰﾀｽまでのﾃﾞﾙﾀﾀｲﾑ(単位μ秒)
// [n][2] ﾀﾝｷﾞﾝｸﾞ          （Bn 6e） のｵﾝorｵﾌ
// [n][3] ﾓｼﾞｭﾚｰｼｮﾝ        （Bn 01） のｵﾝorｵﾌ
```

```
// [n][4] ﾋﾞﾌﾞﾗｰﾄ･ﾚｲﾄ　（Bn 4c）の値
// [n][5] ﾋﾞﾌﾞﾗｰﾄ･ﾃﾞﾌﾟｽ（Bn 4d）の値
// [n][6] ﾋﾞﾌﾞﾗｰﾄ･ﾃﾞｨﾚｲ（Bn 4e）の値
// [n][7] ﾋﾟｯﾁﾍﾞﾝﾄﾞ（En）の値
// [n][8] ベロシティ判別用
// [n][9] エクスプレッション判別用

int Performance_i=0;//配列PerformanceDataの行数(上のｺﾒﾝﾄ文の[n])

int MakeMIDICount=0;      //自作のMIDIファイルを消すときに使う
double LipPressure[48][3]; //唇(電圧)
double BlowPressure[48][3]; //送気圧(電圧)
char chou_chou[30]="ファイルが選択されていません";
char hyousi_hyousi[11];
char DeviceName[100];
unsigned char MIDIPiece[22][26000]; //各トラックごとに読み込む。(0はﾍｯﾀﾞﾁｬﾝｸのみ)
FILE *fp;
HINSTANCE hInstance;
HINSTANCE hLibFBIDA;
HWND hStatic,hDynamic;
HWND hWnd = NULL;
HANDLE hDeviceDio,hDeviceDA;

bool remake=TRUE; //実機に演奏データを送った後にFALSEになる
DASMPLCHREQ ch;
int test[32]={1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int NoteOffFinger[32]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int FingerPosition[48][32]={
{0,1,1,1,1,1,1,1,0,1,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,1,1,1,1,1,1,1,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,1,1,1,1,1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,1,1,1,1,1,1,1,0,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
```

{0,1,1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,1,1,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,1,1,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0},

{0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,1,1,1,1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

```
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
};

static HDC hdc; //確認用
char str[10];      //確認用

// ウィンドウプロシージャ関数のプロトタイプ（メッセージの処理）
LRESULT CALLBACK WinProc(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam);

/*****************************************************************
* メイン関数
*****************************************************************/
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int
nCmdShow){
MSG msg;
WNDCLASSEX wcex;

wcex.cbSize             = sizeof(wcex);
wcex.style              = CS_HREDRAW | CS_VREDRAW | CS_DBLCLKS ;
wcex.lpfnWndProc        = WinProc;
wcex.cbClsExtra         = 0;
wcex.cbWndExtra         = 0;
wcex.hInstance          = hInstance;
wcex.hIcon              = LoadIcon(NULL, "IDI_ICON1");
wcex.hCursor            = LoadCursor(NULL, IDC_ARROW);
wcex.hbrBackground      = (HBRUSH)GetStockObject(WHITE_BRUSH);
wcex.lpszMenuName       = MAKEINTRESOURCE(IDR_MENU1);
wcex.lpszClassName      = APP_NAME;
wcex.hIconSm            = NULL;

if(!RegisterClassEx(&wcex)){
return 0;
}
```

```
hWnd = CreateWindowEx( WS_EX_APPWINDOW,
APP_NAME,
APP_TITLE ,
WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU | WS_MINIMIZEBOX,
CW_USEDEFAULT,
CW_USEDEFAULT,
360,//画面サイズ変更横
410, //画面サイズ変更縦
NULL,
NULL,
hInstance,
NULL );

ShowWindow(hWnd, nCmdShow);
UpdateWindow(hWnd);

while(GetMessage(&msg, NULL, 0, 0) != 0){
TranslateMessage(&msg);
DispatchMessage(&msg);
}

return msg.wParam;
}

/************************************************************
* ウィンドウプロシージャ関数の実体（メッセージの処理）
************************************************************/
LRESULT CALLBACK WinProc(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam){
void MIDIfileopen(HWND,HWND,HWND,HWND,HWND);
void MakeMIDIFile(HWND hCombo);
int i,j,r;
static HWND hButtonWnd1,hButtonPlay,hButtonPlay1;
static HCURSOR hArrow,hWait;
static HWND hCombo;
char StrCombo[300];
```

```
char Str[100];
char FileName[100];
int NoteOffPressure[3]={int(-((2+10)*4096/20+0.5)),0,int((1.5+10)*4096/20+0.5)};
HDC hdc;
PAINTSTRUCT ps;
FILE *fPressureData,*fPressureData2,*fPressureData3;
FILE *filep;

switch(Msg){
case WM_CREATE:
hDeviceDio=DioOpen("FBIDIO1",0);
if(hDeviceDio==INVALID_HANDLE_VALUE)
MessageBox(hWnd,"DOデバイスのオープンに失敗しました","",MB_OK);     //DioOpenに失敗
hDeviceDA=DaOpen("FBIDA1");
if(hDeviceDA==INVALID_HANDLE_VALUE)         //DaOpenに失敗
MessageBox(hWnd,"DAデバイスのオープンに失敗しました","",MB_OK);
ch.ulRange=DA_10V;                                  //バイポーラ±10

for(j=0;j<3;j++){
for(i=0;i<49;i++){
LipP[i][j]=0;
}
}
for(j=0;j<3;j++){
shinpuku[50][j]=0;
syuki[50][j]=0;
}

//圧力データの取得及びセット
fPressureData=fopen("PressureData.sax","r+");
if(fPressureData==NULL)MessageBox(hWnd,"初期圧力データを読み込めませんでした音量強
","",MB_OK);
else{
for(i=0;i<48;i++){
fscanf(fPressureData,"%d %d",&LipP[i][0],&BlowP[i][0]);
LipPressure[i][0]=(double)(LipP[i][0])/100;
```

```
BlowPressure[i][0]=(double)(BlowP[i][0])/100;
}//for

fscanf(fPressureData,"%d %d",&shinpuku[50][0],&syuki[50][0]);
shinpuku2[0]=shinpuku[50][0];
syuki2[0]=syuki[50][0];
fclose(fPressureData);
}//else

//圧力データの取得及びセット2
fPressureData2=fopen("PressureData2.sax","r+");
if(fPressureData2==NULL)MessageBox(hWnd,"初期圧力データを読み込めませんでした音量中
","",MB_OK);
else{
for(i=0;i<48;i++){
fscanf(fPressureData2,"%d %d",&LipP[i][1],&BlowP[i][1]);
LipPressure[i][1]=(double)(LipP[i][1])/100;
BlowPressure[i][1]=(double)(BlowP[i][1])/100;
}//for
fscanf(fPressureData2,"%d %d",&shinpuku[50][1],&syuki[50][1]);
shinpuku2[1]=shinpuku[50][1];
syuki2[1]=syuki[50][1];
fclose(fPressureData2);
}//else

//圧力データの取得及びセット23
fPressureData3=fopen("PressureData3.sax","r+");
if(fPressureData3==NULL)MessageBox(hWnd,"初期圧力データを読み込めませんでした音量弱
","",MB_OK);
else{
for(i=0;i<48;i++){
fscanf(fPressureData3,"%d %d",&LipP[i][2],&BlowP[i][2]);
LipPressure[i][2]=(double)(LipP[i][2])/100;
BlowPressure[i][2]=(double)(BlowP[i][2])/100;
}//for
fscanf(fPressureData3,"%d %d",&shinpuku[50][2],&syuki[50][2]);
```

```
shinpuku2[2]=shinpuku[50][2];
syuki2[2]=syuki[50][2];
fclose(fPressureData3);
}//else


//カーソルハンドルの取得
hArrow=LoadCursor(NULL,IDC_ARROW);
hWait=LoadCursor(NULL,IDC_WAIT);

//ボタン等の作成
hButtonWnd1 = CreateWindow("BUTTON", "実行", WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
150, 150, 80, 30, hWnd, (HMENU)ID_ACTION, hInstance ,NULL);
hCombo = CreateWindow("COMBOBOX", "",WS_CHILD | WS_VISIBLE | WS_VSCROLL |
CBS_DROPDOWNLIST,
70, 50, 220, 70, hWnd, (HMENU)ID_COMBO, hInstance, NULL);
hStatic = CreateWindow("STATIC", "ファイルを選択してください", WS_CHILD | WS_VISIBLE,
0, 0, 400, 18, hWnd, (HMENU)ID_STOP, hInstance ,NULL);
SendMessage(hCombo, CB_INSERTSTRING, -1,(LPARAM)CHOOSE);//ｺﾝﾎﾞBOXに文字列を追加し
SendMessage(hCombo, CB_SETCURSEL,0,0);   //その文字列を表示させる
break;

case WM_LBUTTONDBLCLK:
filep=fopen("PerformanceData.txt","w");
wsprintf(Str,"音階    ⊿ 舌 振  ﾚｲﾄ ﾃﾞﾌ ﾃﾞｨ  bend¥n");
fprintf(filep,Str);
for(r=0;r<1000;r++){
if(PerformanceData[r][0]==0xffffffff){
wsprintf(Str,"-1");
fprintf(filep,Str);
}else{
wsprintf(Str,"%02x",PerformanceData[r][0]);        //%x:16進数
fprintf(filep,Str);
}
wsprintf(Str," %6d",PerformanceData[r][1]);              //⊿
fprintf(filep,Str);
```

```
wsprintf(Str," %2d",PerformanceData[r][2]);              //舌
fprintf(filep,Str);
wsprintf(Str," %3d",PerformanceData[r][3]);              //振
fprintf(filep,Str);
wsprintf(Str," %3d",PerformanceData[r][4]);              //ﾚｲﾄ
fprintf(filep,Str);
wsprintf(Str," %3d",PerformanceData[r][5]);              //ﾃﾞﾌ
fprintf(filep,Str);
wsprintf(Str," %3d",PerformanceData[r][6]);              //ﾃﾞｨ
fprintf(filep,Str);
wsprintf(Str," %5d¥n",PerformanceData[r][7]);            //bend
fprintf(filep,Str);
}
fclose(filep);
SendMessage(hWnd,WM_COMMAND,(WPARAM)ID_MENU_INFO,NULL);
break;

case WM_DESTROY:
DioClose(hDeviceDio);
DaClose(hDeviceDA);
OnDestroy(hWnd);
break;

case WM_COMMAND:
switch(LOWORD(wParam)){
case ID_ACTION://実行ボタンが押されたときの処理
//MIDIから読み込んだデータの初期化
ZeroMemory(PerformanceData,sizeof(PerformanceData));
fp=fopen("test.txt","wb");
GetDlgItemText(hWnd,ID_COMBO,StrCombo,sizeof(StrCombo));
lstrcpy(FileName,"SaxPlayLog.txt");
remove(FileName);
if(strcmp(StrCombo,"選択してください")==0){              //strcmp:文字列の比較「==0」だと一致
//ダイアログボックスに”選択してください”と表示されているときの処理
if(SendMessage(hCombo, CB_SETCURSEL,1,0)==CB_ERR)        //拡張コンボボックス内に楽器名がない場合
```

```
MessageBox(hWnd,"ファイルが選択されていません","ファイルを選択してください",MB_OK);
else{
SendMessage(hCombo, CB_SETCURSEL,0,0);
MessageBox(hWnd,"楽器が選択されていません","楽器を選択してください",MB_OK);
}
}else {
SetWindowText(hStatic,"どちらかのボタンを押してください");
SetCursor(hWait);//カーソルを砂時計に。
MakeMIDIFile(hCombo);//ｺﾝﾎﾞBOXの楽器を除いたMIDIファイル作成する自作関数
OnActionButton(hWnd);//MCIデバイスを開く
DestroyWindow(hButtonPlay);
DestroyWindow(hButtonPlay1);

hButtonPlay = CreateWindow("BUTTON", "全再生", WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON1,
150, 250, 80, 30, hWnd, (HMENU)ID_PLAY, hInstance ,NULL);

hButtonPlay1 = CreateWindow("BUTTON", "ロボットのみ演奏", WS_CHILD | WS_VISIBLE |
BS_PUSHBUTTON2,
90, 300, 200, 30, hWnd, (HMENU)ID_MCISTOP, hInstance ,NULL);//ロボット演奏のみ用のボタン
2013.12.3

SetCursor(hArrow);//カーソルを矢にもどす
ch.ulChNo=2;                                    //チャンネル2唇圧
DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure[0]);
ch.ulChNo=3;                                    //チャンネル3タンギング
DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure[1]);
ch.ulChNo=4;                                    //チャンネル4送気圧
DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure[2]);
}

break;

case ID_PLAY://再生ボタン
OnPlayButtonDown(hWnd);
break;
```

```
case ID_MCISTOP://MCIは停止、ロボットのみ演奏用のボタンへの処理
OnStopButtonDown(hWnd);
break;

case ID_MENU_OPEN://メニューのファイルの開くの処理
MIDIfileopen(hWnd,hCombo,hButtonPlay,hButtonPlay1,hStatic);
break;

case ID_MENU_INFO://メニューのファイルの曲情報の処理
DialogBox(NULL, MAKEINTRESOURCE(IDD_MYDIALOG), NULL, DlgProc);
break;

case ID_MENU_CLOSE://メニューのファイルの閉じるの処理
SendMessage(hWnd, WM_CLOSE, 0, 0L);
break;

case ID_MENU_TUNING://メニューのチューニングの処理
DialogBox(NULL, MAKEINTRESOURCE(IDD_DIALOG1), NULL, TuningDlgProc);
break;
case ID_MENU_MANUAL://メニューのマニュアルの処理
DialogBox(NULL, MAKEINTRESOURCE(IDD_MANUAL), NULL, DlgProc2);
break;
}//switch文

case WM_PAINT://ウィンドウが移動したり隠れていたのが見えるようになったときの処理
hdc = BeginPaint(hWnd, &ps);
ShowMyBMP(hWnd, hdc);
EndPaint(hWnd, &ps);
break;

default://その他、ウィンドウを動かすなど・・・。
return DefWindowProc(hWnd, Msg, wParam, lParam);

}//switch文の終わり
return 0L;
}
```

```
/*****************************************************
*ダイアログプロシージャ
*****************************************************/
BOOL CALLBACK DlgProc(HWND hDlg, UINT Msg, WPARAM wParam, LPARAM lParam)
{
char StrMsg[250];

switch(Msg){
case WM_COMMAND:
switch(LOWORD(wParam)){
case IDOK:
EndDialog(hDlg, 0);
return TRUE;
}
break;

case WM_INITDIALOG:// ダイアログがを作成されたとき
wsprintf(StrMsg,"%d",bunkai2);
SetDlgItemText(hDlg, IDC_EDIT1,StrMsg);
wsprintf(StrMsg,"%d",tempo1);
SetDlgItemText(hDlg, IDC_EDIT2,StrMsg);
wsprintf(StrMsg,"%s",chou_chou);
SetDlgItemText(hDlg, IDC_EDIT3,StrMsg);
wsprintf(StrMsg,"%s",hyousi_hyousi);
SetDlgItemText(hDlg, IDC_EDIT4,StrMsg);
break;

case WM_CLOSE:
EndDialog(hDlg, 0);
return TRUE;
}//switch文

return FALSE;     //上記の命令がなかったとき
}
```

```
/*******************************************************
*ダイアログプロシージャ(マニュアル用)
*******************************************************/
BOOL CALLBACK DlgProc2(HWND hDlg, UINT Msg, WPARAM wParam, LPARAM lParam)
{
char StrMsg[250];

switch(Msg){
case WM_COMMAND:
switch(LOWORD(wParam)){
case IDOK:
EndDialog(hDlg, 0);
return TRUE;
}
break;

case WM_CLOSE:
EndDialog(hDlg, 0);
return TRUE;
}//switch文

return FALSE;     //上記の命令がなかったとき
}

/*******************************************************
*チューニングダイアログプロシージャ
*******************************************************/
BOOL CALLBACK TuningDlgProc(HWND hTuningDlg, UINT Msg, WPARAM wParam, LPARAM
lParam)
{
void wait(double,int,bool*);
void cyutime(double Second2,int now2);

static HWND hUpdown[100], hEdit[100],hParent;    //アップダウンコントロールで使う。
//static HWND hUpdown2[97], hEdit2[97],hUpdown2[98], hEdit2[98];//ビブラート用20150115
```

```
static FILE *fPressureData,*fPressureData2,*fPressureData3;//唇、送気圧を保存しておくファイル
bool PlayCheck=TRUE;
static OPENFILENAME ofn;     //コモンダイアログに使う構造体
static TCHAR strFile[MAX_PATH];   //コモンダイアログ構造体に使用する変数
char strMsg[250];                 //コモンダイアログ時に使用する変数

TCHAR Buff[100][5];         //←名前をつけて保存ボタンのための変数。48鍵盤分

int iBuff[100][3];              //48鍵盤分とノートオフの分の唇圧iBuff[48]で計４９個
int issBuff[100][3];

int Lip[2], Blow[2], Ton[2];
int NoteOffPressure = 0xfff/2;

float dummyA[51], dummyB[51], dLipP[51], dBlowP[51];                    //MEXファイルを開くときに用いる

int nRet, nRetoff;                  //Dioボードへの出力の際に用いる

int i,j;
//              char str[30];

for(i=0;i<100;i++)
SendDlgItemMessage(hTuningDlg,idc_edit[i],EM_SETLIMITTEXT,3,0);//エディットコントロール最大文字数3に指定

switch(Msg)
{
case WM_INITDIALOG://ダイアログボックスが作成されたときの処理。

for(i=0;i<48;i++)
{
hUpdown[i] = GetDlgItem(hTuningDlg, idc_spin[i]);
hEdit[i] = GetDlgItem(hTuningDlg, idc_edit[i]);
SendMessage(hUpdown[i], UDM_SETBUDDY, (WPARAM)hEdit[i], 0);
SendMessage(hUpdown[i], UDM_SETRANGE32, (WPARAM)0, (LPARAM)999);
```

```
SendMessage(hUpdown[i], UDM_SETPOS, 0, (LPARAM)MAKELONG((short)LipP[i][0], 0));

hUpdown[i+48] = GetDlgItem(hTuningDlg, idc_spin[i+48]);
hEdit[i+48] = GetDlgItem(hTuningDlg, idc_edit[i+48]);
SendMessage(hUpdown[i+48], UDM_SETBUDDY, (WPARAM)hEdit[i+48], 0);
SendMessage(hUpdown[i+48], UDM_SETRANGE32, (WPARAM)0, (LPARAM)999);
SendMessage(hUpdown[i+48], UDM_SETPOS, 0, (LPARAM)MAKELONG((short)BlowP[i][0], 0));
}

hUpdown[96] = GetDlgItem(hTuningDlg, idc_spin[96]);
hEdit[96] = GetDlgItem(hTuningDlg, idc_edit[96]);
SendMessage(hUpdown[96], UDM_SETBUDDY, (WPARAM)hEdit[96], 0);
SendMessage(hUpdown[96], UDM_SETRANGE32, (WPARAM)0, (LPARAM)100);
SendMessage(hUpdown[96], UDM_SETPOS, 0, (LPARAM)MAKELONG((short)shinpuku[50][0], 0));

hUpdown[97] = GetDlgItem(hTuningDlg, idc_spin[97]);
hEdit[97] = GetDlgItem(hTuningDlg, idc_edit[97]);
SendMessage(hUpdown[97], UDM_SETBUDDY, (WPARAM)hEdit[97], 0);
SendMessage(hUpdown[97], UDM_SETRANGE32, (WPARAM)0, (LPARAM)100);
SendMessage(hUpdown[97], UDM_SETPOS, 0, (LPARAM)MAKELONG((short)syuki[50][0], 0));

//コモンダイアログの設定
ofn.lStructSize = sizeof (ofn);
ofn.hwndOwner = hTuningDlg;
ofn.lpstrFile = strFile;        //選択されたファイル名を代入するためのバッファ
ofn.nMaxFile = MAX_PATH;
ofn.lpstrFilter = "チューニングデータ¥0*.sax;*.mex¥0¥0";
ofn.nFilterIndex = 1;
ofn.lpstrDefExt = "sax";
ofn.Flags = OFN_FILEMUSTEXIST | OFN_OVERWRITEPROMPT | OFN_NOCHANGEDIR;
ofn.lpstrTitle = "チューニング(強)";   //新たなウインドウのキャプションバー

//圧力データの取得及びセット(強)
fPressureData = fopen("PressureData.sax","r+");
```

```
if(fPressureData == NULL)
{
MessageBox(hTuningDlg,"初期データの取得に失敗しました音量強","NOT FOUND",MB_OK);
}
else
{
for(i=0;i<48;i++)
{
fscanf(fPressureData, "%d %d", &LipP[i][0], &BlowP[i][0]);
//fscanf(fPressureData,"%d %d",&shinpuku[50][0],&syuki[50][0]);
if(dina==0){
SetDlgItemInt(hTuningDlg, idc_edit[i], LipP[i][0], FALSE);
SetDlgItemInt(hTuningDlg, idc_edit[i+48], BlowP[i][0], FALSE);

}

}
fscanf(fPressureData,"%d %d",&shinpuku[50][0],&syuki[50][0]);
if(dina==0){
SetDlgItemInt(hTuningDlg, idc_edit[96], shinpuku[50][0], FALSE);
SetDlgItemInt(hTuningDlg, idc_edit[97], syuki[50][0], FALSE);
}
fclose(fPressureData);
}

//圧力データの取得及びセット(中)
fPressureData2 = fopen("PressureData2.sax","r+");

if(fPressureData2 == NULL)
{
MessageBox(hTuningDlg,"初期データの取得に失敗しました音量中","NOT FOUND",MB_OK);
}
else
{
for(i=0;i<48;i++)
{
```

```
fscanf(fPressureData2, "%d %d", &LipP[i][1], &BlowP[i][1]);

if(dina==1){
SetDlgItemInt(hTuningDlg, idc_edit[i], LipP[i][1], FALSE);
SetDlgItemInt(hTuningDlg, idc_edit[i+48], BlowP[i][1], FALSE);

}
}
fscanf(fPressureData2,"%d %d",&shinpuku[50][1],&syuki[50][1]);
if(dina==1){
SetDlgItemInt(hTuningDlg, idc_edit[96], shinpuku[50][1], FALSE);
SetDlgItemInt(hTuningDlg, idc_edit[97], syuki[50][1], FALSE);
}
fclose(fPressureData2);
}

//圧力データの取得及びセット(弱)
fPressureData3 = fopen("PressureData3.sax","r+");

if(fPressureData3 == NULL)
{
MessageBox(hTuningDlg,"初期データの取得に失敗しました音量弱","NOT FOUND",MB_OK);
}
else
{
for(i=0;i<48;i++)
{
fscanf(fPressureData3, "%d %d", &LipP[i][2], &BlowP[i][2]);

if(dina==2){
SetDlgItemInt(hTuningDlg, idc_edit[i], LipP[i][2], FALSE);
SetDlgItemInt(hTuningDlg, idc_edit[i+48], BlowP[i][2], FALSE);

}
}
fscanf(fPressureData3,"%d %d",&shinpuku[50][2],&syuki[50][2]);
```

```
if(dina==2){
SetDlgItemInt(hTuningDlg, idc_edit[96], shinpuku[50][2], FALSE);
SetDlgItemInt(hTuningDlg, idc_edit[97], syuki[50][2], FALSE);
}
fclose(fPressureData3);
}

//////////待機状態からすぐ吹鳴させるために行う処理//////////
Lip[0] = 200;
Blow[0] = 150;
Ton[0] = 300;

Lip[1] = -1*(int)(((double)(Lip[0])/100+10)*4096/20+0.5);
Blow[1] = (int)(((double)(Blow[0])/100+10)*4096/20+0.5);
Ton[1] = (int)(((double)(Ton[0])/100+10)*4096/20+0.5);

//チャンネル2にリード圧の出力
ch.ulChNo = 2;
DaOutputDA(hDeviceDA, 1, &ch, &Lip[1]);

//チャンネル4に送気圧の出力
ch.ulChNo = 4;
DaOutputDA(hDeviceDA, 1, &ch, &Blow[1]);

//チャンネル3にタンギングの出力
ch.ulChNo = 3;
DaOutputDA(hDeviceDA, 1, &ch, &Ton[1]);
////////////////////////////////////////////////////////////

break;

case WM_COMMAND:       //ボタンが押された時の処理

switch(LOWORD(wParam))
{
case IDC_BUTTON1:        //閉じるボタン
```

```
SendMessage(hTuningDlg, WM_CLOSE, 0, 0);
break;

case IDC_BUTTON2: //名前をつけて保存ボタン
for(i=0; i<96; i++)     //96個のエディットボックスの値を、Buffに保存
{
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[i]), Buff[i], 5);
}
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[96]), Buff[99], 5);
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[97]), Buff[98], 5);
for(i=0; i<48; i++)     //Buffの値をテキスト分から整数に直す
{
iBuff[i][dina] = atoi(Buff[i]);
issBuff[i][dina] = atoi(Buff[i+48]);
}
iBuff[99][dina] = atoi(Buff[99]);
issBuff[98][dina] = atoi(Buff[98]);

if( !GetSaveFileName(&ofn))
{
MessageBox(NULL, "ファイルが選択されていません", "", MB_OK | MB_ICONSTOP);
}
else
{
if(dina==0) fPressureData = fopen(strFile, "w+");
if(dina==1) fPressureData2 = fopen(strFile, "w+");
if(dina==2) fPressureData3 = fopen(strFile, "w+");
}

if(fPressureData == NULL)
{
MessageBox(hTuningDlg, "新規圧力データを保存できませんでした", "", MB_OK);
}
else
{
if(dina==0){
```

```
for(i=0; i<48; i++)
{
fprintf(fPressureData, "%d %d¥n", iBuff[i][0], issBuff[i][0]);
}
fprintf(fPressureData, "%d %d¥n", iBuff[99][0], issBuff[98][0]);
}
if(dina==1){
for(i=0; i<48; i++)
{
fprintf(fPressureData2, "%d %d¥n", iBuff[i][1], issBuff[i][1]);
}
fprintf(fPressureData2, "%d %d¥n", iBuff[99][1], issBuff[98][1]);
}
if(dina==2){
for(i=0; i<48; i++)
{
fprintf(fPressureData3, "%d %d¥n", iBuff[i][2], issBuff[i][2]);
}
fprintf(fPressureData3, "%d %d¥n", iBuff[99][2], issBuff[98][2]);
}
}
fclose(fPressureData);
fclose(fPressureData2);
fclose(fPressureData3);

break;

case IDC_BUTTON3:     //*.saxファイルを開く。エディットボックスにある値をLipP[]BlowP[]に上書き
if(!GetOpenFileName(&ofn))
{
MessageBox(NULL, "ファイルが選択されていません", "", MB_OK | MB_ICONSTOP);
}
else
{
if(dina==0)fPressureData = fopen(strFile, "r+");
if(dina==1)fPressureData2 = fopen(strFile, "r+");
```

```
if(dina==2)fPressureData3 = fopen(strFile, "r+");
if(fPressureData == NULL)
{
MessageBox(hTuningDlg, "圧力データを読み込めませんでした", "", MB_OK);
}
else
{
if(dina==0){
for(i=0; i<48; i++)
{
fscanf(fPressureData, "%d %d¥n", &LipP[i][dina], &BlowP[i][dina]);
}
fscanf(fPressureData,"%d %d",&shinpuku[50][dina],&syuki[50][dina]);
fclose(fPressureData);
}
if(dina==1){
for(i=0; i<48; i++)
{
fscanf(fPressureData2, "%d %d¥n", &LipP[i][dina], &BlowP[i][dina]);
}
fscanf(fPressureData2,"%d %d",&shinpuku[50][dina],&syuki[50][dina]);
fclose(fPressureData2);
}
if(dina==2){
for(i=0; i<48; i++)
{
fscanf(fPressureData3, "%d %d¥n", &LipP[i][dina], &BlowP[i][dina]);
}
fscanf(fPressureData3,"%d %d",&shinpuku[50][dina],&syuki[50][dina]);
fclose(fPressureData3);
}

for(i=0;i<48;i++)
{
SetDlgItemInt(hTuningDlg, idc_edit[i], LipP[i][dina], FALSE);          //リード圧
SetDlgItemInt(hTuningDlg, idc_edit[i+48], BlowP[i][dina], FALSE);      //送気圧
```

```
}
SetDlgItemInt(hTuningDlg, idc_edit[96], shinpuku[50][dina], FALSE);                    //リード圧
SetDlgItemInt(hTuningDlg, idc_edit[97], syuki[50][dina], FALSE);            //送気圧

//////////待機状態からすぐ吹鳴させるために行う処理//////////
Lip[0] = 200;
Blow[0] = 150;
Ton[0] = 300;

Lip[1] = -1*(int)(((double)(Lip[0])/100+10)*4096/20+0.5);
Blow[1] = (int)(((double)(Blow[0])/100+10)*4096/20+0.5);
Ton[1] = (int)(((double)(Ton[0])/100+10)*4096/20+0.5);

//チャンネル2にリード圧の出力
ch.ulChNo = 2;
DaOutputDA(hDeviceDA, 1, &ch, &Lip[1]);

//チャンネル4に送気圧の出力
ch.ulChNo = 4;
DaOutputDA(hDeviceDA, 1, &ch, &Blow[1]);

//チャンネル3にタンギングの出力
ch.ulChNo = 3;
DaOutputDA(hDeviceDA, 1, &ch, &Ton[1]);
////////////////////////////////////////////////////////////

wsprintf(strMsg, "ファイルが選択されました。¥nファイル名は¥n%s¥nです。",strFile);
MessageBox(NULL, strMsg, "", MB_OK | MB_ICONINFORMATION);
}
}
break;

case IDC_BUTTON17:     //*.mexファイルを開く。エディットボックスにある値をLipP[]BlowP[]に上書き。
if(!GetOpenFileName(&ofn))
{
```

```
MessageBox(NULL, "ファイルが選択されていません", "", MB_OK | MB_ICONSTOP);
}
else
{
fPressureData = fopen(strFile, "r+");
if(fPressureData == NULL)
{
MessageBox(hTuningDlg, "初期圧力データを読み込めませんでした", "", MB_OK);
}
else
{
for(i=0; i<49; i++)
{
fscanf(fPressureData, "%f %f %f %f¥n", &dBlowP[i], &dummyA[i], &dLipP[i], &dummyB[i]);
//dLipP[i],dBlowP[i]はPressureData.mexに書き込んである実データ
}
fclose(fPressureData);

for(i=0; i<48; i++)                    //代入
{
LipP[i][dina] = (int)(100 * dLipP[i] + 0.5);
BlowP[i][dina] = (int)(100 * dBlowP[i+1] + 0.5);
}

for(i=0; i<48; i++)
{
SetDlgItemInt(hTuningDlg, idc_edit[i], LipP[i][dina], FALSE);                    //リード圧
SetDlgItemInt(hTuningDlg, idc_edit[i+48], BlowP[i][dina], FALSE);                //送気圧
}

//////////待機状態からすぐ吹鳴させるために行う処理/////////
Lip[0] = 200;
Blow[0] = 150;
Ton[0] = 300;

Lip[1] = -1*(int)(((double)(Lip[0])/100+10)*4096/20+0.5);
```

```
Blow[1] = (int)(((double)(Blow[0])/100+10)*4096/20+0.5);
Ton[1] = (int)(((double)(Ton[0])/100+10)*4096/20+0.5);

//チャンネル2にリード圧の出力
ch.ulChNo = 2;
DaOutputDA(hDeviceDA, 1, &ch, &Lip[1]);

//チャンネル4に送気圧の出力
ch.ulChNo = 4;
DaOutputDA(hDeviceDA, 1, &ch, &Blow[1]);

//チャンネル3にタンギングの出力
ch.ulChNo = 3;
DaOutputDA(hDeviceDA, 1, &ch, &Ton[1]);
////////////////////////////////////////////////////////////

wsprintf(strMsg, "ファイルが選択されました。¥nファイル名は¥n%s¥nです。", strFile);
MessageBox(NULL, strMsg, "", MB_OK | MB_ICONINFORMATION);
}
}
break;

case IDC_BUTTON4:              //吹鳴中止の指令

//////////待機状態からすぐ吹鳴させるために行う処理//////////
Lip[0] = 200;
Blow[0] = 150;
Ton[0] = 300;

Lip[1] = -1*(int)(((double)(Lip[0])/100+10)*4096/20+0.5);
Blow[1] = (int)(((double)(Blow[0])/100+10)*4096/20+0.5);
Ton[1] = (int)(((double)(Ton[0])/100+10)*4096/20+0.5);

//チャンネル2にリード圧の出力
ch.ulChNo = 2;
DaOutputDA(hDeviceDA, 1, &ch, &Lip[1]);
```

```
//チャンネル4に送気圧の出力
ch.ulChNo = 4;
DaOutputDA(hDeviceDA, 1, &ch, &Blow[1]);

//チャンネル3にタンギングの出力
ch.ulChNo = 3;
DaOutputDA(hDeviceDA, 1, &ch, &Ton[1]);
//////////////////////////////////////////////////////

SetTimer(hTuningDlg,ID_MYTIMER, 20000, NULL);       //←WM_TIMER:へ。20秒後待機状態の指令。

break;

case IDC_BUTTON18:   //タンギングボタン　チューニング中、音を安定させるため

//チャンネル3にタンギングONの出力
Ton[0] = 300;
Ton[1]=(int)(((double)(Ton[0])/100+10)*4096/20+0.5);
ch.ulChNo = 3;
DaOutputDA(hDeviceDA, 1, &ch, &Ton[1]);

Sleep(200); //0.2秒後にタンギングOFF

//チャンネル3にタンギングOFFの出力
Ton[0] = 0;
Ton[1]=(int)(((double)(Ton[0])/100+10)*4096/20+0.5);
ch.ulChNo = 3;
DaOutputDA(hDeviceDA, 1, &ch, &Ton[1]);

break;


case IDC_BUTTON14:   //強ボタン　保存先を「強」に

fPressureData2 = fopen("PressureData2.sax","w");
```

```
if(dina==1){
for(i=0; i<96; i++)     //96個のエディットボックスの値を、Buffに保存
{
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[i]), Buff[i], 5);
}
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[96]), Buff[99], 5);
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[97]), Buff[98], 5);

for(i=0; i<48; i++)     //Buffの値をテキスト分から整数に直す
{
iBuff[i][1] = atoi(Buff[i]);
issBuff[i][1] = atoi(Buff[i+48]);
}
iBuff[99][1] = atoi(Buff[99]);
issBuff[98][1] = atoi(Buff[98]);

for(i=0;i<48;i++)
{
fprintf(fPressureData2, "%d %d¥n", iBuff[i][1], issBuff[i][1]);
LipP[i][1]=iBuff[i][1];
BlowP[i][1]=issBuff[i][1];
}
fprintf(fPressureData2, "%d %d¥n", iBuff[99][1], issBuff[98][1]);
shinpuku[50][1]=iBuff[99][1];
syuki[50][1]=issBuff[98][1];

}
else{
for(i=0;i<48;i++)
{
fprintf(fPressureData2, "%d %d¥n", LipP[i][1], BlowP[i][1]);
}
fprintf(fPressureData2, "%d %d¥n", shinpuku[50][1], syuki[50][1]);
}
fprintf(fPressureData2, "%d¥n", dina);
fclose(fPressureData2);
```

```
fPressureData3 = fopen("PressureData3.sax","w");
if(dina==2){
for(i=0; i<96; i++)     //96個のエディットボックスの値を、Buffに保存
{
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[i]), Buff[i], 5);
}
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[96]), Buff[99], 5);
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[97]), Buff[98], 5);

for(i=0; i<48; i++)     //Buffの値をテキスト分から整数に直す
{
iBuff[i][2] = atoi(Buff[i]);
issBuff[i][2] = atoi(Buff[i+48]);
}
iBuff[99][2] = atoi(Buff[99]);
issBuff[98][2] = atoi(Buff[98]);

for(i=0;i<48;i++)
{
fprintf(fPressureData3, "%d %d¥n", iBuff[i][2], issBuff[i][2]);
LipP[i][2]=iBuff[i][2];
BlowP[i][2]=issBuff[i][2];
}
fprintf(fPressureData3, "%d %d¥n", iBuff[99][2], issBuff[98][2]);
shinpuku[50][2]=iBuff[99][2];
syuki[50][2]=issBuff[98][2];
}
else{
for(i=0;i<48;i++)
{
fprintf(fPressureData3, "%d %d¥n", LipP[i][2], BlowP[i][2]);
}
fprintf(fPressureData3, "%d %d¥n", shinpuku[50][2], syuki[50][2]);
}
fprintf(fPressureData3, "%d¥n", dina);
```

```
fclose(fPressureData3);

fPressureData = fopen("PressureData.sax","r+");
for(i=0;i<48;i++)
{
fscanf(fPressureData, "%d %d", &LipP[i][0], &BlowP[i][0]);
SetDlgItemInt(hTuningDlg, idc_edit[i], LipP[i][0], FALSE);
SetDlgItemInt(hTuningDlg, idc_edit[i+48], BlowP[i][0], FALSE);
}
fscanf(fPressureData, "%d %d", &shinpuku[50][0], &syuki[50][0]);
SetDlgItemInt(hTuningDlg, idc_edit[96], shinpuku[50][0], FALSE);
SetDlgItemInt(hTuningDlg, idc_edit[97], syuki[50][0], FALSE);
fclose(fPressureData);

dina=0;

break;

case IDC_BUTTON15:   //中ボタン　保存先を「中」に

fPressureData = fopen("PressureData.sax","w+");
if(dina==0){
for(i=0; i<96; i++)     //96個のエディットボックスの値を、Buffに保存
{
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[i]), Buff[i], 5);
}
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[96]), Buff[99], 5);
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[97]), Buff[98], 5);
for(i=0; i<48; i++)     //Buffの値をテキスト分から整数に直す
{
iBuff[i][0] = atoi(Buff[i]);
issBuff[i][0] = atoi(Buff[i+48]);
}
iBuff[99][0] = atoi(Buff[99]);
issBuff[98][0] = atoi(Buff[98]);
for(i=0;i<48;i++)
```

```
{
fprintf(fPressureData, "%d %d¥n", iBuff[i][0], issBuff[i][0]);
LipP[i][0]=iBuff[i][0];
BlowP[i][0]=issBuff[i][0];
}
fprintf(fPressureData, "%d %d¥n", iBuff[99][0], issBuff[98][0]);
shinpuku[50][0]=iBuff[99][0];
syuki[50][0]=issBuff[98][0];
}
else{
for(i=0;i<48;i++)
{
fprintf(fPressureData, "%d %d¥n", LipP[i][0], BlowP[i][0]);
}
fprintf(fPressureData, "%d %d¥n", shinpuku[50][0], syuki[50][0]);
}
fprintf(fPressureData, "%d¥n", dina);
fclose(fPressureData);

fPressureData3 = fopen("PressureData3.sax","w+");
if(dina==2){
for(i=0; i<96; i++)     //96個のエディットボックスの値を、Buffに保存
{
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[i]), Buff[i], 5);
}
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[96]), Buff[99], 5);
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[97]), Buff[98], 5);
for(i=0; i<48; i++)     //Buffの値をテキスト分から整数に直す
{
iBuff[i][2] = atoi(Buff[i]);
issBuff[i][2] = atoi(Buff[i+48]);
}
iBuff[99][2] = atoi(Buff[99]);
issBuff[98][2] = atoi(Buff[98]);
for(i=0;i<48;i++)
{
```

```
fprintf(fPressureData3, "%d %d¥n", iBuff[i][2], issBuff[i][2]);
LipP[i][2]=iBuff[i][2];
BlowP[i][2]=issBuff[i][2];
}
fprintf(fPressureData3, "%d %d¥n", iBuff[99][2], issBuff[98][2]);
shinpuku[50][2]=iBuff[99][2];
syuki[50][2]=issBuff[98][2];
}
else{
for(i=0;i<48;i++)
{
fprintf(fPressureData3, "%d %d¥n", LipP[i][2], BlowP[i][2]);
}
fprintf(fPressureData3, "%d %d¥n", shinpuku[50][2], syuki[50][2]);
}
fprintf(fPressureData3, "%d¥n", dina);
fclose(fPressureData3);

fPressureData2 = fopen("PressureData2.sax","r+");
for(i=0;i<48;i++)
{
fscanf(fPressureData2, "%d %d¥n", &LipP[i][1], &BlowP[i][1]);
SetDlgItemInt(hTuningDlg, idc_edit[i], LipP[i][1], FALSE);
SetDlgItemInt(hTuningDlg, idc_edit[i+48], BlowP[i][1], FALSE);
}
fscanf(fPressureData2, "%d %d", &shinpuku[50][1], &syuki[50][1]);
SetDlgItemInt(hTuningDlg, idc_edit[96], shinpuku[50][1], FALSE);
SetDlgItemInt(hTuningDlg, idc_edit[97], syuki[50][1], FALSE);
fclose(fPressureData2);

dina=1;

break;

case IDC_BUTTON16:   //強ボタン　保存先を「弱」に
```

```
fPressureData = fopen("PressureData.sax","w+");
if(dina==0){
for(i=0; i<96; i++)     //96個のエディットボックスの値を、Buffに保存
{
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[i]), Buff[i], 5);
}
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[96]), Buff[99], 5);
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[97]), Buff[98], 5);
for(i=0; i<48; i++)     //Buffの値をテキスト分から整数に直す
{
iBuff[i][0] = atoi(Buff[i]);
issBuff[i][0] = atoi(Buff[i+48]);
}
iBuff[99][0] = atoi(Buff[99]);
issBuff[98][0] = atoi(Buff[98]);
for(i=0;i<48;i++)
{
fprintf(fPressureData, "%d %d¥n", iBuff[i][0], issBuff[i][0]);
LipP[i][0]=iBuff[i][0];
BlowP[i][0]=issBuff[i][0];
}
fprintf(fPressureData, "%d %d¥n", iBuff[99][0], issBuff[98][0]);
shinpuku[50][0]=iBuff[99][0];
syuki[50][0]=issBuff[98][0];
}
else{
for(i=0;i<48;i++)
{
fprintf(fPressureData, "%d %d¥n", LipP[i][0], BlowP[i][0]);
}
fprintf(fPressureData, "%d %d¥n", shinpuku[50][0], syuki[50][0]);
}
fprintf(fPressureData, "%d¥n", dina);
fclose(fPressureData);
```

```
fPressureData2 = fopen("PressureData2.sax","w+");
if(dina==1){
for(i=0; i<96; i++)     //96個のエディットボックスの値を、Buffに保存
{
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[i]), Buff[i], 5);
}
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[96]), Buff[99], 5);
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[97]), Buff[98], 5);
for(i=0; i<48; i++)     //Buffの値をテキスト分から整数に直す
{
iBuff[i][1] = atoi(Buff[i]);
issBuff[i][1] = atoi(Buff[i+48]);
}
iBuff[99][1] = atoi(Buff[99]);
issBuff[98][1] = atoi(Buff[98]);
for(i=0;i<48;i++)
{
fprintf(fPressureData2, "%d %d¥n", iBuff[i][1], issBuff[i][1]);
LipP[i][1]=iBuff[i][1];
BlowP[i][1]=issBuff[i][1];
}
fprintf(fPressureData2, "%d %d¥n", iBuff[99][1], issBuff[98][1]);
shinpuku[50][1]=iBuff[99][1];
syuki[50][1]=issBuff[98][1];
}
else{
for(i=0;i<48;i++)
{
fprintf(fPressureData2, "%d %d¥n", LipP[i][1], BlowP[i][1]);
}
fprintf(fPressureData2, "%d %d¥n", shinpuku[50][1], syuki[50][1]);
}
fprintf(fPressureData2, "%d¥n", dina);
fclose(fPressureData2);

fPressureData3 = fopen("PressureData3.sax","r+");
```

```
for(i=0;i<48;i++)
{
fscanf(fPressureData3, "%d %d", &LipP[i][2], &BlowP[i][2]);
SetDlgItemInt(hTuningDlg, idc_edit[i], LipP[i][2], FALSE);
SetDlgItemInt(hTuningDlg, idc_edit[i+48], BlowP[i][2], FALSE);
}
fscanf(fPressureData3, "%d %d", &shinpuku[50][2], &syuki[50][2]);
SetDlgItemInt(hTuningDlg, idc_edit[96], shinpuku[50][2], FALSE);
SetDlgItemInt(hTuningDlg, idc_edit[97], syuki[50][2], FALSE);
fclose(fPressureData3);

dina=2;

break;

case IDC_BUTTON11:   //ビブラートボタン用
if(bib3==0)bib3=1;
else bib3=0;

break;

}//switch(LOWORD(wParam))文終了

if(HIWORD(wParam) == EN_SETFOCUS)
{
tuningflag = 1;
}

if(tuningflag == 1)
{
for(i=0;i<48;i++)
{
if((lParam == (LPARAM)hEdit[i] || lParam == (LPARAM)hEdit[i+48]) &&
(HIWORD(wParam) == EN_CHANGE || HIWORD(wParam) == EN_SETFOCUS))
//                                          if(lParam == (LPARAM)hEdit[i] && HIWORD(wParam) ==
EN_CHANGE)                    //確認用
```

```
{
Lip[0] = (int)SendMessage(hUpdown[i], UDM_GETPOS, 0, 0);

cyulip=Lip[0];

Blow[0] = (int)SendMessage(hUpdown[i+48], UDM_GETPOS, 0, 0);
Ton[0] = 0;

if(bib3==0)Lip[1] = -1*(int)(((double)(Lip[0])/100+10)*4096/20+0.5);
Blow[1] = (int)(((double)(Blow[0])/100+10)*4096/20+0.5);
Ton[1] = (int)(((double)(Ton[0])/100+10)*4096/20+0.5);

//チャンネル2にリード圧の出力
if(bib3==0){
ch.ulChNo = 2;
DaOutputDA(hDeviceDA, 1, &ch, &Lip[1]);
}
//チャンネル4に送気圧を出力
ch.ulChNo = 4;
DaOutputDA(hDeviceDA, 1, &ch, &Blow[1]);

//チャンネル3にタンギングoffの出力
ch.ulChNo = 3;
DaOutputDA(hDeviceDA, 1, &ch, &Ton[1]);

//運指
nRet = DioOutputPoint(hDeviceDio, FingerPosition[i], 1, 32);

//                                                    wsprintf(str, "i=%d¥nLip[0]=%d¥nBlow[0]=%d", i,
Lip[0], Blow[0]);                    //確認用
}//if
}//for
cyutime(5000,0);//ビブラート用5秒間ビブラートを起こす
bib3=0;
}
```

```
break;


case WM_TIMER:      //チューニング画面で20秒何も動作を行わなかった時に発生するメッセージ。
KillTimer(hTuningDlg, ID_MYTIMER);

//////////チャンネル３にタンギングの出力/////////
ch.ulChNo = 3;
DaOutputDA(hDeviceDA, 1, &ch, &NoteOffPressure);

//////////////チャンネル２に唇圧の出力//////////
ch.ulChNo = 2;
DaOutputDA(hDeviceDA, 1, &ch, &NoteOffPressure);

//////////////チャンネル４に送気圧を出力/////////
ch.ulChNo = 4;
DaOutputDA(hDeviceDA, 1, &ch, &NoteOffPressure);

//////////////運指の開放////////////////////////
nRetoff = DioOutputPoint(hDeviceDio, NoteOffFinger, 1, 32);

break;

case WM_CLOSE:


if(dina==0){
fPressureData = fopen("PressureData.sax", "w+");

for(i=0;i<48;i++)
{
GetWindowText(GetDlgItem(hTuningDlg,idc_edit[i]),Buff[i],5);
LipP[i][0] = atoi(Buff[i]);
fprintf(fPressureData, "%d ", LipP[i][0]);
LipPressure[i][0] = int(LipP[i][0]);
```

```
GetWindowText(GetDlgItem(hTuningDlg, idc_edit[i+48]), Buff[i+48], 5);
BlowP[i][0] = atoi(Buff[i+48]);
fprintf(fPressureData, "%d¥n", BlowP[i][0]);
BlowPressure[i][0] = int(BlowP[i][0]);
}
GetWindowText(GetDlgItem(hTuningDlg,idc_edit[96]),Buff[99],5);
shinpuku[50][0] = atoi(Buff[99]);
fprintf(fPressureData, "%d ", shinpuku[50][0]);
shinpuku2[0] = (shinpuku[50][0]);

GetWindowText(GetDlgItem(hTuningDlg,idc_edit[97]),Buff[98],5);
syuki[50][0] = atoi(Buff[98]);
fprintf(fPressureData, "%d ", syuki[50][0]);
syuki2[0] = (syuki[50][0]);
}
else{
for(i=0;i<48;i++){
LipPressure[i][0] = int(LipP[i][0]);
BlowPressure[i][0] = int(BlowP[i][0]);
}
shinpuku2[0] = (shinpuku[50][0]);
syuki2[0] = (syuki[50][0]);
}
fclose(fPressureData);

if(dina==1){
fPressureData2 = fopen("PressureData2.sax", "w+");

for(i=0;i<48;i++)
{
GetWindowText(GetDlgItem(hTuningDlg,idc_edit[i]),Buff[i],5);
LipP[i][1] = atoi(Buff[i]);
fprintf(fPressureData2, "%d ", LipP[i][1]);
LipPressure[i][1] = int(LipP[i][1]);

GetWindowText(GetDlgItem(hTuningDlg, idc_edit[i+48]), Buff[i+48], 5);
```

```
BlowP[i][1] = atoi(Buff[i+48]);
fprintf(fPressureData2, "%d¥n", BlowP[i][1]);
BlowPressure[i][1] = int(BlowP[i][1]);
}
GetWindowText(GetDlgItem(hTuningDlg,idc_edit[96]),Buff[99],5);
shinpuku[50][1] = atoi(Buff[99]);
fprintf(fPressureData2, "%d ", shinpuku[50][1]);
shinpuku2[1] = (shinpuku[50][1]);

GetWindowText(GetDlgItem(hTuningDlg,idc_edit[97]),Buff[98],5);
syuki[50][1] = atoi(Buff[98]);
fprintf(fPressureData2, "%d ", syuki[50][1]);
syuki2[1] = (syuki[50][1]);
}
else{
for(i=0;i<48;i++){
LipPressure[i][1] = int(LipP[i][1]);
BlowPressure[i][1] = int(BlowP[i][1]);
}
shinpuku2[1] =(shinpuku[50][1]);
syuki2[1] =(syuki[50][1]);
}

fclose(fPressureData2);

if(dina==2){
fPressureData3 = fopen("PressureData3.sax", "w+");

for(i=0;i<48;i++)
{
GetWindowText(GetDlgItem(hTuningDlg,idc_edit[i]),Buff[i],5);
LipP[i][2] = atoi(Buff[i]);
fprintf(fPressureData3, "%d ", LipP[i][2]);
LipPressure[i][2] = int(LipP[i][2]);

GetWindowText(GetDlgItem(hTuningDlg, idc_edit[i+48]), Buff[i+48], 5);
```

```
BlowP[i][2] = atoi(Buff[i+48]);
fprintf(fPressureData3, "%d¥n", BlowP[i][2]);
BlowPressure[i][2] = int(BlowP[i][2]);
}
GetWindowText(GetDlgItem(hTuningDlg,idc_edit[96]),Buff[99],5);
shinpuku[50][2] = atoi(Buff[99]);
fprintf(fPressureData3, "%d ", shinpuku[50][2]);
shinpuku2[2] =shinpuku[50][2];

GetWindowText(GetDlgItem(hTuningDlg,idc_edit[97]),Buff[98],5);
syuki[50][2] = atoi(Buff[98]);
fprintf(fPressureData3, "%d ", syuki[50][2]);
syuki2[2] = syuki[50][2];
}
else{
for(i=0;i<48;i++){
LipPressure[i][2] = int(LipP[i][2]);
BlowPressure[i][2] = int(BlowP[i][2]);
}
shinpuku2[2] =(shinpuku[50][2]);
syuki2[2] =(syuki[50][2]);

}
fclose(fPressureData3);


//////////////チャンネル2に唇圧の出力////////////////
ch.ulChNo = 2;
DaOutputDA(hDeviceDA, 1, &ch, &NoteOffPressure);

//////////////チャンネル3にタンギングの出力/////////
ch.ulChNo = 3;
DaOutputDA(hDeviceDA, 1, &ch, &NoteOffPressure);

//////////////チャンネル4に送気圧を出力//////////////
ch.ulChNo = 4;
```

```
DaOutputDA(hDeviceDA, 1, &ch, &NoteOffPressure);

//////////////運指の開放////////////////////////////
nRetoff = DioOutputPoint(hDeviceDio, NoteOffFinger, 1, 32);

EndDialog(hTuningDlg,0);

tuningflag = 0;
bib3=0;
break;

}//switch(Msg)

return FALSE;

}//最初の括弧の閉じ



/*************************************************************
背景をＳＡＸの画像に
*************************************************************/
void ShowMyBMP(HWND hWnd,HDC hdc){
HDC hmdc;
HBITMAP hBitmap;
BITMAP bmp;
HINSTANCE hInstance;
int BMP_W, BMP_H;

hInstance = (HINSTANCE)GetWindowLong(hWnd, GWL_HINSTANCE);
hBitmap = LoadBitmap(hInstance, MAKEINTRESOURCE(IDB_BITMAP1));
GetObject(hBitmap, sizeof(BITMAP), &bmp);
BMP_W = (int)bmp.bmWidth;
BMP_H = (int)bmp.bmHeight;
hmdc = CreateCompatibleDC(hdc);
SelectObject(hmdc, hBitmap);
BitBlt(hdc, 0, 18, BMP_W, BMP_H, hmdc, 0, 0, SRCCOPY);
```

```
StretchBlt(hdc, 0, BMP_H, BMP_W / 2, BMP_H / 2, hmdc, 0, 0, BMP_W, BMP_H, SRCCOPY);
DeleteDC(hmdc);
DeleteObject(hBitmap);
return;
}

/************************************************************
MIDIファイル対応プログラム
その０　ファイル操作メイン関数
************************************************************/
void MIDIfileopen(HWND hWnd,HWND hCombo,HWND hButtonPlay,HWND hButtonPlay1,HWND
hStatic){
int i,j=0,fh,bytetk=0,c=0,count=0,wait=0;
int offset=0xe;
int ss=0;//ComboBoxのカーソル位置の設定用パラメータ
unsigned char buftk[SEEKSAX],midievent=0,noteornot=0;      //SEEKSAX=10000バイト
static char StrMsg[250];
bool s=0;
HANDLE hText= NULL;//出力用テキストのハンドル

//関数宣言
void checkhead(unsigned char *);
void openfile(HWND,HWND,HWND,HWND);
void MIDIBreak(int);
int checktrack(unsigned char *);
bool searchmeta(unsigned char *,int *);

openfile(hCombo,hButtonPlay,hButtonPlay1,hStatic); //ファイル名指定
ZeroMemory(buftk,sizeof(buftk)); //バッファの初期化

//ファイルのオープン
if((fh=open((LPSTR)mciDev.pFileName,O_BINARY|O_RDONLY))==-1)  //第1引数：ファイル名　第2引
数：フラグ
{
                                                    //バイナリデータで読み込み専用
MessageBox(hWnd,"ファイルがオープンできませんでした","オープンしません",MB_OK);
```

```
return;
}

//トラックごとにバッファ(MIDIPiece)に分ける
MIDIBreak(fh);

//分解能の検出(ヘッダ内から)
bunkai2=((MIDIPiece[0][12])<<8)+MIDIPiece[0][13];              //MIDIPiece[0][12]の値を1バイト分(8ビット分)ずらし
//MIDIPiece[0][13]の値を足すことで分解能を表現できる
for(i=1;i<22;i++){
while(1)
{
//トラック１から調や拍子を抜き出す(トラック・チャンクより)
if(j==0)
{
checktrack(MIDIPiece[i]);
}
searchmeta(&MIDIPiece[i][j],&j);
//楽器の名前をトラック２以降から検出　ｺﾝﾎﾞBOXに転送
if(0xc0<=MIDIPiece[i][j] && MIDIPiece[i][j]<=0xcf)                    //0xcnはプログラムチェンジ
{
wsprintf(StrMsg, "%s",GAKKINAME[MIDIPiece[i][j+1]]);       //j+1が楽器名
if(i!=1)
                        //トラック1には楽器名は入っていないため
{
SendMessage(hCombo, CB_INSERTSTRING, -1,(LPARAM)StrMsg);
}
if(MIDIPiece[i][j+1]==GAKKI) //GAKKI=66　テナーサックスのこと
{
SendMessage(hCombo, CB_SETCURSEL, i-1, 0);
}
j=0;
break;
}
j++;
```

```
}
if(MIDIPiece[i+1][0]!=0x4d) break;//つまりチャンクのMThd、MTrkがくるまで0x4dはMのこと
}

close(fh);
}

/************************************************************
ファイルの読み込み(コモンダイアログボックスによるオープン)
************************************************************/
void openfile(HWND hCombo,HWND hButtonPlay,HWND hButtonPlay1,HWND hStatic){
OPENFILENAME ofn;
lstrcpy((LPSTR)mciDev.pFileName, "*.mid");
ZeroMemory(&ofn, sizeof(ofn));

//オープンダイアログの設定
ofn.lStructSize = sizeof(ofn);
ofn.hwndOwner = hWnd;
ofn.lpstrFile = (LPSTR)mciDev.pFileName;
ofn.nMaxFile = 260;//sizeof(szFile);
ofn.lpstrFilter = "MIDIファイル¥0*.mid¥0";
ofn.nFilterIndex = 1;
ofn.lpstrFileTitle = NULL;
ofn.nMaxFileTitle = 0;
ofn.lpstrInitialDir = NULL;
ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;

if( !GetOpenFileName(&ofn) )
MessageBox(hWnd, "ファイルが選択されていません", "オープンしません", MB_OK |
MB_ICONSTOP);
else{
ZeroMemory(TrackSize,sizeof(TrackSize));
SendMessage(hCombo,CB_RESETCONTENT,0,0);//ｺﾝﾎﾞBOX内の初期化
SendMessage(hCombo, CB_INSERTSTRING, -1,(LPARAM)CHOOSE);
SendMessage(hCombo, CB_SETCURSEL,0,0);
SetWindowText(hStatic,"楽器を選択し、実行ボタンを押してください");
```

```
DestroyWindow(hButtonPlay);
DestroyWindow(hButtonPlay1);
tkcount=0;

Performance_i=0; //演奏ﾃﾞｰﾀ番号の初期化
}

}


/*****************************************************************************
その１　ヘッダチャンク確認
*****************************************************************************/
void checkhead(unsigned char *buf){
int bunkai1;//bunkai2はグローバル変数

if(*(buf)==0x4d && *(buf+1)==0x54 && *(buf+2)==0x68 && *(buf+3)==0x64){
bunkai1=(*(buf+12))<<8;
bunkai2=bunkai1+*(buf+13);
}
}

/*****************************************************************************
その２　トラックチャンク確認
*****************************************************************************/

int checktrack(unsigned char *buf){
int tklen[4],tklength=0,i,k;
if(*(buf)==0x4d && *(buf+1)==0x54 && *(buf+2)==0x72 && *(buf+3)==0x6b){

tkcount++;

//データ長を表示
for(i=0;i<4;i++){                                    //データ長が４バイトの可変長表示なので通常の表
示にする。
tklen[i]=*(buf+4+i);
```

```
tklen[i]=tklen[i]<<(24-8*i);
tklength+=tklen[i];
}
TrackSize[tkcount]=tklength;     //TrackSizeはMIDIファイルを作るときに使う。
return tklength+8;                                    //データ長に含まれない長さ８バイトを加える
}
return 0;
}


/*****************************************************************************
その３　メタイベント(ffで始まるイベント)
*****************************************************************************/
bool searchmeta(unsigned char *buf,int *i){
void hyousi(unsigned char *);
void chou(unsigned char *);
void tempo(unsigned char *);

if(*buf==0xff){
switch(*(buf+1)){
case 0x00:
*i+=4;//いまbuftk[i]はffなのでそのデータの終わりまで飛ばす。以下のiの操作も同様
break;
case 0x01:case 0x02:case 0x03:
case 0x04:case 0x05:case 0x06:
case 0x07:case 0x08:case 0x09:              //08・09はおそらく不要
*i+=*(buf+2)+2;//buf+2にイベントの長さがあり、その長さに含まれないff 01などの2バイト分を加えて、
とばす
break;
//調子、拍子、テンポは一度しか使わないから関数ではなくてもよい。
case 0x21:              //20じゃないの？
*i+=3;
break;
case 0x51:
tempo(buf);
*i+=5;
```

```
break;
case 0x58:
hyousi(buf);//拍子を探す
*i+=6;
break;
case 0x59:
chou(buf);
*i+=4;
break;
}//switch文の終了
return TRUE;
}//if文の終了
return FALSE;
}

/*********************************************************************
その４　テンポ
*********************************************************************/
void tempo(unsigned char *buf){
int tem[3],i;
double tempo2=0;//tempo1はグローバル変数

if(*(buf+2)==0x03){
for(i=0;i<3;i++){
tem[i]=*(buf+3+i);
tempo2+=tem[i]<<(16-8*i);
}
tempo2=60000000/tempo2+0.9;//0捨1入する　60000000は60μ秒のこと
tempo1=(int)tempo2;
}
}

/*********************************************************************************
その５　調
*********************************************************************************/
void chou(unsigned char *buf){
```

```
char iroha[46]="CesGesDes As Es　B　F　C　G　D　A　E　HFisCis";
int buf1;

if(*(buf+2)==0x02){
buf1=*(buf+3)+7;//*(buf+3)が０のとき　C　でCesが-7なのでCesを０にするため７をたす
buf1=buf1*3;//irohaが3文字で一組になっているから3倍する
if(*(buf+4)==0){
wsprintf(chou_chou,"%c%c%c　major",iroha[buf1],iroha[buf1+1],iroha[buf1+2]);
}else if(*(buf+4)==1){
wsprintf(chou_chou,"%c%c%c　minor",iroha[buf1],iroha[buf1+1],iroha[buf1+2]);
}
}
}

/*********************************************************************
その６　拍子
*********************************************************************/
void hyousi(unsigned char *buf){
int bunsi,bunbo;
if(*(buf+2)==0x4){
bunsi=*(buf+3);
bunbo=(int)pow((double)2,(double)*(buf+4));//2の*(buf+4)乗という形であらわされているため
wsprintf(hyousi_hyousi,"%d/%d 拍子",bunsi,bunbo);
}
}

/*********************************************************************
その７　信号の種類を検索
*********************************************************************/
unsigned char checkevent(unsigned char *buf,unsigned char midievent){
if(0x80<=*buf && *buf<=0x9f && *(buf+2)<=0x7f){
midievent=1;
}else if(0xa0<=*buf && *buf<=0xaf){
midievent=2;
}else if(0xb0<=*buf && *buf<=0xbf){
midievent=3;
```

```
}else if(0xc0<=*buf && *buf<=0xcf){
midievent=4;
}else if(0xd0<=*buf && *buf<=0xdf){
midievent=5;
}else if(0xe0<=*buf && *buf<=0xef){
midievent=6;
}
return midievent;
}

/*********************************************************************************
その８　ノートオン、ノートオフ
*********************************************************************************/
unsigned char checkonoff(unsigned char *buf,unsigned char *noteornot,HANDLE fp){
int count=0;
char StrMsg[250];
static bool check=FALSE;

void delta(unsigned char *buf,int *,HANDLE);
exnoto=0;
int i;//ベロシティ判別用

for(i=0;i<8;i++){
vel[i]=0;
}


if(remake==FALSE){
Performance_i=0;
remake=TRUE;
}

PerformanceData[Performance_i+1][0]=0;

if(*buf>=0x80){ //ランニングステータス終了
*noteornot=1;//ランニングステータスで動作していないもののノートオフの処理
```

```
wsprintf(StrMsg,"%02x%s",*(buf+1),"のノートオフ1");
if(*(buf+3)>0){
wsprintf(StrMsg,"ノートオフタンギング");
PerformanceData[Performance_i+1][2]=1;
}
write_text(StrMsg);
PerformanceData[Performance_i][1]=TotalTime-TmpTime; //「これまでの合計時間」から「」
PerformanceData[Performance_i+1][0]=-1;
wsprintf(StrMsg,"i=%d,音階=%d",Performance_i+1,PerformanceData[Performance_i+1][0],"ノートオフ2");
write_text(StrMsg);
nowi=Performance_i;//現在のﾉｰﾄが配列の何番目かを一時記憶しておく

TmpTime=TotalTime;
Performance_i++;
check=FALSE;
delta(buf+3,&count,fp);

return 0;
}else{

if(*(buf+1)!=0){                    //ノートオン・オフのベロシティが0x00ではない場合
if(check==FALSE){  //直前にノートオンがなかった場合

wsprintf(StrMsg,"%02x%s",*buf,"のノートオン1");
write_text(StrMsg);
PerformanceData[Performance_i][1]=TotalTime-TmpTime;
PerformanceData[Performance_i+1][0]=*buf;
wsprintf(StrMsg,"i=%d,音階=%d",Performance_i+1,PerformanceData[Performance_i+1][0],"ノートオン2");
write_text(StrMsg);
exnoto=1;

if(*(buf+1)<=0x7f && *(buf+1)>=0x71){
wsprintf(StrMsg,"ベロシティ範囲fff=%3d",*(buf+1));
write_text(StrMsg);
vel[0]=1;
}
```

```
else if(*(buf+1)<=0x70 && *(buf+1)>=0x61){
wsprintf(StrMsg,"ベロシティ範囲ff=%3d",*(buf+1));
write_text(StrMsg);
vel[1]=1;
}
else if(*(buf+1)<=0x60 && *(buf+1)>=0x51){
wsprintf(StrMsg,"ベロシティ範囲f=%3d",*(buf+1));
write_text(StrMsg);
vel[2]=1;
}
else if(*(buf+1)<=0x50 && *(buf+1)>=0x41){
wsprintf(StrMsg,"ベロシティ範囲mf=%3d",*(buf+1));
write_text(StrMsg);
vel[3]=1;
}
else if(*(buf+1)<=0x40 && *(buf+1)>=0x31){
wsprintf(StrMsg,"ベロシティ範囲mp=%3d",*(buf+1));
write_text(StrMsg);
vel[4]=1;
}
else if(*(buf+1)<=0x30 && *(buf+1)>=0x21){
wsprintf(StrMsg,"ベロシティ範囲p=%3d",*(buf+1));
write_text(StrMsg);
vel[5]=1;
}
else if(*(buf+1)<=0x20 && *(buf+1)>=0x11){
wsprintf(StrMsg,"ベロシティ範囲pp=%3d",*(buf+1));
write_text(StrMsg);
vel[6]=1;
}
else if(*(buf+1)<=0x10 && *(buf+1)>=0x01){
wsprintf(StrMsg,"ベロシティ範囲ppp=%3d",*(buf+1));
write_text(StrMsg);
vel[7]=1;
}
else{
```

```
wsprintf(StrMsg,"何か違う=%3d",*(buf+1));
write_text(StrMsg);
}
//ベロシティ判別

if(vel[0]==1)PerformanceData[Performance_i+1][8]=1;
else if(vel[1]==1)PerformanceData[Performance_i+1][8]=2;
else if(vel[2]==1)PerformanceData[Performance_i+1][8]=3;
else if(vel[3]==1)PerformanceData[Performance_i+1][8]=4;
else if(vel[4]==1)PerformanceData[Performance_i+1][8]=5;
else if(vel[5]==1)PerformanceData[Performance_i+1][8]=6;
else if(vel[6]==1)PerformanceData[Performance_i+1][8]=7;
else if(vel[7]==1)PerformanceData[Performance_i+1][8]=8;
else PerformanceData[Performance_i+1][8]=0;//ベロシティ送気圧へ

if(nowonkai==PerformanceData[Performance_i+1][0])PerformanceData[nowi][2]=1;//同音続く場合ﾀﾝｷﾞﾝｸﾞ
ON
if(tongingflag==1)PerformanceData[Performance_i+1][2]=1;//ﾀﾝｷﾞﾝｸﾞONの場合の値=1
nowonkai=PerformanceData[Performance_i+1][0];//現在の音階を一時記憶しておく

TmpTime=TotalTime;
Performance_i++;

check=TRUE;
}
}else{
if(check==TRUE && nowonkai==*buf){
wsprintf(StrMsg,"%02x%s",*buf,"のノートオフ3");
write_text(StrMsg);
PerformanceData[Performance_i][1]=TotalTime-TmpTime;
PerformanceData[Performance_i+1][0]=-1;
nowi=Performance_i;//現在のﾉｰﾄが配列の何番目かを一時記憶しておく

TmpTime=TotalTime;
Performance_i++;
check=FALSE;
```

```
}
}

delta(buf+2,&count,fp);
*noteornot=0;
return count;

}
}

/*************************************************************************
その９　デルタタイム
*************************************************************************/
void delta(unsigned char *buf,int *count,HANDLE fp){
int a=*buf,b=*(buf+1),c=*(buf+2);
double g;
char StrMsg[250];
if(a<=0x80){
wsprintf(StrMsg," デルタタイム%d",a);
write_text(StrMsg);
g=(double)(a*60*1000/(tempo1*bunkai2/1000));
wsprintf(StrMsg," %fμ秒¥n",g);
write_text(StrMsg);
TotalTime+=a*60*1000;
*count=0;
}else if(a>=0x80 && b<0x80){
a=a-0x80;
a=a<<7;
b=*(buf+1);
wsprintf(StrMsg," デルタタイム%d",a+b);
write_text(StrMsg);
g=(double)((a+b)*60*1000/(tempo1*bunkai2/1000));
wsprintf(StrMsg," %fμ秒¥r¥n",g);
write_text(StrMsg);
TotalTime+=(a+b)*60*1000;
*count=1;
```

```
}else if(a>=0x80 && b>=0x80){
a=a-0x80;
b=b-0x80;
a=a<<14;
b=b<<7;
wsprintf(StrMsg," デルタタイム%d",a+b+c);
write_text(StrMsg);
g=(double)((a+b+c)*60*1000/(tempo1*bunkai2/1000));
wsprintf(StrMsg," %fμ秒¥r¥n",g);
write_text(StrMsg);
TotalTime+=(a+b+c)*60*1000;
*count=2;
}
}

/*****************************************************************************
その10　コントロール・チェンジ・メッセージ
*****************************************************************************/
unsigned char checkcontrol(unsigned char *buf,unsigned char *controlornot,HANDLE fp){
int count=0;
char StrMsg[250];
static bool check=FALSE;
void delta(unsigned char *buf,int *,HANDLE);
int k;

for(k=0;k<4;k++){
expre[k]=3;//エクスプレッション判別初期化用
}

if(remake==FALSE){
Performance_i=0;
remake=TRUE;
}

if(*buf>=0x80){ //ランニングステータス終了
*controlornot=1;
```

```
return 0;
}else{

if(*buf==0x6e){ //ﾀﾝｷﾞﾝｸﾞの判別
if(*(buf+1)==0x7f){
tongingflag=1; //ﾀﾝｷﾞﾝｸﾞON
}else if(*(buf+1)==0x00){
tongingflag=0; //ﾀﾝｷﾞﾝｸﾞOFF
}

}else if(*buf==0x6f){ //ﾋﾞﾌﾞﾗｰﾄの種類の判別
if(*(buf+1)==0x7f){
vibflag=1; //送気圧変化
}else if(*(buf+1)==0x00){
vibflag=0; //唇押圧変化
}

}else if(*buf==0x01){ //ﾓｼﾞｭﾚｰｼｮﾝの判別
wsprintf(StrMsg,"%02x%s",*(buf+1),"のモジュレーション");
write_text(StrMsg);
PerformanceData[Performance_i][1]=TotalTime-TmpTime;
PerformanceData[Performance_i+1][0]=nowonkai;
PerformanceData[Performance_i+1][3]=*(buf+1);
PerformanceData[Performance_i+1][4]=nowrate;
PerformanceData[Performance_i+1][5]=nowdepth;
PerformanceData[Performance_i+1][9]=0;

nowmodul=PerformanceData[Performance_i+1][3];//現在のﾓｼﾞｭﾚｰｼｮﾝを一時記憶しておく
TmpTime=TotalTime;
Performance_i++;

}else if(*buf==0x4c){ //ﾋﾞﾌﾞﾗｰﾄ･ﾚｲﾄの判別
wsprintf(StrMsg,"%02x%s",*(buf+1),"のビブラート・レイト");
write_text(StrMsg);
PerformanceData[Performance_i][1]=TotalTime-TmpTime;
PerformanceData[Performance_i+1][0]=nowonkai;
```

```
PerformanceData[Performance_i+1][3]=nowmodul;
PerformanceData[Performance_i+1][4]=*(buf+1);
PerformanceData[Performance_i+1][5]=nowdepth;
PerformanceData[Performance_i+1][9]=nowexp;

nowrate=PerformanceData[Performance_i+1][4];//現在のﾋﾞﾌﾞﾗｰﾄ･ﾚｲﾄを一時記憶しておく
TmpTime=TotalTime;
Performance_i++;

}else if(*buf==0x4d){ //ﾋﾞﾌﾞﾗｰﾄ･ﾃﾞﾌﾟｽの判別
wsprintf(StrMsg,"%02x%s",*(buf+1),"のビブラート・デプス");
write_text(StrMsg);
PerformanceData[Performance_i][1]=TotalTime-TmpTime;
PerformanceData[Performance_i+1][0]=nowonkai;
PerformanceData[Performance_i+1][3]=nowmodul;
PerformanceData[Performance_i+1][4]=nowrate;
PerformanceData[Performance_i+1][5]=*(buf+1);
PerformanceData[Performance_i+1][9]=nowexp;

nowdepth=PerformanceData[Performance_i+1][5];//現在のﾋﾞﾌﾞﾗｰﾄ･ﾃﾞﾌﾟｽを一時記憶しておく
TmpTime=TotalTime;
Performance_i++;

}

else if(*buf==0x0b && exnoto==1){//エクスプレッション判別
wsprintf(StrMsg,"%02x%s",*(buf+1),"のエクスプレッション");
write_text(StrMsg);

if(*(buf+1)<=0x7f && *(buf+1)>=0x51){
wsprintf(StrMsg,"範囲強=%3d",*(buf+1));
write_text(StrMsg);
expre[0]=1;
}
else if(*(buf+1)<=0x50 && *(buf+1)>=0x31){
wsprintf(StrMsg,"範囲中=%3d",*(buf+1));
```

```
write_text(StrMsg);
expre[1]=1;
}
else if(*(buf+1)<=0x30 && *(buf+1)>=0x01){
wsprintf(StrMsg,"範囲弱=%3d",*(buf+1));
write_text(StrMsg);
expre[2]=1;
}
else {
wsprintf(StrMsg,"範囲おかしい=%3d",*(buf+1));
write_text(StrMsg);
//expre[3]=1;
}
if(expre[0]==1)PerformanceData[Performance_i+1][9]=1;
else if(expre[1]==1)PerformanceData[Performance_i+1][9]=2;
else if(expre[2]==1)PerformanceData[Performance_i+1][9]=3;
else PerformanceData[Performance_i+1][9]=0;

nowexp=PerformanceData[Performance_i+1][9];//現在のエクスプレッションを保存

PerformanceData[Performance_i][1]=TotalTime-TmpTime;
PerformanceData[Performance_i+1][0]=nowonkai;
PerformanceData[Performance_i+1][3]=0;
PerformanceData[Performance_i+1][4]=nowrate;
PerformanceData[Performance_i+1][5]=nowdepth;
PerformanceData[Performance_i+1][9]=nowexp;
TmpTime=TotalTime;
Performance_i++;
}

delta(buf+2,&count,fp);
*controlornot=0;
return count;

}
}
```

```
/*************************************************************************
その11　ピッチベンド
*************************************************************************/
unsigned char checkpitchbend(unsigned char *buf,unsigned char *pitchbendornot,HANDLE fp){
int count=0;
char StrMsg[250];
static bool check=FALSE;
void delta(unsigned char *buf,int *,HANDLE);
int a=*buf,b=*(buf+1);

if(remake==FALSE){
Performance_i=0;
remake=TRUE;
}

if(*buf>=0x80){ //ランニングステータス終了
*pitchbendornot=1;
return 0;
}else{

b=b<<7;

wsprintf(StrMsg,"%5d%s",a+b-8192,"のピッチベンド");
write_text(StrMsg);
PerformanceData[Performance_i][1]=TotalTime-TmpTime;
PerformanceData[Performance_i+1][0]=nowonkai;
PerformanceData[Performance_i+1][7]=a+b-8192;

TmpTime=TotalTime;
Performance_i++;

delta(buf+2,&count,fp);
*pitchbendornot=0;
return count;
```

```
}
}

/*****************************************************************************
実行ボタン メッセージ処理：初期化処理
*****************************************************************************/
void OnActionButton(HWND hWnd){
TermMciDev();
if(!InitMciDev()){
MessageBox(hWnd, "MCIデバイスオープン失敗", "MCITEST", MB_OK | MB_ICONERROR);
}
}

/*****************************************************************************
メインウィンドウの終了時の メッセージ処理 ： 終了処理
*****************************************************************************/
void OnDestroy(HWND hWnd){
int i;
char filename[100];

DaClose(hDeviceDA);   //DAﾎﾞｰﾄﾞのﾃﾞﾊﾞｲｽを閉じる

lstrcpy(filename,"test.mid");
TermMciDev();
for(i=0;i<=MakeMIDICount;i++){
if(i!=0)wsprintf(filename,"test%d.mid",i);
remove(filename);
}
PostQuitMessage(0);
}

/*****************************************************************************
MCIの再生 メッセージ処理：再生ボタン押し：演奏スタート
*****************************************************************************/
void OnPlayButtonDown(HWND hWnd){
void SendPerformanceData(void);
```

```
if(!PlayMciDev()){
MessageBox(hWnd, "再生にエラーが発生しました", "MCITEST", MB_OK | MB_ICONERROR);
return;
}
SendPerformanceData();
}
/*****************************************************************************
MCIの再生　メッセージ処理：MCI停止:ロボットの演奏のみスタート
*****************************************************************************/
void OnStopButtonDown(HWND hWnd){
void SendPerformanceData(void);
if(!PlayMciDev1()){
MessageBox(hWnd, "再生にエラーが発生しました", "MCITEST", MB_OK | MB_ICONERROR);
return;
}
SendPerformanceData();
}

/*****************************************************************************
* MCIデバイスのオープン
*****************************************************************************/
BOOL InitMciDev(){
MCI_OPEN_PARMS mciOpenParms;
MCIERROR            ret;
// 開くデバイスタイプによって指定する名前が違う
// wav   : waveaudio
// midi : sequencer
// cd     : cdaudio
ZeroMemory(&mciOpenParms, sizeof(MCI_OPEN_PARMS));
// デバイスタイプの指定
mciOpenParms.lpstrDeviceType = "sequencer";
mciOpenParms.lpstrElementName = (LPCSTR)mciDev.pFileName;
ret = mciSendCommand(NULL, MCI_OPEN, MCI_OPEN_TYPE | MCI_OPEN_ELEMENT,
(DWORD)&mciOpenParms);

if(ret != 0){
```

```
mciDev.isOpen = FALSE;
return FALSE;
}

mciDev.wDeviceID = mciOpenParms.wDeviceID;
mciDev.isOpen = TRUE;
return TRUE;
}

/*****************************************************************************
* MCIデバイスのクローズ
*****************************************************************************/
void TermMciDev(){
if(mciDev.isOpen){
mciSendCommand(mciDev.wDeviceID, MCI_CLOSE, 0, NULL);
mciDev.isOpen = FALSE;
}
}

/*****************************************************************************
* MCIデバイスを再生する
*****************************************************************************/
BOOL PlayMciDev(){
MCI_PLAY_PARMS mciPlayParms;
MCIERROR ret;

if(!mciDev.isOpen) return TRUE;
ZeroMemory(&mciPlayParms, sizeof(MCI_PLAY_PARMS));
mciPlayParms.dwFrom = 0;

if(!BS_PUSHBUTTON1){

ret = mciSendCommand(mciDev.wDeviceID, MCI_PLAY, MCI_FROM,(DWORD)&mciPlayParms);

if(ret != 0){// エラー発生したのでデバイスを閉じる
TermMciDev();
```

```
return FALSE;
}}

return TRUE;
}
/*****************************************************************************
* ロボットのみで演奏する
*****************************************************************************/
BOOL PlayMciDev1(){
MCI_PLAY_PARMS mciPlayParms;
MCIERROR ret;

if(!mciDev.isOpen) return TRUE;
ZeroMemory(&mciPlayParms, sizeof(MCI_PLAY_PARMS));
mciPlayParms.dwFrom = 0;


if(!BS_PUSHBUTTON2){

ret = mciSendCommand(mciDev.wDeviceID, MCI_STOP, MCI_FROM,(DWORD)&mciPlayParms);

if(ret != 0){// エラー発生したのでデバイスを閉じる
TermMciDev();
return FALSE;
}}

return TRUE;
}
/*******************************************
MIDIﾌｧｲﾙをトラックごとに分ける
*******************************************/
void MIDIBreak(int fh){
int i,j,k=14,c,bytetk;
unsigned char buftk[100000];
char tamesi[100]="tamesi";
```

```
//初期化
for(i=0;i<22;i++){
ZeroMemory(&MIDIPiece[i], 20000);
}

//ヘッダチャンクをMIDIPieceに入れる
read(fh,buftk,sizeof(buftk));       //buftkにfhの長さを格納
for(c=0;c<14;c++){                                    //ヘッダチャンクのcバイト目
MIDIPiece[0][c]=buftk[c];
if(c==11)(MIDIPiece[0][c])--;//あとでトラックを一つ減らすので１減らしておく
}

i=1;                                  //i=0のところはヘッダを入れるため
j=0;
lseek(fh,0,0);            //ファイルポインタを先頭から先頭へ移動

while(1){
bytetk=read(fh,buftk,sizeof(buftk));          //bytetkはfhのバイト数
if(bytetk==0)break;                                                    //ファイルの読み込み終了
for(c=0;c<bytetk;c++){

if(buftk[k]==0x00 && buftk[k+1]==0xff && buftk[k+2]==0x03 && buftk[k+3]==0x0c && i!=1){ //system
setup前の文字を見つけて、かつセットアップトラックでもない
while(1){
k++; //ひたすらk増やす
j++; //ひたすらjも増やす
if(buftk[k]==0x4d && buftk[k+1]==0x54 && buftk[k+2]==0x72 && buftk[k+3]==0x6b){ //トラックチャン
ク見つかったら
j=0; //さっきまで読み進めた分をなかったことにしてj=0から数え直し
c=0; //for文をやり直し
(MIDIPiece[0][11])--; //なかったことにしたトラックをヘッダチャンクからひいておく
break;
}
}
}
if(buftk[k+1]==0xc0 && i==1 && buftk[k-4]!=0xff){    //MuseScore対策トラック１でプログラムチェン
```

```
ジを発見した場合
MIDIPiece[1][j]=buftk[k]; MIDIPiece[1][j+1]=buftk[k+1]; MIDIPiece[1][j+2]=buftk[k+2];
MIDIPiece[1][j+3]=0; //プログラムチェンジまで入力しておく
MIDIPiece[1][j+4]=255; MIDIPiece[1][j+5]=47; MIDIPiece[1][j+6]=0;  //トラックエンドでトラック１終わり
i=2; //次のトラック（２）へ
MIDIPiece[2][0]=77; MIDIPiece[2][1]=84; MIDIPiece[2][2]=114; MIDIPiece[2][3]=107; //トラックチャンク　10進数で書いたけど4D 54 72 6Bのこと
MIDIPiece[2][4]=0; MIDIPiece[2][5]=0; MIDIPiece[2][6]=0; MIDIPiece[2][7]=(MIDIPiece[1][7])-j+8; //トラックサイズ入力
MIDIPiece[2][8]=0; //プログラムチェンジ前のデルタタイムまで入力
MIDIPiece[1][7]=j-1;  //トラックサイズを変更(トラックチャンク8個ぶんを引いた値)
(MIDIPiece[0][11])++; //ヘッダチャンクのトラック数を１増やす
k++; //kを１つ読み進める
j=9;  //9から読み始める　以下通常運行
}
MIDIPiece[i][j]=buftk[k];
j++;
if(buftk[k+1]==0x4d && buftk[k+2]==0x54 &&
buftk[k+3]==0x72 && buftk[k+4]==0x6b){
i++;//次のトラックへ
j=0;
}
k++;
}
}

lseek(fh,0,0);//このファイル(fh)のカーソルを最初にもどす
}

/*********************************************************
演奏用MIDIﾌｧｲﾙ作成
*********************************************************/
void MakeMIDIFile(HWND hCombo){              //実行ボタンを押した後、再生ボタンを押す前
void ReadPlayTrack(int);
DWORD dwWritten;
```

```
char StrCombo[30];
char StrComboCopy[30];
HANDLE hMIDIFile;
int DelTrack,i;

static char filename[100];
lstrcpy(filename,"test.mid");          //文字列をバッファにコピー
//↓test.midが存在する場合test1.mid,test2.midなどにする
while((hMIDIFile = CreateFile(
filename,                              //ファイル名
GENERIC_READ | GENERIC_WRITE,     //ファイルの読み書き
0,
NULL,
CREATE_ALWAYS,                         //ファイルを作成、指定ファイルがある場合上書き
FILE_ATTRIBUTE_NORMAL,                 //ファイルの属性指定なし
NULL
))==INVALID_HANDLE_VALUE)          //関数の失敗
{
MakeMIDICount++;                       //初期値=0
wsprintf(filename,"test%d.mid",MakeMIDICount);
}
//ここまで

GetDlgItemText(hWnd,ID_COMBO,StrCombo,sizeof(StrCombo));      //StrCombo:コンボボックスで選択
した楽器

lstrcpy((LPSTR)mciDev.pFileName,filename); //再生するMIDIファイル名を指定
for(DelTrack=0;DelTrack<21;DelTrack++)
{
SendMessage(hCombo, CB_SETCURSEL, DelTrack, 0);
            //DelTrack番目の楽器名を選択
GetDlgItemText(hWnd, ID_COMBO, StrComboCopy, sizeof(StrComboCopy));          //選択している楽器
名をStrComboCopyに代入
if(strcmp(StrCombo,StrComboCopy)==0)break;
            //StrComboとStrComboCopyが等しい時
}
```

```
//このときのDelTrackが削除するトラックの番号
DelTrack++;//ヘッダの分足してる
TrackSize[0]=6;//ヘッダチャンクの長さをTrackSize[i-1+8]にあわせて１４にするため

for(i=0;i<=21;i++){
if(i==DelTrack)i++;
if(MIDIPiece[i][0]!=0x4d)break;                    //ヘッダおよびトラックチャンクの最初が4dで始まらない場合
//つまり，MIDIPiece[i][0]が存在しない場合
WriteFile(hMIDIFile,&MIDIPiece[i],TrackSize[i]+8,&dwWritten,NULL);  //+8はそれぞれのチャンクのキャラクター分？
}

CloseHandle(hMIDIFile);
ReadPlayTrack(DelTrack);
}

/*****************************************************************
test.txtに書き込む
*****************************************************************/
void write_text(const char* a){

fprintf(fp, "%s", a);

}

/*****************************************************************
演奏するトラックを読み込む
*****************************************************************/
void ReadPlayTrack(int PlayTrack){
unsigned char checkevent(unsigned char *,unsigned char midievent);
unsigned char checkonoff(unsigned char *,unsigned char * ,HANDLE);
unsigned char checkcontrol(unsigned char *,unsigned char * ,HANDLE);
unsigned char checkpitchbend(unsigned char *,unsigned char * ,HANDLE);

void delta(unsigned char *buf,int *,HANDLE);
```

```
bool searchmeta(unsigned char *,int *);
int i,midievent=0,del,c=0;
unsigned char noteornot=0,controlornot=0,pitchbendornot=0;
int count;
char StrMsg[100];

tongingflag=0; //ﾀﾝｷﾞﾝｸﾞOFFに初期化
nowonkai=0; //現在の音階を初期化
nowi=0;//現在の音階の位置を初期化
nowmodul=0;nowrate=64;nowdepth=64;//ﾋﾞﾌﾞﾗｰﾄに関する値を初期化
nowexp=0;

//トラック検索のループ開始（ただし初めの8ﾊﾞｲﾄは飛ばす）
for(i=8;i<200000;i++){
wsprintf(StrMsg,"%02x %02x %02x %02x¥r¥n",MIDIPiece[PlayTrack][i],MIDIPiece[PlayTrack][i+1],MIDIP
iece[PlayTrack][i+2],MIDIPiece[PlayTrack][i+3]);
write_text(StrMsg);
//「全体設」の検出
if(MIDIPiece[PlayTrack][i+10]==0x91 && MIDIPiece[PlayTrack][i+11]==0x53 &&
MIDIPiece[PlayTrack][i+12]==0x91 && MIDIPiece[PlayTrack][i+13]==0xcc &&
MIDIPiece[PlayTrack][i+14]==0x90 && MIDIPiece[PlayTrack][i+15]==0xdd){
wsprintf(StrMsg,"全体設定　発見");
write_text(StrMsg);
while(1){
i++;
wsprintf(StrMsg,"%02x %02x %02x %02x¥r¥n",MIDIPiece[PlayTrack][i],MIDIPiece[PlayTrack][i+1],MIDIP
iece[PlayTrack][i+2],MIDIPiece[PlayTrack][i+3]);
write_text(StrMsg);
if(MIDIPiece[PlayTrack][i]==0x4d && MIDIPiece[PlayTrack][i+1]==0x54 &&
MIDIPiece[PlayTrack][i+2]==0x72 && MIDIPiece[PlayTrack][i+3]==0x6b){
i=i+8;
break;
}
}
}
//トラックの終了(ff 2f 00)を確認
```

```
if(MIDIPiece[PlayTrack][i]==0xff && MIDIPiece[PlayTrack][i+1]==0x2f &&
MIDIPiece[PlayTrack][i+2]==0)break;

if(searchmeta(&MIDIPiece[PlayTrack][i],&i)==TRUE)delta(&MIDIPiece[PlayTrack][i+1],&count,fp);

//ランニングステータス対応case文
switch(midievent){
case 0:
midievent=checkevent(&MIDIPiece[PlayTrack][i],midievent);
break;

case 1://ノートオン・オフ
del=checkonoff(&MIDIPiece[PlayTrack][i],&noteornot,fp);//ﾃﾞﾙﾀﾀｲﾑ2byteなら1,3byteなら2を返す(delに)
if(noteornot==0){
i+=(2+del);
}else if(noteornot==1){          //ランニングステータス終わり
i+=(3+del);                                          //次のループのとき同じのを読み込むため
}
midievent=0;//またMIDIイベントの種類検索に戻るため
break;

case 2://An ポリフォニックキープレッシャー
while(1){//ﾗﾝﾆﾝｸﾞｽﾃｰﾀｽ終了までﾙｰﾌﾟ(ﾃﾞﾙﾀﾀｲﾑは検出)
i+=2;
delta(&MIDIPiece[PlayTrack][i],&count,fp);
write_text("¥r¥n");
i+=count;
if(MIDIPiece[PlayTrack][i+1]<0x80)i++;
else break;
}
midievent=0;
break;

case 3://Bn コントロールチェンジ
del=checkcontrol(&MIDIPiece[PlayTrack][i],&controlornot,fp);
if(controlornot==0){
```

```
i+=(2+del);
}else if(controlornot==1){//ランニングステータス終わり
i--;
}
midievent=0;//またMIDIイベントの種類検索に戻るため
break;

case 4://Cn 楽器指定 プログラムチェンジ
while(1){
i+=1;count=0;
delta(&MIDIPiece[PlayTrack][i],&count,fp);
write_text("¥r¥n");
i+=count;
if(MIDIPiece[PlayTrack][i+1]<=0x80)i++;
else break;
}
midievent=0;
break;

case 5://Dn チャンネルプレッシャー
while(1){
i+=1;count=0;
delta(&MIDIPiece[PlayTrack][i],&count,fp);
write_text("¥r¥n");
i+=count;
if(MIDIPiece[PlayTrack][i+1]<=0x80)i++;
else break;
}
midievent=0;
break;

case 6://En ピッチベンド pitchbend
del=checkpitchbend(&MIDIPiece[PlayTrack][i],&pitchbendornot,fp);
if(pitchbendornot==0){
i+=(2+del);
}else if(pitchbendornot==1){//ランニングステータス終わり
```

```
i--;
}
midievent=0;
break;

}//switch文の終了

}//for分の終了

fclose(fp);
}

float Data[2];
/*************************************************************************
実機にデータを送る
*************************************************************************/
void SendPerformanceData(){
void wait(double,int,bool*);
int j=0,NoteOffPressure[3];
NoteOffPressure[0]=(int)(-((2+10)*4096/20+0.5));//唇押え圧
NoteOffPressure[1]=(int)((1+10)*4096/20+0.5);//送気圧
NoteOffPressure[2]=(int)((0+10)*4096/20+0.5);//タンギング
int onkai,i,k,g,n,nowonkai=0;
long x,xPoint=0; //ﾋﾞﾌﾞﾗｰﾄ用のﾙｰﾌﾟに使用
int lip,blow;
int tong; //DAボードに出力する値
int forte,piano; //強弱用　mfの時のblowの値が基本値
double lipVBase[3]; //lipの初期値（中央値）
double VibRate,VibDepth;//ﾋﾞﾌﾞﾗｰﾄの周波数と振幅
bool PlayCheck=TRUE;
MCI_STATUS_PARMS mciStatusParms;
MCIERROR ret;
FILE *fpre;
fpre=fopen("pressure.txt","w");

bib=0;//初期化
```

```
bib2=0;
dina2=5;//初期化
for(n=0;n<6;n++){
tap[n]=0;
taptime[n]=0;
}

tap[0]=60000/tempo1; //テンポデフォルト BPM=120なら500(ミリ秒)が保存されるように

PerformanceData[0][0]=-1;
DaStopSampling(hDeviceDA);
tempodef=tempo1;

//演奏のループ開始
for(i=0;i<10000;i++){
if(i==0){//mciとの始まりの音のズレを修正するため
mciStatusParms.dwItem = MCI_STATUS_MODE;
ret=mciSendCommand(mciDev.wDeviceID, MCI_STATUS, MCI_STATUS_ITEM,
(DWORD)&mciStatusParms);//mciの現在のモードを取得中
if ((DWORD)mciStatusParms.dwReturn != MCI_MODE_PLAY){//もしmciのモードがまだプレイモード
ではなかった時
for(g=1;g<1000;g++){
wait(1,0,&PlayCheck);//1ミリ秒待機でループを続ける
mciStatusParms.dwItem = MCI_STATUS_MODE;
ret=mciSendCommand(mciDev.wDeviceID, MCI_STATUS, MCI_STATUS_ITEM,
(DWORD)&mciStatusParms);
fprintf(fpre,"MCI準備中演奏待機中¥n");
if((DWORD)mciStatusParms.dwReturn == MCI_MODE_PLAY || (DWORD)mciStatusParms.dwReturn ==
MCI_MODE_STOP)break;//もしプレイモードになったらループから抜ける
}
                                                                    //MCI_MODE_STOP
を入れたのは「ロボット演奏のみ」ボタンを押した際にはすぐに抜けるようにするため
}
fprintf(fpre,"全て開始¥n");
}
```

```
onkai=PerformanceData[i][0]-34-SCont;//一番低い音にあわせる

if(onkai==-36-SCont){
//ﾉｰﾄｵﾝの出力（0V出す命令）

}else{
//ピッチベンドによる唇押え圧変化
for(k=0;k<3;k++){
lipVBase[k]=LipPressure[onkai][k]/100+((double)PerformanceData[i][7]/30000);//100で割っているのはチューニングダイアログの値を100で割らないで送り、小数点以下を表示するため
clipVBase[k]=LipPressure[onkai][k];
}
if(onkai==-35-SCont)onkai=nowonkai;
nowonkai=onkai;
//指
if(DioOutputPoint(hDeviceDio,FingerPosition[onkai],1,32)!=0)MessageBox(hWnd,"運指がない
","",MB_OK);

//ﾀﾝｷﾞﾝｸﾞ時間分早めに切り上げる値にする
if(PerformanceData[i][2]==1)PerformanceData[i][1]-=tongingtime*120*bunkai2;

//ﾋﾞﾌﾞﾗｰﾄをする場合
if(PerformanceData[i][3]!=0){

fprintf(fpre,"モジュレーション通ります＝%d¥n",PerformanceData[i][3]);
bib=1;
for(k=0;k<3;k++){
clipVBase[k]=LipPressure[onkai][k];
}
wait(PerformanceData[i][1],i,&PlayCheck);//待ち時間

fprintf(fpre," shinpuku= %d¥n",shinpuku);
fprintf(fpre," syuki= %d¥n",syuki);
fprintf(fpre," sec_pc= %d¥n",sec_pc);
fprintf(fpre,"ﾘｯﾌﾟ V= %lf",dlipVBase);
fprintf(fpre," lip= %d¥n",clip);
```

```
fprintf(fpre,"dina= %d¥n",dina);
fprintf(fpre," pressureonkai= %lf¥n pressureonkai2= %lf¥n onkai=%d¥n
performance=%d¥n",BlowPressure[onkai][dina],LipPressure[onkai][dina],onkai,PerformanceData[i][0]);
fprintf(fpre," perfomance2= %d¥n",PerformanceData[i][7]);


if(PerformanceData[i][0]==-1){
tong=(int)((4+10)*4096/20+0.5);
ch.ulChNo=3;
DaOutputDA(hDeviceDA,1,&ch,&tong);
fprintf(fpre,"lipstop=%d, ﾘｯﾌﾟ休符i=%d¥n",DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure[0]),i);
}
else{
fprintf(fpre,"lip2=%d, dalip=%d, ﾘｯﾌﾟi=%d,音階
=%d¥n",&clip,DaOutputDA(hDeviceDA,1,&ch,&clip),i,PerformanceData[i][0]);
}//チャンネル2に唇圧の出力

if(PerformanceData[i][0]==-1){
fprintf(fpre,"blowstop=%d¥n",DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure[1]));
}
else{
fprintf(fpre,"blow=%d, blow2=%d, dablow=%d, 音階
=%d¥n",blow,&blow,DaOutputDA(hDeviceDA,1,&ch,&blow),PerformanceData[i][0]);
}//チャンネル4に送気圧を出力
if(PerformanceData[i][0]==-1){
tong=(int)((0+10)*4096/20+0.5);
ch.ulChNo=3;
DaOutputDA(hDeviceDA,1,&ch,&tong);
}
}

else if(PerformanceData[i][9]!=0){
fprintf(fpre,"エクス通ります＝%d¥n",PerformanceData[i][9]);
//↓エクスプレッション判別
if(PerformanceData[i][9]==1){ //強
```

```
fprintf(fpre,"expre0¥n");
if(dina2==0){
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
}
else if(dina2==2){
dina=2;
dina2=5;
}
else dina=0;
}
else if(PerformanceData[i][9]==2){ //中

fprintf(fpre,"expre1¥n");
if(dina2==0){
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
}
else if(dina2==2){
dina=2;
dina2=5;
}
else dina=1;
}
else if(PerformanceData[i][9]==3){ //弱

fprintf(fpre,"expre2¥n");
if(dina2==0){
```

```
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
}
else if(dina2==2){
dina=2;
dina2=5;
}
else dina=2;
}
else {

fprintf(fpre,"expre3¥n");
if(dina2==0){
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
}
else if(dina2==2){
dina=2;
dina2=5;
}

}

lip=-1*(int)(((double)lipVBase[dina]+10)*4096/20+0.5);//4096は0xfff、+10は0～20にするため
blow=(int)(((double)BlowPressure[onkai][dina]/100+10)*4096/20+0.5);//20で割るのは±10で20Vあるから。
//(double)を書き加え小数点以下まで出力させる

fprintf(fpre,"ﾘｯﾌﾟ V= %lf",lipVBase);
```

```
fprintf(fpre," lip= %d¥n",lip);
fprintf(fpre,"dina= %d¥n",dina);
fprintf(fpre," pressureonkai= %lf¥n pressureonkai2= %lf¥n onkai=%d¥n
performance=%d¥n",BlowPressure[onkai][dina],LipPressure[onkai][dina],onkai,PerformanceData[i][0]);
fprintf(fpre," perfomance2= %d¥n",PerformanceData[i][7]);

if(PerformanceData[i][0]==-1){
bib=0;
bib2=0;
tong=(int)((4+10)*4096/20+0.5);
ch.ulChNo=3;
DaOutputDA(hDeviceDA,1,&ch,&tong);
fprintf(fpre,"lipstop=%d, ﾘｯﾌﾟ休符i=%d¥n",DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure[0]),i);
}
else{
ch.ulChNo=2;
DaOutputDA(hDeviceDA,1,&ch,&lip);
fprintf(fpre,"lip2=%d, dalip=%d, ﾘｯﾌﾟi=%d,音階
=%d¥n",&lip,DaOutputDA(hDeviceDA,1,&ch,&lip),i,PerformanceData[i][0]);
}//チャンネル2に唇圧の出力

if(PerformanceData[i][0]==-1){
fprintf(fpre,"blowstop=%d¥n",DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure[1]));
}
else{
ch.ulChNo=4;
DaOutputDA(hDeviceDA,1,&ch,&blow);
fprintf(fpre,"blow=%d, blow2=%d, dablow=%d, 音階
=%d¥n",blow,&blow,DaOutputDA(hDeviceDA,1,&ch,&blow),PerformanceData[i][0]);
}//チャンネル4に送気圧を出力
wait(PerformanceData[i][1],i,&PlayCheck);//待ち時間
if(PerformanceData[i][0]==-1){
tong=(int)((0+10)*4096/20+0.5);
ch.ulChNo=3;
DaOutputDA(hDeviceDA,1,&ch,&tong);
```

```
}
}
else{
//普通のﾉｰﾄｵﾝの出力
fprintf(fpre,"ぶろう＝%d",blow);
forte=4096-blow;
piano=blow-2048;

//ベロシティ送気圧変化用
if(PerformanceData[i][8]==1){ //fff

fprintf(fpre,"pafo1¥n");
if(dina2==0){
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
}
else if(dina2==2){
dina=2;
dina2=5;
}
else dina=0;
}
else if(PerformanceData[i][8]==2){ //ff

fprintf(fpre,"pafo2¥n");
if(dina2==0){
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
```

```
}
else if(dina2==2){
dina=2;
dina2=5;
}
else dina=0;
}
else if(PerformanceData[i][8]==3){ //f

fprintf(fpre,"pafo3¥n");
if(dina2==0){
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
}
else if(dina2==2){
dina=2;
dina2=5;
}
else dina=0;
}
else if(PerformanceData[i][8]==4){ //mf

fprintf(fpre,"pafo4¥n");
if(dina2==0){
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
}
else if(dina2==2){
```

```
dina=2;
dina2=5;
}
else dina=1;
}
else if(PerformanceData[i][8]==5){ //mp

fprintf(fpre,"pafo5¥n");
if(dina2==0){
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
}
else if(dina2==2){
dina=2;
dina2=5;
}
else dina=1;
}
else if(PerformanceData[i][8]==6){ //p

fprintf(fpre,"pafo6¥n");
if(dina2==0){
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
}
else if(dina2==2){
dina=2;
dina2=5;
```

```
}
else dina=2;
}
else if(PerformanceData[i][8]==7){ //pp

fprintf(fpre,"pafo7¥n");
if(dina2==0){
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
}
else if(dina2==2){
dina=2;
dina2=5;
}
else dina=2;
}
else if(PerformanceData[i][8]==8){ //ppp

fprintf(fpre,"pafo8¥n");
if(dina2==0){
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
}
else if(dina2==2){
dina=2;
dina2=5;
}
else dina=2;
```

```
}
else
{
fprintf(fpre,"pafo9¥n");
if(dina2==0){
dina=0;
dina2=5;
}
else if(dina2==1){
dina=1;
dina2=5;
}
else if(dina2==2){
dina=2;
dina2=5;
}

}

lip=-1*(int)(((double)lipVBase[dina]+10)*4096/20+0.5);//4096は0xfff、+10は0～20にするため
blow=(int)(((double)BlowPressure[onkai][dina]/100+10)*4096/20+0.5);//20で割るのは±10で20Vあるから。
//(double)を書き加え小数点以下まで出力させる

fprintf(fpre,"ﾘｯﾌﾟ V= %lf",lipVBase);
fprintf(fpre," lip= %d¥n",lip);
fprintf(fpre,"dina= %d¥n",dina);
fprintf(fpre," pressureonkai= %lf¥n pressureonkai2= %lf¥n onkai=%d¥n
performance=%d¥n",BlowPressure[onkai][dina],LipPressure[onkai][dina],onkai,PerformanceData[i][0]);
fprintf(fpre," perfomance2= %d¥n",PerformanceData[i][7]);


if(PerformanceData[i][0]==-1){
bib=0;
bib2=0;
tong=(int)((4+10)*4096/20+0.5);
ch.ulChNo=3;
```

```
DaOutputDA(hDeviceDA,1,&ch,&tong);
}
else{
ch.ulChNo=2;
DaOutputDA(hDeviceDA,1,&ch,&lip);
fprintf(fpre,"lip2=%d, dalip=%d, ﾘｯﾌﾟ i=%d,音階
=%d¥n",&lip,DaOutputDA(hDeviceDA,1,&ch,&lip),i,PerformanceData[i][0]);
}//チャンネル2に唇圧の出力

if(PerformanceData[i][0]==-1){

}
else{
ch.ulChNo=4;
DaOutputDA(hDeviceDA,1,&ch,&blow);
fprintf(fpre,"blow=%d, blow2=%d, dablow=%d, 音階
=%d¥n",blow,&blow,DaOutputDA(hDeviceDA,1,&ch,&blow),PerformanceData[i][0]);
}//チャンネル4に送気圧を出力
wait(PerformanceData[i][1],i,&PlayCheck);//待ち時間
if(PerformanceData[i][0]==-1){
tong=(int)((0+10)*4096/20+0.5);
ch.ulChNo=3;
DaOutputDA(hDeviceDA,1,&ch,&tong);
}

}

//ﾀﾝｷﾞﾝｸﾞをする場合
if(PerformanceData[i][2]==1){
tong=(int)((4+10)*4096/20+0.5);
ch.ulChNo=3;
DaOutputDA(hDeviceDA,1,&ch,&tong);//チャンネル3にタンギングONの出力
wait(tongingtime*tempo1*bunkai2,0,&PlayCheck);//ﾀﾝｷﾞﾝｸﾞの押え中の待ち時間
tong=(int)((0+10)*4096/20+0.5);
ch.ulChNo=3;
DaOutputDA(hDeviceDA,1,&ch,&tong);//チャンネル3にタンギングOFFの出力
```

```
}
fprintf(fpre," 現在のテンポ= %lf¥n",tempo1);
}//if(onkai==-35-SCont)の終り

bib=0;
bib2=0;
if(PlayCheck==FALSE)break;
if(PerformanceData[i+1][0]==0 && PerformanceData[i+2][0]==0 && PerformanceData[i+3][0]==0)break;

}//for演奏のループ終了

fprintf(fpre," xPoint= %d¥n",xPoint);


//自動演奏終了後の処理
tempo1=tempodef;
SetWindowText(hStatic,"全停止中");
if(DioOutputPoint(hDeviceDio,NoteOffFinger,1,32)!=0)MessageBox(hWnd,"終了","",MB_OK);
ch.ulChNo=2;
DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure[0]);
ch.ulChNo=3;
DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure[2]);
ch.ulChNo=4;
DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure[1]);

remake=FALSE;
fclose(fpre);
}

/*********************************************************************
処理待ち　ミリ秒待つ abe減速、加速も追加ver
*********************************************************************/
void wait(double Second,int now,bool* PlayCheck){
MSG msg;
DWORD timer,timer2;
```

```
int NoteOffPressure=0xfff/2;
MCI_PLAY_PARMS mciPlayParms;
MCIERROR ret;
int accel,stop=0;
int n;
int k;

LARGE_INTEGER nFreq, nBefore, nAfter, nNow1, nNow2, nClck1,nClck2,nClck3,nClck4,nClck5;

//変数の初期化
memset(&nFreq,     0x00, sizeof nFreq);
memset(&nBefore, 0x00, sizeof nBefore);
memset(&nAfter,    0x00, sizeof nAfter);
memset(&nNow1,     0x00, sizeof nNow1);
memset(&nNow2,     0x00, sizeof nNow2);

memset(&nClck1,    0x00, sizeof nClck1); //クリック時のカウンタ
memset(&nClck2,    0x00, sizeof nClck2);
memset(&nClck3,    0x00, sizeof nClck3);
memset(&nClck4,    0x00, sizeof nClck4);
memset(&nClck5,    0x00, sizeof nClck5);

FILE *flog;
flog=fopen("SaxPlayLog.txt","a");

accel=0;
*PlayCheck=TRUE;
QueryPerformanceFrequency(&nFreq);
QueryPerformanceCounter(&nBefore);
QueryPerformanceCounter(&nNow1);

timer=GetTickCount();

SetWindowText(hStatic,"Escで全終了、TabでMCIのみ終了");

for(;;){
```

```
if(PeekMessage(&msg,NULL,0,0,PM_REMOVE)!=0){
//演奏中の処置
if(msg.message==WM_KEYDOWN && msg.wParam==0x09){//Tabキーで処理mciデバイスのみ停止
ret = mciSendCommand(mciDev.wDeviceID, MCI_STOP, MCI_FROM,(DWORD)&mciPlayParms);
fclose(flog);
SetWindowText(hStatic,"MCIのみ停止中");}

if(msg.message==WM_KEYDOWN && msg.wParam==0x4c){//Lキーで演奏音を強に対応
dina2=0;
SetWindowText(hStatic,"音量強に変更");
fprintf(flog,"音量強に変更¥n");
}

if(msg.message==WM_KEYDOWN && msg.wParam==0x4d){//Mキーで演奏音を中に対応
dina2=1;
SetWindowText(hStatic,"音量中に変更");
fprintf(flog,"音量中に変更¥n");
}

if(msg.message==WM_KEYDOWN && msg.wParam==0x53){//Sキーで演奏音を弱に対応
dina2=2;
SetWindowText(hStatic,"音量弱に変更");
fprintf(flog,"音量弱に変更¥n");
}

if(msg.message==WM_KEYDOWN && msg.wParam==0x56){//Vキーで演奏音をビブラートにする
bib2=1;
nowmodul=127;
SetWindowText(hStatic,"自分でビブラート発動");
fprintf(flog,"自分でビブラート¥n");
}

if(msg.message==WM_KEYDOWN && msg.wParam==0x26){//↑キー：高速化
if(tempo1<tempodef*2){
tempo1++;
accel=1;
```

```
QueryPerformanceCounter(&nNow2);
TrueTotalTime+=((DWORD)((nNow2.QuadPart - nNow1.QuadPart) * 1000 / nFreq.QuadPart));
fprintf(flog," 加速しました　テンポ= %lf¥n TotalTime= %d¥n",tempo1,TrueTotalTime);
SetWindowText(hStatic,"加速しました");
nNow1.QuadPart=nNow2.QuadPart;
}
else{
QueryPerformanceCounter(&nNow2);
TrueTotalTime+=((DWORD)((nNow2.QuadPart - nNow1.QuadPart) * 1000 / nFreq.QuadPart));
fprintf(flog," これ以上加速できません　テンポ= %lf¥n TotalTime= %d¥n",tempo1,TrueTotalTime);
SetWindowText(hStatic,"これ以上加速できません"); //加速限界追加,元のテンポの2倍
nNow1.QuadPart=nNow2.QuadPart;
}
}
if(msg.message==WM_KEYDOWN && msg.wParam==0x28){//↓キー：低速化
if(tempo1>tempodef/2){
tempo1--;
accel=2;
QueryPerformanceCounter(&nNow2);
TrueTotalTime+=((DWORD)((nNow2.QuadPart - nNow1.QuadPart) * 1000 / nFreq.QuadPart));
fprintf(flog," 減速しました　テンポ= %lf¥n TotalTime= %d¥n",tempo1,TrueTotalTime);
SetWindowText(hStatic,"減速しました");}
else{
QueryPerformanceCounter(&nNow2);
TrueTotalTime+=((DWORD)((nNow2.QuadPart - nNow1.QuadPart) * 1000 / nFreq.QuadPart));
fprintf(flog," これ以上減速できません　テンポ= %lf¥n TotalTime= %d¥n",tempo1,TrueTotalTime);
nNow1.QuadPart=nNow2.QuadPart;
SetWindowText(hStatic,"これ以上減速できません");  //減速限界の追加,元のテンポの1/2
}
}

/* クリックでテンポ取得テスト */
if(msg.message==WM_KEYDOWN && msg.wParam==0x25){//←キー：テンポ取得　クリックにしたかったけどとりあえず手馴れてるキーで
tapCount++;
if(taptime[0]==0){
```

```
QueryPerformanceCounter(&nClck1);
taptime[0]=((DWORD)((nClck1.QuadPart)* 1000000 / nFreq.QuadPart));    //キー押下タイミング(1回目)を保存(ミリ秒)
SetWindowText(hStatic,"1回目");
}
else if(taptime[1]==0){
QueryPerformanceCounter(&nClck2);
taptime[1]=((DWORD)((nClck2.QuadPart)* 1000000 / nFreq.QuadPart));    //キー押下タイミング(2回目)を保存(ミリ秒)
tap[1]=(taptime[1]/1000)-(taptime[0]/1000);                                                    //2回目と1回目の間隔
SetWindowText(hStatic,"2回目");
tempo1=(double)(60000/tap[1]);                              //即時テンポ変更追加
fprintf(flog,"タップ間隔= %lfミリ秒¥n",tap[1]);
}
else if(taptime[2]==0){
QueryPerformanceCounter(&nClck3);
taptime[2]=((DWORD)((nClck3.QuadPart)* 1000000 / nFreq.QuadPart));
tap[2]=(taptime[2]/1000)-(taptime[1]/1000);
tap[0]=(tap[1]+tap[2])/2;
SetWindowText(hStatic,"3回目");
tempo1=(double)(60000/tap[0]);
fprintf(flog,"タップ間隔= %lfミリ秒¥n",tap[2]);
}
else if(taptime[3]==0){
QueryPerformanceCounter(&nClck4);
taptime[3]=((DWORD)((nClck4.QuadPart)* 1000000 / nFreq.QuadPart));
tap[3]=(taptime[3]/1000)-(taptime[2]/1000);
tap[0]=(tap[1]+tap[2]+tap[3])/3;                            //間隔値3つを平均
tempo1=(double)(60000/tap[0]);                                        //テンポを変更
SetWindowText(hStatic,"テンポを変更しました");
fprintf(flog,"タップ間隔= %lfミリ秒¥n テンポ変更　tempo1= %lf¥n%lf と %lf と %lf の平均¥n",tap[3],tempo1,tap[1],tap[2],tap[3]);

}
else{
```

```
QueryPerformanceCounter(&nClck5);
taptime[4]=((DWORD)((nClck5.QuadPart)* 1000000 / nFreq.QuadPart));
tap[4]=(taptime[4]/1000)-(taptime[3]/1000);   //最新のtapと1つ前のtapの間隔を取得
if(tap[4]>(tap[0]*2)||tap[4]<(tap[0]/2)){   //元のテンポと違いすぎた場合
SetWindowText(hStatic,"ERROR!  元のテンポと違いすぎます");
taptime[3]=taptime[4];              //taptime3に「最新のtap」を保存しておく
fprintf(flog,"タップ間隔= %lfミリ秒(アウツ)¥n  テンポ変更   tempo1= %lf¥n%lf  と  %lf  の比較
¥n",tap[4],tempo1,tap[0],tap[4]);
}
else if(tap[4]>((60000/tempodef)*3/2)||tap[4]<((60000/tempodef)*2/3)){   //元のテンポと違いすぎた場合
SetWindowText(hStatic,"ERROR!  元のテンポと違いすぎます");
taptime[3]=taptime[4];              //taptime3に「最新のtap」を保存しておく
fprintf(flog,"タップ間隔= %lfミリ秒(アウト)¥n  テンポ変更   tempo1= %lf¥n%lf  と  %lf  の比較
¥n",tap[4],tempo1,60000/tempodef,tap[4]);
}
else{                                  //元のテンポと違いすぎず、適正範囲だった場合
for(n=1;n<4;n++){
tap[n]=tap[n+1];                  //間隔の保存値を1つずつ更新(最新の4つを使用するため)
}
taptime[3]=taptime[4];
tap[0]=(tap[2]+tap[3])/2;
tempo1=(double)(60000/tap[0]);
SetWindowText(hStatic,"テンポを変更しました!");
fprintf(flog,"タップ間隔= %lfミリ秒¥n  テンポ変更   tempo1= %lf¥n%lf  と  %lf  の平均
¥n",tap[3],tempo1,tap[2],tap[3]);
}



}



}
if(msg.message==WM_KEYDOWN && msg.wParam==0x27){//→キー：テンポリセット   元のテンポに
戻してタップ回数もリセット
tempo1=tempodef;
```

```
SetWindowText(hStatic,"テンポを戻しました");
tapCount=0;
for(n=0;n<6;n++){
tap[n]=0;
taptime[n]=0;
}
}

if(msg.message==WM_KEYDOWN && msg.wParam==0x0d){//Enterキー：テンポ保存　現在のテンポを元のテンポとして保存
tempodef=tempo1;
SetWindowText(hStatic,"現在のテンポを保存しました");

}

if(msg.message==WM_KEYDOWN && msg.wParam==0x11){//Ctrlキー:頭出し　現在のノートを強制終了
fclose(flog);
break;
}

if(msg.message==WM_KEYDOWN && msg.wParam==0x42){//Bキー：一時停止　現在の指，圧を保持したまま停止
SetWindowText(hStatic,"一時停止中");
while(stop==0){
if(PeekMessage(&msg,NULL,0,0,PM_REMOVE)!=0){
if(msg.message==WM_KEYDOWN && msg.wParam==0x4e){//Nキーが押されるまで一時停止
stop++;
}
}
}
SetWindowText(hStatic,"再開");
stop=0;
}

if(msg.message==WM_KEYDOWN && msg.wParam==0x1b){ //Escが押された時全停止
```

```
*PlayCheck=FALSE;

ch.ulChNo=2;   //チャンネル2に唇圧0Vを出力
DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure);
ch.ulChNo=3;   //タンギングに0Vを出力
DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure);
ch.ulChNo=4;   //チャンネル4に送気圧0Vを出力
DaOutputDA(hDeviceDA,1,&ch,&NoteOffPressure);

mciSendCommand(mciDev.wDeviceID, MCI_STOP, 0, NULL);
fclose(flog);
break;
}else {
DefWindowProc(hWnd, msg.message, msg.wParam, msg.lParam);
}
}
timer2=GetTickCount();
QueryPerformanceCounter(&nAfter);

if(bib==1 || bib2==1){//ビブラート項目
if(PeekMessage(&msg, hWnd, 0, 0, PM_NOREMOVE)!=0){
InvalidateRect(hWnd, NULL, TRUE);
break;
}
SetWindowText(hStatic,"ビブラート発動");
sec_pc = (nAfter.QuadPart - nBefore.QuadPart)/(double)nFreq.QuadPart;//時間
for(k=0;k<3;k++){
dlipVBase[k]=clipVBase[k]
+(double)(shinpuku2[dina]*nowmodul/127)*sin(2*PI/(syuki2[dina]/100)*sec_pc);
}
if(dlipVBase[dina]>999)dlipVBase[dina]=999;//上限突破阻止
if(dlipVBase[dina]<0)dlipVBase[dina]=1;//下限突破阻止
clip=-1*(int)(((double)dlipVBase[dina]/100+10)*4096/20+0.5);
fprintf(flog,"%d¥n",clip);
ch.ulChNo=2;
DaOutputDA(hDeviceDA,1,&ch,&clip);
```

```
}

if(((DWORD)((nAfter.QuadPart - nBefore.QuadPart) * 1000000 /
nFreq.QuadPart))>=(1000*Second/(double)(tempo1*bunkai2))){//演奏中の加速と減速
accel=0;
if(nNow2.QuadPart==0x00){TrueTotalTime+=((DWORD)((nAfter.QuadPart - nBefore.QuadPart) * 1000000 /
nFreq.QuadPart)); //トータルタイムにwait時間を加算
fprintf(flog," wait終了¥n TotalTime= %d¥n",TrueTotalTime);
}
else                              {TrueTotalTime+=((DWORD)((nAfter.QuadPart - nNow2.QuadPart) * 1000000 /
nFreq.QuadPart));     //保険
fprintf(flog," wait終了¥n TotalTime= %d¥n",TrueTotalTime);
}
fclose(flog);
break;}
}

}

void cyutime(double Second2,int now2){//チューニング中にビブラートボタンを押した際の処理
MSG msg;
DWORD timer,timer2;
int n;
int k;
double sec_pc2;

LARGE_INTEGER nFreq, nBefore, nAfter;

//変数の初期化
memset(&nFreq,      0x00, sizeof nFreq);
memset(&nBefore, 0x00, sizeof nBefore);
memset(&nAfter,    0x00, sizeof nAfter);

QueryPerformanceFrequency(&nFreq);
QueryPerformanceCounter(&nBefore);
```

```
timer=GetTickCount();

for(k=0;k<=Second2;k++){
timer2=GetTickCount();
QueryPerformanceCounter(&nAfter);

if(bib3==1){//ビブラート項目
if(PeekMessage(&msg, hWnd, 0, 0, PM_NOREMOVE)!=0){
InvalidateRect(hWnd, NULL, TRUE);
break;
}
sec_pc2 = (nAfter.QuadPart - nBefore.QuadPart)/(double)nFreq.QuadPart;//時間

cyulipVBase=cyulip +(double)(shinpuku[50][dina])*sin(2*PI/(syuki[50][dina]/100)*sec_pc2);
if(cyulipVBase>999)cyulipVBase=999;//上限突破阻止
if(cyulipVBase<0)cyulipVBase=1;//一応下限突破阻止
cyulip2=-1*(int)(((double)cyulipVBase/100+10)*4096/20+0.5);
ch.ulChNo=2;
DaOutputDA(hDeviceDA,1,&ch,&cyulip2);
Sleep(1);
}

}


}
```