Confidential

SmileSound
FULL PROGRAMMABLE SOUND DECODER

スマイルサウンドデコーダ取り扱い説明書

Ver.2022.10.7

デスクトップステーション株式会社
DCC電子工作連合

1. 目次

2. はじめに

2.1. スマイルサウンドデコーダとは

スマイルサウンドデコーダは、日本国内で設計・開発され、全世界で共通で使用可能なNMRA DCC規格に準拠したDCCサウンドデコーダです。

全てサウンドフローと呼ばれるスクリプトによって、サウンドプログラミングの記述を行います。同時並列にサウンドフローを実行可能なインタプリタ・サウンドエンジンを備えることにより、様々な鉄道車両の挙動や表現を実現でき、表現豊かなサウンドモデル化に貢献します。

以下の特徴を有します。

- RP2040と16MB(128Mbit)の大容量・高速FLASHメモリを活用したサウンドデコーダ
- 平易でシンプルかつ高機能なインタプリタエンジンを搭載し、最大16点のユーザーカスタムプログラムを同時実行可能。状態遷移を表現可能なサウンドプログラミングを実現
- 最大同時発音数10チャンネル, 32kHz 16bitの再生に対応。省メモリ対応に貢献する16kHz及び8kHzのサンプルレート及び8bit音声もサポート。
- RailCom(BiDi)等、ワールドワイドで普及が進むDCC関連技術の標準搭載
- 専用アダプタを介してUSB経由による高速ファームウェア・サウンドデータの更新
- オープンサウンドデータの膨大なサウンドライブラリを移植可能な機能・パフォーマンスの実現
- アナログ制御には未対応

2.2. 仕様

仕様項目	仕様値・性能	説明
対応電圧	12-19V	
モータ出力電流	0.5A(Mini) / 1.5A(Standard)	連続出力時となります。
SpeedStep	14,28,128	
保護機能	モータ過電流保護,スピーカー短絡保護	
スピーカ容量	3W, 4-32Ω	
サウンド同時発音数	10音	
サーボ機能	未対応	対応予定
Marklin/Mfx	×	メルクリンデジタルは未対応
アナログ制御	×	アナログパワーパックでの運転は未対応
RailCom	○	アドレス読み出しのみサポート

2.3. サポート・有償交換

SmileSoundデコーダのサポートは、デジタル鉄道模型フォーラム（https://desktopstation.net/bb/）にご投稿ください。

デコーダの故障や初期不良は、デスクトップステーションオンラインストアの問い合わせページか、メールでご連絡ください。

初期不良の際は無償で良品と交換させていただきます。ユーザー起因による故障は、有償交換とさせていただきます。期間の定めがありますので、後述の保証規定を参照ください。

有償交換費用 3000円

2.4. 保証規定

【保証範囲】

当社は、この書面に記載された製品について保証します。

【保証期間】

ユーザーの購入日より、1年間とします。 また、有償修理・有償交換は、購入日より、3年間とします。

なお、超上級者品・プロトタイプ品と明記された商品については、ユーザーに一定のリスクを担保して頂く代わりに、価格割引を行っていることから、保証期間は購入日より1ヵ月以内とします。

保証内容】

この書面に記載された内容について、保証期間内に当社の責に帰すべき瑕疵により不具合が発生した場合は、代品との交換または補修を無料で行います。保証期間を経過した場合は、有償になります。

該当製品の販売が終了した場合には、代替製品に変えさせていただく場合がございます。

【有償修理・有償交換】

保証期間内であっても、次のような事項に該当する場合は、有償修理または有償交換となります。

- 購入場所及び購入日を証明する情報(注文メール,納品書,領収書等)の提示がない場合
- ユーザーまたは加工業者の取り付け作業に起因する不具合、故障（例えば、搭載中の絶縁不良等によるショート故障,誤った配線での故障等）
- 表示された商品の性能を超えた用途に使用された場合の不具合(例えば、HO向け製品をGゲージや1番ゲージなどに使用)
- 商品または部品の経年変化（使用に伴う消耗、摩耗など）や経年劣化またはこれらに伴うその他の不具合
- 保管場所・搭載場所の環境に起因する要因。埃、髪・ペットの毛、粉塵、高温多湿、結露、腐食またはその他の不具合
- 商品または部品の材料特性に伴う仕様（基板端面の処理、コネクタの錆・劣化など）
- 天災その他の不可抗力（例えば、暴風、豪雨、高潮、地震、落雷、洪水、地盤沈下、火災など）による不具合またはこれらによって商品の性能を超える事態が発生した場合の不具合
- 操作の誤り、調整不備または適切な維持管理を行わなかったことによる不具合（例えば、車輪やレールの清掃、コマンドステーションのメンテナンスなど）
- ユーザー自身の取付け、修理、改造（必要部品の取り付け・取外しを含む）に起因する不具合

【修理・交換の対象】

次のような事項に該当する場合は、保証・有償修理または有償交換が受けられません。

- 犯罪などの不法な行為に入手された場合
- 弊社または弊社代理店以外の手段で購入・受領された場合
- 競合企業または個人が、製品の分析のために購入・改造された場合
- 日本国内向け製品を、海外で使用されている場合
- 当社が提供するファームウェア以外を使用された場合
- 弊社製品の模造品、弊社の許可なく改造された品
- 弊社からサポートを受けられない旨を説明され、購入・入手された場合（サンプル品、B級品等）

【保証規定の改訂】

本保証規定は、予告なく改訂する場合があります。

3. 環境構築

Windows 10またはWindows 11を想定しています。 Mac等でのエミュレーションソフトウェア環境を使用する場合には、ユーザーの自己責任にて対応ください。

予めDSSPでサウンドデータをUF2(書込み用パッケージ化。ssdx形式への復元は不可能)化して頂くことで、MacやLinuxでも書き込みのみ可能です。

3.1. 必要なハードウェア一覧

SmileSoundを利用するためには、以下の機器が必要となります。保有していない場合、機能に制約を受ける場合があります。

開発機器・治具	仕様
パソコン	Windows10以降のPC。AMDまたはintel製を前提とします。
専用のUSBライター	スマイルワークス製USBライター
ミニUSBケーブル	USBライター用
スマイルサウンドデコーダ	Standard または Mini
デコーダテスター	デコーダの動作確認用
DCCコマンドステーション	DSair2など

3.2. ソフトウェア

本説明書で使用するソフトウェアを以下に掲載します。

ソフトウェア名	説明
Windows 10 または Windows 11	サポートするOS
.Net Framework 4.7.2	無償。DSSPが使用するランタイム環境。Windows 10以降は標準搭載
DSSP	無償。サウンドデータ作成・書き込みツール

ソフトウェア名	説明
Audacity	無償。サウンド編集ツール(トリミング,サンプリング調整など全般)
SpectraLayers Pro	有償。必須ではありません。サウンド編集ツール(主に雑音除去・クリーニング・修復用)

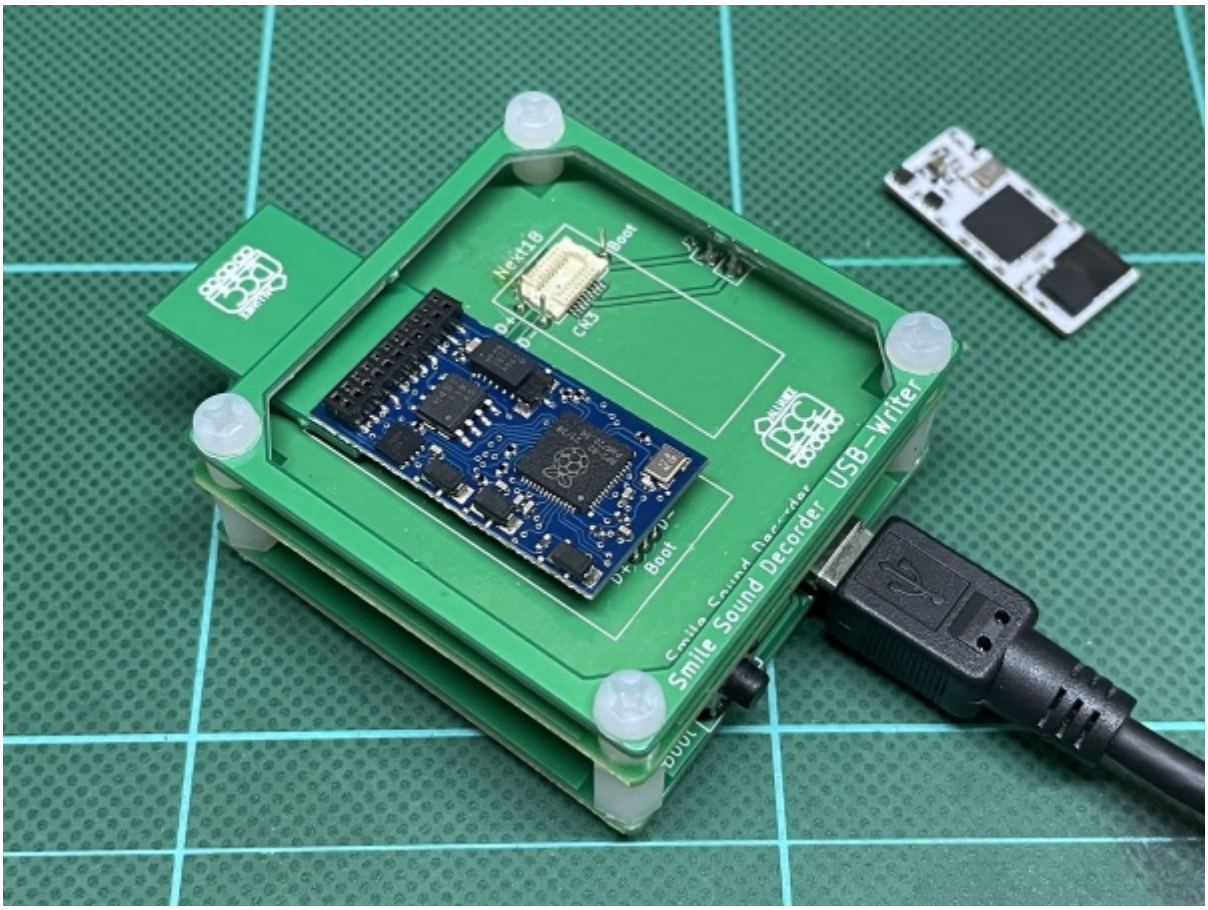
4. USBライター

4.1. USBライターとは

スマイルサウンドデコーダは、現時点でUSB経由での書込みしかサポートしていません。ファームウェアとサウンドデータは、USBライターを介して書き込みを行います。USB配線は、ポゴピン(バネを内蔵した剣山のようなもの）を使って、USBライターとデコーダの間を接続しています。

Next18のスマイルサウンドデコーダは指で簡単に取り外しできますが、MTC21のスマイルサウンドデコーダを取り外すときには、ツメやプラスチックのピン、ギター用のピック等を用いてください。この時、デコーダ基板上の部品に取り外しに使った治具の先などが直接当たらないように気を付けてください。

スマイルサウンドデコーダにファームウェアやサウンドデータを書込む方法は、いくつかの方法があります。

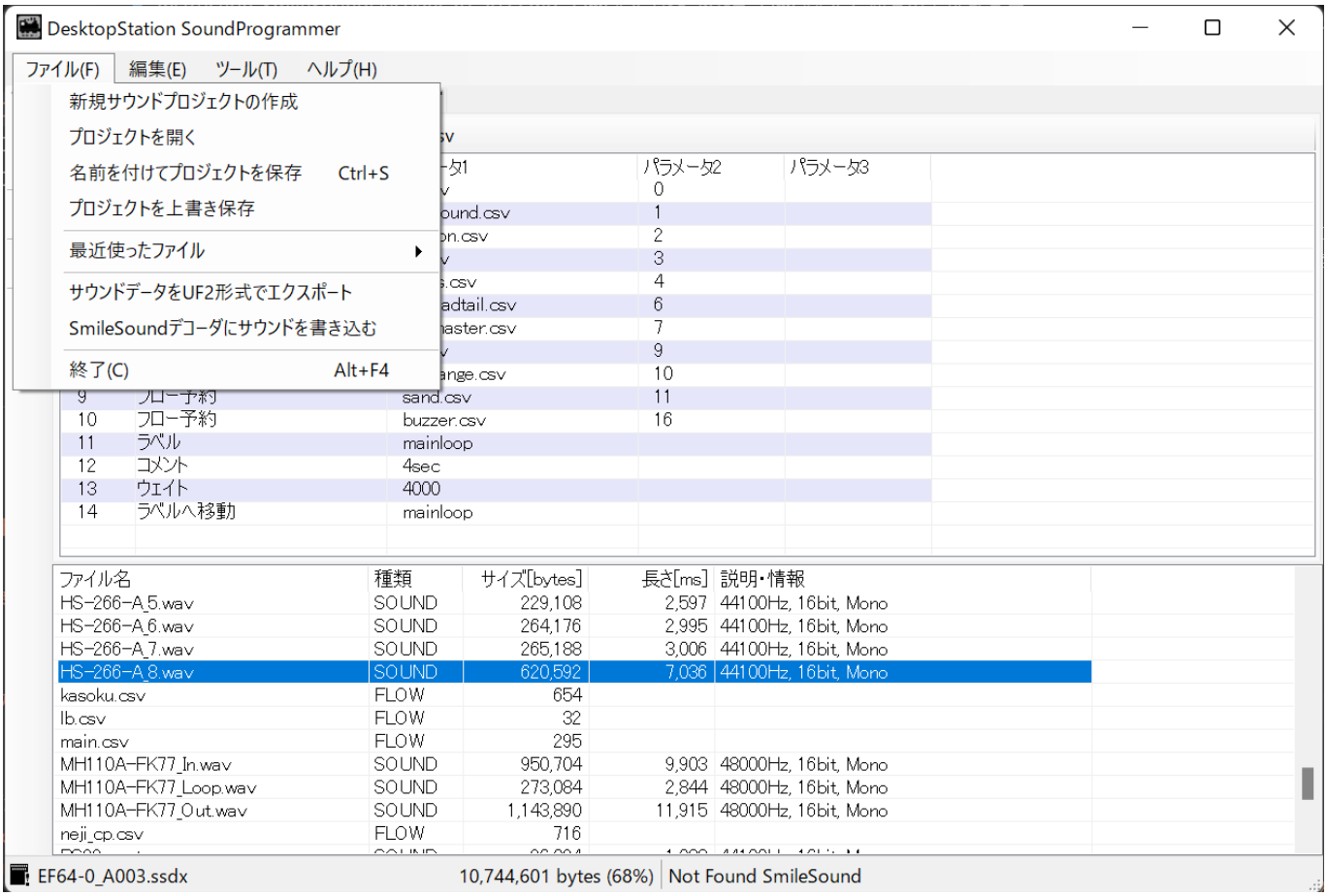


4.2. サウンドの書込み方法

DSSPを使うことで、ファームウェアの書き込みと、サウンドデータの書き込みが行えます。２つの書込み方法がありますが、通常は、DSSPを使った直接書き込み機能である「SmileSoundデコーダにサウンドを書込む」の利用を推奨します。

- サウンドデータをUF2形式でエクスポート
- SmileSoundデコーダにサウンドを書込む

サウンドデータをUF2形式でエクスポートした場合、後述の手動書込みで利用できます。

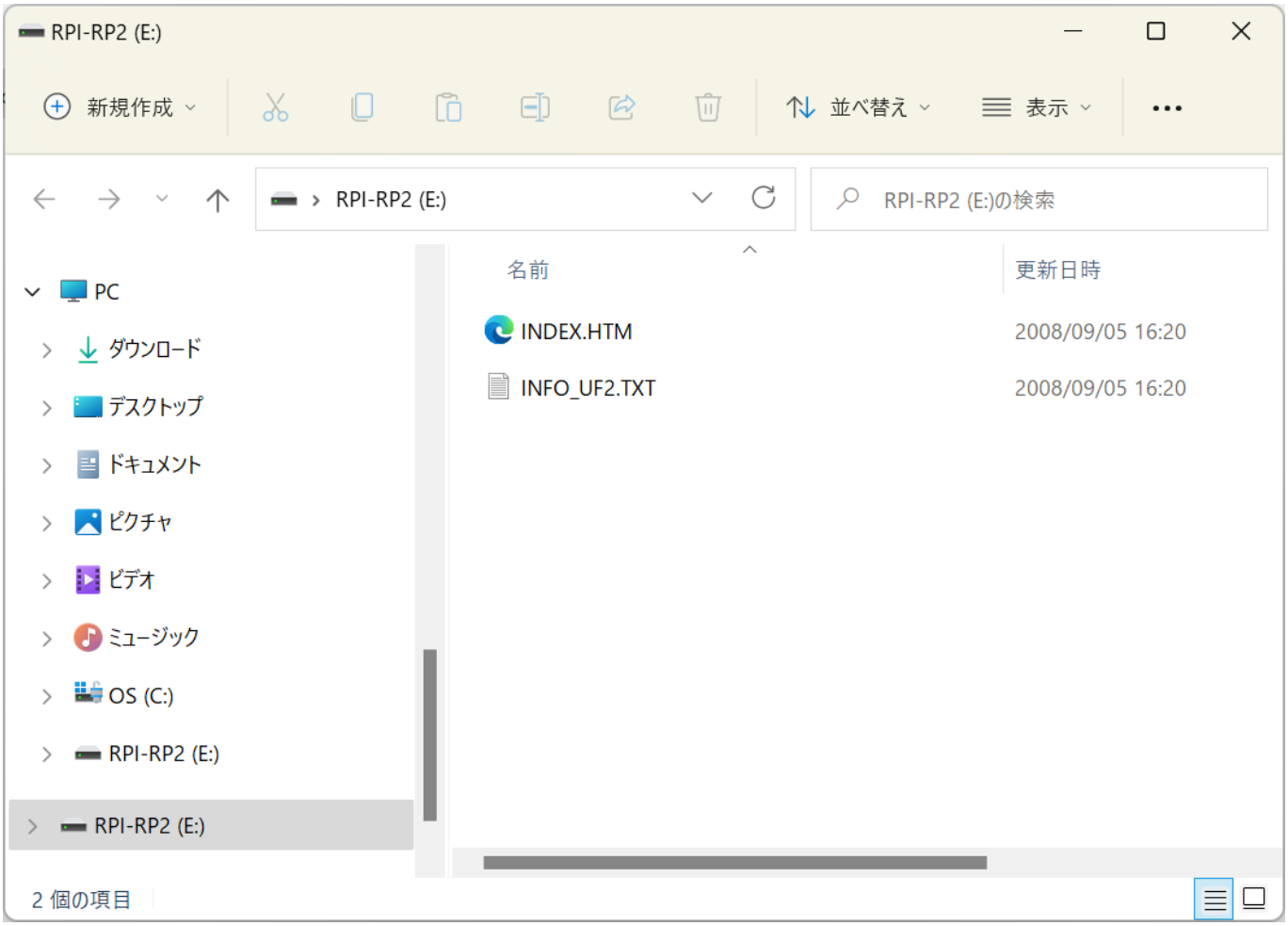


4.3. 手動書込み

ここでは、DSSPを使わずに、エクスポートされたUF2ファイルを書込む方法について説明します。通常、手動書込みの操作は不要ですが、うまくPCにスマイルサウンドデコーダが認識されない時には、この方法を取ってください。

ボタン操作をしないと、パソコンにUSBライターを繋げたとき、自動的にUSBドライブが表示されません。USBライターにある、ボタンを押しながら、パソコンにUSBライターを繋げてください。パソコン上でUSBドライブが表示されれば、ボタンを離して構いません。

DSSPは、ドライブとして認識したスマイルサウンドデコーダも、自動で判別して書き込みを行えます。



手順	操作方法
1	USBライターにスマイルサウンドデコーダを装着する
2	USBライターのボタンを押し続ける
3	USBライターのケーブルをパソコンに差し込む
4	パソコン上にUSBドライブが認識することを確認する
5	USBライターのボタンを離す
6	パソコン上でUF2形式のサウンドデータをD&Dなどするか、DSSPで書込みボタンを押して書込む。
7	書き込みが終わると、自動的にドライブが取り外しされるので、USBケーブルをパソコンから外す。
8	USBライターから、スマイルサウンドデコーダを取り外す
9	書き込み作業完了

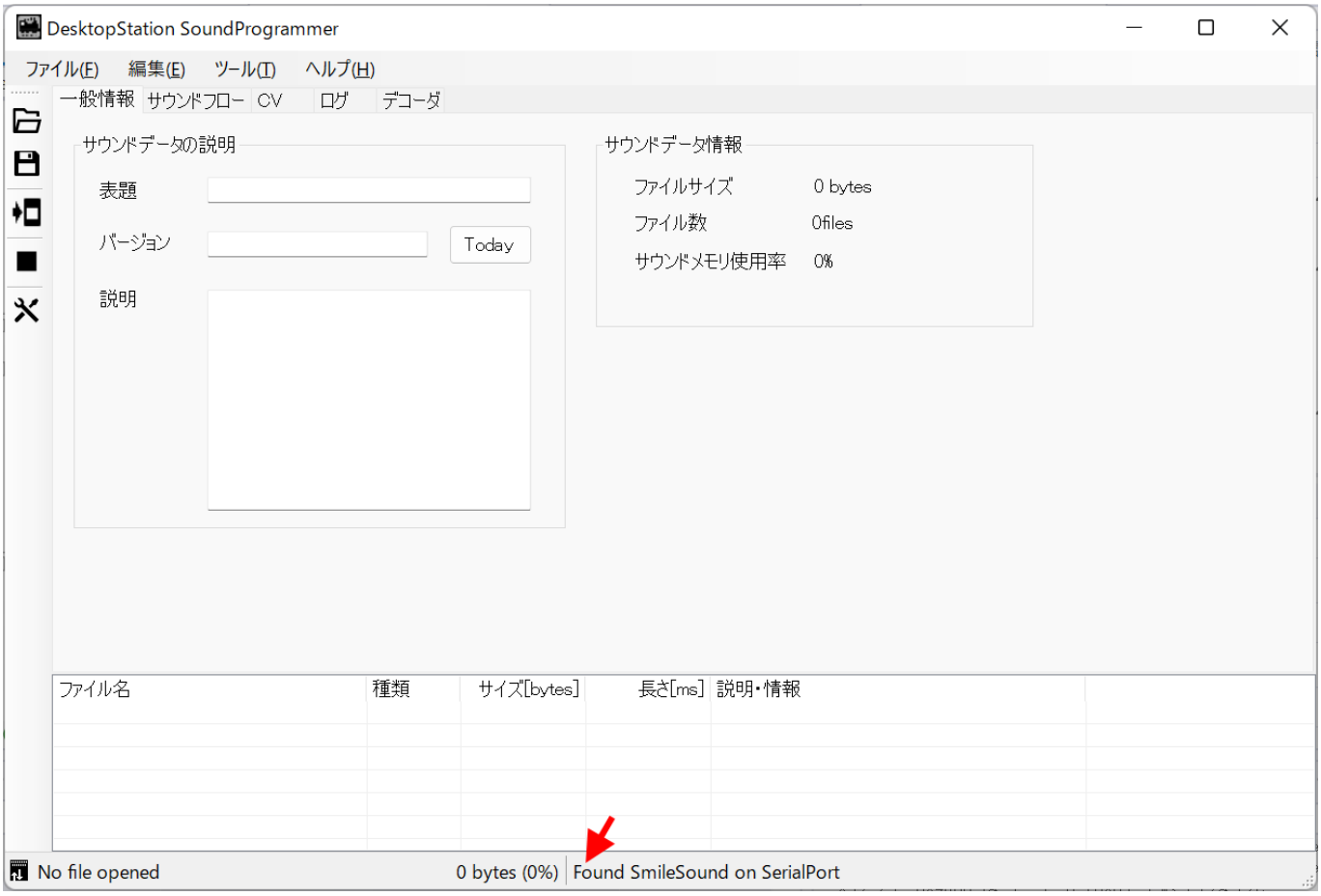
4.4. 自動認識

DSSPでは、ドライブとして認識していないスマイルサウンドデコーダを検出して、強制的にドライブを開く処理を行って、書き込みができるようにする機能が搭載されています。

PCの環境やUSB機器の状況によっては、うまく行かないケースが稀に発生します。そのときは、前述の手動書込み操作を行ってください。

自動認識の場合の操作手順は以下の通りです。

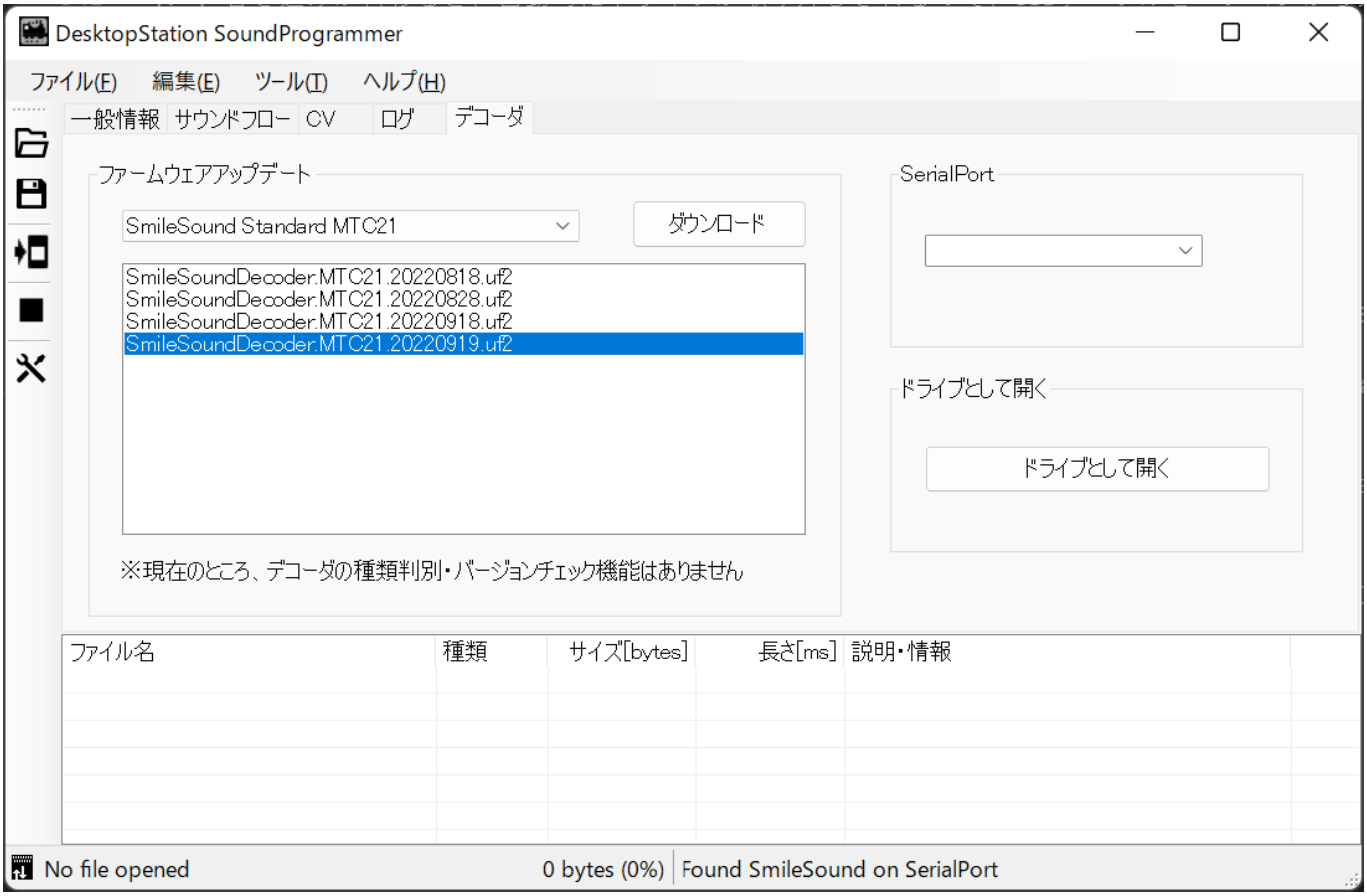
手順	操作方法
1	USBライターにスマイルサウンドデコーダを装着する
2	USBライターのケーブルをパソコンに差し込む
3	DSSPを起動する。DSSPの一番下のバー上に「Found SmileSound on SerialPort」と表示されているのを確認する。
4	DSSPで書き込みたいサウンドデータを開く。または書き込みたいファームウェアを選ぶ。
5	DSSPで書込みボタンを押して書込む。
6	書き込みが終わるまで数分待つ(ファームウェア書き込みは数秒で終了)
7	書き込みが終わると、自動的にドライブが取り外しされるので、USBケーブルをパソコンから外す。
8	USBライターから、スマイルサウンドデコーダを取り外す
9	書き込み作業完了



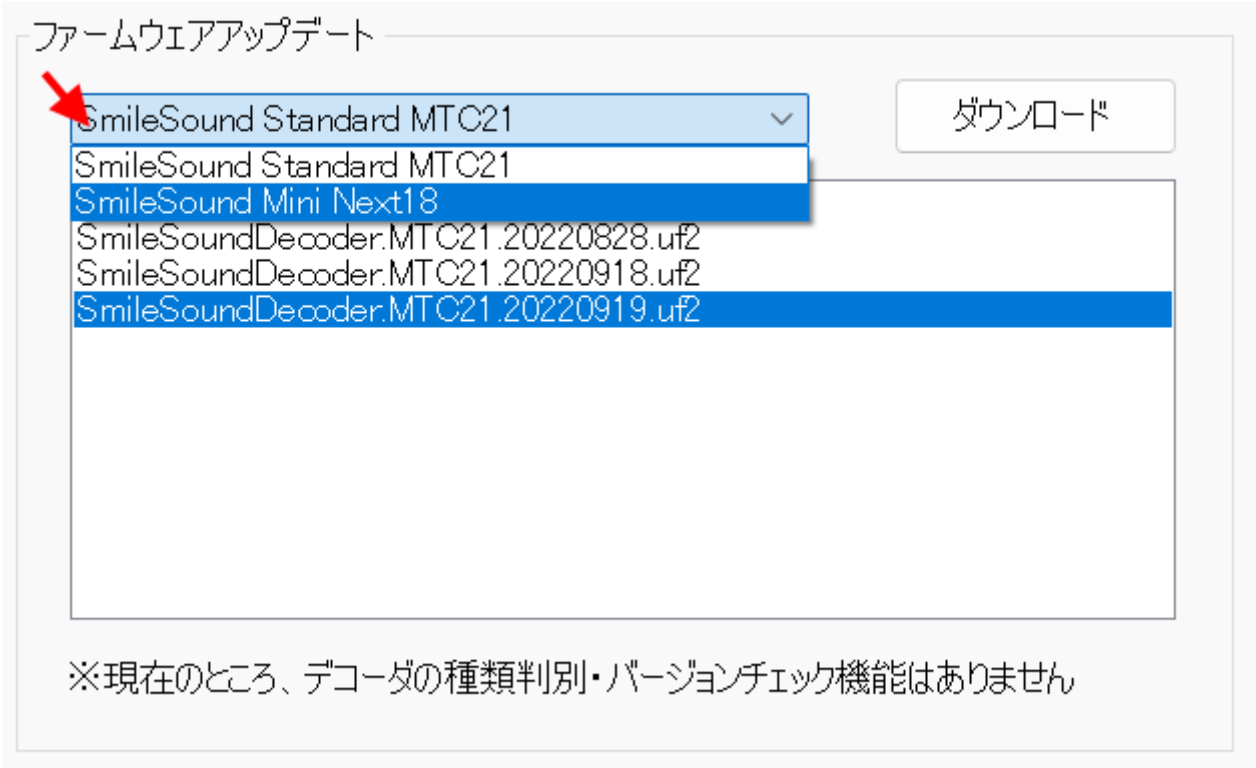
4.5. ファームウェアのアップデート

DSSPを使って、ファームウェアのアップデートを行うことができます。過去の複数のバージョンを用意していますので、トラブル時等にはバージョンを戻すことができます。

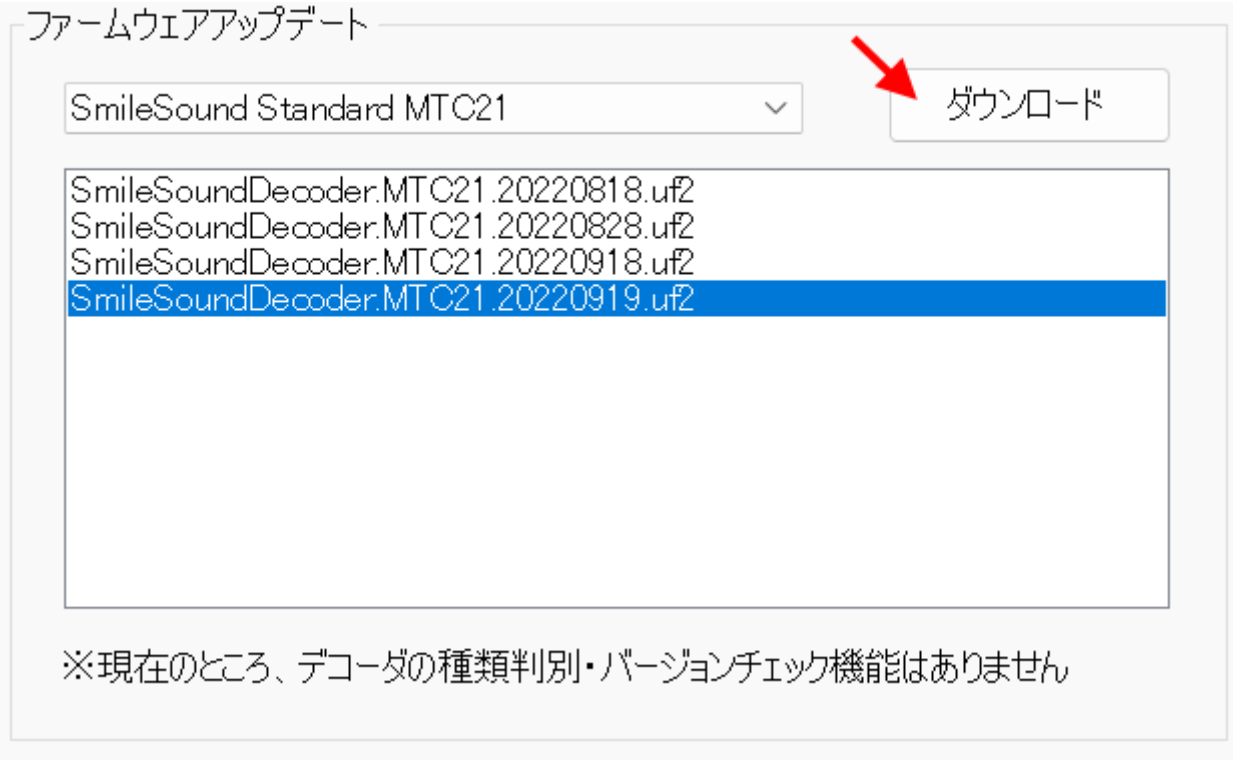
「デコーダ」タブを選択すると、ファームウェアのアップデート操作が行えます。



ファームウェアは、MTC21版とNext18版で異なります。間違えて書込むと、モータが動かないなどの不具合が出ますので、ご注意ください。



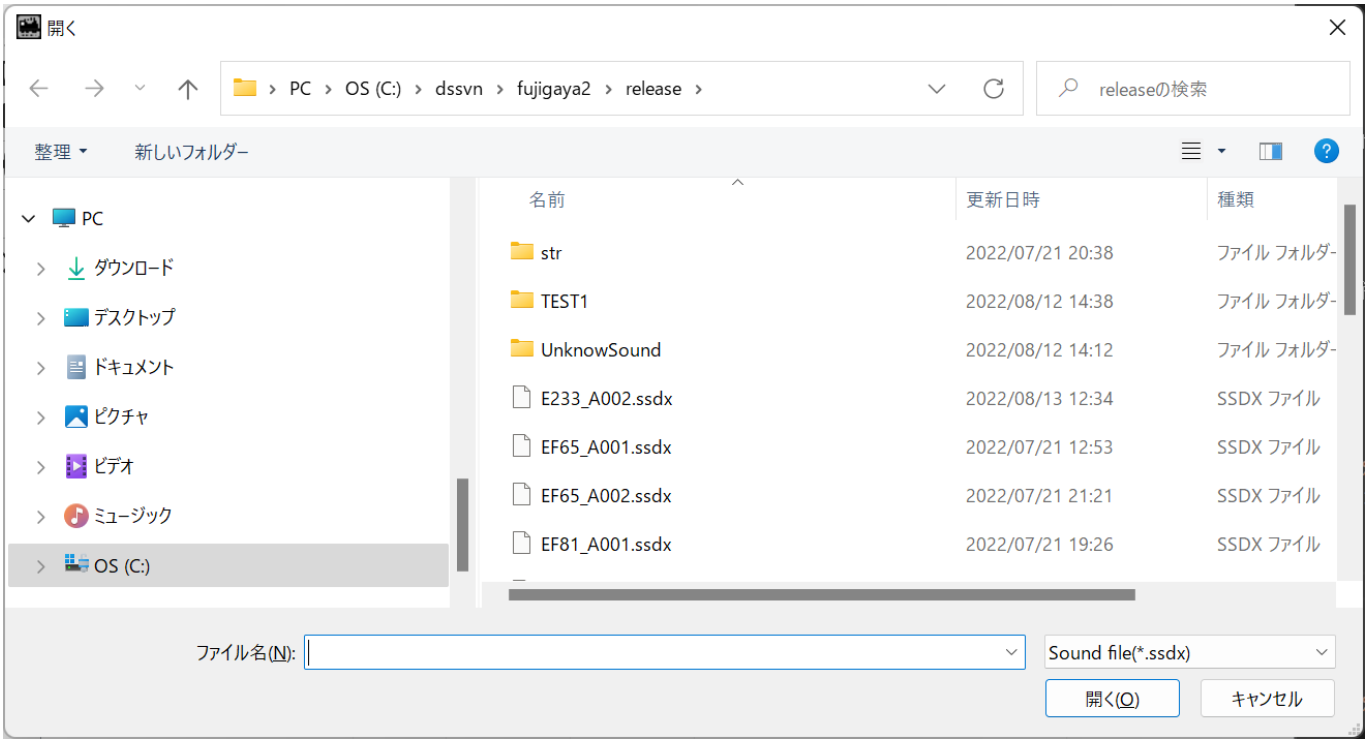
書き込みたいファームウェアの日付（バージョン）を選択したら、「ダウンロード」ボタンを押して書き込みます。 書き込む際には、SmileSoundが認識していることを確認してください。



5. DesktopStation SoundProgrammer(DSSP)

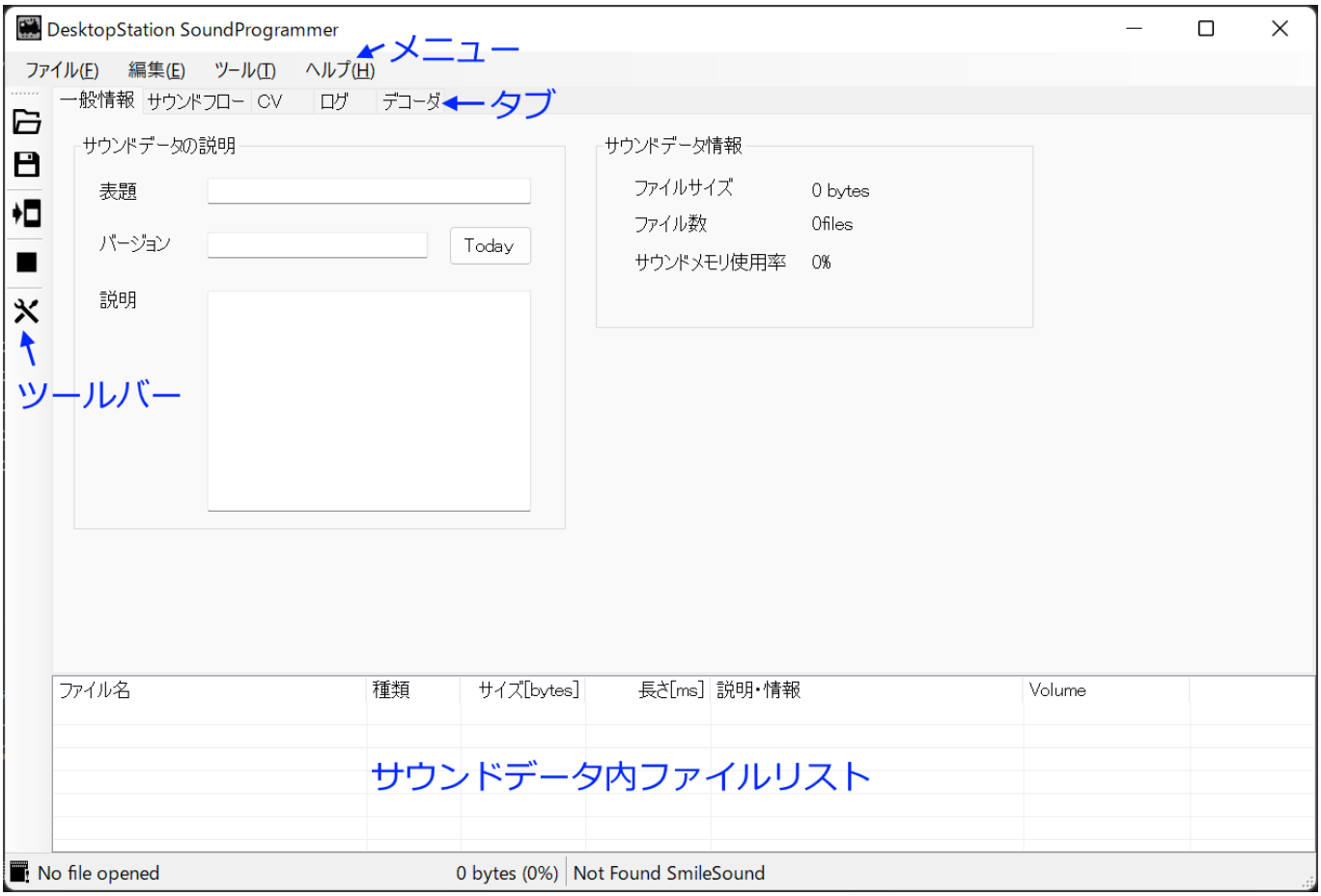
5.1. DSSPとは？

DSSPは、スマイルサウンドデコーダでサウンドデータを作成するためのツールです。スマイルサウンドデコーダに書き込めるUF2形式のファイルを生成できます。



5.2. 画面の説明

DSSPを起動すると、以下のような画面が表示されます。画面は、メニュー・ツールバー・ファイルリストと、機能タブの４つで構成されます。



メニューは、以下のような機能を有しています。

メニュー名	機能の説明
ファイル	サウンドデータを開く・保存する、デコーダに書き込む機能が入っています
編集	サウンドフローの編集や、サウンドデータのファイルのインポート機能等が入っています。
ツール	設定や、DSSPに付随するツールを表示します。
ヘルプ	DSSPのバージョン情報などを表示します。

左側のツールバーには、以下のような機能が割り当てられています。

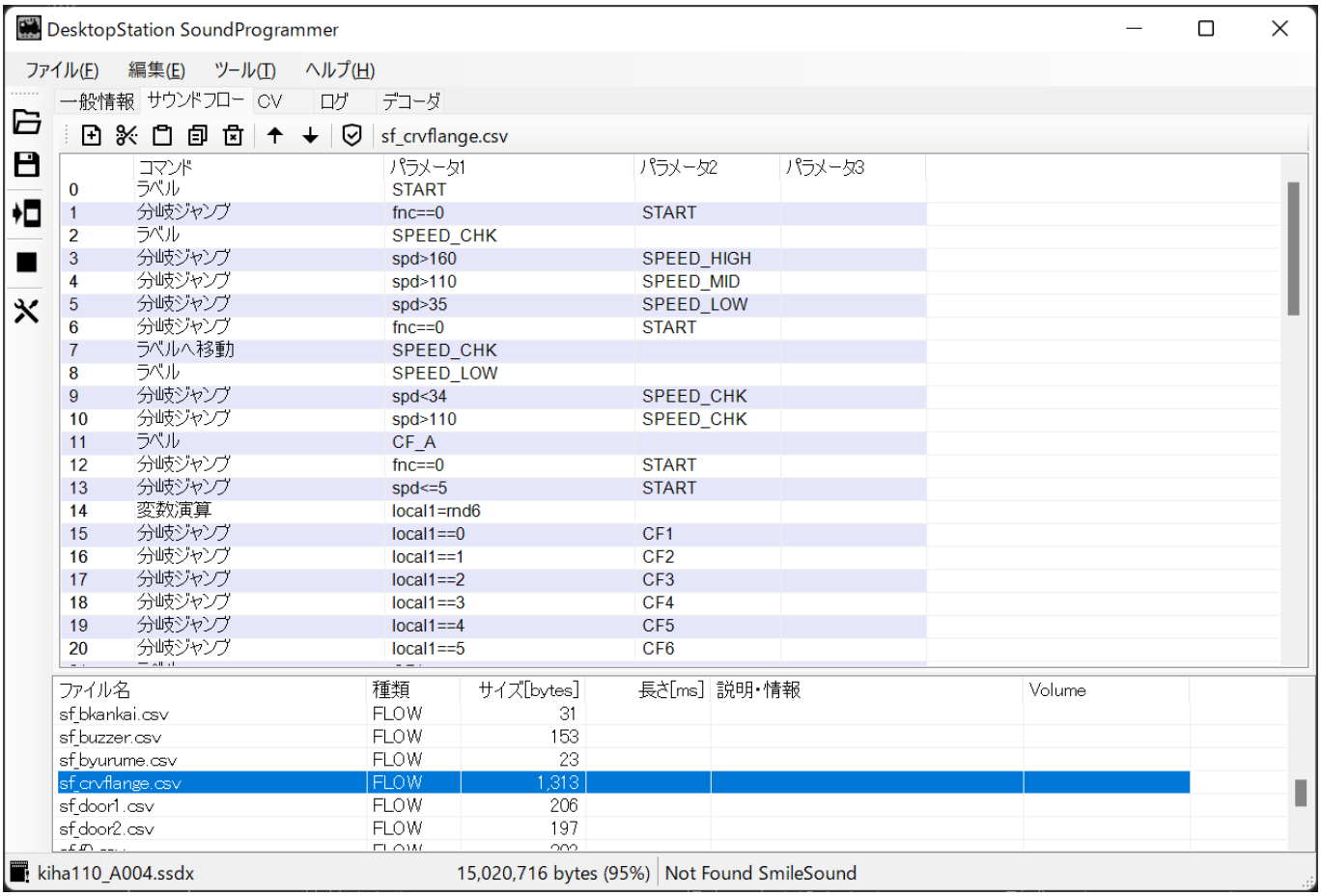
アイコン	機能
	プロジェクトを開く
	プロジェクトを保存（上書き）
	SmileSoundデコーダにサウンドを書込む
	再生中のサウンドを停止

アイコン	機能
	設定を開く

その他の画面の説明は以下の通りです。

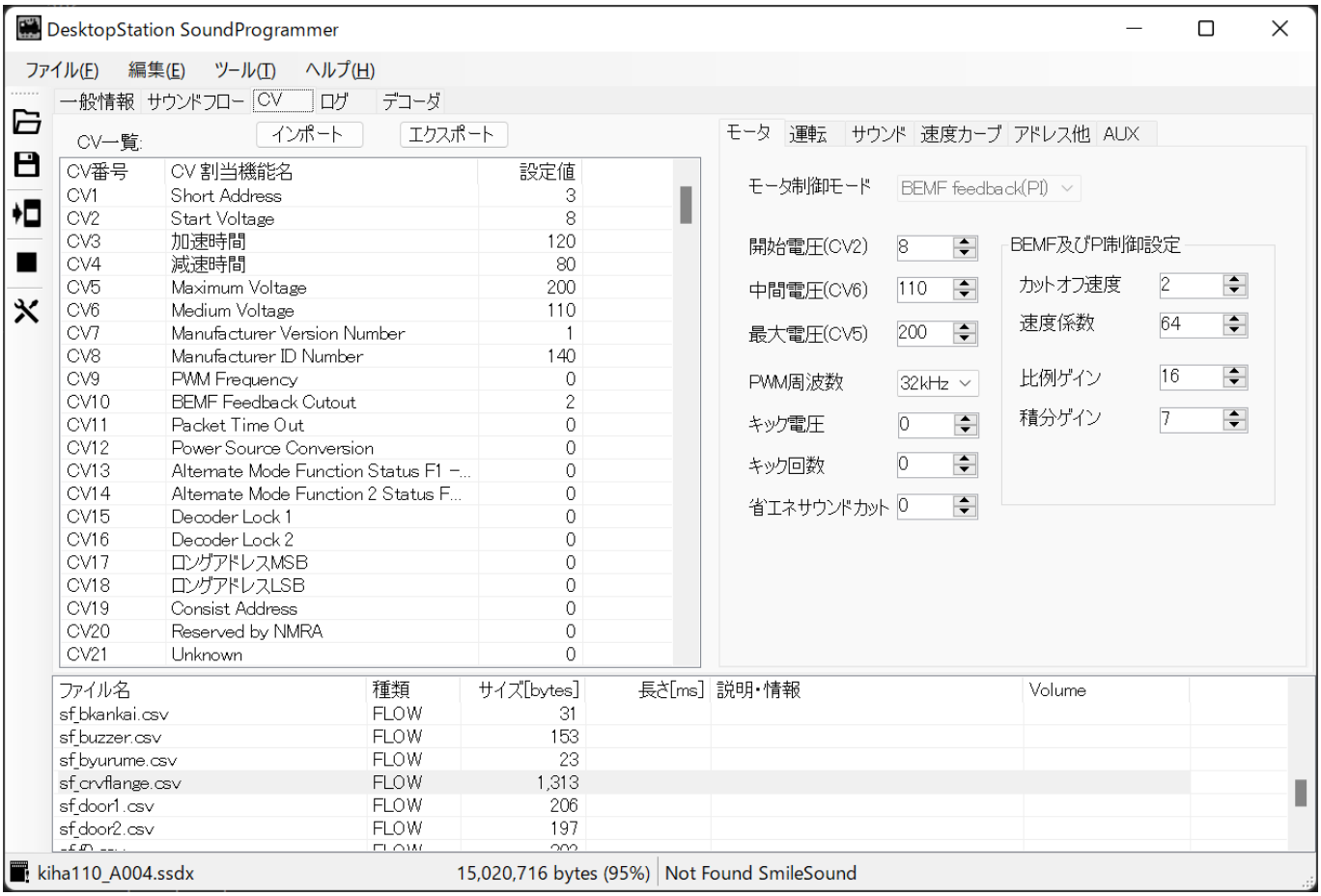
5.2.1. サウンドフロー編集画面

サウンドフロー編集画面は、ファイルリストからサウンドフローファイル(CSV)を開くと、自動で表示されます。



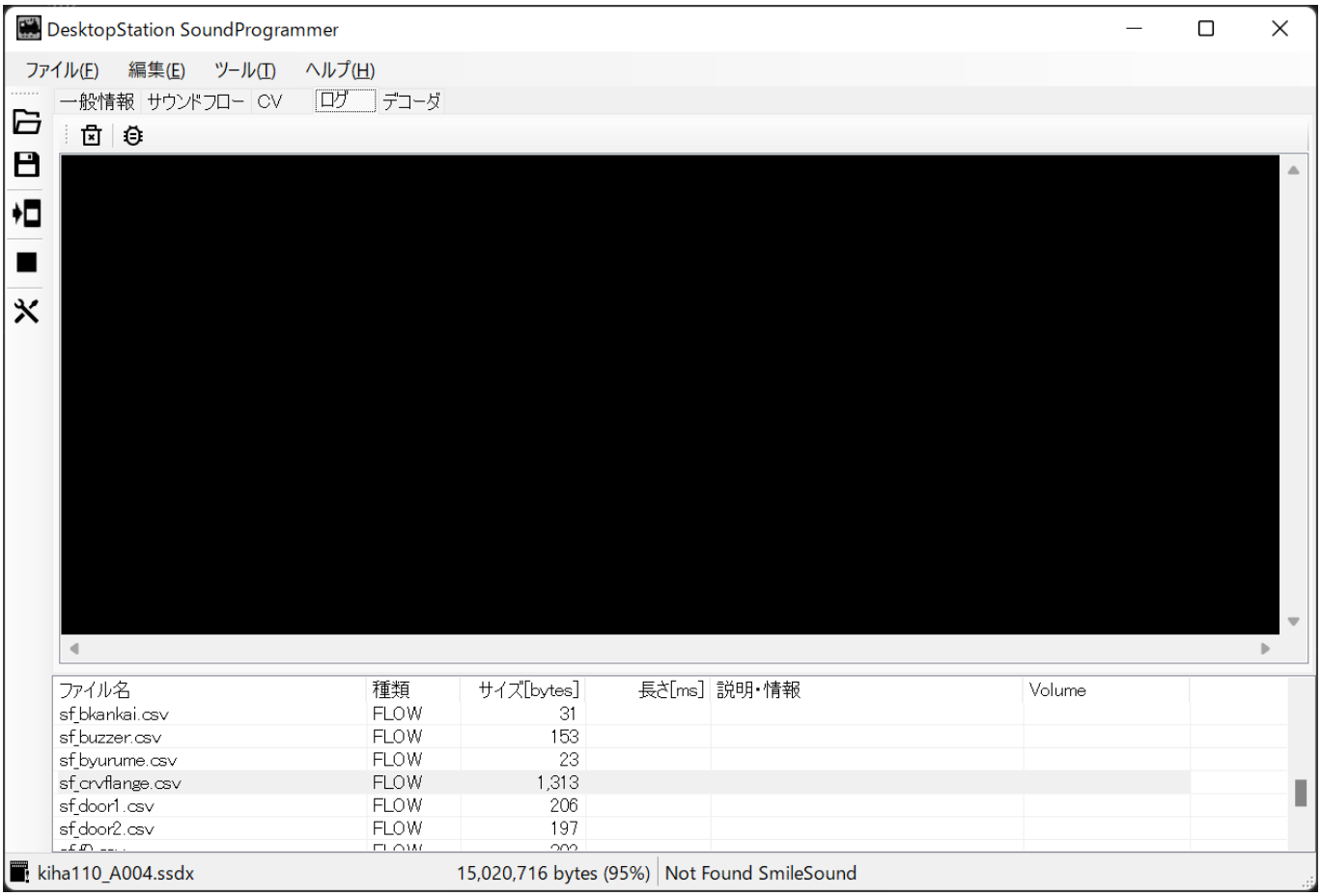
5.2.2. CV・デコーダ設定画面

デコーダの初期設定を行う画面です。サウンドフローと密接に関わるものとなります。



5.2.3. ログ画面

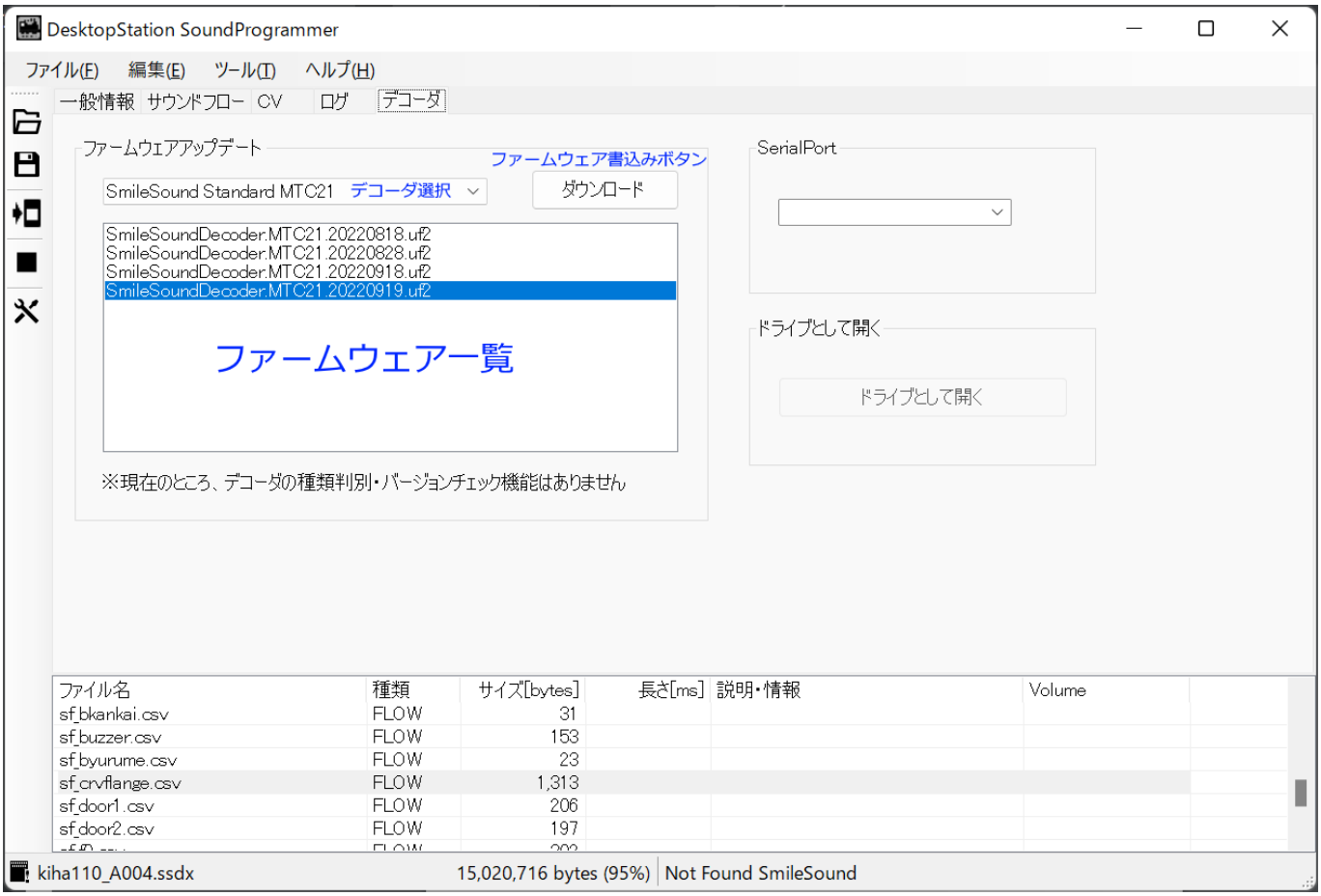
サウンドフローにエラーが出ている場合、ログにエラー状態が表示されます。



5.2.4. ファームウェアアップデート画面

ファームウェアのアップデートを行うための画面です。基本的に最新のファームウェアバージョンをお使いください。

注意：Next18版に、MTC21版を書き込んだりしますと、機能がうまく動かない現象が発生します。



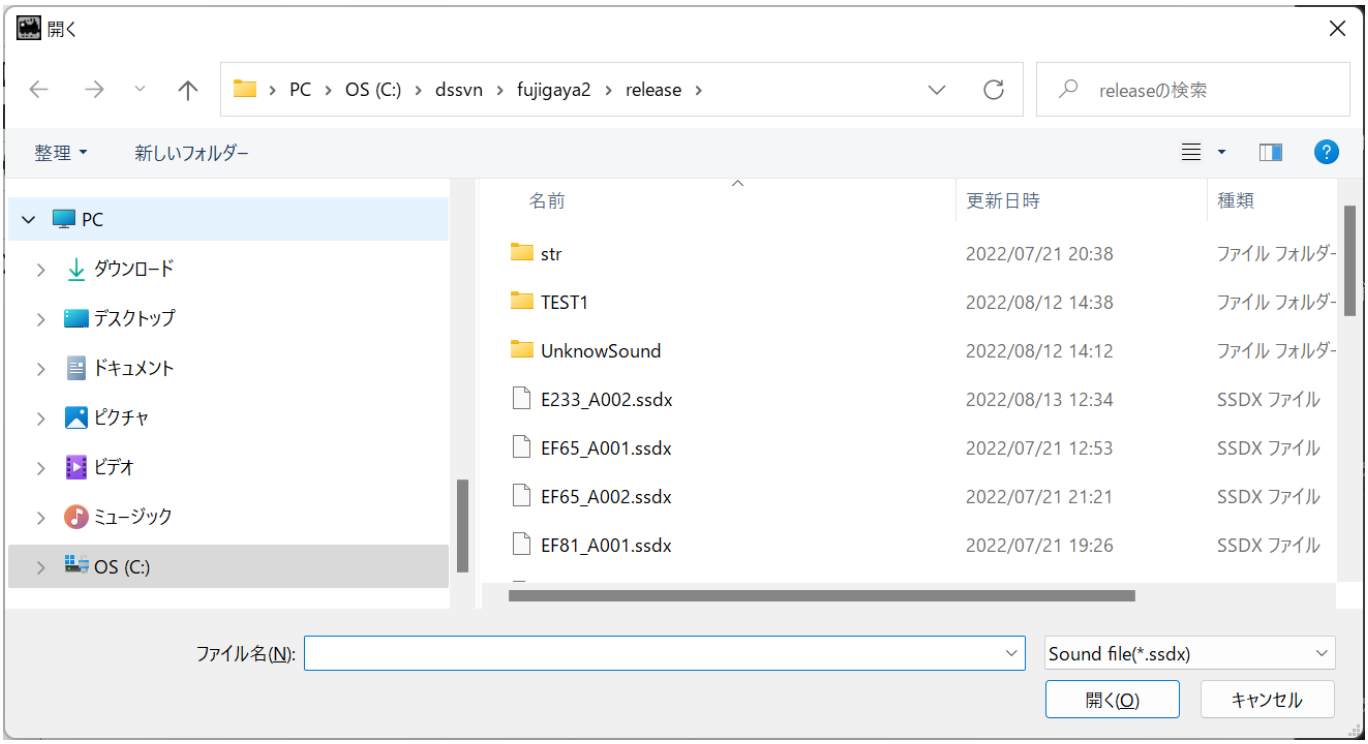
5.3. 使用するファイル

DSSPでは以下のファイルを扱います。

ファイルの拡張子	説明
ssdx	サウンドデータファイルです。
uf2	RP2040に接続されたフラッシュメモリに、USB経由で書き込むためのファイル形式です。USBメモリとしてSmileSoundデコーダを認識させた場合、D&Dで書き込みができます。
csv	サウンドフローファイルです。プロジェクトにインポート・D&Dすることで使用できます。DSSP上で作成もできます。
wav	サウンドフローで使用するWAVファイルです。プロジェクトにインポート・D&Dすることで使用できます。

5.4. サウンドデータを開く

DSSPでssdx形式のサウンドデータを開くことで、編集が可能になります。 サウンドデータのサンプルや、オープンサウンドデータで公開しているssdxファイルを使用することもできます。



5.5. サウンドデータを保存する

サウンドデータは、ssdxというファイル形式で保存することができます。このデータには、WAVファイルやサウンドフロー、CV関連データなど、サウンドデータの書き込みに必要なすべてのデータが含まれます。

注：暗号化機能などは現時点で搭載されていませんので、配布を行う等をする場合には、取り扱いにはご注意ください。

5.6. サウンドデータをデコーダに書込む

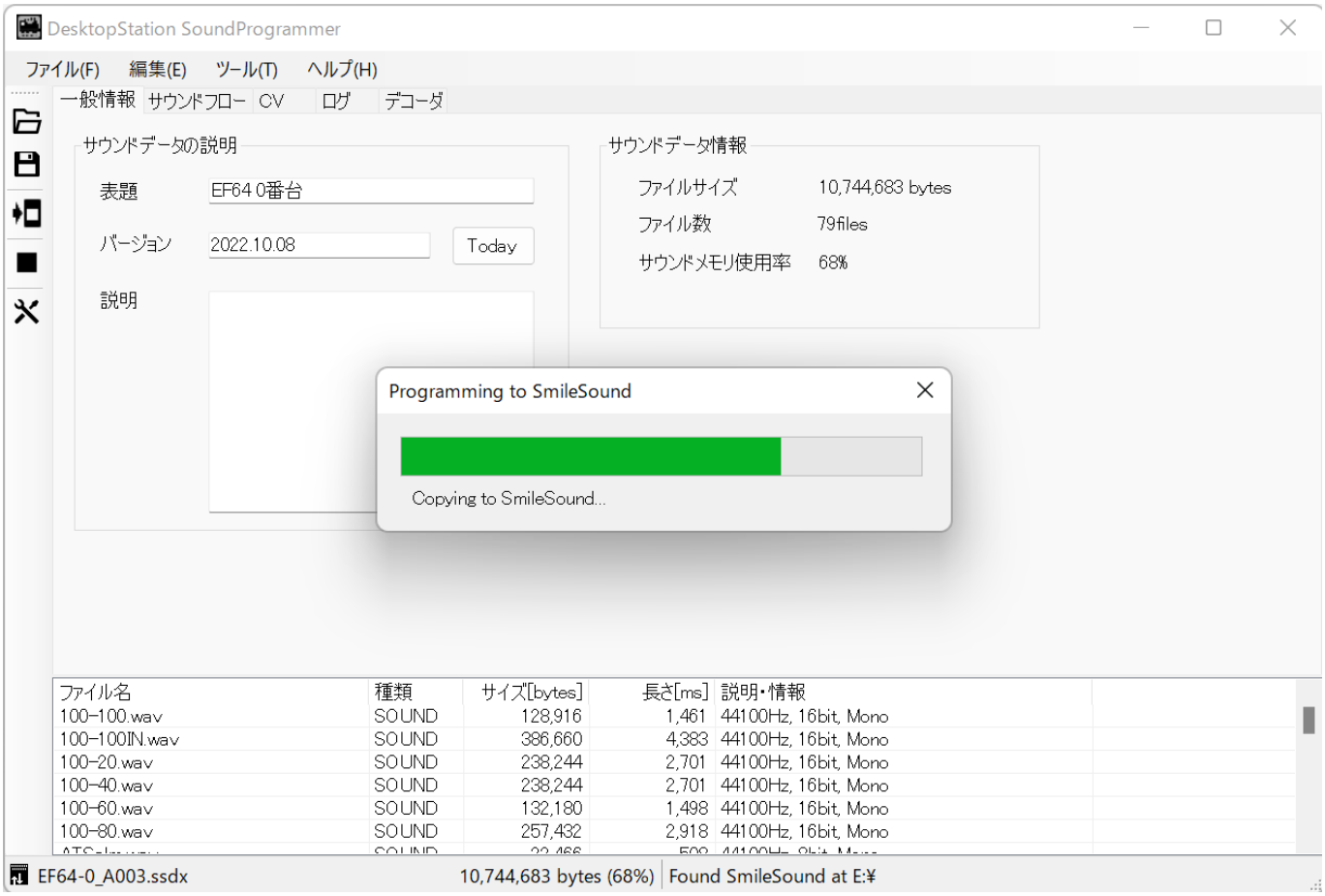
サウンドデータをデコーダに書き込む方法には、以下の方法があります。

- DSSPの書き込み機能を使う
- 手動でデコーダをUSBドライブ認識させて、DSSPでエクスポートしたUF2形式のサウンドデータを書込む(ssdx形式では書き込みできません)

基本的には、DSSPの書き込み機能を使用することを強く推奨します。

書き込みは、画面上左の  のアイコンをクリックするか、ファイルメニューから「SmileSoundにサウンドを書込む」を選びます。

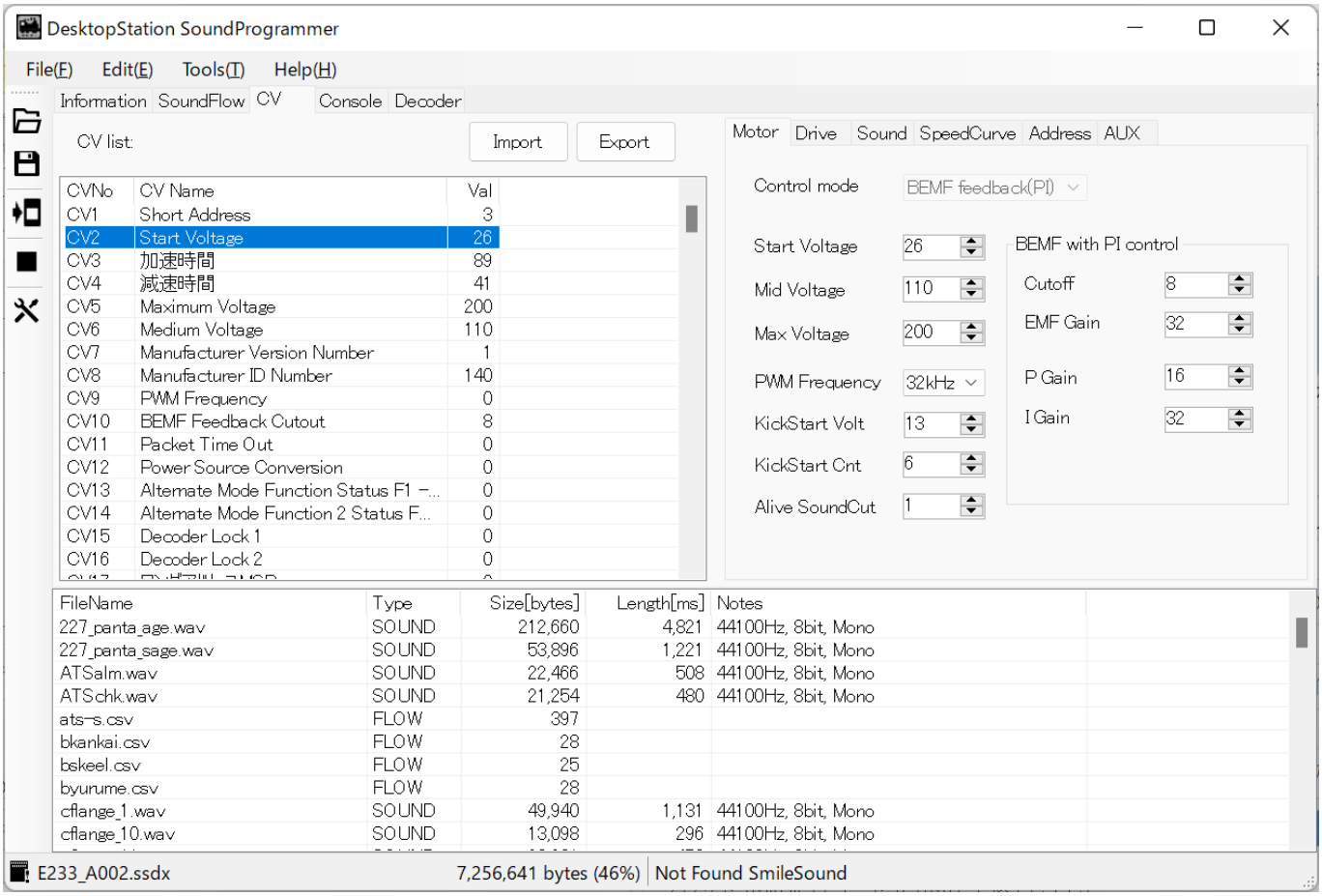
書き込み中は、以下のような画面が表示されます。



5.7. デコーダの設定を変更する

デコーダの設定は、CVを使って行われます。DSSPでは、サウンドデータにCVを紐づけて設定することができますので、CV設定画面を使って設定を行います。

ここで設定した情報は、サウンドデータをデコーダに書き込む際に一緒に書き込まれます。



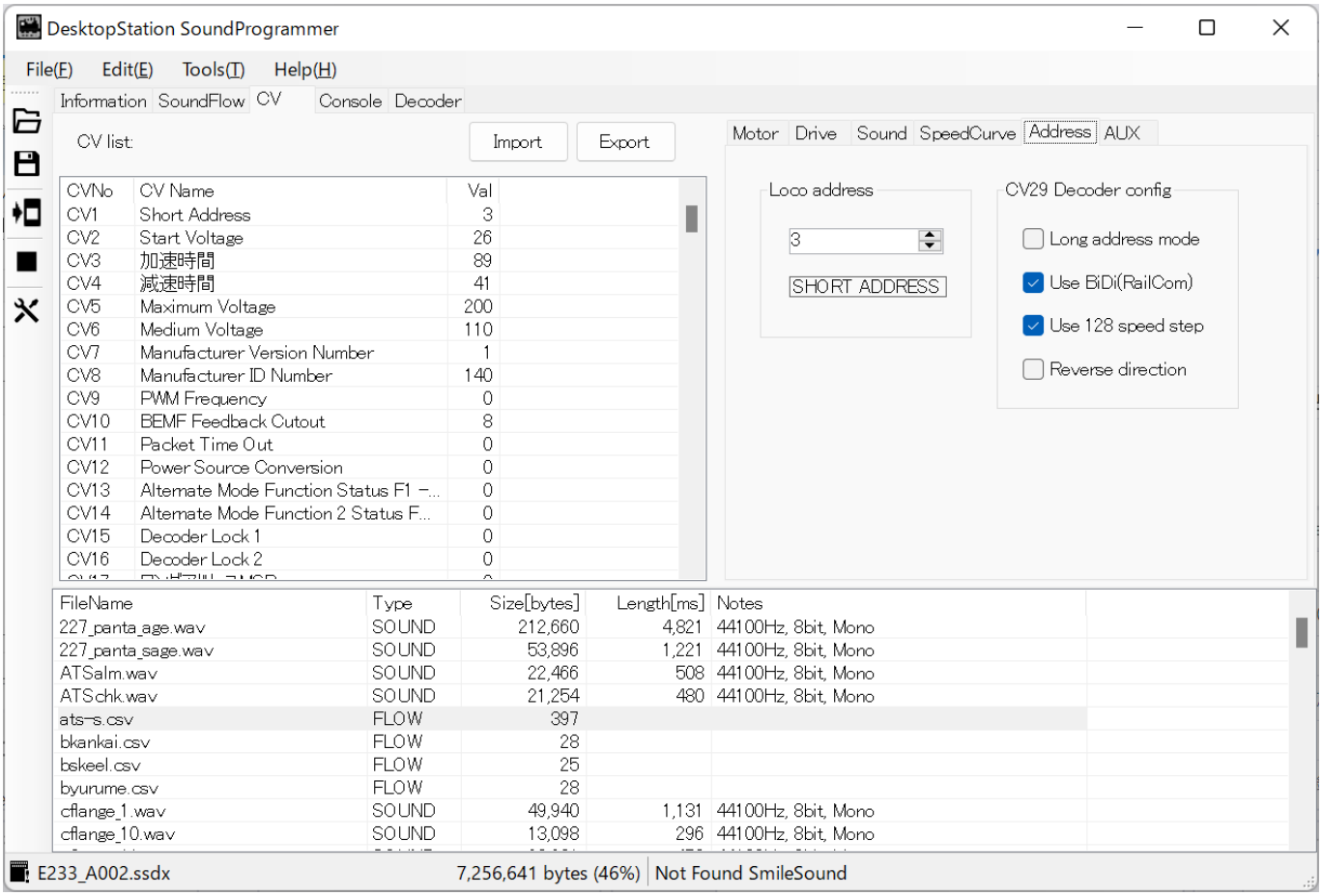
5.7.1. アドレスを変更する

アドレスを変更するには、CV編集タブの「アドレス他」タブから行います。「車両アドレス」に書かれた数字を変更する事で、デコーダの初期アドレスを設定できます。

DSSPでは、入力されたアドレスの数字によって以下のように定義しています。 手動で変更することもできますが、CV29の設定内容を理解したうえで行ってください。

アドレスの範囲	アドレスの種類
1-127	ショートアドレス
128-9999	ロングアドレス

警告 100-127のアドレスには設定しない事を強く推奨します。アメリカと欧州で、100-127をショートアドレスとして扱うか、ロングアドレスで扱うかが異なるためです。日本では、アメリカのコマンドステーションやデコーダが多く出回った過去があるため、アメリカのアドレス体系で100-127をショートアドレスとする場合が多くあります。しかし、欧州のコマンドステーションを使う場合、100-127はロングアドレスで操作することになるため、うまく動かなくなるケースがあります。

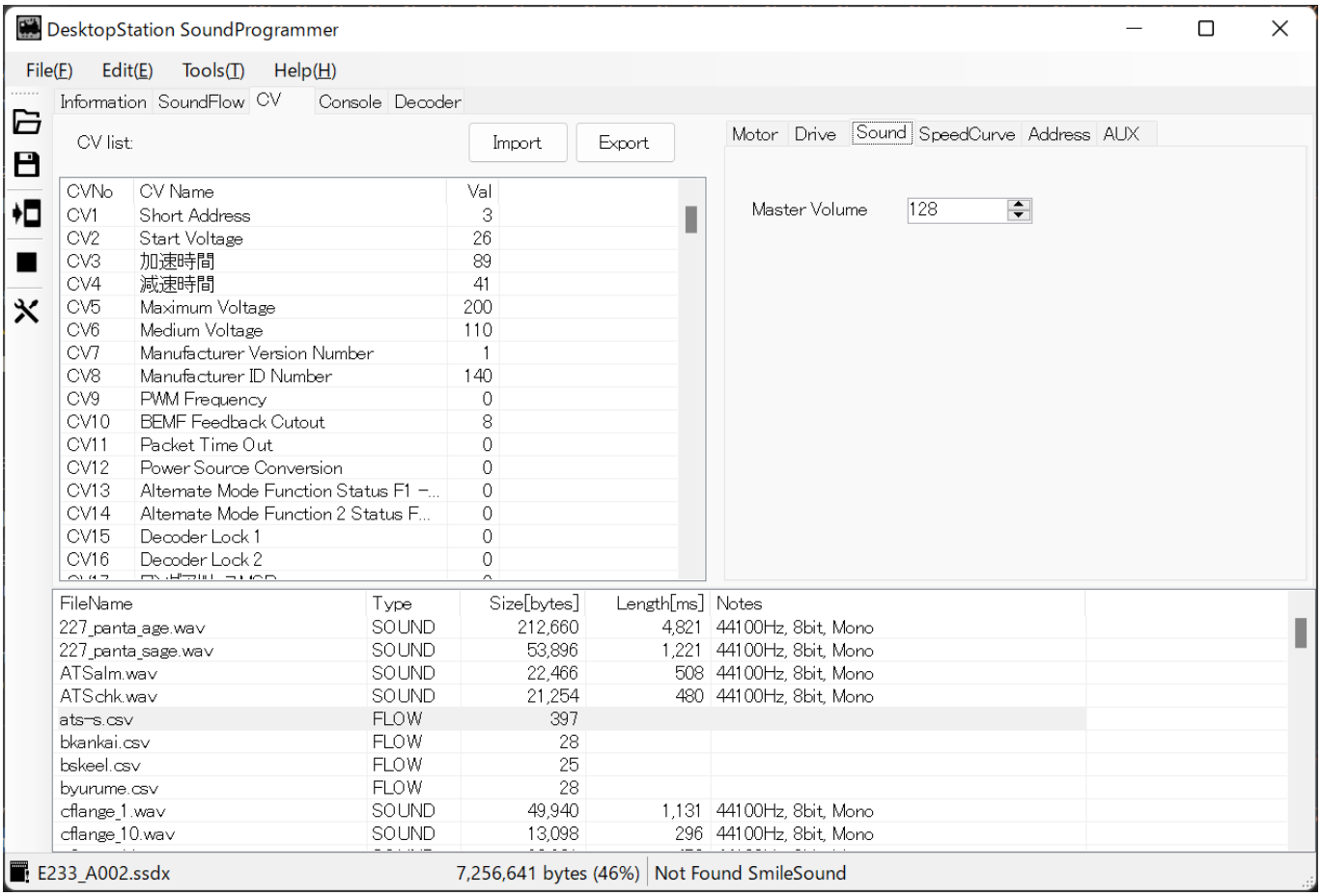


5.7.2. 音量を変更する

音量を変更するには、CV編集タブの「音量」タブから行います。「マスター音量」に書かれた数字を変更する事で、デコーダの初期音量を設定できます。

ポイント なお、オープンサウンドデータで公開されているサウンドデータでは、F7にマスター音量を運転中に操作する機能が含まれています。書き込んだ時のみに、このマスター音量が設定されます。書込んだ以後、運転中にマスター音量を書換えされることを考慮して設定してください。

WAVファイルやサウンドフローの個別の音量調整は、サウンドフローの個別音量調整コマンドvolにて行います。サウンドフローのvolコマンドは、マスター音量とは別に管理されるものとなります。



5.7.3. モータを調整する

モータに合わせた調整は、CV機能を活用して行います。まず、CV編集タブのモータ設定を開きます。

CV番号	Nゲージ汎用	GM コアレス	HO プラ機関車
線路電圧	12V	12V	15V
CV2・開始電圧	8~30	8	16
CV10・カットオフ速度	2	2	2
CV54・BEMF電圧係数	旧国電 128,\ 特急 64,\ 新幹線 32	旧国電 128,\ 特急 64,\ 新幹線 32	28
CV55・比例ゲイン	16※1	16	16
CV56・積分ゲイン	32※1	32	32
CV64・キック上限速度	0	0	0
CV65・キック電圧・回数	0	0	0

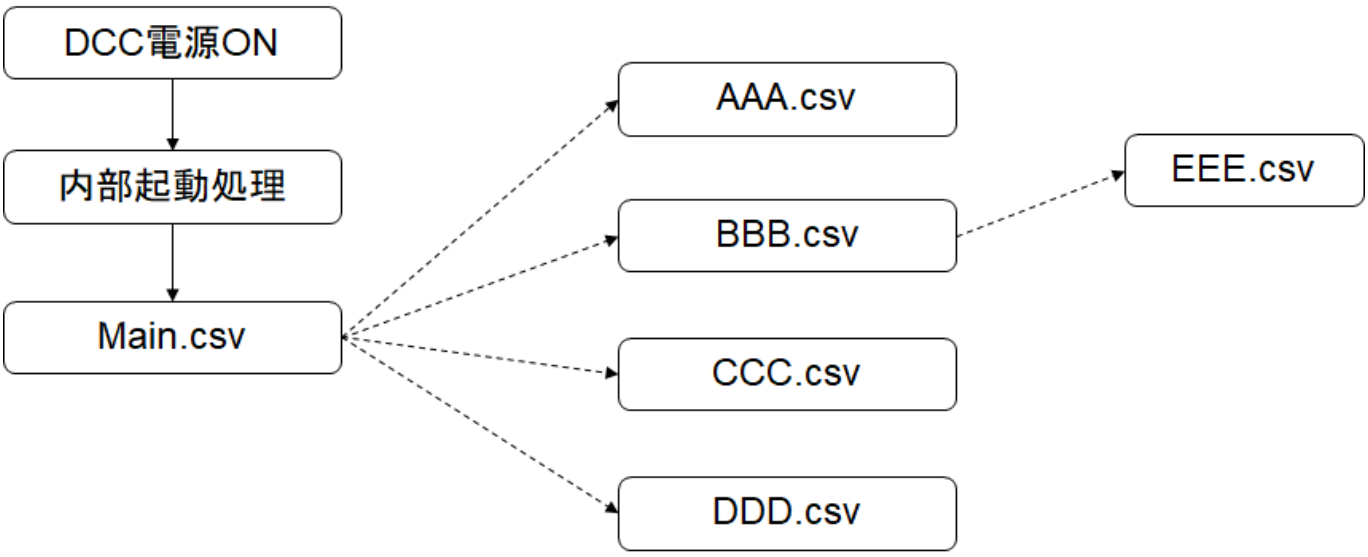
一旦減速するような挙動をした場合は、CV55,CV56を8と16のような小さい値に変えて試してみてください。CV=2も25や30のような大きな値の方が起動がスムーズになる可能性があります。

5.8. サウンドデータを編集する

ここでは、DSSPを使って、サウンドデータを編集する方法を説明します。 サウンドデータは、サウンドフローと呼ばれるCSVファイルに、コマンドを記述して、様々な挙動を作り込んでいきます。挙動とは、走行音の動き、CPの動き、ファンクションとの連動、LED・ライトの動きなどです。

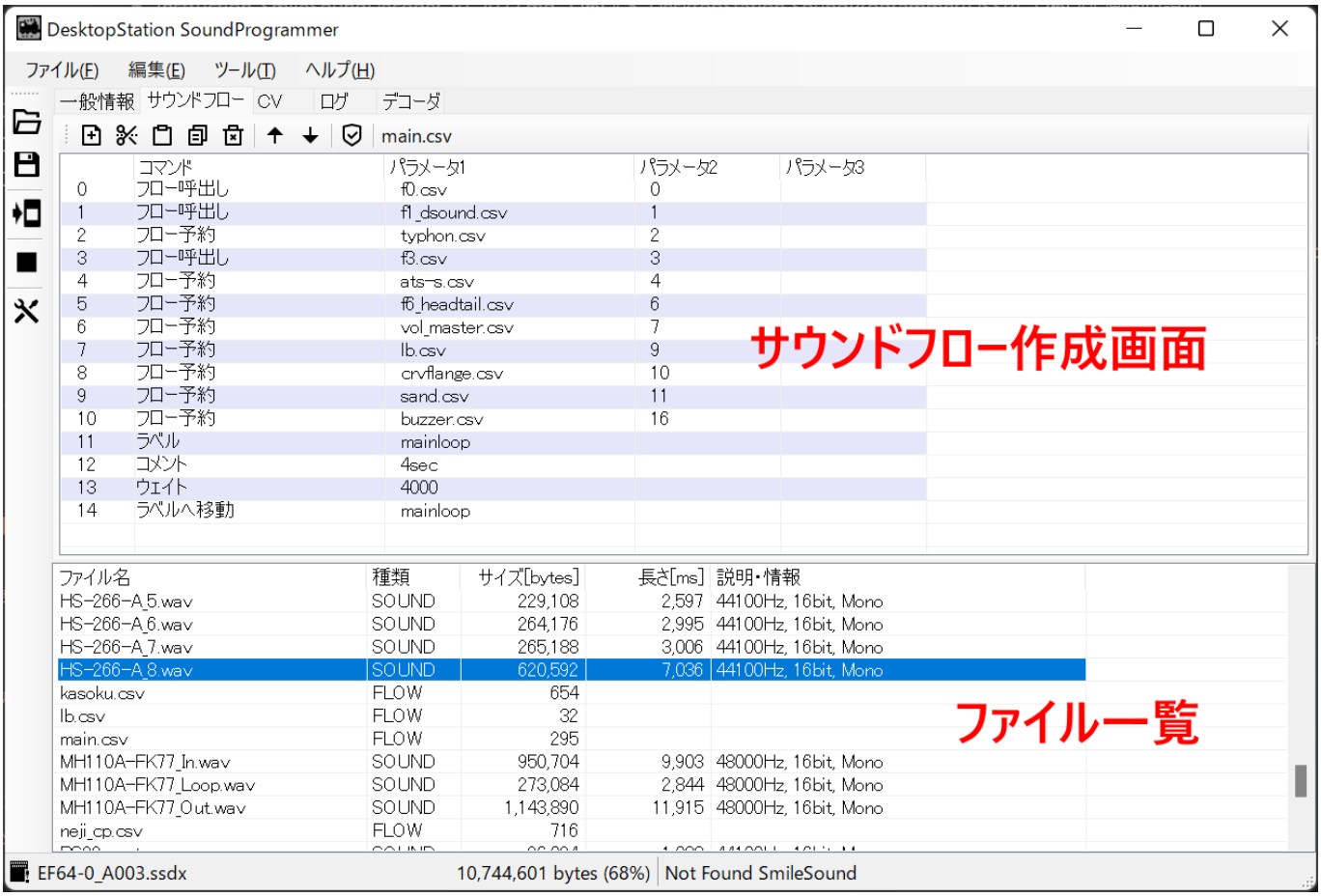
サウンドフローは、同時に１０以上を動かすことができます。同時に動くとき、ユーザーは意識する必要はありません。SmileSoundデコーダのファームウェアが、サウンドフローの同時動作を制御してくれます。

最初にmain.csvが自動で実行されます。main.csvで、他のサウンドフローを呼び出す処理を書くことで、様々なサウンドの動きを表現できます。

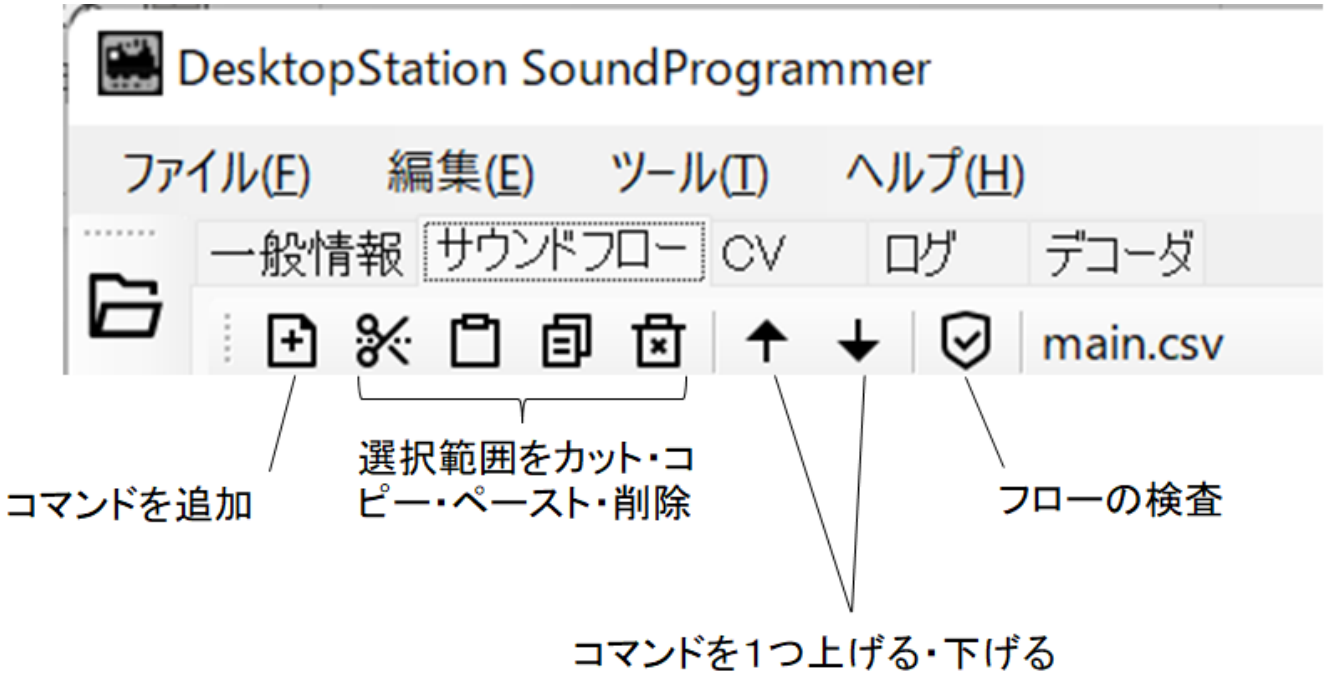


5.8.1. サウンドフローの編集画面

DSSPでサウンドフローを編集する際には、以下のような画面構成となります。



サウンドフローを開いたとき、表示される上部のツールバーの説明は、以下の通りです。

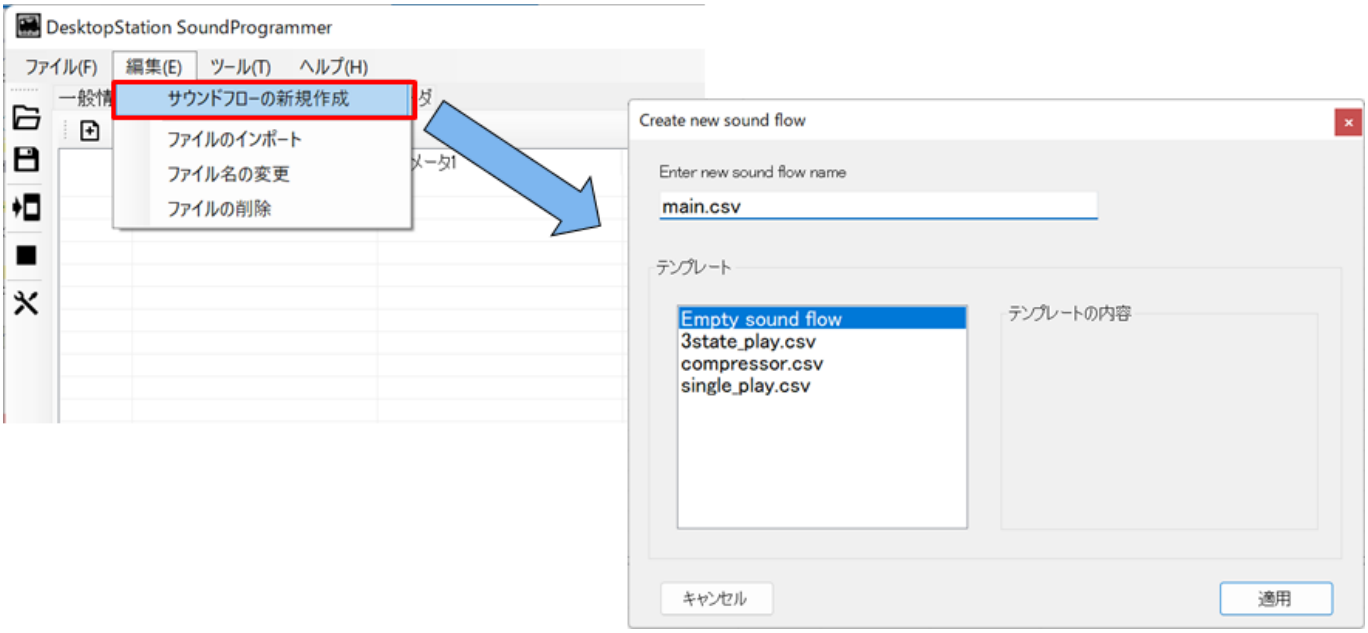


5.8.2. サウンドフローを新規に追加する

新規のプロジェクトを作成した場合、最初に、main.csvを作成します。既にmain.csvがある場合には、お好きな名前でサウンドフローを新規に追加できます。テンプレートもありますので、それをベースに作成いただ

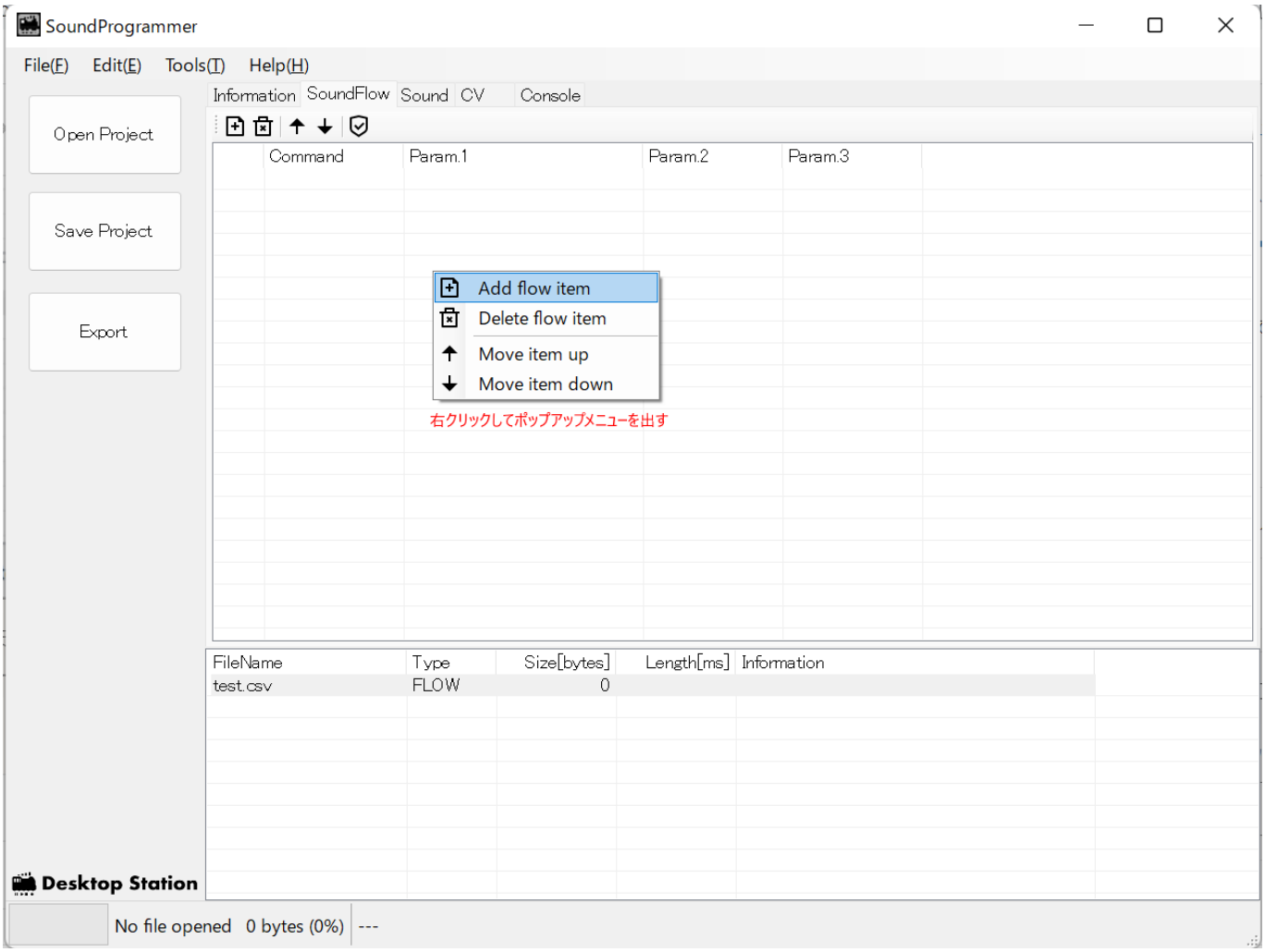
いても構いません。

プロジェクトの中に、main.csvがないと、サウンドフローが正常に機能しなくなります。

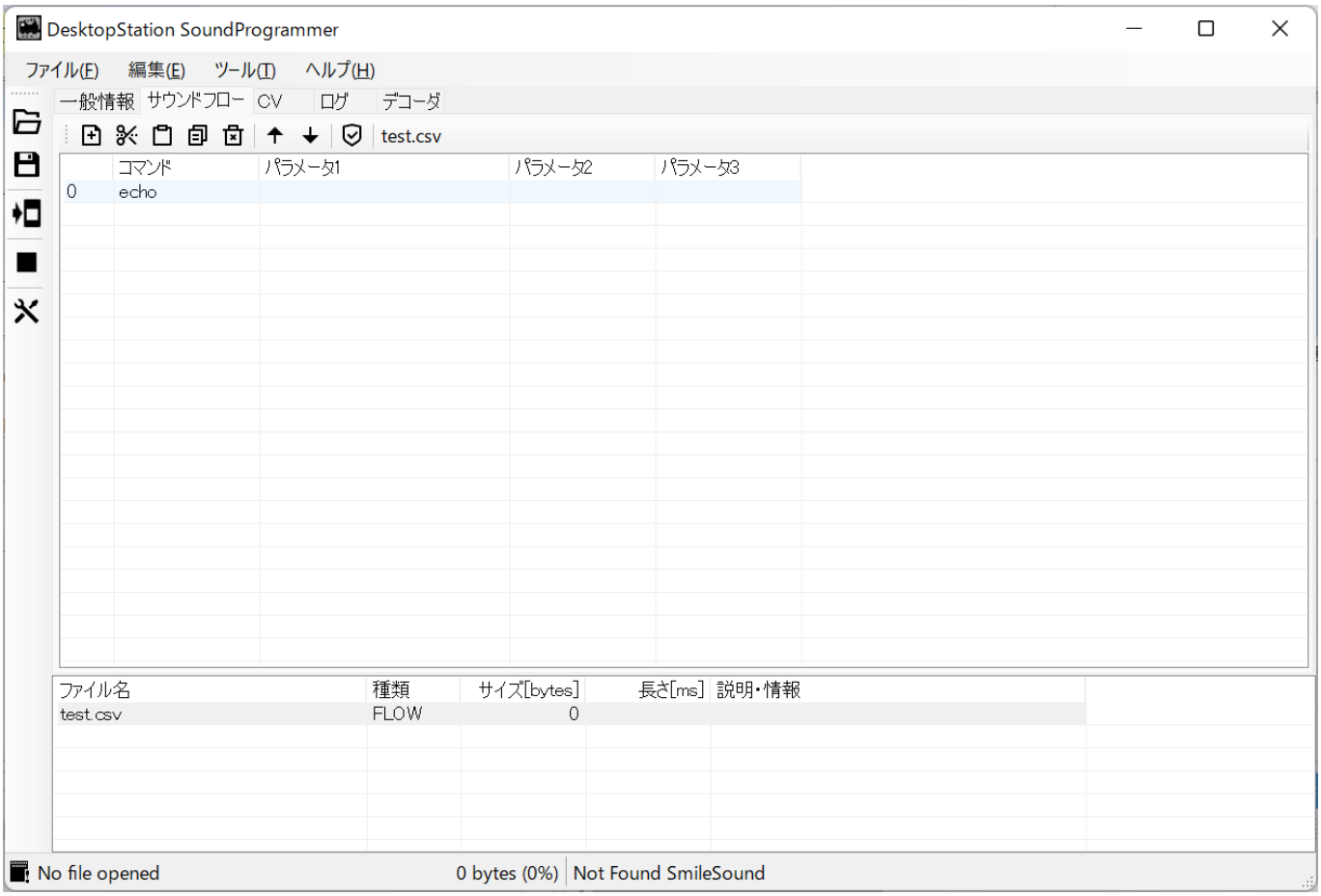


5.8.3. コマンドを追加する

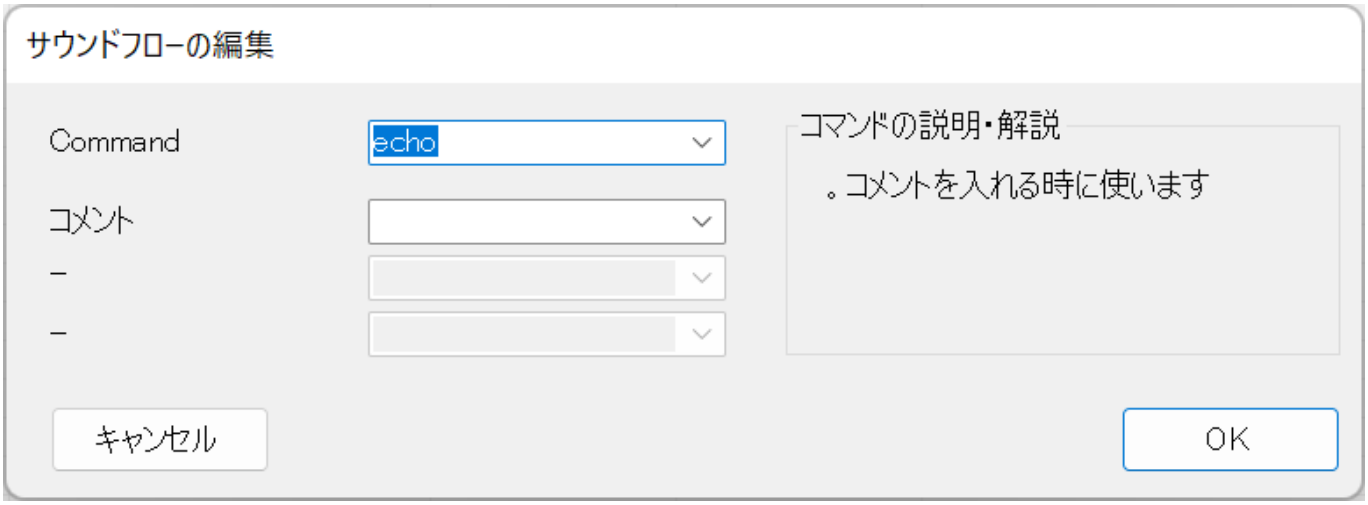
コマンドを入力していきましょう。サウンドフローのリストの上で、右クリックすると、ポップアップメニューが出ます。 
”フローに新規アイテムを追加”をクリックしてみましょう。



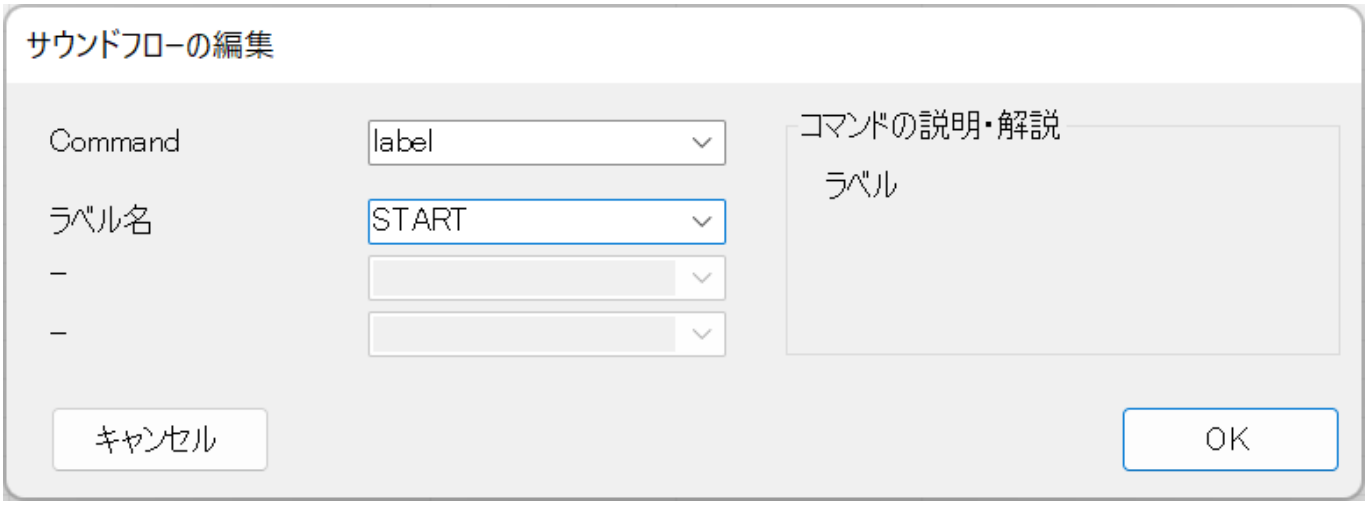
すると、"echo"というコマンドが追加されました。



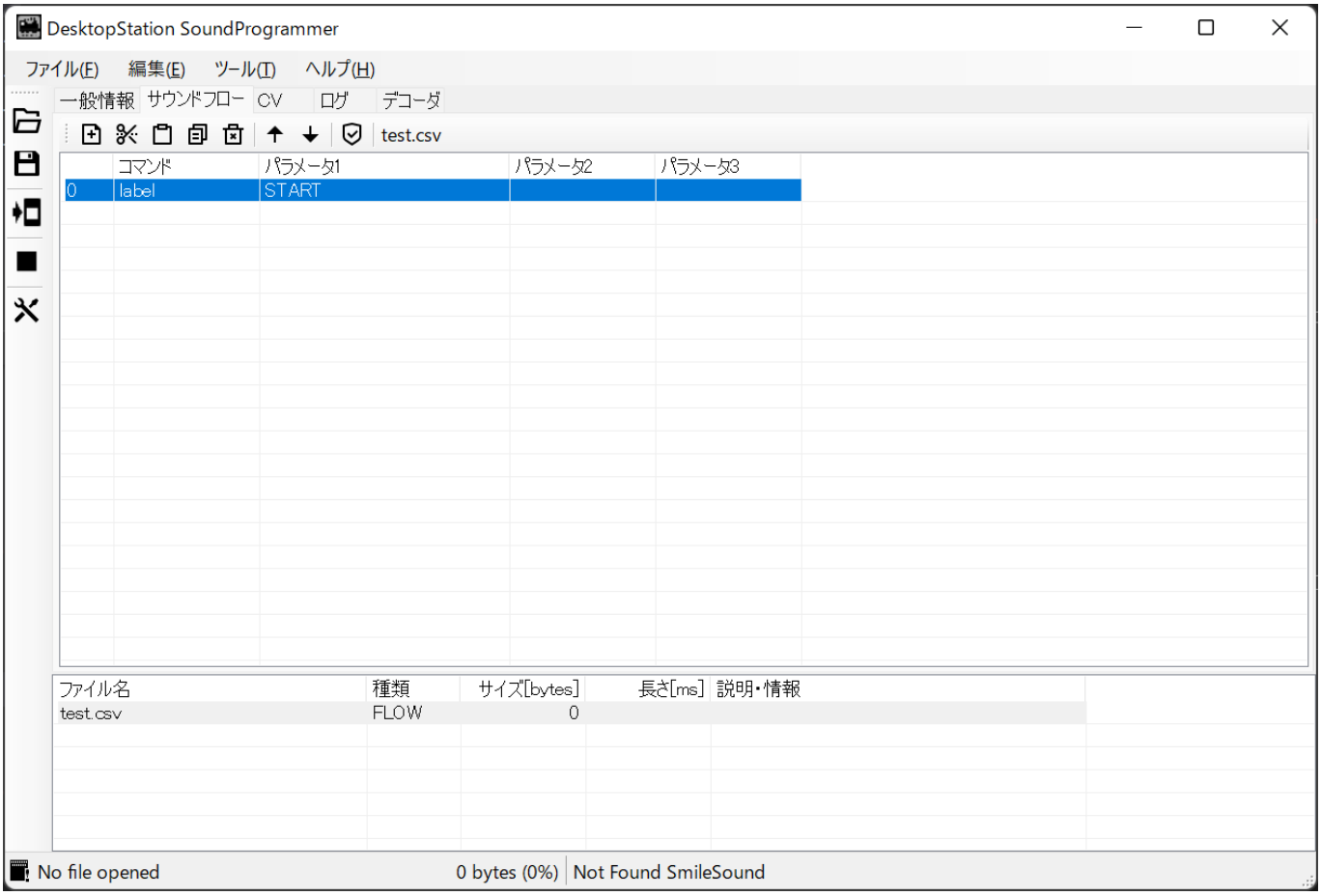
追加されたコマンドをダブルクリックすると、コマンドの編集画面が出てきます。コマンドとパラメータを変えることで、サウンドフローを作り上げていきます。



コマンドにlabel, 引数にSTARTと入れて、OKを押します。



サウンドフローの一覧に反映され、コマンドが変更されました。



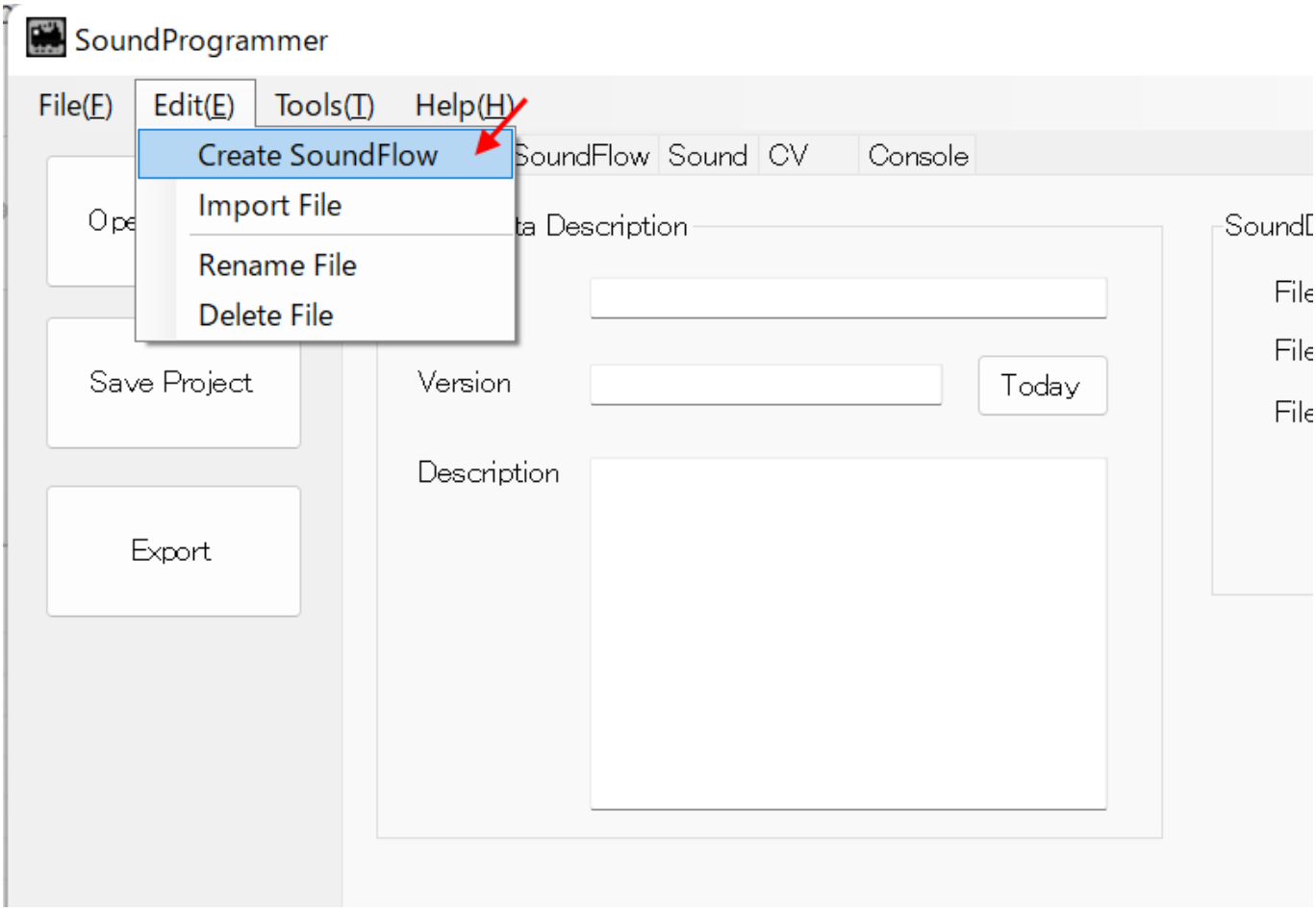
このようにして、コマンドを追加して、サウンドフローを作成していきます。

5.8.4. 再生される音ファイルを差し替える

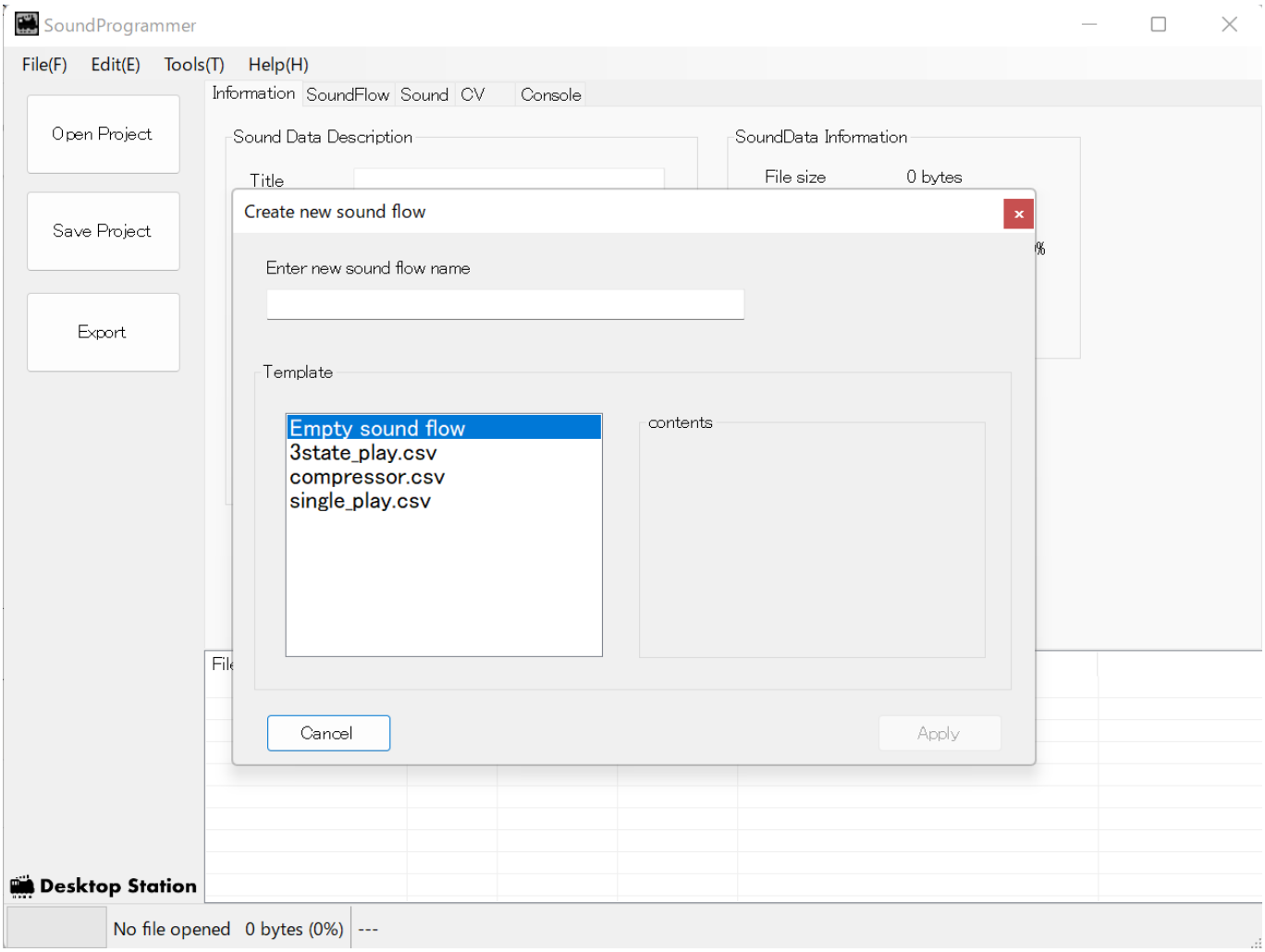
まず、差し替え先のWAVファイルをDSSPにD&Dするか、インポートを行います。 その後、変更するサウンドフローをファイルリストから開き、該当する再生コマンド(playコマンドなど)をダブルクリックし、ファイル名を変更してください。

5.8.5. サウンドフローを追加する

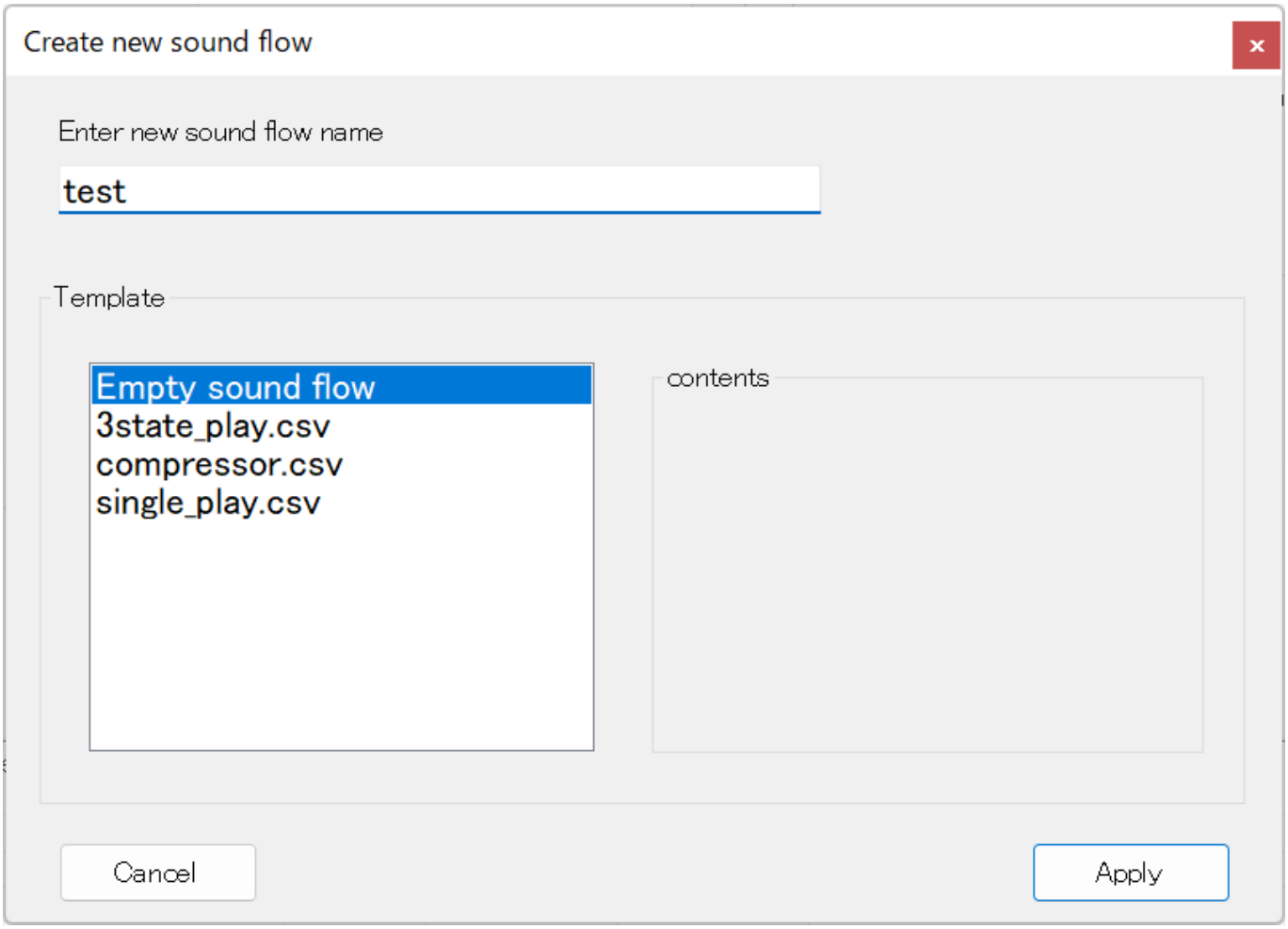
Create Sound Flowをメニューから選びます。



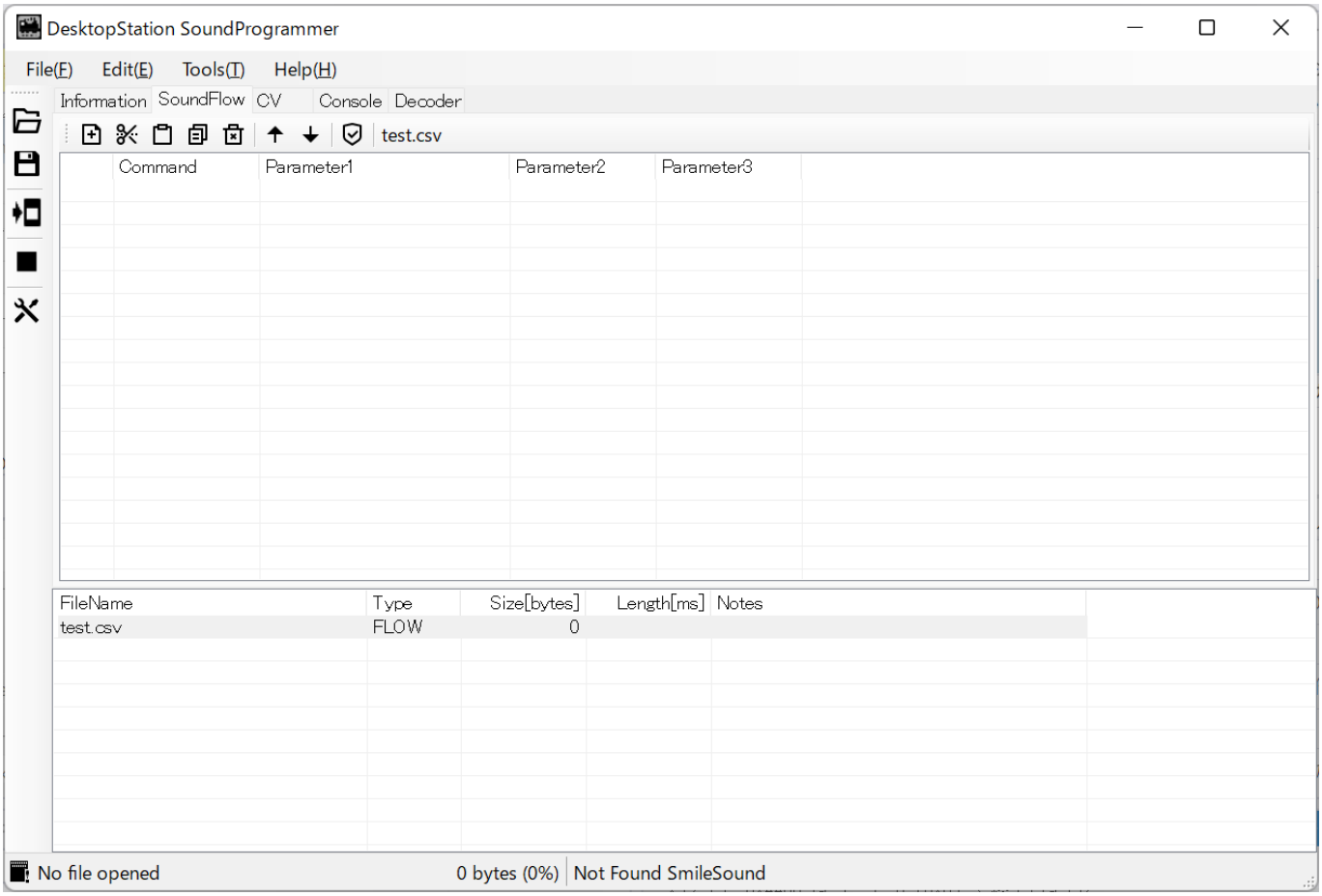
すると、サウンドフローの名前を付ける画面と、テンプレートから選ぶ画面が出てきます。



“test”というサウンドフローを作ってみます。test と入力して、Applyボタンを押しましょう。



すると、test.csvというファイルが作成され、サウンドフローの編集画面に画面が切り替わります。中身が空っぽなので、空のサウンドフローが作られています。



5.8.6. 音を鳴らす

音を鳴らすコマンドは、以下の３種類あります。

- play
- plyx
- sply

音を止めるコマンドは、stopコマンドの１つしかありません。

それぞれ、使い方が大きく異なります。

5.8.7. 分岐とジャンプを組み合わせる

5.8.8. 変数を使う

サウンドフローには、変数が用意されており、ifコマンド等で使用できます。代入可能な変数は、letコマンドなどを使って値をセットすることができます。

使用例としては、他のサウンドフローと連携して動かす場合はshare変数を使います。１つのサウンドフロー内で、条件や状態が複雑に変化し続ける場合に、他のサウンドフローと連携する必要のない変数が必要な場合には、local変数を使用します。

|定義済変数|代入可否 |変数名| 説明 |使用例| |:--|:--|:--|:--| ||local|可|ローカル変数 |呼び出しているサウンドフロー内で使用できる変数 |local1,local2,・・・ local8| |share|可|共有変数 |デコーダ全体のサウンドフローで使用できる変数 |share1,share2,・・・share8 | |spd|不可|現在速度|車両が現在走行中の速度。加減速中は速度が変

化していきます。 |spd | |ref|不可|指令速度|コマンドステーションからの指令速度。車両が加減速中では現在速度spdとズレが生じます。 |ref | |fnc|不可|ファンクション|サウンドフローに割り付いているファンクション状態を0(OFF)または1(ON)で示します。特定のファンクション番号の状態は確認できません。 |fnc| |aux|不可|AUX状態|AUX出力状態を0(OFF)または1(ON)で示します。 |aux1,aux2・・・aux6 | |tmr|可|タイマ|呼び出しているサウンドフロー内で使用できるタイマ。1以上の値をセットすると、1秒ごとに1ずつ減ります。0になると、値を減らすことを止めます。| tmr1,tmr2・・・tmr4| |acc|不可|加減速度| 加速しているとき正の値, 減速しているとき負の値となります。一定速度で走行中はacc==0となります。 |acc | |dir|不可|進行方向| 0のとき直進、1のとき後進 |dir | |cv|不可|SoundCVの設定値 |SoundCVはCV155-CV170の範囲です。
|cv1,・・・,cv16 |

6. サウンドフロー仕様

6.1. サウンドフローとは

サウンドフローは、状態遷移を表現するCSVフォーマットベースのスクリプトです。サウンドフローを活用する事で、DCCコマンドに従って、サウンドやモータ制御、AUX等の操作を行うことができます。

サウンドデコーダの基本的な制御を、全てサウンドフローにて行う事で、ユーザーカスタマイズ性を高め、サウンドデコーダを高機能化・高性能化することを狙っています。サウンドに限らず、DCCデコーダの様々な機能をスクリプトで表現できるようになります。

- サウンドの再生・停止・ループ等の制御
- サウンドのエフェクトの制御（フェードインアウト、再生スピード、音量等）
- サウンドの再生と、モータ制御の連動(VVVF,ディーゼルサウンドの連携)
- CVの初期設定
- ファンクションマッピングの設定
- AUXの制御
- 速度制限(ATCやATSの模擬)

サウンドフローは、最大１６フロー（スクリプト）を並列に実行することができます。つまり、同時に複数の処理を同時並列的に実行することができます。実際には、数ms程度のばらつきで各サウンドフローの各スクリプト行を順番に実行していきますが、十分高速なため、人間には同時に動いているように見えます。

単純な警笛のサウンドフロー例は以下です。これは状態遷移図でも表すことができます。

```
【警笛のサウンドフロー例】
echo,startflow
label,START
if,fnc==1,PLAY_ON,
goto,START
label,PLAY_ON
play,seibuaw_in.wav,0,0
echo,playwav
play,seibuaw_loop.wav,1,0
label,PLAY_LOOP
if,fnc==0,END,PLAY_LOOP
label,END
play,seibuaw_out.wav,0,0
echo,exitflow
goto,START
```

コマンド体系は、後述しています。

6.2. サポートするファイル

サウンドフローでは、以下のファイルをサポートしています。

ファイル形式	説明
wav	音源ファイルです。無圧縮(LPCM)、8bitまたは16bitの整数型形式で、44.1kHz, 32kHz, 22.05kHz, 16kHz, 11.025kHz, 8kHzのサンプリングレートをサポートしています。
csv	サウンドフローの記述ファイルです。カンマ区切りで、ダブルクオーテーションはサポートしません。UTF-8形式としてください。

6.3. サウンドフローの動き

サウンドフローは、常にループを作って状態を監視するように設計します。つまり、ラベルを用いて状態を作り、ifコマンドで条件に合った場合、他の状態（ラベル）へジャンプし、様々な処理を実行するという流れです。

このため、サウンドフローはムリに終わらせる必要はありません。警笛等の通常のサウンドファイルの拡張のようなサウンドフローの場合には終わらせても問題ありません。

たとえば、以下のようなサウンドフローを実行すると、最初にflow1.csvを呼び出して実行した後、label,mainloopと、goto,mainloopの間を永久ループで繰り返します。この時、5秒ごとにシリアル通信で5secというメッセージが出力されます。

```
call,flow1.csv
label,mainloop
echo,5sec
wait,5000
goto,mainloop
```

サウンドフローは並列で実行されるため、その他のサウンドフローの実行は妨げられません。

6.4. CSVスクリプトの記述

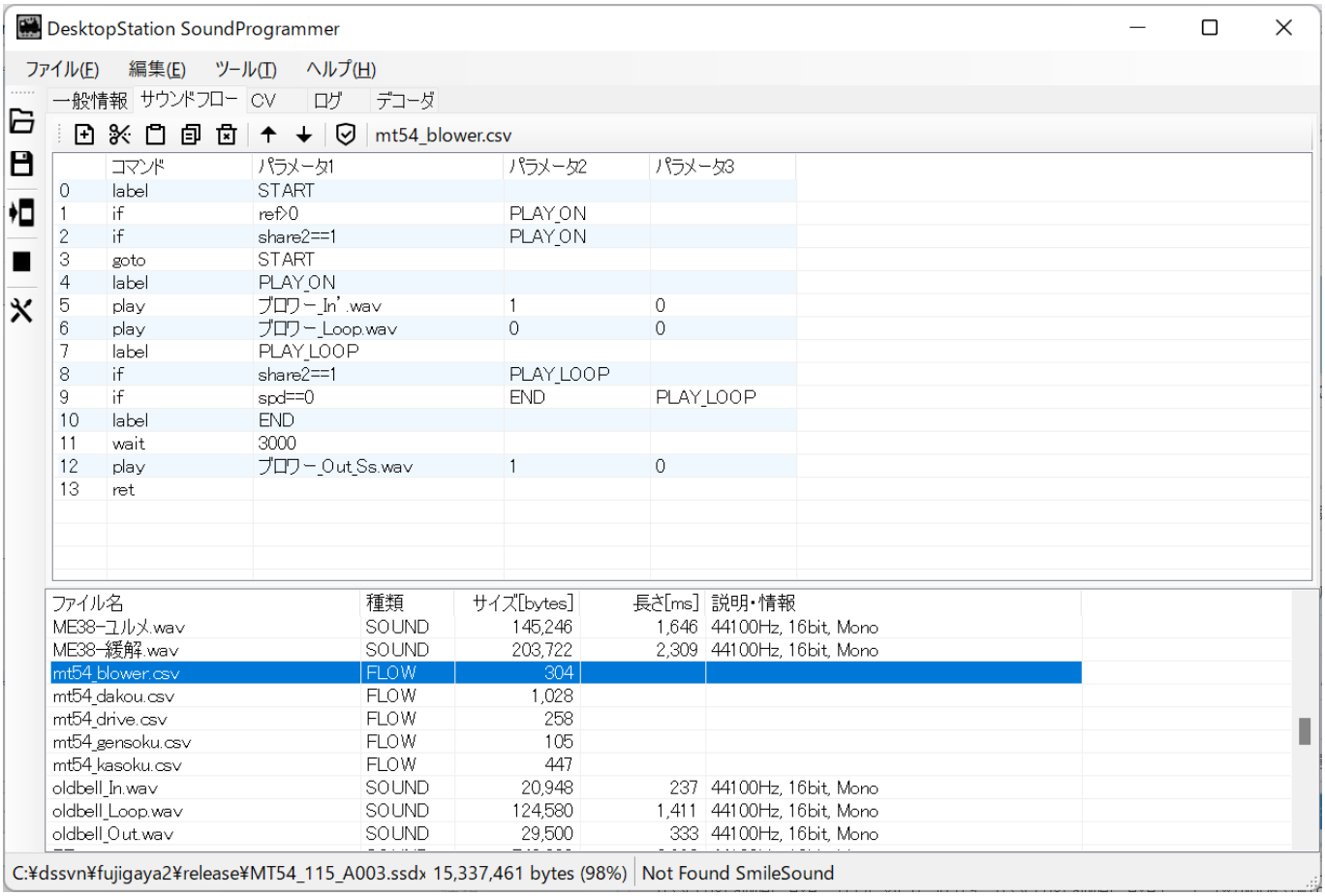
CSV形式とするが、通常のCSVと異なり以下の注意点がある。

- ファイル名に制限はありません。
- UTF-8としてください。
- ファイル名のみ、日本語は用いても構いません。その他は半角英数字・記号のみ
- ファイル名はダブルクォーテーションで括らない("")
- 128行まで（メモリ節約の都合上。今後改善により拡張の可能性あり）
- EOFの行にはコマンドは入れない。空白とする

7. サウンドフローの解説

7.1. 用があるまで待機する

SmileSoundデコーダのサウンドデータを作るときに、重要になるのがサウンドフローです。サウンドフローの中身を、どのように書いて行けば良いか、説明していきます。まずは、0-4行目までに絞って解説します。



まず、サウンドフローの中身に注目していきましょう。ここでは、ブロアーの動きを示します。サウンドをONすると、このブロアーのサウンドフローが動作するように、作られているとします。

	コマンド	パラメータ1	パラメータ2	パラメータ3
0	label	START		
1	if	ref>0	PLAY_ON	
2	if	share2==1	PLAY_ON	
3	goto	START		
4	label	PLAY_ON		
5	play	ブロワー_In'.wav	1	0
6	play	ブロワー_Loop.wav	0	0
7	label	PLAY_LOOP		
8	if	share2==1	PLAY_LOOP	
9	if	spd==0	END	PLAY_LOOP
10	label	END		
11	wait	3000		
12	play	ブロワー_Out_Ss.wav	1	0
13	ret			

最初に、labelというコマンドがあります。これは、「この位置に、目印を置くよ！」という意味になります。目印を置くだけなので、特に音や車両の動きに変化はありません。

次に、ifというコマンドが出てきます。ちょっと難しくなってしまいましたね。これは、「もし、〇〇の条件が成立したら××の目印(label)に移動してください。成立しなかったら、△△の目印(label)に移動してください。」というコマンドです。細かく説明していきます。

〇〇には、パラメータ1に書いてある「ref>0」があります。プログラマではない方は、とっつきにくいのかもしれませんが、「refが0より大きい数だったら」という意味になります。refというのは、SmileSoundデコーダでは、「速度の指令値」になります。ユーザーが指定した速度っということです。つまり、「ref>0」は「デコーダに走ってくれとユーザーが指示した」という意味になります。

××には、パラメータ2、△△にはパラメータ3を使います。つまり、「もし、〇〇の条件が成立したら××の目印(label)に移動してください。成立しなかったら、△△の目印(label)に移動してください。」と説明しましたが、読み替えると、「もし、デコーダに走ってくれとユーザーが指示したの条件が成立したらパラメータ2の目印(label)に移動してください。成立しなかったら、パラメータ3の目印(label)に移動してください。」ということになります。ちなみに、パラメータ3は省略する事ができて、何も書かない場合は、移動しない（=次に進む）となります。

もう一つifがあり、「share==1」などと書いてあります。これは、後々説明していきます。

そのあと、gotoというコマンドがあります。これは、labelで設定した目印に、移動してくれ、という意味です。ここではSTARTとパラメータ1に書いてあるので、STARTに飛んでいきます。つまり、最初に戻るのです。

	コマンド	パラメータ1	パラメータ2	パラメータ3
0	label	START		
1	if	ref>0	PLAY_ON	
2	if	share2==1	PLAY_ON	
3	goto	START		
4	label	PLAY_ON		
5	play	ブロワー_In'.wav	1	0
6	play	ブロワー_Loop.wav	0	0

ブロアーのサウンドフローでは、最初は何らかしらの指示が来るまでは、待機しているという処理を、0～3行目を使って表現していたのであります。

8. コマンド一覧

8.1. aux

ヘッドライト・テールライト、AUXの出力のON/OFFを行います。

コマンド	引数1	引数2	引数3
aux	AUX番号(0,255,1-8)	AUX操作(0 or 1)	-

AUX番号	定義
0	ヘッドライト
255	テールライト
1	AUX1
2	AUX2
3	AUX3
4	AUX4
5	AUX5
6	AUX6
7	AUX7
8	AUX8

【記入例】

```
aux,0,1
aux,255,0
```

8.2. call

他のサウンドフローを呼び出して実行します。他のサウンドフローを呼び出しても、並列実行されるため、動作の影響を受けません。何らかの条件で、複数のサウンドを出す時に使用します。たとえば、発車時のブレーキ緩解音や警笛など。

コマンド	引数1	引数2	引数3
call	サウンドフローファイル(csv)	割り付けるファンクション番号(0-28)	-

flow1.csvを呼び出す例です。

ファンクション番号を特に使用しない場合は、0としてください。サウンドフロー内で、ifコマンドを使ってfnc変数を操作しなければ、特に支障はありません。

```
【記入例】
call,flow1.csv,2
```

8.3. date

他のサウンドフローを呼び出して実行します。ただし、割り付けられたファンクションがOFFからONになったときだけロードして動作します。警笛やドアなど、シンプルなファンクションに使用します。

他のサウンドフローを呼び出しても、並列実行されるため、動作の影響を受けません。何らかの条件で、複数のサウンドを出す時に使用します。たとえば、発車時のブレーキ緩解音や警笛など。

コマンド	引数1	引数2	引数3
date	サウンドフローファイル(csv)	割り付けるファンクション番号(0-28)	-

flow1.csvを呼び出す例です。

ファンクション番号を特に使用しない場合は、0としてください。サウンドフロー内で、ifコマンドを使ってfnc変数を操作しなければ、特に支障はありません。

```
【記入例】
date,flow1.csv,2
```

8.4. echo

コメント行を示します。特に処理は行いません。無視されます。

コマンド	引数1	引数2	引数3
echo	コメント	-	-

コメントを記載した例。

【記入例】
echo,コメントです。

8.5. exit

サウンドフローを終了し、使用していたスロットを開放します。再開はできません。なお、exitを記入しなくても、サウンドフローの最後に到達した場合にも自動でexitと同じ処理が動作します。

終了させたくない場合は、goto等をスクリプト末尾に置いてください。

コマンド	引数1	引数2	引数3
exit	-	-	-

コメントを記載した例。

【記入例】
exit,コメントです。

8.6. if

条件が整うと、指定のラベルにジャンプします。 偽の時の行き先ラベルを省略すると、if文に続いて次のコマンドを実行します。複数のifコマンドを用いて状態遷移を作りたいときに使用してください。

コマンド	引数1	引数2	引数3
if	条件	真の時の行き先ラベル	偽の時の行き先ラベル(省略可)

条件には、様々な変数が使用できます。

変数の種類	変数の説明	使用可能範囲	値の範囲
local	フロー内変数		
share	共有変数	デコーダ全体のサウンドフローで使用できる変数	
spd	デコーダの現在速度	0-255	
ref	指令速度	0-255	
fnc	ファンクション	0-32	
aux	AUX状態	0-6, 255	
tmr	タイマ		
acc	加減速度	-255~+255	
dir	進行方向	0-1	

変数の種類	変数の説明	使用可能範囲	値の範囲
cv	CVの設定値		

式に使用できる等号・不等号は以下の通りです。

変数の種類	変数の説明
==	等しい
>	大なり(超)
>=	以上
<	小なり(未満)
<=>	以下
!=	等しくない

ファンクション操作によってジャンプする例。ここでは、監視ファンクションがONになった時の例です。

```
【記入例】
label,START
if,fnc==1,SOUND_ON,START
～～～
label,SOUND_ON
～～～
label,SOUNDLOOP
if,fnc==0,SOUND_OFF,SOUNDLOOP
～～～
label,SOUND_OFF
```

8.7. goto

行き先ラベルにジャンプします。

コマンド	引数1	引数2	引数3
goto	行き先ラベル名		

TESTにジャンプする例

```
【記入例】
goto,TEST
～～～
label,TEST
```

8.8. label

gotoコマンドやifコマンドからジャンプしてくる位置を設定するものです。

コマンド	引数1	引数2	引数3
label	ラベル名		

TESTにジャンプする例

```
【記入例】
goto,TEST
～～～
label,TEST
```

8.9. let

ユーザーが設定可能な変数に、値を代入するためのコマンドです。簡単な数式のみ対応します。

コマンド	引数1	引数2	引数3
let	代入式(後述)		

代入式に使用できる左辺の変数は、以下となります。

変数の種類	変数の説明	使用可能範囲	値の範囲
local	フロー内変数	local0~local3	0-255
share	共有変数	デコーダ全体のサウンドフローで使用できる変数	share0~share3
tmr	タイマ	tmr0~tmr1	0-255

代入式は、以下のような文を示します。

```
local2=5+1
```

カッコは使えません。

```
× let,share3=(1+9)*2
```

右辺の数値を、変数にしても構いません。

```
let,share1=share1+10
```

右辺の変数には、代入可能変数に加えて、ifコマンドでも利用可能なspdやaccといった変数が使用できます。

```
let,share2=share2+acc
```

8.10. monf

サウンドフローのifコマンドで用いられるfnc変数が監視するファンクション番号を変更できます。たとえば、サウンドフローごとにファンクション機能を作るのに用います。デフォルトはファンクション0(F0)です。

コマンド	引数1	引数2	引数3
monf	監視するファンクション番号(0-32)		

ファンクション8(F8)を、このサウンドフローで監視するように設定する例。fnc変数は、自動的にF8を監視するようになります。

```
【記入例】
monf,8
```

8.11. play

サウンドを再生するコマンドです。内部にwait相当の機能が自動設定されており、playコマンドの次の行には、再生終了直前まで遷移しません。

コマンド	引数1	引数2	引数3
play	WAVファイル名	ループ回数(0 無限ループ,1-100 ループ回数)	即再生(1 or 0)

ループ再生させる場合は、ループON/OFFの引数を0にします。 1 回だけ再生したい場合には1を入れます。2を入れると2回再生します。最大で100回まで指定できます。 playコマンドで、すぐにWAVファイルを再生したい場合は、即再生の引数に1を入れます。このサウンドフロー内で既に再生させているサウンドがある場合、停止してから再生することでよければ、即再生の引数に0を入れます。

他のサウンドがループ再生中に、このコマンドが呼ばれた場合、WAVファイルの再生終了の瞬間で、このWAVファイルに切り替わります。

```
【記入例】
play,seibuaw_in.wav,1,0
echo,playwav
play,seibuaw_loop.wav,0,0
```

8.12. plyx

サウンドを再生するコマンドです。playコマンドのような再生待ちウェイト機能が無く、次の行に再生直後に遷移します。 再生しながら複雑な停止制御や、動き方を実現したい場合にご利用ください。ウェイト以外はplayコマンドと機能は変わりません。

コマンド	引数1	引数2	引数3
plyx	WAVファイル名	ループ回数(0 無限ループ,1-100 ループ回数)	即再生(1 or 0)

ループ再生させる場合は、ループON/OFFの引数を0にします。 1回だけ再生したい場合には1を入れます。2を入れると2回再生します。最大で100回まで指定できます。 playコマンドで、すぐにWAVファイルを再生したい場合は、即再生の引数に1を入れます。このサウンドフロー内で既に再生させているサウンドがある場合、停止してから再生することでよければ、即再生の引数に0を入れます。

他のサウンドがループ再生中に、このコマンドが呼ばれた場合、WAVファイルの再生終了の瞬間で、このWAVファイルに切り替わります。

```
【記入例】
plyx,seibuaw_in.wav,1,0
```

8.13. pit

サウンドを再生ピッチ(再生速度)を変更するコマンドです。 惰行音など、速度に応じたモータや機械軸の音を調整するために使用します。

コマンド	引数1	引数2	引数3
pit	ピッチ(直値)	-	-
pit	ピッチ下限値	ピッチ上限値	-

引数1だけを与えた場合には、速度に関係なく、その再生ピッチに変更します。 引数1と引数2に、ピッチの値を入れた場合、速度に応じてピッチが自動で切り替わります。ピッチ下限値は、必ずピッチ上限値よりも小さい値でなければなりません。

ピッチは、1024を与えると標準の再生速度になります。最低値は512(1/2の再生スピード)、最大値は2048(2倍の再生スピード)となります。 ピッチ上下限値を設定した場合、停止状態のときに下限値の再生スピード、最高速度の時に上限値の最高スピード、５０%速度の時に上下限値の半分の再生スピードで、該当するサウンドフローのスロットの音が再生されます。

サウンドファイルごとに再生ピッチは調整できませんのでご注意ください。

pitコマンドを実行すると、すぐに反映されます。速度に関係なくピッチ変更した場合、速度に応じた自動切換えは即座にOFFされます。

```
【記入例】
pit,512,1024
play,dakou_loop.wav,1,0
```

【記入例】
```
pit,684
play,beep.wav,0,0
```

8.14. ret

サウンドフローの最初に戻るコマンドです。記入されている位置に関係なく、サウンドフローの最初に戻ります。戻った後、そのまま実行は継続されます。

サウンドフローを停止・終了したい場合は、exitコマンドを使用してください。

コマンド	引数1	引数2	引数3
ret	-	-	-

どの場所に置いても、サウンドフローの最初に戻れます。

【記入例】
```
〜〜〜
ret
〜〜〜
```

8.15. set

ユーザーが変更可能な変数に値を格納するコマンドです。

コマンド	引数1	引数2	引数3
set	変数名	設定する値	

変数に設定可能な値は変数によって異なります。

変数の種類	変数の説明	使用可能範囲	値の範囲
local	サウンドフロー内で使用可能なユーザー変数	local1〜local4	0-255
share	その他のサウンドフローと共有して使用可能なユーザー変数	share1〜share4	0-255
tmr	値を設定すると、1秒ごとに自動的に値が減っていくタイマ変数	tmr1〜tmr4	0-255
cv	CVの設定値	cv1-cv1024	0-255

【記入例】
```
play,seibuaw_in.wav,0,0
```

```
echo,playwav
play,seibuaw_loop.wav,1,0
```

8.16. sply

速度に連動してサウンドを再生するコマンドです。再生するWAVファイルの長さを100%として、速度に応じて再生開始位置が自動で変更されます。なお、加速が終わるとサウンドは自動で停止します。

コマンド	引数1	引数2	引数3
sply	WAVファイル名	-	-

オープンサウンドデータに使用されているVVVFサウンドは、LokSoundの仕様上、加速や減速音を６分割や８分割して、速度に合わせるように実装しています。 しかし、分割する事で、加速が止まっているのにもかかわらず音が出続けてしまう事や、ちょっとした加速を再現することが難しい現状があります。

本コマンドを使用することで、速度や加減速状態に応じて、無段階に分割したかのようにWAVファイルを再生することができます。

本コマンドを使用する場合、速度と連動する関係上、WAVファイルの長さ（再生時間）と加速時間(CV2)や減速時間(CV3)を同じ時間に設定する必要があります。ズレている場合、速度とうまく連動しません。

以下の記入例は、加速サウンドフローと減速サウンドフローを表したものです。両方は、モータ駆動直前にロードするように実装するか、サウンドONと紐づいてcallを行うとよいでしょう。

```
【記入例（加速）】
label,START
if,spd<1,START
if,acc<=1,START
sply,E233_kasoku.wav,
goto,START
```

```
【記入例（減速）】
label,START
if,spd<2,START
if,acc>=-1,START
sply,E233_gensoku.wav,
goto,START
```

8.17. stop

playコマンドで再生したサウンドを停止するコマンドです。

コマンド	引数1	引数2	引数3
stop			

ループ中等も問わず、サウンドは停止します。なお、同時に実行中の他のサウンドフローのサウンドは停止しません。

```
【記入例】
stop
```

8.18. slim

モータの速度制限を行うコマンドです。

コマンド	引数1	引数2	引数3
slim	制限速度(0-255)		

停止させたい場合は、制限速度を0にしてください。制限なし（最高速度まで可）とする場合は、制限速度を255と指定してください。

```
【記入例】
slim,255
```

8.19. vol

サウンドのボリュームを調整するコマンドです。

コマンド	引数1	引数2	引数3
vol	音量(0-255)		

無音にする場合は、0にしてください。音量を標準とする場合は、255と指定してください。50%音量は127です。

```
【記入例】
vol,255
```

8.20. volm

サウンドのマスターボリューム（デコーダ全体のサウンド音量）を調整するコマンドです。 サウンドフローのどのファイルからコマンドを実行しても、共通のマスターボリュームが調整されます。

コマンド	引数1	引数2	引数3
volm	音量(0-255)		

無音にする場合は、0にしてください。音量を標準とする場合は、255と指定してください。50%音量は127です。

```
【記入例】
volm,127
```

8.21. wait

ms(ミリ秒)待ちます。並列して実行されているサウンドフローには影響を与えません。

コマンド	引数1	引数2	引数3
wait	待ち時間(ミリ秒)		

5000ミリ秒待つ例を以下に示します。

```
【記入例】
wait,5000
```

8.22. wrnd

ms(ミリ秒)、ランダムに待ちます。並列して実行されているサウンドフローには影響を与えません。

コマンド	引数1	引数2	引数3
wait	ランダム待ち時間上限値(ミリ秒)		
wait	ランダム待ち時間下限値(ミリ秒)	ランダム待ち時間上限値(ミリ秒)	

0-5000msの間でランダムに待つ例と、1000-2000msの間でランダムに待つ例を以下に示します。

```
【記入例(0-5000msの間でランダムにウェイト)】
wrnd,5000
【記入例(1000-2000msの間でランダムにウェイト)】
wrnd,1000,2000
```

8.23. wspd

速度に連動して、ウェイト時間が設定できます。 y=ax+bの一次方程式で表すと、aが速度連動ウェイト時間、xが速度、bがオフセット時間となります。

反比例モード(0)にすると、速度が上がるたびに、ウェイトが短くなります。最高速度の時、ウェイト時間だけ待ちます。蒸気機関車などでは、反比例モードで使用してください。 比例モード(1)にすると、速度が上がるたびに、ウェイトが長くなります。停止時は、オフセット時間だけ待ちます。

コマンド	引数1	引数2	引数3
wspd	速度連動ウェイト時間	反比例(0) or 比例(1)	オフセット時間

速度連動ウェイト時間, オフセット時間にはミリ秒(ms)を設定してください。 オフセット時間は、省略した場合、0msと設定されます。

```
【記入例(0.5秒は最高速度でもウェイトされます)】
wspd,2000,0,500

【記入例(走行中に音を出す)】
label,STOP
if,spd==0,STOP
label,RUNNING
play,pop.wav,1,0
wspd,2000,0,500
if,spd>0,RUNNING,
stop
ret
```

8.24. wtrg

（廃止予定。plyxコマンドやxifコマンドを使用してください。） 直後に使用されるウェイトコマンドを、指定条件が成立したら強制解除します。 playコマンド, waitコマンド, wrndコマンドが影響します。

コマンド	引数1	引数2	引数3
wtrg	条件式		

条件式には、ifコマンドと同じ式が使用できます。

```
【記入例(TEST.wav再生中に、モータが減速したら再生中ウェイトは解除)】
wtrg,acc<0
play,TEST.wav,1,0
```

8.25. xif

条件が整うと、指定のラベルにジャンプします。 偽の時の行き先ラベルを省略すると、if文に続いて次のコマンドを実行します。複数のifコマンドを用いて状態遷移を作りたいときに使用してください。

コマンド	引数1	引数2	引数3
if	条件	真の時の行き先ラベル	偽の時の行き先ラベル(省略可)

条件には、様々な変数が使用できます。

変数の種類	変数の説明	使用可能範囲	値の範囲
local	フロー内変数		
share	共有変数	デコーダ全体のサウンドフローで使用できる変数	
spd	デコーダの現在速度		
ref	指令速度		
fnc	ファンクション		
aux	AUX状態		
tmr	タイマ		
acc	加減速度		
dir	進行方向		
cv	CVの設定値		

ファンクション操作によってジャンプする例。ここでは、監視ファンクションがONになった時の例です。

```
【記入例】
label,START
if,fnc==1,SOUND_ON,START
〜〜〜
label,SOUND_ON
〜〜〜
label,SOUNDLOOP
if,fnc==0,SOUND_OFF,SOUNDLOOP
〜〜〜
label,SOUND_OFF
```

9. CV一覧

CV番号	機能説明	初期値
CV1	ショートアドレス	3
CV2	始動電圧	8
CV3	加速時間(0.6で割った時間が秒)	120
CV4	減速時間(0.6で割った時間が秒)	90
CV5	最大電圧	200
CV6	中間電圧	70
CV9	PWMキャリア	0 (32kHz)
CV10	BEMF カットアウト係数	2
CV17	Long Address LSB	0
CV18	Long Address MSB	0
CV28	RailCom有効設定	1
CV29	デコーダ設定	10
CV54	BEMF係数	96
CV55	PI制御器 Pゲイン	16
CV56	PI制御器 Iゲイン	32
CV62	集電不良対策・自動サウンドOFF	0
CV63	マスターボリューム	100
CV64	キックスタート切替速度	50
CV65	キックスタート電圧	0
CV67-94	スピードカーブ	
CV155-170	SoundCV設定	
CV185	ヘッドライト・AUX出力設定	0
CV186	テールライト・AUX出力設定	0
CV187	AUX1・AUX出力設定	0
CV188	AUX2・AUX出力設定	0
CV189	AUX3・AUX出力設定	0
CV190	AUX4・AUX出力設定	0

CV番号	機能説明	初期値
CV191	AUX5・AUX出力設定	0
CV192	AUX6・AUX出力設定	0
CV193	AUX7・AUX出力設定	0
CV194	AUX8・AUX出力設定	0
CV195	ヘッドライト・AUX追加設定	0
CV197	テールライト・AUX追加設定	0
CV198	AUX1・AUX追加設定	0
CV199	AUX2・AUX追加設定	0
CV200	AUX3・AUX追加設定	0
CV201	AUX4・AUX追加設定	0
CV202	AUX5・AUX追加設定	0
CV203	AUX6・AUX追加設定	0
CV204	AUX7・AUX追加設定	0
CV205	AUX8・AUX追加設定	0

9.1. CV54

モータのBEMF特性を使って、坂道等でも速度が変わらないように制御することができます。この制御のために、BEMF特性に基づいて検出した速度とコマンドステーションからの指令速度の比率・差異を調整・吸収するパラメータになります。

NゲージやHOの車両に使用されるモータ特性から、64～128の間の値を通常は設定します。デフォルトでは96を設定しています。低い値を設定すると、最高速度が低めになる傾向があります。

9.2. CV185～CV194 AUX設定

AUXでライト機能を使用する場合に使用します。

10. FAQ

10.1. Serialモニタが反応しない

内部のソフトがフリーズすると、シリアルモニタが反応しなくなります。ソフトのバグと思ってよいです。

nullアクセス、変数のオーバーフロー、ポインタの不正アクセス、ゼロ割等、数多くの問題があるはずです。

10.2. うまくスケッチやFSがアップロードできない

BOOTSELボタンを押しながらUSBケーブルをPCに差し込んでください。自動シリアル経由リセットでは、うまくリセットが動かない場合があります。特にフリーズさせたとき。

10.3. サウンドをループさせると、変な音が入る

LISTチャンクの除去機能によって通常は、不要なデータは削除されますが、DSSPが対応していない付加情報が含まれていると、うまく再生できない場合があります。また、ID3タグがあっても同様にノイズが載ってしまう場合があります。

10.4. サウンドを再生すると、ノイズが出る

8bit,16bitの48kHz,44.1kHz,32kHz, 21.05kHz,16kHz, 10.25kHz, 8kHzのリニアPCM(LPCM)フォーマットのRIFF WAVEファイルのみに対応しています。float32や圧縮がかかったWAVEファイルは対応していません。

10.5. ファイル名が長いと認識されない

LittleFSの仕様上、32文字までにしなければなりません。できる限り短くしてください。

10.6. USBライターが認識されない

ファームウェアまたはサウンドデータを２回目以降、書き込む時にはボタンを押し続けてUSBケーブルをPCに繋ぐ必要があります。

また、USBライターとデコーダの間の信号接続用のポゴピンの接触が悪くても認識しない場合があります。熱収縮チューブがついていると、書き込みできませんのでご注意ください。

10.7. 走行がギクシャクする

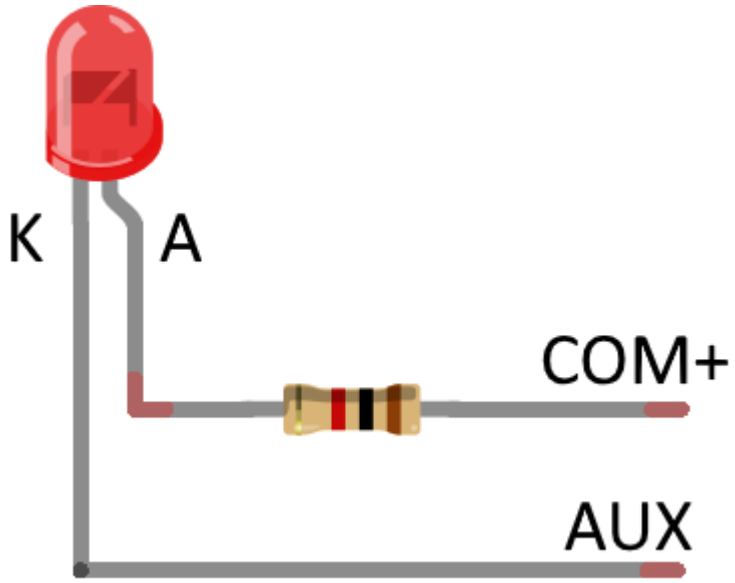
SmileSoundデコーダは、消費電力が多いため、コンデンサの効力が他のデコーダと比べて低くなる傾向があります。車両搭載時には、必ずコンデンサの搭載をしてください。

また、ショートの発生がしていないかの確認もしてください。コマンドステーションの安全機能が働き、線路への供給が停止されることもあります。

コマンドステーション側の問題であることもあります。たとえばPC接続型コマンドステーションであるLokProgrammerのUSBケーブルの接触不良や、ACアダプタの寿命・性能不足等が過去に報告されています。デコーダの異常と勘違いしやすいので、コマンドステーション側の保護機能や動作の確認も合わせて行うようにしてください。

10.8. LEDを直接、AUXやヘッドライト・テールライトに配線して良いですか？

LEDは、必ず抵抗を直列に挿入して使用してください。抵抗を挿入しない場合、ショートが発生する場合があります。コマンドステーションの安全機能が働き、線路への供給が停止されることもあります。



10.9. ExpBoard M21 Groundを使ってDCジャックによる電源供給をするとき、音が鳴らない

CV62に0以外の値が入っていると、集電不良時の対策機能が作動し、サウンド再生を強制的に止める動きをします。DCジャックで供給して使用する場合には、必ずCV62=0を設定してください。

11. ライセンス