Aeropoint GUI

ユーザーズマニュアル

株式会社　ＣＲＩ・ミドルウェア

発行日：2021/03/15

Rev：1.06

この度は Aeropoint GUI をご利用いただき、誠にありがとうございます。このマニュアルには本キットを利用するにあたっての詳細な設定方法やシステムの解説、API の一覧やサンプルプログラムの説明などを掲載しています。

また、お客様が本製品を安全に取り扱いいただくための注意事項を掲載しています。ご利用にあたっては、必ずこのマニュアルをお読みください。

■ 利用条件

- 本キットは、お客様による Aeropoint GUI の評価を目的として配布しています。評価目的以外の利用はできません。お客様の製品の開発・製造にあたりましては、別途弊社との個別契約を締結いただくようお願い申し上げます。

■ ご注意

- 本書の内容の一部、または全部を無断で転載することは禁止します。
- 本書の内容については、将来予告なしに変更することがあります。
- 本書で説明する内容につきましては万全を期して制作しましたが、運用した結果の影響につきましては、責任を負いかねますのでご承知ください。

■ 商標及び登録商標について

- 「CRI」「Aeropoint」CRIWARE ロゴは、日本及びその他の国における株式会社ＣＲＩ・ミドルウェアの登録商標または商標です。
- 「OPTPiX」は株式会社ウェブテクノロジの登録商標です。
- 「RX」「RZ/A」「R-Car」はルネサスエレクトロニクス株式会社の日本、米国およびその他の国における登録商標または商標です。
- 「Microsoft Windows®」及び、「Microsoft Word®」「Microsoft PowerPoint®」は、米国マイクロソフト社の米国および他の国における登録商標です。
- 「Adobe Photoshop」及び、「Adobe PDF」はアドビシステムズ社の米国および他の国における登録商標または商標です。
- その他、本書中の社名や商品名は、各社の登録商標または商標です。

■ ルネサスアライアンスパートナー

- 株式会社ＣＲＩ・ミドルウェアはルネサスエレクトロニクス株式会社のアライアンスパートナーです。

目次

1 はじめに 5
1.1 特長 5
1.1.1 特長１「PowerPoint®で GUI を作る」 5
1.1.2 特長２「サウンド・動画再生」 6
1.1.3 特長３「高画質と安定性を両立」 7
1.1.4 特長４「同時多言語対応」 8
1.2 Aeropoint を利用した開発 9
1.2.1 シンプルに、早く 9
1.2.2 独立したモジュールとして動作 10
1.2.3 ランタイム SDK 11
1.2.4 製品開発へのご協力 12
2 ランタイムキットの紹介 13
2.1 リファレンスキット 13
2.2 FIT キット 13
3 リファレンスキット 14
3.1 必要なもの 14
3.2 セットアップ 15
3.2.1 Aeropoint リファレンスキットの展開 15
3.2.2 Aeropoint ツールキット 16
3.2.3 実機 16
3.2.4 電源との接続 17
3.2.5 microSD カード 17
3.2.6 Aeropoint プレイヤー 18
3.2.7 LAN 接続 19
3.2.8 D-Amp 基板の接続 20
3.2.9 配置例 21
3.3 リファレンスキットの動作確認 22
3.3.1 Welcome スライドの表示 22
3.3.2 サウンド再生 26
3.3.3 キャプチャ 26
3.4 導入済みサンプル 27
3.4.1 会議室デモ 27
3.4.2 トイレリモコン、血圧・脈拍計、ホームコントロール 29
3.4.3 サウンドテスト 30
3.4.4 ビデオテスト 31
3.4.5 I/O テスト 31
4 FIT キット 32
4.1 対応マイコン 32
4.2 e²studio と QE for Display の準備 32
4.3 FIT キットを用いたプロジェクトのセットアップ 33
4.3.1 プロジェクトを新規作成する 33
4.3.2 QE for Display セットアップ 36
4.4 GUI の作成 44
4.5 ライブラリ 45
4.6 プログラムの作成と実行 46
5 Aeropoint ツールキットの機能 48
5.1 ターゲット 48
5.1.1 全て書込み 48
5.1.2 書込み 49
5.1.3 再生 49
5.2 リソース 50
5.2.1 文字 50
5.2.2 動画 51
5.2.3 音 52
5.3 スライド 53
5.3.1 名前 53
5.4 シェイプ 53
5.4.1 名前 53

5.4.2 ID 53
5.4.3 ボタン ID 54
5.4.4 X、Y、幅、高さ、縦横比を固定 54
5.4.5 非表示 54
5.4.6 QR 54
5.4.7 ボタン 55
5.4.8 リセット 55
5.4.9 コメント 55
5.5 エディット 56
5.5.1 図形の接合 56
5.5.2 フリーフォーム 57
5.5.3 オブジェクトスタイルのコピー＆適用 57
5.5.4 最前面・最背面へ移動 58
5.5.5 グループ化・解除 58
5.5.6 配置 59
5.5.7 回転 59
5.5.8 オブジェクトの選択と表示 60
5.5.9 オブジェクトの書式設定 61
5.6 オプション 機能 62
5.6.1 インポート 62
5.6.2 エクスポート 63
5.6.3 フォント作成 64
5.6.4 解像度 64
5.6.5 設定 64
6 作ってみる 65
6.1 準備 65
6.1.1 新規作成 65
6.1.2 解像度変更 66
6.2 どんな GUI にするか考える 67
6.2.1 ふんわりとした全体像 67
6.2.2 なんとなく形にする 68
6.3 見た目を整える 73
6.3.1 メリハリをつける 73
6.3.2 代表的な部品を作る 74
6.3.3 コピペしながら増やす 79
6.4 動きを付ける 83
6.4.1 ボタン 83
6.4.2 画面切り替え 84
6.4.3 数字 86
6.5 試作完了、製品へ 88
6.5.1 たくさん触って微調整 88
6.5.2 人に見てもらう 88
6.5.3 次のステップへ 88
7 ランタイム SDK 89
7.1 アプリケーション開発環境 89
7.1.1 Renesas Flash Programmer 89
7.1.2 Aeropoint ツールキット 90
7.1.3 RX72N Envision Kit 90
7.1.4 e²studio 91
7.1.5 e²studio ツールチェーン 92
7.2 システム構成 93
7.3 基本フロー 94
7.4 Aeropoint ランタイム 95
7.4.1 最小のアプリケーション 96
7.4.2 スライドの表示 102
7.4.3 コールバック 104
7.5 Aeropoint プログラミングリファレンス 107
7.5.1 シンボル、構造体、列挙体 107
7.5.2 UART コマンド 108
7.5.3 プレイヤーAPI 109
7.5.4 スライド切替え API 116
7.5.5 動画 API 117
7.5.6 サウンド API 119

7.5.7 文字表示 API 121
7.5.8 表示切替え API....124
7.5.9 テスト IOAPI 125
7.5.10 デバッグテキスト API 128
7.5.11 プリミティブ表示 API 134
7.5.12 データフラッシュ API 137
7.5.13 シリアル通信 API 139
7.6 サンプルプログラム 151
7.6.1 スライドを切替える....151
7.6.2 画面に文字を表示する 152
7.6.3 サウンドを再生する....153
7.6.4 動画を再生する....154
7.6.5 表示を切り替える 155
7.6.6 テスト IO を制御する....156
7.6.7 画面にデバッグテキストを表示する....157
7.6.8 画面にプリミティブな図形を表示する 159
7.6.9 シリアル通信を行う 160
8 シリアル通信の接続方法 163
9 フォント作成....164
9.1 起動....164
9.2 使い方 165
10 プログラムの書き込み 166
10.1 Renesas Flash Programmer....166
10.2 書き込み手順....166
11 データイメージの書き込み 169
11.1 用意するもの 169
11.2 書き込み手順....169
12 Q&A 171
12.1 Aeropoint ツールキットのアドインが表示されない 171
12.2 スライドのテーマやマスタースライドを利用すると、実機で正しく部品が配置されない 173
13 参考価格 174
14 お問い合わせ 174
14.1 CRI・ミドルウェア....174
15 改訂記録 175

1　はじめに

ようこそ、Aeropoint GUI へ！
Aeropoint GUI は、普及価格帯のマイコンで豊かな映像表現を実現する、グラフィカル UI ミドルウェアです。

1.1　特長

Aeropoint の 4 つの特長を説明します。

1.1.1 特長 1 「PowerPoint®で GUI を作る」

組込みマイコンの分野でも、カラーLCD とタッチ操作による GUI（グラフィカル・ユーザー・インターフェース）を搭載できる機器が増えてきました。スマートフォンに慣れ親しんだユーザーから画面の美しさや快適な操作感が求められる中、これまでの手法でそういった「見た目」や「操作感」を追及すると、開発者に高い負荷を掛けたり、開発の工数が増大したりするなど、課題も少なくありませんでした。

そこで Aeropoint は、誰もが使える PowerPoint®で GUI を制作することで開発者の負担を抑えつつ、PowerPoint®に独自のアドインを追加することで「見た目」にもこだわれる開発環境を提供します。

図 1　Aeropoint アドイン

また実機では、ゲーム業界のノウハウがふんだんに活かされた「Aeropoint ランタイムエンジン」が、GUI に必要となるほぼ全ての表示や制御を担当し、快適な「操作感」を実現します。

図 2　PowerPoint®で描いた、実際に使える GUI

1.1.2 特長２「サウンド・動画再生」

弊社開発の「D-Amp Driver」を用いて、マイコンの ON/OFF 信号のみで高品質なサウンド再生を実現します。

【サウンド再生】

- 回路のデジタル化により、アナログ部品が不要となるため、部品コストを大幅に削減できます。
- 専用 LSI やアナログ部品を使用しないため、ディスコン(製造中止)に伴う設計変更等のリスクがなく、長期利用、長期供給保証が可能になります。
- マイコン以外に必要なハードウェアはトランジスタと抵抗だけになるため、過酷な動作環境で使用する機器にも適しています。
- モノラル 24kHz のサウンドを 2 音同時に再生できます。

図 3　D-Amp Driver

ゲーム業界で培われたストリーミング技術を活用して、普及価格帯のマイコンで初めての本格的な動画再生を実現しています。GUI コンテンツの説明に用いたり、動画広告を取り入れたりするなどの用途が広がります。

【動画再生】

- RX65N/RX72N で唯一(2020/06/25 現在)、動画再生ができます。サウンドも同時に鳴らせます。
- 15fps、30 秒の動画を 8 本まで。外付けメモリーのサイズに応じて拡張可能です。
- 投稿動画や CM 動画などからもエンコードできます。

図 4　実機で再生する動画の様子

1.1.3 特長３「高画質と安定性を両立」

普及価格帯のマイコンで高画質な画面を表示しようとする場合、メモリー容量が不足してしまうことで画面の書き換にチラつきが発生するなど、安定性を欠くことがあります。

Aeropoint は省メモリーと 2 画面表示の仕組みを用いて、高画質と安定性の両立を実現します。

まずは不足しがちなメモリー容量を節約するために、優れた減色技術で定評のある「OPTPiX」を内蔵して画像を省メモリーに変換します。例えばフルカラーの画像を 256 色に変換すれば、容量は 1/4 になります。単純に減色すると実用に耐えないような画像でも、OPTPiX を使えばフルカラーと見違えるほどの画質を維持できます。

図 5　減色方法による画質の違い

画面がチラついて安定しない原因は、表示中の画面を更新するからです。そこで、まずは節約したメモリー容量を活用して、内部的に画面を 2 つ用意します。次に「表示していない方の画面」に画像を書き込み、液晶表示装置が描画を休止する一瞬を狙って画面を切替え、安定した更新を実現します。

図 6　2 画面を用いた画面更新

動画の再生が綺麗なのも、この 2 画面方式を採用できるからです。

1.1.4 特長４「同時多言語対応」

Aeropoint では、最大８種類のフォントを同時に利用できます。これにより、地域ごとに言語を入れ替えて製品を作り直す必要が少なくなります。

図 7　同時多言語対応

PC に登録されているフォントから、実機用フォントを作成するツールも用意しています。製品や画面のイメージに合ったフォントを自由に組込めます。

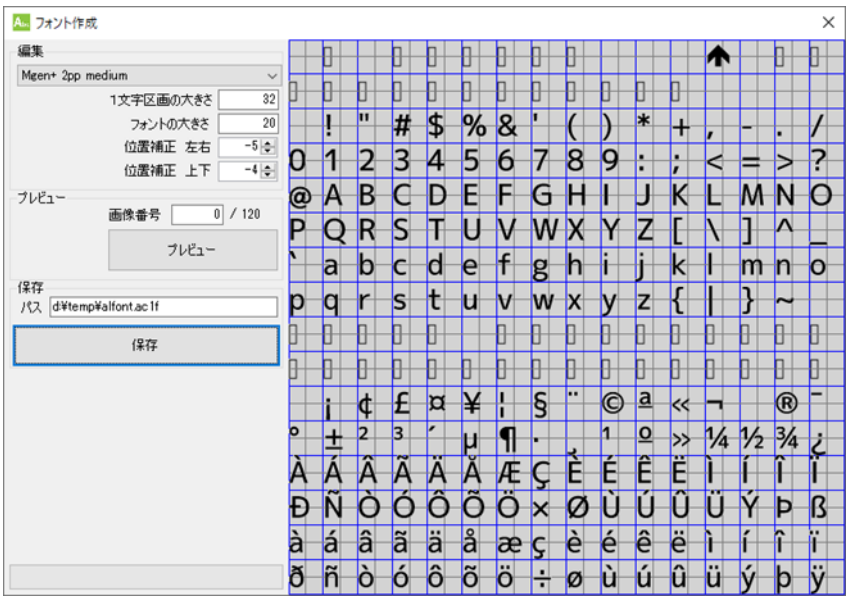


図 8　フォント作成ツール

フォントの利用権につきましては、お客様がご用意ください。PC に標準で登録されているすべてのフォントは、そのフォントを所有する会社などが権利を有しており、それらの会社からの許可なしには再利用できません。

1.2　Aeropoint を利用した開発

1.2.1 シンプルに、早く

Aeropoint を用いた開発は、とてもシンプルです。PowerPoint®で自由にデザインしたら、Aeropoint アドインで GUI の振る舞いを設定します。後は「書込み」クリックすると実機のサンプルボードに転送が行われ、再生準備完了です。

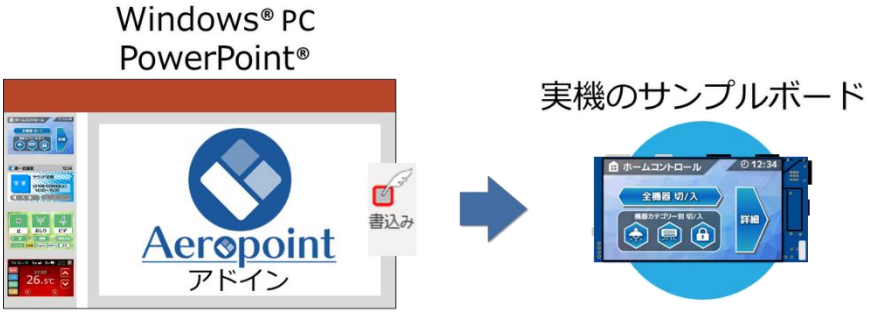


図 9　シンプルな構成

ひとつの画面を実機で表示するまでに必要な時間は数秒程度で、何度でもやり直しができます。プログラムを書く必要もなく、デザイン専用ツールを使った素材の制作も不要であるため、GUI の作り込みにより多くの時間を使えるようになります。

また PowerPoint®ファイルには GUI のデザインはもちろん、サウンドや動画などの開発に必要なすべての情報が格納されているため、担当者間の情報共有はファイルひとつで完結します。

ローカライズ（多言語対応）やバリエーションが必要な場合も、PowerPoint®ファイルでデザイン変更を行うだけです。翻訳担当者がいる場合なら、PowerPoint®ファイルを添付したメールのやり取り一回だけで素早く再展開ができてしまう、という事もありえます。

図 10　素早い再展開

他方、お客様の製品がサンプルボードと同一のマイコンを利用していたとしても、周辺の機器やアーキテクチャの違いによりカスタマイズが必要となります。製品化にあたりましては弊社にてポーティングの作業を実施させていただきますので、「14 お問い合わせ」までお気軽にご連絡ください。

1.2.2 独立したモジュールとして動作

完成したGUIは独立したモジュールとして動作するため、製品設計を大きく見直すことなく外付けできます。GUIを制御するためには、既存のMPUなどからシリアル通信を用いたコマンドをやり取りします。

図 11　コマンドを用いた計測と制御

以下に、コマンドの例を示します。

制御	結果など
スライド表示切替え	ボタンタッチ
シェイプ表示切替え	ログ
テキスト表示	スクリーンキャプチャー
サウンド再生	各種ステータス
動画再生	任意データ

図 12　コマンドの例

本キットではシリアル通信の代わりに、PCで動作する「コネクター」ツールを有線LAN接続して実機と通信します。

図 13　PCコネクター

1.2.3 ランタイム SDK

ランタイム SDK を用いると、C 言語によるカスタマイズが行えます。Aeropoint の主プログラムは非常に簡潔にまとめられているため、導入はとても簡単です。例えば以下の main.c で、Aeropoint のすべての機能が動作します。

```
1  void main(void)
2  {
3      CriBool flag = CRI_TRUE;
4
5      /* Initialize */
6      CriAeroPlayer_Initialize();
7
8      /* Main loop */
9      while( flag )
10     {
11         /* Main */
12         flag = CriAeroPlayer_Main();
13     }
14
15     /* Finalize */
16     CriAeroPlayer_Finalize();
17
18     return;
19 }
```

図 14　最小のプログラム

他方、GUI を利用するためには解像度に応じたメモリー容量が必要となり、また CPU や周辺機器のリソースも利用します。

ランタイム SDK を用いたプログラミングには、ルネサス社の統合環境「e²studio」を利用します。

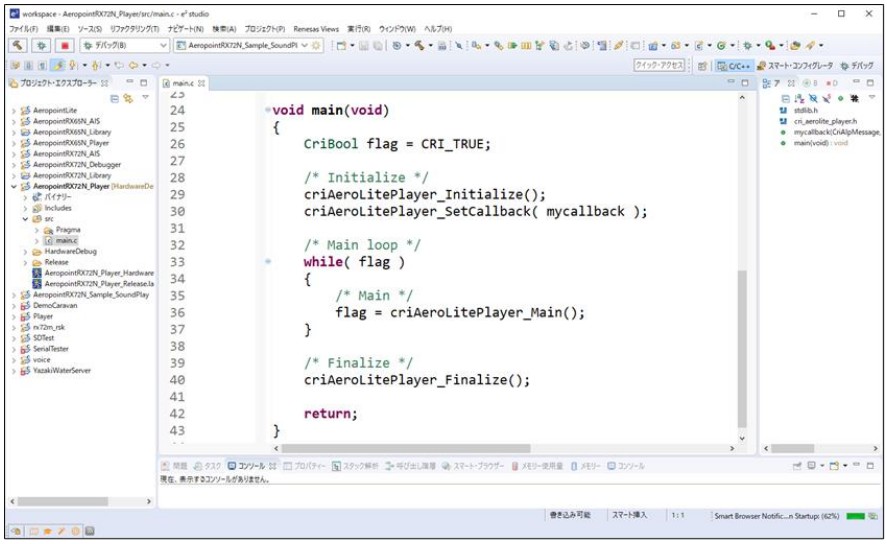


図 15　e²studio

1.2.4 製品開発へのご協力

本キットを用いた検証が好評であれば、製品開発もご検討ください。先ずは、試作やデモが必要という場合にも、ぜひ弊社にご相談ください。

製品開発にあたっては、お客様の HW に合わせた Aeropoint ライブラリを提供いたします。そのうえで、お客様単独で製品を作り切られる場合は「許諾プラン」を、弊社の GUI 開発ノウハウを生かした企画・デザイン設計、音声・サウンド収録、ソフトウェア開発やその一部をご希望される場合は「業務委託プラン」を検討ください。

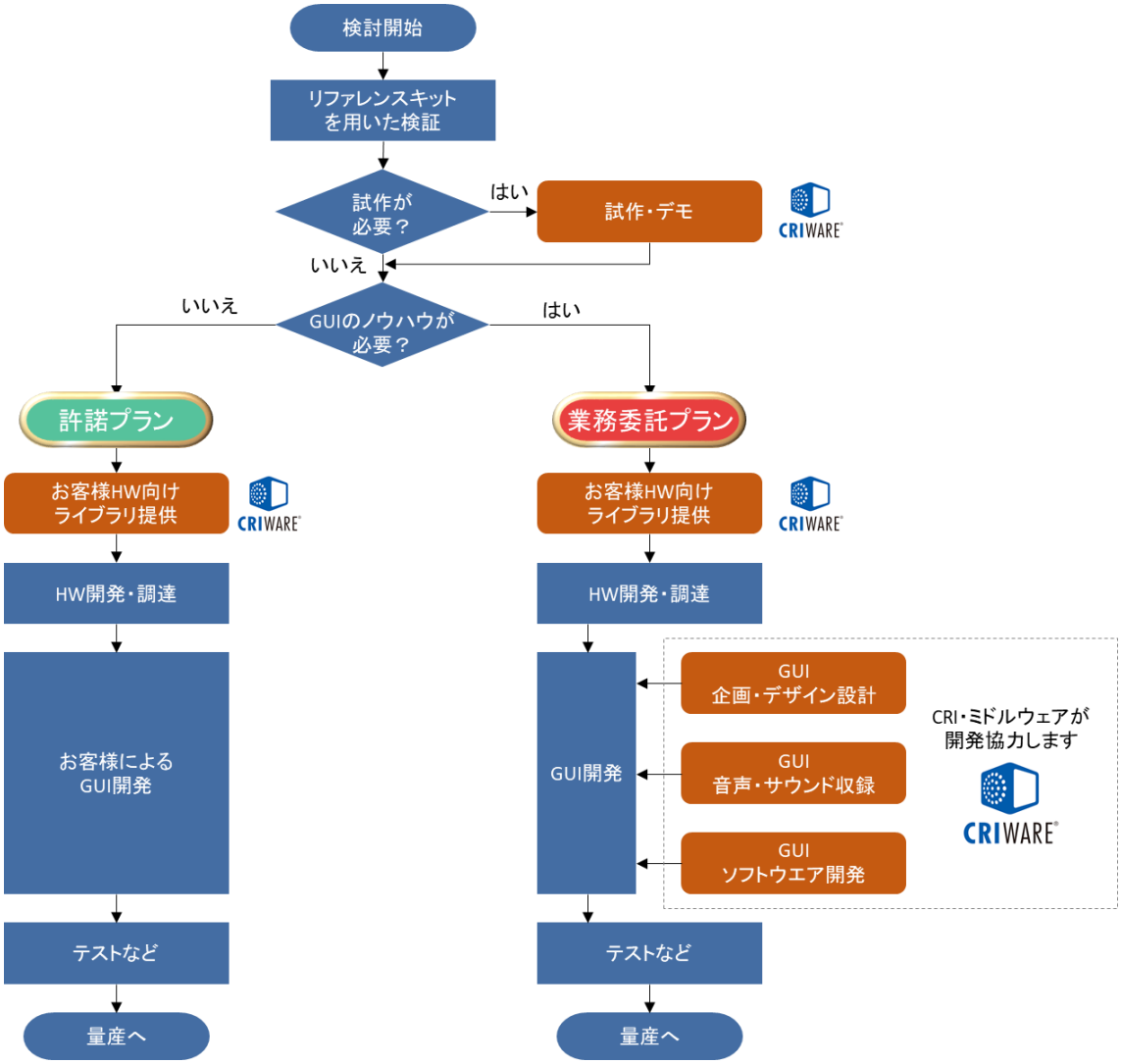


図 16　製品開発へのご協力

製品開発の取組みには様々な形があると思います。都度柔軟に対応させていただきますので、上記例に関わらずご相談頂ければ幸いです。

2　ランタイムキットの紹介

Aeropoint のランタイムキットは、お客様の目的に合わせて「リファレンスキット」と「FIT キット」の２種類を用意しています。

2.1　リファレンスキット

リファレンスキットは、いち早く Aeropoint を試したい方に最適です。ルネサスエレクトロニクス社製の RX72N Envision Kit を用い、すべての機能を体験できます。たくさんの UI サンプルはもちろん、PowerPoint® で作成した UI の表示や、プログラミングによる制御などができます。

詳しくは本書の「3.リファレンスキット」をご覧ください。

図 17 RX72N Envision Kit

2.2　FIT キット

FIT キットは、リファレンスキットを用いた試用を終え、いよいよ製品開発や量産に進まれたい方向けの仕組みです。ユーザー基板での動作を実現するため、Aeropoint の全機能を FIT モジュールとして提供しています。

ルネサスエレクトロニクス社提供の QE for Display ツールを用いれば、ウィザード形式でユーザー基板への Aeropoint 導入が可能になります。

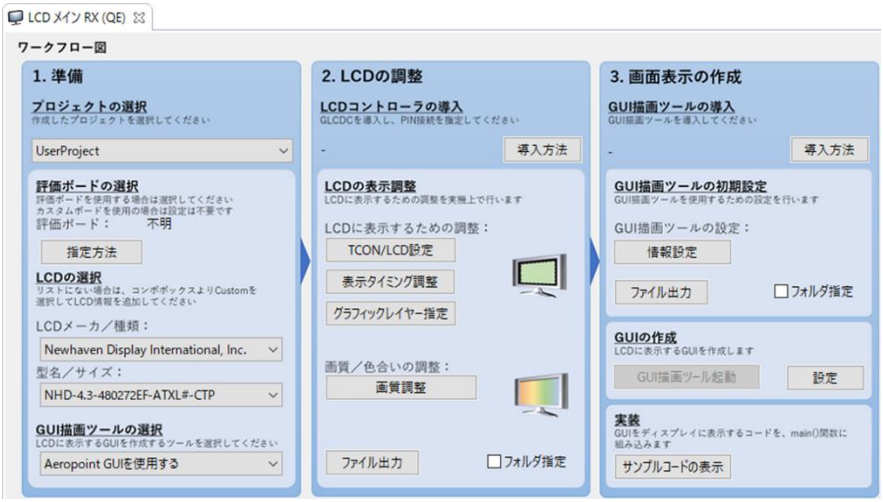


図 18　QE for Display を用いた導入の様子

3　リファレンスキット

さっそく Aeropoint を体験しましょう。

3.1　必要なもの

先ずは必要な機材やソフトウェアを確認します。

- Aeropoint リファレンスキット
  本書を含む、Aeropoint を体験いただくことを目的にした簡易評価パッケージです。以下の内容を含みます。

  - ✧ 基本ソフトウェア
    - AeropointReferenceKit_xxxxxxxx.zip にまとめられています。
      ※ xxxxxxxx はリリースの日付です。
  - ✧ 実機
    - RX72N Envision Kit
  - ✧ 周辺機器
    - 実機の電源ケーブル（USB オス←→USB Type-B オス）
      ※ PC の USB ポートなどに接続して給電します。
      ※ e²studio を用いた開発では、このケーブルを用いた PC との接続が必須です。
    - microSD カード
    - D-Amp 基板、D-Amp 基板接続ケーブル、スピーカー
      ※ サウンドの再生に利用します。

- Windows® PC と PowerPoint®
  PowerPoint®は、2013 以降のスタンドアローン版や Office365 版をご用ください。Web 版は利用できません。

- USB-LAN アダプター、LAN ケーブル
  PC と実機を接続して、データのやり取りなどに利用します。PC に LAN ポートの空きがあれば、USB-LAN アダプターは不要です。

3.2　セットアップ

機材とソフトウェアが揃ったら、セットアップを行います。

3.2.1 Aeropoint リファレンスキットの展開

以下のパッケージを展開してください。

- Aeropoint リファレンスキット
  - AeropointReferenceKit_xxxxxxxx.zip
    ※ xxxxxxxx はリリースの日付です。

次に、フォルダとファイルの構成を確認します。

```
AeropointReferenceKit_xxxxxxxx
│  AeropointUsersManual_xxxxxxxx_jp.pdf ── 本書です。
│
├──Toolkit
│      AeropointLiteToolkit_xxxxxxxxRx_CRI_64bit.exe
│      AeropointLiteToolkit_xxxxxxxxRx_CRI_32bit.exe
│                                  Aeropointのツールキットです。PCにインストールして、PowerPoint®にアドインを追加します。PCの64/32bit構成に合わせたものをご利用ください。
│
├──Runtime
│      AeropointRX72N_Library.zip ──── RX72Nで動作する、Aeropointのランタイムライブラリです。
│
│      AeropointRX72N_Player.zip ──── ユーザーアプリの基本となるプレイヤーのプロジェクトです。
│
└──Samples
    ├──Player
    │      Player.mot ──────────── Aeropointプレイヤーの実行プログラムです。
    │
    ├──SDImage
    │      SDImage_jp.bin ──────────── SDカードのデータイメージです。
    │
    ├──Project
    │      *.zip ──────────────── e2studioでビルドして利用する、サンプルプロジェクトです。e2studioにインポートして利用します。
    │
    └──GUI
           *.pptx ──────────────── GUIを収めた、PowerPoint®ファイルです。
```

図 19　リファレンスキットのフォルダとファイル構成

3.2.2 Aeropoint ツールキット

Aeropoint ツールキットを PC にインストールします。

1. PowerPoint®や他のオフィス製品が起動していれば終了します。

2. インストーラーを実行し、案内に従ってインストールを実施します。
   → 64 ビット版：AeropointLiteToolkit_xxxxxxxxRx_CRI_64bit.exe
   → 32 ビット版：AeropointLiteToolkit_xxxxxxxxRx_CRI_32bit.exe

3. インストールに成功したら、PowerPoint®を起動します。

PowerPoint®に以下のようなメニューが追加されていたら成功です。

図 20　Aeropoint アドイン

メニューが表示されない場合は「12.1 Aeropoint ツールキットのアドインが表示されない」を参考にしてみてください。

3.2.3 実機

RX72N Envision Kit を開梱し、本体を取り出します。

- ルネサスエレクトロニクス社 RX72N Envision Kit
  - https://www.renesas.com/jp/ja/products/software-tools/boards-and-kits/eval-kits/rx72n-envision-kit.html

図 21　RX72N Envision Kit

3.2.4 電源との接続

Envision Kit に給電するために、CN1 コネクターを USB ケーブルで電源に接続します。PC の USB ポートに接続しても良いでしょう。e²studio を用いた開発では、このケーブルを用いた PC との接続が必須です。

図 22　電源との接続

3.2.5 microSD カード

リファレンスキットに付属している SD カードには、Aeropoint を体験していただくために必要なデータがあらかじめ記録されています。SD カードは、実機の SD カードスロットに挿入してください。

SD カードを自分で用意をする場合には、本書の「11 データイメージの書き込み」を参考にして、リファレンスキットに付属する「SDImage_xx.img」を書き込みます。

3.2.6 Aeropoint プレイヤー

リファレンスキットに付属している実機には、Aeropoint ツールキットと連携して GUI を動作させるための Aeropoint プレイヤーがあらかじめ書き込まれています。Aeropoint プレイヤーを自動起動させるため、基板上のディップスイッチを以下のような位置に合わせておきます。

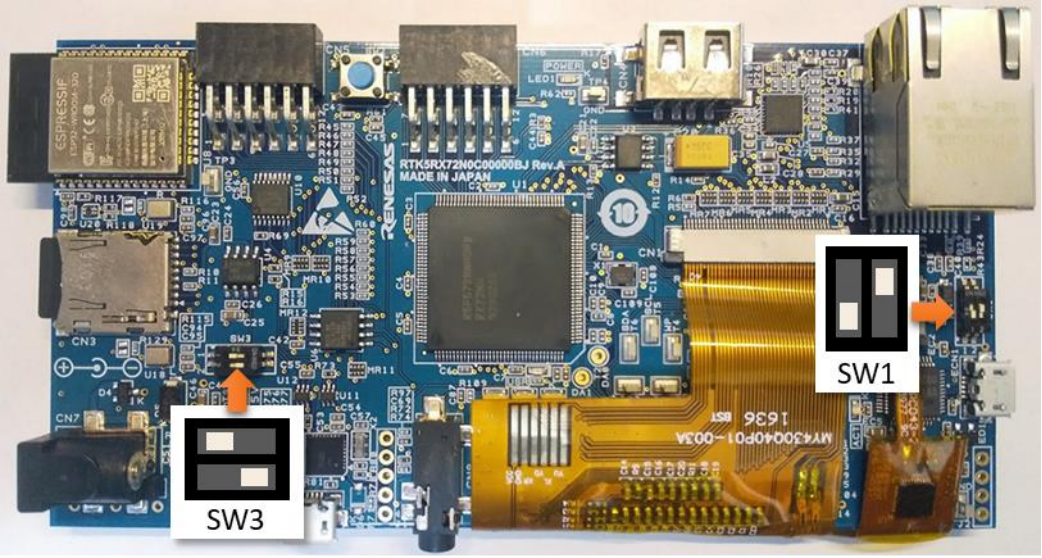


図 23　ディップスイッチの設定

サンプルプログラムなどを実行した後に、Aeropoint プレイヤーに戻すためには、本書の「10 プログラムの書き込み」を参考にして、リファレンスキットに付属する「Player.mot」を書き込みます。

3.2.7 LAN 接続

Aeropoint ツールキットと実機の間でデータのやりとりを行うために、LAN 接続を行います。PC の LAN ポートに空きがあれば、LAN ケーブルで実機と接続してください。空きがない場合は、USB-LAN アダプターを用意して接続してください。

図 24　USB-LAN アダプター

次に LAN 接続のプロパティを開き、IPv4 のプロパティで IP アドレスとサブネットマスクを設定します。

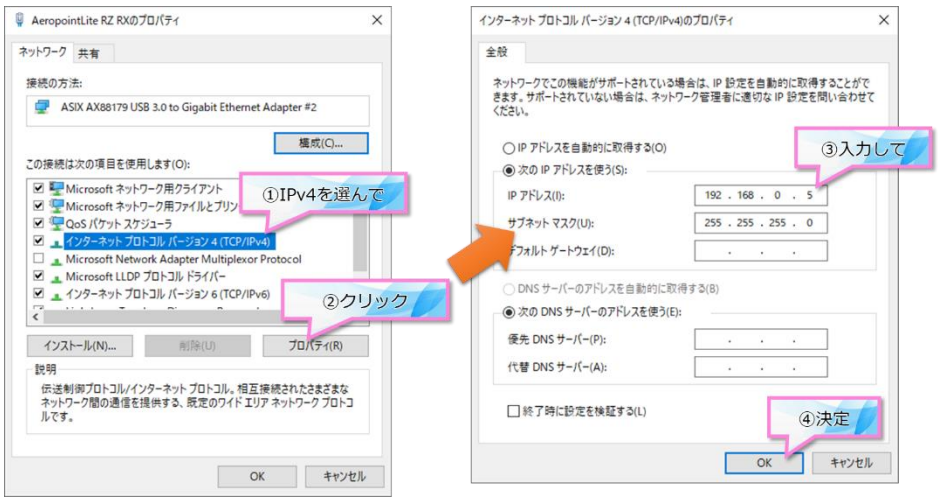


図 25　LAN アダプターのプロパティ

IP アドレス：192.168.0.5
サブネットマスク：255.255.255.0

実機の IP アドレスは、ランタイムのプログラムによって 192.168.0.2 で固定されます。

Aeropoint ツールキットは、「192.168.0.5」アドレスの LAN アダプターに接続されている、「192.168.0.2」アドレスを実機としてアクセスします。ポートは 7743 番を利用します。

3.2.8 D-Amp 基板の接続

D-Amp Driver による発音を行うために、D-Amp 基板を CN6 端子に接続します。

D-Amp 基板は 2 種類ありますので、お手元の基板と図面を比べながら接続方法を確認してください。共通 D-Amp 基板の場合は、接続ケーブルを介して正面中央付近の CN6 端子に差し込みます。
RX72N 専用 D-Amp 基板の場合は、そのまま CN6 端子に差し込みます。

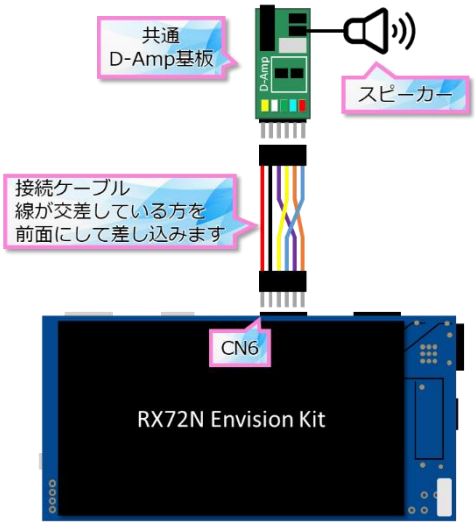


図 26　共通 D-Amp 基板接続の様子

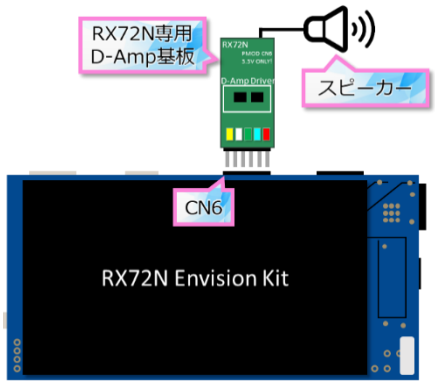


図 27　RX72N 専用 D-Amp 基板接続の様子

基板の接続ができたら、スピーカーともつないでおきます。
共通 D-Amp 基板では「J2」「J3」「J4」のお好きな端子につないでください。
RX72N 専用 D-Amp 基板の場合は、裏面にある「J1」「J2」のお好きな端子につないでください。
スピーカーの配線は 2 本ありますが、各端子のどちらのピンに接続しても構いません。

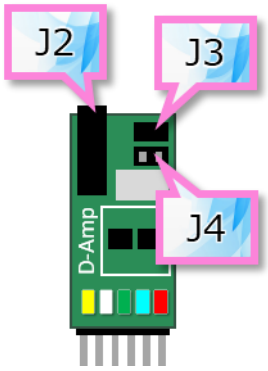


図 29　共通 D-Amp 基板の
スピーカー接続端子

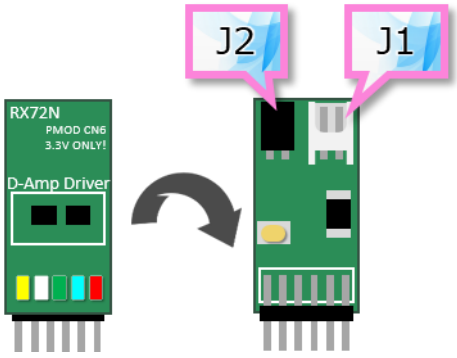


図 28　RX72N 専用 D-Amp 基板の
スピーカー接続端子

3.2.9 配置例

すべて配置すると、以下のような様子になります。

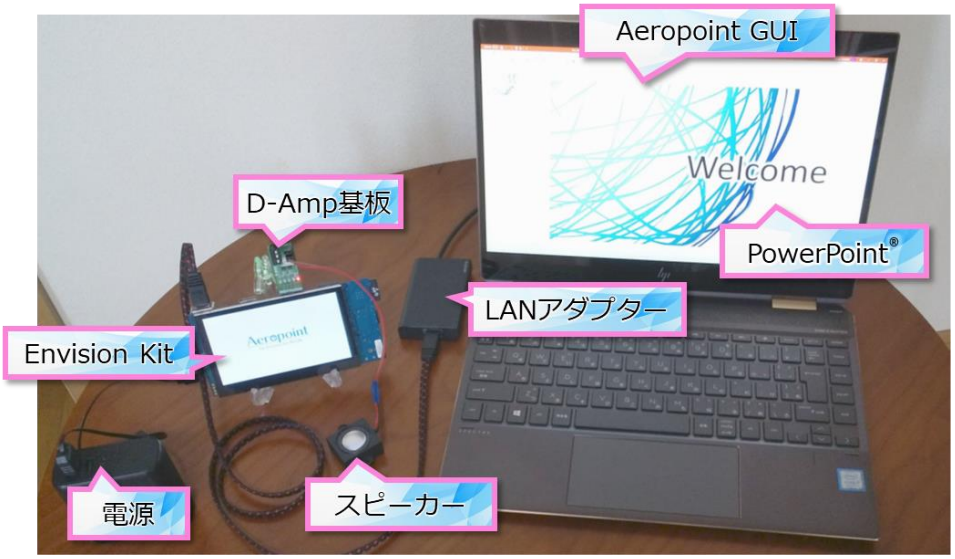


図 30　配置例

3.3 リファレンスキットの動作確認

3.3.1 Welcome スライドの表示

Welcome サンプルを使って、配置した機材が正しく動作するか確認していきます。最初に実機の電源を投入して、「スタートアップ画面」を表示させます。

図 31 スタートアップ画面

次に「SW2」を押します。

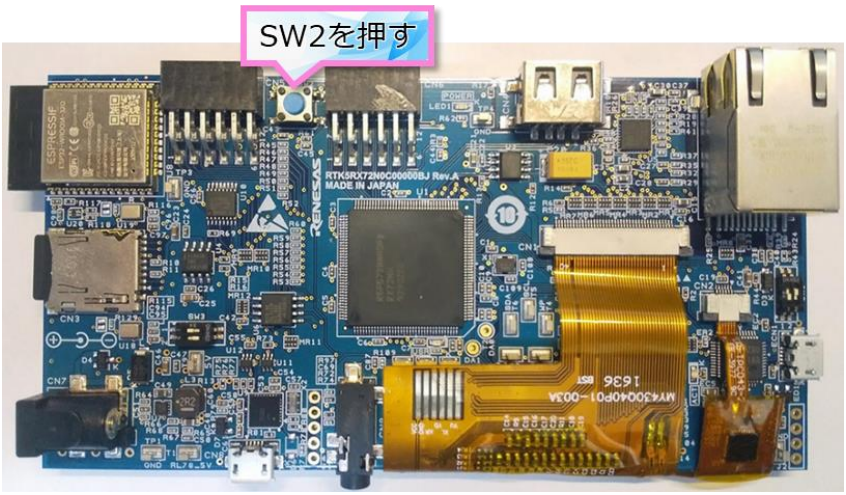


図 32 SW2 を押す

Aeropoint プレイヤーが、SD カードの先頭に登録されている Welcome スライドを読み出して表示します。

図 33 Welcome スライド

もう一度「SW2」を押して、スタートアップ画面に戻しておきます。

次に、開発 PC で PowerPoint®を起動して、リファレンスキットに入っている「Sample_Welcome.pptx」を開きます。

PowerPoint®上部の「AeropointLite」タブを選択し、リボンに表示される「再生」ボタンをクリックします。

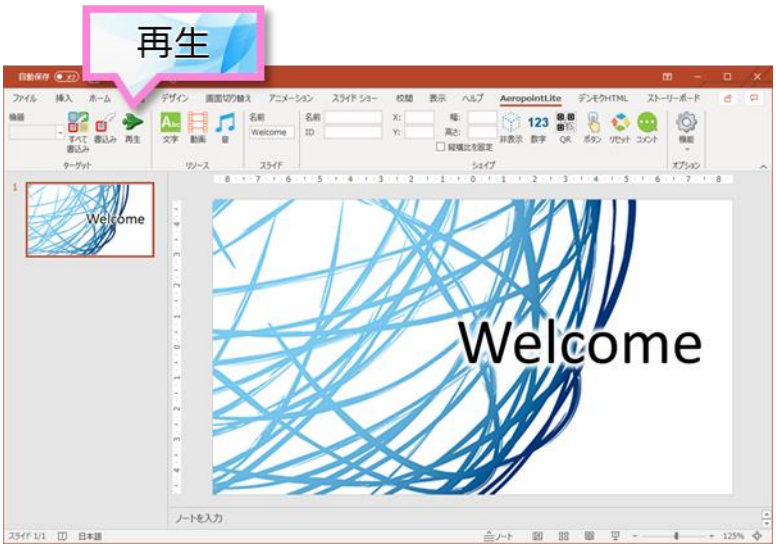


図 34 Welcome サンプルを再生

Aeropoint ツールキットと実機を仲介する「コネクター」アプリケーションが起動し、1 番目のスライドを表示するコマンドを発行します。

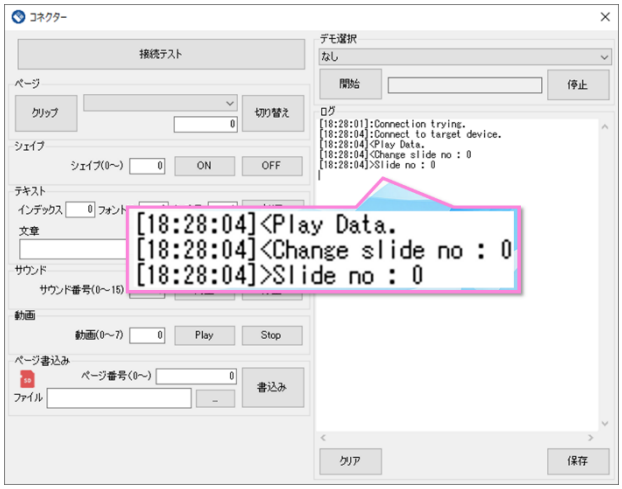


図 35　コネクターとコマンド

コマンドが正しく実機に届いたら、先ほどと同じように Welcome スライドが表示されます。

図 36　改めて Welcome スライドが表示される

うまく表示されずに、コネクターのログにエラーが表示されるようなら、タイムアウトなどが発生していることがあります。コネクターを終了したり、実機の電源を入れ直したりするなどして、うまく接続できるか試してください。それでもうまくいかない場合は、改めて「3.2.7 LAN 接続」の設定を見直し、IP アドレス等を確認してください。

再生ボタンは SD カードに格納された 1 番目スライドを表示します。PowerPoint®で作成したスライドを表示させるには、SD カードの内容を更新する必要があります。先ずは適当に Welcome スライドの見た目を変更します。

以下の図では、適当な図形を追加しました。

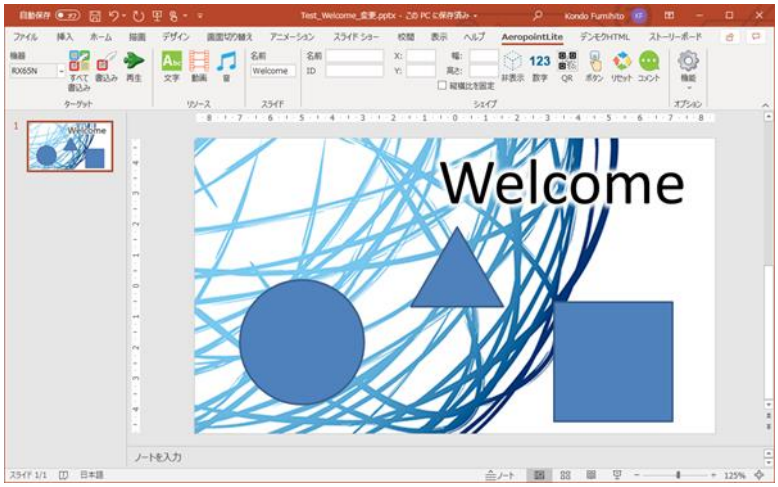


図 37　適当に変更

次に「すべて書込み」をクリックします。実機に全てのデータが書き込まれると、コネクターのログに書込み終了のメッセージが表示されます。

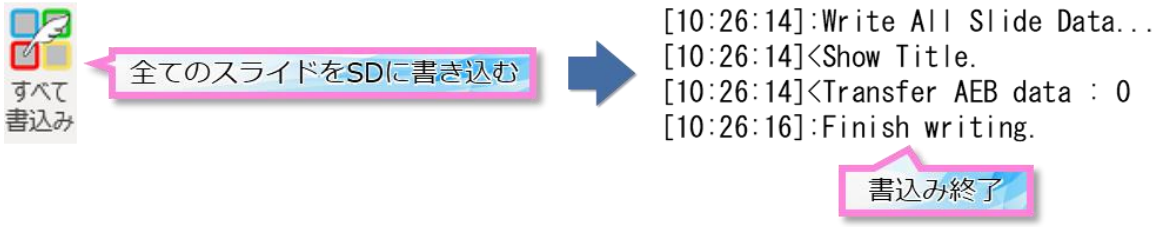


図 38　全て書込み

改めて「再生」をクリックして、実機の表示内容が更新されたら成功です。

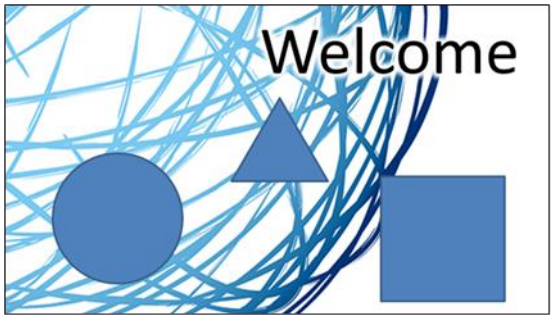


図 39　変更した内容を表示

書込み終了のメッセージがすぐに表示され、実際には書き込めていない場合は、SD カードの内部エラーが発生している場合があります。SD カードを抜き差しして、もう一度ご確認ください。

3.3.2 サウンド再生

D-Amp の基板とスピーカーを正しく接続した状態で、コネクターの「サウンド番号」に適当な値（0~5 くらい）を入力したら、「再生」をクリックします。D-Amp 基板に接続したスピーカーからサウンドが鳴ります。
音量の数値を指定して「設定」をクリックすることで、ボリュームの調整を行うこともできます。

図 40　サウンド再生

3.3.3 キャプチャ

正しく LAN 接続された状態で、「クリップ」をクリックしてしばらく待つと、実機に表示されている画面が PC のクリップボードへコピーされます。

図 41　クリップボードへ

キャプチャした画像は、ペイントなどのアプリケーションに貼り付けて利用します。

図 42　貼り付けて利用

これらの仕組みがうまく動いたら、リファレンスキットの動作確認は終了です。

3.4　導入済みサンプル

リファレンスキットにはいくつかの導入済みのサンプルがあります。これらのサンプルはコネクターを用いて動作の様子を確認できます。

3.4.1 会議室デモ

コネクターのデモ選択で「会議室」を選んだら、「開始」をクリックします。

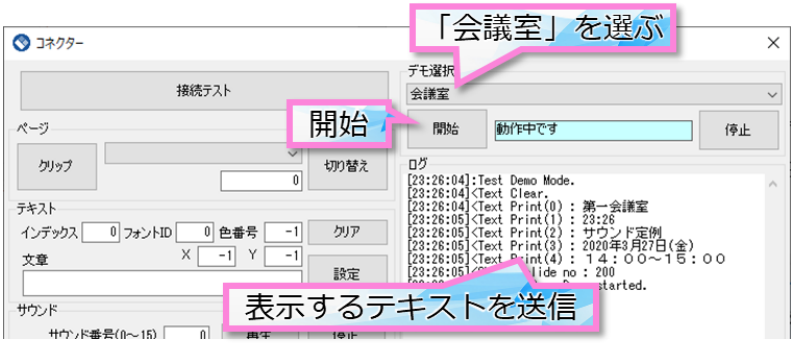


図 43　会議室デモを起動

コネクターの中でデモプログラムが起動し、会議室デモの表示に必要な「会議室名」や「時刻」、「会議の名称」などのテキスト情報を送信、続けて「空室」画面の表示を指示します。
実機がそれらの指示を受け取り、画面が表示されれば成功です。

会議室デモは「入室」と「退出」ボタンを押すと画面を切り替えられます。

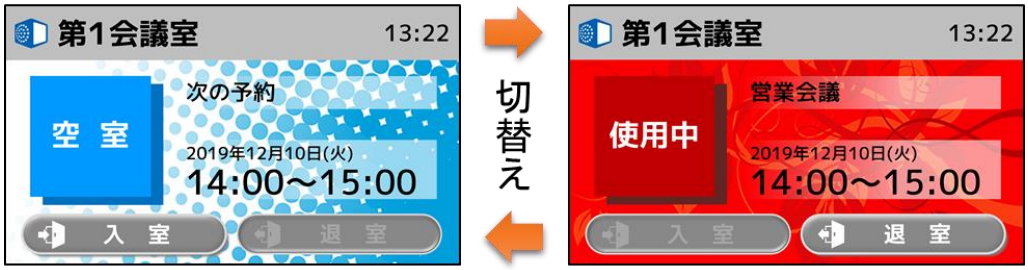


図 44　会議室の画面切り替え

またコネクターのデモプログラムは分単位の時間の変化を検知し、実機に再表示の指示を出します。

```
[11:38:00]<Text Print(1) : 11:38
[11:38:00]<Update Screen.
[11:39:00]<Text Print(1) : 11:39
[11:39:00]<Update Screen.
[11:40:00]<Text Print(1) : 11:40
[11:40:00]<Update Screen.
[11:41:00]<Text Print(1) : 11:41
[11:41:00]<Update Screen.
```

図 45　時間の再表示

会議室デモで表示されている黒文字のテキストは、内容を動的に変更できるフォントシステムを用いて表示しています。コネクターからも表示を切り替えられますので、試してみます。

会議室の名称を変更します。インデックスは「0」、好きな文章を入力したら「設定」をクリックします。

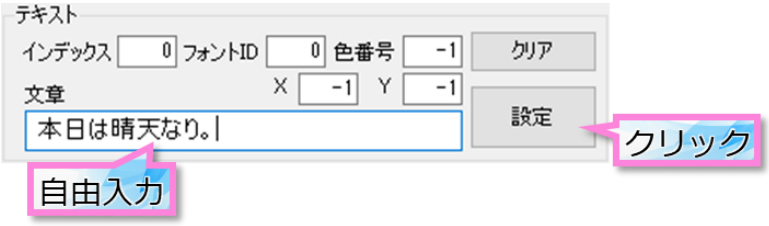


図 46　会議室の名称を変更する

実機はすぐに反応し、表示内容が切り替わります。

図 47　好きな文章に替える

今回のデモでは、フォント ID「0」で日本語の組込みフォントが利用できます。組込みフォントは UTF-8 の 3 バイトコードまでに対応し、最大で 8 種類まで追加できます。詳しくは「9 フォント作成」をご確認ください。

コネクター右上の「停止」ボタンを押すと、デモは終了してスタートアップ画面に戻ります。

3.4.2 トイレリモコン、血圧・脈拍計、ホームコントロール

他にもいくつかのデモがありますので、コネクターで起動してみましょう。

「トイレリモコン」では、PowerPoint®で気軽に画面を増やせてしまうという利便性を活かして、多言語対応をコピー&ペーストと色味や背景の変更だけで済ませてしまいました。

図 48　トイレリモコン

「血圧・脈拍計」ではグラフや数値の変化を確認できます。デモは自動進行で、数秒に一度だけ画面を更新します。

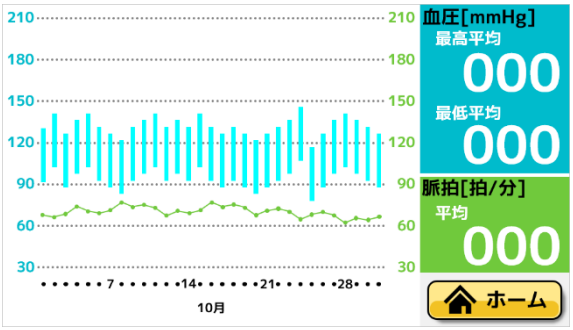


図 49　血圧・脈拍計

「ホームコントロール」では、D-Amp 基板上の LED を制御します。お客様からリクエストをいただければ、任意の GPIO を制御するようにしたランタイム SDK を提供できます。

図 50　ホームコントロール

3.4.3 サウンドテスト

デモ選択で「サウンドテスト」を選択し、開始します。

図 51　サウンドテスト

それぞれのボタンを押すと、D-Amp 基板経由でスピーカーが鳴動します。左列が SE、右列が BGM です。

BGM と SE は個別に再生できるため、音楽をかけながらも UI の SE を絶やさずに済みます。

長尺のサウンドを再生するため、D-Amp は SD カードからのストリーミング再生モードにて実装されています。

また、Aeropoint は他にも画面やフォントに関する情報などを SD カードから読み出しますが、それぞれの読み出しがサウンドの再生を止めてしまうことのないように、ダイナミックローディングを行っています。

3.4.4 ビデオテスト

デモ選択で「ビデオテスト」を選択し、開始します。

図 52　ビデオテスト

ボタンを押すと、対応した動画を再生します。動画を再生中に画面をタッチすると、ビデオテスト画面に戻ります。

3.4.5 I/O テスト

デモ選択で「I/O テスト」を選択し、開始します。

D-Amp 基板の LED の点灯と消灯、背面スイッチのセンシングを行います。

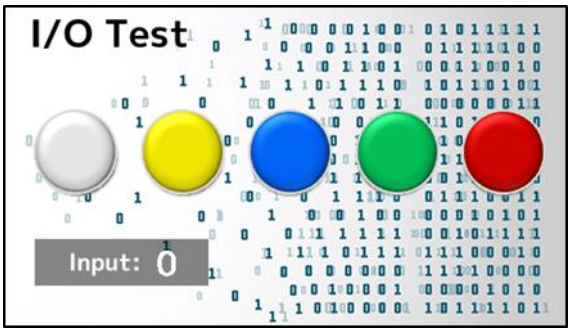


図 53　I/O テスト

4 FIT キット

FIT キットは、お客様のカスタムボードで Aeropoint を動作させる仕組みです。すでにリファレンスキットを用いて Aeropoint ランタイムの動作を確認し、次にツールキットを用いた画面の制作方法を知りたい方は、本章は飛ばして読み進めてください。本章を後回しにしても、Aeropoint の全体の理解に支障はありません。

4.1 対応マイコン

FIT キットは現在の所、以下のマイコンに対応しています。

コンポーネント	デバイス
RXファミリ	RX651, RX65N (ROM容量1.5MB-2MBのみ) RX72M, RX66N, RX72N

図 54 対応マイコン

ここに表示されていないコンポーネントやデバイスについては、順次対応を進めさせていただきます。お急ぎのご要望などございましたら、CRI までお問い合わせください。

4.2 e²studio と QE for Display の準備

FIT キットを利用するためには、e²studio のバージョンを Ver.2021-01 以降に更新しておきます。また、あらかじめ「ディスプレイ対応開発支援ツール QE for Display」を導入する必要があります。それぞれ、ルネサス社のサイトを参考にしてください。

■ 統合開発環境 e²studio
→ https://www.renesas.com/software-tool/e-studio

■ ディスプレイ対応開発支援ツール QE for Display
→ https://www.renesas.com/software-tool/qe-display-development-assistance-tool-display-applications

4.3 FIT キットを用いたプロジェクトのセットアップ

FIT キットを利用して、プロジェクトを新規作成からスタートし、実機で画面が表示されるまでを確認していきます。リファレンスキットで用いた実績十分の RX72N Envision Kit ではなく、あえて RX65N Envision Kit を初期状態から使っていきますので、お客様がカスタムボードを使う場合に近い流れになると思います。

4.3.1 プロジェクトを新規作成する

プロジェクトの新規作成から始めていきます。「ファイル」→「新規」→「C/C++ Project」を選びます。

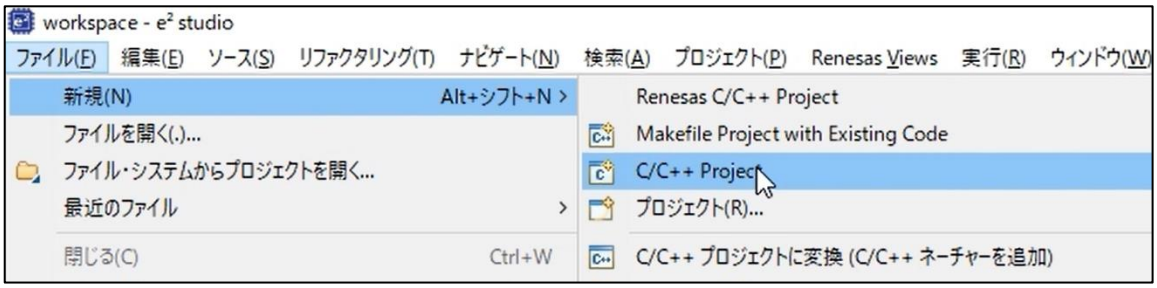


図 55 新規作成

プロジェクト作成ダイアログで「Renesas RX」リストを表示したら、「Renesas CC-RX C/C++ Executable Project」を選び、「次へ」をクリックします。

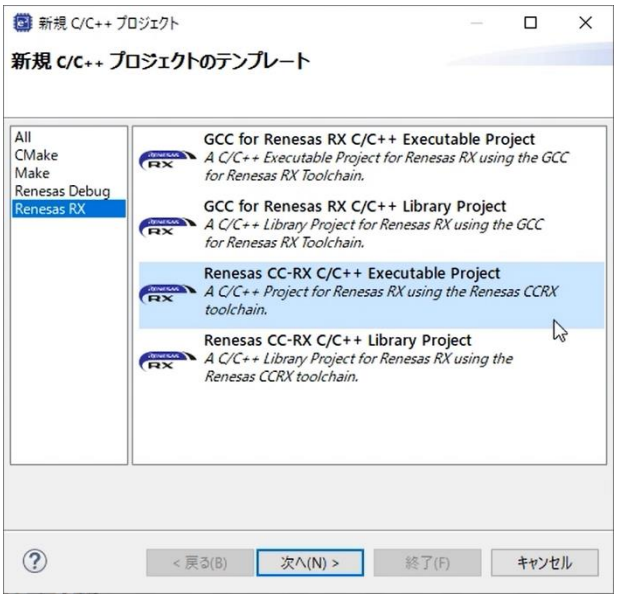


図 56 実行形式プロジェクトを選ぶ

プロジェクト名は「MyProject」にしました。プロジェクト名は半角英数字だけを利用し、数字で始まらないものにしてください。

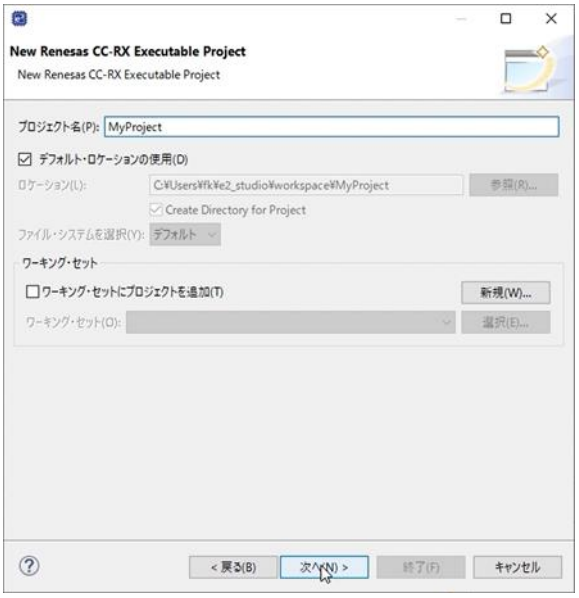


図 57　プロジェクト名を決める

ターゲットボードには「EnvisionRX65N」を選びます。ターゲットボードの選択肢がない場合は、e²studioをアップデートする必要があります。ハードウェアデバッグの構成には「E2 Lite」を選んでおきます。

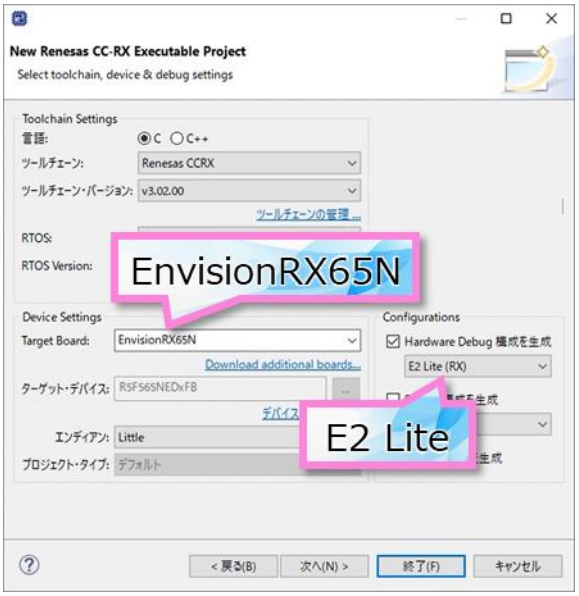


図 58　ターゲットボードなどの設定

スマートコンフィグレーターを有効にしたら「終了」をクリックしてプロジェクトを生成します。

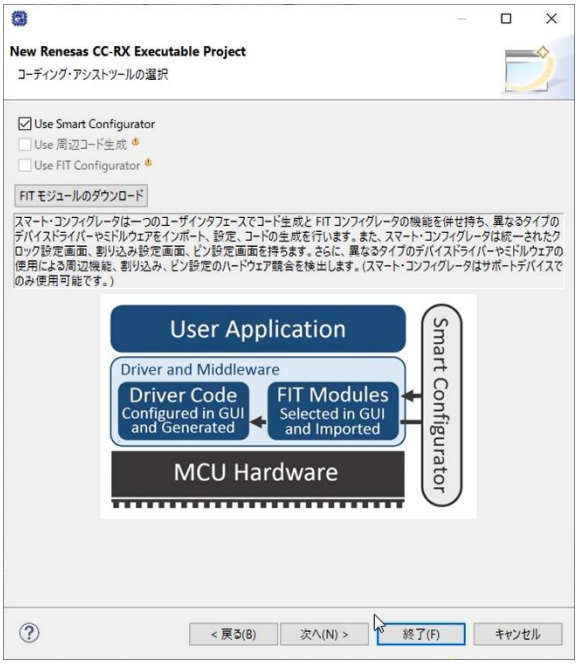


図 59　スマートコンフィグレーターを使う

生成がうまくいくと「MyProject」プロジェクトが出現します。

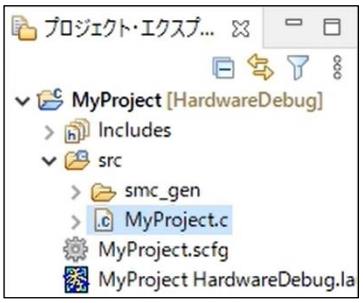


図 60 MyProject

4.3.2 QE for Display セットアップ

「メインメニュー」→「Renesas Views」→「Renesas QE」→「LCD メイン RX(QE)」を選択します。

図 61 QE for Display の起動

「LCD メイン RX(QE)」設定が開いたら、ワークフロー図の手順に沿って設定を行います。まずは、「1.準備」の手順を進めます。「プロジェクトの選択」プルダウンで、先ほど作ったプロジェクトを選びます。

図 62 プロジェクトを選ぶ

プロジェクトで設定したカスタムボードや MCU の設定を自動で読み取って、LCD 関連情報などが設定されています。手動で行うべき設定もありますので、必ず「指定方法」ボタンをクリックしてヘルプも確認します。

今回は、スマートコンフィグレーターでクロックの変更が必要です。「プロジェクト・エクスプローラー」タブにある「MyProject.scfg」ファイルをダブルクリックして、スマートコンフィグレーターを開きます。領域株の「クロック」タブを選んだら逓倍日を「x20.0」にして、コード生成をクリックします。

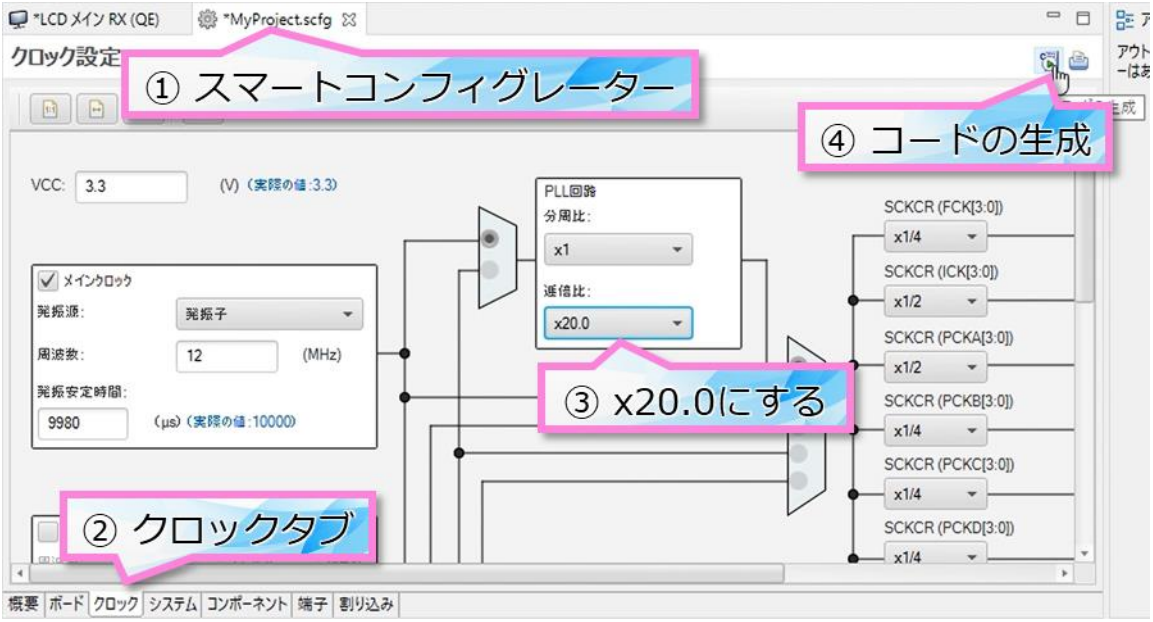


図 63　クロックの変更

「GUI 描画ツールの選択」で「AeropointGUI を使用する」を選んだら、「1.準備」は終了です。

図 64　AeropointGUI を選ぶ

次に、「2.LCD の調整」を進めます。「導入方法」で表示されるヘルプを読むと GLCDC コンポーネントを追加する必要があることが分かります。

再びスマートコンフィグレーターを開いたら、「コンポーネント」タグを選び「コンポーネントの追加」ボタンをクリックします。

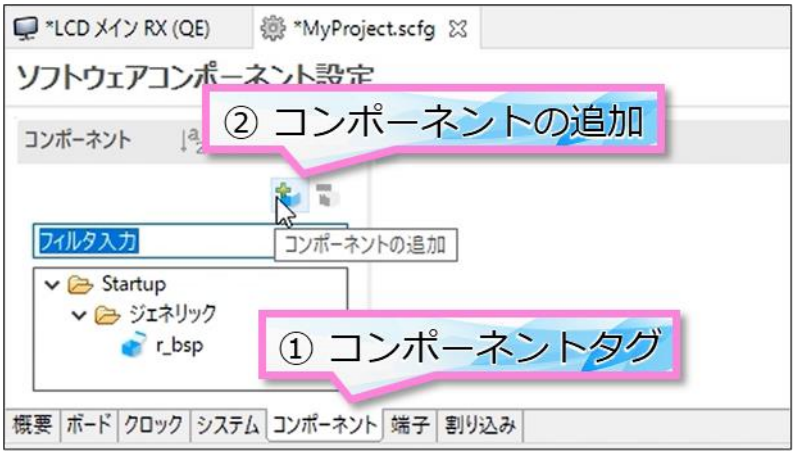


図 65 コンポーネントの追加

ソフトウェアコンポーネントの選択ダイアログが開いたら、GLCDC を選んで「終了」をクリックします。

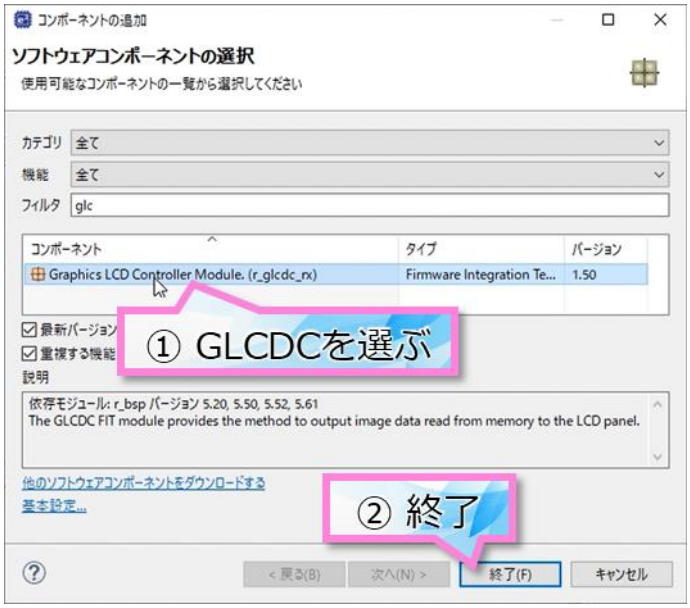


図 66 GLCDC を追加する

GLCDC が追加されますので、コードの生成をしておきます。

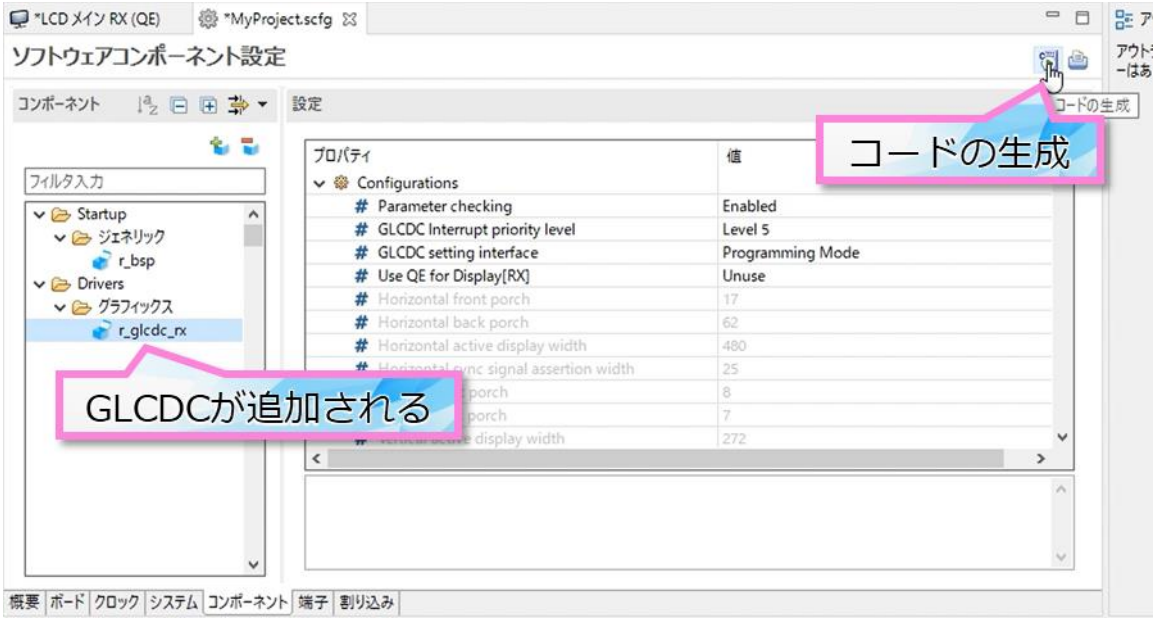


図 67　コードの生成

QE for Display に戻ると、LCD コントローラーが「導入済み」になっています。LCD の表示調整も自動で設定されていますが、一度開いて確認することで設定が有効になりますので、必ず一度は目を通すようにします。ユーザーボードで任意の LCD を利用している場合は、各 LCD の仕様に合った設定を施します。

LCD の設定が済んだら「ファイル出力」をクリックします。

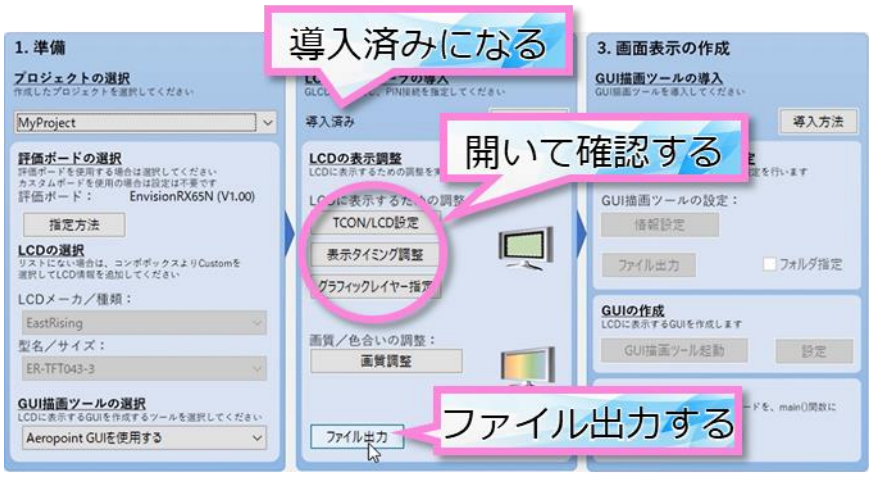


図 68　LCD の設定を出力する

最後に、「3.画面表示の作成」を進めます。スマートコンフィグレーターを開いたら、「コンポーネント」タグを選び「コンポーネントの追加」ボタンをクリックします。コンポーネントがとても多いので、「フィルタ」に「aero」とキーワードを入力して絞り込んだら、Aeropoint を選択して「終了」をクリックします。

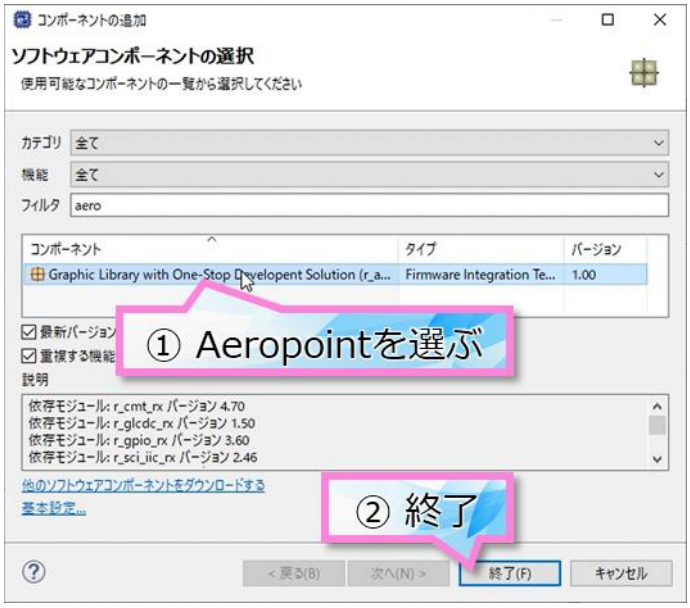


図 69　Aeropoint を追加する

Aeropoint とそれに紐づくコンポーネント群がツリーに追加されます。また、Aeropoint はデフォルトでタッチの仕組みを使う都合上、IIC の CH6 を Supported にしておく必要があります。設定が済んだら、コードの生成をクリックします。

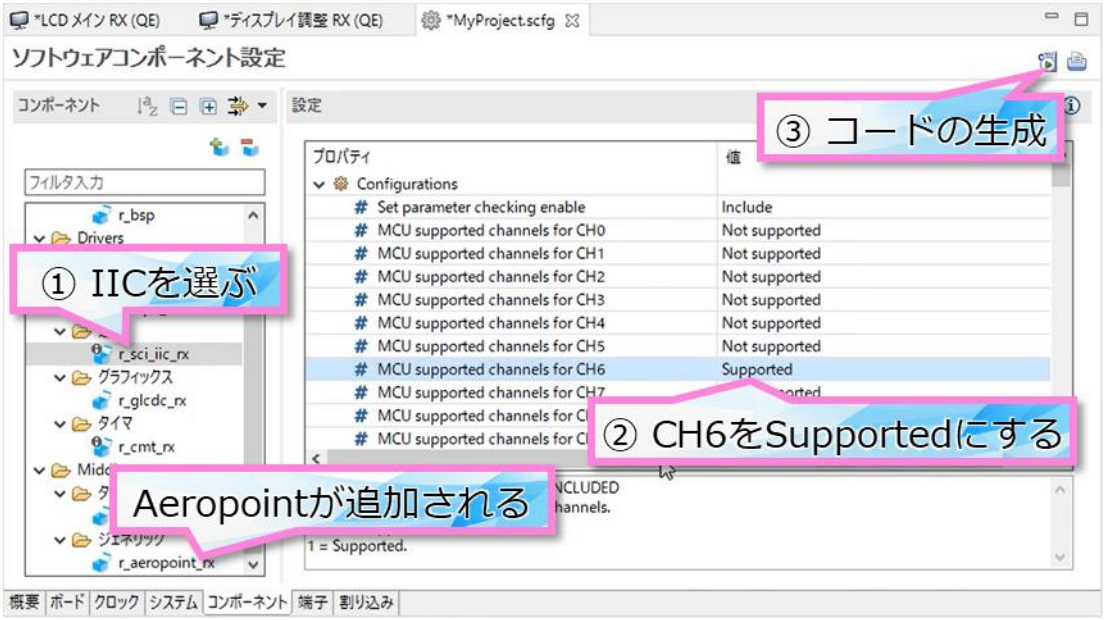


図 70 Aeropoint コンポーネントの追加と IIC の設定

QE for Display に戻ると、「GUI 描画ツールの導入」が「導入済み」と表示されていると思います。Aeropoint の設定を行うため「初期設定」をクリックします。

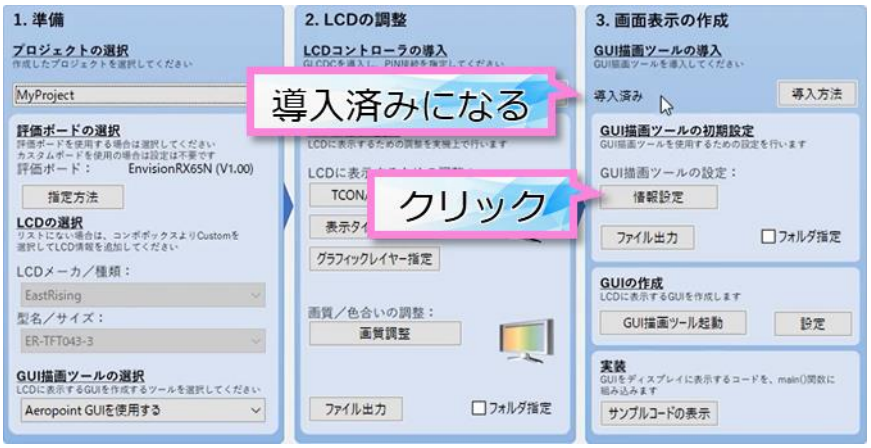


図 71　Aeropoint の情報設定へ

ダイアログの内容を確認していきましょう。

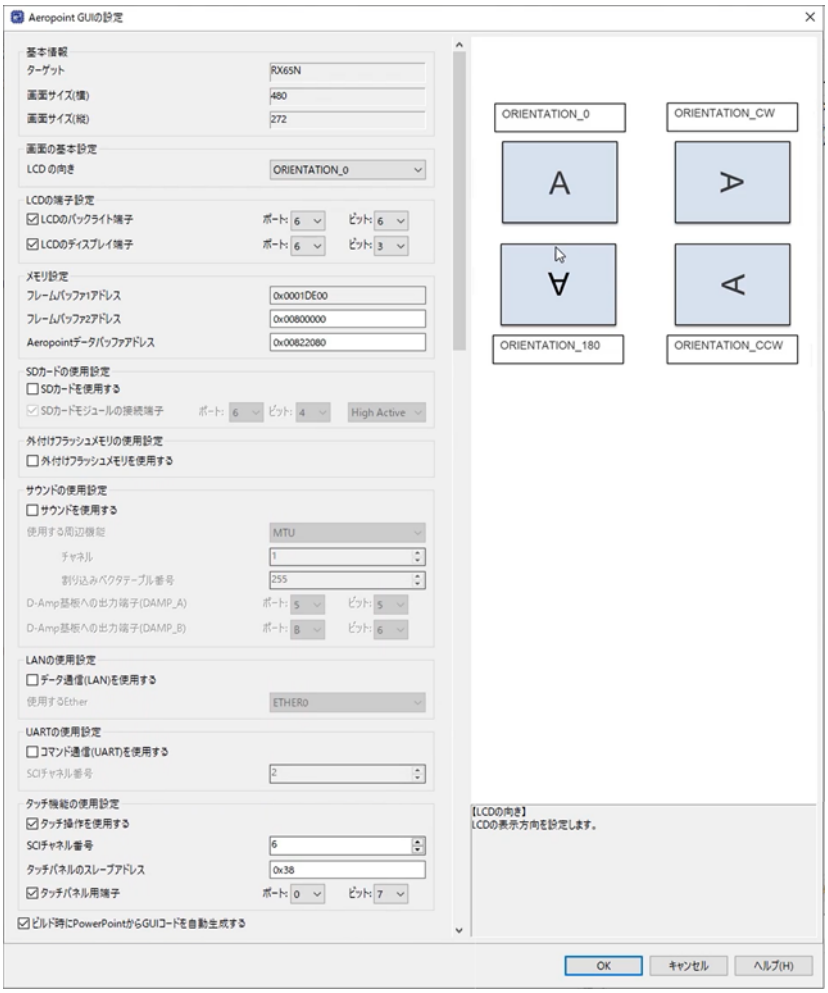


図 72　Aeropoint の設定

「基本情報」グループは、ここまでの設定で決まっています。

「画面の基本設定」グループで、LCD の向きを指定します。

「メモリー設定」グループで設定するフレームバッファーのアドレス以降には、それぞれ画面サイズ（縦）と画面サイズ（横）を掛け合わせた領域が必要です。

例）フレームバッファー１のサイズ　480×272 = 130,560 = 0x1FE00

「機能設定」では、Aeropoint が使用する機能を指定します。各機能は以下のような役割があります。

- SD カードの使用設定
- 外付けフラッシュメモリの使用設定
  - Aeropoint の画面、サウンド、ムービー、フォントなどを SD カードやシリアルフラッシュに格納し、必要に応じでランタイムエンジンが読み出して利用します。
  - 利用する方にチェックを入れたら、もう一方のチェックは外しておきます。
- サウンドの使用設定
  - D-Amp を用いたサウンド再生を行う場合にチェックします。D-Amp を用いたサウンド再生には、専用の回路が必要です。また、現在のところ使用するチャンネルは ch1 で固定です。今後のバージョンで指定できるようになります。
- データ通信（LAN）の使用設定
  - Aeropoint ツールキットと通信を行い、SD カードやシリアルフラッシュなどにデータを書き込んだり、デバッグのためのコマンド通信を行ったりできます。
- コマンド通信（UART）の使用設定
  - コマンド通信を用いて Aeropoint ランタイムを制御できるようにします。
- タッチ機能の使用設定
  - ソフトウェアコンポーネントで IIC の CH6 を Supported にしたのは、このダイアログのタッチの設定で、チャンネル番号が「6」に設定されているからです。

Aeropoint の各機能は下図のように FIT モジュールに依存しています。Aeropoint の「機能設定」で機能を追加する場合は、紐づくコンポーネントの追加やコードの生成も行ってください。

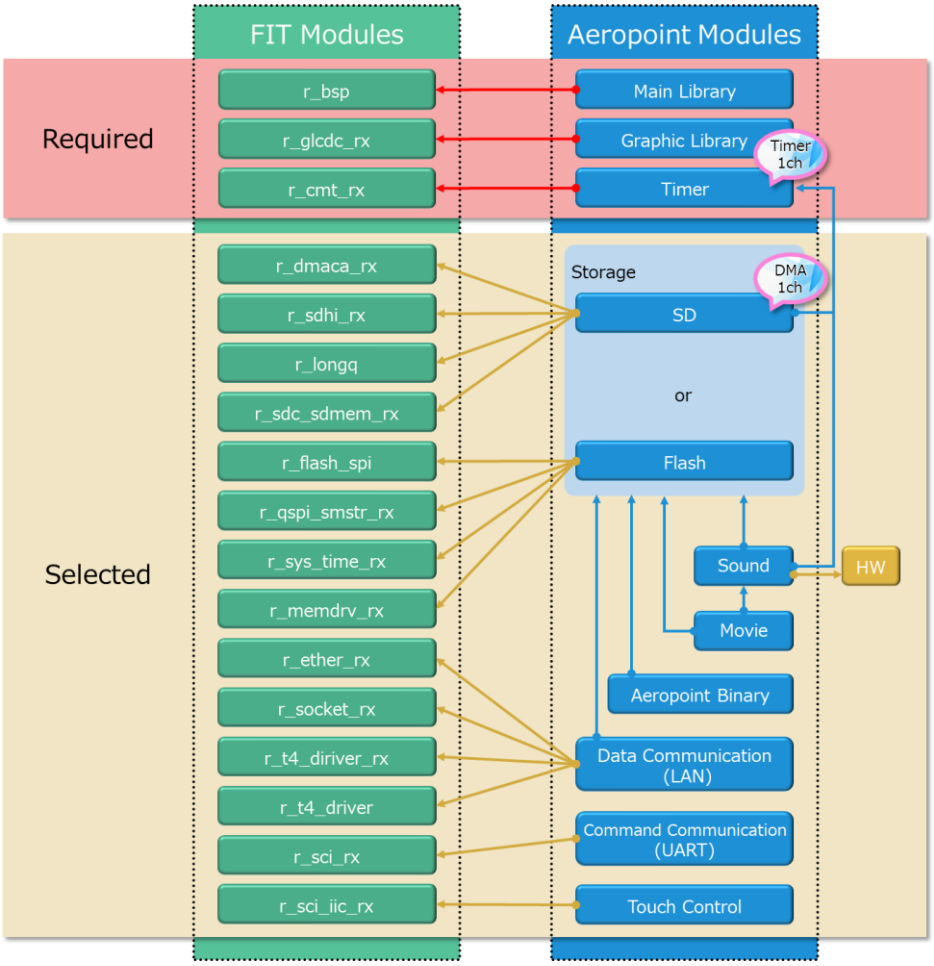


図 73 FIT モジュールと Aeropoint モジュールの依存関係

今回はデフォルト設定のまま「OK」をクリックして、ダイアログを閉じます。

「ファイル出力」をクリックすると、Aeropoint や紐づくコンポーネントのコードが出力されます。

図 74　ファイル出力

これで、プロジェクトのセットアップは終了です。

4.4　GUI の作成

次に「GUI の作成」を進めていきます。ご利用の PC に Aeropoint のツールキットがない場合は「設定」をクリックし、ウィザードの流れに従ってインストールを行います。インストールできたら、「GUI 描画ツール起動」をクリックし、ツールを起動します。

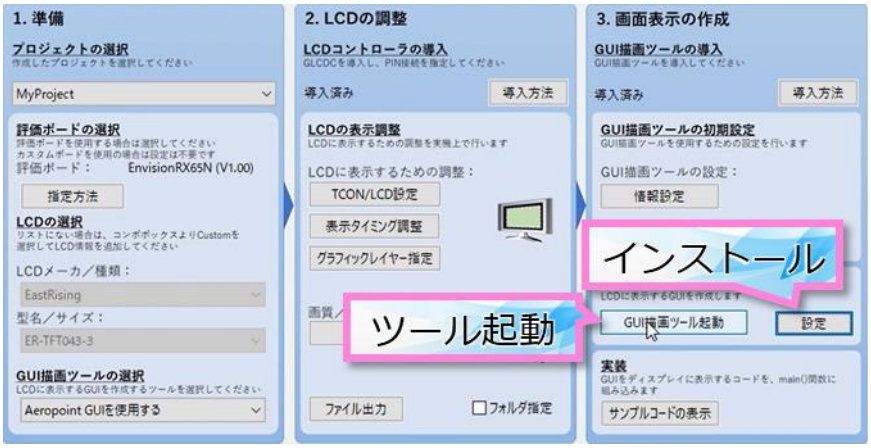


図 75　インストールとツール起動

Aeropoint のアドインが追加された PowerPoint®が起動します。どんなものでも良いので、好きな GUI を描いてみましょう。

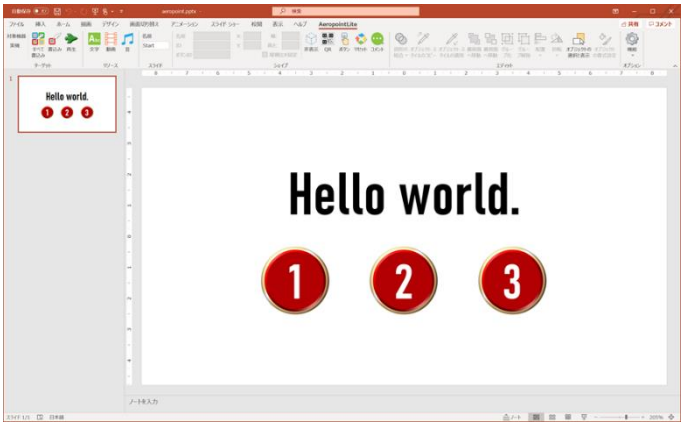


図 76　好きな GUI を描く

GUI が描けたら上書き保存しておきます。ここで作成した PPTX ファイルは自動で MyProject に追加されて、e²studio の管理対象となります。ビルドを行うにあたっては、GUI の中身が更新されていたらデータのエクスポートが行われ、他のコードと同様にコンパイルやリンクの手順をたどって実行形式ファイルに織り込まれます。

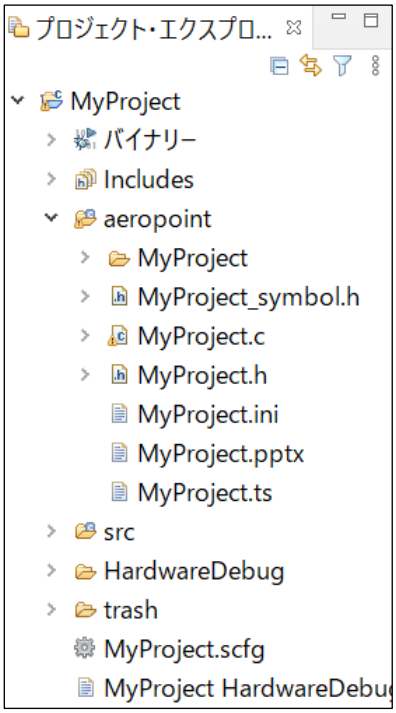


図 77　PPTX ファイルが追加されている

4.5　ライブラリ

FIT キットのランタイムを利用するには、FIT モジュールのほかに、Aeropoint のランタイムライブラリをプロジェクトにリンクする必要があります。生成した FIT キットの「lib」フォルダにあるライブラリを、デバイスや命令セットの組み合わせに応じてご利用ください。

デバイス	命令セット	Aeropointライブラリ
RX65	RXv2	libcri_Atom_Multiplayer_RXv2_CCRX.lib libcri_AeropointLite_RXv2_CCRX.lib
RX72	RXv2	libcri_Atom_Multiplayer_RXv3_CCRX.lib libcri_AeropointLite_RXv2_CCRX.lib
	RXv3	libcri_Atom_Multiplayer_RXv3_CCRX.lib libcri_AeropointLite_RXv3_CCRX.lib

図 78　ランタイムライブラリ

4.6　プログラムの作成と実行

QE for Display には作成したプロジェクトにあったサンプルを表示する機能があります。先ずは画面を出したいという場合には、先ずは「サンプルコードの表示」をクリックしてサンプルコードを表示します。

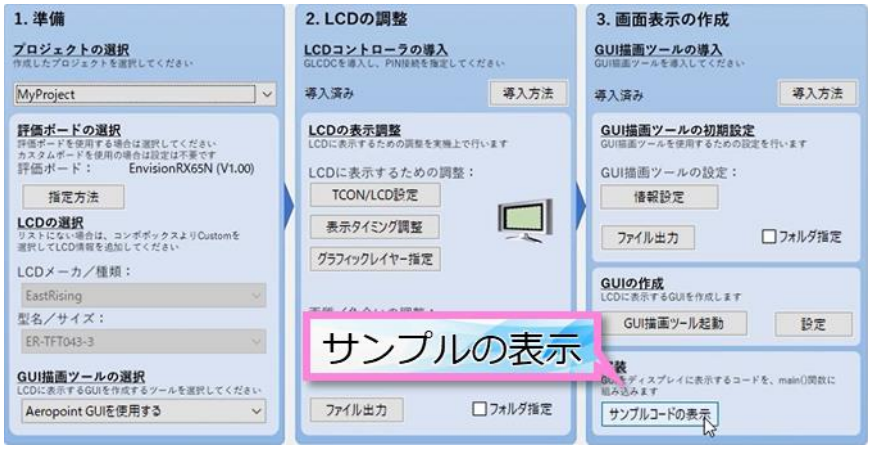


図 79　サンプルコードの表示

サンプルコードが表示されたら「クリップボードにコピー」をクリックします。

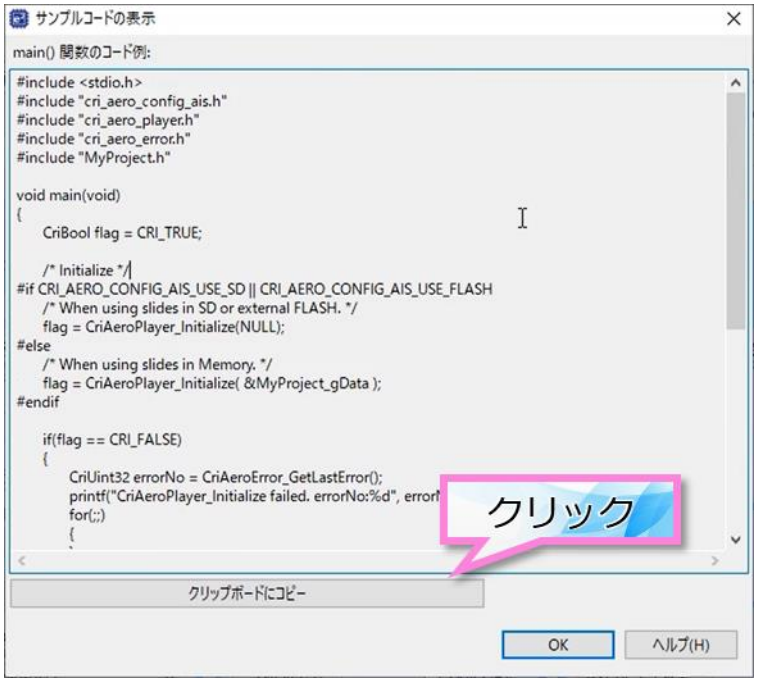


図 80　サンプルコードをコピー

次に、主プログラムである「MyProject.c」に貼り付けます。

```
1  #include <stdio.h>
2  #include "cri_aero_config_ais.h"
3  #include "cri_aero_player.h"
4  #include "cri_aero_error.h"
5  #include "MyProject.h"
6
7  void main(void)
8  {
9          CriBool flag = CRI_TRUE;
10
11         /* Initialize */
12 #if CRI_AERO_CONFIG_AIS_USE_SD || CRI_AERO_CONFIG_AIS_USE_FLASH
13         /* When using slides in SD or external FLASH. */
14         flag = CriAeroPlayer_Initialize(NULL);
15 #else
16         /* When using slides in Memory. */
17         flag = CriAeroPlayer_Initialize( &MyProject_gData );
18 #endif
19
20         if(flag == CRI_FALSE)
21         {
22                 CriUint32 errorNo = CriAeroError_GetLastError();
23                 printf("CriAeroPlayer_Initialize failed. errorNo:%d", errorNo);
24                 for(;;)
25                 {
26                 }
27         }
28
29 #if 1
30         /* Switch from title to slide 1. */
31         /* You can also switch slides by LAN or UART command. */
32         CriAeroPlayer_ReadSlide( 1 );
33 #endif
34
35         /* Main loop */
36         while (flag)
37         {
38                 /* Main */
39                 flag = CriAeroPlayer_Main();
40         }
41
42         /* Finalize */
43         CriAeroPlayer_Finalize();
44
45         return;
46 }
```

図 81 FIT キットを用いた画面の表示

ビルドして実行すると、先ほど作った画面が表示されます。

5　Aeropoint ツールキットの機能

PowerPoint®にアドインされた Aeropoint ツールキットの機能を説明します。

5.1　ターゲット

「ターゲット」グループは「コネクター」ツールを介して実機と通信を行い、SD カードへのデータの書き込みなどを行います。

図 82　ターゲットグループ

5.1.1 全て書込み

「全て書込み」は、すべてのページをエクスポートして、接続されている実機の SD カードにデータを書き込みます。エクスポート時に自動操作を行うため、Undo バッファーがクリアーされます。こまめな保存をお勧めします。

例外として「非表示スライド」に設定しておくと、エクスポートの対象になりません。非表示スライドを以降のスライドは前詰めになり、番号の不一致が発生してしまいます。非表示スライドは、スライドの最後に移動しておきます。

図 83　非表示スライド

5.1.2 書込み

「書込み」は、選択中のスライドをひとつだけ SD カードに書き込みます。複数選択は無効です。

5.1.3 再生

「再生」は、「コネクター」ツールを起動して、実機に最初のスライドを表示するコマンドを送信します。

再生では、スライドを SD カードに書き込むことはしません。実機のスライドを更新したい場合は、全て書込みか、書込みを利用してください。

5.2　リソース

「リソース」は外部データを取り扱います。対象のデータは PowerPoint®ファイルに取り込まれます。

図 84　リソースグループ

5.2.1 文字

「文字」は、「9 フォント作成」ツールで作成したフォントデータ（AC1FC 形式ファイル）を PowerPoint®にインポートして、必要に応じて SD カードへ書き込みを行います。

フォントデータをドロップして「決定」をクリックすると、インポートを行います。実機と LAN 接続されている状態で、フォントのチェックボックスをチェックして「選択しているフォントを転送」をクリックすると、実機に挿入された SD カードにフォントデータを書き込みます。

図 85　フォントリストダイアログ

フォントや動画、音のデータはサイズが大きいため、書き込みに時間が掛かります。一定時間コネクターが応答しなくなる場合がありますが、転送が終わるまでそのままお待ちください。正しく転送が継続している場合は、実機の LAN ポートにある緑色の LED が点滅を繰り返しています。LED が点灯したままか、消灯しているときは転送エラーが発生していますので、改めて転送を試みてください。

5.2.2 動画

「動画」は、MP4 形式ファイルをインポートして、SD カードへの書き込みを行います。

MP4 形式ファイルのままインポートしますので、あまり大きなサイズの動画にはせず、実機の実情に合わせて変換したものをご利用ください。

図 86　動画リストダイアログ

動画の推奨形式は、以下の通りです。

- フォーマット
  - MP4 形式ファイルのみ
- 長さ
  - 30 秒まで
- 解像度
  - 480x272 ピクセル
    ※ RX72N Envision Kit の場合です。
- フレームレート
  - 15fps 以上
    ※ 実機では 15fps 固定の動画となります。
- サウンド
  - モノラル 24kHz

5.2.3 音

「音」は、WAV 形式や MP3 形式のサウンドファイルをインポートして、SD カードへの書き込みを行います。

サウンドは一括で SD カードへの書き込みを行います。

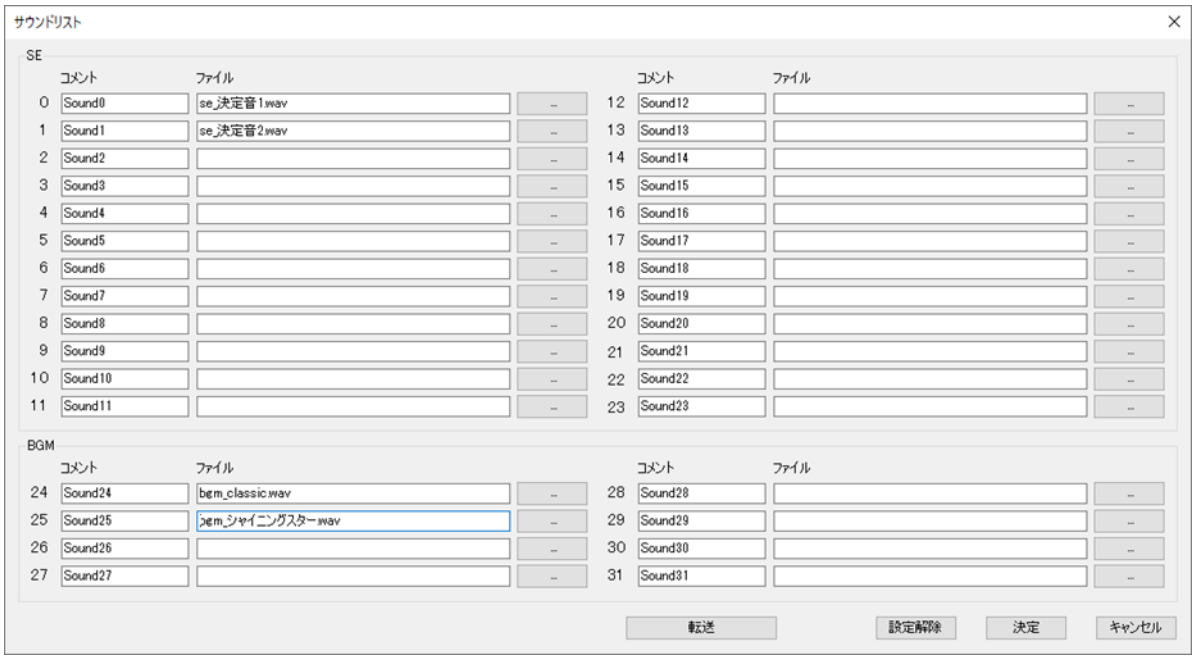


図 87　サウンドリストダイアログ

サウンドの推奨形式は、以下の通りです。

- フォーマット
  - WAV 形式や MP3 形式のサウンドファイル
- 長さ
  - SE は 10 秒程度まで
  - BGM は 3 分程度まで
- サンプルレート
  - 自動でモノラル 24kHz に変換される

5.3 スライド

図 88 スライドグループ

5.3.1 名前

スライドに名前を付けられます。名前に使用できる文字は「半角英数字＋'_'（アンダーバー）」のみとなります。

5.4 シェイプ

図 89 シェイプグループ

5.4.1 名前

シェイプ（図形）をひとつ選択して、名前を付けます。名前に使用できる文字は「半角英数字＋'_'（アンダーバー）」のみとなります。

5.4.2 ID

シェイプの ID は、デフォルトで「-1（無効）」です。無効なシェイプは他の同様のシェイプとまとめられて、実機では背景として最背面に表示されます。そのため、PowerPoint®の前後関係と異なる表示となる場合があります。

0 以上の数値を入力すると独立した部品となり、コマンドやプログラムからの制御を受けられます。

5.4.3 ボタン ID

選択中のシェイプにボタンの設定が加えられている場合に、ボタンとしての ID を表示します。ID の変更もできます。

5.4.4 X、Y、幅、高さ、縦横比を固定

選択しているシェイプの配置やサイズに関する情報を表示しています。数値を変更すると、シェイプの配置やサイズを変更できます。変更にあたって「縦横比を固定」にチェックを入れておくと、縦横比を維持しながら幅や高さの変更ができます。

5.4.5 非表示

「非表示」は、選択中のシェイプについて、実機で表示を行わないようにします。エクスポートも行わないため、出力するデータに画像の情報が入ることもありません。再度「非表示」をクリックすると、非表示状態から表示状態に戻せます。

実機で表示する必要のないレイアウトや、説明用のシェイプなどを非表示にして配置しておくと便利です。

5.4.6 QR

四角シェイプを選択して「QR」をクリックすると、QR コード設定ダイアログが開きます。QR コードとして表示したいデータを入力したら、「決定」をクリックします。

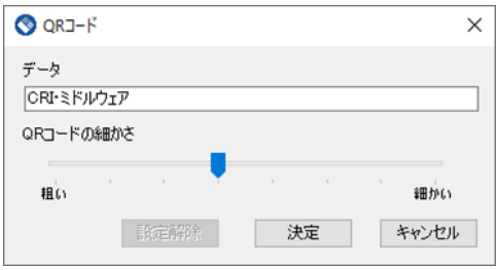


図 90　QR コードのデータ設定

QR を設定しても PowerPoint®上では四角シェイプのままですが、実機で表示すると QR コードとなります。

5.4.7 ボタン

「ボタン」は、選択中のシェイプにボタンとしての振る舞いを設定します。

図 91　ボタンダイアログ

名前に使用できる文字は「半角英数字＋'_'（アンダーバー）」のみとなります。ID に 0 以上の数値を入力すると、タッチされたときにプログラムからのコールバックを受けられます。

「スライド切り替え」を設定すると、タッチした際に前後のスライドや、指定したスライドに画面が切り替わります。

「効果音を再生する」や「動画を再生する」を設定すると、タッチした際に指定した効果音や動画が流れます。

「出力信号を制御する」を設定すると、テスト用の IO を用いて、D-Amp 基板上にある LED の点灯状況を変更できます。

5.4.8 リセット

「リセット」は、QR やボタンなどの設定を解除します。

5.4.9 コメント

「コメント」は、選択中のシェイプにコメントを設定できます。コメントを設定したシェイプを選択すると、リボンにコメントを表示します。

5.5 エディット

エディットは元々PowerPoint®に備わっている機能の中から、特に GUI を制作するにあたって便利なものをまとめたものです。

図 92 エディットグループ

5.5.1 図形の接合

２つ以上の（テキストを含まない）図形を選択すると、「図形の接合」メニューが利用できます。例えば下の図のように、重なり合う図形に様々な接合パターンを適用して、欲しい形状を作ります。

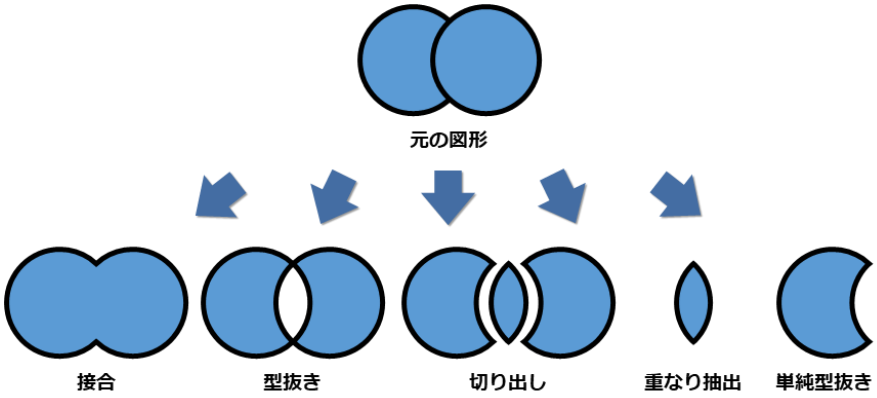


図 93 様々な接合パターン

中でもよく使うのは「単純型抜き」です。複数の図形を選択する際に最初に選んだ図形に対して、それ以降に選択した図形で型抜きできます。

図 94 単純型抜きの活用

図形の接合では PowerPoint®の標準図形にない形を作れるため、オリジナル UI に欠かせない独自性を表現できます。

図 95 オリジナルの十字キーと決定ボタン

5.5.2 フリーフォーム

ここでもうひとつ覚えておきたいテクニックが、「フリーフォーム」を活用した型抜きです。図形の接合の例で挙げた十字キーを作るにあたって、円形をくりぬいた直後は４つのキーでひとつの図形です。このままでは、個別のキー操作ができません。

そこで、まずは「フリーフォーム」で自由な図形を重ねて、次に図形の接合の「単純型抜き」で必要なところだけを切り出します。

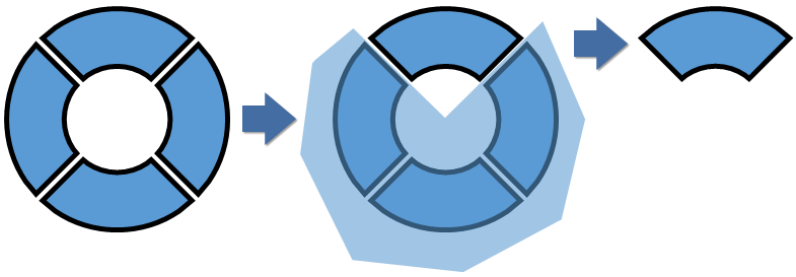


図 96　フリーフォームと単純型抜き

これを４回繰り返すと、４つのキーが作成できます。

5.5.3 オブジェクトスタイルのコピー&適用

PowerPoint®では、図形に対して色の変更や線の追加、テキストの書式や光の当たり具合など、様々な設定ができます。自分の思い通りのデザインを作れる一方で、たくさんの図形に個々に設定をやり直すのは大変です。そこで便利なのが、「オブジェクトスタイルのコピーと適用」です。

任意の図形を選んで「オブジェクトスタイルのコピー」したら、他の図形を選んで「オブジェクトスタイルの適用」します。

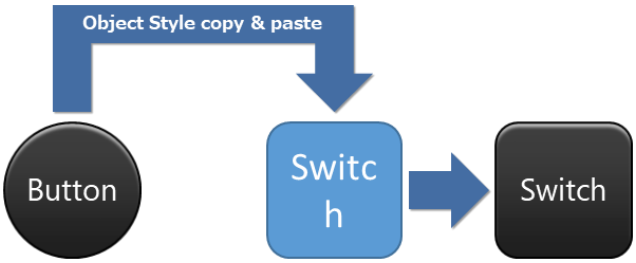


図 97　オブジェクトスタイルのコピーと適用

図形の形状や文章には影響はなく、設定だけをコピーできます。

5.5.4 最前面・最背面へ移動

GUI 制作の作業に慣れてくると、いくつかの図形を組み合わせた表現が増えてきます。新しく作る図形や、コピー＆ペーストで貼り付けた図形は常に前面の配置となるため、前後関係を調整したくなることがあります。そこで便利なのが「最前面・最背面へ移動」です。

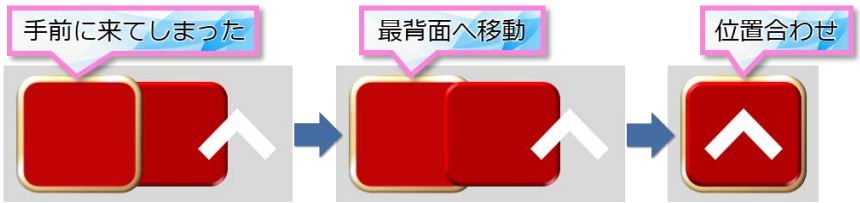


図 98　前後関係の調整

前後関係は複数の図形を選択することで、まとめて調整もできます。

5.5.5 グループ化・解除

複数の図形で作った部品を「グループ化」でひとつにまとめると、その後の配置や回転が楽になります。

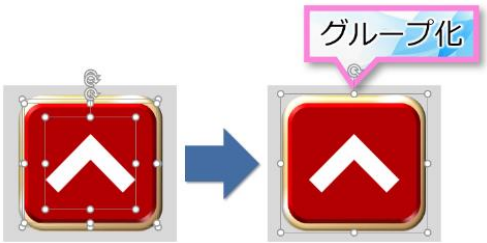


図 99　グループ化

グループ化はいつでも解除できます。また、グループ化したもの同士を再びグループ化もできます。

5.5.6 配置

図形をコピー&ペーストすると、少しずれた位置に再配置されます。マウスでドラッグして調整しようとしてもなかなか難しく、レイアウト変更の障害になりがちです。

そこで活躍するのが「配置」機能です。複数選択した図形の左右を揃えたり、均等に配置したり、中央合わせしたりできます。

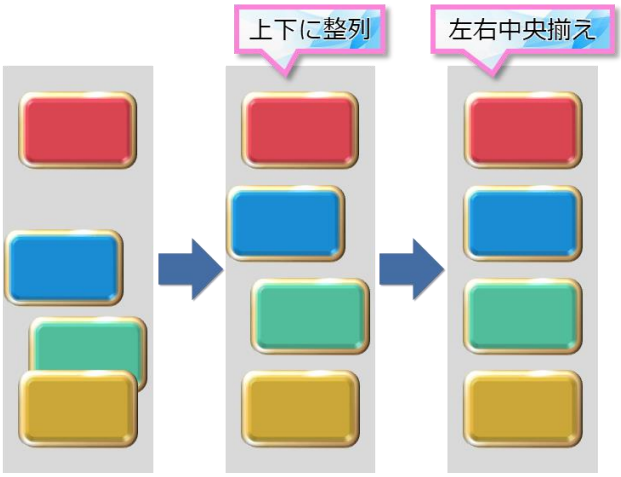


図 100　配置による整列と揃え

5.5.7 回転

上下反転、左右反転も良く使う機能です。角度を指定しての回転もできます。

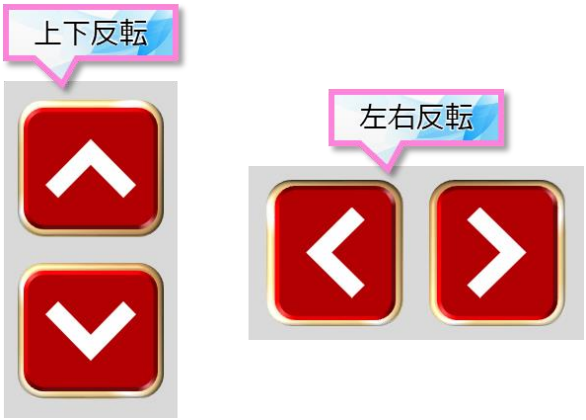


図 101　反転

5.5.8 オブジェクトの選択と表示

図形の前後関係や名前をまとめて管理できます。

図 102　オブジェクトの選択と表示

図形の名前の部分をクリックすると、名前を変更できます。グループ化や解除が行われると、名前はリセットされてしまいます。そのため、仕上げる際に前後関係や名前をまとめて調整するとよいでしょう。

Aeropoint ランタイムを用いてプログラミングを行う場合には、ここで設定する名前を使って図形を参照する場合があります。プログラムから図形を参照するにあたっては「英数字」だけが利用できます。日本語などで付けた名前はエラーになるので注意が必要です。

5.5.9 オブジェクトの書式設定

「オブジェクトの書式設定」で、図形に様々な効果を追加します。

図 103　図形の書式設定

設定できる項目が多くて大変ですが、「塗りつぶし」や「線」はよく使うので、どんなことができるか試しておくと良いでしょう。

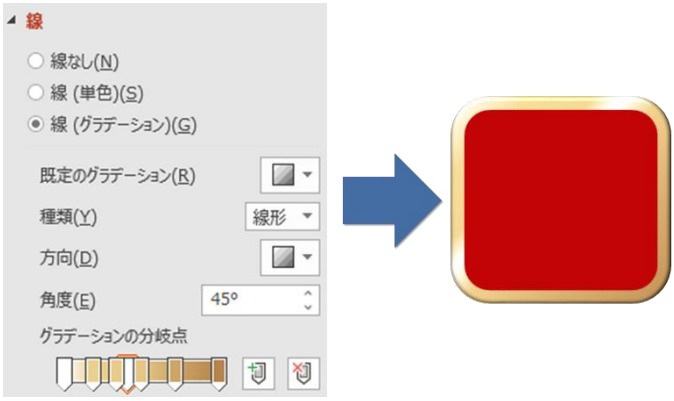


図 104　線に細かいグラデーションを付けて「金ピカ感」を出した例

5.6 オプション 機能

Aeropoint ツールキットの補助的な機能をまとめてあります。

図 105 オプショングループ

5.6.1 インポート

「インポート」は、Photoshop からエクスポートした画像をスライドとして取り込みます。Photoshop からのエクスポートはオプション機能です。ご入用の場合は「14 お問い合わせ」までご連絡ください。

5.6.2 エクスポート

「すべてエクスポート」は、すべてのスライド情報をデータ化して保存します。「エクスポート」は、現在のスライド情報をデータ化して保存します。保存されたデータのうち、AEB 形式ファイルは「コネクター」ツールを用いて SD カードへの書き込みができます。

「すべてエクスポート」では、スライドに設定されたシェイプのシンボル情報をまとめたヘッダーファイルも保存します。出力されるシンボル情報は、シェイプごとに設定されているシェイプ ID とボタン ID です。
保存されるシンボル情報は、Aeropoint アプリケーションを作成する際のプログラミングで使用できます。

```
/* Sample_Remocon Slide Main */
#define Sample_Remocon_Main_BackGround_ShapeID                    (-1)
#define Sample_Remocon_Main_Group_7_ShapeID                       (-1)
#define Sample_Remocon_Main_Group_7_ButtonID                      (6)
#define Sample_Remocon_Main_Group_139_ShapeID                     (-1)
#define Sample_Remocon_Main_Group_139_ButtonID                    (7)
#define Sample_Remocon_Main_Group_165_ShapeID                     (-1)
#define Sample_Remocon_Main_Group_165_ButtonID                    (0)
#define Sample_Remocon_Main_Group_170_ShapeID                     (-1)
#define Sample_Remocon_Main_Group_170_ButtonID                    (1)
#define Sample_Remocon_Main_Group_175_ShapeID                     (-1)
#define Sample_Remocon_Main_Group_175_ButtonID                    (2)
#define Sample_Remocon_Main_Group_180_ShapeID                     (-1)
#define Sample_Remocon_Main_Group_180_ButtonID                    (3)
#define Sample_Remocon_Main_Group_83_ShapeID                      (-1)
#define Sample_Remocon_Main_Group_83_ButtonID                     (4)
#define Sample_Remocon_Main_Group_89_ShapeID                      (-1)
#define Sample_Remocon_Main_Group 89_ButtonID                     (5)
#define Sample_Remocon_Main_xx0_ShapeID                           (0)
#define Sample_Remocon_Main_xx5_ShapeID                           (5)
#define Sample_Remocon_Main_x0x_ShapeID                           (10)
#define Sample_Remocon_Main_x1x_ShapeID                           (11)
#define Sample_Remocon_Main_x2x_ShapeID                           (12)
#define Sample_Remocon_Main_x3x_ShapeID                           (13)
#define Sample_Remocon_Main_x4x_ShapeID                           (14)
#define Sample_Remocon_Main_x5x_ShapeID                           (15)
#define Sample_Remocon_Main_x6x_ShapeID                           (16)
#define Sample_Remocon_Main_x7x_ShapeID                           (17)
#define Sample_Remocon_Main_x8x_ShapeID                           (18)
#define Sample_Remocon_Main_x9x_ShapeID                           (19)
#define Sample_Remocon_Main_1xx_ShapeID                           (100)
#define Sample_Remocon_Main_2xx_ShapeID                           (200)
#define Sample_Remocon_Main_3xx_ShapeID                           (300)
```

図 106　シンボル情報ヘッダーファイル

5.6.3 フォント作成

「フォント作成」は、フォントデータを作成するツールを起動します。
ツールの詳細は「9 フォント作成」をご参照ください。

5.6.4 解像度

「解像度」は、PowerPoint®の解像度を実機と同じにするために設定します。

- RX72N Envision Kit の場合
  - 幅：480 ピクセル
  - 高さ：272 ピクセル

5.6.5 設定

「設定」は、画像データの最適化に関する指定を行います。ほぼ最適なデータを自動で出力するため、変更する必要はありません。

6　作ってみる

ここでは「エアコンのリモコン」をテーマに、GUI を新規作成から作っていきます。せっかくなので、少々攻めたデザインにしてみましょう。この章では、担当である CRI・ミドルウェアの近藤が何を考えながらモノ作りをしているかをリアルに伝えるため、だいぶ砕けた感じで進めます。どうかご笑納くださいませ。

図 107　目指す完成形

また、制作期間の都合でリモコンの全機能を網羅できませんので、試作扱いとします。それでも GUI 開発のエッセンスは一通り伝えられるようにして、お客様自身が Aeropoint を用いた開発を開始できるようにします。

6.1　準備

6.1.1 新規作成

PowerPoint®のファイルを新規作成して、以下の名前で保存します。

- MyRemocon.pptx

ファイル名は「半角英数字＋'_'（アンダーバー）」だけにすると、トラブルが少ないです。また制作の過程においてはこまめな保存やバックアップをするように心がけます。

6.1.2 解像度変更

実機の解像度に合わせるために、PowerPoint®の設定を変更します。「Aeropoint アドインメニュー」→「オプション」グループ→「機能」→「解像度」設定を選びます。

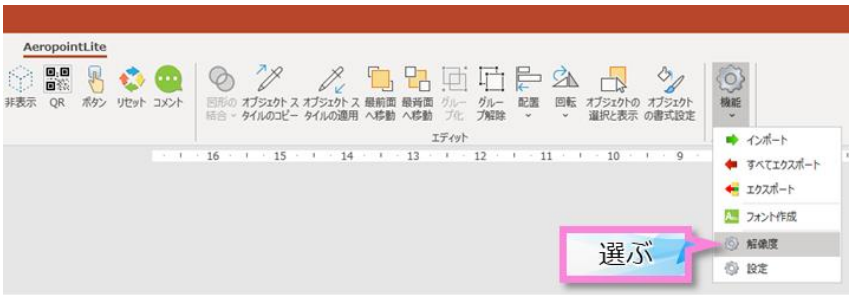


図 108　解像度設定を選ぶ

解像度を 480x272 に変更します。

図 109　解像度変更

6.2 どんな GUI にするか考える

製品開発を始めるにあたっては、要求要件や基本仕様が明らかになっているべきですが、試作やアイディア出しなどでは何もないところから作り始めることもしばしばです。ここでも、なんとなくこういう GUI だったら次の製品として目新しそうだな、という雰囲気だけを頼りにスタートしてみます。

6.2.1 ふんわりとした全体像

せっかく Aeropoint という、画面を作ったら数秒後に実機で試せる環境があるわけですから、何度も試行錯誤をしながら少しずつ良いものにしていく作戦にします。逆に言えば、色々とふんわりした状態で作り始めても怖くはないということです。そこで先ずは仕様書代わりに、なんとなく作りたいものの全体像を考えます。

- せっかくなので攻めたデザインで、高級感を出したい
  - 画面は横向き
  - 全てタッチ操作
  - 他の 3 倍くらいの性能が出そうな「赤」をイメージ
  - 高そうな雰囲気も出したいので、金ピカ感も出したい
  - 高級感を出すのに流行りのフラットデザインとかは無理ゲーなので、意図的に無視する

- 複数の画面にしたい
  - １画面に詰め込まないことで、操作が難しいイメージを回避
  - デザインがだいぶ自由になるはず

- LCD の解像度を活用して、運転状況や運転切替えボタンは表示し続けたい
  - 各画面に表示するボタンはコピペで増やせるから面倒くさくないだろう

- 色をたくさん使いたい
  - カッコ良いから

ふんわりした全体像からコンセプトをひとつに絞り込むなら「高級感」でしょうか。高く売れそうなイメージの方が、製品開発に GO を出してくれる偉い人達にもウケが良いというものです。

今はきっとこのくらい考えておけば十分です。

6.2.2 なんとなく形にする

全体的に赤いイメージでまとめるとして、じゃあどうやったら高級感が出せるかなんて分かりません。GUIの設計をする人はたいていデザイナーではないから、これは仕方ないです。そして困った時は「検索」です。「赤背景　商用フリー」など、身もフタもないキーワードを使います。

図 110　困ったら検索

良さそうな絵が見つかったら、制作者の方に感謝しながらスライドに貼り付けます。本当に商用化や公式に公開されるものなら、素材の権利関係を確認するか、有料の物を購入しましょう。

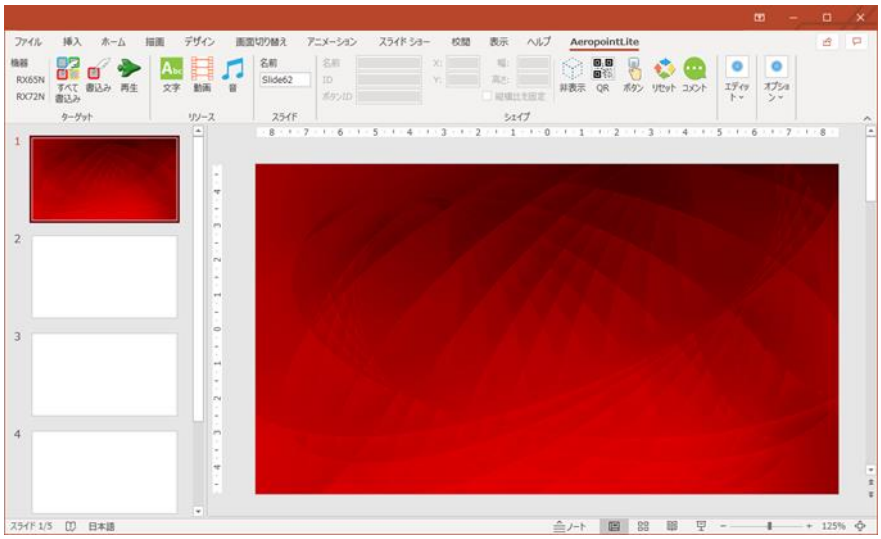


図 111　ただ背景を貼り付ける

いきなりそれらしい高級感がでました。試しに実機で表示しても、何らそん色ありません。なんだかもう勝てる気がします。これを軸に、他のイメージもどんどん形にしていきます。

背景が濃いので、文字は白の方が読みやすいでしょう。

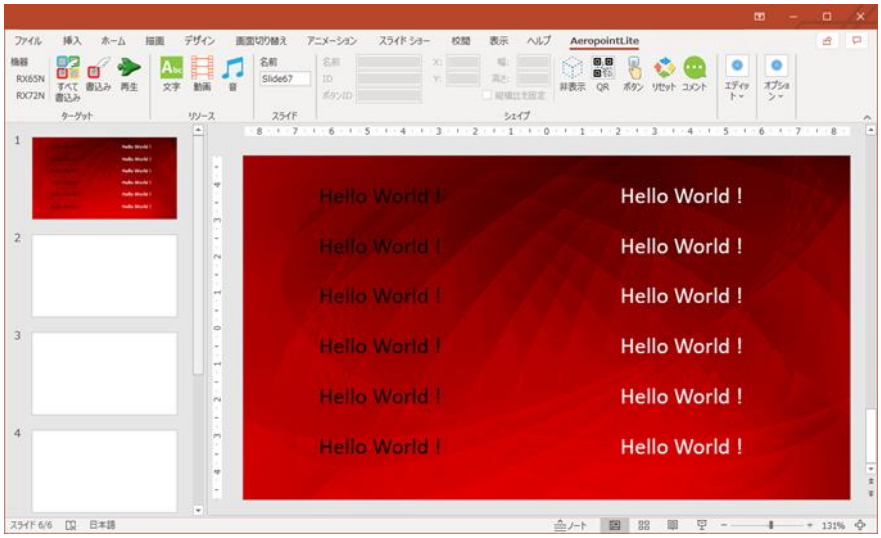


図 112　文字の色を考える

これに仮部品を乗せていきます。先ず分かりやいところで、複数の画面で表示したままになる「運転状況」に関して。画面の上の方がよさそうです。感覚値です。

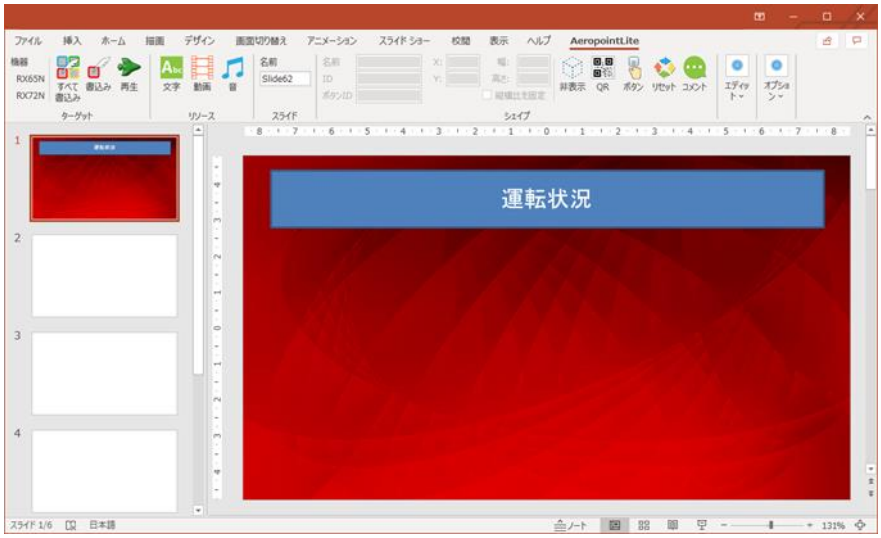


図 113　運転状況の位置

次に暖房・冷房などの「運転切替え」です。大事な機能なので目立った方が良いですが、人によっては季節が変わるときくらいにしか操作しないはずです。目立つように色分けしつつ、右利きの人を想定してあまり触らない左側に配置すると良いかもです。

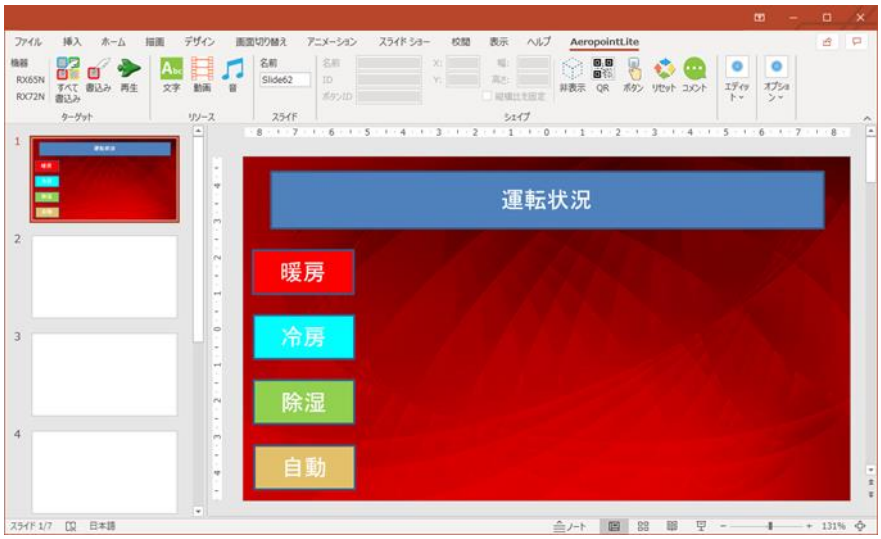


図 114　運転切替えの位置

借りてきた背景の効果で高級感がありましたが、標準の図形をいくつか乗せただけで一気に安っぽくなりました。が、今は仮配置なので気にせず進めます。

複数の画面を表示するためには、切替えボタンが必要です。どんな画面に切り替わるのかを説明したボタンにすると場所を取りそうですし、暖房・冷房のような他に優先的に読ませたいボタンの邪魔になるので、矢印アイコンで表現することにします。順送り、逆送りの 2 つアイコンを置くとして、間違って操作しないようにある程度距離も置きたいです。必然的に、画面の下側でしょうか。

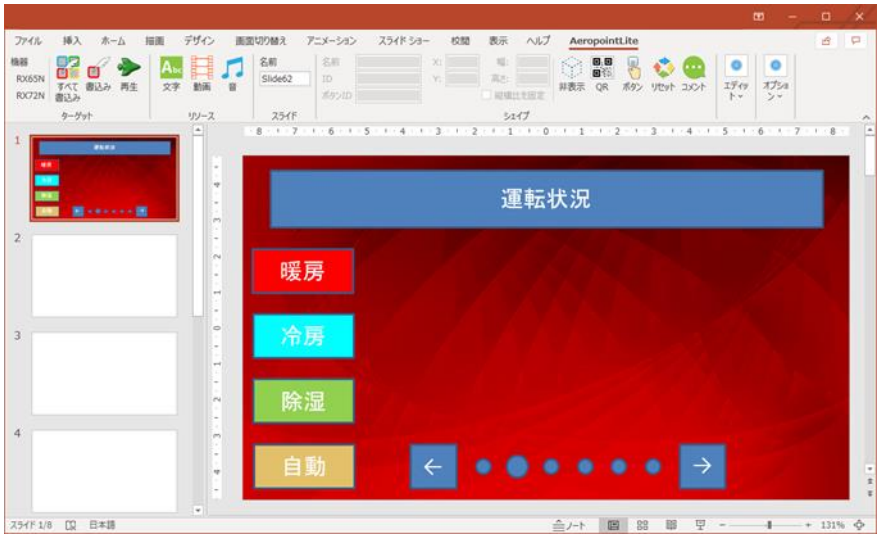


図 115　切替えボタンの位置

現在どの画面を表示しているかを示すために、ツブツブも置いてみました。ツブツブの数は適当です。

全部の画面を作るのは大変なので、一番使いそうな温度を設定する画面を考えます。温度を切替えるボタンが一番使うはずなので、右利きの人を想定して右側に置きます。温度の上げ下げで 2 つのボタンが必要となりますが、距離を置いてしまうと温度の上げ下げの調整がやりづらいと思うので、近付けます。誤操作を防止するという意味では、アイコンを大きめにすれば良いでしょうか。

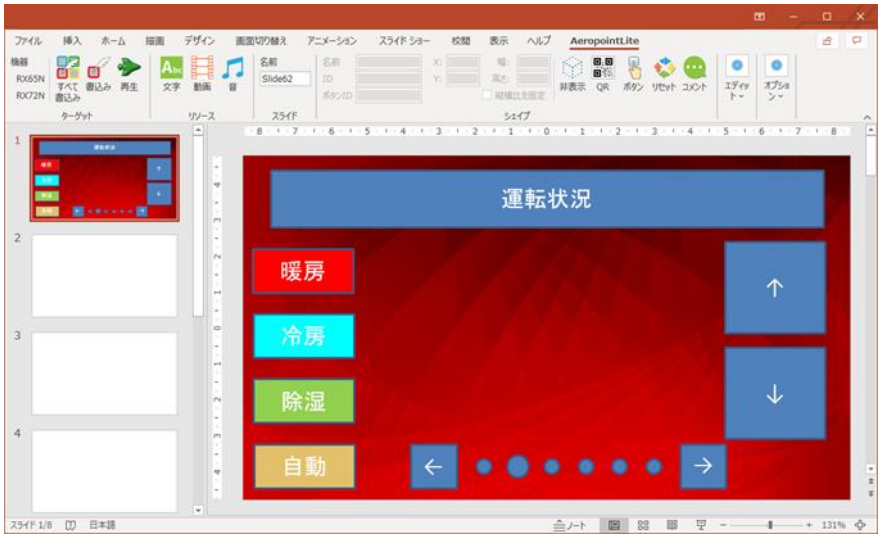


図 116　温度上げ下げの位置

肝心の温度を表示してみます。何の数字だかわかるように、説明も付けます。小数点以下を小さくすると、なんかカッコイイかもです。

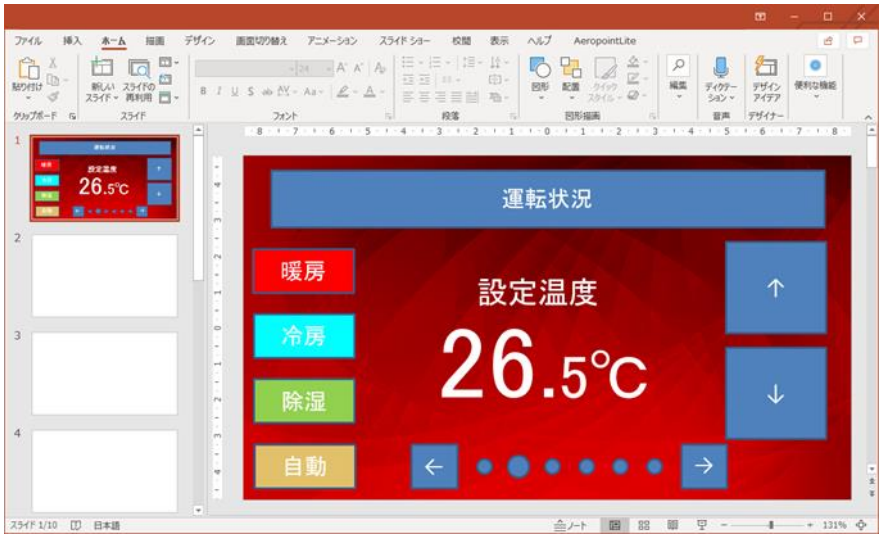


図 117　設定温度の表示

部品もたくさん置いたので、安っぽさも緩和され、まあまあそれらしくなりました。GUI の設計としてはこの辺で良いでしょう。コツとしては、絵的なデザインは程々にして、触ってもらうときにどうあるべきかという機能的なデザインについてある程度考えてやることです。「根本的に最初からやり直しだ！」という悲劇が減ります。

また、Aeropoint を使っているので、こまめに実機で表示してやると良いです。実機の画面のサイズでボタンのサイズが適当か、文字は読み易いかなど、今のうちに整理しておくと後のデザインが楽です。

図 118　実機でサイズ感などを確認

実際に画面に表示して、ボタンになる所を指で触って誤操作せずに済みそうかなど確認します。右矢印と下矢印がちょっと近くで誤爆しそうなので、微調整します。

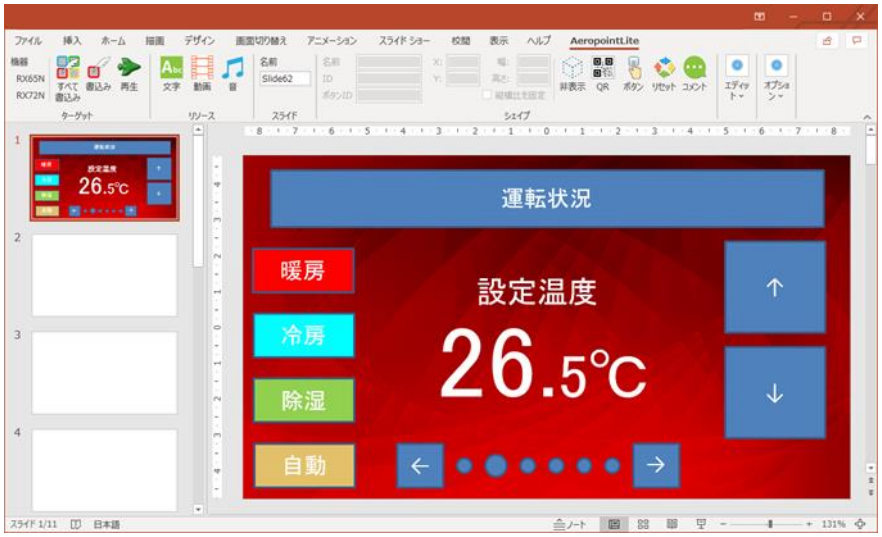


図 119　微調整

こんな感じでしょうか。できました。

6.3 見た目を整える

次は絵的なデザインを進めていきます。機能的なデザインを済ませているので、高級感をだすことに専念します。ずいぶん安っぽくなっていますし。

6.3.1 メリハリをつける

複数の画面に常駐して表示するものを背景の上にそのまま置くと、うっかり別画面で背景を変えるといきなり見づらくなりそうです。また、常駐させるものはスペシャル感が欲しいので「土台」置いてメリハリをつけます。対象は運転状況と運転切替えです。

図 120 土台を置く

土台を置きながら、UI の距離感やサイズ感もどんどん調整します。実機でもこまめに確認します。

やはり濃い背景のときには、黒の土台は鉄板で安定した見た目になります。高級感も少し帰ってきた気がします。

6.3.2 代表的な部品を作る

GUI を制作していると借りてきた素材だけでは大体不足しますし、何か部品のひとつくらい気合を入れて作るとオリジナル感も出てきます。

では何の部品に気合を入れるかなのですが、やはりいちばんタッチすることになるだろう温度調整のボタンでしょう。なんだか図表番号を付けるのも面倒くさくなってきたので、しばらく割愛します。

ここからスタート。

角は取った方がよさそうです。シェイプを新しく作ります。2 番くらいがしっくりきます。

そういえば赤がイメージカラーでした。色を考えるのは思いのほか大変なので、背景で使われている色をコピーして持ってきます。うーん、悩ましいところですが、1 番にしておきます。後で変更できるので、悩むのに時間をかけないようにします。

ここらで配置して眺めてみます。

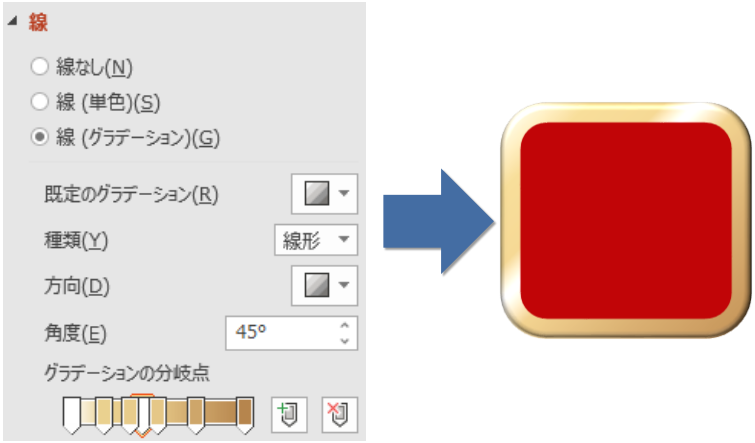
見事に背景に溶け込んでしまい、滅茶苦茶見づらいです。それにここにきて血っぽいイメージが出てきました。アカンですね。ここから挽回するには、だいぶインパクトの強い、脱線したデザインをぶち込む必要がありそうです。大丈夫、まだ慌てる時間帯ではありません。ここでもうひとつのイメージカラーである、金ピカ感の登場でしょう。

実は金色という色は、茶色や黄色や白っぽい色の寄せ集めです。そこに光ったような色分けを加えてやることで、人間の経験則から金色と認識されているだけなのです。必然的に茶色や黄色がたくさん必要なのですが、あれこれ考えるのはしんどいです。さて、困ったときは「検索」です。

私が良く使う、色を勝手に考えてくれるサイト「Picular」で「GOLD」を検索します。

素晴らしいです。

金色を使いすぎると下品になるので、ワンポイントでいきます。ボタンのフチにあしらいます。検索した色を使って、線のプロパティを凄く頑張って指定します。



よし、金ピカ感でました。これの調整だけに 20 分も掛けてしまいました。いざ配置。

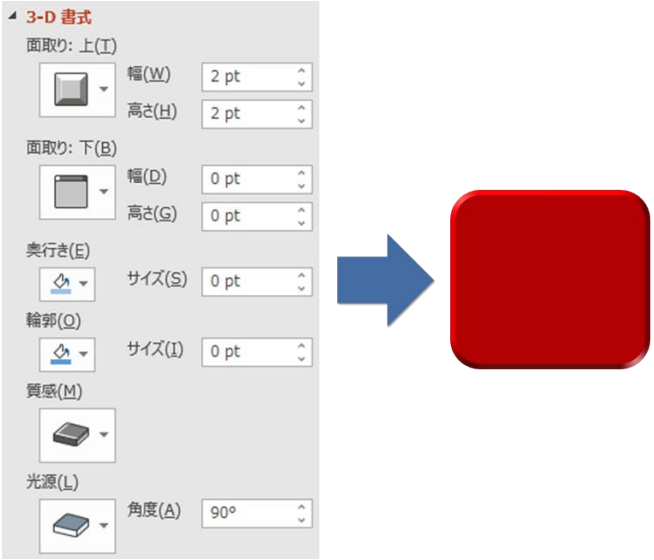
悪くないです。悪くないですが、もうひと押し高級感というか、贅沢感というか、自分でもモヤっとしていてうまく伝えられないのですが、どこか味付けが足りない気がします。

GUI の開発では自分の感覚値はとても大事です。自分が足りないと感じるのなら、きっと本当になにか足りないのです。代表部品なので、ここは時間を掛けてでも頑張る所ですね。

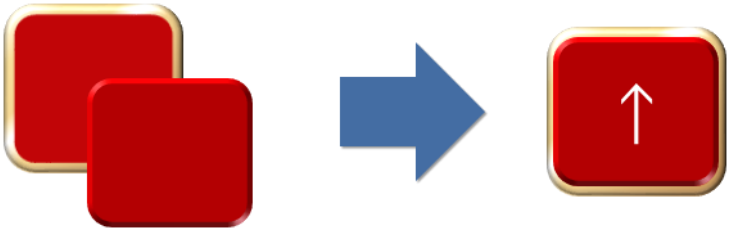
ちなみに制作の途中段階で他人の感覚値を取り入れているとキリがないしイライラするので、そっちは概ね無視して OK です。大体できたあたりで意見をもらうのは有益な場合があります。無視力もとても大事なので、意識して鍛えていきましょう。

3 分くらい熟考したところ、ボタンとしての「押せる感」が足りないような気がします。となると、もっと立体感が欲しいです。しかしひとつのシェイプの設定では、これ以上複雑にすることはできません。そこで、複数のシェイプを重ね合わせていくことにします。

今あるボタンをコピペして、いったん線をなくし、3D 書式で立体感を作ります。

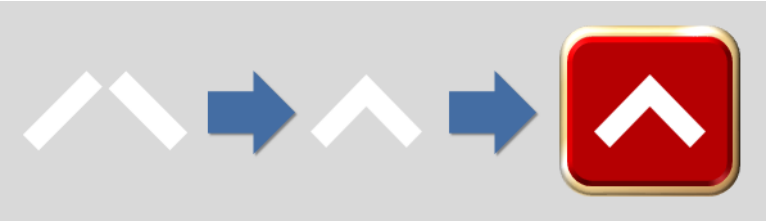


いい感じです。次に、ふたつのシェイプを重ねます。

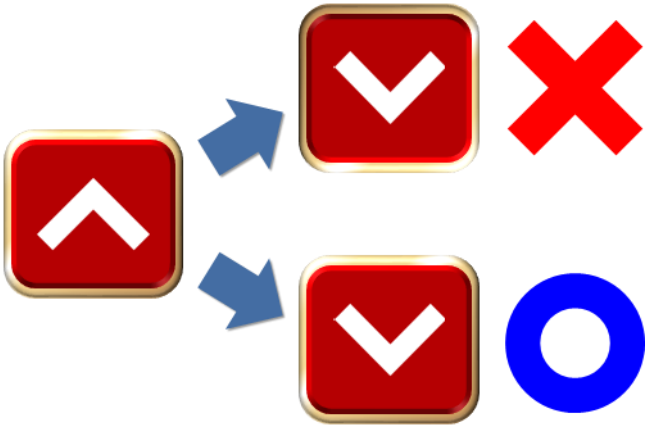


ボタンに高級感が出たので、矢印の安っぽさが際立つようになりました。こういう時は安易に安いほうに流されるのではなく、頑張って難しいほうにアゲていきます。心の強さが必要です。

四角をふたつ使って矢印の代わりになる記号を作ります。



下ボタンを作るにあたって、上ボタンを回転させてしまうと金ピカ感まで逆になってしまいます。中身の矢印だけひっくり返します。



配置してみます。

良い感じに収まりました。代表的な部品として自分的合格です。

6.3.3 コピペしながら増やす

代表的な部品ができたので、他の部品も増やしていきます。画面切り替えのボタンは今作ったボタンの使い回しで良さそうです。コピペして矢印の向きを変えて、小さくします。

ぐぬぅ。このままだと四角ばかりになりそうですし、同じ部品のサイズ違いは手抜き感がありますね。丸にしましょう。四角から丸への変更は「図形の変更」機能で一発です。

ツブツブも丸なので、ボタンも丸いほうが落ち着きますね。

ツブツブを作ります。色は背景からもらいます。ひとつだけ大きなツブツブを置いて、今いる画面を表現します。今いる画面のツブツブは、ボタンの 2 層目をコピペで持ってくればよいでしょう。

ツブツブを仕上げただけなのですが、謎の高級感を醸し出すようになりました。勝てる！

この波に乗って、運転切替えボタンも作ります。コピペで色変更ですね。たくさん色を使うと、カッコイイ気がします。

標準の色だとどうしても安っぽいので、もうひと頑張りします。色を検索して、試行錯誤して柔らかい雰囲気にします。

ここまで印象が変わりますから、やはり色って大切です。

配置してみます。もうエアコンのリモコンだと分かりますね。ボタンが他より少し目立ちますが、運転切替えボタンはいちばん大切なので、それでも良いでしょう。

次に運転状況を作ります。自宅のエアコンのリモコンや検索結果を眺めながら、どういった内容を表示しているともっともらしいのかを考えます。ここまでの編集では、コピペで自然に全体の色味が一致してきたため、全体のイメージとして、少し寂しくなってきています。ここは運転状況を盛り込んでにぎやかにします。とはいえさじ加減なので、個人の勘で良いです。

表示内容はハリボテです。試作の段階で細かいところまで動かして、重箱の隅をつつく系の人から変なダメ出しをもらうくらいなら「試作なので仮です！キリッ！」と言い切ってしまった方が良いです。それでも何か言われたら「貴重なご意見として今後の開発に活かします」とバッサリやりましょう。もちろん、潤沢に時間があれば作り込むのも良いです。

最後に、温度表示をそれらしく調整します。温度そのものは大きく表示されるので視認性が確保できていますが、「設定温度」の文字サイズだと背景に溶け込むかもしれません。すでに土台は他でも使ったので、同じ土台でも少し見え方を工夫します。「設定温度」の文字の裏側に、丸型シェイプを引き延ばして配置して、ぼかしの設定を加えました。また、温度を示す文字に影をつけました。

実機で表示してみます。

問題なさそうです。高級機ならこんな感じのも売ってそう、という気がしてきました。高級感を出していきたいという目的には合っていそうです。

いったん高級機向けのデザインを作ってしまえば、普及機向けは引き算なので、比較的簡単に作れます。

6.4 動きを付ける

ここからは、少し調整しては実機で確認、また少し修正しては実機で確認、と Aeropoint のパワーを活かして作り込んでいきます。

6.4.1 ボタン

Aeropoint ツールキットの「ボタン」機能で、以下のようにボタン ID を割り当てます。

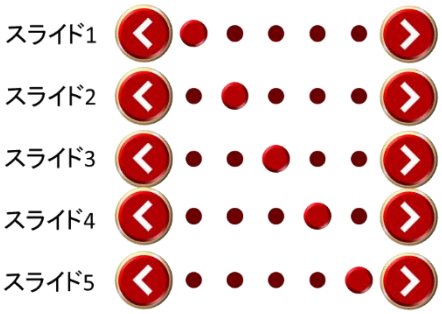
実行してタップすると、Aeropoint ランタイムからコマンドメッセージが送出され、コネクターのログにボタンが押されたことが表示されます。実際の製品開発では、このコマンドにハードウェアが反応するように作っていきます。

```
[14:09:20]>Button press : Slide 0, Button 0
[14:09:21]>Button press : Slide 0, Button 1
[14:09:21]>Button press : Slide 0, Button 2
[14:09:22]>Button press : Slide 0, Button 3
[14:09:22]>Button press : Slide 0, Button 4
[14:09:23]>Button press : Slide 0, Button 5
[14:09:24]>Button press : Slide 0, Button 6
[14:09:25]>Button press : Slide 0, Button 7
```

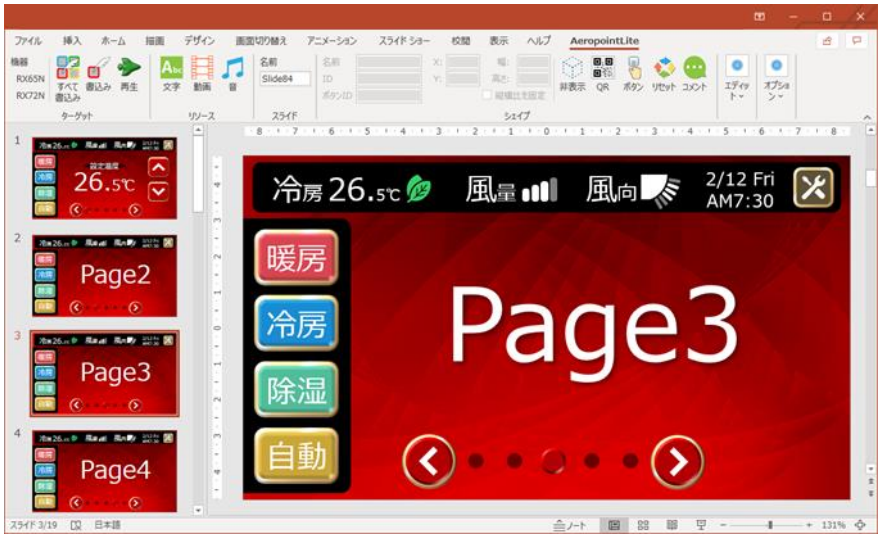
人に見せるデモとして、何らかの HW が反応している様子を見せたほうが良い場合は、ボタンの設定でテスト用の LED と紐づけます。D-Amp 基板にある 5 つの LED を点灯させられます。

6.4.2 画面切り替え

スライド自体をコピペして、5 画面にします。それぞれの画面でツブツブ大の位置を変えます。



新しい画面にはそれとわかるように大きな番号を表示しておきます。



続けて、Aeropoint ツールキットのボタン設定で、左右ボタンで前後のスライドに遷移するように設定を追加します。例えば 1 ページ目の右ボタンでは、2 ページ目に切り替わる設定にします。

実機で動作させて、左右ボタンを押してみます。

6.4.3 数字

テキストの最大サイズは 60 ピクセルという制限がありますので、ここではシェイプの表示 ON/OFF を使って表現します。変化させる数字の数だけシェイプを作り、ID を割り当てておきます。どんな ID にするかは自由なので、分かり易い数字にすると良いです。例えば、今回は小数点を抜きにして考えると 3 桁ですから、10 の位の「1~3」には「100,200,300」を、1 の位の「0~9」には「10~90」を、小数点以下 1 位の「0,5」には「0,5」を割り当てました。

シェイプを全く同じ座標に配置したい場合は、Aeropoint ツールキットのシェイプグループにある X,Y などの座標値に直接値を入力すると便利です。

実機で再生すると、まずはすべてのシェイプが表示されてしまいます。

コネクターでシェイプの ON/OFF コマンドを発行して、不要な数字を消します。

目的の数字が表示できました。

ここまでの試作では部品間の連携を持たせることができませんので、上下のボタンを押しても設定温度が変わることはありません。そこは説明でカバーします。

コマンドに応答する仕組みを作るか、「7 ランタイム SDK」でプログラミングすると連携させられます。
プログラミングでは「5.6.2 エクスポート」で出力したシンボル情報を使用することで、ただの数字では無く意味のある文字で仕組みを作ることができます。

6.5 試作完了、製品へ

6.5.1 たくさん触って微調整

作りながら何度も実機で再生して確認していますが、人間は慣れしまうと細かい点が気にならなくなってしまいます。これに抵抗するために、少し時間をおいてもう一度触る、他の GUI を見たうえで改めて触るなど、工夫をしながら微調整を続けます。

6.5.2 人に見てもらう

やり遂げた、と思ったらそこがスタート地点です。身近な人を皮切りに、いろんな人に見てもらいましょう。ダメ出しされてへコむこともありますが、美味しいもの食べたりして元気を取り戻します。

ダメ出しされる理由で案外多いのは、実際にダメなのではなくて、自分が作ったものに自信が持てずにあやふやな説明をしてしまい、その不安感に相手が共感してしまう、というものがあります。だから安易に安いほうに流されず、自分が作った GUI の設計やデザインに自信を持って説明できるようにしておきます。

自信を持って説明したのにダメ出しされるとさらに大ダメージを受けますが、そんな時はもっと美味しいものを食べたりして復活します。人間はそうやって成長していくのです。きっと。

6.5.3 次のステップへ

コネクターを用いたレビューが好評なら、製品化に向けて次のステップに進みます。

コマンドを用いた製品開発をする場合は、お客様のハードウェアに合わせてコマンドの実装をカスタマイズする必要がありますので、弊社までお問い合わせください。

このまま Envision Kit を用いて検証を進めたい場合は、「7 ランタイム SDK」を参考にしてプログラミングに着手すると良いでしょう。

7 ランタイム SDK

リファレンス SDK では、RX72N の Envision Kit を用いて、Aeropoint のプログラミング体験ができます。他のマイコンをご利用の場合は適宜読み替えて実施してください。

図 121 RX72N Envision Kit

7.1 アプリケーション開発環境

Aeropoint アプリケーションを制作するために、開発環境を整えていきます。

開発環境が必要とするツールキットや統合環境、コンパイラーなどは、供給元によりアップデートされることがあります。本書の説明と異なるバージョンを用いる場合は、供給元から提供される情報も参考にしながら導入を進めてください。

取り急ぎ Aeropoint の仕組みや API の仕様、プログラミング方法などを理解したい場合は、開発環境の構築は後回しにして読み進めてください。

7.1.1 Renesas Flash Programmer

サンプルプログラムの実行バイナリを直接マイコンに書き込むために、PC に Renesas Flash Programmer をインストールしておきます。

- 【無償版】Renesas Flash Programmer V3.06.01
  - https://www.renesas.com/jp/ja/software/D4001217.html

7.1.2 Aeropoint ツールキット

PowerPoint®にアドインする Aeropoint ツールキットが必要です。前章でインストール済みの場合は、追加は必要ありません。
インストールを行う場合は、本キットに付属する以下のツールキットをセットアップしてください。

- Aeropoint ツールキット
  - 32 ビット版：AeropointLiteToolkit_xxxxxxxxRx_CRI_32bit.exe
  - 64 ビット版：AeropointLiteToolkit_xxxxxxxxRx_CRI_64bit.exe

7.1.3 RX72N Envision Kit

リファレンスキットに付属している、ルネサスエレクトロニクス株式会社（以後、ルネサス社）の RX72N Envision Kit をご用意ください。通信販売などでも購入できます。

- RX72N Envision Kit
  - https://www.renesas.com/jp/ja/products/software-tools/boards-and-kits/eval-kits/rx72n-envision-kit.html

7.1.4 e²studio

プログラミングを行う統合環境です。ルネサス社のサイトから e²studio のインストーラーをダウンロードし、案内される手順に従ってセットアップを行います。

- 統合開発環境 e²studio V7.8.0 インストーラー（オフライン用）
  - https://www.renesas.com/jp/ja/software/D4001317.html

セットアップは、おおよそ以下の手順で進みます。

1. ようこそ
   - 「次へ」
2. インストール・フォルダー
   - インストール場所を決めて（デフォルトで可）、「次へ」
3. デバイス・ファミリー
   - 「RX デバイス・サポート」にチェックを入れて、「次へ」
4. 追加コンポーネント
   - 「日本語言語パック」にチェックを入れて、「次へ」
5. コンポーネント
   - 「次へ」
6. 追加ソフトウェア
   - 「Toolchains&Utilities」タブで「Renesas CCRX v3.01.00」にチェックを入れて、「次へ」
   - ツールチェーンがインストール済みの場合は、一覧に表示されません。
   - ツールチェーンは個別にインストールすることもできます。
     - RX コンパイラー CC-RX V3.01.00
       - https://www.renesas.com/jp/ja/software/D4000987.html

この後の手順は、インストーラーの指示に合わせて進めてください。セットアップが終了したら、e²studio の起動確認とツールチェーンの設定をしておきます。

7.1.5 e²studio ツールチェーン

先の項目でインストールした e²studio の起動し、使用するツールチェーンの設定を行います。

1. e²studio の起動
   - インストールした「e²studio」を実行
2. Eclipse ランチャー
   - ワークスペースに任意のパスを指定して、「起動」
3. ToolchainIntegration
   - 「Renesas CCRX - v3.01.00」にチェックを入れて、「登録」
   - このウィンドウは、ツールチェーンをインストールした後の初回起動時にのみ表示されます。
4. e²studio のようこそ画面
   - この画面が表示されたら、インストール&設定完了です。
   - もし「ToolchainIntegration」での設定を行い損ねた場合でも、以下の手順で設定できます。
     - メニューバー「ヘルプ」→「Renesas ツールチェーンの追加」を選択
     - 「Renesas CCRX 」の「v3.01.00」にチェックを入れて「Apply and Close」

「ToolchainIntegration」や「Renesas ツールチェーンの追加」で、「Renesas CCRX v3.01.00」が見当たらない場合、「Renesas CCRX v3.01.00」のインストールが出来ていない可能性があります。

「7.1.4 e²studio」の手順 6 にも記載した通り、ツールチェーンは個別にインストールできます。

以下のページから「RX コンパイラー CC-RX V3.01.00」をダウンロードし、インストールした後に再度上記の手順を試してください。

7.2 システム構成

Aeropoint では、アプリケーション開発向けのライブラリとして、AeropointLite Runtime API（以後、Aeropoint ランタイム）を提供しています。サンプルデータの再生などで利用した Aeropoint プレイヤーも、Aeropoint ランタイムを利用して開発したアプリケーションです。

また、マイコンごとの差異を吸収するためのソフトウェアである Aeropoint Implementation Software（以後、AIS）を、HW のドライバー層の上位に実装しています。AIS は、お客様の製品開発の過程において、カスタマイズされたマイコンや HW 仕様に即した実装を行う場合があります。

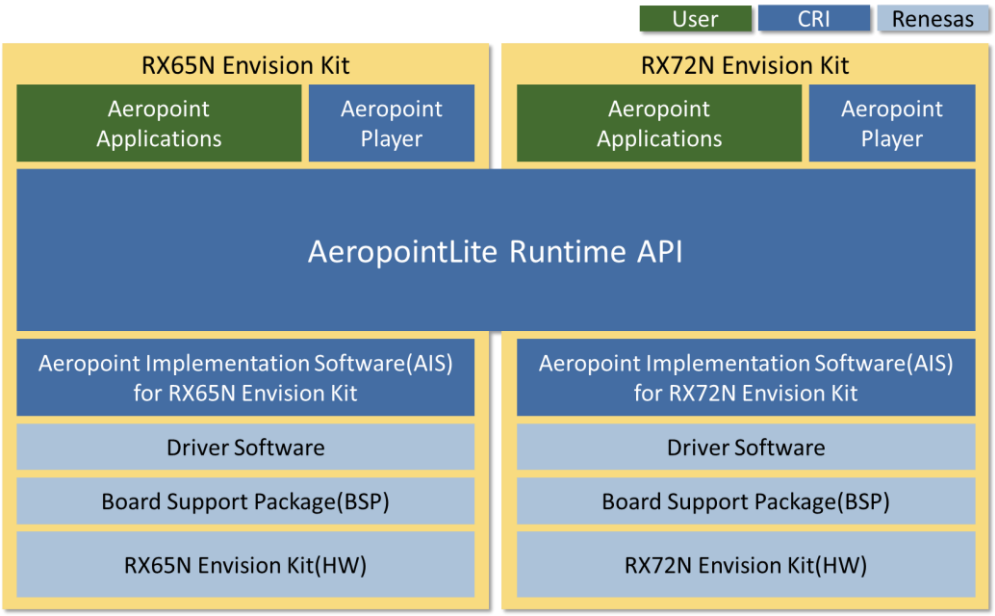


図 122 システム構成

Aeropoint ランタイムを用いたアプリケーション開発では、「プレイヤーAPI」と「ランタイムエンジン API」の２種類の API を利用します。

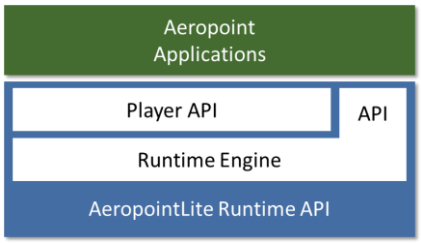


図 123 ランタイム API の構成

プレイヤーAPI は、実機の制御に必要となる基本プログラムをすべて含んでいます。そのため、アプリケーションは初期化や終了、主処理呼び出しなど、わずか数十行のプログラミングで完成します。

7.3 基本フロー

PowerPoint®で作った UI をお客様が操作するまでには、おおよそ以下のようなフローが働きます。

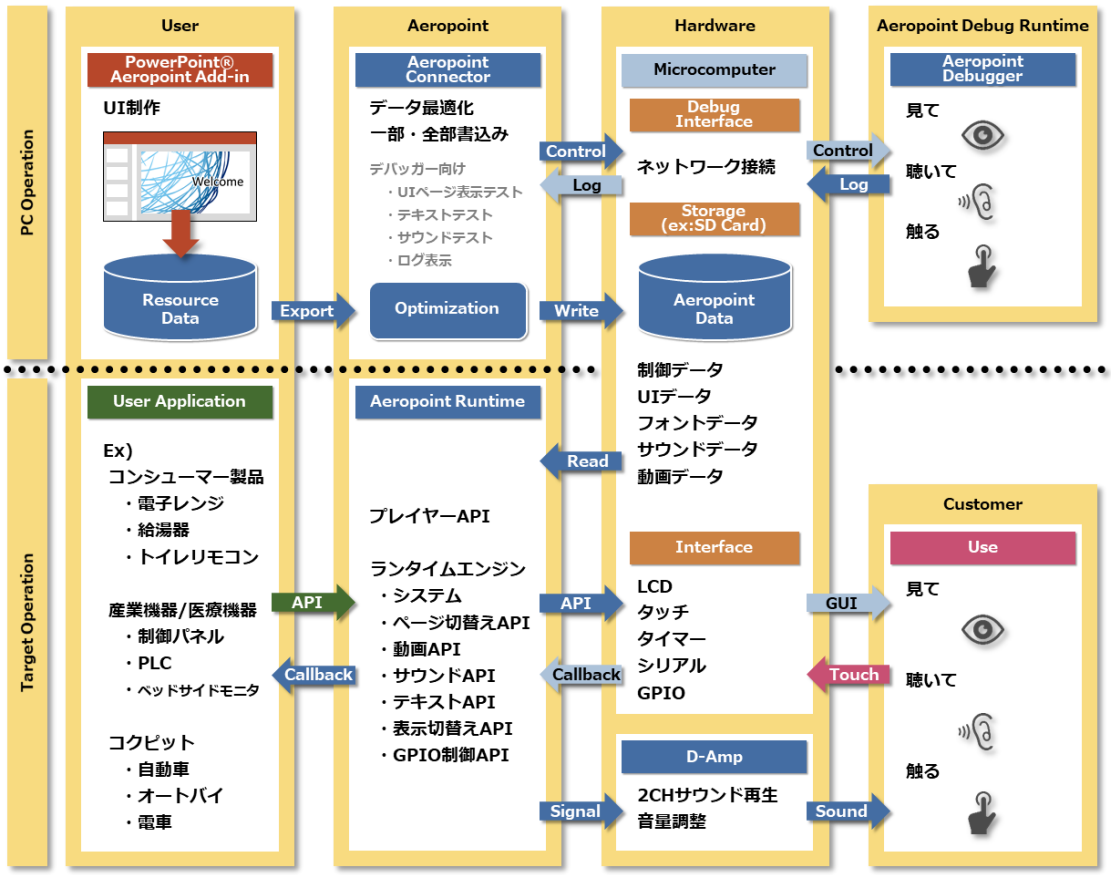


図 124 基本フロー

PC では、PowerPoint®を用いたアプリケーションのデザインや、基本的な振る舞いを制作します。アプリケーション制作の途中であっても、Aeropoint プレイヤーを用いて、ユーザーと同じように「見て・聴いて・触る」プレビューができます。

実機では、プレイヤーAPI を用いて基本アプリケーションを制作します。アプリケーションごとの特長の追加や、アプリケーションに固有の複雑な仕様を実現する場合は、ランタイムエンジン内の各種 API を用いてカスタマイズをします。

7.4 Aeropoint ランタイム

実機では UI やサウンド、動画などのたくさんの機能が動作しており、それらは全て Aeropoint ランタイム内の「ランタイムエンジン」が管理して実行します。ランタイムエンジンは、例えばゲーム開発ならば「ゲームエンジン」に相当するものです。ゲームエンジンと同じように外部ツール（ここでは PowerPoint®）を利用して制作したデータを与えると、コンテンツに必要となる基本的な動作を受け持ってくれます。

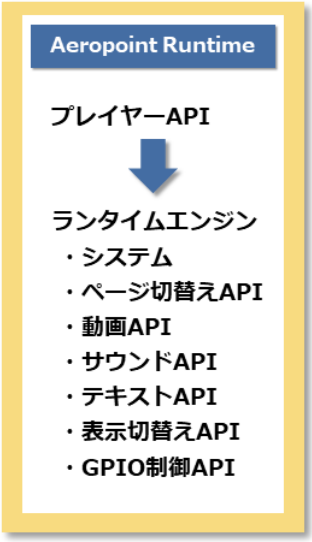


図 125　プレイヤーAPI とランタイムエンジン

しかし、ランタイムエンジンの API は多岐にわたるため、そのまま利用すると実装に必要なコード量が多くなりがちです。そこで Aeropoint ランタイムでは、ランタイムエンジンの動作を取りまとめてアプリケーションの基本的な動作を保証する「プレイヤーAPI」を提供しています。プレイヤーAPI は、ユーザーが Aeropoint アプリケーションを制作するにあたって必ず利用する基本的な仕組みです。

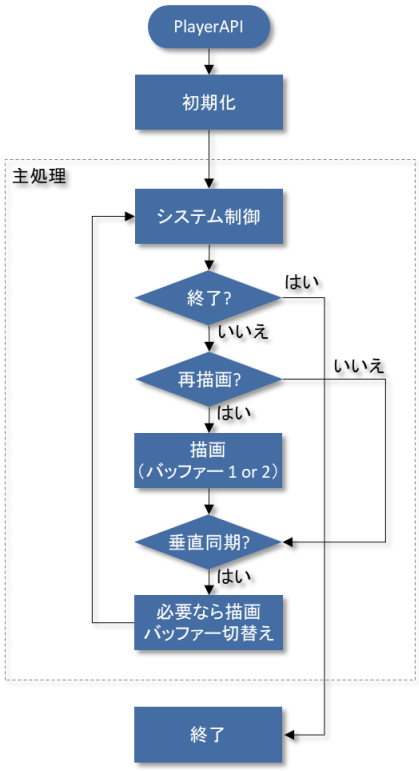
7.4.1 最小のアプリケーション

最小のアプリケーションは、プレイヤーAPI の「初期化・主処理・終了」の 3 つの手続きで完結します。このうち「主処理」関数は、UI のアニメーションや動画の表示などの繰り返しが必要な処理を行うため、回数制限を付けずに呼び出し続けます。主処理を繰り返し呼び出すプログラムを「メインループ」と呼びます。

```
1  void main(void)
2  {
3      CriBool flag = CRI_TRUE;
4
5      /* Initialize */
6      CriAeroPlayer_Initialize();
7
8      /* Main loop */
9      while( flag )
10     {
11         /* Main */
12         flag = CriAeroPlayer_Main();
13     }
14
15     /* Finalize */
16     CriAeroPlayer_Finalize();
17
18     return;
19 }
```

図 126　最小のアプリケーション

プレイヤーAPI を用いてメインループを記述すると、Aeropoint のランタイムエンジンが動作します。最小のアプリケーションは以下のようなフローで動作します。



このプログラムだけでも、PowerPoint®で編集されたデータをストレージ（例えば SD カード）から読み出し、UI 表示やタッチインターフェース、サウンド再生するなどして、「見て・聴いて・触る」アプリケーションとなります。

必要に応じて発生する画面への描画は、2 つあるバッファーのうち、非表示状態にある背面バッファーに行います。描画を終了したら、画面の切替えタイミングである垂直同期信号の到来を待ち、表示中の前面バッファーと背面バッファーを切替えます。これにより、ちらつきのない画面切り替えを実現しています。

図 127　最小のアプリケーションのフロー

実際に、最小のアプリケーションを実行してみます。

1. 実機の準備

基板を以下のように見て、SW1 / SW3 の位置を合わせます。また、SW1 に隣接する ECN1 端子と PC を MicroUSB で接続します。

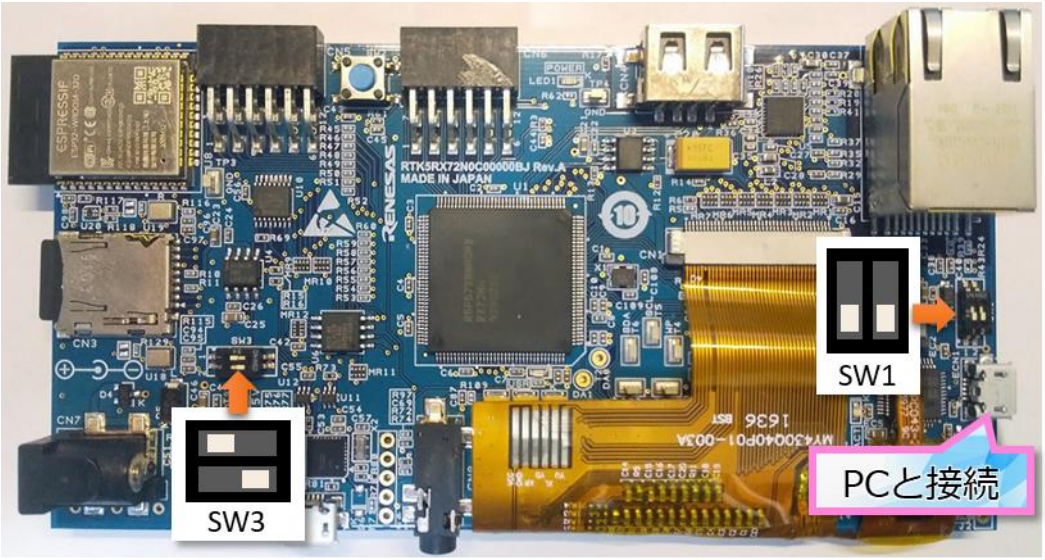


図 128　ディップスイッチの確認

2. e²studio の起動

e²studio を起動します。

ワークスペースは任意のパスで構いません。

しかし、この後にインポートするプロジェクトなどを含めた合計のパス長が 260 文字を超えるとビルドにてエラーとなりますのでご注意ください。

3. インポートウィンドウの起動

プロジェクト・エクスプローラーを右クリックし、「インポート」を選択します。

4. 既存プロジェクトのインポートを選択

インポートウィンドウで「一般」→「既存プロジェクトをワークスペースへ」を選択します。

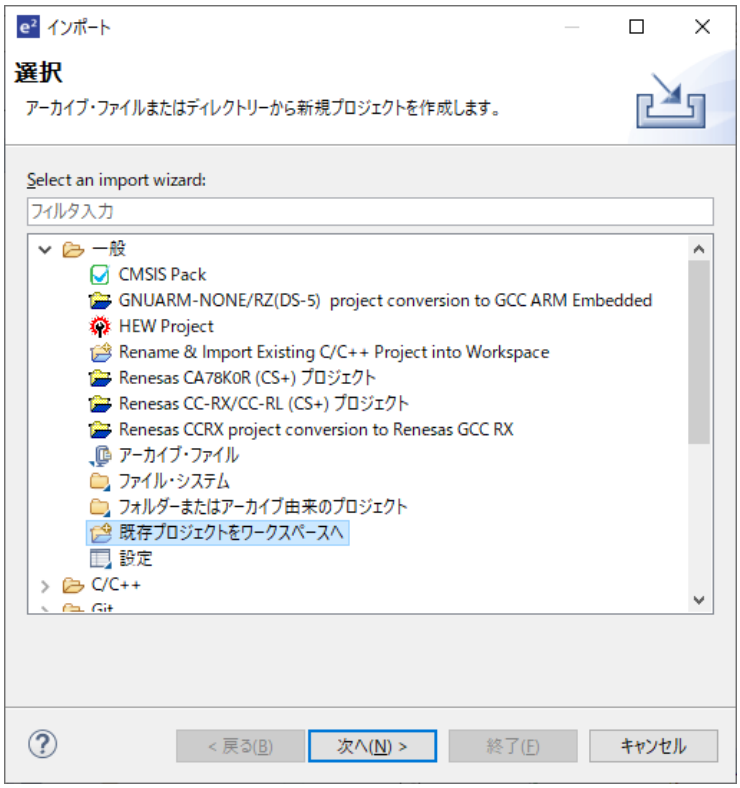


図 129 既存プロジェクトのインポートを選択

5. ライブラリ プロジェクトの ZIP 形式ファイルを選択

ラジオボタンの「アーカイブ・ファイルの選択」を選択し、パッケージ内にあるライブラリ プロジェクトの ZIP 形式ファイルを選択し「終了」を押します。

ライブラリ プロジェクトの ZIP はパッケージの以下のパスにあります。

・Runtime/AeropointRX72N_Library.zip

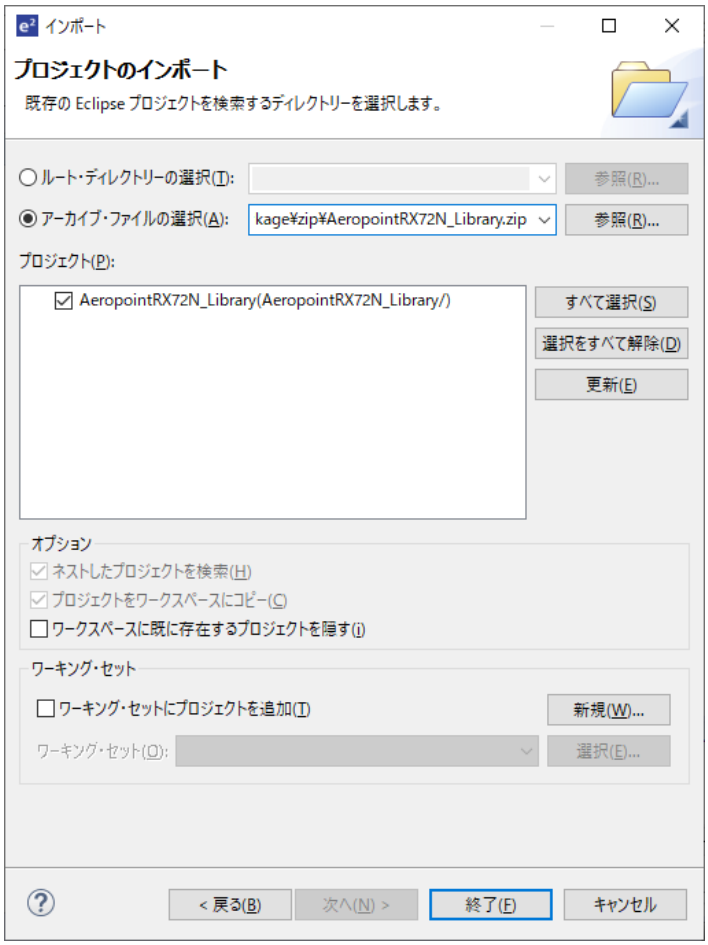


図 130　ライブラリプロジェクトの ZIP 形式ファイルを選択

6. プレイヤー プロジェクトもインポート

先程のライブラリのプロジェクトと同様の手順で、プレイヤー プロジェクト ZIP 形式ファイルもインポートして下さい。

プレイヤー プロジェクトの ZIP はパッケージの以下のパスにあります。

・Runtime/AeropointRX72N_Player.zip

7. プロジェクトのインポートを確認

プロジェクト・エクスプローラーにライブラリプロジェクト「AeropointRX72N_Library」とプレイヤープロジェクト「AeropointRX72N_Player」の 2 つのプロジェクトが表示されていることを確認します。

8. プレイヤープロジェクトをビルド

プレイヤープロジェクトを選択し、右クリックから「プロジェクトのビルド」を選択します。

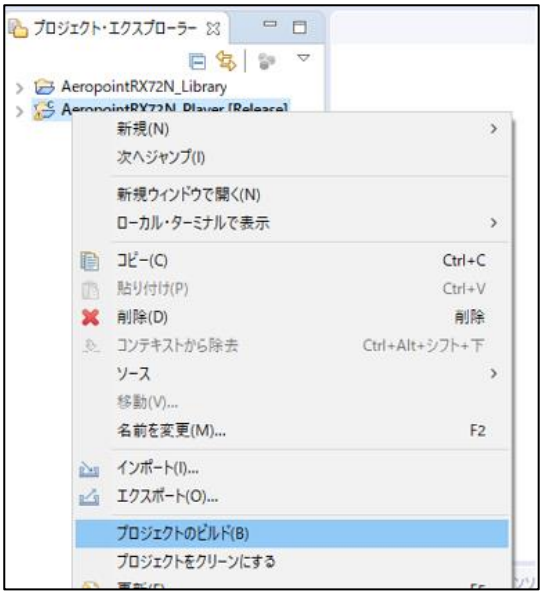


図 131　プレイヤー プロジェクトのビルド

「[ERROR] No toolchain set or toolchain not integrated.」というエラーが出てビルドに失敗する場合、ツールチェーンの設定ができていない可能性があります。「7.1.5 e²studio ツールチェーン」を参考にして、ツールチェーンの設定を行ってみてください。

9. 構成の確認と実行

構成が「AeropointRX72N_Player_Release」になっていることを確認し、デバッグ実行ボタンを押します。

構成が異なる場合、構成の右側にある下矢印から「AeropointRX72N_Player_Release」を選択してください。

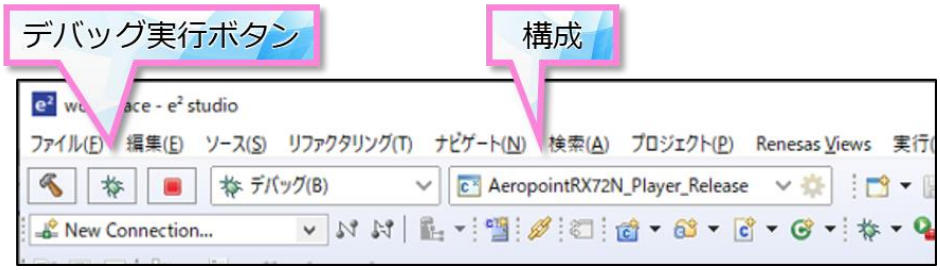


図 132 構成とデバッグ実行ボタン

10. パースペクティブ切り替えの確認

デバッグ画面への移行途中で「パースペクティブ切り替えの確認」が表示されますので、「はい」を選択します。

11. プログラムの実行

実行画面に移ります。

実行直後、プログラムは最初の処理で待機している状態ですので、そのまま「再開」ボタンを 2 回押してください。

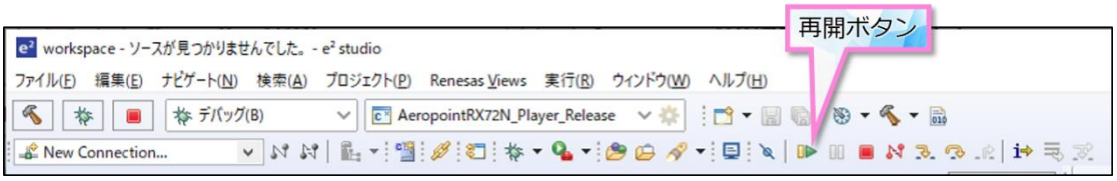


図 133　実行画面の再開ボタン

12. スタートアップ画面の表示

実機の画面にスタートアップ画面が表示されます。

図 134　スタートアップ画面

7.4.2 スライドの表示

最小のアプリケーションに 1 行だけプログラムを追加して、PowerPoint®で作成したスライドを表示してみましょう。

> ⚠ リファレンスキットの付属している、データイメージが書き込み済みの SD カードを、実機の microSD スロットに挿入しておく必要があります。SD カードにデータイメージが書き込まれていない場合は、「11 データイメージの書き込み」を参考にして、準備してください。

e²studio の「プロジェクト・エクスプローラー」からメインプログラムを選び、ダブルクリックします。

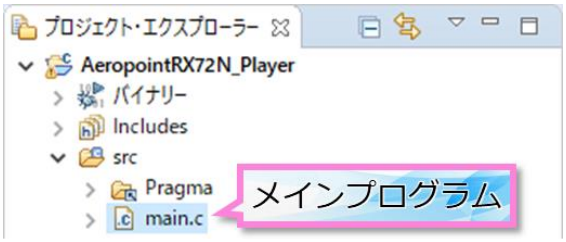


図 135　メインプログラムを開く

エディットウィンドウにメインプログラムが開きます。

```
main.c
1   #include "cri_aerolite_player.h"
2
3   void main(void)
4   {
5       CriBool flag = CRI_TRUE;
6
7       /* Initialize */
8       criAeroLitePlayer_Initialize();
9
10      /* Main loop */
11      while( flag )
12      {
13          /* Main */
14          flag = criAeroLitePlayer_Main();
15      }
16
17      /* Finalize */
18      criAeroLitePlayer_Finalize();
19
20      return;
21  }
22
```

図 136　メインプログラム

メインループが始まる直前に、1 番目スライドを表示する命令として「CriAeroPlayer_ReadSlide(1)」を追記します。以下の赤文字の部分です。

```
1  void main(void)
2  {
3      CriBool flag = CRI_TRUE;
4
5      /* Initialize */
6      CriAeroPlayer_Initialize();
7
8      /* Read Slide 1 */
9      CriAeroPlayer_ReadSlide(1);
10
11     /* Main loop */
12     while( flag )
13     {
14         /* Main */
15         flag = CriAeroPlayer_Main();
16     }
17
18     /* Finalize */
19     CriAeroPlayer_Finalize();
20
21     return;
22 }
```

スライド 1 を表示

図 137　スライドの表示の追記

プログラムを保存したら、再びビルドし実行します。以下のような「Welcome 画面」が表示されたら成功です。

図 138　スライドの表示画面

動作確認ができたら、変更したプログラムは元に戻しておきます。

7.4.3 コールバック

コールバックは、実機でボタンがタップされるなどのイベントが発生した場合に、その内容をユーザー関数に伝える働きをします。

ボタンをタップする場合の働きとしては、スライドの変更やサウンド・動画の再生などを Aeropoint ツールキットで指定できます。それに加えてコールバックに応答することで、それらの働きを変更したり、ユーザー独自のハードウェアを駆動したりするなど、アプリケーションの動作を柔軟に拡張できます。

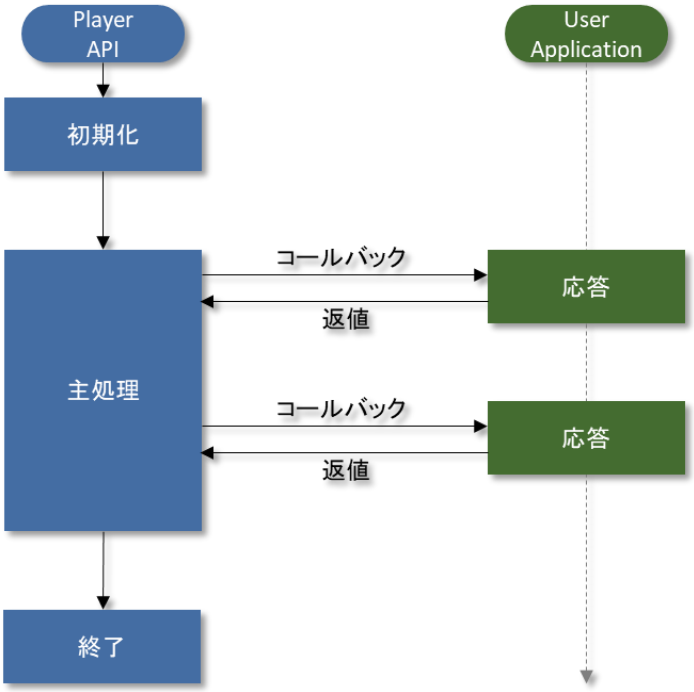


図 139　コールバックのフロー概要

コールバックのサンプルを用いて、その働きを見てみましょう。以下のプロジェクトと GUI を利用します。

- プロジェクト
  - Sample/Project/AeropointRX72N_Sample_Callback.zip
- GUI
  - Sample/GUI/ Sample_Callback.pptx

プロジェクトは e²studio にインポートします。GUI は「5.1.1 全て書込み」を参考にして、SD カードに書き込んでおきます。

main.c を開くと、コールバック関数が記述されています。関数の使用などの詳細はリファレンスを参考にしてください。ここでは、指定されたサウンドを鳴らそうとする動作をキャンセルして、ランダムでサウンドを鳴らすようにしています。

```
CriAeroResponsed mycallback( CriAeroMessage Message, CriSint32 Param, CriSint32 SoundId,
 CriSint32 MovieId )
{
    CriAlpResponsed res = CRI_AERO_RESPONSED_NONE;

    /* Button was pressed */
    if( Message == CRI_AERO_MESSAGE_BUTTON_PRESS ){
        /* SoundId is valid */
        if( SoundId >= 0 ){
            /* Random play */
            CriAeroSound_PlaySound( rand() % 10 );

            /* Cancel default sound control */
            res |= CRI_AERO_RESPONSED_PLAY_SOUND;
        }
    }

    return res;
}
```

ボタンがタップされ

サウンドを鳴らす予定なら

ランダムで鳴らす

デフォルトの動作はキャンセル

図 140 コールバック関数の例

コールバック関数は、メインループを開始する前に Aeropoint ランタイムに登録することで、イベント発生時に呼び出されるようになります。

```
1  void main(void)
2  {
3      CriBool flag = CRI_TRUE;
4
5      /* Initialize */
6      CriAeroPlayer_Initialize();
7
8      /* Set UserCallback */
9      CriAeroPlayer_SetCallback(mycallback);
10
11     /* Main loop */
12     while( flag )
13     {
14         /* Main */
15         flag = CriAeroPlayer_Main();
16     }
17
18     /* Finalize */
19     CriAeroPlayer_Finalize();
20
21     return;
22 }
```

コールバックを登録

図 141 コールバック関数の登録

このプログラムをビルドして実行すると、最小のプログラムで加えた 1 番目のスライドを表示するコードが入っていないため、スタートアップ画面が表示されます。Aeropoint アドインの再生ボタンを押すか、実機 SW2 スイッチを押すと、コールバック GUI を表示します。

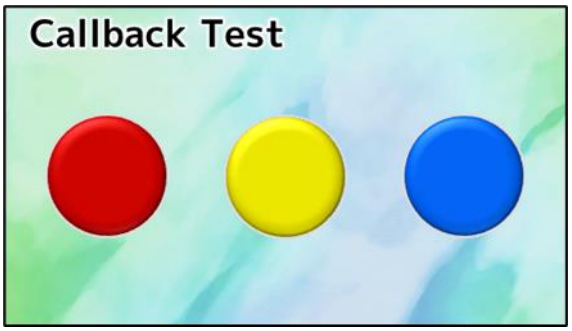


図 142 コールバック動作確認画面

ボタンをタップすると、本来なら Aeropoint ツールキットで設定したサウンドが鳴るはずですが、コールバック関数が働いてランダムで音が鳴ります。

7.5 Aeropoint プログラミングリファレンス

Aeropoint アプリケーションを制作するにあたって必要となるシンボルや構造体、関数などを解説します。

7.5.1 シンボル、構造体、列挙体

色指定列挙体です。フォントなど、色を指定する場合に利用します。

```
/* カラー指定 */
enum {
        CRI_AERO_COLOR_PUNCHTHROUGH = 0,    /* パンチスルー(背景色) */
        CRI_AERO_COLOR_RESERVED_1,          /* 予約 1              */
        CRI_AERO_COLOR_RESERVED_2,          /* 予約 2              */
        CRI_AERO_COLOR_BLACK,               /* 黒                  */
        CRI_AERO_COLOR_WHITE,               /* 白                  */
        CRI_AERO_COLOR_RED,                 /* 赤                  */
        CRI_AERO_COLOR_GREEN,               /* 緑                  */
        CRI_AERO_COLOR_BLUE,                /* 青                  */
        CRI_AERO_COLOR_YELLOW,              /* 黄                  */
        CRI_AERO_COLOR_CYAN,                /* シアン              */
        CRI_AERO_COLOR_MAGENTA,             /* マゼンダ            */

        CRI_AERO_COLOR_END
};

typedef CriUint32 CriAeroColorID;
```

テスト用 LED 指定列挙体です。LED を指定する場合に利用します。

```
/* テスト IO LED 色指定 */
enum
{
        CRI_AERO_TESTIO_LED_GREEN = 0,
        CRI_AERO_TESTIO_LED_BLUE,
        CRI_AERO_TESTIO_LED_YELLOW,
        CRI_AERO_TESTIO_LED_RED,
        CRI_AERO_TESTIO_LED_WHITE,

        CRI_AERO_TESTIO_LED_NUM
};

typedef CriUint32 CriAeroTestIOLedColor;
```

7.5.2 UART コマンド

UART コマンドの一覧です。コマンドやパラメータの間は半角スペースで区切ります。コマンドは¥n で終端します。

UART を行うためのシリアル通信の接続方法については「8 シリアル通信の接続方法」をご参照ください。

項目	コマンド	パラメータ
指定したスライドを表示する	slideno <NO>	NO：スライド番号(1~)
指定したシェイプを表示する	shapeon <ID>	ID：シェイプ ID(0~)
指定したシェイプを非表示にする	shapeoff <ID>	ID：シェイプ ID(0~)
テキストを表示する	textpring <ID> <X> <Y> <fontID> <ColorID> <string>	ID：テキスト ID(0~7) X, Y：表示する座標 fontID：フォント ID(0～15) ColorID：文字色(0～7) string：任意の文字列(UTF-8)
テキストをすべて消去する	textclear	-
タイトルを表示する	showtitle	-
サウンドを再生する	soundplay <ID>	ID：サウンド ID(0~31)
サウンドを停止する	soundstop	-
BGM の音量を変更する	bgmvolume <Vol>	Vol：BGM の音量(0～255)
SE の音量を変更する	sevolume <Vol>	Vol：SE の音量(0～255)
動画を再生する	movieplay <ID>	ID：動画 ID(0~7)
動画を停止する	moviestop	-
ログを出力する	log <string>	string：任意の文字列(UTF-8)
画面を強制的に再描画する	scrupdate	-

7.5.3 プレイヤーAPI

プレイヤーAPI は、すべてのユーザーアプリケーションで利用する基本的な仕組みを提供します。

以下のヘッダーファイルをインクルードします。

```
1 #include "cri_aero_player.h"
```

関数の一覧です。

関数	内容
CriAeroPlayer_Initialize()	プレイヤーを初期化します。
CriAeroPlayer_Main()	繰り返し呼び出す主処理です。
CriAeroPlayer_Finalize()	プレイヤーを終了します。
CriAeroPlayer_Callback()	コールバック関数です。
CriAeroPlayer_SetCallback()	コールバック関数の登録と解除です。

CriAeroPlayer_Initialize()

概要

プレイヤーを初期化します。

書式

void CRIAPI CriAeroPlayer_Initialize();

引数

なし。

戻り値

なし。

説明

プレイヤーを初期化し、Aeropoint ランタイムシステムを起動します。

使用例

```
void main(void)
{
    CriBool flag = CRI_TRUE;

    /* Initialize */
    CriAeroPlayer_Initialize();

    /* Main loop */
    while( flag )
    {
        /* Main */
        flag = CriAeroPlayer_Main();
    }

    /* Finalize */
    CriAeroPlayer_Finalize();

    return;
}
```

注意

再呼び出しを行うと、何もせずに戻ります。

CriAeroPlayer_Main()

概要

繰り返し呼び出す主処理です。

書式

CriBool CRIAPI CriAeroPlayer_Main();

引数

なし。

戻り値

CRI_TRUE : 処理継続。
CRI_FALSE : 処理終了。

説明

UI の表示、サウンドや動画の再生、タッチや GPIO の制御など、Aeropoint の動作に必要なすべての処理を実施します。定期的に呼び続ける必要があります。

戻り値が CRI_FALSE の場合はメインループを抜けて、プレイヤーの終了関数を呼び出してください。ただし、アプリケーションの実行継続が困難なレベルの重大なランタイムエラーを除き、CRI_FALSE が戻ることはありません。

戻り値が CRI_TRUE である場合も、メインループ抜けて終了関数を呼び出すとプレイヤーは終了します。

使用例

```
1  void main(void)
2  {
3      CriBool flag = CRI_TRUE;
4
5      /* Initialize */
6      CriAeroPlayer_Initialize();
7
8      /* Main loop */
9      while( flag )
10     {
11         /* Main */
12         flag = CriAeroPlayer_Main();
13     }
14
15     /* Finalize */
16     CriAeroPlayer_Finalize();
17
18     return;
19 }
```

CriAeroPlayer_Finalize()

概要

プレイヤーを終了します。

書式

void CRIAPI CriAeroPlayer_Finalize();

引数

なし。

戻り値

なし。

説明

プレイヤーを終了し、Aeropoint ランタイムシステムを停止させます。

使用例

```
void main(void)
{
    CriBool flag = CRI_TRUE;

    /* Initialize */
    CriAeroPlayer_Initialize();

    /* Main loop */
    while( flag )
    {
        /* Main */
        flag = CriAeroPlayer_Main();
    }

    /* Finalize */
    CriAeroPlayer_Finalize();

    return;
}
```

注意

再呼び出しを行うと、何もせずに戻ります。

CriAeroPlayer_Callback()

概要

メッセージ発生時に呼び出されるコールバック関数（の型）です。

書式

typedef CriAeroResponsed (*CriAeroPlayer_Callback)(CriAeroMessage Message,
CriSint32 Param, CriSint32 SoundId, CriSint32 MovieId);

型

メッセージ（コールバック関数の第一引数）

```
1 typedef enum {
2     CRI_AERO_MESSAGE_SLIDE_PRE_CHANGE = 0,     /* この後表示スライドが変更される */
3     CRI_AERO_MESSAGE_SLIDE_POST_CHANGE,                /* 表示スライドが変更された */
4
5     CRI_AERO_MESSAGE_BUTTON_PRESS,                     /* ボタン操作が成立した */
6 } CriAeroMessage;
```

応答状況（コールバック関数の返値）

```
1 typedef enum {
2     CRI_AERO_RESPONSED_NONE         = 0,          /* 操作なし */
3
4     CRI_AERO_RESPONSED_SLIDE_PRE_CHANGE   = (1 << 0),  /* 表示スライドが変更される */
5     CRI_AERO_RESPONSED_SLIDE_POST_CHANGE  = (1 << 1),  /* 表示スライドが変更された */
6     CRI_AERO_RESPONSED_BUTTON_PRESS       = (1 << 2),  /* ボタン操作 */
7     CRI_AERO_RESPONSED_PLAY_SOUND         = (1 << 3),  /* サウンド再生 */
8     CRI_AERO_RESPONSED_PLAY_MOVIE         = (1 << 4),  /* 動画再生 */
9 } CriAeroResponsed;
```

引数

Message　：　メッセージの種類。

Param　：　Message が CRI_AERO_MESSAGE_SLIDE_PRE_CHANGE のとき、
→ 次に表示するべきスライド番号。
Message が CRI_AERO_MESSAGE_SLIDE_POST_CHANGE のとき、
→ 表示されたスライド番号。
Message が CRI_AERO_MESSAGE_BUTTON_PRESS のとき、
→ ボタン番号。

SoundId　：　サウンド ID。

MovieId　：　動画 ID。

戻り値

実施した処理の内容に応じて、CriAeroResponsed 内の列挙子を OR 条件で重ねて戻します。有効な処理を行わなかった場合は、CRI_AERO_RESPONSED_NONE を戻します。

説明

この関数はユーザーが用意し、あらかじめプレイヤーに登録しておきます。プレイヤーは主処理の過程で、必要に応じてコールバックを実施します。

使用例

```
CriAeroResponsed mycallback( CriAeroMessage Message, CriSint32 Param, CriSint32 SoundId, CriSint32 MovieId )
{
    CriAeroResponsed res = CRI_AERO_RESPONSED_NONE;

    /* Button was pressed */
    if( Message == CRI_AERO_MESSAGE_BUTTON_PRESS ){
        /* SoundId is valid */
        if( SoundId >= 0 ){
            /* Random play */
            CriAeroSound_PlaySound( rand() % 10 );

            /* Cancel default sound control */
            res |= CRI_AERO_RESPONSED_PLAY_SOUND;
        }
    }

    return res;
}
```

CriAeroPlayer_SetCallback()

概要

プレイヤーにコールバック関数を登録します。

書式

void CRIAPI CriAeroPlayer_SetCallback(const CriAeroPlayer_Callback callback);

引数

callback : ユーザーコールバック関数。
NULL を渡すと登録解除。

戻り値

なし。

説明

コールバック関数の登録、もしくは登録解除を行います。

使用例

```
void main(void)
{
    CriAeroPlayer_Initialize();
    CriAeroPlayer_SetCallback( mycallback );
       :
       :
```

7.5.4 スライド切替え API

SD カードから任意のスライドを読込んで表示し、実行状態にします。

関数の一覧です。

関数	内容
CriAeroPlayer_ReadSlide()	スライドを指定して表示します。

CriAeroPlayer_ReadSlide()

概要

スライドを指定して表示します。

書式

CriSint32 CRIAPI CriAeroPlayer_ReadSlide(CriSint32 SlideNo);

引数

SlideNo　:　表示するスライドを 1~の番号で指定します。
-1 でスタートアップ画面を表示します。

戻り値

SlideNo　:　表示に成功すると、引数と同じ値を返します。
-1　:　表示失敗。

説明

引数で指定されたスライドを表示します。表示に失敗すると、スタートアップ画面を表示します。

図 143　スタートアップ画面

7.5.5 動画 API

動画 API は、PowerPoint®で設定した最大 8 個の動画を再生します。

関数の一覧です。

関数	内容
CriAeroMovie_Play()	動画を再生します。
CriAeroMovie_Stop()	動画を停止します。
CriAeroMovie_IsPlay()	動画が再生中かどうか調べます。

CriAeroMovie_Play()

概要

動画を再生します。

書式

CriBool CRIAPI CriAeroMovie_Play(CriUint32 MovieId);

引数

MovieId : 再生する動画を 0~7 の番号で指定します。

戻り値

CRI_TRUE : 再生成功。

CRI_FALSE : 再生失敗。

説明

引数で指定された動画の再生を開始します。

CriAeroMovie_Stop()

概要

動画を停止します。

書式

void CRIAPI CriAeroMovie_Stop();

引数

なし。

戻り値

なし。

説明

再生中の動画があれば停止します。

CriAeroMovie_IsPlay()

概要

動画が再生中かどうか調べます。

書式

CriBool CRIAPI CriAeroMovie_IsPlay();

引数

なし。

戻り値

CRI_TRUE : 再生中。
CRI_FALSE : 停止中。

説明

動画が再生中かどうか調べます。

7.5.6 サウンド API

サウンド API は、PowerPoint®で設定した最大 32 個のサウンドを再生します。サウンドは SE と BGM の 2 種類があり、同時に再生できます。

関数の一覧です。

関数	内容
CriAeroSound_PlaySound()	サウンドを再生します。
CriAeroSound_StopSE()	SE を停止します。
CriAeroSound_StopBGM()	BGM を停止します。
CriAeroSound_SetVolumeSE()	SE の音量を設定します。
CriAeroSound_SetVolumeBGM()	BGM の音量を設定します。

CriAeroSound_PlaySound()

概要

サウンドを再生します。

書式

CriBool CRIAPI CriAeroSound_PlaySound(CriUint32 SoundId);

引数

SoundId : 再生するすサウンドを 0~31 の番号で指定します。

戻り値

CRI_TRUE : 再生成功。
CRI_FALSE : 再生失敗。

説明

引数で指定されたサウンドの再生を開始します。サウンドは SE ないし BGM に自動で割り当てられます。すでに再生中の場合は中断し、新たに指定されたサウンドを再生します。

CriAeroSound_StopSE() / CriAeroSound_StopBGM()

概要

SE/BGM の再生を停止します。

書式

```
void CRIAPI CriAeroSound_StopSE();
void CRIAPI CriAeroSound_StopBGM();
```

引数

なし。

戻り値

なし。

説明

再生中の SE ないし BGM を停止します。

CriAeroSound_SetVolumeSE() / CriAeroSound_SetVolumeBGM()

概要

SE/BGM の音量を指定します。

書式

```
void CRIAPI CriAeroSound_SetVolumeSE( CriUint32 Volume );
void CRIAPI CriAeroSound_SetVolumeBGM( CriUint32 Volume );
```

引数

Volume　:　音量を 0~100 で指定します。

戻り値

なし。

説明

SE ないし BGM の音量を指定します。

7.5.7 文字表示 API

文字表示 API は、PowerPoint®で設定した最大 16 個のフォントを指定して、画面に文字を表示します。

関数の一覧です。

関数	内容
CriAeroFont_Print()	指定したフォントで画面に文字を表示します。
CriAeroFont_Clear()	文字の表示をクリアーします。

CriAeroFont_Print()

概要

指定したフォントで、画面に文字を表示します。

書式

```
CriBool CRIAPI CriAeroFont_Print(CriSint32 X, CriSint32 Y,
                                 CriSint32 FontId, CriAerolColorID ColorId,
                                 const char *pText);
```

引数

X、Y	:	表示する XY 座標です。
FontId	:	フォントの ID(0~15)を指定します。
ColorId	:	色を指定します。
pText	:	表示する文字列です。 UTF-8 コード(3 バイトコードまで)で指定します。 文字列の長さは 64 バイトまでです。

戻り値

CRI_TRUE	:	表示成功。
CRI_FALSE	:	表示失敗。

説明

画面の任意の位置に文字列を表示します。文字列は他の表示物よりも前面に表示します。
Aeropoint ランタイムはこの関数で指定された内容を保持し、必要に応じて再描画します。

使用例

```
1    CriAeroFont_Print( 10, 20, 0, CRI_AERO_COLOR_WHITE, “Hello, world!” );
```

CriAeroFont_Clear()

概要

文字の表示をクリアーします。

書式

void CRIAPI CriAeroFont_Clear();

引数

なし。

戻り値

なし。

説明

表示中のすべての文字をクリアーします。

7.5.8 表示切替え API

表示切替え API は、PowerPoint®で設定したシェイプの ID を指定して、表示と非表示を切り替えます。

関数の一覧です。

関数	内容
CriAeroPlayer_SetShapeVisible ()	指定した ID のシェイプに対し表示、非表示を設定します。

CriAeroPlayer_SetShapeVisible ()

概要

指定した ID のシェイプに対し表示、非表示を設定します。

書式

CriSint32 CRIAPI CriAeroPlayer_SetShapeVisible(CriSint32 ShapeId, CriBool IsVisible);

引数

ShapeId : 設定するシェイプの ID を指定します。
IsVisible : シェイプの表示、非表示を指定します。
CRI_TRUE で表示、CRI_FALSE で非表示です。

戻り値

CRI_TRUE : 表示成功。
CRI_FALSE : 表示失敗。

説明

画面の任意の位置に文字列を表示します。文字列は他の表示物よりも前面に表示します。
Aeropoint ランタイムはこの関数で指定された内容を保持し、必要に応じて再描画します。

7.5.9 テスト IOAPI

テスト IOAPI は、テスト用に設定している GPIO に対して出力を設定したり、入力を取得します。

関数の一覧です。

関数	内容
CriAeroTestIO_SetLed()	指定した LED の出力を設定します。
CriAeroTestIO_CheckLed()	指定した LED の出力状態を確認します。
CriAeroTestIO_IsSwitch()	スイッチの状態を取得します。

CriAeroTestIO_SetLed ()

概要

指定した LED の出力を設定します。

書式

void CriAeroTestIO_SetLed(CriAeroTestIOLedColor led, CriBool bit);

引数

led : 設定する LED を指定します。

bit : LED に設定する値です。
CRI_TRUE で点灯、CRI_FALSE で消灯します。

戻り値

なし。

説明

引数で指定された LED へ設定を行い、点灯と消灯を制御します。

CriAeroTestIO_CheckLed ()

概要

指定した LED の出力状態を取得します。

書式

CriBool CriAeroTestIO_CheckLed(CriAeroTestIOLedColor led);

引数

led : 出力状態を取得する LED を指定します。

戻り値

CRI_TRUE : 指定した LED は点灯しています。

CRI_FALSE : 指定した LED は消灯しています。

説明

引数で指定された LED の出力状態を取得し、点灯か消灯の状態情報を返します。

CriAeroTestIO_IsSwitch ()

概要

スイッチの状態を取得します。

書式

CriBool CriAeroTestIO_IsSwitch();

引数

なし。

戻り値

CRI_TRUE : スイッチは押されていません。
CRI_FALSE : スイッチが押されています。

説明

スイッチの状態を取得し、押されているかどうかの情報を返します。

7.5.10 デバッグテキスト API

デバッグテキスト API は、フォントデータを利用せずに、画面に文字を表示します。

関数の一覧です。

関数	内容
CriAeroDebugText_SetFreeMode()	フリーモードに切り替えます。
CriAeroDebugText_SetConsoleMode()	コンソールモードに切り替えます。
CriAeroDebugText_FreePrint()	フリーモードで画面に文字を表示します。
CriAeroDebugText_ConsolePrint()	コンソールモードで画面に文字を表示します。
CriAeroDebugText_Clear()	文字の表示をクリアーします。

CriAeroDebugText_SetFreeMode()

概要

フリーモードに切り替えます。

書式

void CRIAPI CriAeroDebugText_SetFreeMode(void);

引数

なし。

戻り値

なし。

説明

デバッグテキストの出力モードをフリーモードにします。フリーモードではCriAeroDebugText_FreePrint()にて座標を指定して文字を表示することができます。

プログラム開始時の既定はフリーモードなので、コンソールモードからフリーモードに戻す際のみに使用します。

フリーモードへ切り替えの際、既に表示されている文字は消去されます。

使用例

```
1    CriAeroDebugText_SetFreeMode();
```

CriAeroDebugText_SetConsoleMode()

概要

コンソールモードに切り替えます。

書式

void CRIAPI CriAeroDebugText_SetConsoleMode(CriSint32 X, CriSint32 Y, CriUint16 lineLimit);

引数

X、Y : コンソールとして利用する矩形範囲の左上の XY 座標です。

lineLimit : コンソールとして扱う行数の上限

戻り値

なし。

説明

デバッグテキストの出力モードをコンソールモードにします。コンソールモードでは CriAeroDebugText_ConsolePrint()にてコンソール状に文字を表示することができます。

コンソールモードへ切り替えの際、既に表示されている文字は消去されます。

使用例

```
1    CriAeroDebugText_SetConsoleMode(240, 0, 17);
```

CriAeroDebugText_FreePrint()

概要

フリーモードで画面に文字を表示します。

書式

```
CriBool CRIAPI CriAeroDebugText_FreePrint(CriSint32 X, CriSint32 Y,
                CriAerolColorID ColorId, CriAerolColorID BackgroundColorId,
                const char *pText);
```

引数

X、Y	:	表示する XY 座標です。
ColorId	:	色を指定します。
BackgroundColorId	:	背景色を指定します。
pText	:	表示する文字列です。 ASCII の印字可能文字(0x20～0x7E)で指定します。 文字列の長さは 64 バイトまでです。

戻り値

CRI_TRUE	:	表示成功。
CRI_FALSE	:	表示失敗。

説明

画面の任意の位置に文字列を表示します。文字列は他の表示物よりも前面に表示します。

Aeropoint ランタイムはこの関数で指定された内容を保持し、必要に応じて再描画します。

最大で 17 行まで表示することができ、最大行数が表示されている状態で実行すると表示失敗します。

コンソールモードにて使用すると常に表示失敗します。

使用例

```
1    CriAeroDebugText_FreePrint( 10, 20, CRI_AERO_COLOR_WHITE, CRI_AERO_COLOR_BLACK, “Hello, world!”);
```

CriAeroDebugText_ConsolePrint()

概要

コンソールモードで画面に文字を表示します。

書式

```
CriBool CRIAPI CriAeroDebugText_ConsolePrint(CriAerolColorID ColorId,
                                             CriAerolColorID BackgroundColorId,
                                             const char *pText);
```

引数

ColorId	:	色を指定します。
BackgroundColorId	:	背景色を指定します。
pText	:	表示する文字列です。 ASCII の印字可能文字(0x20～0x7E)で指定します。 文字列の長さは 64 バイトまでです。

戻り値

CRI_TRUE	:	表示成功。
CRI_FALSE	:	表示失敗。

説明

画面にコンソール状に文字列を表示します。文字列は他の表示物よりも前面に表示します。

Aeropoint ランタイムはこの関数で指定された内容を保持し、必要に応じて再描画します。

コンソールモードでは文字を表示しようとすると、空いている領域の範囲で一番上に表示され、CriAeroDebugText_SetConsoleMode()で設定した行数まで保持することができ、最大行数が表示されている状態で実行した場合は一番古い行が自動で削除されます。

フリーモードにて使用すると常に表示失敗します。

使用例

```
1  CriAeroDebugText_ConsolePrint( 10, 20, CRI_AERO_COLOR_WHITE, CRI_AERO_COLOR_BLACK, “Hello, world!”);
```

CriAeroDebugText_Clear()

概要

表示中のすべての文字をクリアーします。

書式

void CRIAPI CriAeroDebugText_Clear(void);

引数

なし。

戻り値

なし。

説明

表示中のすべての文字をクリアーします。

使用例

```
1    CriAeroDebugText_Clear();
```

7.5.11 プリミティブ表示 API

プリミティブ表示 API は、テストとして用意している簡易的な図形や線を画面に表示します。

関数の一覧です。

関数	内容
CriAeroPrimitiveDraw_EndPoint()	簡易的な図形を表示します。
CriAeroPrimitiveDraw_Line()	直線を表示します。
CriAeroPrimitiveDraw_Bar()	長方形を表示します。
CriAeroPrimitiveDraw_Clear()	プリミティブ表示をクリアーします。

CriAeroPrimitiveDraw_EndPoint()

概要

簡易的な図形を表示します。

書式

void CRIAPI CriAeroPrimitiveDraw_EndPoint(CriAeroEndPointType Type,
CriSint32 X, CriSint32 Y, CriAeroColorID Color);

型

表示図形（第一引数）

```
typedef enum {
        CRI_AERO_ETYPE_RECTANGLE = 0,       // 四角
        CRI_AERO_ETYPE_CIRCLE,              // 円
        CRI_AERO_ETYPE_DIA,                 // ダイア
        CRI_AERO_ETYPE_TRIANGLE,            // 三角
} CriAeroEndPointType;
```

引数

Type ： 表示する図形を指定します。
X、Y ： 表示する XY 座標です。
Color ： 色を指定します。

戻り値

なし。

説明

画面の任意の座標に指定した図形を表示します。現在、図形の大きさは 6x6 のサイズとなります。
Aeropoint ランタイムはこの関数で指定された内容を保持し、必要に応じて再描画します。

CriAeroPrimitiveDraw_Line()

概要

直線を表示します。

書式

```
void CRIAPI CriAeroPrimitiveDraw_Line(CriSint32 X1, CriSint32 Y1,
        CriSint32 X2, CriSint32 Y2, CriAeroColorID Color );
```

引数

X1、Y1 ： 表示する線の始点となる XY 座標です。
X2、Y2 ： 表示する線の終点となる XY 座標です。
Color ： 色を指定します。

戻り値

なし。

説明

指定した 2 点、始点から終点を繋ぐ直線を表示します。
Aeropoint ランタイムはこの関数で指定された内容を保持し、必要に応じて再描画します。

CriAeroPrimitiveDraw_Bar()

概要

長方形を表示します。

書式

```
void CRIAPI CriAeroPrimitiveDraw_Bar( CriSint32 X1, CriSint32 Y1,
        CriSint32 X2, CriSint32 Y2, CriAeroColorID Color );
```

引数

X1、Y1 ： 表示する長方形の角となる XY 座標です。
X2、Y2 ： X1、Y1 の角に対して対角となる XY 座標です。
Color ： 色を指定します。

戻り値

なし。

説明

指定した 2 点を対角とする長方形を表示します。
Aeropoint ランタイムはこの関数で指定された内容を保持し、必要に応じて再描画します。

CriAeroPrimitiveDraw_Clear()

概要

長方形を表示します。

書式

```
void CRIAPI CriAeroPrimitiveDraw_Clear();
```

引数

なし。

戻り値

なし。

説明

表示中のすべてのプリミティブ表示をクリアーします。

7.5.12 データフラッシュ API

データフラッシュ API を通して、データフラッシュへの書き込みが可能です。
情報取得関数で提示された領域はユーザーの利用が可能です。書き込みはブロック単位となります。書き込んだデータは電源を切っても保存されるため、アプリのセーブデータ等に利用できます。

関数の一覧です。

関数	内容
CriAeroDataFlash_QueryAboutDataFlash ()	データフラッシュの情報を確認します。
CriAeroDataFlash_Write ()	データフラッシュに書き込みます。

CriAeroDataFlash_QueryAboutDataFlash ()

概要

データフラッシュの情報を確認します。

書式

```
void CriAeroDataFlash_QueryAboutDataFlash(
CriUint32 *StartAddress,CriUint32 *BytesBlock,CriUint32 *NumBlocks);
```

引数

StartAddress : データフラッシュのスタートアドレスが返却されます。
BytesBlock : １ブロックのバイト数が返却されます。
NumBlocks : データフラッシュの総ブロック数が返却されます。

戻り値

なし。

説明

データフラッシュ上のユーザーデータ領域の情報を取得します。書き込みは BytesBlock で示された単位となります。

CriAeroDataFlash_Write ()

概要

データフラッシュにデータを書き込みます。

書式

CriUint32 CriAeroDataFlash_Write(CriUint32 SrcAddress , CriUint32 DestAddress,CriUint32 NumBlocks);

引数

SrcAddress : 書き込み元のアドレスを指定します。

DestAddress : 書き込み先のアドレスを指定します。StartAddress からの、ブロック単位を前提としています。

NumBlocks : 書き込みの量をブロック数で示します。

戻り値

成功時は 0 を返します。それ以外はエラーです。

説明

データフラッシュにデータを書きこみます。

7.5.13 シリアル通信 API

シリアル通信 API を使用することにより、別の機器とのシリアル通信が可能です。

これらの関数を呼び出すことで「7.5.2 UART コマンド」に書かれた UART コマンドを送信したり、ログ受信時の動作を登録します。

関数の一覧です。

関数	内容
CriAeroSerial_Start()	シリアル通信を開始します。
CriAeroSerial_Stop()	シリアル通信を停止します。
CriAeroSerial_ChangeVolumeBgm()	BGM のボリュームを指定した値に変えるコマンドを送信します。
CriAeroSerial_ChangeVolumeSe()	SE のボリュームを指定した値に変えるコマンドを送信します。
CriAeroSerial_SendMoviePlay()	指定した動画を再生するコマンドを送信します。
CriAeroSerial_SendMovieStop()	動画を停止するコマンドを送信します。
CriAeroSerial_SendScrUpdate()	画面を再描画するコマンドを送信します。
CriAeroSerial_SendShapeOff()	指定したシェイプを非表示にするコマンドを送信します。
CriAeroSerial_SendShapeOn()	指定したシェイプを表示するコマンドを送信します。
CriAeroSerial_SendShowTitle()	タイトルを表示するコマンドを送信します。
CriAeroSerial_SendSlideNo()	指定したスライドを表示するコマンドを送信します。
CriAeroSerial_SendSoundPlay()	指定したサウンドを再生するコマンドを送信します。
CriAeroSerial_SendSoundStop()	サウンドを停止するコマンドを送信します。
CriAeroSerial_SendTestDemo()	コマンドを送信します。
CriAeroSerial_SendTextClear()	テキストを消去するコマンドを送信します。
CriAeroSerial_SendTextPrint()	指定したテキストを表示するコマンドを送信します。
CriAeroSerial_LogString()	ログメッセージを送信します。
CriAeroSerial_SetLogCallback()	ログを受信した際のコールバック関数を登録します。

CriAeroSerial_Start()

概要

シリアル通信を開始します。

書式

void CriAeroSerial_Start(void);

引数

なし。

戻り値

なし。

説明

シリアル通信を開始します。

コマンドを送信する関数より前に呼び出してください。

CriAeroSerial_Stop()

概要

シリアル通信を停止します。

書式

void CriAeroSerial_Stop(void);

引数

なし。

戻り値

なし。

説明

シリアル通信を停止します。

CriAeroSerial_ChangeVolumeBgm() / CriAeroSerial_ChangeVolumeSe()

概要

BGM / SE のボリュームを指定した値に変えるコマンドを送信します。

書式

```
void CriAeroSerial_ChangeVolumeBgm(CriUint32 Vol);
void CriAeroSerial_ChangeVolumeSe(CriUint32 Vol);
```

引数

Vol　:　音量を 0~255 で指定します。

戻り値

なし。

説明

BGM ないし SE のボリュームを指定した値に変えるコマンドをシリアル通信で送信します。
CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SendMoviePlay()

概要

指定した動画を再生するコマンドを送信します。

書式

void CriAeroSerial_SendMoviePlay(CriUint32 Id);

引数

Id　:　再生するすサウンドを 0~31 の番号で指定します。

戻り値

なし。

説明

指定した動画を再生するコマンドをシリアル通信で送信します。

CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SendMovieStop()

概要

動画を停止するコマンドを送信します。

書式

void CriAeroSerial_SendMovieStop(void);

引数

なし。

戻り値

なし。

説明

動画を停止するコマンドをシリアル通信で送信します。

CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SendScrUpdate()

概要

画面を再描画するコマンドを送信します。

書式

void CriAeroSerial_SendScrUpdate(void);

引数

なし。

戻り値

なし。

説明

画面を再描画するコマンドをシリアル通信で送信します。

CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SendShapeOff()

概要

指定したシェイプを非表示にするコマンドを送信します。

書式

void CriAeroSerial_SendShapeOff(CriUint32 Id);

引数

Id　:　非表示にするシェイプの ID を指定します。

戻り値

なし。

説明

指定したシェイプを非表示にするコマンドをシリアル通信で送信します。
CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SendShapeOn()

概要

指定したシェイプを表示するコマンドを送信します。

書式

void CriAeroSerial_SendShapeOn(CriUint32 Id);

引数

Id　:　表示するシェイプの ID を指定します。

戻り値

なし。

説明

指定したシェイプを表示するコマンドをシリアル通信で送信します。
CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SendShowTitle()

概要

タイトルを表示するコマンドを送信します。

書式

void CriAeroSerial_SendShowTitle(void);

引数

なし。

戻り値

なし。

説明

タイトルを表示するコマンドをシリアル通信で送信します。
CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SendSlideNo()

概要

指定したスライドを表示するコマンドを送信します。

書式

void CriAeroSerial_SendSlideNo(CriUint32 No);

引数

No　:　表示するスライドの番号を指定します。

戻り値

なし。

説明

指定したスライドを表示するコマンドをシリアル通信で送信します。
CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SendSoundPlay()

概要

指定したサウンドを再生するコマンドを送信します。

書式

void CriAeroSerial_SendSoundPlay(CriUint32 Id);

引数

Id　:　再生するサウンドを 0~31 の番号で指定します。

戻り値

なし。

説明

指定したサウンドを再生するコマンドをシリアル通信で送信します。

CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SendSoundStop()

概要

サウンドを停止するコマンドを送信します。

書式

void CriAeroSerial_SendSoundStop(void);

引数

なし。

戻り値

なし。

説明

サウンドを停止するコマンドをシリアル通信で送信します。

CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SendTestDemo()

概要

コマンドを送信します。

書式

void CriAeroSerial_SendTestDemo(CriUint32 No);

引数

No　:　実行するデモを番号で指定します。

戻り値

なし。

説明

コマンドをシリアル通信で送信します。

CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SendTextClear()

概要

テキストを消去するコマンドを送信します。

書式

void CriAeroSerial_SendTextClear(void);

引数

なし。

戻り値

なし。

説明

テキストを消去するコマンドをシリアル通信で送信します。
CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SendTextPrint()

概要

コマンドを送信します。

書式

```
void CriAeroSerial_SendTextPrint(CriSint32 Index, CriSint32 X, CriSint32 Y,
                                 CriSint32 FontId, CriSint32 ColorId,
                                 const CriChar8 *pText);
```

引数

Index	:	テキスト ID(0～7)を指定します。
X、Y	:	表示する XY 座標です。
FontId	:	フォントの ID(0~15)を指定します。
ColorId	:	色を指定します。
pText	:	表示する文字列です。 UTF-8 コード(3 バイトコード)で指定します。

戻り値

なし。

説明

コマンドをシリアル通信で送信します。
CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_LogString()

概要

ログメッセージを送信します。

書式

```
void CriAeroSerial_LogString(const CriChar8 *pText);
```

引数

pText　:　転送する文字列です。

戻り値

なし。

説明

ログメッセージをシリアル通信で送信します。
指定した文字列の転送前に「log」というメッセージが転送されます。
CriAeroSerial_Start を呼び出した後に実行してください。

CriAeroSerial_SetLogCallback()

概要

ログを受信した際に呼び出すコールバック関数を登録します。

書式

void CriAeroSerial_SetLogCallback(CRI_AERO_SERIAL_LOG_CALLBACK *pFunction);

型

コールバック関数

```
typedef void CRI_AERO_SERIAL_LOG_CALLBACK(CriUint32 Length,const CriChar8 *pText);
```

引数

pFunction　：　コールバック関数。

戻り値

なし。

説明

ログを受信した際に呼び出すコールバック関数を登録します。

7.6　サンプルプログラム

他のサンプルも参考にしながら、API の使用例を見ていきましょう。

7.6.1 スライドを切替える

- プロジェクト
  - Sample/Project/AeropointRX72N_Sample_SlideRead.zip
- GUI
  - Sample/GUI/Sample_SlideRead.pptx

プロジェクトは e²studio にインポートします。GUI は「5.1.1 全て書込み」を参考にして、SD カードに書き込んでおきます。

3 秒ごとに 3 つのスライドを 1→2→3→1…と順番に表示します。メインループは 1 周で 1/60 秒が経過するので、180 ループで 3 秒が経過します。

```
1    /* Main loop */
2    while( flag )
3    {
4        /* Main */
5        flag = CriAeroPlayer_Main();
6
7        flame_count ++;
8        /* 3second */
9        if( flame_count >= 180 ){
10
11           /* Read next slide */
12           slide_index = (slide_index % 3) + 1;
13           CriAeroPlayer_ReadSlide(slide_index);
14
15           flame_count = 0;
16       }
17
18   }
```

スライド番号を更新し、
表示スライドを切替え

図 144　スライド切り替えのサンプルコード

実行すると以下のような画面が表示されます。

図 145　動作する様子

7.6.2 画面に文字を表示する

- プロジェクト
  - Sample/Project/AeropointRX72N_Sample_TextPrint.zip
- GUI
  - Sample/GUI/Sample_TextPrint.pptx

ボタンをタップするごとに、テキストを昇順か降順に表示します。Aeropoint ツールキットで、上ボタンの ID は 0、下ボタンの ID は 1 に設定しています。

```
CriAeroResponsed mycallback( CriAeroMessage Message, CriSint32 Param, CriSint32 SoundId, CriSint32
 MovieId )
{
    CriAeroResponsed res = CRI_AERO_RESPONSED_NONE;

    /* Button was pressed */
    if( Message == CRI_AERO_MESSAGE_BUTTON_PRESS ){
        /* Up Button */
        if( Param == 0 ){
            /* Index count up */
            s_text_index = (s_text_index + 1)%3;

            /* Clear & Print Text */
            CriAeroFont_Clear();
            CriAeroFont_Print( 65, 120, 0, CRI_AERO_COLOR_WHITE, s_print_text[s_text_index]);
        }
        /* Down Button */
        if( Param == 1 ){

            /* Index count down */
            s_text_index = (s_text_index + 2)%3;

            /* Clear & Print Text */
            CriAeroFont_Clear();
            CriAeroFont_Print( 65, 120, 0, CRI_AERO_COLOR_WHITE, s_print_text[s_text_index]);
        }
    }

    return res;
}
```

文字番号を正順で更新し、文字を表示

文字番号を逆順で更新し、文字を表示

図 146　テキスト表示のサンプルコード

実行すると以下のような画面が表示されます。

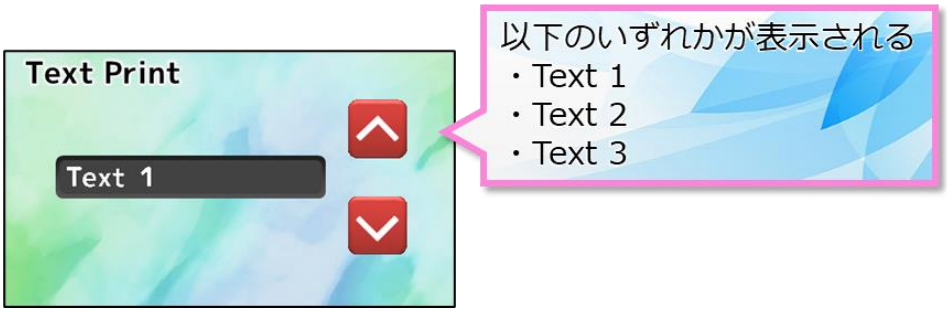


図 147 アプリケーション実行画面

7.6.3 サウンドを再生する

- プロジェクト
  - Sample/Project/AeropointRX72N_Sample_SoundPlay.zip
- GUI
  - Sample/GUI/Sample_SoundPlay.pptx

テキスト表示に再生ボタンを加えて、サウンドを鳴らすようにしました。Aeropoint ツールキットで、再生ボタンの ID は 2 に設定しています。

```
CriAeroResponsed mycallback( CriAeroMessage Message, CriSint32 Param, CriSint32 SoundId, CriSint32
 MovieId )
{
    CriAeroResponsed res = CRI_AERO_RESPONSED_NONE;

    /* Button was pressed */
    if( Message == CRI_AERO_MESSAGE_BUTTON_PRESS ){

        :
        :
        :

        /* Play Button */
        if( Param == 2 ){

            /* Play sound */
            CriAeroSound_PlaySound( s_sound_index );

            /* Cancel default sound control */
            res |= CRI_AERO_RESPONSED_PLAY_SOUND;
        }
    }

    return res;
}
```

サウンドを再生する

サウンドのデフォルト処理をキャンセルする

図 148 サウンド再生のサンプルコード

実行すると以下のような画面が表示されます。再生ボタンでサウンドが鳴ります。

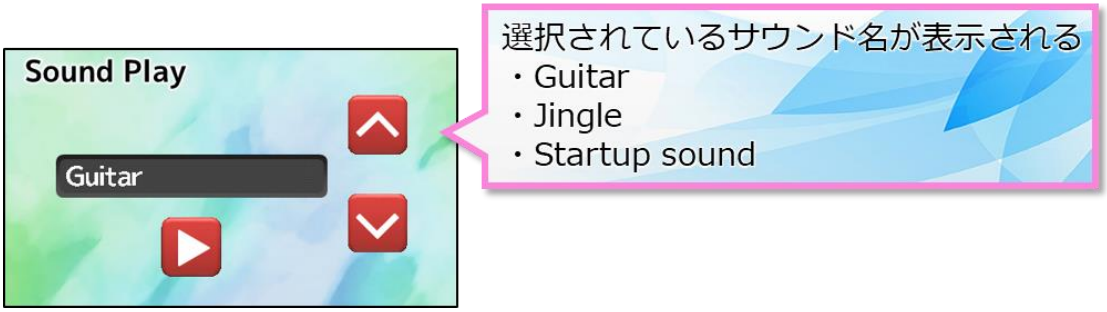


図 149 アプリケーション実行画面

7.6.4 動画を再生する

- プロジェクト
  - Sample/Project/AeropointRX72N_Sample_MoviePlay.zip
- GUI
  - Sample/GUI/Sample_MoviePlay.pptx

サウンドに変えて、動画を再生します。

```
CriAeroResponsed mycallback( CriAeroMessage Message, CriSint32 Param, CriSint32 SoundId, CriSint32
 MovieId )
{
    CriAeroResponsed res = CRI_AERO_RESPONSED_NONE;

    /* Button was pressed */
    if( Message == CRI_AERO_MESSAGE_BUTTON_PRESS ){

        :
        :
        :

        /* Play Button */
        if( Param == 2 ){

            /* Play movie */
            CriAeroMovie_Play( s_movie_index );

            /* Cancel default movie control */
            res |= CRI_AERO_RESPONSED_PLAY_MOVIE;
        }
    }

    return res;
}
```

動画を再生する

動画のデフォルト処理をキャンセルする

図 150　動画を再生する

実行すると以下のような画面が表示されます。再生ボタンで動画を再生します。

図 151 アプリケーション実行画面

7.6.5 表示を切り替える

- プロジェクト
  - Sample/Project/AeropointRX72N_Sample_ShapeSwitch.zip
- GUI
  - Sample/GUI/Sample_ShapeSwitch.pptx

3 秒ごとにシェイプの表示、非表示を繰り返します。メインループは 1 周で 1/60 秒が経過するので、180 ループで 3 秒が経過します。

円形シェイプのシェイプ ID に 1 を設定しています。

```
1   /* Main loop */
2   while( flag )
3   {
4       /* Main */
5       flag = CriAeroPlayer_Main();
6
7       flame_count ++;
8       /* 3second */
9       if( flame_count >= 180 ){
10
11          /* Change shape visible */
12          if(shape_visible == CRI_TRUE)shape_visible = CRI_FALSE;
13          else shape_visible = CRI_TRUE;
14          CriAeroPlayer_SetShapeVisible(1, shape_visible);
15
16          flame_count = 0;
17      }
18  }
```

シェイプの表示、非表示を指定。

図 152　表示切り替えのサンプルコード

実行すると以下のような画面が表示されます。

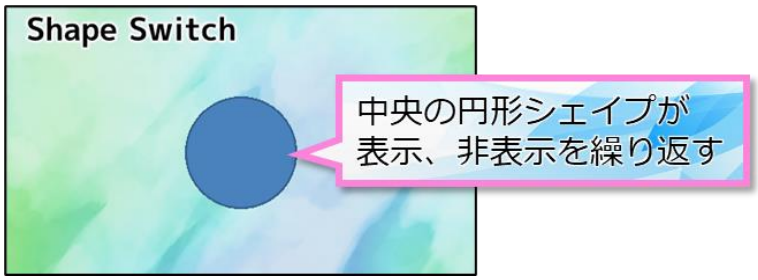


図 153　アプリケーション実行画面

7.6.6 テスト IO を制御する

- プロジェクト
  - Sample/Project/AeropointRX72N_Sample_TestIOSwitch.zip
- GUI
  - 使用しません。

3 秒ごとに緑の LED が点灯と消灯を繰り返します。メインループは 1 周で 1/60 秒が経過するので、180 ループで 3 秒が経過します。

```
1   /* Main loop */
2   while( flag )
3   {
4       /* Main */
5       flag = CriAeroPlayer_Main();
6
7       flame_count ++;
8       /* 3second */
9       if( flame_count >= 180 ){
10
11          /* Check led output */
12          if(CRI_TRUE == CriAeroTestIO_CheckLed(CRI_AERO_TESTIO_LED_GREEN)){
13              led_bit = CRI_FALSE;
14          }
15          else {
16              led_bit = CRI_TRUE;
17          }
18
19          /* Setting led output */
20          CriAeroTestIO_SetLed(CRI_AERO_TESTIO_LED_GREEN, led_bit);
21
22          flame_count = 0;
23      }
24
25  }
```

LED の出力情報を取得し、次の出力値を決める。

LED の出力情報を指定。

図 154　テスト IO 制御のサンプルコード

実行すると D-Amp 基板の LED が以下のように点灯します。

図 155 アプリケーション実行時の LED

7.6.7 画面にデバッグテキストを表示する

- プロジェクト
  - Sample/Project/AeropointRX72N_Sample_DebugTextPrint.zip
- GUI
  - Sample/GUI/Sample_DebugTextPrint.pptx

右矢印ボタンをタップするごとに、枠内にデバッグテキストを表示します。
点線の丸ボタンをタップすると、デバッグテキストを全て削除します。
モードボタンをタップすると、フリーモードとコンソールモードを切り替えます。
Aeropoint ツールキットで、右矢印ボタンの ID は 0、点線ボタンの ID は 1 に設定し、モードボタンではスライドの切り替えを設定しています。

```
1     /* Button was pressed */
2     if( Message == CRI_AERO_MESSAGE_BUTTON_PRESS ){
3         /* Print Button */
4         if( Param == 0 ){
5
6             /* Print DebugText */
7             if(s_mode_index == 0){
8                 print_res = CriAeroDebugText_FreePrint( 50 + (rand()%150), 90 + (rand()%120), CRI_AERO_COLOR_WHITE, CRI_AERO_COLOR_BLACK, s_print_debugtext[s_debugtext_index]);
9             }
10            else if(s_mode_index == 1){
11                print_res = CriAeroDebugText_ConsolePrint(CRI_AERO_COLOR_WHITE, CRI_AERO_COLOR_BLACK, s_print_debugtext[s_debugtext_index]);
12            }
13
14            if(print_res == CRI_TRUE){
15                /* Index count up */
16                s_debugtext_index = (s_debugtext_index + 1)%17;
17            }
18        }
19        /* Clear Button */
20        if( Param == 1 ){
21            /* Clear DebugText */
22            CriAeroDebugText_Clear();
23            s_debugtext_index = 0;
24        }
25    }
26
27    /* Slide was changed */
28    if( Message == CRI_AERO_MESSAGE_SLIDE_PRE_CHANGE ){
29        /* Free print Slide */
30        if(Param == 1){
31            CriAeroDebugText_SetFreeMode();
32            s_mode_index = 0;
33        }
34        /* Console print Slide */
35        else if( Param == 2 ){
36            CriAeroDebugText_SetConsoleMode(50,90,9);
37            s_mode_index = 1;
38        }
39
40        s_debugtext_index = 0;
41    }
```

デバッグテキストの表示。
モードによって表示関数が変わる。

デバッグテキストの削除。

モードの切り替え。

図 156　デバッグテキストの表示のサンプルコード

図 157　アプリケーション実行画面

7.6.8 画面にプリミティブな図形を表示する

- プロジェクト
  - Sample/Project/AeropointRX72N_Sample_PrimitiveDraw.zip
- GUI
  - Sample/GUI/Sample_PrimitiveDraw.pptx

1 秒ごとに色の違う長方形を表示していき、4 秒で全てクリアーを繰り返します。メインループは 1 周で 1/60 秒が経過するので、60 ループで 1 秒が経過します。

```
1     /* Main loop */
2     while( flag )
3     {
4         /* Main */
5         flag = CriAeroPlayer_Main();
6
7         flame_count ++;
8         /* 1second */
9         if( flame_count >= 60 ){
10
11             /* Primitive draw */
12             if(primitive_count == 1){
13                 CriAeroPrimitiveDraw_Bar( 110, 100, 130, 170, CRI_AERO_COLOR_RED);
14             }
15             else if(primitive_count == 2){
16                 CriAeroPrimitiveDraw_Bar( 230, 100, 250, 170, CRI_AERO_COLOR_GREEN);
17             }
18             else if(primitive_count == 3){
19                 CriAeroPrimitiveDraw_Bar( 350, 100, 370, 170, CRI_AERO_COLOR_BLUE);
20             }
21             else {
22                 CriAeroPrimitiveDraw_Clear();
23             }
24
25             primitive_count = (primitive_count+1)%4;
26             flame_count = 0;
27         }
28     }
```

長方形の表示。

プリミティブ表示の削除。

図 158　プリミティブ表示のサンプルコード

実行すると以下のような画面が表示されます。

図 159　アプリケーション実行画面

7.6.9 シリアル通信を行う

- プロジェクト
  - Sample/Project/AeropointRX72N_Connector.zip
- GUI
  - Sample/GUI/Sample_Connector.pptx

UART 通信を使用します。別機材との接続の仕方は「8 シリアル通信の接続方法」参照してください。
転送する No や Id、コマンドをボタンで選択し、シリアル通信で送信を行います。
また、受信したログを表示します。

```
 1  void main(void)
 2  {
 3      CriBool flag = CRI_TRUE;
 4
 5      /* Initialize */
 6      CriAeroPlayer_Initialize();
 7      CriAeroCommandQueue_SetSendStatusMode(CRI_FALSE);
 8
 9      /* Serial Start */
10      CriAeroSerial_Start();
11      /* Set serial reception log callback */
12      CriAeroSerial_SetLogCallback(uartLogCallback);
13
14       :
15       :
16
17      /* Main loop */
18      while (flag)
19      {
20          :
21          :
22
23          /* Main */
24          flag = CriAeroPlayer_Main();
25      }
26
27      /* Serial Stop */
28      CriAeroSerial_Stop();
29
30      /* Finalize */
31      CriAeroPlayer_Finalize();
32
33      return;
34  }
```

シリアル通信の開始、
ログ受信コールバック関数の登録

シリアル通信の停止

図 160　シリアル通信開始部分のサンプルコード

```
 1      /* Button was pressed */
 2     if (Message == CRI_AERO_MESSAGE_BUTTON_PRESS) {
 3         switch (Param) {
 4
 5          :
 6
 7         /* Send button */
 8         case 10: {
 9             switch (nowSlide) {
10             case 1:
11                 /* Page change */
12                 CriAeroSerial_SendSlideNo(nowNumber[nowSlide]);
13                 break;
14             case 2:
15                 /* Shape visible */
16                 CriAeroSerial_SendShapeOn(nowNumber[nowSlide]);
17                 break;
18             case 3:
19                 /* Sound play */
20                 CriAeroSerial_SendSoundPlay(nowNumber[nowSlide]);
21                 break;
22             case 4:
23                 /* Movie play */
24                 CriAeroSerial_SendMoviePlay(nowNumber[nowSlide]);
25                 break;
26             }
27             res = CRI_AERO_RESPONSED_BUTTON_PRESS;
28             break;
29         }
30         case 11: {
31             switch (nowSlide) {
32             case 2:
33                 /* Shape unvisible */
34                 CriAeroSerial_SendShapeOff(nowNumber[nowSlide]);
35                 break;
36             case 3:
37                 /* Sound stop */
38                 CriAeroSerial_SendSoundStop();
39                 break;
40             case 4:
41                 /* Movie stop */
42                 CriAeroSerial_SendMovieStop();
43                 break;
44             }
45             res = CRI_AERO_RESPONSED_BUTTON_PRESS;
46             break;
47         }
48         case 12: {
49             switch (nowSlide) {
50             case 3:
51             case 4:
52                 /* Volume up */
53                 volume++;
54                 if( volume > 4 ){ volume = 4; }
55                 CriAeroSerial_ChangeVolumeBgm( volval[volume] );
56                 CriAeroSerial_ChangeVolumeSe(  volval[volume] );
57                 break;
58             }
59             res = CRI_AERO_RESPONSED_BUTTON_PRESS;
60             break;
61         }
62         case 13: {
63
64              :
65
66         }
67         }
68     }
```

スライドごとに
　指定したスライドの表示、
　指定したシェイプの表示、
　指定したサウンドの再生、
　指定した動画の再生
のコマンドを送信

スライドごとに
　指定したシェイプの非表示、
　サウンドの停止、
　動画の停止
のコマンドを送信

スライドごとに
　BGM の音量設定、
　SE の音量設定、
のコマンドを送信

図 161　シリアル通信コマンド送信部分のサンプルコード

実行すると以下のような画面が表示されます。

図 162　アプリケーション実行画面

8　シリアル通信の接続方法

実機と別の機材とを接続しシリアル通信を行うことで、アプリケーションを外部から操作することができます。

これは UART の仕組みを利用します。Master 側（お客様の MPU など）のデバイスからコマンドを送出すると Slave 側（Aeropoint GUI）のデバイスを操作でき、必要に応じてコマンドの結果が Slave 側のデバイスから Master 側のデバイスへ送出されます。

UART 接続には CN5 端子を使用します。ボーレートは 9600bps です。

以下のように接続してください。例は、RX72N 同士での接続となります。

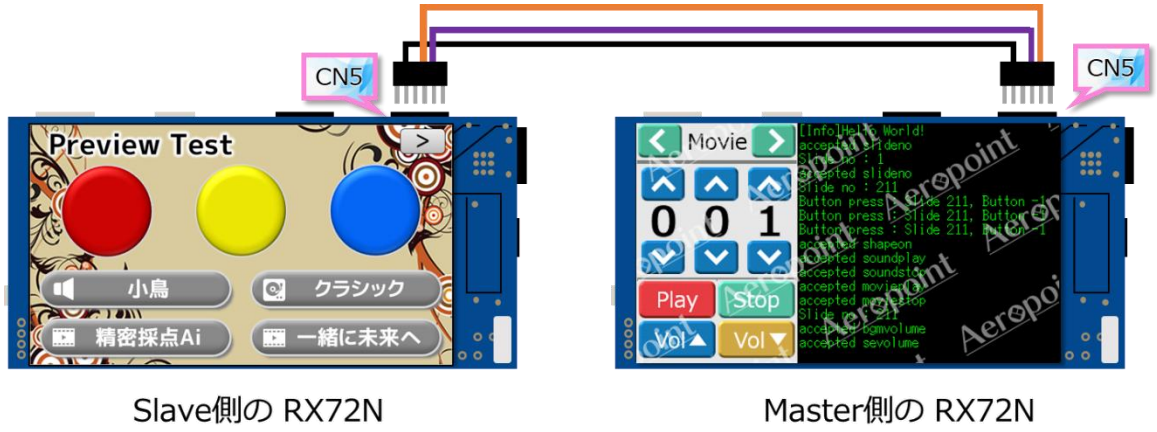


図 163　シリアル通信の接続図

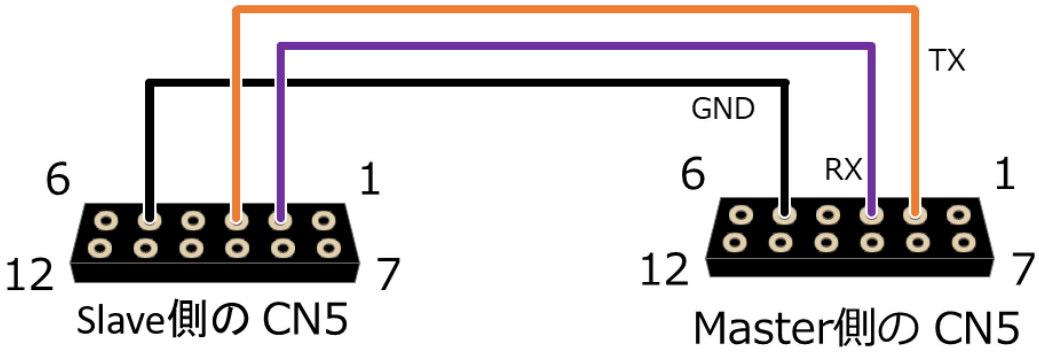


図 164　CN5 のピン接続拡大図

RX72N 同士ではなく別の機材から Aeropoint を操作する場合には、Master 側の RX72N を任意の機材に変更し、接続を行ってください。

9　フォント作成

実機にテキストを表示するためのフォントデータを作成します。

9.1　起動

Aeropoint ツールキットの「オプション」グループから「フォント作成」を選択します。

図 165　オプショングループ

「フォント作成」ツールが起動します。

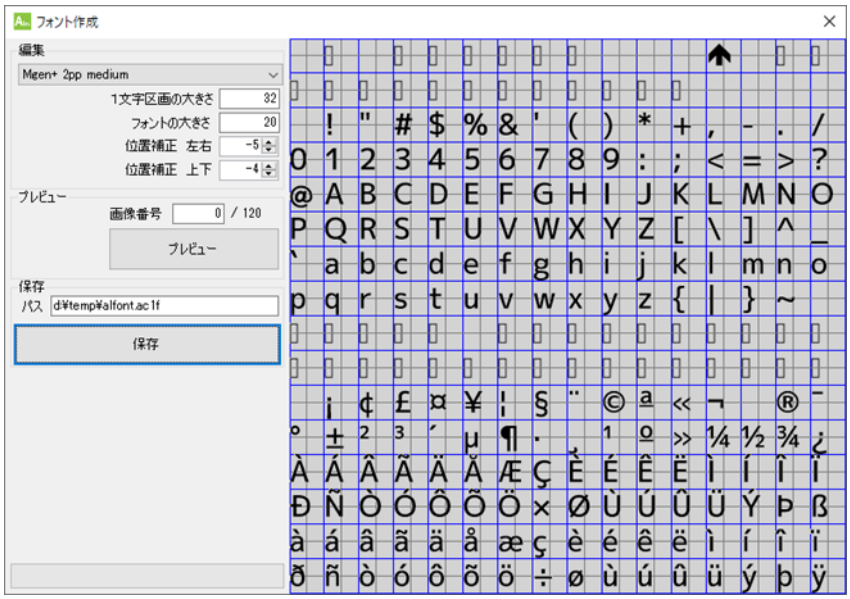


図 166　フォント作成ツール

9.2 使い方

「編集」グループボックスのプルダウンには、PC で利用可能なフォントの一覧が登録されています。フォント作成ツールは、それらフォントをどれかひとつ選んで、実機で利用するデータを作成します。

> フォントを利用するために必要となる権利は、お客様が個別にフォント制作会社と契約するなどして確保する必要があります。PC で表示されるフォントでも、誰かしらの権利保持者がおり、そのまま商用利用できることはありません。

「１文字区画の大きさ」は、ひとつのフォントを格納数する区画の大きさをピクセル単位指定します。フォント作成ツールの右ウィンドウの中で「青枠」で表示されている区画となります。区画の大きさには制限があります。

- 区画の大きさ
  - 最大 60 ピクセル

「フォントの大きさ」を指定したら、「プレビュー」をクリックして、右ウィンドウにフォントの様子を表示させます。フォントの大きさを調整したり位置を補正したりするなど、微調整を行います。「画像番号」を変えてプレビューすると、全角漢字などの文字も確認できます。

個々の文字は、青色の矩形領域の中に左詰めになるようにします。実機で文字を並べて表示する際には、文字と文字の間の空間は自動で詰められます。

調整が終わったら保存先のパスを決めて「保存」をクリックします。取り扱う文字と計算量が多いため、保存には時間が掛かります。保存された AC1FC 形式ファイルは、「5.2.1 文字」などの仕組みを使って実機に挿入された SD カードに書き込んで利用します。

10 プログラムの書き込み

サンプルプログラムの実行バイナリを直接マイコンに書き込むために、PC に Renesas Flash Programmer をインストールしておきます。

10.1 Renesas Flash Programmer

以下のアプリケーションをダウンロードして、インストールします。

- 【無償版】Renesas Flash Programmer V3.06.01
  - https://www.renesas.com/jp/ja/software/D4001217.html

図 167 ダウンロードページ

10.2 書き込み手順

1. 基板のディップスイッチを書き込みモードにする

   基板を以下のように見て、SW1 / SW3 の位置を合わせます。また、SW1 に隣接する ECN1 端子と PC を MicroUSB で接続します。

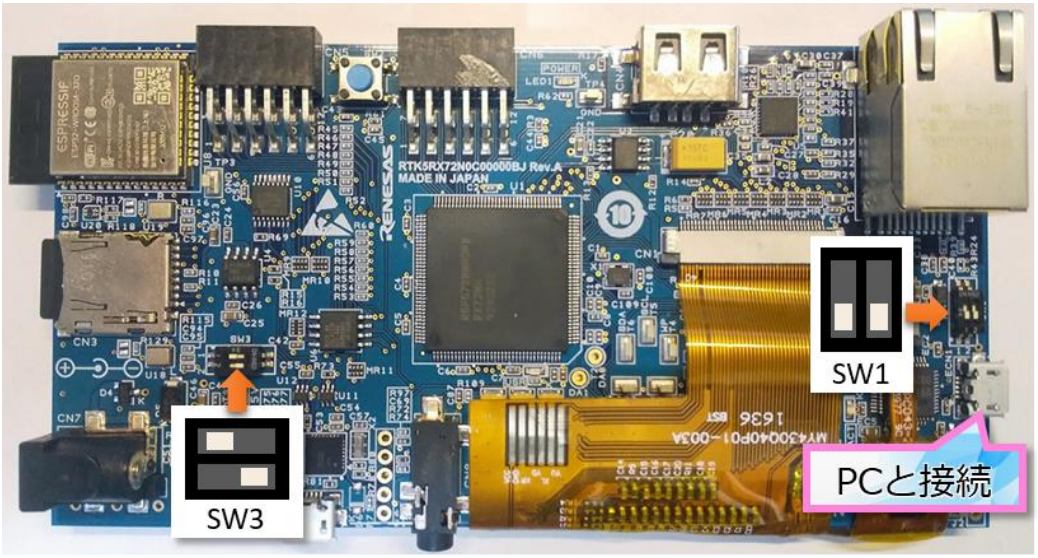


図 168 書き込みモードのディップスイッチの様子

2. Renesas Flash Programmer でプログラムを書き込む

新しいプロジェクトを作成するにあたり、「マイクロコントローラー」を「RX72x」に、「ツール」を「E2 emulator Lite」に、「インターフェース」を「FINE」に設定し「接続」をクリックします。

図 169 Renesas Flash Programmer のプロジェクト設定

ID コードは全て「F」とします。

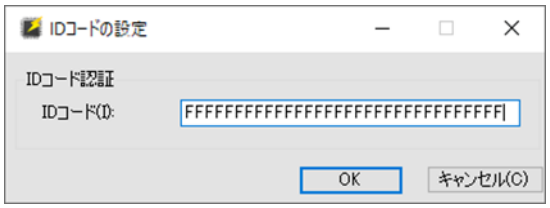


図 170 ID コードの設定

「プログラムファイル」に書き込む mot ファイルを指定したら「スタート」をクリックします。Aeropoint プレイヤーを書き込む場合は、「Sample/Player/Player.mot」を指定します。

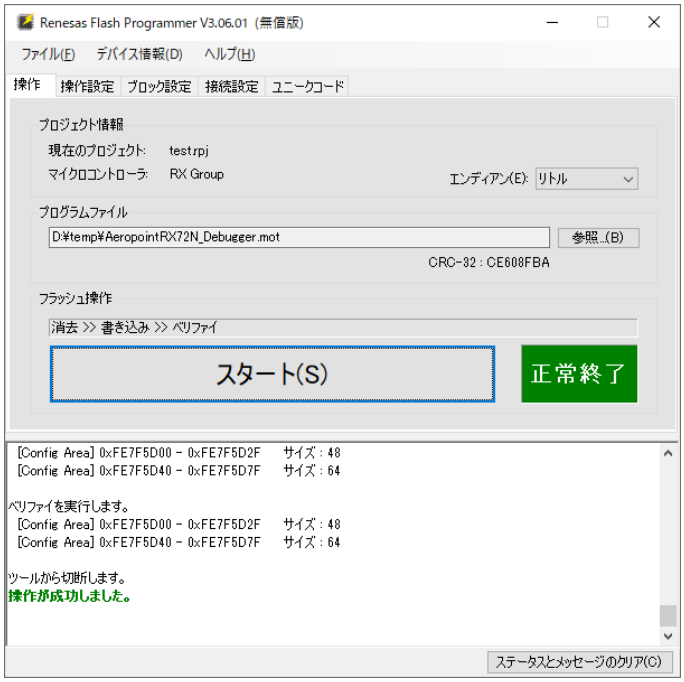


図 171　プログラムの書き込み

3.　基板のディップスイッチを単体起動モードにする

プログラムの書き込みに成功したら、SW1 を変更します。

図 172　プログラム起動モードのディップスイッチの様子

書込みを行ったプログラムが起動します。

11 データイメージの書き込み

リファレンスキットでは、microSD カード内のデータを読み出しながら、画面の表示や動画・サウンドの再生を行います。この章では、付属するデータイメージを microSD カードに正しく書き込むための手順を説明します。

11.1 用意するもの

- microSD カード
  - データイメージは 1GB 程度ですので、あまり大容量なものは必要ありません。

- 書き込むデータイメージ
  - リファレンスキットに付属の「Sample/SDImage/SDImage_xx.img」を用います。

- Win32 Disk Imager
  - イメージの書き込みに使用するツールです。
    以下からダウンロードできます。
    https://sourceforge.net/projects/win32diskimager/
    ※ Release 1.0.0 を用いて検証しました。

11.2 書き込み手順

1. Win32 Disk Imager を起動

   Win32 Disk Imager を起動します。起動時に管理者権限が要求され UAC によるメッセージが表示されますので、「はい」を押します。

図 173　メッセージの例

2. 正しいデバイスが選択されていることを確認する

Device 欄に選択されているドライブレターが正しいデバイス(microSD カード)であることを確認します。

> Device 欄では HDD や SSD などのドライブレターは原則として候補に表示されませんが、複数の USB リムーバブルメディアが接続されている場合は注意が必要です。

3. データイメージを開いてコピーする

Image File 欄の右端にある青いフォルダのアイコンを押し、SD カードに書き込むデータイメージ（ここでは SDImage_xx.img）を選択します。

4. microSD カードに書き込む

すべての項目が正しく設定されていることを確認し、Write ボタンを押します。

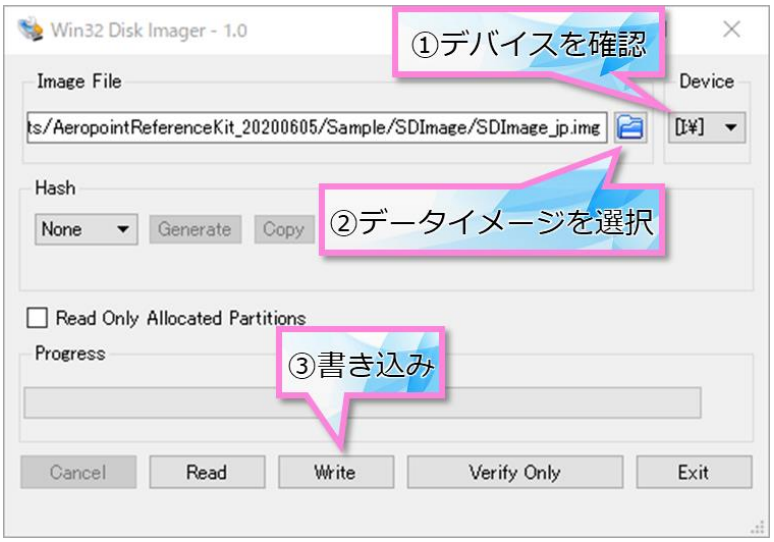


図 174　操作画面

12 Q&A

よくある質問や問題への回答や解決法です。

12.1 Aeropoint ツールキットのアドインが表示されない

「3.2.2 Aeropoint ツールキット」の手順通り Aeropoint ツールキットをインストールしてもメニューに「AeopointLite」が表示されていない場合、以下の手順を試してみてください。

1. メニューバー「ファイル」から「オプション」を選択します。

図 175 ファイルをクリック

図 176 オプションを選択

2. 「PowerPoint のオプション」ウィンドウから、左のメニューで「アドイン」を選択します。

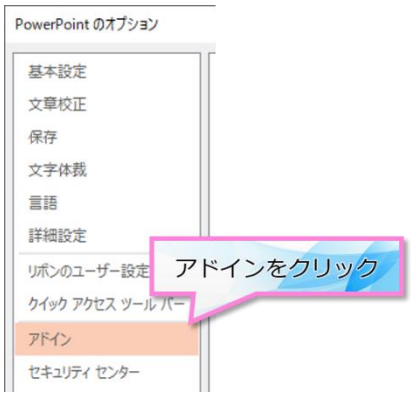


図 177 アドインを選択

3. 「アクティブなアプリケーション」に「AeropointLite32AddIn」が存在するかを確認してください。

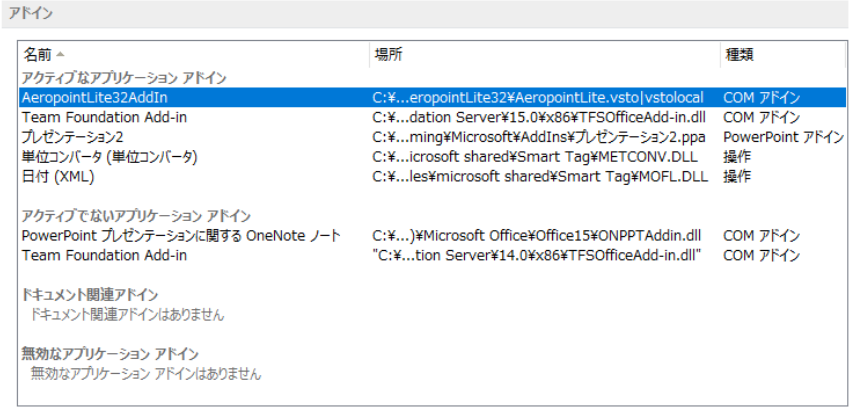


図 178 「AeropointLite32AddIn」が存在する場合の表示

もし「アクティブなアプリケーション」に「AeropointLite32AddIn」が存在しない場合、以下を試してください。

4. アドインのページ下部にある管理から「COM アドイン」を選択し、「設定」をクリックします。

図 179 COM アドインを選択

5. 使用できるアドインにある「AeropointLite32AddIn」のチェックボックスにチェックを入れ、「OK」をクリックします。

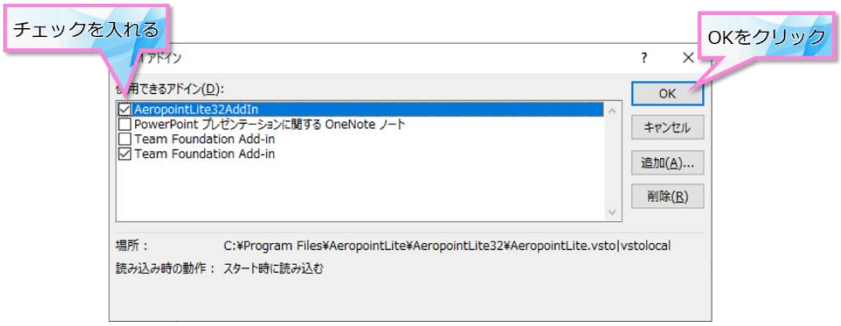


図 180 COM アドインの設定

この時、使用できるアドインの中に「AeropointLite32AddIn」が存在しない場合、更に以下を試してください。

6. アドインのページ下部にある管理から「使用できないアイテム」を選択し、「設定」をクリックします。

図 181 使用できないアイテムを選択

7. 表示されるアイテムの「AeropointLite32AddIn」を選択し、「有効にする」をクリックしてください。

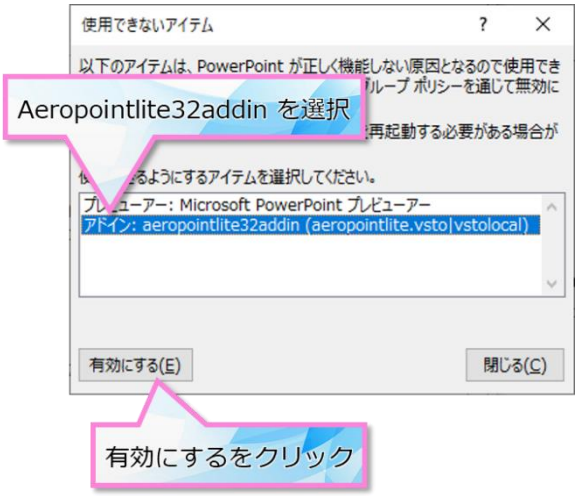


図 182 使用できないアイテムの設定

8. 「AeropointLite32AddIn」を有効にした後、上記 4.～5.の COM アドイン設定の手順が必要です。

12.2 スライドのテーマやマスタースライドを利用すると、実機で正しく部品が配置されない

「スライドのテーマ」や「マスタースライド」での画像の配置には対応していません。これらの仕組みで画像が配置されていると、通常のスライド内に配置された図形の座標を正しく計算できなくなるためです。

13 参考価格

昨今の組込み機器開発における各種原価や人件費の変動が予測しがたい状況を鑑み、Aeropoint も先ずは参考価格を提示させていただいております。

参考価格
https://www.cri-mw.co.jp/product/embedded/Aeropoint/GUI/price78k1e001358i.pdf

実売価格は日々変動しておりますので、Aeropoint のご利用を検討される場合につきましては、先ずは弊社までご相談いただければ幸いです。

お客様のご予算に合わせた金額及び、事業形態に合わせたお支払方法のご相談を承っております。

14 お問い合わせ

14.1 CRI・ミドルウェア

株式会社 ＣＲＩ・ミドルウェア
〒150-0002
東京都渋谷区渋谷 1-7-7 住友不動産青山通りビル 9 階
TEL 03-6418-7081
mail：support-emb@cri-mw.co.jp

15 改訂記録

Rev.	発行日	改訂内容
1.06	2021/03/15	Q&A に追記を行いました。
1.05	2021/03/08	Fit キットに関する情報を追記しました。
1.04	2020/08/28	シリアル通信 API を追加しました。それに伴い、シリアル通信を行うサンプル、シリアル通信の接続方法を追加しました。
1.03	2020/07/17	RX72N 専用 D-Amp 基板の取り付け方法を記載しました。 D-Amp 基板へのスピーカーの接続端子の説明を追加しました。 Welcome スライドを表示の手順にある PowerPoint®のファイル名を修正しました。 PC コネクターからのサウンド再生に音量を調節できる説明を追加しました。 UART コマンドに BGM、SE の音量変更を追加しました。 プリミティブ表示 API の説明、プリミティブ表示のサンプルプログラムを追加しました。 Aeropoint プログラムリファレンスに記載されている構造体、API の名前を変更しました。 構造体や API の名前の変更に合わせ、Aeropoint ランタイム、サンプルプログラムの説明も変更しました。
1.02	2020/07/08	SoundPlay サンプル、MoviePlay サンプルの表示文字を変更しました。
1.01	2020/06/29	Q&A を追加しました。
1.00	2020/06/25	初版発行