RX600 シリーズ
ダイレクトドライブ LCD デモンストレーション

R01AN0331JU0104
Rev.1.04
2016.01.06

はじめに

LCD ダイレクトドライブデモンストレーションの目的は、リアルタイム環境においてルネサス DirectLCD API（アプリケーションプログラムインタフェース）を使用して対話型 TFT-LCD パネルアプリケーションを作成する方法を示すことです。付属するワークスペースとソフトウェアは、ルネサス LCD ダイレクトドライブデモンストレーションプラットフォーム上で実行するサンプルコードとして提供されます。本アプリケーションノートでは、このサンプルコードのプログラム構造の詳細を説明しています。

本アプリケーションノートのユーザは、サンプルコードで使用される API の動作の詳細に関しては“DirectDriveLCDDesignGuide.pdf”および“GAPIUserManual.pdf”（サンプルコードワークスペース内にある）も参照してください。

このサンプルコードは FreeRTOS をリアルタイムオペレーティングシステムとして使用します。FreeRTOS の技術ドキュメントについては、www.freertos.org を参照してください。

対象デバイス

RX62N、RX63N

対象 LCD パネル

標準 TTLRGB、HSynch、VSynch、PixClk および DataEnable インタフェースを備えた LCD パネル

ダイレクトドライブ LCD ソリューションは高度な設定に対応することができ、さまざまなパネル製造業者の TFT-LCD パネルの入力信号を駆動する異なるタイミング構成を生成することができます。ダイレクトドライブ LCD ソリューションから生成される信号タイミングは、パネルの解像度、フレームバッファメモリ、希望するパネルリフレッシュレートおよびアニメーションレートの選択によって異なります。

ダイレクトドライブ LCD ソリューションは、LCDC 非搭載の MCU を用いて TFT-LCD パネル制御を行います。また、TFT-LCD パネルの制御をするために外部バス機能や MTU2 機能、EXDMAC 機能を使用します。そのため、各機能の設定内容によっては TFT-LCD パネルの仕様を満足しない場合があります。詳細はお近くの販社・特約店までお問い合わせください。

本サンプルでは、信号タイミングを構成するためのガイドラインと実例を提供しますが、特定の TFT-LCD パネルの AC タイミング仕様に適合することについて責任を負うものではありません。先の各機能の設定とともに、ダイレクトドライブ LCD ソリューションがパネルタイミングの制約に適合することに関しては TFT-LCD パネル製造業者にお確かめください。

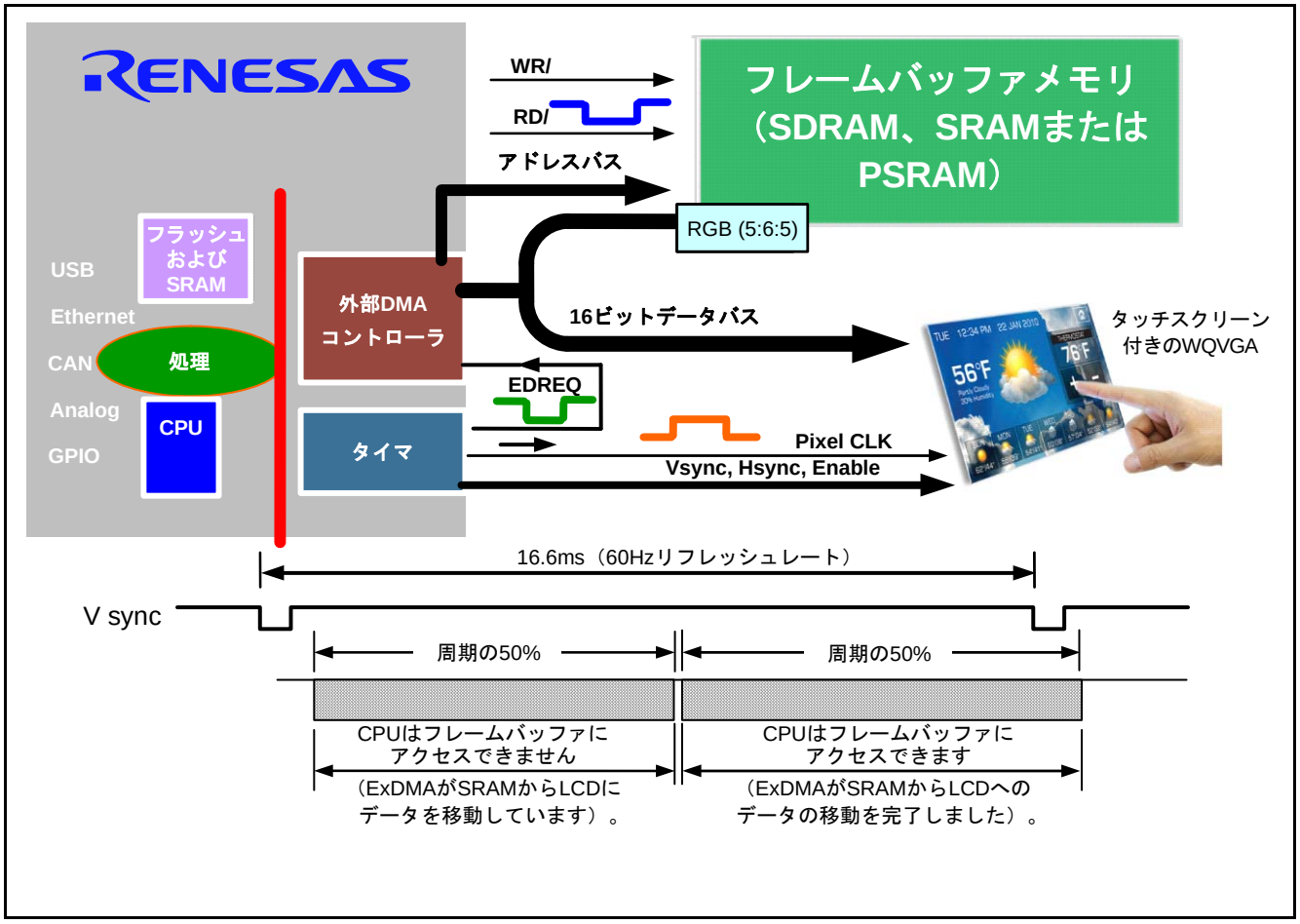
目次

1.　背景

1.1　LCD ダイレクトドライブとは何か

LCD ダイレクトドライブは、ルネサスマイクロコントローラのオンチップペリフェラルを介して TFTLCD パネルを制御することができます。これらのペリフェラルには、外部バスコントローラ（BSC）、外部 DMA コントローラ（ExDMAC）、タイマユニット（TPU または MTU）が含まれます。LCD ダイレクトドライブ API は、使用可能なプロセッサ帯域幅の 2%未満を使用して RAM フレームバッファの内容を LCD パネルに透過的に転送します（100MHz で動作する RX62N 上の 60Hz の WQVGA パネル）。

MCU はいつでもコードを実行し、内部フラッシュおよび RAM からデータにアクセスすることができます。外部 RAM フレームバッファの内容は、LCD 更新サイクルの垂直ブランキング部分で MCU によって更新することができます。MCU による外部バスアクセスの調整は、競合が生じないように API によって処理されます。



2. デモンストレーションアプリケーション

以下の説明は、ルネサス開発ツールについて十分な知識を持っていることを前提としています。

2.1 インストール

デモンストレーションアプリケーションをインストールするためには、対象プラットフォームに適合する、このアプリケーションノートに付属するインストールプログラムを実行します（たとえば、DirectLCD_RX62N_RSK.exe）。必要な HEW、コンパイラ、デバッガコンポーネントがすでにインストールされていることを確認します（DirectLCD インストーラは有効なバージョンを示します）。インストール時に、ワークスペースを解凍する場所が尋ねられます。問題がなければ、デフォルトの場所を使用します。

2.2 コンパイル

“DirectLCD.hws”をダブルクリックしてデモンストレーションワークスペースを開きます。

2.2.1 r_Packages.lib

顧客プロジェクトにおいて付属のコンポーネントの再利用を容易にするために、LCD ダイレクトドライブ API、グラフィックス API および RTOS はライブラリプロジェクト“r_Packages.lib”に分離されます。このライブラリは、ユーザアプリケーションプロジェクト（および DirectLCD デモプロジェクト）に簡単に含めることができます。すべてのソースコードがこのライブラリのために提供され、顧客プロジェクトの一部としてデバッグすることができます。

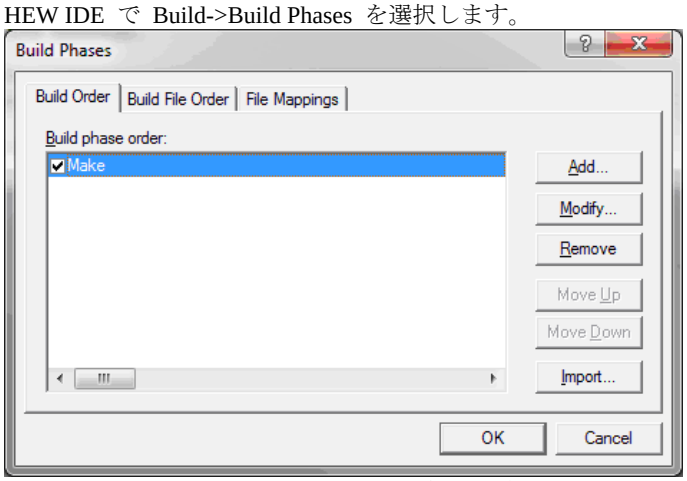
“r_Packages.lib”をビルドするには、プロジェクトを右クリックし、“SetasCurrentProject”を選択して、現在のプロジェクトを“r_Packages”に設定します。

64 ビット版 Windows の場合に一回実行する特別な手順。64 ビット版 WindowsOS におけるパスの問題に対応するために、HEW プログラムディレクトリに別のパスを設定する必要があります。

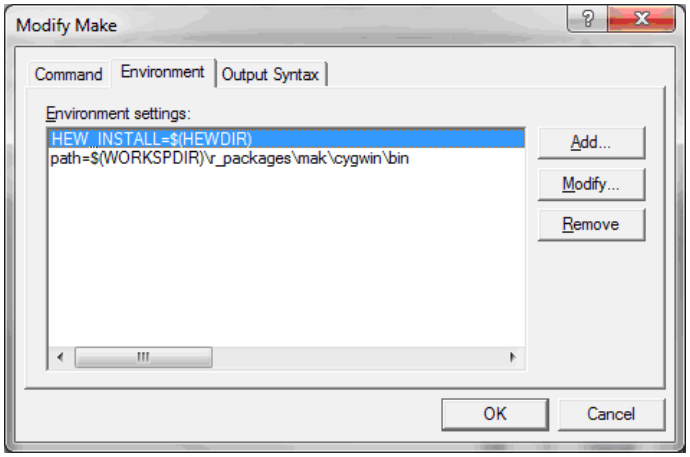
コマンドシェルから、HEW プログラムディレクトリへの別のパスを設定します。次の太字の部分をコマンドとして入力します。

```
C:¥Users¥jbrabend01>cd ¥
C:¥>md Renesas （このディレクトリがご使用のコンピュータ上にすでに存在していることがあります。この場合には「すでに存在している」というメッセージは無視してください。）
C:¥>cd Renesas
C:¥Renesas>mklink /D Hew "c:¥Program Files (x86)¥Renesas¥Hew"
symbolic link created for Hew <<===>> c:¥Program Files (x86)¥Renesas¥Hew
C:¥>exit
```

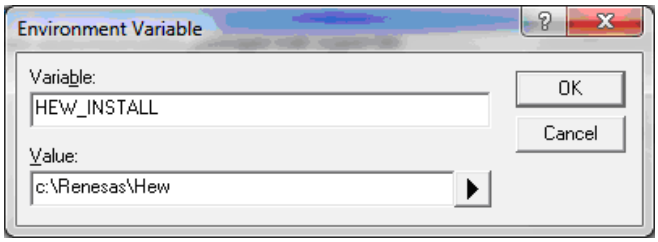
HEW IDE で Build->Build Phases を選択します。



“Make” をクリックして “Modify” を選択し、次に “Environment” タブを選択します。



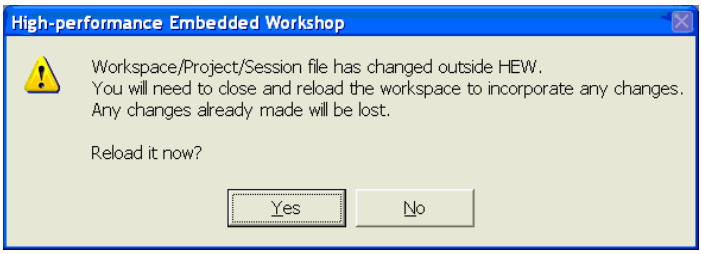
“HEW_INSTALL” をクリックして “Modify” を選択します。



“Value” の内容を HEW プログラムディレクトリに対して新たに作られた別のパスに変更します。“OK” を選択して、すべてのダイアログを終了します。

<F7>を押して、r_Packages.lib の “HWP” 構成をビルドします。ライブラリ以外に、ブラウズ可能なツリーも HEW プロジェクトウィンドウの下にビルドします。“HWP” 構成をビルドするときに通常は、以下のメッ

セージが表示されます（HEW プロジェクトファイルがビルドされるため）。“Yes”をクリックし、次に進みます。



ライブラリが一回ビルドされた後、“New”構成を使用して以降のビルドを実行することができます（HWP 構成はブラウズ可能なプロジェクトを初めてビルドするためにのみ必要です）。“r_Packages.lib”ライブラリは、構成ファイルが変更される場合にのみ再ビルドする必要があります。

2.2.2　DirectLCD.abs

“DirectLCD”プロジェクトを右クリックし、“SetasCurrentProject”を選択して、デモンストレーションアプリケーションプロジェクトを選択します。<F7>を押してプロジェクトをビルドします。

2.3　アプリケーションの実行

デバッガ（E1）を対象ハードウェアプラットフォーム（RX62NRSKLCD ダイレクトドライブ）に接続し、デバッグセッションを確立します。

以前にビルドした“DirectLCD.abs”を対象にダウンロードします。

“DirectLCD.abs”実行可能ファイルをビルドするだけでなく、ビルドプロセスはワークスペースの“Resources”ディレクトリに含まれるすべてのグラフィカルリソースを含む“Resources.bin”ファイルも作成します。このファイルはデモンストレーションアプリケーションで外部シリアルフラッシュに格納されます。このファイルの内容が変更されるたびに、対象 PCB のシリアルフラッシュに保存しなければなりません。これは、DirectLCD プロジェクトディレクトリに格納されている HEW スクリプトファイル“ResourceLoad.hdc”を実行して行われます。このデモンストレーションワークスペースでは、このスクリプトファイルは“Commandline”“Console”でプリロードされ、コンソールの“play”ボタン  を押すだけで実行することができます。このスクリプトを実行する前にデバッガ接続が確立されており、DirectLCD.abs ファイルがダウンロードされていることを確認します。

このスクリプトは“Resources.bin”の内容が変更された場合にのみ実行する必要があります。

“DirectLCD.abs”および“Resources.bin”ファイルが対象にダウンロードされたら、プロジェクトの実行で、デモンストレーション画面が表示されます。

2.3.1　リソースロード画面

電源を投入すると、シリアルフラッシュのリソース内容はランタイムアクセスのために外部 RAM に転送されます。この画面は（プログラミングスクリプトを実行するときに）リソースの読み出しと書き込みの状態を示します。“Resources.bin”ファイルは組み込み CRC 値を持ち、ファイル転送を検証することができます。この画面は、“eventmgr.c”で通常の画面処理を開始する前に生成されます。この画面で使用されるフォントは MCU 内部に格納され、アプリケーションとともにロードされます。その他のすべての画面リソースは、ロードされた“Resources.bin”からアクセスします。

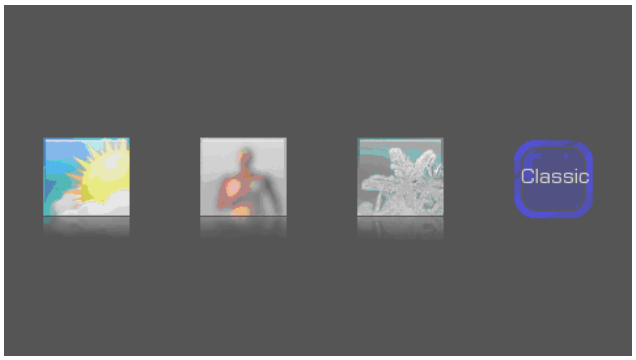
2.3.2　スプラッシュ画面

デモンストレーションはファイル“_SplashScreen.bmp”をロードし、ホーム画面に移行する前に 4 秒間この画面を表示します。ディスプレイにタッチするとこの画面をスキップすることができます。この画面は、“eventmgr.c”で通常の画面処理を開始する前に生成されます。

2.3.3　ホーム画面

フレームワークがリソースをロードすると、“ScreenHomeData”構造によって定義される画面の処理を開始します。このデモンストレーションでは、その構造は“ScreenHome.c”に定義されます。この構造は、_SCR_HM メモリセクションに格納されている画面オブジェクトのリストを参照します。このメモリセクション内のオブジェクトは、さまざまなファイル内で定義され、ビルド時にリンカによって収集され、完全なオブジェクトリストが作成されます。ファイルをプロジェクトにリンクするかどうかにより、オブジェクトの単純なインクルード／除外が可能です。

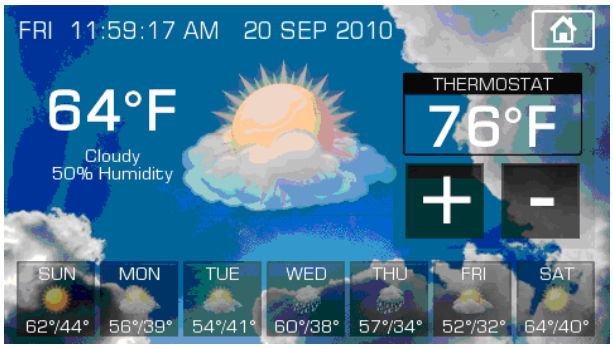
ここでは、4 つの画面ファイルに定義されるオブジェクトが表示されています。これらのオブジェクトを押すと、関連するコールバック関数が実行されます。この画面のオブジェクトの場合、コールバックは新しい画面イベントを送信します。このイベントにより、フレームワークはその画面ファイルと関連するデータ構造を処理します。



2.3.4　サーモスタット画面

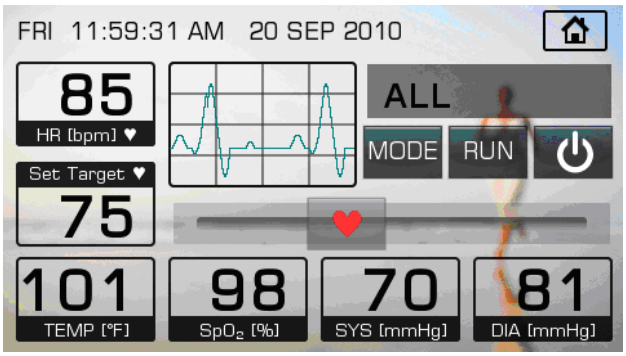
この画面はファイル“ScreenThermo.c”に定義されます。この画面には、アンチエイリアスされたフォント、UTF-8 デコード、アルファブレンディング、アニメーションなど、グラフィクス API の多くの項目が表示されています。この画面で注目すべき点は、画面下部のボタンが、画面形状、予測アイコン、予測テキストおよび曜日テキストを組み合わせることによりランタイムで構成することができることです（ソースコード関数“DayButtonDraw”を参照してください）。これらのコンポーネントは、条件が変更されるときに必要なグラフィックスを作成するために動的に選択することができます。

画面の右上隅にある“Home”ボタンを押すと、前の画面に切り替えるイベントが発生します。すべてのサブ画面で同様な動作が行われます。



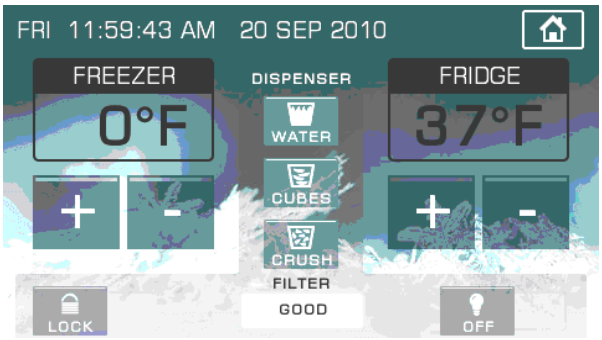
2.3.5　メディカル画面

この画面はファイル“ScreenMed.c”に定義されます。この画面では、表示されるグラフに注目してください。スライダにタッチしない場合は、反復波形が表示されます。ハートマークのスライダを押した場合、グラフにスライダ値が反映されます（ソースコード関数“drawGraph”を参照してください）。さらに、このコードはスライダの使い方を示します。スライダ、ボタンおよびその他のいくつかの動作は、ファイル“ScreenObjs.c”の“SliderHandler”および“ButtonHandler”関数で収集されます。これらの関数はイベント動作を処理し、オブジェクトを画面上に適切に描画し、オブジェクトに関する状態を維持します。これらの関数はグラフィクス API の一部ではありませんが、画面描画の必要性のために API を使用しています。



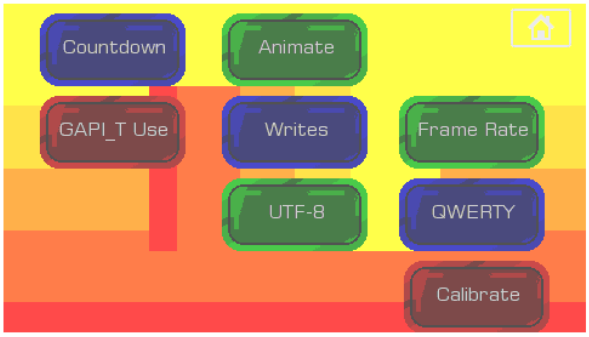
2.3.6　冷蔵庫画面

この画面はファイル“ScreenRefrig.c”に定義されます。この画面では、“DataBoxHandler”を介して温度値を使用して 2 つのデータボックスが保持され、“lock”および“light”ボタンが“SliderSwitchHandler”を使用して描画されます。これらのハンドラは両方とも“ScreenObjs.c”ファイルで提供されます。ユーザオブジェクトが作成されると、簡単に再利用可能な共通のハンドラを作成するのに適しています。これらの提供されるハンドラを検討すると、独自のハンドラを作成する方法がわかります。



2.3.7　クラシックホーム画面

この画面はファイル“ScreenHomeClassic.c”に定義されます。この画面のオブジェクトは、_SCR_HM_CLASSIC セクションでオブジェクトを割り当てるファイルからリンカによって収集されます。これにより、ホーム画面と同様なオブジェクトの単純なインクルードまたは除外を行うことができます。この画面のオブジェクトは、以下に説明するようにボタンと機能の基本的な動作を示します。



2.3.8　カウントダウン画面

この画面はファイル“ScreenCountdown.c”に定義されます。この画面は、作業フレームを複合し、表示フレームに転送して透過的なテキストを明確に表示する方法を示します。また、大きなテキストの場合、フォントのエイリアス除去によりクリアに表示することができます。その他の方法は、1 つのコールバックを使用してマルチボタンオブジェクト（テンキー）を処理することです。この画面は「画面タスク」スレッドを使用して実行中にタイマ値を更新します（マルチスレッドで GAPI が実行されます）。

2.3.9　アニメーション画面

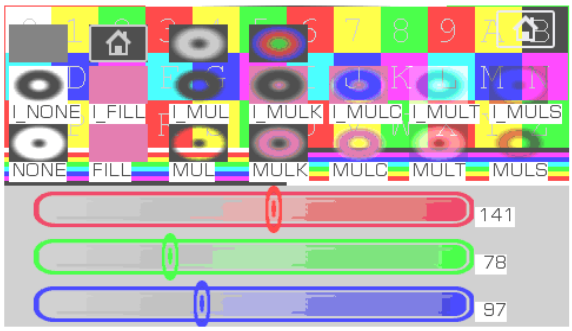
この画面はファイル“ScreenAnimate.c”に定義されます。この画面はオブジェクトをアニメーションにするために GAPI をどのように使用することができるかを示しています。ここでは 2 つの画面タスクを使用して 2 つのイメージを更新します（1 つは太陽、もう 1 つはスクロールするルネサスバナーです）。また、太陽のイメージは、イメージを移動するときに背景にブレンドするためにレンダリング中に使用する透明性（アルファ）チャネルを持っていることに注意してください。

2.3.10　GAPI_T 使用画面

この画面はファイル“ScreenSlider.c”に定義されます。この画面はスライダオブジェクトの複数のインスタンスの使い方と、GAPI コピーモードの動作を示しています。これらのスライダ値は R_GAPI_CopySub の“transparency”属性の色をコントロールしています。

ディミング、フェーデイング、シェーディングのデモンストレーションのため、最上段の最初のアイコンは背景領域を R_GAPI_CopySub ソースの出力領域として設定しています。

最上段の他のアイコンは個々のソースイメージを R_GAPI_CopySub ソースに設定しています。種々のモードで描かれた出力領域の状況はモードラベルと共に表示されています。

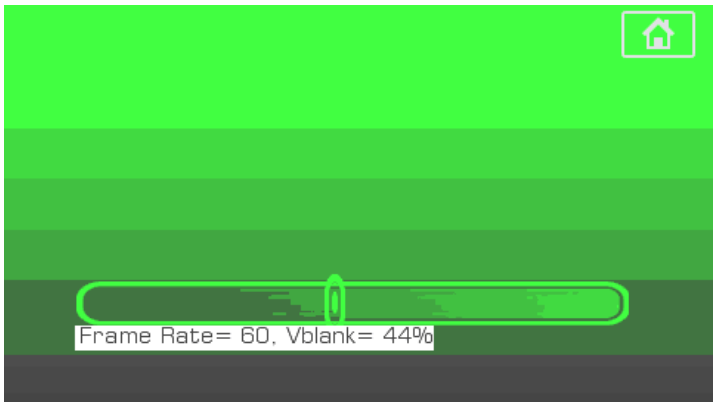


2.3.11　書き込み画面

この画面はファイル“ScreenWrites.c”に定義されます。この画面は、ソースイメージフォーマットに基づいてデータの画面を表示するのにかかる時間を測定します。必要な時間は、イメージサイズ、イメージタイプ、および使用可能な垂直ブランキング時間によって異なります。ここで、基本的にデータをコピーするだけの 16bpp イメージ（4 つのイメージで画面を塗りつぶすために 49ms）と非常に多くの処理を必要とするアルファチャネルを持つ 32bpp イメージ（15 のイメージで画面を塗りつぶすために 369ms）の書き込みの間の相違を確認することができます。いずれの場合も、60Hz リフレッシュおよび 44%垂直ブランキングでデータが書き込まれます。

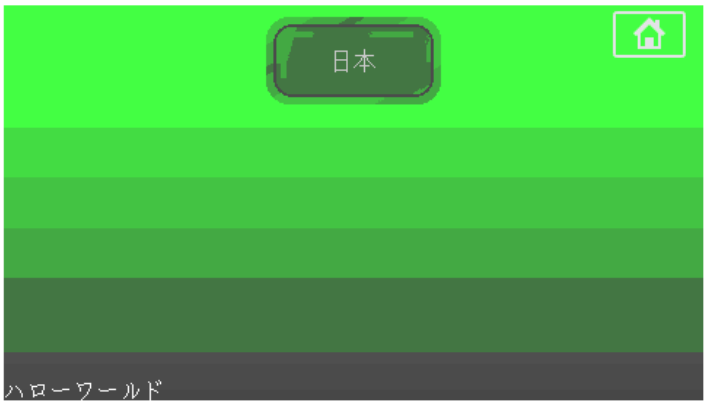
2.3.12　フレームレート画面

この画面はファイル“ScreenIO.c”で定義されます。この画面では、画面フレームリフレッシュレートと垂直ブランキングパーセンテージを調整することができます。ダイレクトドライブ API はこの属性を動的に調節することができます。この画面を使用すると、LCD パネル特性への影響、および書き込みテスト画面を使用して垂直ブランキングパーセンテージがパフォーマンスにどのように影響するかを知ることができます。



2.3.13　UTF-8 画面

この画面はファイル“ScreenUTF.c”に定義され、GAPI による UTF8 文字列処理の実例を示しています。



2.3.14　QWERTY 画面

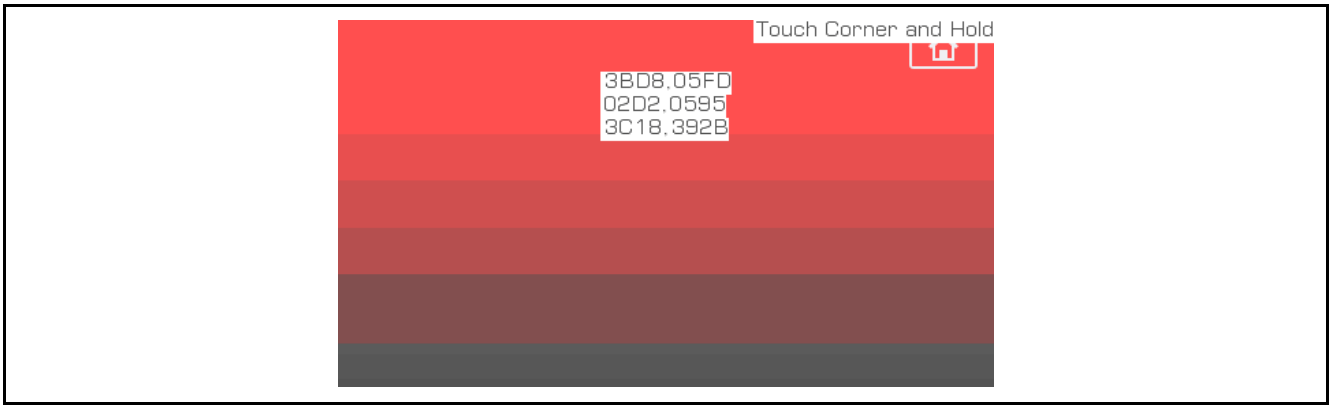
この画面はファイル“ScreenQWERTY.c”に定義され、仮想キーボードの例を示しています。

2.3.15　調整画面

この画面はファイル“ScreenCalibrate.c”に定義されます。この画面では、タッチスクリーンドライバによって使用される調整ポイントを調整することができます。この画面で決定される値は“TouchCalibration”データ構造に入力され、さらに、これらの値は調整手順の一部として内部または外部フラッシュに保存することができます。デモンストレーションコードでは、あらかじめ決定された値がこのデータ構造にロードされます。キットが付属する LCD パネルの場合、付属のタッチスクリーンオーバーレイの整合性は個別に調整する必要はありません。

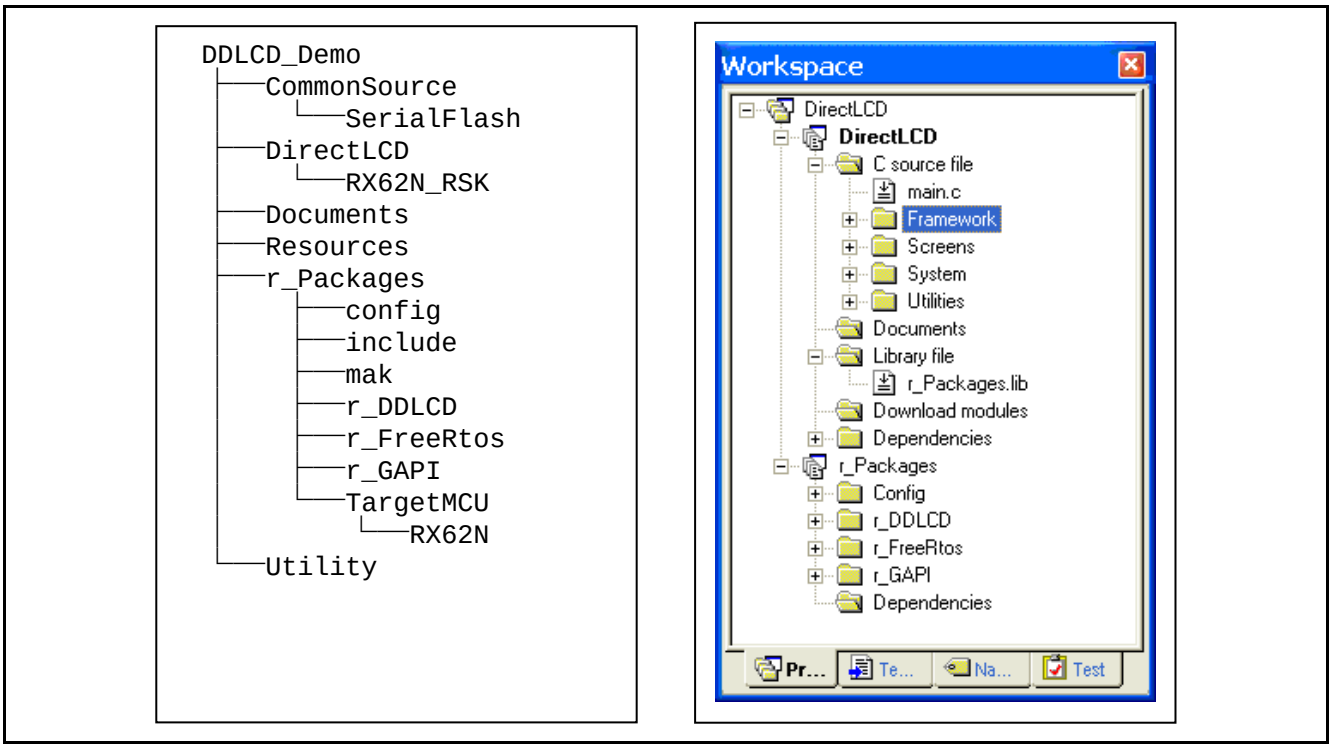
この画面に入ると、調整手順を完了せずに終了することはできません。調整が完了したら、画面へのすべてのタッチは対応する x、y 座標で示されます。手順の有効性を評価するために、ユーザは要求されていないタッチポイントを選択して、結果の動作を確認することができます。たとえば、調整時に右側ではなく左側にタッチすると、タッチ位置がこの軸に対して反転されます。初期調整ポイントはあらかじめ決定されているので、MCU をリセットするだけで適切な動作に戻ります。

正しくない調整定数がある未知のパネルの使用を容易にするため、PBC の SW1 を押すと、調整プロセスが開始します。



3.　コード構成

ソースコードディレクトリを図 1 の左側に、HEW ワークスペースを右側に示します。本アプリケーションノートを理解しやすくするために、プロジェクトを開いて、参照することをお勧めします。



3.1　ファイル説明

このサブディレクトリには、ダイレクトドライブプラットフォームによって共有されるすべての共通のソースファイルが含まれています。これらのファイルの説明を下表に示します。

3.1.1　初期化コード、プロジェクト、オブジェクトおよびその他のファイル

ファイル名	カテゴリ	ファイル説明	位置
resetprg.c	標準 C	リセットベクタの初期化	/DirectLCD
hwsetup _platform.c	標準 C	初期 MCU ハードウェア構成	/DirectLCD
.hwp,.hsf,*.h	HEW	HEW プロジェクトファイル	/DirectLCD
.	HEW	生成されたオブジェクトファイルおよびライブラリ	/DirectLCD/RX62N _RSK
.	文書	プロジェクト文書、API および図式を含む PDF ファイル	/Documents
.	リソース	デモンストレーションプロジェクト内で使用するリソースファイル（グラフィックス、フォント、オーディオなど）。このディレクトリの内容は、プロジェクトビルドプロセスによって 1 つの “resources.bin”ファイルにパッケージされます。このファイルは実行時の使用のためにシリアルフラッシュに書き込まれます。	/Resources
.	ユーティリティ	デモンストレーションで使用するファイルを作成するために使用する Windows アプリケーション	/Utility

3.1.2　アプリケーションデモコード

ファイル名	カテゴリ	ファイル説明	位置
main.c	標準 C	メインプログラム	/CommonSource
Global.h	デモヘッダ	グローバル定義およびマクロ	/CommonSource
LCD_Demo.h	デモヘッダ	LCD ダイレクトドライブデモ用のグラフィクスアプリケーション定義およびマクロ	/CommonSource
EventMgr.c	フレームワーク	タッチスクリーンおよび画面マネージャ機能を制御するためのフレームワークタスク	/CommonSource
Frames.c/ Frames.h	フレームワーク	デモによって割り当てられるメモリラスタを管理するためのルーチンおよび API	/CommonSource
ScreenMgr.c/ ScreenMgr.h	フレームワーク	画面の動作と切り替えを制御するためのルーチンおよび API	/CommonSource
ScreenObjs.c/ ScreenObjs.h	フレームワーク	ボタンやスライダなどの共通の画面オブジェクト動作を提供するルーチンおよび API	/CommonSource
Screen*.c	スクリーン	グラフィックスデモ画面コードを含むファイル 各画面は対応するファイルに含まれます。	/CommonSource
crc.c	ユーティリティ	CRC 値を生成するルーチン	/CommonSource
FindFile.c/ FindFile.h	ユーティリティ	リソースファイルの内容にアクセスするためのルーチンおよび API	/CommonSource
FlashSerial.h	ユーティリティ	シリアルフラッシュアクセスルーチンのための API	/CommonSource
FlashSerial.c	ユーティリティ	シリアルフラッシュアクセスルーチンのための API	/CommonSource/SerialFlash
iResources.h	ユーティリティ	常に内部フラッシュメモリに格納され、ブートプロセス時に使用することができるリソースイメージファイル	/CommonSource
Resources.c	ユーティリティ	シリアルフラッシュメモリにリソースファイルをロード、格納および検証するためのルーチン	/CommonSource
Simple_printf.c	ユーティリティ	単純化された（小さいメモリ、リエントラント）Printf ルーチン	/CommonSource
SPI_via<*port*>.c	ユーティリティ	シリアルペリフェラルに特有のシリアルフラッシュメモリアクセスルーチン	/CommonSource/SerialFlash
Timer_RTC.c/ Timer_RTC.h	ユーティリティ	ボタンやスライダなどの共通の画面オブジェクト動作を提供するルーチンおよび API	/CommonSource
TouchScreen.c/ TouchScreen.h	ユーティリティ	タッチスクリーンドライバ用のルーチンおよび API	/CommonSource

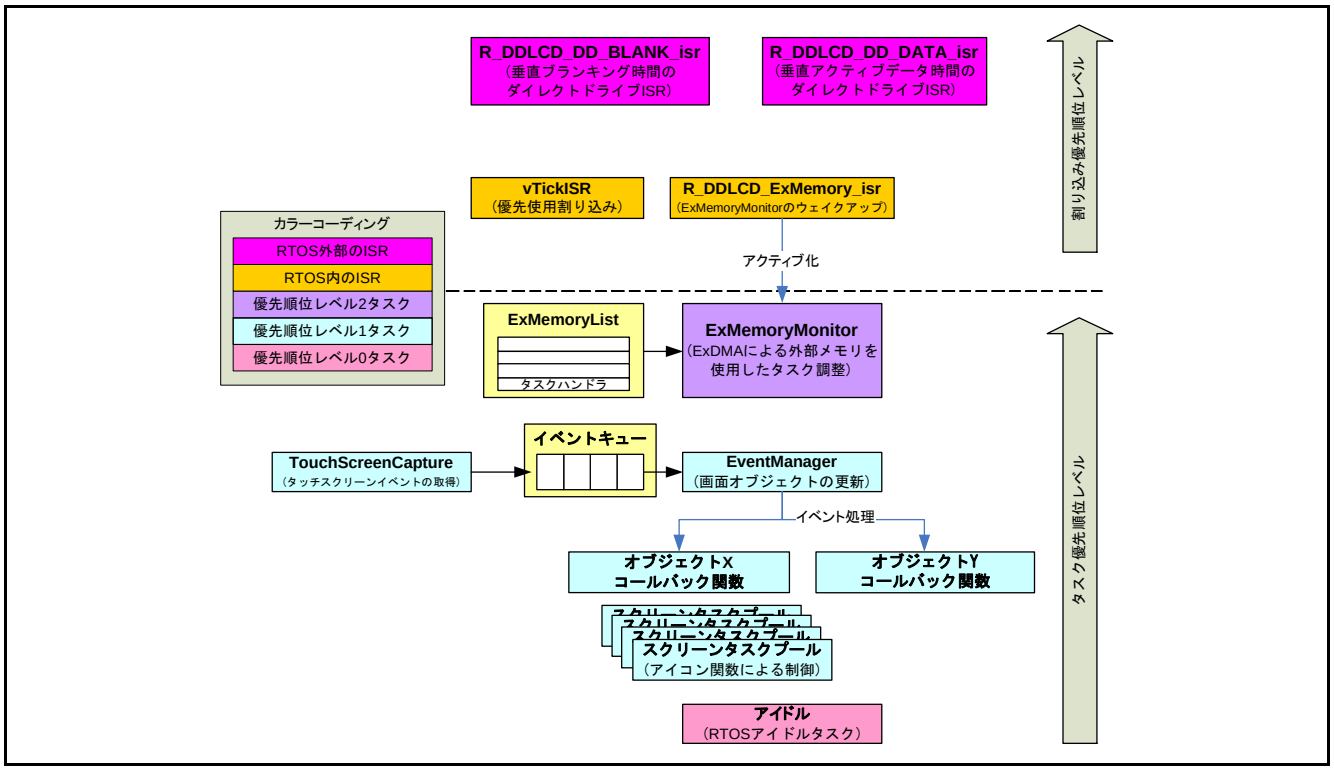
3.1.3 *r_Packages*

“r_Packages.lib”ファイルは DirectLCD プロジェクトに含まれます。このライブラリの内容を以下に説明します。

ファイル名	カテゴリ	ファイル説明	位置
*.h	構成	このディレクトリ内にあるファイルは、ビルドプロセス時にパッケージを構成するために使用します。詳細については、関連するパッケージ文書を参照してください。r_Packages.lib は、これらのファイルを変更した後に再ビルドする必要があります。 **このディレクトリのファイルは、ユーザが変更する唯一の r_Packages ファイルです。**	/r_Packages/config
*.h	API	このディレクトリ内のファイルはパッケージの API へアクセスします。このディレクトリ内のファイルは r_Packages.lib ビルドプロセス時に作成されることに注意してください。 **これはユーザがそのプロジェクトに含める必要がある唯一の r_Packages インクルードパスです。** **サポートされているすべての API 呼び出しは、このディレクトリにあるインクルードファイルを通じてアクセスされます。ライブラリへのアクセスは、これらのパブリック API を通じてのみサポートされます。**	/r_Packages/include
.	ビルド	r_Packages ライブラリビルド環境。この“make”ベースのビルド環境は r_Packages プロジェクトをビルドして呼び出されます。	/r_Packages/mak
.	DDLCD	ルネサスダイレクトドライブ LCD ライブラリパッケージ 詳細および API については、付属のダイレクトドライブユーザマニュアルを参照してください。	/r_Packages/r_DDLCD
.	GAPI	ルネサス GAPI グラフィクスライブラリパッケージ 詳細および API については、付属の GAPI ユーザマニュアルを参照してください。	/r_Packages/r_GAPI
.	RTOS	FreeRTOS ライブラリパッケージ 詳細および API については、FreeRTOS のウェブサイトを参照してください。	/r_Packages/r_FreeRTOS

4.　プログラム動作

以下の図に LCD ダイレクトドライブデモンストレーションプロジェクトで実行されている割り込みサービスルーチン（ISR）およびタスクを示します。



4.1　RTOS

LCD ダイレクトドライブデモンストレーションでは、タスクを管理し、外部バスにアクセスするために RTOS が必要です。このために FreeRTOS が選択されています。しかし、多くの RTOS はシステムの要求に同様に対応しています。別の RTOS への移植を容易にするために、RTOS のすべての使用法がコメント“RTOS_USAGE”を使用してコードに文書化されています。

4.2　割り込み

ドライバが初期化されると、LCD ダイレクトドライブ API は 2 つの“R_DDLCD_DDisrs”を実行します。これらの ISR は、外部フレーム RAM から LCD パネルにデータを転送する ExDMA およびタイマチャネルを制御します。パフォーマンスを最適化するために、1 つの ISR は垂直ブランキング時間にアクティブであり、もう 1 つの ISR はリフレッシュサイクルのデータ転送時にアクティブです。これらの ISR は、リフレッシュサイクルの水平期間（ライン）ごとに 1 回トリガされます。

垂直期間（フレーム）ごとに 2 回（データ転送が始まる前に 1 回、終了後に 1 回）、R_DDLCD_ExMemory_isr がトリガされ、“ExMemoryMonitor”タスクが起動されます。このタスクは外部バスへのソフトウェアのアクセスを調整します。（通常フレーム RAM を更新するために）外部バスを使用するタスクは、“R_DDLCD_ExMemoryAcquire”関数を呼び出してフレームワークに通知するために必要です。外部バスにアクセスするために複数のタスクを同時に登録することができます（設定可能な DD_EXMEMORY_MANAGER_MAX_TASKS 限界まで）。“ExMemoryMonitor”タスクは垂直データ転送の開始時にすべての登録されたタスクを中断し、垂直ブランキング期間の開始時に再開します。ある期間外部 RAM を使用しないタスクは、“R_DDLCD_ExMemoryRelease”を呼び出して自らの登録を抹消します。

登録しないで外部バスにアクセスした場合、結果として、外部バスの競合が生じます。これが発生した場合、MCU コアは ExDMA ペリフェラルが現在のブロック転送を完了するまで待ち状態になります。MCU コアのアクセスで ExDMA による転送開始が遅延する場合は、画面成果物が表示されることがあります。

図に示す最後の ISR は“vTickISR”であり、FreeRTOS スケジューラは現在実行中のタスクを優先使用し、次に実行するタスクを評価します。同じタスク優先順位レベルの複数のタスクを実行する準備ができている場合、各“vTickISR”後にラウンドロビン方式で順に実行されます。

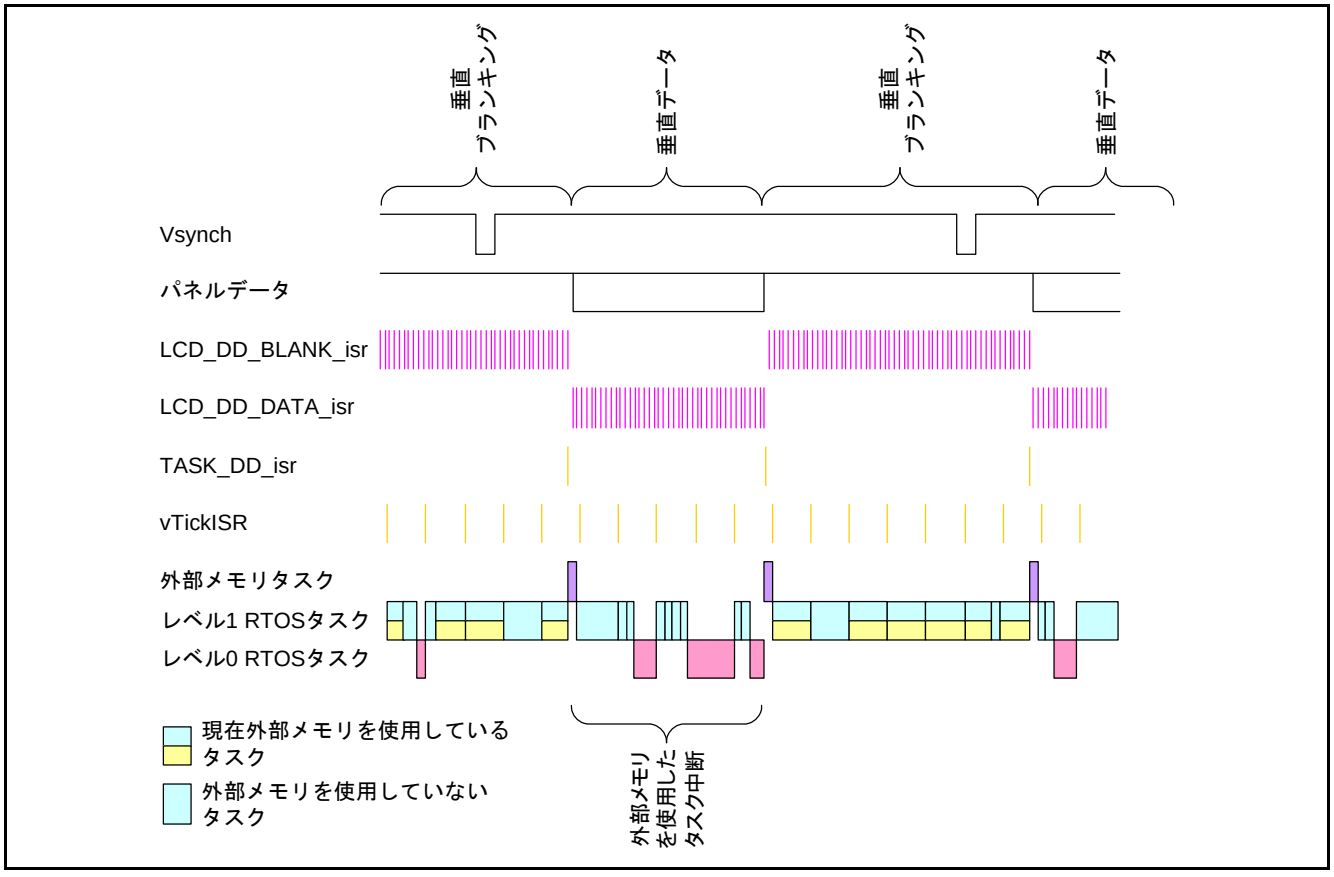
4.3　タスク

“TouchScreenCapture”タスクはタッチスクリーンオーバーレイからイベントを取得します。タッチイベントがこのタスクによって発生したと判定された場合、情報が EventQueue に追加されます。

EventQueue は、プッシュボタン、タイマまたはアプリケーションソースコードなどのその他のソースからイベントを受け入れるためにも使用します。

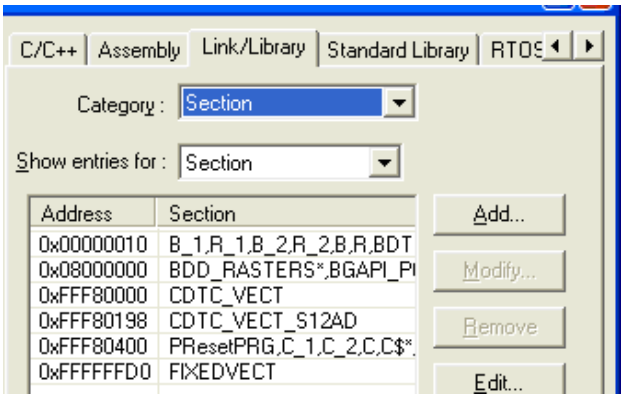
“EventManager”タスクは現在の画面のイベントをどのように処理するかを定めます。受け取ったすべてのイベントは、現在の画面の各オブジェクトと関連するコールバック関数に渡されます。オブジェクトコールバック関数は、イベントを処理するかどうかおよびイベントをどのように処理するかを決定しなければなりません。付属のデモンストレーションコードはイベント処理のさまざまな例を示します。

次の図は LCD パネル Vsynch 信号に関連する ISR の相互作用を示します。この図では、現在外部バスにアクセスしていると登録されているタスクが期間の「垂直データ」部分でどのように中断されるかも示します。



4.4　メモリ使用

LCD ダイレクトドライブデモンストレーションコードは、外部 RAM メモリ（フレームバッファ、リソースストレージおよびプール用）および内部 RAM およびフラッシュメモリ（プログラムストレージ、スタック、変数および定数用）の両方を使用します。メモリ内のこれらのリソース位置はツールチェーンリンカオプションによって決定されます。



RX600 ファミリでは、内部 RAM はメモリスペースの下部にマップされ、内部フラッシュはメモリスペースの上部にマップされ、外部 SDRAM は 0x08000000 に格納されます。

4.4.1　セクション

セクション名	タイプ	説明
B_*	RAM	スタートアップコードによってすべてゼロが書き込まれる BSS メモリ。ヒープはこのセクション内に割り当てられ、RTOS はヒープを広範囲に使用するので、“sbrk.h”の HEAPSIZE マクロによって有効なメモリが割り当てられます。
R_*	RAM	初期化された RAM メモリ。この領域は、D_*セクションに含まれるデータを使用してスタートアップコードによって初期化されます。
BDTC_TABLE	RAM	DTC ユニットによって使用されるデータストレージ構造
SU	RAM	ユーザスタックスペース。RTOS は個別のスタックを各タスクのヒープから割り当てるので、このセクションに最小スペースが必要です。
SI	RAM	割り込みスタックスペース。このメモリはスタック使用のために割り込みの処理時に使用されます。割り込みはネストすることができるので、十分なスペースを割り当てられるように注意しなければなりません。
BDD_RASTERS	外部 RAM	表示ラスタのためにダイレクトドライブ API によって割り当てられるスペース。デモンストレーションでは、バックグラウンド、作業および表示の 3 つのフレームを使用します。
BGAPI_POOL*	外部 RAM	ダイナミックメモリプールのために GAPI によって割り当てられるスペース
BResources	外部 RAM	Resources.bin ファイルのランタイムアクセスのために使用されるメモリ。このファイルは可変サイズなので、デモンストレーションコードでは“ScreenMgr.c”にこの使用のために 2MB が割り当てられます（これはデモンストレーションボードのシリアルフラッシュのサイズです）。
CDTC_VECT*	フラッシュ	これらのセクションは、DTC ペリフェラルによって使用されるベクタテーブルを割り当てます。このテーブルは 4KB 境界に位置合わせしなければならないので、フラッシュの先頭に格納されます。
PResetPrg	フラッシュ	リセットコードのためのプログラムセクション
C*	フラッシュ	デモンストレーションコードの定数変数のために割り当てられるセクション
C$VECT	フラッシュ	MCU の再配置可能なベクタテーブル
D*	フラッシュ	R*セクションの初期化データ
P	フラッシュ	アプリケーションと関連するプログラムメモリ。デモンストレーションアプリケーション全体では最大 40KB のコードスペースのみが必要であることに注意してください。
W	フラッシュ	スイッチテーブルで使用するために割り当てられる定数メモリ
C_SCR_HM*	フラッシュ	ホーム画面の内容を定義する定数構造
C_SCR_HM_CLASSIC*	フラッシュ	クラシックホーム画面の内容を定義する定数構造
FIXEDVECT	フラッシュ	リセットベクタを含む MCU 固定ベクタテーブル

5. リソースストレージアクセス

リソース（BMP、フォント、オーディオファイルなど）は MCU のリニアメモリに格納されたリソースイメージファイルからアクセスします。これらのリソースイメージファイルは、内部フラッシュまたは外部 RAM に格納することができます。外部 RAM を使用する場合は、リソースファイルは電源投入時にアプリケーションコードによって非リニアメモリ（シリアルフラッシュまたは SD カードなど）から転送されます。リソースイメージを外部 RAM からシリアルフラッシュに書き込む前、およびシリアルフラッシュから外部 RAM に読み出した後、リソースイメージの CRC 値が検証されます。

リソースイメージファイルはカスタムユーティリティ“ResourceGen.exe”によって作成されます。このコマンドラインアプリケーションは以下のようにオプション引数を取ります。

オプション	説明
-D<サブディレクトリ>	<サブディレクトリ>内のすべてのファイルを処理します。
-e	レコード名にファイル拡張子を含めます。
-q	抑止モードで実行します（最小出力）。
-b	バイナリデータファイルを生成します。
-c	C 言語のソースアレイ（コンマ区切りアレイ）を生成します。
-f	–i/–m オプションに 0xFF のみを含むレコードを抑制します。
-i[aaaaaaaa]	Intel16 進レコードフォーマットを生成します。[aaaaaaaa]オプションの 16 進アドレス再配置
-m[aaaaaaaa]	Motorola16 進レコードフォーマットを生成します。[aaaaaaaa]オプションの 16 進アドレス再配置

デモンストレーションアプリケーションコードは、HEW カスタムビルドフェーズ機能によりビルドするたびにこのユーティリティを実行します。このビルドフェーズのオプションは“-b-q-DResourcesResources.bin”です。これはリソースディレクトリの内容を取得し、Resources.bin ファイルに含めます。

以下の表にファイルヘッダリソースイメージファイルのフォーマットを示します。各エントリの長さは 32 バイトです。最初のレコードはリソースファイルの CRC およびサイズ情報を提供します。-e オプションを使用しない限り、レコード名に拡張子は含まれません（File_1.bmp は“File_1”になります）。

	レコード名（最大 24 文字）	位置（4 バイト）	サイズ（4 バイト）
0x000000	“BFS_Header”	CRC	リソースファイルサイズ
0x000020	“File_1”	File_1 オフセット位置	File_1 サイズ
0x000040	“File_2”	File_2 オフセット位置	File_2 サイズ
0x000060	…		
0x0xxxx0	0xFFFFFFFFFFFF （終端レコード）	0xFFFFFFFF	0xFFFFFFFF

アプリケーションコードは、“FileFind()”関数を使用してリソース名でイメージファイル内の情報にアクセスします。

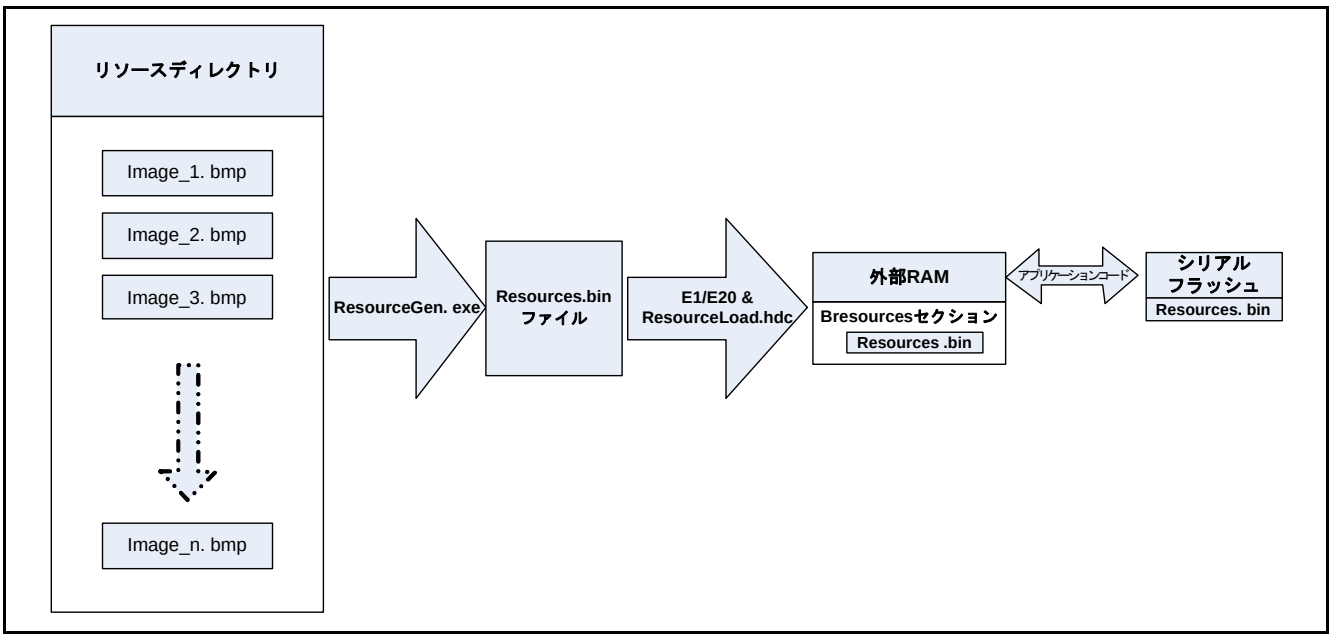
ダイレクトドライブアプリケーションデモンストレーションワークスペースは、HEW ビルドフェーズ（ResourceBuild）、ユーティリティ（Bin_to_mot.exe）、およびこのリソースファイル（Resources.bin）の作成およびロードを管理するスクリプト（ResourceLoad.hdc）を提供します。

このカスタムビルドフェーズは、プロジェクトをビルドするたびに実行されます。“Resources”ディレクトリのすべての内容は Resources.bin ファイルに含まれます。

リソースファイルが内部フラッシュに格納される場合は、“-c”オプションを使用してコンマ区切りアレイを生成して、通常プロジェクトに含めます。デモンストレーションコード内の“iResources.h”ファイルを参照してください。リソースファイルは定数アレイとしてプロジェクトダウンロードの一部となります。

リソースファイルがシリアルフラッシュに格納される場合は、HEW の“CommandLine”ウィンドウからプラットフォームに対して“ResourceLoad.hdc”スクリプトを実行してリソースファイルを手動でロードするこ

とができます。シリアルフラッシュへのプログラミングは、リソースを変更した場合にのみ実行する必要があります（アプリケーションコードはリソースファイルには含まれていません）。



ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com

お問合せ先
http://japan.renesas.com/contact/

改訂記録	RX600 シリーズ　アプリケーションノート ダイレクトドライブ LCD デモンストレーション

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2010.10.8	—	初版発行
1.01	2010.11.18	—	説明の明確化と誤字の訂正
1.02	2012.7.18	8,9,10	画面に関する説明の改訂
1.03	2013.3.11	4	64 ビット Windows における手順の追加
1.04	2016.1.6	1	LCD ソリューションご検討時の LCD 仕様に関する注意文追加

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は､切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。
注2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

RENESAS

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/