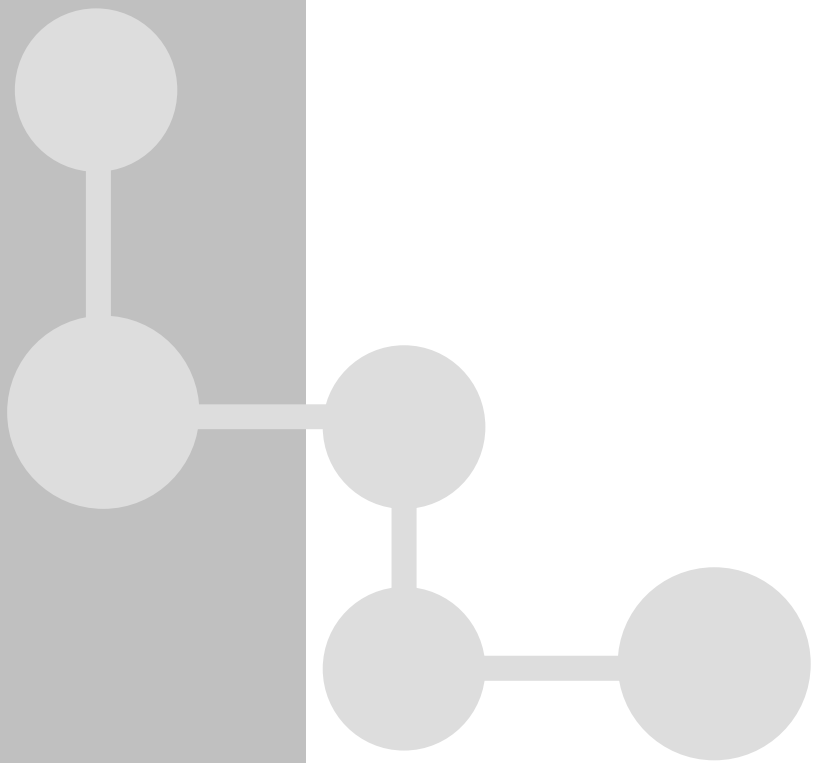
CASIO

CASSIOPEIA

DT-5100 シリーズ

エミュレータ開発解説書



Ver 3.00

変 更 履 歴

No	Revision	更新日	項	改訂内容
1	1.00	03/1/20	初版	初版発行
2	3.00	05/03/15	第2版	表紙をリニューアルしました。
3				
4				
5				
6				
7				
8				
9				
１０				
１１				
１２				
１３				
１４				
１５				
１６				
１７				
１８				
１９				
２０				
２１				
２２				
２３				
２４				
２５				

目次

1　概要

本解説書は DT-5100/870（WindowsCE .NET OS）用のエミュレータについて述べたものです。

1.1　エミュレータの機能

1. PC 上で DT-5100/870 用に開発したアプリケーションのデバッグおよび動作確認ができます。
2. エミュレータスキン上に配置したキーの押下がエミュレータ内アプリケーションに反映されます。
3. 以下のカシオライブラリを擬似的に実行できます。
   - システムライブラリ
   - OBR ライブラリ
   - 2D スキャナライブラリ（DT-870 のみ）
   - Bluetooth ライブラリ
   - BCD 演算ライブラリ
   - カメラライブラリ（DT-5100 のみ）

1.2　開発環境との連携

eMbededdC++4.0 用のエクスポート SDK をインストールすることにより eMbedded Visual C++4.0 と連携できます。

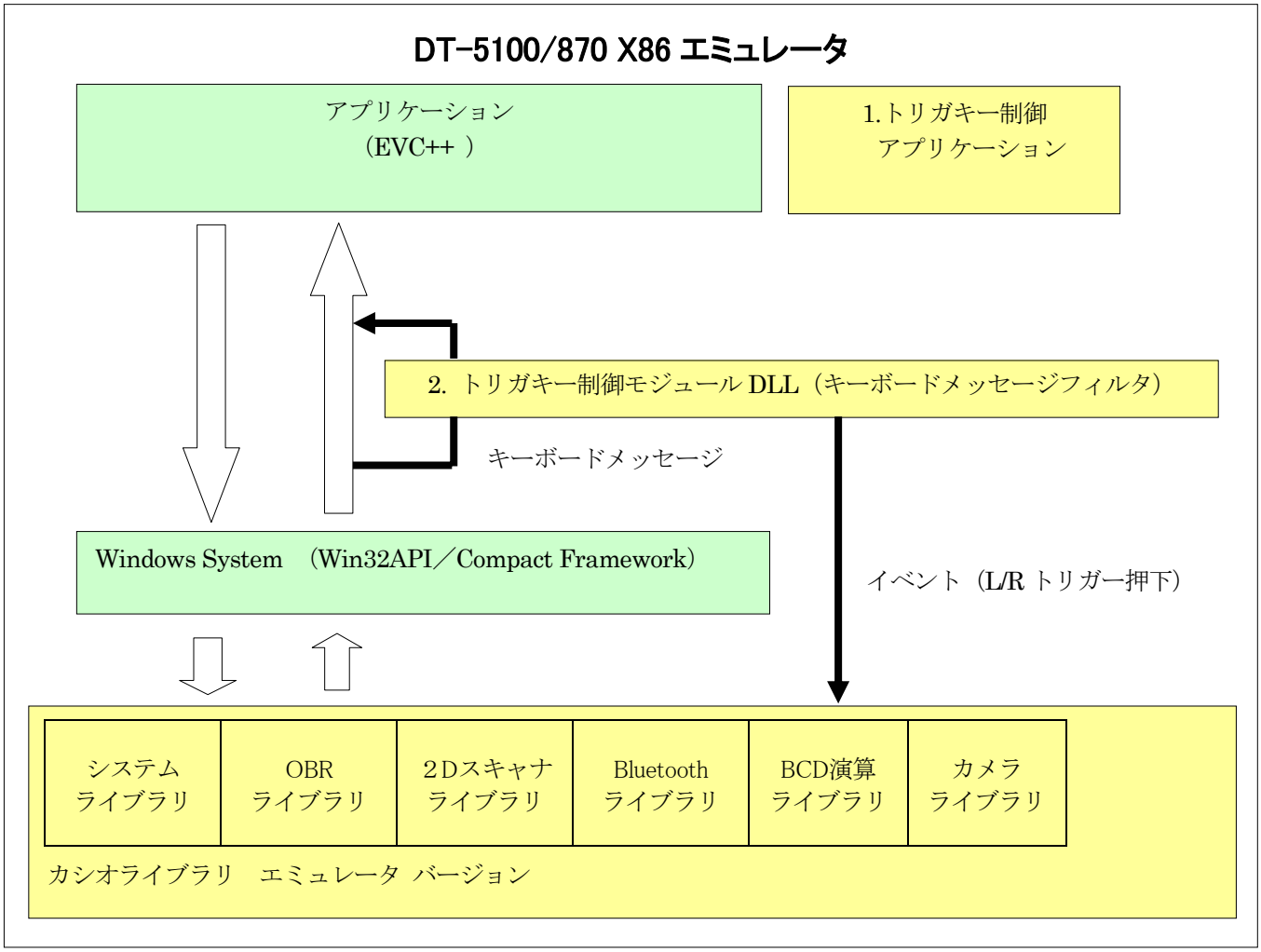
1.3　エミュレータスキン上のキー割り当て

エミュレータではスキン上のキーの押下に対して特定のコードを割り当てられます。
DT-5100/870 のスキン上のキーコード割り当てに関しては「3-4. スキン上のキーコード割り当て」を参照してください。

1.4　エミュレータのソフトウェア構成

DT-5100/870 エミュレータのソフトウェア構成を以下に示します。

1. トリガキーアプリケーション
2. トリガキー制御モジュール
3. カシオライブラリ エミュレータバージョン



3.

2　エミュレータ開発環境

2.1　開発環境

- Microsoft Platform Bulder 4.1（Windows CE.NET）
- Microsoft eMbedded Visual C++ 4.0 / SP1

2.2　ファイル構成

2.2.1 エミュレータソフトウェア

① DT-5100EM_SDKforEVC.msi　　eMbededd C++4.0 用エクスポート SDK
③ DT-870EM_SDKforEVC.msi　　eMbededd C++4.0 用エクスポート SDK

2.2.2 カシオライブラリエミュレータバージョン

以下のファイルは全て上記エミュレータソフトウェアに含まれています。このためインストールを実行すると開発環境内にインストールされます。

① システムライブラリ　　（DT-5100、DT-870）
　DirectClb.dll　　システムライブラリ本体
　ClbSys.h　　ヘッダファイル
　ClbSys.lib　　インポートライブラリ

② レーザライブラリ　　（DT-5100）
　LCUDRV.dll　　OBR ライブラリ本体
　LCUDRV.h　　ヘッダファイル
　LCUDRV.lib　　インポートライブラリ

③ ２D ライブラリ　　（DT-870）
　TDdecodece.dll　　２D スキャナライブラリ本体
　TDdecodece.dll　　ヘッダファイル
　TDdecodece.dll　　インポートライブラリ

④ Bluetooth ライブラリ　（DT-5100、DT-870）
　BTLib.dll　　Bluetooth ライブラリ本体
　BTLib.h　　ヘッダファイル
　BTLib.lib　　インポートライブラリ

⑤ BCB 演算ライブラリ　（DT-5100 国内モデルのみ）
　directbcd.dll　　BCD 演算ライブラリ本体
　ldtbcd.h　　ヘッダファイル
　ldtbcd.lib　　インポートライブラリ

⑥ カメラライブラリ　　（DT-5100）
　camera.dll　　カメラライブラリ本体
　camera.h　　ヘッダファイル
　camera.lib　　インポートライブラリ

2.2.3 トリガーキー制御ソフトウェア

以下のファイルは全て上記エミュレータソフトウェアに含まれています。このためインストールを実行すると開発環境内にインストールされます。

① pxemapl.exe トリガーキー制御モジュール登録／解除アプリリケーション
② pxemapl.lnk トリガーキー制御モジュール登録／解除アプリリケーションのショートカット
③ pxemdll.dll トリガーキー制御モジュール
④ pxemevt.h トリガーキー制御モジュール用ヘッダファイル

2.3 エミュレータの使用方法

2.3.1 EVC++4.0 環境

1） ミュレータエクスポート SDK を PC にインストールする。
2） EVC++4.0 でプロジェクトを作成し、アクティブな構成で「（Win32 （WCE emulator） Debug）を選択する。
3） 「ツール」→「Platform manager を構成」を選択し、
- Device にインストールした SDK のエミュレータを選択
- プロパティのトランスポートに TCP/IP を選択する。
- Start U p Server に Emulator Start Up Server を選択する。

1） プロジェクトをビルドする。
2） デバッグを開始すると作成したエミュレータが自動的に起動し、エミュレータ上でのデバッグができます。

2.3.2 トリガー制御アプリ pxemapl.exe の操作

トリガー制御用アプリは WindowsCE システムが起動すると自動的にトリガー制御モジュールをシステムに組み込みます。正常に組み込まれると、タスクトレイに（PXEM）アイコン表示されます。
アイコンをタップして［×］ボタンで登録を解除できます。（［OK］ボタンでトレイにもどります。）
再度組み込みたいときは、￥Windows￥pxemapl.exe を起動してください。

3　スキン上のトリガキーの処理

3.1　スキン上のトリガキーの処理

スキン上に配置されたキーを押下すると、エミュレータの定義ファイルで指定したキーコードを Windows のメッセージバッファにキューイングしますが、トリガキー（スキャナ）動作は押下したキーコードではなく、スキャンデータを発生する必要があります。
そのために、スキン上のキーの押下を検出する仕組みとしてシステムフックを使用します。
システムフックは、キーボードメッセージがメッセージキューにキューイングされるときに、SetWindowsHookEx（）関数の引数で指定した DLL および DLL 内の関数をコールしてもらいます。
フックモジュールは次の処理を行います。
トリガキーの押下を OBR および２D スキャナライブラリに通知します。
（通知を受け取ったエミュレータライブラリはスキャナデータを作成し、アプリケーションのからの読み出し関数で返却します。）

3.2　トリガー制御モジュールの自動実行

Windows システムが起動したときにシステムフックを登録する（SetWindowsHookEx（）を実行する）ために、登録用のアプリケーションを自動実行するように以下のようにしています。
エミュレータの NK.BIN の「Windows¥スタートアップ」フォルダに` システムフック登録用アプリケーションを格納するようにしています。

3.3 スキン上のキーコード割り当て

エミュレータスキン上のキーコード割り当てを以下に示します。
トリガーキーは他のキーと区別できるように Windows CE では使用しないと思われる F11,F12 のキーコードを割り当てます。

No.	スキン上のキー	割り当てるキーコード	備考
1	電源キー	SHUTDOWN	
2	左トリガー	Key_F11	“PxEmEvt0” 名前付きｲﾍﾞﾝﾄ発生
3	右トリガー	Key_F12	“PxEmEvt1” 名前付きｲﾍﾞﾝﾄ発生
4	↑	Key_Up	
5	←	Key_Left	
6	→	Key_Right	
7	↓	Key_Down	
8	Fn		
9	Program		
10	CLR	Key_Backspace	
11	Enter	KeyPad_Enter	
12	F1		
13	F2		
14	F3		
15	F4		
16	1	Key_1	
17	2	Key_2	
18	3	Key_3	
19	4	Key_4	
20	5	Key_5	
21	6	Key_6	
22	7	Key_7	
23	8	Key_8	
24	9	Key_9	
25	0	Key_0	

3.4 スキン上の左右トリガ ボタンの処理

システムフック処理は左右トリガ ボタンの押下（Key_F11、 Key_F12）を検出すると、次の名前のイベント（名前付き自動リセットイベント）をシグナル状態にします。
左トリガ ボタンのイベント名 ： **“PxEmEvt1”**
右トリガ ボタンのイベント名 ： **“PxEmEvt2”**

従って、左右トリガ ボタンの押下で値を返すような関数では、上記名前で作成したイベント配列を指定し、WaitForMultipleObject（） で待つことで実現できます。
以下はエミュレーター版 **TDWaitForDecode** （） の例です。

```
#include     "pxemevt.h"

Result_t TDWaitForDecode(
                                        DWORD dwTime,
                                        PTCHAR ptcMessage,
                                        PTCHAR ptcCodeID,
                                        PTCHAR ptcAimId,
                                        PTCHAR ptcSymModifier,
                                        PWORD pnLength,
                                        BOOL (*fpCallBack)(void)
                                  )
{
    DWORD       dwEvent;
    Result_t    rt = RESULT_ERR_NOIMAGE;

    // 名前付きイベント作成
    PxEmEvt_Init();           //  ・・・・・・・・・・・・・・・・・・・・・□

    dwEvent = WaitForMultipleObjects ( PXEVENT_NUM, hPxEmEvt, FALSE, dwTime); ・・□

    switch( dwEvent)
    {
    case WAIT_OBJECT_0:          // 左トリガーキー    ・・・・・・・・・・・・□
        wcscpy( ptcMessage, TEXT( "Scanned by LEFT triger"));
        *pnLength = wcslen( TEXT( "Scanned by LEFT triger"));
        rt = RESULT_SUCCESS;
        break;

    case WAIT_OBJECT_0+1:        // 右トリガーキー    ・・・・・・・・・・・・・□’
        wcscpy( ptcMessage, TEXT( "Scanned by RIGHT triger"));
        *pnLength = wcslen( TEXT( "Scanned by RIGHT triger"));
        rt = RESULT_SUCCESS;
        break;

    case WAIT_ABANDONED_0:          //   ABANDONED
    case WAIT_TIMEOUT:           //   WAIT_TIMEOUT
    case WAIT_FAILED:         //   WAIT_FAILED
    default:
        rt = RESULT_ERR_NOIMAGE;
        break;
    }

    // 名前付きイベント削除
    PxEmEvt_End();            // ・・・・・・・・・・・・・・・・・・・・・□

    return rt;
}
```

[説明]

この関数がアプリケーションから呼び出されると以下の処理を行います。

①の PxEmEvt_Init() で名前付きイベントを作成します。

②の WaitForMultipleObjects(PXEVENT_NUM, hPxEmEvt, FALSE, dwTime) でイベントを待ちます。

②, ③'で WaitForMultipleObjects を抜けた要因を判定します。

④の PxEmEvt_End() でイベントを削除します。

①、④の関数および、②のイベント数、イベント配列は以下のヘッダファイルで定義されていますので、このヘッダファイルをインクルードすることによっても利用できます。

インクルードファイル名： pxemevt.h

[pxemevt.h の内容]

```
/****************************************/
/*        イベント定義と初期化、開放関数 */
/****************************************/
#define PXEMEVT_T1      0
#define PXEMEVT_T2      1
#define PXEMENT_NUM  PXEMEVT_T2+1

static   HANDLE          hPxEmEvt[ PXEMEVT_NUM];        // イベントハンドル

void PxEmEvt_Init( void)
{
    int        i;
   TCHAR  szName[ 16];

   // 名前付きイベントを生成する（自動リセット、初期値非シグナル状態）
   for ( i=0; i < PXEMEVT_NUM; i++ ) {
        if( !hPxEmEvt[i]) {
             wsprintf( szName, TEXT("PxEmEvt%d"), i + 1 );
             hPxEmEvt[i] = CreateEvent( NULL, FALSE, FALSE, szName );
        }
   }
}

void PxEmEvt_End( void)
{
   int   i;

   // 名前付きイベントをクローズする
   for ( i=0; i < PXEMEVT_NUM; i++ ) {
        if ( hPxEmEvt[ i])
             CloseHandle( hPxEmEvt[i]);

        hPxEmEvt[ i] = 0;
   }
}
```

DT-5100

エミュレータ機能解説書

Ver3.00
発行元：カシオ計算機株式会社
〒162-8543
東京都渋谷区本町 1-6-2
ｼｽﾃﾑｿﾘｭｰｼｮﾝ営業統轄部
TEL:03-5334-4638