CASIO

DT-970SDK

DT-900/930/940
アプリケーション移行ガイド

このガイドは DT-970 のアプリケーション開発者向けの開発ガイドブックです。
DT-900/930/940 で利用していたアプリケーションを DT-970 に移行する上での手順、ツール等を記載します。

Ver. 1.05

ご注意

- このソフトウェアおよびマニュアルの、一部または全部を無断で使用、複製することはできません。
- このソフトウェアおよびマニュアルは、本製品の使用許諾契約書のもとでのみ使用することができます。
- このソフトウェアおよびマニュアルを運用した結果の影響については、一切の責任を負いかねますのでご了承ください。
- このソフトウェアの仕様、およびマニュアルに記載されている事柄は、将来予告なしに変更することがあります。
- このマニュアルの著作権はカシオ計算機株式会社に帰属します。
- 本書中に含まれている画面表示は、実際の画面とは若干異なる場合があります。予めご了承ください。

© 2013-2019 カシオ計算機株式会社

Microsoft, MS, ActiveSync, Active Desktop, Outlook, Windows, Windows NT, および Windows ロゴは、米国 Microsoft Corporation の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 社の製品は、OEM 各社に、Microsoft Corporation の 100%出資子会社である Microsoft Licensing, Inc.によりライセンス供与されています。

変更履歴

バージョン	変更日付	ページ	内容
1.00	2013.10		新規作成
1.01	2013.12	4-6, 38, 43-44	「TEC IrDA プリンタライブラリ」「シリアル通信制御/送受信切替ライブラリ」追加
		36	AppConverter.ini に "B211DEF.H" 記述追加
1.02	2014.03	4, 21, 29, 39	cu_xxxx() に関して、IrDA から Bluetooth への移行を削除
		13	通信速度の注意事項を追加
1.03	2014.07	55-56	DT-930 との相違に関する注意事項を付録に記載
		57	裏表紙を変更
1.04	2014.09	56	キー検出の処理の相違を記載
1.05	2019.03	55	sprintf 関数の相違を記載

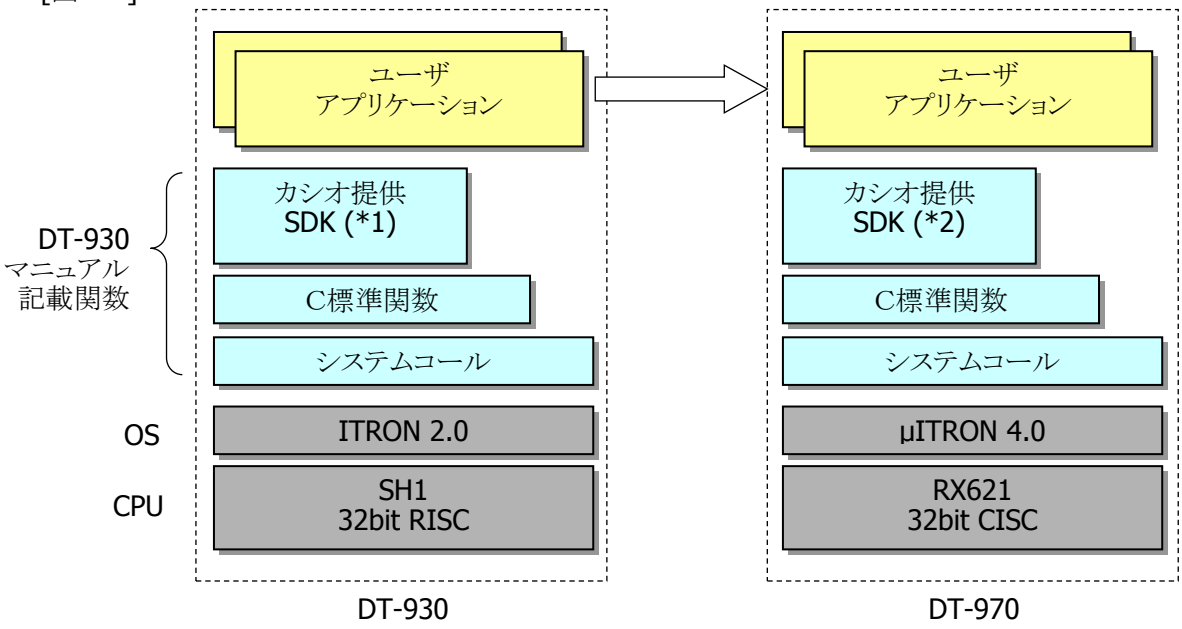
目次

1. 概要

1.1. 基本方針

DT-970 は、DT-930（本資料での DT-930 という記述は、DT-900/930/940 の総称とします）後継機種という位置づけのものです。
このため、基本方針として、DT-930 用のアプリケーションを、極力、そのままの形態で、DT-970 でも動作可能とします。

[図 1-1]

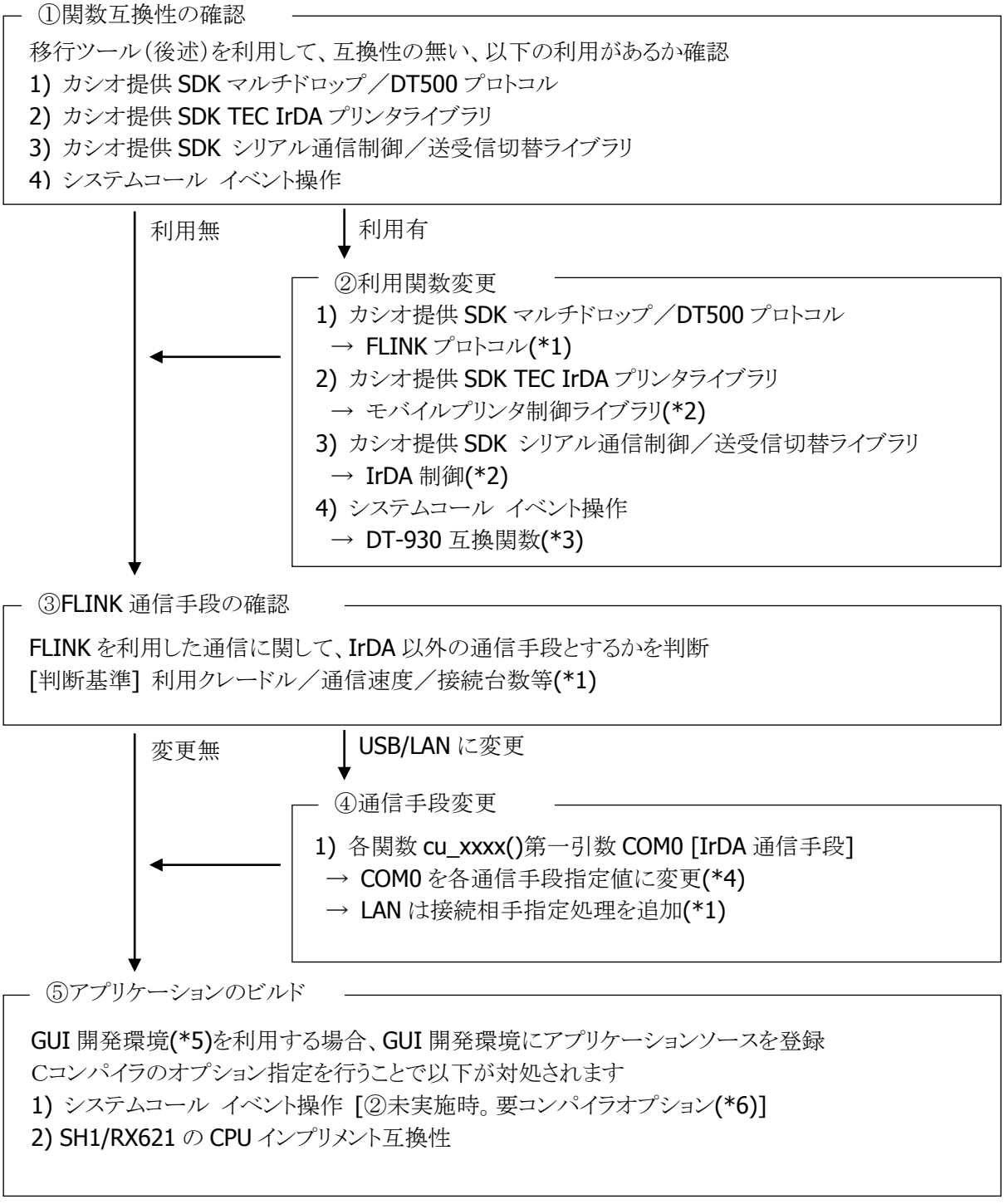


*1…DT-930 専用関数、拡張機能ライブラリの総称としての表記。
*2…DT-970 専用関数、拡張機能ライブラリの総称としての表記。

1.2. 目的

前述、基本方針に基づき、DT-930 用アプリケーションを DT-970 に速やかに移行する手順、及び、事前確認するツールを提供します。

1.3. 全体の流れ



*1…詳細は「2.通信処理の移行」参照
*2…仕様差異があるので、マニュアルを確認の上、ソース修正が必要
*3…移行ツールで自動置換が可能。ソース置換せず後述⑤-1)での対処も可能
*4…移行ツールで自動置換が可能。
*5…詳細は「アプリケーション開発ガイド」を参照。
*6…[-D DT930_CONVERT]を指定。

1.4. 関数互換性

DT-930 アプリケーションの DT-970 での互換性維持の前提としては、DT-930 マニュアルに記載されている関数で作成されているアプリケーションを対象とします。
上記、前提に基づいた、関数互換性を、表 1-1 に記載します。

[表 1-1]

No	分類		互換性	補足
1	カシオ提供 SDK	下記以外	○	
2		マルチドロップ DT500 プロトコル	▲*1	16 関数 →5 章 5-1
3		TEC IrDA プリンタライブラリ	▲*2	5 関数 →5 章 5-2
4		シリアル通信制御 送受信切替ライブラリ	▲*3	28 関数 →5 章 5-3
5	C標準関数	マニュアル記載	○	87 関数 →5 章 5-4
6		マニュアル未記載	対象外	
7	システムコール	マニュアル記載 ファイル／メモリ操作	○	6 関数 →5 章 5-5
8		マニュアル記載 イベント操作	△	3 関数 →5 章 5-6
9		マニュアル未記載	対象外	209 関数→5 章 5-7

○:そのままで互換性有
△:DT930 互換関数を提供
*1:FLINK への移行が必要
*2:モバイルプリンタ制御ライブラリへの移行が必要
*3:IrDA 制御への移行が必要

1) カシオ提供 SDK マルチドロップ／DT500 プロトコル

表 1-1 の No.2「マルチドロップ／DT500 プロトコル」については、DT-970 では機能自体が存在せず「FLINK プロトコル」への変更が必要となります。

「マルチドロップ」から「FLINK プロトコル」への移行等については「2.通信処理の移行」を参照して下さい。

2) カシオ提供 SDK TEC IrDA プリンタライブラリ

表 1-1 の No.3「TEC IrDA プリンタライブラリ」については、DT-970 では機能自体が存在せず「モバイルプリンタ制御ライブラリ」への変更が必要となります。

双方に機能差異があるので、マニュアルを確認の上、ソース修正が必要です。

3) カシオ提供 **SDK** シリアル通信制御／送受信切替ライブラリ

表 1-1 の No.4「シリアル通信制御／送受信切替ライブラリ」を実現する、カシオ IR インターフェイスは、DT-970 では機能自体が存在せず、IrCOMM プロトコルの「IrDA 制御（Ir_xxx）」への変更が必要となります。

カシオ IR インターフェイスと、IrCOMM プロトコルでは機能差異があるので、マニュアルを確認の上、ソース修正が必要です。

DT-970 には従来の「シリアル通信制御」と同一名称の 4 関数(c_open, c_close, c_dout, c_tmdin)が存在しますが、これは「USB HID 通信」を行うものです。

4) システムコール マニュアル記載イベント操作

表 1-1 の No.8「イベント操作」flg_sts, clr_flg, wai_flg については、ITRON バージョン差異により、関数 I/F が変更されています。
このため、これらの 3 関数については、カシオ提供 SDK 内に、DT-930 互換 I/F を持つ 3 関数(it2_flg_sts, it2_clr_flg, it2_wai_flg)を追加提供します。
DT-930 アプリケーションの DT-970 への移行は、以下の何れかの手法で対処可能とします。
ソース可読性／保守性から、基本的には①の手法を推奨します。

[表 1-2]

手法	内容
①	移行ツールで該当 3 関数を DT-930 互換関数に自動置換を実施
②	Cコンパイラオプションとして –D DT930_CONVERT を指定 → カシオ提供 SDK ヘッダのマクロ関数で互換関数に置換(*1)
③	移行ツールで該当 3 関数がソース上に存在するかチェック実施 → ユーザ判断で、以下の何れかの対処を実施 1) DT-930 互換関数への置換 2) DT-970 システムコールの I/F にあわせて変更

*1…具体的には下記のような記述で、コンパイル時プリプロセッサで置換を行います。

［リスト 1-1］

```
#ifdef DT930_CONVERT
#define flg_sts( ) it2_flg_sts( )
#define clr_flg( ) it2_clr_flg( )
#define wai_flg( ) it2_wai_flg( )
#endif
```

5) システムコール マニュアル未記載

表 1-1 の No.9「システムコール マニュアル未記載」は、移行ツールで、ソース上に該当関数の記述があるか否かの確認が可能です。
記述が存在した場合は、ITRON バージョン差異による対処が必要となります。

1.5. CPU インプリメント互換性

DT-930 搭載 CPU（日立 SH1）は、ビックエンディアンでした。
DT-970 搭載 CPU（ルネサス RX621）は、バイエンディアン CPU となっており、データ配置をビックエンディアンで動作させることが可能で、これにより、アプリケーションソース内各種データ操作（short を char にキャストする等）における互換性を維持します。

上記、エンディアン選択等、RX621 インプリメントを SH1 相当とするために、Cコンパイラに対しては、下記のオプション指定が必須となります。（デバイス制御ライブラリ等も、これらのオプションを指定してビルドしています。このため、ユーザアプリケーションも、これらのオプション指定が必要となります）

[表 1-3]

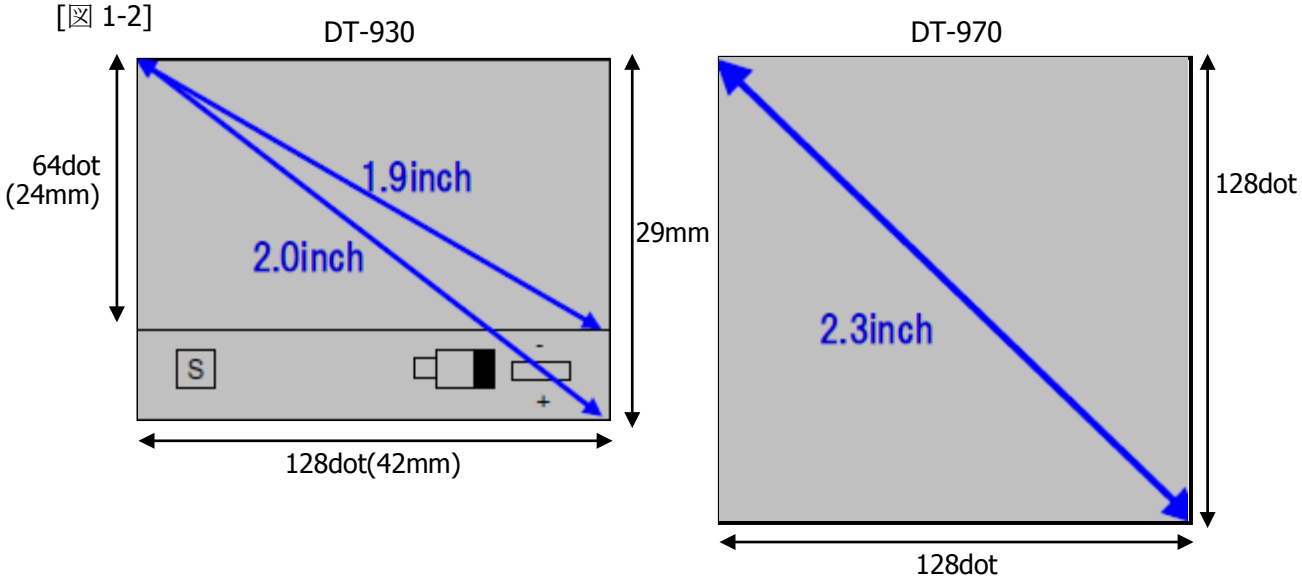
No	分類	記述	補足
1	エンディアン選択	-endian=big	big endian とする
2	ビットフィールド	-bit_order=left	上位ビットから割付
3	符号指定無 ビットフィールド型	-signed_bitfield	signed として扱う
4	符号指定無 char 型	-signed_char	signed char として扱う
5	double 型サイズ	-dbl_size=8	8byte とする

実際にアプリケーションをビルドする上では、上記[表 1-3]に記載した、互換維持のためのオプション以外に「セクション指定」等のオプション指定も必要となります。
これらについては、別途、アプリケーション開発ガイドに明記します。

1.6. 画面表示互換性

DT-930 と DT-970 では画面表示エリアのサイズが異なります。

[図 1-2]

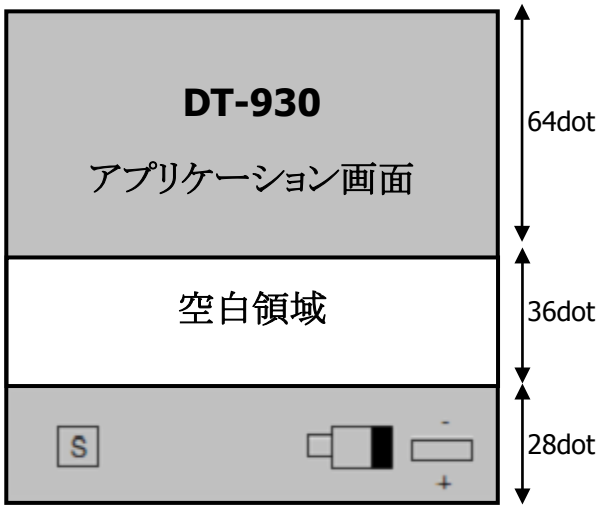


このため、DT-930 アプリケーションを、DT-970 に単純移行した場合は、リンクする ap_start オブジェクト (ap_start1.obj/ap_start2.obj)を選択することで、以下の何れかの表示が可能です。

[表 1-4]

No	リンクするオブジェクト	互換表示形式
1	ap_start1.obj	DT-930 アプリケーションを上部に寄せて表示して、下部 28dot にアイコンを表示[図 1-3]
2	ap_start2.obj	DT-930 アプリケーション画面の文字表示を縦 1.5 倍で表示

[図 1-3]



2. 通信処理の移行

2.1. 概要

本資料としては、まず、DT-930 におけるシステム構成パターンと、DT-970 におけるシステム構成パターンの対比を記載します。
このシステム構成パターンをベースとして、現在システムを DT-970 に置き換える際のシステム構成を御検討下さい。

次に、DT-970 で未サポートとなったマルチドロップを利用しているアプリケーションについて、FLINK へ移行するための技術情報を説明します。

最後に FLINK 利用アプリケーションについて、追加となった通信手段 Bluetooth/USB/LAN へ移行するための技術情報を説明します。

2.2. DT-930 システム構成パターン

[表 2-1]

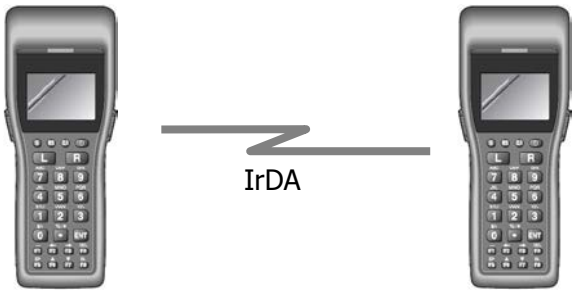
No	構成	通信プロトコル	通信手段	通信速度	接続台数
1	DT-930 端末間	FLINK	IrDA	4Mbps	1 台同士
2	IrDA-USB クレードル	FLINK	IrDA	4Mbps	8 台
3	サテライト IO ボックス	FLINK	IrDA	115Kbps	8 台
4	ベーシック IO ボックス	マルチドロップ	IrDA	19.2Kbps(*1)	8 台

*1…物理的には 115K だが、マルチドロップは 19.2K としている。

1) DT-930 端末間

DT-930 端末間を IrDA で通信する形態で、FLINK で利用可能でした。

DT-970 では、FLINK を利用した IrDA/Bluetooth 通信が可能ですが、IrDA 通信速度は 4Mbps から 115Kbps となります。

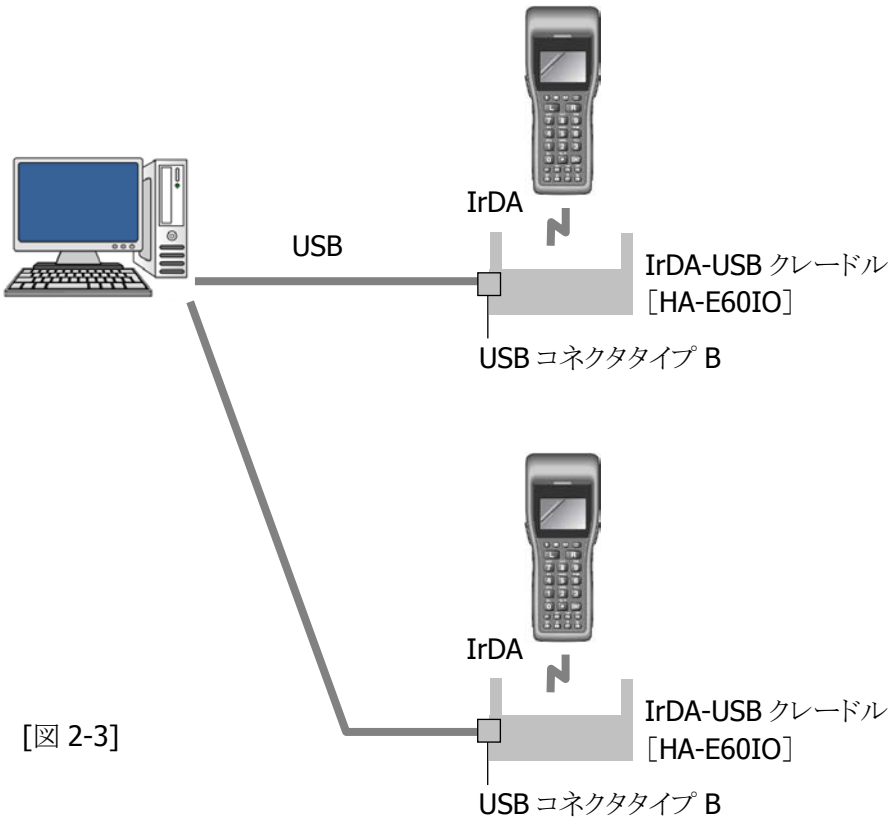


[図 2-1]

2) IrDA-USB クレードル[HA-E60IO]利用

PC に対して、IrDA-USB クレードルで最大 8 台接続して通信する形態で、FLINK で利用可能でした。

DT-970 では、クレードルアタッチメント[HA-N64AT]を利用することで、IrDA-USB クレードルをそのまま利用することが可能ですが、IrDA 通信速度は 4Mbps から 115Kbps となります。
IrDA-USB クレードル以外の接続形態としては、USB ケーブルで直結、及び、USB クレードル、LAN クレードル等で接続する構成が可能となります。(各構成で、PC に対しての DT-970 接続台数／通信手段／通信速度は異なります)
選択する構成によっては、通信手段を IrDA から USB/LAN に変更する必要があります。

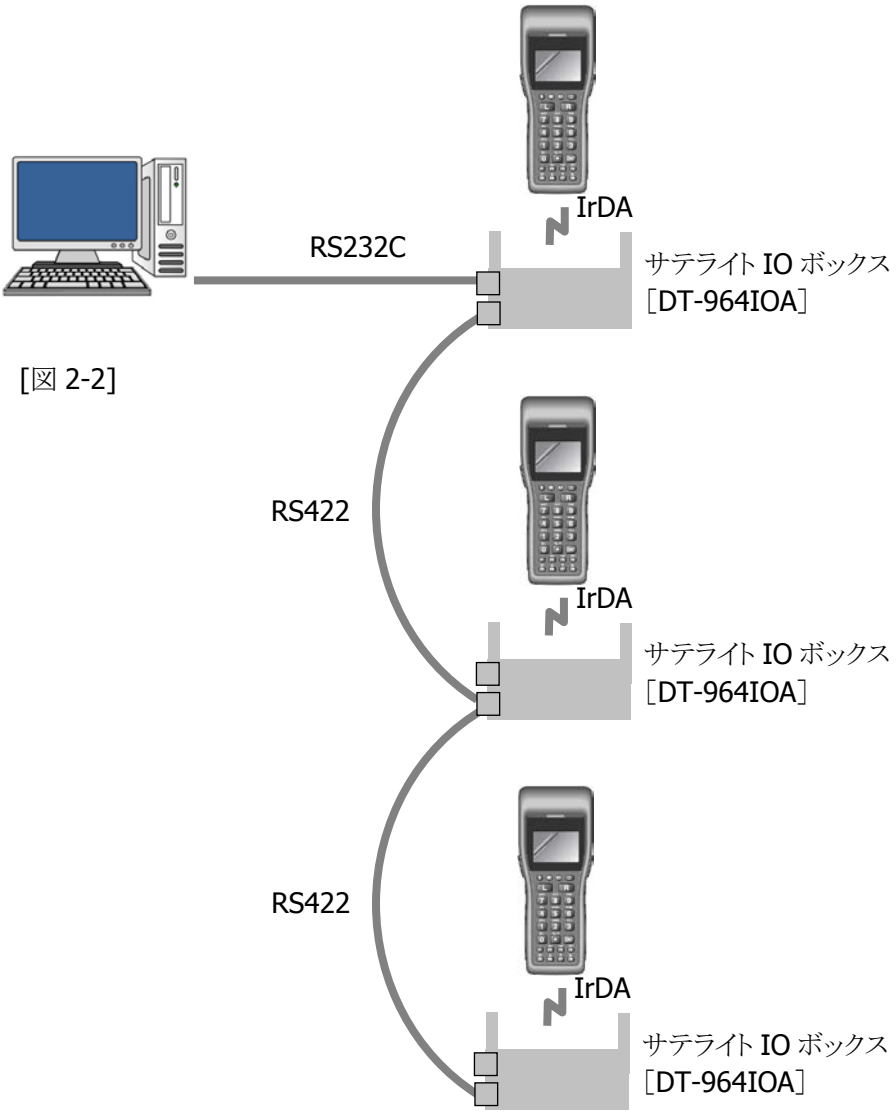


[図 2-3]

3) サテライト IO ボックス[DT-964IOA]利用

PC に対して、サテライト IO ボックスを最大 8 台連鎖接続して通信する形態で、FLINK で利用可能でした。

DT-970 では、本サテライト IO ボックスを利用することはできません。
DT-970 では、USB ケーブルで直結、及び、USB クレードル、LAN クレードル等で接続する構成が可能となります。(各構成で、PC に対しての DT-970 接続台数／通信手段／通信速度は異なります)
選択する構成によっては、通信手段を IrDA から USB/LAN に変更する必要があります。

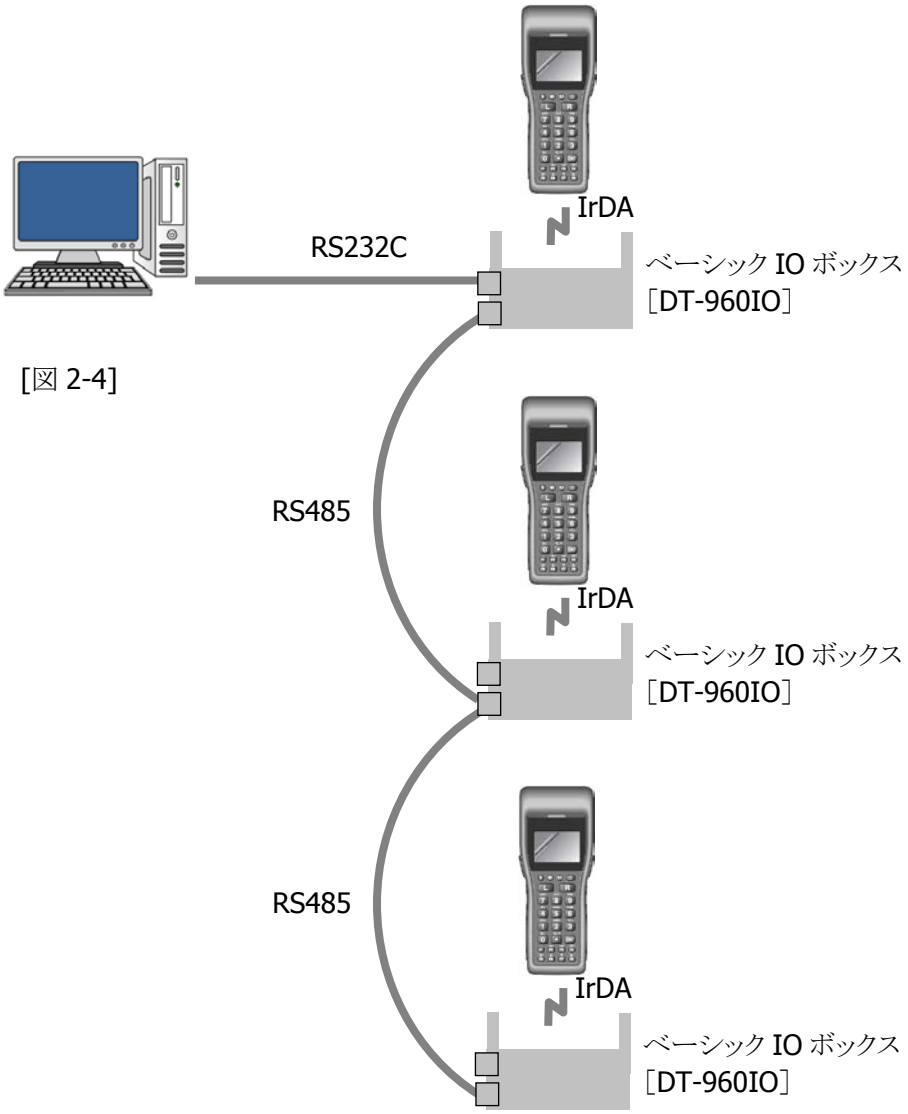


[図 2-2]

4) ベーシック IO ボックス[DT-960IO]利用

PC に対して、ベーシック IO ボックスを最大 8 台連鎖接続して通信する形態で、マルチドロップで利用可能でした。
マルチドロップを利用することで、PC 側から DT-930 端末 ID を識別し、任意の DT-930 毎にファイル転送を行うことができました。

DT-970 では、本ベーシック IO ボックスを利用することはできません。
DT-970 では、USB ケーブルで直結、及び、USB クレードル、LAN クレードル等で接続する構成が可能となります。(各構成で、PC に対しての DT-970 接続台数／通信手段／通信速度は異なります)
DT-970 では、マルチドロップは利用できないため FLINK へのソース改編が必要です。端末 ID を識別したファイル転送を行う必要がある場合は、LAN クレードルを利用した LAN 通信手段とすることで、同様な機能を実現することは可能です。(詳細は後述)



[図 2-4]

2.3. DT-970 システム構成パターン

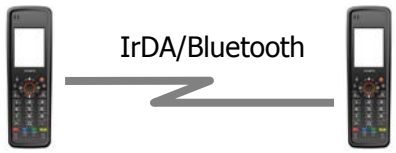
[表 2-2]

No	構成	通信プロトコル	通信手段	通信速度	接続台数
1	DT-970 端末間	FLINK	IrDA	115Kbps(*1)	1 台同士
2		FLINK	Bluetooth	－	1 台同士
3	USB ケーブル直結	FLINK	USB	－	1 台同士
4	USB クレードル	FLINK	USB	－	1 台同士
5	LAN クレードル	FLINK	USB	－	1 台同士
6		FLINK	LAN	－	－
7	IrDA-USB クレードル クレードルアタッチメント	FLINK	IrDA	115Kbps(*1)	8 台

*1…DT-930 と速度が異なります。

1) DT-970 端末間

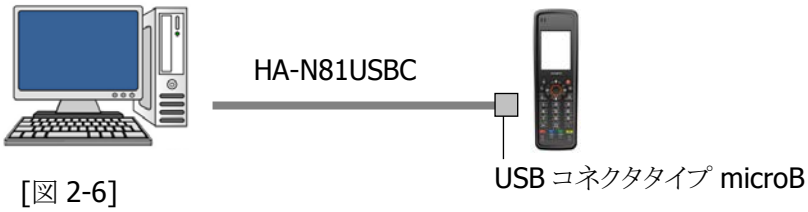
DT-970 端末間を IrDA/Bluetooth で通信する形態で、FLINK で利用可能です。



[図 2-5]

2) USB ケーブル[HA-N81USBC]で直結

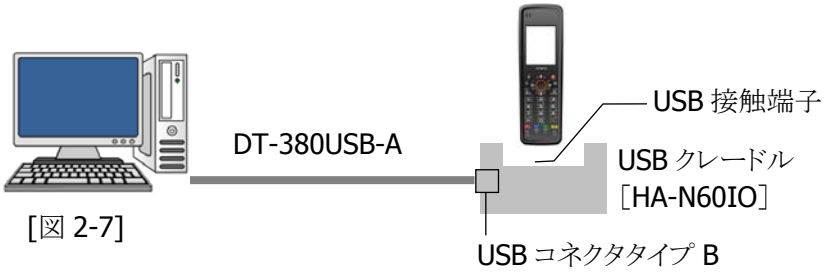
PC に対して、USB ケーブルで1台同士を接続して通信する形態で、FLINK で利用可能です。



[図 2-6]

3) USB クレードル[HA-N60IO]

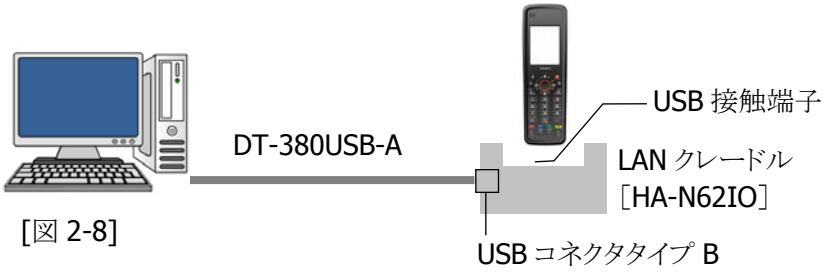
PC に対して、USB クレードルで1台同士を接続して通信する形態で、FLINK で利用可能です。



[図 2-7]

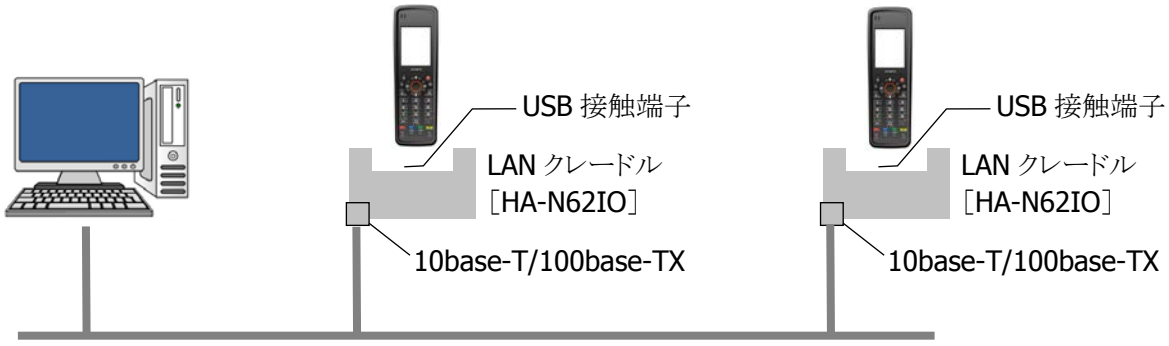
4) LAN クレードル[HA-N62IO]

PC に対して、LAN クレードルで、USB もしくは LAN で接続する形態で、FLINK で利用可能です。USB の場合は、1台同士を接続する形態となります。



[図 2-8]

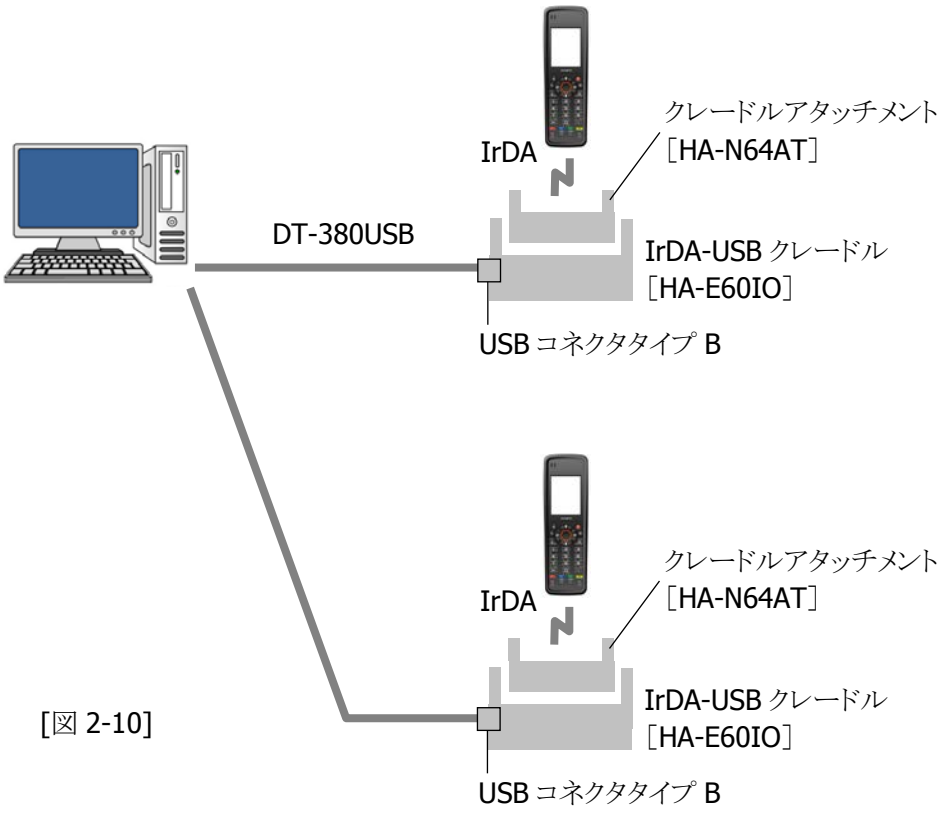
もしくは



[図 2-9]

5) IrDA-USB クレードル［HA-E60IO］、クレードルアタッチメント［HA-N64AT］で接続

DT-930 で利用していた IrDA-USB クレードルに対して、クレードルアタッチメントを追加することで、DT-970 でも利用可能とした形態ですが、IrDA 通信速度は 4Mbps から 115Kbps となります。
DT-930 と同様に、PC に対して最大 8 台接続可能な形態で、FLINK で利用可能です。



［図 2-10］

2.4. マルチドロップから FLINK への移行

1) 端末 ID 識別によるファイル送信

LAN クレードルを用いて、LAN 通信手段で接続する場合、IP アドレスで相手を特定することが可能です。

LMWIN 側で、IP アドレス毎にスクリプトを用意することで、任意の相手毎に、特定のファイルを転送することが可能となります。

IP アドレスについては

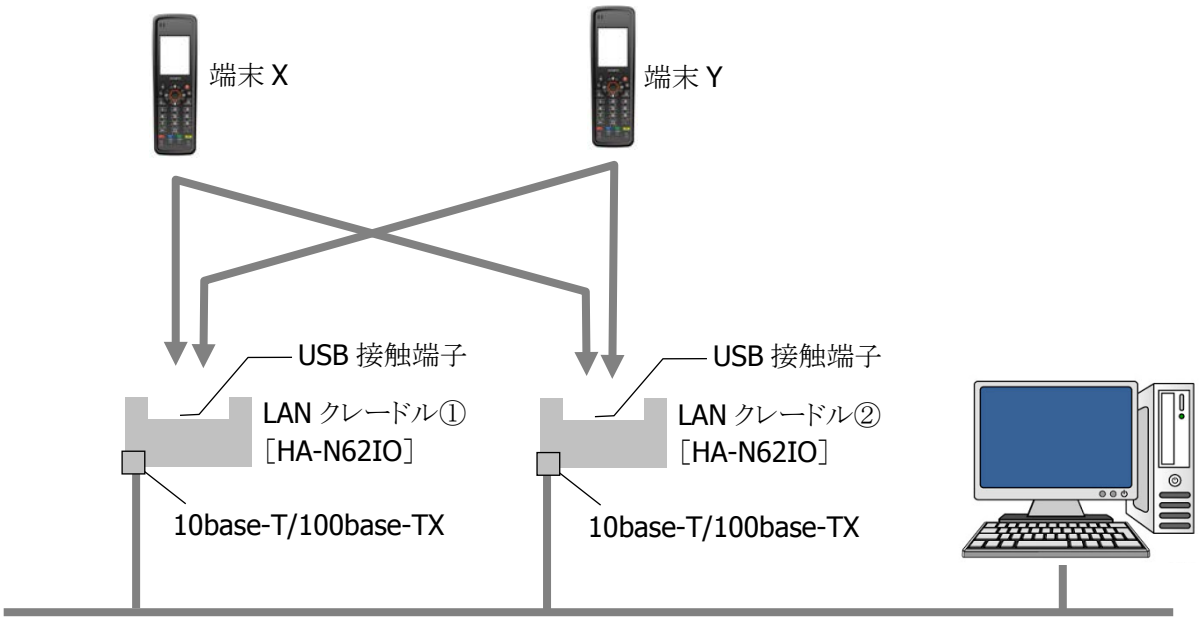
A) DT-970 端末の IP を利用

B) クレードルの IP を利用

の何れかを選択可能です。A)の場合は端末を意識した挙動(*1)となり、B)の場合はクレードルを意識した挙動(*2)となります。

*1…[端末 X][端末 Y]の IP での制御となるので、[端末 X]を[クレードル①][クレードル②]の何れに装着しても、[端末 X]としてのファイル転送処理が可能。

*2…[クレードル①][クレードル②]の IP での制御となるので、[クレードル①] に[端末 X][端末 Y]の何れに装着しても、[クレードル①]としてのファイル転送処理が可能。



[図 2-11]

※マルチドロップでは、許可する端末 ID リストを管理する形態でしたが、上記手法で実現する場合には、IP 毎に同一内容のファイルを複数用意する必要があります。

2) サンプルを用いた改編例

マルチドロップ IrDA ファイル送受信サンプルを例として、FLINK IrDA への改編を説明します。

[リスト 2-1] マルチドロップ IrDA

```
#include <CU_MULTI.H>                                                    (1)

ER              Ret;
DAT_COM_STR     ComParam;
CU_FILE_INFO_FORM  FileInfo[4];
UH              FileCnt;
UW              ComStat, BiosStat;
B               SendRecvMode; // MODE_RECV or MODE_SEND
UB              GraphPos = 0;

cu_stopKeySet( CU_FNC_1 );                                               (2)

memset(&ComParam, 0, sizeof(DAT_COM_STR));
ComParam.speed    = B_19200;                                             (3)
ComParam.length   = CHAR_8;
ComParam.parity   = PARI_NON;
ComParam.stop_bit = STOP_1;

if( cu_open( COM0, CU_CNCT_MULT, &ComParam ) != E_OK ){                  (4)
   cu_readErrStat( COM0, &ComStat, &BiosStat );                          (5)
   Ret = (ComStat & 0xFF);
   cu_stopKeySet( CU_FNC_NON );                                          (2)
   return( Ret );
}
cu_setDrive( CU_DRIVE_B );                                               (6)

memset(&FileInfo, 0, sizeof(CU_FILE_INFO_FORM) * 4);
if( SendRecvMode == MODE_RECV ){
   memcpy( FileInfo[0].fileName, "XXXXXXX DAT", 11 );                    (7)
   memcpy( FileInfo[1].fileName, "YYYYYYY LOD", 11 );
   FileCnt = 2;
   Ret = cu_fileRecv(COM0, 0, CU_KIND_SPECIAL, &FileCnt,                 (8)
                           &FileInfo[0], CU_GRAPH_ON_2, GraphPos );
}else{
   memcpy( FileInfo[0].fileName, "XXXXXXX DAT", 11 )                     (7)
   memcpy( FileInfo[1].fileName, "YYYYYYY DAT", 11 )
   FileCnt = 2;
   Ret = cu_fileSend(COM0, 0, CU_KIND_ALL, &FileCnt,                     (8)
                           &FileInfo[0], CU_GRAPH_ON_2, GraphPos );
}
if( Ret == E_OK ){
   Ret = CU_ERR_NON;
}else{
   cu_readErrStat( COM0, &ComStat, &BiosStat );                          (5)
   Ret = (ComStat & 0xFF);
}
cu_close( COM0 );                                                        (9)
cu_stopKeySet( CU_FNC_NON );                                             (2)
```

[リスト 2-2] FLINK IrDA

```
ER          Ret;
CU_RSPRM    RsParam;
CU_ERRINFO  ErrInfo;
CU_GRAPHSET     GraphSet;
B           SendRecvMode;
B           TargetFiles[1024];
B           *TermDir = "B:¥¥";
B           *HostDir = "C:¥¥HT-DIR¥¥";

cu_stopKeySet(CU_FNC_1);                                                    (A)

if( (Ret = Ir_SetWinMode(SET_WIN2K)) != E_OK ){                             (B)
  return(Ret);
}
memset(&RsParam, 0x00, sizeof(CU_RSPRM));
if(cu_open(COM0,CU_B115K,&RsParam,CU_MODE_HT) != E_OK ){                    (C)
  cu_readErrStat( COM0, &ErrInfo );                                         (D)
  Ret = ErrInfo.category;
  cu_stopKeySet( CU_FNC_NON );                                              (A)
  return(Ret);
}
memset(&GraphSet, 0, sizeof(CU_GRAPHSET));                                  (E)
GraphSet.graphMode = CU_GRAPH_ON_2;
GraphSet.graphPos = 0;
GraphSet.graphCol = 0;
GraphSet.graphName = CU_GRAPH_NM_FILE;
GraphSet.graphLine = 12;

if( SendRecvMode == MODE_RECV ){
  sprintf( TargetFiles, "%s%s::%s%s",                                       (F)
           HostDir, "XXXXXXX.DAT", HostDir, "YYYYYYY.LOD" );
  Ret = cu_fileRecv( COM0, CU_TRANS_NORMAL, TargetFiles,                    (G)
                     TermDir, CU_PROTECT_INVALID, &GraphSet );
}else{
  sprintf( TargetFiles, "%s%s::%s%s",                                       (F)
           TermDir, "XXXXXXX.DAT", TermDir, "YYYYYYY.DAT" );
  Ret = cu_fileSend( COM0, CU_TRANS_NORMAL, TargetFiles,
                     HostDir, CU_PROTECT_INVALID, &GraphSet );              (G)
}
if( Ret == E_OK ){
  Ret = CU_ERR_NON;
}else{
  cu_readErrStat( COM0, &ErrInfo );                                         (D)
  Ret = ErrInfo.category;
}
cu_close( COM0, CU_CLOSE_NORMAL );                                          (H)
cu_stopKeySet( CU_FNC_NON );                                                (A)
```

以降の説明で(1)～(9)はマルチドロップ側ソースを指し、(A)～(H)は FLINK 側ソースを指すものとします。

まず、マルチドロップ利用時のヘッダ定義(1)は、FLINK では不要です。

中断キーの設定[cu_stopKeySet](2)は、FLINK でも(A)のように同一記述です。

マルチドロップでは、通信速度は(3)構造体を[cu_open](4)の引数とする指定方法でしたが、FLINK では直接[cu_open](C)の引数で指定します。
IrDA-USB クレードルを利用する場合、通信速度設定は(B)の記述をすると(C)引数を無視して PC 接続モード動作となります。IrDA-USB クレードルを利用せず、本体間 IrDA 通信の場合は、(B)記述は不要で(C)引数が有効となります。

エラー情報取得[cu_readErrStat](5)は、FLINK では(D)のように下記構造体で、より詳細なエラー情報が取得できます。

［リスト 2-3］

```
typedef struct{
   UB      kind;          /*  エラー種別      */
   UB      command;       /*  コマンド種別    */
   UB      category;      /*  カテゴリ        */
   UB      detail;        /*  エラー詳細      */
   UW      biosStat;      /*  システムエラー  */
}CU_ERRINFO;
```

マルチドロップでは端末側のドライブ指定[cu_setDrive](6)を実施して、送受信ファイルはファイル名のみ指定(7)をする形態でした。
FLINK では、フルパスで指定(F)となり、::をデリミタとした複数ファイ指定が可能です。

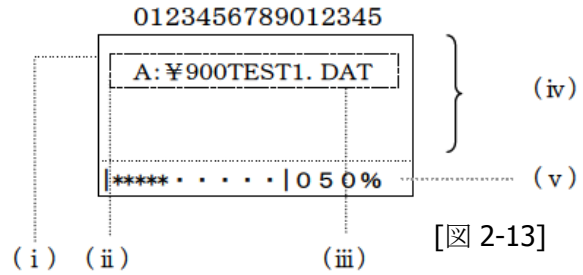
送受信の進捗グラフは、マルチドロップでは[cu_fileRecv/cu_fileSend](8)の第6引数と第7引数で指定していました。
FLINK では、構造体(E)で指定を行い[cu_fileRecv/cu_fileSend](G)に設定行う形態となります。

進捗グラフフォーマットは、マルチドロップでは下記フォーマットでした。

```
CONFIG.HTS          ←転送ファイル名(グラフ表示指定行)
      XXX%          ←進捗のパーセンテージ表示
[*****.....]        ←進捗のグラフ表示(10%単位で.が*に変化します)
```

[図 2-12]

FLINK での進捗グラフフォーマットは下記のようになります。



[図 2-13]

（ｉ）graphPos ファイル名表示先頭行
（ii）graphCol ファイル名表示先頭カラム
（iii）graphName ファイル表示フラグ [全パス表示／ファイル名のみ]
（iv）graphLine ファイル名表示用行数
（ｖ） 進捗グラフ及びパーセンテージ表示（1 行固定）

クローズ処理[cu_close](9)は、FLINK 側(H)では引数が追加となっています。

2.5. FLINK での通信手段変更方法

1) 概要

通信制御関数 cu_xxxx( )の第一引数となっている COM0 は IrDA を指定する値です。
USB/LAN への切り替えは、この第一引数の値を変更することで実現します。

[表 2-3]

No	第一引数	通信手段
1	COM0	IrDA
2	COM9	USB
3	LAN	LAN

通信手段として LAN を利用する場合は、接続相手の指定が必要となります。
指定方法は、cu_open( )の第三引数 CU_RSPRM *rsPrm に対して、下記構造体をキャストして指定する形態となります。

```
typedef struct{
   B  ipaddr[4];
   H  port;
}CU_LANPRM;
```

2) サンプルを用いた改編例

前述 FLINK IrDA サンプル[リスト 2-2]を FLINK USB へ改編する手法を説明します。
cu_xxx( )関数で第一引数が COM0 となっている(a)について、COM9 に変更して下さい。

[リスト 2-4] FLINK USB

```
ER          Ret;
CU_RSPRM    RsParam;
CU_ERRINFO  ErrInfo;
CU_GRAPHSET     GraphSet;
B           SendRecvMode;
B           TargetFiles[1024];
B           *TermDir = "B:¥¥";
B           *HostDir = "C:¥¥HT-DIR¥¥";

cu_stopKeySet(CU_FNC_1);

memset(&RsParam, 0x00, sizeof(CU_RSPRM));
if(cu_open(COM9,CU_B115K,&RsParam,CU_MODE_HT) != E_OK ){                (a)
  cu_readErrStat( COM9, &ErrInfo );                                     (a)
  Ret = ErrInfo.category;
  cu_stopKeySet( CU_FNC_NON );
  return(Ret);
}
memset(&GraphSet, 0, sizeof(CU_GRAPHSET));
GraphSet.graphMode = CU_GRAPH_ON_2;
GraphSet.graphPos = 0;
GraphSet.graphCol = 0;
GraphSet.graphName = CU_GRAPH_NM_FILE;
GraphSet.graphLine = 12;

if( SendRecvMode == MODE_RECV ){
  sprintf( TargetFiles, "%s%s::%s%s",
            HostDir, "XXXXXXX.DAT", HostDir, "YYYYYYY.LOD" );
  Ret = cu_fileRecv( COM9, CU_TRANS_NORMAL, TargetFiles,                (a)
                          TermDir, CU_PROTECT_INVALID, &GraphSet );
}else{
  sprintf( TargetFiles, "%s%s::%s%s",
            TermDir, "XXXXXXX.DAT", TermDir, "YYYYYYY.DAT" );
  Ret = cu_fileSend( COM9, CU_TRANS_NORMAL, TargetFiles,                (a)
                          HostDir, CU_PROTECT_INVALID, &GraphSet );
}
if( Ret == E_OK ){
  Ret = CU_ERR_NON;
}else{
  cu_readErrStat( COM9, &ErrInfo );                                     (a)
  Ret = ErrInfo.category;
}
cu_close( COM0, CU_CLOSE_NORMAL );                                      (a)
cu_stopKeySet( CU_FNC_NON );
```

3. 移行ツール

3.1. 機能

1) 置換関数の検出／置換

表 1-1 の No.6「システムコール イベント操作」flg_sts, clr_flg, wai_flg 記述がソース上に存在するか否かをレポート出力します。
また、これらの関数を DT-930 互換用関数への置換も可能です。
※レポート出力と置換はそれぞれ個別に指定が可能です。

2) 警告関数の検出

表 1-1 の No.2「マルチドロップ／DT500 プロトコル」、及び、No.7「システムコール マニュアル未記載」の記述がソース上に存在するか否かをレポート出力します。

3) 置換ラベルの検出／置換

FLINK での通信手段を IrDA から変更時の対象ラベル COM0 が存在するか否かをレポート出力します。
また、COM0 を他通信手段の定義値[表 2-3]への置換も可能です。
※レポート出力と置換はそれぞれ個別に指定が可能です。

4) 検出／置換を実施した関数／ラベルのサマリー表示

処理対象ファイル全体で、上記 1) 2) 3)処理対象の検出数を、関数毎に画面します。

5) 検出／置換を実施した関数／ラベルのレポート出力

処理対象ファイル毎に、上記 1) 2) 3)処理対象が、ソース上の何行目に存在しているかを、ファイルに対してレポート出力します。

6) 各種定義のカスタマイズ

置換関数、警告関数、置換ラベル、ソースコードページ等の各種定義は、所定のフォーマットに従い、定義内容の追加/編集が可能です。

7) 動作言語環境

動作する Windows OS の言語環境によって、[日本語表記][英語表記]の何れかの表示を行います。（日本語環境の場合は日本語表記、それ以外の言語は英語表記となります）

3.2. 動作環境

本ツールは、Microsoft .NET Framework 3.5 がインストールされている下記 Windows OS を動作対象とします。

- Microsoft Windows XP Professional SP3 (x86)
- Microsoft Windows Server 2003 SP2 (x86)
- Microsoft Windows Vista Business SP2 (x86)
- Microsoft Windows Server 2008 SP2 (x86)
- Microsoft Windows Server 2008 R2 SP1 (x64)
- Microsoft Windows 7 Professional SP1 (x86/x64)

［補足］

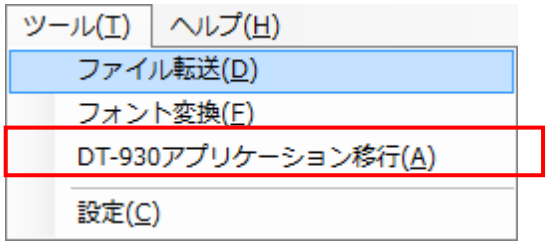
Visual Studio 2008 C# で、Target CPU x86 で生成したツールです。

3.3. 起動方法

本ツールは、以下の2パターンで起動が可能です。

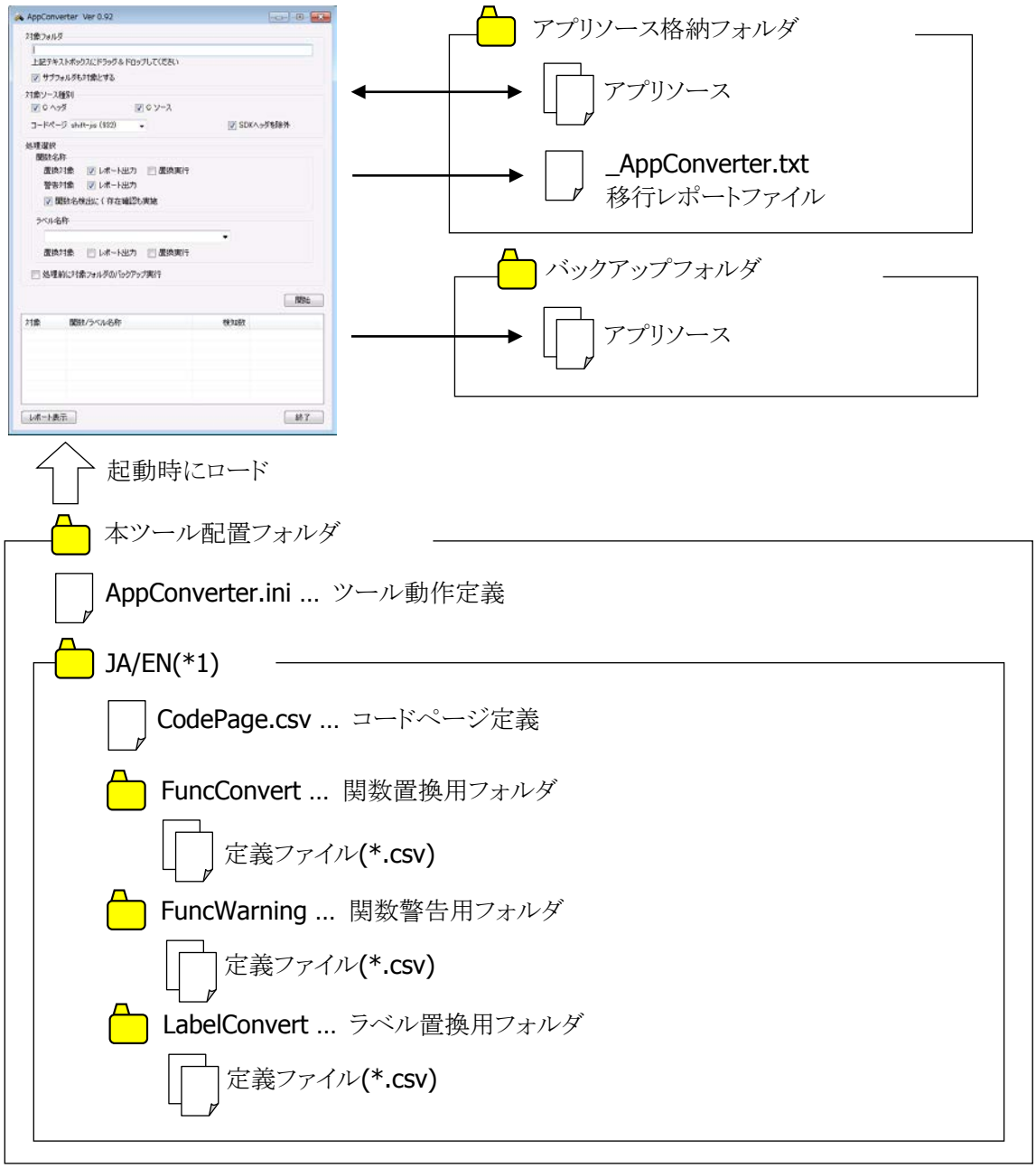
1) エクスプローラ等から AppConverter.exe を直接起動
2) AppConverter.exe のショートカットに対して、フォルダをドラッグ＆ドロップ

※DT-970 GUI 開発環境がインストールされている場合、GUI 開発環境からの起動も可能です。



3.4. ツール構成

[図 3-1]



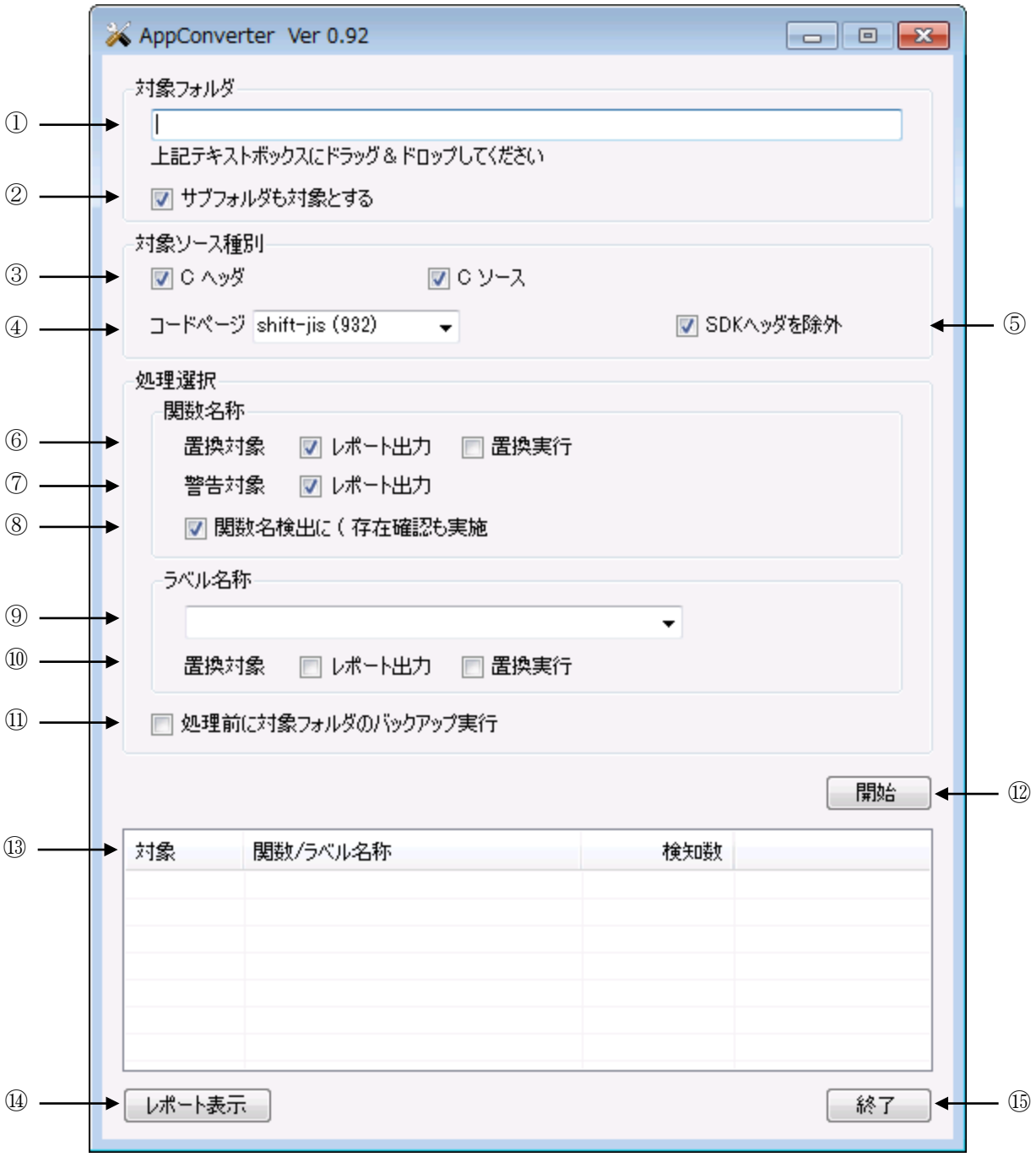
*1…日本語用は[JA]フォルダ、英語用は[EN]フォルダ

本ツール配置フォルダ下の各種定義ファイル(*.ini/*.csv)は、メーカ提供の標準を用意しており、所定のフォーマットに従い、定義内容の追加／編集も可能です。
各定義の詳細内容と初期値は「3.8 定義ファイルフォーマット」に記載します。

3.5. 画面構成

本ツールを起動すると、以下の初期画面を表示します。
各選択のデフォルト状態は、下記、画面の通りとなります。

[図 3-2]



[表 3-1]

項番	内容
①	対象ソースが格納されているフォルダの指定 ショートカットにフォルダをドラッグ＆ドロップ形態で起動した場合には、該当フォルダが初期表示されます。 本テキストボックスに対して、エクスプローラ等から、ドラッグ＆ドロップすると該当フォルダが反映されます。 （フォルダではなく、ファイルをドラッグ＆ドロップした場合は、該当ファイルが格納されているフォルダとなります）
②	対象ソースを指定フォルダ直下のみとするか、サブフォルダ下も対象とするかの指定
③	対象として確認を行うソース種別の指定 C ヘッダ、C ソースを個別に選択できます。 対象とする拡張子は AppConverter.ini で指定可能です。 詳細は「3.8 定義ファイルフォーマット」に記載します。
④	対象ソースを StreamReader/StreamWriter で操作する際の文字コード指定 置換実行を行う場合、ソース文字コード指定が正しく設定されていないと、置換したソース上の記述が字化けしてしまうので、正しく設定して下さい。 選択可能なコードページは、CodePage.csv をメンテすることで追加/削除することが可能です。 詳細は「3.8 定義ファイルフォーマット」に記載します。
⑤	対象ソースから SDK ヘッダを除外するか否かの指定です。 対象とするファイルは AppConverter.ini で指定可能です。 詳細は「3.8 定義ファイルフォーマット」に記載します。
⑥	置換関数に対しての動作指定 レポート出力／置換それぞれを個別に選択できます。 置換を ON すると、自動的に⑪バックアップも ON となります。
⑦	警告関数に対しての動作指定
⑧	関数検知手法の指定 関数検知として、記述までを確認するかの指定です。 詳細は「3.7 詳細動作仕様」に記載します。
⑨	置換ラベルの選択 どのラベルを何に置換するかを選択します。 表示選択する内容は LabelConvert フォルダ下のファイルで定義可能です。 詳細は「3.8 定義ファイルフォーマット」に記載します。

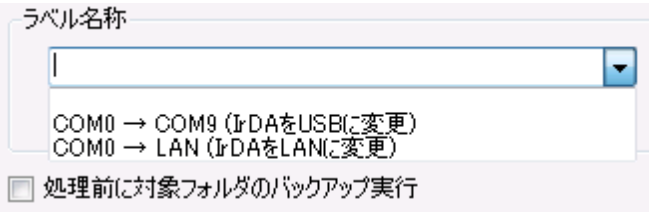
項番	内容
⑩	⑨で選択した内容に対しての動作指定。 レポート出力／置換それぞれを個別に選択できます。 置換を ON すると、自動的に⑪バックアップも ON となります。
⑪	処理前にバックアップを作成するかの指定 ①で選択されているフォルダ名に .bak を付与したフォルダを作成して、該当フォルダ下全てをバックアップします。 ⑤で置換実行が選択されていない場合、ソース更新はされないので、バックアップを行う必要はありません。
⑫	処理開始
⑬	処理サマリー表示 対象ソース全体で、処理対象と検知した関数毎に、その検知数を表示します。
⑭	処理レポート表示 ①で選択されているフォルダ下のレポート(_AppConverter.txt)をテキスト表示に関連付けされたアプリケーション（Windows の設定で既定値は Notepad）で表示します。
⑮	本ツールの終了

3.6. 操作仕様

1) 対象選択

対象フォルダ、対象ソース種別、対象処理を選択して下さい。

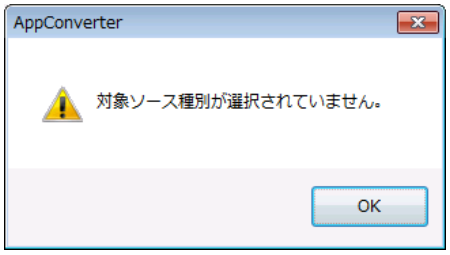
ラベル名称置換選択は、コンボボックスで
[対象]→[置換結果] (説明)
というフォーマットで表示されるので、実施したい置換内容を選択して下さい。



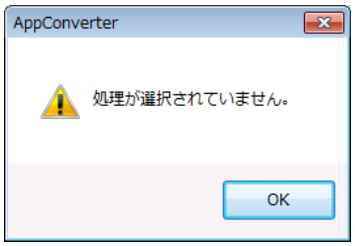
[図 3-3]

2) 処理開始

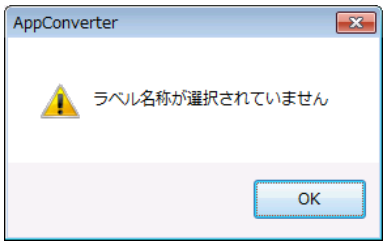
[開始]ボタン押下すると、上記 1)の指定に不足があるかを確認して、不足がある場合には、それぞれ、ダイアログで警告表示をします。



[図 3-4]

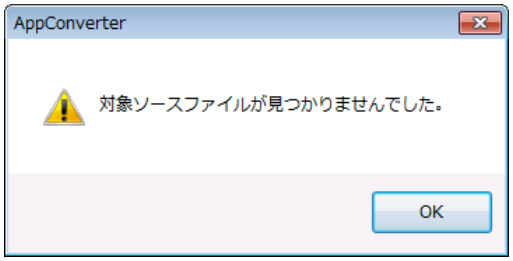


[図 3-5]



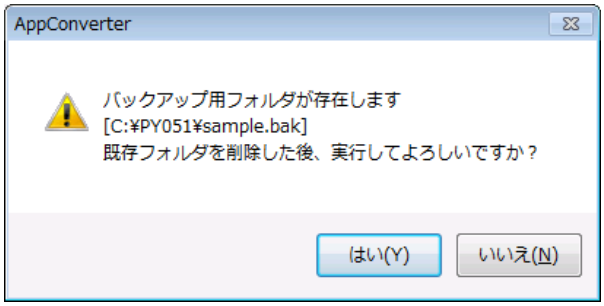
[図 3-6]

指定フォルダが無効、及び、指定フォルダ下に対象ソース種別のファイルが存在しない場合には、以下の警告表示をします。



[図 3-7]

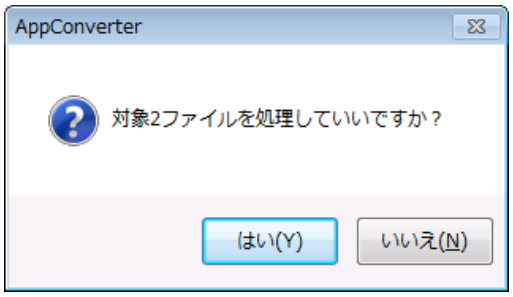
バックアップ指定があり、バックアップ用フォルダが、既に存在していた場合には、以下の確認ダイアログを表示します。



[図 3-8]

[はい]を選択すると、既に存在していたバックアップフォルダを削除して、処理を継続します。
[いいえ]を選択するすると、処理を中断します。

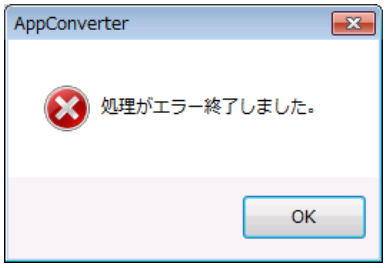
指定内容が正しく、対象ソースが存在した場合には、以下の確認ダイアログを表示し、[はい]で処理を開始します。
[いいえ]を選択するすると、処理を中断します。



[図 3-9]

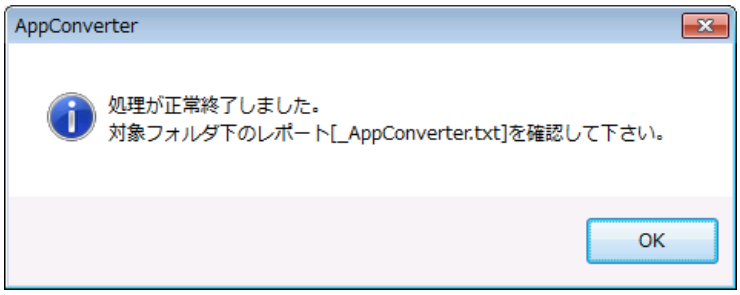
3) 処理結果表示

処理がエラー終了した場合には、以下のダイアログを表示します。

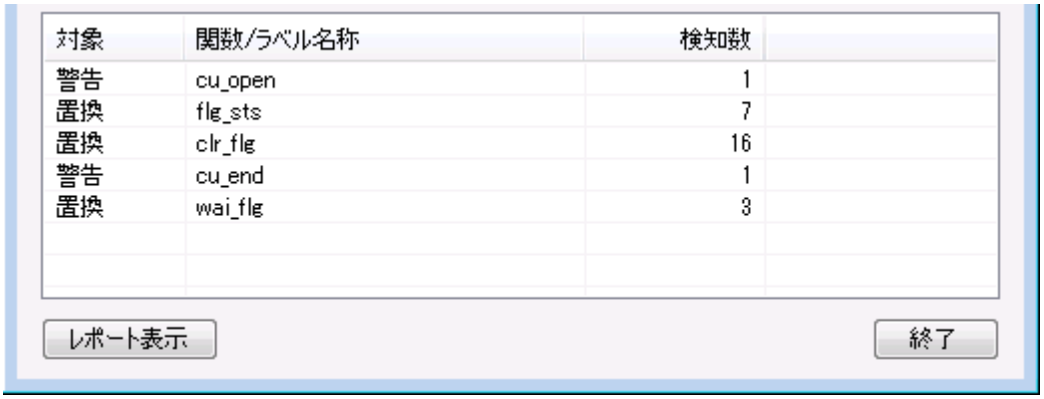


[図 3-10]

処理が正常終了した場合には、以下のダイアログを表示して、画面下部リストビューに処理サマリー表示をします。



[図 3-11]



[図 3-12]

上記、サマリーリストビューでは、カラムヘッダをクリックすることで、カラム毎に、昇順／降順でソートすることができます。

4) 処理結果確認

レポートファイル(_AppConverter.txt)に処理詳細結果が保存されます。
[レポート表示]ボタン押下すると、テキスト表示に関連付けされたアプリケーション（Windows の設定で既定値は Notepad）で表示します。

［リスト 3-1］

```
[ C:¥PY051¥sample¥KISC.H ]
----- 置換:0 警告:0 -----

[ C:¥PY051¥sample¥KISC.C ]
警告: Line 760: cu_open: マルチドロップ/DT500プロトコルはサポート対象外でFLINKのみ利用可能
置換: Line 788: flg_sts: it2_flg_sts に変更が必要
置換: Line 791: clr_flg: it2_clr_flg に変更が必要
置換: Line 807: clr_flg: it2_clr_flg に変更が必要
置換: Line 816: clr_flg: it2_clr_flg に変更が必要
置換: Line 829: clr_flg: it2_clr_flg に変更が必要
置換: Line 838: clr_flg: it2_clr_flg に変更が必要
警告: Line 950: cu_end: マルチドロップはサポート対象外
置換: Line 1351: flg_sts: it2_flg_sts に変更が必要
置換: Line 1354: clr_flg: it2_clr_flg に変更が必要
置換: Line 1360: clr_flg: it2_clr_flg に変更が必要
置換: Line 1549: flg_sts: it2_flg_sts に変更が必要
置換: Line 1552: clr_flg: it2_clr_flg に変更が必要
置換: Line 1566: clr_flg: it2_clr_flg に変更が必要
置換: Line 1841: flg_sts: it2_flg_sts に変更が必要
置換: Line 1844: clr_flg: it2_clr_flg に変更が必要
置換: Line 1850: clr_flg: it2_clr_flg に変更が必要
置換: Line 2920: flg_sts: it2_flg_sts に変更が必要
置換: Line 2923: clr_flg: it2_clr_flg に変更が必要
置換: Line 2978: flg_sts: it2_flg_sts に変更が必要
置換: Line 2981: clr_flg: it2_clr_flg に変更が必要
置換: Line 3566: wai_flg: it2_wai_flg に変更が必要
置換: Line 3567: clr_flg: it2_clr_flg に変更が必要
置換: Line 3577: wai_flg: it2_wai_flg に変更が必要
置換: Line 3578: clr_flg: it2_clr_flg に変更が必要
置換: Line 3588: wai_flg: it2_wai_flg に置換が必要
置換: Line 3589: clr_flg: it2_clr_flg に変更が必要
置換: Line 3649: flg_sts: it2_flg_sts に変更が必要
----- 置換:26 警告:2 -----
```

5) 関数置換動作

置換処理を選択して、置換を実施する場合には、

1.本ツールで置換したことを表すコメント行

2.元ソースをコメント化した行

の 2 行を追加出力します。

［リスト 3-2］

```
// 2013.05.21 by CASIO AppConverter
//     err = flg_sts( &dumy, &ptn, FL_FK_INT_ID);
      err = it2_flg_sts( &dumy, &ptn, FL_FK_INT_ID);
```

3.7. 詳細動作仕様

1) ソース上の関数記述判断

置換対象、警告対象の関数記述判断は、簡易的な字句解析で実施しています。
簡易的な字句解析のため、ソース記述パターンによっては正しく判断されないケースが発生する場合があります。(ある程度イレギュラーな記述とは思いますが、C言語としては、許容範囲内の記述)

上記、関数記述判断については、特定のパターンを許容すると、別のパターンで不整合が発生してしまうため、2種類のルールを用意して、選択可能な形態としました。
[ルール①]と[ルール②]とでは、下記 No6 を実施するか否かの違いとなっています。
本ツールでは、チェックボックス ON 時が[ルール①]の挙動となります。

[表 3-2]

No	関数記述判断の手順	ルール①	ルール②
1	ソースファイルを指定されたコードページでリード	○	○
2	ブロックコメント(*1)を除外	○	○
3	ラインコメント(*2)を除外	○	○
4	リテラル文字列(*3)を除外	○	○
5	ソース1行内で関数名称として有効トークン(*4)抽出	○	○
6	上記トークンの行内で次有効文字が (であるか確認	○	―

1… / ～ */ 記述。複数行にわたるブロックコメントも正しく処理します。
*2… // ～ 記述。
*3…"～" 記述。1行内で完結。"¥"" のエスケープも判断。"" を誤認識しない。
4…[_a-zA-Z][_0-9a-zA-Z] の文字列

[ルール①][ルール②]ともに、下記記述であれば、foo,bar,baz は正しく関数記述と判断されます。

[リスト 3-3]

```
val = foo ( a, b, c );    val=bar(x);

val = foo(bar(x),baz(y));

val =
      foo /* comment */ (
         a,   // comment
         b); // comment
```

下記記述の場合、[ルール①]では foo,bar を関数記述として判断しませんが、[ルール②]では関数記述として判断します。

[リスト 3-4]

```
val = foo
      ( a, b, c );


#define LABEL bar
val =     LABEL ( a, b, c );
```

```
val = bar ( a, b, c );
```

しかし、下記のような記述が存在した場合[ルール②]では、foo,bar を関数記述として誤判断してしまいます。

[リスト 3-5]

```
typedef struct {
    int   foo;
    int   bar;
}
```

2) 置換／警告定義ファイルの対象関数重複

置換／警告の両方を検出する指定を行った場合、置換で該当関数がヒットした場合、警告側の定義は無視します。

置換、警告、それぞれの定義内で重複が存在した場合は、上位に指定されている内容が有効となります。

3.8. 定義ファイルフォーマット

本ツールの各種定義ファイルは、下記 5 種類が存在し、2～5 は日本語用と英語用の2種類のファイルが用意されています。

[表 3-3]

No	定義ファイル種別	定義ファイルのコードページ	
		日本語 OS	左記以外（英語）
1	ツール動作定義	us-ascii(1252) [共通]	
2	コードページ定義	shift-jis(932)	us-ascii(1252)
3	関数置換定義	shift-jis(932)	us-ascii(1252)
4	関数警告定義	shift-jis(932)	us-ascii(1252)
5	ラベル置換定義	shift-jis(932)	us-ascii(1252)

各々のフォーマット詳細と、メーカ提供ファイルの内容を以下に記載します。

1) ツール動作定義

[SourceSuffix]セクションでは、Cヘッダファイル、及び、Cソースファイルとして認識する拡張子を指定します。
個々の拡張子は[*.h]のように、ワイルドカードを含む表記となります。
複数個の拡張子を対象とする場合は[;]で区切り[*.c;*.src]のような表記として下さい。

[FileSearch]セクションでは、検索対象から除外する SDK ヘッダを指定します。
複数ファイルを対象とする場合は[;]で区切り[ITRON.H;BIOS1MAC.H]のような表記として下さい。

メーカ提供ファイルの内容は以下の通りです。

[リスト 3-6]

```
[SourceSuffix]
Header=*.h                                                    ←C ヘッダの拡張子
Source=*.c;*.src                                              ←C ソースの拡張子

[FileSearch]
Ignore=ITRON.H;BIOS1MAC.H;DPLIB.H;IO_APLM.H;CMNDEF.H;B211DEF.H   ←検索除外 SDK ヘッダ
```

AppConverter.ini

2) コードページ定義

コード表記文字列(*1)と、コードページ値(*2)の2項目をカンマ区切りで記述したフォーマットで、メーカ提供ファイルの内容は以下の通りです。

[リスト 3-7]

```
shift-jis,932          ←日本語[シフトJIS]
us-ascii,1252          ←英語
ksc-5601,949           ←韓国語
csbig5,950             ←中国語[繁体字]
csgb2312,936           ←中国語[簡体字]
```

CodePage.csv

*1…Microsoft 文字セット定義ラベル。本データは表示でのみ利用。
*2…Microsoft WIN32 のコードページ。実際のコードページ指定はこの値を利用。

3) 関数置換定義

対象関数と置換結果の 2 項目をカンマ区切りしたフォーマットで、メーカ提供ファイルの内容は以下の通りです。

[リスト 3-8]

```
flg_sts,it2_flg_sts
clr_flg,it2_clr_flg
wai_flg,it2_wai_flg
```

SystemCall.csv

4) 関数警告定義

対象関数と警告出力メッセージの2項目をカンマ区切りしたフォーマットで、メーカ提供ファイルの内容は以下の通りです。

[リスト 3-9]

```
cu_open,マルチドロップ/DT500 プロトコルはサポート対象外で FLINK のみ利用可能
cu_setDrive,マルチドロップ/DT500 プロトコルはサポート対象外
cu_fileSendSet,マルチドロップはサポート対象外
cu_fileSend1,マルチドロップはサポート対象外
cu_end,マルチドロップはサポート対象外
cu_readDirjInfo,マルチドロップはサポート対象外
cu_SetCode,DT500 プロトコルはサポート対象外
ht_MLTsend,マルチドロップはサポート対象外
ht_MLTrecv,マルチドロップはサポート対象外
io_detect,マルチドロップ/DT500 プロトコルはサポート対象外
prn_tecinf_open,TEC IrDA プリンタライブラリはサポート対象外
prn_tecinf_close,TEC IrDA プリンタライブラリはサポート対象外
prn_tecinf_send,TEC IrDA プリンタライブラリはサポート対象外
prn_tecinf_send2,TEC IrDA プリンタライブラリはサポート対象外
prn_tecinf_status,TEC IrDA プリンタライブラリはサポート対象外
c_open,シリアル通信制御はサポート対象外
c_close,シリアル通信制御はサポート対象外
c_status,シリアル通信制御はサポート対象外
c_hold,シリアル通信制御はサポート対象外
c_chkopen,シリアル通信制御はサポート対象外
c_dout,シリアル通信制御はサポート対象外
c_din,シリアル通信制御はサポート対象外
c_tmdin,シリアル通信制御はサポート対象外
c_out,シリアル通信制御はサポート対象外
c_break,シリアル通信制御はサポート対象外
c_txrx,シリアル通信制御はサポート対象外
c_timer,シリアル通信制御はサポート対象外
c_rs,シリアル通信制御はサポート対象外
c_er,シリアル通信制御はサポート対象外
c_errs,シリアル通信制御はサポート対象外
c_iobox,シリアル通信制御はサポート対象外
c_irout,シリアル通信制御はサポート対象外
c_flush,シリアル通信制御はサポート対象外
c_bfsts,シリアル通信制御はサポート対象外
c_errbfring,シリアル通信制御はサポート対象外
c_rderrsts,シリアル通信制御はサポート対象外
c_chghdr,シリアル通信制御はサポート対象外
c_brkevent,シリアル通信制御はサポート対象外
c_mdout,シリアル通信制御はサポート対象外
c_mdin,シリアル通信制御はサポート対象外
c_wp,シリアル通信制御はサポート対象外
c_cimode,シリアル通信制御はサポート対象外
xy_modem,シリアル通信制御はサポート対象外
```

CasioSdk.csv

[リスト 3-10]

```
CRE_TSK,ITRON ネイティブシステムコール
cre_tsk,ITRON ネイティブシステムコール
acre_tsk,ITRON ネイティブシステムコール
act_tsk,ITRON ネイティブシステムコール
  :
 (*3)
```

NativeSystemCall.csv

*3…5 章 5-5 に記載されている関数を全てが列挙されています。

5) ラベル置換定義

対象、置換結果、説明の3項目をカンマ区切りしたフォーマットで、メーカ提供ファイルの内容は以下の通りです。

[リスト 3-11]

```
COM0,COM9,IrDA を USB に変更
COM0,LAN,IrDA を LAN に変更
```

FLINK.csv

4. DT-930 互換関数

4.1. 関数一覧

DT-930 アプリケーション資産を迅速に DT-970 に移行するために、DT-930 と同一 I/F(引数)を持つ関数を、DT-930 互換関数として、カシオ提供 SDK に追加します。

1) イベント操作

[表 4-1]

No	関数名	機能概要
1	it2_flg_sts	イベントフラグ状態参照 （DT-930 flg_sts 互換 I/F）
2	it2_clr_flg	イベントフラグのクリア （DT-930 clr_flg 互換 I/F）
3	it2_wai_flg	イベントフラグ待ち （DT-930 wai_flg 互換 I/F）

5. 関数一覧

5.1. カシオ提供 SDK マルチドロップ／DT500 プロトコル

通信制御は[マルチドロップ][DT500プロトコル][FLINK]の3つが存在し、それぞれの通信制御で公開関数名の重複があります。（例えば cu_open は 3 つの通信制御全てに存在するが引数は異なる）
公開関数名の関数実体への置換は BIOS1MAC.H 上のプリプロセッサ判定で実施されます。
#ifdef CU_MLT_FLAG　　このラベルが有効ならば[マルチドロップ]
#ifdef CU_DT500_FLAG　　このラベルが有効ならば[DT500 プロトコル]
上記 2 つが無効ならば[FLINK]という判定となっています。

各公開関数が[マルチドロップ][DT500 プロトコル][FLINK]それぞれに存在するか否かを下記表にまとめました。（FLINK は移行時の互換性課題はないので、FLINK のみに存在する関数は[No]列を[－]表記で、網掛としています）

下記表で同一関数名でも、[マルチドロップ][DT500 プロトコル][FLINK]それぞれの仕様として、引数／機能に相違があります。

1）基本機能

No	関数名	機能概要	マルチドロップ	DT500	FLINK
1	cu_stopKeySet	通信中断キー指定	○	○	○
2	cu_setDrive	ファイル送受信ドライブ指定	○	○	無
3	cu_open	通信ポート初期化	○	○	○
4	cu_fileSend	ファイル一括送信	○	○	○
5	cu_fileSendSet	送信情報の設定/送信	○	無	無
6	cu_fileSend1	1 ファイルの送信	○	無	無
7	cu_fileRecv	ファイル一括受信	○	○	○
8	cu_msgSend	表示メッセージの送信	○	無	○
9	cu_end	通信を中断	○	無	無
10	cu_close	通信ポートクローズ	○	○	○
11	cu_readErrStat	エラー詳細情報取得	○	○	○
12	cu_readDirjInfo	CU_ERR_DL_RJ 拒否理由値取得	○	無	無
13	cu_SetCode	制御ヘッダコード操作	無	○	無
－	cu_fileAdd	ファイルの追加	無	無	○
－	cu_idle	IDLE 状態へ遷移	無	無	○
－	cu_cmdRecv	コマンド受信待ち	無	無	○
－	cu_fileDelete	ファイル削除	無	無	○

No	関数名	機能概要	マルチドロップ	DT500	FLINK
－	cu_fileMove	ファイル移動	無	無	○
－	cu_makeDir	ディレクトリ作成	無	無	○
－	cu_dateTime	日付時刻の取得／設定	無	無	○
－	cu_getFileInfo	ファイル情報の取得	無	無	○
－	cu_setFileInfo	ファイル情報の設定	無	無	○
－	cu_getDiskInfo	ディスク情報の取得	無	無	○
－	cu_beep	ブザー鳴動	無	無	○
－	cu_getSysInfo	システム情報の取得	無	無	○
－	cu_setIoboxInfo	I/O BOX の情報設定	無	無	○

2）拡張機能

No	関数名	機能概要	マルチドロップ	DT500	FLINK
14	ht_MLTsend	一括送信（接続／切断実施）	○	無	無
15	ht_MLTrecv	一括受信（接続／切断実施）	○	無	無
16	io_detect	IO BOX 検出の有効／無効切替え	○	○	無

5.2. カシオ提供 SDK TEC IrDA プリンタライブラリ

No	関数名	機能概要	互換性	補足
1	prn_tecinf_open	赤外線通信のオープン	▲	
2	prn_tecinf_close	赤外線通信のクローズ	▲	
3	prn_tecinf_send	赤外線でのコマンド送信	▲	
4	prn_tecinf_send2	赤外線でのタイムアウト付きコマンド送信	▲	
5	prn_tecinf_status	赤外線でのステータスリード	▲	

▲:モバイルプリンタ制御ライブラリへの移行が必要

5.3. カシオ提供 SDK シリアル通信制御／送受信切替ライブラリ

1) シリアル通信制御 基本機能

No	関数名	機能概要	互換性	補足
1	c_open	COM のオープン	▲	*1
2	c_close	COM のクローズ	▲	*1
3	c_status	COM ステータスリード	▲	
4	c_hold	COM の占有	▲	
5	c_chkopen	COM のオープンチェック	▲	
6	c_dout	文字列の送信	▲	*1
7	c_din	1文字の受信	▲	
8	c_tmdin	タイムアウト監視の受信	▲	*1
9	c_out	1文字の送信	▲	
10	c_break	ブレイク信号の制御	▲	
11	c_txrx	送受信の有効/無効	▲	
12	c_timer	DR/CS/CD タイムアウト監視値の設定	▲	
13	c_rs	RS 信号の制御	▲	
14	c_er	ER 信号の制御	▲	
15	c_errs	ER/RS 信号の制御	▲	
16	c_iobox	IO ボックス送信の設定	▲	
17	c_irout	IO ボックス送信	▲	
18	c_flush	受信バッファクリア	▲	
19	c_bfsts	受信バッファステータスのリード	▲	
20	c_errbfring	リターンコードバッファリング制御の設定	▲	
21	c_rderrsts	エラーステータスのリード	▲	
22	c_chghdr	受信ハンドラの切り替え	▲	
23	c_brkevent	ブレーク要因の設定	▲	
24	c_mdout	メモリブロックの送信	▲	
25	c_mdin	メモリブロックの受信	▲	
26	c_wp	WakeUp 信号の設定	▲	
27	c_cimode	CI 信号立ち上げモード設定	▲	

▲：IrDA 制御への移行が必要

*1：DT-970 では同一名称関数が「USB HID 通信」として存在

2) シリアル通信制御 拡張機能

No	関数名	機能概要	互換性	補足
28	xy_modem	XYMODEM	▲	

▲：IrDA 制御 XYMODEM（Ir_xy_modem）への移行が必要

3) 送受信切替ライブラリ

No	関数名	機能概要	互換性	補足
－	c_out	シリアル通信制御 c_out の機能拡張版	▲	
－	c_dout	シリアル通信制御 c_dout の機能拡張版	▲	

▲：IrDA 制御への移行が必要

5.4. C標準関数 マニュアル記載

No	関数名	機能概要	互換性	補足
1	isalnum	英字･10 進数字の判定	○	
2	isalpha	英字の判定	○	
3	iscntrl	制御文字の判定	○	
4	isdigit	10 進数字の判定	○	
5	isgraph	空白を除く印字文字の判定	○	
6	islower	英小文字の判定	○	
7	isprint	空白を含む印字文字の判定	○	
8	ispunct	特殊文字の判定	○	
9	isspace	空白類文字の判定	○	
10	isupper	英大文字の判定	○	
11	isxdigit	16 進数字の判定	○	
12	tolower	英大文字を英小文字に変換	○	
13	toupper	英小文字を英大文字に変換	○	
14	acos	浮動小数点数の逆余弦	○	
15	asin	浮動小数点数の逆正弦	○	
16	atan	浮動小数点数の逆正接	○	
17	atan2	浮動小数点どうしを除算した結果の逆正接	○	
18	cos	浮動小数点数のラディアン値の余弦	○	
19	sin	浮動小数点数のラディアン値の正弦	○	
20	tan	浮動小数点数のラディアン値の正接	○	
21	cosh	浮動小数点数の双曲線余弦	○	
22	sinh	浮動小数点数の双曲線正弦	○	
23	tanh	浮動小数点数の双曲線正接	○	
24	exp	浮動小数点数の指数	○	
25	frexp	浮動小数点数を(0.5,1.0)の値として 2 のべき乗の積に分解	○	
26	ldexp	浮動小数点数と 2 のべき乗の乗算	○	
27	log	浮動小数点数との自然対数	○	
28	log10	浮動小数点数の 10 を底とする対数	○	
29	modf	浮動小数点数を整数部分と小数部分に分解	○	
30	pow	浮動小数点数のべき乗	○	
31	sqrt	浮動小数点数の正の平方根	○	
32	ceil	浮動小数点数の小数点以下を切り上げた整数値	○	
33	fabs	浮動小数点数の絶対値	○	
34	floor	浮動小数点数の小数点以下を切り捨てた整数値	○	
35	fmod	浮動小数点数どうしを除算した結果の余り	○	

○:互換/△:代替関数に移行/×:代替無

No	関数名	機能概要	互換性	補足
36	setjmp	現在実行中の関数の実行環境を、指定した記憶域に待避	○	
37	longjmp	setjmp で退避した関数の実行環境を回復し、setjmp 関数を呼び出した位置に制御を移動	○	
38	va_start	可変個の引数を参照するための初期値を設定	○	
39	va_arg	可変個の引数を持つ関数に対し、現在参照中引数の次の引数を参照	○	
40	va_end	可変個の引数を持つ関数の引数への参照を終了	○	
41	fclose	ファイルのクローズ	○	
42	fopen	ファイルのオープン	○	
43	freopen	現在オープンしているファイルをクローズし、新たに指定ファイル名のファイルをオープン	○	
44	sprintf	データを書式に従って変換し、指定領域に出力	○	
45	sscanf	指定領域からデータを入力し、書式に従って変換	○	
46	fread	ファイルから指定領域にデータを入力	○	
47	fwrite	指定領域からファイルにデータを出力	○	
48	fseek	ファイルの現在の読み書き位置を移動	○	
49	ftell	ファイルの現在の読み書き位置を取得	○	
50	rewind	ファイルの現在の読み書き位置をファイル先頭に移動	○	
51	ferror	ファイルがエラー状態であるかを判定	○	
52	clearerr	ファイルのエラー状態をクリア	○	
53	atof	数を表現する文字列を double 型の浮動小数点数に変換	○	
54	atoi	10 進数を表現する文字列を int 型の整数値に変換	○	
55	atol	10 進数を表現する文字列を long 型の整数値に変換	○	
56	strtod	数を表現する文字列を double 型の浮動小数点数に変換	○	
57	strtol	数を表現する文字列を long 型の整数値に変換	○	
58	srand	Rand 関数で生成する疑似乱数列の初期値を設定	○	
59	calloc	記憶域を確保し、確保したすべての領域を 0 クリア	○	
60	free	指定した記憶域を解放	○	
61	malloc	記憶域を確保	○	
62	realloc	記憶域の大きさを指定した大きさに変更	○	
63	abort	プログラムの異常終了 ※本関数の実体はデバイス抽象化ライブラリ	○	
64	exit	プログラムの正常終了 ※本関数の実体はデバイス抽象化ライブラリ	○	

○:互換/△:代替関数に移行/×:代替無

No	関数名	機能概要	互換性	補足
65	bsearch	二分割検索	○	
66	qsort	ソートの実行	○	
67	abs	int 型整数の絶対値	○	
68	div	int 型整数の除算の商と余り	○	
69	labs	long 型整数の絶対値	○	
70	ldiv	long 型整数の除算の商と余り	○	
71	memcpy	複写元の記憶域の内容を、指定サイズ分複写先の記憶域に複写	○	
72	strcpy	複写元の文字列を複写先の記憶域に NULL も含めて複写	○	
73	strncpy	複写元の文字列を指定文字数分、複写先の記憶域に複写	○	
74	strcat	文字列の後に文字列を連結	○	
75	memcmp	指定した 2 つの記憶域の比較	○	
76	strcmp	指定した 2 つの文字列の比較	○	
77	strncmp	指定した 2 つの文字列を、指定文字数分まで比較	○	
78	memchr	指定記憶域で、指定文字が最初に現れる位置を検索	○	
79	strchr	指定文字列で、指定文字が最初に現れる位置を検索	○	
80	strcspn	指定文字列を先頭から調べ、別の指定文字列以外の文字が先頭から何文字続くかを取得	○	
81	strpbrk	指定文字列で、別の指定文字列が最初に現れる位置を検索	○	
82	strrchr	指定文字列で、指定文字が最後に現れる位置を検索	○	
83	strspn	指定文字列を先頭から調べ、別の指定文字列が先頭から何文字続くかを取得	○	
84	strstr	指定文字列で、別の指定文字列が最初に現れる位置を検索	○	
85	memset	指定記憶域の先頭から指定文字を指定した文字数分、設定	○	
86	strerror	エラーメッセージを設定	○	
87	strlen	文字列の長さを取得	○	

○:互換/△:代替関数に移行/×:代替無

5.5. システムコール マニュアル記載 ファイル/メモリ操作

No	関数名	機能概要	互換性	補足
1	open	ファイルのオープン	○	
2	close	ファイルのクローズ	○	
3	read	ファイルのリード	○	
4	write	ファイルのライト	○	
5	lseek	ファイル位置の指定	○	
6	sbrk	メモリ領域の割り当て	○	

○:互換/△:代替関数に移行/×:代替無

5.6. システムコール マニュアル記載 イベント操作

No	関数名	機能概要	互換性	補足
1	flg_sts	イベントフラグ状態参照	△	it2_flg_sts に置換
2	clr_flg	イベントフラグのクリア	△	it2_clr_flg に置換
3	wai_flg	イベントフラグのセットを待機 (タイムアウト無)	△	it2_wai_flg に置換

○:互換/△:代替関数に移行/×:代替無

※上記 3 関数は「1.4 関数互換性」に記載した DT-930 互換関数に置換する対象です。

5.7. システムコール マニュアル未記載

DT-930 で採用していた ITRON2.0 と、DT-970 で採用する μITRON4.0 とでは、システムコール仕様変更（関数名変更、及び、引数変更）により、互換性は維持されていません。
DT-930 アプリケーションの互換性は、マニュアル（デバイス制御ライブラリ リファレンスマニュアル）に記載範囲（5 章 5-3/5-4）としていますが、マニュアル未記載のシステムコールの一覧情報を記載します。
これらのシステムコールを利用している場合は、ITRON2.0 と μITRON4.0 との互換性を確認して頂く必要があります。

本一覧は、ITRON2.0 システムコールをベースとして、ITRON2.0→μITRON4.0 で関数名称が変更となったシステムコールを追記した形態です。
DT-930 マニュアルにはサポートが明記されていないが、ITRON.H として関数プロトタイプ宣言が記述されていたものは[DT-930 ITRON.H]列で○としています。
上記が○となっていて、μITRON4.0 では関数名が変更された関数は[関数名変更]列に変更された関数名を記述しています。

No	関数名	機能概要	DT-930 ITRON.H	関数名変更
1	CRE_TSK	タスク生成（静的 API）	－	
2	cre_tsk	タスク作成（ID 指定）	○	
3	acre_tsk	タスク作成（ID 自動割付）	－	
4	act_tsk	引数指定無のタスク起動	－	
5	iact_tsk	引数指定無のタスク起動[*1]	－	
6	can_act	タスク起動要求のキャンセル	－	
7	sta_tsk	引数指定有のタスク起動	○	
8	ista_tsk	引数指定有のタスク起動[*1]	○	
9	del_tsk	タスクの削除	○	
10	ext_tsk	自タスクを正常終了する	○	
11	exd_tsk	自タスクを正常終了後、削除する	○	
12	ter_tsk	タスクの強制終了	○	
13	chg_pri	タスク優先度の変更	○	
14	ichg_pri	タスク優先度の変更[*1]	○	
15	rot_rdq	タスクの優先順位の回転	○	
16	irot_rdq	タスクの優先順位の回転[*1]	○	
17	rel_wai	タスク待ち状態の強制解除	○	
18	irel_wai	タスク待ち状態の強制解除[*1]	○	
19	get_tid	実行状態のタスク ID の参照	○	
20	iget_tid	実行状態のタスク ID の参照[*1]	－	
21	get_pri	タスク優先度の参照	－	
22	ref_tsk	タスクの状態参照	－	
23	ref_tst	タスクの状態参照（簡易版）	－	
24	tsk_sts	タスク状態の参照	○	→ref_tsk

*1…非タスクコンテキスト用

No	関数名	機能概要	DT-930 ITRON.H	関数名変更
25	sus_tsk	タスクのサスペンド	○	
26	isus_tsk	タスクのサスペンド[*1]	○	
27	rsm_tsk	サスペンドからのタスク再開	○	
28	irsm_tsk	サスペンドからのタスク再開[*1]	○	
29	slp_tsk	タスクのスリープ(タイムアウト無)	○	
30	tslp_tsk	タスクのスリープ(タイムアウト付き)	－	
31	dly_tsk	タスク実行の遅延	－	
32	wai_tsk	タスク実行の遅延	○	→dly_tsk
33	wup_tsk	待ち状態のタスクを起床	○	
34	iwup_tsk	待ち状態のタスクを起床[*1]	○	
35	can_wup	タスクの起床要求の無効化	○	
36	frsm_tsk	強制待ち状態のタスクを強制再開	－	
37	DEF_TEX	タスク例外処理ルーチンの定義(静的 API)	－	
38	def_tex	タスク例外処理ルーチンの定義	－	
39	ras_tex	タスク例外処理の要求	－	
40	iras_tex	タスク例外処理の要求[*1]	－	
41	dis_tex	タスク例外処理の禁止	－	
42	ena_tex	タスク例外処理の許可	－	
43	sns_tex	タスク例外処理禁止状態の参照	－	
44	ref_tex	タスク例外処理の状態参照	－	
45	CRE_FLG	イベントフラグの生成(静的 API)	－	
46	cre_flg	イベントフラグの生成(ID 指定)	－	
47	acre_flg	イベントフラグの生成(ID 自動割付)	－	
48	del_flg	イベントフラグの削除	－	
49	set_flg	イベントフラグのセット	○	
50	iset_flg	イベントフラグのセット[*1]	○	
51	twai_flg	ベントフラグのセットを待機(タイムアウト付き)	－	
52	pol_flg	ベントフラグのセットを待機(ポーリング方式)	○	
53	ref_flg	イベントフラグの状態参照	－	
54	CRE_SEM	セマフォ生成(静的 API)	－	
55	cre_sem	セマフォ生成(ID 指定)	－	
56	acre_sem	セマフォ生成(ID 自動割付)	－	
57	del_sem	セマフォの削除	－	
58	sig_sem	セマフォ資源の返却	○	
59	isig_sem	セマフォ資源の返却[*1]	○	
60	wai_sem	セマフォ資源の獲得(タイムアウト無)	○	
61	twai_sem	セマフォ資源の獲得(タイムアウト付き)	－	
62	pol_sem	セマフォ資源の獲得(ポーリング方式)	－	
63	preq_sem	セマフォ資源の獲得(ポーリング方式)	○	→pol_sem
64	ref_sem	セマフォの状態参照	－	
65	sem_sts	セマフォの状態参照	○	→ref_sem

*1…非タスクコンテキスト用

No	関数名	機能概要	DT-930 ITRON.H	関数名変更
66	CRE_DTQ	データキューの生成(静的 API)	－	
67	cre_dtq	データキューの生成(ID 指定)	－	
68	acre_dtq	データキューの生成(ID 自動割付)	－	
69	del_dtq	データキューの削除	－	
70	snd_dtq	データキューへの送信(タイムアウト無)	－	
71	tsnd_dtq	データキューへの送信(タイムアウト付き)	－	
72	psnd_dtq	データキューへの送信(ポーリング方式)	－	
73	ipsnd_dtq	データキューへの送信(ポーリング方式)[*1]	－	
74	fsnd_dtq	データキューへの強制送信	－	
75	ifsnd_dtq	データキューへの強制送信[*1]	－	
76	rcv_dtq	データキューからの受信(タイムアウト無)	－	
77	trcv_dtq	データキューからの受信(タイムアウト付き)	－	
78	prcv_dtq	データキューからの受信(ポーリング方式)	－	
79	ref_dtq	データキューの状態参照	－	
80	CRE_MBX	メールボックスの生成(静的 API)	－	
81	cre_mbx	メールボックスの生成(ID 指定)	－	
82	acre_mbx	メールボックスの生成(ID 自動割付)	－	
83	del_mbx	メールボックスの削除	－	
84	snd_mbx	メールボックスへの送信	－	
85	isnd_mbx	メールボックスへの送信[*1]	－	
86	snd_msg	メールボックスへの送信	○	→snd_mbx
87	isnd_msg	メールボックスへの送信[*1]	○	→isnd_mbx
88	rcv_mbx	メールボックスからの受信(タイムアウト無)	－	
89	trcv_mbx	メールボックスからの受信(タイムアウト付き)	－	
90	prcv_mbx	メールボックスからの受信(ポーリング方式)	－	
91	rcv_msg	メールボックスからの受信(タイムアウト無)	○	→rcv_mbx
92	prcv_msg	メールボックスからの受信(ポーリング方式)	○	→prcv_mbx
93	ref_mbx	メールボックスの状態参照	－	
94	mbx_sts	メールボックスの状態参照	○	→ref_mbx
95	CRE_MTX	ミューテックスの生成(静的 API)	－	
96	cre_mtx	ミューテックスの生成(ID 指定)	－	
97	acre_mtx	ミューテックスの生成(ID 自動割付)	－	
98	del_mtx	ミューテックスの削除	－	
99	loc_mtx	ミューテックスのロック	－	
100	tloc_mtx	ミューテックスのロック(タイムアウト付き)	－	
101	ploc_mtx	ミューテックスのロック(ポーリング方式)	－	
102	unl_mtx	ミューテックスのロック解除	－	
103	ref_mtx	ミューテックスの状態参照	－	

*1…非タスクコンテキスト用

No	関数名	機能概要	DT-930 ITRON.H	関数名変更
104	CRE_MBF	メッセージバッファの生成(静的 API)	－	
105	cre_mbf	メッセージバッファを生成(ID 指定)	－	
106	acre_mbf	メッセージバッファを生成(ID 自動割付)	－	
107	del_mbf	メッセージバッファの削除	－	
108	snd_mbf	メッセージバッファの送信(タイムアウト無)	－	
109	tsnd_mbf	メッセージバッファの送信(タイムアウト付き)	－	
110	psnd_mbf	メッセージバッファの送信(ポーリング方式)	－	
111	rcv_mbf	メッセージバッファから受信(タイムアウト無)	－	
112	trcv_mbf	メッセージバッファから受信 (タイムアウト付き)	－	
113	prcv_mbf	メッセージバッファから受信(ポーリング方式)	－	
114	ref_mbf	メッセージバッファの状態参照	－	
115	CRE_POR	ランデブポートの生成(静的 API)	－	
116	cre_por	ランデブポートの生成(ID 指定)	－	
117	acre_por	ランデブポートの生成(ID 自動割付)	－	
118	del_por	ランデブポートの削除	－	
119	cal_por	ランデブの呼出し(タイムアウト無)	－	
120	tcal_por	ランデブの呼出し(タイムアウト付き)	－	
121	pcal_por	ランデブの呼出し(ポーリング方式)	－	
122	acp_por	ランデブの受付(タイムアウト無)	－	
123	tacp_por	ランデブの受付(タイムアウト付き)	－	
124	pacp_por	ランデブの受付(ポーリング方式)	－	
125	fwd_por	ランデブの回送	－	
126	rpl_rdv	ランデブの終了	－	
127	ref_por	ランデブポートの状態参照	－	
128	ref_rdv	ランデブの状態参照	－	
129	DEF_INH	割込みハンドラの登録(静的 API)	－	
130	def_inh	割込みハンドラの登録	－	
131	dis_int	割込みの禁止	－	
132	ena_int	割込みの許可	－	
133	ATT_ISR	割込みサービスルーチン生成(静的 API)	－	
134	cre_isr	割込みサービスルーチン生成(ID 指定)	－	
135	acre_isr	割込みサービスルーチン生成 (ID 自動割付)	－	
136	del_isr	割込みサービスルーチンの削除	－	
137	ref_isr	割込みサービスルーチンの状態参照	－	
138	chg_ixx	割込みマスクの変更	－	
139	get_ixx	割込みマスクの参照	－	
140	chg_ims	割込みマスクの変更	○	
141	get_ims	割込みマスクの参照	－	
142	ims_sts	割込みマスクの参照	○	→get_ims
143	ret_int	割込みハンドラからの復帰	－	

No	関数名	機能概要	DT-930 ITRON.H	関数名変更
144	CRE_MPF	固定長メモリプールの生成(静的 API)	－	
145	cre_mpf	固定長メモリプールの生成(ID 指定)	－	
146	acre_mpf	固定長メモリプールの生成(ID 自動付与)	－	
147	del_mpf	固定長メモリプールの削除	－	
148	get_mpf	固定長メモリブロックの獲得(タイムアウト無)	－	
149	tget_mpf	固定長メモリブロックの獲得 (タイムアウト付き)	－	
150	pget_mpf	固定長メモリブロックの獲得 (ポーリング方式)	－	
151	get_blk	固定長メモリブロックの獲得(タイムアウト無)	○	→get_mpf
152	pget_blk	固定長メモリブロックの獲得 (ポーリング方式)	○	→pget_mpf
153	rel_mpf	固定長メモリブロックの返却	－	
154	rel_blk	固定長メモリブロックの返却	○	→rel_mpf
155	ref_mpf	固定長メモリプールの状態参照	－	
156	mpl_sts	固定長メモリプールの状態参照	○	→ref_mpf
157	CRE_MPL	可変長メモリプールの生成(静的 API)	－	
158	cre_mpl	可変長メモリプールの生成(ID 指定)	－	
159	acre_mpl	可変長メモリプールの生成(ID 自動付与)	－	
160	del_mpl	可変長メモリプールの削除	－	
161	get_mpl	可変長メモリブロックの獲得(タイムアウト無)	－	
162	tget_mpl	可変長メモリブロックの獲得 (タイムアウト付き)	－	
163	pget_mpl	可変長メモリブロックの獲得 (ポーリング方式)	－	
164	rel_mpl	可変長メモリブロックの返却	－	
165	ref_mpl	可変長メモリプールの状態参照	－	
166	set_tim	システム時刻の設定	○	
167	get_tim	システム時刻の参照	○	
168	isig_tim	タイムティックの供給	－	
169	sys_clk	カーネルの時間管理処理要求	－	
170	CRE_CYC	周期ハンドラの生成(静的 API)	－	
171	cre_cyc	周期ハンドラの生成(ID 指定)	－	
172	acre_cyc	周期ハンドラの生成(ID 自動付与)	－	
173	del_cyc	周期ハンドラの削除	－	
174	sta_cyc	周期ハンドラの動作開始	－	
175	stp_cyc	周期ハンドラの動作停止	－	
176	ref_cyc	周期ハンドラの状態参照	－	

No	関数名	機能概要	DT-930 ITRON.H	関数名変更
177	CRE_ALM	アラームハンドラの生成(静的 API)	－	
178	cre_alm	アラームハンドラの生成(ID 指定)	－	
179	acre_alm	アラームハンドラの生成(ID 自動付与)	－	
180	del_alm	アラームハンドラの削除	－	
181	sta_alm	アラームハンドラの動作開始	－	
182	stp_alm	アラームハンドラの動作停止	－	
183	ref_alm	アラームハンドラの状態参照	－	
184	DEF_OVR	オーバランハンドラの定義(静的 API)	－	
185	def_ovr	オーバランハンドラの定義	－	
186	sta_ovr	オーバランハンドラの動作開始	－	
187	stp_ovr	オーバランハンドラの動作停止	－	
188	ref_ovr	オーバランハンドラの状態参照	－	
189	loc_cup	CPU ロック状態への移行	－	
190	iloc_cpu	CPU ロック状態への移行[*1]	－	
191	unl_cpu	CPU ロック状態の解除	－	
192	iunl_cpu	CPU ロック状態の解除[*1]	－	
193	sns_loc	CPU ロック状態の参照	－	
194	sns_ctx	コンテキストの参照	－	
195	dis_dsp	ディスパッチの禁止	－	
196	ena_dsp	ディスパッチの許可	－	
197	sns_dsp	ディスパッチ禁止状態の参照	－	
198	sns_dpn	ディスパッチ保留状態の参照	－	
199	ref_sys	システムの状態参照	－	
200	DEF_SVC	拡張サービスコールの定義(静的 API)	－	
201	def_svc	拡張サービスコールの定義	－	
202	cal_svc	サービスコールの呼出し	－	
203	sys_cal	サービスコールの呼出し	－	→cal_svc
204	DEF_EXC	CPU 例外ハンドラの定義(静的 API)	－	
205	def_exc	CPU 例外ハンドラの定義	－	
206	ref_cfg	コンフィギュレーション情報の参照	－	
207	ref_ver	バージョン情報の参照	－	
208	get_ver	バージョン情報の参照	○	→ref_ver
209	ATT_INI	初期化ルーチンの追加(静的 API)	－	

*1…非タスクコンテキスト用

6. 付録

6.1. DT-930 との相違について

1) 変数初期化

DT-970 では、条件分岐等で初期化していない変数を使用すると、変数が不定なため、正しく判定できなくなります。変数は必ず初期化して使用してください。
詳細については、「アプリケーション開発ガイド」を参照してください。

2) ループの処理時間

ループウェイト(for 文などでの繰り返し処理によるウェイト)は、CPU 性能などの動作環境で消費時間は異なります。DT-930 と DT-970 では CPU 性能が異なるため、DT-930 のソースをそのまま DT-970 で利用すると、ウェイトされる時間は異なります。このため、期待する時間ウェイトさせるためにはソース修正が必要となります。

3) 互換表示モード

DT-930 互換表示モード 2 を使用した場合、グラフィック描画には互換性はないため表示位置の修正が必要となります。
互換表示モードについては、「デバイス制御ライブラリリファレンスマニュアル」を参照してください。

4) sprintf 関数のゼロパディング

printf 系書式のフィールド長先頭に 0 を記載して文字列を整形した場合(例:"%05s")、DT-930 では、空白部分は 0 埋めされていましたが、DT-970 では、空白部分はそのままとなります。0 埋めをするには別途アプリケーションの修正が必要となります。
(数値に対する書式 "%05d" の場合は、DT-930, DT-970 ともに 0 埋めされます。)

5) sprintf 関数の符号

printf 系書式のフィールド長先頭に - を記載して文字列を整形した場合(例:"%05s")、DT-930 では、符号を 1 桁にカウントしていましたが、DT-970 では、符号を 1 桁にカウントしません。符号を含む文字列を整形するには別途アプリケーションの修正が必要となります。

6) stddef.h のインクルード

DT-930 では、stdio.h または stdlib.h の中で stddef.h をインクルードしていたため、これらをインクルードしていれば、明示的に stddef.h をインクルードする必要がありませんでした。
しかし、DT-970 では stddef.h の定義内容を利用する場合は、明示的に stddef.h をインクルードする必要があります。

7) 関数のプロトタイプ宣言

DT-970 では、プロトタイプ宣言せずに関数を呼び出すと、ビルド時にエラーとなります。関数を使用する際は必ずプロトタイプ宣言をしてください。
詳細については、「アプリケーション開発ガイド」を参照してください。

※ DT-970 基本開発キット Ver.1.02 以前は、プロトタイプ宣言しなくてもビルドエラーにならず、第 5 引数以降の値が正しく参照されませんでした。

8) pwr_vibrator 関数による振動の継続

DT-930 では pwr_vibrator 関数は停止するまで振動し続けていましたが、DT-970 では振動開始後約 4 秒で自動的に停止されます。
振動を継続するためには、再度 pwr_vibrator 関数で振動を開始してください。
pwr_vibrator 関数については、「デバイス制御ライブラリ リファレンスマニュアル」を参照してください。

9) カシオ IR インターフェイスの廃止

DT-970 では、カシオ Ir インターフェイスはサポートしていません。DT-970 で IrDA を使用するアプリケーションを作成する場合は、IrCOMM プロトコルの IrDA 制御関数（Ir_xxx）を使用してください。 この場合、相手先は IrCOMM に対応している機器に変更する必要があります。
IrDA 制御関数については、「デバイス制御ライブラリ リファレンスマニュアル」を参照してください。

10) トリガーキーの検出処理

DT-970 では、OBR 使用時のトリガーキー、L/R キーの動作が DT-930 と異なります。DT-930 では、トリガーキーを押下したままの状態で OBR_open 関数を実行すると、レーザが発光し読取動作を開始します。DT-970 では、トリガーキーを押下したままの状態で OBR_open 関数を実行しても、レーザを発光しません。DT-970 で DT-930 と同様のトリガーキーの検出処理をするためには、レーザスキャナの設定ファイルにトリガーキーの設定を追加してください。（詳細はソフトウェアマニュアルの「2.8.12 トリガーキー動作設定」を参照してください）
また、DT-930 では、L/R キーが OBR キーに登録されていると、OBR_close 関数実行時にキーロールオーバーが無効になります。DT-970 では、L/R キーは常にキーロールオーバーが有効になります。

カシオ計算機お問い合わせ窓口

製品に関する最新情報

●製品サポートサイト（カシオペア・ハンディターミナル）

https://casio.jp/support/ht/

カシオ計算機株式会社

〒151-8543　東京都渋谷区本町 1-6-2

TEL 03-5334-4638(代)