USB 仮想シリアルポートの実装と RTOS のサービスコールの拡張

苫小牧工業高等専門学校　○菅原 吉平，吉村 斎，阿部 司

要　旨

本研究では，苫小牧工業高等専門学校が開発した TECL ボード上で動作する TOPPERS/ASP カーネルに，USB 仮想シリアルポートを実現した．USB 仮想シリアルポートは，USB Communication Device Class の仕様に準拠するものとした．さらに，これを用いて，TOPPERS/ASP カーネルのサービスコールによる文字列の出力を可能とした．

1. はじめに

今日，情報機器と端末の通信において，USB が広く使用されており，レガシーの RS232C の使用は減少している．

我々は，RX62N マイコン(ルネサス社製)搭載の組込みシステム教材「TECL(Technology Education Computer Laboratory)ボード」を開発した[1]．現在，TECL ボードのシリアル通信ポートは，RX62N マイコンの SCI(Serial Communication Interface)を使用した RS232C であり，実習時には USB と RS232C の変換ケーブルを用いている．TECL ボードに搭載している USB Mini-B 端子は，実験・実習では活用していない．

実験・実習では，TECL ボード上において動作する TOPPERS/ASP カーネル(以下，ASP カーネル)を用いている．先行研究[1]において，RX62N 用に移植した ASP カーネルを使用している．USB Mini-B 端子を使用したシリアル通信を実現するためには，ASP カーネルで USB デバイスドライバが必要となる．

本研究では，USB Communication Device Class(CDC)の仕様に準拠した USB 仮想シリアルポートを ASP カーネルに実現し，アプリケーションプログラムやログデータの出力を，USB を通して行えるようにすることが目的である．

2. 研究概要

本研究の開発・実行環境を**図 1** に示す．TECL ボード上に USB デバイスとして実装する．

本研究では，ルネサス社製の Renesas Starter Kit+ for RX62N 用のサンプルプログラム [1](以下，RSP)を調査し，使用する．RSP は，前述の CDC に準拠している．また，RSP には USB Stack というディレクトリが存在し，この階層を通して，USB 通信でハードウェアを制御する．

先行研究[2]において，TECL ボードでの RSP の動作確認を行い，コールグラフや UML のシーケンス図を用いて，RSP のソースコードの解析を行った．解析の結果としては，RSP の構造は**図 2**[3]のようになっており，USB Stack は**図 2** の網掛け部分であることを確認した．

USB 仮想シリアルポートを実現した手順を以下に示す．

(1)ASP カーネルに USB 仮想シリアルポートを実現することが可能かを調べるため，ユーザアプリケーション上に RSP を実装し，動作を確認する．

(2)ASP カーネルに USB デバイスドライバを実装するために，**図 3** に示した ASP カーネルへの RSP の実装案に従って，RSP を ASP カーネルに移植する．

(3)アプリケーションプログラムから，USB 通信を使用できるようにするため，ASP カーネルにある文字列の出力を行うサービスコールを調査し，USB 仮想シリアルポートを使用して文字列を出力するように変更する．

1.RX62N CS+用ルネサススタータキットのサンプルコード
https://www.renesas.com/ja-jp/software/D3013173.html

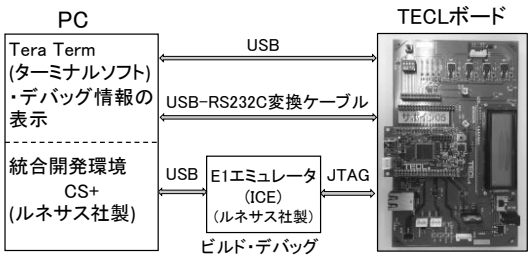

図 1　開発・実行環境

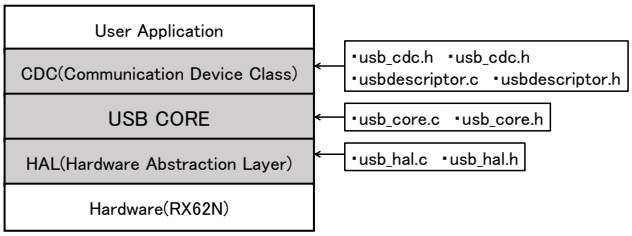

図 2　RSP の階層構造とそれを構成するファイル

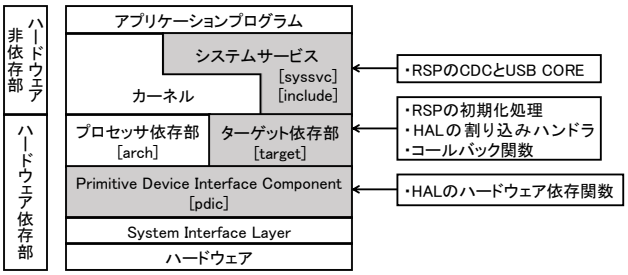

図 3　ASP カーネルの USB 仮想シリアルポートの実装案

3. ASP カーネルにおけるシリアル出力サービスコール

ASP カーネルでは，**図 3** のシステムサービスにおいて，文字列の出力を行う関数(サービスコール)を用意している．このサービスコールは，タスクの状態などのデバッグ情報を得る目的で使用する．文字列の出力を行うサービスコールは以下の 2 つがある．

①serial_wri_dat

②syslog (syslog_n, n=0,…,6)

①は，syssvc/serial.c に存在し，シリアルポートに文字列を送信する関数である．②は，システムログ機能を提供する関数で，include/vasyslog.c に存在する．以下に，syalog を用いたシリアルポートへの出力方法を示す．

(1)文字列をバッファに記録し，システムログタスクにおいてバッファの内容を出力する．その際，システムサービスの serial_wri_dat 関数を経由する．これは，通

常のデバッグ情報の出力に用いられる．

(2)文字列をそのまま低レベル出力する．これは，カーネルの起動バナーの出力に用いられている．この場合は，カーネル起動時に文字列が出力されるため，USB が PC 上で認識された直後に文字列を出力することは，①に比べて困難となる．

本研究では，(1)の方法を用いた文字列の出力を，RX62N マイコンの SCI を使用した RS232C から USB を通して行うように変更する．

4. ASP カーネル上での USB の動作

4.1 アプリケーションプログラムによる動作確認

ASP カーネルのユーザアプリケーションとして，RSP を実装した．また，RSP の関数を使用した文字列の入出力を行うアプリケーションを作成し，正常に動作することを確認した．

4.2 RSP の ASP カーネルへの移植

次に，4.1 節のプログラムを使用し，**図 2** に示した RSP の網掛けの層を，ハードウェア非依存部と依存部とに分割し，**図 3** に示すように ASP カーネルの各階層に配置する．

RSP を調査した結果，CDC と USB CORE はハードウェアに依存せず，HAL は，ハードウェア非依存とハードウェア依存の関数が存在することを確認した．

以上の事実を踏まえて，CDC と USB CORE 内の関数は，ASP カーネルのハードウェア非依存部に配置し，HAL 内の関数は， target ディレクトリと pdic ディレクトリへ配置する．

以下に，ASP カーネルで USB を動作させるために，RSP の移植方法を示す．

(1) システムサービス（syssvc）
- RSP の usb_cdc.c，usb_cdc.h，usbdescriptors.c，usbdescriptors.h を実装した．

(2) プロセッサ依存部（arch/rx620_ccrx/rx620.h）
- RX62N マイコンの USB 周辺レジスタの定義を追記した．

(2) ターゲット依存部（target）
- RSP の usb_hal.c，usb_hal.h 内の USB の初期化処理と割込みハンドラと割込みに関するコールバック関数を実装し，USB の割込みを可能とした．

(3) シリアルドライバ依存部
(Primitive Device Interface Component, pdic)
- RSP の usb_hal.c，usb_hal.h 内のハードウェアに依存している関数を実装した．
- rx620_uart.c の rx620_uart_snd_chr 関数内の SCI の TDR レジスタに値を格納する処理を以下のように変更した．RSP の usb_hal.c 内で定義されていたバッファを使用し，文字列を格納する．RSP の HAL 内の関数を呼び出し，文字列を RX62N マイコンの USB 周辺レジスタの FIFO バッファに格納して出力する．

(4) System Interface Layer (SIL)
- レジスタの値を読書きする場合は，SIL の関数を経由して行うように実装した．

以上の実装方法により実装し，4.1 節のプログラムと同様に動作するプログラムを実行した．その結果，RSP と同様に，アプリケーションプログラムで，USB 仮想シリアルポートを動作させることが可能となった．

```
Sample program starts (exinf = 0).
task1 is running (001).   |
task1 is running (002).   |
task1 is running (003).   |
task1 is running (004).   |
task1 is running (005).   |
task1 is running (006).   |
task1 is running (007).   |
task1 is running (008).   |
```

図 4　sample1 の実行結果

4.3 サービスコールの拡張

4.2 節で実装したプログラムを使用し，CS+で，ASP カーネルの基本的な動作を確認するためのサンプルプログラムである sample1 をビルドし，実行した．Tera Term を用いて確認した実行結果を**図 4** に示す．**図 4** より，USB 仮想シリアルポートを通して文字列の出力が行われることが確認できた．

以上の結果から，ASP カーネルのサービルコールによる文字列の出力を，TECL ボードで動作する USB 仮想シリアルポートを通して行うことが可能となった．

5. おわりに

以下に，本研究の成果を示す．

(1)TECL ボードで動作する ASP カーネルに RSP を移植し，USB 仮想シリアルポートを動作させた．

(2)ASP カーネルのサービスコールによる文字列の出力を，TECL ボードで動作する USB 仮想シリアルポートで行うように変更した．

以下に，今後の課題を示す．

(1)サービスコールによる文字列の入力を，TECL ボードで動作する USB 仮想シリアルポートを用いて行うように実装する．

(2)ASP カーネルの仕様に準拠するために，移植したソースコードを整理する．

(3)**図 1** における開発・実行環境の変更の検討を行う．デバッグ環境を E1 エミュレータから，シリアル通信を用いた RX シリアルデバッガ[4]に変更し，デバッグ情報の表示を，USB 通信により行う．

謝辞

本研究を進めるにあたり，名古屋大学大学院 山本椋太氏，北海道立総合研究機構 大村功氏，堀武司氏にご協力いただきました．心より感謝申し上げます．

参考文献

[1]組込みシステム教材の開発と導入
大西 孝臣, 山本 椋太, 木下 大輔, 三上 剛, 阿部 司, 吉村 斎, 高専教育　第 38 号(2015)　pp.90-95

[2]USB 仮想シリアルポートの基礎研究
菅原 吉平, 吉村 斎, 阿部 司, 山本 椋太, 第 17 回複雑系マイクロシンポジウム講演論文集(2018) pp.25-26

[3]RX62N Group RSK+ USB Function Sample Code User's Manual Rev.1.00 (2010)
http://renesasrulz.com/rx/m/mediagallery/153/download

[4]RX シリアルデバッガ | ルネサス エレクトロニクス
https://www.renesas.com/jp/ja/products/software-tools/tools/monitor-debuggers-ram-monitor/rx-serial-debugger.html