

非同期シリアル・トゥ・イーサネット デバイスサーバ

他の電子機器と通信する手段としてシリアルポートを使用するデバイスの数は驚くほど増えました。事実、多くのデバイスにとってシリアルポートは外界との通信の手段を提供する唯一の機構であると言えます。これにはサーモスタット、POSシステム、リモートモニタ、バーコードリーダ、レシートプリンタ、RFIDトランシーバ、血圧測定器、及び現場で使用されるレガシテストツールから最新の建築オートメーションに至るまで含まれます。これらのデバイスはさらに大きなコンピュータネットワークに参加する直接の手段をもたず、しかしながら、新しいアプリケーションはTCP/IPコネクティビティ及びイーサネットの機能が要求されます。高価で時間のかかる再設計はオプションでない場合が多くあります。

...多くのデバイスにとってシリアルポートは外界との通信の手段を提供する唯一の機構であると言えます。

本論文は、DS80C390又はDS80C400マイクロコントローラを使ってTINIプラットフォーム上に構築されたレガシシステムを改装することにより、スタンドアロンのシリアルデバイスをイーサネットに移す、容易で経済的な方法を考えてみました。デバイスが一旦イーサネットに接続されると、HTTPサーバのようなTINIウェブサービスは容易に実行できます。

RS-232シリアルポート

本文で論じられる非同期シリアル通信は、コンピュータの歴史を初期まで溯ったRS-232-Cの規格に基づいています。RS-232-Cは1969年¹に発表されました。近代的なシリアルポートの殆どが、規格に定義されている全ての信号をサポートするわけではありません。さらに、実行される信号は、ただ単に規格に定義されているものに「かなり近い」信号として使われています。今回は純粋な歴史的な定義を無視し、今日RS-232が使われている方法に話を集中して進めます。

スペース及びマーク

RS-232-Cは+3V~-+25Vの電圧レベルを「スペース」(バイナリ0)として、-3V~-25Vを「マーク」(バイナリ1)と特定します。-3V~+3Vの領域は「スイッチング領域」です。多くのユニバーサル非同期レシーバトランスミッタ(UART)は、0及び1に(比較的)近代的なTTL電圧レベル0V及び+5Vを使います。特別な目的のレベルトランスレータ、例えば、かの有名なMAX-232がTTLとRS-232レベル間の変換を行います。DS80C390/DS80C400のシリアルポートはTTLレベルなので、他のTTLレベルUARTにインタフェースする際のレベルトランスレータを必要としません。

DCE及びDTE

データ通信機器(DCE)及びデータ端末機器(DTE)は、通信チャネルの2つのエンドポイントです。主な違いはシリアルコネクタのピン配置です。ヌルモデムと呼ばれているものを両者間の変換に使うことができます。

表1はヌルモデムを使用した時のDB-9 DTEシリアルコネクタ上の信号及び適応する他のDTE上の信号を表しています。

フロー制御

シリアル通信は1つのピン(TD)によって送信され、もう1つのピン(RD)によって聞かれることで実現されます。しかし、2つのデバイスが自由にRD/TD送信で通信すると、片方がもう一方をオーバーランしてデータの損失が発生する可能性があります。通常実行されるフロー制御には2つの方法があります。

- XON/XOFF(ソフトウェアフロー制御という曖昧な言葉で呼ばれることが多い)
- RTS/CTS(ハードウェアフロー制御という曖昧な言葉で呼ばれることが多い)

¹ NASAがこの時代のコンピュータテープの解読に苦労しているので、この比較は有効です。

DTE PIN	SIGNAL NAME	NULL-MODEM
1	CD (Carrier Detect)	4 (DTR)
2	RD (Receive Data)	3 (TD)
3	TD (Transmit Data)	2 (RD)
4	DTR (Data Terminal Ready)	6 (DSR) and 1 (CD)
5	Common (Signal Ground)	5 (Common)
6	DSR (Data Set Ready)	4 (DTR)
7	RTS (Request To Send)	8 (CTS)
8	CTS (Clear To Send)	7 (RTS)
9	RI (Ring Indicator)	N/C

表1. 2つのDB-9 DTE
シリアルポートの信号を
接続するのにヌルモデムを
使うことが可能です。

XON/XOFFフロー制御方式は、他方を休止させ(XOFF、13h)送信を再開(XON、11h)させるインバンドキ
ャラクタを送信します。XON及びXOFFキャラクタは、
バイナリデータストリーム内で生ずる場合、送信側
によってソフトウェアにエスケープされ、受信側にアン
ラップされなければなりません。

RTS/CTSは追加の信号ラインを使用します。RTS
(request to send-送信リクエスト)は送信側によっ
てアサートされます。受信側はデータを受信する準備が
できるとCTS(clear to Send-送信クリア)で応答し、
レシーブバッファが一杯になるとCTSをクリアします。

デバイスによってはフロー制御をサポートするもの
もあり、しないものもあります。従ってデフォルト設定
は通常「フロー制御なし」となっているため、デバイス
がフロー制御実行される場合はオーバライドされなけ
ればなりません。

速度、データビット、停止ビット及びパリティ

通信を成功させるために正確に設定されなければならないその他のパラメータには、送信速度
(ビットレート)、データの数と停止ビット、及び(存在する場合は)パリティチェックの
タイプがあります。殆どの新しいデバイスは、8データビット、パリティなし、そして1停止
ビットという「8N1」の設定を使います。しかし、レガシシステムは可能性の全範囲を使うこと
で知られており、正確な設定が大切になることもあります。

TINI及びネットワーキング

TINIはダラスセミコンダクタ社が開発したテクノロジープラットフォームで、DS80C390及び
DS80C400マイクロコントローラの迅速な開発を可能にしました。特にTINIはチップセットの
定義を含み、最適化されたJavaランタイム環境と組み込みオペレーティングシステムを内蔵して
います。Javaを使うことにより、プログラマは組み込みの開発にはあまり使われない強力な機能
を利用することができます。マルチスレディング、ガーベジコレクション、インヘリタンス、
バーチャル化、クロスプラットフォーム機能、強力なネットワークサポート、そして最後に、
最も大切な無償で提供される開発ツールの数々です。TINIユーザは普通アセンブリ言語コー
ディングから遮蔽されていますが、スピードクリティカル経路又は低レベルハードウェアの
アクセスの最適化のために、ネイティブ言語サブルーチンはサポートされ、奨励されます(TINI
オペレーティングシステムはネイティブコードで書かれており、その結果として、最近のPCの
シリアルI/Oスループットとあまり変わりません)。

TINIはチップセットの定義
を含み、最適化されたJava
ランタイム環境と集積された
組み込みオペレーティング
システムを含みます。

Java.netパッケージのフルサポートに加えて、TINI JavaランタイムはJavax.commサブシステム
の実行も含みます。TCP/IP及びシリアルポートは両方ともJavaから簡単にアクセスできるので、
TINIシステムは容易にシリアル・トゥ・イーサネットのブリッジを実行します。

下記の例に使われているE10ソケット上のTINI m390検定モジュールは、DS80C390 TINI開発
プラットフォーム(TINI m400はDS80C400を使用)のハードウェアの部分です。SRAM、フラッ
シュメモリ、イーサネット、CANバス、1-Wire、等に加えて、システムには4つのシリアル
ポートがあります。UARTの2つはDS80C390の内部(serial0及びserial1と呼ばれます)で、他の
2つは外部です(16550搭載オプションを使う)。ここで大切なのは、E10ソケット上の両方の
シリアルコネクタはserial0に配線され、DTE/DCEピン割当にのみ異なります。

TINI環境に関しては「The TINI Specification and Developer's Guide」(Addison-Wesley, 2001)に
詳細が記載されています(英文のみ)。www.maxim-ic.com/TINIGuideから無償でPDFコピーが
ダウンロードできます。

例：

まず2つの具体的なアプリケーションから始め、その後どのようなアプリケーションにも
適するように修正できる汎用シリアル・トゥ・イーサネットプログラムから抜粋した簡単な例を
あげます。これらの例はTINI m390/400検定モジュールを使って構築されます。

TINI検定モジュールはイーサネットに複数のシリアルデバイスを接続する「ブラックボックス」として使われます。エンド機器の要件によって、TINIはデータをストレートに流すか、データストリームを分解、解釈、修正することが可能です(図1)。

例をTINI m390/400上のスラッシュデベロッパのシェルから走らせることもできますが、より高度なアプリケーションはフラッシュに常駐し、電力損失の場合、セルフスタートをして、他のTINI構築技術を使い完成品をほとんど破壊不可能にします。

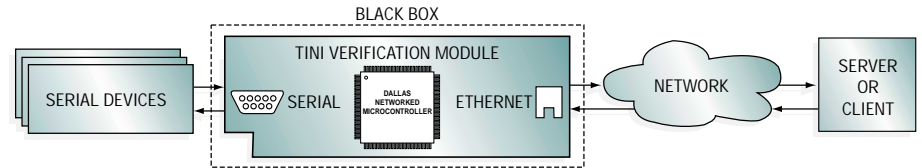


図1. TINIデバイスサーバはシリアルデバイスとイーサネット間のブリッジとして使われます。

例を修正するにはある程度の基本的なネットワークの知識とプログラミングの経験が必要です。ワーキングサンプルコードもダラスのftpサイト(ftp://dalsemi.com)からダウンロードできます。

バーチャルモデム

最初の例「バーチャルモデム」³はTINI m390/400を使い、物理的なモデム及び電話線をTCP/IP接続に取り替えます。工場で、機械の状況、ロード及び効率データを報告するのに1日に何回も中央サーバにモデムを使ってダイヤルインをしなければならない「マシン状態モニタ」のようなレガシデバイスを想定して下さい。サーバ側の拡張しつづけるモデムバンクの必要性を排除し、機器への電話線の変わりに既存のLANを使えるようにするには、

- サーバのソフトウェアをTCP/IPベースのものに書き換える、そして
- 各マシンの従来のモデムをTINIバーチャルモデムに置き換える。

しかし、エンド機器に関してバーチャルモデムは普通のモデムと全く同じように動作するので、マシン状態モニタを修正する必要はありません。

バーチャルモデムは、勿論、上記の設定の代わりにペアで仕様することも可能です。2つのバーチャルモデムを使う時に、サーバソフトウェアを変える必要は全くなく、TINIモジュールは既存のモデムのドロップイン置換え品として取り扱うことができます。

見えない裏の部分で、バーチャルモデムが「ATD」モデムダイヤルコマンドを受け取るとTCP接続を確立します。「ATH」切断コマンドがTCP接続を閉じます。ソフトウェアはいくつもの従来型ATモデムコマンドも実行し、例えば、Microsoft®のWindows®ネットワークによっても真のモデムと認識されます。さらに、バーチャルモデムはTCPポートで聞き取り、エンド機器に「リング」で送られてきた「電話」にも応答します。

下記のコードのフラグメントはTINI m390のシリアルポートを初期化するためのものです。

```
public static void main(String args[])
{
    TINIOS.setSerialBootMessagesState(false);
    TINIOS.setDebugMessagesState(false);
    TINIOS.setConsoleOutputEnabled(false);
    System.out.println("Connecting to serial0 at 9600bps, "
        + "listening on TCP port 8001");
    try {
        CommPortIdentifier portId =
            CommPortIdentifier.getPortIdentifier("serial0");
        SerialPort port = (SerialPort) portId.open("VModemTINI",
            10000);

        TINIOS.setRTSCTSFlowControlEnable(1, false);
        TINIOS.setRTSCTSFlowControlEnable(0, true);
        TCPSerialVirtualModem modem = new
            TCPSerialVirtualModem(port,
                /* Comm speed */ 9600, /*TCP Port */ 8001);
        modem.processInput();
    }
}
```

Java.netパッケージのフルサポートに加えて、TINI JavaランタイムはJavax.commサブシステムの実行も含まれます。TCP/IP及びシリアルポートは両方ともJavaから簡単にアクセスできるので、TINIシステムは容易にシリアル・トゥーイーサネットの橋がけを実行します。

³ www.maxim-ic.comのアプリケーションノート196「Designing a Virtual Modem Using TINI」をご参照下さい。

```

        catch (Exception e) {
            System.out.println("Exception: "+e.toString());
        }
    }
}

```

コードはまず、TINIにおける標準動作である全てのTINI OSデバッグ出力をディセーブルします。ポート識別子を得ると、ポートが開きます(ポートが他のアプリケーションに使用中の場合、待ち時間は2つ目のパラメータでわかります)。次にハードウェアのフロー制御のステートが設定されます。TINI390は、シリアルポート0及び1用にRTS/CTSラインが一式しかないので、プログラムは必要なポートをイネーブルする前に、他のポートのフロー制御をディセーブルします。次にJavaバーチャルモデムが裏付けられます。

バーチャルモデムクラスはATコマンドインタプリタ(ここには出ていませんが、例の中では一番大きな部分を占めます)及びネットワーキングコードで構成されています。下記のコードはシリアルポートのビットレート、データ、停止ビット、パリティを設定し、受信の接続を処理するのがいかに容易かを表しています。

```

/** Creates a new VirtualModem connected to a serial port on
 * one end and a TCP port on the data side.
 * serial -- the serial port this VirtualModem talks to.
 * speed -- the speed the serial port should be set to.
 * tcpport -- the TCP port this VirtualModem listens on.
 * throws IOException when there's a problem with the serial
or TCP port.
 */
public TCPSerialVirtualModem(SerialPort serial, int speed, int
tcpport)
    throws IOException
{
    super(serial);

    try {
        serial.setSerialPortParams(speed, SerialPort.DATABITS_8,
SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);
    }
    catch (UnsupportedCommOperationException e) {
        throw new IOException();
    }
}
...

serverSock = new ServerSocket(tcpport, 1); // backlog of one
listenThread = new listenInbound();
listenThread.start();
}

```

最後に、次のlistenThread()の抜粋が入ってくる接続リクエストを受け入れます。

```

public void run()
{
    int rc;
    Socket s;
    while (running) {
        s = null; // No incoming connection request
        try {
            answered = false;
            s = serverSock.accept();

```

```
// Discard incoming connection if already connected
if (connected)
    throw new IOException();

sock = s; // for answer()
...

```

UPSモニタ

2番目はTINIm390/400を無停電電源装置(UPS)のシリアルポートに接続する例です。ソフトウェアがネットワークUPSツールプロトコル⁴を実行し、多様なプラットフォームの多様なクライアントがUPSの状況をモニタすることを可能にします。このプロジェクトはシリアルポートなしで新しいマッキントッシュコンピュータから既存のUPSをモニタする必要性から生まれたものです。

UPSデバイスには2つの基本的な種類があります。「スマート」と呼ばれているものと、シンプル(又は「dumb-データ処理能力なし」と呼ばれている)ものです。シンプルUPSは状況を数個のシリアルピンを使って信号を送ります。実際にASCIIデータを送るようなことはしません。シリアルピンの数に限りがあるので、例えば報告できる情報が制限されます。

SIGNAL	MEANING
RTS (<i>from UPS</i>)	Low Battery
TD (<i>from UPS</i>)	On Battery
CTS (<i>to UPS</i>)	Kill UPS Power

TINI検定モジュールはイーサネットに複数のシリアルデバイスを接続する「ブラックボックス」として使われます。エンド機器の要件によって、TINIはデータをストレートに流すか、データストリームを分解、解釈、修正することが可能です。

Javaでは状態変更に反応するコードを簡単に実行するのに`Javax.comm.notifyOn...()`方式を使うことが可能です。下記に例をあげます。

```
...
// Listen for DTR changes
try {
    port.addEventListener(this);
} catch (TooManyListenersException e) {
    ...
}
port.notifyOnDSR(true);
...

public void serialEvent(SerialPortEvent ev)
{
    try {
        if (ev.getEventType() == SerialPortEvent.DSR)
            ...
    } catch ...
    ...
}

```

スマートUPSは、シリアルプロトコルを実行し、バッテリー充電率又は温度の値を返してることが可能なので、より興味深いものです。プロトコルは異なった業者間で大きな違いがあり、文書化されていないことが度々あります。ダラスftpサイトのUPSモニタ例はAPCスマートUPSをサポートしますが、他のブランド用に修正することも簡単にできます。

⁴ www.exploits.org/nut/をご覧ください。

下記のコードは、UDPリクエストの受信方法とUDPを介してUPSの状況情報を送信する方法を示しています。

```
// Listen to incoming UDP requests
private class listenUDPThread extends Thread
{
    private DatagramSocket sock;
    private byte[] buffer;
    private DatagramPacket dp;

    public listenUDPThread(DatagramSocket s)
    {
        sock = s;
        buffer = new byte[BUF_SIZE];
        dp = new DatagramPacket(buffer, buffer.length);
    }

    public void run()
    {
        while (running) {
            try {
                sock.receive(dp);
                byte[] data = parseCommand(buffer, dp.getLength());
                sock.send(new DatagramPacket(data, data.length,
                    dp.getAddress(), dp.getPort()));
            }
            catch (Exception e) {
            }
        }
        try {
            sock.close();
        }
        catch (Exception e) {
        }
    }
}
```

...シリアル及びTCP
ポートは入力/出力
ストリームdataIn及び
dataOutと要約され...
CAN及び1-Wire間の
データをブリッジします...

強力なネットワーキングのサポートがJavaに搭載されているので、この例はあらためて説明するまでもありません。while()ループ内のコードはUDPリクエストを受け取るまで待機し、分解し、入ってくるパケットのgetAddress()を使ってリクエストの出所に答えを送信します。

汎用シリアル・トゥ・イーサネットアプリケーション

完全なシリアル・トゥ・イーサネットの例は本論分の範囲外になります(完全な例は『*The TINI Specification and Developer's Guide*』に提示され詳しく説明されています)。しかし、下記のコードフラグメントはシリアルとシリアル・トゥ・イーサネットブリッジのネットワーキング部分でデータを送信するのにいかに効率的にマルチスレディングを使うかを示しています。シリアル及びTCPポートは入力/出力ストリームdataIn及びdataOutと要約され、このコードの層は実際にネットワークに関して何も知る必要がなく、又、例をあげるとCAN及び1-Wire間のデータをブリッジすることが可能です。

```
public GenericBridge()
{
    ...
    running = true;
    dcThread = new dataCopy();
    dcThread.start();
}
```

```

// Thread that copies everything from dataIn to dataOut
private class dataCopy extends Thread
{
    public void run()
    {
        int r = 0;
        while (running && r >= 0) {
            try {
                synchronized (threadLock) {
                    r = dataIn.read(dataBuffer);
                    if (r > 0)
                        dataOut.write(dataBuffer, 0, r);
                }
            }
            catch (Exception e) {
                r = -1;
                ... // Handle error
            }
        }
    }
}

```

結論

多くのレガシデバイス是非同期シリアル通信のみをサポートしますが、最近のアプリケーションはイーサネット接続及びTCP/IPネットワーキングを要求します。DS80C390及びDS80C400マイクロコントローラの強力なJavaランタイム及びTINIテクノロジーを使用することによって、シリアル・トゥ・イーサネットのコンバータを開発するのが容易になり、短時間で完成することが可能です。

Microsoft及びWindowsはMicrosoft Corp.の商標です。