pa-40

DEVELOPMENT OF PLC DRIVER FOR ACCELERATOR CONTROL

Masashi NANAO, Masakatsu MUTOH and Yoshinobu SHIBASAKI

Laboratory of Nuclear Science, Tohoku University
1-2-1 Mikamine, Taihaku-ku, Sendai 982, Japan

ABSTRACT

PLCs are used as an interface between an accelerator and a computer system in the stretcher-booster ring control. A PLC network driver software applicable to MS-DOS is supplied by the PLC maker, however since for WindowsNT it is not supplied, we have developed. We used a WinRT supplied by Blue Water Systems of the US instead of Device Driver Kit(DDK) in order to develop the driver software.

加速器制御用 PLC 制御ドライバの開発

1.はじめに

最近の加速器制御にはプログラマブルコントローラ(以下 PLC とする)が多用されつつある。PLCの利点は値段が安い、プログラミングが容易、各種制御モジュールが豊富なことである。しかし、PLC はスタンドアローンの動作を前提にしているため、外部から制御情報を与えて PLC を遠隔制御する方法は、伝送速度と伝送距離を考慮すると、応用範囲がかなり制限されてしまう。

そこで伝送速度と伝送距離が問題にならないインターフェースを探し、WindowsNT を使ったパソコンから PLC が使えるようにするために、ディバイスドライバを開発した。

2.PLC とパソコンの接続

我々はストレッチャー・ブースタリング(STB)の制御のため、PLC にオムロン社製の "SYSMAC CV1000" シリーズを採用した。パソコンで PLC を制御しようとすれば、両者のインターフェースには RS-232C、GP-IB、SYSMAC LINK、の 3 方式がある。 RS-232C は伝送速度が遅く、早いデータ転送を要求する応用には向かないし、GPIB は伝送速度に問題はないが、距離を延ばすことができない。 SYSMAC LINK はトークン式のメーカ独自のネットワークで、パソコン本体に専用のネットワークカードを入れ、2Mbps の伝送速度、総延長 1km の伝送距離は、速度、距離共問題がないため、我々は SYSMAC LINK をパソコンと PLC とのインターフェースに採用した。

3.ドライバの開発環境

パソコンから PLC を制御するには、両者のコミュニケーションを司るドライバソフトウェアが必要である。メーカから提供されている SYSMAC LINK 用ディバイスドライバは、MS-DOS 対応のものだけで、WindowsNT のパソコンから PLC を使用するには、専用のディバイスドライバを開発しなければならない。

一般的に WindowsNT 用のドライバを開発するには、Device Driver Kit(以下 DDK とする)を使用する方法と、市販の簡易ディバイスドライバ開発ツールを使用する方法がある。 DDK を使用して開発するには、アセンブラレベルのコーディングが必要なため、ハードウェアと WindowsNT の 250

個を越える Application Infrastructure(以下 API とする)の知識が必要になる。

本来、WindowsNT のもとでハードウェアを直接制御するには、ソフトウェアの安全上、ドライバは OS の一部として登録し、ユーザからは直接アクセスできないように、カーネルモードでドライバプログラムを動作させなくてはならない。カーネルモード上で動作するプログラムを書くには、DDK を使用しなければならないが、簡易ツールを使用すると、ハードウェアに対するアクセスの部分は、カーネルモードに組み込まれたそのツールが担当してくれるので、API を知らなくても DDK を使った場合とほぼ同等のドライバを開発することが可能である。今回は開発時間の制約から、後者の方法を採用し、簡易ツールとして Blue Water 社の WinRT を採用した。

WinRT での開発には C 言語を使用し、プログラミング効率を上げることができた。またこのツールを使うとデバッグや修正なども短時間でできるので、全体の開発時間を大幅に短縮することができた。本来なら、ドライバは、システムの持つサービスプログラムと同じようにしたかったが、WinRT からは無理なので、今回開発したドライバは、PLC 制御プログラムのサブプログラムとして組み込んで使うことにした。

4.WinRT の構成要素

WinRT は、WindowsNT の内部に WinRT の機能を組み込む WinRT ドライバ、WinRT 用に書かれてマクロを展開する WinRTpp、WindowsNT にデバイス情報を設定する WinRTreg の3つのプログラムで構成される。

- WinRT ドライバ

  WinRT ドライバは、WinRT を使うプログラムに対して、WindowsNT のデバイス I/O ポートに対する入出力要求、デバイスに割り付けられた物理メモリへのアクセス、割込み制御機能を提供する。

  プリプロセッサ・コマンドやマクロ API を使用してハードウェアに対する I/O 処理を記述することができる。

- WinRTpp…プリプロセッサ

  WinRTpp は、ハードウェア制御プログラムの記述を容易にするためのプリプロセッサである。プリプロセッサ・コマンドを使用して記述されたソースファイルをプリコンパイルし、通常の C コンパイラが解釈可能な形式のソース・ファイルに変換する。第1図に示すように、2組の"%%"で挟まれた行がプリプロセッサによって変換されるコマンドである。

```
%%                          //プリプロセッサコマンド開始
outmb(0x0236,0x81); //236 番地に 0x81 を書き込む
%%                          //プリプロセッサコマンドの終了
```

プリプロセッサ処理後

```
{
    WINRT_CONTROL_ITEM_WinRTpp04[]=
        {//          command     param1      param2
                    {OUTM_B,    0x0236,     0x81},
        };
        (void)WinRTProcessIoBuffer(hWinRT,_WinRTpp04,

        sizeof(_WinRTpp04),&iWinRTlength);
}
```

第1図 WinRTpp 処理。

プリプロセッサ・コマンドを WinRTpp で処理するとマクロ API(例では WinRTProcessIoBuffer())を使用したプログラムに展開される。このマクロ API はさらに Win32·API に展開される。

- WinRTreg…レジストリエディッタ

  WinRTreg は、デバイスに関する情報

(I/O ポート・アドレス、物理メモリ・アドレス、割り込みに関する情報等)をシステムのレジストリ・データベースに登録したり、すでに登録されている情報を変更したりするためのレジストリエディッタである。

6.制作したドライバの内容と性能

メーカ製のドライバと互換性を保つように、ファンクションコールの形式及び引数は同じにし、MS-DOS のもとで作られたプログラムの移植を容易にした。第1表に今回開発したドライバの主な仕様を示す。

また性能は 32bit ネイティブコードで書いてあるので PLC の動作に追従できる高速動作が可能である。

開発したドライバとメーカ製の MS-DOS 用デバイスドライバに同じ処理を実行させ、一定時間内で何回送信と受信(パソコンと PLC 間のメッセージ交換回数)ができるかを測定した。第2表の結果が示すように、メーカ製のドライバが若干速かったが、MS-DOS は基本的にシングルタスクであるため、OS が軽く、又 1 つのプログラムが CPU 資源を独占することを許しているためで、今回それに近い数値が得られたことは、パソコンの能力を充分引き出すドライバが開発できたことを証明している。

7.今後の予定

今回開発したプログラムは、送信プログラムと受信プログラムがスレッド化されてないので、送信だけの応用では、受信部分が無駄になってしまう。システムの CPU 資源を効率よく利用するには、送信プログラムと受信プログラムはそれぞれ独立した動作が望ましく、プログラムのスレッド化が今後の課題である。

送信方向	1:1…SEND,RECV,CMND 命令 1:n…SEND,CMND 命令
データ長	SEND 命令 : 最大 256CH(512ﾊﾞｲﾄ) RECV 命令 : 最大 256CH(512ﾊﾞｲﾄ) CMND 命令 : 最大 542ﾊﾞｲﾄ
データの内容	各命令を実行した場合、以下のデータが送受信されます。 SEND 命令(FUN192) : データ送信要求コマンド／レスポンスデータ RECV 命令(FUN193) : データ受信要求コマンド／レスポンスデータ CMND 命令 : 任意のコマンド／レスポンスデータ
通信ポート No.	ポート 0～7 を設定(8 種類の送受信が同時に実行可能)
レスポンス監視時間	0000 : デフォルト(2 秒)設定となります。 0001～FFFF : ユーザー設定(単位 0.1 秒、6553.5 秒)
再送回数	0～15 回を設定

第 1 表 WindowsNT 用ドライバの性能。

	WindowsNT	MS-DOS(メーカ製)
送信処理 (コマンド時)	3ms	2ms
受信処理 (コマンド時)	3ms	2ms
送信処理 (レスポンス時)	7ms	5.152ms
受信処理 (レスポンス時)	7ms	5.152ms

第 2 表 処理速度の測定比較。