H8マイコン用デバッグモニタの作成

苫小牧高専　○柴田　雅宣，山本　哲也，阿部　司，吉村　斎

要　　旨

H8 マイコン用のデッバッグモニタの作成について発表する．リアルタイムモニタをベースとした組込みシステムの開発を行うためにはデバッグモニタは，必要不可欠である．実験・実習において C 言語を主たる開発言語として利用する場合が多い．また，アセンブリ言語を使用することもある．本研究では，H8 マイコンの利用できる機能を統合的に利用できるデバッグモニタの設計，開発環境について発表する．

1．はじめに

苫小牧高専情報工学科では第 3 学年を対象に Z80 ワンボードマイコンを使用してきた．第 5 学年では，H8 マイコンを用いて PID 制御実験を行っている．Z80 の実験においては，Z80 ワンボードマイコンの部品調達が困難となり，故障時の復旧に支障が生じてきている．また，H8 マイコンボードを用いた PID 制御実験では，RAM の増設を行わないと FLASH メモリの寿命の問題からプログラム開発が困難である．以上に述べた問題点を踏まえて，最近の高性能なワンチップマイコンを利用して，組込み分野の実験・実習システムを構築することを考えた．プロセッサ，コンパイラおよびコンピュータネットワークは，計算機システムを構成する重要な要素であるが，Z80 ワンボードマイコンを用いた実習環境では，システムが陳腐化してこれらの要素を総合的に扱うことができない状況にある．また，従前の実験内容が企業での現状にそぐわない内容であり，学生にはやや退屈な内容になってきている．さらに C 言語をプログラミング実習で学習するが，具体的に機器の制御などに応用する実習は行えていない．

実験・実習システムでは，情報家電などを意識した組込みシステム分野が大きく期待されてきている．本研究では，組込みシステムを統合的に学習するために，RS232C，USB，イーサネットコントローラ，RAM，液晶表示装置（LCD），発光ダイオード（LED）およびスイッチ（SW）などを付加することを前提とした．また，I/O のテストおよびデバッグなどを容易にする必要がある．また，プログラミング環境としては，コンパイラ，リンカ，コード変換，FLASH メモリ書込み，RAM 書込みなどを統合的に行う必要がある．本研究では，以上の機能を利用できることから cygwin 環境を採用した．また，cygwin 環境下で開発したプログラムは，ターミナルソフトから RAM に転送して実行することとした．そのためには，FLASH メモリにコマンドインタプリタ機能を持つデバッグモニタを書き込んで使用することとした．

本研究では，cyawin 環境下でデバッグ作業を行うためのデバッグモニタの開発について報告する．

2．デバッグモニタのコマンド

ユーザプログラムを C 言語で開発し，RAM に転送して実行するデバッグモニタのコマンドとして，以下のモニタコマンドを設定した．

（1）　ユーザプログラムのロード(ld)
（2）　プログラムの実行(go)
（3）　word 単位のメモリダンプ(dw)
（4）　byte 単位のメモリダンプ(db)
（5）　メモリチェック(mc)
（6）　word 単位のデータ設定(ew)
（7）　byte 単位のデータ設定(eb)
（8）　ヘルプ(?)

実現したコマンドインタープリタ付きの初期画面とヘルプ画面を図 1 に示す．

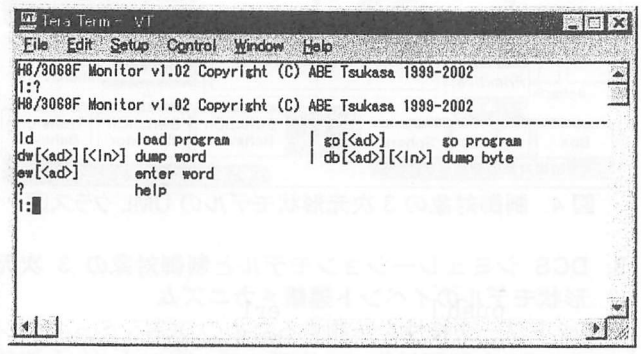

図 1　デバッグモニタの初期画面とヘルプ

3．デバッグモニタの基本構造

デバッグモニタのメインルーチンの主な処理を以下に示す．

（1）　モニタベクタをカレントベクタに設定する．
（2）　ベクタエリアの初期化
（3）　ターミナルの初期化
（4）　割り込みの許可
（5）　モニタの初期メッセージの送出
（6）　ターミナルからのモニタコマンドの受付

デバッグモニタは，パーソナルコンピュータのターミナルソフトからコマンドを受付ける．これには，シリアルコミニュケーション（RS232C）を用いる．ユーザプログラムは，RAM に転送して実行する．ユーザプログラムは，基本的に C 言語のみでプログラム開発を可能とする．組込みシステムでは，資源の制約が厳しい．特にメモリの制約が厳しいため，プログラム資源であるデバッグモニタのサブルーチン群と割込ベクタを，デバッグモニタとユーザプログラムの双方で利用可能とする必要がある．

本研究では，モニタプログラムの関数群のユーザプログラムでの利用は行っていない．しかし，ユーザプログラムで割込みを利用したプログラムを可能とす

るため仮想ベクタを用いてこの問題を解決した. 図２に, 割込みベクタと仮想割込みベクタのアドレスの関係を示すメモリマップを示す.

Head Add.	Contents
000000	Interrupt　Vector
000100	Monitor　Vector
000200	Monitor Program
	・ ・
FFBFFC	Current　Vector
FFC000	Pseudo　Interrupt　Vector
FFC100	・

図２　メモリーマップ

現在のベクタ（Current　Vector）に通常の割込みベクタ（Iterrupt Vector）が割り当てられているが, go 命令を実行し, ユーザプログラムを起動する前に現在の割込みベクタ（Current Vector）を仮想割込みベクタ（pseudo Interrupt Vector）に切換えて, ユーザプログラムで割込みを利用することができるようにしている. 割込みベクタ処理の一部を以下の図３に示す.

```
_IntEntry0x04:
        subs        #2, sp
        stc.w       ccr, @-er7
        push.l      er0
        push.l      er1
        mov.l       @CurrentVector, er0
        mov.b       #0x04, r0l
        mov.l       @er0, er1
        beq         IntError:16
        mov.w       r1, @(10, sp)
        mov.w       e1, r1
        mov.b       r1l, @(9, sp)
        pop.l       er1
        pop.l       er0
        rte
```

図３　割込みベクタの処理

４．ユーザプログラムとデバッグモニタの I/F

ユーザプログラムで割込みを使用する場合には, シリアルポートの初期化と C 言語の main()関数を呼出し, 割込みベクタテーブル(User Vector)を定義する. これにはアセンブリ言語で記述した cstartup.S を使用する. ユーザは, 使用する割込み関数をこの cstartup.S の割込みテーブルに記述し, ユーザプログラム側で以下のように記述することで割込みが使用可となる.

#pragma　interrupt(割込み関数名)

これにより, 学生実験を行う場合, C 言語での開発に集中でき, アセンブリ言語での記述を少なくでき, 煩雑なプログラミング作業での言語知識の切換えを軽減できる。

５．デバッグモニタと JSP カーネルの実行例

デバッグモニタの機能の実行例として、バイト単位のダンプ命令（db）を実行した結果を図４に示す.

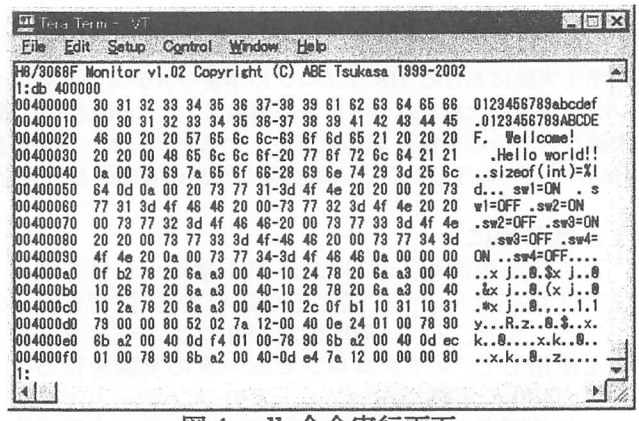

図４　db 命令実行画面

デバッグモニタを用いたユーザプログラムの実行例として、リアルタイム OS として豊橋技術科学大学を中心に開発された μ ITRON4.0 仕様に準拠した JSP カーネルのテストプログラムの実行例を図５に示す.

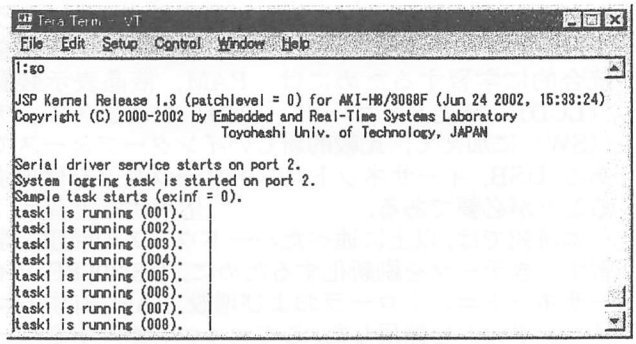

図５　JSP カーネルの実行画面

このサンプルプログラムは, リアルタイム OS のタスクの実行状況をターミナルソフトに報告し, タスクの優先度や状態遷移をターミナルソフトから制御できる.

５．おわりに

組込みシステムの実験・実習に必要な H8/3068F を対象とした, コマンドインタプリタ機能付きデバッグモニタを開発した. 開発には, 阿部の開発した同機能のデバッグモニタをベースとし, コードの改良と機能の追加を行った.

開発したデバッグモニタにより, ユーザプログラムを C 言語のみで記述可能であり, 実験・実習に利用できることを, JSP カーネルを実行することで確認した.

今後の課題としては, 本研究でデバッグモニタの組込みを対象とした CPU は H8/3048F と H8/3068F であるが, 他の CPU への対応がある. また, リアルタイム OS を含む関数群の FLASH メモリへの登録等, 資源の効率的な配置を検討する必要がある.