論　文

PIC マイコンのエミュレータ＆ロジックアナライザ開発

森　久直*　　乾　剛**　　松原　正彦***

Development of an Emulator and Logic Analyzer for PIC

Hisanao Mori*,　Takeshi Inui**,　Masahiko Matsubara***

There is need for an emulator for PIC(Peripheral Interface Controller) which can understand the execution process of programs in the C language. So, we developed an emulator which has standard functions and logic analyzer function. The emulator was designed with VHDL (Very high speed IC Hardware Description Language), and it was implemented in an FPGA (Field Programmable Gate Array). Supported PIC types are PIC16F84A, PIC16F876, and PIC16F877.

キーワード：エミュレータ，ロジックアナライザ，FPGA，VHDL，C

Keywords: Emulator, Logic analyzer, FPGA, VHDL, C

1.　背景と目的

PIC は，メインとなるプロセッサの機能を分散して周辺機器の制御を行うために開発されたコントローラである。そして，小型かつ省電力であり，メモリやタイマ，アナログ入力などの周辺モジュールを持ち，コンパクトな制御回路を開発できる。性能的にもマイコンに近い。そのため，現在ではバーコードリーダや IC タグリーダ，ブラインド制御器など，様々な機器に組込んで利用されている。8bit マイコン市場では 2002 年の数量ベースで世界第 1 位の出荷数を達成しており[1]，PIC は世界中で広く採用されている。

一方で，このような応用製品を効率よく開発するためには，エミュレータが必要である。しかし，市販されているエミュレータは高額なため，中小企業での導入は少ない。また，現存するエミュレータでは，書き込まれたプログラムの実行経過について，蓄積したアドレスと命令コードを解析し，その解析結果を C 言語ソースコードで一括して表示するロジックアナライザの機能などが不足している。

そこで今回は，従来のエミュレータが持つ機能の他にロジックアナライザの機能を付加すると共に，1 チップの FPGA を採用することで，安価で，複数の種類の PIC に対応できるエミュレータを開発する。

＊　IT グループ
＊＊　東京都水道局 東村山浄水管理事務所（前 IT グループ）
＊＊＊　三鷹電工所

2.　開発内容

平成 16 年度の共同開発研究で開発した PIC16F84A 対応版のエミュレータ[2][3]の高機能化を行い，PIC の上位デバイスである PIC16F876 と PIC16F877[4]に対応させた。高機能化の内容は，メモリ容量増加に伴う PIC プロセッサコアのカスタマイズ，タイマ，CCP（キャプチャ，コンペア，パルス幅変調）モジュール，シリアルコントローラ，A/D コンバータの実装，デバッガのカスタマイズおよび GUI 化である。

回路設計は全て VHDL（回路記述言語）により行った。

2. 1　エミュレータ　　開発済みのエミュレータ（PIC16F84A 対応）では，プログラムメモリの容量が 1 K×14bit である。しかし，PIC16F876 と PIC16F877 では，プログラムメモリの容量が 8K×14bit と増加する。そこで，ユーザーのプログラムを書き込むメモリ領域のメモリ容量を 8K×14bit とし，デバッガを書き込むメモリ領域のメモリ容量を 1 K×14bit とした。そして，この二つのメモリ領域にアクセスするために，アドレスの長さを従来の 13bit から 14bit にした。具体的な構成は，図 1 のようになる。

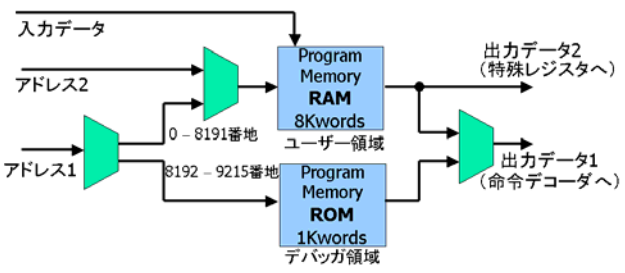

図 1．プログラムメモリの構成

アドレス線は，アドレス１とアドレス２がある。RAM または ROM 内のプログラムを実行する時はアドレス１が使用され，アドレス１の値が 0 番地から 8191 番地の場合は RAM, 8192 番地から 9215 番地の場合は ROM にアクセスする。そして，プログラムメモリから読み出されたデータは，出力データ１として命令デコーダに出力される。RAM に制御プログラムを書き込む時は，アドレス２や入力データという信号線が使用される。

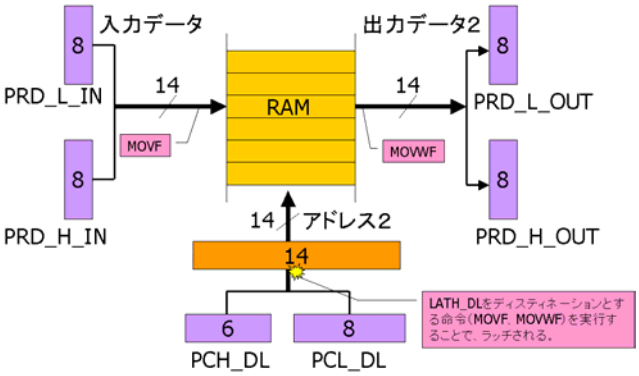

図２．プログラムメモリ用の特殊レジスタ

アドレス２と入力データ，出力データ２は，図２に示す６つの特殊レジスタに接続されている。PIC16F84A 対応のエミュレータでは，プログラムメモリにアクセスするアドレスの信号が 13bit であったが, PIC16F876 および PIC16F877 ではメモリ容量の増加に伴い，アドレスの信号が 14bit となる。そこで，図２に示すように PCH_DL のデータ長を 1bit 増やし, 6bit にする。「PRD_L_IN」，「PRD_H_IN」には RAM へ書き込むデータを，「PRD_L_OUT」，「PRD_H_OUT」には RAM から読み出したデータを書き込む。「PCH_DL」，「PCL_DL」にはアドレスを書き込む。そして，これら 6 つの特殊レジスタは，「File Registers」と呼ばれるレジスタ群の中にある。

アプリケーションから，このプログラムメモリにアクセスするときは，既存の命令コードを使用し，ディスティネーションにダミーとなる特殊レジスタ(LATH_DL)を指定することで可能である。

RAM に書き込むときは，「MOVF　LATH_DL」と書き，RAM から読み出すときは，「MOVWF　LATH_DL」と書く。

PIC16F84A の特殊レジスタは, TMR0 や PORTA, EEDATA など 15 種類のみであった。しかし, PIC16F876 と PIC16F877 では，タイマ１，タイマ２，CCP モジュール，シリアルコントローラ，A/D コンバータ，を実装している。そして，これらのモジュールを制御するための特殊レジスタが更に増加し，54 種類になる。そのため，必要な特殊レジスタを追加した。

また，デバッガに必要となる特殊レジスタについては，PIC16F876 と PIC16F877 の仕様書において未使用になっている部分に割り当てた。具体的には，表１のようになる。

PIC16F84A に対応したエミュレータを, PIC16F876 と PIC16F877 に対応するために, アドレスのビット長を 13bit から 14bit にする事は既に前述した通りである。そして，そのようなアドレスのビット長の拡張により，デバッガに対応した命令拡張が必要になる。

表１．デバッガ用特殊レジスタの割り当てアドレス

レジスタ名	取扱データの種類	アドレス
RCREG	シリアル通信 受信データ(デバッガ用)	8F
TXREG	シリアル通信 送信データ(デバッガ用)	90
COMSTAT	シリアル通信 ステータス(デバッガ用)	95
PRD_L_IN	プログラムメモリ書込データ(下位)	96
PRD_H_IN	プログラムメモリ書込データ(上位)	97
PRD_L_OUT	プログラムメモリ読出データ(下位)	9A
PRD_H_OUT	プログラムメモリ読出データ(上位)	9B
PCH_DL	プログラムメモリアドレス(上位)	9C
PCL_DL	プログラムメモリアドレス(下位)	9D
LATH_DL	プログラムメモリアクセス ダミーデータ	105
ADR_STA_L	ロジックアナライザ開始アドレス(下位)	107
ADR_STA_H	ロジックアナライザ開始アドレス(上位)	108
ADR_STP_L	ロジックアナライザ終了アドレス(下位)	109
ADR_STP_H	ロジックアナライザ終了アドレス(上位)	185
LGA_STAT	ロジックアナライザ ステータス	187
PCL_VAL	プログラムカウンタ(下位)の退避データ	188
PCH_VAL	プログラムカウンタ(上位)の退避データ	189
WSAV	ワーキングレジスタの退避データ	18E
STATUSSAV	PICのステータスの退避データ	18F
STACKL_SAV	スタック(下位)の退避データ	FB
STACKH_SAV	スタック(上位)の退避データ	FC

プログラムの途中で停止させるブレークポイント（BP）実行では，停止させる命令コードに対して，「GOTO *address (11bit)*」という命令コードを上書きする。*address* はモニタープログラムの先頭アドレスである。これにより，BP を設定した命令コードが実行された後は，モニタープログラムへ分岐する。そして，モニタープログラムに分岐する前の各種レジスタの内容を確認する。ここで，注意すべきことは，モニタープログラムへ分岐するコードは１行でなければならないということである。

PIC16F84A 対応時は，ユーザー領域とデバッガ領域共に，１Kword（10bit）であるため，アドレスは 11bit（最上位ビットはユーザー領域とデバッガ領域の選択ビット）必要であり，ちょうど「GOTO　*address (11bit)*」という１行のコードで済ますことができる。しかし，PIC16F876 と PIC16F877 では，アドレスが 14bit 必要となり，これまでの方法が使用できなくなってしまう。そこで，新規に命令「GOTO_MTR」を追加することにした。

GOTO_MTR のオペコードは，「00000000000001」（２進数）であり，これを実行すると，直前のプログラムカウンタを退避させ，「10000000000000」（２進数）に分岐する。もし，GOTO_MTR 実行時の分岐先のアドレス値を変更する場合は，PIC プロセッサコアの VHDL ソースを変更する。

この命令の使用例は，次のようになる（図３）。

１）ブレークポイント（BP）を設定する行の命令コードを退避させ，そこに GOTO_MTR を上書きする。

２）GOTO_MTR の実行により，プログラムカウンタ値の退避と MonitorProgram への分岐を行う。そこで，ユーザーのプログラムが停止する。

３）その後，退避させておいた命令コードを元に戻し，逆アセンブル表示や，レジスタ参照等の処理を実行する。

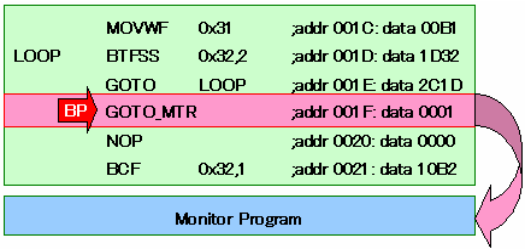

図３．GOTO_MTR の使用例

PIC プロセッサコアの周辺モジュールである，タイマ１，タイマ２，CCP モジュール，シリアルコントローラは，PIC の開発元である Microchip 社から出ているデータシートの仕様に準拠して設計した。

A/D コンバータについては，データシートの仕様に準拠せず，機能を制約して設計した。PIC16F876，PIC16F877 に実装されている A/D コンバータは逐次比較型であり，10 ビットの分解能および 8 チャンネルのポートをもつ。A/D コンバータの設計においては，ユーザーが A/D コンバータの全てのチャンネルを使用することは稀であると判断し，当該エミュレータでは３チャンネルをサポートし，分解能は 8 ビットとする。そして，A/D コンバータの精度を向上させるために，本来の逐次比較型から⊿∑型に変更する。⊿∑型は，回路が簡易であり，最大 24 ビットの高分解能を得ることができる。図４は開発した A/D コンバータの回路図である。A/D コンバータは，主に加算回路，積分器，量子化器から成る⊿∑変調器と，デシメーション・フィルタで構成され,⊿∑変調器をオペアンプと汎用ロジック IC 等で設計し，ローパス・フィルタとデータ間引き器の機能をもつ回路のデシメーション・フィルタを,10MHz のクロック信号生成回路と共に FPGA の中に実現した。この回路の出力結果は，特殊レジスタの ADRESL レジスタに保存される。また，A/D 変換が終了すると，ADCON0 レジスタの２ビット目の GO/DONE がクリアされ，PIR1 レジスタの６ビット目の割込み要求フラグビット ADIF がセットされる。

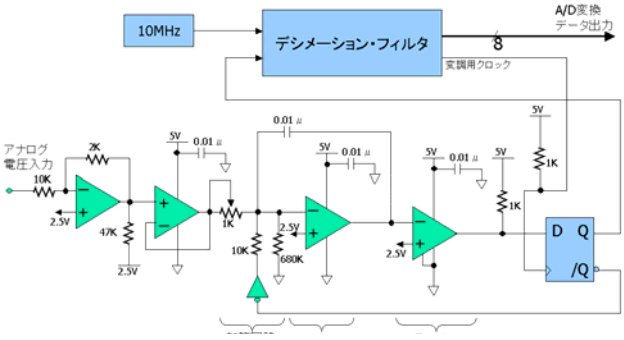

図４．⊿Σ型 A/D コンバータの回路図

2. 2　ロジックアナライザ　PIC はプログラム用の ROM をデバイスに内蔵しているため，ROM のアドレスと命令コードを電気信号として取り出せない。そのため，波形入力形ロジックアナライザが使用不可である。一方で，開発言語がアセンブラの場合は，PIC メーカのツールでプログラムの実行順をソフトウェア的にシミュレーションできる。しかし，開発言語が C 言語 になったとき，PIC メーカーのツールではシミュレーションできない。そこで，プログラムソースコードの実行順を言語レベルでデバッグ及び解析するロジックアナライザが必須となる。

この機能を実現するために，プログラムを実行したときに，アドレスと命令コードを出力するように PIC の内部構造をカスタマイズした。そして，出力されたアドレスと命令コードをメモリにストックするための制御回路と，メモリにストックされた内容をシリアル送信するための制御回路を設計した。この制御回路では，開始アドレスと停止アドレスを設定することにより，そのアドレス間の命令コードをメモリにストックすることができる。また，ロジックアナライザの制御は，PIC プロセッサの内部に新規に設置する特殊レジスタにより行う。

2. 3　デバッガ　エミュレータは，プログラムの開発を効率よく行うための種々のデバッグ処理を，パソコン側からのコマンドによって行う。そのときに，デバッガというソフトウェア・ツールが必須となる。そこで，プログラムのデバッグに必要となる主な機能（プログラムファイルのロード，メモリ内容の表示，プログラム実行など）をサポートするデバッガのプロトタイプ版（MS-DOS 上で動作するタイプ）を開発した。デバッガの開発に当たっては，デバッガ処理のほとんどをパソコン側で行うようにし，エミュレータ側での処理を少なくするように配慮した。即ち，デバッガを分割配置し，通信回線（RS232C）で接続し，両者の協調動作として各種機能を実現させた（図５）。これにより，殆どの処理をパソコン側が負い，エミュレータ側ではわずか３種の基本コマンド（メモリの読み込み，書き込み，ステータス情報の通信）の処理で済ませることが可能になり，エミュレータ側の負荷を軽減させることが出来た。

また，ロジックアナライザの制御を行うために，開始アドレスと終了アドレスを指定する「A」コマンドを新たに用意し，アプリケーション上でロジックアナライザのアドレス範囲を指定できるようにした。100 番地から 107 番地までのプログラム動作を解析したい場合は，「A 100, 107」とコマンドを打ってから，「G」コマンドでプログラムを実行することにより，指定したアドレス範囲の実行経過をロジックアナライザ内部のメモリに蓄積し，シリアル送信できる。

また，MS-DOS プロンプト上のコマンドラインによる操作よりも，Windows 上の GUI 操作の方が容易であることから，最後にデバッガの GUI 化を行った。

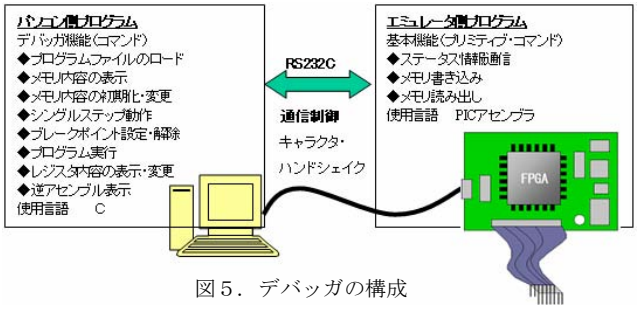

図５．デバッガの構成

3. 動作検証の結果

PIC16F876 と PIC16F877 に対応するために，プロセッサコアの拡張と，タイマ等の設計を VHDL で行った。そして，個々の回路について，FPGA ボードを用いて動作検証を行い，正常動作を確認した。その後，全ての回路を統合し，エミュレータとしての動作検証を行った。

PIC16F876 と PIC16F877 に対応したエミュレータを開発する際に拡張した PIC プロセッサコア，および機能追加したタイマ，CCP モジュール等の動作を同時に確認するためのプログラムを C 言語で作成した。そして，一般的に使用されている C コンパイラ(CCS 社)と，統合開発環境 MPLAB（Microchip 社）を用いて実行コード（HEX ファイル）を生成し，開発したエミュレータに書き込んで動作させた。

プログラムの内容は，①タイマ２と CCP モジュールによる PWM 制御で LED の点灯，②タイマ１による割込み処理で LED を点灯，③カウント値の LCD 表示，④シリアル通信，である。

図６はエミュレータの動作検証を行ったときの環境である。図中の左上の FPGA ボードがエミュレータであり，右下のボードがターゲットである。そして，CCP モジュールの PWM 波形の出力先は LED①になっている。また，タイマ１による割り込み処理では LED②を３つ同時に点灯させている。シリアル通信は FPGA ボードの左上で行い，LCD はターゲットボード上のものを用いた。

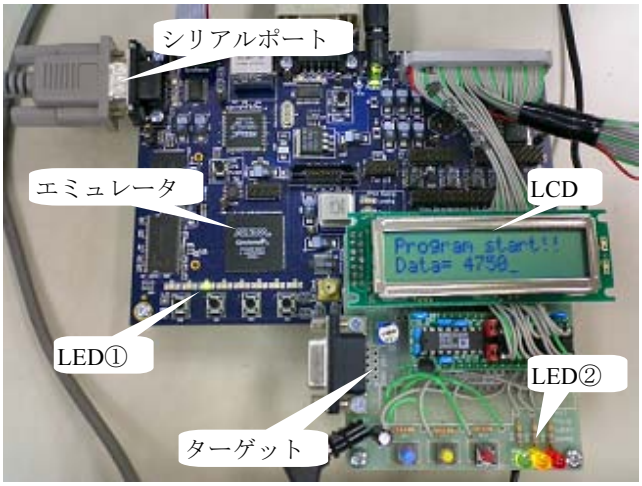

図６．動作検証時の環境

エミュレータの機能について確認したときのプロトタイプ版デバッガの様子が図７である。「R」コマンドで先に生成した実行コード all_test.hex をエミュレータに書き込み，「BS」コマンドでブレークポイントを 193 番地と 20A 番地に設定している。その後，「G」コマンドでエミュレータを始動し，193 番地で止まったときのレジスタの状態と，エミュレータの動作を確認した。そして，その他の各種コマンドによる動作も確認した。そのコマンド一覧を，表２に示す。

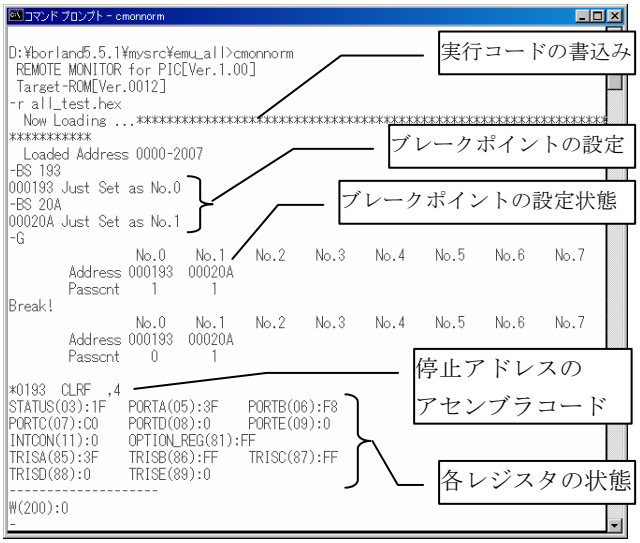

図７．デバッガの動作

表２．エミュレータのコマンド一覧

コマンド	説明	使用方法
R	:Hexファイルからプログラムの読出しとロード	R ~.hex
G	:プログラムの実行	G | G address
B	:ブレークポイントの設定と解除	B(参照) | BS address(設定) | BR No(解除No=0~7)
A	:ロジックアナライザの実行	A start_address, stop_address
T	:プログラムのトレース	T | T step_number
F	:メモリクリア	F
X	:主要レジスター括参照	X
M	:各レジスタ参照と変更	M register_address | M register_address, value
?	:コマンド一覧表示	?
Q	:デバッガの終了	Q

4. まとめ

PIC16F84A，PIC16F876，PIC16F877 に対応したエミュレータを FPGA により実現した。動作検証を行った結果，既存のエミュレータと同様の機能を確認した。従って，ユーザーが慣れ親しんだ C コンパイラやアセンブラなどの開発環境をそのままにして，今回開発したエミュレータを使用することができる。特に，プログラムの実行経過を C 言語ソースコード形式で出力するロジックアナライザ機能は，ユーザーのデバッグ作業の負担を軽減する効果が期待できる。更に，エミュレータの機能を VHDL で記述しているため，VHDL ソースコードを変更することにより，FPGA デバイスを変えることなく，PIC16F84A，PIC16F876，PIC16F877 以外のターゲットに柔軟に対応することが可能である。

（平成 18 年 10 月 25 日受付，平成 18 年 11 月 28 日再受付）

文　献

(1) Microchip Technology inc. : MICROSOLUTIONS，(July.2003)
(2) 産業技術連携推進会議 情報・電子部会 組込み技術研究会 :「組込みシステム開発事例集」，工業調査会, pp. 102-107 (2006)
(3) 森岡澄夫 ：「HDL による高性能ディジタル回路設計」，CQ 出版 (2002)
(4) Microchip Technology inc. : PIC16F87X Data Sheet , (2001)