論　文

擬似入出力装置を用いたプログラム制御機器デバッギングシステム*

中村敏行** 小笠原光孝*** 大川清人****
保刈明彦*** 斉藤　稔****

Abstract

Recently, program controlled equipment (e.g., intelligent terminal) has been widely used. However, there has not been much progress in debugging techniques for developing them.

The most difficult problem is how to find whether trouble comes from software or hardware error.

The method based on the concept of hardware-software convergence is effective in solving the problem because the debugging sphere is limited clearly by using this technique.

Based on this idea, we have developped an original hardware debugging aid system equipped with pseudo I/O memory which responds to I/O commands, a hardware oriented interrupting system, a simplified interface with the system under test and a powerful set of debugging commands.

In this paper, the functional feature and the effect of this debugging aid system are described.

1. まえがき

マイクロプロセッサの出現及び普及は，各方面に種種な形で大きな影響を与えている．ハードウェアに対する最大の影響は，ワイヤードロジックからプログラムロジックへの転換であろう．プログラムロジックは，ワイヤードロジックの行き詰りを打破する多くの利点を持っている．すなわち，フレキシビリティに富み，同一ハードウェアのままで仕様の変更，機能の拡張等に対応することができる．またハードウェア自身も，CPU，メモリ，入出力ハードウェアといった形で標準化され，設計の単純化，合理化が図れる．

一方，これらプログラムロジックの問題点は，汎用機の応用ソフトウェアとは趣きの異なるハードウェア制御ソフトウェアを作成し，これをハードウェアと一体になった状態でデバッグするのにかなり困難を伴うということである．

すなわち，従来作成されるソフトウェアは，汎用機の応用ソフトウェアであり，ハードウェアとの係り合いは非常に限られた範囲でしかなかったが，ハードウェア制御を行うプログラムは，ハードウェアとの干渉が主要部分であり，ハードウェアを常に意識して作成しなければならない．デバッギングにおいても，従来のようにソフトウェアだけ分離して，例えばシミュレータなどでデバッグすることを考えても，実時間でのハードウェアとの干渉を充分に組み込めないため，充分なデバッグができないのが実情である．むしろ，このようなハードウェア制御プログラムの場合は，ハードウェア，ソフトウェアを一体にした状態で，システムとしてデバッギングを行う必要がある．しかし，従来，上に述べた意味でのシステムデバッギングを対象

---

* A Debugging System for Program Controlled Device Equipped by Pseudo I/O Unit by Toshiyuki NAKAMURA (Computer Devision, Mitsubishi Electric Corp.), Mitsutaka OGASAWARA, Akihiko HOKARI (Computer Works, Mitsubishi Electric Corp.), Kiyoto OHKAWA, and Minoru SAITO (Computer Laboratory, Mitsubishi Electric Corp.).

** 三菱電機(株)電子計算機事業部
*** 三菱電機(株)計算機製作所
**** 三菱電機(株)計算機研究部

としたデバッギングエイドには適切なものが容易に見当らず，デバッギングの効率が必ずしもよくなかった．そこで，筆者等はマイクロコンピュータ式デバッギングエイドに，筆者等の創意による擬似入出力装置等のハードウェアを付加してシステムデバッギングの行えるデバッギングエイドを完成し，これによりハードウェア制御プログラムの作成，デバッギングの能率を向上したので，ここにその概要を述べ，読者の参考に供したい．

2. システムデバッギングエイド

2.1 設計思想

筆者等の開発したデバッギングエイドシステムの目指すところは，主として次の二点である．

(イ) システムデバッギングが行えること

(ロ) プログラム開発における能率向上をはかること

以下，この二点の主題について，その意義，従来の装置における実現の程度及び実現方法について述べる．

(1) システム動作状態でのデバッギング

プログラム制御機器のデバッギングのむずかしさは主としてシステム動作におけるハードウェアとソフトウェアの相互干渉の部分にある．従来のデバッギングエイドはこの点が極めて不満であり，いわばそのほとんどが従来からのソフトウェアのデバッギング能力しか具備していなかったと考えられる．それに対し，この点を改良したデバッギングの方式[2]を採り入れた開発用システムも発表されているが，後述するような理由で，これも充分とは言い難い．すなわち，そこで示されているデバッギングの方式は擬似システム構成（論理ターゲットシステム）の状態で，システムプログラムのデバッギングを行い，以後ハードウェアをひとつずつ置換しながらチェックを繰返し，最後に物理ターゲットシステムに到達するという方式であるが（**Fig. 1** 参照），この種の開発用システムの不満な点は，論理ターゲットシステムを構成する擬似システムコンポーネントの範囲が狭く，いわゆるI/O 機器の範囲に限られており，プログラム制御機器として必須と思われる外部ハードウェアの制御をそのままでは模擬できないことである．本システムはこの点に注目し擬似I/O装置と呼称されるハードウェアを開発すると共に，被テストシステムのハードウェアを容易に接続しうる方式を採用し，広範囲のターゲットシステムに対しシステムデバッギングを可能としている．

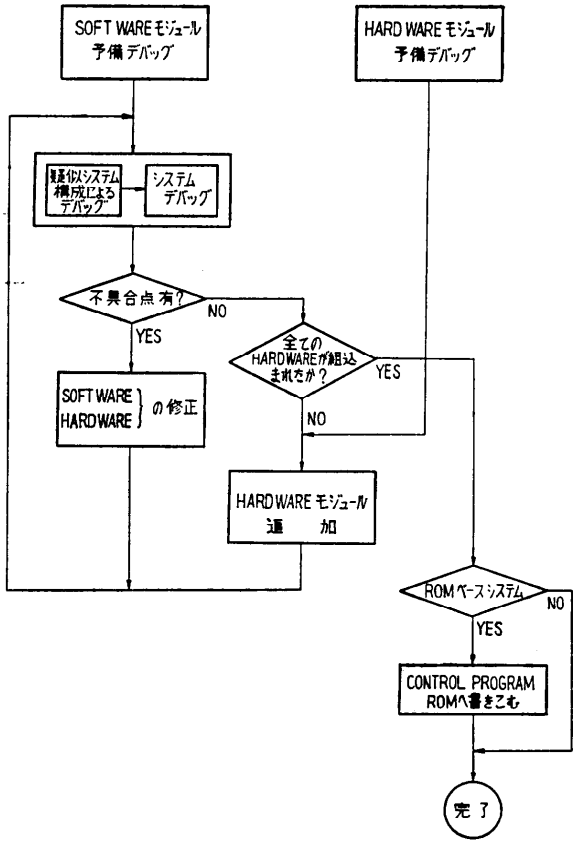


Fig. 1 Procedure of debugging for program controlled system.

擬似I/O装置とは，論理的な存在の有無を指定できる擬似入出力インタフェースレジスタ群であり，これにより自由な構成の論理システムが構築できる．

論理システム動作におけるデータ転送は上記レジスタ群との間で行われ，データの変更，読み出しはプログラムで行うことができる．更に，I/O 命令に対する応答はハードウェアであるため実際のハードウェアにおけるI/O命令実行時間と同じであり，実システムに対する模擬度が高く，多重割込み等を含む複雑な制御プログラムのデバッギングには極めて有利となっている．

被テストシステムのハードウェア結合における問題点は，I/O バスの論理的，物理的差異，プログラムメモリ空間の競合，割込処理機構の競合，その他プロセッサ素子固有の特定メモリアドレスの競合（例えば，

インテル社 8080 及び相当モデルにおける「0番地」)などである.

本システムではこれらの点に関し，バス方式の統一標準化，デバッギングシステム用システムプログラムのコンパクト化，特殊ハードウェアによるプログラム割込み等の方法で対処している.

擬似I/O装置及びこれらハードウェア結合手法の詳細はハードウェアの項で述べる.

(2) プログラム開発能力の向上

プログラムデバッギングにおいて，ブレークポイント，スナップショット，インストラクショントレース等の機能が極めて有効なのは言うまでもないが，マイクロコンピュータのレベルではプログラム容量等の点で，これらをすべて装備しているものは少ない.

本システムでは，前項で述べたプログラム割込みの機能をこれに応用し，比較的小規模のプログラムでバックアップシステムのシミュレータ並みの機能を実現している. これらの詳細な仕様についてはソフトウェアの項で述べる.

2.2 システムの概要

前節で述べたように，本システムはシステムデバッギングを主目的としたシステム構成を持っており，その心臓部となる擬似I/O装置と，被テストシステムとの結合を可能にするプログラム割込発生ハードウェア及び双方向性バスバッファが，本システムのために開発したマイクロコンピュータに接続されている. なお，

Table 1 The specification of debugging-Aid system.

モジュール	項 目	仕 様
CPU	CPU チップ	M 58710S(i-8080 コンパチブル)
メモリ	RAM ROM	4 kB×3 4 kB (BOM 領域)
I/O 機器	システムタイプライタ フレキシブルディスク	プリント 33文字/秒 リーダ 55文字/秒 パンチ 30文字/秒 ～2台
制御パネル	実行制御	スタート/ストップ/コンティニュー. 命令ステップ. アドレスサーチ
	メモリ制御 レジスタ制御	ストア/ストアネクスト エクザミン/エクザミンネクスト ロード/エクザミン
擬似I/O装置	取扱い可能ポート モード I/O 命令実行制御	IN 256ポート，OUT 256ポート マニュアル (SW/テープ表示) オート (プログラム制御) I/O 命令ステップ/連続
I/Oバス	バス信号本数 同期方式	53 (アドレス.16. データ16，割込16，制御4) INTER LOCK/ONE WAY (両方式可)

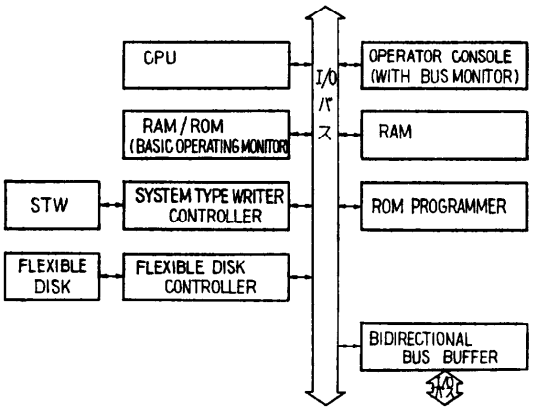


Fig. 2 The configuration of DEBUGGING AID based on μ-COMPUTER.

このマイクロコンピュータは，インテル社 8080 または相当モデル (以下，8080 と総称する) を CPU として使用するよう設計されている.

マイクロコンピュータとしては，4k バイト×3 ブロック (標準) のユーザーメモリ領域を持ち，システム入出力機器としてインクジェットタイプのシステムタイプライタを備えている. その他，プログラム作成及びデバッギングの便を図るためフレキシブルディスク装置，ROM プログラム装置などが接続されている. システムの構成を **Fig. 2** に示し，概略仕様を **Table 1** に示す.

2.3 ハードウェア

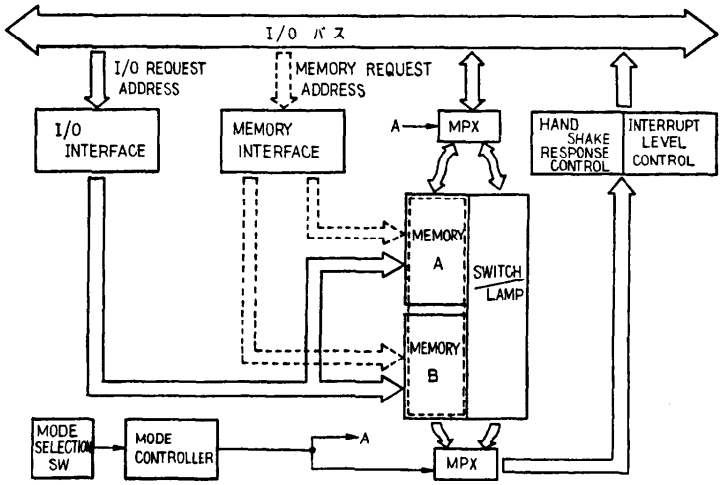
本システムの特徴である擬似I/O装置と，被テストシステム接続のためのハードウェアによる処理 (割込処理機構競合防止策及び「0番地」競合防止策) について述べる. マイクロコンピュータの部分に関しては，一般のものと大きな違いはないので説明は省略する.

なお，ハードウェアの規模としては，全体でおおむね 700 個程度の IC を使用した装置である.

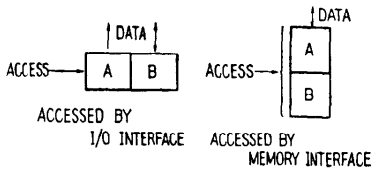
2.3.1 擬似 I/O 装置

(a) 概 要

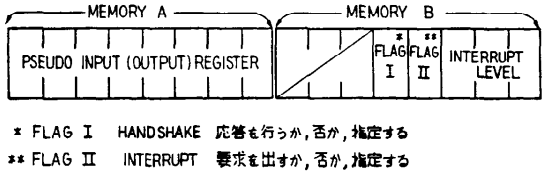
前述したように本システムのために新たに開発された擬似入出力装置は，簡単に言えば擬似入出力インタフェースレジスタ群であり，実質はメモリである. そして 8080 を使用した CPU の論理的に存在するすべての I/O ポートに対応するだけの数が用意されている (8080 においては，IN 256，OUT 256，計 512 である). この I/O レジスタ群は，プログラム実行時には I/O レジスタとして動作し，自動的に CPU とデータの授受を行う. データのセット，出力されたデータの



(a) The configuration of PSEUDO I/O DEVICE



(b) The memory configuration corresponding to access mode



(c) The BIT configuration of PSEUDO I/O DEVICE MEMORY

Fig. 3 Pseudo I/O device.

取り出し等は，プログラム停止時にマニュアルで，または専用割込処理ルーチン中で自動的に行うことができる．

(b) 特　徴

この擬似入出力装置の特徴は，次の通りである．

(イ) ハードウェアであるため，ソフトウェアオーバーヘッドは存在せず，I/O 命令の実行に対し，リアルタイム応答ができる．

(ロ) 個々の I/O ポートに対応して，論理的存在の有無，割込発生の有無，割込レベル，データの設定ができる．

(ハ) パラメータの設定に関し，毎回オペレータが設定しうるマニュアルモードから，プログラムですべて実行されるオートモードまで，4つのモードがあり，用途に応じて使い分けできる．

(ニ) プログラム実行を I/O 命令ごとに止めてチェックできる，I/O 命令ステップが可能．

(ホ) レジスタ群はメモリ素子で構成されているため，多数の入出力レジスタを用意しているにもかかわらず，極めてコンパクトなハードウェアとなっている．

(ヘ) レジスタ群を構成しているメモリ領域は，メインメモリ上にマッピングされているため，BOM (Basic Operating Monitor) により，容易にこのレジスタ群の内容を読み書きすることができる．

(c) ハードウェア構成

Fig. 3 (a) に，擬似入出力装置の概略ハードウェア構成を示す．中心部は，レジスタ群を構成するメモリA，メモリBであり，これに，プログラム実行中に応答を行うための I/O インタフェース及び，レジスタ群のデータを読み書きするための I/O インタフェースが付いている．モードコントローラにより，各種モードに応じた制御がなされる．マニュアルモードの場合には，スイッチランプ群がメモリと置き換えて接続される．ハンドシェイクレスポンスコントローラは，I/O 命令ステップを行う場合の制御を行う部分である．

Fig. 3 (c) は，擬似入出力レジスタのビット構成であり，擬似入出力データレジスタ (メモリA) と，制御レジスタ (メモリB) とからなる．制御レジスタ

の状態により，論理的存在の有無，終了割込の有無とレベルが決定される．

（d） 動作モード

マニュアル，プリセット，セミマニュアル，オートの4つのモードがある．

マニュアルは，スイッチ，ランプ群で入出力レジスタを模擬するもので，同時には，1I/O ポートしか模擬できない．プリセットは，レジスタ群のデータにより自動的に実行され，オペレータは介入できない．セミマニュアルは，マニュアルとプリセットの組み合せで，任意の1I/O ポートのみ設定スイッチ，表示ランプが接続され，オペレータが介入できる．オートは，プリセットと同様であるが，I/O 命令実行後，実行中のプログラムに割込みをかけ，擬似I/O処理ルーチンへ分岐する．擬似I/O処理ルーチンでは，通常レジスタ群のデータ変更または読出しを行う．

2.3.2 被テストシステム接続のためのハードウェア対策

（a） 概 要

2.1 で述べたように，ハードウェア結合のためにはいくつかの問題があるが，ここでは割込処理機構の競合防止と「0番地」競合防止の方策について述べる．

この二つの防止策に対しては，いずれもプログラム割込発生ハードウェアと称するものを用意し，システム割込要求時及びパワーオン時にこのハードウェアを起動してプログラム割込を発生させ，それぞれ割込処理エントリアドレス及び「0番地」を経由することなく所定のアドレスにプログラムを分岐させている．

（b） プログラム割込発生ハードウェアとその動作

プログラム割込発生ハードウェアの構成図を **Fig. 4** に示す．図を基にして動作概要を述べる．まず，起動要因（例えば実行中プログラムのストップ）が生ずると，次の命令フェッチサイクルの先頭でメモリ出力の禁止信号が出される．次いで発生した要因に対応する処理ルーチンを CALL する命令が，本ハードウェアからデータバスにのせられる．CPU はこれを取り込んで実行するため要因に応じたアドレスにプログラムが分岐する．なお，分岐アドレスはハードウェアであらかじめ決められている．

（c） 割込処理機構の競合防止策

デバッギングエイドシステムとして必要な割込みはパワーオンリセット割込，プログラムストップ割込，擬似 I/O 割込及びインストラクションステップ割込（インストラクショントレース等に使用）の4つがある

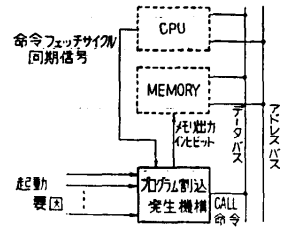


Fig. 4 Program interrupt generator.

が，これらはすべて前述したプログラム割込で実行されている．このため，CPU における割込要求線，これに関連するハードウェア及び割込エントリアドレスはすべてデバッギングエイドには無関係であり，被テストシステムで自由に使用することができる．

（d） 「0番地」の競合防止

「8080」はリセットを行うとプログラムカウンタが「0番地」に設定され，「0番地」のメモリがアクセスされる．通常被テストシステムとデバッギングエイドシステムでは「0番地」付近に格納するプログラムは当然異なるため，デバッギングエイドではこの部分を避けておく必要がある．このデバッギングエイドでは，前述したようにリセット時に生ずるリセット割込をプログラム割込発生要因としてプログラム割込発生ハードウェアで実行させているため CPU は「0番地」の命令を実行することなく，リセット処理ルーチンの実行を開始する．

2.4 ソフトウェア

本システムのソフトウェアを特徴づけるのは，ベーシックオペレーティングモニタと，この管理下で動作するデバッギング機能を提供するデバッギングプログラムである．このプログラムの概略仕様を **Table 2** に示す．

この他，デバッギングエイドとしての機能を発揮さ

Table 2 The specification of debugger program.

分 類	項 目	指定可能数	機 能
デバッギング機能	ブレークポイント	8点	指定したアドレスを実行するとプログラムの実行をストップし，モニタへもどる
	スナップショート	8点	指定したアドレスを実行すると，PC及び全レジスタを OPEN し，再びプログラムを続行する
	インストラクションントレース	4領域	指定した領域内の命令を実行するごとに，PC 及び全レジスタを OPEN しつつ，プログラムを続行する
デバッギングテーブル制御機能	テーブル表示	—	上記三種の設定点をテーブルに表示する
	テーブルクリヤ	—	上記テーブルの内容を消去する（デバッグ機能リセット）

せるため次のようなソフトウェアが用意されている.

(1) 各種ユーティリティプログラム

(2) セルフアセンブラ

(3) 擬似I/O装置用ユーティリティプログラム

3. システムデバッギングエイドの有効性

3.1 定性的評価

本デバッギングエイドを用いることにより初めて可能になった事項及び従来より改良された事項について述べ，本システム実用上の有効性を明らかにしたい.

本システムにより新たに可能になった事項は

(イ) システムデバッギング機能

(ロ) I/O 命令のリアルタイムシミュレーション

であり，改良された事項は

(イ) I/O 命令のデバッギングの高能率化

(ロ) ソフトウェアデバッギングの高能率化

(ハ) 被テストシステムハードウェア接続条件の緩和である.

以下，これらにつき更に詳細に述べる.

(1) システムデバッギング

前章でも述べたようにプログラム制御機器はハードウェア，ソフトウェアが相互に干渉して動作するが故に複雑になっている. 何故なら，相互に干渉しあうため，表に出ている現象とその原因が全く別の所にある場合が多いからである.

ところが，擬似論理システムから出発して，まずシステムプログラムのデバッギングを行い，それからハードウェアを1個1個置換して行くシステムデバッギングの手法を用いるなら，バグの存在位置は通常置換した部分に限定される. このため，この手法を採ることにより，バグの存在位置がハードウェアかソフトウェアか定め難く，バグを追ってハードウェアとソフトウェアを行きつ戻りつするという非能率な状態から脱却できる. これが，本システムデバッギングエイドの最大の利点である.

(2) I/O 命令のリアルタイムシミュレーション

汎用計算機の応用ソフトウェアにおいては，一般にI/O 命令のシミュレーション，特にリアルタイムシミュレーションの必要性が無い場合が多い.

ところが，プログラム制御機器の制御ソフトウェアの場合には，機械的あるいは電気的に動作している部分をI/O動作として制御するため時間の要素が無視できなくなる. 特に前章で述べたように，多重割込が特定の順序で入ることを期待しているプログラムの場合など割込の順序，タイミングとメインプログラムの関係が問題になる. 従って，このような場合にI/O命令のリアルタイムシミュレーションが是非共必要になる.

(3) I/O 命令デバッギングの効率化

前章で述べたように擬似I/O装置には4つの動作モードがあり，これらを使い分けることによりI/O命令に依存するブランチトレースの機能から個々のI/Oデータの値による詳細なアルゴリズムのチェック機能まで実現できる. このためバッチ処理のシミュレーションにおいても効果がある上，インタラクティブにもI/O命令をシミュレートできるため，一般的に見てI/O命令とそれに伴うブランチの多い制御プログラムのデバッギングには特に有効である.

(4) ソフトウェアデバッギングの高能率化

従来バックアップシステムのみで得られていた多点ブレークポイントあるいはスナップショットの設定，多領域インストラクショントレース等の機能を，マイクロコンピュータの上で実現している. このためシステムデバッグ用に手元にあるマイクロコンピュータ上でバックアップシステム並みの機能が使えることになり，ソフトウェアデバッギングの上で極めて有利となっている.

(5) 被テストシステムハードウェア接続条件の緩和

標準バスを設定してインタフェースを標準化し，かつ割込及びメモリ特定アドレスの競合を防ぐためのハードウェアを具備しているため，被テストシステムのみでなく，その一部であるハードウェアモジュールを任意の順序で接続してテストすることが可能であり，この意味で本デバッギングエイドはシステムデバッギング，ソフトウェアデバッギング及びハードウェアデバッギングの三役を兼ねることができ，適用範囲の広い装置と言える.

3.2 定量的評価

デバッギングに要する時間を，テスト時間，机上作業時間，修正作業時間の3つの部分に分け，各々の部分が全体に対して占める割合と，各々の部分における本デバッギングエイド使用による時間短縮率とからデバッギングにおける本デバッギンギエイドの効果を定量的に算出してみる. なお，机上作業時間は厳密に計測されていないのでここでは触れないが，本デバッギングエイドの場合には会話的にテスト，修正が行えるため，大幅に減少する方向にあることが確認されてい

る.

(1) 各部分の全体に対して占める割合

テスト時間と修正作業時間を合せて計算機使用準備及び実行時間とすると，汎用計算機のベーシックソフトウェア開発の場合には，計算機使用準備及び実行時間と机上作業時間の比は，現在の作成方式を前提として，これまでの経験によるデータに基づき，社内的には 3:7 の推定値を用いているが，マイクロプロセッサ応用機器の制御プログラムの場合は，前記の場合に比較してプログラムサイズが小さく複雑でないこともあって，今回の限られた事例によってではあるが，ほぼ 5:5 という値が得られている.

この場合のテスト時間と修正時間の比は，厳密には事例が異なればかなり異なった値をとることもあるかと考えられるが，本論での評価に限るとして平均的事例をもとに，ここでは 1:1 という値を仮定するものとする．すると，デバッギングエイドを使用しない場合には，テスト時間，机上作業時間，修正作業時間の比は，平均 1:2:1 という値になる.

(2) 実行における時間短縮率

本デバッギングエイドを使用した場合は，ホストコンピュータシステムを使用したバッチ処理の場合と異なり，計算機使用の待時間を要しない上，会話的に実行開始ができるため各種の準備作業が不要であり，更に擬似 I/O 装置により I/O 命令部分をインタラクティブに実行できるため，テスト結果を得るために要する時間は大幅に減少する．あるシステム開発について調査した平均実行時間は，シミュレータで 30 分，本デバッギングエイドで 20 分という結果が得られており，この場合では 1/3 の時間が短縮される.

(3) 修正作業における時間短縮率

本デバッギングエイドの場合，パネルから容易にパッチが可能なため，再アセンブル処理の必要性はかなり少なくなる．前述のシステム例においては，平均修正作業時間はシミュレータで 40 分，本デバッギングエイドで 20 分であり，1/2 の時間が短縮される.

(4) 総 合 評 価

以上述べた二点による時間短縮率は **Table 3** に示すようにそれぞれ 8.3%，12.5% となり，総合時間短縮率は約 21% となる．しかし，本デバッギングエイドにおいては命令トレースをインタラクティブに行えるため，本システムを使用することにより机上作業時間がかなり短縮できることもあって，これらを勘案すると総合短縮率は 30% 近くなるものと推定される．これは本デバッギングエイドの使用者が直観的な短縮率として 1/3 という値をあげているのにほぼ符合する.

Table 3 The effect of debugging aid system.

項 目	作業時間比率 (A)	本デバッギングエイドによる時間短縮率 (B)	短縮率 (A)×(B)	$(A)\times\left\{1-\frac{(B)}{100}\right\}$
テスト時間	25%	33%	8.3%	16.7%
机上作業時間	50%	0%	0%	50.0%
修正作業時間	25%	50%	12.5%	12.5%
	100%		20.8%	79.2%

なお，ターゲットシステムでのデバッギングとの比較に関しては定量的な値は得られていないが，ソフトウェア機能，パネル機能が充実している本デバッギングエイドの方が格段に効率が良いことは明らかである.

4. むすび

筆者等が開発したデバッギングエイドシステムについて，その特徴，特徴の基になるハードウェア，システムの概要及びシステムの有効性について述べた.

このシステムは，擬似 I/O 装置を備え，被テストシステムとのハードウェア接続上の制限の緩和を図っているため，プログラム制御機器開発上必須の技術と考えられるシステムデバッギングの手法を適用できる．このシステムを用いることにより，従来の 70% 程度の開発工数で制御プログラムのデバッグが可能であり，このシステムの開発工数 (35 人・月) から考えてかなり有効なシステムと考えている.

なお，このシステムの今後の改良点としては次の項目が掲げられる.

(イ) ソフトウェア製作の効率向上

このシステムはマイクロコンピュータベースであり，アセンブル等の能力は高くない．アセンブルのみ，バックアップの大形機でオフラインで行っても良いが，このシステムとホストコンピュータとを回線で結び，処理は大形機で行い，結果を返送させるという方式も考えられよう．この方式の利点は，ターンアラウンド及び処理の時間が短いことである.

(ロ) 擬似 I/O 装置オートモードのリアルタイム化

擬似 I/O 装置におけるオートモードは，メインプログラムへの割込みにより，データの読み書き，変更を行っているが，高速専用プロセッサ等の使用により，これもリアルタイム化し，本当の意味のリアルタイム擬似 I/O 装置としたい.

(ハ) リアルタイムトレース機能の追加

現在拡張版BOMの機能に含まれるプログラムトレースは，ソフトウェアオーバーヘッド時間を非常に多く必要とするため，プログラムロジックのクリティカルな動作はトレースできない．市販されているロジックアナライザを外部に接続し，あるいは内部に組みこんで，この種の機能を充実させてデバッグの効率化を図る必要がある．というのは，プログラムロジックの誤動作のかなり多くは，このクリティカル動作のあたりに存在するからである．

最後に，データを提供して下さった当社制御器制作所及び通信機製作所の関係者に深謝いたします．

参考文献

1) M. Ogasawara & K. Ohkawa: The Debugging aid System for Microprocessor-based System, Microprocessing & Microprogramming '76 EUROMICRO SYMPOSIUM, pp. 271～276.
2) Robert Garrow, Sterling Hou, James Lalley & Hop Walber: Microcomputer Development System Achieves Hardware Software Harmony, Electronics, May 29 '75.
3) William Davidow: The Coming Merger of Hardware and Software Design, Electronics May 29 '75.
4) 小笠原，保刈，斉藤：プログラム制御機器テスト装置，昭和51年電子通信学会全国大会予稿 No. 1267.
5) 山田，小笠原，水野：拡張デバッグ機能を持つマイクロコンピュータモニタ，昭和50年情報処理学会第16会大会予稿 No. 279.
6) 保刈，小笠原，斉藤：実応答を行う擬似入出力装置，電子通信学会電子計算機研究会資料EC-76-6.
7) 小笠原，保刈，斉藤：制御プログラムデバッギングにおけるハードウェアI/Oシミュレータの効用，昭和52年電子通信学会全国大会予稿S 1-6.
8) マイクロコンピュータ特集，日経エレクトロニクス，11月17日号，'75年.

(昭和52年3月18日受付)
(昭和53年1月18日再受付)