3V-4

V60リアルタイムOSにおける マルチタスクデバッガ

中本　幸一*　　永江　達也**　　古城　隆*
*日本電気(株)　　**日本電気通信システム(株)

1．はじめに

従来，組込み型システムでは，デバッグ手段として、インサーキット・エミュレータ（ＩＣＥ），又はモニタ等が用いられてきた．一方，Ｖ６０のように３２ビット高性能マイクロプロセッサを用いたシステムにおいては、機能の高度化，多様化が要請され，これに伴ってその機能を実現するアプリケーション・ソフトウェアの開発量の急速な増大が予想される．このため，従来からのデバッグ手法では，デバッグ効率の飛躍的向上は難しいと思われる．

Ｖ６０リアルタイムＯＳ［1］（以下Ｖ６０ＲＴＯＳと略する）では，このような事態に対処するために，マルチタスクの環境下で並行に動作する複数のタスク群からなるアプリケーションをデバッグするためのデバッガを提供している．本稿では，本デバッガの機能仕様，並びに実現方式について述べる．

2．従来手法の問題点

従来からのＩＣＥ，モニタ等のデバッグ手段を用いて，先に述べた大規模なアプリケーション・ソフトウェアをデバッグしようとすると，以下のような問題が生じる．

1)デバッグ手段として原始的である

Ｖ６０ＲＴＯＳ上のアプリケーション・ソフトウェアを開発するユーザは，Ｖ６０ＲＴＯＳで提供されているプログラミングモデル［2］（多重仮想空間によるタスク構成，ランデブ通信によるタスク間通信）を使ってアプリケーション・ソフトウェアを開発している．ところが，従来からのデバッグ手段では，これらのプログラミングモデルが直接サポートされないため，アプリケーション・ソフトウェアをデバッグする時に，タスクのメモリ内容やタスク間通信データの参照・変更のために，デバッグするユーザはＯＳの内部構造を知る必要がある．これは，デバッグ効率向上を疎外する．

2)高級言語によるデバッグ手段がない

Ｖ６０ＲＴＯＳのアプリケーション・ソフトウェアは高級言語（リアルタイムＣ言語［3］）により記述される．ところが，従来からのデバッグ手段の多くが，機械語レベルのデバッグ手段しか提供していない．このために，アプリケーション・ソフトウェアのデバッグの時に，高級言語プログラムの機械語への展開の知識が必要となる．これもデバッグ効率向上を疎外する要因となる．

3．マルチタスクデバッガの機能

2節で述べたデバッグ時の問題を解決し，アプリケーションソフトウェアのデバッグ効率を向上させることを目的として，Ｖ６０ＲＴＯＳでは，アプリケーション・ソフトウェア向きのデバッガを提供している．本デバッガは以下の機能を有している．

3．1タスク向きデバッガ

本デバッガは，Ｖ６０ＲＴＯＳの提供するプログラミングモデル・レベルでの以下のようなデバッグ手段を提供している．

1)タスク毎に独立した実行制御手段，データ参照機能

本デバッガでは，実行制御手段として，タスク毎に独立したブレークポイントの設定，Ｖ６０プロセッサのデバッグ機能［4］を利用したアドレストラップ　の設定が可能であり，これらの機能によりタスクの実行を一時的に中断することができる．更に，同一命令部を共有するタスクに対しても，ブレークポイントを設定できる．又，任意の時点でのタスクの有するデータの参照・変更が可能である．これらの機能により，Ｖ６０ＲＴＯＳの提供する多重仮想空間でのタスクのデバッグが容易になる．

2)タスク間通信を契機としたデバッグ手段

Ｖ６０ＲＴＯＳでは，タスク間通信メカニズムとして，Ａｄａのランデブ機能が提供されている．本デバッガでは，ランデブによるタスク間通信を契機としたデバッグ手段が提供されている．即ち，ユーザに指定されたタスク間のランデブのエントリ呼出し・ランデブの成立（アクセプト）・ランデブの終了時に，ランデブ通信データの参照・変更が可能である．又，どのようなタスクでランデブ通信が，どの時点で行われているかを示すための，

Multi-task Debugger on V60 Real-Time Operating System
Yukikazu NAKAMOTO*, Tatuya NAGAE** and Takashi KOJO*
*NEC Corporation
**NEC Communication Systems Ltd

トレース機能を有する．

個別タスクに着目して，タスクの実行状態，各種資源待ちキューの状態を知るための手段を有する．

3．2高級言語デバッガ

本デバッガでは，高級言語で記述されたプログラムを機械語でデバッグするというデバッグ時のセマンティックギャップを解消するために，リアルタイムC言語[3]を対象とした以下のような高級言語デバッガの機能を提供している．

a)プログラムソースファイル中での行番号指定，関数呼出し時指定，関数復帰時指定によるブレークポイントの設定，変数名によるアドレストラップの設定

b)タスク内データ，タスク間通信データのC言語レベルでの表示

3．3マルチウィンドウによるマルチタスクのデバッグ

本デバッガはマルチウィンドウ機能を有している．これにより，

a)タスク毎のデバッグ・セッションをマルチウィンドウのあるウィンドウに割りあてることにより，あるタスクのデバッグの際に他タスクのデバッグセッションが混在しないようにして，現在着目しているタスクのデバッグに集中できるようにする．尚，タスクのデバッグ・セッションのウィンドウへの割りあて方は任意である．

b)あるウィンドウでデバッグ・セッションを開きながら，別のウィンドウでソースファイルを見る．

など可能である．

3．4コマンドインタプリタとしてcsh/shを利用

本デバッガでは，コマンドインタプリタとして，UNIXのcsh/shを利用している．これにより，

a)デバッガ・コマンドの実行結果をファイルに蓄積できる．

b)デバッガ・コマンドのコマンドファイルにすることができる．

c)デバッガ・コマンドの実行結果を他のUNIXコマンド(awk,grep etc)とパイプでの結合による加工が可能である．トレースコマンドの実行結果など対象となるタスク・OSの資源が混在している時に，必要とする情報を抽出したい場合に，この機能は有用と思われる．

4．実現方式

本デバッガの実現の一例を図1に示す．本デバッガは，以下の構成要素からなる．

1)ユーザインタフェース(ホストマシン)

ホストマシンとして現在PC9800，ホストOSとして，PC-UX を用いている．

コマンド，ターゲットマシン管理プロセス，シンボルテーブル管理プロセスから構成される．

2)デバッガタスク（ターゲットマシン）

V60RTOS上で動作するスタティックタスクとして実現している．

3)V60RTOSカーネル（ターゲットマシン）

デバッガのための機能を追加している．

5．おわりに

本デバッガは上記機能の一部を実現し，試行している．今後，この経験を元に機能向上を図っていく予定である．

末筆ながら，本デバッガの開発に協力して頂いた方々に御礼申し上げます．

参考文献

[1]古城他:"V60リアルタイムOSの設計",第33回情処全大,1C-4,1986.

[2]南沢他:"V60リアルタイムOSにおけるタスク",同上,3V-1,1986.

[3]土屋他:"V60リアルタイムOSにおけるタスク間通信",同上,3V-2,1986.

[4]V60ｱｰｷﾃｸﾁｬﾏﾆｭｱﾙ,NEC,1986.

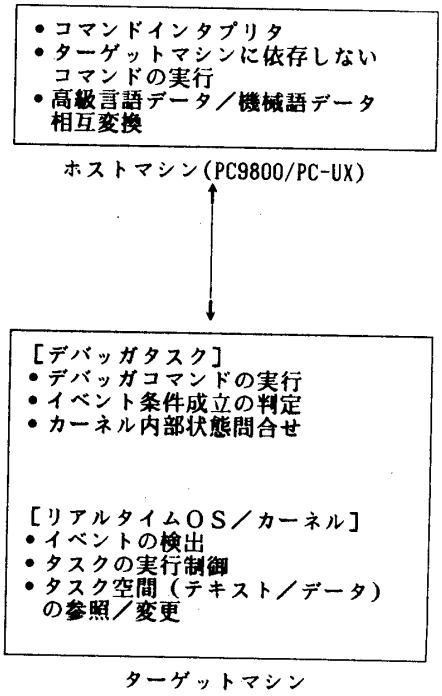

図1．マルチタスクデバッガの実現図