3V-3 V60リアルタイムOSにおけるフォルトトレラント機能

下島 健彦、 雷見 かよ子、 世良 孝文、 古城 隆、 小泉 正

日本電気(株)マイクロコンピュータソフトウェア開発本部

1. はじめに

本稿では、V60用のリアルタイムOSの耐故障性機能について述べる。

システムの一部に障害が発生してもサービスを継続できる耐故障性機能が、特に制御システム等の分野で必要とされてきた。このようなシステムでは、プロセッサ、メモリ、バス、ディスク等を複数使用して冗長系を構成し、異常発生時に自動的に障害部分を検出、切離すことにより耐故障性を実現している。しかしこのためには冗長なハードウェア構成が必要とされるため、従来それらの多くは専用システムとして実現されてきた。また応用分野も、高価な専用システムを用いてでも耐故障性が必要とされるような分野に限られていた。

最近では、こうしたハードウェアはマイクロプロセッサやVLSIメモリを使用する事で極めて安価に実現可能になってきており、またV60のような高機能なマイクロプロセッサでは、障害検出をサポートする機能がチップ上に組込まれ、わずかな付加的ハードウェアにより冗長系を構成できるようになってきた [1]。一方、マイクロプロセッサの処理能力が飛躍的に向上するにつれて、マイクロプロセッサを搭載したシステムが社会的にも重要度の高い分野で使用されるようになり、これに伴なってより高い信頼性が要求されるようになってきた。

本OSでは、耐故障性機能のうちハードウェア構成に依存した部分とそうでない部分を分離し、非依存部の処理を部品化して提供し、また依存部をユーザ記述可能とすることで、汎用のOSとして種々の冗長構成に対応できる耐故障性機能を提供している。本稿では特にリアルタイム核に組込まれた耐故障性機能を中心に報告する。

2. リアルタイム核に組込まれる耐故障性機能

一般に、耐故障システムでは、i. 障害の検出、ii. 障害部分の切離し、iii. 障害部分の修理・取替え iv. 再開処理というステップにより障害に対処する。このうち i. から iii. までのステップは主にハードウェアにより実現される部分であり、冗長系の構成方法に依存した部分である。本リアルタイム核では耐故障性の機能として iv. の再開処理の部分をサポートしている。

Fault Tolerance for V60 Real-Time Operating System

Takehiko SHIMOJIMA, Kayoko UTSUMI, Takafumi SERA, Takasi KOJO and Tadashi Koizumi

NEC Corporation

2.1 再開レベル

再開処理は大きく、障害の程度の判定、障害程度に応じたシステムの再開の2つのフェーズに分けられる。障害の程度は、メモリ上に残されたデータを再開後にどれだけ利用できるかによって、次の4つのレベルを設定している。

A. メモリ内容が全く破壊されている。
B. リードオンリーのメモリの内容(プログラムテキスト、バックアップデータ)は再利用できる。
C. ユーザタスクのデータを引継ぐことができる。
D. メモリ内容はすべて引継ぐことができ、再開処理前のCPUの内部状態もメモリにセーブされている。

障害程度の判定のメカニズムはシステムに依存していて、それをOSが規定してしまった場合、OSの適用性を低くしてしまう。本リアルタイム核では、障害程度の判定ロジックをユーザ記述可能にすることで様々なシステムへの対応を可能にしている。例えば障害判定回路を備えたシステムでは、判定ロジック中で回路の状態を読み出すことにより再開レベルを決定でき、また特定のハードを用いない場合、例えばシステム起動時にメモリの特定部分のチェックサムを計算しておき、判定ロジック中でこれを再計算することにより、ソフトウェアによりメモリの再利用の可否を決定することもできる。

システムの再開処理は上記4レベルに対応した処理を提供している。レベルAでは2次記憶よりシステムをブートし、カーネルとユーザタスクの環境はすべて初期化する。本OSではブートイメージをリードオンリーでメモリ上に保存しており、このバックアップイメージからシステムを再開するのがレベルBである。これにより2次記憶よりのブート時間を削除することができる。レベルCではユーザタスクのデータを引継ぎ、カーネルの環境、ユーザタスクの走行状態は初期化される。レベルDは前処理としてCPUの内部状態をメモリにセーブしておき、カーネルの環境、ユーザタスクの走行状態等をすべて引継いだ再開を行なう。また、再開処理の開始方法には、ハードウェアリセットによるものとシステムコールによるものの2通りを提供している。

2.2 再開処理の応用例

再開処理を実際の冗長構成システムに応用した例を示す。OSは2.1で述べた再開処理のみを提供しており、

系構成の管理や障害のロギング等はＯＳの外部で実現することで、種々の系構成への対応を可能にしている。

２．２．１　３重多数決論理（TMR）システム

ＣＰＵが３重化されたシステムを考える。このシステムではＣＰＵの障害の検出、切離しは多数決論理回路が行なう（図1-1)。障害の起ったＣＰＵを修理・取替えた後、そのＣＰＵと動作中の２つのＣＰＵの状態を一致させる処理（再同期化処理）としてソフトウェアサポートが必要である。これは次のステップで行なう。

1. ２つの正常なＣＰＵがシステムコールによりレベルＤ再開を起動し、正常なＣＰＵの内部状態をメモリにセーブする（図1-2)。
2. ３つのＣＰＵをハードウェアリセットする（図1-3)。
3. ３つのＣＰＵが同期してレベルＤの再開処理を行ない、1.でセーブした正常な状態をリストアする（図1-4)。
4. ３つのＣＰＵは再開処理起動システムコールからリターンし、通常処理を同期実行する。

一般に同期実行している複数ＣＰＵシステムで、障害復帰させたＣＰＵを他の正常なＣＰＵと再同期化させる際にはレベルＤ再開が有効である。再同期化処理はユーザ記述部を除き約 100μ秒かかり、これがサービスの中断時間になる。

メモリが３重化されているようなシステムでは、メモリの再同期化も必要であるが、これはメモリの実装状態を知るシステムコールと物理アドレスによりメモリリード／ライトを行なうシステムコールにより実現する。

２．２．２　デュプレックスシステム

スタンバイ系が同期動作していないデュプレックスシステムでは、プログラムによりチェックポイントを作り、障害により系切り替えを行なった後、最後に実行したチェックポイントまで処理を戻すことで、マクロな意味で処理を継続できる。本ＯＳ上では、系切り替え後にレベルＣでシステムを再開させることでこれを実現することができる。タスクのデータはチェックポイント毎に確定される安定なデータと、作業用のデータとに分けて設計する必要がある。各タスクはチェックポイントで自分の状態を安定なデータに書き込む。レベルＣ再開後、各タスクは先頭から実行されるが、このとき安定なデータを基に自分の状態を前回最後に実行したチェックポイントの状態に再設定することで、前回のチェックポイントから処理を再開できる。

３．ファイルシステムに対する機能

耐故障性を提供すべきもう１つのコンポネントはファイルシステムである。本ＯＳではＵＮＩＸ互換のファイルシステムをリアルタイム核の上位に実現する。耐故障性機能としては、デバイスレベルでのＮ重化をサポートする。これはＮ重化したデバイス全部に同一内容をライトし、リードはマスタ系から行なうもので、ファイル管理部とデバイスドライバの間に系構成に従って多重リード／ライトを行なう層を追加することで実現する。また、この層に対して系構成の変更を指示するためのシステムコールを用意している。ここでも、ＯＳがサポートする機能は多重系を構成するための部品にとどめ、系構成の管理等はＯＳの外部にタスクとして実現することで、様々なハードウェアに柔軟に対応できるようにしている。

４．おわりに

Ｖ６０リアルタイムＯＳにおける耐故障性機能について述べ、これをＴＭＲ、デュプレックスシステムに適用する例を示した。今回述べた機能によりＣＰＵ、メモリ、バス等の障害に対処することが可能である。今後の課題は、レベルＣ再開を利用したチェックポイントの機能を言語とＯＳのレベルで提供することであり、これにより記述性を向上させ、ユーザの設計負担を軽減することが期待できる。

参考文献

[１] V60アーキテクチャ・マニュアル NEC, 1986
[２] 古城 他："V60リアルタイムOSの設計"第33回情処全大講演集1C-4
[３] D.A.Rennels "Fault-Tolerant Computing - Concepts and Examples," IEEE Trans. Comput., Vol.C-33, no.12, pp.1116-1129, Dec.1984

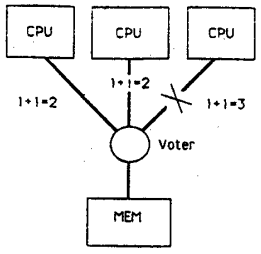

図1-1 多数決論理回路による障害の検出・切離し

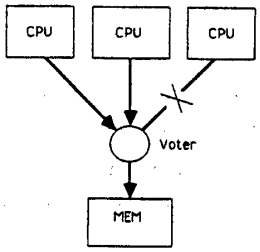

図1-2 レベルＤ再開によるＣＰＵの内部状態のセーブ

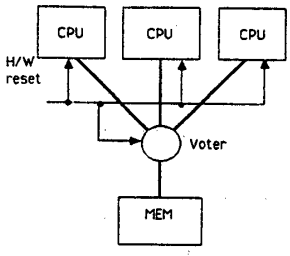

図1-3 ハードウェアリセットによる全ＣＰＵと多数決論理回路の初期化

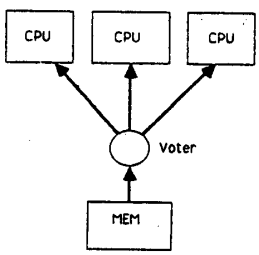

図1-4 レベルＤ再開により全ＣＰＵが同期して正常な内部状態を復帰し、通常処理を再開