システム設計上のデータ処理安全対策

——RAS とデュアル・システム——

川　原　　裕†

1. はじめに

コンピュータが企業内のみならず社会の基幹部分に情報処理システムとして根をおろすようになった今日，データ処理上の安全性の優劣が企業あるいは社会の諸活動に大きな影響を及ぼすことはいうまでもない．

近代社会における価値概念を“bits”の形でデータとして把えて本来の価値に応じた処理の安全性を保障するよう十分に考慮されなければならない．いいかえれば，システムはそのデータ処理上の安全性が犯された場合に生じるであろう損害の程度に応じて，何らかの保護対策が講じられてしかるべきである．

このことをアプリケーション・システム設計上の立場から考えれば，システムが取り扱う多種類のデータ個々の価値に応じた安全性設計を行なうことであり，すでにソフトウェアの分野においてこの概念はある程度確立されているといえよう．しかし，この解決をハードウェアに求めることはコスト／パフォーマンスの上で必ずしも得策ではない．したがって一般的には，そのシステムが処理するデータの中で最も価値の高いものに焦点を絞り，そのデータの安全性が保障されない場合の対策を機器構成の上に求める方法がとられる．機器構成における二重あるいは三重の冗長系がそれにあたる．特に大型のオンライン・システムの機器構成においては，処理するデータの重要性を分類してカテゴリごとに処理系を変えることによって，それぞれのデータの価値に応じた処理の安全性を確保する方法を採用することがある．しかし一般には，ターミナル系や周辺機器の構成に関してこのような方法を適用することはあっても，コンピュータ本体についてはデータ種別に関係なく共通の単一処理系をとることが多い．これは多数の小型より少数の大型の方がシステムの開発，運用の両面においてコスト／パフォーマンスに優れるためであり，多数の小型機による危険分散の方法がプロセス制御システムの一部にみられるにとどまる．大部分の情報処理システムにおいては，むしろコンピュータ本体はそれ自身の信頼性の高いものを採用して処理の集中化をはかり，必要に応じて多重化することが多い．

以下，本稿ではコンピュータの信頼性を追求するための RAS†† 機能と，アプリケーションにおける信頼性確保の手段としてのデュアル・コンピュータ・システムについて述べる．

2. 予防措置と回復措置

データ処理の安全性向上への対策は積極的なものと消極的なものに分けて考えることができる．

積極的な対策とはデータ破壊が生じないような予防措置のことであり，アプリケーション・システム開発の上ではハードウェアの機種とシステム・コントロール・プログラム（以下 SCP と省略する）の選定，および開発プロジェクトの質，特にプログラム設計とテスト・マネジメントに負うところが大きい．

一方，消極的な安全対策とはデータ破壊が発生した場合の回復措置のことで，一般にデータ処理の安全対策とはこの回復措置を指す．例えば機器もしくは処理の二重冗長構成，バックアップ・プログラム，あるいはマニュアル・バックアップなどによる fail soft, fail safe の概念がそれで，アプリケーション・システム設計上できわめて重要な位置を占める．

しかし，データ処理の安全性向上の手段として積極的な対策を忘れてはならない．

最近のコンピュータの開発研究は信頼性への挑戦といっても過言ではあるまい．近年における CPU，メイン・メモリおよびチャネルなどを含むコンピュータ本体の大型化，高速化の傾向は，回路技術における集積化と，システム技術における RAS 機能の二つの

† 日本アイ・ビー・エム（株）研究所テクノロジー担当

†† Reliability, Availability, Serviceability.

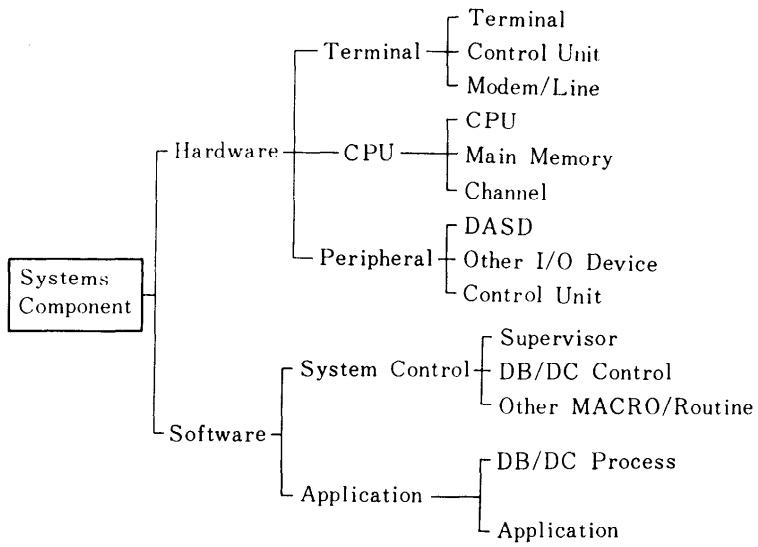

図1 システム構成要素
Fig. 1 Systems Component

テクノロジなくしては実現し得なかったといえる．

集積回路のテクノロジは，photolithography（写真製版）の進歩によって極度の素子縮小化と配線集積化を可能にし，回路の信頼度向上とともに高速化，小容積化，低電力化および低コスト化を達成した．すでに実用段階にある MOS・LSI† は約 3 mm×3 mm の chip 当たり 1 k～2 k bits の集積度をもち，今後さらに集積化は進む傾向にある[1]．

また RAS 機能は，システムの大型化，複雑化にともなって増大するハードウェア内のデータ処理上の危険性をハード・ソフトの両面から減少させるための機能であり，エラーの発見，記録，自動再試行，自動訂正，障害個所の切離しなどによって，従来の CPU ではシステム・ダウンに直結したような障害の多くを事前に押えることを可能にした．

さらに仮想メモリの概念を理論から実用のベースへ引き上げ得たのも，この二つのテクノロジの所産にほかならない．仮想メモリによってアプリケーション・システム開発の環境は著しく向上した[2]．

最近におけるこのような現実は，アプリケーションの立場からみたデータ処理の安全性向上の重要な鍵が，障害の回復措置という消極的な立場から積極的に障害を予防するという姿勢の中にあることを示唆しているように思う．もちろん，完全な信頼度を保障できる機械はあり得ないし，信頼度の数値どおりにシステムの安全運用が保障されるわけではない．しかし，先に上げたテクノロジがどこまで実用化されているかによってシステムの信頼性を予測する目安になることは確実である．

† Metal-Oxide Semiconductor Large Scale Integrated.

3. データ破壊の原因

データ破壊はその原因がシステム外部にある場合と内部にある場合とに分かれる．外部要因には天災地変などの不可抗力，あるいは悪意の第3者によるものなど種々の擾乱があり，システムの環境に応じた対策について議論は多いが，ここでは内部要因に限って検討する．

安全なデータ処理システムを実現するためには，データ破壊がどこで，なぜ起こるかを予想して対策を立てなければならない．外部からの擾乱を考慮しなければ，データ破壊の原因が図1のシステム要素のどこかにあることは疑いない．しかしなぜ発生したかが現象から単純に証明できるとは限らない．むしろ現象から原因が容易に追跡できるようなエラーの大部分はシステム開発時のシミュレーション・テストかシステム・テストの段階に，悪くても運用初期にデバグされている．残るのは，テストを潜り抜けたハード／ソフトのバグとハードウェアの経時変化である．またシステム運用期間には，ハード／ソフトのデバグや機能の追加

などによるシステム要素間の状態変化が連鎖的に他のエラーを生んだり，潜在エラーを顕在化することがしばしば起こる．

オンライン・システムの運用におけるデータ破壊のうちで最も難問とされるのは intermittent error (一過性障害) である．

ハードウェアとソフトウェアはエラーの判定に関して著しい相違がある．ソフトウェアに経時変化を生じる要素がなく正誤の判定は一義的にきまる．それに対してハードウェアの正誤の境界は連続的であり，一般には素子もしくは素子群ごとの動作に関して normal/error range を定めて判定を行なう．したがって経時変化によってある素子がこの range を逸脱することが生じ得るし，デバグや EC (engineering change) が周辺素子の動作に影響を及ぼして range 逸脱を促すこともある．さらに経時変化による接合部，可動部の劣化から起こる接触不良や動作不良がある．このようなところにハードウェアにおける一過性障害の主因が潜んでいる．

一方，ソフトウェアの一過性障害は必ず再現性があるはずであり，丹念なテストによって発見できるに違いない．しかし現象面からはハード／ソフトの分離が難しいことが多い．たとえば，マルチ・プログラミングの下での実行命令の組み合わせとタイミングがハードウェアの一過性障害をたまたま誘発しているような場合は，ほとんど再現の可能性がなく原因不明に終ることが多い．また運用開始後数年を経過して安定しているシステムがソフトウェアの機能追加によって，突然，予想外のエラーでダウンするようなことがある．これらの多くは機能追加によって既存のソフトウェアの動作環境が変化し，潜在エラーが顕在化している．

ソフトウェアによるデータ破壊を最小限にくい止めるには，システム開発のプロジェクト・マネジメント，特にテスト・マネジメントが十分でなければならない．要は，そのための aid が完備している機種と SCP を選択することと，プログラム設計の際にテスト方法を考慮した設計をすることである．他方，ハードウェアについては同機種の他のユーザーでの信頼性の実績値が参考になることは当然であるが，その機器に使用されているテクノロジ・レベルと RAS 機能の充実度によってシステムの安定性を予想し，必要に応じて冗長系機器構成を採用して目的の信頼度に近づけることができる．

4. RAS——Reliability, Availability, Serviceability

ハードウェアの信頼性は，一般に MTBF や MTTR によって定量的に表示され，より安全なデータ処理システムの実現のためにこれらの数値をより良い値にするための努力が払われる．しかしシステムはハードウェア，ソフトウェアおよび人間系の三者によって構成されるから，ハードウェアの信頼性追求だけではシステムの信頼性は向上しない．ソフトウェアの開発，システムの運用，運行，さらに保守といった人間の思考と行動に対する信頼性の追求が一体となって始めて目的が達成される．行動科学，心理学，社会学などがシステムの信頼性の評価と向上に寄与するのはこのような理由による．

しかしながら，安全なデータ処理システムの基本がハードウェアにあることは否定できない事実であり，ソフトウェアも人間系もハードウェアがある程度の信頼性をもって存在することを前提として議論される．

ハードウェアの信頼性向上への対策は，部品，素子そのものの研究とともに，部品／素子障害時のシステム・ダメージの極小化を目指してハードウェア回路および SCP によって自己回復機能をもったコンピュータ・システムを可能にした．

4.1 RAS の概念

RAS は個々につぎのように定義され，機器あるいはシステムの総合的な信頼性能を意味する．

Reliability (信頼度)：一定条件のもとで意図する期間，機器もしくはシステムが規定の機能を故障なく遂行する確率．

Availability (使用可能度)：ある特定の瞬間に機器もしくはシステムがその本来の機能を維持している確率．

Serviceability (保守容易性)：機器もしくはシステムの設計上の構造，構成が保守に関して容易であるかどうかの特性．

信頼度は MTBF を測定の尺度とし，一定期間の故障件数の多少によって評価される．また使用可能度は使用可能時間と power-on 時間の比の大小，および稼動期間中の PM (preventive maintenance), EC の多少が測定の尺度となる．

保守容易性は定量的尺度で測定されるものではなく，システムの設計，構造もしくは構成上の特性であり，いかに保守しやすいか，いかにシステムの運用に

影響を与えず保守できるか，そのためにどのような保守機能を備えているか，また保守要員の数と専門教育がいかに節約できるかといった総合的な保守性を意味する．システムの信頼性はこれらの信頼度，使用可能度，保守容易性の個々によって決定されるものではなく，相互の関連の上に総合的に評価されるべきである．

RAS の基本概念は，ハードウェア，ソフトウェアおよび人間系を含めて統一された信頼性向上の思想のもとにそれぞれの機能を設計しようとする考えに基づいている．

4.2 RAS 機能

RAS 機能は，ハードウェア内で発生したデータ破壊を検知し，可能な限り回復を試みるとともにその状態を保守に備えて記録することによって，故障によるスケジュール外の保守作業の減少と PM 時間の短縮を目的とする．具体的には Recovery 機能と Repair 機能に分かれる．

Recovery 機能は，ハードウェアに redundancy correction と CPU retry の機能を持たせることによりハードウェア自身の障害に対する抵抗力を強める．例えば，CPU 内の主要回路や多数の枝回路を集約する幹回路についてはこれを三重化し，出力端で多数決

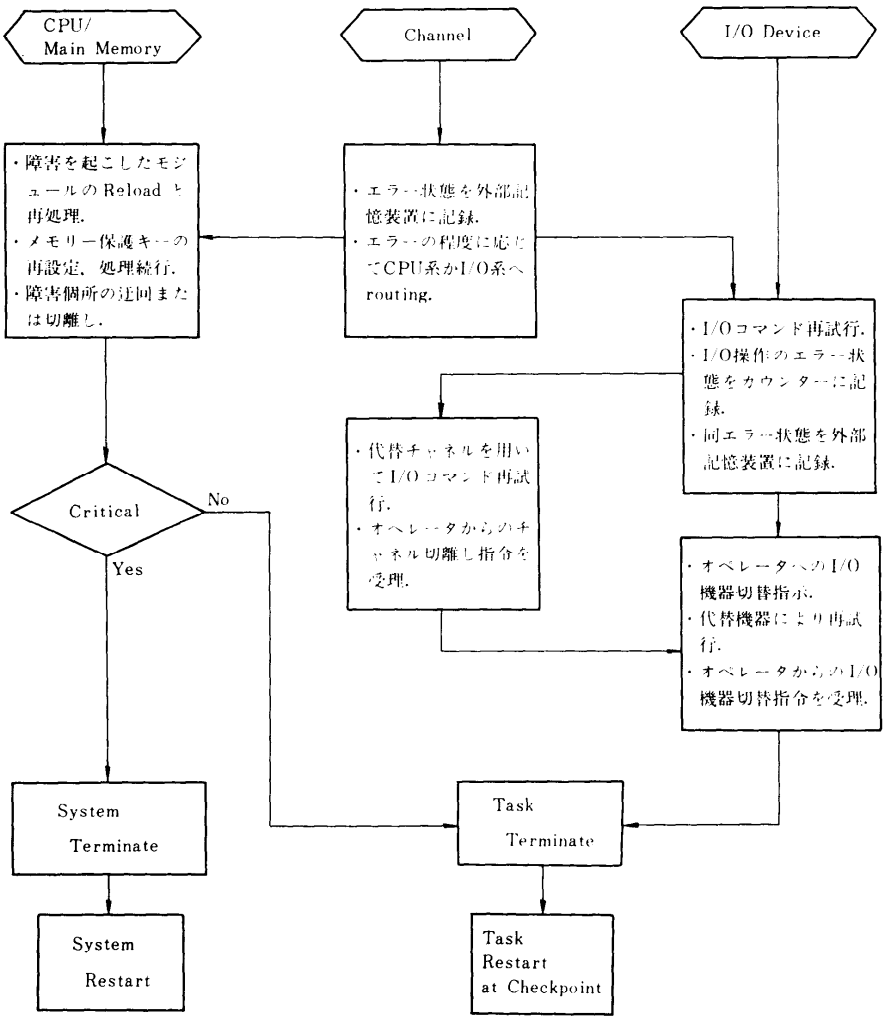

図2 ソフトウェアの Recovery Management

Fig. 2 Recovery Management Concept in SCP

方式により回路の破壊あるいは過渡的な障害を未然に防ぐ．またすべてのデータに特殊な redundancy code を付加することにより，回路内データ転送と入出力オペレーションにおけるデータ内の任意の1ビットのエラーの自動訂正と2ビットのエラーの検出が可能になる．CPU retry は，周期的に CPU 内の主要回路の状態をハードウェア・チェック・ポイントとして保存して redundancy correction によっても回復しない故障に対してチェック・ポイントに戻して再試行する．

このようなハードウェアによる Recovery 機能をさらに補足するために，SCP が図2に示すような Recovery Management を行なう．ソフトウェアの Recovery 機能は，ハードウェアで回復できない故障に対してその故障の程度を判断し，回復のためのあらゆる手をつくし，たとえ回復できないと判断される場合でもシステム全体を停止することなく故障によるシステムのダメージを当該部分だけに抑えるようにする．さらにすべての故障は回復できたか否かにかかわらず詳細に記録される．

ソフトウェアによる Recovery 機能によってもなお回復できない場合には，当該故障に関連するタスクもしくはシステムを停止してリスタートを行なう．この方法には次の四つがある．

1) 異常終了したジョブ・ステップの最初から自動的に SCP が再開始を行なう．
2) プログラム内にチェック・ポイントをとるよう指定しておき，自動的にその点に戻って SCP が再開始を行なう．
3) オペレータが再開始のコマンドを入力し，ジョブ・ステップの最初から再開始する．
4) オペレータが再開始のコマンドを与え，チェック・ポイントに戻って再開始する．

上記1), 2) は SCP が自動的に判断して実行するのに対して，3), 4) はシステム操作者の判断を仰ぐ．SCP とアプリケーション・プログラムは障害の回復に関して interactive に動作する．したがってアプリケーションのプログラミングを容易にするためのマクロ命令を SCP が内臓している必要がある．

Recovery 機能がハードウェアの障害を最小限にくい止めることによってシステムの信頼性向上を目指しているのに対して，Repair 機能はハードウェアの故障の診断とその修理がシステムの使用可能度に与える影響を最小限にすることを目的としている．

診断と修理に要する時間は，まず Recovery 機能が提供するハードウェア・エラーに関する詳細な情報によって，故障が顕在化する前に PM によって抑えることができる．すなわち，外部記憶装置上に記録された error log を利用して，単にどこでエラーが検知されたかを現象面から把えるだけでなく部品，素子に関して多角的に分析し，原因となっている部分を指摘することによって保守要員の判断を助け保守時間の短縮を可能にする．また，オンライン・オペレーション中の SCP のもとで同時にオンライン診断プログラムを実行することによって，I/O 装置，I/O 制御装置の診断を行なう．そして error log を利用して，過去にエラーを記録した機器を自動的に選択して集中的に診断を行なうこともできる．さらに装置の再割当てと切離しを行なってオンライン・オペレーションを中断することなく障害個所の修理作業を許す．

このように Recovery 機能と Repair 機能の充実によってシステムの総合信頼性は一段と向上し，近年における大型で複雑なシステムの実現を可能にしたことは理解するに難くない．

5. デュアルコンピュータ・システム

機器構成に冗長系を採用してデータ処理の安全性を確保する方法としては，基本的に装置の多重化，フォール・バック処置，およびバイパス処置がある．CPU，チャネル，メイン・メモリを含むコンピュータ本体の二重化の方式として J. Martin は，つぎの6種類を定義している[3]．

1) Tandem,
2) Satellite and Parent,
3) Shared File System,
4) Parallel and Load Sharing,
5) Duplex,
6) Twin.

これらの方式はシステムの信頼性，処理能力およびコストに関しておのおの一長一短があり，その選択はアプリケーションの requirement に応じてなされなければならない．そしてこれらの構成を可能にするソフトウェアについて，SCP がどこまでサポートされているかが選定の上で重要なポイントとなる．

これら6方式のうち，システムの信頼度の点で最も優れているのは Twin である．ここでは2台のコンピュータが並行して同一処理を行ない，その結果を比較してもし不一致であればどちらか一方にエラーがあることを知る．

5.1 デュアル・システムの概念

2台のコンピュータ **A, B** に同一処理を並行して実行させ，**A** をオンライン，**B** をスタンドバイとして，**A** に障害が起こったときには **B** が新オンラインとして自動的に処理を継続するようなシステムを Dual System と呼ぶ．

デュアルと混同されやすい類似形として，Twin と Duplex がある．Twin System は **A, B** 両者が対等でともにオンラインであり，それぞれの処理結果を比較して不一致であれば処理方法を変えて再試行する．

Duplex System は，**A** だけがオンライン処理を実行し **B** は単にハードウェアとしてスタンドバイしつつ他の業務に使われる．そして **A** 障害時に **B** は実行中の業務を放棄するかもしくは縮小してオンライン処理を引継ぐ．

Dual System は **A, B** 両者が戦列に参加している点では Duplex よりむしろ Twin に近いが，主従の関係が明確で，**A** の処理に誤りがあると判断されない限り **B** の処理結果は無視され，いわゆるダミーとして動いている．そして **A** 障害の場合は **B** が直ちに処理を引継ぎ，**A** の回復を待って **B** オンライン，**A** スタンドバイとなって主従の関係は逆転する．デュアルにおいては **A** と **B** は同一機種であり，主従の反転によるシステム機能の低下はない．スタンドバイが常時オンラインに追従しているから障害時の切替は秒単位かそれ以下で可能である．

このようにデュアルは Duplex と Twin の中間に位置し，両者に比較して次の特性をもっている．

1) Duplex より障害即応性に優れるがコスト／パフォーマンスは劣る．
2) Twin よりシステムの開発容易性に富むが総合信頼性は低い．

デュアルのもつこのような長短所から，その適用範囲は高信頼性のコンピュータが得やすく，かつ人間系を含んだ fail soft を指向した高信頼性システムが条件となる．マニュアル・バックアップを備えたプロセス制御システムがその好例であろう．

5.2 チェック・ポイントとリスタート

デュアル・システムのチェック・ポイントはスタンドバイ **B** にとられる．**A, B** 間のデータ伝送はチャネル結合あるいはデータ通信による方法が一般的だが，プロセス・コンピュータの場合はディジタル I/O を利用する方法も採用されている[4]．

システムにとって最適のチェック・ポイントをとることはシステム設計時の重要な課題であり，個々のタスクについて次の諸点を明確にしておく必要がある．

- リスタートが必要なタスクか，
- 切替時に発生する処置の遅れの許容量，
- タスクが重複して実行されたときの悪影響とその防止策（例えばファイルの二重書き），
- チェック・ポイントをとるための通常処理の能力低下，
- チェック・ポイント，切替，リスタートのための論理の単純化と標準化．

チェック・ポイントをとるタイミングは，タイマを使って周期的にとる方法とタスク単位にそのタスクの始めか終りにとる方法がある．ここで重要なことはチェック・ポイントのきめの細かさである．リスタートはチェック・ポイントに基いて行なわれるから，チェック・ポイントをあらくとればリスタート時に当該処理の再実行に時間がかかるだけでなく，処理の重複による種々の弊害に対処しなければならない．一方，きめ細かくとり過ぎると通常処理の能力低下をきたす．

この矛盾を避けるためにタスクごと，もしくはメッセージごとにきめの細かさを変えると，チェック・ポイントとリスタートのための論理の単純化と標準化が困難になり，ひいては信頼性を上げる目的が逆にプログラム・エラーのために信頼性の低下をまねくことになりかねない．

5.3 システムの動作

図3にデュアル・システムの概念を示す．以下，同図に従ってデュアルの基本動作を述べる．

まず情報入力系で発生した制御要求データは **A, B** 両者に等しく入力される．データ処理の結果はプロセス制御のための制御項目に分解され，項目ごとに制御順を定めて **A, B** それぞれのデータ・ブロックに収められる．制御項目は基本的に制御対象，制御タイミング，応答許容時間などによって規定される．制御項目に基いて制御出力を出すタイミングは，つぎのいずれかで与えられる．

1) 時刻 trigger,
2) 前制御開始／完了から一定時間後,
3) コンピュータ外からの trigger.

オンライン **A** は，上記の trigger のいずれかによって被制御系に対して制御信号を出し，応答信号を受ける．許容時間内に被制御系からの応答がなければ，当該制御を不良とみなして再制御，迂回制御，あるいは障害状態を人間系に通知してマニュアル・バックア

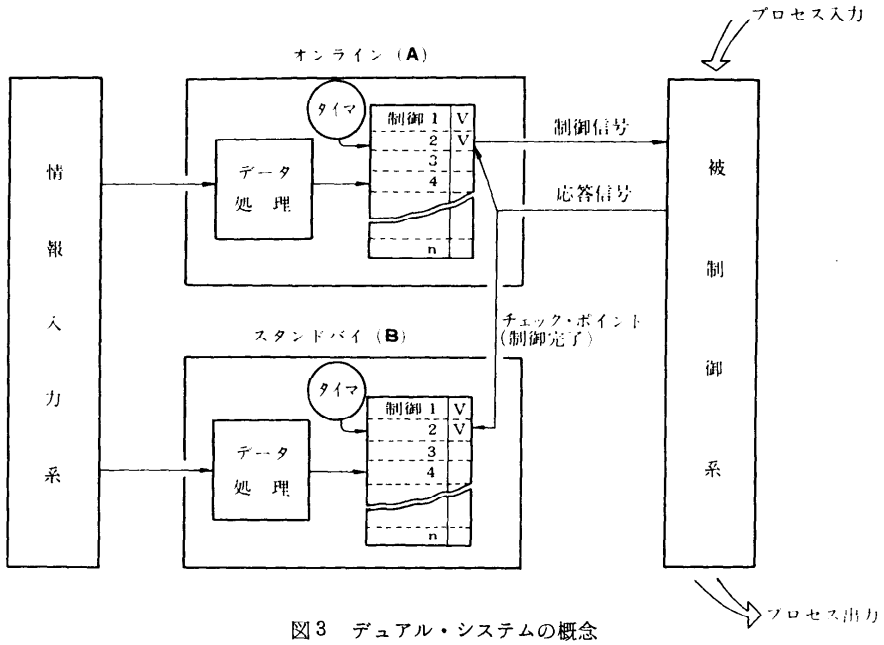

図3　デュアル・システムの概念
Fig. 3　Dual Computer System Concept

ップを要請するなどの被制御系障害処置を行なう．正常に応答信号を受けとった場合，**A** は当該制御完了をデータ・ブロックに記録するとともに**B**に対してチェック・ポイントのデータを転送する．スタンドバイ**B**は，この間**A**から送られてくるはずの制御完了通知を待っている．**A**から正常にデータが送られてくれば，対応する制御項目にチェック・ポイントとして制御完了を記録する．もし**A**からのデータが到着しない場合，**B**の処置にはつぎの方法が考えられる．

1) **A**からのデータ転送許容時間を**B**でチェックしない．（データ到着まで待つ．）
2) データ転送許容時間をチェックし，人間系に通報して判断を待つ．（**B**によるモニター．）
3) データ転送許容時間をチェックし，**A**障害と断定して**A**を停止させ**B**が処置を続行する．

理想的には制御項目の重要度に応じてこれらの3方式を使い分けることが望ましいが，この場合も論理の単純化と標準化を踏まえて設計する必要がある．

A, **B** の切替は上記3）によるもの，2）によって操作者が行なうもの，および**A**自身が内部障害を発見して**B**に通知するか，または人間系に通知して行なうものの各種がある．**B**は制御項目のデータ・ブロックの中で，直前に制御完了した項目のつぎの項目から継続して制御を行なう．

以上がデュアル・システムの動作概要であるが，実際の設計に当っては，前項で述べたチェック・ポイントに関する5項目を十分吟味する必要がある．

6. おわりに

データ処理の安全性の向上に関して，システム開発の立場から2つの方法を述べた．アプリケーション・システムの開発者にとって前者はデータ破壊の予防策であり，後者は回復措置の一例である．

参考文献

1) “超 LSI 時代”がやって来る：日経エレクトロニクス，1973, 6-18, p. 42.
2) 北原重男：システムの有効利用と仮想記憶装置，ビジネス・コミュニケーション，'73, Vol, 10, No. 3, p. 38.
3) James Martin: Design of Real-Time Computer System, Prentice-Hall, Inc.
4) 塚本潤三：放送自動化プロセス・コンピュータの2台並行運転，電気通信学会，信頼性研究会資料，R70-26 (1971-2).

（昭和48年7月12日受付）