(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3727382号
(P3727382)
(45)発行日　平成17年12月14日(2005.12.14)　(24)登録日　平成17年10月7日(2005.10.7)

(51)Int.Cl.[7]	F I
G06F 9/48	G06F 9/46 452F
G05B 19/02	G05B 19/02 T

請求項の数 2　(全 86 頁)

(21)出願番号	特願平7-205600
(22)出願日	平成7年8月11日(1995.8.11)
(65)公開番号	特開平8-235004
(43)公開日	平成8年9月13日(1996.9.13)
審査請求日	平成11年12月15日(1999.12.15)
(31)優先権主張番号	特願平6-323233
(32)優先日	平成6年12月26日(1994.12.26)
(33)優先権主張国	日本国(JP)

(73)特許権者　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目２番３号
(74)代理人　100089118
弁理士　酒井　宏明
(72)発明者　南角　茂樹
東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内

審査官　殿川　雅也

最終頁に続く

(54)【発明の名称】制御ソフトウェア実行システムの制御方法

(57)【特許請求の範囲】

【請求項１】

メモリ内部のリアルタイムオペレーティングシステム内に異なるタスクのスイッチング処理を行なうスケジューラを設け，前記スケジューラによるタスクスイッチの制御に基づき，制御ソフトウェアを実行する複数タスクの実行システムの制御方法において、

前記システムが、タスクの優先度に基づいてタスクの走行順番を決める機能に加えて、

所定の基準周期と指定された各タスクの走行時間割合とに基づいて基準周期内の後半部において前記走行時間割合が指定されたタスクの走行を開始すべき走行時間警告時点を各基準周期内に求め、割り込み処理を行う際に、現在処理中のタスクが走行時間割合が指定されたタスクであるか否かを判定し、現在処理中のタスクが走行時間割合が指定されたタスクでない場合は、前記求めた走行時間警告時点を変更せず、現在処理中のタスクが走行時間割合が指定されたタスクである場合は、当該タスクの走行時間分だけ前記走行時間警告時点を遅延させることにより、基準周期の開始時点から前記走行時間警告時点までの間に前記走行時間割合が指定されたタスクの走行がなかった場合は前記求めた走行時間警告時点から前記走行時間割合が指定されたタスクの走行を実行し、前記走行時間割合が指定されたタスクの走行があった場合は、タスクの走行時間分だけ前記走行時間警告時点を遅延させて、走行時間割合が指定されたタスクについては各基準周期内における走行時間割合を保証する機能

を備えたことを特徴とする制御ソフトウェア実行システムの制御方法。

【請求項２】

タスク単位で予め指定した各所定の連続走行時間の間連続して走らせる事を可能にするために、連続走行を保証したタスクについてはその走行時間を計算し、計算した走行時間が前記連続走行時間を越えた後に当該タスクを終了させる機能をさらに備えることを特徴とする請求項１に記載の制御ソフトウェア実行システムの制御方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

この発明は，制御ソフトウェア実行システムの制御方法に関し，さらに詳細には，数値制御装置（以下，ＮＣ装置）の制御用ソフトウェア（コントロールプログラム）を効率的に実行するＮＣ装置の制御ソフトウェア実行システムの制御方法に関するものである。

【０００２】

【従来の技術】

図７３は，従来におけるＮＣ装置の制御ソフトウェア実行システムの全体構成を示すブロック図であり，図において，５０１はＮＣ装置全体を示す。以下，このＮＣ装置５０１のハードウェア構成例を用いて従来におけるＮＣ装置の一般的概略構成を説明する。

【０００３】

図７３において，１は各部の動作を制御したり，数値制御に必要な演算を実行するメインＣＰＵ，２は装置各部を結ぶシステムバス，３はＮＣ装置５０１の主要機能を実現するための制御ソフトウェアなどを格納するＲＯＭ（不揮発性記憶装置），４は一時記憶やワークエリアなどに用いられるＲＡＭ（揮発性記憶装置）である。

【０００４】

また，５は外部との間においてシリアル通信によりデータのやり取りを行なうＳＩＯインタフェース部，６はＮＣ装置５０１の運転状態を表示したり，ＮＣ装置５０１に与えた指令を確認するための表示装置，７はＮＣ装置５０１に対して指令を与えるためのキーボード（入力装置），８はサーボモータを制御するための指令を演算するサーボ制御部である。このサーボ制御部８にあってはメインＣＰＵ１とは別の専用ＣＰＵを備えてもよい。

【０００５】

また，９はサーボ制御部８から受け取った指令を電力増幅してサーボモータや工作機械主軸（図示せず）を駆動するサーボアンプ，１０は工作機械（図示せず）の加工部を制御するためのサーボモータ，１１は工作機械との間でサーボ制御指令以外のデータをやり取りするためのプログラマブルコントロール部（以下，ＰＣ部という），１２はメインＣＰＵ１に対して入力されるシステムクロック（図示せず）と外部割り込み信号を表している。システムクロックはＮＣ装置５０１全体を制御するために同期をとるためのクロック信号である。また，外部割り込み信号１２は電源異常や非常停止などイベント（緊急の出来事）の発生をメインＣＰＵ１に対して通知するための信号である。

【０００６】

次に，動作について説明する。図７４は，図７３に示したＮＣ装置５０１の制御ソフトウェア実行システムの加工プログラム解析実行手順を示す説明図であり，まず，メインＣＰＵ１は，ＲＯＭ３に書き込まれている制御用のソフトウェアをシステムバス２を介して順次１命令ずつ読み込んで，その内容を実行する。

【０００７】

図７４において，加工プログラム入力処理２１では，ＳＩＯインタフェース部５を介して外部から加工プログラム２０を読み込み，ＲＡＭ４に格納する。その後，加工プログラム２０をブロック（入力する加工プログラムにより決まっている所定の単位）毎に内部で処理し易いような内部データに変換する。

【０００８】

また，補正計算処理２２では，ブロック毎の内部データを処理，演算して移動量を算出する。また，工具径や工具長などの補正を行なう。さらに，内部座標値の更新処理なども行なう。設定表示処理２３では，ＮＣ装置５０１の各種データを表示装置６に表示する。また，キーボード７を用いてオペレータが入力した各種設定データをＲＡＭ４に格納する。

加えて，補間処理２４では，補正計算処理２２の処理結果を用いて微小時間毎の各軸の移動量を算出する。

【０００９】

また，サーボ処理２５では，補間処理２４の処理結果を，さらに，小さな単位時間毎の各軸の移動量に変換する。さらに，サーボモータ１０や工作機械（図示せず）に取り付けている速度検出器および位置検出器（いずれも図示せず）からのフィードバック制御（図示せず）を実行する。プログラマブルコントローラ（以下，ＰＣという）処理２６では，工作機械との間において入出力処理や主軸の制御など，工作機械周辺の制御を実行する。

【００１０】

さて，ＮＣ装置５０１は，図７３を用いて先に説明したようにメインＣＰＵ１に外部割り込み信号１２が入力し，割り込み処理を実行させることで，非常停止などの通常処理の流れ以外の緊急事態に対処することができる。メインＣＰＵ１は，外部割り込み信号１２（図７３参照）が入力されると，予め指定された別処理を実行し，それら別処理が終了した後に通常の命令に復帰する。

【００１１】

図７５は，割り込み処理を示す概念図である。３０～３６は通常の命令，３７～４０は割り込み命令，また，４１は通常の命令への復帰命令である。例えば，メインＣＰＵ１が通常の命令３３を実行中に外部割り込み信号１２（図７３参照）が入力されると，メインＣＰＵ１は，通常の命令３３の処理終了後，割り込みを検出し，予め指定されていた割り込み命令３７の実行を開始する。その後，割り込み命令３７～４０の実行を終了した後，復帰命令４１を実行して，通常の命令３４に復帰し，続いて通常の命令３５，３６を実行する。

【００１２】

なお，割り込み処理の時間が長くなると，割り込み処理の実行中に割り込みが再び入ってくるなど多重割り込みの発生する割合が多くなり，また，割り込み処理中は割り込みを禁止していると割り込みに対して迅速に反応できなくなるなどの問題点があるため，割り込み処理は可能な限り短くすることが望ましい。

【００１３】

ところで，ＮＣ装置５０１の制御ソフトウェアは，以下に示すような特徴を有する。すなわち，

第１に，ＮＣ装置５０１の制御ソフトウェアにより実現すべき機能の多様性のため，ソフトウェアも大量となり，多人数で開発される。

【００１４】

第２に，ＮＣ装置５０１の制御ソフトウェアの機能ごとに，処理に必要な応答時間（ターンアラウンドタイム，デッドライン）が異なる。例えば，サーボ処理２５（図７４参照）は，処理結果の算出が遅れると切削が止まり，被加工物が不良品になってしまうので，実時間処理でなくてはならない。一方，表示装置６への表示処理などは，多少遅れても不都合は生じないので，実時間処理でなくともよい。

【００１５】

第３に，ＮＣ装置５０１の制御ソフトウェアの処理は，バッチ処理型あるいはコンパイラ型（１つの加工プログラムを最初から最後まで全て解析し終わってからサーボの移動データを全て一度に出力する）ではなく，インタプリタ型（加工プログラムを，いくつかに分解して，例えば，ブロックに分解し，そのブロック毎に解析していって，サーボの移動データも少しずつ出力する）である。

【００１６】

第４に，メインＣＰＵ１が実行しなくてはならない，割り込み処理は，多くの種類がある。

【００１７】

これらの特徴により，ＮＣ装置５０１の制御ソフトウェアは，一般にリアルタイムオペレーティングシステム（以下，ＲＴＯＳという）を用い，その制御のもとで実行される機能

ごとのタスクを実行単位とすることが多い。

【００１８】

次に，ＲＴＯＳにより各タスクを制御する方法について説明する。優先順位を一定間隔毎に調べたり，あるタスクを指定時間だけ実行を遅らせたりするために，通常はメインＣＰＵ１（図７３参照）に対してある一定周期でタイマ割り込み（図示せず）を入れる。これをシステムクロックという。

【００１９】

また，ＲＴＯＳは，システムクロック割り込みがある度に各タスクの状態を調べ，実行中のタスクを止めて，別のタスクを実行させたりする。これをスケジューリングあるいはディスパッチという。

【００２０】

さらに，ＲＴＯＳにおいては，各タスクに優先順位（実行の優先度）付けを実行する。この優先順位の意味は，より低い優先順位のタスクを実行中に，より高い優先順位のタスクを実行する準備ができたとき，その優先順位の低いタスクの実行を中断させて優先順位の高いタスクを実行させることをいう。これをタスクのプリエムプト（横取り）という。

【００２１】

図７６は，各タスクにおける動作の時間的関係を示すタイミングムチャートであり，図において，特に，補正計算処理タスクなどは，計算対象タスクの複雑さによって処理時間が大きく変化する。通常は，最も処理時間がかかる場合を想定してシステムクロックの周期を決定する。その結果，それ以外の処理を実行する場合には，図７６に示したＴ１のようなメインＣＰＵ１が有効な処理を何ら実行していないアイドルタイムが発生する。なお，図７６において，他の処理がないときは後述のようにＲＴＯＳ中のループにおいて処理が待機するものである。

【００２２】

通常，ＲＴＯＳは，指定した周期でタスクを走行させたり，タスクの走行時間を測定する機能を有するが，それらはすべてシステムクロックΔＴの整数倍の単位でしか制御測定することができない。

【００２３】

また，図７６に示した３つのタスクの内，サーボ処理タスク，表示設定処理タスクの処理時間（演算時間）は通常一定である。しかし，補正計算処理タスクは加工プログラムの内容によって処理時間が大きく変化する。一方，サーボ処理タスクは，サーボアンプ９に対するデータ通信等を実行しているため，必ずシステムクロック１回につき１回走行する。このように制御しなければ，アンプユニットの方にサーボモータ１０を動作させるデータを送信することができず，サーボモータ１０が停止してしまうからである。

【００２４】

そのため，例えば，補正計算処理タスクを加工プログラム１ブロック毎に終了させて，１回に走行する時間が余り長くならないように制御しているものである。しかし，逆に，補正計算処理タスクの処理時間が短くて，３つのタスクの処理時間の合計がシステムクロックの１／Ｎ（Ｎは整数）のときであってもシステムクロック１回につき各タスクが１回づつしか走行できない。この制限によってＮＣ装置の切削送り速度の上限が決定されている。

【００２５】

したがって，例えば，システムクロック１回に各タスクが２回づつ走行することができるとすると，切削送り速度を２倍にすることができ，加工時間は，図７７に示すように約１／２となる。

【００２６】

以上のＲＴＯＳのもとで，制御ソフトウェアを実行するＮＣ装置５０１は，何種類かのモードを持っている。例えば，実際に自動運転により被加工物を切削している状態や，手動モードで切削している状態，さらには実加工は行なわずグラフィックによりツールパスを表示させて，加工プログラムの正当性のチェックをしている場合などである。

【００２７】
また，加工プログラムの入力手段としては，ＮＣ装置５０１の標準の加工プログラム形式である，ＥＩＡフォーマットのほかにも，例えば，自動プログラミングというものがある。これは，例えば，使用工具と，素材形状，そして最終の加工形状などを入力する，その結果，図７８（ａ），（ｂ）に示すような，その最終形状を加工するように，工具の速度や位置，動作などを自動的に制御するものである。
【００２８】
上記のような，ＥＩＡフォーマットや，自動プログラミングなどの加工プログラムを解析するタスクは，図７９に示すような処理を実行する。すなわち，ＺＺＺ１は，加工プログラムを１ブロック読み出して，メモリ上のワークエリアへ格納する処理である。ＺＺＺ２は，ＺＺＺ１で読み出したデータを解析し，補正計算処理タスクへ渡すデータを作成する処理である。
【００２９】
ＺＺＺ３は，タスク終了のＳＶＣを発行する処理である。このＳＶＣにより，このタスクは他のタスクにメインＣＰＵ１の使用権を譲り渡す。その後，例えば，補正計算処理タスクが，次のデータを要求し，この加工プログラムが解析タスクを起動するまで，システムは待機状態となる。
【００３０】
【発明が解決しようとする課題】
しかしながら，上記に示した構成のＮＣ装置の制御ソフトウェア実行システムにあっては，その制御ソフトウェアを実行するとき，以下のような問題点が発生する。
【００３１】
第１に，ＮＣ装置としての機能を実現する各タスクは，優先順位によってしかその実行の順番が決められない，そのため全てのタスクを周期的に走らせたり，タスク毎に実行時間を決めたりすることができない。
【００３２】
第２に，タスクの制御を時間により実行するような場合であっても，その単位はシステムクロックの単位（システムクロックの整数倍）に限定される。
【００３３】
第３に，タスクの優先順位の決め方が難しく，特に，３個以上のタスクが関係すると，いわゆるプライオリティインバージョン（優先順位の低いタスクが優先順位の高いタスクを止めてしまうような優先順位の逆転現象）が発生する可能性があるため，優先順位だけによるに実行方式には限界がある。
【００３４】
第４に，システムの生成時にタスク間の優先順位を定めてしまうため，処理時間が足りないタスクや，逆に処理時間が余るタスクが存在する。
【００３５】
■第５に，周期を指定したい処理は割り込み処理としなくてはならないため，その処理時間を短くしなくてはならないなどの制限がある。
【００３６】
第６に，ＮＣ装置の運転状態や，モードによって各タスクにかかる負荷は一定ではないにもかかわらず，各タスクの走行状態をシステム生成時に決定して，そのまま変更できない。
【００３７】
第７に，あるタスクが正常ではない（異常状態に陥った）か否かを判定する手段がない。
【００３８】
第８に，加工プログラムのブロックの大きさにかかわらず，加工プログラム解析タスクが，１ブロック毎に終了するようになっているので，１ブロックの大きさによっては，処理時間が足りなかったり，逆に余ったりする場合がある。
【００３９】

この発明は，上記のような問題点を解決するためになされたもので，ＮＣ装置における各タスクの実行の指定方式に柔軟性を持たせることによって各タスクに適切な処理時間を与え，そのことにより，効率のよい制御ソフトウェア実行システムの制御方法を得ることを目的とする。

【００４０】

【課題を解決するための手段】

請求項１の発明（実施例２に対応）では、メモリ内部のリアルタイムオペレーティングシステム内に異なるタスクのスイッチング処理を行なうスケジューラを設け，前記スケジューラによるタスクスイッチの制御に基づき，制御ソフトウェアを実行する複数タスクの実行システムの制御方法において、前記システムが、タスクの優先度に基づいてタスクの走行順番を決める機能に加えて、所定の基準周期と指定された各タスクの走行時間割合とに基づいて基準周期内の後半部において前記走行時間割合が指定されたタスクの走行を開始すべき走行時間警告時点を各基準周期内に求め、割り込み処理を行う際に、現在処理中のタスクが走行時間割合が指定されたタスクであるか否かを判定し、現在処理中のタスクが走行時間割合が指定されたタスクでない場合は、前記求めた走行時間警告時点を変更せず、現在処理中のタスクが走行時間割合が指定されたタスクである場合は、当該タスクの走行時間分だけ前記走行時間警告時点を遅延させることにより、基準周期の開始時点から前記走行時間警告時点までの間に前記走行時間割合が指定されたタスクの走行がなかった場合は前記求めた走行時間警告時点から前記走行時間割合が指定されたタスクの走行を実行し、前記走行時間割合が指定されたタスクの走行があった場合は、タスクの走行時間分だけ前記走行時間警告時点を遅延させて、走行時間割合が指定されたタスクについては各基準周期内における走行時間割合を保証する機能を備えたことを特徴とする。

【００４３】

したがって，全てのタスクの走行時間の割合を指定でき，例えば，ＮＣ装置機能を実現するタスクの走行時間による制御を実行することができる。

【００４４】

請求項２の発明（実施例３に対応）では、請求項１の発明の機能に加えて、タスク単位で予め指定した各所定の連続走行時間の間連続して走らせる事を可能にするために、連続走行を保証したタスクについてはその走行時間を計算し、計算した走行時間が前記連続走行時間を越えた後に当該タスクを終了させる機能をさらに備えるものである。すなわち，個々のタスクを指定した時間連続して走らせる機能を有するものである。

【００４５】

したがって，全てのタスクの連続走行時間を指定でき，例えば，ＮＣ装置機能を実現するタスクに連続した走行時間を保証することができる。

【００９８】

【発明の実施の形態】

以下，この発明に係る制御ソフトウェア実行システムを図について説明する。

【００９９】

〔実施例１〕

まず，実施例１の構成について説明する。図１は，実施例１に係るＮＣ装置の制御ソフトウェア実行システムＡ１の全体構成を示すブロック図であり，この制御ソフトウェア実行システムＡ１は，以下の要素より構成されている。１は各部の動作を制御したり，数値制御に必要な演算を実行するメインＣＰＵ，２はシステムの各部を結ぶシステムバス，３はＮＣ装置の主要機能を実現する制御ソフトウェアなどを格納するＲＯＭ（不揮発性記憶手段），４はデータの一時記憶やワークエリアなどに用いるＲＡＭ（揮発性記憶手段），５は外部との間でシリアル通信によりデータのやり取りを行なうＳＩＯインタフェース部である。

【０１００】

また，図１において，６はＮＣ装置の運転状態を表示したり，ＮＣ装置に与えた指令を確認するためにデータ表示を行う表示装置，７はＮＣ装置に対して指令を与えるための入力

手段としてのキーボード，８はサーボモータを制御するための指令を演算するサーボ制御部であり，該サーボ制御部８はメインＣＰＵ１とは別の専用ＣＰＵを備えてもよい。

【０１０１】

さらに，図１において，９はサーボ制御部８から受け取った指令を電力増幅してサーボモータや工作機械主軸（図示せず）を駆動するサーボアンプ，１０は工作機械（図示せず）の加工部を制御するためのサーボモータ，１１は工作機械との間でサーボ制御指令以外のデータをやり取りするためのプログラマブルコントロール部（以下，ＰＣ部という），１２はメインＣＰＵ１に入力される，システムクロック（図示せず）と外部割り込み信号を表している。ここで，システムクロックはＮＣ装置全体を制御するために同期をとるためのクロック信号であり，また，外部割り込み信号は電源異常や非常停止などイベント（緊急の出来事）の発生をメインＣＰＵ１に通知するための信号である。

【０１０２】

また，図１において，２００は内部割り込み信号であり，この信号はメインＣＰＵ１により，その発生やタイミングなどが制御される。２０１はリアルタイムクロックであり，メインＣＰＵ１により制御され，内部割り込み信号２００を制御したり，時間を読み取ったりする。さらに，２０２はリアルタイムクロック２０１の内部にあるカウンタレジスタであり，これは外部からクロック信号が入力されるたびにその値を１ずつ減らし，０になると再び初期値が設定されるか，あるいはメインＣＰＵ１から再び値を設定されるまで動作を止める。この動作の選択や初期値の設定，変更はメインＣＰＵ１により任意に実行される。なお，２０３はリアルタイムクロック２０１に対してクロック信号を出力する水晶発振器である。

【０１０３】

次に，上記の構成を有するＮＣ装置の制御ソフトウェア実行システムＡ１の動作について説明する。通常，システムクロックとしては６０Ｈｚから１００Ｈｚ（１秒間に６０回から１００回）を用いる。システムクロックをこの数値以上に速くすると，その割り込み処理を実行する回数が多くなり過ぎ，ＮＣ装置の機能を実行するタスクの走行可能な時間が少なくなってしまうからである。しかし，タスクの細かい制御に対しては，これでは最小単位が大き過ぎるため，割り込みは使用せずに，例えば，汎用のリアルタイムクロック２０１を使用する。水晶発振器２０３によって，リアルタイムクロック２０１に，例えば，１０ＭＨｚのクロック信号を入力すると，リアルタイムクロック２０１はその内部カウンタレジスタ２０２を１ずつ減らしていく。１０ＭＨｚのクロック信号の入力の場合は０．１μｓ（０．０００００１秒）単位で測定することができる。

【０１０４】

リアルタイムクロック２０１は，また，プログラマブルインターバルタイマとしての役割も果たすことができる。これは，指定した初期値が０になったときにＣＰＵ１に対して割り込みを発生させる機能である。通常リアルタイムクロック２０１は，上記のようにどちらの使い方もできるタイマを複数チャンネル持っている。

【０１０５】

各タスクは，予め，あるいは，そのタスクが開始させられるときにＳＶＣ（スーパーバイザーコール，または，サービスコール）によってその実行周期を登録する。ここで，ＳＶＣとは，タスクがＲＴＯＳに対して何らかのサービスを要求するときの総称である。何らかのサービスとは，例えば，何らかの条件が成立するまで自分自身の実行を停止させたり，他のタスクと通信を行なったり，他のタスクを起動させたり（実行を開始させたり），停止させたりすることである。

【０１０６】

このデータは，ＲＴＯＳが，例えば，図２に示すような“タスク走行周期テーブル”５０に登録する。図２は，タスク毎に“走行周期”と“繰り返し回数”を有するデータ構造である。ここで，走行周期とは，タスクの次に走るまでのクロック数の値であり，また，繰り返し回数とは，何回そのタスクを起動するかを示す回数である。例えば，この値がマイナスの値ならば∞を示すものとし，その場合にあっては，タスクは繰り返し起動するもの

とする。このタスク走行周期テーブル５０を用いて，ＲＴＯＳは，タスクに起動がかかるたびに図３に示すような“タスク走行待ちリスト”５１を作成する。

【０１０７】

図３において，タスク走行待ちリスト５１の中の各走行待ちブロック５２は，ぞれぞれ，次の走行待ちブロックの位置を示す“リンク”，“タスク名”，今現在からどれだけの時間が経過した後に走らなくてはならないかの“走行予定クロック数”，同時に走らせなくてはならないタスク名を示す“同時タスク名”，それにＴＣＢの位置を示す“ＴＣＢポインタ”等の要素から成るブロックである。

【０１０８】

以下に，図４を用いて具体的な動作について説明する。説明の都合上，水晶発振器２０３に対して入力するクロック信号は，１０ＭＨｚとして内部カウンタレジスタ２０２は，０．１μ秒毎に１減るものとする。すなわち，計測の最小単位は０．１μ秒とする。また，説明中，リストを操作している最中など必要な処理中は，割り込みは禁止しているものとする。さらに，走行待ちブロック５２を作成する場合には，タスク名としては当該タスクを示す名前を入れておくものとする。

【０１０９】

図４は，ＲＴＯＳが，タスク走行待ちリスト５１を作成するときの手順を示したものである。図において，まず，Ｂ１では，要求のあった走行周期を内部クロック数に変換する。例えば，要求が１０ｍ秒であれば，１００，０００（１００，０００×０．１μ秒＝１０ｍ秒）である。これをＲＴ（ｒｅｑｕｅｓｔ　ｔｉｍｅ）とする。次に，Ｂ２では，現在の走行待ちリストに走行待ちブロックがあるか否かを判定し，走行待ちブロックがないと判定した場合には，Ｂ３に処理が移行し，反対に，走行待ちブロックがあると判定した場合には，Ｂ６に処理が移行する。

【０１１０】

Ｂ３は，走行待ちリスト５１に走行待ちブロック５２がなかった場合における処理であり，走行待ちブロック５２を作成し，Ｂ４では，走行待ちリスト５１の先頭に走行待ちブロック５２を置く。その後，Ｂ５では，走行待ちリスト５１の先頭ブロックの走行予定クロック数を，リアルタイムクロック２０１のカウンタレジスタ２０２に設定する。

【０１１１】

次に，Ｂ６は，走行待ちリスト５１に走行待ちブロック５２があると判断した場合における処理であり，最終タスク走行時間ＬＴを０にして，タスクインデックスＴＩを１にする。ここで，ＬＴとは当該タスクを走らせるのは現在からどれだけのクロック数かを計算するための変数であり，ＴＩは走行待ちリスト中で当該タスクを示すインデックスである。その後，Ｂ７では，現在のリアルタイムクロック２０１のカウンタレジスタ２０２の値を読み込んで，先頭の走行待ちブロック５２の走行予定クロック数に入れる。次に，Ｂ８において，走行待ちリスト５１内の，全ての走行待ちブロック５２を調べ終ったか（たどったか）否かを判定する。

【０１１２】

その結果，Ｂ８において，全ての走行待ちブロック５２を調べ終っていないと判定した場合には，Ｂ９に処理が移行する。Ｂ９では，走行待ちブロック５１が，走行待ちリスト５１中にあったと判断した場合の処理であり，走行待ちリスト５１の中から，ＴＩの示す走行待ちブロック５２の，走行予定クロック数を取り出してＬＴに加える。

【０１１３】

その後，Ｂ１０において，ＲＴとＬＴとを比較する。Ｂ１１は，ＲＴとＬＴが等しい場合（ＲＴ＝ＬＴ）の処理であり，走行待ちブロック５２を作成する。このとき，走行予定クロック数は０にしておく。Ｂ１２は，Ｂ１１において作成した走行待ちブロック５２を，走行待ちリスト５１中のＴＩの示すブロックの同時タスク名へつなぐ。ここで，つなぐとは，例えば，Ｂ１１において作成したブロックのアドレスをＴＩが示す走行待ちブロック５２の同時タスク名の場所に置くなど，要するにＢ１１において作成したブロックを，直ちに見つけられるような配置に設定することである。

【０１１４】
Ｂ１３は，ＲＴがＬＴより大きな場合（ＲＴ＞ＬＴ）の処理であり，ＴＩの値を，１プラスして，Ｂ８へ戻り同じ処理を繰り返す。
【０１１５】
Ｂ１４は，ＲＴがＬＴより小さな場合（ＲＴ＜ＬＴ）の処理であり，走行待ちブロック５２を作成する。このとき，走行予定クロック数は，ＲＴ－（ＴＩが示すより１つ前の走行待ちブロック５２の走行予定クロック数）の値にする。その後，Ｂ１５では，ＴＩが示すブロックの走行予定クロック数をＬＴ－ＲＴとし，Ｂ１６では，ＴＩが示す走行待ちブロック５２とその前のブロックとの間にＢ１４において作成した走行待ちブロック５２を挿入する。その後，Ｂ１７では，Ｂ１６で挿入したブロックが，走行待ちリスト５１の先頭か否かを判定し，Ｂ１８において，先頭であると判定した場合には，走行待ちリスト５１の先頭ブロックの走行予定クロック数を，リアルタイムクロック２０１のカウンタレジスタ２０２に設定する。
【０１１６】
Ｂ１９では，Ｂ８において，全ての走行待ちブロックを調べ終ったと判定した場合に，走行予定クロック数がＲＴ－ＬＴの走行待ちブロック５２を作成し，その後，Ｂ２０では，Ｂ１９において作成した走行待ちブロック５２を走行待ちリスト５１の最後に挿入する。
【０１１７】
図５は，走行待ちリスト５１の例である。この例では現在から３０クロック後にタスクＡを起動して，さらにそれから２０クロック後（すなわち，現在から５０クロック後）にタスクＢを起動することを示している。このとき，タスクＣを３５クロック後に起動する要求がＲＴＯＳに対して発せられたとすると，図４に示したフローチャートの手順にしたがって走行待ちリストは図６に示すようになる。
【０１１８】
図４に示したＢ５や，Ｂ１８において走行待ちリスト５１の先頭ブロックの走行予定クロック数を，リアルタイムクロック２０１のカウンタレジスタ２０２に指定し，その指定したクロック数が経過した後，“タスク起動割り込み”が発生する。このタスク起動割り込みの処理について図７を用いて説明する。
【０１１９】
図７において，Ｃ１では，走行待ちリスト５１の先頭の走行待ちブロック５２を走行待ちリスト５１から切り離す処理を実行し，その後，Ｃ２では，走行待ちリスト５１が空になったか否かを判定する。また，Ｃ３は，走行待ちリスト５１が空になったと判定した場合の処理であり，リアルタイムクロック２０１のカウンタレジスタ２０２に０を入れるなどの手段により，以後のタスク起動割り込みが発生しないように，タスク起動割り込みを禁止する。
【０１２０】
Ｃ４は，Ｃ２において，走行待ちリスト５１に走行待ちブロック５２が残っていると判定した場合の処理であり，先頭の走行待ちブロック５２の走行予定クロック数を，リアルタイムクロック２０１のカウンタレジスタ２０２に設定する。その後，Ｃ５では，Ｃ１において切り離した走行待ちブロック５２から起動するタスクを取り出して，そのタスクのＴＣＢをＴＣＢ待ち行列の先頭に配置する。
【０１２１】
その後，Ｃ６では，今走らせたタスクの，タスク走行周期テーブル５０（図２参照）の周期回数を判定する。Ｃ７は，走行回数が∞（負）の場合の処理であり，このタスクの走行待ちブロック５２の，走行予定クロック数を，タスク走行周期テーブル５０の走行周期に変え，走行待ちブロックを走行待ちリストに追加する。
【０１２２】
Ｃ８は，走行回数が正の整数だった場合の処理であり，走行予定回数を１減らす。その後，Ｃ９では，当該タスクに対して図４に示した処理により，このタスクの走行待ちブロック５２を作成して走行待ちリストに追加する。

【０１２３】
Ｃ１０は，走行回数が０の場合の処理であり，この場合，何の処理も実行せずに処理を終了する。Ｃ７とＣ９の処理の後，Ｃ１１では，ＲＴＯＳのスケジューラへ飛ぶ。スケジューラの処理は，従来におけるスケジューラ処理と同様であるので，その説明を省略する。
【０１２４】
次に，タスク起動割り込み以外の一般の割り込み処理について，図８を用いて説明する。Ｄ１は，ある割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する。ただし，この処理は，通常メインＣＰＵ１により割り込みが発生したときに自動的に実行される。Ｄ２は，未使用のＩＣＢをＩＣＢプール（未使用ＩＣＢを保持しておく場所）から１つ獲得する。
【０１２５】
その後，Ｄ３では，ＩＣＢに，自分自身の割り込み処理の，実行環境を格納保存する。この中には，次に実行する命令のアドレスとしてＤ１２の処理における先頭アドレスをＩＣＢに格納することも含まれる。また，Ｄ４では，今作成したＩＣＢを，スケジューラが実行するＩＣＢ待ち行列に追加する。その後，Ｄ５は，Ｄ１の割り込みの禁止を解除する。これ以降は，再び割り込みが発生する可能性がある。以上の処理は割り込み禁止時間をできる限り短くするためのものである。
【０１２６】
次に，Ｄ６は，割り込まれた処理が，割り込み処理あるいはＲＴＯＳか，それともタスクの処理中かを判定する。その結果，Ｄ７は，割り込まれた処理が割り込み処理またはＲＴＯＳを実行中であると判断した場合であって，このときは割り込まれた処理に直接戻る。反対に，Ｄ８は，割り込まれた処理が，タスクの処理を実行中であると判断した場合であって，このときは，まず割り込まれたタスクのＴＣＢを探し出す。その後，Ｄ９では，Ｄ８において見つけたＴＣＢへ割り込まれたタスクの実行環境を格納保存する。
【０１２７】
さらに，Ｄ１０では，作成したＴＣＢを，スケジューラが実行するＴＣＢ待ち行列に優先順位の順番に追加する。また，Ｄ１１では，スケジューラへ飛び，スケジューラからＤ１２に処理が戻ってくるが，スケジューラへ飛んだ後すぐにこの処理が実行されるとは限らず，他のＩＣＢが処理された後に戻る場合もある。ここからが割込毎に固有の本来の割り込み処理であり，割り込み要因によって処理が異なる。さらに，Ｄ１３は，割り込み処理の終了であり，通常のサブルーチンへのリターン命令が一般的であり，このリターン命令によって，スケジューラのＩＣＢ処理に戻る。
【０１２８】
上記のとおり，実施例１に係るＮＣ装置の制御ソフトウェア実行システムＡ１は，通常のプライオリティベースのスケジューリング方式しか持たないＲＴＯＳにシステムクロックよりも細かい精度でタスク起動の制御が行なえる機能を追加し，ＮＣ装置の制御ソフトウェアを実行するものである。
【０１２９】
実施例１によれば，ＮＣ装置の制御ソフトウェア実行システムＡ１は，ＮＣ装置機能を実現するタスクの時間による制御を小さな単位で行なえるようになり，さらにタスクを起動する周期を簡単に指定できる。そのため，タスクに割り当てた時間の無駄を少なくでき，さらに，タスクをその走行時間において制御できるので，システムの構築が容易になると共にＮＣ装置の処理速度が向上する。
【０１３０】
〔実施例２〕
次に，実施例２の構成について説明する。実施例２に係るＮＣ装置の制御ソフトウェア実行システムＡ２の全体構成を示す図９の内容は，実施例１において示した図１とほぼ同様である。ただし，実施例１にあっては，リアルタイムクロック２０１の割り込みポートを１チャンネルしか使用しなかったが，本実施例においては３チャンネル用いている点が異なる。

【０１３１】
以下，構成を詳細に説明する。この制御ソフトウェア実行システムＡ２の１～１１までは，実施例１の図１に示した構成と同様であるため，その説明を省略する。図９において，２００はメインＣＰＵ１に入力される，内部割り込み信号である。この信号は，メインＣＰＵ１により，その発生およびタイミングを制御することができる。２０１はリアルタイムクロックであり，メインＣＰＵ１により制御され，内部割り込み信号２００を制御し，あるいは時間を読み取ったりするものである。
【０１３２】
また，２０２はリアルタイムクロック２０１の内部にあるカウンタレジスタである。これは外部からクロック信号が入力されるたびに，その値を１ずつ減らし，０になると再び初期値が設定される。なお，初期値はメインＣＰＵ１から自由に設定できる。なお，２０３はリアルタイムクロック２０１にクロック信号を入力する水晶発振器である。
【０１３３】
また，２０４，２０５はリアルタイムクロック２０１の内部にある第２，第３のカウンタレジスタである。これは外部からクロック信号が入力されるたびにその値を１ずつ減らしていき，０になると再び初期値が設定されるか，あるいはメインＣＰＵ１から再び値を設定されるまで動作を止める。この動作の選択や初期値の設定，変更はメインＣＰＵ１により任意に実行される。また，２０２，２０４，２０５はそれぞれ独立に設定，解除ができる。本実施例ではカウンタレジスタ２０２，２０４，２０５をタスク走行時間計測用，基準時間割り込み用，走行時間警告用にそれぞれ用いる。
【０１３４】
次に，ＮＣ装置の制御ソフトウェア実行システムＡ２の動作について説明する。図１０は，本実施例において必要なデータ構造を示したもので，２１０は“タスク開始チックメモ”であり，ＲＴＯＳが，タスクを起動する直前に図９に示したリアルタイムクロック２０１の内部カウンタレジスタ２０２の値を記録しておき，その後，タスクが終了してＲＴＯＳに戻ってきたときの内部カウンタレジスタ２０２との差を求めることによりタスクが走った時間を求めるものである。
【０１３５】
また，２１１は“基準周期”を格納しておく領域である。システムに対して予め１つの基準周期を登録しておく。この周期毎に各タスクはどれくらい走行するかを指定する。この指定は時間でもチック数でも構わないが，ここでは説明の都合上，チック数に変換しておくものとする。
【０１３６】
さらに，２１２は“タスク走行状態メモ”である。各タスクは予め，あるいは，そのタスクが開始させられるときにＳＶＣによって，その実行の割合を登録する。例えば，タスク毎の指定走行周期，すなわち，タスクＡが１０％，タスクＢが５％，タスクＣが２０％，基準周期が２０ｍ秒のとき，２０ｍ秒毎にタスクＡが２ｍ秒，タスクＢが１ｍ秒，タスクｃが４ｍ秒走ればよい。
【０１３７】
ここで，実施例１と同様，図１において，水晶発振器２０３へ入力するクロック信号は１０ＭＨｚとして内部カウンタレジスタ２０２は，０．１μ秒毎に１減少する，すなわち，計測の最小単位は０．１μ秒とするとＡ，Ｂ，Ｃ各タスクはクロック数２００，０００チックの間にそれぞれ２０，０００，１０，０００，４０，０００チック分走ればよいことになる。このチック数をタスク走行状態メモ２１２の，“走行チック数”の欄に格納する。ここで，走行する割合を指定しないタスクに対しては，この欄を０に設定しておく。
【０１３８】
また，“走行残りチック数”の欄は，１回の基準周期が終了する毎に，走行チック数の欄をコピーし，タスクが走ったチック数だけ減算していく。１回の基準周期が終る毎に，すべてのタスクの，この欄の内容が０になっていればよい。当然走行チック数の合計は基準周期よりも少なくなければならない。そのエラーチェックは，例えば，タスク状態メモに

新たに登録する毎に行なえばよい。ただし，以下の説明においては，このエラーチェックについては省略する。

【０１３９】

次に，“繰り返し回数”の欄に設定するのは，何回そのタスクを起動するかを示す回数である。例えば，この値がマイナスの値ならば∞を示すとする。その場合には，当該タスクは繰り返し起動するものとする。

【０１４０】

また，“走行禁止フラグ”の欄は，スケジューラが用いる。このフラグがオンになっているタスクはスケジューリングの対象から外す。すなわち，このフラグがオンのタスクはプライオリティが高くても走ることができない。

【０１４１】

次に，割り込み処理の動作について説明する。制御ソフトウェア実行システムＡ２は，稼働し始めたときなど走行割合を保証するタスクが動作し始める前に，予めリアルタイムクロック（プログラマブルインターバルタイマ）２０１（図９参照）の内部カウンタレジスタ２０４に対して，基準周期２１１（図１０参照）の値を設定しておく。

【０１４２】

図１１は，基準周期割り込み処理の手順について説明したフローチャートであり，Ｅ１では，ある割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する。通常，この処理はメインＣＰＵ１によって自動的に実行される。Ｅ２では，タスク走行状態メモ２１２（図１０参照）の走行残りチック数の合計を求める。Ｅ３では，その合計が０以上か否かを判定し，０以上ではないと判定した場合には，Ｅ４においてエラー処理を行なうが，通常は発生しないのでここでは，その説明を省略する。

【０１４３】

反対に，Ｅ３において，合計が０以上であると判定した場合には，Ｅ５においてタスク走行状態メモ２１２（図１０参照）の走行チック数の欄の値を走行残りチック数の欄に，タスク毎にコピーする。その後，Ｅ６では，再びリアルタイムクロック（プログラマブルインターバルタイマ）２０１（図９参照）の内部カウンタレジスタ２０４に対して，基準周期２１１（図１０参照）の値を設定する。

【０１４４】

その後，Ｅ７では，タスク走行状態メモ２１２（図１０参照）のすべてのタスクの走行禁止フラグを，オフにする処理を行う。この処理により，再びすべてのタスクがスケジューリング対象になる。さらに，Ｅ８では，基準周期２１１（図１０参照）からタスク走行状態メモ２１２の走行残りチック数の合計を減算した値をリアルタイムクロック２０１（図９参照）の走行時間警告用の内部カウンタレジスタ２０５に設定する。この結果，周期を保証したタスクだけを走らせる時間しかなくなったときに走行時間警告の割り込みが発生する。

【０１４５】

その後，Ｅ９では，割り込まれた状態がタスク処理中か否かを判定し，タスク処理中であると判定した場合には，Ｅ１０において，タスクの実行環境をＴＣＢへ保存し，Ｅ１１では，Ｅ１０において保存したＴＣＢを，ＴＣＢ待ち行列につなぎ，さらに，Ｅ１２では，割り込みを許可し，Ｅ１３では，スケジューラへ飛ぶ。

【０１４６】

反対に，Ｅ９において，タスク処理中ではないと判定した場合には，Ｅ１４において，割り込まれた処理に戻る。このとき，割り込みの禁止／許可は割り込まれたときの状態に戻す。

【０１４７】

以上の説明においては，以後の割り込み説明において，従来例のように，割り込み処理自身のアドレスをリストに登録しておいて，多重割り込み（割り込み処理中の他の割り込み）を許可する処理を実行してはいないが，その場合にあっても，従来例のように，多重割り込みを許す処理を実行してもよい。ただし，その分オーバヘッド（必要な処理が行なわ

れるまでに要する無駄時間）が発生するので，タスク走行割合などを余分に設定しておく必要がある。

【０１４８】

図１２は，走行時間警告割り込み処理の手順について説明したフローチャートである。Ｆ１は，ある割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する。通常，この処理はメインＣＰＵ１によって自動的に実行される。Ｆ２は，タスク走行状態メモ２１２（図１０参照）の走行残りチック数が，０または負の全てのタスクの走行禁止フラグをオンにする。

【０１４９】

Ｆ３では，全ての走行禁止フラグがオフであるＴＣＢが，全ての走行禁止フラグがオンのＴＣＢの前に来るように，タスク処理の待ち行列を作り直す。すなわち，走行禁止フラグがオンのタスクの方がフラグがオフのタスクよりも前に走るようにする。Ｆ４は，割り込まれた状態がタスク処理中ならば，タスクの実行環境をＴＣＢへ保存して，割り込みを許可し，スケジューラへ飛ぶ。反対に，割り込まれた状態がタスク処理中でなければ，割り込まれた処理に復帰する処理である。

【０１５０】

図１３は，基準周期割り込み以外の，一般の割り込み処理の手順を示すフローチャートである。Ｇ１は，ある割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する，通常この処理はメインＣＰＵ１によって自動的に実行される。Ｇ２は，未使用のＩＣＢをＩＣＢプール（未使用ＩＣＢを保持しておく場所）から１つ獲得する。Ｇ３は，ＩＣＢに，この割り込み処理（自分自身の）の，実行環境を格納（保存）する。この中には，次に実行する命令のアドレスとして後述するＧ１５の処理の先頭アドレスをＩＣＢに格納することも含まれる。

【０１５１】

Ｇ４は，ここで，作成したＩＣＢを，スケジューラが実行するＩＣＢ待ち行列に追加する。Ｇ５は，Ｇ１の割り込みの禁止を解除する。これ以降は，再び割り込みが発生する可能性がある。以上の処理は割り込み禁止時間をできる限り短くするためのものである。

【０１５２】

Ｇ６は，割り込まれた処理が，割り込み処理か，あるいは，ＲＴＯＳか，それともタスクの処理中かを判定する。Ｇ７は，割り込まれた処理が割り込み処理またはＲＴＯＳを実行中の場合で，このときは割り込まれた処理に直接戻る。Ｇ８は，割り込まれた処理がタスクの処理を実行中のときで，このときにあっては，まず，割り込まれたタスクのＴＣＢを探し出す。Ｇ９は，Ｇ８において見つけたＴＣＢへ割り込まれたタスクの実行環境を格納する。

【０１５３】

Ｇ１０は，現在処理を行なっているタスクが，走行する割合を保証する対象のタスクであるか否かを判定する処理である。具体的には，タスク走行状態メモ２１２（図１０参照）の走行残りチック数の欄が１以上のタスクか否かを判定する。Ｇ１１は，Ｇ１０において対象と判定した場合における処理であり，内部カウンタレジスタ２０２（図９参照）の値を，基準周期２１１（図１０参照）に保存してある値から減算する。この結果が当該タスクの，この回に連続して走ったチック数である。これをタスク走行状態メモ２１１の当該タスクの走行残りチック数の欄から減算する。

【０１５４】

Ｇ１２は，Ｇ１１において減算した値を，リアルタイムクロック２０１（図９参照）の走行時間警告用の内部カウンタレジスタ２０５の値に加える。走行周期を保証したタスクが，走行し終った分だけ，走行周期を保証したタスク以外のタスクを走らせるための時間が増えるわけである。

【０１５５】

Ｇ１３は，作成したＴＣＢを，スケジューラが実行するＴＣＢ待ち行列に追加する。このとき，タスク走行状態メモ２１２（図１０参照）の走行禁止フラグの欄が１つのタスクで

もオンになっていれば，ＴＣＢ待ち行列において，走行禁止フラグがオフのタスクのＴＣＢは，走行禁止フラグがオンのタスクのＴＣＢよりも前につなぐ。

【０１５６】

Ｇ１４は，スケジューラへ飛ぶ。Ｇ１５は，スケジューラから戻ってきたときの処理であり，スケジューラへ飛んだ後すぐにこの処理が実行されるとは限らず，他のＩＣＢが処理された後の場合もあるが，いずれはここへ戻ってくる。ここからが割込毎に固有の本来の割り込み処理であり，割り込み要因によって処理内容が異なる。Ｇ１６は，割り込み処理の終了であり，通常のサブルーチンのリターン命令が一般的であり，このリターン命令によって，スケジューラのＩＣＢ処理部に戻る。

【０１５７】

次に，図１４を用いて，ＲＴＯＳのスケジューラの動作について説明する。以下，説明の都合上，本実施例においては，優先順位が最低であって，外部に対して何も処理を行なわない，アイドルタスクが常に走っているものとして説明を行なう。

【０１５８】

Ｈ１は，割り込み処理（ＩＣＢ）の待ち行列があるか否かを調べる処理である。Ｈ２は，Ｈ１において，割り込み処理の待ち行列があったと判断した場合の処理であり，その待ち行列を実行するために最初に割り込みを禁止する。これは，割り込み処理の待ち行列のリストを処理している最中に，割り込みが発生して割り込み処理の待ち行列に追加を行おうとするとリストの整合性が保たれなくなるためである。

【０１５９】

Ｈ３は，割り込み処理の待ち行列のリストから先頭のＩＣＢを取り除く。Ｈ４は，割り込み禁止を解除する。Ｈ５は，Ｈ３において取り除いたＩＣＢからＧ３（図１３参照）において格納した実行アドレスやレジスタなどの割り込み処理の実行環境を取り出して，それを実行する。これが，Ｇ１５（図１３参照）の処理である。Ｈ６は，Ｇ１６（図１３参照）から戻ってくるところであり，再び割り込みを禁止する。Ｈ７は，Ｈ３において取り除いたＩＣＢをＩＣＢプールへ返す。Ｈ８は，割り込み禁止を解除してからＨ１へ戻る。後は割り込み処理待ち行列がある間はＨ１からＨ７の処理を繰り返す。

【０１６０】

Ｈ９は，Ｈ１において，割り込み処理の待ち行列がなかったときの処理であり，ここでは，ＴＣＢの待ち行列があるか否かを判定する。Ｈ１０は，Ｈ９において，ＴＣＢの待ち行列があると判定した場合の処理であり，割り込みを禁止し，その待ち行列を実行するために最初に割り込みを禁止する。Ｈ１１は，当該ＴＣＢのタスクが走行割合を保証するタスクである場合には，タスク開始チックメモ２１０（図１０参照）にリアルタイムクロック２０１（図９参照）の走行時間計測用の内部カウンタレジスタ２０２（図９参照）の値を保存しておく。

【０１６１】

Ｈ１２は，割り込み禁止を解除する。Ｈ１３は，Ｈ１１でタスク処理の待ち行列（リスト）から取り除いたＴＣＢからＧ９（図１３参照）において格納したタスクの実行環境を取り出して，それを実行する。

【０１６２】

以上で述べたように，実施例２に係るＮＣ装置の制御ソフトウェア実行システムＡ２は，通常のプライオリティベースのスケジューリング方式しか持たないＲＴＯＳに，タスク走行時間の保証を与える制御を実行できる機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。

【０１６３】

実施例２によれば，ＮＣ装置の制御ソフトウェア実行システムＡ２は，ＮＣ装置機能を実現するタスクの走行時間による制御を実行することができる。そのため，タスクに割り当てた時間の無駄を少なくできる。さらに，タスクをその走行時間において制御できるので，システムの構築が容易になる。

【０１６４】

〔実施例３〕
次に，実施例３の構成について説明する。実施例３に係るＮＣ装置の制御ソフトウェア実行システムＡ３の全体構成を示す図１５の内容にあっては，実施例２において示した図９の内容とほぼ同様である。ただし，実施例３にあっては，カウンタレジスタ２０２と同様にリアルタイムクロック２０１の内部にあるカウンタレジスタ２０６が設けられている。実施例３では，このカウンタレジスタ２０６を連続走行終了割り込み用に用いる。
【０１６５】
次に，実施例３の動作について説明する。図１６，図１７は，タスクの連続走行を保証するのに必要なデータ構造を示したもので，図１６は，“タスク終了チック数”のリスト構造である。２２０は，“タスク終了チックブロック”を表している。タスク終了ブロックの構成要素は，タスク終了ブロックを連結するための“リンク（ポインタ）”，“タスク名”，“終了チック数”などである。今までの説明と同様本実施例の説明でも内部的には全てチック数により時間の制御を行なうものとする。
【０１６６】
図１７は，“タスク連続走行保証テーブル”２２１の内容を示している。このテーブルには“タスク名”と“連続走行保証チック数”と“実行中フラグ”の欄があり，連続走行保証チック数の値は，システム起動時，またはＳＶＣにより設定する。連続走行を指定しないタスクに対してはこの欄は０に設定しておく。実行中フラグ欄は現在連続走行を保証されたタスクが走っているときはオン，それ以外のときはオフとし，このフラグは同時には，２つ以上オンすることはない。このテーブルはタスク起動のＳＶＣにおいて，そのパラメータとして指定することにすれば必要はなくなるが，以下の説明の都合上，ここではこのテーブルを用いることにする。
【０１６７】
ＲＴＯＳのスケジューラは，連続走行が保証されているタスクが走行しているときに，割り込みが発生しても割り込み処理の終了後における再スケジューリングは行なわないことによって，タスクの連続走行を保証する。連続走行が保証されているタスクの実行を終了させるのは“連続走行終了割り込み”が発生したときである。
【０１６８】
ＲＴＯＳは，タスク起動のＳＶＣが発行されたときは，まず，そのタスクが連続走行を保証されたタスクか否かを判定し，もし該当タスクであると判定した場合には，タスク連続走行保証テーブル（図１７参照）から連続走行チック数をリアルタイムクロック２０１（図１５参照）の内部カウンタレジスタ２０６に設定する。設定したクロック数が経過すると，“連続走行終了割り込み”が発生する。
【０１６９】
最初に，割り込み処理について説明する。図１８は，連続走行割り込みの処理手順について示したフローチャートである。Ｉ１は，ある割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する。通常，この処理はメインＣＰＵ１によって自動的に実行される。Ｉ２は，現在走っている，連続走行を保証されたタスクのＴＣＢを探し出して，現在のタスクの実行環境をＴＣＢへ保存する。
【０１７０】
次に，Ｉ３は，当該タスクの実行中フラグ（図１７参照）をオフする。Ｉ４は，タスク処理の待ち行列を優先順位に応じて作り直す。さらに，Ｉ５は，割り込まれた状態がタスク処理中ならばタスクの実行環境をＴＣＢへ保存してスケジューラへ飛び，そうでなければ割り込まれた処理に復帰させる処理である。このとき，復帰すると同時に割り込みを許可する。
【０１７１】
次に，一般の割り込みの処理手順について図１９のフローチャートを用いて説明する。Ｊ１は，ある割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する，通常，この処理はメインＣＰＵ１によって自動的に実行される。Ｊ２は，未使用のＩＣＢをＩＣＢプール（未使用ＩＣＢを保持しておく場所）から１つ獲得する

。Ｊ３は，ＩＣＢに，この割り込み処理（自分自身の）の，実行環境を格納する。この中には，次に実行する命令のアドレスとしてＪ１３の処理の先頭アドレスをＩＣＢに格納することも含まれる。

【０１７２】

Ｊ４は，ここで，作成したＩＣＢを，スケジューラが実行するＩＣＢ待ち行列に追加する。Ｊ５は，Ｊ１の割り込みの禁止を解除する。これ以降は再び割り込みが発生する可能性がある。以上の処理は，割り込み禁止時間をできる限り短くするためのものである。Ｊ６は，割り込まれた処理が割り込み処理またはＲＴＯＳか，それともタスクの処理中かを判定する。

【０１７３】

Ｊ７は，割り込まれた処理が割り込み処理またはＲＴＯＳを実行中のときで，このときは割り込まれた処理に直接戻る。反対に，Ｊ８は，割り込まれた処理がタスクの処理を実行中のときで，このときは，まず，割り込まれたタスクのＴＣＢを探し出す。Ｊ９は，現在処理を行なっているタスクが，連続走行を保証する対象のタスクであるか否かを判定する処理である。具体的には，タスク連続走行保証テーブル（図１７参照）の連続走行保証チック数の欄が１以上のタスクか否かを判定する。連続走行保証の対象タスクであると判定した場合には，以下のＪ１０，Ｊ１１の処理は実行しないで，Ｊ１２へ処理が移行する。

【０１７４】

Ｊ１０は，連続走行保証の対象外のタスクであると判定した場合の処理で，Ｊ８において見つけたＴＣＢへ割り込まれたタスクの実行環境を格納する。Ｊ１１は，ＴＣＢ待ち行列（図５参照）を優先順位に応じて作り直す。Ｊ１２は，スケジューラへ飛ぶ。Ｊ１３は，スケジューラから戻ってくるとここへ来る処理である。スケジューラへ飛んだ後，すぐにこの処理が実行されるとは限らず，他のＩＣＢが処理された後の場合もあるが，いずれはここへ戻ってくる。ここからが割込毎に固有の本来の割り込み処理であり，割り込み要因によって，その処理内容は異なる。Ｊ１４は，割り込み処理の終了であり，通常のサブルーチンのリターン命令が多い，このリターン命令によって，スケジューラのＩＣＢ処理部に戻る。

【０１７５】

以上で述べたように，実施例３に係るＮＣ装置の制御ソフトウェア実行システムＡ３は，通常のプライオリティベースのスケジューリング方式しか持たないＲＴＯＳに，タスクの連続した走行時間の保証を与える制御を実行できる機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。

【０１７６】

実施例３によれば，ＮＣ装置の制御ソフトウェア実行システムＡ３は，ＮＣ装置機能を実現するタスクに連続した走行時間を保証できるようになる。そのため，タスクに割り当てた時間の無駄を少なくできる。さらに，タスクをその走行時間において制御できるので，システムの構築が容易に行なえる。

【０１７７】

〔実施例４〕

次に，実施例４の構成について説明する。実施例４に係るＮＣ装置の制御ソフトウェア実行システムＡ４の全体構成を示す図２０の内容は，実施例１において示した図１の内容とほぼ同様である。ただし，実施例１にあっては，リアルタイムクロック２０１の割り込みポートを１チャンネルしか使用していないが，本実施例では４チャンネル使用している点が異なる。以下，図２～図４および図１０の内容を参照して本実施例を説明する。

【０１７８】

すなわち，２０２はリアルタイムクロック２０１の内部にあるカウンタレジスタであり，これは外部からクロック信号が入力されるたびにその値を１ずつ減らしていく。０になると再び初期値が設定されるか，あるいはメインＣＰＵ１から再び値を設定されるまで動作を止める。この動作の選択や初期値の設定，変更ははメインＣＰＵ１から自由に実行することができる。

【０１７９】
また，２０４，２０５，２０６もリアルタイムクロック２０１の内部にあるカウンタレジスタである。これらも外部からクロック信号が入力されるたびにその値を１ずつ減らしていく。０になると再び初期値が設定される。また，同様に初期値はメインＣＰＵ１から自由に設定できる。また，２０２，２０４，２０５，２０６は，それぞれ独立に設定，解除することができる。本実施例では，カウンタレジスタ２０２～２０６をそれぞれタスク走行時間計測用，基準時間割り込み用，走行時間警告用，タスク走行割り込み用に用いる。
【０１８０】
次に，ＮＣ装置の制御ソフトウェア実行システムＡ４の動作について説明する。実施例１と同様に，各タスクは，予め，あるいは，そのタスクが開始させられるときにＳＶＣによってその実行周期を登録する。
【０１８１】
このデータは，ＲＴＯＳが，例えば，図２に示したような“タスク走行周期テーブル”５０に登録する。図２は，タスク毎に“走行周期”と“繰り返し回数”を持ったデータ構造である。走行周期は，タスクの次に走るまでのクロック数の値であり，繰り返し回数は何回そのタスクを起動するかを示す回数である。例えば，この値がマイナスの値ならば∞を示すとする。その場合は，そのタスクは繰り返し起動するものとする。このタスク走行周期テーブル５０を用いて，ＲＴＯＳは，タスクに起動がかかるたびに，図３に示したような“タスク走行待ちリスト”５１を作成する。
【０１８２】
図３において，リストの中における走行待ちブロック５２は，それぞれ“タスク名”と，現在からどれだけの時間の経過後に走らなくてはならないかの“走行予定クロック数”，それに同時に走らせなくてはならないタスク名を示す“同時タスク名”から成るブロックである。このタスク走行待ちリストを，ＲＴＯＳが作成する方法は，実施例１において図４に示したものと同様なので，その説明は省略する。
【０１８３】
また，実施例２において示したように，図１０は，タスクの走行割合を保証するのに，必要なデータ構造を示したもので，２１０は“タスク開始チックメモ”である。ＲＴＯＳが，タスクを起動する直前に，リアルタイムクロック２０１（図２０参照）の内部カウンタレジスタ２０２（図２０参照）の値を記録しておく。その後，そのタスクが終了してＲＴＯＳに戻ったときにおける内部カウンタレジスタ２０２（図２０参照）との差を求めることによって，タスクが走った時間を求める。
【０１８４】
また，図１０に示した“基準周期”２１１，“タスク走行状態メモ”２１２，“走行残りチック数”の欄，“繰り返し回数”の欄および“走行禁止フラグ”の欄については，実施例２と同様なので，その説明を省略する。
【０１８５】
図４に示したＢ４や，Ｂ１６において走行待ちリストの先頭ブロックの走行予定クロック数を，リアルタイムクロック２０１（図２０参照）のカウンタレジスタ２０２（図２０参照）に入れ，その指定したクロック数が経過した後，“タスク起動割り込み”が発生する。このタスク起動割り込みの処理については，実施例１の図７に基づいて説明したものと同様であるので，その説明を省略する。
【０１８６】
図２０に示した制御ソフトウェア実行システムＡ４は，稼働し始めたときなど走行割合を保証するタスクが動き始める前に，予めプログラマブルインターバルタイマ２０１の内部におけるカウンタレジスタ２０４に対して，基準周期２１１（図１０参照）の値を設定しておく。この設定したチック数が経過した後に，基準周期割り込みが発生する。
【０１８７】
基準周期割り込みの処理手順については，実施例２の図１１に基づいて説明した手順と同様なので，その説明を省略する。また，走行時間警告割り込みの処理手順についても，実

施例２の図１２に基づいて説明した手順と同様なので，その説明を省略する。また，基準周期割り込み以外の，一般の割り込みの処理手順についても，実施例２の図１３に基づいて説明した手順と同様なので，その説明を省略する。さらに，ＲＴＯＳのスケジューラの動作についても，実施例２の図１４に基づいて説明した手順と同様なので，その説明を省略する。

【０１８８】

ここで，タスク起動割り込み，基準時間割り込み，走行時間警告割り込み，および，一般の割り込みの各処理について，ＴＣＢ待ち行列への操作を中心にまとめる。

【０１８９】

タスク起動割り込みは，この割り込みによって起動するタスクのＴＣＢを，ＴＣＢ待ち行列の先頭に挿入する。また，基準時間割り込みは，全てのタスクを走行可能とする。走行時間警告割り込みは，走行時間の割合を保証しているタスクのＴＣＢを，そうでないタスクのＴＣＢより前に持ってくると共に，タスク走行状態メモ２１２（図１０参照）の走行禁止フラグをオンすることによって，走行時間警告割り込みが発生したことを，一般の割り込みに対して知らせる。さらに，一般の割り込みは，走行時間警告割り込みの処理によって，走行時間警告割り込みが発生した場合には，基準時間割り込みが発生するまでは，走行時間を保証しているタスクのＴＣＢをそうでないタスクのＴＣＢよりＴＣＢ待ち行列の前につなぐ。

【０１９０】

ここで，タスク起動割り込みと一般の割り込みの優先順位に関して，どちらかを選択しておく。タスク起動割り込みを，優先すればタスクの起動時間は正確になるが，タスクの走行割合は多少不正確になる。反対に，一般の割り込みを優先すれば，タスクの走行割合は正確になるが，タスクの起動時間は多少不正確になる。これらは，何らかの手段によって閾値を決めておいてもよい。例えば，タスクの起動時間の遅れが，ある時間までは走行割合を優先するが，それ以上はタスク起動を優先するなどの方法を取ればよい。

【０１９１】

以上で述べたように，実施例４に係るＮＣ装置の制御ソフトウェア実行システムＡ４は，通常のプライオリティベースのスケジューリング方式しか持たないＲＴＯＳに，タスク起動周期とタスク走行割合の保証を与える制御を行なえる機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。

【０１９２】

実施例４によれば，ＮＣ装置の制御ソフトウェア実行システムＡ４は，ＮＣ装置機能を実現するタスクの正確な起動時間と走行時間による制御を行なえるようになる。そのため，タスクに割り当てた時間の無駄を少なくでき，さらに，タスクをその走行時間で制御できるので，システムの構築が容易に行なえる。

【０１９３】

〔実施例５〕

次に，実施例５について説明する。実施例５に係るＮＣ装置の制御ソフトウェア実行システムＡ５の全体構成は，実施例１に示したものと同様であるので，その説明を省略する。

【０１９４】

次に，実施例５に係るＮＣ装置の制御ソフトウェア実行システムＡ５の動作について説明する。実施例１と同様に，各タスクは，予め，あるいは，そのタスクが開始させられるときにＳＶＣによってその実行周期を登録する。

【０１９５】

このデータは，ＲＴＯＳが，例えば，図２に示したような“タスク走行周期テーブル”５０に登録する。このタスク走行周期テーブル５０における“走行周期”と“繰り返し回数”については，実施例１と同様なので，その説明を省略する。

【０１９６】

図３において，リストの中の各走行待ちブロックは，ぞれぞれ“タスク名”と，現在からどれだけの時間の経過後に走らなくてはならないかの“走行予定クロック数”，それに同

時に走らせなくてはならないタスク名を示す“同時タスク名”から成るブロックである。このタスク走行待ちリストを，ＲＴＯＳが作成する方法は，実施例１において図４で示したものと同様なので，その説明は省略する。

【０１９７】

また，実施例３と同様に，図１６，図１７は，タスクの連続走行の走行割合を保証するのに，必要なデータ構造を示したもので，図１６は，“タスク終了チック数”のリスト構造である。２２０は，“タスク終了チックブロック”を表している。タスク終了ブロックの構成要素はタスク終了ブロックを連結するためのポインタ，“タスク名”，“終了チック数”などである。今までの説明と同様本実施例の説明でも内部的には全てチック数で時間の制御を行なうものとする。“実行中フラグ”欄は現在連続走行を保証されたタスクが走っている時はオン，それ以外の時はオフとする。このフラグは同時には２つ以上オンすることはない。

【０１９８】

図１７は，“タスク連続走行保証テーブル”２２１である。このテーブルには“タスク名”と“連続走行保証チック数”の欄があり，連続走行保証チック数の値はシステム起動時，またはＳＶＣにより設定する。連続走行を指定しないタスクに対してはこの欄は０にしておく。このテーブルはタスク起動のＳＶＣにおいて，そのパラメータとして指定することにすれば，必要はなくなるが，以下の説明の都合上，ここではこのテーブルを用いることにする。

【０１９９】

ＲＴＯＳのスケジューラは，連続走行が保証されているタスクが走行しているときに，割り込みが発生しても割り込み処理の終了後における再スケジューリングは行なわないことによって，タスクの連続走行を保証する。連続走行が保証されているタスクの実行を終了させるのは“連続走行終了割り込み”が発生したときである。

【０２００】

ＲＴＯＳはタスク起動のＳＶＣが，発行されたときには，まずそのタスクが連続走行を保証されたタスクか否かを判定して，もし該当タスクであると判定した場合には，タスク連続走行保証テーブル２２１（図１７参照）から連続走行チック数をリアルタイムクロック２０１（図１参照）の内部カウンタレジスタ２０６に設定する。設定したクロック数が経過すると，“連続走行終了割り込み”が発生する。

【０２０１】

連続走行終了割り込みの処理手順については，実施例３に示した図１８の内容と同様なので，その説明を省略する。基準周期割り込み以外の，一般の割り込みの処理手順についても，実施例３に示した図１９の内容と同様なので，その説明を省略する。スケジューラについても，従来例と同じであるので，その説明を省略する。

【０２０２】

ここで，連続走行終了割り込みと，一般の割り込みの処理について，ＴＣＢ待ち行列への操作を中心にまとめる。すなわち，

▲１▼　タスク起動割り込みは，この割り込みによって起動するタスクのＴＣＢを，ＴＣＢ待ち行列の先頭に挿入する，

▲２▼　連続走行終了割り込みは，実行中のタスクの実行中フラグをオフして他のタスクの走行を許可する，

▲３▼　一般の割り込みは，連続走行を保証するタスクの場合には，ＴＣＢ待ち行列のディスパッチを実行しない，

がある。

【０２０３】

ここで，上記▲１▼と▲３▼の優先順位はどちらかを選択しておく。▲１▼を優先すればタスクの起動時間は正確になるが，タスクの連続走行時間は多少不正確になる。一方▲３▼を優先すれば，タスクの連続走行時間は正確になるが，タスクの起動時間は多少不正確になる。これらは，何らかの手段によって閾値を決めておいてもよい。例えば，タスクの

起動時間の遅れが，ある時間までは連続走行を優先するが，それ以上は，タスク起動を優先するなどの方法を取ればよい。

【０２０４】

以上述べたように，実施例５に係るＮＣ装置の制御ソフトウェア実行システムＡ５は，通常のプライオリティベースのスケジューリング方式しか持たないＲＴＯＳに，タスク起動周期とタスクの連続走行の保証を与える制御を行なえる機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。

【０２０５】

実施例５によれば，ＮＣ装置の制御ソフトウェア実行システムＡ５は，ＮＣ装置機能を実現するタスクの正確な起動時間と連続走行時間による制御を実行することができる。そのため，タスクに割り当てた時間の無駄を少なくでき，さらに，タスクをその走行時間内において制御できるので，システムの構築が容易に行なえる。

【０２０６】

〔実施例６〕

次に，実施例６について説明する。図２１は，実施例６に係るＮＣ装置の制御ソフトウェア実行システムＡ６の全体構成を示し，その内容は，実施例１とほぼ同様である。ただし，実施例１にあっては，リアルタイムクロック２０１の割り込みポートを１チャンネルしか使用しないのに対し，本実施例にあっては，２チャンネル使用している。

【０２０７】

図において，２０２，２０４は，リアルタイムクロック２０１の内部にあるカウンタレジスタである。カウンタレジスタ２０２，２０４は，それぞれ独立に設定，解除ができる。本実施例にあっては，カウンタレジスタ２０２，２０４をそれぞれタスク走行時間計測用に用いる。

【０２０８】

次に，ＮＣ装置の制御ソフトウェア実行システムＡ６の動作について説明する。本実施例に係る制御ソフトウェア実行システムＡ６のＲＴＯＳは，予め，例えば，システムを稼働したとき等に，リアルタイムクロック２０１の内部カウンタレジスタ２０２，２０４は，０になると再び初期値が設定され，また，０になっても，割り込みが発生しないように設定しておく。

【０２０９】

また，タスクにはいる直前に，そのカウンタ値を読み取って記憶しておく。また，そのタスクを抜けた直後に，再びカウンタ値を読み取って記憶しておいた値との差を求める。その値が，タスクが走ったクロック数である。その後タスク毎に，この値の和を記憶しておく場所を設けておいて，そこに加えていくことにより，各タスクの走行時間の合計を記録する。

【０２１０】

以下，本実施例に係るＮＣ装置の制御ソフトウェア実行システムＡ６の動作について，さらに詳しく説明する。図２２は，各タスクの走行時間を記録するのに必要なデータ構造であり，２３０はカウンタ値メモ，２３１はタスク走行時間テーブルであり，図に示す通り，タスク毎の合計チック数の欄がある。図２３は，一般の割り込み処理の処理手順を示すフローチャートであり，図２４は，スケジューラの処理手順を示す図である。

【０２１１】

図２３は，一般の割り込みの処理手順について示したフローチャートであり，Ｋ１は，ある割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する。通常この処理はメインＣＰＵ１によって自動的に実行される。Ｋ２は，未使用のＩＣＢをＩＣＢプール（未使用ＩＣＢを保持しておく場所）から１つ獲得する。Ｋ３は，ＩＣＢに，この割り込み処理（自分自身）の，実行環境を格納する。この中には次に実行する命令のアドレスとしてＫ８の処理の先頭アドレスをＩＣＢに格納することも含まれる。Ｋ４は，ここで，作成したＩＣＢを，スケジューラが実行するＩＣＢ待ち行列に追加する。Ｋ５は，Ｋ１の割り込みの禁止を解除する。これ以降は，再び割り込みが発生する可

能性がある。以上の処理は割り込み禁止時間をできる限り短くするためである。

【０２１２】

Ｋ６は，割り込まれた処理が，割り込み処理またはＲＴＯＳか，それともタスクの処理中かを判定する。Ｋ７は，割り込まれた処理が割り込み処理またはＲＴＯＳを実行中であると判定した場合の処理であって，割り込まれた処理に直接戻る。Ｋ８は，割り込まれた処理がタスクの処理を実行中の場合であって，割り込まれたタスクのＴＣＢを探し出す。Ｋ９は，Ｋ８において，見つけたＴＣＢへ割り込まれたタスクの実行環境を格納する。

【０２１３】

Ｋ１０は，リアルタイムクロック２０１（図２１参照）の内部カウンタレジスタ２０４の値を読み込んで，カウンタ値メモ２３０（図２２参照）から減算する。この結果が当該タスクがこの回に連続して走ったチック数である。Ｋ１１は，Ｋ１０において求めた値をタスク走行時間テーブル２３１（図２２参照）の当該タスクの合計チック数の欄に加える。Ｋ１２は，作成したＴＣＢを，スケジューラが実行するＴＣＢ待ち行列に優先順位の順番に追加する。Ｋ１３は，スケジューラへ飛ぶ。

【０２１４】

Ｋ１４は，スケジューラから戻ってきた後に実行する処理である。スケジューラへ飛んだ後すぐにこの処理が実行されるとは限らず，他のＩＣＢが処理された後の場合もあるが，いずれはここへ戻ってくる。ここからが割込毎に固有の本来の割り込み処理であり，割り込み要因によって処理は異なってくる。Ｋ１５は，割り込み処理の終了であり，通常のサブルーチンのリターン命令が一般的的であり，このリターン命令によって，スケジューラのＩＣＢ処理部へ戻る。

【０２１５】

図２４は，本実施例に係るＲＴＯＳのスケジューラの動作を説明したものである。以下，説明の都合上，本実施例においては，優先順位が最低で，外部に対して何ら処理を実行しないアイドルタスクが常に走っているものとする。

【０２１６】

Ｌ１は，割り込み処理（ＩＣＢ）の待ち行列があるか否かを判定する処理である。Ｌ２は，割り込み処理の待ち行列があったときの処理であり，その待ち行列を実行するために最初に割り込みを禁止する。これは，割り込み処理の待ち行列のリストを処理している最中に，割り込みが発生して割り込み処理の待ち行列に追加しようとするとリストの整合性が保たれなくなるためである。

【０２１７】

Ｌ３は，割り込み処理の待ち行列のリストから先頭のＩＣＢを取り除く。Ｌ４は，割り込み禁止を解除する。Ｌ５は，Ｌ３において取り除いたＩＣＢからＫ３（図２３参照）において格納した実行アドレスやレジスタなどの割り込み処理の実行環境を取り出して実行する。これがＫ１４（図２３参照）の処理である。Ｌ６は，Ｋ１５（図２３参照）から戻ってくるところであり，再び割り込みを禁止する。Ｌ７は，Ｌ３において取り除いたＩＣＢをＩＣＢプールへ返す。Ｌ８は，割り込み禁止を解除してからＬ１へ戻る。後は割り込み処理待ち行列がある間はＬ１からＬ７の処理を繰り返す。

【０２１８】

Ｌ９は，ＴＣＢ待ち行列があるか否かを判断するものであり，Ｌ１０は，タスク処理の待ち行列があったと判定した場合の処理であり，その待ち行列を実行するために最初に割り込みを禁止する。Ｌ１１は，割り込み処理の待ち行列のリストから先頭のＴＣＢを取り除く。Ｌ１２は，カウンタ値メモ２３０（図２２参照）にリアルタイムクロック２０１（図２１参照）の走行時間計測用の内部カウンタレジスタ２０４（図２２）の値を保存しておく。

【０２１９】

Ｌ１３は，割り込み禁止を解除する。Ｌ１４は，Ｌ１１においてタスク処理の待ち行列（リスト）から取り除いたＴＣＢからＫ９（図２３参照）において格納したタスクの実行環境を取り出して実行する。このときにあっては，その処理へ飛び，ここへは再び戻ってこ

ない。Ｌ１５は，Ｌ９においてＴＣＢ待ち行列がないと判断した場合における処理であり，割り込み待ちを行う。

【０２２０】

以上のようにして，任意の時間，本システムを実行させた後，タスク走行時間テーブル２３１（図２２参照）を調べれば，各タスクが実際に走った（処理を行なった）時間を確認することができる。

【０２２１】

なお，タスクの走行時間のみではなく，タスク走行時間テーブル２３１（図２２参照）に走行回数の欄も設けて，タスクの走行回数をも記録するようにしておけば，以下で説明するタスクの走行周期，走行割合，走行時間を決めるときにも役立つ。

【０２２２】

さらに，カウンタ値メモ２３０（図２２参照）だけでなく，タスク終了カウンタ値メモも設けて割り込み処理の先頭で，リアルタイムクロック２０１（図２１参照）の内部カウンタレジスタ２０４（図２１参照）の値を保存しておき，割り込み処理（図２３参照）のＫ１０の部分において，カウンタ値メモ２３０からタスク終了カウンタ値メモを減算した値をタスクが連続して走行したチック数とすれば割り込み処理の時間も除いたタスクが純粋に走行した時間が求められる。

【０２２３】

以上で述べたように，実施例６に係るＮＣ装置の制御ソフトウェア実行システムＡ６は，タスクの実際の走行時間を知ることができる機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。

【０２２４】

実施例６によれば，ＮＣ装置の制御ソフトウェア実行システムＡ６は，ＮＣ装置機能を実現するタスクの正確な実行時間を知ることにより，各タスクの走行時間の割り当てなどが，容易に行なえるようになり，システムの構築が容易となる。

【０２２５】

〔実施例７〕

次に，実施例７について説明する。実施例７に係るＮＣ装置の制御ソフトウェア実行システムＡ７の全体構成は，実施例６と同様であるので，その説明を省略する。

【０２２６】

次に，本実施例に係るＮＣ装置の制御ソフトウェア実行システムＡ７の動作について説明する。本実施例における制御ソフトウェア実行システムＡ７（図２１参照）のＲＴＯＳは，予め，例えば，システムを稼働したとき等に，リアルタイムクロック２０１の内部カウンタレジスタが０になると再び初期値が設定され，また，０になっても，割り込みが発生しないように設定しておく。

【０２２７】

また，タスクにはいる直前に，そのカウンタ値を読み取って記憶しておく。そのタスクを抜けた直後に，再びカウンタ値を読み取って記憶しておいた値との差を求める。その値が，今回このタスクが走ったクロック数である。その後，タスク毎に，この値の和ををを記憶しておく場所を設け，そこに加えていくことにより，各タスクの走行時間の合計を記録する。さらに，従来例におけるスケジューラのＩＣＢの処理を，スケジューラでなく，優先順位がもっとも高いタスクで実行するものである。

【０２２８】

以下，本実施例に係るＮＣ装置の制御ソフトウェアを実行する動作について，さらに詳しく説明する。図２５は，スケジューラの処理手順を示すフローチャートであり，図２６は，ＩＣＢを処理する最高優先順位を持った，割り込み処理実行タスクの処理手順を示すフローチャートである。データ構造としては，従来例と同じＩＣＢ（インタラプトコントロールブロック）を用いる。なお，割り込み処理に関しては，従来例と同じであるので，その説明を省略する。

【０２２９】

図２５は，本実施例に係るＲＴＯＳのスケジューラの動作を説明したものであり，アイドルタスクを設けることにより，常に少なくともアイドルタスクは，実行を待っているものとする。Ｍ１は，タスク処理の待ち行列（ＴＣＢ）があったときの処理であり，その待ち行列を実行するために最初に割り込みを禁止する。Ｍ２は，割り込み処理の待ち行列のリストから先頭のＴＣＢを取り除く。Ｍ３は，カウンタ値メモ２３０（図２２参照）にリアルタイムクロック２０１（図２１参照）の走行時間計測用の内部カウンタレジスタ２０４（図２１参照）の値を保存しておく。

【０２３０】

Ｍ４は，割り込み禁止を解除する。Ｍ５は，Ｍ３においてタスク処理の待ち行列（リスト）から取り除いたＴＣＢから割り込み処理で格納したタスクの実行環境を取り出して実行する。このときにあっては，その処理へ飛び，ここへは再び戻ってこない。

【０２３１】

図２６は，本実施例に係る割り込み処理実行タスクの動作を説明したものであり，Ｎ１は，割り込み処理（ＩＣＢ）の待ち行列があるか否かを判定する処理である。Ｎ２は，割り込み処理の待ち行列があったと判断した場合の処理であって，その待ち行列を実行するために最初に割り込みを禁止する。これは，割り込み処理の待ち行列のリストを処理している最中に，割り込みが発生して割り込み処理の待ち行列に追加を行なおうとするとリストの整合性が保たれなくなるためである。

【０２３２】

また，Ｎ３は，割り込み処理の待ち行列のリストから，先頭のＩＣＢを獲得する。Ｎ４は，割り込み禁止を解除する。Ｎ５は，Ｎ３で取り除いたＩＣＢから，割り込み処理で格納した実行アドレスやレジスタなどの割り込み処理の実行環境を取り出して実行する。Ｎ６は，Ｎ３で取り除いたＩＣＢをＩＣＢプールへ返却する。その後は，Ｎ１へ戻り，割り込み処理待ち行列がある間はＮ１からＮ６の処理を繰り返す。Ｎ７は，割り込み処理（ＩＣＢ）の待ち行列が無くなったときの処理でタスク終了のＳＶＣを発行して他のタスクにメインＣＰＵ１の使用権を譲り渡す。

【０２３３】

以上のように，任意の時間，本実施例を走らせた後，タスク走行時間テーブル２３１（図２２参照）を調べれば，各タスクが実際に走った（処理を行なった）時間を知ることができる。

【０２３４】

さらに，割り込み処理もタスクレベルで実行することにより，割り込み処理実行タスクの走行時間が，割り込み処理の時間と考えることができるため，割り込み処理の処理時間も測定できる。また，割り込み処理実行タスク内で，割り込み要因により別の走行時間を記録できるようにしておけば，割り込み要因毎の処理時間を測定できる。

【０２３５】

以上で説明したとおり，実施例７に係るＮＣ装置の制御ソフトウェア実行システムＡ７は，タスクと割り込み処理の実際の走行時間を知ることができる機能を追加した方式であり，その方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。

【０２３６】

実施例７によれば，ＮＣ装置の制御ソフトウェア実行システムＡ７は，ＮＣ装置機能を実現するタスクと割り込み処理の正確な実行時間を知ることにより，各タスクの走行時間の割り当てなどが容易に実行できるようになり，システムの構築が容易となる。

【０２３７】

〔実施例８〕

次に，実施例８について説明する。タスクや割り込み処理の走行時間を計測する手段を備え，さらにタスクの走行時間を変える手段を備えるＮＣ装置において，それを実現するためのＮＣ装置の制御ソフトウェア実行システムＡ８の全体構成は，タスクの走行時間や割合の指定の仕方によって複数例あるが，この実施例では説明の都合上，タスクの走行割合を指定できる場合について説明する。

【０２３８】
実施例８に係るＮＣ装置の制御ソフトウェア実行システムＡ８の全体構成は，実施例２（図９参照）とほぼ同様であるので，その説明を省略する。また，タスクの走行割合を保証する処理については，実施例２（図１１～図１２）とほぼ同様であり，必要なデータ構造も同じであり，タスク開始チックメモ２１０（図１０参照），基準周期を格納しておく領域２１１（図１０参照），タスク走行状態メモ２１２（図１０参照）などを備える。タスク走行状態メモ２１２（図１０参照）は，走行残りチック数，繰り返し回数，走行禁止フラグなどの欄を備える。
【０２３９】
さらに，タスクの走行時間を測定するために，実施例６と同様に，タスク走行時間テーブル２３１（図２２参照）を備える。また，タスク走行時間テーブル２３１（図２２参照）にはタスク毎の合計チック数の欄がある。なお，カウンタ値メモ２３０（図２２参照）はタスク走行状態メモ２１２（図１０参照）と同じ役割を果たすものなので使用しない。また，基準時間割り込み処理の手順，走行時間警告割り込み処理の手順は，実施例２（図１１，図１２参照）と同様なので，その説明を省略する。
【０２４０】
図２７は，一般の割り込みの処理手順について説明したフローチャートであり，Ｏ１は，ある割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する，通常この処理はメインＣＰＵ１によって自動的に実行される。Ｏ２は，未使用のＩＣＢをＩＣＢプール（未使用ＩＣＢを保持しておく場所）から１つ獲得する。
【０２４１】
Ｏ３は，ＩＣＢに，この割り込み処理（自分自身）の，実行環境を格納する。この中には，次に実行する命令のアドレスとしてＯ８の処理の先頭アドレスをＩＣＢに格納することも含まれる。Ｏ４は，ここで，作成したＩＣＢを，スケジューラが実行するＩＣＢ待ち行列に追加する。Ｏ５は，Ｏ１の割り込みの禁止を解除する。これ以降は再び割り込みが発生する可能性がある。以上の処理は割り込み禁止時間をできる限り短くするためのものである。
【０２４２】
Ｏ６は，割り込まれた処理が割り込み処理またはＲＴＯＳか，タスクの処理中かを判定する。Ｏ７は，割り込まれた処理が割り込み処理またはＲＴＯＳを実行中であると判断した場合で，このときは割り込まれた処理に直接戻る。Ｏ８は，割り込まれた処理がタスクの処理を実行中のときで，このときは，まず，割り込まれたタスクのＴＣＢを探し出す。Ｏ９は，Ｏ８において探し出したＴＣＢへ割り込まれたタスクの実行環境を格納する。
【０２４３】
Ｏ１０は，内部カウンタレジスタ２０２（図９参照）の値を，基準周期２１１（図９参照）に保存してある値から減算する。この結果が当該タスクが今回連続して走ったチック数である。これをタスク走行時間テーブル２３１（図２２参照）の当該タスクの合計チック数の欄に加える。Ｏ１１は，現在処理を実行しているタスクが，走行する割合を保証する対象のタスクであるか否かを判定する処理である。具体的には，タスク走行状態メモ２１２（図９参照）の走行残りチック数の欄が１以上のタスクか否かを判定する。
【０２４４】
Ｏ１２は，Ｏ１１において対象のタスクであると判定した場合の処理で，Ｏ１０において求めた値をタスク走行状態メモ２１１の当該タスクの走行残りチック数の欄から減算する。Ｏ１３は，Ｏ１０において求めた値を，リアルタイムクロック２０１（図９参照）の走行時間警告用の内部カウンタレジスタ２０５の値に加える。走行周期を保証したタスクが，走行し終った分だけ，走行周期を保証したタスク以外のタスクを走らせるための時間が増えるわけである。
【０２４５】
Ｏ１４は，作成したＴＣＢを，スケジューラが実行するＴＣＢ待ち行列に，優先順位の順番に追加する。ただし，このとき，タスク走行状態メモ２１２（図１０参照）の走行禁止

フラグの欄が１つのタスクでもオンになっていれば，ＴＣＢ待ち行列において，走行禁止フラグがオフのタスクのＴＣＢは，走行禁止フラグがオンのタスクのＴＣＢよりも前につなぐ。Ｏ１５は，スケジューラへ飛ぶ。

【０２４６】

Ｏ１６は，スケジューラから戻ってくるとここへ来る処理である。スケジューラへ飛んだ後，すぐにこの処理が実行されるとは限らず，他のＩＣＢが処理される場合もあるが，いずれはここへ戻ってくる。ここからが割込毎に固有の本来の割り込み処理であり，割り込み要因によって処理は異なる。Ｏ１７は，割り込み処理の終了であり，通常のサブルーチンのリターン命令が一般的であり，このリターン命令によって，スケジューラのＩＣＢ処理部に戻る。

【０２４７】

図２８は，スケジューラの動作を示すフローチャートであり，本実施例においては，優先順位が最低で，外部に対して何も処理を行なわないアイドルタスクが常に走っているものとする。Ｐ１は，割り込み処理（ＩＣＢ）の待ち行列があるか否かを判定する処理である。Ｐ２は，割り込み処理の待ち行列があったと判断した場合の処理であり，その待ち行列を実行するために最初に割り込みを禁止する。これは，割り込み処理の待ち行列のリストを処理している最中に，割り込みが発生して割り込み処理の待ち行列に追加を実行しようとするとリストの整合性が保たれなくなるためである。

【０２４８】

Ｐ３は，割り込み処理の待ち行列のリストから先頭のＩＣＢを取り除く。Ｐ４は，割り込み禁止を解除する。Ｐ５は，Ｐ３において取り除いたＩＣＢからＯ３（図２７参照）において格納した実行アドレスやレジスタなどの割り込み処理の実行環境を取り出して実行する。これが，Ｏ１６（図２７参照）の処理である。Ｐ６は，Ｏ１７（図２７参照）から戻ってくるところであり，再び割り込みを禁止する。Ｐ７は，Ｐ３において取り除いたＩＣＢをＩＣＢプールへ返す。Ｐ８は，割り込み禁止を解除してからＰ１へ戻る。後は割り込み処理待ち行列がある間はＰ１からＰ７の処理を繰り返す。

【０２４９】

Ｐ９は，ＩＣＢ待ち行列はないと判断した場合の処理であり，タスク処理の待ち行列を実行するために，割り込みを禁止する。Ｐ１０は，割り込み処理の待ち行列のリストから先頭のＴＣＢを獲得する。Ｐ１１は，タスク開始チックメモ２１０（図１０参照）にリアルタイムクロック２０１（図９参照）の走行時間計測用の内部カウンタレジスタ２０２（図９参照）の値を保存しておく。Ｐ１２は，割り込み禁止を解除する。Ｐ１３は，Ｐ１０においてタスク処理の待ち行列（リスト）の先頭から，獲得したＴＣＢからＯ９（図２７参照）において格納したタスクの実行環境を取り出して実行する。このときはその処理へ飛んでいって，ここへは再び戻ってこない。

【０２５０】

以上において説明したように，実施例８に係るＮＣ装置の制御ソフトウェア実行システムＡ８は，タスクの走行割合を設定でき，また，タスクと割り込み処理の実際の走行時間を知ることができる機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。

【０２５１】

実施例８によれば，ＮＣ装置の制御ソフトウェア実行システムＡ８は，ＮＣ装置機能を実現するタスクと，割り込み処理の正確な実行時間を知ることにより，各タスクの走行時間の割り当てなどが容易に行なえるようになり，システムの構築が容易となる。

【０２５２】

〔実施例９〕

次に，実施例９について説明する。実施例９に係るＮＣ装置の制御ソフトウェア実行システムＡ９の全体構成は，実施例２（図９参照）とほぼ同様であるので，その説明を省略する。

【０２５３】

次に，ＮＣ装置の制御ソフトウェア実行システムＡ９の動作について説明する。図２９は，ＮＣ装置の運転モード選択部分の処理手順を示すフローチャートであり，Ｑ１は，ＮＣ装置のモードがグラフィックチェックか否かを判定する部分である。Ｑ２は，グラフィックチェックモードであると判断した場合であり，その場合はグラフィック表示タスクの割り当てを増やし，他の，例えば，自動モード処理タスクの割り当てをその分減らす。

【０２５４】

Ｑ３は，グラフィックチェックモードではないと判断した場合で，自動モードか否かを判定する処理である。Ｑ４は，自動モードであると判断した場合であって，自動モード処理タスクの割り当てを増やし，他の処理タスクの割り当てを減らす。Ｑ５は，手動モードか否かのチェックである。Ｑ６は，手動モードであると判断した場合であって，手動処理タスクの割り当てを増やし，他の処理タスクの割り当てを減らす。それ以降は，その他のモードの場合であるが，その説明は省略する。

【０２５５】

以上において説明したとおり，実施例９に係るＮＣ装置の制御ソフトウェア実行システムＡ９にあっては，ＮＣ装置の運転モードによってタスクの走行割合を変化させて，ＮＣ装置の制御ソフトウェアを実行するものである。

【０２５６】

実施例９によれば，ＮＣ装置の制御ソフトウェア実行システムＡ９は，ＮＣ装置の運転モードによってタスクの走行割合を変化させることにより，モード毎にタスクに適した時間だけタスクを走らせることができ，ＮＣ装置の実行速度が向上する。

【０２５７】

〔実施例１０〕

次に，実施例１０について説明する。実施例１０に係るＮＣ装置の制御ソフトウェア実行システムＡ１０は，同一の加工プログラムを複数回走らせるとき（同じ加工を複数回行なうとき），初回加工時における各タスクの走行状況に基づいて，２回目以降の加工時における各タスクの走行配分を最適化するものである。

【０２５８】

実施例１０に係るＮＣ装置の制御ソフトウェア実行システムＡ１０の全体構成は，実施例２（図９参照）とほぼ同様であるので，その説明を省略するが，以下，説明の都合上，本実施例においては，例えば，実施例８と同様に，タスク毎の走行割合を指定する手段と，実際のタスクの走行状況を記録できる手段とを備えているものとする。

【０２５９】

使用するデータ構造としては，実施例８と同様，タスク開始チックメモ２１０（図１０参照），基準周期を格納しておく領域２１１（図１０参照），タスク走行状態メモ２１２（図１０参照），タスク走行時間テーブル２３１（図２２参照）などを備える。タスク走行状態メモ２１２（図１０参照）は，走行残りチック数，繰り返し回数，走行禁止フラグなどの欄があり，タスク走行時間テーブル２３１（図２２参照）にはタスク毎の合計チック数の欄がある。

【０２６０】

基準時間割り込みの処理手順，走行時間警告割り込みの処理手順は，実施例２（図１１，図１２参照）と同様であり，一般の割り込みの処理手順は，実施例８（図２７参照）と同様なので，その説明を省略する。

【０２６１】

図３０は，複数回加工時における処理の時間を短縮するための，各タスクの走行割り当て時間を最適化するための手順を示すフローチャートであり，Ｒ１は，同一加工の１回目か，２回目以降かを判定する処理である。Ｒ２は，１回目の加工時の処理であると判断した場合で，各タスクの総走行時間（チック数）を計測，記録する。Ｒ３は，走行したタスクの内，アイドルタスクなどの必要のないタスクを除いた，他の必要なタスクの走行時間の合計を計算する。

【０２６２】

Ｒ４は，Ｒ３において求めた合計に基づいて，必要なタスクの走行割合を計算する。走る必要がないタスクの走行割合は０とする。Ｒ５は，Ｒ４において計算した割合をタスク走行状態メモ２１２（図１０参照）の走行割合の欄に登録する。Ｒ６は，２回目以降の加工時における処理であり，Ｒ５で登録したタスク走行状態メモ２１２（図１０参照）の各タスクの走行割合により，各タスクを走行させる。

【０２６３】

以上，説明したとおり，実施例１０に係るＮＣ装置の制御ソフトウェア実行システムＡ１０は，１回目の加工によって，最適なタスクの走行割合を自動的に設定する機能を備え，ＮＣ装置の制御ソフトウェアを実行するものである。

【０２６４】

実施例１０によれば，ＮＣ装置の制御ソフトウェア実行システムＡ１０は，機能を実現する各タスクへの走行時間の割り当てを，自動的に最適なものにすることにより，繰り返し加工における加工時間の短縮を実行できる。

【０２６５】

〔実施例１１〕

次に，実施例１１について説明する。実施例１１に係るＮＣ装置の制御ソフトウェア実行システムＡ１１は，予め登録された，各タスクの標準の走行割合や，全走行時間と，実際に走っているＮＣ装置の制御ソフトウェア実行システムの各タスクのデータが，予め指定した割合以上にずれている場合には，タスクの異常と判定する。

【０２６６】

実施例１１に係るＮＣ装置の制御ソフトウェア実行システムＡ１１の全体構成は，実施例８（図９参照）と同様であるので，その説明を省略する。また，実際に走っているＮＣ装置の制御ソフトウェア実行システムの各タスクのデータを測定するのは，実施例８の場合と同じであるので，その説明を省略する。

【０２６７】

各タスクが正常に走っているか否かを判定するのに必要なデータ構造を，図３１に示す。２５０は，“タスク走行標準データ”で，タスク毎に“走行割合”や“全走行チック数”などの欄がある。これらのデータは，システム生成時に予め作成しておくか，あるいは何らかのＳＶＣによって，そのデータを使うよりも前の，任意の時期に設定しておいてもよい。

【０２６８】

図において，２５１は，この割合までは，タスク走行時間のずれがあっても，タスク異常と見なさない許容度を格納する“走行時間判定許容度”であり，また，２５２は，この範囲内であれば，タスク走行割合に，ずれがあってもタスク異常と見なさない許容度を格納する“走行割合判定許容度”である。この走行割合判定許容度には下限欄と上限欄があり，ずれがこの範囲内ならばタスク異常と見なさない。

【０２６９】

各タスクが正常に走っているか否かを判定するための，タスク走行時間テーブル２３１（図２２参照）とタスク標準データ２５０（図３１参照）との比較は，例えば，タスクが切り替わるときに，スケジューラにより行ってもよいし，あるいは定期的に走るタスク異常判定タスクを設けて，そのタスクにより判定を行なってもよい。

【０２７０】

以下の説明では，タスクが正常に走っているか否かの判定を，異常判定タスクに基づいて行うものとする。また，判定許容度２５１は全走行時間と，走行割合について別々に設けてもよいが，ここでは説明の都合上，同じ値を用いるものとする。

【０２７１】

図３２は，異常判定タスクの動作を示すフローチャートである。Ｓ１は，異常判定タスクの先頭で，タスク走行時間テーブル２３１（図２２参照）の，タスク毎の合計チック数の欄の値，全ての和を求める処理である。Ｓ２は，全てのタスクに対して，以下のＳ３からＳ５までの処理を行なったか否かを判定し，全タスクに対して処理を終了したと判定した

場合には，Ｓ６へ飛ぶ。
【０２７２】
Ｓ３は，タスク走行時間テーブル２３１（図２２参照）の，当該タスクの合計チック数の値と，標準データ２５０（図３１参照）の全走行チック数を比較する。当該タスクのタスク走行時間テーブル２３１（図２２参照）の合計チック数の値が，標準データ２５０（図３１参照）の全走行チック数の値に走行時間判定許容度２５１（図３１参照）の値を乗じたものよりも小さいか，等しければ，このタスクは正常であると見なす。
【０２７３】
Ｓ４は，Ｓ３において，タスクに異常が発生したと判定した場合における処理であり，例えば，オペレータにタスク異常が発生したことを知らせるメッセージを送るなどの，何らかのエラー処理を実行する。Ｓ５は，タスクの走行割合のチェックである。タスク走行時間テーブル２３１（図２２参照）の合計チック数の値を，全タスク分加えて，それで当該タスクの合計チックを割って，当該タスクの走行割合を求める。その値が標準データ２５０（図３１参照）の走行割合判定許容度の下限と上限との間にあれば，このタスクは正常であると見なす。もし，異常があると判定した場合には，Ｓ４に飛ぶ。Ｓ６は，全タスクを判定した結果，異常状態のタスクは見つからなかったので，タスク終了のＳＶＣを発行して，次に呼ばれるまで何もしない。
【０２７４】
以上，説明した通り，実施例１１に係るＮＣ装置の制御ソフトウェア実行システムＡ１１にあっては，ＮＣ装置の機能を実現しているタスクに異常が発生したときにそれを検知する機能を備えたものである。
【０２７５】
実施例１１によれば，ＮＣ装置の制御ソフトウェア実行システムＡ１１は，ＮＣ装置の機能を実現しているタスクに異常が発生したときに，それを検知する機能を備えているため，ＮＣ装置に異常が発生したことを直ちに判定することができる。
【０２７６】
〔実施例１２〕
次に，実施例１２について説明する。実施例１２に係るＮＣ装置の制御ソフトウェア実行システムＡ１２の全体構成は，実施例１１（図９参照）とほぼ同様であるので，その説明を省略する。データ構造についても，実施例１１と同じもの（図２２，図３１参照）を使用するものとする。次に，新たに必要なデータを図３３に示す。図３３において，３６０は，“工具選択使用フラグ”である。このフラグは，ＮＣ装置の制御ソフトウェアのうち，新たな加工を行なう処理部でオフして，工具を選択する処理部でオンする。すなわち，このフラグがオンになっていれば，工具選択処理が呼ばれたことを示す。
【０２７７】
また，３６１は，“工具選択処理スキップフラグ”である。このフラグは，ＮＣ装置の制御ソフトウェアのうち，新たな加工を行なう処理部でオフして，以下に述べる，異常判定タスクで工具を選択したときは，異常判定タスクがオンする。本来の工具選択処理は，工具選択処理スキップフラグがオンになっているときは自分では工具の選択を行なわず，異常判定タスクが選択した工具を用いる。３６２は，本来の工具選択処理と異常判定タスクが選択した工具を伝え合うための“選択工具メモ”である。
【０２７８】
次に，ＮＣ装置の制御ソフトウェア実行システムＡ１２の動作について説明する。図３４は，ＮＣ装置の別工具選択部分の処理手順を示すフローチャートであり，この処理は，実施例１１と同様に異常判定タスクにより実行するものとする。Ｔ１は，異常判定タスクの先頭で，タスク走行時間テーブル２３１（図２２参照）の，タスク毎の合計チック数の欄の値全ての和を求める処理である。Ｔ２は，全てのタスクに対して以下の処理を行なったか否かの判定であり，全てのタスクに対して処理を終了すればＴ６へ飛ぶ。
【０２７９】
Ｔ３は，タスク走行時間テーブル２３１（図２２参照）の，当該タスクの合計チック数の

値と，標準データ２５０（図３１参照）の全走行チック数を比較する。当該タスクのタスク走行時間テーブル２３１（図２２参照）の合計チック数の値が，標準データ２５０（図３１参照）の全走行チック数の値に走行時間判定許容度２５１（図３１参照）の値を乗じたものよりも小さいか，等しければ，このタスクは正常であると見なす。

【０２８０】

Ｔ４は，Ｔ３で，タスクに異常が発生したと判定した場合の処理で，工具選択使用フラグ３６０（図３３参照）がオンになっていなければ，Ｔ５に移行する。Ｔ５は，Ｔ４において，タスクに異常が発生したと判定したときの，通常の処理であり，例えば，オペレータにタスク異常が発生したことを知らせるメッセージを送るなど，何らかのエラー処理を実行する。Ｔ６は，工具選択使用フラグ３６０（図３３参照）がオンになっている場合の処理であり，次工具を選択する。

【０２８１】

次工具の選択とは，例えば，エンドミルならエンドミル，ボールエンドミルならボールエンドミルというように，同じ種類の工具で工具径が異なる工具を選択することをいう。このとき，例えば，加工モードを設けて，加工速度優先なら工具径のより大きな工具を選択し，精度優先なら工具径のより小さな工具を選択する，などしてもよい。

【０２８２】

このとき，エラーを起こした工具は，選択工具メモ３６２（図３３参照）から知ることができる。Ｔ７は，工具が正常に選択できたか否かの判定を行う。Ｔ７において，選択する工具がないと判定した場合には，通常のエラー処理部Ｔ５へと移行する。Ｔ８は，工具が正常に選択できたと判定した場合であって，工具選択使用フラグ３６０（図３３参照）をオフに，工具選択処理スキップフラグ３６１（図３３参照）をオンにして，本来の工具選択処理部に，ここで選択した工具を，選択工具メモ３６２（図３３参照）に設定することによって伝達する。その後，加工プログラム解析タスクを起動するなどにより，再び加工を開始する。

【０２８３】

Ｔ９は，タスクの走行割合のチェックである。タスク走行時間テーブル２３１（図２２参照）の合計チック数の値を，全タスク分加えて，それで当該タスクの合計チックを割って，当該タスクの走行割合を求める。その値が標準データ２５０（図３１参照）の走行割合判定許容度の下限と上限の間にあれば，このタスクは正常であると見なす。もし，異常があると判定した場合には，Ｔ４に移行する。Ｔ１０は，全タスクを判定した結果，異常状態のタスクは見つからなかったので，タスク終了のＳＶＣを発行して，次に呼ばれるまで何もしない。

【０２８４】

以上，説明したとおり，実施例１２に係るＮＣ装置の制御ソフトウェア実行システムＡ１２は，ＮＣ装置の自動加工によって，エラーが発生したと判定したとき，工具を変えて再度自動加工を行なうＮＣ装置の制御ソフトウェアを実行するものである。

【０２８５】

実施例１２によれば，ＮＣ装置の制御ソフトウェア実行システムＡ１２は，ＮＣ装置の自動加工によって，エラーが発生したと判定したとき，工具を変えて再度自動加工を実行することにより，エラーの発生によって加工処理が止まる可能性が低くなり，自動的に連続加工ができ，ＮＣ装置の連続運転の信頼性が高くなる。

【０２８６】

〔実施例１３〕

次に，実施例１３について説明する。実施例１３に係るＮＣ装置の制御ソフトウェア実行システムＡ１３の全体構成は，実施例１１（図９参照）の内容とほぼ同様であるので，その説明を省略する。データ構造についても，実施例１１と同じもの（図２２，図３１参照）を使用するものとする。

【０２８７】

新たに必要なデータを図３５に示す。図３５において，３７０は，“工具速度決定フラグ

”である。このフラグは，ＮＣ装置の制御ソフトウェアのうち，新たな加工を行なう処理部でオフして，工具の速度を計算する処理部でオンする。すなわち，このフラグがオンになっていれば，工具速度計算処理が呼ばれたことを示す。３７１は，“工具速度計算処理スキップフラグ”である。このフラグは，ＮＣ装置の制御ソフトウェアのうち，新たな加工を行なう処理部でオフして，以下に述べる，異常判定タスクにより工具の速度を変更したときは，異常判定タスクがオンする。

【０２８８】

本来の工具速度計算処理は，工具速度計算処理スキップフラグがオンになっているとき，自分自身では速度の計算は実行せず，異常判定タスクが変更した速度を用いる。また，３７２は，本来の工具速度計算処理と異常判定タスクが工具速度を伝え合うための“工具速度メモ”である。３７３は，工具速度変更割合下限値である。この割合を下回っては工具速度は変更できない。３７４は，工具速度変更割合上限値である。この割合を上回っては工具速度は変更できない。工具速度変更割合下限値３７３や工具速度変更割合上限値３７４は予め指定されている。

【０２８９】

次に，ＮＣ装置の制御ソフトウェアの実行動作について説明する。図３６は，ＮＣ装置の別速度計算部分の処理手順を示すフローチャートである。この処理は，実施例１１と同様に異常判定タスクで行なうものとする。図において，Ｕ１は，異常判定タスクの先頭で，タスク走行時間テーブル２３１（図２２参照）の，タスク毎の合計チック数の欄の値，全ての和を求める処理である。Ｕ２は，全てのタスクに対して以下の処理を行なったか否かの判定であり，全てのタスクに対して処理を終了すれば，処理はＵ６へ移行する。

【０２９０】

Ｕ３は，タスク走行時間テーブル２３１（図２２参照）の，当該タスクの合計チック数の値と，標準データ２５０（図３１参照）の全走行チック数を比較する。当該タスクのタスク走行時間テーブル２３１（図２２参照）の合計チック数の値が，標準データ２５０（図３１参照）の全走行チック数の値に走行時間判定許容度２５１（図３１参照）の値を乗じたものよりも小さいか，等しければ，このタスクは正常であると見なす。

【０２９１】

Ｕ４は，Ｕ３において，タスクに異常が発生したと判定した場合における処理であり，工具速度決定フラグ３７０（図３５参照）がオンになっていなければ，Ｕ５に移行する。Ｕ５は，Ｕ４において，タスクに異常が発生したと判定した場合の通常の処理であり，例えば，オペレータにタスク異常が発生したことを知らせるメッセージを送るなどの，何らかのエラー処理を実行する。

【０２９２】

Ｕ６は，工具速度決定フラグ３７０（図３５参照）がオンになっていると判定した場合における処理であり，工具速度を変更する。ここで，速度の変更とは，速度を変えることであるが，このとき，例えば，加工モードを設けて，加工速度優先なら工具速度をある割合ずつ増していき，精度優先なら工具速度をある割合ずつ減らしていくなどしてもよい。このとき，エラーを起こした工具速度は，工具速度メモ３７２（図３５参照）により確認する。

【０２９３】

Ｕ７は，工具速度が決定できたか否かの判定である。Ｕ６において，工具速度が，工具速度変更割合下限値３７３（図３５参照）と工具速度変更割合上限値３７４（図３５参照）との間に入らないときは，通常のエラー処理部Ｕ５へと移行する。Ｕ８は，工具速度を変更できた場合で，工具速度決定フラグ３７０（図３５参照）をオフに，工具速度計算処理スキップフラグ３７１（図３５参照）をオンにして，本来の工具速度計算処理部に，ここで変更した工具速度を，工具速度メモ３７２（図３５参照）に設定することによって伝える。その後，加工プログラム解析タスクを起動するなどにより，再び加工を開始する。

【０２９４】

Ｕ９は，タスクの走行割合のチェックである。タスク走行時間テーブル２３１（図２２参

照）の合計チック数の値を，全タスク分加えて，それで当該タスクの合計チックを割って，当該タスクの走行割合を求める。その値が標準データ２５０（図３１参照）の走行割合判定許容度の下限と上限との間にあれば，このタスクは正常であると見なす。もし，異常があると判定したときにはＵ４へ移行する。Ｕ１０では，全タスクを判定した結果，異常状態のタスクは見つからなかったので，タスク終了のＳＶＣを発行して，次に呼ばれるまで何もしない。

【０２９５】

以上，説明したとおり，実施例１３に係るＮＣ装置の制御ソフトウェア実行システムＡ１３は，ＮＣ装置の自動加工によって，エラーが発生したと判定したとき，工具速度を変えて再度自動加工を行なうＮＣ装置の制御ソフトウェアを実行するものである。

【０２９６】

実施例１３によれば，ＮＣ装置の制御ソフトウェア実行システムＡ１３は，ＮＣ装置の自動加工によって，エラーが発生したと判定したとき，工具速度を変えて再度自動加工を行なうことにより，エラーの発生により加工が止まる可能性が低くなり，自動的に連続加工ができＮＣ装置の連続運転の信頼性が高くなる。

【０２９７】

〔実施例１４〕

次に，実施例１４について説明する。実施例１４に係るＮＣ装置の制御ソフトウェア実行システムＡ１４の全体構成は，実施例１１（図９参照）とほぼ同様であるので，その説明を省略する。

【０２９８】

データ構造についても，実施例１１と同じもの（図２２，図３１参照）を使うものとする。新たに必要なデータを図３７に示す。図３７において，３８０は，“加工プログラム番号メモ”である。この加工プログラム番号メモを見て，加工プログラム解析タスクは，この番号の加工プログラムの解析を実行する。

【０２９９】

また，３８１は，“加工プログラムスケジューリングテーブル”である。ここへ加工を行なう加工プログラムの“番号”と“回数”などを，予め登録しておく。３８２は，“加工不能プログラムメモ”である。加工が失敗した加工プログラムの番号を，異常判定タスクが記録しておく。

【０３００】

次に，ＮＣ装置の制御ソフトウェア実行システムＡ１４の動作について説明する。図３８は，ＮＣ装置の別加工プログラム選択の処理手順を示すフローチャートである。この処理は，実施例１１と同様に異常判定タスクにより実行する。Ｖ１は，異常判定タスクの先頭で，タスク走行時間テーブル２３１（図２２参照）の，タスク毎の合計チック数の欄の値，全ての和を求める処理である。Ｖ２は，全てのタスクに対して以下の処理を行なったか否かの判定であり，全てのタスクに対して処理を終了すれば，処理はＶ１０へ移行する。

【０３０１】

Ｖ３は，タスク走行時間テーブル２３１（図２２参照）の，当該タスクの合計チック数の値と，標準データ２５０（図３１参照）の全走行チック数を比較する。当該タスクのタスク走行時間テーブル２３１（図２２参照）の合計チック数の値が，標準データ２５０（図３１参照）の全走行チック数の値に走行時間判定許容度２５１（図３１参照）の値を乗じたものよりも小さいか，等しければ，Ｖ９において，このタスクは正常であると見なす。

【０３０２】

Ｖ４は，Ｖ３において，タスクに異常が発生したと判定した場合における処理であり，異常が発生した加工プログラム番号を，加工不能プログラムメモ３８２（図３７参照）に記録して，加工プログラムスケジューリングテーブル３８１（図３７参照）から次に加工する加工プログラム番号を選択して，加工プログラム番号メモ３８０（図３７参照）に格納する。

【０３０３】

Ｖ５は，次に加工する加工プログラムが決定できたか否かの判定である。Ｖ４において，加工プログラムスケジューリングテーブル３８１（図３７参照）に登録されている加工プログラムを全て実行し終るとＶ７の処理に移行する。Ｖ６は，次に加工する加工プログラムが決定できた場合であって，加工プログラム解析タスクを起動するなどにより，再び加工を開始する。

【０３０４】

Ｖ７は，次に加工する加工プログラムが決定できなかった場合における処理であり，加工不能プログラムメモ３８２（図３７参照）に，データが設定されているか否かにより，加工できなかったプログラムがあるか否かを判定する。Ｖ８は，加工できなかったプログラムがあったと判定した場合における処理であり，加工不能プログラムメモ３８２（図３７参照）に基づいて，何らかの手段でオペレータに異常が発生したことを通知する。また，Ｖ１０では，全タスクを判定した結果，異常状態のタスクは見つからなかったので，タスク終了のＳＶＣを発行して，次に呼ばれるまで何もしない。

【０３０５】

以上，説明したとおり，実施例１４に係るＮＣ装置の制御ソフトウェア実行システムＡ１４は，ＮＣ装置の自動加工によって，エラーが発生したと判定したとき，別の加工プログラムにより，再度自動加工を行なうＮＣ装置の制御ソフトウェアを実行するものである。

【０３０６】

実施例１４によれば，ＮＣ装置の制御ソフトウェア実行システムＡ１４は，ＮＣ装置の自動加工によって，エラーが発生したと判定したときは，別の加工プログラムで，再度自動加工を行なうことにより，エラーの発生により加工が止まる可能性が低くなり，自動的に連続加工ができＮＣ装置の連続運転の信頼性が高くなる。

【０３０７】

〔実施例１５〕

次に，実施例１５について説明する。実施例１５に係るＮＣ装置の制御ソフトウェア実行システムＡ１５の全体構成は，実施例２（図９参照）と同じであるので，その説明を省略する。ただし，本実施例にあっては，リアルタイムクロック２０１（図９参照）の内部レジスタは，２０２と２０４の２チャンネルのみを使用する。内部カウンタレジスタ２０２（図９参照）は，タスクの走行チック数の計測用，内部カウンタレジスタ２０４（図９参照）は，タスク停止割り込み用として使用する。

【０３０８】

図３９は，本実施例において，必要なデータ構造を示したものであり，４００は“タスク停止テーブル”である。各タスクは，予め，あるいは，そのタスクが開始させられるときにＳＶＣによって，タスクが起動してから停止するまでの，チック数をこのタスク停止テーブル４００の“停止チック数”の欄に登録しておく。また，指定しないタスクはこの値を０としておく。

【０３０９】

２１０（図１０参照）は“タスク開始チックメモ”である。ＲＴＯＳが，タスクを起動する直前に，リアルタイムクロック２０１（図９参照）の内部カウンタレジスタ２０２（図９参照）の値を記録しておく，そして，そのタスクが終了してＲＴＯＳに戻って来たときの，内部カウンタレジスタ２０２（図９参照）との差を求めて，タスク走行時間を求める。４１１は，“タスク停止割り込みメモ”である。タスク停止割り込みをすぐに処理できないときは，このメモをオンしておいて後に処理する。

【０３１０】

本実施例に係るＮＣ装置の制御ソフトウェア実行システムＡ１５は，タスクを起動するたびに，もし指定されていれば，タスク停止テーブル４００（図３９参照）の停止チック数を，リアルタイムクロック２０１（図９参照）の内部カウンタレジスタ２０４に設定する。そして，そのタスク停止割り込みが発生すれば，当該タスクを強制的に停止させる。

【０３１１】

もし，タスク停止割り込みが発生する前に，他のタスクによってメインＣＰＵ１を占有さ

れた場合は，リアルタイムクロック２０１（図９参照）の内部カウンタレジスタは２０２（図９参照）と，タスク開始チックメモ２１０（図１０参照）から，当該タスクの走行チック数を求め，その分をタスク停止テーブル４００（図３９参照）の停止チック数から減ずる。このようにして，指定タスクの最大走行チック数を制限することができる。

【０３１２】

次に，ＮＣ装置の制御ソフトウェア実行システムＡ１５の動作について，詳細に説明する。最初に割り込みの処理手順について説明する。図４０は，タスク停止割り込みの処理手順を示すフローチャートである。Ｗ１は，ある割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する。通常この処理はＣＰＵによって自動的に実行される。Ｗ２は，割り込まれた処理が，タスク処理中か，それ以外かを判定する処理部分である。Ｗ３は，タスク処理中ではないと判定した場合であり，この場合には，タスク停止割り込みメモ４１１（図３９参照）をオンする。Ｗ４は，割り込み禁止を解除して，割り込まれた処理へ戻る。

【０３１３】

また，Ｗ５は，タスク処理中に割り込みが発生したと判定した場合であって，まず，割り込まれたタスクのＴＣＢを探し出す。Ｗ６は，Ｗ５において見つけたＴＣＢへ割り込まれたタスクの実行環境を格納する。Ｗ７は，タスク停止ＳＶＣと同様の当該タスクの終了処理を実行する。例えば，当該ＴＣＢのステータスの欄を停止にして，ＴＣＢ待ち行列から，停止中タスクの待ち行列へつなぎ変える，などの処理である。Ｗ８は，スケジューラへ飛ぶ。

【０３１４】

図４１は，タスク停止割り込み以外の，一般の割り込み処理の手順を示すフローチャートである。Ｘ１は，ある割り込みが発生したときにおける割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する，通常この処理はメインＣＰＵ１によって自動的に実行される。Ｘ２は，未使用のＩＣＢをＩＣＢプール（未使用ＩＣＢを保持しておく場所）から１つ獲得する。

【０３１５】

Ｘ３は，ＩＣＢに，この割り込み処理（自分自身）の，実行環境を格納する。この中には次に実行する命令のアドレスとしてＸ８の処理の先頭アドレスをＩＣＢに格納することも含まれる。Ｘ４は，ここで，作成したＩＣＢを，スケジューラが実行するＩＣＢ待ち行列に追加する。Ｘ５は，Ｘ１の割り込みの禁止を解除する。これ以降は再び割り込みが発生する可能性がある。以上の処理は割り込み禁止時間をできる限り短くするためのものである。

【０３１６】

また，Ｘ６は，割り込まれた処理が，割り込み処理またはＲＴＯＳか，それともタスクの処理中かを判定する。Ｘ７は，割り込まれた処理が，割り込み処理またはＲＴＯＳを実行中のときであって，このときは割り込まれた処理に直接戻る。Ｘ８は，割り込まれた処理がタスクの処理を実行中の場合であって，このときは，まず割り込まれたタスクのＴＣＢを探し出す。Ｘ９は，Ｘ８において見つけたＴＣＢへ，割り込まれたタスクの実行環境を格納する。Ｘ１０は，タスク停止割り込みメモ４１１（図３９参照）がオンか否かを判定する処理である。

【０３１７】

Ｘ１１は，タスク停止割り込みメモ４１１（図３９参照）がオンのときの処理であり，タスク停止ＳＶＣと同様の，当該タスクの終了処理を実行する。例えば，当該ＴＣＢのステータスの欄を停止にして，ＴＣＢ待ち行列から，停止中タスクの待ち行列へつなぎ変える等の処理である。

【０３１８】

Ｘ１２は，タスク停止割り込みメモ４１１（図３９参照）がオフのときの処理であり，リアルタイムクロック２０１（図９参照）の内部カウンタレジスタ２０２（図９参照）と，タスク開始チックメモ２１０（図１０参照）との差から，当該タスクの走行チック数を求

め，タスク停止テーブル４００（図３９参照）の停止チック数から減ずる。Ｘ１３は，Ｘ９において作成したＴＣＢを，スケジューラが実行するＴＣＢ待ち行列にプライオリティの順に追加する。Ｘ１４は，スケジューラへ飛ぶ。

【０３１９】

また，Ｘ１５は，スケジューラから戻ってくるとここへ来る処理である。スケジューラへ飛んだ後すぐにこの処理が行なわれるとは限らず，他のＩＣＢが処理された後かも知れないがいずれはここへ戻ってくる。ここからが割込毎に固有の本来の割り込み処理であり，割り込み要因によって処理は異なる。Ｘ１６は，割り込み処理の終了であり，通常のサブルーチンのリターン命令が一般的であり，このリターン命令によって，スケジューラのＩＣＢ処理部に戻る。

【０３２０】

図４２を用いて，本実施例に係るＮＣ装置の制御ソフトウェア実行システムＡ１５のスケジューラの動作を説明する。以下，説明の都合上，本実施例においては，優先順位が最低で，外部に対して何も処理を行なわない，アイドルタスクが常に走っているものとする。

【０３２１】

Ｙ１では，割り込み処理（ＩＣＢ）の待ち行列があるか否かを判定する。Ｙ２は，割り込み処理の待ち行列があったときの処理であり，その待ち行列を実行するために最初に割り込みを禁止する。これは，割り込み処理の待ち行列のリストを処理している最中に，割り込みが発生して割り込み処理の待ち行列に追加を行なおうとするとリストの整合性が保たれなくなるためである。

【０３２２】

Ｙ３は，割り込み処理の待ち行列のリストから先頭のＩＣＢを取り除く。Ｙ４は，割り込み禁止を解除する。Ｙ５は，Ｙ３において取り除いたＩＣＢからＸ３（図４１参照）において格納した実行アドレスやレジスタなどの割り込み処理の実行環境を取り出して実行する。これが，Ｘ１５（図４１参照）の処理である。Ｙ６は，Ｘ１６（図４１参照）から戻ってくるところであり，再び割り込みを禁止する。Ｙ７は，Ｙ３において取り除いたＩＣＢをＩＣＢプールへ返す。Ｙ８は，割り込み禁止を解除してからＹ１へ戻る。後は割り込み処理待ち行列がある間はＹ１からＹ７の処理を繰り返す。

【０３２３】

Ｙ９は，割り込みを禁止する。Ｙ１０は，割り込み処理の待ち行列のリストから，先頭のＴＣＢを獲得する。Ｙ１１は，当該ＴＣＢのタスクが，タスク停止テーブル４００（図３９参照）の停止チック数が，０以上のタスクである場合は，当該タスクのタスク停止テーブル４００（図３９参照）の停止チック数の値を，リアルタイムクロック２０１（図９参照）の内部カウンタレジスタ２０４に設定する。この結果，設定した数だけのチック数が経過すると，タスク停止割り込みが発生する。Ｙ１２は，割り込み禁止を解除する。Ｙ１３は，Ｙ１０においてタスク処理の待ち行列（リスト）から獲得した，ＴＣＢからＸ９（図４１参照）において格納したタスクの実行環境を取り出して実行する。その処理へ飛んでいって，ここへは戻ってこない。

【０３２４】

以上，説明したように，実施例１５に係るＮＣ装置の制御ソフトウェア実行システムＡ１５は，通常のプライオリティベースのスケジューリング方式しか持たないＲＴＯＳに，タスク走行時間の制限を行なえる機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。

【０３２５】

実施例１５によれば，ＮＣ装置の制御ソフトウェア実行システムＡ１５は，ＮＣ装置機能を実現するタスクの，走行時間による制限を行なえるようになる。そのため，タスクに割り当てた時間を越えて，タスクが走ることがなくなり，システムの構築が容易となる。

【０３２６】

〔実施例１６〕

次に，実施例１６について説明する。実施例１６に係るＮＣ装置の制御ソフトウェア実行

システムＡ１６は，加工プログラム解析処理タスクなどを，従来例のように，１ブロック処理する毎にタスクを抜けるのではなく，加工プログラムが存在する限り，解析を続けるように作成しておき，解析の中断は，時間によるものとする。

【０３２７】

実施例１６に係るＮＣ装置の制御ソフトウェア実行システムＡ１６の加工プログラム解析タスクの実行手順を図４３に基づいて説明する。Ｚ１は，加工プログラム解析タスクの先頭であり，指定された加工プログラムのブロックを全て解析し終えたか否かを判定する。Ｚ２は，全ての加工プログラムのブロックの解析を終えたと判定した場合の処理であり，タスク終了ＳＶＣを発行して，再び起動されるまで何もしない。Ｚ３は，未解析の加工プログラムのブロックが存在する場合の処理であり，ブロックの解析を実行する。以後，全ブロックの解析が終了するまで，Ｚ１の処理を繰り返す。

【０３２８】

以上で述べたように，実施例１６に係るＮＣ装置の制御ソフトウェア実行システムＡ１６は，加工プログラム解析タスクを，プログラムがある限り，解析し続けるようにして，解析の中断を時間によって実行するＮＣ装置の制御ソフトウェアを実行するものである。

【０３２９】

実施例１６によれば，ＮＣ装置の制御ソフトウェア実行システムＡ１６は，加工プログラムの解析を行なうタスクの走行，中断を時間によって行なうようにしたので，加工プログラムのブロックの大きさによる，処理時間の余りが無くなり，特に，微小線分からなる，加工プログラムの解析において処理能力の向上が期待できる。さらにタスクの作成方法として，加工プログラムのブロック毎に，終了するなどの繁雑な処理が不必要になる。

【０３３０】

以上のように，この発明の各実施例に係るＮＣ装置の制御ソフトウェア実行システムによれば，ＮＣ装置の機能を，分担して実現しているタスクの制御をきめ細かく実行できるので，各タスクの処理時間を最適化することができるため，無駄時間を少なくして，全体の処理時間を早めることができる。また，ＮＣ装置の制御ソフトウェア実行システムの異常を，システム自体が判定できる手段を備えているため，異常状態が発生した時の回避処理も容易となる。

【０３３１】

〔実施例１７〕

次に，実施例１７について説明する。図４４は，実施例１７に係る制御ソフトウェア実行システムＡ１７の全体構成を示すブロック図であり，この制御ソフトウェア実行システムＡ１７は，以下の要素より構成されている。１は各部の動作を制御したり，数値制御に必要な演算を実行するメインＣＰＵ，２はシステムの各部を結ぶシステムバス，３はＮＣ装置の主要機能を実現する制御ソフトウェアなどを格納するＲＯＭ（不揮発性記憶手段），４はデータの一時記憶やワークエリアなどに用いるＲＡＭ（揮発性記憶手段），５は外部との間でシリアル通信によりデータのやり取りを行なうＳＩＯインタフェース部である。

【０３３２】

また，図４４において，６はＮＣ装置の運転状態を表示したり，ＮＣ装置に与えた指令を確認するためにデータ表示を行う表示装置，７はＮＣ装置に対して指令を与えるための入力手段としてのキーボード，８はサーボモータを制御するための指令を演算するサーボ制御部であり，該サーボ制御部８はメインＣＰＵ１とは別の専用ＣＰＵを備えてもよい。

【０３３３】

さらに，図４４において，９はサーボ制御部８から受け取った指令を電力増幅してサーボモータや工作機械主軸（図示せず）を駆動するサーボアンプ，１０は工作機械（図示せず）の加工部を制御するためのサーボモータ，１１は工作機械との間でサーボ制御指令以外のデータをやり取りするためのプログラマブルコントローラ部（以下，ＰＣ部という），１２はメインＣＰＵ１に入力される，外部割り込み信号を表している。ここで，外部割り込み信号は電源異常や非常停止などイベント（緊急の出来事）の発生をメインＣＰＵ１に対して通知するための信号である。

【０３３４】
また，図４４において，５００はメインＣＰＵ１に対して入力される，システムクロックである。ここで，システムクロックとは，ＮＣ装置全体を制御するために同期をとるためのクロック信号であり，システム内における各タスクの優先順位を一定間隔毎に調べたり，タスクを指定時間だけ実行を遅らせたりするために，ＣＰＵ１に対してある一定周期でタイマ割り込みを入れる。この周期は少なくともマイクロ秒オーダーの十分粒度の小さいものにする。例えば，１０Ｍヘルツのクロック信号を使用する。この場合は，０．１μ秒（０．０００００００１秒）単位でタスクの制御が実行できる。
【０３３５】
ＲＴＯＳは，システムクロック割り込みがある毎に各タスクの状態を調べ，実行中のタスクを止めて，別のタスクを実行させたり（これをスケジューリング，あるいは，ディスパッチという）する。
【０３３６】
図４４において，５０１はＲＡＭ４内に格納されたシステムクロック数メモである。このシステムクロック数メモ５０１は所定の条件のときに，システムクロック５００の入力がある毎にＲＴＯＳが減らしていく。５０２はＲＡＭ４内に格納されたシステムクロック数メモ有効フラグである。このメモが有効を示す値に設定されている場合のみ前記システムクロック数メモ５０１を有効と見なすものである。
【０３３７】
また，５０８はＲＡＭ４内に格納されたディスパッチクロック数である。ここに設定されたシステムクロックの回数分だけシステムクロックの入力があったときに，ディスパッチなど従来の実施例で行っていたのと同様のＲＴＯＳの処理を実行する。このディスパッチクロック数５０８の設定は，通常電源投入時に１回だけ行えばよいが，システムが稼働している最中に変更することにより，スケジューラの呼び出し頻度を変えることも可能である。
【０３３８】
さらに，５０９はＲＡＭ４内に格納されたディスパッチクロック数メモであり，システムクロック５００の入力がある毎に，この値を減らしていき，０になったときにディスパッチなどの従来の実施例で行ったのと同様のＲＴＯＳの処理を実行する。
【０３３９】
次に，上記の構成を有するＮＣ装置の制御ソフトウェア実行システムＡ１７の動作について説明する。
【０３４０】
各タスクは，予め，あるいは，そのタスクが開始させられるときにＳＶＣ（スーパーバイザーコール，または，サービスコール）によってその実行周期を登録する。ここで，ＳＶＣとは，タスクがＲＴＯＳに対して何らかのサービスを要求するときの総称である。何らかのサービスとは，例えば，何らかの条件が成立するまで自分自身の実行を停止させたり，他のタスクと通信を行なったり，他のタスクを起動させたり（実行を開始させたり），停止させたりすることである。
【０３４１】
このデータは，ＲＴＯＳが，例えば，上記図２に示すような“タスク走行周期テーブル”５０に登録する。図２は，タスク毎に“走行周期”と“繰り返し回数”を有するデータ構造である。ここで，走行周期とは，タスクの次に走るまでのクロック数の値であり，また，繰り返し回数とは，何回そのタスクを起動するかを示す回数である。例えば，この値がマイナスの値ならば∞を示すものとし，その場合にあっては，タスクは繰り返し起動するものとする。このタスク走行周期テーブル５０を用いて，ＲＴＯＳは，タスクに起動がかかるたびに図３に示すような“タスク走行待ちリスト”５１を作成する。
【０３４２】
図３において，タスク走行待ちリスト５１の中の各走行待ちブロック５２は，ぞれぞれ，次の走行待ちブロックの位置を示す“リンク”，“タスク名”，今現在からどれだけの時

間が経過した後に走らなくてはならないかの“走行予定クロック数”，同時に走らせなくてはならないタスク名を示す“同時タスク名”，それにＴＣＢの位置を示す“ＴＣＢポインタ”等の要素から構成されるブロックである。

【０３４３】

以下に，図４５のフローチャートを用いて具体的な動作について説明する。説明の都合上，システムに入力するシステムクロック５００の周波数は１０Ｍヘルツ，すなわち，０．１μ秒毎にシステムクロック５００が入力されるものとする。また，説明中，リストを操作している最中など必要な処理中は，割り込みを禁止しているものとする。さらに，走行待ちブロック５２を作成する場合には，タスク名としては当該タスクを示す名前を入れておくものとする。

【０３４４】

図４５は，ＲＴＯＳが，タスク走行待ちリスト５１を作成するときの手順を示したものである。図において，まず，ＢＢ１では，要求のあった走行周期を内部クロック数に変換する。例えば，要求が１０ｍ秒であれば，１００，０００（１００，０００×０．１μ秒＝１０ｍ秒）と変換する。これをＲＴ（ｒｅｑｕｅｓｔ　ｔｉｍｅ）とする。次に，ＢＢ２では，現在の走行待ちリストに走行待ちブロックがあるか否かを判定し，走行待ちブロックがないと判定した場合には，ＢＢ３に処理が移行し，反対に，走行待ちブロックがあると判定した場合には，ＢＢ６に処理が移行する。

【０３４５】

ＢＢ３は，走行待ちリスト５１に走行待ちブロック５２がなかった場合における処理であり，走行待ちブロック５２を作成し，ＢＢ４では，走行待ちリスト５１の先頭に走行待ちブロック５２を置く。その後，ＢＢ５では，走行待ちリスト５１の先頭ブロックの走行予定クロック数を，システムクロック数メモ５０１へ格納する。ＢＢ２１では，システムクロック数メモ有効フラグ５０２へ有効を示す値を設定する。

【０３４６】

次に，ＢＢ６は，走行待ちリスト５１に走行待ちブロック５２があると判断した場合における処理であり，最終タスク走行時間ＬＴを０にして，タスクインデックスＴＩを１にする。ここで，ＬＴとは当該タスクを走らせるのは現在からどれだけのクロック数かを計算するための変数であり，ＴＩは走行待ちリスト中で当該タスクを示すインデックスである。

【０３４７】

その後，ＢＢ７では，現在のシステムクロック数メモ５０１の値を読み込んで，先頭の走行待ちブロック５２の走行予定クロック数に入れる。次に，ＢＢ８において，走行待ちリスト５１内の，全ての走行待ちブロック５２を調べ終ったか（たどったか）否かを判定する。

【０３４８】

その結果，ＢＢ８において，全ての走行待ちブロック５２を調べ終っていないと判定した場合には，ＢＢ９に処理が移行する。ＢＢ９では，走行待ちブロック５２が，走行待ちリスト５１中にあったと判断した場合の処理であり，走行待ちリスト５１の中から，ＴＩの示す走行待ちブロック５２の，走行予定クロック数を取り出してＬＴに加える。

【０３４９】

その後，ＢＢ１０において，ＲＴとＬＴとを比較する。ＢＢ１１は，ＲＴとＬＴが等しい場合（ＲＴ＝ＬＴ）の処理であり，走行待ちブロック５２を作成する。このとき，走行予定クロック数は０にしておく。ＢＢ１２は，ＢＢ１１において作成した走行待ちブロック５２を，走行待ちリスト５１中のＴＩの示すブロックの同時タスク名へつなぐ。ここで，つなぐとは，例えば，ＢＢ１１において作成したブロックのアドレスをＴＩが示す走行待ちブロック５２の同時タスク名の場所に置くなど，要するにＢＢ１１において作成したブロックを，直ちに見つけられるような配置に設定することである。

【０３５０】

ＢＢ１３は，ＲＴがＬＴより大きな場合（ＲＴ＞ＬＴ）の処理であり，ＴＩの値を，１プ

ラスして，ＢＢ８へ戻り同じ処理を繰り返す。
【０３５１】
ＢＢ１４は，ＲＴがＬＴより小さな場合（ＲＴ＜ＬＴ）の処理であり，走行待ちブロック５２を作成する。このとき，走行予定クロック数は，ＲＴ－ＬＴ＋（ＴＩが示すより１つ前の走行待ちブロック５２の走行予定クロック数）の値にする。
【０３５２】
その後，ＢＢ１５では，ＴＩが示すブロックの走行予定クロック数をＬＴ－ＲＴとし，ＢＢ１６では，ＴＩが示す走行待ちブロック５２とその前のブロックとの間にＢＢ１４において作成した走行待ちブロック５２を挿入する。
【０３５３】
その後，ＢＢ１７では，ＢＢ１６で挿入したブロックが，走行待ちリスト５１の先頭か否かを判定し，ＢＢ１８において，先頭であると判定した場合には，走行待ちリスト５１の先頭ブロックの走行予定クロック数を，システムクロック数メモ５０１に格納する。
【０３５４】
ＢＢ１９では，ＢＢ８において，全ての走行待ちブロックを調べ終ったと判定した場合に，走行予定クロック数がＲＴ－ＬＴの走行待ちブロック５２を作成し，その後，ＢＢ２０では，ＢＢ１９において作成した走行待ちブロック５２を走行待ちリスト５１の最後に挿入する。
【０３５５】
上記実施例における図５は，走行待ちリスト５１の例である。この例では現在から３０クロック後にタスクＡを起動して，さらにそれから２０クロック後（すなわち，現在から５０クロック後）にタスクＢを起動することを示している。このとき，タスクＣを３５クロック後に起動する要求がＲＴＯＳに対して発せられたとすると，図４５に示したフローチャートの手順にしたがって走行待ちリストは図６に示すようになる。
【０３５６】
図４５に示したＢＢ５や，ＢＢ１８において走行待ちリスト５１の先頭ブロックの走行予定クロック数を，システムクロック数メモ５０１に格納し，その指定したクロック数が経過した後，後述するシステムクロック割り込みからタスク起動処理が呼ばれる。このタスク起動処理について図４６を用いて説明する。
【０３５７】
図４６において，ＣＣ１では，走行待ちリスト５１の先頭の走行待ちブロック５２を走行待ちリスト５１から切り離す処理を実行し，その後，ＣＣ２では，走行待ちリスト５１が空になったか否かを判定する。また，ＣＣ３は，走行待ちリスト５１が空になったと判定した場合の処理であり，システムクロック数メモ有効フラグ５０２を無効にすることにより，以降タスク起動処理が呼ばれないようにして，処理を修了する。
【０３５８】
また，ＣＣ４は，ＣＣ２において，走行待ちリスト５１に走行待ちブロック５２が残っていると判定した場合の処理であり，先頭の走行待ちブロック５２の走行予定クロック数を，システムクロック数メモ５０１に設定する。その後，ＣＣ５では，ＣＣ１において切り離した走行待ちブロック５２から起動するタスクを取り出して，そのタスクのＴＣＢをＴＣＢ待ち行列の先頭に配置する。
【０３５９】
その後，ＣＣ６では，今走らせたタスクの，タスク走行周期テーブル５０（図２参照）の周期回数を判定する。ＣＣ７は，走行回数が∞（負）の場合の処理であり，このタスクの走行待ちブロック５２の，走行予定クロック数を，タスク走行周期テーブル５０の走行周期に変え，走行待ちブロックを走行待ちリストに追加する。
【０３６０】
ＣＣ８は，走行回数が正の整数だった場合の処理であり，走行予定回数を１減らす。その後，ＣＣ９では，当該タスクに対して図４に示した処理により，このタスクの走行待ちブロック５２を作成して走行待ちリストに追加する。ＣＣ１０は，走行回数が０の場合の処

理であり，この場合，何の処理も実行せずに処理を終了する。
【０３６１】
次に，システムクロック割り込み処理について，図４７を用いて説明する。ＤＤ１は，システムクロック割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する。ただし，この処理は，通常メインＣＰＵ１により割り込みが発生したときに自動的に実行される。
【０３６２】
次に，ＤＤ２０はディスパッチクロック数メモ５０９から１減じる処理である。ＤＤ１４はシステムクロック数メモ有効フラグ５０２が有効になっているか否かを判定する判定処理であり，有効になっていないと判定した場合には，後述するＤＤ２１の処理へ移行し，反対に，有効になっていると判定した場合には，ＤＤ１７の処理を実行する。
【０３６３】
ＤＤ２１はディスパッチクロック数メモ５０９が０になったか否かを判定するする処理であり，ＤＤ２２はディスパッチクロック数メモ５０９が０になった場合の処理で，改めてディスパッチクロック数５０８の値で初期化する。その後，ＤＤ２からのディスパッチ処理を実行する。
【０３６４】
ＤＤ１５はディスパッチクロック数メモ５０９が０になっていない場合の処理で，割り込み禁止を解除する処理である。ＤＤ１６は割り込まれた処理にスケジューラを介さずに直接戻る処理である。ＤＤ１７は割り込みクロック数メモ５０１から１減ずる処理である。
【０３６５】
ＤＤ１８は割り込みクロック数メモ５０１が０に等しくなったか否かを判定するための処理であり，この結果が０に等しくない場合には，まだタスク起動処理を呼ばないため，ディスパッチを行うか否かを判定するためにＤＤ２１へ移行する。ＤＤ１９は，図４６において説明したタスク起動処理を呼ぶための処理である。タスク起動処理から戻ってくるとＤＤ２３の処理へと移行する。ＤＤ２３の処理はディスパッチクロック数メモ５０９が０になったか否かを判定する処理であり，ＤＤ２４はディスパッチクロック数メモ５０９が０になった場合の処理で，改めてディスパッチクロック数５０８の値で初期化する。
【０３６６】
また，ＤＤ２は，未使用のＩＣＢをＩＣＢプール（未使用ＩＣＢを保持しておく場所）から１つ獲得する。
【０３６７】
その後，ＤＤ３では，ＩＣＢに，自分自身の割り込み処理の，実行環境を格納保存する。この中には，次に実行する命令のアドレスとしてＤＤ１２の処理における先頭アドレスをＩＣＢに格納することも含まれる。また，ＤＤ４では，今作成したＩＣＢを，スケジューラが実行するＩＣＢ待ち行列に追加する。その後，ＤＤ５は，ＤＤ１の割り込みの禁止を解除する。これ以降は，再び割り込みが発生する可能性がある。以上の処理は割り込み禁止時間をできる限り短くするためのものである。
【０３６８】
次に，ＤＤ６は，割り込まれた処理が，割り込み処理あるいはＲＴＯＳか，それともタスクの処理中かを判定する。その結果，ＤＤ７は，割り込まれた処理が割り込み処理またはＲＴＯＳを実行中であると判断した場合であって，このときは割り込まれた処理に直接戻る。反対に，ＤＤ８は，割り込まれた処理が，タスクの処理を実行中であると判断した場合であって，このときは，まず割り込まれたタスクのＴＣＢを探し出す。その後，ＤＤ９では，ＤＤ８において見つけたＴＣＢへ割り込まれたタスクの実行環境を格納保存する。
【０３６９】
さらに，ＤＤ１０では，作成したＴＣＢを，スケジューラが実行するＴＣＢ待ち行列に優先順位の順番に追加する。また，ＤＤ１１では，スケジューラへ飛び，スケジューラからＤＤ１２に処理が戻ってくるが，スケジューラへ飛んだ後すぐにこの処理が実行されるとは限らず，他のＩＣＢが処理された後に戻る場合もある。ここからが割込毎に固有の本来

の割り込み処理であり，割り込み要因によって処理が異なる。さらに，ＤＤ１３は，割り込み処理の終了であり，通常のサブルーチンへのリターン命令が一般的であり，このリターン命令によって，スケジューラＩＣＢ処理に戻る。

【０３７０】

以上の説明の中では，また以後の割り込みの説明において，従来例のように，割り込み自身のアドレスをリストに登録しておいて，多重割り込み（割り込み処理中の他の割り込み）を許可する処理は，従来例と同様であるので，その説明は省略する。

【０３７１】

上記のとおり，実施例１７に係るＮＣ装置の制御ソフトウェア実行システムＡ１７は，通常のプライオリティベースのスケジューリング方式しか持たないＲＴＯＳのシステムクロックを細かい時間粒度による制御を可能にし，なおかつ，毎回のシステムクロック割り込みによるスケジューラ呼び出しを必要なときだけに制限して，ＲＴＯＳにオーバーヘッドもなく，タスク起動の制御が実行される機能を追加し，ＮＣ装置の制御ソフトウェアを実行するものである。

【０３７２】

実施例１７によれば，ＮＣ装置の制御ソフトウェア実行システムＡ１７は，ＮＣ装置機能を実現するタスクの時間による制御を小さな時間単位で行なえるようになり，さらにタスクを起動する周期を簡単に指定できる。そのため，タスクに割り当てた時間の無駄を少なくでき，さらに，タスクをその走行時間で制御できるので，システムの構築が容易になると共にＮＣ装置の処理速度が向上する。

【０３７３】

〔実施例１８〕

次に，実施例１８の構成について説明する。実施例１８に係るＮＣ装置の制御ソフトウェア実行システムＡ１８の全体構成を示す図１８の内容は，実施例１７において示した図４４とほぼ同様である。ただし，実施例１７にあっては，システムクロック数メモ５０１しかなかったが，本実施例では他に，積算システムクロック数メモ５０３と走行時間警告用システムクロック数メモ５０５を備えていることが異なる。

【０３７４】

以下，実施例１８に係るＮＣ装置の制御ソフトウェア実行システムＡ１８の構成を詳細に説明する。なお，この制御ソフトウェア実行システムＡ１８の１～１２，５００，５０８，５０９は，上記実施例１７の図４４に示したものと同様であるので，その説明を省略する。

【０３７５】

図４８において，５０１はＲＡＭ４内に格納された基準時間割り込み用のシステムクロック数メモである。このシステムクロック数メモ５０１はシステムクロック５００の入力がある毎にＲＴＯＳが減らしていく。５０２はＲＡＭ４内に格納されたシステムクロック数メモ有効フラグであり，システムクロック数メモ５０１が有効か否かを示す。５０３はＲＡＭ４内に格納されたタスク走行時間計測用の積算システムクロック数メモであり，システムクロック５００の入力がある毎にＲＴＯＳが増加させる。５０５はＲＡＭ４内に格納された走行時間警告用に用いる，走行時間警告用システムクロック数メモである。

【０３７６】

次に，ＮＣ装置の制御ソフトウェア実行システムＡ１８の動作について説明する。図４９は，本実施例において必要なデータ構造を示したもので，５１０は“タスク開始システムクロックメモ”であり，ＲＴＯＳが，タスクを起動する直前に図４８に示した積算システムクロック数メモ５０３の値を記録しておき，その後，タスクが終了してＲＴＯＳに戻ってきたときの積算システムクロック数メモ５０３との差を求めることによって，タスクの走行時間を求めるためのものである。

【０３７７】

また，５１１は“基準周期”を格納しておく領域である。システムに対して予め１つの基準周期を登録しておく。この周期毎に各タスクはどれくらい走行するかを指定する。この

指定は時間でもクロック数でも構わないが，ここでは説明の都合上，クロック数に変換しておくものとする。

【０３７８】

さらに，５１２は“タスク走行状態メモ”である。各タスクは予め，あるいは，そのタスクが開始させられるときにＳＶＣによって，その実行の割合を登録する。例えば，タスク毎の指定走行周期，すなわち，タスクＡが１０％，タスクＢが５％，タスクＣが２０％，基準周期が２０ｍ秒のとき，２０ｍ秒毎にタスクＡが２ｍ秒，タスクＢが１ｍ秒，タスクｃが４ｍ秒走ればよい。

【０３７９】

ここで，実施例１７と同様，図４８におけるシステムクロック５００は，１０Ｍヘルツとする。すなわち，基準となる最小単位時間は０．１μ秒とすると，Ａ，Ｂ，Ｃ各タスクはクロック数２０００００クロックの間に２００００，１０００，４００００クロック分走ればよいことになる。このクロック数をタスク走行状態メモ５１２の“走行クロック数”の欄に格納する。ここで，走行する割合を指定しないタスクに対してはこの欄は０にしておく。

【０３８０】

また，“走行残りクロック数”の欄は，１回の基準周期が終了する毎に，走行クロック数の欄をコピーし，タスクが走ったクロック数だけ減算していく。１回の基準周期が終る毎に，すべてのタスクの，この欄の内容が０になっていればよい。当然走行クロック数の合計は基準周期よりも少なくてはいけない。そのエラーチェックは，例えば，タスク状態メモに新たに登録する毎に行なえばよい。ただし，以下の説明において，このエラーチェックについては省略する。

【０３８１】

次に，“繰り返し回数”の欄に設定するのは，何回そのタスクを起動するかを示す回数である。例えば，この値がマイナスの値ならば∞を示すとする。その場合には，当該タスクは繰り返し起動するものとする。

【０３８２】

また，“走行禁止フラグ”の欄は，スケジューラが用いる。このフラグがオンになっているタスクはスケジューリングの対象から外す。すなわち，このフラグがオンのタスクはプライオリティが高くても走ることができない。

【０３８３】

次に，基準周期リセット処理について説明する。制御ソフトウェア実行システムＡ１８は，稼働し始めたときなど走行割合を保証するタスクが動作し始める前に，予めシステムクロック数メモ５０１に，基準周期５１１の値を設定しておく。その指定したクロック数が経過した後，後述するシステムクロック割り込みから基準周期処理が呼ばれる。

【０３８４】

図５０は，基準周期リセット処理の手順について説明したフローチャートである。基準周期リセット処理は，基準周期毎のシステムおよび全タスクの再初期化処理と考えられる。ＥＥ２では，タスク走行状態メモ５１２の走行残りクロック数の合計を求める。ＥＥ３では，その合計が０以上か否かを判定し，０以上ではないと判定した場合には，ＥＥ４において何らかのエラー処理を行なうが，通常は発生しないのでここでは，その説明を省略する。

【０３８５】

反対に，ＥＥ３において，合計が０以上であると判定した場合には，ＥＥ５においてタスク走行状態メモ５１２の走行クロック数の欄の値を走行残りクロック数の欄に，タスク毎にコピーする。その後，ＥＥ６では，再びシステムクロック数メモ５０１に対して基準周期５１１の値を設定する。

【０３８６】

その後，ＥＥ７では，タスク走行状態メモ５１２のすべてのタスクの走行禁止フラグを，オフにする処理を行う。この処理により，再びすべてのタスクがスケジューリング対象に

なる。さらに，ＥＥ８では，基準周期５１１からタスク走行状態メモ５１２の走行残りクロック数の合計を減算した値を走行時間警告用システムクロック数メモ５０５に設定する。その結果，毎回の基準周期に残り時間が周期を保証したタスクだけを走らせる時間しかなくなったときに走行時間警告処理が呼ばれる。この後，呼び出された処理に戻る。

【０３８７】

図５１は，走行時間警告処理の手順について説明したフローチャートである。ＦＦ２は，タスク走行状態メモ５１２の走行残りクロック数が，０または負の全てのタスクの走行禁止フラグをオンにする。

【０３８８】

ＦＦ３では，全ての走行禁止フラグがオフであるＴＣＢが，全ての行禁止フラグがオンのＴＣＢの前に来るように，タスク処理の待ち行列を作り直す。すなわち，走行禁止フラグがオンのタスクの方がフラグがオフのタスクよりも前に走るようにする。

【０３８９】

図５２は，システムクロック割り込み処理の手順を示すフローチャートである。以下の説明において，有効フラグはシステムクロック数メモ有効フラグ５０２に関するものしか説明していないが，走行時間警告用システムクロック数メモ５０５に関しても同様の有効フラグが必要である。本フローチャートの中では，走行時間警告用システムクロック数メモ５０５用の有効フラグに対する処理は，システムクロック数メモ有効フラグ５０２と同様であるので，その説明は省略する。

【０３９０】

ＧＧ１は，ある割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する，通常この処理はメインＣＰＵ１によって自動的に実行される。ＧＧ２０は積算システムクロック数メモ５０３に１を加える処理である。ＧＧ２１はｓｋｉｐフラグをオフにする処理である。ここで，ｓｋｉｐフラグとは，ＲＡＭ４上に設けた今回のシステムクロック割り込みでスケジューリングなどの処理を実行するか否かを示すために用いるフラグである。

【０３９１】

ＧＧ２２はシステムクロック数メモ５０１から１減ずる処理である。ＧＧ２３はＧＧ２２の処理の結果システムクロック数メモ５０１が０になったか否かを判定する処理である。ＧＧ２４はシステムクロック数メモ５０１が０になった場合の処理で，図５０に示した基準周期リセット処理を呼ぶ。ＧＧ２５はｓｋｉｐフラグをオフにする処理である。ＧＧ２６は走行時間警告用システムクロック数メモ５０５から１減ずる処理である。

【０３９２】

また，ＧＧ２７はＧＧ２６の処理の結果，走行時間警告用システムクロック数メモ５０５が０になったか否かを判定する処理である。ＧＧ２８は走行時間警告用システムクロック数メモ５０５が０になった場合の処理であって図５１に示した走行時間警告処理を呼ぶ。ＧＧ２９はｓｋｉｐフラグをオフにする処理である。ＧＧ３２はディスパッチクロック数メモ５０９から１減算する処理である。ＧＧ３３はＧＧ３２の結果，０か否かを判定する処理である。

【０３９３】

さらに，ＧＧ３４はディスパッチクロック数メモ５０９をディスパッチクロック数５０８で再初期化する処理である。ＧＧ３５はｓｋｉｐフラグをオフにする処理である。ＧＧ３０はｓｋｉｐフラグがオンかオフかを判定する処理であり，オフの場合には，スケジューリング処理等を実行するため，処理はＧＧ２に移行する。ＧＧ３１はｓｋｉｐフラグがオンの場合の処理であり，割り込まれた処理に直接復帰するものである。

【０３９４】

ＧＧ２は，未使用のＩＣＢをＩＣＢプール（未使用ＩＣＢを保持しておく場所）から１つ獲得する。ＧＧ３は，ＩＣＢに，この割り込み処理（自分自身の）の，実行環境を格納（保存）する。この中には，次に実行する命令のアドレスとして後述するＧＧ１５の処理の先頭アドレスをＩＣＢに格納することも含まれる。

【０３９５】
ＧＧ４は，ここで，作成したＩＣＢを，スケジューラが実行するＩＣＢ待ち行列に追加する。ＧＧ５は，ＧＧ１の割り込みの禁止を解除する。これ以降は，再び割り込みが発生する可能性がある。以上の処理は割り込み禁止時間をできる限り短くするためのものである。
【０３９６】
ＧＧ６は，割り込まれた処理が，割り込み処理か，あるいは，ＲＴＯＳか，それともタスクの処理中かを判定する。ＧＧ７は，割り込まれた処理が割り込み処理またはＲＴＯＳを実行中の場合で，このときは割り込まれた処理に直接戻る。ＧＧ８は，割り込まれた処理がタスクの処理を実行中のときで，このときにあっては，まず，割り込まれたタスクのＴＣＢを探し出す。ＧＧ９は，ＧＧ８において見つけたＴＣＢへ割り込まれたタスクの実行環境を格納する。
【０３９７】
ＧＧ１０は，現在処理を行なっているタスクが，走行する割合を保証する対象のタスクであるか否かを判定する処理である。具体的には，タスク走行状態メモ５１２の走行残りクロック数の欄が１以上のタスクか否かを判定する。ＧＧ１１は，ＧＧ１０において対象と判定した場合における処理であり，当該タスクを起動したときの積算システムクロック数メモ５０３の値から，記録してあるタスク開始システムクロックメモ５１０の値を減じた当該タスクの走行クロック数を，タスク走行状態メモ５１２の当該タスクの走行残りクロック数の欄から減算する。
【０３９８】
ＧＧ１２は，ＧＧ１１において減算した値（当該タスクの走行クロック数）を，走行時間警告用システムクロック数メモ５０５の値に加える。走行周期を保証したタスクが，走行し終った分だけ，走行周期を保証したタスク以外のタスクを走らせるための時間が増えるわけである。
【０３９９】
ＧＧ１３は，作成したＴＣＢを，スケジューラが実行するＴＣＢ待ち行列に追加する。このとき，タスク走行状態メモ５１２の走行禁止フラグの欄が１つのタスクでもオンになっていれば，ＴＣＢ待ち行列において，走行禁止フラグがオフのタスクのＴＣＢは，走行禁止フラグがオンのタスクのＴＣＢよりも前につなぐ。
【０４００】
ＧＧ１４は，スケジューラへ飛ぶ。ＧＧ１５は，スケジューラから戻ってきたときの処理であり，スケジューラへ飛んだ後すぐにこの処理が実行されるとは限らず，他のＩＣＢが処理された後の場合もあるが，いずれはここへ戻ってくる。ここからが割込毎に固有の本来の割り込み処理であり，割り込み要因によって処理内容がは異なる。ＧＧ１６は，割り込み処理の終了であり，通常のサブルーチンのリターン命令が一般的であり，このリターン命令によって，スケジューラのＩＣＢ処理部に戻る。
【０４０１】
次に，ＲＴＯＳのスケジューラの動作を図５３を用いて説明する。以下，説明の都合上，本実施例においては，優先順位が最低であって，外部に対して何も処理を行なわない，アイドルタスクが常に走っているものとして説明を行なう。
【０４０２】
ＨＨ１は，割り込み処理（ＩＣＢ）の待ち行列があるか否かを調べる処理である。ＨＨ２は，ＨＨ１において，割り込み処理の待ち行列があったと判断した場合の処理であり，その待ち行列を実行するために最初に割り込みを禁止する。これは，割り込み処理の待ち行列のリストを処理している最中に，割り込みが発生して割り込み処理の待ち行列に追加をしようとするとリストの整合性が保たれなくなるためである。
【０４０３】
ＨＨ３は，割り込み処理の待ち行列のリストから先頭のＩＣＢを取り除く。ＨＨ４は，割り込み禁止を解除する。ＨＨ５は，ＨＨ３において取り除いたＩＣＢからＧＧ３において

格納した実行アドレスやレジスタなどの割り込み処理の実行環境を取り出して，それを実行する。これが，ＧＧ１５の処理である。ＨＨ６は，ＧＧ１６から戻ってくるところであり，再び割り込みを禁止する。ＨＨ７は，ＨＨ３において取り除いたＩＣＢをＩＣＢプールへ返す。ＨＨ８は，割り込み禁止を解除してからＨＨ１へ戻る。後は割り込み処理待ち行列がある間はＨＨ１からＨＨ７の処理を繰り返す。

【０４０４】

ＨＨ９は，タスク処理の待ち行列があったときの処理で，その待ち行列を実行するために最初に割り込みを禁止する。ＨＨ１０は割り込み処理待ちの行列のリストから先頭のＴＣＢを取り除く。ＨＨ１１は当該ＴＣＢのタスクが走行割合を保証するタスクである場合には，タスクを開始し，システムクロック数メモ５０１に積算システムクロック数メモ５０３の値を保存しておく。

【０４０５】

ＨＨ１２は割り込み禁止を解除する。ＨＨ１３は，ＨＨ１１でタスク処理の待ち行列（リスト）から取り除いたＴＣＢからＧＧ９で格納したタスクの実行環境を取り出して実行する。そのときは，その処理へ飛んでいってここへは再び戻ってこない。

【０４０６】

以上で述べたように，実施例１８に係るＮＣ装置の制御ソフトウェア実行システムＡ１８は，通常のプライオリティベースのスケジューリング方式しか持たないＲＴＯＳに，タスク走行時間の保証を与える制御を実行できる機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。

【０４０７】

上記実施例１８によれば，ＮＣ装置の制御ソフトウェア実行システムＡ１８は，ＮＣ装置機能を実現するタスクの走行時間による制御を実行することができる。そのため，タスクに割り当てた時間の無駄を少なくできる。さらに，タスクをその走行時間において制御できるので，システムの構築が容易になる。

【０４０８】

〔実施例１９〕

次に，実施例１９の構成について説明する。実施例１９に係るＮＣ装置の制御ソフトウェア実行システムＡ１９の全体構成を示す図５４の内容にあっては，実施例１８において示した図４８の内容とほぼ同様である。以下，構成について詳細に説明する。この制御ソフトウェア実行システムＡ１９の１～１２，５００，５０８，５０９は，上記実施例１７の図４４に示した構成と同様であるので，その説明を省略する。

【０４０９】

この制御ソフトウェア実行システムＡ１９は，次の構成要素がさらに加えられている。すなわち，タスクの走行を修了させるために用いるタスク連続走行終了クロック数メモ５０６と，上記タスク連続走行終了クロック数メモ有効フラグ５０７である。

【０４１０】

次に，実施例１９の動作について説明する。図５５，図５６は，タスクの連続走行を保証するのに必要なデータ構造を示したもので，図５５は，“タスク終了クロック数”のリスト構造である。５２０は，“タスク終了クロックブロック”を表している。タスク終了ブロックの構成要素は，タスク終了ブロックを連結するための“リンク（ポインタ）”，“タスク名”，“終了クロック数”などである。今までの説明と同様本実施例の説明でも内部的には全てクロック数により時間の制御を行なうものとする。

【０４１１】

図５６は，“タスク連続走行保証テーブル”５２１の内容を示している。このテーブルには“タスク名”と“連続走行保証クロック数”と“実行中フラグ”の欄があり，連続走行保証クロック数の値はシステム起動時，またはＳＶＣにより設定する。連続走行を指定しないタスクに対してはこの欄は０にしておく。実行中フラグ欄は現在連続走行を保証されたタスクが走っているときはオン，それ以外のときはオフとする。このフラグは同時には，２つ以上オンすることは無い。

【０４１２】
このテーブルはタスク起動のＳＶＣにおいて，そのパラメータとして指定することにすれば必要はなくなるが，以下の説明の都合上，ここではこのテーブルを用いることにする。
【０４１３】
ＲＴＯＳのスケジューラは，連続走行が保証されているタスクが走行しているときに，割り込みが発生しても割り込み処理の終了後の再スケジューリングは行わないことによって，タスクの連続走行を保証する。連続走行が保証されているタスクの実行を終了させるのは，所定の条件が成立したとき，例えば，“連続走行終了処理”を呼んだときである。
【０４１４】
ＲＴＯＳはタスク起動のＳＶＣが発行されたときは，まずそのタスクが連続走行を保証されたタスクか否かを判定して，もし該当するタスクなら，タスク連続走行保証テーブル５２１の連続走行クロック数をタスク連続走行終了クロック数メモ５０６に代入する。システムクロックが入力される毎にこの値から１ずつ減じていって，０になったときにタスク連続走行終了処理を呼ぶ。
【０４１５】
最初に，タスク連続走行終了処理について説明する。図５７は，タスク連続走行終了処理について示したフローチャートである。ここで，ＩＩ２では現在走っている連続走行を保証されたタスクのＴＣＢを探し出して，現在のタスクの実行環境をＴＣＢへ保存する。ＩＩ３ではタスク連続走行保証テーブル５２１内の，当該タスクの実行中フラグをオフする処理である。ＩＩ４では同様のタスク処理の待ち行列を優先順位に応じて作り直す。
【０４１６】
次に，システムクロック割り込み処理の手順について図５８のフローチャートを用いて説明する。まず，ＪＪ１では，割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する，通常この処理はＣＰＵによって自動的に行われる。ＪＪ２０は，タスク連続走行終了クロック数メモ有効フラグ５０７が有効か否かの判定処理である。
【０４１７】
ＪＪ２１は，タスク連続走行終了クロック数メモ有効フラグ５０７が有効でなかった場合の処理で，ディスパッチクロック数メモ５０９から１を減算する。ＪＪ２２は，上記ディスパッチクロック数メモ５０９が０になったか否かを判定する処理である。ＪＪ２３は，ディスパッチクロック数メモ５０９の値をディスパッチクロック数５０８で再初期化する処理である。ＪＪ２４は，割り込み禁止の解除処理である。ＪＪ２５は，割り込まれた処理に直接復帰する処理である。
【０４１８】
また，ＪＪ２６は，タスク連続走行終了クロック数メモ５０６から１減算する処理である。ＪＪ２７は，ＪＪ２６の結果を判定する処理であり，０でなければＪＪ２１へ飛ぶ。ＪＪ２８は，図５７において説明したタスク連続走行終了処理を呼ぶ処理である。
【０４１９】
ＪＪ２９は，ディスパッチクロック数メモ５０９から１を減算する。ＪＪ３０は，上記ディスパッチクロック数メモ５０９が０になったか否かを判定する処理である。この結果が，０でない場合は，ＪＪ２へ飛ぶ。ＪＪ３１は，ディスパッチクロック数メモ５０９の値をディスパッチクロック数５０８で再初期化する処理である。この後ＪＪ２へ飛ぶ。
【０４２０】
ＪＪ２は，未使用のＩＣＢをＩＣＢプール（未使用ＩＣＢを保持しておく場所）から１つ獲得する。ＪＪ３は，ＩＣＢに，この割り込み処理（自分自身の）の，実行環境を格納する。この中には次に実行する命令のアドレスとしてＪＪ１３の処理の先頭アドレスをＩＣＢに格納することも含まれる。
【０４２１】
ＪＪ４は，今作成したＩＣＢを，スケジューラが実行するＩＣＢ待ち行列に追加する。ＪＪ５は，ＪＪ１の割り込みの禁止を解除する。これ以降は再び割り込みが発生する可能性

がある。以上の処理は割り込み禁止時間をできる限り短くするためである。

【０４２２】

ＪＪ６は，割り込まれた処理が割り込み処理またはＲＴＯＳか，それともタスクの処理中かを判定する。ＪＪ７は，割り込まれた処理が割り込み処理またはＲＴＯＳを実行中のときで，このときは割り込まれた処理に直接戻る。ＪＪ８は，割り込まれた処理がタスクの処理を実行中のときで，このときはまず，割り込まれたタスクのＴＣＢを探し出す。

【０４２３】

ＪＪ９は，今処理を行っているタスクが，連続走行を保証する対象のタスクであるか否かを判定する処理である。具体的には，タスク連続走行保証テーブル５２１（図５６参照）の連続走行保証クロック数の欄が１以上のタスクか否かを判定する。連続走行保証の対象タスクの場合は，以下のＪＪ１０，ＪＪ１１の処理は行わないで，ＪＪ１２へ飛ぶ。ＪＪ１０は，連続走行保証の対象外のタスクの場合の処理で，ＪＪ８で見つけたＴＣＢへ割り込まれたタスクの実行環境を格納する。

【０４２４】

ＪＪ１１は，ＴＣＢ待ち行列を優先順位に応じて作り直す。ＪＪ１２は，スケジューラへ飛ぶ。ＪＪ１３は，スケジューラから戻ってくるとここへ来る処理である。スケジューラへ飛んだ後すぐにこの処理が行われるとは限らず，他のＩＣＢが処理された後かも知れないがいずれはここへ戻ってくる。

【０４２５】

ここからが割込毎に固有の本来の割り込み処理であり，割り込み要因によって処理は異なる。ＪＪ１４は，割り込み処理の終了であり，通常のサブルーチンのリターン命令が多い，このリターン命令によって，スケジューラのＩＣＢ処理部に戻る。

【０４２６】

以上で述べたように，実施例１９に係るＮＣ装置の制御ソフトウェア実行システムＡ１９は，通常のプライオリティーベースのスケジューリング方式しか持たないＲＴＯＳに，タスクの連続した走行時間の保証を与える制御を実行できる機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。

【０４２７】

上記実施例１９によれば，ＮＣ装置の制御ソフトウェア実行システムＡ１９は，ＮＣ装置機能を実現するタスクに連続した走行時間を保証できるようになる。そのため，タスクに割り当てた時間の無駄を少なくできる。さらに，タスクをその走行時間において制御できるので，システムの構築が容易に行える。

【０４２８】

〔実施例２０〕

次に，実施例２０の構成について説明する。実施例２０に係るＮＣ装置の制御ソフトウェア実行システムＡ２０の全体構成を示す図５９の内容は，実施例１７においてし示した図４４の内容とほぼ同様である。

【０４２９】

図５９に示したＲＡＭ４において，５０１はシステムクロック数メモ，５０３は積算システムクロック数メモ，５０５はタスクの走行時間警告用の走行時間警告用システムクロック数メモ，５０６はタスク連続走行終了クロック数メモである。なお，５０８は，図４４と同様にシステムクロック何回に１回ディスパッチ処理を呼ぶかを示すディスパッチクロック数であり，５０９はディスパッチクロック数メモである。

【０４３０】

次に，ＮＣ装置の制御ネットワーク実行システムＡ２０の動作について説明する。実施例１７と同様に，各タスクは，予め，あるいはそのタスクが開始させられるときにＳＶＣによってその実行周期を登録する。

【０４３１】

このデータは，ＲＴＯＳが，例えば，図２に示したような“タスク走行周期テーブル”５０に登録する。図２は，タスク毎に“走行周期”と“繰り返し回数”を持ったデータ構造

である。走行周期はタスクの次に走るまでのクロック数の値であり，繰り返し回数は何回そのタスクを起動するかを示す回数である。例えば，この値がマイナスの値ならば∞を示すとする。その場合はずっとそのタスクは繰り返し起動するものとする。他のタスク走行周期テーブル５０を用いて，ＲＴＯＳは，タスクに起動かかかるたびに図３のような“タスク走行待ちリスト”５１を作成する。

【０４３２】

図３において，リストの中における走行待ちブロック５２は，それぞれ“タスク名”と，今現在からどれだけの時間の経過後に走らなくてはならないかの“走行予定クロック数”，それに同時に走らせなくてはならないタスク名を示す“同時タスク名”から成るブロックである。

【０４３３】

このタスク走行待ちリストを，ＲＴＯＳが作成する方法は，実施例１７において図４５で示したものと同様なので，その説明は省略する。

【０４３４】

また，実施例１８において示したように，図４９に示したは，タスクの走行割合を保証するのに，必要なデータ構造を示したもので，５１０は“タスク開始システムクロックメモ”である。ＲＴＯＳが，タスクを起動する直前に，積算システムクロックメモ５０３の値を記録しておく，そしてそのタスクが終了してＲＴＯＳに戻って来たときの積算システムクロックメモ５０３との差を求めることによって，タスクが走った時間を求める。

【０４３５】

また，図４９に示した“基準周期”５１１，“タスク走行状態メモ”５１２“走行残りクロック数”の欄，“繰り返し回数”の欄，および“走行禁止フラグ”の欄については，実施例１８と同様なので，その説明を省略する。

【０４３６】

図４５に示したＢＢ４や，ＢＢ１６において，走行待ちリストの先頭ブロックの走行予定クロック数を，システムクロック数メモ５０１に格納した。その指定したクロック数が経過した後，ＲＴＯＳはタスク起動処理を呼ぶ。

【０４３７】

図５９に示した，制御ソフトウェア実行システムＡ２０は，稼働し始めたときなど走行割合を保証するタスクが動き始める前に，予めシステムクロック数メモに対して，基準周期５１１の値を設定しておく。この設定したクロック数が経過した後に，基準周期処理をＲＴＯＳが呼ぶ。基準周期処理の手順については，実施例１８の図５０に示した手順と同様なので，その説明を省略する。

【０４３８】

また，走行時間警告処理の手順についても，実施例１８の図５１に示した手順と同様なので，その説明を省略する。

【０４３９】

さらに，システムクロック割り込み処理の手順についても，実施例１８の図５２に示した手順と同様なので，その説明を省略する。

【０４４０】

また，ＲＴＯＳのスケジューラの動作についても，実施例１８の図５３に示した手順と同様なので，その説明を省略する。

【０４４１】

ここで，タスク起動処理，基準時間処理，走行時間警告処理，そして，システムクロックの割り込みの処理について，ＴＣＢ待ち行列への操作を中心にまとめる。

【０４４２】

タスク起動処理は，この処理によって起動するタスクのＴＣＢを，ＴＣＢ待ち行列の先頭に挿入する。また，基準時間処理は，全てのタスクを走行可能とする。走行時間警告処理は，走行時間の割合を保証しているタスクのＴＣＢを，そうでないタスクのＴＣＢより前に持っていると共に，タスク走行状態メモ５１２の走行禁止フラグをオンすることによっ

て，走行時間警告処理が発生したことを，システムクロック割り込みのディスパッチ処理に対して知らせる。

【０４４３】

システムクロック割り込みのディスパッチ処理は，走行時間警告処理によって，走行時間警告処理が発生した場合には，基準時間処理が発生するまでは，走行時間を保証しているタスクのＴＣＢをそうでないタスクのＴＣＢよりＴＣＢ待ち行列の前につなぐ。

【０４４４】

ここで，タスク起動処理とシステムクロックのディスパッチ処理の優先順位はどちらかを選択しておく。タスク起動処理を，優先すればタスクの起動時間は正確になるが，タスクの走行割合は多少不正確になる。一方，ディスパッチ処理を優先すれば，タスクの走行割合は正確になるが，タスクの起動時間は多少不正確になる。これらは，何らかの手段によって閾値を決めておいてもよい。例えば，タスクの起動時間の遅れが，ある時間までは走行割合を優先するがそれ以上は，タスク起動を優先するなどの方法を取ればよい。

【０４４５】

以上で述べたように，実施例２０に係るＮＣ装置の制御ソフトウェア実行システムＡ２０は，通常のプライオリティーベースのスケジューリング方式しか持たないＲＴＯＳに，タスク起動周期とタスク走行割合の保証を与える制御を行える機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。

【０４４６】

上記実施例２０によれば，ＮＣ装置の制御ソフトウェア実行システムＡ２０は，ＮＣ装置機能を実現するタスクの正確な起動時間と走行時間による制御を行えるようになる。そのため，タスクに割り当てた時間の無駄を少なくできる。さらに，タスクをその走行時間で制御できるので，システムの構築が容易に行える。

【０４４７】

〔実施例２１〕

次に，実施例２１について説明する。実施例２１に係るＮＣ装置の制御ソフトウェア実行システムＡ２１の全体構成は，実施例１７に示したもの（図４４参照）と同様であるので，その説明を省略する。

【０４４８】

次に，実施例２１に係るＮＣ装置の制御ソフトウェア実行システムＡ２１の動作について説明する。実施例２０と同様に，各タスクは予め，あるいはそのタスクが開始させられるときにＳＶＣによってその実行周期を登録する。

【０４４９】

このデータは，ＲＴＯＳが，例えば図２に示したようなタスク走行周期テーブル５０に登録する。このタスク走行周期テーブル５０における“走行周期”と“繰り返し回数”については，実施例１７と同様なので，その説明を省略する。

【０４５０】

図３において，リストの中の各走行待ちブロックは，それぞれ“タスク名”と，現在からどれだけの時間の経過後に走らなくてはならないのか“走行予定クロック数”，それに同時に走らせなくてはならないタスク名を示す“同時タスク名”から成るブロックである。

【０４５１】

このタスク走行待ちリストを，ＲＴＯＳが作成する方法は，実施例１７において図４５で示したものと同様なので，その説明は省略する。

【０４５２】

また，実施例１９と同様に，図５５，図５６は，タスクの連続走行を走行割合を保証するのに，必要なデータ構造を示したもので，図５５は，“タスク終了クロック数”のリスト構造である。５２０は，“タスク終了クロックブロック”を表している。タスク終了ブロックの構成要素はタスク終了ブロックを連結するためのポインター，“タスク名”，“終了クロック数”などである。今までの説明と同様本実施例の説明でも内部的には全てクロック数で時間の制御を行うものとする。“実行中フラグ”は現在連続走行を保証をされた

タスクが走っているときはオン，それ以外のときはオフとする。このフラグは同時には，２つ以上オンすることは無い。

【０４５３】

図５６は，“タスク連続走行保証テーブル”５２１である。このテーブルには“タスク名”と“連続走行保証クロック数”の欄があり，連続走行保証クロック数の値はシステム起動時，またはＳＶＣにより設定する。連続走行を指定しないタスクに対してはこの欄は０にしておく。このテーブルはタスク起動のＳＶＣにおいて，そのパラメータとして指定することにすれば，必要はなくなるが，以下の説明の都合上，ここではこのテーブルを用いることにする。

【０４５４】

ＲＴＯＳのスケジューラは，連続走行が保証されているタスクが走行しているときに，割り込みが発生しても割り込み処理の終了後の再スケジューリングは行わないことによってタスクの連続走行を保証する。連続走行が保証されているタスクの実行を終了させるのは“連続走行終了割り込み”が発生したときである。

【０４５５】

ＲＴＯＳはタスク起動のＳＶＣが，発行されたときは，まずそのタスクが連続走行を保証されたタスクか否かを判定して，もし該当タスクなら，タスク連続走行保証テーブル（図５６参照）から連続走行クロック数をシステムクロック数メモに設定する。設定したクロック数が経過すると，“連続走行終了処理”ＲＴＯＳから呼ばれる。

【０４５６】

連続走行終了処理の手順については，実施例１９の図５７に基づいて説明したものと同様なので，その説明を省略する。

【０４５７】

基準周期割り込み以外の，一般の割り込み処理の手順についても，実施例１９の図５８に基づいて説明したのと同様なので，その説明を省略する。

【０４５８】

スケジューラも，実施例２と同様に，同じスケジューラ（図１４参照）と同じであるので，その説明を省略する。

【０４５９】

ここで，連続走行処理と，システムクロック割り込みのディスパッチ処理について，ＴＣＢ待ち行列への操作を中心にまとめる。すなわち，

▲１▼　第１に，タスク起動処理は，この処理によって起動するタスクのＴＣＢを，ＴＣＢ待ち行列の先頭に挿入する。

▲２▼　第２に，連続走行終了処理は，実行中のタスクの実行中フラグをオフして他のタスクの走行を許可する。

▲３▼　第３に，システムクロック割り込みのディスパッチ処理は，連続走行を保証するタスクの場合は，ＴＣＢ待ち行列のディスパッチを行わない。

【０４６０】

ここで，▲１▼と▲３▼の優先順位はどちらかを選択しておく。▲１▼を優先すればタスクの起動時間は正確になるが，タスクの連続走行時間は多少不正確になる。一方，▲３▼を優先すれば，タスクの連続走行時間は正確になるが，タスクの起動時間は多少不正確になる。これらは，何らかの手段によって閾値を決めておいてもよい。例えば，タスクの起動時間の遅れが，ある時間までは連続走行を優先するがそれ以上は，タスク起動を優先するなどの方法を取ればよい。

【０４６１】

以上で述べたように，実施例２１に係るＮＣ装置の制御ソフトウェア実行システムＡ２１は，通常のプライオリティーベースのスケジューリング方式しか持たないＲＴＯＳに，タスク起動周期とタスクの連続走行の保証を与える制御を行える機能を追加した方式で，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。

【０４６２】

上記実施例２１によれば，ＮＣ装置の制御ソフトウェア実行システムＡ２１は，ＮＣ装置機能を実現するタスクの正確な起動時間と連続走行時間による制御実行することができる。そのため，タスクに割り当てた時間の無駄を少なくできる。さらに，タスクをその走行時間で制御できるので，システムの構築が容易に行える。

【０４６３】

〔実施例２２〕

次に，実施例２２について説明する。図６０は，実施例２２に係るＮＣ装置の制御ソフトウェア実行システムＡ２２の全体構成を示し，その内容は，実施例１７とほぼ同様である。

【０４６４】

次に，ＮＣ装置の制御ソフトウェア実行システムＡ２２の動作について説明する。本実施例に係る制御ソフトウェア実行システムＡ２２のＲＴＯＳは，予め，例えば，システムを稼働したときなどに，積算システムクロック数メモ５０３を０に初期化しておく。これらの値は，システムクロック割り込みの毎にシステムクロック割り込み処理の中で１加算されている。

【０４６５】

また，タスクにはいる直前に，そのクロック数メモの値を読み取って記憶しておく。また，そのタスクを抜けた直後に，再びクロック数メモの値を読み取って記憶しておいた値との差を求める。その値が，今回このタスクが走ったクロック数である。そしてタスク毎に，この値の合計を記憶しておく場所を設けておいて，そこに加えていくことにより，各タスクの走行時間の合計を記録する。

【０４６６】

以下，本実施例に係るＮＣ装置の制御ソフトウェア実行システムＡ２２の動作について，さらに詳細に説明する。図６１は，各タスクの走行時間を記録するのに必要なデータ構造であり，ここで，５３０はカウンタ値メモ，５３１はタスク走行時間テーブルであり，図に示す通り，タスク毎の合計クロック数の欄がある。

【０４６７】

図６２は，システムクロック割り込み処理の処理手順を示すフローチャートであり，図６３は，スケジューラの処理手順を示すフローチャートである。

【０４６８】

図６２は，システムクロック割り込み処理の手順について説明したフローチャートであり，ＫＫ１は割り込みが発生したときの割り込み処理の先頭であり，最初に他の割り込みの発生を禁止する，通常この処理はメインＣＰＵ１によって自動的に行われる。ＫＫ１６はシステムクロック数メモ５０１に１を加算する処理である。ＫＫ１７はシステムクロック数メモ５０１に１を加算する処理である。

【０４６９】

ＫＫ１８はディスパッチクロック数メモ５０９から１減算する処理であり，ＫＫ１９はその結果を０と比較する。ＫＫ２０はディスパッチクロック数メモ５０９が０になったときの処理で，ディスパッチクロック数５０８で再初期化したのち，ディスパッチを行うためＫＫ２の処理へ進む。ＫＫ１９の判定結果が０でない場合は直接割り込まれた処理に戻るため，ＫＫ７の処理へ移行する。

【０４７０】

ＫＫ２は未使用のＩＣＢをＩＣＢプール（未使用ＩＣＢを保持しておく場所）から１つ獲得する。ＫＫ３はＩＣＢに，この割り込み処理（自分自身）の，実行環境を格納する。この中には次に実行する命令のアドレスとしてＫＫ８の処理の先頭アドレスをＩＣＢに格納することも含まれる。ＫＫ４はここで作成したＩＣＢを，スケジューラが実行するＩＣＢ待ち行列に追加する。ＫＫ５はＫＫ１の割り込みの禁止を解除する。これ以降は再び割り込みが発生する可能性がある。以上の処理は割り込み禁止時間をできる限り短くするためである。

【０４７１】

ＫＫ６は割り込まれた処理が割り込み処理またはＲＴＯＳか，それともタスクの処理中かを判定する。ＫＫ７は割り込まれた処理が割り込み処理またはＲＴＯＳを実行中のときで，このときは割り込まれた処理に直接戻る。ＫＫ８は割り込まれた処理がタスクの処理を実行中のときで，このときはまず，割り込まれたタスクのＴＣＢを探し出す。ＫＫ９は，ＫＫ８において見つけたＴＣＢへ割り込まれたタスクの実行環境を格納する。

【０４７２】

ＫＫ１０は，積算システムクロック数メモ５０３の値から，カウンタ値メモ５３０の値を減算する。この結果，当該タスクがこの回に連続して走ったクロック数である。ＫＫ１１は，ＫＫ１０で求めた値をタスク走行時間テーブル５３１の当該タスクの合計クロック数の欄に加える。ＫＫ１２は，作成したＴＣＢを，スケジューラが実行するＴＣＢ待ち行列に優先順位の順番に追加する。

【０４７３】

ＫＫ１３はスケジューラへ飛ぶ。ＫＫ１４はスケジューラから戻ってくるとここへ来る処理である。スケジューラへ飛んだ後すぐにこの処理が行われるとは限らず，他のＩＣＢが処理された後かも知れないがいずれはここへ戻ってくる。ここからが割込毎に固有の本来の割り込み処理であり，割り込み要因によって処理は異なる。ＫＫ１５は割り込み処理の終了であり，通常のサブルーチンのリターン命令が多い，このリターン命令によって，スケジューラのＩＣＢ処理ブロックに戻る。

【０４７４】

図６３は，本実施例におけるＲＴＯＳのスケジューラの動作を示したフローチャートである。説明の都合上，本実施例においては，優先順位が最低で，外部に対して何も処理を行わないアイドルタスクが常に走っているものとする。すなわち，アイドルタスクは，常に走っているものとして説明を行う。

【０４７５】

ＬＬ１は割り込み処理（ＩＣＢ）の待ち行列があるか否かを調べる処理である。ＬＬ２は割り込み処理の待ち行列が合ったときの処理でその待ち行列を実行するために最初に割り込みを禁止する。これは，割り込み処理の待ち行列のリストを処理している最中に，割り込みが発生して割り込み処理の待ち行列に追加を行おうとするとリストの整合性が保たれなくなるためである。

【０４７６】

ＬＬ３は割り込み処理の待ち行列のリストから先頭のＩＣＢを取り除く。ＬＬ４は割り込み禁止を解除する。ＬＬ５は，ＬＬ３で取り除いたＩＣＢからＫＫ３（図６２参照）で格納した実行アドレスやレジスタなどの割り込み処理の実行環境を取り出して実行する。これがＫＫ１４（図６２参照）の処理である。

【０４７７】

ＬＬ６は，ＫＫ１５（図６２参照）から戻っているところであり，再び割り込みを禁止する。ＬＬ７は，ＬＬ３で取り除いたＩＣＢをＩＣＢプールへ返す。ＬＬ８は割り込み禁止を解除してからＬＬ１へ戻る。後は割り込み処理待ち行列がある間はＬＬ１からＬＬ７の処理を繰り返す。ＬＬ９はタスク処理の待ち行列があったときの処理でその待ち行列を実行するために最初に割り込みを禁止する。

【０４７８】

ＬＬ１０は割り込み処理の待ち行列のリストから先頭のＴＣＢを取り除く。ＬＬ１１はカウンタ値メモ５３０にシステムクロック数メモ５０１の値を保存しておく。ＬＬ１２は割り込み禁止を解除する。ＬＬ１３はＬＬ１１でタスク処理の待ち行列（リスト）から取り除いたＴＣＢからＫＫ９（図６２参照）で格納したタスクの実行環境を取り出して実行する。このときはその処理へ飛んでいってここへは再び戻ってこない。

【０４７９】

以上のようにして，任意の時間，本実施例に係るシステムを走らせた後，タスク走行時間テーブル５３１を調べれば，各タスクが実際に走った（処理を行った）時間を知ることができる。

【０４８０】
なお，タスクの走行時間のみではなく，タスク走行時間テーブル５３１に走行回数の欄を設けて，タスクの走行回数も記録するようにしておけば，以下に述べるタスクの走行周期，走行割合，走行時間を決めるときにも役立つ。
【０４８１】
さらに，カウンタ値メモ５３０だけでなく，タスク終了カウンタ値メモも設けて割り込み処理の先頭で，システムクロック数メモの値をそこへ保存しておき，スケジューラのＫＫ１０（図６２参照）の部分において，カウンタ値メモ５３０からタスク終了カウンタ値メモを減算した値を，タスクが連続して走行したクロック数とすれば割り込み処理の時間も除いたタスクが純粋に走行した時間が求められる。
【０４８２】
以上で述べたように，実施例２２に係るＮＣ装置の制御ソフトウェア実行システムＡ２２は，タスクの実際の走行時間を知ることができる機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。
【０４８３】
実施例２２によれば，ＮＣ装置の制御ソフトウェア実行システムＡ２２は，ＮＣ装置機能を実現するタスクの正確な実行時間を知ることにより，各タスクの走行時間の割り当てなどが，容易に行えるようになり，システムの構築が容易に行える。
【０４８４】
〔実施例２３〕
次に，実施例２３について説明する。実施例２３に係るＮＣ装置の制御ソフトウェア実行システムＡ２３の全体構成は，実施例２２と同様であるので，その説明を省略する。
【０４８５】
次に，本実施例に係る，ＮＣ装置の制御ネットワーク実行システムＡ２３の動作について説明する。本実施例では，制御ソフトウェア実行システムＡ２３（図６０参照）のＲＴＯＳは，予め，例えば，システムを稼働したとき等に，システムクロック数メモ５０１を０で初期化しておく。
【０４８６】
また，タスクにはいる直前に，そのカウンタ値を読み取って記憶しておく。そのタスクを抜けた直後に，再びカウンタ値を読み取って記憶しておいた値との差を求める。その値が，今回このタスクが走ったクロック数である。そして，タスク毎に，この値の和を記憶しておく場所を設けておいて，そこに加えていくことにより，各タスクの走行時間の合計を記録する。
【０４８７】
さらに，従来例におけるスケジューラのＩＣＢの処理を，スケジューラでなく，優先順位がもっとも高いタスクで実行するものである。
【０４８８】
以下，本実施例に係るＮＣ装置の制御ソフトウェア実行システムについて，さらに詳細に説明する。図６４は，スケジューラの処理手順を示すフローチャートであり，図６５は，ＩＣＢを処理する最高優先順位を持った，割り込み処理実行タスクの処理を示すフローチャートである。
【０４８９】
データ構造としては，従来例と同じＩＣＢ（インタラプトコントロールブロック）を用いる。割り込み処理に関しては，従来例と同じであるので，その説明を省略する。
【０４９０】
図６４は，本実施例に係るＲＴＯＳのスケジューラの動作を説明したものであり，アイドルタスクを設けることにより，常に少なくともアイドルタスクは，実行を待っているものとする。ＭＭ１はタスク処理の待ち行列（ＴＣＢ）があったときの処理でその待ち行列を実行するために最初に割り込みを禁止する。ＭＭ２は割り込み処理の待ち行列のリストから先頭のＴＣＢを取り除く。

【０４９１】
ＭＭ３はカウンタ値メモ５３０にシステムクロック数メモ５０１の値を保存しておく。ＭＭ４は割り込み禁止を解除する。ＭＭ５はＭＭ３でタスク処理の待ち行列（リスト）から取り除いたＴＣＢから割り込み処理で格納したタスクの実行環境を取り出して実行する。このときはその処理へ飛んでいってここへは再び戻ってこない。
【０４９２】
本実施例における割り込み処理タスクの動作は，上記図２６と同様である。すなわち，Ｎ１は，割り込み処理（ＩＣＢ）の待ち行列があるか否かを判定する処理である。Ｎ２は，割り込み処理の待ち行列があったと判断した場合の処理であって，その待ち行列を実行するために最初に割り込みを禁止する。これは，割り込み処理の待ち行列のリストを処理している最中に，割り込みが発生して割り込み処理の待ち行列に追加を行なおうとするとリストの整合性が保たれなくなるためである。
【０４９３】
また，Ｎ３は，割り込み処理の待ち行列のリストから，先頭のＩＣＢを獲得する。Ｎ４は，割り込み禁止を解除する。Ｎ５は，Ｎ３で取り除いたＩＣＢから，割り込み処理で格納した実行アドレスやレジスタなどの割り込み処理の実行環境を取り出して実行する。Ｎ６は，Ｎ３で取り除いたＩＣＢをＩＣＢプールへ返却する。その後は，Ｎ１へ戻り，割り込み処理待ち行列がある間はＮ１からＮ６の処理を繰り返す。Ｎ７は，割り込み処理（ＩＣＢ）の待ち行列が無くなったときの処理でタスク終了のＳＶＣを発行して他のタスクにメインＣＰＵ１の使用権を譲り渡す。
【０４９４】
以上のように，任意の時間，本実施例を走らせた後，タスク走行時間テーブル５３１を調べれば，各タスクが実際に走った（処理を行なった）時間を知ることができる。
【０４９５】
さらに，割り込み処理もタスクレベルで実行することにより，割り込み処理実行タスクの走行時間が，割り込み処理の時間と考えることができるため，割り込み処理の処理時間も測定することができる。また，割り込み処理実行タスク内で，割り込み要因により別の走行時間を記録できるようにしておけば，割り込み要因毎の処理時間を測定することができる。
【０４９６】
以上に説明した通り，実施例２３に係るＮＣ装置の制御ソフトウェア実行システムＡ２３は，タスクと割り込み処理の実際の走行時間を知ることができる機能を追加した方式であり，その方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。
【０４９７】
上記実施例２３によれば，ＮＣ装置の制御ソフトウェア実行システムは，ＮＣ装置機能を実現するタスクと割り込み処理の正確な実行時間を知ることにより，各タスクの走行時間の割り当てなどが容易に行えるようになり，システムの構築が容易に行える。
【０４９８】
〔実施例２４〕
次に，実施例２４について説明する。実施例２４に係るＮＣ装置の制御ソフトウェア実行システムＡ２４の全体構成は，実施例１８と同様であるので，その説明を省略する。
【０４９９】
次に，本実施例に係るＮＣ装置の制御ソフトウェア実行システムＡ２４の動作について説明する。タスクや割り込み処理の走行時間を計測する手段を備え，さらにタスクの走行時間を変える何らかの手段を備えるＮＣ装置において，それを実現するためのＮＣ装置の制御ソフトウェア実行システムの全体構成図は，タスクの走行時間や割合の指定の仕方によって複数の構成があるが，この実施例では，説明の都合上タスクの走行割合を指定できる場合について説明する。
【０５００】
タスクの走行割合を保証する処理については，実施例１８（図５０～図５１参照）とほぼ

同様であり，必要なデータ構造も同じであり，タスク開始システムクロックメモ５１０（図４９参照），基準周期５１１を格納しておく領域（図４９参照），タスク走行状態メモ５１２（図４９参照）などを備える。タスク走行状態メモ５１２（図４９参照）は，“走行残りクロック数”，“繰り返し回数”，“走行禁止フラグ”などの欄を備える。

【０５０１】

さらに，タスクの走行時間を測定するために，実施例２２と同様に，タスク走行時間テーブル５３１（図６１参照）を備える。また，タスク走行時間テーブル５３１（図６１参照）にはタスク毎の合計クロック数の欄がある。なお，カウンタ値メモ５３０（図６１参照）はタスク走行状態メモ５１２（図４９参照）と同じ役目をするので使用しない。

【０５０２】

また，基準時間処理の手順，走行時間警告処理の手順は，実施例１８（図５０，図５１参照）と同様なので，その説明を省略する。

【０５０３】

図６５は，システムクロック割り込み処理の手順について説明した図である。以下の説明において，システムクロック数メモ５０１および走行時間警告用システムクロック数メモ５０５に関しては，実施例１８のシステムクロック割り込み処理（図４７参照）と同様に，それぞれに有効フラグを設けてその有効フラグが“有効”である場合のみシステムクロック数メモ５０１，走行時間警告用システムクロック数メモ５０５に対する処理を行えばよいが，その説明が煩雑になるため本実施例の説明に係るフローチャートではその処理の説明を省略する。

【０５０４】

ＯＯ１は割り込みが発生したときの割り込み処理の先頭であり，最初に，他の割り込みの発生を禁止する，通常この処理はＣＰＵによって自動的に行われる。ＯＯ２０はディスパッチクロック数メモ５０９，システムクロック数メモ５０１，走行時間警告用システムクロック数メモ５０５のそれぞれから１減算する処理であり，ＯＯ２１は積算システムクロック数メモ５０３に１加える処理である。

【０５０５】

ＯＯ２２はｓｋｉｐフラグをオンする処理である。ここで，ｓｋｉｐフラグとは，ＲＡＭ４上に設けた，このシステムクロック割り込みでスケジューリングなどの処理を行うか否かを示すために用いるフラグである。ＯＯ２３は走行時間警告用システムクロック数メモが０になったか否かの判定であり，０になった場合は，ＯＯ２４のタクス起動処理（図４６参照）を呼ぶ処理へ移行する。その後，ＯＯ２５でｓｋｉｐフラグをオフにする。ＯＯ２３の判定において，走行時間警告用システムクロック数メモ５０５が０でない場合にはＯＯ２６の処理へ移行する。

【０５０６】

ＯＯ２６はシステムクロック数メモ５０１が０になったか否かの判定を行う処理であり，その結果，０であれば，ＯＯ２７で基準周期処理（図１２参照）を呼び，ＯＯ２８でｓｋｉｐフラグをオフにする。

【０５０７】

ＯＯ２６の判定結果が０の場合には，ＯＯ２９ではディスパッチクロック数メモ５０９が０か否かの判定を行う。その結果が０であれば，ＯＯ３０でｓｋｉｐフラグをオフにして，ＯＯ３１でディスパッチクロック数５０８の値をディスパッチクロック数メモ５０９へ代入することにより，ディスパッチクロック数メモ５０９の再初期化を実行する。

【０５０８】

ＯＯ３２はｓｋｉｐフラグがオンかオフかの判定を行う処理で，オンの場合は，ＯＯ３３で割り込み禁止の解除を行った後，ＯＯ７へ飛び割り込まれた処理に直接復帰する。オフの場合にはＯＯ２へ飛び，通常のディスパッチ処理を実行する。

【０５０９】

ＯＯ２は，未使用のＩＣＢをＩＣＢプール（未使用ＩＣＢを保持しておく場所）から１つ獲得する。ＯＯ３は，ＩＣＢに，この割り込み処理（自分自身の）の，実行環境を格納す

る。この中には次に実行する命令のアドレスとしてＯＯ８の処理を先頭アドレスをＩＣＢに格納することも含まれる。

【０５１０】

ＯＯ４は，ここで作成したＩＣＢを，スケジューラが実行するＩＣＢ待ち行列に追加する。ＯＯ５はＯＯ１の割り込みの禁止を解除する。これ以降は，再び割り込みが発生する可能性がある。以上の処理は割り込み禁止時間をできる限り短くするためである。ＯＯ６は割り込まれた処理が割り込み処理またはＲＴＯＳか，それともタスクの処理中かを判定する。

【０５１１】

ＯＯ７は割り込まれた処理が割り込み処理またはＲＴＯＳを実行中のときで，このときは割り込まれた処理に直接戻る。ＯＯ８は割り込まれた処理がタクスの処理を実行中のときで，このときはまず，割り込まれたタスクのＴＣＢを探し出し，ＯＯ９で当該ＴＣＢへ割り込まれたタスクの実行環境を格納する。

【０５１２】

ＯＯ１０は積算システムクロック数メモ５０３の値から，基準周期５１１に保存してある値を減算する。この結果が当該タスクが今回連続して走ったクロック数である。これをタスク走行時間テーブル５３１の当該タスクの合計クロック数の欄に加える。

【０５１３】

ＯＯ１１は，ここで処理を行っているタスクが，走行する割合を保証する対象のタスクであるか否かを判定する処理である。具体的には，タスク走行状態メモ５１２の走行残りクロック数の欄が１以上のタスクか否かを判定する。ＯＯ１２は，ＯＯ１１で対象と判定した場合の処理で，ＯＯ１０で求めた値をタスク走行状態メモ５１２の当該タスクの走行残りクロック数の欄から減算する。

【０５１４】

ＯＯ１３は，ＯＯ１０で求めた値を走行時間警告用の走行時間警告用システムクロック数メモ５０５に加える。走行周期を保証したタスクが，走行し終わった分だけ，走行周期を保証したタスク以外のタスクを走らせるための時間が増えるわけである。

【０５１５】

ＯＯ１４は作成したＴＣＢを，スケジューラが実行するＴＣＢ待ち行列に，優先順位の順番に追加する。ただし，このとき，タスク走行状態メモ５１２（図４９）の走行禁止フラグの欄が１つのタスクでもオンになっていれば，ＴＣＢ待ち行列において，走行禁止フラグがオフのタスクのＴＣＢは，走行禁止フラグがオンのタスクのＴＣＢよりも前につなぐ。

【０５１６】

ＯＯ１５はスケジューラへ飛ぶ。ＯＯ１６はスケジューラから戻ってくるとここへ来る処理である。スケジューラへ飛んだ後すぐにこの処理が行われるとは限らず，他のＩＣＢが処理された後かも知れないがいずれはここへ戻ってくる。ここからが割込毎に固有の本来の割り込み処理であり，割り込み要因によって処理は異なる。ＯＯ１７は割り込み処理の終了であり，通常のサブルーチンのリターン命令が多い，このリターン命令によってスケジューラのＩＣＢ処理部に戻る。

【０５１７】

図６６は，スケジューラの動作を示すフローチャートである。本実施例においては，優先順位が最低で，外部に対して何も処理を行わないアイドルタスクが常に走っているものとする。

【０５１８】

図６６において，ＰＰ１は割り込み処理（ＩＣＢ）の待ち行列が有るか否かを調べる処理であり，ＰＰ２は割り込み処理の待ち行列があったときの処理でその待ち行列を実行するために最初に割り込みを禁止する。これは，割り込み処理の待ち行列のリストを処理している最中に，割り込みが発生して割り込み処理の待ち行列に追加を行おうとするとリストの整合性が保たれなくなくためである。

【０５１９】
ＰＰ３は割り込み処理の待ち行列のリストから先頭のＩＣＢを取り除く。ＰＰ４は割り込み禁止を解除する。ＰＰ５はＰＰ３で取り除いたＩＣＢからＯＯ３（図６５参照）で格納した実行アドレスやレジスターなどの割り込み処理の実行環境を取り出して実行する。
【０５２０】
ＰＰ６はＯＯ１７（図６５参照）から戻ってくるところであり，再び割り込みを禁止する。ＰＰ７はＰＰ３で取り除いたＩＣＢをＩＣＢプールへ返す。ＰＰ８は割り込み禁止を解除してからＰＰ１へ戻る。後は割り込み処理待ち行列がある間はＰＰ１からＰＰ７の処理を繰り返す。
【０５２１】
ＰＰ９はタスク処理の待ち行列を実行するために，割り込みを禁止する。ＰＰ１０は割り込み処理の待ち行列のリストから先頭のＴＣＢを獲得する。ＰＰ１１はタスク開始システムクロックメモ５１０にタスク走行時間計測用の積算システムクロック数メモ５０３の値を保存しておく。
【０５２２】
ＰＰ１２は割り込み禁止を解除する。ＰＰ１３はＰＰ１０でタスク処理の待ち行列（リスト）の先頭から，獲得したＴＣＢからＯＯ９（図６５参照）が格納したタスクの実行環境を取り出して実行する。このときはその処理へ飛んでいって，ここへは再び戻ってこない。
【０５２３】
以上において説明したように，実施例２４に係るＮＣ装置の制御ソフトウェア実行システムＡ２４はタスクの走行割合を設定でき，また，タスクの割り込み処理の実際の走行時間を知ることができる機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。
【０５２４】
実施例２４によれば，ＮＣ装置の制御ソフトウェア実行システムＡ２４は，ＮＣ装置機能を実現するタスクと，割り込み処理の正確な実行時間を知ることにより，各タスクの走行時間の割り当てが，容易に行えるようになりシステムの構築が容易となる。
【０５２５】
〔実施例２５〕
次に，実施例２５について説明する。実施例２５に係るＮＣ装置の制御ソフトウェア実行システムＡ２５の全体構成は，実施例１８（図４８参照）とほぼ同様であるので，その説明を省略する。
【０５２６】
次に，ＮＣ装置の制御ソフトウェア実行システムＡ２５の動作について説明する。ＮＣ装置の運転モード選択部分の処理手順は図２９に示したものと同様である。すなわち，Ｑ１は，ＮＣ装置のモードがグラフィックチェックか否かを判定する部分である。Ｑ２は，グラフィックチェックモードであると判断した場合であり，その場合はグラフィック表示タスクの割り当てを増やし，他の，例えば，自動モード処理タスクの割り当てをその分減らす。
【０５２７】
Ｑ３は，グラフィックチェックモードではないと判断した場合で，自動モードか否かを判定する処理である。Ｑ４は，自動モードであると判断した場合であって，自動モード処理タスクの割り当てを増やし，他の処理タスクの割り当てを減らす。Ｑ５は，手動モードか否かのチェックである。Ｑ６は，手動モードであると判断した場合であって，手動処理タスクの割り当てを増やし，他の処理タスクの割り当てを減らす。それ以降は，その他のモードの場合であるが，その説明は省略する。
【０５２８】
以上において，説明したとおり，実施例２５に係るＮＣ装置の制御ソフトウェア実行システムＡ２５は，ＮＣ装置の運転モードによってタスクの走行割合を変化させて，ＮＣ装置

の制御ソフトウェアを実行するものである。
【０５２９】
実施例２５によれば，ＮＣ装置の制御ソフトウェア実行システムＡ２５は，ＮＣ装置の運転モードによってタスクの走行割合を変化させることにより，モード毎にタスクに適した時間だけ，タスクを走らせることができ，ＮＣ装置の実行速度が向上する。
【０５３０】
〔実施例２６〕
次に，実施例２６について説明する。実施例２６に係るＮＣ装置の制御ソフトウェア実行システムＡ２６は，同一の加工プログラムを複数回走らせるとき（同じ加工を複数回行うとき），初回加工時における各タスクの走行状況に基づいて２回目以降の加工時における各タスクの走行配分を最適化するものである。
【０５３１】
実施例２６に係るＮＣ装置の制御ソフトウェア実行システムＡ２６の全体構成は，上記実施例１８（図４８参照）とほぼ同様であるので，その説明を省略するが，以下の説明の都合上，本実施例においては，例えば，実施例２４と同様に，タスク毎の走行割合を指定する手段と，実際のタスクの走行状況を記録できる手段を備えているものとする。
【０５３２】
使用するデータ構造としては，実施例２４と同様，タスク開始システムクロックメモ５１０（図４９参照），基準周期５１１を格納しておく領域（図４９参照），タスク走行状態メモ５１２（図４９参照），タスク走行時間テーブル５３１（図６１参照）などを備える。タスク走行状態メモ５１２（図４９参照）は，“走行残りクロック数”，“繰り返し回数”，“走行禁止フラグ”などの欄があり，タスク走行時間テーブル５３１（図６１参照）にはタスク毎の“合計クロック数”の欄がある。
【０５３３】
基準時間割り込み処理の手順，走行時間警告割り込み処理の手順は，実施例１８（図５０，図５１参照）と同様であり，システムクロック割り込み処理の手順は，実施例２４（図６６参照）と同様なので，その説明は省略する。
【０５３４】
複数回加工時の処理を時間を短縮するための，各タスクの走行割当時間の最適化するための手順は図３０のフローチャートと同様である。すなわち，Ｒ１は同一加工の１回目か，２回目以降かを判定する処理である。Ｒ２は１回目の加工時の処理で，各タスクの総走行時間（クロック数）を計測し，記録する。Ｒ３は走行したタスクの内，アイドルタスクなどの必要のないタスクを除いた他の必要なタスクの走行時間の合計を計算する。
【０５３５】
Ｒ４は，Ｒ３で求めた合計に基づいて，必要なタスクの走行割合を計算する。走る必要がないタスクの走行割合は０とする。Ｒ５は，Ｒ４で計算した割合をタスク走行状態メモ５１２（図４９参照）の走行割合の欄に登録する。Ｒ６は２回目以降の加工時の処理で，Ｒ５で登録したタスク走行状態メモ５１２（図４９参照）の各タスクの走行割合により各タスクを走行させる。
【０５３６】
以上説明した通り，実施例２６に係るＮＣ装置の制御ソフトウェア実行システムＡ２６は，１回目の加工によって最適なタスクの走行割合を自動的に設定する機能を備えてＮＣ装置の制御ソフトウェアを実行するものである。
【０５３７】
実施例２６によれば，ＮＣ装置の制御ソフトウェア実行システムＡ２６は，機能を実現する各タスクの走行時間の割り当てを自動的に最適なものにすることにより，繰り返し加工における加工時間の短縮を実行できる。
【０５３８】
〔実施例２７〕
次に，実施例２７について説明する。実施例２７に係るＮＣ装置の制御ソフトウェア実行

システムＡ２７は，予め登録された，各タスクの基準の走行割合や，全走行時間と，実際に走っているＮＣ装置の制御ソフトウェア実行システムの各タスクのデータが，予め指定した割合以上にずれている場合は，タスクの異常と判定する。

【０５３９】

実施例２７に係るＮＣ装置の制御ソフトウェア実行システムＡ２７の全体構成は，実施例２４（図４８参照）と同様であるので，その説明は省略する。また，実際に走っているＮＣ装置の制御ソフトウェア実行システムの各タスクのデータを測定するのは，実施例と同様であるので，その説明は省略する。

【０５４０】

各タスクが正常に走っているか否かを判定するのに必要なデータ構造を，図６７に示す。５５０は“タスク走行標準データ”で，タスク毎に“走行割合”や“全走行クロック数”などの欄がある。それらのデータは，システム生成時に予め作成しておくか，あるいは何らかのＳＶＣによって，そのデータを使うよりも前の，任意の時期に設定しておいてもよい。

【０５４１】

図において，５５１は，この割合まではタスク走行時間のずれがあっても，タスク異常と見做さない許容度を格納する“走行時間判定許容度”である。５５２は，この範囲内であれば，タスク走行割合にずれがあってもタスク異常と見なさない許容度を格納する“走行割合判定許容度”である。この走行割合判定許容度には下限欄と上限欄があり，それがこの範囲内ならばタスク異常と見なさないものである。

【０５４２】

各タスクが正常に走っているか否かを判定するための，タスク走行時間テーブル５３１（図６１参照）とタスク走行標準データ５５０（図６７参照）の比較は，例えば，タスクが切り換えるときに，スケジューラで行ってもよいし，あるいは，定期的に走るタスク異常判定タスクを設けて，そのタスクにより判定を行ってもよい。以下の説明では，タスクが正常に走っているか否かの判定を，異常判定タスクで行うものとする。また，走行時間判定許容度５５１は全走行時間と，走行割合について別々に設けてもよいが，ここでは説明の都合上，同じ値を用いるものとする。

【０５４３】

図６８は，異常判定タスクの動作を示すフローチャートである。ＳＳ１は，異常判定タスクの先頭で，タスク走行時間テーブル５３１（図６１参照）の，タスク毎の合計クロック数の欄の値の全ての和を求める処理である。ＳＳ２は，全てのタスクに対して，以下のＳＳ３からＳＳ５までの処理を行ったか否かの判定であり，全タスクに対して処理を終了すればＳＳ６へ処理が移行する。

【０５４４】

ＳＳ３は，タスク走行時間テーブル５３１（図６１参照）の，当該タスクの合計クロック数の値と，タスク走行標準データ５５０の全走行クロック数を比較する。当該タスクのタスク走行時間テーブル５３１（図６１参照）の合計クロック数の値が，タスク走行標準データ５５０の全走行クロック数の値に走行時間判定許容度５５１の値を乗じたものよりも小さいか等しければ，このタスクは正常であると見なす。

【０５４５】

ＳＳ４は，ＳＳ３において，タスクに異常が発生したと判定したときの処理で，例えば，オペレーターにタスク異常が発生したことを知らせるメッセージを送るなどの，何れかのエラー処理を行う。

【０５４６】

ＳＳ５は，タスクの走行割合のチェックである。タスク走行時間テーブル５３１（図６１参照）の合計クロック数の値を，全タスク分加えて，それで当該タスクの合計クロックを割ることにより当該タスクの走行割合を求める。その値がタスク走行標準データ５５０の走行割合判定許容度の下限と上限の間にあれば，このタスクは正常であると見なす。もし，異常があると判定したときはＳＳ４に処理が移行する。ＳＳ６は，全タスクを判定した

結果，異常状態のタスクは見つからなかったので，タスク終了のＳＶＣを発行し，次に呼ばれるまで何もしない。

【０５４７】

以上で説明した通り，実施例２７に係るＮＣ装置の制御ソフトウェア実行システムＡ２７にあっては，ＮＣ装置の機能を実現しているタスクに異常が発生したときにそれを検知する機能を備えている。

【０５４８】

実施例２７によれば，ＮＣ装置の制御ソフトウェア実行システムＡ２７は，ＮＣ装置の機能を実現しているタスクに異常が発生したときにそれを検知する機能を備えているため，ＮＣ装置に異常が発生したことが直ちに判定することができる。

【０５４９】

〔実施例２８〕

次に，実施例２８について説明する。実施例２８に係るＮＣ装置の制御ソフトウェア実行システムＡ２８の全体構成は，実施例１８（図４８参照）と同じであるので，その説明は省略する。

【０５５０】

ただし，本実施例では，ＲＡＭ４上のデータとしてはタスク走行時間計測用の積算システムクロックメモ５０３，タスク停止用に使用するシステムクロック数メモ５０１，ディスパッチクロック数５０８，ディスパッチクロック数メモ５０９を使用する。

【０５５１】

図６９は，本実施例において，必要なデータ構造を示したものであり，６００は“タスク停止テーブル”である。各タスクは予め，あるいはそのタスクが開始させられるときにＳＶＣによって，タスクが起動してから停止するまでのクロック数をこのタスク停止テーブル６００の“停止クロック数”の欄に登録しておく。指定しないタスクはこの値を０としておく。

【０５５２】

５１０（図４９参照）は“タスク開始システムクロックメモ”である。ＲＴＯＳが，タスクを起動する直前に，積算システムクロック数メモ５０３（図４８参照）の値を記録しておく。そして，そのタスクが終了してＲＴＯＳに戻って来たときの，積算システムクロック数メモ５０３との差を求めてタスク走行時間を求める。６１１は“タスク停止クロックメモ”である。タスク停止処理をすぐに処理できないときは，このメモをオンしておいて後で処理する。

【０５５３】

本実施例に係る，ＮＣ装置の制御ソフトウェア実行システムＡ２８は，タスクを起動するたびに，指定されていればタスク停止テーブル６００の停止クロック数をシステムクロック数メモ５０１に設定する。そして，毎回のシステムクロック割り込み時に，ＲＴＯＳがシステムクロック数メモ５０１を減じていって０になったところで，タスク停止処理を呼んで当該タスクを強制的に停止させる。

【０５５４】

もし，タスク停止処理を呼び前に，他のタスクによってＣＰＵを横取りされた場合は，積算システムクロックメモ５０３（図４８参照）と，タスク停止クロックメモ６１１から，当該タスクの走行クロックを求め，その分をタスク停止テーブル６００の停止クロック数から減ずる。このようにして，指定タスクの最大走行クロック数を制限することができる。

【０５５５】

次に，ＮＣ装置の制御ソフトウェア実行システムＡ２８の動作について説明する。最初にタスク停止処理について説明する。図７０は，タスク停止処理の手順について説明したフローチャートである。

【０５５６】

ＷＷ２は割り込まれた処理がタスク処理中か，それ以外かを判定する処理部分である。Ｗ

Ｗ３はタスク処理中ではなかったときで，その場合には，タスク停止クロックメモ６１１をオンする。ＷＷ４は割り込み禁止を解除して，割り込まれた処理へ戻る。ＷＷ５はタスク処理中に割り込みが発生した場合で，まず割り込まれたタスクのＴＣＢを探し出す。

【０５５７】

また，ＷＷ６は，ＷＷ４で見つけたＴＣＢへ割り込まれたタスクの実行環境を格納する。ＷＷ７はタスク停止ＳＶＣと同様の当該タスクの終了処理を行う。例えば，当該ＴＣＢのステータスの欄を停止にして，ＴＣＢ待ち行列から，停止中タスクの待ち行列へつなぎ変える等の処理である。

【０５５８】

図７１は，システムクロック割り込み処理の手順を示すフローチャートである。以下の説明において，システムクロック数メモ５０１に関しては，実施例１８のシステムクロック割り込み処理（図４７参照）と同様に，有効フラグを設けてその有効フラグが“有効”である場合のみシステムクロック数メモ５０１に対する処理を行えばよいが，説明が煩雑になるため，本実施例の説明のフローチャートではその処理を省略する。

【０５５９】

ＸＸ１は，割り込みが発生した時の割り込み処理の先頭であり，最初の他の割り込みの発生を禁止する，通常この処理はＣＰＵによって自動的に行われる。

【０５６０】

ＸＸ２０は，ディスパッチクロック数メモ５０９，システムクロック数メモ５０１のそれぞれから１減算する処理であり，ＸＸ２１は積算システムクロック数メモ５０３に１加える処理であり，ＸＸ２２はｓｋｉｐフラグをオンする処理である。ここで，ｓｋｉｐフラグは，ＲＡＭ４上に設けた，このシステムクロック割り込みでスケジューリングなどの処理を行うか否かを示すために用いるフラグである。

【０５６１】

ＸＸ２３は，システムクロック数メモ５０１が，０になったか否かの判定を行う処理であり，その結果が０と判断した場合は，ＸＸ２４で，図４１に示したタスク停止処理を呼ぶ。その後，ＸＸ２５でｓｋｉｐフラグをオフにする。ここで，ｓｋｉｐフラグとはＲＡＭ４上にあり，ディスパッチ処理を呼ぶか否かの判定に使用するフラグである。

【０５６２】

ＸＸ２３の結果が０でないと判断したときは，ＸＸ２６で，ディスパッチクロック数メモ５０９が０か否かの判定を行う。その結果が，０であれば，ＸＸ２７でｓｋｉｐフラグをオフにして，ＸＸ２８で，ディスパッチクロック数５０８の値をディスパッチクロック数メモ５０９へ代入することにより，ディスパッチクロック数メモ５０９の再初期化を実行する。

【０５６３】

ＸＸ２９は，ｓｋｉｐフラグがオンかオフかの判定を行う処理で，オンの場合にはＸＸ３０で割り込み禁止の解除を行った後，ＸＸ７へ飛び，割り込まれた処理に直接復帰する。オフの場合にはＸＸ２へ飛び，通常のディスパッチ処理を行う。

【０５６４】

ＸＸ２は，未使用のＩＣＢをＩＣＢプール（未使用ＩＣＢを保持しておく場所）から１つ獲得する。ＸＸ３は，ＩＣＢに，この割り込み処理（自分自身の）の，実行環境を格納する。この中には次に実行する命令のアドレスとしてＸＸ８の処理の先頭アドレスをＩＣＢに格納することも含まれる。ＸＸ４は，ここで，作成したＩＣＢを，スケジューラが実行するＩＣＢ待ち行列に追加する。ＸＸ５は，ＸＸ１の割り込みの禁止を解除する。これ以降は再び割り込みが発生する可能性がある。異常の処理は割り込み禁止時間をできる限り短くするためである。

【０５６５】

ＸＸ６は，割り込まれた処理が，割り込み処理またはＲＴＯＳか，それともタスクの処理中かを判定する。ＸＸ７は，割り込まれた処理が割り込み処理またはＲＴＯＳを実行中のときで，このときは割り込まれた処理に直接戻る。ＸＸ８は，割り込まれた処理がタスク

の処理を実行中のときで，このときはまず割り込まれたタスクのＴＣＢを探し出す。ＸＸ９は，ＸＸ８で見つけたＴＣＢへ，割り込まれたタスクの実行環境を格納する。

【０５６６】

ＸＸ１０は，タスク停止クロックメモ６１１がオンか否かを調べる処理である。ＸＸ１１は，タスク停止クロックメモ６１１がオンのときの処理で，タスク停止ＳＶＣと同様の，当該タスクの終了処理を行う。例えば，当該ＴＣＢのステータスの欄を停止にしてＴＣＢ待ち行列から，停止中タスクの待ち行列へつなぎ変える等の処理である。

【０５６７】

ＸＸ１２は，タスク停止クロックメモ６１１がオフのときの処理で，積算システムクロック数メモ５０３と，タスク開始クロックメモの差から当該タスクの走行クロック数を求め，タスク停止テーブル６００の停止クロック数から減ずる。ＸＸ１３は，ＸＸ９において作成したＴＣＢを，スケジューラが実行するＴＣＢ待ち行列にプライオリティーの順に追加する。ＸＸ１４は，スケジューラへ飛ぶ。

【０５６８】

ＸＸ１５は，スケジューラから戻ってくるとここへ来る処理である。スケジューラへ飛んだ後すぐにこの処理が行われるとは限らず，他のＩＣＢが処理された後かもしれないが，いずれはここへ戻ってくる。ここらから割込毎に固有の本来の割り込み処理であり，割り込み要因によって処理は異なる。ＸＸ１６は，割り込み処理の終了であり，通常のサブルーチンのリターン命令が多い，このリターン命令によって，スケジューラのＩＣＢ処理部に戻る。

【０５６９】

図７２を用いて，本実施例に係るＮＣ装置の制御ソフトウェア実行システムＡ２８のスケジューラの動作を説明する。以下の，説明の都合上，本実施例においては，優先順位が最低で，外部に対して何も処理を行わないアイドルタスクが常に走っているものとする。

【０５７０】

ＹＹ１は，割り込み処理（ＩＣＢ）の待ち行列があるか否かを調べる処理である。ＹＹ２は，割り込み処理の待ち行列があったときの処理でその待ち行列を実行するために最初に割り込みを禁止する。これは，割り込み処理の待ち行列のリストを処理している最中に，割り込みが発生して割り込み処理の待ち行列に追加を行おうとするとリストの整合性が保たれなくなるためである。

【０５７１】

ＹＹ３は割り込み処理の待ち行列のリストから先頭のＩＣＢを取り除く。ＹＹ４は割り込み禁止を解除する。

【０５７２】

ＹＹ５は，ＹＹ３で取り除いたＩＣＢからＸＸ３で格納した実行アドレスやレジスターなどの割り込み処理の実行環境を取り出して実行する。これがＸＸ１５（図７１参照）である。ＹＹ６は，ＸＸ１６から戻ってくるところであり，再び割り込みを禁止する。ＹＹ７は，ＹＹ３で取り除いたＩＣＢをＩＣＢプールへ返す。ＹＹ８は，割り込み禁止を解除してからＹＹ１へ戻る。後は割り込み処理待ち行列がある間はＹＹ１からＹＹ７の処理を繰り返す。

【０５７３】

ＹＹ９は割り込みを禁止する。ＹＹ１０は割り込み処理の待ち行列のリストから先頭のＴＣＢを獲得する。ＹＹ１１は当該ＴＣＢのタスクが，タスク停止テーブル６００の停止クロック数が０以上のタスクである場合は，当該タスクのタスク停止テーブル６００の停止クロック数の値をシステムクロック数メモ５０１に代入する。

【０５７４】

この設定した数だけのクロック数が経過すると，図７１で説明したシステムクロック割り込みがタスク停止処理を呼ぶ。ＹＹ１２は，割り込み禁止を解除する。ＹＹ１３は，ＹＹ１０でタスク処理の待ち行列（リスト）から獲得したＴＣＢからＸＸ９（図７１）で格納したタスクの実行環境を取り出して実行する，そしてその処理へ飛んで行き，ここへは戻

ってこない。
【０５７５】
以上，説明したように，実施例２８に係るＮＣ装置の制御ソフトウェア実行システムＡ２８は，通常のプライオリティーベースのスケジューリング方式しか持たないＲＴＯＳに，タスク走行時間の制限を行える機能を追加した方式であり，この方式に基づいてＮＣ装置の制御ソフトウェアを実行するものである。
【０５７６】
実施例２８によれば，ＮＣ装置の制御ソフトウェア実行システムは，ＮＣ装置機能を実現するタスクの，走行時間による制限を行えるようになる。そのため，タスクに割り当てた時間を超えてタスクが走ることがなくなり，システムの構築が容易となる。
【０５７７】
〔実施例２９〕
次に，実施例２９について説明する。実施例２９に係るＮＣ装置の制御ソフトウェア実行システムは，加工プログラム解析処理タスクなどを，従来例のように，１ブロック処理する毎にタスクを抜けるのではなく，加工プログラムが存在する限り，解析を続けるようにして作成しておき，解析の中断は，時間によるものとする。
【０５７８】
実施例２９に係るＮＣ装置の制御ソフトウェア実行システムＡ２９の，加工プログラム解析タスクの実行手順を図４３に基づいて説明する。Ｚ１は，加工プログラム解析タスクの先頭であり，指定された加工プログラムのブロックを全て解析し終えたかを判定する。
【０５７９】
Ｚ２は，全ての加工プログラムのブロックの解析を終えた場合の処理で，タスク終了ＳＶＣを発行して，再び起動させるまで何もしない。Ｚ３は，未解析の加工プログラムのブロックが存在する場合の処理で，ブロックの解析を実行する。以後，全ブロックの解析が終了するまでＺ１の処理を繰り返す。
【０５８０】
以上で説明したように，実施例２９に係るＮＣ装置の制御ソフトウェア実行システムＡ２９は，加工プログラム解析タスクを，プログラムがある限り，解析し続けるようにして，解析の中断を時間によって行う，ＮＣ装置の制御ソフトウェアを実行するものである。
【０５８１】
上記実施例２９によれば，ＮＣ装置の制御ソフトウェア実行システムＡ２９は，加工プログラムの解析を行うタスクの走行，中断を時間によって行うようにしたので，加工プログラムのブロックの大きさによる，処理時間の余りが無くなり，特に微小線分からなる加工プログラムの解析において処理能力の向上が期待できる。さらに，タスクの作成方法として，加工プログラムのブロック毎に，終了するなどの煩雑な処理が不必要となる。
【０５８２】
（各実施例の効果）
以上のように，この発明の各実施例に係るＮＣ装置の制御ソフトウェア実行システムによれば，ＮＣ装置の機能を分担して実現している，タスクの制御をきめ細かく実行でき，各タスクの処理時間を最適化することができるため，無駄時間を少なくして，全体の処理時間を早めることができる。また，ＮＣ装置の制御ネットワーク実行システムの異常を，システム自体が判定できる手段を備えているため，異常事態が発生したときの回避処理も容易となる。
【０５８４】
【発明の効果】
この発明に係る制御ソフトウェア実行システムの制御方法にあっては，ＮＣ装置機能を実現するタスクの走行時間割合を保証する制御を実行することができるため，タスクに割り当てた時間の無駄を少なくでき，さらに，タスクを設定した走行時間割合で制御できるため，システムの構築が容易になる。
【０５８５】

また，次の発明に係る制御ソフトウェア実行システムの制御方法にあっては，ＮＣ装置機能を実現するタスクに連続した走行時間を保証できるようになるため，タスクに割り当てた時間の無駄を少なくでき，さらに，タスクをその走行時間において制御できるため，システムの構築が容易になる。

【図面の簡単な説明】

【図１】　実施例１に係るＮＣ装置の制御ソフトウェア実行システムの全体構成を示すブロック図である。

【図２】　タスク走行周期テーブルの構造を示す説明図である。

【図３】　タスク走行待ちリストの構造を示す説明図である。

【図４】　実施例１に係るＮＣ装置の制御ソフトウェア実行システムがタスク走行待ちリストを作成するときの処理手順を示すフローチャートである。

【図５】　走行待ちリストの構造を示す説明図である。

【図６】　走行待ちリストに，ブロックを追加した後の状態を示す説明図である。

【図７】　タスク起動割り込みの処理手順を示すフローチャートである。

【図８】　一般の割り込みの処理手順を示すフローチャートである。

【図９】　実施例２に係るＮＣ装置の制御ソフトウェア実行システムの全体構成を示すブロック図である。

【図１０】　実施例２に係るＮＣ装置の制御ソフトウェア実行システムにおいて必要なデータ構造を示す説明図である。

【図１１】　基準割り込みの処理手順を示すフローチャートである。

【図１２】　走行時間警告割り込みの処理手順を示すフローチャートである。

【図１３】　一般の割り込みの処理手順を示フローチャートである。

【図１４】　スケジューラの処理手順を示すフローチャートである。

【図１５】　実施例３に係るＮＣ装置の制御ソフトウェア実行システムの全体構成を示すブロック図である。

【図１６】　タスク終了チック数リストの構造を示す説明図である。

【図１７】　タスク連続走行保証テーブルの構造を示す説明図である。

【図１８】　連続走行割り込みの処理手順を示すフローチャートである。

【図１９】　一般の割り込みの処理手順を示すフローチャートである。

【図２０】　実施例４に係るＮＣ装置の制御ソフトウェア実行システムの全体構成を示すブロック図である。

【図２１】　実施例６に係るＮＣ装置の制御ソフトウェア実行システムの全体構成を示すブロック図である。

【図２２】　実施例６に係るＮＣ装置の制御ソフトウェア実行システムにおいて必要なデータ構造を示す説明図である。

【図２３】　一般の割り込みの処理手順を示すフローチャートである。

【図２４】　スケジューラの処理手順を示すフローチャートである。

【図２５】　スケジューラの処理手順を示すフローチャートである。

【図２６】　割り込み処理タスクの処理手順を示すフローチャートである。

【図２７】　一般の割り込みの処理手順を示すフローチャートである。

【図２８】　スケジューラの処理手順を示すフローチャートである。

【図２９】　運転モード選択処理の処理手順を示すフローチャートである。

【図３０】　タスク走行時間の割り当てを最適化するための処理手順を示すフローチャートである。

【図３１】　実施例１１に係るＮＣ装置の制御ソフトウェア実行システムにおいて必要なデータ構造を示す説明図である。

【図３２】　異常判定タスクの処理手順を示すフローチャートである。

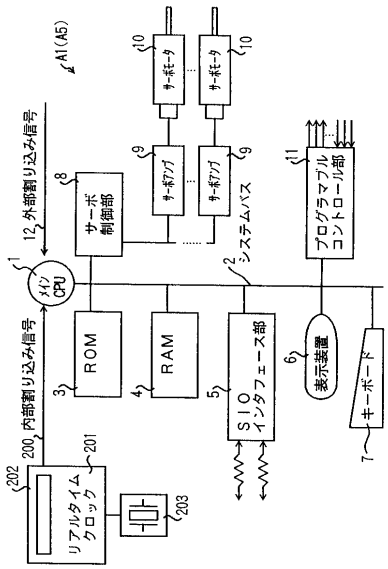
【図３３】　実施例１２に係るＮＣ装置制御ソフトウェア実行システムにおいて必要なデータ構造を示す説明図である。

【図３４】　工具選択処理の処理手順を示すフローチャートである。

【図３５】　実施例１３に係るＮＣ装置の制御ソフトウェア実行システムにおいて必要なデータ構造を示す説明図である。
【図３６】　速度再計算処理の処理手順を示すフローチャートである。
【図３７】　実施例１４に係るＮＣ装置の制御ソフトウェア実行システムにおいて必要なデータ構造を示す説明図である。
【図３８】　別加工プログラムを選択する処理の処理手順を示すフローチャートである。
【図３９】　実施例１５に係るＮＣ装置の制御ソフトウェア実行システムにおける必要なデータ構造を示す説明図である。
【図４０】　タスク停止割り込み処理の処理手順を示すフローチャートである。
【図４１】　一般の割り込み処理の処理手順を示すフローチャートである。
【図４２】　スケジューラの処理手順を示すスローチャートである。
【図４３】　加工プログラム解析タスクの処理手順を示すフローチャートである。
【図４４】　実施例１７（２１）に係るＮＣ装置の制御ソフトウェア実行システムの全体構成を示すブロック図である。
【図４５】　実施例１７に係るＮＣ装置の制御ソフトウェア実行システムがタスク走行待ちリストを作成するときの処理手順を示すフローチャートである。
【図４６】　タスク起動の処理手順を示すフローチャートである。
【図４７】　システムクロック割り込みの処理手順を示すフローチャートである。
【図４８】　実施例１８（２４，２５，２６，２７，２８）に係るＮＣ装置の制御ソフトウェア実行システムの全体構成を示すブロック図である。
【図４９】　実施例１８に係るＮＣ装置の制御ソフトウェア実行システムにおいて必要なデータ構造を示す説明図である。
【図５０】　基準周期リセット処理の手順を示すフローチャートである。
【図５１】　走行時間警告処理の手順を示すフローチャートである。
【図５２】　システムクロック割り込み処理の手順を示すフローチャートである。
【図５３】　スケジューラの処理手順を示すフローチャートである。
【図５４】　実施例１９に係るＮＣ装置の制御ソフトウェア実行システムの全体構成を示すブロック図である。
【図５５】　タスク終了クロック数リストの構造を示す説明図である。
【図５６】　タスク連続走行保証テーブルの構造を示す説明図である。
【図５７】　タスク連続走行終了処理の手順を示すフローチャートである。
【図５８】　システムクロック割り込み処理の手順を示すフローチャートである。
【図５９】　実施例２０に係るＮＣ装置の制御ソフトウェア実行システムの全体構成を示すブロック図である。
【図６０】　実施例２２（２３）に係るＮＣ装置の制御ソフトウェア実行システムの全体構成を示すブロック図である。
【図６１】　実施例２２に係るＮＣ装置の制御ソフトウェア実行システムにおいて必要なデータ構造を示す説明図である。
【図６２】　システムクロック割り込み処理の手順を示すフローチャートである。
【図６３】　スケジューラの処理手順を示すフローチャートである。
【図６４】　スケジューラの処理手順を示すフローチャートである。
【図６５】　システムクロック割り込み処理の手順を示すフローチャートである。
【図６６】　スケジューラの処理手順を示すフローチャートである。
【図６７】　実施例２７に係るＮＣ装置の制御ソフトウェア実行システムにおいて必要なデータ構造を示す説明図である。
【図６８】　異常判定タスクの処理手順を示すフローチャートである。
【図６９】　実施例２８に係るＮＣ装置の制御ソフトウェア実行システムにおける必要なデータ構造を示す説明図である。
【図７０】　タスク停止処理の処理手順を示すフローチャートである。
【図７１】　システムクロック割り込み処理の手順を示すフローチャートである。

【図７２】　スケジューラの処理手順を示すフローチャートである。
【図７３】　従来におけるＮＣ装置の制御ソフトウェア実行システムの全体構成を示すブロック図である。
【図７４】　ＮＣ装置の制御ソフトウェア実行システムの，加工プログラムの解析実行手順を示す説明図である。
【図７５】　割り込み処理を示す概念図である。
【図７６】　各タスク動作の時間的関係を示すタイミングチャートである。
【図７７】　各タスク動作の時間的関係を示すタイミングチャートである。
【図７８】　自動プログラム動作の状態を示す説明図である。
【図７９】　解析タスクの処理の処理手順を示すフローチャートである。
【符号の説明】
１　メインＣＰＵ，２　システムバス，３　ＲＯＭ，４　ＲＡＭ，５　ＳＩＯインタフェース，６　表示装置，７　キーボード，８　サーボ制御部，９　サーボアンプ，１０　サーボモータ，１２　外部割り込み信号，２００　内部割り込み信号，２０１　リアルタイムクロック（プログラマブルインタラプトコントローラ），２０２　内部カウンタレジスタ，２０３　水晶発信器，２０４～２０７内部カウンタレジスタ，２１０　タスク開始チックメモ，２１１　基準周期，２１２　タスク走行状態メモ，２２０　タスク終了チェック数リスト，２３０　カウンタ値メモ，２３１　タスク走行テーブル，２５０　タスク走行標準テーブル，２５１　走行時間判定許容度，２５２　走行割合判定許容度，３６０　工具選択使用フラグ，３６２　工具選択処理スキップフラグ，３６３　選択工具メモ，３７０　工具速度決定フラグ，３７１　工具速度計算処理スキップフラグ，３７２　工具速度メモ，３７３　工具速度変更割合，３８０　加工プログラムメモ，３８１　加工プログラムスケジューリングテーブル，３８２　加工不能プログラムメモ，４００　タスク停止テーブル，４１１　タスク停止割り込みメモ，５００　システムクロック，５０１　システムクロック数メモ，５０２　システムクロック数メモ有効フラグ，５０３　積算システムクロック数メモ，５０５　走行時間警告用システムクロック数メモ，５０６　タスク連続走行終了クロック数メモ，５０７　タスク連続走行終了クロック数メモ有効フラグ，５０８　ディスパッチクロック数，５０９　ディスパッチクロック数メモ，５１０　タスク開始システムクロックメモ，５１１　基準周期，５１２　タスク走行状態メモ，５２０　タスク終了クロックブロック，５２１　タスク連続走行保証テーブル，５３０　カウンタ値メモ，５３１　タスク走行時間テーブル，５５０　タスク走行標準データ，５５１　走行時間判定許容度，５５２　走行割合判定許容度，６００タスク停止テーブル，６１１　タスク停止クロックメモ

【図１】



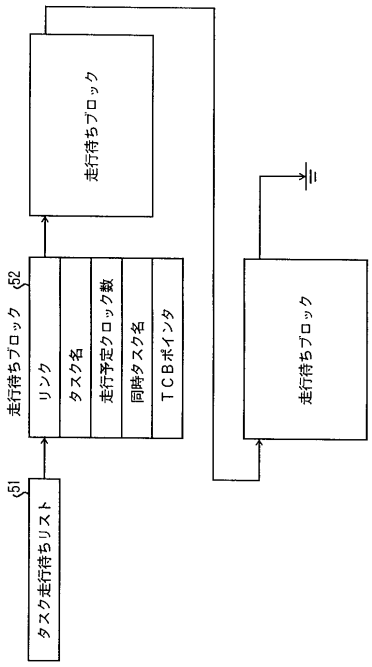
【図２】

タスク走行周期テーブル

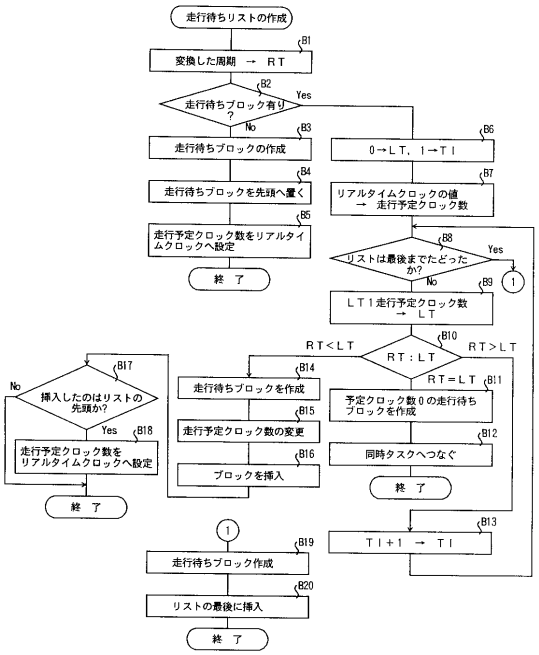
タスク名	走行周期	繰り返し回数
サーボ処理タスク	２０	∞
補正計算タスク	５０	１
プログラム解析タスク	１００	２
表示タスク	５００	∞
⋮	⋮	⋮

50

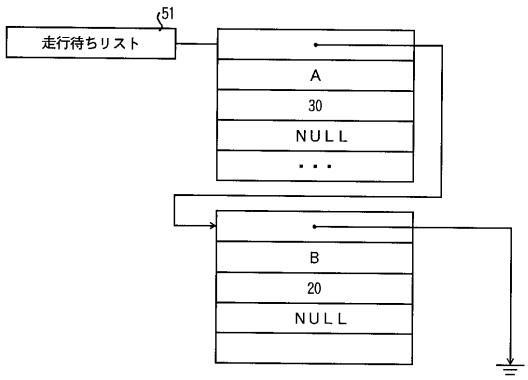
【図３】



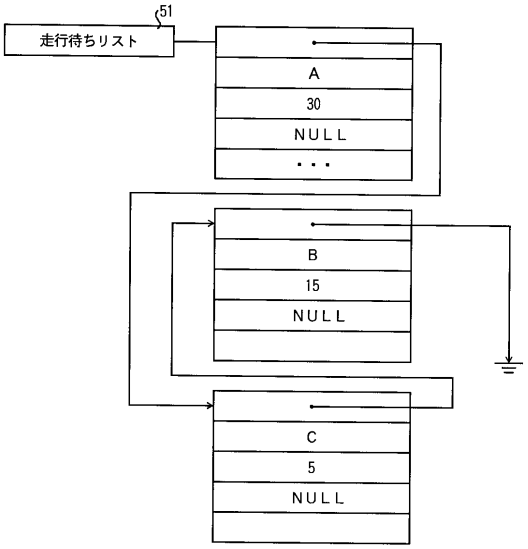
【図４】



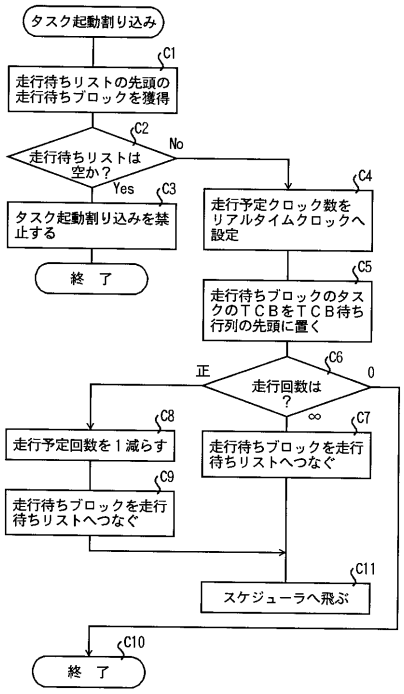
【図５】


51
走行待ちリスト
A
30
NULL
・・・
B
20
NULL

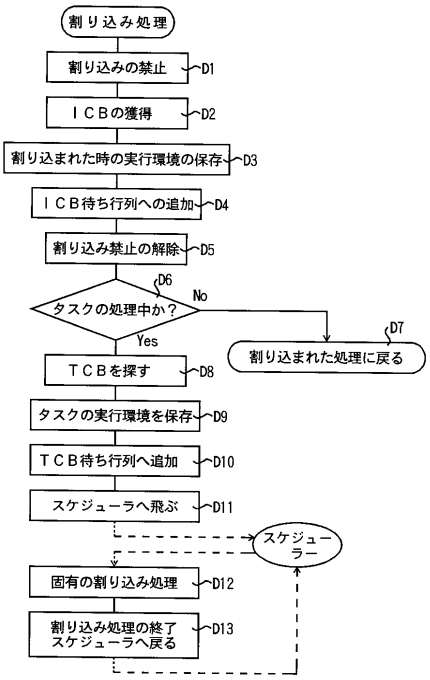
【図６】


51
走行待ちリスト
A
30
NULL
・・・
B
15
NULL
C
5
NULL

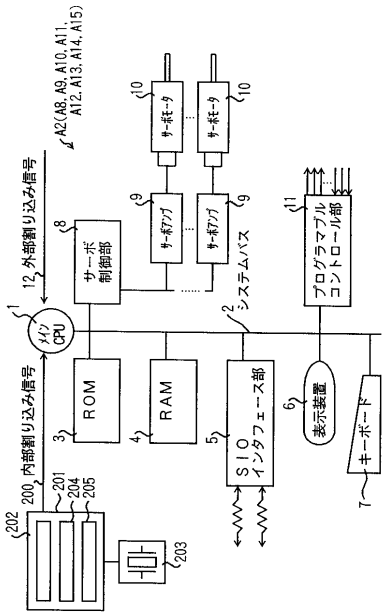
【図７】


タスク起動割り込み
C1
走行待ちリストの先頭の走行待ちブロックを獲得
C2
走行待ちリストは空か？
No
Yes
C3
タスク起動割り込みを禁止する
終 了
C4
走行予定クロック数をリアルタイムクロックへ設定
C5
走行待ちブロックのタスクのＴＣＢをＴＣＢ待ち行列の先頭に置く
C6
走行回数は？
正
0
∞
C8
走行予定回数を１減らす
C9
走行待ちブロックを走行待ちリストへつなぐ
C7
走行待ちブロックを走行待ちリストへつなぐ
C11
スケジューラへ飛ぶ
C10
終 了

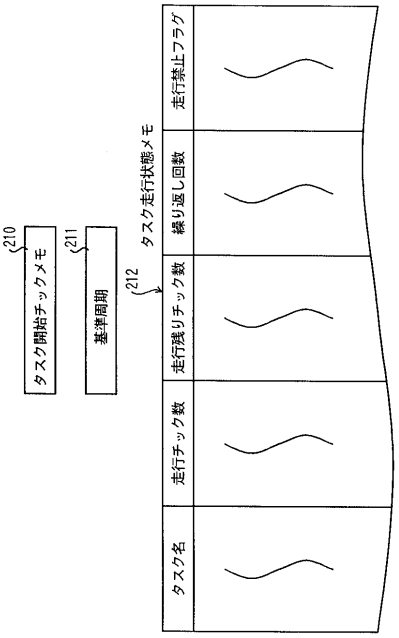
【図８】


割り込み処理
割り込みの禁止
D1
ＩＣＢの獲得
D2
割り込まれた時の実行環境の保存
D3
ＩＣＢ待ち行列への追加
D4
割り込み禁止の解除
D5
D6
タスクの処理中か？
No
Yes
D7
割り込まれた処理に戻る
ＴＣＢを探す
D8
タスクの実行環境を保存
D9
ＴＣＢ待ち行列へ追加
D10
スケジューラへ飛ぶ
D11
スケジューラー
固有の割り込み処理
D12
割り込み処理の終了スケジューラへ戻る
D13

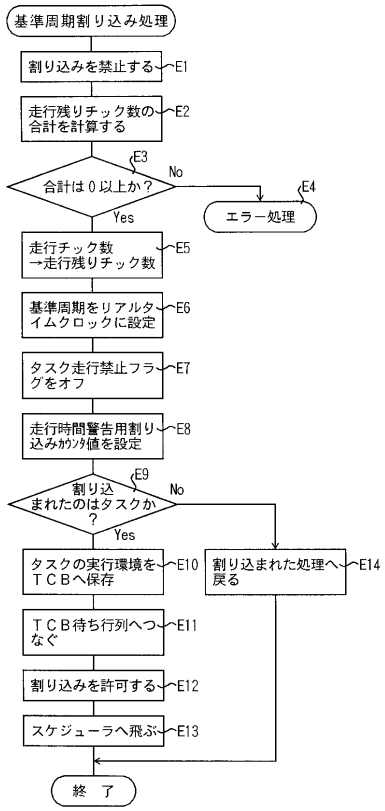
【図９】



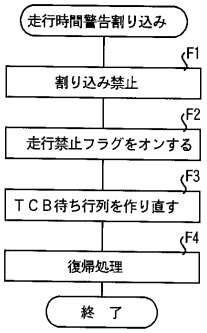
【図１０】



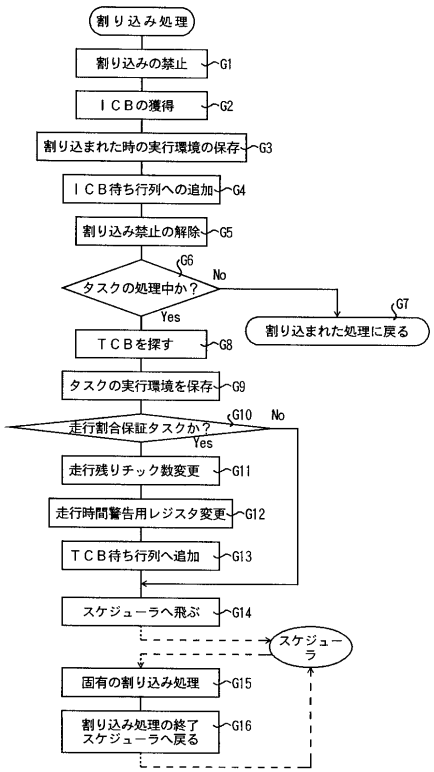
【図１１】



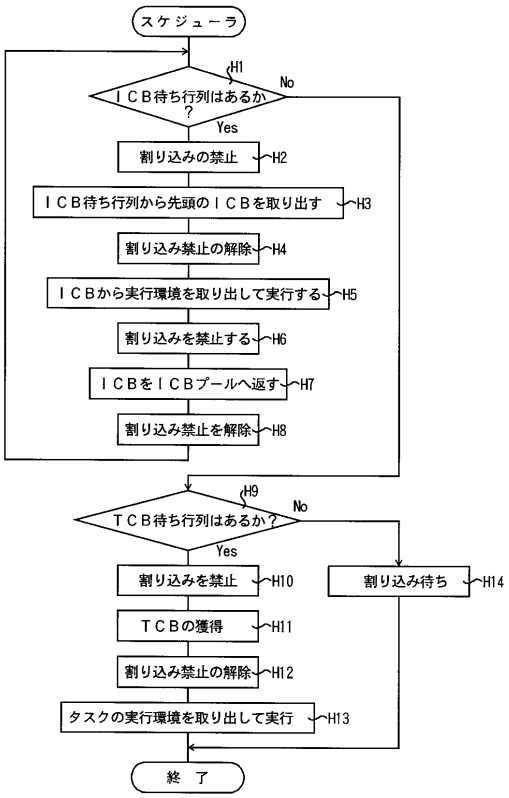
【図１２】



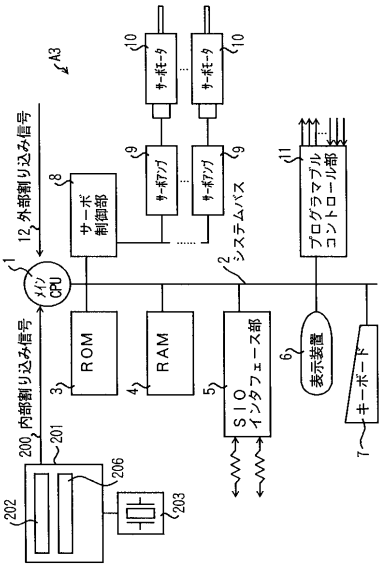
【図１３】



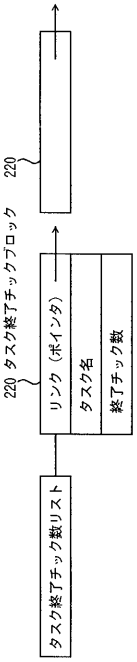
【図１４】



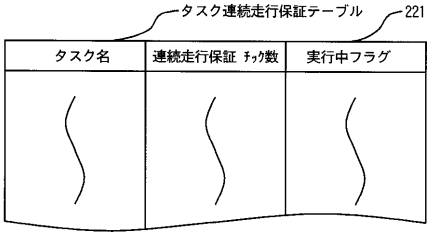
【図１５】



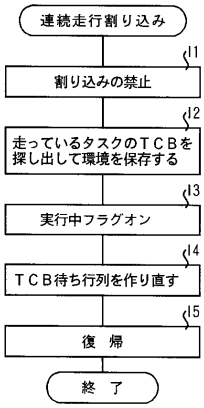
【図１６】



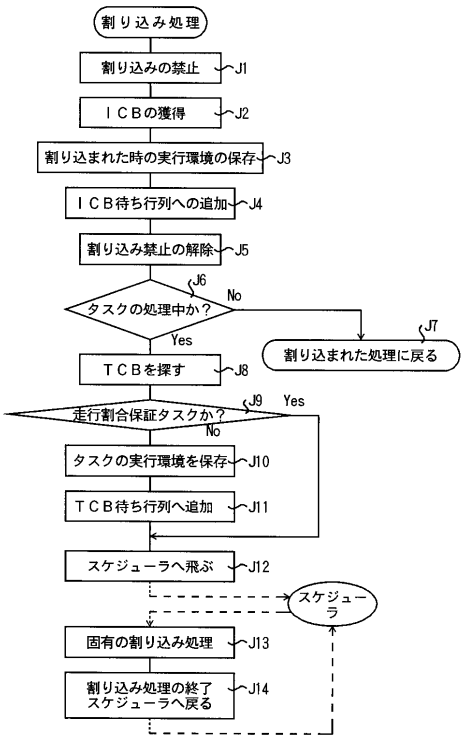
【図１７】



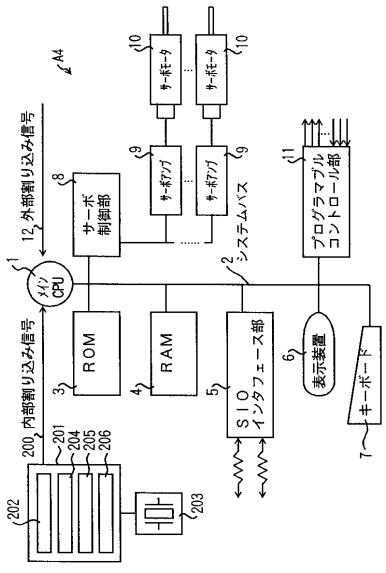
【図１８】



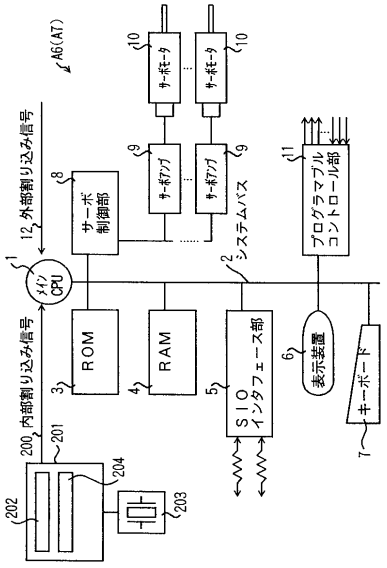
【図１９】



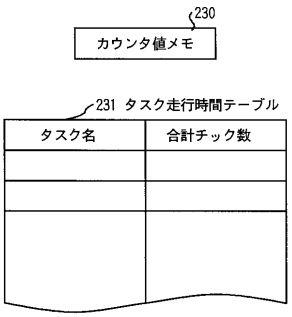
【図２０】



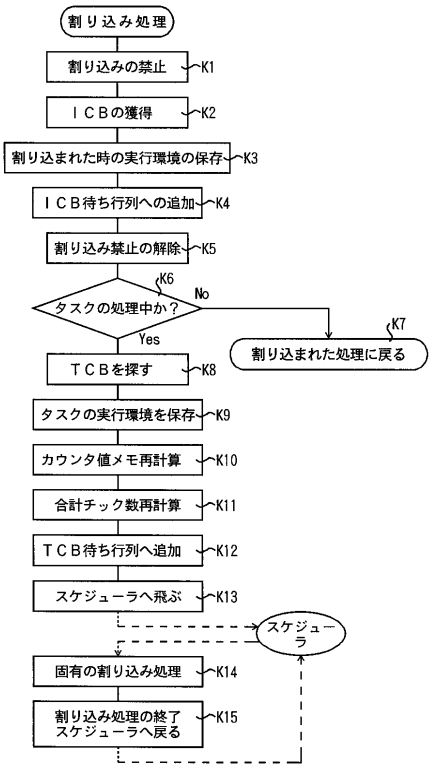
【図２１】



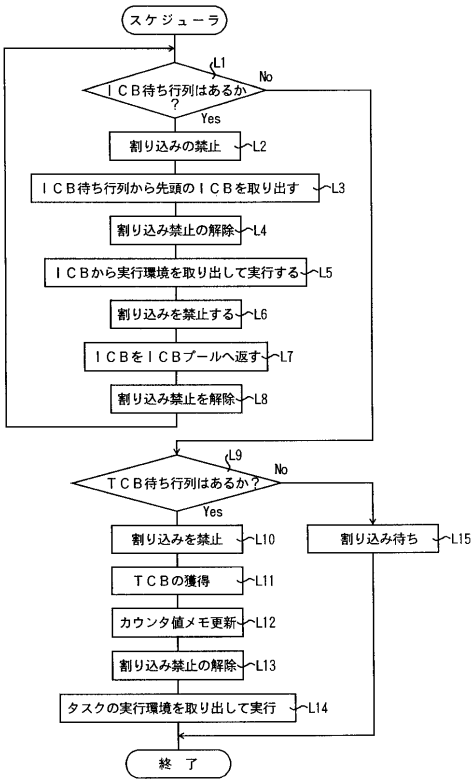
【図２２】



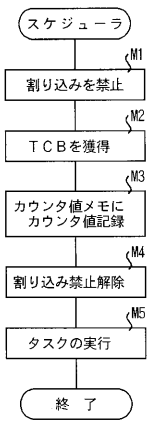
【図２３】



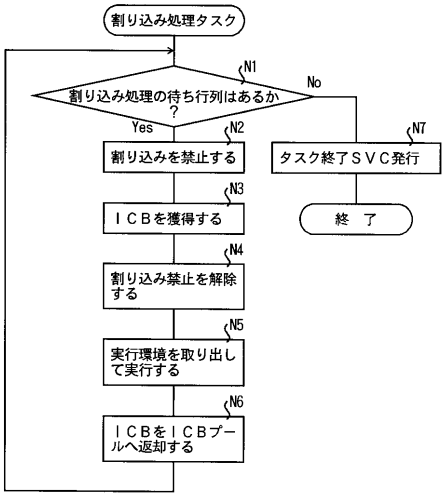
【図２４】



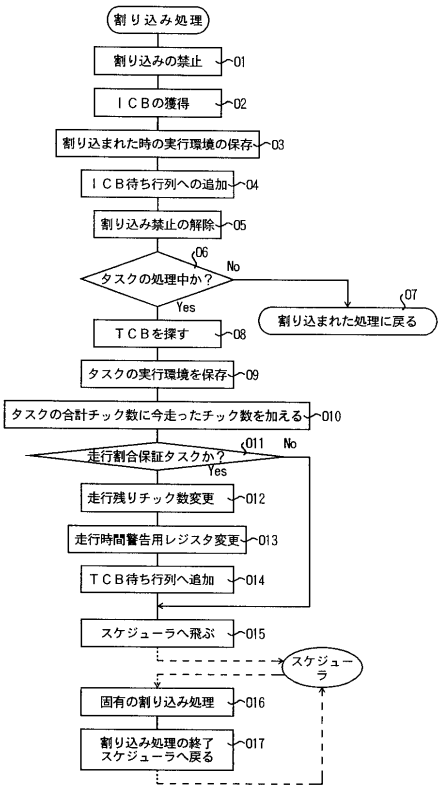
【図２５】



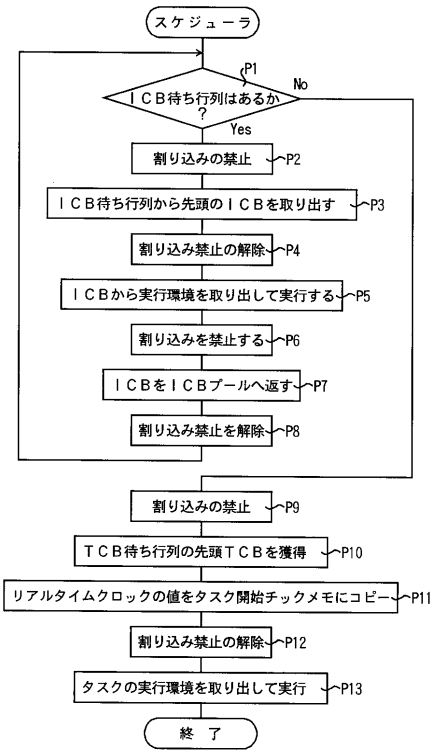
【図２６】



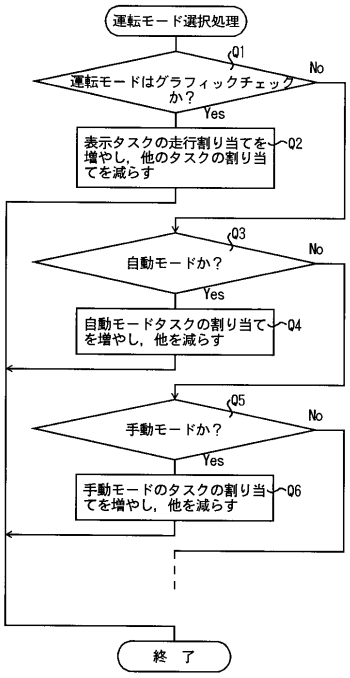
【図２７】



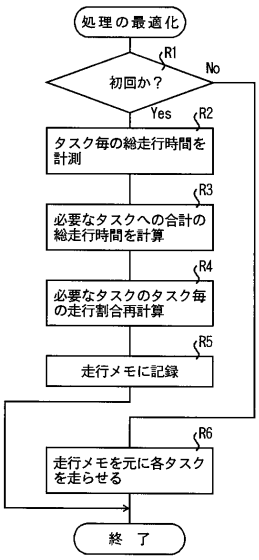
【図２８】



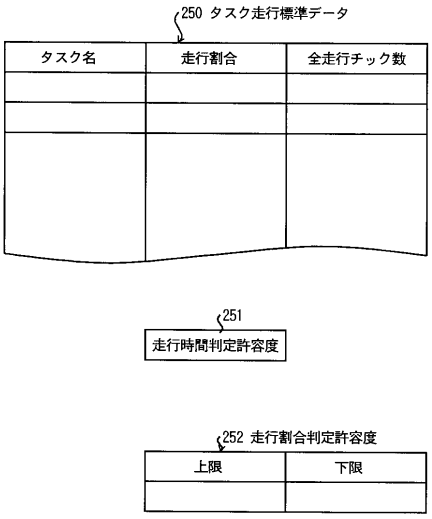
【図２９】



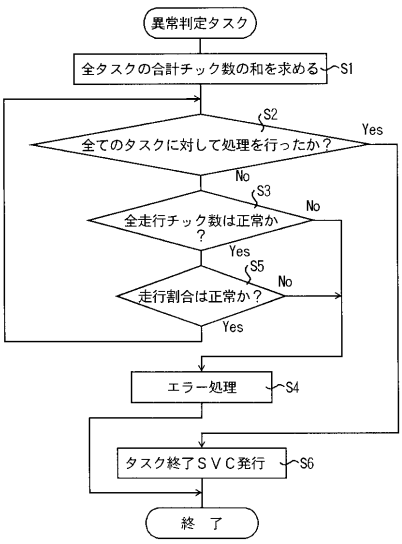
【図３０】



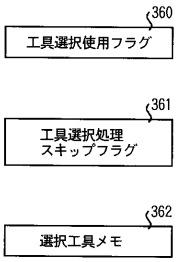
【図３１】



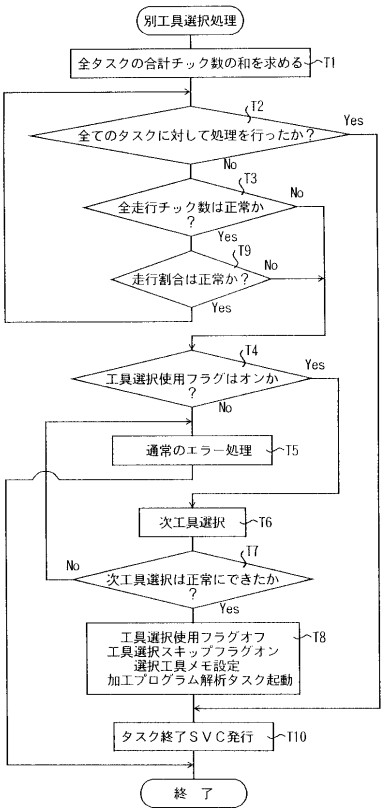
【図３２】



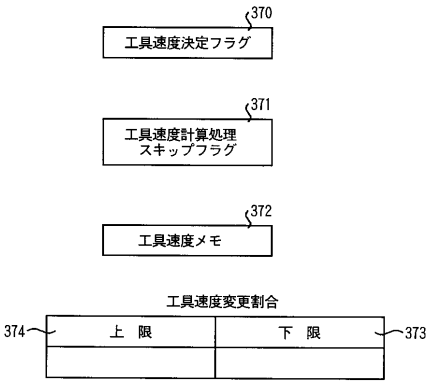
【図３３】



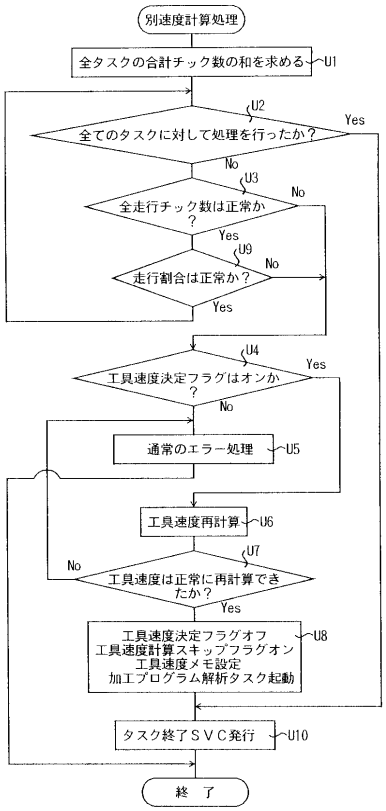
【図３４】



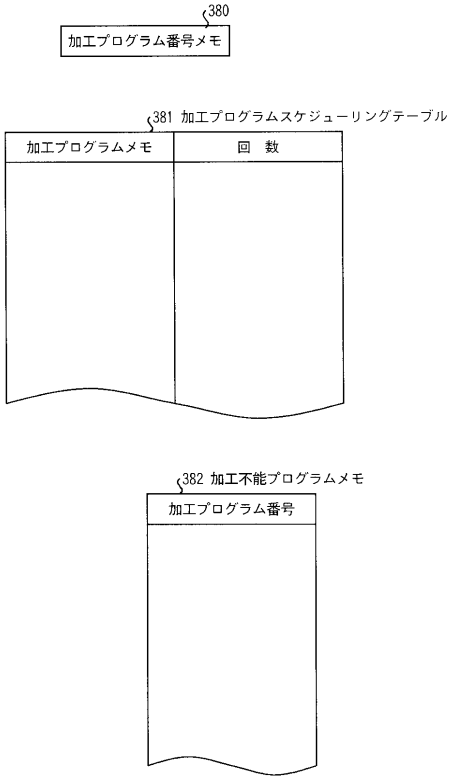
【図３５】



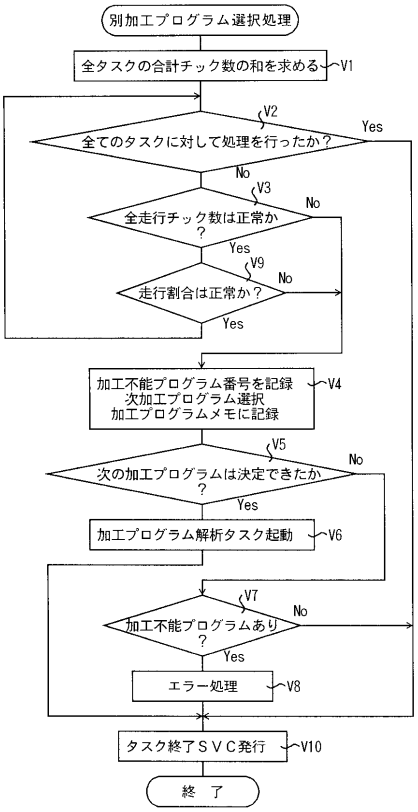
【図３６】



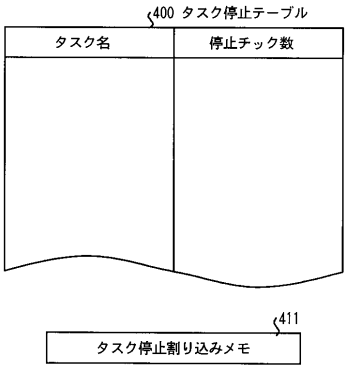
【図３７】



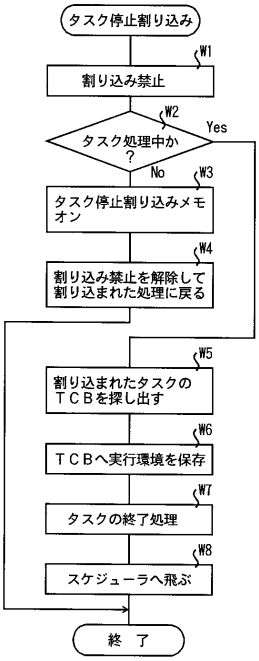
【図３８】



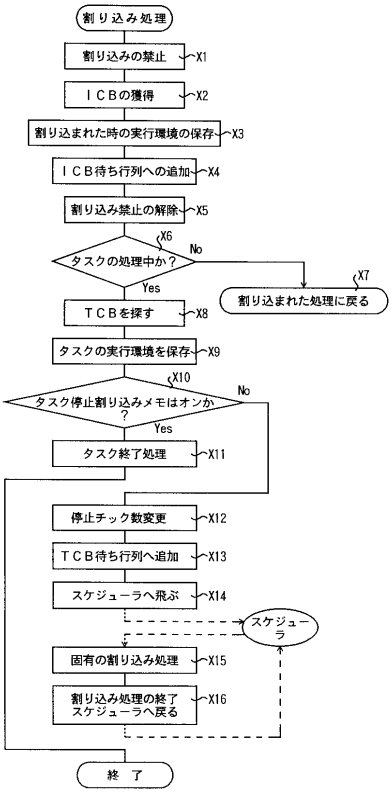
【図３９】



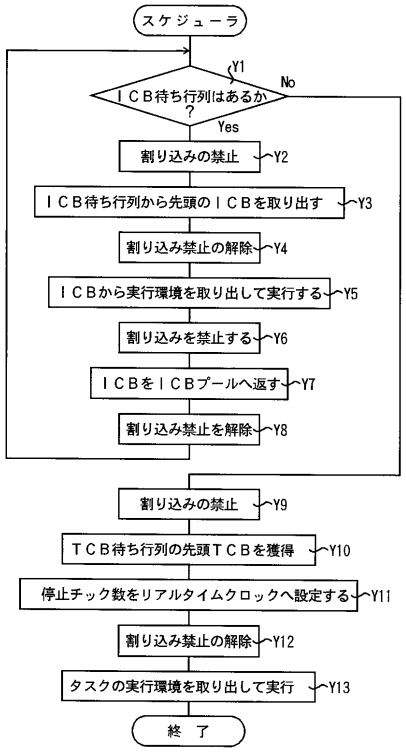
【図４０】



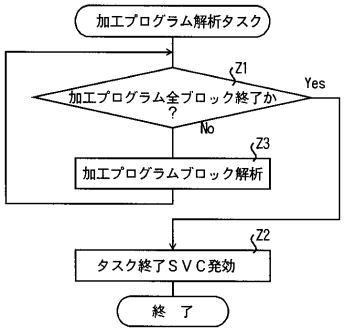
【図４１】



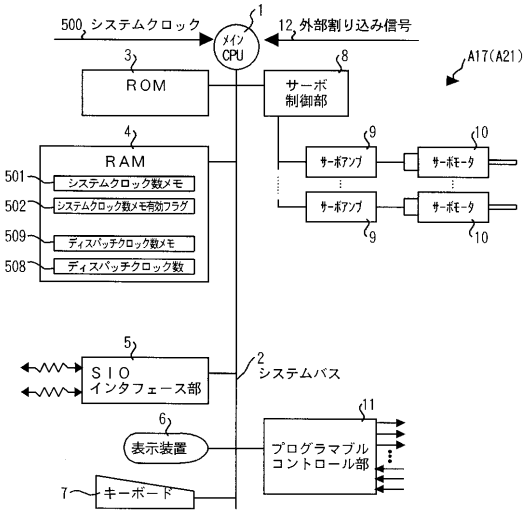
【図４２】



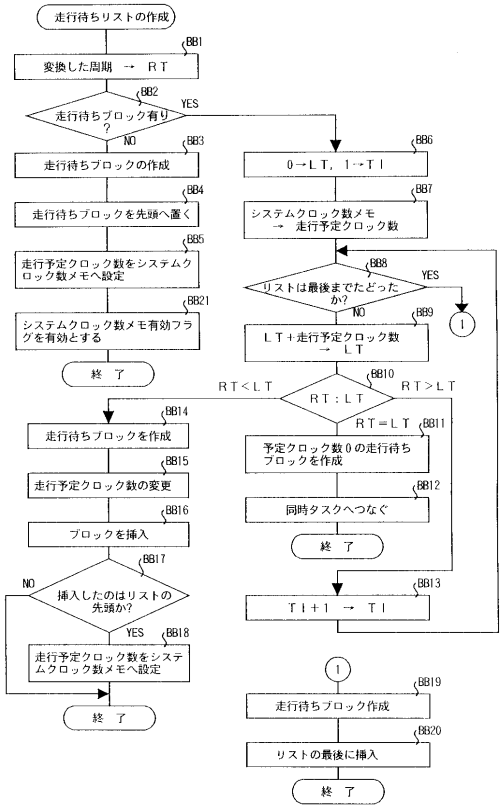
【図４３】



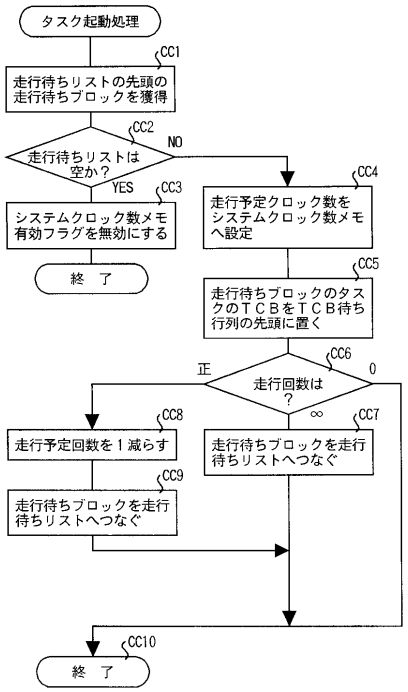
【図４４】



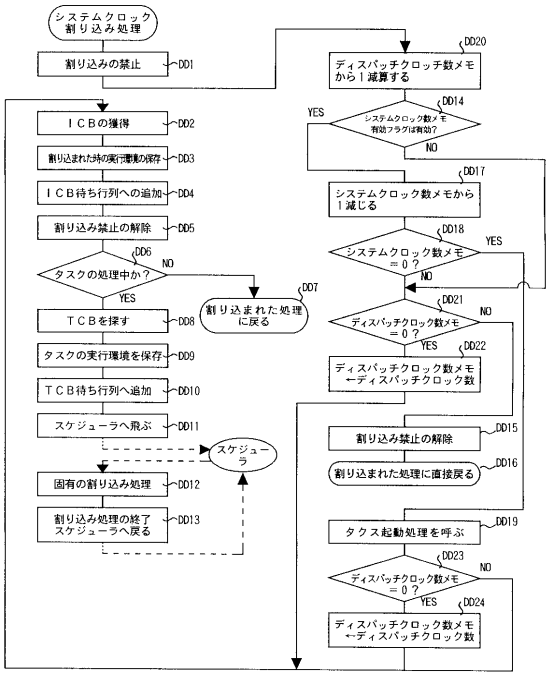
【図４５】



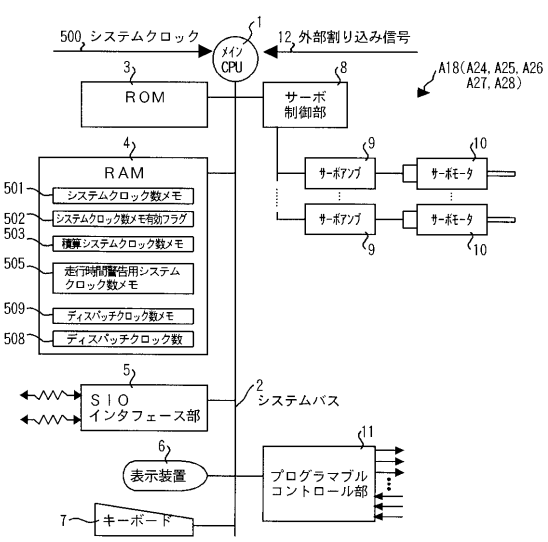
【図４６】



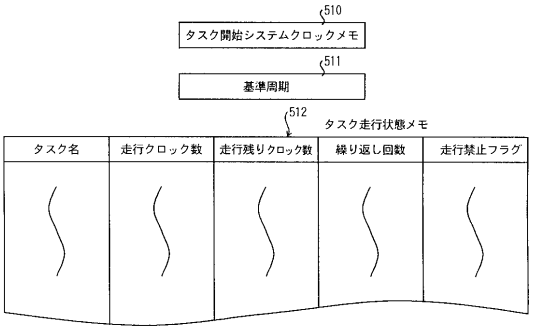
【図４７】



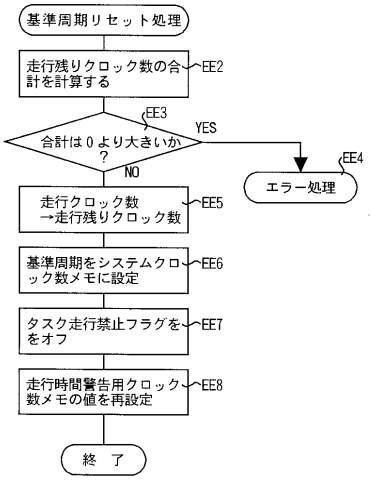
【図４８】



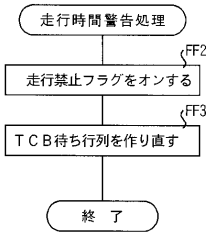
【図４９】



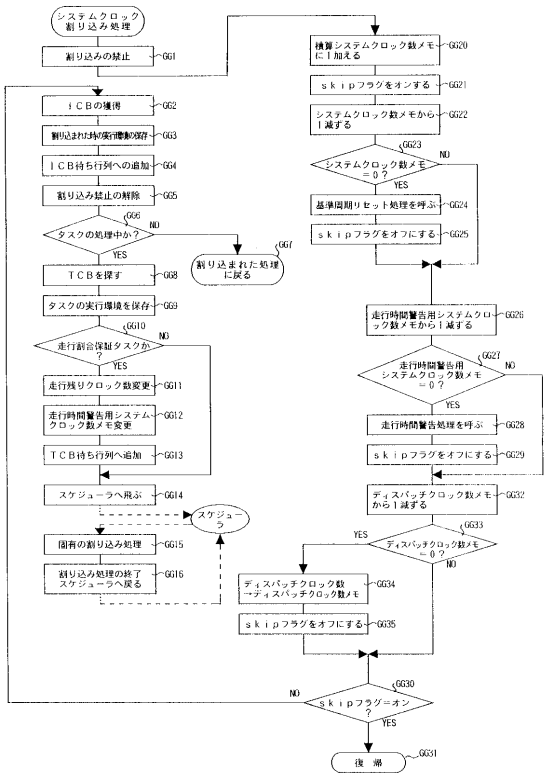
【図５０】



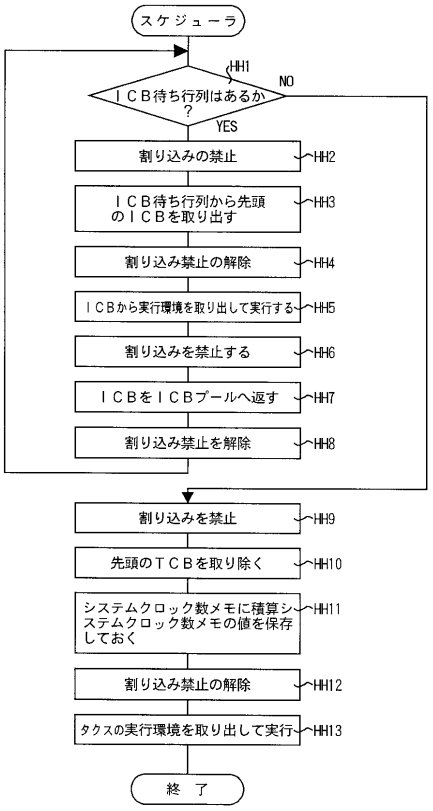
【図５１】



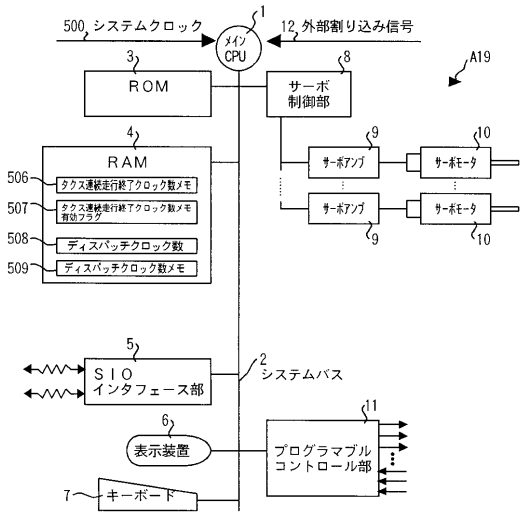
【図５２】



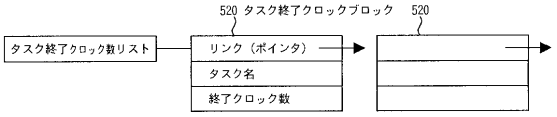
【図５３】



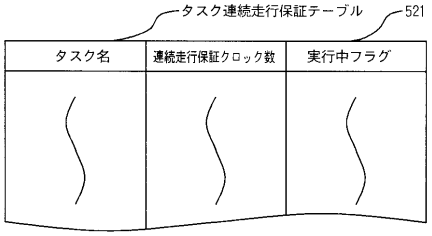
【図５４】



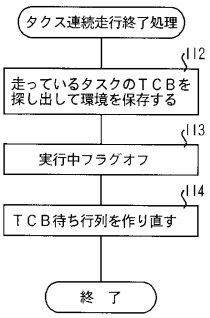
【図５５】



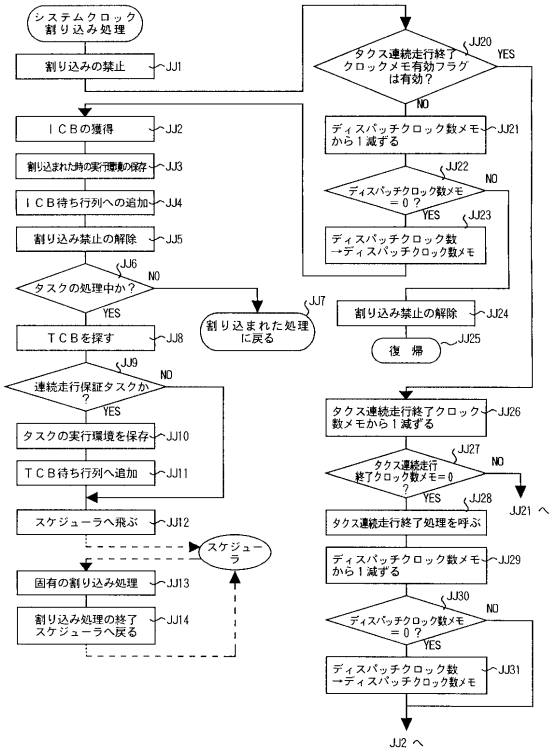
【図５６】



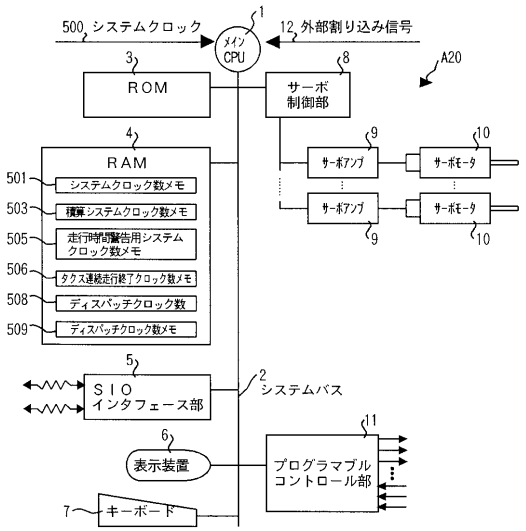
【図５７】



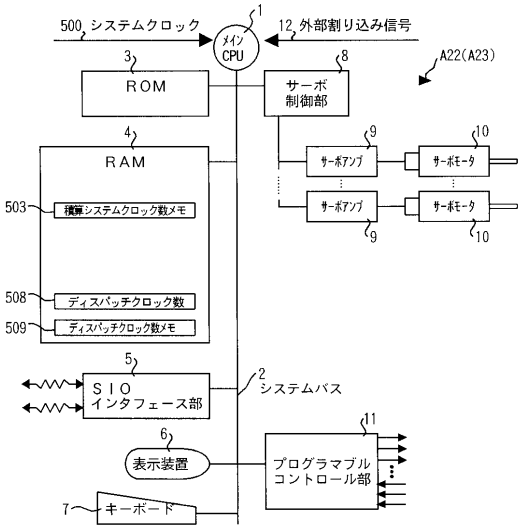
【図５８】



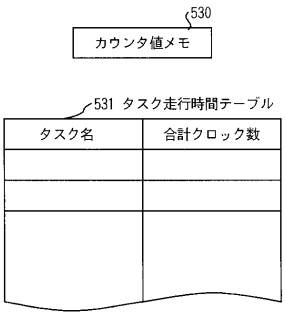
【図５９】



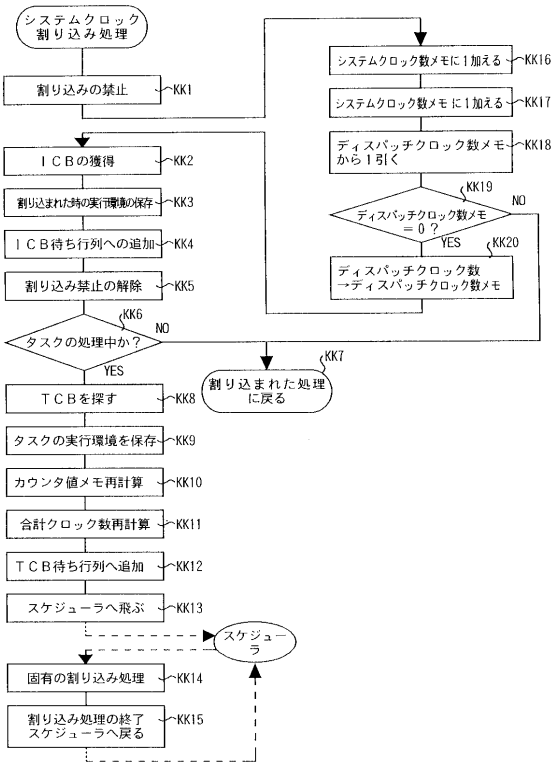
【図６０】



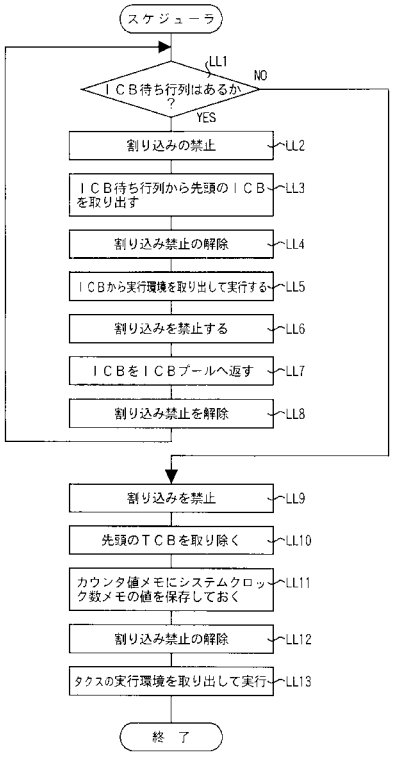
【図６１】



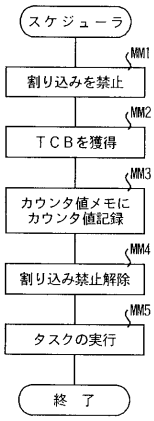
【図６２】



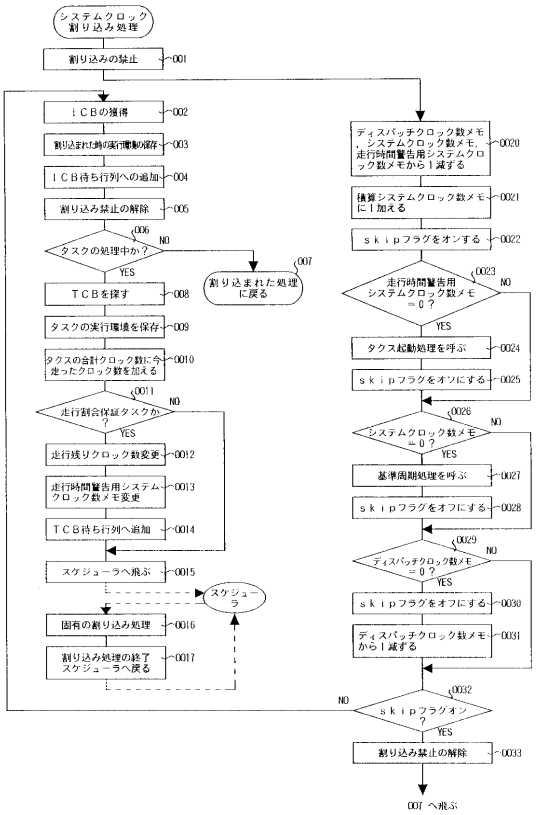
【図６３】



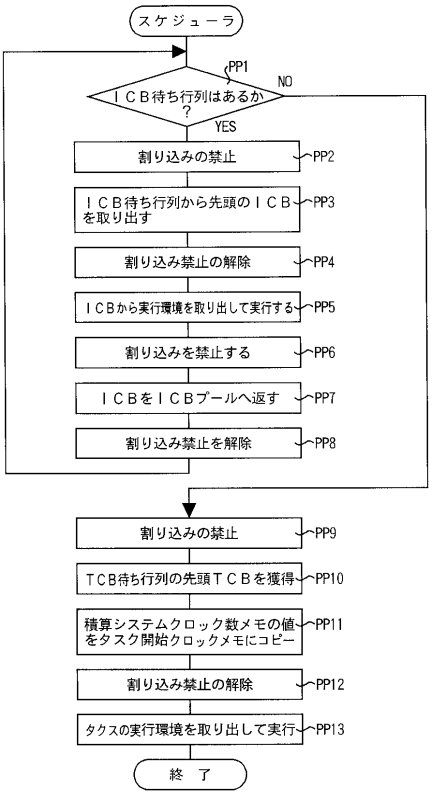
【図６４】



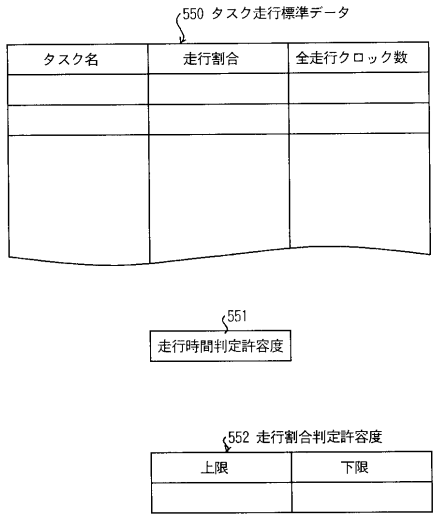
【図６５】



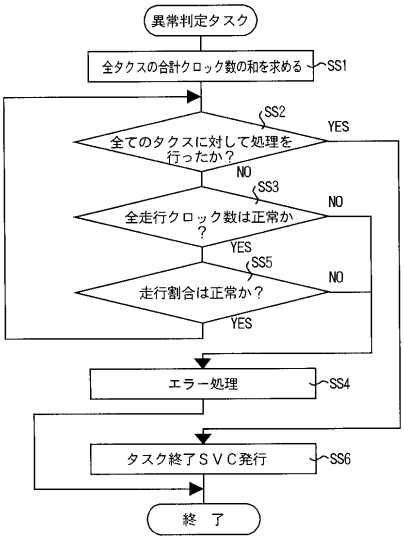
【図６６】



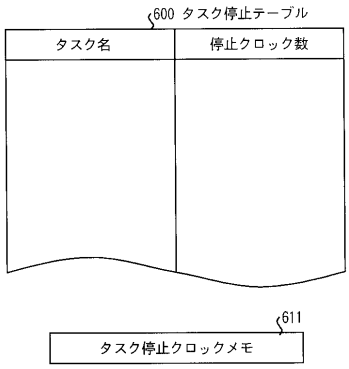
【図６７】



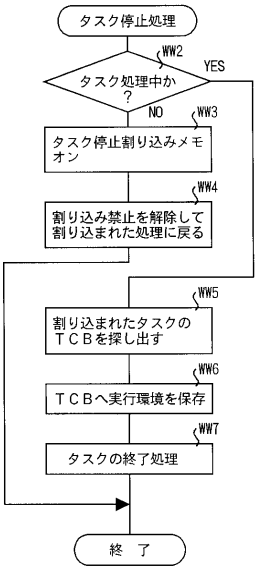
【図６８】



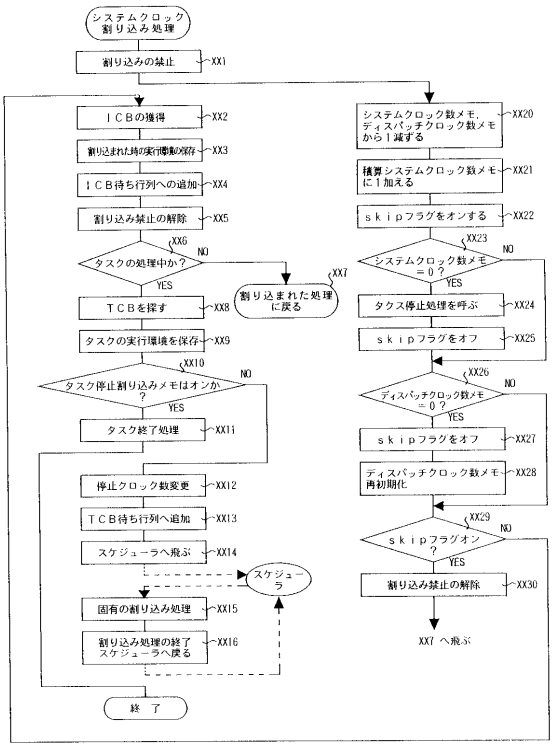
【図６９】



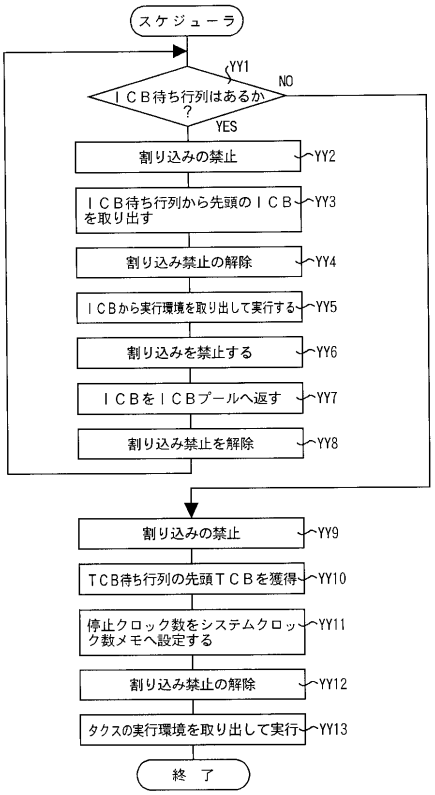
【図７０】



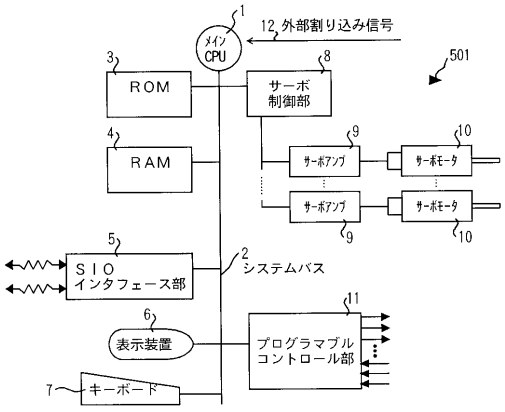
【図７１】



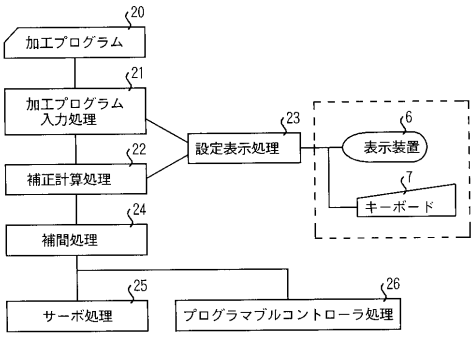
【図７２】



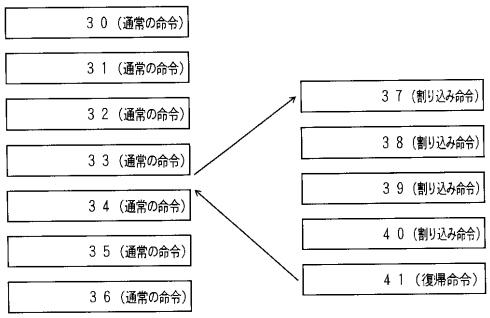
【図７３】



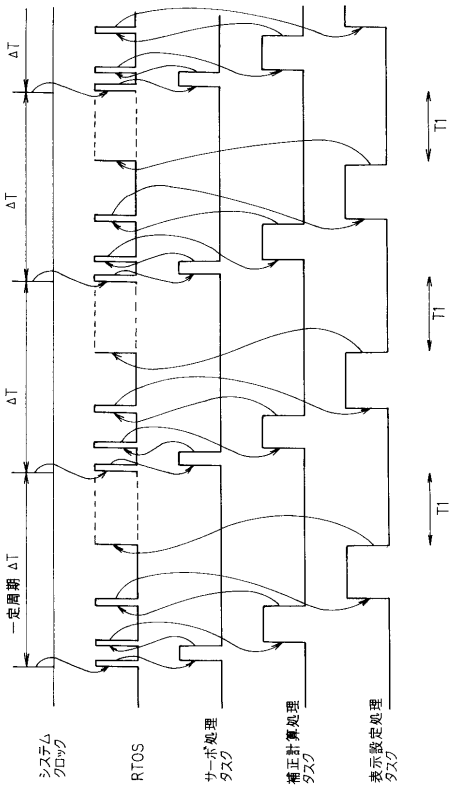
【図７４】



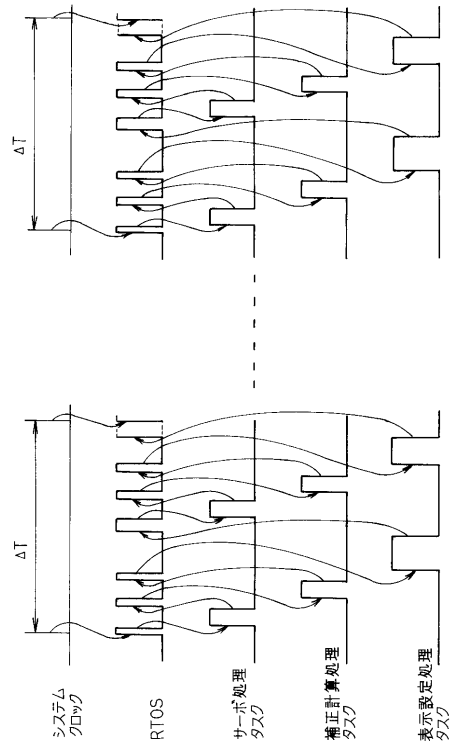
【図７５】



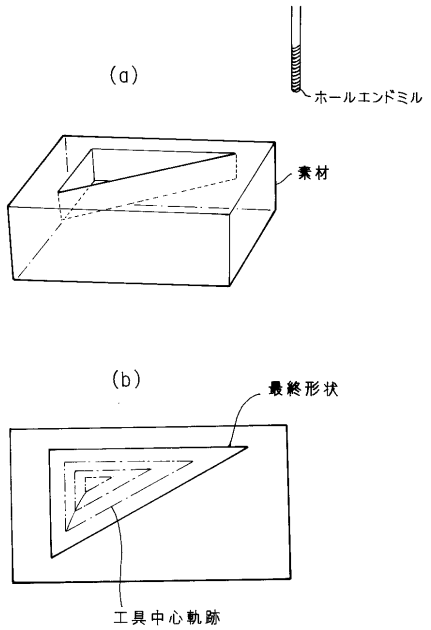
【図７６】



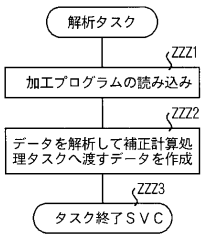
【図７７】



【図７８】



【図７９】


解析タスク
ZZZ1
加工プログラムの読み込み
ZZZ2
データを解析して補正計算処理タスクへ渡すデータを作成
ZZZ3
タスク終了ＳＶＣ

フロントページの続き

(56)参考文献 特開平０４－２６６１２０（ＪＰ，Ａ）
特開昭６２－２１２８４２（ＪＰ，Ａ）
特開平０６－１６８１４４（ＪＰ，Ａ）
特開平０４－１３４５３４（ＪＰ，Ａ）
特開平０４－３６２７４２（ＪＰ，Ａ）
特開昭６３－１５６２４７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F 9/46 - 9/54
G06B 19/02