リアルタイム Linux を利用した振動モータの遠隔制御システム

福本 隆浩 *1　　山田 健仁 *2　　百田 正広 *2

A Remote Control System of Vibration Motors using Real-Time Linux

Takahiro FUKUMOTO, Takehito YAMADA and Masahiro MOMOTA

Abstract

Synchronous control of vibration motors has been realized using a microcomputer (PC) with a real time operating system (RT-Linux). The vibration motors are AC motors with centrifugal weights, and consist of two axes. Vector inverters drive the vibration motors. Based on the digital PID control theory, the PC calculates the velocity reference value for control of the vector inverters. Moreover, various motor control states were realized by adding vibration mode (maximum, middle, zero and backward direction rotation). Use of AT compatible small-sized single board microcomputer achieved a miniaturization and rich functions of the PC system. The real time processing of digital control attained in the function of the RT-Linux. The RT-Linux is extended the function of the Linux operating system, so the RT-Linux could realize a remote control by using the network function in the Linux. In this research, communication control between two personal computers connected to LAN and Ethernet was realized. By using RT-Linux for one personal computer, remote control with the real-time operation became possible. The UNIX network socket on Linux is used for the communication control, and TCP/IP is used as the communication protocol.

Key Words : Synchronous Control, PID Control, Vibration Motor, Real-Time Linux, Network Socket, TCP/IP

1. まえがき

遠心重式振動機（以下振動モータ）は，一般的に生産現場の加振手段として使用されている．振動モータは振動数が高く，間欠運転に適している．また，設置や取り扱いが容易で，加振力調整が可能であるという特徴を持っている[1]．

実験対象の振動モータは 2 軸で構成されているもので，AC モータを 2 個用いることによって様々な振動を発生させることができる．しかし，2 個のモータの各軸の位相制御が適切でないと所望の振動を得ることはできない．

この制御をコンピュータによるディジタル制御で実現するが，最近では汎用のコンピュータシステムにフリーソフトのリアルタイム OS を利用した高精度なコンピュータによる機器制御が実用的になってきている．リアルタイム OS とは，処理をリアルタイムに実現することを重視し，そのための機能を実装した OS のことである．この機能を備えたものが Real Time Linux（以下 RT-Linux）である．

*1 情報電子工学専攻

*2 情報電子工学科

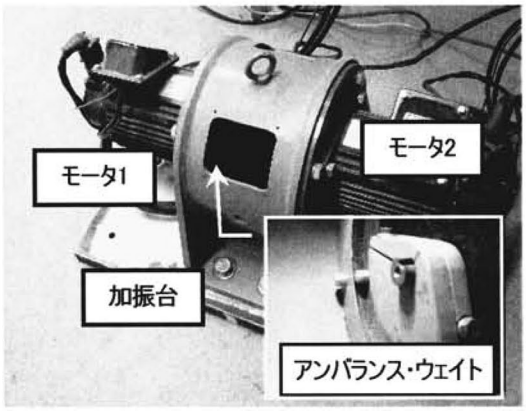

図1 振動用モータによる機構部の構成

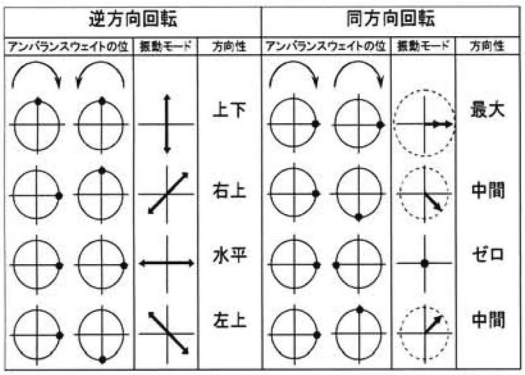

図2 アンバランス・ウェイトの位置関係による振動モード

過去の研究において，ワンボードマイコン（PC）とRT-Linuxを利用して2軸モータの基本的な同期制御を実現した（詳細については参考文献1を参照）．今回はさらに振動モードを追加することにより多様性のある2軸モータの同期制御システムを実現する．さらにLinux上のUNIXネットワークソケットを利用しコマンド入力によるサーバ，クライアント型の遠隔制御を実現する．

2. 振動モータの遠隔制御システム構成

2. 1 振動モータの機構部の構成

図1に機構部の構成を示す．遠心重式の振動モータでは，モータの軸部にアンバランス・ウェイトを取り付ける．モータを2個用いれば，アンバランス・ウェイトの位置関係や回転方向によって任意の振動を作り出すことができる．図2にモータの回転方向や各軸に取り付けられたアンバランス・ウェイトの位置関係によって任意の振動が発生する状態を示す．

表1 制御対象，遠隔制御コンピュータ仕様

制御対象側PC（AT互換機）	CPU：Cyrix Media GXtm 400MHz
	OS：Silicon-Linux Version 2.2.16
遠隔制御側PC（AT互換機）	CPU：AMD-K6 3D processor 233MHz
	OS：Redhat Linux Version 2.2.12
ネットワーク（イーサネット）	徳山高専学内LAN 通信速度：10 [Mbps]

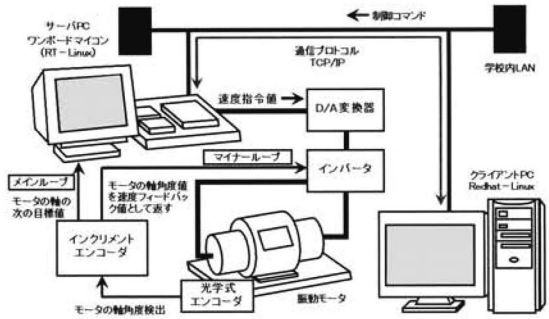

図3 2軸モータ遠隔制御システム構成

同期運転には比例要素（P要素）と積分要素（I要素）を備えたPI制御を用いる．また，速度フィードバック制御をかけることで，モータの一定回転速度（安定状態）を保つようにしている．

2. 2 システム構成

本研究で用いる制御対象側コンピュータと遠隔制御側コンピュータの仕様を表1に示す．また，2軸モータ遠隔制御システム構成を図3に示す．インバータはモータの軸に取り付けられた光学式エンコーダからの回転パルスを基に速度フィードバックをかけて制御系の安定化をはかっている．このループをマイナーループとし，メインループはPCのディジタル制御で実現する．メインループは光学式エンコーダからの回転パルスをインクリメントエンコーダで受けて，サンプリング時間毎にその値を読み取り，モータの軸角度（位置）の検出を行う．そして目標値（位置指令値）と軸角度の偏差を求め，偏差から目標速度を演算してD/A変換器でアナログ信号にした後，インバータに速度指令値として与える．

また，イーサネットで繋がったLAN上の2台のPC間で通信を行う．これにはLinux上のUNIXネットワークソケットを利用し，通信プロトコルとしてTCP/IPを使用している．クライアント側からコマンドを入力し，サーバ側でコマンドを受け取り，コマンドに応じて2軸モータの運転・停止・実験結果の表示などを行っている．

3. RT-Linuxによる実装

本研究では，モータの2軸同期制御システムをPI制御器によるディジタル制御器で実現する．また，そのディジタル制御器をプログラムに組み込む際にリアルタイム処理を備えているRT-Linuxを使用した[2]．

3. 1 RT-Linuxの特徴

RT-Linuxはフリーソフトウェアである．LinuxにRT-Linuxをインストールすることで，リアルタイム処理用のシステムが実現できる．LinuxとRT-Linuxのデータ処理の流れは，図4のような形になる．図中の点線で囲まれた部分がRT-Linuxで拡張される部分である．ハードウェアからの割り込みをRT-Linuxのリアルタイムカーネルで受け付け，Linuxまたはリアルタイムタスクに処理を渡す．リアルタイムタスクとLinuxのプロセス間の情報交換機能としてリアルタイムFIFO（First-In-First-Out）が設けられている．

この機能を用いれば，例えばリアルタイム性の必要な処理のみをRT-Linuxのリアルタイムプログラムとして動作させ，リアルタイム性の必要がない処理をLinux上で構成することによって，リアルタイムに処理しているデータをLANに流すようなシステムを作ることができる．

3. 2 リアルタイムプログラム

過去の研究では，図2に示す振動モードにおける同方向回転における方向性最大についてのみ実現した．今回はさらに同方向回転における方向性中間，ゼロ，逆方向回転における振動モードの追加を行った．

2軸モータ制御プログラムのフローチャートを図5に示す．全体的な流れは各振動モードで同じである．しかし，振動モードで各モータの位相差や回転方向が異なるために，軸角度や速度調整が困難になる．そこでプログラム上では各モードに回転方向，振動モードのフラグを設け，そのフラグの値によってエンコーダの判定基準値やモータの制御方法などを設定・変更するようにした．

例えば，同方向回転における方向性最大の同期制御では，それぞれのモータのエンコーダの値（以下パルス数）の差の判定基準値を0として判定を行っている．それに対して同方向回転における方向性ゼロについては，それぞれのモータの位相差を180度に保ちながら一定速度回転を行わなければならない．そこでモータ1回転（360度，1000パルス）に対し，パルス数の判定基準値を500にすることで，各モータの位相差を180度に保ちながら一定速度回転を実現することができる．

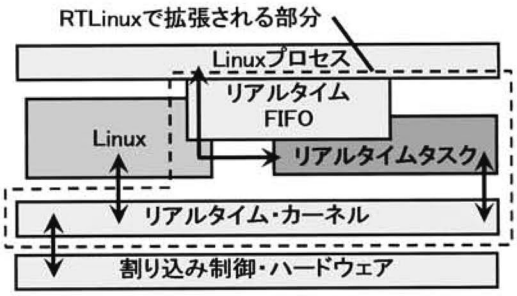

図4 RT-Linuxデータ処理フロー

逆方向回転に関しては，片方のモータを逆回転するために位相差は0～180度の範囲で変化する．その際にパルス数の差が500を超えてしまう場合が考えられ，モータが誤動作を起こしてしまう（パルス数の差が500パルス以内の場合は同方向回転の場合と同様）．それを改善するために，パルス数の差が500を超えた場合は最大パルス数（1000）を基準に以下のような処理を行い，判定パルス数を求めた．

① モータ1のパルス数 ＞ モータ2のパルス数
判定基準パルス数 ＝｜ 1000－モータ1のパルス数
＋モータ2のパルス数 ｜

② モータ2のパルス数 ＞ モータ1のパルス数
判定基準パルス数 ＝｜ 1000－モータ2のパルス数
＋モータ1のパルス数 ｜

ここで，｜｜は絶対値を示す．

このような方法により振動モードを変更した場合でも誤動作をすることなくそれぞれの振動モードを実現することができた．

4. 実験

実験では，ディジタル制御器におけるサンプリング周波数1kHz（サンプリング周期1 ms），目標回転速度420rpmで，2軸同期制御実験を行った．現在のパルス数と過去（1つ前）のパルス数と目標位置から偏差を求める．この偏差よりPI制御演算を行う．これで求まった値を速度指令値に変換し，D/A変換器に送る．この間にパルス数を測定しておき，制御実験後FIFOを介してLinux側へ観測データを転送し，実験値を評価する．評価としてそれぞれのパルス数の差から位相差を求め，各振動モードが実現できているかどうかについて検討した．また，振動モードについては同方向回転における振動モード最大に加え，同方向回転における振動モード中間，ゼロ，逆方向回転について実験を行った．

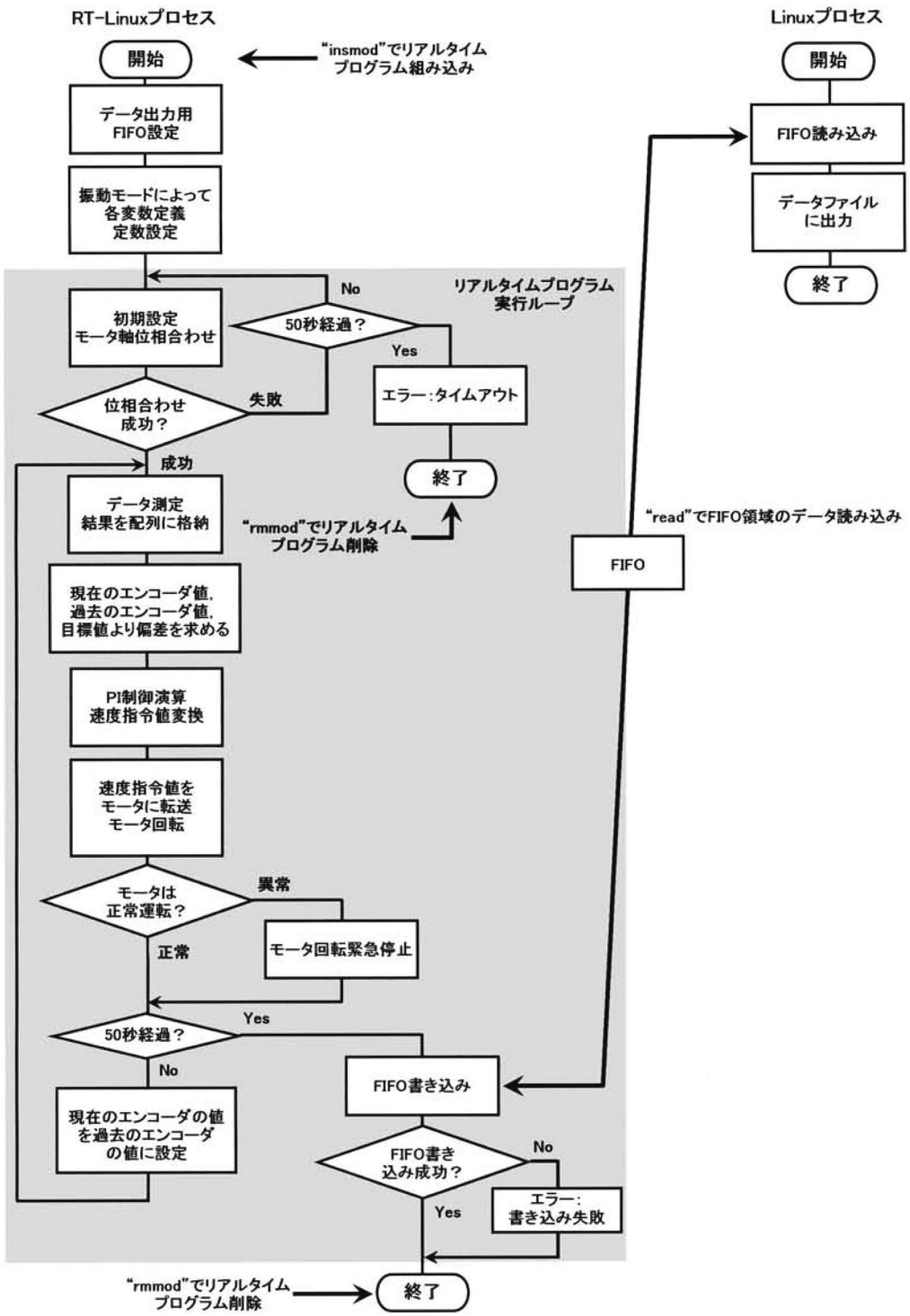

図5 2軸モータ制御プログラムのフローチャート

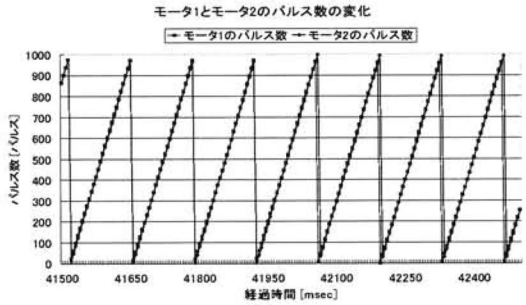

図6 振動モードが最大のときの2軸同期制御後の各モータのパルス数の変化

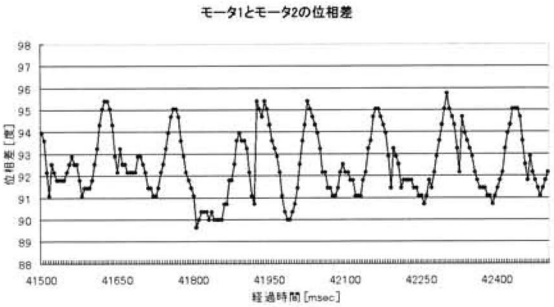

図9 振動モードが中間のときの2軸同期制御後のモータ間の位相差

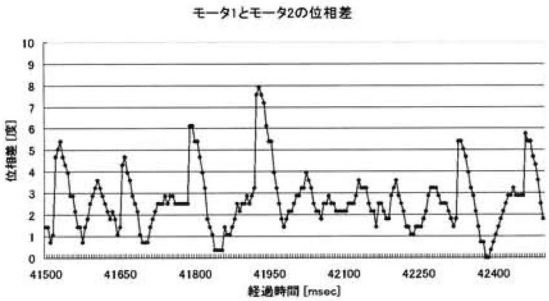

図7 振動モードが最大のときの2軸同期制御後のモータ間の位相差

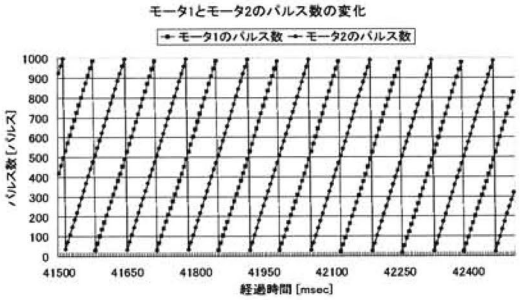

図10 振動モードがゼロのときの2軸同期制御後の各モータのパルス数の変化

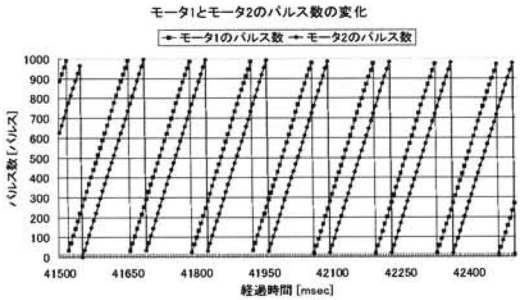

図8 振動モードが中間のときの2軸同期制御後の各モータのパルス数の変化

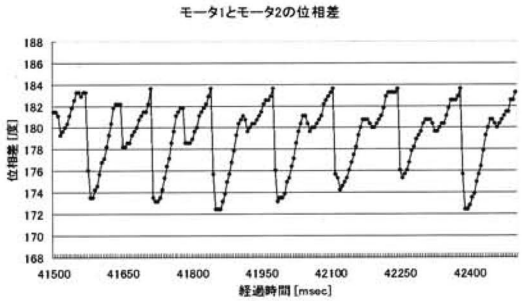

図11 振動モードがゼロのときの2軸同期制御後のモータ間の位相差

4. 1 同方向回転による2軸同期実験

2軸での同期制御を行うには，2つのモータの位相を合わせなければならない．そのために，最初に低速度回転で2つのモータ軸の角度合わせ（位相合わせ）を行い，それから2軸同期制御を行うと言う手法を採った．各モータのパルス数の差が5パルス以内になるまで低速度回転から位相を合わせ，その後，モータの目標速度まで回転速度を上げる．振動モード最大，中間，ゼロについての実験を行った．

振動モード最大のときの同期を取った後の結果について，図6，7に示す．図6は2軸同期後の各モータのパルス数の変化である．これを見るとそれぞれのモータのパルス数がほぼ一致していることから同期が取れていることが分かる．図7は，2軸同期後の各モータ間の位相差（軸角度のずれ）である．最大で約8度の位相誤差が生じているが，ほぼ同期を取って回転していることが分かる．

振動モード中間における同期を取った後の結果を図8,9に示す．図8は経過時間に伴う各モータのパルス数の変化である．先ほどの振動モード最大のときに比べる

と経過時間とともにそれぞれのパルス数の変化に一定の差が出ていることが分かる．これを元にそれぞれのパルス数の差を求め，位相差がどの程度出ているかを検討した結果が図9である．振動モード中間においては，位相差が90度であれば同期が取れていることになる．これより最大で約6度の位相誤差が生じているが，ほぼ90度の位相差を保って回転していることが分かる．

振動モードゼロにおける同期を取った後の結果を図10,11に示す．図10は各モータのパルス数の変化である．先ほどの振動モード最大のときに比べると経過時間とともにそれぞれのモータのパルス数の間に差が出ていることが分かる．これを元に位相差がどの程度出ているかを検討した結果が図11である．振動モードゼロにおいては，位相差が180度であれば同期が取れていることになる．これより最大で約8度の位相誤差が生じているが，ほぼ180度の位相差を保って回転していることが分かる．それぞれの振動モードにおける位相誤差の主な原因として，光学式エンコーダのスリット精度の問題が考えられる．

4. 2 逆方向回転による2軸同期実験

一方のモータを逆方向回転させた場合の同期実験について検討する．これについても，同方向回転と同様に最初に低速度回転で2つのモータ軸の角度合わせ（位相合わせ）を行い，それから2軸同期制御を行う．

同期を取った後の結果を図12，13に示す．図12は2軸同期後の各モータのエンコーダの値である．一方（モータ1）を逆方向回転しているので，モータ1のパルス数の変化は減少する方向になっている．これを見ると経過時間によるそれぞれのエンコーダの値に差が出ていることが分かる．図13はそれぞれのパルス数の差から位相差を求めた結果である．これを見ると一定の周期を保って位相差が0～180度変化していることが分かる．同方向回転のときと違って位相差は周期的に変化している．これより同期がとれていることが分かる．

5. 遠隔制御システム

本研究では，イーサネットで繋がったLAN上の2台のPC間での通信により遠隔制御を実現する．そのためにLinux上のUNIXネットワークソケットを利用し，通信プロトコルとしてTCP/IPを使用した．クライアント側（遠隔制御側PC），サーバ側（制御対象側PC）のプログラムのフローチャートを図14に示す．また，図15にサーバ側とクライアント側の通信状況を示す．この図はサーバ側プログラム起動後，クライアント側プログラム

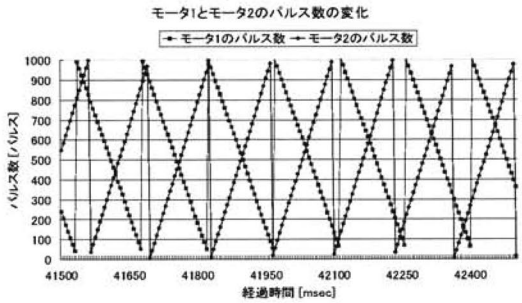

図12 逆方向回転2軸同期制御後の各モータのパルス数の変化

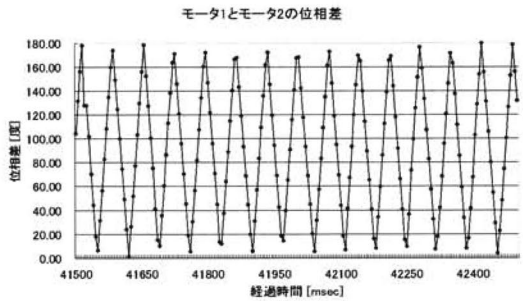

図13 逆方向回転2軸同期制御後のモータ間の位相差

を起動し，“start 1,420”，“quit”と入力した場合のパケットの流れを図示したものである．図15の番号は図14のフローチャートのどの処理をしているかを表している．

サーバ側では，`socket()`と`bind()`，`listen()`を実行するとコネクションを受け付けられるようになる．クライアント側が`socket()`，`connect()`を実行すると，コネクション確立のためのSYNセグメントが送信される．コネクション確立後，サーバ側で`accept()`が実行されると，コネクションを利用してホスト間で通信ができるようになり，サーバ側からコマンド入力促進用のプロンプトが送られてくる．クライアントは，受け取ったメッセージを画面に表示する．クライアント側の処理は，`select()`によるイベントドリブン型の処理になる．キー入力やメッセージの受信がなければ何も処理しない．

クライアント側でキーボードからの入力があると，そのメッセージがそのままサーバ側に送信される．サーバはメッセージを受信すると，それを解釈してそれに対応したコマンドを実行する．そして，コマンドの処理結果をクライアントに転送する．このコマンド処理は短時間で終わるために，通常はメッセージの送信と前のパケットの確認応答が1つのTCPセグメントで

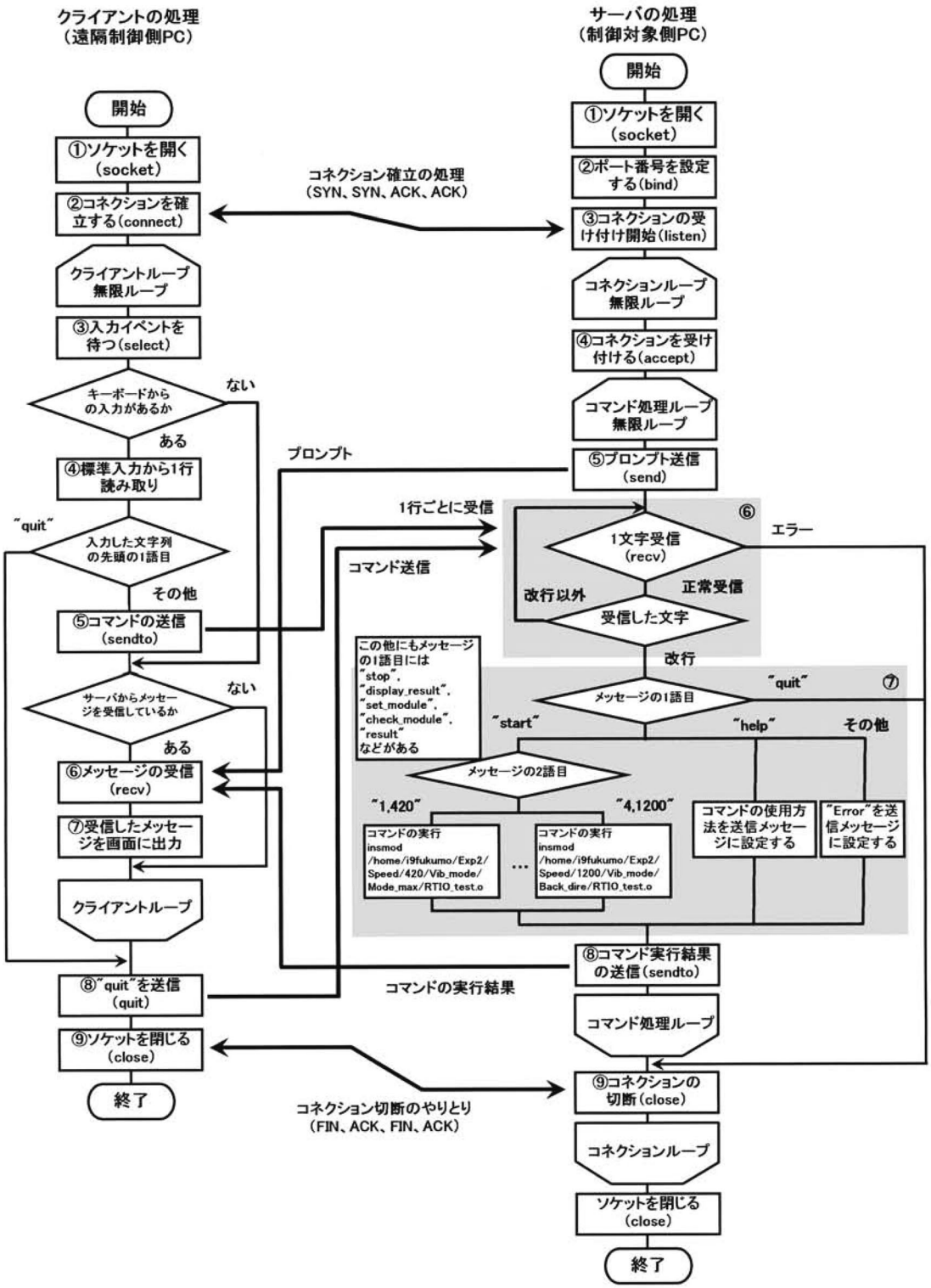

図14　クライアント側，サーバ側のプログラムのフローチャート

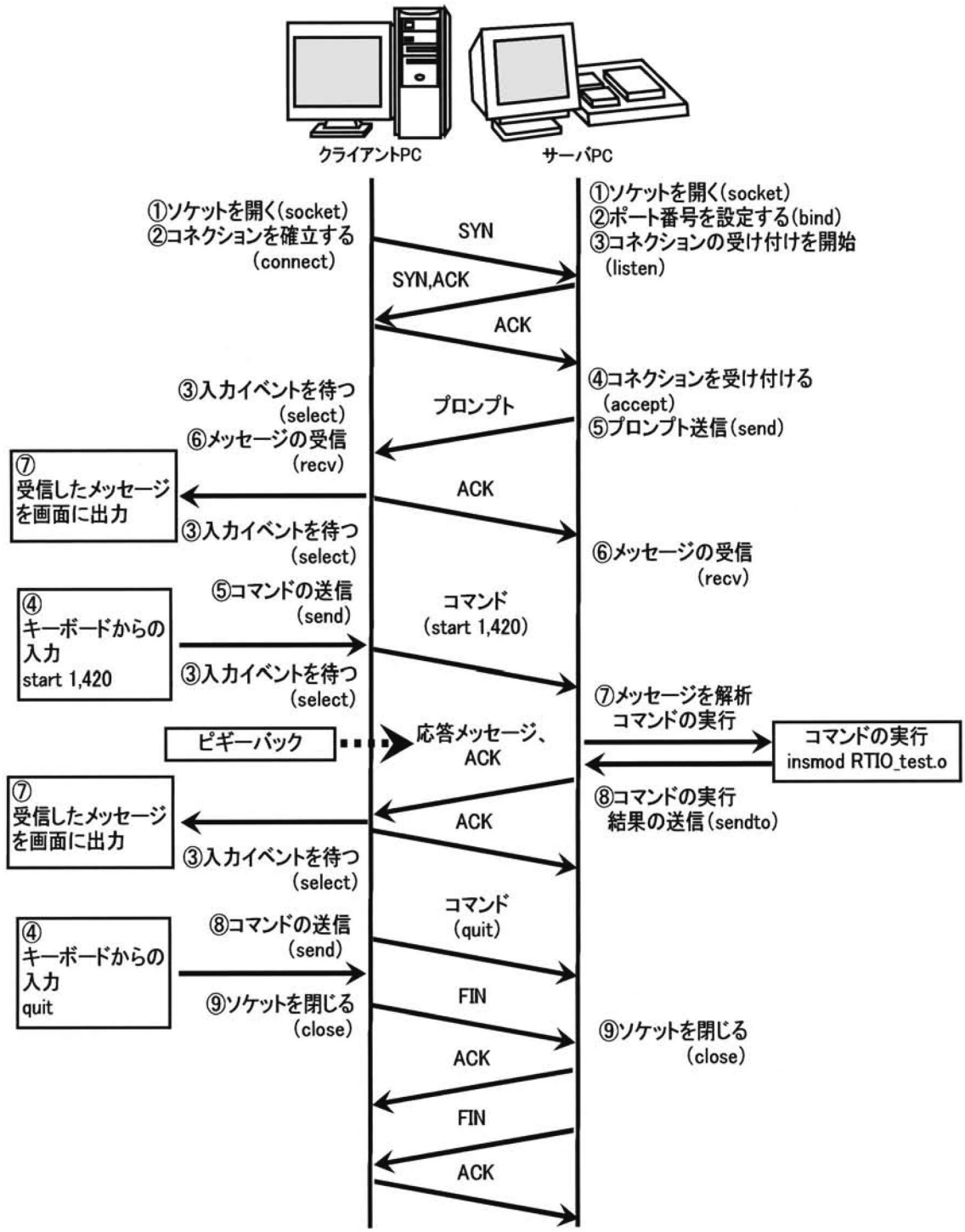

図15 サーバ側とクライアント側の通信状況

表2 使用できるコマンド一覧

コマンド	コマンドの意味
set_module	必要なモジュールの組み込み.
start 振動モード，回転速度	振動モード，回転速度を指定したモータ制御プログラムを実行. 振動モード：1（最大） 2（中間） 3（ゼロ） 4（逆方向回転） 回転速度：420，720，1200 [rpm]
stop	モータ制御プログラムモジュール削除（モータ停止）.
check_module	モータ制御プログラムなどの組み込まれているモジュールの確認.
result	モータ制御プログラムでの結果をサーバ側のファイルに出力.
display_result	実験結果ファイルを表示.
help	使用できるコマンドの表示.
quit	通信の終了.

行われる．これをピギーバックと呼ぶ．

モータを運転させる場合は，クライアント側から“start 1,420”というコマンドをサーバ側に送信する．サーバ側がそのコマンドを受信すると，振動モード“1”（最大），回転数420rpmのモータ制御プログラム（リアルタイムプログラム）のモジュールを組み込み実行する（モジュール組み込みコマンド：“insmod RTIO_test.o”）．実現した振動モードは，同方向回転における最大，中間，ゼロ，逆方向回転の4種類である．回転速度については，420rpm，720rpm，1200rpmの3種類である．それぞれの振動モード，回転速度が設定されたモータ制御プログラムが別々のディレクトリに格納されており，コマンドによって専用のプログラムを実行するようにしている．

モータを停止させる場合は，クライアント側から“stop”というコマンドをサーバ側に送信する．サーバ側がそのコマンドを受信すると，組み込んだモータ制御プログラムのモジュールを削除し，モータを停止する（モジュール削除コマンド：“rmmod RTIO_test”）．このようにサーバ側（RT-Linux側）では，モジュールの組み込みと削除によってモータの運転，停止を行っている．これによって，振動モードや回転速度の変更がそれぞれのモータ制御プログラムのモジュールの組み込み

図16 クライアント側プログラム実行画面

み，削除で容易に行うことができる．遠隔制御で使用できるコマンド一覧を表2に示す．

クライアント側で“quit”と入力すると，クライアントは“quit”という文字列をサーバへ送り，closeを実行する．closeが実行されると，TCPのコネクションが切断される[3]．

5. 1 遠隔制御実験

遠隔制御実験では，クライアント側からコマンドを入力し，そのコマンドをサーバ側で受け取ったときの実行状態について調べた．目標回転速度420rpmで，同

方向回転による方向性最大のモータ制御プログラムを実行し，その後にモータの測定結果ファイルを表示する“display_result”コマンドを入力した場合の結果を図16に示す．クライアント側よりコマンドを入力するとそのコマンドがサーバ側に送信される（図16の①）．サーバ側ではそのコマンドを解析し，クライアント側から送信されたコマンド内容をディスプレイ上に表示する（図16の②）．次にそのコマンドに応じた処理を実行して，その出力結果をバッファに格納する処理を行う．最後にこのバッファの内容をクライアント側に送信し，クライアント側ではバッファの内容をディスプレイ上に表示する（図16の③）．実験結果のファイル内容は左から経過時間[msec]，モータ1のパルス数，モータ2のパルス数，モータ1の回転速度，モータ2の回転速度，パルス差，位相差の順に示している．

6. まとめ

本研究では，RT-Linuxを用いた振動モータの制御性能に関して実験を行った．RT-Linux下のディジタルPI制御器を用いての一定回転速度および2軸同期の実現性を検討した．

メインループにおいてディジタルPI制御器を構成し，モータの一定速度回転を実現した．また，データ測定としてRT-LinuxシステムのFIFO領域への書き込み，Linux側からのFIFO領域への読み込み，データのファイルへの書き込みなどを実現し，データの測定環境を確立した．実験の結果，各振動モードを通して最大2%の位相誤差が生じていることが分かった．今後はこの誤差が実際の振動においてどれくらい影響があるのかについて検討していく．

また，TCP/IPを通信プロトコルとした遠隔制御を実現し，クライアント側からコマンドによるモータの運転，停止などが可能となった．これには，RT-Linuxの特徴の1つであるモジュールの組み込み，削除を併用することで実現した．これにより振動モード，回転速度を変えることも可能となった．現在は，回転速度のモードが3種類しか実現できていないので，回転速度のモードを増やすことが課題である．

文　献

1) 福本 隆浩，山田 健仁，百田 正広：“リアルタイムLinuxを利用した振動モータの同期制御”，平成14年度徳山工業高等専門学校研究紀要，第26号，PP.61-66（2002）
2) 森友 一朗，薬師 輝久，馬場 秀忠：“RTLinuxリアルタイム処理プログラミングハンドブック”，（株式会社）秀和システム，PP.105-125（2000）
3) 村山 公保：“基礎からわかるTCP/IPネットワーク実験プログラミング”，オーム社開発局，PP.150-167（2001）

（2003.9.2 受理）