

# Linux 搭載プロトコル変換器の開発と Web リモートモニタシステムへの適用

奥田 篤, 波 通隆

Development of Network Protocol Converter using Embedded Linux,  
and application to Web Remote Monitoring System.

Atsushi OKUDA, Michitaka NAMI

## 抄 録

組み込み用 Linux を搭載した CAN-TCP/IP 通信プロトコル変換器を開発した。このプロトコル変換器を用いて、WWW を通信環境に利用して CAN バスに接続した制御ノードの振る舞いを遠隔から監視できるリモートモニタシステムを構築した。

**キーワード：**CAN , TCP/IP , 組み込み Linux , プロトコル変換 , WWW , リモートモニタ

## Abstract

A communication protocol converter that exchanges CAN and TCP/IP packets was developed using a embedded linux package as operating system. And a web-based monitoring system that can be used to observe the behavior of control nodes that are connected to a CAN bus was constructed using the converter.

**KEY-WORDS :** CAN, TCP/IP, Embedded Linux, Protocol Conversion, WWW, Remote Monitor

## 1 . はじめに

近年の農業機械・畜産機械は自動化・高機能化が進められ、急速に電子制御システムへと移行している。それに伴い、多数搭載されるようになったセンサやコントロールユニットなどを効率良く統合する為の制御ネットワークの構築が技術的課題と成っている<sup>1,2)</sup>。その様なことから、自動車の車載ネットワークで実績のある Controller Area Network (CAN) が注目されており、欧米では農業機械・畜産機械にも採用され始めている。CAN に基づく上位通信規格についても開発・標準化が進められており、今後、農業機械のデータ通信ネットワークは CAN が標準になると考えられる<sup>3)</sup>。

また、コンピュータの著しい普及に伴い、農業・畜産業経営のコンピュータ化が進んでおり、農業機械・畜産機械も家屋などに配置されたコンピュータとの通信が行えることが求められている。特に近年のインターネットの浸透により事実上の標準ネットワークとなった Ethernet と TCP/IP (Transport Control Protocol/Internet Protocol) プロトコルに基づくネットワークとの親和性を持つことが重要視されている。

従って、本道の農業機械・畜産機械メーカーも、CAN を利用した制御ネットワークの構築や、ローカルな制御ネットワークと TCP/IP ネットワークとの接続による分散型制御ネットワークの構築に関わる技術の蓄積を図ることが求められている。

このような状況を踏まえて、我々は制御ネットワークの広域分散化に関する技術開発を進めてきた。その一環として組み込み用 Linux を搭載した CAN-TCP/IP 通信プロトコル変換

事業名：一般試験研究

課題名：制御用高速ネットワークの構築および広域ネットワークとの統合に関する技術の開発

器を開発した。さらに、プロトコル変換器を用いて、WWW（World Wide Web）を通信環境に利用してCANバスに接続した制御ノードの振る舞いを遠隔から監視できるリモートモニタシステムの構築を試みた。本報では、開発したプロトコル変換器とリモートモニタシステムへの適用事例の詳細について紹介する。

## 2. CAN-TCP/IP 通信プロトコル変換器の開発

### 2.1 プロトコル変換器の概要

開発したプロトコル変換器（図1）は、CANバスとEthernetへのインタフェースを備えるハードウェア、ハードウェア上で動作するオペレーティングシステム（OS）、CANデバイスを制御するためのデバイスドライバ、OS上で動作しCANパケットとTCP/IPパケットの交換を行うプロトコル変換ソフトウェアから構成した（図2）。それぞれの詳細については以下に述べる。

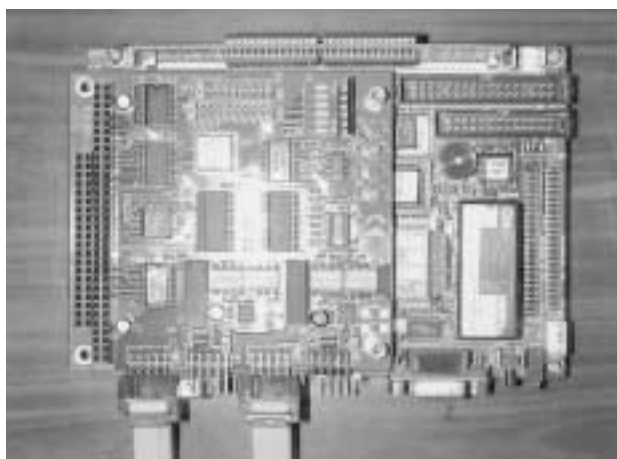


図1 プロトコル変換器の外観

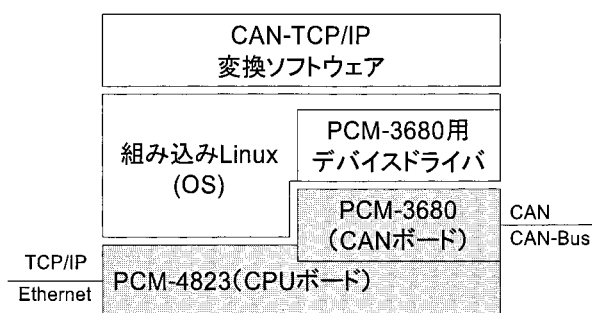


図2 プロトコル変換器の構成

### 2.2 ハードウェア

Ethernetコントローラを有するCPUボード（Advantech社PCM-4823）とCANボード（Advantech社PCM-3680）をPC/104バスで接続してプロトコル変換器のハードウェアを構成した。それぞれのボードの仕様を表1、表2に示す。

表1 CPUボードのスペック 表2 CANボードのスペック

| PCM-4823 (Advantech) |                       |
|----------------------|-----------------------|
| CPU                  | AMD DX5-133           |
| ROM (Flashメモリ)       | DiskOnChip 2000 8MB   |
| RAM                  | 8MB                   |
| Ethernet コントローラ      | RTL 8019AS (NE2000互換) |
| 外部バス                 | PC/104                |

| PCM-3680 (Advantech) |                |
|----------------------|----------------|
| ポート数                 | 2              |
| CANコントローラ            | Philips 82C200 |
| CANトランシーバ            | Philips 82C250 |
| 外部バス                 | PC/104         |

CPUボードには、記憶領域として8MバイトのFlash ROM（M-Systems社DiskOnChip 2000）と8MバイトのRAMを搭載した。

### 2.3 オペレーティングシステム

#### 2.3.1 組み込みシステム向けOSとしてのLinux

Linuxは、マルチタスク、仮想メモリ、共有ライブラリ、デマンドローディング、メモリ管理などの機能を含んだUNIXクローンのOSであり、1991年に原型が開発されて以来10年以上にわたり修正・拡張が続けられてきており、現在ではネットワークサーバ分野で広く用いられている。もともとデスクトップ環境向けに開発されたLinuxだが、カーネルやシステム構成により、ファイルシステムをROM/RAM上に構築したディスクレスシステムや、loginコンソールなどを必要としないヘッドレスシステムとしても使用できるため、組み込みシステムのOSとして利用することも可能である<sup>4,5)</sup>。

Linuxには、

- ・ソースが入手可能
- ・ロイヤリティフリー
- ・デバイスドライバが豊富
- ・ネットワークプロトコルスタックやミドルウェアが豊富
- ・動作が安定している

などの特徴があることから、商用リアルタイムOSなどを利用する場合に比べて、システム開発の期間やコストを著しく低減させることができる。

そのため、多くのメーカーにより組み込み用途向けLinux（組み込みLinux）の試作・製品化への検討、各種製品への搭載に向けた開発が活発に進められている。今後、ネットワークサーバ分野での利用に加えて、組み込みシステム分野での利用にも急速に伸びていくと考えられている<sup>4,5)</sup>。

#### 2.3.2 組み込みLinuxパッケージの作成

Linuxカーネル2.0.35に基づくLinuxディストリビューションSlackware-3.6を組み込み（ROM化）用途向けに改変して、プロトコル変換器へ搭載する組み込みLinuxパッケージを作成した。

パッケージは次の手順で作成した。

- 1) Linux カーネルソースの入手
- 2) 開発ツール・環境の整備
- 3) イメージローダの作成
- 4) Linux カーネルのコンフィグレーションとビルド
- 5) デバイスドライバの開発
- 6) ルートファイルシステム（ユーザランド）の構築
- 7) アプリケーションの開発
- 8) ハードウェアへの搭載（ROM 化）

パッケージの作成に際して、前述したハードウェアにハードディスクドライブ（HDD）を接続して Slackware-3.6 をインストールし、開発環境として利用した。必要に応じて、HDD 中にインストールされている Linux を起動して作業に供した。

このパッケージをオペレーティングシステムとしてプロトコル変換器のハードウェアに搭載した。パッケージの概要とハードウェアへの搭載手順の詳細を以下に述べる。

### 2.3.3 作成したパッケージの概要

開発したパッケージの大きさは、ユーザランドおよびアプリケーション（後述するプロトコル変換ソフトウェアなど）も含めて、約 6 M バイトである。ファイルシステムは標準 Linux ファイルシステム（Ext2）を採用した。

ユーザランドの構成を図 3 に示す。このユーザランドでは、シェルとして bash（Bourne-Again SHell）を採用した。

DiskOnChip 2000（DOC）を認識して DOC 中からコードをロードできる様に Linux Loader（LILO）を改変し、電源投入時にパッケージ中の Linux カーネルを起動するブートローダとして利用した。

ブート手順は、初期 RAM ディスク（Initial RAM disk; INITRD<sup>6)</sup>）を経由するシーケンスを採用した。

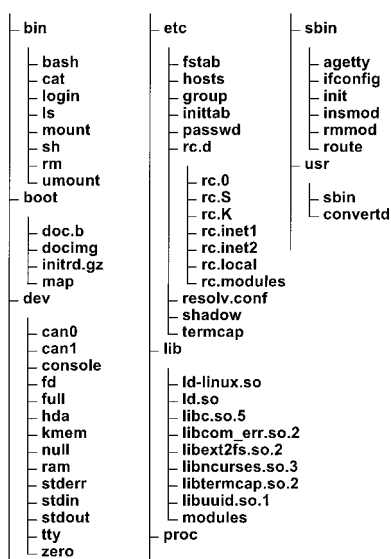


図3 ルートファイルシステムの構成（抜粋）

### 2.3.4 ハードウェアへのパッケージの搭載

プロトコル変換器ハードウェアへの組み込み Linux パッケージの搭載は以下の手順で行った。ハードウェアに HDD を接続して Slackware-3.6 をインストールし、開発用の環境として利用した。HDD から Linux を起動して作業を行った。

まず、作業用 Linux に DOC を HDD 互換の記憶デバイスとして認識させるために、DOC 用 Flash ROM ドライバ（M-Systems 社 TrueFFS ドライバ<sup>7)</sup>）を読み込み可能なカーネルモジュールロードブルモジュールとしてビルドする。得られたロードブルモジュールを実行中の Linux カーネルへ組み込んでカーネルに DOC を認識させる。これ以降、作業用 Linux から DOC を操作できる。そこで、ファイルシステム中の適当な場所へ DOC をマウントし、DOC 上へ Linux パーティションを作成する。次いで、この Linux パーティションにカーネルとユーザランドをコピーする。

ブート時に必要となる INITRD を作成するために、RAM ディスクなどの記憶デバイスを一時的に用意し、その中に DOC 用ロードブルモジュールとロードブルモジュールインストーラを含む最小限のルートパーティションを構築する。このパーティションを、DOC 上の Linux パーティションへファイルとして書き出し圧縮する。得られた圧縮ファイルを INITRD として利用する。

最後に、DOC のブートセクタにブートローダを書き込み、作業 Linux を終了する。ハードウェアから HDD を取り外し、DOC から組み込み Linux を起動する。

組み込み Linux は次の手順に従って起動する。ブートローダが、カーネルと INITRD を RAM 上へ読み込み、制御をカーネルへ移行する。次いで、カーネルが INITRD を RAM 上に展開する。展開した INITRD をルートパーティションとしてファイルシステムのルートポイントへマウントし、ロードブルモジュールインストーラを起動して DOC 用ロードブルモジュールをカーネルに組み込む。INITRD をアンマウントして、本来のルーパーティションである DOC をルートポイントにマウントする。引き続き、所定のブート手順に必要なスクリプトが順次が実行されて、組み込み Linux が立ち上がる。

## 2.4 デバイスドライバ

CAN デバイスドライバは、キャラクタ型のロードブルモジュールとして実装した。開発したドライバは、CAN 規格を満足する様に、同規格で定められている以下の機能を提供できる。

- ・データフレームの送受信
- ・リモートフレームの処理
- ・受信フィルタ
- ・エラーカウント
- ・オーバーランカウント

ドライバが動作対象としている CAN コントローラは Phi-

lips 社82C200であり、ポート I/O とメモリマップド I/O の何れのアーキテクチャで設計されたハードウェアにも対応する様に実装した。

デバイスドライバは、モジュール初期化関数、モジュール終了処理関数、システムコール群、割り込み処理関数から構成している。システムコールは、Linux デバイスドライバの標準的なプログラミングインタフェース<sup>8)</sup>に従って、open、close、read、write、ioctl の各関数を実装した。

CAN バスの両端に、搭載している OS にデバイスドライバを組み入れたプロトコル変換器ハードウェアと、CAN プロトコルアナライザ( Vector 社 CANalyzer 2.5 )を接続して、デバイスドライバの動作試験を行った( 図 4 )。デバイスドライバを利用する試験用ソフトウェア作成し、これをハードウェア上で動作させて試験を行った。プロトコルアナライザとの間で、バス上の通信量( バスレート)が幅広く変化するように送出データ量を変化させながら、種々の通信速度で、パケットの送信および受信を試みた。その結果、パケット受信時・送信時ともに CAN コントローラが CAN 規格を満足するように制御されており、規格上限である 1 Mbps まで良好に通信できることを確認した( 表 3、表 4 )。

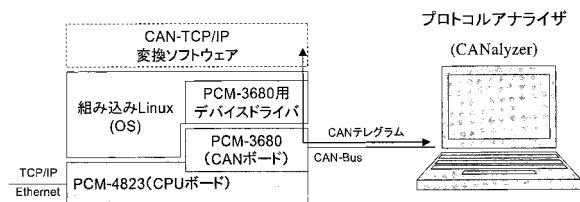


図 4 CAN デバイスドライバの動作試験方法

表 3 CAN デバイスドライバの動作試験結果(パケット受信)

| ボーレート<br>(kbps) | エラーフレーム数<br>バス負荷 |     |
|-----------------|------------------|-----|
|                 | 7%               | 70% |
| 125             | 0                | 0   |
| 500             | 0                | 0   |
| 1000            | 0                | 0   |

表 4 CAN デバイスドライバの動作試験結果(パケット送信)

| ボーレート<br>(kbps) | エラーフレーム数<br>バス負荷 |     |
|-----------------|------------------|-----|
|                 | 7%               | 70% |
| 125             | 0                | 0   |
| 500             | 0                | 0   |
| 1000            | 0                | 0   |

2.5 プロトコル変換ソフトウェア

プロトコル変換ソフトウェアは、CAN プロトコルのインタラクション層と TCP/IP プロトコルのトランスポート層との間でパケット交換を行う様に実装した( 図 5 )。TCP/IP プロトコル側には、表に示す機能が提供されるので、望みのソケットタイプと I/O モデルを組み合わせ使用できる。

データグラム用ソケットと多重化 I/O モデルを利用した場合の、プロトコル変換ソフトウェアの動作フローを図 6 に示す。ソフトウェアが起動されると、ソケットを生成し、ソ

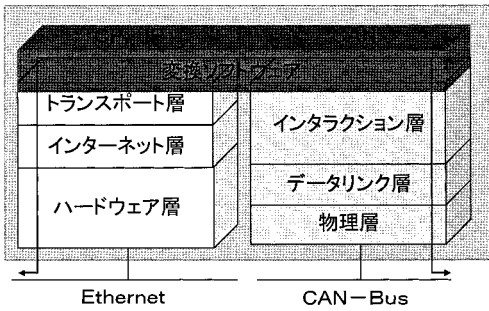


図 5 プロトコル変換ソフトウェアの概略

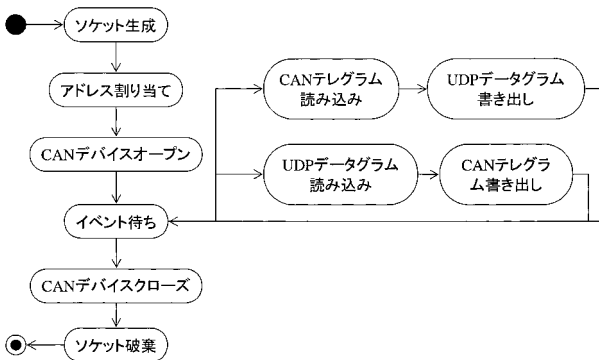


図 6 プロトコル変換ソフトウェアの動作フロー

ケットにアドレスを割り当てる。併せて、CAN デバイスを初期化し、イベント待ち状態になる。CAN テレグラムの受信を知らせるシグナルが入力されると、CAN コントローラの受信バッファからテレグラムを読み出し、テレグラム中のデータを取り出して UDP データグラムに組み立て直す。データグラムを Ethernet コントローラの送信バッファに書き出した後、イベント待ち状態に戻る。UDP データグラムを受信した場合も同様に CAN テレグラムとして送出処理が行われる。終了シグナルが与えられた場合は、CAN デバイスの終了処理を行い、ソケットを破棄して、実行を停止する。

2.6 通信試験と結果

プロトコル変換器の CAN および Ethernet インタフェースにそれぞれ CAN プロトコルアナライザと Ethernet プロトコルアナライザ( MOTEX 社 LanScope PRO )を接続して通信試験を行った( 図 7 )。

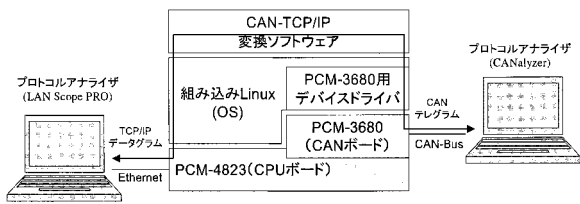


図 7 プロトコル変換器の動作試験方法

Ethernet プロトコルアナライザからプロトコル変換器を経由して CAN プロトコルアナライザへ、あるいはその逆方向で、伝送路上の通信量が幅広く変化する様に送出データ量を変化させながら、パケットの通信を試みた。試験中のプロトコルアナライザの画面の一例を図 8、図 9 にそれぞれ示す。通信試験の結果、TCP ストリームあるいは UDP データグラムと CAN テレグラムとが望み通りに交換されており、プロトコル変換器が良好に動作することを確認した。



図 8 動作試験中の CAN プロトコルアナライザの出力例



図 9 動作試験中の Ethernet プロトコルアナライザの出力例

### 3. Web リモートモニタシステムへの適用

#### 3.1 制御ネットワークに於ける WWW の意義

インターネットの急速な普及に伴い、WWW も非常に広く利用されるようになり、今日ではインターネットに於ける標準的なサービスとして位置付けられる。デスクトップコンピュータで利用されている OS のほとんど全てがインターネット接続に要するソフトウェアモジュールを備えており、かつその上で動作する何らかの Web ブラウザが無料で入手できるため、デスクトップコンピュータでは極めて容易に WWW サービスを利用できる。WWW は、サーバ・クライアント間の通信プロトコル (Hyper Text Transfer Protocol;

HTTP) とその上を流通するドキュメントから構成されるが、HTTP とドキュメント記述言語 (Hyper Text Markup Language; HTML) のいずれも標準化が成されており、数多くあるブラウザ・サーバの実装をどのような組み合わせでも、同一ドキュメントがほぼ同様に表示されることが期待できる。従って、ソフトウェアアプリケーションを開発する際、ユーザへのインタフェースに Web ブラウザを利用すると、開発の効率を向上することができる。そのため、制御ネットワークの構築に利用される機器に於いても、通信環境への WWW の採用やユーザインタフェースでの Web ブラウザの利用が進んでいる。

#### 3.2 リモートモニタシステムの概要

開発したプロトコル変換器を利用して、WWW を通信環境として用い、CAN バス上の制御ノードの状態をホスト PC 上でモニタできる、リモートモニタシステム (図10) を構築した。

ユーザからの HTTP リクエストに従いモニタ対象の CAN ノードと通信を行い、取得した応答データに基づき XML ページを生成・応答するために、HTTP サーバ・XML パーサ・リモートモニタソフトウェアをそれぞれ変換器上に実装した。システムの構築には以下のソフトウェアを利用した。

- Apache HTTP Server 1.3.17 (The Apache Software Foundation)
- Xerces-C++ 1.7.0 (The Apache Software Foundation)

また、リモートモニタソフトウェアは HTTP サーバと連携して動作する CGI プログラムとして実装した。

ユーザは、ホスト PC 上のブラウザ中での作業により、モニタ対象である CAN ノードの状態を容易にモニタできる。すなわち、ブラウザからシステムにアクセスすると、システムから入力フォームが応答されてブラウザに表示される。ユーザは状態をモニタする CAN ノードを特定するための識別子と CAN ノードに与えるコマンドデータをフォームに入力しシステムに送信する。システムは受け取ったフォームデータをデコードして得られた識別子とコマンドデータから CAN パケットを組み立てて CAN バス上へ送出する。当該 CAN ノードからの応答パケットを受信すると、システムはパケット中のデータを利用して Web ページ (xhtml ファイル) を生成しユーザ側の PC へ送信する。ユーザ側の PC では、

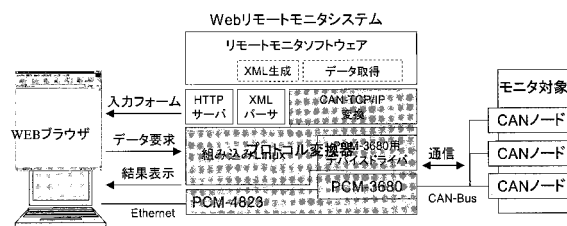


図10 Web リモートモニタシステムの構成

受け取った Web ページがブラウザ中へ表示されるので、ユーザは CAN ノードの状態を知ることが出来る。

### 3.3 通信試験と結果

リモートモニタシステムを、Ethernet でユーザ側 PC と、CAN バスで CAN プロトコルアナライザと接続して、システムの動作試験を行った(図11)。モニタ対象の CAN ノードは、プロトコルアナライザにエミュレーションさせた。車速センサ・エンジン回転計・水温計・ギヤボックスが制御ノードとして接続されたエンジン制御用バスをエミュレーションモデルとして作成し、模擬的なトラフィックデータ(図12)をプロトコルアナライザからバス上に出している状態で、ユーザ側 PC のブラウザからリモートモニタシステムへアクセスして、モニタ対象からのデータ取得を試みた。試験中にブラウザへ表示された Web ページの例を図13、図14に示す。システムにアクセスすると、応答された入力フォームがブラウザに表示される(図13)。状態をモニタしたい CAN ノード(この例では、車速センサ・エンジン回転計・水温計・ギヤボックス)の識別子(それぞれ、100・101・201・1020)をフォームに入力してシステムに送信すると、望み通りに、システムから当該 CAN ノードの応答結果が送り返されてきてブラウザに表示された(図14)。

試験結果から、ユーザがインタフェースとして使用するブラウザでの作業により、モニタ対象の状態を正しくモニタできることを明らかにした。従って、リモートモニタシステムの有効性・有用性が確認された。

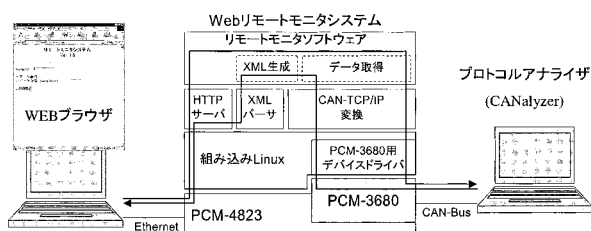


図11 Web リモートモニタシステムの動作試験方法

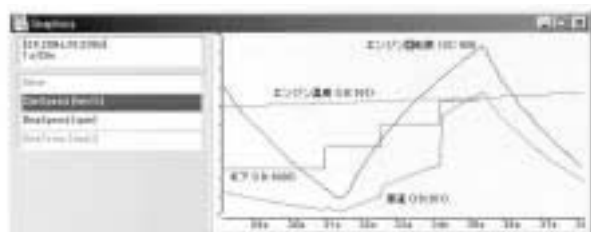


図12 試験中に CAN バスへ送出したトラフィックデータ

## 4 . おわりに

組み込み用 Linux を OS として利用する CAN-TCP/IP 通

信プロトコル変換器を開発した。これを用いて Web を通信環境に利用して CAN バスに接続した制御ノードの振る舞いを遠隔から監視できるリモートモニタシステムの構築し、その有効性を確認した。

今後は、農業車両等のバス化や FA ネットワークの構築を行う際の開発ツール(バスモニタ・制御ノード)としてプロトコル変換器の実用化を図りながら、本道農業機械・畜産機械メーカーへ CAN システム開発や分散制御ネットワーク構築に関する技術展開を行う。



図13 動作試験中の Web ブラウザの表示例（データ要求）



図14 動作試験中の Web ブラウザの表示例（応答表示）

## 引用文献

- 1) J. Marquering: LBS in der Applikationstechnik, *Proceedings of The 2nd Int. Symposium on Electronic Farm Communication with LBS*, pp.30-63, (2000)
- 2) R. Freimann: Autonavagation and Implement Controlled Process Automation on ISO-CAN basis, *Proceedings of The 2nd Int. Symposium on Electronic Farm Communication with LBS*, pp.64-87, (2000)

- 3 ) 元林浩太：CAN による制御・通信のためのトラクタ作業機ネットワーク，農機誌，Vol.63 No.4，pp.17-22，(2001)
- 4 ) 日攄英孝他：技術者のための UNIX 系 OS 入門，CQ 出版社，222PP.，(2000)
- 5 ) 日本エンベデットリナックスコンソーシアム：組み込み Linux 活用技法入門，CQ 出版社，64PP.，(2002)
- 6 ) W. Almesberger・H. Lermen:Using the initial RAM disk (initrd)，(1996)
- 7 ) R. Dick 他：Using the DiskOnChip with Linux OS, M-Systems,(2000)
- 8 ) A. Rubini:Linux Device Driver, O'Reilly&Associates, 442PP.,(1998)