分散制御ロボットにおける CAN コンポーネント

ユーザーマニュアル

平成 19 年 11 月 29 日

芝浦工業大学　水川研究室

〒135-8548 東京都江東区豊洲 3-7-5
芝浦工業大学研究棟 11Q32 水川研究室
TEL：03-5859-8209
FAX：03-5859-8201
Mail： shibaura.hri.goiken@gmail.com

指導教員：水川真
作成者：三浦俊宏

目次

1. はじめに

本書は，芝浦工業大学水川研究室で研究開発を行っている CAN を使用した分散制御ロボット(PAR)の CAN 通信機能要素を RT-Component 化したものについてその機能及び仕様，使用方法等を説明するものです．

本書では，以後 CAN 通信の機能要素を RT-Component 化したものを CAN Controller Component と呼び，CAN Controller Component とこれを動作確認及び評価を行うために作成した他の RT-Component を含めて CAN Components と呼びます．

1.1. CAN 通信の概要[2]

CAN(Controller Area Network)は，自動車内で複数の ECU(Electronic Control Unit)を LAN(Local Area Network)で接続して通信すること及びワイヤーハーネス削減などの目的で，1986 年ドイツの電装機器メーカであるロバート・ボッシュ社によって開発されたシリアル通信プロトコルです．その後，ISO(International Organization for Standardization–国際規格協会)にて共通規格化され，現在では CAN の高い性能と信頼性が認められ FA(Factory Automation)，船舶，産業機器，医療機器など多方面で導入されています．

CAN 通信の概略仕様を表 1-1 に示します．

表 1-1 CAN 通信の概略仕様

項目	仕様
伝送方法	半二重シリアル通信
ネットワークトポロジー	バス型（マルチマスタ）
アクセス制御方法	CSMA/NBA
同期方式	同歩同期＋位相補正
データ長	最大 8[Byte]
誤り制御	CRC方式
符号化方式	NRZ方式＋ビットスタッフ
最大通信速度	1Mbps(ISO11898)，125kbps(ISO11519)

CAN は半二重シリアル通信を行うためのプロトコルです．通信を行う複数のノードは CAN バスで接続します．バスに参加するノードにマスタ/スレーブという区別は無く，CAN バスが空いていればどのノードも送信を開始することが出来ます．複数のノードが同時に送信を開始した場合には，CSMA/NBA 方式でアービトレーションが行われ，1 つのノードが送信権利を得ます．送信データは最大 8 バイトまでの可変長で，ID(Identifier-識別子)や CRC コードの付加したメッセージとして全ノードに対しブロードキャストされます．ID はメッセージの優先度やデータの意味を示し，ネットワークの設計時に各ノードが送受信する ID を割り当てます．受信側ではバスに送信されたメッセージの ID を選別して自ノードで必要なデータを取り込みます．

メッセージは，NRZ 変調とビットスタッフ処理によってエンコード（符号化）され，CAN トランシーバを通してバスに送出されます．

CAN バスは一般に 2 線式の作動信号で構成します．バスの値は論理ビット”0”と”1”に対応する 2 値で”ドミナント”と”レセシブ”と呼ばれます．バスの電気的特性は，ボッシュ社の仕様には規定されておらず，ISO11898 などの規定を適用します．ISO11898 での最大通信速度は 1Mbps です．図 1-1 に CAN ネットワークの構成を示します．

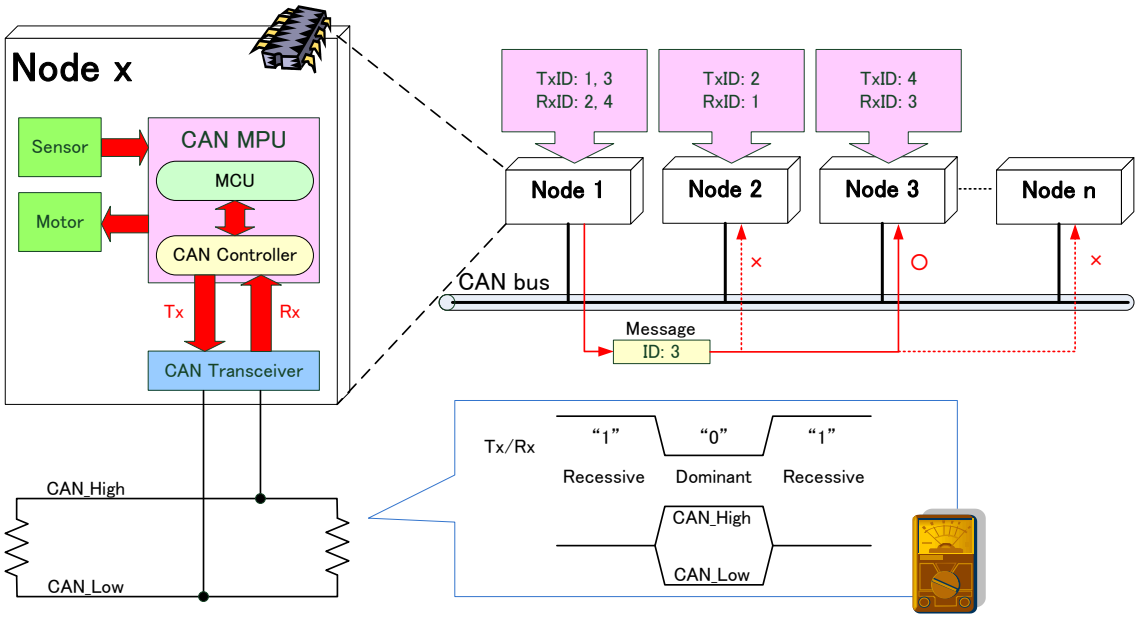

図 1-1 CAN ネットワークの構成例

1.2. PAR (Physical Agent Robot)

PAR とは，本研究室で研究開発を行っている遠隔地との情報伝達，人間との相互作用を実現する物理エージェントシステム(PAS:Physical Agent System) [1]や屋外環境での自律移動で使用しているロボットのことです.

PAR は機能要素ごとに処理系を搭載する(以下，サブシステム)分散制御系を採用し，このサブシステム間のネットワークには CAN を採用しています. CAN は当初車載系の LAN として開発されたものであるが，その信頼性や洗練された故障検出機能などが認められ，ビル・オートメーション・医療機器・海洋関係の電子機器など幅広い分野で使用されています. これによって, PAR はサブシステムの組み合わせで容易に構築することが可能となり，多様な機能要求に応じることが可能なシステム構成となっています.

1.3. 本書の目的

本書では，CAN 通信の機能要素を RT-Component 化し提供することによって技術の共有・蓄積への貢献を目的とし，CAN Controller Component の実装を行いました.

1.4. 本書の構成

本書は以下の構成で記述します.

章番号	タイトル	構成
2.	CAN Components の概要	CAN Components の機能概要について記述しています.
3.	CAN Components の仕様	CAN Components の各 RT-Component の仕様について記述しています.
4.	CAN Components をご利用の前にご準備いただきたいこと	CAN Components をご利用する前にご準備いただくものについて記述しています.
5.	CAN Components の利用	提供するプロジェクトのビルド方法等について記述しています.
6.	CAN Controller Component の使用例	CAN Controller Component を使用した例について記述しています.
7.	動作環境	CAN Components の動作環境について記述しています.
8.	ライセンスに等ついて	CAN Components のライセンス等について記述しています.
9.	連絡先	連絡先について記述しています.

2. CAN Components の概要

CAN Components は CAN Controller Component とこれを動作確認及び評価するために作成した RT-Component のことです.

2.1. システム構成

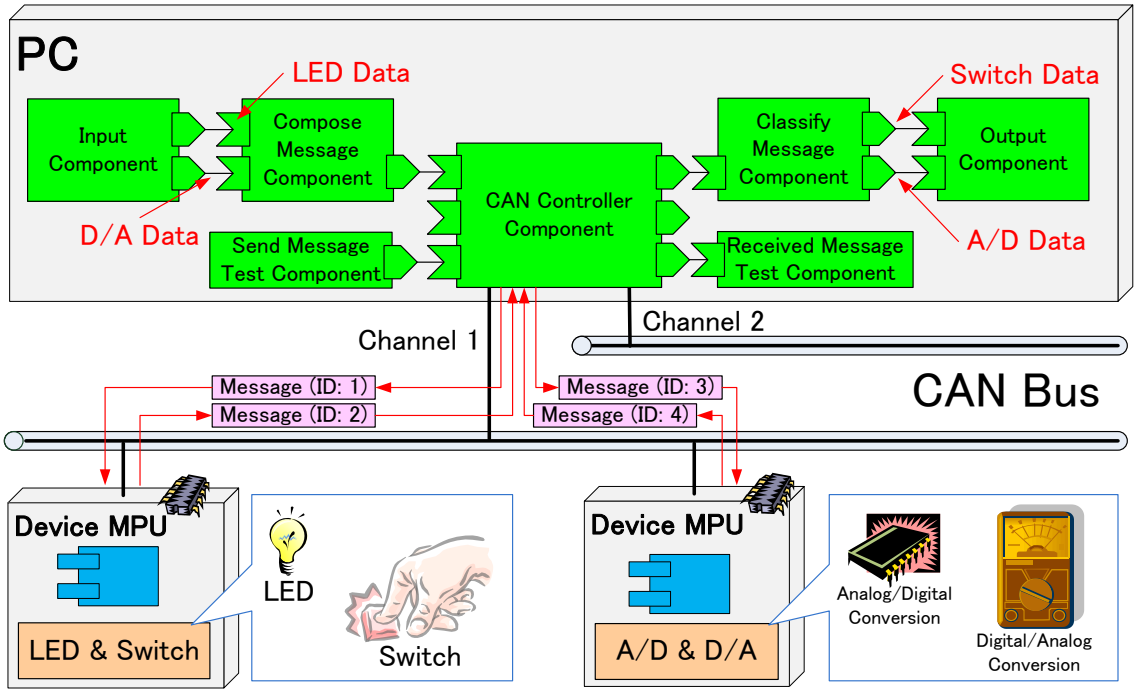

図 2-1 CAN Components のシステム構成

CAN の通信機能要素を RT-Component 化した CAN Controller Component の動作確認及び評価を行うにあたり図 2-1 のようなシステムを作成しました．このシステムの動作内容は以下のようになっています.

まず, コンソールで入力した LED 及び D/A 変換の指令値を Input Component で出力し，このデータをもとに Compose Message Component で ID，DLC などの設定をし，CAN メッセージの生成を行い出力します.

CAN Controller Component は，2 つの CAN バスに CAN メッセージを送受信することが可能な構成となっており，それぞれの CAN バスに送受信できるように入力ポート及び出力ポートを持たせています．このように CAN Controller Component では入出力ポートに対応した CAN バスに CAN メッセージの送受信を行います.

Classify Message Component では，Device MPU で出力された CAN メッセージを分析しデータの振り分けを行います．そして，スイッチの状態及び A/D 変換の処理結果を各出力ポートから出力するようになっています．Output Component は，Classify Message Component から出力されたそれぞれのデータをコンソール画面に出力します.

Send Message Test Component 及び Received Message Test Component は CAN Controller Component へ直接 CAN メッセージの送受信をテストするための RT-Component です.

以上の RT-Component を使用することによって CAN 通信の部分を意識することなく LED やスイッチの 1 個 1 個を制御することが可能となりました.

また，本システムでは LED 及びスイッチ，A/D 変換，D/A 変換を制御する構成となっていますが，CAN バスに接続された各 Device MPU に合わせて Compose Message Component 及び Classify Message Component を作り変えることによって容易に各 Device

MPU を制御することが可能となります．これによって様々な CAN バスを用いた分散制御システムやロボットを容易に開発することや設計の変更，機能の追加を行うことが可能なシステム構成となっています．

2.2. 適用範囲

CAN の通信機能要素を RT-Component 化するにあたり，ベクター・ジャパン社製 XL ファミリー製品を使用し，CAN Controller Component の実装を行いました．

そのため，本書の内容はベクター・ジャパン社製の CAN インターフェイスに対応した XL ファミリー製品が適用範囲となっております．

また，本製品のドライバ適用範囲が Windows 系 OS となっているため，本書の適用範囲も Windows 系 OS となっています．

3. CAN Components の仕様

3.1. CAN Controller Component

3.1.1. 概要

CAN Controller Component は，ベクター・ジャパン社製 XL ファミリー製品を使用し，CAN 通信の機能要素を RT-Component 化したものです．

CAN Controller Component は，2 つの CAN バスに CAN メッセージを送受信することが可能な構成となっており，それぞれの CAN バスに送受信できるように入力ポート及び出力ポートを持たせています．また，両方の CAN バスに送受信可能な入力ポート及び出力ポートも持たせています．

さらに，CAN Controller Component には受信する CAN メッセージ ID のフィルタ機能を実装しており，RTC-Link 上で受信したい ID のメッセージ設定を動的に行うことが可能となっています．これによって，任意の ID を持つ CAN メッセージのみを受信することが可能となり処理負荷を軽くすることが可能です．

3.1.2. 入出力ポート仕様

CAN Controller Component の外観を図 3-1 に示します．CAN Controller Component は，入力ポート及び出力ポートを 3 つずつ用意しており，それぞれのポート仕様については表 3-1 に示した通りです．

各入力ポートは，CAN バスへ送信するメッセージデータを入力としており，このデータ長は CAN メッセージ長を示す DLC の値によって変わるようになっています．DLC が 8 の場合は表 3-2 のようになり，DLC が 4 の場合は表 3-3 のようになります．

各出力ポートのデータ長も入力ポートと同様で DLC の値によって変わります. DLC が 8 の場合は表 3-4 のようになり，DLC が 4 の場合は表 3-5 のようになります．

表 3-4 及び表 3-5 にある Message Type とは，受信したチャンネル名と CAN Controller Component が送信したメッセージか Device MPU から送信されたメッセージかどうかを示したデータとなっています. Message Type のデータ構成を表 3-6 に示します. また，表 3-6 に示したデータの説明を表 3-7 に示します．

また，表 3-4 及び表 3-5 にある Time[nsec]とは CAN Controller Component が CAN バスに接続を行ってからの経過時間（CAN Controller Component が Active になってからの経過時間）となっています．

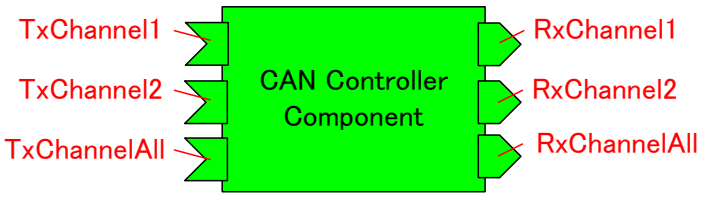

図 3-1 CAN Controller Component の外観

表 3-1 CAN Controller Component のポート仕様

ポート種別	ポート名称	データ型	説明
InPort	TxChannel1	TimedOctetSeq	入力されたデータをCANバスのチャンネル1に送信する.
	TxChannel2	TimedOctetSeq	入力されたデータをCANバスのチャンネル2に送信する.
	TxChannelAll	TimedOctetSeq	入力されたデータをCANバスのチャンネル1と2に送信する.
OutPort	RxChannel1	TimedOctetSeq	CANバスのチャンネル1で受信したデータを出力する.
	RxChannel2	TimedOctetSeq	CANバスのチャンネル2で受信したデータを出力する.
	RxChannelAll	TimedOctetSeq	CANバスのチャンネル1及び2で受信したデータを出力する.

表 3-2 入力データ仕様　(DLC=8 の場合)

データ名称	Bit数			
	8bit	8bit	8bit	8bit
ID	data[3]	data[2]	data[1]	data[0]
DLC	data[4]			
D0	data[5]			
D1	data[6]			
D2	data[7]			
D3	data[8]			
D4	data[9]			
D5	data[10]			
D6	data[11]			
D7	data[12]			

表 3-3 入力データ仕様 (DLC=4 の場合)

データ名称	Bit数			
	8bit	8bit	8bit	8bit
ID	data[3]	data[2]	data[1]	data[0]
DLC	data[4]			
D0	data[5]			
D1	data[6]			
D2	data[7]			
D3	data[8]			

表 3-4 出力データ仕様 (DLC=8 の場合)

データ名称	Bit数							
	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
ID	data[3]	data[2]	data[1]	data[0]				
DLC	data[4]							
D0	data[5]							
D1	data[6]							
D2	data[7]							
D3	data[8]							
D4	data[9]							
D5	data[10]							
D6	data[11]							
D7	data[12]							
MessageType	data[13]							
Time[nsec]	data[21]	data[20]	data[19]	data[18]	data[17]	data[16]	data[15]	data[14]

表 3-5 出力データ仕様 (DLC=4 の場合)

データ名称	Bit数							
	8bit	8bit	8bit	8bit	8bit	8bit	8bit	8bit
ID	data[3]	data[2]	data[1]	data[0]				
DLC	data[4]							
D0	data[5]							
D1	data[6]							
D2	data[7]							
D3	data[8]							
MessageType	data[9]							
Time[nsec]	data[17]	data[16]	data[15]	data[14]	data[13]	data[12]	data[11]	data[10]

表 3-6 Message Type(8bit)の構成

データ名称	Bit数							
	1bit	1bit	1bit	1bit	1bit	1bit	1bit	1bit
Message Type				Tx/Rx	Channel種別			

表 3-7 Message Type の仕様

データ名称	説明
Tx/Rx	"1"の場合は送信, "0"の場合は受信したメッセージである
Channel種別	"0"の場合は"CANcardXL Channel1"からのメッセージ, "1"の場合は"CANcardXL Channel2"からもメッセージ, "2"の場合は"Virtual Channel1"からのメッセージ, "3"の場合は"Virtual Channel2"からのメッセージ (※これはCANcardXLだけを使用した場合)

3.1.3. Configuration

CAN Controller Component には受信する CAN メッセージ ID のフィルタ機能を実装しており，RTC-Link の”ConfigurationView”で動的に設定することが可能です．

このフィルタ機能は，ID_Form に設定した ID から ID_To に設定した ID までの CAN メッセージのみを受信するようになっています．RTC-Link 画面を図 3-2 に示し，設定内容の詳細を表 3-8 に示します．

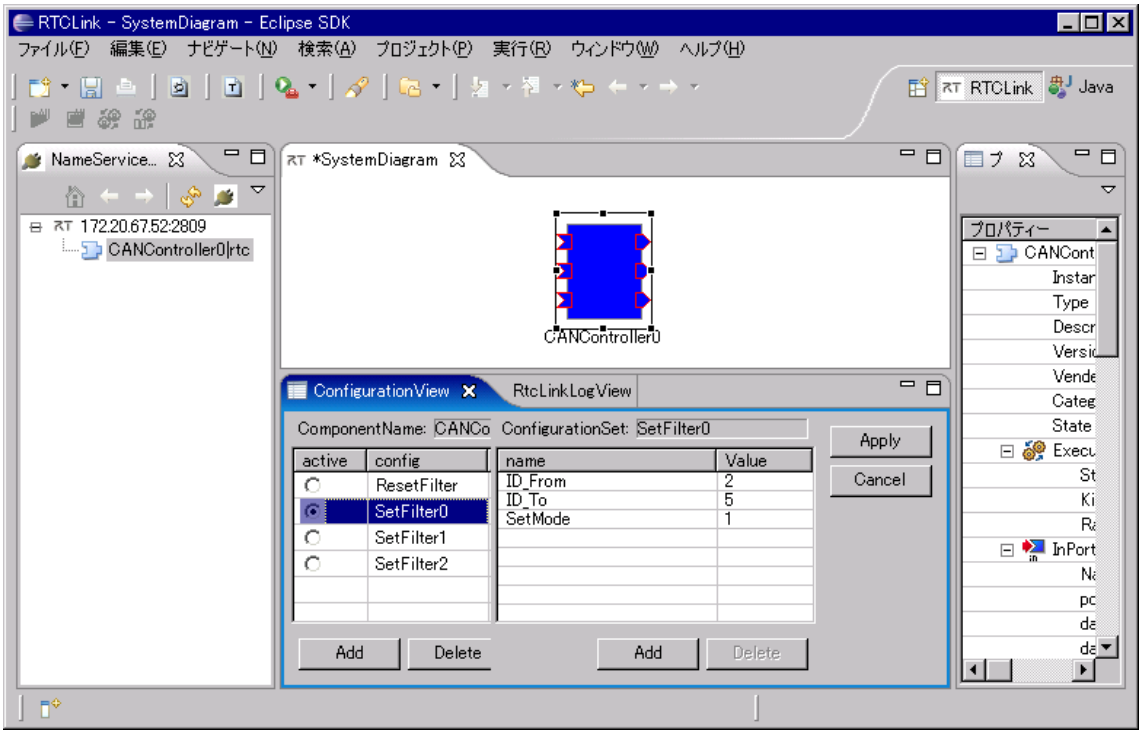

図 3-2 RTC-Link の画面

表 3-8 Configuration View での設定内容

名称	データ型	説明
ID_From	unsigned long	受信したいCANメッセージの始めのID(10進数)
ID_To	unsigned long	受信したいCANメッセージの終わりのID(10進数)
SetMode	bool	1:フィルタ機能を有効にする. 0:フィルタ機能を無効にする.

3.2. Send Message Test Component

3.2.1. 概要

Send Message Test Component は，CAN Controller Component へ送信したい(CAN バスへ送信したい)メッセージデータを生成して出力する RT-Component です.

この RT-Component でメッセージデータを生成するためには，まず ID 及び DLC(データ長)を入力します．そして，DLC で指定した Byte 数のデータを入力して CAN Controller Component へ送信するためのメッセージデータの生成を行います.

3.2.2. 入出力ポート仕様

Send Message Test Component の外観を図 3-3 にポートの仕様については表 3-9 に示します.

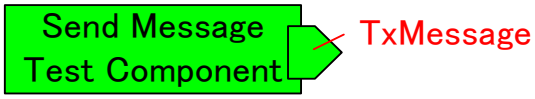

図 3-3 Send Message Test Component

表 3-9 Send Message Test Component のポート仕様

ポート種別	ポート名称	データ型	説明
OutPort	TxMessage	TimedOctetSeq	CANバスへ送信するデータをCAN Controller Componentへ出力する.

3.3. Received Message Test Component

3.3.1. 概要

Received Message Test Component は，CAN Controller Component から出力されたメッセージデータを入力とし，入力されたデータをコンソール画面に出力する RT-Component です.

3.3.2. 入出力ポート仕様

Received Message Test Component の外観を図 3-4 に，ポートの仕様については表 3-10 に示します.

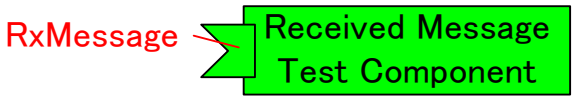

図 3-4 Received Message Test Component

表 3-10 Received Message Test Component のポート仕様

ポート種別	ポート名称	データ型	説明
InPort	RxMessage	TimedOctetSeq	CAN Controller Componentで出力されたデータを入力とする.

3.4. Compose Message Component

3.4.1. 概要

Compose Message Component は入力ポートから入ってきたデータに対し，あらかじめ設定しておいた ID 及び DLC を設定し CAN Controller Component へ出力するメッセージデータの生成を行う RT-Component です.

この RT-Component は，CAN バスに接続されている各 Device MPU の受信データに合わせて設計するようになっています．提供するプログラムでは，入力ポートに Device MPU へ送信する LED の指令値及び D/A 変換の指令値を用意しています.

3.4.2. 入出力ポート仕様

Compose Message Component の外観を図 3-5 に，ポートの仕様については表 3-11 に示します.

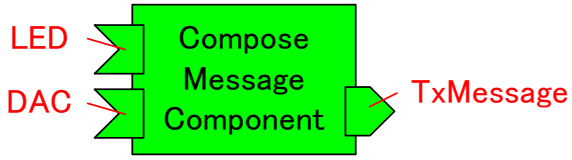

図 3-5 Compose Message Component

表 3-11 Compose Message Component のポート仕様

ポート種別	ポート名称	データ型	説明
InPort	LED	TimedOctet	LEDの制御データを入力とする.
	DAC	TimedShort	D/A変換の制御データを入力とする.
OutPort	TxMessage	TimedOctetSeq	CANバスへ送信するデータをCAN Controller Componentへ出力する.

3.5. Classify Message Component

3.5.1. 概要

Classify Message Component は，入力ポートから入ってきたメッセージデータを参照しあらかじめ設定しておいた ID 及び DLC を見てデータを振り分け，それぞれ対応する出力ポートから出力する RT-Component です.

この RT-Component は，CAN バスに接続されている各 Device MPU の送信データに合わせて設計するようになっています. 提供するプログラムでは，出力ポートに Device MPU から送信されたスイッチの状態及び A/D 変換の処理結果を出力するポートを用意しています.

3.5.2. 入出力ポート仕様

Classify Message Component の外観を図 3-6 に，ポートの仕様については表 3-12 に示します.

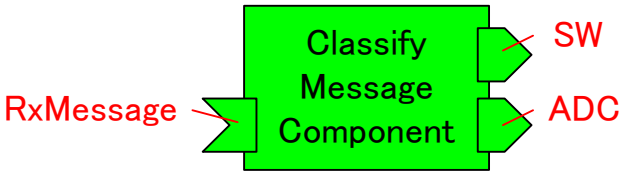

図 3-6 Classify Message Component

表 3-12 Classify Message Component のポート仕様

ポート種別	ポート名称	データ型	説明
InPort	RxMessage	TimedOctetSeq	CAN Controller Componentで出力されたデータを入力とする.
OutPort	SW	TimedOctet	スイッチの状態データを出力する.
	ADC	TimedShort	A/D変換の状態データを出力する.

3.6. Input Component

3.6.1. 概要

Input Component は, Compose Message Component へコンソール画面で入力した LED 及び D/A 変換の指令値を出力する RT-Component です.

3.6.2. 入出力ポート仕様

Input Component の外観を図 3-7 に，ポートの仕様については表 3-12 に示します.

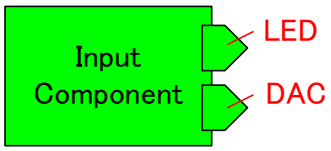

図 3-7 Input Component

表 3-13 Input Component のポート仕様

ポート種別	ポート名称	データ型	説明
OutPort	LED	TimedOctet	LEDの制御データを出力する.
	DAC	TimedShort	D/A変換の制御データを出力する.

3.7. Output Component

3.7.1. 概要

Output Component は，Classify Message Component で振り分けされ出力され Device MPU のスイッチの状態及び A/D 変換の処理結果をコンソール画面へ出力する RT-Component です.

3.7.2. 入出力ポート仕様

Output Component の外観を図 3-8 に，ポートの仕様については表 3-14 に示します.

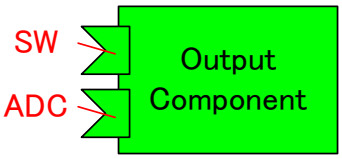

図 3-8 Output Component

表 3-14 Output Component のポート仕様

ポート種別	ポート名称	データ型	説明
OutPort	SW	TimedOctet	スイッチの状態データを入力とする.
	ADC	TimedShort	A/D変換の状態データを入力とする.

4. CAN Components をご利用の前にご準備いただきたいこと

4.1. 準備していただく製品

CAN Controller Component を使用するためには，ベクター・ジャパン社製 XL ファミリーの CAN インターフェイス製品が必要となります．

本書では，XL ファミリーCAN インターフェイス製品の 1 つである CANcardXL を使用し説明をします．CANcardXL を図 4-1 に示します．

図 4-1 CANcardXL の外観

4.2. ダウンロードしていただくファイル

CAN Controller Component はベクター・ジャパン社製の製品を使用しています．そのため，製品を制御する API のソースコード及びライブラリ等が必要となります．必要なソースコード等は以下に示す 3 つです．（これらのソースコードやライブラリ等は開発元になります"Vector Informatik"に著作権があります．）

- vxlapi.h
- vxlapi.lib
- vxlapi.dll

以上の 3 つは以下に記載する方法で入手可能です．

(1) ベクター・ジャパン社様のホームページ（http://www.vector-japan.co.jp/）に行きます．
(2) ダウンロードページから"XL Driver Library"という名前のパッケージをダウンロードします．(本プロジェクトでは，” XL Driver Library 6.4”（ファイル名：xl_lib64.zip）を用いて作成いたしました．)
(3) ダウンロードしたファイルを解凍しその中に入っている Setup.exe を実行し，適当な場所にインストールします．
(4) インストールしたフォルダ(XL Driver Library)の中に”bin”というフォルダがあります．この bin の中に 3 つのソースコード及びライブラリがあります．

5. CAN Components の利用

5.1. ビルド方法

CAN Components のビルド方法については，OpenRTM-aist-0.4.0 及びベクター・ジャパン社製 XL ファミリーで CAN インターフェイスを持った製品のドライバをインストールしていることを前提とし説明いたします.

ビルド方法は以下の順で行います.

(1) 本プロジェクトを適当な場所に解凍します.
(2) (1)で解凍した" CANComponents"のフォルダの中にある" VectorJapanFiles"という名前のフォルダ内に「3.2. ダウンロードしていただくファイル」でダウンロードした 3 つのソースコード及びライブラリ(vxlapi.h, vxlapi.lib, vxlapi.dll)をコピーします.
(3) Visual Studio 2005 で "CANComponents" のフォルダの中にある"CANComponens.sln"を開き，ビルドを行います.

以上の方法で正常にビルドが終了しましたら完了です.

5.2. 起動方法

CAN Controller Component を起動する場合以下の手順で起動を行ってください．(1)～(6)のチャンネル設定は初めて起動する場合もしくはチャンネル設定の変更を行う場合の作業となります．2 回目以降は省略可能です．(製品を使用しないで動作確認を行う方法については付録の 1.に記載しています.)

また，CAN Controller Component 以外の RT-Component は CAN Components のフォルダの中にある"bin"フォルダにあるそれぞれの exe ファイルを実行するだけで起動します.

(1) 最初に CAN Controller Component のチャンネルの設定を行います．これは，ベクター・ジャパン社から提供されているドライバ設定ソフト"vcanconf.exe"を使用します．まず，製品が PC に接続されていることを確認してから"vcanconf.exe"を起動してください．("vcanconf.exe"はベクター・ジャパン社ホームページからダウンロード可能です．ドライバのファイルをダウンロードすると中に入っています.)

(2) 図 5-1 に示す"Vector Hardware Config"の画面で正常にドライバが設定できていることを確認します.(本書では, CANcardXL という製品を使っているため"Hardware"のところに"CANcardXL"と表示されています.)

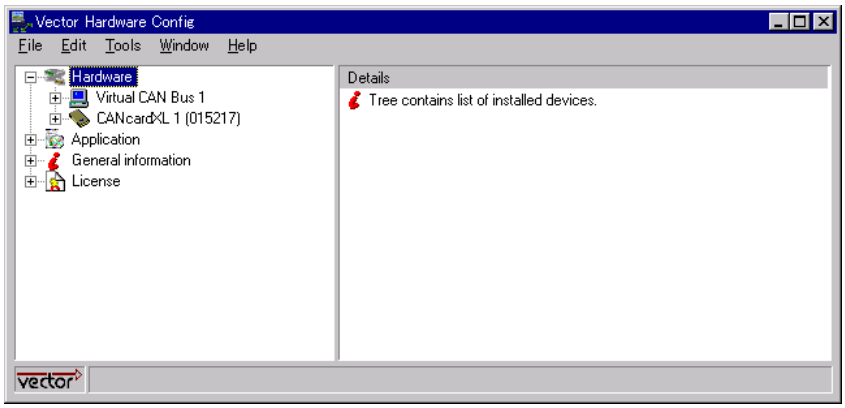

図 5-1 Vector Hardware Config

(3) 図 5-2 に示すように[Application]→[右クリック]→[Add application]の順でアプリケーションの追加を行います.

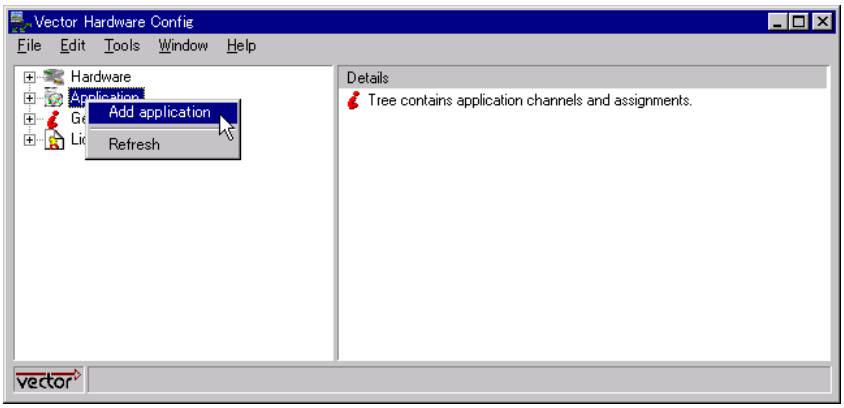

図 5-2 Vector Hardware Config（アプリケーションの追加設定）

(4) 図 5-3 に示す"Application setting"の画面で"Application name"の欄に"CANControllerComponent"と入力し，"OK"ボタンを押します.

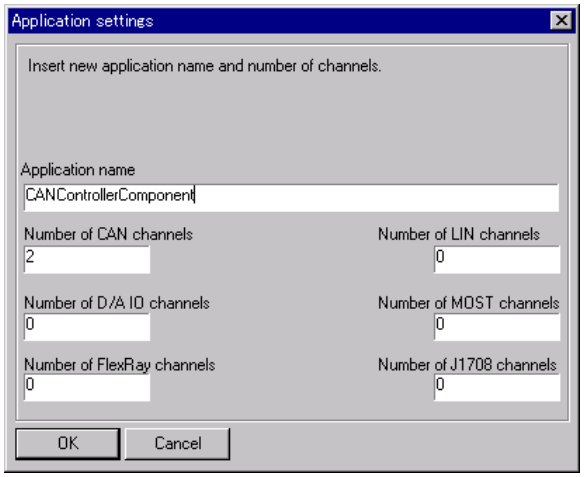

図 5-3 Application setting（アプリケーションの追加設定）

(5) 図 5-4 に示すように左側の欄で"CANControllerComponent"を選択し，右側の設定画面で[CAN 1 を選択]→[右クリック]→[CANcardXL]→[CANcardXL 1 Channel 1]を選択します. 同様に[CAN 2 を選択]→[右クリック]→[CANcardXL]→[CANcardXL 1 Channel 2]を選択します．以上の設定で使用する CAN バスを 2 つ選択します.

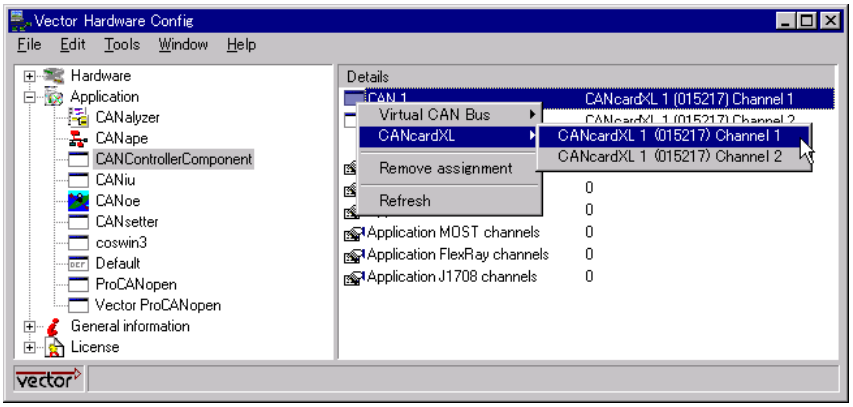

図 5-4 Vector Hardware Config（使用する CAN バスの選択設定）

(6) 図 5-5 のように設定されていることを確認し，”Vector Hardware Config”の画面を閉じてください.

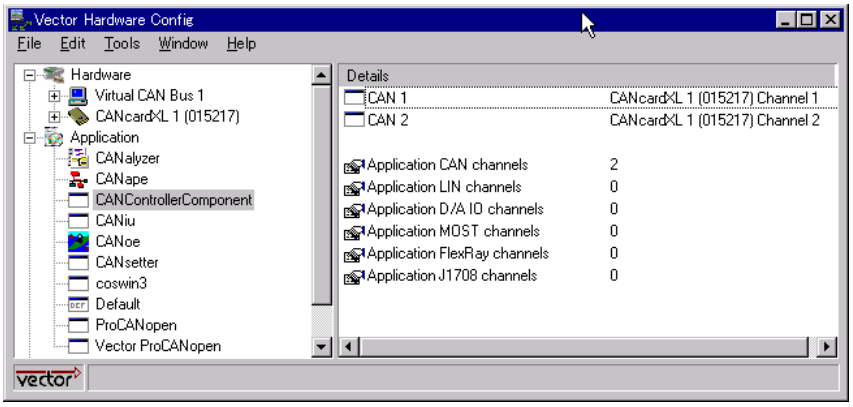

図 5-5 Vector Hardware Config

(7) 本書と同様に, CANcardXL を使用している場合はそれぞれのチャンネルにケーブルが接続されていることを確認してください．ケーブルが接続されていない場合，次の(8)でエラーが出ます.

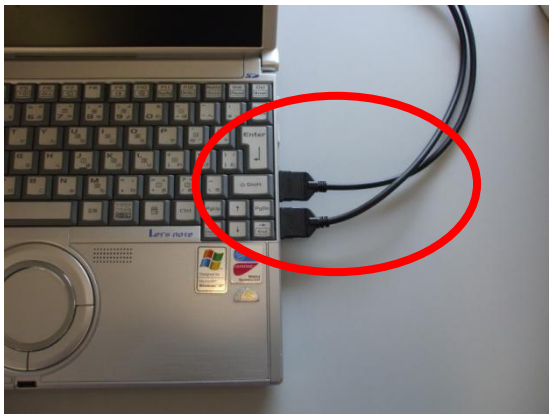

図 5-6 CANcardXL とケーブルを接続

(8) OpenRTM-aist ツールの”rtm-naming.bat”を起動後，”CANComponents”のフォルダの中にある”bin”のフォルダに”CANController.exe”がありますのでこれを起動し

てください．起動すると図 5-7 に示すような画面が出ます．画面に先ほど設定したチャンネルが表示されていれば正常に起動しています．

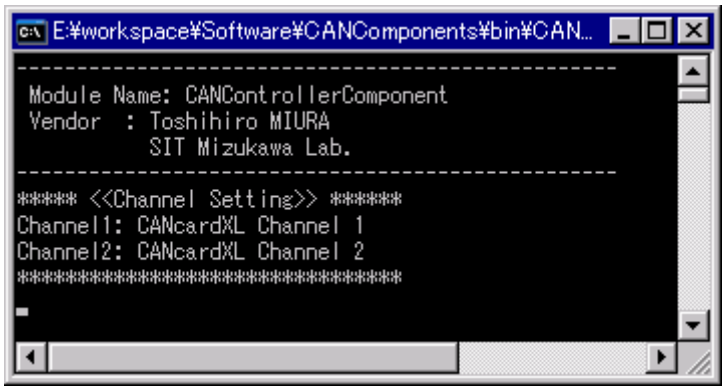

図 5-7 起動画面

6. CAN Controller Component の使用例

6.1. 分散制御ロボットにおける使用例

CAN Controller Component を使用した例として，本コンテストにて同研究室から参加している『屋外自律移動ロボットにおける GPS コンポーネント』との組合せで構成した分散制御ロボット，PAR-NE07(Physical Agent Robot for Natural Environment)[4]があります.

この PAR-NE07 とは, 本研究室で研究開発を行っている分散制御ロボットの 1 つであり, 屋外環境でロボットを運用する為の技術獲得を目的に開発された屋外自律型移動ロボットです．PAR-NE07 の外観を図 6-1 にシステム構成を図 6-2 に示します.

CAN Controller Component は，PAR-NE07 のように CAN バスに接続された複数の Device との通信を容易に実現することを可能とします．そして，各 Device の構成の変更や新しい Device の追加などを行う場合でも Compose Message Component 及び Classify Message Component を修正することによって容易に対応することが可能です.

これによって，容易に分散制御ロボットの開発，機能の拡張・変更を行うことが可能となります.

図 6-1 PAR-NE07 の外観

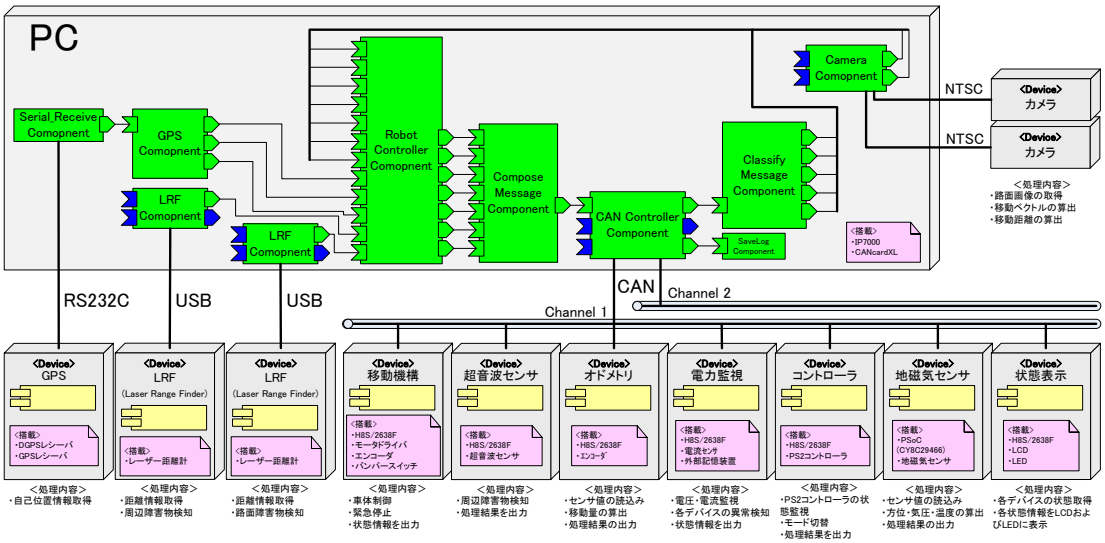

図 6-2 PAR-NE07 のシステム構成

7. 動作環境

7.1. 動作対象環境

- Visual Studio 2005 がインストールされている Windows 環境
- RT-Middleware (OpenRTM-aist-0.4.0)
- ベクター・ジャパン社製 XL ファミリー製品

7.2. 動作確認済み環境

- Windows XP (Visual Studio 2005 でコンパイル)
- RT-Middlware (OpenRTM-aist-0.4.0)
- CANcardXL（ベクター・ジャパン社製 XL ファミリー製品）
- CANboradXL（ベクター・ジャパン社製 XL ファミリー製品）

7.3. 開発環境

- Windows XP
- Microsoft　Visual Studio 2005 (VC++8.0)
- RT-Middleware (OpenRTM-aist-0.4.0)
- CANcardXL（ベクター・ジャパン社製 XL ファミリー製品）

8. ライセンス等

ベクター・ジャパン社製の CAN インターフェイスを制御する API("vxlapi"等)のソースコード及びライブラリを除き，CAN Components の著作権は，芝浦工業大学水川研究室に帰属します.

API("vxlapi"等)のソースコード及びライブラリの著作権は，開発元の "Vector Informatik"にあります.

9. 連絡先

≪CAN Components について≫

芝浦工業大学　水川研究室
〒135-8548
東京都江東区豊洲 3-7-5 芝浦工業大学　研究棟 11Q32　水川研究室
TEL：03-5859-8209　　FAX：03-5859-8201
Mail：shibaura.hri.goiken@gmail.com
指導教員：水川　真
作成者：三浦　俊宏
URL：http://www.hri.ee.shibaura-it.ac.jp/

≪CANcardXL 製品および API について≫

ベクター・ジャパン株式会社（東京本社）
〒140-0002
東京都品川区東品川 2-3-12　シーフォートスクエアセンタービル 18F
TEL：03-5769-6970　　FAX：03-5769-6975
URL：http://www.vector-japan.co.jp/

≪付録≫

1.　製品を使用しないで動作確認を行う方法

CAN Controller Component は，ベクター・ジャパン社製 XL ファミリ製品のドライバをインストールすることで，以下の方法で製品を使用しなくても動作確認を行うことが可能です．

(1)　『5.2. 起動方法』の(5)の使用する CAN バスの設定の部分で”Virtual CAN Bus”をします．図 7-0-1 のように設定を行ってください．

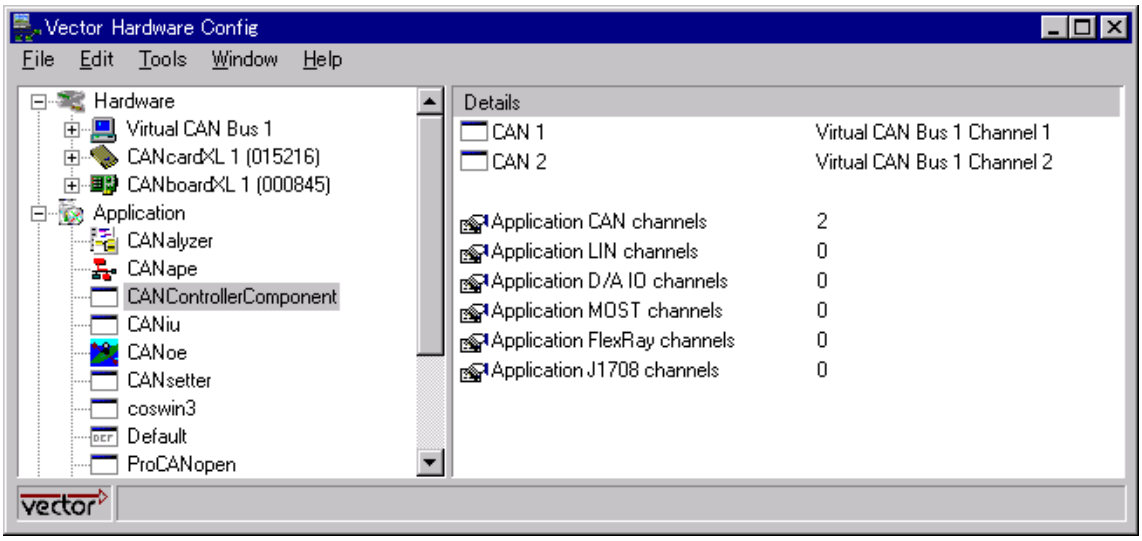

図 7-0-1 Vector Hardware Config

(2)　『5.2. 起動方法』の(8)で図 7-0-2 に示すような起動画面になっていれば正常に起動しています．これで，仮想 CAN バスを使用した動作確認を行うことが可能です．

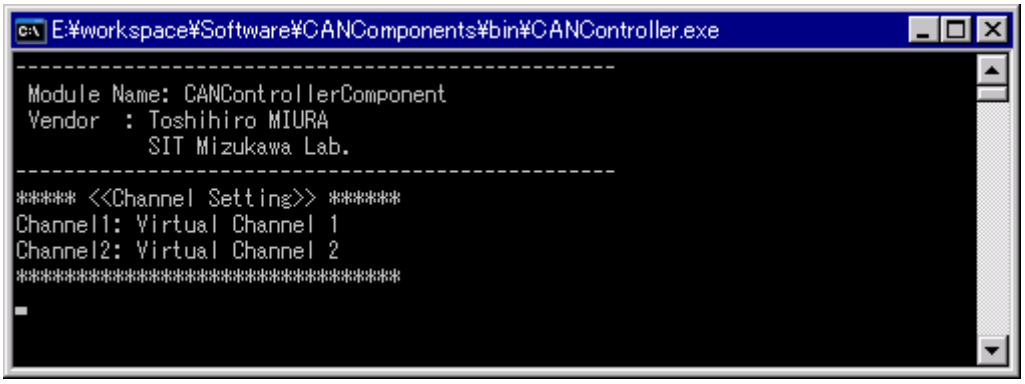

図 7-0-2 起動画面

≪参考資料≫

[1] 水川　他：物理エージェント（PAS）を用いた，『遠隔地間人間協調系の基本検討』，SI2000，pp125-126
[2] 五十嵐資朗・佐藤正幸・玉城礼二[著]，『CAN 入門講座－組み込みマイコンで学ぶ CAN プロトコルとプログラミング－』，株式会社　電波新聞社（発行）
[3] ベクター・ジャパン株式会社ホームページ，http://www.vector-japan.co.jp/
[4] 池田　他：屋外自律移動ロボットの実装と要素技術，SI2007

≪謝辞≫

本コンポーネント作成にあたり，ベクター・ジャパン様から提供されている API，ライブラリ、DLL を使用させて頂きました．
また，ベクター・ジャパン様には，度重なる質問や相談に快く応じて頂きました．
多大なご指導ご協力を頂きましたことを，ここに記して感謝申し上げます．