文書番号：PSAPRT-D-005

知能化技術開発プロジェクト⑥-③
環境自己位置同定モジュール
解説書

第 1.7 版　2009 年 12 月 10 日

NECソフト株式会社

＜改版履歴表＞

版数	発行日	該当頁	改版内容	承認	作成
1.0	2008/03/26	全頁	・新規作成	峰岸	高田
1.1	2008/08/26	全頁	● 2.1ハードウェア構成(Windows 版) ● 2.2ソフトウェア構成(Windows 版) ● 4実行環境の構築手順（Windows 版） ● 5コンポーネントの起動手順（Windows 版） ● 6開発環境の構築手順（Windows 版） の追加		寺井
1.2	2008/12/27	全頁	● 2.1ハードウェア構成(Windows 版)を ハードウェア構成（Windows 版・UNIX 版）に変更 ● 2.2ソフトウェア構成(Windows 版)を ソフトウェア構成（Windows 版・UNIX 版）に変更 ● 図 2-3 SW 構成(Windows 版・UNIX 版)に Ubuntu を追加 ● 表 2-3 SW 詳細(Windows 版・UNIX 版)の SW2-1 に Ubuntu8.04 の追加		寺井
			● 7実行環境の構築手順（UNIX 版） ● 8コンポーネントの起動手順（UNIX 版） ● 9開発環境の構築手順（UNIX 版） の追加		
1.3	2009/02/17	全頁	● 3.1.4設定ファイル ● 3.1.5ログファイル ● 4.1.4赤外線受信モジュール設定ファイルの転送 ● 7.1.4赤外線受信モジュール設定ファイルの転送 の追加		寺井
			● 3.2.7コンフィグレーションセット定義 ● 3.2.9設定ファイル ● 3.2.10ログファイル ● 3.3.7コンフィグレーションセット定義 ● 3.3.8設定ファイル ● 3.3.11ログファイル ● 3.4.7コンフィグレーションセット定義 ● 3.4.9設定ファイル ● 3.4.10ログファイル ● 図 4-1 ● 図 7-4 の修正		

1.4	2009/02/26	83 頁～	● １０実行環境の構築手順（Teacube 版 UNIX 環境） ● １１コンポーネントの起動手順（Teacube 版 UNIX 環境） ● １２開発環境の構築手順（Teacube 版 UNIX 環境） の追加		寺井
1.5	2009/03/25	20 頁～	● ３.１.４設定ファイル ● ３.１.５ログファイル ● 表 ３-５ ● ３.２.９設定ファイル ● ３.２.１０ログファイル ● 表 ３-１０ ● ３.３.８設定ファイル ● ３.３.１１ログファイル ● ３.３.１３アクティビティ図 ● 表 ３-１５ ● ３.４.９設定ファイル ● ３.４.１０ログファイル ● ３.４.１２アクティビティ図 ● 図 ４-１ ● ４.２.３環境自己位置同定コンポーネントのインストール ● ５.１コンポーネントの起動手順 ● ６.１.８赤外線受信モジュールのビルド ● ６.２.４環境自己位置同定コンポーネントのビルド ● 図 ７-１ ● ７.２.３環境自己位置同定コンポーネントのインストール ● ８.１コンポーネントの起動手順 ● ９.１.９赤外線受信モジュールのビルド ● ９.２.１０UCODE 受信コンポーネントのビルド～９.２.１３テストコンポーネントのビルド ● 図 １０-１ ● １０.２.５環境自己位置同定コンポーネントの転送 ● １２.１.９赤外線受信モジュールのビルド～１２.２.１４テストコンポーネントのビルド の修正		寺井
		32 頁～	● ３.３.１２データ不正モード ● ３.４.１１データ不正モード ● ６.１.２環境自己位置同定モジュールのインストール～６.１.７LibRtcStr ライブラリのビルド ● ６.２.２環境自己位置同定モジュールのインストール～６.２.３共通ライブラリのビルド		

			● ９．１．２nkf のインストール確認～９．１．８ LibRtcStr ライブラリのビルド ● ９．２．２nkf のインストール確認～９．２．８ LibRtcStr ライブラリのビルド ● １２．１．２nkf のインストール確認～１２．１．８ LibRtcStr ライブラリのビルド ● １２．２．３nkf のインストール確認～１２．２．９ LibRtcStr ライブラリのビルド の追加		
1.6	2009/10/08		● 2.5.5ucode 解決データキャッシュを SQLite に変更 ● 3.3.9ucode 解決データキャッシュに ucode データキャッシュ(データベース)を追加 ● ６.２.３共通ライブラリのビルドにLibSqlite3を追加 ● 9.2.9LibSqlite3 ライブラリのビルドの追加 ● １０.２.５環境自己位置同定コンポーネントの転送の 10. に共有ライブラリの転送とシステム設定の変更を追加 ● １２.２.２.１POSIX ライブラリのインストール、１２.２.２.１omniORB ライブラリのインストール、１２.２.２.２ACE ライブラリのインストール、１２.２.２.３OpenRTM ライブラリのインストールの追加 ● １２.２.５LibRtcIniFile ライブラリのビルド、１２.２.６LibRtcLog ライブラリのビルド、１２.２.７ LibRtcRs232c ライブラリのビルド、１２.２.８ LibRtcSemaphore ライブラリのビルド、１２.２.９LibRtcStr ライブラリのビルドの修正 ● １２.２.１０LibSqlite3 ライブラリのビルドの追加 ● １２.２.１２UCODE 解決コンポーネントのビルドのビルドディレクトリ修正		寺井
1.7	2009/12/10		● ３.１.１、３.２.１、３.３.１、３.４.１の Module version を 1.1.0 に変更 ● ３．５緯度経度海抜型構造体定義（Position.idl）IDL 定義を変更		寺井

＜目次＞

はじめに

本書は、「搭乗用移動知能の構築を簡便にするモジュール群の開発～環境インフラと連動するパーソナルモビリティ～」（次世代ロボット知能化技術開発プロジェクト：⑥移動知能（社会・生活分野）の開発）における「環境自己位置同定モジュール（ucode モジュール）」の解説書である。

本書では、以下の内容を解説する。

- 環境自己位置同定モジュールの仕様
  赤外線を受信する機器となるユビキタスコミュニケーター（以降：UCと示す）と、赤外線を送信する機器となる赤外線マーカを用いて、現在の自己位置を特定する方式について解説する。
  １章で環境自己位置同定モジュールの概要、２章でモジュールの機能設計、３章でRTコンポーネント仕様(インタフェース設計)について述べる。
- 環境自己位置同定モジュールの環境構築、起動手順
  ４章、７章、１０章で実行環境の構築手順、５章、８章、１１章でコンポーネントの起動手順、６章、７章、１２章で開発環境の構築手順について述べる。

関連文書

No.	文書名	発行元	発行日	版数	文書番号
1	「次世代ロボット知能化技術開発プロジェクト，⑥移動知能（社会・生活分野）の開発～環境インフラと連動するパーソナルモビリティ～」に係る委託業務実施計画書(平成 19 年度～平成 23 年度)	芝浦工業大学 千葉工業大学 NEC ソフト(株)	-	-	-
2	The Robotic Technology Component Specification	Object Management Group	2007/8/18	-	-
3	ユビキタス ID アーキテクチャ	T-Engine フォーラム，ユビキタスIDセンター	-	-	-
4	ucode ユビキタスコード	T-Engine フォーラム，ユビキタスIDセンター	-	-	-
5	ucode 解決プロトコル(標準版)	T-Engine フォーラム，ユビキタスIDセンター	-	-	-
6	ucode 解決プロトコル(簡易版)	T-Engine フォーラム，ユビキタスIDセンター	-	-	-
7	ucode 解決ゲートウェイ	T-Engine フォーラム，ユビキタスIDセンター	-	-	-
8	T-Kernel 仕様書（Ver1.00.00）	T-Engine フォーラム，ユビキタスIDセンター	-	-	-
9	IrDA 型赤外線タグ仕様	国土交通省　国土技術政策総合研究所　道路空間高度化研究室	2007/03/23	-	-
10	Robotic Localization Service RFP	Object Management Group	2007/6/25	-	-

用語定義

- **RT ミドルウェア(RT-Middleware: RTM)**
  ロボット機能要素(RT 機能要素)のソフトウェアモジュールを複数組み合わせてロボットシステム(RT システム)を構築するためのソフトウェアプラットフォーム。

- **RT コンポーネント(RT-Component: RTC)**
  RT 機能要素をソフトウェアモジュール化したもの。RT コンポーネントには、他のコンポーネントとデータをやり取りしたり、通信したりするためのポートと呼ばれるインタフェースがあり、複数のコンポーネントのポートをつなぎ合わせ、それぞれの RT コンポーネント機能の集合体として構築される。

- **OpenRTM-aist**
  産業技術総合研究所・知能システム研究部門が実装・配布・メンテナンスを行っている RT ミドルウェアの実装例。C++言語および Python 言語用の実装を提供している。

- **CORBA(Common Object Request Broker Architecture)**
  Object management Group(OMG)が定義した標準規格であり、様々なコンピュータ上で様々なプログラミング言語で書かれたソフトウェアコンポーネントの相互利用にするための基盤技術。

- **T-Engine**
  携帯情報端末など比較的高度なユーザインタフェースを持つ機器のための開発用プラットフォーム。CPU ボードのサイズは 75mm x 120mm と規格で決められている。LCD、拡張ボードなどを接続できるようになっている。具体的なパッケージとして各種 CPU に対応した「T-Engine 開発キット(パーソナルメディア株式会社)」がある。

- **T-Kernel**
  T-Engine 用の組み込みオペレーティングシステム。従来からの ITRON と同様、スタティックメモリアロケーションによるカーネルベースでのプログラミングが可能。しかし、T-Engine 本来の目的である「ミドルウェアの流通」を実現するためには、ダイナミックメモリアロケーションが可能なプロセスベースでのプログラミングが可能な T-Kernel Extension を使うことが望まれる。

- **ucode**
  実世界上の識別対象それぞれに固有に割り付ける識別子

- **ucode 関係データベース(UCR Database)**
  ucode 間または ucode と情報の関係を管理するデータベースのこと

- **ユビキタスコミュニケーター（UC）**
  人間とユビキタス環境との効率的なコミュニケーションを実現する携帯情報ツール。RFID タグ（詳細は以下参照）の認識情報を持ち、E コマース（詳細は以下参照）、自律的移送支援など、幅広い応用が可能となる機器。ユビキタスコンピューティングの基盤研究所である YRP ユビキタスネットワーキング研究所（詳細は以下参照）が開発元である。

- **RFID タグ**
  ID 情報を埋め込んだタグから、電磁界や電波などを用いた近距離（周波数帯によって数 cm～数 m）の無線通信によって情報をやりとりするもの、および技術全般を指す。IC タグそのものを指したり、パッシブタイプの IC タグのみを指したりすることもある。

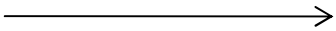
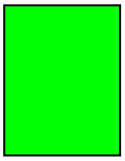
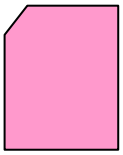
- **ロバスト性**
  ロバスト（Robust）は、頑健さのことを示す。外乱や設計誤差などの不確定な変動に対して、システム特性が現状を維持できること。

- **OMG**
  オブジェクト指向技術の標準化、普及をすすめるため、1989 年に設立された業界団体。オブジェクトと呼ばれるソフトウェア部品をネットワーク上に分散配置し、ハードウェアや OS、プログラミング言語の違いに関係なく、相互にデータや処理要求を交換するための基盤となるソフトウェア(ORB と呼ばれる)の共通仕様「CORBA」を策定し、普及をはかっている。

表記規則

本書では、オブジェクト指向を用いたシステム開発で使用される UML(Unified Modeling Language)の表記法を使用している。UML 図の表記は OMG により定められた UML 2.0 に準拠する。

また，本書における図で使用する矢印、および RT コンポーネントの表記規則を以下に定める。

No.	表記	意味
1		データフロー・値変更を表す
2		制御・関数呼び出し・コマンドフローを表す
3		ユーザ操作を表す
4		RT コンポーネントを表す
5		RT コンポーネントの入力ポートを表す
6		RT コンポーネントの出力ポートを表す
7		RT コンポーネントのサービスポートを表す
8		他の RT コンポーネントを表す 本書では、RT コンポーネントとの接続の例としての掲載であるため、詳細な記述を行わない

1. 概要

1.1. 実現する知能ロボットモジュール

移動知能ロボットの『本質安全・自己保全』『運用・安全』知能を実現するためには，自己位置検出機能の確実性が要求される。一般に自己位置を検出するにはオドメトリとGPSの情報を利用するが，地下道などを長距離移動する場合は，自己位置検出精度が大きく低下する。『環境自己位置同定モジュール』(以降，本モジュール)では，従来のような内界・外界センサ以外に，環境インフラ情報である ucode を積極利用することで，屋内外においてシームレスに位置情報を獲得可能とし，移動知能のロバスト性向上を目指す。

ucode は国土交通省が進めている自立移動支援プロジェクトにも適用されている環境インフラ情報コードである。銀座 4 丁目地下街および街外，および上野公園一帯ではすでに ucode タグが設置済みであり，実証実験も行われている。本モジュールでは ucode アーキテクチャに準拠した認識モジュールを実装することで，既存の環境インフラ情報を利用した自己位置同定が可能である。

なお本モジュールは ucode 格納機器として IrDA 型赤外線タグを対象し，赤外線信号を受信することにより自己位置同定を行う機能を有する。

2. 機能設計

2.1. ハードウェア構成(Windows 版･UNIX 版)

本書で定義する機能を実現するためのハードウェア構成を図 2-1に示す。

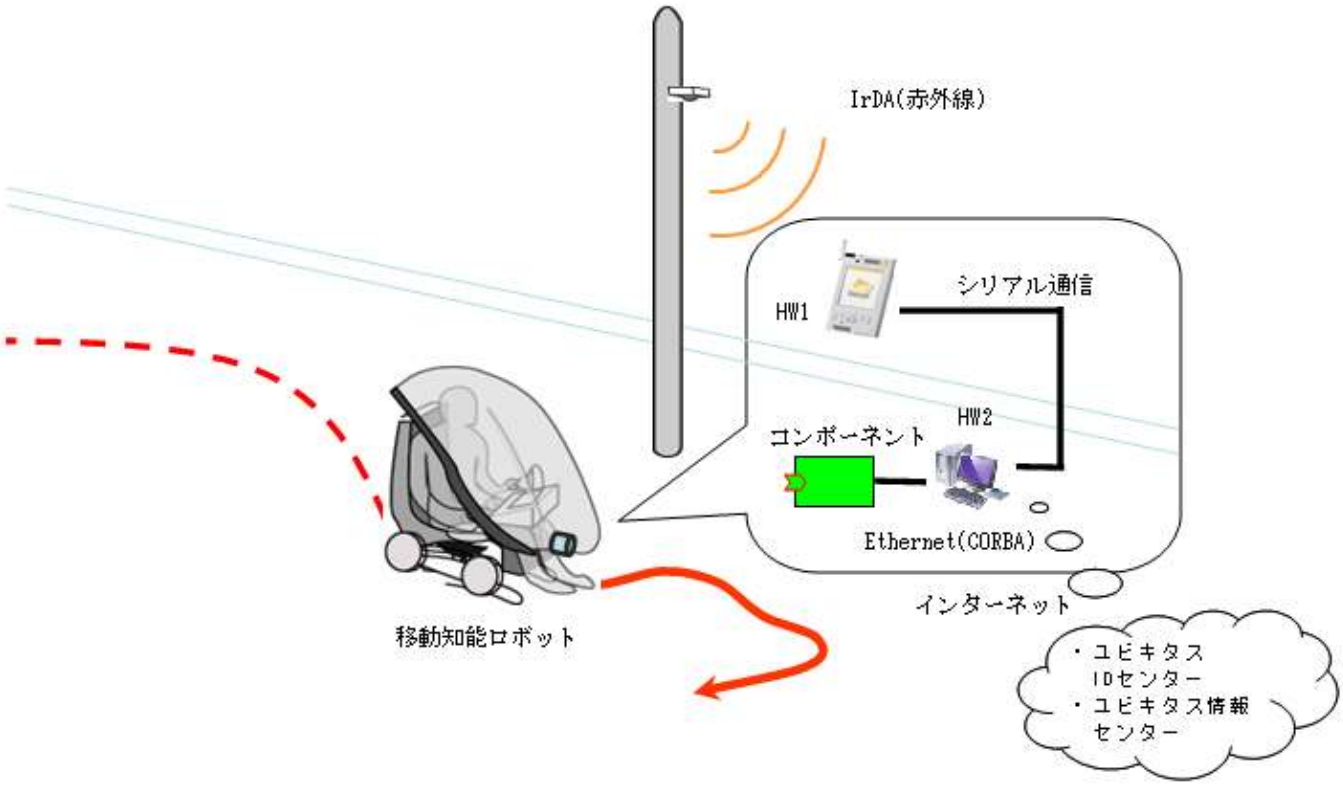


図 2-1 HW 構成(Windows 版･UNIX 版)

表 2-1 HW の詳細スペック(Windows 版・UNIX 版)

No.	名称		備考
HW1	ユビキタスコミュニケーター		
	CPU	ルネサステクノロジ SH7727 (SH3-DSP,144MHz)	
	フラッシュメモリ	8M byte	
	SDRAM	32M byte	
	ディスプレイ	TFT カラー, VGA(640x480), 262,144 色, タッチパネル	
	内臓通信機能	無線 LAN, Bluetooth	
	非接触 I/F	13.56MHz(eTRON カード対応) 2.45GHz(RFID タグ μチップ対応)	
	カード I/F	SD×1, miniSD×1, SIM×1	
	赤外線 I/F	入力×1, 出力×1	
	カメラ	CMOS カメラ 30 万画素 CCD カメラ 200 万画素	
	音声	ステレオスピーカ, マイク, ヘッドフォンステレオ	
	専用 ASIC	動画＆静止画アクセラレータ, ビデオキャプチャ, 暗号処理	
	寸法	縦 151mm x 横 78mm x 高さ 22mm (突起物除く)	
	重量	約 245g	
HW2	PC/AT 互換 PC		I/O ポートに USB か RS232C(シリアル)が必須

2.2. ソフトウェア構成（Windows 版・UNIX 版）

2.2.1. ソフトウェア環境（HW1）

HW1 のソフトウェア構成を図 2-2に示す。

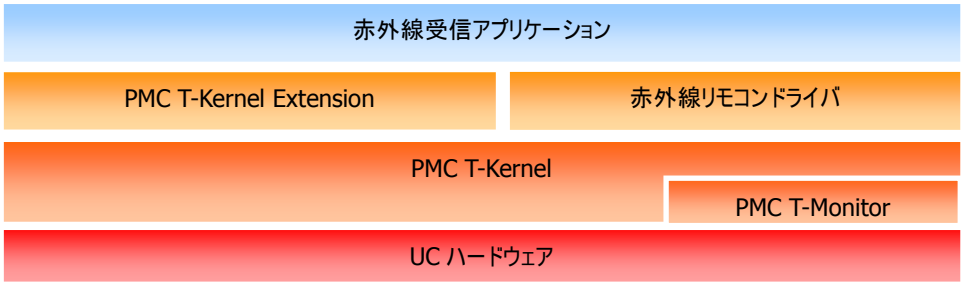


図 2-2 SW 構成(Windows 版・UNIX 版)

表 2-2 SW 詳細(Windows 版・UNIX 版)

No.	名称	バージョン	備考
SW1-1	PMC T-Monitor		UC 評価キット付属
SW1-2	PMC T-Kernel		UC 評価キット付属
SW1-3	PMC T-Kernel Extension		UC 評価キット付属
SW1-4	赤外線リモコンドライバ		UC 評価キット付属
SW1-5	赤外線受信アプリケーション		詳細については2.5.1、3.1で説明

2.2.2. ソフトウェア環境(HW2)

HW2 のソフトウェア構成を図 2-3に示す。

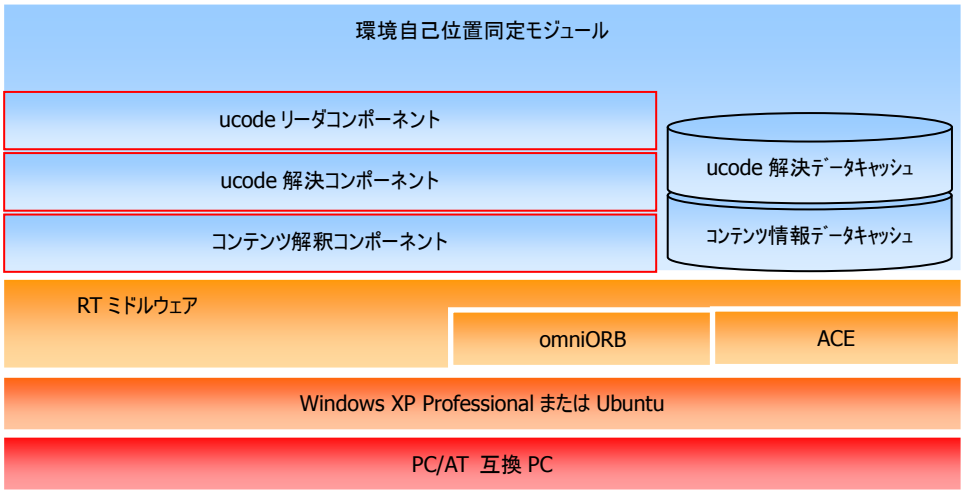


図 2-3 SW 構成(Windows 版･UNIX 版)

表 2-3 SW 詳細(Windows 版･UNIX 版)

No.	名称		備考
SW2-1	Windows 版の場合、 Windows XP Proffessional UNIX 版の場合、Ubuntu8.04		
SW2-2	RT ミドルウェア		OpenRTM-aist の環境一式を指す。RT ミドルウェアには omniORB、ACE の機能が含まれる。
SW2-3	環境自己位置同定モジュール	ucode リーダコンポーネント	詳細については、3.2を参照のこと
		ucode 解決コンポーネント	詳細については、2.5.2、3.3を参照のこと
		コンテンツ解釈コンポーネント	詳細については、2.5.4、3.4を参照のこと
SW2-4	ucode 解決データキャッシュ		詳細については、2.5.5を参照のこと。
SW2-5	コンテンツ情報データキャッシュ		詳細については、2.5.6を参照のこと。

2.3. ハードウェア構成（Teacube 版）

本書で定義する機能を実現するためのハードウェア構成を図 2-4に示す。

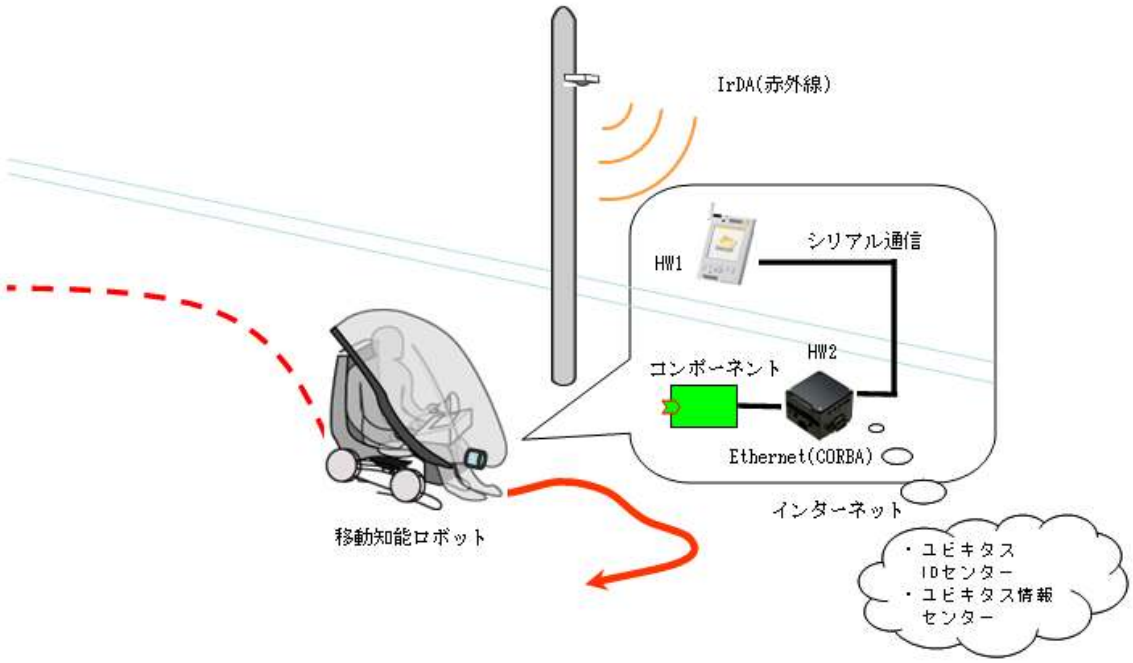


図 2-4 HW 構成（Teacube 版）

表 2-4 HW の詳細スペック(Teacube 版)

No.	名称		備考
HW1	ユビキタスコミュニケーター		
	CPU	ルネサステクノロジ SH7727 (SH3-DSP,144MHz)	
	フラッシュメモリ	8M byte	
	SDRAM	32M byte	
	ディスプレイ	TFT カラー, VGA(640x480), 262,144 色, タッチパネル	
	内臓通信機能	無線 LAN, Bluetooth	
	非接触 I/F	13.56MHz(eTRON カード対応) 2.45GHz(RFID タグ μチップ対応)	
	カード I/F	SD×1, miniSD×1, SIM×1	
	赤外線 I/F	入力×1, 出力×1	
	カメラ	CMOS カメラ 30 万画素 CCD カメラ 200 万画素	
	音声	ステレオスピーカ, マイク, ヘッドフォンステレオ	
	専用 ASIC	動画＆静止画アクセラレータ, ビデオキャプチャ, 暗号処理	
	寸法	縦 151mm x 横 78mm x 高さ 22mm (突起物除く)	
	重量	約 245g	
HW2	Teacube/VR5701		左記 HW スペックを満たす組込みボード
	CPU	NEC エレクトロニクス VR5701 (MIPS コア, 最大 266 / 333 MHz)	
	フラッシュメモリ	16 M byte	
	SDRAM	64M byte	
	入出力 I/F	USB(Host) ×2	
		RS232C(シリアル) ×2	
		コンパクトフラッシュ(IDE)	
		10/100Base-T	
		外部 CRT 出力 (最大 1280×1024 ドット 65536 色)	
		eTRON	
		ヘッドフォン出力	
		マイク入力	
	その他機能	RTC(Real Time Clock)	
	電源	AC アダプタ	
	寸法	縦 52mm×横 52mm×高さ 45mm	
	重量	165mm	

2.4. ソフトウェア構成（Teacube 版）

2.4.1. ソフトウェア環境（HW1）

HW1 のソフトウェア構成を図 2-5に示す。

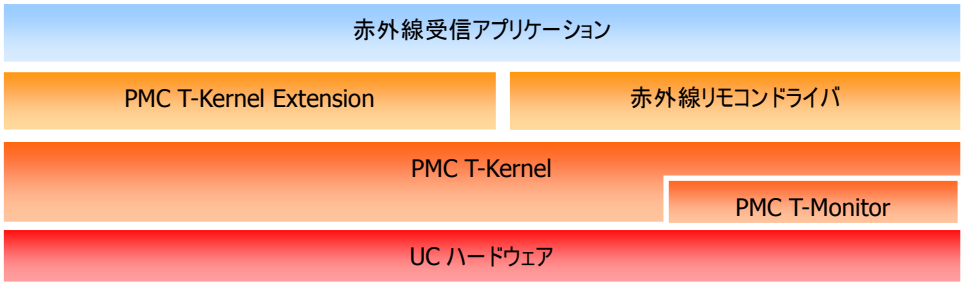


図 2-5 SW 構成（Teacube 版）

表 2-5 SW 詳細（Teacube 版）

No.	名称	バージョン	備考
SW1-1	PMC T-Monitor		UC 評価キット付属
SW1-2	PMC T-Kernel		UC 評価キット付属
SW1-3	PMC T-Kernel Extension		UC 評価キット付属
SW1-4	赤外線リモコンドライバ		UC 評価キット付属
SW1-5	赤外線受信アプリケーション		詳細については2.5.1、3.1で説明

2.4.2. ソフトウェア環境(HW2)

HW2 のソフトウェア構成を図 2-6に示す。

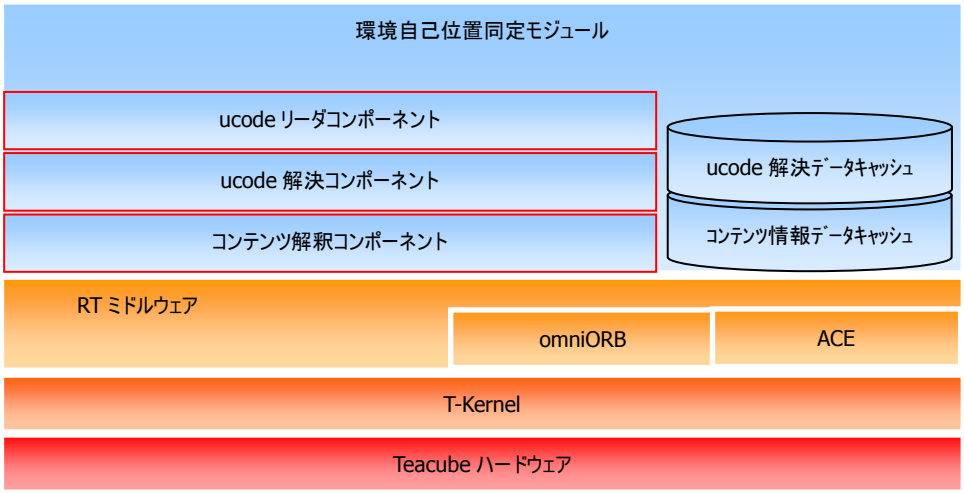


図 2-6 SW 構成(Teacube 版)

表 2-6 SW 詳細(Teacube 版)

No.	名称		備考
SW2-1	T-Kernel		
SW2-2	RT ミドルウェア		T-Kernel 上で実装する OpenRTM-aist の環境一式を指す。RT ミドルウェアには omniORB、ACE の機能が含まれる。
SW2-3	環境自己位置同定モジュール	ucode リーダコンポーネント	詳細については、3.2を参照のこと
		ucode 解決コンポーネント	詳細については、2.5.2、3.3を参照のこと
		コンテンツ解釈コンポーネント	詳細については、2.5.4、3.4を参照のこと
SW2-4	ucode 解決データキャッシュ		詳細については、2.5.5を参照のこと。
SW2-5	コンテンツ情報データキャッシュ		詳細については、2.5.6を参照のこと。

2.5. 機能仕様

2.5.1. ucode 受信機能

IrDA 型赤外線タグより発信された ucode を受信する機能を有する。IrDA 型赤外線タグより発信される光信号，および赤外線タグフレームフォーマットは関連文書9(IrDA 型赤外線タグ仕様)に従う。

受信した ucode は2.5.2記載の ucode 解決にて使用される。

2.5.2. ucode 解決機能

2.5.1の機能により取得した ucode を ucode 解決する機能を有する。

ucode 解決とは、現実世界のモノと情報とを管理する ucode 関係データベースから、状況に応じた適切な情報を取得するための機能である。本モジュールではこの ucode 解決機能を ucodeに対応する位置情報が定義されたコンテンツのデータキャッシュファイルパスを取得する用途で使用する。

2.5.3. 位置情報コンテンツ取得機能

2.5.2の機能により取得したコンテンツのデータキャッシュファイルパスを基に、コンテンツ情報を取得する機能を有する。コンテンツ情報中に位置情報を含めることで自己位置を同定することが可能である。

2.5.4. コンテンツ解釈機能

コンテンツ情報を解釈し、コンテンツ情報より自己位置を取得する機能を有する。取得した自己位置情報は The Robotic Technology Component Specification に従い，CORBA インタフェースにより外部 RT コンポーネントに対し出力可能である。

2.5.5. ucode 解決データキャッシュ

2.5.2で述べた ucode 解決データキャッシュの詳細について以下に示す。

ucode 解決データキャッシュは、データベース(SQLite)に格納する。

表 2-7 ucode 解決データテーブル(ucode_resolve)

項目(カラム名)	項目(和名)	型	主キー	備考
ucode	ucode	text	○	
ucode_resolve_info	ucode 解決情報	text		

ucode 解決データキャッシュには初期起動時にデータが登録されていません。
その為、ucode 解決データキャッシュに登録するデータを ucode 解決データキャッシュファイルから取り込み、
ucode 解決データキャッシュに登録します。

ucode 解決データキャッシュファイルは、カンマ区切りのテキスト形式で表現する。

ucode,ucode_resolve_info とする。

表 2-8 ucode 解決データ定義

フィールド名	意味	説明
ucode	ucode	128bit ucode 値の 16 進文字列表現
ucode_resolve_info	ucode 解決情報	コンテンツキャッシュファイルパス

```
00000000000000000000000000101E3,./Contents/00000000000000000000000000101E3
00000000000000000000000000010164,./Contents/00000000000000000000000000010164
```

図 2-7 ucode 解決データ例

2.5.6. コンテンツ情報データキャッシュ

2.5.3で述べたコンテンツ情報データキャッシュの詳細について以下に示す。

コンテンツ情報データキャッシュは、カンマ区切りのテキスト形式で表現する。

lat,lng,level,margin とする。

表 2-9 コンテンツ情報データ定義

フィールド名	意味	説明
lat	緯度	センサの緯度 浮動小数点形式で表記する。
lng	経度	センサの経度 浮動小数点形式で表記する
level	海抜	センサの海抜 浮動小数点形式で表記する。
margin	誤差	自己位置からセンサまでの誤差 メートル単位で表記する。

```
141.121212,43.154554,512,0
```

図 2-8 コンテンツ情報データ例

2.6. モジュール構成図

本モジュールのモジュールソフトウェア構成図を図 2-9に示す。

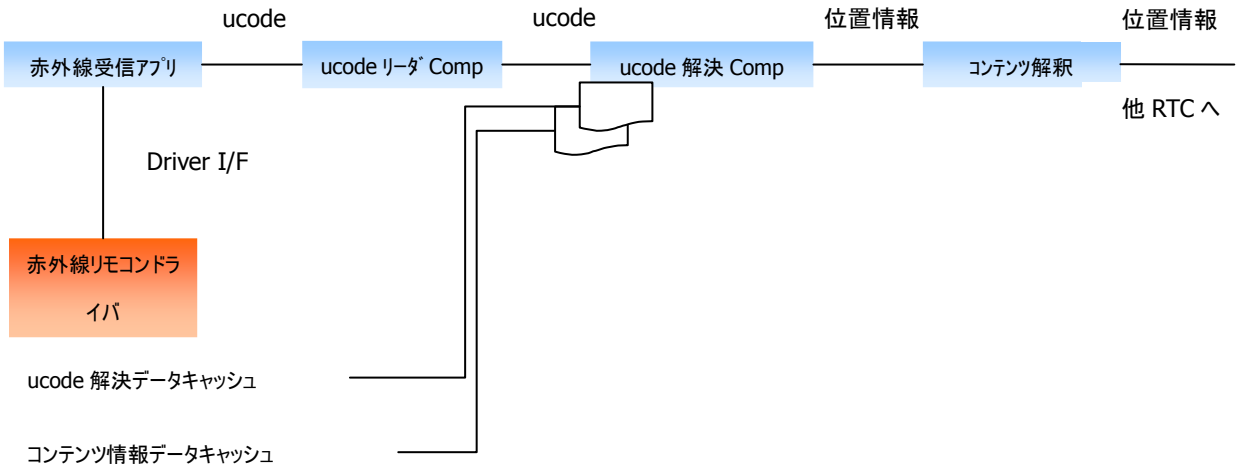


図 2-9 環境自己位置同定モジュールソフトウェア構成詳細

- 赤外線受信アプリケーション
  赤外線センサからの ucode を受信する機能。
  受信した ucode をシリアル通信で送信する機能

- ucode リーダコンポーネント
  赤外線受信アプリケーションより ucode をシリアル通信で受信する機能。

- ucode 解決コンポーネント
  ucode リーダコンポーネントより ucode を取得し、ucode 解決を行い、解決結果を基にコンテンツ情報を取得する。
  ucode 解決結果、コンテンツ情報の取得はデータキャッシュより取得する。

- コンテンツ解釈コンポーネント
  ucode 解決コンポーネントよりコンテンツ情報を取得し、コンテンツ情報より任意のデータを取得する。

- ucode 解決データキャッシュ
  2.5.5の図 2-7 ucode 解決データ例の様な形で設定する。

- コンテンツデータキャッシュ
  2.5.6の図 2-8 コンテンツ情報データ例の様な形で設定する。

3. コンポーネント仕様(インタフェース設計)

3.1. 赤外線受信アプリケーション仕様

3.1.1. モジュール定義

表 3-1 モジュール定義

Module name	IrReceiveLook
Module version	1.1.0
Module vender	NEC Soft Ltd

3.1.2. シリアル通信仕様

シリアルで送信するデータ形式を以下に示す

<STX>X-UIDC-UCODE=xxxx<ETX>

※<STX>:0x02 <ETX>:0x03 xxxx:ucode の値

図 3-1 シリアル通信データ形式

3.1.3. アクティビティ図

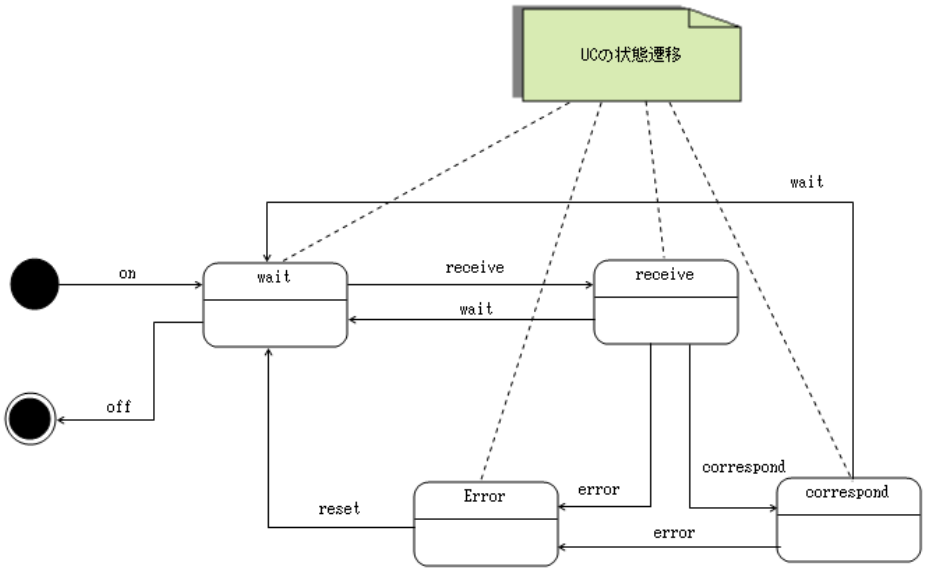


図 3-2 アクティビティ図

3.1.4. 設定ファイル

赤外線受信アプリケーションの設定ファイル仕様を以下に示す。

- 設定ファイル: IrReceive.ini
- 設置場所: IrReceive アプリケーションと同一階層

設定項目: 下表の通り

設定ファイルが上記の設置場所にない場合は、以下の表 3-2の説明にあるデフォルトの値になります。

表 3-2 赤外線受信アプリケーション設定ファイル

パラメータ	意味	説明
LOG_FILE_PATH	ログファイルディレクトリパス	デフォルト: /SYS/Log/ 指定されたパスとファイル名でファイル作成に失敗した場合は、ログ出力は行われません。
LOG_FILE_NAME	ログファイル名	13 文字以内 デフォルト: IrReceive 指定されたファイル名が 13 文字より長い場合は、デフォルトになります。
LOG_FILE_MODE	ログファイルモード	1:番号 2:日単位 デフォルト:1 1,2 以外の場合は、デフォルトの値になります。 必ず、1 を指定してください。
LOG_FILE_MAX_NUM	ログファイル最大数	1～7 の範囲 デフォルト: 3 範囲外の場合は、デフォルトの値になります。
LOG_FILE_COUNTER	ログファイルカウンタ	1～ログファイル最大数の範囲 デフォルト:1 範囲外の場合は、デフォルトの値になります。
LOG_FILE_OUTPUT	ログファイル出力可否	0:OFF 1:ON デフォルト: 0 (OFF) 0,1 以外の場合は、デフォルトの値になります。
CONSOLE_OUTPUT	コンソール出力可否	0:OFF 1:ON デフォルト: 0 (OFF) 0,1 以外の場合は、デフォルトの値になります。
PORT	シリアルポート	デフォルト:rsa 指定されたポートのオープンに失敗した場合は、IrReceiveLook は起動しません。
BAUD_RATE	シリアル設定 ボーレート	300, 600,1200,2400,4800,9600,19200,38400, 57600,115200 デフォルト 115200
DATA	シリアル設定 データ	0:7bit 1:8bit デフォルト:1 範囲外の場合は、デフォルトの値になります。
PARITY	シリアル設定 パリティ	0:なし 1:奇数 2:偶数 デフォルト:0 範囲外の場合は、デフォルトの値になります。
STOP	シリアル設定 ストップ	0:1bit 1:2bit デフォルト:0

```
* ログファイル
* ログファイルディレクトリパス
LOG_FILE_PATH=/SYS/log/
* ログファイル名
LOG_FINE_NAME=IrReceive
* ログファイルモード(1:番号 2:日単位)
LOG_FILE_MODE=1
* ログファイル作成最大数(1～7)
LOG_FILE_MAX_NUM=3
* ログファイルカウンタ
LOG_FILE_COUNTER=1
* ログファイル出力(0:OFF 1:ON)
LOG_FILE_OUTPUT=1
* コンソール出力(0:OFF 1:ON)
CONSOLE_OUTPUT=1
* シリアルポート設定
* ポート
PORT=rsa
* ボーレート
BAUD_RATE=115200
* データ(0:7bit 1:8bit)
DATA=3
* パリティ(0:なし 1:奇数 2:偶数)
PARITY=0
* ストップ(0:1bit 1:2bit)
STOP=0
```

図 3-3 赤外線受信アプリケーション設定ファイル例

3.1.5. ログファイル

赤外線受信アプリケーションのログファイル仕様を以下に示す。

- ログファイル:
  - ✧ ログファイルモードが番号の場合

    ファイル名:設定ファイル指定ファイル名-ログファイルカウンタ

    起動ごとに作成

    設定ファイル指定最大数分だけ作成される。(最大数に達した場合古いファイルから削除)
  - ✧ ログファイルモードが日単位の場合

    ファイル名:設定ファイル指定ファイル名-YYMMDD (YYMMDD は年月日)

    日付ごとに作成 (起動した時点で同日のログファイルがある場合は追記)

    設定ファイル指定最大数分だけ作成される。(最大数に達した場合古いファイルから削除)
- 設置場所: 設定ファイル指定ディレクトリパス以下

3.2. ucode リーダコンポーネント仕様

3.2.1. コンポーネント定義

表 3-3 コンポーネント定義

Module name	RtcUcdReceive
Module description	Ucode Reader Component
Module version	1.1.0
Module vender	NEC Soft Ltd
Module category	Generic
Component type	COMMUTATIVE
Component’s activity type	EVENT_DRIVEN
Number of maximum instance	1

3.2.2. コンポーネント図

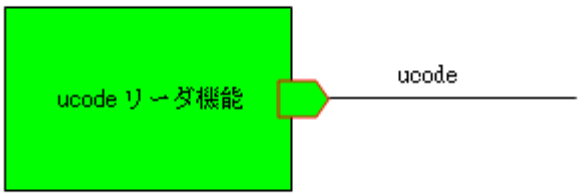


図 3-4 コンポーネント図

3.2.3. 入力ポート定義

本コンポーネントは入力ポートを持たない。

3.2.4. 出力ポート定義

本コンポーネントは以下の出力ポートを持つ。

表 3-4 出力ポート定義

Name	Type	説明
Ucode	TimedString	ucode 解決コンポーネントに渡す

```
0000000000000000000000000010164
```

図 3-5 出力ポートのデータ例

3.2.5. サービスプロバイダポート定義

本コンポーネントはサービスプロバイダを持たない。

3.2.6. サービスコンシューマポート定義

本コンポーネントはサービスコンシューマを持たない。

3.2.7. コンフィグレーションセット定義

本コンポーネントは以下のコンフィグレーションセットを持つ。

表 3-5 コンフィグレーションセット定義

Name	Property Name	Type	説明
default	outputtime	unsigned int	同一 UCODE 受信時の OutPort への出力周期(秒) 0～25 の範囲 範囲外の値を適用された場合は反映されません。
log	console	unsigned int	コンソール出力可否 0:OFF 1:ON 0，1 以外の値を適用された場合は反映されません。
	logfile	unsigned int	ログ出力可否 0:OFF 1:ON 0，1 以外の値を適用された場合は反映されません。
serial	port	std::string	シリアルポート
	baudrate	unsigned int	ボーレート 300，600,1200,2400,4800,9600,19200,38400, 57600,115200 上記以外の値を適用された場合は反映されません。
	data	unsigned int	データ 0:7bit 1:8bit 0，1 以外の値を適用された場合は反映されません。
	parity	unsigned int	パリティ 0:なし 1:奇数 2:偶数 0，1，2 以外の値を適用された場合は反映されません。
	stopbits	unsigned int	ストップ 0:1bit 1:2bit 0，1 以外の値を適用された場合は反映されません。
	timeout	unsigned int	シリアル通信タイムアウト時間（ミリ秒） 0～25000 の範囲 範囲外の値を適用された場合は反映されません。

3.2.8. シリアル通信仕様

シリアルで受信するデータ形式を以下に示す

<STX>X-UIDC-UCODE=xxxx<ETX>

※<STX>:0x02 <ETX>:0x03 xxxx:ucode の値

図 3-6 シリアル通信データ形式

3.2.9. 設定ファイル

ucode リーダコンポーネントの設定ファイル仕様を以下に示す。

- 設定ファイル: UcdReceive.ini
- 設置場所: RtcUcdReceive コンポーネントと同一階層

設定項目: 下表の通り

設定ファイルが上記の設置場所にない場合は、以下の表 3-6の説明にあるデフォルトの値になります。

表 3-6 ucode リーダコンポーネント設定ファイル

パラメータ	意味	説明
LOG_FILE_PATH	ログファイルディレクトリパス	デフォルト: ./log/ 指定されたパスとファイル名でファイル作成に失敗した場合は、ログ出力は行われません。
LOG_FILE_NAME	ログファイル名	Teacube 版:13 文字以内 デフォルト: UcdReceive Teacube 版:指定されたファイル名が 13 文字より長い場合は、デフォルトになります。
LOG_FILE_MODE	ログファイルモード	1:番号 2:日単位 デフォルト:2 1,2 以外の場合は、デフォルトの値になります。
LOG_FILE_MAX_NUM	ログファイル最大数	1～7 の範囲 デフォルト: 3 範囲外の場合は、デフォルトの値になります。
LOG_FILE_COUNTER	ログファイルカウンタ	1～ログファイル最大数の範囲 デフォルト:1 範囲外の場合は、デフォルトの値になります。
LOG_FILE_OUTPUT	ログファイル出力可否	0:OFF 1:ON デフォルト: 0 (OFF) 0, 1 以外の場合は、デフォルトの値になります。
CONSOLE_OUTPUT	コンソール出力可否	0:OFF 1:ON デフォルト: 0 (OFF) 0, 1 以外の場合は、デフォルトの値になります。
PORT	シリアルポート	Windows 版デフォルト:COM1 UNIX 版デフォルト:/dev/ttyS0 Teacube 版デフォルト:rsa 指定されたポートのオープンに失敗した場合は、RtcUcdReceive は起動しません。
BAUD_RATE	シリアル設定 ボーレート	300, 600,1200,2400,4800,9600,19200,38400, 57600,115200 デフォルト 115200
DATA	シリアル設定 データ	0:7bit 1:8bit デフォルト:1 範囲外の場合は、デフォルトの値になります。
PARITY	シリアル設定 パリティ	0:なし 1:奇数 2:偶数 デフォルト:0 範囲外の場合は、デフォルトの値になります。
STOP	シリアル設定 ストップ	0:1bit 1:2bit デフォルト:0
READ_TIMEOUT	シリアル通信タイムアウト時間(ミリ秒)	0～25000 の範囲 デフォルト:3000
OUT_CYCLE	同一 UCODE を受信時の出力周期(秒)	同一 UCODE を受信した場合に OutPort へ出力する周期。 0～25 の範囲 デフォルト:5

```
* ログファイル
* ログファイルディレクトリパス
LOG_FILE_PATH=./log/
* ログファイル名
LOG_FILE_NAME=UcdReceive
* ログファイルモード(1:番号 2:日単位)
LOG_FILE_MODE=2
* ログファイル作成最大数(1～7)
LOG_FILE_MAX_NUM=3
* ログファイルカウンタ
LOG_FILE_COUNTER=1
* ログファイル出力(0:OFF 1:ON)
LOG_FILE_OUTPUT=1
* コンソール出力(0:OFF 1:ON)
CONSOLE_OUTPUT=1
* シリアルポート
PORT=rsa
* ボーレート
BAUD_RATE=115200
* データ(0:7bit 1:8bit)
DATA=1
* パリティ(0:なし 1:奇数 2:偶数)
PARITY=0
* ストップ(0:1bit 1:2bit)
STOP=0
* シリアル通信タイムアウト時間
READ_TIMEOUT=5000
* 同じ ucode を受信した場合の time 設定(秒)
OUT_CYCLE=10
```

図 3-7 ucode リーダコンポーネント設定ファイル例

3.2.10. ログファイル

ucode リーダコンポーネントのログファイル仕様を以下に示す。

- ログファイル:
  - ✧ ログファイルモードが番号の場合
    ファイル名:設定ファイル指定ファイル名-ログファイルカウンタ
    起動ごとに作成
    設定ファイル指定最大数分だけ作成される。(最大数に達した場合古いファイルから削除)
  - ✧ ログファイルモードが日単位の場合
    ファイル名:設定ファイル指定ファイル名-YYMMDD (YYMMDD は年月日)
    日付ごとに作成 (起動した時点で同日のログファイルがある場合は追記)
    設定ファイル指定最大数分だけ作成される。(最大数に達した場合古いファイルから削除)
- 設置場所: 設定ファイル指定ディレクトリパス以下

3.2.11. アクティビティ図

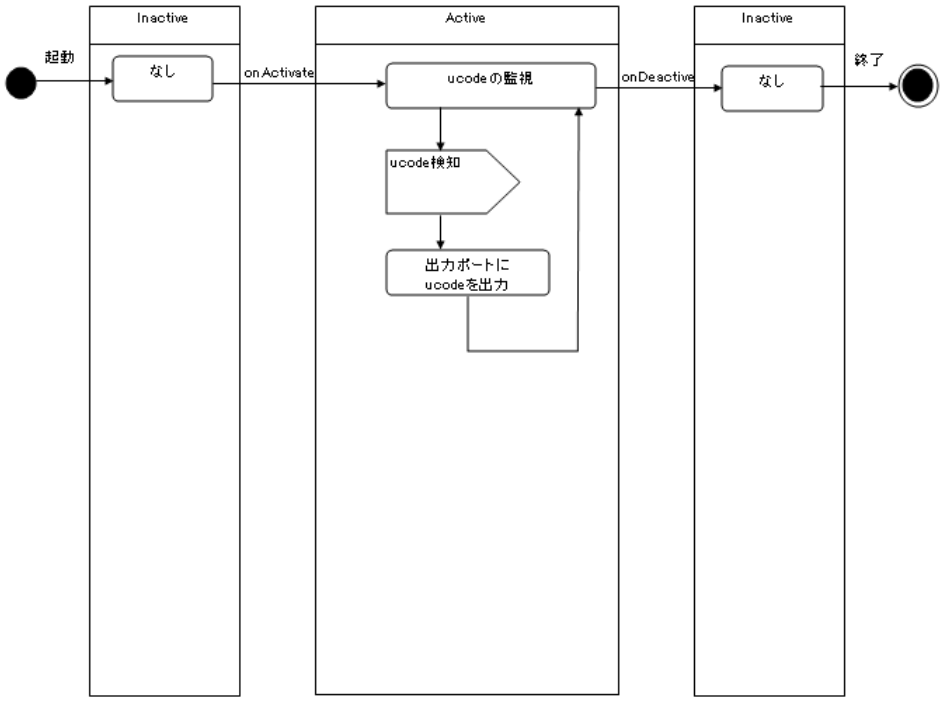


図 3-8 アクティビティ図

- アクティブ状態:
  定期的に赤外線受信アプリケーションからシリアル通信で送信される ucode の監視を繰り返し行う。
  ucode を検知した場合は ucode を取得し、そのまま3.2.4で定義した出力ポートより ucode を出力する。

  エラー時の処理
  - シリアル通信で受信したデータが不正の場合はログファイルにエラーを出力し、出力ポートには何も出力しない。

- 非アクティブ状態:
  本モジュールは何も行わない。

- エラー状態:
  本モジュールは何も行わない。

3.3. ucode 解決コンポーネント仕様

3.3.1. コンポーネント定義

表 3-7 コンポーネント定義

Module name	RtcUcdResolve
Module description	Ucode Resolve Component
Module version	1.1.0
Module vender	NEC Soft Ltd
Module category	Generic
Component type	COMMUTATIVE
Component's activity type	EVENT_DRIVEN
Number of maximum instance	1

3.3.2. コンポーネント図

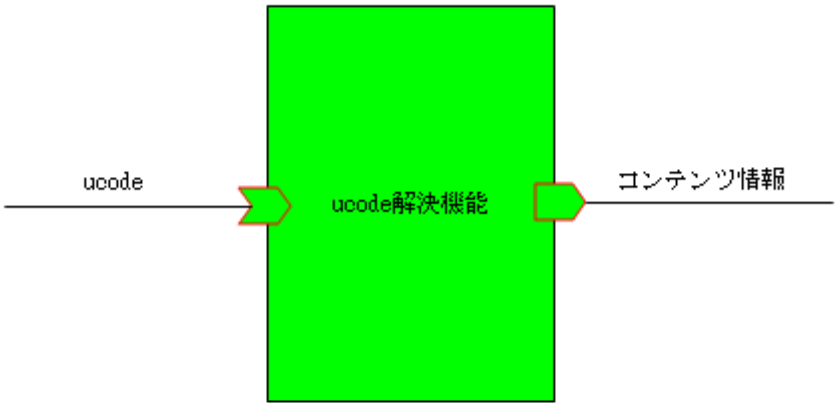


図 3-9 コンポーネント図

3.3.3. 入力ポート定義

本コンポーネントは以下の入力ポートを持つ。

表 3-8 入力ポート定義

Name	Type	説明
ucode	TimedString	ucode リーダコンポーネントより取得した ucode

```
0000000000000000000000000000010164
```

図 3-10 入力ポートのデータ例

3.3.4. 出力ポート定義

本コンポーネントは以下の出力ポートを持つ。

表 3-9 出力ポート定義

Name	Type	説明
contents	TimedString	コンテンツ情報（位置情報）

```
141.22222222,43.11111111,123,1
```

図 3-11 出力ポートのデータ例

3.3.5. サービスプロバイダポート定義

本コンポーネントはサービスプロバイダを持たない。

3.3.6. サービスコンシューマポート定義

本コンポーネントはサービスコンシューマを持たない。

3.3.7. コンフィグレーションセット定義

本コンポーネントは以下のコンフィグレーションセットを持つ。

表 3-10 コンフィグレーションセット定義

Name	Parameter Name	Type	説明
default	illegal_data_mode	std::string	データ不正モード normal または strict normal, strict 以外で適用された場合は反映されません。
log	console	unsigned int	コンソール出力可否 0:OFF 1:ON 0, 1 以外の値を適用された場合は反映されません。
	logfile	unsigned int	ログ出力可否 0:OFF 1:ON 0, 1 以外の値を適用された場合は反映されません。

3.3.8. 設定ファイル

Ucode 解決コンポーネントの設定ファイル仕様を以下に示す。

- 設定ファイル: UcdResolve.ini
- 設置場所: RtcUcdResolve コンポーネントと同一階層

設定項目: 下表の通り

設定ファイルが上記の設置場所にない場合は、以下の表 3-11の説明にあるデフォルトの値になります。

表 3-11 ucode 解決コンポーネント設定ファイル

パラメータ	意味	説明
LOG_FILE_PATH	ログファイルディレクトリパス	デフォルト: ./log/ 指定されたパスとファイル名でファイル作成に失敗した場合は、ログ出力は行われません。
LOG_FILE_NAME	ログファイル名	Teacube 版:13 文字以内 デフォルト: UcdReceive Teacube 版:指定されたファイル名が 13 文字より長い場合は、デフォルトになります。
LOG_FILE_MODE	ログファイルモード	1:番号 2:日単位 デフォルト:2 1,2 以外の場合は、デフォルトの値になります。
LOG_FILE_MAX_NUM	ログファイル最大数	1～7 の範囲 デフォルト: 3 範囲外の場合は、デフォルトの値になります。
LOG_FILE_COUNTER	ログファイルカウンタ	1～ログファイル最大数の範囲 デフォルト:1 範囲外の場合は、デフォルトの値になります。
LOG_FILE_OUTPUT	ログファイル出力可否	0:OFF 1:ON デフォルト: 0 (OFF) 0, 1 以外の場合は、デフォルトの値になります。
CONSOLE_OUTPUT	コンソール出力可否	0:OFF 1:ON デフォルト: 0 (OFF) 0, 1 以外の場合は、デフォルトの値になります。
ILLEGAL_DATA_MODE	データ不正モード	normal または strict デフォルト:normal データ不正モードの詳細は3.3.12を参照のこと

```
* ログファイル
* ログファイルディレクトリパス
LOG_FILE_PATH=./log/
* ログファイル名
LOG_FILE_NAME=UcdResolve
* ログファイルモード(1:番号 2:日単位)
LOG_FILE_MODE=2
* ログファイル作成最大数(1～7)
LOG_FILE_MAX_NUM=3
* ログファイルカウンタ
LOG_FILE_COUNTER=1
* ログファイル出力(0:OFF 1:ON)
LOG_FILE_OUTPUT=1
* コンソール出力(0:OFF 1:ON)
CONSOLE_OUTPUT=1
* データ不正モード(normal または strict)
ILLEGAL_DATA_MODE=normal
```

図 3-12 ucode 解決コンポーネント設定ファイル例

3.3.9. ucode 解決データキャッシュ

ucode データキャッシュの仕様を以下に示す。

- DB ファイル: UcdResolveDB
- 設置場所: RtcUcdResolve コンポーネントと同一階層

設定項目: 詳細に関しては2.5.5を参照のこと

ucode データキャッシュファイルの仕様を以下に示す。

- 設定ファイル: UcdResolve.txt
- 設置場所: RtcUcdResolve コンポーネントと同一階層

設定項目: 詳細に関してはを参照のこと

3.3.10. コンテンツ情報データキャッシュ

ucode データキャッシュの仕様を以下に示す。

- 設定ファイル: ucode 解決データキャッシュのコンテンツキャッシュファイルパスに記載
- 設置場所: ucode 解決データキャッシュのコンテンツキャッシュファイルパスに記載

設定項目: 詳細に関しては2.5.6を参照のこと

3.3.11. ログファイル

ucode 解決コンポーネントのログファイル仕様を以下に示す。

- ログファイル:
  - ✧ ログファイルモードが番号の場合
    ファイル名:設定ファイル指定ファイル名-ログファイルカウンタ
    起動ごとに作成
    設定ファイル指定最大数分だけ作成される。(最大数に達した場合古いファイルから削除)
  - ✧ ログファイルモードが日単位の場合
    ファイル名:設定ファイル指定ファイル名-YYMMDD (YYMMDD は年月日)
    日付ごとに作成 (起動した時点で同日のログファイルがある場合は追記)
    設定ファイル指定最大数分だけ作成される。(最大数に達した場合古いファイルから削除)
- 設置場所: 設定ファイル指定ディレクトリパス以下

3.3.12. データ不正モード

不正なデータがある場合のコンポーネントの状態遷移を決めるモード

- strict
  不正なデータを検出した場合、コンポーネントをエラー状態に遷移させるモード
- normal
  不正なデータを検出した場合でもコンポーネントはエラー状態に遷移せずアクティブ状態のまま動作させるモード

3.3.13. アクティビティ図

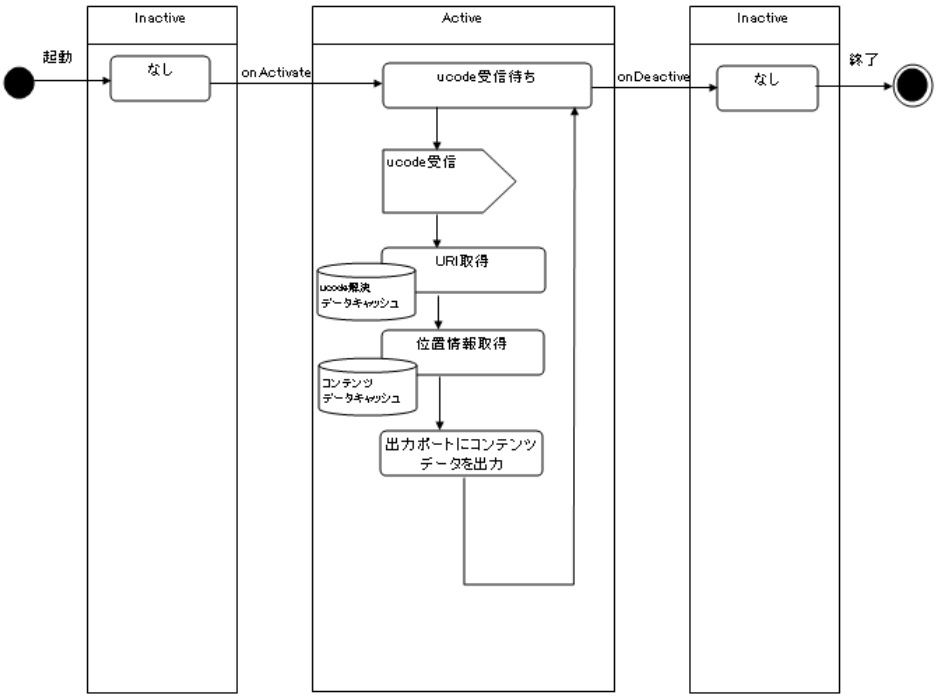


図 3-13 アクティビティ図

- アクティブ状態

  ucode リーダコンポーネントから出力される ucode の監視を繰り返し行う。

  ucode を取得した場合は、取得した ucode をキーに ucode データキャッシュから該当するコンテンツデータキャッシュファイルパスを取得する。

  コンテンツデータキャッシュからコンテンツ情報を取得し、3.3.4で定義した出力ポートよりコンテンツ情報を出力する。

  エラー時の処理

  - 入力ポートから受け取ったデータが不正の場合はログファイルにエラーを出力し、出力ポートには何も出力しない。
  - データ不正モードが strict の場合は、コンポーネントはエラー状態に遷移する。

- 非アクティブ状態の時は、本モジュールは何も行わない。

3.4. コンテンツ解釈コンポーネント仕様

3.4.1. コンポーネント定義

表 3-12 コンポーネント定義

Module name	RtcCntInterpret
Module description	Contents Interpret Component
Module version	1.1.0
Module vender	NEC Soft Ltd
Module category	Generic
Component type	COMMUTATIVE
Component’s activity type	EVENT_DRIVEN
Number of maximum instance	1

3.4.2. コンポーネント図

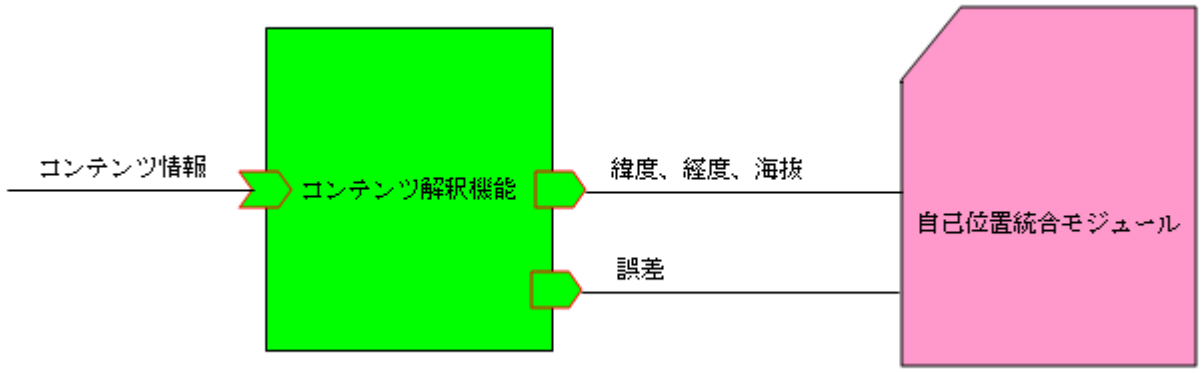


図 3-14 コンポーネント図

3.4.3. 入力ポート定義

本コンポーネントは以下の入力ポートを持つ。

表 3-13 入力ポート定義

Name	Type	説明
Contents	TimedString	コンテンツ情報（位置情報）

```
141.22222222,43.11111111,123,1
```

図 3-15 入力ポートのデータ例

3.4.4. 出力ポート定義

本コンポーネントは以下の出力ポートを持つ。

表 3-14 出力ポート定義

Name	Type	説明
Position	TimedPosition	センサの緯度経度海抜型構造体 3.5を参照のこと
margin	TimedDouble	自己位置からセンサまでの誤差

```
lat=141.22222222
lng=43.11111111
level=123.0
margin=1.0
```

図 3-16 出力ポートのデータ例

3.4.5. サービスプロバイダポート定義

本コンポーネントはサービスプロバイダを持たない。

3.4.6. サービスコンシューマポート定義

本コンポーネントはサービスコンシューマを持たない。

3.4.7. コンフィグレーションセット定義

本コンポーネントは以下のコンフィグレーションセットを持つ。

表 3-15 コンフィグレーションセット定義

Name	Parameter Name	Type	説明
default	illegal_data_mode	std::string	データ不正モード normal または strict normal, strict 以外で適用された場合は反映されません。
log	console	unsigned int	コンソール出力可否 0:OFF 1:ON 0, 1 以外の値を適用された場合は反映されません。
	logfile	unsigned int	ログ出力可否 0:OFF 1:ON 0, 1 以外の値を適用された場合は反映されません。

3.4.8. IDL 定義

緯度経度海抜型構造体定義仕様（Position.idl）は、3.5を参照のこと。

3.4.9. 設定ファイル

コンテンツ解決コンポーネントの設定ファイル仕様を以下に示す。

- 設定ファイル: CntInterpret.ini
- 設置場所: RtcCntInterpret コンポーネントと同一階層

設定項目: 下表の通り

設定ファイルが上記の設置場所にない場合は、以下の表 3-16の説明にあるデフォルトの値になります。

表 3-16 コンテンツ解釈コンポーネント設定ファイル

パラメータ	意味	説明
LOG_FILE_PATH	ログファイルディレクトリパス	デフォルト: ./log/ 指定されたパスとファイル名でファイル作成に失敗した場合は、ログ出力は行われません。
LOG_FILE_NAME	ログファイル名	Teacube 版:13 文字以内 デフォルト: UcdReceive Teacube 版:指定されたファイル名が 13 文字より長い場合は、デフォルトになります。
LOG_FILE_MODE	ログファイルモード	1:番号 2:日単位 デフォルト:2 1,2 以外の場合は、デフォルトの値になります。
LOG_FILE_MAX_NUM	ログファイル最大数	1～7 の範囲 デフォルト: 3 範囲外の場合は、デフォルトの値になります。
LOG_FILE_COUNTER	ログファイルカウンタ	1～ログファイル最大数の範囲 デフォルト:1 範囲外の場合は、デフォルトの値になります。
LOG_FILE_OUTPUT	ログファイル出力可否	0:OFF 1:ON デフォルト: 0 (OFF) 0, 1 以外の場合は、デフォルトの値になります。
CONSOLE_OUTPUT	コンソール出力可否	0:OFF 1:ON デフォルト: 0 (OFF) 0, 1 以外の場合は、デフォルトの値になります。
ILLEGAL_DATA_MODE	データ不正モード	normal または strict デフォルト:normal データ不正モードの詳細は3.4.11を参照のこと

```
* ログファイル
* ログファイルディレクトリパス
LOG_FILE_PATH=./log/
* ログファイル名
LOG_FILE_NAME=CntInterpret
* ログファイルモード(1:番号 2:日単位)
LOG_FILE_MODE=2
* ログファイル作成最大数(1～7)
LOG_FILE_MAX_NUM=3
* ログファイルカウンタ
LOG_FILE_COUNTER=1
* ログファイル出力(0:OFF 1:ON)
LOG_FILE_OUTPUT=1
* コンソール出力(0:OFF 1:ON)
CONSOLE_OUTPUT=1
* データ不正モード(normal または strict)
ILLEGAL_DATA_MODE=normal
```

図 3-17 コンテンツ解釈コンポーネント設定ファイル例

3.4.10. ログファイル

コンテンツ解釈コンポーネントの設定ファイル仕様を以下に示す。

- ログファイル:
  - ✧ ログファイルモードが番号の場合
    ファイル名:設定ファイル指定ファイル名-ログファイルカウンタ
    起動ごとに作成
    設定ファイル指定最大数分だけ作成される。(最大数に達した場合古いファイルから削除)
  - ✧ ログファイルモードが日単位の場合
    ファイル名:設定ファイル指定ファイル名-YYMMDD (YYMMDD は年月日)
    日付ごとに作成 (起動した時点で同日のログファイルがある場合は追記)
    設定ファイル指定最大数分だけ作成される。(最大数に達した場合古いファイルから削除)
- 設置場所: 設定ファイル指定ディレクトリパス以下

3.4.11. データ不正モード

不正なデータがある場合のコンポーネントの状態遷移を決めるモード

- strict
  不正なデータを検出した場合、コンポーネントをエラー状態に遷移させるモード
- normal
  不正なデータを検出した場合でもコンポーネントはエラー状態に遷移せずアクティブ状態のまま動作させるモード

3.4.12. アクティビティ図

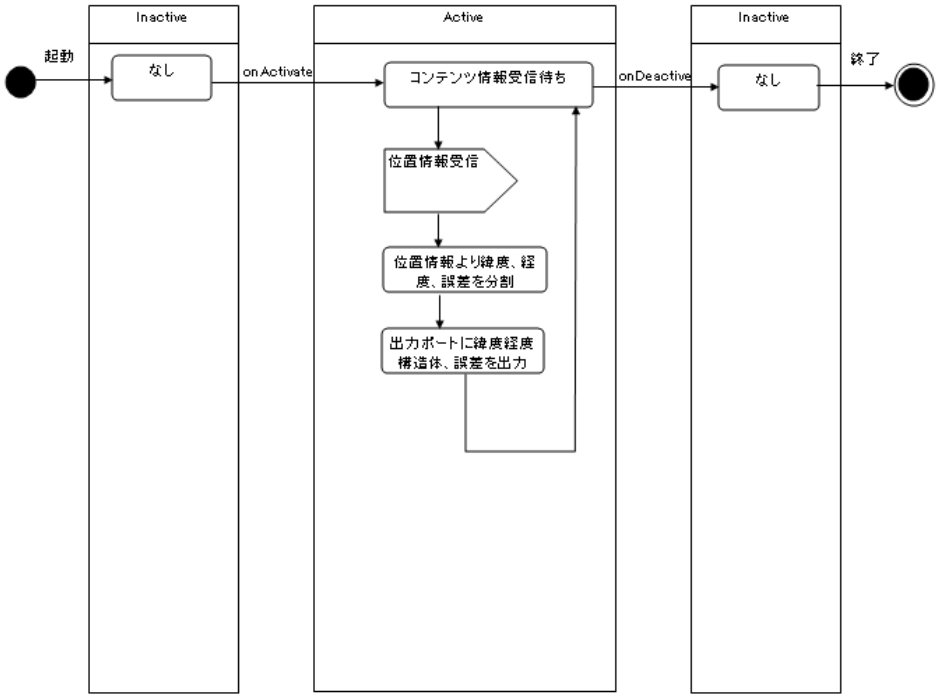


図 3-18 アクティビティ図

- アクティブ状態

  ucode 解決コンポーネントから出力されるコンテンツ情報の監視を繰り返し行う。

  コンテンツ情報を取得した場合は、取得した位置情報の内容を、緯度、経度、海抜、誤差に分割し、3.4.4で定義した出力ポートより緯度経度海抜型構造体、誤差を出力する。

  エラー時の処理

  - 入力ポートから受け取ったデータが不正の場合はログファイルにエラーを出力し、出力ポートには何も出力しない。
  - データ不正モードが strict の場合は、コンポーネントはエラー状態に遷移する。

- 非アクティブ状態の時は、本モジュールは何も行わない。

3.5. 緯度経度海抜型構造体定義（Position.idl）IDL 定義

表 3-17 緯度経度海抜型構造体（Position.idl）IDL 定義インタフェース

項目名	形式
緯度	double
経度	double
海抜	double
タイムスタンプ	Time
緯度経度海抜構造体配列	sequence<struct TimedPosition>

```
#ifndef POSITION_IDL
#define POSITION_IDL

//RTM 共用 IDL
#include "BasicDataType.idl"

module RTC
{
	//緯度経度定義基本構造体
	struct Position
	{
		double lat;		//緯度
		double lng;		//経度
		double level;		//海抜
	};

	//Timestamp 付緯度経度構造体
	struct TimedPosition
	{
		RTC::Time tm;	//タイムスタンプ
		Position data;	//緯度経度海抜定義基本構造体
	};

	//Timestamp 付緯度経度海抜 Sequencial 構造体
	struct TimedPositionSeq
	{
		RTC::Time tm;				//タイムスタンプ
		sequence<TimedPosition> data;	//緯度経度海抜定義基本構造体
	};
};
#endif
```

3.6. コンポーネント接続例

以下に、環境自己位置同定モジュールのコンポーネント接続例を示す。

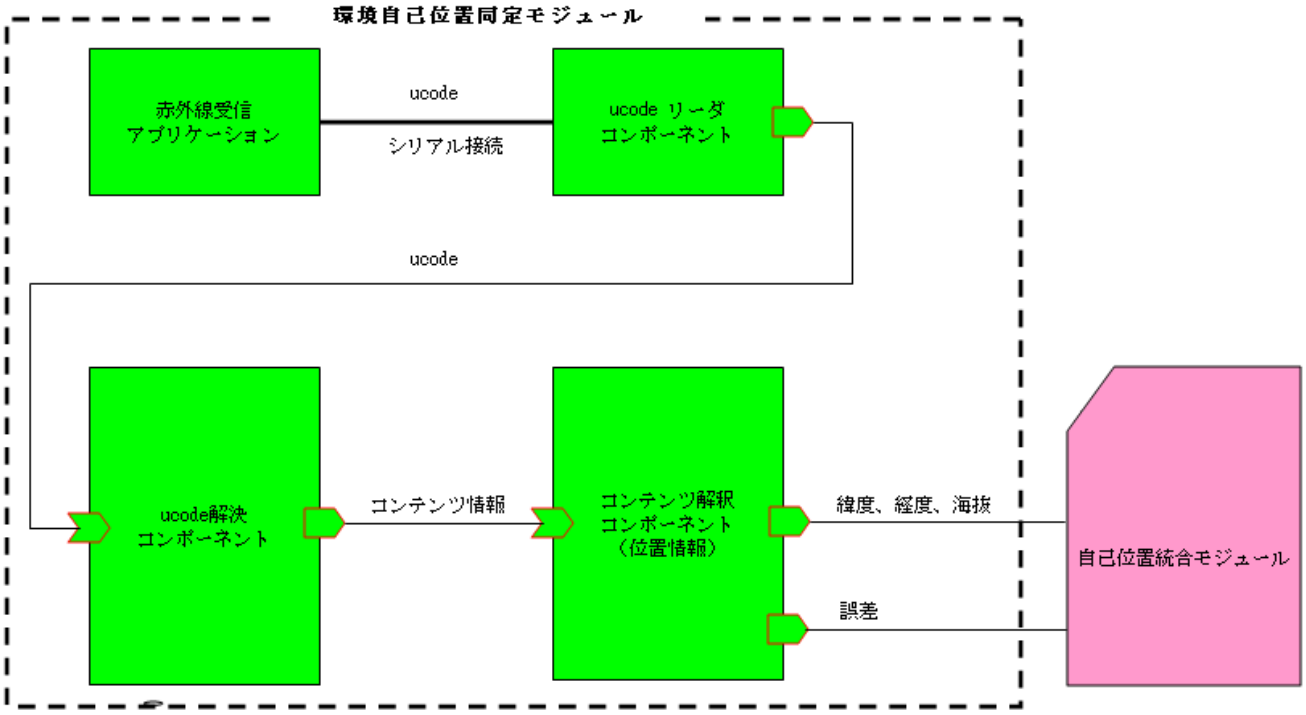


図 3-19 環境自己位置同定モジュール コンポーネント接続例

3.7. 各コンポーネントの処理シーケンス

3.7.1. 赤外線受信アプリケーションの ucode 取得シーケンス

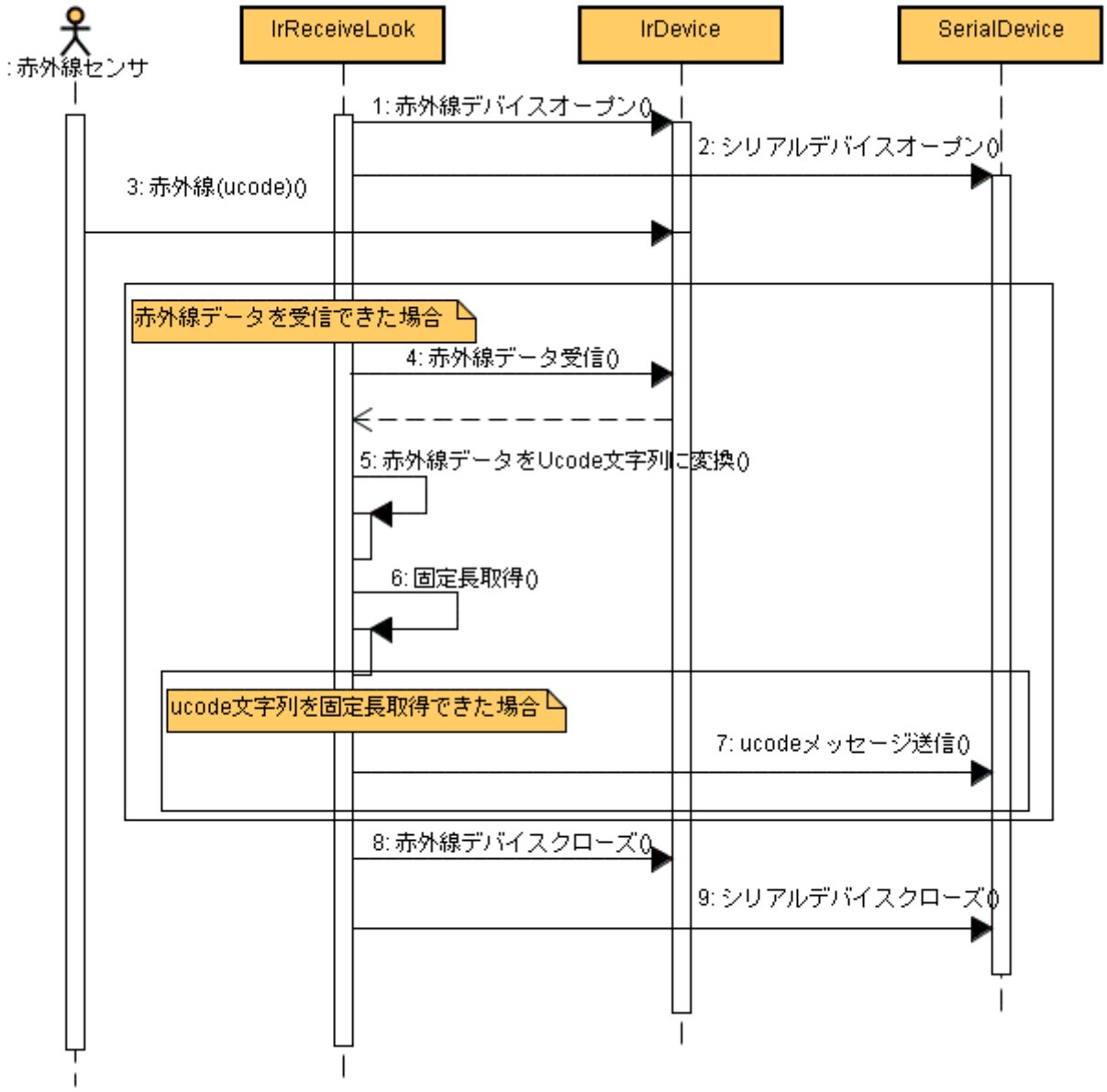


図 3-20 赤外線受信アプリケーションの ucode 取得シーケンス図

赤外線受信アプリケーションの ucode 取得シーケンス図の説明を以下に示す。

（1） IrReceiveLook が IrDevice のデバイスオープンを行う。
（2） IrReceiveLook が SerialDevice(UC のシリアル I/F)のデバイスオープンを行う。
（3） 赤外線センサから ucode 情報を含んだ赤外線が IrDevice(UC の赤外線 I/F)へ送られる。
（4） IrDevice が赤外線データを受信できた場合 IrReceiveLook へ赤外線データを渡す。
（5） IrReceiveLook が IrDevice から受け取った赤外線データを ucode 文字列(ASCII コード)に変換する。
（6） IrReceiveLook が ucode 文字列を固定長取得する。
（7） Ucode 文字列が固定長確保できた場合 IrReceiveLook が SerialDevice へ ucode メッセージを送信する。
（8） IrReceiveLook が IrDevice のデバイスクローズを行う。
（9） IrReceiveLook が SerialDevice のデバイスクローズを行う。

3.7.2. ucode リーダコンポーネントの起動シーケンス

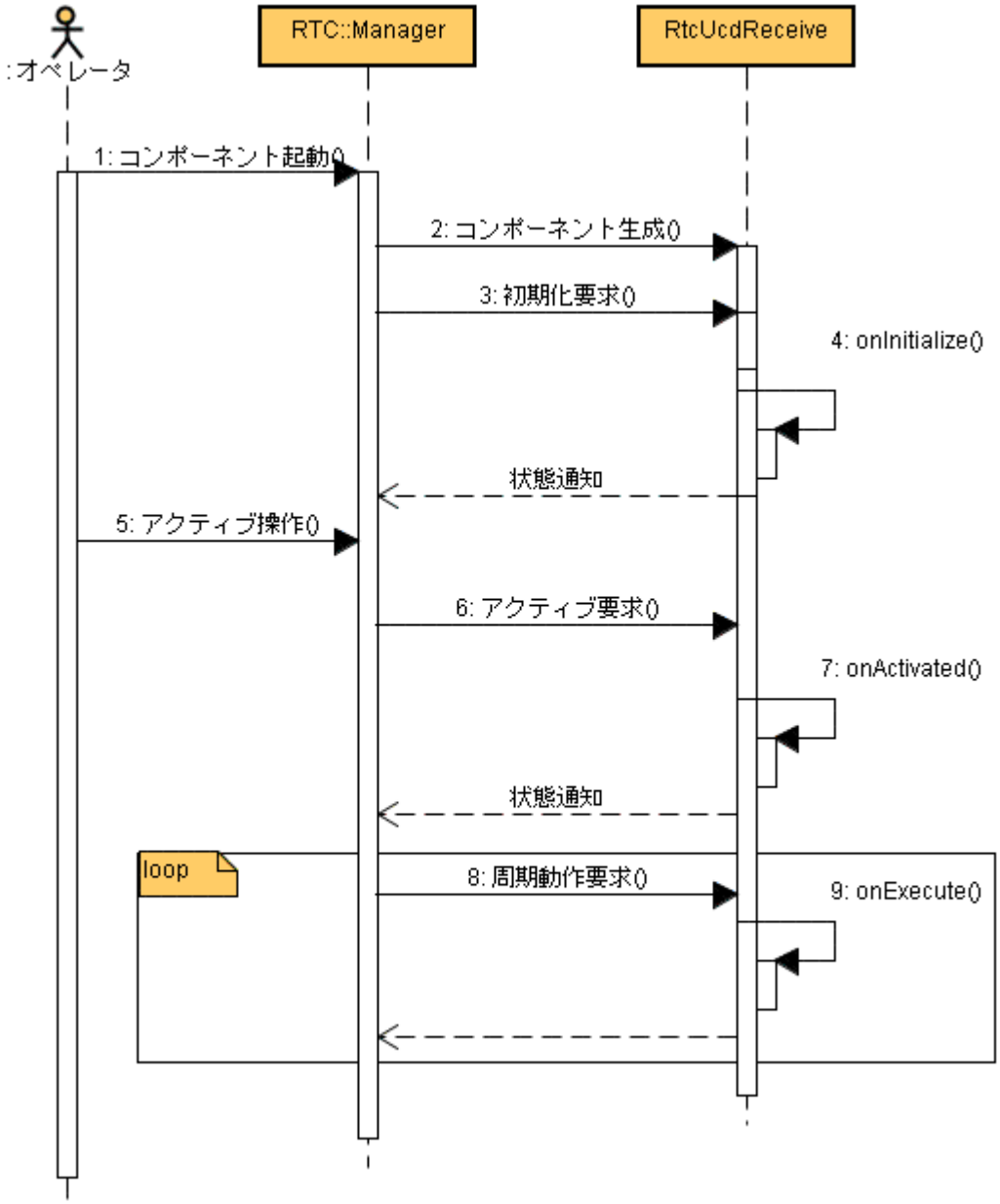


図 3-21 ucode リーダコンポーネントの起動シーケンス図

ucode リーダコンポーネントの起動シーケンス図の説明を以下に示す。

（1） オペレータからの起動コマンド
（2） （1）により、RTC::Maneger が ucode リーダコンポーネントの生成を行う。
（3） RTC::Maneger から RtcUcdReceive へ初期化要求を行う。
（4） RtcUcdReceive で初期化処理が行われ、状態を RTC::Maneger へ返す。
（5） オペレータからのアクティブ操作
（6） （5）により RTC::Maneger が RtcUcdReceive へアクティブ要求を行う。
（7） RtcUcdReceive でアクティブ処理が行われ、コンポーネントがアクティブな状態となる。状態を RTC::Maneger へ返す。
（8） コンポーネントがアクティブな状態かつ正常な状態である時、RTC::Maneger は RtcUcdReceive へ周期動作を要求し続ける。
（9） RtcUcdReceive の周期処理が行われる。状態を RTC::Maneger へ返す。

3.7.3. ucode リーダコンポーネントの動作シーケンス

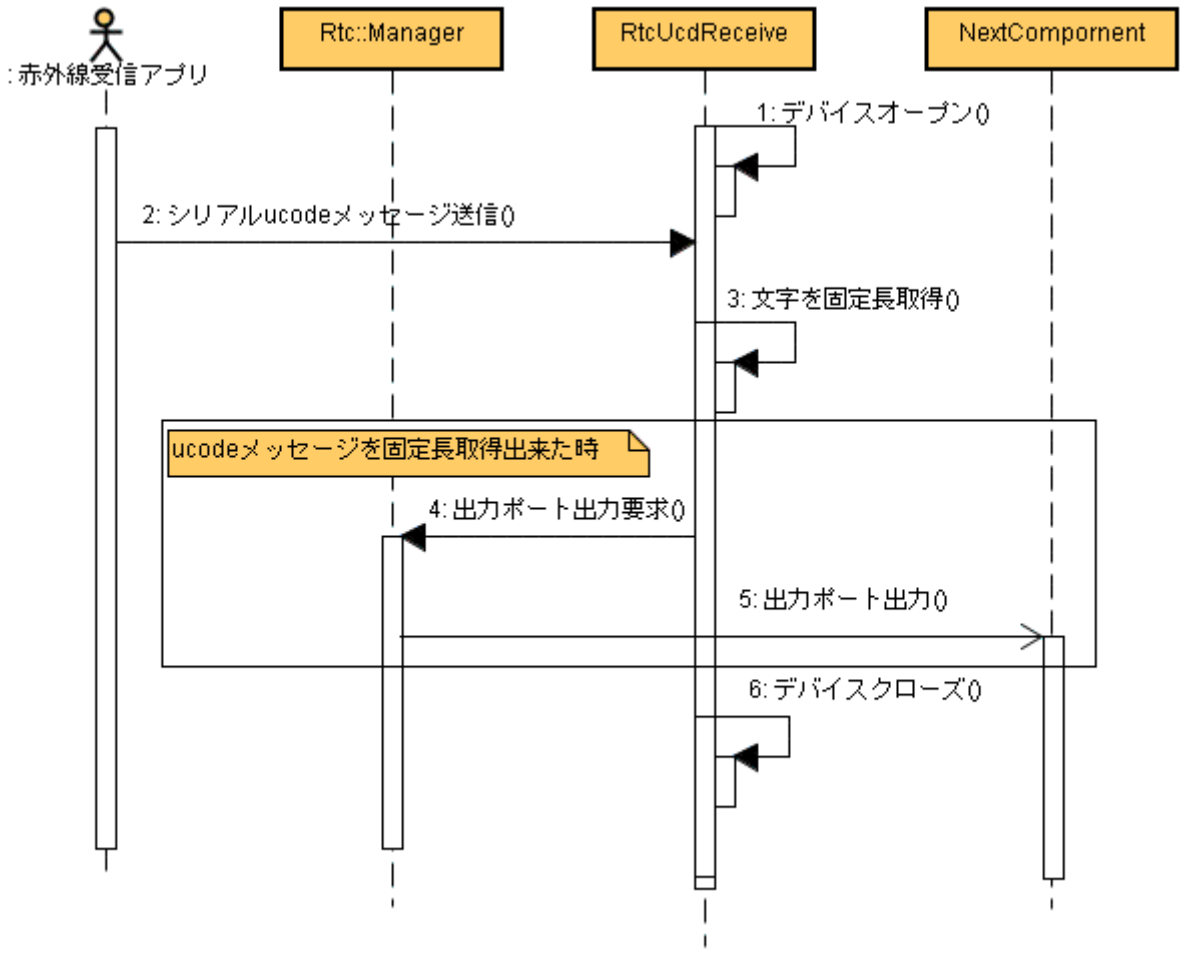


図 3-22 赤外線リーダコンポーネントの動作シーケンス図

ucode リーダコンポーネントの動作シーケンス図の説明を以下に示す。

（1） RtcUcdReceive が入力デバイス(シリアルポート)をオープンし、ucode 文字列が 1 文字受信されると受信データのリードを開始する。
（2） 赤外線受信アプリケーションから RtcUcdReceive へ ucode 文字列が送信される。
（3） RtcUcdReceive が ucode 文字列を固定長受信する。ucode 固定長は 32 桁の文字列とする。
（4） ucode 文字列を固定長取得した後、RtcUcdReceive が入力デバイスデータを初期化し、入力デバイスのクローズ処理を行う。
（5） 文字が固定長取得できている、かつ、正常な状態である時、RtcUcdReceive が ucode 文字列をバインドされた変数に格納し、RTC::Maneger へ出力ポート出力要求を行う。
（6） RTC::Maneger が出力ポートデータの出力を行う。

3.7.4. ucode 解決コンポーネントの起動シーケンス

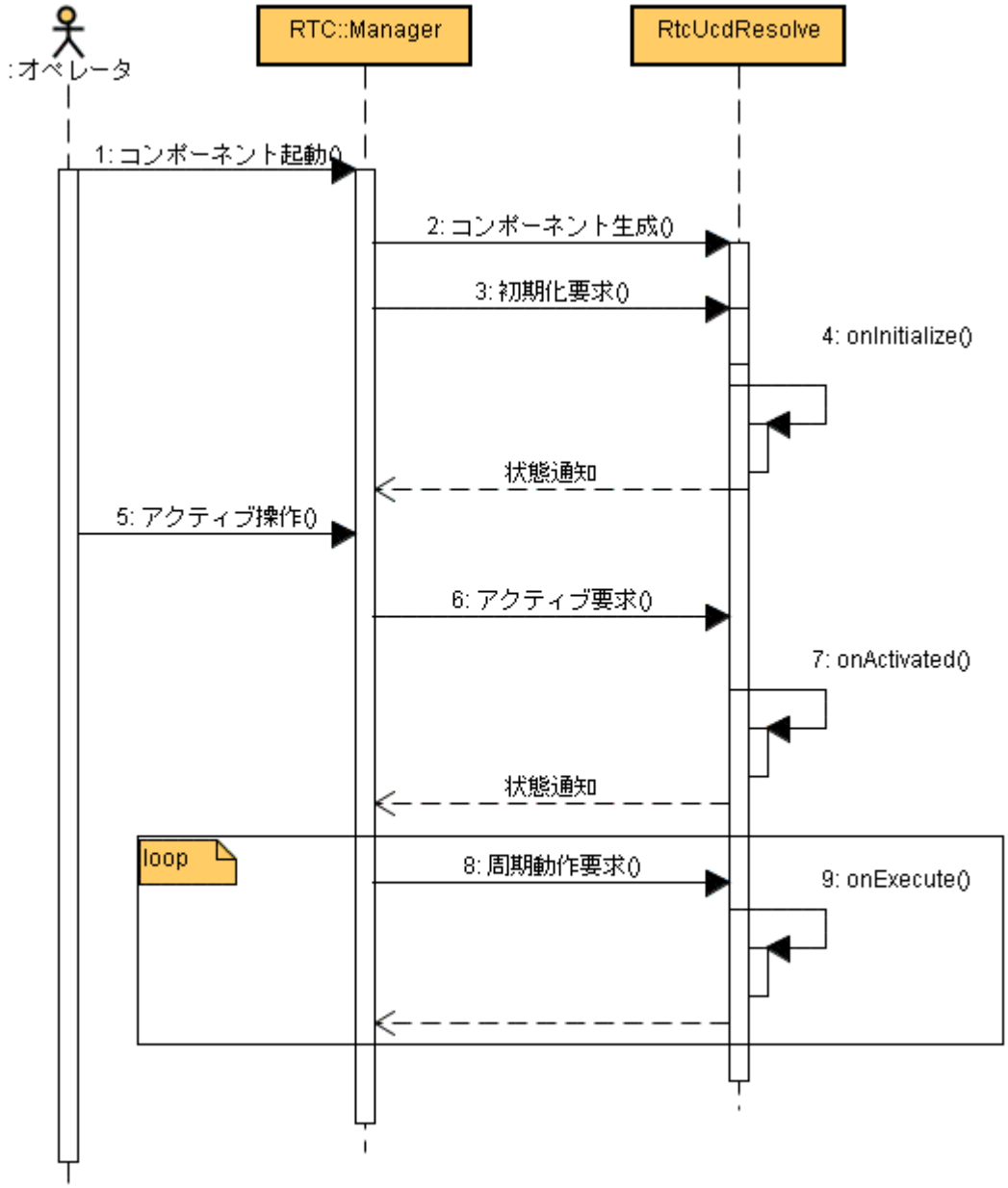


図 3-23 ucode 解決コンポーネントの起動シーケンス図

ucode 解決コンポーネントの起動シーケンス図の説明を以下に示す。

（1） オペレータからの起動コマンド
（2） （1）により、RTC::Maneger が ucode 解決コンポーネントの生成を行う。
（3） RTC::Maneger から RtcUcdResolve へ初期化要求を行う。
（4） RtcUcdResolve で初期化処理が行われ、状態を RTC::Maneger へ返す。
（5） オペレータからのアクティブ操作
（6） （5）により RTC::Maneger が RtcUcdResolve へアクティブ要求を行う。
（7） RtcUcdResolve でアクティブ処理が行われ、コンポーネントがアクティブな状態となる。状態を RTC::Maneger へ返す。
（8） アクティブな状態かつ正常な状態である時、RTC::Maneger は RtcUcdResolve へ周期動作を要求し続ける。
（9） RtcUcdResolve の周期処理が行われる。状態を RTC::Maneger へ返す。

3.7.5. ucode 解決コンポーネントの動作シーケンス

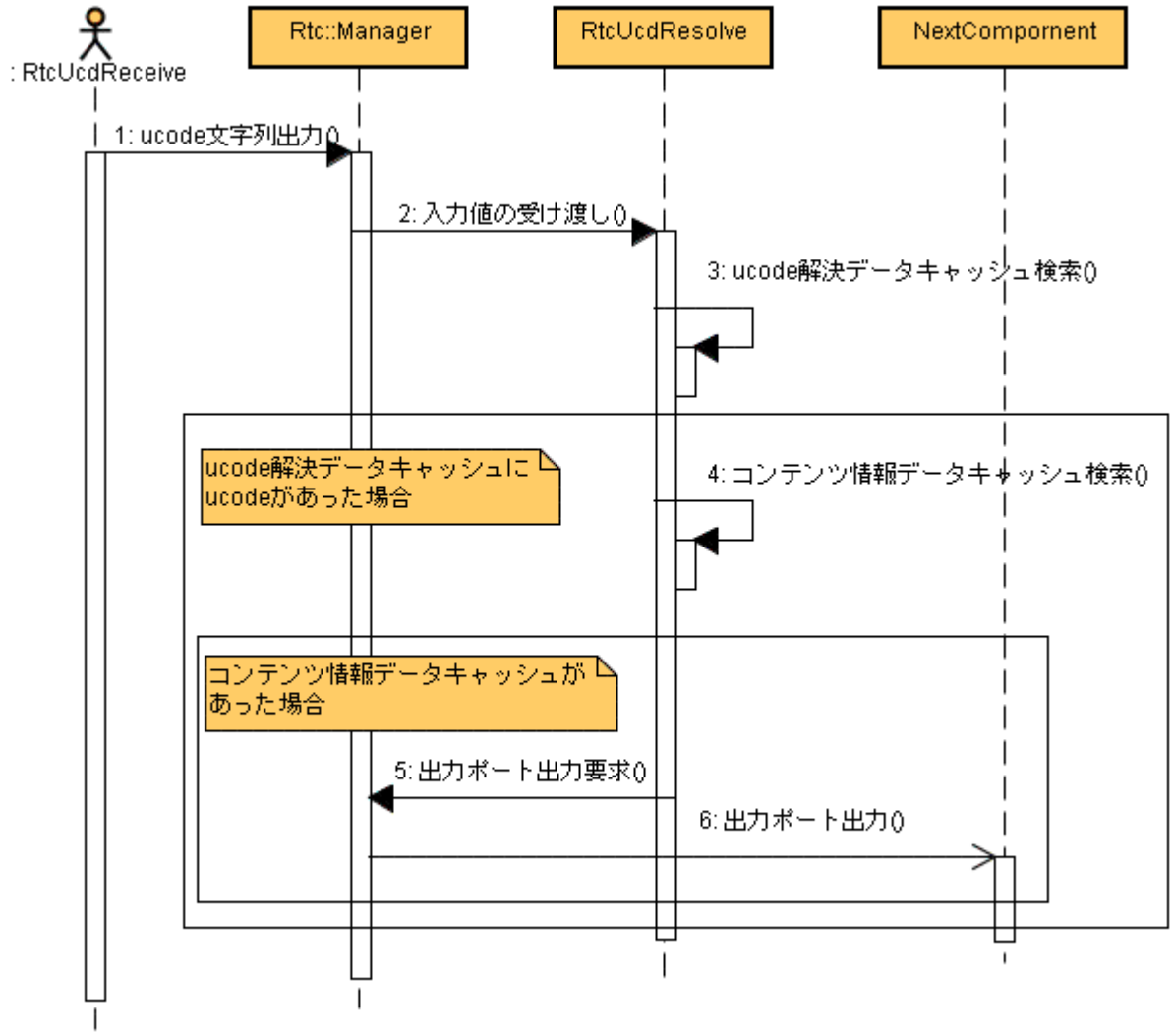


図 3-24 ucode 解決コンポーネントの動作シーケンス図

ucode 解決コンポーネントの動作シーケンス図の説明を以下に示す。

（1） RtcUcdReceive（赤外線リーダコンポーネント)の出力ポートから ucode 解決コンポーネントへ ucode 文字列が出力される。
（2） RTC::Maneger が入力ポートの入力値を RtcUcdResolve へ受け渡す。
（3） コンテンツデータキャッシュファイルパス取得の為、ucode 解決データキャッシュから検索を行う。
（4） コンテンツデータ取得の為、コンテンツデータキャッシュを検索を行う。
（5） 正常なコンテンツを取得した時、RtcUcdResolve がバインドされた変数にコンテンツ文字列を格納し、RTC::Maneger へ出力ポート出力要求を行う。
（6） RTC::Maneger が出力ポートデータの出力を行う。

3.7.6. コンテンツ解釈コンポーネントの起動シーケンス

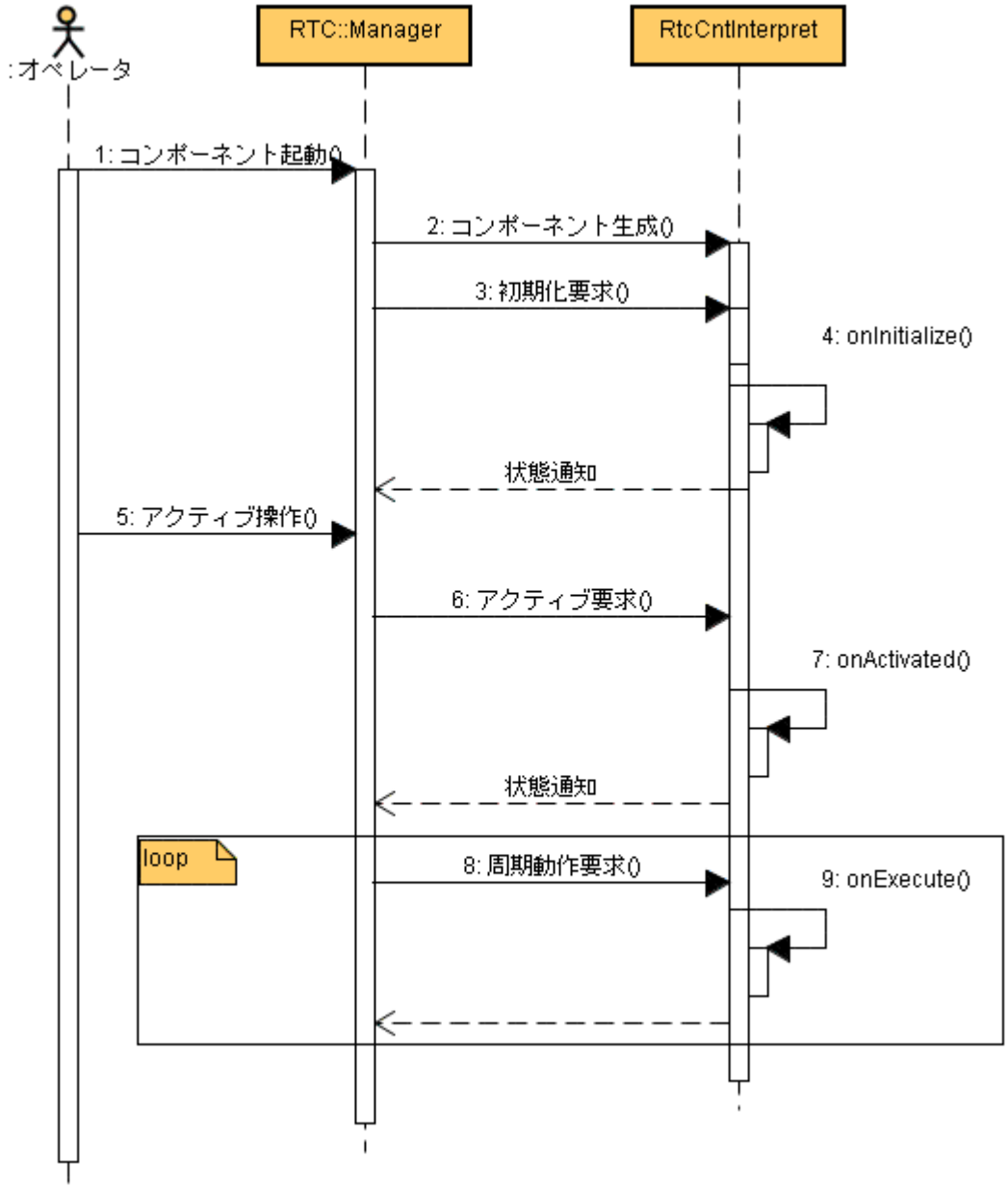


図 3-25 コンテンツ解釈コンポーネントの起動シーケンス図

コンテンツ解釈コンポーネントの起動シーケンス図の説明を以下に示す

（1） オペレータからの起動コマンド
（2） （1）により、RTC::Maneger がコンテンツ解釈コンポーントの生成を行う。
（3） RTC::Maneger から RtcCntInterpret へ初期化要求を行う。
（4） RtcCntInterpret で初期化処理が行われ、状態を RTC::Maneger へ返す。
（5） オペレータからのアクティブ操作
（6） （5）により RTC::Maneger が RtcCntInterpret へアクティブ要求を行う。
（7） RtcCntInterpret でアクティブ処理が行われ、コンポーネントがアクティブな状態となる。状態を RTC::Maneger へ返す。
（8） コンポーネントがアクティブな状態かつ正常な状態である時、RTC::Maneger は RtcCntInterpret へ周期動作を要求し続ける。
（9） RtcCntInterpret の周期処理が行われる。状態を RTC::Maneger へ返す。

3.7.7. コンテンツ解釈コンポーネントの動作シーケンス

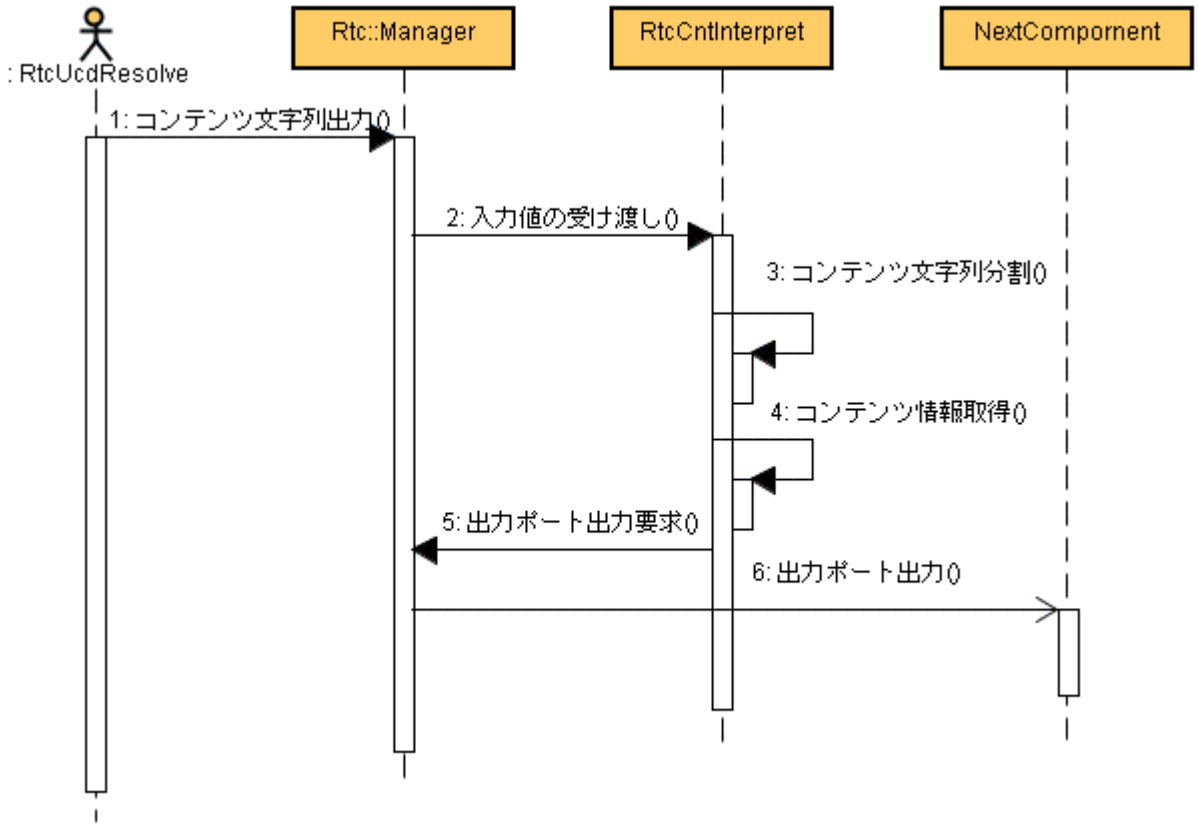


図 3-26 コンテンツ解釈コンポーネントの動作シーケンス図

コンテンツ解釈コンポーネントの動作シーケンス図の説明を以下に示す。

（1） RtcUcdResolve（ucode 解決コンポーネント)の出力ポートよりコンテンツ解釈コンポーネントへコンテンツ文字列が出力される。
（2） RTC::Maneger が入力ポートの入力値を RtcCntInterpret へ受け渡す。
（3） RtcCntInterpret が、取得したコンテンツ文字列を出力データ型に合わせて分割する。
（4） RtcCntInterpret が、出力ポートにバインドされた変数にそれぞれの値を格納する。
（5） RtcCntInterpret が RTC::Maneger へ出力ポートの出力要求を行う。
（6） RTC::Maneger が出力ポートデータの出力を行う。

4. 実行環境の構築手順（Windows 版）

4.1. UC のモジュール実行環境構築

4.1.1. Tera Term Pro のインストール

UC 評価キットに含まれる **T-Engine/SH7727 開発キット GNU 開発環境（Windows 版）説明書**の **1.2 Tera Term Pro のインストール**を参考にインストールを行ってください。

4.1.2. ディスクの作成

1. PC と UC をシリアルケーブルで接続してください。
2. Tera Term Pro を起動して、「Serial」を選択して「port」に PC と UC が接続されているポートを指定してください。
3. UC 評価キットに含まれる **UC 評価キット取扱説明書**の **2. 1 システムディスク作成方法**を参考にシステムディスクを作成してください。

4.1.3. 赤外線受信モジュールの転送

1. PC と UC をシリアルケーブルで接続してください。
2. Tera Term Pro を起動して、「Serial」を選択して「port」に PC と UC が接続されているポートを指定してください。
3. UC の電源を入れてください。
4. Tera Term Pro 上で
   [/SYS]% recv –cd IrReceiveLook
   を実行してください。
5. Tera Term Pro 上で
   Target: IrReceiveLook
   SIrReceiveLook
   で受信待ちに入ったら、Tera Term Pro の「File」「Transfer」「XMODEM」「Send」で送信するファイル「IrReceiveLook」を指定して転送してください。

4.1.4. 赤外線受信モジュール設定ファイルの転送

1. IrReceive.ini を環境に合わせて変更してください。

   下記のシリアルポート設定の指定は必須です。必ず、PC と UC が接続されているポートを指定してください。

```
* ログファイル
* ログファイルディレクトリパス
LOG_FILE_PATH=/SYS/log/
* ログファイル名
LOG_FINE_NAME=IrReceive
* ログファイルモード(1:番号 2:日単位)
LOG_FILE_MODE=1
* ログファイル作成最大数(1～7)
LOG_FILE_MAX_NUM=3
* ログファイルカウンタ
LOG_FILE_COUNTER=1
* ログファイル出力(0:OFF 1:ON)
LOG_FILE_OUTPUT=1

* コンソール出力(0:OFF 1:ON)
CONSOLE_OUTPUT=1

* シリアルポート設定
* ポート
PORT=rsa
* ボーレート
BAUD_RATE=115200
* データ(0:7bit 1:8bit)
DATA=1
* パリティ(0:なし 1:奇数 2:偶数)
PARITY=0
* ストップ(0:1bit 1:2bit)
STOP=0
```

図 4-1 赤外線受信アプリケーション設定ファイル例

2. PC と UC をシリアルケーブルで接続してください。
3. Tera Term Pro を起動して、「Serial」を選択して「port」に PC と UC が接続されているポートを指定してください。
4. UC の電源を入れてください。
5. Tera Term Pro 上で

   [/SYS]% recv –cd IrReceive.ini

   を実行してください。
6. Tera Term Pro 上で

   Target: IrReceive.ini

   SIrReceive.ini

   で受信待ちに入ったら、Tera Term Pro の「File」「Transfer」「XMODEM」「Send」で送信するファイル「IrReceive.ini」を指定して転送してください。

4.1.5. 自動起動の設定

UC 電源投入時に赤外線受信モジュールを自動起動させるため、以下のように STARTUP.CLI を修正してください。

```
*
*           @(#)STARTUP.CLI (UC-SH7727)
*
*           システム起動用 CLI 初期コマンドスクリプト
*           (C) Copyright 2003-2005 by Personal Media Corporation
*
/SYS/.xcli
*
* alias do /SYS/$$PROGRAM.BOX/DLED /SYS/USR 0
*
IrReceiveLook -rf1 -m1 -e0 -a1 R
*
if &DBG == 0
            do
            exit
else
            do &
endif
```

図 4-2 自動起動 STARTUP.CLI 修正例

1. PC と UC をシリアルケーブルで接続してください。
2. Tera Term Pro を起動して、「Serial」を選択して「port」に PC と UC が接続されているポートを指定してください。
3. UC の電源を入れてください。
4. Tera Term Pro 上で
   [/SYS]% recv –cd STARTUP.CLI
   を実行してください。
5. Tera Term Pro 上で
   Target: STARTUP.CLI
   SSTARTUP.CLI
   で受信待ちに入ったら、Tera Term Pro の「File」「Transfer」「XMODEM」「Send」で送信するファイル「STARTUP.CLI」を指定して転送してください。

4.1.6. 非デバッグモードの設定

UC のシステムを非デバッグモードで動作させるため、以下のように DEVCONF を修正してください。

```
#
#           @(#)DEVCONF (UC-SH7727)
#
#           デバイス構成定義
#           Copyright (C) 2003-2005 by Personal Media Corporation
#

# デバッグモード
DEBUGMODE 0

# PCCARD マネージャ (0:不使用 1:使用)
CardMgrEnable     0
```

図 4-3 非デバッグモード DEVCONF 修正例

1. PC と UC をシリアルケーブルで接続してください。
2. Tera Term Pro を起動して、「Serial」を選択して「port」に PC と UC が接続されているポートを指定してください。
3. UC の電源を入れてください。
4. Tera Term Pro 上で

   [/SYS]% recv –cd DEVCONF

   を実行してください。
5. Tera Term Pro 上で

   Target: DEVCONF

   SDEVCONF

   で受信待ちに入ったら、Tera Term Pro の「File」「Transfer」「XMODEM」「Send」で送信するファイル「DEVCONF」を指定して転送してください。

(非デバッグモードを元に戻す方法)

非デバッグモードで起動すると、

- コンソールへの出力は無視され、入力はできない。
- コンソールはシリアルポートデバイス(rsa)としてのみ利用可能。

になります。

元に戻す場合は、

1. SD カードをすべて外して、システムを Flash ROM ディスクから起動します。
2. SD カードを入れて、
   - ① [/SYS]% att sda0 /A
   - ② [/SYS]% cd /A
   - ③ [/A]% recv –cd DEVCONF

   を実行して転送してください。

   ※DEVCONF の DEBUGMODE を 1 に修正から転送してください。

4.2. Windows のコンポーネント実行環境構築

4.2.1. OpenRTM-aist のインストール

1. OpenRTM-aist ダウンロード（C++版）
   ビルド済みパッケージ(Windows)の Visual Studio 2005 用の下記の一式をダウンロードしてください。

   - OpenRTM-aist-0.4.2-jp_vc8.msi
   - ACE-5.5_vc8.msi
   - omniORB_4.0.7-jp_vc8.msi
   - python-2.4.2.msi
   - PyYAML-3.0.5.win32-py2.4.exe
   - Microsoft Visual C++ 2005 SP1 再頒布可能パッケージ

   再配布可能パッケージとは VisualStudio のランタイムライブラリです。VisualStudio(VC++)がインストールされていない PC 上で RTC を動作させる場合にはインストールしてください。

2. OpenRTM-aist C++版インストール
   Windows 系システムでのインストールを参考にインストールを行ってください。

3. OpenRTM-aist C++版インストールの確認
   インストールの確認を参考に動作確認を行ってください。

4.2.2. RtcLink・RtcTemplate のインストール

1. OpenRTM-aist ダウンロード（RtcLink・RtcTemplate）
   全部入りパッケージの Windows 用全部入りをダウンロードしてください。

   - eclipse32_rtclink041_rtctemplate042_win32.zip

2. RtcLink・RtcTemplate のインストール
   Java 実行環境（JRE）のインストールを参考に Java 実行環境をインストールしてください。
   インストールの仕方を参考に Eclipse、RtcLink、RtcTemplate をインストールしてください。

4.2.3. 環境自己位置同定コンポーネントのインストール

1. 任意のディレクトリ（ここでは C:¥Workspace¥とします）に
   - selfposition.tgz

   をコピーしてください。

2. 1. でコピーした selfposition.tgz を解凍してください。

3. 2.で解凍して出来た selfposition フォルダ以下の bin¥win¥UcdReceive.ini を環境に合わせて変更してください。
   下記のシリアルポートの指定は必須です。必ず、PC と UC が接続されているポートを指定してください。

```
* ログファイル
* ログファイルディレクトリパス
LOG_FILE_PATH=./log/
* ログファイル名
LOG_FILE_NAME=UcdReceive
* ログファイルモード(1:番号 2:日単位)
LOG_FILE_MODE=2
* ログファイル作成最大数(1～7)
LOG_FILE_MAX_NUM=3
* ログファイルカウンタ
LOG_FILE_COUNTER=1
* ログファイル出力(0:OFF 1:ON)
LOG_FILE_OUTPUT=1

* コンソール出力(0:OFF 1:ON)
CONSOLE_OUTPUT=1

* シリアルポート
PORT=rsa
* ボーレート
BAUD_RATE=115200
* データ(0:7bit 1:8bit)
DATA=1
* パリティ(0:なし 1:奇数 2:偶数)
PARITY=0
* ストップ(0:1bit 1:2bit)
STOP=0
* シリアル通信タイムアウト時間
READ_TIMEOUT=5000

* 同じ ucode を受信した場合の time 設定(秒)
OUT_CYCLE=10
```

図 4-4 ucode リーダコンポーネント設定ファイル例

4. 2.で解凍して出来た selfposition フォルダ以下の bin¥win¥UcdResolve.txt を環境に合わせて変更してください。
UcdResolve.txt の詳細は2.5.5を参照のこと

UcdResolve.txt は、
UCODE,UCODE に対応するコンテンツキャッシュファイルパス
の形式で記載してください。

```
00000000000000000000000000101E3,./Contents/00000000000000000000000000101E3
00000000000000000000000000010164,./Contents/00000000000000000000000000010164
```

図 4-5 UcdResolve.txt 例

5. 4. で指定したコンテンツキャッシュファイルパスにコンテンツキャッシュファイルを修正する。
（上記、図 4-5のデータ例の場合は
C:¥Workspace¥selfposition¥bin¥win¥Contents¥00000000000000000000000000101E3
C:¥Workspace¥selfposition¥bin¥win¥Contents¥00000000000000000000000000010164
を作成してください。）

作成したコンテンツキャッシュファイルを環境に合わせて変更してください。
コンテンツキャッシュファイルの詳細は2.5.6を参照のこと

コンテンツキャッシュファイルは
緯度, 経度, 海抜, 誤差
の形式で記載してください。

```
141.121212,43.154554,512,0
```

図 4-6 ./Contents/00000000000000000000000000101E3 例

5. コンポーネントの起動手順（Windows 版）

5.1. コンポーネントの起動手順

1. CORBA ネームサーバの起動
   プログラムメニューの「OpenRTM-aist」→「example」→「NameService.bat」を実行
   あるいは、OpenRTM-aist インストールディレクトリ/bin にある rtm-naming.bat をダブルクリックして起動します。

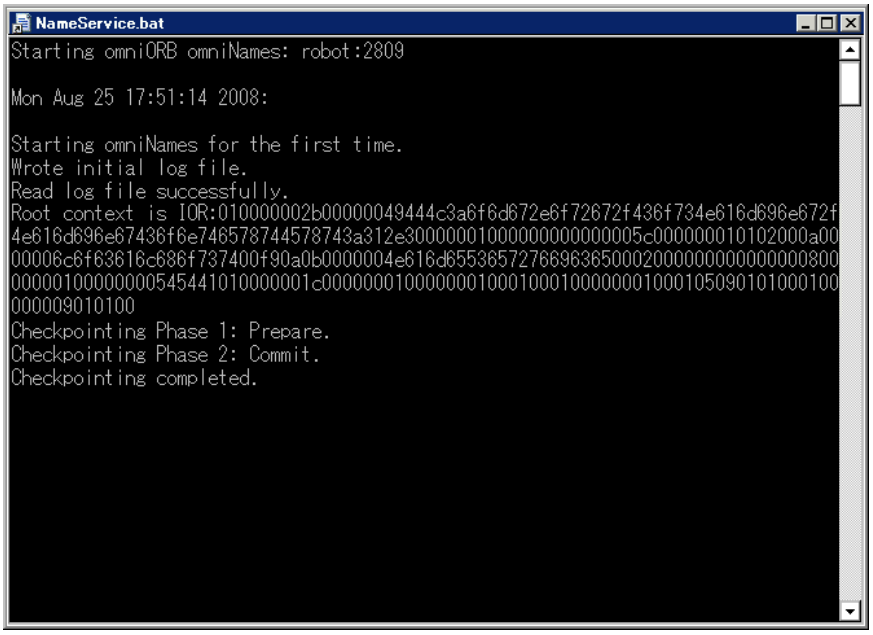


図 5-1 CORBA ネームサーバの起動

2. UCODE 受信コンポーネントの起動
   4.2.3で selfposition.tgz を解凍して出来た selfposition フォルダの bin¥win 以下にある「RtcUcdReceive.exe」をダブルクリックして起動します。コンソールが表示されるのを確認してください。

3. UCODE 解決コンポーネントの起動
   4.2.3で selfposition.tgz を解凍して出来た selfposition フォルダの bin¥win 以下にある「RtcUcdResolve.exe」をダブルクリックして起動します。コンソールが表示されるのを確認してください。

4. コンテンツ解釈コンポーネントの起動
   4.2.3で selfposition.tgz を解凍して出来た selfposition フォルダの bin¥win 以下にある「RtcCntInterpret.exe」をダブルクリックします。コンソールが表示されるのを確認してください。

5. テストコンポーネントの起動

   4.2.3で selfposition.tgz を解凍して出来たフォルダの bin¥win 以下にある「TestSelfLocalization.exe」をダブルクリックします。コンソールが表示されるのを確認してください。

6. Eclipse の起動

   4.2.2でインストールした Eclipse のインストールフォルダから「eclipse.exe」を探し、ダブルクリックします。
   図 5-2のような画面が表示されますのでワークスペースに任意のディレクトリ(ここでは C:¥Workspace¥selfposition¥とします。)を指定してください。

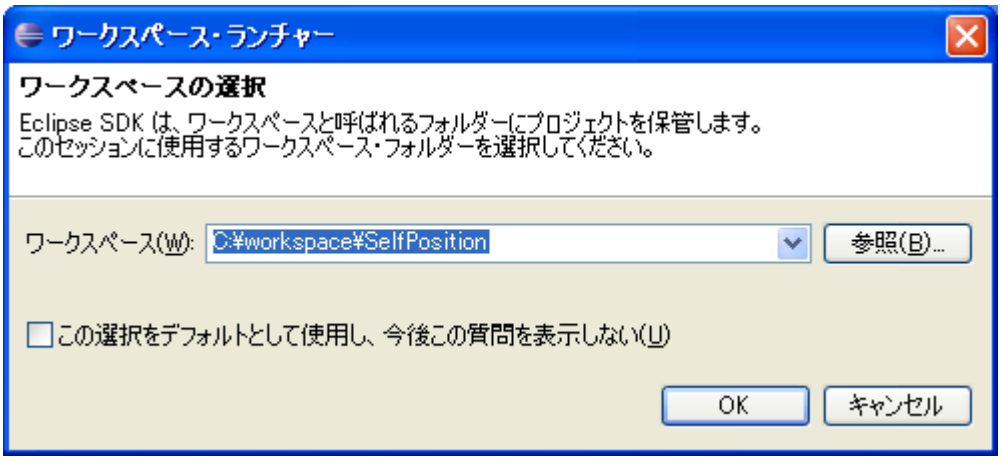


図 5-2 ワークスペース・ランチャー

7. RtcLink の起動

Eclipse のメニューから「ウィンドウ(W)」→「パースペクティブを開く(O)」→「その他(O)」を選択すると、図 5-3のような画面が表示されますので「RTCLink」を選択すると RtcLink が起動されます。

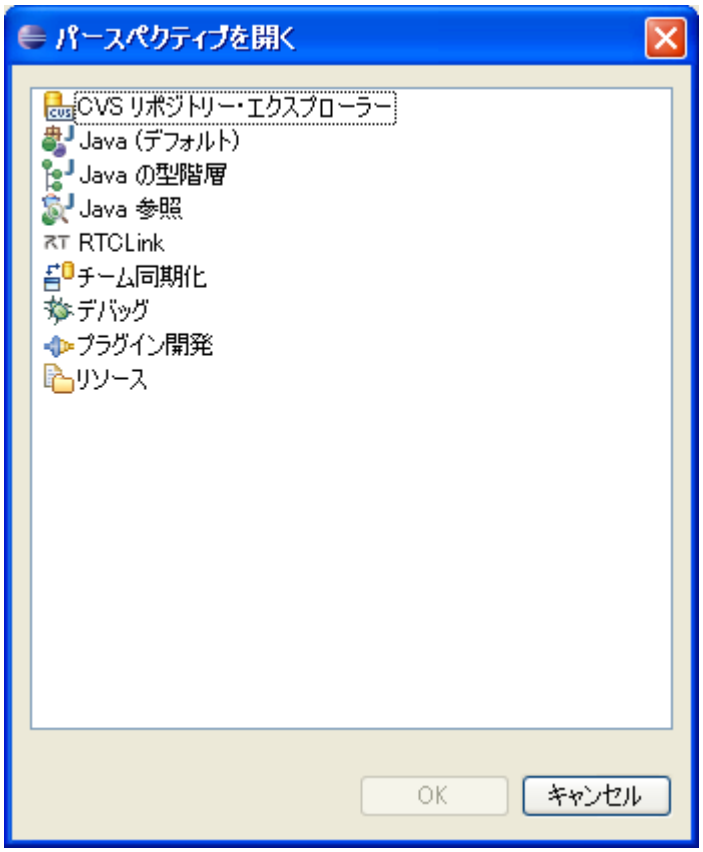


図 5-3 パースペクティブ選択画面

8. ネームサーバへの接続

ネームサーバビュー（図 5-4の赤線で囲まれた箇所）の上部に存在するボタン（図 5-4の青線で囲まれた箇所）を押下してください。図 5-5のような画面が表示されますので「localhost」を入力して「OK」ボタンを押下げしてください。

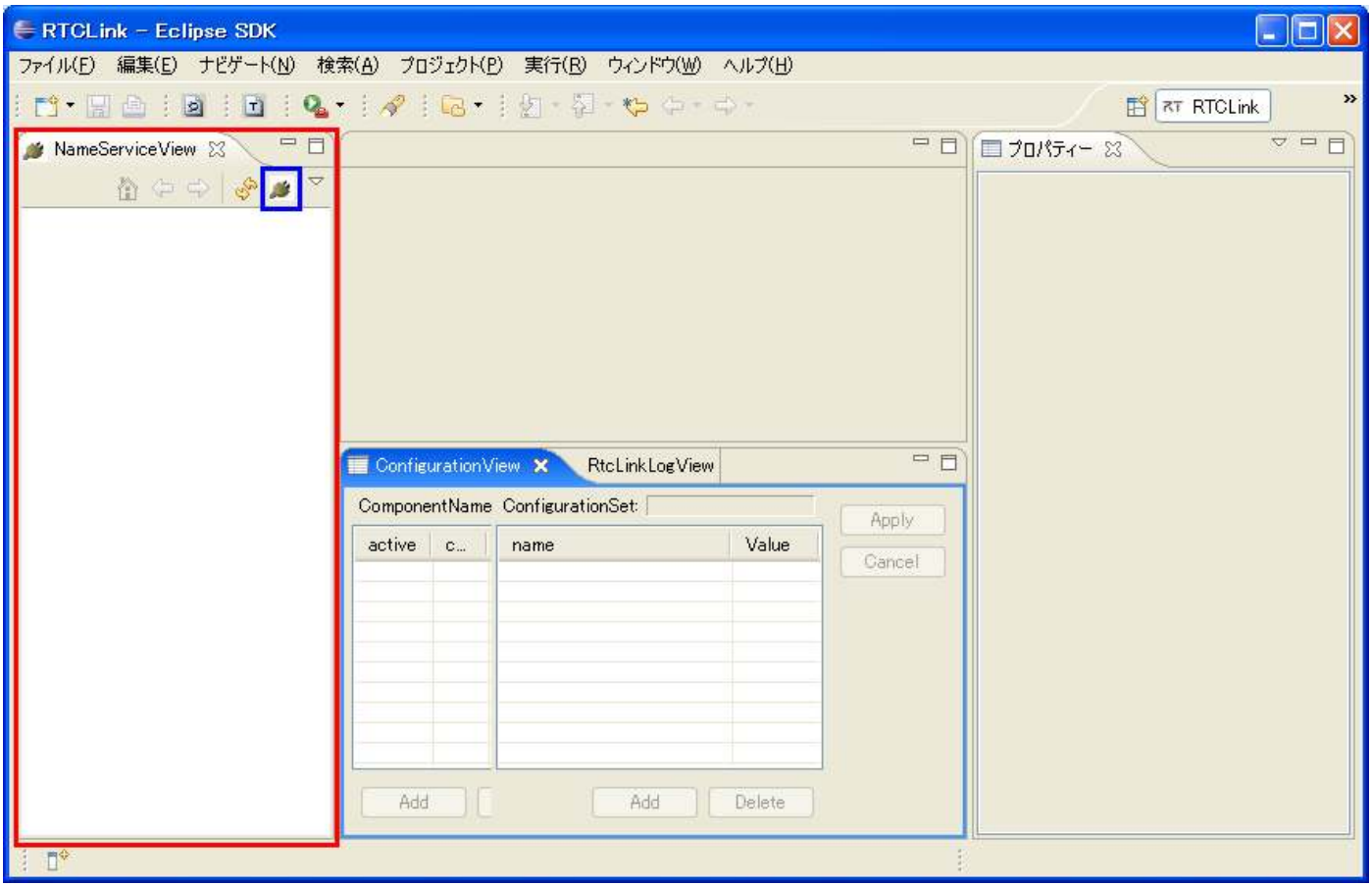


図 5-4 RtcLink 起動後の画面

図 5-5 Connect Name Server

9. ネームサーバに登録されているコンポーネントの確認

図 5-6のように、

- RtcCntInterpret
- RtcUcdReceive
- RtcUcdResolve
- TestSelfLocalization

が登録されていることを確認してください。

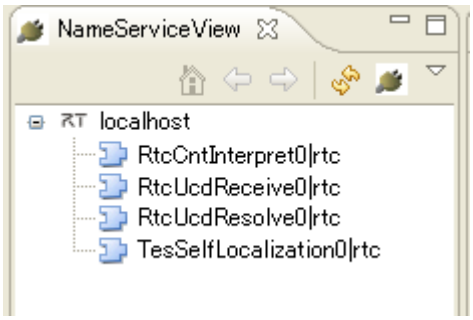


図 5-6 ネームサービスビュー例

10. システムエディタを開く

ツールバーの「Open New System Editor」ボタン（図 5-7の青色で囲まれた箇所）を押下げしてください。

システムエディタ（図 5-7の赤色で囲まれた箇所）が開かれるのを確認してください。

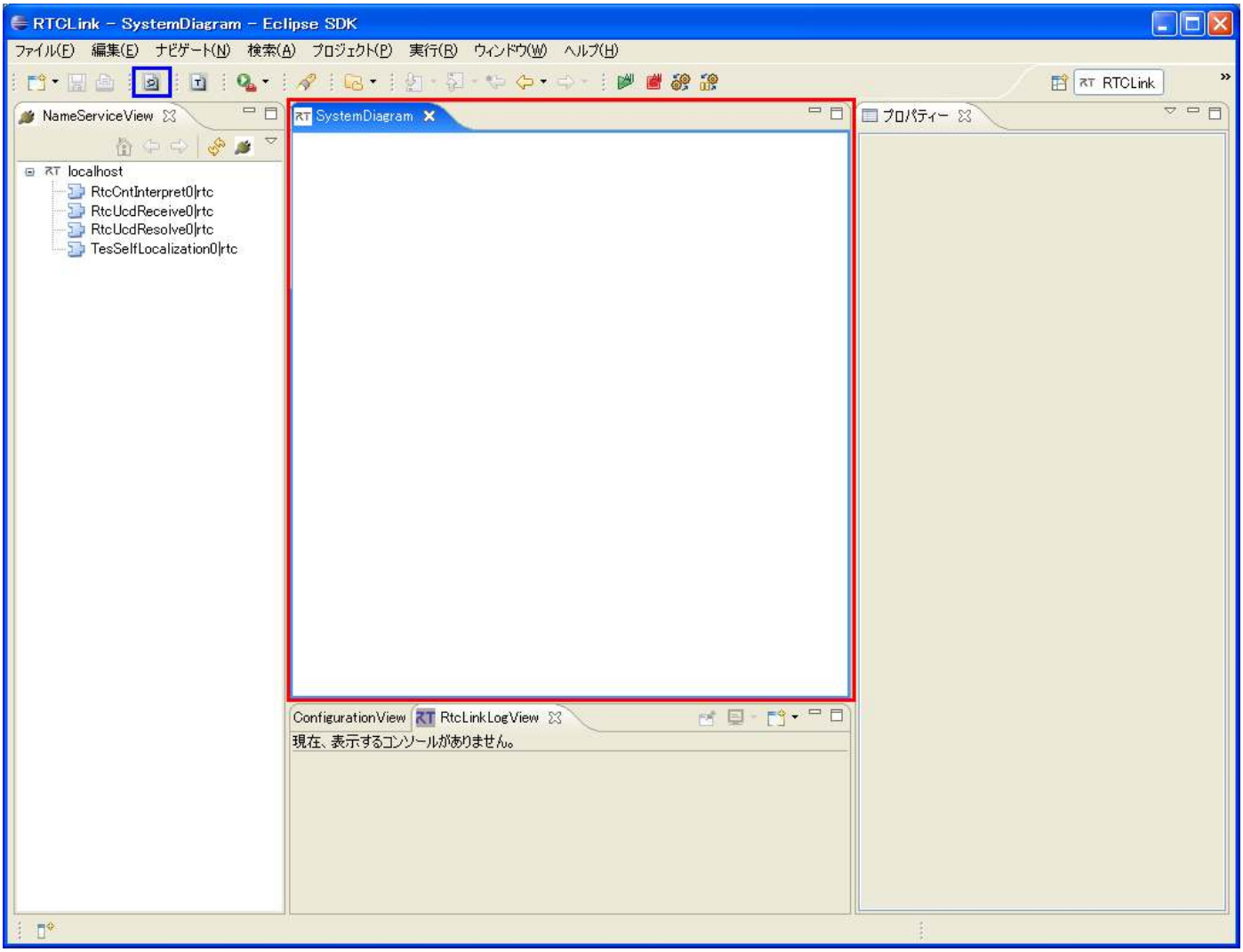


図 5-7 新規システムエディタを開いた後の画面

11. コンポーネントをシステムエディタに配置

システムエディタにネームサーバビューから RT コンポーネントをドラッグ＆ドロップしてください。（図 5-8）
図 5-9のようにシステムエディタにコンポーネントが表示されるのを確認してください。

以下の全てのコンポーネントをシステムエディタに配置してください。

- RtcCntInterpret
- RtcUcdReceive
- RtcUcdResolve
- TestSelfLocalization

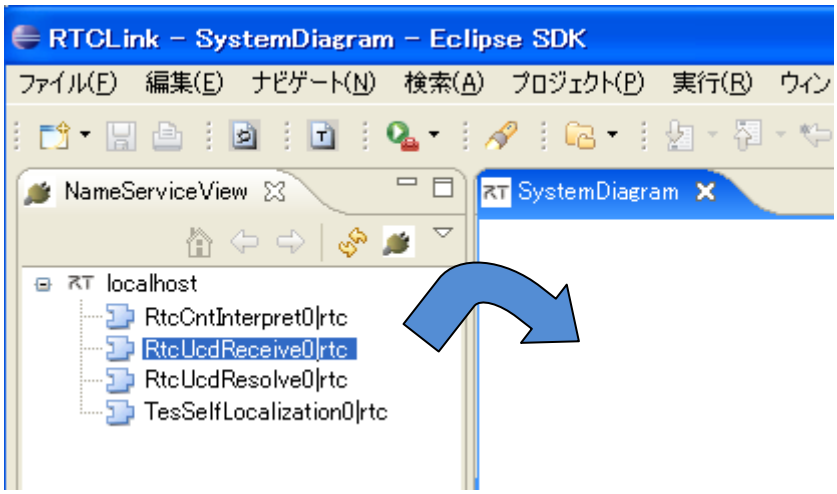


図 5-8 コンポーネントをシステムエディタに配置

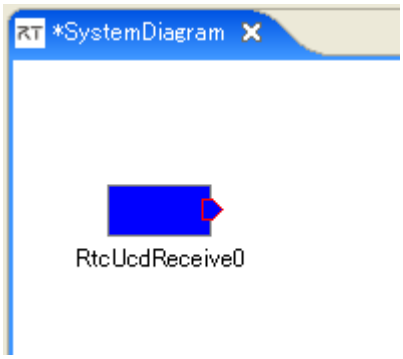


図 5-9 コンポーネント表示の例

12. コンポーネントのポート間を接続

ポートとポート（図 5-10の赤色で囲まれた箇所）をドラッグ＆ドラッグしてください。

ドラッグ＆ドロップ実行後、図 5-11の画面が表示されますので「OK」ボタンを押下げしてください。

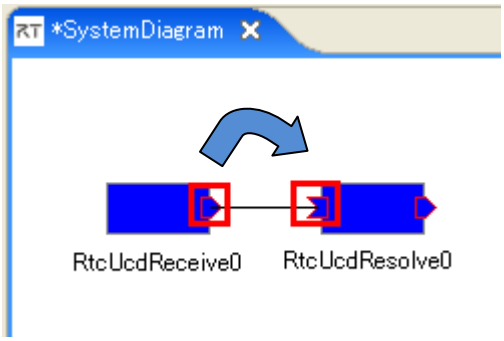


図 5-10 ポート間の接続

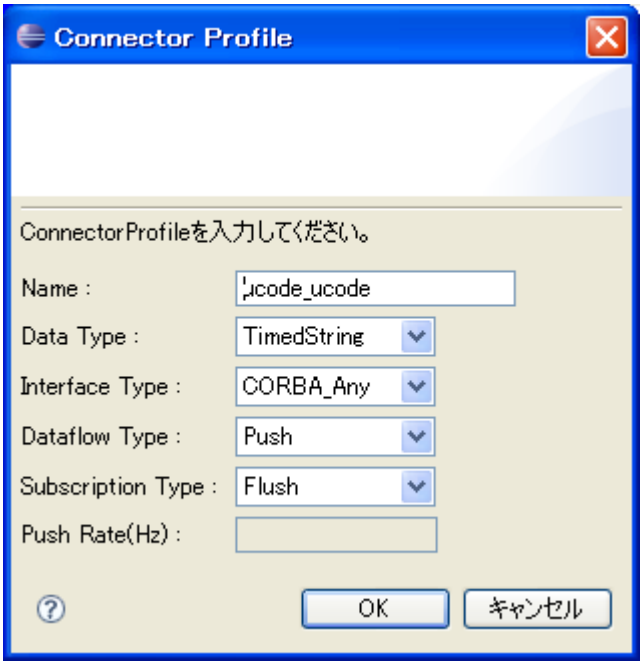


図 5-11 Connector Profile

13. 全コンポーネントのポート間接続

12.を参考に図 5-12のようになるようにポート間を接続してください。

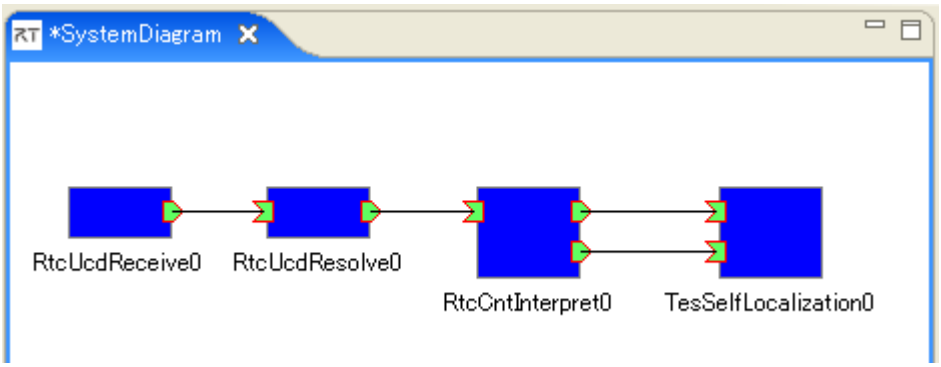


図 5-12 全コンポーネントのポート間接続例

14. コンポーネントの状態変更

システムエディタ上で右クリックして図 5-13の「All Active」を選択してください。

図 5-14のようになるのを確認してください。

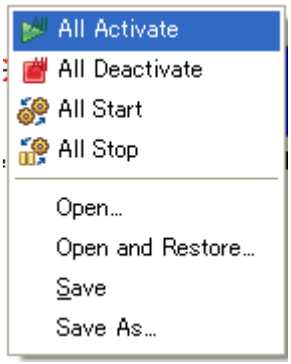


図 5-13 コンテキストメニュー

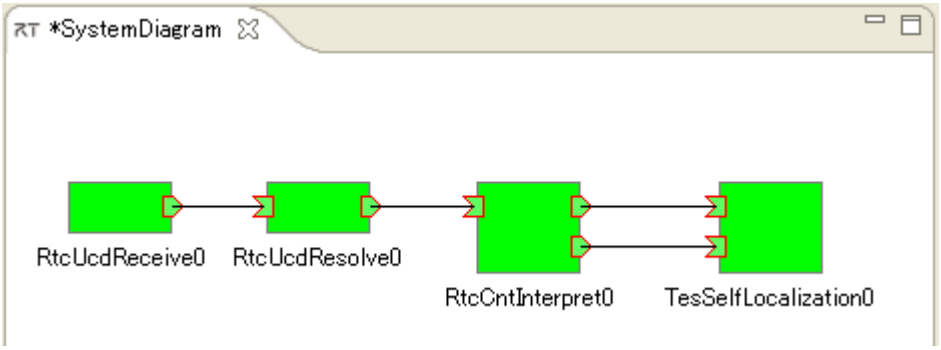


図 5-14 全コンポーネントの Active 化例

15. UC の起動

UC の電源を入れてください。

16. 赤外線マーカの設定

赤外線マーカ（図 5-15）の側面につまみ（図 5-16）で ucode の下二桁を変更してください。

（4.2.3の手順 4. で変更した UcdResolve.txt に記載のある ucode になるように変更してください。）

図 5-16ですと、赤色で囲まれたつまみが E、青色で囲まれたつまみが 3 なので下二桁は E3 になります。

ucode は 00000000000000000000000000000101E3 となります。

図 5-15 赤外線マーカ

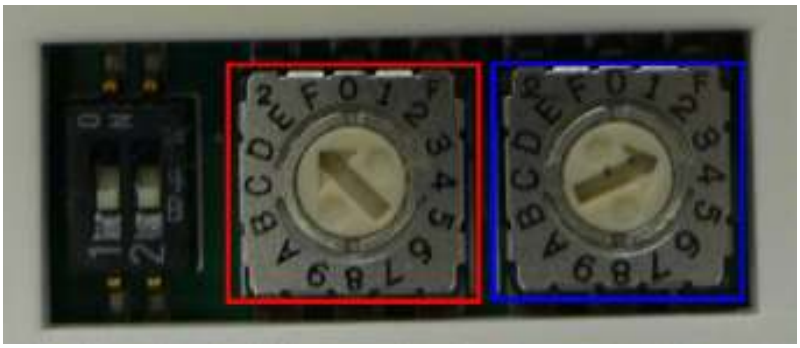


図 5-16 赤外線マーカ側面のつまみの例

17. 赤外線マーカの起動

赤外線マーカ（図 5-15）の電源を入れてください。

黄色の LED が点滅するのを確認してください。

18. 動作結果確認

UC に赤外線マーカを近付けてください。

TesSelfLocalization.exe のコンソールに緯度、経度、海抜、誤差が表示されるのを確認してください。（図 5-17の赤線で囲まれた箇所）

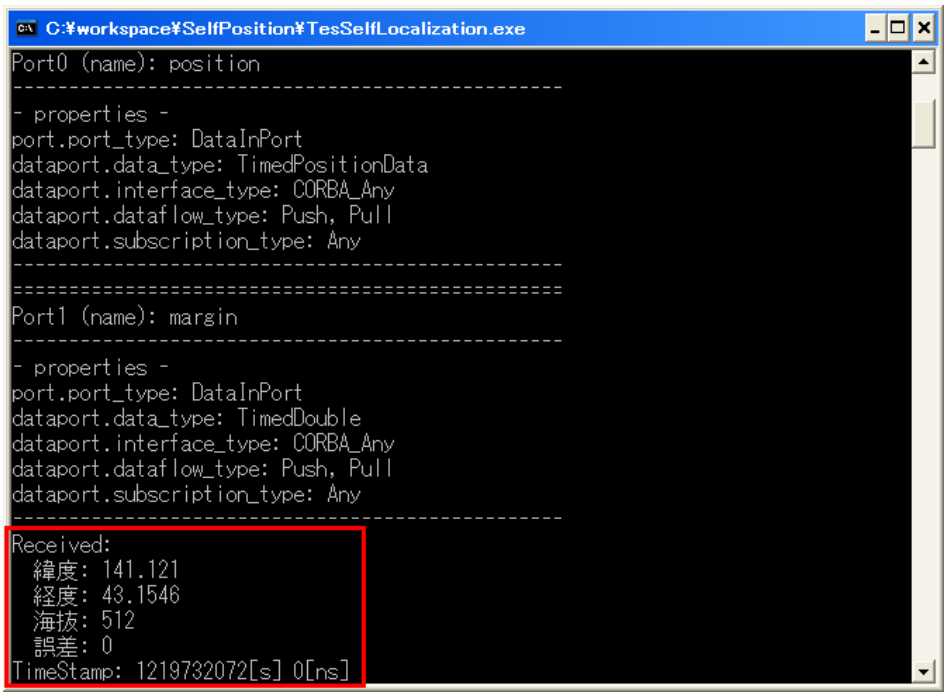


図 5-17 TesSelfLocalization.exe のコンソール

6. 開発環境の構築手順（Windows 版）

6.1. UC のモジュール開発環境構築

6.1.1. GNU 開発環境（Windows 版）のインストール

UC 評価キットに含まれる **T-Engine/SH7727 開発キット GNU 開発環境（Windows 版）説明書**の**第 1 章 インストール**を参考にインストールを行ってください。

6.1.2. 環境自己位置同定モジュールのインストール

1. 6.1.1でインストールした開発環境のベースディレクトリ以下に
   - selfposition.tgz

   をコピーして解凍してください。

6.1.3. LibRtcIniFile ライブラリのビルド

1. Cygwin を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcIniFile.tgz
$ cd LibRtcIniFile/build/sh7727
$ make install
$ ls
```

6.1.4. LibRtcLog ライブラリのビルド

1. Cygwin を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcLog.tgz
$ cd LibRtcLog/build/sh7727
$ make install
$ ls
```

6.1.5. LibRtcRs232c ライブラリのビルド

1. Cygwin を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcRs232c.tgz
$ cd LibRtcRs232c/build/sh7727
$ make install
$ ls
```

6.1.6. LibRtcSemaphore ライブラリのビルド

1. Cygwin を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcSemaphore.tgz
$ cd LibRtcSemaphore/build/sh7727
$ make install
$ ls
```

6.1.7. LibRtcStr ライブラリのビルド

1. Cygwin を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcStr.tgz
$ cd LibRtcStr/build/sh7727
$ make install
$ ls
```

6.1.8. 赤外線受信モジュールのビルド

1. Cygterm を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/src
$ tar xvzf IrReceiveLook.tgz
$ cd IrReceiveLook/sh7727/
$ make
$ ls
```

6.2. Windows のコンポーネントの開発環境構築

6.2.1. OpenRTM-aist のインストール

1. OpenRTM-aist ダウンロード(C++版)
   ビルド済みパッケージ(Windows)の Visual Studio 2005 用の下記の一式をダウンロードしてください。

   - OpenRTM-aist-0.4.2-jp_vc8.msi
   - ACE-5.5_vc8.msi
   - omniORB_4.0.7-jp_vc8.msi
   - python-2.4.2.msi
   - PyYAML-3.0.5.win32-py2.4.exe

2. OpenRTM-aist C++版インストール
   Windows 系システムのビルドを参考に下記をインストールしてください。

   - 「Visual Studio 2005 Professional」
     あるいは
     「Visual C++ 2005 Express Edition」&「Microsoft PlatformSDK」
   - OpenRTM-aist-0.4.2-jp_vc8.msi
   - ACE-5.5_vc8.msi
   - omniORB_4.0.7-jp_vc8.msi
   - python-2.4.2.msi
   - PyYAML-3.0.5.win32-py2.4.exe

3. OpenRTM-aist C++版インストールの確認
   インストールの確認を参考に動作確認を行ってください。

6.2.2. 環境自己位置同定モジュールのインストール

1. 任意のディレクトリ(ここでは C:¥Workspace¥とします)に
   - selfposition.tgz

   をコピーして解凍してください。

6.2.3. 共通ライブラリのビルド

1. 6.2.2で解凍して出来た selfposition フォルダの lib 以下にある
   - LibRtcIniFile.tgz
   - LibRtcLog.tgz
   - LibRtcRs232c.tgz
   - LibRtcSemaphore.tgz
   - LibRtcStr.tgz
   - LibSqlite3.tgz

   を解凍してください。

2. 1. で解凍して出来たフォルダの build¥win 以下にある VC++ Project ファイル（拡張子が.vcproj のファイル）をダブルクリックし、「Visual Studio 2005 Proffesional」あるいは「Visual C++ 2005」で起動してください。

3. 「Visual Studio 2005 Proffesional」あるいは「Visual C++ 2005」のメニューから「ビルド」→「ソリューションのビルド」を選択してコンポーネントのビルドをしてください。
   ※ビルドして出来たライブラリファイル（拡張子が.lib のファイル）は selfposition¥lib¥win 以下にコピーしてください。

6.2.4. 環境自己位置同定コンポーネントのビルド

1. 6.2.2で解凍して出来たフォルダの src 以下にある
   - RtcUcdReceive.tgz
   - RtcUcdResolve.tgz
   - RtcCntInterpret.tgz
   - TestSelfLocalization.tgz

   を解凍してください。

2. 解凍して出来たフォルダの build/win 以下にある VC++ Project ファイル（拡張子が.vcproj のファイル）をダブルクリックし、「Visual Studio 2005 Proffesional」あるいは「Visual C++ 2005」で起動してください。

3. 「Visual Studio 2005 Proffesional」あるいは「Visual C++ 2005」のメニューから「ビルド」→「ソリューションのビルド」を選択してコンポーネントのビルドをしてください。

7. 実行環境の構築手順（UNIX 版）

7.1. UC のモジュール実行環境構築

7.1.1. GNU 開発環境のインストール

UC 評価キットに含まれる**T-Engine/SH7727 開発キット GNU 開発環境説明書**の**2. 開発環境のインストール**を参考にインストールを行ってください。

7.1.2. ディスクの作成

1. PC と UC をシリアルケーブルで接続してください。
2. 端末を起動して、
   $ /usr/local/te/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0
   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。
   を実行してください。
3. UC 評価キットに含まれる**UC 評価キット取扱説明書**の**2. 1 システムディスク作成方法**を参考にシステムディスクを作成してください。

7.1.3. 赤外線受信モジュールの転送

1. PC と UC をシリアルケーブルで接続してください。
2. 端末を起動して IrReceiveLook があるディレクトリまで移動してください。
3. 端末上で
   $ /usr/local/te/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0
   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。
   を実行してください。
4. UC の電源を入れてください。
5. 端末上で
   [/SYS]% recv –cd IrReceiveLook
   を実行してください。

7.1.4. 赤外線受信モジュール設定ファイルの転送

1. IrReceive.ini を環境に合わせて変更してください。

   下記のシリアルポート設定の指定は必須です。必ず、PC と UC が接続されているポートを指定してください。

```
* ログファイル
* ログファイルディレクトリパス
LOG_FILE_PATH=/SYS/log/
* ログファイル名
LOG_FINE_NAME=IrReceive
* ログファイルモード(1:番号 2:日単位)
LOG_FILE_MODE=1
* ログファイル作成最大数(1～7)
LOG_FILE_MAX_NUM=3
* ログファイルカウンタ
LOG_FILE_COUNTER=1
* ログファイル出力(0:OFF 1:ON)
LOG_FILE_OUTPUT=1

* コンソール出力(0:OFF 1:ON)
CONSOLE_OUTPUT=1

* シリアルポート設定
* ポート
PORT=rsa
* ボーレート
BAUD_RATE=115200
* データ(0:7bit 1:8bit)
DATA=3
* パリティ(0:なし 1:奇数 2:偶数)
PARITY=0
* ストップ(0:1bit 1:2bit)
STOP=0
```

図 7-1 赤外線受信アプリケーション設定ファイル例

2. PC と UC をシリアルケーブルで接続してください。
3. 端末を起動して IrReceive.ini があるディレクトリまで移動してください。
4. 端末上で

   $ /usr/local/te/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0

   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。

   を実行してください。
5. UC の電源を入れてください。
6. 端末上で

   [/SYS]% recv –cd IrReceive.ini

   を実行してください。

7.1.5. 自動起動の設定

UC 電源投入時に赤外線受信モジュールを自動起動させるため、以下のように STARTUP.CLI を修正してください。

```
*
*           @(#)STARTUP.CLI (UC-SH7727)
*
*           システム起動用 CLI 初期コマンドスクリプト
*           (C) Copyright 2003-2005 by Personal Media Corporation
*
/SYS/.xcli
*
* alias do /SYS/$$PROGRAM.BOX/DLED /SYS/USR 0
*
IrReceiveLook -rf1 -m1 -e0 -a1 R
*
if &DBG == 0
            do
            exit
else
            do &
endif
```

図 7-2 自動起動 STARTUP.CLI 修正例

1. PC と UC をシリアルケーブルで接続してください。
2. 端末を起動して修正した STARTUP.CLI があるディレクトリまで移動してください。
3. 端末上で

   $ /usr/local/te/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0

   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。

   を実行してください。
4. UC の電源を入れてください。
5. 端末上で

   [/SYS]% recv –cd STARTUP.CLI

   を実行してください。

7.1.6. 非デバッグモードの設定

UC のシステムを非デバッグモードで動作させるため、以下のように DEVCONF を修正してください。

```
#
#           @(#)DEVCONF (UC-SH7727)
#
#           デバイス構成定義
#           Copyright (C) 2003-2005 by Personal Media Corporation
#

# デバッグモード
DEBUGMODE 0

# PCCARD マネージャ (0:不使用 1:使用)
CardMgrEnable     0
```

図 7-3 非デバッグモード DEVCONF 修正例

1. PC と UC をシリアルケーブルで接続してください。
2. 端末を起動して修正した DEVCONF があるディレクトリまで移動してください。
3. 端末上で

   $ /usr/local/te/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0

   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。

   を実行してください。
4. UC の電源を入れてください。
5. 端末上で

   [/SYS]% recv –cd DEVCONF

   を実行してください。

(非デバッグモードを元に戻す方法)

非デバッグモードで起動すると、

- コンソールへの出力は無視され、入力はできない。
- コンソールはシリアルポートデバイス(rsa)としてのみ利用可能。

になります。

元に戻す場合は、

1. SD カードをすべて外して、システムを Flash ROM ディスクから起動します。
2. SD カードを入れて、
   - ① [/SYS]% att sda0 /A
   - ② [/SYS]% cd /A
   - ③ [/A]% recv –cd DEVCONF

   を実行して転送してください。

   ※DEVCONF の DEBUGMODE を 1 に修正してから転送してください。

7.2. UNIX のコンポーネント実行環境構築

7.2.1. OpenRTM-aist のインストール

1. OpenRTM-aist C++版インストール
   C++版インストール(UNIX)を参考にインストールを行ってください。

2. OpenRTM-aist C++版インストールの確認
   サンプルを使ってテストを参考に動作確認を行ってください。

7.2.2. RtcLink・RtcTemplate のインストール

1. OpenRTM-aist ダウンロード(RtcLink・RtcTemplate)
   全部入りパッケージの **UNIX 用全部入り**をダウンロードしてください。

   - eclipse32_rtclink041_rtctemplate041_linux.zip

2. RtcLink・RtcTemplate のインストール
   Java 実行環境(JRE)のインストールを参考に Java 実行環境をインストールしてください。
   インストールの仕方を参考に Eclipse、RtcLink、RtcTemplate をインストールしてください。

7.2.3. 環境自己位置同定コンポーネントのインストール

1. 任意のディレクトリ(ここでは/home/user/とします)に
   - selfposition.tgz

   をコピーしてください。
   ※selfposition/bin/linux 以下にある上記、ファイル群は文字コード EUC-JP、改行コード LF になっています。
   環境に合わせて文字コード、改行コードを変更してください。

2. 1. でコピーした selfposition.tgz を解凍してください。

3. 2.で解凍して出来た selfposition フォルダ以下の bin¥linux¥UcdReceive.ini を環境に合わせて変更してください。
   下記のシリアルポートの指定は必須です。必ず、PC と UC が接続されているポートを指定してください。

```
* ログファイル
* ログファイルディレクトリパス
LOG_FILE_PATH=./log/
* ログファイル名
LOG_FILE_NAME=UcdReceive
* ログファイルモード(1:番号 2:日単位)
LOG_FILE_MODE=2
* ログファイル作成最大数(1～7)
LOG_FILE_MAX_NUM=3
* ログファイルカウンタ
LOG_FILE_COUNTER=1
* ログファイル出力(0:OFF 1:ON)
LOG_FILE_OUTPUT=1

* コンソール出力(0:OFF 1:ON)
CONSOLE_OUTPUT=1

* シリアルポート
PORT=/dev/ttyS0
* ボーレート
BAUD_RATE=115200
* データ(0:7bit 1:8bit)
DATA=1
* パリティ(0:なし 1:奇数 2:偶数)
PARITY=0
* ストップ(0:1bit 1:2bit)
STOP=0
* シリアル通信タイムアウト時間
READ_TIMEOUT=5000

* 同じ ucode を受信した場合の time 設定(秒)
OUT_CYCLE=10
```

図 7-4 ucode リーダコンポーネント設定ファイル例

4. 2. で解凍して出来た selfposition フォルダ以下の bin¥linux¥UcdResolve.txt を環境に合わせて変更してください。
UcdResolve.txt の詳細は2.5.5を参照のこと

UcdResolve.txt は、
UCODE,UCODE に対応するコンテンツキャッシュファイルパス
の形式で記載してください。

```
00000000000000000000000000101E3,./Contents/00000000000000000000000000101E3
00000000000000000000000000010164,./Contents/00000000000000000000000000010164
```

図 7-5 UcdResolve.txt 例

5. 4. で指定したコンテンツキャッシュファイルパスにコンテンツキャッシュファイルを修正する。
（上記、図 7-5 UcdResolve.txt 例のデータ例の場合は
/home/user/selfposition/bin/linux/Contents/00000000000000000000000000101E3
/home/user/selfposition/bin/linux/Contents/00000000000000000000000000010164
を作成してください。）

作成したコンテンツキャッシュファイルを環境に合わせて変更してください。
コンテンツキャッシュファイルの詳細は2.5.6を参照のこと

コンテンツキャッシュファイルは
緯度, 経度, 海抜, 誤差
の形式で記載してください。

```
141.121212,43.154554,512,0
```

図 7-6 ./Contents/00000000000000000000000000101E3 例

8. コンポーネントの起動手順（UNIX 版）

8.1. コンポーネントの起動手順

1. CORBA ネームサーバの起動確認
   端末を起動して、
   $rtm-naming 2809
   を実行して
   Checkpointing Phase 1: Prepare.
   Checkpointing Phase 2: Commit.
   Checkpointing completed.
   と表示されるのを確認してください。

   既に起動されている場合は
   omniORB: Error: Unable to create an endpoint of this description: giop:tcp::2809
   Failed to initialise the POAs. Is omniNames already running?
   と表示されるのを確認してください。

2. UCODE 受信コンポーネントの起動
   ① 端末を起動して、7.2.3で任意のディレクトリ（/home/user/selfposition/bin/linux）に移動してください。
   ② 端末上で
   $ ./RtcUcdReceiveComp
   を実行してください。

3. UCODE 解決コンポーネントの起動
   ① 端末を起動して、7.2.3で任意のディレクトリ（/home/user/SelfPosition/bin/linux）に移動してください。
   ② 端末上で
   $./RtcUcdResolveComp
   を実行してください。

4. コンテンツ解釈コンポーネントの起動
   ① 端末を起動して、7.2.3で任意のディレクトリ（/home/user/SelfPosition/bin/linux）に移動してください。
   ② 端末上で
   $./RtcCntInterpretComp
   を実行してください。

5. テストコンポーネントの起動
   ① 端末を起動して、7.2.3で任意のディレクトリ(/home/user/selfposition/bin/linux)に移動してください。
   ② 端末上で
   $./TestSelfLocalizationComp
   を実行してください。

6. Eclipse の起動
   ① 端末を起動して、7.2.2でインストールした Eclipse のインストールフォルダに移動してください。
   ② 端末上で
   $./eclipse
   を実行してください。
   ③ 図 8-1のような画面が表示されますのでワークスペースに任意のディレクトリ(ここでは、/home/user/selfposition とします。)を指定してください。

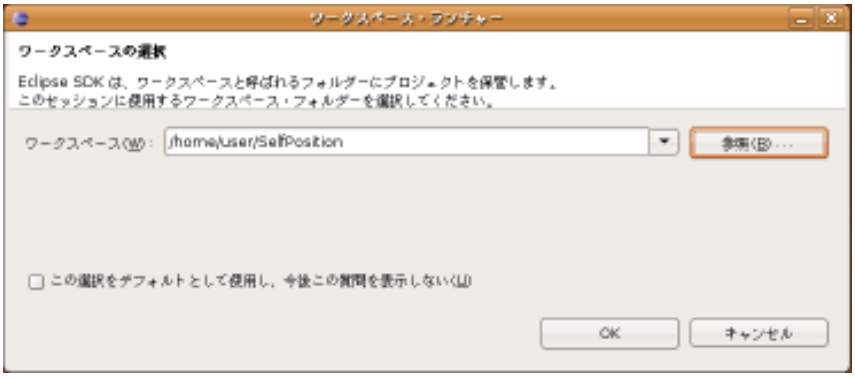


図 8-1 ワークスペース・ランチャー

7. RtcLink の起動
8. ネームサーバへの接続
9. ネームサーバに登録されているコンポーネントの確認
10. システムエディタを開く
11. コンポーネントをシステムエディタに配置
12. コンポーネントのポート間を接続
13. 全コンポーネントのポート間接続
14. コンポーネントの状態変更
15. UC の起動
16. 赤外線マーカの設定
17. 赤外線マーカの起動
18. 動作結果確認

5コンポーネントの起動手順(Windows 版)の5.1コンポーネントの起動手順の7～18と同様のため、そちらを参照願います。

9. 開発環境の構築手順（UNIX 版）

9.1. UC のモジュール開発環境構築

9.1.1. GNU 開発環境のインストール

UC 評価キットに含まれる **T-Engine/SH7727 開発キット GNU 開発環境説明書**の **2. 開発環境のインストール**と **3. インストール後の環境整備**を参考にインストールを行ってください。

9.1.2. nkf のインストール確認

1. 端末を起動してください。

2. 端末上で
   $nkf –version
   を実行してください。

3. 2.でバージョン情報が表示されない場合は、端末上で
   $sudo apt-get install nkf
   を実行してください。

4. 再度、2.を実行してバージョン情報が表示されるのを確認してください。

9.1.3. 環境自己位置同定モジュールのインストール

1. 9.1.1でインストールした開発環境のベースディレクトリ以下に
   - selfposition.tgz

   をコピーして解凍してください。

9.1.4. LibRtcIniFile ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

$ cd $BD/selfposition/lib

$ tar xvzf LibRtcIniFile.tgz

$ cd LibRtcIniFile/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/sh7727

$ make install

$ ls

9.1.5. LibRtcLog ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

$ cd $BD/selfposition/lib

$ tar xvzf LibRtcLog.tgz

$ cd LibRtcLog/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/sh7727

$ make install

$ ls

9.1.6. LibRtcRs232c ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

$ cd $BD/selfposition/lib

$ tar xvzf LibRtcRs232c.tgz

$ cd LibRtcRs232c/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/sh7727

$ make install

$ ls

9.1.7. LibRtcSemaphore ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

$ cd $BD/selfposition/lib

$ tar xvzf LibRtcSemaphore.tgz

$ cd LibRtcSemaphore/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/sh7727

$ make install

$ ls

9.1.8. LibRtcStr ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib

$ tar xvzf LibRtcStr.tgz

$ cd LibRtcStr/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/sh7727

$ make install

$ ls
```

9.1.9. 赤外線受信モジュールのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/src

$ tar xvzf IrReceiveLook.tgz

$ cd IrReceiveLook/sh7727/

$ make

$ ls
```

9.2. UNIX のコンポーネントの開発環境構築

9.2.1. OpenRTM-aist のインストール

1. OpenRTM-aist ダウンロード（C++版）
   C++版インストール（UNIX）を参考にインストールを行ってください。

2. OpenRTM-aist C++版インストールの確認
   サンプルを使ってテストを参考に動作確認を行ってください。

9.2.2. nkf のインストール確認

1. 端末を起動してください。

2. 端末上で
   $nkf –version
   を実行してください。

3. 2.でバージョン情報が表示されない場合は、端末上で
   $sudo apt-get install nkf
   を実行してください。

4. 再度、2.を実行してバージョン情報が表示されるのを確認してください。

9.2.3. 環境自己位置同定モジュールのインストール

1. 任意のディレクトリ（ここでは/home/user/とします）に
   - selfposition.tgz

   をコピーして解凍してください。

9.2.4. LibRtcIniFile ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd /home/user/selfposition/lib

$ tar xvzf LibRtcIniFile.tgz

$ cd LibRtcIniFile/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/linux

$ make

$ make install

$ ls
```

9.2.5. LibRtcLog ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd /home/user/selfposition/lib

$ tar xvzf LibRtcLog.tgz

$ cd LibRtcLog/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/linux

$ make

$ make install

$ ls
```

9.2.6. LibRtcRs232c ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

$ cd /home/user/selfposition/lib

$ tar xvzf LibRtcRs232c.tgz

$ cd LibRtcRs232c/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/linux

$ make

$ make install

$ ls

9.2.7. LibRtcSemaphore ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

$ cd /home/user/selfposition/lib

$ tar xvzf LibRtcSemaphore.tgz

$ cd LibRtcSemaphore/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/linux

$ make

$ make install

$ ls

9.2.8. LibRtcStr ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd /home/user/selfposition/lib

$ tar xvzf LibRtcStr.tgz

$ cd LibRtcStr/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/linux

$ make

$ make install

$ ls
```

9.2.9. LibSqlite3 ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd /home/user/selfposition/lib

$ tar xvzf LibSqlite3.tgz

$ cd LibSqlite3/amalgamation/build/linux

$ tar xvzf sqlite-amalgamation-3.6.18.tar.gz

$ cd sqlite-3.6.18

$ ./configure

$ sudo make install

$ ls
```

9.2.10. UCODE 受信コンポーネントのビルド

1. 端末を起動して以下のコマンドを実行してください。

$ cd /home/user/selfposition/src

$ tar xvzf RtcUcdReceive.tgz

$ cd RtcUcdReceive/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/linux

$ make

$ ls

9.2.11. UCODE 解決コンポーネントのビルド

1. 端末を起動して以下のコマンドを実行してください。

$ cd /home/user/selfposition/src

$ tar xvzf RtcUcdResolve.tgz

$ cd RtcUcdResolve/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/linux

$ make

$ ls

9.2.12. コンテンツ解釈コンポーネントのビルド

1. 端末を起動して以下のコマンドを実行してください。

$ cd /home/user/selfposition/src

$ tar xvzf RtcCntInterpret.tgz

$ cd RtcCntInterpret/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/linux

$ make

$ ls

9.2.13. テストコンポーネントのビルド

1. 端末を起動して以下のコマンドを実行してください。

$ cd /home/user/selfposition/src

$ tar xvzf TestSelfLocalization.tgz

$ cd TestSelfLocalization/sh

$ sh 環境に合わせた文字コード改行コード.sh
※文字コードを EUC、改行コードを LF の場合は EUC_LF.sh
※文字コードを UTF-8、改行コードを LF の場合は UTF-8_LF.sh

$ cd ../build/linux

$ make

$ ls

10.実行環境の構築手順(Teacube 版 UNIX 環境)

10.1. UC のモジュール実行環境構築

10.1.1. GNU 開発環境のインストール

UC 評価キットに含まれる**T-Engine/SH7727 開発キット GNU 開発環境説明書**の**2. 開発環境のインストール**を参考にインストールを行ってください。

※Teacube の開発環境もインストールする為、インストールするディレクトリを usr/local/te から usr/local/uc に変更してください。

10.1.2. ディスクの作成

1. PC と UC をシリアルケーブルで接続してください。
2. 端末を起動して、

   $ /usr/local/uc/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0

   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。

   を実行してください。
3. UC 評価キットに含まれる**UC 評価キット取扱説明書**の**2. 1 システムディスク作成方法**を参考にシステムディスクを作成してください。

10.1.3. 赤外線受信モジュールの転送

1. PC と UC をシリアルケーブルで接続してください。
2. 端末を起動して IrReceiveLook があるディレクトリまで移動してください。
3. 端末上で

   $ /usr/local/uc/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0

   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。

   を実行してください。
4. UC の電源を入れてください。
5. 端末上で

   [/SYS]% recv –cd IrReceiveLook

   を実行してください。

10.1.4. 赤外線受信モジュール設定ファイルの転送

1. IrReceive.ini を環境に合わせて変更してください。

   下記のシリアルポート設定の指定は必須です。必ず、PC と UC が接続されているポートを指定してください。

```
* ログファイル
* ログファイルディレクトリパス
LOG_FILE_PATH=/SYS/log/
* ログファイル名
LOG_FINE_NAME=IrReceive
* ログファイルモード(1:番号 2:日単位)
LOG_FILE_MODE=1
* ログファイル作成最大数(1～7)
LOG_FILE_MAX_NUM=3
* ログファイルカウンタ
LOG_FILE_COUNTER=1
* ログファイル出力(0:OFF 1:ON)
LOG_FILE_OUTPUT=1

* コンソール出力(0:OFF 1:ON)
CONSOLE_OUTPUT=1

* シリアルポート設定
* ポート
PORT=rsa
* ボーレート
BAUD_RATE=115200
* データ(0:7bit 1:8bit)
DATA=3
* パリティ(0:なし 1:奇数 2:偶数)
PARITY=0
* ストップ(0:1bit 1:2bit)
STOP=0
```

図 10-1 赤外線受信アプリケーション設定ファイル例

2. PC と UC をシリアルケーブルで接続してください。
3. 端末を起動して IrReceive.ini があるディレクトリまで移動してください。
4. 端末上で

   $ /usr/local/uc/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0

   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。

   を実行してください。
5. UC の電源を入れてください。
6. 端末上で

   [/SYS]% recv –cd IrReceive.ini

   を実行してください。

10.1.5. 自動起動の設定

UC 電源投入時に赤外線受信モジュールを自動起動させるため、以下のように STARTUP.CLI を修正してください。

```
*
*           @(#)STARTUP.CLI (UC-SH7727)
*
*           システム起動用 CLI 初期コマンドスクリプト
*           (C) Copyright 2003-2005 by Personal Media Corporation
*
/SYS/.xcli
*
* alias do /SYS/$$PROGRAM.BOX/DLED /SYS/USR 0
*
IrReceiveLook -rf1 -m1 -e0 -a1 R
*
if &DBG == 0
            do
            exit
else
            do &
endif
```

図 10-2 自動起動 STARTUP.CLI 修正例

1. PC と UC をシリアルケーブルで接続してください。
2. 端末を起動して修正した STARTUP.CLI があるディレクトリまで移動してください。
3. 端末上で

   $ /usr/local/uc/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0

   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。

   を実行してください。
4. UC の電源を入れてください。
5. 端末上で

   [/SYS]% recv –cd STARTUP.CLI

   を実行してください。

10.1.6. 非デバッグモードの設定

UC のシステムを非デバッグモードで動作させるため、以下のように DEVCONF を修正してください。

```
#
#           @(#)DEVCONF (UC-SH7727)
#
#           デバイス構成定義
#           Copyright (C) 2003-2005 by Personal Media Corporation
#

# デバッグモード
DEBUGMODE 0

# PCCARD マネージャ (0:不使用 1:使用)
CardMgrEnable     0
```

図 10-3 非デバッグモード DEVCONF 修正例

1. PC と UC をシリアルケーブルで接続してください。
2. 端末を起動して修正した DEVCONF があるディレクトリまで移動してください。
3. 端末上で

   $ /usr/local/uc/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0

   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。

   を実行してください。
4. UC の電源を入れてください。
5. 端末上で

   [/SYS]% recv –cd DEVCONF

   を実行してください。

(非デバッグモードを元に戻す方法)

非デバッグモードで起動すると、

- コンソールへの出力は無視され、入力はできない。
- コンソールはシリアルポートデバイス(rsa)としてのみ利用可能。

になります。

元に戻す場合は、

1. SD カードをすべて外して、システムを Flash ROM ディスクから起動します。
2. SD カードを入れて、
   - ① [/SYS]% att sda0 /A
   - ② [/SYS]% cd /A
   - ③ [/A]% recv –cd DEVCONF

   を実行して転送してください。

   ※DEVCONF の DEBUGMODE を 1 に修正してから転送してください。

10.2. Teacube のコンポーネント実行環境構築

10.2.1. OpenRTM-aist のインストール

1. OpenRTM-aist C++版インストール
   C++版インストール(UNIX)を参考にインストールを行ってください。

2. OpenRTM-aist C++版インストールの確認
   サンプルを使ってテストを参考に動作確認を行ってください。

10.2.2. RtcLink・RtcTemplate のインストール

1. OpenRTM-aist ダウンロード(RtcLink・RtcTemplate)
   全部入りパッケージの **UNIX 用全部入り**をダウンロードしてください。

   - eclipse32_rtclink041_rtctemplate041_linux.zip

2. RtcLink・RtcTemplate のインストール
   Java 実行環境(JRE)のインストールを参考に Java 実行環境をインストールしてください。
   インストールの仕方を参考に Eclipse、RtcLink、RtcTemplate をインストールしてください。

10.2.3. GNU 開発環境のインストール

Teacube/VR5701 評価キットに含まれる **Teacube/VR5701 評価キット GNU 開発環境説明書**の **2.開発環境のインストール**を参考にインストールを行ってください。

※UC の開発環境もインストールする為、インストールするディレクトリを usr/local/te から usr/local/teacube に変更してください。

10.2.4. ディスクの作成

1. PC と Teacube をシリアルケーブルで接続してください。
2. 端末を起動して、

   $ /usr/local/teacube/tool/Linux-i686/etc/gterm –b –X –l/dev/ttyS0

   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。

   を実行してください。
3. Teacube/VR5701 評価キットに含まれる **Teacube/VR5701 評価キット取扱説明書**の **2.1Teacube 上での起動ディスク作成**を参考に起動ディスクを作成してください。

10.2.5. 環境自己位置同定コンポーネントの転送

1. 任意のディレクトリ（ここでは/home/user/とします）に
   - selfposition.tgz

   をコピーしてください。

2. 端末を起動して

   $cd /home/user/

   を実行してください。

3. 端末上で

   $tar –xvzf selfposition.tgz

   $cd selfposition/bin/vr5701

   を実行してください。

4. /home/user/selfposition/bin/vr5701/rtc.conf を環境に合わせて変更してください。

   下記のネームサーバの指定は必須です。必ず、ネームサーバを起動する PC の IP アドレス:ポートを指定してください。

```
* rtc.conf
corba.nameservers: localhost:2809
naming.formats: %n.rtc
```

図 10-4 コンフィグレーションファイル例

5. /home/user/selfposition/bin/vr5701/UcdReceive.ini を環境に合わせて変更してください。
下記のシリアルポートの指定は必須です。必ず、PC と UC が接続されているポートを指定してください。

```
* ログファイル
* ログファイルディレクトリパス
LOG_FILE_PATH=./log/
* ログファイル名
LOG_FILE_NAME=UcdReceive
* ログファイルモード(1:番号 2:日単位)
LOG_FILE_MODE=2
* ログファイル作成最大数(1～7)
LOG_FILE_MAX_NUM=3
* ログファイルカウンタ
LOG_FILE_COUNTER=1
* ログファイル出力(0:OFF 1:ON)
LOG_FILE_OUTPUT=1

* コンソール出力(0:OFF 1:ON)
CONSOLE_OUTPUT=1

* シリアルポート
PORT=rsa
* ボーレート
BAUD_RATE=115200
* データ(0:7bit 1:8bit)
DATA=1
* パリティ(0:なし 1:奇数 2:偶数)
PARITY=0
* ストップ(0:1bit 1:2bit)
STOP=0
* シリアル通信タイムアウト時間
READ_TIMEOUT=5000

* 同じ ucode を受信した場合の time 設定(秒)
OUT_CYCLE=10
```

図 10-5 ucode リーダコンポーネント設定ファイル例

6. /home/user/selfposition/bin/vr5701/UcdResolve.txt を環境に合わせて変更してください。
   UcdResolve.txt の詳細は2.5.5を参照のこと

   UcdResolve.txt は、
   UCODE,UCODE に対応するコンテンツキャッシュファイルパス
   の形式で記載してください。

```
00000000000000000000000000101E3,./Contents/00000000000000101E3
00000000000000000000000000010164,./Contents/00000000000000010164
```

図 10-6 UcdResolve.txt 例

7. 6. で指定したコンテンツキャッシュファイルパスにコンテンツキャッシュファイルを作成する。
   （上記、図 10-6 UcdResolve.txt 例のデータ例の場合は
   /home/user/Selfposition/bin/vr5701/Contents/00000000000000101E3
   /home/user/Selfposition/bin/vr5701/Contents/00000000000000010164
   を作成してください。）

   作成したコンテンツキャッシュファイルを環境に合わせて変更してください。
   コンテンツキャッシュファイルの詳細は2.5.6を参照のこと
   コンテンツキャッシュファイルは
   緯度, 経度, 海抜, 誤差
   の形式で記載してください。

```
141.121212,43.154554,512,0
```

図 10-7 ./Contents/00000000000000101E3 例

8. 端末上で
   $ /usr/local/teacube/tool/Linux-i686/etc/gterm –b –X –l/dev/ttyS0
   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。
   を実行してください。

9. Teacube の電源を入れてください。

10. 端末上で

[/SYS]% ux/mkdir log
[/SYS]% ux/mkdir Contents
[/SYS]% cd Contents
[/SYS/Contents]% recv –cd Contents/00000000000000101E3
[/SYS/Contents]% recv –cd Contents/00000000000000010164
[/SYS]% cd ../
[/SYS]% ux/mkdir lib
[/SYS]% cd lib
[/SYS]% recv –cd lib/libdl.so.2
[/SYS]% recv –cd lib/libSQLite.so.1
[/SYS]% cd ../
[/SYS]% recv –cd rtc.conf
[/SYS]% recv –cd RtcUcdReceive
[/SYS]% recv –cd UcdReceive.ini
[/SYS]% recv –cd RtcUcdResolve
[/SYS]% recv –cd UcdResolve.ini
[/SYS]% recv –cd UcdResolve.txt
[/SYS]% recv –cd RtcCntInterpret
[/SYS]% recv –cd CntInterpret.ini
[/SYS]% sysconf TMaxSemId 500
[/SYS]% sysconf TMaxFlgId 500
[/SYS]% sysconf TMaxTskId 500

を実行してください。

10.2.6. Teacube のネットワーク設定

1. PC と Teacube をシリアルケーブルで接続してください。
2. 端末上で

$ /usr/local/teacube/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0
※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。

を実行してください。

3. Teacube の電源を入れてください。
4. 端末上で

[/SYS]% netconf –c

で環境に合わせてネットワークの設定を行ってください。

10.2.7. 自動起動の設定

Teacube 電源投入時に環境自己位置同定コンポーネントを自動起動させるため、以下のように STARTUP.CLI を修正してください。

```
*
*           @(#)STARTUP.CLI (Teacube/VR5701 : T-Shell)
*
*           システム起動用 CLI 初期コマンドスクリプト
*           (C) Copyright 2004 by Personal Media Corporation
*
* CLI 起動
/SYS/.xcli

* 仮身一覧
* alias do /SYS/$$PROGRAM.BOX/DLED /SYS/USR 0

* コンソール
alias do /SYS/$$PROGRAM.BOX/CONSOLE /SYS/USR 0

* 環境自己位置同定モジュール起動
setup.sh &

* コンソール起動
if &DBG == 0
            do
            exit
else
            do &
endif
```

図 10-8 自動起動 STARTUP.CLI 修正例

1. PC と Teacube をシリアルケーブルで接続してください。
2. 端末を起動して修正した STARTUP.CLI、setup.sh があるディレクトリまで移動してください。
3. 端末上で

   $ /usr/local/teacube/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0

   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。

   を実行してください。
4. teacube の電源を入れてください。
5. 端末上で

   [/SYS]% recv –cd STARTUP.CLI

   [/SYS]% recv –cd setup.sh

   を実行してください。

10.2.8. 非デバッグモードの設定

UC のシステムを非デバッグモードで動作させるため、以下のように DEVCONF を修正してください。

```
#
#           @(#)DEVCONF (Teacube/VR5701 : T-Shell)
#
#           デバイス構成定義
#           Copyright (C) 2004 by Personal Media Corporation
#

#  デバッグモード
DEBUGMODE 0

# USB マネージャ (0:不使用  1:使用)
USBMgrEnable      1
```

図 10-9 非デバッグモード DEVCONF 修正例

1. PC と Teacube をシリアルケーブルで接続してください。
2. 端末を起動して修正した DEVCONF があるディレクトリまで移動してください。
3. 端末上で

   $ /usr/local/teacube/tool/Linux-i686/etc/gterm –B –X -l/dev/ttyS0

   ※/dev/ttyS0 の箇所は、使用する回線デバイス名を指定してください。

   を実行してください。
4. teacube の電源を入れてください。
5. 端末上で

   [/SYS]% recv –cd DEVCONF

   を実行してください。

(非デバッグモードを元に戻す方法)

非デバッグモードで起動すると、

- コンソールへの出力は無視され、入力はできない。
- コンソールはシリアルポートデバイス(rsa)としてのみ利用可能。

になります。

元に戻す場合は、

1. Teacube の DIP SW4 を ON にしてモニタ起動にしてください。
   詳細は、**Teacube/VR5701 評価キット取扱説明書**の **1.3 ハードウェア取扱方法**を参照願います。
2. 端末上で
   TM> bd rda
   [/SYS]% det hda0
   [/SYS]% att hda0 /A
   [/SYS]% cd /A
   [/A]% recv –cd DEVCONF
   を実行してください。
   ※DEVCONF の DEBUGMODE を 1 に修正してから転送してください。

11.コンポーネントの起動手順（Teacube 版　UNIX 環境）

11.1.　コンポーネントの起動手順

1. CORBA ネームサーバの起動確認
   端末を起動して、
   $rtm-naming 2809
   を実行して
   Checkpointing Phase 1: Prepare.
   Checkpointing Phase 2: Commit.
   Checkpointing completed.
   と表示されるのを確認してください。

   既に起動されている場合は
   omniORB: Error: Unable to create an endpoint of this description: giop:tcp::2809
   Failed to initialise the POAs. Is omniNames already running?
   と表示されるのを確認してください。

2. 環境自己位置同定コンポーネントの起動
   ① Teacube と UC をシリアルケーブルで接続してください。
   ② Teacube に LAN ケーブルを接続してください。
   ③ Teacube の電源を入れてください。

3. テストコンポーネントの起動
   ① 端末を起動して、/home/user/selfposition/bin/vr5701 に移動してください。
   ② 端末上で
   $./TestSelfLocalizationComp
   を実行してください。

4. Eclipse の起動
   ① 端末を起動して、10.2.2でインストールした Eclipse のインストールフォルダに移動してください。
   ② 端末上で
      $./eclipse
      を実行してください。
   ③ 図 11-1のような画面が表示されますのでワークスペースに任意のディレクトリ(ここでは、/home/user/SelfPosition とします。)を指定してください。

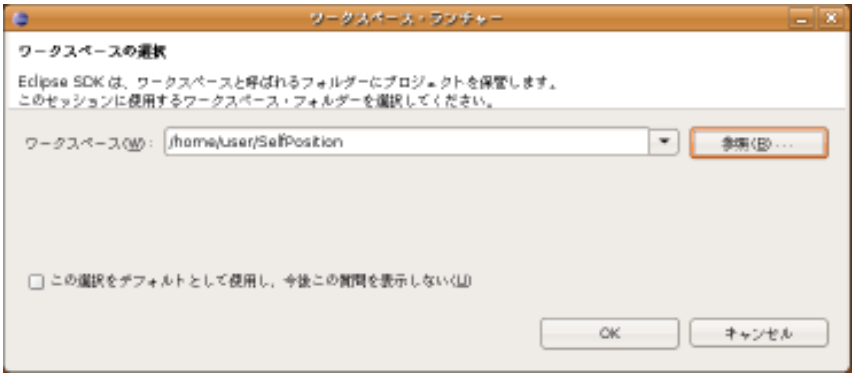


図 11-1 ワークスペース・ランチャー

5. RtcLink の起動
6. ネームサーバへの接続
7. ネームサーバに登録されているコンポーネントの確認
8. システムエディタを開く
9. コンポーネントをシステムエディタに配置
10. コンポーネントのポート間を接続
11. 全コンポーネントのポート間接続
12. コンポーネントの状態変更
13. UC の起動
14. 赤外線マーカの設定
15. 赤外線マーカの起動
16. 動作結果確認

5コンポーネントの起動手順(Windows 版)の5.1コンポーネントの起動手順の7～18と同様のため、そちらを参照願います。

※7. ネームサーバへの接続では localhost ではなく、ネームサーバを起動した PC の IP アドレスを指定してください。

12.開発環境の構築手順(Teacube 版 UNIX 環境)

12.1. UC のモジュール開発環境構築

12.1.1. GNU 開発環境のインストール

UC 評価キットに含まれる **T-Engine/SH7727 開発キット GNU 開発環境説明書**の **2. 開発環境のインストール**と **3. インストール後の環境整備**を参考にインストールを行ってください。

※Teacube の開発環境もインストールする為、インストールするディレクトリを usr/local/te から usr/local/uc に変更してください。

12.1.2. nkf のインストール確認

1. 端末を起動してください。

2. 端末上で
   $nkf –version
   を実行してください。

3. 2.でバージョン情報が表示されない場合は、端末上で
   $sudo apt-get install nkf
   を実行してください。

4. 再度、2.を実行してバージョン情報が表示されるのを確認してください。

12.1.3. 環境自己位置同定モジュールのインストール

1. 12.1.1でインストールした開発環境のベースディレクトリ以下に
   - selfposition.tgz

   をコピーして解凍してください。

12.1.4. LibRtcIniFile ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcIniFile.tgz
$ cd LibRtcIniFile/sh
$ sh EUC_LF.sh
$ cd ../build/sh7727
$ make install
$ ls
```

12.1.5. LibRtcLog ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcLog.tgz
$ cd LibRtcLog/sh
$ sh EUC_LF.sh
$ cd ../build/sh7727
$ make install
$ ls
```

12.1.6. LibRtcRs232c ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcRs232c.tgz
$ cd LibRtcRs232c/sh
$ sh EUC_LF.sh
$ cd ../build/sh7727
$ make install
$ ls
```

12.1.7. LibRtcSemaphore ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcSemaphore.tgz
$ cd LibRtcSemaphore/sh
$ sh EUC_LF.sh
$ cd ../build/sh7727
$ make install
$ ls
```

12.1.8. LibRtcStr ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcStr.tgz
$ cd LibRtcStr/sh
$ sh EUC_LF.sh
$ cd ../build/sh7727
$ make install
$ ls
```

12.1.9. 赤外線受信モジュールのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/src
$ tar xvzf IrReceiveLook.tgz
$ cd IrReceiveLook/sh7727/
$ make
$ ls
```

12.2. Teacube のコンポーネントの開発環境構築

12.2.1. GNU 開発環境のインストール

Teacube/VR5701 評価キットに含まれる **Teacube/VR5701 評価キット GNU 開発環境説明書**の **2.開発環境のインストールと 3. インストール後の環境整備**を参考にインストールを行ってください。

※UC の開発環境もインストールする為、インストールするディレクトリを usr/local/te から usr/local/teacube に変更してください。

12.2.2. OpenRTM-aist のインストール

12.2.2.1 POSIX ライブラリのインストール

1. nes-libposix-0.1.0.tar.gz を12.2.1でインストールした開発環境のベースディレクトリ/lib 以下にコピーしてください。
2. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/lib
$ tar xvzf nes-libposix-0.1.0.tar.gz
$ cd nes-libposix-0.1.0
$ mkdir vr5701
$ cd vr5701
$ ln –s ../Makefile
$ make
$ cd ../
$ mkdir vr5701.so
$ cd vr5701.so
$ ln –s ../Makefile
$ make
```

3. .bashrc 設定ファイルに環境変数 **export NES_POSIX_PATH=$BD/lib/nes-posix-0.1.0** を追加してください。

12.2.2.1 omniORB ライブラリのインストール

1. omniORB-4.0.7.tar.gz を12.2.1でインストールした開発環境のベースディレクトリ/lib 以下にコピーしてください。
2. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/lib
$ tar xvzf omniORB-4.0.7.tar.gz
$ cd omniORB-4.0.7
$ mkdir vr5701
$ cd vr5701
$ ln –s ../Makefile.te Makefile
$ make
$ cd ../
$ mkdir vr5701.so
$ cd vr5701.so
$ ln –s ../Makefile.te Makefile
$ make
```

12.2.2.2 ACE ライブラリのインストール

1. libace.tar.gz を12.2.1でインストールした開発環境のベースディレクトリ/lib 以下にコピーしてください。
2. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/lib
$ tar xvzf libace.tar.gz
$ cd libace
$ mkdir vr5701
$ cd vr5701
$ ln –s ../ace/Makefile
$ make install
$ cd ../
$ mkdir vr5701.so
$ cd vr5701.so
$ ln –s ../ace/Makefile
$ make install
```

12.2.2.3 OpenRTM ライブラリのインストール

1. OpenRTM-aist-0.4.1.tar.gz を12.2.1でインストールした開発環境のベースディレクトリ/lib 以下にコピーしてください。
2. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/lib
$ tar xvzf OpenRTM-aist-0.4.1.tar.gz
$ cd OpenRTM-aist-0.4.1/rtm
$ mkdir vr5701
$ cd vr5701
$ ln –s ../Makefile
$ make install
$ cd ../
$ mkdir vr5701.so
$ cd vr5701.so
$ ln –s ../Makefile
$ make install
```

12.2.3. nkf のインストール確認

1. 端末を起動してください。

2. 端末上で
   $nkf –version
   を実行してください。

3. 2.でバージョン情報が表示されない場合は、端末上で
   $sudo apt-get install nkf
   を実行してください。

4. 再度、2.を実行してバージョン情報が表示されるのを確認してください。

12.2.4. 環境自己位置同定モジュールのインストール

1. 12.2.1でインストールした開発環境のベースディレクトリ以下に
   - selfposition.tgz

   をコピーして解凍してください。

12.2.5. LibRtcIniFile ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcIniFile.tgz
$ cd LibRtcIniFile/sh
$ sh EUC_LF.sh
$ cd ../build/vr5701
$ make install
$ ls
$ cd ../vr5701.so
$ make install
$ ls
```

12.2.6. LibRtcLog ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcLog.tgz
$ cd LibRtcLog/sh
$ sh EUC_LF.sh
$ cd ../build/vr5701
$ make install
$ ls
$ cd ../vr5701.so
$ make install
$ ls
```

12.2.7. LibRtcRs232c ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcRs232c.tgz
$ cd LibRtcRs232c/sh
$ sh EUC_LF.sh
$ cd ../build/vr5701
$ make install
$ ls
$ cd ../vr5701.so
$ make install
$ ls
```

12.2.8. LibRtcSemaphore ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcSemaphore.tgz
$ cd LibRtcSemaphore/sh
$ sh EUC_LF.sh
$ cd ../build/vr5701
$ make install
$ ls
$ cd ../vr5701.so
$ make install
$ ls
```

12.2.9. LibRtcStr ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibRtcStr.tgz
$ cd LibRtcStr/sh
$ sh EUC_LF.sh
$ cd ../build/vr5701
$ make install
$ ls
$ cd ../vr5701.so
$ make install
$ ls
```

12.2.10. LibSqlite3 ライブラリのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd $BD/selfposition/lib
$ tar xvzf LibSqlite3.tgz
$ cd LibSqlite3/amalgamation/sh
$ sh EUC_LF.sh
$ cd ../build/vr5701.so
$ make install
$ ls
```

12.2.11. UCODE 受信コンポーネントのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd /home/user/selfposition/src
$ tar xvzf RtcUcdReceive.tgz
$ cd RtcUcdReceive/sh
$ sh EUC_LF.sh
$ cd ../build/vr5701
$ make
$ ls
```

12.2.12. UCODE 解決コンポーネントのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd /home/user/selfposition/src
$ tar xvzf RtcUcdResolve.tgz
$ cd RtcUcdResolve/sh
$ sh EUC_LF.sh
$ cd ../build/vr5701.dl
$ make
$ ls
```

12.2.13. コンテンツ解釈コンポーネントのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd /home/user/selfposition/src
$ tar xvzf RtcCntInterpret.tgz
$ cd RtcCntInterpret/sh
$ sh EUC_LF.sh
$ cd ../build/vr5701
$ make
$ ls
```

12.2.14. テストコンポーネントのビルド

1. 端末を起動して以下のコマンドを実行してください。

```
$ cd /home/user/selfposition/src
$ tar xvzf TestSelfLocalization.tgz
$ cd TestSelfLocalization/sh
$ sh EUC_LF.sh
$ cd ../build/vr5701
$ make
$ ls
```

保 護 紙