WM-RP-0xS
サンプルプログラム解説（AP-RX63N-0A）

２０１３年３月０７日

目　次

1. 概要

1.1 概要

本アプリケーションノートでは、弊社製 AP-RX63N-0A 上で動作する、WM-RP-0xS 用サンプルプログラムについて解説します。
本サンプルプログラムの概要を以下に示します。

サンプルプログラム	ターゲットボード	動作内容
TCP/IP 通信サンプルプログラム （アドホックモード　クリエータ）	・AP-RX63N-0A	・TCP/IP　アドホックモード（クリエータ） エコーバックサーバ ・シリアル通信 ・タイマ割り込み
TCP/IP 通信サンプルプログラム （アドホックモード　ジョイナー）	・AP-RX63N-0A	・TCP/IP　アドホックモード（ジョイナー） エコーバックサーバ ・シリアル通信 ・タイマ割り込み
TCP/IP 通信サンプルプログラム （インフラストラクチャモード）	・AP-RX63N-0A	・TCP/IP　インフラストラクチャモード エコーバックサーバ ・シリアル通信 ・タイマ割り込み
UDP 通信サンプルプログラム （アドホックモード　クリエータ）	・AP-RX63N-0A	・UDP　アドホックモード（クリエータ） エコーバックサーバ ・シリアル通信 ・タイマ割り込み
UDP 通信サンプルプログラム （アドホックモード　ジョイナー）	・AP-RX63N-0A	・UDP　アドホックモード（ジョイナー） エコーバックサーバ ・シリアル通信 ・タイマ割り込み
UDP 通信サンプルプログラム （インフラストラクチャモード）	・AP-RX63N-0A	・UDP　インフラストラクチャモード エコーバックサーバ ・シリアル通信 ・タイマ割り込み

1.2　動作環境

各サンプルプログラムの動作確認に必要な機器を以下に記します。

サンプルプログラム	動作確認に必要な機器
TCP/IP 通信サンプルプログラム （アドホックモード　クリエータ）	・アドホック通信可能なホスト PC
TCP/IP 通信サンプルプログラム （アドホックモード　ジョイナー）	・アドホック通信可能なホスト PC
TCP/IP 通信サンプルプログラム （インフラストラクチャモード）	・ネットワーク通信可能なホスト PC
UDP 通信サンプルプログラム （アドホックモード　クリエータ）	・アドホック通信可能なホスト PC
UDP 通信サンプルプログラム （アドホックモード　ジョイナー）	・アドホック通信可能なホスト PC
UDP 通信サンプルプログラム （インフラストラクチャモード）	・ネットワーク通信可能なホスト PC

1.3　ネットワーク構成イメージ図

以下に、「インフラストラクチャ」と「アドホック」時のネットワーク構成イメージ図を示します。

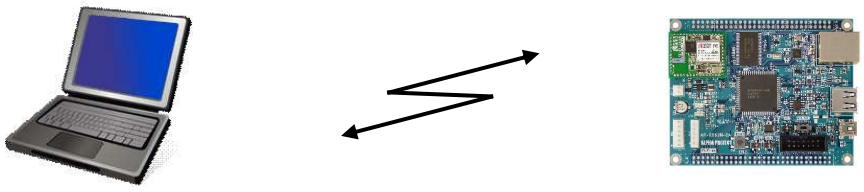
・インフラストラクチャ

アクセスポイント経由で、無線通信を行います。

無線 LAN 搭載 PC　　アクセスポイント　　AP-RX63N-0A

・アドホック

アドホック通信では、クリエータ（親）となった機器に、ジョイナー（子）となった機器が接続して通信を行います。



無線 LAN 搭載 PC
（アドホック可能）

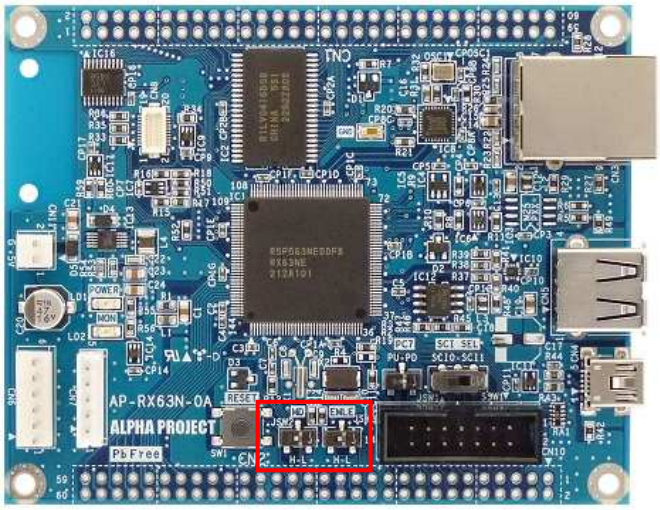
アドホック
クリエータ or ジョイナー

AP-RX63N-0A

アドホック
クリエータ or ジョイナー

1.4　動作モード

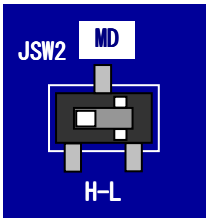
本サンプルプログラムは、AP-RX63N-0A で動作します。CPU 動作モード、各メモリ設定は下記のようになっています。
モードの設定方法等につきましては、「AP-RX63N-0A ハードウェアマニュアル」をご覧下さい。



CPU ボードの設定を製品出荷時の状態とし、使用方法に合わせて以下の各スイッチの設定を行って下さい。

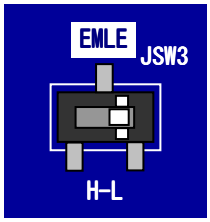
プログラム動作時

・JSW2



〈JSW2 設定〉
H　　：シングルチップモード

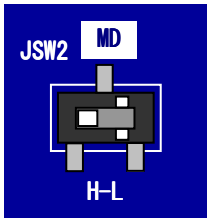
・JSW3



〈JSW3 設定〉
L　　：オンチップエミュレータを使用しない

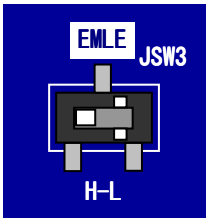
E1 エミュレータ使用時

・JSW2



〈JSW2 設定〉
H　　：シングルチップモード

・JSW3



〈JSW3 設定〉
H　　：オンチップエミュレータを使用する

Fig 1.4-1　動作モード設定

1.5　開発環境について

本サンプルプログラムは総合開発環境 High-performance Embedded Workshop を用いて開発されております。
サンプルプログラムに対応する開発環境、コンパイラのバージョンは次のようになります。

開発環境	バージョン	コンパイラ名	バージョン	備考
High-performance Embedded Workshop	V 4.09 00.007 以降	RXC　※1	V1.2.1.0 以降	RX ファミリ用 C/C++コンパイラパッケージに付属

※1：「RX ファミリ用 C/C++コンパイラパッケージ」です。ルネサスエレクトロニクス社のウェブサイトより評価版をダウンロードできます。

1.6　ワークスペースについて

本サンプルプログラムの総合開発環境 High-performance Embedded Workshop ワークスペースは次のフォルダに格納されています。

サンプルプログラム	フォルダ
TCP/IP サンプルプログラム（アドホックモード　クリエータ）	¥sample¥adhoc¥ap_rx63n_0a_tcpip_create¥ap_rx63n_0a_tcpip_create.hws
TCP/IP サンプルプログラム（アドホックモード　ジョイナー）	¥sample¥adhoc¥ap_rx63n_0a_tcpip_join¥ap_rx63n_0a_tcpip_join.hws
UDP サンプルプログラム（アドホックモード　クリエータ）	¥sample¥adhoc¥ap_rx63n_0a_udp_create¥ap_rx63n_0a_udp_create.hws
UDP サンプルプログラム（アドホックモード　ジョイナー）	¥sample¥adhoc¥ap_rx63n_0a_udp_join¥ap_rx63n_0a_udp_join.hws
TCP/IP サンプルプログラム（インフラストラクチャモード）	¥sample¥infrastructure¥ap_rx63n_0a_tcp¥ap_rx63n_0a_tcp.hws
UDP サンプルプログラム（インフラストラクチャモード）	¥sample¥infrastructure¥ap_rx63n_0a_udp¥ap_rx63n_0a_udp.hws

2.　サンプルプログラムの構成

2.1　フォルダ構成

サンプルプログラムは下記のようなフォルダ構成になっています。

sample	
├ adhoc	アドホックモードサンプルプログラムフォルダ
│ ├ ap_rx63n_0a_tcpip_create	TCP/IP 通信サンプルプログラムフォルダ（アドホック　クリエータ）
│ │ ├ Debug	デバッグ動作用ワークフォルダ
│ │ ├ Release	ROM 化動作用ワークフォルダ
│ │ └ src	ソースファイルフォルダ
│ ├ ap_rx63n_0a_tcpip_join	TCP/IP 通信サンプルプログラムフォルダ（アドホック　ジョイナー）
│ │ ├ Debug	デバッグ動作用ワークフォルダ
│ │ ├ Release	ROM 化動作用ワークフォルダ
│ │ └ src	ソースファイルフォルダ
│ ├ ap_rx63n_0a_udp_create	UDP 通信サンプルプログラムフォルダ（アドホック　クリエータ）
│ │ ├ Debug	デバッグ動作用ワークフォルダ
│ │ ├ Release	ROM 化動作用ワークフォルダ
│ │ └ Src	ソースファイルフォルダ
│ ├ ap_rx63n_0a_udp_join	UDP 通信サンプルプログラムフォルダ（アドホック　ジョイナー）
│ │ ├ Debug	デバッグ動作用ワークフォルダ
│ │ ├ Release	ROM 化動作用ワークフォルダ
│ │ └ src	ソースファイルフォルダ
└ infrastructure	インフラストラクチャモードサンプルプログラムフォルダ
├ ap_rx63n_0a_tcp	TCP/IP 通信サンプルプログラムフォルダ（インフラストラクチャ）
│ ├ Debug	デバッグ動作用ワークフォルダ
│ ├ Release	ROM 化動作用ワークフォルダ
│ └ src	ソースファイルフォルダ
├ ap_rx63n_0a_udp	UDP 通信サンプルプログラムフォルダ（インフラストラクチャ）
├ Debug	デバッグ動作用ワークフォルダ
├ Release	ROM 化動作用ワークフォルダ
└ src	ソースファイルフォルダ

2.2　ファイル構成

サンプルプログラムは以下のファイルで構成されています。

＜¥sample フォルダ内＞

adhoc	・・・	アドホックモードサンプルプログラム
infrastructure	・・・	インフラストラクチャモードサンプルプログラム

［アドホック］

＜¥sample¥adhoc フォルダ内＞

ap_rx63n_0a_tcpip_create	・・・	TCP/IP 通信サンプルプログラムフォルダ （アドホック　クリエータ）
ap_rx63n_0a_tcpip_join	・・・	TCP/IP 通信サンプルプログラムフォルダ （アドホック　ジョイナー）
ap_rx63n_0a_udp_create	・・・	UDP 通信サンプルプログラムフォルダ （アドホック　クリエータ）
ap_rx63n_0a_udp_join	・・・	UDP 通信サンプルプログラムフォルダ （アドホック　ジョイナー）

［インフラストラクチャ］

＜¥sample¥infrastructure フォルダ内＞

ap_rx63n_0a_tcp	・・・	TCP/IP 通信サンプルプログラムフォルダ
ap_rx63n_0a_udp	・・・	UDP 通信サンプルプログラムフォルダ

※以下、「TCP/IP アドホック　クリエータ」のフォルダ　及びファイル構成に関して記述致しますが、それ以外の
「TCP/IP アドホック　ジョイナー」
「UDP アドホック　クリエータ」
「UDP アドホック　ジョイナー」
「TCP/IP　インフラストラクチャ」
「UDP インフラストラクチャ」
は、フォルダ名、ファイル名が違うだけとなります。

＜¥sample¥adhoc¥ap_rx63n_0a_tcpip_create フォルダ内＞

ap_rx63n_0a_tcpip_create.hws	・・・	TCP/IP 通信サンプルプログラム HEW 用ワークスペース ファイル（アドホック　クリエータ）

＜¥sample¥adhoc¥ap_rx63n_0a_ tcpip_create¥ap_rx63n_0a_tcpip_create フォルダ内＞

ap_rx63n_0a_tcpip_create.hwp	・・・	TCP/IP 通信サンプルプログラム HEW 用プロジェクト ファイル（アドホック　クリエータ）

＜¥sample¥adhoc¥ap_rx63n_0a_tcpip_create¥ap_rx63n_0a_tcpip_create¥Debug フォルダ内＞

ap_rx63n_0a_tcpip_create.abs	・・・	TCP/IP 通信サンプルプログラムデバッグ動作用オブジェクト ファイル（アドホック　クリエータ） (elf 形式)
ap_rx63n_0a_tcpip_create.mot	・・・	TCP/IP 通信サンプルプログラムデバッグ動作用 モトローラ S フォーマット形式ファイル （アドホック　クリエータ）
ap_rx63n_0a_tcpip_create.map	・・・	TCP/IP 通信サンプルプログラムデバッグ動作用マップファイル （アドホック　クリエータ） コンパイル後は、.obj, .lib 等のファイルが生成されます

＜¥sample¥adhoc¥ap_rx63n_0a_tcpip_create¥ap_rx63n_0a_tcpip_create¥Release フォルダ内＞

ap_rx63n_0a_tcpip_create.abs	・・・	TCP/IP 通信サンプルプログラム ROM 化動作用オブジェクト ファイル（アドホック　クリエータ） （elf 形式）
ap_rx63n_0a_tcpip_create.mot	・・・	TCP/IP 通信サンプルプログラム ROM 化動作用 モトローラ S フォーマット形式ファイル （アドホック　クリエータ）
ap_rx63n_0a_tcpip_create.map	・・・	TCP/IP 通信サンプルプログラム ROM 化動作用マップファイル （アドホック　クリエータ） コンパイル後は、.obj,.lib 等のファイルが生成されます

＜¥sample¥adhoc¥ap_rx63n_0a_tcpip_create¥ap_rx63n_0a_tcpip_create¥src フォルダ内＞

ap_rx63n_0a_tcp.c	・・・	main 関数処理
dbsct.c	・・・	CPU 初期化処理ファイル
intprg.c	・・・	割り込みベクタ処理関数ファイル
resetprg.c	・・・	PowerOn リセット時の動作処理ファイル
sample.c	・・・	サンプルプログラムメイン処理ファイル
sbrk.c	・・・	ヒープメモリ処理ファイル
sci.c	・・・	シリアル処理ファイル
sci_spi.c	・・・	簡易 SPI 処理ファイル
tmr.c	・・・	タイマ処理ファイル
vecttbl.c	・・・	割込ベクタテーブルファイル
wm_rp_04s.c	・・・	WiFi モジュールサンプルドライバファイル
BoardDepend.h	・・・	ボード依存ファイル
common.h	・・・	共通ヘッダファイル
iodefine.h	・・・	RX63N 内部レジスタ定義ヘッダファイル
sbrk.h	・・・	ヒープメモリサイズ定義ファイル
stacksct.h	・・・	スタックサイズ定義ファイル
typedefine.h	・・・	typedef 定義用ヘッダファイル
vect.h	・・・	割り込みベクタ処理関数定義ヘッダファイル
wm_rp_04s.h	・・・	WiFi モジュールサンプルドライバヘッダファイル
lowlvl.src	・・・	入出力関連低レベル処理ファイル

※以下のフォルダ内のファイルは、「Redpine Signals 社」の SPI 用ライブラリファイルとなります。

＜¥sample¥adhoc¥ap_rx63n_0a_tcpip_create¥ap_rx63n_0a_tcpip_create¥src¥API_Lib フォルダ内＞

rsi_api_sysinit.c	・・・	WM-RP 初期化処理
rsi_hal_mcu_interrupt.c	・・・	MCU 依存割り込み処理
rsi_hal_mcu_ioports.c	・・・	MCU 依存 I/O 処理
rsi_hal_mcu_spi.c	・・・	MCU 依存 SPI 処理
rsi_hal_mcu_timers.c	・・・	MCU 依存タイマ処理
rsi_interrupt.c	・・・	WM-RP 各割り込み確認処理
rsi_lib_util.c	・・・	データ変換用ユーティリティ
rsi_spi_api.c	・・・	SPI による送信時の各処理定義データファイル
rsi_spi_band.c	・・・	band 処理
rsi_spi_bootloader.c	・・・	bootloader 処理
rsi_spi_disconnect.c	・・・	disconnect 処理
rsi_spi_execute_cmd.c	・・・	SPI による各処理コマンド送信処理
rsi_spi_feat_select.c	・・・	Feature Select 処理
rsi_spi_framerdwr.c	・・・	SPI Frame description 処理
rsi_spi_functs.c	・・・	SPI による送信コマンド C1,C2,C3,C4 処理
rsi_spi_fwupgrade.c	・・・	FW upgrade 処理
rsi_spi_http_get.c	・・・	HTTP Get Request 処理
rsi_spi_http_post.c	・・・	HTTP Post Request 処理
rsi_spi_iface_init.c	・・・	WM-RP 初期化コマンド送信処理
rsi_spi_init.c	・・・	init 処理
rsi_spi_interrupt_handler.c	・・・	WM-RP からの割り込みステータス取得処理
rsi_spi_ipparam.c	・・・	Set IP Parameters 処理
rsi_spi_join.c	・・・	join 処理
rsi_spi_memrdwr.c	・・・	SPI Memory read/write 処理
rsi_spi_mode_select.c	・・・	TCP/IP Bypass 処理
rsi_spi_power_mode.c	・・・	Power Mode 処理
rsi_spi_query_bssid_nwtype.c	・・・	Query MAC address and Network Type of Scanned Networks 処理
rsi_spi_query_conn_status.c	・・・	Query Connection Status 処理
rsi_spi_query_dhcp_parms.c	・・・	Query DHCP Information 処理
rsi_spi_query_dns.c	・・・	DNS Request 処理
rsi_spi_query_fwversion.c	・・・	Query Firmware Version 処理
rsi_spi_query_macaddress.c	・・・	Query MAC Address 処理
rsi_spi_query_net_parms.c	・・・	Query Network Parameters 処理
rsi_spi_query_rssi.c	・・・	Query RSSI Value 処理
rsi_spi_read_packet.c	・・・	Receive data 処理
rsi_spi_regrdwr.c	・・・	WM-RP 内部レジスタアクセス処理
rsi_spi_scan.c	・・・	Scan 処理
rsi_spi_send_data.c	・・・	Send Data 処理
rsi_spi_send_ludp_data.c	・・・	Listening UDP Send 処理
rsi_spi_send_wps_data.c	・・・	WPS Send 処理
rsi_spi_set_listen_interval.c	・・・	Set a Listen Interval 処理
rsi_spi_set_mac_addr.c	・・・	Set MAC Address 処理
rsi_spi_sleep_timer.c	・・・	Sleep timer 設定処理
rsi_spi_socket.c	・・・	Open a Socket 処理
rsi_spi_socket_close.c	・・・	Close a Socket 処理

rsi_spi_store_config.c　…　Connecting to a Preconfigured Access Point 処理
rsi_spi_wepkeys.c　…　WEP Key 設定処理
rsi_spi_wl_bootloader.c　…　Wireless bootloader 処理
rsi_spi_wlfw_upgrade.c　…　Wireless FWupgrade 処理
rsi_hal.h　…　ハードウェア依存処理

rsi_lib_util.h　…　データ変換用ユーティリティヘッダファイル
rsi_spi_api.h　…　SPI による送信時の各処理定義データヘッダファイル
Makefile　…　メイクファイル

＜¥sample¥adhoc¥ap_rx63n_0a_tcpip_create¥ap_rx63n_0a_tcpip_create¥src¥API_Lib¥Firmware フォルダ内＞

ffdata　…　FW ファイル　※このファイルは、弊社からは提供しておりません。
ffinst1　…　FW ファイル　※このファイルは、弊社からは提供しておりません。
ffinst2　…　FW ファイル　※このファイルは、弊社からは提供しておりません。
iudata　…　FW ファイル　※このファイルは、弊社からは提供しておりません。
iuinst1　…　FW ファイル　※このファイルは、弊社からは提供しておりません。
iuinst2　…　FW ファイル　※このファイルは、弊社からは提供しておりません。
sbdata1　…　bootloader ファイル
sbdata2　…　bootloader ファイル
sbinst1　…　bootloader ファイル
sbinst2　…　bootloader ファイル

＜¥sample¥adhoc¥ap_rx63n_0a_tcpip_create¥ap_rx63n_0a_tcpip_create¥ src¥API_Lib¥Wireless_Upgrade フォルダ内＞

※機能として存在しておりますが、基本的な無線通信を行う場合においては特に使用致しません。

cbdata　…　Wireless bootloader ファイル
cbinst1　…　Wireless bootloader ファイル
cbinst2　…　Wireless bootloader ファイル
cfdata　…　Wireless upgrade FW ファイル
cfinst1　…　Wireless upgrade FW ファイル
cfinst2　…　Wireless upgrade FW ファイル
cudata　…　Wireless upgrade FW ファイル
cuinst1　…　Wireless upgrade FW ファイル
cuinst2　…　Wireless upgrade FW ファイル
DeviceConfigGUI-v3.3.jar　…　Wireless upgrade PC ソフト
wlan_file.rps　…　Wireless upgrade PC ソフト用ファイル

＜¥sample¥adhoc¥ap_rx63n_0a_tcpip_create¥ap_rx63n_0a_tcpip_create¥ src¥API_Lib¥wps フォルダ内＞

このフォルダ内のファイルは、WPS パケット送信処理を行う為のファイル郡です。

※機能として存在しておりますが、基本的な無線通信を行う場合においては特に使用致しません。

＜¥sample¥adhoc¥ap_rx63n_0a_tcpip_create¥ap_rx63n_0a_tcpip_create¥src¥Applications¥MCU フォルダ内＞

rsi_app_util.c	・・・	アプリ作成用ユーティリティ
rsi_config_init.c	・・・	各種ネットワーク設定
rsi_app_util.h	・・・	アプリ作成用ユーティリティヘッダファイル
rsi_config.h	・・・	各種ネットワーク設定定義ヘッダファイル
rsi_global.h	・・・	各種処理上の構造体などの定義ヘッダファイル
Makefile	・・・	メイクファイル
main.c	・・・	参考用の main プログラム

3. TCP/IP 通信サンプルプログラム

3.1 ビルド・デバッグ方法

TCP/IP 通信サンプルプログラムのビルド・デバッグ方法を以下に記します。
アドホックモード（クリエータ・ジョイナー）、インフラストラクチャモードを問わず、ビルド・デバッグの方法は同一です。

（１） ビルド

① HEW を起動し、「**Table 3.1-1 TCP/IP 通信サンプルプログラム HWS 一覧**」からビルド・デバッグを行うサンプルプログラムの HWS ファイルを読み込みます。

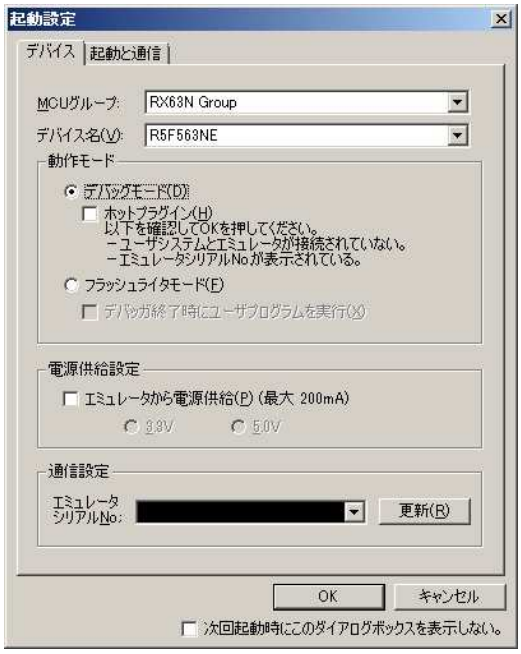
サンプルプログラムの種類	読み込むファイル
アドホックモード　クリエータ	¥sample¥adhoc¥ap_rx63n_0a_tcpip_create¥ap_rx63n_0a_tcpip_create.hws
アドホックモード　ジョイナー	¥sample¥adhoc¥ap_rx63n_0a_tcpip_join¥ap_rx63n_0a_tcpip_join.hws
インフラストラクチャモード	¥sample¥infrastructure¥ap_rx63n_0a_tcp¥ap_rx63n_0a_tcp.hws

Table 3.1-1 TCP/IP 通信サンプルプログラム HWS ファイル一覧

② 最初の読み込みを行ったときに、「このワークスペースのディレクトリが移動されました。」という内容の確認メッセージが表示されますので「はい」を選択して下さい。

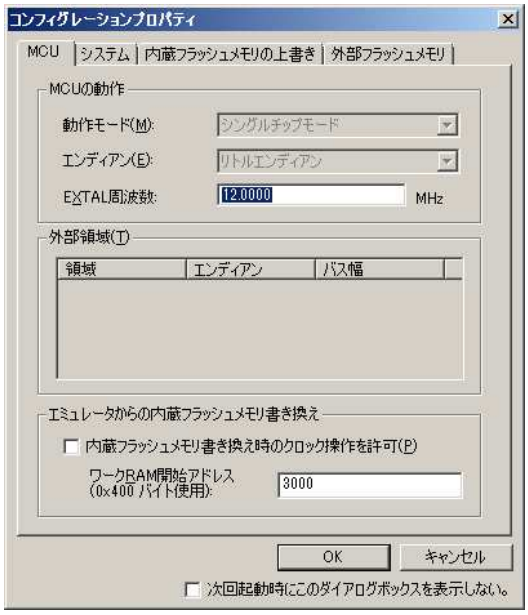
③ 最初の読み込みを行ったときに、コンパイラバージョンによって、バージョンの選択を行うダイアログが表示されることがあります。表示された場合には、使用するコンパイラバージョンを選択して下さい。

④ E1 エミュレータを接続していると「起動設定」画面が表示されます。
デバッグ時は以下の画像を参考に設定して下さい。
（「起動と通信」タブの設定はデフォルトで良いです。）



⑤　「異なる MCU 用ファームウェアがダウンロードされています。更新しますか？」というメッセージが表示される場合があります。その場合「ＯＫ」を選択して下さい。

⑥　次に、「コンフィグレーションプロパティ」画面が表示されます。
デバッグ時は以下の画像を参考に設定して下さい。
（その他「システム」「内蔵フラッシュメモリの上書き」「外部フラッシュメモリ」タブの設定はデフォルトで良いです。）



⑦　[ビルド]ボタン横のリストボックスから、[Debug]または[Release]を選択します。
[Debug]を選択した場合、¥Debug ワークフォルダ内にデバッグ動作用のオブジェクトが生成されます。
[Release]を選択した場合、¥Release ワークフォルダ内に ROM 化動作用のオブジェクトが生成されます。

⑧　メニューの［ビルド］－［ビルド］を実行して下さい。モトローラファイル（拡張子が mot のファイル）、アブソリュートファイル（拡張子が abs のファイル）が出力されます。このとき、マップファイルはワークフォルダに作成されます。

HEW 及び E1 エミュレータの詳細な使用方法につきましては、マニュアルを参照して下さい。

（２）E1 エミュレータでのデバッグ

①　（１）ビルドの①から⑥に従って CPU ボードと E1 エミュレータを接続して下さい。

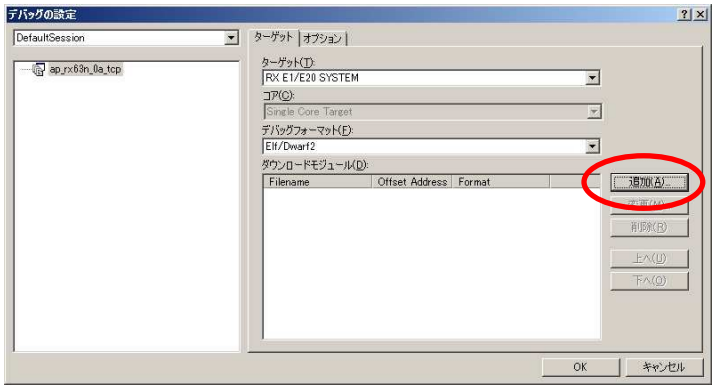
②　メニュー［デバッグ］－［デバッグの設定］を実行して下さい。

以下の画面の様に設定を行います。

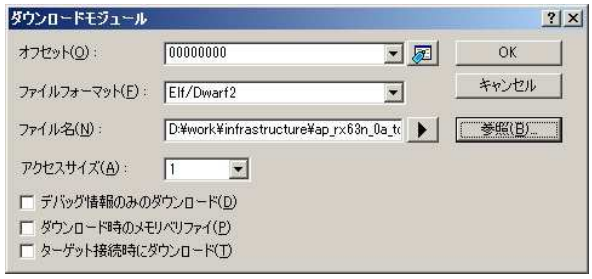
・「ターゲット：RX E1/E20 SYSTEM」

・「デバッグフォーマット：Elf/Dwarf2」

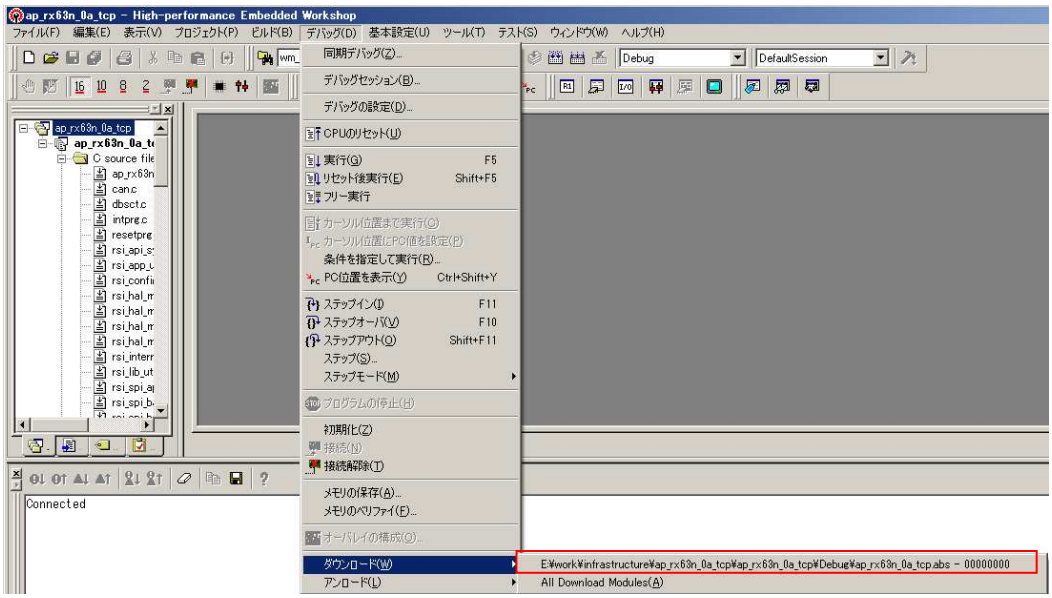
画面上の「追加」をクリックします。



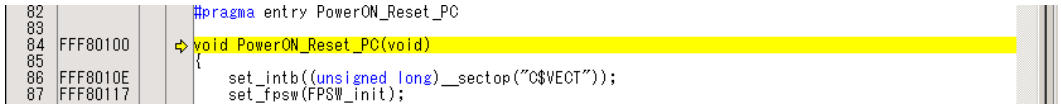
下記画面が表示されますので、「ファイル名（N）」－「参照（B）」より「ap_rx63n_0a_tcp.abs」ファイルを選択します。



③　メニュー［デバッグ］－［ダウンロード］より②で指定した、「ap_rx63n_0a_tcp.abs」ファイルを実行して下さい。



④　以下の画像の様に「PowerON_Reset_PC」関数からプログラムが開始出来る状態となります。

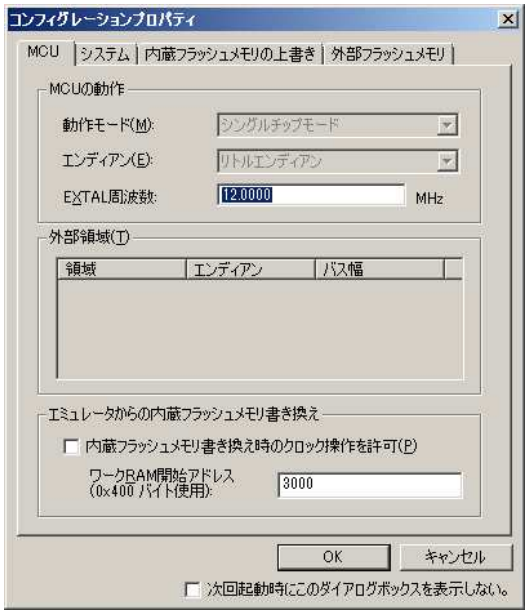


（３）ROM 化動作

①　（１）ビルドの①から③に従って下さい。

②　E1 エミュレータを接続していると「起動設定」画面が表示されます。
ROM 化動作時は以下の画像を参考に設定して下さい。
（「起動と通信」タブの設定はデフォルトで良いです。）

③　次に、「コンフィグレーションプロパティ」画面が表示されます。
ROM 化時は以下の画像を参考に設定して下さい。
（その他「システム」「内蔵フラッシュメモリの上書き」「外部フラッシュメモリ」タブの設定はデフォルトで良いです。）



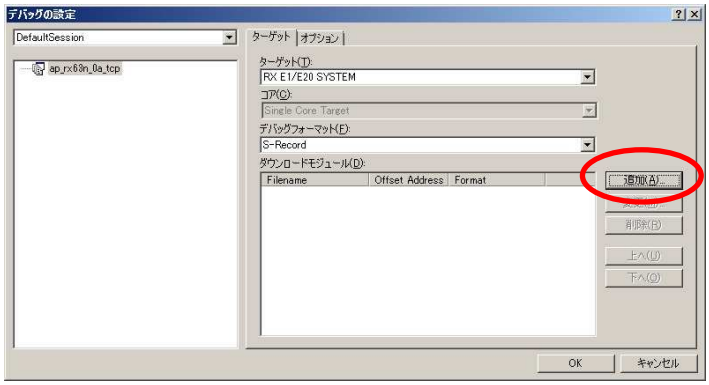
④　メニュー［デバッグ］－［デバッグの設定］を実行して下さい。

以下の画面の様に設定を行います。

・「ターゲット：RX E1/E20 SYSTEM」

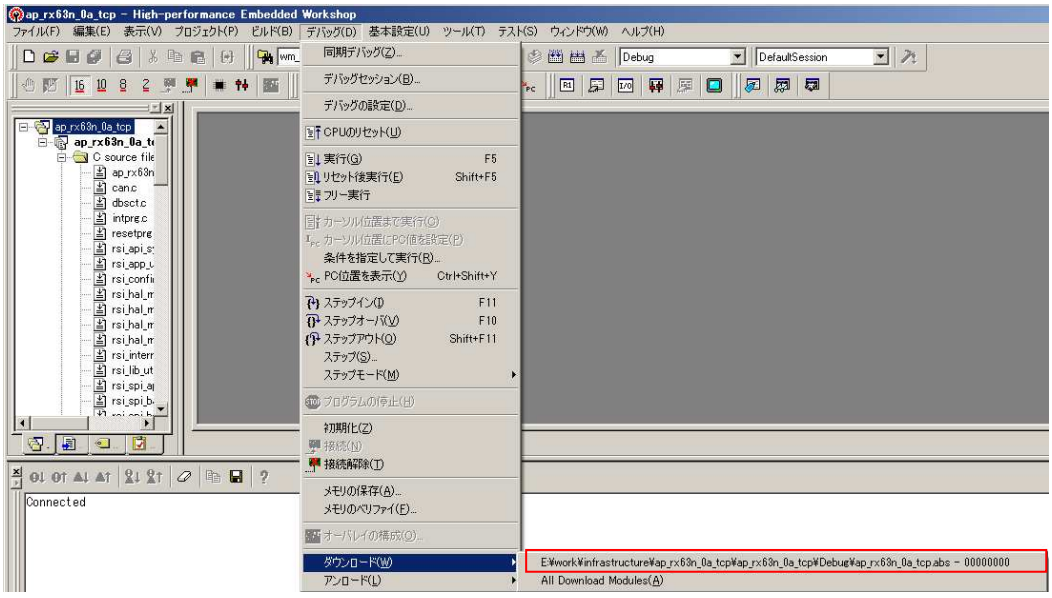
・「デバッグフォーマット：S-Record」

画面上の「追加」をクリックします。

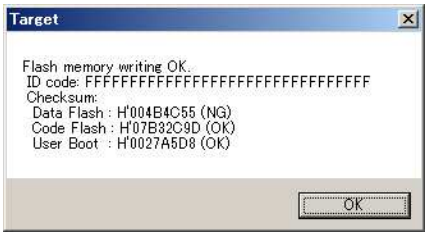


下記画面が表示されますので、「ファイル名（N）」－「参照（B）」より「ap_rx63n_0a_tcp.mot」ファイルを選択します。

⑤　メニュー［デバッグ］－［ダウンロード］より②で指定した、「ap_rx63n_0a_tcp.mot」ファイルを実行して下さい。



⑥　ダウンロードが完了すると以下の画面が表示されます。メッセージに従って ROM 化動作を実行して下さい。



3.2　動作説明（TCP/IP 通信）

3.2.1　サンプルプログラム概要（TCP/IP 通信　アドホックモード）

TCP/IP 通信サンプルプログラム（アドホックモード）は、下記の動作を行います。

1）クリエータ

- SPI 接続された WM-RP-0xS に対してコマンドを送信し、TCP/IP エコーバックサーバを構築します。
  その後、受信したデータをそのまま送信元に送信します。
  動作確認は、ホスト PC 上のターミナルソフト（ハイパーターミナルなど）を使用して行ってください。
  ※　TCP/IP エコーバックサーバ動作の詳細は、「3.2.3　TCP/IP 通信エコーバックサーバ動作」を参照してください。

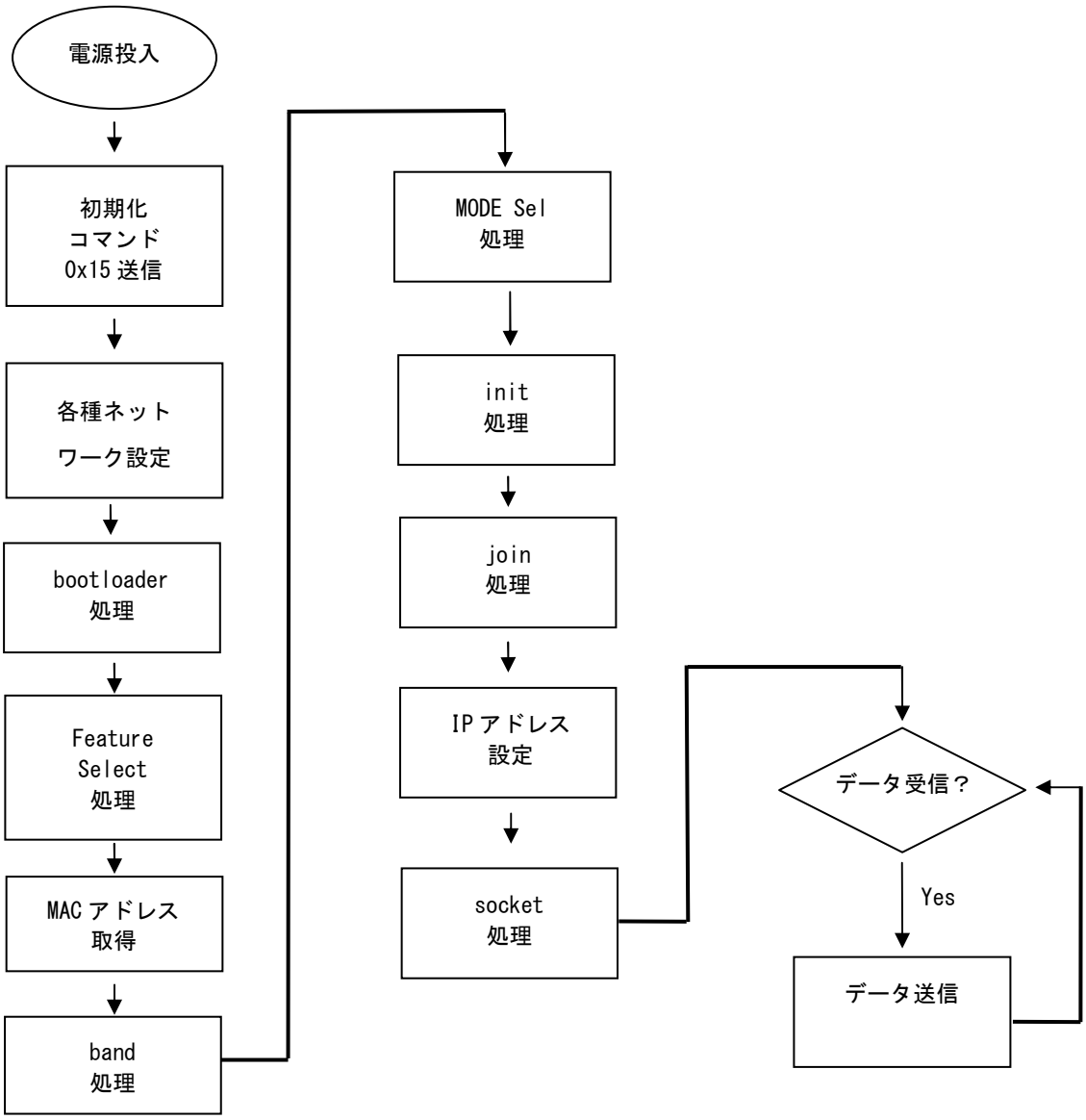


Fig 3.2-1　TCP/IP 通信サンプルプログラム　アドホッククリエイト　WM-RP-0xS 制御フロー

2）ジョイナー　※ジョイナー処理は、「scan」処理が加わります。

- SPI 接続された WM-RP-0xS に対してコマンドを送信し、TCP/IP エコーバックサーバを構築します。
  その後、受信したデータをそのまま送信元に送信します。
  動作確認は、ホスト PC 上のターミナルソフト（ハイパーターミナルなど）を使用して行ってください。
  ※　TCP/IP エコーバックサーバ動作の詳細は、「3.2.3　TCP/IP 通信エコーバックサーバ動作」を参照してください。

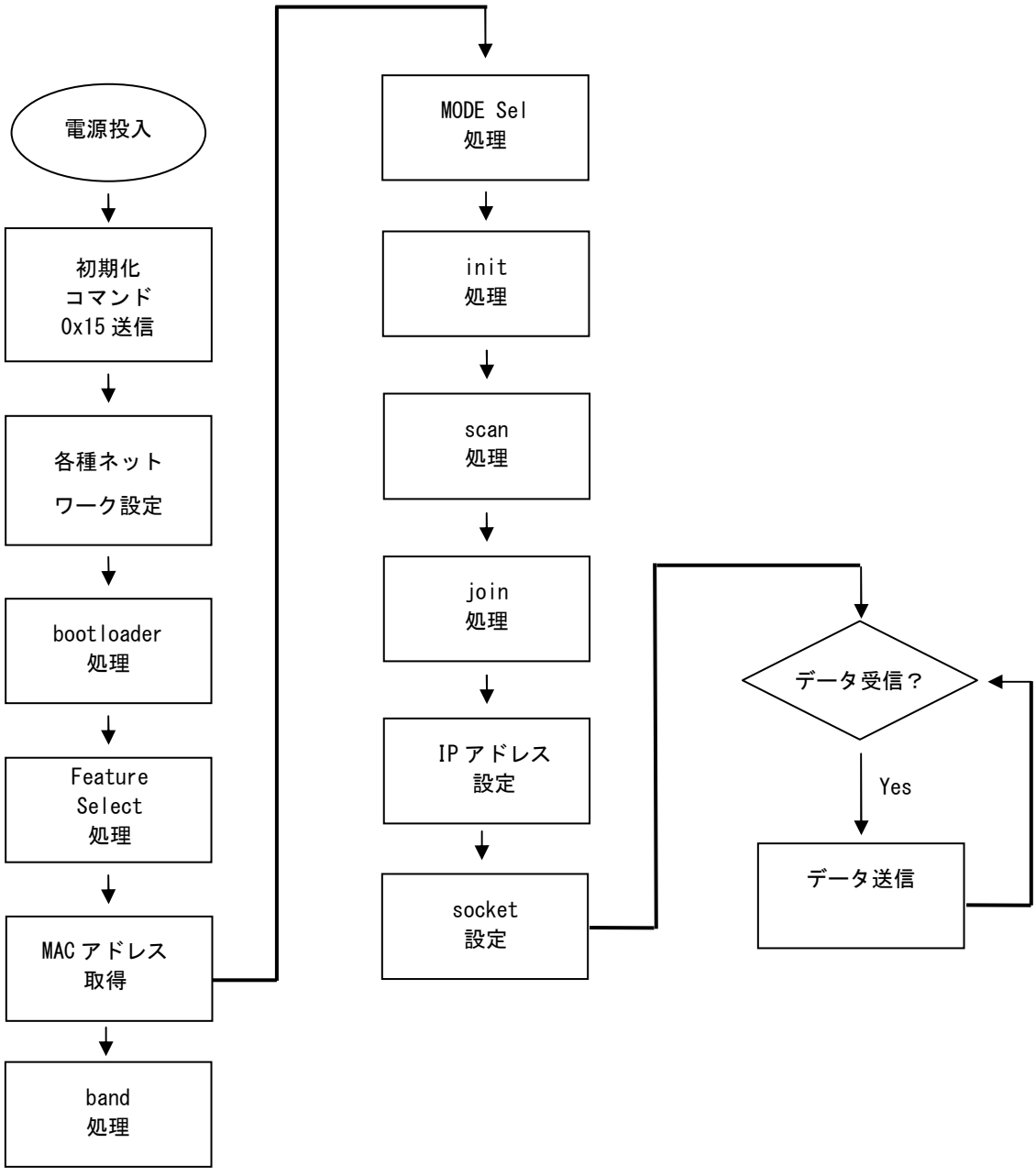


Fig 3.2-2　TCP/IP 通信サンプルプログラム　アドホックジョイナー　WM-RP-0xS 制御フロー

3.2.2　サンプルプログラム概要（TCP/IP 通信　インフラストラクチャモード）

TCP/IP 通信サンプルプログラム（インフラストラクチャモード）は、下記の動作を行います。

- SPI 接続された WM-RP-0xS に対してコマンドを送信し、インフラストラクチャモードで
  アクセスポイントに接続した後、TCP/IP エコーバックサーバを構築します。
  その後、受信したデータをそのまま送信元に送信します。
  動作確認は、ホスト PC 上のターミナルソフト（ハイパーターミナルなど）を使用して行ってください。
  ※　TCP/IP エコーバックサーバ動作の詳細は、「3.2.3　TCP/IP 通信エコーバックサーバ動作」を
  　参照してください。

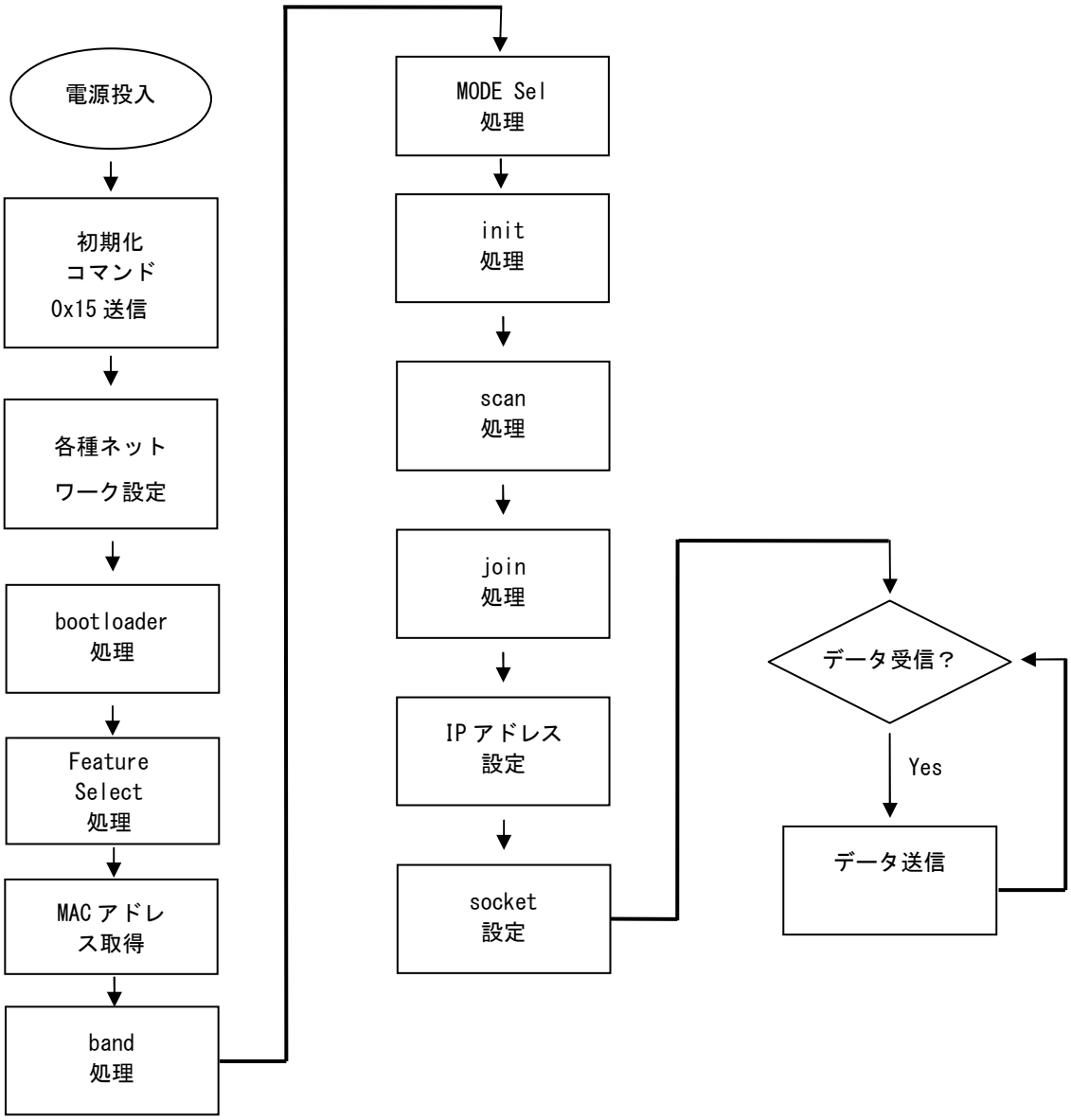


Fig 3.2-3　TCP/IP 通信サンプルプログラム　インフラストラクチャ　WM-RP-0xS 制御フロー

3.2.3　TCP/IP 通信エコーバックサーバ動作

（1）　TCP/IP ネットワーク設定

以下にサンプルプログラムのネットワーク設定を記します。

TCP/IP ネットワーク設定（アドホックモード）	
使用帯域	2.4GHz
使用チャンネル	3ch
ネットワーク接続	アドホックモード
送信レート	自動設定
送信レベル	ハイレベル
PSK	-
アクセスポイント SSID	WM-RP_RX63NSample
IP アドレス	192.168.1.200
サブネットマスク	255.255.255.0
ゲートウェイ	192.168.1.253
使用するポート	8001

Table3.2-1　TCP/IP ネットワーク設定（アドホックモード）

TCP/IP ネットワーク設定（インフラストラクチャモード）	
使用帯域	2.4GHz
使用チャンネル	3ch
ネットワーク接続	インフラストラクチャモード
送信レート	自動設定
送信レベル	ハイレベル
PSK	WM-RP_RX63NSamplePSK
アクセスポイント	WM-RP_RX63NSample
IP アドレス	192.168.1.200
サブネットマスク	255.255.255.0
ゲートウェイ	192.168.1.253
使用するポート	8001

Table3.2-2　TCP/IP ネットワーク設定（インフラストラクチャモード）

※　これらの設定は弊社の環境において設定した値となっています。ご利用時は、お使いの環境の
ネットワーク管理者に問い合わせ、ご利用になられる環境に沿ったそれぞれ適切な値を設定しビルドしてください。

（２）　TCP/IP 通信エコーバックサーバ動作（アドホックモード　クリエータ）

以下の手順に従い、TCP/IP 通信エコーバックサーバの動作を確認してください。

① CPU ボードに電源を投入し、サンプルプログラムを動作させます。AP-RX63N-0A 上の LD2 が点滅します。
② アドホック通信機器の設定を行います。
その際、使用する設定は「Table3.2-1　TCP/IP ネットワーク設定（アドホックモード）」で設定した値となります。
③ アドホック機器を無線 LAN ネットワークに接続し、エコーバックが行われることを確認してください。
④ 以上で TCP/IP 通信エコーバックサーバ動作（アドホックモード：クリエータ）の動作確認は終了です。

（３）　TCP/IP 通信エコーバックサーバ動作（アドホックモード　ジョイナー）

以下の手順に従い、TCP/IP 通信エコーバックサーバの動作を確認してください。

① アドホック通信機器の設定を行い、無線 LAN ネットワークに接続します。
その際、使用する設定は「Table3.2-1　TCP/IP ネットワーク設定（アドホックモード）」で設定した値となります。
② CPU ボードに電源を投入し、サンプルプログラムを動作させます。AP-RX63N-0A 上の LD2 が点滅します。
③ アドホック通信機器の側からエコーバックが行われることを確認してください。
④ 以上で TCP/IP 通信エコーバックサーバ動作（アドホックモード　クリエータ）の動作は終了です。

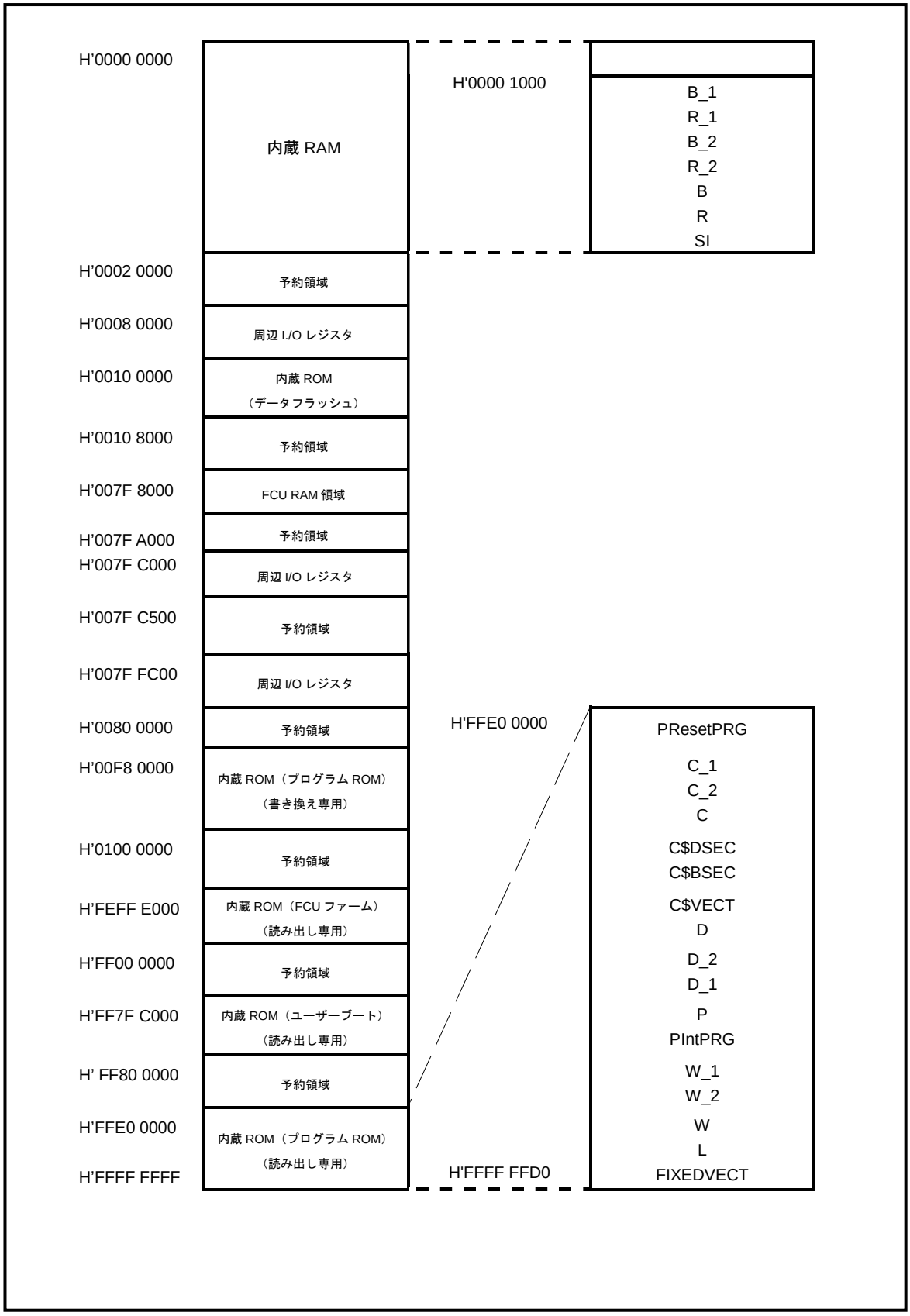
（４）　TCP/IP 通信エコーバックサーバ動作（インフラストラクチャモード）

以下の手順に従い、TCP/IP 通信エコーバックサーバの動作を確認してください。

① CPU ボードに電源を投入し、サンプルプログラムを動作させます。AP-RX63N-0A 上の LD2 が点滅します。
② ホスト PC 上でターミナルソフト（ハイパーターミナルなど）を起動し、Ethernet 接続の設定を行います。
その際、使用する設定は「Table3.2-2　TCP/IP ネットワーク設定（インフラストラクチャモード）」で設定した値となります。
③ ターミナルソフトを使用し、エコーバックが行われることを確認してください。
④ 以上で TCP/IP 通信エコーバックサーバ動作（インフラストラクチャモード）の動作確認は終了です。

3.3　メモリマップ（TCP/IP 通信サンプルプログラム共通）

メモリマップを以下に示します。シングルチップモード



4. UDP 通信サンプルプログラム

4.1　ビルド・デバッグ方法（UDP 通信サンプルプログラム）

UDP 通信サンプルプログラムのビルド・デバッグ方法を以下に記します。
アドホックモード、インフラストラクチャモードを問わず、ビルド・デバッグの方法は同一です。

（１）　ビルド

① HEW を起動し、「**Table 4.1-1　TCP/IP 通信サンプルプログラム HWS ファイル一覧**」からビルド・デバッグを行うサンプルプログラムの HWS ファイルを読み込みます。

サンプルプログラムの種類	読み込むファイル
アドホックモード　クリエータ	¥sample¥adhoc¥ap_rx63n_0a_udpip_create¥ap_rx63n_0a_udpip_create.hws
アドホックモード　ジョイナー	¥sample¥adhoc¥ap_rx63n_0a_udpip_join¥ap_rx63n_0a_udpip_join.hws
インフラストラクチャモード	¥sample¥infrastructure¥ap_rx63n_0a_udp¥ap_rx63n_0a_udp.hws

Table 4.1-1　UDP 通信サンプルプログラム HWS ファイル一覧

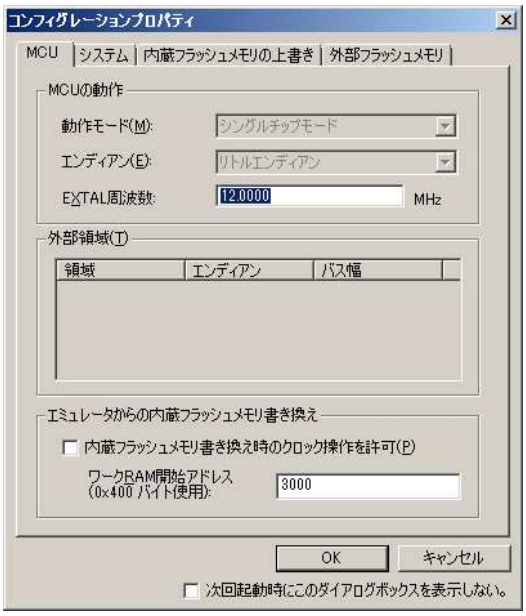
② 最初の読み込みを行ったときに、「このワークスペースのディレクトリが移動されました。」という内容の確認メッセージが表示されますので「はい」を選択して下さい。

③ 最初の読み込みを行ったときに、コンパイラバージョンによって、バージョンの選択を行うダイアログが表示されることがあります。表示された場合には、使用するコンパイラバージョンを選択して下さい。

④ E1 エミュレータを接続していると「起動設定」画面が表示されます。
デバッグ時は以下の画像を参考に設定して下さい。
（「起動と通信」タブの設定はデフォルトで良いです。）

⑤　「異なる MCU 用ファームウェアがダウンロードされています。更新しますか？」というメッセージが表示される場合があります。その場合「ＯＫ」を選択して下さい。

⑥　次に、「コンフィグレーションプロパティ」画面が表示されます。
デバッグ時は以下の画像を参考に設定して下さい。
（その他「システム」「内蔵フラッシュメモリの上書き」「外部フラッシュメモリ」タブの設定はデフォルトで良いです。）



⑦　[ビルド]ボタン横のリストボックスから、[Debug]または[Release]を選択します。
[Debug]を選択した場合、¥Debug ワークフォルダ内にデバッグ動作用のオブジェクトが生成されます。
[Release]を選択した場合、¥Release ワークフォルダ内に ROM 化動作用のオブジェクトが生成されます。

⑧　メニューの［ビルド］－［ビルド］を実行して下さい。モトローラファイル（拡張子が mot のファイル）、アブソリュートファイル（拡張子が abs のファイル）が出力されます。このとき、マップファイルはワークフォルダに作成されます。

HEW 及び E1 エミュレータの詳細な使用方法につきましては、マニュアルを参照して下さい。

（２）E1 エミュレータでのデバッグ

①　（１）ビルドの①から⑥に従って CPU ボードと E1 エミュレータを接続して下さい。

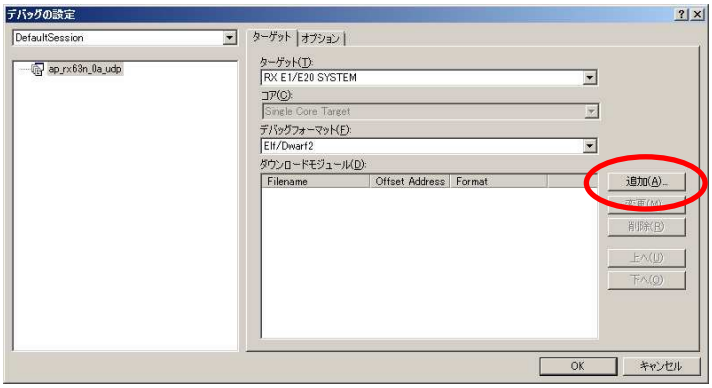
②　メニュー［デバッグ］－［デバッグの設定］を実行して下さい。

以下の画面の様に設定を行います。

・「ターゲット：RX E1/E20 SYSTEM」

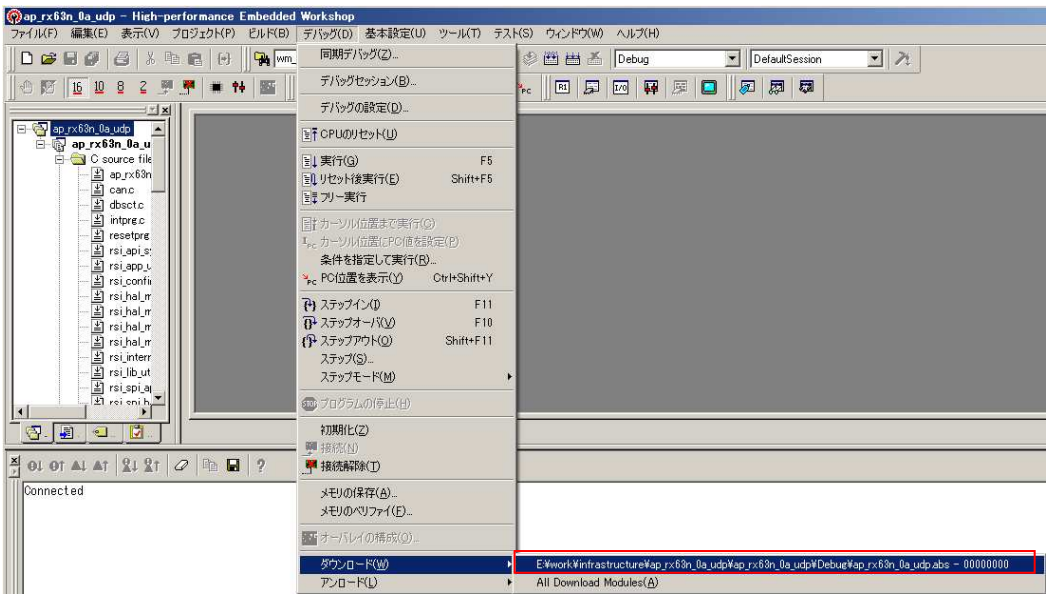
・「デバッグフォーマット：Elf/Dwarf2」

画面上の「追加」をクリックします。

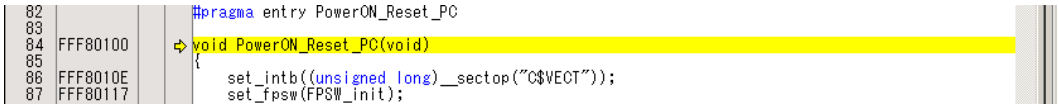


下記画面が表示されますので、「ファイル名（N）」－「参照（B）」より「ap_rx63n_0a_udp.abs」ファイルを選択します。

③　メニュー［デバッグ］－［ダウンロード］より②で指定した、「ap_rx63n_0a_udp.abs」ファイルを実行して下さい。



④　以下の画像の様に「PowerON_Reset_PC」関数からプログラムが開始出来る状態となります。



（３）ROM 化動作

①　（１）ビルドの①から③に従って下さい。

②　E1 エミュレータを接続していると「起動設定」画面が表示されます。

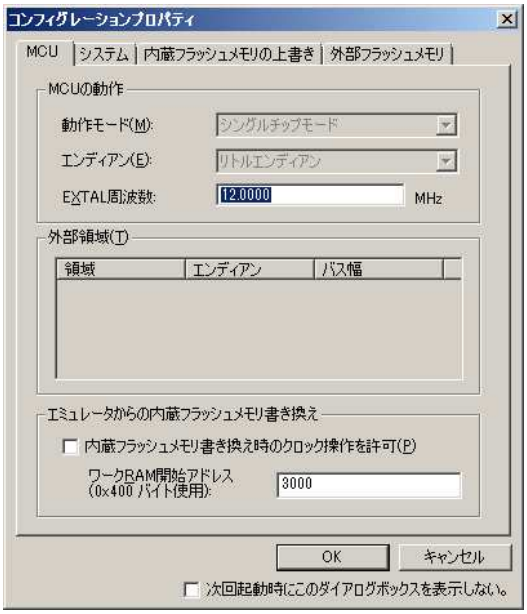
ROM 化動作時は以下の画像を参考に設定して下さい。

（「起動と通信」タブの設定はデフォルトで良いです。）

③　次に、「コンフィグレーションプロパティ」画面が表示されます。

ROM 化時は以下の画像を参考に設定して下さい。

（その他「システム」「内蔵フラッシュメモリの上書き」「外部フラッシュメモリ」タブの設定はデフォルトで良いです。）



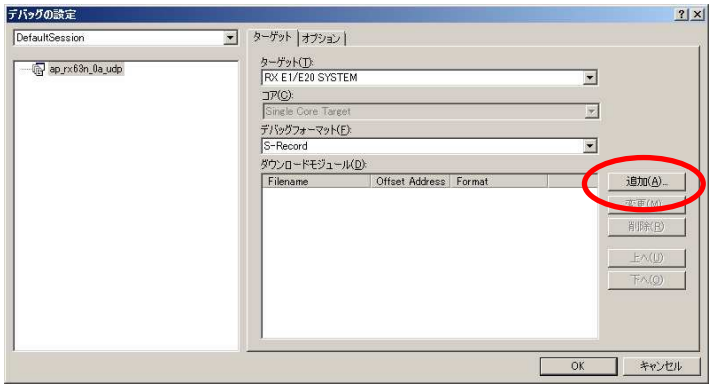
④　メニュー［デバッグ］－［デバッグの設定］を実行して下さい。

以下の画面の様に設定を行います。

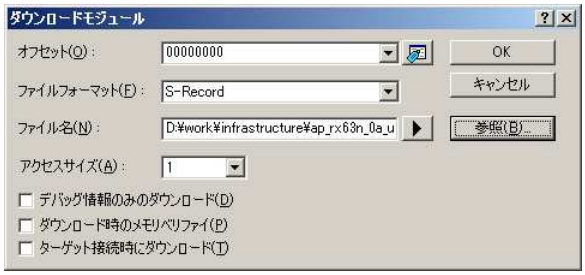
・「ターゲット：RX E1/E20 SYSTEM」

・「デバッグフォーマット：S-Record」

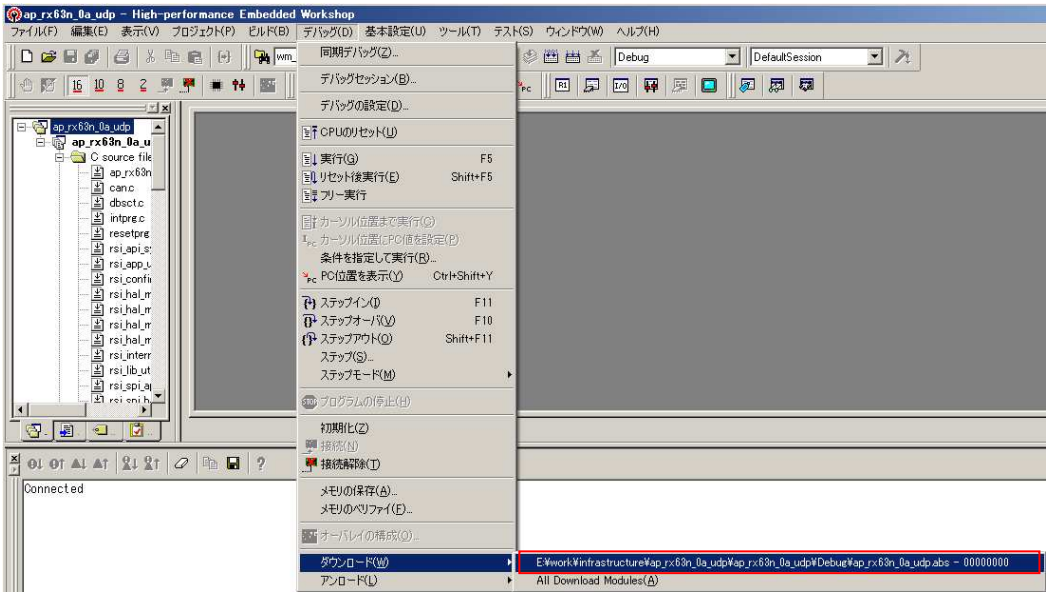
画面上の「追加」をクリックします。



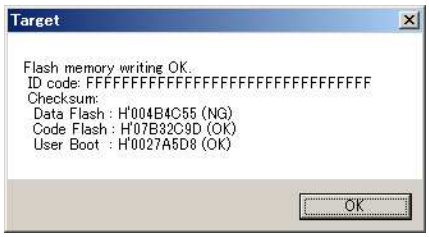
下記画面が表示されますので、「ファイル名（N）」－「参照（B）」より「ap_rx63n_0a_udp.mot」ファイルを選択します。



⑤　メニュー［デバッグ］－［ダウンロード］より②で指定した、「ap_rx63n_0a_udp.mot」ファイルを実行して下さい。



⑥　ダウンロードが完了すると以下の画面が表示されます。メッセージに従って ROM 化動作を実行して下さい。



4.2　動作説明（UDP 通信）

4.2.1　サンプルプログラム概要（UDP 通信　アドホックモード）

UDP 通信サンプルプログラム（アドホックモード）は、下記の動作を行います。

1）　クリエータ

- SPI 接続された WM-RP-0xS に対してコマンドを送信し、UDP ポートを開放します。その後、UDP 通信で受信したデータをそのまま UDP 通信で送信元に送信します。
  動作確認は、ホスト PC 上のターミナルソフト（ハイパーターミナルなど）を使用して行ってください。
  ※　UDP エコーバックサーバ動作の詳細は、「4.2.3　UDP 通信エコーバックサーバ動作」を参照してください。

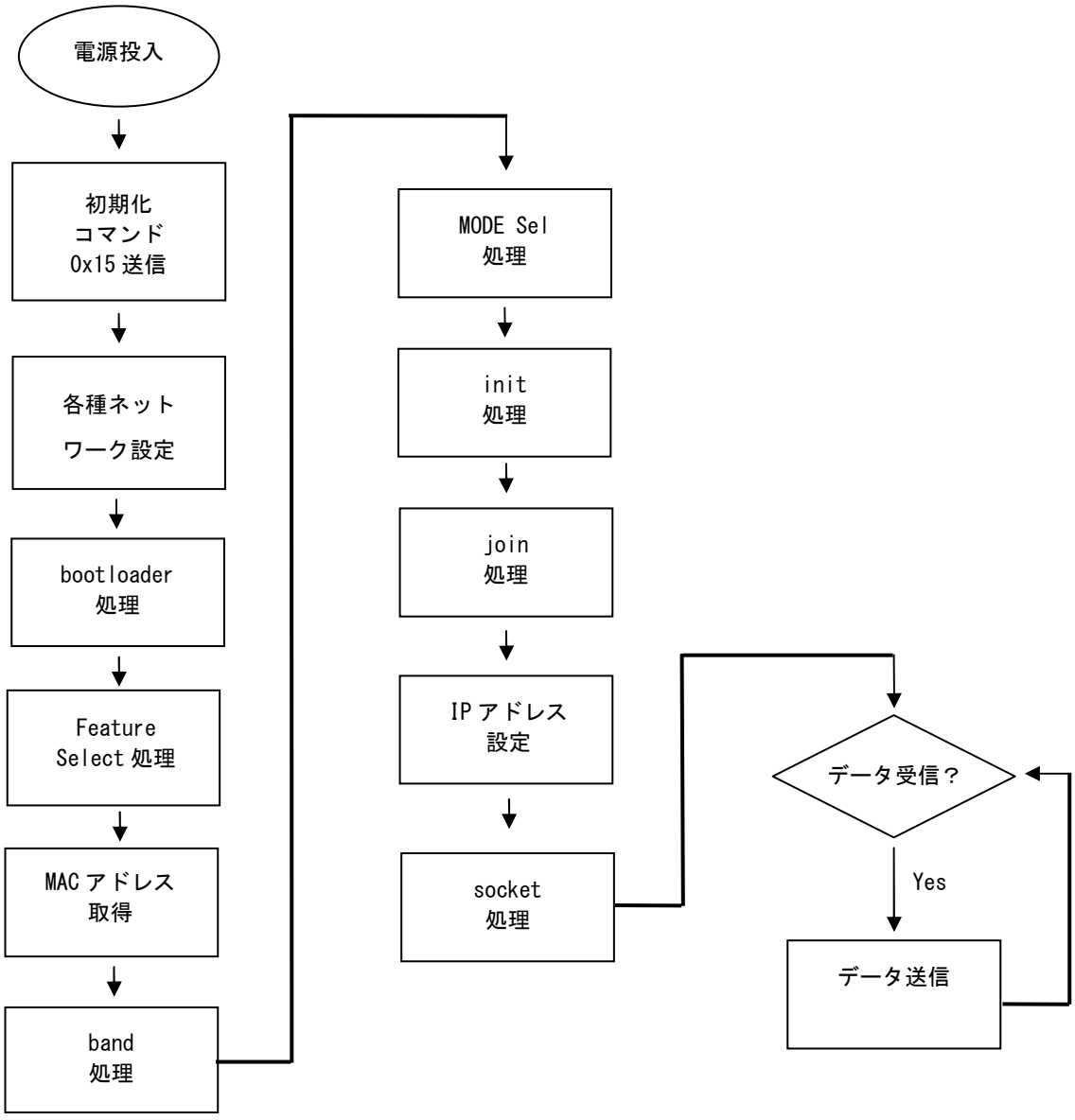


Fig 4.2-1　UDP 通信サンプルプログラム　アドホッククリエイト　WM-RP-0xS 制御フロー

2）ジョイナー　※ジョイナー処理は、「scan」処理が加わります。

- SPI 接続された WM-RP-0xS に対してコマンドを送信し、アドホックモードでアクセスポイントに接続した後、UDP 通信で受信したデータをそのまま UDP 通信で送信元に送信します。
  動作確認は、ホスト PC 上のターミナルソフト（ハイパーターミナルなど）を使用して行ってください。
  ※　UDP エコーバックサーバ動作の詳細は、「4.2.3　UDP 通信エコーバックサーバ動作」を参照してください。

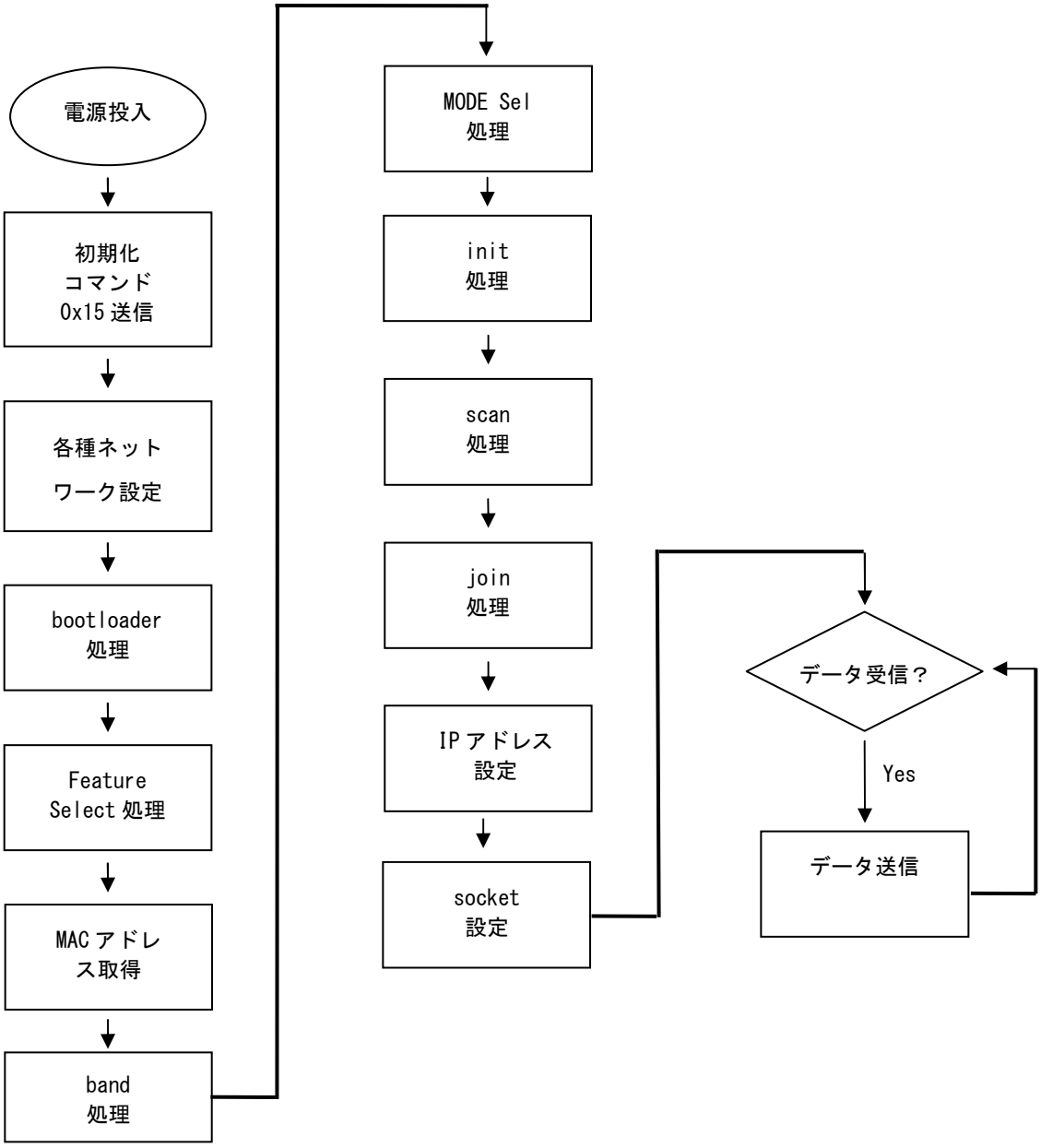


Fig 4.2-2　UDP 通信サンプルプログラム　アドホックジョイン　WM-RP-0xS 制御フロー

4.2.2　サンプルプログラム概要（UDP 通信　インフラストラクチャモード）

TCP/IP 通信サンプルプログラム（インフラストラクチャモード）は、下記の動作を行います。

- SPI 接続された WM-RP-0xS に対してコマンドを送信し、UDP エコーバックサーバを構築します。
  UDP 通信で受信したデータをそのまま UDP 通信で送信元に送信します。
  動作確認は、ホスト PC 上のターミナルソフト（ハイパーターミナルなど）を使用して行ってください。
  ※　UDP エコーバックサーバ動作の詳細は、「4.2.3　UDP 通信エコーバックサーバ動作」を
  　参照してください。

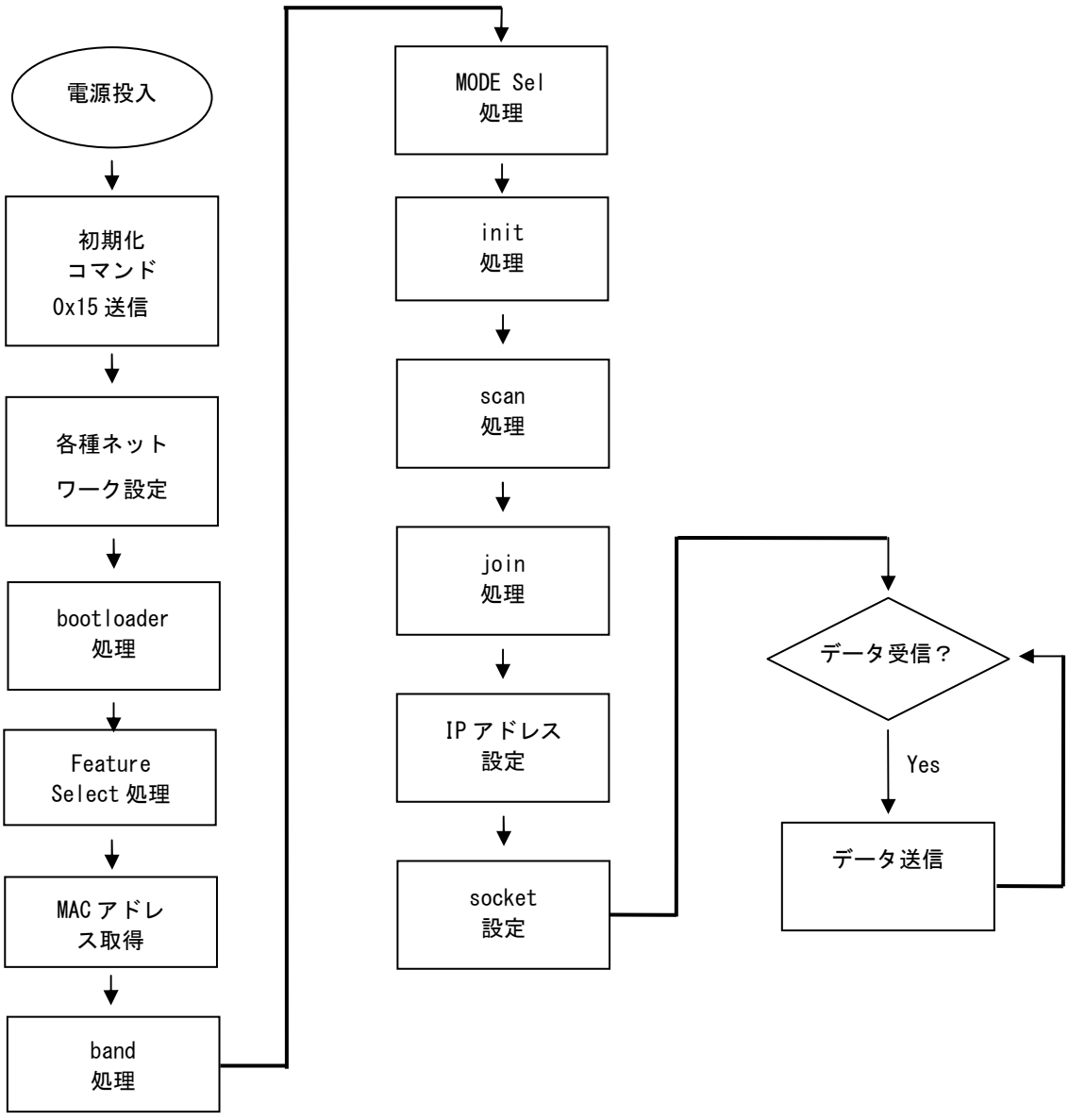


Fig 4.2-3　UDP 通信サンプルプログラム　インフラストラクチャ　WM-RP-0xS 制御フロー

4.2.3　UDP 通信エコーバックサーバ動作

（1）　ネットワーク設定

以下に UDP 通信エコーバックサーバのネットワーク設定を記します。

UDP 通信サンプルプログラムネットワーク設定 （アドホックモード）	
使用帯域	2.4GHz
使用チャンネル	3ch
ネットワーク接続	アドホックモード
送信レート	自動設定
送信レベル	ハイレベル
PSK	-
アクセスポイント	WM-RP_RX63NSample
IP アドレス	192.168.1.200
サブネットマスク	255.255.255.0
ゲートウェイ	192.168.1.253
使用ポート	8001

Table4.2-1　UDP ネットワーク設定（アドホックモード）

UDP 通信サンプルプログラムネットワーク設定 （インフラストラクチャモード）	
使用帯域	2.4GHz
使用チャンネル	3ch
ネットワーク接続	インフラストラクチャモード
送信レート	自動設定
送信レベル	ハイレベル
PSK	WM-RP_RX63NSamplePSK
アクセスポイント	WM-RP_RX63NSample
IP アドレス	192.168.1.200
サブネットマスク	255.255.255.0
ゲートウェイ	192.168.1.253
使用ポート	8000

Table4.2-2　UDP ネットワーク設定（インフラストラクチャモード）

※　これらの設定は弊社の環境において設定した値となっています。ご利用時は、お使いの環境の
ネットワーク管理者に問い合わせ、ご利用になられる環境に沿ったそれぞれ適切な値を設定してください。

（２）　UDP 通信エコーバックサーバ動作（アドホックモード　クリエータ）

以下の手順に従い、UDP 通信エコーバックサーバの動作を確認してください。

① CPU ボードに電源を投入し、サンプルプログラムを動作させます。AP-RX63N-0A 上の LD2 が点滅します。
② アドホック通信機器の設定を行います。
その際、使用する設定は「Table4.2-1　UDP 通信サンプルプログラムネットワーク設定（アドホックモード）」で設定した値となります。
③ アドホック機器を無線 LAN ネットワークに接続し、エコーバックが行われることを確認してください。
④ 以上で UDP 通信エコーバックサーバ動作（アドホックモード　クリエータ）の動作確認は終了です。

（３）　UDP 通信エコーバックサーバ動作（アドホックモード　ジョイナー）

以下の手順に従い、UDP 通信エコーバックサーバの動作を確認してください。

① アドホック通信機器の設定を行い、無線 LAN ネットワークに接続します。
その際、使用する設定は「Table4.2-1　UDP 通信サンプルプログラムネットワーク設定（アドホックモード）」で設定した値となります。
② CPU ボードに電源を投入し、サンプルプログラムを動作させます。AP-RX63N-0A 上の LD2 が点滅します。
③ アドホック通信機器の側からエコーバックが行われることを確認してください。
④ 以上で UDP 通信エコーバックサーバ動作（アドホックモード　ジョイナー）の動作は終了です。

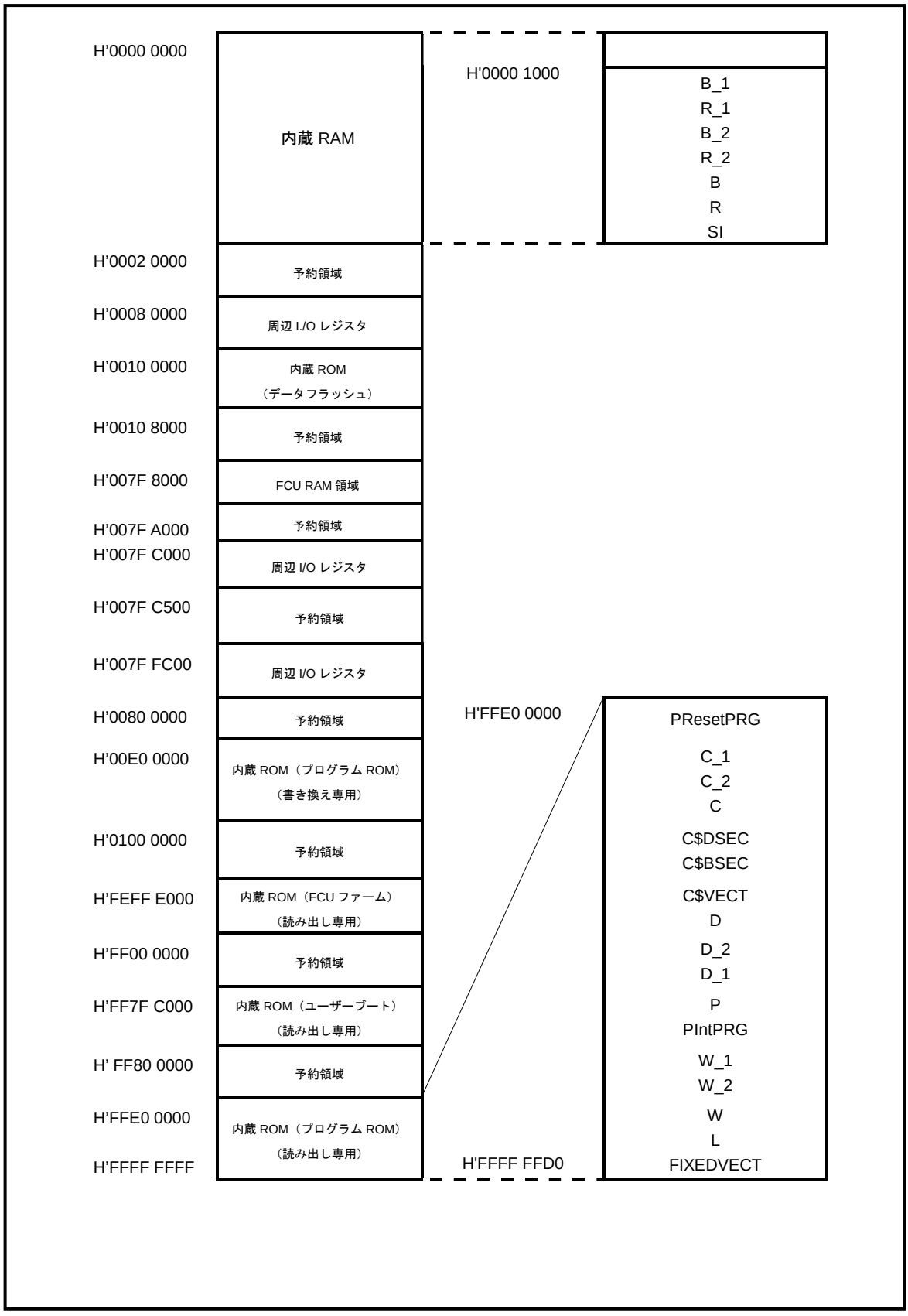
（４）　UDP 通信エコーバックサーバ動作（インフラストラクチャモード）

以下の手順に従い、UDP 通信エコーバックサーバの動作を確認してください。

① CPU ボードに電源を投入し、サンプルプログラムを動作させます。AP-RX63N-0A 上の LD2 が点滅します。
② ホスト PC 上でターミナルソフト（ハイパーターミナルなど）を起動し、Ethernet 接続の設定を行います。
その際、使用する設定は「Table4.2-2　UDP 通信サンプルプログラムネットワーク設定
（インフラストラクチャモード）」で設定した値となります。
③ ターミナルソフトを使用し、エコーバックが行われることを確認してください。
④ 以上で UDP 通信エコーバックサーバ動作（インフラストラクチャモード）の動作確認は終了です。

4.3　メモリマップ（UDP 通信サンプルプロラム共通）

メモリマップを以下に示します。



5. WM-RP-0XS 制御方法

5.1 概要

WM-RP-0xS はホスト CPU とのインタフェースに SPI を採用しています。
ホスト CPU は SPI から各種バイナリコマンドを送信することで WM-RP-0xS の操作を行い、初期化、
ネットワークの設定、データの送受信などを行います。

5.2 SPI

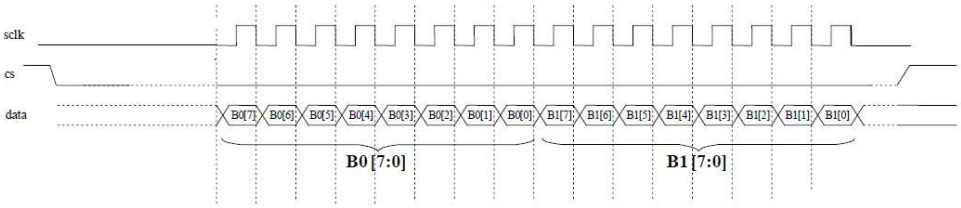
5.2.1 SPI 仕様

WM-RP-0xS の SPI 仕様を以下に記します。

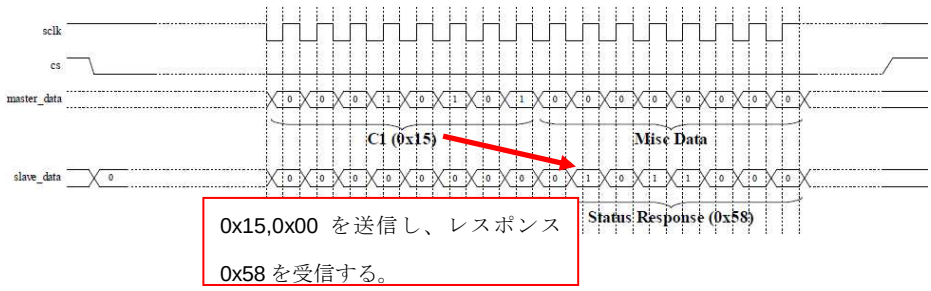
機能	仕様
通信方式	SPI 4 線式 SCK, SCS, MOSI, MISO
SPI クロック	25MHz (MAX)
データ	ホスト CPU が SPI マスター、MSB ファースト
割り込み	INTR 信号 ※ハイレベル割り込みです。 エッジ割り込みでは、ありません。

Table 5.2-1　SPI 仕様

下記画像は、SPI 動作時のタイミングです。ホスト CPU が SPI マスターとなり MSB ファーストにてデータを送受信します。
データは、CLK の立ち上がりで有効です。



下記画像は、WM-RP-0xS の初期化処理時の動作です。ホスト CPU から「0x15」を送信し、そのレスポンスである「0x58」を
WM-RP-0xS から受信する為に「0x00」（画像では、MiscData」）を送信しています。



5.2.2　SPI 通信の基本的な流れ

WM-RP-0xS とホスト CPU との SPI 通信は以下の流れで行われます。

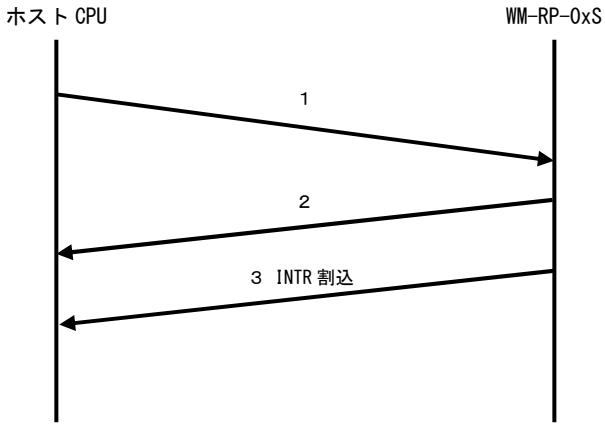


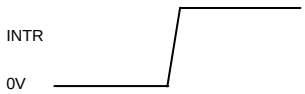
Fig 5.2-1　SPI インタフェース制御フロー

1. WM-RP-0xS へのコマンド入力です。バイナリコマンドを送信することで、ホスト CPU から WM-RP-0xS を制御することができます。
2. WM-RP-0xS からのレスポンス（0x58、0x55 など）です。バイナリデータにてホスト CPU に応答が返されます。
3. その後、コマンドによって INTR 割り込みが入りレスポンス受信を行います。

※　送信するコマンドおよびレスポンスの詳細に関しましては、
「RS9110-N-11-22_24_28-Software_PRM.pdf」を参照してください。

5.2.3　INTR 割り込み信号

割り込みは、WM-RP-0xS からのハイレベル割り込みとなります。※エッジ割り込みではありません。
ただし、AP-RX63N-0A では、WM-RP-0xS からの割り込みレベルを反転させる回路が組み込まれている為、
CPU（RX63N）から見る割り込みレベルはローレベルとなります。



WM-RP-0xS が以下の状態の時、INTR 割り込みが発生します。

- 無線通信の相手より、データを受信した。
- WM-RP-0xS へのコマンド処理による、レスポンスデータがある場合。
- WM-RP-0xS 内部 SPI 受信メモリバッファがフルとなった。
  ※　SPI 受信用メモリは、1460Byte で 7 個のメモリが用意されています。
- PowerMode1 の起床時

割り込み発生時は、必ず適切な処理を他の処理よりも先に処理して下さい。
例えば、無線データ送信処理中に割り込みが発生した場合は送信処理完了後に割り込みの処理を行って下さい。
割り込み処理が適切に処理されなかった場合、WM-RP-0xS からの応答が 0x54 ビジーレスポンスとなります。
この 0x54 ビジーレスポンスとなった場合、最悪の場合電源を OFF -> ON する必要が生じます。

5.2.4　無線データ送信処理

無線データ送信時は、送信処理を行う場合は、必ず rsi_intHandler()関数を呼び出し rsi_strIntStatus.bufferFull を確認して下さい。

これは、WM-RP モジュール内部バッファが FULL か？を確認しております。

もし、FULL となった場合は FULL が解除されてから送信を行う様にプログラムして下さい。

以下は、rsi_spi_send_data.c ファイルの rsi_send_data()関数（無線データ送信）内 92 行目～97 行目の処理です。

```
        rsi_intHandler();
        if(rsi_strIntStatus.bufferFull == RSI_TRUE)
//      if(rsi_checkBufferFullIrq() == RSI_TRUE)
        {
                return RSI_BUFFER_FULL;
        }
```

5.3　Redpine Signals 社提供のライブラリ

「¥Driver¥Driver_TCP¥API_Lib」内のファイルは WM-RP-0xS を使用するに当たって Redpine Signals 社が用意したライブラリとなります。
サンプルプログラムは、このライブラリを使用しています。詳細は、「¥Driver¥Driver_TCP¥Documentation」内の「RS9110-N-11-22_24_28_SPI_API_Library_Manual.pdf」を参照してください。
また、合わせて「RS9110-N-11-22_24_28-Software_PRM.pdf」も参照してください。

※各 pdf 上で「5GHz」の設定、「14CH」の設定に関して記述がありますが、
WM-RP-0xS は、「2.4GHｚ、1-13CH」にて技術基準適合証明を取得しております。
よって「2.4GHｚ、1-13CH」以外の設定を行って動作させた場合　弊社は一切責任を負いませんのでご了承下さい。

5.3.1　各種変数等

＜１＞　rsi_api 構造体

「¥src¥Applications¥MCU」フォルダ内「rsi_global.h」ファイルの 1011 行目～1030 行目で定義されています。
この構造体は、主に WM-RP-0xS を初期化する時のパラメータを格納する為の構造体となっています。
この構造体へのパラメータ設定は、
「¥src¥Applications¥MC」フォルダ内「rsi_config_init.c」ファイルの「rsi_init_struct()」関数にて行われます。
サンプルプログラムでは、「wm_rp_04s.c」ファイルの 81 行目にて、グローバル変数として「rsi_api rsi_strApi」の様に定義して使用しています。

＜２＞　rsi_uCmdRsp 構造体

「¥src¥Applications¥MCU」フォルダ内「rsi_global.h」ファイルの 952 行目～981 行目で定義されています。
この構造体は、主に WM-RP-0xS からのレスポンスを格納する為の構造体となっています。

サンプルプログラムでは、「wm_rp_04s.c」ファイルの 57 行目にて、static なグローバル変数として「static volatile rsi_uCmdRsp uCmdRspFrame」の様に定義して使用しています。

この構造体ですが、WM-RP-0xS からの何らかのレスポンスを状況に応じて適切な変数に格納する様に作られています。

例）「band」コマンドを実行する場合　以下の様な処理となります。

```
retval = rsi_band( rsi_strApi.band );                          /* band コマンド送信 */
if( retval == RSI_SUCCESS ){
  RSI_RESPONSE_TIMEOUT( RSI_BANDTIMEOUT );                     /* レスポンス待ち */
  rsi_read_packet( &uCmdRspFrame );                            /* レスポンス取得 */
  rsi_clearPktIrq();
  if( uCmdRspFrame.mgmtResponse.rspCode[0] == RSI_RSP_BAND ){  /* band のレスポンスコード=0x97 */
    if( uCmdRspFrame.mgmtResponse.status == 0x00 ){
    } else {
    }
  }
}
```

※詳細は、「RS9110-N-11-22_24_28_SPI_API_Library_Manual.pdf」ファイルの「2.1.8 Read Packet data structure (From module):」を参照して下さい。

＜３＞　タイマ用変数

「wm_rp_04s.c」ファイルの 80 行目～98 行目で以下の 3 つのグローバル変数を定義しています。

- uint32 rsi_spiTimer1
- uint32 rsi_spiTimer2
- uint32 rsi_spiTimer3

この 3 つの変数は、「¥src¥Applications¥MCU」フォルダ内「rsi_global.h」ファイルの 80 行目～98 行目で定義されている以下のマクロに影響します。

- #define RSI_INC_TIMER_2　　rsi_spiTimer2++
- #define RSI_INC_TIMER_1　　rsi_spiTimer1++
- #define RSI_INC_TIMER_3　　rsi_spiTimer3++
- #define RSI_RESET_TIMER1　　rsi_spiTimer1=0
- #define RSI_RESET_TIMER2　　rsi_spiTimer2=0
- #define RSI_RESET_TIMER3　　rsi_spiTimer3=0

これら 6 個のマクロを使用する場合は、「uint32 rsi_spiTimer1」「uint32 rsi_spiTimer2」「uint32 rsi_spiTimer3」変数をサンプルプログラムの様にグローバル変数として定義して下さい。

実際に、「tmr.c」ファイルの「cmt_init()」関数内で、

- RSI_RESET_TIMER1
- RSI_RESET_TIMER2
- RSI_RESET_TIMER3

を使用しており、また「tmr.c」ファイルの「Excep_CMTU0_CMT0 ()」割り込み関数内で

- RSI_INC_TIMER_2
- RSI_INC_TIMER_1
- RSI_INC_TIMER_3

を使用しております。

この変数が使用されるのは、タイムアウト処理が殆どです。
例えば、「¥src¥Applications¥MCU」フォルダ内「rsi_global.h」ファイルの 151 行目の
「#define RSI_RESPONSE_TIMEOUT(A)」マクロでは「RSI_RESET_TIMER3」を使ってタイムアウト処理を行っています。

＜４＞　INTR 割り込み状態格納変数

「wm_rp_04s.c」ファイルの 79 行目にて、グローバル変数として「volatile rsi_intStatus rsi_strIntStatus」を定義しています。
ライブラリでは、INTR 割り込み時のステータスを「rsi_strIntStatus」変数に格納するようになっています。

5.3.2　無線通信用の主なコマンドファイル

※「¥src¥API_Lib」フォルダ内の主なコマンドファイルに関して記述します。

＜１＞　Band コマンド

コマンド説明	
説明	使用周波数帯域の設定を行います。
コマンドファイル名	rsi_spi_band.c
使用方法	引数に設定値を指定する。
パラメータ説明	0:2.4GHz
レスポンスコード	0x97
レスポンス詳細	-

＜２＞　Init コマンド

コマンド説明	
説明	Band コマンド送信後に要求されるコマンドです。
コマンドファイル名	rsi_spi_init.c
使用方法	rsi_band()関数処理後、rsi_init()関数を呼ぶ
パラメータ説明	-
レスポンスコード	0x94
レスポンス詳細	-

＜３＞　Scan コマンド

コマンド説明	
説明	指定したチャンネルを走査します。
コマンドファイル名	rsi_spi_scan.c
使用方法	予め、rsi_api 構造体で定義した変数にスキャンするチャンネルと SSID を設定しておきます。
パラメータ説明	rsi_api 構造体で定義した変数ポインタ指定
レスポンスコード	0x95
レスポンス詳細	指定した SSID からのパラメータが、 rsi_uCmdRsp 構造体で定義した変数の rsi_scanResponse 構造体内に格納されます。

＜４＞　Join コマンド

コマンド説明	
説明	SSID への接続を行います。
コマンドファイル名	rsi_spi_join.c
使用方法	予め、rsi_api 構造体で定義した変数に ・ネットワークタイプ ・セキュリティタイプ ・送信データレート ・送信出力パワー ・セキュリティキー ・接続する SSID ・IBSS モード ・IBSS チャンネル を設定しておきます
パラメータ説明	rsi_api 構造体で定義した変数ポインタ指定
レスポンスコード	0x96
レスポンス詳細	-

＜５＞　IP Config コマンド

コマンド説明	
説明	IP アドレス等を設定します。
コマンドファイル名	rsi_spi_ipparam.c
使用方法	予め、rsi_api 構造体で定義した変数に ・WM-RP-0xS の IP アドレス ・ネットマスク ・ゲートウェイ ・DNS サーバ ・DHCP の使用、未使用 を設定しておきます。
パラメータ説明	rsi_api 構造体で定義した変数ポインタ指定
レスポンスコード	-
レスポンス詳細	-

＜６＞　Socket タイプコマンド

コマンド説明	
説明	Socket タイプを設定します。
コマンドファイル名	rsi_spi_socket.c
使用方法	予め、rsi_api 構造体で定義した変数に ・WM-RP-0xS の PORT 番号 ・接続先の PORT 番号 ・TCP or UDP, サーバ or クライアント ・接続先の IP アドレス を設定しておきます。
パラメータ説明	rsi_api 構造体で定義した変数ポインタ指定
レスポンスコード	0x02
レスポンス詳細	この処理が成功すると、 rsi_uCmdRsp 構造体で定義した変数の rsi_socketFrameRcv 構造体内にソケットディスクリプタが格納されます。 ソケットディスクリプタは、無線データ送受信等において必要になるパラメータですのでグローバル変数等に保持して使用して下さい。

＜７＞　Send data コマンド

コマンド説明	
説明	オープンしたソケットからデータを送信します。
コマンドファイル名	rsi_spi_send_data.c
使用方法	Socket タイプコマンドで取得したソケットディスクリプタを 指定して関数を呼び出します。
パラメータ説明	以下のパラメータを指定します。 ・ソケットディスクリプタ ・送信データバッファ ・送信データサイズ ・TCP or UDP の指定
レスポンスコード	-
レスポンス詳細	-

＜８＞　Receive data コマンド

コマンド説明	
説明	WM-RP-0xS からの受信を行います。
コマンドファイル名	rsi_spi_read_packet.c
使用方法	状況に応じて WM-RP-0xS からの受信を行います。 １、INTR 割り込みが入った時 WM-RP-0xS 内部レジスタからどういった割り込みが入ったかステータスを取得します。 これは、「rsi_spi_interrupt_handler.c」ファイルの「rsi_intHandler()」関数内にて「rsi_irqstatus()」関数を呼ぶ事で行っています。 ２、接続先からの無線データか？ WM-RP-0xS 内部レジスタの内容に応じて無線データか？ 或いは WM-RP-0xS のレスポンスデータか？ に応じて WM-RP-0xS からの受信処理を行います。
パラメータ説明	rsi_uCmdRsp 構造体で定義した変数
レスポンスコード	0x07
レスポンス詳細	この処理が成功すると、 rsi_uCmdRsp 構造体で定義した変数にレスポンスデータ、もしくは無線データが格納されます。

＜９＞　Close socket コマンド

コマンド説明	
説明	オープンしたソケットのクローズを行います。
コマンドファイル名	rsi_spi_socket_close.c
使用方法	rsi_socket_close()関数にクローズするソケットディスクリプタを指定して呼び出します。
パラメータ説明	クローズするソケットディスクリプタ
レスポンスコード	0x06
レスポンス詳細	-

＜１０＞　Disassociate コマンド

コマンド説明	
説明	接続しているアクセスポイントからの切断を行います。
コマンドファイル名	rsi_spi_disconnect.c
使用方法	rsi_disconnect ()関数を呼び出します。
パラメータ説明	-
レスポンスコード	0x0C
レスポンス詳細	-

＜１１＞　Remote close

コマンド説明	
説明	接続先が切断した時の処理です。
コマンドファイル名	-
使用方法	rsi_uCmdRsp 構造体で定義した変数の rsi_recvRemTerm 構造体のレスポンスデータを確認します。
パラメータ説明	-
レスポンスコード	0x05
レスポンス詳細	接続先が切断（ソケットクローズなど）した場合、 INTR 割り込みによって、 rsi_recvRemTerm 構造体に 0x05 が格納されます。 接続先が切断した事を知る為に、常にこの変数を監視するようにプログラムして下さい。

＜１２＞　WM-RP-0xS 初期化用コマンド

コマンド説明	
説明	WM-RP-0xS 初期化コマンドです。 電源 ON 後、必ず最初に 1 度実行して下さい。
コマンドファイル名	rsi_spi_iface_init.c
使用方法	rsi_spi_iface_init()関数を呼び出します。
パラメータ説明	-
レスポンスコード	0x58
レスポンス詳細	

＜１３＞　WM-RP-0xS 各種設定用ファイル

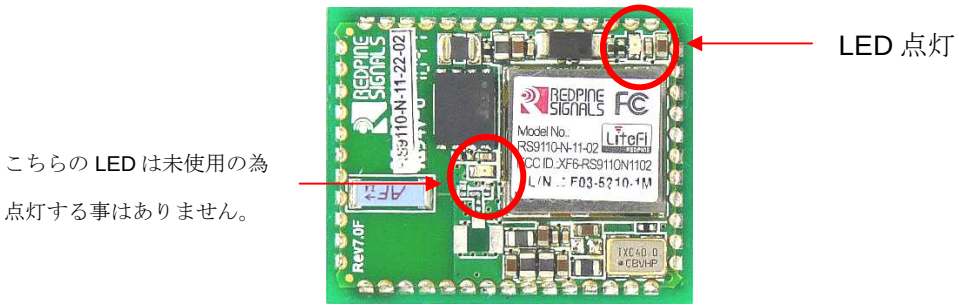
コマンド説明	
説明	WM-RP-0xS に対して実行する各種設定、コマンド等の値を変数に設定します。
コマンドファイル名	rsi_config_init.c（「¥src¥Applications¥MCU」フォルダ内）
使用方法	rsi_init_struct()関数を呼び出します。
パラメータ説明	rsi_api 構造体で定義した変数ポインタを指定します。
レスポンスコード	-
レスポンス詳細	-

＜１４＞　bootloader 処理

コマンド説明	
説明	WM-RP-0xS に対して bootloader 処理を行います。
コマンドファイル名	rsi_bootloader.c
使用方法	＜１２＞　WM-RP-0xS 初期化用コマンドの次に必ず rsi_bootloader()関数を呼び出します。 また、bootloader()関数内では以下のファイルを include して使用しています。 「¥src¥API_Lib¥Firmware」フォルダ内 ● sbinst1 ● sbinst2 ● sbdata1 ● sbdata2 ※rsi_global.h ファイル内の 「#define RSI_LOAD_SBDATA2_FROM_HOST」定義の値を必ず「1」にしておいて下さい。
パラメータ説明	-
レスポンスコード	-
レスポンス詳細	-

6. WM-RP-0XS 上の LED に関して

bootloader 終了後、WM-RP-0xS 上の以下の部分の LED が点灯します。



7. サンプルプログラムに関して

7.1　ネットワークの設定

7.1.1、ポート番号

「¥src¥Applications¥MCU」フォルダ内「rsi_config_init.c」ファイル

37 行目～39 行目にて、ポート番号の設定をしています。

必要に応じて以下の定義の変更をして下さい。

- #define RSI_MODULE_SOCKET_ONE　　WM-RP-0xS のポート番号
- #define RSI_TARGET_SOCKET_ONE　　接続先相手のポート番号

7.1.2、TCP or UDP , Server or Client

「¥src¥Applications¥MCU」フォルダ内「rsi_config_init.c」ファイル

41 行目～42 行目にて、TCP or UDP , Server or Client の設定をしています。

必要に応じて以下の定義の変更をして下さい。

※この定義に指定可能な値は、「¥src¥API_Lib」フォルダ内「rsi_spi_api.h」ファイルの

390 行目～394 行目の「SOCKET Defines」定義を使用して下さい。

- #define RSI_SOCKET_TCP_CLIENT_TYPE　　WM-RP-0xS の TCP or UDP , Server or Client の設定

7.1.3、インフラストラクチャ or アドホック

「¥src¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル

42 行目～44 行目にて、インフラストラクチャ or アドホックの設定をしています。

必要に応じて以下の定義の変更をして下さい。

※この定義に指定可能な値は、「¥src¥API_Lib」フォルダ内「rsi_spi_api.h」ファイルの

420 行目～424 行目の「NETWORK Type」定義を使用して下さい。

- #define RSI_NETWORK_TYPE　　WM-RP-0xS　インフラストラクチャ or アドホック

7.1.4、WM-RP-0xS の IP アドレス

「¥src¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル

54 行目～55 行目にて、WM-RP-0xS の IP アドレスを設定をしています。

必要に応じて以下の定義の変更をして下さい。

- #define　　RSI_MODULE_IP_ADDRESS　　WM-RP-0xS の IP アドレス

7.1.5、ゲートウェイ

「¥src¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル

57 行目～58 行目にて、ゲートウェイのアドレスを設定をしています。

必要に応じて以下の定義の変更をして下さい。

- #define RSI_GATEWAY　　ネットワーク接続　ゲートウェイ

7.1.6、接続先相手の IP アドレス

「¥src¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル

60 行目～61 行目にて、接続先相手の IP アドレスを設定をしています。

必要に応じて以下の定義の変更をして下さい。

- #define RSI_TARGET_IP_ADDRESS　　接続先相手の IP アドレス

7.1.7、ネットマスク

「¥src¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル

63 行目～64 行目にて、ネットマスクのアドレスを設定をしています。

必要に応じて以下の定義の変更をして下さい。

- #define RSI_NETMASK　　ネットワーク接続　ネットマスク

7.1.8、無線アクセスポイントセキュリティタイプ

「¥src¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル

66 行目～67 行目にて、無線アクセスポイントのセキュリティタイプを設定をしています。

必要に応じて以下の定義の変更をして下さい。

※この定義に指定可能な値は、「¥src¥API_Lib」フォルダ内「rsi_spi_api.h」ファイルの

412 行目～417 行目の「SECURITY Type Defines」定義を使用して下さい。

- #define RSI_SECURITY_TYPE　　無線アクセスポイントセキュリティタイプ

7.1.9、無線アクセスポイントセキュリティキー

「¥src¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル

71 行目～76 行目にて、無線アクセスポイントのセキュリティキーを設定をしています。

必要に応じて以下の定義の変更をして下さい。

- #define RSI_PSK　　無線アクセスポイントセキュリティキー

7.1.10、無線アクセスポイントスキャン SSID

「¥src¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル

86 行目～87 行目にて、無線アクセスポイントのスキャン SSID を設定をしています。

必要に応じて以下の定義の変更をして下さい。

- #define RSI_SCAN_SSID　　無線アクセスポイントスキャン SSID

7.1.11、無線アクセスポイントスキャン CH

「¥src¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル

91 行目にて、無線アクセスポイントのスキャン CH を設定をしています。

必要に応じて以下の定義の変更をして下さい。

- #define RSI_SCAN_CHANNEL　　無線アクセスポイントスキャン CH

7.1.12、無線アクセスポイント参加(Join)SSID

「¥src¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル

93 行目～94 行目にて、無線アクセスポイントの参加(Join)SSID を設定をしています。

必要に応じて以下の定義の変更をして下さい。

- #define RSI_JOIN_SSID　　無線アクセスポイント参加(Join)SSID

7.1.13、DHCP 使用 or 未使用

「¥src¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル

135 行目～136 行目にて、DHCP 使用 or 未使用を設定をしています。

必要に応じて以下の定義の変更をして下さい。

※この定義に指定可能な値は、「¥src¥API_Lib」フォルダ内「rsi_spi_api.h」ファイルの

386 行目～387 行目の「TCPIP Defines」定義を使用して下さい。

- #define RSI_IP_CFG_MODE　　DHCP 使用 or 未使用

7.2　環境依存部（ハードウェア等）

サンプルプログラムは、WM-RP-0xS の制御 CPU として RX63N を使用しました。
WM-RP-0xS をお使いになる環境に合わせて、以下の処理を変更して下さい。

7.2.1、エンディアン
「¥src¥Applications¥MCU」フォルダ内「rsi_global.h」ファイル
53 行目～54 行目にて、エンディアンを設定をしています。
必要に応じて以下の定義の変更をして下さい。

- #define RSI_LITTLE_ENDIAN　　0:Big endian , 1:Little endian

7.2.2、WM-RP-0xS レスポンス待ちタイムアウト処理定義
「¥src¥Applications¥MCU」フォルダ内「rsi_global.h」ファイル
76 行目～169 行目にて、WM-RP-0xS レスポンス待ちのタイムアウト処理定義をしています。
必要に応じて各定義の値を変更して下さい。
※タイムアウト処理は、WM-RP-0xS からのレスポンス待ちで使用します。
通常であれば、レスポンスが返ってくるまでの待ちとなりますので、環境に合わせて値を設定して下さい。

- #define RSI_TICKS_PER_SECOND　　この値をベースにタイムアウト処理が行われます。
使用される制御 CPU に合わせて適宜変更して下さい。

7.2.3、SPI
「¥src¥API_Lib」フォルダ内「rsi_hal_mcu_spi.c」ファイル
にて、SPI の処理を行っております。
WM-RP-0xS を使用する環境に合わせて変更をして下さい。
また、「¥src¥API_Lib」フォルダ内「rsi_hal.h」ファイル
41 行目～44 行目にて、SPI を使用する為の定義をしています。
必要に応じて各定義の値を変更して下さい。

- #define RSI_SPI_SEND_BYTE(A)　　SPI による 8bit ライト
- #define RSI_SPI_SEND_4BYTE(A)　　SPI による 32bit ライト
- #define RSI_SPI_READ_BYTE(B)　　SPI による 8bit リード
- #define RSI_SPI_READ_4BYTE(B)　　SPI による 32bit リード

7.2.4、割り込み
「¥src¥API_Lib」フォルダ内「rsi_hal_mcu_interrupt.c」ファイル
にて、割り込みの処理を行っております。
WM-RP-0xS を使用する環境に合わせて変更をして下さい。

※本サンプルプログラムでは、ポーリング処理（ポート入力）を行う事で WM-RP-0xS からの割り込み処理を行っております。
サンプルプログラム「¥src」フォルダ内「wm_rp_04s.c」の「wm_rp_04s_check_intr_level()」関数をメインから常に呼び出す事で、INTR 信号がローレベルかを確認しております。
合わせて、「¥src¥API_Lib」フォルダ内「rsi_interrupt.c」ファイルも参考にして下さい。具体的な割り込みの処理を記述しております。

7.2.5、ハードウェアタイマ

「¥src¥API_Lib」フォルダ内「rsi_hal_mcu_timers.c」ファイルにて、ハードウェアタイムの処理をしています。
必要に応じて処理を変更して下さい。

7.2.6、I/O ポート

「¥src¥API_Lib」フォルダ内「rsi_hal_mcu_ioports.c」ファイルにて、I/O ポートの処理をしています。
必要に応じて処理を変更して下さい。

※サンプルプログラムでは、使用しておりません。

ご注意

・本文書の著作権は株式会社アルファプロジェクトが保有します。
・本文書の内容を無断で転載することは一切禁止します。
・本文書に記載されているサンプルプログラムの著作権は株式会社アルファプロジェクトが保有します。
・本文書に記載されている内容およびサンプルプログラムについての技術サポートは一切受け付けておりません。
・本サンプルプログラムに関して、ルネサスエレクトロニクスへのお問い合わせはご遠慮ください。
・本文書の内容およびサンプルプログラムに基づき、アプリケーションを運用した結果、万一損害が発生しても、弊社およびルネサスエレクトロニクスでは一切責任を負いませんのでご了承下さい。
・本文書の内容については、万全を期して作成いたしましたが、万一ご不審な点、誤りなどお気付きの点がありましたら弊社までご連絡下さい。
・本文書の内容は、将来予告なしに変更されることがあります。

商標について

・RX63N は、株式会社ルネサスエレクトロニクスの登録商標、商標または商品名称です。
・RX は、株式会社ルネサスエレクトロニクスの登録商標、商標または商品名称です。

本文書では下記のように省略して記載している場合がございます。ご了承下さい。
・High-performance Embedded Workshop は HEW

・その他の会社名、製品名は、各社の登録商標または商標です。

ALPHA PROJECT Co.,LTD.
株式会社アルファプロジェクト
〒431-3114
静岡県浜松市東区積志町８３４
http://www.apnet.co.jp
E-MAIL : query@apnet.co.jp