Renesas Synergy™
WM-RP-0xS サンプルプログラム解説 (AP-S3A7-0A)

１版　２０１８年０４月２３日

1.　概要

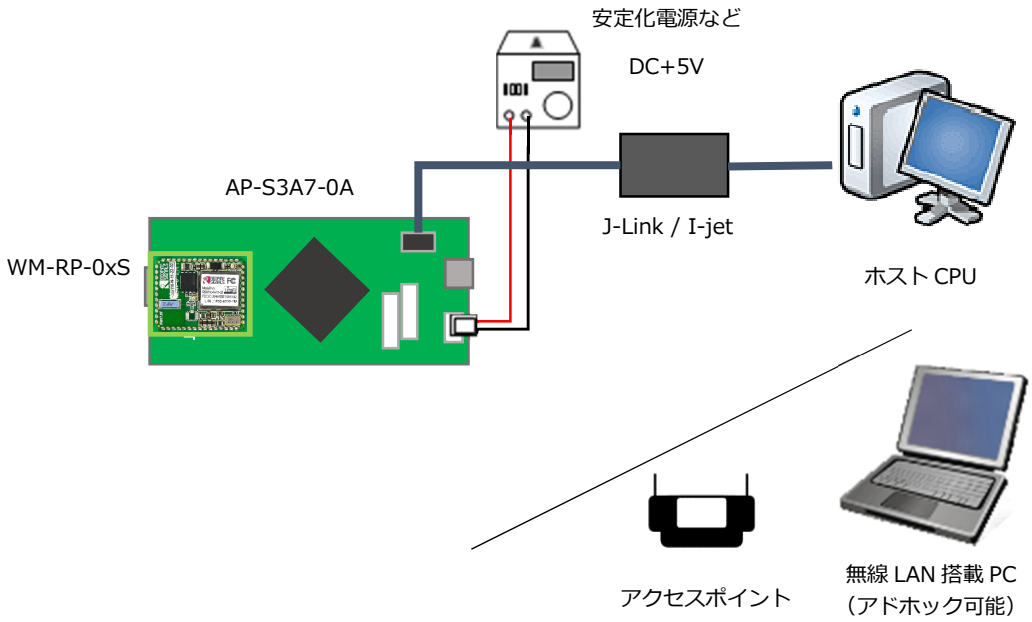
1.1　概要

本アプリケーションノートでは、AP-S3A7-0A(S3A7 CPU)上で動作する WM-RP-0xS 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で使用する主な機能を以下に記します。

デバイス	機能	動作内容
AP-S3A7-0A	・シリアルペリフェラルインタフェース（SPI0）	・SPI 通信 ・ネットワーク通信によるエコーバックサーバ

1.2　接続概要

本サンプルプログラムの動作を確認する上で必要な CPU ボードの接続例を以下に示します。



※AP-S3A7-0A と J-Link を直接接続することはできません。
　AP-S3A7-0A 側(ハーフピッチコネクタ)と J-Link 側(フルピッチコネクタ)を接続するための変換アダプタが必要となります。
　変換アダプタについては、J-Link 取扱店へご確認ください。

1.3　動作内容・動作環境

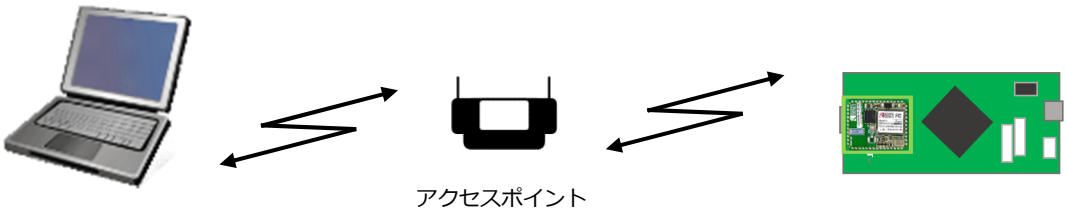
本サンプルプログラムは、6 種類の通信モードを切り替えて使用することができます。
通信モードを切り替える方法は、「3.1.4　TCP/IP 通信動作モード選択方法」および「3.1.8　UDP 通信動作モード選択方法」で説明します。

各動作の内容と、動作確認に必要な機器を以下に示します。

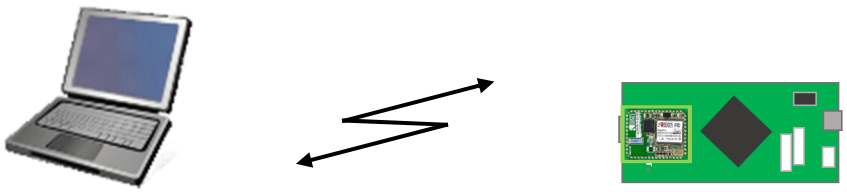
動作内容	動作確認に必要な機器
TCP/IP 通信（アドホックモード　クリエータ） エコーバックサーバ	・アドホック通信可能なホスト PC
TCP/IP 通信（アドホックモード　ジョイナー） エコーバックサーバ	・アドホック通信可能なホスト PC
TCP/IP 通信（インフラストラクチャモード） エコーバックサーバ	・ネットワーク通信可能なホスト PC ・アクセスポイント
UDP 通信（アドホックモード　クリエータ） エコーバックサーバ	・アドホック通信可能なホスト PC
UDP 通信（アドホックモード　ジョイナー） エコーバックサーバ	・アドホック通信可能なホスト PC
UDP 通信（インフラストラクチャモード） エコーバックサーバ	・ネットワーク通信可能なホスト PC ・アクセスポイント

「インフラストラクチャモード」と「アドホックモード」のネットワーク構成イメージを以下に示します。

・インフラストラクチャモード
　アクセスポイント経由で、無線通信を行います。



・アドホックモード
　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。



1.4　本サンプルプログラムの入手方法

本サンプルプログラムおよび本書含むアプリケーションノートは、弊社 Web サイトの WM-WP シリーズ製品ページで公開されています。

株式会社アルファプロジェクト
WM-RP シリーズ製品ページ　http://www.apnet.co.jp/product/superh/wm-rp.html

（CPU ボードの情報については、
AP-S3A7-0A 製品ページ　http://www.apnet.co.jp/product/synergy/ap-s3a7-0a.html　をご覧ください。）

1.5　開発環境について

本サンプルプログラムは統合開発環境「e2 studio」と「Synergy Software Package（以下、SSP）」を用いて開発されています。
本サンプルプログラムに対応する開発環境、SSP、コンパイラ、デバッガのバージョンは次の通りです。

ソフトウェア	バージョン	備考
e2 studio	v6.2.0	–
SSP	v1.4.0	–
GCC ARM Embedded	v4.9.3	–
AP-S3A7-0A 用 Custom BSP	v1.4.0 - ap010000	–

デバッガ	ファームバージョン	備考
J-Link	v4.94h	Segger Microcontroller Systems 社

または

ソフトウェア	バージョン	備考
IAR Embedded Workbench® for Renesas Synergy™	v8.21.1	IAR Embedded Workbench shared components : v8.100.13
SSP	v1.4.0	–
SSC	v6.2.0	Renesas Synergy™ Standalone Configurator
AP-S3A7-0A 用 Custom BSP	v1.4.0 - ap010000	–

デバッガ	ファームバージョン	備考
I-jet	v1.60 (Probe)	I-jet-Trace : v1.63 IAR システムズ社

1.6　ワークスペースについて

本サンプルプログラムのプロジェクトファイル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
WM-RP-0xS サンプルプログラム プロジェクトフォルダ	¥sample¥ap_s3a7_0a_sample_wmrp

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	
├ Custom BSP	AP-S3A7-0A 用 Custom BSP フォルダ
└ ap_s3a7_0a_sample_wmrp	WM-RP-0xS サンプルプログラムフォルダ
├ .module_descriptions	モジュール情報ファイルフォルダ
├ .settings	設定ファイルフォルダ
├ script	リンカ設定ファイルフォルダ
└ src	ソースフォルダ
└ wifi	WiFi モジュール制御ソースフォルダ
├ API_Lib	「Redpine Signals 社」の SPI 用ライブラリファイル
└ Applications	「Redpine Signals 社」の SPI 用ライブラリファイル

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。
本節では、サンプルプログラムの作成にあたって追加したファイルについてのみ記述し、自動生成ファイルなどに関しては説明を省略します。

<¥sample¥Custom BSP フォルダ内>

AlphaProject.ap_s3a7_0a.1.4.0-ap010000.pack	・・・	AP-S3A7-0A 用 Custom BSP （「3.3.1　Custom BSP の適用方法」参照）

<¥sample¥ap_s3a7_0a_sample_wmrp フォルダ内>

.cproject	・・・	CPROJECT ファイル
.project	・・・	PROJECT ファイル
configuration.xml	・・・	Synergy コンフィギュレータファイル
ap_s3a7_0a_R7FS3A77C3A01CFP.pincfg	・・・	AP-S3A7-0A 用ピンコンフィグファイル
R7FS3A77C3A01CFP.pincfg	・・・	S3A7 CPU 用　デフォルト ピンコンフィグファイル ※ AP-S3A7-0A 用の設定はしてありません。
ap_s3a7_0a_sample_wmrp Debug.jlink	・・・	AP-S3A7-0A　WM-RP-0xS サンプルプログラム デバッグおよびランタイム設定ファイル
ap_s3a7_0a_sample_wmrp Debug.launch	・・・	AP-S3A7-0A　WM-RP-0xS サンプルプログラム デバッグおよびランタイム設定ファイル
ap_s3a7_0a_sample_wmrp.eww	・・・	Embedded Workbench ワークスペースファイル
ap_s3a7_0a_sample_wmrp.ewd	・・・	Embedded Workbench EWD ファイル
ap_s3a7_0a_sample_wmrp.ewp	・・・	Embedded Workbench EWP ファイル

<¥sample¥ap_s3a7_0a_sample_wmrp¥script フォルダ内>

S3A7.ld	・・・	e2 studio 用 リンカスクリプトファイル
S3A7.icf	・・・	Embedded Workbench 用 リンカ設定ファイル

<¥sample¥ap_s3a7_0a_sample_wmrp¥src フォルダ内>

hal_entry.c	・・・	hal_entry 関数ソースファイル
ioport_app.c	・・・	I/O ポート制御ソースファイル
ioport_app.h	・・・	I/O ポート制御ヘッダファイル
wifi_thread_entry.c	・・・	WiFi メインアプリケーションソースファイル
common_app.h	・・・	共通ヘッダファイル

<¥sample¥ap_s3a7_0a_sample_wmrp¥src¥wifi フォルダ内>

wm_rp_04s.c	・・・	WiFi モジュールサンプルドライバファイル
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

※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¥ap_s3a7_0a_sample_wmrp¥src¥wifi¥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¥ap_s3a7_0a_sample_wmrp¥src¥wifi ¥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¥ap_s3a7_0a_sample_wmrp¥src¥wifi ¥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¥ap_s3a7_0a_sample_wmrp ¥src¥wifi¥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ためのファイル群です。

※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しません。

<¥sample¥ap_s3a7_0a_sample_wmrp¥src¥wifi ¥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.　AP-S3A7-0A サンプルプログラム

3.1　動作説明

3.1.1　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、以下の動作を行います。

（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、「3.1.3　TCP/IP 通信エコーバックサーバ動作」を参照してください。

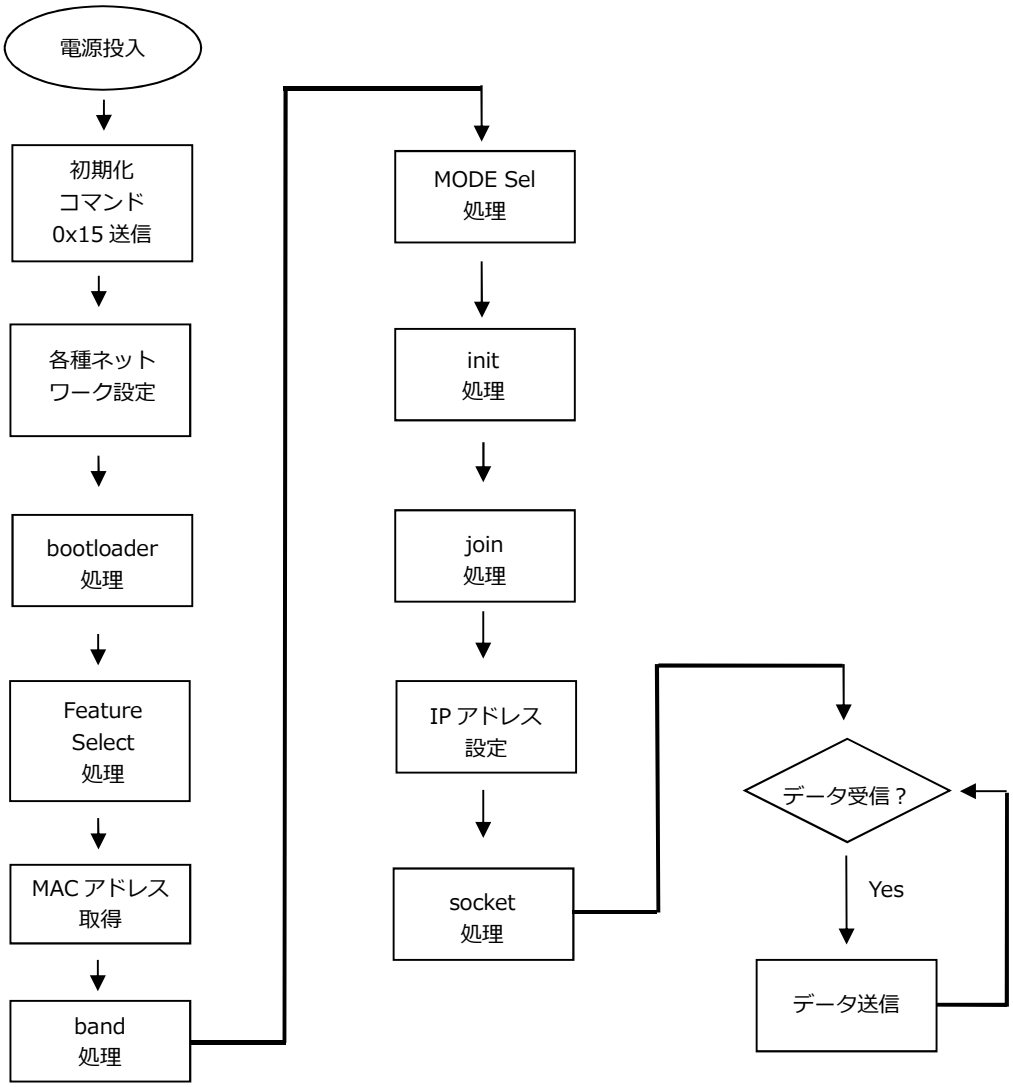


Fig 3.1-1　TCP/IP 通信サンプルプログラム　アドホッククリエータ　WM-RP-0xS 制御フロー

（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、「3.1.3　TCP/IP 通信エコーバックサーバ動作」を参照してください。

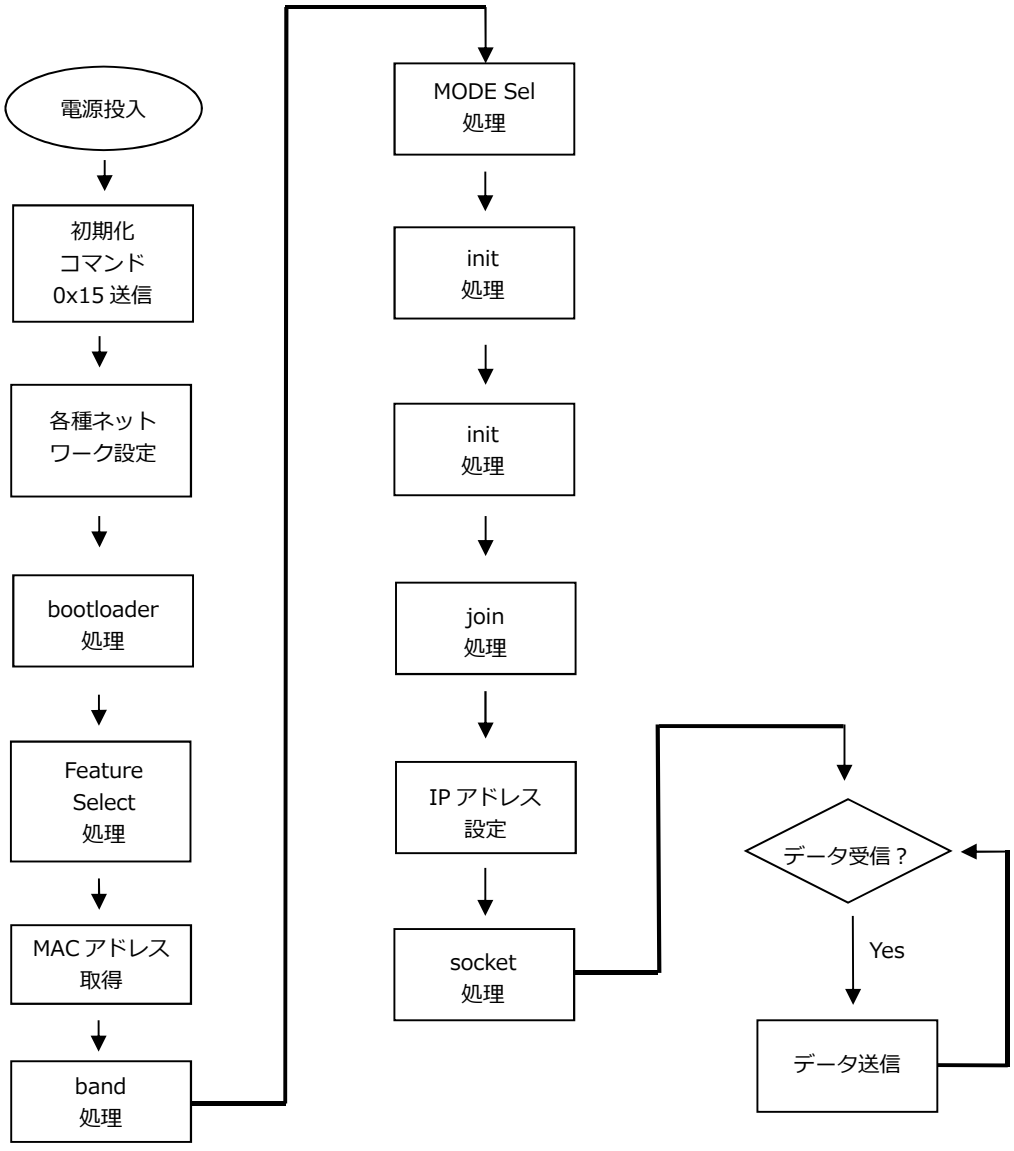


Fig 3.1-2　TCP/IP 通信サンプルプログラム　アドホックジョイナー　WM-RP-0xS 制御フロー

3.1.2　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、以下の動作を行います。

- 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、「3.1.3　TCP/IP 通信エコーバックサーバ動作」を
  　参照してください。

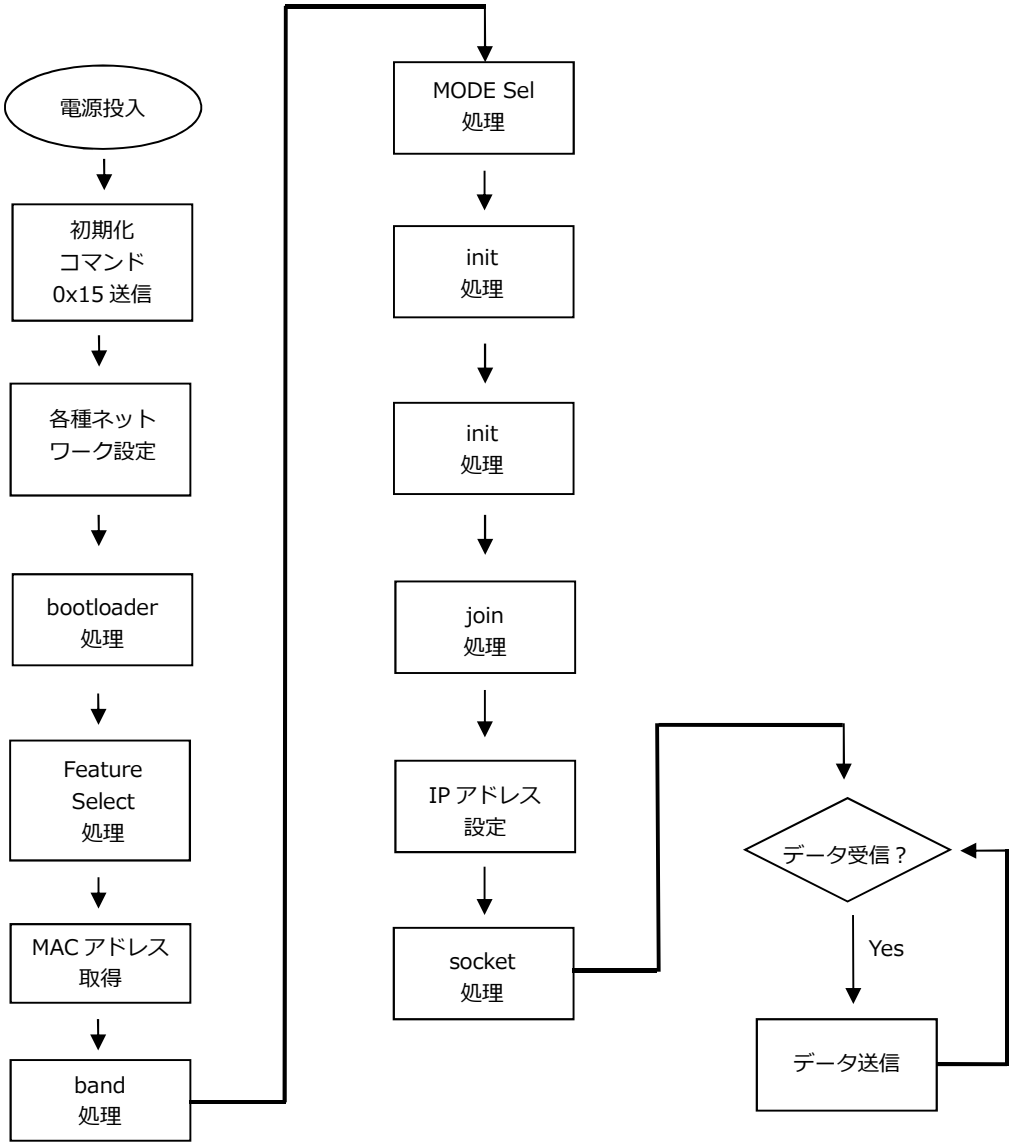


Fig 3.1-3　TCP/IP 通信サンプルプログラム　インフラストラクチャ　WM-RP-0xS 制御フロー

3.1.3　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デフォルトのネットワーク設定を記します。
設定を変更する場合は、「6.　サンプルプログラムの設定変更に関して」をご参照ください。

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_AP-S3A7-0ASample
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	8000

Table3.1-1　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_AP-S3A7-0ASamplePSK
アクセスポイント	WM-RP_AP-S3A7-0ASample
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	8000

Table3.1-2　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、ご利用の環境に沿った適切な値をそれぞれ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）

「3.1.4　TCP/IP 通信動作モード選択」を参考にモードを設定し、ビルドを行った後、
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。
無線 LAN ネットワーク接続に成功すると、AP-S3A7-0A 上の LD1 が点灯します。
② アドホック通信機器の設定を行います。
この際には、「Table3.1-1　TCP/IP ネットワーク設定（アドホックモード）」を参考に設定します。
③ アドホック通信機器を無線 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。
アドホック通信機器側で通信を切断すると、CPU ボードはエコーバックサーバを終了し、LD1 が点滅します。
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）

「3.1.4　TCP/IP 通信動作モード選択」を参考にモードを設定し、ビルドを行った後、
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。
この際には、「Table3.1-1　TCP/IP ネットワーク設定（アドホックモード）」を参考に設定します。
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。
無線 LAN ネットワーク接続に成功すると、AP-S3A7-0A 上の LD1 が点灯します。
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。
アドホック通信機器側で通信を切断すると、CPU ボードはエコーバックサーバを終了し、LD1 が点滅します。
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）

「3.1.4　TCP/IP 通信動作モード選択」を参考にモードを設定し、ビルドを行った後、
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。
無線 LAN ネットワーク接続に成功すると、AP-S3A7-0A 上の LD1 が点灯します。
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。
この際には、「Table3.1-2　TCP/IP ネットワーク設定（インフラストラクチャモード）」を参考に設定します。
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。
ホスト PC 上で通信を切断すると、CPU ボードはエコーバックサーバを終了し、LD1 が点滅します。
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。

3.1.4　TCP/IP 通信動作モード選択方法

本サンプルプログラムで、TCP/IP 通信の動作モードを選択する方法を説明します。
以下のようにソースコードの変更を行い、動作確認を行ってください。（ソースコードの例はデフォルト設定です。）

- プロトコルの選択

  ¥ap_s3a7_0a_sample_wmrp¥src¥common_app.h

35	/* Set the com protocol */
36	#define MODULE_TCP_SERVER
37	//#define MODULE_TCP_CLIENT
38	//#define MODULE_UDP

  TCP/IP 通信・サーバ機能を行う場合、36 行目のみ有効にしてください。
  (37、38 行目はコメントアウト)

- ネットワークの選択

  ¥ap_s3a7_0a_sample_wmrp¥src¥common_app.h

40	/* Set the network */
41	//#define MODE_OPEN
42	#define MODE_INFRASTRUCTURE

  ・アドホックモード（クリエータ、ジョイナー）の場合
  41 行目を有効に、42 行目は無効にしてください。
  ・インフラストラクチャモードの場合
  41 行目は無効に、42 行目を有効にしてください。

- IBSS モードの選択

  ¥ap_s3a7_0a_sample_wmrp¥src¥common_app.h

44	/* Set the IBSS mode(only when MODE_OPEN is defined) */
45	//#define MODE_IBSS_CREATOR
46	#define MODE_IBSS_JOINER

  ・アドホックモード　クリエータの場合
  45 行目を有効に、46 行目を無効にしてください。
  ・アドホックモード　ジョイナーの場合
  45 行目を無効に、46 行目を有効にしてください。
  ・インフラストラクチャモードの場合
  設定する必要はありません。

UDP 通信の設定については「3.1.8　UDP 通信動作モード選択」をご覧ください。
ネットワークの設定などの変更箇所に関しては、「6.　サンプルプログラムの設定変更に関して」を参考にしてください。

3.1.5　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、以下の動作を行います。

（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、「3.1.7　UDP 通信エコーバックサーバ動作」を参照してください。

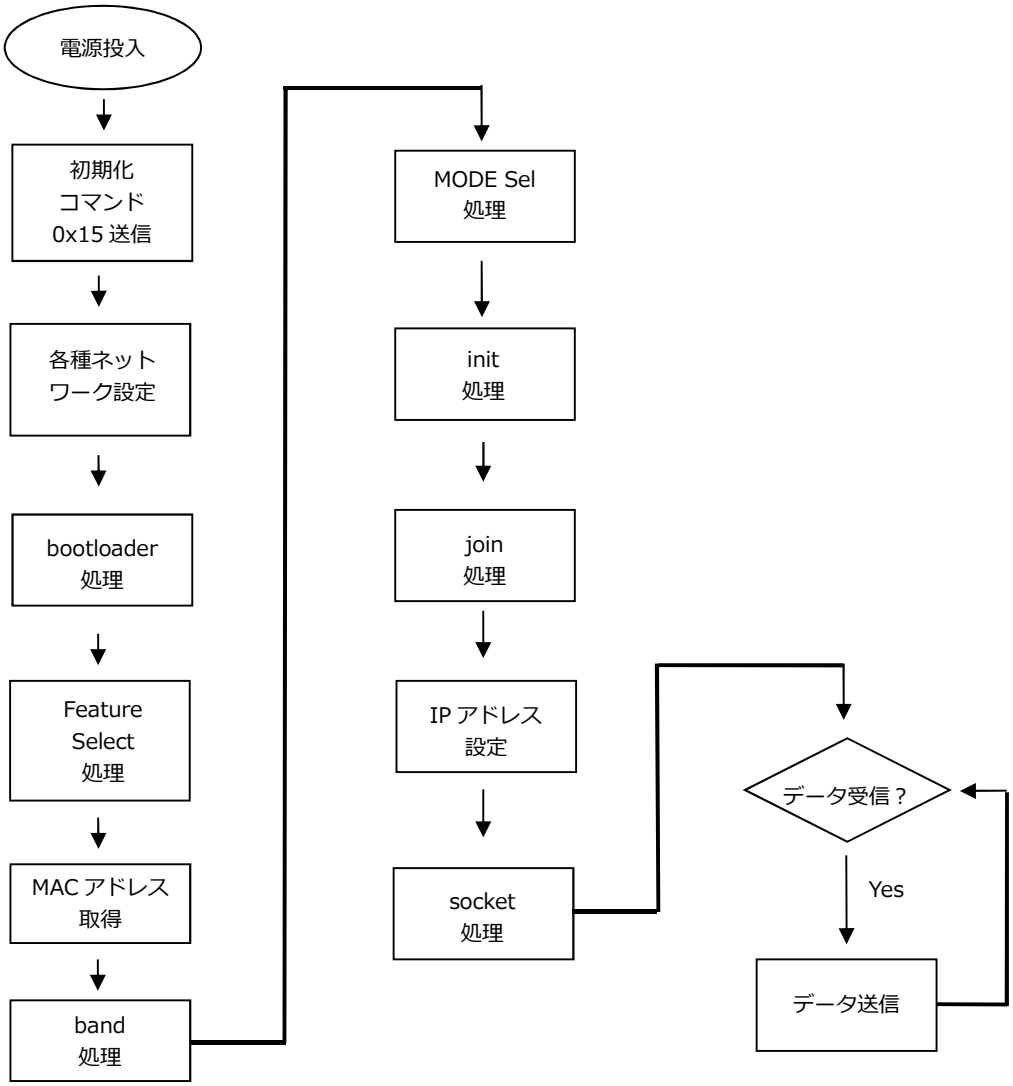


Fig 3.1-4　UDP 通信サンプルプログラム　アドホッククリエータ　WM-RP-0xS 制御フロー

（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、「3.1.7　UDP 通信エコーバックサーバ動作」を参照してください。

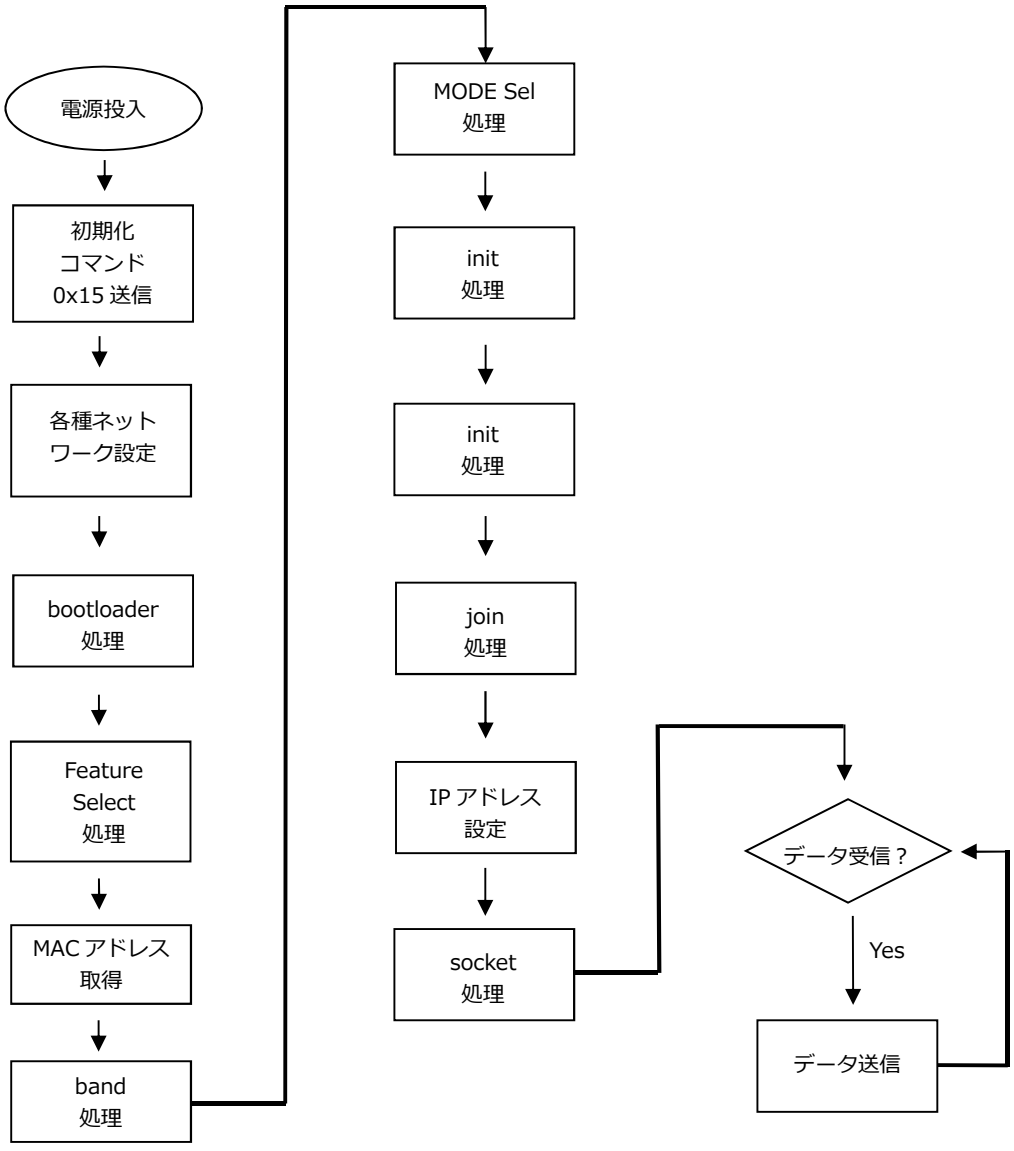


Fig 3.1-5　TCP/IP 通信サンプルプログラム　アドホックジョイナー　WM-RP-0xS 制御フロー

3.1.6　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）

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）は、以下の動作を行います。

- 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、「3.1.7 UDP 通信エコーバックサーバ動作」を参照してください。

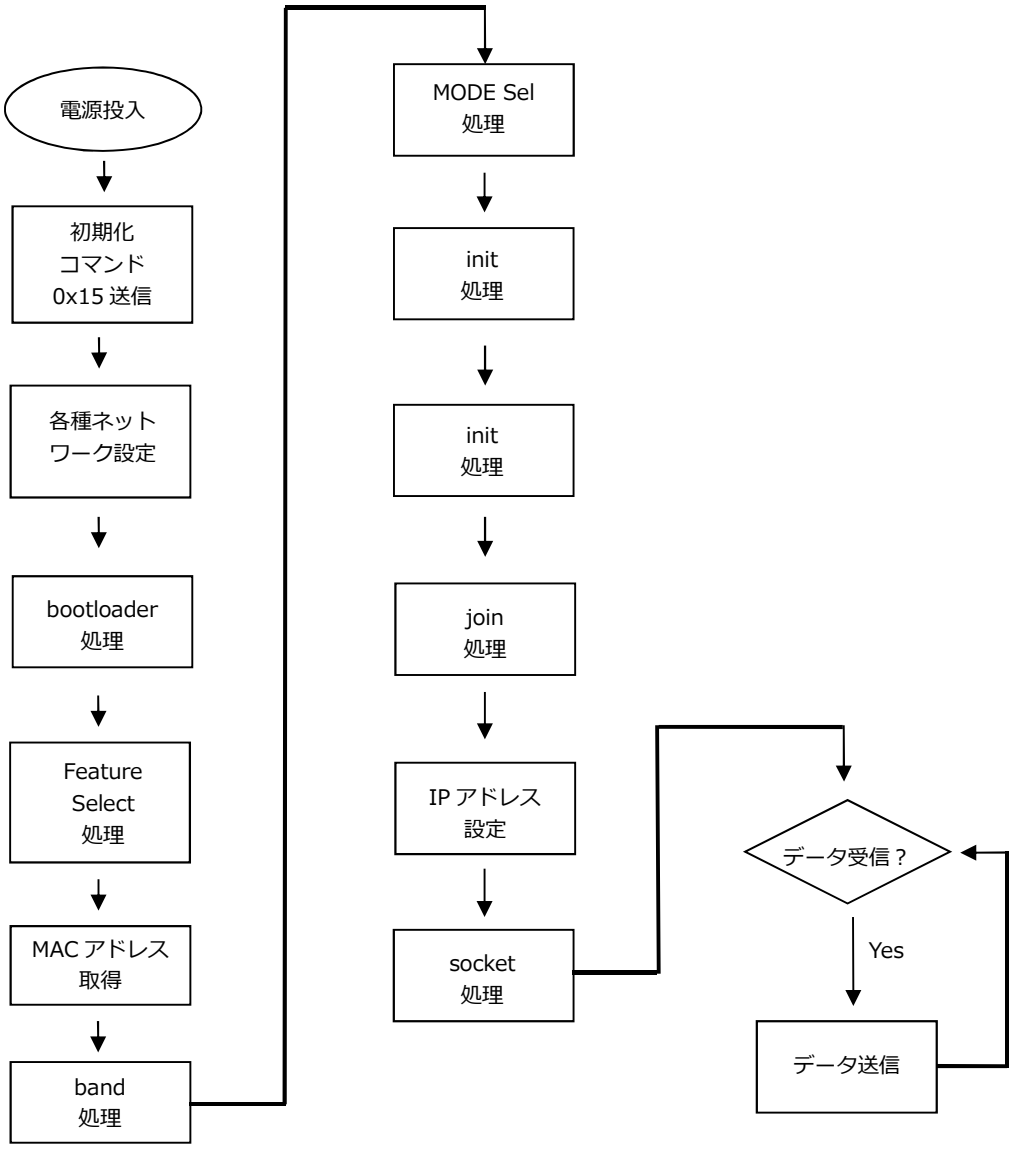


Fig 3.1-6　UDP 通信サンプルプログラム　インフラストラクチャ　WM-RP-0xS 制御フロー

3.1.7　UDP 通信エコーバックサーバ動作

（１）　UDP ネットワーク設定

以下にサンプルプログラムのデフォルトのネットワーク設定を記します。
設定を変更する場合は、「6.　サンプルプログラムの設定変更に関して」をご参照ください。

UDP 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_AP-S3A7-0ASample
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	8000

Table3.1-3　UDP 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_AP-S3A7-0ASamplePSK
アクセスポイント	WM-RP_AP-S3A7-0ASample
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	8000

Table3.1-4　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、ご利用の環境に沿った適切な値をそれぞれ設定しビルドしてください。

（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）

「3.1.8　UDP 通信動作モード選択」を参考にモードを設定し、ビルドを行った後、
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。
無線 LAN ネットワーク接続に成功すると、AP-S3A7-0A 上の LD1 が点灯します。
② アドホック通信機器の設定を行います。
この際には、「Table3.1-3　UDP ネットワーク設定（アドホックモード）」を参考に設定します。
③ アドホック通信機器を無線 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）　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）

「3.1.8　UDP 通信動作モード選択」を参考にモードを設定し、ビルドを行った後、
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。
この際には、「Table3.1-3　UDP ネットワーク設定（アドホックモード）」を参考に設定します。
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。
無線 LAN ネットワーク接続に成功すると、AP-S3A7-0A 上の LD1 が点灯します。
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）　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）

「3.1.8　UDP 通信動作モード選択」を参考にモードを設定し、ビルドを行った後、
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。
無線 LAN ネットワーク接続に成功すると、AP-S3A7-0A 上の LD1 が点灯します。
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。
この際には、「Table3.1-4　UDP ネットワーク設定（インフラストラクチャモード）」を参考に設定します。
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。

3.1.8　UDP 通信動作モード選択方法

本サンプルプログラムで、UDP 通信の動作モードを選択する方法を説明します。
以下のようにソースコードの変更を行い、動作確認を行ってください。（ソースコードの例はデフォルト設定です。）

- プロトコルの選択

  ¥ap_s3a7_0a_sample_wmrp¥src¥common_app.h

```
35  /* Set the com protocol */
36  #define MODULE_TCP_SERVER
37  //#define MODULE_TCP_CLIENT
38  //#define MODULE_UDP
```

  UDP 通信を行う場合、38 行目のみ有効にしてください。
  (36、37 行目はコメントアウト)

- ネットワークの選択

  ¥ap_s3a7_0a_sample_wmrp¥src¥common_app.h

```
40  /* Set the network */
41  //#define MODE_OPEN
42  #define MODE_INFRASTRUCTURE
```

  ・アドホックモード（クリエータ、ジョイナー）の場合
  41 行目を有効に、42 行目は無効にしてください。
  ・インフラストラクチャモードの場合
  41 行目は無効に、42 行目を有効にしてください。

- IBSS モードの選択

  ¥ap_s3a7_0a_sample_wmrp¥src¥common_app.h

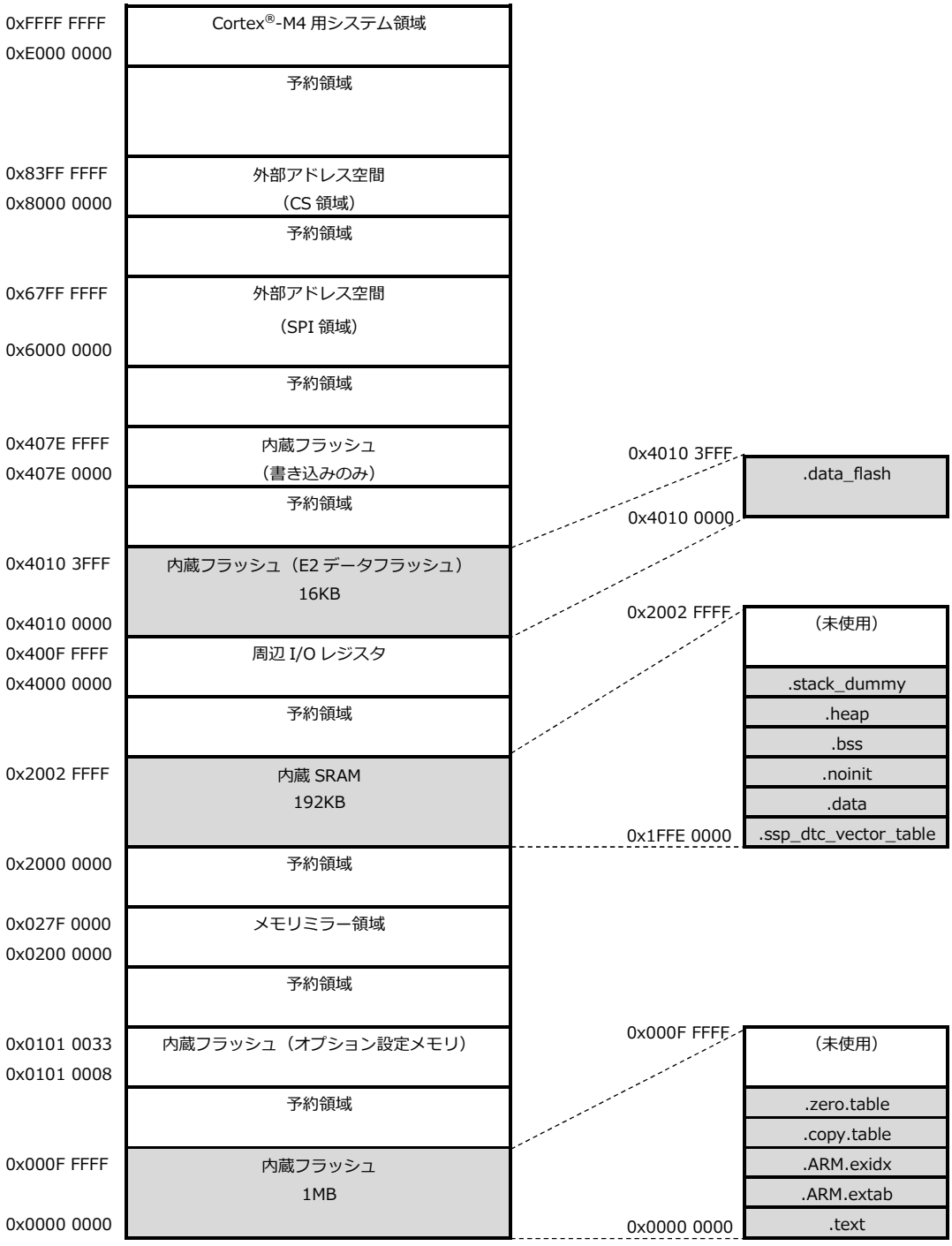
```
44  /* Set the IBSS mode(only when MODE_OPEN is defined) */
45  //#define MODE_IBSS_CREATOR
46  #define MODE_IBSS_JOINER
```

  ・アドホックモード　クリエータの場合
  45 行目を有効に、46 行目を無効にしてください。
  ・アドホックモード　ジョイナーの場合
  45 行目を無効に、46 行目を有効にしてください。
  ・インフラストラクチャモードの場合
  設定する必要はありません。

TCP/IP 通信の設定については「3.1.4　TCP/IP 通信動作モード選択」をご覧ください。
ネットワークの設定などの変更箇所に関しては、「6.　サンプルプログラムの設定変更に関して」を参考にしてください。

3.2　メモリマップ

e2 studio のプロジェクトのメモリマップを以下に示します。



3.3　e2 studio を用いたプロジェクトのビルド・デバッグ

サンプルプログラムを CPU ボード上で実行するためには、e2 studio 上に一度サンプルプログラムをインポートし、ビルドを行う必要があります。

e2 studio 上へのサンプルプログラムのインポート方法、サンプルプログラムのビルド・デバッグ方法については本節で説明します。

3.3.1　Custom BSP の適用方法

e2 studio 上にサンプルプログラムをインポートする前に、CPU ボード用の Custom BSP を準備する必要があります。サンプルプログラム内の「¥sample¥Custom BSP」にある「**AlphaProject.ap_s3a7_0a.1.4.0-ap010000.pack**」を「<e2 studio のインストールフォルダ>¥internal¥projectgen¥arm¥Packs」のフォルダへコピーしてください。

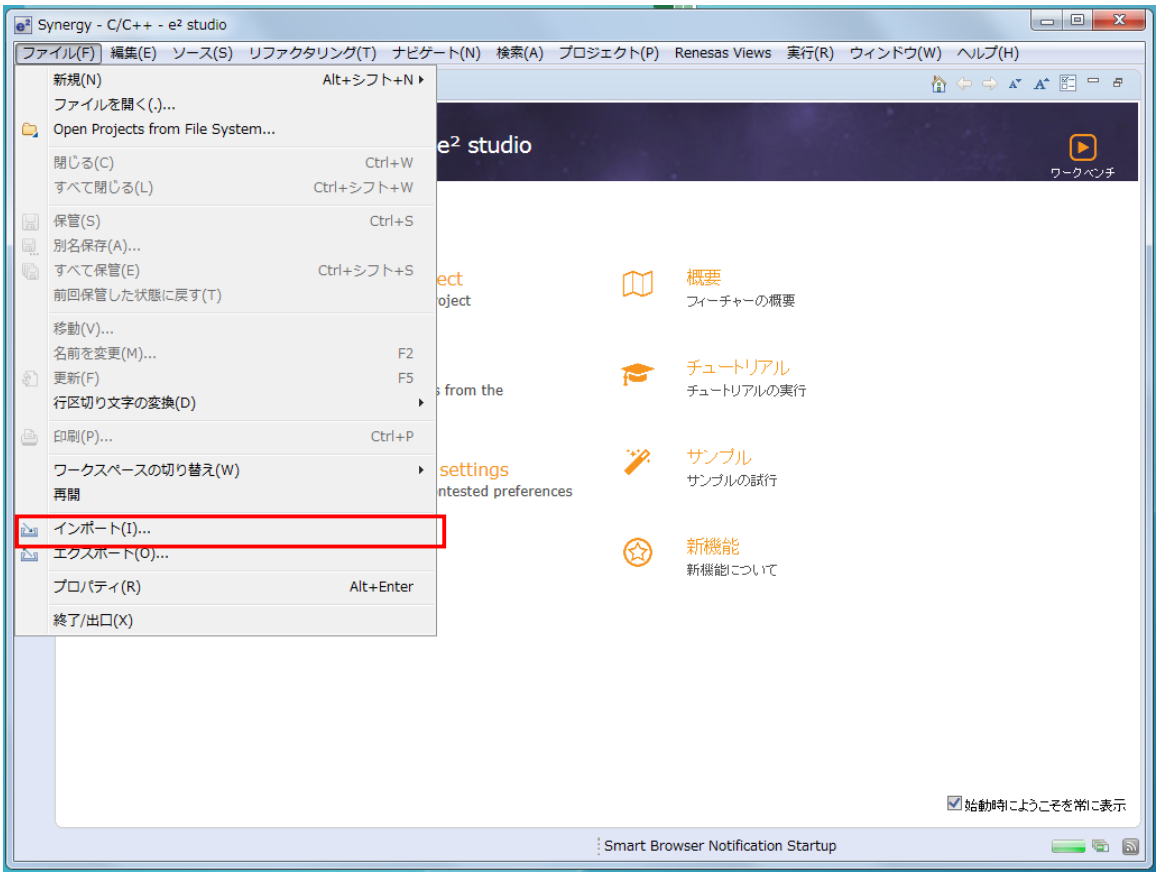
例：”C:¥Renesas¥e2_studio”に e2 studio をインストールしている場合、下記フォルダに Custom BSP ファイルをコピーしてください。

C:¥Renesas¥e2_studio¥internal¥projectgen¥arm¥Packs

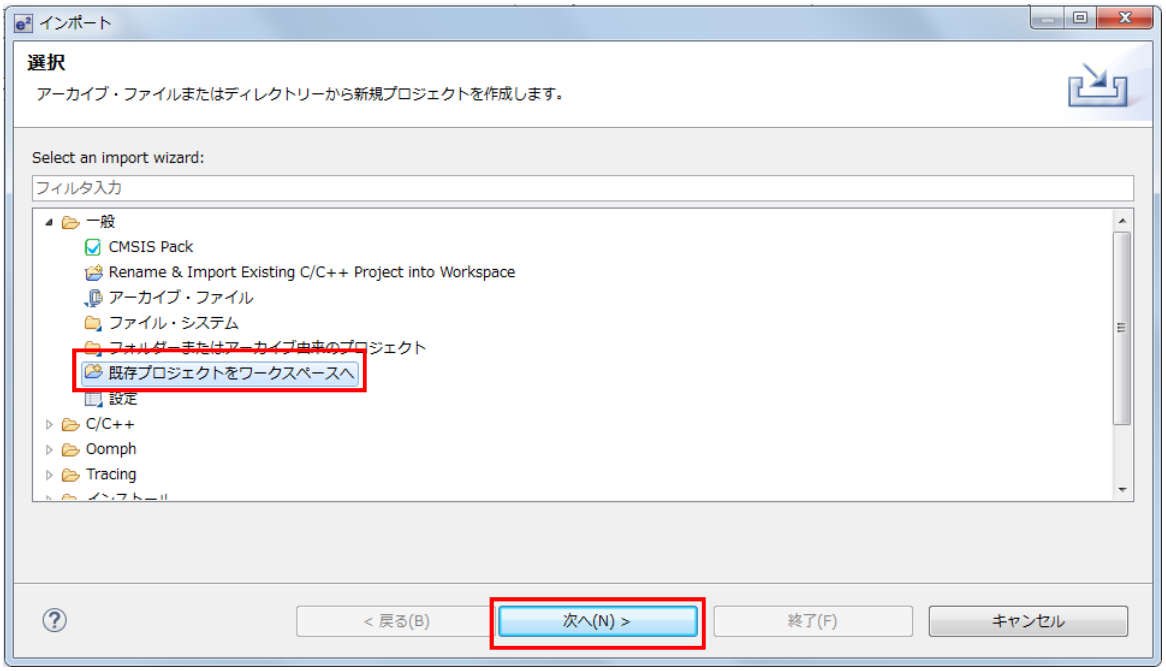
以上で Custom BSP の適用は完了です。

3.3.2　インポート方法

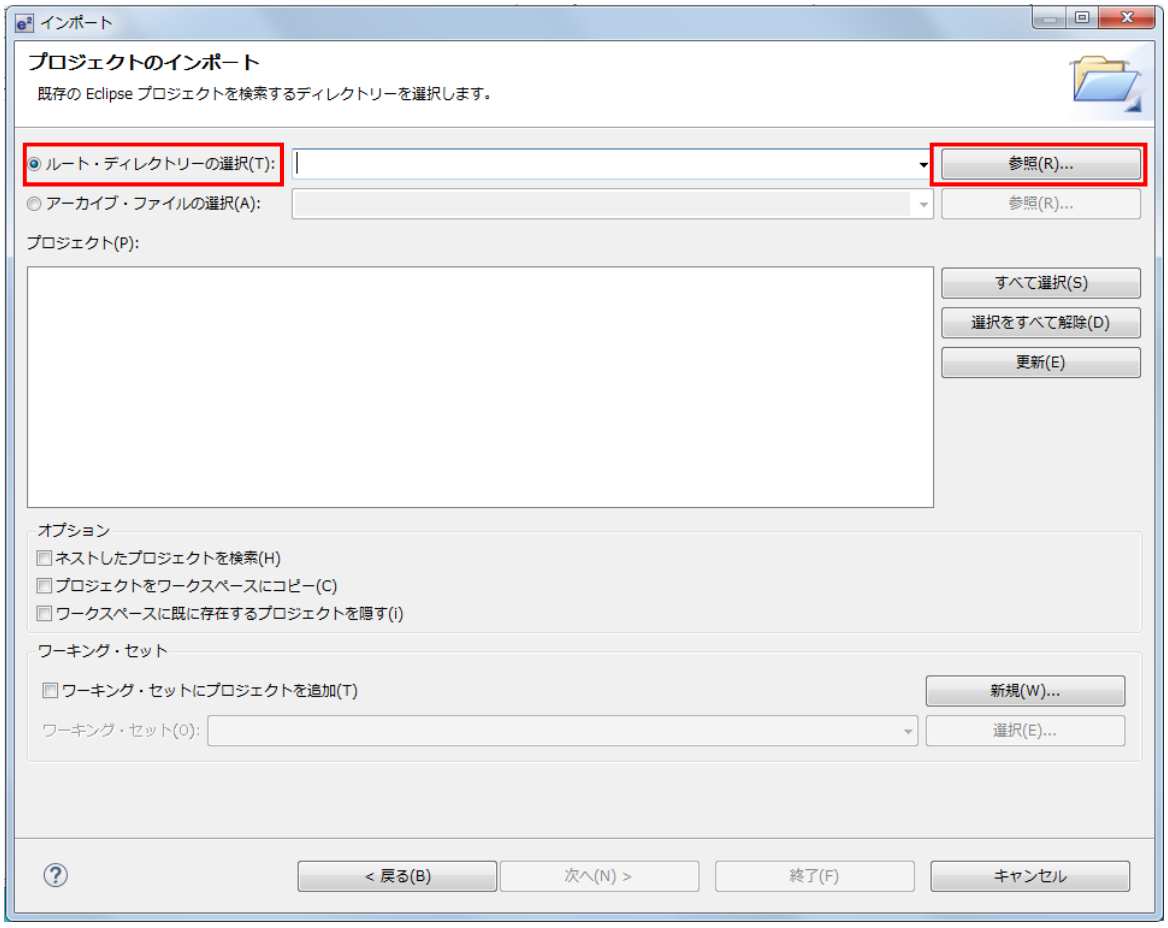
①　e2 studio を起動し、ツールバーの［ファイル］→［インポート］を選択します。



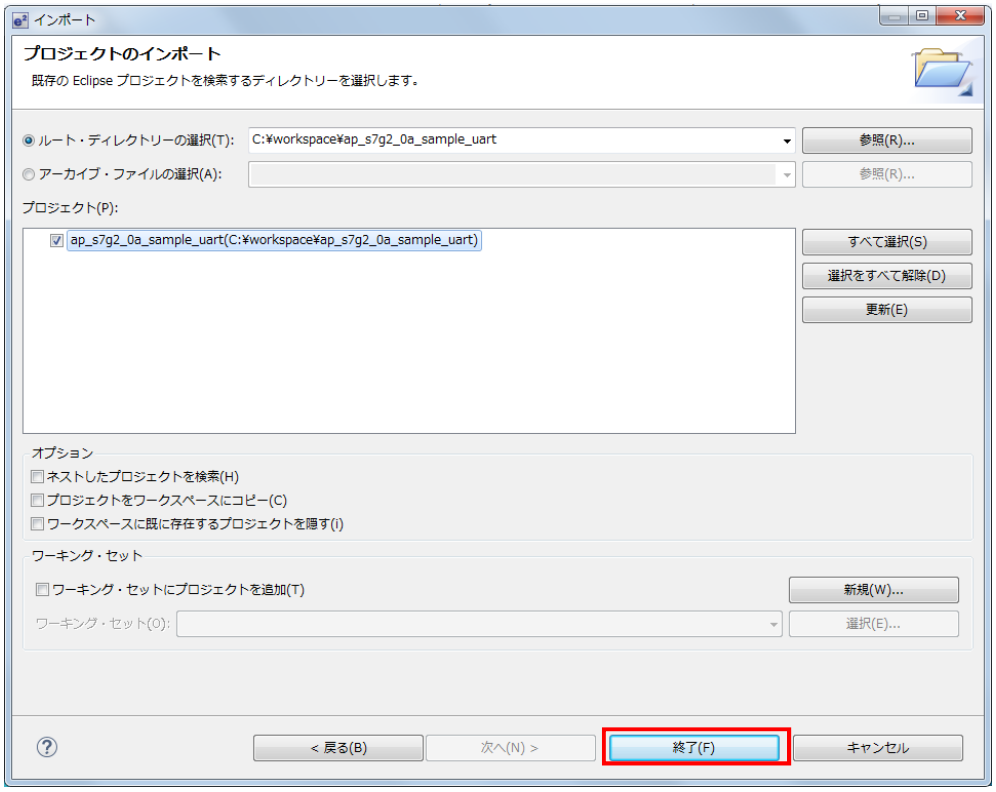
②　［既存のプロジェクトをワークスペースへ］を選択し［次へ］を選択します。



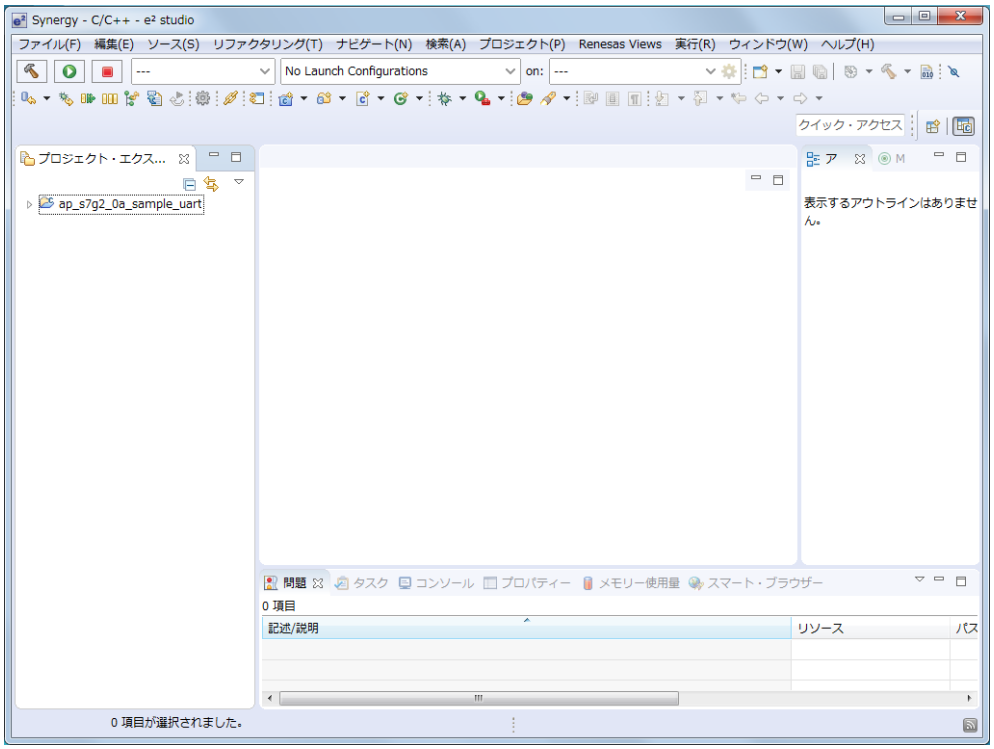
③　［ルート・ディレクトリーの選択］を選択し、［参照］からサンプルプログラムのフォルダを選択します。



④　[終了] を選択します。



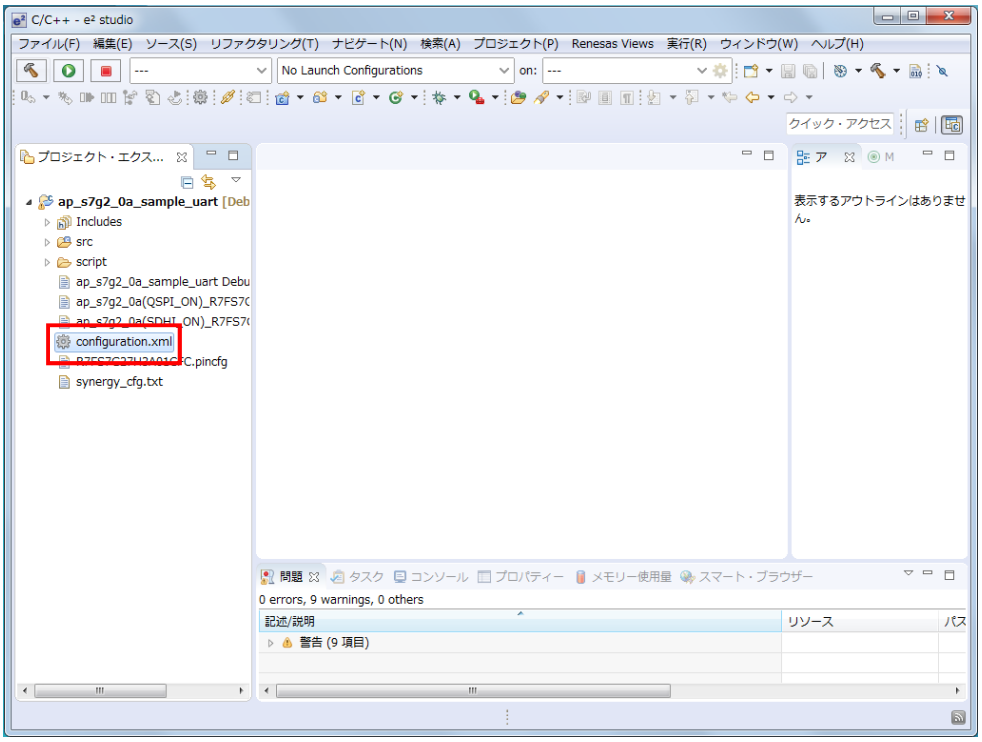
⑤　ナビゲーションウィンドウにサンプルプログラムのプロジェクトが追加されていることを確認します。



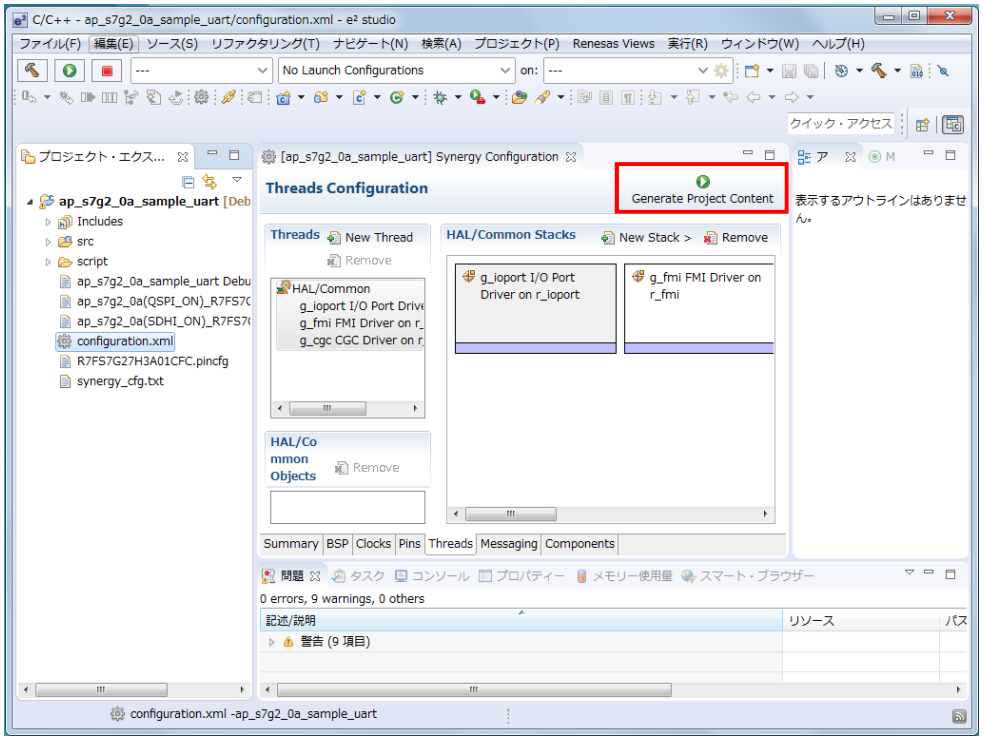
以上でプロジェクトのインポートは完了です。

3.3.3　ビルド方法

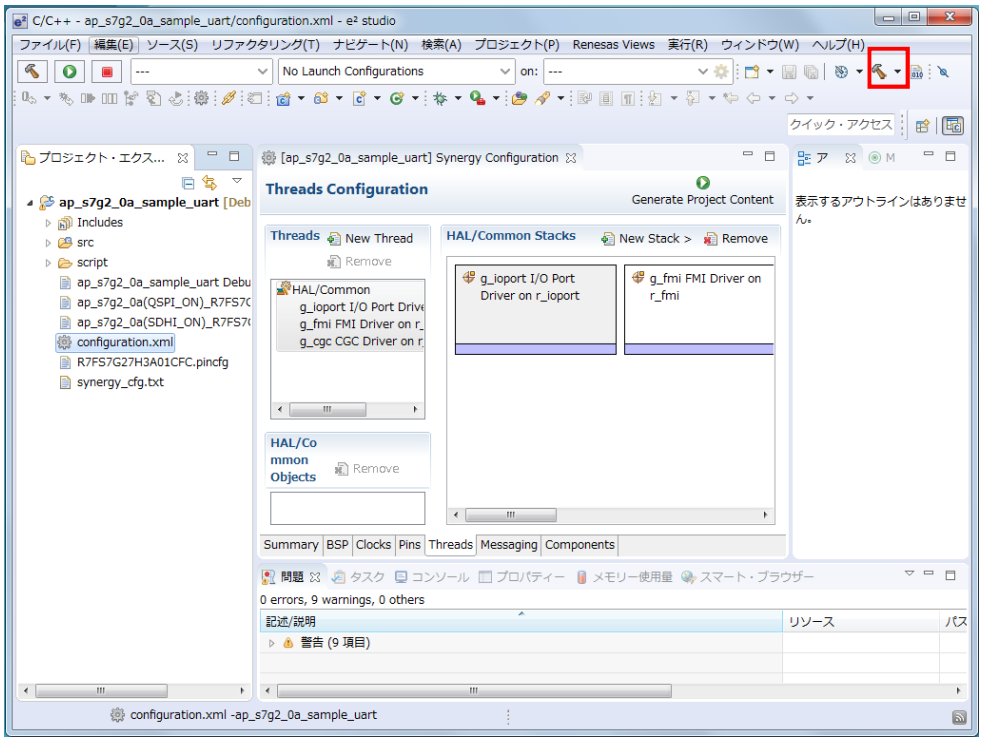
①　プロジェクトのコンフィギュレータファイルを開きます。



②　[Generate Project Content] をクリックし、自動作成ファイルを出力して設定をプロジェクトに適用します。



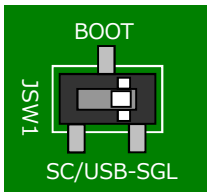
③　ツールバーからビルドアイコンを選択します。
ビルドが成功すると、¥Debug ワークフォルダにオブジェクトファイルが生成されます。



e2 studio の詳細な使用方法に関しては、 e2 studio のマニュアルを参照してください。

3.3.4　デバッグ方法

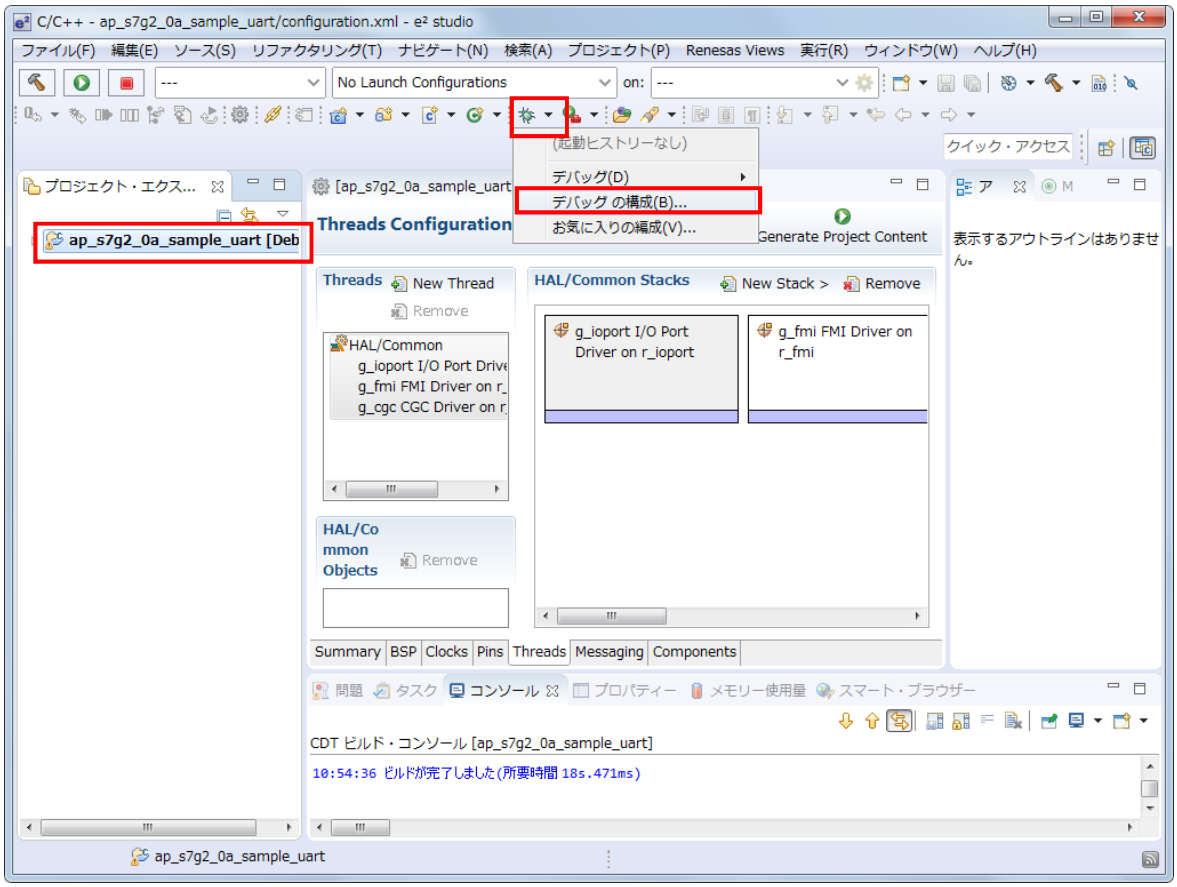
①　「3.3.3　ビルド方法」を参考に、プロジェクトをビルドしてください。
②　ボード上のディップスイッチを以下のように設定してください。



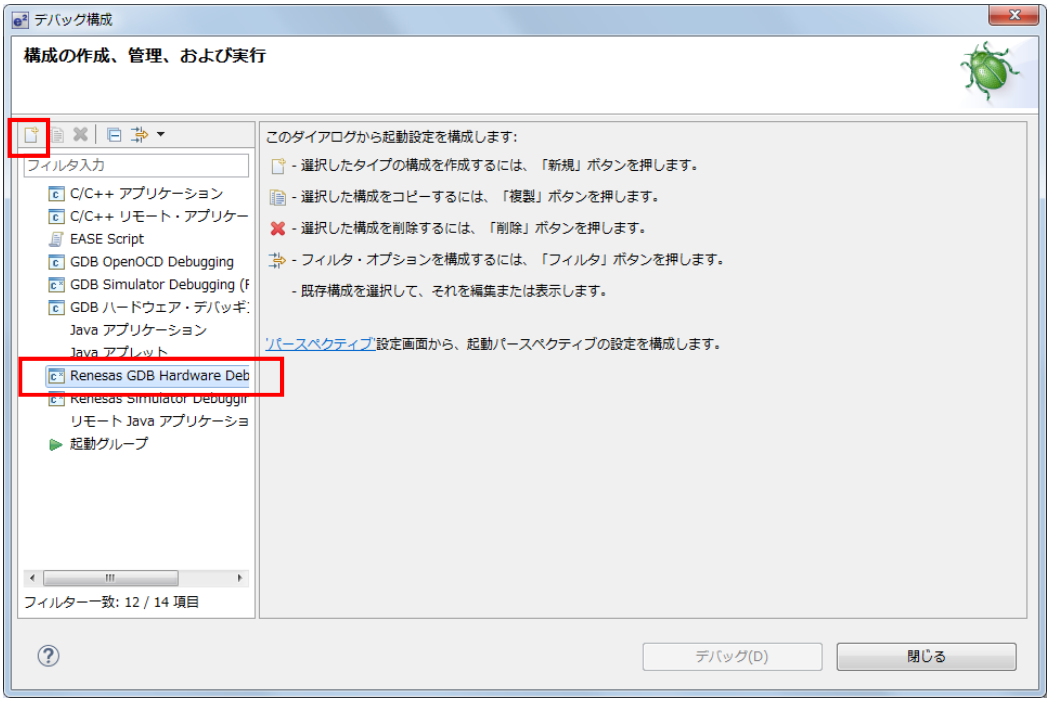
JSW1：SGL

シングルチップモード

③　ボードに電源を投入してください。
④　プロジェクトを選択し、メニューバーから [デバッグの構成] を開きます。

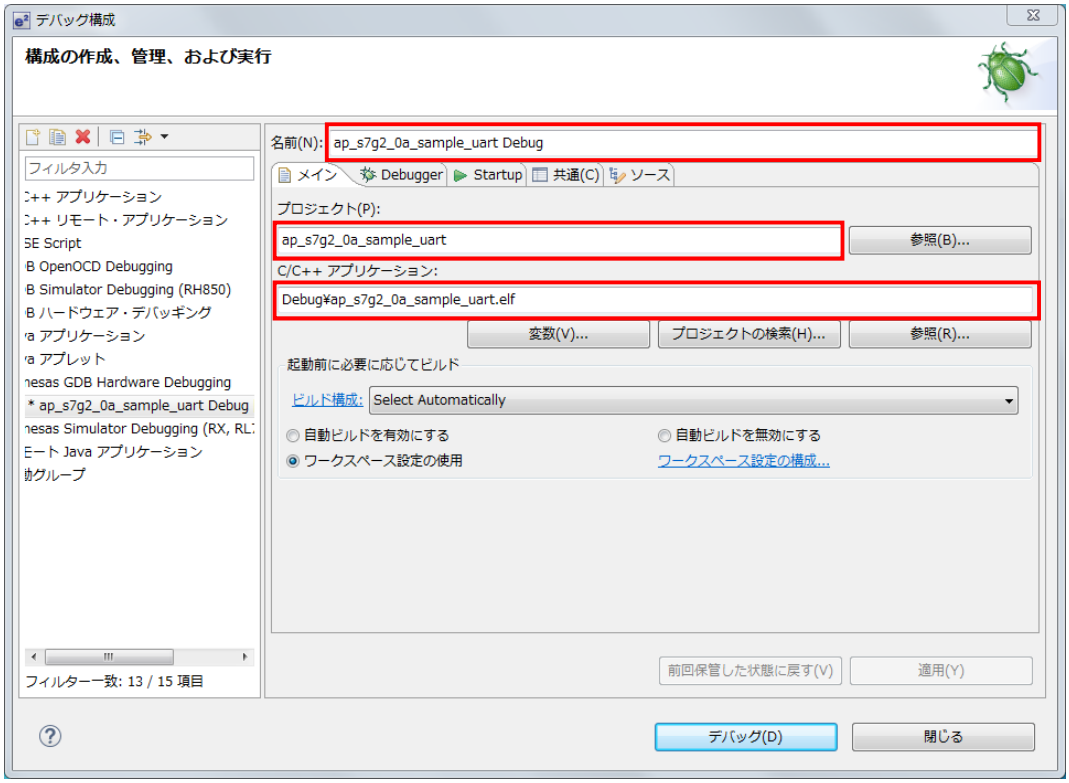


⑤　[Renesas GDB Hardware Debug] を選択後、 [新規の起動構成] をクリックします。

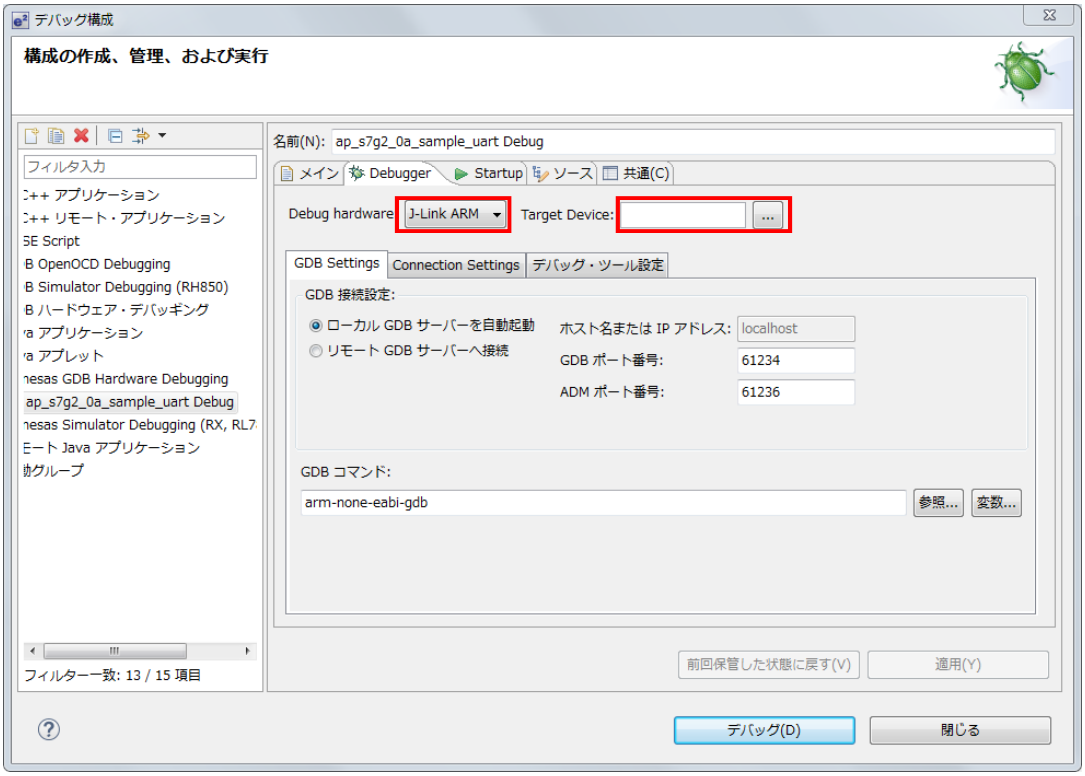


⑥　新規作成されたデバッグ構成を選択し［メイン］タブを開き、各項目を以下のように設定してください。
（図は「ap_s7g2_0a_sample_uart」をデバッグする際の例です。）

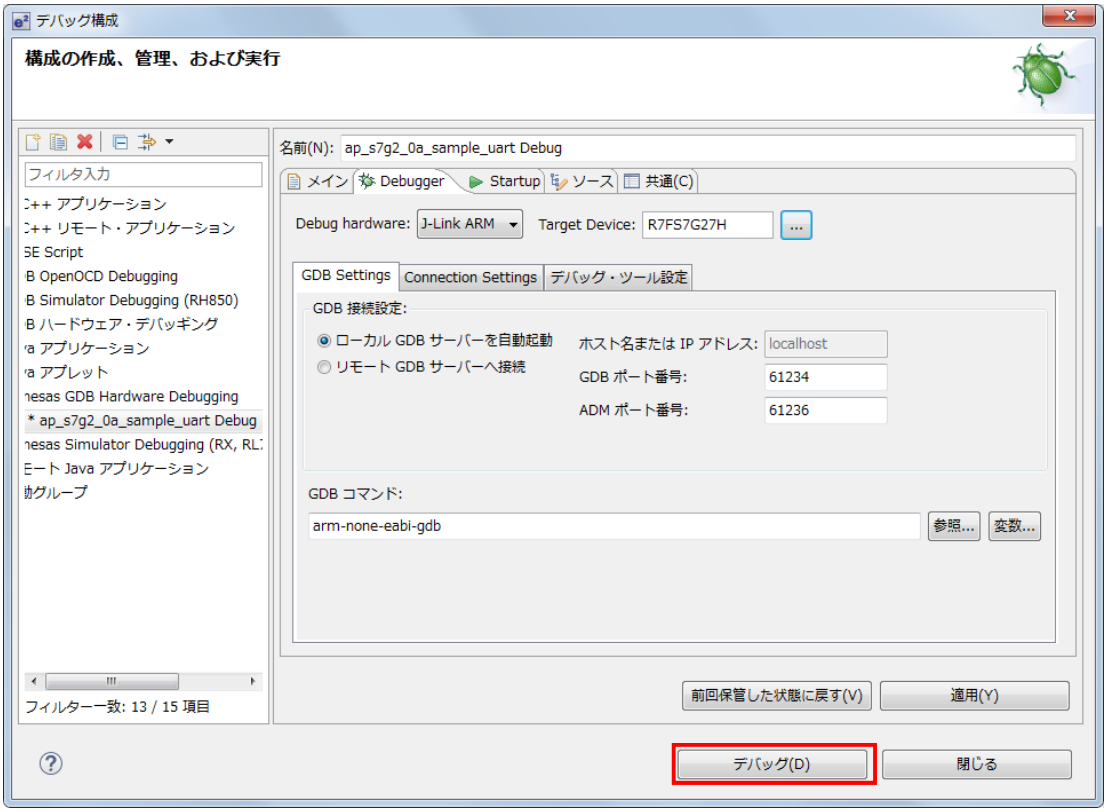
- [名前]：任意の値を入力。
- [プロジェクト]：「参照」ボタンを押し、**プロジェクト**を選択。
- [C/C++アプリケーション]：「プロジェクトの検索」ボタンを押し、**ビルドで生成した elf ファイル**を選択。



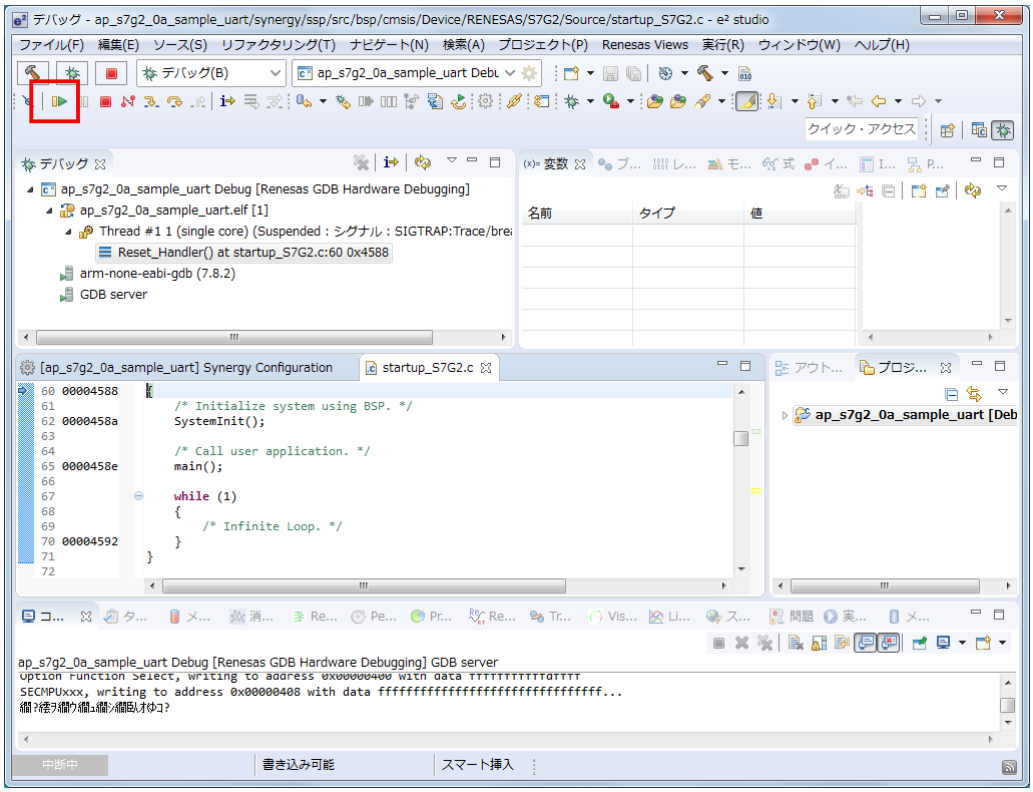
⑦　[Debugger] タブを選択し、 [Debug hardware] に [J-Link ARM] を、 [Target Device] に「**R7FS3A77C**」を設定します。



⑧　[デバッグ] を選択します。



⑨　ボードとの接続が完了したらプログラムを実行し、サンプルプログラムを動作させてください。



3.4　EWSYN を用いたプロジェクトのビルド・デバッグ

本節では、本サンプルプログラムを IAR Embedded Workbench® for Renesas Synergy™（以下、EWSYN）および Renesas Synergy™ Standalone Configurator（以下、SSC）を用いてビルド・デバッグする方法を説明します。

3.4.1　Custom BSP の適用方法

EWSYN でサンプルプログラムのビルド・デバッグを行うためには、SSC に Custom BSP を適用する必要があります。
サンプルプログラム内の「¥sample¥Custom BSP」にある「**AlphaProject.ap_s3a7_0a.1.4.0-ap010000.pack**」を、「<SSC のインストールフォルダ>¥internal¥projectgen¥arm¥Packs」のフォルダへコピーしてください。

例：”C:¥Renesas¥Synergy¥SSC_v6_2_0_R20180102”に SSC をインストールしている場合、下記フォルダに Custom BSP ファイルをコピーしてください。

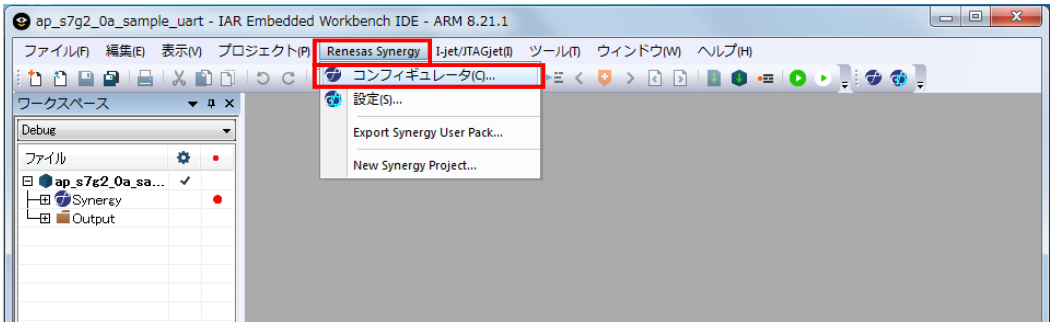
C:¥Renesas¥Synergy¥SSC_v6_2_0_R20180102¥internal¥projectgen¥arm¥Packs

以上で Custom BSP の適用は完了です。

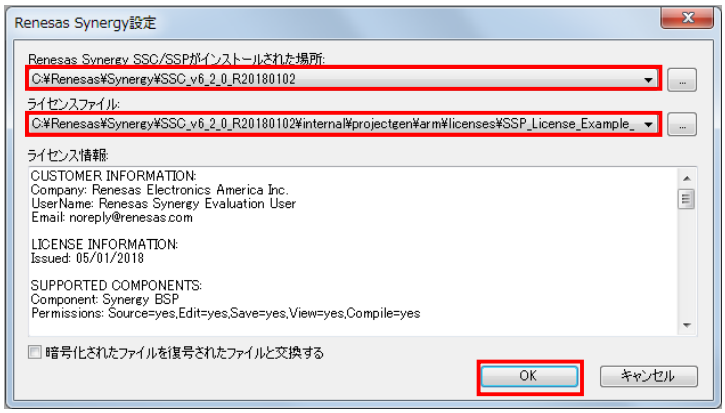
3.4.2　ビルド方法

EWSYN でビルドを行う際は、e2 studio が生成した「synergy」フォルダを削除する必要があります。
事前に「¥sample¥ap_s3a7_0a_sample_wmrp¥synergy」を削除してください。

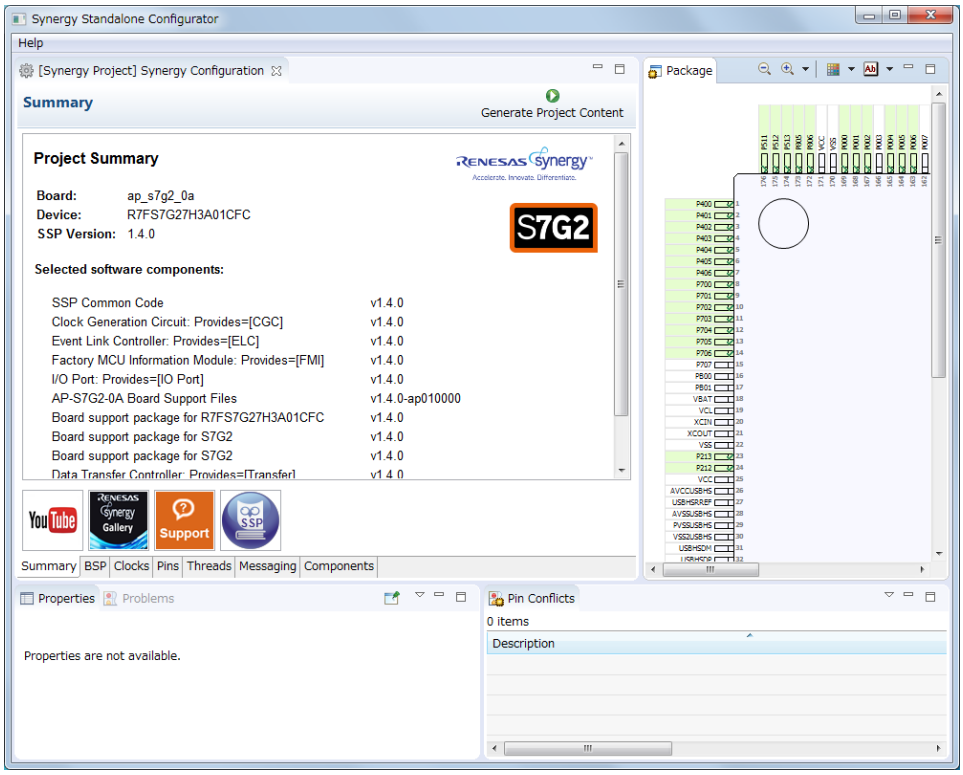
① 「¥sample¥ap_s3a7_0a_sample_wmrp」内のワークスペースファイル「**ap_s3a7_0a_sample_wmrp.eww**」を実行します。
② [Renesas Synergy] – [コンフィギュレータ] を選択して、コンフィギュレータを起動します。



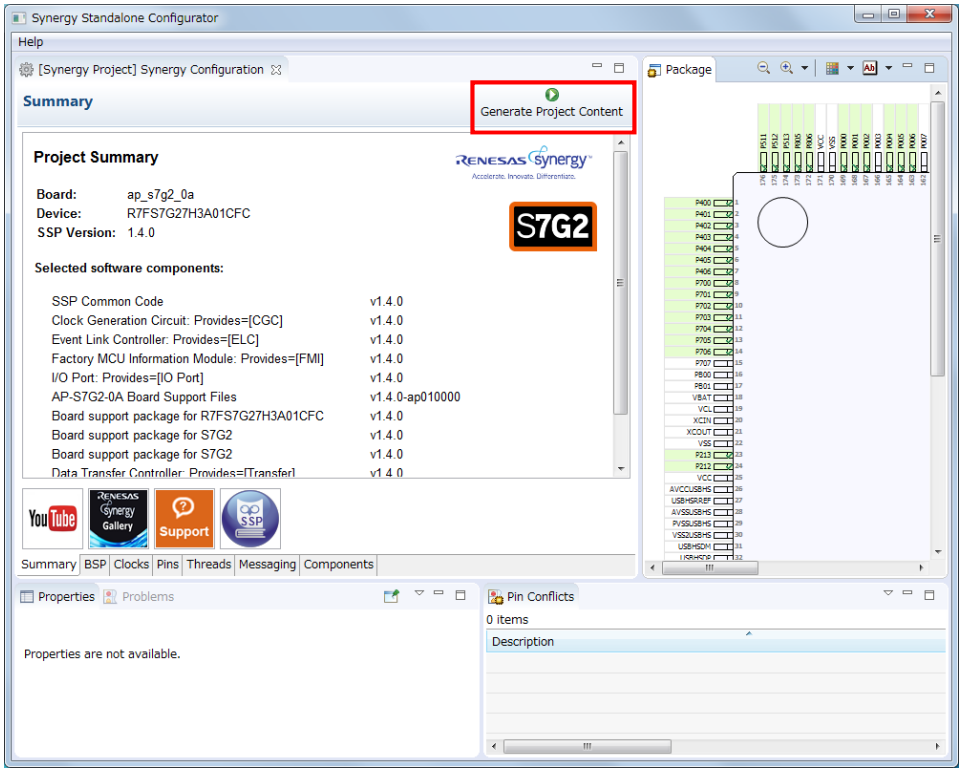
[Renesas Synergy 設定] が表示された場合は、SSC をインストールした場所と、ライセンスファイルを登録します。



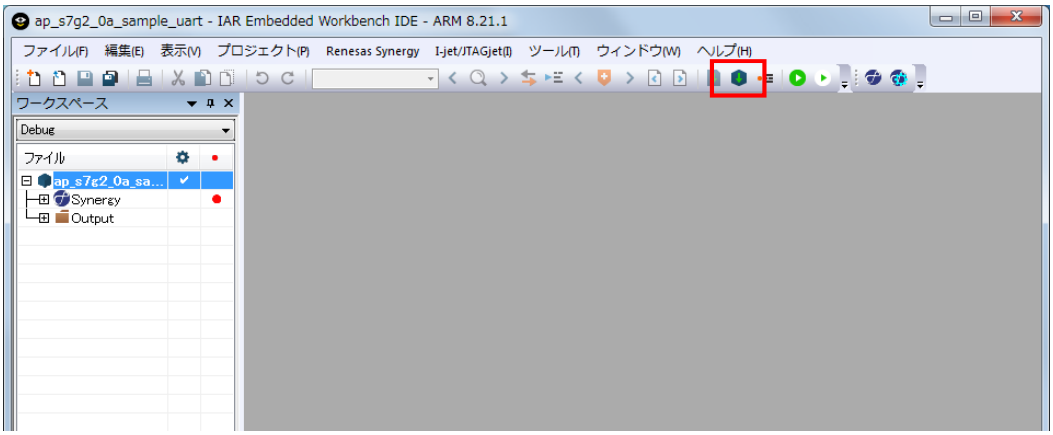
③　SSC が起動します（SSC の起動には、少し時間がかかる場合があります）。



④　[Generate Project Content] を押し、自動作成ファイルを出力して設定をプロジェクトに適用します。



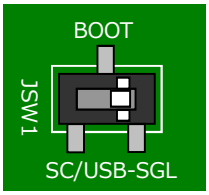
⑤　メイクボタンを押して、メイク（ビルド）を行います。
メイクは、ワークスペースのプロジェクトを右クリックしたメニューからも行うことができます。
（SSC は開いたままでも、終了しても構いません。）



3.4.3　デバッグ方法

EWSYN でデバッグを行う方法を説明します。
デバッグを行う環境・プロジェクトに従って、適切に設定してください。

①　プロジェクトのビルドを行います。
②　ボード上のディップスイッチを以下のように設定してください。

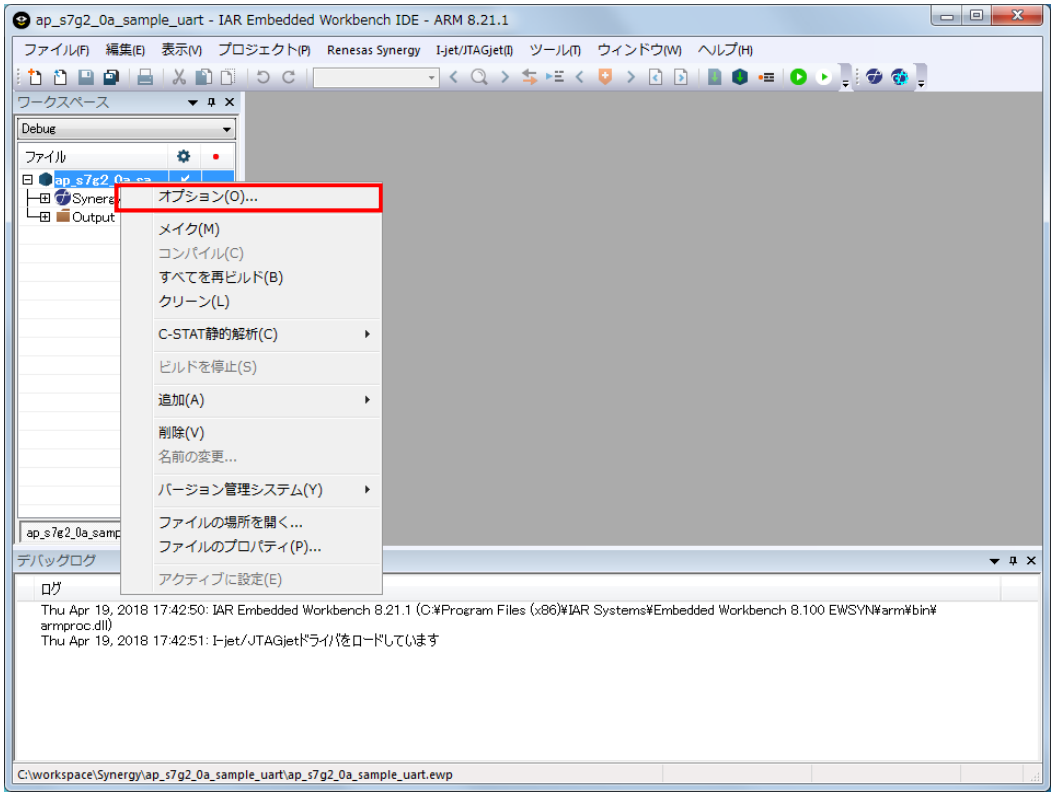


JSW1：SGL

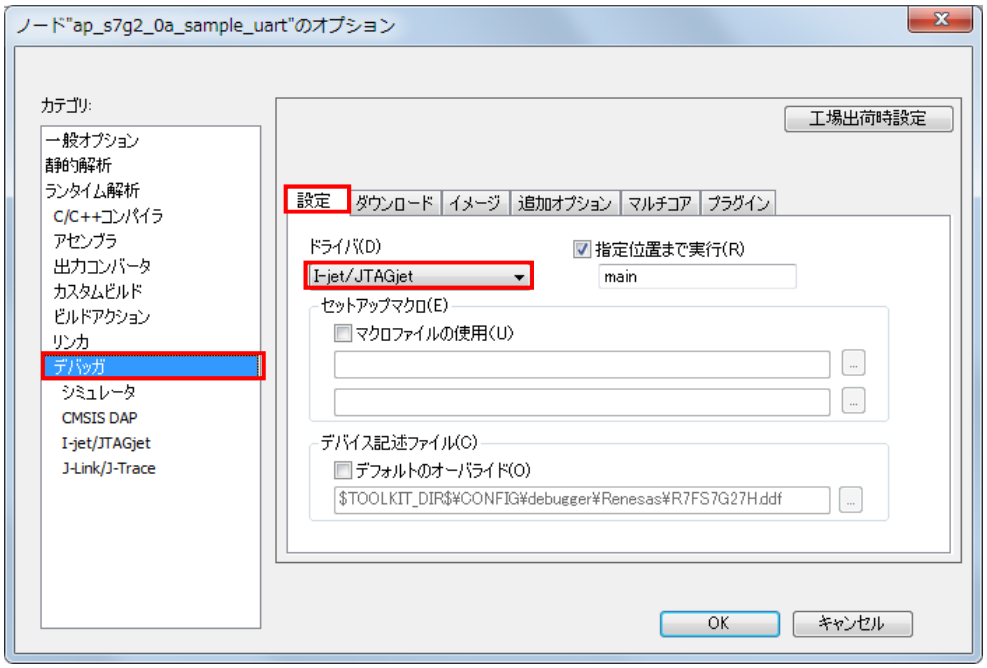
シングルチップモード

③　ボードに電源を投入してください。

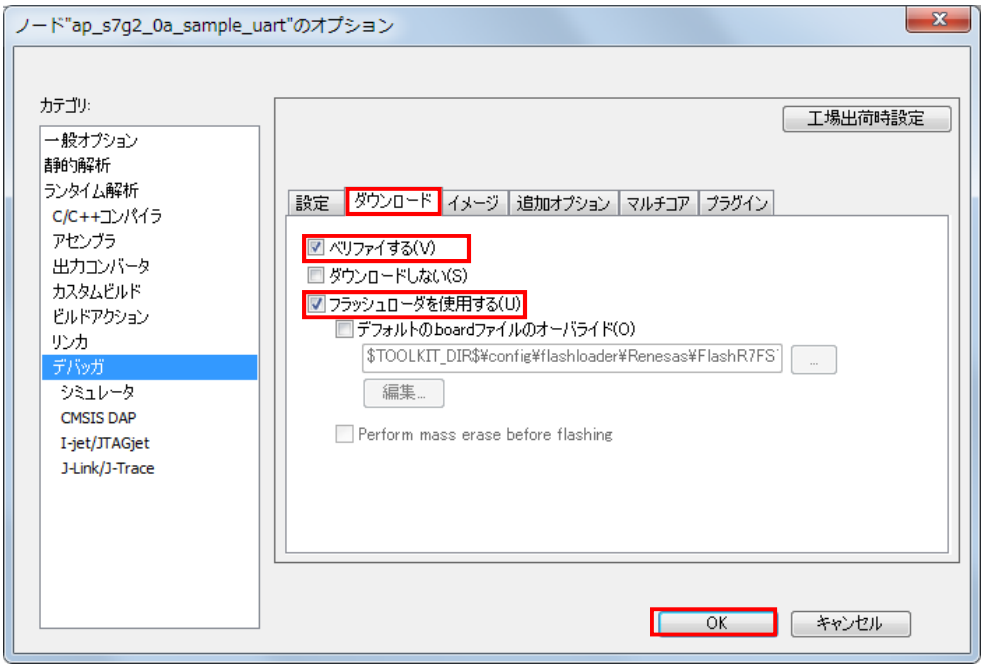
④　ワークスペースのプロジェクトを右クリックし、 [オプション] を選択します。



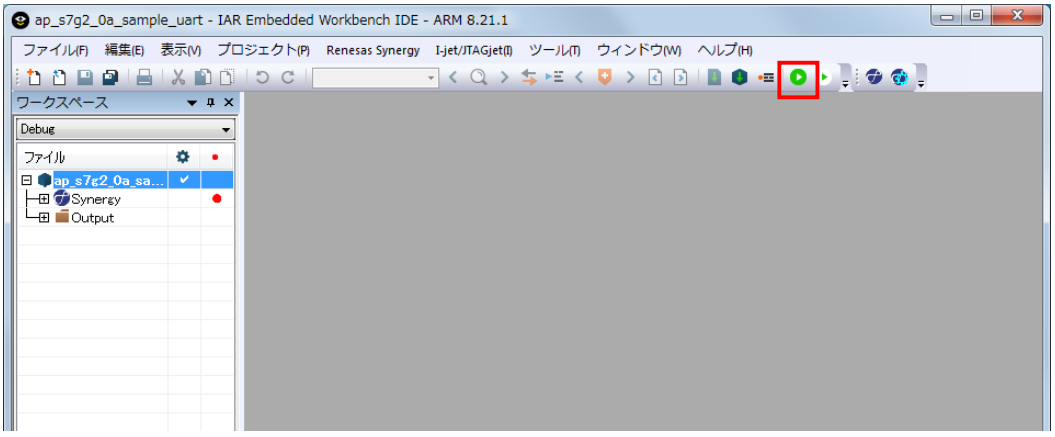
⑤　カテゴリから [デバッガ] を選択し、 [設定] タブの [ドライバ] に [I-jet / JTAGjet] を設定します。



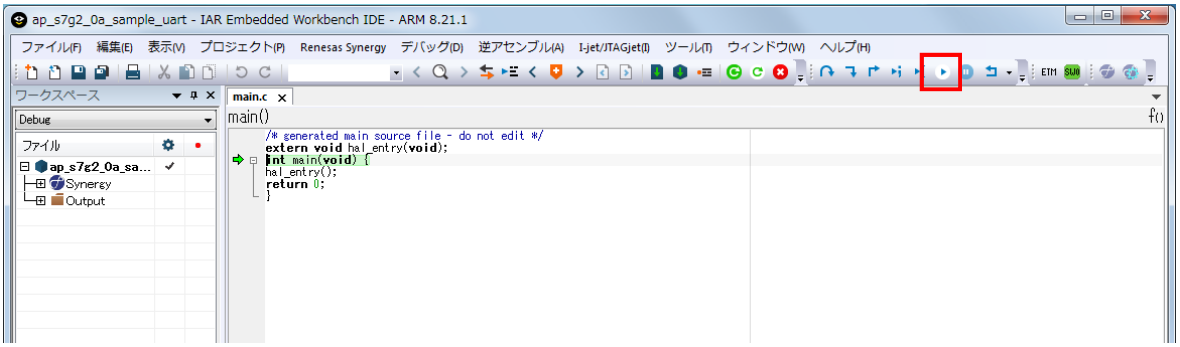
⑥　[タウンロード] タブを開き、 [ベリファイする] と [フラッシュローダを使用する] にチェックを入れます。
設定後、 [OK] を押して終了します。



⑦　ダウンロードしてデバッグのボタンを押して、プログラムをダウンロードしてデバッグを開始します。



⑧　実行ボタンを押すと、プログラムを実行できます。



4. WM-RP-0XS 制御方法

4.1　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。

4.2　SPI

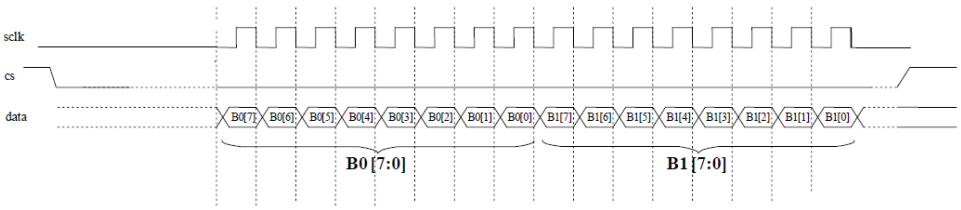
4.2.1　SPI 仕様

WM-RP-0xS の SPI 仕様を以下に記します。

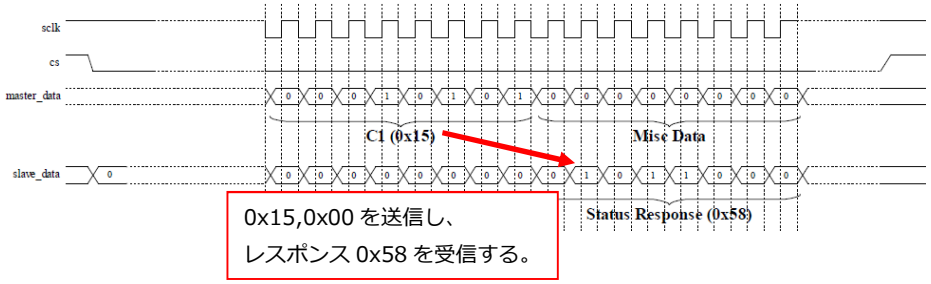
機能	仕様
通信方式	SPI 4 線式 SCK,SCS,MOSI,MISO
SPI クロック	25MHz(MAX)
データ	ホスト CPU が SPI マスター、MSB ファースト
割り込み	INTR 信号 ※ハイレベル割り込みです。 エッジ割り込みでは、ありません。

Table 4.2-1　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 から受信するために「0x00」（画像では、MiscData」）を送信しています。



4.2.2　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。

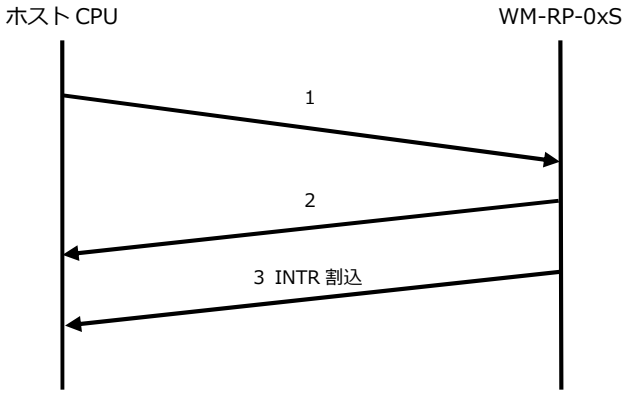


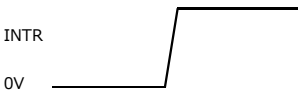
Fig 4.2-1　SPI インタフェース制御フロー

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。

4.2.3　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。



WM-RP-0xS が以下の状態のとき、INTR 割り込みが発生します。

- 無線通信の相手より、データを受信したとき。
- 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とき。
- WM-RP-0xS 内部 SPI 受信メモリバッファがフルとなったとき。
  ※　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行ってください。
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ください。
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。

4.2.4　無線データ送信処理

無線データ送信処理を行う場合、必ず rsi_intHandler()関数を呼び出し rsi_strIntStatus.bufferFull を確認してください。
これは、WM-RP モジュール内部バッファが FULL であるかを確認しています。
もし FULL となった場合は、FULL が解除されてから送信を行うようにプログラムしてください。

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。

```
92              rsi_intHandler();
93              if(rsi_strIntStatus.bufferFull == RSI_TRUE)
94      //      if(rsi_checkBufferFullIrq() == RSI_TRUE)
95              {
96                      return RSI_BUFFER_FULL;
97              }
```

4.3　Redpine Signals 社提供のライブラリ

「¥ap_s3a7_0a_sample_wmrp¥src¥wifi¥API_Lib」内のファイルは Redpine Signals 社が用意したライブラリです。
本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、「RS9110-N-11-22_24_28_SPI_API_Library_Manual_v1.7.pdf」を参照してください。
また、合わせて「RS9110-N-11-22_24_28-Software_PRM_v4.14.pdf」も参照してください。
これらのデータシートは、WM-RP 製品ページより「データシート」（parts.zip）としてダウンロードできます。
（「parts.zip」のフォルダ構成は、フォルダ内「readme.txt」にてご確認ください。）

※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ください。

4.3.1　各種変数等

＜ 1 ＞　rsi_api 構造体

「¥src¥wifi¥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ため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¥wifi¥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 80 行目にて、グローバル変数として「rsi_api rsi_strApi」のように定義して使用しています。

＜ 2 ＞　rsi_uCmdRsp 構造体

「¥src¥wifi¥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ための構造体であり、
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ように作られています。

サンプルプログラムでは、「wm_rp_04s.c」ファイルの 56 行目にて、static なグローバル変数として「static volatile rsi_uCmdRsp uCmdRspFrame」のように定義して使用しています。

例）「band」コマンドを実行する場合　以下のような処理となります。

例		
	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」の「2.1.8 Read Packet data structure (From module):」を参照してください。

＜３＞　タイマ用変数

「wm_rp_04s.c」ファイルの 72 行目～75 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¥wifi¥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、変数「uint32 rsi_spiTimer1」「uint32 rsi_spiTimer2」「uint32 rsi_spiTimer3」をサンプルプログラムのようにグローバル変数として定義してください。

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、殆どの場合タイムアウト処理です。
例えば、「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_global.h」ファイルの 151 行目の
マクロ「#define RSI_RESPONSE_TIMEOUT(A)」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 78 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。

4.3.2　無線通信用の主なコマンドファイル

※「¥src¥wifi¥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します。
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()関数を呼びます。
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ください。

< 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 ・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 割り込みが入ったとき 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()」関数を呼ぶことで行っています。 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より、無線データか、或いは WM-RP-0xS のレスポンスデータであるか、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。 接続先が切断したことを知るために、常にこの変数を監視するようにプログラムしてください。

< 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 度実行してください。
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

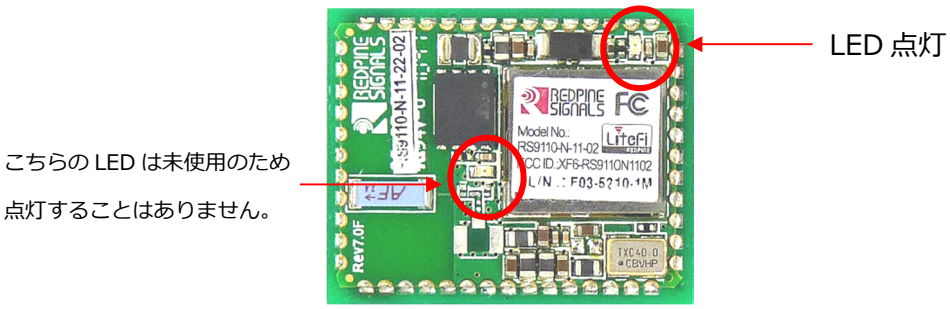
コマンド説明	
説明	WM-RP-0xS に対して実行する各種設定、コマンド等の値を変数に設定します。
コマンドファイル名	rsi_config_init.c（「¥src¥wifi¥Applications¥MCU」フォルダ内）
使用方法	rsi_init_struct()関数を呼び出します。
パラメータ説明	rsi_api 構造体で定義した変数ポインタを指定
レスポンスコード	-
レスポンス詳細	-

< 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¥wifi¥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」にしてください。
パラメータ説明	-
レスポンスコード	-
レスポンス詳細	-

5. 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。



6. サンプルプログラムの設定変更に関して

6.1　ネットワークの設定

6.1.1　ポート番号

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config_init.c」ファイル
38 行目～40 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を変更してください。

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

6.1.2　TCP or UDP ,　Server or Client

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config_init.c」ファイル
42 行目～52 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を変更してください。
※この定義に指定する値は、「¥src¥wifi¥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」定義を使用してください。
※本サンプルプログラムでは、「¥src」フォルダ内「common_app.h」ファイルの 35 行目～38 行目の定義を
　変更することで指定しています。
　詳細は「3.1.4　TCP/IP 通信動作モード選択方法」および「3.1.8　UDP 通信動作モード選択方法」を参照してください。

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

6.1.3　インフラストラクチャ or アドホック

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル
44 行目～48 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を変更してください。
※この定義に指定する値は、「¥src¥wifi¥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」定義を使用してください。
※本サンプルプログラムでは、「¥src」フォルダ内「common_app.h」ファイルの 40 行目～42 行目の定義を
　変更することで指定しています。
　詳細は「3.1.4　TCP/IP 通信動作モード選択方法」および「3.1.8　UDP 通信動作モード選択方法」を参照してください。

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

6.1.4　WM-RP-0xS の IP アドレス

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル
63 行目～64 行目にて、WM-RP-0xS の IP アドレスを設定しています。
必要に応じて以下の定義を変更してください。

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

6.1.5　ゲートウェイ

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル
66 行目～67 行目にて、ゲートウェイのアドレスを設定しています。
必要に応じて以下の定義を変更してください。

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

6.1.6　接続先の IP アドレス

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル
69 行目～70 行目にて、接続先の IP アドレスを設定しています。
必要に応じて以下の定義を変更してください。

- #define RSI_TARGET_IP_ADDRESS　　接続先の IP アドレス

6.1.7　ネットマスク

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル
72 行目～73 行目にて、ネットマスクのアドレスを設定しています。
必要に応じて以下の定義を変更してください。

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

6.1.8　無線アクセスポイントセキュリティタイプ

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル
75 行目～77 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しています。
必要に応じて以下の定義を変更してください。
※この定義に指定する値は、「¥src¥wifi¥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」定義を使用してください。

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

6.1.9　無線アクセスポイントセキュリティキー

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル
80 行目～86 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しています。
必要に応じて以下の定義を変更してください。

- #define RSI_PSK　　無線アクセスポイントセキュリティキー

6.1.10　無線アクセスポイントスキャン SSID

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル
96 行目～97 行目にて、無線アクセスポイントのスキャン SSID を設定しています。
必要に応じて以下の定義を変更してください。

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

6.1.11　無線アクセスポイントスキャン CH

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル
101 行目にて、無線アクセスポイントのスキャン CH を設定しています。
必要に応じて以下の定義を変更してください。

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

6.1.12　無線アクセスポイント参加(Join)SSID

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル
103 行目～104 行目にて、無線アクセスポイントの参加(Join)SSID を設定しています。
必要に応じて以下の定義を変更してください。

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

6.1.13　DHCP 使用 or 未使用

「¥src¥wifi¥Applications¥MCU」フォルダ内「rsi_config.h」ファイル
148 行目～150 行目にて、DHCP 使用 or 未使用を設定しています。
必要に応じて以下の定義を変更してください。
※この定義に指定する値は、「¥src¥wifi¥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」定義を使用してください。

- #define RSI_IP_CFG_MODE　　DHCP 使用 or 未使用

6.2　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 S3A7 を使用しました。
WM-RP-0xS を使用する環境に合わせて、以下の処理を変更してください。

6.2.1　エンディアン

「¥src¥wifi¥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しています。
必要に応じて以下の定義の変更をしてください。

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

6.2.2　WM-RP-0xS レスポンス待ちタイムアウト処理定義

「¥src¥wifi¥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ください。
※通常、アプリケーションは WM-RP-0xS からのレスポンスを待機します。
　この定義では、最大待ち時間（タイムアウト）を環境に合わせて設定してください。

- #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。
　　使用する制御 CPU に合わせて適宜変更してください。

6.2.3　SPI

「¥src¥wifi¥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ください。
また、「¥src¥wifi¥API_Lib」フォルダ内「rsi_hal.h」ファイル
41 行目～44 行目にて、SPI を使用するため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ください。

- #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

6.2.4　割り込み

「¥src¥wifi¥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ください。

※本サンプルプログラムでは、ポーリング処理（ポート入力）により 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ことで、INTR 信号がローレベルかを確認しております。
あわせて、「¥src¥wifi¥API_Lib」フォルダ内「rsi_interrupt.c」ファイルも参考にしてください。
具体的な割り込みの処理を記述しております。

6.2.5　ハードウェアタイマ

「¥src¥wifi¥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ください。

6.2.6　I/O ポート

「¥src¥wifi¥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ください。
※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内容についての質問等のサポートは一切受け付けておりませんのでご了承ください。
・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ください。
・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株式会社への問い合わせはご遠慮ください。
・本文書の内容に基づき、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、弊社では一切責任を負いませんのでご了承ください。
・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

・Renesas Synergy™およびS3A7 は、ルネサスエレクトロニクス株式会社の登録商標、商標または商品名称です。
・e2 studio は、ルネサスエレクトロニクス株式会社の登録商標、商標または商品名称です。
・Synergy Software Package は、ルネサスエレクトロニクス株式会社の登録商標、商標または商品名称です。
・Renesas Synergy™ Standalone Configurator は、ルネサスエレクトロニクス株式会社の登録商標、商標または商品名称です。
・IAR Embedded Workbench® for Renesas Synergy™は、IAR システムズ株式会社の登録商標、商標または商品名称です。

・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
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