RX671 グループ

RX671 タッチキーと LCD を用いた OTA 対応フラットパネル HMI PoC

要旨

RX671 タッチキーと LCD を用いた OTA 対応フラットパネル HMI PoC（以下、RX671 PoC）は、静電容量式タッチセンサによる HMI を備えた IoT 機器の開発に最適なキットです。

本アプリケーションノートでは、RX671 PoC を使って、タッチ機能及びシリアル接続 LCD による LCD 表示を実現する HMI ソリューションをご紹介します。また、クラウドに接続して OTA によるファームウェアアップデートを行う機能も合わせてご紹介します。

本アプリケーションノートで説明するサンプルプログラムは、以下のライブラリを使用して構成しています。

- LCD 表示　：組込用 GUI ソフトウェア　emWin　（以下、emWin と記載）

動作確認デバイス

RX671 グループ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様に合わせて変更し、十分評価してください。

動作確認ツール

RX671 PoC

セキュリティに関する詳細は「ルネサス MCU ファームウェアアップデートの設計方針（R01AN5548）」を参照してください。

目次

1. 概要

本アプリケーションノートは、RX671 PoC の構成と操作方法を説明します。RX671 PoC は、LCD（240×320）、タッチボタンとタッチスライダーを搭載し、電子レンジの設定や表示の制御を想定したデモ動作を行います。また、RX671 PoC は、クラウドから OTA によりプログラムを書き換えることが出来ます。

RX671 PoC 全体図を以下に示します。

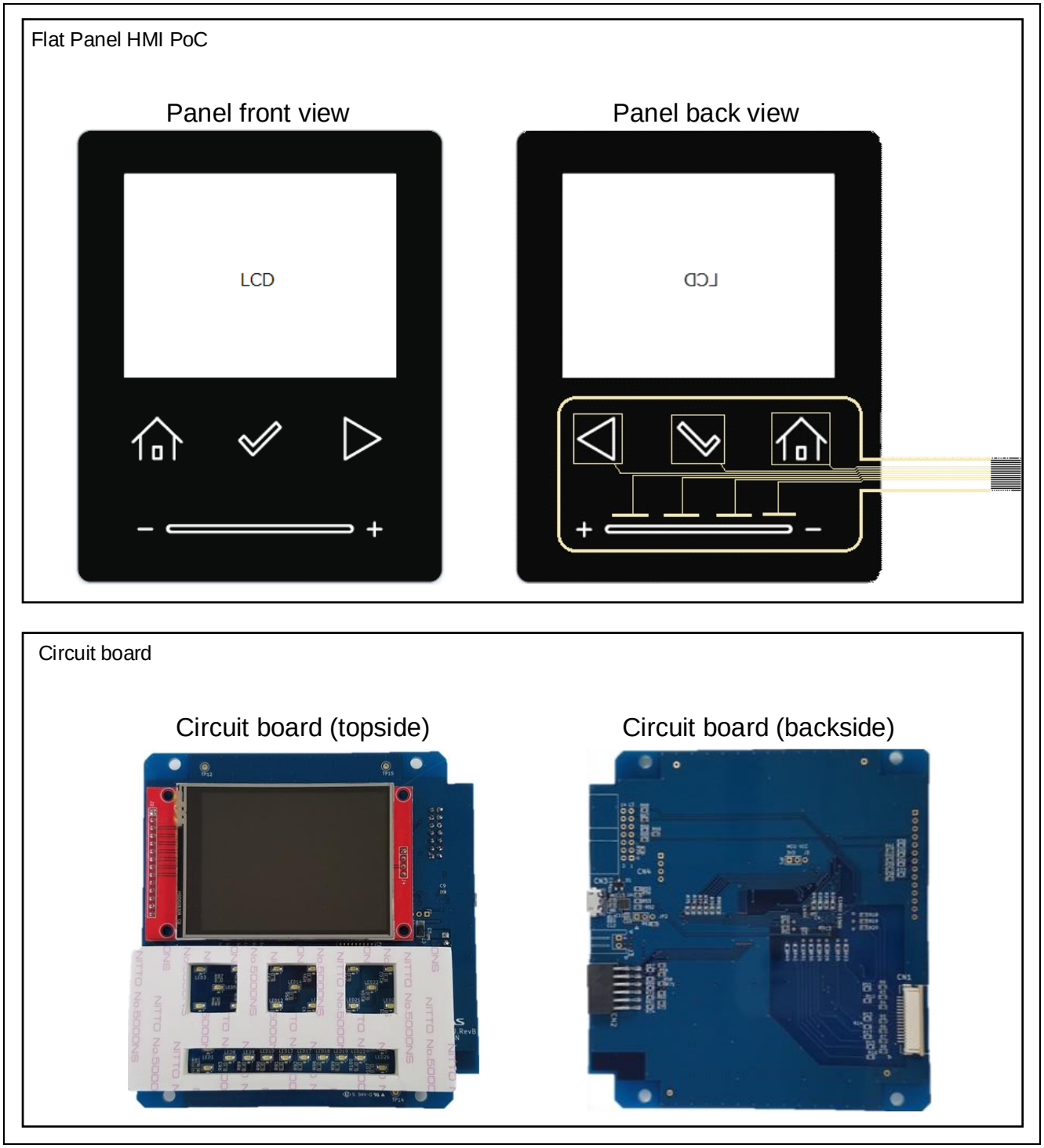


図 1-1　RX671 PoC 全体図

システムの構成を以下に示します。

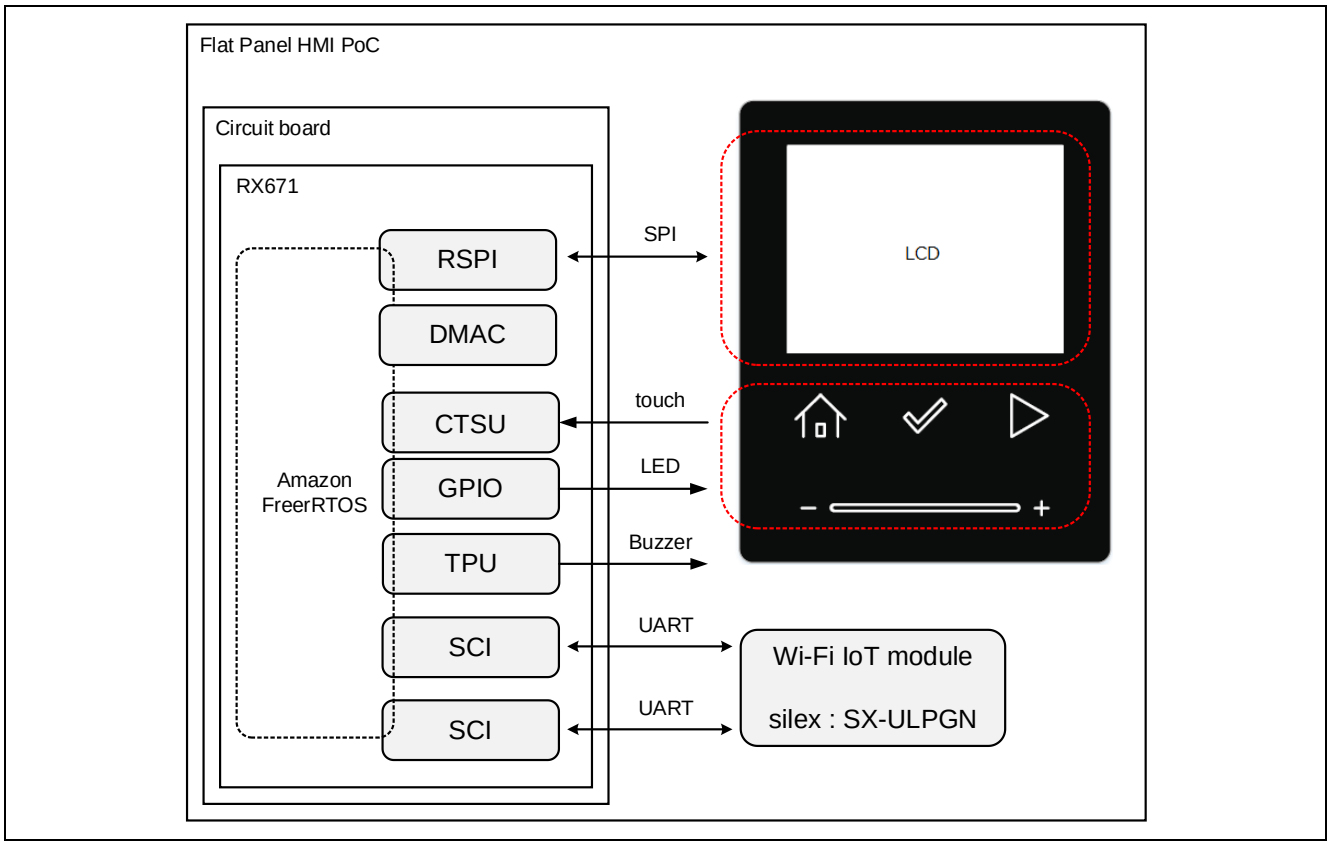


図 1-2　システムの構成

ソフトウェアの構成を以下に示します。

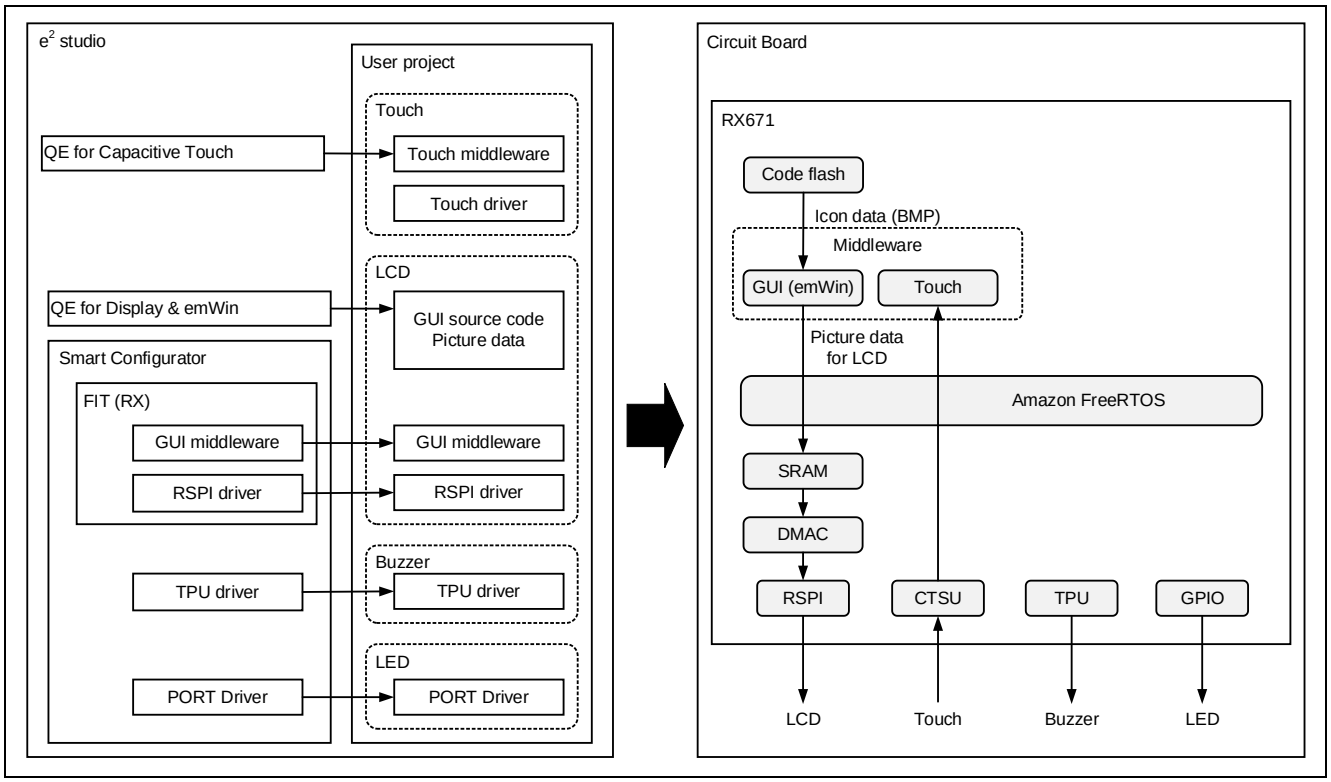


図 1-3　ソフトウェア構成

2. 動作確認条件

本サンプルプログラムは、下記の条件で動作を確認しています。

表 2-1　動作確認条件

項目	内容
使用マイコン	R5F5671EHDFM（RX671　グループ）
動作周波数	動作周波数 (ICLK)：120MHz 周辺動作周波数 (PCLKB)：60MHz
動作電圧	3.3 V
総合開発環境	ルネサスエレクトロニクス e^2 studio Version 2023-01 (23.1.0)
C コンパイラ	ルネサスエレクトロニクス C/C++ Compiler Package for RX Family V3.05.00
	コンパイラオプション 統合開発環境のデフォルト設定が適用されます。
スマート・コンフィグレータ	V2.16.0
ボードサポートパッケージ（r_bsp）	V7.20
エンディアン	Little Endian
動作モード	Single chip mode
プロセッサモード	Super visor mode
サンプルコードバージョン	V1.00
エミュレータ	E2 Emulator Lite

表 2-2　動作確認条件（LCD、Wi-Fi）

項目	内容
LCD モジュール	2.8 TFT SPI 240×320 serial port module
Wi-Fi モジュール	Wi-Fi-Pmod-Expansion-Board（RTK00WFMX0B00000BE）

3.　サンプルプログラム

3.1　デモ画面のフローチャート

本サンプルプログラムのデモ画面フローチャートを以下に示します。各画面の詳細は 5 章を参照してください。

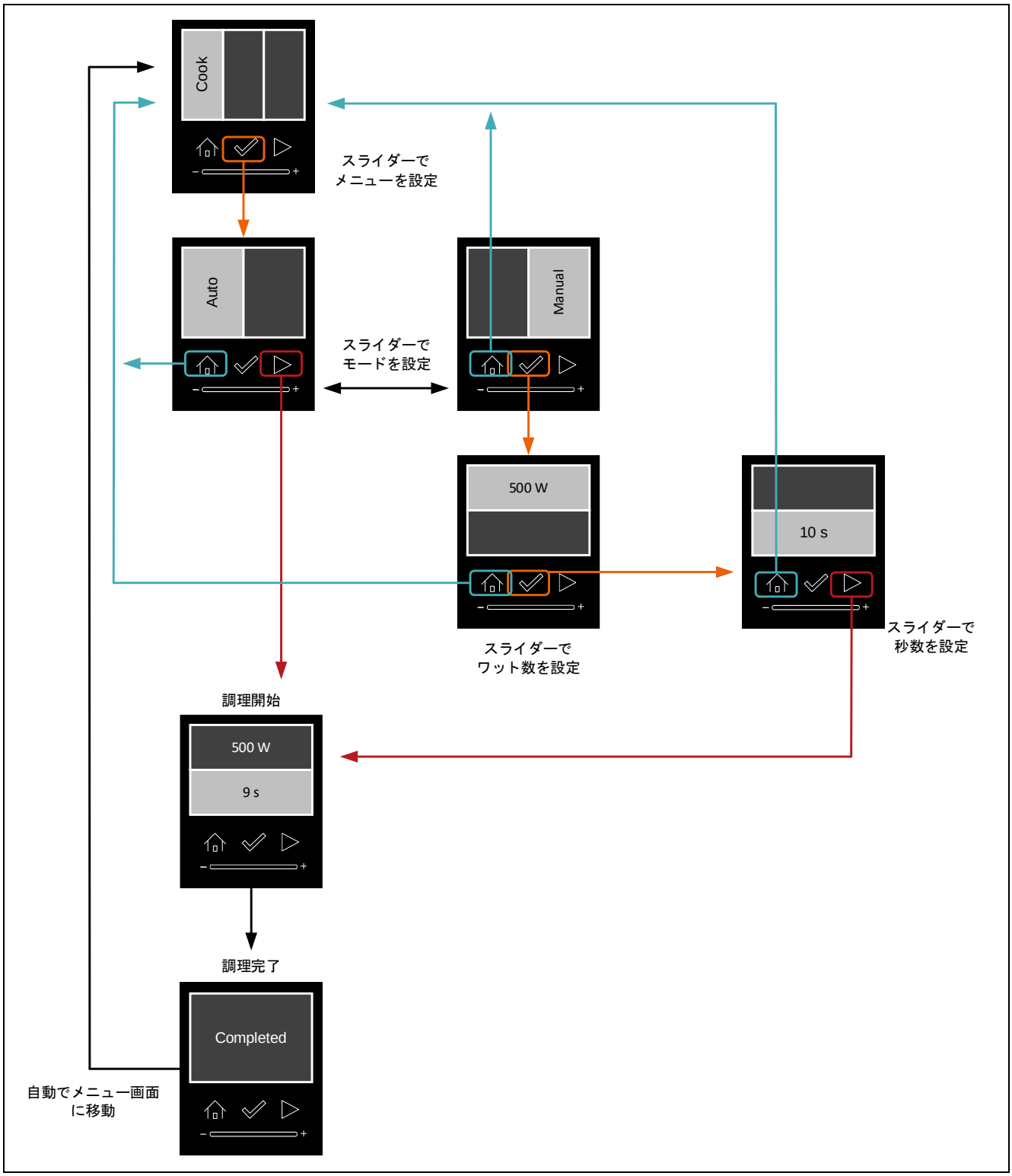


図 3-1 デモ画面のフローチャート（Cook）

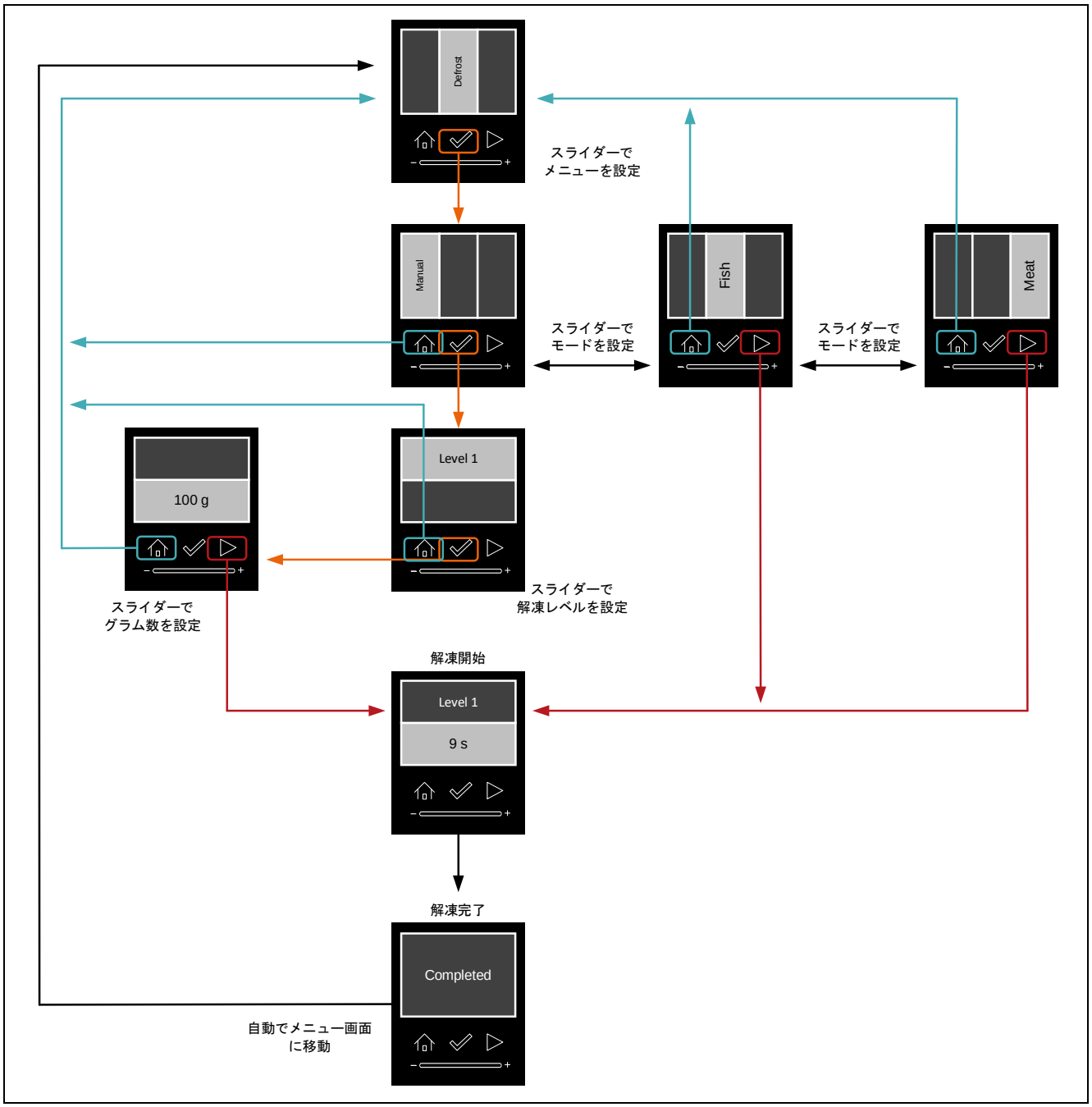


図 3-2 デモ画面のフローチャート（Defrost）

Recipe は、ファームウェアが ver.0.90 の場合使用できません。

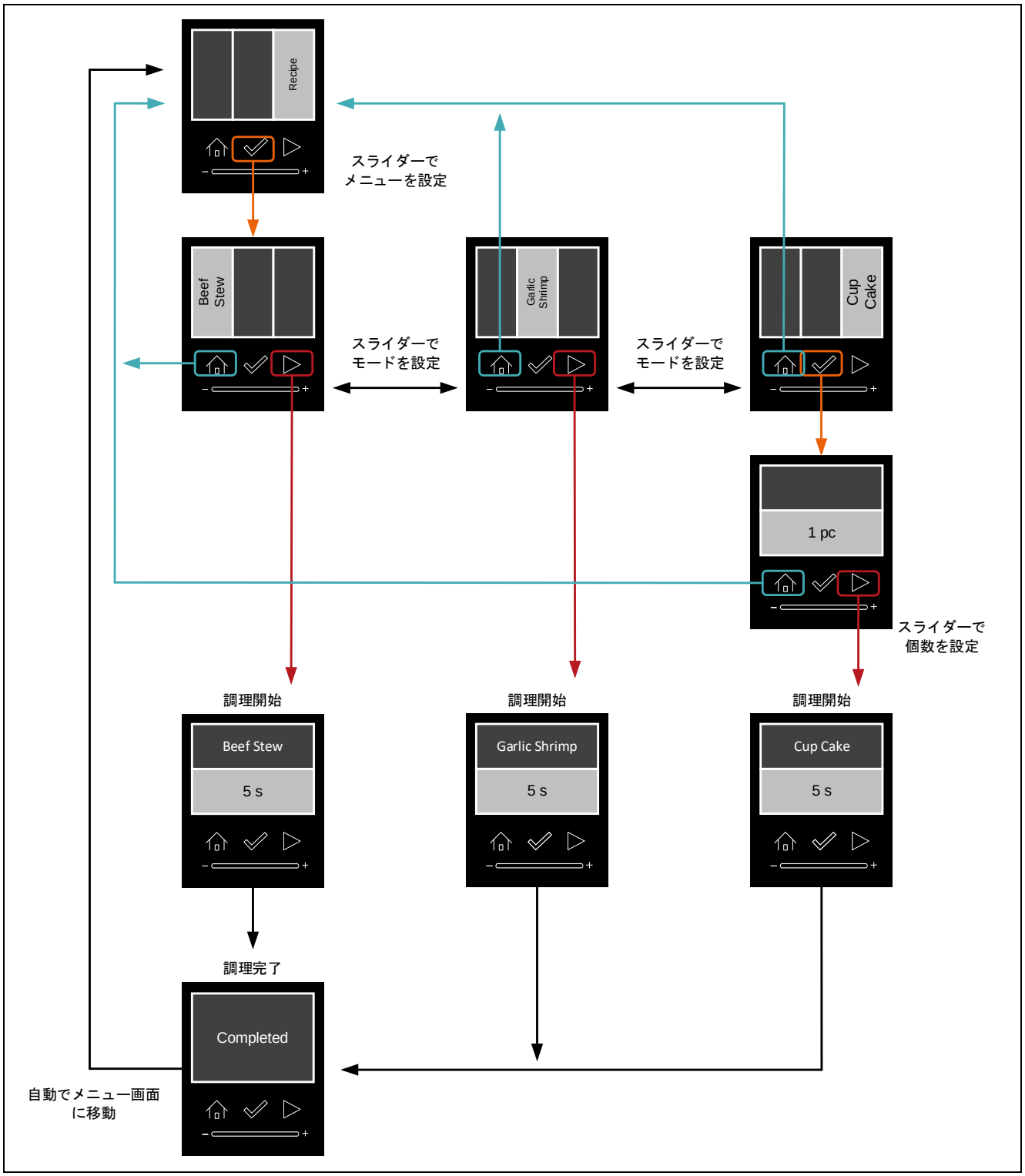


図 3-3 デモ画面のフローチャート（Recipe）

3.2　フローチャート

3.2.1　LCD 制御フローチャート

LCD 制御の全体フローチャートを以下に示します。

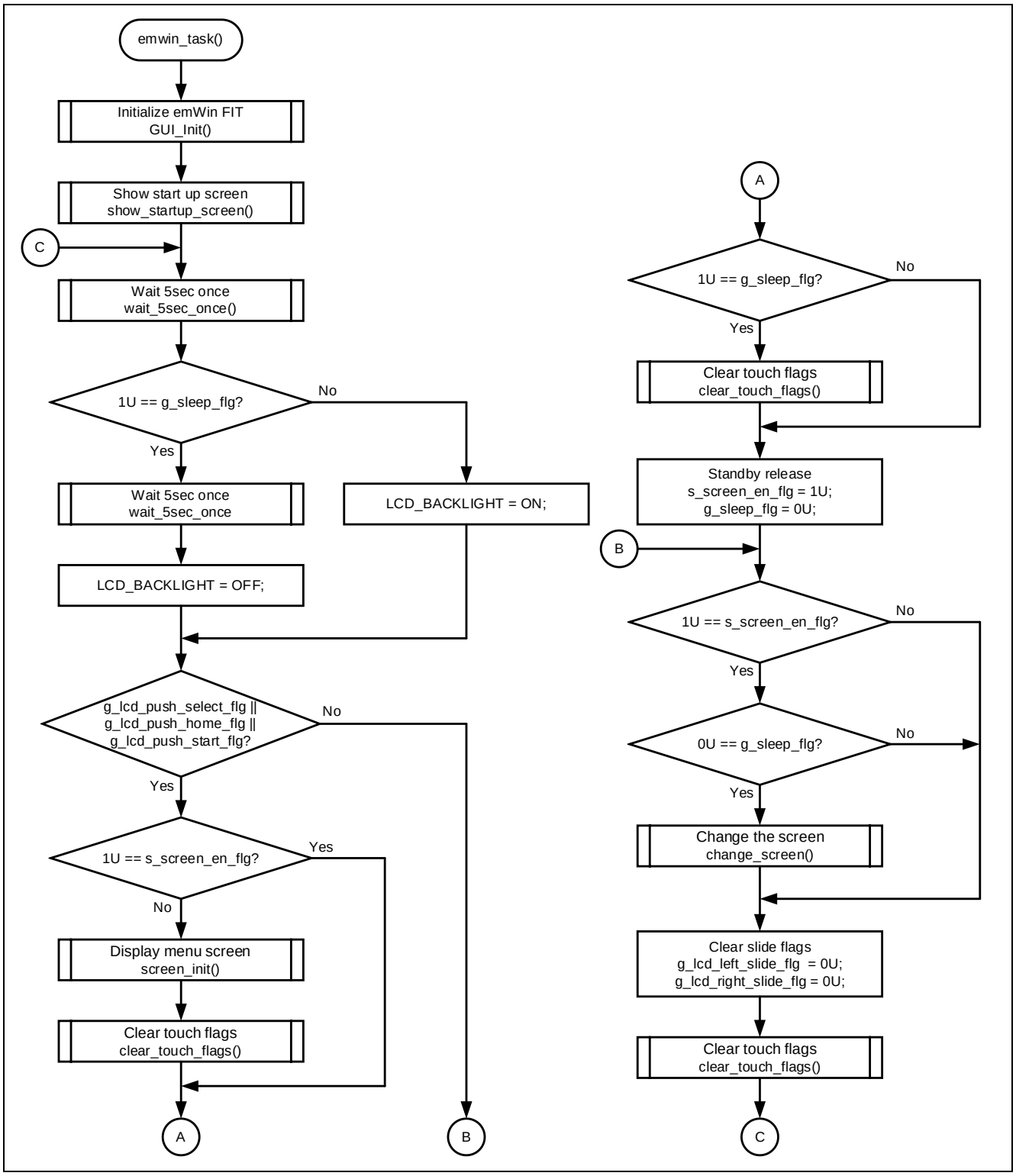


図 3-4 LCD 制御のフローチャート

3.2.2　タッチ操作時の処理

タッチ操作時のフローチャートを以下に示します。

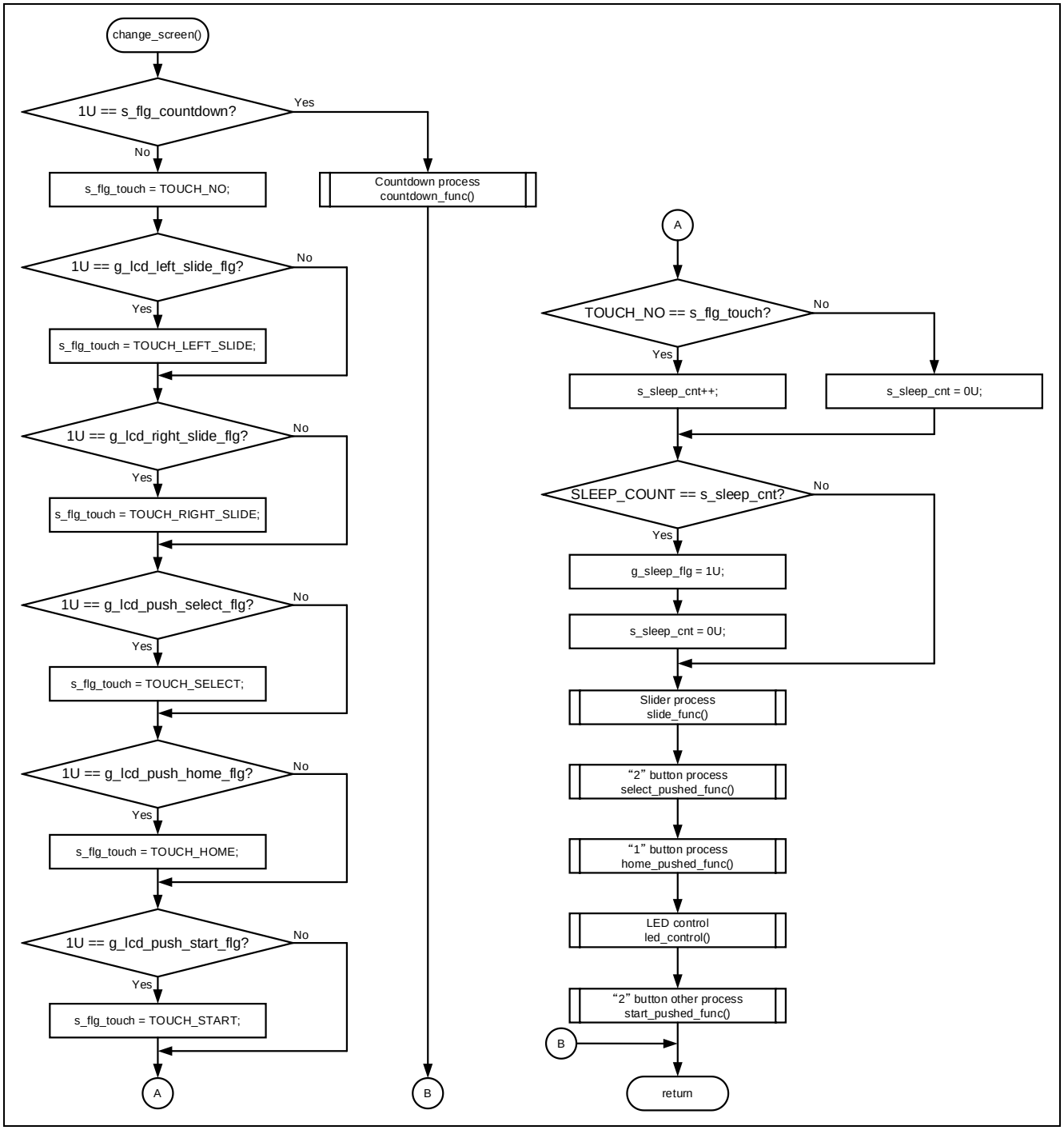


図 3-5 タッチ操作時の処理のフローチャート

3.2.3　タッチスライダー操作時の処理

タッチスライダー操作時のフローチャートを以下に示します。

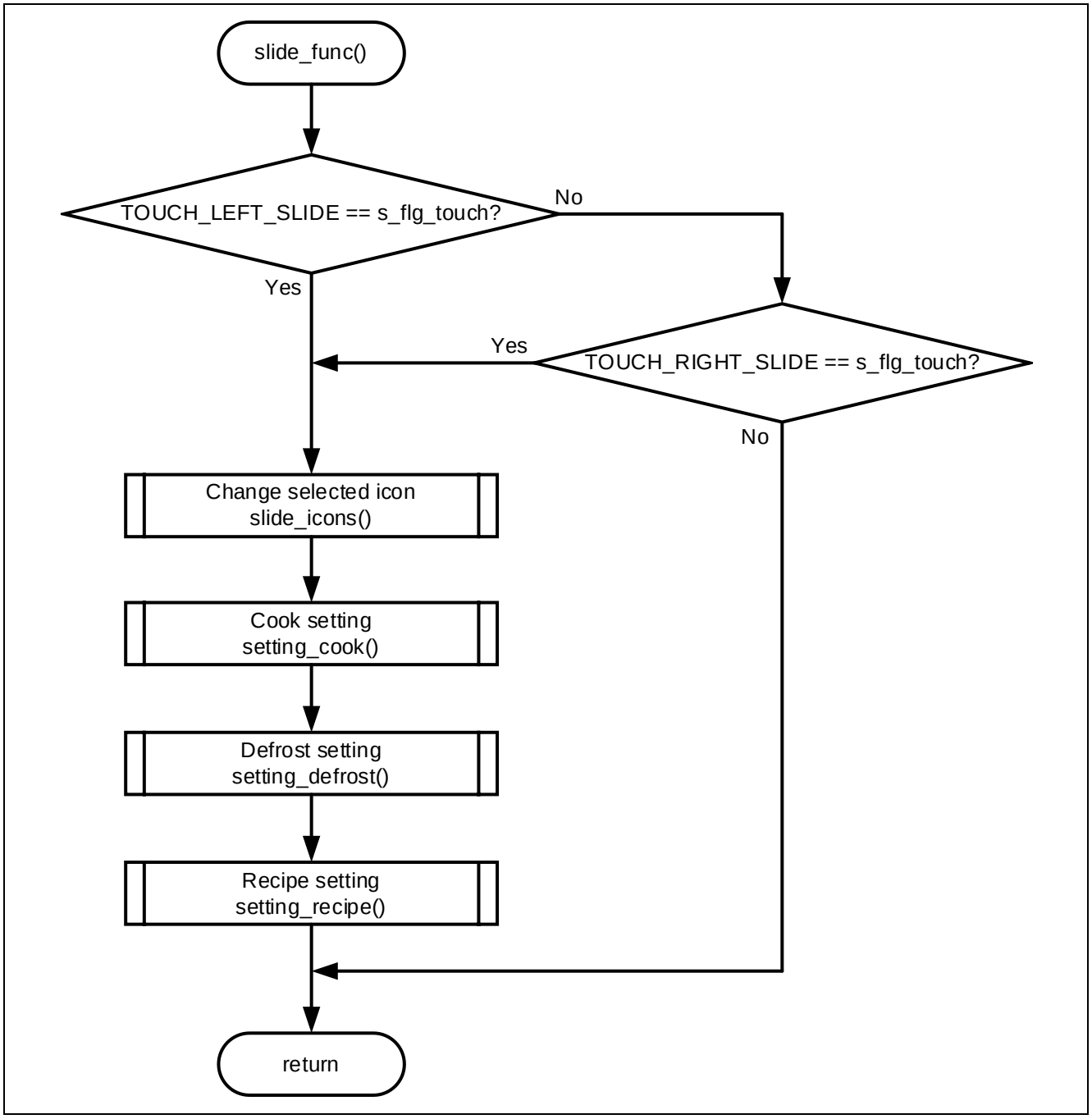


図 3-6 タッチスライダー操作時のフローチャート

3.2.4　「home」ボタンタッチ時の処理

「home」ボタンタッチ時のフローチャートを以下に示します。

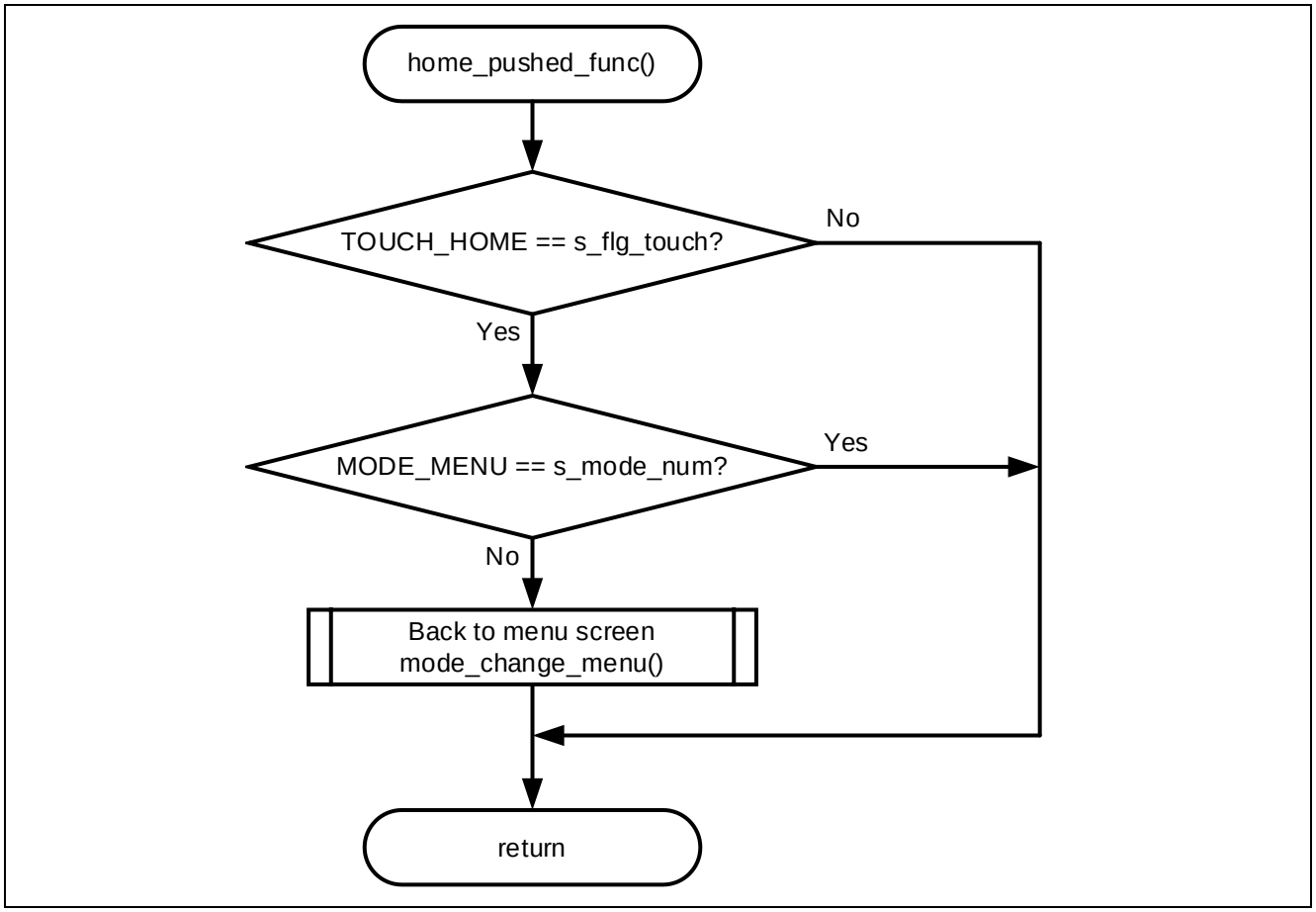


図 3-7 「home」ボタンタッチ時のフローチャート

3.2.5　「select」ボタンタッチ時の処理

「select」ボタンタッチ時のフローチャートを以下に示します。

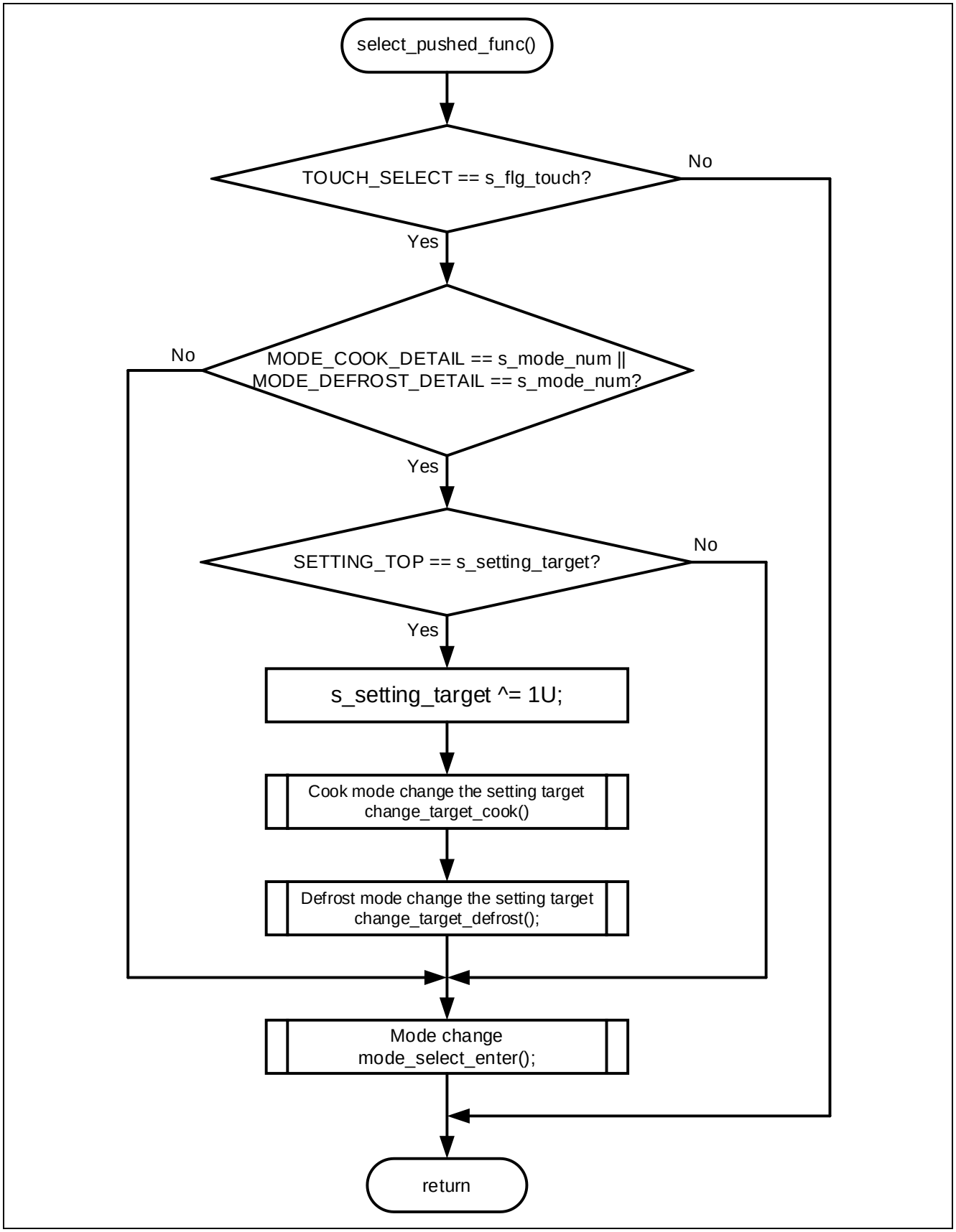


図 3-8 「select」ボタンタッチ時のフローチャート

3.2.6　「start」ボタンタッチ時の処理

「start」ボタンタッチ時のフローチャートを以下に示します。

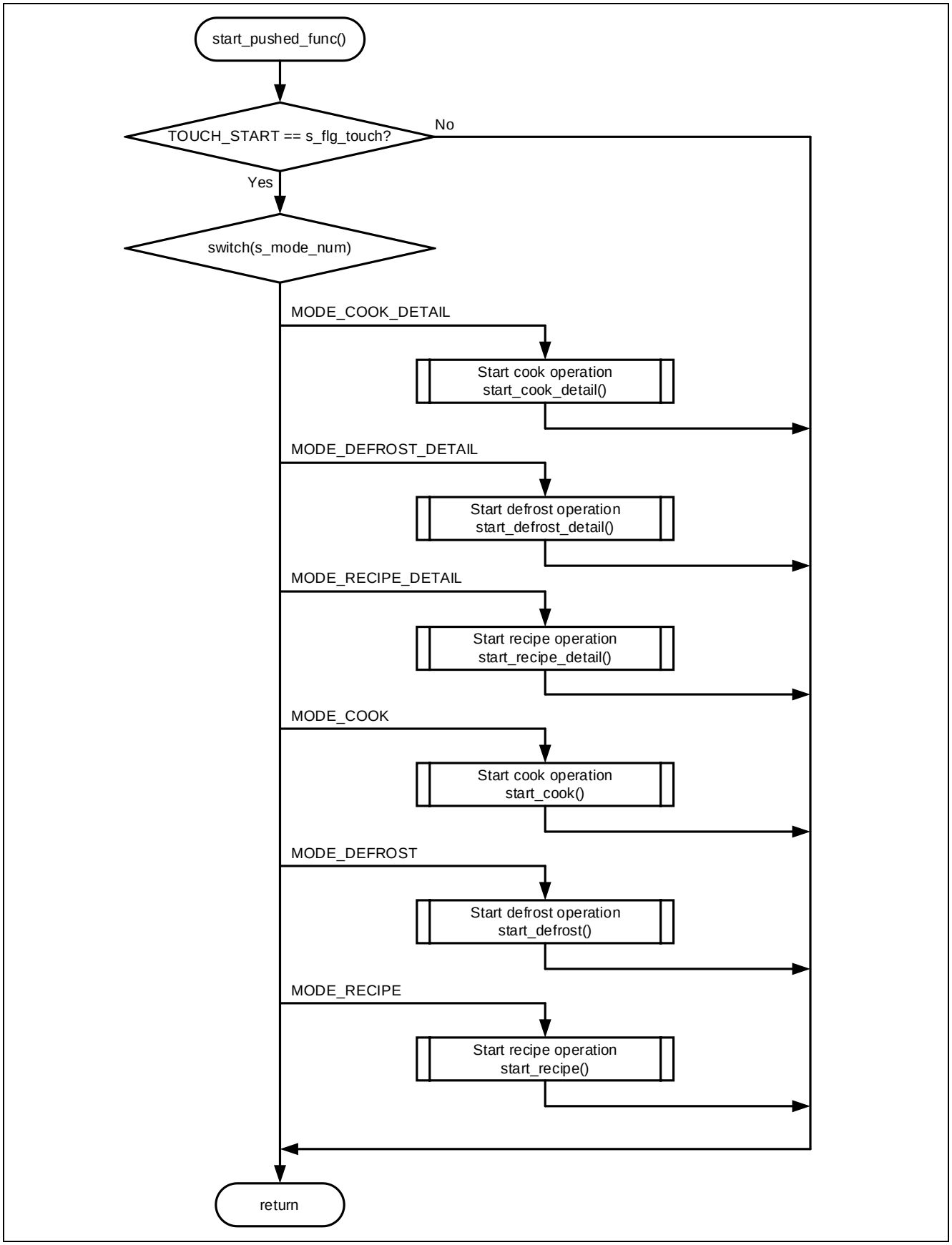


図 3-9 「start」ボタンタッチ時のフローチャート

3.2.7　調理中の処理

調理中のフローチャートを以下に示します。

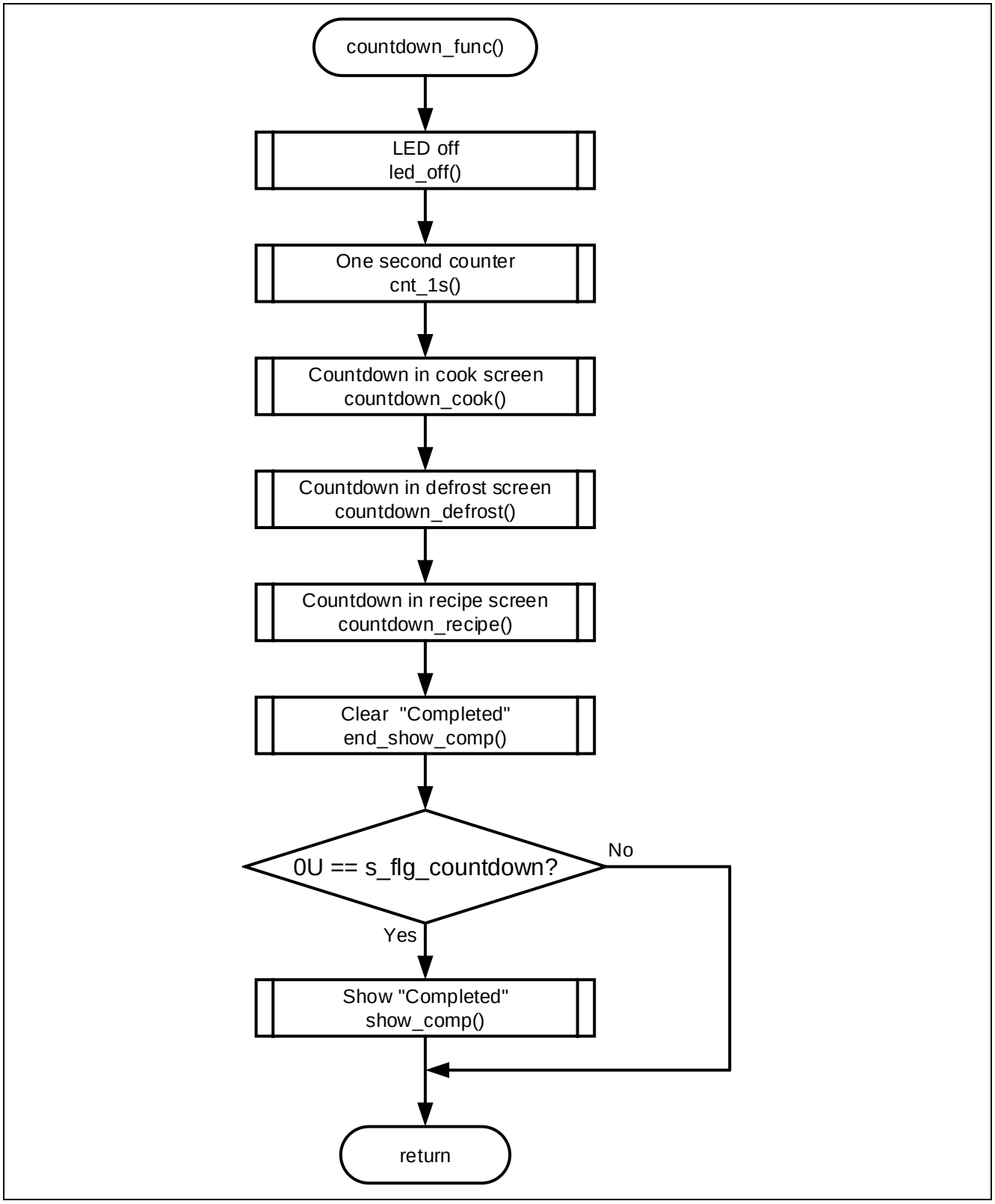


図 3-10 調理中のフローチャート

3.2.8　タッチ制御フローチャート

タッチ制御の全体フローチャートを以下に示します。

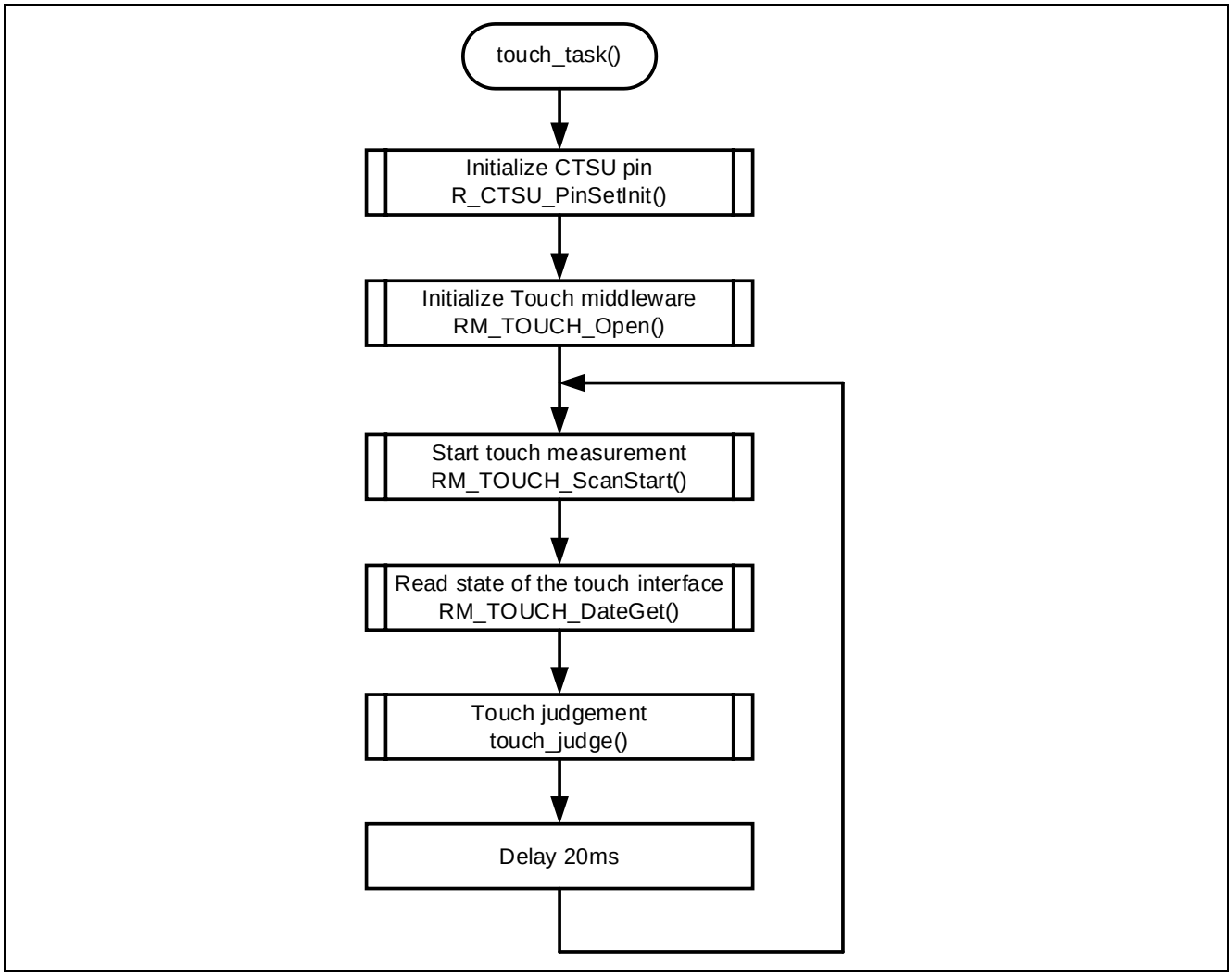


図 3-11 タッチ制御のフローチャート

3.2.9　タッチ判定処理

タッチ判定のフローチャートを以下に示します。

タッチスライダーの右側に触れてからタッチスライダーの左側に触れた場合に、タッチスライダーが左側にスライドしたと判定します。逆側も同じです。

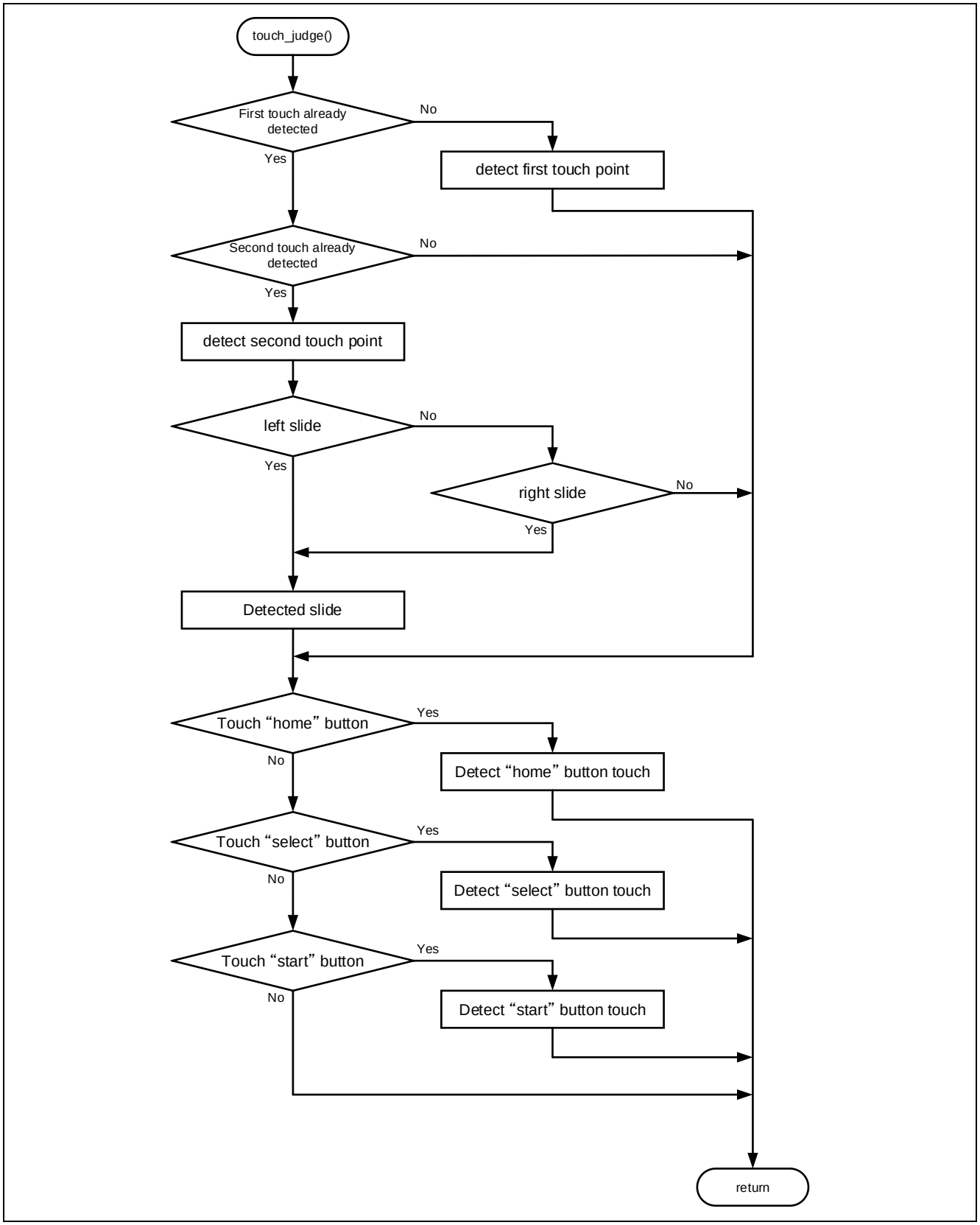


図 3-12 タッチ判定のフローチャート

3.2.10 初期画面表示の処理

初期画面表示のフローチャートを以下に示します。

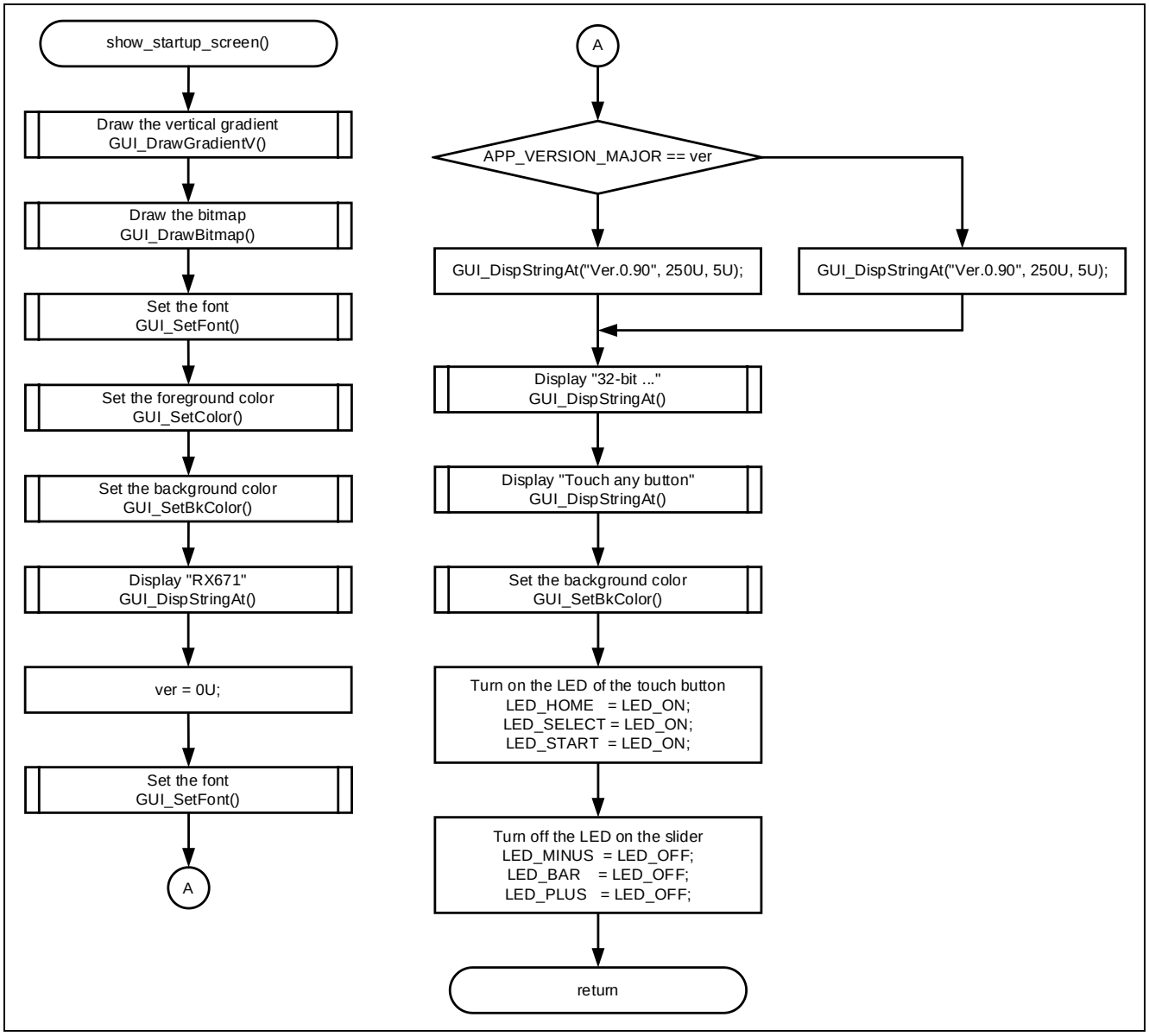


図 3-13 初期画面表示のフローチャート

3.2.11 5 秒待機処理

5 秒待機のフローチャートを以下に示します。

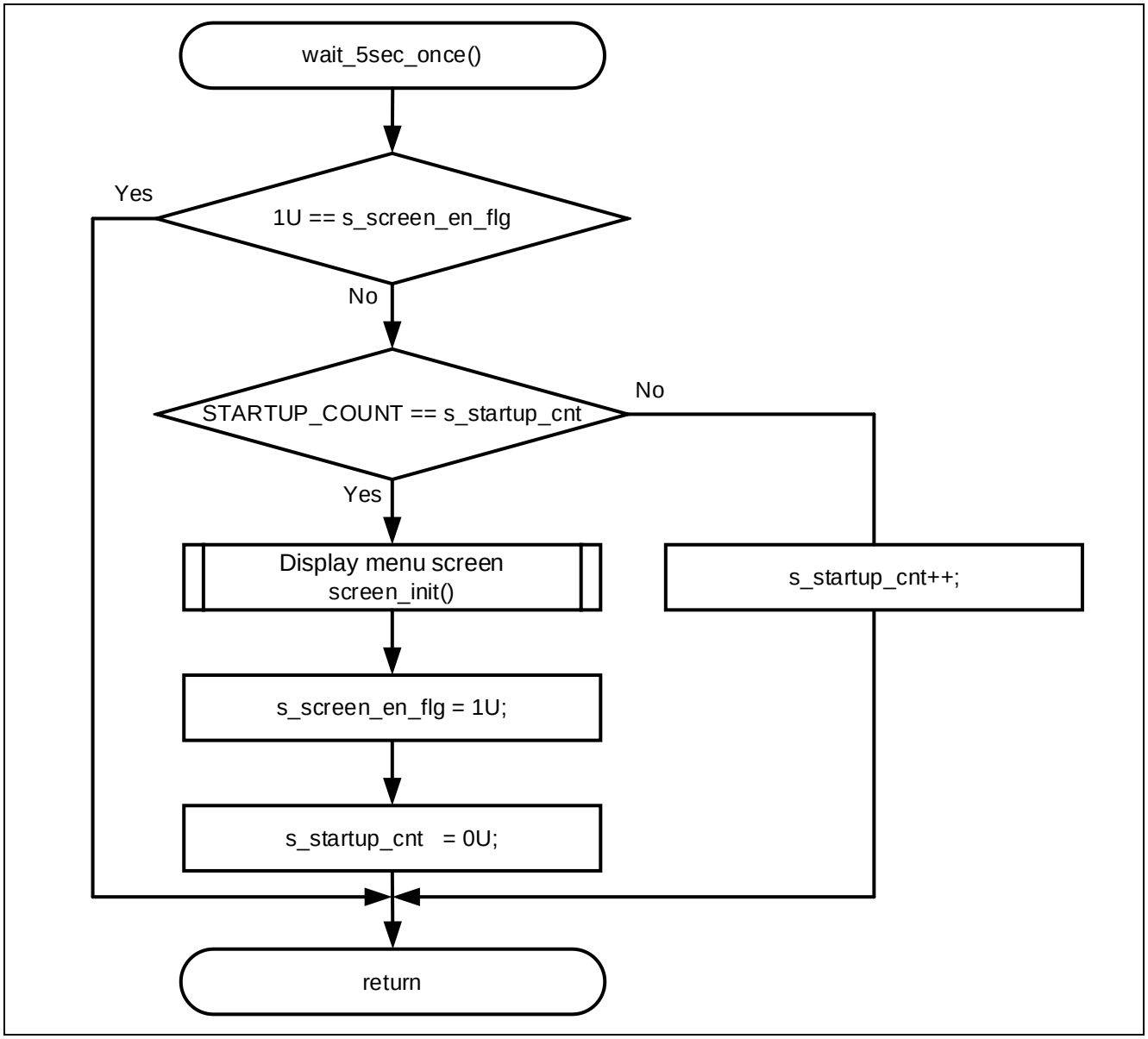


図 3-14 5 秒待機のフローチャート

3.2.12 タッチ用フラグクリア処理

タッチ用フラグクリアのフローチャートを以下に示します。

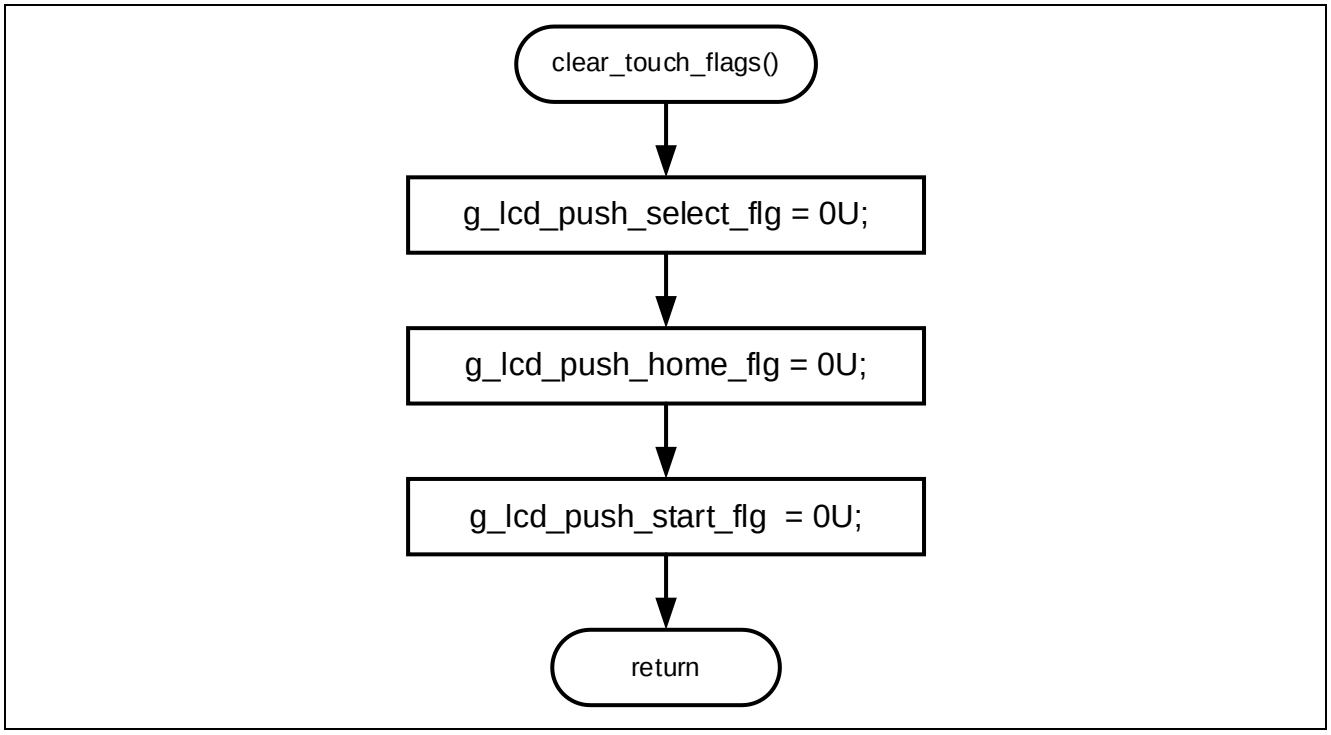


図 3-15 タッチ用フラグクリアのフローチャート

3.2.13 画面初期化処理

画面初期化のフローチャートを以下に示します。

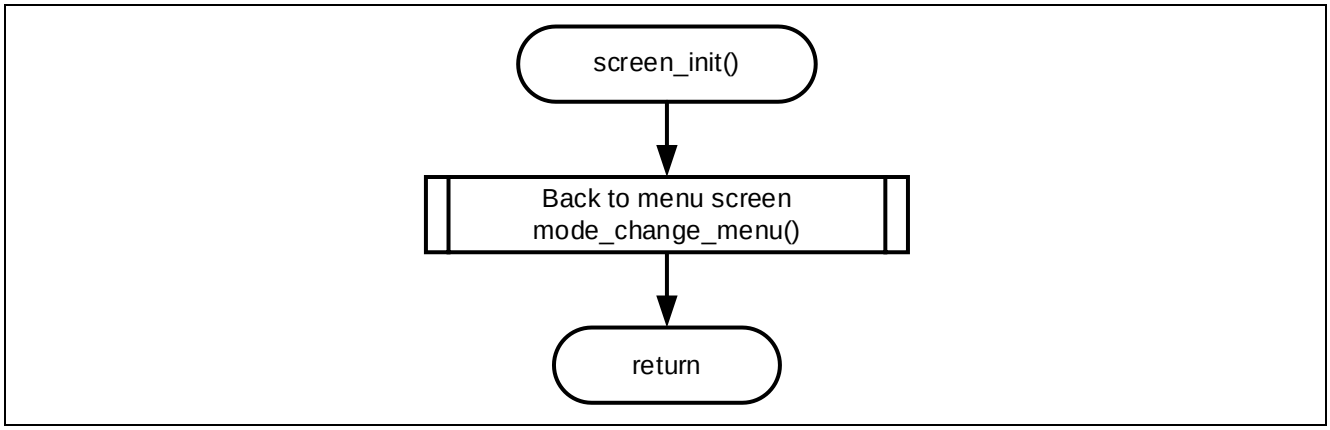


図 3-16 画面初期化のフローチャート

3.2.14 メニュー画面移動時の処理

メニュー画面移動時のフローチャートを以下に示します。

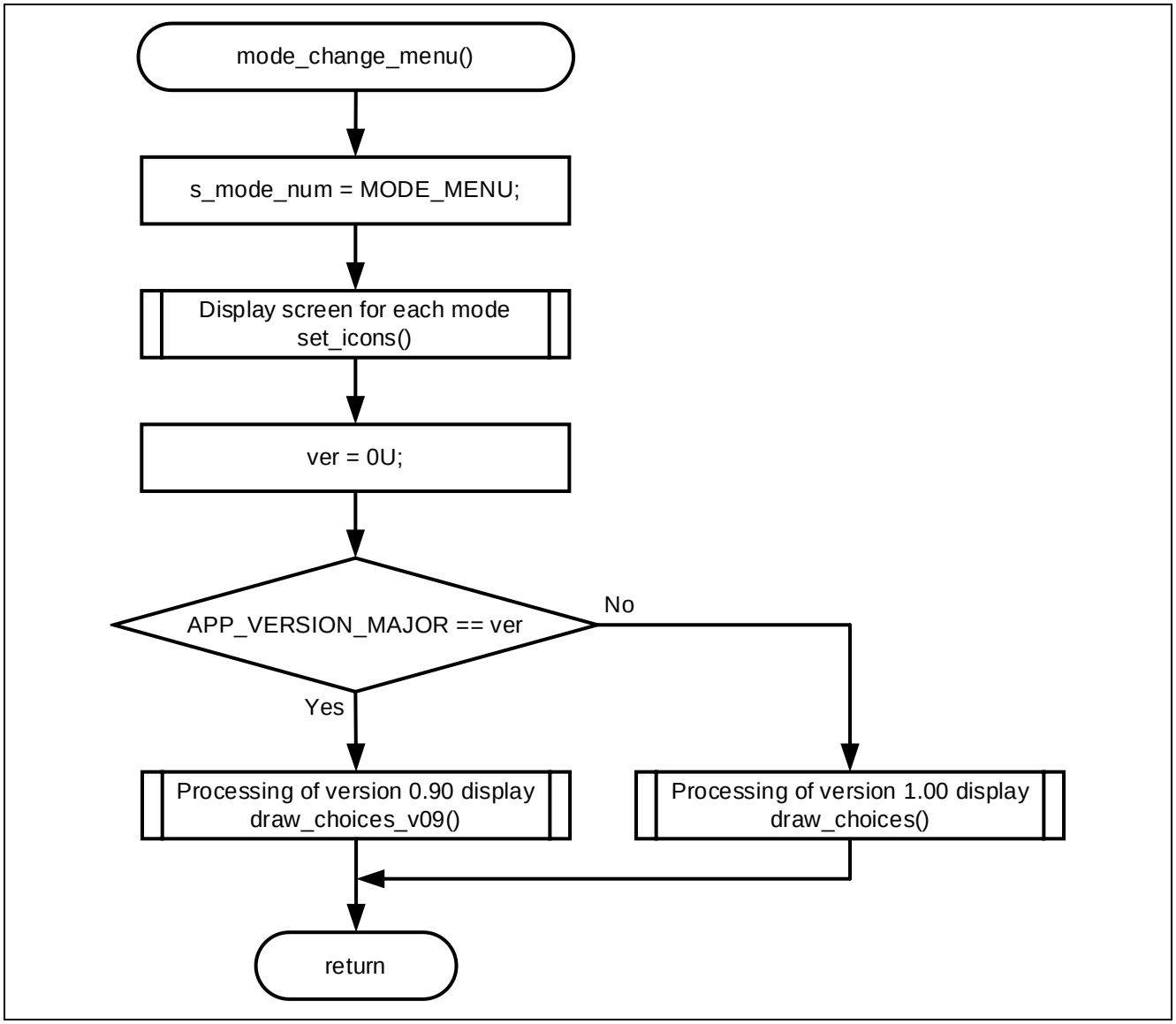


図 3-17 メニュー画面移動時のフローチャート

3.2.15 LED パターンをスリープに設定する処理

LED パターンをスリープに設定するフローチャートを以下に示します。

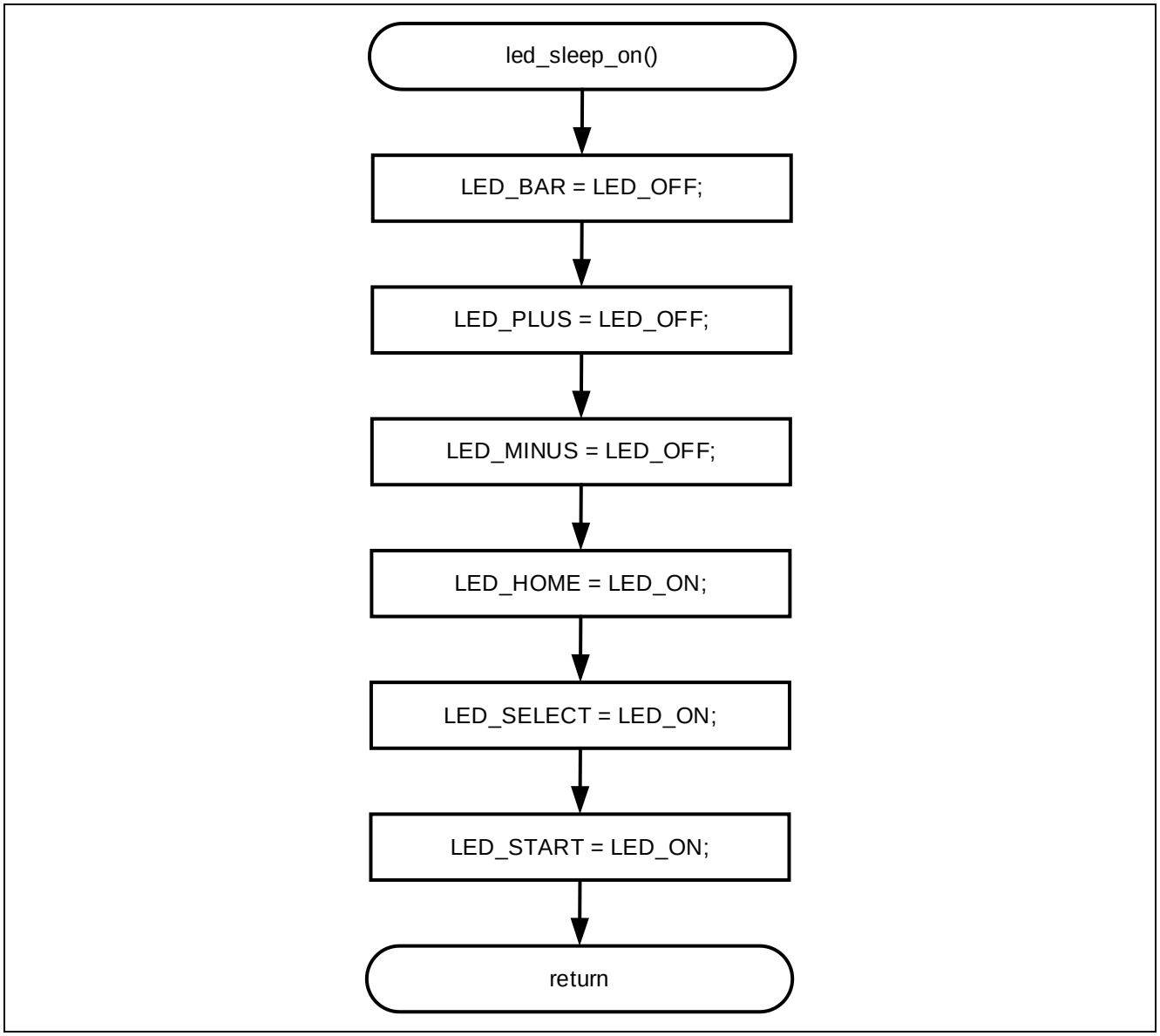


図 3-18 LED パターンをスリープに設定するフローチャート

3.3　使用端子一覧

本サンプルプログラムの使用端子一覧を以下に示します。

表 3-1　使用端子一覧

端子名	入出力	用途
P12/RXD2	入力	UART2 受信端子
P13/TXD2	出力	UART2 送信端子
PE1/TXD12	出力	UART12 送信端子
PE2/RXD12	入力	UART12 受信端子
PA6/CTS12	入力	CTS 信号入力端子
P16/RXD3	入力	USB-UART3 受信端子
P17/TXD3	出力	USB-UART3 送信端子
P27/RSPCKB-A	入出力	RSPI1 クロック端子
PE7/MISOB-B	入力	RSPI1 MISO 端子
PE6/MOSIB-B	出力	RSPI1 MOSI 端子
PC4/TSCAP	-	TSCAP 端子
P34/TS0	入力	静電容量計測端子
P26/TS3	入力	静電容量計測端子
P53/TS12	入力	静電容量計測端子
PC6/TS13	入力	静電容量計測端子
PC5/TS14	入力	静電容量計測端子
PC1/TS15	入力	静電容量計測端子
PC0/TS16	入力	静電容量計測端子
PD40～43、PD2、PD3	出力	LED 端子
PB6/TIOCA5	出力	ブザー端子

3.4　サンプルプログラムの構成

3.4.1　使用する周辺機能

本サンプルプログラムで使用する周辺機能を以下に示します。

表 3-2　使用する周辺機能一覧

周辺機能	用途
RSPI1	LCD モジュールとの SPI 通信
DMAC	RAM から RSPI への転送に使用
SCI2、SCI12	Wi-Fi モジュールとの UART 通信（2 系統）
S12AD0	CTSU FIT 内部で使用
CTSU	タッチボタン、タッチスライダーで使用
CMT0	emWin FIT 内部で使用
CMT2	RTOS で使用
PORT	LED で使用
TPU5	ブザーで使用

3.4.2　使用するコンポーネント

本サンプルプログラムで使用するコンポーネントを以下に示します。

表 3-3　使用するコンポーネント一覧

コンポーネント名	略号	バージョン
ADC Driver	r_s12ad_rx	5.00
AWS_device_shadow	AWS_device_shadow	1.0.110
AWS_ggd	AWS_ggd	1.0.110
AWS_mqtt	AWS_mqtt	1.0.110
AWS_secure_socket	AWS_secure_socket	1.0.110
AWS_tcp_ip	AWS_tcp_ip	1.0.110
Board Support Package	r_bsp	7.21
Byte-based circular buffer library	r_byteq	2.10
CMT driver	r_cmt_rx	5.40
CTSU QE API	r_ctsu_qe	2.20
DMAC driver	r_dmaca_rx	3.00
Flash API for RX100, RX200, RX600, and RX700	r_flash_rx	4.91
FreeRTOS_kernel	FreeRTOS_kernel	1.0.110
FreeRTOS_Object	FreeRTOS_Object	1.0.112
GPIO Driver	r_gpio_rx	4.70
Graphic Library with Graphical User Interface	r_emwin_rx	6.32.a.1.00
PWM モードタイマ	Config_TPU5	1.12.0
RSPI Driver	r_rspi_rx	3.04
SCI Driver	r_sci_rx	4.60
Touch QE API	rm_touch_qe	2.20
Wi-Fi Module control functions for Renesas MCUs	r_wifi_sx_ulpgn	1.16
ポート	Config_PORT	2.4.1

3.4.3　周辺機能の設定

本サンプルプログラムで使用しているスマート・コンフィグレータの設定を以下に示します。スマート・コンフィグレータの設定における各表の項目、設定内容は設定画面の表記で記載しています。

記載のない設定は、デフォルトの設定とします。

表 3-4　スマート・コンフィグレータの設定 (1/6)

分類	項目	設定、説明
スマート・コンフィグレータ >> クロック		「クロック」タブを以下の設定とする
	VCC	3.3 (V)
	メインクロック	停止：チェックを外す
	PLL 回路設定	分周比：1 倍 逓倍比：15 倍
	HOCO クロック	動作：チェックを入れる リセット後、HOCO 発振が有効
	LOCOクロック	停止：チェックを外す
	システムクロック	クロックソース：HOCO Flash IF クロック (FCLK)：60MHz システムクロック (ICLK)：120MHz 周辺モジュールクロック (PCLKA)：120MHz 周辺モジュールクロック (PCLKB)：60MHz 周辺モジュールクロック (PCLKC)：60MHz 周辺モジュールクロック (PCLKD)：60MHz
スマート・コンフィグレータ >> システム		デバッグインタフェース設定：FINE
スマート・コンフィグレータ >> コンポーネント >> r_bsp		以下の変更以外はデフォルトの設定とする
	Heap size	0x4000
	ROM Cache Enable Register	Disabled
	Software Interrupt Unit2 (SWINT2)	Used
	Software Interrupt Task Buffer Number	8
	Initial value of the software interrupt priority	Priority level 1
	Serial terminal select	Enabled
	Channel for serial terminal	Channel 3
	Bitrate for serial terminal	115200
	Interrupt priority serial terminal	Priority level 3
	HOCO Trimming select	Disabled
スマート・コンフィグレータ >> コンポーネント >> r_dmaca_rx		デフォルトの設定とする
スマート・コンフィグレータ >> コンポーネント >> r_s12ad_rx		以下の変更以外はデフォルトの設定とする
	リソース >> S12AD	
	S12AD0	チェックを入れる
スマート・コンフィグレータ >> コンポーネント >> r_ctsu_qe		以下の変更以外はデフォルトの設定とする
	TSCAP 端子	使用する：チェックを入れる
	TS0 端子	使用する：チェックを入れる
	TS3 端子	使用する：チェックを入れる
	TS12 端子	使用する：チェックを入れる
	TS13 端子	使用する：チェックを入れる
	TS14 端子	使用する：チェックを入れる
	TS15 端子	使用する：チェックを入れる
	TS16 端子	使用する：チェックを入れる

表 3-5　スマート・コンフィグレータの設定 (2/6)

分類	項目	設定、説明
スマート・コンフィグレータ >> コンポーネント >> r_gpio_rx		デフォルトの設定とする
スマート・コンフィグレータ >> コンポーネント >> r_flash_rx		以下の変更以外はデフォルトの設定とする
	Enable code flash programming (FLASH_CFG_CODE_FLASH_ENABLE)	Includes code to program ROM area
	Enable BGO/Non-blocking data flash operations (FLASH_CFG_DATA_FLASH_BGO)	Enable BGO mode
	Enable BGO/Non-blocking code flash operations (FLASH_CFG_CODE_FLASH_BGO)	Enable BGO mode
	Enable code flash self-programming (FLASH_CFG_CODE_FLASH_RUN_FROM_ROM)	Programming code flash while executing from another segment in ROM
スマート・コンフィグレータ >> コンポーネント >> r_rspi_rx		以下の変更以外はデフォルトの設定とする
	Dummy data of reception	0x00
	RSPI channel 0	Unused
	RSPI channel 1	Used
	RSPI channel 2	Unused
	Interrupt priority level of RSPI channel 1	Level 3
	RSPI1	チェックを入れる
	RSPCKB 端子	使用する：チェックを入れる
	MOSIB 端子	使用する：チェックを入れる
	MISOB 端子	使用する：チェックを入れる
スマート・コンフィグレータ >> コンポーネント >> r_sci_rx		以下の変更以外はデフォルトの設定とする
	Use ASYNC mode	Include
	Use SSPI mode	Include
	Include software support for channel 2 (SCI_CFG_CH2_INCLUDED)	Include
	Include software support for channel 3 (SCI_CFG_CH3_INCLUDED)	Include
	Include software support for channel 12 (SCI_CFG_CH12_INCLUDED)	Include
	ASYNC mode TX queue buffer size for channel 3 (SCI_CFG_CH3_TX_BUFSIZ)	2048
	ASYNC mode TX queue buffer size for channel 12 (SCI_CFG_CH12_TX_BUFSIZ)	2048
	ASYNC mode RX queue buffer size for channel 3 (SCI_CFG_CH3_RX_BUFSIZ)	2048
	ASYNC mode RX queue buffer size for channel 12 (SCI_CFG_CH12_RX_BUFSIZ)	2048
	リソース >> SCI	
	SCI2	チェックを入れる
	RXD2/SMISO2/SSCL2 端子	使用する：チェックを入れる
	TXD2/SMOSI2/SSDA2 端子	使用する：チェックを入れる
	SCI3	チェックを入れる
	RXD3/SMISO3/SSCL3 端子	使用する：チェックを入れる
	TXD3/SMOSI3/SSDA3 端子	使用する：チェックを入れる
	SCI12	チェックを入れる
	RXD12/SMISO12/SSCL12 端子	使用する：チェックを入れる
	TXD12/SMOSI12/SSDA12 端子	使用する：チェックを入れる
	CTS12#/RTS12#/SS12#端子	使用する：チェックを入れる

表 3-6　スマート・コンフィグレータの設定 (3/6)

分類	項目	設定、説明
スマート・コンフィグレータ >> コンポーネント >> r_cmt_rx		デフォルトの設定とする
スマート・コンフィグレータ >> コンポーネント >> rm_touch_qe		デフォルトの設定とする
スマート・コンフィグレータ >> コンポーネント >> r_byteq		デフォルトの設定とする
スマート・コンフィグレータ >> コンポーネント >> r_wifi_sx_ulpgn		以下の変更以外はデフォルトの設定とする
	SCI Channel number for SX-ULPGN Initial Command Port for AT command communication (WIFI_CFG_SCI_CHANNEL)	12
	SCI Channel number for SX-ULPGN Second Command Port for AT command communication (WIFI_CFG_SCI_SECOND_CHANNEL)	2
	General-purpose port PDR register connected to the SX-ULPGN EN pin (WIFI_CFG_RESET_PORT)	PORTH
	Configure RTS Port No. for WIFI_CFG_SCI_CHANNEL (WIFI_CFG_RTS_PORT)	PORTH
	Configure RTS Pin No. for WIFI_CFG_SCI_CHANNEL (WIFI_CFG_RTS_PIN)	1
	Socket Receive buffer size (WIFI_CFG_SOCKETS_RECEIVE_BUFFER_SIZE)	1024
スマート・コンフィグレータ >> コンポーネント >> r_emwin_rx		以下の変更以外はデフォルトの設定とする
	Configurations >> BasicSetting	
	Work area size for GUI	10000
	Horizontal LCD size	240
	Vertical LCD size	320
	Color depth	16 bit per pixel
	LCD orientation	ORIENTATION_CCW
	Configurations >> Select LCD Interface	
	LCD interface	LCD_IF_RSPI
	Configurations >> Select LCD Interface >> SPI Interface Setting	
	LCD interface channel number	1
	Select LCD Driver IC	LCD_DRV_IC_ILI9341
	Communication baud rate of LCD interface	30000000
	Use or unused display cache	Unuse：チェックを外す
	Configurations >> Select LCD Interface >> LCD Interface Pin Setting	
	Use Display Signal Pin	Use Display Signal Pin
	Display Signal Pin	GPIO_PORT_A_PIN_1
	Use Backlight Pin	Use Backlight Pin
	Backlight Pin	GPIO_PORT_D_PIN_6
	Use Data/Command Pin	Use Data/Command Pin
	Data/Command Pin	GPIO_PORT_A_PIN_2
	Use Chip Select Pin	Use Chip Select Pin
	Chip Select Pin	GPIO_PORT_D_PIN_5
	Configurations >> Select Touch Interface	
	Use Touch function	Not use Touch function：チェックを外す

表 3-7　スマート・コンフィグレータの設定 (4/6)

分類	項目	設定、説明
スマート・コンフィグレータ >> コンポーネント >> FreeRTOS_Kernel		以下の変更以外はデフォルトの設定とする
	The total amount of RAM available in the FreeRTOS heap	(size_t)(200U * 1024U)
	Tick vector	_CMT2_CMI2
スマート・コンフィグレータ >> コンポーネント >> FreeRTOS_Object		以下の変更以外はデフォルトの設定とする
	Tasks	Initialize: kernel start Task Code: touch_task Task Name: touch_task Stack Size: 512 Task Handler: NULL Parameter: NULL Priority: 1
		Initialize: kernel start Task Code: emwin_task Task Name: emwin_task Stack Size: 512 Task Handler: NULL Parameter: NULL Priority: 1
スマート・コンフィグレータ >> コンポーネント >> AWS_device_shadow		デフォルトの設定とする
スマート・コンフィグレータ >> コンポーネント >> AWS_ggd		デフォルトの設定とする
スマート・コンフィグレータ >> コンポーネント >> AWS_mqtt		デフォルトの設定とする
スマート・コンフィグレータ >> コンポーネント >> AWS_secure_soket		デフォルトの設定とする
スマート・コンフィグレータ >> コンポーネント >> AWS_tcp_ip		デフォルトの設定とする

表 3-8　スマート・コンフィグレータの設定 (5/6)

分類	項目	設定、説明
スマート・コンフィグレータ >> 端子 >> シリアルコミュニケーションインタフェース >> SCI2		以下の設定以外はチェックを外す
	RXD2	使用する：チェックを入れる 端子割り当て：P12 に設定
	TXD2	使用する：チェックを入れる 端子割り当て：P13 に設定
スマート・コンフィグレータ >> 端子 >> シリアルコミュニケーションインタフェース >> SCI3		以下の設定以外はチェックを外す
	RXD3	使用する：チェックを入れる 端子割り当て：P16 に設定
	TXD3	使用する：チェックを入れる 端子割り当て：P17 に設定
スマート・コンフィグレータ >> 端子 >> シリアルコミュニケーションインタフェース >> SCI12		以下の設定以外はチェックを外す
	CTS12	使用する：チェックを入れる 端子割り当て：PA6 に設定
	RXD12	使用する：チェックを入れる 端子割り当て：PE2 に設定
	TXD12	使用する：チェックを入れる 端子割り当て：PE1 に設定
スマート・コンフィグレータ >> 端子 >> シリアルコミュニケーションインタフェース >> RSPI1		以下の設定以外はチェックを外す
	MISOB	使用する：チェックを入れる 端子割り当て：PE7 に設定
	MOSIB	使用する：チェックを入れる 端子割り当て：PE6 に設定
	RSPCKB	使用する：チェックを入れる 端子割り当て：P27 に設定

表 3-9 スマート・コンフィグレータの設定（6/6）

分類	項目	設定、説明
スマート・コンフィグレータ >> コンポーネント >> Config_PORT		以下の変更以外はデフォルトの設定とする
	PORT4	チェックを入れる
	P40	出力：チェックを入れる 1 を出力：チェックを入れる
	P41	出力：チェックを入れる 1 を出力：チェックを入れる
	P42	出力：チェックを入れる 1 を出力：チェックを入れる
	P43	出力：チェックを入れる 1 を出力：チェックを入れる
	PORTD	チェックを入れる
	PD2	出力：チェックを入れる 1 を出力：チェックを入れる
	PD3	出力：チェックを入れる 1 を出力：チェックを入れる
スマート・コンフィグレータ >> コンポーネント >> Config_TPU5		以下の変更以外はデフォルトの設定とする
	カウンタクリア要因	TGRB5 コンペアマッチ
	カウンタクロックの選択	PCLK/64、立ち上がりエッジ
	TIOCA5 端子	端子初期出力は 0、コンペアマッチで 1 出力
	TGRB コンペアマッチ一致時の動作	TIOCA5 端子から 0 出力
	PWM 周期	504µs
	TGRA 初期値	38
	TGRB 初期値	472

3.4.4　ファイル構成

本サンプルプログラムのファイル構成を以下に示します。

表 3-10　ファイル構成

フォルダ名、ファイル名	説明
application_code	-
├ LCD	-
│　　├ Resource	画像とフォント用フォルダ
│　　└ Source	-
│　　　　├ LCD_custom_func.c	LCD 関連ソースファイル
│　　　　└ LCD_custom_func.h	LCD 関連ヘッダファイル
├ renesas_code	-
│　　├ frtos_startup	Amazon FreeRTOS 生成フォルダ
│　　├ utility	
│　　├ main_task.c	Amazon FreeRTOS メインタスク
│　　└ frtos_skeleton	-
│　　　　├ emwin_task.c	emWin 制御用タスク
│　　　　├ task_function.h	emwin_task.c , touch_task.c 用インクルードファイル
│　　　　└ touch_task.c	タッチ制御用タスク
├ touch	-
│　　├ touch_func.c	タッチ関連ソースファイル
│　　└ touch_func.h	タッチ関連ヘッダファイル
├ qe_gen	QE for capacitive touch 生成フォルダ
└ main.c	メイン処理ソースファイル
config_files	Amazon FreeRTOS 生成フォルダ
demos	
freertos_kernel	
libraries	
QE-Touch	QE for capacitive touch 生成フォルダ
vendors	Amazon FreeRTOS 生成フォルダ
└ renesas	-
└ boards	-
└ rx671-rsk	-
└ aws_demos	-
└ src	-
└ smc_gen	スマート・コンフィグレータ生成
├ Config_PORT	
├ Config_TPU5	
├ general	
├ r_cmt_rx	
├ r_config	
├ r_ctsu_qe	
├ r_dmaca_rx	
├ r_emwin_rx	
├ r_gpio_rx	
├ r_pincfg	
├ r_rspi_rx	
└ rm_touch_qe	

3.4.5　変数一覧

本サンプルプログラムで使用する変数一覧を以下に示します。

表 3-11　サンプルプログラムで使用する変数一覧

変数名	型	内容
g_sleep_flg	uint8_t	タッチボタンの状態
g_lcd_left_slide_flg	uint8_t	タッチスライダーを左にスライドしたことを示すフラグ
g_lcd_right_slide_flg	uint8_t	タッチスライダーを右にスライドしたことを示すフラグ
g_lcd_push_select_flg	uint8_t	「select」ボタンをタッチしたことを示すフラグ
g_lcd_push_home_flg	uint8_t	「home」ボタンをタッチしたことを示すフラグ
g_lcd_push_start_flg	uint8_t	「start」ボタンをタッチしたことを示すフラグ
g_screen_en_flg	uint8_t	初期画面の状態を示すフラグ
s_flg_countdown	uint8_t	LCD 上でカウントダウン中であることを示すフラグ
s_flg_touch	uint8_t	タッチボタンの状態
s_startup_cnt	uint8_t	初期画面の表示時間管理用カウンタ
s_sleep_cnt	uint16_t	無操作時間管理用カウンタ
s_mode_num	uint8_t	モードの状態
s_setting_target	uint8_t	画面の状態を示すフラグ

3.4.6　定数一覧

本サンプルプログラムで使用する定数一覧を以下に示します。

表 3-12　サンプルプログラムで使用する定数一覧

定数名	設定値	内容
LCD_BACKLIGHT	(PORT7.PODR.BIT.B1)	LCD のバックライトを制御する端子
OFF	(0U)	バックライトオフ時の値
ON	(1U)	バックライトオン時の値
TOUCH_NO	(0U)	無操作時の値
TOUCH_LEFT_SLIDE	(4U)	タッチスライダーを左にスライドしたことを表す値
TOUCH_RIGHT_SLIDE	(3U)	タッチスライダーを右にスライドしたことを表す値
TOUCH_SELECT	(1U)	「select」ボタンをタッチしたことを表す値
TOUCH_HOME	(2U)	前の画面に移動時の値
SLEEP_COUNT	(SLEEP_TIME / DELAY_TIME)	プログラム中のカウンタがこの値と等しくなった場合、LCD を消灯する
MORE_RECIPE_DETAIL	(6U)	Resipe モード時の詳細設定の値
MORE_COOK_DETAIL	(3U)	Cook モード時の詳細設定の値
MORE_DEFROST_DETAIL	(4U)	Defrost モード時の詳細設定の値
MODE_COOK	(1U)	Cook モードの調理開始の値
MODE_DEFROST	(2U)	Defrost モードの調理開始の値
MODE_RECIPE	(5U)	Recipe モードの調理開始の値
TOUCH_START	(5U)	各モードの実行開始の値
SETTING_TOP	(0U)	初期画面の値
MODE_MENU	(0U)	モードが未選択の値
LED_HOME	(PORT4.PODR.BIT.B0)	P40
LED_SELECT	(PORT4.PODR.BIT.B1)	P41
LED_START	(PORT4.PODR.BIT.B2)	P42
LED_MINUS	(PORT4.PODR.BIT.B3)	P43
LED_BAR	(PORTD.PODR.BIT.B3)	PD3
LED_PLUS	(PORTD.PODR.BIT.B2)	PD2
LED_ON	(1U)	LED オンの値
LED_OFF	(0U)	LED オフの値
APP_VERSION_MAJOR	(1U)	バージョンの表示
STARTUP_COUNT	(STARTUP_TIME / DELAY_TIME)	プログラム中のカウンタがこの値と等しくなるまで初期画面を表示する

3.4.7　関数一覧

本サンプルプログラムで使用する関数一覧を以下に示します。

表 3-13　サンプルプログラムで使用する関数一覧

関数名	概要
emwin_task	LCD 制御
GUI_Init	emWin の初期化
screen_init	LCD にメニュー画面を表示
change_screen	LCD の画面更新
slide_func	タッチスライダー操作時の処理
select_pushed_func	「select」ボタンタッチ時の処理
home_pushed_func	「home」ボタンタッチ時の処理
start_pushed_func	「start」ボタンタッチ時の処理
countdown_func	カウントダウン処理
slide_icons	メニュー画面や、Cook、Defrost、Recipe のモード選択画面のカーソル移動処理
setting_cook	Cook モード時のワット数、秒数設定
setting_defrost	Defrost モード時のレベル、グラム数設定
setting_recipe	Recipe モード時、カップケーキの個数設定
mode_select_enter	モードを変更し、モードに応じた LCD 画面表示をおこなう
change_target_cook	Cook モード時の詳細設定画面の設定対象変更
change_target_defrost	Defrost モード時の詳細設定画面の設定対象変更
mode_change_menu	メニュー画面に移動
start_cook	Cook モードの調理開始
start_defrost	Defrost モードの解凍開始
start_recipe	Recipe モードの調理開始
countdown_cook	Cook モードで調理中の処理
countdown_defrost	Defrost モードで解凍中の処理
countdown_recipe	Recipe モードで調理中の処理
touch_task	CTSU の初期化、タッチ判定関数のコール
touch_judge	タッチ判定
show_startup_screen	初期画面表示の処理
led_sleep_on	LED パターンを sleep に設定
wait_5sec_once	5 秒待機処理
clear_touch_flags	タッチ用フラグクリア処理
draw_choices_v09	バージョン 0.90 の表示処理
draw_choices	バージョン 1.00 の表示処理
cnt_1s	1 秒カウント処理
end_show_comp	調理完了画面待機処理
show_comp	調理完了画面表示処理
start_cook_detail	Cook モード時の詳細設定開始
start_defrost_detail	Defrost モード時の詳細設定開始
start_recipe_detail	Recipe モード時の詳細設定開始
led_off	LED オフの処理

3.4.8　関数仕様

本サンプルプログラムの関数仕様を以下に示します。

[関数名] emwin_task

概要	LCD 制御
ヘッダ	task_function.h
宣言	void touch_task (void * pvParameters)
説明	emWin FIT を初期化し、LCD 制御を行います。
引数	pvParameters
リターン値	なし
備考	なし

[関数名] GUI_Init

概要	emWin の初期化
ヘッダ	GUI.h
宣言	void GUI_Init (void)
説明	emWin の内部データ構造と変数を初期化します。
引数	なし
リターン値	なし
備考	なし

[関数名] screen_init

概要	LCD にメニュー画面を表示
ヘッダ	LCD_custom_func.h
宣言	void screen_init (void)
説明	LCD にメニュー画面を表示します。
引数	なし
リターン値	なし
備考	なし

[関数名] change_screen

概要	LCD の画面更新
ヘッダ	LCD_custom_func.h
宣言	void change_screen (void)
説明	タッチ操作に応じて、LCD の画面表示を行います。
引数	なし
リターン値	なし
備考	なし

[関数名] show_startup_screen

概要	初期画面表示の処理
ヘッダ	LCD_custom_func.h
宣言	void show_startup_screen(void)
説明	初期画面表示を行います。
引数	なし
リターン値	なし
備考	なし

[関数名] led_sleep_on

概要	LED パターンを sleep に設定
ヘッダ	LCD_custom_func.h
宣言	LED パターンを sleep に設定します。
説明	void led_sleep_on (void)
引数	なし
リターン値	なし
備考	なし

[関数名] touch_task

概要	CTSU の初期化、タッチ判定関数のコール
ヘッダ	task_function.h
宣言	void touch_task (void * pvParameters)
説明	CTSU を初期化し、タッチ判定関数をコールします。
引数	pvParameters
リターン値	なし
備考	なし

[関数名] touch_judge

概要	タッチ判定
ヘッダ	touch_func.h
宣言	void touch_judge (uint64_t button_status, uint16_t slider_position)
説明	タッチ判定を行い、判定結果をフラグにセットします。
引数	button_status, slider_position
リターン値	なし
備考	なし

3.4.9　ROM/RAM 使用量

本サンプルプログラムの ROM/RAM 使用量を以下に示します。

表 3-14　ROM 使用量

Size(KByte)	description
550	Amazon FreeRTOS
200	LCD Graphic data
115	emWin, LCD control
15	demo program, LCD_custom_func
24	Other
Total 904KByte	MAX 1024KByte (88% Used) : 1024KByte x 2Bank

表 3-15　RAM 使用量

Size(KByte)	description
200	OS Heap area
16	Heap area
64	AWS demo program
37	AWS cloud (exp. OTA)
11	emWin
18	Other
Total 356KByte	MAX 384KByte (92.7% Used)

4.　プロジェクトのインポート方法

サンプルプログラムは e² studio のプロジェクト形式で提供しています。本章では、e² studio および CS+ へプロジェクトをインポートする方法を示します。インポート完了後、ビルドおよびデバッガの設定を確認してください。

4.1　e² studio での手順

e² studio でご使用になる際は、以下の手順で e² studio にインポートしてください。

なお、e² studio で管理するプロジェクトのフォルダ名、およびそのフォルダに至るファイルパスには、空白文字の他、半角カナ文字、全角文字、半角記号 (特に'$','#','%') が混じらないようにしてください。

（使用する e² studio のバージョンによっては画面が異なる場合があります。）

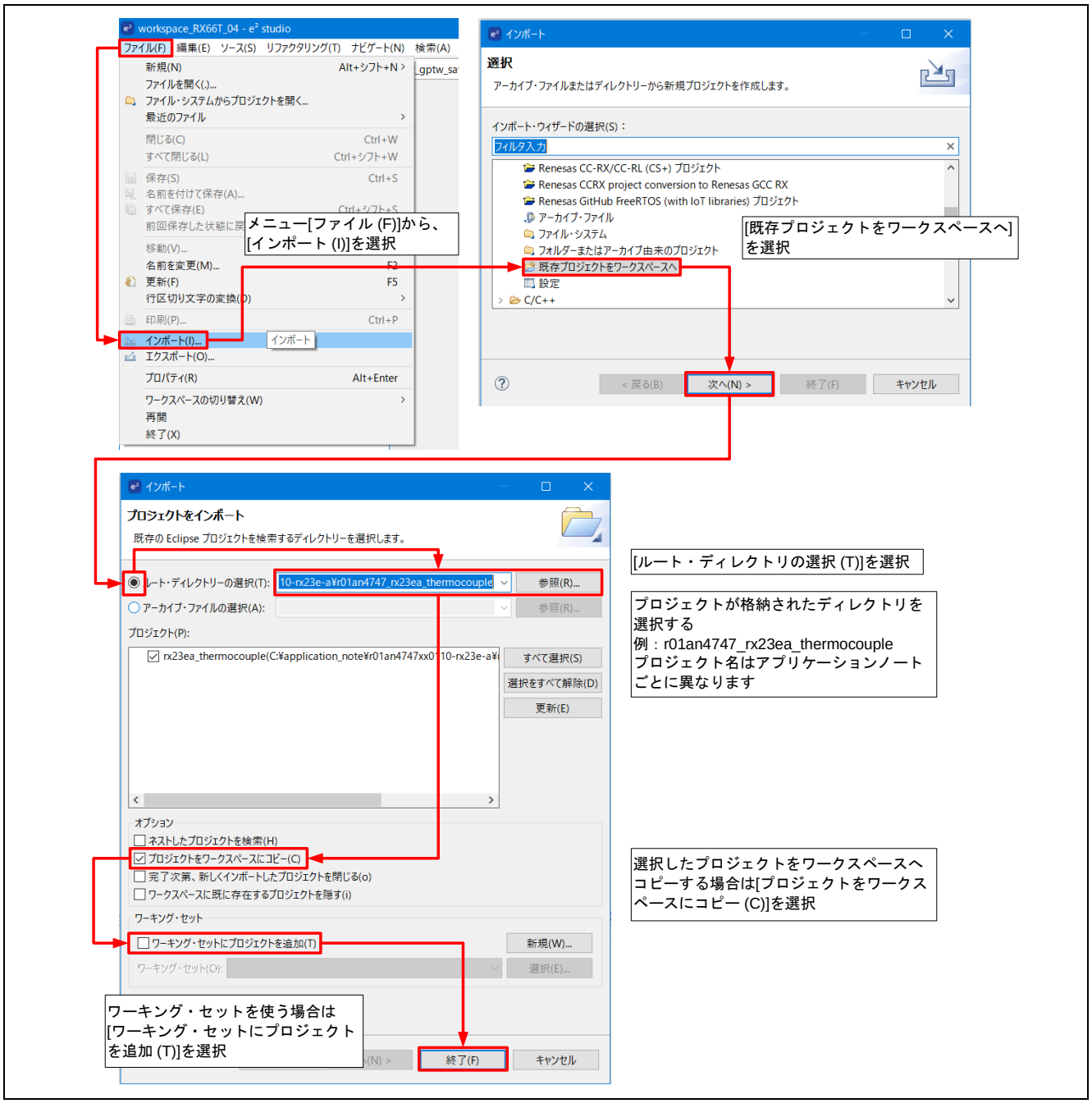


図 4-1　プロジェクトを e² studio にインポートする方法

4.2　CS+ での手順

CS+ でご使用になる際は、以下の手順で CS+ にインポートしてください。

なお、CS+で管理するプロジェクトのフォルダ名、およびそのフォルダに至るファイルパスには、 空白文字の他、半角カナ文字、全角文字、半角記号 (特に'$','#','%') が混じらないようにしてください。

（使用する CS+ のバージョンによっては画面が異なる場合があります。）

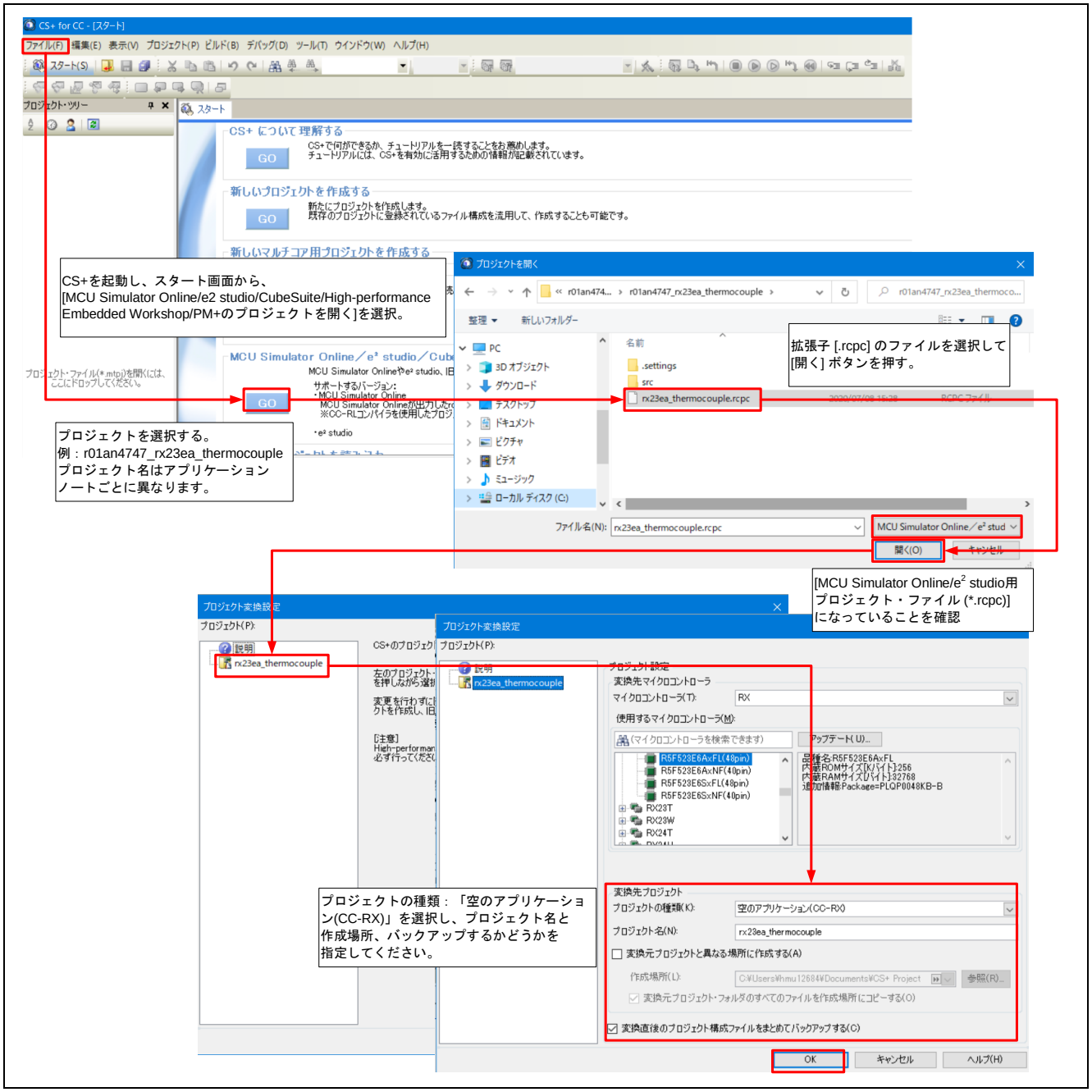


図 4-2　プロジェクトを CS+ にインポートする方法

5. デモの起動

E2 エミュレータ Lite コネクタを外して RX671 PoC の電源を投入すると、デモプログラムがスタートします。本デモプログラムは、電子レンジの表示と設定の制御を想定しています。調理の条件やレシピの選択を LCD で確認しながら、タッチボタンとタッチスライダーで設定します。

以降はタッチボタンをボタン、タッチスライダーをスライダーと記載します。

図 5-1 デモ画面と操作パネル

5.1　RX671 PoC の電源オン～メニュー画面

RX671 PoC の電源を入れると、LCD パネルに RX ロゴマークと RX671 の特長（初期画面）を約 5 秒間表示します。表示が終わるとデモプログラムがスタートし、メニュー画面となります。

また、初期画面が表示されているときに、いずれかのボタンをタッチすることで、すぐにメニュー画面に移動できます。

図 5-2 デモ開始時

5.2　メニュー画面

メニュー画面が表示されているときに、スライダーを操作することで、「Cook」、「Defrost」、「Recipe」を選択できます。

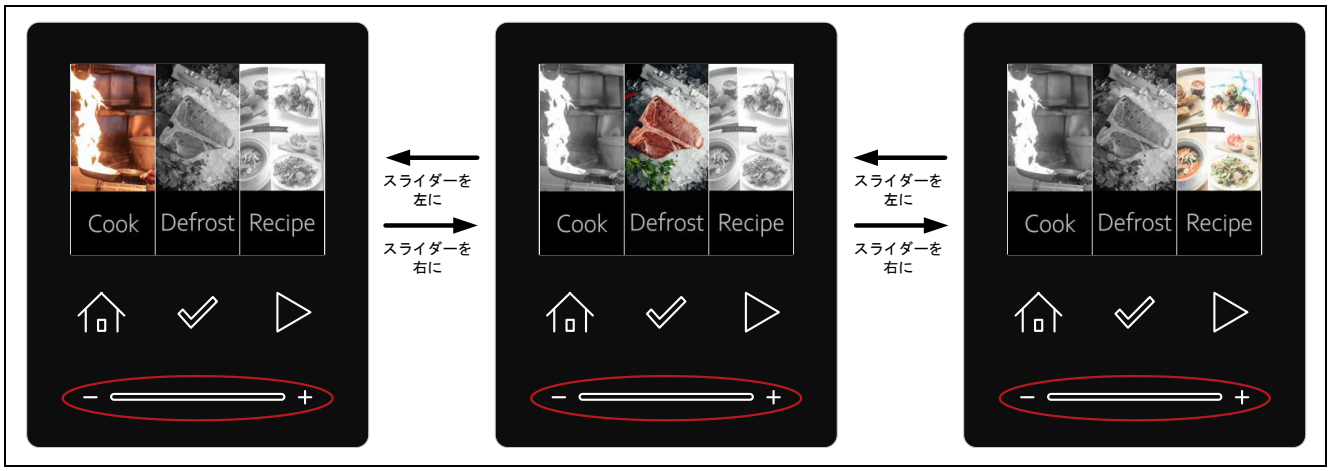


図 5-3 メニュー画面の操作方法

5.3　Cook の設定

5.3.1　モード選択画面に移動する

メニュー画面で、「Cook」が選択されているときに、「select」ボタンをタッチすると、Cook のモード選択画面に移動します。

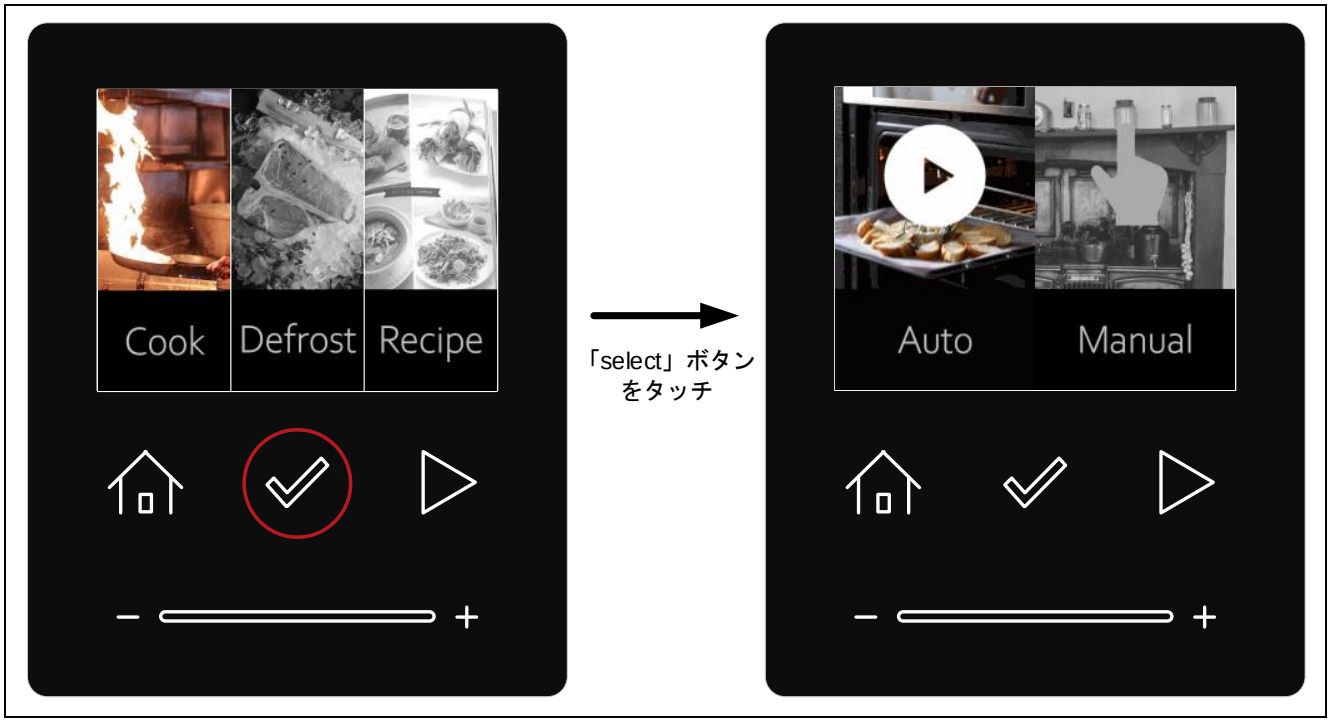


図 5-4 Cook のモード選択画面へ

5.3.2　モードを選択する

Cook のモード選択画面が表示されているときに、スライダーを操作することで、「Auto」か「Manual」を選択できます。

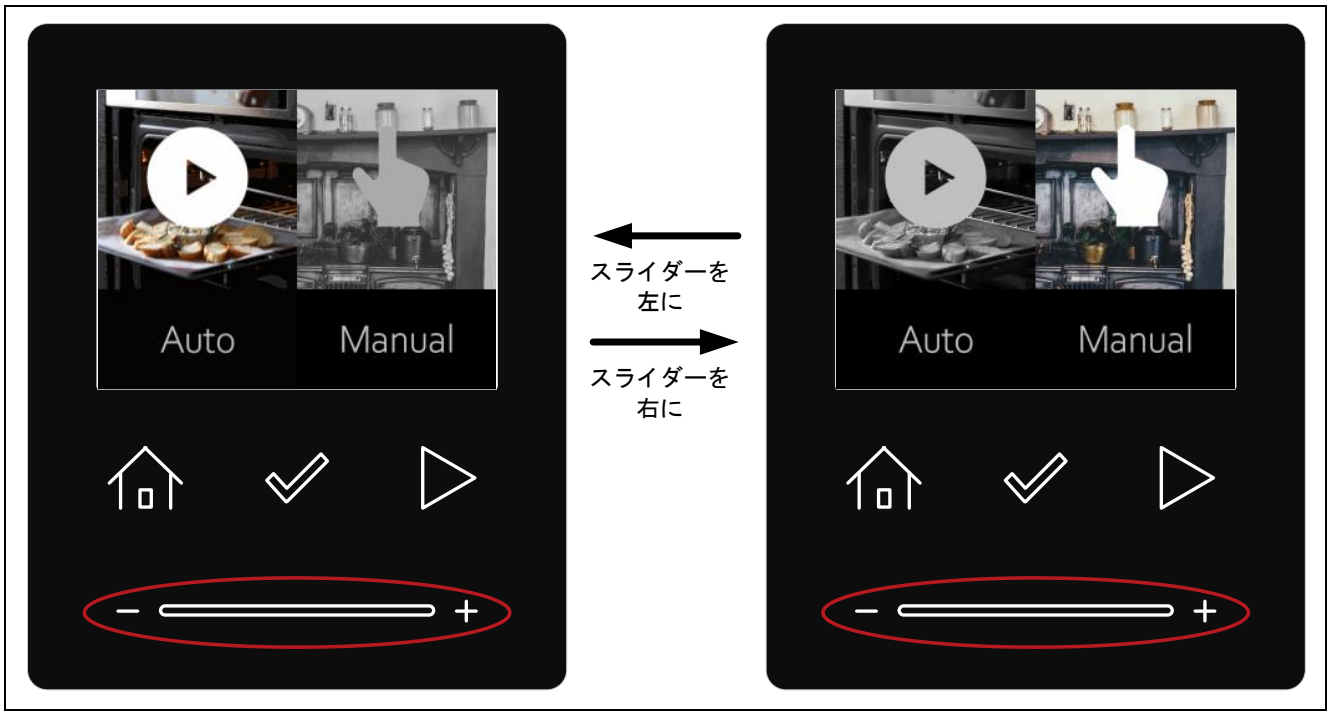


図 5-5 Cook のモード選択画面の操作方法

5.3.3　Auto を選択する

Cook のモード選択画面が表示されているときに、「start」ボタンをタッチすると、調理を開始します。

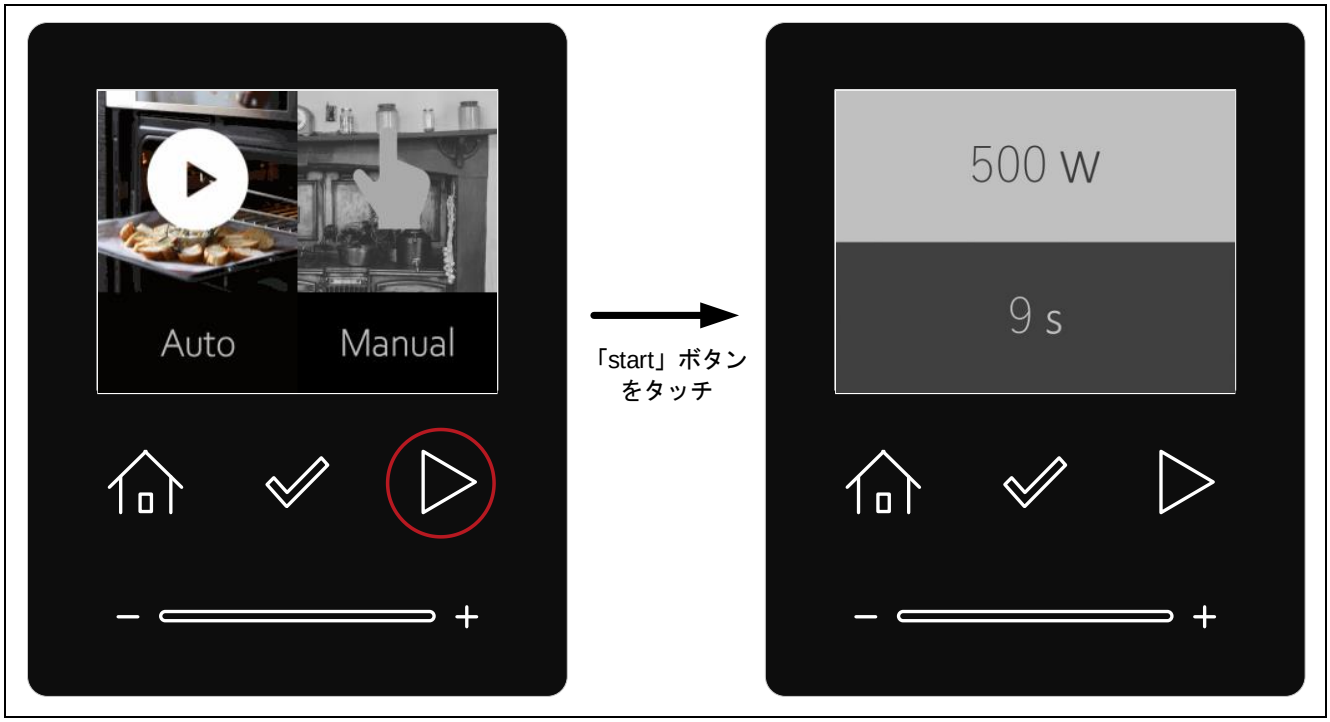


図 5-6 Auto モードで調理開始

5.3.4　Manual を選択する

モード選択画面で、「Manual」が選択されているときに、「select」ボタンをタッチすると、Cook の詳細設定画面に移動します。

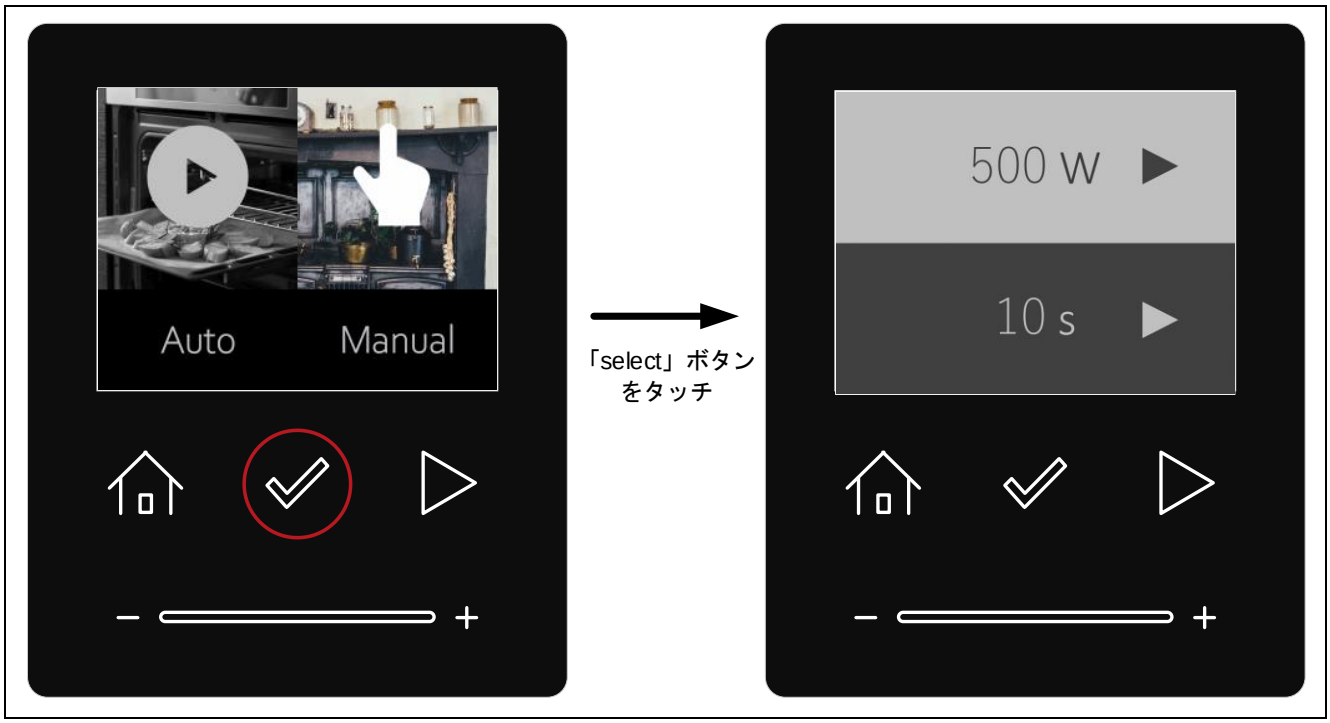


図 5-7 Cook モードの詳細設定画面へ

5.3.4.1　ワット数を設定する

ワット数は、カーソルが上側にあるときにスライダーで設定できます。
設定できるワット数は、「500W」、「600W」、「700W」です。

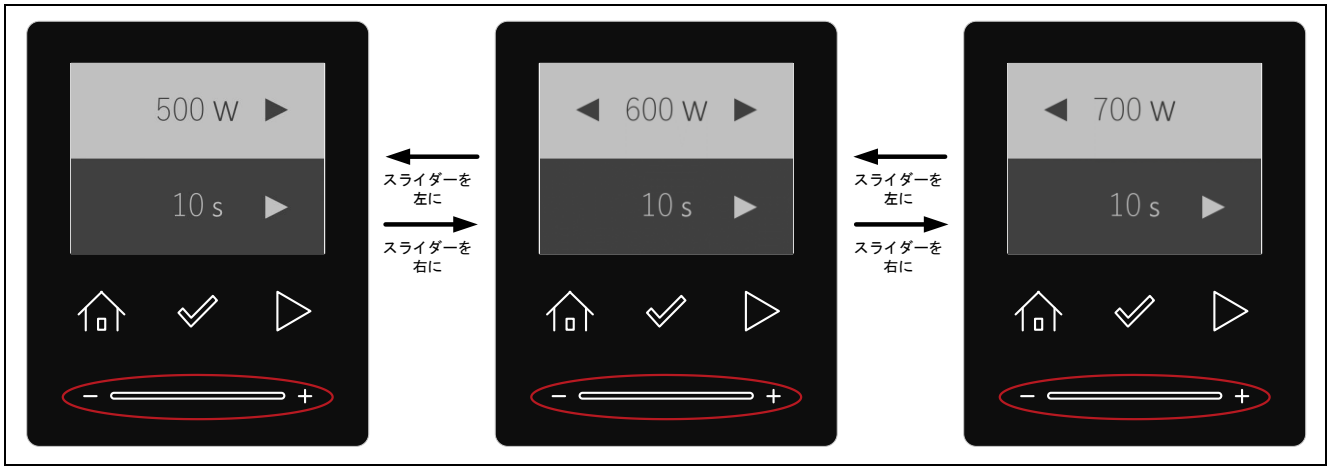


図 5-8 ワット数を設定

5.3.4.2　カーソルを移動する

Cook の詳細設定画面が表示されているときに、「select」ボタンをタッチすることで、カーソルを移動できます。
背景が薄い色の項目が選択されています。

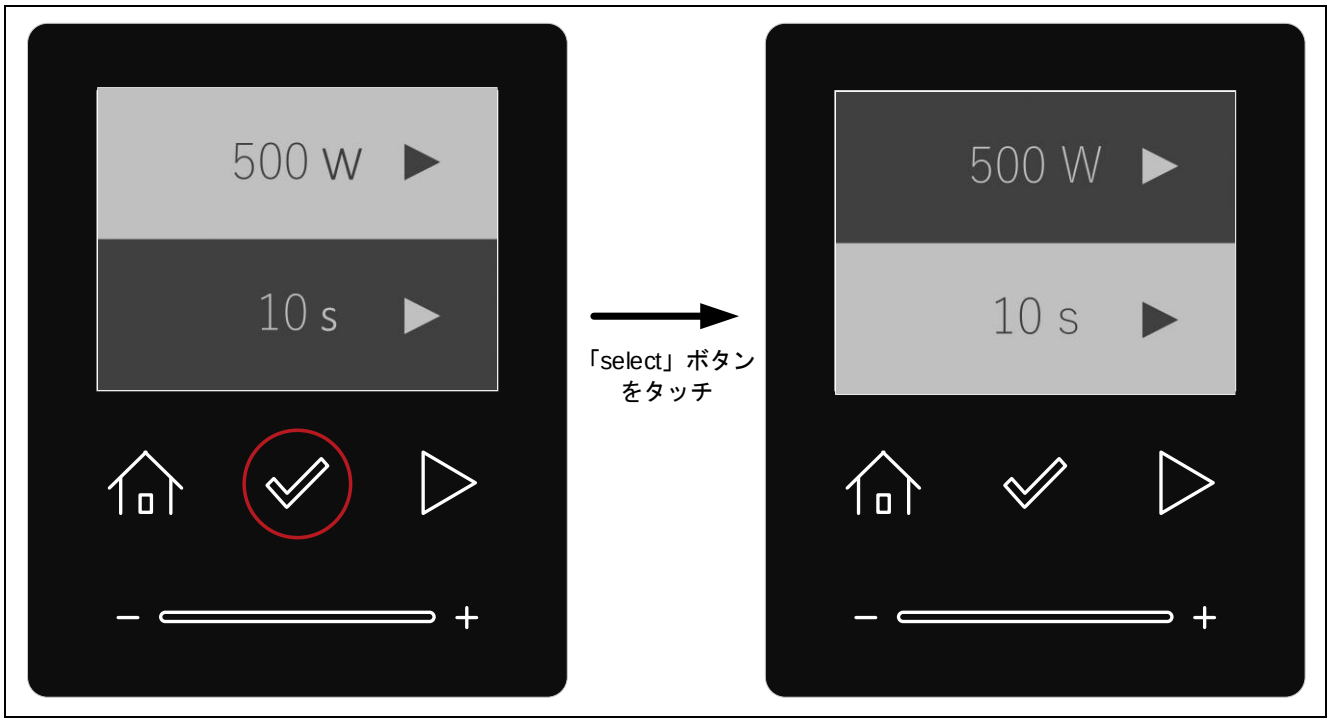


図 5-9 Cook の詳細設定画面のカーソルの操作方法

5.3.4.3　秒数を設定する

秒数は、カーソルが下側にあるときにスライダーで設定できます。
設定できる秒数は、「10s」、「20s」、「30s」です。

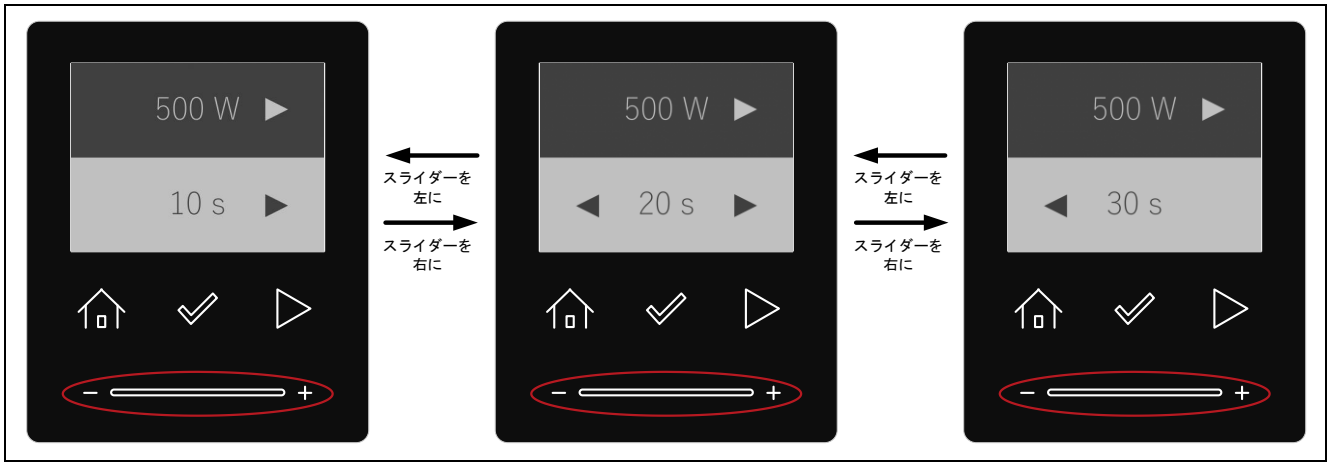


図 5-10 秒数を設定

5.3.4.4　調理を開始する

Cook の詳細設定画面が表示され、カーソルが下にあるときに、「start」ボタンをタッチすると、調理を開始します。

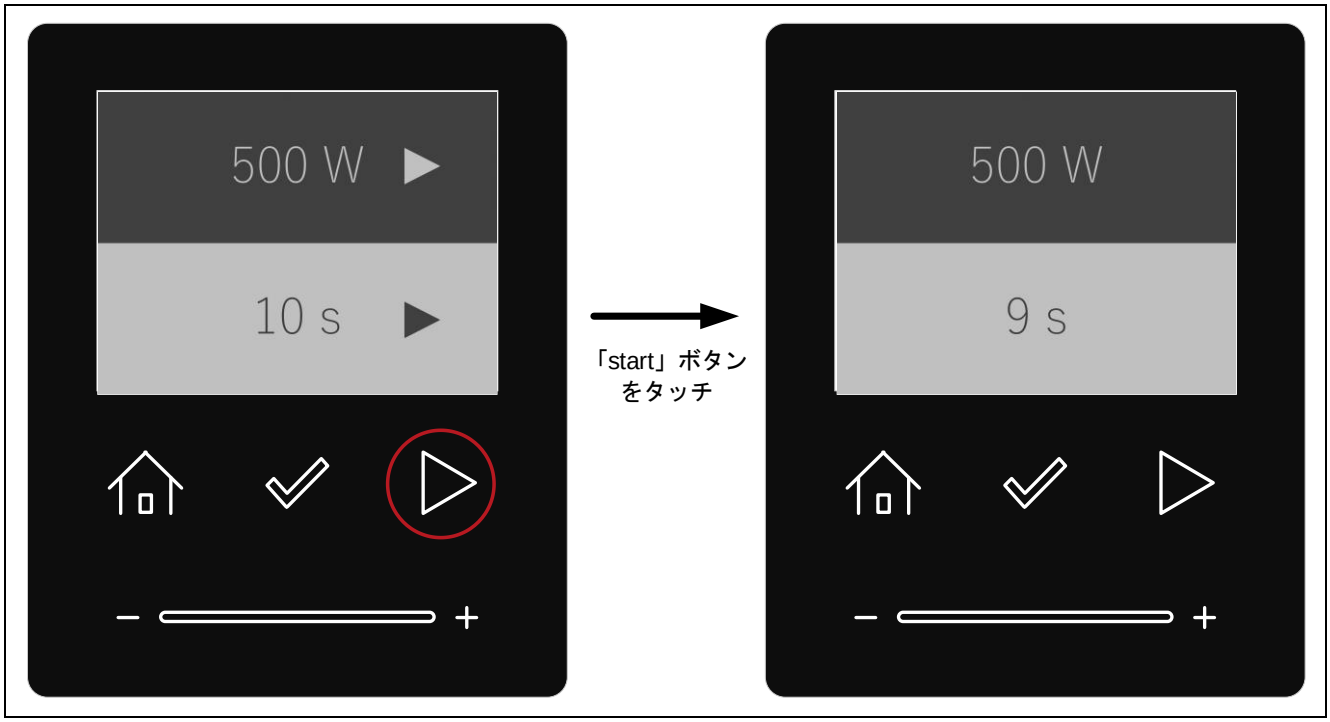


図 5-11 マニュアルモードで調理開始

5.4　Defrost の設定

5.4.1　モード選択画面に移動する

メニュー画面で「Defrost」が選択されているときに、「select」ボタンをタッチすると、Defrost のモード選択画面に移動します。

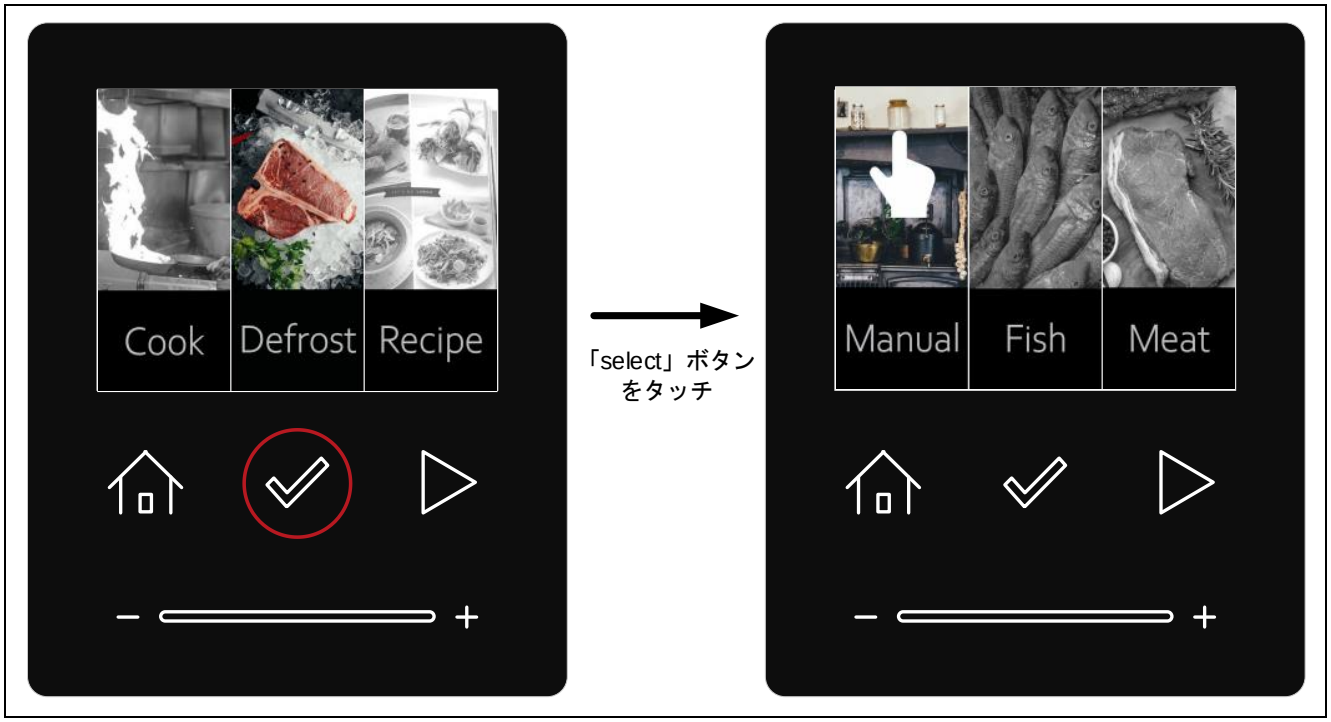


図 5-12 Defrost のモード選択画面へ

5.4.2　モードを選択する

Defrost のモード選択画面が表示されているときに、スライダーを操作することで、「Manual」、「Fish」、「Meat」を選択できます。

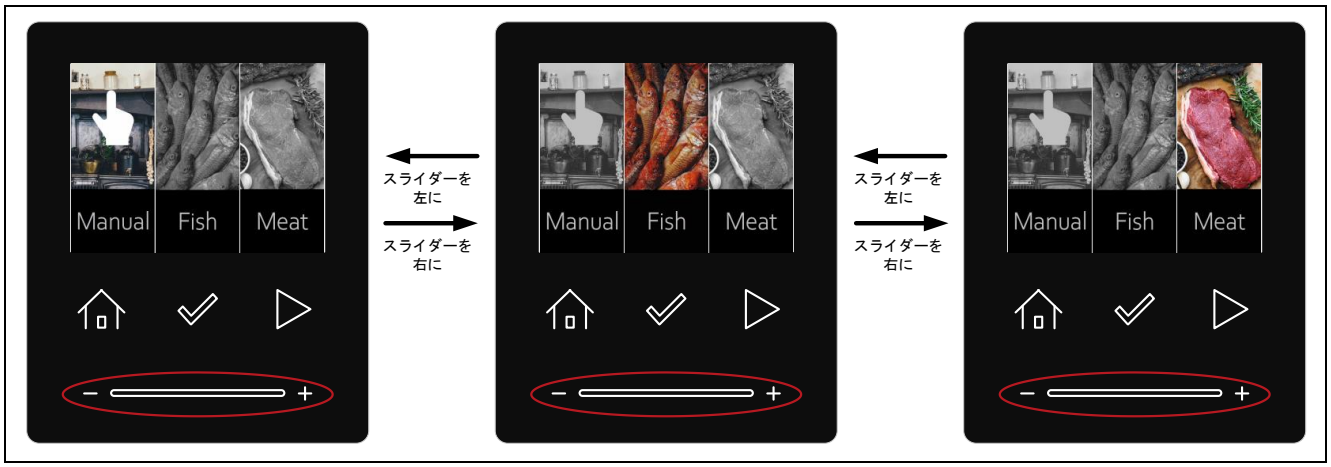


図 5-13 Defrost のモード選択画面の操作方法

5.4.3　Manual を選択する

Defrost のモード選択画面で「Manual」が選択されているときに、「select」ボタンをタッチすると、Defrost の詳細設定画面に移動します。

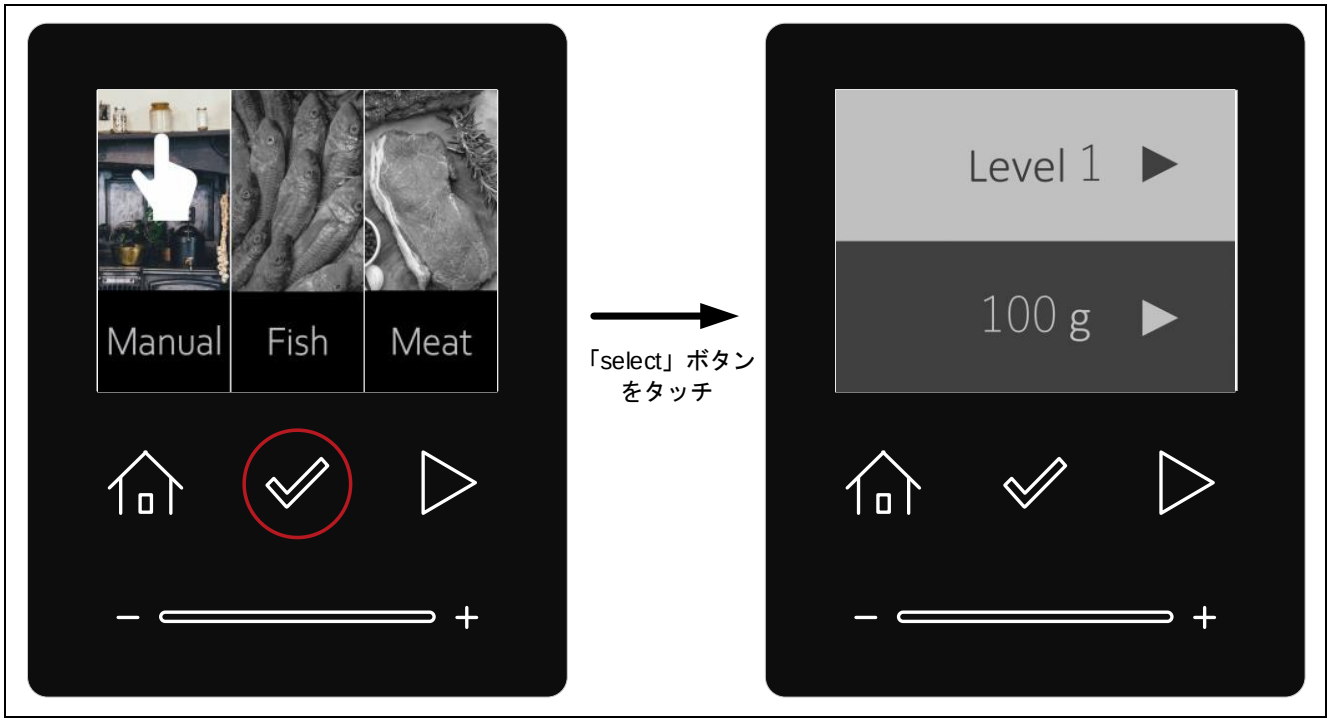


図 5-14 Defrost の詳細設定画面へ

5.4.3.1　解凍のレベルを設定する

解凍のレベルは、カーソルが上にある時にスライダーで設定できます。
設定できるレベルは、「Level 1」、「Level 2」、「Level 3」です。

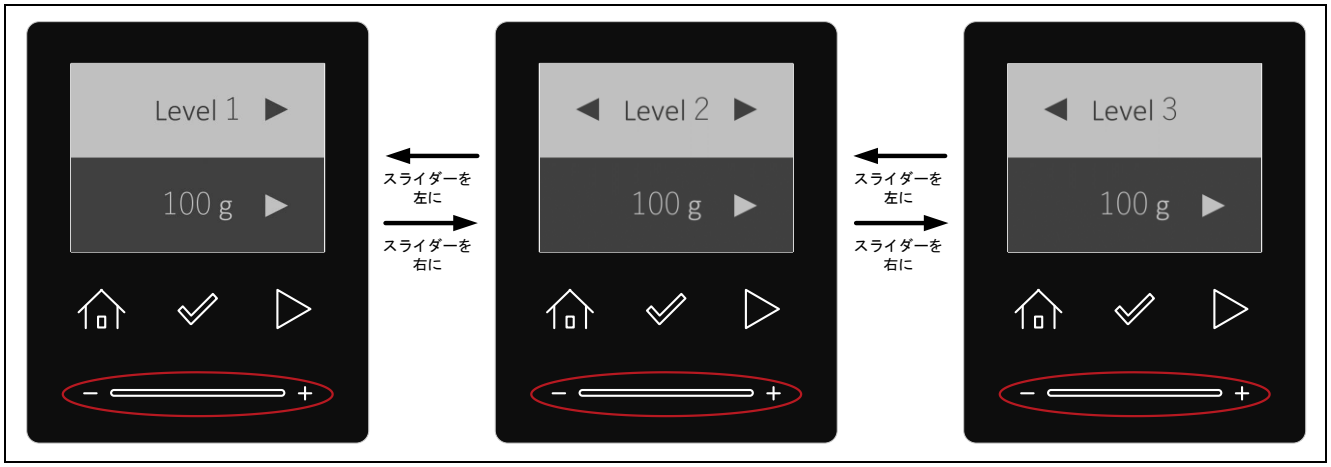


図 5-15 解凍のレベルを設定する

5.4.3.2　カーソルを移動する

Defrost の詳細設定画面が表示されているときに、「select」ボタンをタッチすると、カーソルが移動します。
背景が薄い色の項目が選択されています。

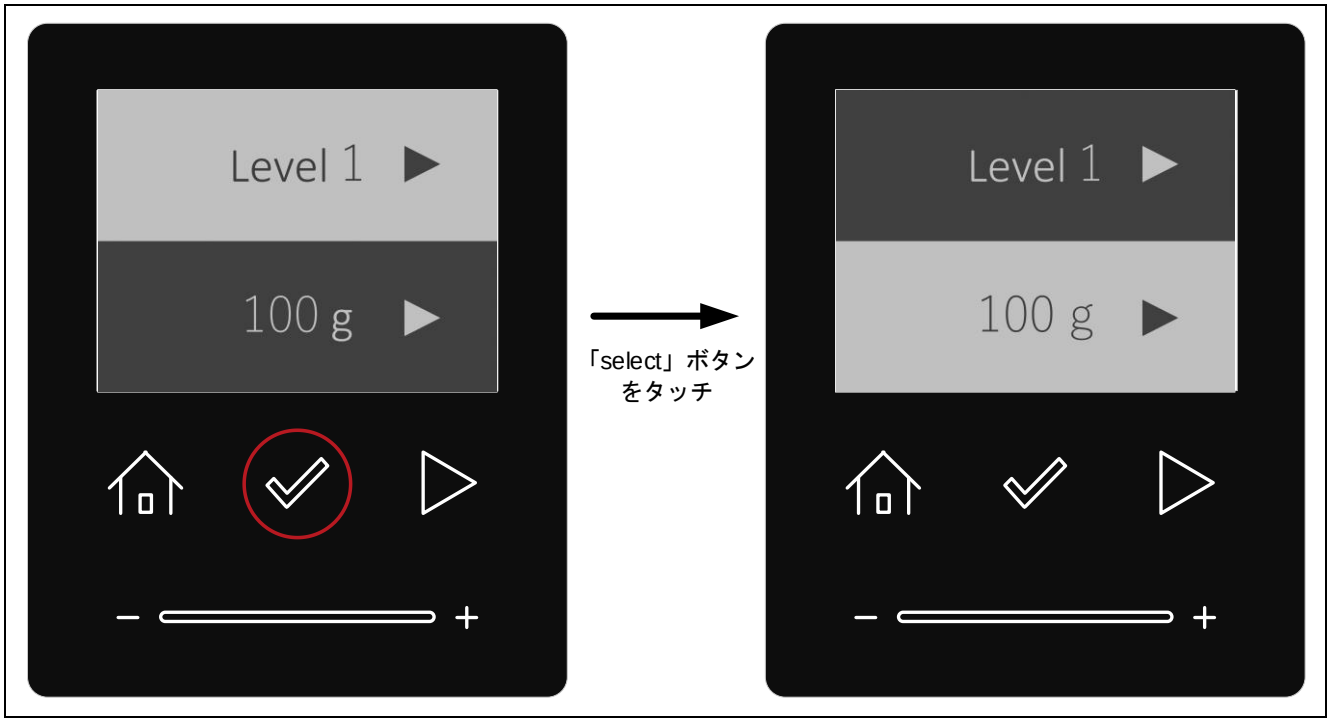


図 5-16 Defrost の詳細設定画面のカーソルの操作方法

5.4.3.3　グラム数を設定する

グラム数は、カーソルが下にあるときに、スライダーで設定できます。
指定できるグラム数は、「100g」、「200g」、「300g」です。

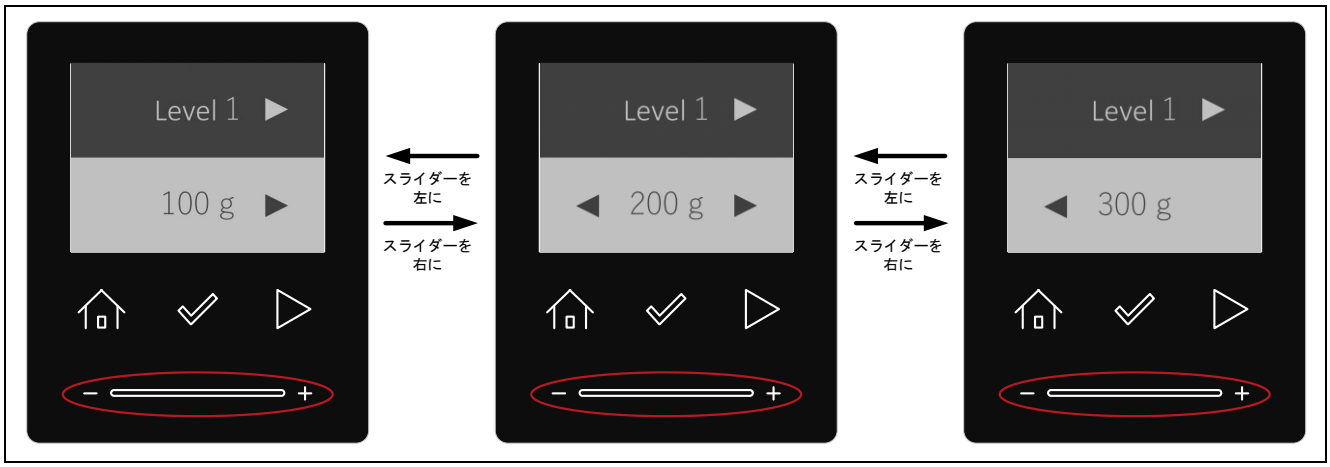


図 5-17 グラム数を設定する

5.4.3.4　解凍を開始する

Defrost の詳細設定画面が表示され、カーソルが下にあるときに「start」ボタンをタッチすると、解凍を開始します。

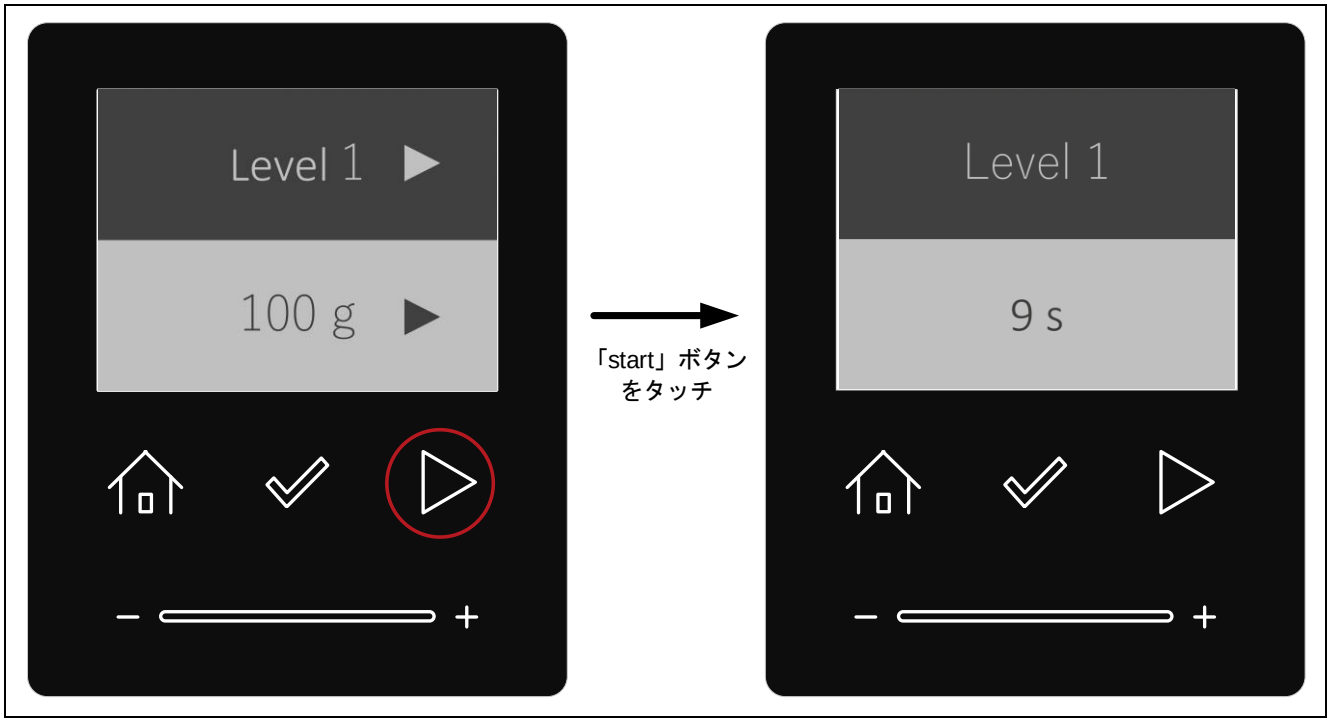


図 5-18 解凍開始

5.4.4　Fish を選択する

Defrost のモード選択画面で「Fish」が選択されているときに、「start」ボタンをタッチすると、Fish に対応した設定で解凍を開始します。

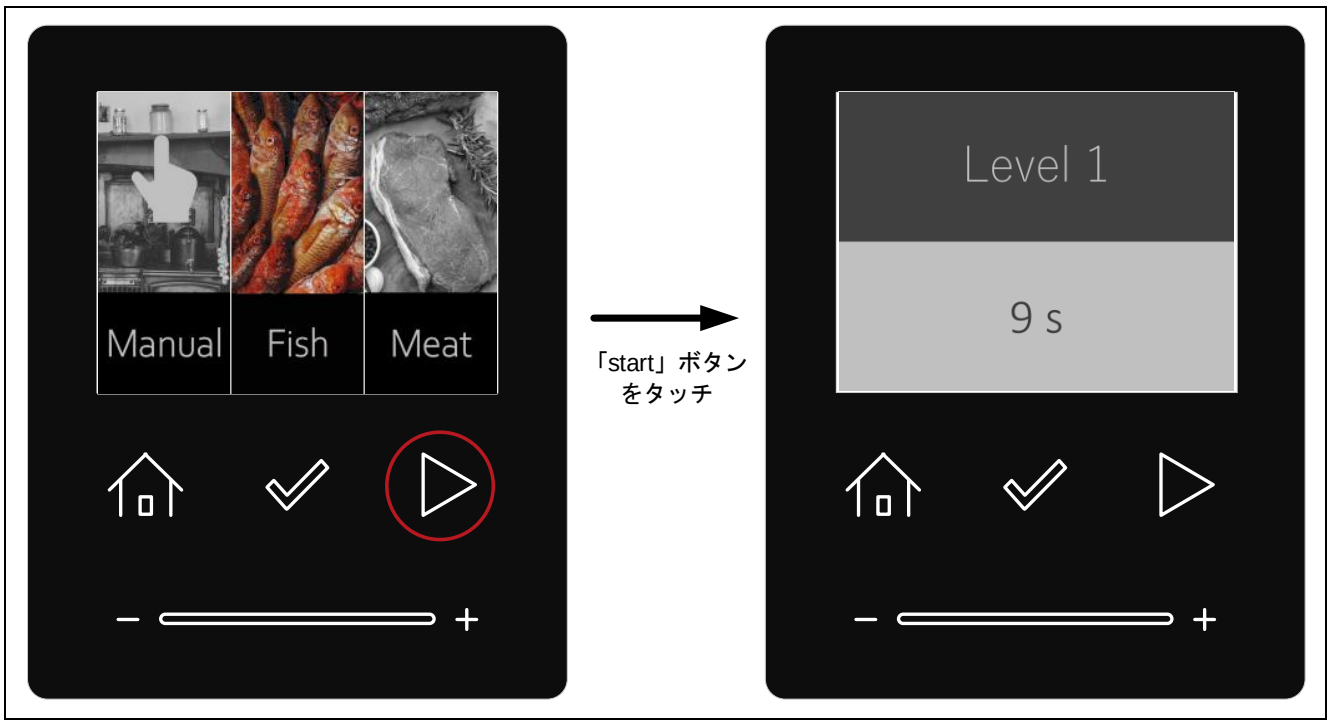


図 5-19 Fish モードで解凍開始

5.4.5　Meat を選択する

Defrost のモード選択画面で「Meat」が選択されているときに、「start」ボタンをタッチすると、Meat に対応した設定で解凍を開始します。

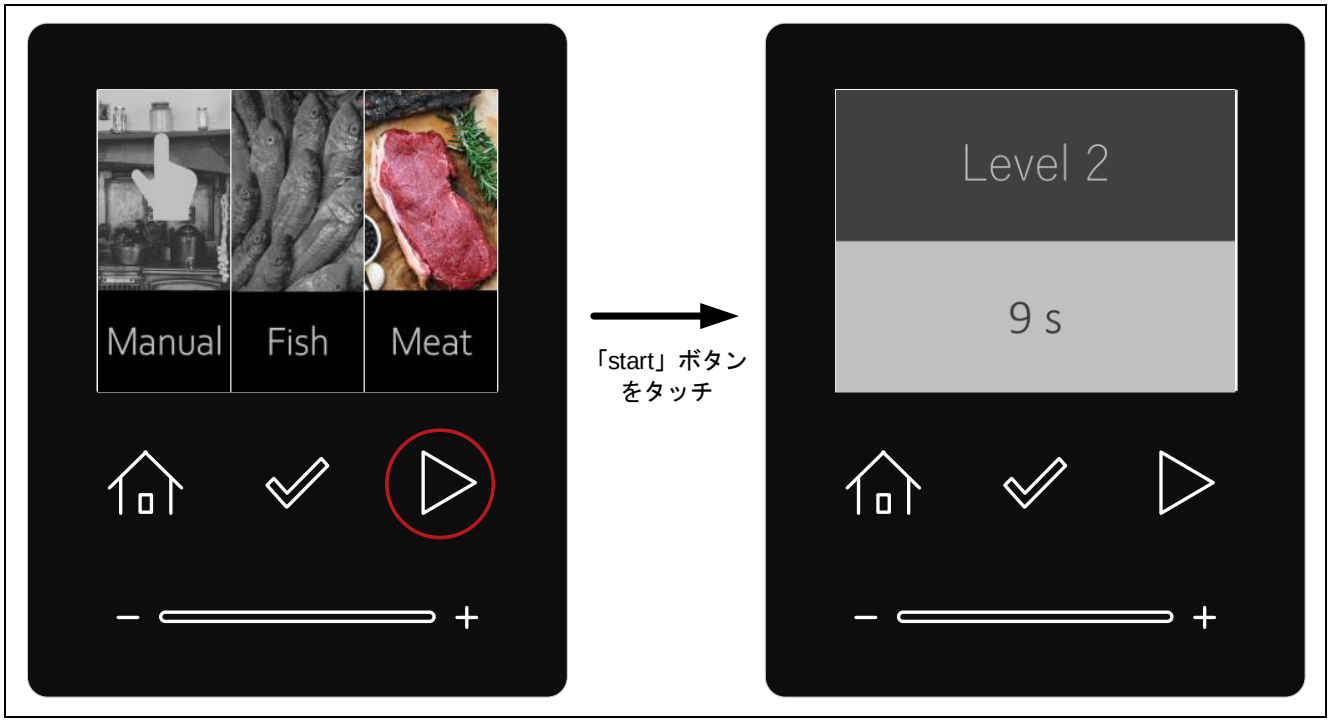


図 5-20 Meat モードで解凍開始

5.5　Recipe の設定

Recipe は、ファームウェアが ver.0.90 の場合使用できません。

5.5.1　レシピ選択画面に移動する

メニュー画面で「Recipe」が選択されているときに、「select」ボタンをタッチすると、レシピ選択画面に移動します。

図 5-21 レシピ選択画面へ

5.5.2　レシピを選択する

レシピ選択画面が表示されているときに、スライダーを操作することで、「Beef Stew」、「Garlic Shrimp」、「Cup Cake」を選択できます。

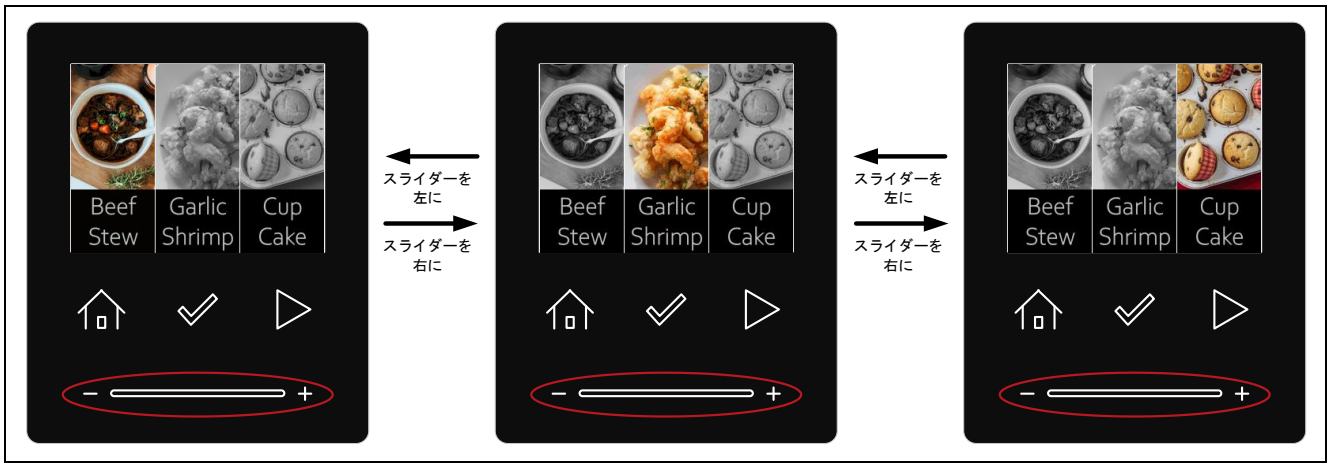


図 5-22 レシピ選択画面の操作方法

5.5.3　Beef Stew を選択する

レシピ選択画面で「Beef Stew」が選択されているときに、「start」ボタンをタッチすると、Beef Stew に対応した設定で調理を開始します。

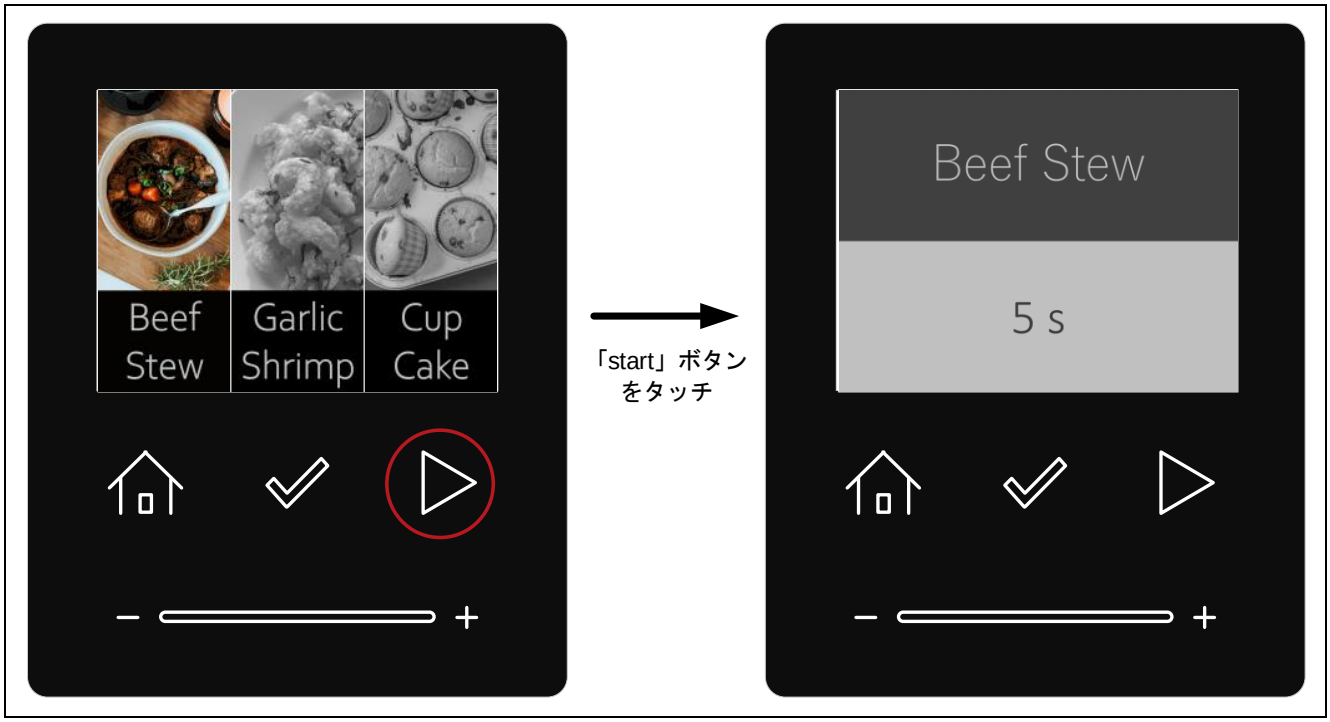


図 5-23 Beef Stew モードで調理開始

5.5.4　Garlic Shrimp を選択する

レシピ選択画面で「Garlic Shrimp」が選択されているときに、「start」ボタンをタッチすると、Garlic Shrimp に対応した設定で調理を開始します。

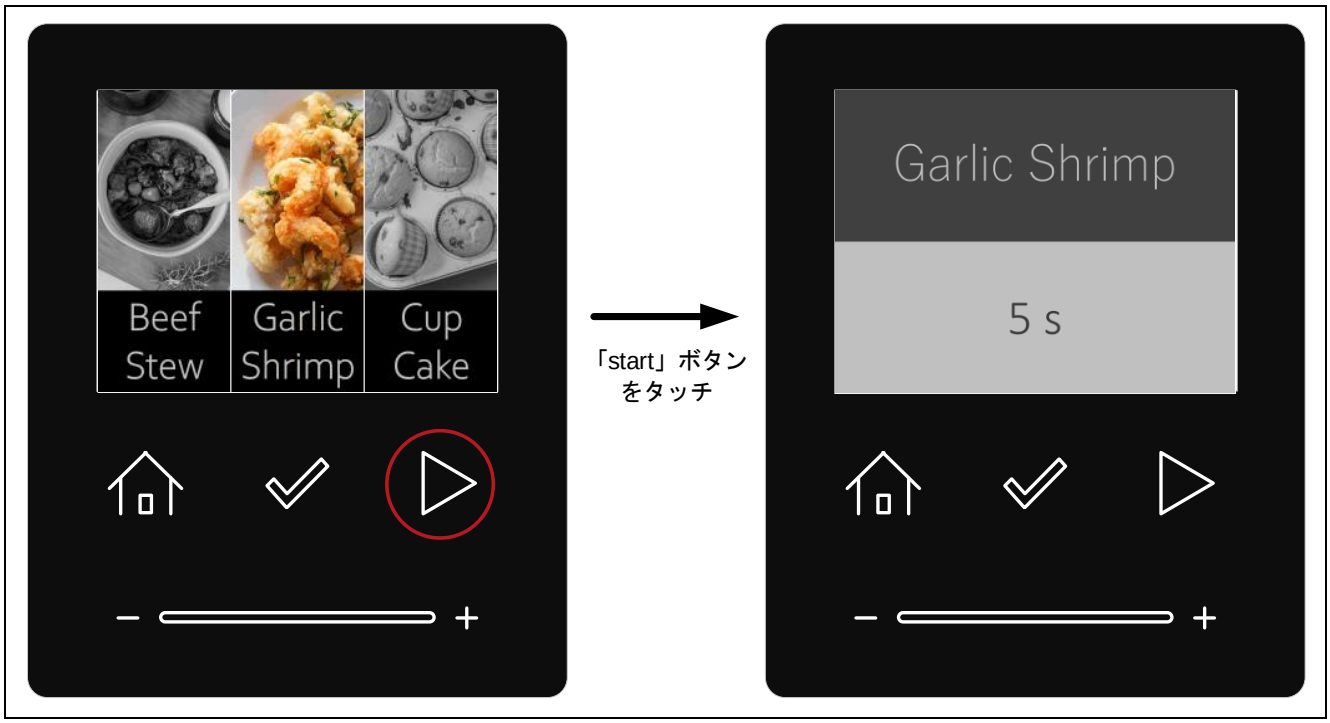


図 5-24 Garlic Shrimp モードで調理開始

5.5.5　Cup Cake を選択する

レシピ選択画面で「Cup Cake」が選択されているときに、「select」ボタンをタッチすると、Cup Cake の詳細設定画面に移動します。

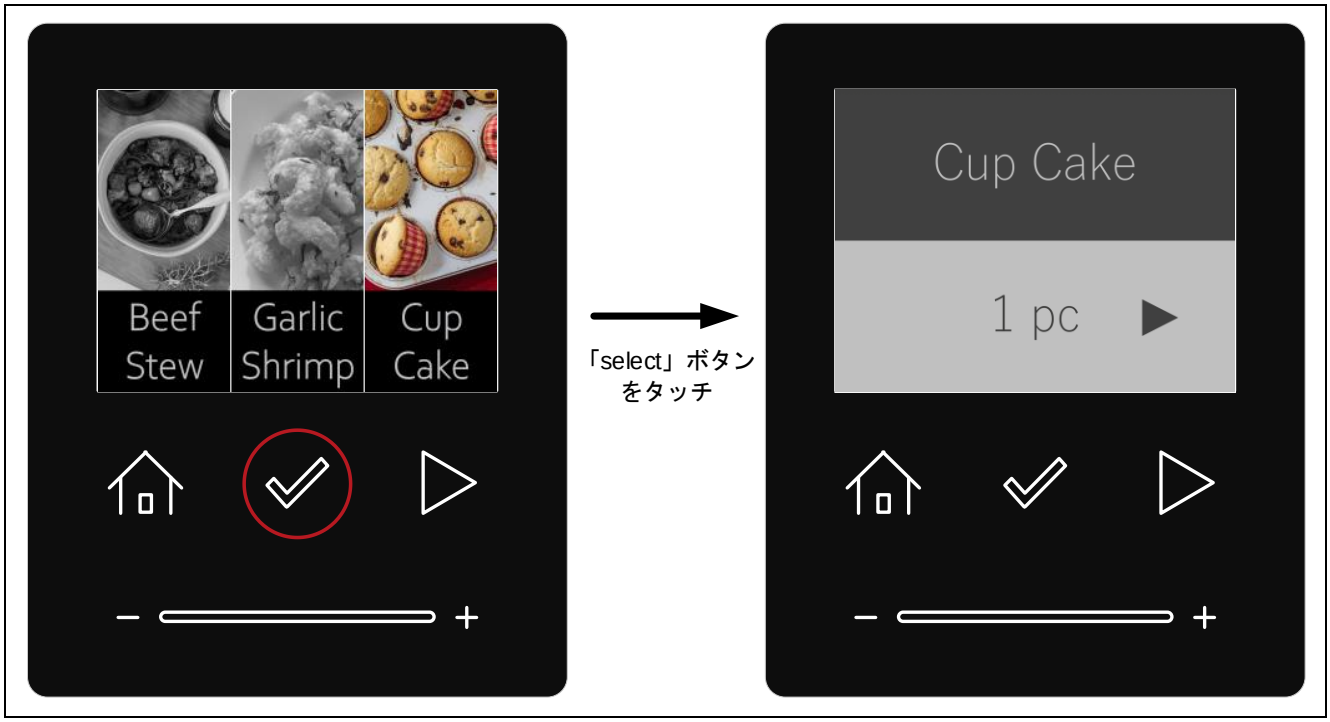


図 5-25 Cup Cake の詳細設定画面へ

5.5.5.1　個数を設定する

個数は、スライダーで設定できます。
指定できる個数は、「1pc」、「2pcs」、「3pcs」です。

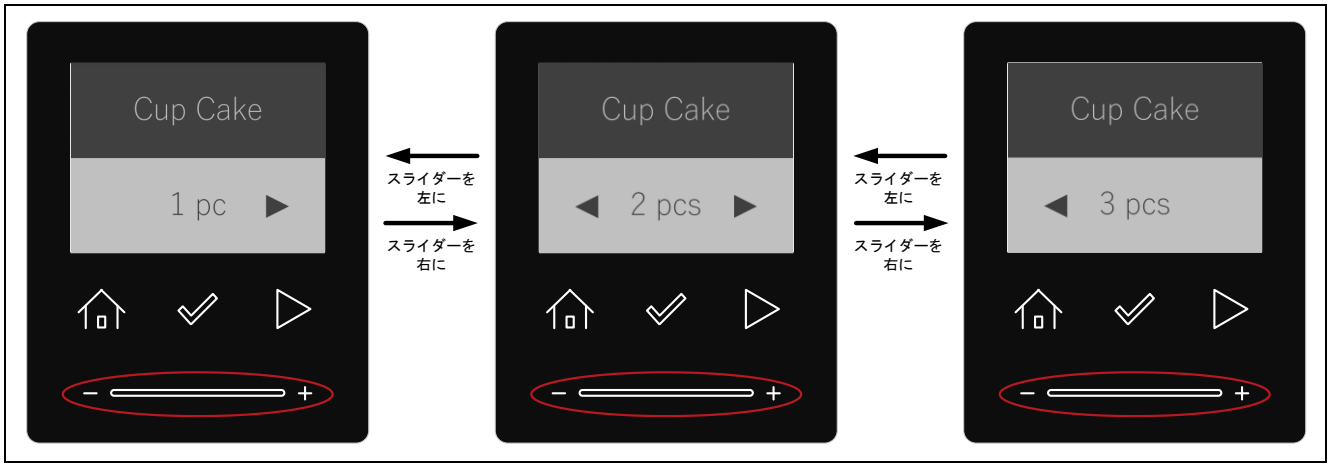


図 5-26 個数を設定する

5.5.5.2　調理を開始する

Cup Cake の詳細設定画面が表示されているときに、「start」ボタンをタッチすると、調理を開始します。

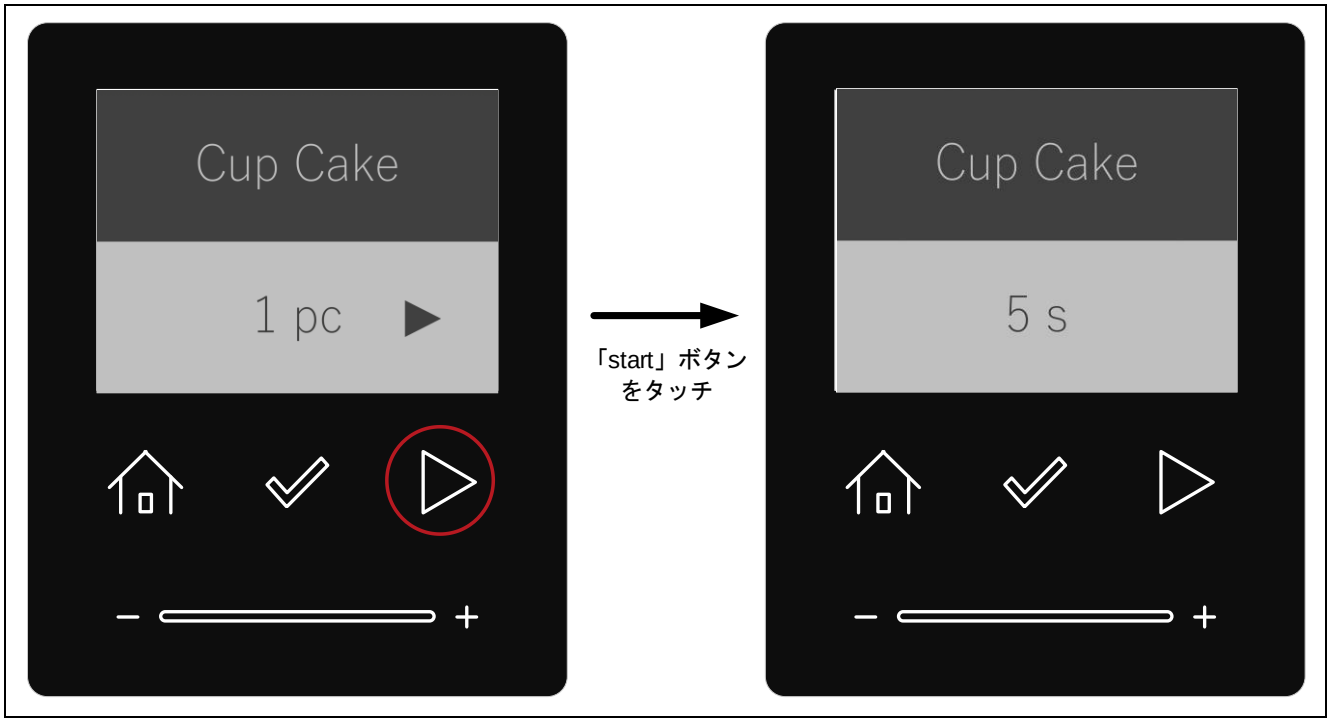


図 5-27 Cup Cake モードで調理開始

5.6　「home」ボタンについて

「home」ボタンは、どの画面からもメニュー画面に戻ることができます。

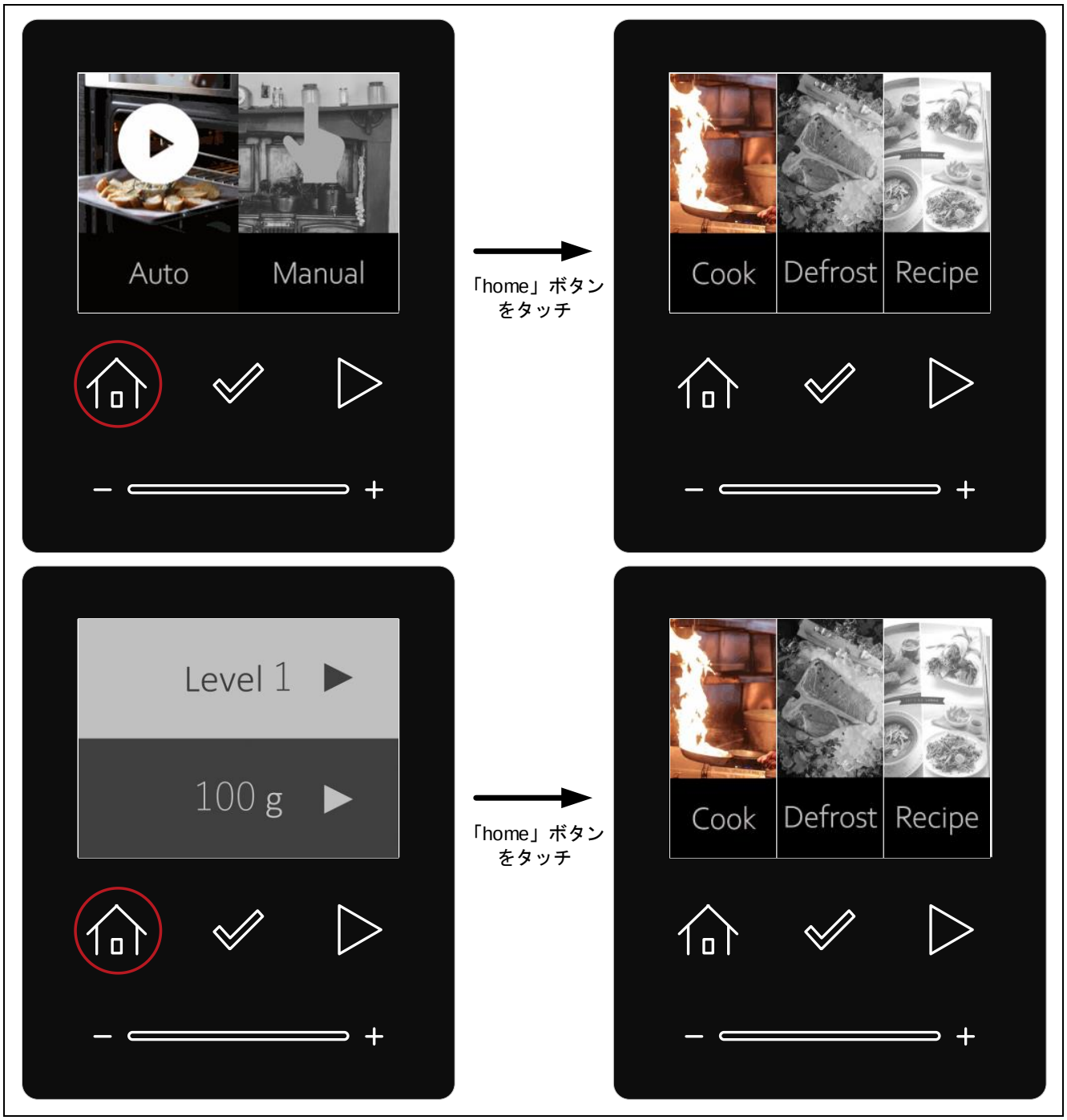


図 5-28　「home」ボタンの動作例

5.7　調理完了画面について

各調理が完了すると調理完了画面が 3 秒間表示されます。その後、自動でメニュー画面へと移動します。

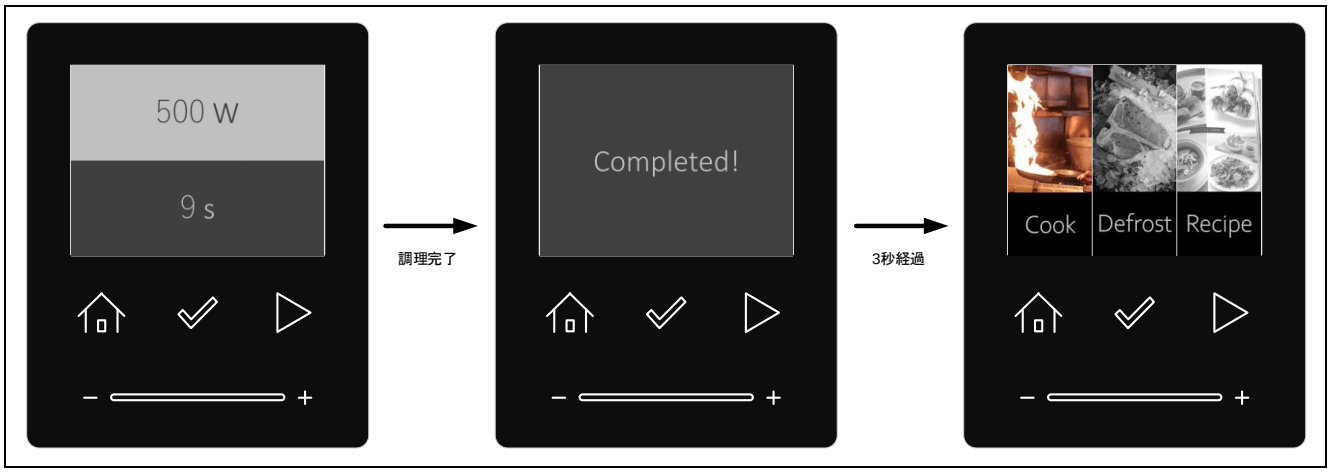


図 5-29 調理完了動作例

5.8　LCD 自動消灯機能

10 秒間タッチ操作を行わないと、LCD の電源が OFF になります。いずれかのボタンをタッチすることで、直前の画面に復帰します。

6.　ファームウェアのバージョン更新

本アプリケーションノートでは、2 つのバージョンのファームウェアを準備しています。一つ目のファームウェアのバージョンは 0.90 で、ファイル名は userprog_v0.90.rsu です。二つ目のファームウェアのバージョンは 1.00 で、ファイル名は userprog_v1.00.rsu です。初期状態では、FOTA に対応したブートローダ込みのバージョン 0.90 のファームウェアが書き込まれています。

ファームウェアのバージョンの違いを以下に示します。現在使用しているファームウェアのバージョンは、初期画面で確認できます。バージョンによって、メニュー画面に違いがあります。

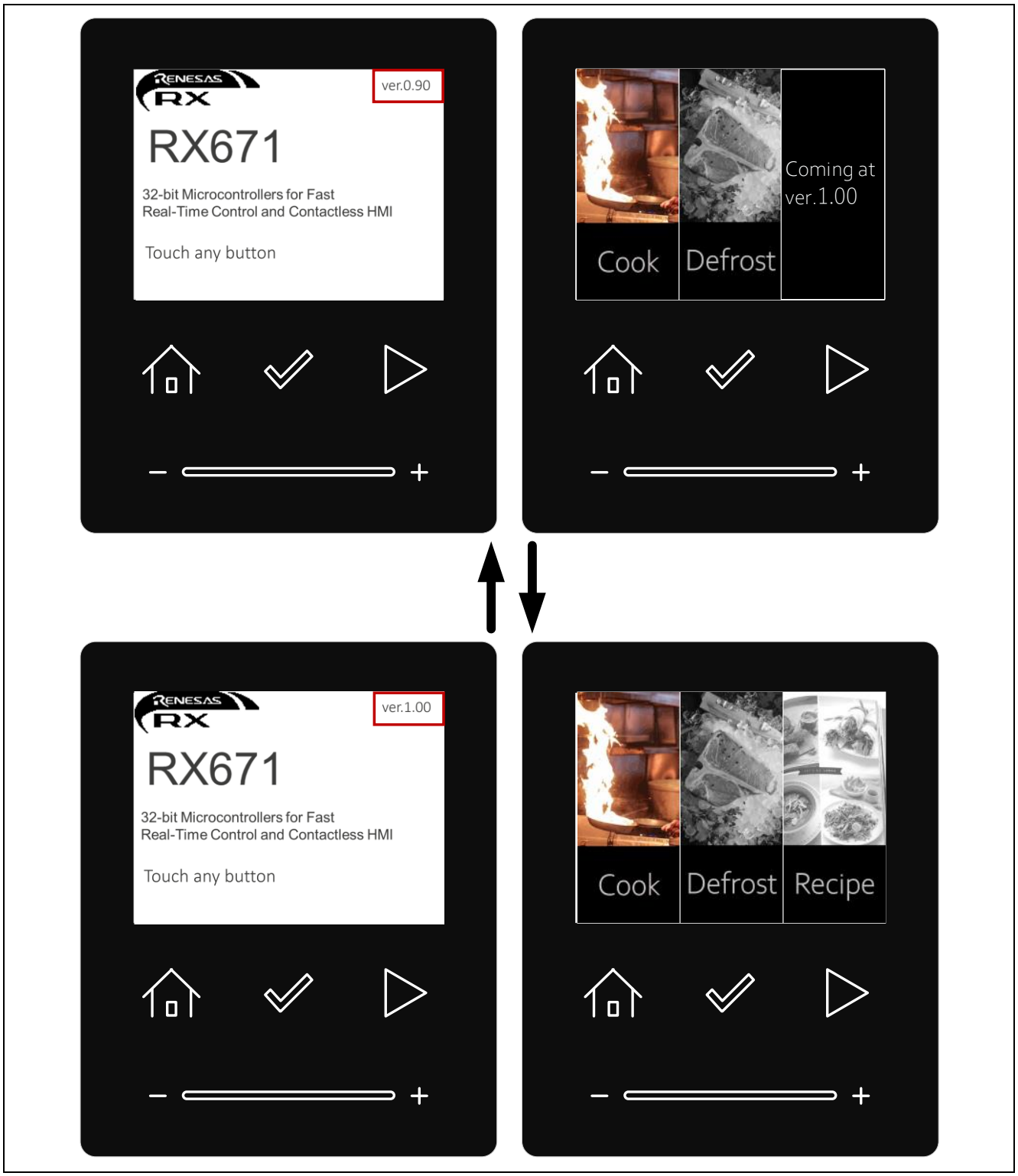


図 6-1 ファームウェアのバージョンによる違い

1. 「RX65N における Amazon Web Services を利用した FreeRTOS OTA の実現方法」の「1.2 Amazon S3 バケットの作成」の説明に従って、OTA 更新ファームウェアである userprog_v0.90.rsu、または userprog_v1.00.rsu を Amazon S3 バケットにアップロードし保存します。

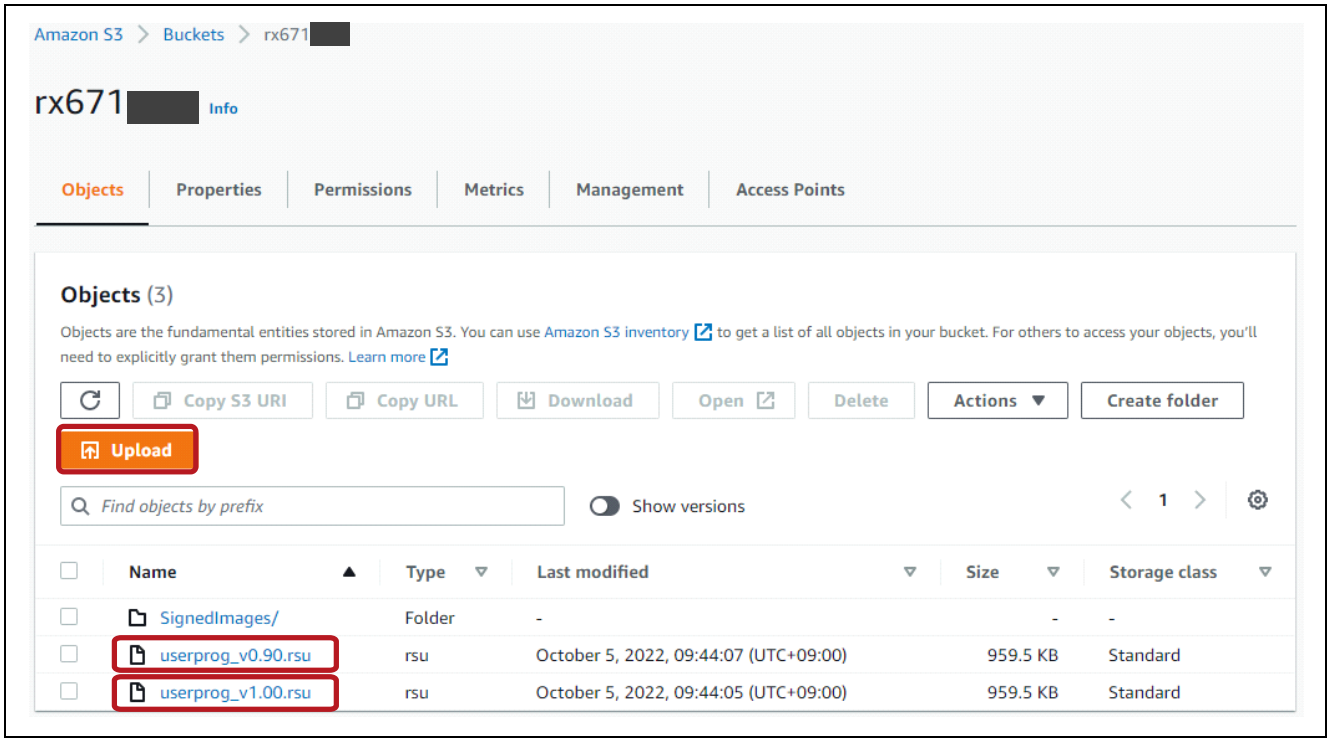


図 6-2 userprog.rsu アップロード

2. RX671 PoC のファームウェアを更新するためのジョブを作成します。

AWS IoT Jobs は、1 台または複数台の接続済みデバイスに待機中の「ジョブ」があることを通知します。ジョブを使用することで、大量のデバイスを管理したり、デバイス上のファームウェアやセキュリティ証明書を更新したり、デバイスの再起動や診断などの管理タスクを実行したりできます。

—[AWS IoT] → [管理] → [ジョブ] → [作成] → [OTA 更新ジョブの作成] → ジョブ名を設定 → [次へ]を選択します。

—FreeRTOS OTA 更新ジョブを次のように作成します。

モノの名前を選択します。（図 6-3 (a)、図 7-11）

コード署名プロファイルを選択します。（図 6-3 (b)）

S3 から FOTA を行うファームウェアイメージを選択します。（図 6-3 (c)）

IAM ロールを選択します。（図 6-3 (d)）

—[Next]をクリックします。

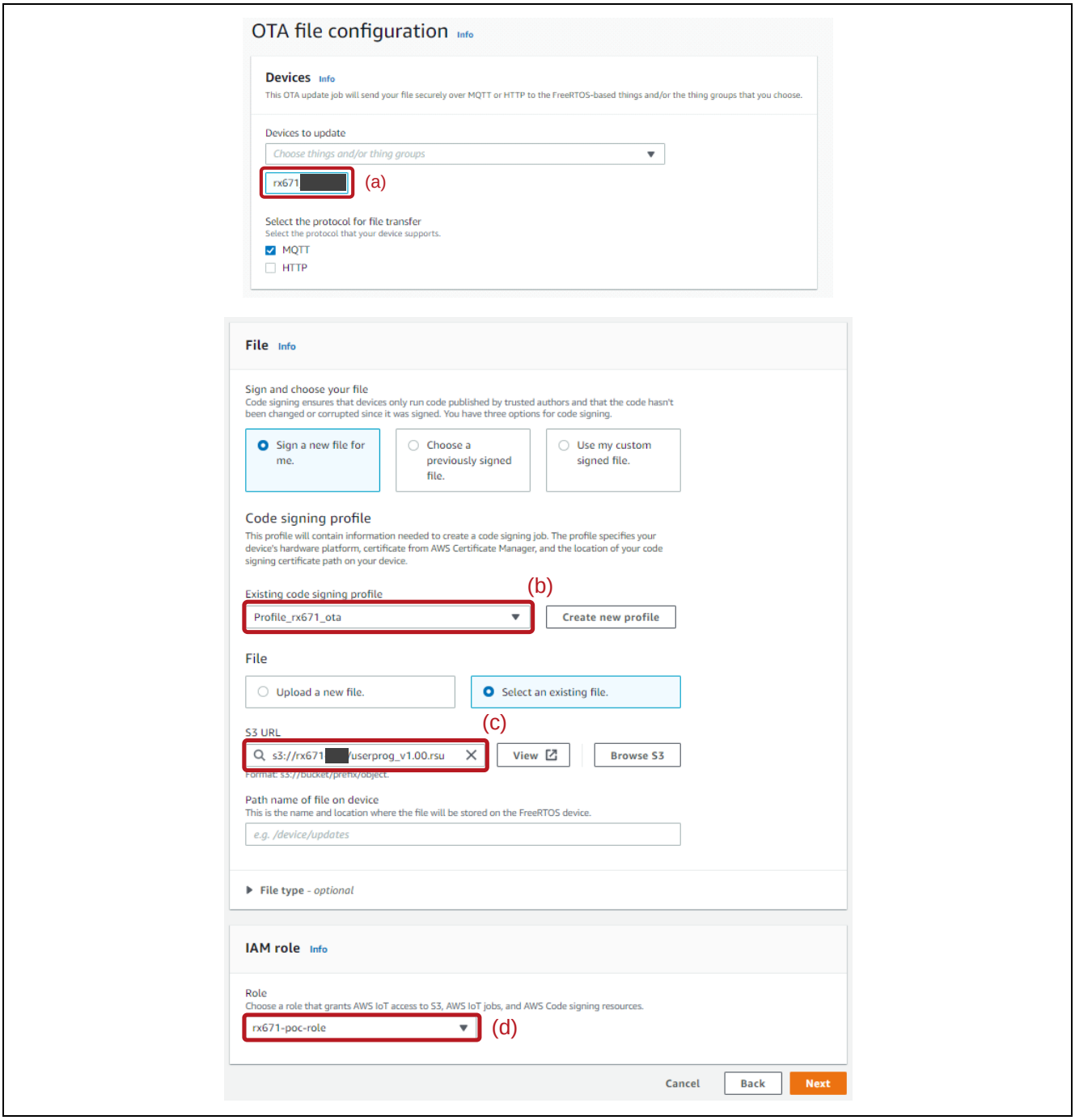


図 6-3 Job 作成（1）

3. [Create job]をクリックします。

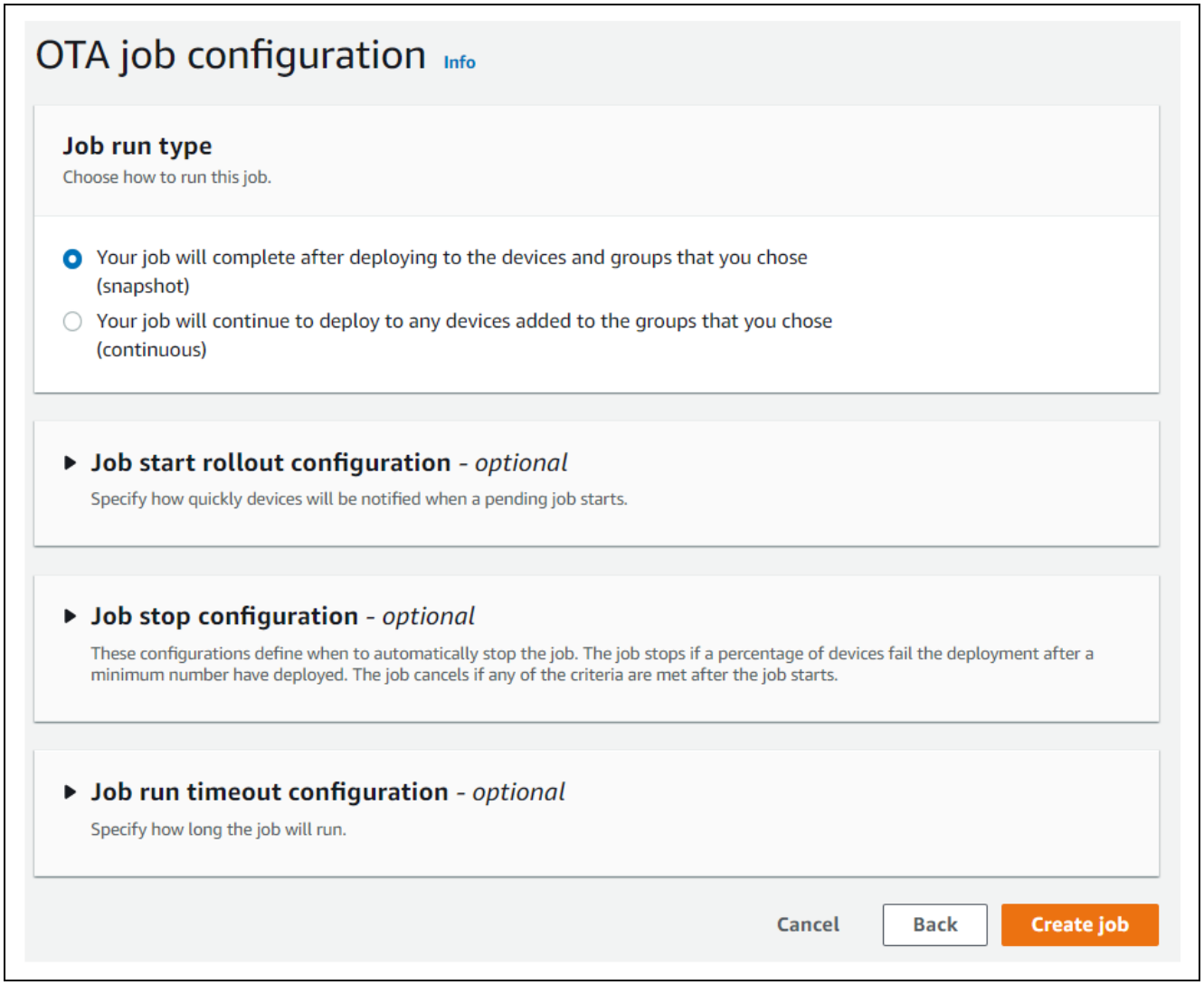


図 6-4 Job 作成（2）

4. Tera Term を起動し、ファームウェアが更新されたことを確認します。
OTA デモバージョンは 1.00 であり、更新が正常に行われています。

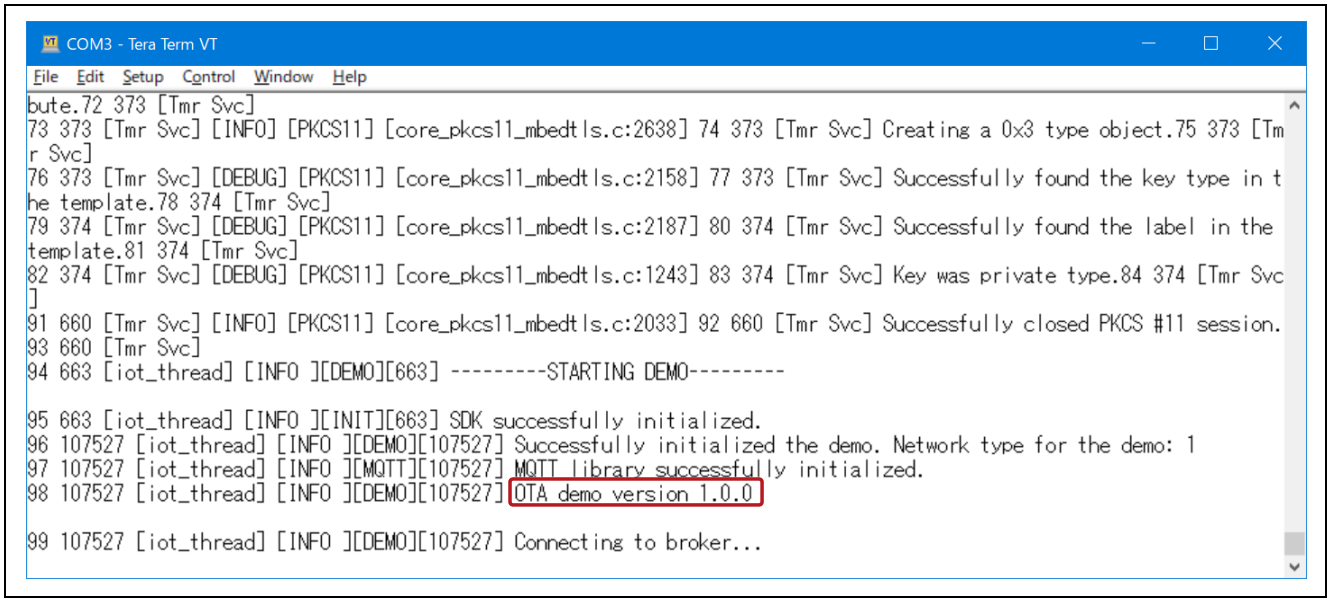


図 6-5 実行結果確認

Tera Term の設定を以下に示します。Tera Term をインストールしていない場合は、次からダウンロードしてください。https://ttssh2.osdn.jp/index.html.en

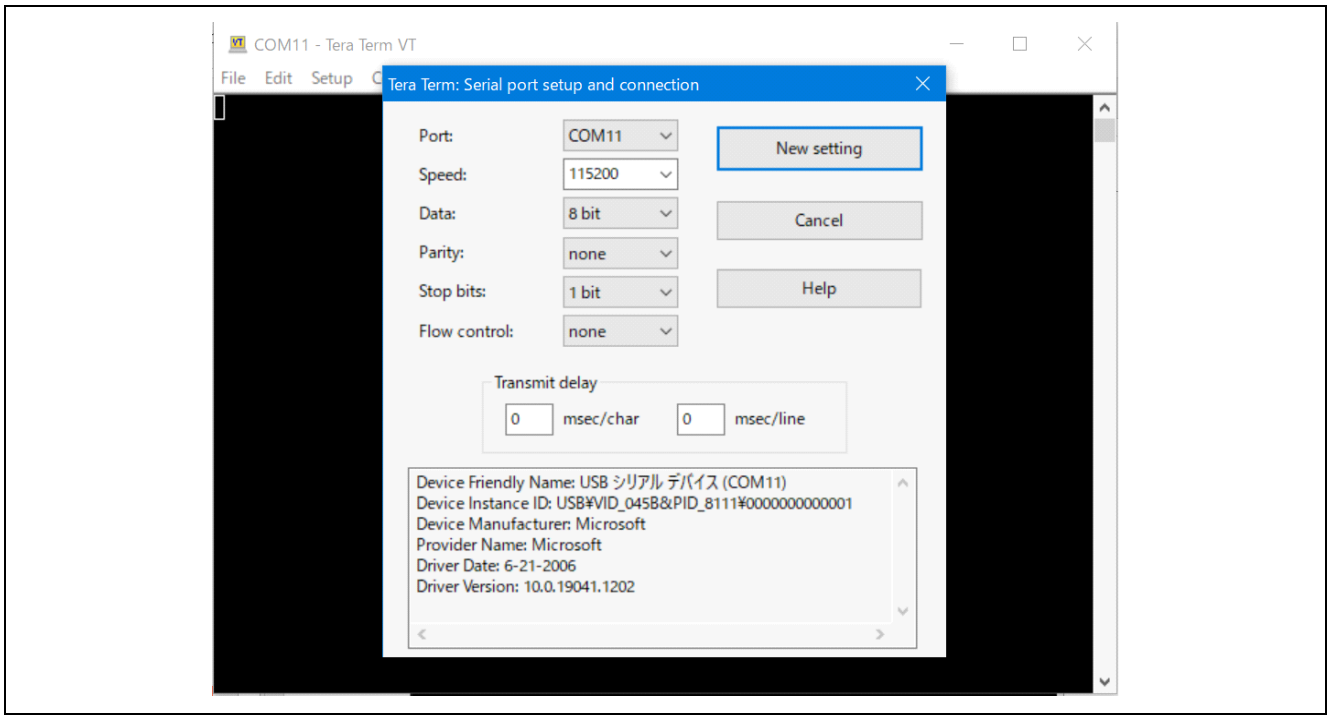


図 6-6 TeraTerm

5. ジョブのステータスが「Succeeded」であることを確認します。

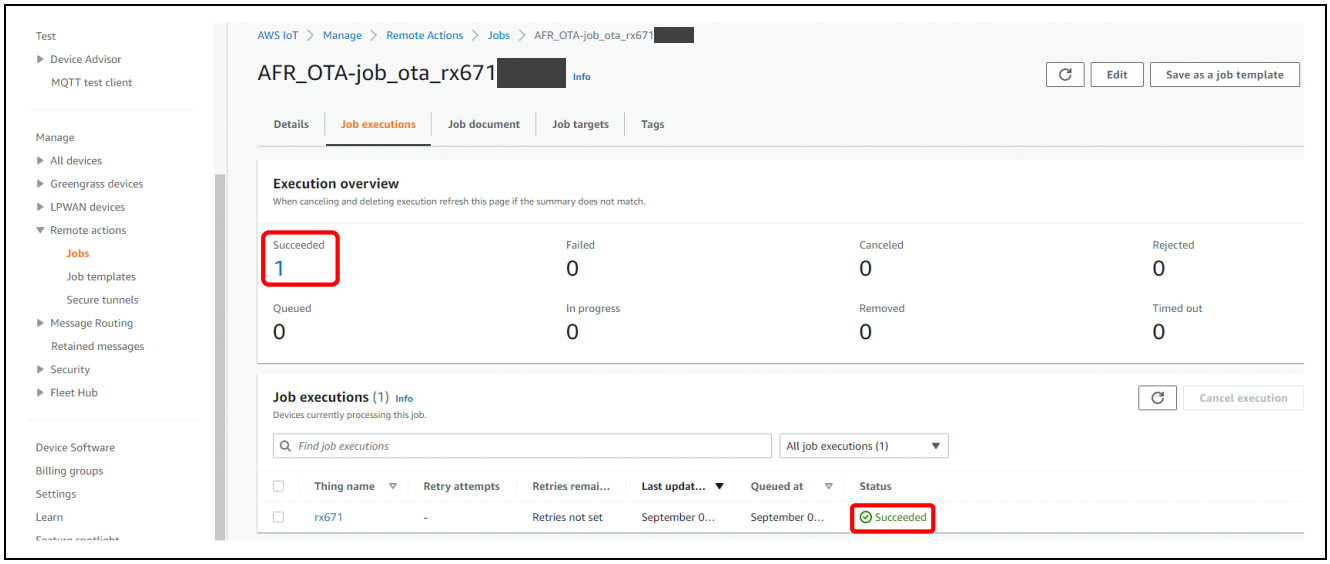


図 6-7 Succeeded 確認

7.　OTA に対応したユーザプログラムの作成方法

本項では、OTA によりクラウドからプログラムを書き換える方法を説明します。
プログラムの書き換えはバックグラウンドで行われ、次の電源投入時に自動的に新しいプログラムに切り替わります。
最初に、ユーザは Amazon FreeRTOS パッケージのバージョンを選択できます。選択されたバージョンが自動的に GitHub からダウンロードされ、プロジェクトにインポートされます。そのため、ユーザは Amazon FreeRTOS の設定とプログラムの作成に集中できます。

7.1　AWS の準備

クラウドからの OTA を行うにあたりクラウド環境を準備する必要があります。
クラウドは AWS を使用します。AWS の準備については、以下を参照してください。

- RX65N における Amazon Web Services を利用した FreeRTOS OTA の実現方法（R01AN5549）

7.2　ヘッダファイルのインポートと設定、および aws_demos と boold_loader の構築

Amazon FreeRTOS プロジェクトのインポート方法を下図に示します。

1. e² studio を起動します。
2. [ファイル] → [インポー...]を選択します。
3. [Renesas GitHub FreeRTOS (with IoT libraries) Project]を選択します。

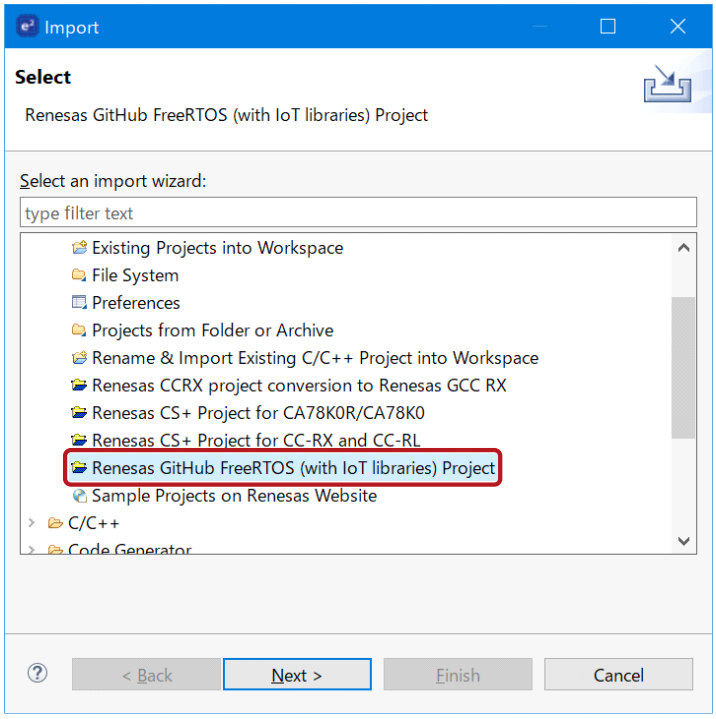


図 7-1 プロジェクトのインポート

4. [Check for more version…]をクリックし、「FreeRTOS (with IoT libraries)」ダイアログを表示させます。

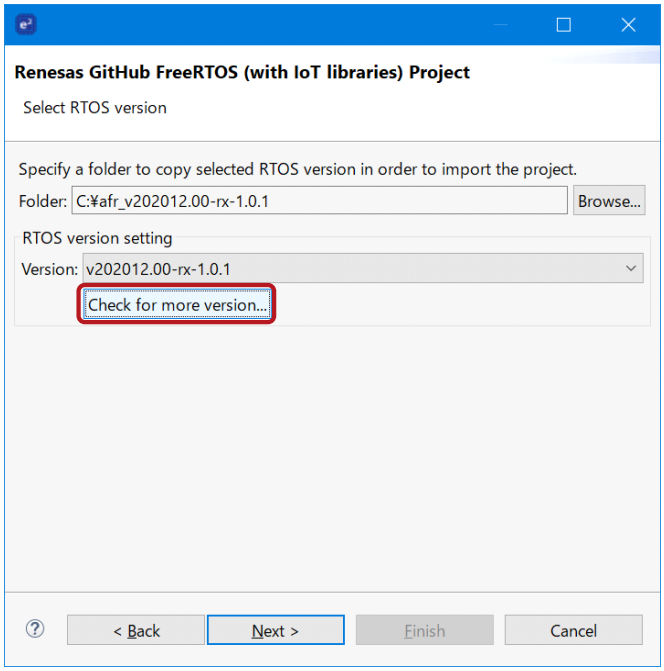


図 7-2 「FreeRTOS (with IoT libraries)」ダイアログ

5. 最新バージョンを選択します（最新バージョンが表示されない場合、e² studio のワークスペースを新規に作成してください）。

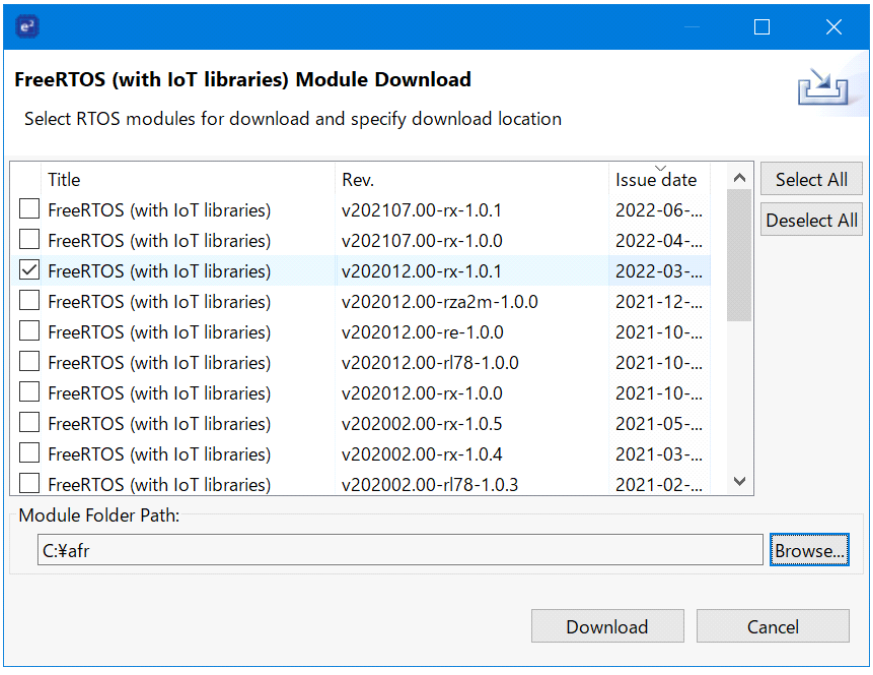


図 7-3 OS のバージョンを選択する

6. エンドユーザライセンス契約に同意します。

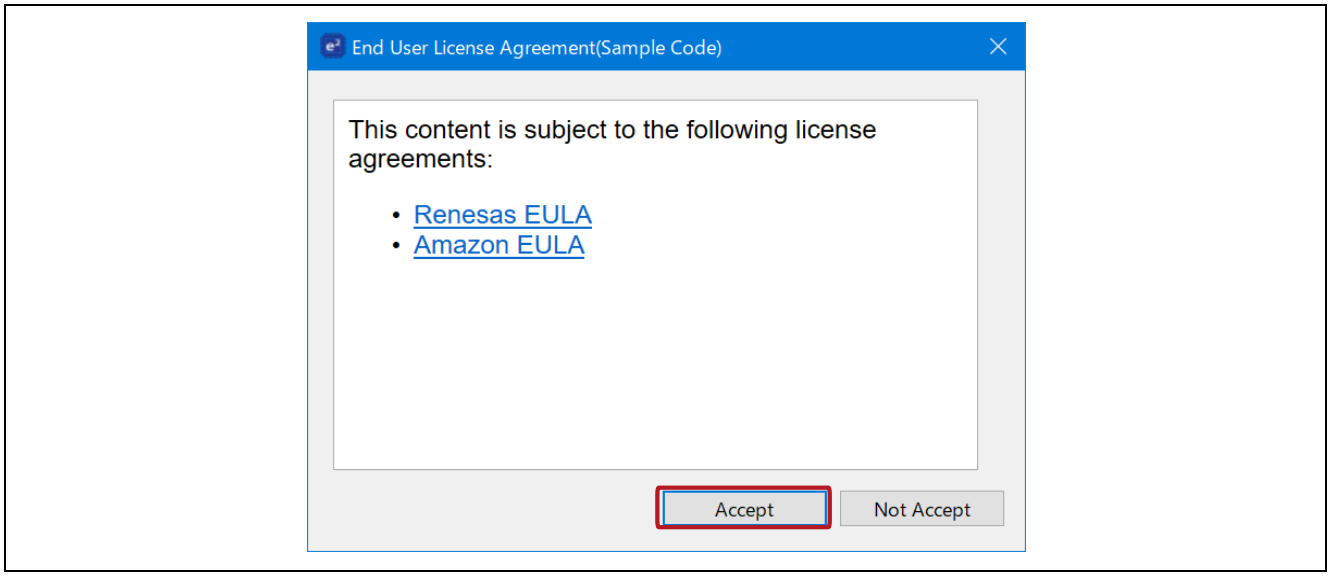


図 7-4 エンドユーザライセンス契約に同意

7. ダウンロードが完了するまで待ちます。

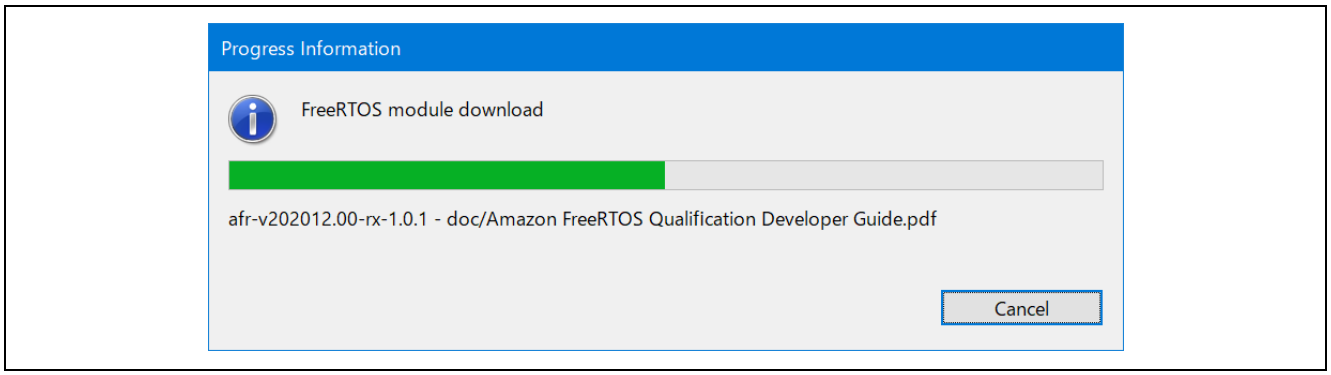


図 7-5 ダウンロード待ち

8. インポートするプロジェクトを選択します。[aws_demos]と[boot_loader]プロジェクトを選択します。

本デモで使用している aws_demos 及び boot_loader は 2023 年 5 月現在正式リリースされておりません。
個別提供をご希望の方は弊社代理店及び営業窓口までお問い合わせください。

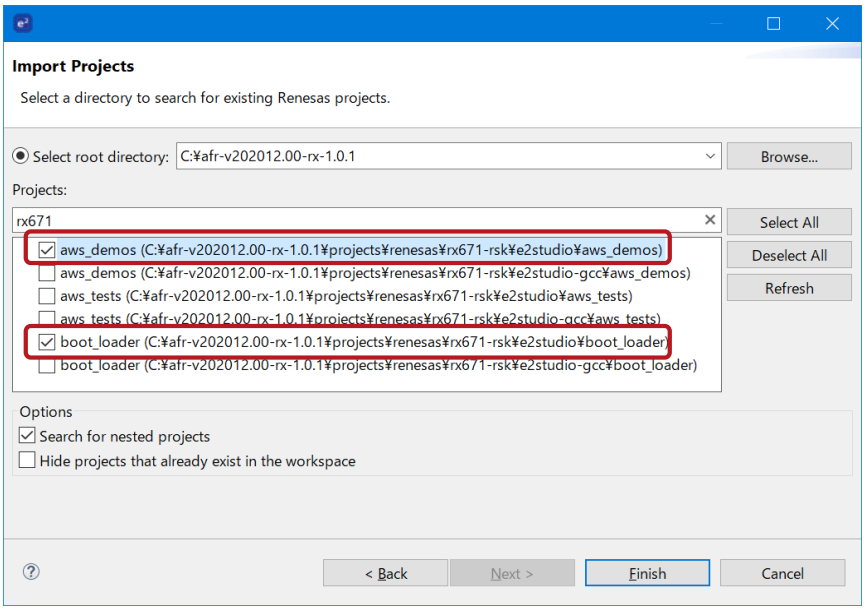
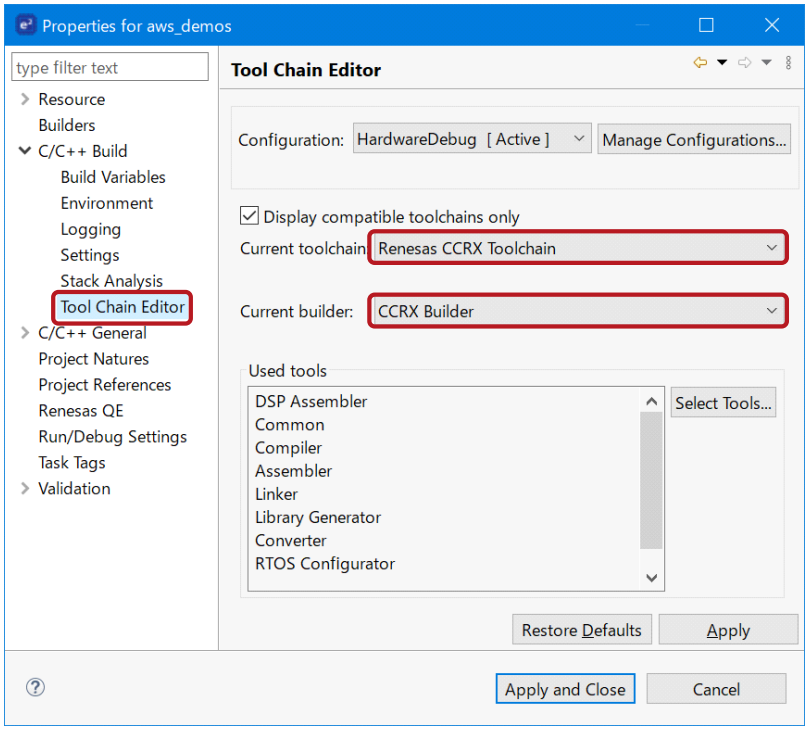


図 7-6 インポートするプロジェクトの選択

9. 両方のプロジェクトの[プロジェクト] → [プロパティ] → [C/C++ビルド] → [ツールチェーン・エディタ]を開き、「ツールチェーン」と「ビルダー」を選択して、ツールチェーンを設定します。また、[設定] → [Toolchain]を選択してバージョンを設定します。



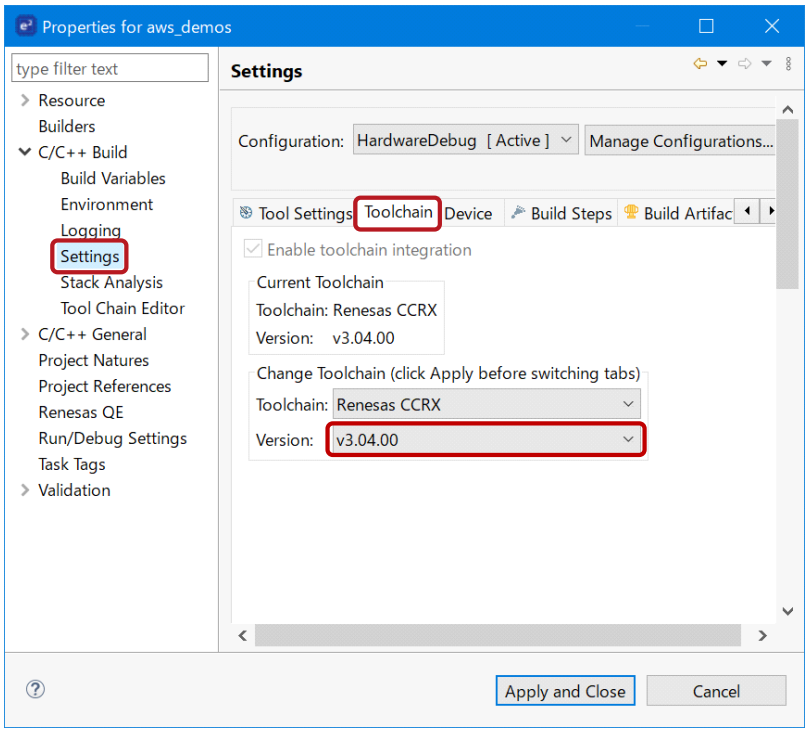


図 7-7 ツールチェーンとバージョン指定

10. [プロジェクト] → [プロパティ] → [C/C++ビルド] → [設定] → [Converter] → [出力]を開き、
[Motorola S format file（モトローラ S 形式ファイルを出力する）]を設定します。

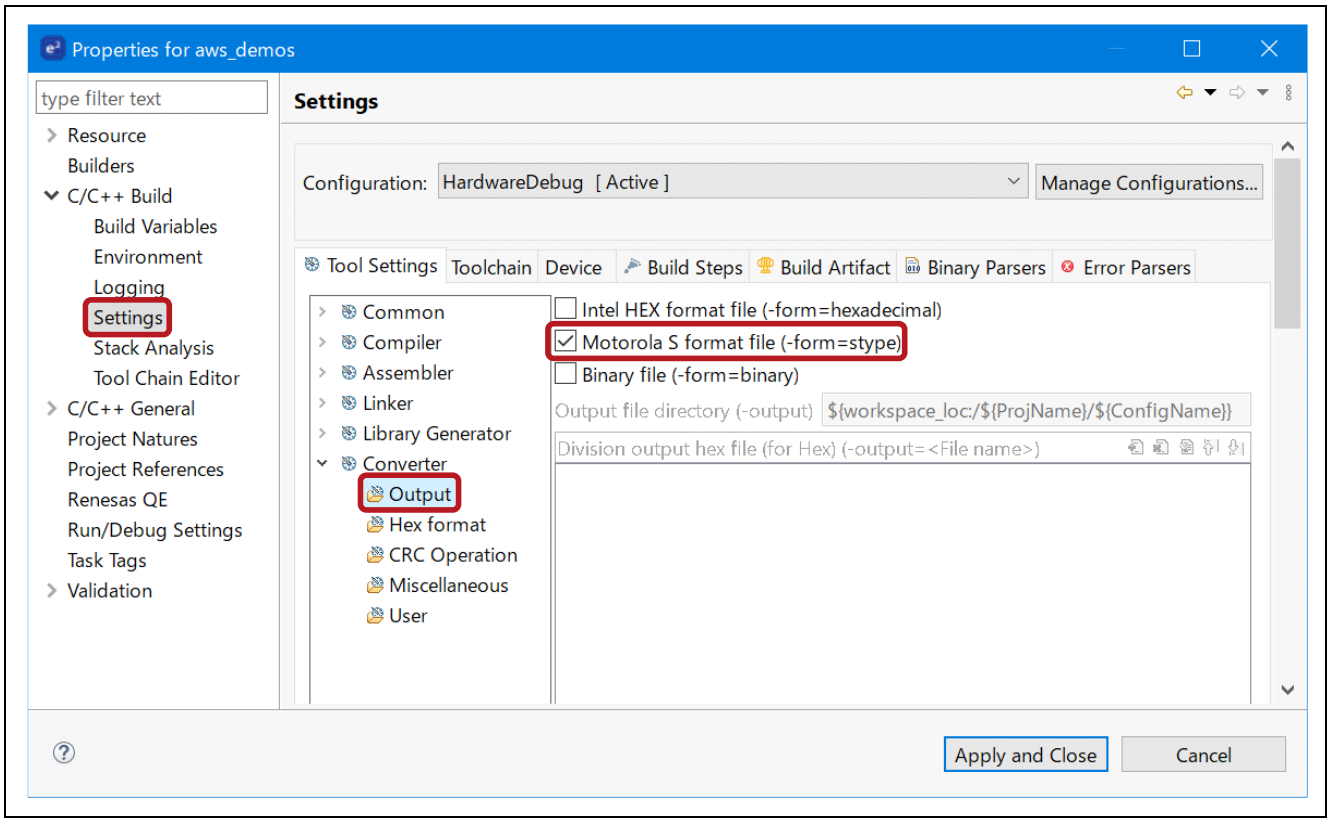


図 7-8 Motorola S format ファイル出力設定

11. 公開鍵を入力します。
    boot_loader プロジェクトで以下のファイル
    projects¥renesas¥rx671-rsk¥e2studio¥boot_loader¥src¥key¥code_signer_public_key.h
    を開き公開鍵を入力します。
    「ルネサス MCU ファームウェアアップデートの設計方針」の「7.3 OpenSSL での ECDSA+SHA256 用の鍵ペア生成方法」を参照して、公開鍵を生成してください。
    完了したらビルドし、boot loader プロジェクトの boot_loader.mot ファイルを生成してください。

本デモで使用している aws_demos 及び boot_loader は 2023 年 5 月現在正式リリースされておりません。
個別提供をご希望の方は弊社代理店及び営業窓口までお問い合わせください。

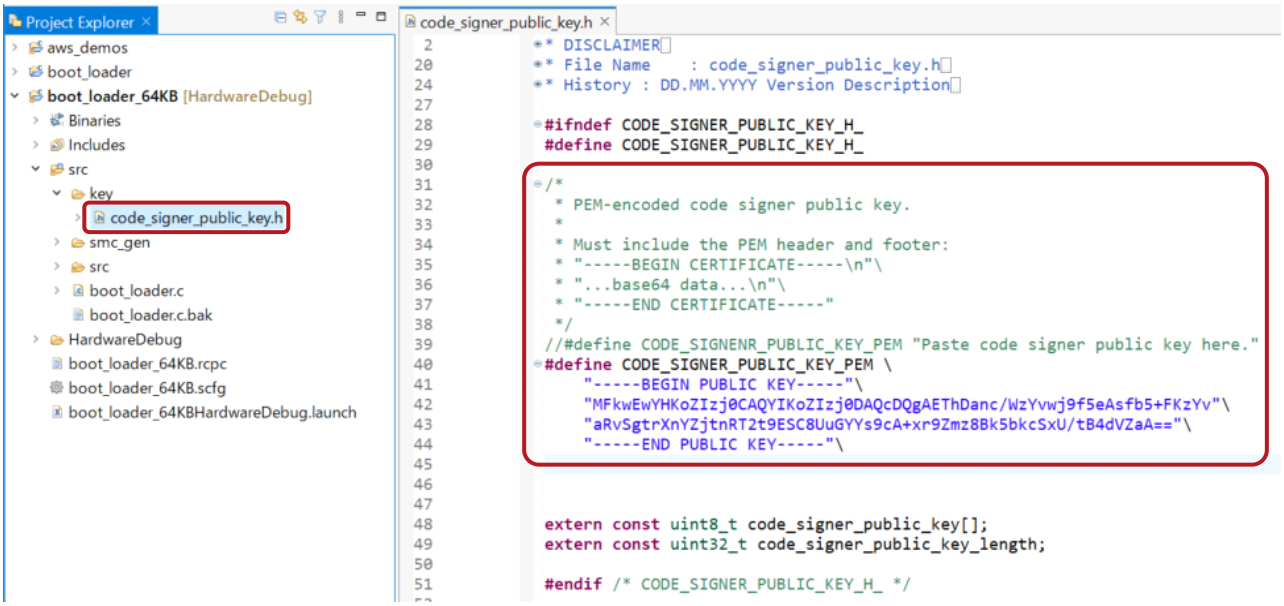


図 7-9 公開鍵の入力

12. AWS IoT コンソールを開きます。
—AWS IoT コンソールに移動します。
—[設定]を選択します。エンドポイントをメモします（図 7-10 (e)）。

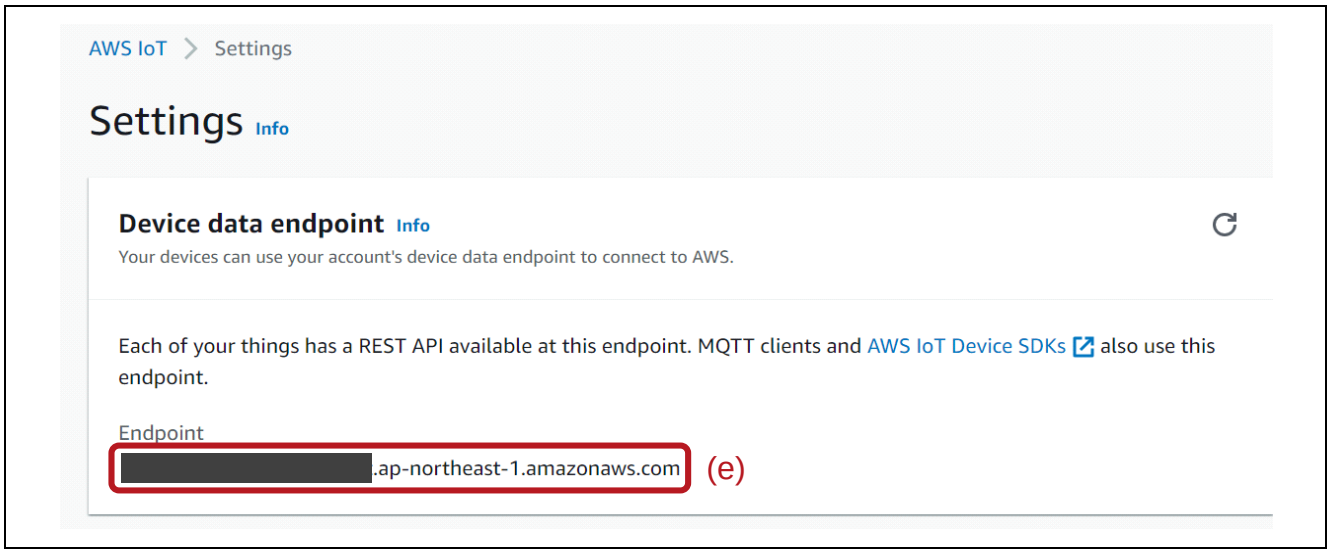


図 7-10 AWS エンドポイント確認

—[管理] → [モノ]を選択します。AWS IoT のモノの名前をメモします（図 7-11 (f)）。

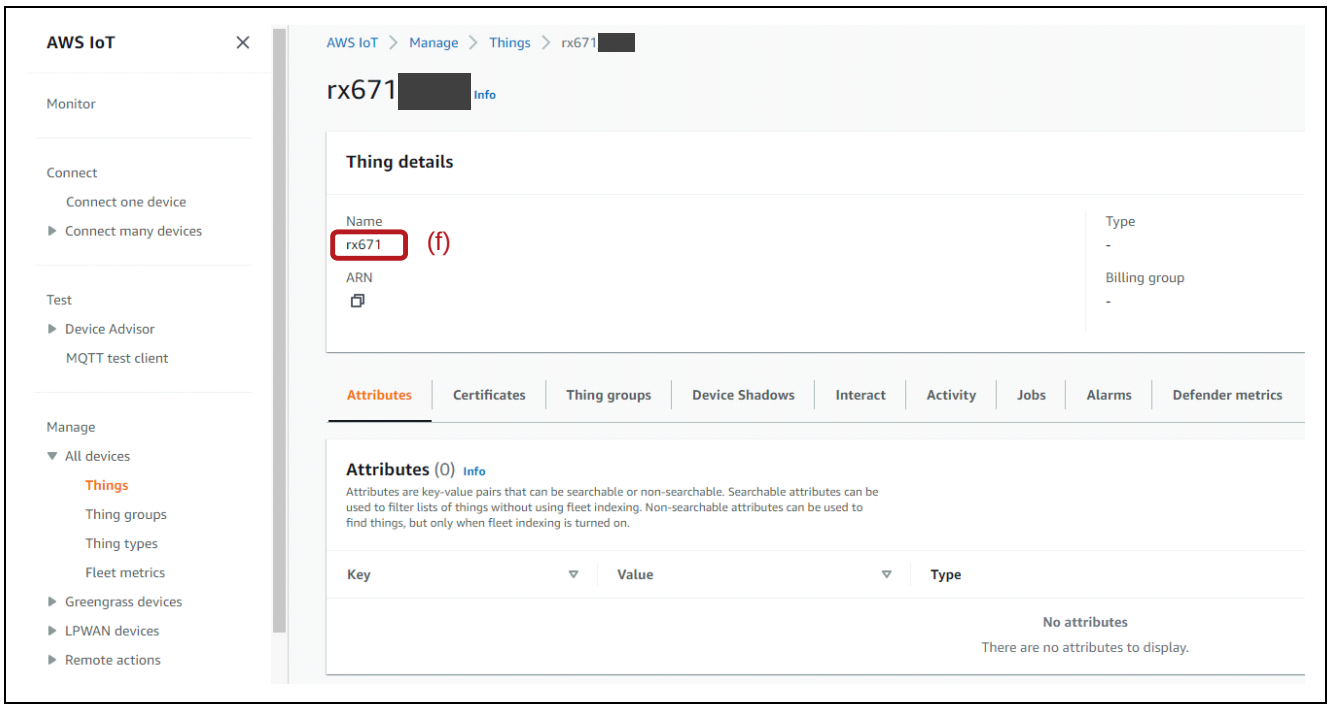


図 7-11 モノの名前

13. aws_demo プロジェクトを開きます。
—/demos/include/aws_clientcredential.h を開いて次の値を指定します。
#define clientcredentialMQTT_BROKER_ENDPOINT　"図 7-10 (e)でメモしたエンドポイント"
#define clientcredentialIOT_THING_NAME　"図 7-11 (f)でメモしたモノの名前"

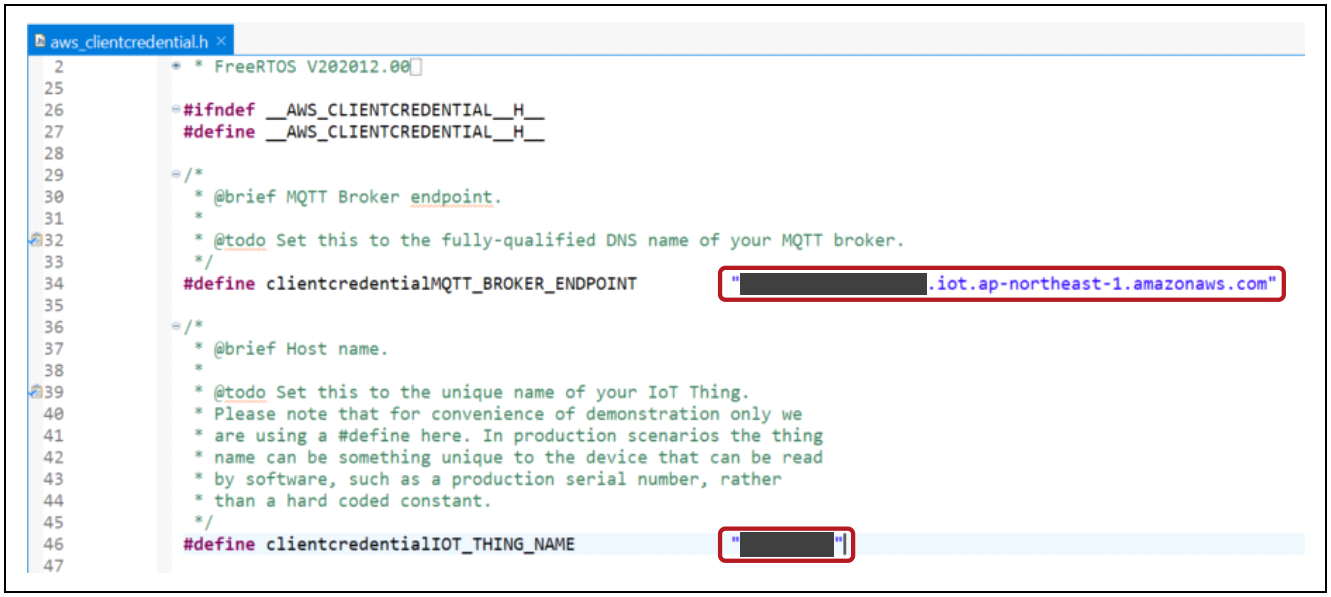


図 7-12 エンドポイントとモノの名前の入力

14. 「Certificate Configuration Tool」を開きます。
—7.1 の 5.でダウンロードした FreeRTOS が格納されたパスに移動します。
—[tools] → [certificate_congiguration] → CertificateConfigurator.html を開きます。
—「RX65N における Amazon Web Services を利用した FreeRTOS OTA の実現」の 1.1 の(4)でダウンロードした証明書の PEM ファイルとプライベートキーの PEM ファイルをインポートします。
—awa_clientcredential_keys.h を生成します。

図 7-13 clientcredential キー生成

15. 再度 aws_demo プロジェクトを開きます。
    ―上記で生成した aws_clientcredentia_keys.h を/demos/include/にあるファイルと置き換えます。
    ―/demos/include/aws_ota_codesigner_certificate.h を開いて次の値を指定します。
    signingcredentialSIGNING_CRETIFICATE_PEM [] = “xxxx”;
    “xxxx”には、secp256r1.crt の値を指定します。
    secp256r1.crt の値をコピーするときに、各行の最後に”¥”が必要です。忘れずに追加してください。
    secp256r1.crt については、「ルネサス MCU ファームウェアアップデートの設計方針」の
    「7.3 OpenSSL での ECDSA+SHA256 用の鍵ペア生成方法」を参照してください。

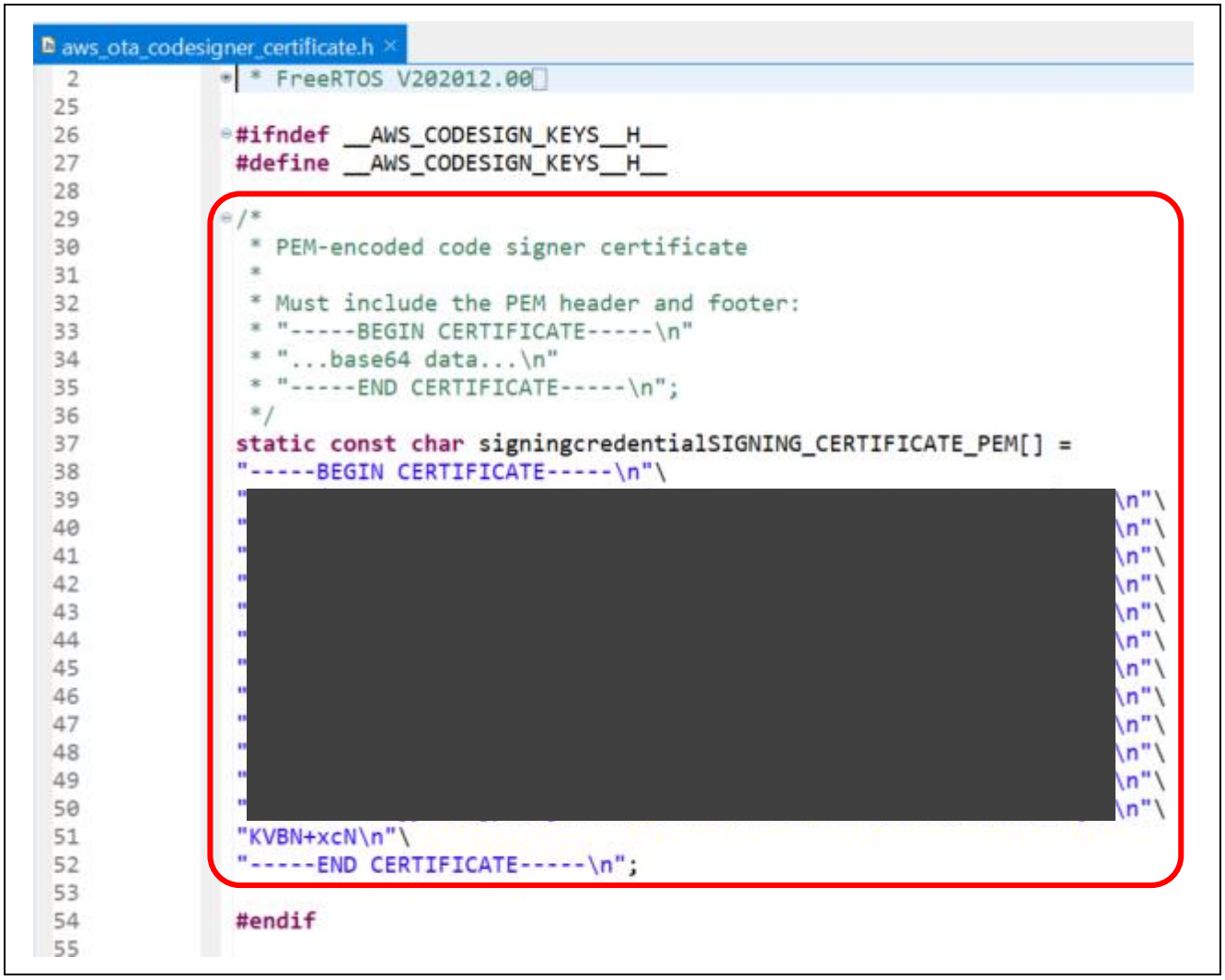


図 7-14 clientcredential 入力

7.3　ファームウェアの初期バージョンのインストール

1. FreeRTOSApplicationConfig.h の設定を確認します。

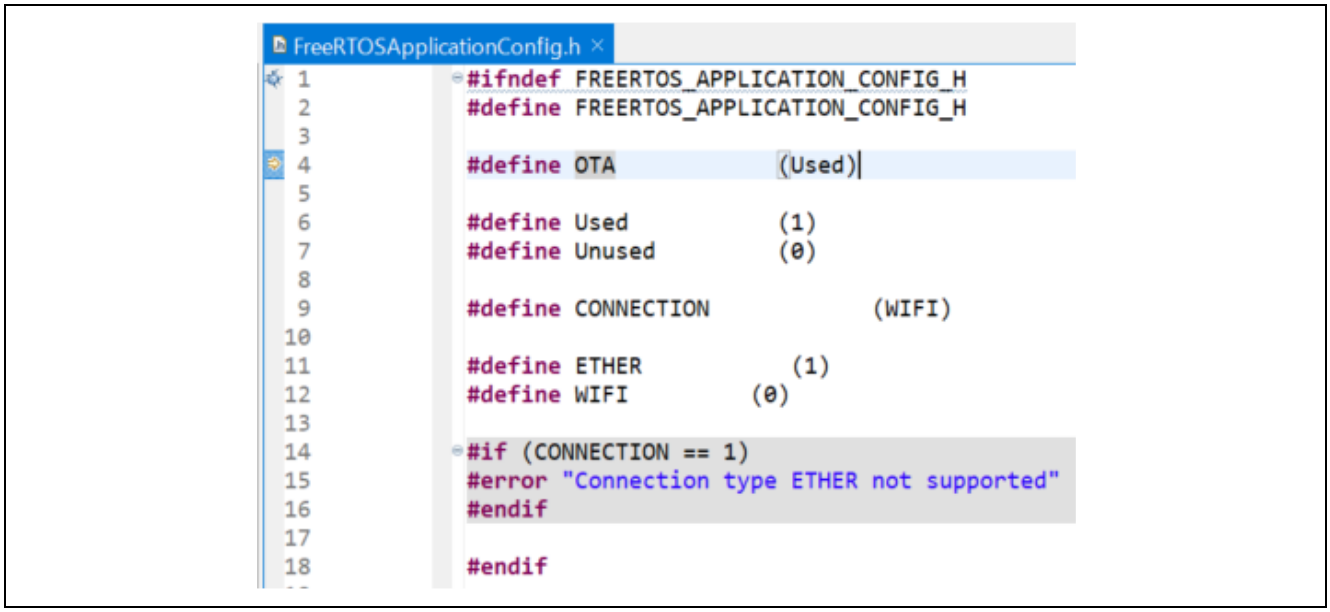


図 7-15 設定確認

2. amazon-freertos/demos/include/aws_application_version.h を開き、ファームウェアの初期バージョンを 0.90 に設定します。

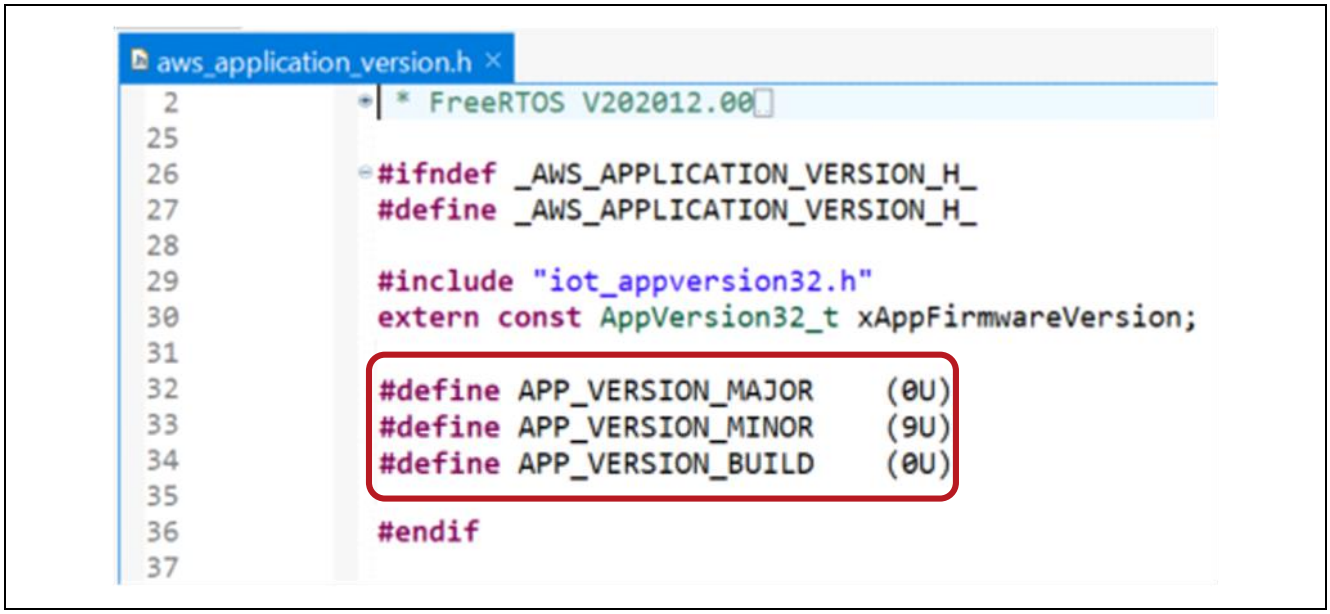


図 7-16 ファームウェア初期バージョン定義

3. [プロジェクト] → [プロパティ] → [C/C++ビルド] → [設定] → [ツール設定]タブ → [Linker] → [セクション] → […]で Section Viewer を開き、セクションを以下のように設定します。

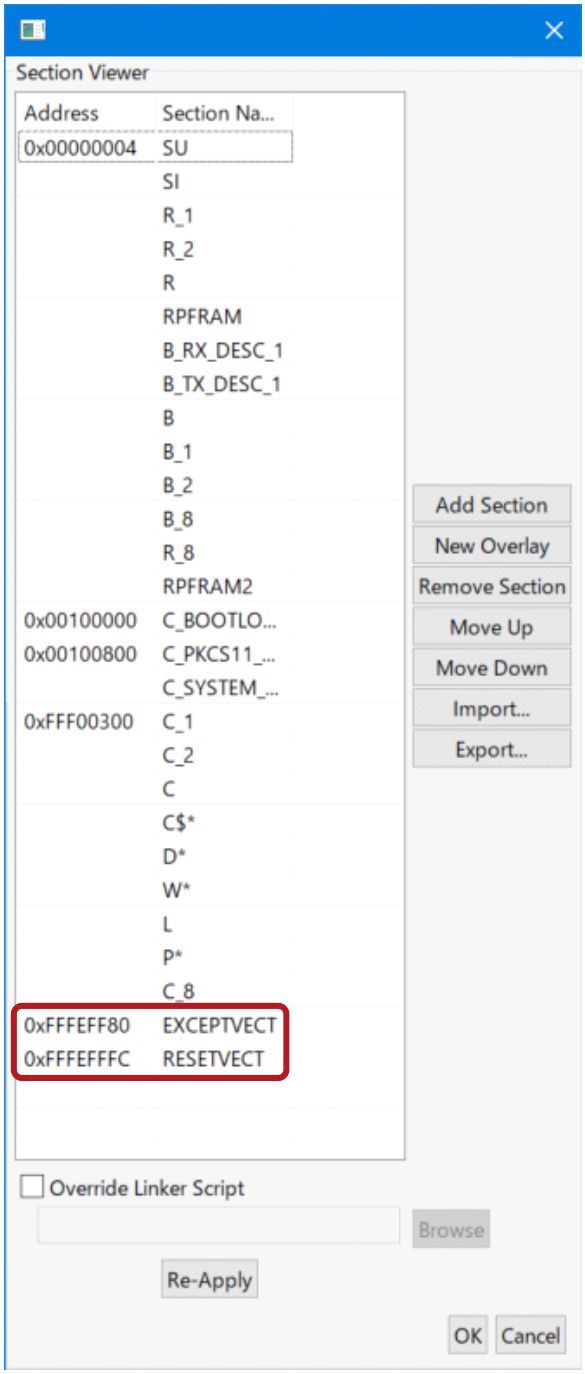


図 7-17 セクション設定

4. aws_demos.mot ファイルをビルドして作成します。

5. Renesas Secure Flash Programmer から userprog.mot を作成します。
   userprog.mot は、aws_demos.mot と boot_loader.mot を組み合わせたファイルです。このファイルを RX671 PoC に読み込むことで、初期ファームウェアをインストールできます。
   Renesas Secure Flash Programmer release 1.0.1 からダウンロードし Renesas Secure Flash Programmer.exe を実行します（一緒に置かれているファイルも必要ですのでダウンロードしてください）。
   —[Initial Firm]タブを選択して、下図のようにパラメータを設定します。
   - Private Key Path：secp256r1.privatekey へのパス
   - Boot Loader File Path：boot_loader.mot へのパス
     （¥projects¥renesas¥rx671-rsk¥e2studio¥boot_loader¥HardwareDebug）
   - Bank 0 User Program File Path：aws_demos.mot へのパス
     （¥projects¥renesas¥rx671-rsk¥e2studio¥aws_demos¥HardwareDebug）

   —[Generate]をクリックすると、userprog.mot が生成され、init_firmware フォルダに保存されます。
   Generate succeeded と表示されたことを確認してください。

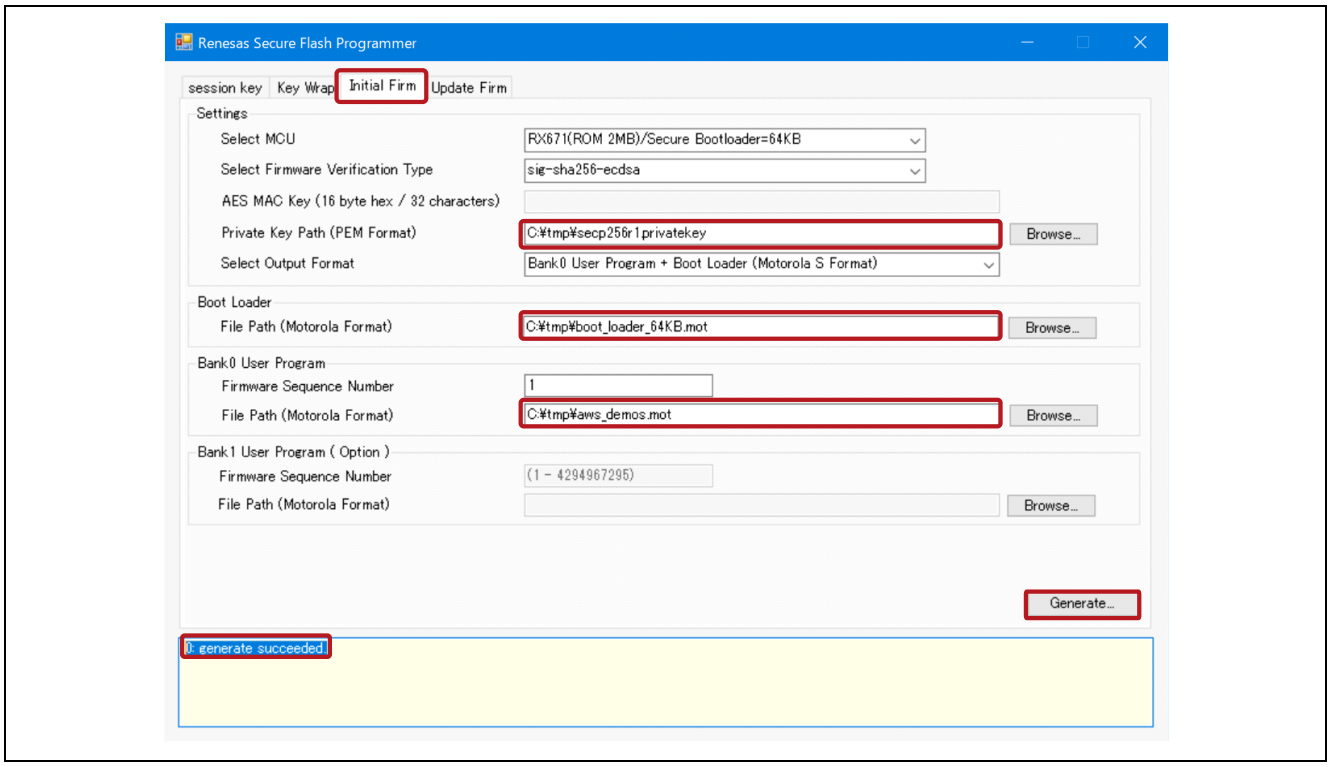


図 7-18 userprog.mot 作成

6. RX671 PoC のフラッシュ ROM を消去します。
   —以下から最新の Renesas Flash Programmer をダウンロードしてください。
   https://www.renesas.com/rfp
   —Renesas Flash Programmer で以下のプロジェクトを開いてください。
   ¥vendors¥renesas¥rx_mcu_boards¥boards¥rx671-rsk¥aws_demos¥flash_project¥erase_from_bank¥erase.rpj
   — [Operation]タブを選択し、[Start]をクリックしてフラッシュ ROM を消去します。

要修正

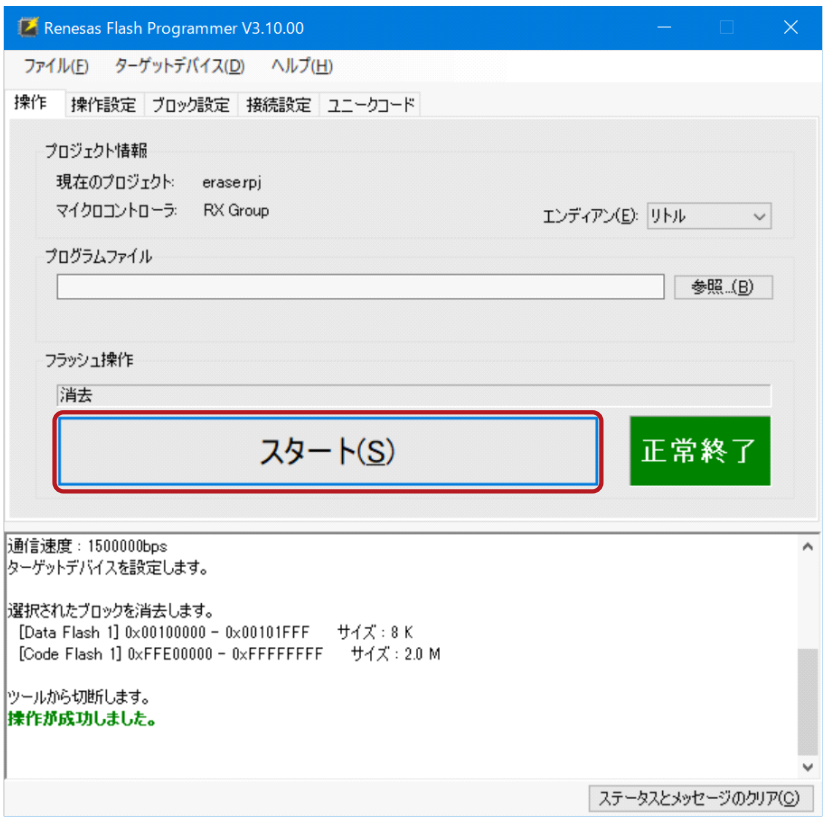


図 7-19 Flash ROM 消去

7. RX671 PoC に初期ファームウェアを書き込みます。
   —Renesas Flash Programmer で新しいプロジェクトを作成してください。（例：flash_project.rpj）
   —[Operation]タブを選択し、Program File の init_firmware フォルダに保存されている userprog.mot を設定してください。
   —[Start]をクリックします。

要修正

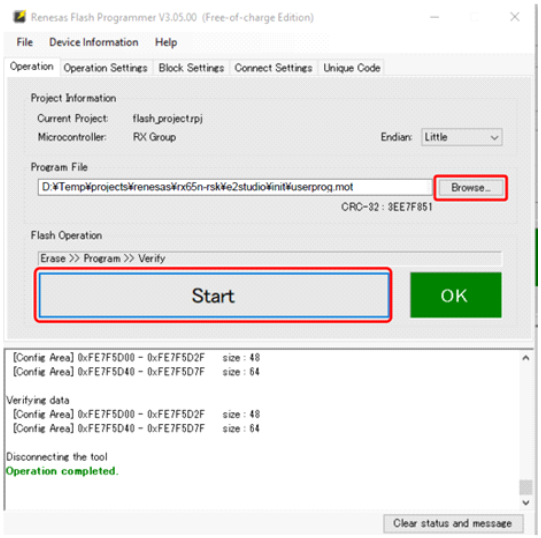


図 7-20 初期ファームウェア書き込み

8. Tera Term を起動します。
　Tera Term の設定を以下に示します。

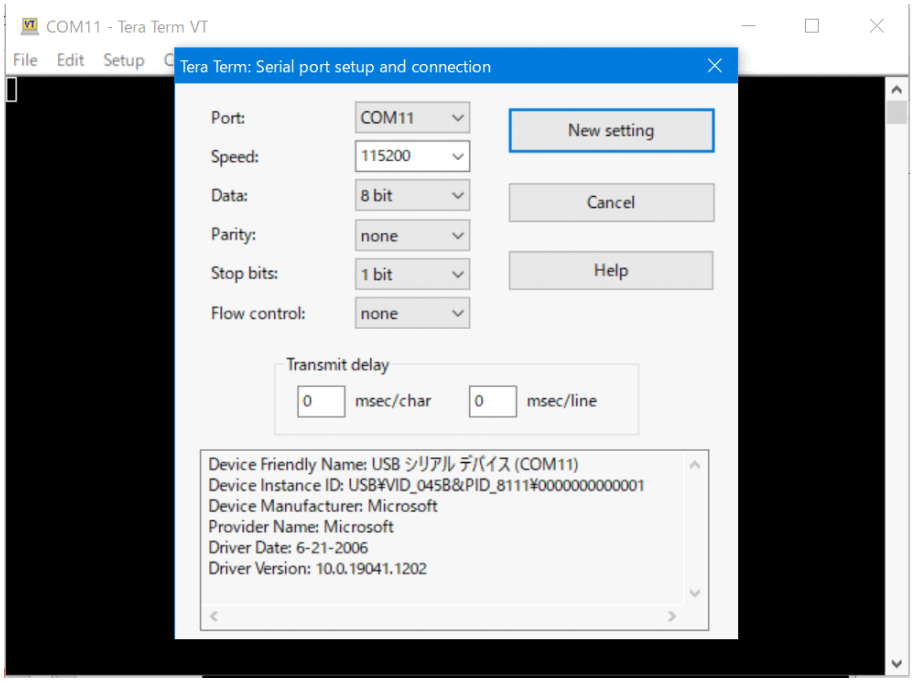


図 7-21 Tera Term

RX671 PoC にはバージョン 0.90（初期バージョン）がインストールされています。これで、RX671 PoC は OTA 更新を受信できるようになりました。
出力ログを以下に示します。

```
------------------------------------------------
RX671 secure boot program
------------------------------------------------
Checking data flash ROM status.
Loading user code signer public key: not found.
provision the user code signer public key: OK.
Checking code flash ROM status.
bank 0 status = 0xfc [LIFECYCLE_STATE_INITIAL_FIRM_INSTALLED]
bank 1 status = 0xff [LIFECYCLE_STATE_BLANK]
bank info = 1. (start bank = 0)
started 10us software timer using CMT channel 0.
integrity check scheme = sig-sha256-ecdsa
bank0(execute area) on code flash integrity check...OK
erase bank1 secure boot mirror area...OK
OK
copy secure boot (part2) from bank0 to bank1...OK
jump to user program
0 83 [Tmr Svc] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:449] 1 83 [Tmr Svc] PKCS #11
module was successfully initialized.2 83 [Tmr Svc]
3 83 [Tmr Svc] [INFO] [PKCS11] [core_pkcs11_mbedtls.c:1504] 4 83 [Tmr Svc] PKCS #11
successfully initialized.5 83 [Tmr Svc]
6 83 [Tmr Svc] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:1717] 7 83 [Tmr Svc] Successfully
Returned a PKCS #11 slot with ID 1 with a count of 1.8 83 [Tmr Svc]
9 83 [Tmr Svc] [WARN] [PKCS11] [core_pkcs11_mbedtls.c:1749] 10 83 [Tmr Svc]
C_GetTokenInfo is not implemented.11 83 [Tmr Svc]
12 83 [Tmr Svc] [WARN] [PKCS11] [core_pkcs11_mbedtls.c:1839] 13 83 [Tmr Svc] C_InitToken
is not implemented.14 83 [Tmr Svc]
70 373 [Tmr Svc] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:471] 71 373 [Tmr Svc]
Successfully found object class attribute.72 373 [Tmr Svc]
73 373 [Tmr Svc] [INFO] [PKCS11] [core_pkcs11_mbedtls.c:2638] 74 373 [Tmr Svc] Creating a
0x3 type object.75 373 [Tmr Svc]
```

```
76 373 [Tmr Svc] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:2158] 77 373 [Tmr Svc]
Successfully found the key type in the template.78 374 [Tmr Svc]
79 374 [Tmr Svc] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:2187] 80 374 [Tmr Svc]
Successfully found the label in the template.81 374 [Tmr Svc]
82 374 [Tmr Svc] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:1243] 83 374 [Tmr Svc] Key was
private type.84 374 [Tmr Svc]
91 660 [Tmr Svc] [INFO] [PKCS11] [core_pkcs11_mbedtls.c:2033] 92 660 [Tmr Svc]
Successfully closed PKCS #11 session.93 660 [Tmr Svc]
94 663 [iot_thread] [INFO ][DEMO][663] ---------STARTING DEMO---------

95 663 [iot_thread] [INFO ][INIT][663] SDK successfully initialized.
96 107527 [iot_thread] [INFO ][DEMO][107527] Successfully initialized the demo. Network
type for the demo: 1
97 107527 [iot_thread] [INFO ][MQTT][107527] MQTT library successfully initialized.
98 107527 [iot_thread] [INFO ][DEMO][107527] OTA demo version 0.9.0

99 107527 [iot_thread] [INFO ][DEMO][107527] Connecting to broker...

100 107527 [iot_thread] [INFO ][DEMO][107527] MQTT demo client identifier is rx671_POC
(length 9).
101 109439 [iot_thread] [WARN] [PKCS11] [core_pkcs11_mbedtls.c:1499] 102 109439
[iot_thread] Failed to initialize PKCS #11. PKCS #11 was already initialized.103 109439
[iot_thread]
104 109439 [iot_thread] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:1717] 105 109439
[iot_thread] Successfully Returned a PKCS #11 slot with ID 1 with a count of 1.106 109439
[iot_thread]
107 109439 [iot_thread] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:1953] 108 109439
[iot_thread] Assigned a 0x2 Type Session.109 109439 [iot_thread]
110 109439 [iot_thread] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:1964] 111 109439
[iot_thread] Assigned Mechanisms to no operation in progress.112 109439 [iot_thread]
113 109439 [iot_thread] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:1980] 114 109439
[iot_thread] Current session count at 0115 109439 [iot_thread]
167 111156 [iot_thread] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:1073] 168 111156
[iot_thread] Found object in list by handle.169 111156 [iot_thread]
170 111156 [iot_thread] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:3780] 171 111157
[iot_thread] Successfully started sign operation.172 111157 [iot_thread]
173 112138 [iot_thread] [DEBUG] [PKCS11] [core_pkcs11_mbedtls.c:3966] 174 112138
[iot_thread] Ended Sign operation.175 112138 [iot_thread]
176 112223 [iot_thread] [INFO ][MQTT][112223] Establishing new MQTT connection.
177 112223 [iot_thread] [INFO ][MQTT][112223] (MQTT connection 23f18, CONNECT operation
240b8) Waiting for operation completion.
178 112351 [NetRecv] [INFO] [MQTT] [core_mqtt_serializer.c:970] 179 112351 [NetRecv]
CONNACK session present bit not set.180 112351 [NetRecv]
181 112351 [NetRecv] [INFO] [MQTT] [core_mqtt_serializer.c:912] 182 112351 [NetRecv]
Connection accepted.183 112351 [NetRecv]
184 112351 [iot_thread] [INFO ][MQTT][112351] (MQTT connection 23f18, CONNECT operation
240b8) Wait complete with result SUCCESS.
185 112351 [iot_thread] [INFO ][MQTT][112351] New MQTT connection 6a54 established.
186 112353 [iot_thread] [OTA_AgentInit_internal] OTA Task is Ready.
187 112353 [OTA Agent T] [prvPAL_GetPlatformImageState] is called.
188 112353 [OTA Agent T] Function call: prvPAL_GetPlatformImageState: [2]
189 112353 [OTA Agent T] [prvExecuteHandler] Called handler. Current State [Ready] Event
[Start] New state [RequestingJob]
190 112358 [OTA Agent T] [INFO ][MQTT][112358] (MQTT connection 23f18) SUBSCRIBE
operation scheduled.
191 112358 [OTA Agent T] [INFO ][MQTT][112358] (MQTT connection 23f18, SUBSCRIBE
operation 2a960) Waiting for operation completion.
192 112473 [OTA Agent T] [INFO ][MQTT][112473] (MQTT connection 23f18, SUBSCRIBE
operation 2a960) Wait complete with result SUCCESS.
193 112473 [OTA Agent T] [prvSubscribeToJobNotificationTopics] OK:
$aws/things/rx671_POC/jobs/$next/get/accepted
194 112478 [OTA Agent T] [INFO ][MQTT][112478] (MQTT connection 23f18) SUBSCRIBE
operation scheduled.
195 112478 [OTA Agent T] [INFO ][MQTT][112478] (MQTT connection 23f18, SUBSCRIBE
operation 241f8) Waiting for operation completion.
196 112585 [OTA Agent T] [INFO ][MQTT][112585] (MQTT connection 23f18, SUBSCRIBE
operation 241f8) Wait complete with result SUCCESS.
197 112585 [OTA Agent T] [prvSubscribeToJobNotificationTopics] OK:
$aws/things/rx671_POC/jobs/notify-next
```

```
198 112585 [OTA Agent T] [prvRequestJob_Mqtt] Request #0
199 112594 [OTA Agent T] [INFO ][MQTT][112594] (MQTT connection 23f18) MQTT PUBLISH
operation queued.
200 112594 [OTA Agent T] [INFO ][MQTT][112594] (MQTT connection 23f18, PUBLISH operation
241f8) Waiting for operation completion.
201 112670 [OTA Agent T] [INFO ][MQTT][112670] (MQTT connection 23f18, PUBLISH operation
241f8) Wait complete with result SUCCESS.
202 112670 [OTA Agent T] [prvExecuteHandler] Called handler. Current State
[RequestingJob] Event [RequestJobDocument] New state [WaitingForJob]
203 112672 [OTA Agent T] [prvParseJSONbyModel] Extracted parameter [ clientToken:
0:rx671_POC ]
204 112672 [OTA Agent T] [prvParseJSONbyModel] Extracted parameter [ timestamp:
1662611090 ]
205 112672 [OTA Agent T] [prvParseJSONbyModel] parameter not present: execution
206 112672 [OTA Agent T] [prvParseJSONbyModel] parameter not present: jobId
207 112672 [OTA Agent T] [prvParseJSONbyModel] parameter not present: jobDocument
208 112672 [OTA Agent T] [prvParseJSONbyModel] parameter not present: afr_ota
209 112672 [OTA Agent T] [prvParseJSONbyModel] parameter not present: protocols
210 112672 [OTA Agent T] [prvParseJSONbyModel] parameter not present: files
211 112672 [OTA Agent T] [prvParseJSONbyModel] parameter not present: filepath
212 112672 [OTA Agent T] [prvParseJSONbyModel] parameter not present: filesize
213 112672 [OTA Agent T] [prvParseJSONbyModel] parameter not present: fileid
214 112672 [OTA Agent T] [prvParseJSONbyModel] parameter not present: certfile
215 112672 [OTA Agent T] [prvParseJSONbyModel] parameter not present: sig-sha256-ecdsa
216 112672 [OTA Agent T] [prvParseJobDoc] No active jobs available in the service for
execution.
217 112674 [OTA Agent T] [prvParseJobDoc] Ignoring job without ID.
222 113353 [iot_thread] [INFO ][DEMO][113353] State: Ready  Received: 1   Queued: 0
Processed: 0   Dropped: 0

223 114353 [iot_thread] [INFO ][DEMO][114353] State: WaitingForJob  Received: 1   Queued:
0   Processed: 0   Dropped: 0

224 115353 [iot_thread] [INFO ][DEMO][115353] State: WaitingForJob  Received: 1   Queued:
0   Processed: 0   Dropped: 0

225 116353 [iot_thread] [INFO ][DEMO][116353] State: WaitingForJob  Received: 1   Queued:
0   Processed: 0   Dropped: 0

226 117353 [iot_thread] [INFO ][DEMO][117353] State: WaitingForJob  Received: 1   Queued:
0   Processed: 0   Dropped: 0

227 118353 [iot_thread] [INFO ][DEMO][118353] State: WaitingForJob  Received: 1   Queued:
0   Processed: 0   Dropped: 0
```

8. 制限事項

本アプリケーションノートの制限事項を以下に記載します。

- Big エンディアンの FreeRTOS OTA プログラムは正常に動作しません。
  Little エンディアンでプログラムをビルドし動作させてください。

9. 参考資料

- RX671 グループ ユーザーズマニュアル ハードウェア編（R01UH0905）
- Renesas Starter Kit+ for RX671 ユーザーズマニュアル（R20UT4879）
- RX ファミリ QE と FIT を使用した静電容量タッチアプリケーションの開発（R01AN4516）
- RX ファミリ QE for Display シリアル接続 LCD を使用した GUI 画面表示アプリケーション開発ガイド（R20AN0688）
- ルネサス MCU ファームウェアアップデートの設計方針（R01AN5548）
- RX65N における Amazon Web Services を利用した FreeRTOS OTA の実現方法（R01AN5549）

最新版をルネサス　エレクトロニクスホームページから入手してください。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Apr.24.23	ー	初版

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。