User's Manual

R-IN32M3 シリーズ

プログラミングマニュアル　（ドライバ編）

- R-IN32M3-EC
- R-IN32M3-CL

資料番号：R18UZ0008JJ0601
発行年月：2019.4.19
ルネサス エレクトロニクス
www.renesas.com

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 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。お客様の機器の設計において、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には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、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に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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
   高品質水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）
   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等） を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等

8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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おりますが、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、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。お客様が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. 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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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その総株主の議決権の過半数を直接または間接に保有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。

製品ご使用上の注意事項

ここでは、CMOS デバイスの一般的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、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】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. リザーブアドレスのアクセス禁止
   【注意】リザーブアドレスのアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。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。
4. クロックについて
   【注意】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。

このマニュアルの使い方

1. 目的と対象者

このマニュアルはイーサネット通信 LSI「R-IN32M3 シリーズ」の機能を理解し、それを用いた応用設計をするユーザを対象とします。このマニュアルを使用するには、電気回路、論理回路マイクロコンピュータに関する基本的な知識が必要です。

本製品は、注意事項を十分確認の上、使用してください。注意事項は、各章の本文中、各章の最後、注意事項の章に記載しています。

改訂記録は旧版の記載内容に対して訂正または追加した主な箇所をまとめたものです。改訂内容すべてを記録したものではありません。詳細は、このマニュアルの本文でご確認ください。
本文中の★印は、本版で改訂された主な箇所を示しています。この"★"を PDF 上でコピーして「検索する文字列」に指定することによって、改版箇所を容易に検索できます

関連資料　　関連資料は暫定版の場合がありますが、この資料では「暫定」の表示をしておりません。あらかじめご了承ください。また各コアの開発・企画段階で資料を作成しているため、関連資料は個別のお客様向け資料の場合があります。

R-IN32M3に関する資料

資料名	資料番号
R-IN32M3 シリーズ　データシート	R18DS0007JJ****
R-IN32M3 シリーズ　ユーザーズ・マニュアル R-IN32M3-EC	R18UZ0002JJ****
R-IN32M3 シリーズ　ユーザーズ・マニュアル R-IN32M3-CL	R18UZ0004JJ****
R-IN32M3 シリーズ　ユーザーズ・マニュアル 周辺機能編	R18UZ0006JJ****
R-IN32M3 シリーズ　プログラミング・マニュアル（ドライバ編）	このマニュアル
R-IN32M3 シリーズ　プログラミング・マニュアル（OS 編）	R18UZ0010JJ****
R-IN32M3 シリーズ　ユーザーズ・マニュアル　TCP/IP スタック編	R18UZ0018JJ****

2. 数や記号の表記

データ表記の重み：左が上位桁、右が下位桁
アクティブ・ローの表記：
　　xxxZ（端子、信号名称のあとにZ ）
　　またはxxx_N （端子、信号名称のあとに_N ）
　　またはxxnx （端子、信号名称にnを含む）
注：
　　本文中につけた注の説明
注意：
　　気をつけて読んでいただきたい内容
備考：
　　本文の補足説明
数の表記：
　　2 進数 … xxxx , xxxxBまたはn’bxxxx（nビット）
　　10 進数 … xxxx
　　16 進数 … xxxxHまたはn’hxxxx（nビット）

2のべき数を示す接頭語（アドレス空間、メモリ容量）：
　　K （キロ）… $2^{10} = 1024$
　　M （メガ）… $2^{20} = 1024^2$
　　G （ギガ）… $2^{30} = 1024^3$
データ・タイプ：
　　ワード … 32 ビット
　　ハーフワード … 16 ビット
　　バイト … 8 ビット

目次

図の目次

表の目次

1. 概説

ソフトウェア開発を速やかに進められるよう、R-IN32M3 では各機能の使用例を示したサンプル・ソフトウェア（以降サンプル・ソフト）を準備しています。

本書では、R-IN32M3 各機能のドライバ、ミドルウエアの仕様、および各開発ツール依存部（スタートアップ・ルーチン等）の動作について記しています。

1.1 構成

サンプル・ソフトのレイヤ構成図を以下に示します。

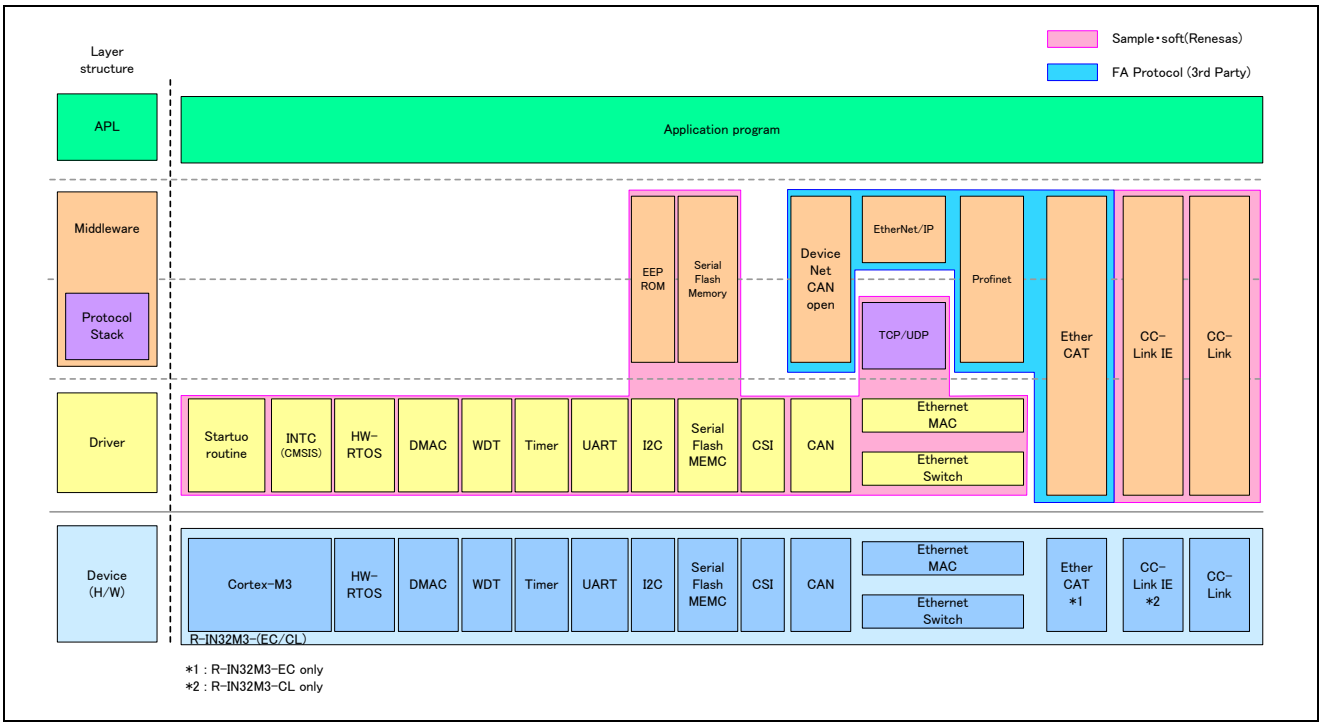

図 1.1 サンプル・ソフトのレイヤ構成図

1.2　開発環境

サンプルソフトは下記のツールチェーンで動作を確認しています。
また、サンプルソフトでは Arm® Cortex®マイクロコントローラ・ソフトウェア・インタフェース規格（CMSIS）の V2.10 を採用しています。詳細は CMSIS のドキュメントを参照してください。

表 1.1　ソフトウェア開発ツール一覧（ツールチェーン）

ツールチェーン	IDE	コンパイラ	デバッガ	ICE
IAR	Embedded Workbench for Arm V6.60 ～ 最新版 (最新版をお使いください) (IAR Systems)			I-jet JTAGjet-Trace-CM (IAR Systems)
Keil MDK-Arm	µVision V5.18.0.0 (Arm)			ULINK2 ULINKpro (Arm)
GNU	-	Sourcery G++ Lite for ARM EABI 2012.09-63 (Mentor Graphics)	microVIEW-PLUS Ver.5.11PL3 (DTSインサイト★)	adviceLUNA 2.03-00 (DTSインサイト★)

表 1.2　ソフトウェア開発ツール一覧（開発環境）

ツール種類	製品名	バージョン	ツールベンダ
開発環境	cygwin注	-	レッドハット
メイク・ツール	GNU make（cygwin注用）	3.81	GPL

注. cygwin のインストールについては、http://cygwin.com/ を参照してください。
また、GNU make も cygwin をインストールする際にあわせてインストールしてください。

2. ファイル構成

本章では、サンプル・ソフトのディレクトリ構成、ファイル構成を示します。

2.1 ディレクトリ構成

表 2.1　サンプル・ソフトのディレクトリ構成

ディレクトリ	内容
./	サンプル・ソフト格納ディレクトリ
./CMSIS	Arm Cortex-M3関連定義（CMSISをそのまま使用） 格納ディレクトリ
./Device	デバイス依存ファイル格納ディレクトリ
./ Device /Renesas/RIN32M3	R-IN32M3依存プログラム格納ディレクトリ
./ Device /Renesas/RIN32M3/Include	インクルード・ファイル格納ディレクトリ
./ Device /Renesas/RIN32M3/Library	ライブラリ格納ディレクトリ
./ Device /Renesas/RIN32M3/Source	ソース格納ディレクトリ

2.2 ./Device/Renesas/RIN32M3/Include：インクルード・ファイル

以下にインクルード・ファイルのファイル構成を示します。

表 2.2　インクルード・ファイル・ディレクトリのファイル構成

ディレクトリ	ファイル	内容
csi/	csi.h	CSIドライバのプロトタイプ宣言
iic/	iic.h	I2Cドライバのプロトタイプ宣言
sromc/	sromc.h	シリアル・フラッシュROMドライバのプロトタイプ宣言
timer/	timer.h	Timerドライバのプロトタイプ宣言
uart/	uart.h	UARTドライバのプロトタイプ宣言
hwos/	hwos_hwfnc.h	HW-RTOSドライバ・ヘッダ・ファイル
wdt/	wdt.h	ウォッチドッグ・タイマのプロトタイプ宣言
dmac/	dmac.h	DMAドライバ・ヘッダ・ファイル
./	errcodes.h	エラー定義ファイル
./	itron.h	ITRON一般（データ・タイプ、定数、マクロ）定義ファイル
./	kernel.h	主要機能（サービス・コール、データ・タイプ、定数、マクロ）定義ファイル
./	RIN32M3.h	R-IN32M3シリーズのデバイス定義インクルードファイル
./	RIN32M3_EC.h	R-IN32M3-EC用デバイス定義ファイル
./	RIN32M3_CL.h	R-IN32M3-CL用デバイス定義ファイル
./	system_RIN32M3.h	CMSISに準じたシステム情報定義

2.3 ./Device/Renesas/RIN32M3/Library：ライブラリ

以下にライブラリのファイル構成を示します。

表 2.3　ライブラリ・ディレクトリのファイル構成

ディレクトリ	ファイル	内容
ARM/	libos.a	H/W-RTOSドライバ・ライブラリ
GCC/	libos.a	H/W-RTOSドライバ・ライブラリ
IAR/	libos.a	H/W-RTOSドライバ・ライブラリ

2.4 ./Device/Renesas/RIN32M3/Source：ソース

以下にソース・ディレクトリの構成を示します。

表 2.4　ソース・ディレクトリのディレクトリ構成

ディレクトリ	内容
Driver	ドライバ関連
Middleware	ミドルウェア
Project	サンプル・アプリケーション
Templates	スタートアップ・ファイル等

2.4.1 ./Device/Renesas/RIN32M3/Source/Driver：ドライバ関連

以下にドライバのソース・ファイルの構成を示します。

表 2.5　ドライバ関連ディレクトリのファイル構成

ディレクトリ	ファイル	内容
can/	can_cfg.c	CAN コンフィギュレーションテーブル
can/	can_data.c	CAN データドライバ
can/	can_init.c	CAN 初期化ドライバ
can/	can_mode.c	CAN モードドライバ
can/	can_status.c	CAN ステータスドライバ
can/	can_xfer.c	CAN 転送ドライバ
csi/	csi.c	CSI ドライバ
dmac/	dmac.c	DMAC ドライバ
iic/	iic.c	IIC ドライバ
sromc/	sromc.c	シリアル・フラッシュ ROM ドライバ
timer/	timer.c	Timer ドライバ
uart/	uart.c	UART ドライバ
wdt/	wdt.c	ウォッチドッグ・タイマ・ドライバ

2.4.2 ./Device/Renesas/RIN32M3/Source/Middleware：ミドルウェア

以下にミドルウェアのソース・ファイルの構成を示します。

表 2.6　ミドルウェア・ディレクトリのファイル構成

ディレクトリ	ファイル	内容
eeprom/	eeprom.c	EEPROMミドルウェア　サンプル・ソース
	eeprom.h	EEPROMミドルウェア　ヘッダ・ファイル
flash/	flash.c	パラレル・フラッシュROMミドルウェア　サンプル・ソース
	flash.h	パラレル・フラッシュROMミドルウェア　ヘッダ・ファイル
sflash/	sflash.c	シリアル・フラッシュROMミドルウェア　サンプル・ソース
	sflash.h	シリアル・フラッシュROMミドルウェア　ヘッダ・ファイル

2.4.3 ./Device/Renesas/RIN32M3/Source/Project：サンプル・アプリケーション

以下にサンプル・アプリケーションの構成を示します。サンプル・アプリケーション動作概要は「7. アプリケーション例」を参照してください。

表 2.7　サンプル・アプリケーション・ディレクトリのファイル構成

ディレクトリ	ファイル	内容
osless_sample/	main.c	メイン処理ソース・ファイル
osless_sample/ARM/	main.uvoptx	デバッガ（MDK-Arm）関連ファイル
	main.uvprojx	デバッガ（MDK-Arm）プロジェクト・ファイル
	Makefile	メイクファイル
	scat_boot_extrom.ld	リンカ・スクリプト（ブート・コード：Flash ROM配置）
	scat_boot_iram.ld	リンカ・スクリプト（ブート・コード：命令RAM配置）
	scat_boot_sflash.ld	リンカ・スクリプト（ブート・コード：Serial Flash ROM配置）
osless_sample/GCC/	Makefile	メイクファイル
	scat_boot_extrom.ld	リンカ・スクリプト（ブート・コード：Flash ROM配置）
	scat_boot_iram.ld	リンカ・スクリプト（ブート・コード：命令RAM配置）
	scat_boot_sflash.ld	リンカ・スクリプト（ブート・コード：Serial Flash ROM配置）
	rin32m3ec.mvp	デバッガ（adviceLUNA）プロジェクト・ファイル
	rin32m3ec.mvr	デバッガ（adviceLUNA）関連ファイル
osless_sample/IAR/	main.eww	IARプロジェクト・ファイル
	main.ewd	IARプロジェクト関連ファイル
	main.ewp	IARプロジェクト関連ファイル
	boot_norflash.icf	リンカ構成ファイル（ブート・コード：Flash ROM配置）
	iram.icf	リンカ構成ファイル（ブート・コード：命令RAM配置）
	boot_serialflash.icf	リンカ構成ファイル（ブート・コード：Serial Flash ROM配置）
	init.mac	デバッガ用マクロファイル

2.4.4 ./Device/Renesas/RIN32M3/Source/Templates：スタートアップ・ファイル等

以下にスタートアップ・ファイル等のソース・ファイルの構成を示します。

表 2.8　スタートアップ関連ディレクトリのファイル構成

ディレクトリ	ファイル	内容
Templates/ARM/	startup_RIN32M3.c	スタートアップ・ファイル（MDK-Arm用）
	syscalls.c	ライブラリ関数の再定義ファイル（MDK-Arm用）
Templates/GCC/	startup_RIN32M3.c	スタートアップ・ファイル（GCC用）
	syscalls.c	ライブラリ関数の再定義ファイル（GCC用）
Templates/IAR/	cstartup_M.c	スタートアップ・ファイル（IAR用）
	vectors_M.c	ベクタ配置ファイル
	vectors_rom.c	ベクタ配置ファイル（ROMブート用）
	syscalls.c	ライブラリ関数の再定義ファイル（IAR用）
Templates/	system_RIN32M3.c	スタートアップ・ファイル（共通）

3. ソフトウェア開発手順

ここでは、ソフトウェア開発の一連の手順を説明します。

3.1 設計フロー

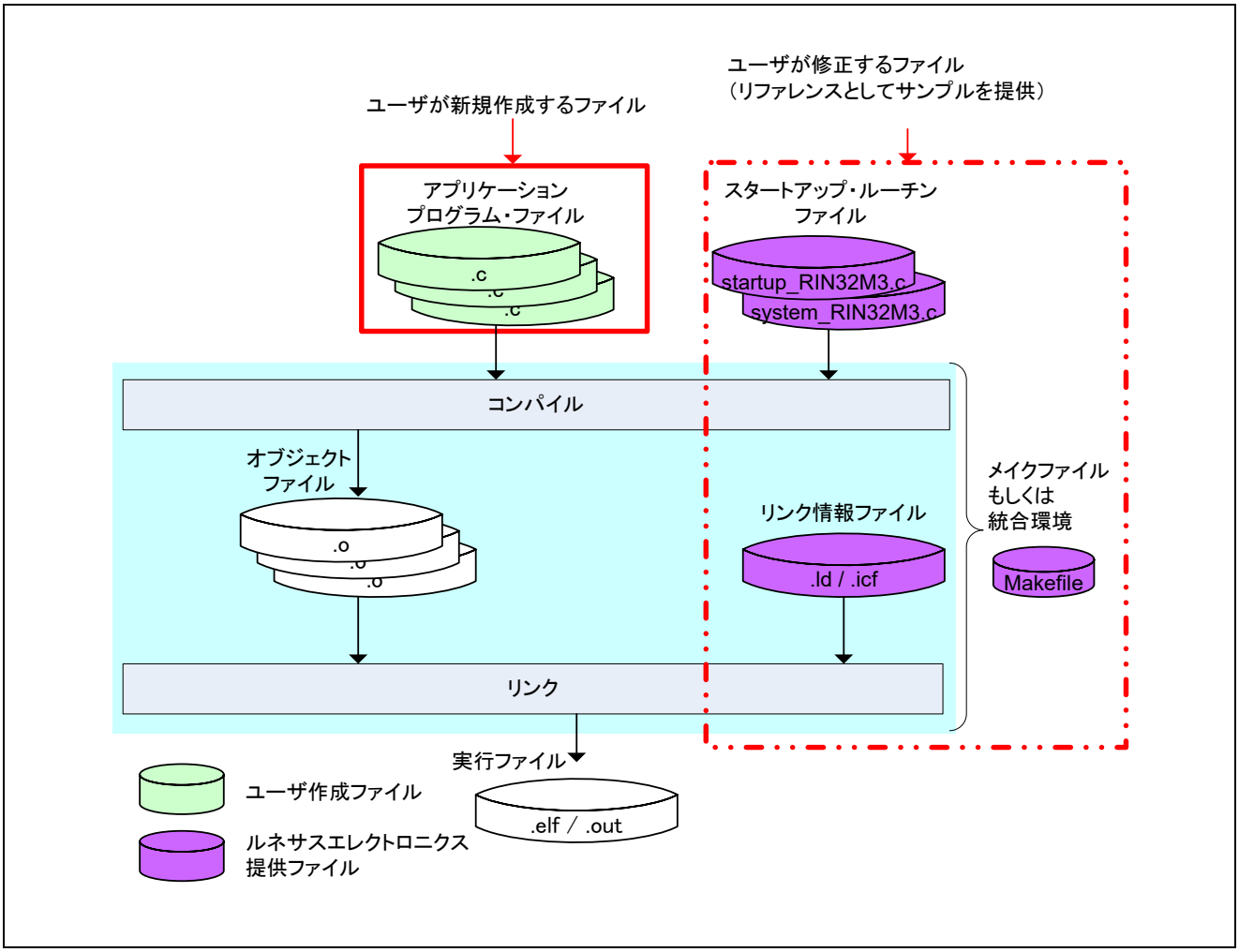

図 3.1　ファイル相関図

3.2　メモリ配置

3.2.1　メモリ配置★

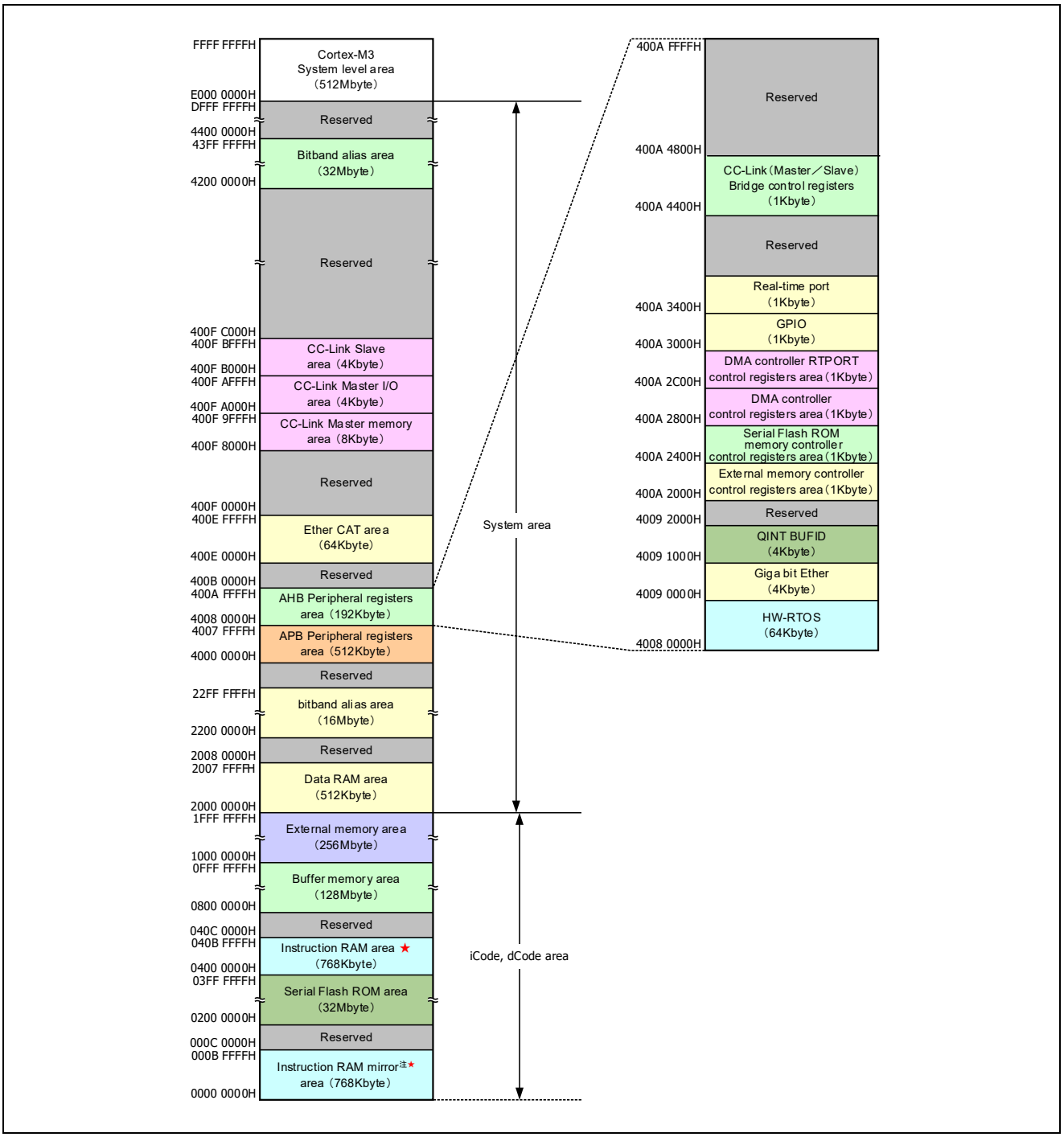

図 3.2　メモリ・マップ（全体）　（R-IN32M3-EC）

★注. 上記 Instruction RAM mirror area(768K バイト)はブート・モードにより実際にアクセスが発生するアドレスが変化します。詳細は「R-IN32M3 シリーズ ユーザーズ・マニュアル周辺機能編」の「5.3 ブート・モードによるメモリ MAP の違い」を参照してください。

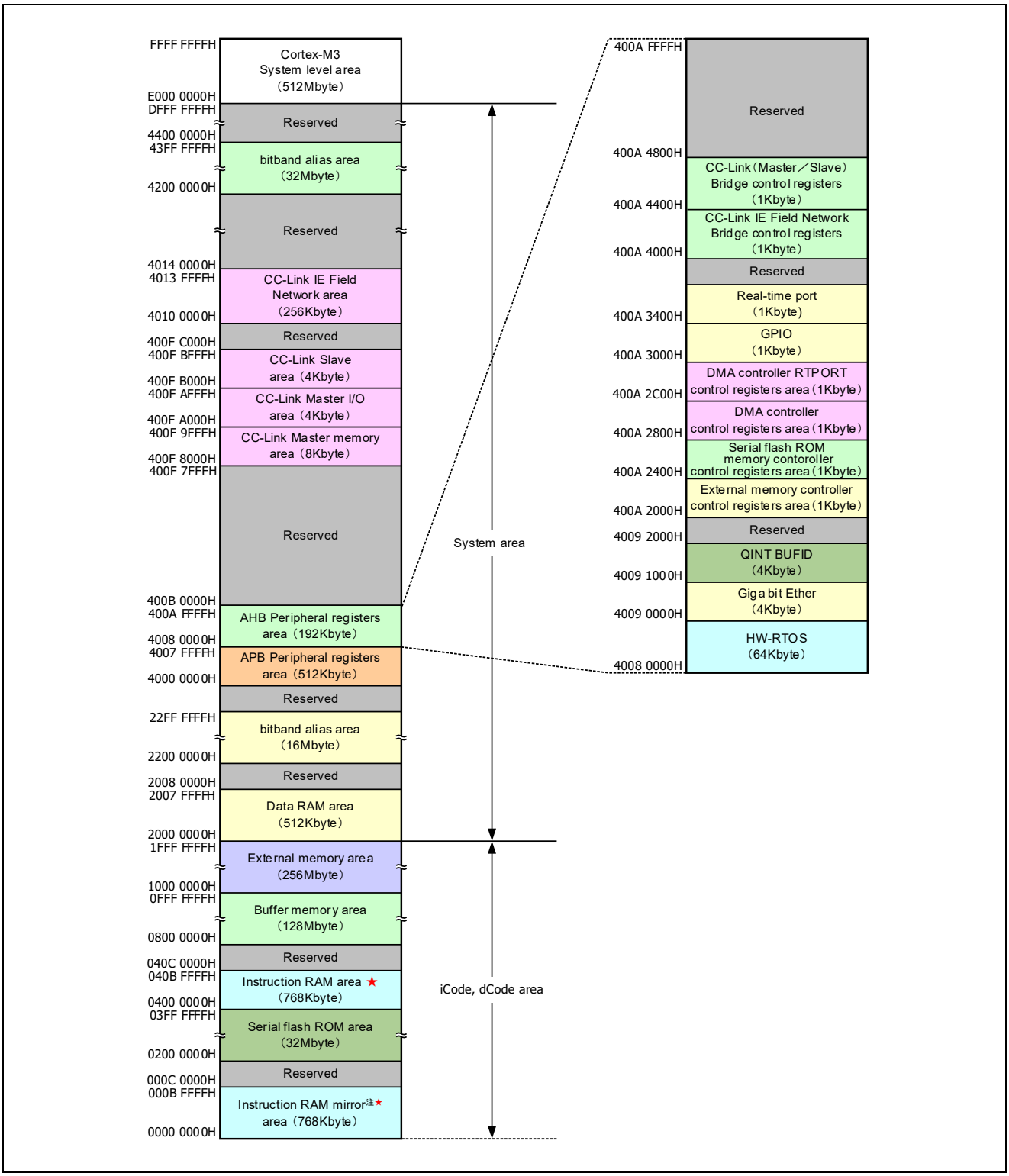

図 3.3　メモリ・マップ（全体）　（R-IN32M3-CL）

★注. 上記 Instruction RAM mirror area(768K バイト)はブート・モードにより実際にアクセスが発生するアドレスが変化します。詳細は「R-IN32M3 シリーズ ユーザーズ・マニュアル周辺機能編」の「5.3 ブート・モードによるメモリ MAP の違い」を参照してください。

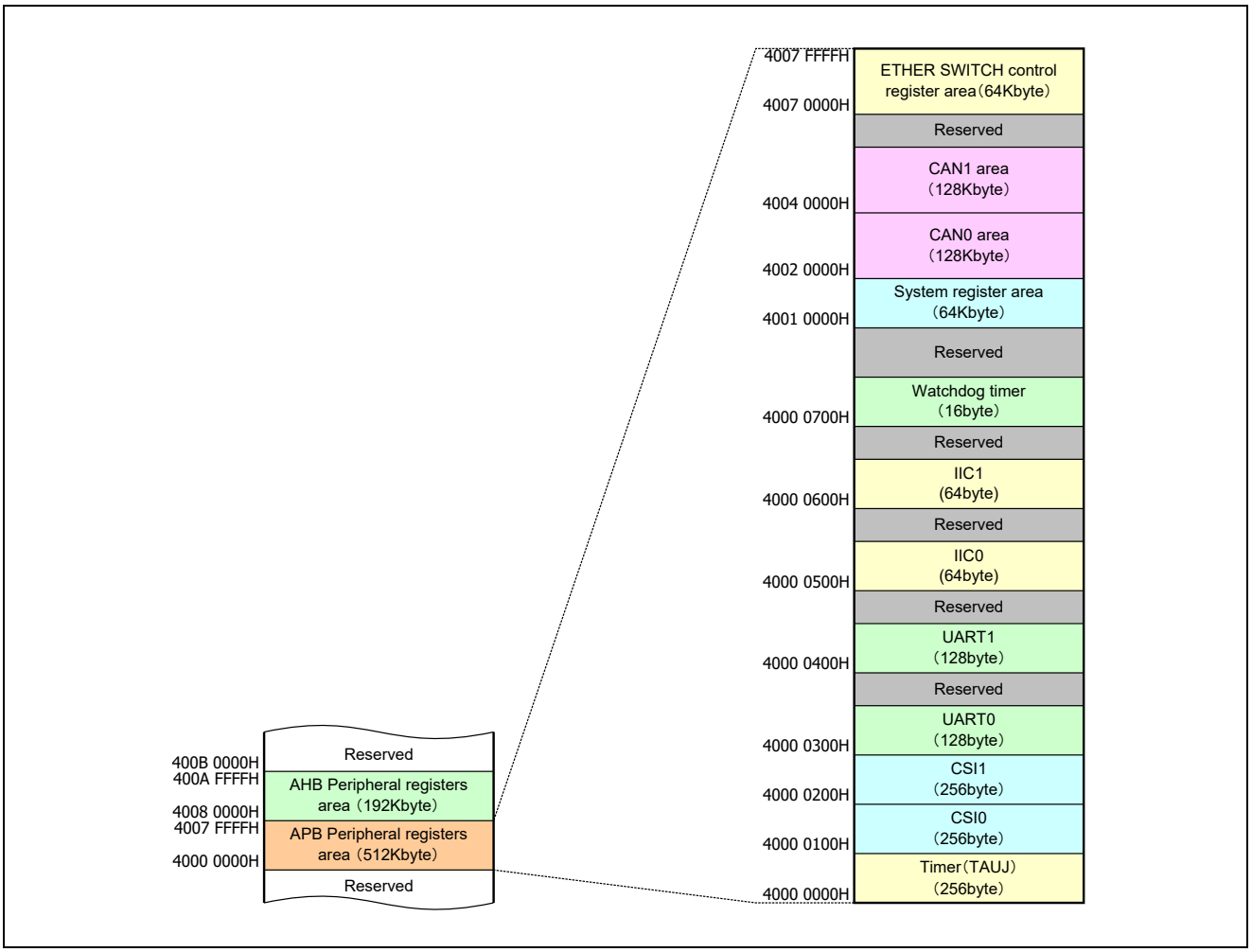

図 3.4　メモリ・マップ（APB 周辺レジスタ領域）（R-IN32M3-EC/CL 共通）

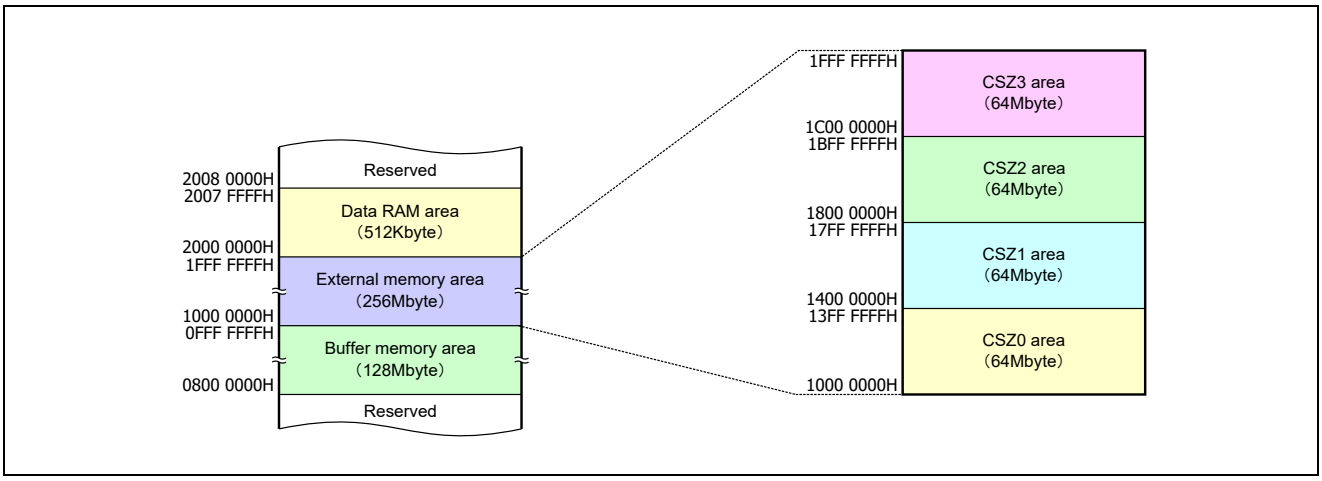

図 3.5　メモリ・マップ（外部メモリ領域）（R-IN32M3-EC/CL 共通）

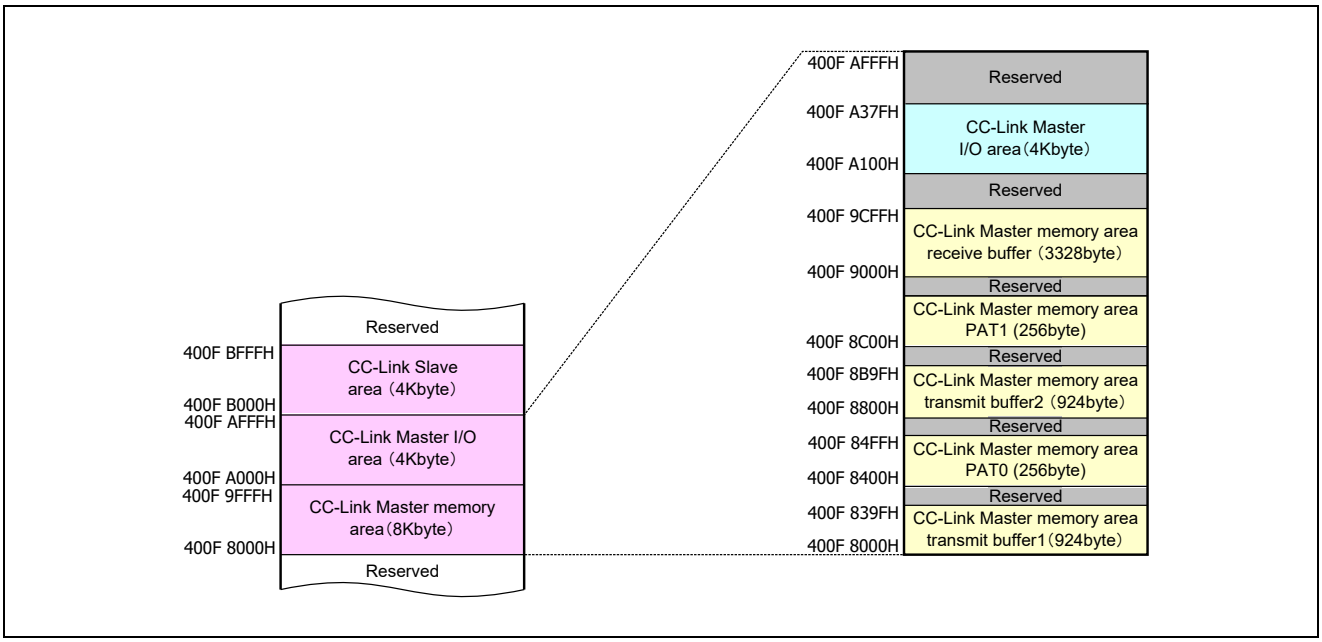

図 3.6　メモリ・マップ（CC-Link マスタ領域）（R-IN32M3-EC/CL 共通）

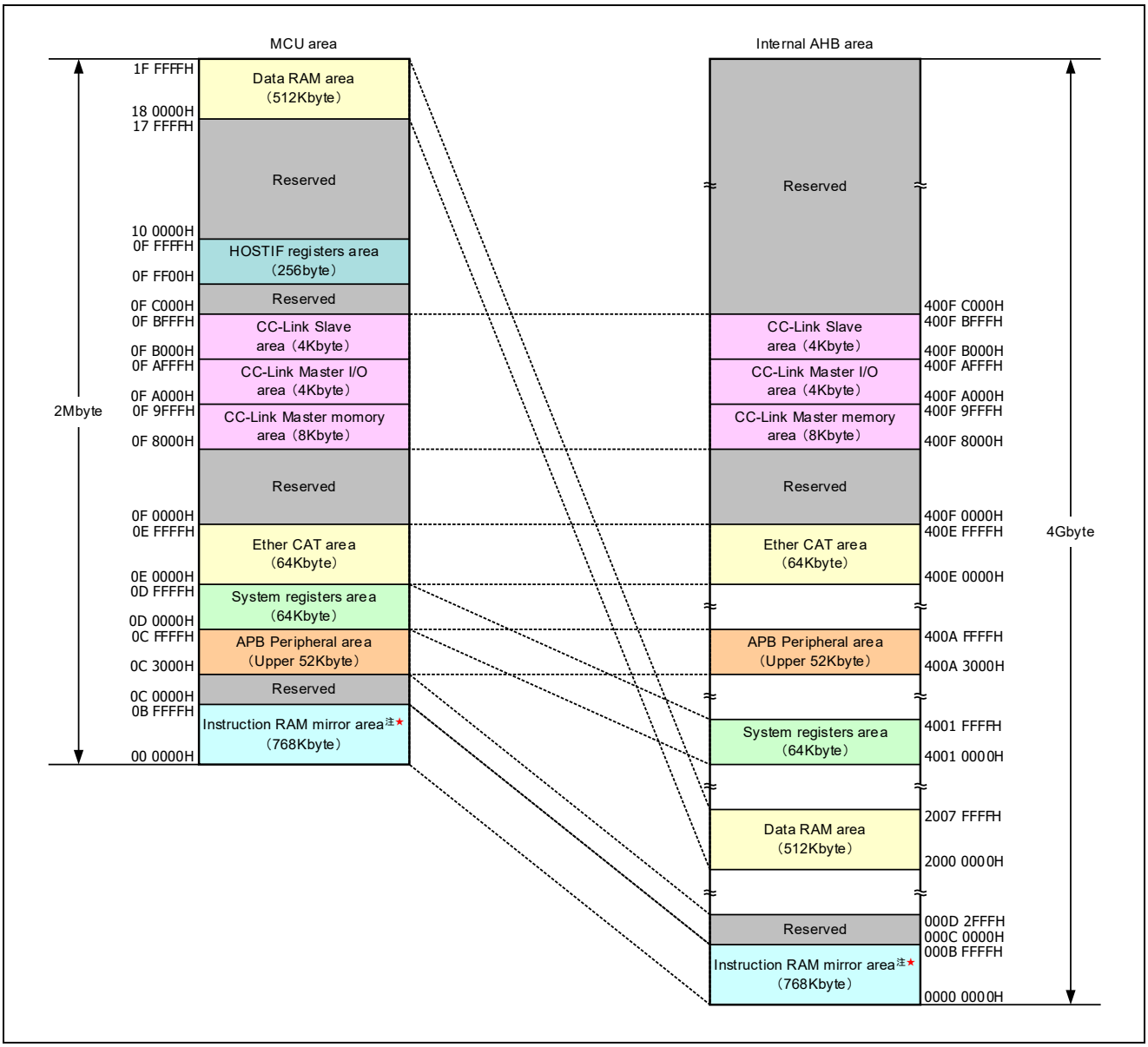

図 3.7　外部マイコン・インタフェース空間　（R-IN32M3-EC）

★注. 上記 Instruction RAM mirror area(768K バイト)はブート・モードにより実際にアクセスが発生する領域が以下のように変化します。詳細は「R-IN32M3 シリーズ ユーザーズ・マニュアル周辺機能編」の「5.3 ブート・モードによるメモリ MAP の違い」および「4. バス構成」を参照してください。

BOOT1	BOOT0	ブート・モード	アクセス先領域	備考
0	0	外部メモリ・ブート	―	外部マイコン・インタフェースの使用不可
0	1	外部シリアル・フラッシュ ROM ブート	予約領域	アクセス不可
1	0	外部マイコン・ブート	命令 RAM 領域	―
1	1	命令 RAM ブート	命令 RAM 領域	デバッグ時のみ使用可

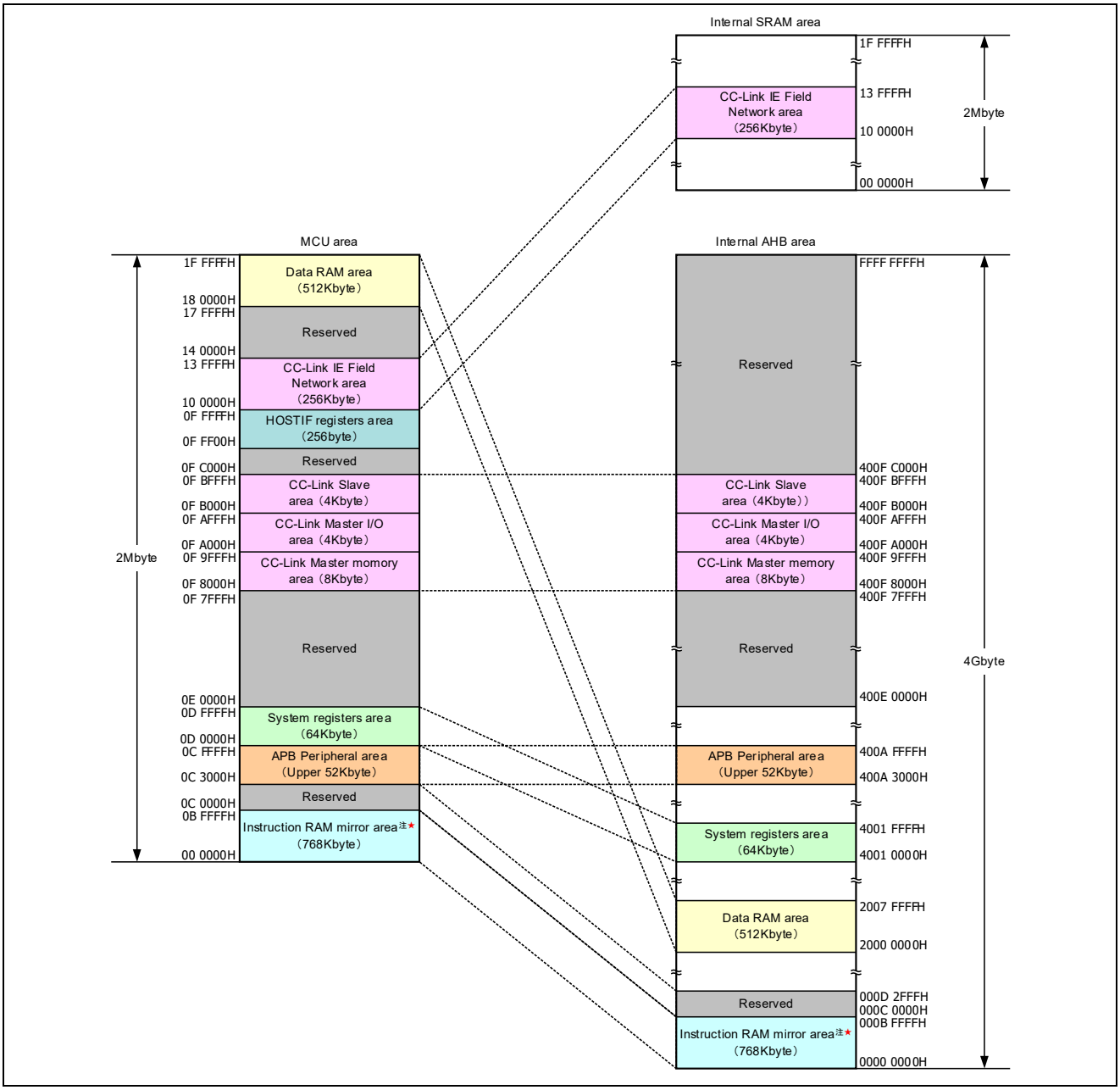

図 3.8　外部マイコン・インタフェース空間　（R-IN32M3-CL）

★注. 上記 Instruction RAM mirror area(768K バイト)はブート・モードにより実際にアクセスが発生する領域が以下のように変化します。詳細は「R-IN32M3 シリーズ ユーザーズ・マニュアル周辺機能編」の「5.3 ブート・モードによるメモリ MAP の違い」および「4. バス構成」を参照してください。

BOOT1	BOOT0	ブート・モード	アクセス先領域	備考
0	0	外部メモリ・ブート	ー	外部マイコン・インタフェースの使用不可
0	1	外部シリアル・フラッシュ ROM ブート	予約領域	アクセス不可
1	0	外部マイコン・ブート	命令 RAM 領域	ー
1	1	命令 RAM ブート	命令 RAM 領域	デバッグ時のみ使用可

3.2.2　プログラム配置例

パラレル・フラッシュブート時のプログラム配置例を以下に示します。

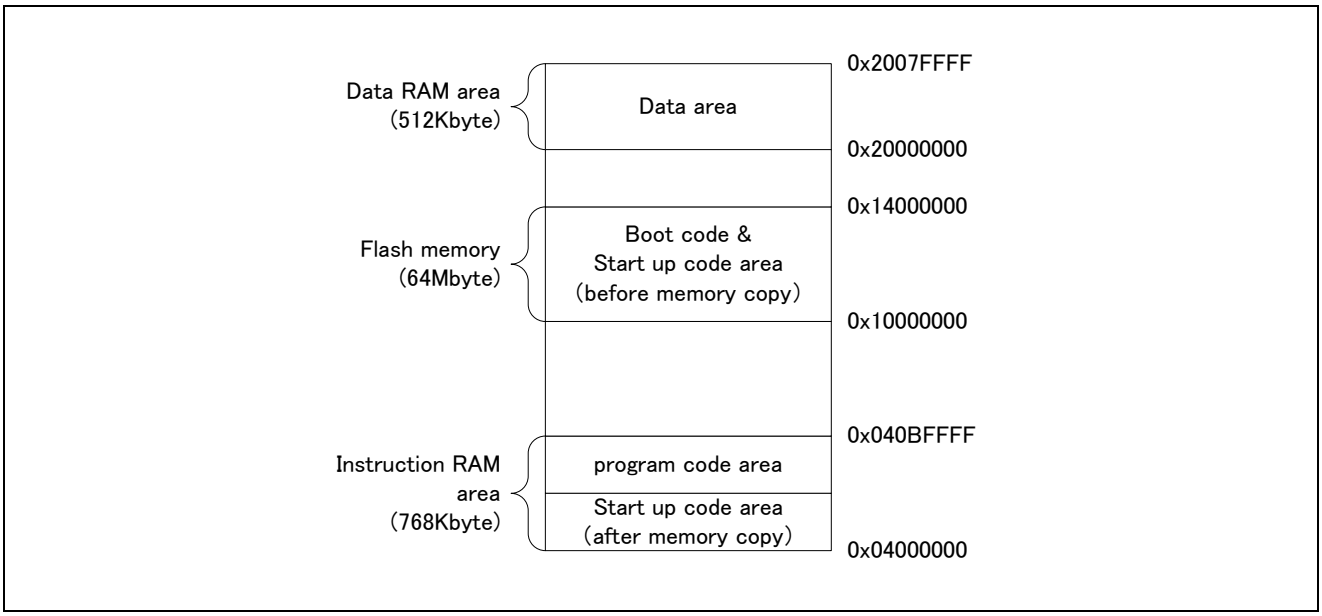

図 3.9　プログラム配置例

4. データ・タイプとマクロ

本章では、サンプル・ソフトで使用するデータ・タイプ、マクロについて解説します。

4.1 データ・タイプ

以下に、サンプル・ソフトで規定しているデータ・タイプ一覧を示します。

表 4.1　データ・タイプ

マクロ	型	意味
ER_RET	int	関数の戻り値で使用するエラー・コード
IRQn	enum	割り込み番号

4.2　マクロ定義

本サンプル・ソフトの各定義一覧を示します。

4.2.1　定数

以下に、本サンプル・ソフトにて規定している定数一覧です。

表 4.2　定数（一般）

定数	値	意味
NULL	((void*))0	無効ポインタ

表 4.3　定数（システム）

定数	値	意味
RIN32M3_SYSCLK	100000000	システムの周波数（単位: Hz）
SYS_UART_CH	1	システムで使用するUARTのチャネル番号

表 4.4　定数（エラー・コード）

定数	値	意味
ER_OK	0	正常終了
ER_NG	-1	異常終了
ER_SYS	-2	未定義エラー
ER_PARAM	-3	不正パラメータの検出
ER_NOTYET	-4	プロセスの未了
ER_NOMEM	-5	メモリ範囲外
ER_BUSY	-6	ビジー状態
ER_INVAL	-7	無効ステート
ER_TIMEOUT	-8	タイムアウト発生

4.2.2　条件付きコンパイル用定義

条件付きコンパイルに使用されるマクロ定義です。

表 4.5　条件付きコンパイルに使用されるマクロ定義

定義名	定義内容	使用されるファイル
OSLESS	H/W-RTOS 使用の有無を選択 定義有り：H/W-RTOS 未使用 定義無し：H/W-RTOS 使用	startup_RIN32M3.c cstartup_M.c
RIN32M3_CL	R-IN32M3-CL 使用時	RIN32M3.h system_RIN32M3.c flash.h

注意. R-IN32M3-CL をご使用の場合には、”RIN32M3_CL”を define してコンパイルしてください。
このマクロ定義を行わない場合、R-IN32M3-EC 向けのコードが生成されます。

5.　R-IN32M3 レジスタ定義（RIN32N3.h）

インクルード・ファイル RIN32M3.h では、R-IN32M3-EC または R-IN32M3-CL の割り込み定義とレジスタ定義を行っています。レジスタ定義を表 5.1、表 5.2 に示します。各レジスタの詳細については同表中のレジスタ詳細をご参照ください。

注意.　R-IN32M3-CL をご使用の場合には、”RIN32M3_CL”を define してコンパイルしてください。このマクロ定義を行わない場合、R-IN32M3-EC 向けの割り込み定義およびレジスタ定義が行われます。

5.1　APB 周辺レジスタ

表 5.1　APB 周辺レジスタ定義

#define	機能	レジスタ詳細
#define RIN_TMR_BASE	32bitタイマ・レジスタ（TAUJ2）	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」14章を参照
#define RIN_CSI0_BASE	CSIチャネル0レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」17章を参照
#define RIN_CSI1_BASE	CSIチャネル1レジスタ	
#define RIN_UART0_BASE	UARTチャネル0レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」16章を参照
#define RIN_UART1_BASE	UARTチャネル1レジスタ	
#define RIN_IIC0_BASE	I2Cチャネル0レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」18章を参照
#define RIN_IIC1_BASE	I2Cチャネル1レジスタ	
#define RIN_WDT_BASE	ウォッチドッグ・タイマ・レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」15章を参照
#define RIN_SYS_BASE	システム・レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」の各章を参照
#define RIN_CAN0_BASE	CANチャネル0レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」19章を参照
#define RIN_CAN1_BASE	CANチャネル1レジスタ	
#define RIN_ETHSW_BASE	イーサネット・スイッチ・レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」8章を参照

5.2　AHB 周辺レジスタ

表 5.2　AHB 周辺レジスタ定義

コード	機能	レジスタ詳細
#define RIN_HWOS_BASE	HW-RTOSレジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」7章を参照
#define RIN_ETH_BASE	ギガビット・イーサネットMACレジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」7章を参照
#define RIN_MEMC_BASE	非同期式SRAM MEMCレジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」9章を参照
#define RIN_SROM_BASE	シリアル・フラッシュROMメモリ・コントローラ・レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」12章を参照
#define RIN_DMAC0_BASE	DMACチャネル0レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」13章を参照
#define RIN_DMAC1_BASE	DMACチャネル1レジスタ	
#define RIN_DMAC2_BASE	DMACチャネル2レジスタ	
#define RIN_DMAC3_BASE	DMACチャネル3レジスタ	
#define RIN_RTDMAC_BASE	RTDMACレジスタ	
#define RIN_GPIO_BASE	ポート・レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル R-IN32M3-CL」7章または「R-IN32M3シリーズ ユーザーズ・マニュアル R-IN32M3-EC」8章を参照
#define RIN_RTPORT_BASE	RTポート・レジスタ	
#define RIN_CCI_BRG_BASE[注1]	CC-Link IE Fieldブリッジ制御レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル R-IN32M3-CL」6章を参照
#define RIN_CC_BR_BASE	CC-Linkブリッジ制御レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」20章を参照
#define RIN_SMC_BASE	同期式バースト・アクセスMEMCレジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル 周辺機能編」10章を参照
#define RIN_ECAT_BASE[注2]	EtherCAT®レジスタ	「R-IN32M3シリーズ ユーザーズ・マニュアル R-IN32M3-EC」6章を参照
#define RIN_CCLSLAVE_BASE	CC-Linkリモートデバイス局レジスタ	「R-IN32シリーズ ユーザーズ・マニュアル CC-Linkリモートデバイス局編」を参照

注 1. R-IN32M3-CL をご使用の場合のみ利用できます。
2. R-IN32M3-EC をご使用の場合のみ利用できます。

6. ドライバ

ドライバの関数説明を示します。

6.1 ドライバ関数一覧

サンプル・ソフトで用意している API の一覧を示します。

表 6.1 タイマドライバ関数一覧

関数名	関数概要
clock_init	システム・タイマ（チャネル 0）の初期化
timer_interval_init	インターバル・タイマ・モード初期化
timer_onecount_hwtrg_init	ワン・カウント・タイマ・モード初期化（ハードウェアトリガ）
timer_start	タイマ動作開始
timer_stop	タイマ動作停止
timer_check_act	タイマ動作状態確認

表 6.2 UART ドライバ関数一覧

関数名	関数概要
uart_init	UART 初期化
uart_write	1 バイト・キャラクタの送信
uart_read	1 バイト・キャラクタの受信
uart_check_receivedata	受信データの有無を確認

表 6.3 IIC ドライバ関数一覧

関数名	関数概要
iic_init	IIC コントローラ初期化
iic_start_condition	スタート・コンディション送信
iic_stop_condition	ストップ・コンディション送信
iic_write	1 バイト・キャラクタの送信
iic_read	1 バイト・キャラクタの受信

表 6.4　CSI ドライバ関数一覧

関数名	関数概要
csi_init	CSI コントローラ初期化
csi_write	1 バイト・キャラクタ送信
csi_read	1 バイト・キャラクタ受信
csi_check_tx	送信データ確認（スレーブ用）
csi_check_rx	受信データ確認（スレーブ用）
csi_change_mode	送受信モード切り替え（スレーブ用）

表 6.5　DMAC ドライバ関数一覧

関数名	関数概要
dmac_memcpy	メモリ・コピー（DMA 転送）

表 6.6　シリアル・フラッシュ ROM ドライバ関数一覧

関数名	関数概要
sromc_init	SPI バス制御初期化
sromc_write	SPI バスへデータ書き込み
sromc_read	SPI バスからデータ読み込み

表 6.7　ウォッチドッグ・タイマドライバ関数一覧

関数名	関数概要
wdt_init	ウォッチドッグ・タイマ初期化
wdt_start	ウォッチドッグ・タイマ起動
wdt_clear	カウント・クリア
wdt_wait_reset	リセット待ち

表 6.8　CAN ドライバ関数一覧

関数名	関数概要
can_enable	CAN コントローラ有効化
can_init	CAN コントローラ初期化
can_shutdown	CAN コントローラ強制終了
can_get_mode	CAN 動作モード取得
can_set_mode	CAN 動作モード設定
can_get_id_data_dlc	CAN 受信データ取得(CANID,Data,DLC)
can_get_data_dlc	CAN 受信データ取得(Data,DLC)
can_set_id_data_dlc	CAN 送信データ設定(CANID,Data,DLC)
can_set_data	CAN 送信データ設定
can_tx_req	CAN データ送信リクエスト
can_get_txinfo	CAN データ送信情報取得
can_get_rxinfo	CAN データ受信情報取得
can_get_ch_status	CAN チャネルステータス取得
can_clr_ch_status	CAN チャネルステータスクリア
can_get_bus_staus	CAN バスステータス取得

6.2 タイマ制御

6.2.1 タイマ・モジュールの初期化

clock_init

(1) 概要

タイマ・モジュールの初期化

(2) C 言語形式

```
void clock_init( void );
```

(3) パラメータ

なし

(4) 機能

clock 関数を使うための、システムタイマの初期設定を行います。

(5) 戻り値

なし

備考. 「R-IN32M3 シリーズ ユーザーズ・マニュアル 周辺機能編」の「TAUJ2 動作機能一覧」に記載されている「TAUJ2TTINm 入力位置検出機能」を使用しています。

6.2.2　インターバル・タイマ・モード初期化

timer_interval_init

(1) 概要

インターバル・タイマ・モード初期化

(2) C 言語形式

```
ER_RET timer_interval_init( uint8_t ch, uint_32t i_time );
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t　ch	チャネル選択引数 0：チャネル0 1：チャネル1 2：チャネル2 3：チャネル3
I	uint32_t　i_time	インターバル時間（1～42,949ms）

(4) 機能

チャネル選択引数で選択したチャネルをインターバル・タイマ・モードに設定します。

また、インターバル時間引数で与えたサイクルで割り込みを出力しますが、本サンプルでの割り込み優先度は最低順位です。

チャネル選択引数が 0～3 以外またはインターバル時間が 1～42,949ms 以外の場合パラメータ・エラーを返します。

- Timerクロック設定：以下に記載
  - > 基準クロック（PCLK）周波数100MHz
  - > カウントクロック周波数100MHz

(5) 戻り値

戻り値	意味
ER_OK	初期化成功
ER_PARAM	パラメータ・エラー ・　指定チャネルが0～3以外の場合 ・　インターバル時間が1～42,949ms以外の場合

備考. 「R-IN32M3 シリーズ ユーザーズ・マニュアル 周辺機能編」の「TAUJ2 動作機能一覧」に記載されている「インターバル・タイマ機能」を使用しています。

6.2.3　ワン・カウント・タイマ・モード初期化（ハードウェアトリガ）

timer_onecount_hwtrg_init

(1) 概要

ワン・カウント・タイマ・モード初期化（ハードウェアトリガ）

(2) C 言語形式

```
ER_RET timer_onecount_hwtrg_init( uint8_t ch,uint32_t o_time, uint32_t trg );
```

(3) パラメータ

I/O	パラメータ		説明
I	uint8_t	ch	チャネル選択引数 0：チャネル0 1：チャネル1 2：チャネル2 3：チャネル3
I	uint32_t	o_time	ワン・カウント出力時間（10～4,294,967,285ns）
I	uint32_t	trg	トリガ要因選択引数（要因のIRQ番号 + 4）

(4) 機能

チャネル選択引数で選択したチャネルをワン・カウント・タイマ・モードに設定し、トリガ要因選択引数で選択した割り込み信号をトリガとして、カウントを開始します。

ワン・カウント時間引数で与えた時間が経過すると、カウント動作を停止します。

カウント中のトリガ検出は行いません。

チャネル選択引数またはワン・カウント時間が設定可能な値以外の場合、パラメータ・エラーを返します。

本動作時のタイマのカウントクロック周期は、100MHz に設定してください。

注意. タイマのカウントクロック周期が、割り込みのパルス幅より長い場合、割り込みを検出できません。

(5) 戻り値

戻り値	意味
ER_OK	初期化成功
ER_PARAM	パラメータ・エラー ・　指定チャネルが0～3以外の場合 ・　ワン・カウント時間が1～42,949,672ns以外の場合

備考. 「R-IN32M3 シリーズ ユーザーズ・マニュアル 周辺機能編」の「TAUJ2 動作機能一覧」に記載されている「ディレイ・カウント機能」を使用しています。

6.2.4 タイマ動作開始

timer_start

(1) 概要

タイマ動作開始

(2) C 言語形式

```
ER_RET timer_start( uint8_t ch );
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1 2：チャネル2 3：チャネル3

(4) 機能

チャネル選択引数で選択したチャネルのタイマの動作を開始します。
チャネル選択引数が 0～3 以外の場合パラメータ・エラーを返します。

(5) 戻り値

戻り値	意味
ER_OK	タイマ動作開始成功
ER_PARAM	パラメータ・エラー ・指定チャネルが0～3以外の場合

6.2.5　タイマ動作停止

timer_stop

(1)　概要

タイマ動作停止

(2)　C 言語形式

```
ER_RET timer_stop( uint8_t ch );
```

(3)　パラメータ

I/O	パラメータ	説明
I	uint8_t　ch	チャネル選択引数 0：チャネル0 1：チャネル1 2：チャネル2 3：チャネル3

(4)　機能

チャネル選択引数で選択したチャネルのタイマの動作を停止します。

(5)　戻り値

戻り値	意味
ER_OK	タイマ動作停止成功
ER_PARAM	パラメータ・エラー ・指定チャネルが0～3以外の場合

6.2.6 タイマ動作状態確認

timer_check_act

(1) 概要

タイマ動作状態確認

(2) C 言語形式

```
ER_RET timer_check_act( uint8_t ch );
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1 2：チャネル2 3：チャネル3

(4) 機能

チャネル選択引数で選択したチャネルのタイマが動作中か停止しているのかを確認します。

(5) 戻り値

戻り値	意味
1	選択チャネルのタイマ動作中
0	選択チャネルのタイマ停止中
ER_PARAM	パラメータ・エラー ・指定チャネルが0～3以外の場合

6.3 UART 制御

6.3.1 UART モジュールの初期化

uart_init

(1) 概要

UART モジュールの初期化

(2) C 言語形式

```
ER_RET uart_init(uint8_t ch);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能

チャネル選択引数で指定したチャネルのボー・レートや、ビット・サイズなどの、各種設定を行います。
チャネル選択引数が 0 または 1 以外の場合、ER_PARAM（パラメータ・エラー）を返します。
チャネルの選択は system_RIN32M3.h にて定義します。

(5) 戻り値

戻り値	意味
ER_OK	初期化成功
ER_PARAM	指定チャネルが0または1以外の場合、パラメータ・エラー

6.3.2 UART による 1 バイト・キャラクタ・データの送信

uart_write

(1) 概要

UART による 1 バイト・キャラクタ・データの送信

(2) C 言語形式

```
ER_RET uart_write(uint8_t ch,uint8_t data);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t data	1バイト・キャラクタの送信データ

(4) 機能

選択したチャネルへ 1 バイト・キャラクタのデータを送信します。ただし、送信 FIFO が FULL の場合は、空きができるまで待機します。

チャネル選択引数が 0 または 1 以外の場合、ER_PARAM（パラメータ・エラー）を返します。

チャネルの選択は system_RIN32M3.h にて定義します。

(5) 戻り値

戻り値	意味
ER_OK	送信成功
ER_PARAM	パラメータ・エラー ・指定チャネルが0または1以外の場合

6.3.3 UART による 1 バイト・キャラクタ・データの受信

uart_read

(1) 概要

UART による 1 バイト・キャラクタ・データの受信

(2) C 言語形式

```
ER_RET uart_read(uint8_t ch,uint8_t *data);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
O	uint8_t * data	1バイト・キャラクタの受信データポインタ

(4) 機能

選択したチャネルから 1 バイト・キャラクタのデータを受信します。受信データがある場合、受信データを引数 data のポインタとして渡し、戻り値に 1 を返します。受信データがない場合、戻り値に 0 を返します。また、チャネル選択引数が 0 または 1 以外の場合、ER_PARAM（パラメータ・エラー）を返します。

チャネルの選択は system_RIN32M3.h にて定義します。

(5) 戻り値

戻り値	意味
1	受信データあり
0	受信データなし
ER_PARAM	パラメータ・エラー ・指定チャネルが0または1以外の場合

6.3.4 受信データの有無を確認

uart_check_receivedata

(1) 概要

受信データの有無を確認

(2) C 言語形式

```
ER_RET uart_check_receivedata(uint8_t ch);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能

指定したチャネル部の受信用 FIFO が空であるかどうかを確認します。
チャネル選択引数が 0 または 1 以外の場合、ER_PARAM（パラメータ・エラー）を返します。
チャネルの選択は system_RIN32M3.h にて定義します。

(5) 戻り値

戻り値	意味
1	受信データあり
0	受信データなし
ER_PARAM	パラメータ・エラー ・指定チャネルが0または1以外の場合

6.4 IIC 制御

6.4.1 IIC コントローラの初期化

iic_init

(1) 概要

IIC コントローラの初期化

(2) C 言語形式

```
ER_RET iic_init(uint8_t ch);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能★

選択したチャネルの IIC の初期設定を行います。
選択したチャネルが 0 または 1 以外の場合は ER_PARAM（パラメータ・エラー）を返します。

- IICクロック設定
  - ＞ 高速モード　：400kHz
- IICタイミング設定
  - ＞ ストップとスタートの間隔　：130×PCLK周期(ns)
  - ＞ SCLロー・レベル期間　：130×PCLK周期(ns)
  - ＞ SCLハイ・レベル期間　：116×PCLK周期(ns)
  - ＞ セットアップ・サイクル
    - スタート・コンディション　：116×PCLK周期(ns)
    - ストップ・コンディション　：116×PCLK周期(ns)
  - ＞ ホールド・サイクル
    - スタート・コンディション　：116×PCLK周期(ns)
    - データ　：32×PCLK周期(ns)

備考 1. IIC クロック設定は SDAn および SCLn の立ち上がり時間、立ち下がり時間が共に 20ns の場合を想定し、400kHz となるよう設定しています。ご利用の環境にあわせて適宜レジスタ設定を変更してください。詳細は「R-IN32M3 シリーズ ユーザーズ・マニュアル 周辺機能編」をご覧ください。
備考 2. PCLK 周期 = 10ns

(5) 戻り値

戻り値	意味
ER_OK	初期化成功
ER_PARAM	パラメータ・エラー ・指定チャネルが0または1以外の場合

6.4.2 スタート・コンディション送信

iic_start_condition

(1) 概要

スタート・コンディション送信

(2) C 言語形式

```
ER_RET iic_start_condition(uint8_t ch);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能

選択したチャネルへスタート・コンディション送信します。
選択したチャネルが 0 または 1 以外の場合は ER_PARAM（パラメータ・エラー）を返します。

(5) 戻り値

戻り値	意味
ER_OK	送信成功
ER_PARAM	パラメータ・エラー ・指定チャネルが0または1以外の場合

6.4.3 ストップ・コンディション送信

iic_stop_condition

(1) 概要

ストップ・コンディション送信

(2) C 言語形式

```
ER_RET iic_stop_condition(uint8_t ch);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能

選択したチャネルへストップ・コンディション送信します。
選択したチャネルが 0 または 1 以外の場合は ER_PARAM（パラメータ・エラー）を返します。

(5) 戻り値

戻り値	意味
ER_OK	送信成功
ER_PARAM	パラメータ・エラー ・指定チャネルが0または1以外の場合

6.4.4　1 バイト・キャラクタの送信

iic_write

(1) 概要

1 バイト・キャラクタの送信

(2) C 言語形式

```
ER_RET iic_write(uint8_t ch, uint8_t data);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t　ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t　data	1バイト・キャラクタの送信データ

(4) 機能

選択したチャネルへ 1 バイト・キャラクタのデータを送信します。
選択したチャネルが 0 または 1 以外の場合は ER_PARAM（パラメータ・エラー）を返します。
8 ビット・データの送信毎にデバイスから ACK が返らない場合は ER_NG（送信失敗）を返します。

(5) 戻り値

戻り値	意味
ER_OK	送信成功
ER_NG	送信失敗 ・デバイスからACKが返らない場合
ER_PARAM	パラメータ・エラー ・指定チャネルが0または1以外の場合

6.4.5 1 バイト・キャラクタの受信

iic_read

(1) 概要

1 バイト・キャラクタの受信

(2) C 言語形式

```
ER_RET iic_read(uint8_t ch, uint8_t *data, uint32_t last);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
O	uint8_t * data	1バイト・キャラクタの受信データポインタ
I	uint32_t last	最終データ指定引数 0：最終データ以外 other：最終データ

(4) 機能

選択したチャネルから 1 バイト・キャラクタのデータを受信します。
選択したチャネルが 0 または 1 以外の場合は ER_PARAM（パラメータ・エラー）を返します。
最終データ指定引数が 0 の場合は、ACK を出力し、0 以外の場合は ACK を出力しません。

(5) 戻り値

戻り値	意味
ER_OK	受信成功
ER_PARAM	パラメータ・エラー ・指定チャネルが0または1以外の場合

6.5 CSI 制御

6.5.1 CSI コントローラの初期化

csi_init

(1) 概要

CSI コントローラの初期化

(2) C 言語形式

```
ER_RET csi_init(uint32_t ch, uint32_t mode);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint32_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint32_t mode	マスタ、スレーブ・モード選択引数 0：マスタ 1：スレーブ

(4) 機能

チャネル選択引数で選択したチャネルの CSI の初期設定を行います。

チャネル引数またはマスタ、スレーブ・モード選択引数が、0 または 1 以外の場合は、ER_PARAM（パラメータ・エラー）を返します。

マスタ、スレーブ・モード共通の初期設定を以下に示します。

- CSIデータ設定 ：データ長8、MSBファースト、エラー検出無し
- CSIタイミング設定
  - セットアップ・サイクル ：0.5シリアル・クロック
  - ホールド・サイクル ：0.5シリアル・クロック

選択したモードごとの初期設定を以下に示します。

(a) マスタ・モード選択時

シリアル・クロック周波数： 16.667MHz
デフォルト・レベル： High

(b) スレーブ・モード選択時

CSI クロック設定： マスタからの入力クロック使用

備考 1. チップ・セレクト端子機能は使用していません。
2. FIFO 機能は使用していません。

(5) 戻り値

戻り値	意味
ER_OK	初期化成功
ER_PARAM	パラメータ・エラー ・指定チャネルまたは指定モードが0または1以外の場合

6.5.2 1 バイト・キャラクタ送信

csi_write

(1) 概要

1 バイト・キャラクタ送信

(2) C 言語形式

```
ER_RET csi_write(uint32_t ch, uint8_t data);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint32_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t data	1バイト・キャラクタの送信データ

(4) 機能

選択したチャネルが送信モードのときに 1 バイト・キャラクタのデータ送信を行います。マスタモード時に CSI コントローラの状態が送信モードでない場合は送信モードへ変更します。

チャネル引数またはマスタ、スレーブ・モード選択引数が、0 または 1 以外の場合は、ER_PARAM（パラメータ・エラー）を返します。また、スレーブ・モード時に CSI コントローラの状態が送信モードでない場合は、ER_INVAL（モード・エラー）を返します。

(5) 戻り値

戻り値	意味
ER_OK	送信成功
ER_PARAM	パラメータ・エラー ・指定チャネルが0, 1以外の場合
ER_INVAL	モード・エラー ・スレーブ・モード時CSIコントローラの状態が送信モードではない場合

6.5.3　1 バイト・キャラクタ受信

csi_read

(1) 概要

1 バイト・キャラクタ受信

(2) C 言語形式

```
ER_RET csi_read(uint32_t ch, uint8_t* data);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint32_t　ch	チャネル選択引数 0：チャネル0 1：チャネル1
O	uint8_t *　data	1バイト・キャラクタの受信データポインタ

(4) 機能

選択したチャネルが受信モードのときに 1 バイト・キャラクタのデータ受信を行います。マスタモード時に CSI コントローラの状態が受信モードでない場合は受信モードへ変更します。

チャネル引数またはマスタ、スレーブ・モード選択引数が、0 または 1 以外の場合は、パラメータ・エラーを返し、CSI コントローラの状態が受信モードでない場合は ER_INVAL（モード・エラー）を返します。また、スレーブ・モード時に CSI コントローラの状態が受信モードでない場合は、ER_INVAL（モード・エラー）を返します。

(5) 戻り値

戻り値	意味
ER_OK	受信成功
ER_PARAM	パラメータ・エラー ・指定チャネルが0, 1以外の場合
ER_INVAL	モード・エラー ・CSIコントローラの状態が受信モードではない場合

6.5.4　送信データ確認（スレーブ用）

csi_check_tx

(1) 概要

送信データ確認（スレーブ用）

(2) C 言語形式

```
ER_RET csi_check_tx(uint32_t ch);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint32_t　ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能

選択したチャネルが送信モードのときに CSI の送信データの有無を戻り値に返します。CSI コントローラがマスタモードの場合は送信データを蓄えないため、常に ER_OK（送信データなし）を返します。

チャネル引数が、0 または 1 以外の場合は ER_PARAM（パラメータ・エラー）を返し、CSI コントローラの状態が送信モードでない場合は ER_INVAL（モード・エラー）を返します。

(5) 戻り値

戻り値	意味
ER_OK	送信データなし
ER_NOTYET	送信データあり
ER_PARAM	パラメータ・エラー ・指定チャネルが0, 1以外の場合
ER_INVAL	モード・エラー ・CSIコントローラの状態が送信モードではない場合

6.5.5　受信データ確認（スレーブ用）

csi_check_rx

(1) 概要

受信データ確認（スレーブ用）

(2) C 言語形式

```
ER_RET csi_check_rx(uint32_t ch);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint32_t　ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能

選択したチャネルが受信モードのときに CSI の受信データの有無を戻り値に返します。CSI コントローラがマスタモードの場合は、受信データを蓄えないため、常に ER_NOTYET（受信データなし）を返します。

また、チャネル選択引数が、0 または 1 以外の場合は、ER_PARAM（パラメータ・エラー）を返し、CSI コントローラの状態が受信★モードでない場合は ER_INVAL（モード・エラー）を返します。

(5) 戻り値

戻り値	意味
ER_OK	受信データあり
ER_NOTYET	受信データなし
ER_PARAM	パラメータ・エラー ・指定チャネルが0, 1以外の場合
ER_INVAL	モード・エラー ・CSIコントローラの状態が受信モードではない場合

6.5.6 送受信モード切り替え（スレーブ用）

csi_change_mode

(1) 概要

送受信モード切り替え

(2) C 言語形式

```
ER_RET csi_change_mode(uint32_t ch, uint32_t mode);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint32_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint32_t mode	転送モード選択引数 0：受信モード 1：送信モード

(4) 機能

選択したチャネルの CSI の送受信モードを設定します。チャネル引数または転送モード選択引数が、0 または 1 以外の場合は、ER_PARAM（パラメータ・エラー）を返します。

- 転送モード選択引数が受信モードの場合は以下に設定切り替え
  - > 送信動作停止設定
  - > 受信動作許可設定
- 転送モード選択引数が送信モードの場合は以下に設定切り替え
  - > 送信動作許可
  - > 受信動作禁止

(5) 戻り値

戻り値	意味
ER_OK	モード切り替え成功
ER_PARAM	パラメータ・エラー ・指定チャネルが0または1以外の場合 ・指定モードが0または1以外の場合

6.6 DMA コントローラ制御

6.6.1 メモリ・コピー（DMA 転送）

dmac_memcpy

(1) 概要

メモリ・コピー（DMA 転送）

(2) C 言語形式

```
void *dmac_memcpy(void *dst, const void *src, uint32_t n);
```

(3) パラメータ

I/O	パラメータ		説明
I	void*	dst	転送先アドレス
I	void*	src	ソースアドレス
I	uint32_t	n	転送バイト数

(4) 機能

指定したソースアドレスから指定した転送先のアドレスへ DMA 転送によりメモリのコピーをします。転送終了時に dst（転送先のアドレス）を返します。

(5) 戻り値

戻り値	意味
dst	転送先アドレス

6.7 シリアル・フラッシュ ROM 制御

6.7.1 SPI バス制御初期化

sromc_init

(1) 概要

SPI バス制御初期化

(2) C 言語形式

```
void sromc_init(void);
```

(3) パラメータ

なし

(4) 機能

シリアル･フラッシュ ROM コントローラを初期化します。

- シリアル・フラッシュROMクロック設定
  - > シリアル・クロック周波数 : 25MHz
  - > シリアル・クロックのデフォルト・レベル : High
- シリアル・フラッシュROMデータ設定 : データ長8、MSBファースト
- シリアル・フラッシュROMタイミング設定
  - > データ入出力
    - 入力セットアップ・サイクル : 0.5シリアル・クロック
    - 出力ホールド・サイクル : 0.5シリアル・クロック
  - > SPIバスのデバイス選択信号の最小ハイ幅 : 8シリアル・クロック
  - > セットアップ・サイクル
    - SPIバスのデバイス選択信号 : 1.5シリアル・クロック
    - シリアル・データ出力 : 0.5シリアル・クロック
  - > ホールド・サイクル
    - SPIバスのデバイス選択信号 : 1.5シリアル・クロック
    - シリアル・データ出力 : 0.5シリアル・クロック

(5) 戻り値

なし

6.7.2 SPI バスへデータ書き込み

sromc_write

(1) 概要

SPI バスへデータ書き込み

(2) C 言語形式

```
void sromc_write(uint8_t data, uint32_t first, uint32_t last);
```

(3) パラメータ

I/O	パラメータ		説明
I	uint8_t	data	1バイト・キャラクタの書き込みデータ
I	uint32_t	first	SPIバスへの直接アクセスモード設定用フラグ 0：直接アクセス・モード設定無し other：直接アクセス・モード設定
I	uint32_t	last	ROMアクセスモード設定用フラグ 0：ROMアクセス・モード設定無し other：ROMアクセス・モード設定

(4) 機能

data 引数で与えた書き込みデータを SPI バスへ書き込みます。
first 引数が 0 以外の場合、データ書き込み前に直接アクセス・モードに設定します。
last 引数が 0 以外の場合、データ書き込み後に ROM アクセス・モードに設定します。

(5) 戻り値

なし

6.7.3 SPI バスからデータ読み出し

sromc_read

(1) 概要

SPI バスからデータ読み出し

(2) C 言語形式

```
void sromc_read(uint8_t* data, uint32_t first, uint32_t last);
```

(3) パラメータ

I/O	パラメータ		説明
O	uint8_t *	data	1バイト・キャラクタの読み出しデータのポインタ
I	uint32_t	first	SPIバスへの直接アクセスモード設定用フラグ 0：直接アクセス・モード設定無し other：直接アクセス・モード設定
I	unit32_t	last	ROMアクセスモード設定用フラグ 0：ROMアクセス・モード設定無し other：ROMアクセス・モード設定

(4) 機能

SPI バスから読み出したデータを data 引数で指定したポインタへ格納します。
first 引数が 0 以外の場合、読み出し前に直接アクセス・モードに設定します。
last 引数が 0 以外の場合、読み出し後に ROM アクセス・モードに設定します。

(5) 戻り値

なし

6.8 ウォッチドッグ・タイマ制御

6.8.1 ウォッチドッグ・タイマ初期化

wdt_init

(1) 概要

ウォッチドッグ・タイマ初期化

(2) C 言語形式

```
ER_RET wdt_init(void);
```

(3) パラメータ

なし

(4) 機能

ウォッチドッグ・タイマを初期化します。

- エラーモード : リセット・モード（カウンタ・オーバフローでリセット）
- カウンタ・オーバフロー・インターバル時間 : 1.342s
- ウィンドウ・オープン期間 : 100%

(5) 戻り値

戻り値	意味
ER_OK	初期化成功

6.8.2 ウォッチドッグ・タイマ起動

wdt_start

(1) 概要

ウォッチドッグ・タイマ起動

(2) C 言語形式

```
ER_RET wdt_start(void);
```

(3) パラメータ

なし

(4) 機能

ウォッチドッグ・タイマを起動します。起動後の停止は出来ません。

(5) 戻り値

戻り値	意味
ER_OK	起動成功

6.8.3 カウント・クリア

wdt_clear

(1) 概要

カウント・クリア

(2) C 言語形式

```
ER_RET wdt_clear(void);
```

(3) パラメータ

なし

(4) 機能

ウォッチドッグ・タイマのカウントをクリアします。

(5) 戻り値

戻り値	意味
ER_OK	カウント・クリア成功

6.8.4 リセット待ち

wdt_wait_reset

(1) 概要

リセット待ち

(2) C 言語形式

```
void wdt_wait_reset(void);
```

(3) パラメータ

なし

(4) 機能

カウンタのオーバフロー発生によるウォッチドッグ・タイマのリセット出力を待ちます。

(5) 戻り値

なし

6.9 CAN 制御

6.9.1 CAN コントローラの有効化

can_enable

(1) 概要

CAN コントローラの有効化

(2) C 言語形式

```
ER_RET can_enable(uint8_t ch)
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能

チャネル選択引数で指定したチャネルの CAN コントローラモジュールの起動を行います。
下記エラー発生時、ER_PARAM を返します。
・チャネル選択引数が 0 または 1 以外
・選択チャネルが使用不可
下記エラー発生時、ER_INVAL を返します。
・メッセージバッファ RAM の読み出しエラー
・CAN モジュールが既に起動済み
ソフト・リセット実行中の場合は、ER_BUSY を返します。

注意　can_init の前に呼び出す必要があります。

(5) 戻り値

戻り値	意味
ER_OK	CANコントローラ有効化成功
ER_PARAM	パラメータ・エラー
ER_INVAL	CAN Enable無効
ER_BUSY	ソフト･リセット実行中

6.9.2 CAN コントローラの初期化

can_init

(1) 概要

CAN コントローラの初期化

(2) C 言語形式

```
ER_RET can_init(uint8_t ch);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能

チャネル選択引数で指定したチャネルの CAN コントローラモジュールを CAN コンフィギュレーションテーブルに従い初期化します。

下記エラー発生時、ER_PARAM を返します。

・チャネル選択引数が 0 または 1 以外

・選択チャネルが使用不可

CAN コントローラが初期化モード以外の場合、ER_INVAL を返します。

(5) 戻り値

戻り値	意味
ER_OK	CANコントローラ初期化成功
ER_PARAM	パラメータ・エラー
ER_INVAL	CAN初期化無効

6.9.2.2 CAN コントローラ初期化コンフィギュレーションテーブル

CANコントローラの初期化ドライバ呼出時に設定されるレジスタの一部をコンフィギュレーションテーブルとして持ちます。

(1) CAN コントローラチャネル設定テーブル

can_ch_info

CAN コントローラのチャネル毎　初期設定用テーブル

・C 言語形式

```
const CAN_CHINFO_TypeDef can_ch_info[CAN_CH_NUM]
```

・CAN_CHINFO_TypeDef 構造体メンバ

I/O	メンバ	説明
I	uint8_t　use	チャネル有効設定 bit7　：チャネル有効設定 0 – チャネル無効 1 – チャネル有効 bit6-bit0：チャネル番号
I	uint16_t　FCNnCMIECTL	割込み許可設定(複合指定可能) 送信中断割り込み許可： CAN_CMIECTL_SET_TRXABT ウェイク・アップ割り込み許可： CAN_CMIECTL_SET_WAKUP アービトレーション・ロスと割り込み許可： CAN_CMIECTL_SET_ARBLST CANプロトコルエラー割り込み許可： CAN_CMIECTL_SET_PRTERR CAN エラー・ステータス割り込み許可： CAN_CMIECTL_SET_ERRSTS メッセージバッファへのメッセージ受信完了割込み許可： CAN_CMIECTL_SET_RX メッセージバッファからのメッセージ送信完了割込み許可： CAN_CMIECTL_SET_TX

(2) CAN コントローラボーレート設定テーブル

can_bps_info

CAN コントローラのチャネル毎　ボーレート設定用テーブル

・C 言語形式

```
const CAN_BPSINFO_TypeDef can_bps_info[CAN_CH_NUM]
```

・CAN_BPSINFO_TypeDef構造体メンバ

I/O	メンバ	説明
I	uint8_t　FCNnGMCSPRE	システム・クロック設定
I	uint8_t　FCNnCMBRPRS	ビット・レート・プリスケーラ設定
I	uint16_t　FCNnCMBTCTL	ビット・レート設定

参考　「R-IN32M3 シリーズ ユーザーズ・マニュアル 周辺機能編」の「19.13　ボー・レートの設定」

(3) CAN コントローラ　メッセージバッファ　ID マスク設定テーブル

can_msg_msk_info

CAN コントローラのチャネル毎　受信メッセージバッファ　ID マスク設定用テーブル

・C言語形式

```
const uint32_t can_msg_msk_info[CAN_CH_NUM][CAN_NUM_OF_MASK]
```

・can_msg_msk_info配列

I/O	メンバ	説明
I	uint32_t	IDマスクレジスタ(FCNnCMMKCTLmW)設定

参考　「R-IN32M3 シリーズ ユーザーズ・マニュアル 周辺機能編」の「19.7.4　マスク機能」

(4) CAN コントローラメッセージバッファ設定テーブル

can_msg_info

CAN コントローラのチャネル毎　メッセージバッファ設定用テーブル

・C 言語形式

```
const CAN_MSGINFO_TypeDef can_msg_info[CAN_CH_NUM][CAN_MSG_BUF_NUM]
```

・CAN_MSGINFO_TypeDef構造体メンバ

I/O	メンバ	説明
I	uint32_t　FCNnMmMID0W	CAN_ID設定 標準ID指定：CAN_SET_STD_ID(CAN_ID) 拡張ID指定：CAN_SET_EXT_ID(CAN_ID)
I	uint8_t　FCNnMmSTRB	メッセージバッファモード指定 送信モード：CAN_MSGBUF_INI_TX 受信モード(マスクレジスタ使用なし)：CAN_MSGBUF_INI_RX 受信モード(マスク1レジスタ使用)：CAN_MSGBUF_INI_RX_MSK1 受信モード(マスク2レジスタ使用)：CAN_MSGBUF_INI_RX_MSK2 受信モード(マスク3レジスタ使用)：CAN_MSGBUF_INI_RX_MSK3 受信モード(マスク4レジスタ使用)：CAN_MSGBUF_INI_RX_MSK4 受信モード(マスク5レジスタ使用)：CAN_MSGBUF_INI_RX_MSK5 受信モード(マスク6レジスタ使用)：CAN_MSGBUF_INI_RX_MSK6 受信モード(マスク7レジスタ使用)：CAN_MSGBUF_INI_RX_MSK7 受信モード(マスク8レジスタ使用)：CAN_MSGBUF_INI_RX_MSK8
I	uint8_t　FCNnMmDTLGB	DLC(Data Length Code)設定 送信モード：0～8 受信モード：0

6.9.3 CAN コントローラ強制終了

can_shutdown

(1) 概要

CAN コントローラの強制終了

(2) C 言語形式

```
ER_RET can_shutdown(uint8_t ch);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能

チャネル選択引数で指定したチャネルの CAN コントローラモジュールの強制終了を行います。
下記エラー発生時、ER_PARAM を返します。
・チャネル選択引数が 0 または 1 以外
・選択チャネルが使用不可
CAN コントローラを強制終了してもモジュール起動状態の場合、ER_INVAL を返します。

(5) 戻り値

戻り値	意味
ER_OK	CAN コントローラ初期化成功
ER_PARAM	パラメータ・エラー
ER_INVAL	CAN 強制終了無効

6.9.4 CAN 動作モード取得

can_get_mode

(1) 概要

CAN コントローラの動作モードを取得

(2) C 言語形式

```
ER_RET can_get_mode(uint8_t ch);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能

チャネル選択引数で指定したチャネルの CAN コントローラモジュールの動作モードを取得します。
下記エラー発生時、ER_PARAM を返します。
・チャネル選択引数が 0 または 1 以外
・選択チャネルが使用不可

(5) 戻り値

戻り値	意味
bit7 = 0	動作モード取得完了 ・bit2 - bit0 ：動作モード 000b – 初期化モード 001b – 通常モード 101b – セルフテストモード ・bit4 - bit3 ：パワーセーブモード 00b – 非パワーセーブモード 01b – CANスリープモード 11b – CANストップモード ・bit7 - bit5 ：0
ER_PARAM	パラメータ・エラー

6.9.5 CAN 動作モード設定

can_set_mode

(1) 概要

CAN コントローラの動作モードを設定

(2) C 言語形式

```
ER_RET can_set_mode(uint8_t ch,uint16_t mode);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t　ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t　mode	動作モード引数 SET_CAN_INIT　：初期化モード SET_CAN_NORM　：通常モード SET_CAN_SELF　：セルフテストモード

(4) 機能

チャネル選択引数で指定したチャネルの CAN コントローラモジュールの動作モードを設定します。
下記エラー発生時、ER_PARAM を返します。
・チャネル選択引数が 0 または 1 以外
・選択チャネルが使用不可
下記エラー発生時、ER_INVAL を返します。
・動作モード引数が初期化モードで、現状の動作モードが初期化モードの場合
・動作モード引数が初期化モード以外で、現状の動作モードが初期化モード以外の場合

注意　初期化モード以外の動作モードから別の動作モードへ切り替える場合は、一度初期化モードに切り替える必要があります。

(5) 戻り値

戻り値	意味
ER_OK	動作モード設定成功
ER_PARAM	パラメータ・エラー
ER_INVAL	動作モード設定無効

6.9.6 CAN 受信データ取得(CANID,Data,DLC)

can_get_id_data_dlc

(1) 概要

CAN コントローラの受信メッセージバッファより CAN ID、受信データ、DLC を取得する。

(2) C 言語形式

```
ER_RET can_get_id_data_dlc(uint8_t ch,uint8_t bufno,
                           uint32_t *canid,uint8_t *data,uint8_t *dlc);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t bufno	受信データ取得先メッセージバッファ番号
O	uint32_t *canid	CAN_ID格納先ポインタ
O	uint8_t *data	受信データ格納先ポインタ
O	uint8_t *dlc	受信データ数格納先ポインタ

(4) 機能

指定チャネル、バッファ番号により CAN コントローラ メッセージバッファから CAN ID、受信データ、DLC を取得します。

下記エラー発生時、ER_PARAM を返します。

- チャネル選択引数が 0 または 1 以外
- 選択チャネルが使用不可
- バッファ番号が範囲外
- メッセージバッファ使用不可

下記エラー発生時、ER_INVAL を返します。

- メッセージバッファに新しいデータが存在しない
- メッセージバッファが更新中

備考　メッセージバッファ番号は CAN データ受信バッファ番号取得ドライバから取得してください。

(5) 戻り値

戻り値	意味
ER_OK	受信データ取得成功
ER_PARAM	パラメータ・エラー
ER_INVAL	受信データ取得無効

6.9.7 CAN 受信データ取得(Data,DLC)

can_get_data_dlc

(1) 概要

CAN コントローラの受信メッセージバッファより受信データ、DLC を取得する。

(2) C 言語形式

```
ER_RET can_get_data_dlc(uint8_t ch,uint8_t bufno,uint8_t *data,uint8_t *dlc)
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t bufno	受信データ取得先メッセージバッファ番号
O	uint8_t *data	受信データ格納先ポインタ
O	uint8_t *dlc	受信データ数格納先ポインタ

(4) 機能

指定チャネル、バッファ番号により CAN コントローラ メッセージバッファから受信データ、DLC を取得します。

下記エラー発生時、ER_PARAM を返します。

・チャネル選択引数が 0 または 1 以外
・選択チャネルが使用不可
・バッファ番号が範囲外
・メッセージバッファ使用不可

下記エラー発生時、ER_INVAL を返します。

・メッセージバッファに新しいデータが存在しない
・メッセージバッファが更新中

備考　メッセージバッファ番号は CAN データ受信バッファ番号取得ドライバから取得してください。

(5) 戻り値

戻り値	意味
ER_OK	受信データ取得成功
ER_PARAM	パラメータ・エラー
ER_INVAL	受信データ取得無効

6.9.8　CAN 送信データ設定(CAN_ID,Data,DLC)

can_set_id_data_dlc

(1) 概要

CAN コントローラのメッセージバッファへ CAN_ID と DLC を指定して送信データを設定する。

(2) C 言語形式

```
ER_RET can_set_id_data_dlc(uint8_t ch,uint8_t bufno,
                           uint32_t canid,uint8_t *data,uint8_t dlc)
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t　ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t　bufno	送信データ設定先メッセージバッファ番号
I	uint32_t　canid	CAN_ID 標準 ID ： CAN_SET_STD_ID(canid) 拡張ID ： CAN_SET_EXT_ID(canid)
I	uint8_t　*data	送信データへのポインタ
I	uint8_t　dlc	送信データサイズ

(4) 機能

指定チャネル、バッファ番号の CAN コントローラ メッセージバッファにデータ(CAN_ID,Data,DLC)を設定します。

下記エラー発生時、ER_PARAM を返します。

- チャネル選択引数が 0 または 1 以外
- 選択チャネルが使用不可
- バッファ番号が範囲外
- メッセージバッファ使用不可

データ送信中の場合、ER_INVAL を返します。

備考　CAN_ID は CAN_SET_STD_ID または CAN_SET_EXT_ID マクロを使用して設定してください。

(5) 戻り値

戻り値	意味
ER_OK	受信データ取得成功
ER_PARAM	パラメータ・エラー
ER_INVAL	送信データ設定無効

6.9.9 CAN 送信データ設定

can_set_data

(1) 概要

CAN コントローラのメッセージバッファへ送信データを設定する。

(2) C 言語形式

```
ER_RET can_set_data(uint8_t ch,uint8_t bufno,uint8_t *data)
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t bufno	送信データ設定先メッセージバッファ番号
I	uint8_t *data	送信データへのポインタ

(4) 機能

指定チャネル、バッファ番号の CAN コントローラ メッセージバッファにデータを設定します。
下記エラー発生時、ER_PARAM を返します。

- チャネル選択引数が 0 または 1 以外
- 選択チャネルが使用不可
- バッファ番号が範囲外
- メッセージバッファ使用不可

データ送信中の場合、ER_INVAL を返します。

注意　CAN_ID、DLC は初期設定値が送信されます。
メッセージバッファ毎の CAN_ID、DLC は CAN 送信データ設定(CANID,Data,DLC)ドライバの呼び出しにより初期設定値から変更可能です。

(5) 戻り値

戻り値	意味
ER_OK	受信データ取得成功
ER_PARAM	パラメータ・エラー
ER_INVAL	送信データ設定無効

6.9.10 CAN データ送信リクエスト

can_tx_req

(1) 概要

CAN コントローラへデータ送信をリクエストを行う。

(2) C 言語形式

```
ER_RET can_tx_req(uint8_t ch,uint8_t bufno)
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t bufno	送信リクエストメッセージバッファ番号

(4) 機能

指定チャネル、バッファ番号による CAN コントローラ メッセージバッファにデータ送信リクエストを行います。

下記エラー発生時、ER_PARAM を返します。

- チャネル選択引数が 0 または 1 以外
- 選択チャネルが使用不可
- バッファ番号が範囲外
- メッセージバッファ使用不可

送信データが設定されていない場合、ER_INVAL を返します。

(5) 戻り値

戻り値	意味
ER_OK	データ送信リクエスト成功
ER_PARAM	パラメータ・エラー
ER_INVAL	データ送信リクエスト無効

6.9.11 CAN データ送信情報取得

can_get_txinfo

(1) 概要

CAN コントローラのデータ送信情報を取得する

(2) C 言語形式

```
ER_RET can_get_txinfo(uint8_t ch,uint8_t bufno)
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t bufno	バッファ番号

(4) 機能

指定チャネル、バッファ番号の CAN コントローラ メッセージバッファのデータ送信状態を取得。
下記エラー発生時、ER_PARAM を返します。

- チャネル選択引数が 0 または 1 以外
- 選択チャネルが使用不可
- バッファ番号が範囲外
- メッセージバッファ使用不可

データ送信中の場合、ER_BUSY を返します。

(5) 戻り値

戻り値	意味
ER_OK	送信データなし(バッファ・エンプティ)
ER_PARAM	パラメータ・エラー
ER_BUSY	データ送信中

6.9.12 CAN データ受信バッファ番号取得

can_get_rxinfo

(1) 概要

CAN コントローラのデータ受信したメッセージバッファ番号を取得する

(2) C 言語形式

```
ER_RET can_get_rxinfo(uint8_t ch,uint8_t bufno)
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t bufno	メッセージバッファ探索開始バッファ番号

(4) 機能

指定チャネルの CAN コントローラ メッセージバッファから受信データの探索を行う。
バッファ番号を始点として、以降のメッセージバッファを探索する。
下記エラー発生時、ER_PARAM を返します。

- チャネル選択引数が 0 または 1 以外
- 選択チャネルが使用不可
- バッファ番号が範囲外
- メッセージバッファ使用不可

受信データが見つからなかった場合、ER_INVAL を返します。

(5) 戻り値

戻り値	意味
bit7 = 0	受信データあり bit6 – bit0 : 受信バッファ番号
ER_PARAM	パラメータ・エラー
ER_INVAL	受信データなし

6.9.13 CAN チャネルステータス取得

can_get_ch_status

(1) 概要

CAN コントローラのチャネルステータスを取得する

(2) C 言語形式

```
ER_RET can_get_ch_status(uint8_t ch)
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能

指定チャネルの CAN コントローラ チャネルステータスを取得する
下記エラー発生時、ER_PARAM を返します。
・チャネル選択引数が 0 または 1 以外
・選択チャネルが使用不可

(5) 戻り値

戻り値	意味
bit7 = 0	チャネルステータス取得 bit0 ： メッセージバッファmからの送信完了 bit1 ： メッセージバッファmへの受信完了 bit2 ： CAN エラーステータス bit3 ： CAN プロトコルエラー bit4 ： アービトレーション・ロスト bit5 ： CAN スリープモードからの復帰 bit6 ： 送信中断
ER_PARAM	パラメータ・エラー

6.9.14 CAN チャネルステータスクリア

can_clr_ch_status

(1) 概要

CAN コントローラのチャネルステータスをクリアする

(2) C 言語形式

```
ER_RET can_clr_ch_status(uint8_t ch,uint8_t clrdat)
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t clrdat	チャネルステータスクリアデータ bit0 ： メッセージバッファmからの送信完了 bit1 ： メッセージバッファmへの受信完了 bit2 ： CAN エラーステータス bit3 ： CAN プロトコルエラー bit4 ： アービトレーション・ロスト bit5 ： CAN スリープモードからの復帰 bit6 ： 送信中断

(4) 機能

指定チャネルの CAN コントローラ チャネルステータスをクリアする
下記エラー発生時、ER_PARAM を返します。
・チャネル選択引数が 0 または 1 以外
・選択チャネルが使用不可

(5) 戻り値

戻り値	意味
ER_OK	チャネルステータスクリア成功
ER_PARAM	パラメータ・エラー

6.9.15 CAN バスステータス取得

can_get_bus_staus

(1) 概要

CAN コントローラのバスステータスを取得する

(2) C 言語形式

```
ER_RET can_get_bus_staus(uint8_t ch)
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	チャネル選択引数 0：チャネル0 1：チャネル1

(4) 機能

指定チャネルの CAN コントローラ バスステータスを取得する。
下記エラー発生時、ER_PARAM を返します。
・チャネル選択引数が 0 または 1 以外
・選択チャネルが使用不可

(5) 戻り値

戻り値	意味
bit7 = 0	チャネルステータス取得 bit1 - bit0 : 受信エラーカウンタステータス bit3 - bit2 : 送信エラーカウンタステータス bit4: バスオフステータス bit7 – bit5 : 0
ER_PARAM	パラメータ・エラー

7. ミドルウェア

ミドルウェアの関数説明を示します。

7.1 ミドルウェア関数一覧

サンプル・ソフトで使用されている API の一覧を示します。

表 7.1 EEPROM 関数一覧

関数名	関数概要
eep_init	EEPROM 制御の初期化
eep_write	EEPROM データ書き込み
eep_read	EEPROM データ読み出し

表 7.2 パラレル・フラッシュ ROM ドライバ関数一覧

関数名	関数概要
flash_init	パラレル・フラッシュ ROM 制御初期化
flash_program	データ書き込み
flash_read_data	データ読み出し
flash_erase	データ消去
flash_read_cfi	CFI テーブル・リード

表 7.3 シリアル・フラッシュ ROM ドライバ関数一覧

関数名	関数概要
sflash_init	シリアル・フラッシュ ROM 制御初期化
sflash_program	シリアル・フラッシュ ROM へデータ・プログラム
sflash_read	シリアル・フラッシュ ROM データ読み出し
sflash_erase	シリアル・フラッシュ ROM のデータ消去

7.2 EEPROM 制御

7.2.1 EEPROM 制御の初期化

eep_init

(1) 概要

EEPROM 制御の初期化

(2) C 言語形式

```
ER_RET eep_init(uint8_t ch, uint8_t iic_adr);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint8_t ch	IICチャネル選択引数 0：チャネル0 1：チャネル1
I	uint8_t iic_adr	デバイス選択引数（0～7）

(4) 機能

iic_init で、IIC チャネル選択引数で指定した IIC コントローラを初期化します。
IIC チャネル選択引数は iic_init のチャネル選択引数（iic_ch_eep）として使用します。
デバイス選択引数でライト時、リード時に動作させるデバイスを選択します。
IIC チャネル選択引数が範囲外の場合、iic_init にて ER_PARAM（パラメータ・エラー）を返します。
また、デバイス選択引数が範囲外の場合、ER_PARAM（パラメータ・エラー）を返します。

(5) 戻り値

戻り値	意味
ER_PARAM	パラメータ・エラー ・チャネル選択引数が0, 1以外の場合 ・デバイス選択引数が範囲外の場合
ER_OK	初期化成功

7.2.2　EEPROM データ書き込み

eep_write

(1) 概要

EEPROM データ書き込み

(2) C 言語形式

```
ER_RET eep_write(uint32_t eep_addr, uint8_t *data, uint32_t len);
```

(3) パラメータ

I/O	パラメータ		説明
I	uint32_t	eep_addr	EEPROM書き込み開始アドレス
I	uint8_t *	data	送信データのポインタ
I	uint32_t	len	ライト・サイズ指定引数

(4) 機能

IIC コントローラを動作させ、eep_adr 指定アドレスから len 指定サイズ分のデータを EEPROM へ送信します。len 指定サイズがページ・サイズ（デバイス固有）を超える場合、ページ・サイズ分のライトを len 指定サイズまで繰り返します。書き込みデータは、*data 指定アドレスから len 指定サイズ分のデータを EEPROM に書き込みます。

指定した最終アドレスがデバイス容量（デバイス固有）を超える場合 ER_PARAM（パラメータ・エラー）を返します。

IIC コントローラからの戻り値が ER_OK（送信成功）以外の場合の動作を以下に示します。

- デバイス選択情報送信時　：ストップ・コンディション発行、書き込み失敗を返す
- バイト・アドレス（High）送信時　：ストップ・コンディション発行、書き込み失敗を返す
- バイト・アドレス（Low）送信　：ストップ・コンディション発行、書き込み失敗を返す
- 指定したライト・サイズ分のデータ送信　：ストップ・コンディション発行、書き込み失敗を返す

(5) 戻り値

戻り値	意味
ER_OK	書き込み成功
ER_NG	書き込み失敗 ・IICドライバ処理結果がエラーの場合
ER_PARAM	パラメータ・エラー ・指定した最終アドレスがデバイス容量を超える場合、

7.2.3　EEPROM データ読み出し

eep_read

(1) 概要

EEPROM データ読み出し

(2) C 言語形式

```
ER_RET eep_read(uint32_t eep_addr, uint8_t *data, uint32_t len);
```

(3) パラメータ

I/O	パラメータ		説明
I	uint32_t	eep_addr	読み出し開始アドレス
O	uint8_t *	data	受信データのポインタ
I	uint32_t	len	リード・サイズ

(4) 機能

IIC コントローラを動作させ、eep_adr 指定アドレスから len 指定サイズ分のデータを EEPROM へ受信します。読み出しデータは、EEPROM 内のデータを*data 指定アドレスから len 指定サイズ分の領域に書き出します。指定した最終アドレスがデバイス容量（デバイス固有）を超える場合 ER_PARAM（パラメータ・エラー）を返します。

IIC コントローラからの戻り値が ER_OK（送信成功）以外の場合の動作を以下に示します。

- デバイス選択情報送信時　：ストップ・コンディション発行、書き込み失敗を返す
- バイト・アドレス（High）送信時　：ストップ・コンディション発行、書き込み失敗を返す
  バイト・アドレス（Low）送信　：ストップ・コンディション発行、書き込み失敗を返す
- リードするデバイス選択情報送信時　：通信失敗を返す
- 指定したライト・サイズ分のデータ送信　：ストップ・コンディション発行、書き込み失敗を返す

(5) 戻り値

戻り値	意味
ER_OK	読み出し成功
ER_NG	読み出し失敗 ・IICドライバ処理結果がエラーの場合
ER_PARAM	パラメータ・エラー ・指定した最終アドレスがデバイス容量を超える場合、

7.3 パラレル・フラッシュ ROM 制御

7.3.1 パラレル・フラッシュ ROM 制御の初期化

flash_init

(1) 概要

パラレル・フラッシュ ROM 制御の初期化

(2) C 言語形式

```
ER_RET flash_init(void);
```

(3) パラメータ

なし

(4) 機能

パラレル・フラッシュ ROM 制御の初期化を行います。

(5) 戻り値

戻り値	意味
ER_OK	初期化成功

7.3.2　データ書き込み

flash_program

(1) 概要

データ書き込み

(2) C 言語形式

```
ER_RET flash_program(uint16_t* buf, uint32_t addr, uint32_t size);
```

(3) パラメータ

I/O	パラメータ		説明
I	uint16_t*	buf	書き込みデータ格納領域の開始アドレス
I	uint32_t	addr	書き込みアドレス
I	uint32_t	size	書き込みサイズ（バイト指定）

(4) 機能

addr 引数指定の書き込みアドレスから size 引数指定のサイズ分のパラレル・フラッシュ ROM 領域へデータを書き込みます。書き込みデータは、*buf 引数指定のアドレス領域のデータを使用します。

(5) 戻り値

戻り値	意味
ER_OK	成功
ER_PARAM	パラメータ・エラー ・ addr引数が16ビットのアドレス境界値でない場合 ・ size引数が16ビットのアドレス境界値でない場合 ・ addr引数とsize引数の加算値がパラレル・フラッシュROMの最大サイズを超える場合

7.3.3　データ読み出し

flash_read_data

(1) 概要

データ読み出し

(2) C 言語形式

```
ER_RET flash_read_data(uint16_t* buf, uint32_t addr, uint32_t size);
```

(3) パラメータ

I/O	パラメータ		説明
O	uint16_t*	buf	読み出しデータの書き込み開始アドレス
I	uint32_t	addr	読み出しアドレス
I	uint32_t	size	読み出しサイズ（バイト指定）

(4) 機能

addr 引数指定の読み出しアドレスから size 引数指定のサイズ分のパラレル・フラッシュ ROM 領域からデータを読み出します。読み出したデータは、*buf 引数指定のアドレス領域へ書き出します。

(5) 戻り値

戻り値	意味
ER_OK	成功
ER_PARAM	パラメータ・エラー ・ addr引数が16ビットのアドレス境界値でない場合 ・ size引数が16ビットのアドレス境界値でない場合 ・ addr引数とsize引数の加算値がパラレル・フラッシュROMの最大サイズを超える場合

7.3.4 データ消去

flash_erase

(1) 概要

データ消去

(2) C 言語形式

```
ER_RET flash_erase(uint32_t addr, uint32_t size);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint32_t addr	消去アドレス
I	uint32_t size	消去サイズ（バイト指定）

(4) 機能

addr 引数指定の消去アドレスから size 引数指定のサイズ分のパラレル・フラッシュ ROM 領域に格納されたデータを消去します。消去単位は、パラレル・フラッシュ ROM のセクタ・サイズに依存します。

(5) 戻り値

戻り値	意味
ER_OK	成功
ER_PARAM	パラメータ・エラー ・ addr引数とsize引数の加算値がパラレル・フラッシュROMの最大サイズを超える場合

7.3.5 CFI データのリード

flash_read_cfi

(1) 概要

CFI データのリード

(2) C 言語形式

```
ER_RET flash_read_cfi(uint16_t* buf, uint32_t addr);
```

(3) パラメータ

I/O	パラメータ		説明
O	uint16_t*	buf	バッファアドレスのポインタ
I	uint32_t	addr	リード・アドレス指定引数

(4) 機能

リード・アドレス指定引数で指定したアドレスから CFI データを参照し、バッファ・アドレスのポインタに格納します。

(5) 戻り値

戻り値	意味
ER_OK	成功
ER_PARAM	パラメータ・エラー ・ addr引数が16ビットのアドレス境界値でない場合 ・ addr引数がパラレル・フラッシュROMの最大サイズを超える場合
ER_INVAL	CFIをサポートしていない

7.4 シリアル・フラッシュ ROM 制御

7.4.1 シリアル・フラッシュ ROM 制御初期化

sflash_init

(1) 概要

シリアル・フラッシュ ROM 制御初期化

(2) C 言語形式

```
ER_RET sflash_init(uint32_t ch);
```

(3) パラメータ

I/O	パラメータ	説明
I	uint32_t ch	ドライバ選択引数 0：シリアル・フラッシュROMコントローラ

(4) 機能

ドライバ選択引数で指定したドライバを用いてシリアル・フラッシュ ROM 制御の初期化を行います。
ドライバ選択引数が 0 以外の場合は、ER_PARAM（パラメータ・エラー）を返します。

(5) 戻り値

戻り値	意味
ER_PARAM	パラメータ・エラー ・ドライバ選択引数が0以外の場合

7.4.2　シリアル・フラッシュ ROM のデータ書き込み

sflash_program

(1) 概要

シリアル・フラッシュ ROM へデータ書き込み

(2) C 言語形式

```
ER_RET sflash_program(uint32_t ch, uint8_t* buf, uint32_t addr, uint32_t size);
```

(3) パラメータ

I/O	パラメータ		説明
I	uint32_t	ch	ドライバ選択引数 0：シリアル・フラッシュROMコントローラ
I	uint8_t *	buf	書き込みデータ格納バッファ・ポインタ
I	uint32_t	addr	書き込み開始アドレス
I	uint32_t	size	書き込みデータサイズ

(4) 機能

ドライバ選択引数で指定したドライバを用いて、バッファ内のデータをシリアル・フラッシュ ROM の addr 引数及び size 引数で指定した領域に書き込みます。

ドライバ選択引数が 0 以外の場合は、ER_PARAM（パラメータ・エラー）を返します。

(5) 戻り値

戻り値	意味
ER_OK	成功
ER_PARAM	パラメータ・エラー ・ドライバ選択引数が0以外の場合

7.4.3　シリアル・フラッシュ ROM のデータ読み出し

sflash_read

(1) 概要

シリアル・フラッシュ ROM のデータ読み出し

(2) C 言語形式

```
ER_RET sflash_read(uint32_t ch, uint8_t* buf, uint32_t addr, uint32_t size);
```

(3) パラメータ

I/O	パラメータ		説明
I	uint32_t	ch	ドライバ選択引数 0：シリアル・フラッシュROMコントローラ
O	uint8_t *	buf	読み出しデータ格納バッファ・ポインタ
I	uint32_t	addr	読み出し開始アドレス
I	uint32_t	size	読み出しデータサイズ

(4) 機能

ドライバ選択引数で指定したドライバを用いて、シリアル・フラッシュ ROM の addr 引数及び size 引数で指定した領域のデータをバッファへ格納します。

ドライバ選択引数が 0 以外の場合は、ER_PARAM（パラメータ・エラー）を返します。

(5) 戻り値

戻り値	意味
ER_OK	成功
ER_PARAM	パラメータ・エラー ・ドライバ選択引数が0以外の場合

7.4.4　シリアル・フラッシュ ROM のデータ消去

sflash_erase

(1) 概要

シリアル・フラッシュ ROM のデータ消去

(2) C 言語形式

```
ER_RET sflash_erase(uint32_t ch, uint32_t addr, uint32_t size);
```

(3) パラメータ

I/O	パラメータ		説明
I	uint32_t	ch	ドライバ選択引数 0：シリアル・フラッシュROMコントローラ
I	uint32_t	addr	データ消去開始アドレス
I	uint32_t	size	データ消去データサイズ

(4) 機能

ドライバ選択引数で指定したドライバを用いて、シリアル・フラッシュ ROM の addr 引数及び size 引数で指定した領域を含む最小消去可能サイズのデータを消去します。

ドライバ選択引数が 0 以外の場合は、ER_PARAM（パラメータ・エラー）を返します。

(5) 戻り値

戻り値	意味
ER_OK	成功
ER_PARAM	パラメータ・エラー ・ドライバ選択引数が0以外の場合

8. アプリケーション例

8.1 OS レス・サンプル

以下に OS レス・サンプルを示します。OS レス・サンプルはハードウェア OS を使用しないシングル・タスクのプログラム・サンプルです。

8.1.1 OS レス・サンプル・フロー

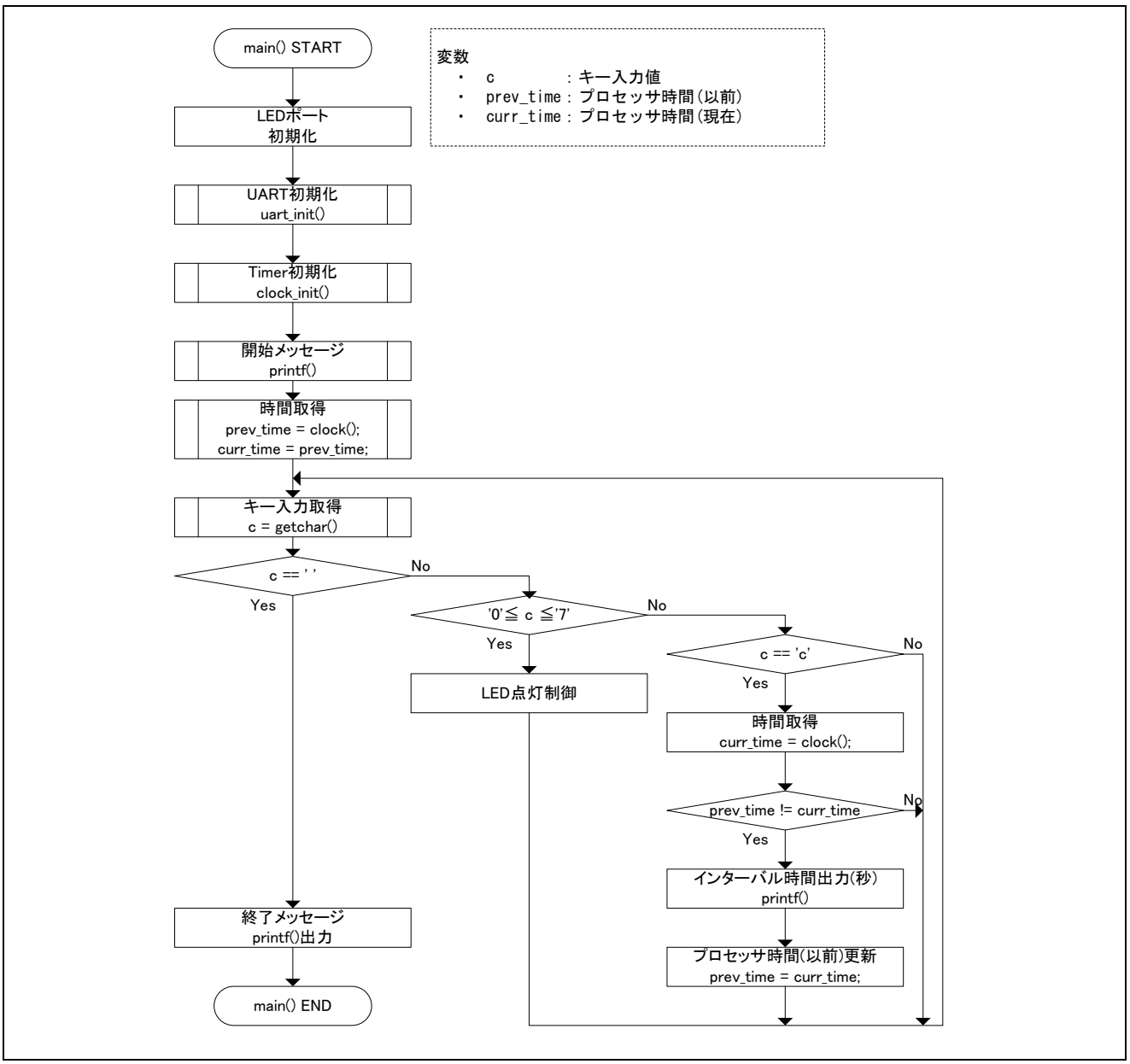

図 8.1　OS レス・サンプル　フローチャート

8.1.2　実行結果

以下に、OS レス・サンプル実行結果を示します。

リスト 8.1　OS レス・サンプル：起動

```
hello world
- compiler  =ARM 4.2 (EDG gcc mode)
- boot mode =Parallel Flash
```

備考 1. ”compiler”の表示は、ビルド環境により変化します。
2. ”boot mode”の表示は、ブート設定により変化します。

リスト 8.2　OS レス・サンプル：LED 操作（キー入力=’0’～’7’）

```
hello world
- compiler  =ARM 4.2 (EDG gcc mode)
- boot mode =Parallel Flash
0                                          (← LED0 のレベルが 1 回反転します)
11                                         (← LED1 のレベルが 2 回反転します)
222                                        (← LED2 のレベルが 3 回反転します)
3333                                       (← LED3 のレベルが 4 回反転します)
44444                                      (← LED4 のレベルが 5 回反転します)
555555                                     (← LED5 のレベルが 6 回反転します)
6666666                                    (← LED6 のレベルが 7 回反転します)
77777777                                   (← LED7 のレベルが 8 回反転します)
```

リスト 8.3　OS レス・サンプル：インターバル時間表示（キー入力=’c’）

```
hello world
- compiler  =ARM 4.2 (EDG gcc mode)
- boot mode =Parallel Flash
c (interval =464369 ms)                    (← プログラム起動から 1 回目の'c'入力までの時間を表示)
c (interval =2771 ms)                      (← 1 回目の'c'入力から 2 回目の'c'入力までの時間を表示)
c (interval =187253 ms)                    (← 2 回目の'c'入力から 3 回目の'c'入力までの時間を表示)
```

リスト 8.4　OS レス・サンプル：終了（キー入力=’ ’）

```
hello world
- compiler  =ARM 4.2 (EDG gcc mode)
- boot mode =Parallel Flash
                                           (← ' '入力でメイン処理からブレークします)
bye
```

リスト 8.5　OS レス・サンプル：インターバル時間表示（キー入力=その他）

```
hello world
- compiler  =ARM 4.2 (EDG gcc mode)
- boot mode =Parallel Flash
890-^¥!"#$%&'()=~|qwertyuiop@[QWERTYUIO`{  (← 何も処理はありません)
asdfghjkl:;]ASDFGHJKL+*}zxvbnm,./¥ZXVBNM<>?_  (← 何も処理はありません)
```

8.2 EEPROM サンプル

以下に EEPROM サンプルを示します。EEPROM サンプルは、EEPROM へのデータのライト、リード、ダウンロードを行います。

注意. 本アプリケーションでは、「CR+LF」の改行文字コードを期待しています。

8.2.1 EEPROM サンプル・フロー

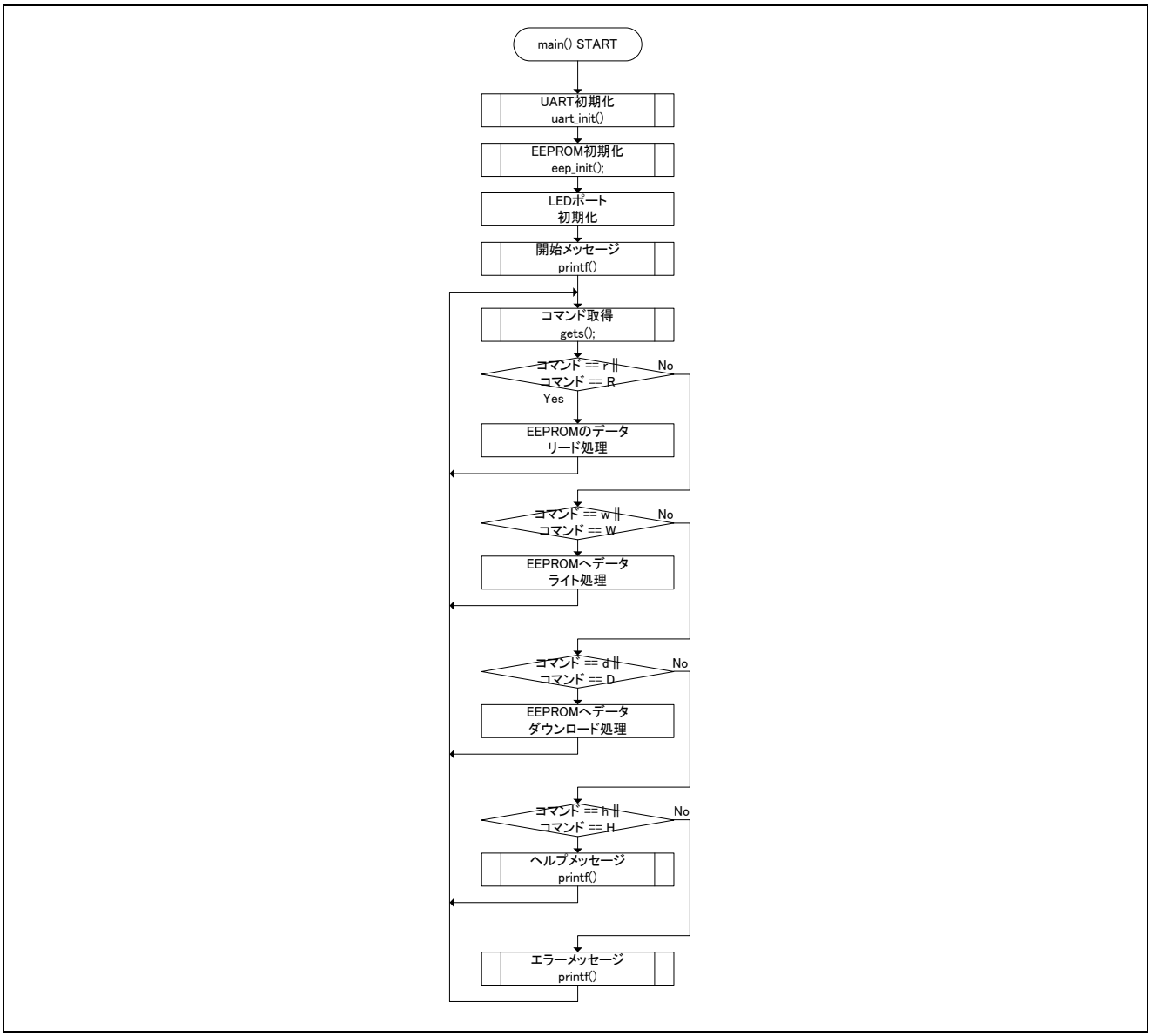

図 8.2　EEPROM サンプル　フローチャート（全体フロー）

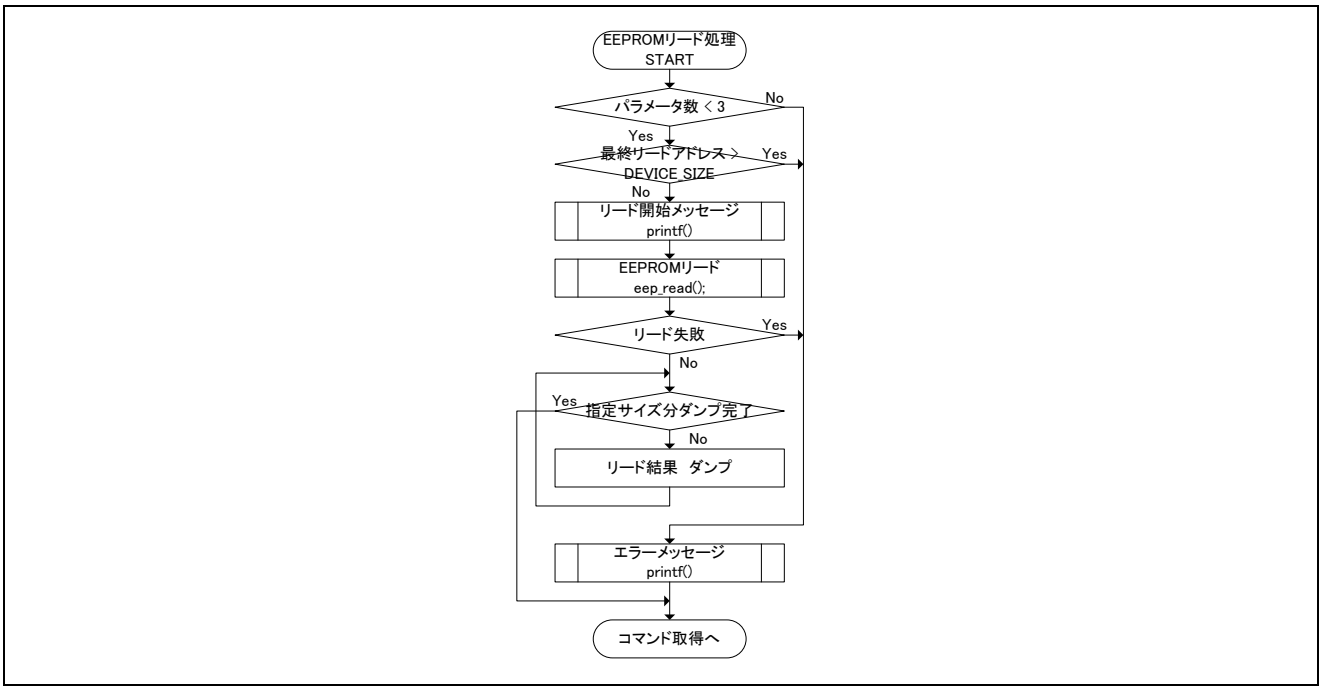

図 8.3　EEPROM サンプル　フローチャート（リードコマンド実行時）

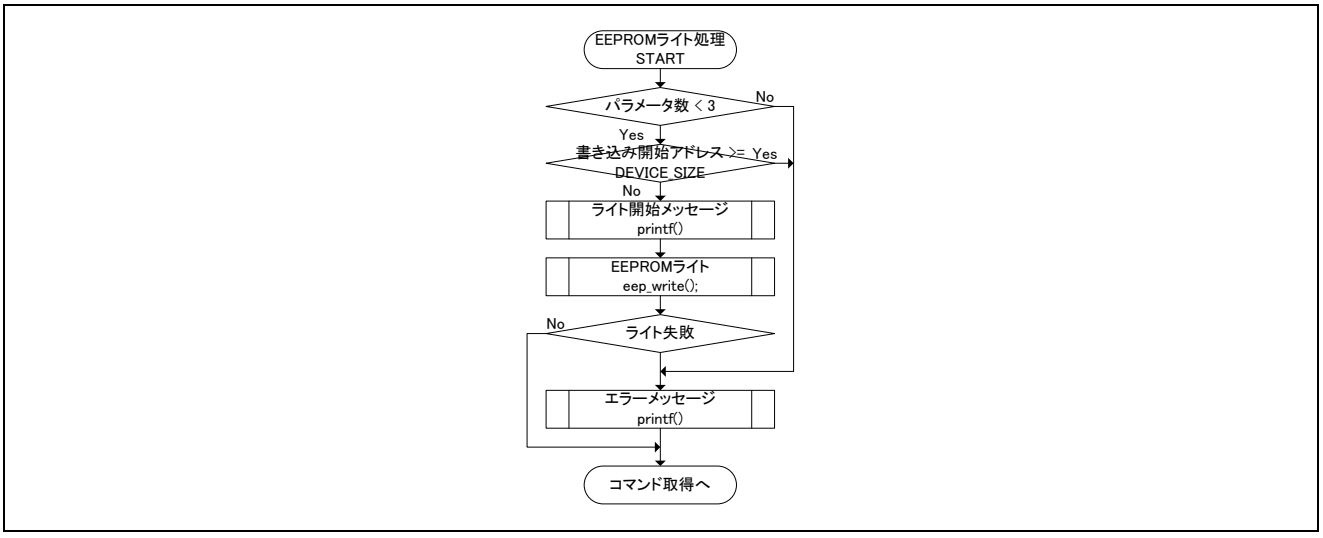

図 8.4　EEPROM サンプル　フローチャート（ライトコマンド実行時）

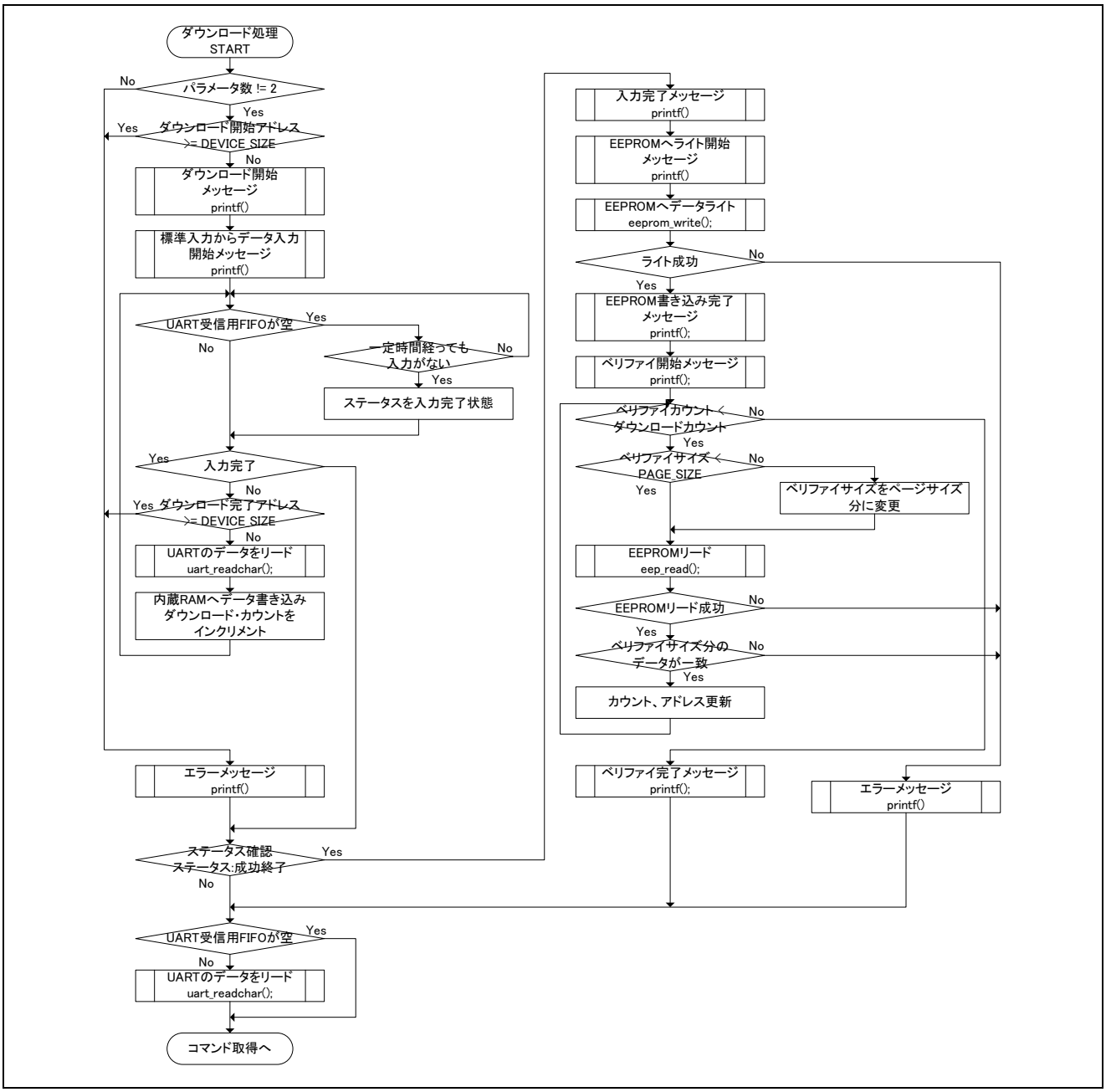

図 8.5　EEPROM サンプル　フローチャート（ダウンロードコマンド実行時）

8.2.2　実行結果

以下に、EEPROM サンプル実行結果を示します。本実行結果の対象デバイスは AT24C16C です。

リスト 8.6　EEPROM サンプル：起動

```
I2C EEPROM Writer
>
```

リスト 8.7　EEPROM サンプル：リードコマンド実行

```
I2C EEPROM Writer
> r 0 800
  READ : address=0x00000000 size=0x00000800                    (←アドレス 0 から 2Kbyte リード)
  0x00000000: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff   (←1 ライン最大 16byte ずつ表示)
  0x00000010: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
  0x00000020: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
                              :
                              :
  0x000007D0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
  0x000007E0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
  0x000007F0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
```

リスト 8.8　EEPROM サンプル：リードコマンド　パラメータ・エラー

```
I2C EEPROM Writer
> r 7ff                                                        (←引数は 3 つ以上で有効)
  parameter error : (2) - - -                                  (←パラメータの数を表示)
> r 7ff 800                                                    (←デバイスの最大容量は 2Kbyte)
  parameter error : (3) r 7FF 800                              (←読み出し最終アドレスが最大容量を超える場合)
```

リスト 8.9　EEPROM サンプル：ライトコマンド実行

```
I2C EEPROM Writer
> w 0 a                                                        (←アドレス 0 へ a をライト)
  WRITE : address=0x00000000 data=0x0a                         (←開始アドレスとデータ(アスキーコード)を表示)
```

リスト 8.10　EEPROM サンプル：ライトコマンド　パラメータ・エラー

```
I2C EEPROM Writer
> w 100                                                        (←引数は 3 つ以上で有効)
  parameter error : (2) - - -                                  (←パラメータの数を表示)
> w 8ff a                                                      (←デバイスの最大容量は 2Kbyte)
  parameter error : (3) w 8ff a                                (←書き込み開始アドレスがデバイスの最大容量を超える場合)
```

リスト 8.11　EEPROM サンプル：ダウンロードコマンド実行

```
I2C EEPROM Writer
> d 0                                                          (←アドレス 0 からダウンロード)
  DOWNLOAD : address=0x00000000                                (←開始アドレス表示)
  receive download data from standard input to buffer  ... done (2048   (←標準出力から内蔵 RAM へ入力完了サイズ表
byte)                                                          示)
  write download data from buffer to eeprom           ... done  (←内蔵 RAM から EEPROM へ書き込み完了)
  verify download data between eeprom and buffer      ... done  (←ベリファイ完了)
```

リスト 8.12　EEPROM サンプル：ダウンロードコマンド　パラメータエラー

```
I2C EEPROM Writer
> d 0 0                                                        (←引数は 2 つで有効)
  parameter error : (3) - - -                                  (←エラーとなるパラメータの数を表示)
> d 8ff                                                        (←デバイスの最大容量(2Kbyte)を超える開始アドレス)
  parameter error : (2) d 8ff                                  (←エラーとなるパラメータを表示)
```

リスト 8.13　EEPROM サンプル：ヘルプコマンド実行

```
I2C EEPROM Writer
> h                                    (←コマンドのヘルプ表示)
  r [addr] [size]                      (←リードコマンドの引数は開始アドレスとリードサイズです)
  w [addr] [data]                      (←ライトコマンドの引数は開始アドレスと 8bit データです)
  d [addr]                             (←ダウンロードコマンドの引数は開始アドレスです)
  h                                    (←ヘルプコマンドです)
```

リスト 8.14　EEPROM サンプル：上記以外のコマンド実行（コマンドエラー）

```
I2C EEPROM Writer
> x 0 800                              (←r,R/w,W/d,D/h,H 以外のコマンドはコマンドエラーです)
  Command error !!
```

9. 開発ツール依存設定

9.1 Arm

9.1.1 スタートアップ

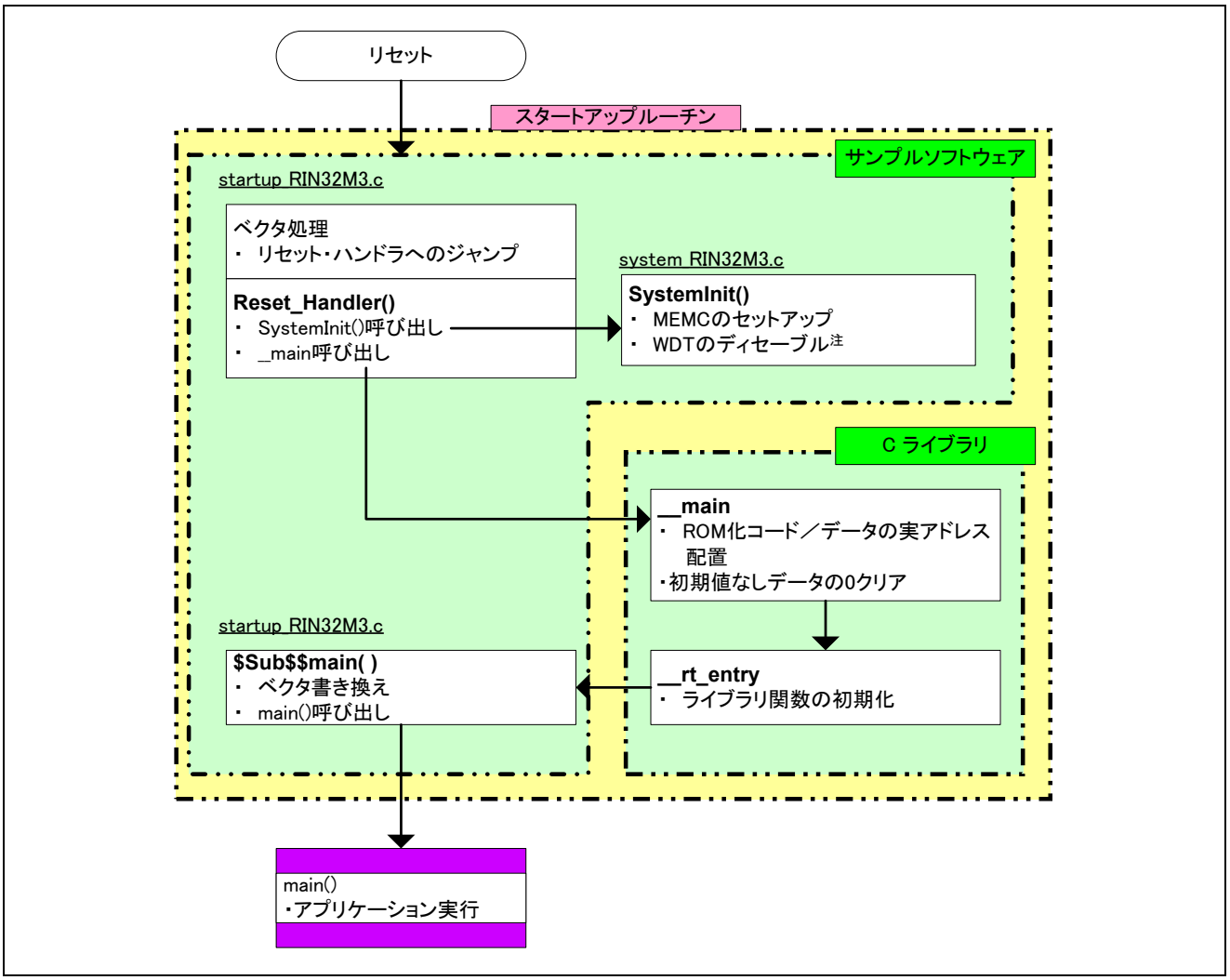

図 9.1　Arm 使用時のスタートアップ・ルーチン

注. R-IN32M3-CL 使用時

9.1.2　ライブラリ関数の再定義

syscalls.c により、ライブラリ関数の再定義を行います。

表 9.1　Arm ライブラリ関数の再定義

関数名	関数概要
fputc	UART へ 1 バイト分の送信処理を行います。 戻り値は、送信データを返します。
fgetc	UART から 1 バイトの受信処理を行い、受信結果をエコーバックします。 戻り値は、受信データを返します。
ferror	ファイル・エラー時の処理を行います。処理は実装されていません。 戻り値は、EOF を返します。
_sys_exit	システム終了時の処理を行います。関数内部で無限ループに入ります。
_ttywrch	UART へ 1 バイト分の送信処理を行います。 戻り値は、ありません。
_clock_init	何も行いません。
clock	インターバル・タイマのインターバル・カウンタを使用して、経過プロセッサ時間（clock_t 型）を返します。時刻精度は、80ns です。

9.2 GNU

9.2.1 スタートアップ

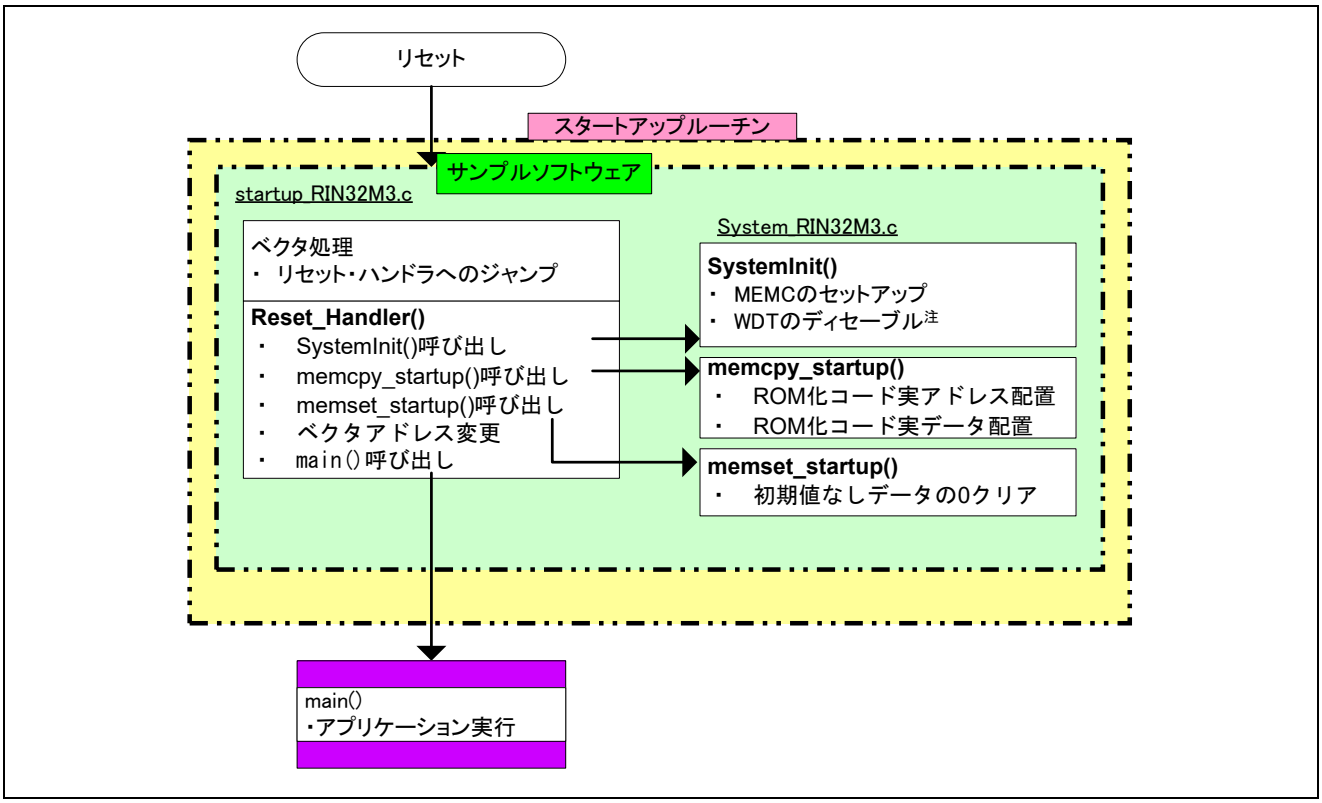

図 9.2　GNU 使用時のスタートアップ・ルーチン

注. R-IN32M3-CL 使用時

9.2.2　ライブラリ関数の再定義

syscalls.c により、ライブラリ関数の再定義を行います。

表 9.2　GNU ライブラリ関数の再定義

関数	概要
_read_r	UART から 1 バイトの受信処理を行い、受信結果をエコーバックします。戻り値は、1（受信バイト数）を返します。
_lseek_r	何も行いません。戻り値は、0 を返します。
_write_r	UART へ指定バイト数分の送信処理を行います。 戻り値は、送信バイト数を返します。
_open_r	何も行いません。戻り値は、-1 を返します。
_close_r	何も行いません。戻り値は、0 を返します。
_sbrk_r	スタック・ポインタの値がヒープ領域に重なっているかの確認を行います。戻り値は、領域が重なっている場合、-1 を返します。領域が重なっていない場合、ヒープのエンド・アドレスを返します。
_fstat_r	ファイル構造体を 0 初期化します。戻り値は、0 を返します。
_isatty_r	何も行いません。戻り値は、1 を返します。
_times_r	インターバル・タイマのインターバル・カウンタを使用して、ユーザ時間（tms.tms_utime）を返します。ユーザ・タイムの精度は、80ns です。戻り値は -1 を返します。 システム時間（tms. tms_stime）、子プロセスのユーザ時間（tms. tms_cutime）、子プロセスのシステム時間（tms. tms_cstime）は常に 0 を返します。

9.3 IAR

9.3.1 スタートアップ

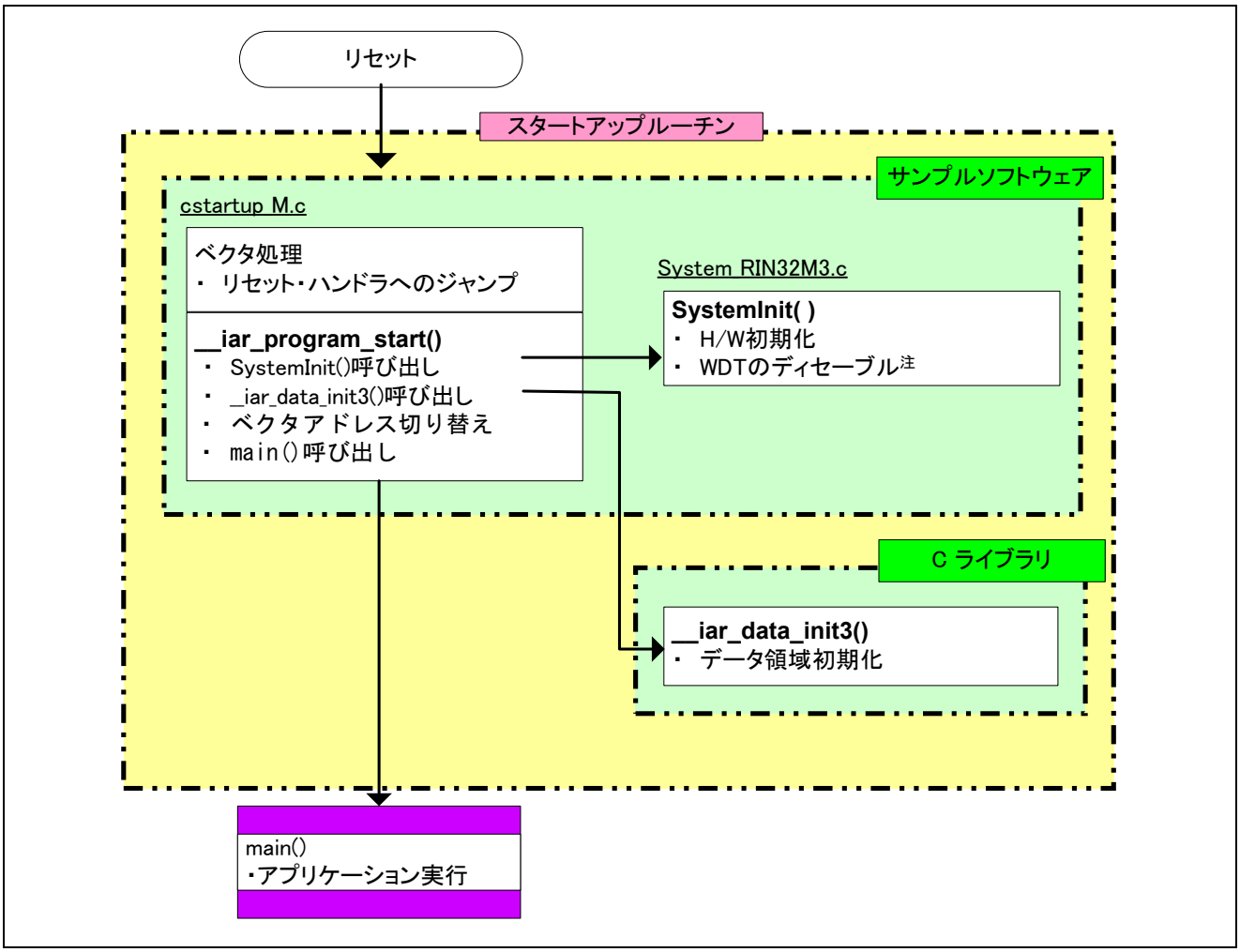

図 9.3　IAR 使用時のスタートアップ・ルーチン

注. R-IN32M3-CL 使用時

9.3.2　ライブラリ関数の再定義

syscalls.c により、ライブラリ関数の再定義を行います。

表 9.3　IAR ライブラリ関数の再定義

関数名	関数概要
__write	UART から 1 バイトの送信処理を行います。 戻り値は、送信データサイズを返します。
__read	UART から 1 バイトの受信処理を行います。 戻り値は、受信データサイズを返します。
clock	インターバル・タイマのインターバル・カウンタを使用して、経過プロセッサ時間（clock_t 型）を返します。時刻精度は、80ns です。

改訂記録	R-IN32M3 シリーズ R-IN32M3 シリーズプログラミング・マニュアル（ドライバ編）

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2013.03.29	-	新規発行
2.00	2013.06.13	このマニュアルの使い方	*R-IN32M3 に関する資料* の資料番号を修正
		全般	表現の統一 「参照ください」→「参照してください」
		1	*図 1.1サンプル・ソフトのレイヤ構成図* 修正。CC-Link IE と CC-Link は 3rd Party がサポート。Modbus TCP、FL-net は現時点で非対応のため削除。
		2	*表 1.1ソフトウェア開発ツール一覧（ツールチェーン）* 更新 *表 1.2ソフトウェア開発ツール一覧（開発環境）* 更新
		4	*表 2.2インクルード・ファイル・ディレクトリのファイル構成* ファイル追加 「RIN32M3_CL.h」
		5	*表 2.3ライブラリ・ディレクトリのファイル構成* 説明内容修正 「HW-RTOS ドライバ用プロジェクト・ファイル」→「IAR プロジェクト関連ファイル」。メイクファイル削除。
		8	*表 2.7サンプル・アプリケーション・ディレクトリのファイル構成* microVIEW のプロジェクトファイル名を変更「Cortex-M3」->「rin32m3ec」
		23	*5.2.1タイマ・モジュールの初期化* 備考欄追加
		24	*5.2.2インターバル・タイマ・モード初期化* (2)C 言語形式の脱字修正 「timer_」、 備考欄追加
		25	*5.2.3ワン・カウント・タイマ・モード初期化（ソフトウェアトリガ）* 項目名修正 「ソフトトリガ」→「ソフトウェアトリガ」、備考欄追加
		26	*5.2.4ワン・カウント・タイマ・モード初期化（ハードウェアトリガ）* 機能説明修正、注意書き追加、備考欄追加
		48, 49	*5.7.2SPI バスへデータ書き込み* 誤記修正「ダイレクト・アクセス」→「直接アクセス」
		56	*6.2.2EEPROM データ書き込み* (4)機能の修正（デバイス選択情報送信時にリトライを行わない）
		57	*6.2.3EEPROM データ読み出し* (4)機能の修正（デバイス選択情報送信時にリトライを行わない）
		62	*6.3.5CFI データのリード* 用語修正 「CFI テーブル」→「CFI データ」
		64	*6.4.2シリアル・フラッシュ ROM のデータ書き込み* 「書き込み前に消去は行わない」に内容を修正
		69	*7.2EEPROM サンプル* 期待する改行文字コードの記述を、注意書きに変更
		71	*図 7.5EEPROM サンプル　フローチャート(ダウンロードコマンド実行時)* 誤記修正 「内臓」→「内蔵」
		72	*リスト 7.11　EEPROM サンプル：ダウンロードコマンド実行* 誤記修正 「内臓」→「内蔵」
3.00	2013.09.27	1	*1.1 サンプル・ソフトのレイヤ構成図* DMA を追加
		2	*表 1.1ソフトウェア開発ツール一覧（ツールチェーン）* IAR のバージョン変更: 6.50 -> 6.60.1
		5	*表 2.3ライブラリ・ディレクトリのファイル構成* Make 関連ファイル削除。
		6	*表 2.4ソース・ディレクトリのディレクトリ構成* ディレクトリのパスを削除

Rev.	発行日	改訂内容	
		ページ	ポイント
3.00	2013.09.27	6	*表 2.5 ドライバ関連ディレクトリのファイル構成* hwos ディレクトリを削除
		8	*表 2.7サンプル・アプリケーション・ディレクトリのファイル構成* 不要ファイル削除:Cortex-M3_sample.uer, main.dep, main.dni
		8	*表 2.7サンプル・アプリケーション・ディレクトリのファイル構成* ファイル追加: scat_boot_sflash.ld, boot_serialflash.icf, init.mac
		8	*表 2.7サンプル・アプリケーション・ディレクトリのファイル構成* ファイル名変更: “cortex-M3_sample.mvp(mvr)”->”rin32m3ec.mvp(mvr)”, “scat.ld”->”scat_boot_extrom.ld”, “main.icf”->”boot_norflash.icf”
		9	*表 2.8スタートアップ関連ディレクトリのファイル構成* syscalls.c の説明文修正。
		9	*表 2.8スタートアップ関連ディレクトリのファイル構成* ファイル追加:vectors_rom.c
		10	*図 3.1ファイル相関図* リンク情報ファイルと実行ファイルを拡張子で表記
		17	*3.2.2プログラム配置例* タイトルおよび内容を微修正
		19	*表 4.3定数（システム） システムで使用する UART のチャネル番号を変更*
		20	*表 4.5条件付きコンパイルに使用されるマクロ定義* 定義ファイルの項目を変更
		75	*8.1.2ライブラリ関数の再定義* タイトル修正:「システムコールのリエントラント処理」->「ライブラリ関数の再定義」
		77	*8.2.2ライブラリ関数の再定義* タイトル修正:「システムコールのリエントラント処理」->「ライブラリ関数の再定義」
		78	*図 8.3IAR 使用時のスタートアップ・ルーチン 構成を変更*
		79	*8.3.2ライブラリ関数の再定義* タイトル修正:「システムコールのリエントラント処理」->「ライブラリ関数の再定義」
4.00	2013.12.26	8	*表 2.7サンプル・アプリケーション・ディレクトリのファイル構成* J-Link 設定ファイルを削除
		20	*表 4.5条件付きコンパイルに使用されるマクロ定義* R-IN32M3-CL 用の定義を追加
		41-44	「マクロ」という言葉を「CSI コントローラ」に変更
5.00	2017.02.28	2	*表 1.1 ソフトウェア開発ツール一覧を更新*
		6	*表 2. 5ドライバ関連ディレクトリのファイル構成を追加*
		8	*表 2.7 サンプル・アプリケーション・ディレクトリのファイル構成* ARM のファイル群を MDK-ARM 版の情報に変更（表記変更）
		9	*表 2.8 スタートアップ関連ディレクトリのファイル構成* ARM のファイル群を RealView 用から MDK-ARM 用に変更（表記変更）
		21-22	*5.* R-IN32M3 レジスタ定義（RIN32N3.h）を追加（補足）
		23	*表 6.1 タイマドライバ関数一覧* TAUJ2 の timer_onecount_swtrg_init 関数を削除（削除）
		―	*6.2 タイマ制御* TAUJ2 の timer_onecount_swtrg_init 関数を削除（削除）
		28	*6.2.4 タイマ動作開始* 戻り値 ER_OK の意味を修正（誤記訂正）
		29	*6.2.5 タイマ動作停止* 誤記（開始→停止）を訂正。戻り値 ER_OK の意味を修正（誤記訂正）
		35	*6.4.1 IIC コントローラの初期化* 機能説明を修正し、備考を追加（表記変更）
		56-73	*5.9 章 CAN 制御の追加*

Rev.	発行日	改訂内容	
		ページ	ポイント
5.00	2017.02.28	85	*7.4.3 シリアル・フラッシュ ROM のデータ読み出し* パラメータ説明の誤記（書き込み→読み出し）訂正（誤記訂正）
		86	*7.4.4 シリアル・フラッシュ ROM のデータ消去* パラメータ説明の誤記（書き込み→データ消去）訂正（誤記訂正）
6.00	2018.12.28	2	*1.2 開発環境* 推奨インサーキット・エミュレータ情報の変更
		11,12 15,16	*3.2.1 メモリ配置* 以下のメモリ・マップそれぞれに対し、Instruction RAM mirror area(768K バイト)はブート・モードによりアクセス発生アドレスが変化する注意を追加 図 3.2メモリ・マップ（全体）　（R-IN32M3-EC） 図 3.3メモリ・マップ（全体）　（R-IN32M3-CL） 図 3.7外部マイコン・インタフェース空間　（R-IN32M3-EC） 図 3.8外部マイコン・インタフェース空間　（R-IN32M3-CL）
		11,12	*3.2.1 メモリ配置* 以下のメモリ・マップに対し、Instruction RAM area と Instruction RAM mirror area の入れ替わりを修正 図 3.2メモリ・マップ（全体）　（R-IN32M3-EC） 図 3.3メモリ・マップ（全体）　（R-IN32M3-CL）
		15,16	*3.2.1 メモリ配置* 以下のメモリ・マップの、Instruction RAM area を Instruction RAM mirror area へ修正 図 3.7外部マイコン・インタフェース空間　（R-IN32M3-EC） 図 3.8外部マイコン・インタフェース空間　（R-IN32M3-CL）
		35	*6.4.1 IIC コントローラの初期化* 機能説明のタイミング設定値を修正し、備考に PCLK の説明を追加
		46	「6.5.5 受信データ確認（スレーブ用）」 「（4）機能」説明文中の誤記を修正
		―	誤記訂正、表現訂正、他文書との記載内容統一
6.01	2019.04.19	2	*1.2 開発環境* 表記更新

[メ　モ]

R-IN32M3シリーズ プログラミング・マニュアル
ドライバ編

発行年月日　2013年03月29日　Rev.1.00
　　2019年04月19日　Rev.6.01

発行　ルネサス エレクトロニクス株式会社
　　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

R-IN32M3 シリーズ
プログラミング・マニュアル（ドライバ編）

ルネサス エレクトロニクス株式会社

R18UZ0008JJ0601

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/

Colophon 3.1