

要旨

本アプリケーションノートでは、RZ/T1 の R-IN Engine 搭載製品 初期設定を行うサンプルプログラムについて説明します。

R-IN Engine 搭載製品 初期設定サンプルプログラムの特長を以下に示します。

- Cortex-R4 コアはリセット解除後に初期設定を行い、Cortex-M3 コアのリセット解除処理を行います。その後 CPU 間割り込みを発生させ、一定周期で LED0 の点滅処理を行います。
- Cortex-M3 コアはリセット解除後に初期設定を行い、Cortex-R4 からの CPU 間割り込み要求を待ちます。CPU 間割り込み要求を受け付けた後は、LED1 を点灯させます。
- スイッチ SW2 を押下すると、Cortex-R4 コアは外部端子割り込み処理により、LED0 の点灯／消灯の状態を LED データとして共有メモリ領域に書き込みます。一方 Cortex-M3 コアは常に共有メモリの LED データを読み込み、点灯／消灯状態を LED1 に反映させます。

対象デバイス

RZ/T1 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あわせて変更し、十分評価してください。

目次

1.	仕様	4
2.	動作環境	5
3.	関連アプリケーションノート	6
4.	周辺機能説明	7
5.	ハードウェア説明	8
5.1	ハードウェア構成例	8
5.2	使用端子一覧	8
6.	ソフトウェア説明	9
6.1	動作概要	9
6.1.1	使用準備	10
6.2	メモリマップ	10
6.2.1	サンプルプログラムのセクション配置	10
6.2.2	MPU の設定	13
6.2.3	例外処理ベクタテーブル	13
6.3	使用割り込み一覧	14
6.4	固定幅整数一覧	14
6.5	定数／エラーコード一覧	15
6.6	構造体／共用体／列挙型一覧	15
6.7	大域変数一覧	16
6.8	関数一覧	16
6.9	関数仕様	17
6.9.1	R_SHM_Init	17
6.9.2	R_SHM_memcpy	17
6.9.3	R_SHM_Load_uint32	17
6.9.4	R_SHM_Load_int32	18
6.9.5	R_SHM_Load_uint16	18
6.9.6	R_SHM_Load_int16	18
6.9.7	R_SHM_Load_uint8	19
6.9.8	R_SHM_Load_int8	19
6.9.9	main (Cortex-R4)	19
6.9.10	init_cm3 (Cortex-R4)	20
6.9.11	main (Cortex-M3)	20
6.9.12	R_IRQ9_isr (Cortex-R4)	20
6.9.13	IRQ_INTERCPU_IRQHandler	21
6.10	フローチャート	22
6.10.1	共用メモリドライバ初期化処理	22
6.10.2	共用メモリ領域用メモリコピー処理	23
6.10.3	共用メモリ領域用 uint 4 バイトデータロード処理	24
6.10.4	共用メモリ領域用 int 4 バイトデータロード処理	24
6.10.5	共用メモリ領域用 uint 2 バイトデータロード処理	24

6.10.6	共用メモリ領域用 int 2 バイトデータロード処理.....	24
6.10.7	共用メモリ領域用 uint 1 バイトデータロード処理.....	24
6.10.8	共用メモリ領域用 int 1 バイトデータロード処理.....	24
6.10.9	メイン処理 (Cortex-R4).....	25
6.10.10	Cortex-M3 コア初期化処理.....	26
6.10.11	メイン処理 (Cortex-M3).....	27
6.10.12	R_IRQ9 割り込み (IRQ 端子割り込み 5) 処理.....	28
6.10.13	CPU 間割り込み処理.....	28
7.	サンプルプログラム.....	29
8.	参考ドキュメント.....	30
付録 1.	各開発環境における補足内容.....	31

1. 仕様

表 1.1 に使用する周辺機能と用途を、図 1.1 に動作環境を示します。

表 1.1 使用する周辺機能と用途

周辺機能	用途
クロック発生回路 (CPG)	CPUクロックおよび低速オンチップオシレータで使用
割り込みコントローラ (ICUA)	外部割り込み入力端子 (IRQ5)、CPU間割り込みで使用
拡張内蔵RAM	共用メモリ領域 (拡張内蔵 Instruction RAM (以降、RAM I と略す) および Cortex-M3用プログラム領域メモリ (拡張内蔵Data RAM (以降、RAM D と略す)) で使用
エラーコントロールモジュール (ECM)	ERROROUT#端子の初期化
汎用入出力ポート	LEDの点灯および消灯のための端子制御に使用

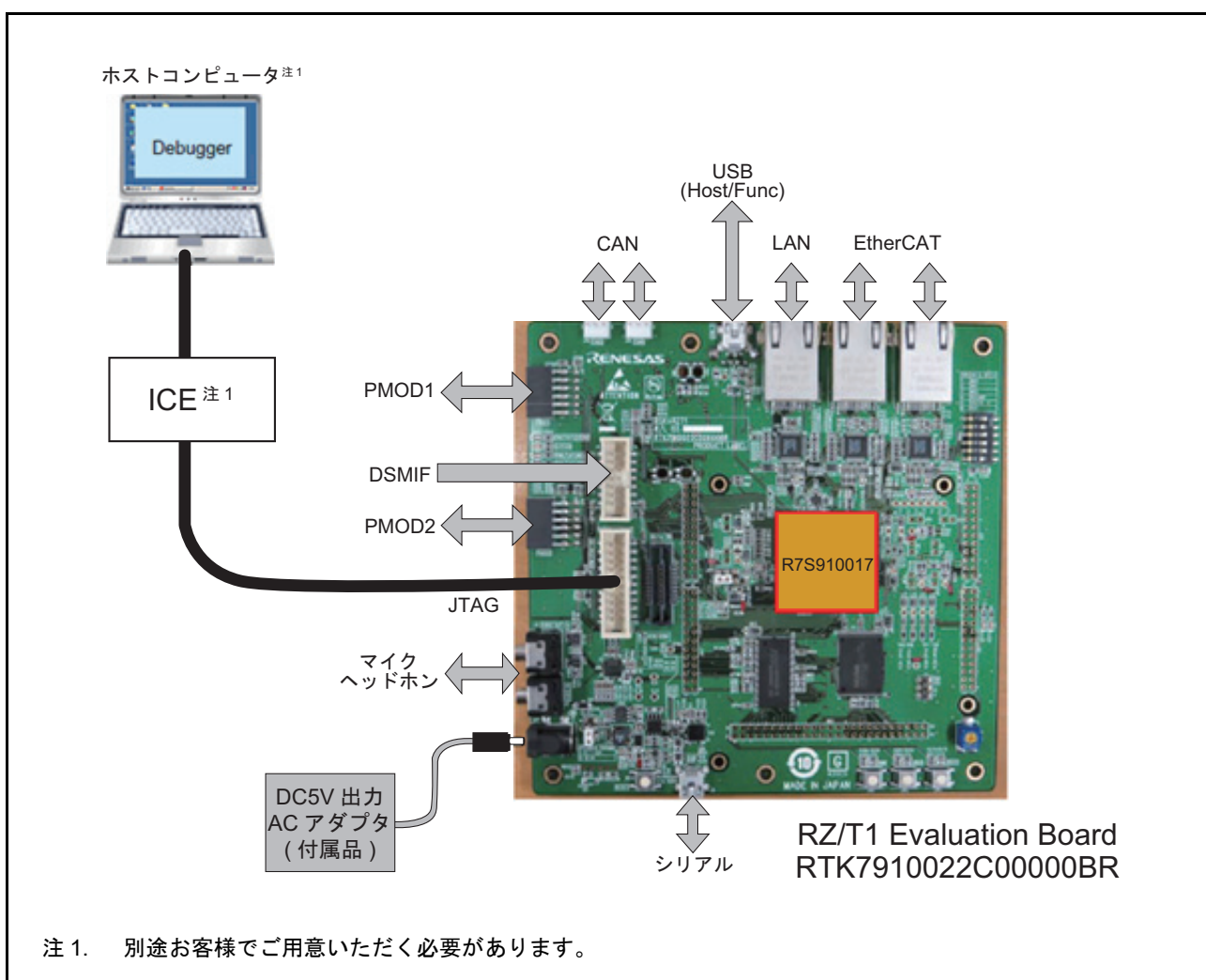


図 1.1 動作環境

2. 動作環境

本アプリケーションノートのサンプルプログラムは、以下の環境を想定しています。

表2.1 動作環境

項目	内容
使用マイコン	RZ/T1グループ
動作周波数	CPUCLK = 450MHz (Cortex-R4)、 ICLK = 150MHz (Cortex-M3)
動作電圧	3.3V
統合開発環境	IARシステムズ製 Embedded Workbench® for Arm Version 8.20.2 Arm製 DS-5™ 5.26.2 RENESAS製 e2studio 6.1.0
動作モード	SPIブートモード 16ビットバスブートモード
使用ボード	RZ/T1 Evaluation Board (RTK7910022C00000BR)
使用デバイス (ボード上で使用する機能)	<ul style="list-style-type: none">• NORフラッシュメモリ (CS0、CS1空間に接続) メーカー名: Macronix International Co...、型名: MX29GL512FLT2I-10Q• SDRAM (CS2、CS3空間に接続) メーカー名: Integrated Silicon Solution Inc、型名: IS42S16320D-7TL• シリアルフラッシュメモリ メーカー名: Macronix International Co...、型名: MX25L51245G

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

- RZ/T1 グループ初期設定

4. 周辺機能説明

クロック発生回路 (CPG)、割り込みコントローラ (ICUA)、エラーコントロールモジュール (ECM)、拡張内蔵 RAM、汎用入出力ポートについての基本的な内容は、RZ/T1 グループ・ユーザーズマニュアルハードウェア編を参照してください。

5. ハードウェア説明

5.1 ハードウェア構成例

図 5.1 にハードウェア構成例を示します。

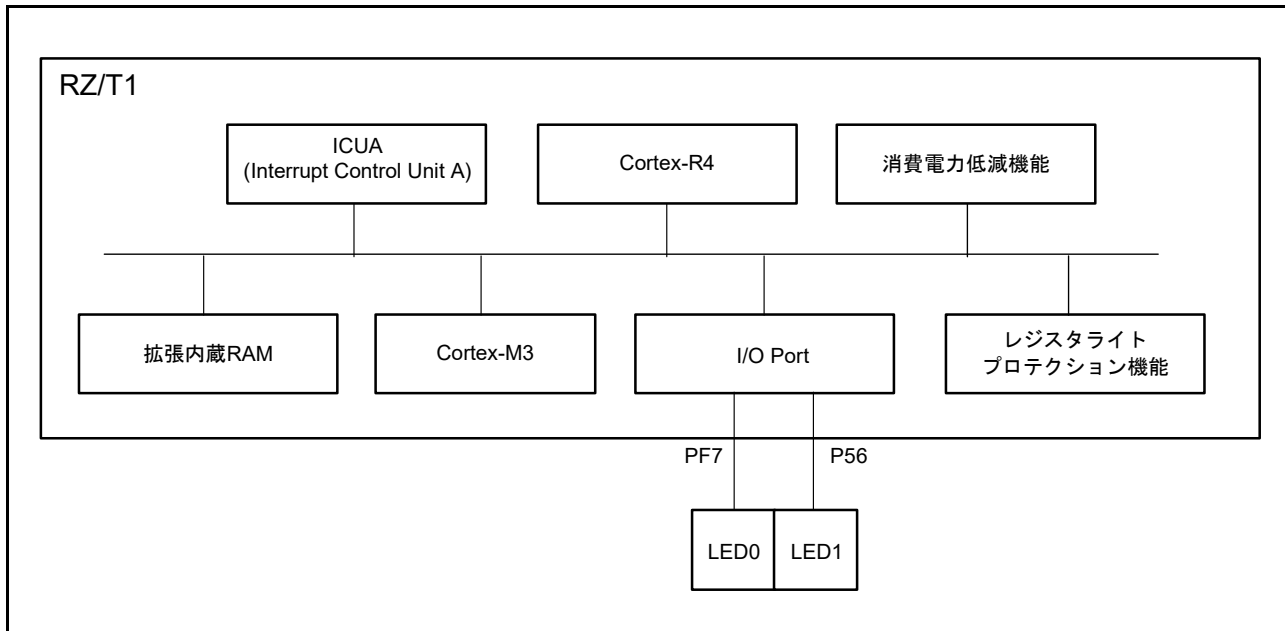


図 5.1 ハードウェア構成例

5.2 使用端子一覧

表 5.1 に使用端子と機能を示します。

表 5.1 使用端子と機能

端子名	入出力	内容
MD0	入力	動作モードの選択 MD0="L"、MD1="L"、MD2="L" (SPIブートモード) MD0="L"、MD1="H"、MD2="L" (16ビットバスブートモード)
MD1	入力	
MD2	入力	
IRQ5	入力	SW2 (IRQ端子割り込み)
PF7	出力	LED0の点灯および消灯
P56	出力	LED1の点灯および消灯

6. ソフトウェア説明

以降、特に明記しない場合は EWARM (IAR 社製) を使用した場合について説明を行います。

6.1 動作概要

本サンプルプログラムでは、Cortex-R4 コアと Cortex-M3 コアによる初期設定を行います。

- ① Cortex-R4 コア側のプログラムは、電源投入後に初期設定を行い Cortex-M3 コアのプログラムコードを外付けフラッシュメモリから拡張内蔵 RAM へコピーし (RAM 実行版はコピー処理をしません)、Cortex-M3 コアのリセット解除を行います。その後、LED 制御用のポート設定や割り込みを設定し、CPU 間割り込みを発生させて Cortex-M3 側へ初期化の完了を通知します。
- ② Cortex-M3 コア側のプログラムは、ソフトウェアリセット 2 解除後に初期設定を行い Cortex-R4 コアからの CPU 間割り込みを待ちます。
- ③ Cortex-R4 コア側は、LED0 を一定周期で点滅させます。また、スイッチ SW2 の押下を検出すると、共用メモリ領域 (拡張内蔵 RAM) に LED データとして LED0 の点灯状態 (点灯/消灯) を書き込みます。一方、Cortex-M3 コア側は常時共用メモリの LED データの値を読み込み、その値に応じて LED1 の点灯表示を行います (初期値は点灯)。例えば LED0 が消灯時に SW2 を押下すると、LED1 にも点灯状態が反映され消灯状態になります。

図 6.1 に R-IN Engine 搭載製品 初期設定処理の概要動作を示します。

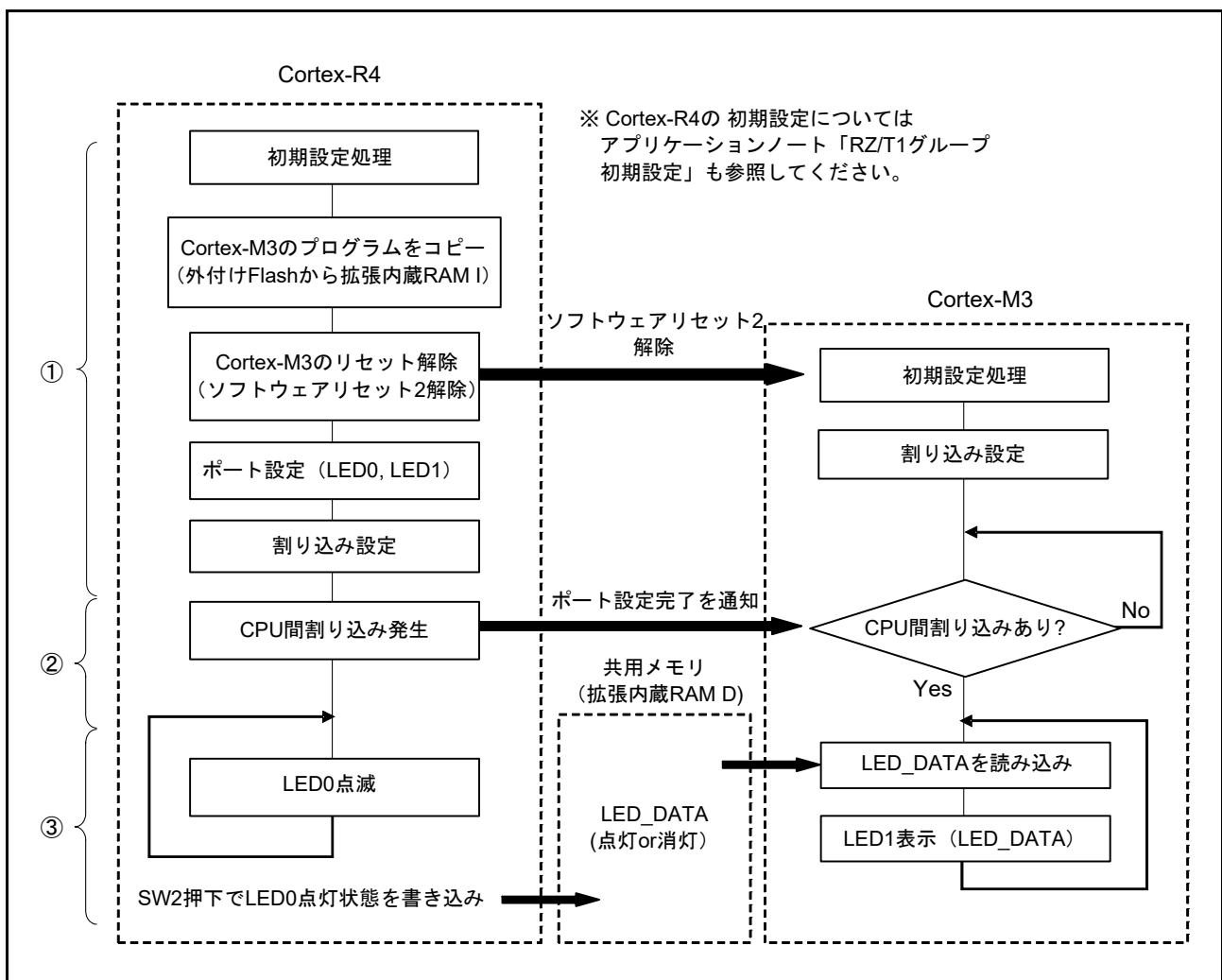


図 6.1 R-IN Engine 搭載製品 初期設定処理の概要動作

6.1.1 使用準備

使用するプロジェクトによって、RZ/T1 Evaluation Board (RTK7910022C00000BR) 上にある SW4 の設定が異なります。表 6.1 に SW4 の設定を示します。SW4 の各設定につきましては、RZ/T1 Evaluation Board RTK7910022C00000BR ユーザーズマニュアルに記載しています。詳細は、8. 参考ドキュメントを参照してください。

表 6.1 SW4 の設定

サンプルプログラム	SW4-1	SW4-2	SW4-3	SW4-4	SW4-5	SW4-6
16ビットバスブートモード版	ON	OFF	ON	ON	ON	OFF
SPIブートモード版	ON	ON	ON	ON	ON	OFF
RAM実行版	上記いずれかの SW4 設定					

6.2 メモリマップ

RZ/T1 グループのアドレス空間と RZ/T1 評価ボードのメモリマッピングについては、アプリケーションノート「RZ/T1 グループ初期設定」およびユーザーズマニュアル ハードウェア編に記載しています。

6.2.1 サンプルプログラムのセクション配置

表 6.2 に使用するセクション、図 6.2 にセクション配置 (16 ビットバスブート版の例) を示します。

Cortex-R4 のセクションの詳細については、アプリケーションノート「RZ/T1 グループ初期設定」に記載しています。

表 6.2 使用するセクション (Cortex-R4)

領域の名前	内容	タイプ	ロード領域	実行領域
VECTOR_WBLOCK	リセット、例外ベクタテーブル	Code	—	ATCM
USER_PRG_WBLOCK	ユーザアプリケーションプログラム領域 (実行用)	Code	—	ATCM
USER_DATA_WBLOCK	ユーザアプリケーションプログラム用変数領域 (実行用)	Data	—	ATCM
CSTACK	スタック領域	Data	—	ATCM
SVC_STACK	スーパーバイザ (SVC) モードのスタック領域	Data	—	ATCM
IRQ_STACK	IRQモードのスタック領域	Data	—	ATCM
FIQ_STACK	FIQモードのスタック領域	Data	—	ATCM
UND_STACK	未定義 (UND) モードのスタック領域	Data	—	ATCM
ABT_STACK	アポート (ABT) モードのスタック領域	Data	—	ATCM
LDR_DATA_WBLOCK	ローダプログラム用変数領域 (実行用)	Data	—	BTCM
LDR_PRG_WBLOCK	ローダプログラム領域 (実行用)	Code	—	BTCM
ldr_param	ローダ用パラメータ注1	Data	FLASH注2	—
LDR_PRG_RBLOCK	ローダプログラム領域 (格納用) 注1	Code	FLASH注2	—
LDR_DATA_RBLOCK	ローダプログラム用変数領域 (格納用) 注1	Data	FLASH注2	—
VECTOR_RBLOCK	リセット、例外ベクタテーブル領域 (格納用) 注1	Code	FLASH注2	—
USER_PRG_RBLOCK	ユーザアプリケーションプログラム領域 (格納用) 注1、 Cortex-M3用ユーザアプリケーションプログラム領域 (格納用) 注3	Code	FLASH注2	—
USER_DATA_RBLOCK	ユーザアプリケーションプログラム用変数領域 (格納用) 注1	Data	FLASH注2	—

注1. RAM実行版では、存在しません。

注2. 16ビットバスブートモード版では、NORフラッシュメモリ、SPIブートモード版では、シリアル・フラッシュメモリとなります。

注3. RAM実行版では存在しません。代わりにCortex-M3のセクション配置に基づいて開発ツールで直接、実行領域である拡張内蔵RAM I にダウンロードされます。

表 6.3 使用するセクション (Cortex-M3)

領域の名前	内容	タイプ	ロード領域	実行領域
vectors	ベクタ領域	Code	拡張内蔵 RAM I	拡張内蔵 RAM I
readonly	ユーザアプリケーションプログラム領域	Code	拡張内蔵 RAM I	拡張内蔵 RAM I
_SHARED_MEM	共用メモリ領域	Data	拡張内蔵 RAM D	拡張内蔵 RAM D
readwrite	ユーザアプリケーションプログラム用変数領域	Data	拡張内蔵 RAM D	拡張内蔵 RAM D
HEAP	ヒープ領域	Data	拡張内蔵 RAM D	拡張内蔵 RAM D
CSTACK	スタック領域	Data	拡張内蔵 RAM D	拡張内蔵 RAM D

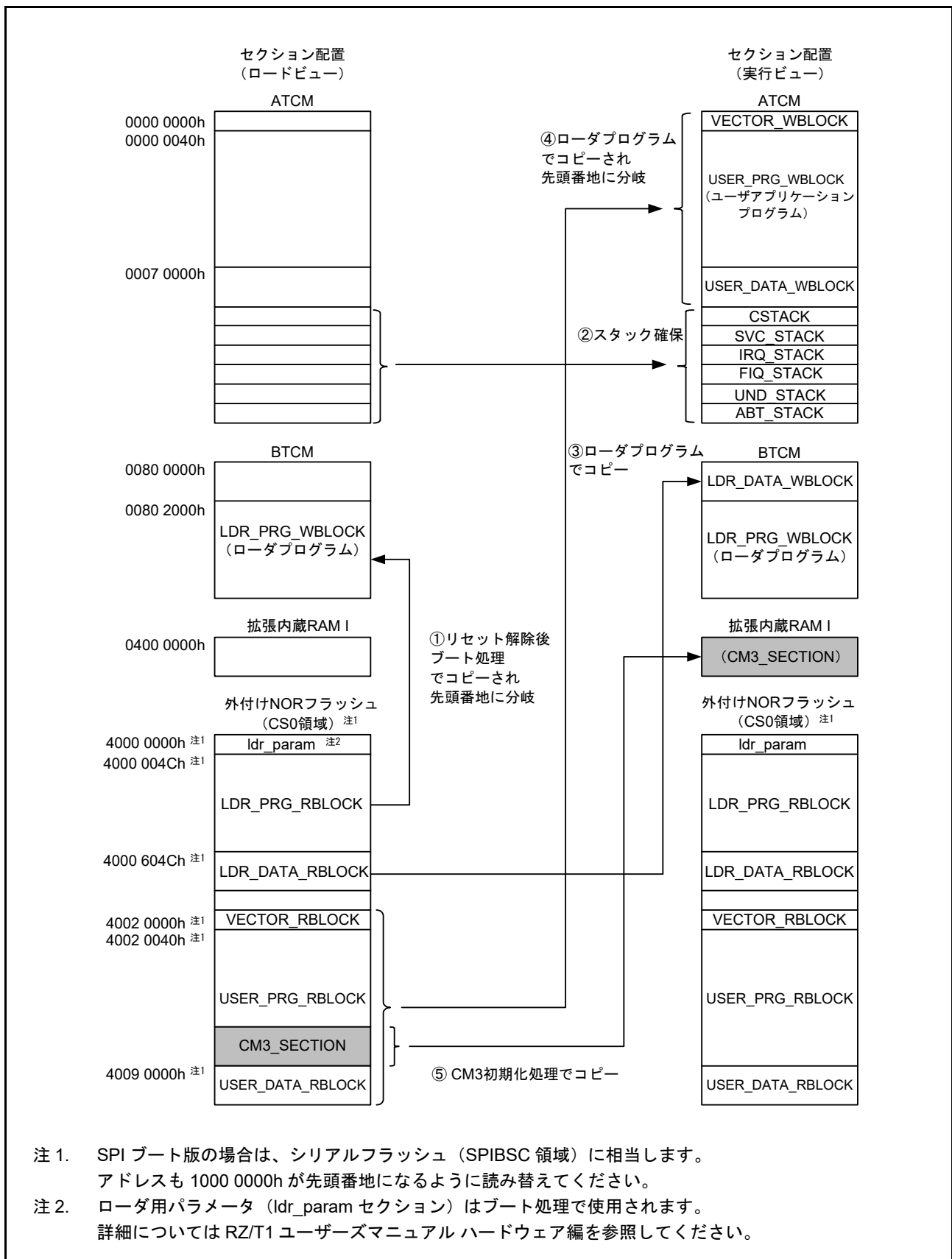


図 6.2 セクション配置 (Cortex-R4, 16 ビットバスブートの例)

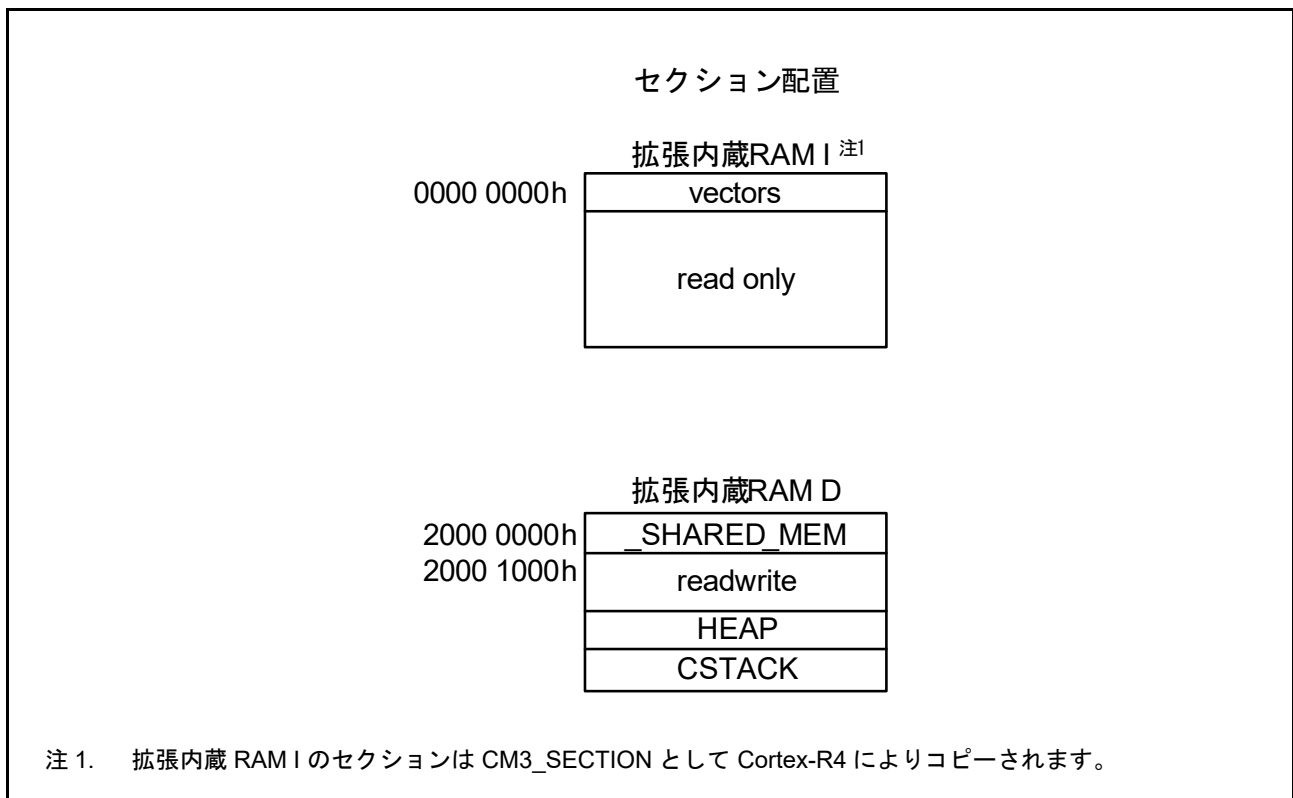


図 6.3 セクション配置 (Cortex-M3)

6.2.2 MPU の設定

MPU の設定は、アプリケーションノート「RZ/T1 グループ初期設定」に記載しています。

6.2.3 例外処理ベクタテーブル

例外処理のベクタテーブルは、アプリケーションノート「RZ/T1 グループ初期設定」に記載しています。

6.3 使用割り込み一覧

表 6.4 にサンプルプログラムで使用する割り込みを示します。

表6.4 サンプルプログラムで使用する割り込み

割り込み (Source ID)	優先度	処理概要
CPU間割り込み (IRQ1)	15	Cortex-R4からCortex-M3へ初期化設定の完了を通知します (Cortex-R4、Cortex-M3)。
IRQ 端子割り込み5 (IRQ9)	15	SW2が押されるとLED0の点灯状態 (点灯/消灯) が共用メモリに書き込まれます (Cortex-R4コアのみ)。

6.4 固定幅整数一覧

表 6.5 にサンプルプログラムで使用する固定幅整数を示します。

表6.5 サンプルプログラムで使用する固定幅整数

シンボル	内容
int8_t	8ビット整数、符号あり (標準ライブラリにて定義)
int16_t	16ビット整数、符号あり (標準ライブラリにて定義)
int32_t	32ビット整数、符号あり (標準ライブラリにて定義)
int64_t	64ビット整数、符号あり (標準ライブラリにて定義)
uint8_t	8ビット整数、符号なし (標準ライブラリにて定義)
uint16_t	16ビット整数、符号なし (標準ライブラリにて定義)
uint32_t	32ビット整数、符号なし (標準ライブラリにて定義)
uint64_t	64ビット整数、符号なし (標準ライブラリにて定義)

6.5 定数／エラーコード一覧

表 6.6 にサンプルプログラムで使用する定数を示します。Cortex-R4 コアのプログラムの詳細についてはアプリケーションノート「RZ/T1 グループ初期設定」に記載しています。

表 6.6 サンプルプログラムで使用する定数

定数名	設定値	内容
LED_OUTPUT_HIGH	(1)	LED点灯表示用の出力データ
SHM_SUCCESS	(0)	共用メモリドライバの正常制御フラグ
SHM_ERR	(-1)	共用メモリドライバの異常制御フラグ
SHM_SEMFNO_TOTAL	(8)	共用メモリドライバ用セマフォフラグの総数
SHM_SEMFNO_0	(0)	共用メモリドライバ用セマフォフラグ0
SHM_SEMFNO_1	(1)	共用メモリドライバ用セマフォフラグ1
SHM_SEMFNO_2	(2)	共用メモリドライバ用セマフォフラグ2
SHM_SEMFNO_3	(3)	共用メモリドライバ用セマフォフラグ3
SHM_SEMFNO_4	(4)	共用メモリドライバ用セマフォフラグ4
SHM_SEMFNO_5	(5)	共用メモリドライバ用セマフォフラグ5
SHM_SEMFNO_6	(6)	共用メモリドライバ用セマフォフラグ6
SHM_SEMFNO_7	(7)	共用メモリドライバ用セマフォフラグ7

6.6 構造体／共用体／列挙型一覧

図 6.4 にサンプルプログラムで使用する構造体／共用体／列挙型を示します。Cortex-R4 コアのプログラムの詳細についてはアプリケーションノート「RZ/T1 グループ初期設定」に記載しています。

```

/* Shared memory struct */
/* Size MAX 4KB */
struct st_shm
{
    uint32_t LED_DATA;
};

/* struct of [SYSTEM.SYTSEMFn] register */
typedef struct
{
    union
    {
        unsigned long LONG;
        struct
        {
            unsigned long SEMF:1;
            unsigned long :31;
        } BIT;
    } SYTSEMF;
} st_sytsemf;

```

図 6.4 サンプルプログラムで使用する構造体／共用体／列挙型

6.7 大域変数一覧

表 6.7 に大域変数を示します。Cortex-R4 コアのプログラムの詳細についてはアプリケーションノート「RZ/T1 グループ初期設定」に記載しています。

表 6.7 大域変数一覧

型	変数名	内容	使用関数
uint8_t	g_ready_flag	Cortex-R4初期化完了フラグ	init_main.c (Cortex-M3)
uint32_t	g_led_data	LED表示用データ変数	main.c (Cortex-R4) init_main.c (Cortex-M3)

6.8 関数一覧

表 6.8 に関数を示します。Cortex-R4 コアのプログラムの詳細についてはアプリケーションノート「RZ/T1 グループ初期設定」に記載しています。

表 6.8 関数一覧

関数名	ページ番号
R_SHM_Init	17
R_SHM_memcpy	17
R_SHM_Load_uint32	17
R_SHM_Load_int32	18
R_SHM_Load_uint16	18
R_SHM_Load_int16	18
R_SHM_Load_uint8	19
R_SHM_Load_int8	19
main (Cortex-R4)	19
init_cm3 (Cortex-R4)	20
R_IRQ9_isr (Cortex-R4)	20
main (Cortex-M3)	20
IRQ_INTERCPU_IRQHandler (Cortex-M3)	21

注. 特に記載がない関数は各コアで共通に使用します。

6.9 関数仕様

サンプルプログラムの関数仕様を示します。

6.9.1 R_SHM_Init

R_SHM_Init	
概要	共用メモリドライバ初期化処理
宣言	int32_t R_SHM_Init (uint32_t semfno)
説明	共用メモリドライバを使用するために指定したセマフォレジスタの初期設定を行います。
引数	uint32_t semfno 初期化するセマフォレジスタ番号を指定します。 設定可能範囲 (0 ~ 7)
リターン値	SHM_SUCCESS : 成功 SHM_ERR : 失敗
補足	なし

6.9.2 R_SHM_memcpy

R_SHM_memcpy	
概要	共用メモリ領域用メモリコピー処理
宣言	int32_t R_SHM_memcpy(void *dst, void *src, size_t size)
説明	共用メモリ領域に対する多バイトデータのコピー処理を行います。
引数	void *dst 転送先メモリ領域のポインタアドレス void *src 転送元メモリ領域のポインタアドレス size_t size 転送メモリのサイズ
リターン値	SHM_SUCCESS : 成功 SHM_ERR : 失敗
補足	なし

6.9.3 R_SHM_Load_uint32

R_SHM_Load_uint32	
概要	共用メモリ領域用 uint 4 バイトデータロード処理
宣言	int32_t R_SHM_Load_uint32(uint32_t *dst, uint32_t *src)
説明	共用メモリ領域に対する uint 4 バイトデータのロード処理を行います。
引数	uint32_t *dst 転送先メモリ領域のポインタアドレス uint32_t *src 転送元メモリ領域のポインタアドレス
リターン値	なし
補足	なし

6.9.4 R_SHM_Load_int32

R_SHM_Load_int32

概要	共用メモリ領域用 int 4 バイトデータロード処理	
宣言	int32_t R_SHM_Load_int32(int32_t *dst, int32_t *src)	
説明	共用メモリ領域に対する int 4 バイトデータのロード処理を行います。	
引数	int32_t *dst	転送先メモリ領域のポインタアドレス
	int32_t *src	転送元メモリ領域のポインタアドレス
リターン値	なし	
補足	なし	

6.9.5 R_SHM_Load_uint16

R_SHM_Load_uint16

概要	共用メモリ領域用 uint 2 バイトデータロード処理	
宣言	int32_t R_SHM_Load_uint16(uint16_t *dst, uint16_t *src)	
説明	共用メモリ領域に対する uint 2 バイトデータのロード処理を行います。	
引数	uint16_t *dst	転送先メモリ領域のポインタアドレス
	uint16_t *src	転送元メモリ領域のポインタアドレス
リターン値	なし	
補足	なし	

6.9.6 R_SHM_Load_int16

R_SHM_Load_int16

概要	共用メモリ領域用 int 2 バイトデータロード処理	
宣言	int32_t R_SHM_Load_int16(int16_t *dst, int16_t *src)	
説明	共用メモリ領域に対する int 2 バイトデータのロード処理を行います。	
引数	int16_t *dst	転送先メモリ領域のポインタアドレス
	int16_t *src	転送元メモリ領域のポインタアドレス
リターン値	なし	
補足	なし	

6.9.7 R_SHM_Load_uint8

R_SHM_Load_uint8

概要	共用メモリ領域用 uint 1 バイトデータロード処理	
宣言	int32_t R_SHM_Load_uint8(uint8_t *dst, uint8_t *src)	
説明	共用メモリ領域に対する uint 1 バイトデータのロード処理を行います。	
引数	uint8_t *dst	転送先メモリ領域のポインタアドレス
	uint8_t *src	転送元メモリ領域のポインタアドレス
リターン値	なし	
補足	なし	

6.9.8 R_SHM_Load_int8

R_SHM_Load_int8

概要	共用メモリ領域用 int 1 バイトデータロード処理	
宣言	int32_t R_SHM_Load_int8(int8_t *dst, int8_t *src)	
説明	共用メモリ領域に対する int 1 バイトデータのロード処理を行います。	
引数	int8_t *dst	転送先メモリ領域のポインタアドレス
	int8_t *src	転送元メモリ領域のポインタアドレス
リターン値	なし	
補足	なし	

6.9.9 main (Cortex-R4)

main

概要	メイン処理
宣言	int main (void)
説明	Cortex-R4 コア用ユーザアプリケーションプログラムです。
引数	なし
リターン値	なし
補足	なし

6.9.10 init_cm3 (Cortex-R4)

init_cm3

概要	Cortex-M3 コア初期化処理
宣言	void init_cm3(void)
説明	Cortex-M3 用プログラムコードを外付け Flash から拡張内蔵 RAM へコピーし、Cortex-M3 のリセット解除を行います。
引数	なし
リターン値	なし
補足	RAM 実行版では、Cortex-M3 のリセット解除のみを行い、プログラムのコピーは行いません。

6.9.11 main (Cortex-M3)

main

概要	メイン処理
宣言	int main(void)
説明	Cortex-M3 コア用ユーザアプリケーションプログラムです。
引数	なし
リターン値	なし
補足	なし

6.9.12 R_IRQ9_isr (Cortex-R4)

R_IRQ9_isr

概要	IRQ9 割り込み (IRQ 端子割り込み 5) 処理
宣言	void R_IRQ9_isr(void)
説明	LED0 の点灯状態を共有メモリに書き込みます。
引数	なし
リターン値	なし
補足	なし

6.9.13 IRQ_INTERCPU_IRQHandler

IRQ_INTERCPU_IRQHandler

概 要	CPU 間割り込み処理
宣 言	void IRQ_INTERCPU_IRQHandler(void)
説 明	Cortex-R4 コアからの割り込み要求通知による割り込み処理です。g_ready_flag をセットします。
引 数	なし
リターン値	なし
補 足	なし

6.10 フローチャート

6.10.1 共用メモリドライバ初期化処理

図 6.5 に共用メモリドライバ初期化処理のフローチャートを示します。

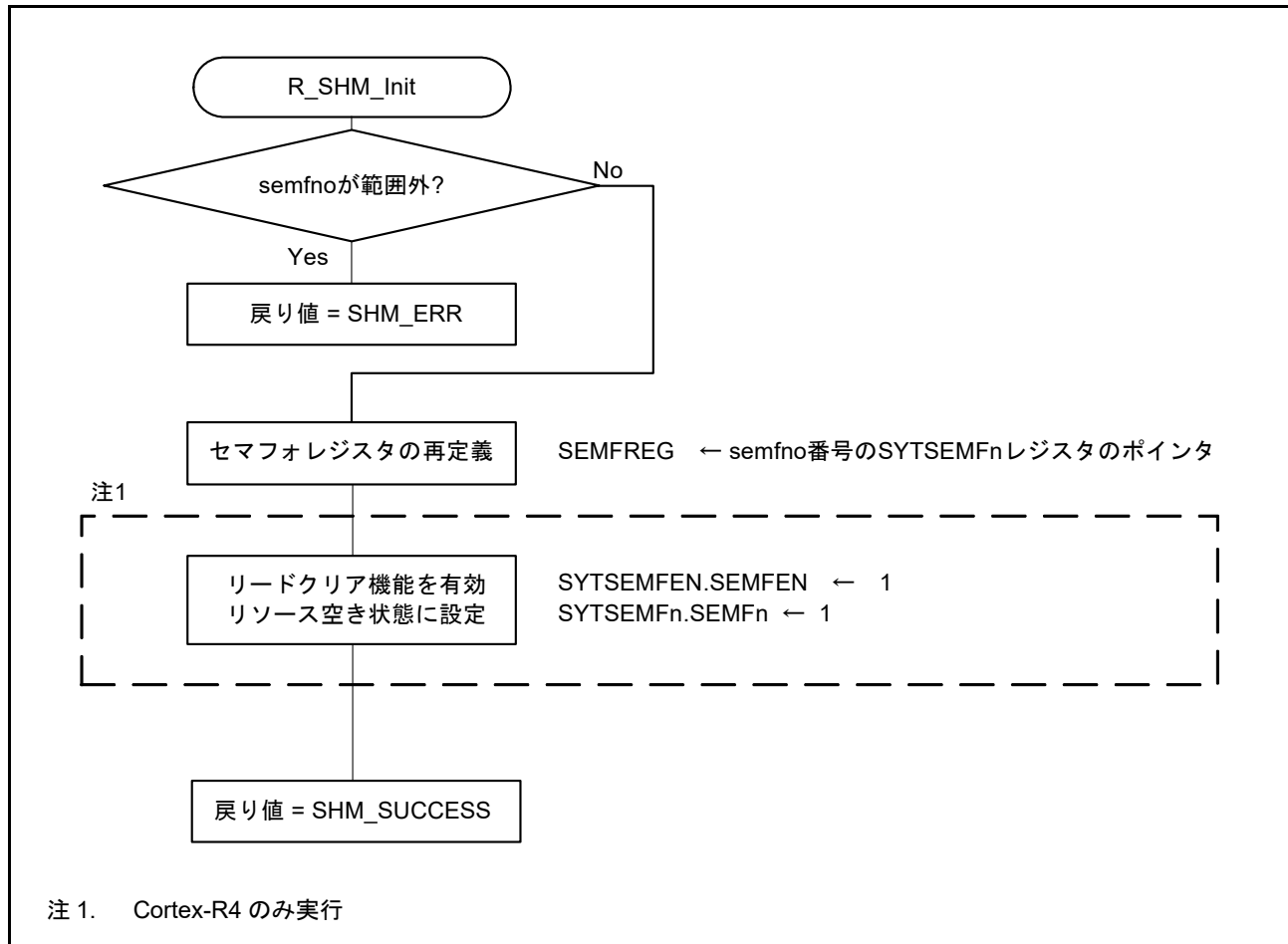


図 6.5 共用メモリドライバ初期化処理

6.10.2 共用メモリ領域用メモリコピー処理

図 6.6 に共用メモリ領域用メモリコピー処理のフローチャートを示します。

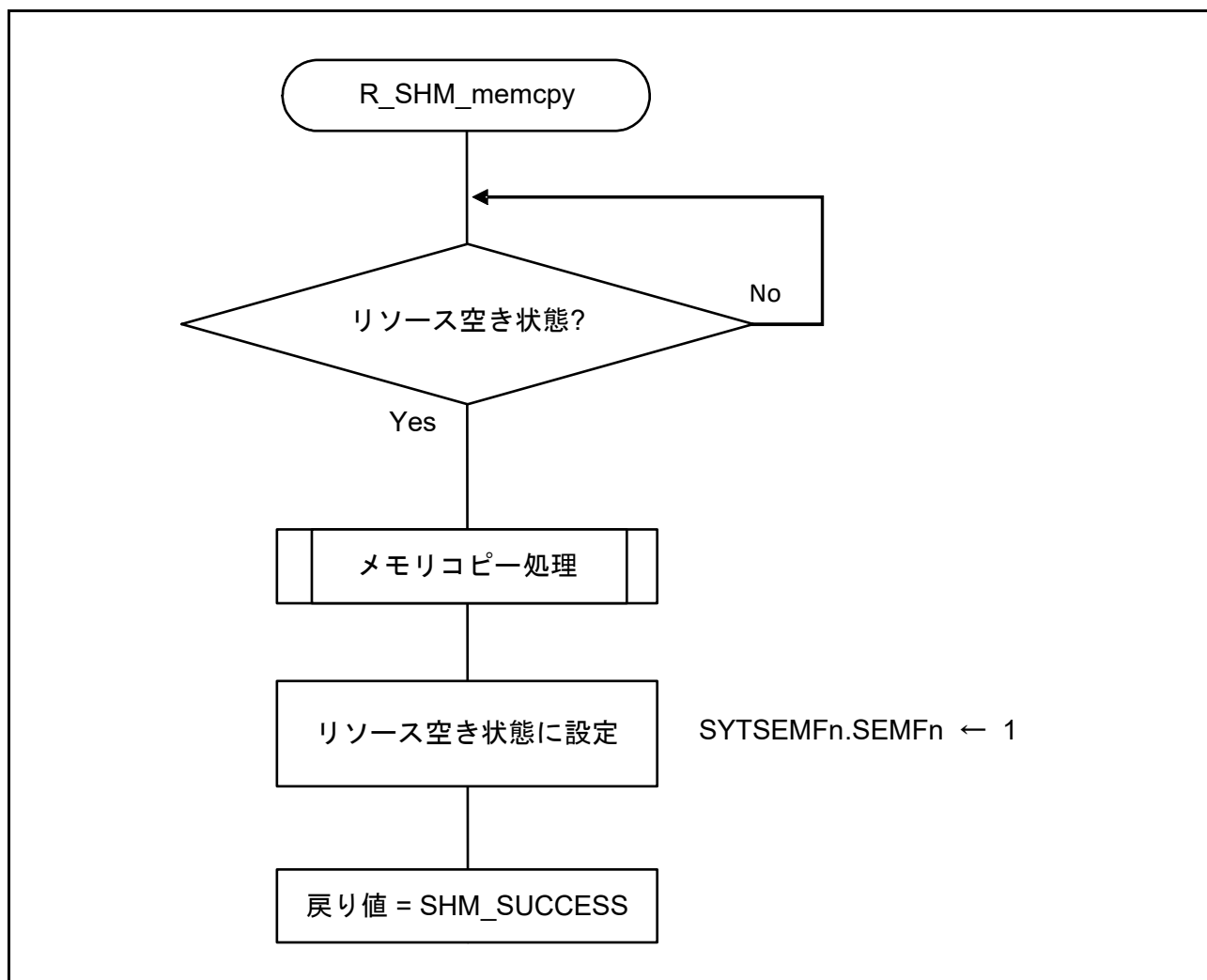


図 6.6 共用メモリ領域用メモリコピー処理

6.10.3 共用メモリ領域用 uint 4 バイトデータロード処理

図 6.7 に共用メモリ領域用 uint 4 バイトデータロード処理のフローチャートを示します。

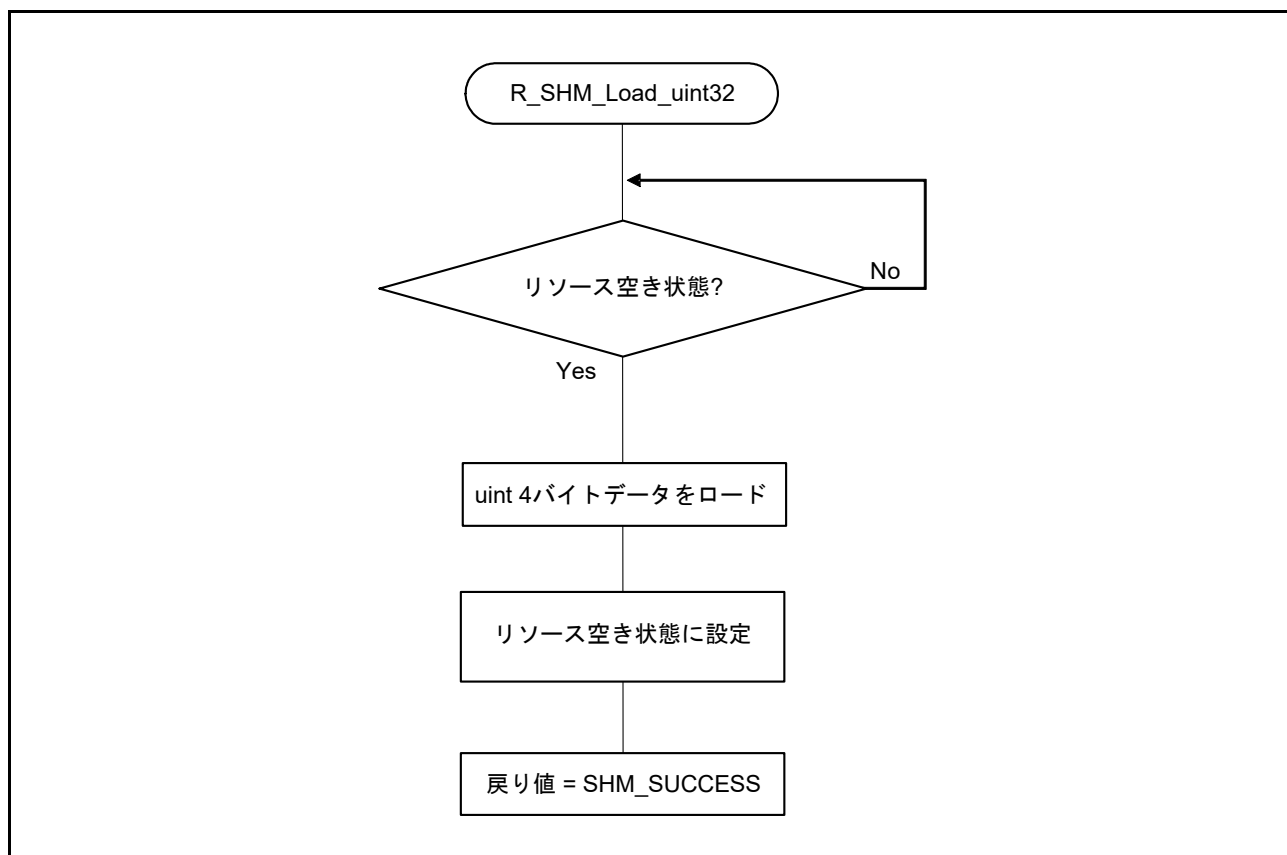


図 6.7 共用メモリ領域用 uint 4 バイトデータロード処理

6.10.4 共用メモリ領域用 int 4 バイトデータロード処理

フローチャートは図 6.7 を参照してください。但し、ロードするデータは int 4 バイトです

6.10.5 共用メモリ領域用 uint 2 バイトデータロード処理

フローチャートは図 6.7 を参照してください。但し、ロードするデータは uint 2 バイトです

6.10.6 共用メモリ領域用 int 2 バイトデータロード処理

フローチャートは図 6.7 を参照してください。但し、ロードするデータは int 2 バイトです

6.10.7 共用メモリ領域用 uint 1 バイトデータロード処理

フローチャートは図 6.7 を参照してください。但し、ロードするデータは uint 1 バイトです

6.10.8 共用メモリ領域用 int 1 バイトデータロード処理

フローチャートは図 6.7 を参照してください。但し、ロードするデータは int 1 バイトです

6.10.9 メイン処理 (Cortex-R4)

図 6.8 にメイン処理 (Cortex-R4) のフローチャートを示します。

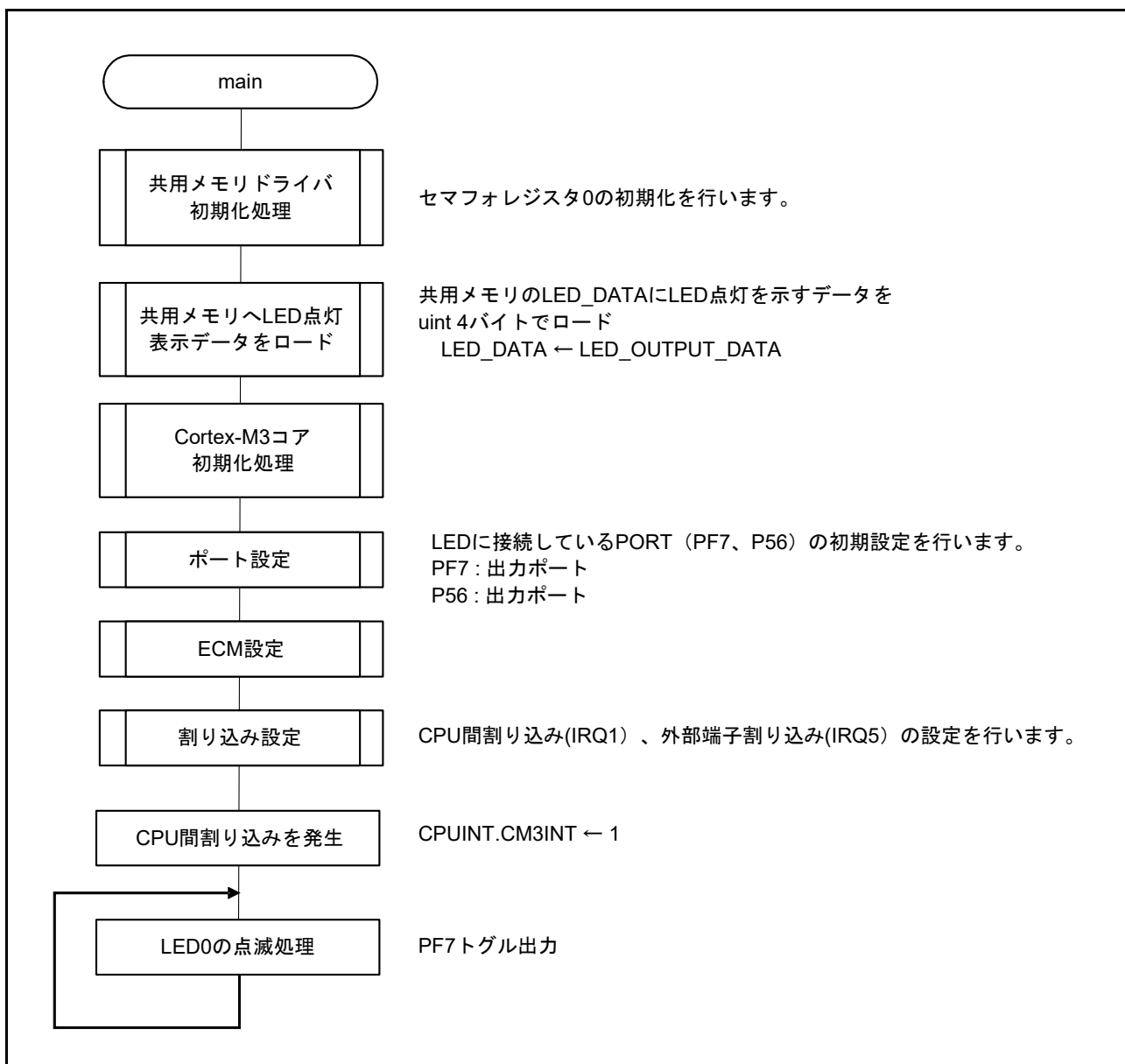


図 6.8 メイン処理 (Cortex-R4)

6.10.10 Cortex-M3 コア初期化処理

図 6.9 に Cortex-M3 コア初期化処理のフローチャートを示します。

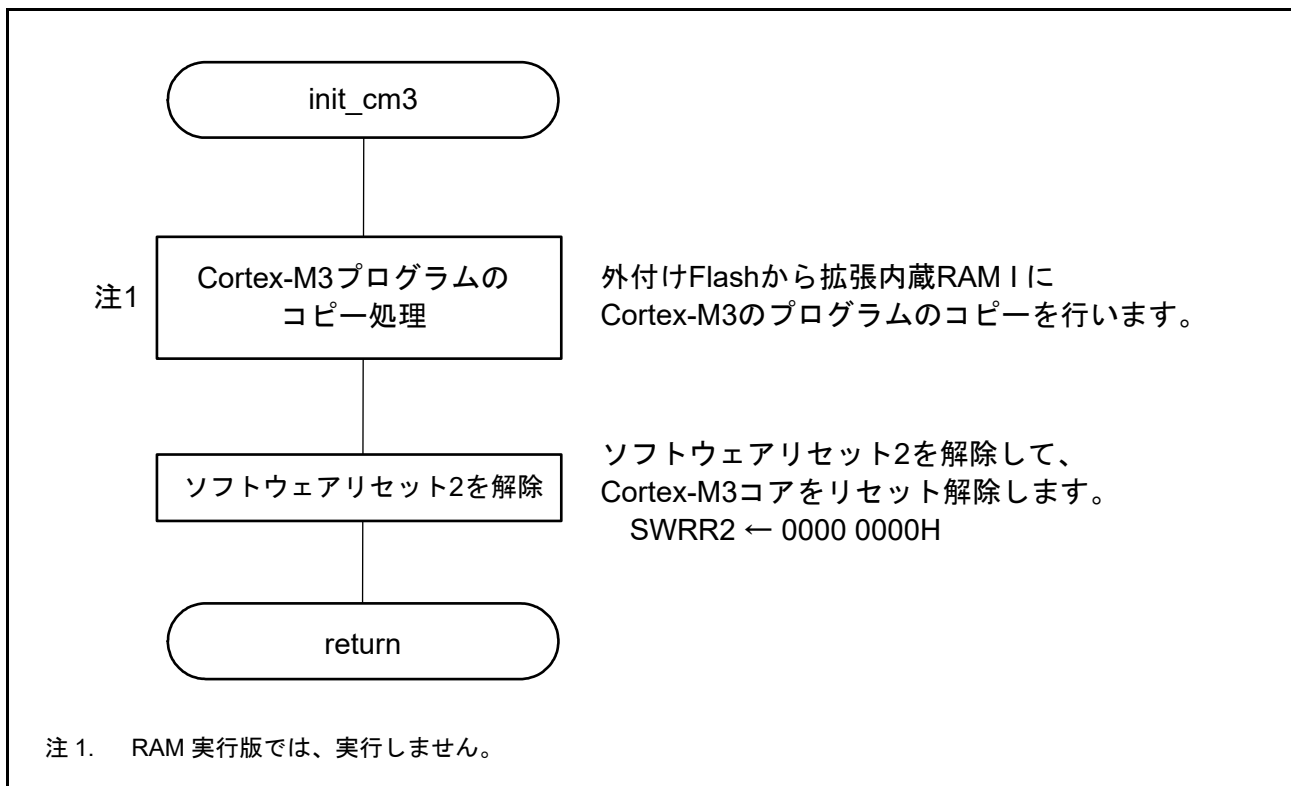


図 6.9 Cortex-M3 コア初期化処理

6.10.11 メイン処理 (Cortex-M3)

図 6.10 にメイン処理 (Cortex-M3) のフローチャートを示します。

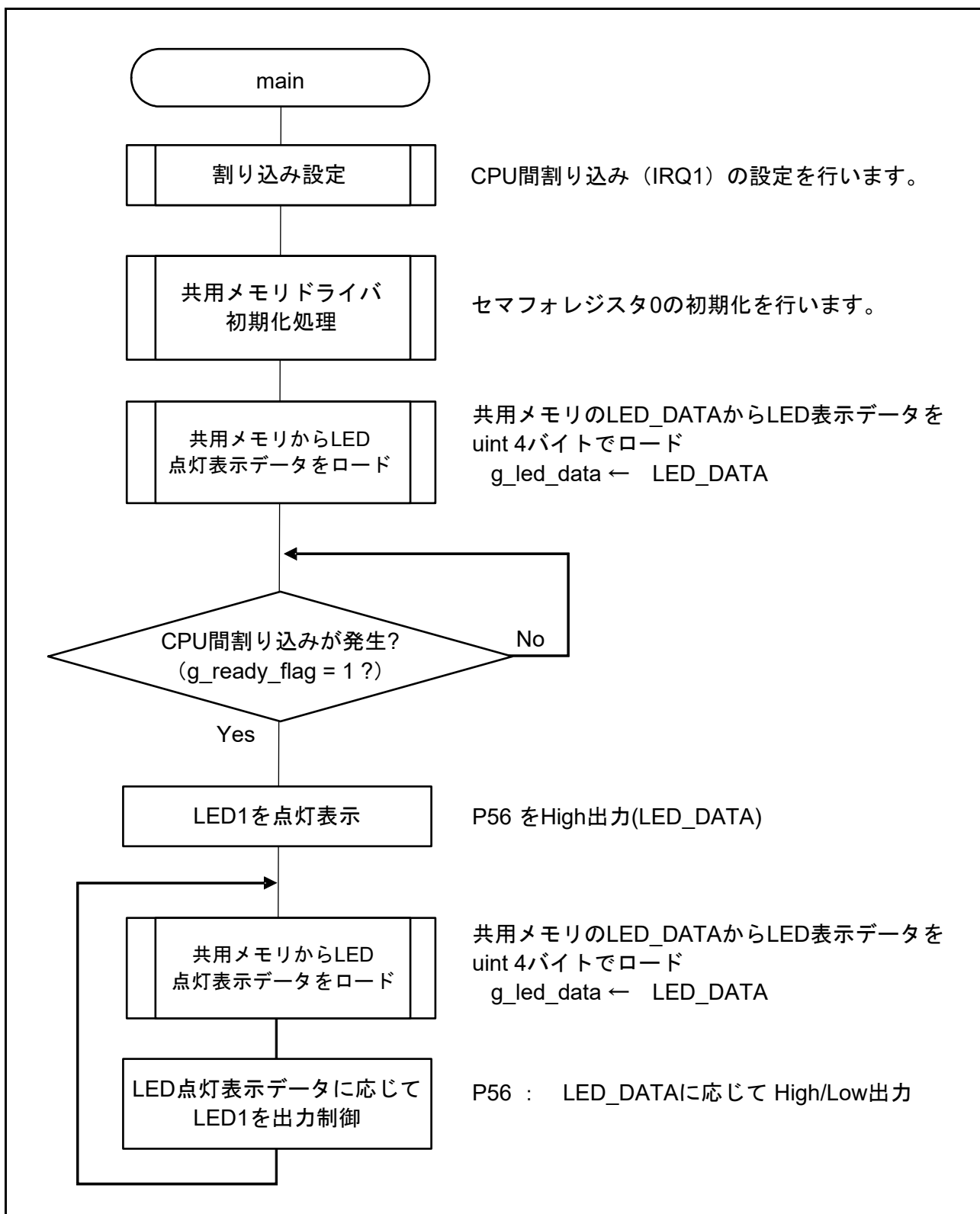


図 6.10 メイン処理 (Cortex-M3)

6.10.12 R_IRQ9 割り込み（IRQ 端子割り込み 5）処理

図 6.11 に R_IRQ9 割り込み（IRQ 端子割り込み 5）処理のフローチャートを示します。

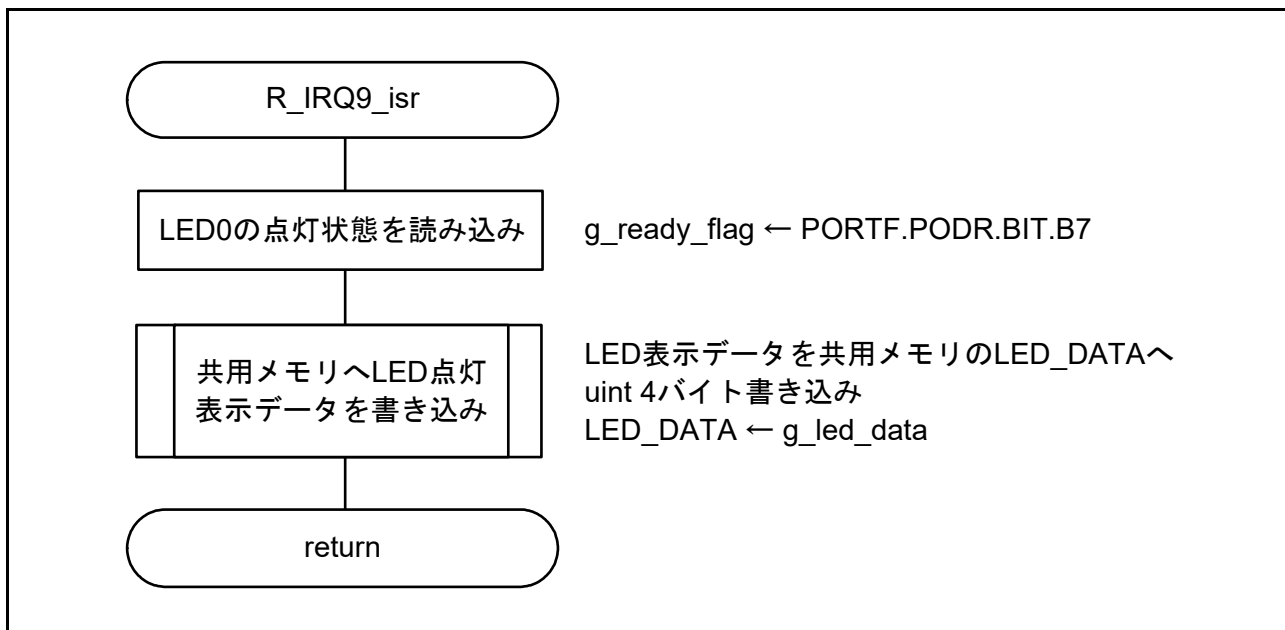


図 6.11 R_IRQ9 割り込み（IRQ 端子割り込み 5）処理

6.10.13 CPU 間割り込み処理

図 6.12 に CPU 間割り込み処理のフローチャートを示します。

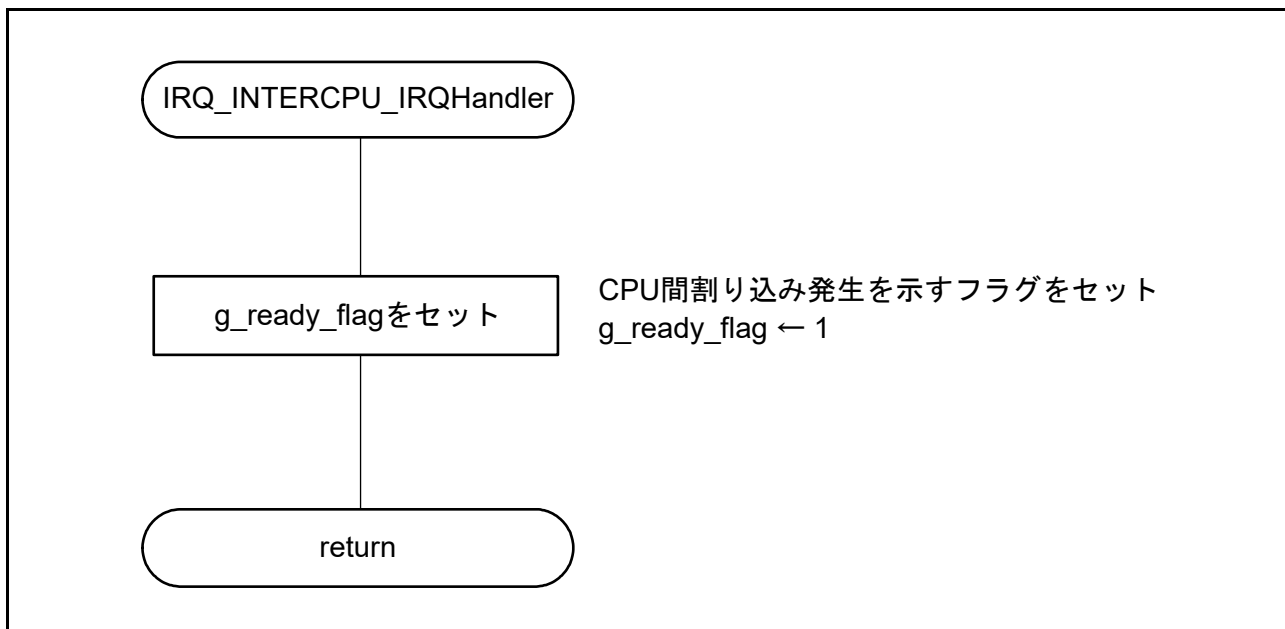


図 6.12 CPU 間割り込み処理

7. サンプルプログラム

サンプルプログラムは、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。

8. 参考ドキュメント

- ユーザーズマニュアル：ハードウェア

RZ/T1 グループ ユーザーズマニュアルハードウェア編

(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。)

RZ/T1 Evaluation Board RTK7910022C00000BR ユーザーズマニュアル

(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。)

- テクニカルアップデート／テクニカルニュース

(最新の情報をルネサス エレクトロニクスホームページから入手してください。)

- ユーザーズマニュアル：開発環境

IAR 統合開発環境 (IAR Embedded Workbench for Arm) に関しては、IAR ホームページから入手してください。

(最新版を IAR ホームページから入手してください。)

Arm 統合開発環境 (Development Studio 5) に関しては、最新版を Arm ホームページから入手してください。

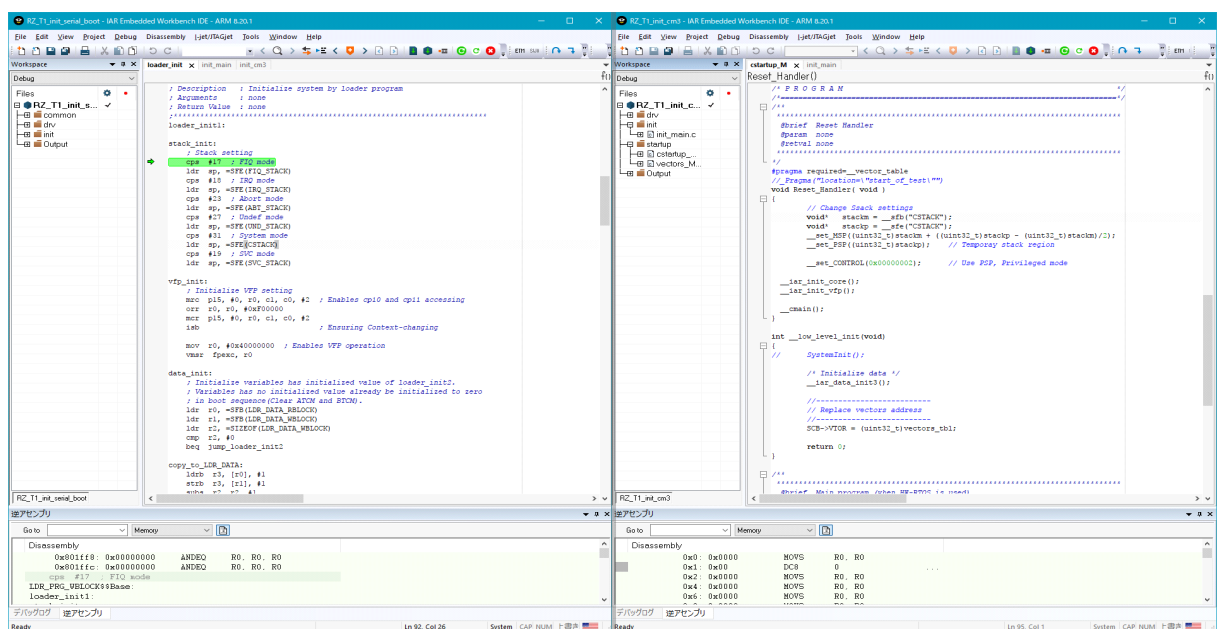
付録 1.各開発環境における補足内容

ここでは、本サンプルプログラムを各開発環境で Cortex-R4 コアと R-IN エンジン (Cortex-M3 コア) のデバッグを行う際の設定やデバッグ手順を説明します。

(EWARM: IAR 社製)

■ サンプルプログラムのデバッグ手順例 (SPI ブートモードの場合)

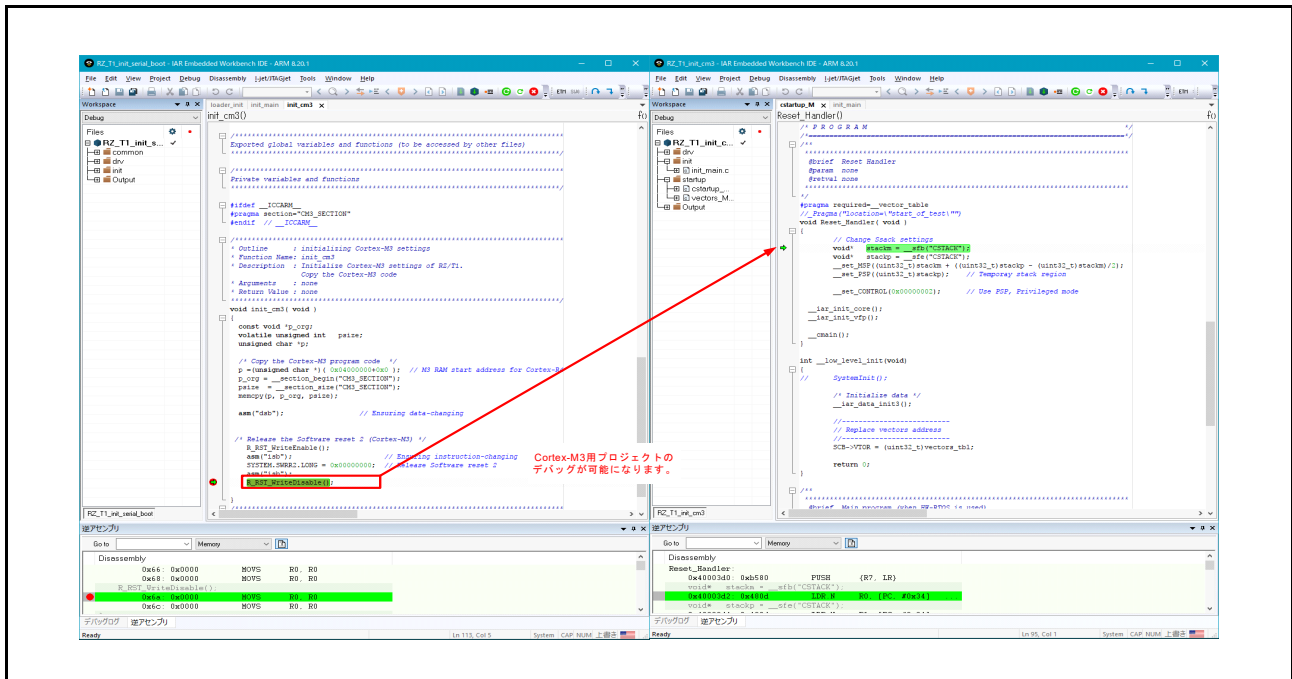
- (1) EWARM を起動し、[File] → [Open] → [Workspace] で RZ_T1_init_cm3.eww を指定して Cortex-M3 用プロジェクトを起動します。
- (2) [Project] → [Rebuild All] を実行すると、バイナリファイル RZ_T1_init_cm3.bin が生成されます。
- (3) (1) とは別の EWARM を起動し、[File] → [Open] → [Workspace] で RZ_T1_init_serial.eww を指定し、Cortex-R4 用プロジェクトを起動します。また、[Project] → [Rebuild All] でビルドを実行します。
- (4) RZ/T1 評価ボードと I-jet を接続した状態で、Cortex-R4 用プロジェクトで [Project] → [Download and Debug] を選択します。続けて Cortex-M3 用プロジェクトで [Project] → [Attach to Running Target] を選択します。



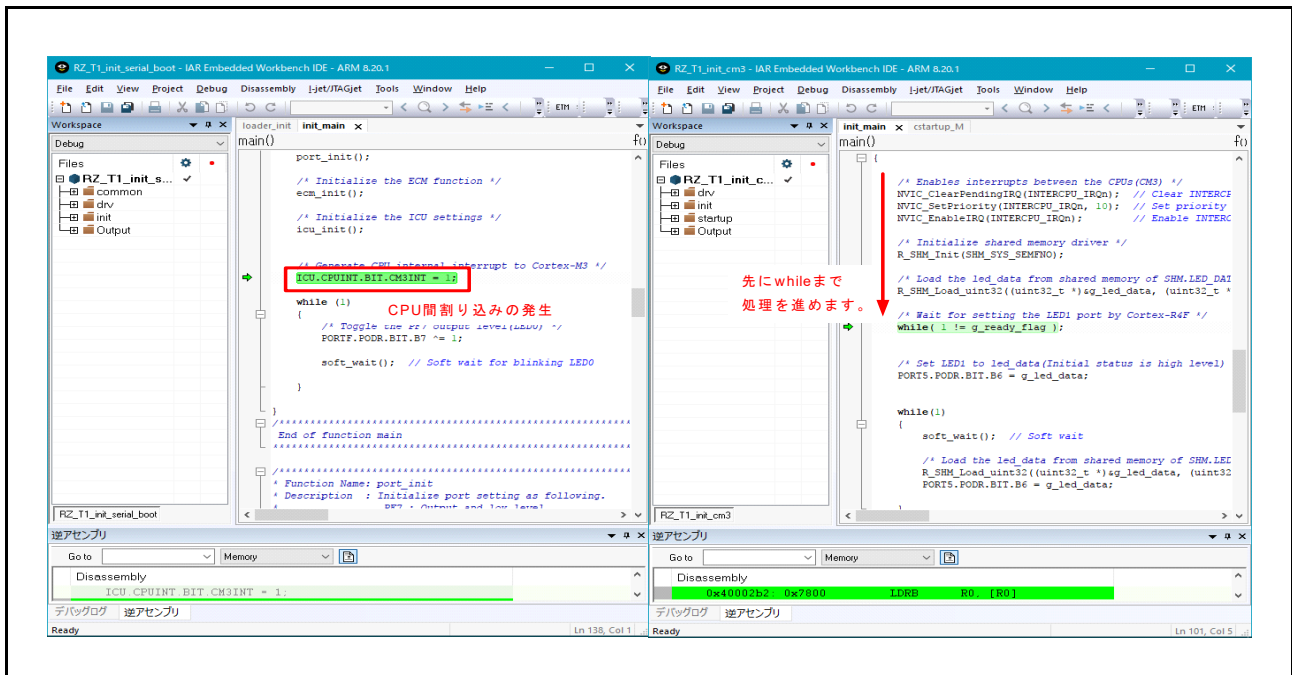
The screenshot displays two instances of the IAR Embedded Workbench IDE. The left instance shows the source code for the 'loader_init' function in assembly, with a disassembly view below it. The right instance shows the source code for the 'RZ_T1_init_cm3' project, also with a disassembly view. The disassembly views show instructions like ANDROID, MOV, DCS, and MOVs.

注. この時点で、Cortex-M3 コアはリセット状態のため、Cortex-M3 用プロジェクトのデバッグ実行は禁止です。Cortex-R4 用プロジェクトのデバッグのみ実行可能です。

- (5) Cortex-R4 用プロジェクトの `init_cm3` 関数内の Cortex-M3 のリセット解除処理を実行すると、Cortex-M3 コアがリセット解除され Cortex-M3 用プロジェクトのデバッグが可能になります。



- (6) 本サンプルプログラムでは Cortex-R4 コアから Cortex-M3 コアへ CPU 間割り込みを使用しています。実際の動作と同じ順序でデバッグするには、先に Cortex-M3 用プロジェクトを実行させて割り込み設定や共用メモリのデータロードを行います。次に Cortex-R4 用プロジェクトで CPU 間割り込みを発生させます。



- (7) 以降は、各コアのデバッグを継続して実行可能です。

■ サンプルプログラムのプロジェクト設定

R-IN Engine 搭載製品 初期設定サンプルプログラムに関する EWARM の設定を以下に示します。Cortex-R4 用プロジェクトの詳細設定はアプリケーションノート「RZ/T1 グループ初期設定」を参照してください。

表 サンプルプログラムのプロジェクト設定 (Cortex-R4)

ビルドアクション			
ビルドアクション設定	プリビルドコマンドライン	cmd /c "copy /Y "\$PROJ_DIR\$%.%.¥Cortex-M3¥Device¥Renesas¥RIN_Engine¥Source¥Project¥Init¥IAR¥Debug¥Exe¥RZ_T1_init_cm3.bin" "\$PROJ_DIR\$¥cm3.bin¥RZ_T1_init_cm3.bin""	
リンカ			
入力注 ¹	シンボルキープ	CM3_SECTION	
	ローバイナリイメージ	ファイル: \$PROJ_DIR\$¥cm3.bin¥RZ_T1_init_cm3.bin シンボル: CM3_SECTION セクション: CM3_SECTION アラインメント: 4	
デバッグ			
追加オプション	コマンドラインオプションの使用	チェックする	
		同期	非同期
		--drv_default_breakpoint=1 --macro_param_etm_rute=2 --jet_emu_param=UseCTI=0注 ²	--drv_default_breakpoint=1 --macro_param_etm_rute=2 --jet_sigprobe_opt=shared
マルチコア	対称型マルチコア	コアの数: 1	
	非対称型マルチコア マルチコアマスターモードの有効化	同期	非同期
		チェックする ポート: 53461 スレーブワークスペース: \$PROJ_DIR\$\\..¥Cortex-M3¥Device ¥Renesas¥RIN_Engine¥Project¥Init ¥IAR¥RZ_T1_init_cm3.eww スレーブプロジェクト: RZ_T1_init_cm3 スレーブ構成: Debug スレーブを実行中のターゲットに アタッチ: チェックする	チェックしない

注1. Cortex-M3用バイナリイメージをセクション CM3_SECTIONに取り込みます。

注2. EWARM V8.10.1以降から必要になります。

表 サンプルプログラムのプロジェクト設定 (Cortex-M3)

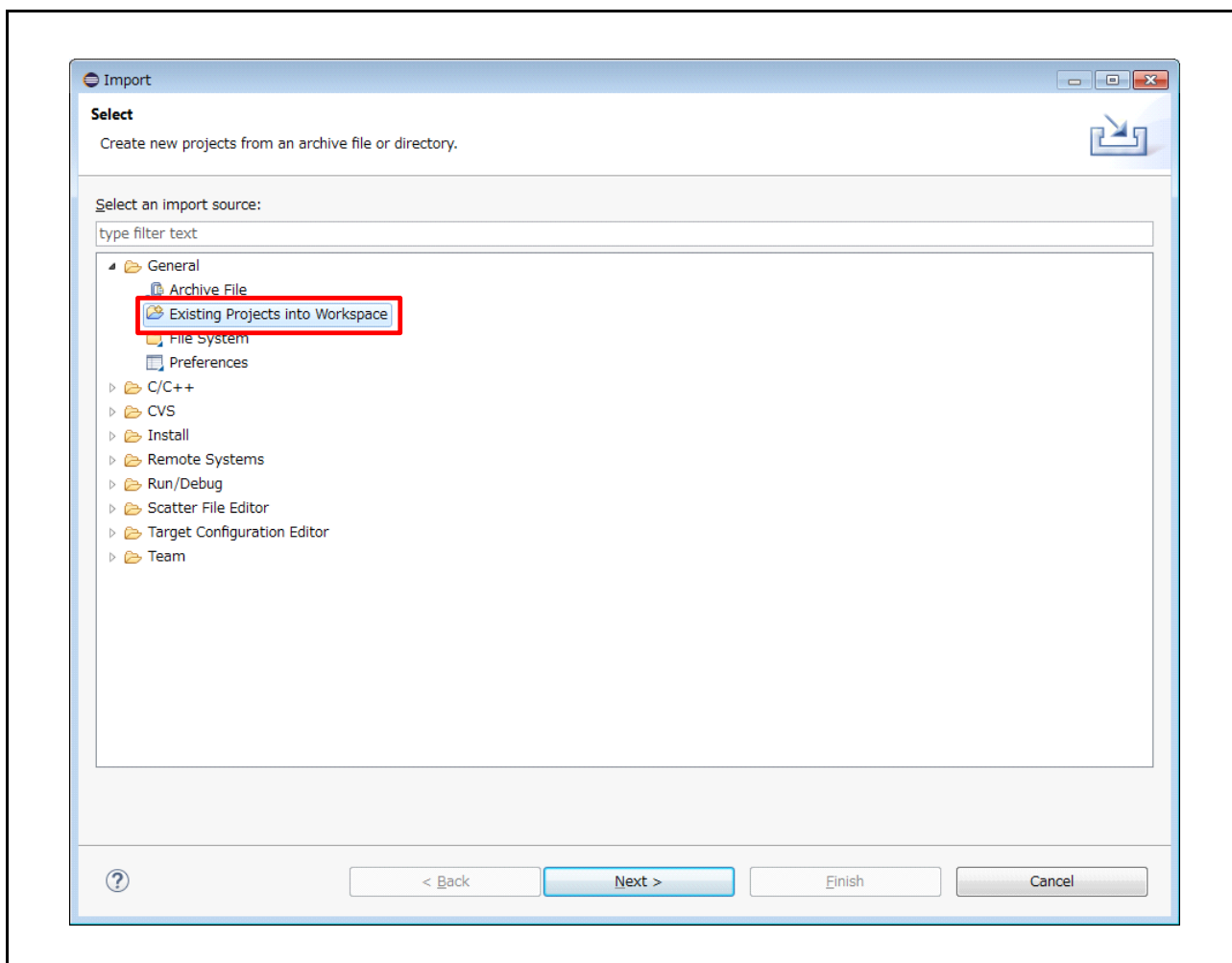
出力コンバータ			
出力	追加出力ファイルを生成	チェックする: 出力フォーマット: バイナリ	
	出力ファイル デフォルトのオーバーライド	チェックする: \$PROJ_DIR\$¥RZ_T1_init_cm3.bin	
デバッグ			
追加オプション	コマンドラインオプションの使用	同期	非同期
		--drv_default_breakpoint=1	--drv_default_breakpoint=1 --jet_sigprobe_opt=shared

オプションの詳細については IAR 社 C-SPY デバッグガイドの「C-SPY マクロ」や関連するドキュメントを参照してください。

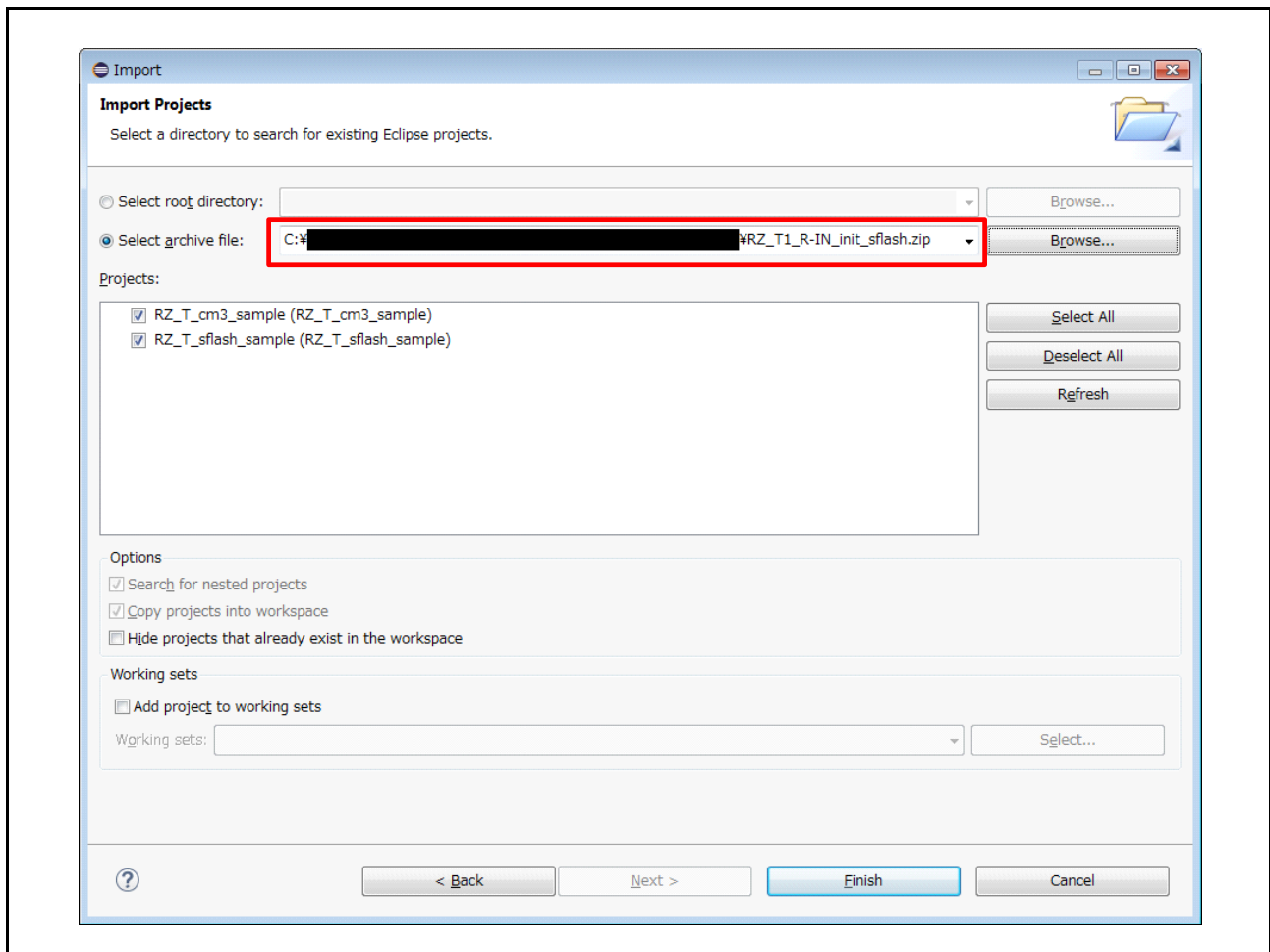
(DS-5 : Arm 社製)

■ サンプルプログラムのデバッグ手順例 (SPI ブートモードの場合)

- (1) ご利用 PC 上にサンプルプログラムを格納するためのワークスペースとして空のフォルダを作成します。
- (2) DS-5 を起動し、ワークスペースに①で作成したフォルダを指定します。
- (3) [File] → [Import...] でインポート画面を開き、[General] → [Existing Projects into Workspace] を選択して [Next] を押します。



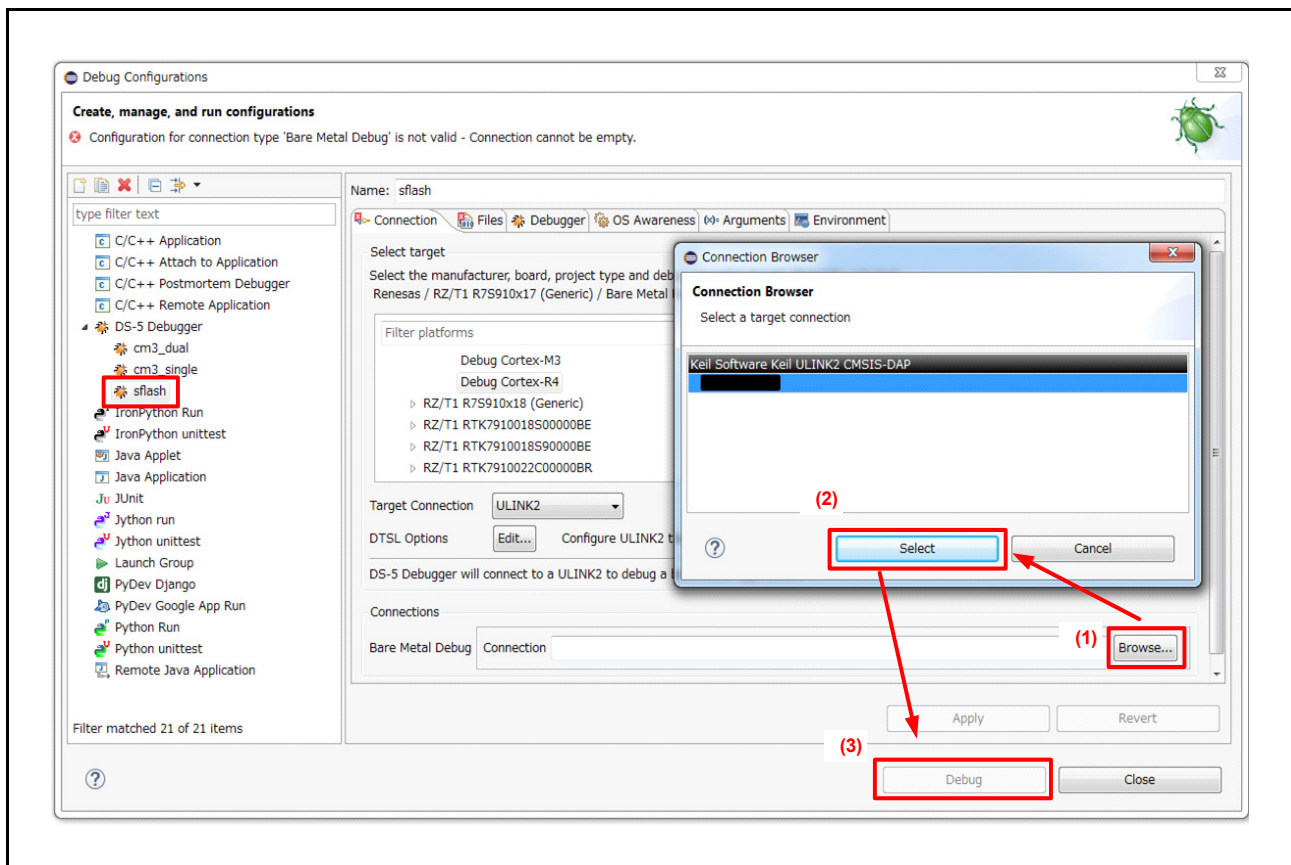
- (4) [Select archive file:] のラジオボタンにチェックを付け、[Browse...] を選択し、圧縮されたサンプルプログラム (RZ_T1_R-IN_init_sflash.zip) を選択し、[Finish] を押します。



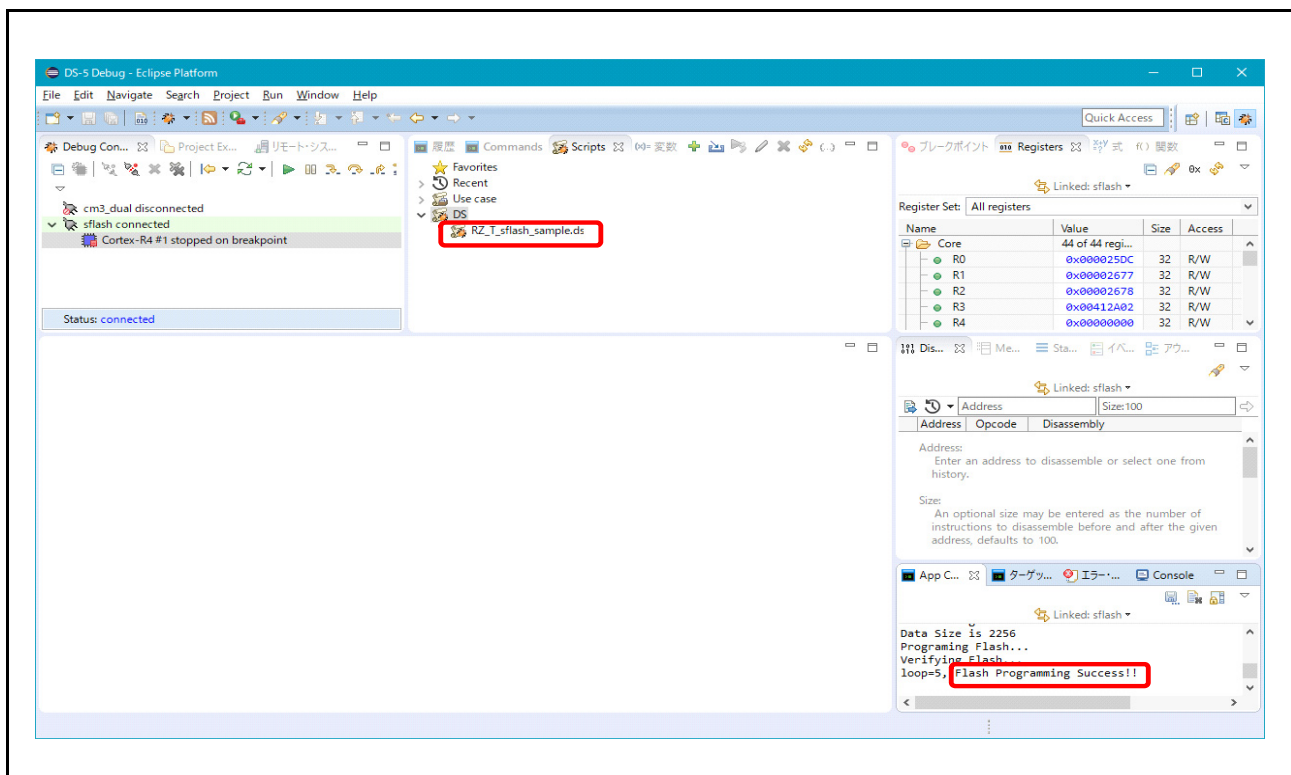
- (5) [Project] → [Build All] を実行後、[Run] → [Debug Configurations] を選択します。
- (6) RZ/T1 評価ボードと ULINK2 を接続した状態で、[DS-5 Debugger] → [sflash] を選択しデバッグ構成画面を表示します。次に下記画面の [Browse] を選択し接続ブラウザを開きます。検出されたターゲット接続を選択後、デバッグ構成画面の [Debug] を選択し Cortex-R4 コアのデバッグを開始します。

備考

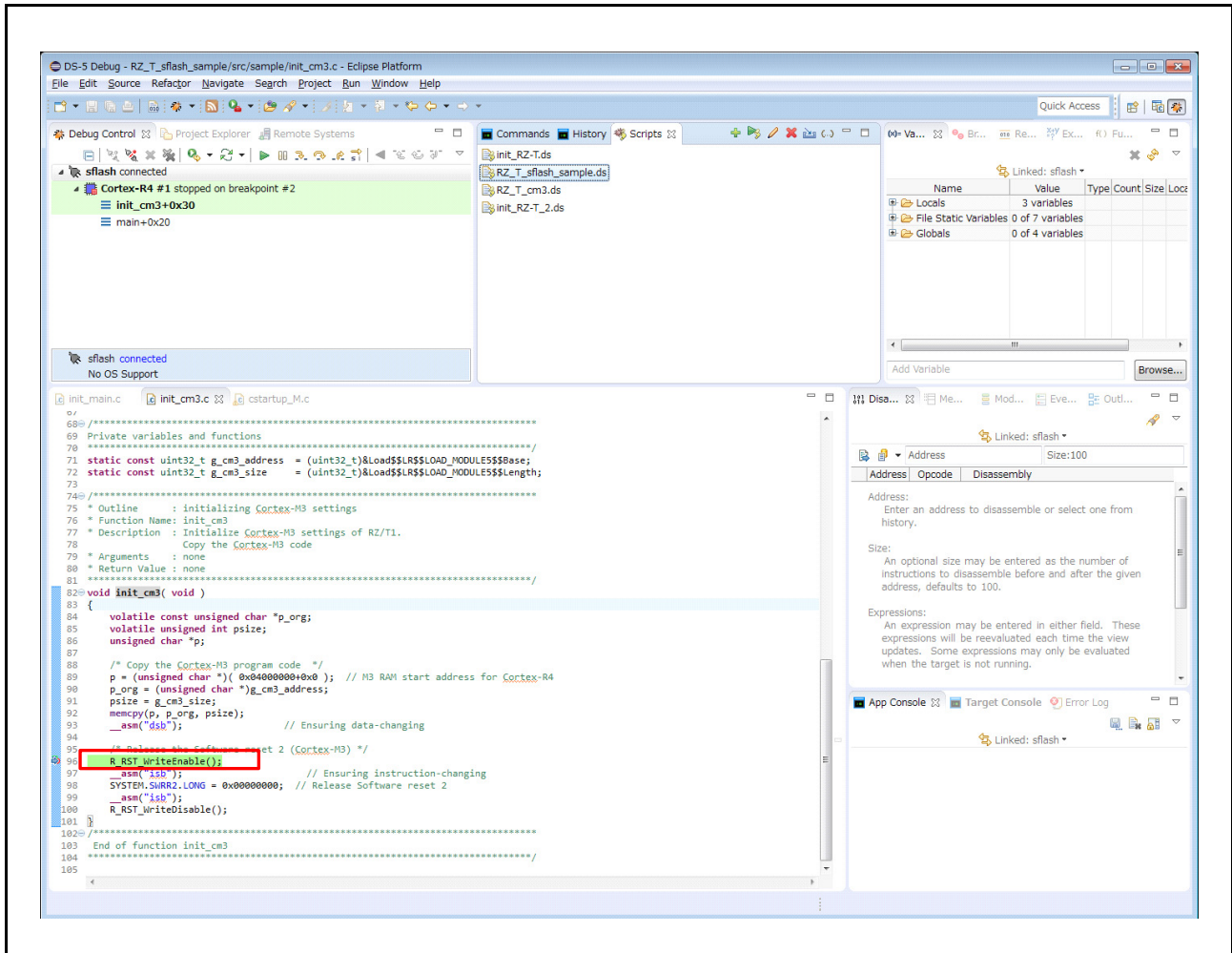
RAM 実行版の場合は、以降の手順 (7)、(8)、(9) の代わりにデバッグ接続 ram (Cortex-R4) の接続解除のみを行います。



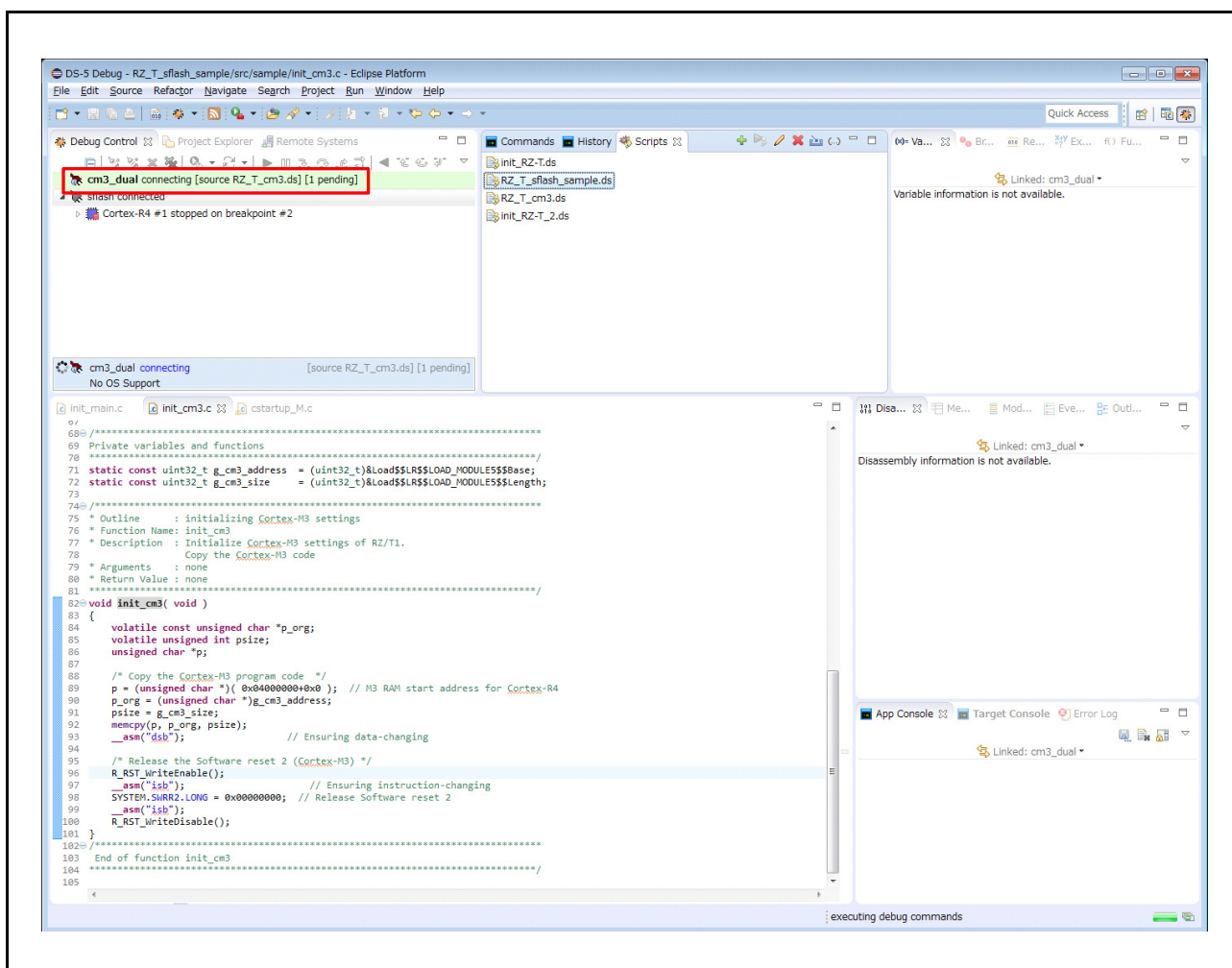
(7) ターゲット接続後、自動で ¥RZ_T_sflash_sample¥script_sflash フォルダにあるシリアルフラッシュ書き込み用のスクリプトファイル RZ_T_sflash_sample.ds が実行されます。



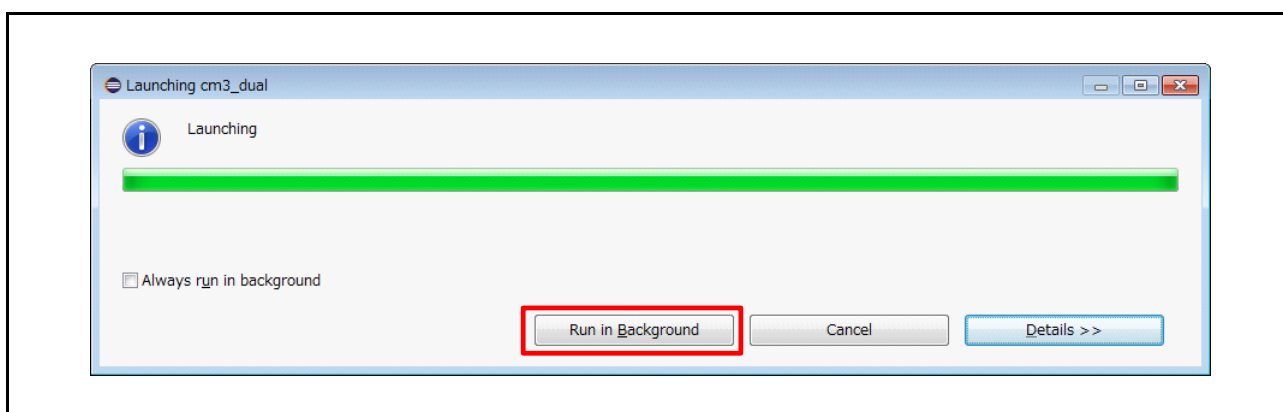
- (8) スクリプトによるフラッシュ書き込みが完了すると、アプリケーションコンソール画面に「Flash Programming Complete」が表示され外付けフラッシュへの書き込みが完了します。
- (9) System Reset を実行し、init_cm3 関数内の Cortex-M3 のリセット解除処理の直前にハードウェアブレークポイントを設定し、その処理まで実行します（ハードウェアブレークポイントは対象の処理の行番号を右クリックし [DS-5 Breakpoints] → [Toggle Hardware Breakpoint] を選択することで設定します）。



- (10) 次に [Run] → [Debug Configurations...] で Debug Configurations 画面を開き、[DS-5 debugger] → [cm3_dual] を選択し、⑥と同様に ULINK2 のターゲット接続を選択してデバッグ接続を行います（この時、スクリプトファイル: RZ_T_cm3.ds によって Cortex-M3 にリセットベクタキャッチが設定されます）。

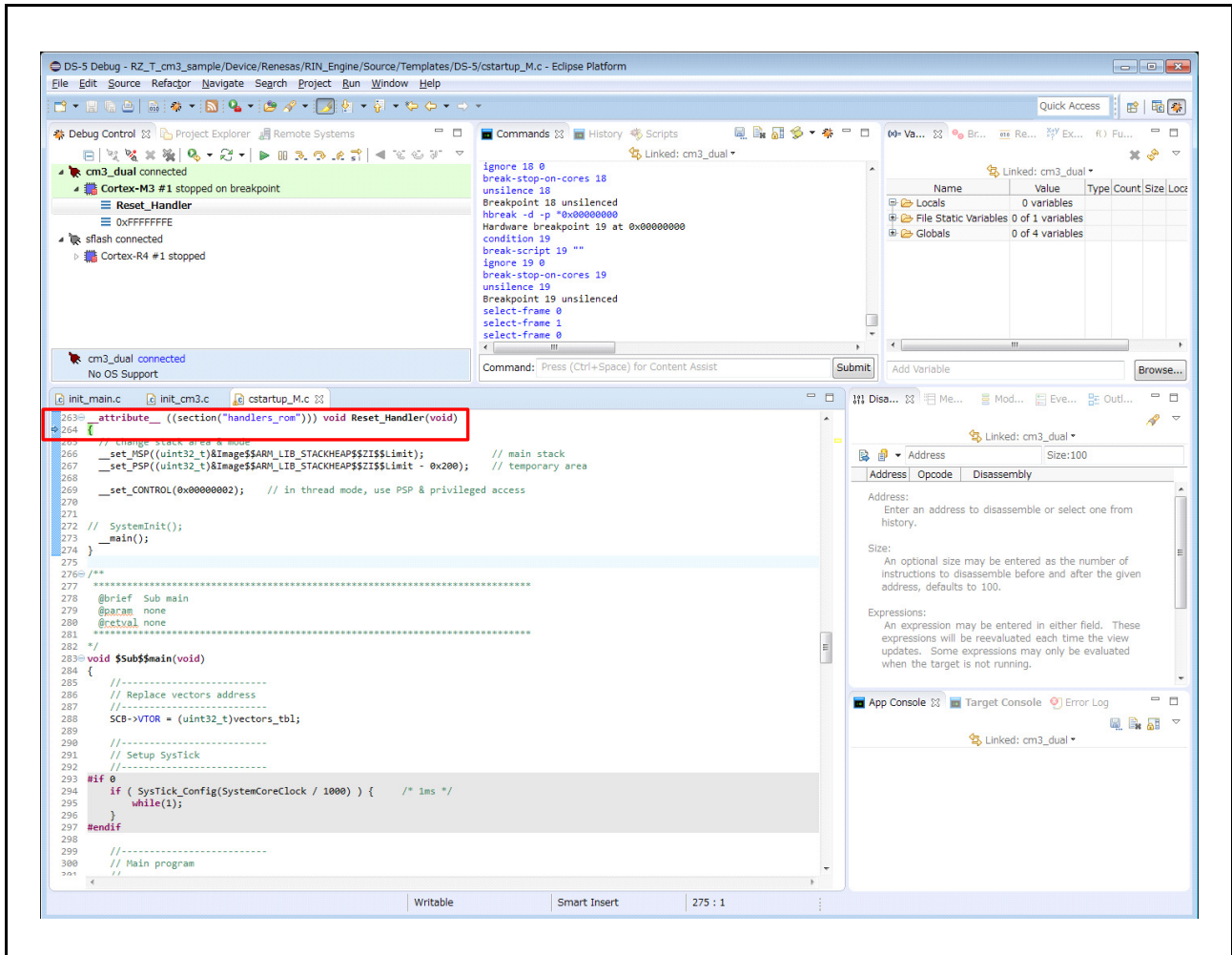


注. この時点では Cortex-M3 コアはリセット状態のため、Cortex-M3 用プロジェクトのデバッグ実行は禁止です。



注. Cortex-M3 に接続時に上記ダイアログが表示されることがありますが、「Run in Background」を選択してください。

(11) 次にデバッグ接続 sflash (Cortex-R4) にて Cortex-M3 のリセット解除処理を実行すると、Cortex-M3 コアがリセット解除され cm3_dual 接続 (Cortex-M3) のデバッグが可能になります (⑩でのリセットベクタキャッチ設定によって、リセットハンドラでブレイクします)。



(12) この後は (EWARM: IAR 社製) ■ サンプルプログラムのデバッグ手順例 ⑥ 以降と同様の手順で各コアのデバッグが可能です。

備考

デバッグ接続 cm3_single を使用することで、Cortex-M3 コア単体でのデバッグが可能です。ただし Debug Configurations で ULINK2 のターゲット接続の選択は必要です。

■ サンプルプログラムのプロジェクト設定

R-IN Engine 搭載製品 初期設定サンプルプログラムに関する DS-5 の設定を以下に示します。

表 サンプルプログラムのプロジェクト設定 (Cortex-R4)

Project->Properties-> C/C++ Build -> Settings		
Build Steps注1	Pre-build steps Command:	fromelf --bin --output =../cm3.bin ../RZ_T_cm3_sample/Debug/RZ_T_cm3_sample.axf
	Post-build steps Command:	after_build.bat \${ProjName} (RAM実行版は指定なし)

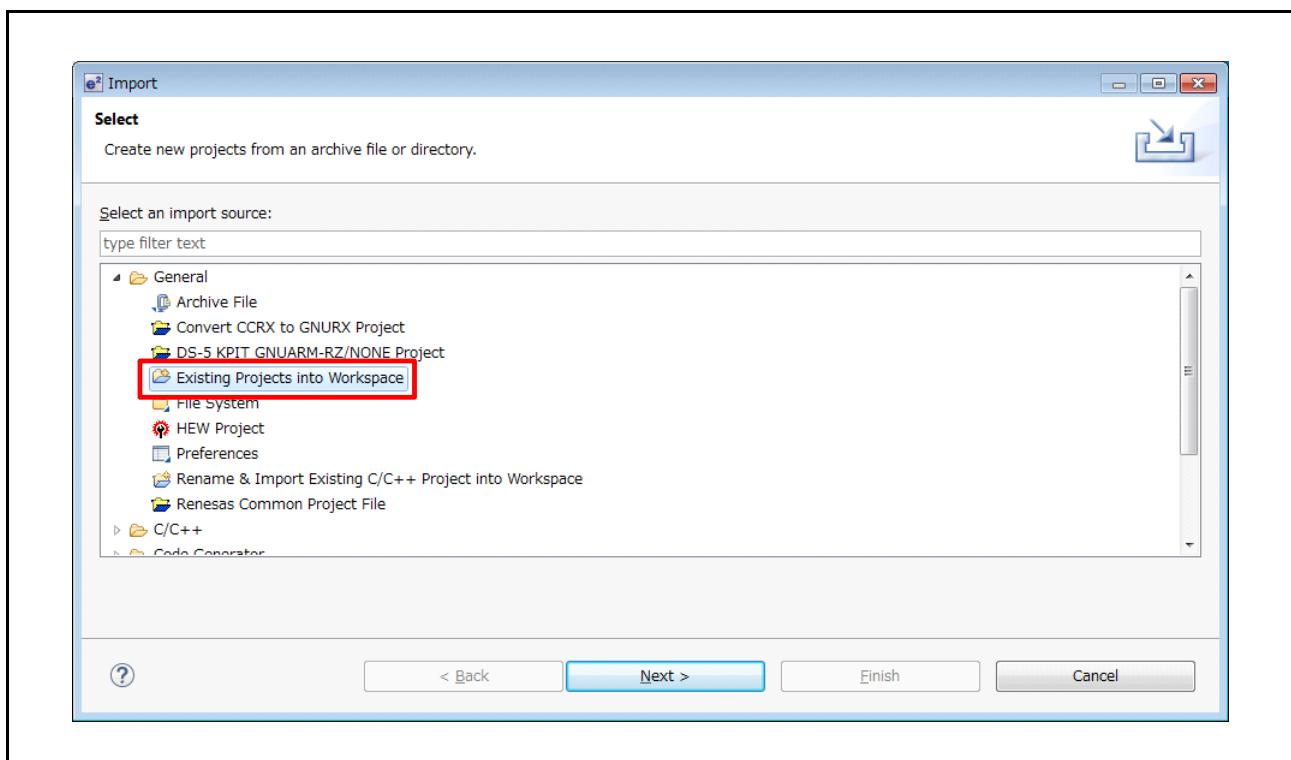
注1. Cortex-M3用バイナリイメージをセクション CM3_SECTION として取り込みます。

オプションの詳細については Arm 社の関連するドキュメントを参照してください。

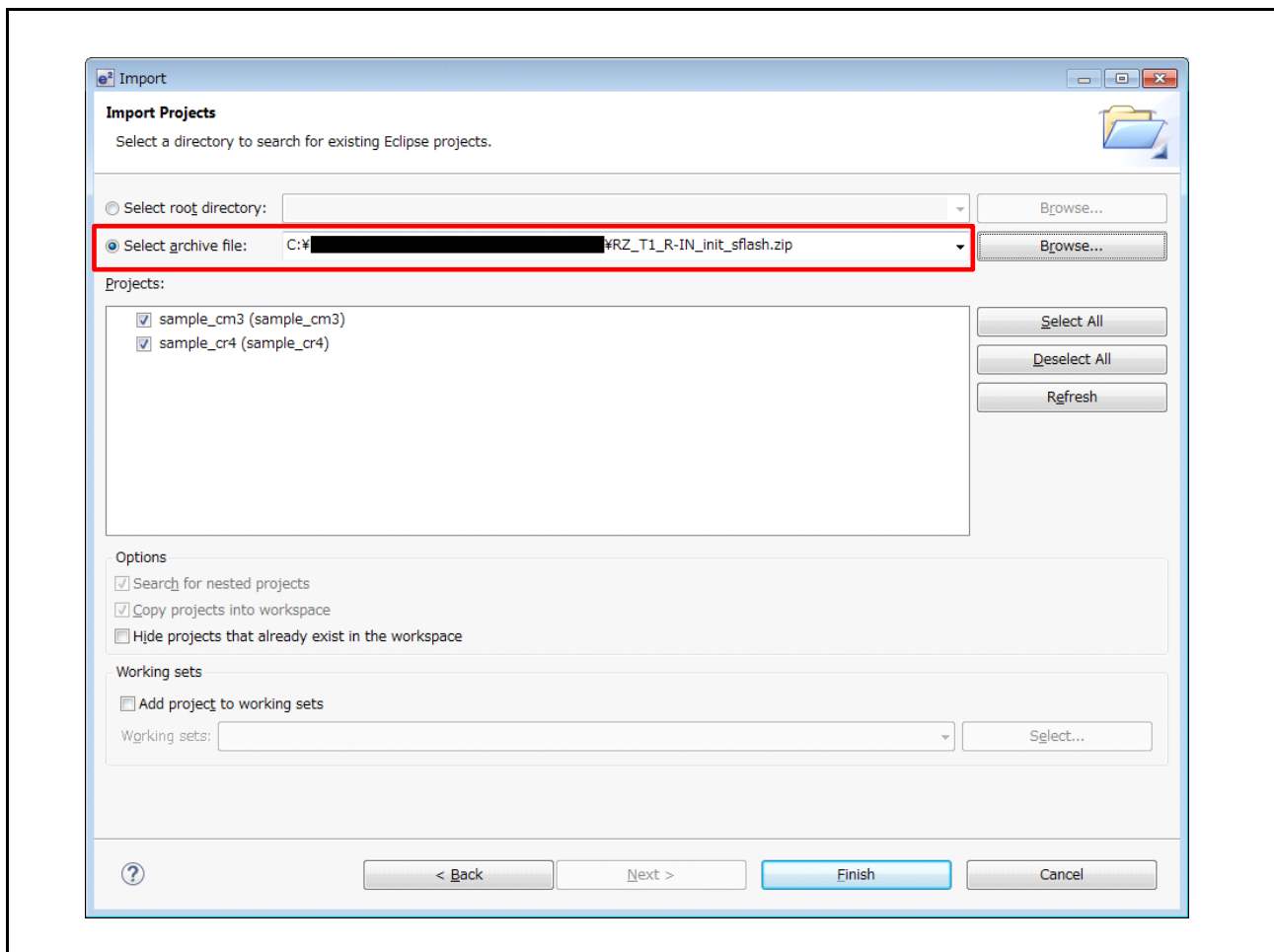
(e2studio : RENESAS 社製)

■ サンプルプログラムのデバッグ手順例 (SPI ブートモードの場合)

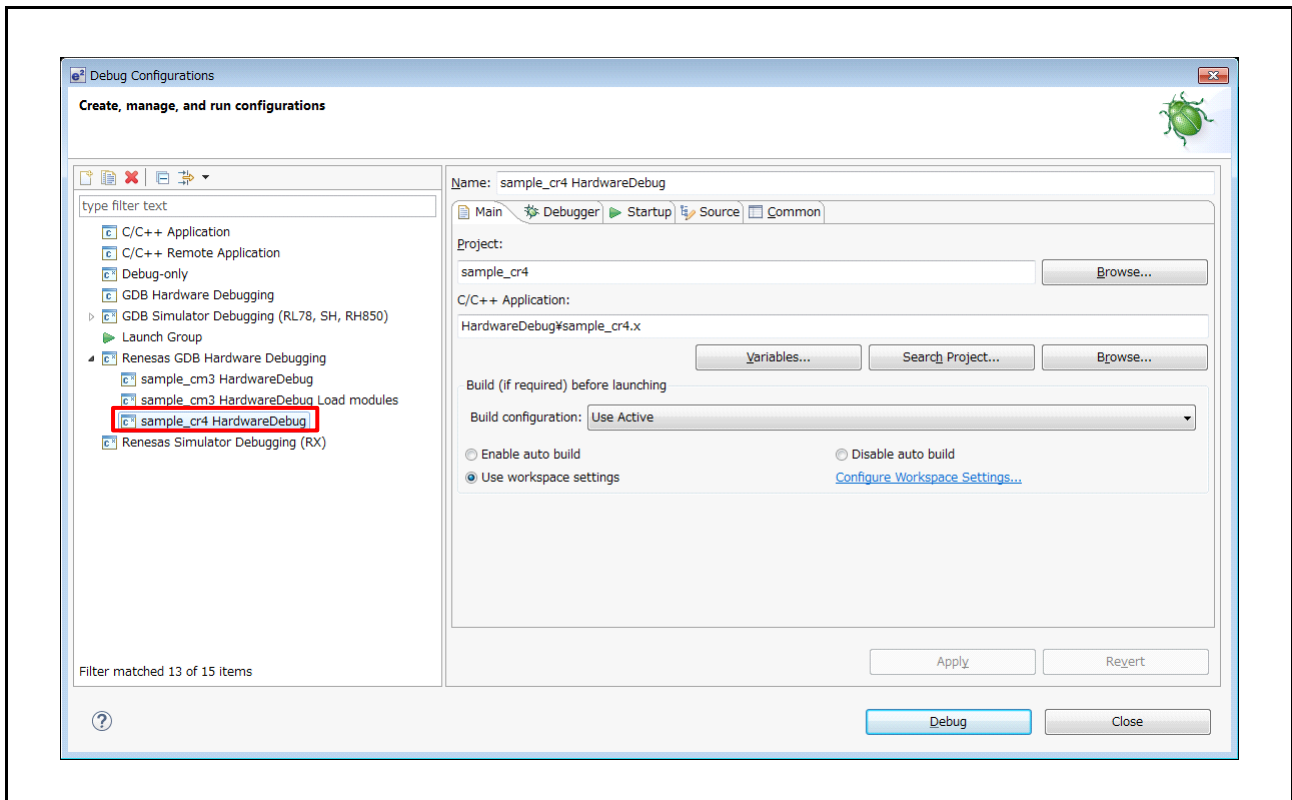
- (1) ご利用 PC 上にサンプルプログラムを格納するためのワークスペースとして空のフォルダを作成します。
- (2) e2studio を起動し、ワークスペースに①で作成したフォルダを指定します。
- (3) [File] → [Import...] でインポート画面を開き、[General] → [Existing Projects into Workspace] を選択して [Next] を押します。



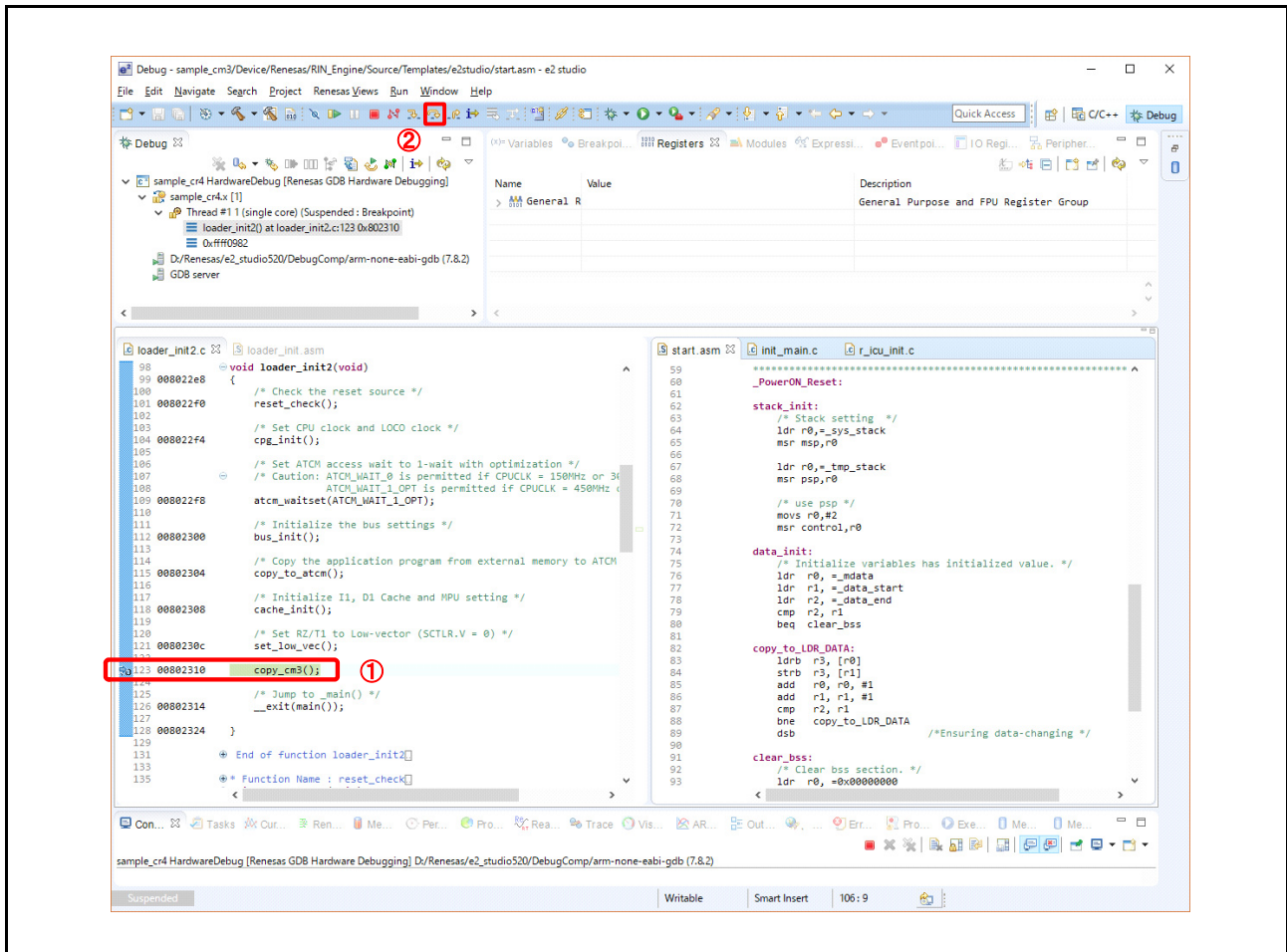
- (4) [Select archive file:] のラジオボタンにチェックを付け、[Browse...] を選択し、圧縮されたサンプルプログラム (RZ_T1_R-IN_init_sflash.zip) を選択し、[Finish] を押します。



- (5) [Project] → [Build All] を実行後、[Run] → [Debug Configurations...] を選択し、Debug Configurations 画面を開きます。
- (6) RZ/T1 評価ボードと J-Link を接続した状態で、[Renesas GDB Hardware Debugging] → [sample_cr4 HardwareDebug] を選択後、[Debug] を押すことで、Cortex-R4 コアのデバッグを開始します。



(7) ターゲット接続後、[Resume] を実行し Cortex-M3 のプログラム書き込み処理の前まで実行します。

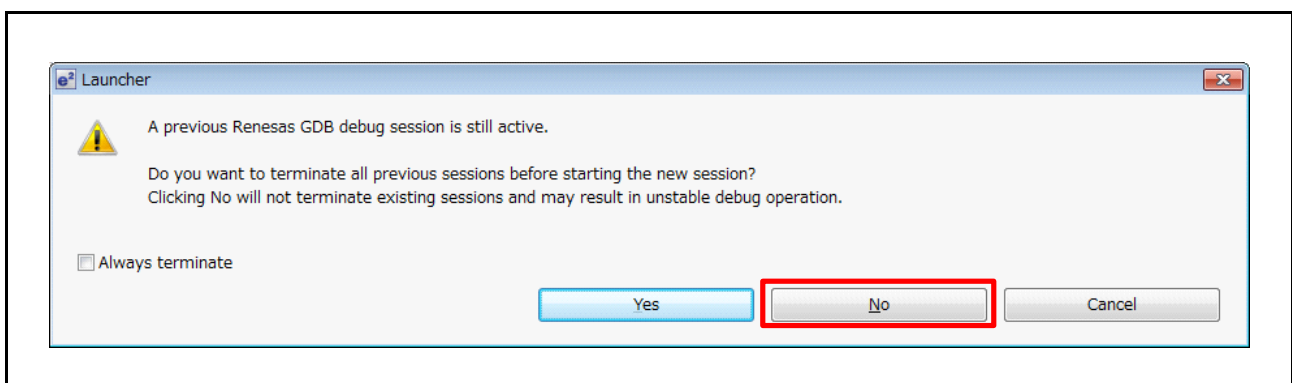


(8) 次に [Run] → [Debug Configurations...] で Debug Configurations 画面を開き、[Renesas GDB Hardware Debugging] → [sample_cm3 HardwareDebug] を選択し、[Debug] を押します。

この時点では Cortex-M3 コアのプログラムが書き込まれていないため、Cortex-M3 用プロジェクトのデバッグ実行は禁止です。

注. [Debug] を押すと、既に実行しているデバッグ接続を切断するかどうかを選択する

下記ダイアログが表示されます。今回は Cortex-R4 コアと R-IN エンジン (Cortex-M3 コア) の制御を行うため “NO” を選択してください。



- (9) [Debug] ビューの「Sample_cr4.x」内の「Thread #xxx」にフォーカスがある状態で [Run] → [Step Over F6] を押して、Cortex-M3 のプログラム書き込み処理を実行します。
- (10) [Debug] ビューの「Sample_cm3.x」内の「Thread #xxx」を選択した状態で [Resume] を実行すると、Cortex-M3 コアのデバッグが可能になります。

備考

デバッグ時にはデバッガから Cortex-M3 のリセット解除を行います。この後に Cortex-R4 によって再度リセット解除処理が実行されることとなりますが、問題ありません (デバッガを使用しない場合は、Cortex-R4 のリセット解除処理によって Cortex-M3 のリセットが解除されません)。

- (11) この後は (EWARM : IAR 社製) ■ サンプルプログラムのデバッグ手順例 ⑥ 以降と同様の手順で各コアのデバッグが可能です。

備考

「Sample_cr4.x」内の「Thread #xxx」を選択した場合は Cortex-R4 側のデバッグが、「Sample_cm3.x」内の「Thread #xxx」を選択した場合は Cortex-M3 側のデバッグが可能になります。

■ サンプルプログラムのプロジェクト設定

R-IN Engine 搭載製品 初期設定サンプルプログラムに関する e2studio の設定を以下に示します。

表 サンプルプログラムのプロジェクト設定 (Cortex-R4)

Project->Properties->Project References		
Project references for 'sample_cr4' 注1	sample_cm3	チェック

注1. Cortex-R4のプロジェクトをビルドする際、自動的にCortex-M3のプロジェクトをビルドします。

表 サンプルプログラムのプロジェクト設定 (Cortex-M3)

Project->Properties-> C/C++ Build -> Settings		
Build Steps注1	Post-build steps Command:	arm-none-eabi-objcopy -I elf32-littlearm -O binary sample_cm3.x cm3.bin & arm-none-eabi-objcopy -I binary -O elf32-littlearm -B arm --rename-section .data =.cm3, alloc, data, readonly,load,contents cm3.bin cm3.o & copy / Y cm3.o ..%.¥sample_cr4¥cm3.bin¥cm3.o

注1. Cortex-R4側でCortex-M3用バイナリイメージをセクション .cm3 として取り込むためにバイナリデータをコピーします。

表 サンプルプログラムのデバッグ構成

Run -> Debug Configurations		
Debugger -> Connection Settings -> Connection	Reset on connection	"NO"
	Reset before run	【sample_cr4 HardwareDebug】 "YES" 【sample_cm3 HardwareDebug、 sample_cm3 HardwareDebug Load modules】 "NO"
Startup	Runtime Options	【sample_cr4 HardwareDebug、sample_cm3 HardwareDebug】 「Set break point at:」にチェックつけ、 「_PowerON_Reset」を設定注1
	Run Commands	【sample_cm3 HardwareDebug】 「set \$pc=&_PowerON_Reset」を設定注1

注1. "_PowerON_Reset"は各プロジェクトのエントリポイントです。エントリポイントが異なる場合は置き換えてください。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

<http://japan.renesas.com/>

お問い合わせ先

<http://japan.renesas.com/contact/>

改訂記録	R-IN Engine 搭載製品 初期設定 アプリケーションノート
------	-----------------------------------

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2015.10.15	—	初版発行
1.10	2016.02.15	全般	
		—	"デュアルコア制御" を、"R-IN Engine 搭載製品 初期設定"へ記述変更
		2. 動作環境	
		5	表2.1 動作環境 統合開発環境 一部修正
		6. ソフトウェア説明	
		9	説明文追加
		15	6.2.4 必要メモリサイズ 表6.7 タイトル一部追加、表6.8、表6.9 追加
		付録1.各開発環境における補足内容	
		33	付録1.各開発環境における補足内容、一部記述変更
		36 ~ 48	(DS-5 : ARM 社製)、(e2studio : RENESAS 社製) 全面改訂
1.20	2017.08.03	全体	Cortex-R4F → Cortex-R4に変更
		2. 動作環境	
		5	表2.1 動作環境 統合開発環境の内容変更
		6. ソフトウェア説明	
		—	6.2.4 必要メモリサイズ 削除
		付録1.各開発環境における補足内容	
		31	(EWARM : IAR 社製) ■サンプルプログラムのデバッグ手順例(SPIブートモードの場合) (4)を変更: [Debug without Downloading] → [Attach to Running Target]
		33	(EWARM : IAR 社製) ■サンプルプログラムのプロジェクト設定 (表)サンプルプログラムのプロジェクト設定(Cortex-R4): 変更、注2, 注3を削除
		44	(e2studio : RENESAS 社製) ■サンプルプログラムのデバッグ手順例(SPIブートモードの場合) (7): 説明を変更、図を変更
		44	(e2studio : RENESAS 社製) ■サンプルプログラムのデバッグ手順例(SPIブートモードの場合) (8): 説明を追加
		45	(e2studio : RENESAS 社製) ■サンプルプログラムのデバッグ手順例(SPIブートモードの場合) (9): 説明を変更、図を削除
		46	(e2studio : RENESAS 社製) ■サンプルプログラムのプロジェクト設定 (表)サンプルプログラムのデバッグ構成を変更、注1を追加
		1.30	2018.04.27
2. 動作環境			
5	表2.1 動作環境 統合開発環境の内容を変更、IARシステムズ製統合開発環境のバージョンを変更		
6. ソフトウェア説明			
11	表6.3 使用するセクション(Cortex-M3) HEAPをreadwriteの下に移動、MAIN_STACK領域を削除		
13	図6.3 セクション配置(Cortex-M3) HEAPをreadwriteの下に移動		
13	6.3 セクション配置(Cortex-M3) MAIN_STACK領域を削除		
16	表6.7 大域変数一覧 型(uint32_t)の使用関数を変更		
付録1.各開発環境における補足内容			
31, 32	(EWARM : IAR社製) サンプルプログラムのデバッグ手順例 (4)~(6)の図を変更		
33	(表)サンプルプログラムのプロジェクト設定(Cortex-R4) デバッグ/追加オプションの同期を変更、注2.を追加		
36	(DS-5 : ARM社製) サンプルプログラムのデバッグ手順例 (7)の説明、および図を変更		
45	(表)サンプルプログラムのプロジェクト設定(Cortex-M3) 変更		
45	(表)サンプルプログラムのデバッグ構成 Startup/Run Commandsの説明を変更		
1.31	2018.07.13	—	サンプルプログラムの更新にともなう改版

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。

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. 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（予約領域）のアクセス禁止

【注意】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、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）がありません。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。

5. 製品間の相違について

【注意】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っていると、内部 ROM、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、ノイズ輻射量などが異なる場合があります。型名が異なる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。お客様の機器・システムの設計において、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、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、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。
5. 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。

6. 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。
7. 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。
8. 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。
9. 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。
10. 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。
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禁じます。
12. 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.4.0-1 2017.11)



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/>