

サイプレスはインフィニオン テクノロジーズになりました

この表紙に続く文書には「サイプレス」と表記されていますが、これは同社が最初にこの製品を開発したからです。新規および既存のお客様いずれに対しても、引き続きインフィニオンがラインアップの一部として当該製品をご提供いたします。

文書の内容の継続性

下記製品がインフィニオンの製品ラインアップの一部として提供されたとしても、それを理由としてこの文書に変更が加わることはありません。今後も適宜改訂は行いますが、変更があった場合は文書の履歴ページでお知らせします。

注文時の部品番号の継続性

インフィニオンは既存の部品番号を引き続きサポートします。ご注文の際は、データシート記載の注文部品番号をこれまで通りご利用下さい。

Traveo II ファミリ MCU スタートアップガイド

著者: Umeno, Kazuo

関連製品ファミリ: Traveo™ II ファミリ CYT2/CYT3/CYT4 シリーズ

関連アプリケーションノート: [関連ドキュメント](#) 参照

本アプリケーションノート (AN220118) では、Arm® Cortex®-M4F, M7 および M0+ベースのマイクロコントローラである Traveo™ II 製品を簡単に紹介します。また、Traveo II 開発環境の設定方法についても説明します。

目次

1 はじめに	1	3.5 IAR Embedded Workbench for Arm を使用した SDL の使用例	35
2 特長	2	3.6 開発ツール	42
2.1 CYT2B シリーズ	2	3.7 フラッシュプログラミングツール	43
2.2 CYT4B シリーズ	3	4 まとめ	44
2.3 CYT4D シリーズ	5	5 用語集	44
2.4 その他のシリーズ	6	6 関連ドキュメント	45
3 開発環境とツール	7	改版履歴	46
3.1 評価ボード	7	ワールドワイドな販売と設計サポート	47
3.2 サンプルドライバライブラリ	9		
3.3 ペリフェラルドライバ	19		
3.4 GHS MULTI を使用した SDL の使用例	21		

1 はじめに

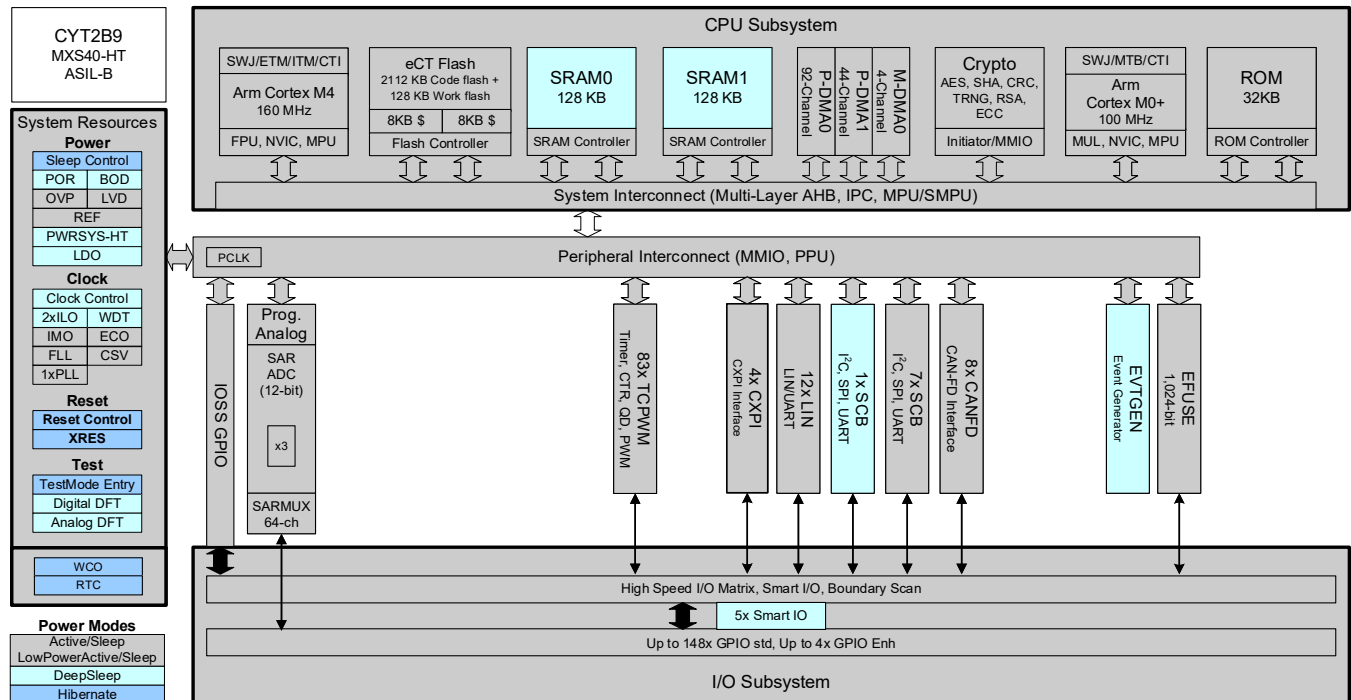
このアプリケーションノートでは、Traveo II ファミリの CYT2, CYT3, および CYT4 シリーズの特長と概要と使い始めるための開発環境やツールについて説明します。このシリーズでは、Arm Cortex-M ベースの CPU コアと、CAN FD, メモリ, アナログ, およびデジタル周辺機能を 1 つのチップに搭載しています。CYT2 シリーズは、1 つの Arm Cortex-M4F ベースの CPU (CM4) と Cortex-M0+ベースの CPU (CM0+) を持ちます。CYT4 シリーズは、2 つの Arm Cortex-M7 ベースの CPU (CM7) と CM0+を持ち、CYT3 シリーズは、1 つの CM7 と CM0+を持ちます。CYT2B シリーズには、64 ピンから 176 ピン LQFP パッケージの製品ラインアップがあります。CYT4B シリーズには、176 ピン TEQFP パッケージから 320 ボール BGA パッケージの製品ラインアップがあります。CYT4D シリーズには、327 ボール BGA と 500 ボール BGA パッケージの製品ラインアップがあります。また、CYT3BB および CYT4BB シリーズには 100 ピン TEQFP パッケージから 272 ボール BGA パッケージの製品ラインアップがあります。本書で記載されている機能や用語については、Architecture Technical Reference Manual (Architecture TRM)の Getting Start 章を参照してください。詳細は[関連ドキュメント](#)を参照してください。

2 特長

2.1 CYT2B シリーズ

CYT2B シリーズは、ボディコントロールユニットなどの車載システム向けの Traveo II MCU です。このデバイスは、メイン処理用の Arm Cortex-M4 CPU と、周辺およびセキュリティ処理用の別個の Cortex-M0+ CPU をベースにしており、高度な 40nm プロセスで製造されています。また、このデバイスは、Controller Area Network with Flexible Data rate (CAN FD) および Local Interconnect Network (LIN) などのボディコントロール機能をサポートしています。これらの製品は、安全なコンピューティングプラットフォームを可能にし、サイプレスの低消費電力フラッシュメモリと、複数の高性能アナログおよびデジタル機能を組み込んでいます。図 1 に CYT2B9 のブロックダイアグラムを示します。

図 1. CYT2B9 のブロックダイアグラム



2.1.1 機能概要

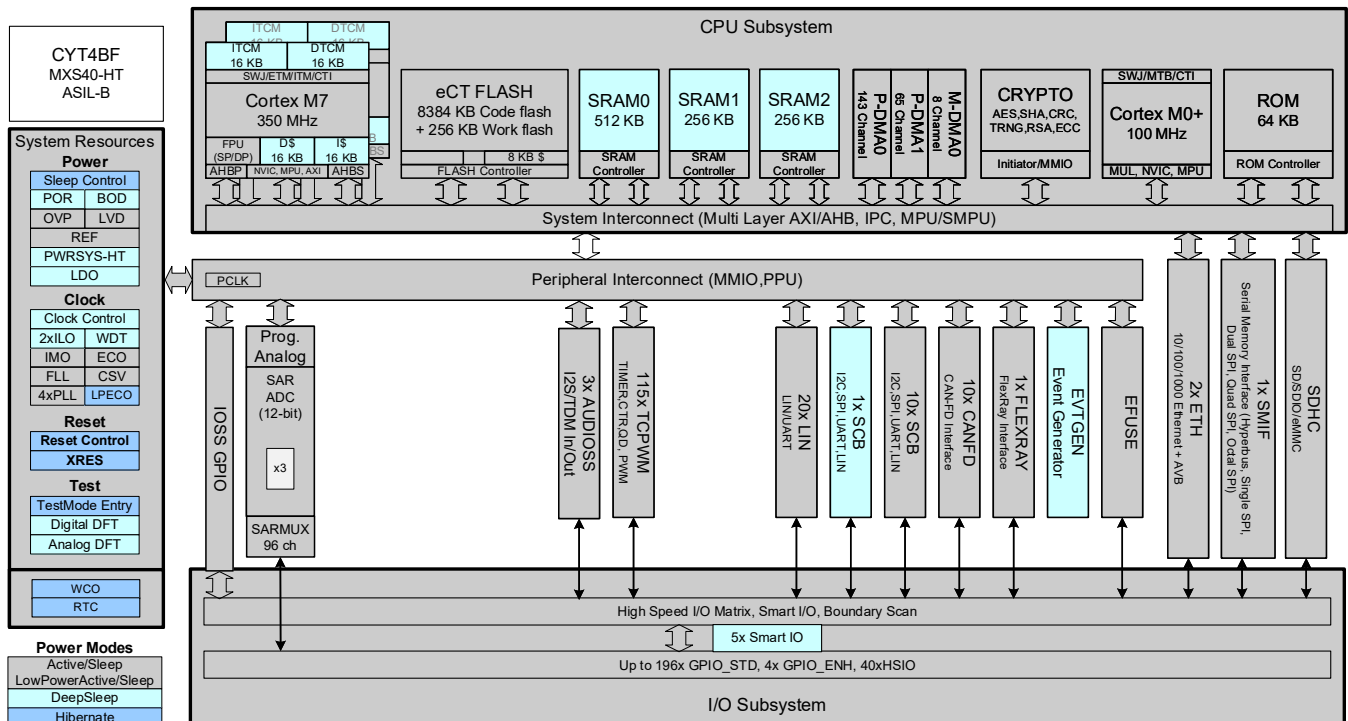
- デュアル CPU サブシステム
 - 160 MHz 32 ビット CM4 シングルサイクル乗算、浮動小数点演算ユニット(FPU)、メモリ保護ユニット(MPUs) 付き
 - 100 MHz 32 ビット CM0+ MPU 付き
- 搭載メモリ
 - 最大 2112 KB (1984 KB + 128 KB) のコードフラッシュと最大 128 KB (96 KB + 32 KB) のワークフラッシュ
 - RWW (Read-While-Write) により、コードフラッシュから実行中にコードフラッシュ/ワークフラッシュを更新可能
 - シングルおよびデュアルバンクモードをサポート
- CAN FD
 - 従来の CAN と比較してデータレートが向上、ただし、物理層トポロジとデバイス (トランシーバ) による制限あり
 - 最大 8 Mbps をサポート

- 最大 6 つの CAN FD チャンネルをサポート
- ISO11898-1: 2015 規格に準拠
- CXPI (CYT2B9 のみ搭載しています。CYT2B7 では搭載していません。)
- 暗号エンジン
- ASIL-B の機能安全
- 2.7 V ~ 5.5 V の低電力動作
- デバッグインタフェース
- IEEE-1149.1-2001 準拠の JTAG コントローラおよび SWD プロトコル
 - JTAG I/F および SWD からのフラッシュプログラミング
- 業界標準ツールとの互換性
 - コードの開発とデバッグのための Green Hills Software (GHS) MULTI または IAR Embedded Workbench for Arm (EWARM)
 - Cortex-M4 プロセッサ用 Arm Embedded Trace Macrocell (ETM) をサポート
- パッケージラインアップ: LQFP-64/80/100/144/176 ピン

2.2 CYT4B シリーズ

CYT4B シリーズは、ハイエンドボディコントロールユニットなどの車載システム向けの Traveo II MCU です。このデバイスには、メイン処理用に 2 つの CM7 と、周辺およびセキュリティ処理用の CM0+ CPU を搭載しており、高度な 40nm プロセスで製造されています。このデバイスは、CAN FD, LIN, ギガビットイーサネット, FlexRay などのボディコントロール機能をサポートしています。この製品は、安全なコンピューティングプラットフォームを可能にし、サイプレスの低消費電力フラッシュメモリと、複数の高性能アナログおよびデジタル機能を組み込んでいます。図 2 に CYT4BF のブロックダイアグラムを示します。

図 2. CYT4BF のブロックダイアグラム



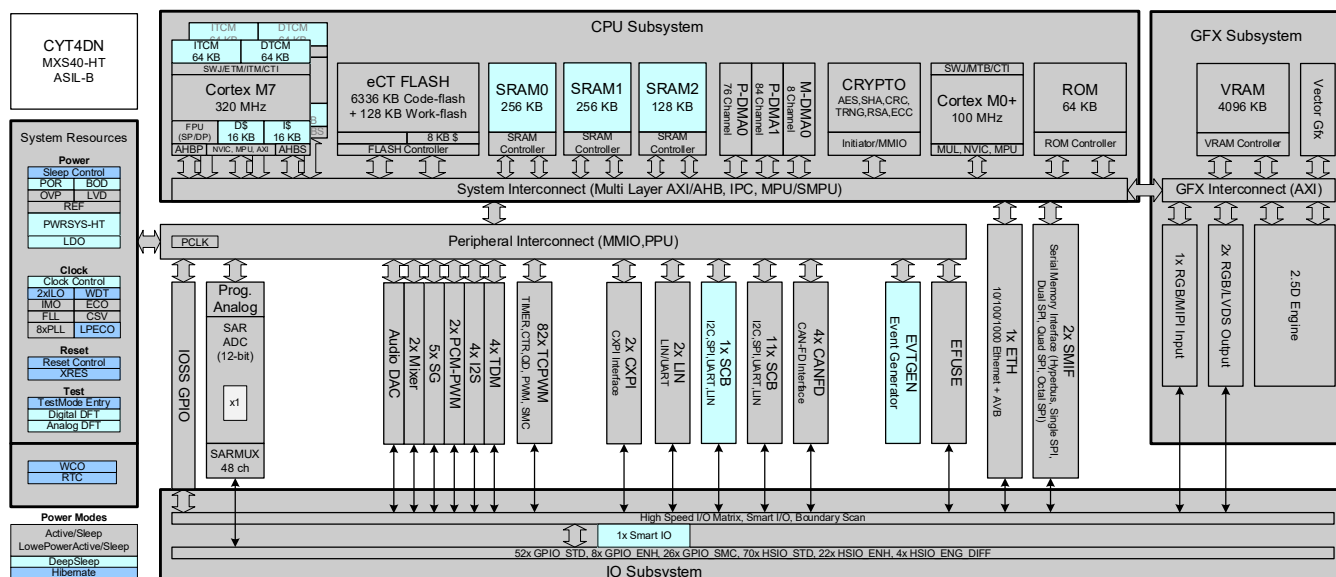
2.2.1 機能概要

- デュアル CPU サブシステム
 - 2 つの 350 MHz 32 ビット Arm Cortex-M7 CPU には、それぞれシングルサイクル乗算、シングル/ダブル浮動小数点演算ユニット(FPU)、メモリ保護ユニット(MPUs)付き
 - 100 MHz 32 ビット CM0+ MPU 付き
- 搭載メモリ
 - 最大 8384 KB (8128 KB + 256 KB) のコードフラッシュと最大 256 KB (192 KB + 64 KB) のワークフラッシュ
 - RWW (Read-While-Write) により、コードフラッシュから実行中にコードフラッシュ/ワークフラッシュを更新可能
 - シングルおよびデュアルバンクモードをサポート
- CAN FD
 - 従来の CAN と比較してデータレートが向上、ただし、物理層トポロジとデバイス (トランシーバ) による制限あり
 - 最大 8 Mbps をサポート
 - 最大 10 の CAN FD チャンネルをサポート
 - ISO11898-1: 2015 規格に準拠
- イーサネット MAC
 - 最大 2 ch × 10/100/1000
- FlexRay
 - 1 インタフェース : ch A/ch B (オプション)
- SMIF (Serial Memory Interface)
 - HyperBus, Single SPI, Dual SPI, Quad SPI, Octal SPI
- SDHC
 - embedded MultiMedia Card (eMMC), または Secure Digital (SD), または SDIO (Secure Digital Input Output)
- オーディオ
 - 3 I²S (Inter-IC Sound) インタフェース/TDM (Time Division Multiplexing) インタフェース
- 暗号エンジン
- ASIL-B の機能安全
- シングル電源供給: V_{DD}=2.7 V to 5.5 V (最大 300 mA) / デュアル電源供給: V_{DD}=2.7 V to 5.5 V and V_{CCD}=1.15 V (300 mA を超える場合)
- デバッグインタフェース
- IEEE-1149.1-2001 準拠の JTAG コントローラおよび SWD プロトコル
 - JTAG I/F および SWD からのフラッシュプログラミング
- 業界標準ツールとの互換性
 - コードの開発とデバッグのための GHS/MULTI または IAR EWARM
 - Cortex-M7 プロセッサ用 Arm Embedded Trace Macrocell (ETM) をサポート
- パッケージラインアップ : TEQFP-176 ピン, BGA-272, BGA-320

2.3 CYT4D シリーズ

CYT4D シリーズは、インストルメントクラスターやヘッドアップディスプレイ(HUD)などの車載システム向けの Traveo II MCU です。このデバイスには、2D グラフィクスエンジン、サウンド機能、メイン処理用に 2 つの Arm Cortex-M7 CPU と、周辺およびセキュリティ処理用の Cortex-M0+ CPU を搭載しています。このデバイスは、CAN FD、LIN、CXPI、ギガビットイーサネットなどの機能をサポートしており、高度な 40nm プロセスで製造されています。この製品は、安全なコンピューティングプラットフォームを可能にし、サイプレスの低消費電力フラッシュメモリと、複数の高性能アナログおよびデジタル機能を組み込んでいます。図 3 に CYT4DN のブロックダイアグラムを示します。

図 3. CYT4DN のブロックダイヤグラム



2.3.1 機能概要

■ グラフィクスサブシステム

- 2D および 2.5D(遠近法による 3D 効果)グラフィックスレンダリングをサポート
- 最大 RGB24 ビットカラー解像度
- 4096 KB エンベデッドビデオメモリ (VRAM)
- 2 つのディスプレイをサポートする最大 2 つのビデオ出力インターフェースは以下を含む
 - 1 パラレル RGB (最大表示サイズ: 1600 × 600)
 - 2 FPD-link/1 LVDS (最大表示サイズ: 1920 × 720)
 - 1 FPD-link/2 LVDS (最大表示サイズ: 2880 × 1080)
- ITU 656 またはパラレル RGB / YUV 入力用のビデオ入力処理用の 1 キャプチャーエンジンは以下のいずれか
 - RGB (最大キャプチャサイズ: 1600 × 600)
 - 最大ワイド HD 解像度ビデオ入力用 4 レーン MIPI CSI-2 インタフェース (最大キャプチャサイズ: 2 レーン 1920 × 720, 4 レーン 2880 × 1080)
 - ITU656 (標準カメラキャプチャ: 最大 800 × 480)

■ サウンドサブシステム

- デジタルオーディオ機器接続のための NXPI 社の I²S バス仕様に基づく 4 I²S インタフェースサポート
- 4 TDM インタフェース
- 2 パルス符号変調 - パルス幅変調 (PCM-PWM) インタフェース
- 最大 5ch サウンドジェネレータ (SG) インタフェース
- 5 入カストリームを備える 2ch PCM オーディオストリームミキサー

- 1 オーディオ D/A コンバータ (DAC)
- デュアル CPU サブシステム
 - 2 つの 320 MHz 32 ビット Arm Cortex-M7 CPU には、それぞれシングルサイクル乗算、シングル/ダブル浮動小数点演算ユニット(FPU)、メモリ保護ユニット(MPUs)付き
 - 100 MHz 32 ビット CM0+ MPU 付き
- 搭載メモリ
 - 最大 6336 KB (6080 KB + 256 KB) のコードフラッシュと最大 128 KB (96 KB + 32 KB) のワークフラッシュ
 - RWW (Read-While-Write) は、コードフラッシュ/ワークフラッシュをコードフラッシュ実行中に更新可能
 - シングルおよびデュアルバンクモードをサポート
- CAN FD
 - 従来の CAN と比較してデータレートが向上、ただし、物理層トポロジとデバイス (トランシーバ) による制限あり
 - 最大 8 Mbps をサポート
 - 最大 4 の CAN FD チャンネルをサポート
 - ISO11898-1: 2015 規格に準拠
- イーサネット MAC
 - 1 ch × 10/100/1000
- SMIF (Serial Memory Interface)
 - HyperBus, Single SPI, Dual SPI, Quad SPI, Octal SPI
- 暗号エンジン
- ASIL-B の機能安全
- シングル電源供給: $V_{DD0}=2.7\text{ V to }5.5\text{ V}$ (最大 300 mA) / デュアル電源供給: $V_{DD0}=2.7\text{ V to }5.5\text{ V}$ and $V_{CC0}=1.15\text{ V}$ (300 mA を超える場合)
- デバッグインタフェース
- IEEE-1149.1-2001 準拠の JTAG コントローラおよび SWD プロトコル
 - JTAG I/F および SWD からのフラッシュプログラミング
- 業界標準ツールとの互換性
 - コードの開発とデバッグのための GHS/MULTI または IAR EWARM
 - Cortex-M7 プロセッサ用 Arm Embedded Trace Macrocell (ETM) をサポート
- パッケージラインアップ: BGA-500

2.4 その他のシリーズ

その他のシリーズのブロックダイアグラムと機能概要については、[関連ドキュメント](#)のデバイスデータシートを参照してください。

3 開発環境とツール

3.1 評価ボード

CYTVII-B-E-1M/CYTVII-B-8M/CYTVII-C-2D-6M 評価ボードは、2 枚のボードを組み合わせで使用します。CPU ボード (CYTVII-B-E-1M-xxx-CPU/CYTII-B-H-xxx-CPU) は、ベースボード (CYTVII-B-E-BB) に接続します。CYTVII-C-2D-6M-xxx-CPU ボードはスタンドアロンで使用します。CPU ボードは、ベースボードとともに、ベースボード上で使用可能なすべての周辺機器にアクセスでき、完全な機能を備えた評価プラットフォームが実現します。CYTVII-B-E-1M-xxx-CPU/CYTII-B-H-8M-xxx-CPU/CYTVII-C-2D-6M-xxx-CPU には、デバッグインタフェース用の Cortex デバッグコネクタ, Cortex Debug + ETM コネクタ, Arm Standard JTAG, および Arm Mictor-38 コネクタがあります。この評価ボードは、デバイス性能評価とソフトウェア開発に使用されます。ボードの詳細については、評価ボードのユーザガイドを参照してください。

図 4. 評価ボード

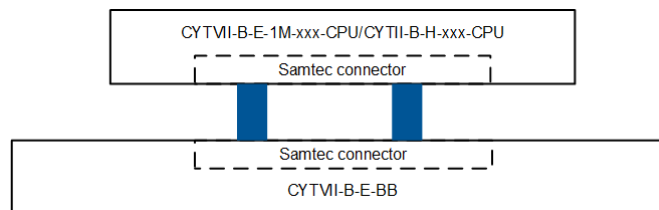


表 1. 評価ボード型格

ボード型格	CYT2B7/B9 シリーズ パッケージ(ピン数)
CYTVII-B-E-1M -176-CPU	176 ピン LQFP
CYTVII-B-E-1M -144-CPU	144 ピン LQFP
CYTVII-B-E-1M -100-CPU	100 ピン LQFP
CYTVII-B-E-1M -80-CPU	80 ピン LQFP
CYTVII-B-E-1M -64-CPU	64 ピン LQFP
ボード型格	CYT4B シリーズ パッケージ(ピン数)
CYTVII-B-H-8M -320-CPU	320 ボール BGA
CYTVII-B-H-8M -272-CPU	272 ボール BGA
CYTVII-B-H-8M-176-CPU	176 ピン TEQFP
ボード型格	CYT4D シリーズ パッケージ(ピン数)
CYTVII-C-2D-6M-500-CPU	500 ボール BGA
CYTVII-C-2D-6M-327-CPU	327 ボール BGA

3.1.1 ベースボードで使用可能な機能

表 2. CYTVII-B-E-BB の周辺機能一覧

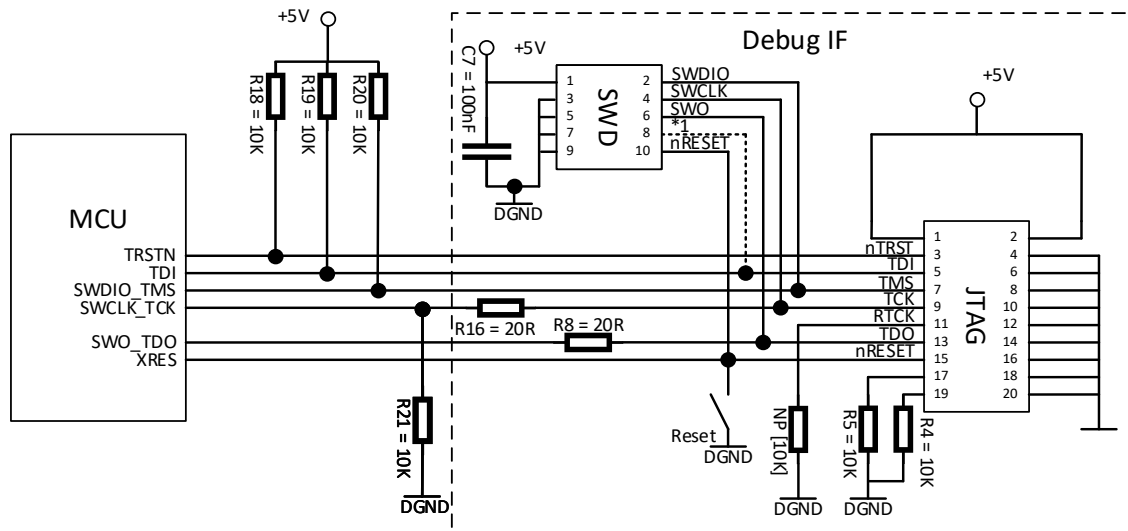
周辺機能	説明	チャネル数
CAN FD	コネクタ (D-sub 9 ピン) と トランシーバ (TJA1057GT, 6 チャネル と TJA1145T/FDJ, 4 チャネル)	10 チャネル
LIN	コネクタ (D-sub 9 ピン) と トランシーバ (TJA1021T/20)	6 チャネル
I ² C	ピンヘッダ	8 チャネル
CXPI	コネクタ (D-sub 9 ピン) と トランシーバ (S6BT112A)	1 チャネル
ADC	ピンヘッダ	72 チャネル

周辺機能	説明	チャンネル数
FlexRay	コネクタ (D-sub 9 ピン) と トランシーバ (AS8221)	2 チャンネル
Potentiometer	ADC 制御用 (Alps RK09K1130A8G)	1
User Button	ユーザ制御用	5 ボタン
User LEDs	緑色	10 LED

3.1.2 ツール接続図

CYT2, CYT3, および CYT4 シリーズ MCU には、デバッグツールに接続するための JTAG/SWD ポートがあります。SWD のデバッグピンは JTAG と共有され、TCPWM/SCB/LIN/CAN などの他の機能とも共有されます。詳細は [データシート](#) を参照してください。図 5 に CYT2, CYT3, および CYT4 シリーズの基本接続ダイアグラムの例を示します。

図 5. CYT2, CYT3, CYT4 シリーズの基本接続ダイアグラム



- 注意事項:

*1: SWDコネクタで JTAG接続する場合に必要

表 3. Cortex Debug+ETM コネクタ

ピン番号	信号名	ピン番号	信号名
1	VCC	2	SWDIO/TMS
3	GND	4	SWDCLK/TCK
5	GND	6	SWO/TDO/EXTa/TRACECTL
7	KEY	8	NC/EXTb/TDI
9	GNDDetect	10	nRESET
11	GND/TgtPwr+Cap	12	TRACECLK
13	GND/TgtPwr+Cap	14	TRACEDATA[0]
15	GND	16	TRACEDATA[1]
17	GND	18	TRACEDATA[2]
19	GND	20	TRACEDATA[3]

3.2 サンプルドライバライブラリ

サイプレスはサンプルソフトウェアとしてスタートアップを含むサンプルドライバライブラリ (SDL) を提供しています。SDL は、さまざまな周辺機器にアクセスするためのシンプルなインタフェースを提供し、システム検証、ハードウェア起動、ベンチマーク、実現性検討、およびデモに使用されます。また、公式の AUTOSAR 製品でカバーされていないドライバの参考資料としても役立ちます。SDL は、いかなる自動車規格にも適合しないため、量産目的の製品には使用できません。SDL は、デバイスヘッダファイル、スタートアップコード、ペリフェラルドライバを同梱しています。SDL には、デバイス固有のリソースにアクセスするための API を提供する一連のファームウェアドライバが含まれています。

3.2.1 SDL 構造

SDL パッケージには、スタートアップ、デバイスヘッダファイル、各ペリフェラル用の API を提供するドライバとそのサンプル、GHS および IAR ツールチェーンの開発ツールサポートファイルが含まれています。例として CYT2B の SDL プロジェクトの構造を次に示します。

TVII_Sample_Driver_Library_revision	Top folder
—doc	Documents folder
—common	Common for TVII-B-E-1M/2M, TVII-B-H-4M/8M, TVII-C-2D-6M folder
hdr	
cmsis	
include	
cmsis_armcc.h	CMSIS compiler ARMCC (Arm Compiler 5) header file
cmsis_armclang.h	CMSIS compiler armclang (Arm Compiler 6) header file
cmsis_compiler.h	CMSIS compiler generic header file
cmsis_gcc.h	CMSIS compiler GCC header file
cmsis_ghs.h	CMSIS GHS specific definitions
cmsis_iccarm.h	CMSIS compiler ICCARM (IAR Compiler for Arm) header file
cmsis_version.h	CMSIS Core(M) Version definitions
core_cm0plus.h	CMSIS Cortex-M0+ Core Peripheral Access Layer Header File
core_cm4.h	CMSIS Cortex-M4 Core Peripheral Access Layer Header File
core_cm7.h	CMSIS Cortex-M7 Core Peripheral Access Layer Header File
mpu_armv7.h	CMSIS MPU API for Armv7-M MPU
—cy_project.h	Project-specific header file
src	
drivers	
canfd	CAN FD API
cy_canfd.c	CAN FD source file
cy_canfd.h	CAN FD header file
xxx	xxx API
cy_xxx.c	xxx source file
cy_xxx.h	xxx header file
mw	
button	
cy_button.c	Button middleware source file
cy_button.h	Button middleware header file
semihosting	
cy_semihosting.c	Semihosting middleware source file
cy_semihosting.h	Semihosting middleware header file
sw_timer	
cy_sw_tmr.c	Software Timer middleware source file
cy_sw_tmr.h	Software Timer middleware header file
flash	
cy_mw_flash.c	Simplified Flash API source file
cy_mw_flash.h	Simplified Flash API header file
startup	
ghs	GHS
startup_cm0plus.arm	

	startup_cm4.arm	
	startup_cm7.arm	
	iar	IAR
	startup_cm0plus.s	
	startup_cm4.s	
	startup_cm7.s	
	startup.c	C startup code
	startup_customize.h	C startup code before main()
tvii-be1m		TVII-B-E-1M folder
hdr		
	ip	CANFD IP Register Definitions
	cyip_canfd.h	CANFD IP definitions
	cyip_xxxxx.h	xxxxx IP definitions
	:	
	mcureg	
	cyreg_canfd.h	CANFD register definition header
	cyreg_xxxxx.h	xxxxx register definition header
	:	
src		
	drivers	
	adc	
	cy_adc.c	ADC source file
	cy_adc.h	ADC header file
	cpu	
	flash	
	sysclk	
	sysfit	
	examples	
	adc	
	canfd	
	:	
	system	
	system_cyt2b7.h	
	system_tvii-be1m_cm0plus.c	
	system_tvii-be1m_cm4.c	
	cy_interrupt_map.c	System interrupt handlers
	cy_interrupt_map_cm0plus.h	System interrupt handlers CM0+ header file
	cy_interrupt_map_cm4.h	System interrupt handlers CM4 header file
	main_cm0plus.c	Main file for CM0+
	main_cm4.c	Main file for CM4
tools		
	ghs	GHS MULTI workspaces
	_gpj	
	cym0plus.gpj	
	cm4.gpj	
	crypto_client.gpj	
	crypto_library.gpj	
	crypto_server.gpj	
	debugging.gpj	
	xxxx.gpj	
	debugging	For debugging
	tvii_debug.py	
	tvii-be1m.con	
	tvii-be1m.ghpcfg	
	flash	For Flash execution
	cm0plus	
	cm4	
	cm0plus_only.ghsmc	
	cm0plus_with_cm4.ghsmc	
	sram	For RAM execution

				cm0plus	
				cm4	
				cm0plus_only.ghsmc	
				cm0plus_with_cm4.ghsmc	
			iar		IAR workspaces
			debugging		
			flash		
			setting		
			sram		
			└─tviibe2m		TVII-B-E-2M folder
			└─tviibe4m		TVII-B-E-4M folder
			└─tviibe512k		TVII-B-E-512KB folder
			└─tviibe4m		TVII-B-H-4M folder
			└─tviibh8m		TVII-B-H-8M folder
			└─tviic2d4m		TVII-C-2D-4M folder
			└─tviic2d6m		TVII-C-2D-6M folder

3.2.2 CYT2B シリーズの CPU (CM0+および CM4) スタートアップシーケンス

以下の手順でスタートアップシーケンスを説明します。

1. システムリセット (@0x0000 0000)
2. CM0+が ROM boot を実行 (@0x0000 0004)
 - i. トリミングを適用
 - ii. eFuse および supervisory flash 内のデバッグアクセスポート (DAP) アクセス制限とシステム保護設定を適用
 - iii. フラッシュブートを認証し (SECURE life-cycle stage), 制御をフラッシュブートに移行
3. CM0+がフラッシュブートを実行 (Supervisory flash@0x1700 2000)
 - i. デバッグピンを SWD/JTAG 仕様に従って構成
4. CM0+がユーザアプリケーションを実行開始
 - i. CM0+ベクタテーブルを SRAM に移動 (CM0+ベクタテーブルベースを更新)
 - ii. CM4_VECTOR_TABLE_BASE (@ 0x4020 0200) をフラッシュに記述されている CM4 ベクタテーブルに設定 (CM4 リンカ定義ファイルで指定)
 - iii. CM4 の電源を起動
 - iv. CM0+ユーザアプリケーションを実行
5. CM4 が ROM ブートを実行 (@ 0x0000 0004)
 - i. CM4 はそのリセットハンドラに分岐
 - ii. CM4 ユーザアプリケーションの実行を継続

3.2.3 CYT4B/CYT4D シリーズの CPU (CM0+および CM7) スタートアップシーケンス

以下の手順でスタートアップシーケンスを説明します。

1. システムリセット (@0x0000 0000)
2. CM0+が ROM boot を実行 (@0x0000 0004)
 - i. トリミングを適用
 - ii. eFuse および supervisory flash 内のデバッグアクセスポート (DAP) アクセス制限とシステム保護設定を適用
 - iii. フラッシュブートを認証し (SECURE life-cycle stage), 制御をフラッシュブートに移行
3. CM0+がフラッシュブートを実行 (Supervisory flash@0x1700 2000)
 - i. デバッグピンを SWD/JTAG 仕様に従って構成
4. CM0+がユーザアプリケーションを実行開始
 - i. CM0+ベクタテーブルを SRAM に移動 (CM0+ベクタテーブルベースを更新)
 - ii. メモリサブシステムの WAIT を設定

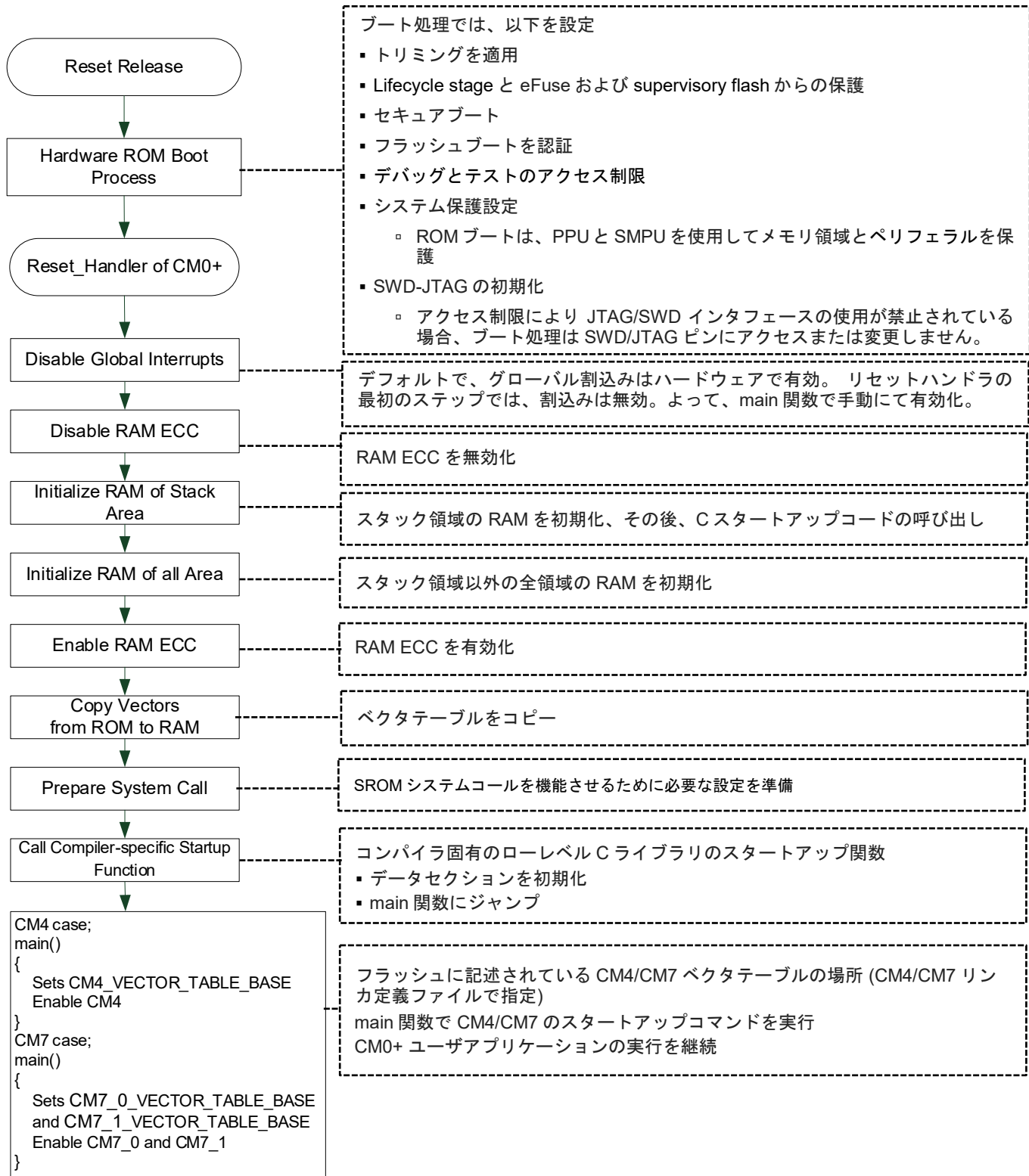
- iii. ルートクロックを設定し、コア外部電源 (PMIC)、グラフィックサブシステムなどを有効化
 - iv. CM7_0 (CLK_HF1) と CM7_1 (CLK_HF1)のクロックを設定
 - v. CM7_0 (CM7_0_VECTOR_TABLE_BASE @0x4020 0200) と CM7_1 (CM7_1_VECTOR_TABLE_BASE @0x4020 0600) をフラッシュに記述されているベクタテーブルに設定(リンカ定義ファイルで指定)
 - vi. CM7_0 と CM7_1 の両 CPU コアの電源を起動
 - vii. CPU_WAIT を無効化
 - viii. CM0+ユーザアプリケーションを実行
5. CM7_0 および/または CM7 はコードフラッシュまたは SRAM から直接実行
- i. CM7_0/CM7_1 はそのリセットハンドラに分岐
 - ii. CM7_0/CM7_1 ユーザアプリケーションの実行を継続

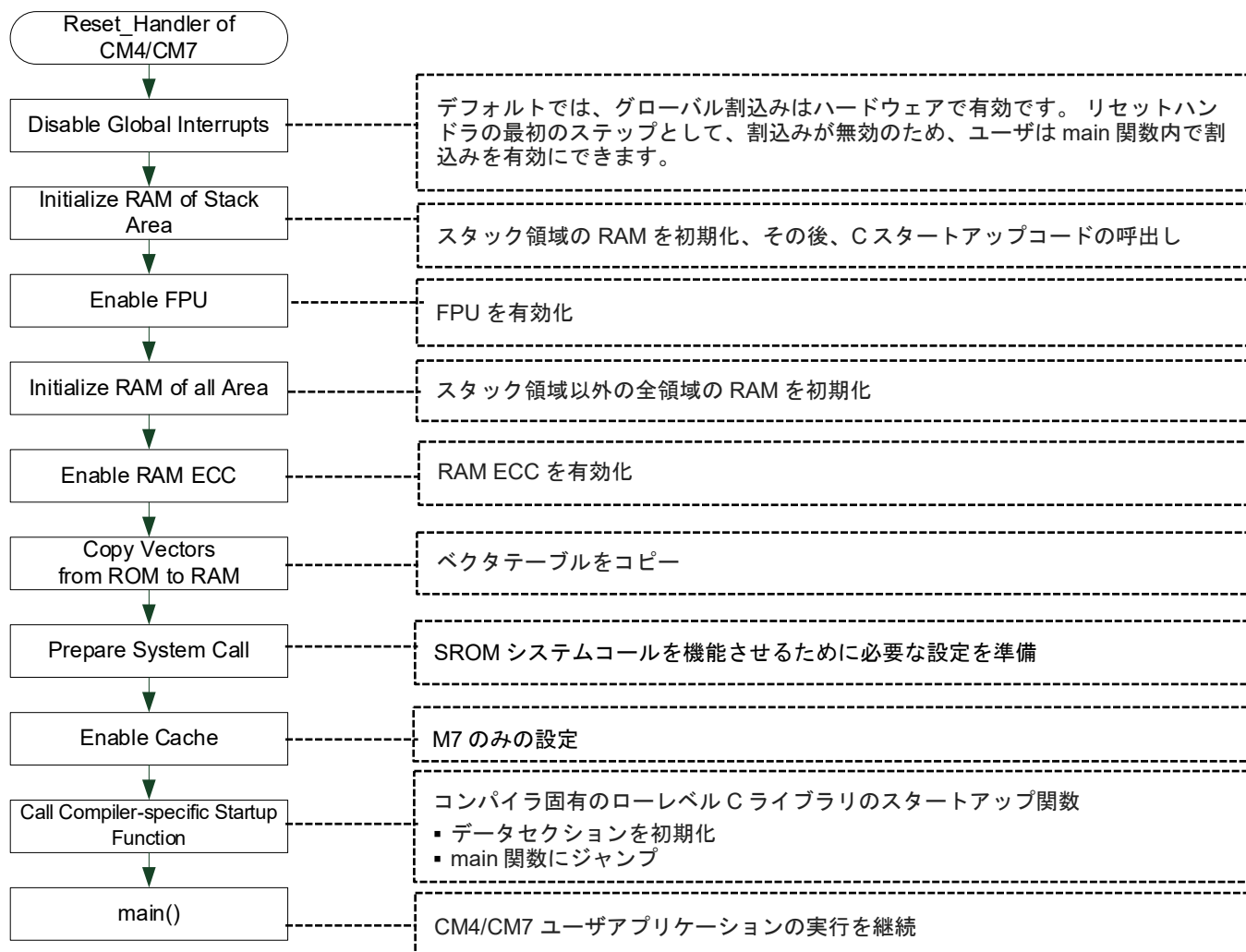
3.2.4 スタートアップフローの例

スタートアップフロー処理は、ブート処理の完了後に実行されます。ブート処理では、lifecycle stage に応じて、主に protection states やデバッグアクセス制限の設定、アプリケーション認証を行います。スタートアップ処理では、スタック設定, RAM 初期化, クロック設定のプログラム実行に必要な初期設定を行い、メインルーチンにジャンプします。リセット後、CM0+でブート処理が開始されます。CM4/CM7 は CM0+ユーザアプリケーションによるブート処理後にスタートアップされます。

図 6. デバイススタートアップフロー

ブート処理の実行





3.2.5 サンプルコード

図 7. CM0+スタートアップのサンプルコード

```

;*****
;* Start-up Code
;*****
        .weak Reset_Handler
        .section ".text", "ax"
        .thumb

Reset_Handler:
_start:

; Disable global interrupts
        CPSID    I

; Update Vector Table Offset Register with address of user ROM table
; (will be updated later to user RAM table address in C startup code)
        LDR    r0, =__vector_table
        LDR    r1, =VTOR
        STR    r0, [r1]
        DSB

; CM0+ bus width is 32-bit, but SRAM is built with 64-bit based ECC on Traveo II parts
; with CM7 core
; Set CPUSS->RAMx_CTL0.ECC_CHECK_DIS bits to avoid causing unintentional ECC faults during
; startup while SRAM ECC has not been initialized yet
; Generic code can be used, even if RAMx_CTL0 (x > 0) registers are not implemented in a
; device
; or if no ECC_CHECK_DIS bits are available in the registers in case of m4cpuss with 32-
; bit ECC SRAM
        MOVS    r0, #1
        LSLS    r0, r0, #19
        LDR    r1, =CPUSS_RAM0_CTL0
        LDR    r2, [r1]
        ORRS    r2, r0
        STR    r2, [r1]
        LDR    r1, =CPUSS_RAM1_CTL0
        LDR    r2, [r1]
        ORRS    r2, r0
        STR    r2, [r1]
        LDR    r1, =CPUSS_RAM2_CTL0
        LDR    r2, [r1]
        ORRS    r2, r0
        STR    r2, [r1]

; Initialize ECC of startup stack (needed for local variables in C startup code) by
; processing 8 bytes per loop iteration,
; because the ECC initialization feature uses this generic granularity that will cover
; any memory (SRAM/TCM) in any TVII device
; Prerequisite: Stack Pointer (SP) has not been modified (from the vector table init
; value) by above code (otherwise code must be adapted)
        MOVS    r0, #0 ; clear value
        MOVS    r1, #0 ; clear value
        LDR    r2, Cy_u32StartupStackStartAddress
startup_stack_ecc_init_loop:
        STM     r2!, {r0, r1}
        CMP     r2, sp
        BNE     startup_stack_ecc_init_loop

; Call C startup code (no ANSI C context established yet!)
        LDR     r7, =Startup_Init
        BLX     r7

; Call system library
        LDR     r5, pool_myaddr
        SUB     r5, r5, 1 ; unset the low order bit
myaddr:

```



```

      MOV    r2, pc
      SUB    r5, r2, r5                ; Now r5 contains picbase

      MOV    r0, 0
      MOV    fp, r0                  ; * No previous frame *

      LDR    r0, pool_baseptrs
      ADD    r0, r0, r5 ; * Add in picbase *
      MOV    r1, 0 ; * Clear argc reg *
      MOV    r2, 0 ; * Clear argv reg *
      MOV    r3, 0 ; * Clear envp reg *
      MOV    r4, 0 ; * Clear global regs *
      MOV    r5, 0
      MOV    r6, 0

      LDR    r7, =__ghs_ind_crt0
      BLX    r7

      ; Note: Control flow does not necessarily return here.
      ; On some tool-chains (e.g. IAR) control flow will never return from
      ; the system library.
      Cy_Main_Exited:
          B    Cy_Main_Exited

      .endif _start
  
```

図 8. CM0+ main 関数のサンプルコード

```

int main(void)
{
    __enable_irq(); /* Enable global interrupts. */
    /* Enable CM4.  CY_CORTEX_M4_APPL_ADDR must be updated if CM4 memory layout is changed. */
    Cy_SysEnableCM4(CY_CORTEX_M4_APPL_ADDR);

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    Cy_GPIO_Pin_Init(USER_LED0_PORT, USER_LED0_PIN, &user_led0_port_pin_cfg);

    for(;;)
    {
        DELAY(150000);
        Cy_GPIO_Inv(USER_LED0_PORT, USER_LED0_PIN);
    }
}
  
```

図 9. CM4 スタートアップのサンプルコード

```

;*****
; * Start-up Code
;*****
        .weak Reset_Handler
        .section ".text", "ax"
        .thumb2

Reset_Handler:
_start:

    ; Disable global interrupts
        CPSID    I

    ; Update Vector Table Offset Register with address of user ROM table
    ; (will be updated later to user RAM table address in C startup code)
        LDR    r0, =__vector_table
        LDR    r1, =VTOR
        STR    r0, [r1]
        DSB
  
```

```

; Initialize ECC of startup stack (needed for local variables in C startup code) by
; processing 8 bytes per loop iteration,
; because the ECC initialization feature uses this generic granularity that will cover
; any memory (SRAM/TCM) in any TVII device
; Prerequisite: Stack Pointer (SP) has not been modified (from the vector table init
; value) by above code (otherwise code must be adapted)
    MOVS r0, #0 ; clear value
    MOVS r1, #0 ; clear value
    LDR r2, Cy_u32StartupStackStartAddress
startup_stack_ecc_init_loop:
    STRD r0, r1, [r2], #8
    CMP r2, sp
    BNE startup_stack_ecc_init_loop

; Call C startup code (no ANSI C context established yet!)
    LDR r7, =Startup_Init
    BLX r7

; Call system library
    LDR r5, pool_myaddr
    SUB r5, r5, 1 ; unset the low order bit
myaddr:
    MOV r2, pc
    SUB r5, r2, r5 ; Now r5 contains picbase

    MOV r0, 0
    MOV fp, r0 ; * No previous frame *

    LDR r0, pool_baseptrs
    ADD r0, r0, r5 ; * Add in picbase *
    MOV r1, 0 ; * Clear argc reg *
    MOV r2, 0 ; * Clear argv reg *
    MOV r3, 0 ; * Clear envp reg *
    MOV r4, 0 ; * Clear global regs *
    MOV r5, 0
    MOV r6, 0

    LDR r7, =__ghs_ind_crt0
    BLX r7

; Note: Control flow does not necessarily return here.
; On some tool-chains (e.g. IAR) control flow will never return from
; the system library.
Cy_Main_Exited:
    B Cy_Main_Exited

.endf _start

```

図 10. CM7 スタートアップのサンプルコード

```

;*****
; * Start-up Code
;*****
    .weak Reset_Handler
    .section ".text", "ax"
    .thumb2

Reset_Handler:
_start:

; Disable global interrupts
    CPSID I

; Update Vector Table Offset Register with address of user ROM table
; (will be updated later to user RAM table address in C startup code)
    LDR r0, =__vector_table
    LDR r1, =VTOR

```

```

        STR    r0, [r1]
        DSB

; Allow write access to Vector Table Offset Register and ITCM/DTCM configuration register
(CPUSS_CM7_X_CTL.PPB_LOCK[3] and CPUSS_CM7_X_CTL.PPB_LOCK[1:0])
        LDR    r1, =CPUSS_IDENTITY          ; first check master identity (CM7_0 or CM7_1?)
        LDR    r0, [r1]
        LSR    r0, r0, #8
        AND    r0, r0, #0xf
        CMP    r0, #MASTER_ID_CM7_0
        ITE    EQ                                ; load address of corresponding CPUSS_CM7_x_CTL
register
        LDREQ  r1, =CPUSS_CM7_0_CTL
        LDRNE  r1, =CPUSS_CM7_1_CTL
        LDR    r0, [r1]
        BIC    r0, r0, #0xb
        STR    r0, [r1]
        DSB

; Enable ITCM and DTCM
        LDR    r1, =ITCMCR
        LDR    r0, [r1]
        ORR    r0, r0, #0x7 ; Set ITCMCR.EN, .RMW and .RETEN fields
        STR    r0, [r1]

        LDR    r1, =DTCMCR
        LDR    r0, [r1]
        ORR    r0, r0, #0x7 ; Set DTCMCR.EN, .RMW and .RETEN fields
        STR    r0, [r1]

        DSB
        ISB

; Initialize ECC of startup stack (needed for local variables in C startup code) by
processing 8 bytes per loop iteration,
; because the ECC initialization feature uses this generic granularity that will cover
any memory (SRAM/TCM) in any TVII device
; Prerequisite: Stack Pointer (SP) has not been modified (from the vector table init
value) by above code (otherwise code must be adapted)
        MOVS   r0, #0 ; clear value
        MOVS   r1, #0 ; clear value
        LDR    r2, Cy_u32StartupStackStartAddress
startup_stack_ecc_init_loop:
        STRD   r0, r1, [r2], #8
        CMP    r2, sp
        BNE    startup_stack_ecc_init_loop

; Call C startup code (no ANSI C context established yet!)
        LDR    r7, =Startup_Init
        BLX    r7

; Call system library
        LDR    r5, pool_myaddr
        SUB     r5, r5, 1                                ; unset the low order bit
myaddr:
        MOV    r2, pc
        SUB     r5, r2, r5                                ; Now r5 contains picbase

        MOV    r0, 0
        MOV    fp, r0                                ; * No previous frame *

        LDR    r0, pool_baseptrs
        ADD    r0, r0, r5 ; * Add in picbase *
        MOV    r1, 0 ; * Clear argc reg *
        MOV    r2, 0 ; * Clear argv reg *
        MOV    r3, 0 ; * Clear envp reg *
        MOV    r4, 0 ; * Clear global regs *
        MOV    r5, 0
        MOV    r6, 0

```

```

    LDR r7, =__ghs_ind_crt0
    BLX r7

; Note: Control flow does not necessarily return here.
; On some tool-chains (e.g. IAR) control flow will never return from
; the system library.
Cy_Main_Exited:
    B    Cy_Main_Exited

.endf _start

```

3.3 ペリフェラルドライバ

ペリフェラルドライバは、再利用可能なハードウェアにアクセスするための API を提供する一連のファームウェアドライバです。これらの API は、各ペリフェラルの初期化および制御を行います。

3.3.1 ペリフェラルドライバー一覧

表 4 に、サイプレスが SDL 用に提供するペリフェラルドライバを示します。

表 4. ペリフェラルドライバー一覧

項	モジュール	説明	Traveo II ファミリ		
			CYT2B	CYT3B/4B	CYT4D
1	ADC	Analog-to-Digital Converter	○	○	○
2	AudioSS	サウンドシステム: I2S, DAC, Mixer, PWM, SG, TDM	-	-	○
3	AXIDMA	AXI バス上の M-DMA	-	-	○
4	CANFD	CAN FD	○	○	○
5	CPU	CPU コア機能のイネーブル設定	○	○	○
6	CRYPTO	Cryptography	○	○	○
7	CXPI	CXPI インタフェースの通信設定	○	-	○
8	DMA	M-DMA, P-DMA	○	○	○
9	ETHERNET	自動車用およびギガビットイーサネット PHY をサポートする基本のイーサネットドライバ	-	○	○
10	EVTGEN	Event generator	○	○	○
11	FLASH	Code Flash, Work Flash	○	○	○
12	FLEXRAY	FlexRay 通信設定	-	○	-
13	FPDLINK	FPD-Link あるいは LVDS	-	-	○
14	GPIO	GPIO	○	○	○
15	GFX_ENV	グラフィックスの環境設定	-	-	○
16	I²S	Inter-IC サウンド管理。I2S は、オーディオコーデックやシンプル DAC などの外部 I2S デバイスとデジタルオーディオストリーミングデータの送受信を管理	-	○	-
17	IPC	Inter-Processor Communication	○	○	○
18	LIN	Local Interconnect Network	○	○	○
19	LVD	Low-Voltage-Detection	○	○	○
20	MCWDT	Multi-Counter Watchdog Timers	○	○	○
21	MIPICSI2	ビデオ入力	-	-	○
22	MPU	Memory Protection Unit	○	○	○

項	モジュール	説明	Traveo II ファミリ		
			CYT2B	CYT3B/4B	CYT4D
23	PROT	Protection Unit	○	○	○
24	SCB	Serial Communication Block (SCB) - EZI2C, SCB - I2C, SCB - SPI, SCB - UART	○	○	○
25	SD_HOST	SD および eMMC デバイス設定	-	○	-
26	SMART IO	デバイスポート上の GPIO (ピン) と HSIOM (マルチブレックスピン) の間のスマート I/O ハードウェアのアクセス設定により、周辺機器と GPIO 信号を GPIO ポートでシンプルなロジック操作で実行可能	○	○	○
27	SMIF	外部メモリデバイスを Traveo II に接続するための SPI ベースの通信インタフェース。SMIF は、Octal-SPI, Dual Quad-SPI, Quad-SPI, DSPI, および SPI をサポート。さらに、HyperRAM や HyperFlash デバイスなどの Hyper Bus インタフェースもサポート	-	○	○
28	SROM	Internal SROM driver	○	○	○
29	SYSCLK	System Clock	-	○	○
30	SYSFLT	System Fault	○	○	○
31	SYSINT	System Interrupt	○	○	○
32	SYSLIB	System Library	○	○	○
33	SYSPM	System Power Management	○	○	○
34	SYSREGHC/ SYSPMIC	REGHC/PMIC Control and Status	-	○	-
35	SYSRESET	リセット要因を読み出し、クリアするための API を提供	○	○	○
36	SYSRTC	System Real-Time Clock	○	○	○
37	SYSTICK	SysTick Timer	○	○	○
38	SYSWDT	フリーラン Watchdog Timer	○	○	○
39	TCPWM	Timer Counter PWM	○	○	○
40	TRIGMUX	Trigger multiplexer	○	○	○

3.4 GHS MULTI を使用した SDL の使用例

3.4.1 SDL の環境

ここでは、CYT2B7 シリーズ SDL の操作例を示します。

SDL の動作環境:

- CPU: CYT2B7 シリーズ
- ターゲットボード: Traveo II Evaluation board
- IDE: GHS MULTI V7.1.4 (compiler version 2017.1.4)
- API モジュール: GPIO

3.4.2 GHS MULTI と SDL のインストール

GHS MULTI のインストール、ライセンスの設定、およびプローブファームウェアの設定を事前に行ってください。

SDL のガイドに従って SDL を事前にインストールしてください。

3.4.3 デバッグスクリプトとコンフィグレーションファイルの設定

以下のフォルダにあるスクリプトファイル *tvii_detect.py* と *multi.irc* を使い実行します。

`C:\...\TVII_Sample_Driver_Library_revision\misc\tools\ghs\debugging\AppData_GHS`

このプロセスには、プローブの設定、ターゲットとの接続、およびプログラムの RAM へのダウンロードが含まれます。

デバッグスクリプトの検出 (D) ボタンを使用してプログラムを RAM (R) にロードできます。

ファイル *tvii_detect.py* と *multi.irc* を PC の `<%APPDATA%\GHS>` フォルダ (つまり、`C:\Users\<LOGIN_NAME>\AppData\Roaming\GHS`) にコピーします。

AppData フォルダが見つからない場合は、次の手順に従って隠しフォルダを表示してください。

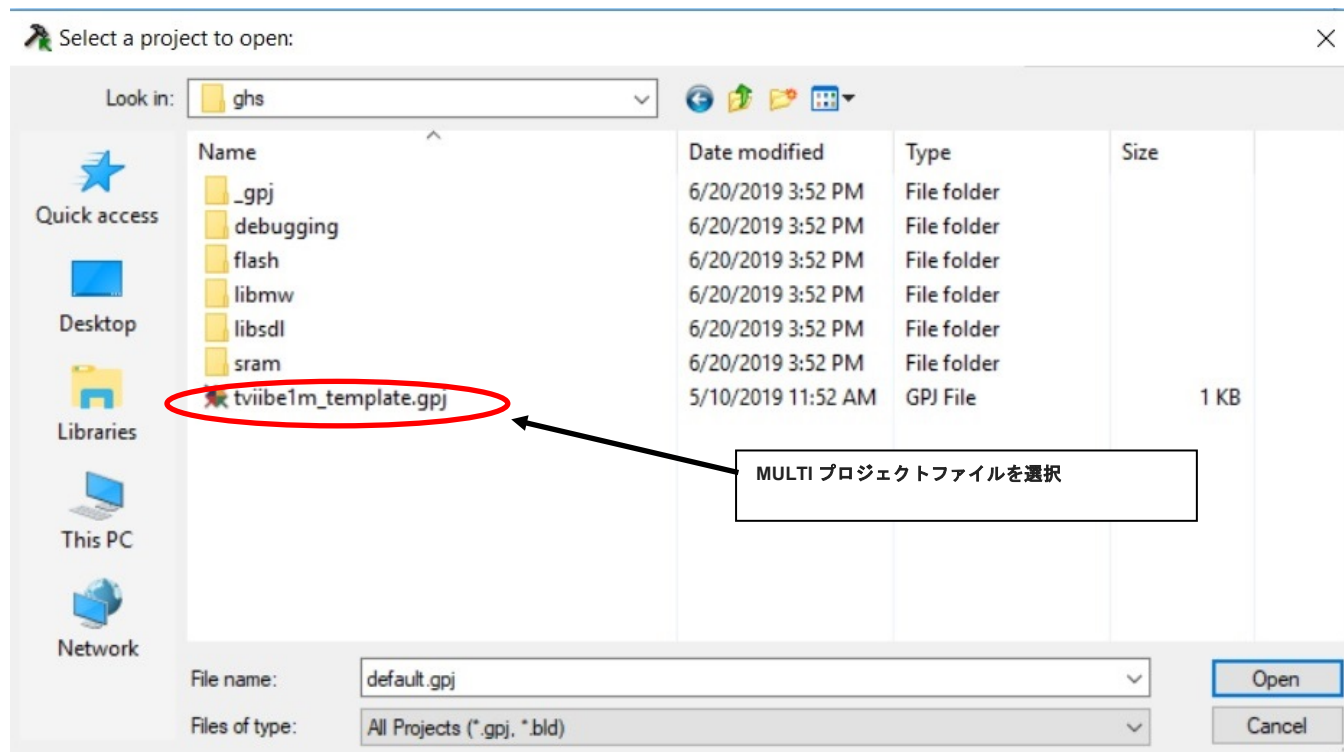
1. Windows のエクスプローラで **View** タブをクリック
2. **Option** をクリック
3. フォルダーオプションダイアログにて **View** タブをクリック
4. **Show hidden files, folders, and drivers** を選択
5. **OK** をクリック

3.4.4 MULTI によるセットアップ

1. MULTI プロジェクトマネージャを起動し、SDL プロジェクトのプロジェクトファイルを開きます。

C:\...\TVII_Sample_Driver_Library_revision\tools\ghs\tviibe1m_template.gpj

図 11. SDL プロジェクトの選択



2. ビルドの場合、*sram.gpj* を選択して図 12 に示すように **hammer** アイコンをクリックします。ビルドが開始され、正常終了のメッセージが図 13 のように表示されます。

図 12. SDL プロジェクトのビルド

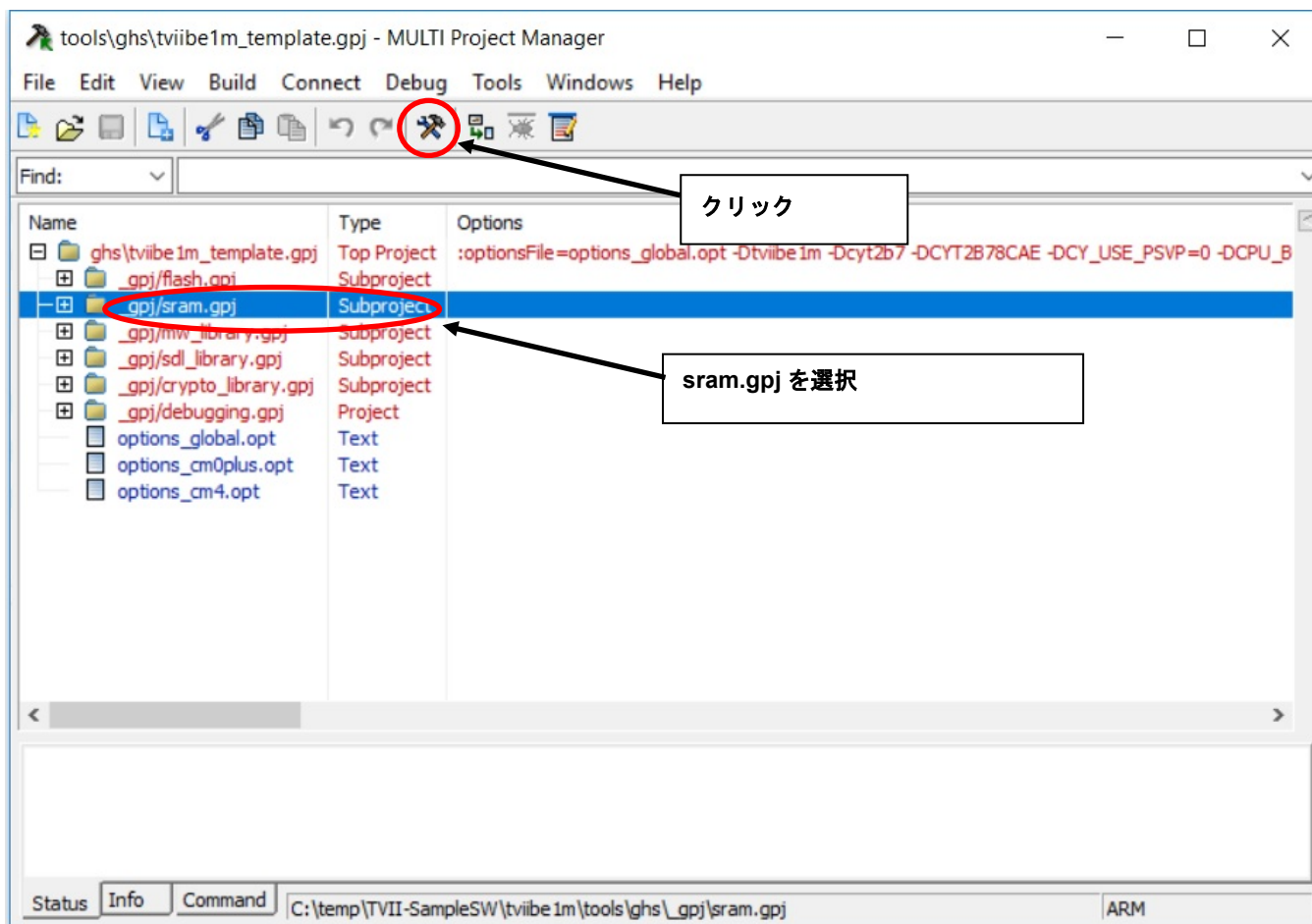
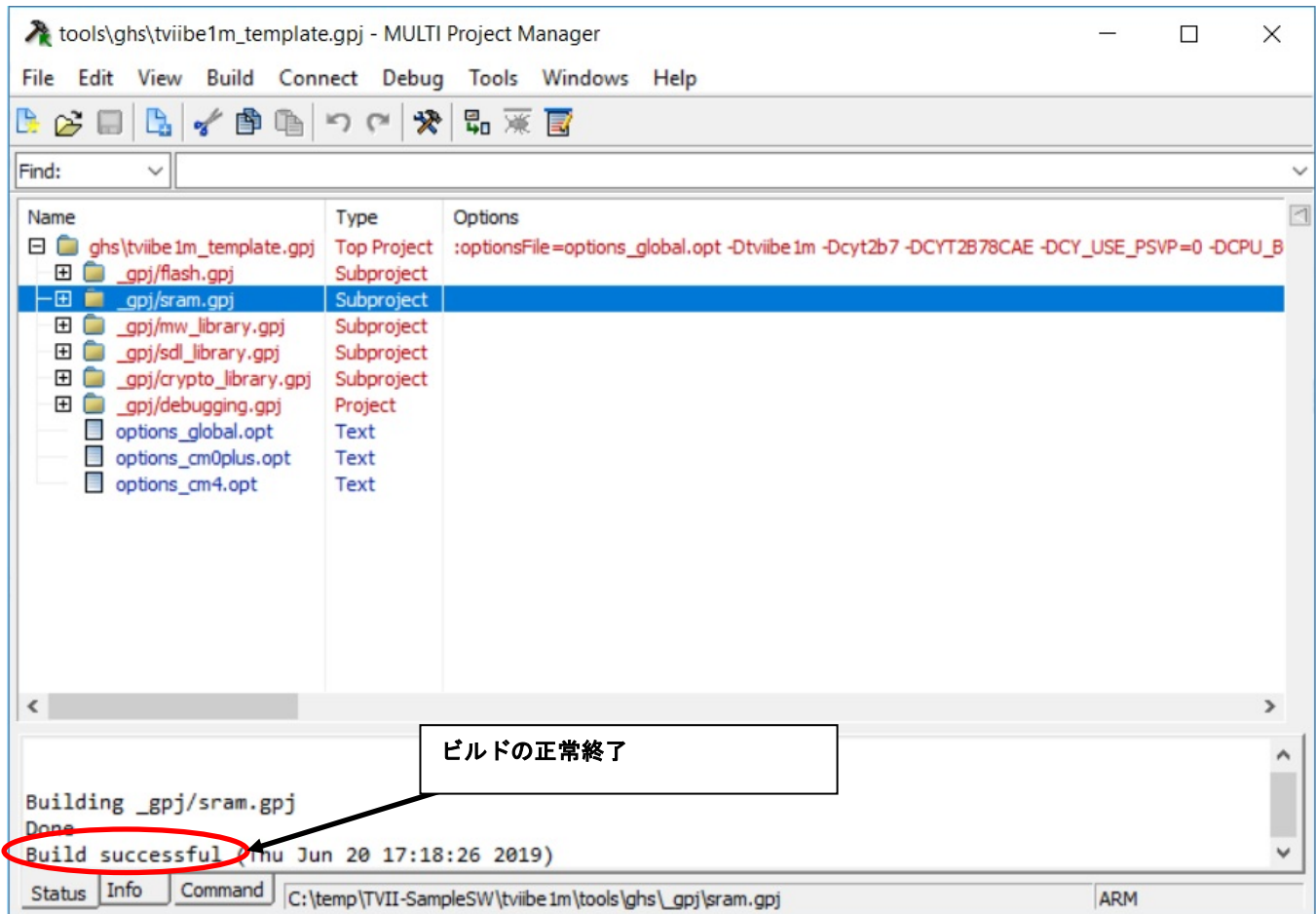


図 13. SDL プロジェクトのビルドの正常終了



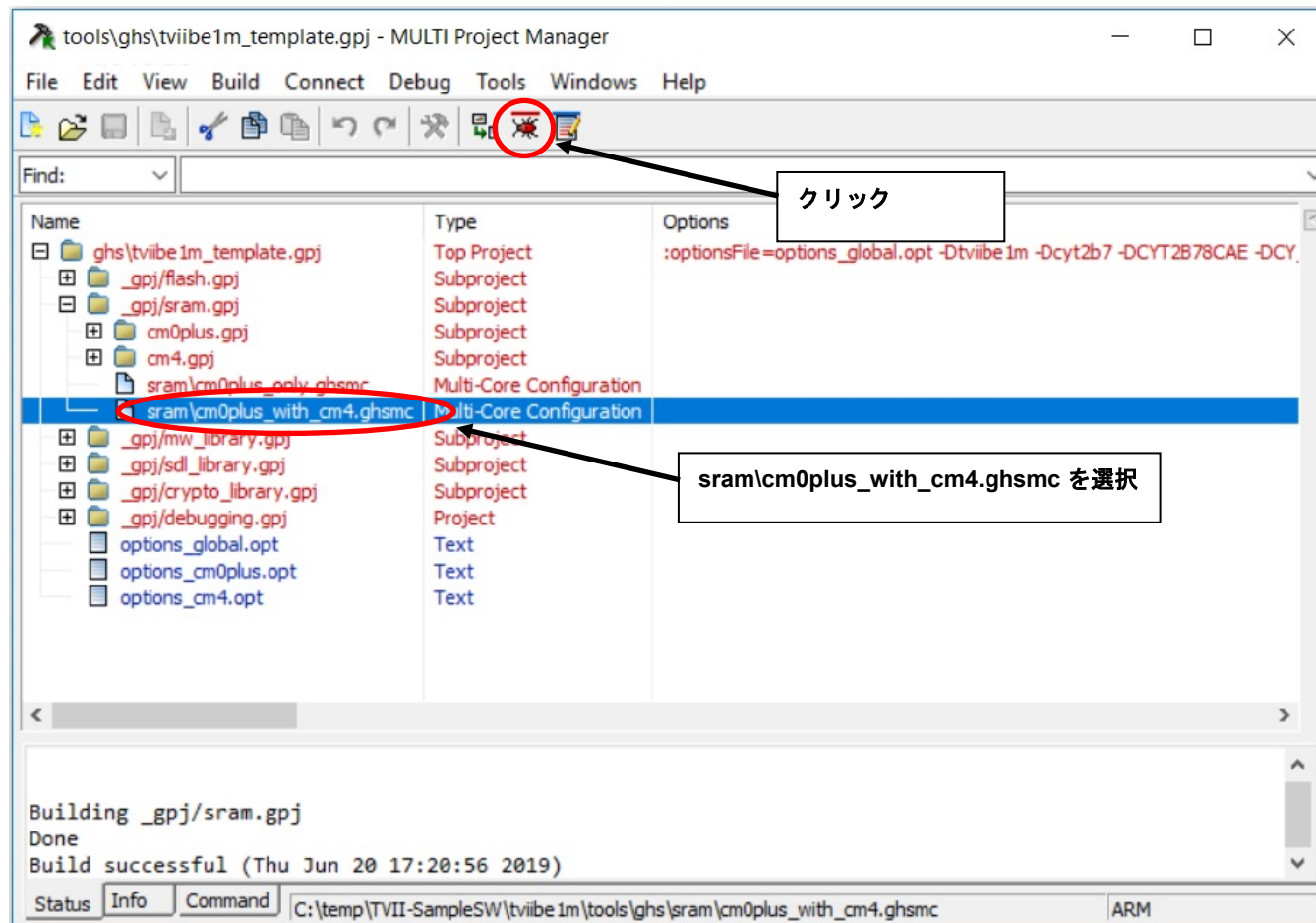
2つのデバッグ方法があります。

- 1) RAM へのダウンロードによるデバッグ
- 2) フラッシュメモリへのダウンロードによるデバッグ

3.4.4.1 RAM へのダウンロードによるデバッグ

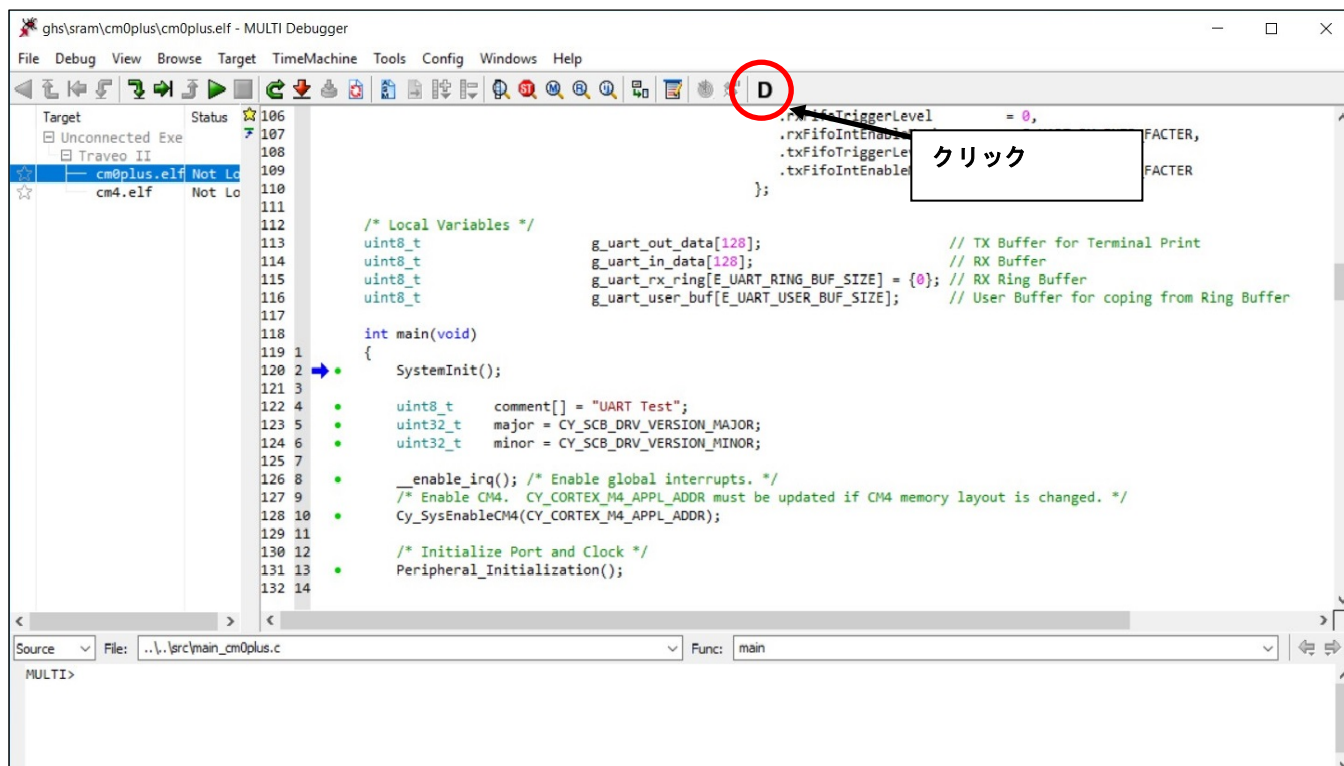
1. RAM 上でデバッグ実行する場合は、sram\cm0plus_with_cm4.ghsmc を選択し、図 14 のように Bug アイコンをクリックします。次に、RAM 上のプログラムをデバッグ開始します。

図 14. SDL プロジェクトのデバッグ



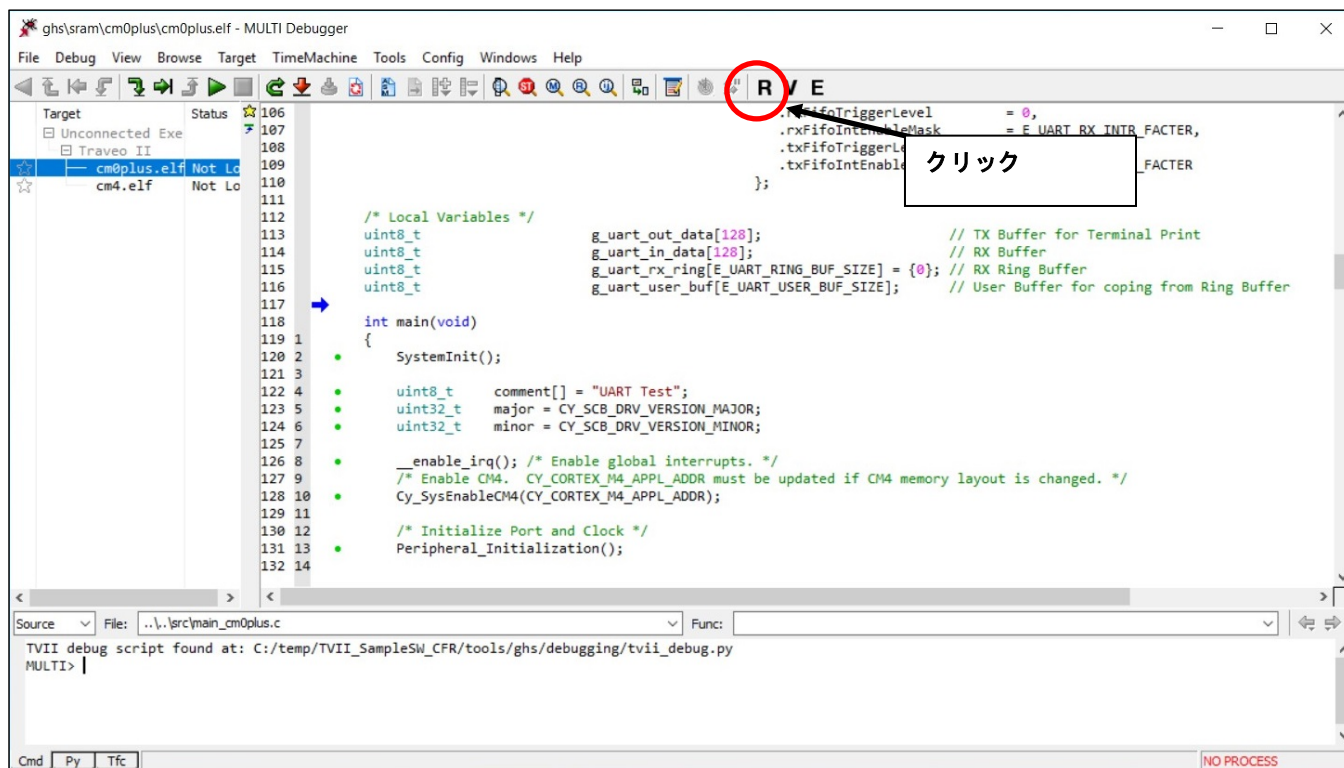
2. SDL プロジェクトでデバッグスクリプトを検出するために、図 15 に示すように **D** アイコンをクリックします。

図 15. SDL プロジェクトのデバッグ



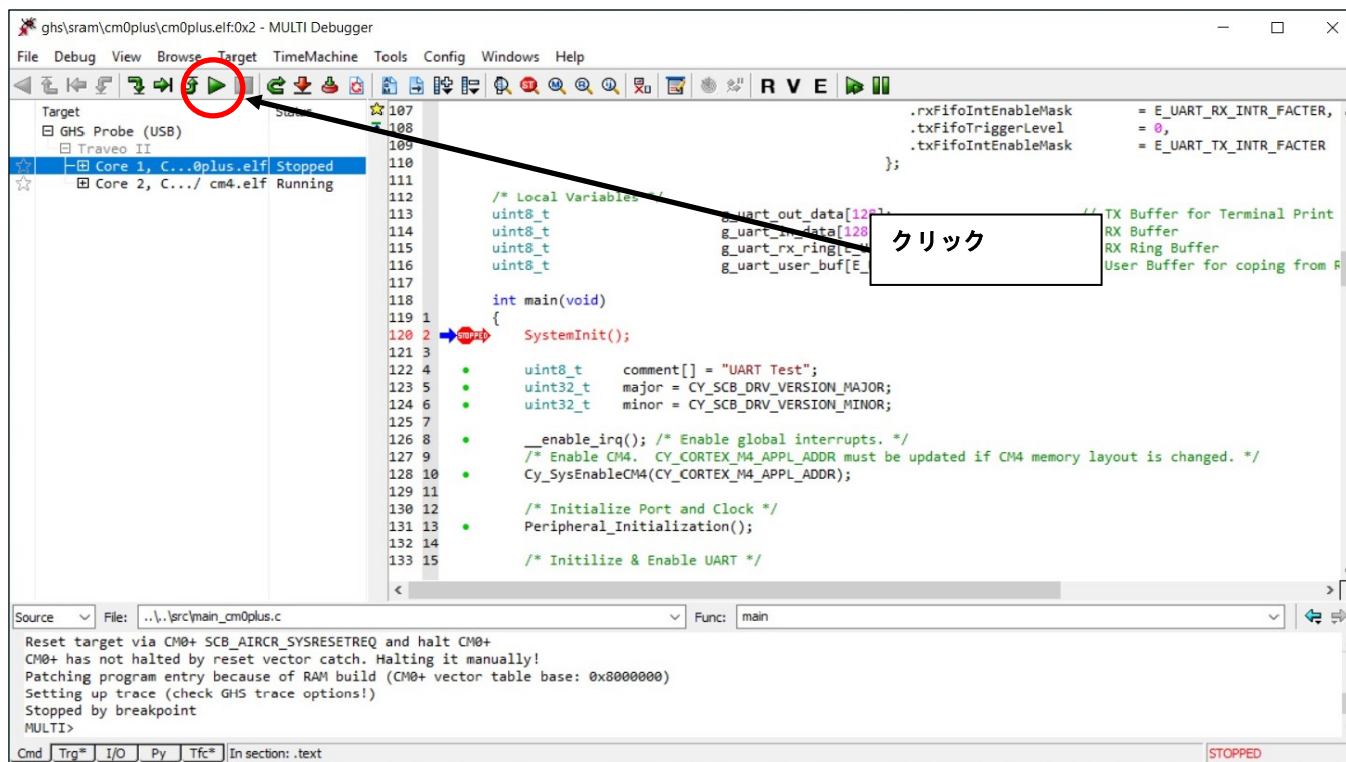
3. プログラムを RAM にロードするには、図 16 に示すように **R** アイコンをクリックします。プログラムの RAM へのロードが開始されます。

図 16. RAM へのロード



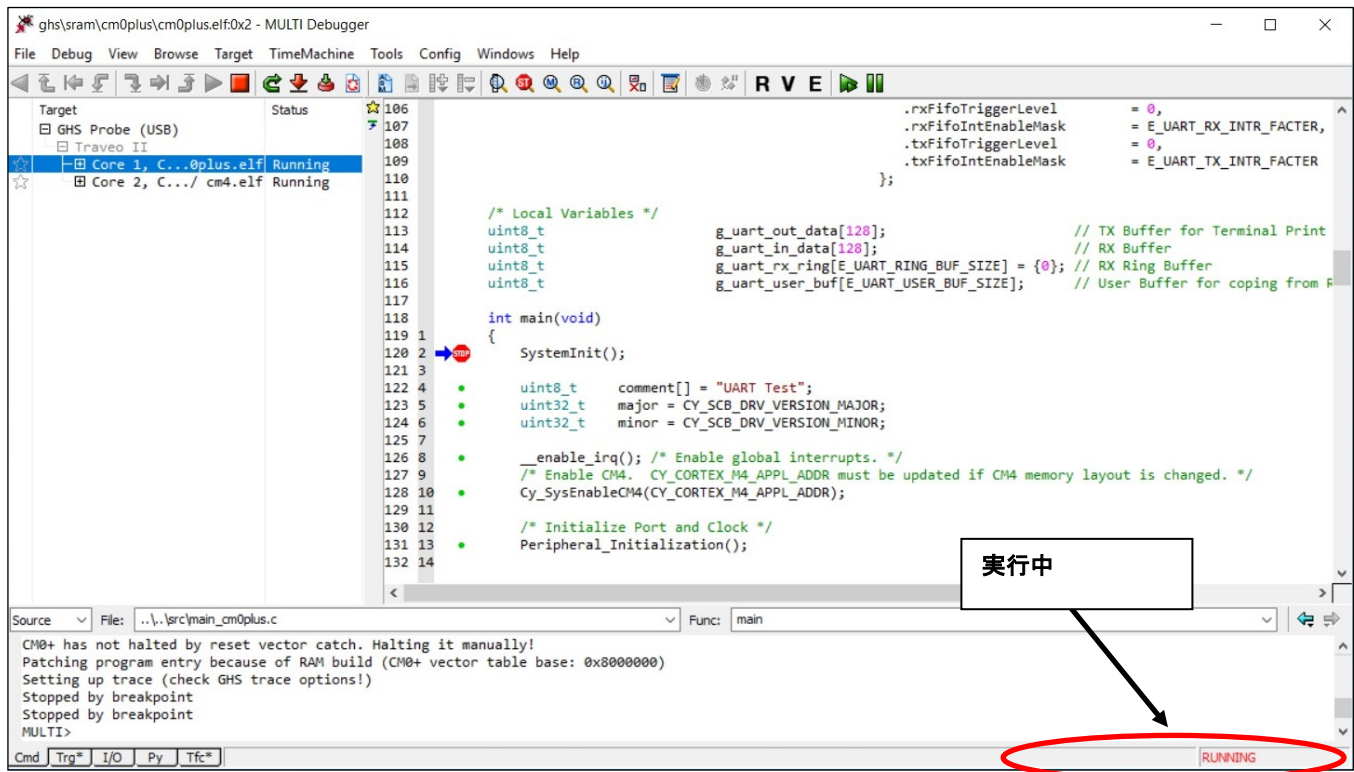
4. 図 17 に示すように**実行アイコン**をクリックします。

図 17. RAM 上でのプログラム実行



操作が実行されると、CM0+と CM4 の両方のプログラムがプログラム実行状態になり、図 18 に示すようにデバッグが可能になります。そして、ボードの LED が点滅し始めます。

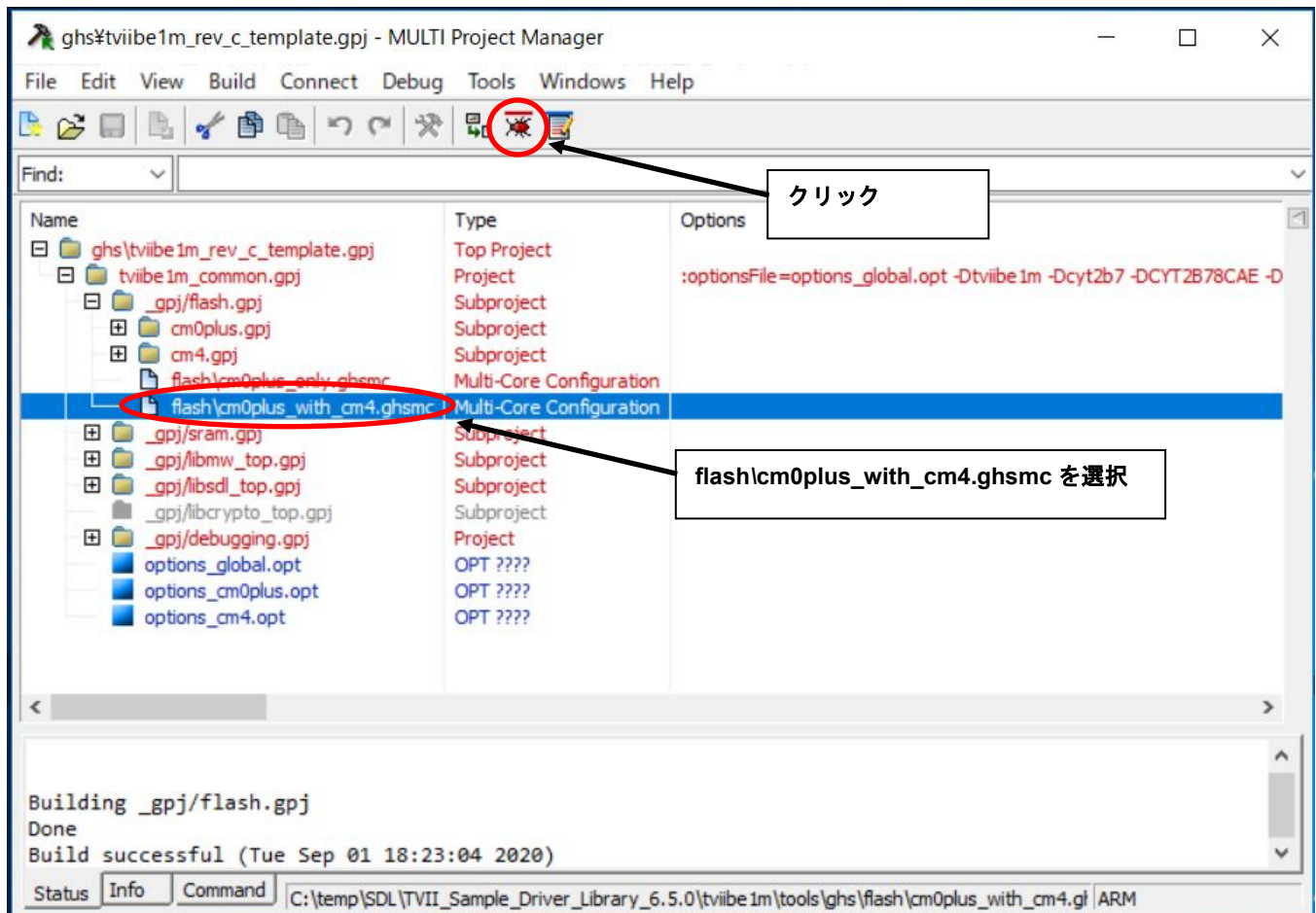
図 18. RAM 上でのプログラム実行中



3.4.4.2 フラッシュメモリへのダウンロードによるデバッグ

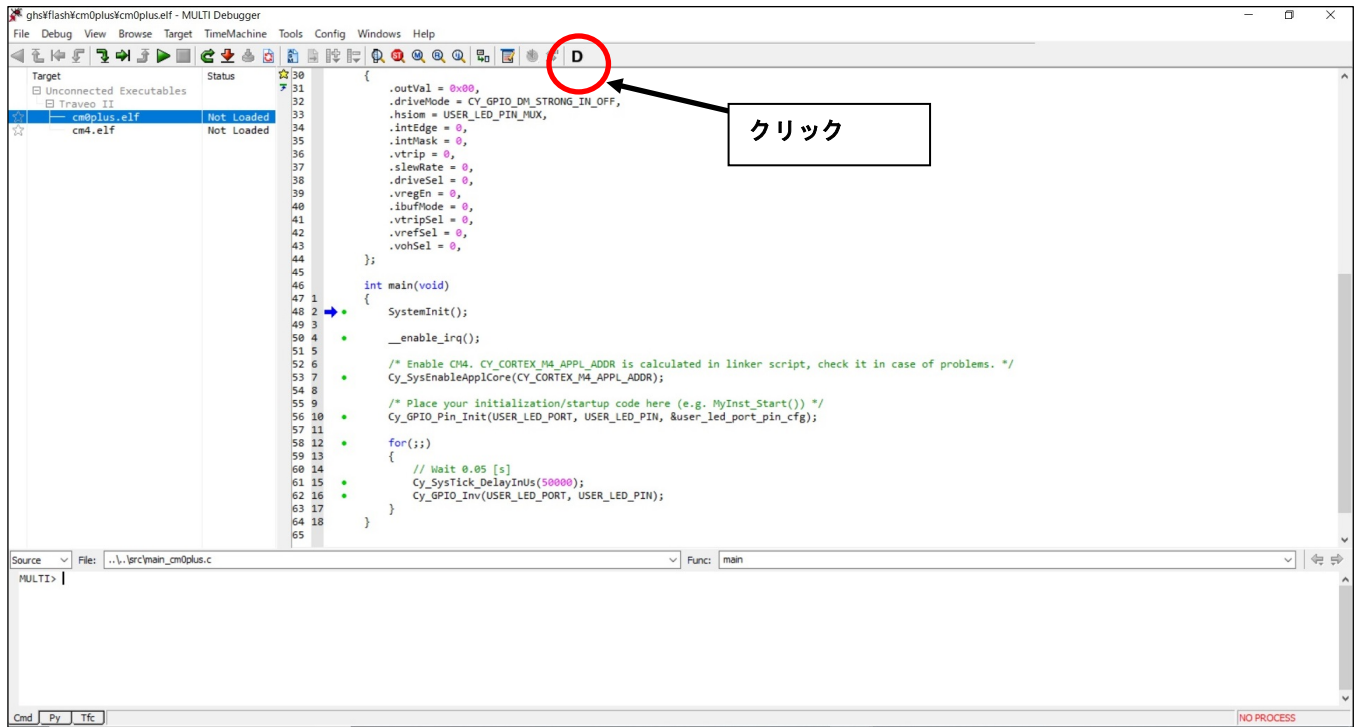
1. フラッシュメモリ上でデバッグ実行する場合は、`flash\cm0plus_with_cm4.ghsmc` を選択し、図 19 のように **Bug** アイコンをクリックします。次に、フラッシュメモリ上のプログラムをデバッグ開始します。

図 19. SDL プロジェクトのデバッグ



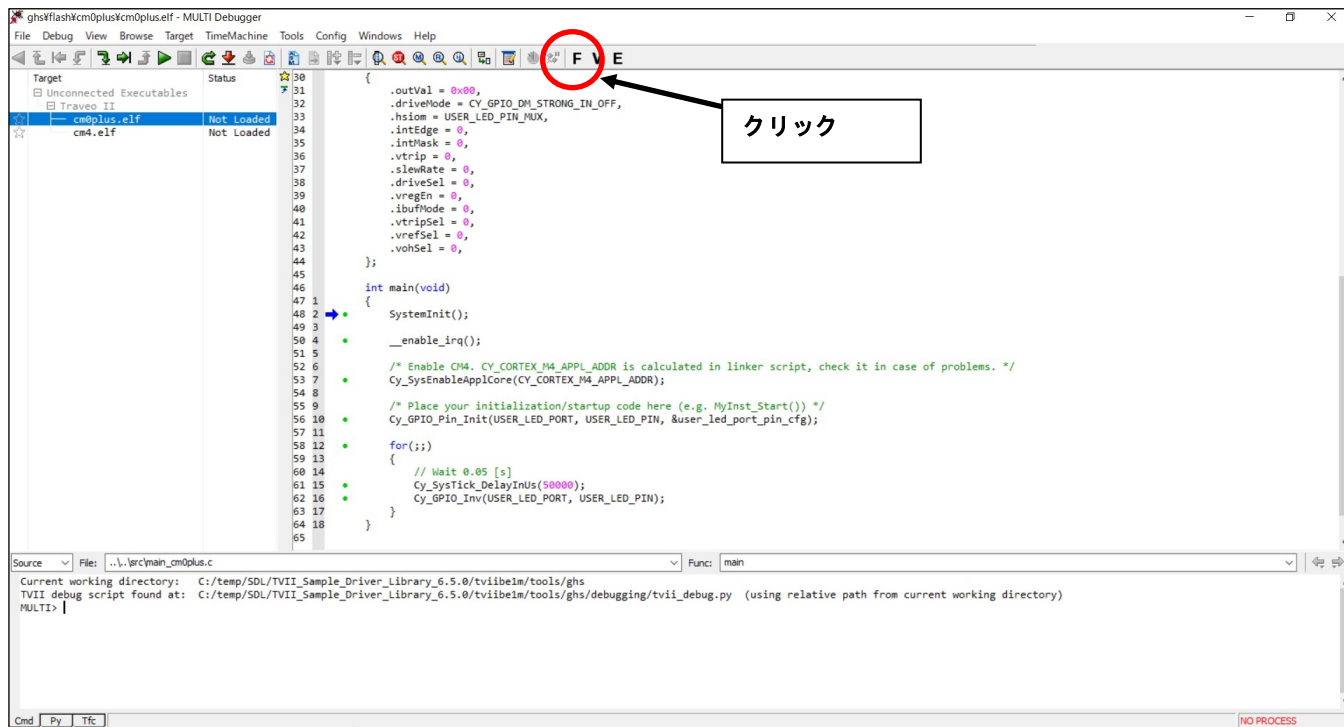
2. SDL プロジェクトでデバッグスクリプトを検出するために、図 20 に示すように **D** アイコンをクリックします。

図 20. SDL プロジェクトのデバッグ



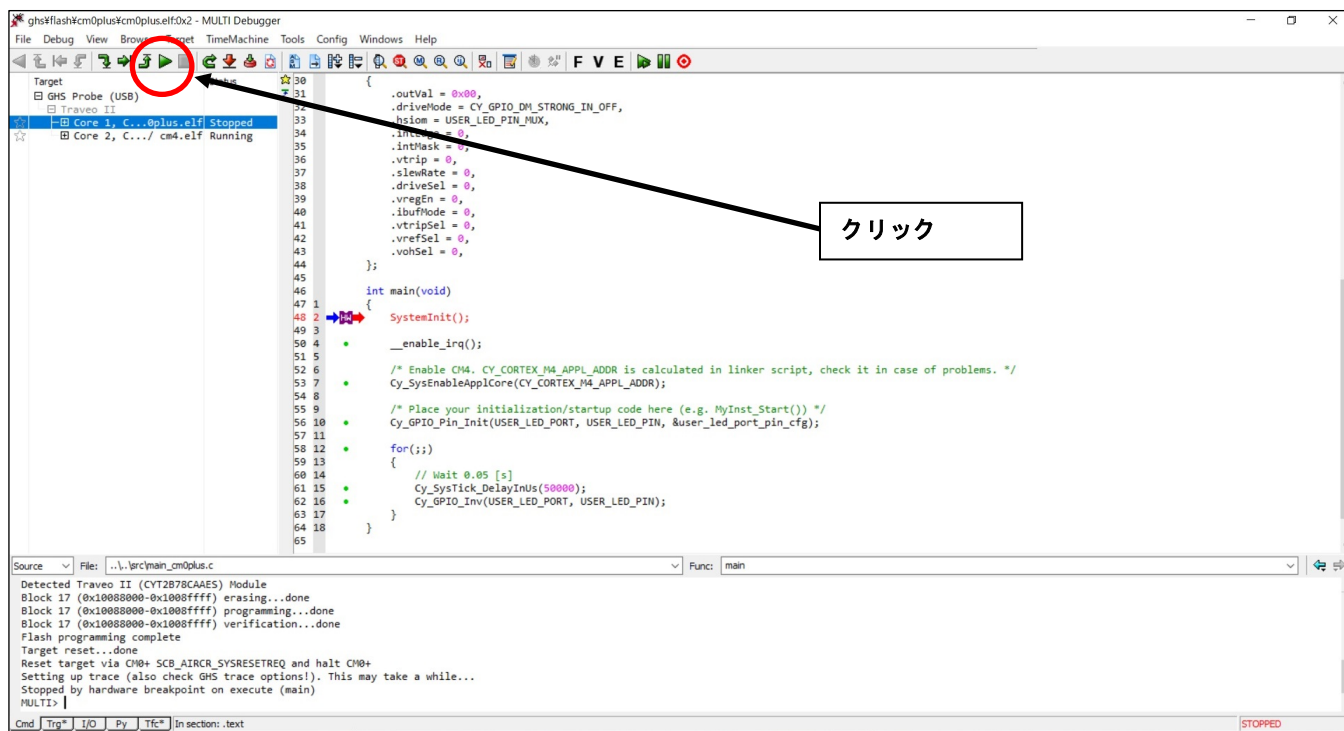
3. プログラムをフラッシュメモリにロードするには、図 21 に示すように F アイコンをクリックします。プログラムのフラッシュメモリへのロードが開始されます。

図 21. フラッシュメモリへのロード



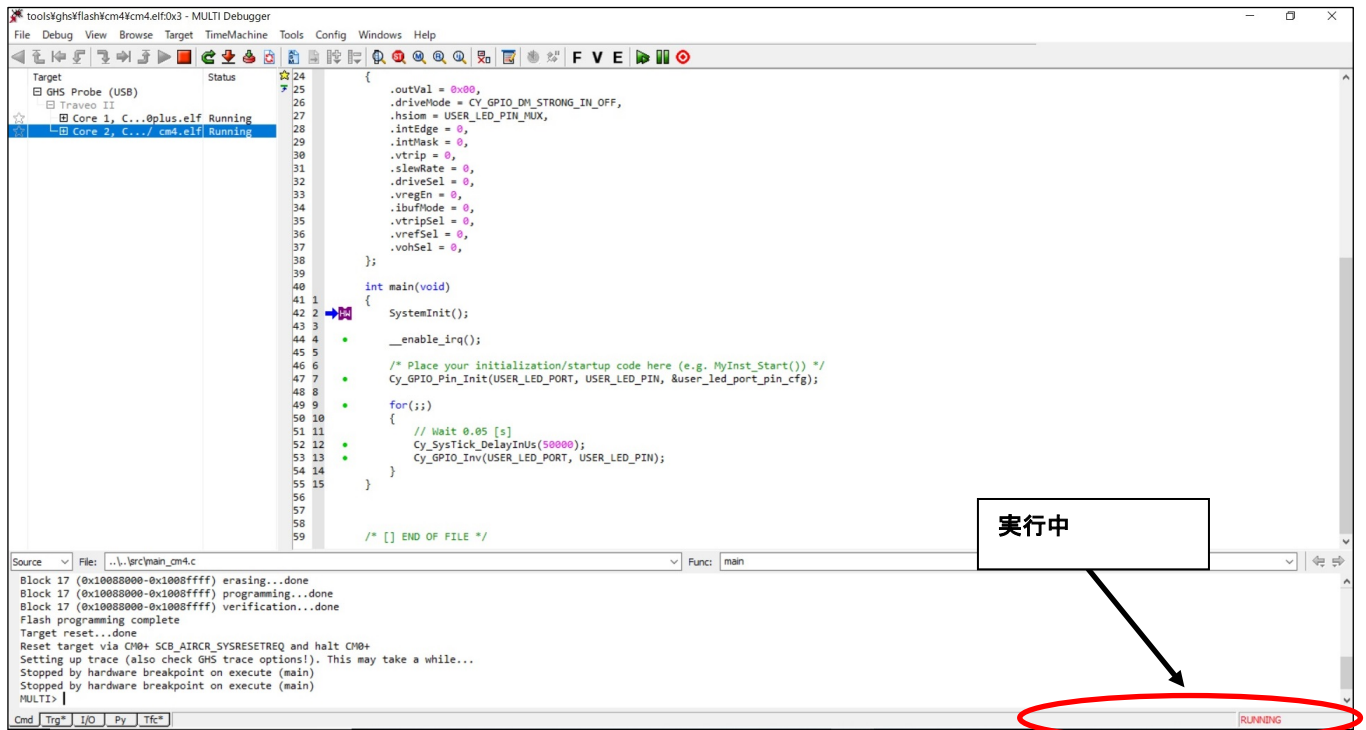
4. 図 22 に示すように**実行**アイコンをクリックします。

図 22. フラッシュメモリ上でのプログラム実行



操作が実行されると、CM0+と CM4 の両方のプログラムがプログラム実行状態になり、デバッグが可能になります。そして、ボードの LED が点滅し始めます。

図 23. フラッシュメモリ上でのプログラム実行中



3.5 IAR Embedded Workbench for Arm を使用した SDL の使用例

3.5.1 SDL の環境

ここでは、CYT2B7 シリーズ SDL の操作例を示します。

SDL の動作環境:

- CPU: CYT2B7 シリーズ
- ターゲットボード: Traveo II Evaluation board
- IDE: IAR Embedded Workbench for Arm (compiler version 8.42.1)
- API モジュール: GPIO

3.5.2 GHS MULTI と SDL のインストール

IAR Embedded Workbench for Arm, ライセンスの設定, および I-Jet の設定を事前に行ってください。

SDL のガイドに従って SDL を事前にインストールしてください。

3.5.3 IAR EW for Arm によるセットアップ

2つのデバッグ方法があります。

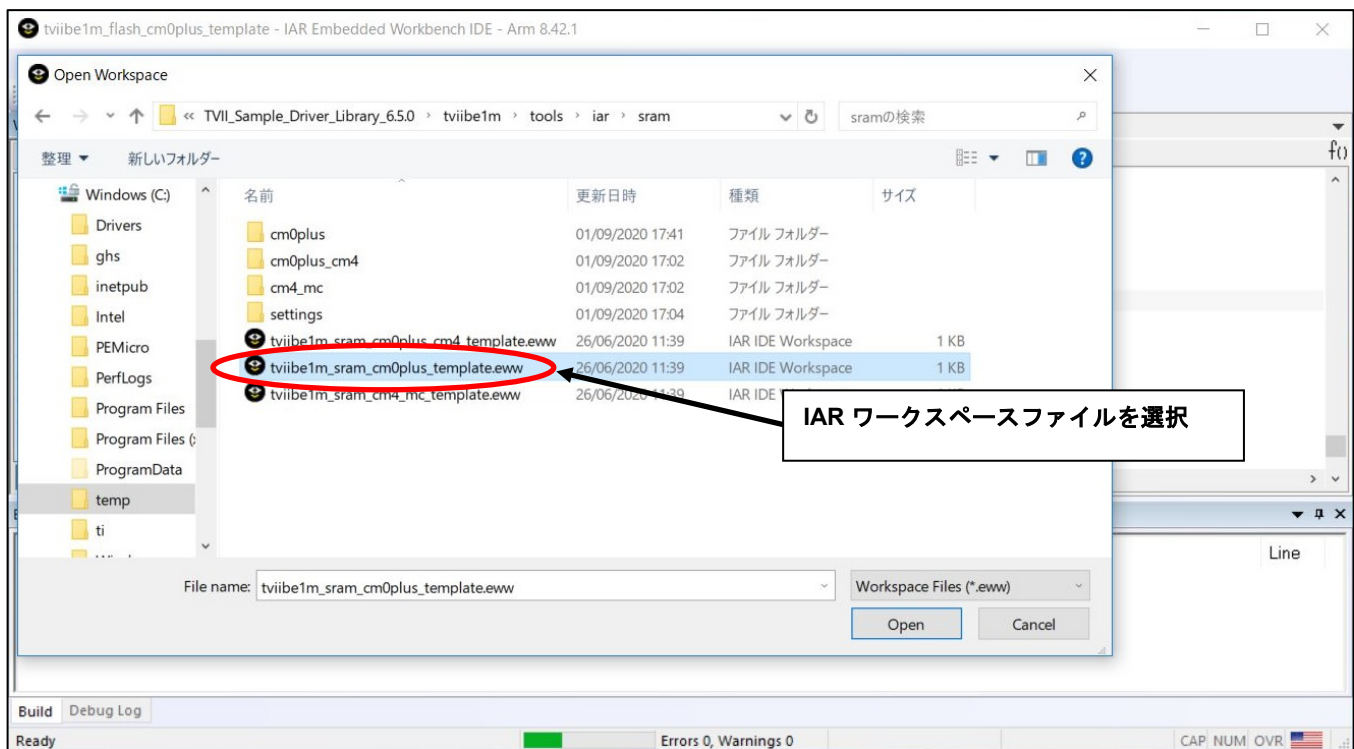
- 1) RAM へのダウンロードによるデバッグ
- 2) フラッシュメモリへのダウンロードによるデバッグ

3.5.3.1 RAM へのダウンロードによるデバッグ

1. IAR EW for Arm を起動し、SDL ワークスペースファイルを開きます:

`C:\...\TVII_Sample_Driver_Library_revision\tviibe1m\tools\iar\sram\tviibe1m_sram_cm0plus_template.eww`

図 24. SDL ワークスペースの選択



- ビルドの場合、図 25 に示すように、ワークスペースで cm0plus.eww を選択し、右クリックして[Rebuild All]メニューを選択します。[Rebuild All]が開始され、正常終了のメッセージが図 26 のように表示されます。

図 25. SDL プロジェクトのリビルド

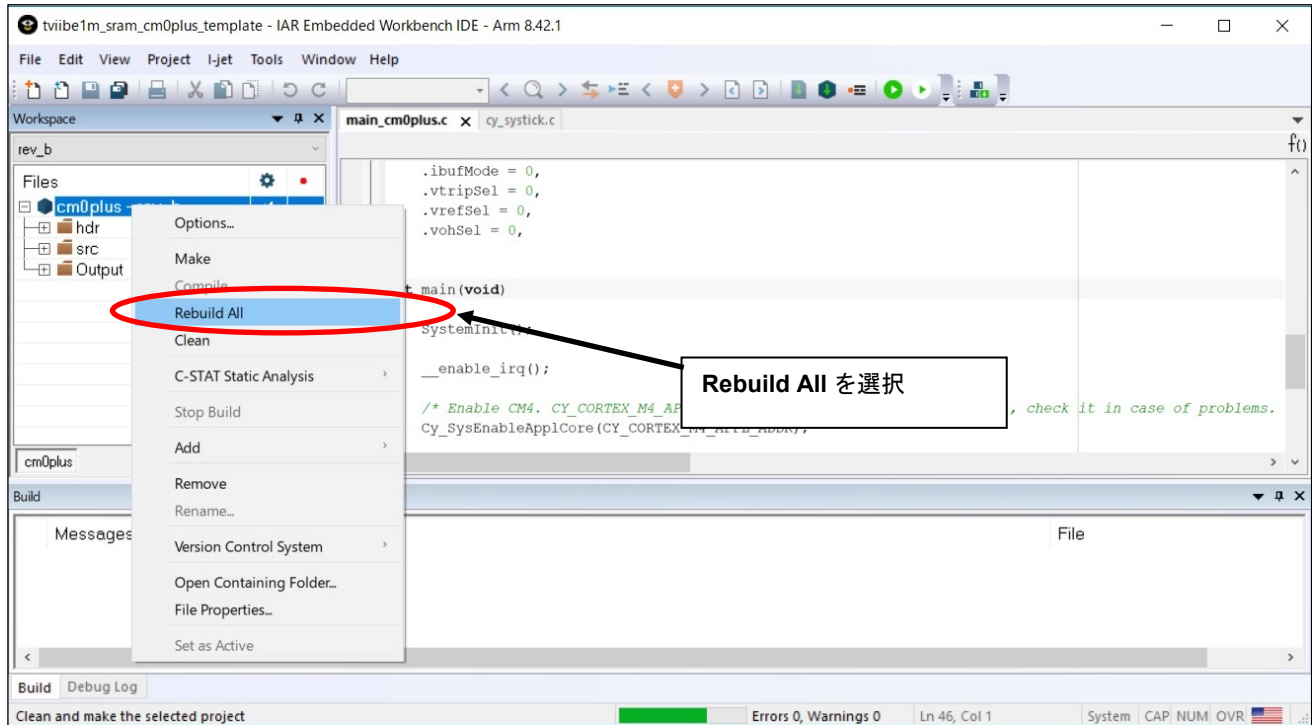
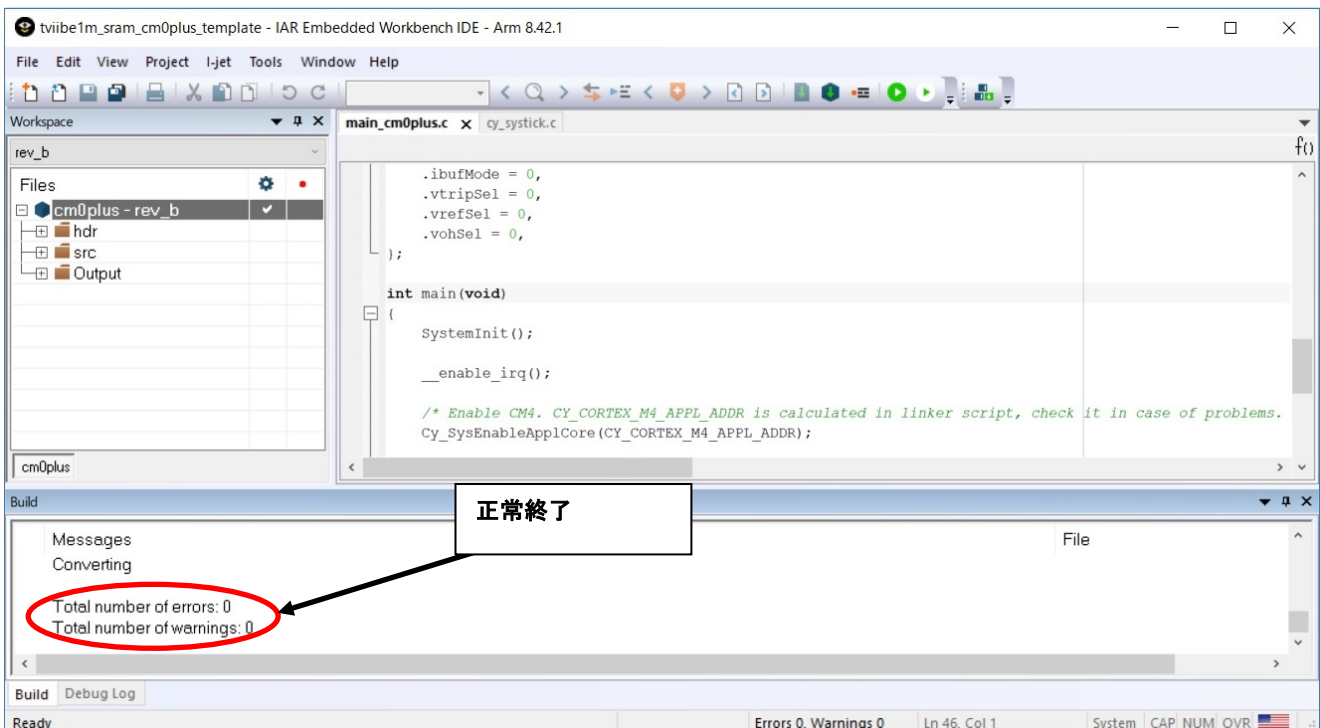
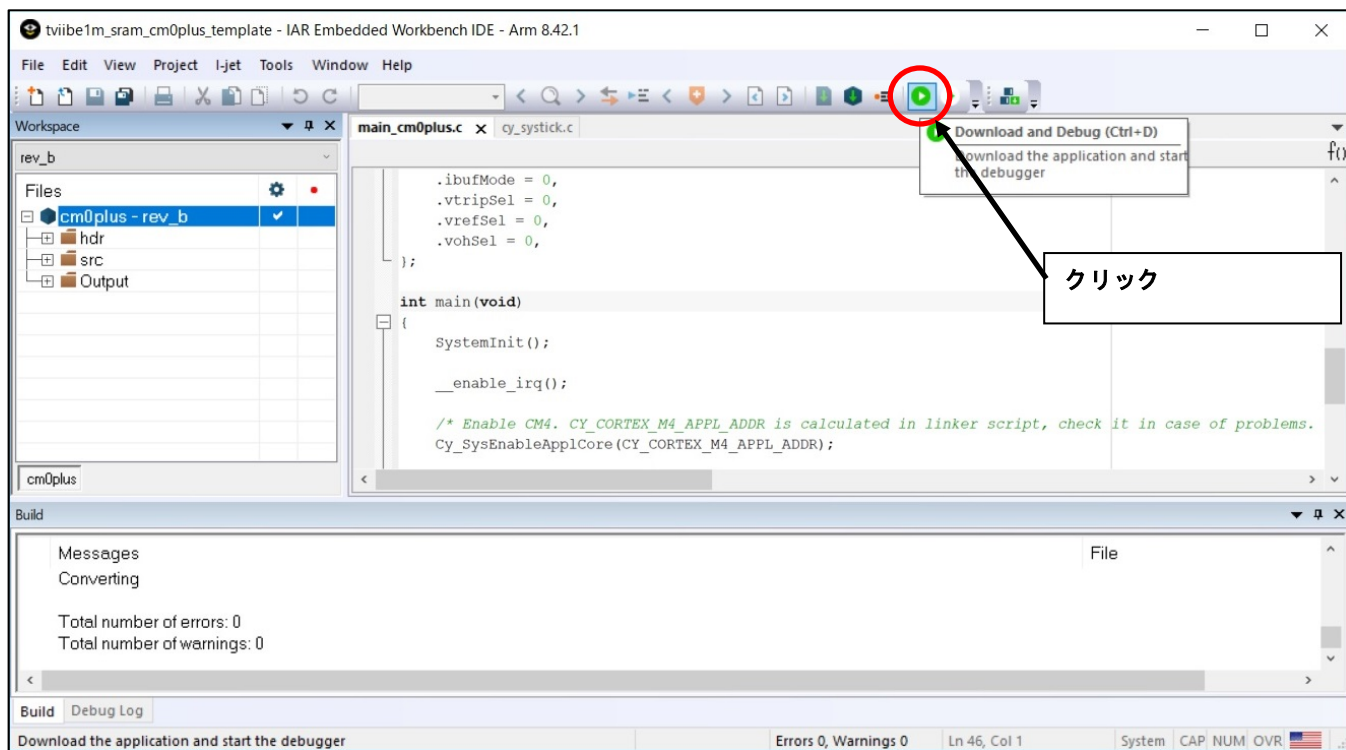


図 26. SDL プロジェクトのリビルドの正常終了



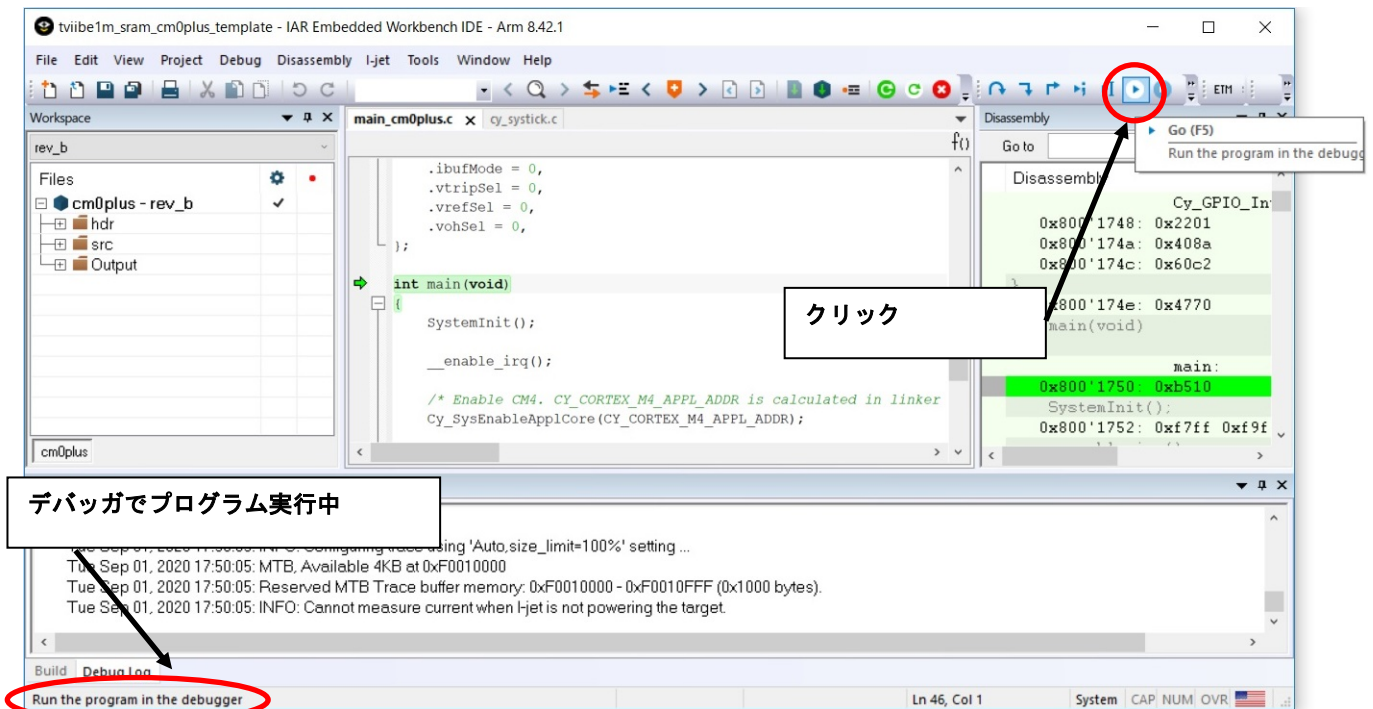
3. プログラムを RAM にロードするには、図 27 に示すようにダウンロードアイコンをクリックします。

図 27. SDL Project のデバッグ



4. 図 28 に示すように、Go アイコンをクリックします。

図 28. RAM 上でのプログラム実行



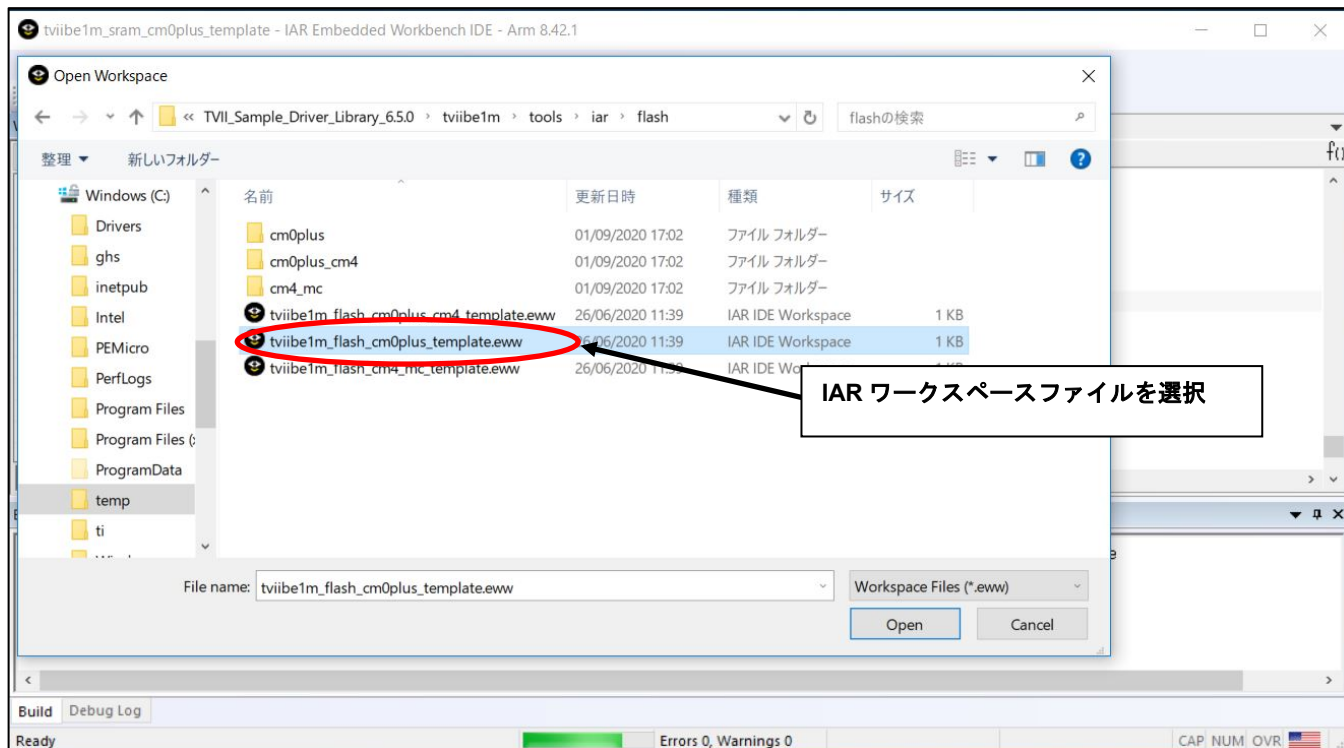
操作が実行されると、CM0+のプログラムがプログラム実行状態になり、図 28 に示すようにデバッグが可能になります。そして、ボードの LED が点滅し始めます。

3.5.3.2 フラッシュメモリへのダウンロードによるデバッグ

1. IAR EW for Arm を起動し SDL ワークスペースファイルを開きます:

C:\...\TVII_Sample_Driver_Library_revision\tviibe1m\tools\iar\flash\tviibe1m_flash_cm0plus_template.eww

図 29. SDL ワークスペースの選択



2. ビルドの場合、図 30 に示すように、ワークスペースで cm0plus.eww を選択し、右クリックして **Rebuild All** メニューを選択します。[Rebuild All]が開始され、正常終了のメッセージが図 31 のように表示されます。

図 30. SDL プロジェクトのリビルド

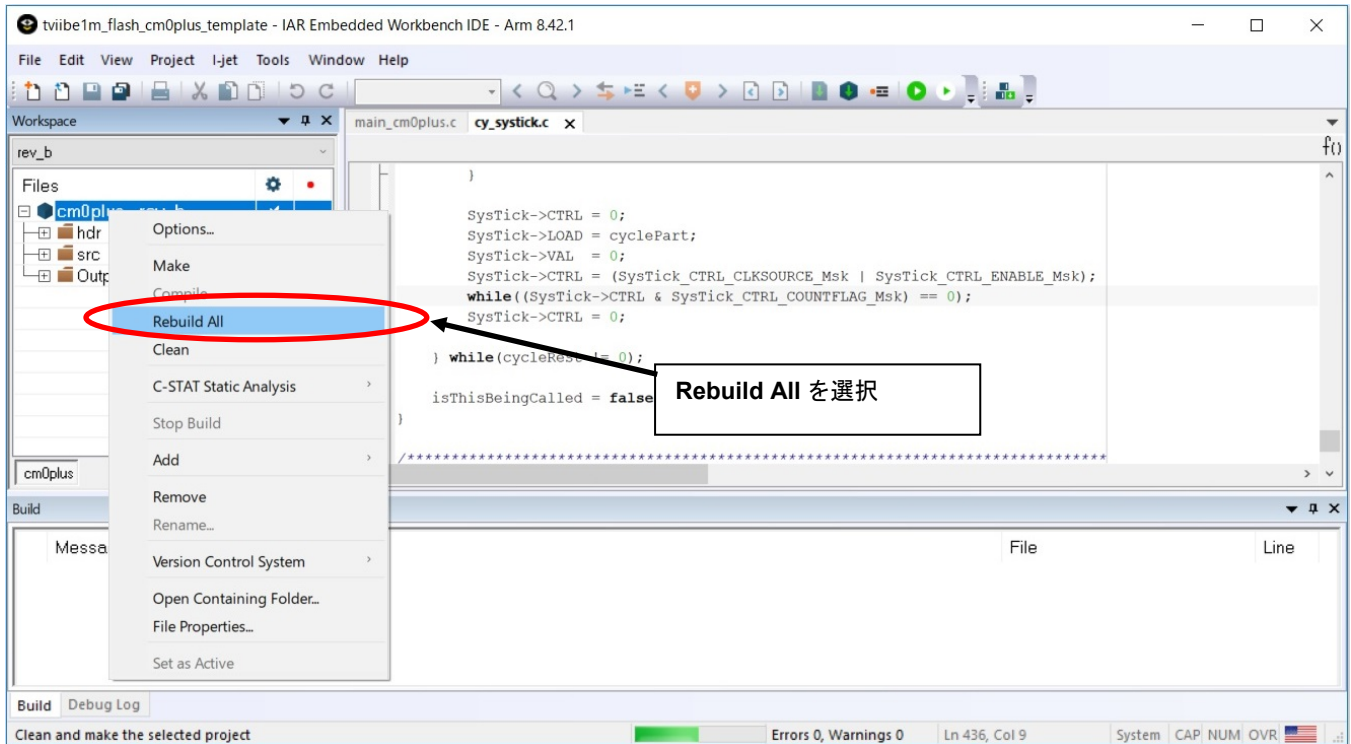
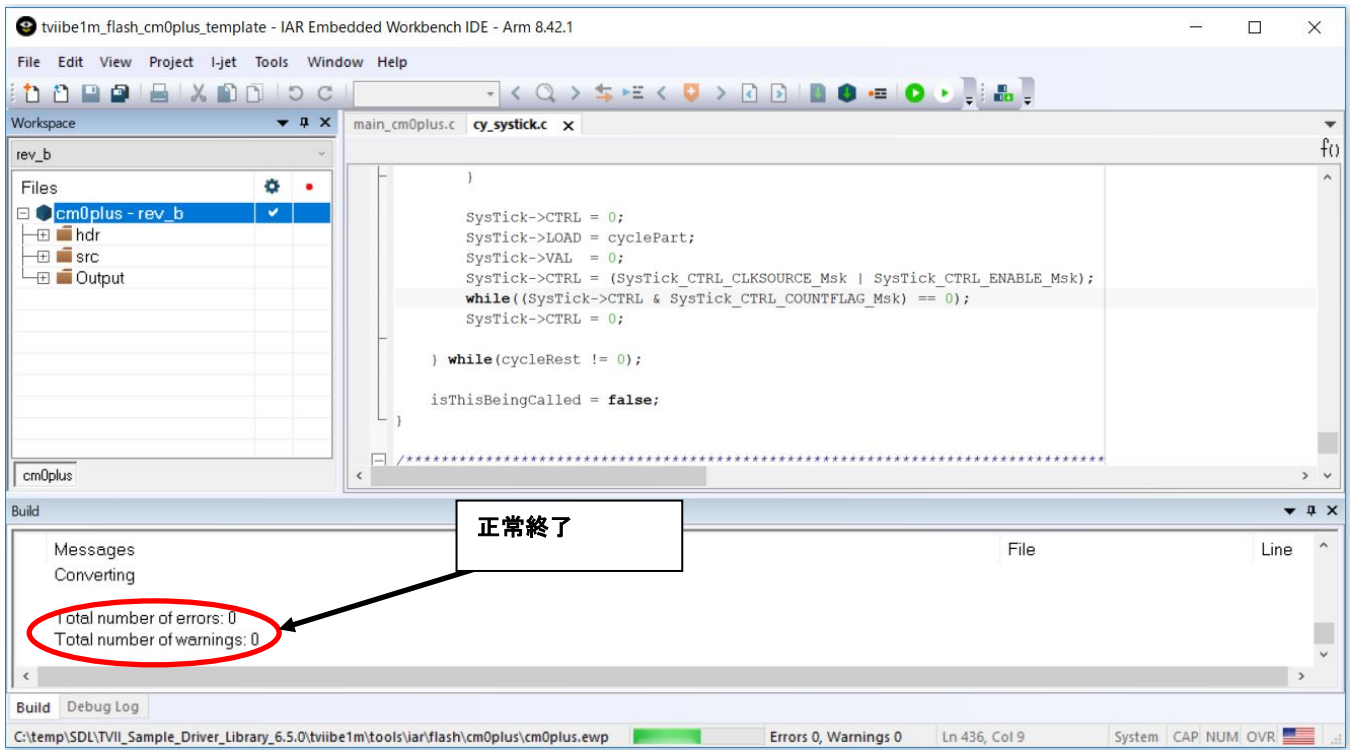
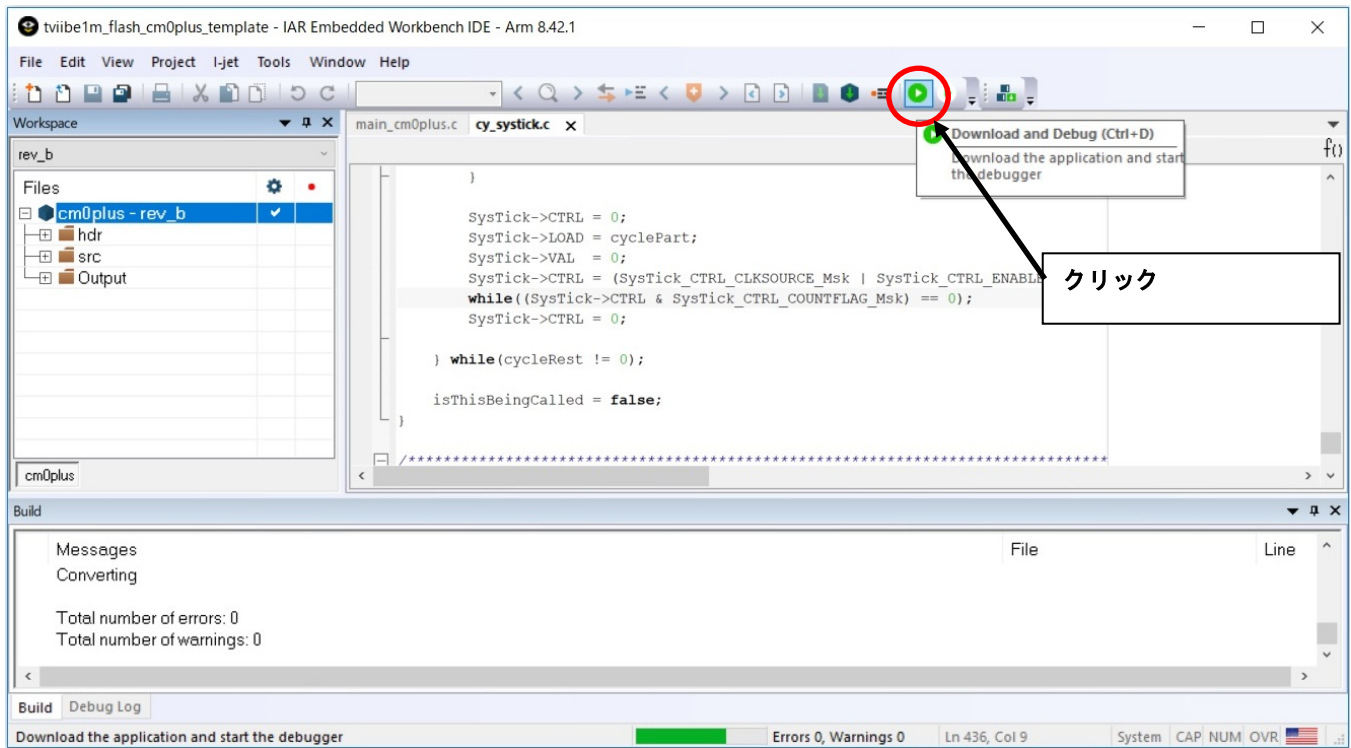


図 31. SDL プロジェクトのリビルドの正常終了



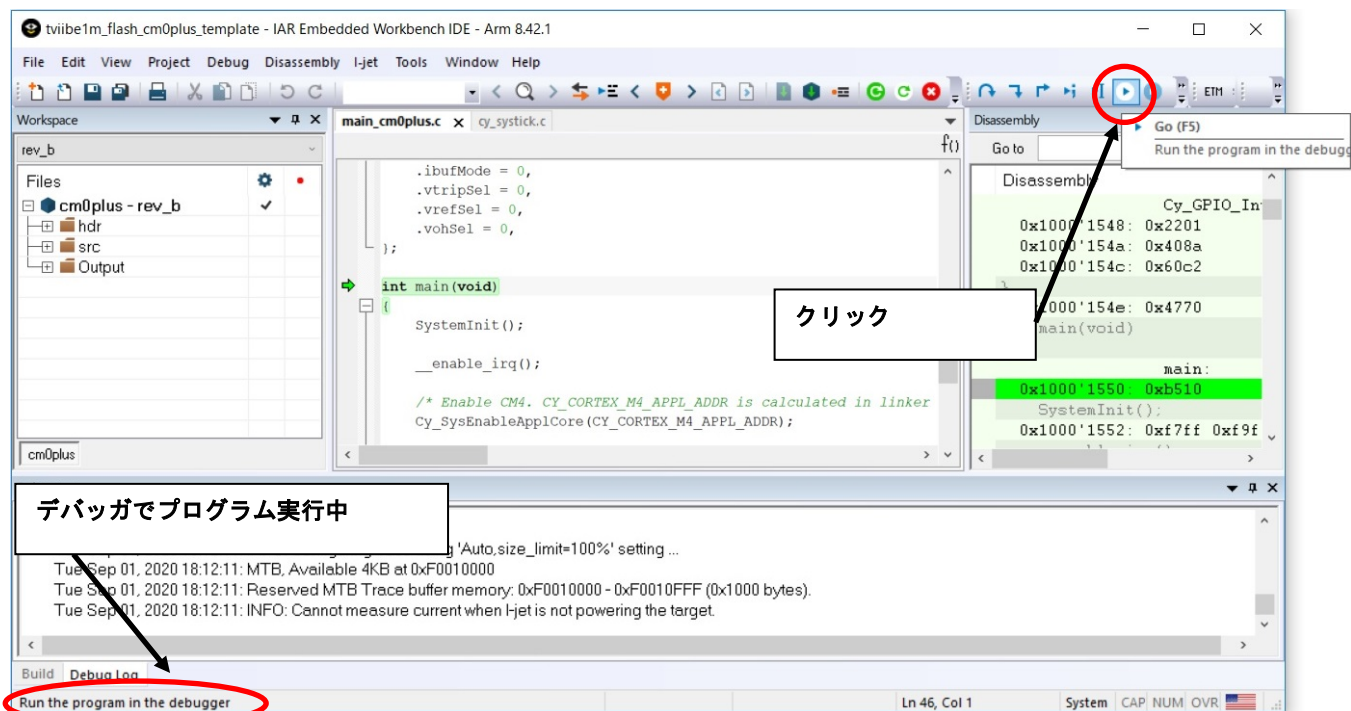
3. プログラムをフラッシュメモリにロードするには、図 32 に示すようにダウンロードアイコンをクリックします。

図 32. SDL プロジェクトのデバッグ



4. 図 33 に示すように、Go アイコンをクリックします。

図 33. フラッシュメモリ上でのプログラム実行



操作が実行されると、CM0+のプログラムがプログラム実行状態になり、図 33 に示すようにデバッグが可能になります。そして、ボードの LED が点滅し始めます。

3.6 開発ツール

表 5 に、サポートする開発ツールを示します。

表 5. 開発ツール

ベンダ	エミュレータ/プローブ	コンパイラ
Green Hills Software Ltd.	GHS Probe (5.6.4/5.6.6)	MULTI V7 (version 7.1.4)
IAR Systems AB	I-JET	Embedded Workbench for Arm (8.42.1)
iSYSTEM AG	i-TAG Family	-
Lauterbach GmbH	TRACE 32-ICE	-

3.7 フラッシュプログラミングツール

表 6 に、サポートするフラッシュプログラミングツールを示します。サイプレス Minipro3/4 プログラマは開発目的のプログラマです。量産の場合は、サードパーティの JTAG プログラマが適しています。

表 6. フラッシュプログラミングツール

ベンダ	フラッシュプログラマ	ソフトウェア	備考
サイプレス セミコンダクタ	Minipro3	Cypress Programmer 1.0	CYT2B7 のみサポート
	Minipro4	Cypress Auto Flash Utility	
SEGGER Microcontroller GmbH	J-Link	J-Flash	
	Flasher Arm (for mass-production)		
株式会社 DTS インサイト	NETIMPRESS AF430	Remote Controller AZ490	

CYTVII-B-E-1M-xxx-CPU/CYTVII-B-H-8M-xxx-CPU/CYTVII-C-2D-6M-xxx-CPU には、JTAG と SWD の両機能のための Cortex デバッグコネクタがあります。SWD は MiniProg3 と互換性があります。

表 7 に、Cortex デバッグコネクタのピン割当てを示します。

表 7. Cortex デバッグコネクタ

ピン番号	信号名
1	VCC
2	SWDIO/TMS
3	GND
4	SWDCLK/TCK
5	GND
6	SWO/TDO
7	KEY
8	NC/TDI
9	GNDDetect
10	nRESET

4 まとめ

本アプリケーションノートは、Traveo II 開発環境をセットアップするためのガイドです。サイプレスは、Traveo II ファミリを使い始める際に役立つ、豊富な評価ボードとサンプルソフトウェアを提供します。CYT2/CYT3/CYT4 シリーズを評価する際は、営業担当者またはサイプレステクニカルサポートに連絡してください。

5 用語集

表 8. 用語集

用語	説明
CAN FD	Controller Area Network with Flexible Data rate
LIN	Local Interconnect Network
CXPI	Clock Extension Peripheral Interface
RWW	Read-While-Write (読み込み中の書き込み)
MPU	Memory Protection Unit (メモリ保護ユニット)
Crypto	Cryptography Accelerator (暗号エンジン)
ASIL-B	Automotive Safety Integrity Level-B
JTAG	Joint Test Action Group
SWD	Single Wire Debug
ETM	Embedded Trace Macrocell
CMSIS	Cortex Microcontroller Software Interface Standard
Gigabit Ethernet	1 Gbps Ethernet
SDL	Sample Driver Library
ADC	Analog-to-Digital Converter
DMA	Direct Memory Access
EVTGEN	Event Generator
FDP-Link	Flat Panel Display Link
LVDS	Low Voltage Differential Signaling (小振幅差動信号)
GPIO	General-Purpose Input/Output (汎用入出力)
IPC	Inter-Processor Communication
MCWDT	Multi-Counter Watchdog Timers
MIPI	Mobile Industry Processor Interface
CSI2	Camera Serial Interface-2
PROT	Protection Unit (保護ユニット)
SCB	Serial Communication Block
SYSCLK	System Clock
SYSFLT	System Fault
SYSINT	System Interrupt
SYSLIB	System Library
SYSPM	System Power Management
REGHC	High Current Regulator
SYSRTC	System Real-Time Clock
SYSTICK	Systick Timer
SYSWDT	フリーラン Watchdog Timer
TCPWM	Timer Counter PWM
TRIGMUX	Trigger Multiplexer

6 関連ドキュメント

以下は Traveo II ファミリシリーズのアプリケーションノート、データシート、テクニカルリファレンスマニュアルです。これらの資料を入手するには、[テクニカルサポート](#)に連絡してください。

表 9. 関連ドキュメント

Application Notes	説明
AN220270 - Traveo II ボディファミリマイコンのハードウェアデザインガイド	Traveo II のハードウェア環境を設定する方法について説明します。
Device Datasheet	説明
CYT2B7 Datasheet 32-Bit Arm® Cortex®-M4F Microcontroller Traveo™ II Family	対象製品の機能と電氣的仕様を説明しています。アドレスマップ、I/O マップ、端子配列図、代替機能端子配列図、割込みソースリスト、トリガグループ、ペリフェラルクロック、フォールト、バスマスタなどの製品固有の情報について説明します。
CYT2B9 Datasheet 32-Bit Arm® Cortex®-M4F Microcontroller Traveo™ II Family	
CYT4BF Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family	
CYT4DN Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family	
CYT3BB/4BB Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family	
Architecture Technical Reference Manual	説明
Traveo™ II Automotive Body Controller Entry Family Architecture Technical Reference Manual (TRM)	製品ファミリのアーキテクチャと機能について説明します。
Traveo™ II Automotive Body Controller High Family Architecture Technical Reference Manual (TRM)	
Traveo™ II Automotive Cluster 2D Family Architecture Technical Reference Manual (TRM)	
Registers Technical Reference Manual	説明
Traveo™ II Automotive Body Controller Entry Registers Technical Reference Manual (TRM) for CYT2B7	製品ファミリのレジスタリスト、初期値、およびレジスタの属性について説明します。
Traveo™ II Automotive Body Controller Entry Registers Technical Reference Manual (TRM) for CYT2B9	
Traveo™ II Automotive Body Controller High Registers Technical Reference Manual (TRM) for CYT4BF	
Traveo™ II Automotive Body Controller High Registers Technical Reference Manual (TRM) for CYT3BB/4BB	
Traveo™ II Automotive Cluster 2D Registers Technical Reference Manual (TRM)	

改版履歴

文書名: AN220118 - Traveo II ファミリ MCU スタートアップガイド

文書番号: 002-21379

版	ECN	発行日	変更内容
**	6094796	03/12/2018	これは英語版 002-20118 Rev. **を翻訳した日本語版 002-21379 Rev. **です。
*A	6466430	01/30/2019	これは英語版 002-20118 Rev. *A を翻訳した日本語版 002-21379 Rev. *A です。
*B	6645262	08/09/2019	これは英語版 002-20118 Rev. *B を翻訳した日本語版 002-21379 Rev. *B です。
*C	6856807	04/17/2020	これは英語版 002-20118 Rev. *C を翻訳した日本語版 002-21379 Rev. *C です。 主な変更点は以下のとおりです。 対象の製品番号を変更しました。(CYT2/CYT4 シリーズ) 対象の製品番号を追加しました。(CYT3 シリーズ)
*D	7019574	11/10/2020	これは英語版 002-20118 Rev. *D を翻訳した日本語版 002-21379 Rev. *D です。 主な変更点は以下のとおりです。 3.4.4 MULTI によるセットアップを変更しました。 3.5 IAR Embedded Workbench for Arm を使用した SDL の使用例を追加しました。

ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店、および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、[サイプレスのロケーション ページ](#)をご覧ください。

製品

Arm® Cortex® Microcontrollers	cypress.com/arm
車載用	cypress.com/automotive
クロック&バッファ	cypress.com/clocks
インターフェース	cypress.com/interface
IoT(モノのインターネット)	cypress.com/iot
メモリ	cypress.com/memory
マイクロコントローラ	cypress.com/mcu
PSoC	cypress.com/psoc
電源用 IC	cypress.com/pmuc
タッチ センシング	cypress.com/touch
USB コントローラー	cypress.com/usb
ワイヤレス	cypress.com/wireless

PSoC® ソリューション

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6 MCU](#)

サイプレス開発者コミュニティ

[コミュニティ](#) | [サンプルコード](#) | [Projects](#) | [ビデオ](#) | [ブログ](#) | [トレーニング](#) | [Components](#)

テクニカルサポート

cypress.com/support

Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

 **CYPRESS**
EMBEDDED IN TOMORROW™

Cypress Semiconductor
An Infineon Technologies Company
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2017-2020. 本書面は、Cypress Semiconductor Corporation 及び Spansion LLC を含むその子会社 (以下「Cypress」という。) に帰属する財産である。本書面 (本書面に含まれ又は言及されているあらゆるソフトウェア若しくはファームウェア (以下「本ソフトウェア」という。)) を含む) は、アメリカ合衆国及び世界のその他の国における知的財産法令及び条約に基づき Cypress が所有する。Cypress はこれらの法令及び条約に基づく全ての権利を留保し、本段落で特に記載されているものを除き、その特許権、著作権、商標権又はその他の知的財産権のライセンスを一切許諾しない。本ソフトウェアにライセンス契約書が伴っておらず、かつ Cypress との間で別途本ソフトウェアの使用方法を定める書面による合意がない場合、Cypress は、(1) 本ソフトウェアの著作権に基づき、(a) ソースコード形式で提供されている本ソフトウェアについて、Cypress ハードウェア製品と共に用いるためにのみ、かつ組織内部でのみ、本ソフトウェアの修正及び複製を行うこと、並びに (b) Cypress のハードウェア製品ユニットに用いるためにのみ、(直接又は再販売者及び販売代理店を介して間接のいずれかで) 本ソフトウェアをバイナリーコード形式で外部エンドユーザーに配布すること、並びに (2) 本ソフトウェア (Cypress により提供され、修正がなされていないもの) が抵触する Cypress の特許権のクレームに基づき、Cypress ハードウェア製品と共に用いるためにのみ、本ソフトウェアの作成、利用、配布及び輸入を行うことについての非独占的で譲渡不能な一身専属的ライセンス (サブライセンスの権利を除く) を付与する。本ソフトウェアのその他の使用、複製、修正、変換又はコンパイルを禁止する。

適用される法律により許される範囲内で、Cypress は、本書面又はいかなる本ソフトウェア若しくはこれに伴うハードウェアに関しても、明示又は黙示を問わず、いかなる保証 (商品性及び特定の目的への適合性の黙示の保証を含むがこれらに限られない) も行わない。いかなるコンピューティングデバイスも絶対に安全ということはない。従って、Cypress のハードウェアまたはソフトウェア製品に講じられたセキュリティ対策にもかかわらず、Cypress は、Cypress 製品への権限のないアクセスまたは使用といったセキュリティ違反から生じる一切の責任を負わない。加えて、本書面に記載された製品には、エラーと呼ばれる設計上の欠陥またはエラーが含まれている可能性があり、公表された仕様とは異なる動作をする場合がある。適用される法律により許される範囲内で、Cypress は、別途通知することなく、本書面を変更する権利を留保する。Cypress は、本書面に記載のある、いかなる製品若しくは回路の適用又は使用から生じる一切の責任を負わない。本書面で提供されたあらゆる情報 (あらゆるサンプルデザイン情報又はプログラムコードを含む) は、参照目的のためのみに提供されたものである。この情報で構成するあらゆるアプリケーション及びその結果としてのあらゆる製品の機能性及び安全性を適切に設計、プログラム、かつテストすることは、本書面のユーザーの責任において行われるものとする。Cypress 製品は、兵器、兵器システム、原子力施設、生命維持装置若しくは生命維持システム、蘇生用の設備及び外科的移植を含むその他の医療機器若しくは医療システム、汚染管理若しくは有害物質管理の運用のために設計され若しくは意図されたシステムの重要な構成部分としての使用、又は装置若しくはシステムの不具合が人身傷害、死亡若しくは物的損害を生じさせるようなその他の使用 (以下「本目的外使用」という。) のためには設計、意図又は承認されていない。重要な構成部分とは、その不具合が装置若しくはシステムの不具合を生じさせるか又はその安全性若しくは実効性に影響すると合理的に予想できるような装置若しくはシステムのあらゆる構成部分をいう。Cypress 製品のあらゆる本目的外使用から生じ、若しくは本目的外使用に関連するいかなる請求、損害又はその他の責任についても、Cypress はその全部又は一部を問わず一切の責任を負わず、かつ Cypress はそれら一切から本書により免除される。Cypress は Cypress 製品の目的外使用から生じ又は本目的外使用に関連するあらゆる請求、費用、損害及びその他の責任 (人身傷害又は死亡に基づく請求を含む) から免責補償される。

Cypress、Cypress のロゴ、Spansion、Spansion のロゴ及びこれらの組み合わせ、WICED、PSoC、CapSense、EZ-USB、F-RAM、及び Traveo は、米国及びその他の国における Cypress の商標又は登録商標である。Cypress のより完全な商標のリストは、cypress.com を参照すること。その他の名称及びブランドは、それぞれの権利者の財産として権利主張がなされている可能性がある。