



μC3 と STM32CubeMX の共存ガイド

改訂記録

Rev.	発行日	改訂内容	
		章番号	内容
2.0	2026.02.19	—	全面的に改訂
2.1	2026.04.06	1.3	CPU 名称の表記をデータシートに従い修正

目次

改訂記録	2
目次.....	3
1 はじめに.....	4
1.1 概要.....	4
1.2 注意事項	4
1.3 動作確認条件	5
1.4 フォルダ構成	5
2 基本方針.....	6
3 起動シーケンス.....	7
3.1 μ C3 の起動シーケンス.....	7
3.2 μ C3 と STM32CubeMX の共存例の起動シーケンス	8
4 共存手順.....	9
4.1 プロジェクトの作成	9
4.1.1 STM32CubeMX プロジェクトの作成	9
4.1.2 EWARM プロジェクトの作成	10
4.2 μ C3 のソースコードの変更	13
4.2.1 リセットハンドラ.....	13
4.2.2 クロック	14
4.2.3 SysTick.....	14
4.2.4 main 関数	15
4.3 STM32CubeMX のソースコードの変更.....	16
4.3.1 main 関数	16
5 共存例の動作手順	17
6 補足事項.....	18
6.1 STM32CubeMX でソースコードを再生成する場合.....	18
6.2 割込み関数の登録.....	19
6.3 HAL_Delay で無限ループになる場合	22

1 はじめに

1.1 概要

本資料は、当社の μ C3 と他社 SDK (STM32CubeMX) の組み合わせ例を参考情報として提供するものです。

μ C3 および μ Net3 の使用方法については、ユーザーズガイドをご確認ください。

本資料は IAR Systems 社 Embedded Workbench for ARM を例としていますが、GCC など他のコンパイラ環境についてもリンカ設定ファイルやアセンブラコードの記述を読み替えることで対応可能です。

1.2 注意事項

μ C3 および μ Net3 はプロプライエタリなソフトウェアであり、他社 SDK との組み合わせを前提とした設計・保証は行っておりません。組み合わせ利用については、ユーザーの判断と責任にてご対応ください。

μ Net3 は弊社から提供する MAC コントローラドライバを使用することを推奨します。

本資料での記載内容では、周辺ペリフェラルを使用せずに μ C3 の起動のみを確認しています。資料内に記載の動作確認環境で検証を行っておりますが、すべての環境やバージョンでの動作を保証するものではありません。

本資料に関するお問い合わせは、当社サポート窓口までご連絡ください（保守サービス契約が必要です）。なお、本資料に含まれない応用的な使用方法や異なるハードウェア・ソフトウェア環境での動作については、サポートできない場合があります。また、他社 SDK 自体に関するお問合せは SDK 提供元にお問合せください。

1.3 動作確認条件

本資料のサンプルは下記の条件で動作確認しています。

項目	説明
μ C3（評価版） ¹	➤ μ C3/Compact + μ Net3 Cortex-M7 STM32H7 シリーズ EWARM 評価版 Release 1.0.1 ➤ μ C3/Standard Cortex-M7 STM32H7 シリーズ EWARM 評価版 Release 1.0.2
CPU	STMicroelectronics 社 STM32H743ZIT6 (Cortex-M7)
コンパイラ IDE	IAR Systems 社 Embedded Workbench for ARM ² v8.50.9
JTAG エミュレータ	オンボード ST-LINK
使用ボード	STMicroelectronics 社 NUCLEO-H743ZI2
SDK	STM32CubeMX 6.16.0
ターミナルソフト	Tera Term 4.104

1.4 フォルダ構成

本資料と共存例のフォルダ構成は、以下の通りです。

.	
uC3_STM32CubeMX_IntegrationGuide.docx.pdf	... 本資料
└─Sample	... μ C3 と STM32CubeMX の共存例
└─Compact	
└─STM32H7	
└─M7	
└─NUCLEO-H743ZI.CUBEMX	... μ C3/Compact の共存例
└─Standard	
└─STM32H7	
└─NUCLEO-H743ZI.UART.CUBEMX	... μ C3/Standard の共存例

¹ <https://www.eforce.co.jp/download/> よりダウンロードできます。

² EWARM と略します。

2 基本方針

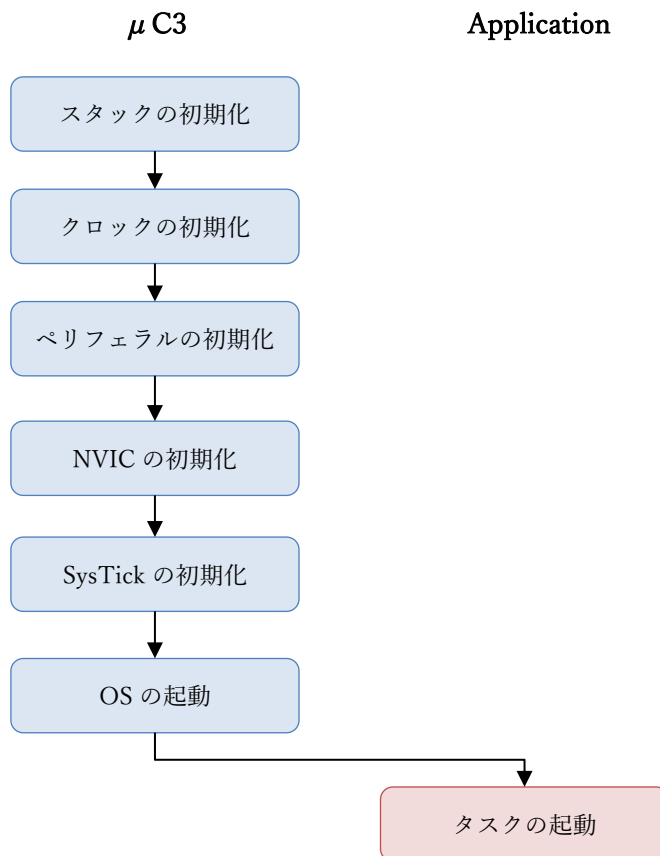
μ C3 と STM32CubeMX を組み合わせて利用するための基本的な方針について説明します。各機能について、 μ C3 と STM32CubeMX のどちらが主として管理すべきか、利用する際に留意すべきことを説明します。

項目	主担当	説明
リセットハンドラ	μ C3	➤ スタックの初期化等を行うため OS 側で管理する。
ベクタテーブル・割込み	μ C3	➤ 割込みは OS の排他制御等で利用するため、OS 側で割込みの有効・無効、割込みレベルの設定、ベクタテーブルの管理を行う。 ➤ 割込み関数は、OS で管理するため、割込みサビスルーチン（または割込みハンドラ）として定義する。 ➤ OS 起動直後は割込みが無効となるため、最初に起動するタスクや初期化ハンドラ等で適宜割り込みを有効化する。
SysTick	μ C3	➤ OS の時間管理のため、OS 付属のドライバを使用する。 ➤ STM32CubeMX へのチック供給処理を追加する。
リンカ設定ファイル	μ C3	➤ OS で必要なセクション定義を行う。
クロック	任意	➤ STM32CubeMX のコードを使用してもよい。
その他ペリフェラル	任意	➤ STM32CubeMX のコードを使用してもよい。ただしオプション製品（ μ Net3 や μ C3-FileSystem 等）を利用する場合は、同梱のドライバの使用を推奨する。 ➤ ヘッドファイルのレジスタ定義等が競合する場合、どちらかを無効化するなど調整する。

3 起動シーケンス

3.1 μ C3 の起動シーケンス

μ C3 単体での起動シーケンスは以下の通りです。

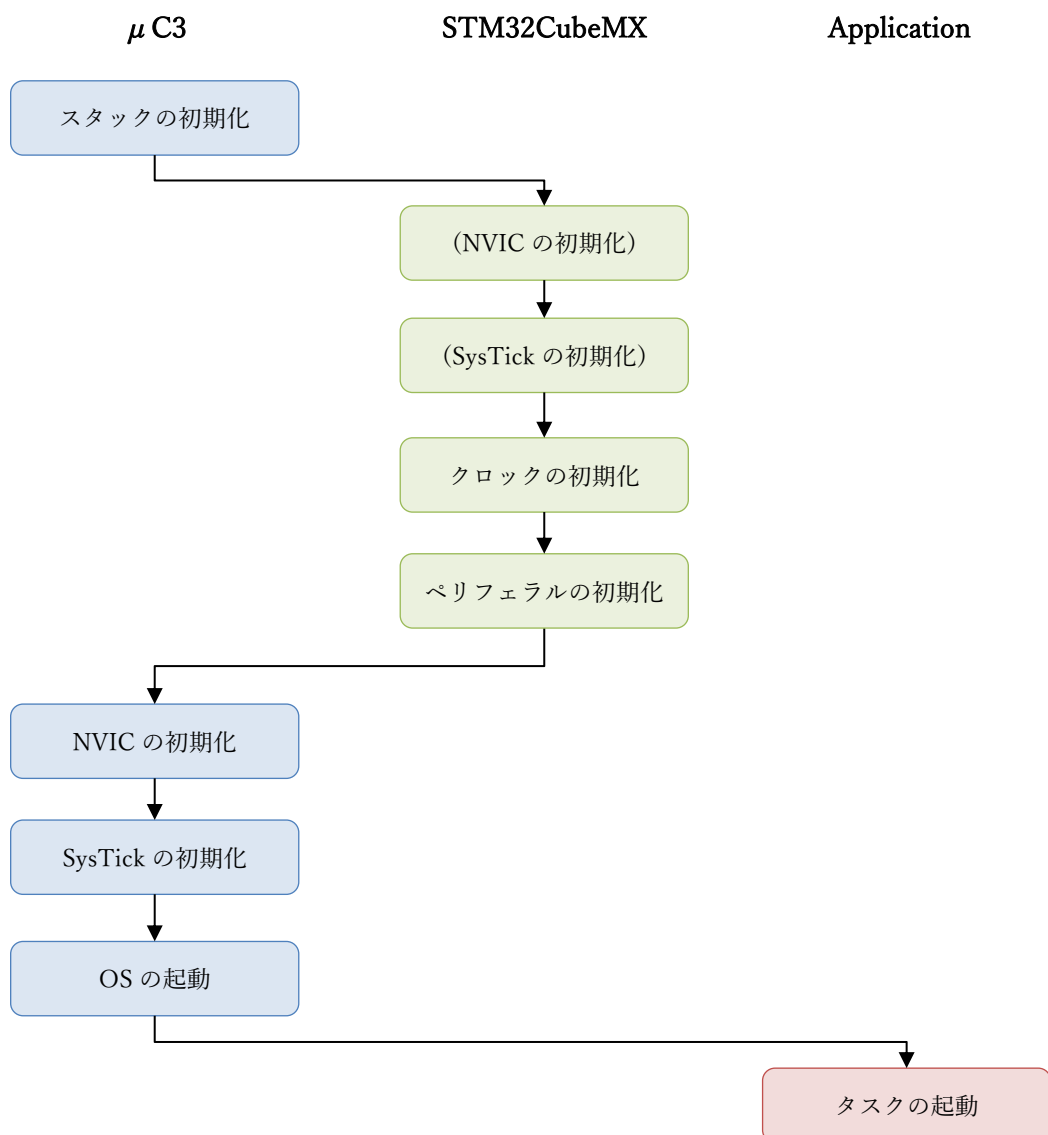


3.2 μ C3 と STM32CubeMX の共存例の起動シーケンス

μ C3 と STM32CubeMX の共存例の起動シーケンスは以下の通りです。

μ C3 のリセットハンドラでスタックの初期化を実施後、STM32CubeMX の初期化処理を実行し各種ペリフェラルやクロックの初期化を実行します。

その後、 μ C3 の初期化処理を実行します。NVIC や SysTick は μ C3 で管理するため、ここで再度初期化を実行します。最後に OS の起動を実行し、タスクを起動します。



4 共存手順

ここでは「3. 2 μ C3 と STM32CubeMX の共存例の起動シーケンス」を実現するためのプロジェクト作成手順、ソースコードの変更手順を説明します。

例として、 μ C3/Compact の設定例・ソースコードを示します。 μ C3/Standard でも基本的に同様の対応となりますが、詳細は μ C3/Standard の共存例のプロジェクトを参照してください。

4.1 プロジェクトの作成

4.1.1 STM32CubeMX プロジェクトの作成

STM32CubeMX にて以下の操作を行い、プロジェクトを作成します。

1. [File] > [New Project]を選択し、CPU 型番を選択して新規 STM32CubeMX プロジェクトを作成します。
2. 必要に応じてクロックやペリフェラルの設定を行います。
3. [File] > [Save Project As..]を選択し、STM32CubeMX プロジェクトを保存します。以下このフォルダを STM32CubeMX プロジェクトフォルダと呼びます。

例：

C:\uC3¥Sample¥Compact¥STM32H7¥M7¥NUCLEO-H743ZI.CUBEMX

C:\uC3¥Sample¥Standard¥STM32H7¥NUCLEO-H743ZI.CUBEMX

4. 画面右上の[GENERATE CODE]を選択し、ソースコードおよび EWARM プロジェクトを生成します。

4.1.2 EWARM プロジェクトの作成

「4.1.1 STM32CubeMX プロジェクトの作成」で生成した EWARM プロジェクトに対して、以下の変更を行います。

1. μ C3 関連のソースファイル、ライブラリ、リンカ設定ファイルを STM32CubeMX プロジェクトフォルダ配下に格納します。

例：

(STM32CubeMX プロジェクトフォルダ)¥uC3

2. EWARM を起動し、[ファイル]>[ワークスペースを開く]を選択し、EWARM プロジェクトを開きます。

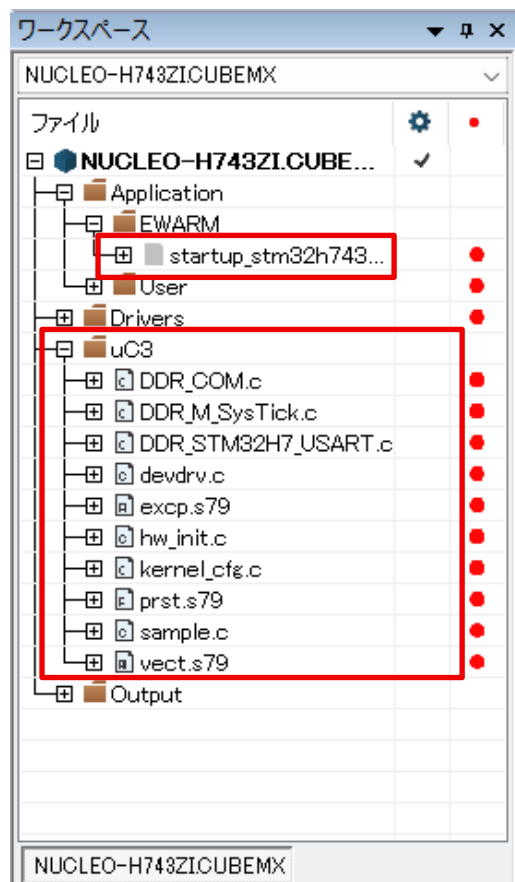
例：

(STM32CubeMX プロジェクトフォルダ)¥EWARM¥Project.eww

3. μ C3 関連のソースファイルを登録します。

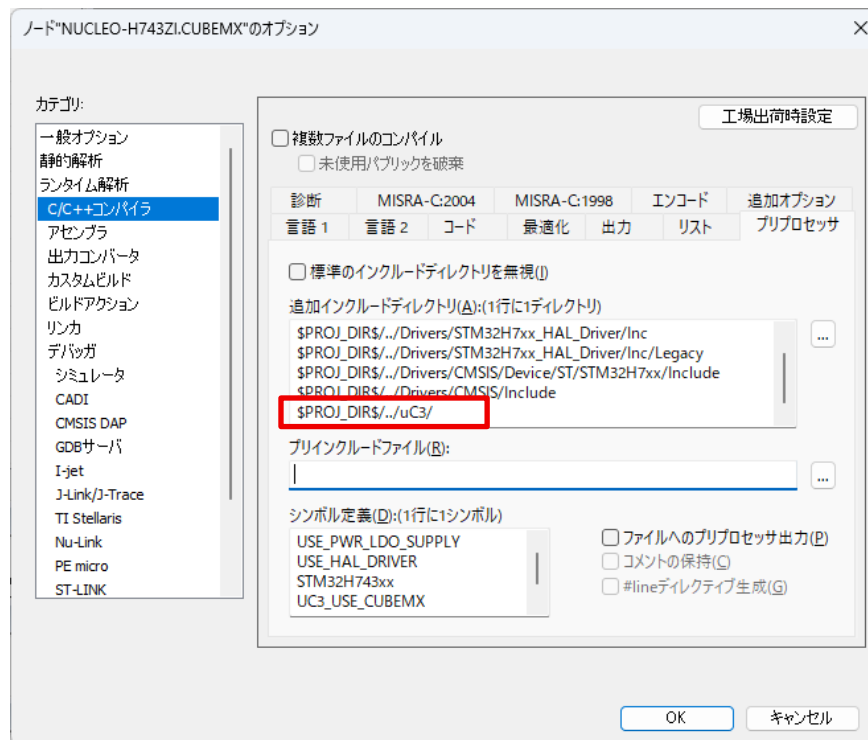
STM32CubeMX のベクタテーブルのソースファイルをビルドから除外します。

例：



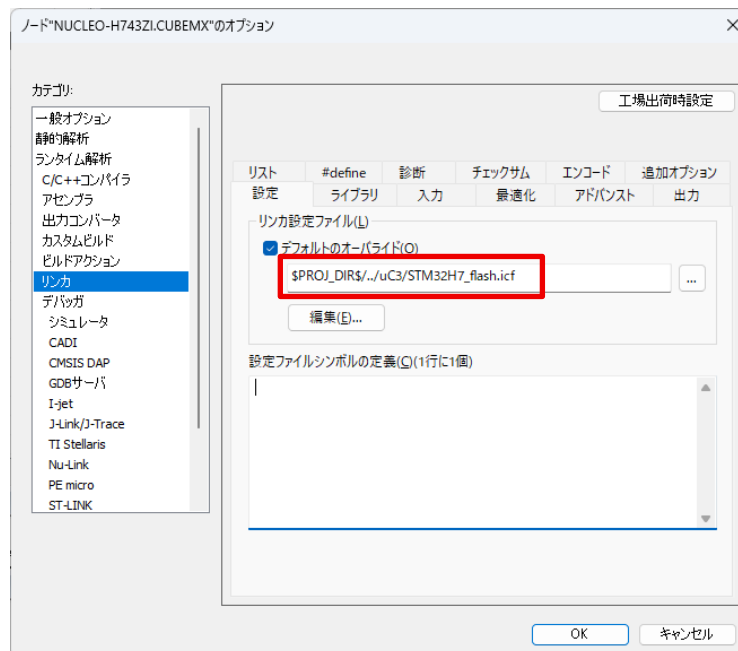
4. インクルードディレクトとして μ C3 関連のソースファイルを追加します。

例：



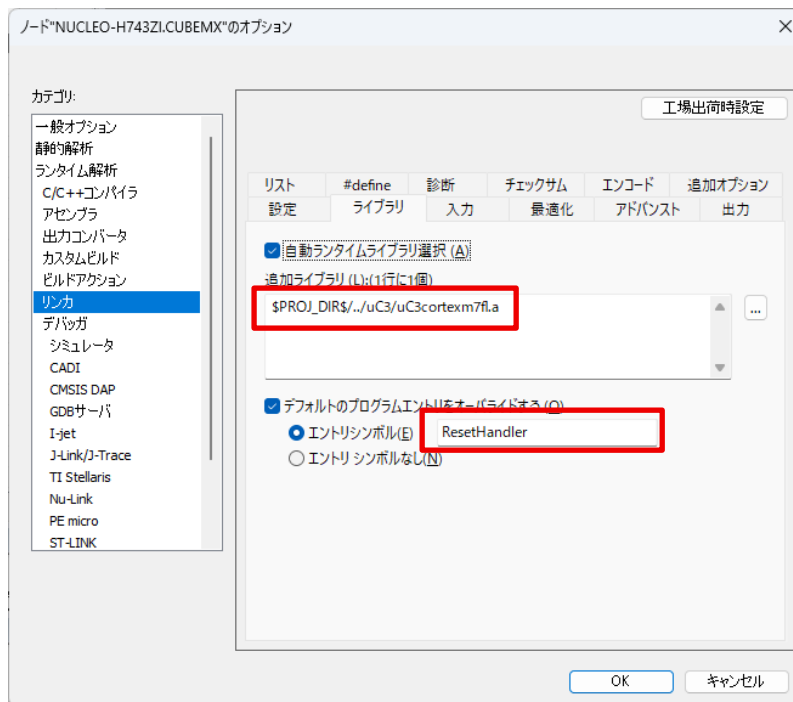
5. μ C3 のリンク設定ファイルを指定します。

例：



6. μ C3 のライブラリ、リセットハンドラを指定します。

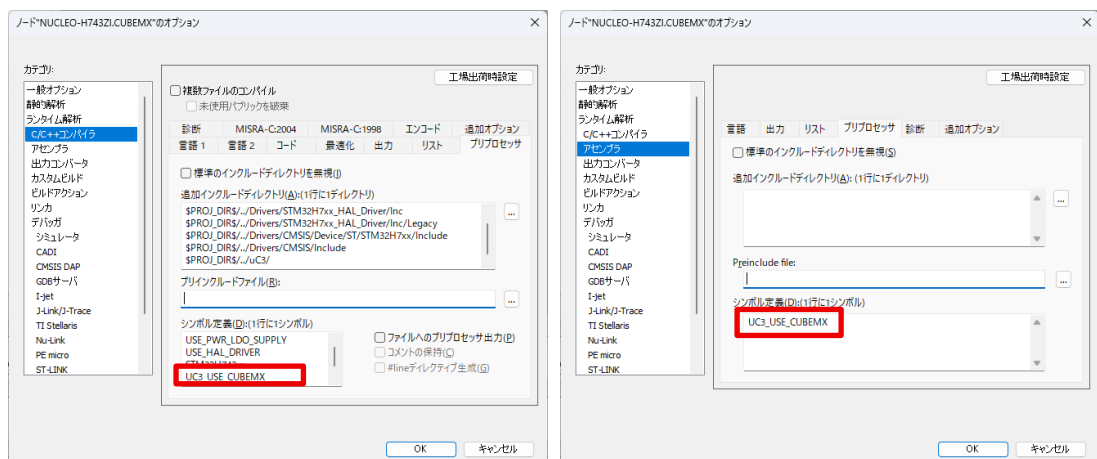
例：



7. 変更箇所を分かりやすくするため、シンボル定義「UC3_USE_CUBEMX」を追加します。この定義は任意です。

また μ C3/Standard の場合、「DISABLE_HOOK」を追加します。

例：



4.2 μ C3 のソースコードの変更

μ C3 のソースコードの変更箇所について説明します。

4.2.1 リセットハンドラ

μ C3 のリセットハンドラに対して以下の変更を行います。

- STM32CubeMX の初期化関数 `SystemInit` の呼び出しを追加
- クロックやペリフェラルの初期化処理を削除

設定例：prst.s79

```
#ifdef UC3_USE_CUBEMX
    IMPORT SystemInit
#endif
IMPORT __iar_program_start

PUBLIC ResetHandler
ResetHandler:
    mrs    r0, control
    ldr    r1, =SFE(CSTACK)
    orr    r0, r0, #0x02
    ldr    r2, =SFE(HSTACK)
    bic    r0, r0, #0x01
    msr    psp, r1
    msr    msp, r2
    msr    control, r0

#ifdef UC3_USE_CUBEMX
    ldr    r0, =SystemInit
    blx    r0
#else
    (クロックの設定)
    (GPIO の設定)
#endif

; Enter the iar code
    ldr    r0, =__iar_program_start
    bx     r0

END
```

4.2.2 クロック

STM32CubeMX のクロック設定にあわせて、 μ C3 のドライバで使用するクロック周波数の指定を行います。 μ C3 コンフィグレータが同梱されている場合は、ソースコードを直接変更せずに、 μ C3 コンフィグレータでクロックの指定可能です。

設定例：DDR_M_SysTick_cfg.h

```
#define RELOADV      64000UL    /* Divider ratio per 1msec */
```

設定例：DDR_STM32H7_USART_cfg.h

```
#define CFG_CLK_PCLK1      (32000000UL)
#define CFG_CLK_PCLK2      (32000000UL)
#define CFG_CLK_HSI        (64000000UL)
#define CFG_CLK_CSI        (40000000UL)

#define USART_3
#define CFG_USART3_SRC      0U
#define CFG_USART3_CLK      (CFG_CLK_PCLK1)
```

4.2.3 SysTick

μ C3 の SysTick ドライバを変更し、STM32CubeMX へのチック供給関数 HAL_IncTick の呼び出しを追加します。

設定例：DDR_M_SysTick.c

```
void _ddr_m_systick(VP_INT exinf)
{
    if ((REG_SYSTICK.CTRSTS & 0x00010000UL) != 0UL) {
#ifdef UC3_USE_CUBEMX
        extern void HAL_IncTick(void);
        HAL_IncTick();
#endif
        (void)isig_tim();
    }
}
```

4.2.4 main 関数

STM32CubeMX 側の main 関数から、 μ C3 側の main 関数を呼び出すために、main をリネームします。

また STM32CubeMX 側で SysTick が動作している可能性があるため、 μ C3 側で再初期化を行う前に一度停止させる処理を追加します。

設定例：sample.c

```
#ifdef UC3_USE_CUBEMX
int uc3_entry(void)
#else
int main(void)
#endif
{
    ER ret = E_OK;

#ifdef UC3_USE_CUBEMX
    REG_SYSTICK.CTRSTS &= ~0x00000001UL;
    REG_SNVIC.ICSR = 0x02000000U; // Clear pending SysTick (PENDSTCLR)
#endif

    /* Initialize hardware */
    cortex_m_init_peripheral();
    init_peripheral();

    /* Generate system and start */
    ret = start_uC3();

    return ret;
}
```

4.3 STM32CubeMX のソースコードの変更

4.3.1 main 関数

STM32CubeMX の main 関数から、 μ C3 の main 相当の関数の呼び出しを追加します。

設定例：main.c

```
int main(void)
{
  ...略...
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
#ifdef UC3_USE_CUBEMX
    extern int uc3_entry(void);
    uc3_entry();
#endif
  }
  /* USER CODE END 3 */
}
```

5 共存例の動作手順

「1.4 フォルダ構成」に記載の共存例について、動作手順を説明します。

1. 評価ボードの CN1 とホスト PC を USB ケーブルで接続します。
2. ホスト PC で TeraTerm を起動し、以下で評価ボードの仮想 COM ポートに接続します。

スピード(E):	115200	▼
データ(D):	8 bit	▼
パリティ(A):	none	▼
ストップビット(S):	1 bit	▼
フロー制御(F):	none	▼

3. EWARМ で下記プロジェクトを開きます。
(共存例フォルダ)¥ EWARМ¥Project.eww
4. EWARМ の[プロジェクト]>[ダウンロードしてデバッグ]をクリックします。
プロジェクトのビルドとダウンロードが行われ、プログラムが実行されます (main 関数で停止する)。
5. EWARМ の[デバッグ]>[実行]をクリックし、プログラムを実行します。

プログラムが実行されると、TeraTerm に文字列を表示します。以降は、入力された文字列をエコーバックします (μC3/Compact の場合は、Enter キーが押下されるか文字数の上限に達したときに入力文字をエコーバック)。

また、500ms 間隔で LED を点滅します。



6 補足事項

6.1 STM32CubeMX でソースコードを再生成する場合

STM32CubeMX で生成されたソースコード内のコメント「USER CODE BEGIN」から「USER CODE END」までの間にユーザコードを追加することで、STM32CubeMX でソースコードを再生成した場合にユーザコードが保持されます。

それ以外の箇所への変更は上書きされる可能性があるため、適宜バックアップしてからソースコードを再生成することを推奨します。

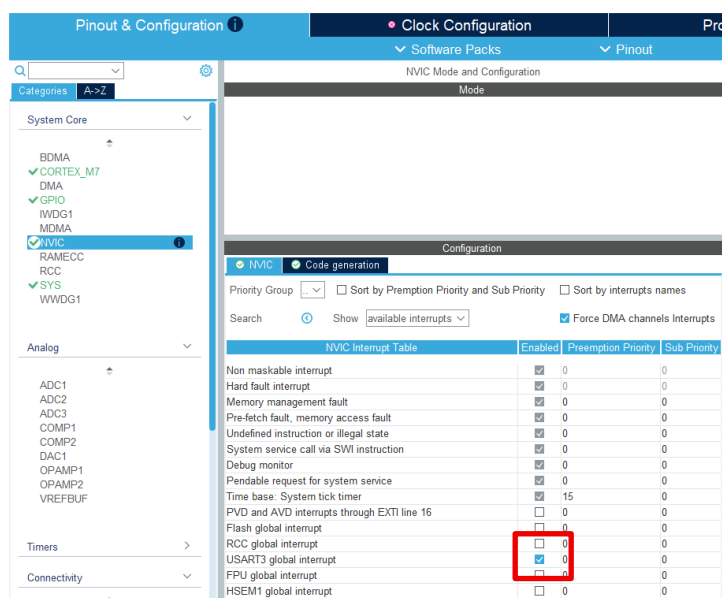
6.2 割込み関数の登録

STM32CubeMX の割込み関数を使用する場合、 μ C3 へ割込みサービスルーチン（または割込みハンドラ）として登録が必要です。

例として、STM32CubeMX の USART3 の割込み関数を割込みサービスルーチンとして登録し使用する手順を説明します。

1. STM32CubeMX の NVIC の設定で、USART3 の割込みを有効にして、ソースコードを再生成します。

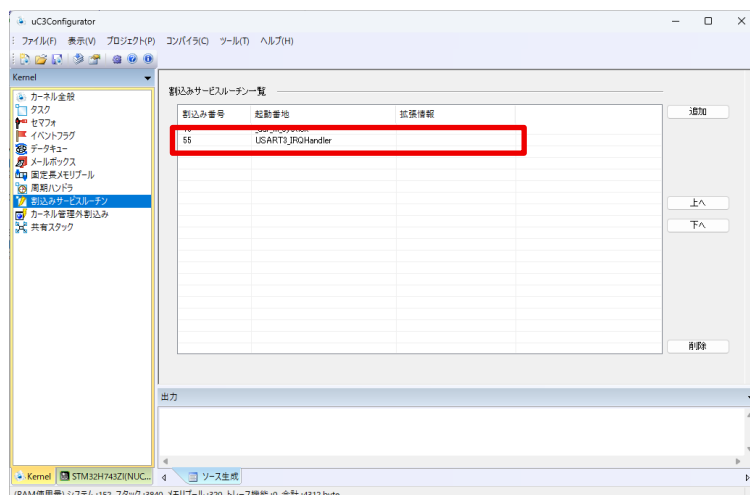
割込みのレベルは μ C3 で管理するため、「Preemption Priority」「Sub Priority」の設定は不要です。



2. STM32CubeMX の USART3 の割り込み関数 USART3_IRQHandler を割り込みサービスルーチンとして登録します。

(A) コンフィグレータで登録する場合 (μC3/Compact および一部の μC3/Standard)

例：



(B) システムコールで登録する場合 (μC3/Standard)

例：

```
/*
 * 初期化ハンドラ
 */
static void initpr(void)
{
    ...略...
#ifdef UC3_USE_CUBEMX
    extern void USART3_IRQHandler(void);

    const T_CISR cisr = {
        (TA_HLNG | TA_FPU),
        (VP_INT)0,
        55, // 割り込み番号
        (FP)USART3_IRQHandler,
        224
    };

    (void)acore_isr((T_CISR *)&cisr);
#endif
}
```

3. μ C3 起動直後は各割込みが無効となっているため、USART3 の割込みを有効にします。
また各割込みの割込みレベルは最低優先度に初期化されるため、必要に応じて割込みレベルを設定します。

例：

```
void MainTask(VP_INT exinf)
{
#ifdef UC3_USE_CUBEMX
    vset_ip1(55, 224); // 割込みレベルの設定
    ena_int(55);      // 割込みの有効化
#endif
...略...
}
```

6.3 HAL_Delay で無限ループになる場合

一部のペリフェラルを有効にした場合に、STM32CubeMX 側の main 関数内の初期化処理にて、STM32CubeMX の関数 HAL_Delay で指定時間待つコードが出力される場合があると報告がありました。「4 共存手順」では、そのようなケースは考慮していないため、HAL_Delay で無限ループとなります。

以下のように追加で、STM32CubeMX 側の初期化処理中に、STM32CubeMX 側の割込みを利用する設定が必要となります。

1. 「4. 1. 2 EWARM プロジェクトの作成」でビルドから除外した STM32CubeMX のベクタテーブルのソースファイルをビルド対象とします。
2. μ C3 の定義との競合を回避するため、STM32CubeMX のベクタテーブルをリネームします。

コード例：startup_stm32h743xx.s

```
SECTION .intvec:CODE:NOROOT(2)

EXTERN __iar_program_start
EXTERN SystemInit
EXTERN ExitRun0Mode

#ifdef UC3_USE_CUBEMX
    SECTION .text:CODE:NOROOT(8)
    PUBLIC __vector_table_cubemx

    DATA
__vector_table_cubemx
#else
    PUBLIC __vector_table

    DATA
__vector_table
#endif
DCD sfe(CSTACK)
DCD Reset_Handler ; Reset Handler
```

3. STM32CubeMX の関数 SystemInit の最後で、STM32CubeMX 側のベクタテーブルを設定します。

コード例：system_stm32h7xx.c

```
void SystemInit (void)
{
    ...略...

    #ifdef UC3_USE_CUBEMX
        extern uint32_t __vector_table_cubemx[];
        SCB->VTOR = (uint32_t)__vector_table_cubemx;
    #endif
}
```

4. (μC3/Standard の場合)

μC3 側のリセットハンドラの手前で、割込みを無効にしている処理を削除します。

コード例：prst.s79

```
...略...
        PUBLIC ResetHandler
ResetHandler:
    #ifndef UC3_USE_CUBEMX
        cpsid    i                ; set PRIMASK
    #endif
    ...略...
```

μ C3 と STM32CubeMX の共存ガイド

イー・フォース株式会社 <https://www.eforce.co.jp/>

Copyright (C) 2026 eForce Co.,Ltd. All Rights Reserved.