

μ T-Kernel 3.0
構築手順書 (NUCLEO-L476 +
EthernetShield2)

Rev 3.00.01

May, 2023



μT-Kernel 3.0

目次

1. はじめに	2
1.1. 本書について	2
1.2. 表記について	2
2. 概要	3
2.1. 対象とするハードウェアとソフトウェア	3
2.2. 対象とする開発環境	4
2.3. デバイスドライバ	4
2.4. 関連ドキュメント	4
3. 開発環境の準備	6
3.1. STM32CubeIDE のインストール	6
4. プロジェクトの作成	7
4.1. μT-Kernel 3.0 を Github からクローン	7
4.2. ブランチを NUCLEO-L476RG lwIP 対応版に変更	9
4.3. NUCLEO-L476RG lwIP のサブモジュールの初期化	10
4.4. NUCLEO-L476RG lwIP のプロジェクトをワークスペースに追加	12
4.5. NUCLEO-L476RG と EthernetShield2 の接続	15
4.6. ビルド、実行	17
5. μT-Kernel 3.0 のディレクトリ／ファイル構成	25
5.1. μT-Kernel 3.0 のソースコード	25

1. はじめに

本品はユーシーテクノロジー(株)が移植した NUCLEO-L476RG(NUCLEO-L476)対応の μT-Kernel 3.0 である。



本品は、トロンプフォーラムの配布する μT-Kernel 3.0 をベースに、STMicroelectronics 製の評価ボード NUCLEO-L476RG に、Arduino 社製 Ethernet Shield 2(以降、EthernetShield2)を搭載して動作させるための機種依存部を追加してある。

1.1. 本書について

本書は NUCLEO-L476RG 向けの μT-Kernel 3.0 において EthernetShield2 のアプリケーションを開発する際の構築手順について記載した構築手順書である。本書の対象となる実装は、ユーシーテクノロジー(株)が公開している μT-Kernel 3.0 BSP(Board Support Package)に含まれている。

以降、単に OS や RTOS と称する場合は μT-Kernel 3.0 を示し、**本実装**と称する場合は NUCLEO-L476RG 向けの μT-Kernel 3.0 のソースコードの実装を示すものとする。

1.2. 表記について

表記	説明
[]	[]はソフトウェア画面のボタンやメニューを表す。
「 」	「 」はソフトウェア画面に表示された項目などを表す。
	注意が必要な内容の場合に記述する。
	補足やヒントなどの内容の場合に記述する。
<TARGET>	ターゲットボード用のディレクトリ名を表す。
<CPU>	CPU 用のディレクトリ名を表す。
<CORE>	CPU コア用のディレクトリ名を表す。

2. 概要

本書では、μ T-Kernel 3.0 BSP の使用方法について説明する。

μ T-Kernel 3.0 BSP は、特定のマイコンボード等のハードウェアに対して移植した μ T-Kernel 3.0 の開発および実行環境一式を提供するものである。

2.1. 対象とするハードウェアとソフトウェア

開発対象のハードウェアおよびソフトウェアは以下である。

表 2-1 開発対象のハードウェアとソフトウェア

分類	名称	備考
マイコン	STM32L476RGT6U (ARM Cortex-M4F)	STMicroelectronics NV
OS	μ T-Kernel 3.00.06	トロンフォーラム
実機 (マイコンボード)	NUCLEO-L476RG (NUCLEO-L476)	STMicroelectronics NV
イーサネット シールド	EthernetShield2	Arduino 社
ネットワーク プロトコルスタック	lwIP 2.1.2	オープンソース

- ❶ 本実装の最新版は、ユーシーテクノロジー(株)の GitHub リポジトリにて公開している。

https://github.com/UCTechnology/mtk3_bsp

- ❷ μ T-Kernel 3.0 の最新版は以下の GitHub リポジトリにて公開されている。

https://github.com/tron-forum/mtkernel_3

- ❸ 対象マイコンボード(NUCLEO-L476RG)に関しては STMicroelectronics NV のサイトを参照のこと。

<https://www.st.com/ja/evaluation-tools/nucleo-l476rg.html>

- ❹ 対象イーサネットシールド(EthernetShield2)に関しては Arduino のサイトを参照のこと。

<https://store.arduino.cc/products/arduino-ethernet-shield-2>

- ❺ 対象の TCP/IP プロトコルスタック(lwIP 2.1.2) に関しては lwIP のサイトを参照のこと。

<https://savannah.nongnu.org/projects/lwip/>

2.2. 対象とする開発環境

対象とする開発環境は以下である。

開発を行うホスト PC の OS は Windows とする。動作確認は Windows 11 にて行った。

表 2-2 開発環境

分類	名称	備考
開発環境	STM32CubeIDE Version:1.11.2	STMicroelectronics NV

① バージョンは動作確認に使用したバージョンを示している。

2.3. デバイスドライバ

μT-Kernel 3.0 BSP では、トロンフォーラムが提供する μT-Kernel 3.0 のサンプル・デバイスドライバを、対象となる実機に移植して実装している。

以下に本実装に含まれるデバイスドライバを示す。

表 2-3 本実装に含まれるデバイスドライバ

種別	デバイス名	デバイス
UART	serb	USART2
Net	neta	EthernetShield2

2.4. 関連ドキュメント

表 2-4 関連ドキュメント一覧

分類	名称	発行
OS 仕様	μT-Kernel 3.0 仕様書 (Ver.3.00.01)	トロンフォーラム TEF020-S004-3.00.00
デバイスドライバ	μT-Kernel 3.0 デバイスドライバ 説明書(Ver.1.00.2)	トロンフォーラム TEF033-W007-210331
実装仕様書	μT-Kernel 3.0 実装仕様書 (NUCLEO-L476)	ユーシーテクノロジー(株)
構築手順書 (本書)	μT-Kernel 3.0 構築手順書 (NUCLEO-L476+ EthernetShield2)	ユーシーテクノロジー(株)

① トロンフォーラムが発行するドキュメントは、トロンフォーラムの Web ページ、または GitHub で公開する μT-Kernel 3.0 のソースコードに含まれている。

<https://www.tron.org/ja/specifications/>

https://github.com/tron-forum/mtkernel_3

https://tron-forum.github.io/mtk3_spec_jp/index.html

- ① ユーシーテクノロジー(株)が発行するドキュメントは、ユーシーテクノロジー(株)の GitHub で公開する μT-Kernel 3.0 のソースコードに含まれている。
https://github.com/UCTechnology/mtk3_bsp

3. 開発環境の準備

μT-Kernel 3.0 BSP を使用するにあたり、以下の手順で開発環境の準備を行う。

3.1. STM32CubeIDE のインストール

STMicroelectronics International NV の Web サイト（下記）から STM32CubeIDE をダウンロードする。

<https://www.st.com/ja/development-tools/stm32cubeide.html>

左上の「STM32 用統合開発環境」の下に表示されている [ソフトウェア入手] をクリックするとインストーラの一覧が表示される、この中の STM32CubeIDE-Win の [最新バージョンを取得] をクリックするとライセンス契約が表示される。ライセンス契約に同意して、ユーザー情報を入力すると、インストーラのリンクが入ったメールが送られてきて、それからインストーラがダウンロードできる。

インストーラを実行し、指示に従って STM32CubeIDE のインストールを進める。

- ◆ 2023/02/08 時点での STM32CubeIDE の最新バージョンは、Version 1.11.2 である。本資料では移植作業に使用した Version 1.11.2 を基に説明する。

4. プロジェクトの作成

STM32CubeIDE を利用した NUCLEO-L476RG 用の μ T-Kernel 3.0 lwIP 対応版のプロジェクトの構築手順は以下のとおりである。

- (1) μ T-Kernel 3.0 を Github からクローン
- (2) ブランチを NUCLEO-L476RG lwIP 対応版に変更
- (3) NUCLEO-L476RG lwIP のサブモジュールの初期化
- (4) NUCLEO-L476RG lwIP のプロジェクトをワークスペースに追加
- (5) NUCLEO-L476RG と EthernetShield2 の接続
- (6) ビルド、実行

以下、各作業の詳細について説明する。

4.1. μ T-Kernel 3.0 を Github からクローン

以下の例では、μ T-Kernel 3.0 を Github からクローンする手順を説明する。利用ツールなどによる差異に関しては適宜読み替えること。

- (1) UCT が配布する μ T-Kernel 3.0 の BSP のサイトにアクセスする。
https://github.com/UCTechnology/mtk3_bsp
- (2) [<> Code ▼] を開いて Clone 用の URL をクリップボードにコピーする。

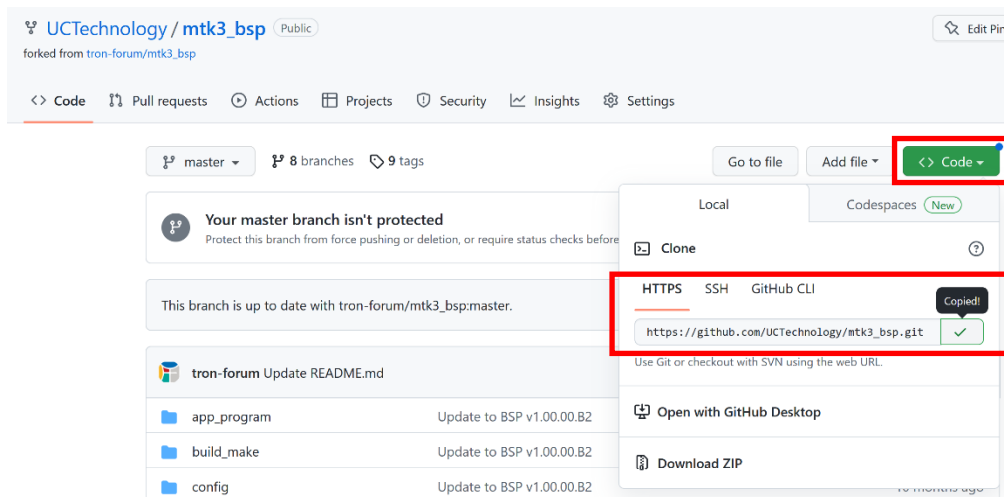


図 1 Clone 用の URL をクリップボードにコピー

- (3) STM32CubeIDE のワークスペースのフォルダをエクスプローラで開く。

(4) 右クリックメニューの[TortoiseGit]から[Git Clone...]を選択する。

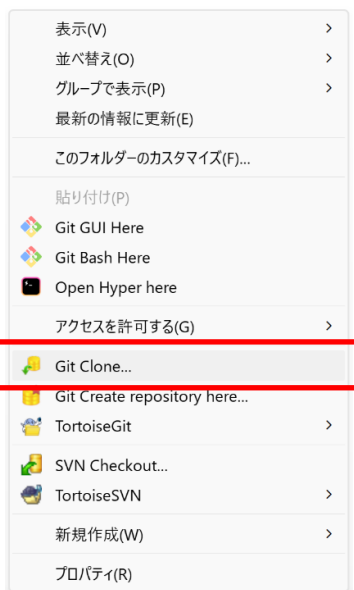


図 2 クローンのメニュー操作

(5) Git Clone ダイアログが表示される。クローン先のパスを プロジェクト名（例では `nucleo_l476_lwip` とする）に変更し、[OK]をクリックする。

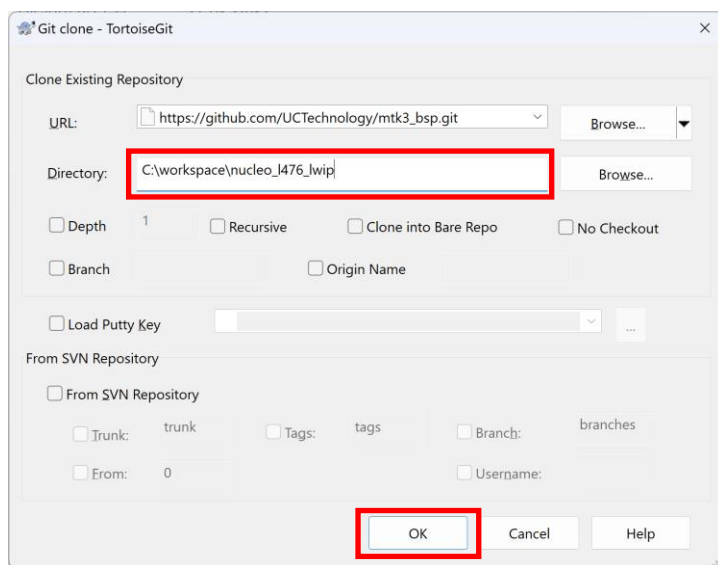


図 3 クローンの設定

(6) ワークスペースのフォルダに `nucleo_l476_lwip` が追加される。

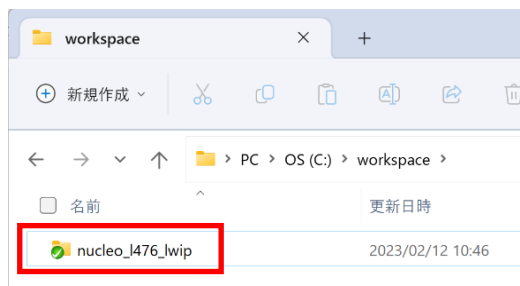


図 4 プロジェクト `nucleo_l476_lwip` 追加後の構成

4.2. ブランチを NUCLEO-L476RG lwIP 対応版に変更

- (1) プロジェクトとして追加した `nucleo_l476_lwip` の右クリックメニューの [TortoiseGit] から [Switch/Checkout...] を選択する。

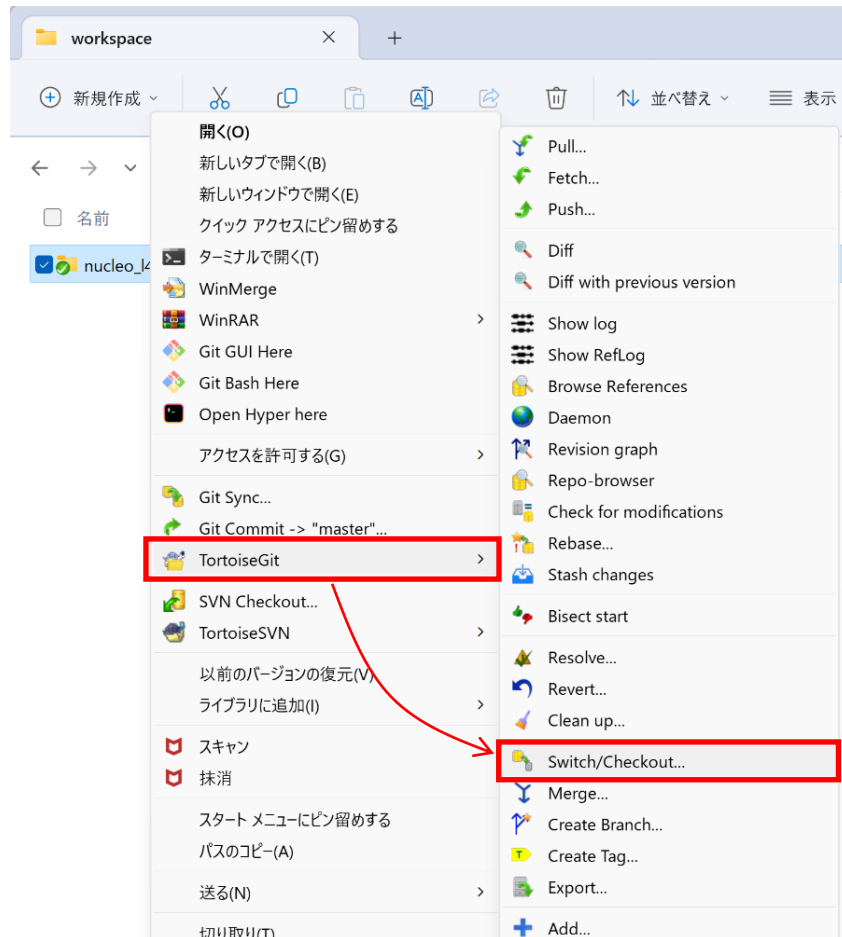


図 5 ブランチ切り替えのメニュー

(2) Switch/Checkout ダイアログが表示される。

ブランチとして `remotes/origin/nucleo_stm32l476_lwip` を一覧の中から選択し、新しいブランチとして `nucleo_stm32l476_lwip` を作成することにして、[OK]をクリックする。

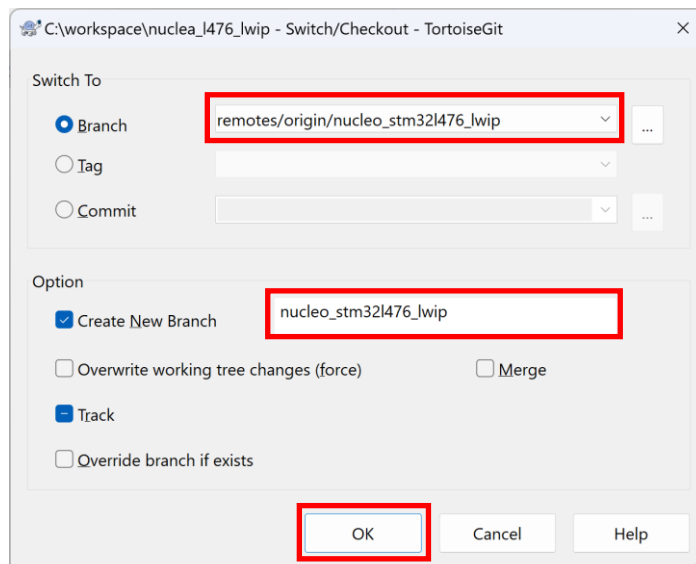


図 6 ブランチの切り替え

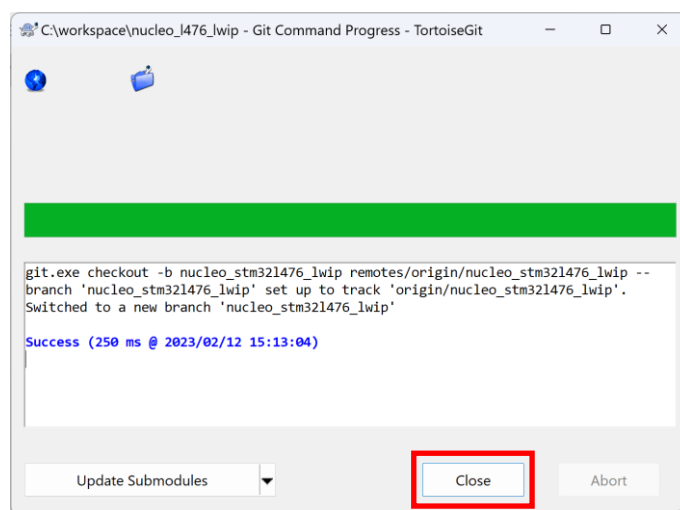


図 7 ブランチの切り替え完了

[Close]をクリックし、ブランチの切り替えが完了すると、NUCLEO-L476 lwIP 用のファイルやディレクトリが追加される。

4.3. NUCLEO-L476RG lwIP のサブモジュールの初期化

NUCLEO-L476RG lwIP はサブモジュールとして、以下の外部のモジュールを利用している。

<https://github.com/lwip-tcpip/lwip.git>

<https://github.com/STMicroelectronics/STM32CubeL4.git>

NUCLEO-L476RG lwIP のプロジェクトを利用するには、上記のサブモジュールを

以下の手順で初期化が必要がある。

- (1) `nucleo_l476_lwip` の右クリックメニューの[TortoiseGit]から[Submodule Update...]を選択する。

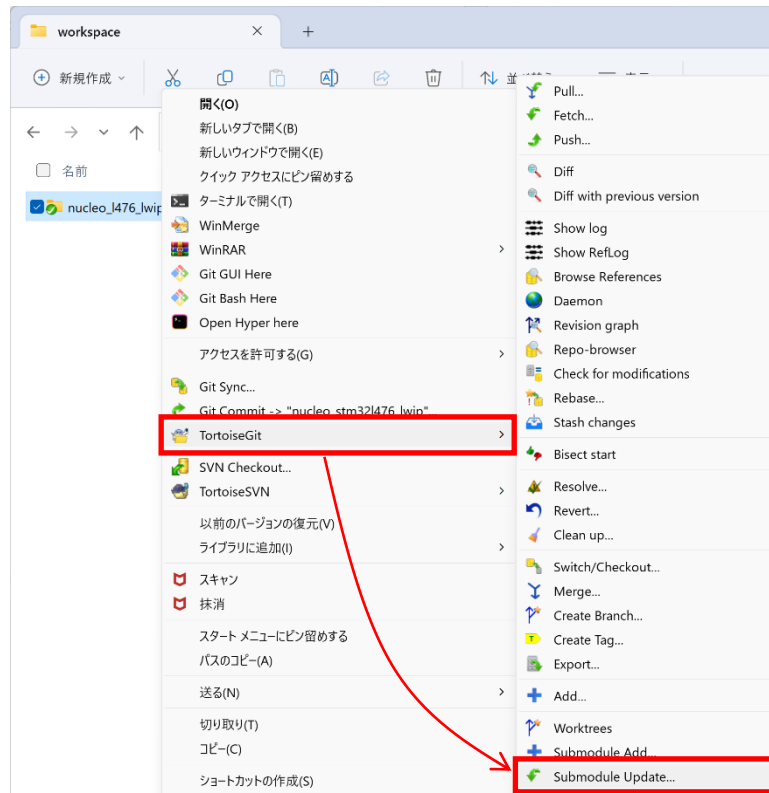


図 8 ブランチの切り替え完了

- (2) Submodule Update ダイアログが表示される。[OK]をクリックするとサブモジュールの初期化が始まる。

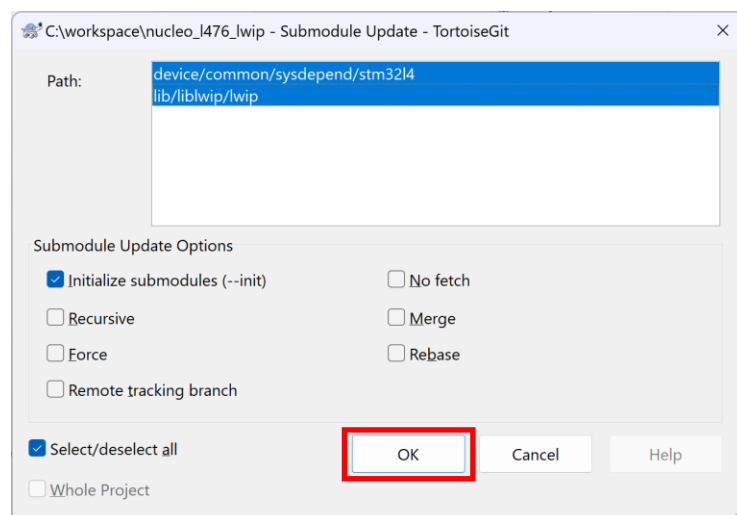


図 9 サブモジュールの初期化

- 💡 `device/common/sysdepend/stm32l4` の更新には 1GiB 以上必要となる。
`workspace` は十分な残量のあるドライブに配置しておくこと。

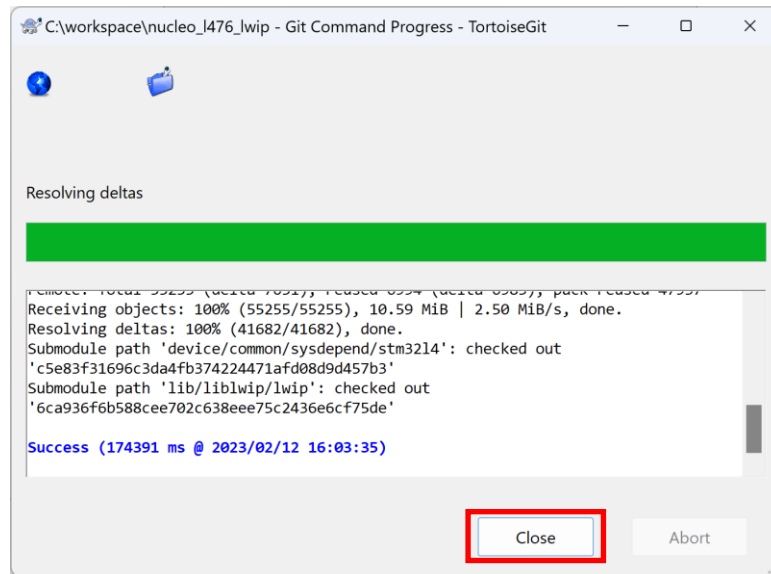


図 10 サブモジュールの初期化完了

[Close] をクリックし、サブモジュールの初期化が完了すると、lwIP と STM32CubeL4 モジュールのファイルやディレクトリが追加される。

4.4. NUCLEO-L476RG lwIP のプロジェクトをワークスペースに追加

- (1) STM32CubeIDE を起動し、workspace のフォルダを選択して、[Launch] をクリックする。

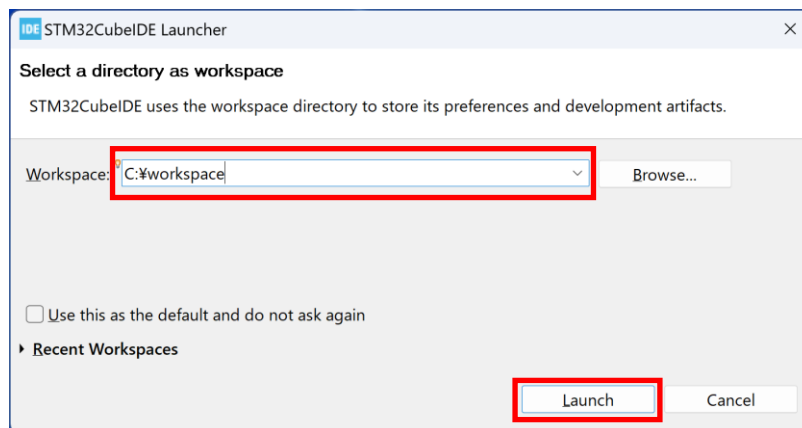


図 11 ワークスペースの選択

(2) [Information Center]タブの[×]をクリックしてタブを閉じる。

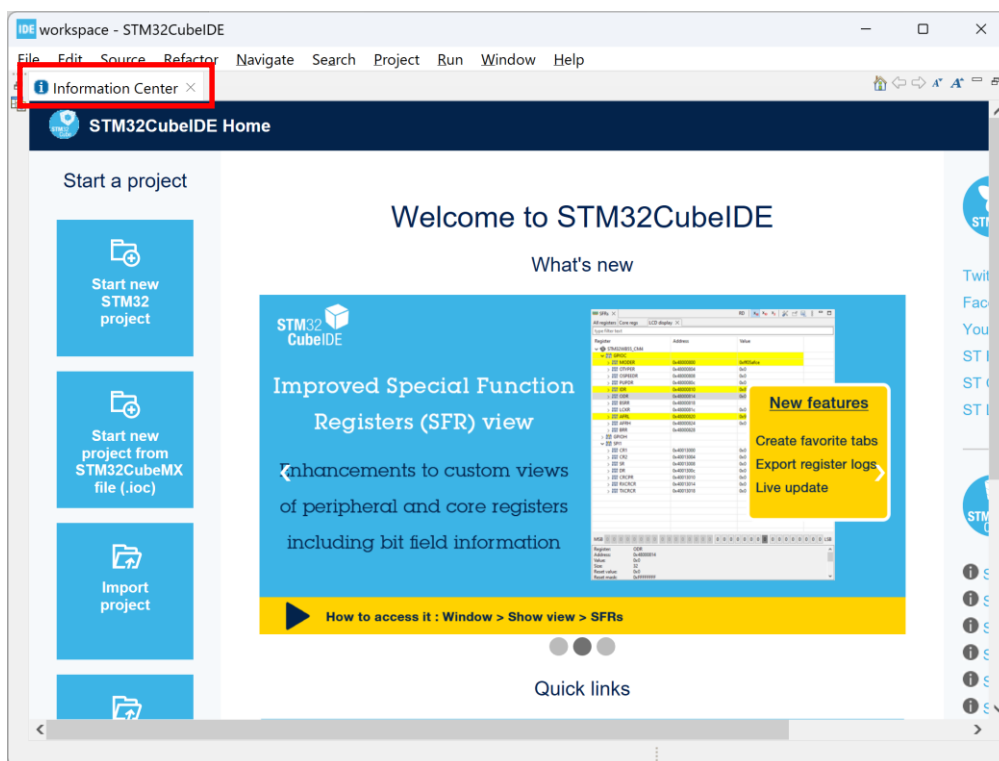


図 12 [Information Center]タブを閉じる

(3) [Project Explorer]の[Import projects...]を選択する。

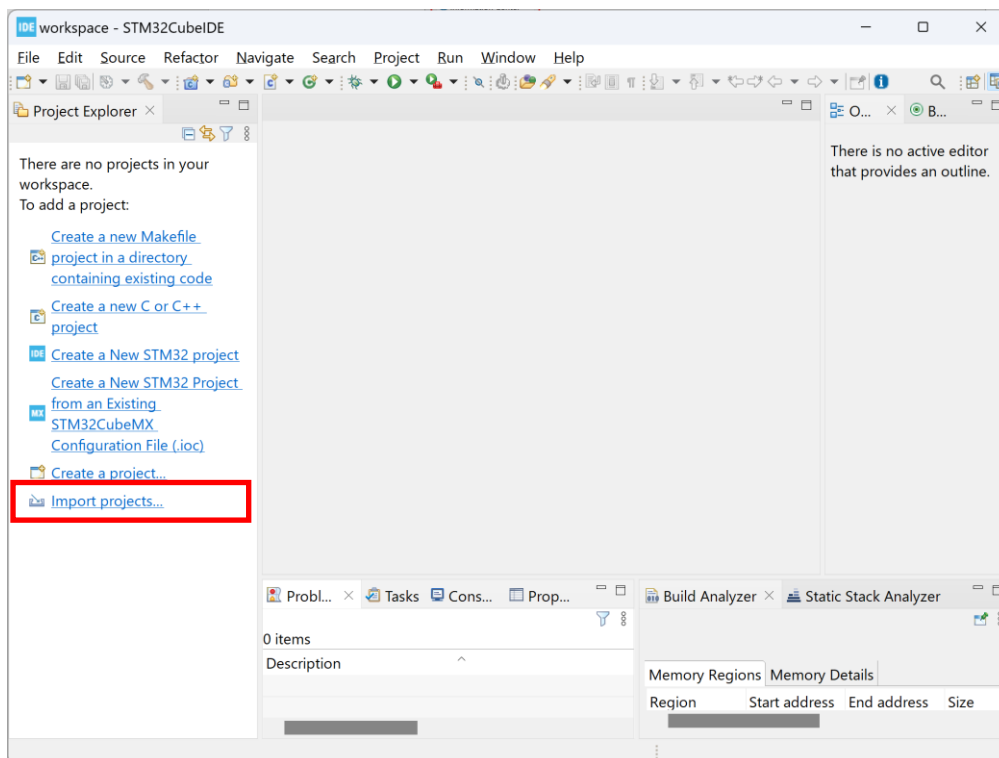


図 13 [Import projects...]を選択

- (4) Import ダイアログの中の [General] の [Existing Projects into Workspace] を選択し、[Next] をクリックする。

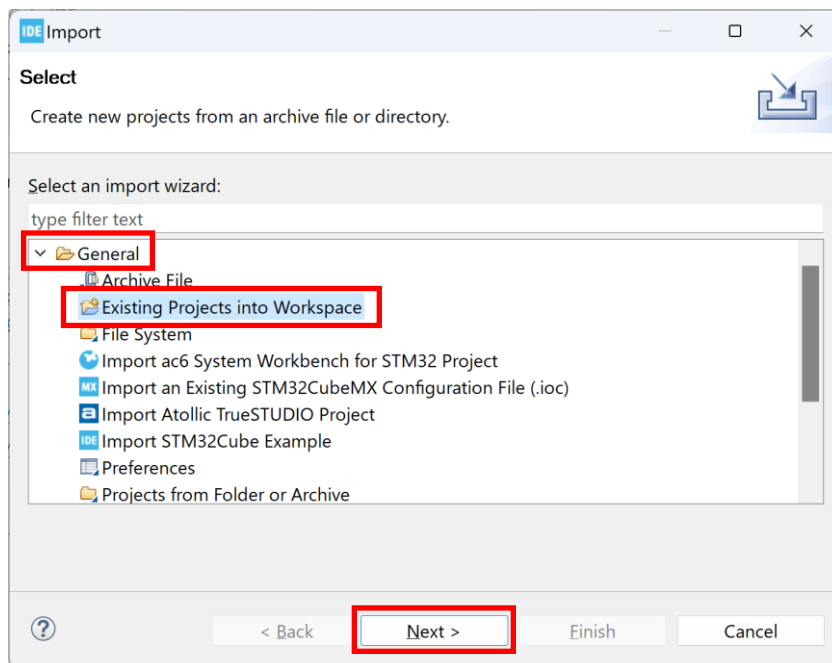


図 14 既存のプロジェクトのインポートを選択

- (5) [Search for nested projects] のチェックを外し、[Browse...] で nucleo_l476_lwip フォルダを選択し、[Finish] をクリックする。

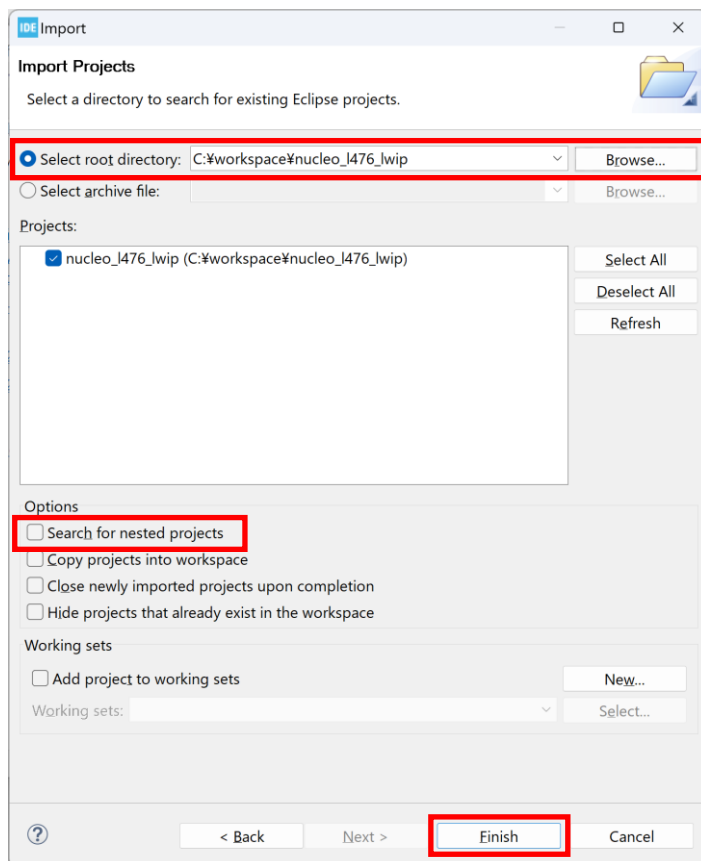


図 15 インポートするプロジェクトの選択

(6) [Project Explorer]に `nucleo_l476_lwip` が追加される。

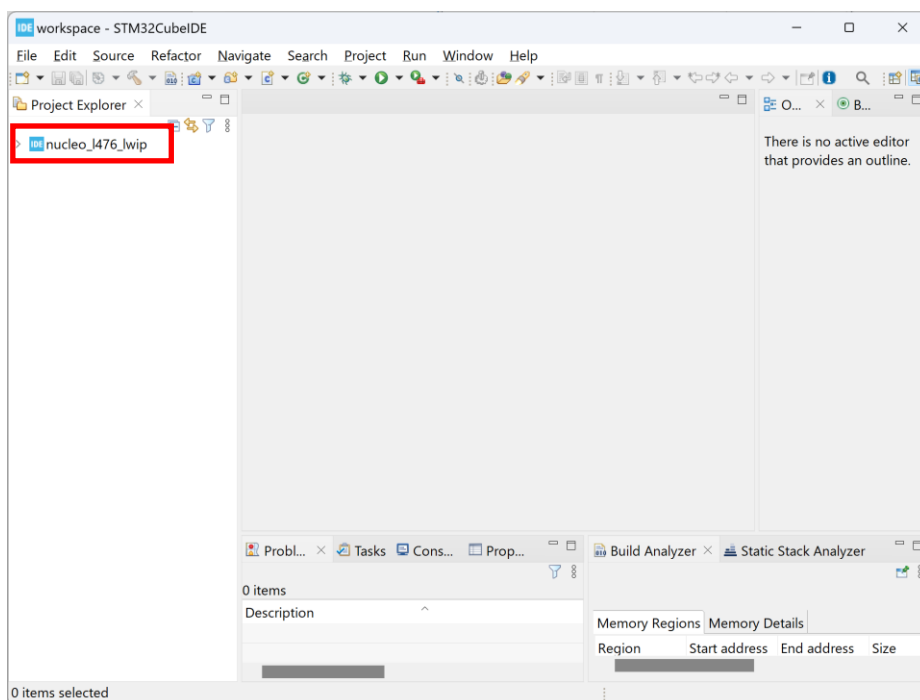


図 16 プロジェクト追加完了

4.5. NUCLEO-L476RG と EthernetShield2 の接続

EthernetShield2 の SPI 通信インタフェースは ICSP 端子を利用している(下図)。一方の NUCLEO-L476RG には ICSP の端子がなく、Arduino インタフェースの D10～13 に SPI 通信インタフェースが接続されている。

表 4-1 SPI 通信用の端子

ICSP PIN 番号	信号名	NUCLEO-L476RG Arduino 互換 PIN
3	SCK	D13
1	MISO	D12
4	MOSI	D11
-(NC)	CS	D10

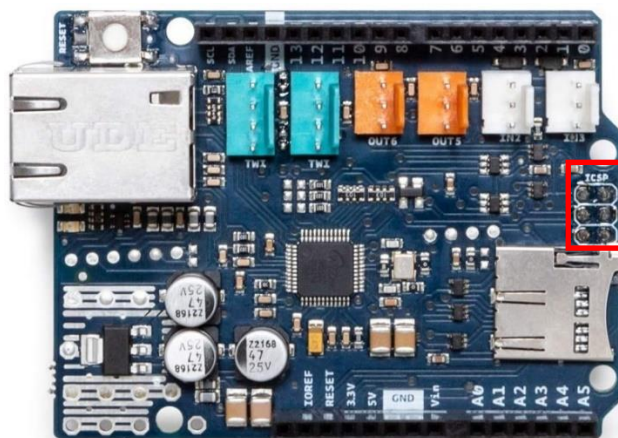


図 17 EthernetShield2 の SPI インタフェース

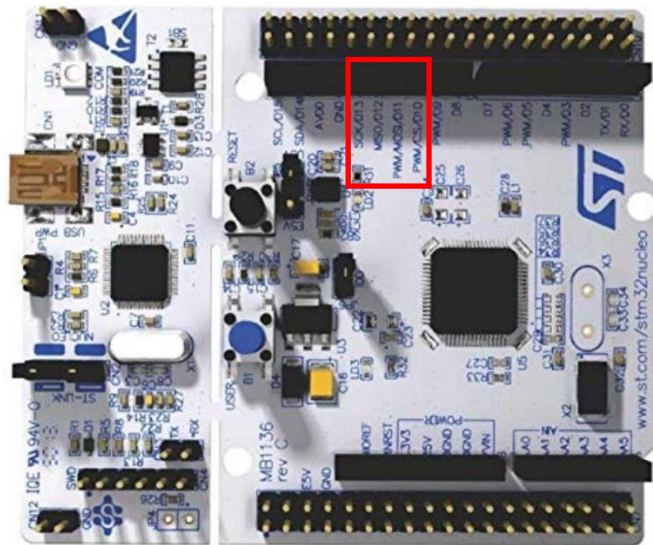


図 18 NUCLEO-L476RG の SPI インタフェース

このため、ボード間で SPI 通信を行うには、NUCLEO-L476RG の SPI 端子と EthernetShield2 の ICSP 端子とを以下のように接続する必要がある。(下記は同一ボードの裏面と表面である。)

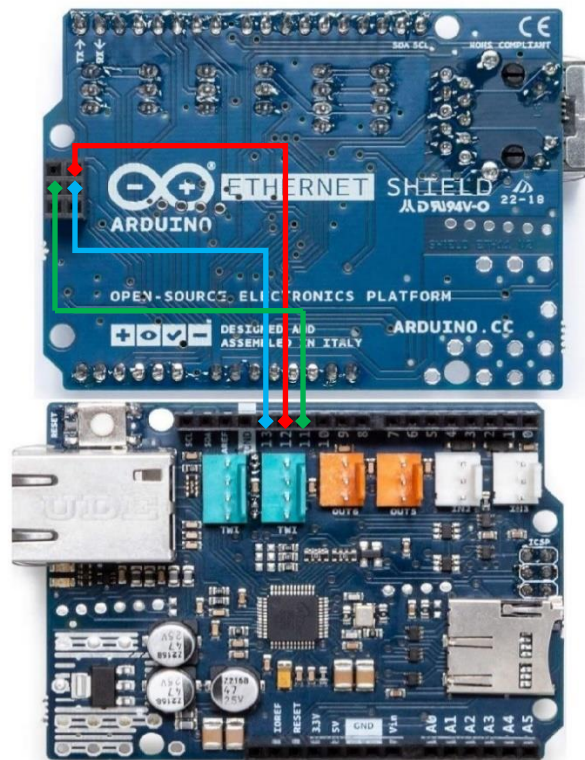


図 19 EthernetShield2 の ICSP と SPI の接続

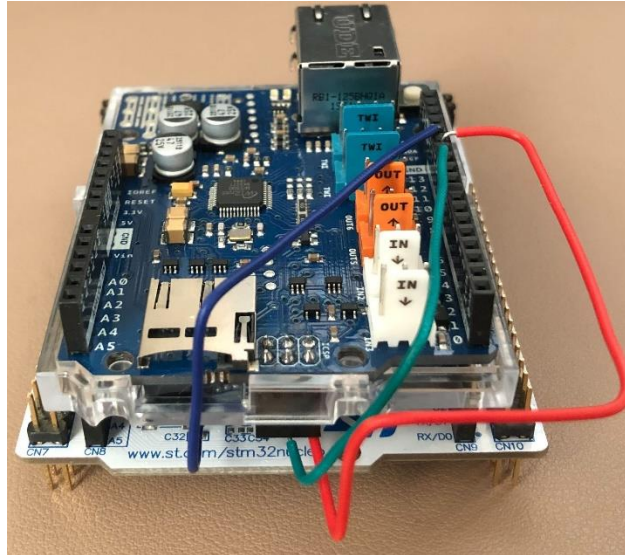


図 20 EthernetShield2 を NUCLEO-L476RG に装着

4.6. ビルド、実行

前節までの操作で必要な設定が完了したので、μT-Kernel 3.0 版の lwIP サンプルプロジェクトをビルドする。

(1) Project Explorer で `nucleo_l476_lwip` を選択する。

この操作によりツールバーでのビルドが可能となる。

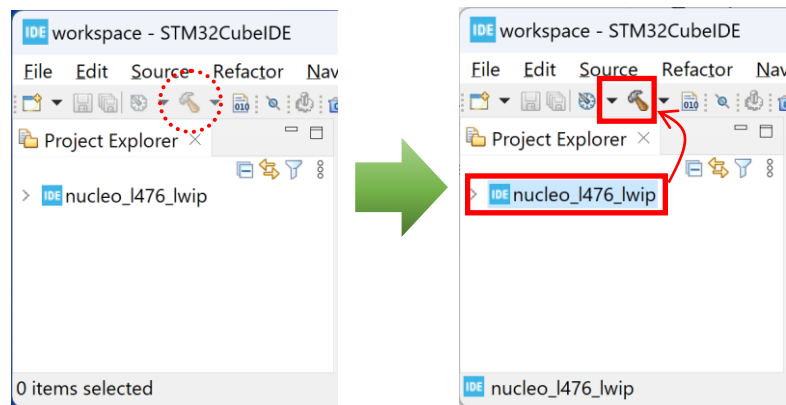



図 21 プロジェクトを選択するとビルドが可能となる。

(2) ツールバーの  [Build] をクリックするとビルドが開始される。

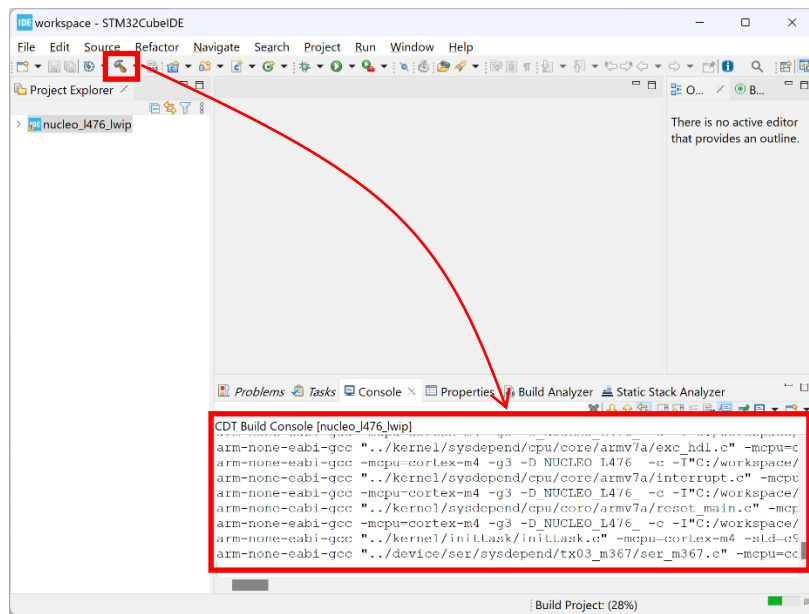


図 22 ビルド開始

(3) ビルドが完了すると Console に **Build Finished. 0 errors,** と表示される。

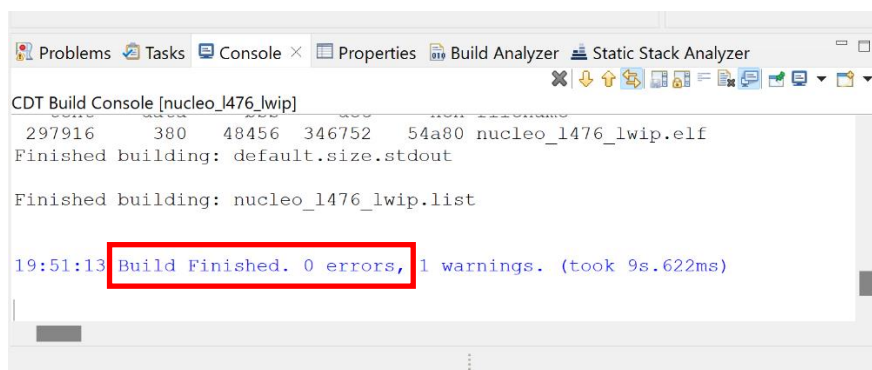


図 23 ビルド完了

❗ ここではワーニングは無視して構わない。

(4) EthernetShield2 を NUCLEO-L476RG に装着し、LAN ケーブルを接続する。

- (5) ターゲットボードの CN1 と PC を USB ケーブルで接続する。

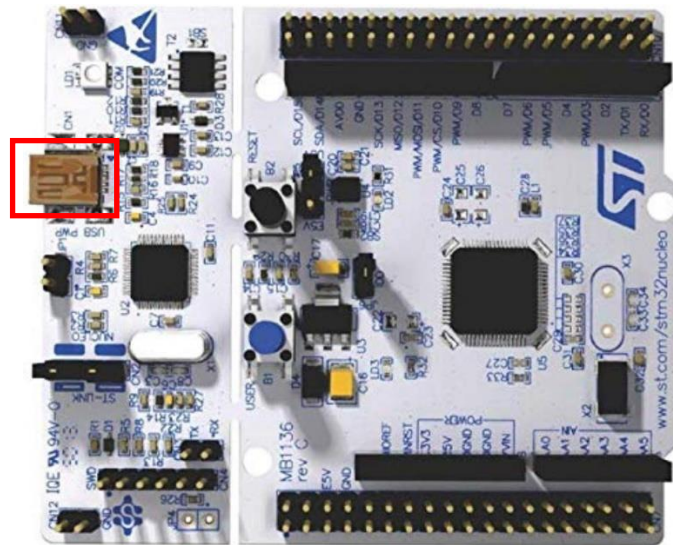


図 24 デバッグ/コンソール(USART2)兼用の接続先(CN1)

- (6) 通信ソフトを起動する。

本書では Tera Term を用いているが、特に限定はされていない。
通信パラメータは以下のとおりである。

通信速度	115,200 bps
データ長	8 bits
パリティビット	なし
ストップビット長	1 bit

- (7) デバッグのドロップダウンリストを選択して [Debug Configurations...] をクリックする。

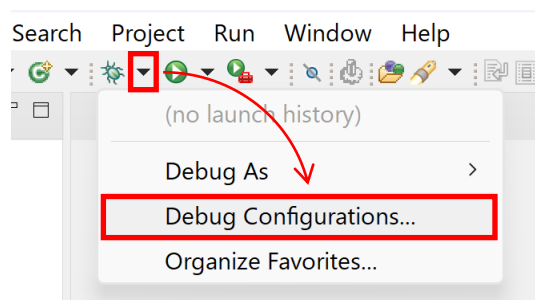



図 25 デバッグコンフィグレーションを開く

- (8) Debug Configurations ダイアログで[STM32 C/C++ Application]を選択し、右上の  [New launch configuration] をクリックして新しい実行コンフィグレーションを作成する。

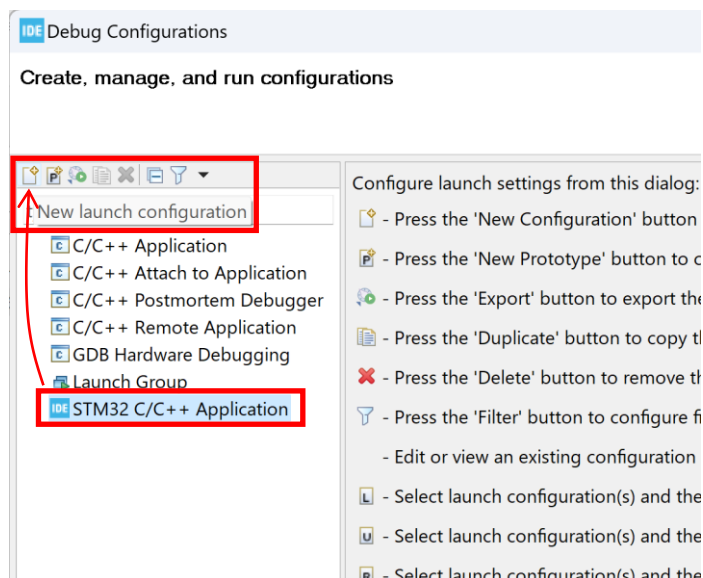


図 26 新しいデバッグコンフィグレーションの生成

- (9) 新しいデバッグコンフィグレーションの Name を `nucleo_l476_lwip_debug` に設定し、[Browse...] でプロジェクト `nucleo_l476_lwip` を選択して [Debug] をクリックする。

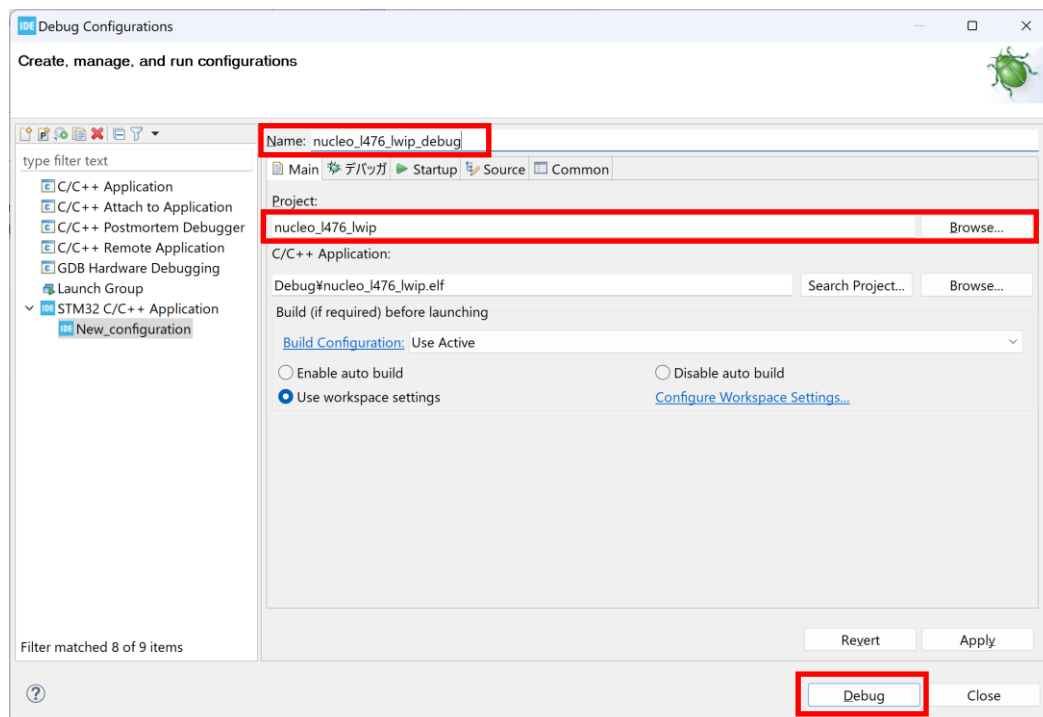


図 27 デバッグコンフィグレーションの設定

- (10) 書き込みが完了すると自動的にデバッグモードに切り替わり、プログラムが実行されて `main` 関数の先頭で停止する。

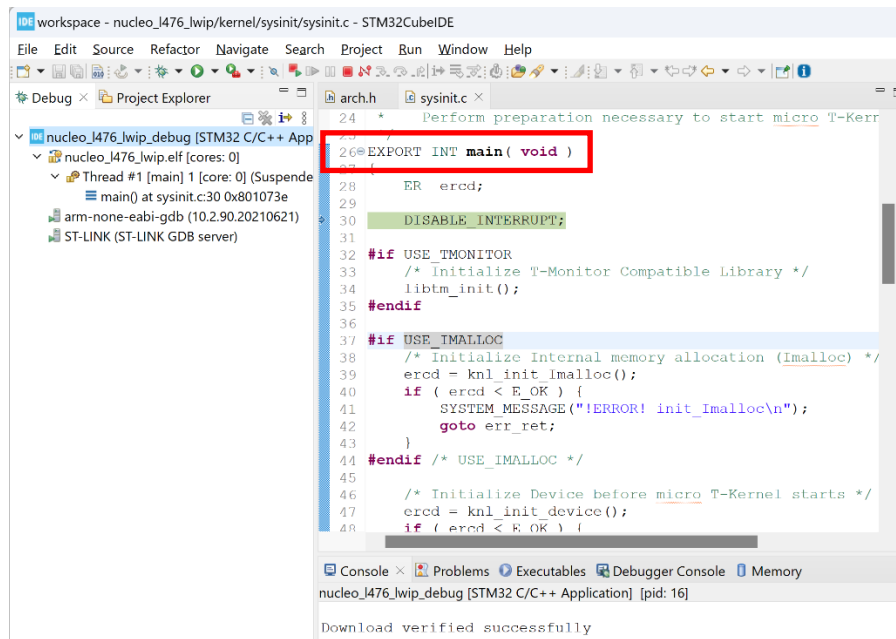


図 28 main 関数の先頭で停止した状態

- (11) [Resume (F8)]で実行すると、通信ソフトに DHCP の実行結果や HTTPD 起動のメッセージが表示される。

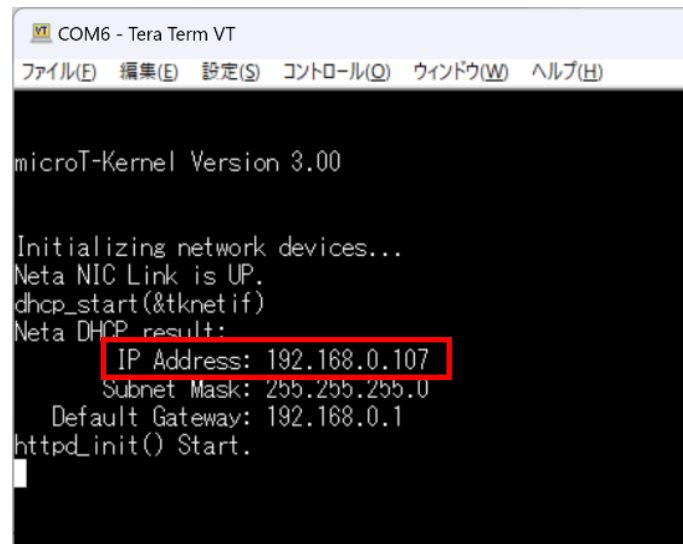


図 29 μ T-Kernel 3.0 からコンソールに表示されたメッセージ

この状態で、PC 側のブラウザで、コンソールに表示された IP アドレス(図 29 では 192.168.0.107)を入力すると、lwIP に組み込まれているサンプルのページがブラウザに表示される。

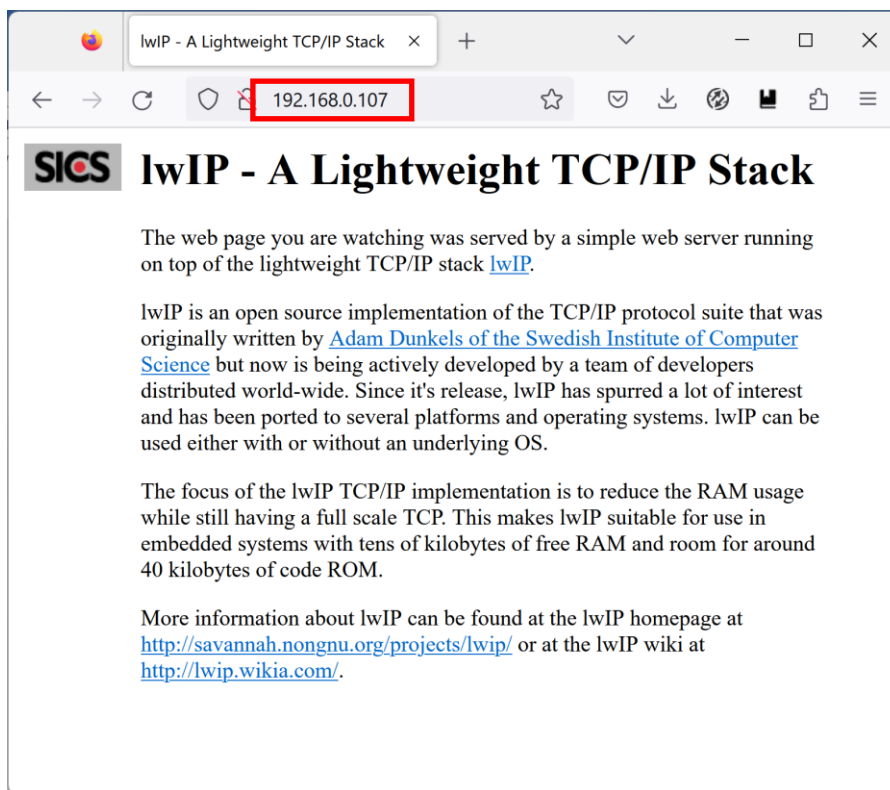



図 30 ブラウザでのサンプルページの表示

(12) 実行したプログラムは、[Terminate]で終了する。

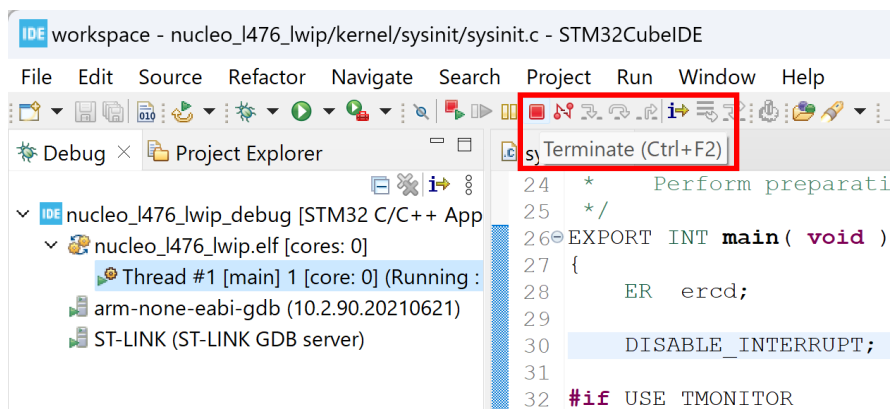


図 31 プログラムを[Terminate]で終了

終了したプログラムは[Remove All Terminated Launches]で削除する。

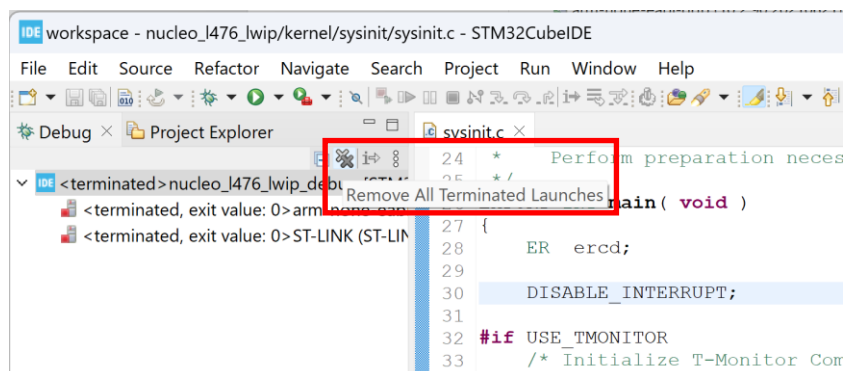



図 32 終了したプログラムは[Remove～]で削除

続いてデバッグのコンフィギュレーションを調整する。

デフォルトでは Non-OS の設定になっているので、main 関数の先頭で停止する(図 28)。これを μ T-Kernel 3.0 に合わせて初期タスクから呼び出される usermain 関数の先頭で停止するように変更する。

- (13) メニューの  の横の [▼] でプルダウンメニューを表示し、その中から [Debug Configurations] を選択して Debug Configurations を開く。

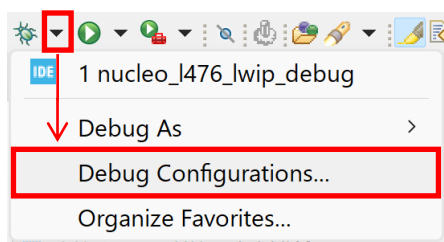




図 33 [Debug Configurations]を選択

- ①  をクリックすると前回使用したデバッグ構成でデバッグが開始される。
上図のメニューを開くためには、 の横の [▼] をクリックする必要がある。

- (14) 左 ペ イ ン の 中 か ら [STM32 C/C++ Application] の 中 の [nucleo_l476_lwip_debug] を選択する。

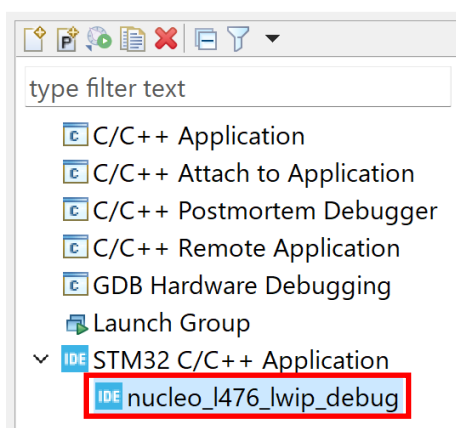


図 34 [nucleo_l476_lwip_debug]を選択

(15) [Startup]タブを選択し、その中の Set breakpoint at:に usermain と入力する。

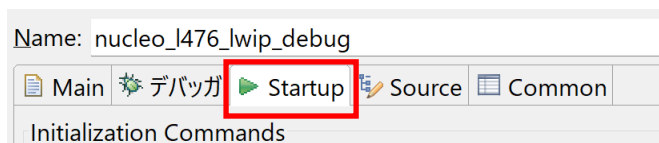


図 35 [Startup]タブを選択

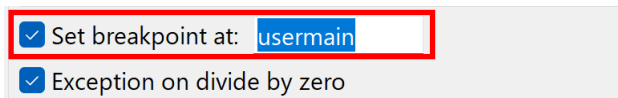


図 36 プログラム実行後のブレークポイントとして usermain 関数を指定

(16) 設定が完了したら右下の [Apply]→[Close] で終了する。

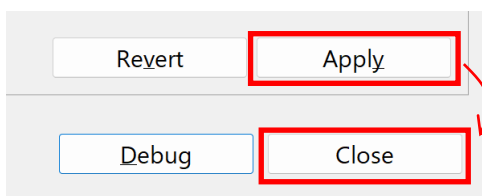


図 37 [Apply]→[Close]で設定を保存して終了

(17) [nucleo_l476_lwip_debug]を選択してデバッグ実行すると usermain 関数の先頭で停止する。

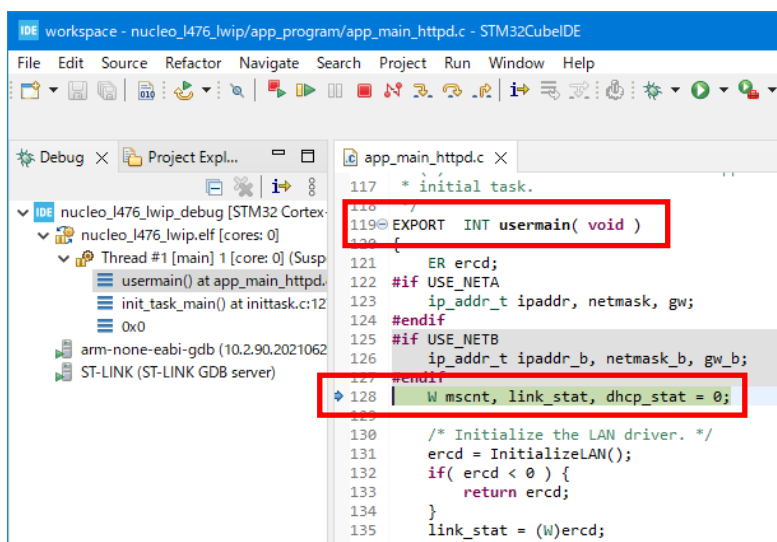


図 38 usermain 関数の先頭で停止

(18) [Resume (F8)] で実行すると、通信ソフトにメッセージが表示される。

(図 29 参照)

以上で μT-Kernel 3.0 の初期タスクから呼び出される usermain 関数の実行まで実行できたことになる。

5. μT-Kernel 3.0 のディレクトリ／ファイル構成

NUCLEO-L476 版の μT-Kernel 3.0 のディレクトリおよびファイルの構成は、μT-Kernel 3.0 の正式リリース版に準じて以下のように構成してある。

表 5-1 プロジェクトのファイル構成

ディレクトリ名 またはファイル名	内容
app_program/	μT-Kernel 3.0 用アプリケーションのディレクトリ
config/	コンフィギュレーション
device/	デバイスドライバ
docs/	ドキュメント
etc/	リンカファイル等
include/	各種定義ファイル
kernel/	μT-Kernel 3.0 本体
lib/	ライブラリ
build_make/	Make 構築ディレクトリ(本実装では使用しない)
.cproject	Eclipse のプロジェクトファイル
.project	CDT 用プロジェクトファイル
.settings/	Eclipse の各種プラグイン用設定ファイル
README.md	本実装の ReadMe ファイル
ucode.png	μT-Kernel 3.0 の ucode
.git	Git 用のファイル
.gitmodule	Git 用のファイル

5.1. μT-Kernel 3.0 のソースコード

config/、device/、docs/、etc/、include/、kernel/、lib/ の各ディレクトリには μT-Kernel 3.0 のソースコードが含まれる。

サンプルプログラム程度であれば特に変更する必要はないが、タスクやセマフォの最大数などの調整が必要な場合は config/以下のファイルで調整することになる。

- ① μT-Kernel 3.0 のコンフィギュレーションについては「μT-Kernel 3.0 共通実装仕様書」を参照のこと

build_make/は gcc を用いて開発する場合に利用する。本実装では STM32CubeIDE を利用するので、これらのディレクトリは利用しない。

μT-Kernel 3.0

構築手順書 (NUCLEO-L476 + EthernetShield2)

Rev 3.00.01 (May, 2023)

ユーシーテクノロジー株式会社

141-0031 東京都品川区西五反田 2-12-3 第一誠実ビル 9F

©2022-2023 Ubiquitous Computing Technology Corporation All Rights Reserved.