EmbiTeK

テクニカルアップデート　2020 / 03 / 25

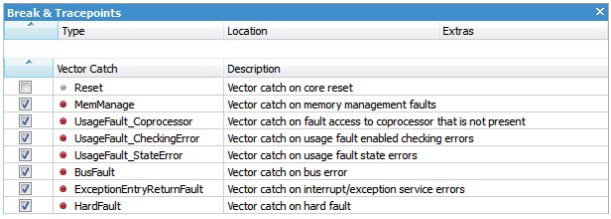
対応 CPU コア　Cortex-M

J-Link Debugger「OZONE」新機能

システムクラッシュの原因特定をサポートする「例外検出・例外分析」

組込ソフトウェア開発中に発生するファームウェアのクラッシュ。原因としては、不正なメモリアクセス、電源供給されていない／不足している周辺機能へのアクセス、無駄なメモリ実行、ゼロ除算など、すべてのケースで、「HardFault ハンドラ」で終了します。
　この場合、開発者はデバッガに残された情報を元に、原因推定し、問題点を特定する必要があります。
「OZONE（オゾン）」は障害発生を自動検出し、クラッシュが発生した原因、及びその事象以前に発生したことを分析できる追加の情報を提供可能に。

ユーザメリット
- ■ 例外の自動検出
- ■ システムクラッシュの原因を探る情報の提供

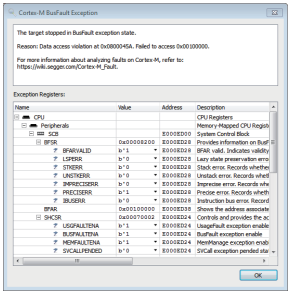


例外・システムクラッシュの検出

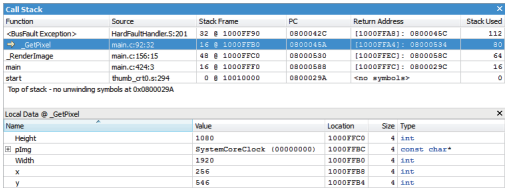
「OZONE（オゾン）」は Arm デバイスのベクターキャッチを使用します。
通常のブレイクポイントに加えて、Arm コア搭載デバイスは、例外ハンドラが実行される直前に、例外ベクトルがフェッチされたタイミングで、実行を中断できます。この機能により、ハードウェアブレイクポイントを利用せずに**「例外の自動ブレイク」が可能となります。**
「HardFault」でのベクターキャッチにより、システムのクラッシュを即座に検出し、分析できます。
「OZONE（オゾン）」では、ブレイクポイントウィンドウで、ベクターキャッチを有効または無効に設定できます。

例外・システムクラッシュの理由についての情報表示

　「Cortex-M BusFault Exception」機能では、Arm Cortex-M マイコンから提供された様々なレジスタ情報を提供します。
　システムクラッシュが発生した際にすべての情報を手動で取得するのは、面倒ですが、**「OZONE（オゾン）」は関連する情報を自動で分析し、クラッシュの原因とクラッシュが発生した場所、アクセスできなかったアドレスなど詳細情報を表示できます。**



例外・システムクラッシュの分析

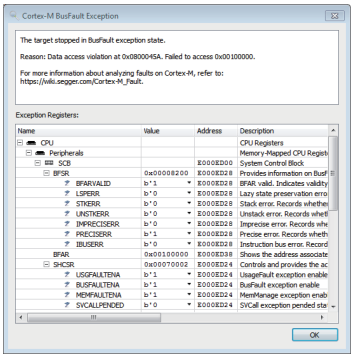


例外・クラッシュの理由が情報表示され、次のステップは、発生箇所の特定となります。
障害のステータスレジスタに加えて「OZONE（オゾン）」は例外状態であっても、すべての通常デバッグ情報を提供します。**[呼び出しスタック]** ウィンドウには、例外が発生した場所が表示されます。**[Registers]** ウィンドウと **[Local]** ウィンドウを利用して、障害箇所の分析ができます。
「バス違反例外」は、不正なアクセスより数命令後に発生する可能性があります。この場合、「Ozone（オゾン）」の命令トレースは、以前に実行された命令を識別することに有益です。

技術情報：Cortex-M の障害・例外について

Cortex-M の例外

Cortex-M コア搭載マイコンでの障害は、不正なメモリアクセス、電源供給されていない／不足している周辺機能へのアクセス、無駄なメモリ実行、ゼロ除算など、すべてのケースで、「HardFault ハンドラ」で終了します。



■ HardFault 例外

「HardFault」例外はデフォルトの例外であり、別の例外に関連付けられていないエラーが発生した場合に発生します。
HardFault の優先度は -1 に固定されています。NMI を除く他のすべての割込み、例外よりも高い優先度を持っています。
　したがって、アプリケーションコード、割り込み、または別の例外でエラーが発生した場合、HardFault 例外ハンドラーを常に入力できます。
HardFault　IRQ：-13　／ベクターテーブル：例外番号 3

■ MemManage 例外

MemManage 例外は、メモリ保護違反（MPU）により利用でき、メモリアクセス違反の例外を発生させます。
MemManage　IRQ：-12　／ベクターテーブル：例外番号 4
構成可能な優先度を持っています。

■ BusFault 例外

「BusFault 例外」は、不正なアクセス、ベクターキャッチによるメモリアクセスエラーにより、発生します。
BusFault　IRQ：-11　／ベクターテーブル：例外番号 5
構成可能な優先度を持っています。BusFaults は、システム制御ブロック（SCB）で明示的に有効にできます。BusFault が有効になっていない場合、HardFault が発生します。

■ UsageFault 例外

「UsageFault 例外」は、実行エラーの場合、発生します。ロード / ストアの複数の命令での非境界整列アクセスは常にキャッチされます。その他の非境界整列アクセスの例外、およびゼロ除算は、SCB でさらに有効にできます。
UsageFault　IRQ：-10　／ベクターテーブル：例外番号 6
構成可能な優先度を持っています。UsageFault が有効になっていない場合、代わりに HardFault が発生します。

Cortex-M の例外処理

例外が発生した場合、割込と同様にベクターテーブルから読み取れる例外ハンドラが呼び出されます。例外処理は、通常「開発中」と「本稼働中」のファームウェアで異なる方法で実行されます。

■ 開発中ファームウェアの例外処理

ファームウェアの開発中にエラーが発生した場合、開発者は「エラーを解決するために何が問題なのかを分析したい」と考えています。
「Cortex-M NVIC」は、クラッシュの理由を分析するための様々なレジスタ情報を提供し、ほとんどのケースで、コールスタックとクラッシュが発生した場所のレジスタ内容を復元できます。
この作業を行う上で、開発者は特別な例外ハンドラーをコードに追加する必要はありません。「OZONE（オゾン）」デバッガは、単純にベクターキャッチまたはブレークポイントでブレークし、分析を行います。

■ 本稼働ファームウェアでの例外処理

本稼働中のファームウェアでも例外が発生する可能性があるため、こういった例外を把握する必要があります。
一部の例外はエラーが原因ではない場合があります。例として、デバッガが接続されていないときに BKPT 命令を実行すると、HardFault が発生します。こういった場合、ファームウェアはプログラムの実行を継続できます。
システムエラーではあるが、クラッシュではない場合、特定のケースから回復することもできます。
例外ハンドラーで、エラーの理由を分析し、エラーを引き起こしたタスクを終了するなどして、回復できます。
クラッシュの場合、システムは単にリセットするだけで回復できます。より高度なシナリオでは、リセット前にクラッシュダンプを生成し、再起動時に送信または保存できます。

Cortex-M の例外分析

「OZONE（オゾン）」では、例外の原因分析を簡素化する特別な機能を提供します。
「OZONE（オゾン）」は、ターゲットシステムがクラッシュしたことを検出すると、ターゲットの状態を自動的に分析し、必要なすべての情報を提供します。**[例外ウィンドウ]** には、クラッシュの原因と発生場所および追加の NVIC レジスタが表示されます。また、**[コールスタックウィンドウ]** は、複数の例外にまたがってエラーの場所に簡単に移動できるように、巻き戻すことができます。

障害ステータスレジスタ

Cortex-M システム制御ブロック（SCB）には、例外の構成を可能にし、障害に関する情報を提供するいくつかのレジスタが含まれています。

■ **HardFault ステータスレジスタ（HFSR）**
HFSR は、アドレス 0xE000ED2C の SCB にあります。32bit レジスタ

[31] DEBUGEVT- デバッガが使用するために予約されています。常に 0 を書き込みます。
[30] FORCED-1 の場合、HardFault は別の例外のエスカレーションによって引き起こされています。
[1] VECTTBL-1 の場合、例外処理にベクターテーブルを読取ることにより BusFault が発生。

■ **UsageFault Status Register（UFSR）**
UFSR は、アドレス 0xE000ED28 の構成可能レジスタ（CFSR）の一部である 16 ビットの擬似レジスタです。
また、0xE000ED2A へのハーフワードアクセスで直接アクセスすることもできます。

[9] DIVBYZERO-1 の場合、除数 0 で SDIV または UDIV 命令が実行されます。
[8] UNALIGNED-1 の場合、LDM、STM、LDRD、アラインされていないアドレスでの STRD の実行、
　またはトラップが有効な場合の単一のロードまたはストアの実行。
[3] NOCP-1 の場合、サポートされていない（使用できない、有効になっていない）コプロセッサへのアクセス。
[2] INVPC-1 の場合、違法または無効な EXC_RETURN 値が PC にロードされます。
[1] INVSTATE-1 の場合、無効な状態で実行されます。
　例えば、Thumb ビットが EPSR で設定されていない、または EPSR で無効な状態です。
[0] UNDEFINSTR-1 の場合、未定義の命令の実行。

■ **BusFault ステータスレジスタ（BFSR）および BusFault アドレスレジスタ（BFAR）**
BFSR は、CFSR の 8 ビットの擬似レジスタです。0xE000ED29 で直接アクセスできます。
BFAR は、0xE000ED38 にある 32 ビットレジスタです。

[7] BFARVALID-1 の場合、BFAR には BusFault の原因となったアドレスが含まれます。
[5] LSPERR-1f 1、浮動小数点の遅延スタック保存中の障害。
[4] STKERR-1 の場合、例外エントリのスタッキングの障害。
[3] UNSTKERR-1 の場合、例外復帰時のアンスタックの失敗。
[2] IMPRECISERR-1 の場合、戻りアドレスは障害、たとえば前に発生した障害に関連していません。
[1] PRECISERR-1 の場合、リターンアドレス命令が障害を引き起こしました。
[0] IBUSERR-1 の場合、命令フェッチの失敗。

■ **MemManage 障害ステータスレジスタ（MMFSR）および MemManage 障害アドレスレジスタ（MMFAR）**
MMFSR は、CFSR の 8 ビットの擬似レジスタです。0xE000ED28 で直接アクセスできます。
MMFAR は、0xE000ED34 にある 32 ビットレジスタです。

[7] MMARVALID-1 の場合、MMFAR には MemManageFault の原因となったアドレスが含まれます。
[5] MLSPERR-1f 1、浮動小数点の遅延スタック保存中の障害。
[4] MSTKERR-1 の場合、例外エントリのスタックの失敗。
[3] MUNSTKERR-1 の場合、例外復帰時のアンスタックの失敗。
[1] DACCVIOL-1 の場合、データアクセス違反。
[0] IACCVIOL-1 の場合、命令アクセス違反。

■ **スタック回復**
例外入力時に、例外ハンドラーは、障害が発生したときにどのスタックが使用されたかを確認できます。
ビット EXC_RETURN [2] が設定されている場合、MSP が使用。それ以外の場合、PSP が使用。
スタックを使用して、CPU レジスタ値を回復できます。

■ **CPU レジスタリカバリ**
例外エントリでは、いくつかの CPU レジスタがスタックに格納され、そこからエラー分析のために読み取ることができます。

J-Link Software

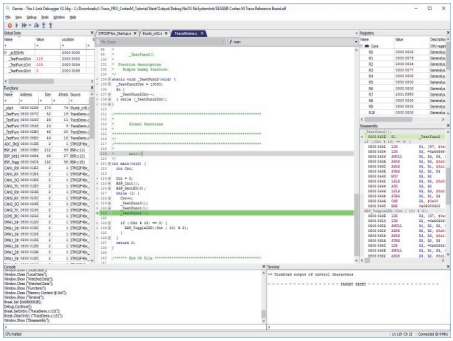
J-Link を最大限に活用するバンドルソフトウェア

ソフトウェアデバッグ・開発支援

J-Link Debugger「OZONE」

対応 CPU コア　ARM / RISC-V

J-Link バンドルデバッグソフトウェア (J-Link PLUS 以上)

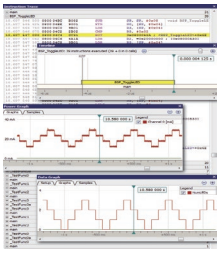


フリーの GNU コンパイラや IAR EWARM、Keil MDK など商用コンパイラから生成されたデバッグ情報含む ELF ファイルを利用してデバッグができる J-Link バンドルソフトウェアとなります。

OZONE はデバッガとしてのフル機能（命令トレース、パワーグラフ、ライブウォッチ、リアルタイムターミナル I / O など）を備えています。

これまで高価なデバッガが必要であった関数プロファイリングやコードカバレッジなども J-Trace や J-Link との組み合わせで実現することができます。デバッグ人員の増減にも対応しやすく、高いコストパフォーマンスを実現します。

パフォーマンスアナライザ



J-Link と J-Trace PRO の機能により、ターゲットシステムのパフォーマンス分析に使用できます。J-Trace PRO と組み合わせることで、アプリケーションの実行時間を記録し分析することができます。タイムラインで視覚化してファンクションや割込みの経路を簡単に表示することができます。J-Link の高速サンプリング技術 (RTT) により、「OZONE」はシステム変数を時系列に可視化することができます。

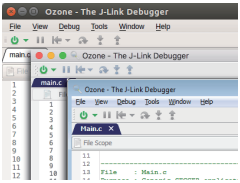
電力プロファイリング

ターゲットデバイスの電力消費量をデバッグセッションで記録できます。これは特定のコンポーネントを有効／無効にするときの消費電力の変化を知る必要がある場合に特に役立ちます。

テストの自動化

デバッグのワークフローを自動化するスクリプトにより、テストを自動化して、進めることができます。デバッガのほとんどの機能をスクリプトにより制御可能で、開発生産性を向上。

マルチプラットフォーム対応

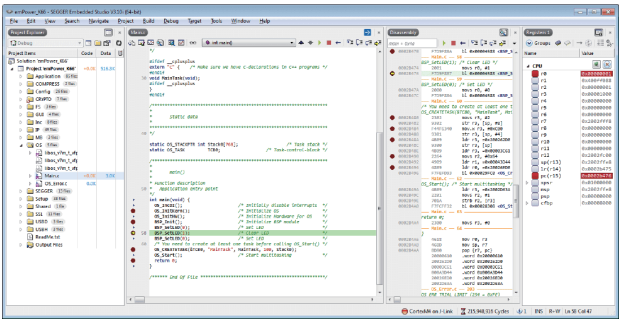


Arm / RISC-V コアに対応した「OZONE」は Windows、Mac、Linux のクライアント OS で利用することができます。利用するコンパイラもプロジェクトに合わせて設定可能なため、より生産的な開発ツールとして、利用可能です。

J-Link Unlimited Flash-BP

対応 CPU コア　ARM / Cortex

マイコンのブレイクポイント数を無制限に設定可能な拡張機能（J-Link PLUS 以上）



「J-Link Unlimited Flash-BP」拡張機能を使うことで、ユーザーはフラッシュメモリでデバッグするとき、無制限のブレークポイントを設定できます。この機能がないデバッガでは、フラッシュに設定できるブレークポイント数は、CPU のデバッグユニットでサポートされているハードウェアブレイクポイントの数に制限されます。（ARM 7/9 では 2、Cortex-M では 4-6）

利用可能な開発ツール

SEGGER 社製：J-Link Debugger「Ozone」 /　SEGGER Embedded Studio
他社製：J-Link DLL、J-Link RDI 接続をサポートする統合開発環境（IAR EWARM / Arm MDK-ARM など）

Monitor Mode Debug

対応 CPU コア　Cortex-M

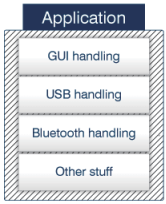
モータ制御 / 無線通信アプリケーションに最適なモニターモードデバッグ機能

J-Link モニタモードデバッグを使用すると、実行優先順位の高いコードを通常通り実行しながら、優先順位の低いアプリケーションを停止・実行することができます。これにより、開発者に新しい手法のアプリケーションデバッグ手法を提供します。通信タスクを維持したまま、割り込み、イベント、その他の信号を任意でオン / オフできるため、従来の手法ではデバッグが困難だった多様なシステム状況を開発者は把握し、デバッグする事ができます。

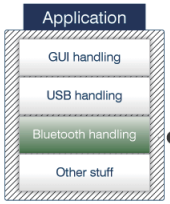
モニターモードでバッグが有効なアプリケーション

■ PWM を利用したモータ制御アプリケーション
モニターモードデバッグを利用することにより、PWM 制御を続けながら、アプリケーションデバッグする事が可能です。

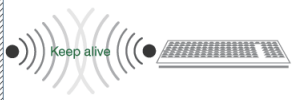
■ Bluetooth 通信アプリケーション
モニターモードデバッグを利用することにより、Bluetooth を「keep alive」実行続けながら、デバッグ処理を行う事ができます。



■ 通常のデバッグ
通信も停止してしまうため、通信中の挙動が見えない。



■ モニターモードデバッグ
通信部分を続けながら、デバッグ対応



利用可能な開発ツール

SEGGER 社製：J-Link Debugger「Ozone」 /　SEGGER Embedded Studio
他社製：IAR EWARM / Arm MDK-ARM

J-Link Software

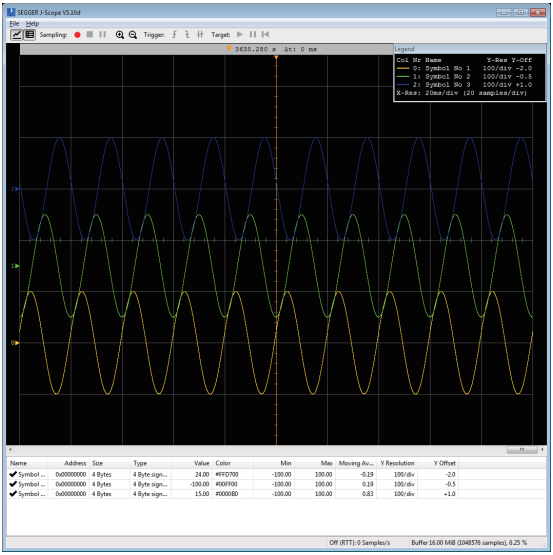
J-Link を最大限に活用するバンドルソフトウェア

ソフトウェアデバッグ・開発支援

J-Scope

RAM モニタリング機能

対応 CPU コア　Cortex-M



ハードウェアリソース不要

「J-Scope」は JTAG 経由で必要な情報を取得します。そのため特別なハードウェアは不要、新たに情報取得のためのピンを立てる必要はありません。

高速なデータ取得

J-Link の RTT インターフェースを利用することで、最小転送周期：28.5 µ Sec（2-byte 変数 x8 (=16byte) の場合 / Cortex-M4 ）を実現します。

各種開発ツールと併用可能

お客様が利用している各種開発ツールと併用が可能です。

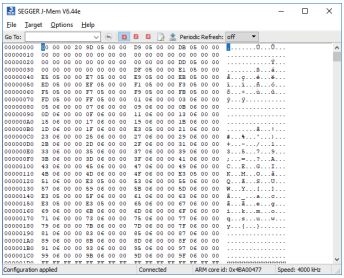
「J-Scope」はターゲットが動作している間、リアルタイムで RAM 上のデータを分析し可視化するためのソフトウェアです。複数の変数の値をオシロスコープのようなスタイルで表示できます。ELF ファイルを読み込み、視覚化する変数の数を選択することができます。

J-MEM

RAM データ表示、変更書込ツール

対応 CPU コア　すべての対応コア

- RAM 変更
- SFR 領域に書込
- RAM 領域全体に希望の値を入力
- メモリセクションを .bin ファイルに保存
- 表示されたメモリの内容の更新間隔を設定可能

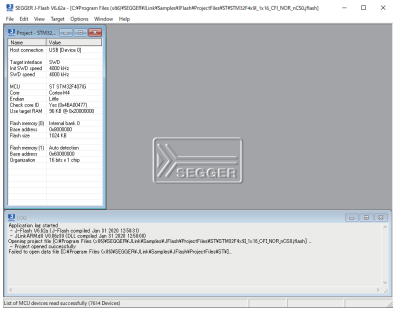


Flash 書込ツール

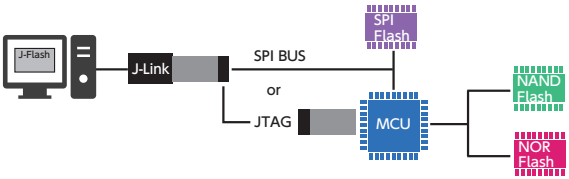
J-Flash / J-Flash SPI

対応 CPU コア　対応する CPU コア

J-Link を利用して、内蔵／外部フラッシュに書込（J-Link PLUS 以上）



フラッシュ書込みについて、専用ツールでサポートします。マイコン内蔵、マイコン経由だけでなく、SPI バスに接続して直接 SPI フラッシュメモリに書き込み可能な「J-Flash SPI」も併せて提供。5,200 種以上のマイコン内蔵フラッシュ、300 種以上の SPI フラッシュに対応します。未対応のデバイス、専用 SoC などへも「Open Flash Loader」の仕組みを利用することで対応可能。当社で定義ファイルのカスタム対応も承ることができます。



外部 NOR/SPI フラッシュ対応

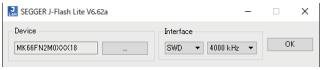
JTAG からマイコン経由での書込みの他、対応する SPI フラッシュでは、SPI バスから直接書込む事が可能です。

マルチバンク対応

同じハードウェアに存在する異なるフラッシュを 1 回のセッションで書込むことができます。

J-Link BASE でも利用可能な「J-Flash LITE」

J-Flash LITE は、「J-Flash / J-Flash SPI」と書込アルゴリズムなどが異なりますので、サポート対象外・量産利用不可のソフトウェアとなります。

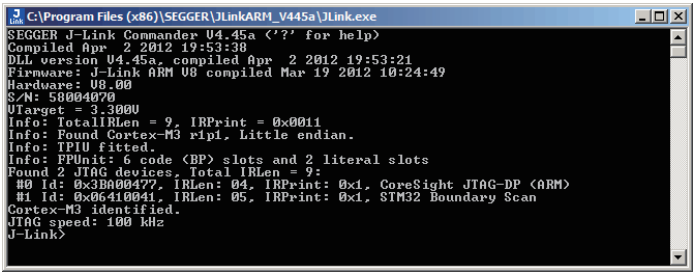


汎用ツール

J-Link Commander

対応 CPU コア　対応する CPU コア

ハードウェアデバッグ・不具合解析に利用できるコマンドラインツール



ハードウェア開発において、役立つツール「J-Link Commander」基本的なマイコンの接続確認やペリフェラルに変数を与えて、期待値通りの信号が出るかなど、ハードウェアデバッグに最適なツールです。ソフトウェア開発においてもマイコンへの接続、フラッシュへの書込など、様々な局面で開発をサポートします。J-Link の接続、動作がおかしいなどトラブルシューティングとしてもご利用頂けます。

J-Link Software 導入

J-Link を最大限に活用するバンドルソフトウェア

ダウンロード

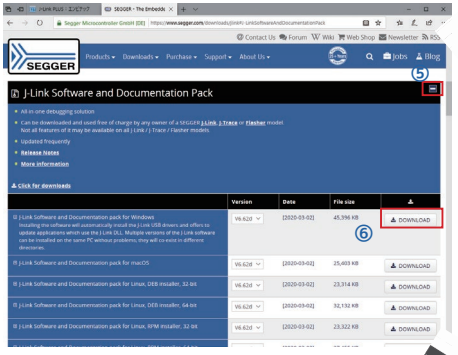
EmbiTeK www.embitek.co.jp へアクセスしてください。

J-Link 製品ページに移動

① トップメニュー [製品] を選択
② [JTAG/SWD デバッガツール] を選択
③ 任意の J-Link 製品を選択してください。

SEGGER 社ダウンロードページに移動

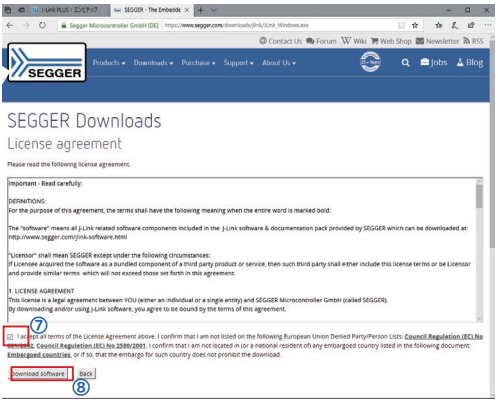
④ [J-Link 用ソフトウェア・ドライバ] をクリック



SEGGER 社 J-Link ソフトウェアダウンロードページ

⑤ SEGGER 社ダウンロードページで、「J-Link Software and Dcumentation Pack のアコーディオンアインコン [+] をクリック

⑥ ご利用 OS(Windows / Mac / Linux) 用に合わせて [DOWNLOAD] ボタンをクリック



J-Link ソフトウェアダウンロード

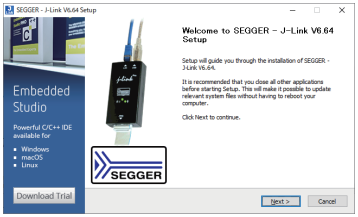
⑦ 利用規約に同意する旨、チェックを入れてください。

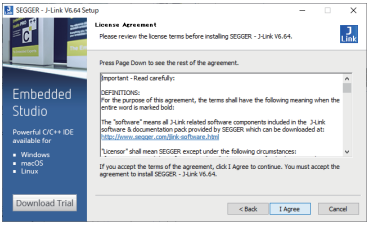
⑧ [Download Software] ボタンをクリック

インストール

J-Link ソフトウェアのインストール

ダウンロードが完了したら、インストーラの指示に従い、インストールしてください。





インストールフォルダ構成

インストールフォルダ

```
<SEGGER\JLink>
│ JFlash.exe ····· フラッシュ書込ツール
│ JFlashLite.exe ····· ノンサポート書込ツール（評価のみ）
│ JFlashSPI.exe ····· SPI 書込ツール
│ JFlashSPI_CL.exe ····· SPI 書込ツール（コマンドライン）
│ JLink.exe ····· J-Link コマンダーツール
│ JLinkConfig.exe ····· J-Link 設定ツール
│ JLinkDLLUpdater.exe ···· J-LinkDLL アップデータ
│ JLinkGDBServer.exe ····· J-Link GDB サーバ
│ JLinkGDBServerCL.exe ··· J-Link GDB サーバ（コマンドライン）
│ JLinkGUIServer.exe
│ JLinkLicenseManager.exe·· ライセンスマネージャ
│ JLinkRDIConfig.exe ····· RDI コンフィグツール
│ JLinkRegistration.exe
│ JLinkRemoteServer.exe ···· リモートデバッグサーバ
│ JLinkRemoteServerCL.exe·· リモートデバッグサーバ（C/L）
│ JLinkRTTClient.exe ····· J-LinkRTT クライアント
│ JLinkRTTLogger.exe ····· J-LinkRTT ロガー
│ JLinkRTTViewer.exe ····· J-LinkRTT ビューワ
│ JLinkSTM32.exe
│ JLinkSTR91x.exe
│ JLinkSWOViewer.exe ···· SWO ビューワ
│ JLinkSWOViewerCL.exe · SWO ビューワ（コマンドライン）
│ JMem.exe ····· JMEM メモリビューワ
│ JRun.exe
│ JTAGLoad.exe
│ SWOAnalyzer.exe
│
├── Devices ····· デバイス定義ファイル
│ ├── AnalogDevices
│ ├── ATMEL
│ ·····
│
├── Doc ····· マニュアル・リリースノート
│ ├── Manuals
│ └── ReleaseNotes
├── ETC
│ └── JFlash
├── GDBServer
├── RDDI
│
│
├── Samples ····· すぐに使用可能なサンプルプロジェクト
│ ├── DCC
│ │ ├── GNU
│ │ └── IAR
│ ├── GDB
│ │ ├── GDBInit
│ │ └── Projects
│ ├── JFlash
│ │ └── ProjectFiles ····· J-Flash サンプルプロジェクト
│ │ ├── AnalogDevices
│ │ ├── ARM
│ │ ├── Atmel
│ │ ·····
│ │
│ ├── JFlashSPI
│ │ └── ProjectFiles ····· J-Flash SPI サンプルプロジェクト
│ ├── JLink
│ │ ├── Projects
│ │ ├── Scripts
│ │ └── SettingsFiles
│ ├── RDI ····· J-Flash SPI サンプルプロジェクト
│ │ ├── Projects
│ │ │ ├── IAR
│ │ │ └── KEIL
│ │ └── SetupFiles
│ └── RTT
└── USBDriver
```

最新版の「J-Link ソフトウェア」「J-Link ファームウェア」でのサポート対応となります。

サポートご依頼時については、最新版 J-Link ソフトウェアのダウンロード・インストールをお願いいたします。