お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 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お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、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に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、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、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。お客様が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。
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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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その総株主の議決権の過半数を直接または間接に保有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。

フラッシュ開発ツールキット(Flash Development Toolkit)
ユーザプログラムモードの使用方法(SH7086 応用編)

要旨

このアプリケーションノートは、ルネサスフラッシュ開発ツールキット(Flash Development Toolkit)の使用方法と、フラッシュ開発ツールキットを使ったSH7086(SHファミリ)ユーザプログラムモードの使用方法を以下のとおり説明します。

・ブートモード1(ユーザブートエリアへの書き込み)
・ブートモード2(ユーザエリアへの書き込み)
・ユーザブートモード
・ユーザプログラムモード

これらの説明から、ブートモード、ユーザブートモード、ユーザプログラムモードの違いを理解し、ユーザプログラムモードの使い方を理解してください。
このアプリケーションノートでは、ユーザプログラムモード、ユーザブートモード共用で使う内蔵フラッシュメモリの書き込み消去の処理をしているユーザプログラムモード用ファイルの説明をしています。ユーザプログラムモード、ユーザブートモードでフラッシュメモリの書き込み消去をするときは、このユーザプログラムモード用ファイルを参照してください。

本書ではフラッシュ開発ツールキット 4.01 を使用しています。

目次

1. SH7086

1.1 フラッシュメモリ構成

SH7086(SHファミリ)のフラッシュメモリには、ユーザマット(ユーザエリア)、ユーザブートマット(ユーザブートエリア)の2種類のメモリマットがあります。
このほかに、フラッシュメモリ書き込み消去の制御プログラムを格納したエリアがあり、ブートマット(ブートエリア)と呼んでいます。このアプリケーションノートでは、それぞれ、ブートエリア、ユーザエリア、ユーザブートエリアと呼ぶことにします。

エリア	種類	サイズ	ブロック
ブートエリア	制御プログラム	—	—
ユーザブートエリア	フラッシュメモリ	12k バイト	1 ブロック
ユーザエリア	フラッシュメモリ	512k バイト	16 ブロック 4k バイト×8 32k バイト×1 64k バイト×7

1.2 動作モード

SH7086は、8種類の動作モード(モード0～7)があります。動作モードはFWE端子、モード端子(MD1、MD0)の設定で決まります。
モード0～2は、外部メモリおよび周辺デバイスをアクセスできる外部拡張モードです。
モード2、5、6では、プログラム実行開始後にバスコントローラにより、外部アドレス空間をエリアごとに設定できます。
モード3は、外部メモリおよび周辺デバイスへのアクセスをプログラム実行開始時に切り替えることができるシングルチップ起動拡張モードです。
モード4～7は、フラッシュメモリに書き込み/消去を行えるブートモード/ユーザブートモード/ユーザプログラムモードです。
FWE端子、モード端子(MD1、MD0)は、LSIの動作中に変化させないでください。
詳しくは、ハードウェアマニュアルを参照してください。

MCU 動作モード	端子設定			モード名	内蔵 ROM	CS0 空間のバス幅			
	FWE	MD1	MD0			SH7083	SH7084	SH7085	SH7086
モード 0	0	0	0	MCU 拡張モード 0	無効	8	8	16	16
モード 1	0	0	1	MCU 拡張モード 1	無効	16	16	32	32
モード 2	0	1	0	MCU 拡張モード 2	有効	BSC の CS0BCR により設定			
モード 3	0	1	1	シングルチップモード	有効	—			
モード 4	1	0	0	ブートモード	有効	—			
モード 5	1	0	1	ユーザブートモード	有効	BSC の CS0BCR により設定			
モード 6	1	1	0	ユーザプログラムモード	有効	BSC の CS0BCR により設定			
モード 7	1	1	1		有効	—			

1.3 オンボードプログラミングモード

オンボードプログラミングモードには、ブートモード、ユーザプログラムモード、ユーザブートモードの3種類があります。

項目	ブートモード	ユーザプログラムモード	ユーザブートモード
動作モード	モード 4	モード 6 モード 7	モード 5
機能	内蔵 SCI インタフェースを使用するプログラムモードで、ユーザエリアとユーザブートエリアの書き換えができます。 本モードでは、ホストと本 LSI 間のビットレートを自動であわせることができます。 最初にユーザエリアとユーザブートエリアが全面消去されます。	任意のインタフェースで、ユーザエリアの書き換えができます。	任意のインタフェースのユーザブートプログラム作成が可能で、ユーザエリアの書き換えが可能です。
制御プログラム	ブートエリア (内蔵ブートプログラム)	ユーザエリア (ユーザ作成ユーザプログラム)	ユーザブートエリア (ユーザ作成ユーザブートプログラム)
書き込み/消去可能エリア	ユーザエリア ユーザブートエリア	ユーザエリア	ユーザエリア
全面消去	○ (自動)	○	○
ブロック分割消去	○*1	○	○
書き込みデータ転送	ホストから SCI 経由	任意のデバイスから RAM 経由	任意のデバイスから RAM 経由
リセット起動	内蔵ブートプログラム 格納エリア(ブートエリア)	ユーザエリア	ユーザブートエリア*2
ユーザプログラムモードへの遷移	モード設定変更、 リセット	FLSHE ビット設定変更	モード設定変更、 リセット

【注】*1 いったん全面消去が行われます。その後、特定ブロックの消去を行うことができます。
*2 いったん組み込みプログラム格納エリアから起動し、フラッシュ関連レジスタのチェックが実行された後、ユーザブートエリアのリセットベクタから起動します。

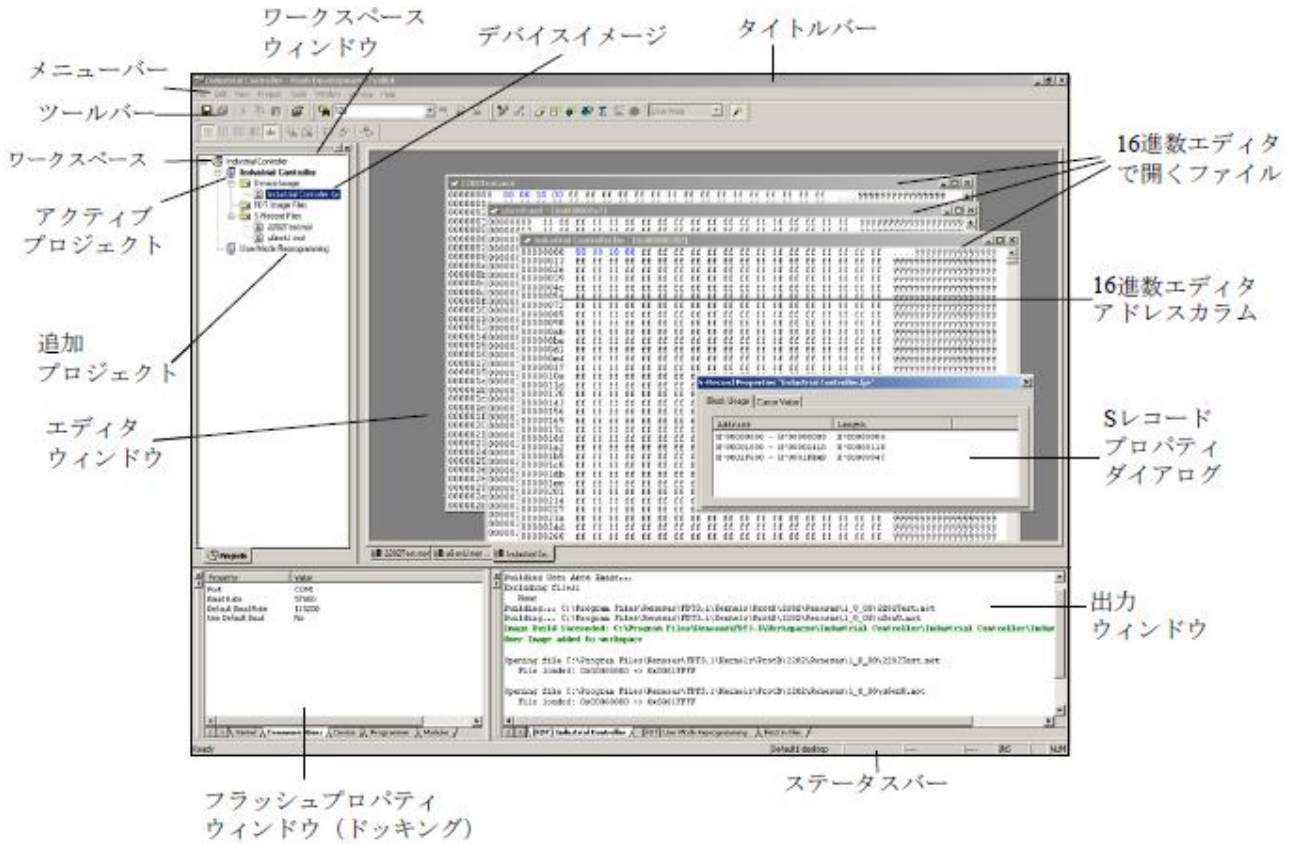
ユーザブートエリアの書き込み/消去は、ブートモードでのみ可能です。
ブートモードでは、いったんユーザエリアとユーザブートエリアが全面消去されます。
その後、コマンド方式でユーザエリアまたはユーザブートエリアの書き込みができますが、この状態になるまではエリア内容の読み出しはできません。
ユーザブートエリアだけ書き込んでユーザエリアの書き換えはユーザブートモードで実施する、あるいは、ユーザブートモードは使用しないためユーザエリアだけ書き換えるなどの使い方が可能です。
ユーザブートモードでは、ユーザプログラムモードと異なるモード端子設定で、任意のインタフェースのブート動作を実現できます。

2. フラッシュ開発ツールキットの機能

フラッシュ開発ツールキットは、高機能でかつ使い勝手の良いグラフィカルユーザインタフェースをもつルネサスフラッシュマイコン用オンボードフラッシュ書き込みツールです。
フラッシュ開発ツールキット は、ルネサスHigh-performance Embedded Workshop(HEW)とともに使用することで、ルネサスのフラッシュマイコンを使用している組み込みソフトウェア開発者に一貫した環境を提供します。
また、フラッシュ開発ツールキットは汎用のSレコード形式または16進数ファイルのエディタとして使用することもできます。

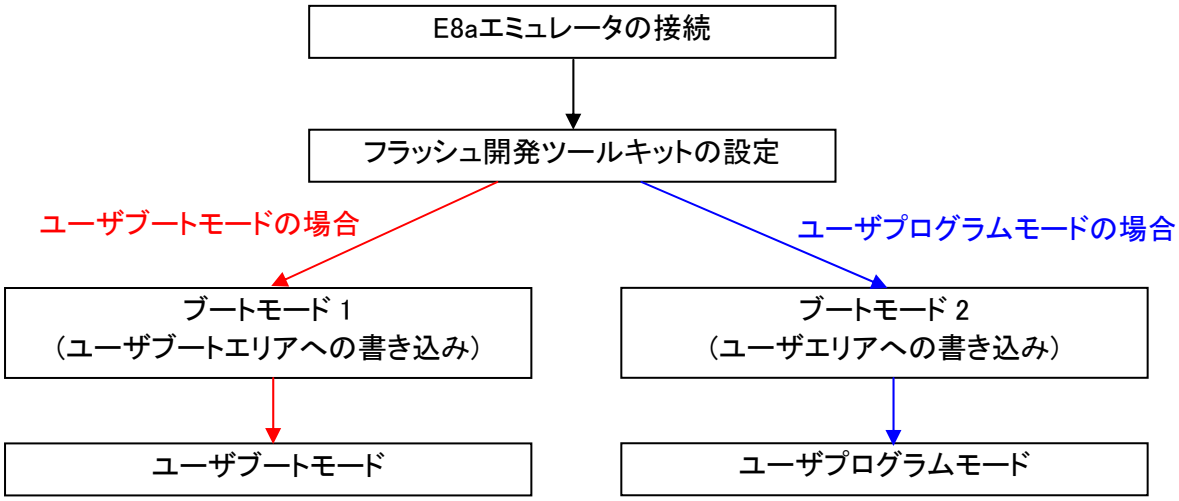
2.1 主な機能

・デバイスとの接続:デバイスをフラッシュ開発ツールキットインタフェースに接続します。
・デバイスとの切断:デバイスをフラッシュ開発ツールキットインタフェースから切断します。
・ブロック消去:[ブロック消去]ダイアログボックスを開き、デバイスのフラッシュメモリの特定ブロックまたは全ブロックを消去します。
・ブランクチェック:ターゲットデバイスのフラッシュ部が空白である/ないをチェックします。
・アップロード:ターゲットデバイスからデータをアップロードします。
・対象ファイルのダウンロード:16進数エディタでアクティブなファイルをダウンロードします。
・フラッシュのチェックサム:フラッシュメモリのデータのチェックサムを返します。
・フラッシュエリア指定:非書き込み(アップロード、ブランクチェックなど)操作が行われるフラッシュ領域を設定します。
・フラッシュ開発ツールキットには簡単に操作できるシンプルインターフェースモードとベーシックシンプルインターフェースモードがあります。
詳しくは[ルネサスフラッシュ開発ツールキット 4.01 ユーザーズマニュアル]を参照してください。



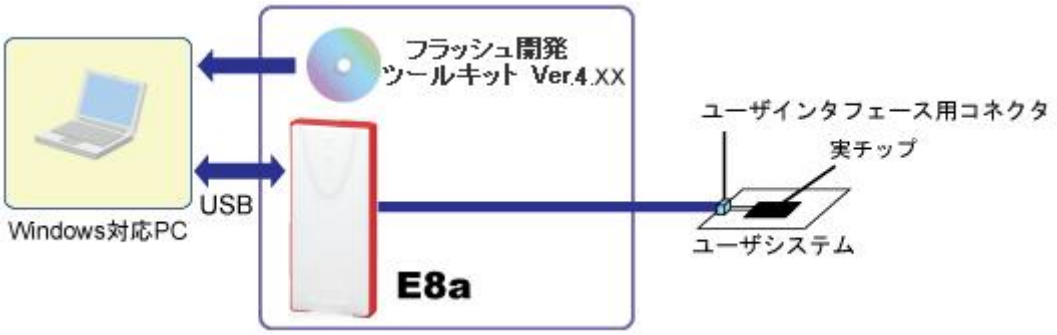
3. 各モードのフラッシュ開発ツールキット操作方法

ユーザブートモードで書き込みを行う場合はユーザブートエリアに、ユーザプログラムモードで書き込みを行う場合はユーザエリアに、予めユーザプログラムモード用ロードモジュールファイル(ユーザブートモード共用)が書き込まれている必要があります。
本書ではブートモードでユーザブートエリア/ユーザエリアにユーザプログラムモード用ロードモジュールファイルを書き込む説明を行ってから、ユーザブートモード、ユーザプログラムモードの説明を行います。手順としては以下のようになります。



3.1 E8a エミュレータの接続

E8aエミュレータはホストコンピュータとユーザシステム間に接続し、フラッシュ開発ツールキット(Flash Development Toolkit)を使って、ユーザシステム(オンボード)上のフラッシュマイコンに内蔵されたフラッシュメモリに対してユーザアプリケーションプログラムの書き込み/消去を行える機能を持ちます。

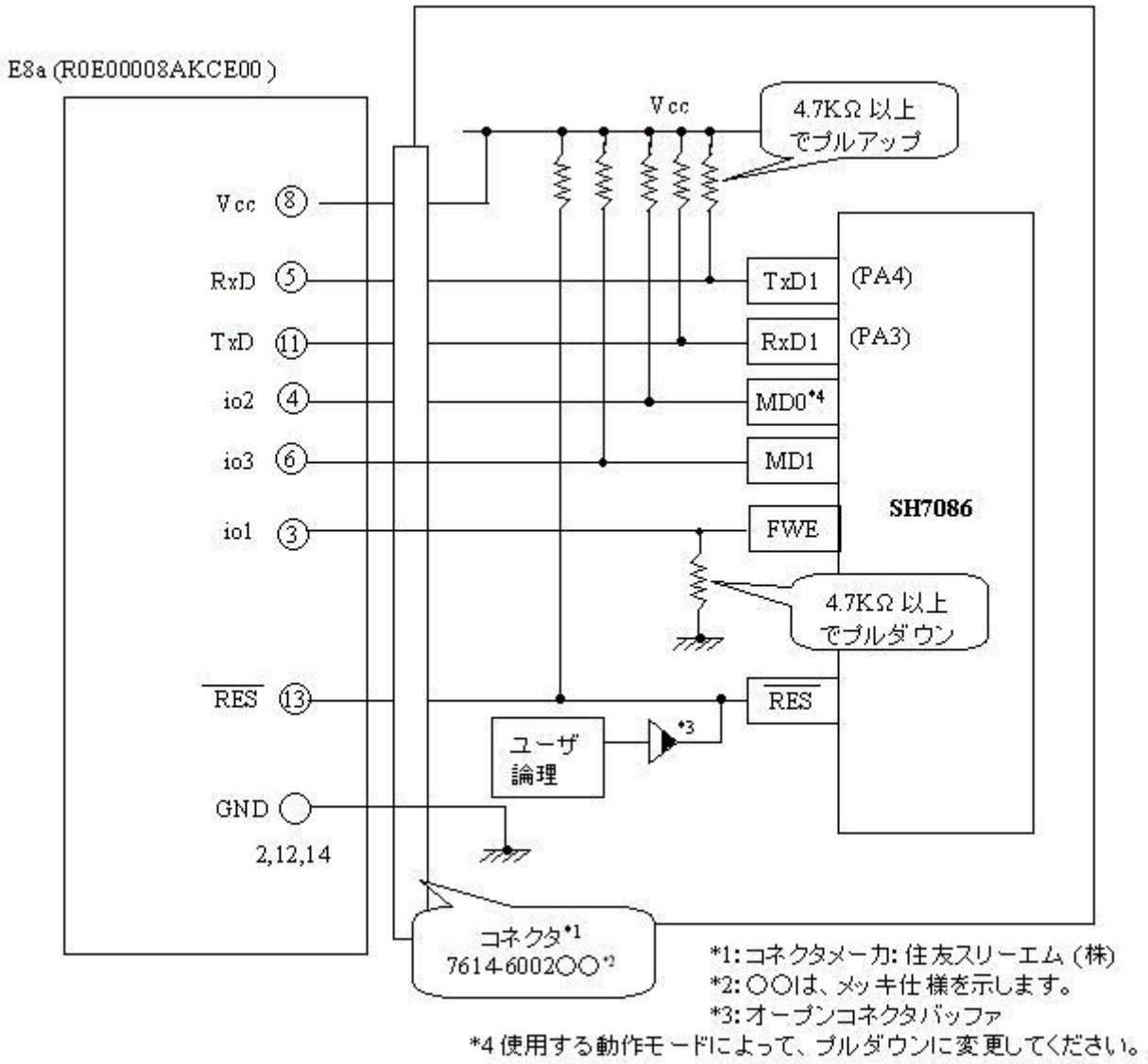


E8a エミュレータのピン番号、端子名とフラッシュ開発ツールキットピン設定の対応表を以下に示します。

ピン番号	E8a端子名	フラッシュ開発ツールキットピン設定
1	io0 / CLK	D
2	GND	-
3	io1	C
4	io2	A
5	RxD (ユーザ側 TxD)	-
6	io3	E
7	Io4/SIO	B
8	UVcc	-
9	UVcc2	-
10	Io6	F
11	TxD (ユーザ側 RxD)	-
12	GND	-
13	/RES	-
14	GND	-

【注】2,8,12,13,14 の端子は必ず接続してください。

E8a エミュレータと SH7086 の接続例を以下に示します。プルアップ及びプルダウンの抵抗値は参考値ですので、貴社システムにてご評価頂けるようお願い申し上げます。



このアプリケーションノートではSH7086ユーザシステムとして株式会社北斗電子製CPUボードHSB70865Fを使って説明しています。詳細は株式会社北斗電子のURLを参照してください。
株式会社北斗電子のURLは次のとおりです。
http://www.hokutodenshi.co.jp/

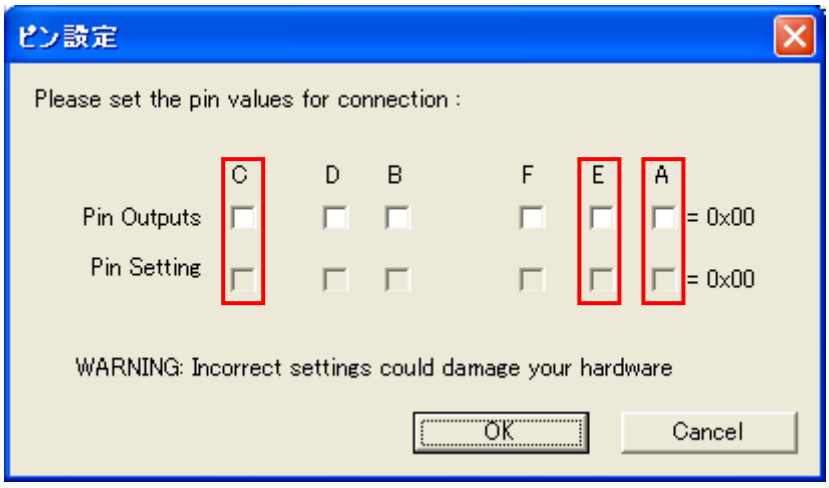
HSB70865F のピン番号と端子名の対応表を以下に示します。

ピン番号	端子名	ピン番号	端子名
1	_RES	2	GND
3	FWE	4	GND
5	MD0	6	GND
7	MD1	8	GND
9	I/O0	10	GND
11	I/O1	12	GND
13	I/O2	14	GND
15	TxD	16	GND
17	RxD	18	Vcc
19	SCK	20	Vcc

E8aエミュレータは14pin接続、HSB70865Fは20pin接続ですので本書では株式会社北斗電子製14⇔20pin変換コネクタFDM-E8aJを使用しています。詳細は株式会社北斗電子のURLを参照してください。
株式会社北斗電子のURLは次のとおりです。
http://www.hokutodenshi.co.jp/
FDM-E8aJ の変換対応表を以下に示します。

20pin(HSB70865F端子名)	14pin(FDTピン設定)
1pin (_RES)	13pin
2,4,6,8,10,12,14,16pin (GND)	2,12,14pin
15pin (TxD)	5pin
17pin (RxD)	11 pin
18pin (Vcc)	8pin
20pin (Vcc)	9pin
3pin (FWE)	3pin (C)
5pin (MD0)	4pin (A)
7pin (MD1)	6pin (E)
9pin (I/O0)	7pin (B)
11 pin (I/O1)	10pin (F)
13pin (I/O2)	-
19pin (SCK)	1pin (D)

SH7086 でフラッシュ開発ツールキットを使用する場合、FWE 端子、MD1 端子、MD0 端子を操作します。
このアプリケーションノートではフラッシュ開発ツールキットのピン設定の C ピン(FWE)、E ピン(MD1)、A ピン(MD0)を設定します。

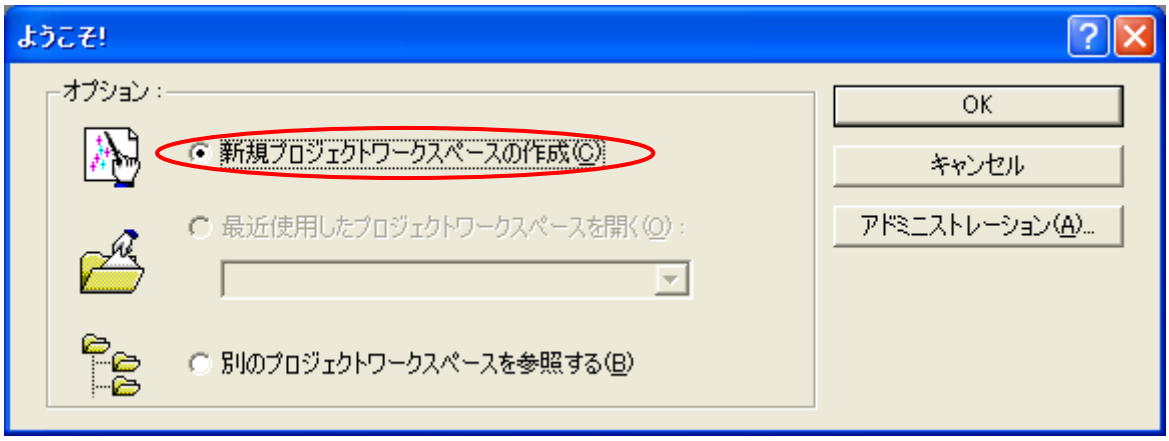


3.2 フラッシュ開発ツールキットの設定

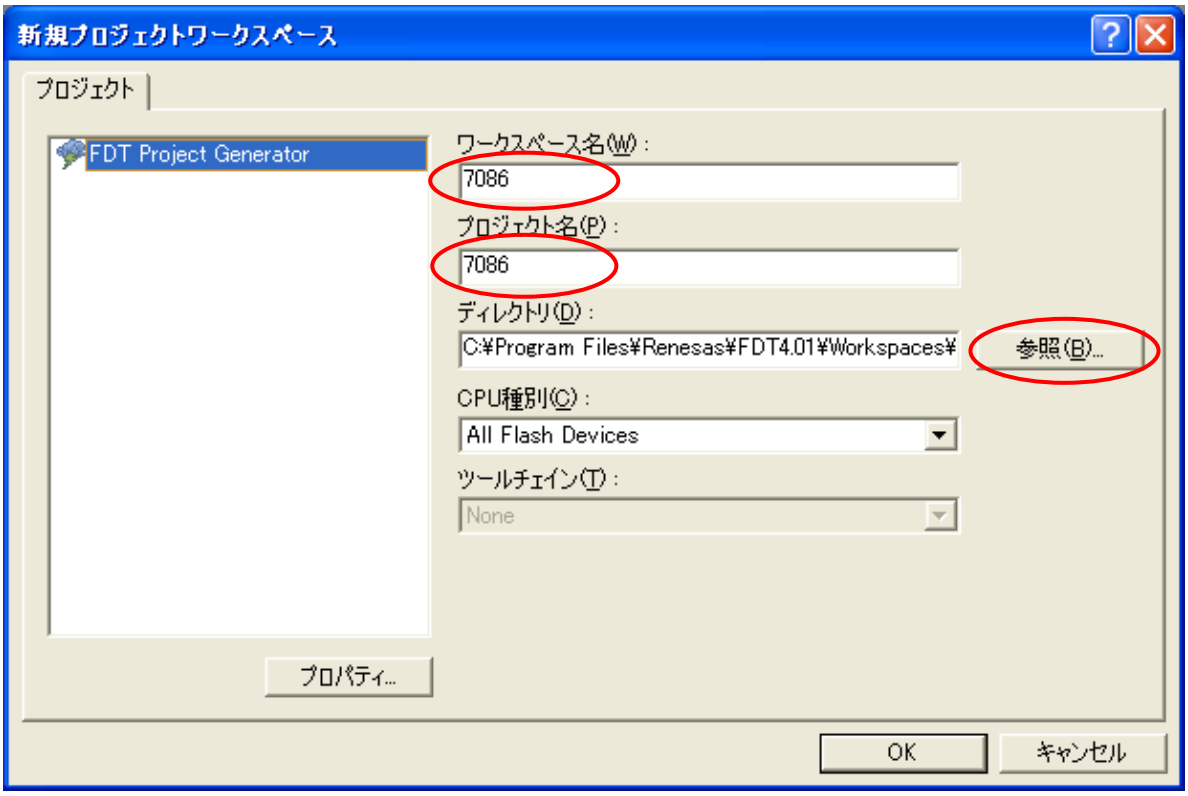
フラッシュメモリにプログラムを書き込むため、最初にフラッシュ開発ツールキットを設定します。

(1) フラッシュ開発ツールキットを起動します。
すべてのプログラムから[Flash Development Toolkit 4.01]を選択してください。

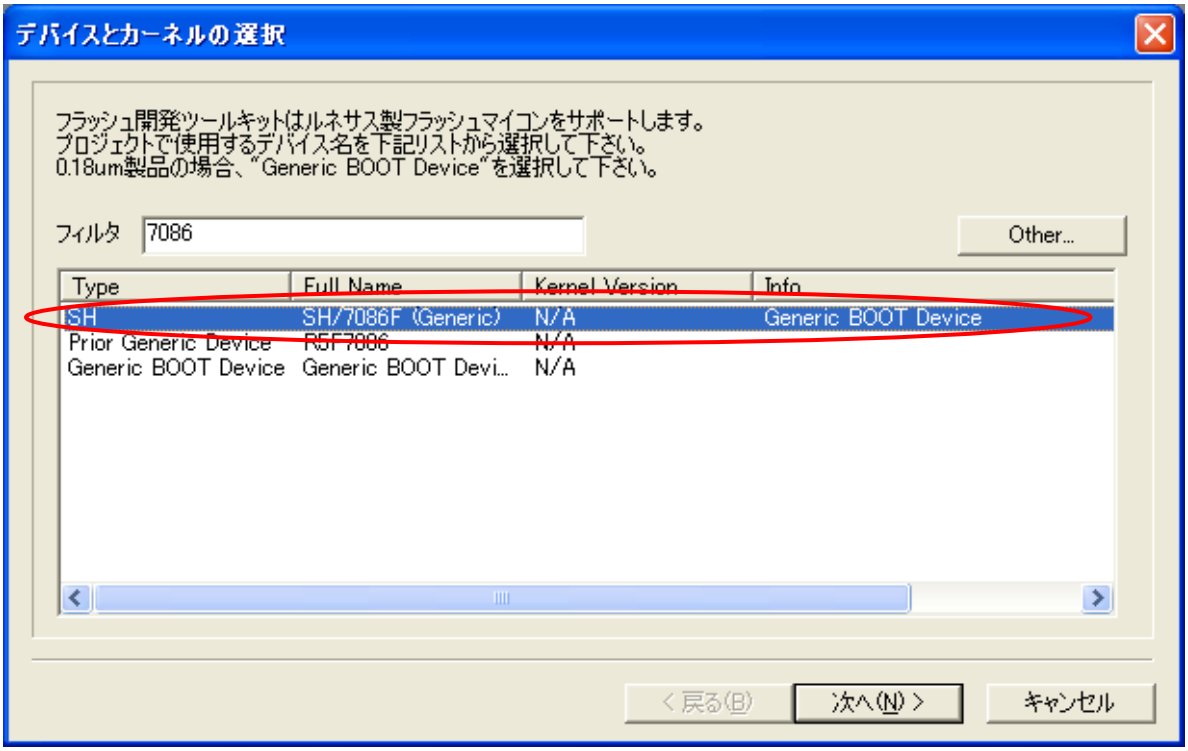
(2) フラッシュ開発ツールキットの[ようこそ!]画面が表示されます。
[新規プロジェクトワークスペースの作成]を選択して[OK]をクリックしてください。
2回目以降の起動では、前回選択したデバイスとポートの情報は保持されますので、[最近使用したプロジェクトワークスペースを開く]を選択して[OK]をクリックしてください。



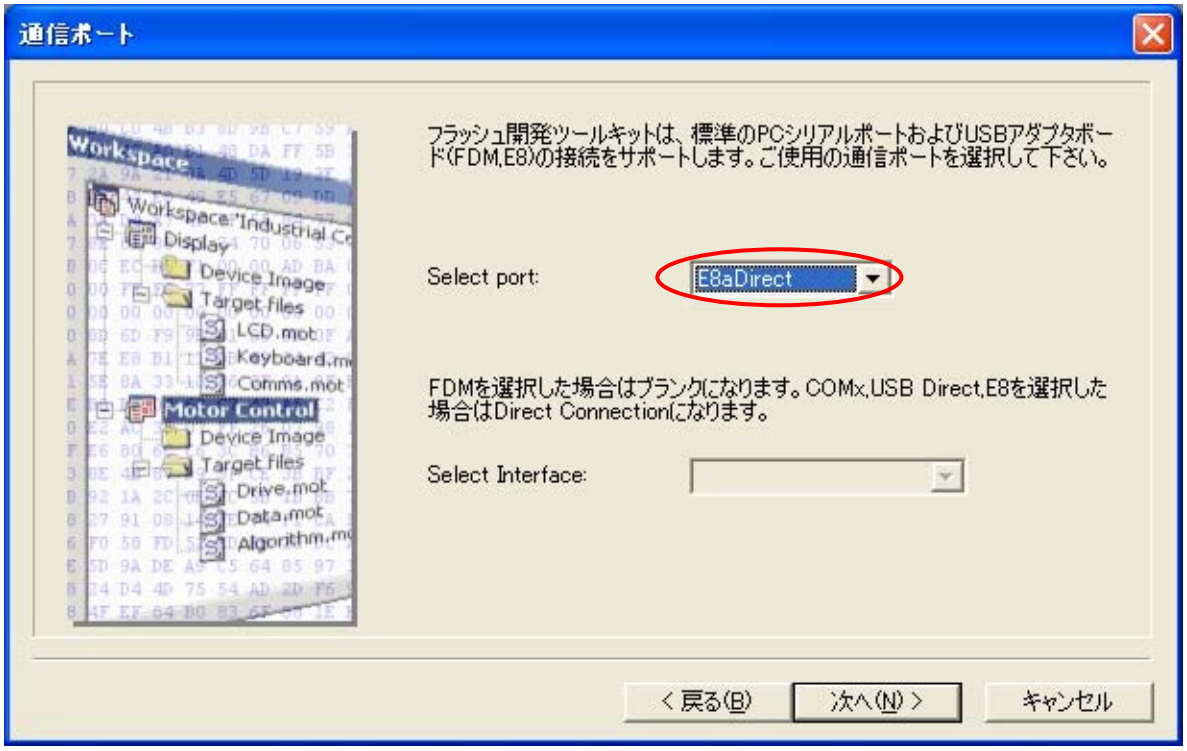
(3) 新規プロジェクトワークスペースを設定します。
ワークスペース名、プロジェクト名を指定してください。
ここでは、ワークスペース名とプロジェクト名を同じに指定します。
[参照]を選択するとワークスペースの保存場所を選択できます。
選択が終了したら[OK]をクリックしてください。



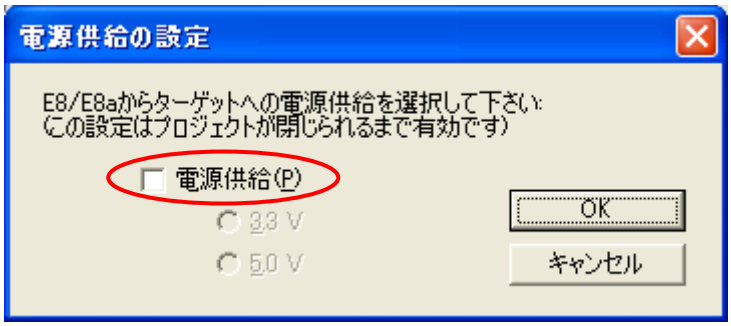
(4) 対象デバイスを選択します。
SH/7086F (Generic)を選択して[次へ]をクリックしてください。



(5) 通信ポートを選択します。
プルダウンメニューから E8aDirect を選択して[次へ]をクリックしてください。

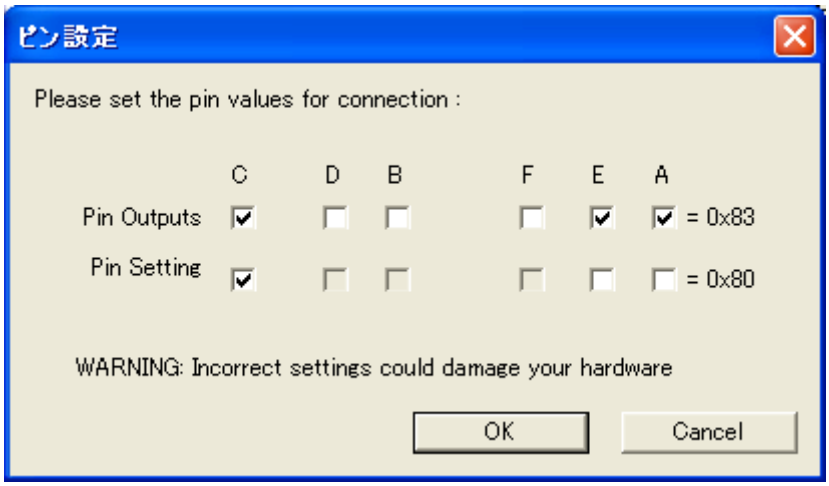


(6) 電源供給の設定をします。
[電源供給]のチェックをせずに[OK]をクリックしてください。



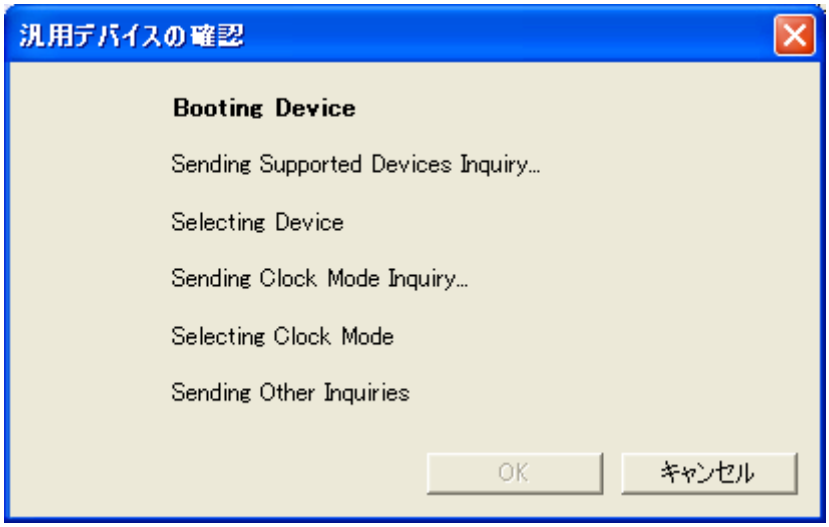
(7) ブートモードの E8a ピン設定をします。
C ピン(FWE)を 1 に、E ピン(MD1)、A ピン(MD0)を 0 に設定して[OK]をクリックしてください。

MCU 動作モード	端子設定			モード名
	FWE	MD1	MD0	
モード 4	1	0	0	ブートモード

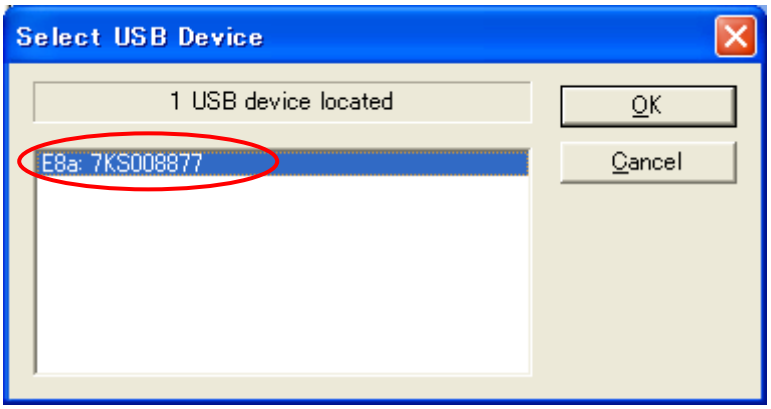


接続が終了したら[OK]をクリックしてください。

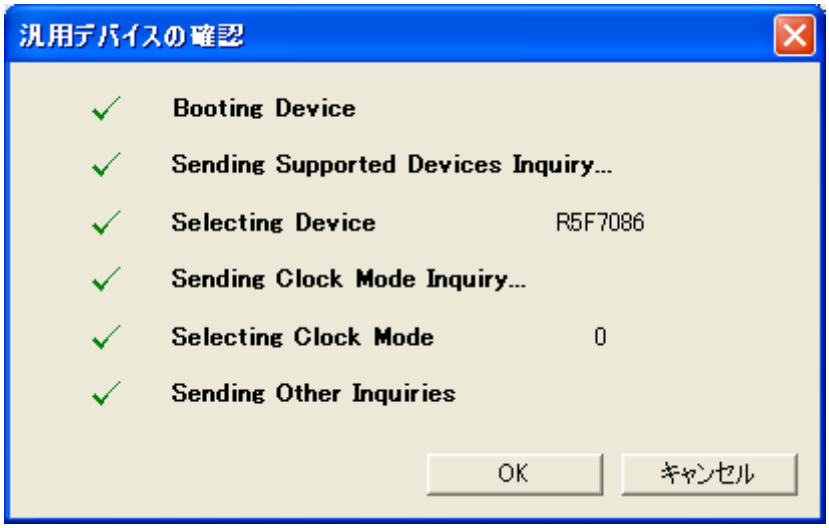
(8) デバイスの確認をします。



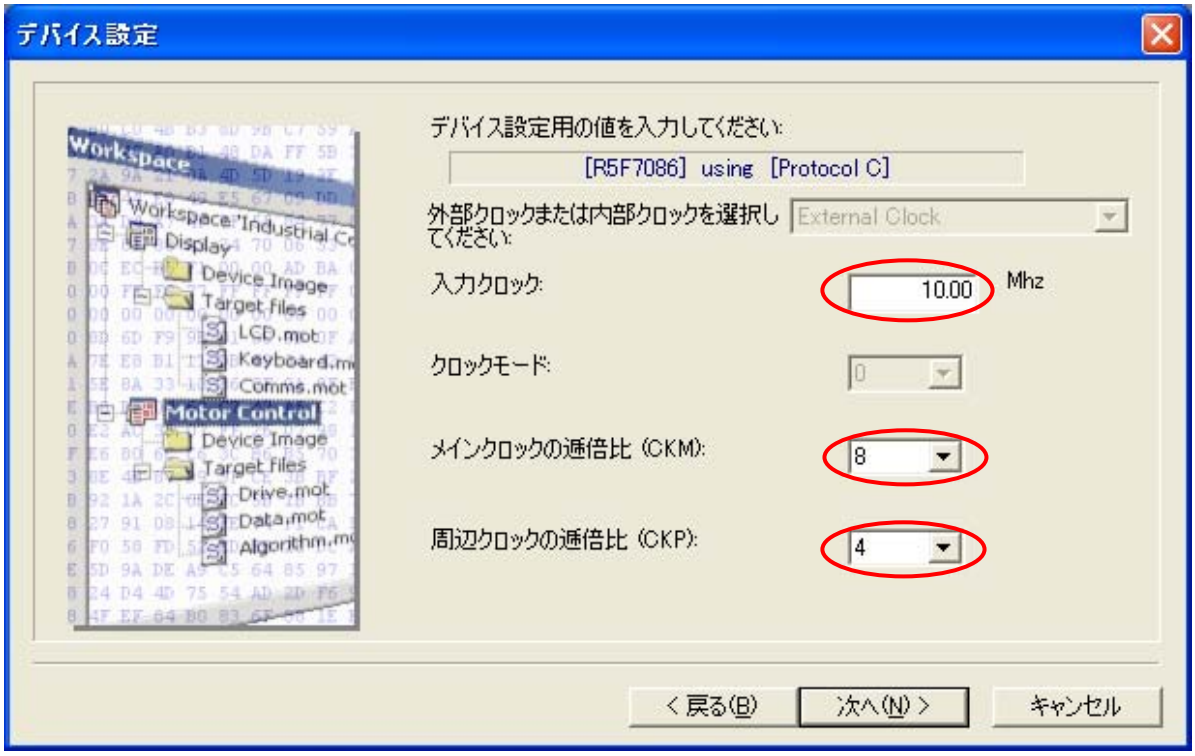
E8a を選択して[OK]をクリックしてください。



デバイスの確認が終了します。[OK]をクリックしてください。

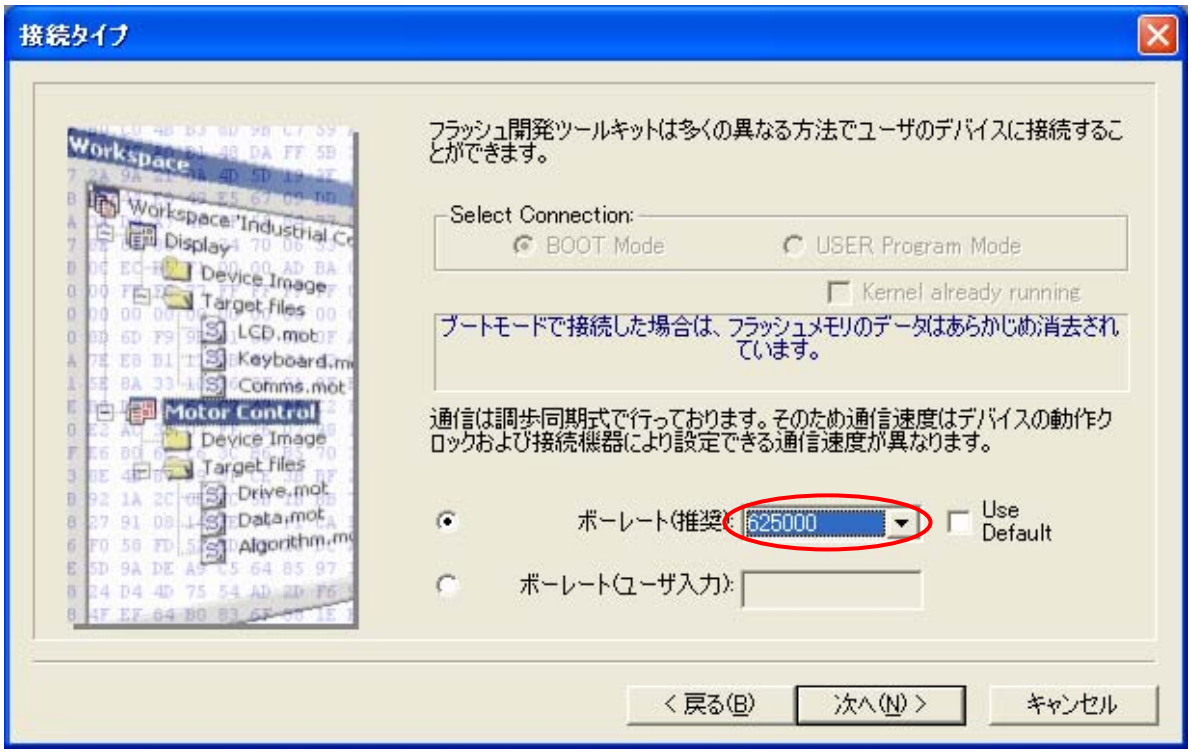


(9) デバイスの設定をします。
[入力クロック]にはボードに使用しているクロックの周波数をMHz単位で入力します。
10.00MHzを入力してください。
[メインクロックの逓倍比]を 8、[周辺クロックの逓倍比]を 4 に設定して[次へ]をクリックしてください。

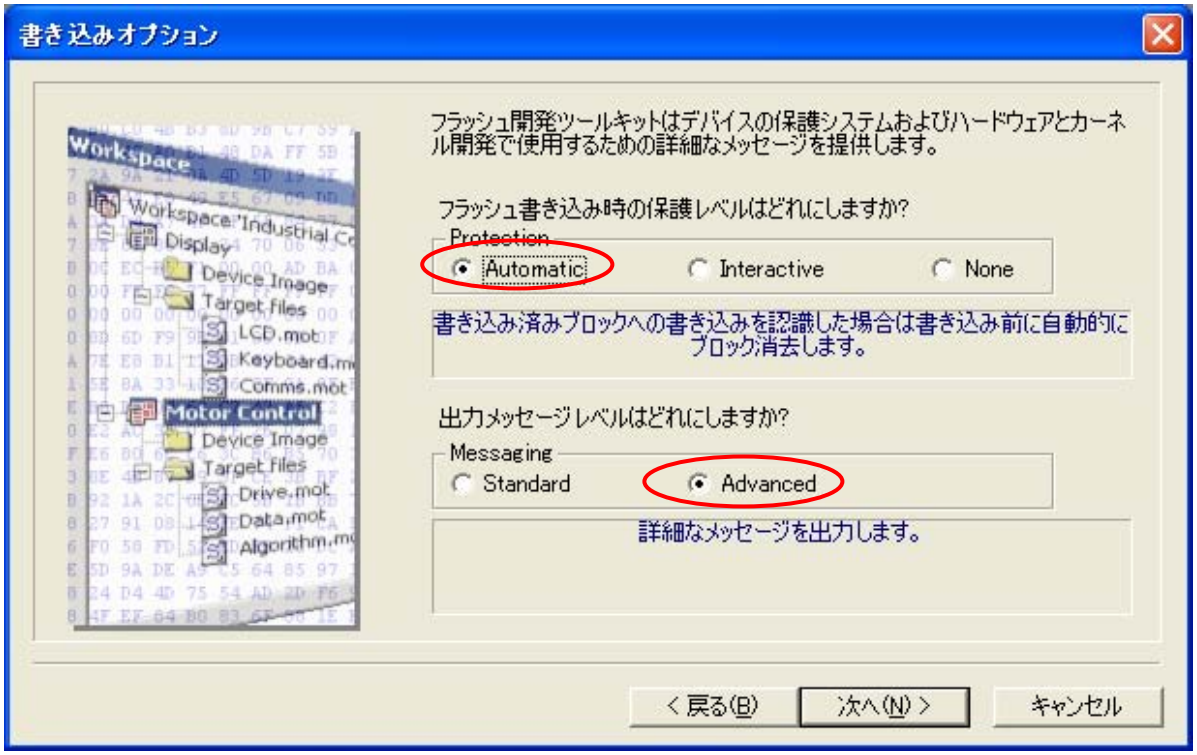


入力クロックとは、マイコンに直接入力している周波数です。ユーザシステムに接続されている水晶発信子またはセラミック発信子の周波数を入力してください。入力クロックと動作周波数(PLL出力)とは異なります。

(10) 接続タイプを設定します。
プルダウンメニューからボーレートを設定します。
625000 を選択して[次へ]をクリックしてください。

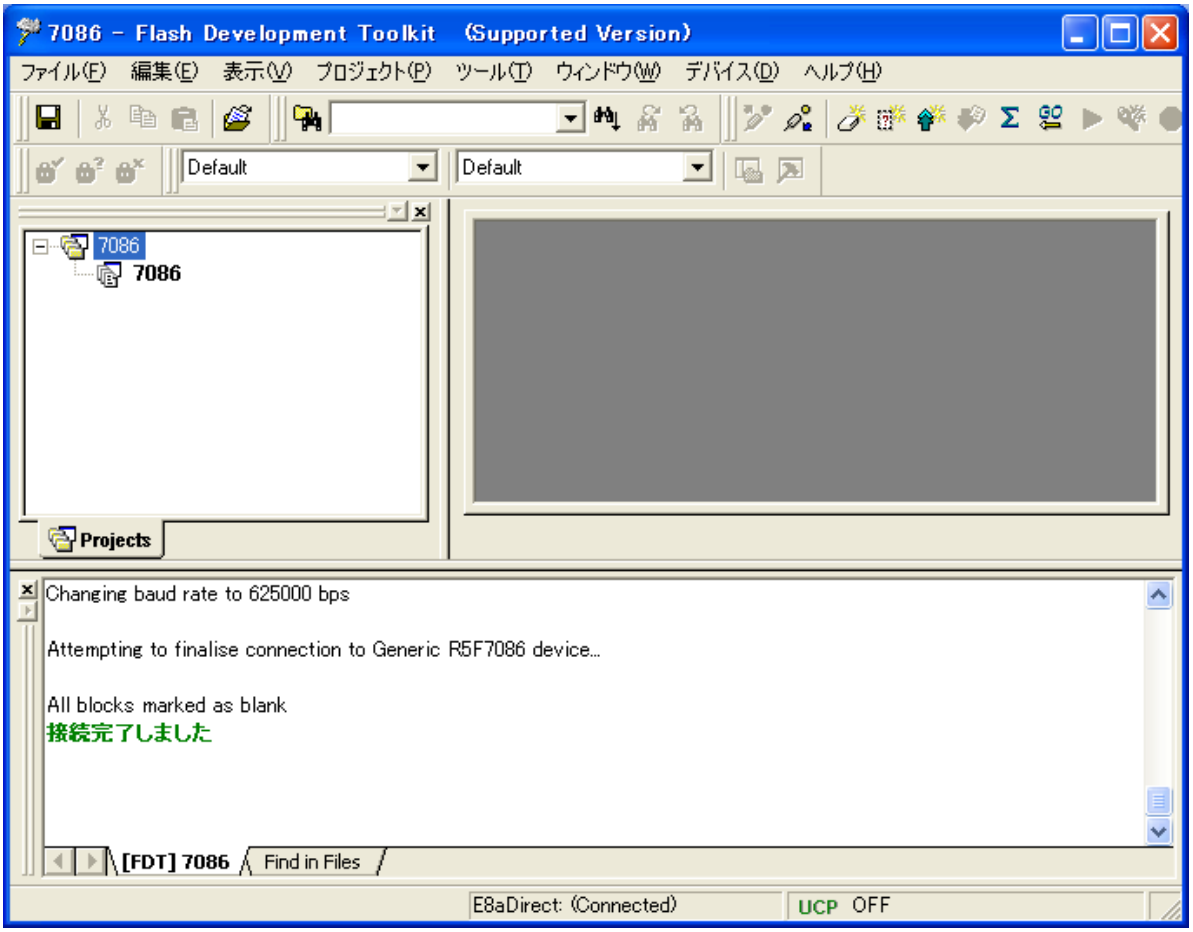


(11) 書き込みオプションを選択します。
保護レベルは[Automatic]、出力メッセージレベルは[Advanced]を選択して[次へ]をクリックしてください。

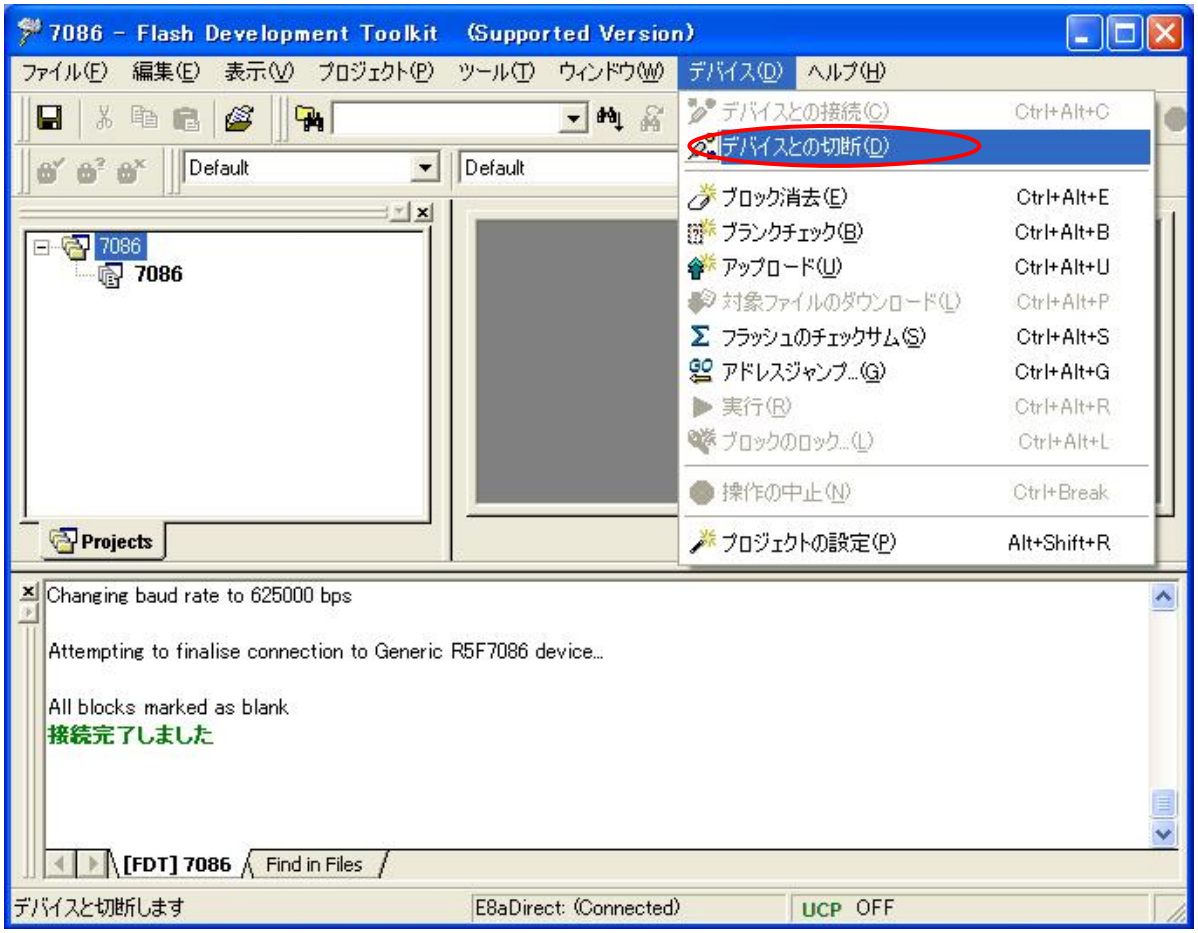


(12) デバイスをリセットモードで再起動したときのE8aのピンを設定します。
ここでは必要ないので[完了]をクリックしてください。

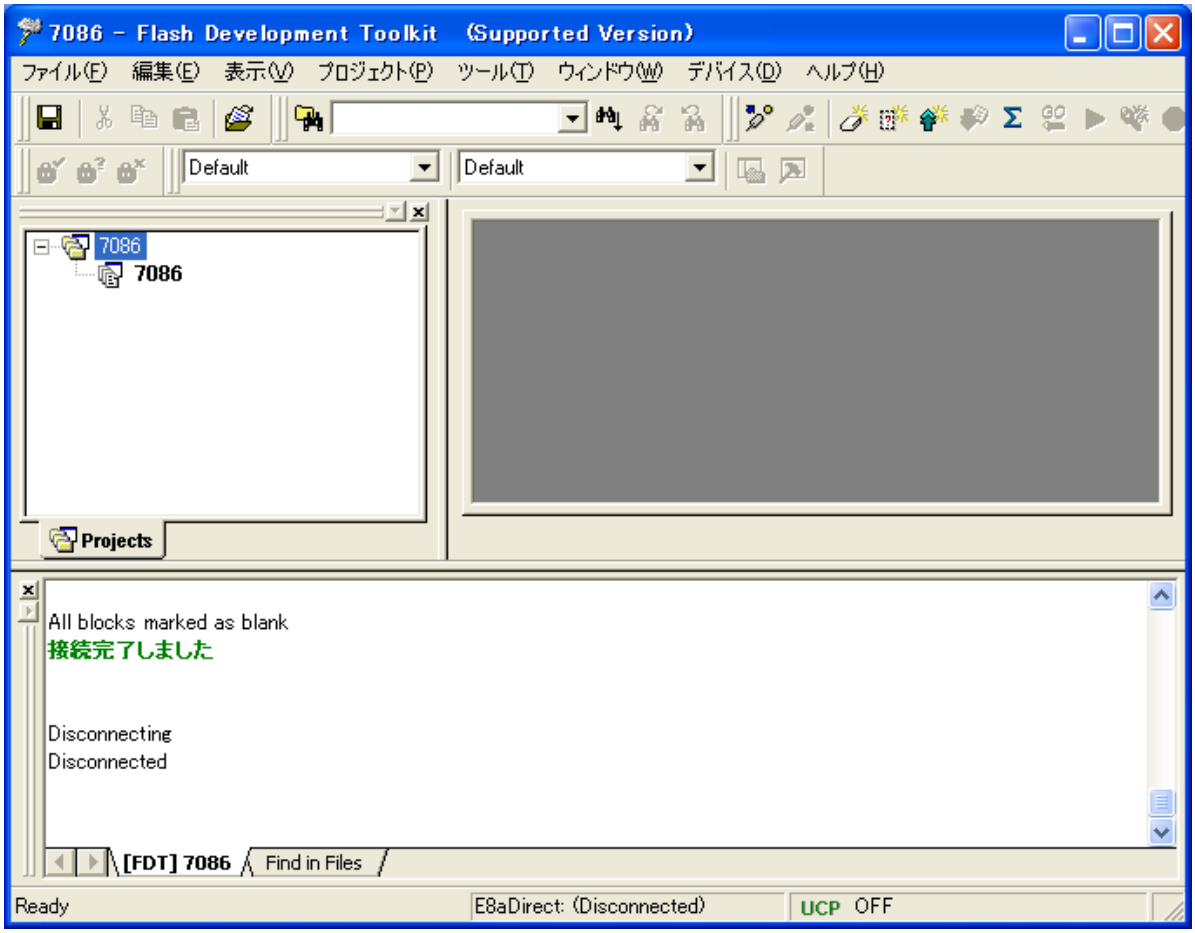
(13) ブートモードで、SH7086ボードがフラッシュ開発ツールキットに接続されました。
このとき、ユーザブートエリア、ユーザエリアの内容は消去されています。



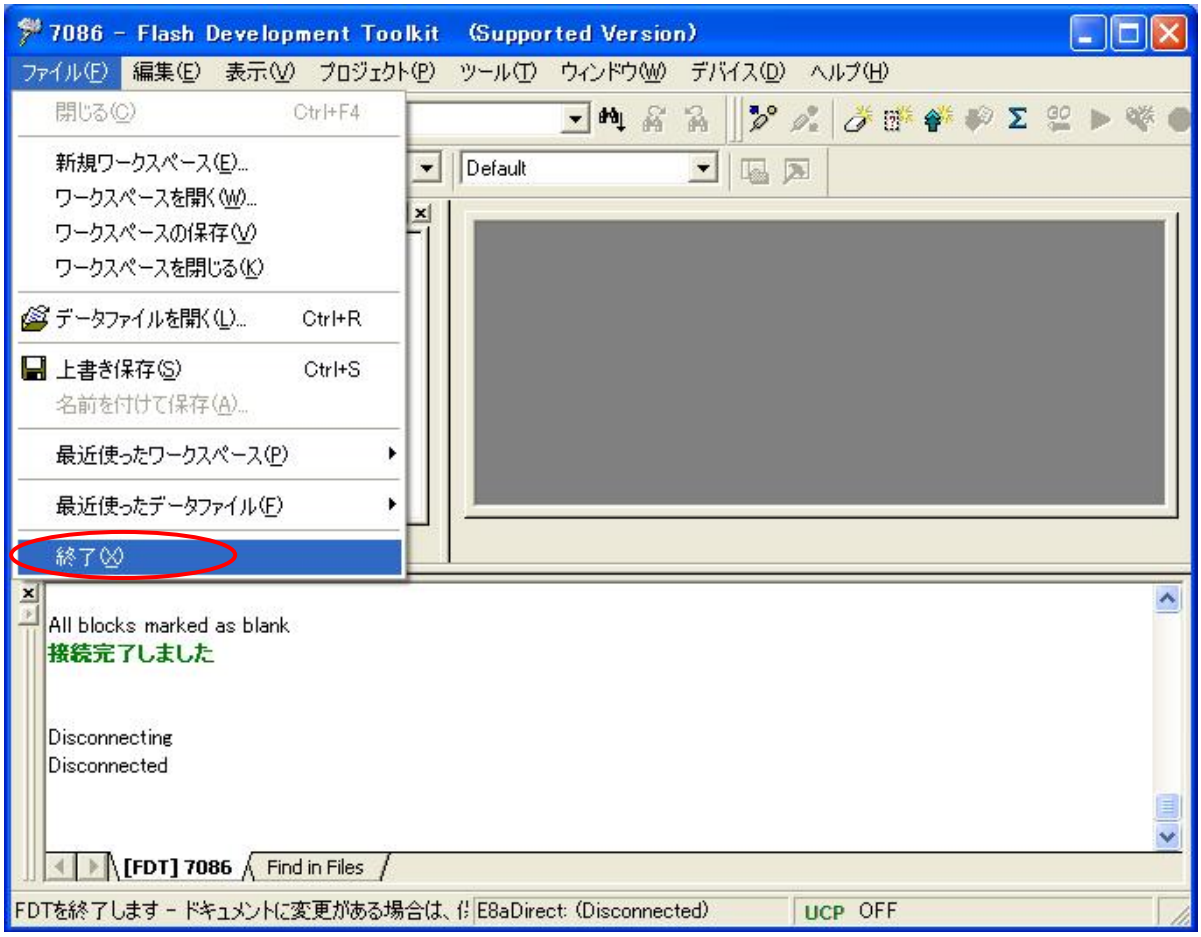
(14) デバイスを切断します。
[デバイス]メニューから[デバイスとの切断]をクリックしてください。



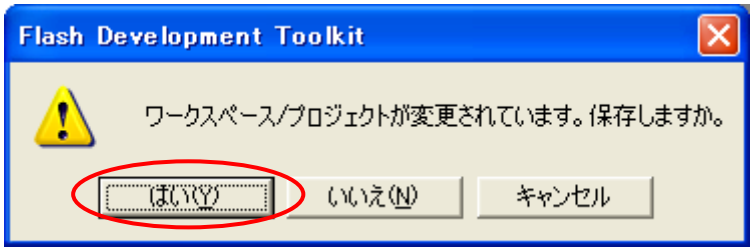
デバイスが切断されます。



(15) ワークスペースを保存して、終了します。
[ファイル]メニューから[終了]をクリックしてください。



[はい]をクリックしてください。

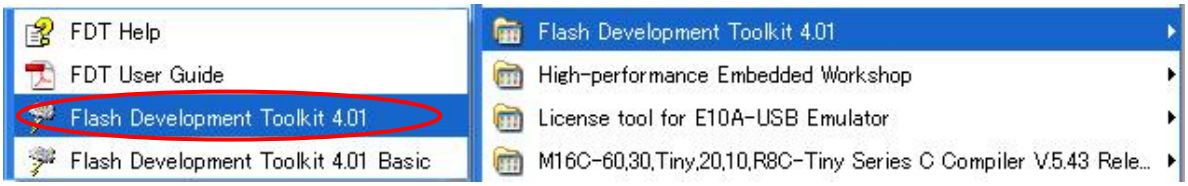


フラッシュ開発ツールキットが終了します。
フラッシュ開発ツールキットのワークスペースが 7086.AWS ファイルとして保存されます。

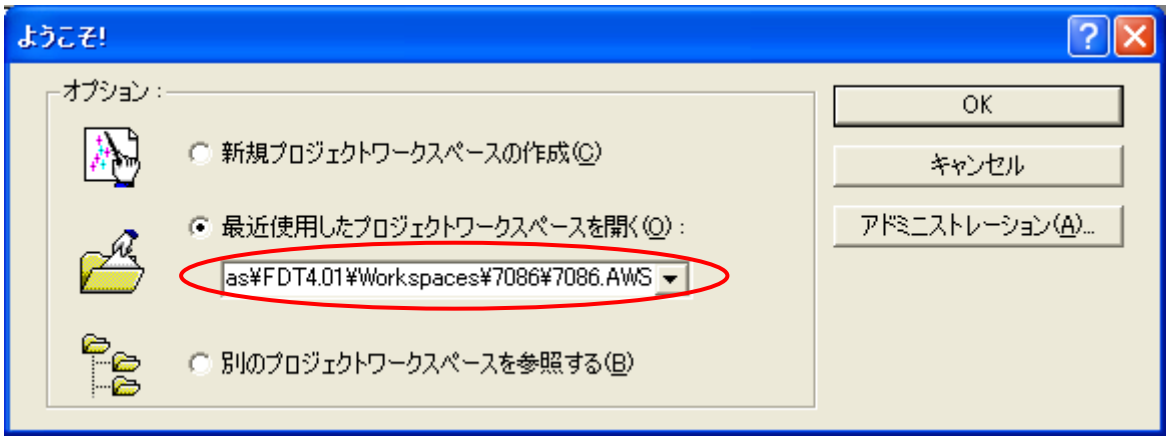
3.3 ブートモード 1(ユーザブートエリアへの書き込み)

ブートモードでユーザブートエリアへユーザプログラムモード用ロードモジュールファイルを書き込みます。
書き込むプログラムは、7086F.motファイル(Sタイプファイル)です。ここでは、保存したワークスペースファイル(7086.AWS)を使って起動します。
このプログラムは、すでにビットレートを周波数に対応して修正してあります。ビットレートの修正は、6.1 ヘッダファイル (1)ビットレートの設定(GenTest.h)を参照してください。

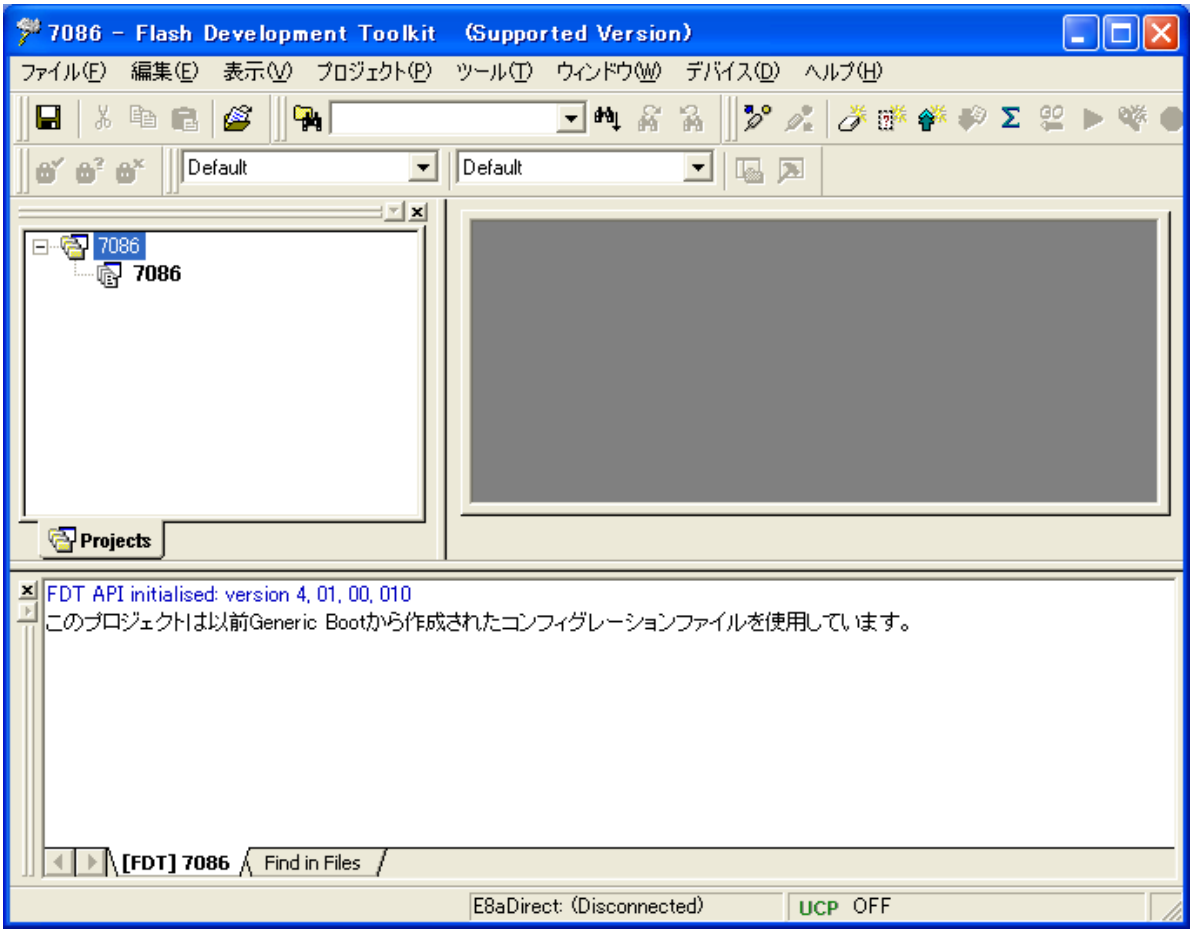
(1) フラッシュ開発ツールキットを起動します。
すべてのプログラムから[Flash Development Toolkit 4.01]を選択してください。



(2) フラッシュ開発ツールキットの[ようこそ!]画面が表示されます。
[最近使用したプロジェクトワークスペースを開く]を選択し、プロジェクトワークスペースファイル 7086.AWS を選択して[OK]をクリックしてください。

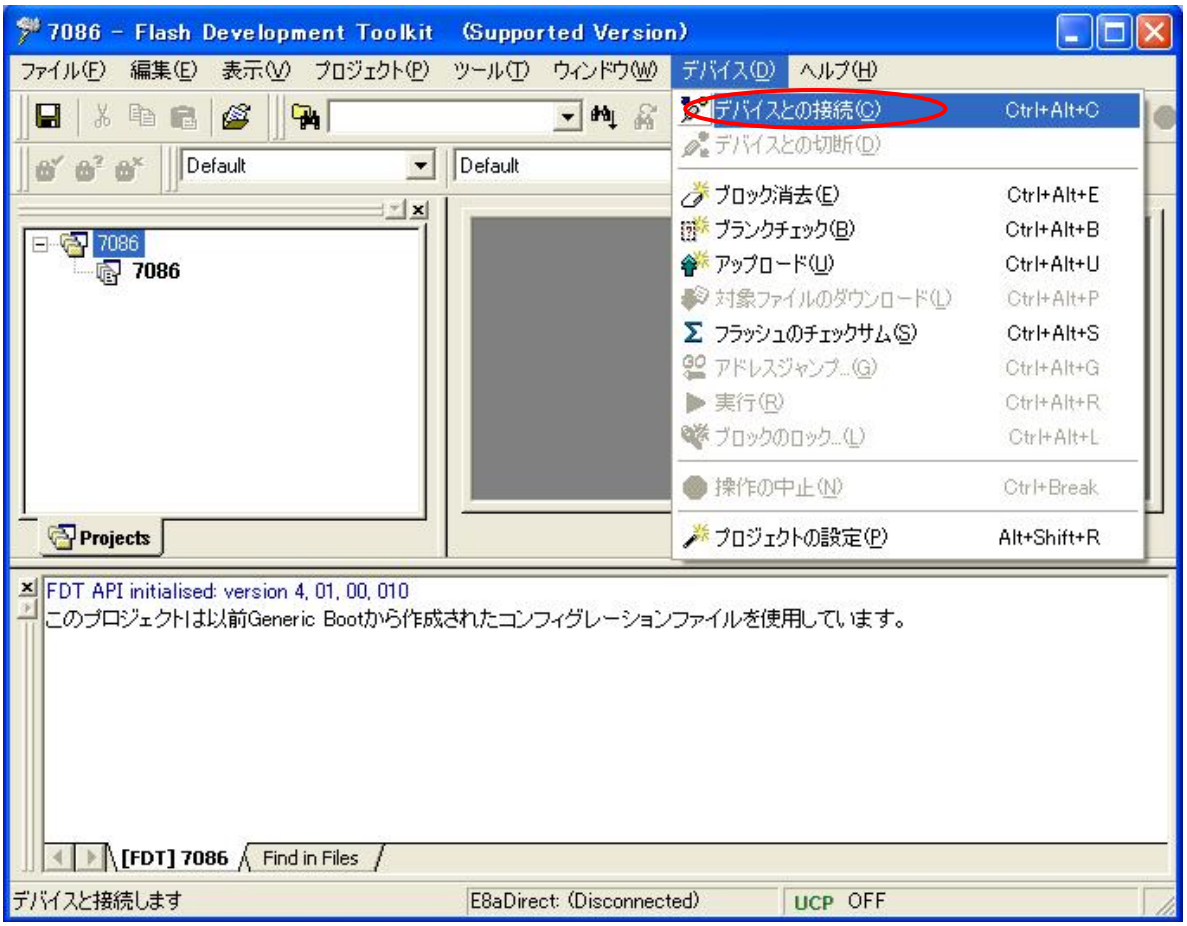


7086 プロジェクトが表示されます。

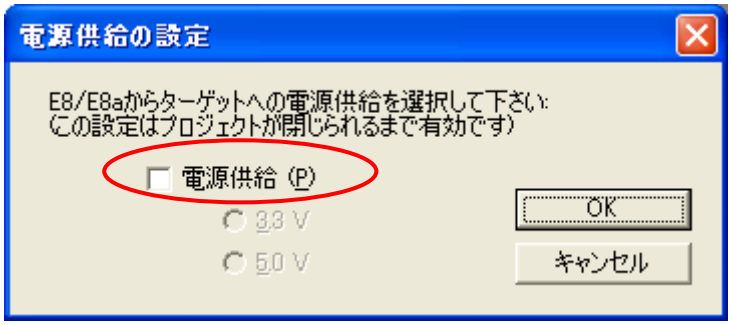


この操作は、直接プロジェクトワークスペースファイル 7086.AWS を開く（またはダブルクリック）でも、フラッシュ開発ツールキットを起動させることができます。

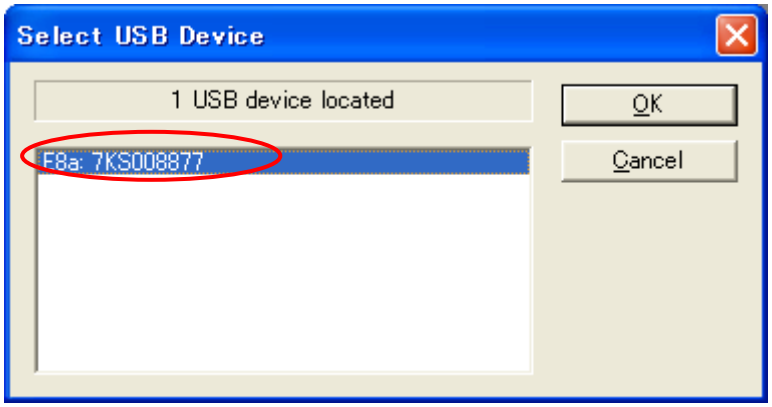
(3) [デバイス]メニューから[デバイスとの接続]をクリックしてください。



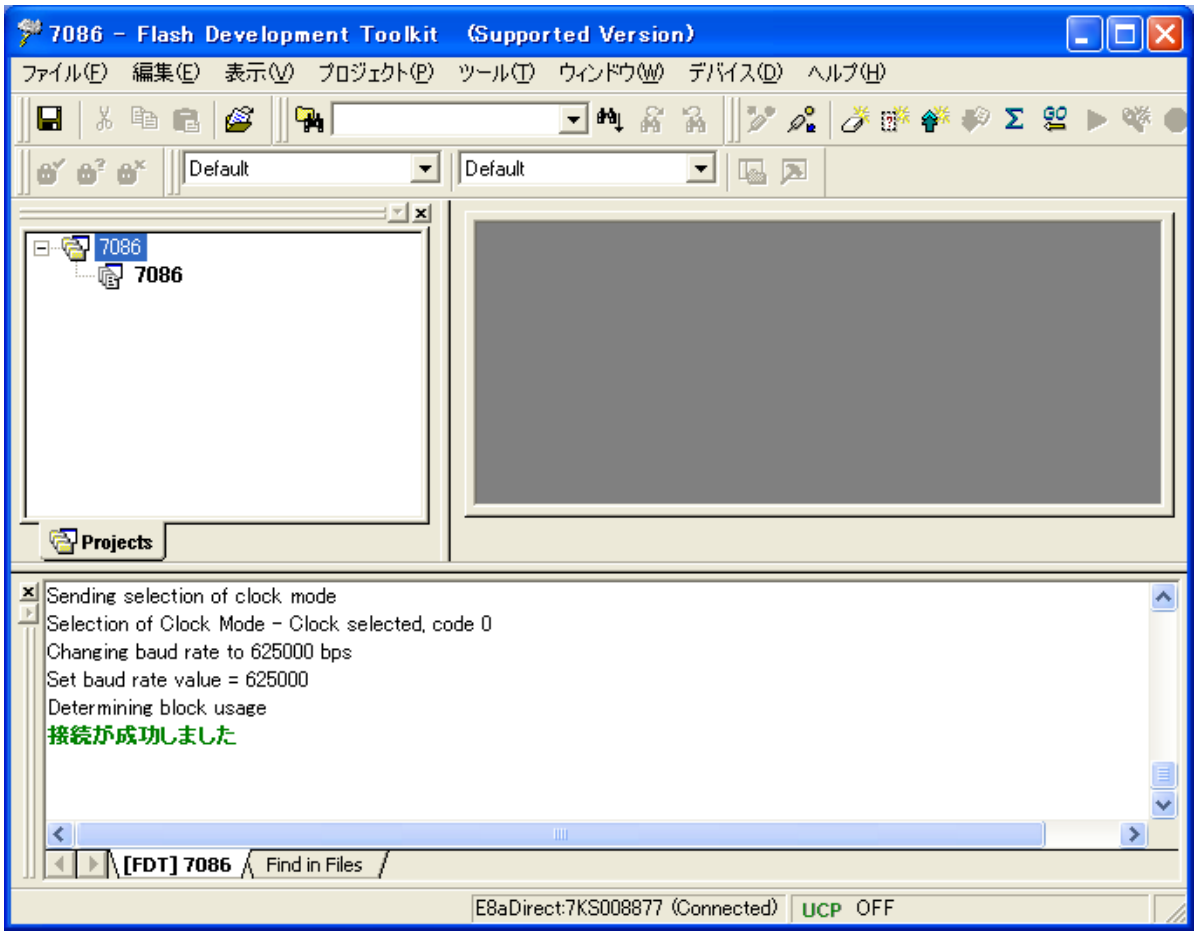
[電源供給]のチェックをせずに[OK]をクリックしてください。



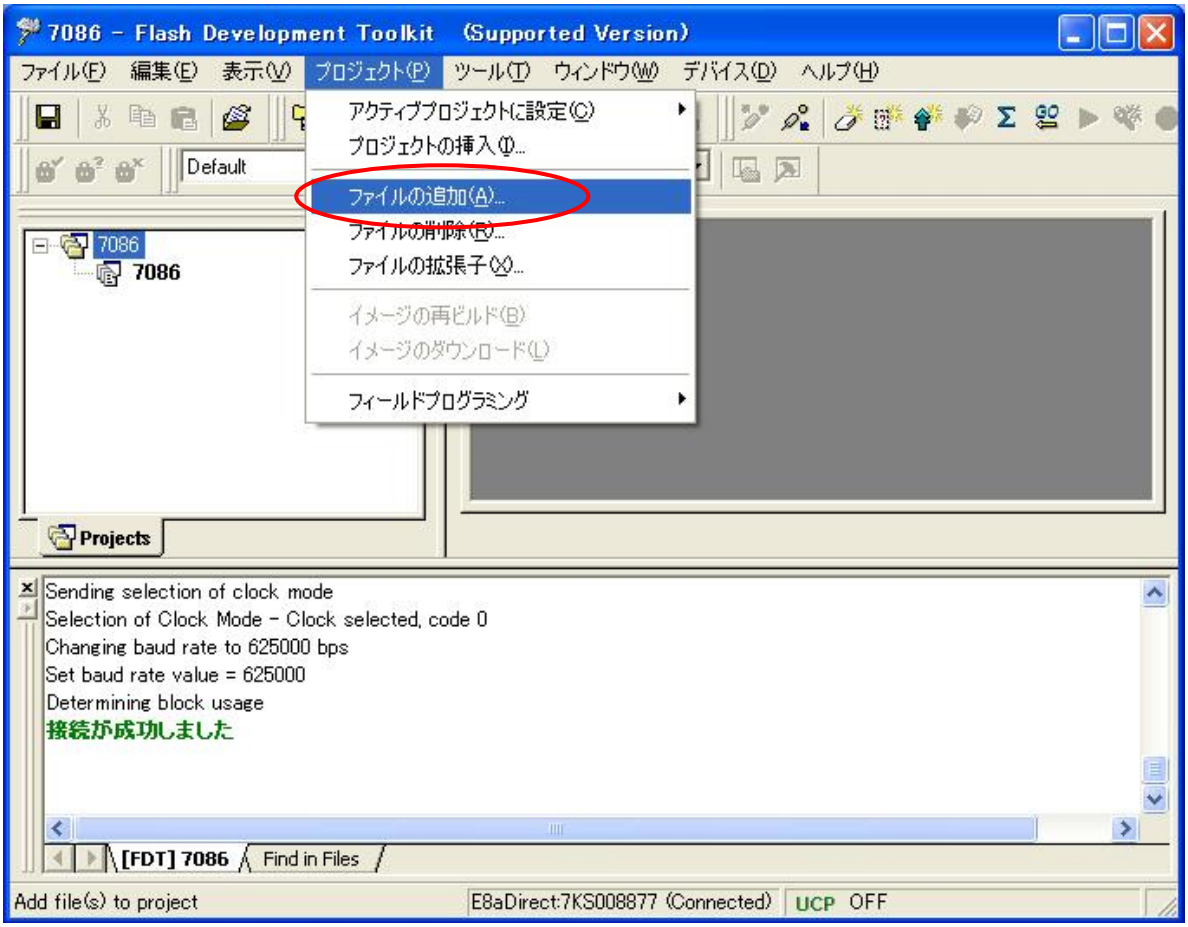
E8a を選択して[OK]をクリックしてください。



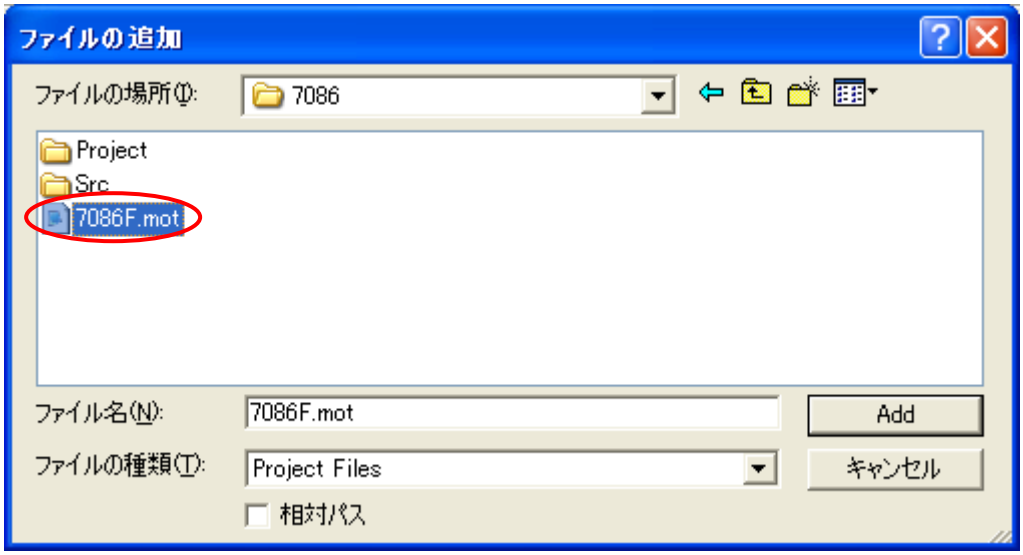
デバイスが接続されます。



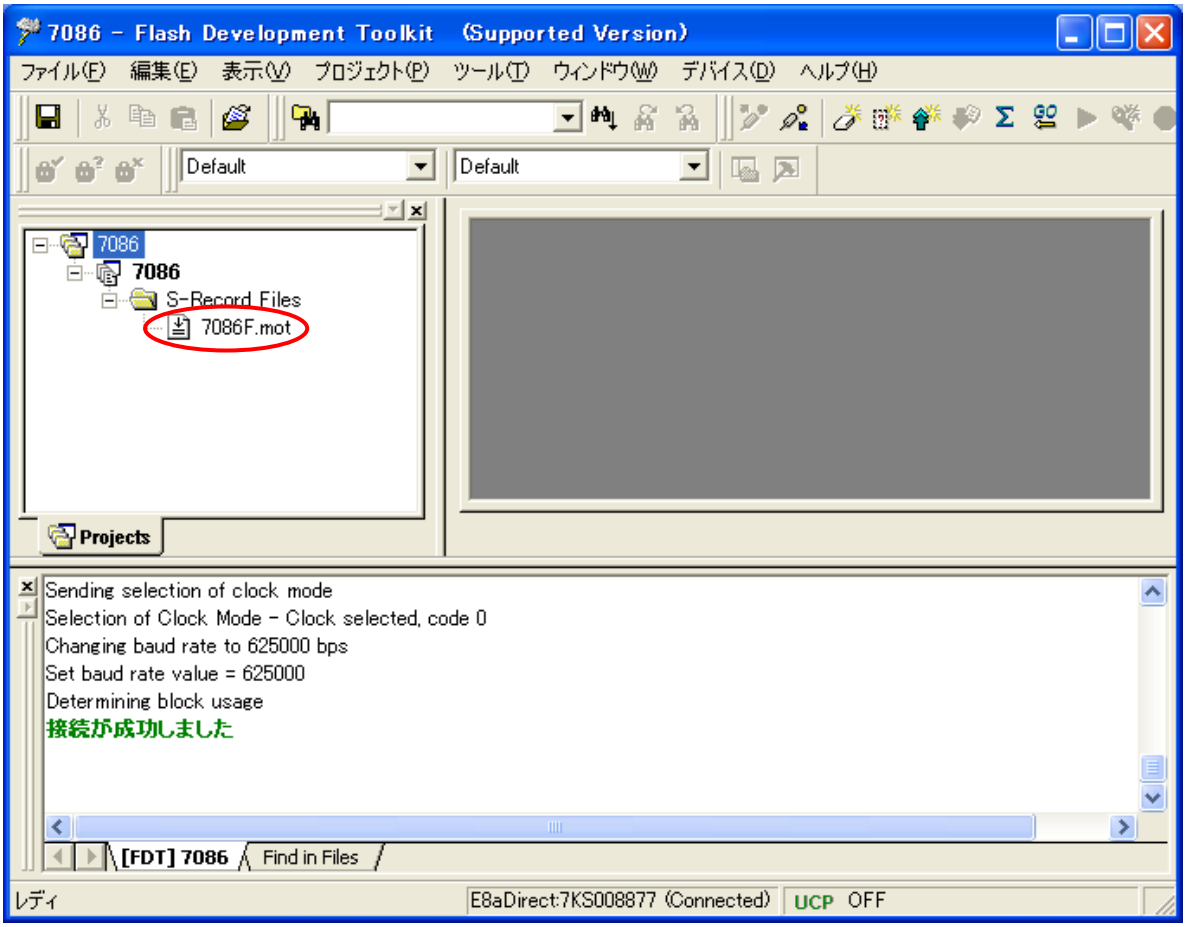
(4) 書き込みファイルを選択します。
[プロジェクト]メニューから[ファイルの追加]をクリックしてください。



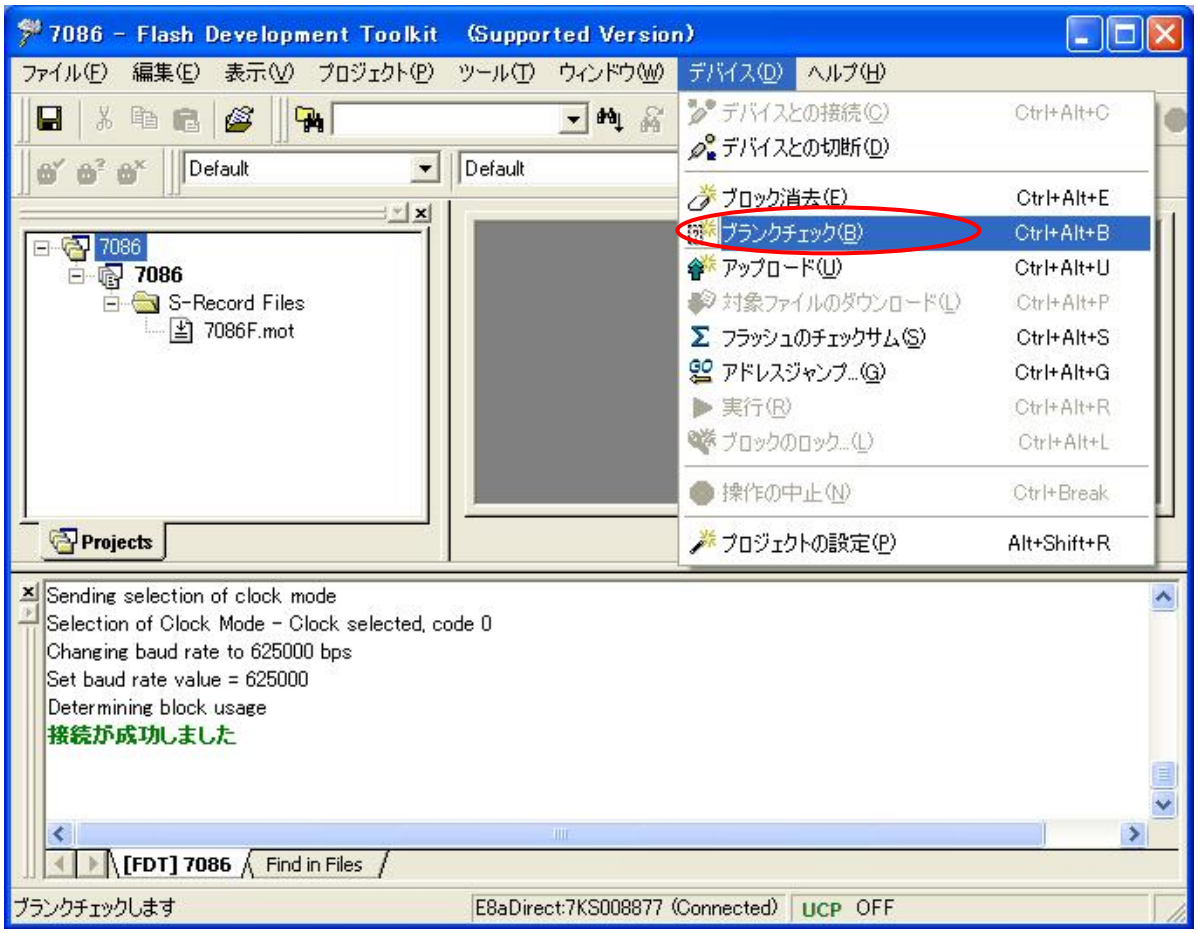
(5) [ファイルの追加]ダイアログボックスから[7086F.mot]ファイルを選択して[Add]をクリックしてください。



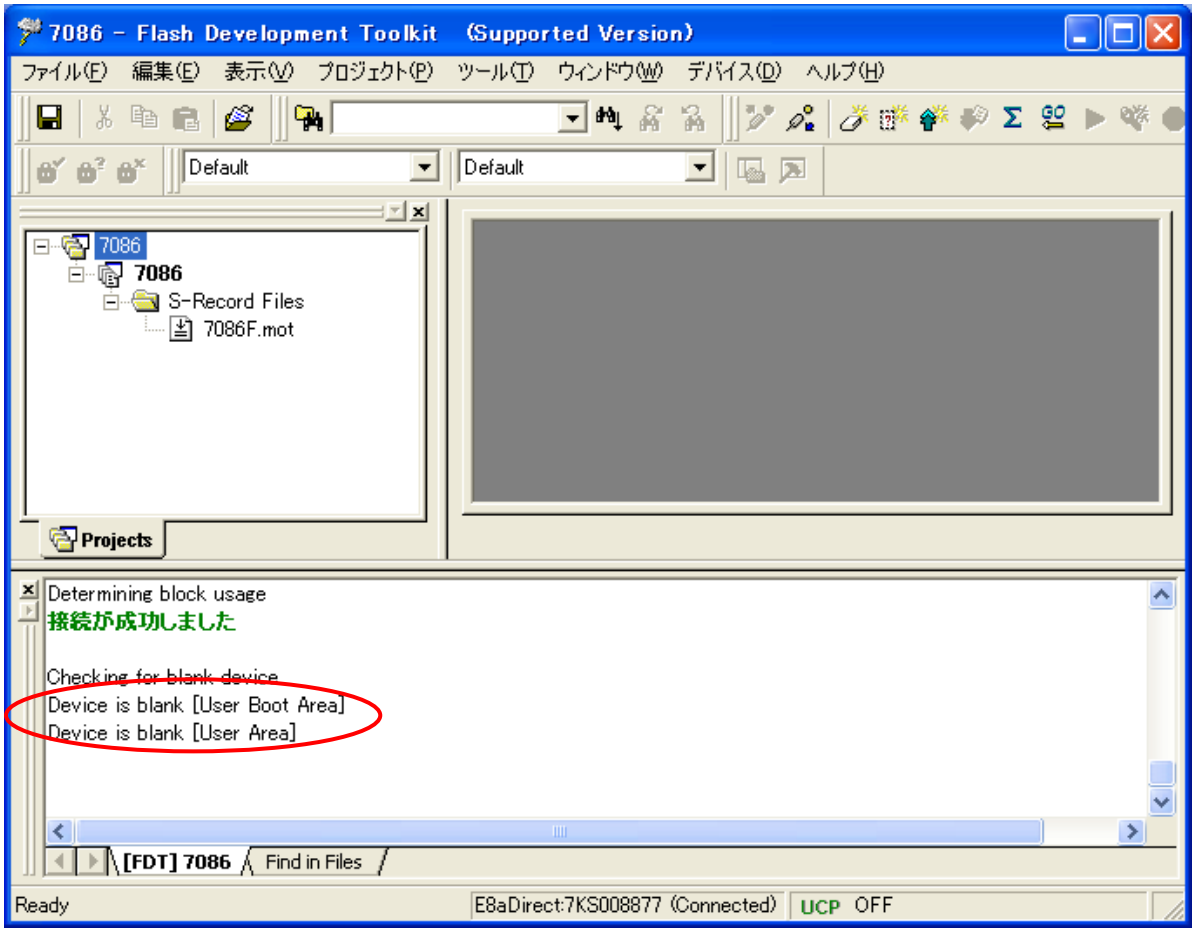
プロジェクトに 7086F.mot ファイルが追加されます。



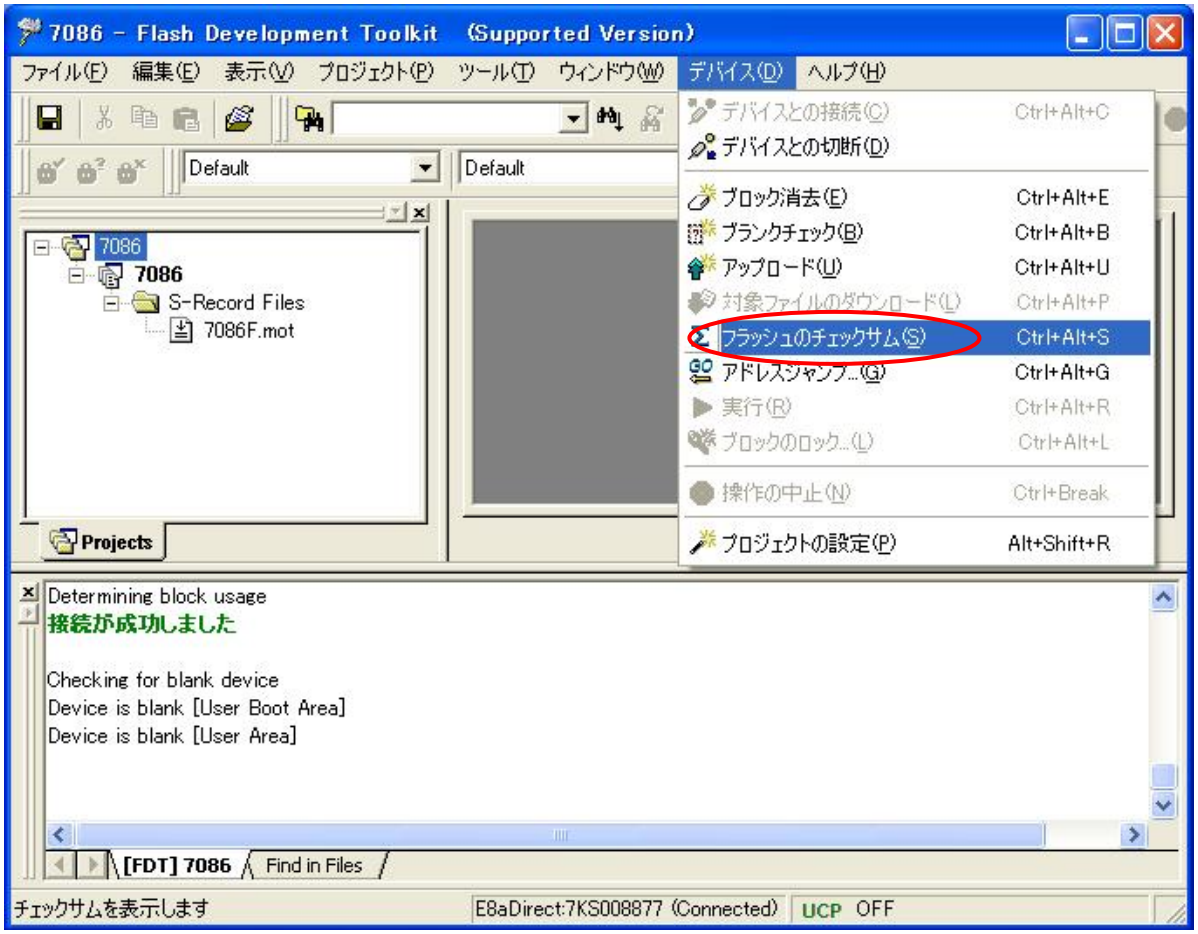
(6) ユーザブートエリア、ユーザエリアに何も書き込まれていないことを確認するために、ブランクチェックを行います。
[デバイス]メニューから[ブランクチェック]をクリックしてください



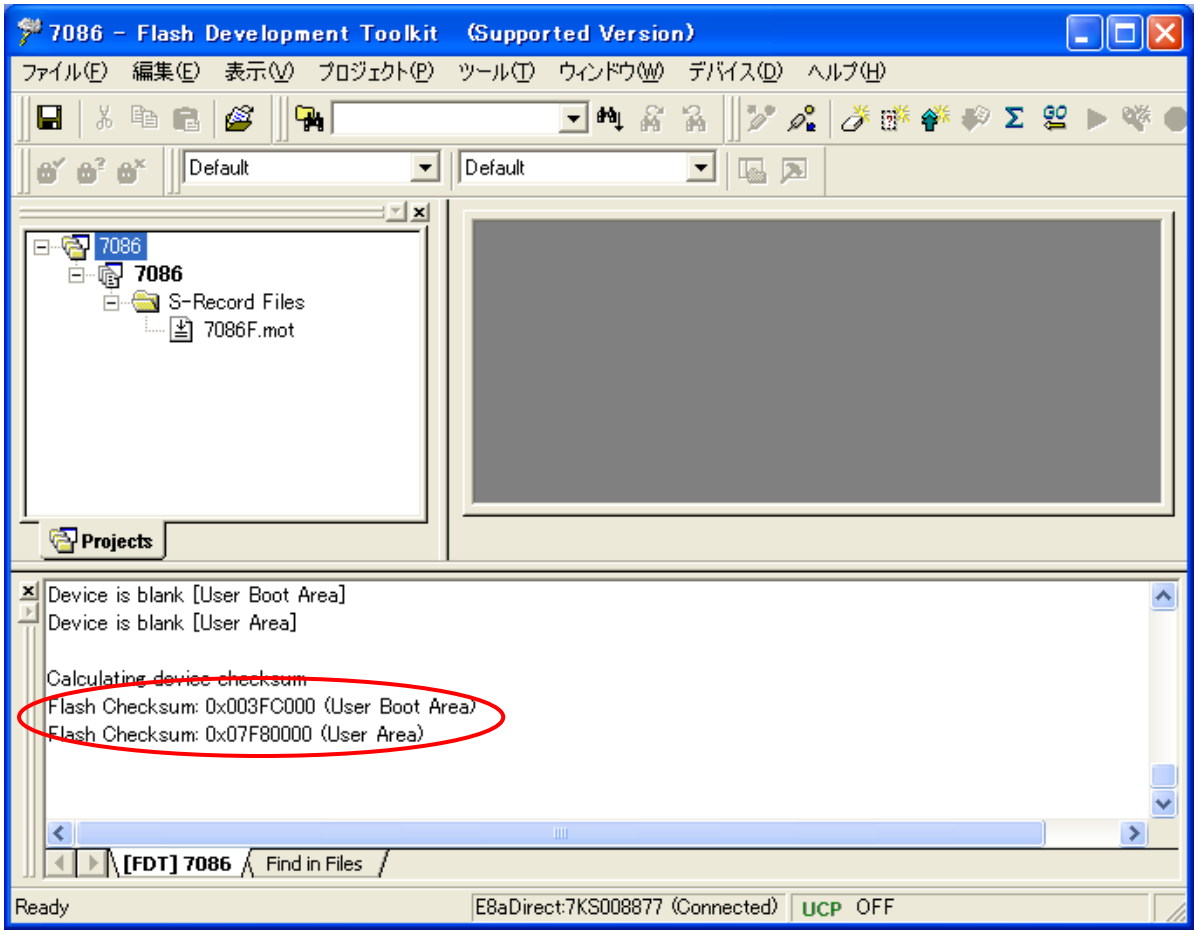
ブランクチェックの結果が表示されます。ユーザブートエリア、ユーザエリアに何も書き込まれていません。



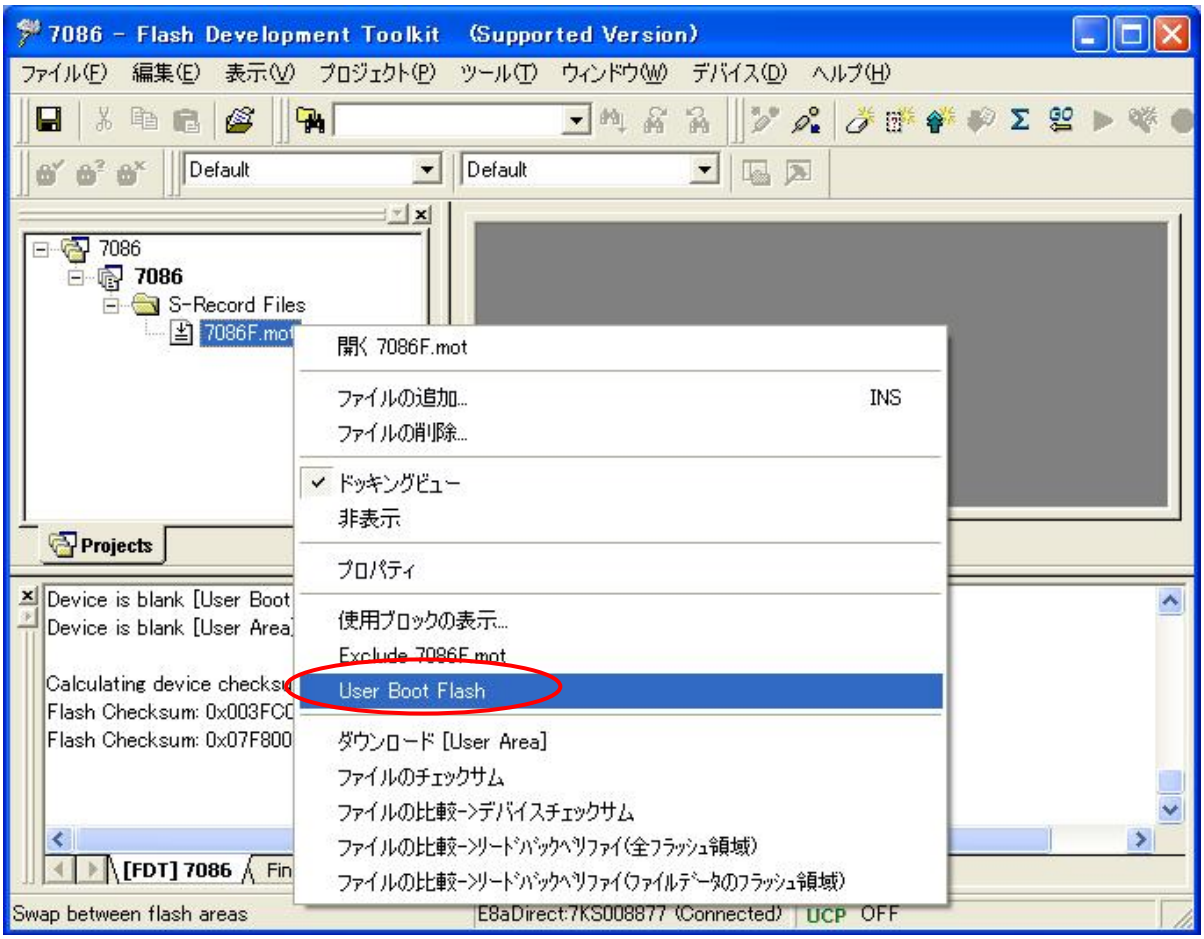
(7) ユーザブートエリア、ユーザエリアに何も書き込まれていないときのチェックサムの値を確認します。
[デバイス]メニューから[フラッシュのチェックサム]をクリックしてください。



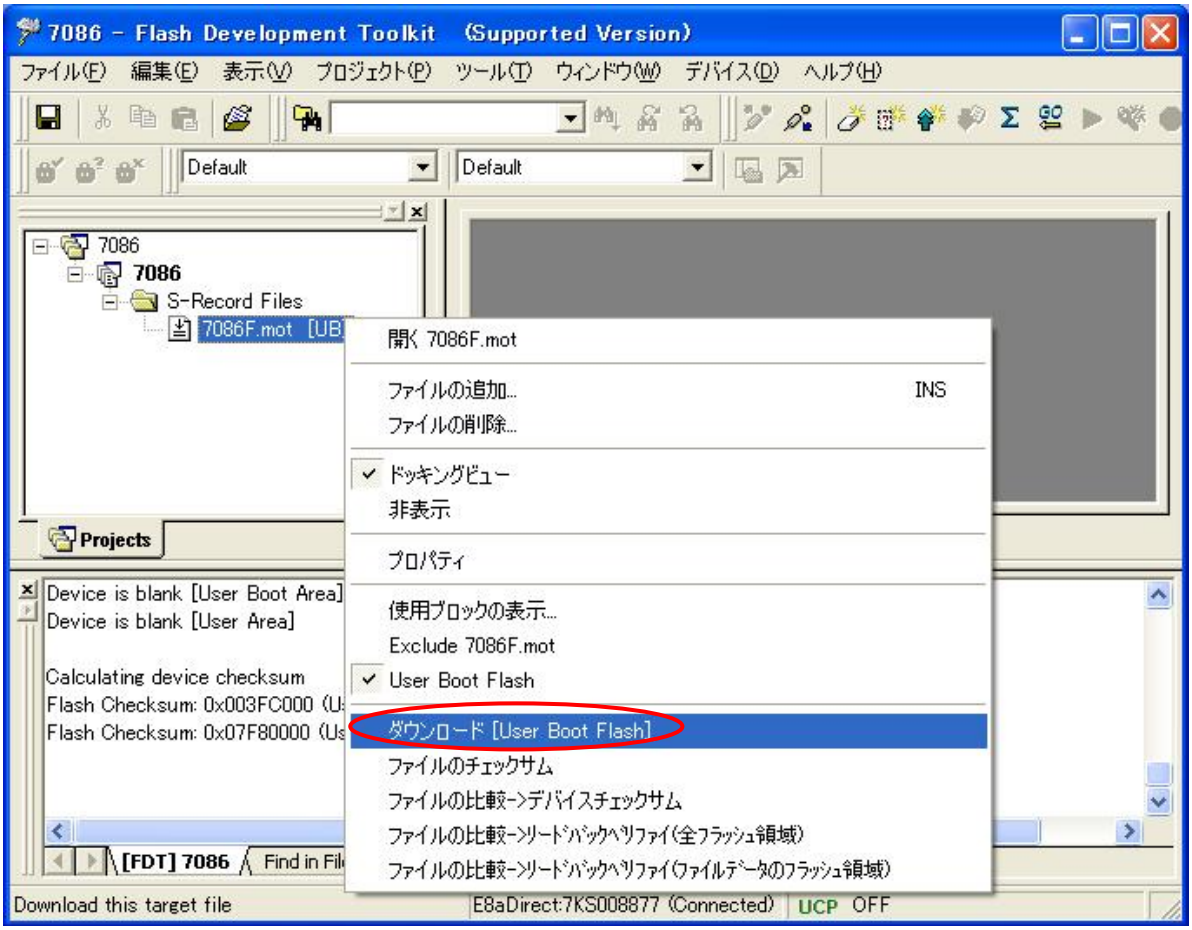
チェックサムの結果が表示されます。



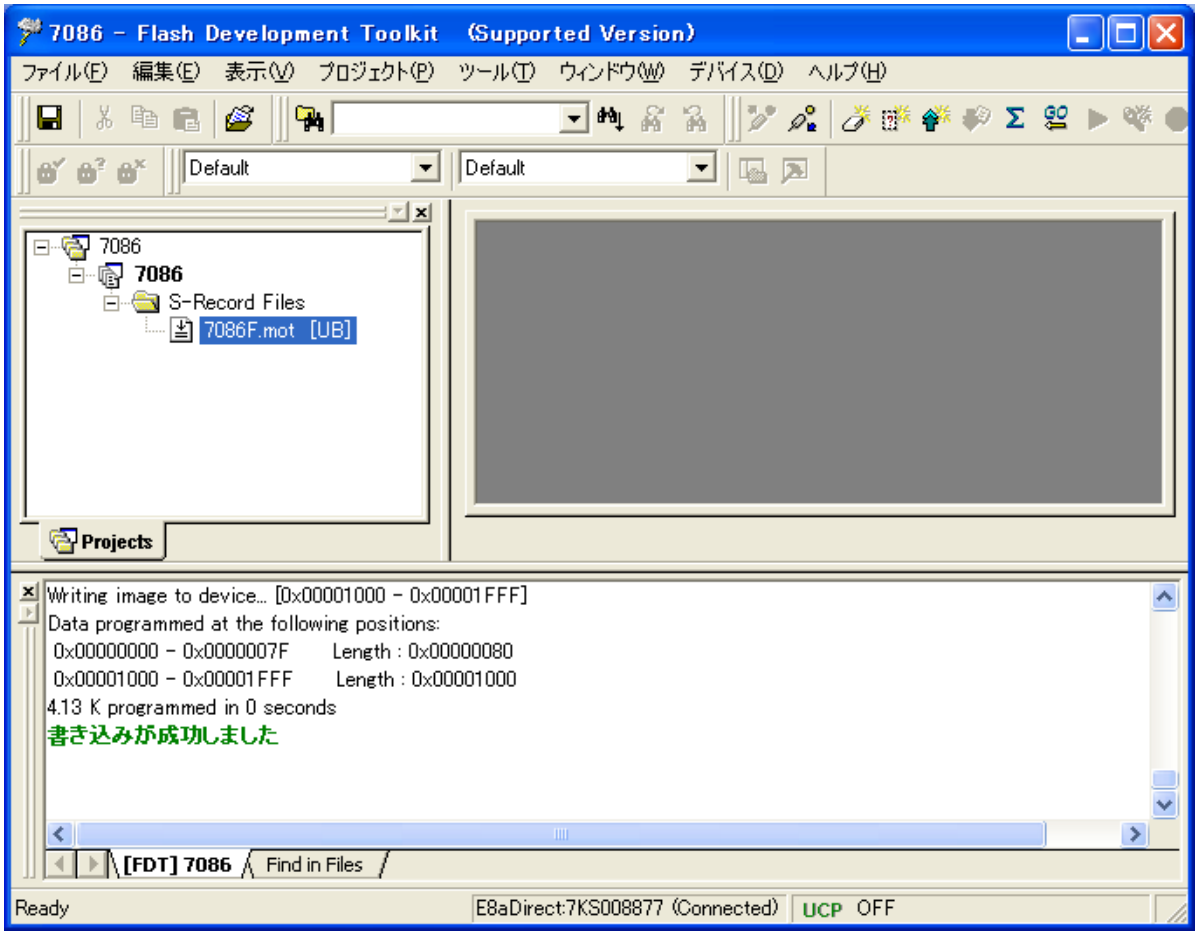
(8) ユーザブートエリアに書き込むため、ユーザブートエリアの設定を行います。
7086F.motファイルを右クリックし、ポップアップメニューを表示させ、[User Boot Flash]をクリックし、ダウンロード先をユーザブートエリアに設定します。



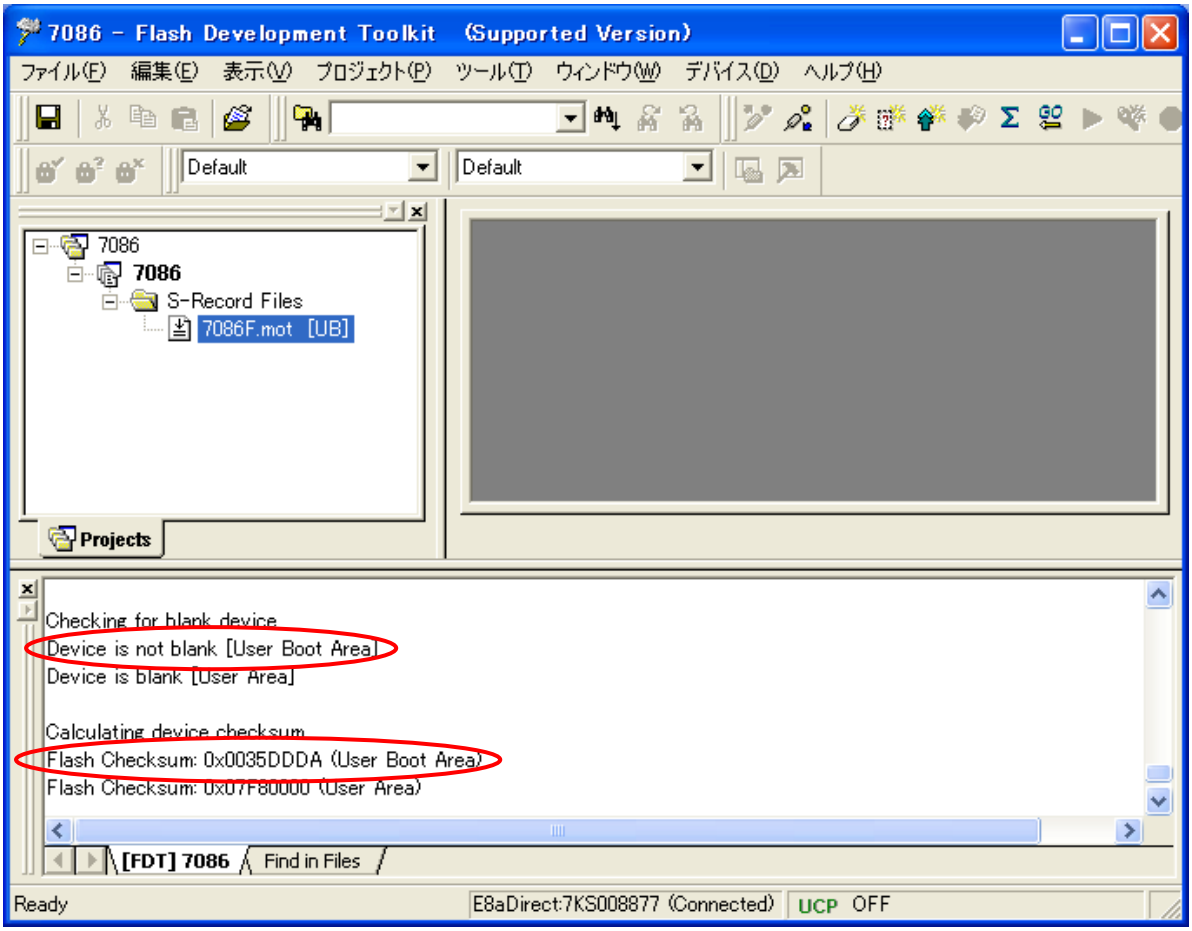
(9) 再度、7086F.motファイルを右クリックし、ポップアップメニューを表示させ、[ダウンロード[User Boot Flash]]をクリックし、7086F.motファイルをユーザブートエリアへダウンロードします。



ユーザブートエリアにプログラムがダウンロードされました。

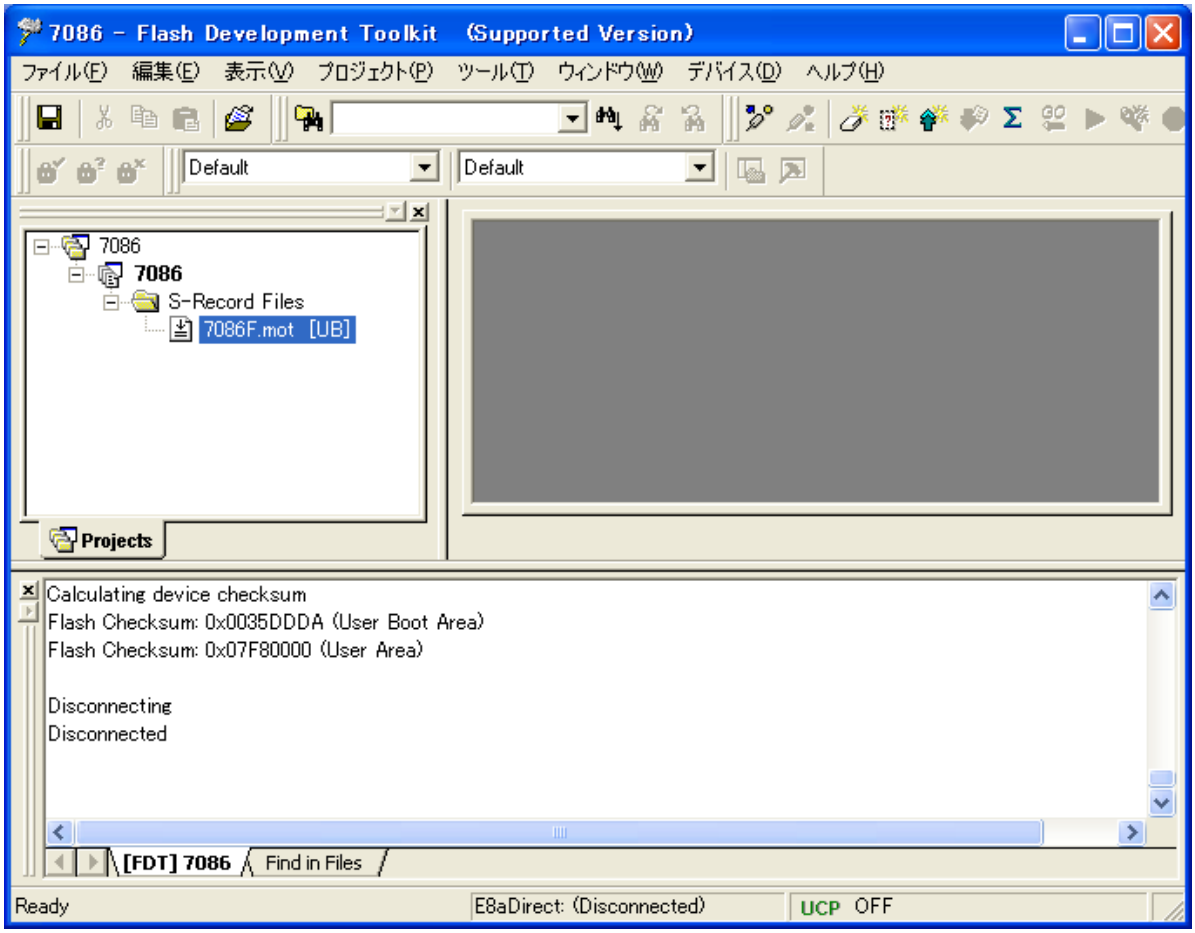


(10) 書き込まれたことを確認するために、ブランクチェックとチェックサムを行います
[デバイス]メニューから[ブランクチェック]をクリックしてください。
[デバイス]メニューから[フラッシュのチェックサム]をクリックしてください。
ブランクチェックとチェックサムの結果が表示されます。

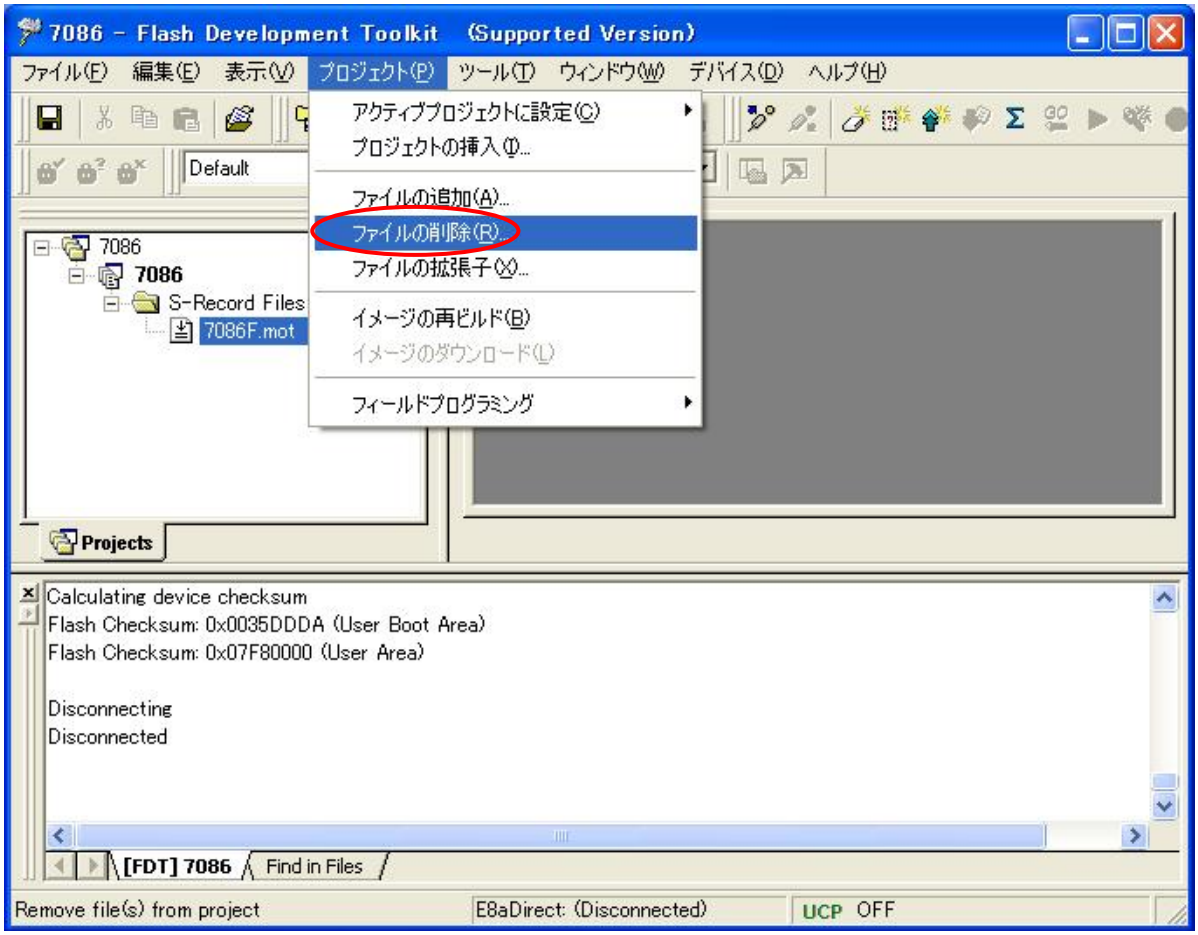


ユーザブートエリアにプログラムが書き込まれたことがわかります。

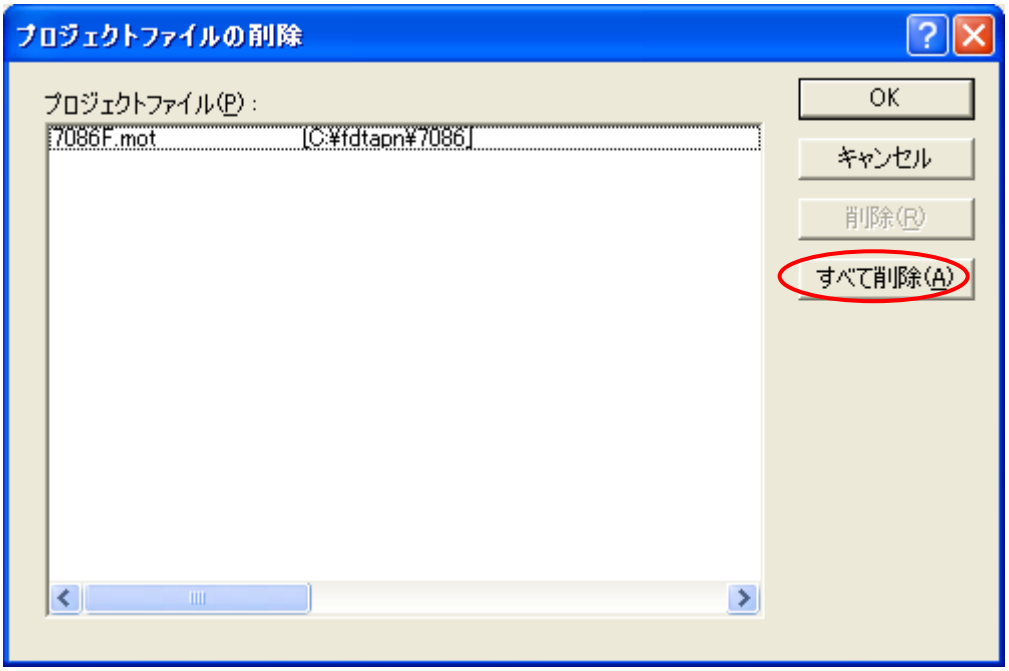
(11) [デバイス]メニューから[デバイスとの切断]をクリックし、デバイスを切断してください。



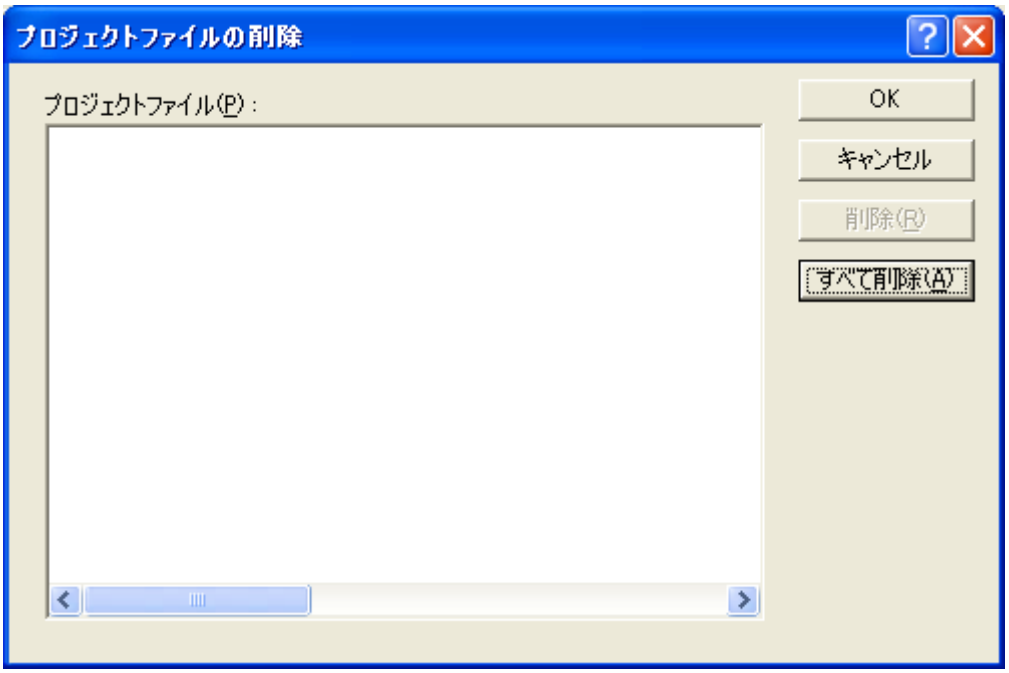
(12) ファイルを削除します。
[プロジェクト]メニューから[ファイルの削除]をクリックしてください。



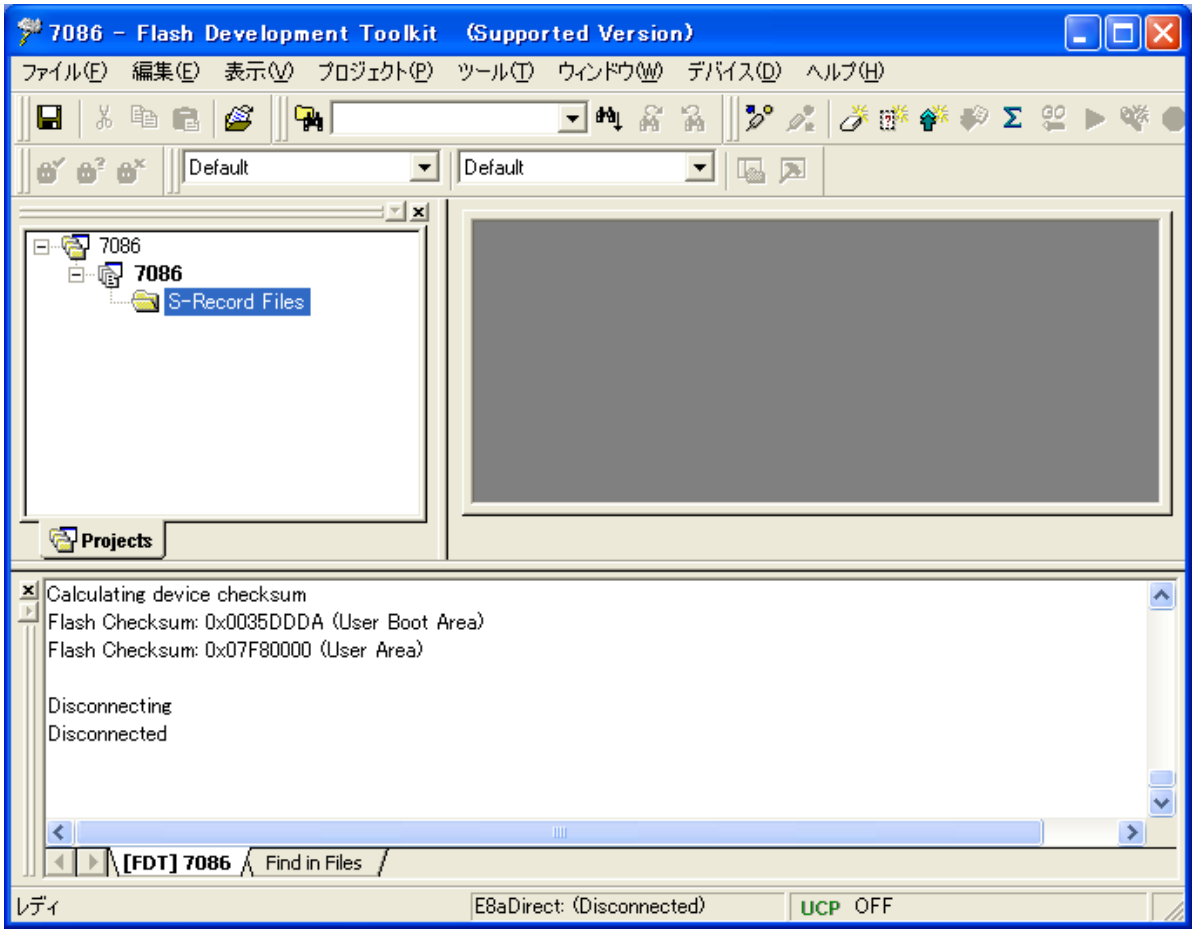
(13) [プロジェクトファイルの削除]ダイアログボックスが表示されます。[すべて削除]をクリックしてください。



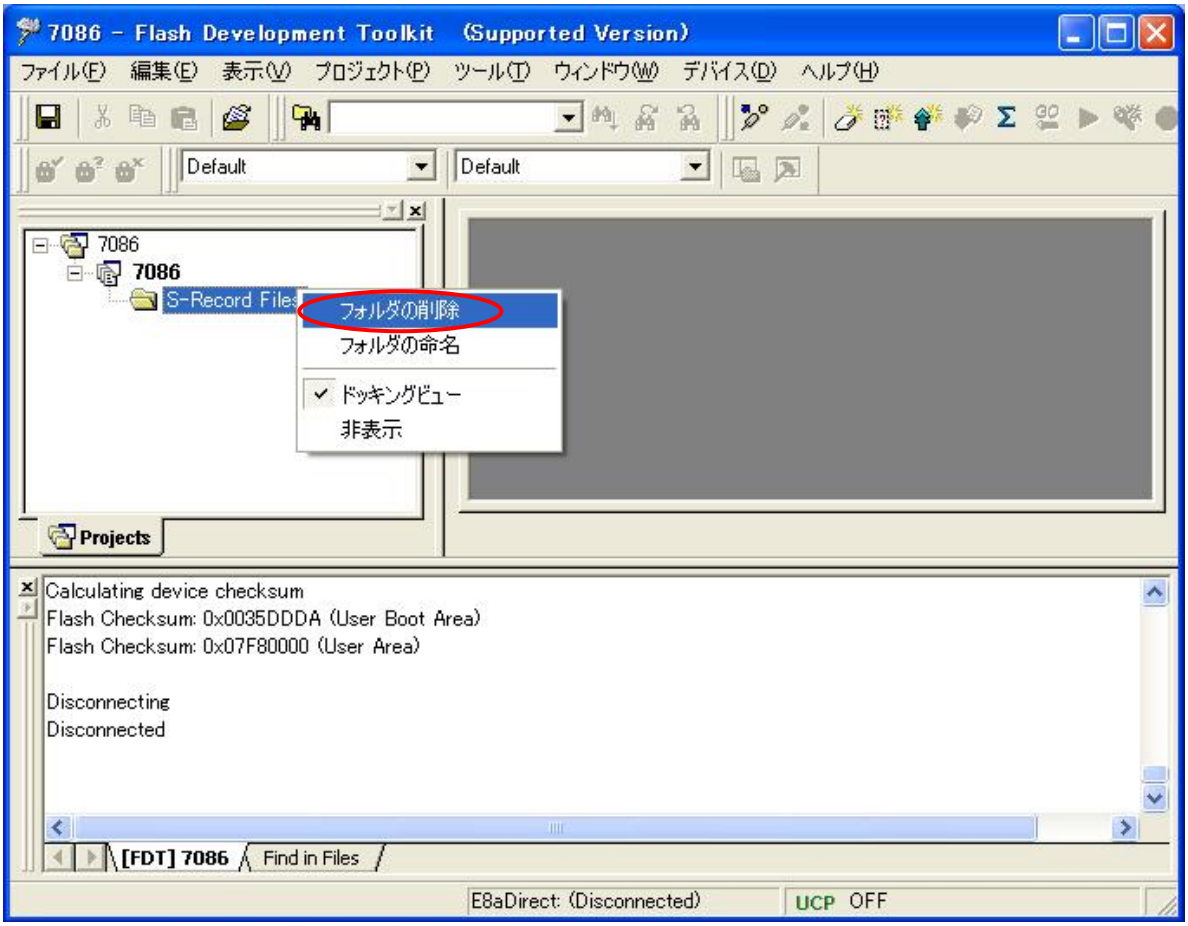
[OK]をクリックしてください。



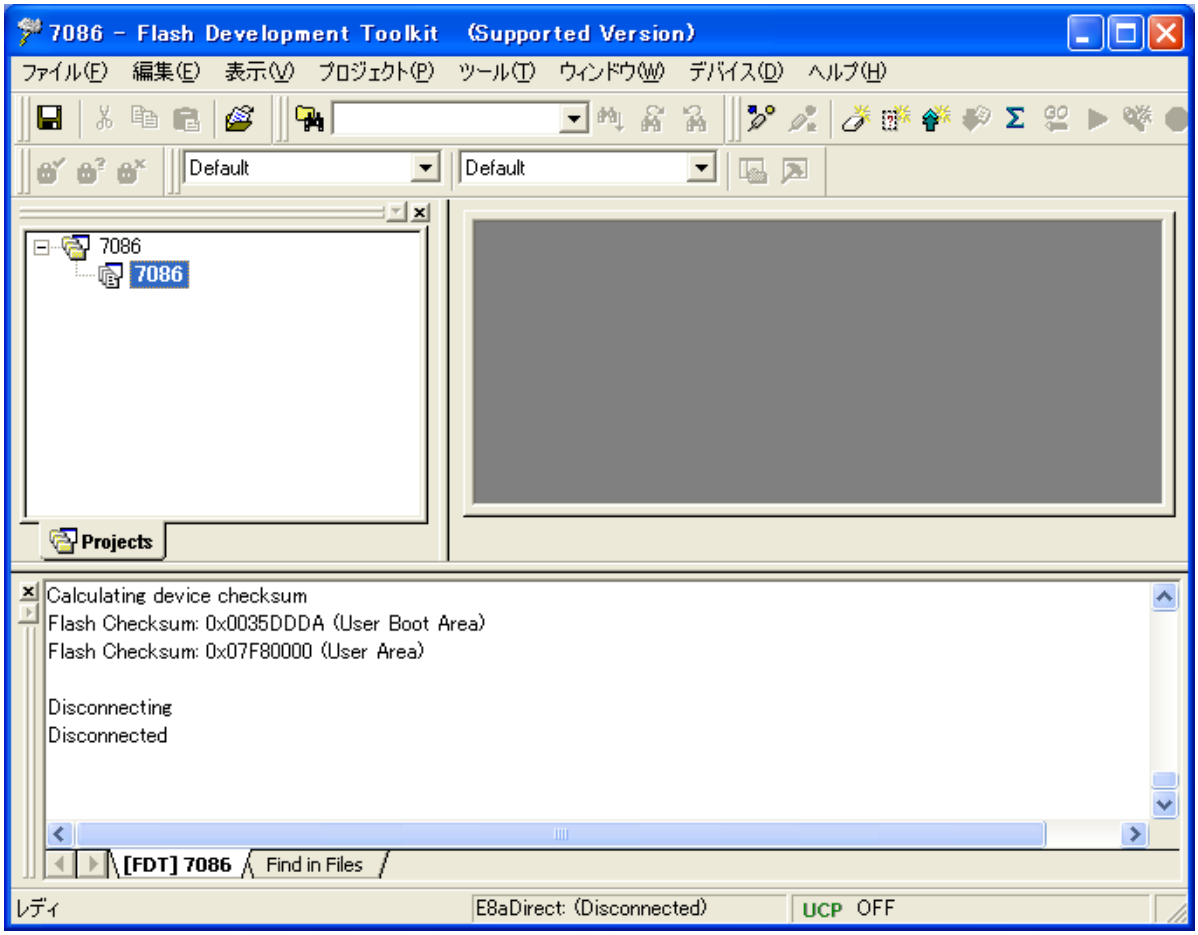
ファイルが削除されます。



(14) フォルダを削除します。
フォルダを右クリックしてポップアップメニューを開き、[フォルダの削除]をクリックしてください。



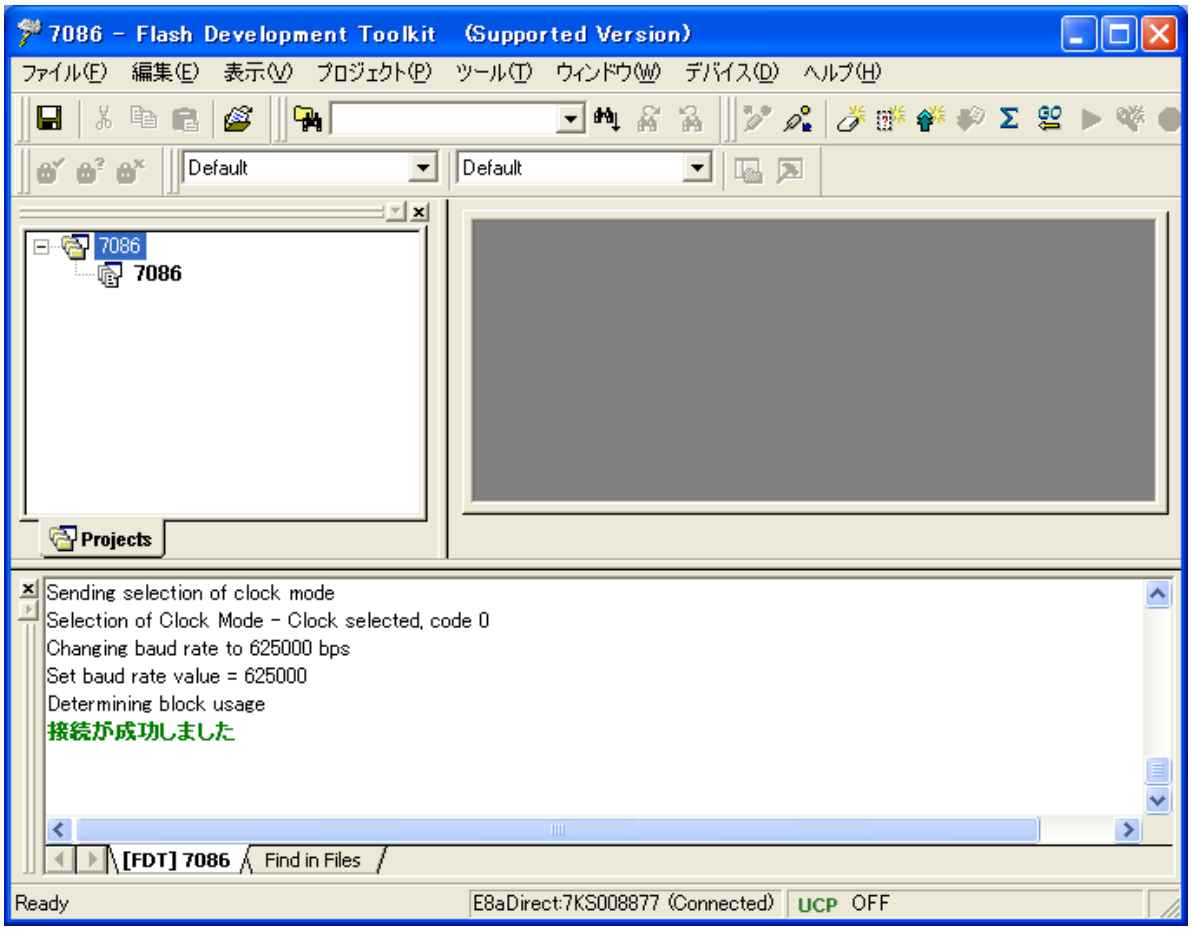
フォルダが削除されます。



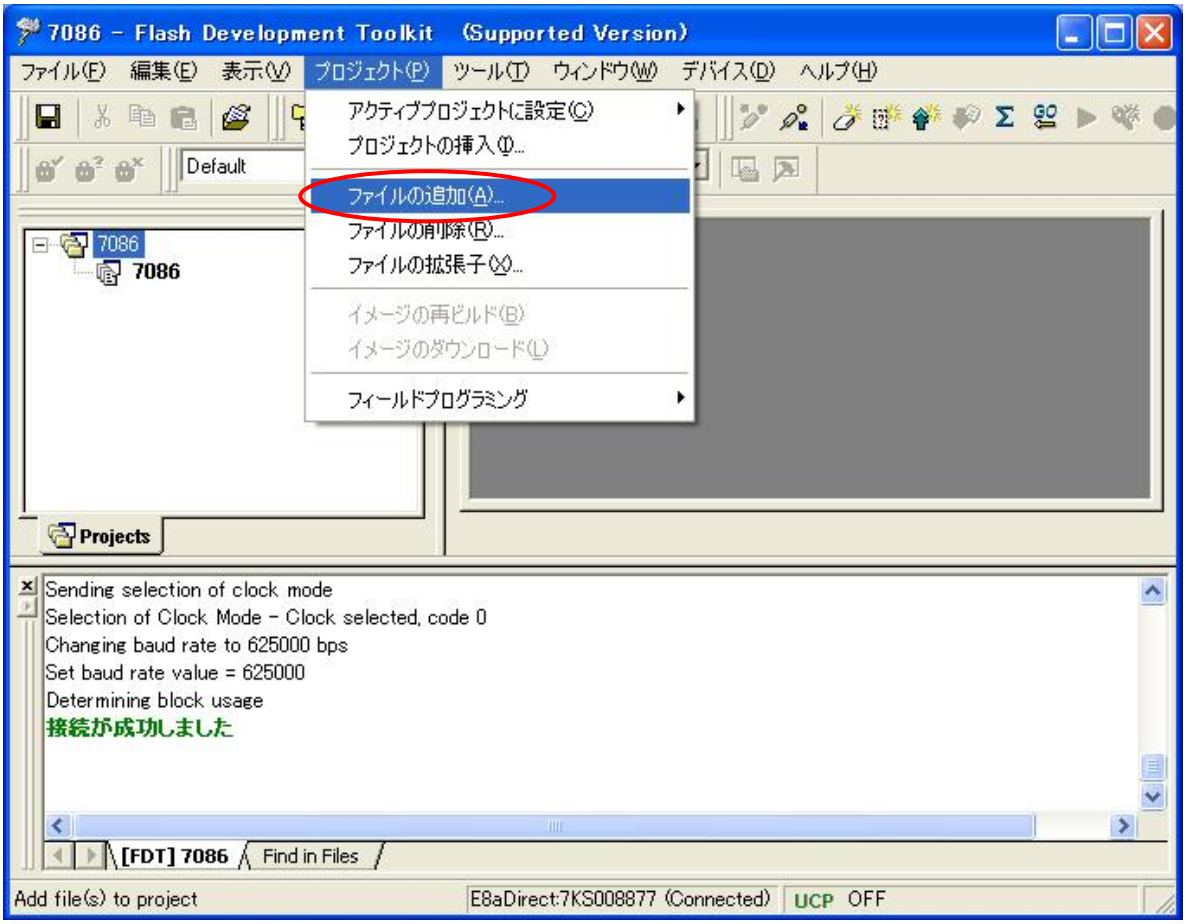
3.4 ブートモード 2(ユーザエリアへの書き込み)

ブートモードでユーザエリアへユーザプログラムモード用ロードモジュールファイルを書き込みます。
書き込むプログラムは、3.3 ブートモード1(ユーザブートエリアへの書き込み)と同じものです。

(1) [Flash Development Toolkit 4.01]を起動し、プロジェクトワークスペースファイル 7086.AWS を開き、デバイスと接続してください。

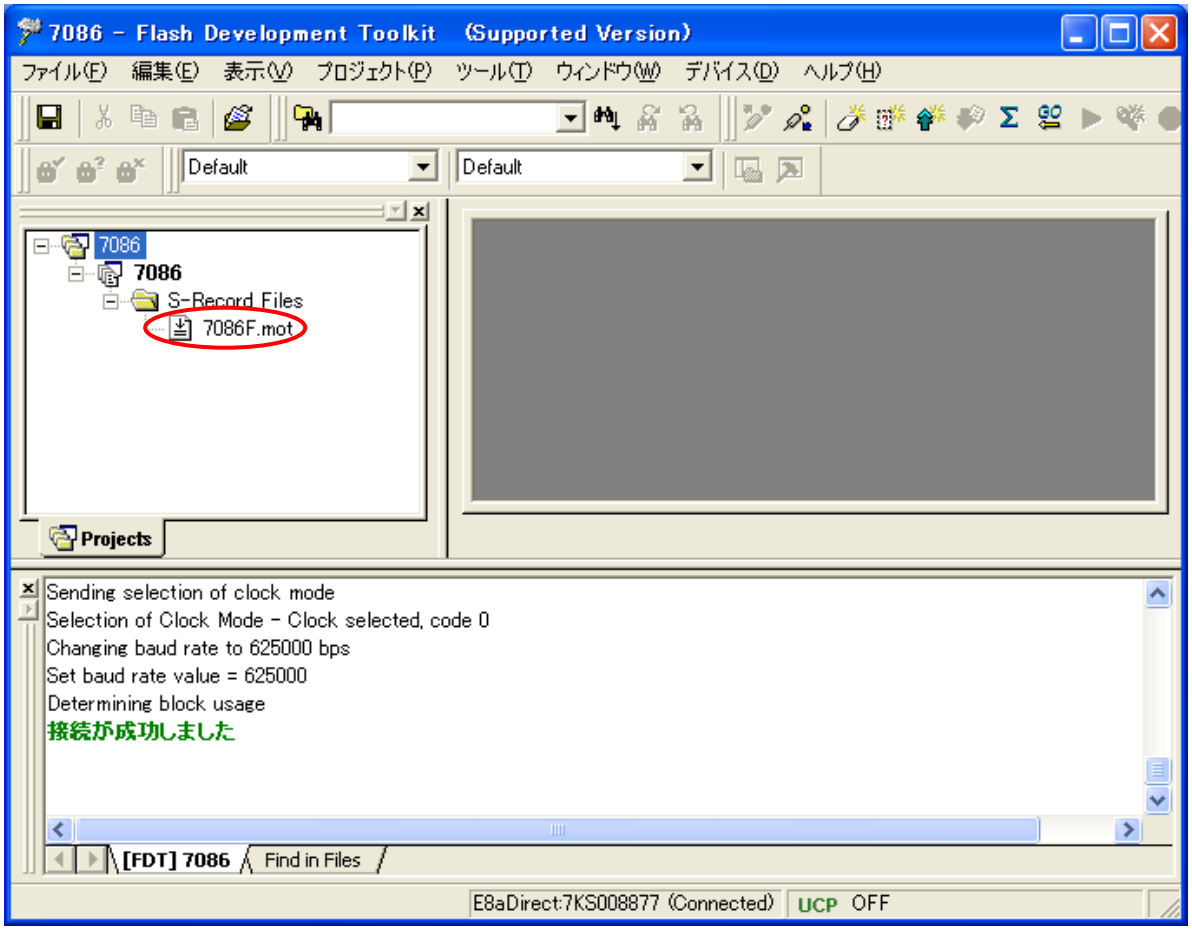


(2) 書き込みファイルを選択します。
[プロジェクト]メニューから[ファイルの追加]をクリックしてください。

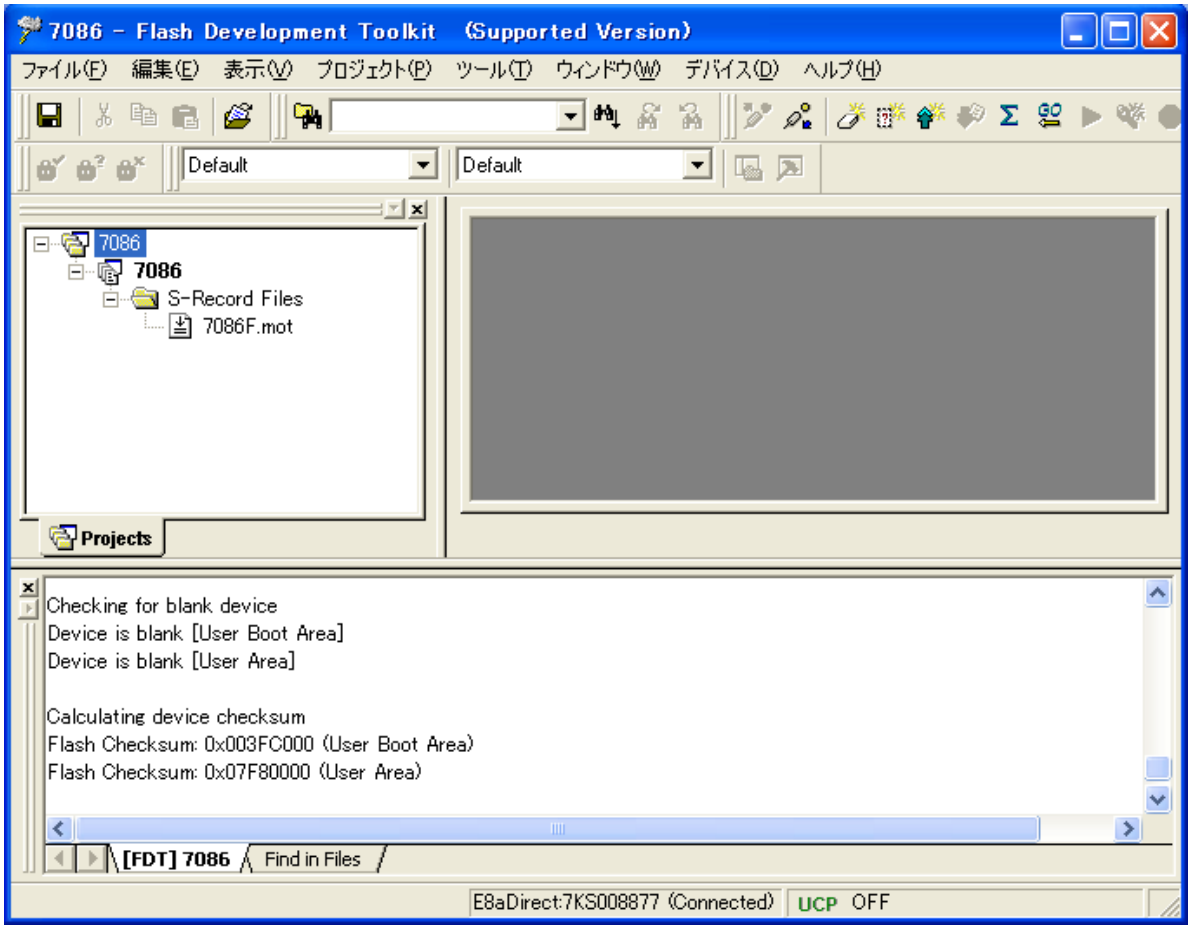


[ファイルの追加]ダイアログボックスから[7086F.mot]ファイルを選択して[Add]をクリックしてください。

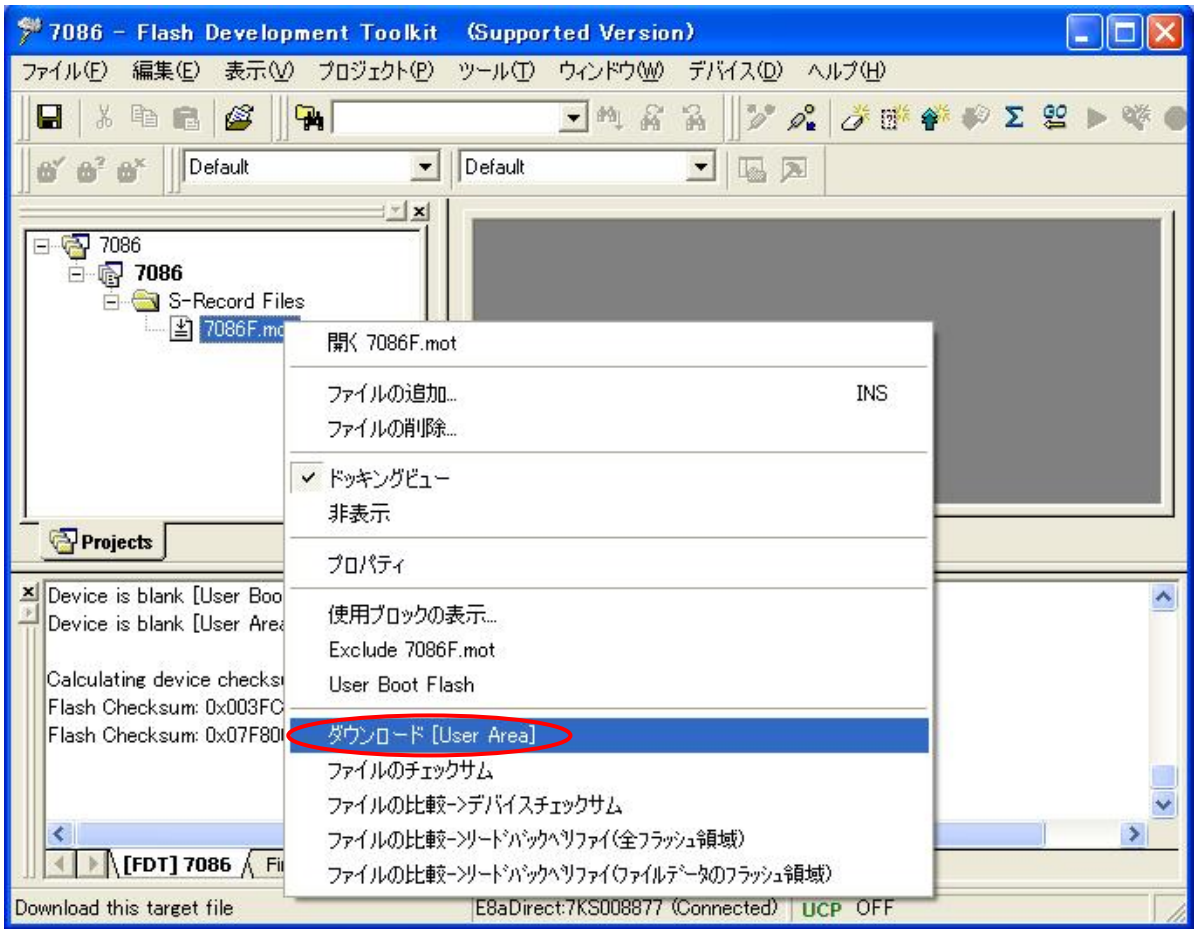
プロジェクトに 7086F.mot ファイルが追加されます。



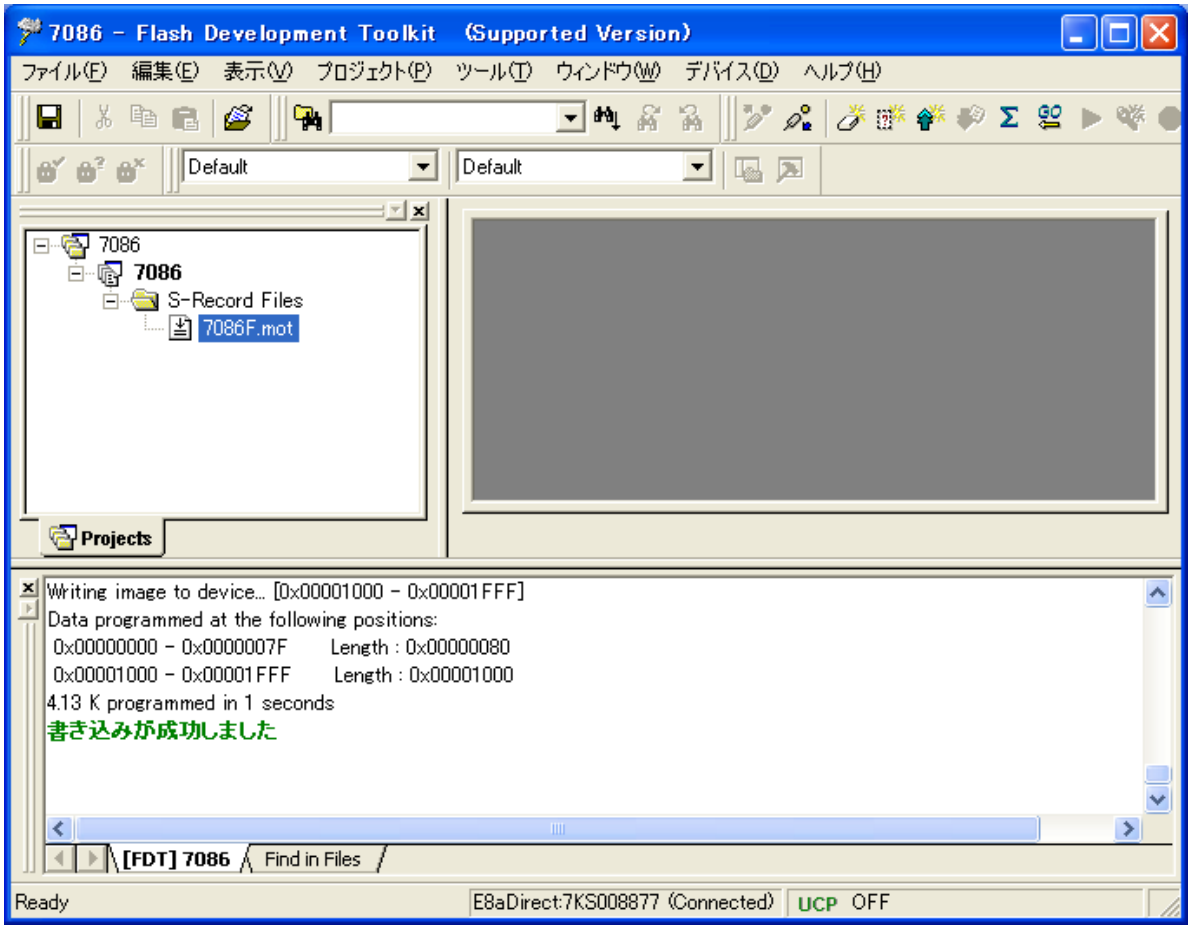
(3) 7086F.mot ファイルを書き込む前に、ブランクチェックとチェックサムを行います。
[デバイス]メニューから[ブランクチェック]をクリックしてください。
[デバイス]メニューから[フラッシュのチェックサム]をクリックしてください。
ブランクチェックとチェックサムの結果が表示されます。



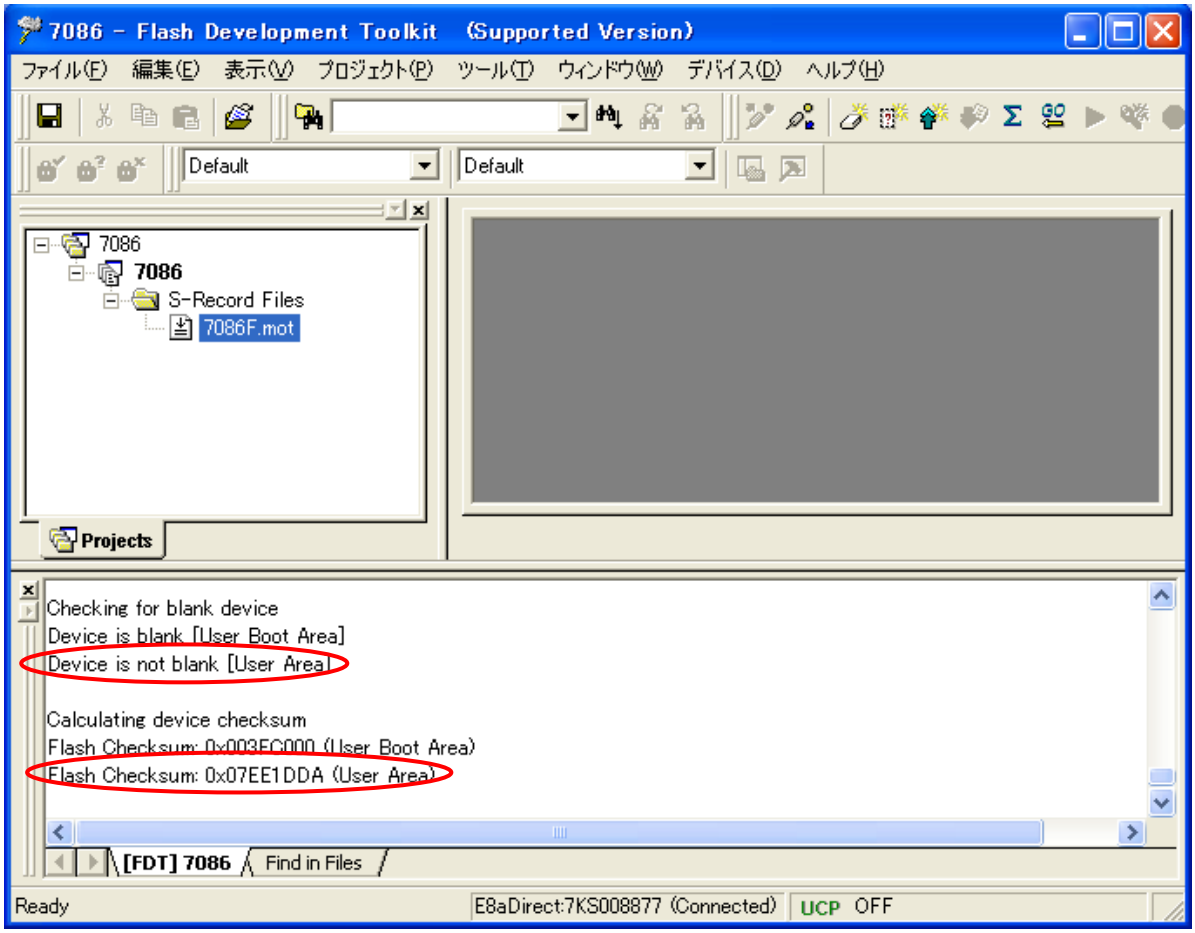
(4) 7086F.mot ファイルを右クリックし、ポップアップメニューを表示させ、[ダウンロード[User Area]]をクリックし、7086F.mot ファイルをユーザエリアへダウンロードします。
デフォルトは[ダウンロード[User Area]]です。



ユーザエリアにプログラムがダウンロードされました。



(5) 書き込まれたことを確認するために、ブランクチェックとチェックサムを行います。
[デバイス]メニューから[ブランクチェック]をクリックしてください。
[デバイス]メニューから[フラッシュのチェックサム]をクリックしてください。
ブランクチェックとチェックサムの結果が表示されます。

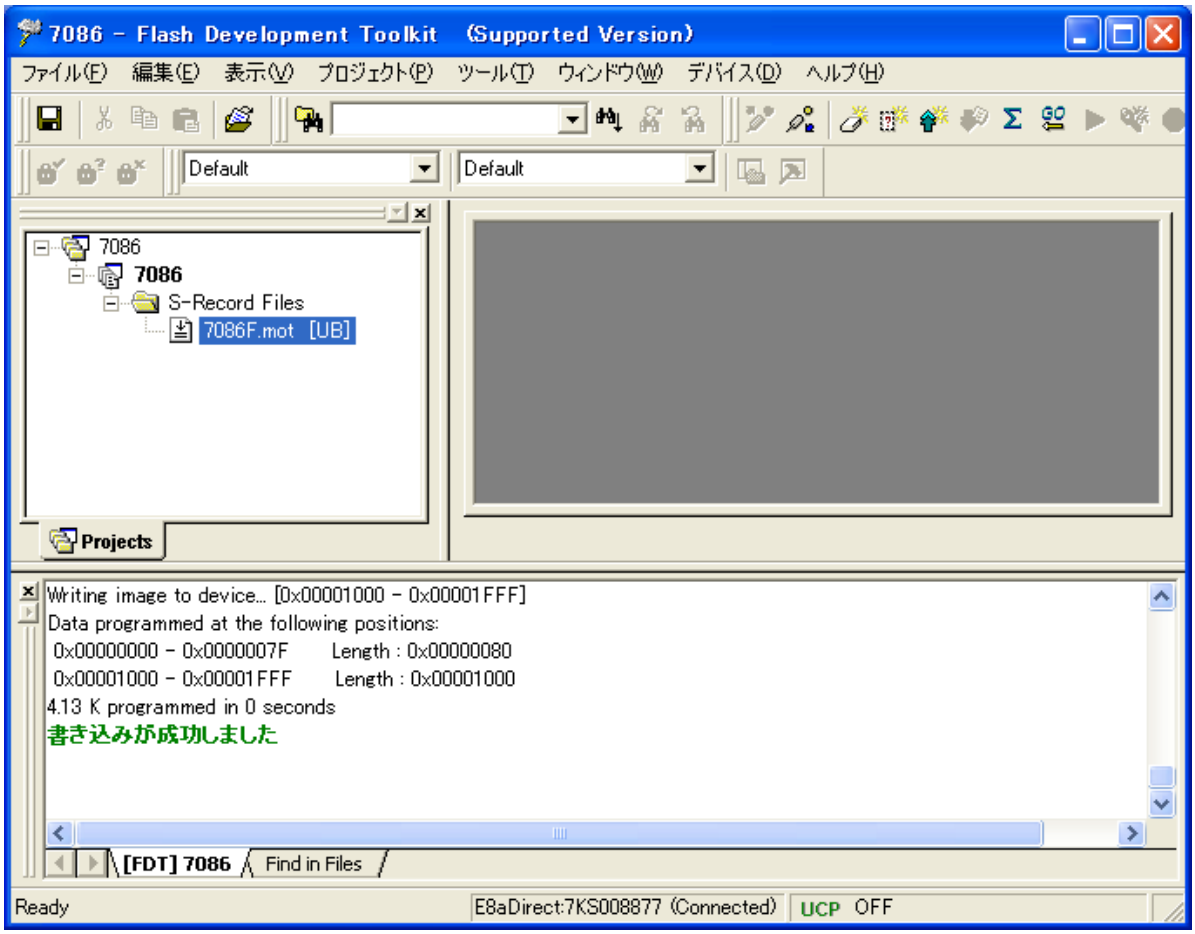


ユーザエリアにプログラムが書き込まれたことがわかります。

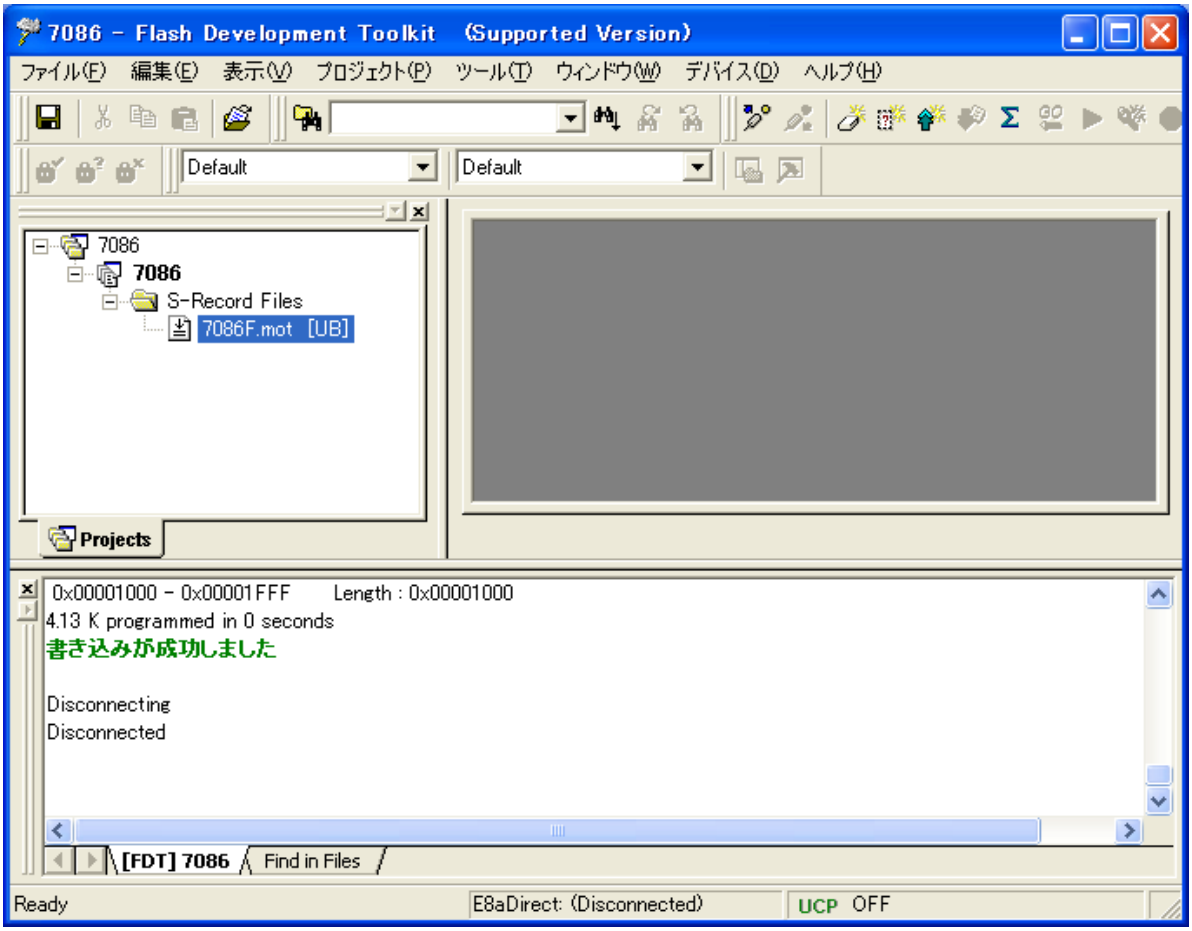
3.5 ユーザブートモード

ユーザブートモードは、ユーザエリアの書き込み消去ができます。ユーザブートエリアへの書き込み消去はできません。予めユーザブートエリアにユーザプログラムモード用ロードモジュールファイルが書き込まれている必要があります。

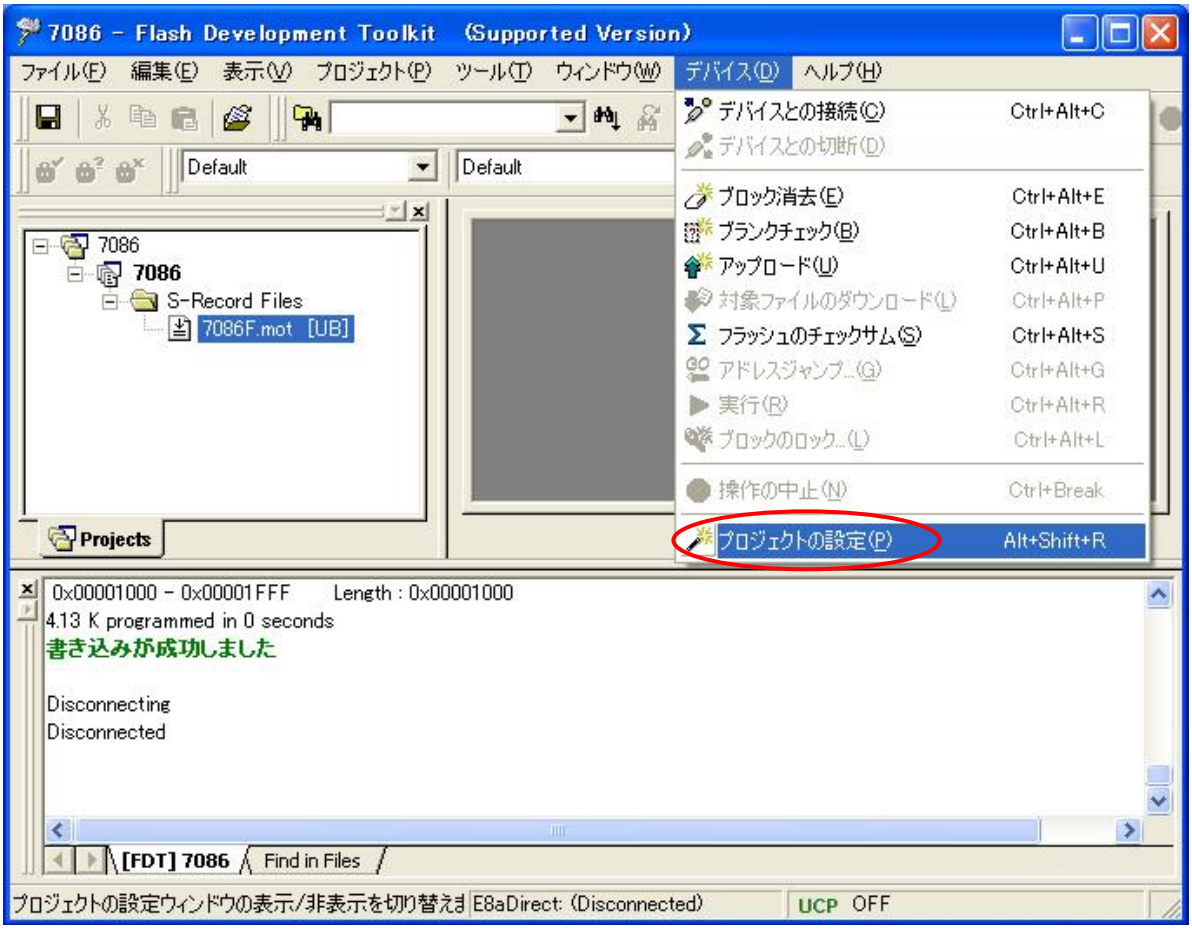
(1) [Flash Development Toolkit 4.01]を起動し、プロジェクトワークスペースファイル7086.AWSを開き、ブートモードで 7086F.mot ファイルをユーザブートエリアへ書き込んでください。



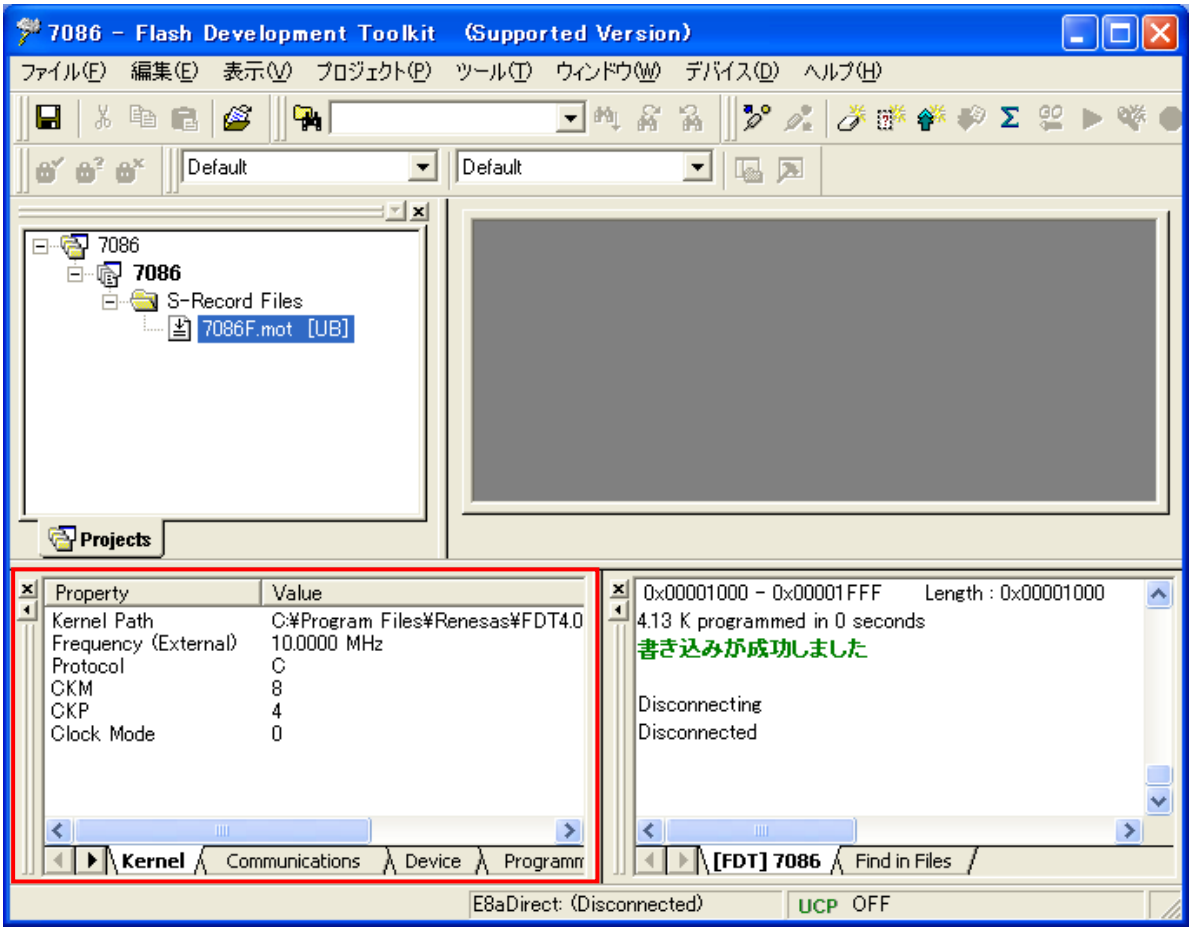
(2) [デバイス]メニューから[デバイスとの切断]をクリックし、デバイスを切断してください。



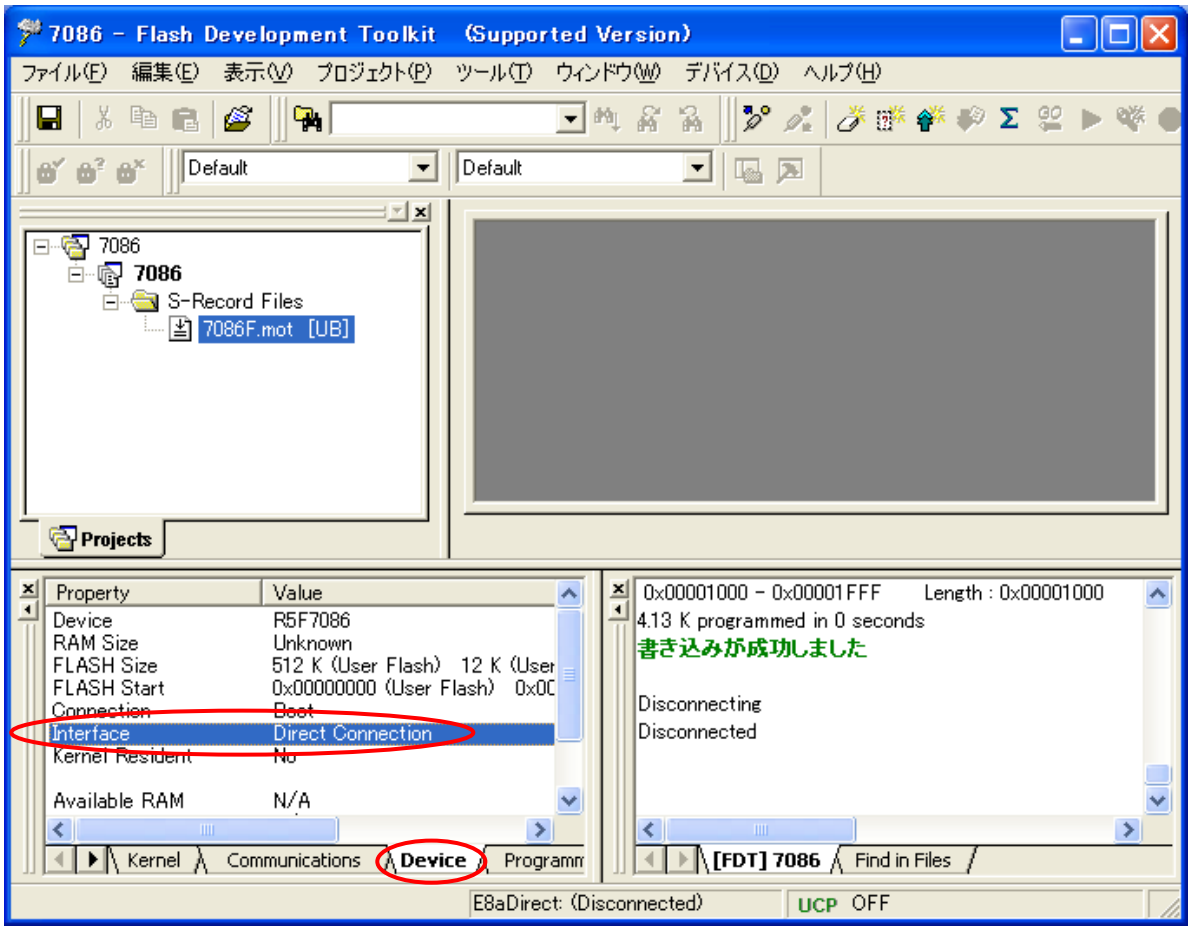
(3) [デバイス]メニューから[プロジェクトの設定]をクリックしてください。



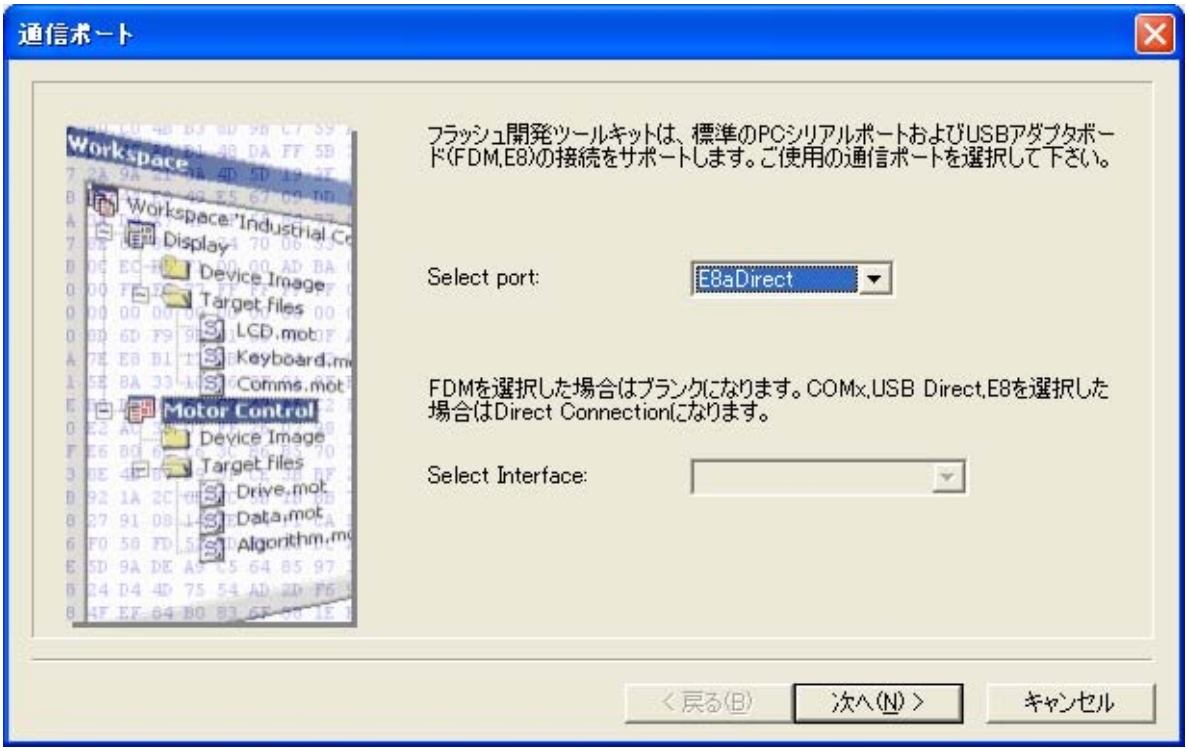
プロジェクト設定ウィンドウが表示されます。



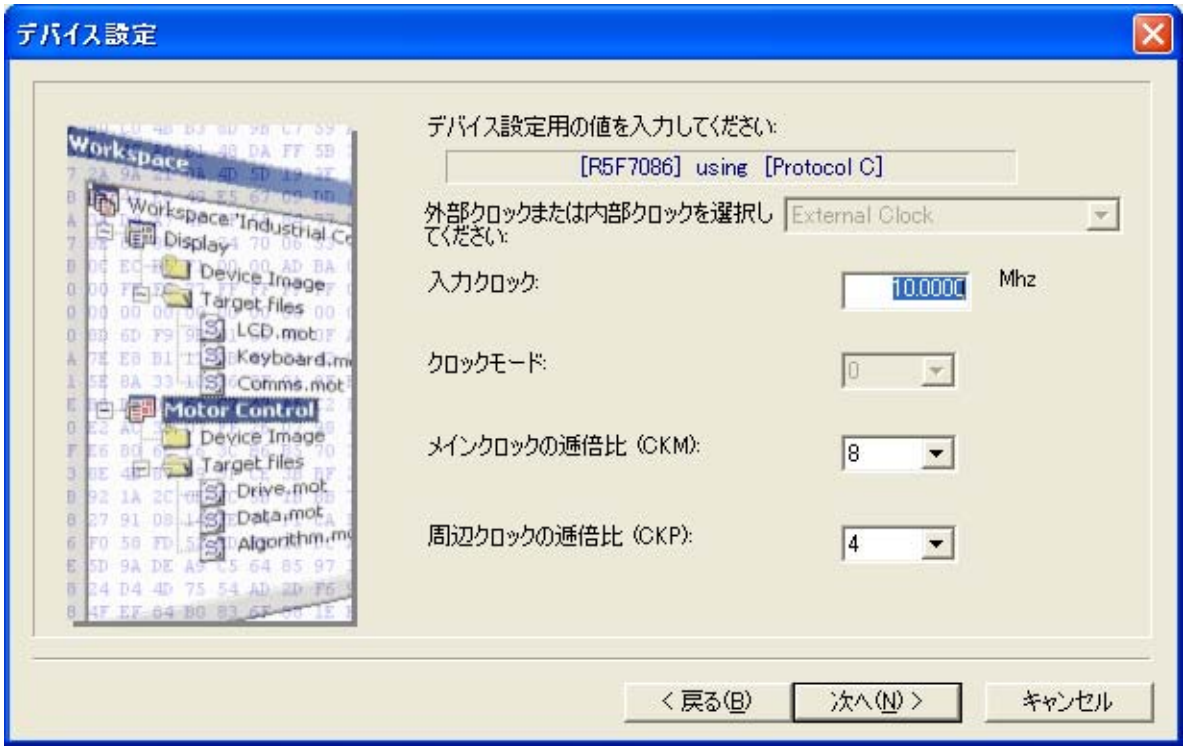
(4) ユーザブートモードを設定します。
プロジェクト設定ウィンドウの[Device]タブを選択し、[Interface] [Direct Connection]をダブルクリックしてください。



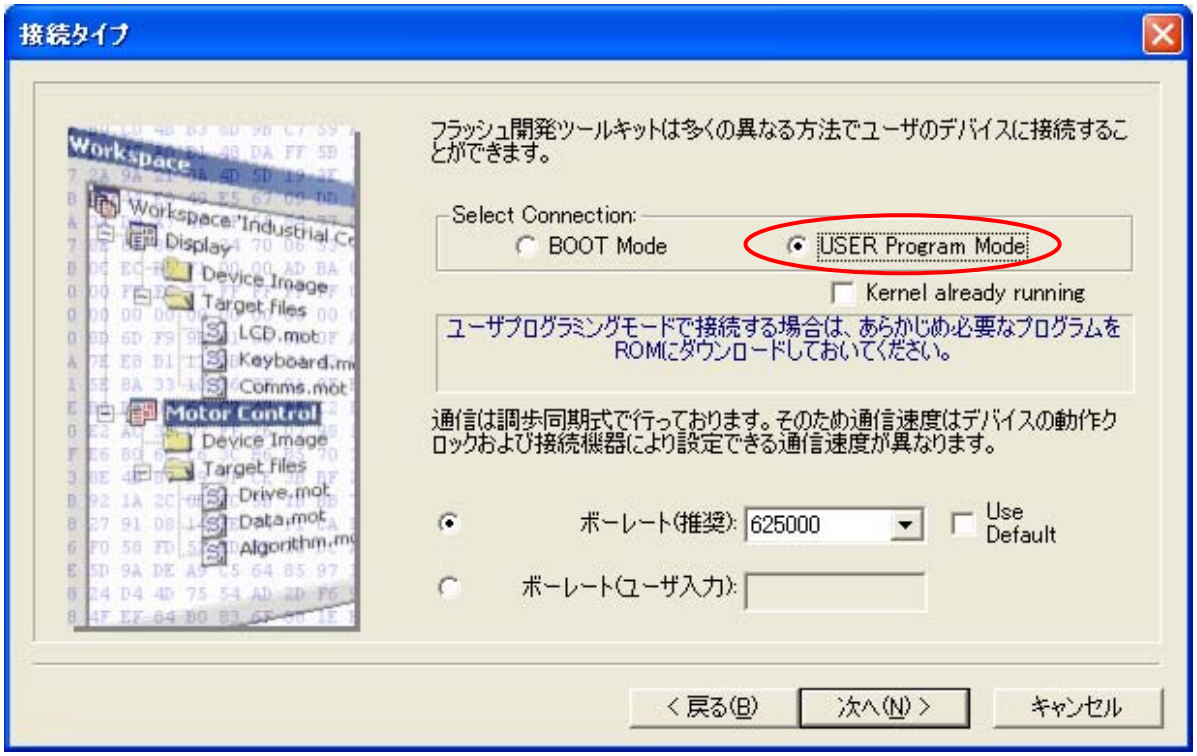
[次へ]をクリックしてください。



[次へ]をクリックしてください。

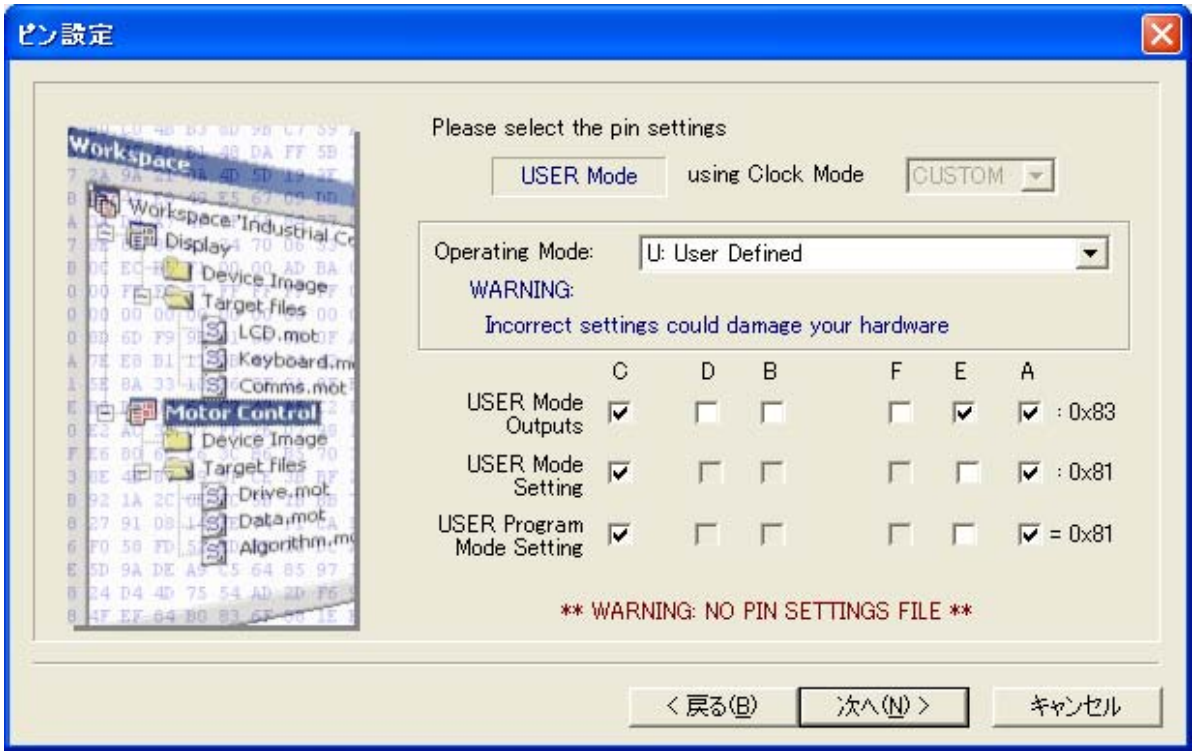


接続選択の[USER Program Mode]を選択し、[次へ]をクリックしてください。



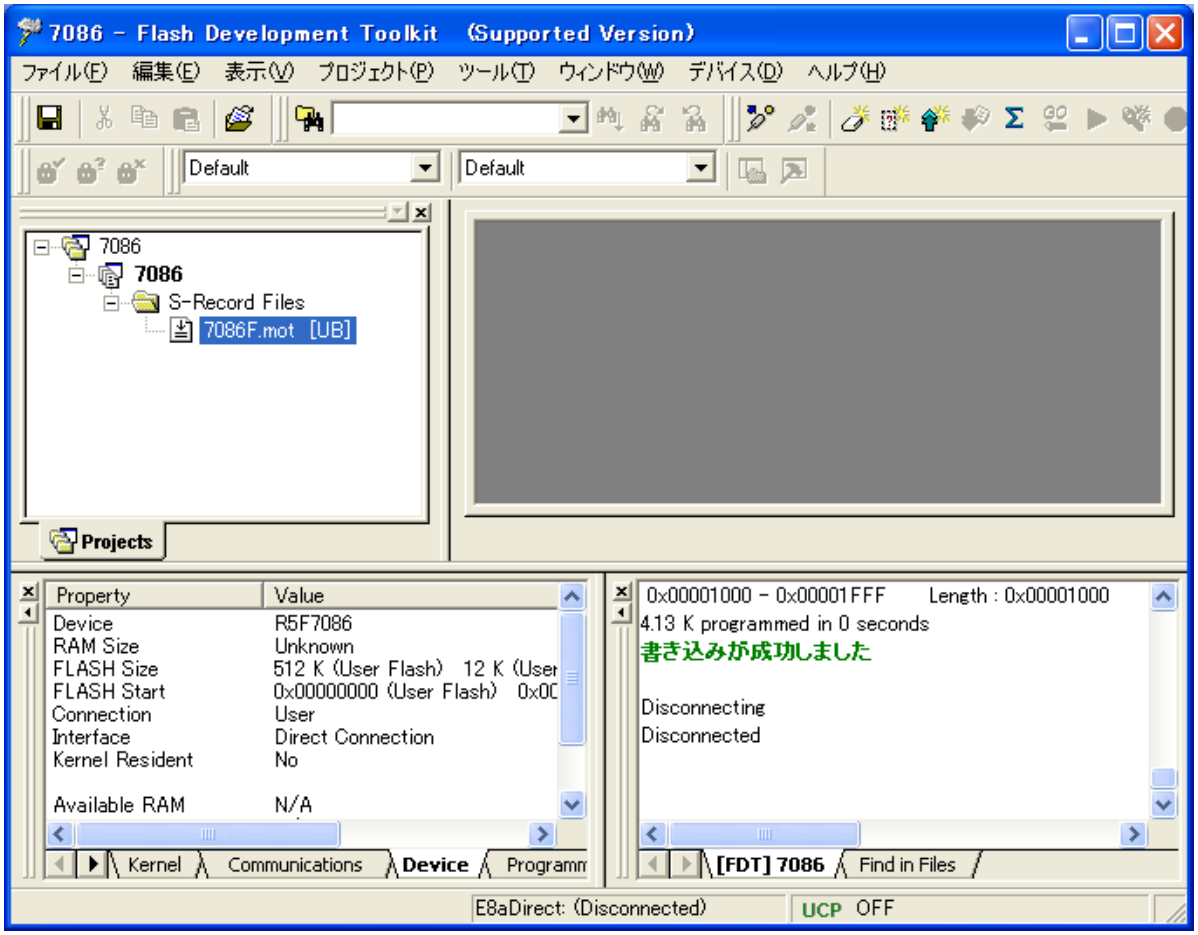
ユーザブートモードの E8a ピン設定をします。
C ピン(FWE) 、A ピン(MD0)を 1 に、E ピン(MD1)を 0 に設定して[次へ]をクリックしてください。

MCU 動作モード	端子設定			モード名
	FWE	MD1	MD0	
モード 5	1	0	1	ユーザブートモード

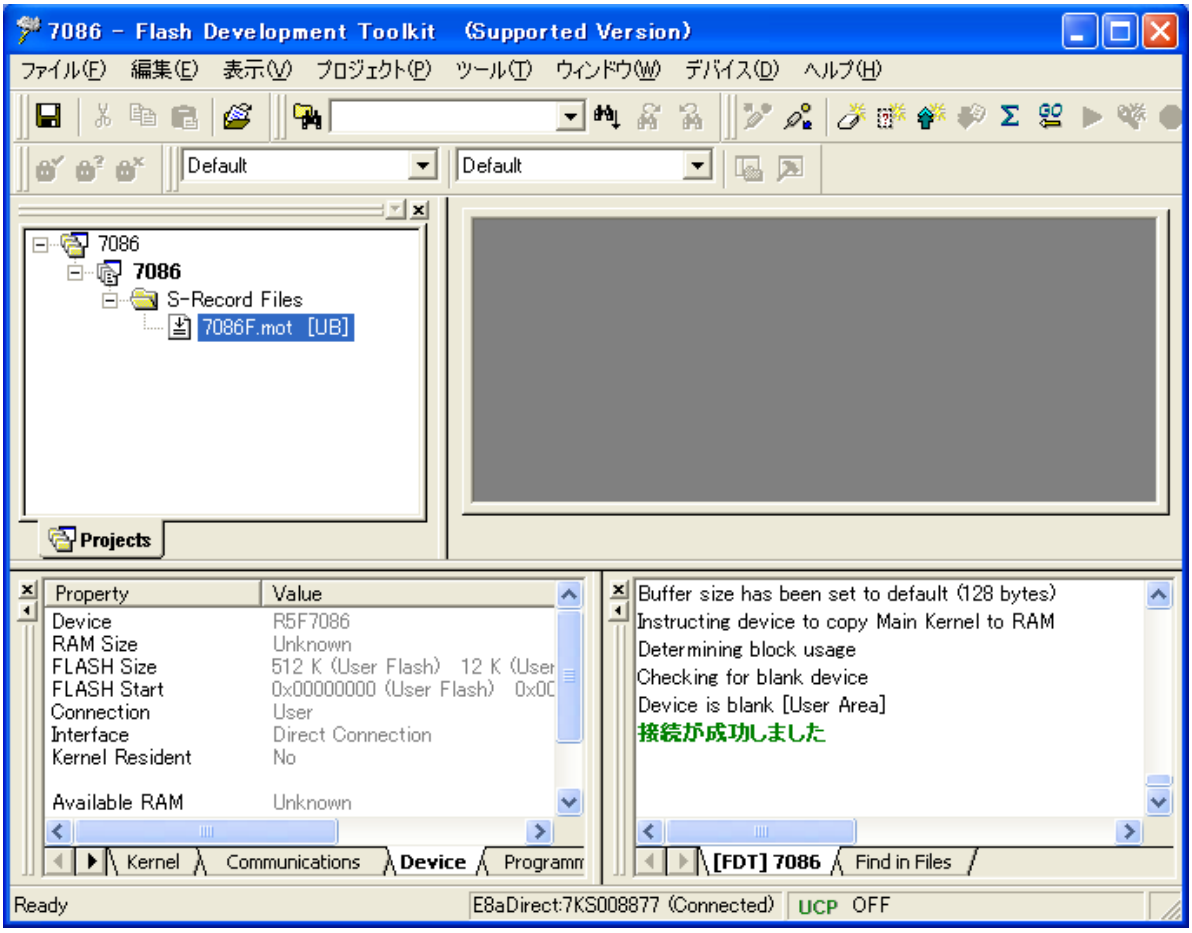


[完了]をクリックしてください。

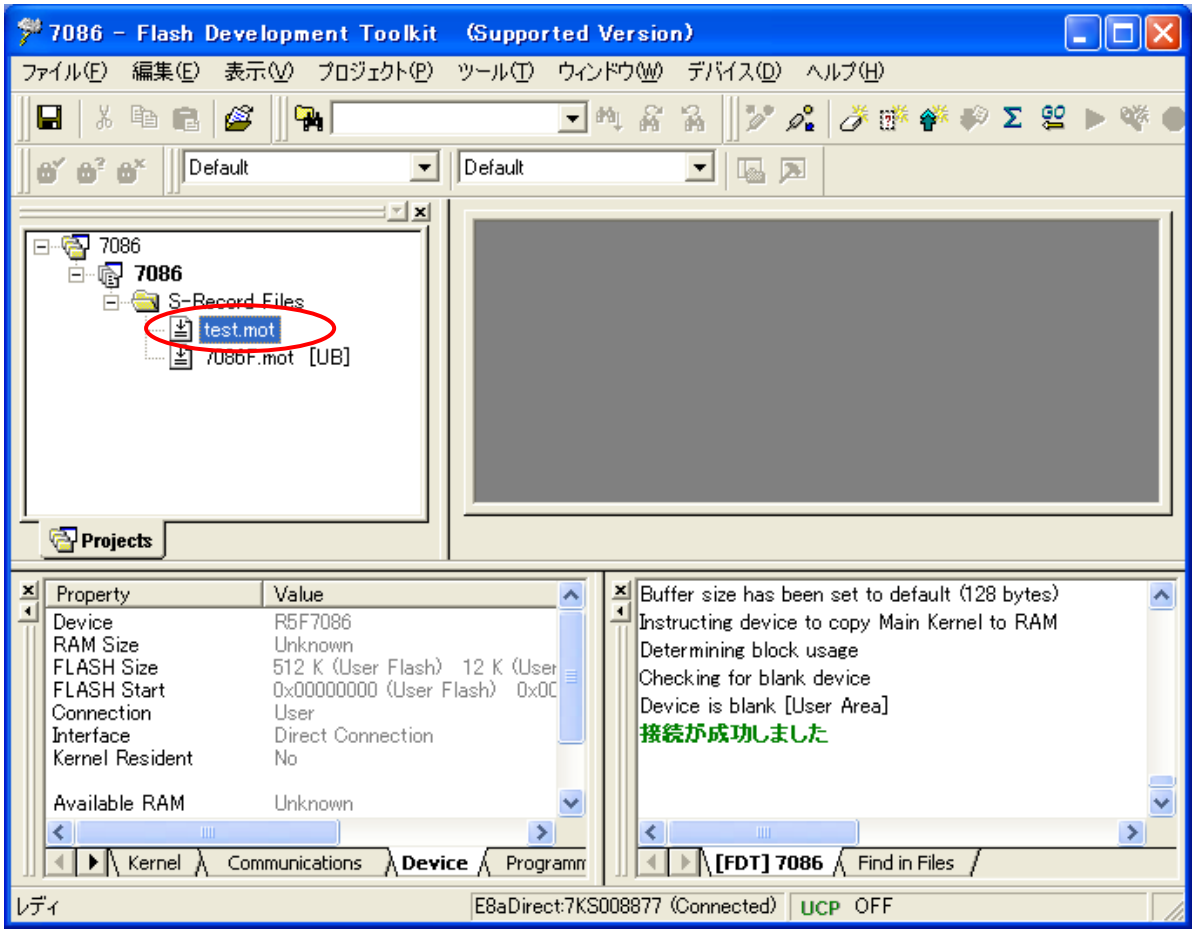
ユーザブートモードを設定しました。



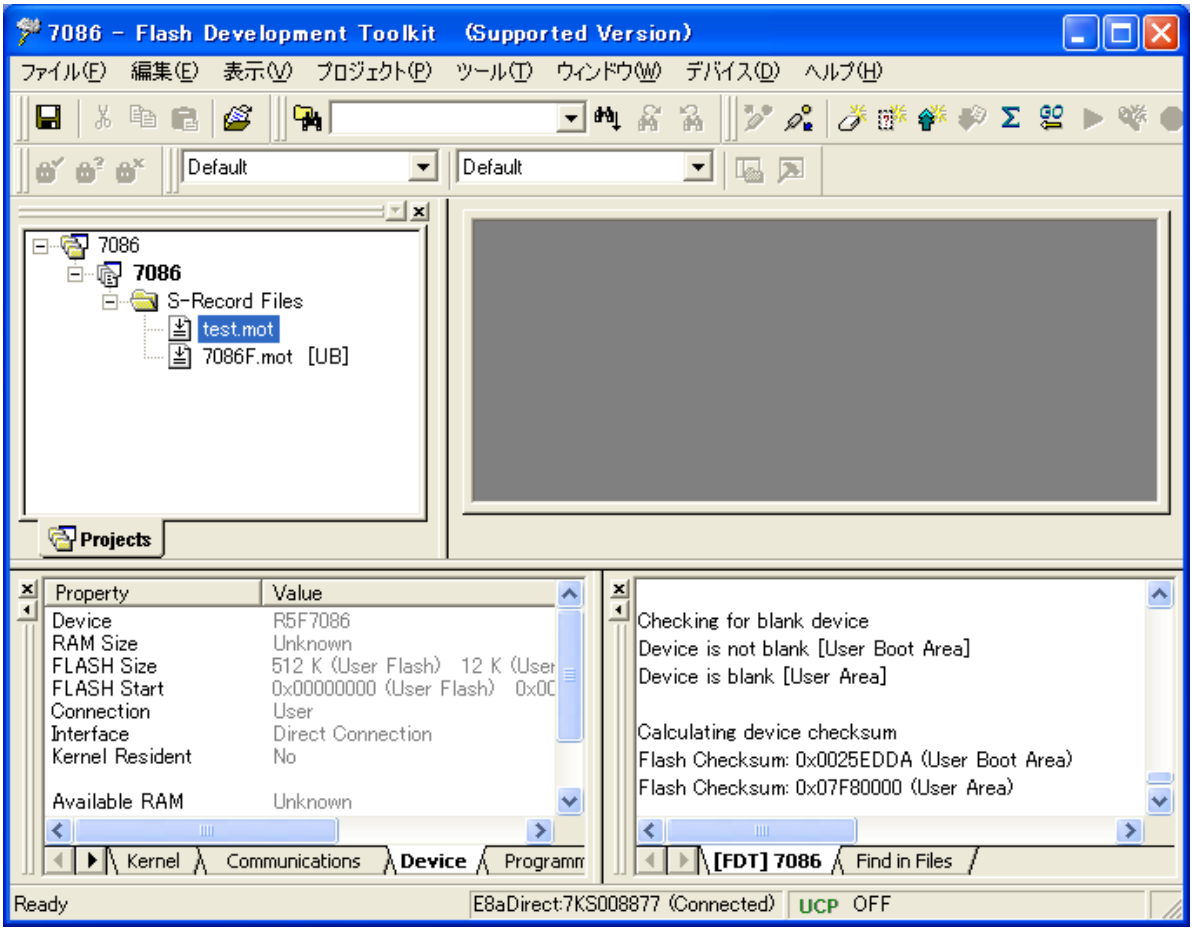
(5) [デバイス]メニューから[デバイスとの接続]をクリックし、デバイスを接続してください。



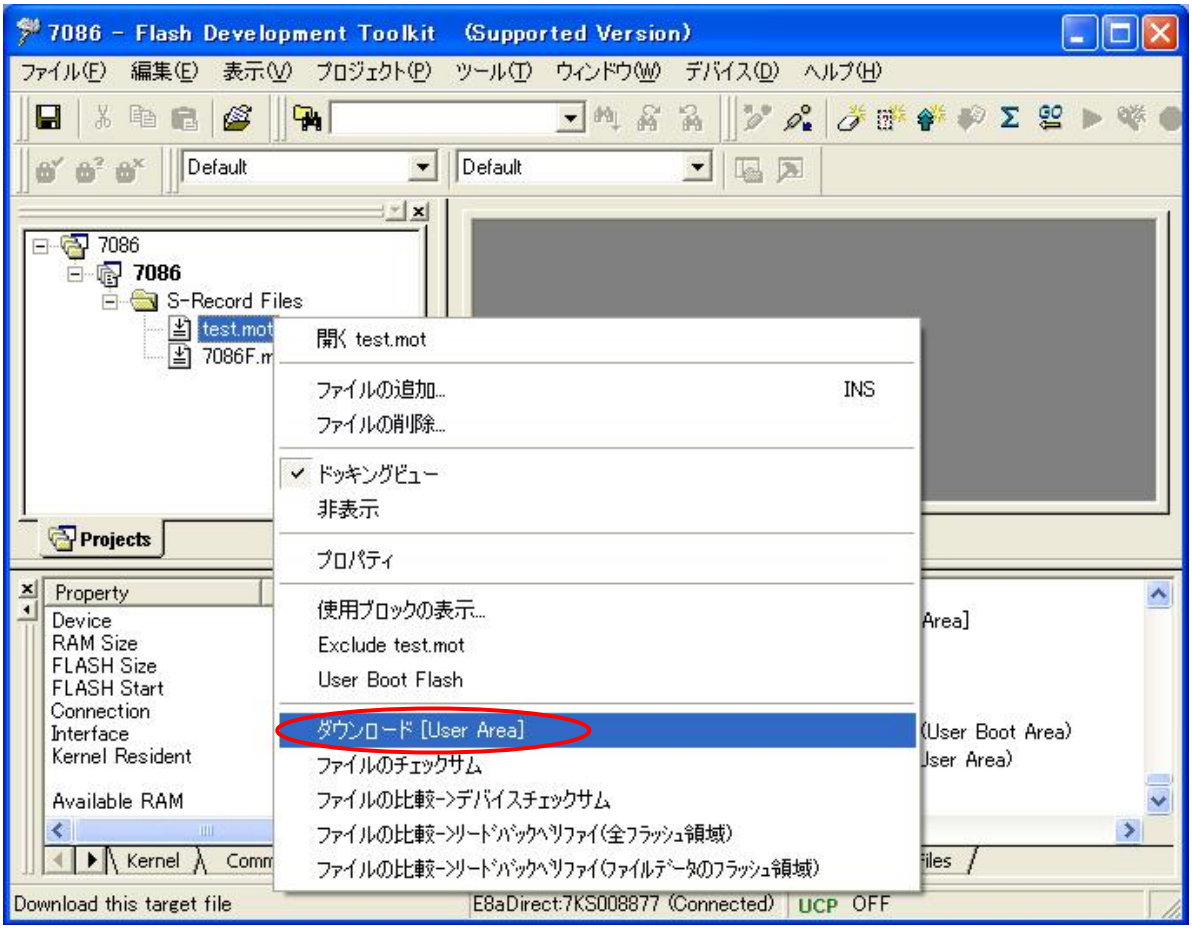
(6) ユーザエリアへ書き込むプログラムを追加してください。ここでは test.mot ファイルを書き込みます。



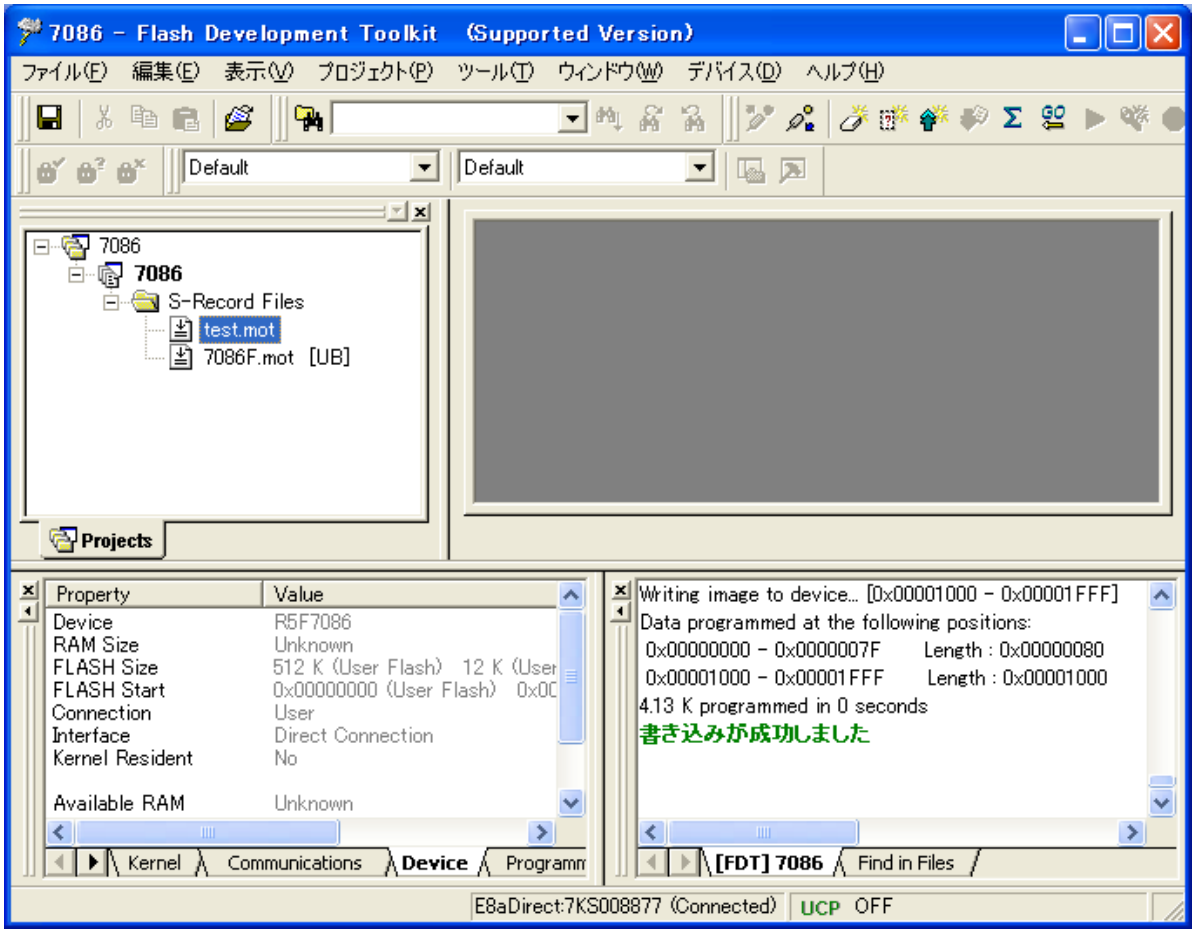
(7) test.mot ファイルを書き込む前に、ブランクチェックとチェックサムを行います。
[デバイス]メニューから[ブランクチェック]をクリックしてください。
[デバイス]メニューから[フラッシュのチェックサム]をクリックしてください。
ブランクチェックとチェックサムの結果が表示されます。



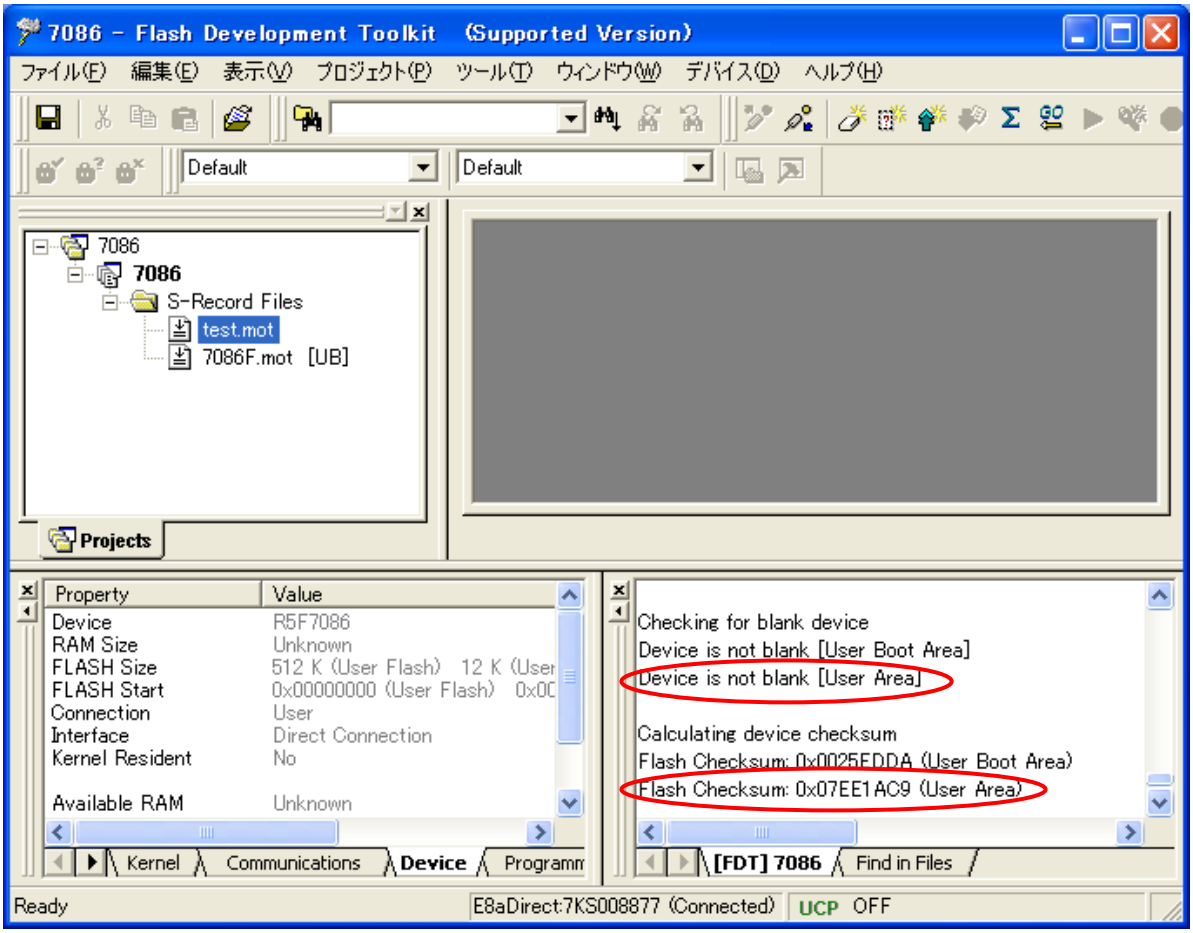
(8) ユーザブートモードで、ユーザエリアへ書き込みます。
test.mot ファイルを右クリックし、ポップアップメニューを表示させ、[ダウンロード[User Area]]をクリックし、test.mot ファイルをユーザエリアへダウンロードします。



ユーザエリアにプログラムがダウンロードされました。



(9) 書き込まれたことを確認するために、ブランクチェックとチェックサムを行います。
[デバイス]メニューから[ブランクチェック]をクリックしてください。
[デバイス]メニューから[フラッシュのチェックサム]をクリックしてください。
ブランクチェックとチェックサムの結果が表示されます。

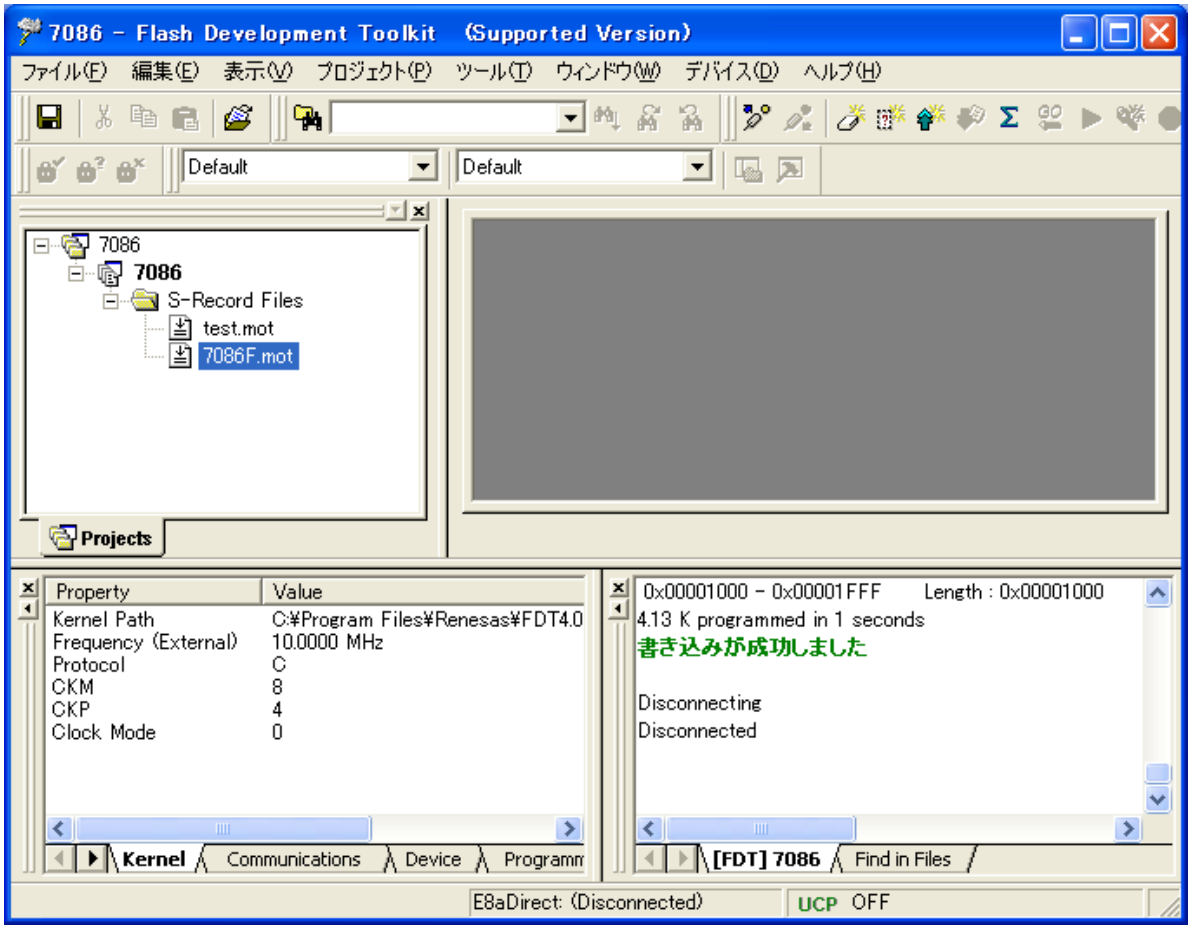


ユーザエリアにプログラムが書き込まれたことがわかります。

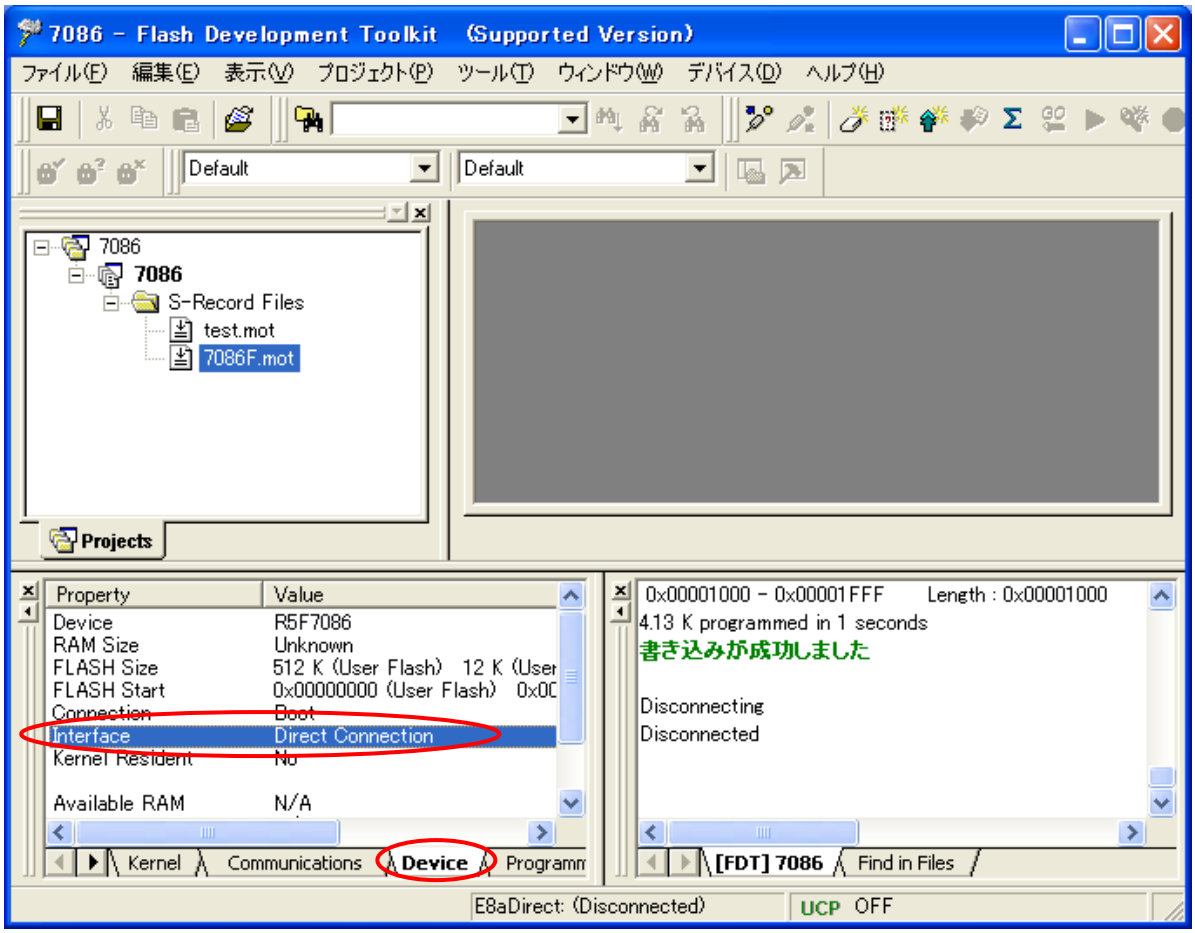
3.6 ユーザプログラムモード

ユーザプログラムモードは、ユーザエリアの書き込み消去ができます。ここでは3.5 ユーザブートモードと同じくtest.motファイルを書き込みます。ユーザブートエリアへの書き込み消去はできません。予めユーザエリアにユーザプログラムモード用ロードモジュールファイルが書き込まれている必要があります。

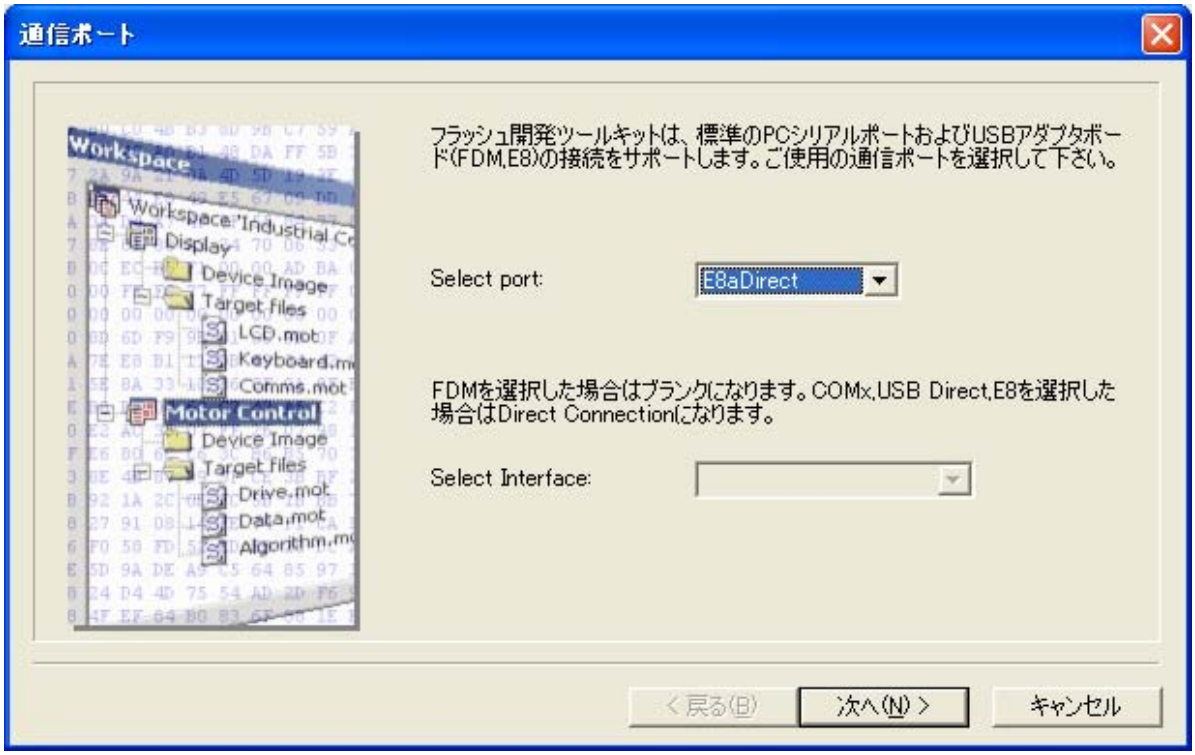
(1) [Flash Development Toolkit 4.01]を起動し、プロジェクトワークスペースファイル7086.AWSを開き、ブートモードで 7086F.mot ファイルをユーザエリアへ書き込んでください。
書き込んだ後、デバイスとの切断をし、[デバイス]メニューから[プロジェクトの設定]をクリックしてプロジェクト設定ウィンドウを表示してください。



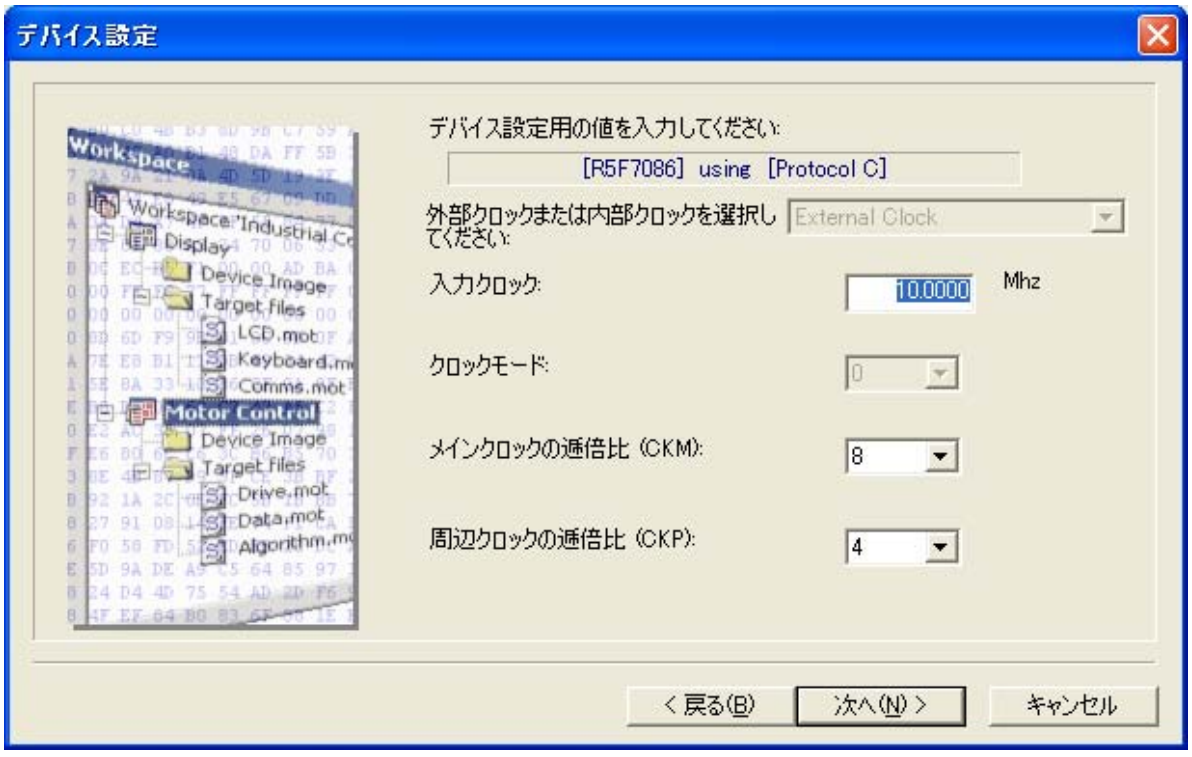
(2) ユーザプログラムモードを設定します。
プロジェクト設定ウィンドウの[Device]タブを選択し、[Interface] [Direct Connection]をダブルクリックしてください。



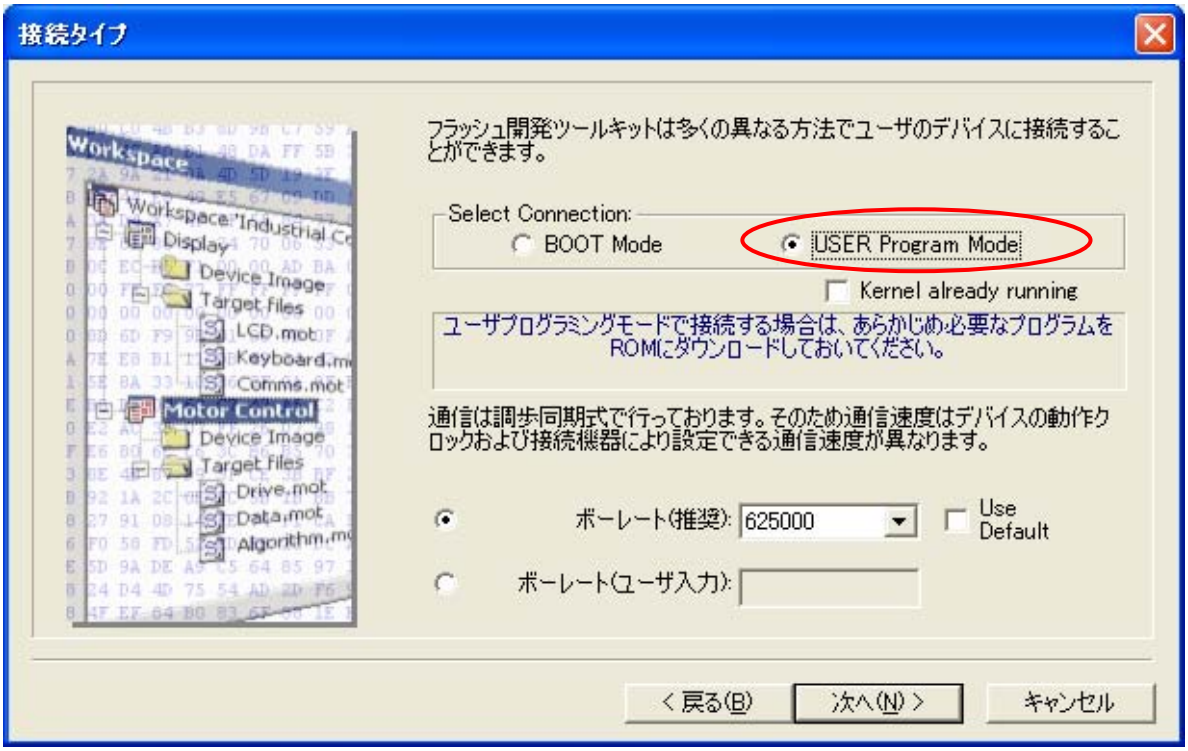
[次へ]をクリックしてください。



[次へ]をクリックしてください。

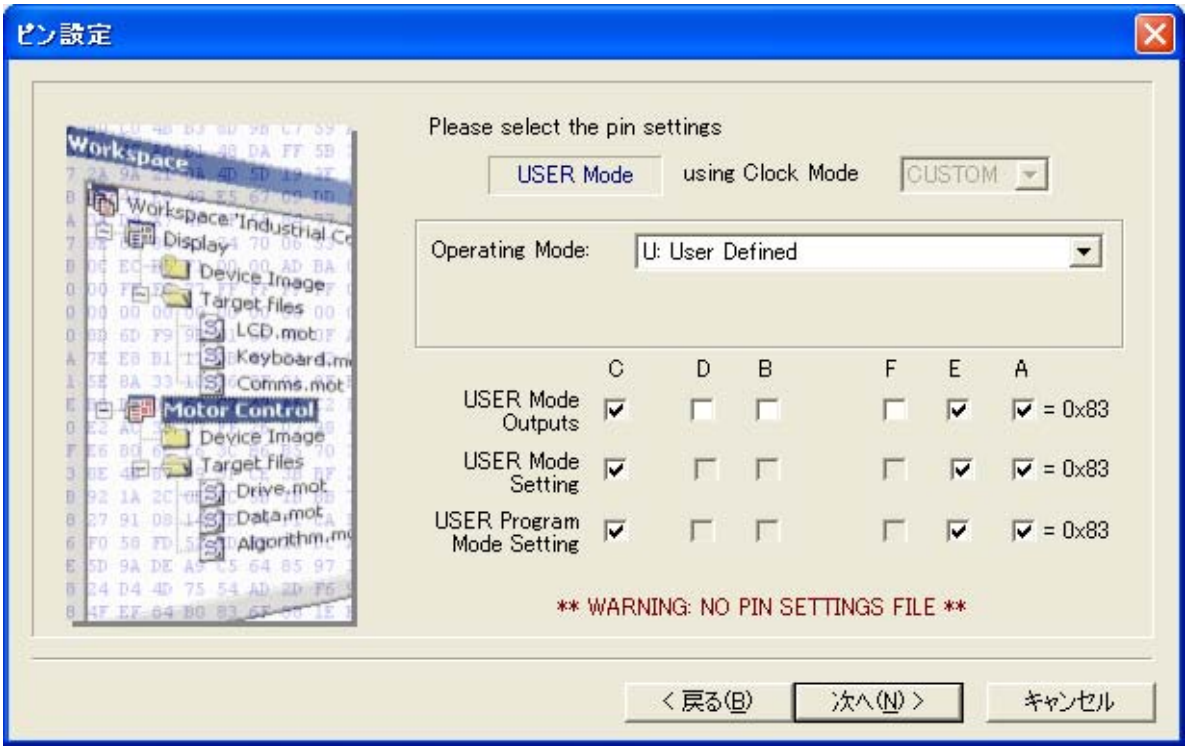


接続選択の[USER Program Mode]を選択し、[次へ]をクリックしてください。



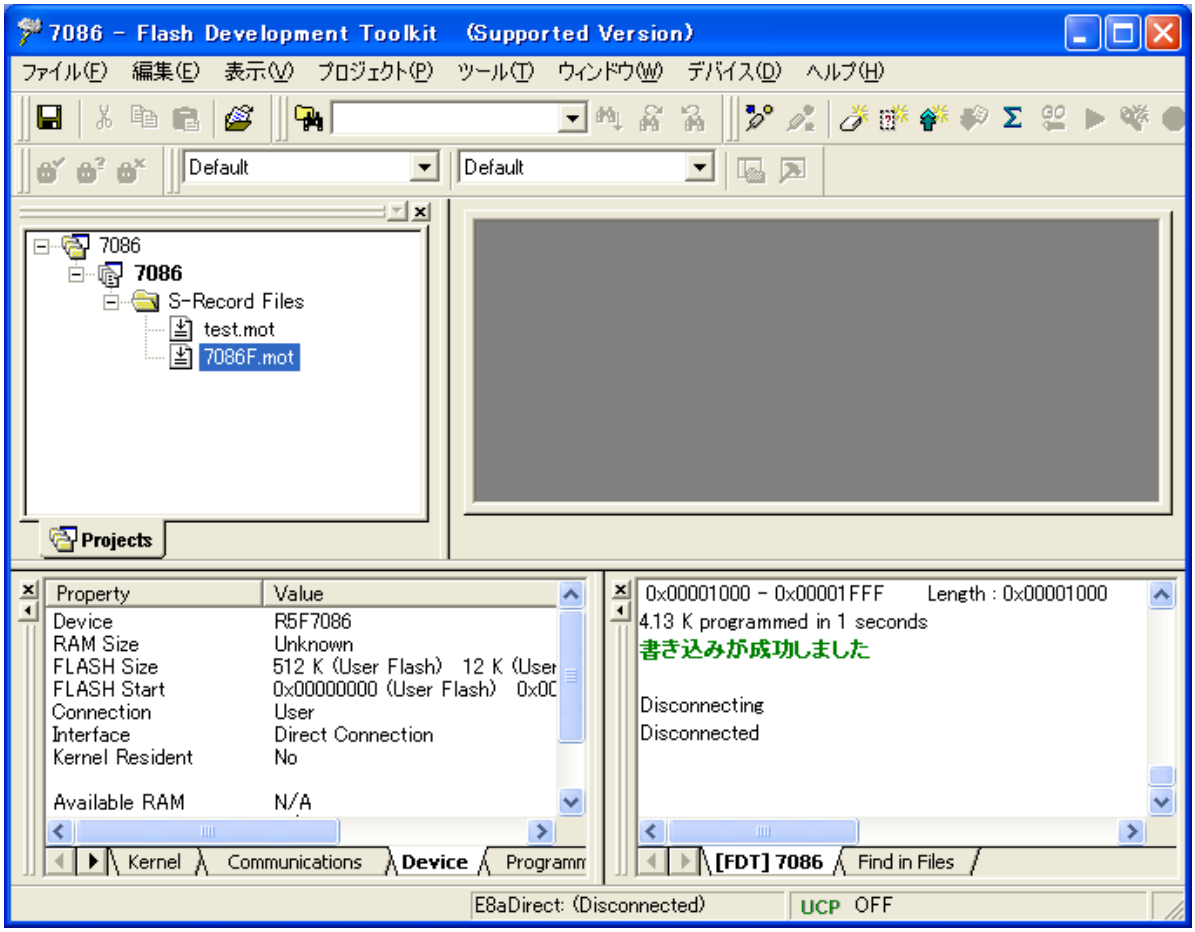
ユーザプログラムモードの E8a ピン設定をします。
ここではモード 7 に設定します。
C ピン(FWE)、E ピン(MD1)、A ピン(MD0)を 1 に設定して[次へ]をクリックしてください。

MCU 動作モード	端子設定			モード名
	FWE	MD1	MD0	
モード 6	1	1	0	ユーザプログラムモード
モード 7	1	1	1	

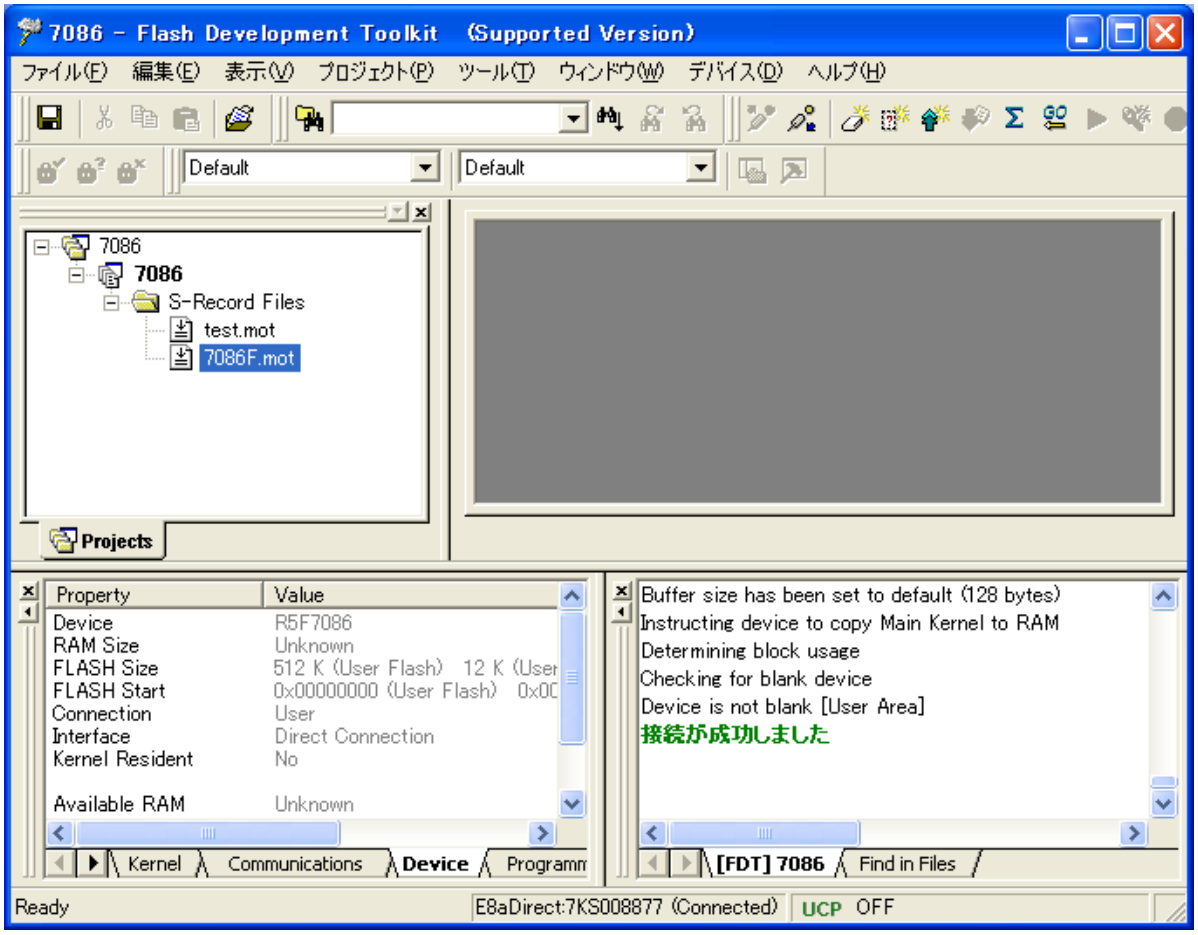


[完了]をクリックしてください。

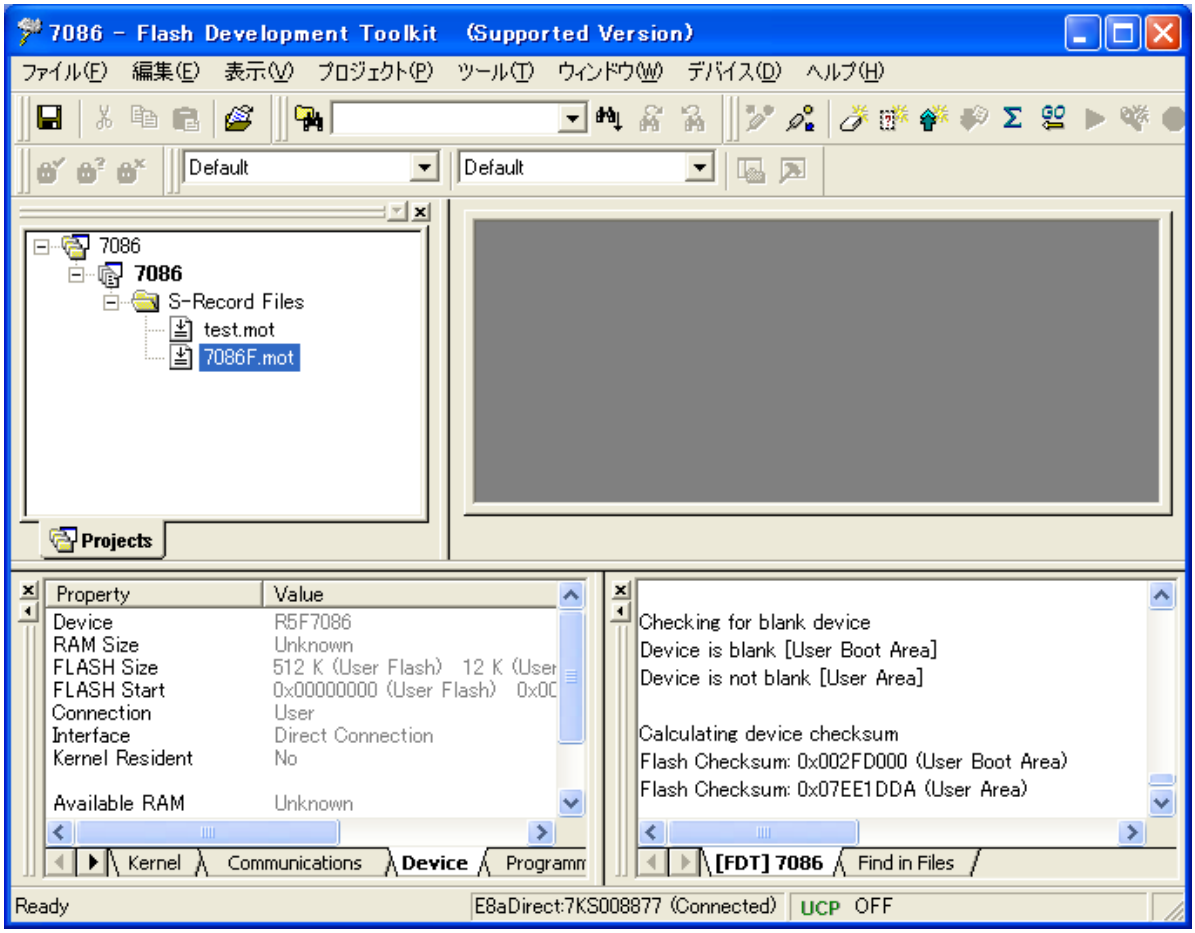
ユーザプログラムモードを設定しました。



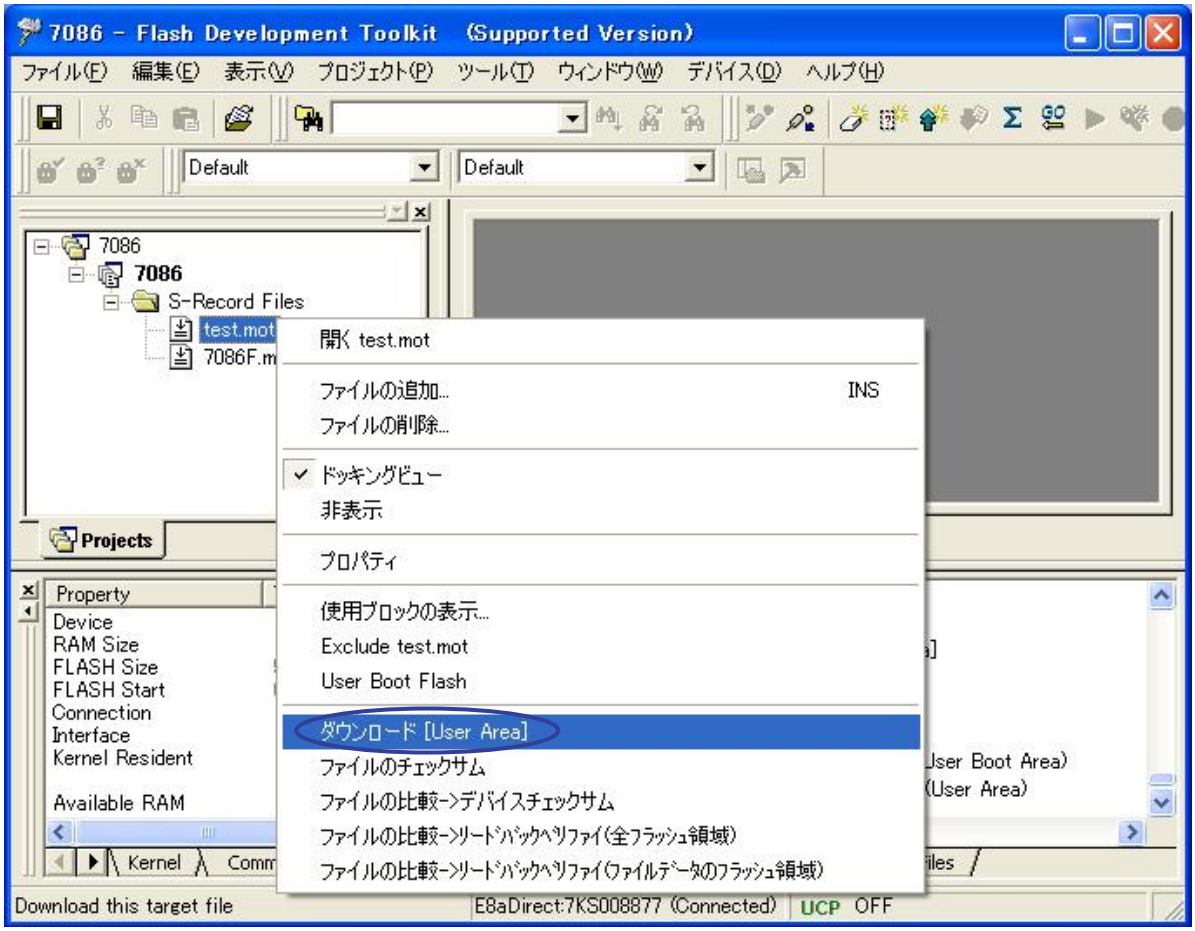
(3) [デバイス]メニューから[デバイスとの接続]をクリックし、デバイスを接続してください。



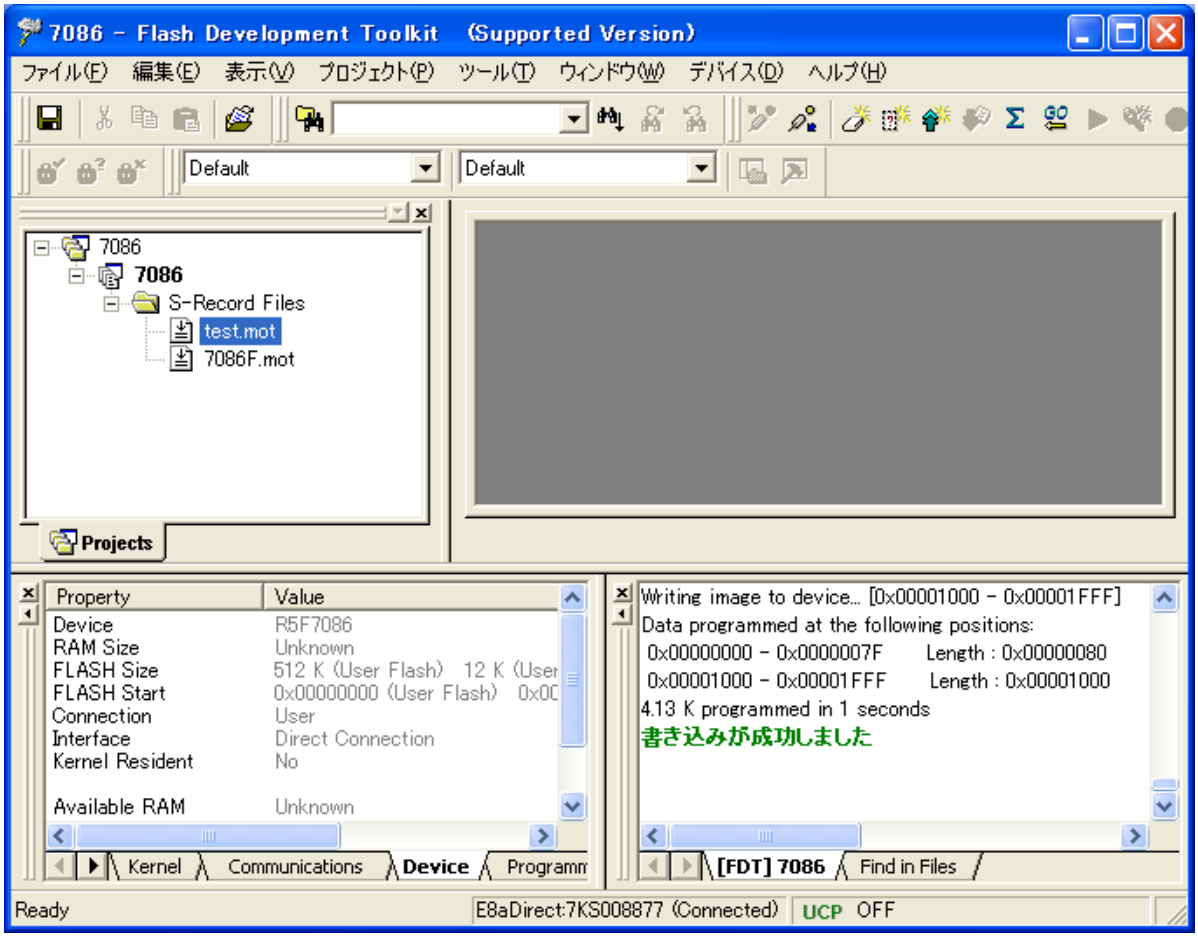
(4) test.mot ファイルを書き込む前に、ブランクチェックとチェックサムを行います。
[デバイス]メニューから[ブランクチェック]をクリックしてください。
[デバイス]メニューから[フラッシュのチェックサム]をクリックしてください。
ブランクチェックとチェックサムの結果が表示されます。



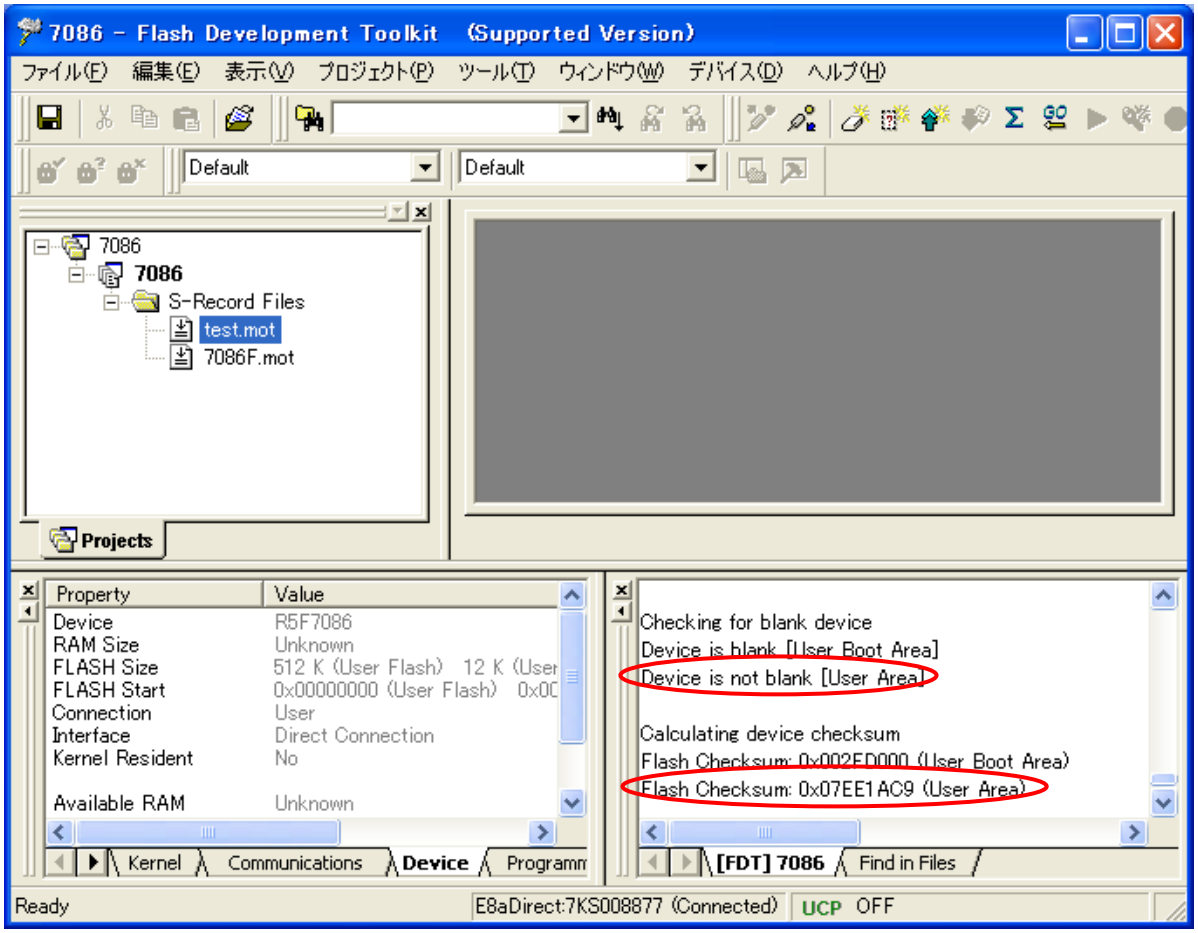
(5) ユーザプログラムモードで、ユーザエリアへ書き込みます。
test.mot ファイルを右クリックし、ポップアップメニューを表示させ、[ダウンロード[User Area]]をクリックし、test.mot ファイルをユーザエリアへダウンロードします。



ユーザエリアにプログラムがダウンロードされました。



(6) 書き込まれたことを確認するために、ブランクチェックとチェックサムを行います。
[デバイス]メニューから[ブランクチェック]をクリックしてください。
[デバイス]メニューから[フラッシュのチェックサム]をクリックしてください。
ブランクチェックとチェックサムの結果が表示されます。



ユーザエリアにプログラムが書き込まれたことがわかります。

4. フラッシュ開発ツールキットの処理

フラッシュ開発ツールキットにはブートモード、ユーザブートモード、ユーザプログラムモードの接続方法があり、それぞれ、以前のセッションから実行継続する指定ができます。通常は新規接続処理を使います。
16進数で表しているコードはフラッシュ開発ツールキットのコマンドコードです。詳しくはハードウェアマニュアル 23.フラッシュメモリを参照してください。

モード	新規接続処理	以前のセッションから実行継続
ブートモード	ボーレート合わせ込み H'27(書き込みサイズ問い合わせ) H'10(デバイス選択) H'11(クロックモード選択) H'3F(新ボーレート設定)	H'27(書き込みサイズ問い合わせ) H'4F(ステータス要求) H'4D(ユーザエリアブランクチェック)
ユーザブートモード ユーザプログラムモード	H'27(書き込みサイズ問い合わせ) H'10(デバイス選択) H'11(クロックモード選択) H'3F(新ボーレート設定)	H'27(書き込みサイズ問い合わせ) H'4F(ステータス要求) H'4D(ユーザエリアブランクチェック)

5. ユーザプログラムモード用ファイル

ユーザプログラムモード用ファイルについて説明します。

5.1 ソースファイル一覧

ソースファイル一覧を以下に示します。

ファイル	ファイル名	内容
コマンド関数	CmdFunc.c	コマンド処理ソースファイル
コマンド関数ヘッダ	CmdFunc.h	コマンド関数定義ファイル
コマンドヘッダ	commands.h	コマンドコード定義ファイル
デバイス情報ヘッダ	DeviceInfo.h	デバイス情報定義ファイル
消去関数	FDTErase.c	消去関数ソースファイル
メイン関数	FDTUMain.c	メインカーネル関数ソースファイル
メイン関数ヘッダ	FDTUMain.h	メインカーネル関数定義ファイル
書き込み関数	FDTWrite.c	書き込み関数ソースファイル
テスト関数	GenTest.c	ユーザプログラムモードテスト関数ソースファイル
テスト関数ヘッダ	GenTest.h	ユーザプログラムモードテスト定義ファイル
IO アドレスヘッダ	io7086.h	周辺モジュールレジスタ定義ファイル
ライブラリヘッダ	KAlg.h	書き込み消去ライブラリ定義ファイル
デバイスヘッダ	KDevice.h	デバイス情報定義ファイル
構造体ヘッダ	KStruct.h	構造体定義ファイル
タイプヘッダ	KTypes.h	タイプ定義ファイル
RAM アドレス定義	rom2ram.src	RAM アドレス定義ファイル
開始関数	Strt7086.src	開始関数アセンブラファイル
マイクロ関数	Ugenu.c	マイクロカーネル関数ソースファイル
マイクロ関数ヘッダ	uGenu.h	マイクロカーネル定義ファイル

5.2 モジュール一覧

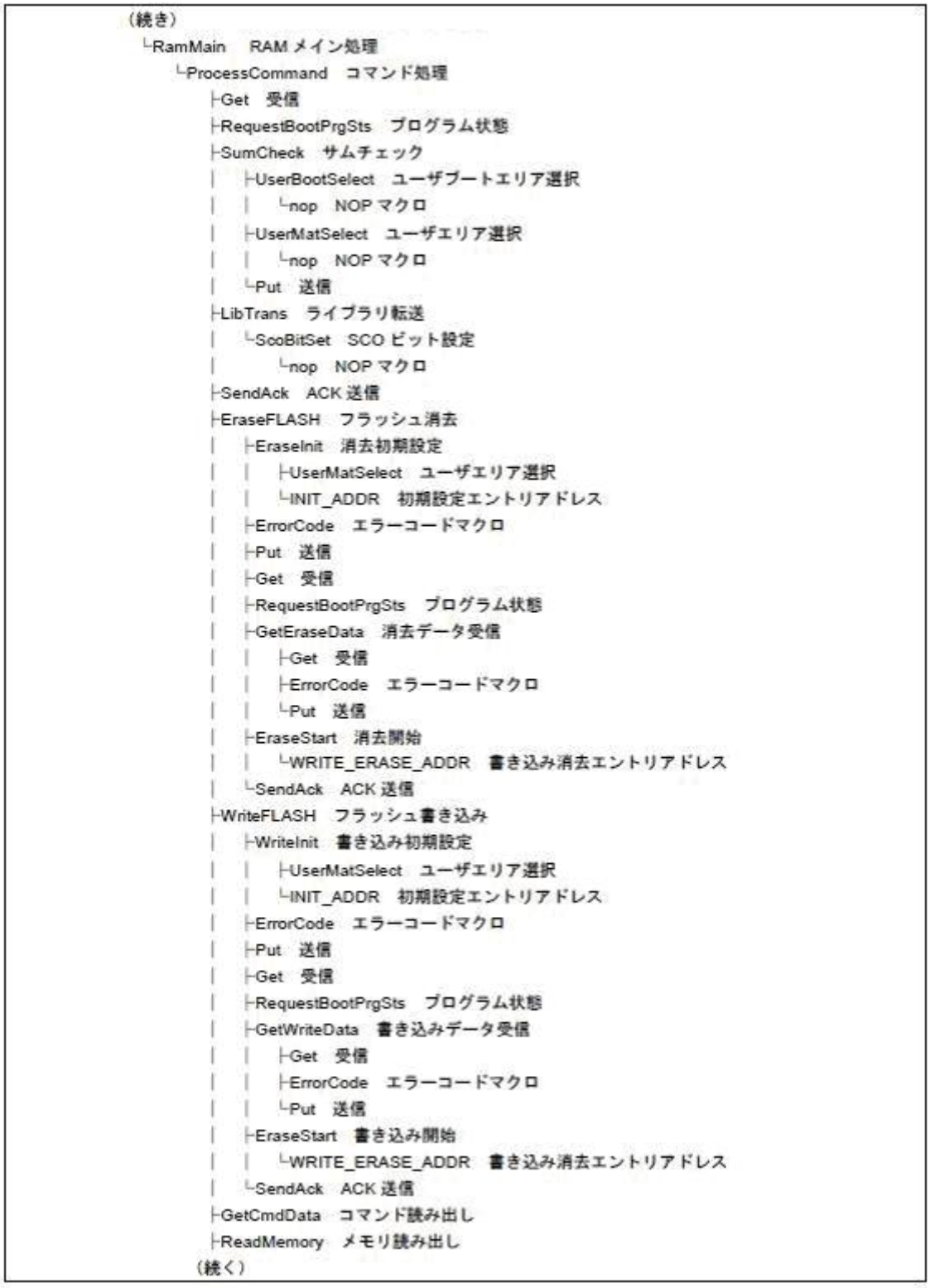
モジュール一覧を以下に示します。

ファイル	モジュール	モジュール名	機能
CmdFunc.c	参照関数	ReferFunc	参照関数
	デバイス選択	SelectDevice	デバイス選択
	クロックモード選択	SelectClockMode	クロックモード選択
	新ボーレート設定	SetNewBaudRate	新ボーレート設定
	プログラム状態	RequestBootPrgSts	プログラム状態
	サムチェック	SumCheck	サムチェック
	ACK 送信	SendAck	アック送信
	ブランクチェック	CheckBlank	ブランクチェック
	メモリ読み出し	ReadMemory	メモリ読み出し
	コマンド読み出し	GetCmdData	コマンド読み出し
FDTErase.c	フラッシュ消去	EraseFLASH	フラッシュ消去
	消去データ受信	GetEraseData	消去データ受信
	消去初期設定	EraseInit	消去初期設定
	消去開始	EraseStart	消去開始
FDTUMain.c	RAM メイン	RamMain	RAM メイン処理
	コマンド処理	ProcessCommand	コマンド処理
	ライブラリ転送	LibTrans	ライブラリ転送
	SCO ビット設定	ScoBitSet	SCO ビット設定
	ユーザブートエリア選択	UserBootSelect	ユーザブートエリア選択
	ユーザエリア選択	UserMatSelect	ユーザエリア選択
FDTWrite.c	フラッシュ書き込み	WriteFLASH	フラッシュ書き込み
	書き込みデータ受信	GetWriteData	書き込みデータ受信
	書き込み初期設定	WriteInit	書き込み初期設定
	書き込み開始	WriteStart	書き込み開始
GenTest.c	メイン処理	main	テストメイン処理
	SCI 初期設定	InitSCI	SCI 初期設定
	受信	Get	受信
	送信	Put	送信
Strt7086.src	開始	startup	スタックポインタの設定と開始
Ugenu.c	ROM メイン	RomMain	ROM メイン処理
	コマンド関数	CmdFunc	コマンドの受付と制御
	転送開始	TransStart	プログラムの転送開始
	コピー	RamCopy	プログラムの RAM へのコピー

5.3 モジュール階層構造

モジュール階層構造を以下に示します。

```
RESET_VECTOR　リセットベクタ
  └startup　開始
      └main　メイン処理
          ├InitSCI　SCI 初期設定
          └RomMain　ROM メイン処理
              ├TransStart　転送開始
              |  └RamCopy　コピー
              ├CmdFunc　コマンド関数
              |  ├Get　受信
              |  ├SendAck　ACK 送信
              |  |  └Put　送信
              |  ├ReferFunc　参照関数
              |  |  └Put　送信
              |  ├GetCmdData　コマンド読み出し
              |  |  └Get　受信
              |  ├SelectDevice　デバイス選択
              |  |  ├SendAck　ACK 送信
              |  |  ├ErrorCode　エラーコードマクロ
              |  |  └Put　送信
              |  ├SelectClockMode　クロックモード選択
              |  |  ├SendAck　ACK 送信
              |  |  ├ErrorCode　エラーコードマクロ
              |  |  └Put　送信
              |  ├SetNewBaudRate　新ビットレート設定
              |  |  ├ErrorCode　エラーコードマクロ
              |  |  ├Put　送信
              |  |  ├cal_brr　BRR 計算
              |  |  ├SendAck　ACK 送信
              |  |  └Get　受信
              |  ├RequestBootPrgSts　プログラム状態
              |  |  └Put　送信
              |  └Put　送信
              └RamMain　RAM メイン処理
            (続く)
```


(続き)
RamMain RAM メイン処理
ProcessCommand コマンド処理
Get 受信
RequestBootPrgSts プログラム状態
SumCheck サムチェック
UserBootSelect ユーザブートエリア選択
nop NOP マクロ
UserMatSelect ユーザエリア選択
nop NOP マクロ
Put 送信
LibTrans ライブラリ転送
ScoBitSet SCO ビット設定
nop NOP マクロ
SendAck ACK 送信
EraseFLASH フラッシュ消去
EraseInit 消去初期設定
UserMatSelect ユーザエリア選択
INIT_ADDR 初期設定エントリアドレス
ErrorCode エラーコードマクロ
Put 送信
Get 受信
RequestBootPrgSts プログラム状態
GetEraseData 消去データ受信
Get 受信
ErrorCode エラーコードマクロ
Put 送信
EraseStart 消去開始
WRITE_ERASE_ADDR 書き込み消去エントリアドレス
SendAck ACK 送信
WriteFLASH フラッシュ書き込み
WriteInit 書き込み初期設定
UserMatSelect ユーザエリア選択
INIT_ADDR 初期設定エントリアドレス
ErrorCode エラーコードマクロ
Put 送信
Get 受信
RequestBootPrgSts プログラム状態
GetWriteData 書き込みデータ受信
Get 受信
ErrorCode エラーコードマクロ
Put 送信
EraseStart 書き込み開始
WRITE_ERASE_ADDR 書き込み消去エントリアドレス
SendAck ACK 送信
GetCmdData コマンド読み出し
ReadMemory メモリ読み出し
(続く)

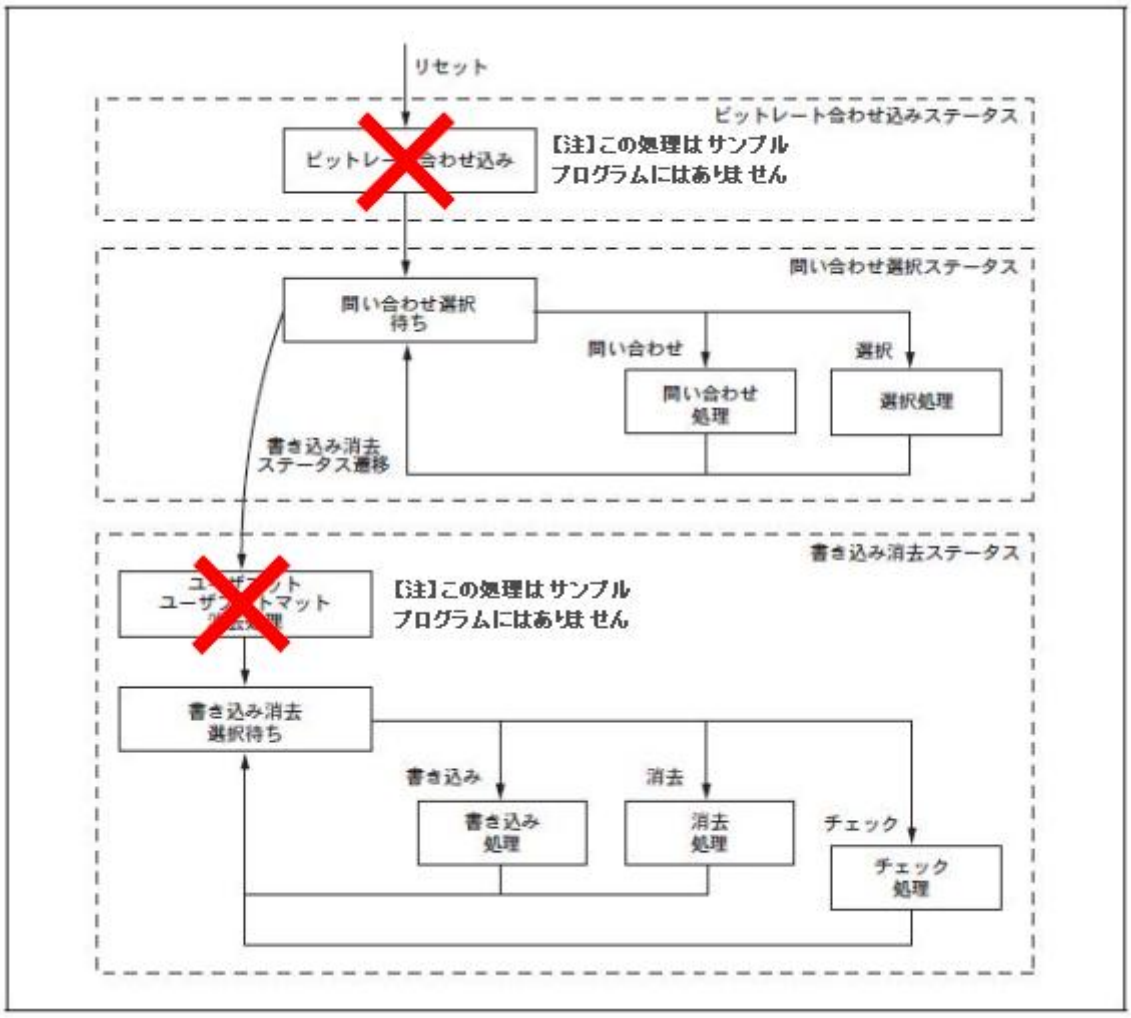

（続き）
ReadMemory　メモリ読み出し
UserBootSelect　ユーザブートエリア選択
UserMatSelect　ユーザエリア選択
ErrorCode　エラーコードマクロ
Put　送信
CheckBlank　ブランクチェック
UserBootSelect　ユーザブートエリア選択
UserMatSelect　ユーザエリア選択
ErrorCode　エラーコードマクロ
Put　送信
SendAck　ACK 送信
Put　送信

5.4 プログラムの流れ

モジュール階層構造を参照して、プログラムの流れを説明します。

(1) プログラムの処理フロー

プログラムの処理フローを以下に示します。ユーザプログラムモードでは、ブートで行うビットレート合わせ込み、ユーザエリア消去処理は行いません。そのため、フラッシュメモリに書き込まれたプログラムとデータは保存することができます。



(2) メイン処理(main)

メイン処理の流れを以下に示します。

1.リセットベクタから開始(startup)へ分岐します。

2.開始(startup)はスタックポインタを設定し、メイン処理(main)を呼びます。

3.メイン処理(main)は、SCI初期設定(InitSCI)を呼んで、ROMメイン処理(RomMain)へ分岐します。

4.ROMメイン処理(RomMain)は、RAMメイン処理をRAMに転送し、コマンドを受け付け、処理し、指定を設定します。
設定が終了すると、RAM上のRAMメイン処理(RamMain)へ分岐します。

5.RAMメイン処理(RamMain)は受信したコマンドを処理し、以下の処理をします。
書き込み消去ライブラリのライブラリ転送(LibTrans)
フラッシュメモリの消去(EraseFLASH)
フラッシュメモリの書き込み(WriteFLASH)
ユーザブートエリア、ユーザエリアのメモリ読み出し(ReadMemory)
ユーザブートエリア、ユーザエリアのサムチェック(SumCheck)
ユーザブートエリア、ユーザエリアのブランクチェック(CheckBlank)

【注】 ROMメイン処理(RomMain)は、マイクロカーネルとも呼ばれます。ROM上で動作します。
RAM メイン処理(RamMain)は、メインカーネルとも呼ばれます。RAM 上で動作します。

(3) ROM メイン処理(RomMain)

ROMメイン処理(RomMain)の流れを以下に示します。

1.転送開始(TransStart)で、ROM上のプログラムをRAMへ転送します。
ライブラリ転送、消去、書き込みをRAM上で処理するためです。

2.コマンド関数(CmdFunc)でコマンドを処理し、問い合わせに対応し、選択を設定します。

3.問い合わせ対応は参照関数(ReferFunc)とプログラム状態(RequestBootPrgSts)で以下のコマンドに対応します。
サポートデバイス問い合わせ
クロックモード問い合わせ
逓倍比問い合わせ
動作周波数問い合わせ
ユーザブートエリア情報問い合わせ
ユーザエリア情報問い合わせ
消去ブロック情報問い合わせ
書き込みサイズ問い合わせ
ブートプログラムステータス問い合わせ

4.選択の設定のコマンドは以下のモジュールで設定します。
デバイス選択(SelectDevice):デバイスコードの選択
クロックモード選択(SelectClockMode):選択されているクロックモードの通知
新ビットレート設定(SetNewBaudRate):新ビットレートの選択

5.問い合わせ選択が完了し、RAM に転送した RAM メイン処理(RamMain)に分岐します。

(4) RAM メイン処理(RamMain)

RAMメイン処理(RamMain)の流れを以下に示します。

1.コマンド処理(ProcessCommand)で、コマンドを処理します。処理するコマンドを以下に示します。
ユーザプログラムモード用ファイルは、ユーザプログラミングモードで動作させるため、ユーザブートエリア書き込み選択、ユーザブートエリアのブロック消去はできません。
ユーザエリア書き込み選択
128バイト書き込み
消去選択
ブロック消去
メモリリード
ユーザブートエリアのサムチェック
ユーザエリアのサムチェック
ユーザブートエリアのブランクチェック
ユーザエリアのブランクチェック
ブートプログラムステータス問い合わせ

2.ユーザエリア書き込み選択コマンドでは、ライブラリ転送(LibTrans)で書き込みライブラリを転送し、フラッシュ書き込み(WriteFLASH)へ分岐します。

3.フラッシュ書き込み(WriteFLASH)では、書き込み初期設定(WriteInit)で、周波数を設定します。
次にコマンドを読み込み、128バイト書き込みならば、書き込みデータ受信(GetWriteData)で書き込みデータを受信し、書き込み開始(EraseStart)で、フラッシュメモリに書き込みます。
128バイトデータが書き込み終了のアドレスデータならば、書き込むデータと書き込み先アドレスにそれぞれ書き込み終了コードを設定して、書き込み終了(実際は書き込み開始(EraseStart)を呼ぶ)させ、書き込み処理を終了します。

4.消去選択コマンドでは、ライブラリ転送(LibTrans)で消去ライブラリを転送し、フラッシュ消去(EraseFLASH)へ分岐します。

5.フラッシュ消去(EraseFLASH)では、消去初期設定(EraseInit)で、周波数を設定します。
次にコマンドを読み込み、ブロック消去ならば、消去データ受信(GetEraseData)で消去データを受信し、消去開始(EraseStart)で、指定されたブロックを消去します。
消去データが消去終了データならば、消去処理を終了します。

6.メモリリードコマンドでは、コマンド読み出し(GetCmdData)で、読み出すアドレスが指定されます。メモリ読み出し(ReadMemory)でユーザブートエリア、ユーザエリアのメモリを読み出します。

7.ユーザブートエリアのサムチェック、ユーザエリアのサムチェックコマンドでは、サムチェック(SumCheck)でユーザブートエリア、ユーザエリアのサムチェックをします。

8.ユーザブートエリアのブランクチェック、ユーザエリアのブランクチェックコマンドでは、ブランクチェック(CheckBlank)でユーザブートエリア、ユーザエリアのブランクをチェックします。

9.ブートプログラムステータス問い合わせコマンドでは、プログラム状態(RequestBootPrgSts)でブートの処理状態を送信します。

6. ユーザプログラムモード用ファイルのソース

ユーザプログラムモード用ファイルの主なソースを以下に示します。

6.1 ヘッダファイル

ユーザプログラムモード用ファイルは、以下のヘッダファイルを使っています。

(1) ビットレートの設定(GenTest.h)

ビットレートの設定をしています。

```
                                    /* RATE:9600/bps CLOCK:10MHz Main:x2 Peripheral:x2 */
#define MA_BRR_SCI                  0x40        /* Bit rate register channel 1 */
```

ユーザプログラムモードは9600bpsで接続します。そのため、SCIモジュールのビットレートレジスタ(BRR)の値を動作周波数に従って設定することが必要です。ここでは、20MHz(10MHz×2)なので、9600bpsにするために、MA_BRR_SCIを64(0x40)に設定しています。ビットレート9600bps時の動作周波数とBBRレジスタの設定値の関係を以下に示します。

動作周波数 φ(MHz)	BRRの設定	誤差(%)
10	32	-1.36
12	38	0.16
14	45	-0.93
16	51	0.16
18	58	-0.69
20	64	0.16
22	71	-0.54
24	77	0.16
26	84	-0.43
28	90	0.16
30	97	-0.35
32	103	0.16
34	110	-0.29
36	116	0.16
38	123	-0.24
40	129	0.16

ボードの動作周波数に対応して、MA_BRR_SCIの値を設定して、HEWでビルドし、Sタイプファイルのプログラムを作成します。

(2) IO レジスタ定義(io7086.h)
SCI モジュールと ROM に関係するレジスタとビット、PFC レジスタを定義しています。

```
/************************************************************************/
/*    SH/7086F Internal I/O Include File                                */
/************************************************************************/
/*-----------------------------------------------------------------------*/
/*              SCI                                                      */
/*-----------------------------------------------------------------------*/
/*                      CHANNEL 1                                        */
/*-----------------------------------------------------------------------*/
#define SCI_SMR         (*(volatile unsigned char *)0xFFFFC080)
#define SCI_BRR         (*(volatile unsigned char *)0xFFFFC082)
#define SCI_SCR         (*(volatile unsigned char *)0xFFFFC084)
#define SCI_TDR         (*(volatile unsigned char *)0xFFFFC086)
#define SCI_SSR         (*(volatile unsigned char *)0xFFFFC088)
#define SCI_RDR         (*(volatile unsigned char *)0xFFFFC08A)

#define TE                              (unsigned char)0x20
#define RE                              (unsigned char)0x10
#define TE_RE                   (unsigned char)(TE | RE)
#define TDRE                    (unsigned char)0x80
#define RDRF                    (unsigned char)0x40
#define RDRF_ERR_CLR (unsigned char)0x87
#define TEND                    (unsigned char)0x04

/*-----------------------------------------------------------------------*/
/*              FLASH                                                    */
/*-----------------------------------------------------------------------*/
/*                                                                       */
/*-----------------------------------------------------------------------*/
#define FCCS            (*(volatile unsigned char *)0xFFFFCC00)
#define FPCS            (*(volatile unsigned char *)0xFFFFCC01)
#define FECS            (*(volatile unsigned char *)0xFFFFCC02)
#define FKEY            (*(volatile unsigned char *)0xFFFFCC04)
#define FMATS                   (*(volatile unsigned char *)0xFFFFCC05)
#define FTDAR                   (*(volatile unsigned char *)0xFFFFCC06)

/*-----------------------------------------------------------------------*/
/*              PFC                                                      */
/*-----------------------------------------------------------------------*/
/*                                                                       */
/*-----------------------------------------------------------------------*/
#define PACRL2                          (*(volatile unsigned short *)0xFFFFD114)
 #define PA4MD0                         (unsigned short)0x0001
#define PACRL1                          (*(volatile unsigned short *)0xFFFFD116)
 #define PA3MD0                         (unsigned short)0x1000
```

```
/*------------------------------------------------------------------------*/
/*                                                                        */
/*------------------------------------------------------------------------*/
/*                                                                        */
/*------------------------------------------------------------------------*/
#define FRQCR                   (*(volatile unsigned short *)0xFFFFE800)

/*------------------------------------------------------------------------*/
/*                                                                        */
/*------------------------------------------------------------------------*/
/*                                                                        */
/*------------------------------------------------------------------------*/
#define STBCR3                  (*(volatile unsigned char *)0xFFFFE806)
 #define MSTP12                 (unsigned char)0x10
```

(3) マクロ定義（FDTUMain.h、KAlg.h）
プログラムで使うラベルを定義しています。

・FDTUMain.h

```
/* ＤＥＦＩＮＥ　*/
enum {
        FmatsUserBootMat = 0xaa,
        FmatsUserMat = 0x00,
        WriteMode = 0x01,
        EraseMode = 0x01,
        FkeyEnable = 0xA5
};
```

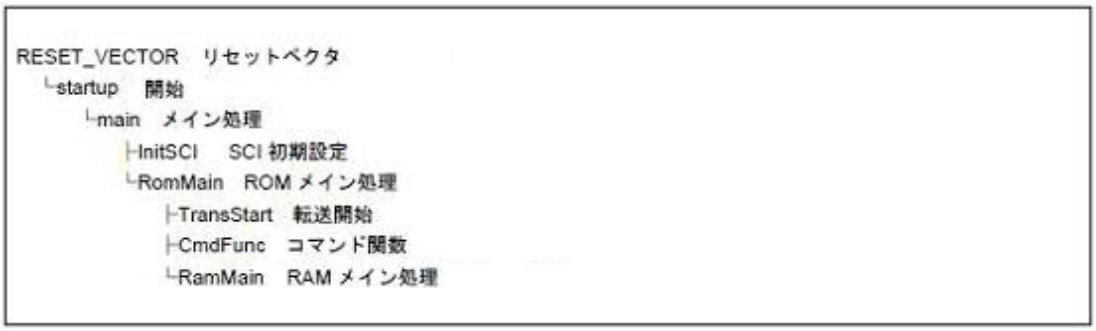
・KAlg.h

```
/* ＤＥＦＩＮＥＳ */
#define LOOP_END                        1
#define bufSize                                 0x80
#define BLOCK_NO_ERROR                  0x09
#define ERASE_END                       0xFF
#define WRITE_END                       0xFFFFFFFF
#define ADDRESS_ERROR                   0x03
#define WRITE_ERASE_ENABLE              0x5A
```

6.2 メイン処理と ROM メイン処理

(1) モジュール階層構造

メイン処理と ROM メイン処理のモジュール階層構造を以下に示します。



リセットベクタから開始に分岐し、スタックポインタを設定して(Strt7086.src)、メイン処理(GenTest.c、main)に分岐します。
メイン処理では、SCIを初期設定(GenTest.c、InitSCI)し送受信可能にし、ROMメイン処理(Ugenu.c、RomMain)に分岐します。
ROMメイン処理は、RAMメイン処理その他をRAMに転送し(Ugenu.c、TransStart)、コマンドを処理(Ugenu.c、CmdFunc)します。データ終了で、RAMメイン処理(FDTUMain.c、RamMain)に分岐します。
メイン処理、ROM メイン処理は ROM 上で実行されます。

(2) リセットベクタ(GenTest.c、GenTest.h)

リセットベクタを以下に示します。

・GenTest.c

```
/*Declare the vector table*/
#pragma section _VECT
const DWORD RESET_VECTOR = (DWORD)RESET_JMP_ADDRESS;
#pragma section
```

・GenTest.h

```
#define RESET_JMP_ADDRESS     0x1000
```

(3) 転送開始(Ugenu.c、rom2ram.src)

RAMメイン処理そのほかの転送は、転送テーブル(rom2ram.src)にしたがって、以下のモジュールをROMからRAMへ転送します。セクションは、ROMオプションを使っています。

セクション	モジュール
P_RAM_SCI	Get、Put (GenTest.c)
P_RAM_MAIN	RamMain ほか (FDTUMain.c)
P_RAM_CMD	RequestBootPrgSts など (CmdFunc.c)
P_RAM_WRITE	WriteFLASH ほか (FDTWrite.c)
P_RAM_ERASE	EraseFLASH (FDTErase.c)

(4) コマンド関数(Ugenu.c、commands.h 、CmdFunc.c、DeviceInfo.h)

コマンド関数(CmdFunc)は、問い合わせと設定コマンドの処理を行います。コマンドは、マクロ定義し(commands.h)、それぞれのコマンドに対応した処理(CmdFunc.c)をします。問い合わせコマンドに対しては、コマンドに対応したレスポンス(DeviceInfo.h)を出力する処理(CmdFunc.c 、ReferFunc)になっています。

6.3 RAM メイン処理

RAMメイン処理は、ライブラリの転送、フラッシュメモリの消去、フラッシュメモリの書き込みです。これらの処理は、RAM上で実行されます。

(1) ライブラリ転送（FDTUMain.c）

・LibTrans
commanndIDがprepareErase（0x48）ならばFECSをEraseMode（0x01）にして消去ライブラリを選択、それ以外（prepareUserAreaWrite、0x43）ならばFPCSをWriteMode（0x01）にして、書き込みライブラリを選択します。FKEYをFkeyEnable（0xA5）にして転送を選択し、SCOビットを設定します。

```
/*
//////////////////////
// LibTrans Function //
//////////////////////
*/
void LibTrans(BYTE commandID)
{
	if (commandID == prepareErase){
		FECS = EraseMode;
	}else{
		FPCS = WriteMode;
	}

	FKEY = FkeyEnable;

	ScoBitSet();
}
```

・ScoBitSet
FTDARレジスタにライブラリ転送先アドレスを設定し、VBRレジスタを0x84000000に、FCCSレジスタのSCOビットを1に設定します。SCOビット設定の後にはNOP命令が4個以上必要です。
転送時エラーが発生したかどうかを判定するため、転送前にライブラリ転送先アドレスの0xFFを書き込み、転送後0x00であることを確認します。

```
/*
//////////////////////
// ScoBitSet Function //
//////////////////////
*/
BYTE ScoBitSet(void)
{
	volatile BYTE i;
	WORD	work;
	void	**save_vbr;

	/* Transmission error check initialization */
	*((volatile BYTE *)TRANS_RAM_ADDR) = 0xFF;

	FTDAR = FTDAR_VALUE;

	save_vbr = (void **)get_vbr();
	set_vbr((void **)0x84000000);

	work = FRQCR;
	FRQCR = 0x36DB;

	FCCS |= 0x01;					/* SCO interruption */

/*	for(i=0; i < 2; i++);*/
	nop();
	nop();
	nop();
	nop();

	set_vbr(save_vbr);
	FRQCR = work;

	/* Transmission error check */
	if(0x00 == *((volatile BYTE *)TRANS_RAM_ADDR)) {
		return(NORMAL);				/* Transmission normal end */
	}

	return(ABNORMAL);				/* Transmission error */
}
```

TRANS_RAM_ADDR と FTDAR_VALUE は KDevice.h で以下のように定義されています。

```
	/* SCO define */
	#define TRANS_RAM_ADDR		0xFFFF9000
	#define FTDAR_VALUE			0x00	/* RAMTOP+0Kb */
```

(2) エリア選択(FDTUMain.c)
ユーザブートエリア、またはユーザエリアを選択するために、FMATSレジスタにFmatsUserBootMat(0xaa)またはFmatsUserMat(0x00)を設定します。設定後にはNOP命令が2個以上必要です。

```
/*
//////////////////////////////
// UserBootSelect Function //
//////////////////////////////
*/
void UserBootSelect(void)
{
	volatile BYTE i;

	FMATS = FmatsUserBootMat;
	for(i=0; i < 1; i++);
/*
	nop();
	nop();
*/
}


/*
//////////////////////////////
// UserMatSelect Function //
//////////////////////////////
*/
void UserMatSelect(void)
{
	volatile BYTE i;

	FMATS = FmatsUserMat;
	for(i=0; i < 1; i++);
/*
	nop();
	nop();
*/
}
```

(3) フラッシュメモリ消去（FDTErase.c）

・EraseInit
ユーザエリアを選択し、動作周波数を指定して消去ライブラリの初期設定します。動作周波数は、FDTで指定した動作周波数は新ビットレート選択でデバイスに送信されます。ライブラリの初期設定ではこの動作周波数を使います。

```
/*
////////////////////////
// EraseInit Function //
////////////////////////
*/
BYTE EraseInit(void)
{
    InitPtr ERASE_INIT = (InitPtr)INIT_ADDR;

    FKEY = WRITE_ERASE_ENABLE;
    return ((*ERASE_INIT)(Frequency,0));
}
```

・EraseStart
消去したいブロック番号を指定して、消去ライブラリを呼びます。ブロック番号は、フラッシュ開発ツールキットから受信しています。詳細はユーザプログラムモード用ファイルのソースを参照してください。

```
/*
/////////////////////////
// EraseStart Function //
/////////////////////////
*/
BYTE EraseStart(BYTE blk_no)
{
    ErasePtr ERASE_BLOCK = (ErasePtr)WRITE_ERASE_ADDR;

    return ((*ERASE_BLOCK)(blk_no));
}
```

INIT_ADDR と WRITE_ERASE_ADDR は KDevice.h で以下のように定義されています。

```
#define TRANS_RAM_ADDR          0xFFFF9000
#define INIT_ADDR               (TRANS_RAM_ADDR+32)
#define WRITE_ERASE_ADDR        (TRANS_RAM_ADDR+16)
```

(4) フラッシュメモリ書き込み（FDTWrite.c）

・WriteInit
ユーザエリアを選択し、動作周波数を指定して書き込みライブラリの初期設定します。

```
/*
////////////////////////
// WriteInit Function //
////////////////////////
*/
BYTE WriteInit(void)
{
	InitPtr WRITE_INIT = (InitPtr)INIT_ADDR;

	FKEY = WRITE_ERASE_ENABLE;
	return ((*WRITE_INIT)(Frequency,0));
}
```

・WriteStart
書き込むデータの格納アドレスと書き込み先のアドレスを指定して、書き込みライブラリを呼び出します。
書き込むデータと書き込み先アドレスは、フラッシュ開発ツールキットから受信しています。詳細はユーザプログラムモード用ファイルのソースを参照してください。

```
/*
/////////////////////////
// WriteStart Function //
/////////////////////////
*/
BYTE WriteStart(BYTE *data, DWORD adr)
{
	WritePtr WRITE_DATA = (WritePtr)WRITE_ERASE_ADDR;

	return ((*WRITE_DATA)((BYTE *)data, (BYTE *)adr));
}
```

・書き込み終了処理の実行（WriteFLASH）
フラッシュメモリ書き込みの終了処理の一部です。詳細はユーザプログラムモード用ファイルのソースを参照してください。
書き込みデータ受信（GetWriteData）で、書き込むデータ格納アドレスと書き込み先アドレスを受信します。
もし書き込み先アドレスがWRITE_END（0xFFFFFFFF）のときは、書き込み終了処理を行います。
書き込みライブラリを読み出します。

```
/* Acquisition of command data    */
if (GetWriteData(pData, &pAddress, add_sum)){
        return;
}
if (pAddress != WRITE_END){
        /* A setup of boot status */
        BootStatus = MODE_WRITE_RUN;

        /* Program start */
        if (ErrorStatus = WriteStart(pData, pAddress)){
```

7. プログラミングガイド

ここではフラッシュマイコン標準ブートプログラムを使ったプログラムの書き方を説明します。プログラムの例、注意事項などを説明します。詳しくはハードウェアマニュアルを参照してください。

7.1 機能概要

フラッシュマイコン標準ブートプログラムは、転送ライブラリ、消去ライブラリ、書き込みライブラリで構成されています。これらの機能は以下のとおりです。

・書き込みライブラリ、消去ライブラリを指定のRAMエリアに転送
・初期設定で動作周波数を指定
・ブロック番号を指定することにより、ブロック消去
・書き込むデータと書き込み先アドレスを指定し書き込み
・ユーザブートエリア、ユーザエリアを選択

7.2 制御レジスタと制御ビット

ライブラリ転送機能、ユーザブートエリアに関する制御レジスタと制御ビットを以下に示します。

(1) 機能の選択

転送と、書き込み消去の選択はFKEYレジスタで行います。書き込み消去ライブラリ転送はFKEYレジスタをH'A5に設定し、書き込み消去処理はH'5Aに設定します。

状態	内容	機能
転送イネーブル	H'A5	ライブラリの転送が可能 SCO ビット書き込みが可能
書き込み消去イネーブル	H'5A	フラッシュメモリの書き込み消去が可能

(2) ライブラリダウンロードの起動

ライブラリを転送するときは SCO ビット(FCCS レジスタのビット 0)を 1 に設定します。

状態	内容	機能
ソースプログラムコピー ディスイーブル	0	ライブラリの RAM へのダウンロードは行いません
ソースプログラムコピーイネーブル	1	ライブラリの RAM へのダウンロードリクエストを発行します ただし、FKEY に H'A5 が書かれていて、内情 RAM 上で実行中であること ダウンロードが完了すると SCO ビットは 0 にクリアされます

(3) ライブラリの選択

ライブラリの選択は、FPCS、FECS レジスタの対応ビットを 1 に設定します。

転送プログラム	レジスタ	ビット名	ビット
書き込みプログラム	FPCS レジスタ	PPVS ビット	ビット 0
消去プログラム	FECS レジスタ	EPVB ビット	ビット 0

(4) ユーザブートエリアの選択

ユーザブートエリアの選択は FMATS レジスタを H'AA に設定します。

状態	内容	機能
ユーザエリア選択	H'AA 以外	ユーザエリア選択
ユーザブートエリア選択	H'AA	ユーザブートエリア選択

【注】RAM 上でのみ切換可能です。

(5) 転送先の選択

ライブラリ転送先のRAMアドレスをFTDARレジスタで設定します。設定が正しくないときはFTDARレジスタのビット7が1に設定されます。

転送先アドレス	設定値	機能
RAM の先頭+20k バイト	H'00	プログラムダウンロード先頭アドレスを H' FF9000 に設定
RAM の先頭+24k バイト	H'01	プログラムダウンロード先頭アドレスを H' FFA000 に設定
RAM の先頭+28k バイト	H'02	プログラムダウンロード先頭アドレスを H' FFB000 に設定
RAM の先頭+16k バイト	H'03	プログラムダウンロード先頭アドレスを H' FF8000 に設定

7.3 ライブラリの使い方

ライブラリの使い方を説明します。

(1) 転送

転送は次の手順で行います。

1.転送する書き込みライブラリ、または消去ライブラリを選択します。書き込みライブラリはFPCSレジスタのPPVSビット(ビット0)を1にセットします。
消去ライブラリはFECSレジスタのEPVBビット(ビット1)を1にセットします。
2.FTDARレジスタにRAMの転送先を指定します。
3.FKEYレジスタをH'A5に設定し、転送可能状態にします。
4.転送結果を確認できるように、RAMの転送先の先頭1バイトをH'FFに設定してください。
5.VBRレジスタをH'84000000に設定してください。
6.FCCSレジスタのSCOビット(ビット0)を1に設定します。ビット設定命令のあとにNOPを4命令置いてください。
7.RAM の先頭 1 バイトに戻り値が設定されていますので、H'00 であることを確認してください。

(2) 消去

消去は次の手順で行います。

1.消去初期設定エントリ(転送先＋32バイト)を呼び出し、動作周波数(R4)を設定します。
処理結果はR0レジスタに設定されます。
2.FKEYレジスタをH'5Aに設定し、消去書き込み可能状態にします。
3.FMATSレジスタでユーザブートエリアまたはユーザエリアを選択します。ユーザブートエリアの場合はH'AAを設定し、ユーザエリアのときはH'AA以外、たとえばH'00を設定します。FMATS設定のあとには2個のNOP命令を置いてください。
4.消去ブロック番号をR4レジスタに設定して、消去エントリ(転送先＋16バイト)を呼び出します。
5.処理結果は R4 レジスタに設定されます。

(3) 書き込み

書き込みは次の手順で行います。

1.書き込み初期設定エントリ(転送先＋32バイト)を呼び出し、動作周波数(R4)を設定します。
処理結果はR0レジスタに設定されます。
2.FKEYレジスタをH'5Aに設定し、消去書き込み可能状態にします。
3.FMATSレジスタでユーザブートエリアまたはユーザエリアを選択します。ユーザブートエリアの場合はH'AAを設定し、ユーザエリアのときはH'AA以外、たとえばH'00を設定します。FMATS設定のあとには2個のNOP命令を置いてください。
4.書き込むデータのアドレスをR4レジスタに設定し、書き込み先アドレスをR5レジスタに設定し、書き込みエントリ(転送先＋16バイト)を呼び出します。
5.処理結果はR4レジスタに設定されます。
6.書き込みエントリを呼び出します。

7.4 モジュール機能一覧

ライブラリは、転送ライブラリ、消去ライブラリ、書き込みライブラリがあります。それぞれのモジュールの機能を以下に示します。

ライブラリ	モジュール名	エントリ	機能
転送	転送開始	SCO ビットを 1 にセット	指定されたプログラム種別とプログラムコードに対応したプログラムを転送する
消去	消去初期設定	(転送先+32 バイト)	指定された動作周波数から消去時のウェイト時間を計算する
	ブロック消去	(転送先+16 バイト)	指定されたブロックを消去
書き込み	書き込み初期設定	(転送先+32 バイト)	指定された動作周波数から書き込み時のウェイト時間を計算する
	データ書き込み	(転送先+16 バイト)	指定されたデータを指定された書き込み先アドレスに書き込む

7.5 モジュール仕様

参考までに、ライブラリのモジュール仕様を以下に示します。
詳しくはハードウェアマニュアルを参照してください。

(1) 転送開始

名称	転送開始
型式	なし FCCS レジスタの SCO ビットを 1 にセットすることでライブラリを転送
機能	プログラム転送
引数	なし
入力	書き込みライブラリのとき FPCS レジスタの PPVS ビット（ビット 0）を 1 にセット 消去ライブラリのとき FECS レジスタの EPVB ビット（ビット 0）を 1 にセット FTDAR レジスタに RAM の転送先を指定 FKEY レジスタを H'A5 に設定 転送先の RAM の先頭 1 バイトを H'FF に設定 VBR レジスタを H'84000000 に設定
戻り値	なし
出力	FTDAR レジスタ TDER ビット（ビット 7）：パラメータチェックフラグ 正常終了：0 FTDAR レジスタ値異常：1（ダウンロードは中断） 転送先の RAM の先頭 1 バイト：処理結果 正常終了：H'00 FKEY レジスタ値異常：H'03 多重選択異常：H'05
処理	FPCS レジスタの PPVS ビット、FECS レジスタの EPVB ビットで点するライブラリを選択 ライブラリ選択に異常があれば処理結果を設定して戻る FKEY が H'A5 出なければ、処理結果を設定して戻る FTDAR が異常ならば、TDER ビットを 1 にセットして戻る FTDAR で指定した RAM にライブラリを転送 SCO ビットを 0 にクリア SCO ビットを設定した命令の次に戻る

(2) 消去初期設定

名称	消去初期設定
型	typedef BYTE (*InitPtr)(WORD);
機能	消去初期設定
引数	WORD：動作周波数
戻り値	処理結果 正常終了：H'00 動作周波数が異常：H'03
処理	動作周波数から消去時のウェイト時間を計算する

(3) ブロック消去

名称	ブロック消去
型	typedef BYTE (*ErasePtr)(BYTE);
機能	ブロック消去
引数	BYTE : 消去ブロック番号
戻り値	処理結果 正常終了 : H'00 消去ブロック番号が異常 : H'09 FKEY エラー : H'11 消去エラー : H'21 エラープロテクト : H'41
処理	FWE、FKEY、ブロック番号をチェックし、エラーならばエラーを設定して戻る ブロック番号からアドレスを求める ブロックに対応するアドレスを消去する 消去時エラーが発生したらエラーを設定して戻る 正常に終了すれば戻る

(4) 書き込み初期設定

名称	書き込み初期設定
型	typedef BYTE (*InitPtr)(WORD);
機能	書き込み初期設定
引数	WORD : 動作周波数
戻り値	処理結果 正常終了 : H'00 動作周波数が異常 : H'03
処理	動作周波数から書き込み時のウェイト時間を計算する

(5) 書き込み

名称	書き込み
型	typedef BYTE (*WritePtr)(BYTE *, BYTE *);
機能	書き込み処理
引数	BYTE *(第 1 引数) : 書き込むデータ格納アドレス BYTE *(第 2 引数) : 書き込み先アドレス
戻り値	処理結果 正常終了 : H'00 書き込みデータアドレスが異常 : H'03 書き込みアドレスが異常 : H'05 FKEY エラー : H'11 書き込みエラー : H'21 エラープロテクト : H'41
処理	FWE、FKEY、書き込みアドレスをチェックし、エラーならばエラーを設定して戻る データをベリファイし書き込む 書き込んだデータをベリファイし OK ならば戻る OK でなければ再度書き込む 書き込み回数を超えたら書き込み回数エラーで戻る 書き込み OK ならば戻る

ホームページとサポート窓口

ルネサステクノロジホームページ
http://japan.renesas.com/
お問合せ先
http://japan.renesas.com/inquiry
csc@renesas.com

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2008.07.15	－	初版発行

本資料ご利用に際しての留意事項

1. 本資料は、お客様に用途に応じた適切な弊社製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について弊社または第三者の知的財産権その他の権利の実施、使用を許諾または保証するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例など全ての情報の使用に起因する損害、第三者の知的財産権その他の権利に対する侵害に関し、弊社は責任を負いません。
3. 本資料に記載の製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的、あるいはその他軍事用途の目的で使用しないでください。また、輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、それらの定めるところにより必要な手続を行ってください。
4. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例などの全ての情報は本資料発行時点のものであり、弊社は本資料に記載した製品または仕様等を予告なしに変更することがあります。弊社の半導体製品のご購入およびご使用に当たりましては、事前に弊社営業窓口で最新の情報をご確認頂きますとともに、弊社ホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意下さい。
5. 本資料に記載した情報は、正確を期すため慎重に制作したものですが、万一本資料の記述の誤りに起因する損害がお客様に生じた場合においても、弊社はその責任を負いません。
6. 本資料に記載の製品データ、図、表などに示す技術的な内容、プログラム、アルゴリズムその他応用回路例などの情報を流用する場合は、流用する情報を単独で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断して下さい。弊社は、適用可否に対する責任は負いません。
7. 本資料に記載された製品は、各種安全装置や運輸・交通用、医療用、燃焼制御用、航空宇宙用、原子力、海底中継用の機器・システムなど、その故障や誤動作が直接人命を脅かしあるいは人体に危害を及ぼすおそれのあるような機器・システムや特に高度な品質・信頼性が要求される機器・システムでの使用を意図して設計、製造されたものではありません(弊社が自動車用と指定する製品を自動車に使用する場合を除きます)。これらの用途に利用されることをご検討の際には、必ず事前に弊社営業窓口へご照会下さい。なお、上記用途に使用されたことにより発生した損害等について弊社はその責任を負いかねますのでご了承願います。
8. 第7項にかかわらず、本資料に記載された製品は、下記の用途には使用しないで下さい。これらの用途に使用されたことにより発生した損害等につきましては、弊社は一切の責任を負いません。
   1) 生命維持装置。
   2) 人体に埋め込み使用するもの。
   3) 治療行為(患部切り出し、薬剤投与等)を行なうもの。
   4) その他、直接人命に影響を与えるもの。
9. 本資料に記載された製品のご使用につき、特に最大定格、動作電源電圧範囲、放熱特性、実装条件およびその他諸条件につきましては、弊社保証範囲内でご使用ください。弊社保証値を越えて製品をご使用された場合の故障および事故につきましては、弊社はその責任を負いません。
10. 弊社は製品の品質及および信頼性の向上に努めておりますが、特に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。弊社製品の故障または誤動作が生じた場合も人身事故、火災事故、社会的損害など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、誤動作防止設計などの安全設計(含むハードウエアおよびソフトウエア)およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特にマイコンソフトウエアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願い致します。
11. 本資料に記載の製品は、これを搭載した製品から剥がれた場合、幼児が口に入れて誤飲する等の事故の危険性があります。お客様の製品への実装後に容易に本製品が剥がれることがなきよう、お客様の責任において十分な安全設計をお願いします。お客様の製品から剥がれた場合の事故につきましては、弊社はその責任を負いません。
12. 本資料の全部または一部を弊社の文書による事前の承諾なしに転載または複製することを固くお断り致します。
13. 本資料に関する詳細についてのお問い合わせ、その他お気付きの点等がございましたら弊社営業窓口までご照会下さい。

© 2008. Renesas Technology Corp., All rights reserved.