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。

Application Note

RENESAS

フラッシュ開発ツールキット

アプリケーションノート（応用編）

ユーザプログラムモード（H8/3694F）

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2005.10

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。

ルネサスフラッシュ開発ツールキット
（Flash Development Toolkit）

アプリケーションノート（応用編）

ユーザプログラムモード（H8/3694F）

Rev.1.00

株式会社　ルネサステクノロジ

目次

(注)本アプリケーションノートは、
H8S,H8/300 SERIES C/C++ COMPILER (V.6.01.00.009) を使用して動作確認を致しました。

1. はじめに

このアプリケーションノートは、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 H8/3694F（H8/300H Tiny シリーズ）ユーザプログラムモード（ユーザモード）の使用法を以下のとおり説明します。

（1） ブートモード（ユーザエリアへの書き込み）

（2） ユーザプログラムモード（ユーザモード）

これらの説明から、ブートモードと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を理解してください。

このアプリケーションノートでは、ブートモードの制御プログラムを参考にして作成したサンプルプログラムの説明をしています。サンプルプログラムは内蔵フラッシュメモリの書き込み消去の処理をしています。ユーザプログラム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、このサンプルプログラムを参照してください。

2. H8/3694F（H8/300H Tiny シリーズ）

2.1 フラッシュメモリ構成

H8/3694Fのフラッシュメモリ版には、32kバイトのフラッシュメモリが内蔵されています。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ます。このアプリケーションノートでは、制御プログラムを格納したエリアをブートエリア、フラッシュメモリをユーザエリアと呼ぶことにします。フラッシュメモリ構成を 表 2-1に示します。

表 2-1 フラッシュメモリ構成

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	—	—
ユーザエリア	フラッシュメモリ	32k バイト	5 ブロック 1k バイト×4 28k バイト×1

2.2 プログラミングモード

フラッシュメモリの書き込み／消去を行うためのモードとしてオンボードで書き込み／消去ができるブートモードとPROM ライタで書き込み／消去を行うライタモードが用意されています。このほかユーザプログラムモードでもオンボードで書き込み／消去を行うことが可能です。リセット状態からリセットスタートするとH8/3694F はTEST 端子、$\overline{\text{NMI}}$ 端子およびポートの入力レベルによって 表 2-2のように異なるモードへ遷移します。各端子の入力レベルは少なくともリセット解除の 4 ステート前に確定させる必要があります。

ブートモードに遷移すると、LSI 内部に組み込まれているブートプログラムが起動します。ブートプログラムは SCI3 を経由して外部に接続されたフラッシュ開発ツールキットから書き込み制御プログラムを内蔵 RAM に転送し、フラッシュメモリを全面消去したうえで書き込み制御プログラムを実行します。オンボード状態での初期書き込みや、ユーザプログラムモードで書き込み／消去ができなくなった場合の強制復帰等に使用できます。

ユーザプログラムモードではユーザが用意した書き込み／消去プログラムに分岐することで任意のブロックを消去し書き換えることができます。

詳しくは、ハードウェアマニュアルを参照してください。

表 2-2 プログラミングモード選択方法

リセット解除後の LSI の状態		TEST	$\overline{\text{NMI}}$	P85	PB0	PB1	PB2
オンボード	ユーザプログラムモード	0	1	X	X	X	X
プログラミングモード	ブートモード	0	0	1	X	X	X
ライタモード		1	X	X	0	0	0

【注】X：Don’t care

2.3 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の 2 種類があります。オンボードプログラミングモードを 表 2-3に示します。

表 2-3 オンボードプログラミングモード

項目	ブートモード	ユーザプログラムモード
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。
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）
書き込み／消去可能エリア	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
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

【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。

ブートモードでは、いったんユーザエリアが全面消去されます。その後、コマンド方式でフラッシュメモリ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。

3. 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 F-ZTAT 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 F-ZTAT 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。

[注]: F-ZTAT(Flexible - Zero Turn Around Time)は、株式会社ルネサステクノロジの商標です。

3.1 おもな機能

- 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 3.4 ユーザーズマニュアル」を参照してください。

フラッシュ開発ツールキット グラフィカルユーザインタフェースの画面を 図 3-1に示します。

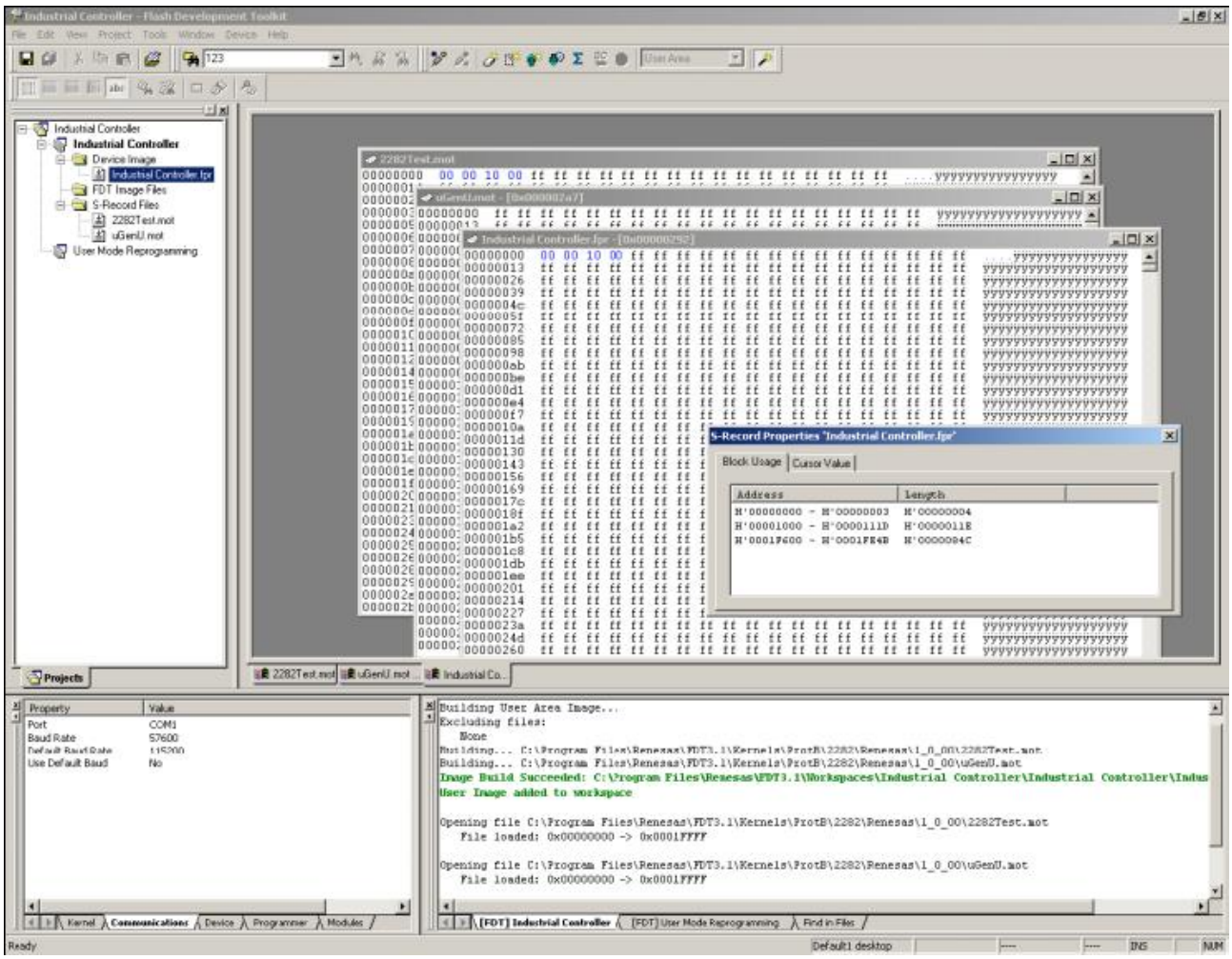


図 3-1 フラッシュ開発ツールキット グラフィカルユーザインタフェース

4. フラッシュ開発ツールキットの操作方法

4.1 アダプタボードの接続

F-ZTAT*マイコンオンボード書き込み用アダプタボード HS0008EAUF1H（以下、アダプタボードと記載します）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 F-ZTAT 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。

アダプタボードの接続を 図 4-1に示します。

［注］F-ZTAT(Flexible - Zero Turn Around Time)は、株式会社ルネサステクノロジの商標です。

［注］FDM（Flash Development Module）はアダプタボードの古い名称です。

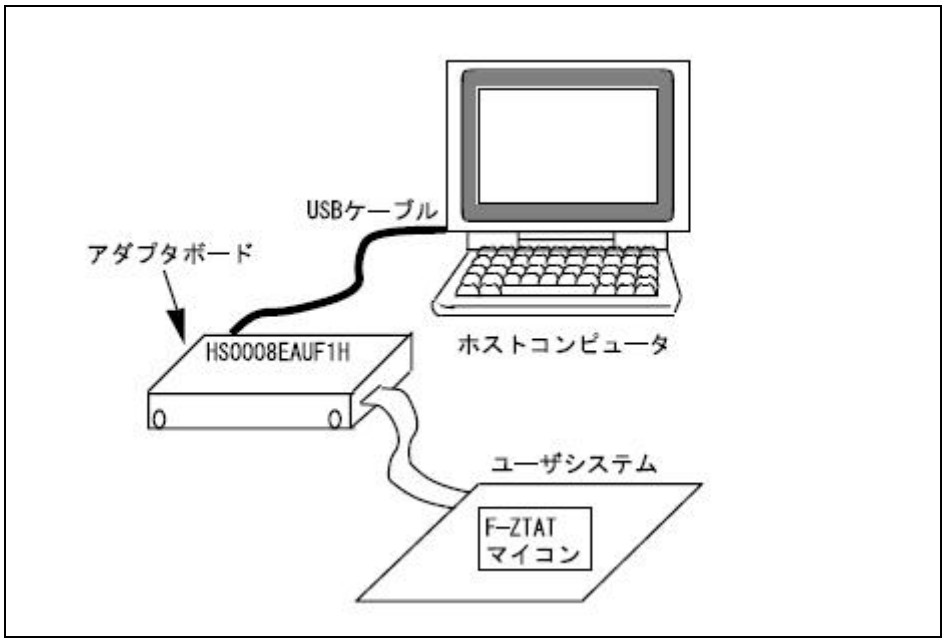


図 4-1 アダプタボードの接続

アダプタボードとユーザシステムとを接続するユーザインタフェースケーブル信号対応表を以下に示します。

表 4-1 HS0008EAUF1H ユーザインタフェースケーブル信号対応表

No	信号名	No	信号名
1	$\overline{\text{RES}}$	2	GND
3	FWx	4	GND
5	MD0	6	GND
7	MD1	8	GND
9	MD2（IO0）	10	GND
11	MD3（IO1）	12	GND
13	MD4（IO2）	14	GND
15	RXD(ユーザ側 TXD)	16	GND*1
17	TXD(ユーザ側 RXD)	18	VIN(Vcc または PVcc)*2
19	SCK（NC）	20	VIN(PVcc)*2

*1:No.16 ピンはユーザシステムが正しく接続されているかを認識するために必ずＧＮＤ接続してください。

*2:Vcc,PVcc を持つデバイスの場合は、ユーザインタフェースコネクタの VIN 端子に Vcc または PVcc (18 ピン),PVcc (20 ピン)をそれぞれ必ず供給してください。また、Vcc = PVcc の条件で使用する際および Vcc,PVcc の混在が無いデバイスを使用の場合は VIN 端子 Vcc または PVcc (18 ピン),PVcc(20 ピン) 2 本とも Vcc を必ず供給してください。

4.1.1 H8/3694F ユーザシステム

このアプリケーションノートでは、H8/3694F ユーザシステムとして、H8/3694F 評価用の株式会社北斗電子製 CPU ボード HSB タイプ F の HSB8/3694F CPU ボードを使って、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

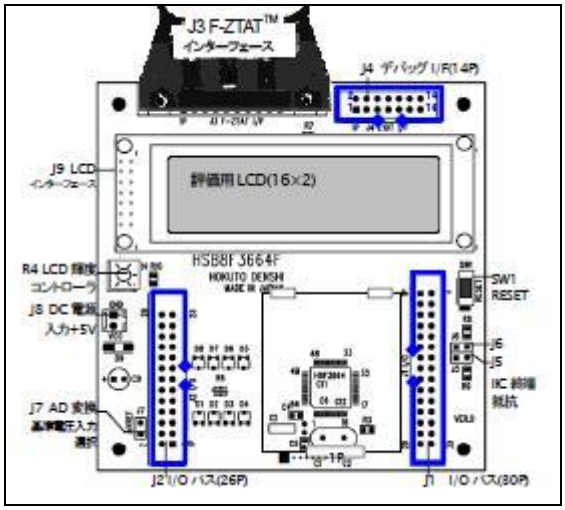


図 4-2 H8/3694F 評価用 CPU ボード

4.1.2 アダプタボードの接続

H8/3694Fとルネサス製アダプタボード(HS0008EAUF1H)の接続例を 図 4-3に示します。プルアップおよびプルダウンの抵抗値は参考値ですので、ユーザシステムにてご評価頂けるようお願いします。

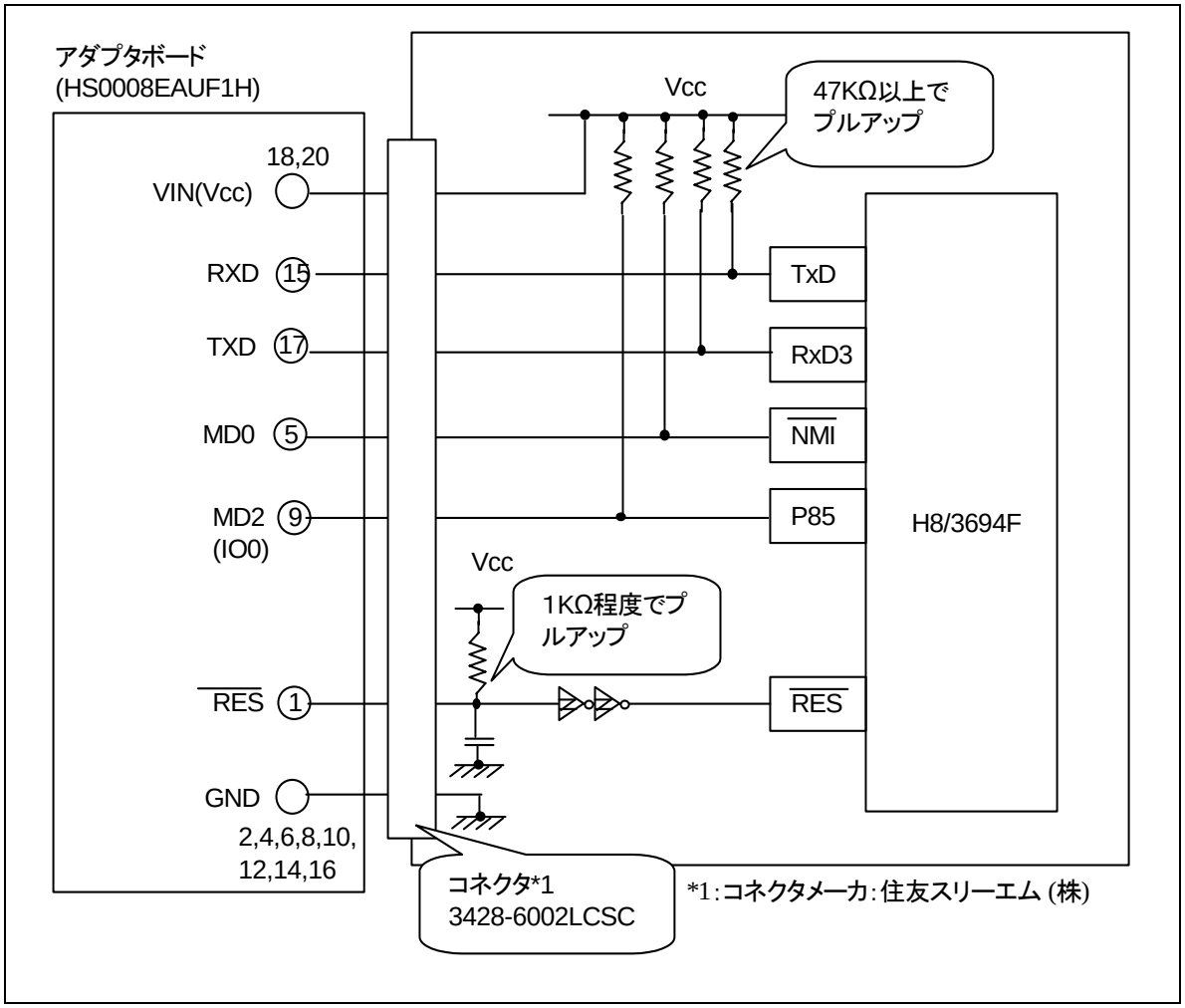


図 4-3 H8/3694F とアダプタボードの接続例

4.1.3 アダプタボードの端子の設定

H8/3694Fユーザシステムとルネサス製アダプタボード(HS0008EAUF1H)の接続したときのブートモード時の端子の設定例を 表 4-2に示します。

表 4-2 H8/3694F とアダプタボードの端子の設定例(ブートモード時)

ピン番号	アダプタボード端子	デバイス端子	入出力	出力レベル
1	$\overline{\text{RES}}$	$\overline{\text{RES}}$	出力（デフォルト）	アダプタボード
3	FWx	NC	NC	—
5	MD0	$\overline{\text{NMI}}$	出力	ロー（0）
7	MD1	NC	NC	—
9	MD2（IO0）	P85	出力	ハイ（1）
11	MD3（IO1）	NC	NC	—
13	MD4（IO2）	NC	NC	—
15	RXD	TXD	入力（デフォルト）	アダプタボード
17	TXD	RXD	出力（デフォルト）	アダプタボード
19	SCK（NC）	NC	NC（デフォルト）	—

［注］NC： No Connection（接続なし）を意味します。

4.1.4　　H8/3694F ユーザシステムのプログラミングモード

H8/3694F ユーザシステムは北斗電子製の HSB8/3694F CPU ボードを使います。このユーザシステムはプログラミングモードを選択するために、アダプタボードの MD2(IO0)、MD0 端子から信号を出力します。

ブートモードを選択するときは、MD2(IO0)を出力ハイ（1）に設定し、MD0 を出力ロー（0）に設定します。ユーザプログラムモードを選択するときは、MD2(IO0)を入力に設定し、MD0 をハイ（1）に設定します。、

プログラミングモード選択方法を 表 4-3に示します。

表 4-3 プログラミングモード選択方法

リセット解除後の LSI の状態		TEST	$\overline{\text{NMI}}$	P85	PB0	PB1	PB2
		—	MD0	MD2(IO0)	—	—	—
オンボード プログラミングモード	ユーザプログラムモード	0	1	X	X	X	X
	ブートモード	0	0	1	X	X	X
ライタモード		1	X	X	0	0	0

【注】X：Don’t care

4.1.5　　H8/3694F ユーザシステムのシリアル入出力

H8/3694F ユーザシステム HSB8/3694F CPU ボードのシリアル入出力は、ブートモードでは F-ZTAT インターフェース（J3）に接続されています。しかし、ユーザプログラミングモードでは、IO バス（26P）（J2）に接続されます。そのため、ユーザプログラミングモードで、フラッシュ開発ツールキットからアダプタボードを使ってシリアル送受信するためには、F-ZTAT インターフェース（J3）と IO バス（26P）（J2）の TXD（J3-15、J2-20）、RXD（J3-17、J2-21）をそれぞれ接続する必要があります。

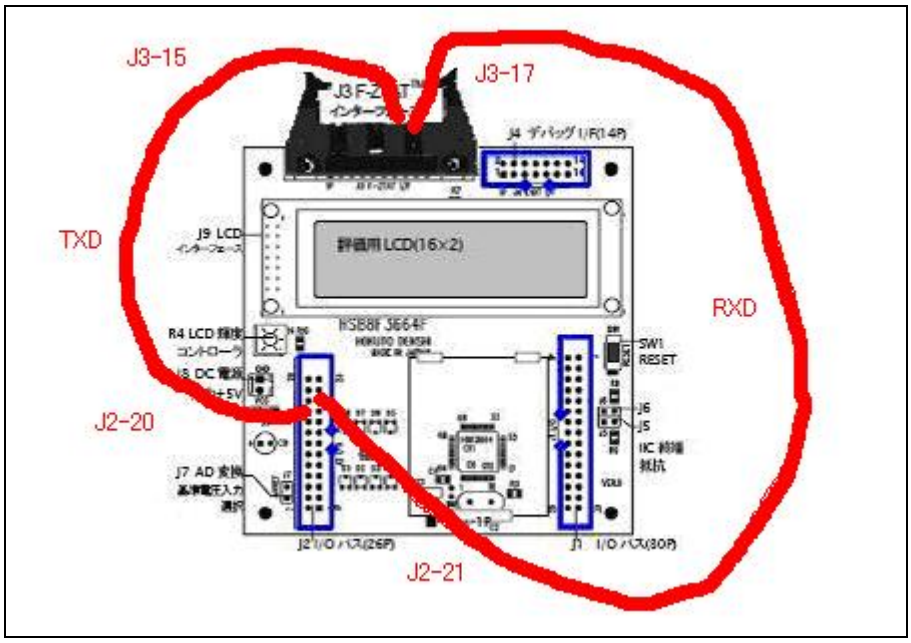


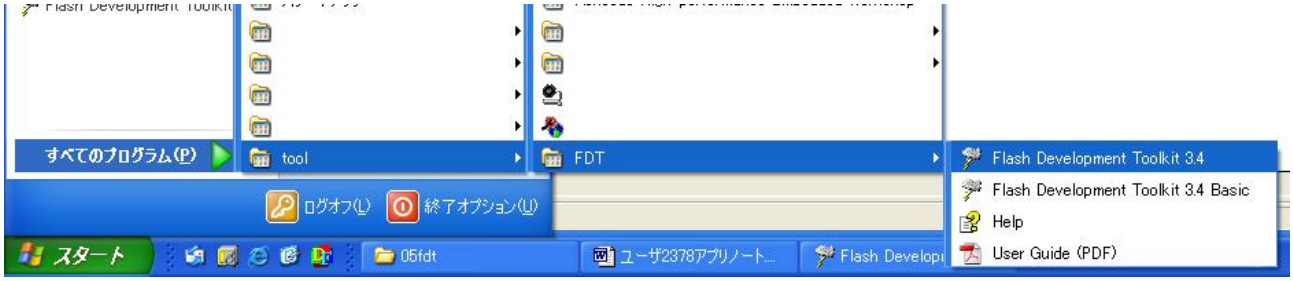
図 4-4 H8/3694F ユーザプログラミングモードのシリアル送受信

4.2　　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。

4.2.1　　フラッシュ開発ツールキットの起動

すべてのプログラムから‘Flash Development Toolkit 3. 4 ’を選択します。

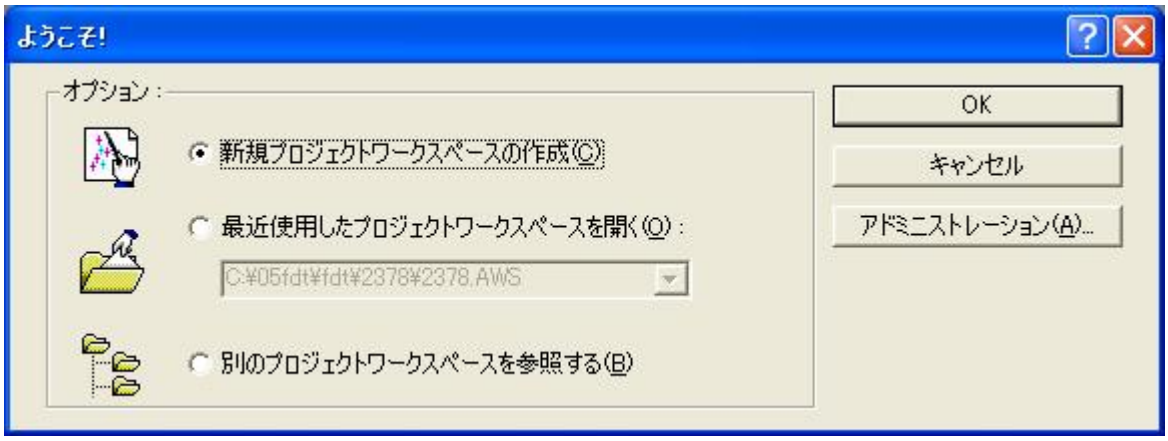


4.2.2　　オプションの選択

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(C)’を選択します。

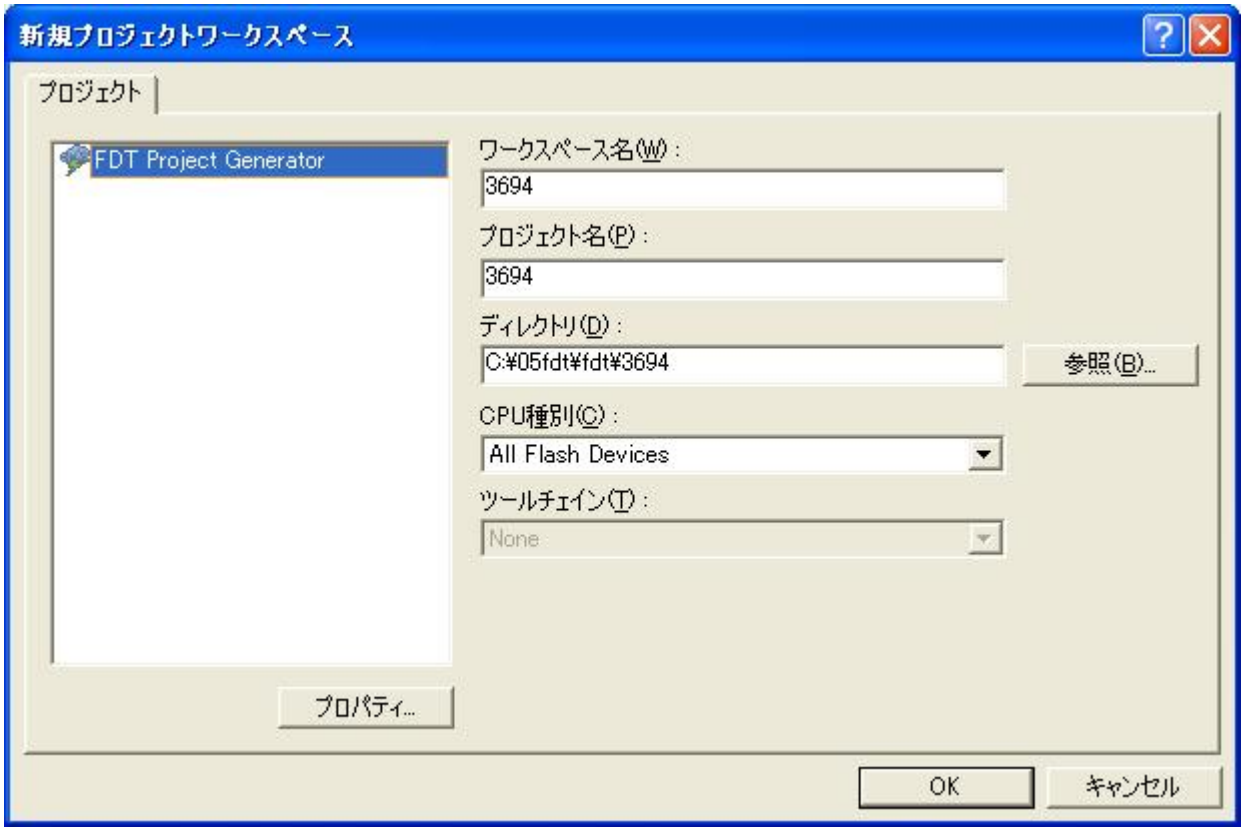
２回目以降の起動では、前回選択したデバイスとポートの情報は保持されますので、‘最近使用したプロジェクトワークスペースを開く(O):’選択します。



選択が終了したら‘OK’をクリックします。

4.2.3　　新規プロジェクトワークスペースの設定

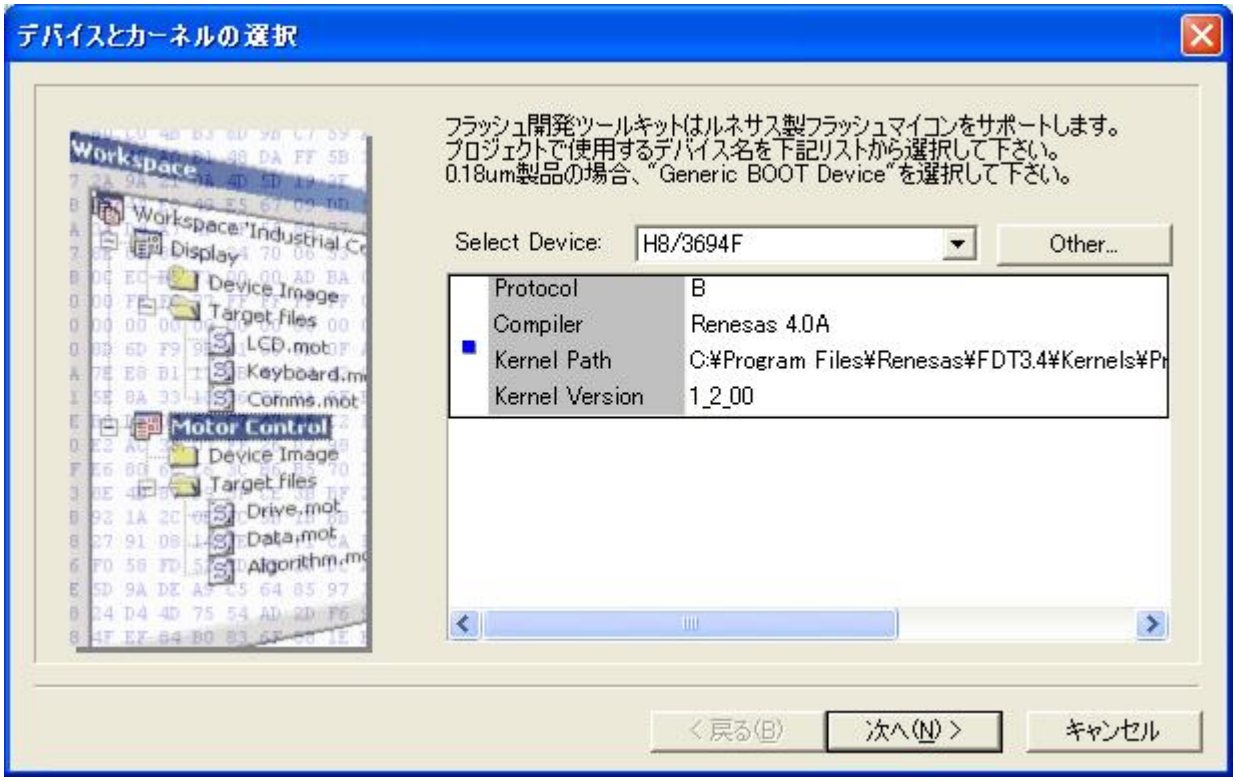
新規プロジェクトワークスペースの設定します。‘参照(B)’ディレクトリを選択し、ワークスペース名にデバイスを指定します。必要ならば、プロジェクト名を指定します。ここでは、ワークスペース名とプロジェクト名を同じに指定します。



選択が終了したら‘OK’をクリックします。

4.2.4 デバイスとカーネルの選択

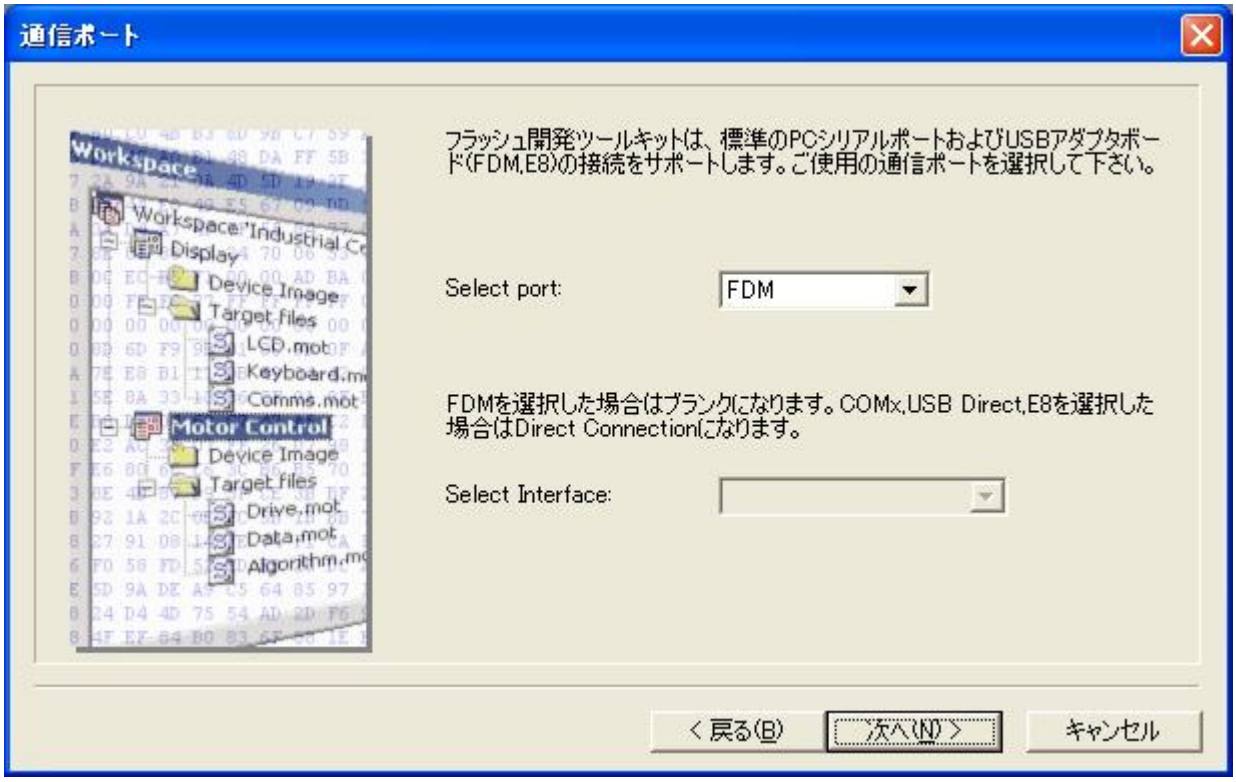
プルダウンメニューから対象デバイスを選択します。ここでは H8/3694F を選択します。



選択が終了したら‘次へ(N)>’をクリックします。

4.2.5　　通信ポートの選択

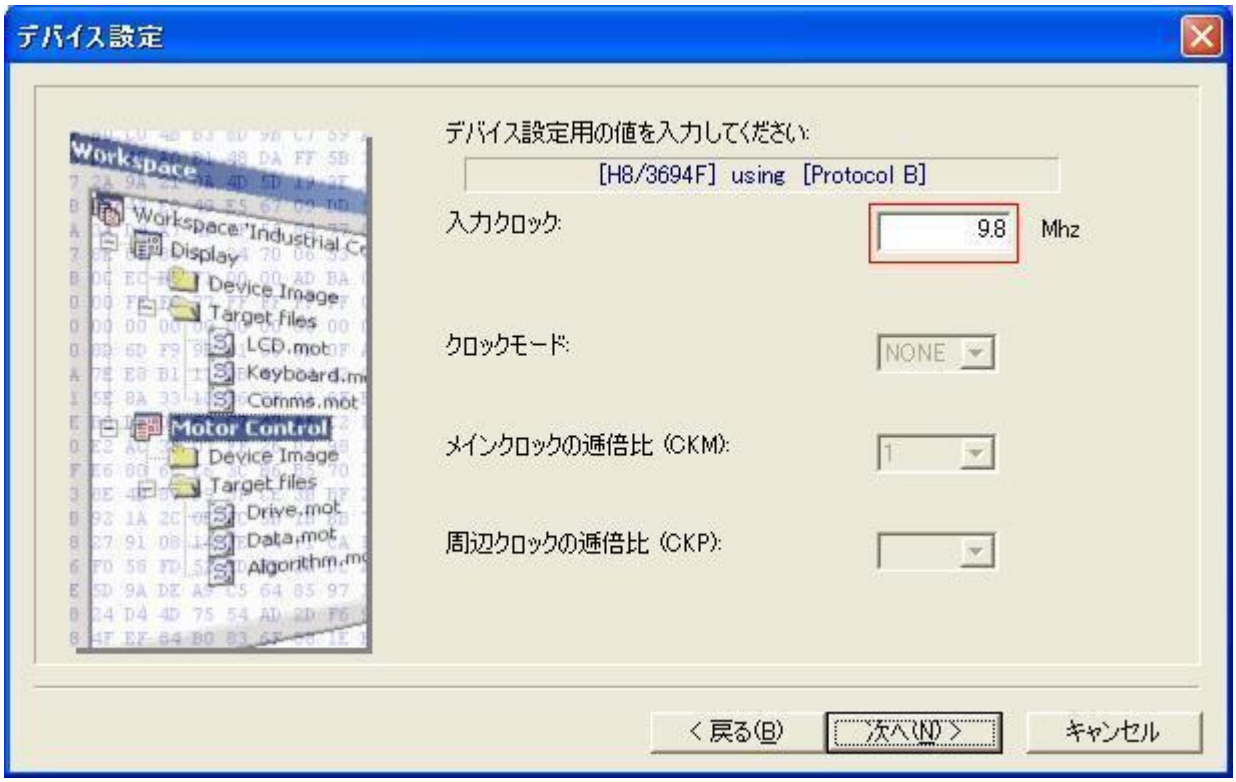
プルダウンメニューからアダプタボード（FDM）を選択します。



選択が終了したら‘次へ(N)>’をクリックします。

4.2.6　　デバイス設定（入力クロックの設定）

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。たとえば、9.8MHz を入力します。

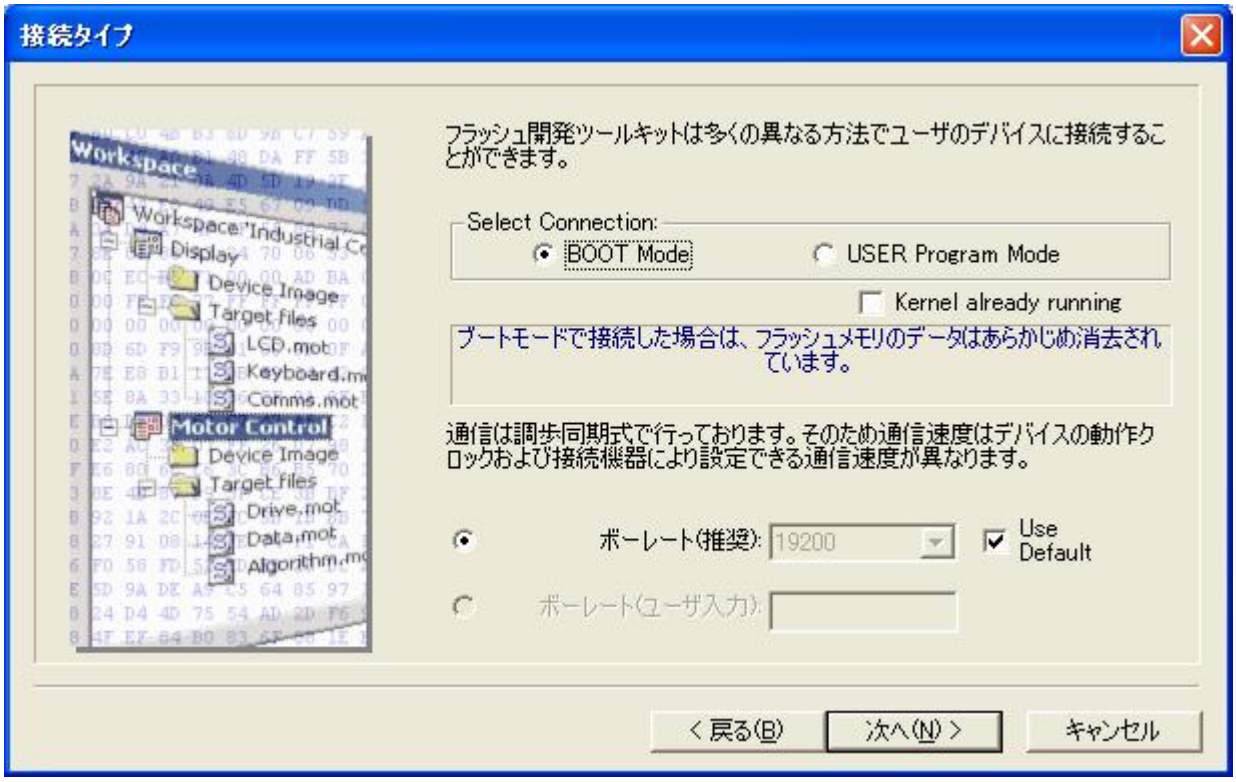


選択が終了したら‘次へ(N)＞’をクリックします。

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有効数字 3 桁で入力してください。入力クロックと動作周波数（PLL 出力）とは異なります。

4.2.7　　接続タイプの選択（通信速度）

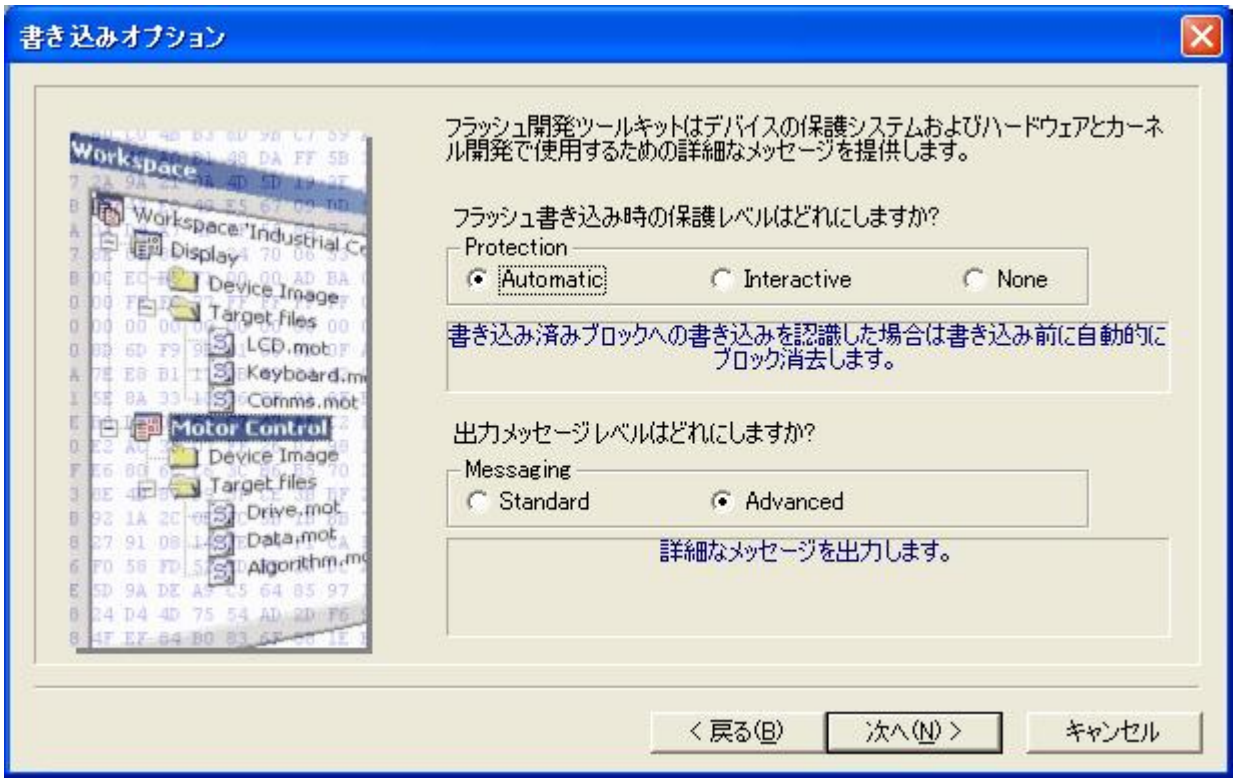
ボーレートを設定します。たとえば Use Default を選択します。



選択が終了したら‘次へ(N)>’をクリックします。

4.2.8　　書き込みオプションの選択（保護レベル、出力メッセージレベル）

保護レベル、出力メッセージレベルを選択します。たとえば、保護レベルは Automatic、出力メッセージレベルは Advanced を選択します。

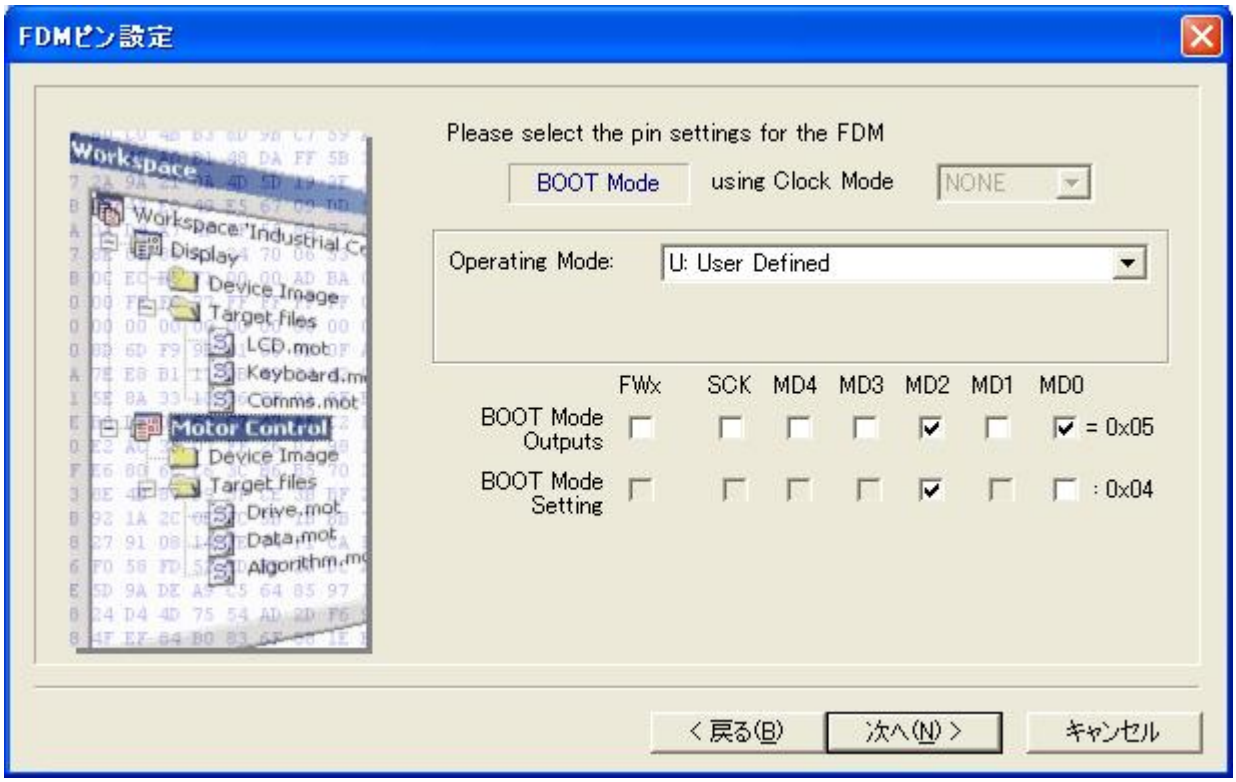


選択が終了したら‘次へ(N)>’をクリックします。

4.2.9 アダプタボードピン設定

ブートモードのアダプタボード（FDM）ピンを設定します。

H8/3694F のブートモードでは、P85 を出力ハイ（1）、$\overline{\text{NMI}}$ を出力ロー（0）に設定します。H8/3694F のユーザシステムは、MD2（IO0）は P85 に、MD0 は $\overline{\text{NMI}}$ に接続されています。そのため、MD2（IO0）を出力ハイ（1）、MD0 を出力ロー（0）に設定します。FEW 端子はないので設定の必要はありません。



選択が終了したら‘次へ(N)>’をクリックします。

H8/3694Fとルネサス製アダプタボード(HS0008EAUF1H)の接続例を 図 4-5に示します。プルアップおよびプルダウンの抵抗値は参考値ですので、ユーザシステムにてご評価頂けるようお願い申し上げます。

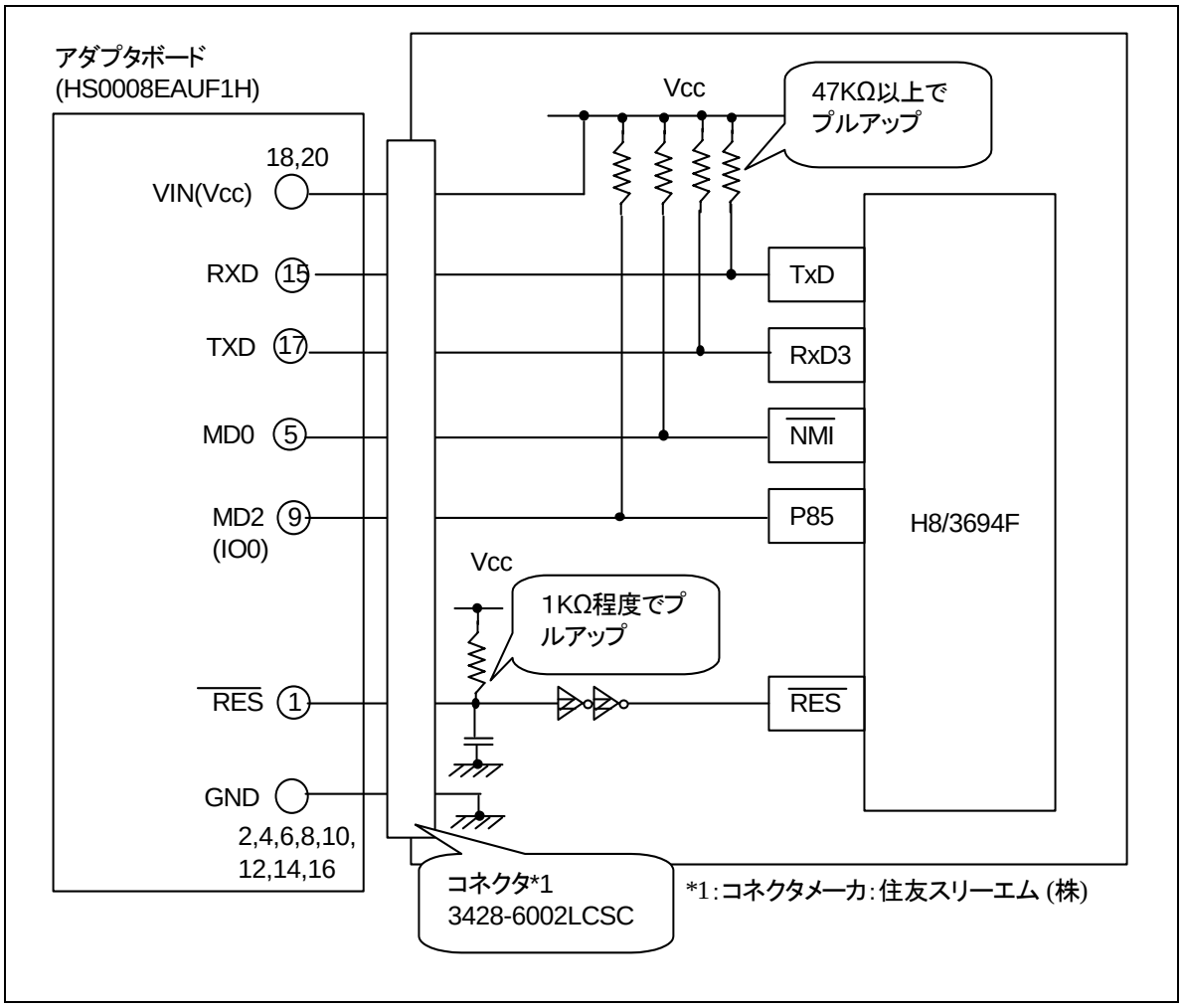


図 4-5 H8/3694F とアダプタボードの接続例

H8/3694Fユーザシステムとルネサス製アダプタボード(HS0008EAUF1H)の接続したときのブートモード時の端子の設定例を 表 4-4に示します。

表 4-4 H8/3694F とアダプタボードの端子の設定例(ブートモード時)

ピン番号	アダプタボード端子	デバイス端子	入出力	出力レベル
1	$\overline{\text{RES}}$	$\overline{\text{RES}}$	出力（デフォルト）	アダプタボード
3	FWx	NC	NC	—
5	MD0	$\overline{\text{NMI}}$	出力	ロー（0）
7	MD1	NC	NC	—
9	MD2（IO0）	P85	出力	ハイ（1）
11	MD3（IO1）	NC	NC	—
13	MD4（IO2）	NC	NC	—
15	RXD	TXD	入力（デフォルト）	アダプタボード
17	TXD	RXD	出力（デフォルト）	アダプタボード
19	SCK（NC）	NC	NC（デフォルト）	—

［注］NC： No Connection（接続なし）を意味します。

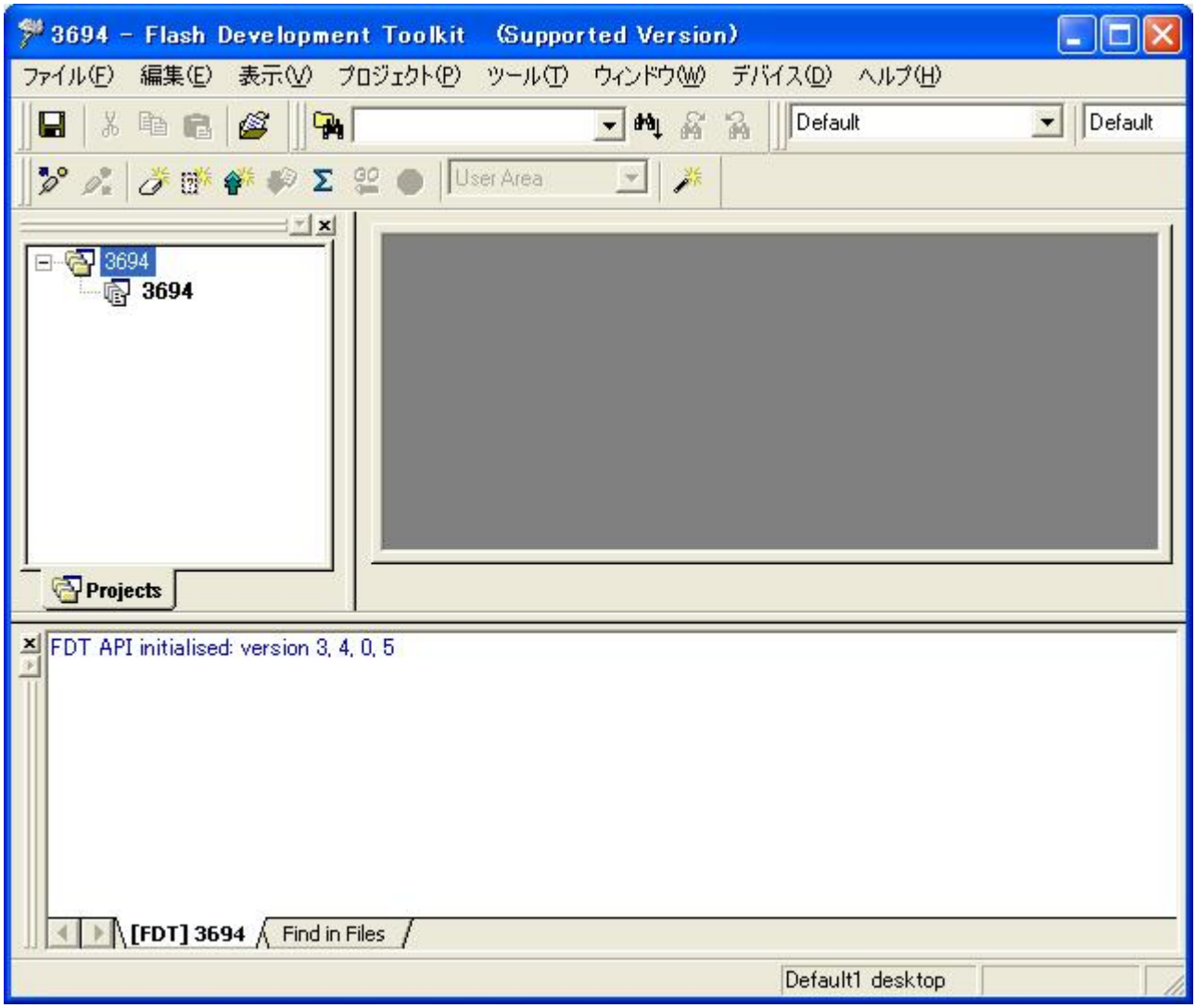
4.2.10　　リセットモードピン設定

デバイスをリセットモードで再起動したときのアダプタボードのピンを設定します。ここでは必要ありません。

選択が終了したら‘完了’をクリックします。

4.2.11　　設定の完了

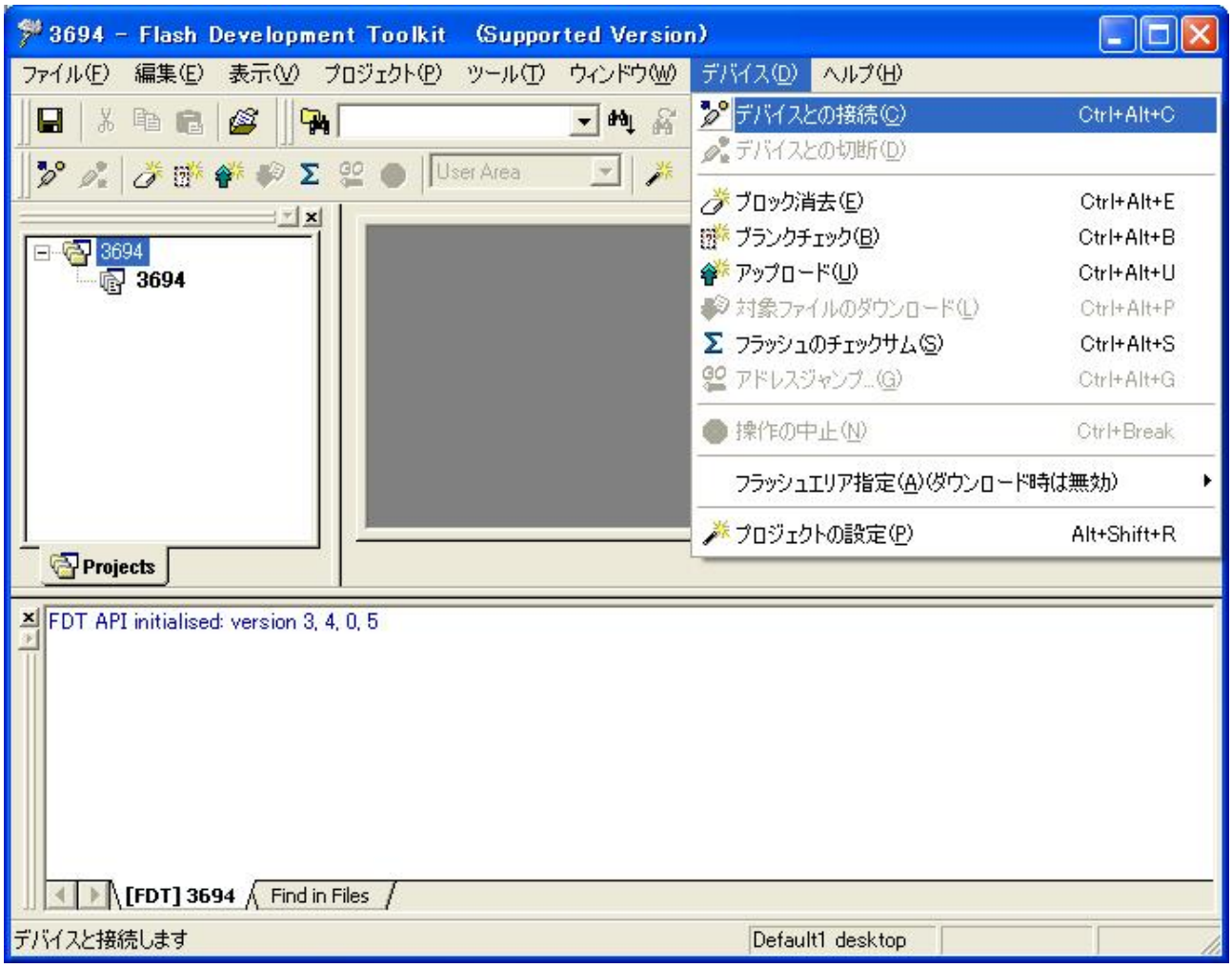
ブートモードで、H8/3694F ボードがフラッシュ開発ツールキットに設定されました。



4.2.12　　デバイスとの接続

アダプタボード（FDM）をパソコンに接続し、H8/3694F のボードをアダプタボードに接続し、電源を入れてください。

接続が完了したら、‘デバイス(D)’からプルダウンメニューを開き、‘デバイスとの接続(C)’をクリックします。



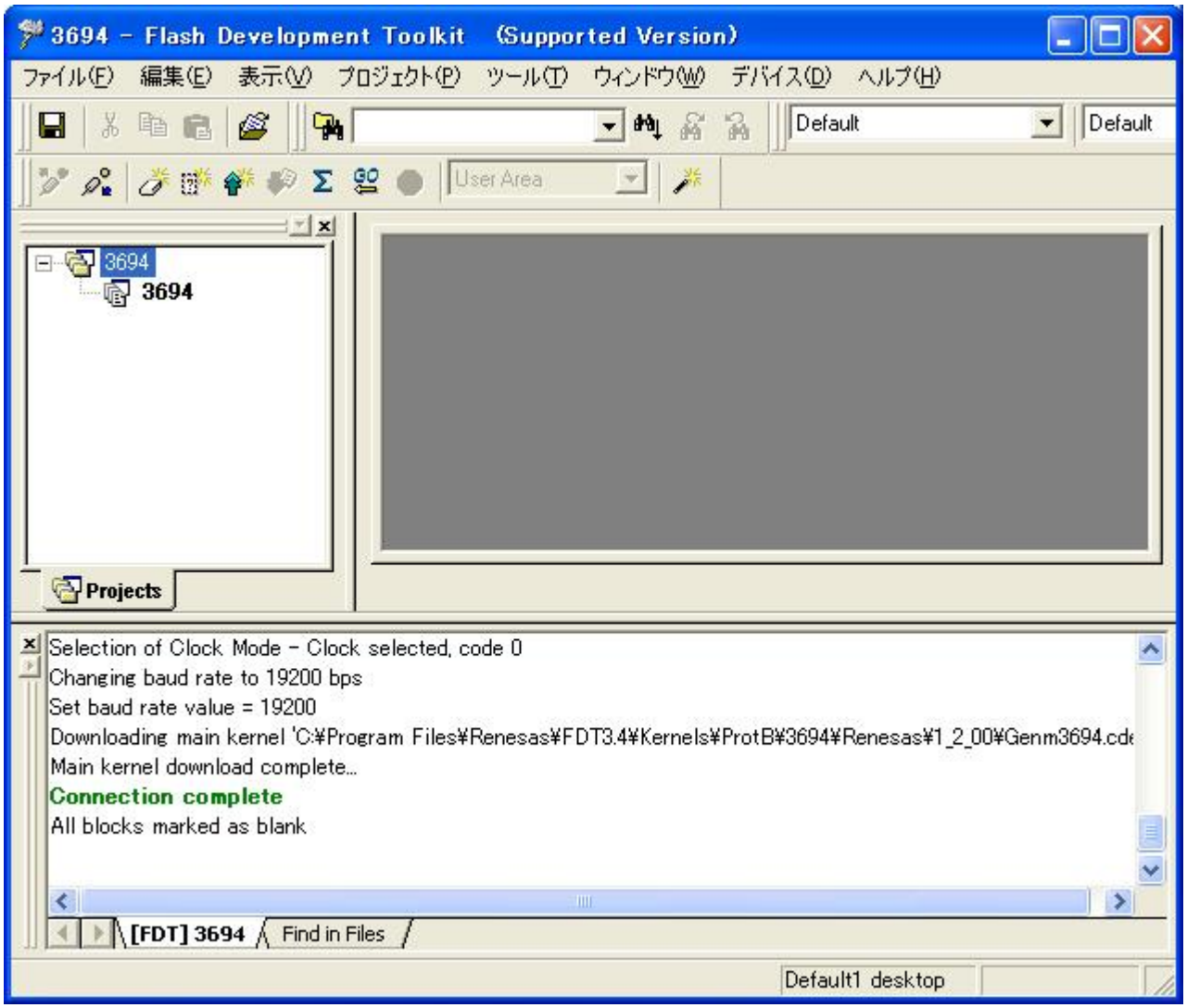
アダプタボード（FDM）を選択します。

選択が終了したら‘OK’をクリックします。

4.2.13　　接続の完了

ブートモードで、H8/3694F 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、ユーザエリアの内容は消去されています。



4.3　　ブートモード（ユーザエリアへの書き込み）

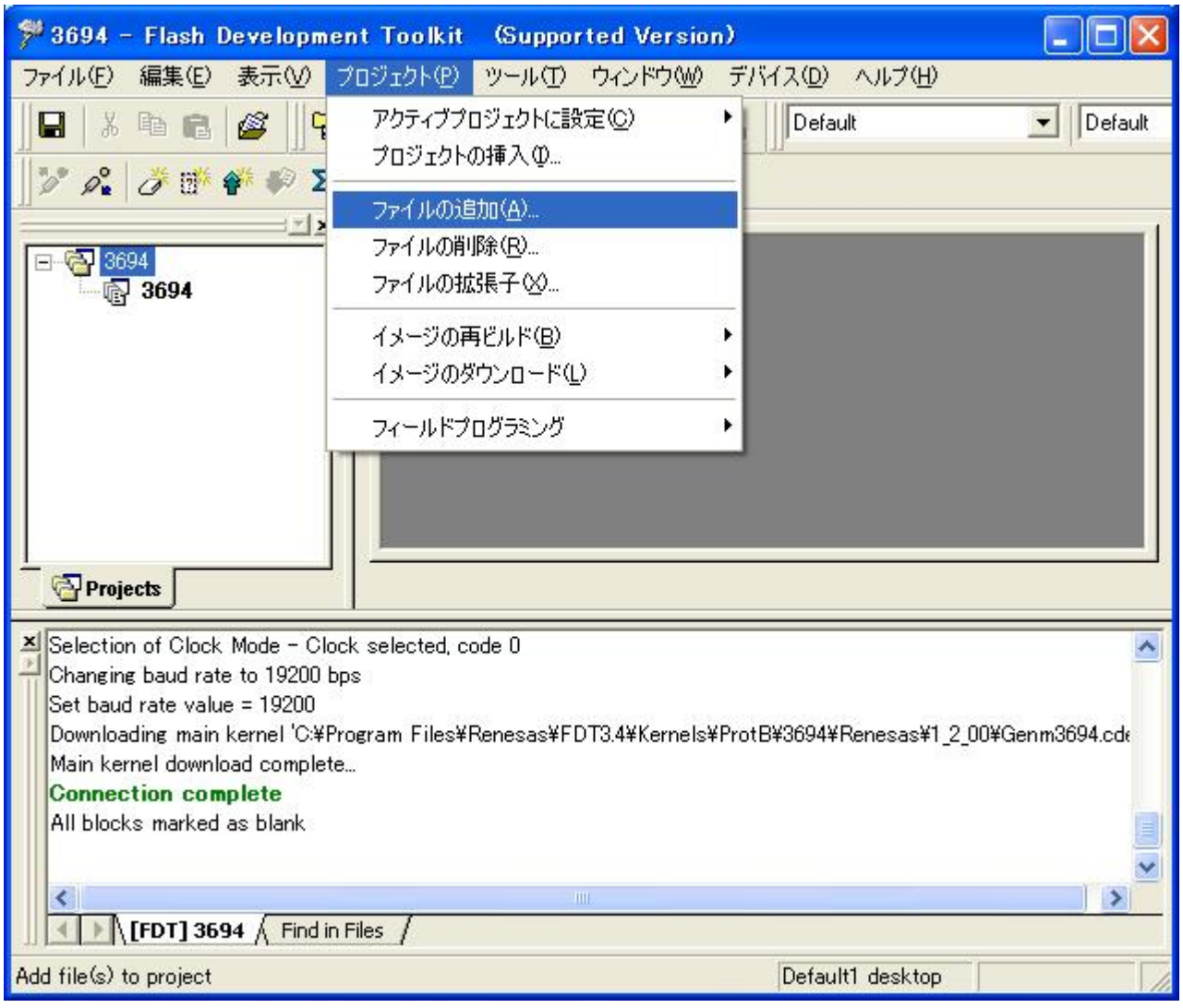
ブートモードでユーザエリアへプログラムを書き込みます。書き込むプログラムは、フラッシュ開発ツールキットに付属しているサンプルテストプログラムの 3694Test.motとuGenU.motファイル（Sタイプファイル）です。このプログラムは、ビットレートを周波数に対応して修正しなくてはなりません。ビットレートの修正は、「7.1.1　ビットレートの設定（GenTest.h)」を参照してください。

プログラムは、フラッシュ開発ツールキットの Renesas¥FDT3.4¥Kernels¥ProtB フォルダにあります。フラッシュ開発ツールキットのプログラムを Program Files フォルダにインストールしたときのフォルダ詳細は次のとおりです。

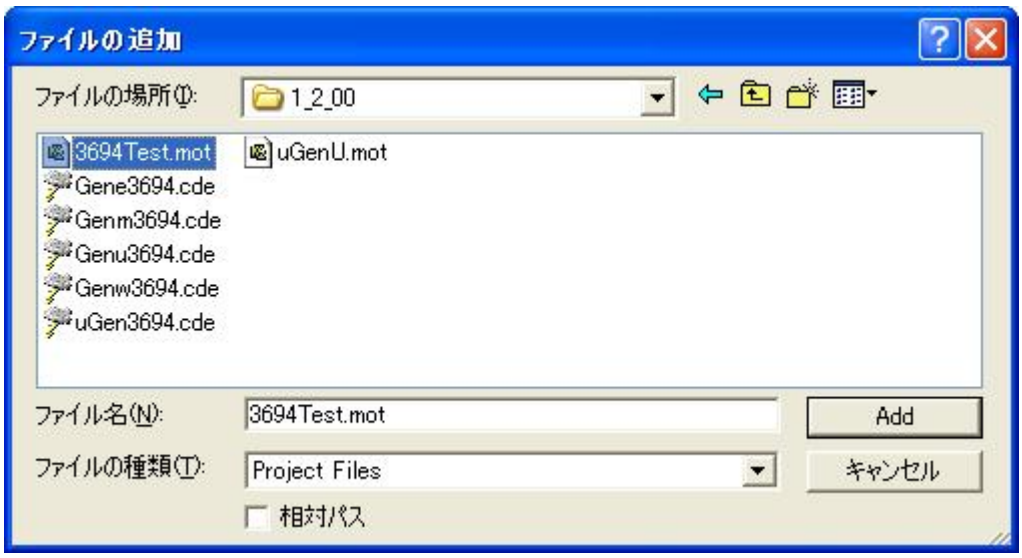
C:¥Program Files¥Renesas¥FDT3.4¥Kernels¥ProtB¥3694¥Renesas¥1_2_00

4.3.1　　ファイルの選択

書き込みファイルを選択するため、‘プロジェクト(P)’のプルダウンメニューから‘ファイルの追加(A)…’を選択します。

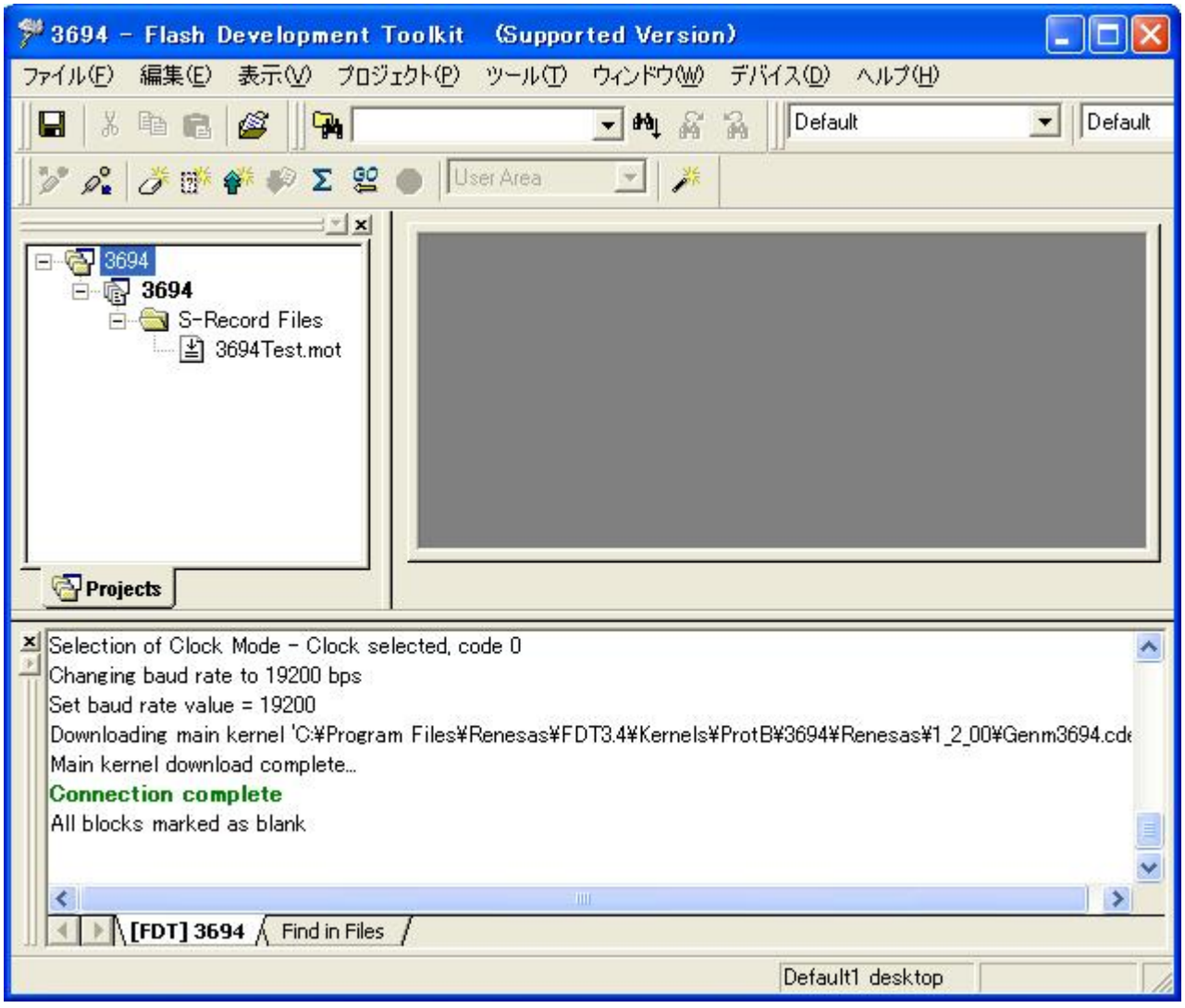


ファイルの追加ダイアログボックスから‘3694Test.mot’ファイルを追加します。

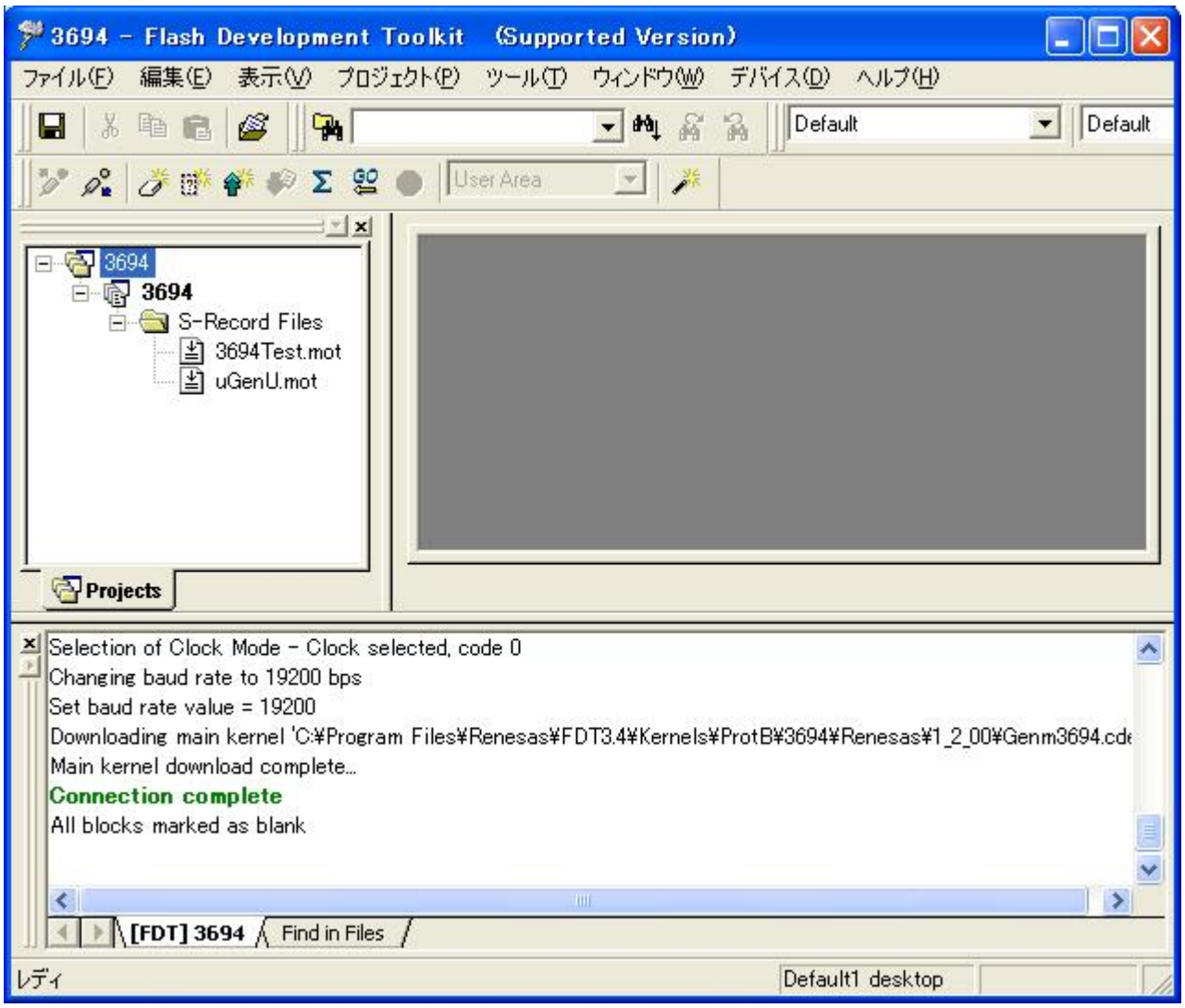


選択が終了したら‘Add’をクリックします。

プロジェクトに‘3694Test.mot’ファイルが追加されます。

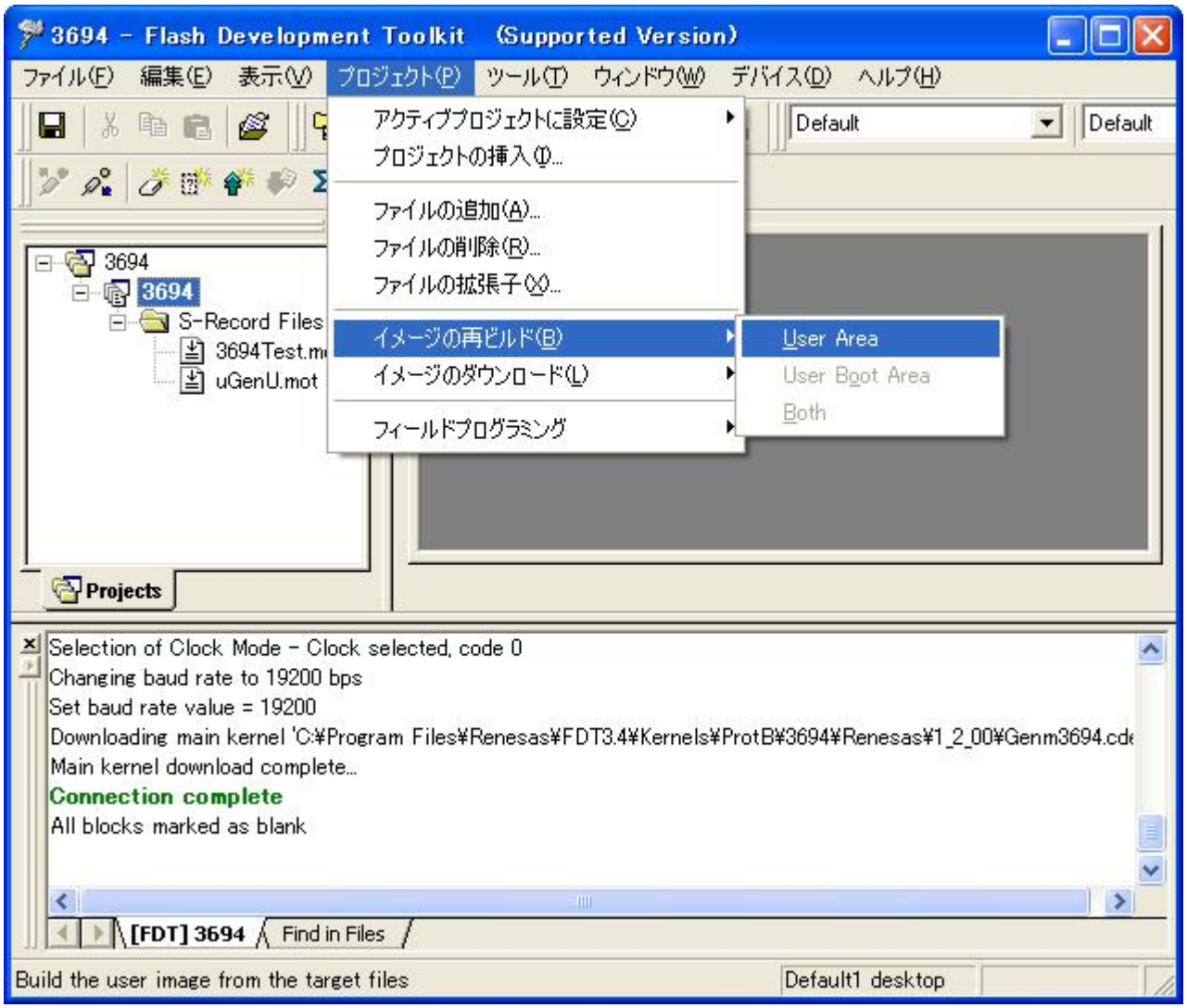


同じように‘uGenU.mot’を追加します。

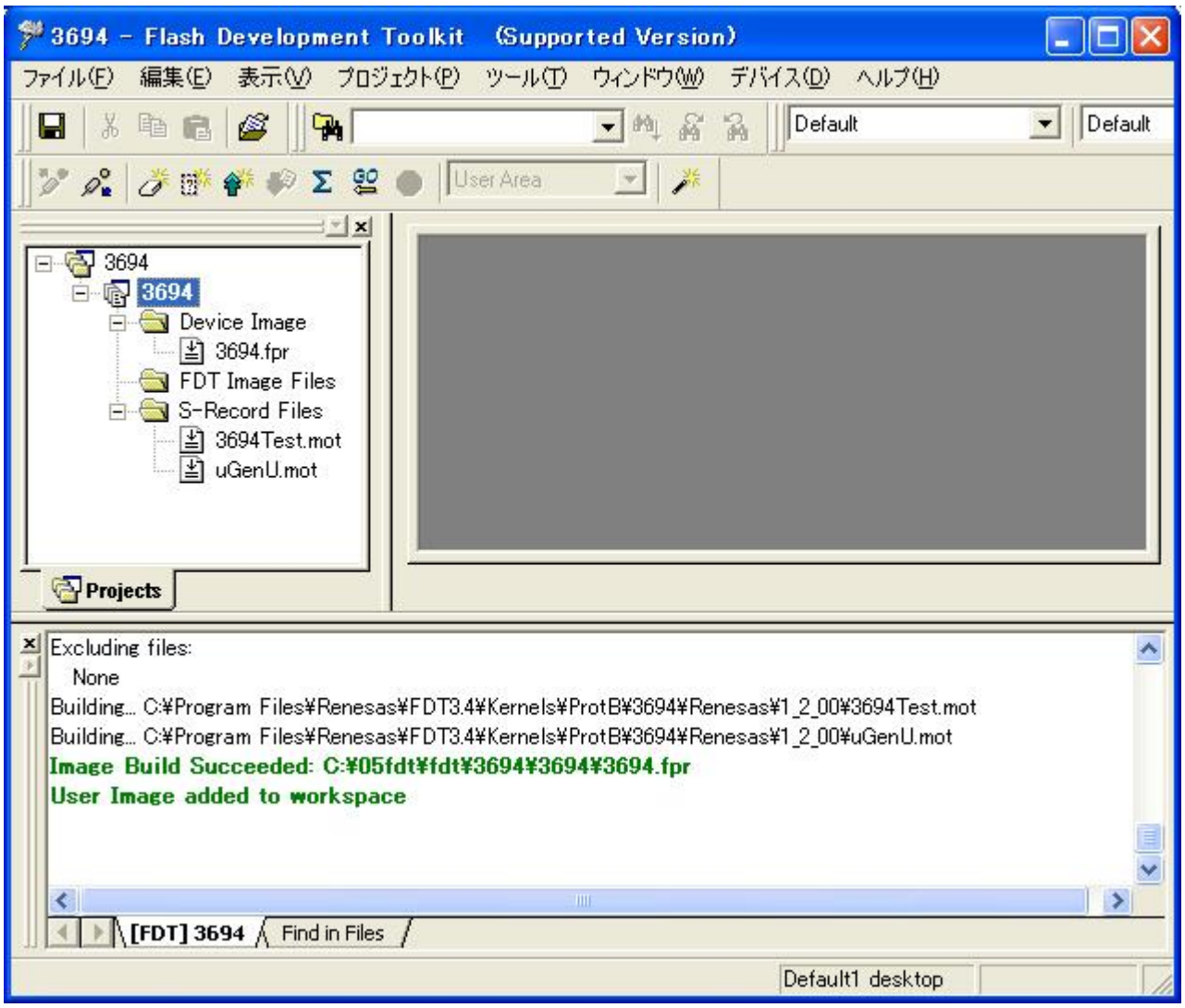


4.3.2　　イメージのビルド

書き込みファイルが複数あるので、ユーザエリアデバイスイメージをビルドします。‘プロジェクト(P)’のプルダウンメニューから‘イメージの再ビルド(B)’を選択し、‘User　Area’を選択します。



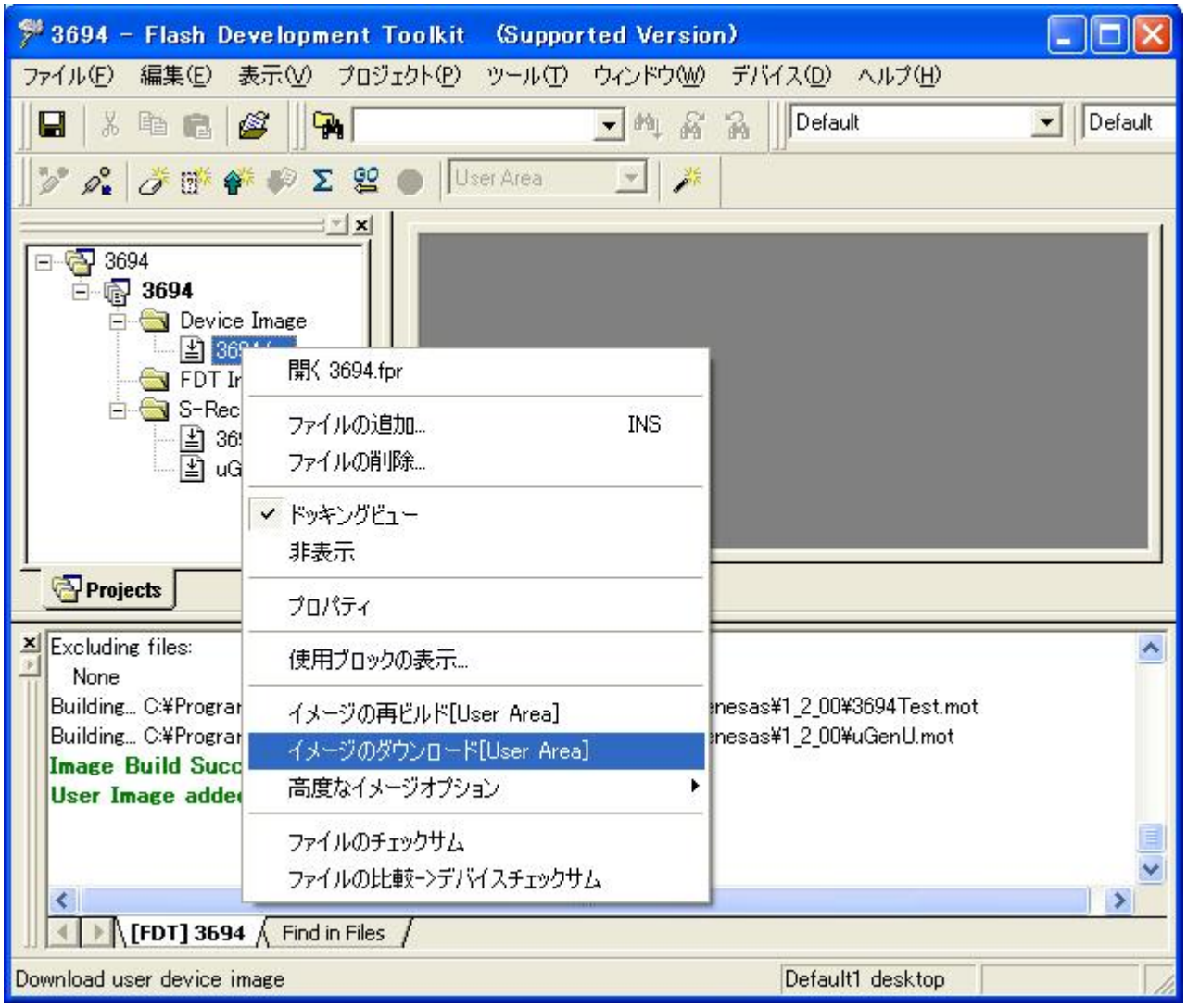
イメージファイル‘3694.fpr’が作成されます。



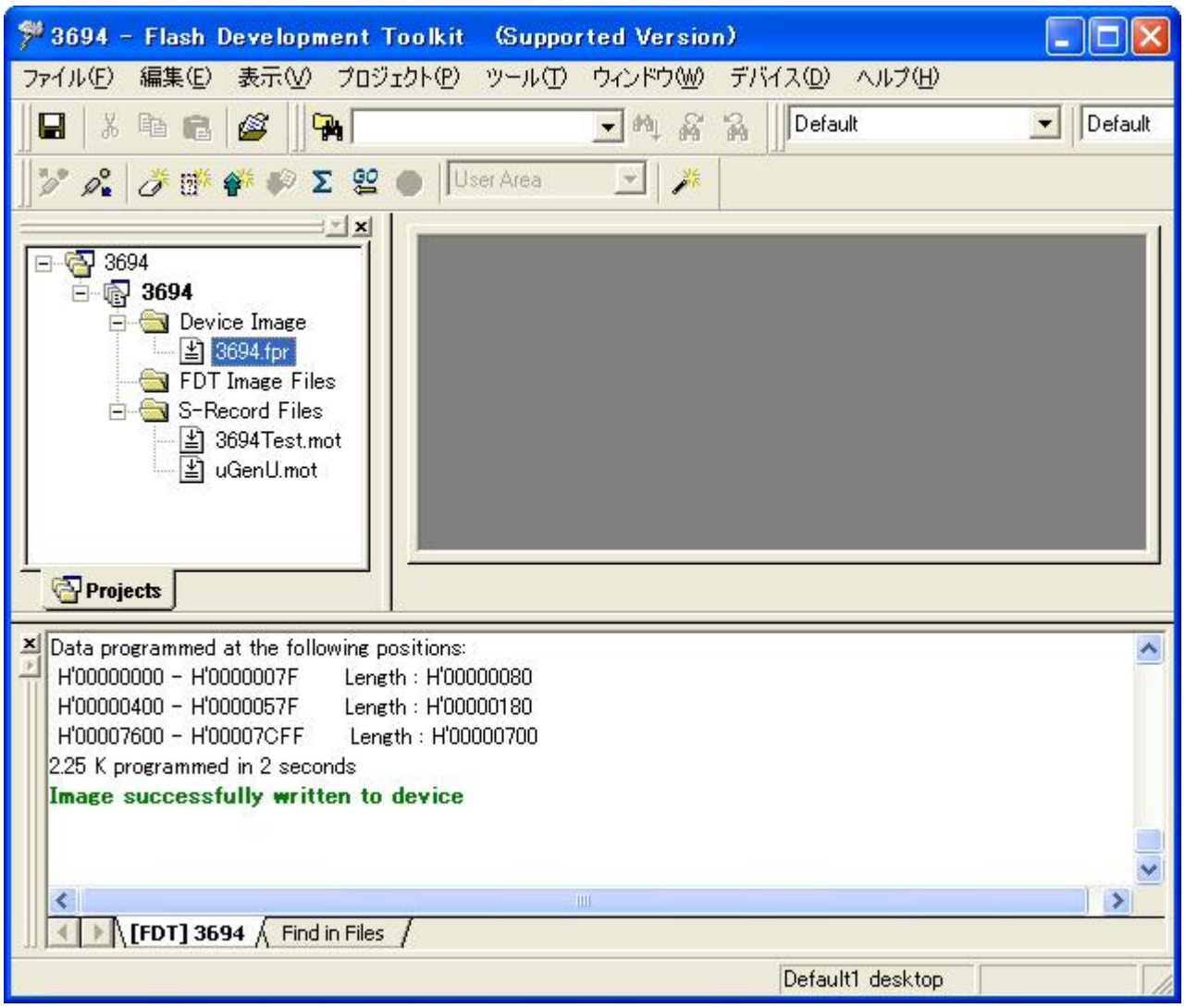
4.3.3　　書き込み

ユーザエリアに書き込みます。

3694.fpr 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、‘イメージのダウンロード［User Area］’をクリックし、3694.fpr ファイルをユーザエリアへダウンロードします。



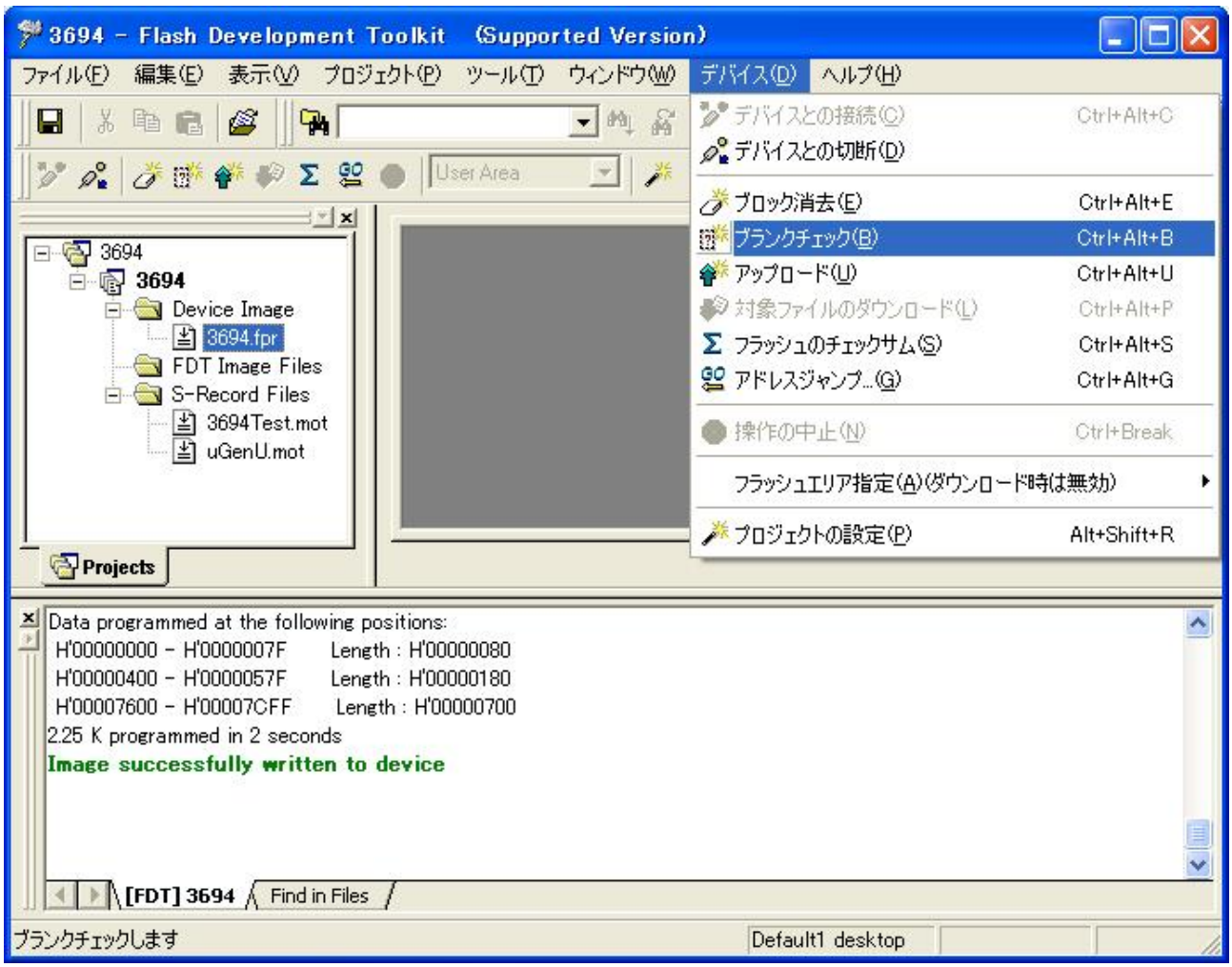
ユーザエリアにプログラムがダウンロードされたのを確認することができます。



4.3.4　　ブランクチェック

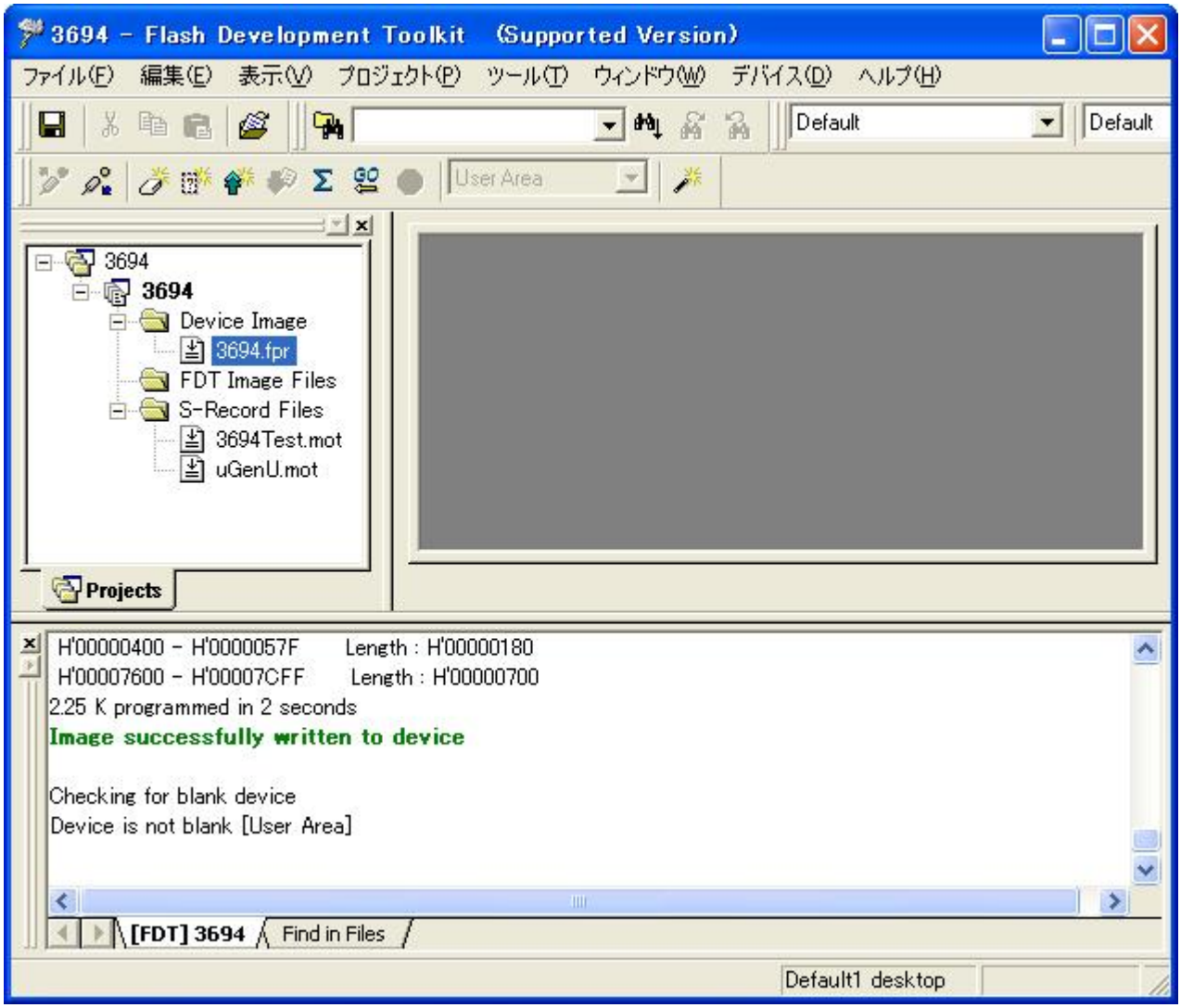
書き込まれたことを確認するために、ブランクチェックを行います。

‘デバイス(D)’からプルダウンメニューを開き、‘ブランクチェック(B)’をクリックします。



選択したエリアのブランクチェックの結果が表示されます。

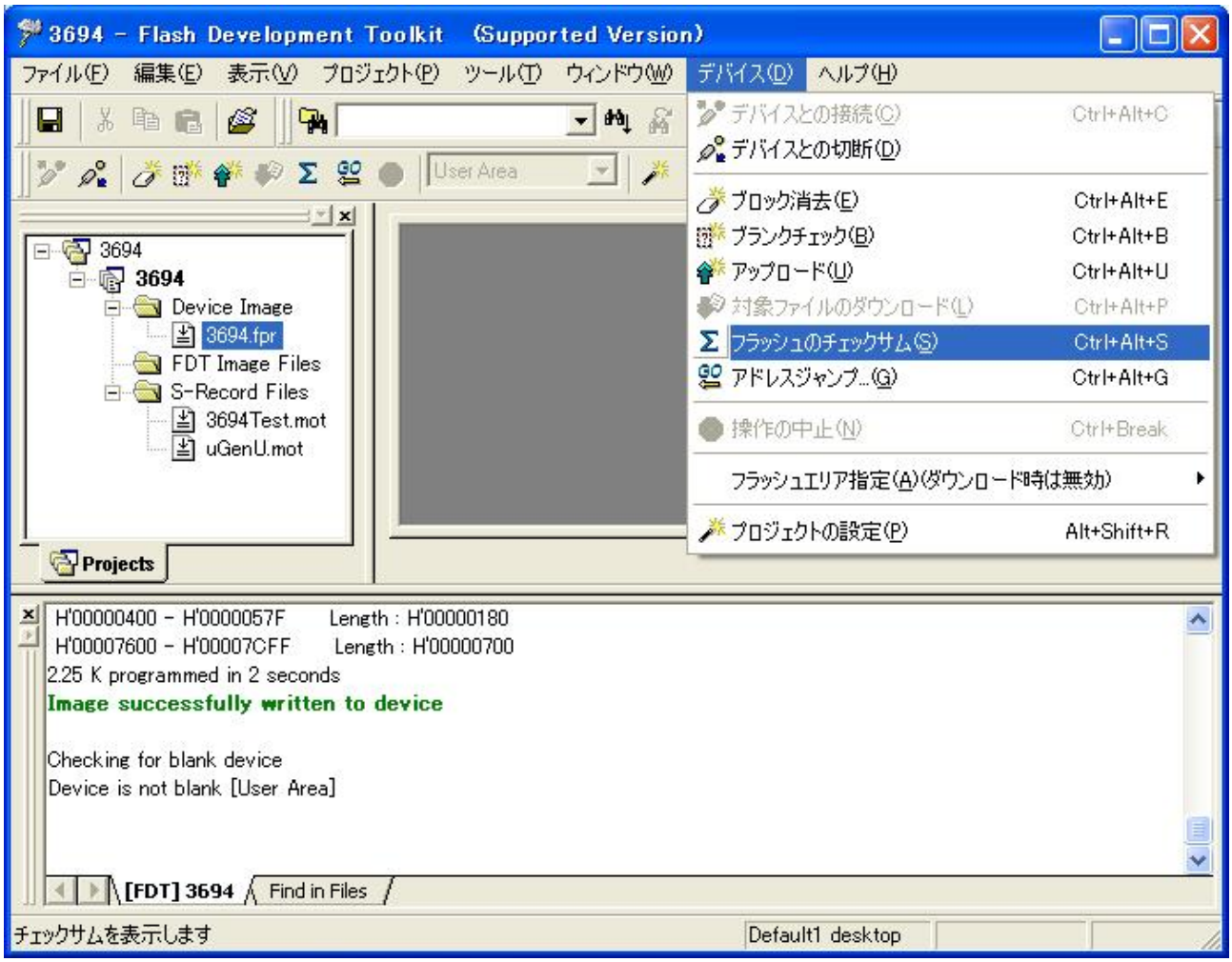
ユーザエリアはブランクではありません。



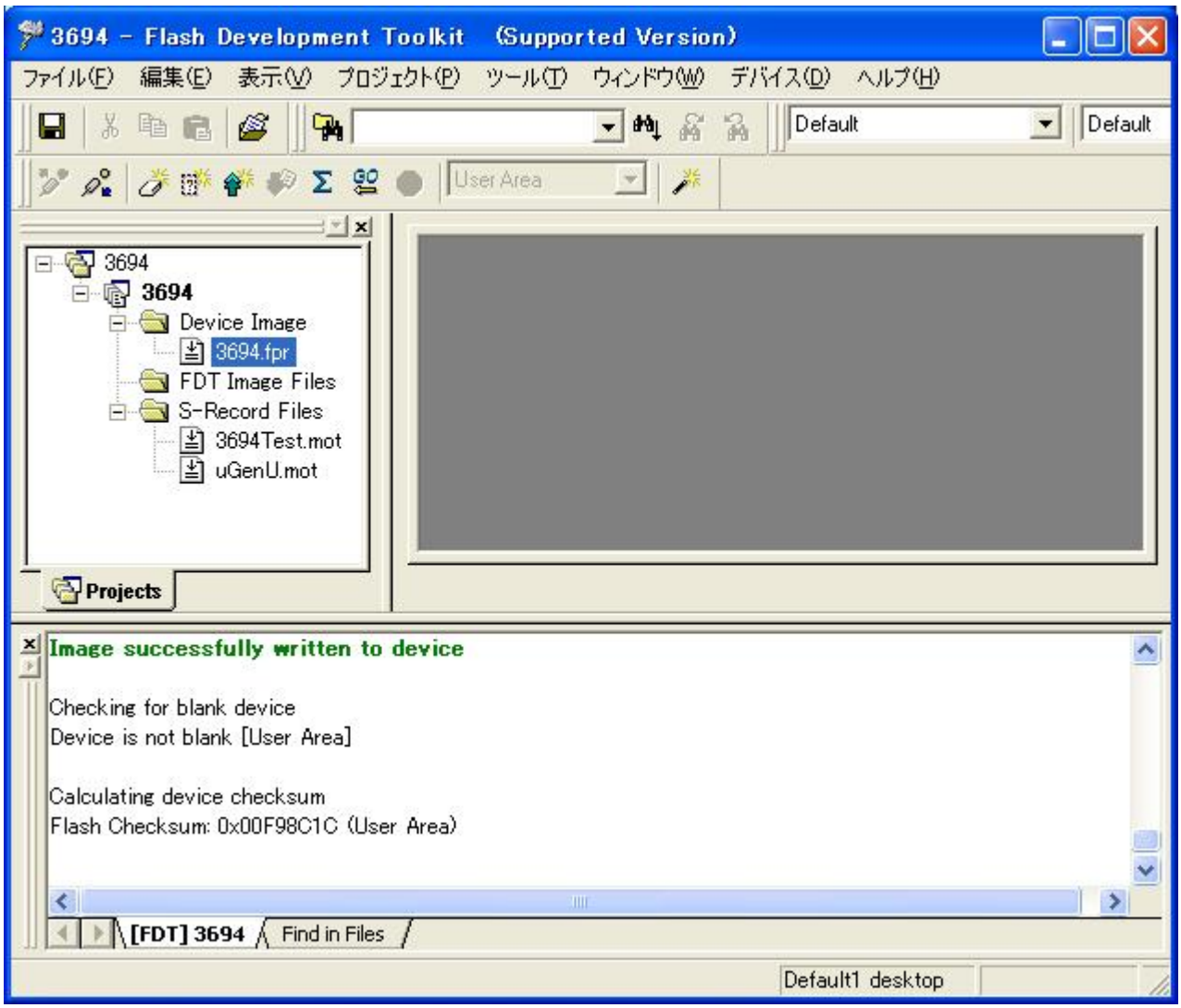
4.3.5　　チェックサム

書き込まれたことを確認するためにチェックサムを表示します。

‘デバイス(D)’からプルダウンメニューを開き、‘フラッシュのチェックサム(S)’をクリックします。



チェックサムの結果が表示されます。



ユーザエリアがブランクのときのチェックサムは以下の表示になります。

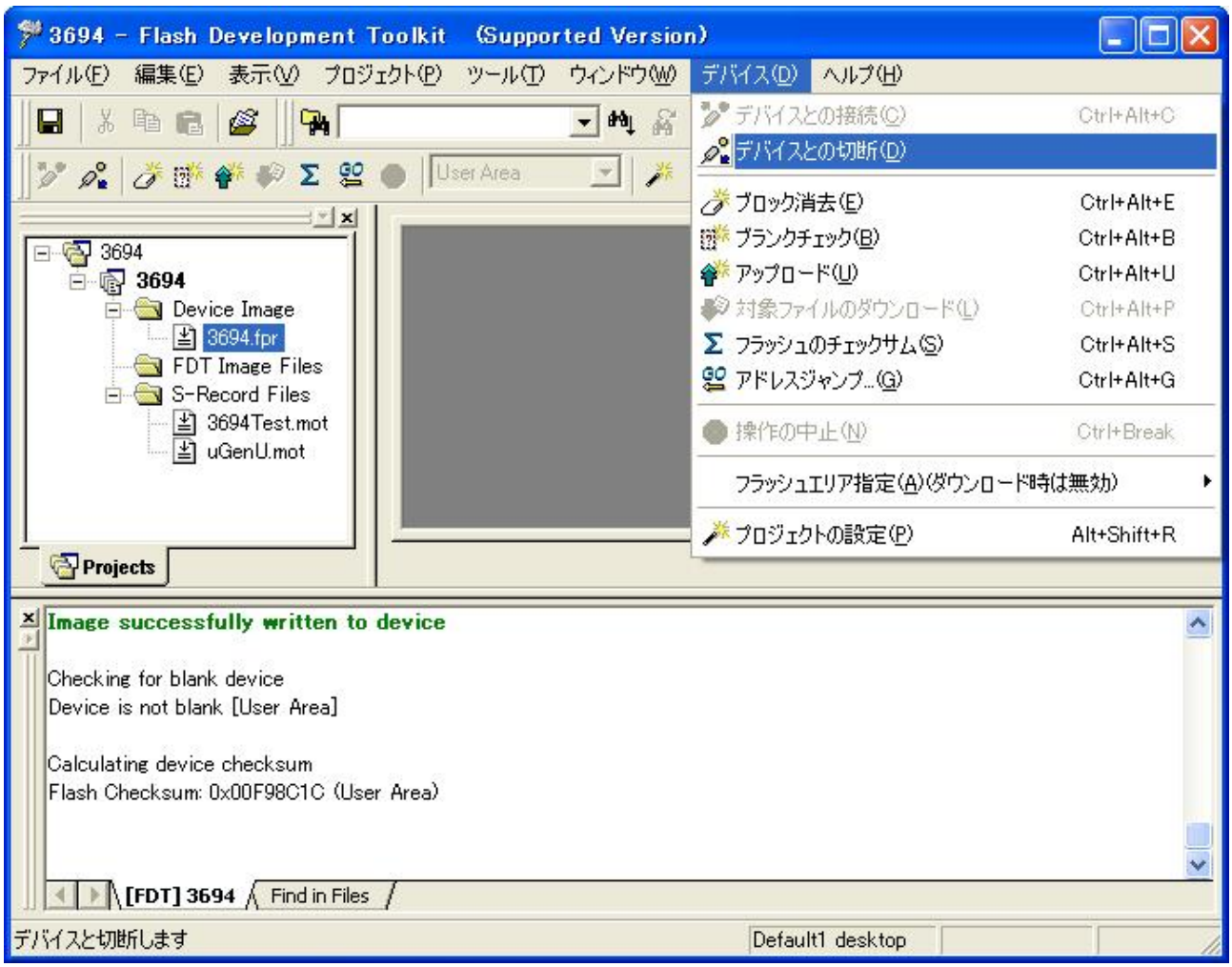
Calculating device checksum

Flash Checksum: 0x00FF0000 (User Area)

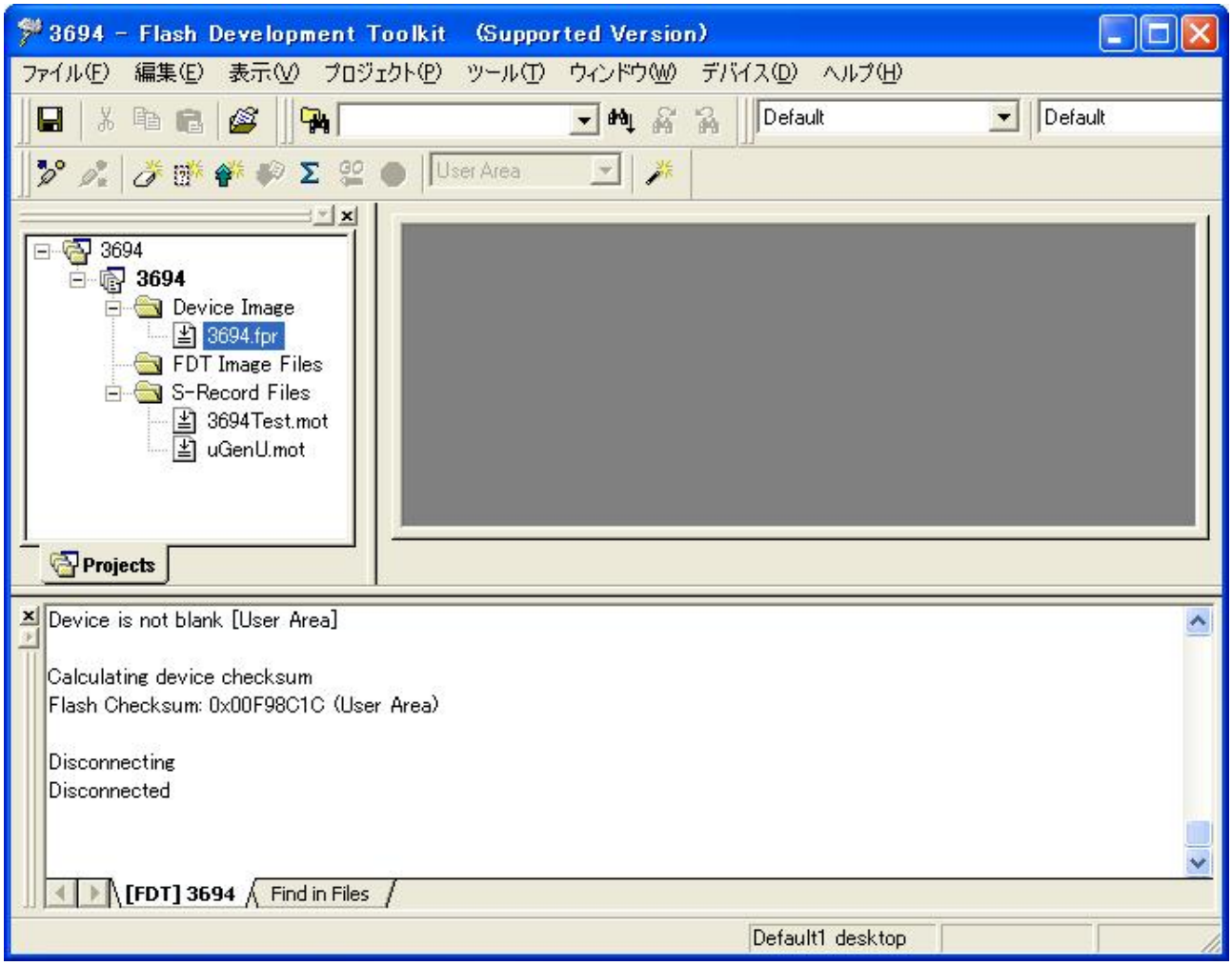
4.3.6　　デバイスとの切断

書き込み完了後、デバイスを切断します。

‘デバイス(D)’からプルダウンメニューを開き、‘デバイスとの切断(D)’をクリックします。



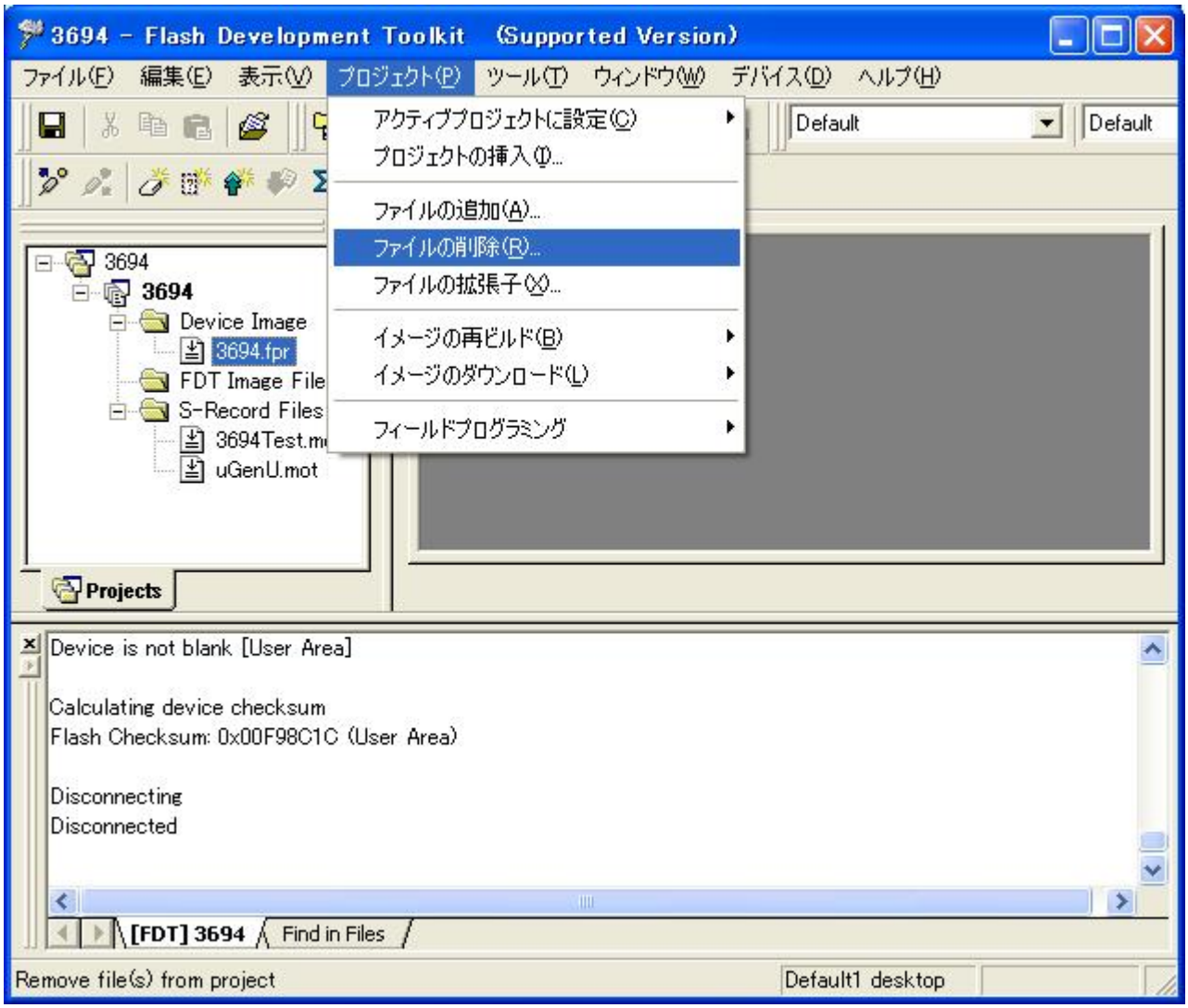
デバイスが切断されます。



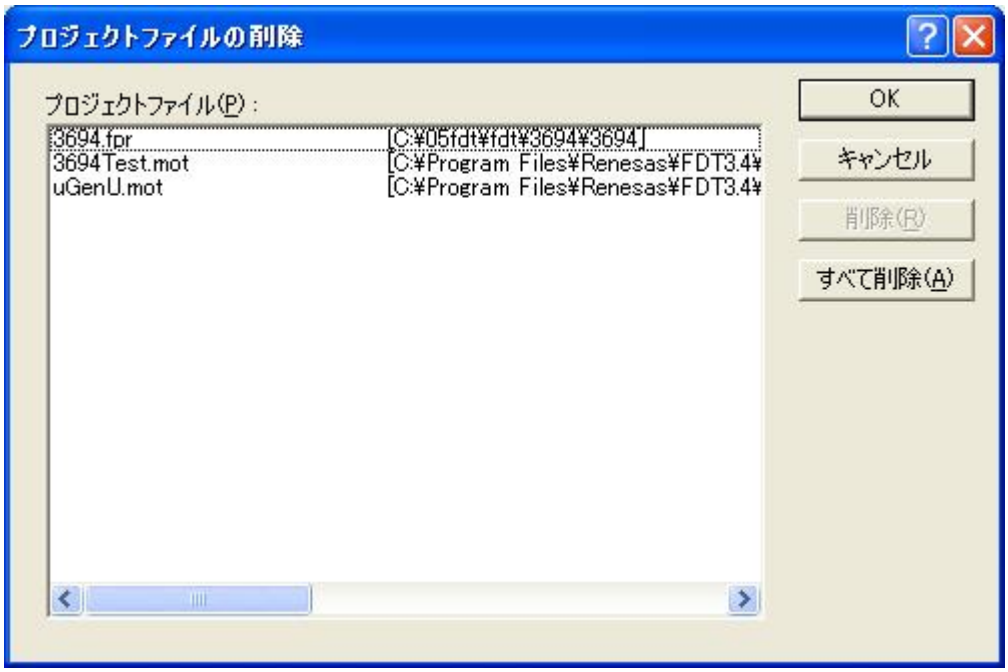
4.3.7　　ファイルの削除

ファイルを削除します。

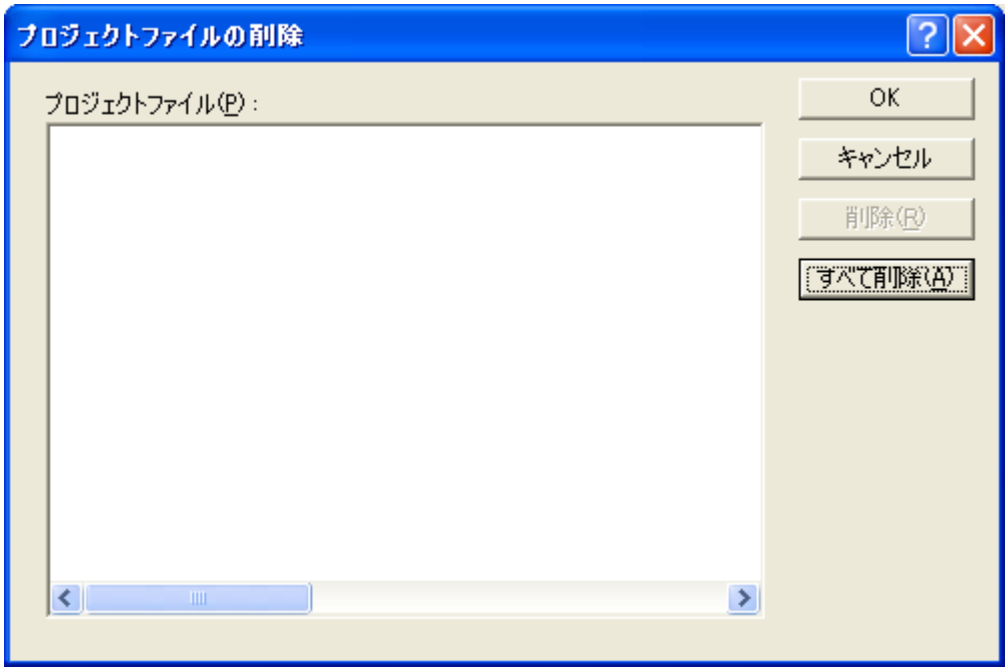
‘プロジェクト(P)’からプルダウンメニューを開き、‘ファイルの削除(R)’をクリックします。



デバイスが切断されます。

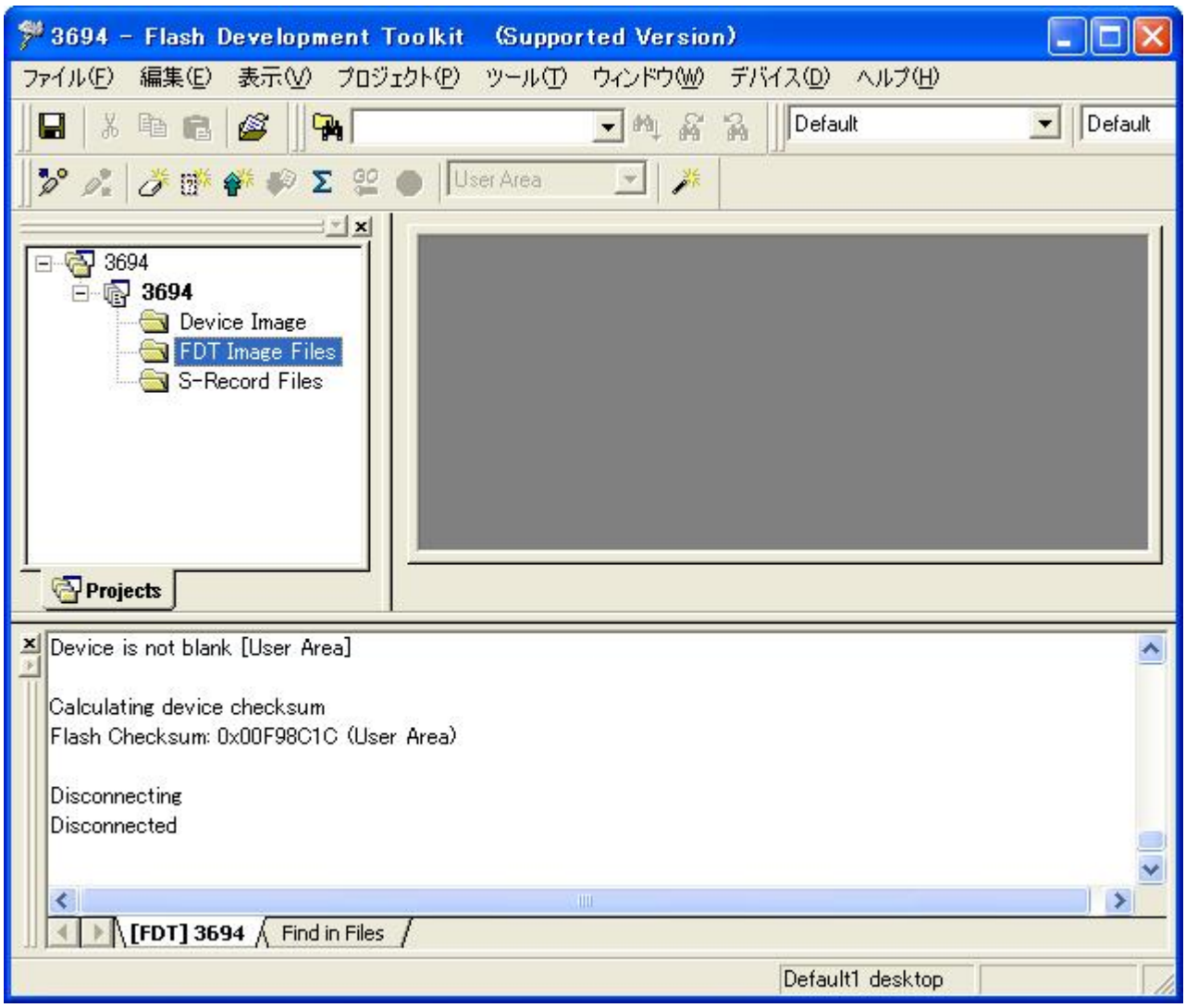


‘すべて削除(A)’をクリックします。



‘OK’をクリックします。

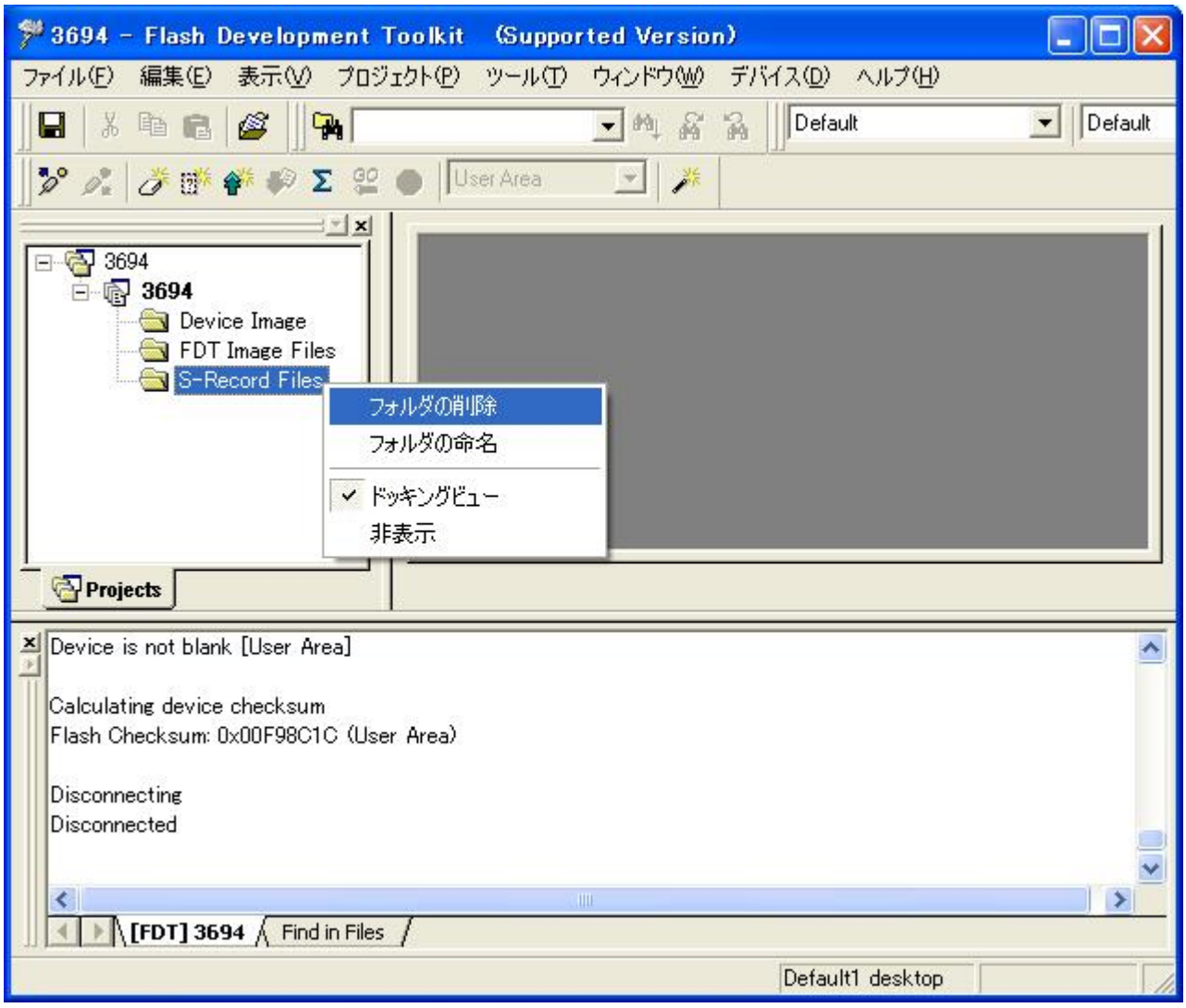
ファイルが削除されます。



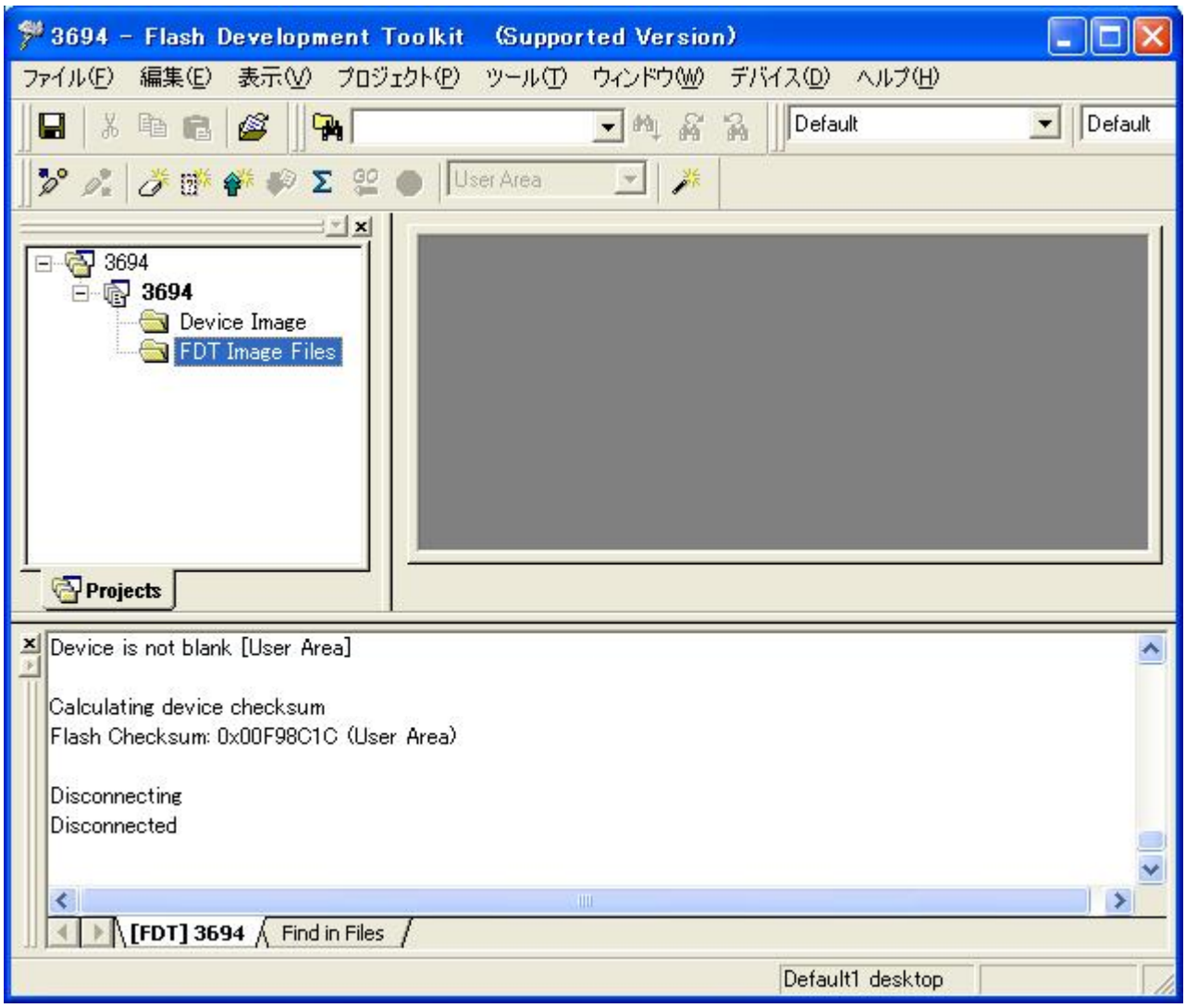
4.3.8　　フォルダの削除

フォルダを削除します。

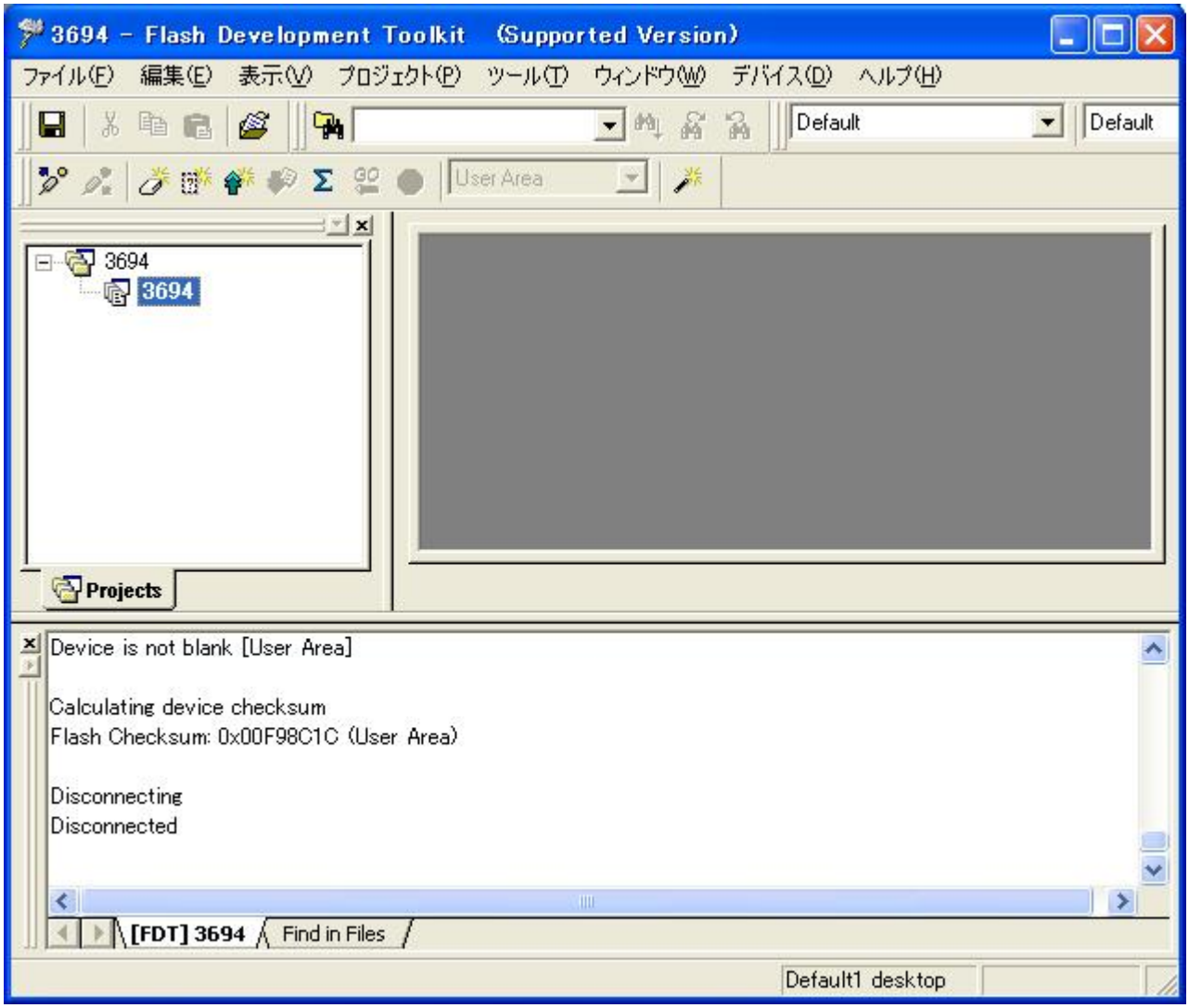
フォルダを右クリックしてポップアップメニューを開き、‘フォルダの削除’をクリックします。



フォルダが削除されます。



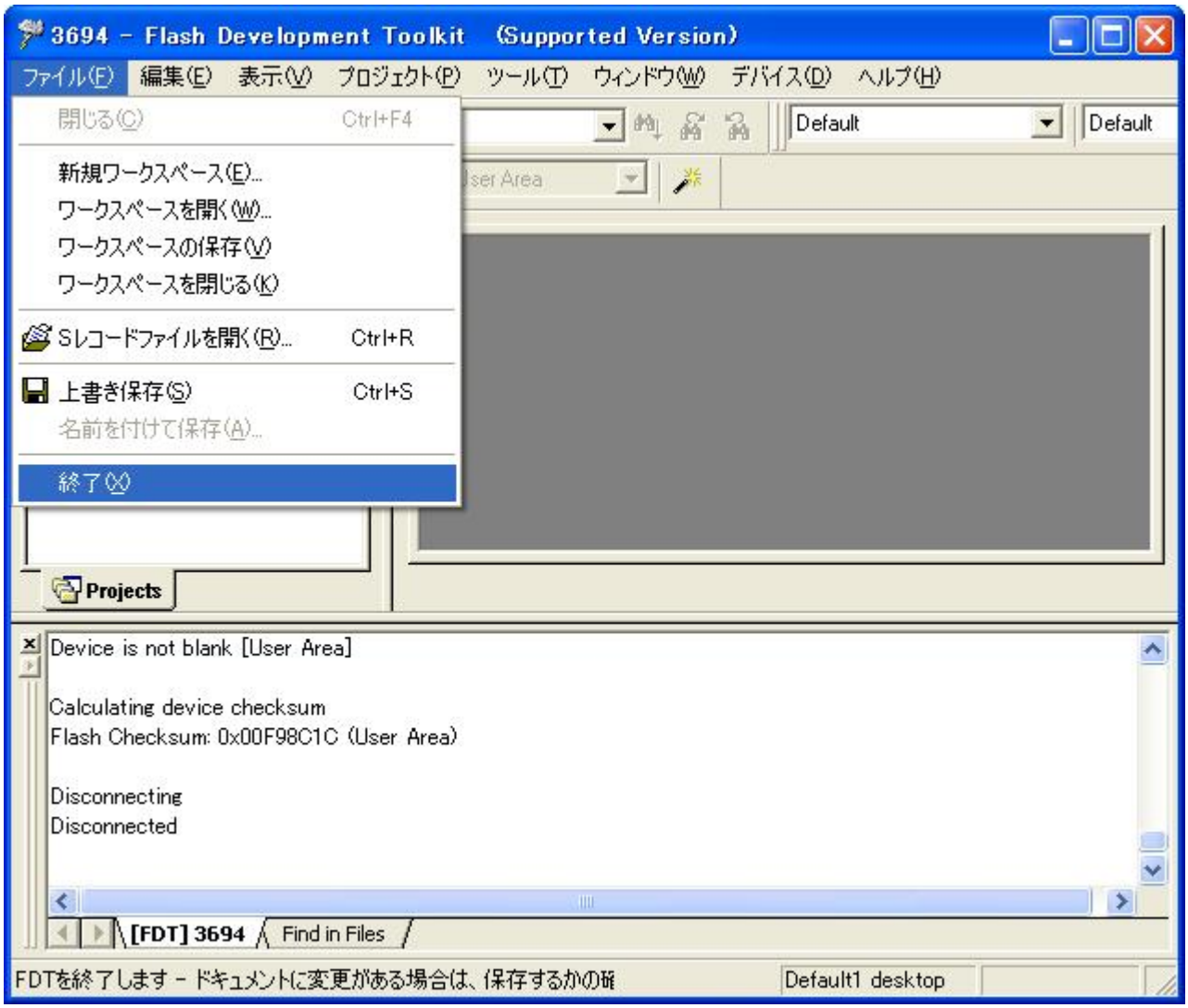
同じように、Device Image フォルダ、FDT Image Files フォルダを削除します。



4.3.9　　終了

ワークフォルダを保存して、終了します。

‘ファイル(F)’からプルダウンメニューを開き、‘終了(X)’をクリックします。



セッションの保存を選択します。

‘はい(Y)’をクリックします。

フラッシュ開発ツールキットが終了します。

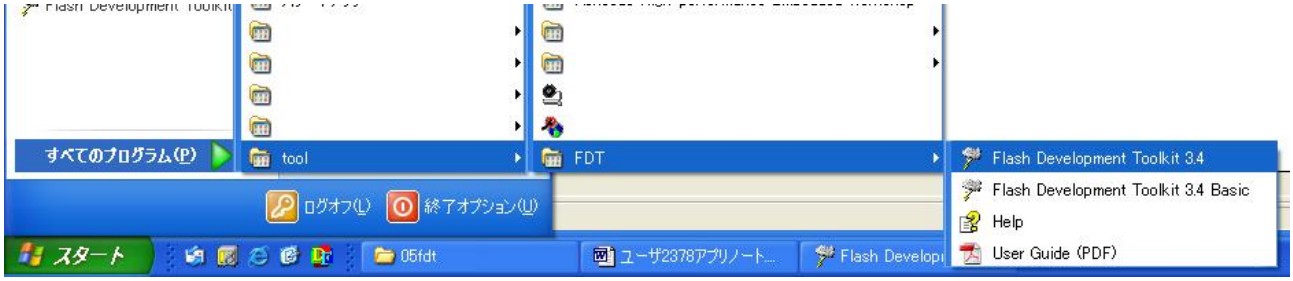
フラッシュ開発ツールキットのワークファイルスペースが 3694.AWS ファイルとして保存されます。

4.4　　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。

4.4.1　　フラッシュ開発ツールキットの起動

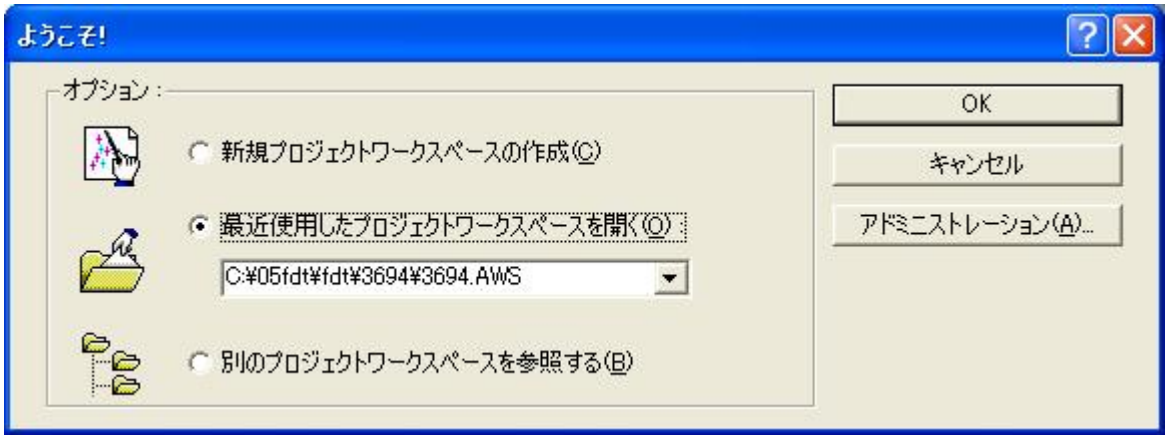
すべてのプログラムから‘Flash Development Toolkit 3. 4 ’を選択します。



4.4.2　　オプションの選択

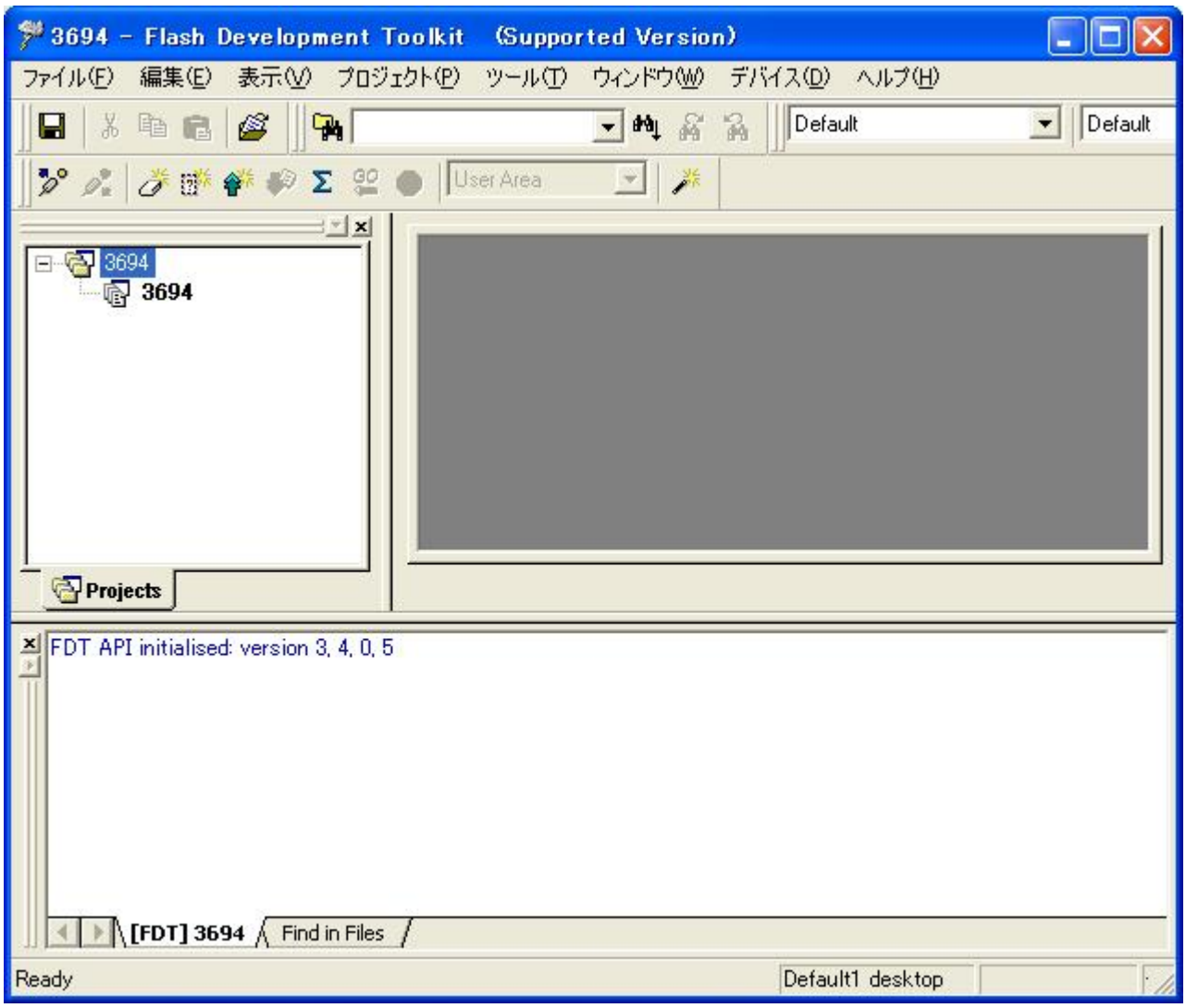
フラッシュ開発ツールキット のようこそ画面が表示されます。

‘最近使用したプロジェクトワークスペースを開く(O)’を選択し、プロジェクトワークスペースファイル 3694.AWS を選択します。



選択が終了したら‘OK’をクリックします。

3694 プロジェクトが表示されます。



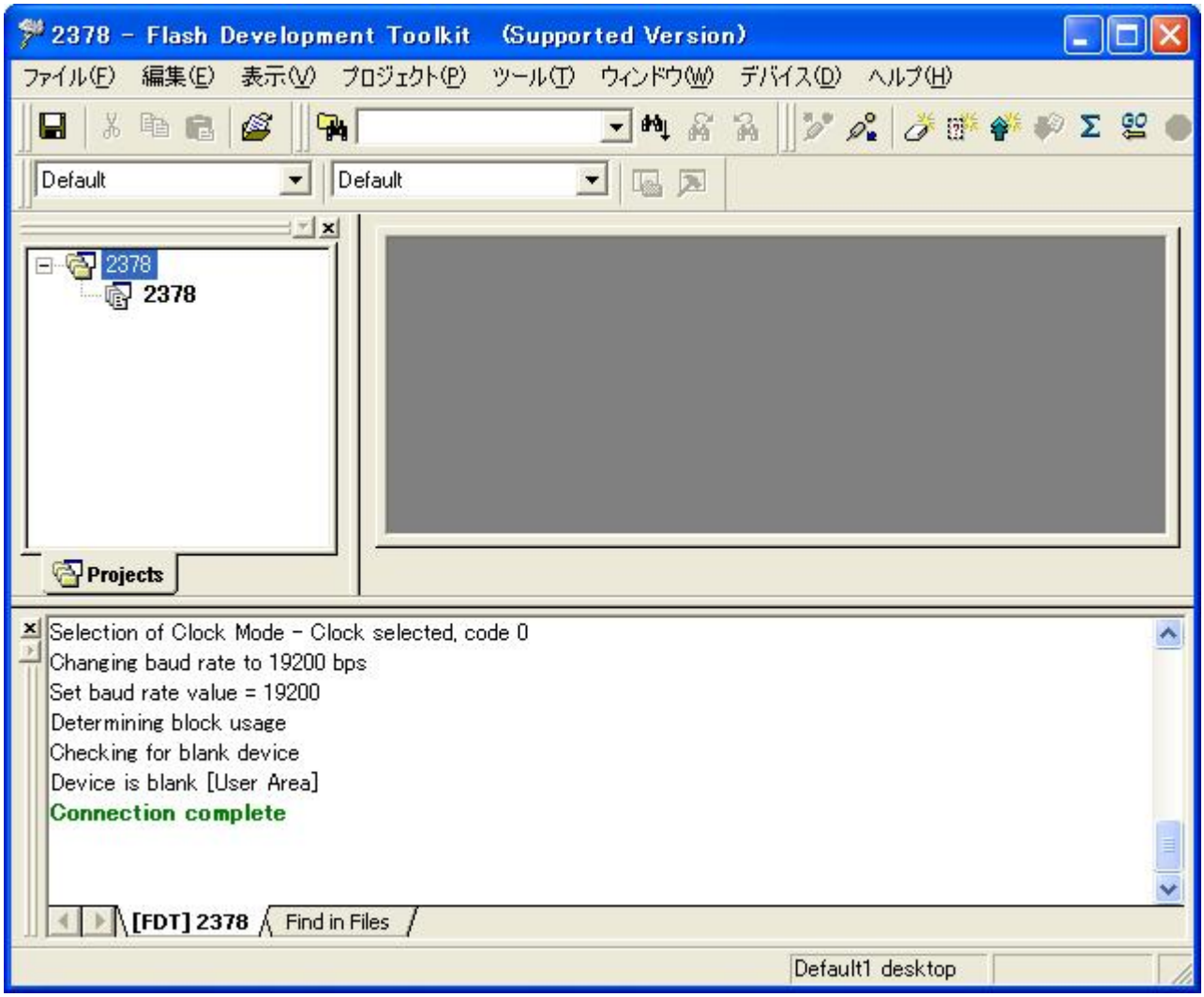
この操作は、直接プロジェクトワークスペースファイル 3694.AWS を開く（またはダブルクリック）でも、フラッシュ開発ツールキットを起動させることができます。

4.4.3　　デバイスとの接続

アダプタボード（FDM）をパソコンに接続し、H8S/2378F のボードをアダプタボードに接続し、電源を入れてください。

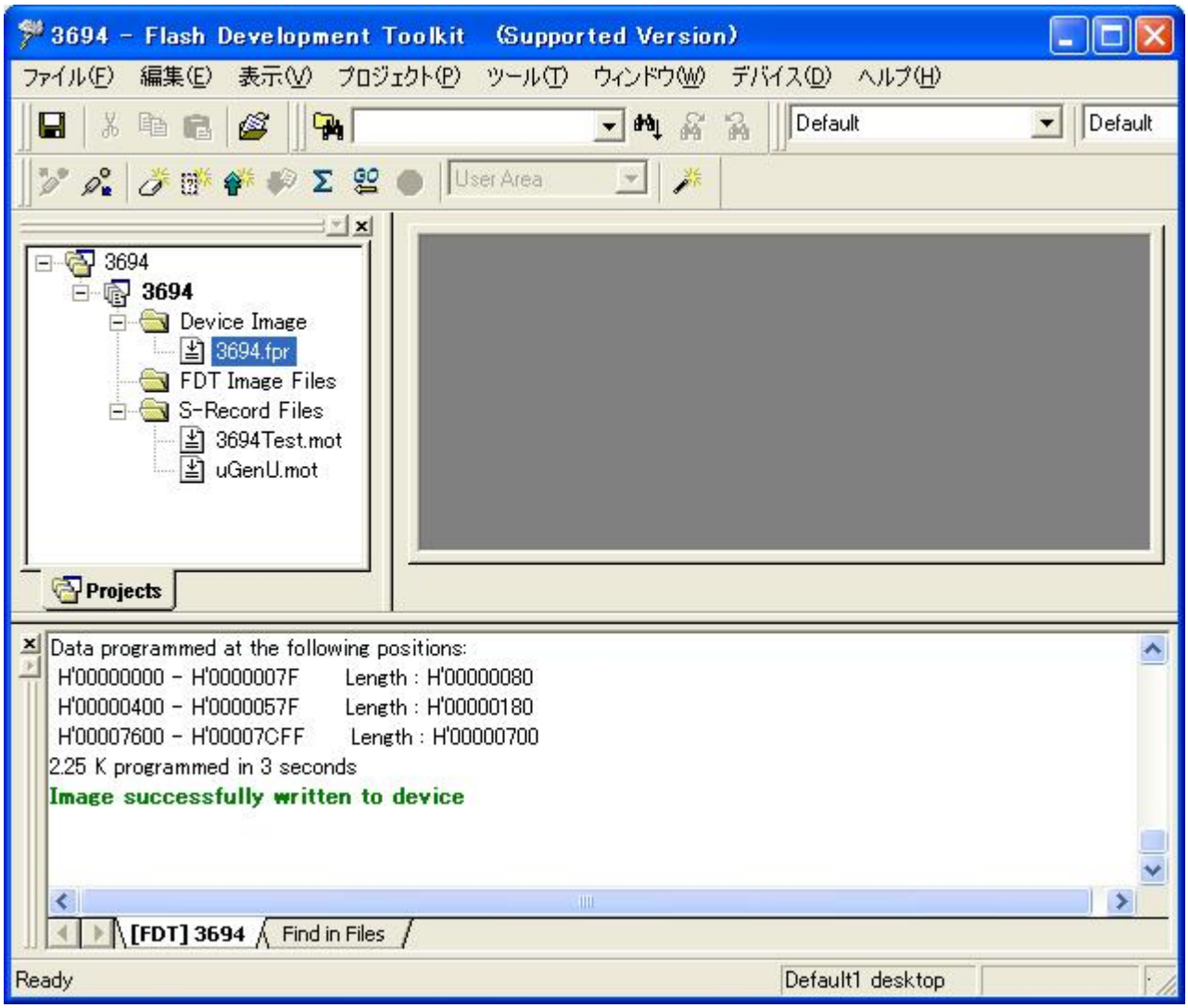
接続が完了したら、‘デバイス(D)’からプルダウンメニューを開き、‘デバイスとの接続(C)’をクリックします。

デバイスが接続されます。



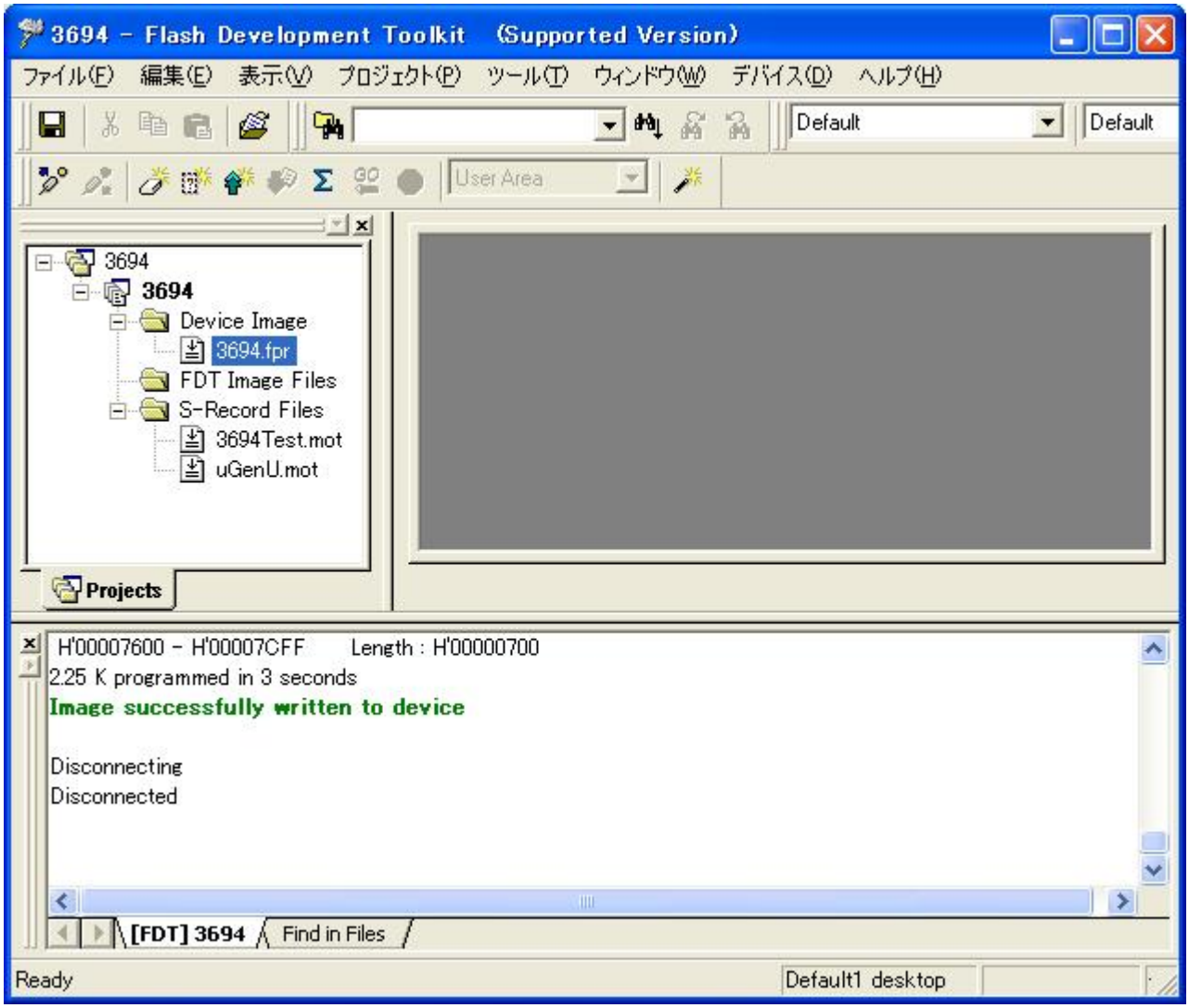
4.4.4 ユーザエリアへのプログラムの書き込み

3694Test.mot と uGenU.mot ファイルを追加し、イメージをビルドして 3694.fpr ファイルを作成します。次に、3694.fpr ファイルをダウンロードして、ユーザエリアにプログラムを書き込みます。



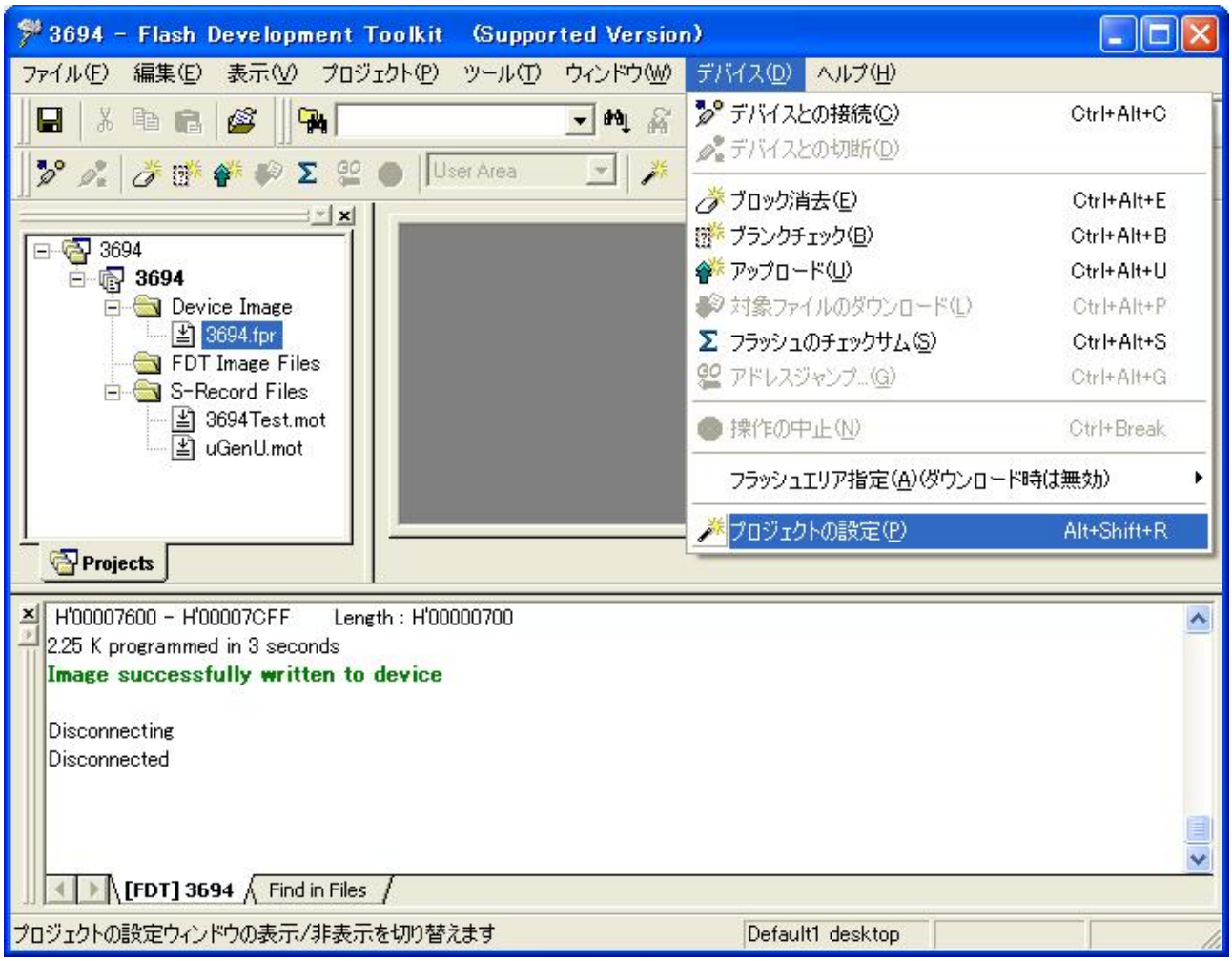
4.4.5 デバイスとの切断

‘デバイス(D)’からプルダウンメニューを開き、‘デバイスとの切断(D)’をクリックします。

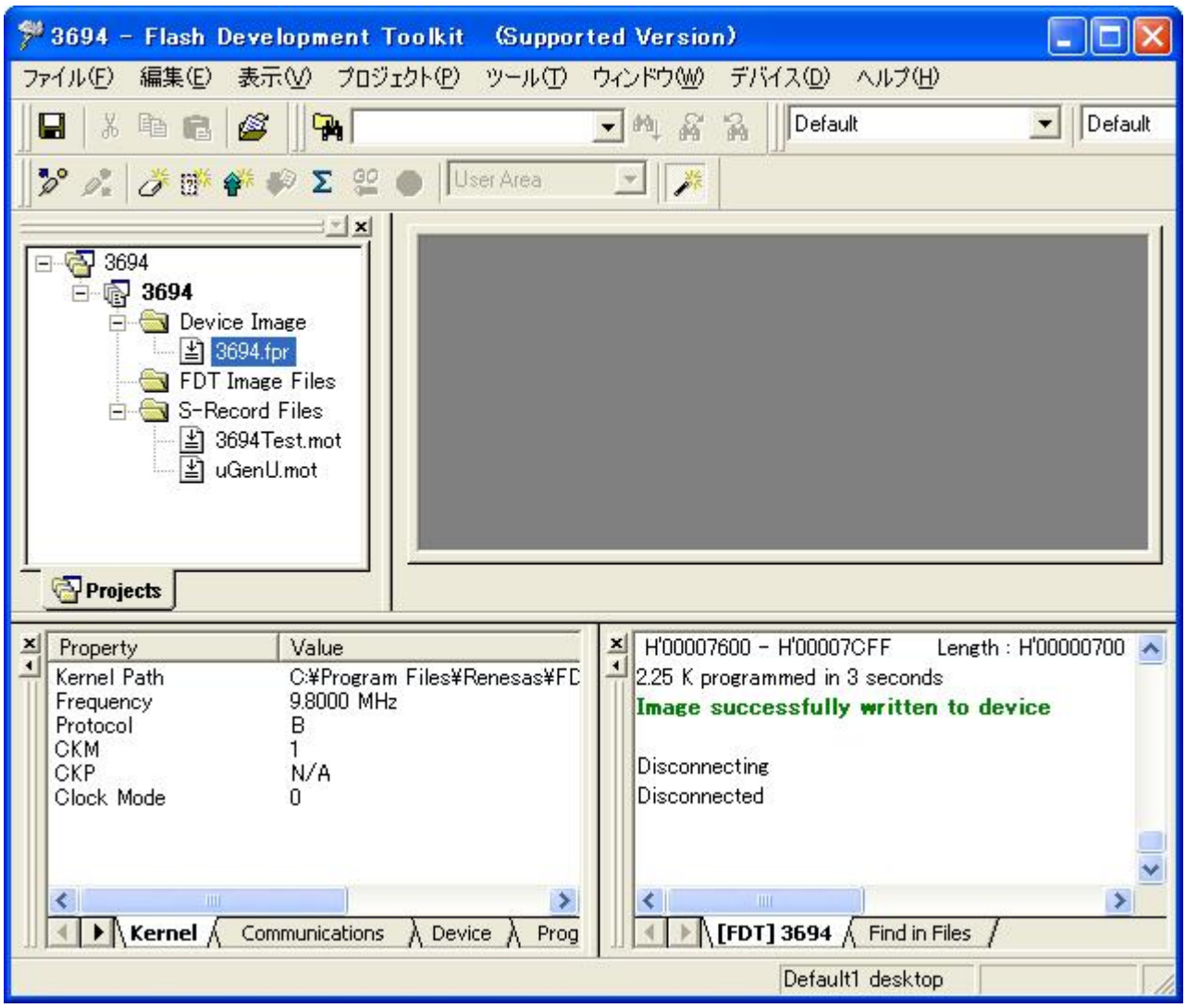


4.4.6　　プロジェクトの設定

‘デバイス(D)’からプルダウンメニューを開き、‘プロジェクトの設定(P)’をクリックします。

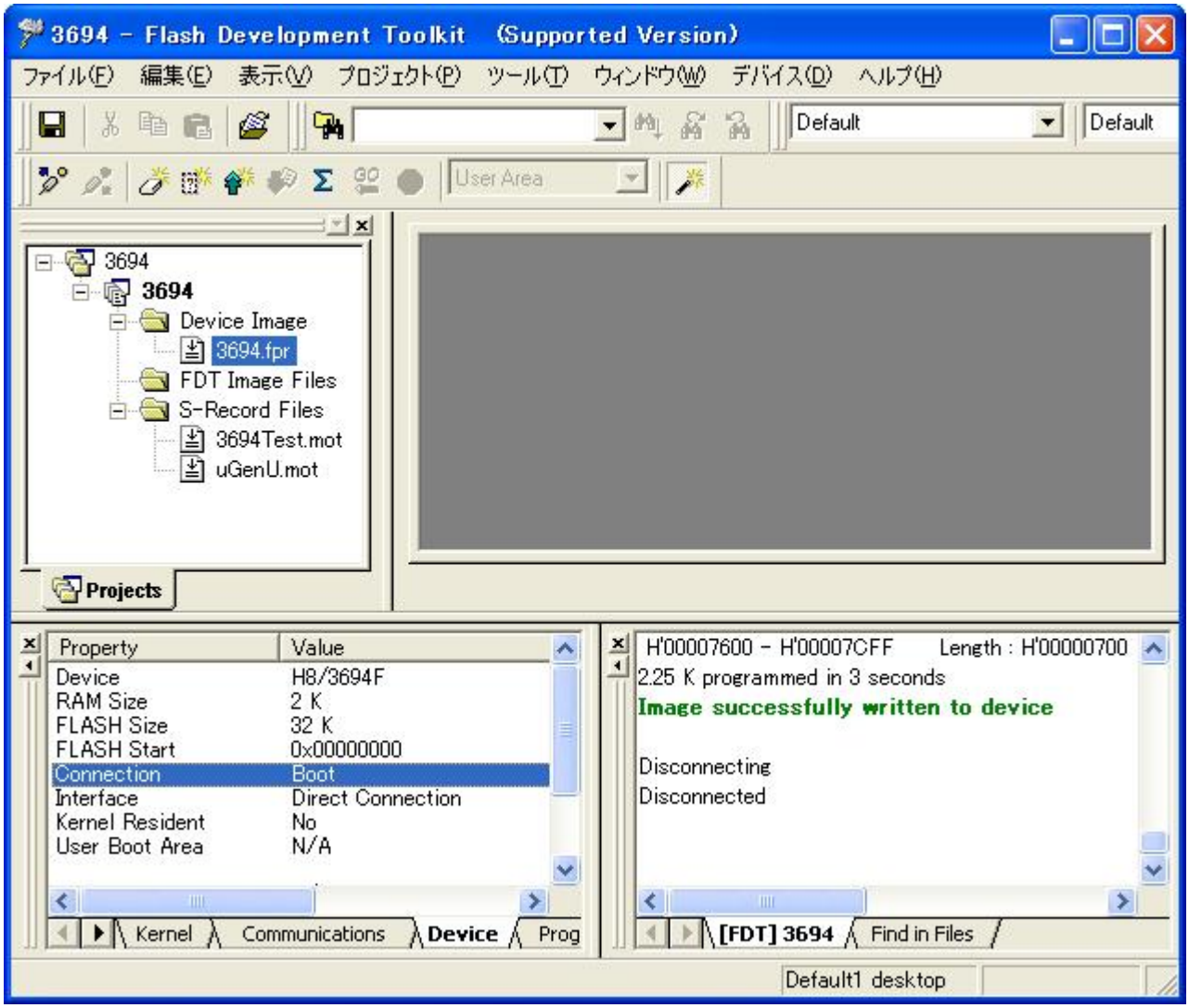


プロジェクト設定ウィンドウが表示されます。



4.4.7 ユーザプログラムモードの設定

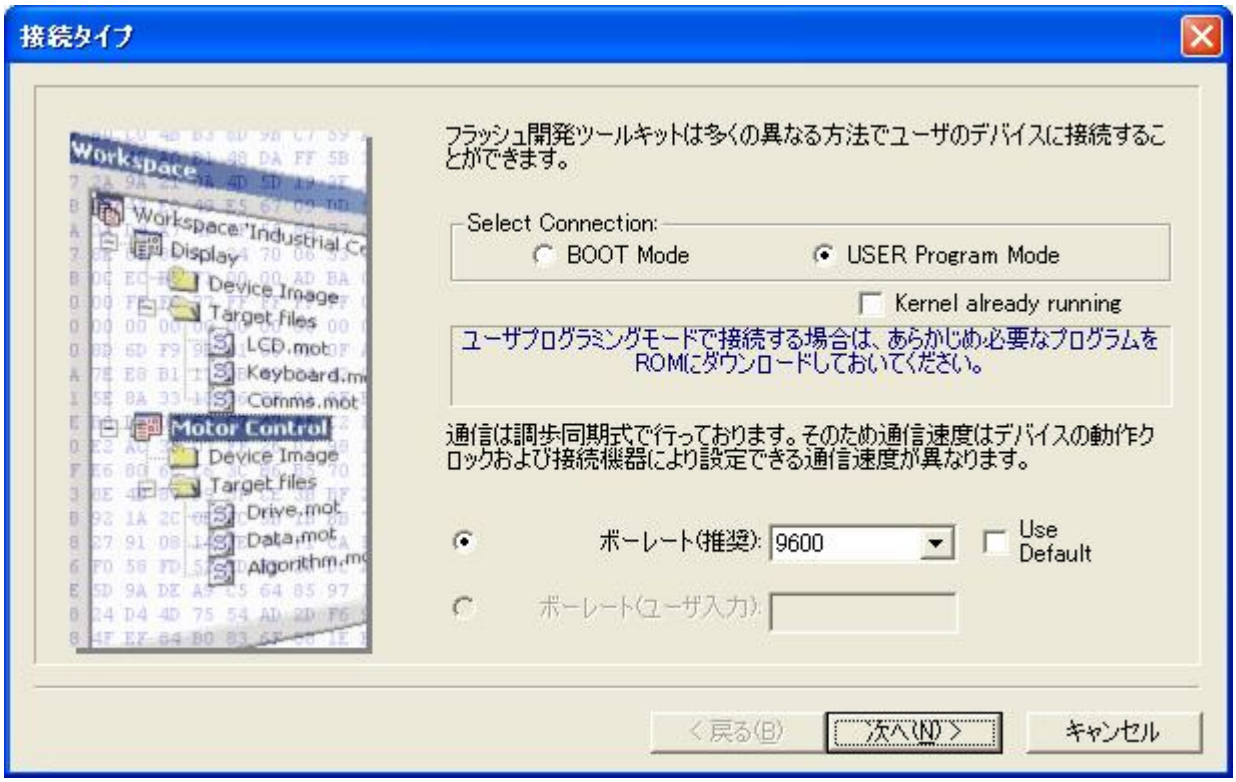
プロジェクト設定ウィンドウの‘Device’タブを選択し、‘Connection’‘Boot’をダブルクリックします。



接続タイプを設定します。

接続選択からユーザプログラムモードを選択します。

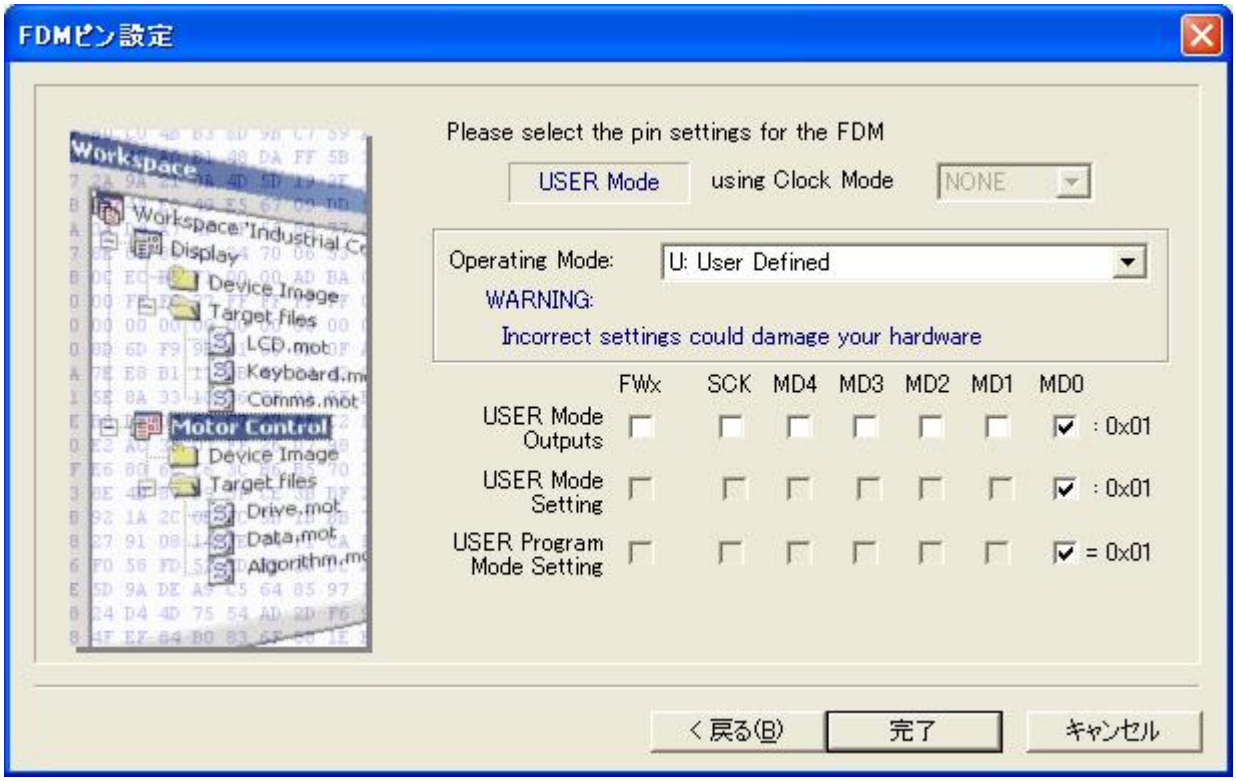
ボーレートを 9600bps に設定します。



選択が終了したら‘次へ(N)>’をクリックします。

ユーザプログラムモードのアダプタボード（FDM）ピンを設定します。

H8/3694F のユーザプログラムモードでは、$\overline{\text{NMI}}$ を出力ハイ（1）に設定します。H8/3694F のユーザシステムは、MD0 は $\overline{\text{NMI}}$ に接続されています。そのため、MD0 を出力ハイ（1）に設定します。FEW 端子はないので設定の必要はありません。



選択が終了したら‘完了’をクリックします。

H8/3694Fとルネサス製アダプタボード(HS0008EAUF1H)の接続例を 図 4-6に示します。プルアップおよびプルダウンの抵抗値は参考値ですので、ユーザシステムにてご評価頂けるようお願い申し上げます。

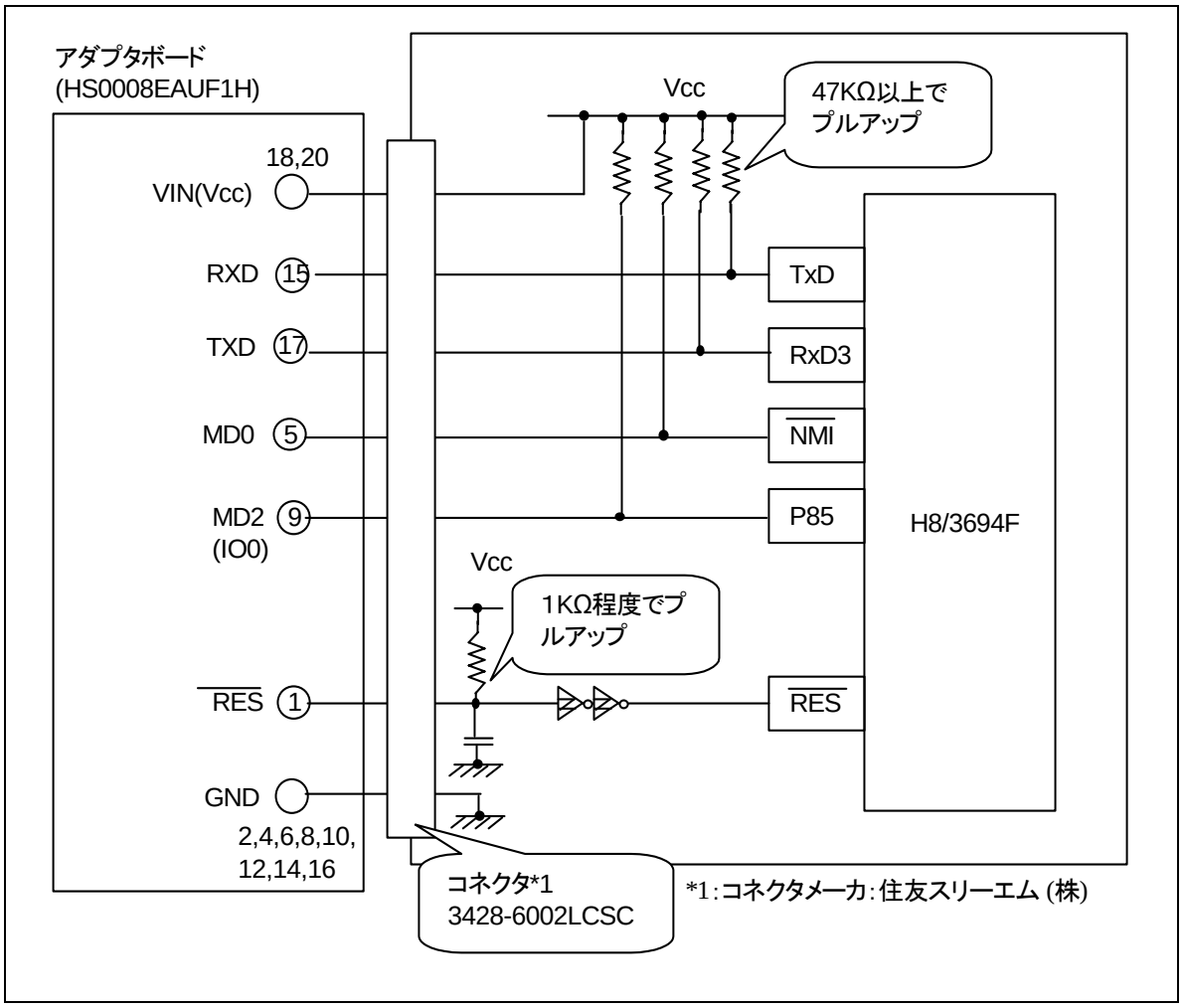


図 4-6 H8/3694F とアダプタボードの接続例

H8/3694Fユーザシステムとルネサス製アダプタボード(HS0008EAUF1H)の接続したときのユーザプログラムモード時の端子の設定例を 表 4-5に示します。

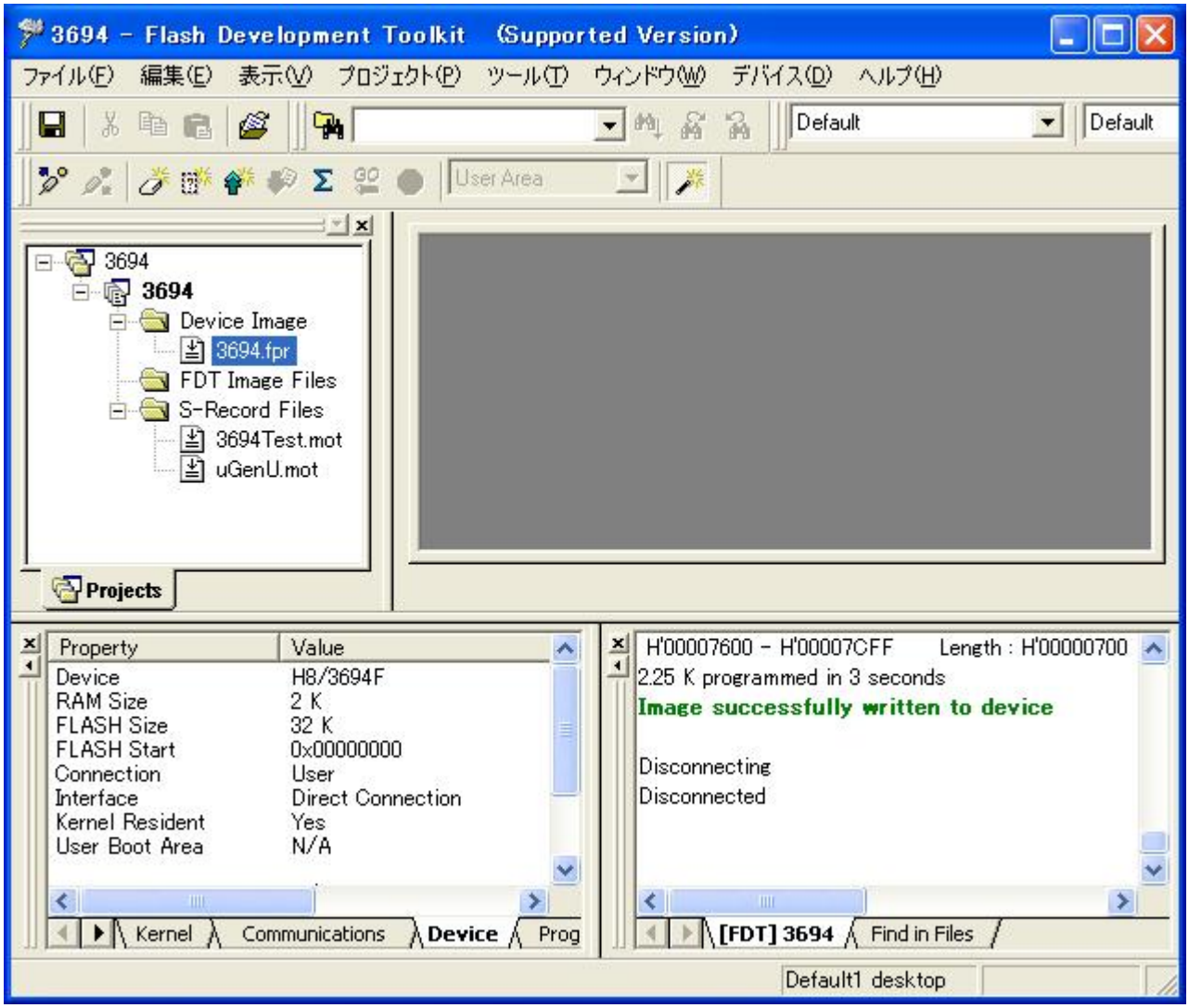
表 4-5 H8/3694F とアダプタボードの端子の設定例(ユーザプログラムモード時)

ピン番号	アダプタボード端子	デバイス端子	入出力	出力レベル
1	$\overline{\text{RES}}$	$\overline{\text{RES}}$	出力（デフォルト）	アダプタボード
3	FWx	NC	NC	—
5	MD0	$\overline{\text{NMI}}$	出力	ハイ（1）
7	MD1	NC	NC	—
9	MD2（IO0）	P85	入力	—
11	MD3（IO1）	NC	NC	—
13	MD4（IO2）	NC	NC	—
15	RXD	TXD	入力（デフォルト）	アダプタボード
17	TXD	RXD	出力（デフォルト）	アダプタボード
19	SCK（NC）	NC	NC（デフォルト）	—

［注］NC： No Connection（接続なし）を意味します。

4.4.8 設定の完了

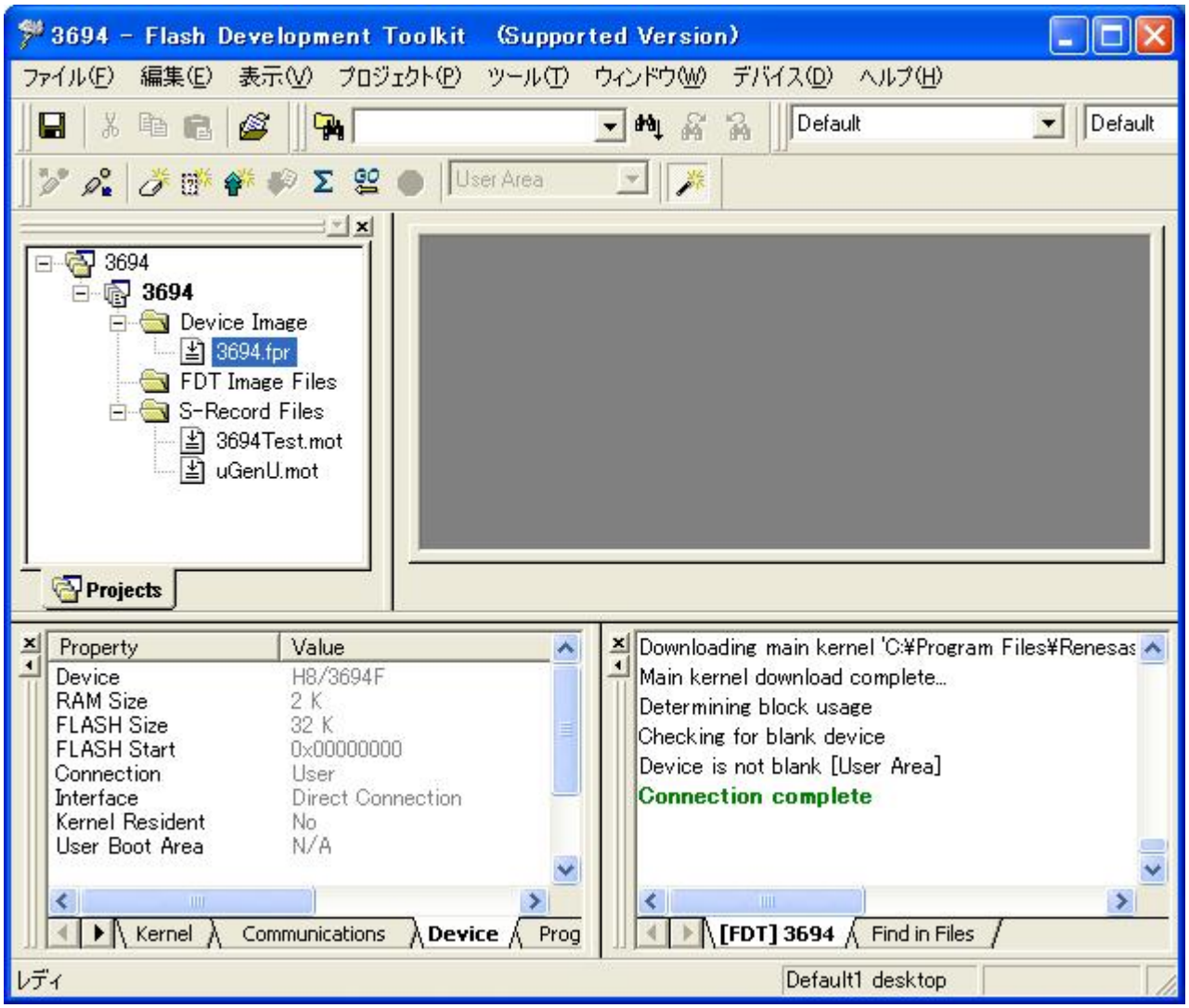
ユーザプログラムモードを設定しました。



4.4.9　　デバイスとの接続

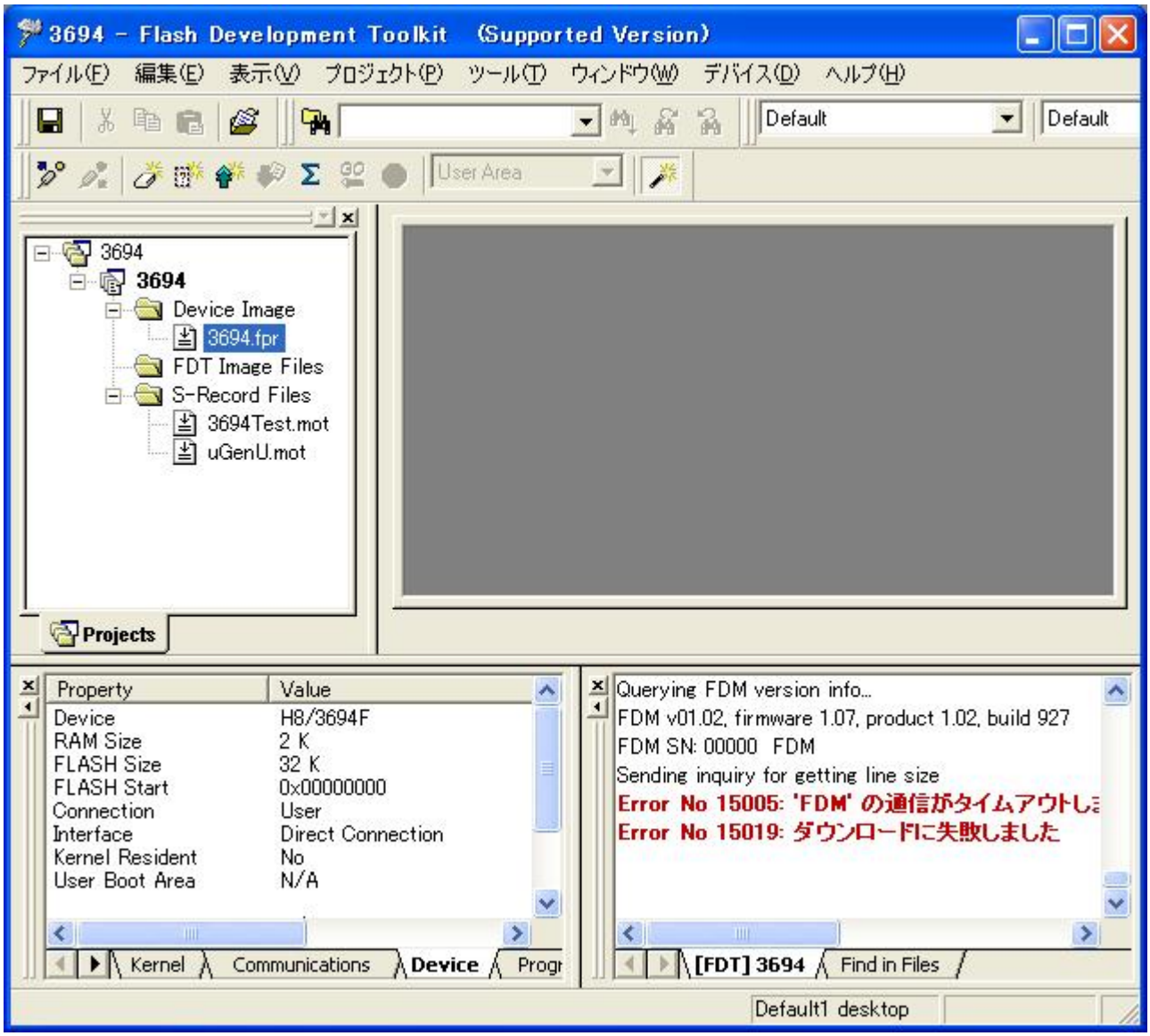
設定が完了したら、‘デバイス(D)’からプルダウンメニューを開き、‘デバイスとの接続(C)’をクリックします。

ユーザプログラムモードでの接続が完了します。



4.4.10　　タイムアウト

デバイスの接続でタイムアウトエラーが出ることがあります。



原因はいろいろありますが、次の二つの修正をしていない可能性がありますので、確認してください。

（１）　サンプルプログラムのビットレートを 9600bpsに修正する。

ビットレートの修正は、「7.1.1　ビットレートの設定（GenTest.h)」を参照してください。

（２）　シリアル入力をI/Oバス（26P）(J2）に接続する。

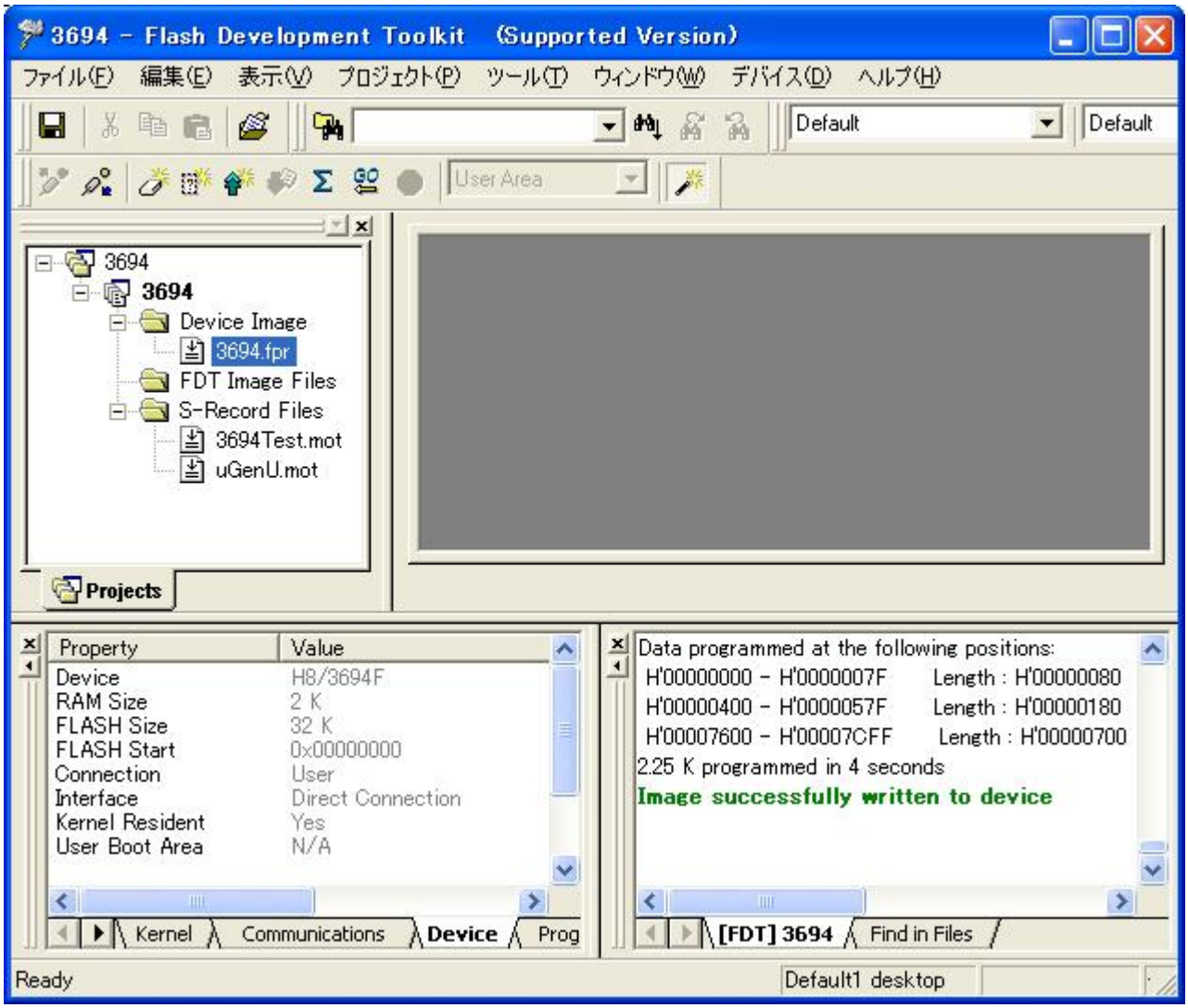
接続方法は、「4.1.5　H8/3694Fユーザシステムのシリアル入出力」を参照してください。

4.4.11　　書き込み

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。

3694.fpr 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、‘イメージのダウンロード［User Area］’をクリックし、3694.fpr 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たのを確認することができます。



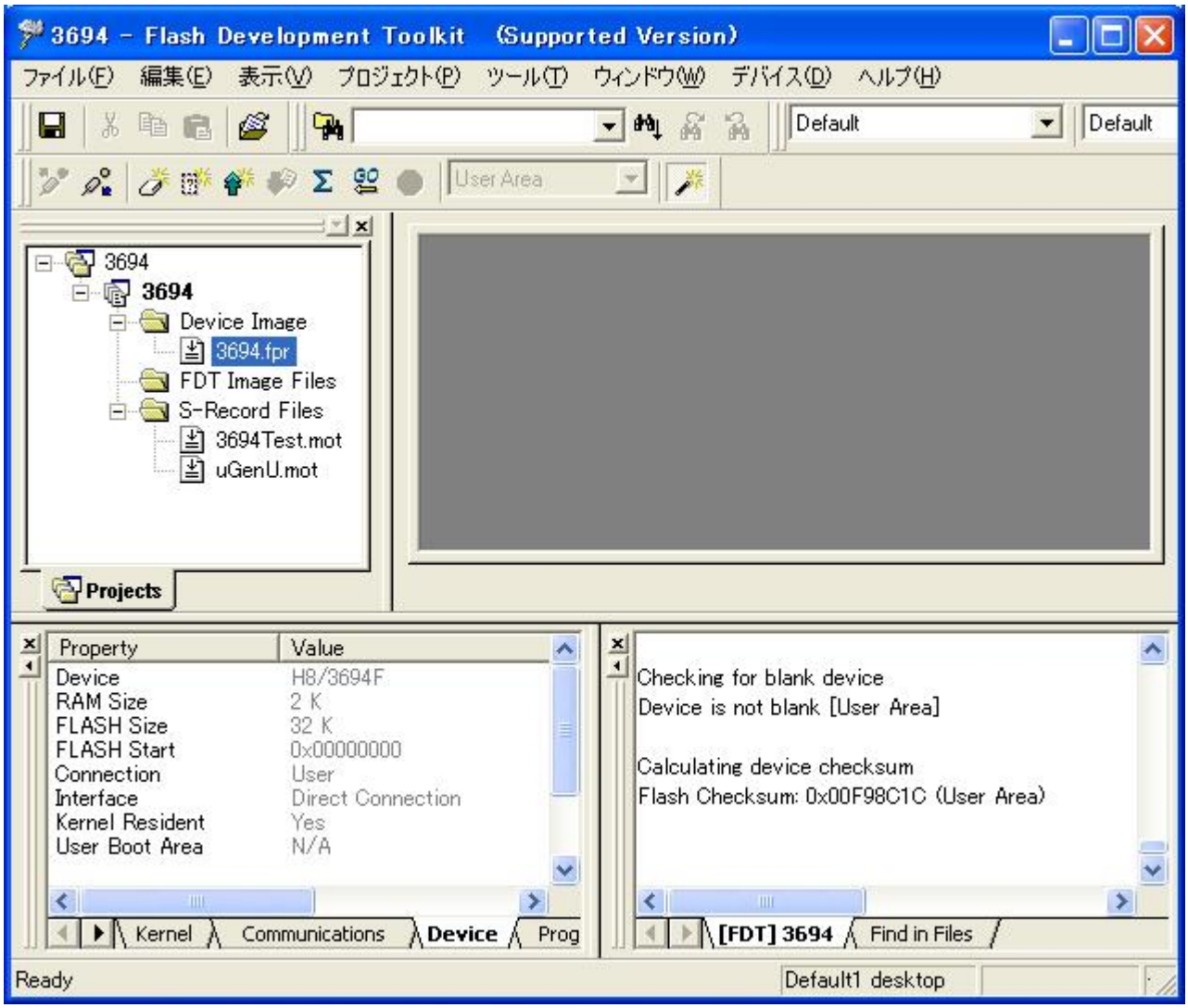
4.4.12　　ブランクチェックとチェックサム

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(D)’からプルダウンメニューを開き、‘ブランクチェック(B)’をクリックします。

‘デバイス(D)’からプルダウンメニューを開き、‘フラッシュのチェックサム(S)’をクリックします。

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. 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、ユーザプログラムモードの 2 つの接続方法があり、それぞれ、以前のセッションから実行継続（メインカーネルに直接接続）する指定ができます。フラッシュ開発ツールキットの接続方法を 表 5-1に示します。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。

表 5-1　フラッシュ開発ツールキットの接続方法

モード	通常処理	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 マイクロカーネル転送 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) メインカーネル転送	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) メインカーネル転送	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)

6. サンプルプログラム

H8/3694F のユーザプログラムモードサンプルプログラムについて説明します。

6.1 プログラムの構成

ユーザプログラムモードのサンプルプログラムの構成を 表 6-1に示します。

表 6-1 プログラムの構成

項番	プログラム	機能	格納と起動
1	メイン処理モジュール	マイクロカーネルに分岐	ブートモードで事前に ROM に格納 リセットで起動
2	マイクロカーネル	問い合わせ選択コマンド処理 送受信モジュールを RAM に転送 メインカーネルを受信し、RAM に格納 メインカーネルに分岐	ブートモードで事前に ROM に格納 メイン処理モジュールから分岐
3	メインカーネル	書き込み消去チェックコマンド処理 書き込み消去プログラムを受信し、RAM に格納 書き込み消去プログラムを呼び出す	マイクロカーネルが受信し、RAM に格納 マイクロカーネルから分岐
4	書き込みカーネル	フラッシュ書き込み	メインカーネルが受信し、RAM に格納 メインカーネルから呼び出し
5	消去カーネル	フラッシュ消去	メインカーネルが受信し、RAM に格納 メインカーネルから呼び出し

6.2 ファイル構成

プログラムファイルは、C:¥Program Files¥Renesas¥FDT3.4¥Kernels¥ProtB¥3694¥Renesas¥1_2_00 フォルダにあります。それぞれのプログラムのファイル構成を以下に示します。これらのプログラムは、ユーザ独自で作成するユーザプログラムモードプログラムのサンプルとして提供しているものです。

6.2.1 メイン処理モジュール

表 6-2 メイン処理モジュールのファイル構成

項番	ファイル名	内容	種類
1	3694Test.mot	ユーザプログラムモード メイン処理モジュール ロードモジュール	S-Type 形式
2	bTest.bat	ユーザプログラムモード メイン処理モジュール バッチファイル	MS-DOS バッチファイル
3	l3694t.xcl	ユーザプログラムモード メイン処理モジュール リンクサブファイル	リンケージエディタ コマンドファイル
4	Strt3694.src	スタック初期設定 ソースファイル	アセンブラ ソースファイル
5	GenTest.c	ユーザプログラムモード メイン処理モジュール メイン ソースファイル	C ソースファイル
6	GenTest.h	GenTest.c の関数の原型宣言	C ヘッダファイル
7	io3694.h	SCI、ポートのレジスタ定義	C ヘッダファイル
8	KDevice.h	デバイス特有の情報（カーネル配置定義 など）	C ヘッダファイル
9	KStruct.h	構造体定義　など	C ヘッダファイル
10	KTypes.h	型定義　など	C ヘッダファイル

6.2.2 マイクロカーネル

表 6-3 マイクロカーネルのファイル構成

項番	ファイル名	内容	ファイルの種類
1	uGenU.mot	ユーザプログラムモード マイクロカーネル ロードモジュール	S-Type 形式
2	ub3694u.bat	ユーザプログラムモード マイクロカーネル バッチファイル	MS-DOS バッチファイル
3	ul3694u.xcl	ユーザプログラムモード マイクロカーネル リンクサブファイル	リンケージエディタ コマンドファイル
4	uGenu.c	ユーザプログラムモード マイクロカーネル メイン ソースファイル	C ソースファイル
5	CmdFunc.c	コマンド関数 ソースファイル	C ソースファイル（共通）
6	CmdFunc.h	CmdFunc.c の関数の原型宣言	C ヘッダファイル
7	Commands.h	コマンド ID の定義	C ヘッダファイル
8	DeviceInfo.h	デバイス特有の情報（問い合わせ応答データ）	C ヘッダファイル
9	Extern.h	関数、変数の外部参照定義	C ヘッダファイル
10	uGenu.h	uGenu.c の関数の原型宣言	C ヘッダファイル
11	io3694.h	SCI、ポートのレジスタ定義	C ヘッダファイル
12	KDevice.h	デバイス特有の情報（カーネル配置定義 など）	C ヘッダファイル
13	KStruct.h	構造体定義　など	C ヘッダファイル
14	KTypes.h	型定義　など	C ヘッダファイル
15	H8runtime.lib	ランタイム ライブラリ	カーネル再生成時に必要 （ファイル提供していないので、ライブラリ構築ツールを使用し、生成する必要あり） ＊BuildAll.bat 参照

6.2.3 メインカーネル

表 6-4 メインカーネルのファイル構成

項番	ファイル名	内容	ファイルの種類
1	Genu3694.cde	ユーザプログラムモード メインカーネル ロードモジュール	バイナリ形式
2	b3694u.bat	ユーザプログラムモード メインカーネル バッチファイル	MS-DOS バッチファイル
3	l3694u.xcl	ユーザプログラムモード メインカーネル リンクサブファイル	リンケージエディタ コマンドファイル
4	FDTUMain.c	ユーザプログラムモード メインカーネル ソースファイル	C ソースファイル
5	CopyFunc.c	ユーザプログラムモード メインカーネル コピー関数 ソースファイル	C ソースファイル
6	CmdFunc.c	コマンド関数 ソースファイル	C ソースファイル（共通）
7	CmdFunc.h	CmdFunc.c の関数の原型宣言	C ヘッダファイル
8	Commands.h	コマンド ID の定義	C ヘッダファイル
9	DeviceInfo.h	デバイス特有の情報（問い合わせ応答データ）	C ヘッダファイル
10	Extern.h	関数、変数の外部参照定義	C ヘッダファイル
11	FDTBMain.h	FDTBMain.c の関数の原型宣言	C ヘッダファイル
12	io3694.h	SCI、ポートのレジスタ定義	C ヘッダファイル
13	KDevice.h	デバイス特有の情報（カーネル配置定義 など）	C ヘッダファイル
14	KStruct.h	構造体定義　など	C ヘッダファイル
15	KTypes.h	型定義　など	C ヘッダファイル

6.2.4 書き込みカーネル

表 6-5 書き込みカーネルメインのファイル構成

項番	ファイル名	内容	ファイルの種類
1	Genw3694.cde	書き込みカーネル ロードモジュール	バイナリ形式
2	b3694w.bat	書き込みカーネル バッチファイル	MS-DOS バッチファイル
3	l3694w.xcl	書き込みカーネル リンクサブファイル	リンケージエディタ コマンドファイル
4	F3694w.src	書き込みカーネル ソースファイル	アセンブラ ソースファイル
5	FDTWrite.c	書き込みカーネル メイン ソースファイル	C ソースファイル
6	WriteTime.c	書き込み用ｳｪｲﾄ時間算出 ｿｰｽﾌｧｲﾙ	C ソースファイル
7	Commands.h	ｺﾏﾝﾄﾞ ID の定義	C ヘッダファイル
8	F3694asm.inc	フラッシュメモリレジスタ定義　など	アセンブラ インクルードファイル
9	io3694.h	SCI、ポートのレジスタ定義	C ヘッダファイル
10	KAlg.h	書き込み／消去の関数定義	C ヘッダファイル
11	KDevice.h	デバイス特有の情報（カーネル配置定義 など）	C ヘッダファイル
12	KStruct.h	構造体定義　など	C ヘッダファイル
13	KTypes.h	型定義　など	C ヘッダファイル
14	H8runtime.lib	ランタイム ライブラリ	カーネル再生成時に必要 （ファイル提供していないので、ライブラリ構築ツールを使用し、生成する必要あり） ＊BuildAll.bat 参照

6.2.5 消去カーネル

表 6-6 消去カーネルのファイル構成

項番	ファイル名	内容	ファイルの種類
1	Gene3694.cde	消去カーネル ロードモジュール	バイナリ形式
2	b3694e.bat	消去カーネル バッチファイル	MS-DOS バッチファイル
3	l3694e.xcl	消去カーネル リンクサブファイル	リンケージエディタ コマンドファイル
4	F3694e.src	消去カーネル ソースファイル	アセンブラ ソースファイル
5	FDTErase.c	消去カーネル ソースファイル	C ソースファイル
6	EraseTime.c	消去用ウェイト時間算出 ソースファイル	C ソースファイル
7	Commands.h	コマンド ID の定義	C ヘッダファイル
8	F3694asm.inc	フラッシュメモリレジスタ定義 など	アセンブラ インクルードファイル
9	io3694.h	SCI、ポートのレジスタ定義	C ヘッダファイル
10	KAlg.h	書き込み／消去の関数定義	C ヘッダファイル
11	KDevice.h	デバイス特有の情報（カーネル配置定義 など）	C ヘッダファイル
12	KStruct.h	構造体定義　など	C ヘッダファイル
13	KTypes.h	型定義　など	C ヘッダファイル
14	H8runtime.lib	ランタイム ライブラリ	カーネル再生成時に必要 （ファイル提供していないので、ライブラリ構築ツールを使用し、生成する必要あり） ＊BuildAll.bat 参照

6.3　　プログラムとファイルの関係

プログラムとファイルの関係を 表 6-7に示します。

表 6-7 プログラムとファイルの関係

プログラム名	バッチ ファイル	ソース ファイル	ヘッダ ファイル	サブコマンド ファイル	ライブラリ ファイル	出力ファイル
メイン処理モジュール （ユーザプログラムモード）	bTest.bat	GenTest.c strt3694.src	GenTest.h io3694.h KDevice.h KStruct.h KTypes.h	l3694t.xcl	―	3694Test.mot
マイクロカーネル （ユーザプログラムモード）	ub3694u.bat	Ugenu.c CmdFunc.c	Ugenu.h CmdFunc.h Commands.h DeviceInfo.h Extern.h io3694.h KDevice.h KStruct.h KTypes.h	ul3694u.xcl	H8runtime.lib	uGenU.mot
メインカーネル （ユーザプログラムモード）	b3694u.bat	FDTUMain.c CopyFunc.c CmdFunc.c	FDTUMain.h CmdFunc.h Commands.h DeviceInfo.h Extern.h io3694.h KDevice.h KStruct.h KTypes.h	l3694u.xcl	―	Genu3694.cde
書き込みカーネル	b3694w.bat	FDTWrite.c WriteTime.c F3694w.src	F3694asm.inc Commands.h io3694.h KAlg.h KDevice.h KStruct.h KTypes.h	l3694w.xcl	H8runtime.lib	Genw3694.cde
消去カーネル	b3694e.bat	FDTErase.c EraseTime.c F3694e.src	F3694asm.inc Commands.h io3694.h KAlg.h KDevice.h KStruct.h KTypes.h	l3694e.xcl	H8runtime.lib	Gene3694.cde
マイクロカーネル （ブートモード）	ub3694.bat	Ugen.c CmdFunc.c strt3694.src	Ugen.h CmdFunc.h	ul3694.xcl	―	uGen3694.cde
メインカーネル （ブートモード）	b3694m.bat	FDTBMain.c CmdFunc.c	FDTBMain.h CmdFunc.h	l3694m.xcl	―	Genm3694.cde
全ビルドバッチファイル	BuildAll.bat	―		―	H8runtime.lib	―

［注］ブートモードプログラムと全ビルドバッチファイルを含みます。

6.4 ビルド

提供されたプログラムを使うときはビルドは必要ありません。動作周波数が異なる場合など、再作成が必要なときはビルドが必要です。

ビルドを実行するとすべての生成ファイルは削除されます。現在のファイルが必要になることもありますので、コピーを作成してから、ビルドを実行してください。

6.4.1 SET コマンド

ビルドを実行するとき、環境の設定が必要になります。下記のコマンドを set.bat バッチファイルに作成し、ビルドの前に、set を実行してください。

```
set PATH=%PATH%;C:¥Hew3¥Tools¥Renesas¥H8¥6_1_0¥Bin
set CH38=C:¥Hew3¥Tools¥Renesas¥H8¥6_1_0¥Include
set CH38TMP=C:¥Hew3¥Tools¥Renesas¥H8¥6_1_0¥Ctemp
```

6.4.2 ライブラリファイル

ビルドを実行するとき、ライブラリファイルが必要です。ライブラリファイルは、ファイル提供していないので、ライブラリ構築ツールを使用し、生成する必要があります。コマンドは BuildAll.bat を参照してください。下記のコマンドで、ライブラリファイルを作成します。BuildAll を実行することにより、ライブラリファイルを含めすべてのプログラムを作成することができます。

```
REM -- LIBRARY COMPILE --
"%CH38%¥..¥bin¥lbg38" -output=H8runtime.lib -head=runtime -cpu=300HN
```

6.4.3 出力ファイル

アクセサリからコマンドプロンプトウィンドウを開き、バッチファイルを実行し、それぞれ出力ファイルを作成します。

表 6-8 バッチファイルと出力ファイル

項番	プログラム	バッチファイル	出力ファイル	出力ファイルの種類
1	メイン処理モジュール	bTest.bat	3694Test.mot	S タイプファイル
2	マイクロカーネル	ub3694u.bat	uGenU.mot	S タイプファイル
3	メインカーネル	b3694u.bat	Genu3694.cde	バイナリロードモジュールファイル
4	書き込みカーネル	b3694w.bat	Genw3694.cde	バイナリロードモジュールファイル
5	消去カーネル	b3694e.bat	Gene3694.cde	バイナリロードモジュールファイル
6	ライブラリ	BuildAll.bat	H8runtime.lib	ライブラリファイル

6.5 モジュール一覧

モジュール一覧を 表 6-9に示します。

表 6-9 モジュール一覧

プログラム	ファイル	モジュール	機能
メイン処理モジュール	Strt2378.src	startup	開始
	GenTest.c	main	メイン処理
		WDTStop	ウォッチドックタイマ停止
		InitSCI	SCI 初期設定
		Get	受信
		Put	送信
		JumpCopy	コピー分岐
マイクロカーネル	Ugenu.c	StartFDTUserKernel	FDT 開始
		PrepareFDTUserKernel	FDT 準備
		PrepareRAM	RAM 準備
		CmdFunc	コマンド関数
	CmdFunc.c	ReferFunc	参照関数
		SelectDevice	デバイス選択
		SelectClockMode	クロックモード選択
		SetNewBaudRate	新ボーレート設定
		SendSciBreak	ブレーク送信
		RequestBootPrgSts	プログラム状態
		SendAck	ACK 送信
		GetCmdData	コマンド読み出し
メインカーネル	FDTUMain.c	Kernelmain	メインカーネル
		ProcessCommand	コマンド処理
	CopyFunc.c	CopyFunction	コピー関数
	CmdFunc.c	RequestBootPrgSts	プログラム状態
		SumcheckUserArea	ユーザエリアサムチェック
		SendAck	ACK 送信
		CheckBlank	ブランクチェック
		ReadMemory	メモリ読み出し
		GetCmdData	コマンド読み出し
書き込みカーネル	FDTWrite.c	WriteFLASH	フラッシュ書き込み
		RequestBootPrgSts	プログラム状態
		SendAck	ACK 送信
		GetWriteData	書き込みデータ受信
	WriteTime.c	WriteWaitTime	書き込み待ち時間
	F3694w.src	flash_write	データ書き込み
		CalCount	時間計算
消去カーネル	FDTErase.c	EraseFLASH	フラッシュ消去
		RequestBootPrgSts	プログラム状態
		SendAck	ACK 送信
		GetEraseData	消去データ受信
	EraseTime.c	EraseWaitTime	消去待ち時間
	F3694e.src	block_erase	ブロック消去
		CalCount	時間計算

6.6　　モジュール階層構造

モジュール階層構造を 図 6-1に示します。

```
VECT（0x0000）　リセットベクタ
  └startup（0x400）（Strt3694.src）　開始（メイン処理モジュール）
      └main（GenTest.c）　メイン処理
          ├WDTStop　ウォッチドックタイマ停止
          ├InitSCI　 SCI 初期設定
          └JumpCopy　コピー分岐
              └CopyFDT（0x7600）　マイクロカーネルコピー
                  └StartFDTUserKernel（uGenu.c）　マイクロカーネル開始（マイクロカーネル）
                      ├PrepareFDTUserKernel　マイクロカーネル準備
                      │　 ├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
                      │　 │　 │　 ├ErrorCode　エラーコードマクロ
                      │　 │　 │　 └Put　送信
                      │　 │　 ├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
                      │　 │　 │　 ├ErrorCode　エラーコードマクロ
                      │　 │　 │　 ├Put　送信
                      │　 │　 │　 └SendAck　ACK 送信
                      │　 │　 ├SetNewBaudRate　新ビットレート選択
                      │　 │　 │　 ├ErrorCode　エラーコードマクロ
                      │　 │　 │　 ├Put　送信
                      │　 │　 │　 ├SendAck　ACK 送信
                      │　 │　 │　 ├SendSciBreak　ブレーク送信
                      │　 │　 │　 │　 ├Get　受信
                      │　 │　 │　 │　 └SendAck　ACK 送信
                      │　 │　 │　 └Get　受信
                      │　 │　 ├RequestBootPrgSts　プログラム状態
                      │　 │　 │　 └Put　送信
                      │　 │　 └Put　送信
                      │　 └PrepareRAM　RAM 準備
                      │　　 ├Get　受信
                      │　　 ├ErrorCode　エラーコードマクロ
                      │　　 ├Put　送信
                      │　　 └SendAck　ACK 送信
                      └RAMStartAddress（0xF780）　RAM 開始アドレス
                    （続く）
```

図 6-1 モジュール階層構造（1）

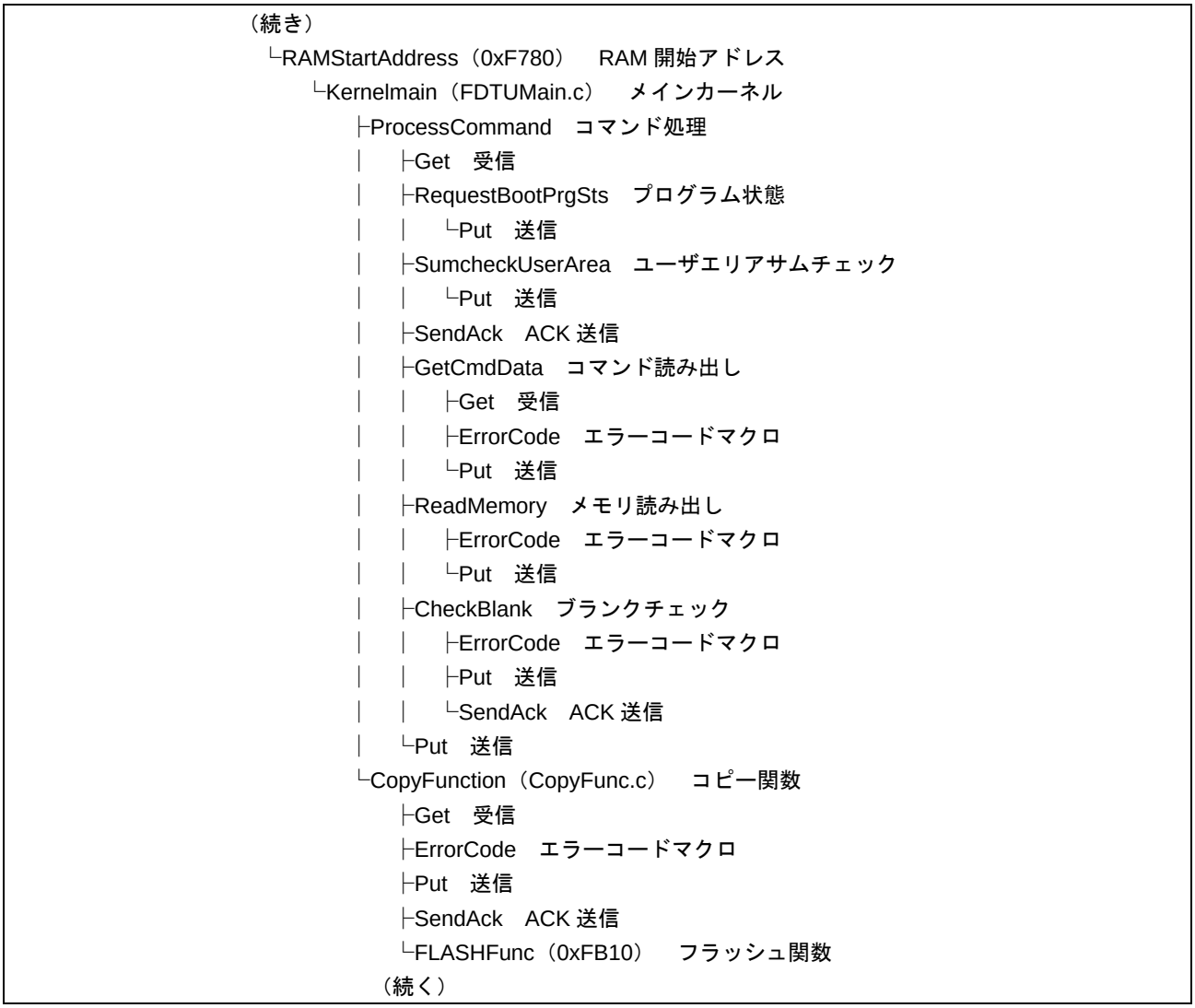


図 6-1 モジュール階層構造(2)

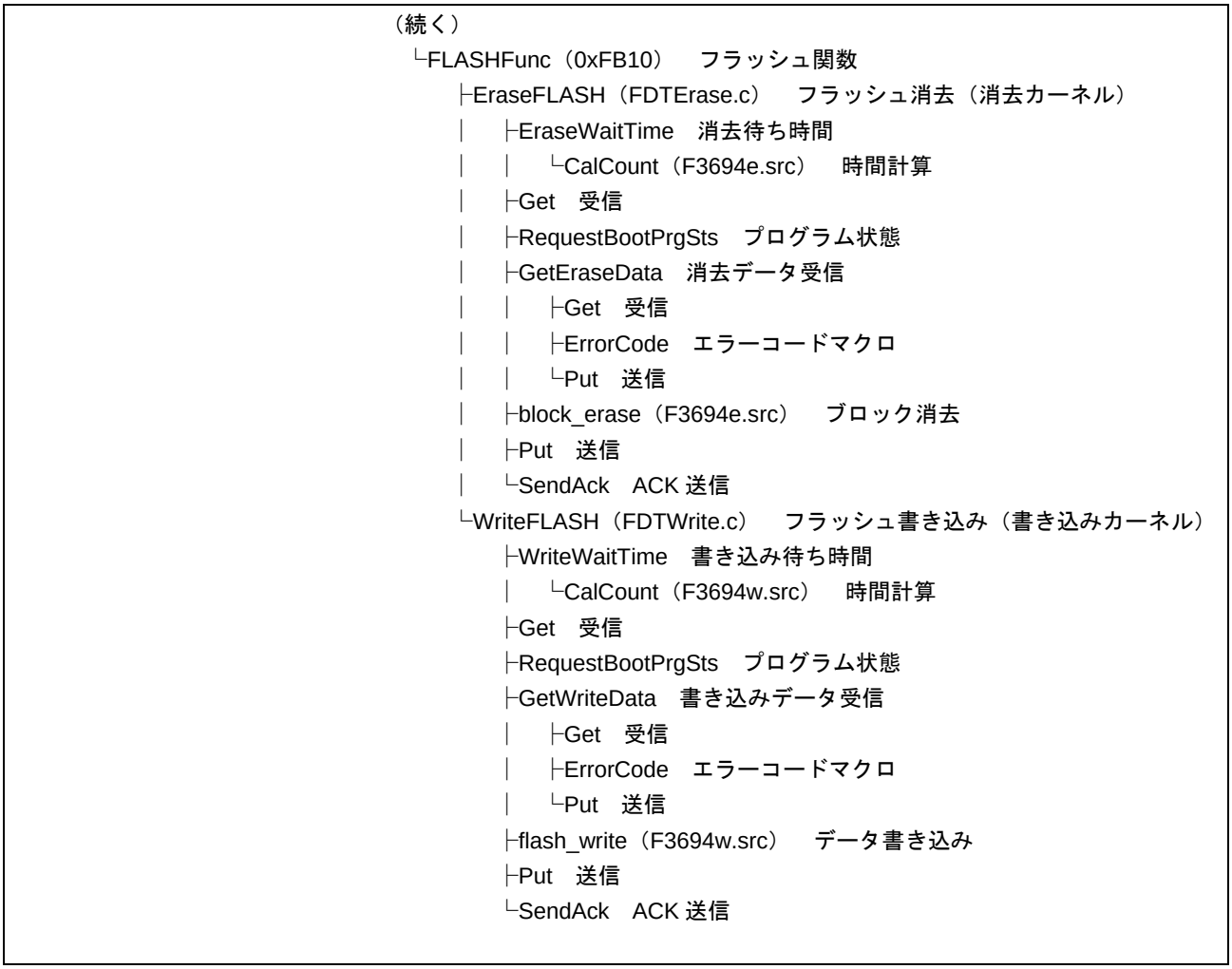


図 6-1 モジュール階層構造(3)

6.7 プログラムの処理フロー

サンプルプログラムの処理フローを 図 6-2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。

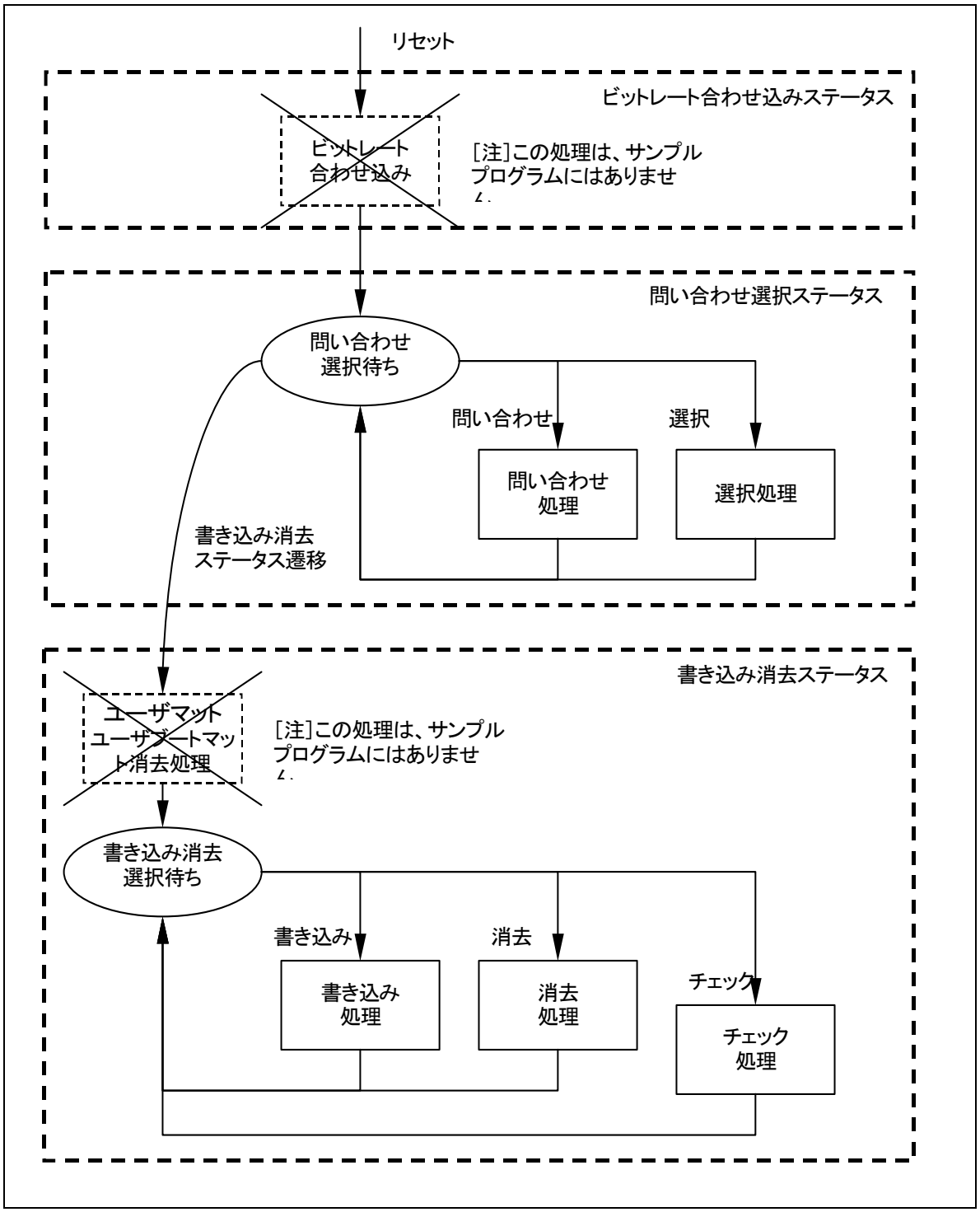


図 6-2 プログラムの処理フロー

6.8 ユーザプログラムモードコマンドシーケンス

ユーザプログラムモードのデバイス接続時、フラッシュ書き込み時、フラッシュ消去時のフラッシュ開発ツールキットとマイコンとのコマンドシーケンスを図 6-3、図 6-4、図 6-5に示します。

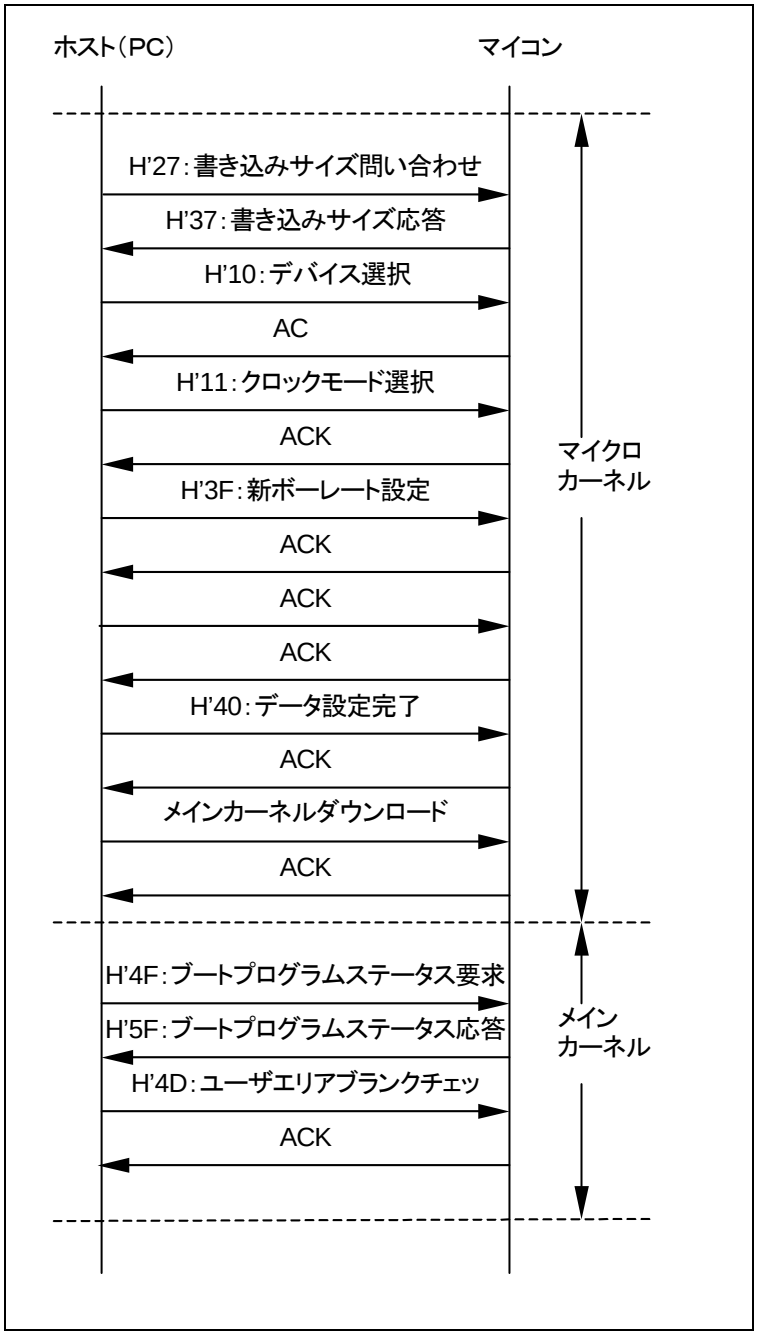


図 6-3 デバイス接続時

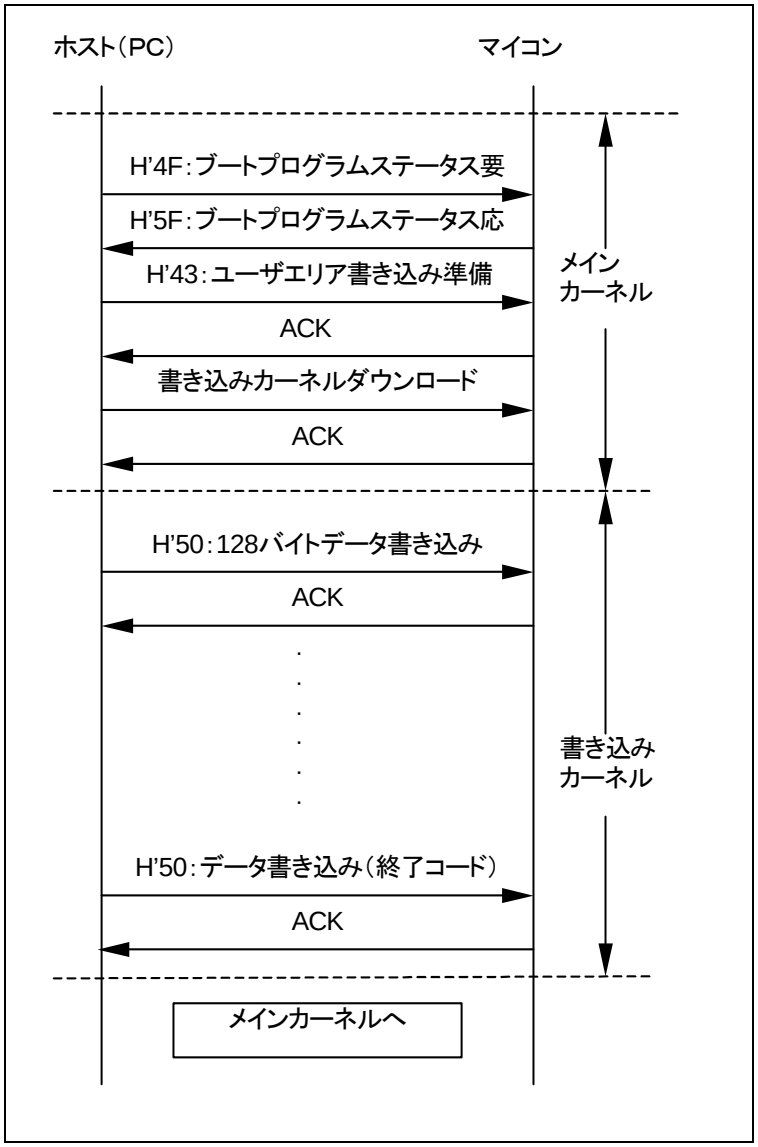


図 6-4 フラッシュ書き込み時

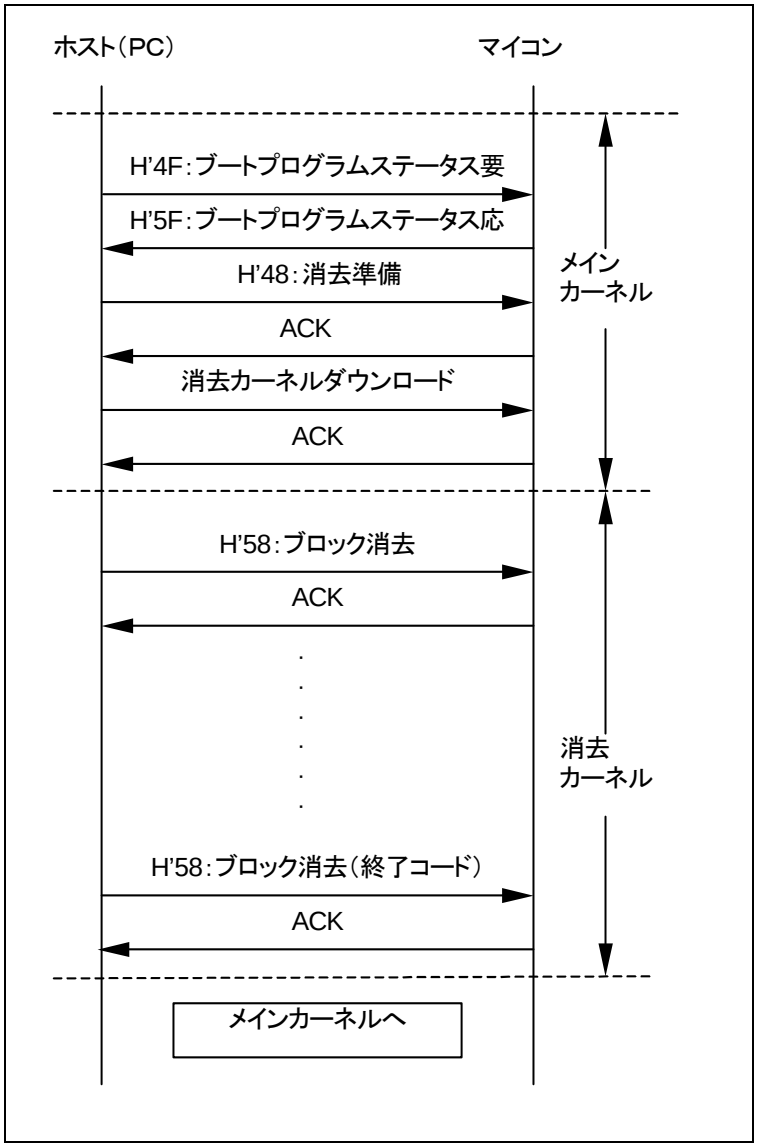


図 6-5 フラッシュ消去時

6.9 プログラムシーケンス

サンプルプログラムのプログラムシーケンスを説明します。プログラムシーケンスの概要を 表 6-10に示します。

表 6-10 プログラムシーケンス

項番	シーケンス	処理
1	前準備	・ ユーザプログラムモードでフラッシュ開発ツールキットを使用する場合は、あらかじめメイン処理モジュール、マイクロカーネルをフラッシュメモリに書き込んでおく必要があります ・ フラッシュメモリが全面消去されても問題ないときは、フラッシュ開発ツールキットのブートモードを使用して書き込むことができます
2	メイン処理モジュール	・ パワーオンリセットで、リセットベクタからメイン処理モジュールへ分岐 ・ スタックポインタを初期設定 ・ SCI を初期設定 ・ SCI インタフェース関数（Put, Get）のアドレスとサイズをスタックに積む（マイクロカーネルへの引数渡し） ・ マイクロカーネルへ分岐
3	マイクロカーネル	・ デバイス仕様問い合わせ選択コマンドの受信、応答処理 ・ データ設定完了コマンド受信の後、メインカーネルを受信し、RAM に格納 ・ SCI インタフェース関数（Put, Get）を RAM に格納 ・ メインカーネルへ分岐
4	メインカーネル	・ 書き込み消去チェックコマンドの受信、応答処理 ・ 書き込み消去準備コマンド受信の後、書き込み消去カーネルを受信し、RAM に格納 ・ 書き込み消去カーネルを呼ぶ
5	書き込みカーネル	・ 書き込みデータ、書き込み先アドレスを受信 ・ フラッシュメモリへの書き込み ・ 書き込み終了データ受信の後、メインカーネルに戻る
6	消去カーネル	・ 消去ブロック番号を受信 ・ フラッシュメモリをブロック消去 ・ 消去終了データ受信の後、メインカーネルに戻る

6.9.1 前準備

前準備の流れを以下に示します。

（1） ユーザプログラムモードでフラッシュ開発ツールキットを使用する場合は、あらかじめメイン処理モジュール、マイクロカーネルをフラッシュメモリに書き込んでおく必要があります。
フラッシュメモリが全面消去されても問題ないときは、フラッシュ開発ツールキットのブートモードを使用して書き込むことができます。あるいは、PROM ライタを使ってライタモードで書き込むこともできます。

（2） メイン処理モジュール、マイクロカーネルをフラッシュメモリに書き込んだ後、マイコンの端子をユーザプログラムモードに設定して、リセットをかけてユーザプログラムモードを起動します。

6.9.2 メイン処理モジュール

メイン処理モジュールの流れを以下に示します。メイン処理モジュールは ROM 上で動作します。

（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、ウォッチドックタイマ停止（WDTStop）、SCI 初期設定（InitSCI）を呼んで、コピー分岐（JumpCopy）へ分岐します。
ウォッチドックタイマ停止（WDTStop）はウォッチドックタイマを停止させ、SCI 初期設定（InitSCI）は SCI のビットレートを設定します。

（4） マイクロカーネルコピー（CopyFDT）は、SCI インターフェース関数（Get、Put）のアドレスとサイズを変数エリアに設定し、マイクロカーネルコピー（CopyFDT）を介してマイクロカーネル開始（StartFDTUserKernel）へ分岐します。

6.9.3 マイクロカーネル

マイクロカーネルの流れを以下に示します。マイクロカーネルは ROM 上で動作します。

（1） 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。

（2） 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

（3） 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）
クロックモード選択（SelectClockMode）
新ビットレート選択（SetNewBaudRate）

（4） 書き込み消去ステータス遷移コマンドで、コマンド処理を終了し、RAM 準備（PrepareRAM）を呼びます。

（5） RAM 準備（PrepareRAM）は、メインカーネル（Kernelmain）を受信して RAM に格納し、SCI インターフェース関数（Get、Put）を RAM へ転送します。

（6） RAM に転送したメインカーネル（Kernelmain）に分岐します。

6.9.4 メインカーネル

メインカーネルの流れを以下に示します。メインカーネルは RAM 上で動作します。

（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。
メモリリード（ReadMemory）
ユーザエリアのサムチェック（SumcheckUserArea）
ユーザエリアのブランクチェック（CheckBlank）
ブートプログラムステータス問い合わせ（RequestBootPrgSts）

（2） ユーザエリア書き込み選択コマンド、消去選択コマンドを受信すると、コピー関数（CopyFunction）を呼び出します。

（3） コピー関数（CopyFunction）は、コマンドに対応する書き込みカーネル（WriteFLASH）、消去カーネル（EraseFLASH）を受信し、RAM に格納し、書き込みカーネル（WriteFLASH）、消去カーネル（EraseFLASH）を呼びます。

6.9.5 書き込みカーネル

書き込みカーネルの流れを以下に示します。書き込みカーネルは RAM 上で動作します。

（1） フラッシュ書き込み（WriteFLASH）は、書き込み待ち時間（WriteWaitTime）で待ち時間を計算します。
その後、コマンドを受信します。

（2） 受信したコマンドがブートプログラムステータス問い合わせコマンドならば、プログラム状態（RequestBootPrgSts）を呼びます。

（3） 受信したコマンドが 128 バイト書き込みコマンドならば、書き込みデータ受信（GetWriteData）で書き込みデータを受信します。

（4） 受信した書き込みデータが書き込み終了（アドレスデータが H’FFFFFFFF）でなければ、データ書き込み（flash_write）で、フラッシュメモリに書き込みます。

（5） 受信した書き込みデータが書き込み終了ならば、書き込みを終了し、メインカーネルへ戻ります。

6.9.6 消去カーネル

消去カーネルの流れを以下に示します。消去カーネルは RAM 上で動作します。

（1） フラッシュ消去（EraseFLASH）は、消去待ち時間（EraseWaitTime）で待ち時間を計算します。
その後、コマンドを受信します。

（2） 受信したコマンドがブートプログラムステータス問い合わせコマンドならば、プログラム状態（RequestBootPrgSts）を呼びます。

（3） 受信したコマンドがブロック消去コマンドならば、消去データ受信（GetEraseData）で消去ブロック番号を受信します。

（4） 受信した消去ブロック番号が消去終了（H’FF）でなければ、ブロック消去（block_erase）で、フラッシュメモリをブロック消去します。

（5） 受信した消去ブロック番号が消去終了ならば、消去を終了し、メインカーネルへ戻ります。

6.10　メモリマップ

サンプルプログラムのプログラムシーケンスに対応するメモリマップを 表 6-11に示します。

表 6-11 プログラムシーケンスとメモリマップ

ROM /RAM	アドレス	シーケンス				
		メイン処理モジュール	マイクロカーネル	メインカーネル	書き込みカーネル	消去カーネル
ROM	H'0000～	リセットベクタ			書き込み可能エリア	消去可能エリア
	H'0400～	メイン処理モジュール				
	H'7600～	マイクロカーネル	マイクロカーネル			
RAM	H'F780～		メインカーネル Put, Get	メインカーネル Put, Get	メインカーネル Put, Get	メインカーネル Put, Get
	H'FB08～		変数			
	H'FB10～			書き込みカーネルまたは消去カーネル	書き込みカーネル	消去カーネル
	H'FF10～	グローバル変数	グローバル変数	グローバル変数	グローバル変数	グローバル変数
	～H'FF7F	スタック	スタック	スタック	スタック	スタック

7. サンプルプログラムのソース

サンプルプログラムの主なソースを以下に示し、機能、処理を説明します。

7.1 ヘッダファイル

サンプルプログラムは、以下のヘッダファイルを使っています。

7.1.1 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。

```
                                        /* 20MHz 9600bps */
//#define MA_BRR_SCI                0x40        /* Bit rate register channel 3 */
                                        /* 9.8MHz 9600bps */
#define MA_BRR_SCI                0x1f        /* Bit rate register channel 3 */
```

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、9.8MHzなので、9600bpsにするために、MA_BRR_SCIを 31（0x1f）に設定しています。動作周波数とBBRレジスタの設定値の関係を 表 7-1に示します。

表 7-1 動作周波数と BBR レジスタの設定値（ビットレート 9600（bit/s）のとき）

動作周波数 φ （MHz）	BRR の設定	誤差 （%）
8	25	0.16
9.8304	31	0.00
10	32	－1.36
12	38	0.16
12.288	39	0.00
14	45	－0.93
14.7456	47	0.00
16	51	0.16
17.2032	55	0.00
18	58	－0.69
19.6608	63	0.00
20	64	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、バッチファイルまたは 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。

7.1.2 IO レジスタ定義（io3694.h）

SCI モジュールと WDT に関係するレジスタとビットを定義しています。

```
/************************************************************************/
/*      H8/3694F,36014F,36024F,36064F Internal I/O Include File         */
/************************************************************************/

/************************************************************************/
/*                      SCI                                             */
/*----------------------------------------------------------------------*/
/*                              CHANNEL 3                               */
/************************************************************************/
#define SCI_SMR             (*(volatile unsigned char *)0xFFA8)
#define SCI_BRR             (*(volatile unsigned char *)0xFFA9)
#define SCI_SCR3            (*(volatile unsigned char *)0xFFAA)
  #define TE                        (unsigned char)0x20
  #define RE                        (unsigned char)0x10
  #define TE_RE                     (unsigned char)(TE | RE)
#define SCI_TDR             (*(volatile unsigned char *)0xFFAB)
#define SCI_SSR             (*(volatile unsigned char *)0xFFAC)
  #define TDRE                      (unsigned char)0x80
  #define RDRF                      (unsigned char)0x40
  #define ERR_CLR                   (unsigned char)0xC7
  #define TEND                      (unsigned char)0x04
#define SCI_RDR             (*(volatile unsigned char *)0xFFAD)

/************************************************************************/
/*                      I/O Port                                        */
/*----------------------------------------------------------------------*/
/*                          Port 1              (in use : TXD)          */
/************************************************************************/
#define PMR1                (*(volatile unsigned char *)0xFFE0)
  #define TXD                       (unsigned char)0x02

/************************************************************************/
/*                      I/O Port                                        */
/*----------------------------------------------------------------------*/
/*                          Port 2              (in use : P22/TXD at SCI Break) */
/************************************************************************/
#define PCR2                (*(volatile unsigned char *)0xFFE5)
  #define PCR22                     (unsigned char)0x04
```

```
#define PDR2                    (*(volatile unsigned char *)0xFF5D)
  #define P22                           (unsigned char)0x04

/************************************************************************/
/*                      WDT                                             */
/*----------------------------------------------------------------------*/
/*                                                                      */
/************************************************************************/
#define TCSRWD                  (*(volatile unsigned char *)0xFFC0)
```

7.1.3 マクロ定義（KAlg.h）

プログラムで使うラベルを定義しています。ERASE_END でブロック消去コマンドの消去終了を判定します。WRITE_END で 128 バイト書き込みコマンドの書き込み終了を判定します。

```
/* D E F I N E S */
#define LOOP_END                        1
#define bufSize                         0x80
#define BLOCK_NO_ERROR                  0xE1
#define ERASE_END                       0xFF
#define WRITE_END                       0xFFFFFFFF
#define ADDRESS_ERROR                   0xF1
#define ADDRESS_BOUNDARY_ERROR          0xF2
```

7.2　　メイン処理モジュール（Strt3694.src、GenTest.c）

7.2.1　　モジュール階層構造

メイン処理モジュールのモジュール階層構造を 図 7-1に示します。

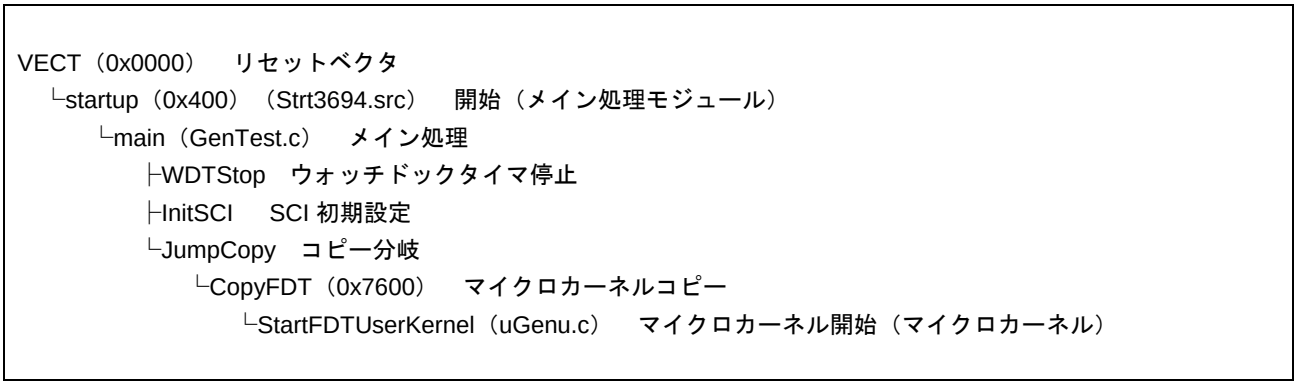


図 7-1 メイン処理モジュールのモジュール階層構造

7.2.2　　リセットベクタ（GenTest.c、GenTest.h）

CVECT セクションにリセットベクタ H’400 を設定します。

（１）GenTest.c

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

（２）GenTest.h

```
/*
   This value specifies the address to where to program
   will JMP on startup. This value should be the link address
   for the associated asm file.
*/
#define RESET_JMP_ADDRESS          0x400
```

7.2.3　　スタック（Strt3694.src）

スタックポインタを H’FF80 に設定します。

```
        MOV.L     #H'FF80, ER7
```

7.2.4 メイン処理（main）

```
WDTStop();
InitSCI();
JumpCopy();
```

ウォッチドックタイマを停止し、SCI を初期設定し、マイクロカーネルへ分岐します。

7.2.5 コピー分岐（JumpCopy）

（１） JumpCopy

```
/* Create Function Pointer & assign address to it */
FuncPtr CopyFDT = (FuncPtr)USER_KERNEL_LINK_ADDRESS;
                                        /*This is where the linker has put the code*/
/* Store structure elements */
ParamFDT.GetFuncPtr = (GetPtr)Get;
ParamFDT.PutFuncPtr = (PutPtr)Put;
ParamFDT.PutSize = (WORD)((DWORD)Dummy - (DWORD)Put);
ParamFDT.GetSize = (WORD)((DWORD)Put - (DWORD)Get);

ParamFDT.RAMStartAddress = RAM_START_ADDRESS;

/* Jump to CopyFDT */
(*CopyFDT)((paramFDT *)&ParamFDT);
```

（２） GenTest.h

```
/*
  These defines relate to the USER kernel.
  In order to call the user kernel we must know the address it was
  linked at.
*/
#define USER_KERNEL_LINK_ADDRESS        0x7600
```

ParamFDT に SCI インターフェース関数（GET、Put）のアドレスとサイズを設定し、CopyFDT に分岐します。CopyFDT のアドレスは USER_KERNEL_LINK_ADDRESS で示す H’7600 です。H’7600 にはマイクロカーネルのマイクロカーネル開始（StartFDTUserKernel）が書き込まれています。

7.3　　マイクロカーネル（uGenu.c、CmdFunc.c）

7.3.1　　モジュール階層構造

マイクロカーネルのモジュール階層構造を 図 7-2に示します。

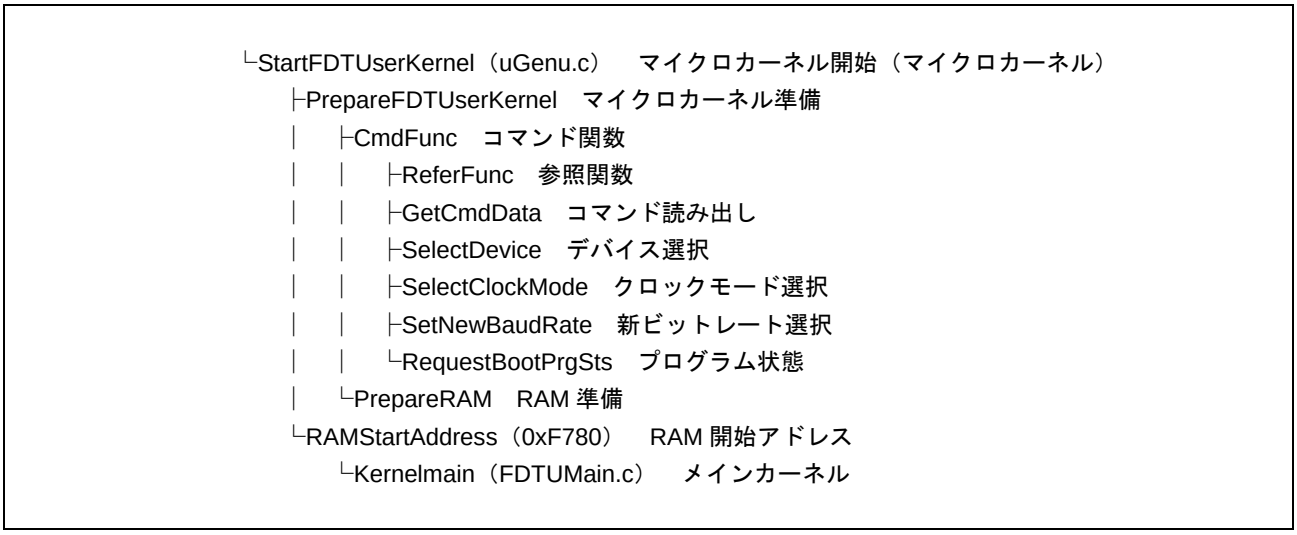


図 7-2 マイクロカーネルのモジュール階層構造

7.3.2　　マイクロカーネル開始（StartFDTUserKernel）

（１）　StartFDTUserKernel

```
PrepareFDTUserKernel(parameters);

/* Pass execution to the main kernel */
(*((FuncPtr)parameters->RAMStartAddress))(parameters);
```

（２）　GenTest.h

```
/*Use these defines to specify the range of RAM FDT can use*/
#define RAM_START_ADDRESS          0xF780
```

マイクロカーネル準備を呼び、RAM RAMStartAddress にあるモジュールを呼びます。マイクロカーネル準備は、メインカーネルを RAM RAMStartAddress 以降に格納します。RAMStartAddress は H’F780 に設定されています。

7.3.3 マイクロカーネル準備（PrepareFDTUserKernel）

```
while(1){
        /* Command Function */
        CmdFunc(parameters->PutFuncPtr, parameters->GetFuncPtr);
        /*Prepare RAM */
        if(!PrepareRAM(parameters, &ParamFDT)){
                break;
        }
}
```

コマンド関数を呼び、コマンド関数が終了（書き込み消去ステータス遷移コマンドを受信）すると、RAM 準備を呼びます。RAM 準備が終了（メインカーネル受信格納が正常終了）すると、マイクロカーネル準備が終了します。

7.3.4 コマンド関数（CmdFunc、CmdFunc.c）

コマンド関数の骨組みを以下に示します。

```
while(1){
    /* Acquisition of a command ID */
    add_sum = Get(&commandID, 1);
    switch(commandID)
    {
        case finishDataSet:
            return;
        case supportDevice:
            ReferFunc(commandID, deviceData, sizeof(deviceData), Put);
            break;
        case selectDevice:
                SelectDevice(cmdBuf.bdata, Put);
            break;
        case referClockMode:
            ReferFunc(commandID, clockModeData, sizeof(clockModeData), Put);
            break;
        case selectClockMode:
                SelectClockMode(cmdBuf.bdata, Put);
            break;
        case referRatio:
            ReferFunc(commandID, ratioData, sizeof(ratioData), Put);
            break;
        case setNewBaudRate:
            SetNewBaudRate(commandID, (BaudRate *)cmdBuf.bdata, Put, Get);
            break;
        case referUserRomInfo:
            ReferFunc(commandID, usrRomData, sizeof(usrRomData), Put);
            break;
        case referEraseBlockInfo:
            ReferFunc(commandID, eraseBlkData, sizeof(eraseBlkData), Put);
            break;
        case referWriteSystem:
            ReferFunc(commandID, writeSysData, sizeof(writeSysData), Put);
            break;
        case referFrequency:
            ReferFunc(commandID, frequencyData, sizeof(frequencyData), Put);
            break;
```

```
                case referWriteSize:
                        ReferFunc(commandID, writeSizeData, sizeof(writeSizeData), Put);
                        break;
                case requestBootPrgSts:
                        RequestBootPrgSts(Put);
                        break;
                default:
                        cBuff[0] = COMMAND_ERROR;
                        cBuff[1] = commandID;
                        Put(cBuff, 2);
                        break;
        }
    }
```

コマンド関数は、書き込み消去ステータス遷移コマンドを受信すると処理を終了します。それ以外のコマンドはそれぞれの処理をし、コマンド受信待ちになります。

それぞれのコマンド処理モジュールは、CmdFunc.c にあります。CmdFunc.c は、マイクロカーネルだけでなくメインカーネルのコマンド処理モジュールも含みます。マイクロカーネルかメインカーネルかの区分は、#ifdef で判定しています。

また、このコマンド処理モジュールはユーザプログラムモードだけでなくブートモードにも使われます。ユーザプログラムモードでは SCI インターフェース関数（Get、Put）の引数がありますが、ブートモードでは引数がありません。

7.3.5 RAM 準備（PrepareRAM）

```
kernelPos = (BYTE *)parameters->RAMStartAddress;
/* Receive size of User Kernel module from host */
add_sum = Get((BYTE *)&kernelSize, sizeof(kernelSize));
/* Download kernel to beginning of allowable RAM */
add_sum+= Get(kernelPos, kernelSize);
/* Adjust start position of RAM */
parameters->RAMStartAddress += kernelSize;

/*
        Copy the Get and Put functions into memory, first Get() then Put()
*/
pSrc = (BYTE *)parameters->GetFuncPtr;
pDest = (BYTE *)parameters->RAMStartAddress;
/* Now perform the copy */
for(i = 0; i < parameters->GetSize; i++, pSrc++, pDest++)
{
        *pDest = *pSrc;
}

parameters->RAMStartAddress += parameters->GetSize;
pSrc = (BYTE *)parameters->PutFuncPtr;
pDest = (BYTE *)parameters->RAMStartAddress;
/* Now perform the copy */
for(i = 0; i < parameters->PutSize; i++, pSrc++, pDest++)
{
        *pDest = *pSrc;
}
```

RAM 準備は、次の処理を行います。

（１） メインカーネルの受信と RAM への格納

（２） Get 関数の RAM へのコピー

（３） Put 関数の RAM へのコピー

それぞれ RAM の開始アドレスとサイズを設定し、RAM に格納します。フラッシュメモリの消去書き込みをするために、RAM 上で実行させます。

7.4 メインカーネル（FDTUMain.c、CmdFunc.c、CopyFunc.c）

7.4.1 モジュール階層構造

メインカーネルのモジュール階層構造を 図 7-3に示します。

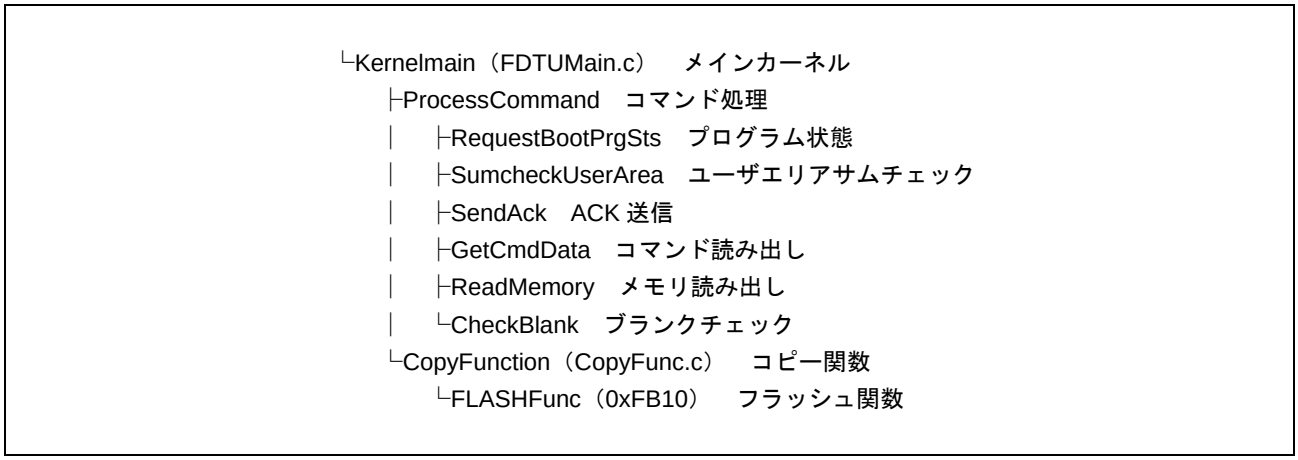


図 7-3 メインカーネルのモジュール階層構造

7.4.2 メインカーネル（Kernelmain）

```
/* Main control processing loop */
while (1)
{
        if(ProcessCommand(&commandID, parameters))
        {
                CopyFunction(commandID, parameters);
        }
}
```

メインカーネルではコマンド処理（ProcessCommand）を繰り返し実行します。コマンド処理で、コマンドを受信し、コマンドを処理します。コピー関数（CopyFunction）はコマンドが、ユーザエリア書き込み選択、消去選択のときのみ呼ばれ、それぞれ消去カーネル、書き込みカーネルを受信し、RAM に格納します。消去カーネル、書き込みカーネルで、書き込み、消去を行い、書き込み、消去が終了すると、再びコマンド処理が呼ばれます。

7.4.3 コマンド処理（ProcessCommand）

コマンド処理の骨組みを以下に示します。

```
/* Acquisition of a command ID */
add_sum = Get(commandID, 1);
switch(*commandID)
{
        case requestBootPrgSts:
                RequestBootPrgSts(Put);
                break;
        case sumcheckUserArea:
                SumcheckUserArea(Put);
                break;
        case prepareErase:
        case prepareUserAreaWrite:
                return(TRUE);
        case readMemory:
                        ReadMemory(cmdBuf.bdata, Put);
                break;
        case checkBlank:
                CheckBlank(Put);
                break;
        default:
                cBuff[0] = COMMAND_ERROR;
                cBuff[1] = *commandID;
                Put(cBuff, 2);
                break;
}
return(FALSE);
```

消去選択コマンド、書き込み選択コマンドのときは、TRUE を返します。それ以外のコマンドは、それぞれコマンド処理モジュールで処理され、FALSE を返します。

コマンド処理モジュールは、CmdFunc.c にあります。

7.4.4 コピー関数（CopyFunction）

（１）CopyFunction

```
BYTE *funcAddress = (BYTE *)FUNC_START, add_sum;
FLASHFuncPtr FLASHFunc = (FLASHFuncPtr)FUNC_START;

/* Acquire size of function to be downloaded */
add_sum = Get((BYTE *)&size, sizeof(size));
/* Download function to RAM address received */
add_sum+= Get(funcAddress, size);

/* Pass execution to the FLASH function */
(*FLASHFunc)(Put, Get);
```

（２）KDevice.h

```
#define FUNC_START          0xFB10      /* Write/Erase function start position */
#define WRITE_DATA          0xFE90      /* write-data area start position */
```

消去カーネル、書き込みカーネルを受信し、FUNC_START で示す H’FB10 から始まる RAM に格納します。格納後は、FUNC_START を呼び出し、消去または書き込みを実行します。消去、書き込みの判定は、コマンド処理（ProcessCommand）で受信したコマンドに従います。

7.5　　消去カーネル（FDTErase.c、EraseTime.c、F3694e.src）

7.5.1　　モジュール階層構造

消去カーネルのモジュール階層構造を 図 7-4に示します。

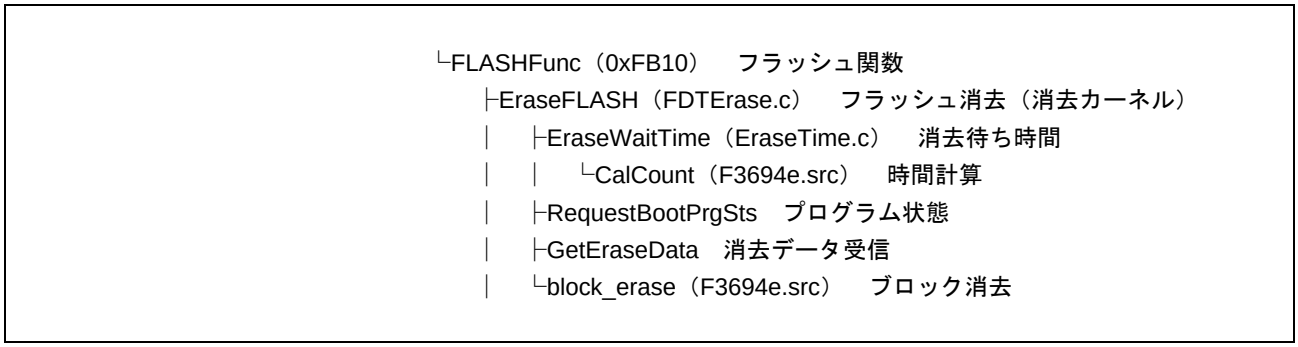


図 7-4 消去カーネルのモジュール階層構造

7.5.2　　フラッシュ消去（EraseFLASH）

フラッシュ消去の骨組みを以下に示します。

```
/* Waiting time calculation of erase processing */
EraseWaitTime();

do{
        /* Acquisition of a command ID */
        add_sum = Get(&commandID, 1);
        /* Is it a demand of boot status command? */
        if (commandID == requestBootPrgSts){
                RequestBootPrgSts(Put);
        }else{
                /* Acquisition of command data    */
                if(GetEraseData(&blk_no, add_sum, Put, Get)){
                        return;
                }
                if (blk_no != ERASE_END){
                        /* Erase start */
                        rsts = block_erase(blk_no);
                        if(rsts){
                                if (rsts == BLOCK_NO_ERROR){
                                        cBuff[1] = ERASE_BLOCK_NO_ERROR;
                                }else{
                                        cBuff[1] = ERASE_ERROR;
                                }
                                return;
                        }
```

```
                }else{
                        end_flg = LOOP_END;
                }
            }
        }while(!end_flg);
```

消去待ち時間（EraseWaitTime）で消去の待ち時間を計算します。

次にコマンドを受信します。コマンドがプログラムステータス問い合わせならば、プログラム状態（RequestBootPrgSts）でステータスを応答します。

コマンドがブロック消去で、消去終了でなければ、ブロック番号を指定してブロック消去（block_erase）を呼びます。

消去終了ならば、消去を終了してメインカーネルに戻ります。

7.5.3 消去待ち時間（EraseWaitTime、CalCount）

（１）EraseWaitTime

```
SWES_W = CalCount(1)+1;
SWEC_W = ESUS_W = CalCount(100)+1;
ESUC_W = EC_W = CalCount(10)+1;
ES_W = CalCount(10000);
EVS_W = CalCount(20)+1;
EVC_W = CalCount(4)+1;
DLCH_W = CalCount(2)+1;
```

（２）CalCount

```
FREQ:   .EQU        H'FF10          ; Frequency(Global data) import from "KDevice.h"
LCNT:   .EQU        D'600           ; 1μs loop counter
; _CalCount     .EQU    $
                SUB.W       E0,E0
                MOV.W       @FREQ,R1            ;frequency
                MULXU.W     R1,ER0
                MOV.W       #LCNT,R1
                DIVXU.W     R1,ER0
                RTS
```

消去待ち時間（EraseWaitTime）は消去実行時レジスタのビットを 1 にセットまたは 0 にクリア後の待ち時間（μs）を計算します。時間計算（CalCount）は与えられた周波数を元に、1 命令 6 サイクルかかるとして何命令処理時間分待てばよいかを計算します。周波数は、新ビットレート選択で与えられた値をもとに計算した動作周波数です。H'FF10 に 10KHz 単位で格納されています。

フラッシュ開発ツールを使わず専用のインターフェースでフラッシュメモリを消去するときは、動作周波数の設定、消去待ち時間計算は、ユーザマニュアルを参照してプログラムを作成してください。

計算した消去待ち時間の値の例を 表 7-2に示します。

表 7-2 消去待ち時間の値の例(動作周波数 20MHz)

消去待ち時間	変数	時間（μs）	ソフトウェアループ回数
SWE ビットセット後	SWES_W	1	4
SWE ビットクリア後	SWEC_W	100	334
ESU ビットセット後	ESUS_W	100	334
ESU ビットクリア後	ESUC_W	10	34
E ビットセット後	ES_W	10000（10ms）	33333
E ビットクリア後	EC_W	10	34
EV ビットセット後	EVS_W	20	67
EV ビットクリア後	EVC_W	4	14
ダミーライト後	DLCH_W	2	7

7.6　書き込みカーネル（FDTWrite.c、WriteTime.c、F3694w.src）

7.6.1　モジュール階層構造

書き込みカーネルのモジュール階層構造を 図 7-5に示します。

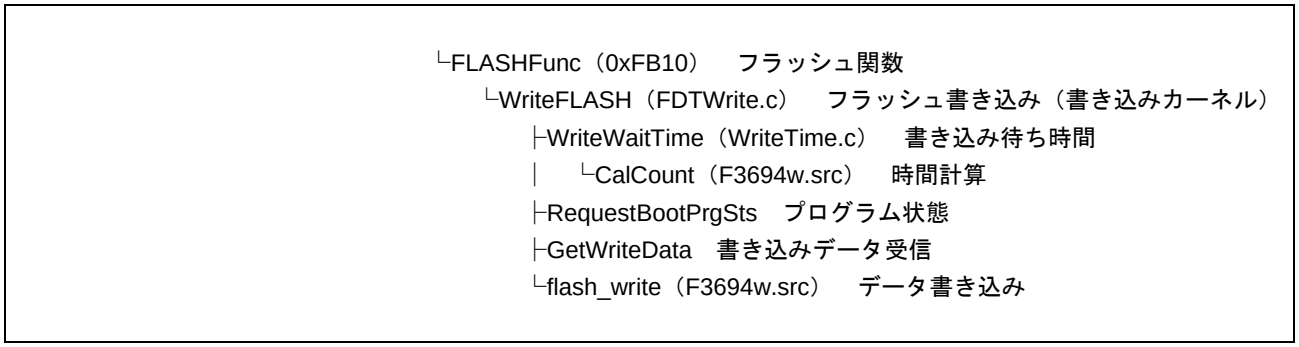


図 7-5 書き込みカーネルのモジュール階層構造

7.6.2　フラッシュメモリ書き込み（WriteFLASH）

フラッシュメモリ書き込みの骨組みを以下に示します。

```
/* Waiting time calculation of write processing */
WriteWaitTime();

do{
        /* Acquisition of a command ID */
        add_sum = Get(&commandID, 1);
        /* Is it a demand of boot status command? */
        if (commandID == requestBootPrgSts){
                RequestBootPrgSts(Put);
        }else{
                /* Acquisition of command data    */
                if(GetWriteData(&pAddress, add_sum, Put, Get)){
                        return;
                }
                if (pAddress != WRITE_END){
                        /* Write-in start */
                        rsts = flash_write((BYTE *)WRITE_DATA, (BYTE *)pAddress);
                        if(rsts)
                        {
                                if (rsts == ADDRESS_ERROR ||
                                        rsts == ADDRESS_BOUNDARY_ERROR)
                                {
                                        cBuff[1] = WRITE_ADDRESS_ERROR;
                                }else{
                                        cBuff[1] = WRITE_ERROR;
```

```
                    }
                    return;
                }
            }else{
                end_flg = LOOP_END;
            }
        }
    }while(!end_flg);
```

書き込み待ち時間（WriteWaitTime）で書き込みの待ち時間を計算します。

次にコマンドを受信します。コマンドがプログラムステータス問い合わせならば、プログラム状態（RequestBootPrgSts）でステータスを応答します。

コマンドが 128 バイト書き込みで、書き込み終了でなければ、書き込みアドレスと書き込みデータを指定してデータ書き込み（flash_write）を呼びます。

書き込み終了ならば、書き込みを終了してメインカーネルに戻ります。

7.6.3 書き込み待ち時間（WriteWaitTime、CalCount）

（１）WriteWaitTime

```
P10S_W = CalCount(10);
P30S_W = CalCount(30);
PSUS_W = CalCount(50);
SWEC_W = PSUS_W * 2;
P200S_W = SWEC_W * 2;
PSUC_W = PC_W = PVS_W = DLCH_W = CalCount(5);
```

（２）CalCount

```
FREQ:   .EQU        H'FF10          ; Frequency(Global data) import from "KDevice.h"
LCNT:   .EQU        D'600           ; 1μs loop counter
_CalCount       .EQU    $
                SUB.W       E0,E0
                MOV.W       @FREQ:16,R1         ;frequency
                MULXU.W     R1,ER0
                MOV.W       #LCNT,R1
                DIVXU.W     R1,ER0
                RTS
```

書き込み待ち時間（WriteWaitTime）は書き込み実行時レジスタのビットを 1 にセットまたは 0 にクリア後の待ち時間（μs）を計算します。時間計算（CalCount）は与えられた周波数を元に、1 命令 6 サイクルかかるとして何命令処理時間分待てばよいかを計算します。周波数は、新ビットレート選択で与えられた値をもとに計算した動作周波数です。H'FF10 に 10KHz 単位で格納されています。

フラッシュ開発ツールを使わず専用のインターフェースでフラッシュメモリを書き込みするときは、動作周波数の設定、書き込み待ち時間計算は、ユーザマニュアルを参照してプログラムを作成してください。

計算した書き込み待ち時間の値の例を 表 7-2に示します。

表 7-3 書き込み待ち時間の値の例（動作周波数 20MHz）

消去待ち時間	変数	時間（μs）	ソフトウェアループ回数
PSU ビットセット後	PSUS_W	50	166
PSU ビットクリア後	PSUC_W	5	16
追加書き込み時書き込み時間	P10S_W	10	33
P ビットセット後 書き込み時書き込み時間 （書き込み回数 1～6）	P30S_W	30	100
書き込み時書き込み時間 （書き込み回数 7～1000）	P200S_W	200	666
SWE ビットクリア後	SWEC_W	100	34
P ビットクリア後	PC_W	5	16
PV ビットセット後	PVS_W	5	16
ダミーライト後	DLCH_W	5	16

8. 書き込み／消去カーネル（提供プログラム）の使い方

フラッシュ開発ツールキットインタフェース部分（メイン処理モジュール、マイクロカーネル）を必要とせず、フラッシュメモリ書き込み／消去の論理モジュール部分のみをユーザ独自の開発プログラムと結合させて使用することもできます。書き込み／消去カーネルの一部でもある論理モジュールについて以下に説明します。

8.1 書き込み

8.1.1 使用ファイル一覧

ファイル名	内容	言語
F3694w.src	128 バイト書き込み処理　ソースファイル	アセンブラ
F3694asm.inc	128 バイト書き込み処理／ブロック消去処理 共通ヘッダファイル	アセンブラ

8.1.2 モジュール仕様

名称	128 バイト書き込み処理
型式	unsigned char flash_write(unsigned char *data, unsigned char *adr)
機能	128 バイト書き込み処理
引数	data：書き込みデータ先頭アドレス adr：書き込み先アドレス
戻り値	処理結果 正常終了：H’00 最大書き込み回数エラー：H’01 書き込み済エラー：H’02 FWE エラー：H’D1 FLER エラー：H’D2 書き込みアドレスエラー：H’F1 128 バイト境界アドレスエラー：H’F2
入力	書き込みウェイト時間
処理	128 バイト単位での書き込みを実行する 詳細はハードウェアマニュアルのプログラム/プログラムベリファイフロー参照してください
注意事項	モジュール使用時は、事前に書き込みウェイト時間（ソフトウェアループ回数）をクローバル変数領域に設定しておいてください。

8.2 消去

8.2.1 使用ファイル一覧

ファイル名	内容	言語
F3694e.src	ブロック消去処理　ソースファイル	アセンブラ
F3694asm.inc	128 バイト書き込み処理／ブロック消去処理 共通ヘッダファイル	アセンブラ

8.2.2 モジュール仕様

名称	ブロック消去処理
型式	unsigned char block_erase (unsigned char blk_no)
機能	ブロック消去処理
引数	blk_no：ブロック番号
戻り値	処理結果 正常終了：H'00 消去ｴﾗｰ：H'01 FEW ｴﾗｰ：H'D1 FLER ｴﾗｰ：H'D2 ブロック番号エラー：H'E1 最大消去回数エラー：H'E2
入力	消去ウェイト時間
処理	ブロック毎の消去を実行する 詳細はハードウェアマニュアルのイレース/イレースベリファイフロー参照してください
注意事項	モジュール使用時は、事前に消去ウェイト時間（ソフトウェアループ回数）をクローバル変数領域に設定しておいてください。

フラッシュ開発ツールキット　アプリケーションノート（応用編）
ユーザプログラムモード（H8/3694F）

発行年月日　2005年10月28日　　Rev.1.00

発行　　株式会社ルネサス テクノロジ　営業統括部
〒100-0004　東京都千代田区大手町2-6-2

編集　　株式会社 ルネサス ソリューションズ　ツール開発部

フラッシュ開発ツールキット
アプリケーションノート（応用編）

ルネサス エレクトロニクス株式会社
神奈川県川崎市中原区下沼部1753 〒211-8668

RJJ06J0003-0100