本ドキュメントは Cypress (サイプレス) 製品に関する情報が記載されております。本ドキュメントには、「MB」から始まるシリーズ名，品名およびオーダ型格が記載されておりますが、これらはすべて「CY」から始まるシリーズ名，品名およびオーダ型格として、新規および既存のお客様に引き続き提供してまいります。

オーダ型格の調べ方について

1. www.cypress.com/pcnにアクセスしてください。
2. SEARCH PCNS フィールドに、オーダ型格などのキーワードを入力し、「Apply」をクリックしてください。
3. 該当するタイトル(Title)をクリックしてください。
4. 「Affected Parts List」ファイルを開いてください。
   当該ファイルに記載されている各種変更情報をご利用ください。

詳しいお問い合わせ先

Cypress 製品およびそのソリューションの詳細につきましては、お近くの営業所へお問い合わせください。

サイプレスについて

サイプレスは、世界で最も革新的な車載や産業機器、スマート家電、民生機器および医療機器製品向けに、最先端の組み込みシステム ソリューションを提供するリーディング カンパニーです。サイプレスのマイクロコントローラーや、アナログ IC、ワイヤレスおよび USB ベースのコネクティビティ ソリューション、高い信頼性と高性能を提供するメモリ製品は、各種機器メーカーの差異化製品の開発と早期市場参入を支援します。サイプレスは、ベストクラスのサポートと開発リソースをグローバルに提供することで、彼らが従来市場を破壊しまったく新しい製品カテゴリを歴史的なスピードで市場投入できるよう支援します。詳細はサイプレスのウェブサイト (japan.cypress.com) をご覧ください。

AN204377

FM3/FM4 ファミリ IEC61508 SIL2 セルフテストライブラリ

関連製品ファミリ: セクション 2 を参照

このアプリケーションノートは、サイプレス製マイコン FM3 ファミリ、FM4 ファミリを対象とし、IEC61508 の SIL2 相当のソフトウェアによるマイコン向け自己診断処理(Self Test Library、以下 STL) の利用を検討されている方を対象としています。

Contents

1 はじめに 1
1.1 この文書について 1
1.2 背景 1
1.3 開発・評価環境 2
1.4 注意事項 2
2 対象製品 3
3 IEC61508 の概要 4
3.1 用語の定義 4
3.2 安全ライフサイクル 5
3.3 SIL(安全度水準) 6
3.4 故障の定義 6
3.5 ハードウェア安全性評価と アーキテクチャ制約 7
3.6 SFF(安全側故障比率) 7
4 STL 概要 8
4.1 診断の対応範囲 8
4.2 診断の種類 8
4.3 故障検知 9
4.4 STL の構成 9
5 API 概要 11
5.1 ROM 診断 11
5.2 RAM 診断 12
5.3 CPU 診断 13
5.4 BUS 診断 18
6 STL 性能 19
6.1 ROM/RAM サイズ 19
6.2 診断処理速度 19
7 参考文書 22
8 その他の情報 22
A 付録 23
A.1 フラッシュの CRC コード作成方法 23
9 改訂履歴 27
セールス, ソリューションおよび法律情報 28

1 はじめに

1.1 この文書について

このアプリケーションノートは、サイプレス製マイコン FM3 ファミリ、FM4 ファミリを対象とし、IEC61508 の SIL2 相当のソフトウェアによるマイコン向け自己診断処理(Self Test Library、以下 STL) の利用を検討されている方を対象としています。

注意事項： STL がご必要な場合は、弊社代理店や営業にお問い合わせください。

1.2 背景

近年、各産業分野においてシステムの安全に対する考え方やその手法が益々重要視されており、国際的に標準化する動きがあります。IEC(International Electrotechnical Commission) が定めた IEC61508 は様々な分野を対象としており、他の安全規格で制定されていない分野は全て IEC61508 の適用が可能です。IEC61508 が規定されたことにより、製品開発において規格の対応を要求されることが多くなってきています。

1.3 開発・評価環境

本 STL は表 1 に示す環境で開発・評価されています。

表 1. 開発・評価環境

マイコン	[FM3] MB9BF506R [FM4] MB9BF568R
統合開発環境	IAR Embedded Workbench for ARM6.60 kickstart KEIL μVision V5.0.5.15
評価ボード	[FM3(80MHz,Flash アクセス 2wait)] MB9BF506R-SK MCB9B500 [FM4(160MHz, フラッシュアクセラレータ Enable)] SK-FM4-U120-9B560
最適化	[IAR]高(バランス) [KEIL]Level3

1.4 注意事項

提供する STL は認証機関による認証を受けたものではないため、本 STL を変更なく利用しても認証を取得できるとは限りません。また、本 STL は SFF が 90%以上かつ 99%未満での SIL2 を達成することが前提として設計されています。(3.5 ハードウェア安全性評価とアーキテクチャ制約)

2 対象製品

本アプリケーションノートに記載されている内容の対象製品は、下記のとおりです。

(FM3:TYPE0)

シリーズ名	品種型格（パッケージサフィックスは除く）
MB9B500B	MB9BF504NB, MB9BF505NB, MB9BF506NB MB9BF504RB, MB9BF505RB, MB9BF506RB
MB9B400A	MB9BF404NA, MB9BF405NA, MB9BF406NA MB9BF404RA, MB9BF405RA, MB9BF406RA
MB9B300B	MB9BF304NB, MB9BF305NB, MB9BF306NB MB9BF304RB, MB9BF305RB, MB9BF306RB
MB9B100A	MB9BF102NA, MB9BF104NA, MB9BF105NA, MB9BF106NA MB9BF102RA, MB9BF104RA, MB9BF105RA, MB9BF106RA

(FM4)

シリーズ名	品種型格（パッケージサフィックスは除く）
MB9B560R	MB9BF566R, MB9BF567R, MB9BF568R MB9BF566N, MB9BF567N, MB9BF568N MB9BF566M, MB9BF567M, MB9BF568M
MB9B460R	MB9BF466R, MB9BF467R, MB9BF468R MB9BF466N, MB9BF467N, MB9BF468N MB9BF466M, MB9BF467M, MB9BF468M
MB9B360R	MB9BF366R, MB9BF367R, MB9BF368R MB9BF366N, MB9BF367N, MB9BF368N MB9BF366M, MB9BF367M, MB9BF368M
MB9B160R	MB9BF166R, MB9BF167R, MB9BF168R MB9BF166N, MB9BF167N, MB9BF168N MB9BF166M, MB9BF167M, MB9BF168M

3 IEC61508 の概要

3.1 用語の定義

このアプリケーションノートで使用する用語とその意味を表 2 にまとめます。

表 2. 用語/意味

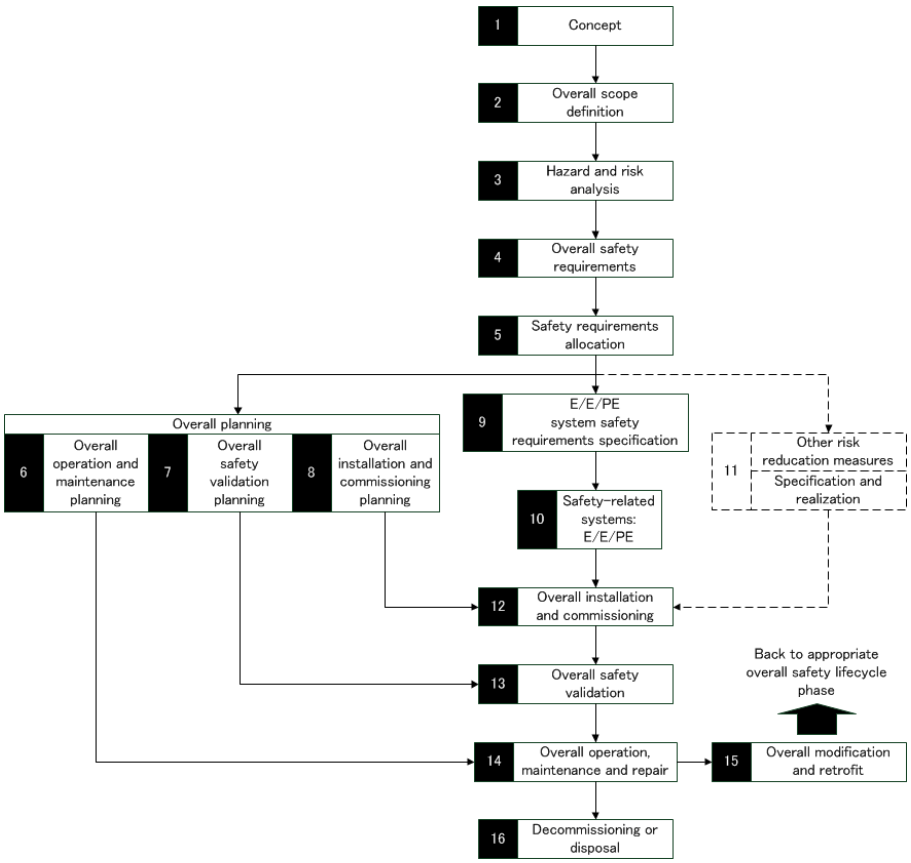
用語	意味
安全関連系	ある特定の制御システムに対し、リスクを許容範囲以下まで軽減させ安全を確保するためのシステム
安全機能	安全関連系が持つ機能
機能安全	安全機能により、リスクを許容範囲以下まで軽減させ安全を確保すること
安全ライフサイクル	安全関連系への要求事項に対して、構想から廃棄に至るまでの規定した業務プロセス。全工程を 16 のフェーズに分けている
SIL	Safety Integrity Level の略。安全度水準。安全関連系が安全機能の遂行に失敗するリスクの度合い。1～4 までの水準があり、4 が最高
プルーフテスト	安全関連系の故障に対し、自動検出が不可能な故障を見つけるための定期テスト
低頻度作動要求モード	安全関連系への動作要求の頻度が、1(回/年)以下かつ、プルーフテスト頻度の 2 倍以下である運用モード
高頻度作動要求モード	安全関連系への動作要求の頻度が、1(回/年)より大きい、またはプルーフテスト頻度の 2 倍よりも大きい運用モード
連続モード	通常運用の一部として安全機能が常に動作している運用モード
PFDavg	Average Probability of dangerous Failure on Demand の略。安全機能作動要求時の危険側故障確率の平均値
PFH	Probability of dangerous Failure per Hour の略。単位時間あたりの危険側故障確率の平均値
ランダム故障	ハードウェアの故障等、時間的に無秩序に発生する故障
決定論的原因故障	ソフトウェアのバグ等、開発工程で故障の原因が入り込み必然的に発生する故障
安全側故障率	安全関連系が故障した際、制御対象が危険な状況を導かない故障
危険側故障率	安全関連系が故障した際、制御対象が危険な状況を導く故障
DC	Diagnostic Coverage の略。自己診断率。全ての危険側故障に対する検出可能な危険側故障の割合
SFF	Safe Failure Fraction の略。安全側故障比率。全ての故障に対する、全ての安全側故障と検出可能な危険側故障の割合
フォールトトレランス	障害発生時に、安全機能を遂行できる機能ユニットの能力
タイプ	以下の 3 つの構成条件を全て満たすものをタイプ A と呼び、タイプ A 以外をタイプ B と呼ぶ。例として、タイプ A には抵抗、コンデンサ、タイプ B には ASIC が挙げられます。マイコンはタイプ B に属すると考えます。 全てのコンポーネントの故障モードを定義できる 障害条件化でサブシステムの挙動を完全に決定できる 検出された及び未検出の危険側故障率の主張を支援できるような、現場に即した、十分に依存可能なデータがある

3.2 安全ライフサイクル

ある特定の制御システムに対し、安全を確保するためのシステムを安全関連系と呼びます。IEC61508 は、この安全関連系に対して要求事項を規定しています。(ただし、電気・電子・プログラマブル電子系(以下、E/E/PES)で構成される安全関連系のみを対象としています)

この安全関連系に対する要求事項は、構想、設計、開発、保守、廃棄に至るまでを規定しており、この一連の体系を安全ライフサイクルと呼びます。(図 1 参照)安全関連系に対する管理や開発等は全て、この安全ライフサイクル上で運用することが求められています。

図 1. 安全ライフサイクル



3.3 SIL(安全度水準)

IEC61508 は安全関連系の安全機能に対して故障発生頻度の許容範囲を定めており、その指標を SIL と呼びます。SIL には 1～4 までの水準があり、各水準には表 3 の通り具体的な数値が規定されています。

表 3. SIL の数値目標

SIL	低頻度作動要求モード (PFDavg)	高頻度作動要求または連続モード (PFH)
4	$\geqq 10^{-5}$ to $< 10^{-4}$	$\geqq 10^{-9}$ to $< 10^{-8}$
3	$\geqq 10^{-4}$ to $< 10^{-3}$	$\geqq 10^{-8}$ to $< 10^{-7}$
2	$\geqq 10^{-3}$ to $< 10^{-2}$	$\geqq 10^{-7}$ to $< 10^{-6}$
1	$\geqq 10^{-2}$ to $< 10^{-1}$	$\geqq 10^{-6}$ to $< 10^{-5}$

SIL は安全ライフサイクルのフェーズ 1～5 で決定され、それ以降のフェーズでは決定された SIL を満たすよう進めていくことになります。

3.4 故障の定義

IEC61508 は、安全関連系の故障をランダム故障と決定論的原因故障の 2 つに分類しています。それらの故障のうち、安全側故障率を λS、危険側故障率を λD、全体の故障率を λ とすると、次の式 (1) が成り立ちます。

$\lambda = \lambda_S + \lambda_D$ ・・・式 (1)

さらにその故障を検出可能と検出不可能なもので分類すると、表 4 の通りです。

表 4. 故障の分類

	検出可能	検出不可能
安全側	自己診断可能な安全側故障	自己診断不可能な安全側故障
危険側	自己診断可能な危険側故障(λ_{DD})	自己診断不可能な危険側故障(λ_{DU})

表 4 より、次の式 (2) が得られます。

$\lambda_D = \lambda_{DD} + \lambda_{DU}$ ・・・式 (2)

自己診断テストによる検出可能な故障の割合である DC は、次の式 (3) で表されます。

$DC = \lambda_{DD} / \lambda_D$ ・・・式 (3)

この DC は SFF (安全故障比率) を考える上で重要になってきます。(SFF については 3.6 を参照してください)

3.5 ハードウェア安全性評価とアーキテクチャ制約

決定論的原因故障については安全ライフサイクルの実施、ランダム故障については安全要求仕様を満たす安全なハードウェアアーキテクチャ設計が求められています。

ハードウェアの安全性を評価する際、表 3 の数値を達成することだけでなく、安全関連系を構成するサブシステムに対し一定のフォールトトレランスを満たさなければいけません。このハードウェアのフォールトトレランス、サブシステムのタイプ、SFF、の 3 つから、主張できる SIL の上限が決められています。これをアーキテクチャ制約と呼びます。表 5 と表 6 にタイプ A とタイプ B のアーキテクチャ制約をそれぞれ示します。N は、N+1 の障害により安全機能の損失を招く場合があることを意味します。

表 5. タイプ A のアーキテクチャ制約

SFF	ハードウェアのフォールトトレランス(N)		
	0	1	2
< 60%	SIL1	SIL2	SIL3
60% - < 90%	SIL2	SIL3	SIL4
90% - < 99%	SIL3	SIL4	SIL4
≧ 99%	SIL3	SIL4	SIL4

表 6. タイプ B のアーキテクチャ制約

SFF	ハードウェアのフォールトトレランス(N)		
	0	1	2
< 60%	許されない	SIL1	SIL2
60% - < 90%	SIL1	SIL2	SIL3
※90% - < 99%	SIL2	SIL3	SIL4
≧ 99%	SIL3	SIL4	SIL4

※弊社が提供する STL のターゲット

3.6 SFF(安全側故障比率)

IEC61508 では、SFF を次の式(4)で定義しています。

$SFF = (\Sigma\lambda_S + \Sigma\lambda_{DD}) / (\Sigma\lambda_S + \Sigma\lambda_D)$・・・式(4)

これは、SFF は全故障発生率に対する安全側故障発生率の比であることを意味します。ただし前提として、自己診断テストが十分有効と仮定し λ_{DD} と λ_S を同等に扱うものとしています。

ここで、式(3)と式(4)から、SFF と DC の関係を表したものが式(5)です。ただし、$\eta = \Sigma\lambda_S / \Sigma\lambda_D$ としています。

$SFF = (\eta + DC) / (\eta + 1)$・・・式(5)

通常、半導体装置は η=1 で考えます。故に、次の式(6)が得られます。

$SFF = 0.5 + 0.5*DC$・・・式(6)

つまり SFF は DC に依存しているため、DC の値が非常に重要です。このため、IEC61508-2 Annex A では、SFF を算出する際に考慮すべきコンポーネントとそのテスト技法、またそのテスト技法を使用した際の DC の最大値が規定されています。

4 STL 概要

4.1 診断の対応範囲

IEC61508-2 Annex A で制定される、診断するべきコンポーネントとそのテスト技法において、弊社が STL として提供する箇所が表 7 です。

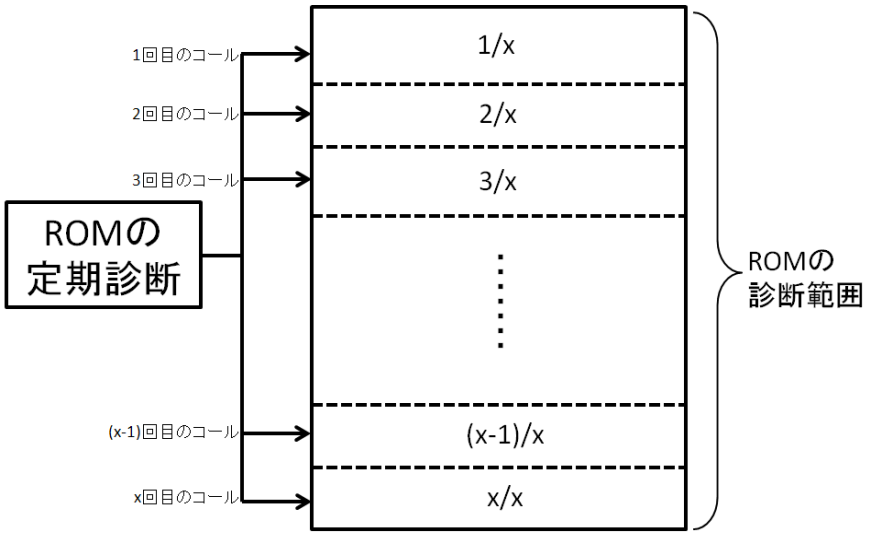
表 7. 診断の対応範囲

コンポーネント	対象範囲	診断技法	DC の最大値
BUS	Data paths	Inspection using test patterns	99%≦
CPU	Processing units	Self-test by software:walking bit(one-channel)	90%≦ to <99%
ROM	Invariable memory ranges	Signature of a double word(16-bit)	99%≦
RAM	Variable memory ranges	RAM test galpat or transparent galpat	99%≦

4.2 診断の種類

診断にはスタートアップ診断と定期診断の 2 種類をコンポーネントごとに用意しています。スタートアップ診断はリセット時に実施することを想定しており、1 度コールすると対象コンポーネントの全範囲を診断します。定期診断はメイン周期で実施することを想定しており、対象コンポーネントに対して分割して診断し、複数回実施することで全範囲を診断できます。例えば ROM 定期診断を x 分割している場合、ROM 定期診断を x 回コールすると図 2 のようになります。

図 2. ROM 定期診断コール時のイメージ(x 分割)



4.3 故障検知

診断時に故障を検知すると、故障回数をカウントし再度同じ診断を連続で実施します。ユーザは、故障検知に対する閾値を設定することができ、故障カウントがこの閾値を超えると故障ステータスがセットされます。故障ステータスはそのコンポーネントの故障の有無を示します。

4.4 STL の構成

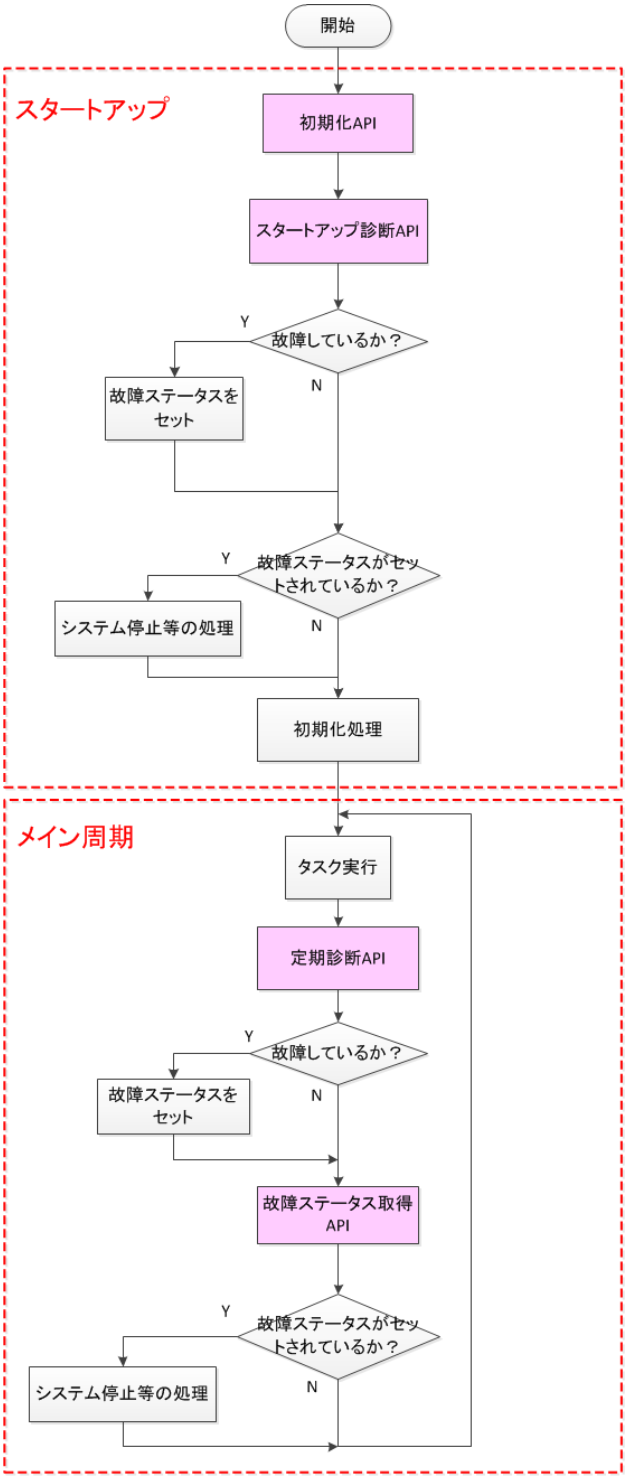
STL はコンポーネントごとに表 8 の API を持ちます。

表 8. API の種類

API	概要
初期化 API	診断時に使用する変数を初期化します。
スタートアップ診断 API	スタートアップ診断を実施します。
定期診断 API	定期診断を実施します。
故障ステータス取得 API	最新の故障ステータスを取得します。

STL の API 使用例を図 3 のフローチャートに示します。この使用例は各コンポーネント共通です。

図 3. API 使用例



5 API 概要

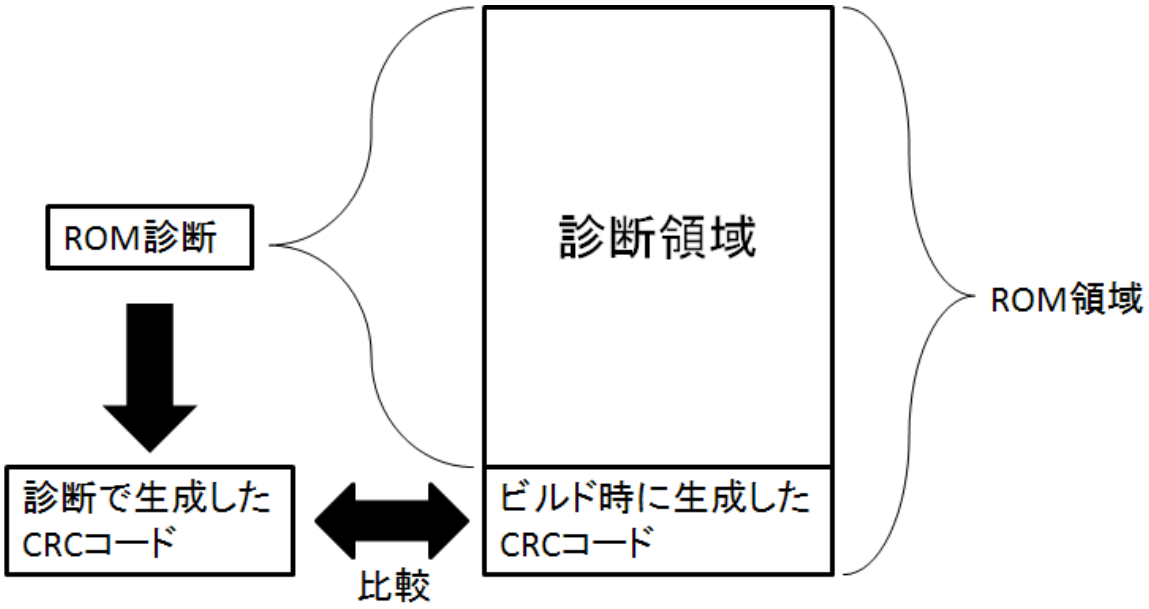
5.1 ROM 診断

5.1.1 診断の概要

ROM に対するテスト技法は”Signature of a double word(16-bit)”を使用します。詳細は IEC61508-7 A4.4 を参照してください。

ROM 診断の概要を図 4 に示します。ビルド時に生成した CRC コードを特定の ROM 領域に配置します。次にシステム起動後、スタートアップ診断もしくは定期診断により CRC コードを生成し、それをビルド時に生成した CRC コードと比較して一致することを確認します。本 STL では CRC16 よりさらに検出精度の高い CRC32 を使用して CRC コードを算出します。診断で生成する CRC コードは、ソフトウェアで算出するか、もしくはハードウェアマクロを使用して算出するかをユーザが選択できます。

図 4. ROM 診断のイメージ



5.1.2 設定可能な内容

ROM 診断の API を使用するにあたり、ユーザが設定可能な内容を下記にまとめます。

1. 診断領域の数
2. 各診断領域のスタートアドレスとエンドアドレス
3. 故障検知に対する閾値
4. 1 度の定期診断で計算するテスト長(byte)

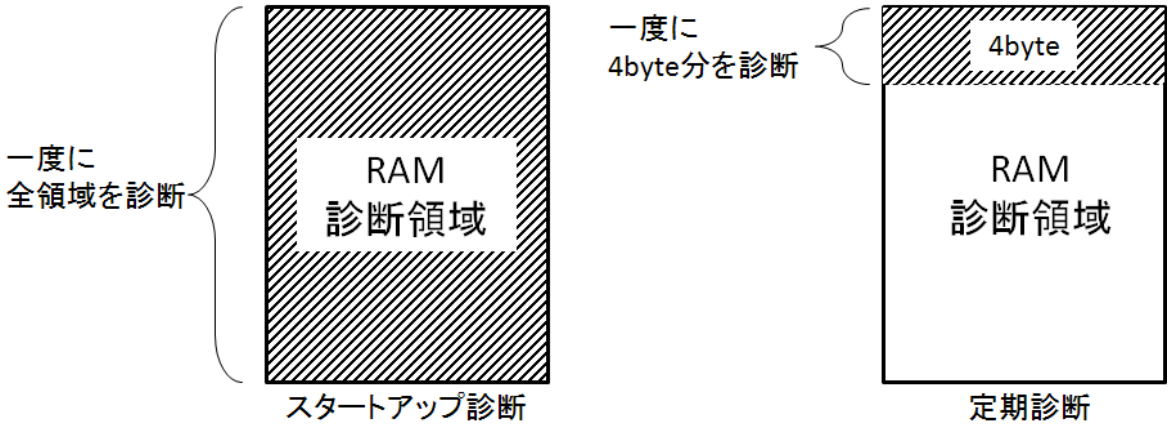
5.2 RAM 診断

5.2.1 診断の概要

RAM に対するテスト技法は”RAM test galpat or transparent galpat”を使用します。詳細は IEC61508-7 A5.3 を参照してください。

スタートアップ診断、定期診断共に galpat を使用します。スタートアップ診断は対象領域全てに対して実施するのに対し、定期診断は 4byte 分を実施します。(図 5 を参照)

図 5. RAM 診断



5.2.2 設定可能な内容

RAM 診断の API を使用するにあたり、ユーザが設定可能な内容を下記にまとめます。

1. 診断領域の数
2. 各診断領域のスタートアドレスとエンドアドレス(スタートアップ診断と定期診断で別々の領域を指定可能)
3. 故障検知に対する閾値

5.3 CPU 診断

5.3.1 診断の概要

CPU に対するテスト技法は”Self-test by software:walking bit(one-channel)”を使用します。詳細は IEC61508-7 A3.2 を参照してください。

CPU 診断は、表 9 に示す 16bit の ARMv7-M Thumb 命令セットの各命令（FM3 ファミリ、FM4 ファミリ共通です。また、Thumb-2 命令セットの検査が必要な場合には、別途用意する必要があります）の正常動作と、レジスタ（R0～R14）の正常動作を診断します。診断方法は、walking bit パターンを入力値として演算し、結果が期待値通りかを確認します。

表 9. ARMv7-M Thumb 命令セット(16bit)

No.	命令	構文	内容
1	ADC	<Rd>, <Rm>+	レジスタの値と C フラグをレジスタの値に加算 Rd = Rd + Rm + C
2	ADD	<Rd>, <Rn>, #<immed_3>+	3 ビットのイミディエート値をレジスタに加算 Rd + Rn + immed_3
3	ADD	<Rd>, #<immed_8>+	8 ビットのイミディエート値をレジスタに加算 Rd + Rd + immed_8
4	ADD	<Rd>, <Rn>, <Rm>+	下位レジスタの値を下位レジスタの値に加算 Rd = Rn + Rm
5	ADD	<Rd>, <Rm>	上位レジスタの値を下位または上位レジスタの値に加算 Rd = Rn + Rm
6	ADD	<Rd>, PC, #<immed_8>*4	PC アドレス + 4 ×（8 ビットのイミディエート値）でレジスタに加算 Rd = PC + 4*immed_8
7	ADD	<Rd>, SP, #<immed_8>*4	SP アドレス + 4 ×（8 ビットのイミディエート値）でレジスタに加算 Rd = SP + 4*immed_8
8	ADD	SP, #<immed_7>*4	4 ×（7 ビットのイミディエート値）を SP に加算 SP = SP + 4*immed_7
9	AND	<Rd>, <Rm>+	レジスタの値をビット単位に AND（論理積） Rd = Rd AND Rm
10	ASR	<Rd>, <Rm>, #<immed_5>+	イミディエートの数値により算術右シフト Rd = Rd >> Rs
11	B<cond>	<target_address_8>	条件付き分岐 if <cond> then PC = (PC + 4) + (SignExtend (target_address_8) * 2)
12	B	<target_address_11>	無条件分岐 PC = (PC + 4) + (SignExtend (target_address_11) * 2)
13	BIC	<Rd>, <Rm>+	ビット・クリア Rd = Rd AND (NOT Rm)
14	BKPT	<immed_8>	ソフトウェア・ブレイクポイント
15	BL	<target_address11>	リンク付き分岐

No.	命令	構文	内容
16	BLX	<Rm>	リンク付き分岐と状態遷移（Rm［ビット 0］は 1 にする）
17	BX	<Rm>	分岐と状態遷移（Rm［ビット 0］は 1 にする）
18	CBNZ	<Rn>, <label>	比較して非 0 で分岐（前方分岐のみ）
19	CBZ	<Rn>, <label>	比較して 0 で分岐（前方分岐のみ）
20	CMN	<Rn>, <Rm>+	レジスタの値の 2 の補数（負数、negation)を別のレジスタの値と比較（Rn －（-Rm））を計算しフラグを更新
21	CMP	<Rn>, #<immed_8>+	レジスタの値と 8 ビットのイミディエートイミディエート値との比較
22	CMP	<Rn>, <Rm>+	レジスタと比較
23	CMP	<Rn>, <Rm>+	上位レジスタを下位または上位レジスタと比較
24	CPSIE CPSID	< i または f > < i または f >	プロセッサ状態の変更 CPSIE は PRIMASK(i)または FAULTMASK(f) をクリアすることで割込みを有効にする CPSID は PRIMASK(i)または FAULTMASK(f)をクリアすることで割込みを無効にする
25	CPY	<Rd>, <Rm>	上位レジスタまたは下位レジスタの値を別の上位レジスタまたは下位レジスタにコピー
26	EOR	<Rd>, <Rm>+	レジスタ値のビット単位で Exclusive OR
27	IT IT<x> IT<x><y> IT<x><y><z>	<cond> <cond> <cond> <cond>	IF-THEN 条件付のブロックを作る命令 条件<cond>に基いて、この命令に続く 1 つから 4 つの命令を条件付きで実行
28	LDMIA	<Rn>!, <registers>	ロード・マルチプル・インクリメント・アフターは、Rn により指定したアドレスから、複数ワードをレジスタにロード
29	LDR	<Rd>, [<Rn>, #<immed_5>*4]	ベース・レジスタのアドレス+5 ビットのイミディエートをオフセットしたメモリから、ワードをロード
30	LDR	<Rd>, [<Rn>, <Rm>]	ベース・レジスタのアドレス+レジスタのオフセットのメモリから、ワードをロード
31	LDR	<Rd>, [PC, #<immed_8>*4]	PC アドレス ＋8 ビットのイミディエートをオフセットしたメモリから、ワードをロード
32	LDR	<Rd>, [SP, #<immed_8>*4]	SP アドレス ＋8 ビットのイミディエートをオフセットしたメモリから、ワードをロード
33	LDRB	<Rd>, [<Rn>, #<immed_5>]	ベース・レジスタのアドレス ＋5 ビットのイミディエートをオフセットしたメモリから、バイト［7 : 0］をレジスタにロード
34	LDRB	<Rd>, [<Rn>, <Rm>]	ベース・レジスタのアドレス ＋ レジスタオフセットのメモリから、バイト[7 : 0]をレジスタにロード
35	LDRH	<Rd>, [<Rn>, #<immed_5>*2]	ベース・レジスタのアドレス ＋5 ビットのイミディエートをオフセットしたメモリから、ハーフ・ワード[15 : 0]をレジスタにロード

No.	命令	構文	内容
36	LDRH	<Rd>, [<Rn>, <Rm>]	ベース・レジスタのアドレス + レジスタオフセットのメモリから、ハーフ・ワード[15 : 0] をレジスタにロード
37	LDRSB	<Rd>, [<Rn>, <Rm>]	ベース・レジスタのアドレス + レジスタオフセットのメモリから、バイト[7 : 0]をレジスタに符号付でロード
38	LDRSH	<Rd>, [<Rn>, <Rm>]	ベース・レジスタのアドレス + レジスタオフセットのメモリから、ハーフ・ワード[15 : 0]をレジスタに符号付でロード
39	LSL	<Rd>, <Rm>, #<immed_5>+	イミディエートの数値により論理左シフト Rd = Rd << immed_5
40	LSL	<Rd>, <Rs>+	レジスタの値により論理左シフト Rd = Rd << Rs
41	LSR	<Rd>, <Rm>, #<immed_5>+	イミディエートの数値により論理右シフト Rd = Rd >> immed_5
42	LSR	<Rd>, <Rs>+	レジスタの値により論理右シフト Rd = Rd >> Rs
43	MOV	<Rd>, #<immed_8>+	8 ビットのイミディエート値をレジスタに移動
44	MOV	<Rd>, <Rn>+	下位レジスタの値を下位レジスタに移動
45	MOV	<Rd>, <Rm>	上位または下位レジスタの値を上位または下位レジスタに移動
46	MUL	<Rd>, <Rm>+	レジスタの値を乗算 Rd = Rd * Rm
47	MVN	<Rd>, <Rm>+	レジスタの値の否定(1 の補数、complement)をレジスタに移動 Rd = NOT(Rm)
48	NEG	<Rd>, <Rm>+	レジスタの値を負(2 の補数、nagatevi)にしてレジスタへ保存 Rd = 0 -Rm
49	NOP		無操作
50	ORR	<Rd>, <Rm>+	レジスタ値のビット単位で OR(論理和) Rd = Rd OR Rm
51	POP	<registers>	スタックから複数のレジスタをポップ
52	POP	<registers, PC>	スタックから複数のレジスタおよび PC をポップ
53	PUSH	<registers>	複数のレジスタをスタックへプッシュ
54	PUSH	<registers, LR>	複数のレジスタおよび LR をスタックへプッシュ
55	REV	<Rd>, <Rn>	ワードの中でバイト順を反転して、レジスタへコピー Rd = {Rn[7:0], Rn[15:8], Rn[23:16], Rn[31:24]}
56	REV16	<Rd>, <Rn>	2 つのハーフ・ワードの中でそれぞれバイト順を反転して、レジスタへコピー Rd = {Rn[23:16], Rn[31:24], Rn[7:0], Rn[15:8]}
57	REVSH	<Rd>, <Rn>	下位ハーフ・ワード[15:0]の中でバイト順を反転、符号拡張した値をレジスタへコピー Rd = SignExtend ({Rn[7:0], Rn[15:8]})

No.	命令	構文	内容
58	ROR	<Rd>, <Rs>+	レジスタで指定した値だけ右ローテート
59	SBC	<Rd>, <Rm>+	レジスタの値からレジスタの値とボロー(~C)をレジスタの値から減算 Rd = Rd - Rm -NOT(C)
60	SEV		
61	STMIA	<Rn>!, <registers>	連続したメモリ・アドレスに、複数のレジスタをワード単位でストア
62	STR	<Rd>, [<Rn>, #<immed_5>*4]	レジスタのアドレス+5 ビットのイミディエートのオフセットへ、レジスタの値をワードでストア
63	STR	<Rd>, [<Rn>, <Rm>]	ベース・レジスタ+レジスタ・オフセットのアドレスへ、レジスタの値をワードでストア
64	STR	<Rd>, [SP, #<immed_8>*4]	SP アドレス+8 ビットのイミディエートのオフセットへ、レジスタの値をワードでストア
65	STRB	<Rd>, [<Rn>, #<immed_5>]	レジスタのアドレス+5 ビットのイミディエート値のオフセットへ、レジスタの値をバイト[7:0]でストア
66	STRB	<Rd>, [<Rn>, <Rm>]	レジスタのアドレス+レジスタ・オフセットのアドレスへ、レジスタの値をバイト[7:0]でストア
67	STRH	<Rd>, [<Rn>, #<immed_5>*2]	レジスタのアドレス+5 ビットのイミディエートのオフセットへ、レジスタ値をハーフワード[15:0]でストア
68	STRH	<Rd>, [<Rn>, <Rm>]	レジスタのアドレス+レジスタ+オフセットへ、レジスタの値をハーフワード[15:0]でストア
69	SUB	<Rd>, <Rn>, #<immed_3>+	3 ビットのイミディエート値をレジスタから減算 Rd = Rn -immed_3
70	SUB	<Rd>, #<immed_8>+	8 ビットのイミディエート値をレジスタから減算 Rd = Rd -immed_8
71	SUB	<Rd>, <Rn>, <Rm>+	レジスタの値を減算 Rd = Rn - Rm
72	SUB	SP, #<immed_7>*4	4 ×（7 ビットのイミディエート値）を SP から減算 SP = SP - immed_7*4
73	SVC	<immed_8>	8 ビットのイミディエート値の呼び出しコードにより、オペレーティング・システムのサービスを呼び出し （スーパーバイザ・コール）
74	SXTB	<Rd>, <Rm>	レジスタからバイト[7:0]を抽出し、レジスタへ移動して、32 ビットに符号拡張
75	SXTH	<Rd>, <Rm>	レジスタからハーフ・ワード[15:0]を抽出し、レジスタへ移動して、32 ビットに符号拡張
76	TST	<Rn>, <Rm>+	別のレジスタの値と論理積を実行して、レジスタのセットされているビットをテスト Rn AND Rm

No.	命令	構文	内容
77	UXTB	<Rd>, <Rm>+	レジスタからバイト[7:0]を抽出し、レジスタへ移動して、32 ビットにゼロ拡張する
78	UXTH	<Rd>, <Rm>+	レジスタからハーフ・ワード[15:0]を抽出し、レジスタへ移動して、32 ビットにゼロ拡張する
79	WFE		イベント待ち
80	WFI		割込み待ち

ただし、No.49,60,79,80 は対象外としています。

ARMv7-M Thumb 命令セット以外に、FM4 ファミリ用として Cortex-M4 がサポートする命令セットから、DSP 乗算命令と、FPU 乗算命令のテストを行います。（他命令の検査が必要な場合には、そのテストを別途用意する必要があります。）CPU コアの内部において、レジスタから DSP、FPU へのパスが全ての命令で同一と判断できないため、命令ごとにレジスタを変えてテストします。その際、テストの入力値にウォーキングビットパターンを使用します。テスト対象の DSP 乗算命令と FPU 乗算命令に含まれるアセンブラ命令の一覧を以下に示します。

表 10. テスト対象のアセンブラ命令（DSP 乗算命令）

No.	命令	構文	内容
1	SMMLA	SMMLA <Rd>,<Rn>,<Rm>,<Ra>	最上位 32 ビット累算付き 32 ビット乗算
2	SMMLS	SMMLS <Rd>,<Rn>,<Rm>,<Ra>	最上位 32 ビット減算付き 32 ビット乗算
3	SMMUL	SMMUL <Rd>,<Rn>,<Rm>	最上位 32 ビットを返す 32 ビット乗算
4	SMMLAR	SMMLAR <Rd>,<Rn>,<Rm>,<Ra>	丸め付き最上位 32 ビット累算付き 32 ビット乗算
5	SMMLSR	SMMLSR <Rd>,<Rn>,<Rm>,<Ra>	丸め付き最上位 32 ビット減算付き 32 ビット乗算
6	SMMULR	SMMULR <Rd>,<Rn>,<Rm>	丸め付き最上位 32 ビットを返す 32 ビット乗算

表 11. テスト対象のアセンブラ命令（FPU 乗算命令）

No.	命令	構文	内容
1	VMUL.F32	VMUL.F32 <Sd>, <Sn>, <Sm>	単精度乗算
2	VMLA.F32	VMLA.F32 <Sd>, <Sn>, <Sm>	乗算後に単精度累算
3	VMLS.F32	VMLS.F32 <Sd>, <Sn>, <Sm>	乗算後に単精度減算
4	VNMLA.F32	VNMLA.F32 <Sd>, <Sn>, <Sm>	乗算後に単精度累算してから補数
5	VNMLS.F32	VNMLS.F32 <Sd>, <Sn>, <Sm>	乗算後に単精度減算してから補数

5.3.2 設定可能な内容

CPU 診断の API を使用するにあたり、ユーザが設定可能な内容を下記にまとめます。

1. 診断する命令
2. 故障検知の閾値

5.4 BUS 診断

5.4.1 診断の概要

BUS に対するテスト技法は”Inspection using test pattern”を使用します。詳細は IEC61508-7 A7.4 を参照してください。

BUS 診断で診断する機能を以下に挙げます。

［診断する BUS 機能］

- I-code bus
- D-code bus
- System bus
- DMAC bus
- DSTC bus (FM4 のみ)
- ビットバンド制御
- アンアラインドアクセス
- アービトレーション

BUS の定期診断は、診断する BUS 機能を、FM3 ファミリの場合は 3 つに分割、FM4 ファミリの場合は 4 つに分割して診断します。

5.4.2 設定可能な内容

BUS 診断の API を使用するにあたり、ユーザが設定可能な内容を下記にまとめます。

1. 故障検知の閾値

6 STL 性能

以下に、IAR Embedded Workbench for ARM6.60 kickstart 版を用いた STL の性能を示します。

測定対象のマイコンは FM3（MB9BF506R）です。

6.1 ROM/RAM サイズ

表 12 にコンポーネントごとの ROM/RAM サイズを記載します。

表 12. ROM/RAM サイズ

単位は byte

	ROM サイズ(合計)	RAM サイズ(合計)
RAM 診断	584	24
ROM 診断※	298	28
BUS 診断	784	248
CPU 診断	7742	148
合計	9408	448

※ROM 診断に記載されている ROM サイズは、CRC コードの算出にハードウェアマクロを使用した場合の数値です。ソフトウェアを使用した場合、1306(byte)になります。

6.2 診断処理速度

コンポーネントごとの診断処理速度を記載します。数値は正常系で測定したものです。

6.2.1 RAM の診断処理速度

RAM のスタートアップ診断と定期診断の処理速度を表 13, 表 14 にそれぞれ記載します。

表 13. RAM のスタートアップ診断処理速度

領域数	診断領域(byte)	スタートアップ診断(ms)
1	100	12
1	500	278
1	1000	1100
8	100	14
8	500	290
8	1000	1130

表 14. RAM の定期診断処理速度

診断領域(byte)	定期診断(µs)
4(固定)	19.6

6.2.2 ROM の診断処理速度

ROM のスタートアップ診断と定期診断の処理速度を表 15, 表 16 にそれぞれ記載します。

表 15. ROM のスタートアップ診断処理速度

CRC コード算出方法	領域数	診断領域(byte)	スタートアップ診断(µs)
HW	1	500	110
HW	1	1000	211
HW	4	500	110
HW	4	1000	211
SW	1	500	380
SW	1	1000	752
SW	4	500	380
SW	4	1000	752

表 16. ROM の定期診断処理速度

CRC コード算出方法	テスト対象領域(byte)	定期診断(µs)
HW	10	9.6
HW	50	13.8
HW	100	19
SW	10	4.5
SW	50	19.4
SW	100	37.6

6.2.3 CPU の診断処理速度

CPU のスタートアップ診断にかかる処理速度は、246 µs です。

定期診断にかかる処理速度は表 17 に記載します。測定は命令郡(例えば、LDR,LDRB,LDRH は LDR 郡)ごとに実施しています。

表 17. CPU の定期診断処理速度

命令群	定期診断(µs)
ADD	10.2
ASR	8.2
論理演算	17.4
分岐	18
比較	10.4
IT	20.2
LDR	8.4
LDM	13.8
STR	7.8
STM	10.2
SHIFT	5
MLA	4.4
MOV	14
REV	5.6
SUB	6.6
SXT	7.2
レジスタ	7.2

6.2.4 BUS の診断処理速度

BUS のスタートアップ診断にかかる処理速度は、88.8 µs です。

定期診断にかかる処理速度は表 18 に記載します。測定は、分割した 3 つのテスト(TestNo1～3)ごとに実施しています。

表 18. BUS の定期診断処理速度

TestNo	定期診断(µs)
TestNo1	9.68
TestNo2	76
TestNo3	2.8

7 参考文書

[1]. IEC61508 1-7(ed2.0)

[2]. ARMv7-M Architecture Application Level Reference Manual, 2008

[3]. MB9BF504NB-DS706-00021-2v0-J.pdf (MB9B500 Data Sheet)

[4]. MB9B560R-DS709-00001.pdf(MB9B560R Data Sheet)

[5]. MB9Bxxx-MN706-0002-4v0-E (FM3 Series Periphery Manual)

[6]. MN709-00001-1v0-J.pdf(FM4 Series Periphery Manual)

[7]. Cortex-M3 Revision r2p1 Technical Reference Manual

[8]. Cortex-M4 Revision r0p1 Technical Reference Manual

8 その他の情報

より詳細な情報をご要望の場合は、下記 URL よりご連絡ください。

www.cypress.com/contact-us

E-mail をご利用の場合は、下記 E-mail アドレス宛にお送りください。

customercare@cypress.com

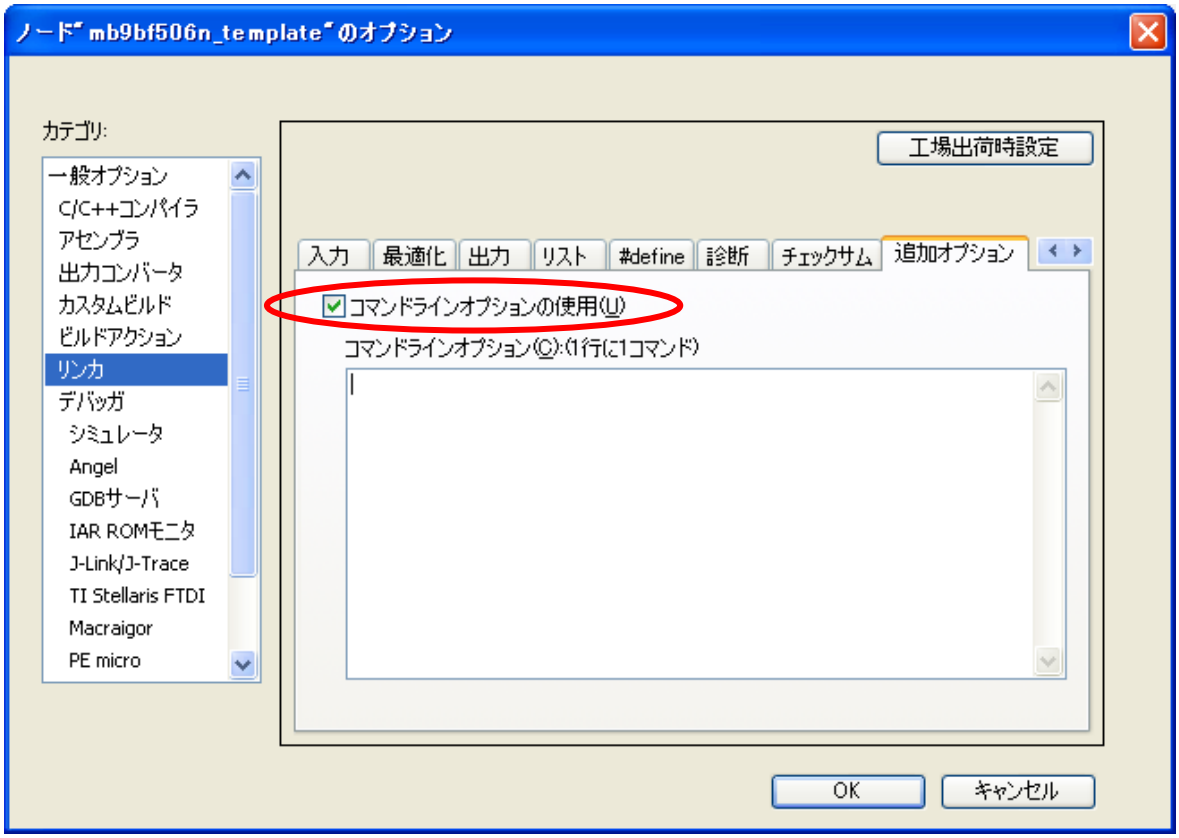
A 付録

A.1 フラッシュの CRC コード作成方法

5.1 ROM 診断 で使用するデータ/プログラムの CRC コードの生成方法について、IAR Embedded Workbench の例を以下に説明します。詳細は IAR のマニュアルを参照してください。

A.1.1 コマンドラインの起動

「プロジェクト」→「オプション」→「リンカ」→「追加オプション」タブを選択し、"コマンドラインオプションの使用"にチェックを入れます。



A.1.2 コマンドの入力

--place_holder コマンド

CRC コードを作成して ROM にセクションを生成するために、"--place_holder"オプションを使用します。セクションのサイズを 4byte、アライメントを 1 に設定するため下記のコマンドを設定します。

--place_holder __checksum,4,.checksum,1

--fill コマンド

CRC コードを作成するためには、対象領域の未使用領域に対して任意の値をフィルする必要があります。そのために、"--fill"コマンドを使用します。対象領域 0x00000000-0x00003FFF にフィル値 0xFF を設定するコマンドは以下の設定です。

--fill 0xFF;0x0000-0x3FFF

対象領域 0x00000000-0x00003FFF、0x5000-0x5FFF、0x6500-0x6FFF にフィル値 0xFF を設定するコマンドは以下の設定です。

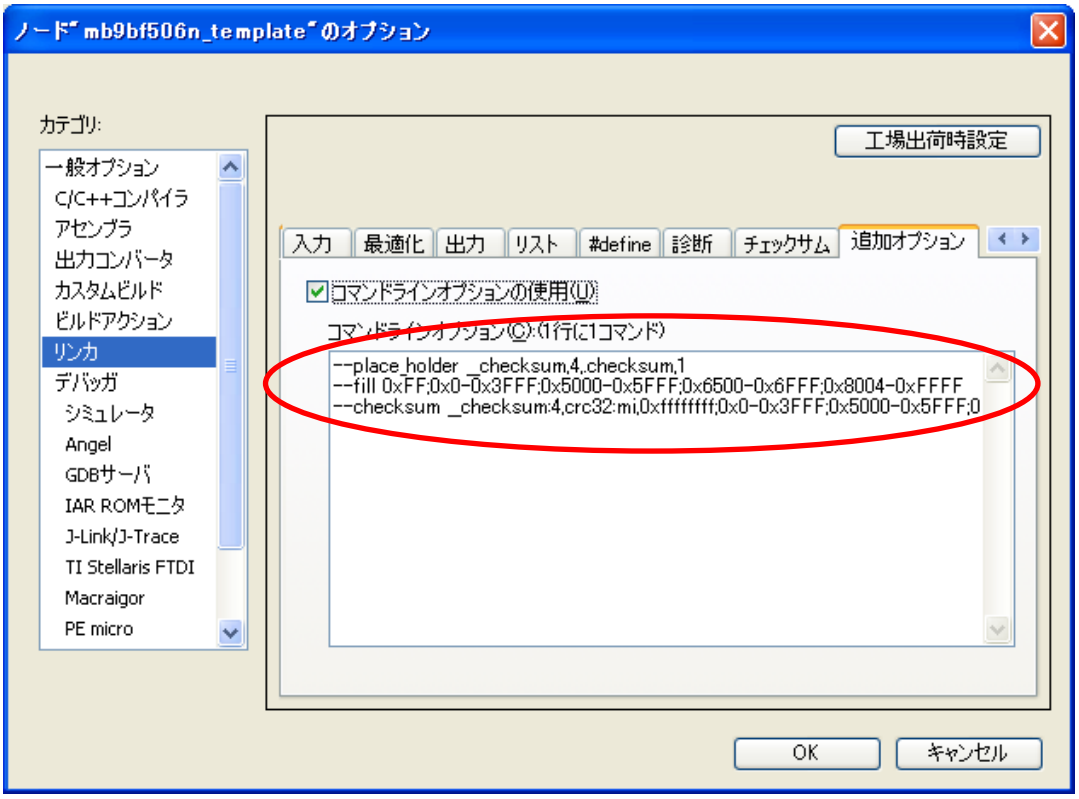
--fill 0xFF;0x0-0x3FFF;0x5000-0x5FFF;0x6500-0x6FFF

--checksum コマンド

使用する CRC のアルゴリズムを設定します。CRC コードを格納するシンボル名を__checksum、サイズを 4byte、アルゴリズムを CRC32、計算を LSB first、CRC コードを 0xFFFFFFFF で初期化、対象領域 0x00000000-0x00003FFF、0x5000-0x5FFF、0x6500-0x6FFF の設定をするコマンドは以下のとおりです。

--checksum __checksum:4,crc32:mi,0xffffffff;0x0-0x3FFF;0x5000-0x5FFF;0x6500-0x6FFF

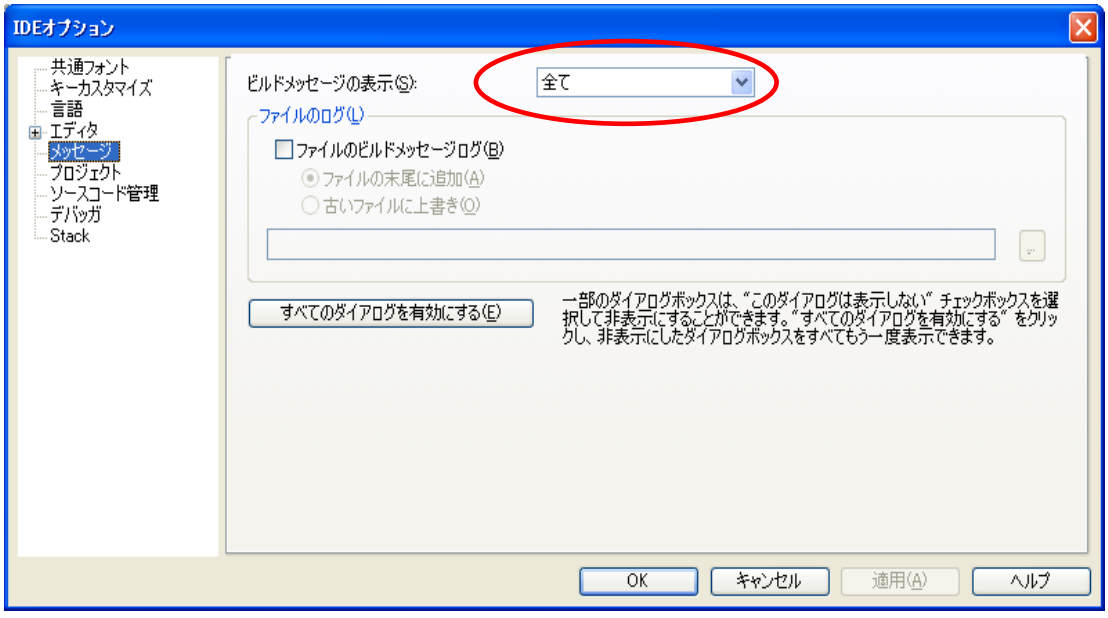
0.,0.,0.を入力した後、"OK"を押してウィンドウを閉じます。



A.1.3 メッセージウィンドウ表示内容の設定

メイク時のメッセージをメッセージウィンドウへ表示するように設定します。

「ツール」→「オプション」→「メッセージ」を選択します。"ビルドメッセージの表示(S):"　のコンボボックスで"全て" を選択します。右下の"OK"または"適用"ボタンを押します。



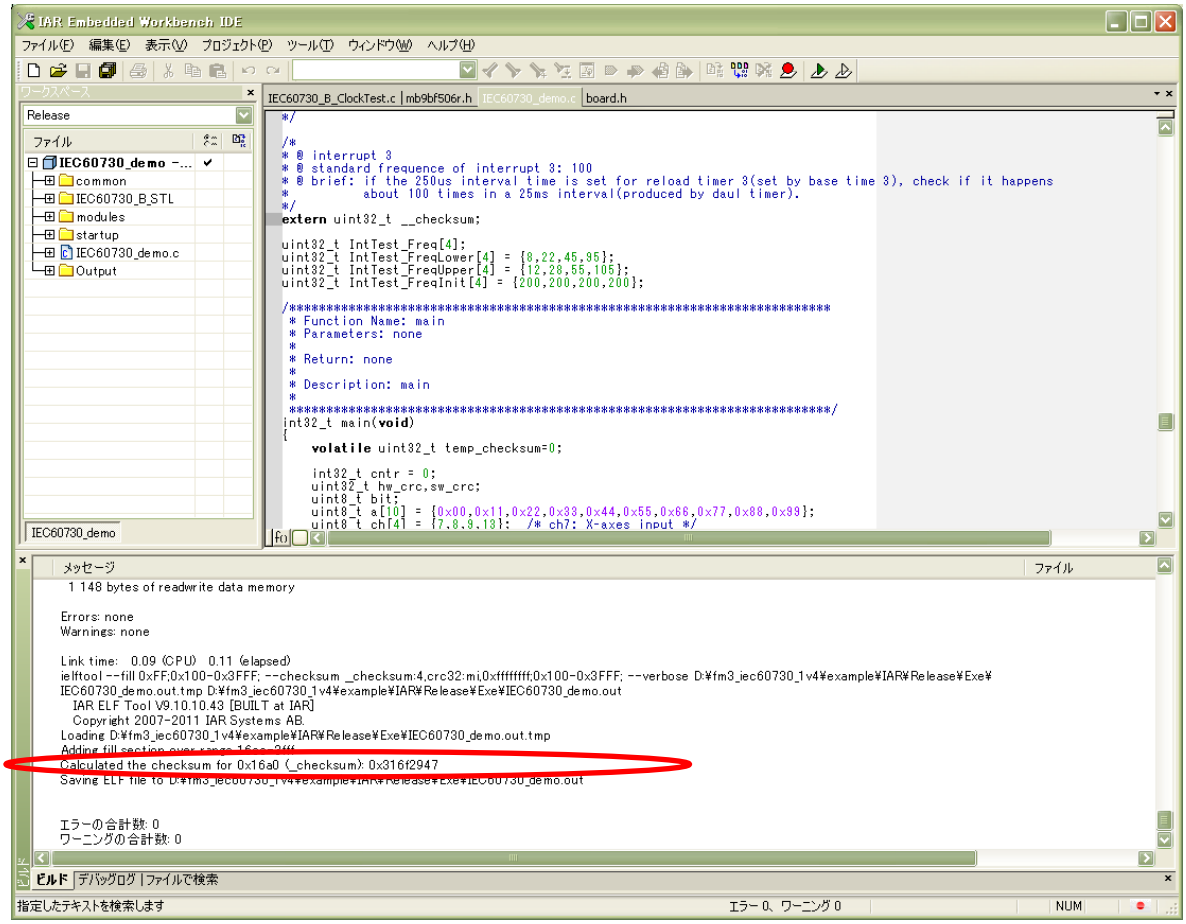
A.1.4 リンカ設定ファイルの設定

生成した CRC コードをフラッシュメモリに格納するため、リンカ設定ファイルに設定を追加します。debug モードの場合は” mb9bf506_ram.icf”ファイル、release モードの場合は” mb9bf506.icf”ファイルです。0x8000 番地に CRC コードを格納する場合は以下の設定を追加します。

```
define symbol __ICFEDIT_checksum_start__ = 0x00008000;
define block CHECKSUM  {ro section .checksum};
define symbol __ICFEDIT_checksum_start__ = 0x00008000;
```

A.1.5 CRC コード作成

メイク(ビルド) を実行し、CRC コードが作成されたことを確認します。



9 改訂履歴

ドキュメント名: AN204377 - FM3/FM4 ファミリ IEC61508 SIL2 セルフテストライブラリ

ドキュメント番号 : 002-04378

Revision	ECN	変更者	発行日	変更内容
**	-	NNAK	04/24/2014	1 版: 初版
				2 版の変更内容 1 はじめに - FM4 に対応 5.3.1 診断の概要 - CPU 診断に FM4 の DSP 命令と FPU 命令を追加 5.4 BUS 診断 – BUS 診断に DSTCbus テストを追加
*A	5613356	NNAK	02/01/2017	Cypress テンプレート形式に更新されました。 これは英語版 002-04377 Rev.*A を翻訳した日本語版です。
*B	5888462	NNAK	09/19/2017	Cypress の新ロゴを適用。

セールス, ソリューションおよび法律情報

ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店、および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、サイプレスのロケーション ページをご覧ください。

製品

ARM® Cortex® Microcontrollers	cypress.com/arm
車載用	cypress.com/automotive
クロック&バッファ	cypress.com/clocks
インターフェース	cypress.com/interface
IoT (モノのインターネット)	cypress.com/iot
メモリ	cypress.com/memory
マイクロコントローラ	cypress.com/mcu
PSoC	cypress.com/psoc
電源用 IC	cypress.com/pmic
タッチ センシング	cypress.com/touch
USB コントローラー	cypress.com/usb
ワイヤレス／RF	cypress.com/wireless

PSoC® ソリューション

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

サイプレス開発者コミュニティ

フォーラム | WICED IOT Forums | Projects | ビデオ | ブログ | トレーニング | Components

テクニカルサポート

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2012-2017. 本書面は、Cypress Semiconductor Corporation 及び Spansion LLC を含むその子会社（以下、「Cypress」という。）に帰属する財産である。本書面（本書面に含まれ又は言及されているあらゆるソフトウェア又はファームウェア（以下、「本ソフトウェア」という。）を含む）は、アメリカ合衆国及び世界のその他の国における知的財産法令及び条約に基づき、Cypress が所有する。Cypress はこれらの法令及び条約に基づく全ての権利を留保し、また、本段落で特に記載されているものを除き、Cypress の特許権、著作権、商標権又はその他の知的財産権のライセンスを一切許諾していない。本ソフトウェアにライセンス契約書が伴っておらず、かつ、あなたが Cypress との間で別途本ソフトウェアの使用方法を定める書面による合意をしていない場合、Cypress は、あなたに対して、（1）本ソフトウェアの著作権に基づき、（a）ソースコード形式で提供されている本ソフトウェアについて、Cypress ハードウェア製品と共に用いるためにのみ、組織内部でのみ、本ソフトウェアの修正及び複製を行うこと、並びに（b）Cypress のハードウェア製品ユニットに用いるためにのみ、（直接又は再販売者及び販売代理店を介して間接のいずれかで）エンドユーザーに対して、バイナリーコード形式で本ソフトウェアを外部に配布すること、並びに（2）本ソフトウェア（Cypress により提供され、修正がなされていないもの）に抵触する Cypress の特許権のクレームに基づき、Cypress ハードウェア製品と共に用いるためにのみ、本ソフトウェアの作成、利用、配布及び輸入を行うことについての非独占的で譲渡不能な一身専属的ライセンス（サブライセンスの権利を除く）を付与する。本ソフトウェアのその他の使用、複製、修正、変換又はコンパイルを禁止する。

適用される法律により許される範囲内で、Cypress は、本書面又はいかなる本ソフトウェアに関しても、明示又は黙示をとわず、いかなる保証（商品性及び特定の目的への適合性の黙示の保証を含むがこれらに限られない）も行わない。適用される法律により許される範囲内で、Cypress は、別途通知することなく、本書面を変更する権利を留保する。Cypress は、本書面に記載のあるいかなる製品又は回路の適用又は使用から生じる一切の責任を負わない。本書面で提供されたあらゆる情報（あらゆるサンプルデザイン情報又はプログラムコードを含む）は、参照目的のためのみに提供されたものである。この情報で構成するあらゆるアプリケーション及びその結果としてのあらゆる製品の機能性及び安全性を適切に設計し、プログラムし、かつテストすることは、本書面のユーザーの責任において行われるものとする。Cypress 製品は、兵器、兵器システム、原子力施設、生命維持装置若しくは生命維持システム、蘇生用の設備及び外科的移植を含むその他の医療機器若しくは医療システム、汚染管理若しくは有害物質管理の運用のために設計され若しくは意図されたシステムの重要な構成部分として用いるため、又はシステムの不具合が人身傷害、死亡若しくは物的損害を生じさせることになるその他の使用（以下、「本目的外使用」という。）のためには、設計、意図又は承認されていない。重要な構成部分とは、装置又はシステムのその構成部分の不具合が、その装置若しくはシステムの不具合を生じさせるか又はその安全性若しくは実効性に影響すると合理的に予想できる、機器又はシステムのあらゆる構成部分をいう。Cypress 製品のあらゆる本目的外使用から生じ、若しくは本目的外使用に関連するいかなる請求、損害又はその他の責任についても、Cypress はその全部又は一部をとわず一切の責任を負わず、かつ、あなたは Cypress をそれら一切から免除するものとし、本書により免除する。あなたは、Cypress 製品の本目的外使用から生じ又は本目的外使用に関連するあらゆる請求、費用、損害及びその他の責任（人身傷害又は死亡に基づく請求を含む）から Cypress を免責補償する。

Cypress、Cypress のロゴ、Spansion、Spansion のロゴ及びこれらの組み合わせ、WICED、PSoC、CapSense、EZ-USB、F-RAM、及び Traveo は、米国及びその他の国における Cypress の商標又は登録商標である。Cypress の商標のより完全なリストは、cypress.com を参照のこと。その他の名称及びブランドは、それぞれの権利者の財産として権利主張がなされている可能性がある。