応用記述

AVR1610 : XMEGAでのIEC60730等級B適合への指針

Atmel 8ビット マイクロ コントローラ

要点

- 等級B部品検査必要条件の一覧
- 一般的な等級B検査用ファームウェア ライブラリ

説明

IEC60730は製品設計と操作の両方の多くの局面を扱う家庭応用のための安全規格です。この規格は保安装置に対する他の規格、例えばIEC60335によっても参照されます。この規格を持つ系全体の適合は操作を安全にするよう認証されるべき電気製品に対して必要です。

この応用記述は電子制御に関する規格の付属書Hでの適合への手引きです。更に、IAR Embedded Workbench®とAtmel® AVR® GCCに対する認証されたファームウェア ライブラリと使用例を供給します。ファームウェア ライブラリは規格の殆どの一般的な部分を網羅するように設計され、一方で多くは応用と実装の両方の選択に依存します。

この資料の概要は次の通りです。第1章はIEC60730規格からのいくつかの定義を紹介します。第2章は等級Bソフトウェアに関する必要条件を記述します。第3章は組み込み自己検査の概要を示します。第4章は認証されたライブラリとインクルードされるファイルを紹介します。第5～10章はライブラリ内の組み込み自己検査の詳細を記述します。最後に、第11章はAtmel AVR XMEGA®で含まれる他の安全機能を紹介します。

目次

1. IEC60730の付属書Hでの定義

1.1. ソフトウェア等級

IEC60730安全規格の付属書Hは電気製品に対する制御ソフトウェアの3つ等級を定義します。

- 等級A – 装置の安全に対して信頼に足ることが意図されない制御機能(H.2.21.1)
- 等級B – 電気製品でソフトウェア誤り以外の誤りが起きた場合の危険を防ぐことが意図されるコードを含むソフトウェア(H.2.21.2)
- 等級C – 他の保護する装置の使用なしで危険を防ぐことが意図されるコードを含むソフトウェア(H.2.21.3)

保護制御機能に使用されるソフトウェアは等級Bまたは等級Cのどちらかです。この応用記述は殆どの電気製品に適用する等級Bを扱います。

1.2. 管理構造

等級B管理は定義された3つの1つに従って設計することができます(H.11.12.2)

- 機能検査を持つ単一チャネル – 検査データがそれの動作に先立って機能部に導入される単一チャネル構造(H.2.16.5)
- 周期的自己検査を持つ単一チャネル – 動作中に制御の成分が周期的に検査される単一チャネル構造(H.2.16.6)
- 比較なしの2重チャネル – 指定した動作を実行することを意味する互いに独立した2つの機能を含む構造(H.2.16.1)

用語の「チャネル」は電気製品内のMCU数に当て嵌まります。単一チャネル構造はそれが最低費用になるため、好まれる構造の傾向にあります。

機能検査を持つ単一チャネル構造はシステムが製造の時点でだけ検査されることを意味します。この構造は応用が周期的な検査を必要とするどの部品も使用しない場合にだけ使用することができます。けれども、これが製品の動作中に検出されるべき欠陥を許すため、周期的検査はより安全な任意選択です。代表的に、検査間の間隔時間はそれが危険を引き起こす関連した部品での欠陥を負うよりも短くなければなりません。

比較なし2重チャネルは2つのMCUが異なる作業で、どちらのMCUも他方が正しく動作していることを調べることができる特別な構造です。1つの手法は2つの間の制御作業を共用するよりもむしろ1つのMCUを厳密に指揮監視に使用することです。この場合では、監視側として度々安価なデバイスを使用することができます。

この応用記述は単一チャネル構造に集中します。

2. 等級B必要条件

電気製品に関して等級Bに従うため、管理ソフトウェアは表2-1.のシステム部品に対して指定された欠陥を検出して処理しなければなりません。それらの検査と欠陥検出に加えて、ソフトウェアは認証を通るために正しく資料化されなければなりません。これにはプログラムの流れ、制御とデータの流れ、タイミング、誤り樹形、全般設計原理を含みます。

表2-1. 検査に対する部品と欠陥/異常の概要 (H.11.12.7)

検査する部品	検査に対する欠陥/異常
1. CPU	–
1.1. レジスタ	動かない
1.3. プログラム カウンタ	動かない
2. 割り込み処理と実行	ない/過ぎる頻度の割り込み
3. クロック	不正な周波数
4. メモリ (注1)	–
4.1. 不変メモリ	全ての単一ビット欠陥
4.2. 可変メモリ	DC欠陥
4.3. アドレス指定	動かない
5. 内部データ経路 (注1)	–
5.1. データ	動かない
5.2. アドレス指定	不正なアドレス
6. 外部通信	–
6.1. データ	ハミング距離3
6.3. タイミング	時間内の不正な点
7. I/O周辺	–
7.1. デジタルI/O	H.27.1で指定する欠陥条件 (注2)
7.2.1. A/D変換器とD/A変換器	H.27.1で指定する欠陥条件 (注2)
7.2.2. アナログ多重器	不正なアドレス指定
9. 独自チップ	静的と動的な機能仕様外の如何なる出力

注1: AVRに関してフラッシュ、SRAM、EEPROMが内部のため、これらは本質的に同じです。

注2: 本表は様々外部部品を一覧し短絡や開放の欠陥が検出されなければならないかを示します。

これらの検査の多くは必然的に応用依存です。例として、I/O周辺について、入出力信号のもっともらしい検査を行うことが必要とされます。これは応用とそれの実装の両方に同様に依存します。この応用記述で供給されるﾌｧｰﾑｳｪｱ ﾗｲﾌﾞﾗﾘは、従って表2-1.での必要条件の全てを網羅することができません。

3. 部品検査

表2-1.での個別部品に対する受け入れ可能な欠陥検出測定と考慮が下で説明されます。

注: 機能検査を持つ単一ﾁｬﾈﾙ構造は、電気製品で使用される部品に対して受け入れ可能な測定下で一覧にされていない機能検査の場合に使用することができません。

3.1. CPUﾚｼﾞｽﾀ – 部品 1.1

検査目的：ﾚｼﾞｽﾀ内の動かないﾋﾞｯﾄの検出

CPUﾚｼﾞｽﾀはMCUの最も重要な部分で、欠陥ﾚｼﾞｽﾀで正しい動作が不可能なため、検査されなければなりません。

欠陥検出に対して受け入れ可能な測定は静的ﾒﾓﾘ検査またはﾊﾟﾘﾃｨ検査付き語保護を使用する機能検査や周期的自己検査です。このﾗｲﾌﾞﾗﾘでは、ﾊﾟﾘﾃｨ検査がAtmel AVR XMEGA系統で任意選択ではないﾊｰﾄﾞｳｪｱ実装を必要とするため、最初の方法を選びました。けれども、ﾃﾞｰﾀ冗長性に対して他の方法をｿﾌﾄｳｪｱで、例えば同じ値を保持するように2つのﾚｼﾞｽﾀを使用して実装することができ、この場合ではAtmel AVR CPUの特徴の32個の汎用ﾚｼﾞｽﾀが便利です。

3.2. ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ – 部品 1.3

検査目的：ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ内の動かないﾋﾞｯﾄの検出

正しく機能するﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀはMCUで成功裏に走行するためにどのｿﾌﾄｳｪｱでも重要です。けれども、ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀはそのようなどの検査も最初の場所で正しく機能するためにﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀを信頼するため、動かないﾋﾞｯﾄに対して直接的に検査することができません。

欠陥検出に対して受け入れ可能な測定は機能検査、周期的自己検査、独立時間枠監視、またはﾌﾟﾛｸﾞﾗﾑの流れの論理的な監視です。

好まれる解決策は応用の間接時間枠監視にｳｫｯﾁﾄﾞｯｸﾞを使用することです。ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀが誤りを起こす場合、ｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏは不正な時間でﾘｾｯﾄ、または全てでﾘｾｯﾄがなく、結局、ﾃﾞﾊﾞｲｽ ﾘｾｯﾄを引き起こします。それが統合安全機能のため、XMEGA系統のｳｫｯﾁﾄﾞｯｸﾞは、ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ欠陥を捕らえるために頼るのに先立って、通常と窓の両動作で正しい動作を検査されなければなりません。

それが検査されてしまった後、ｳｫｯﾁﾄﾞｯｸﾞは全時間に於いて窓動作で許可されるべきです。更に、閉区間は少なくても開区間と同じ長さ、即ち総周期の最低50%であるべきです。

XMEGAのｳｫｯﾁﾄﾞｯｸﾞ機能のより多くの情報についてはAtmel AVR1310応用記述を参照してください。

3.3. 割り込み – 部品 2

検査目的：割り込みが起きて意図した頻度で処理されることを確認

殆どの応用はそれらの動作に対して割り込みを信頼し、従ってそれらが意図された時に起こり、それによって処理されることを確認することが重要です。

欠陥検出に対して受け入れ可能な測定は機能検査や割り込みの時間枠監視を含みます。これは電気製品が一旦使用されると、欠陥動作検出を手助けするため、時間枠監視が推奨されます。どの割り込み検査もXMEGA割り込み制御器を間接的にも検査します。

3.4. ｸﾛｯｸ – 部品 3

検査目的：ｼｽﾃﾑ ｸﾛｯｸ周波数での予期せぬ変動の検出

正しい動作と正確なﾀｲﾐﾝｸﾞを持つためにMCUに対してｼｽﾃﾑ ｸﾛｯｸの周波数は仕様内であることを確認されなければなりません。

欠陥検出に対して受け入れ可能な測定は周波数または時間枠の監視です。ｸﾘｽﾀﾙ発振器に基づくｸﾛｯｸについて、必要条件はそれが(低)高調波で発振していないことを検出することです。

ｼｽﾃﾑ ｸﾛｯｸ周波数の検証のために参照基準ｸﾛｯｸが必要とされます。Atmel AVR XMEGA系統の事象ｼｽﾃﾑは外部ｸﾛｯｸ信号またはRTC(実時間計数器)によって起動され、周波数比較を許すﾀｲﾏ/ｶｳﾝﾀ捕獲を許します。ﾀｲﾏ/ｶｳﾝﾀの上昇または下降の計数をするのに事象ｼｽﾃﾑを使用することも可能です。

追加の安全機能として、XMEGA系統は外部発振器に対するｸﾛｯｸ欠陥(停止)検出が特徴です。

3.5. SRAM – 部品 4.2, 4.3, 5

検査目的：ﾋﾞｯﾄが動かないのと、SRAM内とﾃﾞｰﾀ ﾊﾞｽでの連結欠陥、それとどのｱﾄﾞﾚｽ指定の問題をも検出

内部SRAMはﾃﾞｰﾀの揮発性記憶として使用され、これに関連するどんな欠陥も電気製品管理に関して破滅的になり得ます。

欠陥検出に対して受け入れ可能な測定は周期的検査やﾃﾞｰﾀ冗長性、例えばﾊﾟﾘﾃｨ ﾋﾞｯﾄを含みます。後者はこの目的に対して特別なﾊｰﾄﾞｳｪｱ実装なしで厄介になり得、例えば最良の選択としてﾏｰﾁ算法での周期的検査に任せます。

SRAM全体が1回で検査できない場合、検査されたﾒﾓﾘ領域は連結欠陥の検出を許すためにいくつかの重複を持たなければなりません。

3.6. フラッシュとEEPROM – 部品 4.1

検査目的：不揮発性メモリ内の全ての単一ビット欠陥を検出

全てのAtmel AVRマイクロ コントローラでは、応用ソフトウェアがフラッシュ メモリに格納され、一方EEPROMは、例えばデバイス特定設定と定数に使用することができます。安全に動作するデバイスについて、それらの不揮発性メモリは化けに対して調べられなければなりません。

欠陥検出に対して受け入れ可能な測定は単一または複数のチェックサムを用いる周期的自己検査や単一ビット データ冗長性、例えばハードウェアでのパリティ検査です。これが一度に検査されるフラッシュまたはEEPROMの1領域を許すため、複数チェックサムが推奨される任意選択です。この方法は目標のメモリ範囲に対してチェックサムを計算し、その後に不揮発性メモリの何処かに格納された参照基準チェックサムをその結果と比較します。

3.6.1. 自己プログラミングでの注意

例えば書き込み中の電力不足が予測不能な結果を持つため、過酷な環境でフラッシュの自己プログラミングを実行することは推奨されません。

自己プログラミングが絶対的に必要なら、偶然の自己書き込みが不可能なように、施錠ビットによって保護されるブートローダでこれを行うことが推奨されます。この場合、ブートローダはそれ内のどのコードが実行されるのにも先立ってフラッシュの応用領域を調べるべきです。

全てのAtmel AVR XMEGAはフラッシュ メモリ内のブートローダが特徴です。

3.7. 外部通信 – 部品 6

検査目的：データ転送、通信の流れとタイミングの正しいことを確認

外部装置との通信は多くの応用の重要部分です。通信が雑音に影響され易く、通信線のどちらかの終りが不正に動くかもしれないため、これは欠陥の潜在的な供給元を表します。その結果として、或る装置での雑音や不完全な動作が他方での不完全な動作を引き起こさないことを保証するような手段が取られなければなりません。

データでの欠陥検出に対して受け入れ可能な測定は反復、CRCまたはハミング符号、または規約検査のような複数ビット データ冗長性です。タイミングでの欠陥検出に対して受け入れ可能な測定は時間枠監視または計画された送信です。通信の流れでの欠陥検出に対して受け入れ可能な測定はタイミングに対するそれらと同じですが、論理的な監視も含みます。

要するに、時間超過、計画化、転送冗長性を持つ通信ドライバに基づく状態機構が使用されるべきです。

3.8. 入出力周辺 – 部品 7

検査目的：入出力が意図されるようで、信号が正しく配線されることを確認

アナログやデジタルの入出力(I/O)は全ての応用で必要とされ、周辺またはMCUそれ自身のどちらかで不完全な動作の検出に使用することができます。

欠陥検出に対して受け入れ可能な測定はソフトウェアが何時でも意図された入力を得て望む出力を与えることの確認を意味する、もっともらしい検査です。

4. Atmel AVR XMEGA用等級Bライブラリ

このライブラリの目的はXMEGA系統マイクロ コントローラに基づく安全且つ信頼に足る応用の設計を簡単化することです。更に、このライブラリはお客様に対して設計簡素化と検定処理が保証されています。

このライブラリは多くの異なる応用で自己検査部を組み込むのに充分柔軟なように開発されています。そして最終使用者は応用がIEC 60730等級Bに従うようにそれを形態設定して使用する責任があります。

Atmel Studio 6に含まれるAVR GCCコンパイラとIAR™の支援も追加されます。使用者応用で自己診断ルーチンがどう組み込まれるかを示すために多数の例が含まれています。

ライブラリのソース コードはDoxygen(www.doxygen.org)での自動資料生成用に準備されています。この資料化は本資料で提供される情報を提供します。

4.1. 異常処理

検査部が可能な限り標準的であるように、使用者によって形態設定され得る多数の異常処理部を定義しました。異常は重要か重要でないかに分けられ、異常処理部に対して既定値を持ちます。

重要な異常は処理することができない、例えばレジスタ自己診断ルーチンの結果を返すべきレジスタが動かないビットを持つ時のそれらです。重要な異常は既定によってCPUを停止、即ちCPUは無限繰り返し実行のままにします。原則としてこれはウォッチドッグ リセットを引き起こし、(以降WDTとして参照される)ウォッチドッグ タイマによってシステム リセットが発行された後で取られるべき活動は適切なように形態設定することができます。

重要でない異常は例えそれらが正しい動作からMCUを妨げても、それらは未だプログラムによって処理することができます。例えば(検査の結果を返すものやスタック レジスタ以外の)1つのレジスタが動かない場合、プログラムはシステムを安全な状態に置くためにいくつかの活動を取ることができます。重要でない異常は既定によって異常全域フラグを設定します。このフラグはclassb_errorと呼ばれ、システムを安全な状態に置くために主応用によって使用することができます。これは例に従った手法でした。

4.2. ソース ファイル

共通ﾌｧｲﾙ:

- avr_compiler.h：このﾌｧｲﾙはIARとGCCとのｺｰﾄﾞ共通性を保証するためのいくつかの全般的な定義とﾏｸﾛを含みます。
- classb_rtc_common.c/h：これは実時間計数器用の任意選択ﾄﾞﾗｲﾊﾞです。
- error_habdler.h：異常処理部と形態設定可能な活動に関連するﾏｸﾛと定義

CPUﾚｼﾞｽﾀ:

- classb_cpu.h：検査用の設定、定義、ﾏｸﾛ
- classb_cpu_gcc/iar.c：自己診断ﾙｰﾁﾝの実装
- UserApplication.c：検査が組み込まれた応用例

CPUﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ (ｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏ検査):

- classb_wdt.c/h：検査用のｺｰﾄﾞとﾍｯﾀﾞのﾌｧｲﾙ
- UserApplication.c：検査が組み込まれた応用例

割り込み管理部:

- classb_interrupt_monitor.c/h：検査用のｺｰﾄﾞとﾍｯﾀﾞのﾌｧｲﾙ
- UserApplication.c：検査が組み込まれた応用例

ｼｽﾃﾑ ｸﾛｯｸ:

- classb_freq.c/h：検査用のｺｰﾄﾞとﾍｯﾀﾞのﾌｧｲﾙ
- UserApplication.c：検査が組み込まれた応用例

可変ﾒﾓﾘ (SRAMに対するﾏｰﾁ検査):

- classb_sram.c/h：検査用のｺｰﾄﾞとﾍｯﾀﾞのﾌｧｲﾙ
- UserApplication.c：検査が組み込まれた応用例

不変ﾒﾓﾘ (ﾌﾗｯｼｭ ﾒﾓﾘとEEPROMに対するCRC検査):

- classb_crc.h：全般的な形態設定を持つﾍｯﾀﾞ ﾌｧｲﾙ
- classb_crc_hw.c/h：ﾊｰﾄﾞｳｪｱ実装用のｺｰﾄﾞとﾍｯﾀﾞのﾌｧｲﾙ
- classb_crc_sw.c/h：ｿﾌﾄｳｪｱ実装用のｺｰﾄﾞとﾍｯﾀﾞのﾌｧｲﾙ
- UserApplication.c：検査が組み込まれた応用例

ｱﾅﾛｸﾞ入出力:

- classb_analog.c/h：検査用のｺｰﾄﾞとﾍｯﾀﾞのﾌｧｲﾙ
- UserApplication.c：検査が組み込まれた応用例

5. レジスタ

自己診断ﾙｰﾁﾝのため、CPUﾚｼﾞｽﾀは動かないﾋﾞｯﾄといくつかの連結欠陥に対して検査されます。基本的な算法は次のように動きます。最初の段階でﾚｼﾞｽﾀは既知の値で設定されます。第2段階でﾚｼﾞｽﾀのﾋﾞｯﾄは状態変更を強制されます。第3段階でﾚｼﾞｽﾀの内容は検証されます。その後、ﾚｼﾞｽﾀのﾋﾞｯﾄは再び値変更を強制され、最後にﾚｼﾞｽﾀの内容が再び検証されます。両方の場合で、ﾚｼﾞｽﾀの内容が不正なら、異常変数が設定されます。

ﾚｼﾞｽﾀ(R0～R31)とI/Oﾚｼﾞｽﾀの2つの形式を検査します。これらは独立したﾒﾓﾘ空間に置かれ、異なる命令でｱｸｾｽされます。加えて考慮されなければならない多数の問題があります。先ず第一に、それらを使用することができないため、いくつかのﾚｼﾞｽﾀを破壊的に操作することができません。この形式のﾚｼﾞｽﾀと特定のｺﾝﾊﾟｲﾗはどのﾚｼﾞｽﾀが防がれなければならないかを決めます。我々はそれらのﾚｼﾞｽﾀの値を格納するのに1つの補助ﾚｼﾞｽﾀを使用しました。次に、R0～R15のﾚｼﾞｽﾀとI/Oﾚｼﾞｽﾀは即値命令、例えばLDIとCPIを使用することができません。従って、別の補助ﾚｼﾞｽﾀを通して設定と比較を行うことを選びました。

ﾚｼﾞｽﾀはこの順に従って検査されます(GCCｺﾝﾊﾟｲﾗ実装)。

1. 戻り値ﾚｼﾞｽﾀ：R24
2. 補助ﾚｼﾞｽﾀ：R31,R30
3. ｽﾀｯｸ ﾎﾟｲﾝﾀ：SPL,SPH
4. ﾚｼﾞｽﾀ ﾌｧｲﾙ：R29～R25,R23～R0
5. 拡張ｱﾄﾞﾚｽ指定ﾚｼﾞｽﾀ：RAMPD/X/Y/Z,EIND
6. ｽﾃｰﾀｽ ﾚｼﾞｽﾀ：SREG (割り込みﾌﾗｸﾞを除く)

最初の3つの段階に於いて自己診断ﾙｰﾁﾝに対して重要なそれらのﾚｼﾞｽﾀを検査します。

- 戻り値ﾚｼﾞｽﾀ：この自己検査の結果を配給するのに使用されます。
- 補助ﾚｼﾞｽﾀ1：保護されなければならないそれらのﾚｼﾞｽﾀの値を格納するのに使用されます。
- 補助ﾚｼﾞｽﾀ2：LDIとCPIの命令とで使用できないﾚｼﾞｽﾀ(R0～R15)検査時に値を複写するのに使用されます。
- ｽﾀｯｸ ﾎﾟｲﾝﾀ：主ﾌﾟﾛｸﾞﾗﾑへ戻るのに必要です。

補助ﾚｼﾞｽﾀはｽﾀｯｸ ﾎﾟｲﾝﾀを検査するのに必要とされるため、重要です。これらのﾚｼﾞｽﾀで異常がある場合、ﾃﾞﾊﾞｲｽは既定によって無限繰り返しの実行に留まります。けれども、代わりに異常処理部を呼ぶことが可能です。

以降の段階で、ﾚｼﾞｽﾀ ﾌｧｲﾙの残りとI/Oﾚｼﾞｽﾀが検査されます。異常があれば、自己診断ﾙｰﾁﾝは異常処理部を呼んで検査の値を戻します。けれども、重要なﾚｼﾞｽﾀと同じ動き、即ちどれかのﾚｼﾞｽﾀで欠陥がある場合にﾃﾞﾊﾞｲｽが停止するように形態設定することが可能です。

自己診断ﾙｰﾁﾝはﾚｼﾞｽﾀ操作に対して多くの柔軟性を提供するようにｲﾝﾗｲﾝ ｱｾﾝﾌﾞﾗで書かれます。ｱｾﾝﾌﾞﾘ ｺｰﾄﾞ書きを簡単化するため多数のﾏｸﾛを定義しました。

応用例はLEDを点灯し、基本的に無限繰り返しである主ﾌﾟﾛｸﾞﾗﾑに割り込むことができるｽｲｯﾁを構成設定します。ｽｲｯﾁ割り込みﾙｰﾁﾝでは、LEDが交互点滅してﾚｼﾞｽﾀ検査が呼ばれます。原則としてCPUﾚｼﾞｽﾀは正しく動くべきで、従ってこのﾚｼﾞｽﾀ検査は異常が見つからないでしょう。ｽｲｯﾁ割り込みはその後、順当に実行を続ける主ﾌﾟﾛｸﾞﾗﾑへ戻り、全体異常(ﾌﾗｸﾞ)は設定されず、応用は無限繰り返しに留まります。主ﾌﾟﾛｸﾞﾗﾑはその後に釦押下によって何時でも再び割り込むことができます。

異常を模倣するため、自己診断ﾙｰﾁﾝ内に中断点を設定することができます。応用はその後に開始することができ、中断点で停止後に、動かないﾋﾞｯﾄを模倣するためにﾚｼﾞｽﾀの内容を変更することができます。実行はその後に再開することができ、自己診断ﾙｰﾁﾝがこの異常を検出したことを知るでしょう。変更したﾚｼﾞｽﾀとCLASSB_ERROR_CRITとCLASSB_ERROR_NON_CRITの定数に依存し、(終り無き繰り返しの抜け出しとLEDのOFF切り替えを導く)異常全体変数が1に設定されるか、またはCPUが自己診断ﾙｰﾁﾝ内側の終り無き繰り返しで動かないかのどちらかに出会うでしょう。何れの場合も、応用はそれ以上、釦の押下に応答しません。

6. ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ

ｳｫｯﾁﾄﾞｯｸﾞ ﾄﾞｯｸﾞ ﾀｲﾏ(WDT)は、逸脱や行き詰まるｺｰﾄﾞのような異常状況からの回復を許す、正しいﾌﾟﾛｸﾞﾗﾑ動作を監視するためのｼｽﾃﾑ機能です。WDTはCPUから独立してｸﾛｯｸ駆動され、予め定義される時間経過周期に形態設定され、許可時に定常的に走行します。

WDTが時間経過周期内でﾘｾｯﾄ(WDR)されなければ、ﾏｲｸﾛ ｺﾝﾄﾛｰﾗにﾘｾｯﾄを発行します。これは応用ｺｰﾄﾞからWDTﾘｾｯﾄ(WDR)命令を実行することによって行うことができます。加えて、Atmel AVR XMEGAのWDTは窓動作を持ちます。これはWDTがﾘｾｯﾄされなければならない間の総時間経過周期内側に時間枠または窓を定義することを可能にします。WDTが早すぎまたは遅すぎのどちらかで、この窓の外側でﾘｾｯﾄされた場合、ｼｽﾃﾑ ﾘｾｯﾄが発行されます。標準動作と比べて、これはWDTﾘｾｯﾄ命令の一定実行を引き起こすｺｰﾄﾞ異常の状況を見つけることもできます。従って、等級Bｿﾌﾄｳｪｱがこの窓動作を使用し、閉区間が総周期の最低50%であることが必要とされます。

WDTは許可された場合に活動動作と全ての休止動作形態で走行します。それはCPUから独立したｸﾛｯｸ元から走行し、動作を継続し、そして例え主ｸﾛｯｸが無くてもｼｽﾃﾑ ﾘｾｯﾄを発行します。

Atmel AVR XMEGAには偶然によってWDT設定を変更できないことを保証する、形態設定変更保護機構があります。安全性を増すために、WDT設定を固定化するためのﾋｭｰｽﾞも利用可能です。

WDTがAtmel AVR XMEGA系統での統合安全機能であるため、標準と窓の動作でこの部署を検査する自己診断ﾙｰﾁﾝを設計しました。これは応用の事前初期化領域でﾘｾｯﾄ後、即ちmain関数に先立って実行されます。この検査の流れ図は図6-1.で示されます。

自己診断ﾙｰﾁﾝは基本的に以下を確実にします。

- ｼｽﾃﾑ ﾘｾｯﾄはWDT時間経過後に発行されます。
- WDTはﾘｾｯﾄされ得ます。
- ﾃﾞﾊﾞｲｽは窓動作で時期を得ないWDTﾘｾｯﾄでﾘｾｯﾄされます。

流れ図はﾃﾞﾊﾞｲｽがWDT検査中に多数回ﾘｾｯﾄされることを示します。従って、検査段階の経緯を保つために、SRAM変数とﾃﾞﾊﾞｲｽのﾘｾｯﾄ ﾌﾗｸﾞが自己診断ﾙｰﾁﾝによって使用されます。更に、使用者は低電圧検出(BOD)またはｿﾌﾄｳｪｱ ﾘｾｯﾄに対して何を行うか、または検査が"OK"状態の時にｳｫｯﾁﾄﾞｯｸﾞによってﾘｾｯﾄをどう処理するかを形態設定することができます。

WDT発振器のﾀｲﾐﾝｸﾞを調べるために、自己診断ﾙｰﾁﾝは実時間計数器(RTC)を使用します。RTCはCPUとWDTから独立したｸﾛｯｸ元を持ちます。両部署は公称32.768kHzの周波数で動く発振器を持ちます。けれども、WDT発振器は減らされた精度を犠牲にして低消費電力用に最適化されます。RTCの発振器は正確と仮定することができます。RTCはWDTの周期を推測するのに使用され、ﾌﾟﾛｸﾞﾗﾑはこの推測が間隔(T/2,3T/2:ここでのTはWDTの公称周期)内であることを調べます。

注: RTCはこの自己診断ﾙｰﾁﾝによって暗黙的に検査され、RTCとWDT間の周波数での違いが50%よりも大きければ、異常状態が設定されます。

意図された(異常なしの)実行は次の通りです。

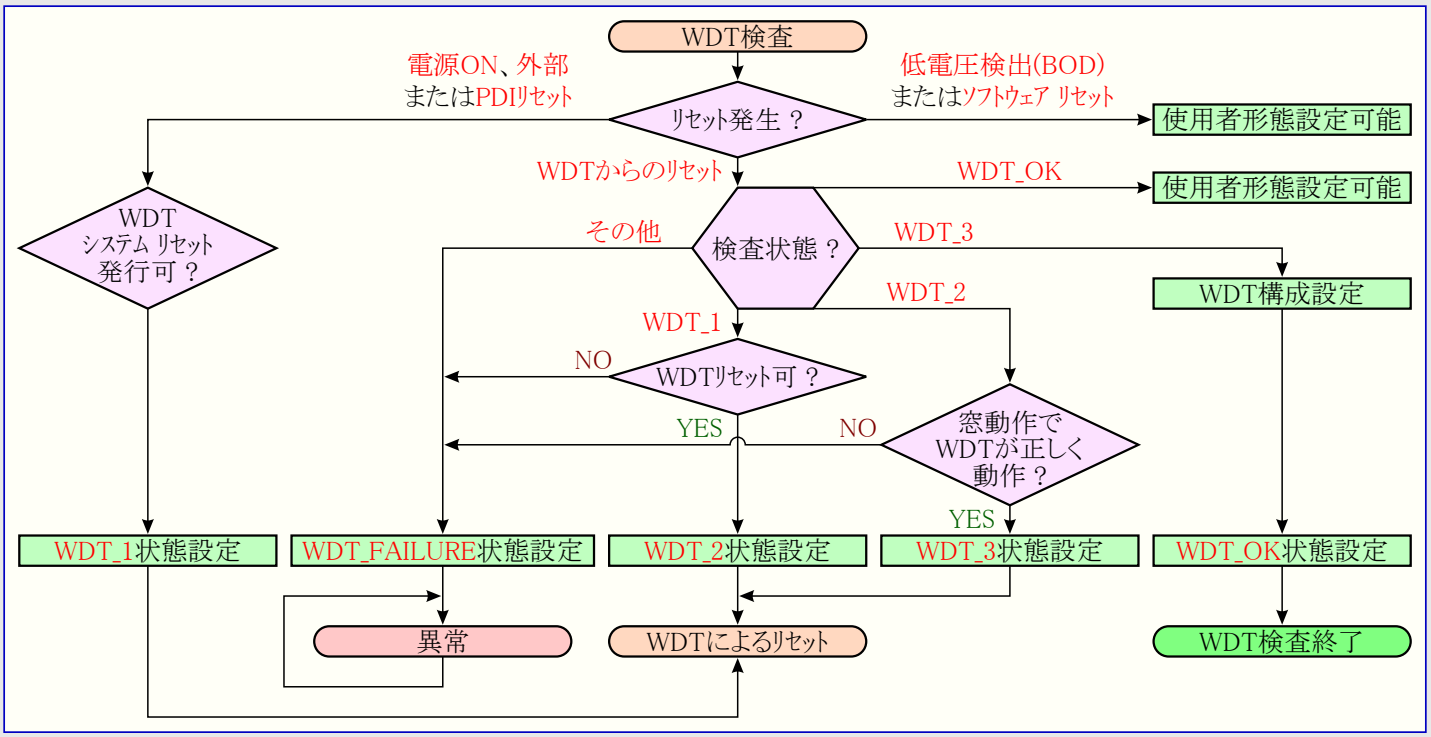
1. 通電または外部ﾘｾｯﾄ後、WDTがｼｽﾃﾑ ﾘｾｯﾄを発行できることを調べます。検査状態1設定とWDTによるｼｽﾃﾑ ﾘｾｯﾄ。
2. WDTがﾘｾｯﾄされ得ることを調べます。検査状態2を設定してWDTがｼｽﾃﾑ ﾘｾｯﾄを発行します。
3. 窓動作が正しく動くことを調べます。検査状態3を設定してWDTがｼｽﾃﾑ ﾘｾｯﾄを発行します。
4. 設定に従ってWDTを構成設定します。検査状態"OK"を設定してmain関数を続けます。

最初の段階はWDTがｼｽﾃﾑ ﾘｾｯﾄを発行できることを確実にすることです。これは基本的にWDTを形態設定ﾌｧｲﾙで指定されたように構成設定し、それがﾘｾｯﾄを発行するまで待つことによって行われます。加えて、検査の後半の段階で必要とされるｳｫｯﾁﾄﾞｯｸﾞ周期の推測にRTCが使用されます。これは小さな周期(概ね3ms)でRTCを構成設定し、ｼｽﾃﾑ ﾘｾｯﾄまでRTC周期数を計数することによって行われます。待つための形態設定可能な最大時間があり、この時間後にﾌﾟﾛｸﾞﾗﾑが異常状態を設定します。

第2段階はWDTがﾘｾｯﾄされ得ることを確実にして、WDTのﾀｲﾐﾝｸﾞを調べることです。異常状態が一時的に設定され、その後に推測したWDT周期が最低よりも大きいことを調べます。WDTとRTCの間の周波数での違いを調べることは、間隔内で両部署が意図したように動くことの確信を与えます。次に、WDTが構成設定され、前の段階で推測されたWDT周期の概ね3/4を待つのにRTCを使用します。これはWDTが意図したよりも速く経過しないことを調べます。この後でWDTはﾘｾｯﾄされ、ﾌﾟﾛｸﾞﾗﾑは再び周期の3/4を待ちます。WDTをﾘｾｯﾄする機構で何か問題があった場合、ｼｽﾃﾑ ﾘｾｯﾄがあると同時にﾌﾟﾛｸﾞﾗﾑは2回目を待ちます。これは総待ち時間が推測されたWDT周期の1.5倍であるためです。加えて、この早期ｼｽﾃﾑ ﾘｾｯﾄは検査を異常状態に置きます。更に、WDTが正しくﾘｾｯﾄされたとの仮定で、ｼｽﾃﾑ ﾘｾｯﾄはその周期の概ね1/4で来るべきです。その後に次の検査状態が設定され、ﾌﾟﾛｸﾞﾗﾑはｼｽﾃﾑ ﾘｾｯﾄを待ちます。

第3段階はWDTが窓動作で正しく動くことを調べます。これは基本的にWDTと検査の次の状態を構成設定してWDTの早期ﾘｾｯﾄを行うことから成ります。時間制限に従わないと仮定すれば、WDTはｼｽﾃﾑ ﾘｾｯﾄを発行すべきです。従って、時期を得ないWDTﾘｾｯﾄ後にﾌﾟﾛｸﾞﾗﾑはWDT周期の1/4間、ｼｽﾃﾑ ﾘｾｯﾄを待ちます。窓動作で何か問題があった場合、ﾃﾞﾊﾞｲｽはﾘｾｯﾄされず、ﾌﾟﾛｸﾞﾗﾑは異常状態を設定します。

図6-1. ｳｫｯﾁﾄﾞｯｸﾞ検査



第4と第5の段階ではﾌﾟﾛｸﾞﾗﾑが単純にWDTを窓動作で構成設定し、OK状態で検査します。この後に主応用が形態設定可能な設定に従ってWDTをﾘｾｯﾄする責任があることに注意してください。

検査が異常状態だったなら、使用者形態設定可能な異常処理部が呼ばれます。信頼に足るｿﾌﾄｳｪｱ応用のために動いているWDTが重要なので、既定によってﾃﾞﾊﾞｲｽは単純に停止するでしょう。

計数器と検査状態変数はｺﾝﾊﾟｲﾗがﾘｾｯﾄ後にそれらを初期化しないように宣言されます。これはﾘｾｯﾄに渡り、それらの使用を許します。Atmel AVR XMEGAはそれが検査の最初の繰り返しかどうかを決めるのに使用されるﾘｾｯﾄ原因を格納するﾚｼﾞｽﾀを持ちます。

実演応用はｽｲｯﾁ(SW0)用の割り込みを構成設定し、LEDをONにし、その後にclassb_error変数が設定されない限り、WDTがﾘｾｯﾄされる繰り返しに留まります。この変数が設定されたなら、LEDはOFFに切り替えられ、応用が終了します。

SW0釦が押されると、ﾌﾟﾛｸﾞﾗﾑはWDT時間超過を導くWDTのﾘｾｯﾄを停止し、従ってｼｽﾃﾑがﾘｾｯﾄします。この"意図せぬ"ﾘｾｯﾄはこの実演でclassb_error変数を設定するように形態設定される自己診断ﾙｰﾁﾝによって捕らえられ、WDTを構成設定して主応用を続けるべきです。従って、釦押下後、LEDはOFFに切り替えられて応用は応答しません。

6.1. 自己診断ﾙｰﾁﾝはどう検査され得るか?

自己診断ﾙｰﾁﾝの最初の段階はｼｽﾃﾑ ﾘｾｯﾄがなければ異常状態を設定します。ｼｽﾃﾑ ﾘｾｯﾄを発行できることからそれを妨げるWDTでの問題は、WDTを開始しないことによって単純に繰り返され得ます。異常状態が設定され、ﾃﾞﾊﾞｲｽは動かないでしょう。

WDTまたはRTCの周波数不全は中断点を設定し、それが確信の間隔の外であるようにrtc_count変数の値を変更することによって模倣動作することができます。

WDTﾘｾｯﾄ機構での機能不全はWDTがﾘｾｯﾄされる行を取り去ることによって模倣動作することができます。与えられたﾌﾟﾛｸﾞﾗﾑの構造で、自己診断ﾙｰﾁﾝの第2段階はWDTが負わせられた時間制限に従わなければ、異常状態を設定します。

WDT窓動作での機能不全はその動作の構成設定を取り去ることによって模倣動作することができます。ｺｰﾄﾞはその後にWDTをﾘｾｯﾄしてWDT周期の1/4間待ちます。その時点でWDT周期は推測されたWDT周期と等しいかより大きいでしょう(総時間経過周期は開区間と閉区間の合計です)。従って、ﾃﾞﾊﾞｲｽは異常状態設定前にﾘｾｯﾄされません。

WDTによって発行されたｼｽﾃﾑに対して形態設定された活動は実演応用を通して検査することができます。釦押下はWDT経過を導き、その後に自己診断ﾙｰﾁﾝはこの場合にWDTを構成設定してclassb_error変数を設定する、形態設定された活動を実行します。

7. 割り込み処理

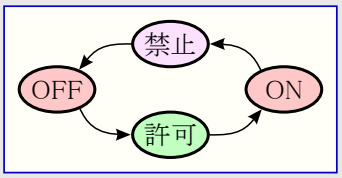
割り込みは周辺での状態の変化を合図し、これはﾌﾟﾛｸﾞﾗﾑ実行後に使用することができます。組み込み応用は、例えば実時間で事象に応答するのに割り込みを使用します。従って、ｼｽﾃﾑが割り込み機能での異常を検出できることが重要です。特に、等級B応用は割り込みが度々意図されるように(何れにしても)実行されていないかどうかを検出することができるべきです。

選ばれた方法は実時間計数器(RTC、CPUｸﾛｯｸから独立したｸﾛｯｸ)での時間枠監視です。要するに、監視されるべきどの割り込みもそれが実行される時毎に計数器を増加しなければなりません。更に、RTCは割り込み計数器が調べられる場所で周期的に割り込みを生成します。何れかの計数器が形態設定可能な割り込み指定範囲外なら、異常処理部が呼ばれます。

割り込み監視部は登録と活性の両方が成されたこれらの割り込みの頻度を調べます。割り込みの登録は調べるための割り込みでの以下の情報、識別子、意図される頻度、許容変動の割り込み監視情報を与えることを意味します。割り込み活性は登録された割り込みの検査を開始すべきことを監視部に告げます。動的な割り込みの監視を活性にするために、登録された各割り込みに関連する状態変数を実装しました。状態間で可能な遷移が図7-1.で示されます。

どの割り込みに対しても既定状態はOFFで、この状態では割り込み管理部が割り込みの頻度を調べません。割り込みの状態は主応用が割り込み頻度の調査の開始を監視部に求める時に許可に設定されるべきです。次回に割り込み監視部が実行されると、割り込み状態はONに変わります。これは割り込み計数器が割り込み監視部と同期される、即ち割り込み計数器が割り込み監視部の区間後、正確に増加を開始することを保証します。同様に、割り込みがこれ以上監視されるべきでないと主応用が決めた場合、割り込み状態は禁止に設定されるべきです。割り込み監視部は次にそれが実行される時に状態をOFFに変えます。

図7-1. 割り込み状態



注: RTC周期当たりに1つの許可/禁止の要求だけであるべきです。

実装された割り込み監視部は主応用の強化を更に増す2つの機能を持ちます。最初のものは割り込みがONの場合にだけ増加する割り込み計数器で、従ってこの計数器は割り込みがOFFと同時に0にされるべきです。これは登録された全ての割り込みに対して割り込み計数器によって調べられ、さもなければ異常処理部が呼ばれます。2つ目の機能は割り込みを許可するONまたは割り込みを禁止するOFFが、CLASSB_STRICTが定義されている場合に異常処理部を呼ぶことです。

RTCは周期的に割り込み監視関数を呼びます。全ての登録した割り込みがそれらの状態に従って処理されます。活性な割り込みはそれらの頻度を調べられます。異常が無ければ、割り込み監視部は計数器をﾘｾｯﾄします。さもなければ異常処理部が呼ばれ、監視が直ちに終了します(これは形態設定可能です)。不活性割り込みの計数器は一貫性に関して0と比較されます。主応用が割り込みを活性にするように要求された場合、それの状態はONに設定され、その逆も同様です。後者の場合に割り込み計数器がﾘｾｯﾄされることに注意してください。

割り込みを監視するため、以下の段階に従うべきです。

1. 割り込みは割り込みに識別子を与えることによってclassb_int_identifiersで宣言されるべきです。
2. 主応用はclassb_intmon_reg_int()を呼ぶことによって割り込みを登録しなければなりません。その後に割り込みに対する構造体が作成されます。
3. RTCは周期的に割り込みを生成して割り込み監視部を呼び戻すように構成設定されなければなりません。
4. 監視されなければならない割り込みは各実行でclassb_intmon_increase()を呼ぶべきです。
5. 主応用は監視部が割り込みの調査を開始することを要求しなければなりません。これはclassb_intmon_set_state()で割り込み状態を許可に変更することによって行われます。
6. 或る時点で割り込みがこれ以上監視されるべきでないなら、主応用は状態を禁止に変更することができます。

応用例ではﾀｲﾏ/ｶｳﾝﾀ(TC)が監視部によって調べられる周期的溢れ割り込みを生成するように構成設定されます。加えて、応用は2つのｽｲｯﾁ割り込みを構成設定します。最初のものはTC割り込みの頻度を変更します。2つ目のものは監視部でのTC割り込みを不活性にします。更に、応用が正しく動いていることを示すためにLEDがONに切り替えられます。応用は釦が押されない限り、LED ONで繰り返しに留まります。最初の釦が先に押されたなら、TC割り込みの頻度が変更され、故に監視部は異常ﾌﾗｸﾞを設定します。それは主応用を繰り返しに留まってLEDをOFFに切り替える結果に帰着します。けれども、監視部でのTC割り込みを不活性にするために2つ目の釦を押すことができます。この場合、最初の釦押下は未だ割り込み頻度変更に至りますが、監視部は異常を生成しません。

割り込み監視部は動いているRTCを頼ります。後者はﾗｲﾌﾞﾗﾘの他の部品(WDTとCPUの周波数検査)を支援し、関連するｿﾌﾄｳｪｱ単位部で調査されます。従って、RTCで欠陥がないと仮定することができます。けれども、応用の信頼性を増すために、登録した割り込みは割り込み内で検査され得る計数器に対する最大値を持つことができます。計数器がこの値に達した場合、割り込み監視部が動いていないと仮定され得ます。

監視部は以下のように検査することができます。まず第一に、割り込みが構成設定され得て、その後に異常を生成するために周波数が変更され得ます。これは応用例で実装されます。2つ目に、割り込みの計数器はﾃﾞﾊﾞｯｸﾞ中に変更することができます。これは割り込みの活性と不活性の両方を行うことができます。更に、ｼﾝﾎﾞﾙのCLASSB_STRICTが定義される場合、ﾌﾟﾛｸﾞﾗﾑは異常を導く割り込みを2度、活性または不活性にすることができます。

8. システム クロック

自己診断ルーチンは実時間計数器(RTC)とタイマ/カウンタ(TC)を使用して実装されています。この周辺機能がCPUと同じクロック領域を持つためTCを選びました。けれども、RTCは独立したクロック元からクロック駆動することができます。例えばCPUが内部2MHz発振器から、RTCが内部32.768kHz発振器からクロック駆動することができます。

RTCは周期的に割り込みを生成するように構成設定されます。この割り込みでは、TC内の計数器がそれの意図された値と比較されて、それが形態設定可能な値内でなければ、異常処理部が呼ばれます。

TC溢れ割り込みでは16ビット溢れ計数器変数が増加されます。これは殆どの場合でTCがRTCよりもかなり高い周波数で走行し、故にそれが32ビット計数器を持つ必要があるからです。加えて、溢れ計数器が形態設定された閾値よりも大きい場合に異常処理部が呼ばれます。溢れ計数器がRTC割り込みで解除されると仮定すれば、閾値よりも大きな溢れ計数器はRTC割り込みが意図されるように度々起きないことを意味します。

これら2つの測定(TCに対するRTCの調査とその逆)を考慮すると、自己検査単位部はRTCかTCのどちらかの欠陥を検出します。未検出異常の危険は、RTCとTCの両方での欠陥があるけれど、それらの周波数の比率が正しい場合の筋書きに対して減らされます。

RTC割り込みはまず第一に、実効計数器が32ビットを持つように溢れ計数器の値がTCに追加される関数を呼び戻します。その後に32ビット計数器は参照基準と比較されます。差が意図されるよりも大きければ、異常処理部が呼ばれます。その後に溢れ計数器とTCはリセットされます。

応用例は前の例と同様です。既定により、XMEGAデバイスは自己診断ルーチンのパラメータを構成設定する時に考慮されるシステム周波数である内部2MHz発振器で走行します。釦が押されたなら、システム クロックは32MHz発振器での走行に設定されます。これは検査を失敗させてLEDがOFFに切り替えられます。

システム周波数自己検査の実装は異なるRTCとTCの形態設定を使用することが可能なように柔軟です。RTC周波数とRTC割り込み周期はRTCに関連するファイルで形態設定することができます。異なるRTC構成設定は、クロック元がCPUから独立し、CLASSB_RTC_INT_PERIODとCLASSB_RTC_FREQの定数がRTC設定に従って定義されている限り、我々の自己診断ルーチンと共通です。更に、TC部署、前置分周器、(%での)検査に対する許容、システム周波数を形態設定することも可能です。後者は主応用によって設定される実際のシステム周波数に対応しなければならず、即ち自己診断ルーチンはその設定に従ってクロック系を変更しません。

この自己診断ルーチンを検査するには多数の方法があります。先ず第一に、応用例は釦押下後にシステム周波数を変更します。次に、F_CPUシステム周波数定数は2MHzである実際のシステム周波数と合わないように変更することができます。これは許容値に於ける変更と組み合わせることができます。第三に、RTCまたはTCでの問題を模倣動作するために、構成設定関数を注釈にすることができます。

9. メモリ

本章ではメモリに対する標準的な必要条件と実装した自己診断検査を記述します。9.1.項は不変メモリ、即ちAtmel AVR XMEGA系統での内部フラッシュ メモリとEEPROMを参照します。9.2.項は内部SRAMである可変メモリを参照します。

9.1. 不変メモリ

巡回冗長検査(CRC)はデータ内の偶然の誤りを見つけるのに使用される誤り検出技術検査算法です。この方法は一般的に、データ送信や、データとプログラム メモリ内に存在するデータの正確性を判定するのに使用されます。

CRC算法は入力データの流れ、またはデータの塊を処理し、後で誤りを検出するのに使用することができるチェックサム出力を生成します。これを行うための2つの一般的な方法は以下から成ります。

- データの最初のチェックサムを計算してそれを格納します。誤りを検出するために2つ目のチェックサムを計算して先のものと比較することができます。それらが異なる場合、誤りがあります。
- データの最初のチェックサムを計算してそれをデータ領域に追加します。誤りを検出するために、新しいデータ領域に対して2つ目のチェックサムを計算することができます。結果のチェックサムは固定の実装指定値であるべきで、さもなければ誤りがあります。

代表的に、任意長のデータ塊に適用されるnビットCRCはnビットよりも長くないどんな単一集中誤りも検出し、より長い全ての集中誤りの1-2-n断片を検出します。データに誤りがある場合、応用は修正活動、例えば再びデータを送ることを要求、または単に不正データを使用しない、を取るべきです。

我々はCRCに対するハードウェアやソフトウェアの実装に基づいて自己診断単位部を開発しました。更に一般的に使用される2つのCRC規格が支援されます。

- 16ビットCRC CCITT
- 32ビットCRC IEEE® 802.3

ソフトウェア実装は全てのXMEGAデバイスによって使用することができます。この場合、CPUはデータを読んでCRCチェックサムを計算します。2つのソフトウェア実装を選ぶことが可能です。

- 参照表：これは計算速度向上のためにCRC参照表を使用します。参照表は(16ビットに対して)512または(32ビットに対して)1024バイトのフラッシュ メモリが必要です。
- 直接計算：これは多項式の除算を用いて各バイトに対してチェックサムを計算します。この除算はフラッシュ メモリ内で空間を占有しませんが、参照表法よりも遅くなります。

ソフトウェア実装で、使用されるCRC32多項式は$EDB88320、初期剰余は$FFFFFFFFで、生成されたチェックサムはビット逆順で補数にされます。使用されるCCITT多項式は$1021、初期剰余として$0000です。この場合、チェックサムはビット逆順と補数のどちらでもありません。

ハードウェア実装は新しいXEMGA AUデバイスで含まれたCRC部署をインターフェースするためのドライバを頼ります。これは他のXMEGAデバイスがソフトウェア実装を使用しなければならないことを意味します。この場合、例えCRCハードウェア部署がチェックサムを計算しても、CPUは未だデータを読んで本部署に供給することを必要とし得ます。フラッシュ メモリでの32ビットCRC計算については、CPUからの支援なしに処理全体を走行することができます。けれども、16ビット計算については、CPUがフラッシュ メモリからデータを読んでCRC部署に転送しなければなりません。SRAM、EEPROMまたは周辺機能内のデータのCRC計算については、DMA単位転送が構成設定されなければ、CPUがデータをCPU部署に転送しなければなりません。CRC-CCIT多項式は$1021でIEEE 802.3のそれは$04C11DB7です。初期値と最終チェックサムに対する構成設定は先に記述されるのと同じです。

EEPROM内に格納されたデータに対してチェックサムを計算するのに以下の関数が利用可能です。

- CLASSB_CRC16_EERPOM_SW
- CLASSB_CRC16_EERPOM_HW
- CLASSB_CRC32_EERPOM_SW
- CLASSB_CRC32_EERPOM_HW

同様に、フラッシュ メモリ内に格納されたデータに対してチェックサムを計算するのに以下の関数が利用可能です。

- CLASSB_CRC16_FLASH_SW
- CLASSB_CRC16_FLASH_HW
- CLASSB_CRC32_FLASH_SW
- CLASSB_CRC32_FLASH_HW

注: ハードウェアとソフトウェアの実装は等価CRC算法が同じチェックサムを生成するように形態設定されています。けれども、処理時間に於いて重要な違いがあります。

例として、ハードウェア部署を使用してフラッシュ メモリに対して32ビットCRCを計算する関数を記述します。先ず第一に、関数は必要とする情報、計算の形式、開始アドレス、データのバイト数、それとEEPROM内に格納された"正しい"チェックサムへのポインタを受け取ります。第二として、"新しい"チェックサムを計算するのにCRC部署のドライバが使用されます。最後に、チェックサムが計算され、何か異常があった場合に異常処理部が呼ばれます。

応用例では予め計算されたチェックサムといくつかのデータがEEPROM内に格納されます。その後、プログラムはシステムの正しい機能を示すためにLEDを点灯して割り込みを生成するように釦を構成設定します。その後に異常全体フラグが設定されない限り、プログラムは繰り返しに留まります。釦が押されたなら、フラッシュとフラッシュ内の応用領域それとEEPROM内に格納されたデータが検査されます。誤りがあった場合、異常フラグが設定され、主応用はLEDをOFFに切り替えます。通常状態では、フラッシュ内の応用領域の内容とEEPROM内に格納されたデータは正しいでしょう。これは釦押下が異常フラグ設定に至らず、その後にプログラムが主繰り返しに戻り、LEDがONに留まることを意味します。釦は再び押すことができ、再び同じ結果になり得ます。

自己診断検査が正しく動くことを調べるために、EEPROM内に格納されたデータまたはフラッシュ内の応用領域の内容を変更することができます。利便性のため、応用例はEEPROM内に格納されたデータの変更、またはLEDをOFFに切り替える行の後の追加繰り返しのコンパイルを定義することができる2つのシンボルを持ちます。この場合、LEDは釦が押されるまでONです。その後にCRC検査はデータが変更されたために失敗して異常フラグが設定されます。これは繰り返しを去る主応用とOFFに切り替えられつつあるLEDに至ります。

もう一度の釦押下は無効です。最後に、全ての等価CRC計算が同じチェックサムを導くことを調べるために、それらを呼んでその結果を比較することが可能です。ソフトウェア実装に関する算法(参照表または直接計算)が対応するヘッダ ファイルで設定することによって選択され、1つの実行中に1つ算法だけを呼ぶことができます。或る算法または他方の選択がフラッシュの内容を変更するため、代わりにEEPROMに対する各種実行に渡るチェックサムが比較されるべきです。

9.2. 可変メモリ

実装された特定マーチ算法は次のように記述することができるマーチX検査として知られます。

算法1. マーチX検査

⇕(*w*0);⇑(*r*0,*w*1);⇓(*r*1,*w*0);⇕(*r*0)

最初の段階は何れかの順で全てのメモリ位置に0を書くことです。第2段階は以下のように最下位アドレスで開始して各ビットで実行される3つの操作から成ります。

- ビットを読んでそれが0であることを確認します。それが1ならば失敗が起きました。
- その位置に1を書きます。
- 次のビットへ繰り返します。

第3段階は以下のように(第2段階に対して)逆のアドレス順で行われる3つの操作から成ります。

- ビットを読んでそれが1であることを確認します。それが0ならば失敗が起きました。
- その位置に0を書きます。
- 次のビットへ繰り返します。

第4と最終の段階は何れかの順で全ビットが0であることを確認するから成ります。第2と第3の段階で使用される実際のアドレス順はそれらが正確に逆順で行われる限り、問題になりません。更に、この検査は第3と第4の段階が飛ばされる、良く知られたマーチC検査と等価です。

マーチXは以下の欠陥を検出することができます。

- アドレス復号器
- 単一セル欠陥：動かない、遷移またはデータ保持力の欠陥
- メモリ セル間の欠陥：いくつかの連結欠陥(CF:Couping Fault)

我々が記述したマーチ検査はビット志向メモリ(BOM:Bit-Oriented Memory)用に定義されます。けれども、XMEGAのSRAMは語志向メモリ(WOM:Word-Oriented Memory)です。$r0$, $r1$, $w0$, $w1$を各々、r^D, $r^{\bar{D}}$, w^D, $w^{\bar{D}}$によって置き換え、ここでのDはどんなデータ環境でも有り得、BOMマーチ検査は内部語CFを網羅するWOMマーチ検査に変換されます。特に、我々の実装ではデータ環境D=$00を選びました。内部語CFを網羅するため、各種データ環境で内部語マーチ検査を結びつけることが可能です。

メモリ セル配列の列内のビットの物理位置の考慮が重要です。XMEGAの場合、1語からのビットは交互配置、即ち他の語からのビットによって物理的に分離されます。従って、異なる語からのセル間のCFの可能性は減らされます。いくつかの欠陥模式に於いて侵略セルが物理的に犠牲セル近隣でなければならないため、多分、いくつかのソースは交互配置されるメモリに対して内部語CFを考慮しません。加えて、内部語マーチ検査はいくつかの内部語CF、例えば内部語反転CFを網羅します。それらの要素を考慮に取り入れ、検査の内部語部分は規格の必要条件を満たすのに多分充分です。けれども、提案された自己診断ルーチンは環境の流れ($55,$AA,$AA,$CC,$0F,$F0)Dの付加的なマーチ要素を追加するように形態設定することができます。これは無制限内部語CF模式で考慮される内部語状態CFに対する適用範囲を加えます。

検査速度向上のため、メモリは順に検査される形態設定可能な領域数に分割されます。検査の最も単純な動きはメモリ領域間で重複がない時です。この場合、最後の1つの可能性を除き、全ての領域は同じ大きさを持ちます。(緩衝部として参照される)最初のメモリ領域は予約され、検査によって他の領域が検査されつつある間にそれらの内容を格納するのに使用されます。これは実装した一致検査が破壊的と仮定すれば必要です。

マーチX算法が一度に1つのメモリで走行すると仮定すれば、内部語CFが検出されない可能性を減らすメモリ領域間の形態設定可能な重複があります。毎回メモリ領域は検査され、直前の領域の部分が更に検査されます。これはそれが最初の領域であるため、緩衝部に適用されないことに注意してください。緩衝部の大きさは(2つ目の領域の大きさが同様に減らされる)前の場合に関して拡大されることが必要です。

応用にどう組み込まれ得るかを示す応用例が含まれます。基本的に、系の正しい動きを合図するLEDが点灯され、その後にプログラムは異常がない限り、SRAMが検査される繰り返しに留まります。

自己診断ルーチンは次のように検査することができます。先ず第一に、マーチX算法を実装する関数に多数の中断点を設定することができます。次に、いくつかのメモリ位置の内容は内部語結合欠陥の各種形式の模式に変更することができます。検査単位部はその後に応用が主繰り返しを抜けてLEDがOFFに切り替えられるのを導く異常フラグを設定します。

9.3. 更なる適用範囲

SRAM、フラッシュ、EEPROMがXMEGAでの内部メモリであることを考慮し、先に記述した自己診断ルーチンはメモリ アドレス指定と内部データ経路に対する規格の仕様を網羅します(表H.11.12.7での副部品4.3と部品5)。

10. アナログ入出力

A/D変換器(ADC)とD/A変換器(DAC)が意図したように動くことを保証するためにもっともらしい検査が行われます。この検査は基本的に以下の段階から成ります。

1. DACは基準電圧として1Vを使用して5つの値(0%,25%,50%,75%,100%)を出力するように形態設定されます。
2. ADCは内部入力とDACと同じ基準電圧を使用するように形態設定されます。
3. ADCの多重器は生成された各値に対してADCがDACから読むようにADCが構成設定されます。
4. 意図される値はADC特性に従った閾値とADC形態設定に従って設定されます。
5. 読み込み値が閾値よりも大きく意図した値から離れた場合、検査は異常処理部を呼びます。これは異常処理部ヘッダ ファイルで設定されます。

我々はアナログ入出力検査が応用にどう組み込まれるかを示す応用例を含めました。系の正しい動きを合図するLEDが点灯され、その後にプログラムは異常が無い限り、繰り返しに留まります。釦が押された時に、ADCとDACが検査され、プログラムは主繰り返しに戻ります。

自己診断ルーチンを検査する違う方法があります。先ず第一に、ADCに書かれる値またはADCの意図される値は変更することができ、そしてこれはADC読み込み値不正を導くでしょう。代わりに、読み込み測定がほぼ確実に範囲外であるように閾値を減らすことができます。これらの全ての場合で、検査単位部は応用が主繰り返しを抜け出してLEDのOFF切り替えを導く、異常フラグを設定します。次に、部署での失敗はDACとADCの部署の許可ビットの解除(これは部署を禁止します。)によって模倣動作することもできます。これは自己診断ルーチンの内側に中断点を設定して手動でそのビットを解除することによって行われます。更に、これは自己検査が変換の終りを待って動かないことに至り、そしてこれは結局WDTのシステム リセット発行を導きます。

11. XMEGAの付加安全機能

Atmel AVR XMEGA系統は安全性を増すのに使用することができるいくつかの付加機能を持ちますが、等級B必要条件のどれでも直接扱いません。

- 形態設定変更保護(CCP:Configuration Change Protection)は時間制限ｺｰﾄﾞ手順に従わない場合に、或るI/Oﾚｼﾞｽﾀと不揮発性ﾒﾓﾘの変更を妨げます。
- ﾋｭｰｽﾞは上書きからいくつかの形態設定とﾌﾗｯｼｭの領域を固定化するのに使用することができます。
- ﾋｭｰｽﾞは応用ｿﾌﾄｳｪｱによって容易に読んで検証することができます。
- 低電圧検出(BOD:Brown-Out Detection)、電源ONﾘｾｯﾄ(POR:Power On Reset)、ｽﾊﾟｲｸ検出(SD:Spike Detection)は電力低下、多すぎる変動などの時にｺｰﾄﾞ実行を保護することができます。
- 内部温度感知器は不正な動作状況やﾊｰﾄﾞｳｪｱ欠陥を検出するのに使用することができます。
- ｸﾛｯｸ ｼｽﾃﾑはｸﾛｯｸ元を切り替える前に発振器の安定性を調べます。
- XOSC低下/停止検出は例え外部発振器が低下/停止しても、ﾃﾞﾊﾞｲｽが動作を停止しないことを保証します。

12. 謝辞

13. 参照と示唆される文献

- "IEC 60730-1: Automatic electrical controls for household and similar use", 国際電気標準会議, Ed. 3.2, 2007-03
- Michael Lee BushnellとVishwani D. Agrawalによる"Essentials of electronic testing for digital, memory, and mixed-signal VLSI"
- "A Designer's Guide to Built-in Self-Test", C. E. Stroud, Kluwer Academic Publishers, 2002
- Atmel AVR040:電磁適合性(EMC)設計の考察
- Atmel AVR042:AVRﾊｰﾄﾞｳｪｱ設計の考察
- Atmel AVR1310:XMEGAｳｫｯﾁﾄﾞｯｸﾞ ﾀｲﾏの使い方

14. 改訂履歴

資料改訂	日付	注釈
42008A	2012年6月	初版資料公開

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
TEL (+1)(408) 441-0311
FAX (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
TEL (+852) 2245-6100
FAX (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parking 4
D-85748 Garching b. Munich
GERMANY
TEL (+49) 89-31970-0
FAX (+49) 89-3194621

Atmel Japan G.K.
141-0032 東京都品川区
大崎1-6-4
新大崎勧業ﾋﾞﾙ 16F
ｱﾄﾒﾙ ｼﾞｬﾊﾟﾝ合同会社
TEL (+81)(3)-6417-0300
FAX (+81)(3)-6417-0370

本応用記述はAtmelのAVR1610応用記述(doc42008.pdf Rev.42008A-06/12)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。