応用記述

AVR3004：安全機能を持つQTouch

Atmel QTouch

要点

- IEC60730等級B規格の重要性
- 等級BでのAtmel® QTouch®応用
- 安全機能を検証するための検査の筋書き

概要

様々な家庭用品と白物家電、即ち。食器洗浄器、洗濯機、冷蔵庫、冷凍庫、電磁調理器、乾燥機などで使用される電子装置の信頼性と安全な動作に対して、国際電気標準会議(IEC)はIEC60730規格を導入しました。IEC60730の等級Bｿﾌﾄｳｪｱ種別は安全でない動作を防ぐｿﾌﾄｳｪｱ実装が必要です。

この資料の目的はAtmel容量性接触解決策に基づくAtmel QTouchﾗｲﾌﾞﾗﾘに安全機能を追加する方法を記述することです。この資料は故障状況を模倣して安全機能を検証するための検査の筋書きを提供します。提供される指針はIEC60730等級Bの必要条件に適応するためのQTouch応用の開発を使用者に許します。完全な応諾に関して使用者は必要とされるように検査を独自化することができます。

目次

1. 安全検査の概要

IEC60730に従って、単一チャネル応用(1つのﾏｲｸﾛ ｺﾝﾋﾟｭｰﾀだけを使用する応用)は以下を実行することができるﾌｧｰﾑｳｪｱ設定を持たなければなりません。

- 始動機能検査
- 周期的機能検査

単一チャネル機能検査は今日使用される最も一般的で実装が最も容易な機構です。機能の組が始動で検査され、いくつかの機能は周期的に検査されます。検査結果に基づき、適切な活動が取られます。

検査機能は2つの部分、即ち、等級BとFMEAに分離されます。MCUの内部的な部署を検査する機能検査は等級B検査で網羅され、I/Oﾚﾍﾞﾙでの接続性を検査するQTouchに対する機能検査は故障種別と影響解析(FMEA:Failure Mode Effect Analysis)で網羅されます。一般的に、ﾘｾｯﾄが発行されるか、またはMCUに対して内部的である異常状態の間中CPUはｱｲﾄﾞﾙを保持されます。

1.1. 等級B

等級B検査はMCUの様々な部署/周辺機能の機能的な検査を網羅します。等級Bで網羅される検査はどの応用に対しても共通です。以下の検査はMCU機能を調べるために実行されます。

- CPUとｽﾀｯｸ ﾚｼﾞｽﾀ
- ｳｫｯﾁﾄﾞｯｸﾞ機能
- CPUｽﾃｰﾀｽ ﾚｼﾞｽﾀ
- SRAM
- 計時器機能
- 割り込み機能

以下の副項は各々の機能的な検査を記述します。検査が実行される順番は重要で、検査は上で言及されるのと同じ順で処理されなければなりません。例えば、汎用ﾚｼﾞｽﾀはSRAMを調べる前に調べられるべきです。また、等級Bでのどんな故障も応用での停止またはﾘｾｯﾄのどちらかに帰着すべきです。

1.1.1. CPUとｽﾀｯｸ ﾚｼﾞｽﾀ

汎用CPUﾚｼﾞｽﾀのR0～R25,X,Y,Zは固着に対して検査されます。この検査は各ﾚｼﾞｽﾀで$55と$AAの値を連続的に書いて読むことによって実行されます。同様に、ﾎﾟｲﾝﾀ ﾚｼﾞｽﾀも固着に対して検査されます。このｺｰﾄﾞはｱｾﾝﾌﾞﾘ言語で書かれ、主関数実行前に実行されなければなりません。

1.1.2. ｳｫｯﾁﾄﾞｯｸﾞ機能

ｳｫｯﾁﾄﾞｯｸﾞはﾘｾｯﾄを発行するように形態設定され、これはﾘｾｯﾄで監視されます。ｳｫｯﾁﾄﾞｯｸﾞ ﾘｾｯﾄ成功で検査は通過として宣言されます。

1.1.3. CPUｽﾃｰﾀｽ ﾚｼﾞｽﾀ

CPUｽﾃｰﾀｽ ﾚｼﾞｽﾀのSREGは$55と$AAを書いて読むことによって固着を調査されます。

1.1.4. SRAM

最初に、SRAMは$55と$AAを書いて読むことによって固着に対して調査されます。また、SRAM位置に対応するｱﾄﾞﾚｽの補数がその位置に書かれて読み戻されます。これはｱﾄﾞﾚｽ ｴﾝｺｰﾀﾞの正しい機能を保証します。その後、行進B検査が行われます。行進B検査の間、SRAMは2つの部分に分割されて各々の部分が独立して検査されます。

注: 行進B検査はAtmelの「AVR998:AVR®ﾏｲｸﾛ ｺﾝﾄﾛｰﾗでのIEC60730等級B適合への指針」で記述されます。

1.1.5. 計時器機能

最初に$55と$AAを書いて読むことによって計時器ﾚｼﾞｽﾀが固着に対して調査されます。

その後に計時器は計時器計数ﾚｼﾞｽﾀ値が比較ﾚｼﾞｽﾀ値と一致する時に割り込みﾌﾗｸﾞが設定されるように形態設定されます。特定時間後に割り込みﾌﾗｸﾞが調べられます。割り込みﾌﾗｸﾞが設定されていない場合、検査は失敗として宣言されます。

1.1.6. 割り込み機能

割り込み機能を検査するため、1つの計時器が溢れ割り込みを発行するように形態設定されてそれが監視されます。割り込みが起きなければ、検査は失敗として宣言されます。

1.2. FMEA

MCUの内部機能が正しく動くにも関わらず、ﾋﾟﾝでの物理的な障害は接触動作での不規則な動きを導くかもしれません。これらの異常は識別されて是正処置がとられるべきです。この資料で言及されるFMEA検査はQTouch応用特定で他の応用に使用することはできません。

FMEA検査は(SNSとSNSKの両方の)接触ﾋﾟﾝでの入出力ﾋﾟﾝ ﾚﾍﾞﾙの機能的な検査を網羅します。以下の検査がSNSとSNSKのﾋﾟﾝで実行されます。

- VCCへの短絡
- GNDへの短絡
- 他の接触ﾋﾟﾝへの短絡
- 採取ｺﾝﾃﾞﾝｻ(Cs)短絡
- 異常な信号値

以下の副項は各々のFMEA機能検査を記述します。これらの検査が実行を必要とする特定の順番はありません。

1.2.1. VCCへの短絡

全てのﾁｬﾈﾙのSNSとSNSKのﾋﾟﾝはそれらがVCCに短絡されているかどうかを調査されます。特定ﾁｬﾈﾙのSNSまたはSNSKがVCCに短絡されている場合、対応するﾁｬﾈﾙが欠陥として報告されます。

1.2.2. GNDへの短絡

全てのﾁｬﾈﾙのSNSとSNSKのﾋﾟﾝはそれらがGNDに短絡されているかどうかを調査されます。特定ﾁｬﾈﾙのSNSまたはSNSKがGNDに短絡されている場合、対応するﾁｬﾈﾙが欠陥として報告されます。

1.2.3. 他の接触ﾋﾟﾝへの短絡

この検査に於いて、接触ﾋﾟﾝはそれらが同じﾎﾟｰﾄからの他の接触ﾋﾟﾝに短絡されていないかどうかを調査されます。2つの接触ﾋﾟﾝが短絡されている場合、両方の接触ﾋﾟﾝに対応するﾁｬﾈﾙが欠陥として報告されます。

1.2.4. 採取ｺﾝﾃﾞﾝｻ(Cs)短絡

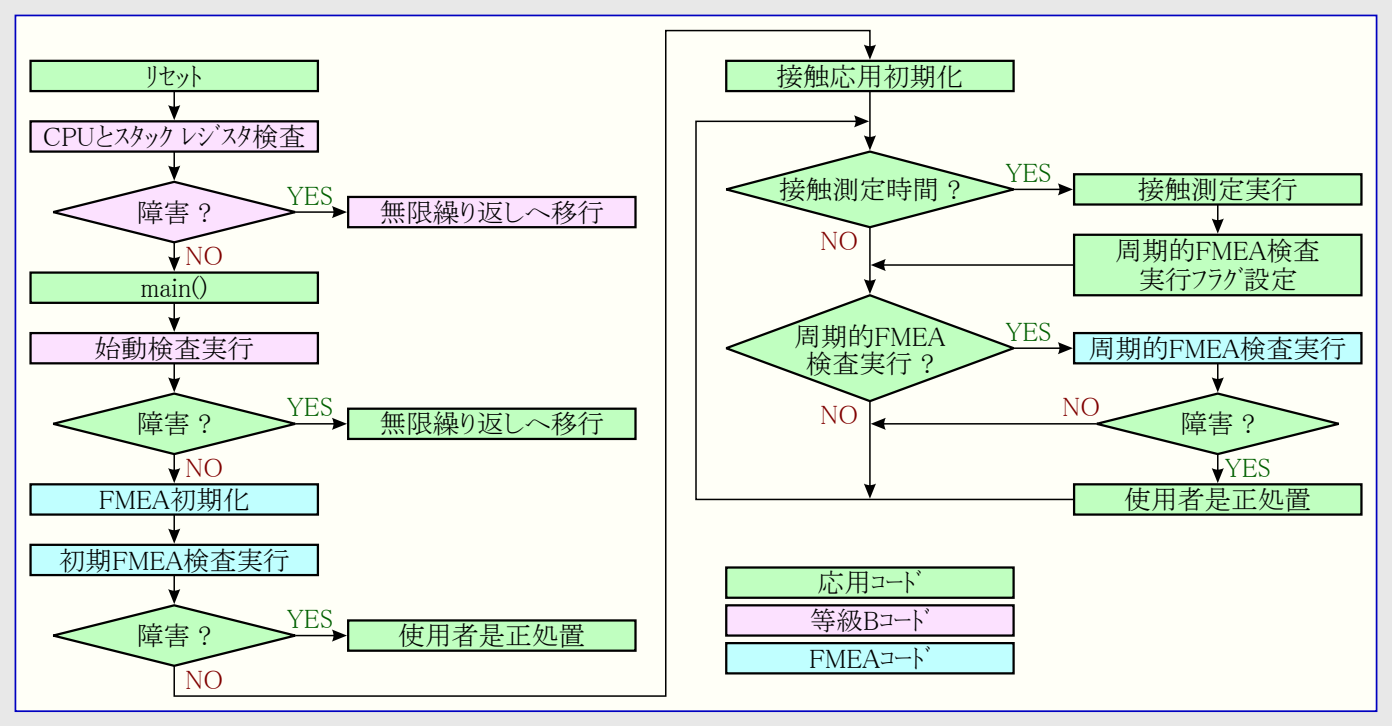
全てのﾁｬﾈﾙの採取ｺﾝﾃﾞﾝｻが短絡に関して調べられます。採取ｺﾝﾃﾞﾝｻが短絡されている場合は対応するﾁｬﾈﾙが欠陥として報告されます。内部ﾎﾟｰﾄ形態設定に於いて、Cs短絡は「採取ｺﾝﾃﾞﾝｻ短絡」と「他の接触ﾋﾟﾝへの短絡」の異常として報告されるでしょう。

1.2.5. 異常な信号値

各ﾁｬﾈﾙの信号値は予め定義された範囲で調査されます。ﾁｬﾈﾙの信号値が範囲内でない場合はﾁｬﾈﾙが欠陥として報告されます。

1.3. 応用流れ図

以下の流れ図は等級BとFMEAが統合されたQTouch応用を記述します。



1.4. 機能検査を実行するAPIの概要

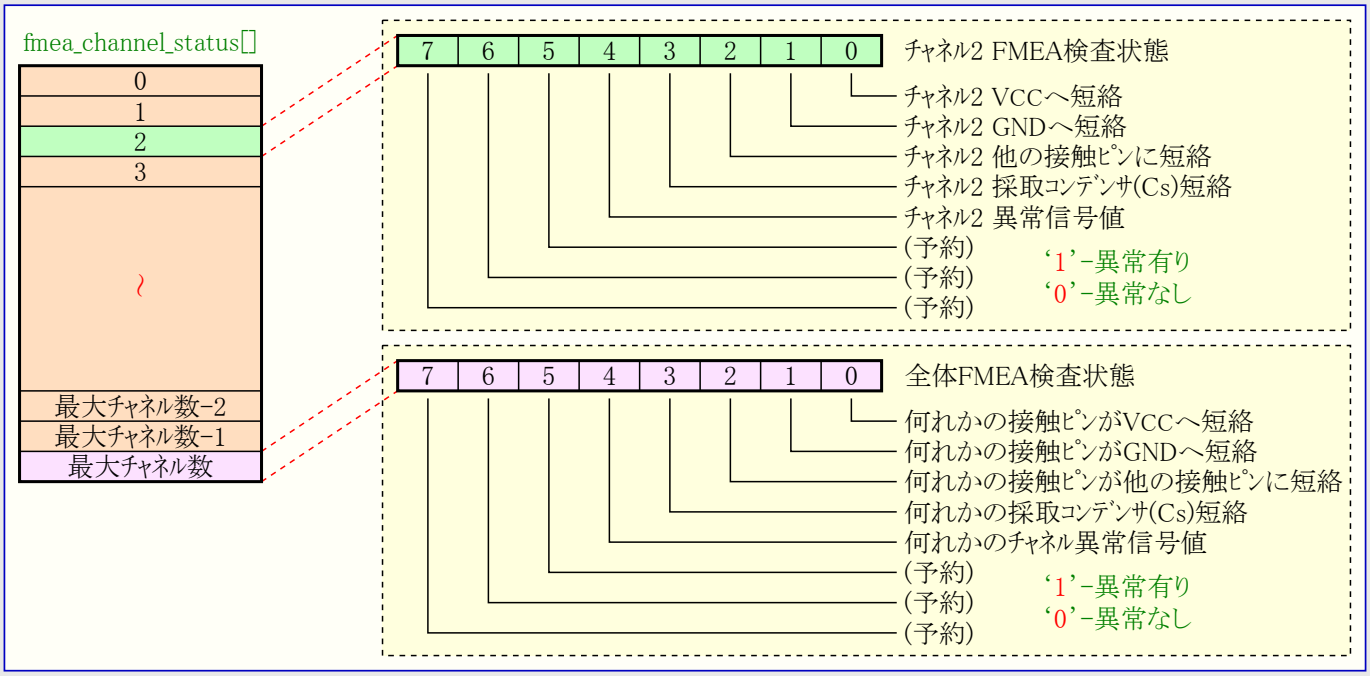
機能検査は各単位部が固有の検査を実行するように単位分割化されます。下表は重要なAPIの一覧を与えます。

API	説明
void __low_level_init(void)	CPUとｽﾀｯｸ ﾚｼﾞｽﾀで固着化検査を実行
uint8_t watchdog_test(void)	ｳｫｯﾁﾄﾞｯｸﾞ部署を形態設定してｳｫｯﾁﾄﾞｯｸﾞ機能検査を実行
uint8_t cpu_status_test(void)	ｽﾃｰﾀｽ ﾚｼﾞｽﾀで固着化検査を実行
uint8_t sram_test(void)	SRAMで固着化と行進Bの検査を実行
uint8_t timer0_test(void)	計時器機能検査を実行
uint8_t timer1_test(void)	
uint8_t timer2_test(void)	
uint8_t interrupt_test(void)	割り込み機能を調査
uint8_t startup_test(void)	様々な等級B検査関数呼び出しと統合異常状態を提供
void qt_fmea_init(void)	FMEA検査で使用される変数とﾎﾟｰﾄを初期化
fmea_ret_t fmea_util_check_short_to_vcc(void)	SNS/SNSKﾋﾟﾝがVCCに短絡しているかどうかを調査
fmea_ret_t fmea_util_check_short_to_vss(void)	SNS/SNSKﾋﾟﾝがGNDに短絡しているかどうかを調査
fmea_ret_t fmea_util_check_short_to_pins(void)	接触ﾋﾟﾝが同じﾎﾟｰﾄの他の接触ﾋﾟﾝに短絡かどうかを調査
fmea_ret_t fmea_util_check_short_cap(void)	ﾁｬﾈﾙの採取ｺﾝﾃﾞﾝｻが短絡しているかどうかを調査
fmea_ret_t fmea_util_check_unusual_signal(void)	ﾁｬﾈﾙの信号値が指定された範囲を超えるかどうかを調査
uint8_t qt_fmea_run_diagnostics(uint8_t select_checks)	渡される入力に基づく様々なFMEA検査を実行
uint8_t qt_fmea_read_error_status(uint8_t channel_num)	特定ﾁｬﾈﾙの異常状態を返す
void qt_fmea_save_error_status_to_eeprom(void)	FMEA異常状態をEEPROMに格納
uint8_t qt_fmea_read_error_status_from_eeprom (uint8_t channel_num)	EEPROMから特定ﾁｬﾈﾙの異常状態を読み込み
void perform_initial_fmea_check(void)	初期FMEA検査実行と異常で使用者定義活動を講じる
void perform_periodic_fmea_check(void)	周期的FMEA検査実行と異常で使用者定義活動を講じる

1.5. FMEA異常状態報告

FMEA異常は幅が1ﾊﾞｲﾄで深さが最大QTouchﾁｬﾈﾙ数+1の配列(fmea_channel_status)で更新されます。各ﾊﾞｲﾄが1つのﾁｬﾈﾙに対応します。配列の最後のﾊﾞｲﾄは”全体FMEA検査状態”ﾊﾞｲﾄです。ﾁｬﾈﾙでの異常は対応する1ﾊﾞｲﾄと配列の最後のﾊﾞｲﾄに捕獲されます。使用者は異常状態を調べるのに配列全体を通して常に走査する必要はなく、”全体FMEA検査状態”ﾊﾞｲﾄを使用することができます。ﾁｬﾈﾙでの異常は図1-1.で示されるように更新されます。

図1-1. 検査結果



異常の例

- チャネル0のSNS/SNSKﾋﾟﾝがGNDに短絡している場合、第0ﾊﾞｲﾄと最終ﾊﾞｲﾄのﾋﾞｯﾄ1位置が1に設定されます。
- チャネル3のSNS/SNSKﾋﾟﾝがVCCに短絡している場合、第3ﾊﾞｲﾄと最終ﾊﾞｲﾄのﾋﾞｯﾄ0位置が1に設定されます。
- チャネル2とチャネル3のSNSﾋﾟﾝが同じﾎﾟｰﾄで短絡している場合、第2と第3のﾊﾞｲﾄと最終ﾊﾞｲﾄのﾋﾞｯﾄ2位置が1に設定されます。
- チャネル4のSNSとSNSKのﾋﾟﾝがお互いに短絡していてそれらが同じﾎﾟｰﾄの場合、第4ﾊﾞｲﾄ最終ﾊﾞｲﾄのﾋﾞｯﾄ2と3の位置が1に設定されます。
- チャネル5の採取ｺﾝﾃﾞﾝｻが短絡している場合、第5ﾊﾞｲﾄと最終ﾊﾞｲﾄのﾋﾞｯﾄ4位置が1に設定されます。
- チャネル6が指定された範囲を超える信号値を持つ場合、第6ﾊﾞｲﾄと最終ﾊﾞｲﾄのﾋﾞｯﾄ5位置が1に設定されます。

2. 試験の筋書き

本章は等級BとFMEAの検査ｺｰﾄﾞを試験する可能な方法を記述します。これらの試験の筋書きはAtmlel ATtiny88とAtmel ATmega324Aのﾃﾞﾊﾞｲｽに向けられます。

等級B検査はMCUの様々な周辺機能で機能的な調査を実行します。これらの試験は手動で異常を導入するこによって検査することができ、ﾃﾞﾊﾞｯｸﾞ動作で例のﾌﾟﾛｼﾞｪｸﾄを実行することによって実行されることが必要です。

等級Bと異なり、FMEA検査はMCUの入出力で実行されます。故に、使用者は容易に異常を模倣してFMEA検査の機能性を調べることができます。これらの検査項目は全てのﾃﾞﾊﾞｲｽに対して共通です。使用者は監視(Watch)ｳｨﾝﾄﾞｳを通して結果(fmea_channel_status配列)を見るか、またはPCでそれらを見るためにUARTを通して配列を送信することができます。

2.1. 事前必要条件

- 検査を実行するためのﾊｰﾄﾞｳｪｱ基盤
- ﾃﾞﾊﾞｯｸﾞ ツール (Atmel AVR JTAGICE mkⅡ、JTAGICE3、またはAVR ONE!)
- Atmel Studio 6でのﾃﾞﾊﾞｯｸﾞと監視ｳｨﾝﾄﾞｳの知識
- 例ﾌﾟﾛｼﾞｪｸﾄの構築とﾃﾞﾊﾞｲｽ内への書き込み

2.2. 検査項目

下表は様々な検査項目と期待する結果を記述します。使用者は検査1～5を実行するためにﾃﾞﾊﾞｯｸﾞ動作で例ﾌﾟﾛｼﾞｪｸﾄを走行することが必要です。残りの検査はﾃﾞﾊﾞｯｸﾞ動作またはﾃﾞﾊﾞｲｽ書き込み後のどちらでも実行することができます。現在の検査のために含められたどの追加ｺｰﾄﾞも次の検査項目を走行する前に取り去られるべきです。

SI番号	検査概要	検査実行手順	期待する結果
1	ｳｫｯﾁﾄﾞｯｸﾞ検査	**異常なし状態** - PD0を出力として汎用入出力を形態設定します。 - main.cでstartup_test()を呼ぶ前にﾋﾟﾝをLowにします。 - startup_test()を呼んだ後にﾋﾟﾝをHighにします。 - (Alt+F5)押下によってﾃﾞﾊﾞｯｸﾞを開始します。 - 応用を走らせます(F5)。 **異常状態模倣** - (Alt+F5)押下によってﾃﾞﾊﾞｯｸﾞを開始します。 - watchdog_test()でwhile(1)文に中断点(ﾌﾞﾚｰｸﾎﾟｲﾝﾄ)を置きます。 - 応用を走らせます(F5)。 - 監視ｳｨﾝﾄﾞｳを通してWDTCSRの値を0に変更します。これはｳｫｯﾁﾄﾞｯｸﾞ単位部がﾘｾｯﾄ発行に形態設定されないことを保証します。	**異常なし状態** PD0がHighであるべきです。 **異常状態** PD0がLowであるべきです。
2	CPUｽﾃｰﾀｽ ﾚｼﾞｽﾀ検査	**異常なし状態** - PD0とPD1を出力として汎用入出力を形態設定します。 - startup_test()を呼ぶ前に両ﾋﾟﾝをLowにします。 - startup_test()を呼んだ後にPD0ﾋﾟﾝをHighにします。 - qt_fmea_init()を呼ぶ前にPD1ﾋﾟﾝをHighにします。 - (Alt+F5)押下によってﾃﾞﾊﾞｯｸﾞを開始します。 - 応用を走らせます(F5)。 **異常状態模倣** - (Alt+F5)押下によってﾃﾞﾊﾞｯｸﾞを開始します。 - cpu_status_test()で"a=SREG;"に中断点を置きます。 - 監視ｳｨﾝﾄﾞｳを通してSREGの値を0に変更します。これは不正値読み込みに帰着し、検査異常を導きます。 - 応用を走らせます(F5)。	**異常なし状態** PD0とPD1がHighであるべきです。 **異常状態** PD0がHigh、PD1がLowであるべきです。

次頁へ続く

SI番号	検査概要	検査実行手順	期待する結果
3	SRAM検査	**異常なし状態** ”CPUステータス レジスタ検査”と同様に”**異常なし状態**”検査を実行します。 **異常状態模倣** – (Alt+F5)押下によってデバッグを開始します。 – sram_test()で”for”の何処かに中断点を置きます。 – 応用を走らせます(F5)。 – 現在検査中のSRAM位置の値を無作為な値に変更します。これはSRAM検査を失敗させます。 – 応用を走らせます(F5)。	**異常なし状態** PD0とPD1がHighであるべきです。 **異常状態** PD0がHigh、PD1がLowであるべきです。
4	計時器0検査 計時器1検査 計時器2検査 ”計時器2検査”はATmega324Aデバイスに対してだけ利用可能です。	**異常なし状態** ”CPUステータス レジスタ検査”と同様に”**異常なし状態**”検査を実行します。 **異常状態模倣** – (Alt+F5)押下によってデバッグを開始します。 – timer0_test()で”i=0”文に中断点を置きます。 – 応用を走らせます(F5)。 – 監視ウィンドウを通して”TCCRnA/B”の値を0に変更します。ATtiny88については”TCCR0A”、ATmega324Aについては”TCCR0B”。 – 応用を走らせます(F5)。 計時器1と計時器2で同様に検査を実行します。	**異常なし状態** PD0とPD1がHighであるべきです。 **異常状態** PD0がHigh、PD1がLowであるべきです。 計時器1と計時器2に対する異常状態は同じです。
5	割り込み検査	**異常なし状態** ”CPUステータス レジスタ検査”と同様に”**異常なし状態**”検査を実行します。 **異常状態模倣** – (Alt+F5)押下によってデバッグを開始します。 – interrupt_test()で”TCNT0=254”文に中断点を置きます。 – 応用を走らせます(F5)。 – 監視ウィンドウを通してTIMSK0の値を変更します。 – 応用を走らせます(F5)。	**異常なし状態** PD0とPD1がHighであるべきです。 **異常状態** PD0がHigh、PD1がLowであるべきです。

6	VCCへ短絡	**異常なし状態**
		– (Alt+F5)押下によってデバッグを開始します。
		– 応用を走らせます(F5)。
		
		異常状態模倣
		– qt_fmea_init()の後に以下のコードを追加します。
		while(1)
		fmea_util_check_short_to_vcc();
		– (Alt+F5)押下によってデバッグを開始します。
		– fmea_util_check_short_to_vcc()に中断点を置きます。
		– チャネル”N”のSNSかSNSKのどちらかをVCCに接続し、応用を走らせます(F5)。
		– 同様に他のチャネルのSNSかSNSKのピンをVCCに接続し、応用を走らせます。
			異常なし状態
			fmea_channel_status配列の全領域が$00であるべきです。
			異常状態
			fmea_channel_status[N]とfmea_channel_status配列の最終バイトのビット0が1に設定されるべきです。
			
			他のチャネル調査中、配列の各々のバイトが調査されるべきです。

次頁へ続く

SI番号	検査概要	検査実行手順	期待する結果

7	GNDへ短絡	**異常なし状態**
		- (Alt+F5)押下によってﾃﾞﾊﾞｯｸﾞを開始します。
		- 応用を走らせます(F5)。
		
		異常状態模倣
		- qt_fmea_init()の後に以下のｺｰﾄﾞを追加します。
		while(1)
		fmea_util_check_short_to_vss();
		- (Alt+F5)押下によってﾃﾞﾊﾞｯｸﾞを開始します。
		- fmea_util_check_short_to_vss()に中断点を置きます。
		- チャネル"N"のSNSかSNSKのどちらかをGNDに接続し、応用を走らせます(F5)。
		- 同様に他のチャネルのSNSかSNSKのﾋﾟﾝをGNDに接続し、応用を走らせます。
			異常なし状態
			fmea_channel_status配列の全領域が$00であるべきです。
			異常状態
			fmea_channel_status[N]とfmea_channel_status配列の最終ﾊﾞｲﾄのﾋﾞｯﾄ1が1に設定されるべきです。
			
			他のチャネル調査中、配列の各々のﾊﾞｲﾄが調査されるべきです。

8	他の接触ﾋﾟﾝへの短絡
	
	注: ﾎﾟｰﾄを渡って短絡される接触ﾋﾟﾝはこの検査で調べられません。
		異常なし状態
		- (Alt+F5)押下によってﾃﾞﾊﾞｯｸﾞを開始します。
		- 応用を走らせます(F5)。
		
		異常状態模倣
		- qt_fmea_init()の後に以下のｺｰﾄﾞを追加します。
		while(1)
		fmea_util_check_short_to_pins();
		- (Alt+F5)押下によってﾃﾞﾊﾞｯｸﾞを開始します。
		- fmea_util_check_short_to_pins()に中断点を置きます。
		- チャネル"N"の接触ﾋﾟﾝとチャネル"M"の接触ﾋﾟﾝを接続し、応用を走らせます(F5)。
		- 同様に他のチャネルを調べます。
			異常なし状態
			fmea_channel_status配列の全領域が$00であるべきです。
			異常状態
			fmea_channel_status[N]とfmea_channel_status[M]とfmea_channel_stat us配列の最終ﾊﾞｲﾄのﾋﾞｯﾄ2が1に設定されるべきです。
			他のチャネル調査中、配列の各々のﾊﾞｲﾄが調査されるべきです。

9	採取ｺﾝﾃﾞﾝｻ(Cs)短絡	**異常なし状態**
		- (Alt+F5)押下によってﾃﾞﾊﾞｯｸﾞを開始します。
		- 応用を走らせます(F5)。
		
		異常状態模倣
		- qt_fmea_init()の後に以下のｺｰﾄﾞを追加します。
		while(1)
		fmea_util_check_short_cap();
		- (Alt+F5)押下によってﾃﾞﾊﾞｯｸﾞを開始します。
		- fmea_util_check_short_to_cap()に中断点を置きます。
		- チャネルNの採取ｺﾝﾃﾞﾝｻを短絡し、応用を走らせます(F5)。
		- 同様に他のチャネルを調べます。
			異常なし状態
			fmea_channel_status配列の全領域が$00であるべきです。
			異常状態
			fmea_channel_status[N]とfmea_channel_status配列の最終ﾊﾞｲﾄのﾋﾞｯﾄ3が1に設定されるべきです。
			
			注: 感知部が内部ﾎﾟｰﾄとして形態設定される場合、ﾋﾞｯﾄ位置2も設定されます。

10	異常信号値	**異常なし状態** - FMEA_UNUSUAL_SIG_VAL_HIGHとFMEA_UNUSUAL_SIG_VAL_LOWに正しい値を設定します。 - (Alt+F5)押下によってﾃﾞﾊﾞｯｸﾞを開始します。 - 応用を走らせます(F5)。 **異常状態模倣** - 信号値が指定された範囲を超えられるようにチャネル"N"の採取ｺﾝﾃﾞﾝｻを変更しま。 - (Alt+F5)押下によってﾃﾞﾊﾞｯｸﾞを開始します。 - qt_fmea_perform_periodic_check()に中断点を置きます。 - 応用を走らせます(F5)。	**異常なし状態** fmea_channel_status配列の全領域が$00であるべきです。 **異常状態** fmea_channel_status[N]とfmea_channel_status配列の最終ﾊﾞｲﾄのﾋﾞｯﾄ4が1に設定されるべきです。 他のチャネル調査中、配列の各々のﾊﾞｲﾄが調査されるべきです。

3. 提供物

QTouchﾌﾟﾛｼﾞｪｸﾄの例がそれらに追加される安全機能と共に提供されます。例ﾌﾟﾛｼﾞｪｸﾄはAtmel ATmega324AとAtmel ATtiny88のﾃﾞﾊﾞｲｽ用に提供されます。使用者は他のAtmel tinyAVR®とmegaAVR®のﾃﾞﾊﾞｲｽに対して安全機能を許可するために”ClassB.c”ﾌｧｲﾙ内の検査関数を容易に変更することができます。”qfmea.c”と”qfmea.h”のﾌｧｲﾙはAtmel tinyAVRとmegaAVRのどのﾃﾞﾊﾞｲｽに対しても共通で、GCCとIAR™のｺﾝﾊﾟｲﾗ用に独立して提供されますが、これらは機能的に同じです。

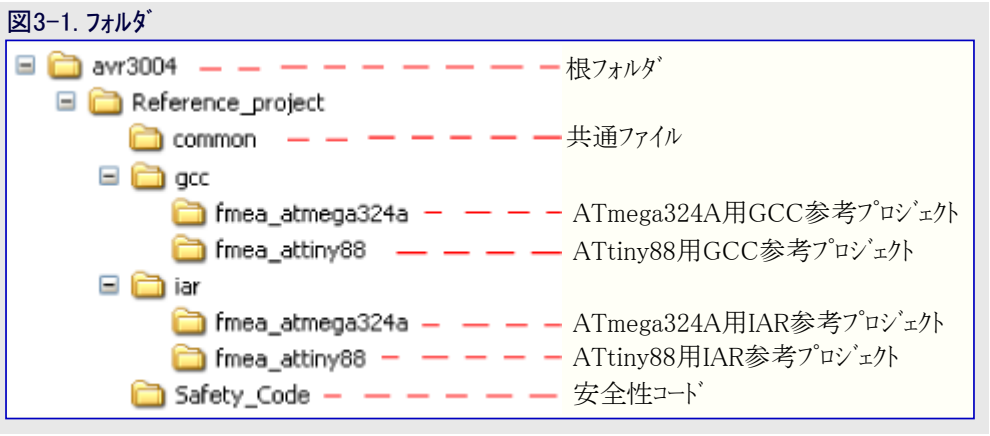
注: 検査ｺｰﾄﾞまたは参照ﾌﾟﾛｼﾞｪｸﾄは検査異常でのどんな是正ｺｰﾄﾞも持ちません。使用者は4.章で言及されるように各々の場所に是正ｺｰﾄﾞを追加することが必要です。

注: FEMAｺｰﾄﾞは更なるｱｸｾｽのためにEEPROM内にFMEA異常状態配列を格納します。QDebug規約もEEPROMを使用します。故に、同時に1つの機能だけが許可される、またはそれらが独立したEEPROMﾒﾓﾘ空間を使用するように注意が払われるべきです。

3.1. ﾌｫﾙﾀﾞとﾌｧｲﾙ

図3-1.はﾌｫﾙﾀﾞの詳細を提供します。

図3-1. ﾌｫﾙﾀﾞ



以下のﾌｧｲﾙは”Safety_Code”ﾌｫﾙﾀﾞで利用可能です。

ﾌｧｲﾙ名	説明
low_level_init.c	CPUとｽﾀｯｸ ﾚｼﾞｽﾀの検査用の実装を含みます。
ClassB.c	MCUの内部単位部/周辺機能(ｳｫｯﾁﾄﾞｯｸﾞ、割り込み、SRAM、CPUｽﾃｰﾀｽ ﾚｼﾞｽﾀ、計時器)用検査を含みます。
ClassB.h	関数原型宣言とﾏｸﾛ定義を含みます。
qfmea.c	FMEA検査の実装を含みます。
qfmea.h	FMEA検査の関数原型宣言、全体変数宣言、ﾏｸﾛ定義を含みます。

4. QTouchﾌﾟﾛｼﾞｪｸﾄへの安全機能移植

Atmel ATmega324AとAtmel ATtiny88のﾃﾞﾊﾞｲｽ用に例ﾌﾟﾛｼﾞｪｸﾄが提供されるとは言え、安全ｺｰﾄﾞは小さな努力で様々なAtmel tinyAVRやmegaAVRのﾃﾞﾊﾞｲｽに移植することができます。以下の副項はQTouch応用に等級BとFMEAのｺｰﾄﾞを移植する手順を提供します。

4.1. QTouchﾌﾟﾛｼﾞｪｸﾄ作成

- QTouchﾗｲﾌﾞﾗﾘ 5.0または最終版をﾀﾞｳﾝﾛｰﾄﾞしてｲﾝｽﾄｰﾙしてください。
- 適切なﾗｲﾌﾞﾗﾘと選択したﾃﾞﾊﾞｲｽ用の例ﾌﾟﾛｼﾞｪｸﾄを選ぶためにLibrary_Selection_Guide.xlsを参照してください。
- ｲﾝｽﾄｰﾙされた次の場所を参照してください。例ﾌﾟﾛｼﾞｪｸﾄをｱｸｾｽするには、”..¥Atmel¥Atmel_QTouch_Libraries_5.0¥Generic_QTouch_Libraries¥AVR_Tiny_Mega_XMega¥QTouch¥example_projects”
- ”Atmel Studio 6”または”IAR Embedded Workbench®”で例ﾌﾟﾛｼﾞｪｸﾄを開いてください。
- 必要条件によって”touch_config.h”でﾊﾟﾗﾒｰﾀを形態設定してください。ﾊﾟﾗﾒｰﾀのより多くの詳細については「ﾗｲﾌﾞﾗﾘ使用者の手引き」を参照してください。
- ｺｰﾄﾞを構築して書き込んでください。
- Atmel QT600とQTouch分析器の手助けで接触動作が正しい機能であることを確認してください。

4.2. 等級BとFMEAの統合

- "..¥Safety_Code"から全ファイルをプロジェクト フォルダへ複写してください。
- "qfmea.c"、"ClassB.c"、"low_level_init.c"のファイルをプロジェクトに追加してください。
- 選択したデバイスに基づいて"ClassB.c"で変更を行ってください。
- main.cで、「#include "touch_api.h"」後に右のコードを含めてください。

```
/*  now include touch api.h with the localization
#include "touch_api.h"

#ifdef _FMEA_
/* include classb and fmea header files */
#include "qfmea.h"
#include "classb.h"
#endif
```

- main.cで、「static void config_sensor(void);」後に右のコードを含めてください。

```
static void config_sensors(void);

#ifdef _FMEA_
/* perform initial fmea test */
void perform_initial_fmea_check(void);
/* perform periodic fmea test */
void perform_periodic_fmea_check(void);

/* fmea error status array */
extern uint8_t fmea_channel_status[FMEA_REPORT_SIZE];
/* fmea config structure */
extern qt_fmea_config_t qt_fmea_config;
#endif

#if defined(_ROTOR_SLIDER_)
```

- main.cで、「uint16_t burst_flag = 0u;」後に右の変数定義を含めてください。

```
uint16_t burst_flag = 0u;

#ifdef _FMEA_
/*Declare ClassB and FMEA related variables */
uint8_t test_status = 0;
uint8_t time_to_check_signal_value = 0;
#endif
```

- main.cで、「uint8_t time_to_check_signal_value = 0;」後に等級B始動検査を呼び出して調べるために右のコードを含めてください。

```
/* watchdog, timer0, timer1, interrupt, RAM test */
test_status = startup_test();
/* reset watchdog timer */
asm("wdr");

if(test_status != 1)
{
    /* Startup test failed */
    while(1);
}
```

- init_timer_isr()を呼ぶ前に右のコードを追加してください。

```
asm("wdr");

#ifdef _FMEA_

/* Assign EEPROM address to updated FMEA error status */
qt_fmea_config.eeprom_start_address = 0x100;
/* signal values which are beyond this range will be reported as error */
qt_fmea_config.hi_signal_threshold = 90;
qt_fmea_config.lo_signal_threshold = 30;

#endif

/* configure timer ISR to fire regularly */
```

- 右で示されるようにEEPROMｱﾄﾞﾚｽを割り当てた後に右のFMEA初期化ｺｰﾄﾞを追加してください。

```
qt_fmea_config.lo_signal_threshold = 30;

/* initialize fmea test */
qt_fmea_init();
/* reset watchdog timer */
asm("wdr");

#endif
```

- main.cで、qt_fmea_init()関数後に右のｺｰﾄﾞを含めることによって初期FMEA検査を走らせてください。

```
qt_fmea_init();
/* reset watchdog timer */
asm("wdr");

/* perform initial fmea tests and take corrective actions */
perform_initial_fmea_check();
/* reset watchdog timer */
asm("wdr");

#endif
```

- main.cで、startup_test()，qt_fmea_init()，qt_fmea_run_initial_test()，init_system()，config_sensors()，qt_init_sensing()，QDebug_ProcessCommands()関数呼び出し後に右のｳｫｯﾁﾄﾞｯｸﾞ ﾘｾｯﾄ ｺｰﾄﾞを追加してください。「/* 時間が重要なﾎｽﾄ応用ｺｰﾄﾞをここで実行 */」の後にも追加してください。

```
/* initialise host app, pins, watchdog, etc */
init_system();
/* reset watchdog timer */
asm("wdr");
```

- 「time_to_measure_touch = 0u;」後に右のｺｰﾄﾞを追加してください。

```
time_to_measure_touch = 0u;
/* reset watchdog timer */
asm("wdr");

#ifdef _FMEA_
time_to_check_signal_value = 1;
#endif
```

- 「/* 時間が重要でないﾎｽﾄ応用ｺｰﾄﾞをここで実行 */」の後に右のｺｰﾄﾞを追加してください。

```
/* Time Non-critical host application code goes here */
#ifdef _FMEA_
if(time_to_check_signal_value == 1)
{
    time_to_check_signal_value = 0;
    /* perform periodic fmea tests and take corrective actions */
    perform_periodic_fmea_check();
}
#endif
```

- 例ﾌﾟﾛｼﾞｪｸﾄからperform_initial_fmea_check()とperform_periodic_fmea_check()関数の定義を複写してmain.cの最後に同じ物を追加してください。
- main.cﾌｧｲﾙで以下の変数に以下の正しい値が設定されることを確実にしてください。
  qt_fmea_config.hi_signal_threshold：これよりも小さな信号値は”qt_fmea_run_diagnostics”関数で失敗として宣言されるでしょう。
  qt_fmea_config.lo_signal_threshold：これよりも大きな信号値は”qt_fmea_run_diagnostics”関数で失敗として宣言されるでしょう。
  qt_fmea_config.eeprom_start_address：EEPROMに格納されるFMEA異常状態の開始ｱﾄﾞﾚｽ。
- main.cで以下の場所に是正ｺｰﾄﾞを含めてください。
  - /* 接触ﾋﾟﾝVCCに短絡異常 */
  - /* 接触ﾋﾟﾝVSS(GND)に短絡異常 */
  - /* 接触ﾋﾟﾝ他の接触ﾋﾟﾝに短絡異常 */
  - /* Cs短絡異常 */
  - /* 異常信号異常 */

5. 参照

- Atmel AVR998:AVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗでのIEC60730等級B適合への指針
- Atmel AVR1610:XMEGAでのIEC60730等級B適合への指針
- QTAN0079:釦、摺動子、輪の感知部設計の手引き
- Atmel QTouchﾗｲﾌﾞﾗﾘ使用者の手引き

6. 改訂履歴

資料改訂	日付	注釈
42041A	2012年11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AVR3004応用記述(Rev.42041A-11/2012)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。