ミックスドシグナル MCU

MD6603　シリコンエラッタ

目次

以下に、現時点における MD6603 のエラー内容を示します。設計の際は以下の内容に十分ご注意ください。

1. フラッシュメモリコントローラの制限事項

1.1. 再プロテクト直後のイレース／プログラム（ライト）実行時の注意事項

● 内容

フラッシュメモリを再プロテクトした直後のイレースやプログラム（ライト）は、ウェイト期間が入った後に、処理が実行されます。ウェイト期間は、FMTIME レジスタの値*によって決まります。ウェイト期間が入った場合でも、フラッシュメモリ本体へのイレースやプログラム（ライト）は、正常に実行されます。再プロテクトを再度実行しなければ、以降のイレースやプログラム（ライト）時に、ウェイト期間は入りません。

* FMTIME レジスタの値を $\text{FMTIME} = \frac{\text{CLKFAST}}{1\times10^6} - 1$（たとえば、CLKFAST = 60 MHz、FMTIME = 59）で指定した場合は、ウェイト期間は 32 ms です。

● 対策方法

再プロテクト処理を実行しなければ、この不具合は発生しません。

本 LSI では、フラッシュメモリ上のプログラムであれば、プロテクトを解除することなく、フラッシュメモリのイレースやプログラム（ライト）ができます。フラッシュメモリ上のプログラムで動作する場合、プロテクトの解除処理や再プロテクト処理は不要です。

● ツールによる対策方法

ありません。

2. 8051 CPU に関する制限事項

2.1. XDATA BUS バッファに関する制限事項

● 内容

バス上の RAM にあるプログラムの実行中に、XDATA BUS バッファの機能は使用できません。

● 対策方法

ありません。

● ツールによる対策方法

ありません。

2.2. 16 ビットレジスタライトと割込みの競合

● 内容

16 ビットレジスタへのライトと割込みの受付が競合しないように、16 ビットレジスタ（SFR BUS、XDATA BUS）の上位バイトにライトする前に、INTMST.INTME ビットを 0 に設定して、割込みの受付をディスエーブルにしてください。その後、NOP 命令を 2 命令挿入後、上位バイトにライトしてください。上位バイトにライトしたあと、INTMST.INTME ビットを 1 に設定して、割込みの受付を再開してください。

● 対策方法

SFR と XDATA 空間の、上位側と下位側が同一アドレスにアサインされている 16 ビット幅レジスタに、C 言語でデータをライトする場合は、以下の対策をしてください。

(1) defines.h ファイルの先頭で、以下のマクロを定義してください（以下、「マクロ(1)」とする）。

```
#define WRITE16(RegisterName,Value)
  do{
    const uint16_t value = (Value);
    _UNSAFE_##RegisterName = (uint8_t)value;
    if(INTMST & 1){
      INTMST = INTMST & 0xfe;
      __asm__("nop");
      __asm__("nop");
      _UNSAFE_##RegisterName = (uint8_t)(value >> 8);
      INTMST = INTMST | 0x01;
    }else{
      _UNSAFE_##RegisterName = (uint8_t)(value >> 8);
    }
  }while(0)
```

(2) マクロ(1)を使用せずに 16 ビット幅レジスタにライトすることを防ぐため、defines.h ファイルのレジスタ定義で const 指定をしてください。const 指定することで、マクロ(1)を使用せずに 16 ビット幅レジスタにライトすると、コンパイルエラーが発生します。

```
const __sfr __at (0xC4) DSP0_R0  ;
__sfr __at (0xC4) _UNSAFE_DSP0_R0 ;
...
const __xdata __at(0xF788) uint8_t DSP0R8;
__xdata __at(0xF788) uint8_t _UNSAFE_DSP0R8;
```

(3) 16 ビット幅レジスタにデータをライトする場合は、以下のようにマクロ(1)を使用してください。

```
WRITE16(BUF_A0_LH,  0x1234);      //SFR
WRITE16(DSP0_C0_HL, 0xabcd);      //XDATA
```

● ツールによる対策方法

MD Studio には、新規のプロジェクト生成時にできる defines.h ファイル内に、上記の定義が記述されています。ただし、テンプレートなしで新規プロジェクトを生成すると、defines.h ファイルはありません。

2.3. RAM1 領域へのアクセス

● 内容

EPU 動作中に、CPU から 512 バイトの RAM1（0x0400～0x05FF）領域にアクセスすると、これらが競合し正常に動作しない可能性があります。

● 対策方法

EPU 動作中は、CPU から 512 バイトの RAM1（0x0400～0x05FF）領域にアクセスしないでください。

● ツールによる対策方法

ありません。

2.4. XDATA 空間へのアクセス

● 内容

EPU 動作中に、CPU から XDATA 空間のアドレス 0xC000～0xDFFF の範囲にアクセスすると、これらが競合し、正常に動作しない可能性があります。

● 対策方法

EPU 動作中は、CPU から XDATA 空間のアドレス 0xC000～0xDFFF の範囲にアクセスしないでください。

● ツールによる対策方法

ありません。

2.5. XDATA 空間の周辺レジスタへの MOVX 命令

● 内容

EPU 動作中に、CPU から XDATA 空間の周辺レジスタに、MOVX 命令を連続して実行すると、正常に動作しない場合があります。以下に連続した MOVX 命令の例を示します。

例 1：CPU から XDATA 周辺レジスタの同一アドレスに、同一データを連続してライトする

```
movx @dptr, a（ライト）
movx @dptr, a（ライト）
```

例 2：CPU から XDATA 周辺レジスタの同一アドレスに連続してライトし、リードする

```
movx @dptr, a（ライト）
movx a, @dptr（リード）
```

● 対策方法

2 つの MOVX 命令の間に、他の命令（NOP 命令など）が実行サイクルの 2 サイクル分以上挿入されていれば問題ありません。

C 言語で記述するプログラムの場合は、次の対策をしてください。EPU 動作中に、CPU から XDATA 空間の周辺レジスタに、MOVX 命令を連続して実行する場合は、CPU からの連続アクセスの間に、NOP 命令「`__asm__("nop");`」を 2 つ挿入してください。

例 1：CPU から XDATA 周辺レジスタの同一アドレスに、同一データを連続してライトする

```
movx @dptr, a（ライト）
nop
nop                    ←NOP 命令を 2 つ挿入
movx @dptr, a（ライト）
```

例 2：CPU から XDATA 周辺レジスタの同一アドレスに連続してライトし、リードする

```
movx @dptr, a（ライト）
nop
nop                    ←NOP 命令を 2 つ挿入
movx a, @dptr（リード）
```

● ツールによる対策方法

MD Studio を使用して以下の対策をする場合は、上記のようなユーザ側のプログラム対策は不要です。

(1) SDCC のコンパイルオプションに「`--peep-file md6603.peep`」が追記されています。

(2) md6603.peep ファイルは、ソースプログラムの階層にスケルトンから自動生成されます。md6603.peep ファイルが、ソースプログラムの階層に置いてあることを必ず確認してください。md6603.peep ファイルの内容は以下のとおりです。

```
replace {
  movx @dptr,a
  movx @dptr,a
} by {
  movx @dptr,a
  nop
  nop
  movx @dptr,a
}
replace {
  movx @dptr,a
  movx a,@dptr
} by {
  movx @dptr,a
  nop
  nop
  movx a,@dptr
}
```

2.6. CPU が UART レジスタにアクセスしている際の EPU 動作

2.6.1. XDATA BUS 競合による EPU アクセスの失敗

● 内容

CPU が UART レジスタにアクセスしている間に、EPU が XDATA 空間にアクセスすると、バスが競合して、EPU のアクセスが失敗する可能性があります。

● 対策方法

(1) EPU から UART にアクセスする（CPU アクセスの代行）（表 2-1 参照）
CPU の代わりに、EPU から UART レジスタにアクセスする方法です。CPU とハンドシェークすることで EPU の 1 つのスレッドを制御して、EPU が UART レジスタにアクセスできるようにします。
CPU と EPU は、SFR BUS 上の SPR や、RAM0（0x0000～0x03FF）に、ハンドシェークのためのフラグや、アクセスするアドレスデータを、リード／ライトします。

(2) CPU から UART にアクセスする（表 2-1 参照）
EPU が XDATA 空間に 2 回連続でアクセスすることで、バス競合による失敗を回避する方法です。1 回目は成功または失敗しますが、2 回目は成功します。ここで、失敗とは EPU からのアクセスが、XDATA 空間に対して発行されないことを意味します。アクセスが失敗した場合、ライト時であればラ

イトアクセスは実行されず、リード時であれば不定値を読み出します。
ハードウェアを起動するための命令（ADC トリガ、EPU 起動、UART 送受信、DSAC トリガ、PWM リトリガなど）を、周辺機能レジスタにライトする場合は、1 回目は影響がない値をライトするか、上記の対策(1)を行ってください。

(2)-1　EPU のスレッド動作中にスレッド切換えが発生する場合

- EPU が XDATA 空間にバイトアクセスする
  ワードアクセス命令 W_SA モード（同一アドレスへの 2 回連続アクセス）を使用して、バイトをリード／ライトしてください。ワードアクセス命令に関する注意事項については、2.6.2 項を参照してください。
- EPU が XDATA 空間にワードアクセスする
  16 ビット長の周辺機能レジスタへのアクセスは禁止です（実際は、レジスタの下位バイトと上位バイトに連続でアクセスすることです）。16 ビット長の周辺機能レジスタにアクセスする場合は、上記の対策(1)を行ってください。

(2)-2　EPU のスレッド動作中にスレッド切換えが発生しない場合

- 一度に動作するスレッドが 1 つの場合、または複数のスレッドが動作中にラウンドロビンでスレッド切換えが発生しない場合、バイト／ワードアクセス命令にかかわらず、XDATA 空間に 2 回連続でアクセスしてください。

表 2-1　対策一覧

対策方法	使用する命令	
	EPU のバイトアクセス命令	EPU のワードアクセス命令
(1) EPU から UART にアクセス（代行）	使用可	使用可
(2)-1 CPU から UART にアクセス（スレッド切換えが発生する場合）	ワードアクセス命令で代用（2.6.2 項参照）	使用禁止
(2)-2 CPU から UART にアクセス（スレッド切換えが発生しない場合）	バイトアクセス命令を 2 回連続で実行	ワードアクセス命令を 2 回連続で実行

● **ツールによる対策方法**

MD Studio は、本制限事項に対応しています（上記の対策(1)のみ）。MD Studio では、EPU に UART アクセスを代行させる対策プログラムをスケルトンから自動生成し、提供しています。

CPU から UART レジスタにアクセスする際は、必ず以下の関数を用いて UART レジスタにアクセスしてください。EPU に UART アクセスを代行させる対策プログラムによる RAM1 の消費量は、44 バイトです。EPU の使用方法は、以下のとおりです。

- 本対策プログラムでは、EPU のスレッド 5 を使用します。スレッド 5 の優先順位制御は、固定優先に設定してください。また、命令の追加などで、スレッド 5 を書き換えないでください。スレッド 5 は、4.1 項の対策プログラムでも使用します。
- EPU の UART アクセス用のスレッド 5 は、通常は停止しています。スレッド 5 は、CPU から関数がコールされたときだけ動作し、再度停止します（すなわち、EPU の命令フェッチを止めて、消費電力を低減します）。
- 提供する関数の引数の addr には、UART のレジスタアドレスを記述してください。UART のレジスタアドレスは、errata.h ファイルで定義される「ADDR_*UART レジスタ名*」を使用できます。
  提供する関数は以下のとおりです。

```
uint8_t UART_Reg_RD(uint16_t addr);       //UART レジスタの値を読み出し、値を返す。
void UART_Reg_WR(uint16_t addr, uint8_t data);       //UART レジスタに書き込む。
```

- エラッタファイルのインクルードについて
  UART アクセス対策や、スリープモード対策（4.1 項参照）の有効／無効は、以下の 3 つから選択できます。ただし、これら 3 つ以外の組合せ（UART アクセス対策無効、スリープモード対策有効）は選択しないでください。選択した場合、スリープモードで UART アクセス対策の関数を使用するため、ビルドエラーとなります。

  各対策プログラムの有効／無効の組合せ
  - UART アクセス対策有効、スリープモード対策有効
    - epu.easm ファイル

```
.include "errata_uart.easm"
.include "errata_sleep.easm"
```

  - UART アクセス対策有効、スリープモード対策無効
    （プロジェクト生成時は、この組合せの epu.easm ファイルが出力されています）
    - epu.easm ファイル

```
.include "errata_uart.easm"
#.include "errata_sleep.easm"
```

  - UART アクセス対策無効、スリープモード対策無効
    - epu.easm ファイル

```
#thread(5){
#    #Please set proper priority of thread 5.
#    priority:None
#    launch:None
#    pc:ERRATA_TH5_START
#    EIC:0
#}
#.include "errata_uart.easm"
#.include "errata_sleep.easm"
```

2.6.2. ワードアクセス命令で 2 回連続アクセスするときの注意事項

ワードアクセス命令を使用して同一アドレスに 2 回連続でアクセスする場合、リード（LOADX）とライト（STOREX）時で、アクセスはそれぞれ次の 2 通りの結果になります。

リードアクセス時：`LOADX.W_SA Rn, @AddrX`

- 1 回目も 2 回目も成功
- 1 回目は失敗（LOAD 値不定）、2 回目は成功

ライトアクセス時：`STOREX.W_SA @AddrX, Rn`

- 1 回目も 2 回目も成功
- 1 回目は失敗（アクセス消失）、2 回目は成功

1 回目と 2 回目の両方が成功すると、1 回目のライトで誤動作する可能性があるため、EPU からレジス

タにライトする際は、必ず以下の設定をしてください。

- 周辺機能の設定レジスタにライトする
  1 回目と 2 回目に、同じ値をライトしてください。
- 周辺機能レジスタのフラグをクリアする
  1 回目は 0 をライトし、2 回目にクリアしてください。
- ハードウェアを起動するための命令（ADC トリガ、EPU 起動、UART 送受信、DSAC トリガ、PWM リトリガなど）を、周辺機能レジスタにライトする
  1 回目は影響がない値をライトするか、2.6.1 項の対策(1)を行ってください。

2.7. XDATA 空間への CPU と EPU の同時アクセス

● **内容**

CPU と EPU が、同じタイミングで XDATA 空間にアクセスし、その後 EPU が 1 サイクル命令あけて、XDATA 空間にアクセスすると、正常に動作しない場合があります。EPU の XDATA BUS アクセス命令 STOREX（B_LO、B_HI、W、W_SA）および LOADX（B_SE、B_ZE、W、W_SA）すべてが対象です。以降、これらの EPU の XDATA BUS アクセス命令を“STOREX／LOADX 命令”と表記して説明します。CPU の XDATA BUS アクセス命令（MOVX @DPTR, A、MOVX @Rm, A、MOVX A, @DPTR、MOVX A, @Rm）のすべてが対象です。以降、これらの命令は“MOVX 命令”と表記して説明します。

以下の不具合例のように、EPU と CPU の XDATA 空間のアクセス先（表 2-2 参照）によって、EPU がハングアップしたり、1 サイクル命令あけて実行された CPU か EPU のアクセスが消失したりする場合があります。アクセスの消失とは、ライト時であればライトアクセスは実行されず、リード時であれば不定値を読み出すことを示します。

【不具合例】

(E-1)の STOREX 命令と(C-1)の MOVX 命令が、XDATA 空間の周辺レジスタに同時にアクセスすると、(C-3)の MOVX 命令の XDATA BUS アクセスが消失します。

EPU から XDATA 空間の周辺レジスタに 1 サイクル命令あけて 2 回目のアクセス

```
(E-1)  STOREX.B_LO @addr, Rm
(E-2)  MOV Rn, Rm
(E-3)  STOREX.B_LO @addr, Rm
```

このとき、CPU から XDATA 空間の周辺レジスタに 1 サイクル命令あけて 2 回目のアクセス

```
(C-1)  movx @dptr, a
(C-2)  nop
(C-3)  movx @dptr, a
```

表 2-2　命令のアクセス先と発生現象

EPU 命令 (E-1) アクセス先	EPU 命令 (E-3) アクセス先	CPU 命令 (C-1)（1 番目の命令のアクセス先）		
		周辺レジスタ	UART	RAM0/1
周辺レジスタ（UART 含む）	周辺レジスタ（UART 含む）	(C-3) 消失(1)	(E-3) 消失(3)	アクセス成功
	RAM0/1	EPU ハングアップ(2)	(E-3) 消失(3)	アクセス成功
RAM0/1	XDATA 空間（周辺レジスタ、UART、RAM0/1）	アクセス成功	(E-3) 消失(3)	アクセス成功

(1) CPU の 1 サイクル命令後の 2 回目の XDATA BUS アクセス(C-3)が消失

(2) EPU の 1 サイクル命令後の 2 回目の XDATA BUS（RAM0/1 への）アクセス(E-3)から、EPU は次の CPU の XDATA BUS アクセスが発生するまでハングアップします。
(3) EPU の 1 サイクル命令後の 2 回目の XDATA BUS アクセス(E-3)が消失します（2.6 項の現象）。
CPU の 1 回目の XDATA BUS アクセスのアクセス先が UART レジスタの場合、EPU が 2 回連続でアクセスする対策（2.6 項参照）をしても、以下の不具合例のように CPU の 1 サイクル命令後の 2 回目の XDATA BUS アクセス(C-3)が消失します。2.6 項の UART の対策とともに、本項に示す対策も実施してください。

【不具合例】
(E-1)の STOREX 命令と(C-1)の MOVX 命令が、XDATA 空間に同時にアクセスすると、(E-3)の STOREX 命令と(C-3)の MOVX 命令の XDATA BUS アクセスが消失します。

EPU から XDATA 空間に 1 サイクル命令あけて 2 回連続アクセス

```
(E-1)  STOREX.B_LO @addr, Rm
(E-2)  MOV Rn, Rm
(E-3)  STOREX.B_LO @addr, Rm
(E-4)  STOREX.B_LO @addr, Rm
```

このとき、CPU から XDATA 空間に 1 サイクル命令あけて 2 回目のアクセス

```
(C-1)  movx @dptr, a（アクセス先は UART のレジスタ）
(C-2)  nop
(C-3)  movx @dptr, a
```

● **対策方法**

(1) EPU で対策する場合

以下の不具合例 1～4 の(1)～(3)のように、STOREX／LOADX 命令、STOREX／LOADX 命令以外の 1 サイクル命令、STOREX／LOADX 命令の実行順になると本不具合が発生します。そのため、EPU が実行する STOREX／LOADX 命令は連続させるか、または 2 サイクル命令（1 サイクル命令 2 つ、または 2 サイクル命令 1 つ）以上あけてください。各命令の実行サイクル数は、MD6603 データシートの EPU の動作の項を参照してください。

【不具合例】

例 1：

```
(1)  STOREX.B_LO @addr, Rm
(2)  MOV Rn, Rm
(3)  STOREX.B_LO @addr, Rm
```

例 2：

```
(1)  STOREX.B_LO @addr, Rm
(2)  ADD Rn, Rm
(3)  LOADX.B_SE Rn, @addr
```

例 3：

```
(1)  LOADX.B_SE Rn, @addr
(2)  OR Rn, Rm
(3)  STOREX.W @addr, Rm
```

例 4：

```
(1)  LOADX.W_SA Rn, @addr
(2)  INC Rn, Rm
(3)  LOADX.B_SE Rn, @addr
```

(1)-1　固定優先の場合

STOREX／LOADX 命令、STOREX／LOADX 命令以外の 1 サイクル命令、STOREX／LOADX 命令の命令列は、同一スレッド内に記載しないでください。

以下の対策例 1 のように STOREX／LOADX 命令を連続させるか、または 2 サイクル命令以上あけてください。2 サイクル命令以上あける場合は、対策例 2 のように、動作しても影響のない命令を挿入するか、動作上の問題がなければ命令順序を入れ替えてください。

【対策例】

例 1：

```
(1) STOREX.B_LO @addr, R0
(2) LOADX.B_SE R1, @addr
```

例 2：

```
(1) STOREX.B_LO @addr, R1
(2) MOV R0, R0
(3) MOV R1, R1
(4) STOREX.B_LO @addr, R0
```

スレッド切換えがある場合は、次の注意が必要です。あるスレッドに以下の不具合例の命令列を含み、それより優先順位が下位のスレッドに STOREX／LOADX 命令を含む場合、下位スレッドの STOREX／LOADX 命令の実行直後に上位のスレッドが実行されると、WAIT 命令解除のタイミングによっては、不具合の原因となる命令列が発生する場合があります。動作タイミングを確認し、必要に応じて EVTWAIT／TIMWAIT 命令と STOREX／LOADX 命令の間を 2 サイクル命令以上あけてください。

【不具合例】

例 1：

```
(1) EVTWAIT #evt
(2) STOREX.B_LO @addr, Rm
```

例 2：

```
(1) EVTWAIT #evt
(2) MOV Rn, Rm
(3) STOREX.B_LO @addr, Rm
```

例 3：

```
(1) TIMWAIT #time（#time が 0 の場合も不具合になる）
(2) STOREX.B_LO @addr, Rm
```

例 4：

```
(1) TIMWAIT #time（#time が 0 の場合も不具合になる）
(2) MOV Rn, Rm
(3) STOREX.B_LO @addr, Rm
```

【対策例】

例 1：

```
(1) EVTWAIT #evt
(2) MOV R0, R0
(3) MOV R0, R0
(4) STOREX.B_LO @addr, R1
```

例 2：

```
(1) TIMWAIT #time
(2) MOV R1, R1
(3) MOV R1, R1
(4) STOREX.B_LO @addr, R0
```

(1)-2　ラウンドロビンの場合

複数のスレッドで STOREX／LOADX 命令を使用する場合、ラウンドロビンは使用しないでください。

(2) CPU で対策する場合

CPU に MOVX 命令、1 サイクル命令、MOVX 命令の実行順がある場合は、MOVX 命令と MOVX 命令の間を 2 サイクル命令以上あけてください。EPU で対策した場合は、CPU の対策は不要です。各命令の実行サイクル数は、MD6603 データシートの 8051 CPU の命令コードマップの項を参照してください。ただし、EPU 側に STOREX／LOADX 命令、STOREX／LOADX 命令以外の 1 サイクル命令、STOREX／LOADX 命令（アクセス先が RAM0/1）の命令実行順があり、表 2-2 の注釈(2)のように EPU がハングアップする恐れがある場合は、必ず EPU で対策をしてください。

【対策例】

```
movx @dptr,a
nop            （NOP 命令を 2 つ挿入するなど、MOVX 命令の間に 2 サイクル以上の命令を挿入）
nop
movx @dptr,a
```

● **ツールによる対策方法**

MD Studio は、CPU の 2 回目のアクセス消失の対策として、md6603.peep ファイルを提供しています。このファイルは、EPU の 2 回目のアクセス先が RAM0/1 の場合の EPU のハングアップには対応していませんので、上記の EPU の対策を実施してください。

(1) SDCC のコンパイルオプションに「--peep-file md6603.peep」が追記されています。
(2) md6603.peep ファイルは、ソースプログラムの階層にスケルトンから自動生成されます。
md6603.peep ファイルが、ソースプログラムの階層に置いてあることを必ず確認してください。
この .peep ファイルは、CPU の MOVX 命令と MOVX 命令の間が 1 サイクル命令以下の場合、MOVX 命令の間を 2 サイクル命令あける置換を実施します。以下に置換例を示します。

【置換例】

置換前：

```
movx @dptr,a
clr a
movx @dptr,a
```

置換後：

```
movx @dptr,a
nop
clr a
movx @dptr,a
```

3. POC（PWM Output Controller）に関する制限事項

3.1. クリアウェイト機能使用時の動作制限

● **内容**

制御遅延付加機能を使用し、かつクリアウェイト機能をイネーブルにした場合（POCDTCn レジスタを 0x81、0x82、0x83 のいずれかの値に設定した場合）、POC 制御をクリアする際に以下の制限があります。

(1) PWM による自動リリース機能は使用できません。必ず、POCCRn.ARELEN = 0 に設定してください。
(2) POCSTS.PWMnF ビットに 1 を書き込んで POC 制御をクリアする場合は、POCSTS.PWMnF ビットに書き込む前に POCSTS レジスタを読み出し、クリアを行うビットが 1（制御中）であることを必ず確認してください。この操作を行わない場合、想定しない動作が発生する可能性があります。

● **対策方法**

ありません。

● **ツールによる対策方法**

ありません。

4. システムコントローラ（SYSC）に関する制限事項

4.1. スリープモード復帰時に関する注意事項

● 内容

本 LSI が以下の動作状態になると、CPU はスリープモードから復帰しません。

(1) CPU がスリープモードに入るための命令を、レジスタに書き込む前後に、優先度が高い割込み信号を受け取った場合
(2) 優先度が低い割込みの処理中に、CPU がスリープモードに入るための命令をレジスタに書き込んだ場合

また、以下の動作状態では、CPU が優先度の高い割込みを受け取った場合だけ、CPU はスリープモードから復帰します。優先度が低い割込みの場合は、CPU はスリープモードから復帰しません。

(1) CPU がスリープモードに入るための命令をレジスタに書き込む前後に、優先度が低い割込み信号を受け取った場合

(2) 優先度が低い割込みの処理中に、CPU がスリープモードに入るための命令をレジスタに書き込んだ場合

通常の状態（割込み処理中ではない）で、CPU がスリープモードに入るための命令をレジスタに書き込む前後に、割込み信号を受け取らなかった場合、優先度にかかわらず、CPU は割込み信号でスリープモードから復帰します。

ここで、優先度が高い割込みとは、INTC の INTLVLn レジスタに対応するビットが、1 に設定されている割込みを意味します。また、優先度が低い割込みとは、INTLVLn レジスタに対応するビットが、0 に設定されている割込みを意味します。

● 対策方法

スリープモードを使用する場合は、以下の処理をしてください。

(1) スリープモードに入る前の対策
コンパレータや GPIO などに入力される外部信号による割込みなど、CPU が発生のタイミングを制御できない割込みは、スリープモードに入る前にディスエーブルにしてください。発生のタイミングを CPU で制御できる割込みは、スリープモードに入るための命令を実行中に、割込みが発生しないようにしてください。
(2) スリープモードからの復帰要因の対策
スリープモードからの復帰要因になる割込みは、CPU で発生のタイミングを制御できるものを使用してください。割込みのタイミングを制御できないものを復帰要因として使用する場合は、CPU がスリープモードに入った後に、EPU で復帰要因の割込みをイネーブルにしてください。EPU は INTC のレジスタにアクセスできないので、割込みを発行するそれぞれの周辺モジュールの割込み機能のイネーブル／ディスエーブルを設定してください。スリープモードからの復帰要因の割込みは、優先度を高くすることを推奨します。

● ツールによる対策方法

Goto_Sleep 関数を実行するための初期化関数「`void Sleep_Init(void)`」と、スリープモードに入るための関数「`void Goto_Sleep(void)`」を、MD Studio のスケルトンで提供しています。これらの関数を使用する場合は、epu.easm ファイル（EPU アセンブリファイル）に、errata_uart.easm ファイルと errata_sleep.easm ファイルの 2 つをインクルードしてください。

errata_sleep.easm ファイルがスケルトンから出力されると、ファイル内の include 文は、「`#include errata_sleep.easm`」と記述されています。スリープモードを使用する場合は、上記の include 文の「#」を削除して、この記述を有効にしてください。epu.easm ファイルに、errata_sleep.easm ファイルをインクルードすると、本 LSI は、main 関数内で Sleep_Init() 関数を自動で実行します。このとき、RAM0

の領域を 74 バイト消費します。ユーザが Goto_Sleep() 関数をコールするだけで、本 LSI はスリープモードに入り、かつ割込みで正しくスリープモードから復帰できます。

Goto_Sleep() 関数は、以下の処理を実行します。

(1) すべての周辺モジュールの割込みイネーブル状態を取得し、RAM0 内に保持します。
(2) すべての周辺モジュールの割込みをディスエーブルにします。
(3) EPU のスレッド 5 を起動します。（2.6.1 項の対策プログラムも、スレッド 5 を使用します）
(4) CPU がスリープモードに入ります。
(5) スレッド 5 が、RAM0 内に取得しておいた周辺モジュールのイネーブルだった割込みを再度イネーブルにします。その後、スレッド 5 はウェイトします。
(6) スリープモードから復帰後、main 関数に戻ります。

errata_sleep.easm ファイルをインクルードした場合、スリープモード対策は、RAM1 の 100 バイトを使用します。EPU のスレッド 5 は、対策プログラム専用のスレッドとして使用されます。スレッド 5 の優先順位制御は、固定優先に設定してください。それ以外の設定は変更しないでください。

WDT モジュールをインターバルタイマに使用している場合、Goto_Sleep() 関数は、割込みを無効にするために WDT のカウントを停止させます。このため、Goto_Sleep() 関数を実行すると、WDT のインターバルタイマの周期は、本来の設定よりも長くなります。

ただし、割込みシーケンスの中で Goto_Sleep() 関数を実行しないでください。CPU がスリープモードから復帰できなくなる可能性があります。本対策プログラムを適用する場合、2.6.1 項で定義される関数「`UART_Reg_RD(uint16_t addr)`」と、「`UART_Reg_WR(uint16_t addr, uint8_t data)`」を必ず使用して、UART レジスタにアクセスしてください。

5. 12 ビット SAR ADC に関する制限事項

5.1. サンプリングサイクル設定に関する制限事項

● **内容**

ADNSMPmn レジスタのサンプリングサイクルを設定する SHTIME ビットにおいて、以下のサンプリングサイクルでは正しい変換値を得られません。

5、37、69、101、133、165、197、229

● **対策方法**

ADNSMPmn レジスタのサンプリングサイクルを設定する SHTIME ビットにおいて、以下のサンプリングサイクルの指定を禁止します。

5、37、69、101、133、165、197、229

● **ツールによる対策方法**

ありません。

注意書き

- 本書に記載している製品（以下、「本製品」という）のデータ、図、表、およびその他の情報（以下、「本情報」という）は、本書発行時点のものです。本情報は、改良などで予告なく変更することがあります。本製品を使用する際は、本情報が最新であることを弊社販売窓口に確認してください。
- 本製品は、一般電子機器（家電製品、事務機器、通信端末機器、計測機器など）の部品に使用されることを意図しております。本製品を使用する際は、納入仕様書に署名または記名押印のうえ、返却をお願いします。高い信頼性が要求される装置（輸送機器とその制御装置、交通信号制御装置、防災装置、防犯装置、各種安全装置など）に本製品を使用することを検討する際は、必ず事前にその使用の適否について弊社販売窓口へ相談いただき、納入仕様書に署名または記名押印のうえ、返却をお願いします。本製品は、極めて高い信頼性が要求される機器または装置（航空宇宙機器、原子力制御、その故障や誤動作が生命や人体に危害を及ぼす恐れのある医療機器（日本における法令でクラスⅢ以上）など）（以下「特定用途」という）に使用されることは意図されておりません。特定用途に本製品を使用したことでお客様または第三者に生じた損害などに関して、弊社は一切その責任を負いません。
- 本製品を使用するにあたり、本製品に他の製品や部材を組み合わせる際、あるいはこれらの製品に物理的、化学的、その他の何らかの加工や処理を施す際は、使用者の責任においてそのリスクを必ず検討したうえで行ってください。
- 弊社は、品質や信頼性の向上に努めていますが、半導体製品は、ある確率で欠陥や故障が発生することは避けられません。本製品が故障し、その結果として人身事故、火災事故、社会的な損害などが発生しないように、故障発生率やディレーティングなどを考慮したうえで、使用者の責任において、本製品が使用される装置やシステム上で、十分な安全設計および確認を含む予防措置を必ず行ってください。ディレーティングについては、納入仕様書および弊社ホームページを参照してください。
- 本製品は、耐放射線設計をしておりません。
- 本書に記載している回路定数、動作例、回路例、パターンレイアウト例、設計例、推奨例、本書に記載しているすべての情報、およびこれらに基づく評価結果などは、使用上の参考として示したものです。
- 本情報に起因する使用者または第三者のいかなる損害、および使用者または第三者の知的財産権を含む財産権とその他一切の権利の侵害問題について、弊社は一切その責任を負いません。
- 本情報を、文書による弊社の承諾なしに転記や複製をすることを禁じます。
- 本情報について、弊社の所有する知的財産権およびその他の権利の実施、使用または利用を許諾するものではありません。
- 使用者と弊社との間で別途文書による合意がない限り、弊社は、本製品の品質（商品性、および特定目的または特別環境に対する適合性を含む）ならびに本情報（正確性、有用性、および信頼性を含む）について、明示的か黙示的かを問わず、いかなる保証もしておりません。
- 本製品を使用する際は、特定の物質の含有や使用を規制する RoHS 指令など、適用される可能性がある環境関連法令を十分に調査したうえで、当該法令に適合するように使用してください。
- 本製品および本情報を、大量破壊兵器の開発を含む軍事用途やその他軍事利用の目的で使用しないでください。また、本製品および本情報を輸出または非居住者などに提供する際は、「米国輸出管理規則」や「外国為替及び外国貿易法」など、各国で適用される輸出管理法令などを遵守してください。
- 弊社物流網以外における本製品の落下などの輸送中のトラブルについて、弊社は一切その責任を負いません。
- 本書は、正確を期すために慎重に製作したものですが、本書に誤りがないことを保証するものではありません。万一、本情報の誤りや欠落に起因して、使用者に損害が生じた場合においても、弊社は一切その責任を負いません。
- 本製品を使用する際の一般的な使用上の注意は弊社ホームページを、特に注意する内容は納入仕様書を参照してください。
- 本書で使用されている個々の商標、商号に関する権利は、弊社を含むその他の原権利者に帰属します。

DSGN-CJZ-16003

変更履歴

Revision	日付	変更内容
1.0	2017/12/27	新規作成
1.1	2018/04/06	項番号修正（2.6.1 項追加）
1.2	2021/05/07	● 2.5 項　対策方法 誤：2 つの MOVX 命令の間に、他の命令（NOP 命令など）が挿入されていれば問題ありません。 正：2 つの MOVX 命令の間に、他の命令（NOP 命令など）が実行サイクルの 2 サイクル分以上挿入されていれば問題ありません。 ● 2.6.1 項　ツールによる対策方法 誤：スレッド 5 と他のスレッドとのラウンドロビン設定は可能です。 正：スレッド 5 の優先順位制御は、固定優先に設定してください。 ● 2.7 項　追加 ● 4.1 項　ツールによる対策方法 誤：スレッドの優先順位以外の設定は変更しないでください。 正：スレッド 5 の優先順位制御は、固定優先に設定してください。それ以外の設定は変更しないでください。