AN1982

tinyAVR 0と1系及びmegaAVR 0系での割り込みシステム

要点

- 割り込み制御器概要
- 割り込み優先権構成設定
- ﾍﾞｸﾀ表構成設定
- ｺｰﾄﾞ例

序説

ﾏｲｸﾛ ｺﾝﾄﾛｰﾗ(MCU)は通信、ﾀｲﾐﾝｸﾞと波形生成のような特化した作業を実行するように設計された広範囲なﾊｰﾄﾞｳｪｱ単位部(周辺機能)を含みます。一般的に、周辺機能はその状態、動作完了、ﾃﾞｰﾀ有効性、または周辺機能が新しい命令を受け取る準備が整ったことを示すかなりの信号を持ちます。それは関連する更新に対して各周辺機能を調べるために中央処理部(CPU)に上げます。CPUはその後に各単位部からの要求を処理しなければなりません。増加する周辺機能単位部数で、CPUは全ての単位部を調べて処理するのに増々より時間を費やします。このCPU負荷はより長い応答時間とより高い電力消費を引き起こします。伝統的に、MCUは周辺装置を監視して処理するのに、主にﾎﾟｰﾘﾝｸﾞと割り込みの2つの方法を使います。

ﾎﾟｰﾘﾝｸﾞは監視される周辺機能によって更新された状態ﾋﾞｯﾄを手動で読んで調査することを意味します。これは各種周辺機能を調べるための頻度と順番を決めるのに高度な自由度を設計者に与えます。応用が特定の状態を待つだけの単純使用の場合では、この状態ﾋﾞｯﾄを調べてそれが発生した時に直ちに処理する、非常にきつい繰り返しを作ることができます。例えこの手法が非常に高速な応答を許すと言えども、ﾎﾟｰﾘﾝｸﾞ法を用いるいくつかの主な欠点があります。1つ目は調べる状態ﾋﾞｯﾄ数が増えると応答性が落ちます。2つ目は全ての状態ﾋﾞｯﾄを検査するｺｰﾄﾞの実行中にCPUが活動動作での走行を必要とし、これは電力消費を増します。

応用で必要とされる周辺機能と状態ﾋﾞｯﾄの数が増すため、割り込みｼｽﾃﾑはより良い応答性を提供します。割り込みｼｽﾃﾑは周辺機能がCPU実行に割り込みための要求をおくることができる仕組みです。周辺機能は処理が必要とされる時にCPUを促すため、CPUに対して活発に状態ﾋﾞｯﾄをﾎﾟｰﾘﾝｸﾞする必要がありません。けれども、周辺機能が他の周辺機能の要求に関わらず、常にCPUからの処理を要求することができると、これは割り込み要求衝突の危険を引き起こします。従って、割り込み制御器は保留中の割り込みがCPUによって処理される順番を決めるのに使われます。伝統的に、tinyAVR®とmegaAVR®のﾃﾞﾊﾞｲｽに対する割り込み処理は割り込みﾍﾞｸﾀ ｱﾄﾞﾚｽに基づいて予め定義される優先権を使います。XMEGA® MCU系統は割り込みを3つの優先ﾚﾍﾞﾙに割り当てることを使用者に許す設定可能な多段割り込み制御器(PMIC:Programmable Multilevel Interrupt Controller)を用いて優先待ち行列を仕立てることが可能です。割り込み処理技法はtinyAVR 0と1系及びmegaAVR 0系でもっと構成設定可能になりました。この応用記述は予め定義される優先権と多段制御部を使う新しい割り込み制御器の機能を詳細に記述します。

目次

1. 関連ﾃﾞﾊﾞｲｽ

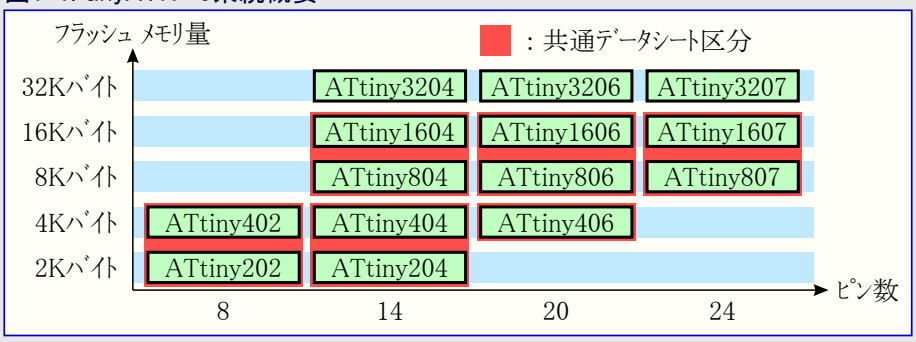
本章はこの資料に関連するﾃﾞﾊﾞｲｽを一覧にします。

1.1. tinyAVR® 0系統

下図はﾋﾟﾝ数の変種とﾒﾓﾘ量を展開してtinyAVR® 0系統ﾃﾞﾊﾞｲｽを示します。

- これらのﾃﾞﾊﾞｲｽが完全にﾋﾟﾝと機能が互換のため、垂直方向移植はｺｰﾄﾞ変更なしで可能です。
- 左への水平方向移植はﾋﾟﾝ数、従って利用可能な機能を減らします。

図1-1. tinyAVR® 0系統概要



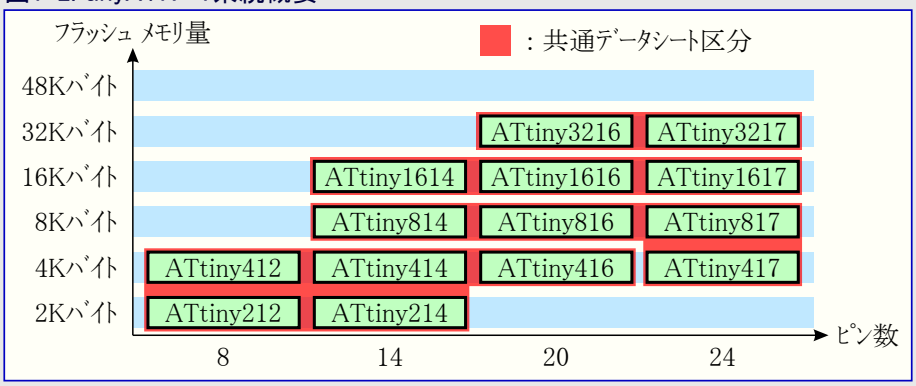
異なるﾌﾗｯｼｭ ﾒﾓﾘ量を持つﾃﾞﾊﾞｲｽは一般的に異なるSRAMとEEPROMの量を持ちます。

1.2. tinyAVR® 1系統

下図はﾋﾟﾝ配置変種とﾒﾓﾘ量を展開してtinyAVR® 1系統ﾃﾞﾊﾞｲｽを示します。

- これらのﾃﾞﾊﾞｲｽがﾋﾟﾝ互換で同じまたはより多くの機能を提供するため、垂直上方向移植はｺｰﾄﾞ変更なしに可能です。下方向移植はより少ない利用可能ないくつかの周辺機能の実体のためにｺｰﾄﾞ変更が必要かもしれません。
- 左への水平方向移植はﾋﾟﾝ数、従って利用可能な機能を減らします。

図1-2. tinyAVR® 1系統概要



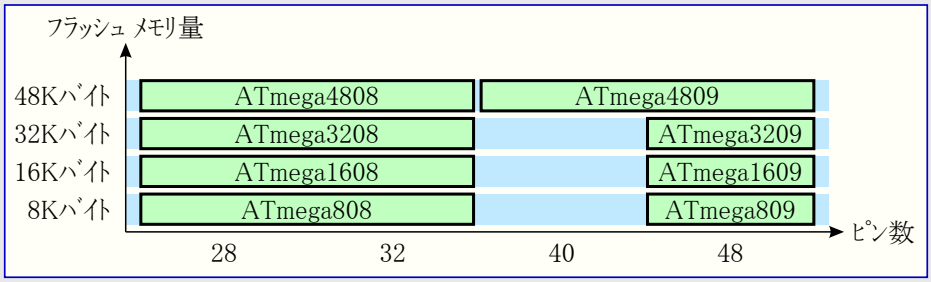
異なるﾌﾗｯｼｭ ﾒﾓﾘ量を持つﾃﾞﾊﾞｲｽは一般的に異なるSRAMとEEPROMの量を持ちます。

1.3. megaAVR® 0系統

下図はﾋﾟﾝ配置変種とﾒﾓﾘ量を展開してmegaAVR® 0系統ﾃﾞﾊﾞｲｽを示します。

- これらのﾃﾞﾊﾞｲｽが完全にﾋﾟﾝと機能が互換のため、垂直方向移植はｺｰﾄﾞ変更なしで可能です。
- 左への水平方向移植はﾋﾟﾝ数、従って利用可能な機能を減らします。

図1-3. megaAVR® 0系統概要



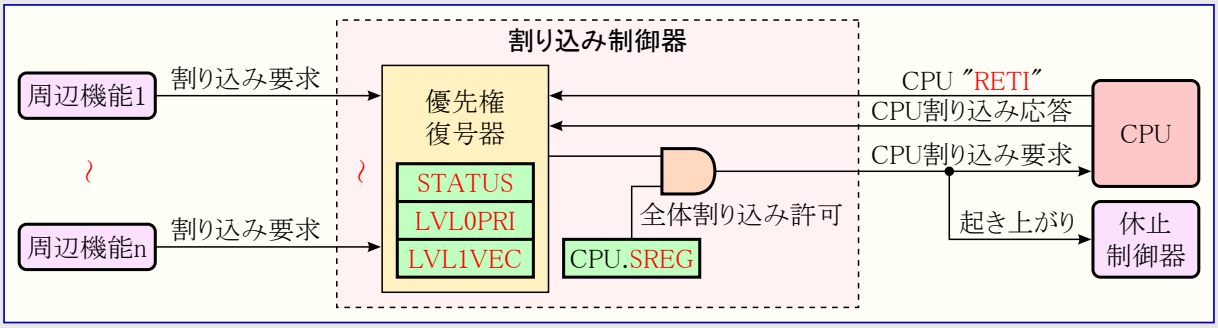
異なるﾌﾗｯｼｭ ﾒﾓﾘ量を持つﾃﾞﾊﾞｲｽは一般的に異なるSRAMとEEPROMの量を持ちます。

2. 割り込み制御器概要

tinyAVR 0と1系統及びmegaAVR 0系統の割り込み制御器は専用の割り込みﾍﾞｸﾀを持つ多数の個別割り込み要求(IRQ)に対する支援を持ちます。これは割K身が起動される時に遅延を減らすのに役立ちます。

割り込みが許可されて割り込みが起こると、CPUは割り込み制御器からIRQを受け取ります。割り込みの優先ﾚﾍﾞﾙと何れかの進行中の割り込み処理ﾙｰﾁﾝ(ISR:Interrupt Service Routine)の優先ﾚﾍﾞﾙに基づいて、IRQは応答され(てそのISRが実行され)るか、または保留中を保たれるかのどちらかです。割り込み制御器相互作用の図的表現については**図2-1.**をご覧ください。

図13-1. CPUINT構成図



割り込み制御器は割り込み優先ﾚﾍﾞﾙとﾍﾞｸﾀ ｱﾄﾞﾚｽの組み合わせによって保留中のISRに対して実行の順番を決めます。固有のﾍﾞｸﾀ ｱﾄﾞﾚｽと各種優先動作がCPUINTﾚｼﾞｽﾀから以下の利用可能な優先ﾚﾍﾞﾙで構成設定可能です。

1. 遮蔽不可割り込み
2. ﾚﾍﾞﾙ1優先割り込み
3. ﾚﾍﾞﾙ0優先割り込み

割り込みﾍﾞｸﾀ割り当てについてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄを参照してください。

割り込み要求ﾌﾗｸﾞ

唯一の例外としての遮蔽不可割り込みを除き、IRQを生成するために全体割り込み許可(I)ﾌﾗｸﾞが設定(1)されなければなりません。これらの情報は「**遮蔽不可割り込み**」下で見つけることができます。

割り込み要求ﾌﾗｸﾞの大多数は対応するISRで解除(0)されなければなりません。これはいくつかの割り込み元が割り込みﾍﾞｸﾀを共用するためです。他のﾌﾗｸﾞはISRに入る時、またはﾃﾞｰﾀ ﾚｼﾞｽﾀを読む時にﾊｰﾄﾞｳｪｱによって解除(0)されます。詳細な情報についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄで各周辺機能の章を参照してください。

割り込み遅延

CPUの割り込み遅延は以下の活動に対して必要とされるｸﾛｯｸ周期数によって決められます。

1. 進行中の命令の完了
   - 命令時間に応じて1～3ｸﾛｯｸ周期
2. ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀをｽﾀｯｸに押し込み(保存)
   - 2ｸﾛｯｸ周期
3. 割り込みﾍﾞｸﾀ表に格納された無条件分岐命令を実行
   - ﾌﾗｯｼｭ ﾒﾓﾘの大きさに応じて2～3ｸﾛｯｸ周期、**表2-1.**をご覧ください。

表2-1. 割り込み無条件分岐命令

ﾌﾗｯｼｭ ﾒﾓﾘ容量	無条件分岐命令	無条件分岐実行時間
≦8Kﾊﾞｲﾄ	RJMP	2ｸﾛｯｸ周期
＞8Kﾊﾞｲﾄ	JMP	3ｸﾛｯｸ周期

総割り込み遅延は上で一覧にしたﾊﾟﾗﾒｰﾀに応じて5～8ｸﾛｯｸ周期です。より多くの情報については「AVR命令一式手引書」をご覧ください。

ﾃﾞﾊﾞｲｽが休止動作形態の場合、応答時間は使った休止動作形態からのCPU始動時間に加えて、5ｸﾛｯｸ周期増加します。

3. 静的優先権割り込み

tinyAVR 0と1系統及びmegaAVR 0系統は既定によって静的優先権を持つ割り込みﾍﾞｸﾀ表を使います。これは他の旧tinyAVR®とmegaAVR®と同じで、殆どの標準的な応用に対してこれは好まれる動作の方法です。

静的優先権割り込みの仕組みは優先ﾚﾍﾞﾙ0割り込みにだけ適用できます。より高いﾚﾍﾞﾙの割り込みの情報は「高優先権割り込みﾍﾞｸﾀ」と「遮蔽不可割り込み」の章で見つかります。

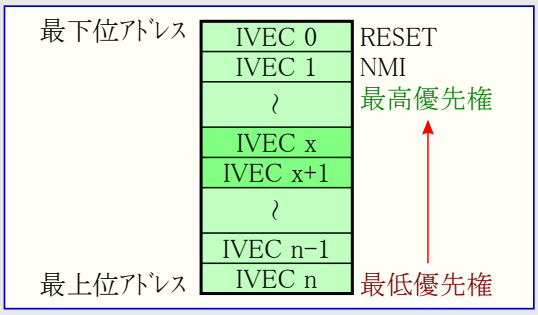
3.1. 操作

静的優先割り込みﾍﾞｸﾀ表使用時、最低割り込みｱﾄﾞﾚｽが最高優先権を持ちます。これは2つ以上の割り込みが保留中の時の実行順序が昇順で割り込みﾍﾞｸﾀｱﾄﾞﾚｽによって設定されることを意味します。図表的表現については図3-1.をご覧ください。

CPUがISRを実行している間、新しいISRは走行しているﾙｰﾁﾝが完了するまで保留中を維持され、(実行される)その時は最低ﾍﾞｸﾀ ｱﾄﾞﾚｽ(最高優先権)を持つ割り込みが先に実行されます。

静的優先割り込み機構では、割り込み優先ﾚﾍﾞﾙ0(CPUINT.LVL0PRI)ﾚｼﾞｽﾀが最高優先割り込みに対する開始点を定義し、故にLVL0PRIに臨む割り込みﾍﾞｸﾀ ｱﾄﾞﾚｽを書くことによってﾍﾞｸﾀ表内の優先権を静的にずらすことが可能です。このﾚｼﾞｽﾀの既定値は$00で、故にﾘｾｯﾄ後、最低ｱﾄﾞﾚｽ割り込みﾍﾞｸﾀが最高優先権を持ちます。

図3-1. 静的優先機構を用いる割り込み優先権



3.2. 例

静的優先割り込み機構はtinyAVR 0と1系統及びmegaAVR 0系統のﾃﾞﾊﾞｲｽの割り込み制御器に対する既定設定で、故にCPUINTﾚｼﾞｽﾀは始動でこれ用に構成設定されます。以下のｺｰﾄﾞ断片は静的優先割り込み機構を構成設定する方法の例を示します。

静的優先割り込み機構構成設定

```
// io.hはﾚｼﾞｽﾀとﾍﾞｸﾀの定義を持つﾃﾞﾊﾞｲｽ ﾍｯﾀﾞﾌｧｲﾙを含みます。
#include <avr/io.h>
// interrupt.hはISR関連内容を含みます。
#include <avr/interrupt.h>

void static_priority_interrupt_example(void) {
    // 必要なら、ﾗｳﾝﾄﾞﾛﾋﾞﾝ計画許可ﾋﾞｯﾄを解除(0)
    CPUINT.CTRLA &= ~CPUINT_LVL0RR_bm;

    // 割り込み優先権をずらす方法の例
    CPUINT.LVL0PRI = PORTA_PORT_vect_num;

    // 全体割り込み許可
    sei();
}

ISR(PORTA_PORT_vect){
    // この割り込みは最高優先権を持ちます。
}
```

4. ﾗｳﾝﾄﾞﾛﾋﾞﾝ優先機構

静的優先割り込み機構が使われるいくつかの場合で、割り込み枯渇が発生するかもしれません。これは応用があまりにも頻繁に起動されるより高い優先権(より低いﾍﾞｸﾀ ｱﾄﾞﾚｽ)の割り込みを持つ時に、より低い優先権の割り込みに対する処理不足を引き起こします。ﾗｳﾝﾄﾞﾛﾋﾞﾝ優先機構はこの割り込み枯渇を防ぐことが狙いです。

ﾗｳﾝﾄﾞﾛﾋﾞﾝ優先機構は優先ﾚﾍﾞﾙ0割り込みに対して有効です。より高いﾚﾍﾞﾙの割り込みの記述は「高優先権割り込みﾍﾞｸﾀ」と「遮蔽不可割り込み」の章下で見つかります。

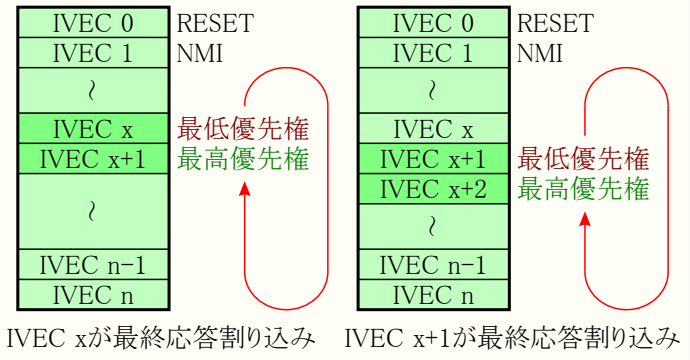
4.1. 操作

ﾗｳﾝﾄﾞﾛﾋﾞﾝ優先機構使用時、割り込み制御器は最後に応答された割り込みのｱﾄﾞﾚｽを格納し、複数の割り込み要求が保留中の場合に、格納されたｱﾄﾞﾚｽは次の割り込みが処理されることになっている時に最低優先権を持ちます。結果として、より高い次の割り込みﾍﾞｸﾀ ｱﾄﾞﾚｽが最高優先権を割り当てられます。この仕組みの図表的表現については図4-1.をご覧ください。

ﾗｳﾝﾄﾞﾛﾋﾞﾝ優先機構は制御A(CPUINT.CTRLA)ﾚｼﾞｽﾀのﾗｳﾝﾄﾞﾛﾋﾞﾝ優先権許可(LVL0RR)ﾋﾞｯﾄに'1'を書くことによって許可されます。割り込み優先ﾚﾍﾞﾙ0(CPUINT.LVL0PRI)ﾚｼﾞｽﾀは最後に応答された割り込みのｱﾄﾞﾚｽを含みます。これは割り込み制御器がどの割り込みを次に実行するかを決める時に最低優先権割り込みになります。

CPUINT.LVL0PRIの値を変更することにより、CPUに対してその割り込みが最低で次が最高の優先権を持つ割り込みを変更することが可能です。このﾚｼﾞｽﾀの既定値は$00で、故にﾘｾｯﾄ後の最低位置割り込みﾍﾞｸﾀは最高優先権を持ちます。

図4-1. ﾗｳﾝﾄﾞﾛﾋﾞﾝ割り込み機構を用いる割り込み優先権



4.2. 例

以下のｺｰﾄﾞ断片はﾗｳﾝﾄﾞﾛﾋﾞﾝ計画機構を構成設定する方法の例を示します。

ﾗｳﾝﾄﾞﾛﾋﾞﾝ割り込み計画機構構成設定

```
// io.hはﾚｼﾞｽﾀとﾍﾞｸﾀの定義を持つﾃﾞﾊﾞｲｽ ﾍｯﾀﾞﾌｧｲﾙを含みます。
#include <avr/io.h>
// interrupt.hはISR関連内容を含みます。
#include <avr/interrupt.h>

void round_robin_interrupt_scheduling_example(void) {
    // ﾗｳﾝﾄﾞﾛﾋﾞﾝ計画許可ﾋﾞｯﾄを設定(1)
    CPUINT.CTRLA |= CPUINT_LVL0RR_bm;

    // 初期割り込み優先権をずらす方法の例
    CPUINT.LVL0PRI = PORTA_PORT_vect_num;

    // 全体割り込み許可
    sei();
}

ISR(PORTA_PORT_vect) {
    // この割り込みは初期に最低優先権を持ちますが、割り込みが実行される毎に割り込み制御器によってLVL0PRIが更新
    // されるためにそれが変わります。
}
```

Atmel | START例

Atmel | STARTを用いるﾗｳﾝﾄﾞﾛﾋﾞﾝ割り込み計画用の使用事例も利用可能です。より多くの情報については「Atmel | STARTからのｿｰｽ ｺｰﾄﾞ取得」章をご覧ください。

5. 高優先権割り込みﾍﾞｸﾀ

既に定義されたものの上でより高い割り込み優先権が必要とされる時に、1つの割り込みﾍﾞｸﾀを優先ﾚﾍﾞﾙ1に割り当てることができます。その割り込みは全てのﾚﾍﾞﾙ0割り込みを超える優先権を持ち、進行中のﾚﾍﾞﾙ0割り込み処理部に割り込み能力を持ちます。

全ての割り込み元は遮蔽不可割り込み(NMI)を除き、既定によってﾚﾍﾞﾙ0優先権を割り当てられます。NMIはﾚﾍﾞﾙ1よりも高い優先権を持つ唯一の割り込みです。より多くの情報については「遮蔽不可割り込み」章をご覧ください。

5.1. 操作

高優先ﾍﾞｸﾀ機能を使うには、単に優先ﾚﾍﾞﾙ1保持割り込みﾍﾞｸﾀ(CPUINT.LVL1VEC)ﾚｼﾞｽﾀに臨む割り込みﾍﾞｸﾀ ｱﾄﾞﾚｽ(番号)を書いてください。この割り込みはﾚﾍﾞﾙ1優先権を割り当てられます。この機能はﾚｼﾞｽﾀ値が$00の時に禁止されます。割り込みﾍﾞｸﾀ配置についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄを参照してください。

ﾚﾍﾞﾙ1割り込みによって割り込まれた場合、ﾚﾍﾞﾙ0処理部はﾚﾍﾞﾙ1処理部が終わった時に実行を再開します。ﾚﾍﾞﾙ1割り込みが時間が重要なﾚﾍﾞﾙ0割り込みのどの実行も邪魔しないのを確実にすることが重要です。ﾚﾍﾞﾙ1割り込みが実行中のﾚﾍﾞﾙ0割り込みに割り込んだかを調べるためにCPUINT.STATUSﾚｼﾞｽﾀのﾚﾍﾞﾙ0割り込み実行中(LVL0EX)ﾋﾞｯﾄを読むことができます。

5.2. 例

以下のｺｰﾄﾞ断片は高優先権割り込みﾍﾞｸﾀを構成設定する方法の例を示します。

高優先権割り込み構成設定

```
// io.hはﾚｼﾞｽﾀとﾍﾞｸﾀの定義を持つﾃﾞﾊﾞｲｽ ﾍｯﾀﾞﾌｧｲﾙを含みます。
#include <avr/io.h>
// interrupt.hはISR関連内容を含みます。
#include <avr/interrupt.h>

void high_priority_interrupt_example(void) {
    // ﾚﾍﾞﾙ1割り込みﾍﾞｸﾀを設定
    CPUINT.LVL1VEC = PORTA_PORT_vect_num;

    // 全体割り込み許可
    sei();
}

ISR(PORTA_PORT_vect){
    // この割り込みはCPUINT.LVL1VECで構成設定後にﾚﾍﾞﾙ1優先権を持ちます。

    // ﾚﾍﾞﾙ0 ISRが割り込まれたか調査
    if (CPUINT.STATUS & CPUINT_LVL0EX_bm) {

    }
}
```

6. 遮蔽不可割り込み

遮蔽不可割り込み(NMI:Non-Maskable Interrupt)は通常、ｼｽﾃﾑで重要な割り込みで、常に他の全ての割り込みを超える優先権を取ります。また、それらは例え全体割り込み(I)が禁止されていても応答されます。

6.1. 操作

変更することができない固有のNMI優先ﾚﾍﾞﾙを持つ特別なﾍﾞｸﾀがあります。それらは他の割り込みと同じ方法で許可されなければなりません。利用可能なNMI元についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄの割り込みﾍﾞｸﾀ配置を参照してください。

CPUINT.STATUSﾚｼﾞｽﾀのﾚﾍﾞﾙ0割り込み実行中(LVL0EX)とﾚﾍﾞﾙ1割り込み実行中(LVL1EX)のﾋﾞｯﾄを読むことにより、応用はNMIによって何れかのﾚﾍﾞﾙ0またはﾚﾍﾞﾙ1のISRが割り込まれたかどうかを調べることができます。

6.2. 例

以下のｺｰﾄﾞ断片は遮蔽不可割り込みがどう実装され得るかを実演します。

遮蔽不可割り込み構成設定、CRCSCAN

```
// io.hはﾚｼﾞｽﾀとﾍﾞｸﾀの定義を持つﾃﾞﾊﾞｲｽ ﾍｯﾀﾞﾌｧｲﾙを含みます。
#include <avr/io.h>
// interrupt.hはISR関連内容を含みます。
#include <avr/interrupt.h>

void nmi_example(void) {
    // NMIでCRCSCANを構成設定
    CRCSCAN.CTRLB = CRCSCAN_MODE_PRIORITY_gc | CRCSCAN_SRC_FLASH_gc;
    CRCSCAN.CTRLA = CRCSCAN_NMIEN_bm | CRCSCAN_ENABLE_bm;
}

ISR(CRCSCAN_NMI_vect) {
    // CRC操作失敗の場合、ここでNMI割り込みが処理されます。
    // NMI要求はｼｽﾃﾑがﾘｾｯﾄされるまで活性に留まり、禁止することができません。

    // ﾚﾍﾞﾙ0または1のISRが割り込まれたか調査
    if (CPUINT.STATUS & (CPUINT_LVL0EX_bm | CPUINT_LVL1EX_bm) {

    }
}
```

CRCSCAN周辺機能を用いるためのｺｰﾄﾞ ﾌﾟﾛｼﾞｪｸﾄを準備する方法のより多くの情報については「AN2521 – tinyAVR® 1系ﾃﾞﾊﾞｲｽでのCRCSCAN」応用記述を参照してください。

7. 簡潔ﾍﾞｸﾀ表

応用が少しの割り込みだけを使う場合、それらのﾍﾞｸﾀ位置だけが必要とされ、従って、ﾍﾞｸﾀ表の殆どは未使用空間です。簡潔ﾍﾞｸﾀ表(CVT:Compact Vector Table)は、これが割り込みﾍﾞｸﾀ表を少数のﾍﾞｸﾀに切り詰めるため、割り込み管理の仕組みをもっと効率的にする手法です。

7.1. 操作

tinyAVR 0と1系統及びmegaAVR 1系統のMCUは割り込みﾍﾞｸﾀ表が**表7-1.**で再構成される3つのﾍﾞｸﾀのCVT動作形態を支援します。各ﾍﾞｸﾀは対応する優先ﾚﾍﾞﾙの全ての割り込みを処理するのに使われます。

これは応用が少しの割り込みだけを必要とする時に非常に有用で有り得ます。ﾍﾞｸﾀ表の大きさを減らすことはそれがより少ないﾌﾗｯｼｭ ﾒﾓﾘを占有し、応用ｺｰﾄﾞ用の空間を開放することを意味します。けれども、この手法を使う時に、どの割り込みが実際に処理要求しているかを見つけ出すためにISRに追加のｺｰﾄﾞが必要とされ、これが実行時間の追加となるかもしれないことに注意してください。

簡潔ﾍﾞｸﾀ表は制御A(CPUINT.CTRLA)の簡潔ﾍﾞｸﾀ表(CVT)ﾋﾞｯﾄに'1'を書くことによって許可されます。このｼｽﾃﾑで重要なﾚｼﾞｽﾀは予期せぬ変更を防ぐために構成設定変更保護(CCP:Configuration Change Protection)を持ちます。CCPの詳細については関連ﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄで「CPU」章を参照してください。

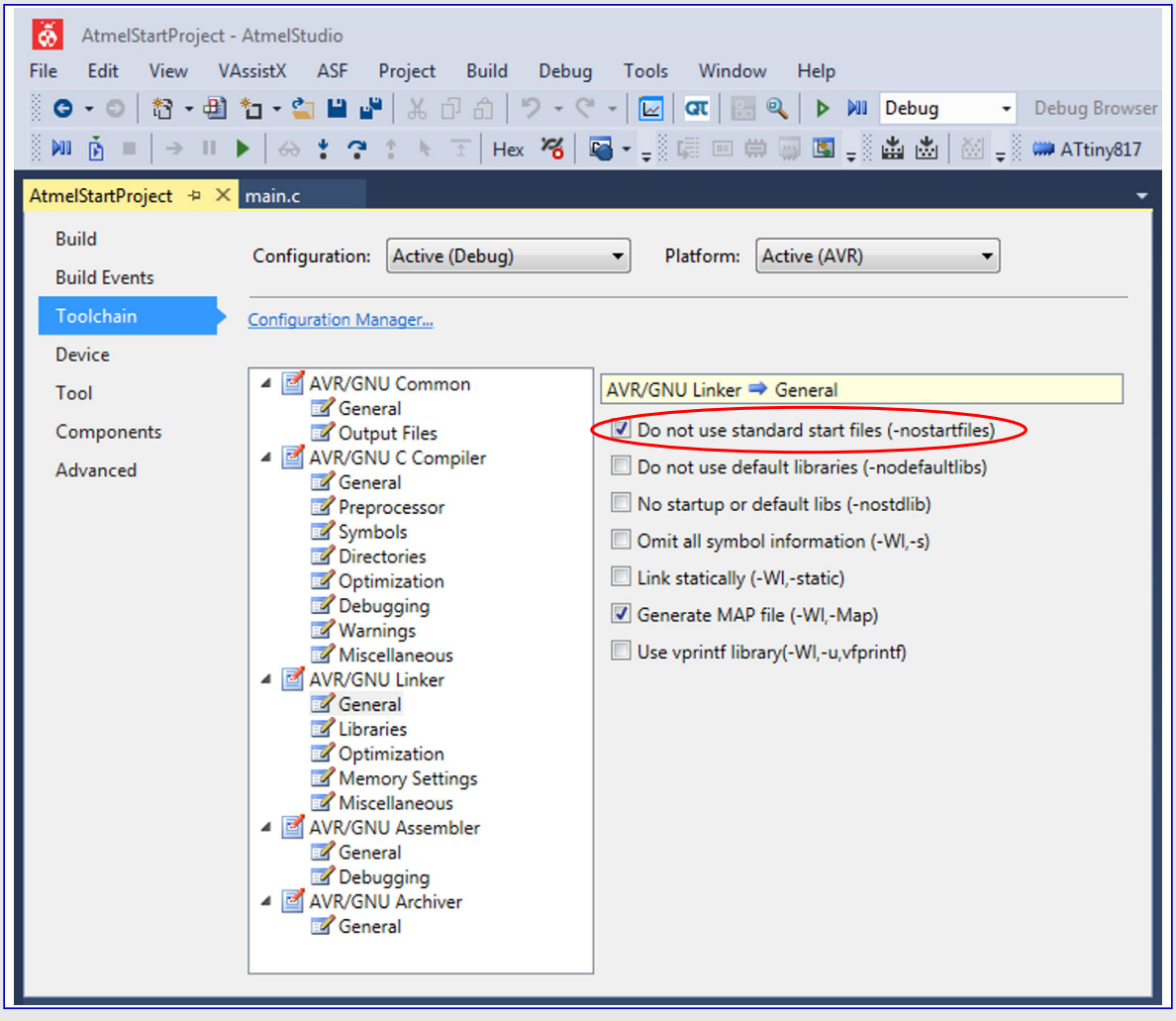
表7-1. 簡潔ﾍﾞｸﾀ配置

ﾍﾞｸﾀ番号	要求元	定義
0	RESET	ﾘｾｯﾄ
1	NMI	遮蔽不可割り込みﾍﾞｸﾀ
2	LVL1	ﾚﾍﾞﾙ1割り込みﾍﾞｸﾀ
3	LVL0	ﾚﾍﾞﾙ0割り込みﾍﾞｸﾀ

7.2. 例

より小さなﾍﾞｸﾀ表を上手く利用するにはAVR GCCによって使われる標準開始ﾌｧｲﾙを置き換えることが必要です。これらはﾌﾟﾛｼﾞｪｸﾄをｺﾝﾊﾟｲﾙする時にﾘﾝｶ ﾌﾗｸﾞの-nostartfilesを設定することによって禁止されます。Atmel Studio 7.0ではこれが**図7-1.**で見られるように、Project Properties(ﾌﾟﾛｼﾞｪｸﾄ ﾌﾟﾛﾊﾟﾃｨ) (Alt+F7) – Toolchain(ﾂｰﾙﾁｪｰﾝ) – AVR/GNU Linker(AVR/GNUﾘﾝｶ) – General(全般)で見つけることができます。

図7-1. Atmel Studio 7.0での"標準開始ﾌｧｲﾙ不使用"構成設定



以下のｺｰﾄﾞ断片は標準開始ﾌｧｲﾙを置き換えることが必要とされるｺｰﾄﾞの例を示します。将来のﾂｰﾙﾁｪｰﾝ公開では簡潔ﾍﾞｸﾀ表を構成設定するのにこの段階が必要とされないでしょう。

簡潔ﾍﾞｸﾀ表用始動ｺｰﾄﾞ例

```
// io.hはﾚｼﾞｽﾀとﾍﾞｸﾀの定義を持つﾃﾞﾊﾞｲｽ ﾍｯﾀﾞﾌｧｲﾙを含みます。
#include <avr/io.h>

// ｺﾝﾊﾟｲﾗにmainと呼ばれるものがあることを知らせます。
extern int main(void);

// ISRを予め宣言することによるｺﾝﾊﾟｲﾗへの親切
void __vector_cvt_nmi(void) __attribute__((weak, alias("dummy_handler")));
void __vector_cvt_lvl1(void) __attribute__((weak, alias("dummy_handler")));
void __vector_cvt_lvl0(void) __attribute__((weak, alias("dummy_handler")));

// ﾍﾞｸﾀ領域構成設定
// RJMP命令は8Kのｺｰﾄﾞ空間を扱うことができ、故にこれは8K以下のﾌﾗｯｼｭ ﾒﾓﾘを持つﾃﾞﾊﾞｲｽのﾍﾞｸﾀ表に使われます。
// 他のﾃﾞﾊﾞｲｽはJMP命令を使います。
__attribute__((section(".vectors"), naked)) void vectors(void)
{
#if (PROGMEM_SIZE <= 0x2000)
    asm("rjmp __ctors_end");
    asm("rjmp __vector_cvt_nmi");
    asm("rjmp __vector_cvt_lvl1");
    asm("rjmp __vector_cvt_lvl0");
#else
    asm("jmp __ctors_end");
    asm("jmp __vector_cvt_nmi");
    asm("jmp __vector_cvt_lvl1");
    asm("jmp __vector_cvt_lvl0");
#endif
}

// AVRｺｱ初期化
__attribute__((section(".init2"), naked)) void init2(void)
{
    // GCCは0のR1を期待
    asm("clr r1");
    // ｽﾃｰﾀｽ ﾚｼﾞｽﾀが既知の状態を持つことを保証
    SREG = 0;
    // ｽﾀｯｸ ﾎﾟｲﾝﾀをｽﾀｯｸの最後にします。
    SPL  = INTERNAL_SRAM_END & 0xff;    // (下位ﾊﾞｲﾄ)
    SPH  = (INTERNAL_SRAM_END >> 8);    // (上位ﾊﾞｲﾄ)
}

// 主処理
__attribute__((section(".init9"), naked)) void init9(void)
{
    main();

    // AVR libcで定義された抜け出し処理部へ飛ぶ。
    // (割り込みを禁止して空の無限繰り返しを走行)
    asm("jmp _exit");
}

// 未使用ISR用偽装処理部偽名
void dummy_handler(void)
{
    while (1)
        ;
}
```

以下のｺｰﾄﾞ断片は前の始動ｺｰﾄﾞ例で定義された割り込みﾍﾞｸﾀ名を用いて簡潔ﾍﾞｸﾀ表を構成設定する方法の例を示します。

簡潔ﾍﾞｸﾀ表構成設定ｺｰﾄﾞ

```
// io.hはﾚｼﾞｽﾀとﾍﾞｸﾀの定義を持つﾃﾞﾊﾞｲｽ ﾍｯﾀﾞﾌｧｲﾙを含みます。
// これはCCPﾏｸﾛの_PROTECTED_WRITE()を含むxmega.hもｲﾝｸﾙｰﾄﾞします。
#include <avr/io.h>
// interrupt.hはISR関連内容を含みます。
#include <avr/interrupt.h>

void compact_vector_table_example(void) {
    // 簡潔ﾍﾞｸﾀ表を設定
    // 構成設定された他のﾋﾞｯﾄを守るために読み-変更-書き
    _PROTECTED_WRITE(CPUINT.CTRLA, (CPUINT.CTRLA | CPUINT_CVT_bm));

    // 全体割り込み許可
    sei();
}

ISR(__vector_cvt_lvl0) {
    // 全てのﾚﾍﾞﾙ0割り込みはここで処理されます。
    if (PORTA.INTFLAGS) {
        // 他のCVT動作形態でPORTA割り込みを処理する方法の例
    }
}

ISR(__vector_cvt_lvl1) {
    // 許可されたなら、ﾚﾍﾞﾙ1割り込みはここで処理されます。
}

ISR(__vector_cvt_nmi) {
    // 許可されたなら、NMI割り込みはここで処理されます。
}
```

Atmel | START例

Atmel | STARTを用いる簡潔ﾍﾞｸﾀ表用の使用事例も利用可能です。より多くの情報については「Atmel | STARTからのｿｰｽ ｺｰﾄﾞ取得」章をご覧ください。

8. ﾍﾞｸﾀ表再配置

既定により、ﾍﾞｸﾀ表はtinyAVR 0と1系統及びmegaAVR 0系統の応用領域に置かれます。けれども、ﾍﾞｸﾀ表をﾌﾞｰﾄ領域の先頭に再配置することが可能で、ﾌﾞｰﾄﾛｰﾀﾞはそのｺｰﾄﾞ内でﾍﾞｸﾀ表を再配置して、主応用に移行する前にその位置を元に戻すことができます。

簡潔ﾍﾞｸﾀ表(CVT)の使用とﾍﾞｸﾀ表の再配置を組み合わせることも可能で、CVTのより多くの情報については「簡潔ﾍﾞｸﾀ表」章をご覧ください。

8.1. 操作

ﾌﾗｯｼｭ ﾒﾓﾘ内のﾌﾞｰﾄ領域の先頭への割り込みﾍﾞｸﾀ表移動は制御A(CPUINT.CTRLA)の割り込みﾍﾞｸﾀ選択(IVSEL)ﾋﾞｯﾄに'1'を書くことによって許可されます。このｼｽﾃﾑで重要なﾚｼﾞｽﾀ ﾋﾞｯﾄは予期せぬ変更を防ぐために構成設定変更保護(CCP:Configuration Change Protection)を持ちます。CCPの詳細についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄで「CPU」章を参照してください。

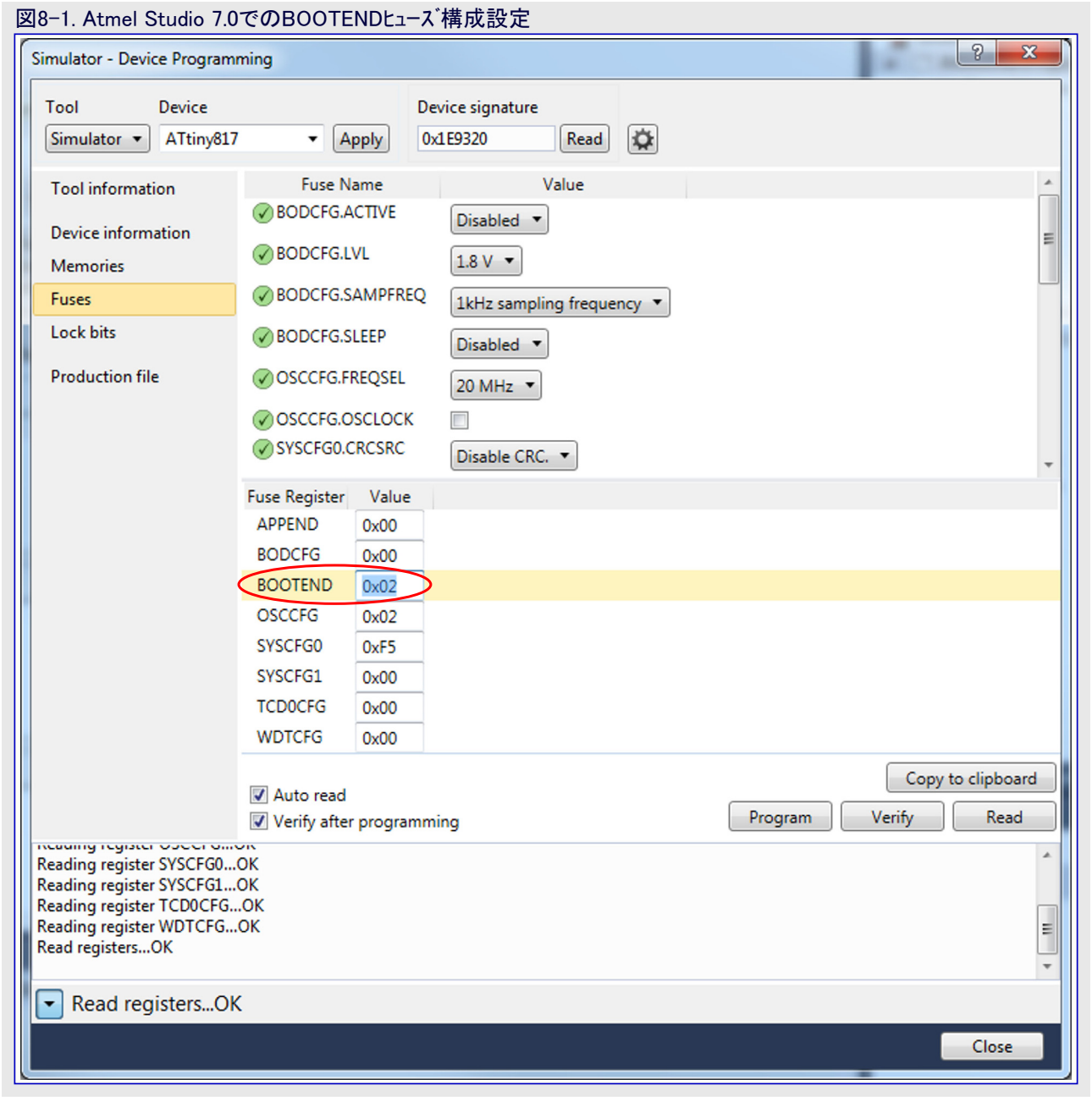
応用領域がﾌﾞｰﾄ領域の後に置かれるため、割り込みﾍﾞｸﾀ表の実際のｱﾄﾞﾚｽはBOOTENDﾋｭｰｽﾞによって決められます。このﾋｭｰｽﾞは256ﾊﾞｲﾄの塊でﾌﾞｰﾄ領域の大きさを設定します。IVSELが設定(1)されると、割り込みﾍﾞｸﾀ表はﾌﾗｯｼｭ ﾒﾓﾘの先頭です。BOOTENDと(応用領域の大きさの)APPENDの両ﾋｭｰｽﾞが$00に設定される場合にﾌﾗｯｼｭ ﾒﾓﾘ全体がﾌﾞｰﾄに構成設定され、割り込みﾍﾞｸﾀ表はﾌﾗｯｼｭ ﾒﾓﾘの先頭になります。

ﾘｾｯﾄ ﾍﾞｸﾀのｱﾄﾞﾚｽはIVSELの両状態で$0000(ﾌﾗｯｼｭ ﾒﾓﾘ先頭)です。これはﾌﾞｰﾄ割り込みﾍﾞｸﾀ表での最低ｱﾄﾞﾚｽに対応し、ﾘｾｯﾄ ﾍﾞｸﾀがﾘｾｯﾄ発生後に実行される最初の命令であることを意味します。

8.2. 例

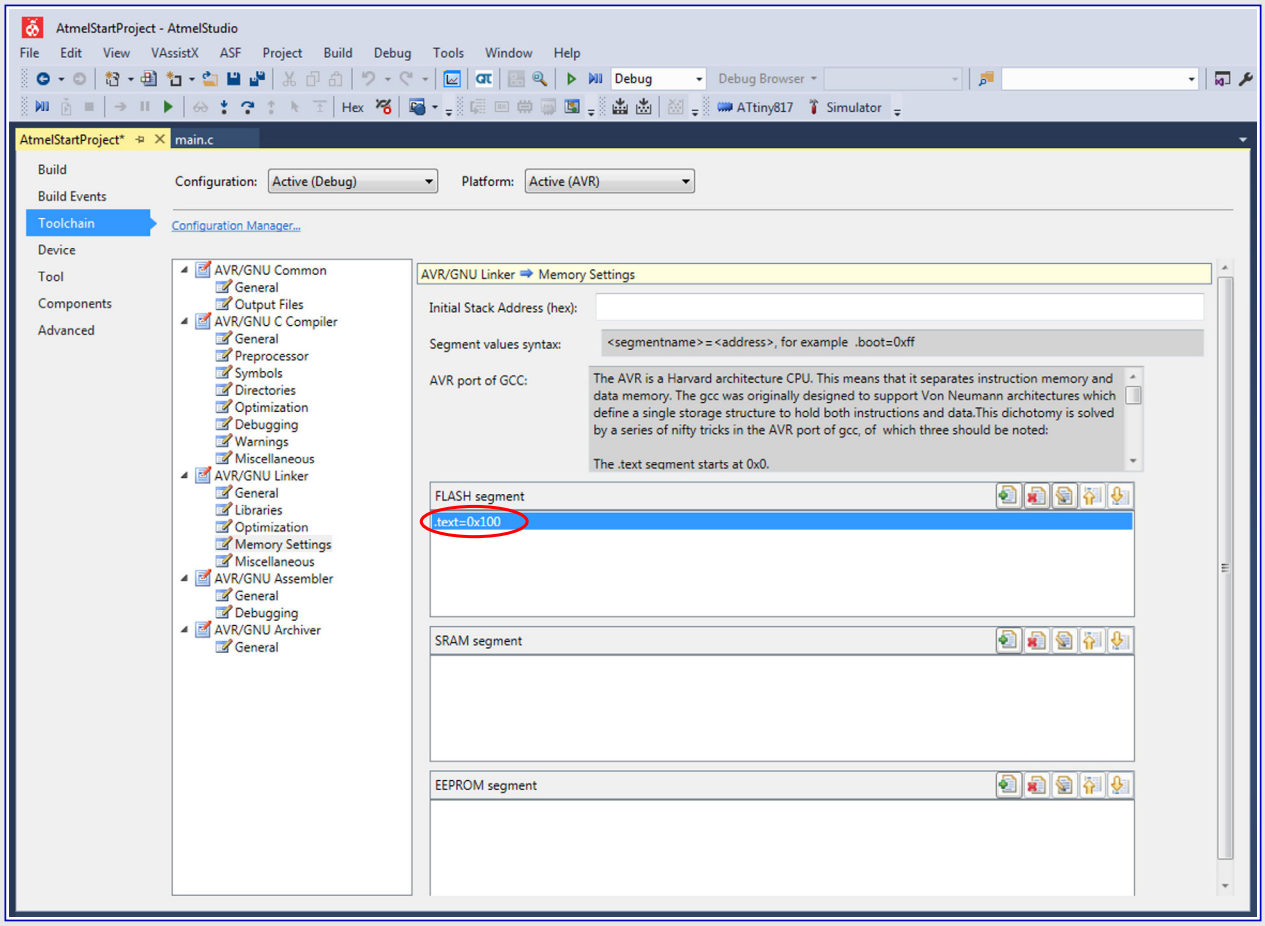
この例は512ﾊﾞｲﾄの大きさを持つﾌﾞｰﾄﾛｰﾀﾞ領域を使います。これはBOOTENDﾋｭｰｽﾞが512/256=$02に構成設定されのが必要とされることを意味します。Atmel Studio 7.0で、このﾋｭｰｽﾞは図8-1.で見られるように、Device Programming(ﾃﾞﾊﾞｲｽ ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ) (ctrl+Shift+P) – Fuses(ﾋｭｰｽﾞ)を用いてﾃﾞﾊﾞｲｽに書くことができます。

図8-1. Atmel Studio 7.0でのBOOTENDﾋｭｰｽﾞ構成設定



AVR® GCCﾌﾟﾛｼﾞｪｸﾄも応用領域の先頭を512ﾊﾞｲﾄ移動するように構成設定されなければなりません。これはﾘﾝｶでtext区部の先頭を変更することによって行われ、この値は16進の語の大きさ、例えば、-Wl,-section-start=bootloader=0x100で設定されなければなりません。Atmel Studi 7.0で、これはProject Properties(ﾌﾟﾛｼﾞｪｸﾄ ﾌﾟﾛﾊﾟﾃｨ) (Alt+F7) – Toolchain(ﾂｰﾙﾁｪｰﾝ) – AVR/GNU Linker(AVR/GNUﾘﾝｶ) – Memory Settings(ﾒﾓﾘ設定)で見つけることができ、**図8-2.**で見られるように、FLASH segment(ﾌﾗｯｼｭ ﾒﾓﾘ区部)に.text=0x100を追加します。

図8-2. Atmel Studio 7.0のAVR GCCでの応用領域先頭構成設定



以下のｺｰﾄﾞ例はﾍﾞｸﾀ表を再配置する方法を示します。

ﾍﾞｸﾀ表再配置構成設定

```
// io.hはﾚｼﾞｽﾀとﾍﾞｸﾀの定義を持つﾃﾞﾊﾞｲｽ ﾍｯﾀﾞﾌｧｲﾙを含みます。
// これはCCPﾏｸﾛの_PROTECTED_WRITE()を含むxmega.hもｲﾝｸﾙｰﾄﾞします。
#include <avr/io.h>
// interrupt.hはISR関連内容を含みます。
#include <avr/interrupt.h>

void compact_vector_table_example(void) {
    // 割り込みﾍﾞｸﾀ選択ﾋﾞｯﾄを設定
    // 構成設定された他のﾋﾞｯﾄを守るために読み-変更-書き
    _PROTECTED_WRITE(CPUINT.CTRLA, (CPUINT.CTRLA | CPUINT_IVSEL_bm));

    // 全体割り込み許可
    sei();
}

ISR(PORTA_PORT_vect) {
    // IVSELﾋﾞｯﾄ設定後、割り込みﾍﾞｸﾀ表は今やBOOTEND×256ﾊﾞｲﾄからﾌﾗｯｼｭ ﾒﾓﾘの先頭($0000)に移動されます。

    // 応用ｺｰﾄﾞで割り込みを使う時には、IVSELが解除(0)され、ﾌﾗｯｼｭ ﾒﾓﾘ内の.text領域でｺﾝﾊﾟｲﾙされたｺｰﾄﾞは
    // BOOTEND×128語に設定されなければなりません。
}
```

9. 要約

ポーリングと割り込み間での選択は応用必要条件に大いに依存しますが、少しの指針に従うことができます。

最初に、その製品が電池給電されるか、そうでなければ制限された電力の場合、ポーリングを避けてください。ポーリングは活動動作に留まることをCPUに必要とし、潜在的に割り込み駆動システムよりももっとたくさん電力を消費します。けれども、増された電力消費が問題なく、1つや2つの状態ビットだけが調査を必要とするなら、ポーリングはソフトウェアでの実装が非常に容易です。

2つ目は、効率的で予測可能な方法で複雑な応用を管理して維持することは困難かもしれません。これは利用可能な割り込み優先機構と1つの割り込みをより高い優先レベルに高めることの可能性とでより容易にさせます。

3つ目は、応用がコード量で制限される場合、これは簡潔ベクタ表を使うことによって減らすことができます。いくつかの場合で、ブートローダに対してだけ簡潔ベクタ表を使い、その後に応用領域で完全な大きさの表に切り替え戻すことが有利かもしれません。これは潜在的にブート領域の大きさを減らすことができ、応用領域のためにもっと空間を残します。

この応用記述で示されるように、新しいtinyAVR 0と1系統及びmegaAVR 0系統のMCUは割り込み処理に関して強力な流れ制御を提供します。これはそれらの設計の必要条件に対して電力消費と応答性を独自化することを設計者に許します。

10. 関連資料

以下の資料はこの応用記述によって網羅されるデバイスと話題に関係しています。

- AVR命令一式手引書
  - http://www.microchip.com/mymicrochip/filehandler.aspx?ddocname=en589391
- AN2521 – tinyAVR® 1系デバイスでのCRCSCAN
  - http://www.microchip.com//wwwAppNotes/AppNotes.aspx?appnote=en599876

11. Atmel | STARTからのソース コード取得

コード例は画像使用者インターフェース(GUI)を通して応用コードの構成設定を許すウェブに基づくAtmel | STARTを通して利用可能です。コードは下の直接コード例リンクまたはAtmel | START先頭頁のBROWSE EXAMPLES(例検索)釦経由Atmel Studio 7.0とIAR Embedded Workbench®の両方に対してダウンロードすることができます。

Atmel | STARTウェブ ページ : http://microchip.com/start

コード例

- CPUINTラウンドロビン割り込み機構:
  - http://start.atmel.com/#example/Atmel:cpuint_examples_tinyavr_megaavr_01:1.0.0::Application:CPUINT_Round-Robin_Scheme:
- CPUINT簡潔ベクタ表:
  - http://start.atmel.com/#example/Atmel:cpuint_examples_tinyavr_megaavr_01:1.0.0::Application:CPUINT_Compact_Vector_Table:

例プロジェクトについての詳細と情報に関してはAtmel | STARTでUser guide(使用者の手引き)を押下してください。User guide釦はAtmel | STARTプロジェクト構成設定部内の一覧画面でプロジェクト名をクリックすることにより、例閲覧部で見つけることができます。

Atmel Studio

DOWNLOAD SELECTED EXAMPLE(選んだ例をダウンロード)をクリックすることにより、Atmel | STARTで例閲覧部からAtmel Studio用.atzipファイルとしてコードをダウンロードしてください。Atmel | START内からファイルをダウンロードするには、EXPORT PROJECT(プロジェクトをエクスポート)に続いてDOWNLOAD PACK(一括ダウンロード)をクリックしてください。

ダウンロードした.atzipファイルをダブル クリックしてください。プロジェクトがAtmel Studio 7.0に導入されます。

IAR Embedded Workbench

IAR Embedded Workbenchでプロジェクトをインポートする方法の情報についてはAtmel | START使用者の手引きを開き、Using Atmel Start Output in External Tools(外部ツールでAtmel START出力を使用)とIAR Embedded Workbenchを選んでください。Atmel | START使用者の手引きへのリンクは共に頁の右上隅に置かれたAtmel | START先頭頁からAbout(これについて)またはプロジェクト構成設定部内のHelp And Support(手助けと支援)をクリックすることによって見つけることができます。

12. 改訂履歴

資料改訂	日付	注釈
A	2017年12月	初版資料公開
B	2018年2月	割り込みベクタ選択(IVSEL)の記述更新

Microchipｳｪﾌﾞ ｻｲﾄ

Microchipはhttp://www.microchip.com/で当社のｳｪﾌﾞ ｻｲﾄ経由でのｵﾝﾗｲﾝ支援を提供します。このｳｪﾌﾞ ｻｲﾄはお客様がﾌｧｲﾙや情報を容易に利用可能にする手段として使われます。お気に入りのｲﾝﾀｰﾈｯﾄ ﾌﾞﾗｳｻﾞを用いてｱｸｾｽすることができ、ｳｪﾌﾞ ｻｲﾄは以下の情報を含みます。

- **製品支援** – ﾃﾞｰﾀｼｰﾄと障害情報、応用記述と試供ﾌﾟﾛｸﾞﾗﾑ、設計資源、使用者の手引きとﾊｰﾄﾞｳｪｱ支援資料、最新ｿﾌﾄｳｪｱ配布と保管されたｿﾌﾄｳｪｱ
- **全般的な技術支援** – 良くある質問(FAQ)、技術支援要求、ｵﾝﾗｲﾝ検討ｸﾞﾙｰﾌﾟ、Microchip相談役ﾌﾟﾛｸﾞﾗﾑ員一覧
- **Microshipの事業** – 製品選択器と注文の手引き、最新Microchip報道発表、ｾﾐﾅｰとｲﾍﾞﾝﾄの一覧、Microchip営業所の一覧、代理店と代表する工場

お客様への変更通知ｻｰﾋﾞｽ

Microchipのお客様通知ｻｰﾋﾞｽはMicrochip製品を最新に保つのに役立ちます。加入者は指定した製品系統や興味のある開発ﾂｰﾙに関連する変更、更新、改訂、障害情報がある場合に必ず電子ﾒｰﾙ通知を受け取ります。

登録するにはhttp://www.microchip.com/でMicrochipのｳｪﾌﾞ ｻｲﾄをｱｸｾｽしてください。”Support”下で”Customer Change Notification”をｸﾘｯｸして登録指示に従ってください。

お客様支援

Microchip製品の使用者は以下のいくつかのﾁｬﾈﾙを通して支援を受け取ることができます。

- 代理店または販売会社
- 最寄りの営業所
- 現場応用技術者(FAE:Field Aplication Engineer)
- 技術支援

お客様は支援に関してこれらの代理店、販売会社、または現場応用技術者(FAE)に連絡を取るべきです。最寄りの営業所もお客様の手助けに利用できます。営業所と位置の一覧はこの資料の後ろに含まれます。

技術支援はhttp://www.microchip.com/supportでのｳｪﾌﾞ ｻｲﾄを通して利用できます。

Microchipﾃﾞﾊﾞｲｽ ｺｰﾄﾞ保護機能

Microchipﾃﾞﾊﾞｲｽでの以下のｺｰﾄﾞ保護機能の詳細に注意してください。

- Microchip製品はそれら特定のMicrochipﾃﾞｰﾀｼｰﾄに含まれる仕様に合致します。
- Microchipは意図した方法と通常条件下で使われる時に、その製品系統が今日の市場でその種類の最も安全な系統の1つであると考えます。
- ｺｰﾄﾞ保護機能を破るのに使われる不正でおそらく違法な方法があります。当社の知る限りこれらの方法の全てはMicrochipのﾃﾞｰﾀｼｰﾄに含まれた動作仕様外の方法でMicrochip製品を使うことが必要です。おそらく、それを行う人は知的財産の窃盗に関与しています。
- Microchipはそれらのｺｰﾄﾞの完全性について心配されているお客様と共に働きたいと思います。
- Microchipや他のどの半導体製造業者もそれらのｺｰﾄﾞの安全を保証することはできません。ｺｰﾄﾞ保護は当社が製品を”破ることができない”として保証すると言うことを意味しません。

ｺｰﾄﾞ保護は常に進化しています。Microchipは当社製品のｺｰﾄﾞ保護機能を継続的に改善することを約束します。Microchipのｺｰﾄﾞ保護機能を破る試みはﾃﾞｼﾞﾀﾙ ﾐﾚﾆｱﾑ著作権法に違反するかもしれません。そのような行為があなたのｿﾌﾄｳｪｱや他の著作物に不正なｱｸｾｽを許す場合、その法律下の救済のために訴権を持つかもしれません。

法的通知

ﾃﾞﾊﾞｲｽ応用などに関してこの刊行物に含まれる情報は皆さまの便宜のためにだけ提供され、更新によって取り換えられるかもしれません。皆さまの応用が皆さまの仕様に合致するのを保証するのは皆さまの責任です。**Microchipはその条件、品質、性能、商品性、目的適合性を含め、明示的にも黙示的にもその情報に関連して書面または表記された書面または黙示の如何なる表明や保証もしません。**Microchipはこの情報とそれの使用から生じる全責任を否認します。生命維持や安全応用でのMicrochipﾃﾞﾊﾞｲｽの使用は完全に購入者の危険性で、購入者はそのような使用に起因する全ての損害、請求、訴訟、費用からMicrochipを擁護し、補償し、免責にすることに同意します。他に言及されない限り、Microchipのどの知的財産権下でも暗黙的または違う方法で許認可は譲渡されません。

商標

DNVによって認証された品質管理ｼｽﾃﾑ

ISO/TS 16949

Microchipはその世界的な本社、ｱﾘｿﾞﾅ州のﾁｬﾝﾄﾞﾗｰとﾃﾝﾍﾟ、ｵﾚｺﾞﾝ州ｸﾞﾗｼｬﾑの設計とｳｪﾊｰ製造設備とｶﾘﾌｫﾙﾆｱとｲﾝﾄﾞの設計ｾﾝﾀｰに対してISO/TS-16949:2009認証を取得しました。当社の品質ｼｽﾃﾑの処理と手続きはPIC® MCUとdsPIC® DSC、KEELOQ符号飛び回りﾃﾞﾊﾞｲｽ、直列EEPROM、ﾏｲｸﾛ周辺機能、不揮発性ﾒﾓﾘ、ｱﾅﾛｸﾞ製品用です。加えて、開発ｼｽﾃﾑの設計と製造のためのMicrochipの品質ｼｽﾃﾑはISO 9001:2000認証取得です。

本応用記述はMicrochipのAN1982応用記述(40001982B-2018年2月)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。

世界的な販売とサービス

米国	亜細亜/太平洋	亜細亜/太平洋	欧州
本社 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術支援: http://www.microchip.com/support ｳｪﾌﾞｱﾄﾞﾚｽ: www.microchip.com **ｱﾄﾗﾝﾀ** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **ｵｰｽﾁﾝ** TX Tel: 512-257-3370 **ﾎﾞｽﾄﾝ** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **ｼｶｺﾞ** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **ﾀﾞﾗｽ** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **ﾃﾞﾄﾛｲﾄ** Novi, MI Tel: 248-848-4000 **ﾋｭｰｽﾄﾝ** TX Tel: 281-894-5983 **ｲﾝﾃﾞｱﾅﾎﾟﾘｽ** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **ﾛｻﾝｾﾞﾙｽ** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **ﾛｰﾘｰ** NC Tel: 919-844-7510 **ﾆｭｰﾖｰｸ** NY Tel: 631-435-6000 **ｻﾝﾎｾ** CA Tel: 408-735-9110 Tel: 408-436-4270 **ｶﾅﾀﾞ - ﾄﾛﾝﾄ** Tel: 905-695-1980 Fax: 905-695-2078	**ｵｰｽﾄﾗﾘｱ - ｼﾄﾞﾆｰ** Tel: 61-2-9868-6733 **中国 - 北京** Tel: 86-10-8569-7000 **中国 - 成都** Tel: 86-28-8665-5511 **中国 - 重慶** Tel: 86-23-8980-9588 **中国 - 東莞** Tel: 86-769-8702-9880 **中国 - 広州** Tel: 86-20-8755-8029 **中国 - 杭州** Tel: 86-571-8792-8115 **中国 - 香港特別行政区** Tel: 852-2943-5100 **中国 - 南京** Tel: 86-25-8473-2460 **中国 - 青島** Tel: 86-532-8502-7355 **中国 - 上海** Tel: 86-21-3326-8000 **中国 - 瀋陽** Tel: 86-24-2334-2829 **中国 - 深圳** Tel: 86-755-8864-2200 **中国 - 蘇州** Tel: 86-186-6233-1526 **中国 - 武漢** Tel: 86-27-5980-5300 **中国 - 西安** Tel: 86-29-8833-7252 **中国 - 廈門** Tel: 86-592-2388138 **中国 - 珠海** Tel: 86-756-3210040	**ｲﾝﾄﾞ - ﾊﾝｶﾞﾛｰﾙ** Tel: 91-80-3090-4444 **ｲﾝﾄﾞ - ﾆｭｰﾃﾞﾘｰ** Tel: 91-11-4160-8631 **ｲﾝﾄﾞ - ﾌﾟﾈｰ** Tel: 91-20-4121-0141 **日本 - 大阪** Tel: 81-6-6152-7160 **日本 - 東京** Tel: 81-3-6880-3770 **韓国 - 大邱** Tel: 82-53-744-4301 **韓国 - ｿｳﾙ** Tel: 82-2-554-7200 **ﾏﾚｰｼｱ - ｸｱﾗﾙﾝﾌﾟｰﾙ** Tel: 60-3-7651-7906 **ﾏﾚｰｼｱ - ﾍﾟﾅﾝ** Tel: 60-4-227-8870 **ﾌｨﾘﾋﾟﾝ - ﾏﾆﾗ** Tel: 63-2-634-9065 **ｼﾝｶﾞﾎﾟｰﾙ** Tel: 65-6334-8870 **台湾 - 新竹** Tel: 886-3-577-8366 **台湾 - 高雄** Tel: 886-7-213-7830 **台湾 - 台北** Tel: 886-2-2508-8600 **ﾀｲ - ﾊﾞﾝｺｸ** Tel: 66-2-694-1351 **ﾍﾞﾄﾅﾑ - ﾎｰﾁﾐﾝ** Tel: 84-28-5448-2100	**ｵｰｽﾄﾘｱ - ヴｪﾙｽ** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **ﾃﾞﾝﾏｰｸ - ｺﾍﾟﾝﾊｰｹﾞﾝ** Tel: 45-4450-2828 Fax: 45-4485-2829 **ﾌｨﾝﾗﾝﾄﾞ - ｴｽﾎﾟｰ** Tel: 358-9-4520-820 **ﾌﾗﾝｽ - ﾊﾟﾘ** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **ﾄﾞｲﾂ - ｶﾞﾙﾋﾝｸﾞ** Tel: 49-8931-9700 **ﾄﾞｲﾂ - ﾊｰﾝ** Tel: 49-2129-3766400 **ﾄﾞｲﾂ - ﾊｲﾙﾌﾞﾛﾝ** Tel: 49-7131-67-3636 **ﾄﾞｲﾂ - ｶｰﾙｽﾙｰｴ** Tel: 49-721-625370 **ﾄﾞｲﾂ - ﾐｭﾝﾍﾝ** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **ﾄﾞｲﾂ - ﾛｰｾﾞﾝﾊｲﾑ** Tel: 49-8031-354-560 **ｲｽﾗｴﾙ - ﾗｰﾅﾅ** Tel: 972-9-744-7705 **ｲﾀﾘｱ - ﾐﾗﾉ** Tel: 39-0331-742611 Fax: 39-0331-466781 **ｲﾀﾘｱ - ﾊﾟﾄﾞﾊﾞ** Tel: 39-049-7625286 **ｵﾗﾝﾀﾞ - ﾃﾞﾙｰﾈﾝ** Tel: 31-416-690399 Fax: 31-416-690340 **ﾉﾙｳｪｰ - ﾄﾛﾝﾊｲﾑ** Tel: 47-7289-7561 **ﾎﾟｰﾗﾝﾄﾞ - ﾜﾙｼｬﾜ** Tel: 48-22-3325737 **ﾙｰﾏﾆｱ - ﾌﾞｶﾚｽﾄ** Tel: 40-21-407-87-50 **ｽﾍﾟｲﾝ - ﾏﾄﾞﾘｰﾄﾞ** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **ｽｳｪｰﾃﾞﾝ - ｲｪｰﾃﾎﾞﾘ** Tel: 46-31-704-60-40 **ｽｳｪｰﾃﾞﾝ - ｽﾄｯｸﾎﾙﾑ** Tel: 46-8-5090-4654 **ｲｷﾞﾘｽ - ｳｫｰｷﾝｶﾞﾑ** Tel: 44-118-921-5800 Fax: 44-118-921-5820