TB3213

RTCでの開始に際して

序説

著者: Victor Berzan, Microchip Technology Inc.

実時間計数器(RTC:Real-Time Counter)は計数ﾚｼﾞｽﾀで(前置分周した)ｸﾛｯｸ周期を計数し、計数ﾚｼﾞｽﾀの内容を定期ﾚｼﾞｽﾀと比較ﾚｼﾞｽﾀに対して比較します。RTCは比較一致や溢れで割り込みと事象の両方を生成することができます。計数器が比較ﾚｼﾞｽﾀ値と等しい後の最初の計数で比較割り込みや事象を、計数器値が定期ﾚｼﾞｽﾀ値と等しい後の最初の計数で溢れ割り込みや事象を生成します。溢れは計数器値を0にもﾘｾｯﾄします。

この技術概要はtinyAVR® 0及び1系とmegaAVR® 0系のﾏｲｸﾛ ｺﾝﾄﾛｰﾗでRTC単位部がどう動くかを記述します。これは以下の使用事例を網羅します。

- **RTC溢れ割り込み**:

  RTCを初期化、溢れ割り込みを許可、各溢れでLEDを交互切り替えします。

- **RTC周期的割り込み**:

  RTC PITを初期化、周期的割り込みを許可、各周期的割り込みでLEDを交互切り替えします。

- **休止からのRTC PIT起き上がり**:

  RTC PITを初期化、周期的割り込みを許可、ﾃﾞﾊﾞｲｽを休止動作に構成設定、CPUを休止に置く、PIT割り込みがCPUを起き上がらせます。

注: ｺｰﾄﾞ例はATmega4809 Xplained Pro (ATMEGA4809-XPRO)で開発されました。

目次

1. 関連ﾃﾞﾊﾞｲｽ

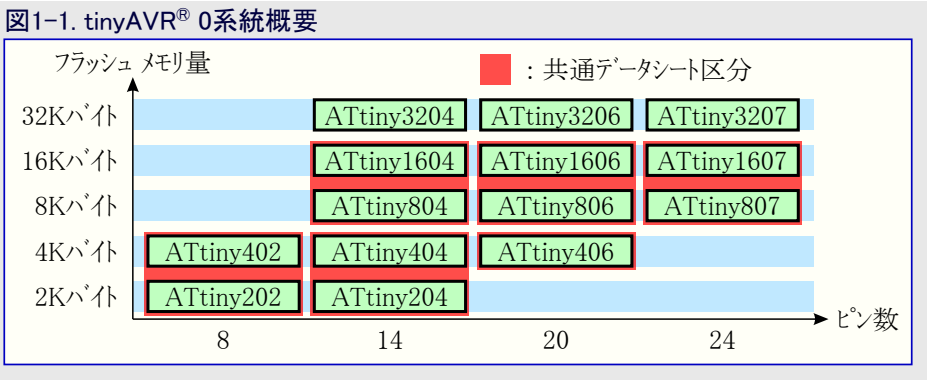
本章はこの資料に関連するﾃﾞﾊﾞｲｽを一覧にします。

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



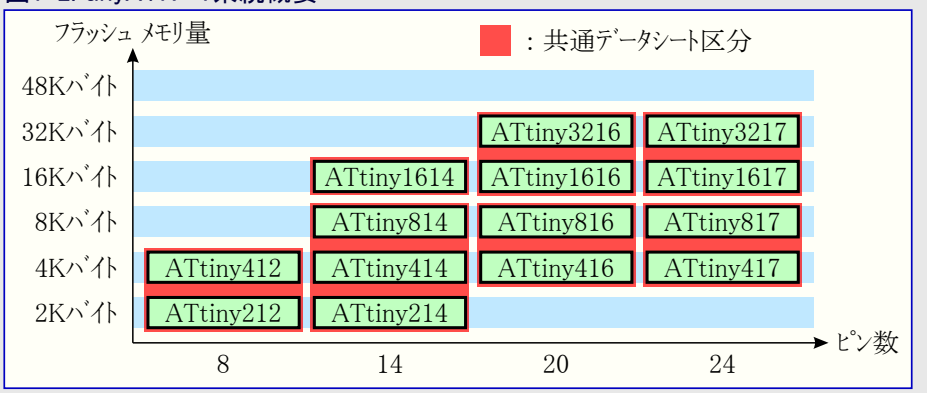
異なるﾌﾗｯｼｭ ﾒﾓﾘ量を持つﾃﾞﾊﾞｲｽは一般的に異なるSRAMとEEPROMの量を持ちます。

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。

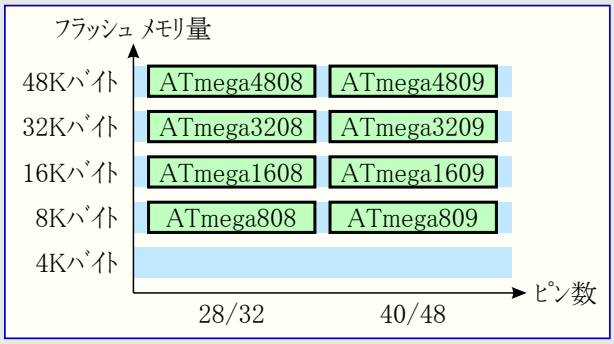
1.3. megaAVR® 0系統

右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。

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。

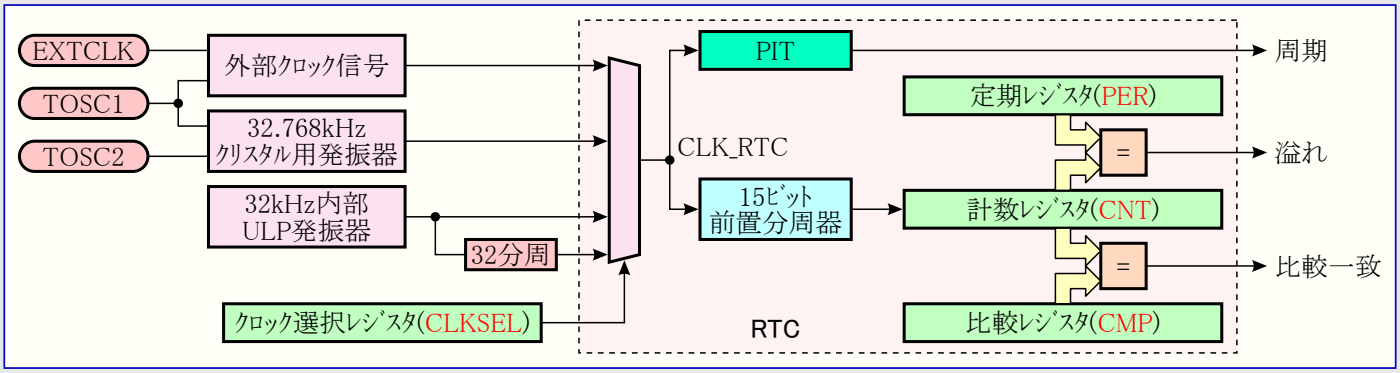
図1-3. megaAVR® 0系統概要



2. 概要

RTC周辺機能は実時間計数器(RTC:Real-Time Counter)と周期的割り込み計時器(PIT:Periodic Interrupt Timer)の2つのﾀｲﾐﾝｸﾞ機能を提供します。PIT機能はRTC機能と独立して許可することができます。

図2-1. 構成図



PIT機能とRTC機能は前置分周器内の同じ計数器で動いています。CNTを増加するｸﾛｯｸ信号の周期はRTC.CTRLAのPRESCALERﾋﾞｯﾄ領域に書くことによって構成設定されます。RTC.PITCTRLAのPERIODﾋﾞｯﾄ領域はPIT周期出力として使われる15ﾋﾞｯﾄ前置分周器からのﾋﾞｯﾄを選びます。

3. RTC溢れ割り込み

RTCを動かすにはRTC周辺機能と望む活動(割り込み供給、出力事象)を許可する前にRTC計数器に対する供給元ｸﾛｯｸが構成設定されなければなりません。この例では供給元ｸﾛｯｸとして32.768kHz外部用発振器が使われます。

発振器を構成設定するには、最初にCLKCTRL.XOSC32KCTRALﾚｼﾞｽﾀのENABLEﾋﾞｯﾄを解除(0)することによってそれが禁止されなければなりません。

図3-1. CLKCTRL.XOSC32KCTRLA – ENABLEﾋﾞｯﾄ解除(0)

供給元選択(SEL)とｸﾘｽﾀﾙ始動時間(CSUT)のﾋﾞｯﾄは許可(ENABLE)ﾋﾞｯﾄが設定(1)される、または主ｸﾛｯｸ状態(CLKCTRL.MCLKSTATUS)ﾚｼﾞｽﾀの32.768kHzｸﾘｽﾀﾙ用発振器状態安定(XOSC32KS)ﾋﾞｯﾄが'1'である限り、変更することができません。

安全な方法で設定を変更するには、ENABLEﾋﾞｯﾄに'0'を書き、新設定でXOSC32Kを許可する前に、XOSC32KSが'0'になるまで待ってください。

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
			CSUT1,0			SEL	RUNSTDBY	ENABLE
ｱｸｾｽ種別	R	R	R/W	R/W	R	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ0 – ENABLE : 許可** (Enable)

このﾋﾞｯﾄが'1'を書かれると、TOSC1とTOSC2に対する各々の入力ﾋﾟﾝの構成設定が無効にされます。また、供給元選択(SEL)とｸﾘｽﾀﾙ始動時間(CSUT)のﾋﾞｯﾄは読み込み専用になります。

このﾋﾞｯﾄはこの発振器の意図せぬ許可を防ぐためにI/O保護されます。

```
uint8_t temp;
temp = CLKCTRL.XOSC32KCTRLA;
temp &= ~CLKCTRL_ENABLE_bm;
CPU_CCP = CCP_IOREG_gc;
CLKCTRL.XOSC32KCTRLA = temp;
```

使用者はその後に対応する状態ﾋﾞｯﾄが'0'になるのを待たなければなりません。

図3-2. CLKCTRL.MCLKSTATUS – XOSC32KS読み込み

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	EXTS	XOSC32KS	OSC32KS	OSC20MS				SOSC
ｱｸｾｽ種別	R	R	R	R	R	R	R	R
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ6 – XOSC32KS : 32.768kHzｸﾘｽﾀﾙ用発振器状態** (XOSC32K Status)

この状態ﾋﾞｯﾄはこの供給元が主ｸﾛｯｸとして、または別の周辺機能によって要求された場合にだけ利用可能です。この発振器のｽﾀﾝﾊﾞｲ時走行(RUNSTDBY)ﾋﾞｯﾄが設定(1)であるけれども発振器が未使用/要求なしの場合、このﾋﾞｯﾄは0になります。

図3-2 (続き). CLKCTRL.MCLKSTATUS – XOSC32KS読み込み

値	0	1
説明	XOSC32Kは安定ではありません。	XOSC32Kは安定です。

```
while(CLKCTRL.MCLKSTATUS & CLKCTRL_XOSC32KS_bm)
{
    ;
}
```

CLKCTRL.XOSC32KCTRLAレジスタでSELビットを解除(0)することによって外部用発振器が選ばれなければなりません。

図3-3. CLKCTRL.XOSC32KCTRLA – SELビット解除(0)

ビット	7	6	5	4	3	2	1	0
			CSUT1,0			SEL	RUNSTDBY	ENABLE
アクセス種別	R	R	R/W	R/W	R	R/W	R/W	R/W
リセット値	0	0	0	0	0	0	0	0

● **ビット2 – SEL : 供給元選択** (Source Select)

このビットは外部供給元形式を選びます。この発振器が許可される(ENABLE=1の)時に書き込み保護にされます。

値	0	1
説明	外部クリスタル	TOSC1ピンでの外部クロック

```
temp = CLKCTRL.XOSC32KCTRLA;
temp &= ~CLKCTRL_SEL_bm;
CPU_CCP = CCP_IOREG_gc;
CLKCTRL.XOSC32KCTRLA = temp;
```

その後、発振器はCLKCTRL.XOSC32KCTRLAレジスタでENABLEビットを設定(1)することによって許可されなければなりません。

```
temp = CLKCTRL.XOSC32KCTRLA;
temp |= CLKCTRL_ENABLE_bm;
CPU_CCP = CCP_IOREG_gc;
CLKCTRL.XOSC32KCTRLA = temp;
```

その後、使用者は全てのレジスタに対して同期されるのを待たなければなりません。

図3-4. RTC.STATUS

ビット	7	6	5	4	3	2	1	0
					CMPBUSY	PERBUSY	CNTBUSY	CTRLABUSY
アクセス種別	R	R	R	R	R/W	R/W	R/W	R/W
リセット値	0	0	0	0	0	0	0	0

● **ビット3 – CMPBUSY : 比較同期中** (Compare Synchronization Busy)

このビットはRTCがRTCクロック領域に比較(RTC.CMP)レジスタを同期中多忙かどうかを示します。

● **ビット2 – PERBUSY : 定期同期中** (Period Synchronization Busy)

このビットはRTCがRTCクロック領域に定期(RTC.PER)レジスタを同期中多忙かどうかを示します。

● **ビット1 – CNTBUSY : 計数器同期中** (Counter Synchronization Busy)

このビットはRTCがRTCクロック領域に計数(RTC.CNT)レジスタを同期中多忙かどうかを示します。

● **ビット0 – CTRLABUSY : 制御A同期中** (Control A Synchronization Busy)

このビットはRTCがRTCクロック領域に制御A(RTC.CTRLA)レジスタを同期中多忙かどうかを示します。

```
while (RTC.STATUS > 0)
{
    ;
}
```

RTC周期はRTC.PERﾚｼﾞｽﾀで設定されます。

図3-5. RTC.PER – 周期設定

RTC.PERHとRTC.PERLのﾚｼﾞｽﾀ対は16ﾋﾞｯﾄ値のPERを表します。下位ﾊﾞｲﾄ[7～0](接尾辞L)は変位原点でｱｸｾｽできます。上位ﾊﾞｲﾄ[15～8](接尾辞H)は変位+1でｱｸｾｽすることができます。16ﾋﾞｯﾄ ﾚｼﾞｽﾀの読み書きのより多くの詳細については「mega AVR 0系 手引書(DS40002015)」から「CPU」章の「16ﾋﾞｯﾄ ﾚｼﾞｽﾀのｱｸｾｽ」を参照してください。

RTCｸﾛｯｸとｼｽﾃﾑ ｸﾛｯｸの領域間同期のため、ﾚｼﾞｽﾀ更新からそれが効果を持つまでに2 RTCｸﾛｯｸ周期の遅延があります。応用ｿﾌﾄｳｪｱはこのﾚｼﾞｽﾀへ書く前に状態(STATUS)ﾚｼﾞｽﾀの定期同期中(PERBUSY)ﾌﾗｸﾞが解除されていることを調べることが必要です。

ﾋﾞｯﾄ	15	14	13	12	11	10	9	8
	PER15~8							
ｱｸｾｽ種別	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	1	1	1	1	1	1	1	1

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	PER7~0							
ｱｸｾｽ種別	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	1	1	1	1	1	1	1	1

- **ﾋﾞｯﾄ15~8 – PER15~8 : 定期値上位ﾊﾞｲﾄ** (Periodic high byte)

これらのﾋﾞｯﾄは16ﾋﾞｯﾄ定期ﾚｼﾞｽﾀの上位ﾊﾞｲﾄを保持します。

- **ﾋﾞｯﾄ7~0 – PER7~0 : 定期値下位ﾊﾞｲﾄ** (Periodic low byte)

これらのﾋﾞｯﾄは16ﾋﾞｯﾄ定期ﾚｼﾞｽﾀの下位ﾊﾞｲﾄを保持します。

32.768kHz外部ｸﾘｽﾀﾙ用発振器がRTC.CLKSELﾚｼﾞｽﾀで選ばれます。

図3-6. RTC.CLKSEL – ｸﾛｯｸ選択

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
							CLKSEL1,0	
ｱｸｾｽ種別	R	R	R	R	R	R	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

- **ﾋﾞｯﾄ1,0 – CLKSEL1,0 : ｸﾛｯｸ選択** (Clock Select)

これらのﾋﾞｯﾄ書き込みはRTCｸﾛｯｸ(CLK_RTC)用の供給元を選びます。

RTCをXOSC32KまたはTOSC1での外部ｸﾛｯｸを使うように構成設定すると、それに応じてｸﾛｯｸ制御器のXOSC32K制御A(CLKCTRL.XOSC32KCTRLA)ﾚｼﾞｽﾀでXOSC32Kが許可(ENABLE=1)され、供給元選択(SEL)ﾋﾞｯﾄとｽﾀﾝﾊﾞｲ時走行(RUNSTDBY)ﾋﾞｯﾄが構成設定されなければなりません。

値	0 0	0 1	1 0	1 1
名称	INT32K	INT1K	TOSC32K	EXTCLK
説明	OSCULP32Kからの32kHz	OSCULP32Kからの1kHz	XOSC32Kからの32.768kHz	TOSC1ﾋﾟﾝからの外部ｸﾛｯｸ

```
RTC.CLKSEL = RTC_CLKSEL_TOSC32K_gc;
```

RTCをﾃﾞﾊﾞｯｸﾞ動作でも動かすのを許すため、RTC.DBGCTRLﾚｼﾞｽﾀでDBGRUNﾋﾞｯﾄが設定(1)されます。

図3-7. RTC.DBGCTRL – DBGRUNﾋﾞｯﾄ設定(1)

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
								DBGRUN
ｱｸｾｽ種別	R	R	R	R	R	R	R	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

- **ﾋﾞｯﾄ0 – DBGRUN : ﾃﾞﾊﾞｯｸﾞ時走行** (Debug Run)

値	0	1
説明	RTCはﾃﾞﾊﾞｯｸﾞ動作中断で停止し、事象を無視	RTCはCPU停止中のﾃﾞﾊﾞｯｸﾞ動作中断で走行継続

```
RTC.DBGCTRL |= RTC_DBGRUN_bm;
```

RTC前置分周器はRTC.CTRLAレジスタで設定されます。スタンバイ動作でも動くことをRTCに許すため、RTC.CTRLAレジスタでRUNSTDBYビットが設定(1)されます。RTCを許可するため、RTC.CTRLAレジスタでRTCENビットが設定(1)されます。

図3-8. RTC.CTRLA – 前置分周器、RUNSTDBYビット、RTCENビット設定

ビット	7	6	5	4	3	2	1	0
	RUNSTDBY	PRESCALER3~0				CORREN		RTCEN
アクセス種別	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
リセット値	0	0	0	0	0	0	0	0

● **ビット7 – RUNSTDBY : スタンバイ時走行** (Run Standby)

値	0	1
説明	スタンバイ休止動作でRTC禁止	スタンバイ休止動作でRTC許可

● **ビット6~3 – PRESCALER3~0 : 前置分周器** (Prescaler)

これらのビットはCLK_RTCクロック信号の前置分周を定義します。RTCクロックとシステム クロックの領域間同期のため、レジスタ更新からそれが効果を持つまでに2 RTCクロック周期の遅延があります。応用ソフトウェアはこのレジスタへ書く前に状態(RTC.STATUS)レジスタの制御A同期中(CTRLABUSY)フラグが解除(0)されていることを調べることが必要です。

値	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1
名称	DIV1	DIV2	DIV4	DIV8	DIV16	DIV32	DIV64	DIV128
説明	CLK_RTC/1	CLK_RTC/2	CLK_RTC/4	CLK_RTC/8	CLK_RTC/16	CLK_RTC/32	CLK_RTC/64	CLK_RTC/128
値	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
名称	DIV256	DIV512	DIV1024	DIV2048	DIV4096	DIV8192	DIV16384	DIV32768
説明	CLK_RTC/256	CLK_RTC/512	CLK_RTC/1024	CLK_RTC/2048	CLK_RTC/4096	CLK_RTC/8192	CLK_RTC/16384	CLK_RTC/32768

● **ビット0 – RTCEN : RTC許可** (RTC Enable)

値	0	1
説明	RTC禁止	RTC許可

```
RTC.CTRLA = RTC_PRESCALER_DIV32_gc | RTC_RTCEN_bm | RTC_RUNSTDBY_bm;
```

溢れ割り込みはRTC.INTCTRLレジスタでOVFビットを設定(1)することによって許可されます。

図3-9. RTC.INTCTRL – OVFビット設定(1)

ビット	7	6	5	4	3	2	1	0
							CMP	OVF
アクセス種別	R	R	R	R	R	R	R/W	R/W
リセット値	0	0	0	0	0	0	0	0

● **ビット0 – OVF : 溢れ割り込み許可** (Overflow Interrupt Enable)

計数器溢れ(即ち、計数(CNT)値が定期(PER)値と一致して0に丸められる時の)割り込みを許可します。

```
RTC.INTCTRL |= RTC_OVF_bm;
```

割り込みが起こるには全体割り込みも許可されなければなりません。

```
sei();
```

RTC溢れ割り込み処理ルーチン(ISR:Interrupt Service Routine)は下の例でLEDを交互切り替えします。

```
ISR(RTC_CNT_vect)
{
    RTC.INTFLAGS = RTC_OVF_bm;
    LED0_toggle();
}
```

注: RTC.INTFLAGSのOVFビットはISR関数内でそれに'1'を書くことによって解除(0)されなければなりません。

GitHubでコード例を見てください。
貯蔵庫を閲覧するにはクリックしてください。

助言: 完全なｺｰﾄﾞ例は「**追補**」章でも利用可能です。

4. RTC周期的割り込み

この特定例に対する供給元ｸﾛｯｸ構成設定はRTC溢れ割り込み例に対してと同じです。周期的割り込みはRTC.PITINTCTRLﾚｼﾞｽﾀのPIﾋﾞｯﾄを設定(1)することによって許可されます。

図4-1. RTC.PITINTCTRL – PIﾋﾞｯﾄ設定(1)

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
								PI
ｱｸｾｽ種別	R	R	R	R	R	R	R	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ0 – PI : 周期割り込み許可** (Periodic Interrupt)

値	0	1
説明	周期割り込み禁止	周期割り込み許可

```
RTC.PITINTCTRL = RTC_PI_bm;
```

PIT周期はRTC.PITCTRLAﾚｼﾞｽﾀで設定されます。PITはRTC.PITCTRLAﾚｼﾞｽﾀでPITENﾋﾞｯﾄを設定(1)することによって許可されます。

図4-2. RTC.PITCTRLA – PITENﾋﾞｯﾄ設定(1)

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
		PERIOD3~0						PITEN
ｱｸｾｽ種別	R	R/W	R/W	R/W	R/W	R	R	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ6~3 – PERIOD3~0 : 周期** (Period)

このﾋﾞｯﾄ領域は各割り込み間のRTCｸﾛｯｸ周期数を選びます。

値	0000	0001	0010	0011	0100	0101	0110	0111
名称	OFF	CYC4	CYC8	CYC16	CYC32	CYC64	CYC128	CYC256
説明	割り込みなし	4周期	8周期	16周期	32周期	64周期	128周期	256周期
値	1000	1001	1010	1011	1100	1101	1110	1111
名称	CYC512	CYC1024	CYC2048	DIV4096	CYC8192	CYC16384	CYC32768	–
説明	512周期	1024周期	2048周期	4096周期	8192周期	16384周期	32768周期	(予約)

● **ﾋﾞｯﾄ0 – PITEN : 周期割り込み計時器許可** (Periodic Interrupt Timer Enable)

このﾋﾞｯﾄへの'1'書き込みは周期割り込み計時器を許可します。

```
RTC.PITCTRLA = RTC_PERIOD_CYC32768_gc | RTC_PITEN_bm;
```

割り込みが起こるには全体割り込みも許可されなければなりません。

```
sei();
```

RTC PIT用割り込み処理ﾙｰﾁﾝ(ISR:Interrupt Service Routine)は下の例でLEDを交互切り替えします。

```
ISR(RTC_PIT_vect)
{
    RTC.PITINTFLAGS = RTC_PI_bm;
    LED0_toggle();
}
```

注: RTC.PITINTFLAGSのPIﾋﾞｯﾄはISR関数内でそれに'1'を書くことによって解除(0)されなければなりません。

GitHubでｺｰﾄﾞ例を見てください。
貯蔵庫を閲覧するにはｸﾘｯｸしてください。

助言: 完全なｺｰﾄﾞ例は「**追補**」章でも利用可能です。

5. 休止からRTC PIT起き上がり

PIT割り込みはCPUを休止から起こすことができます。

休止動作形態はSLPCTRL.CTRLAﾚｼﾞｽﾀで構成設定されます。休止機能はSLPCTRL.CTRLAﾚｼﾞｽﾀのSENﾋﾞｯﾄを設定(1)することによって許可されます。

図5-1. SLPCTRL.CTRLA – 休止動作形態とSENﾋﾞｯﾄ設定

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
						SMODE1,0		SEN
ｱｸｾｽ種別	R	R	R	R	R	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

● **ﾋﾞｯﾄ2,1 – SMODE1,0 : 休止動作形態** (Sleep Mode)

これらのﾋﾞｯﾄ書き込みは休止許可(SEN)ﾋﾞｯﾄが'1'を書かれ、SLEEP命令が実行される時に移行される休止動作形態を選びます。

値	0 0	0 1	1 0	1 1
名称	IDLE	STANDBY	PDOWN	–
説明	ｱｲﾄﾞﾙ休止動作許可	ｽﾀﾝﾊﾞｲ休止動作許可	ﾊﾟﾜｰﾀﾞｳﾝ休止動作許可	(予約)

● **ﾋﾞｯﾄ0 – SEN : 休止許可** (Sleep Enable)

選択された休止動作にMCUを移行するためにSLEEP命令が実行される前に、このﾋﾞｯﾄは'1'を書かれなければなりません。

```
SLPCTRL.CTRLA |= SLPCTRL_SMODE_PDOWN_gc;
SLPCTRL.CTRLA |= SLPCTRL_SEN_bm;
```

以下の関数を呼ぶことによってCPUを休止に置くことができます。

```
sleep_cpu();
```

PIT割り込みはCPUを休止から起こします。割り込みが起こるには全体割り込みも許可されなければなりません。

```
sei();
```

RTC PIT用割り込み処理ﾙｰﾁﾝ(ISR:Interrupt Service Routine)は下の例でLEDを交互切り替えします。

```
ISR(RTC_PIT_vect)
{
    RTC.PITINTFLAGS = RTC_PI_bm;
    LED0_toggle();
}
```

注: RTC.PITINTFLAGSのPIﾋﾞｯﾄはISR関数内でそれに'1'を書くことによって解除(0)されなければなりません。

GitHubでｺｰﾄﾞ例を見てください。
貯蔵庫を閲覧するにはｸﾘｯｸしてください。

助言: 完全なｺｰﾄﾞ例は「**追補**」章でも利用可能です。

6. 参照

RTCとPITの動作形態についてより多くの情報は以下のﾘﾝｸで見つけることができます。

1. ATmega4809製品頁 : https://www.microchip.com/wwwproducts/en/ATMEGA4809
2. 'megaAVR® 0系手引書' (DS40002015)
3. 'megaAVR® 0系ATmega3209/4809 – 48ﾋﾟﾝ ﾃﾞｰﾀｼｰﾄ' (DS40002016)
4. ATmega4809 Xplained Proｳｪﾌﾞ頁 : https://www.microchip.com/developmenttools/ProductDetails/atmega4809-xpro

7. 追補

例7-1. RTC溢れ割り込みｺｰﾄﾞ例

```
/* RTC周期 */
#define RTC_EXAMPLE_PERIOD    (511)

#include <avr/io.h>
#include <avr/interrupt.h>

void RTC_init(void);
void LED0_init(void);
void LED0_toggle(void);

void RTC_init(void)
{
    uint8_t temp;

    /* 32.768kHz発振器初期化 */
    /* 発振器禁止 */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_ENABLE_bm;
    /* 保護されたﾚｼﾞｽﾀへの書き込み許可 */
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.XOSC32KCTRLA = temp;

    while (CLKCTRL.MCLKSTATUS & CLKCTRL_XOSC32KS_bm)
    {
        ; /* XOSC32KSが0になるまで待機 */
    }

    /* SEL=0(外部ｸﾘｽﾀﾙ使用) */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_SEL_bm;
    /* 保護されたﾚｼﾞｽﾀへの書き込み許可 */
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.XOSC32KCTRLA = temp;

    /* 発振器許可 */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp |= CLKCTRL_ENABLE_bm;
    /* 保護されたﾚｼﾞｽﾀへの書き込み許可 */
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.XOSC32KCTRLA = temp;

    /* RTC初期化 */
    while (RTC.STATUS > 0)
    {
        ; /* 全ﾚｼﾞｽﾀが同期されるまで待機 */
    }

    /* 周期設定 */
    RTC.PER = RTC_EXAMPLE_PERIOD;

    /* 32.768kHz外部ｸﾘｽﾀﾙ用発振器(XOSC32K) */
    RTC.CLKSEL = RTC_CLKSEL_TOSC32K_gc;

    /* ﾃﾞﾊﾞｯｸﾞで走行: 許可 */
    RTC.DBGCTRL |= RTC_DBGRUN_bm;

    RTC.CTRLA = RTC_PRESCALER_DIV32_gc    /* 32分周 */
                | RTC_RTCEN_bm            /* 許可: 許可 */
                | RTC_RUNSTDBY_bm;        /* ｽﾀﾝﾊﾞｲで走行: 許可 */
```

例7-1 (続き). RTC溢れ割り込みｺｰﾄﾞ例

```
    /* 溢れ割り込み許可 */
    RTC.INTCTRL |= RTC_OVF_bm;
}

void LED0_init(void)
{
    /* High (OFF)設定 */
    PORTB.OUT |= PIN5_bm;
    /* 出力設定 */
    PORTB.DIR |= PIN5_bm;
}

void LED0_toggle(void)
{
    PORTB.IN |= PIN5_bm;
}

ISR(RTC_CNT_vect)
{
    /* '1'書き込みによって割り込み要求ﾌﾗｸﾞ解除(0) */
    RTC.INTFLAGS = RTC_OVF_bm;

    LED0_toggle();
}

int main(void)
{
    LED0_init();
    RTC_init();

    /* 全体割り込み許可 */
    sei();

    while (1)
    {
    }
}
```

例7-2. RTC周期的割り込みｺｰﾄﾞ例

```
#include <avr/io.h>
#include <avr/interrupt.h>

void RTC_init(void);
void LED0_init(void);
void LED0_toggle(void);

void RTC_init(void)
{
    uint8_t temp;

    /* 32.768kHz発振器初期化 */
    /* 発振器禁止 */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_ENABLE_bm;
    /* 保護されたﾚｼﾞｽﾀへの書き込み許可 */
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.XOSC32KCTRLA = temp;
```

例7-2 (続き). RTC周期的割り込みｺｰﾄﾞ例

```
    while (CLKCTRL.MCLKSTATUS & CLKCTRL_XOSC32KS_bm)
    {
        ; /* XOSC32KSが0になるまで待機 */
    }

    /* SEL=0 (外部ｸﾘｽﾀﾙ使用) */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_SEL_bm;
    /* 保護されたﾚｼﾞｽﾀへの書き込み許可 */
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.XOSC32KCTRLA = temp;

    /* 発振器許可 */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp |= CLKCTRL_ENABLE_bm;
    /* 保護されたﾚｼﾞｽﾀへの書き込み許可 */
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.XOSC32KCTRLA = temp;

    /* RTC初期化 */
    while (RTC.STATUS > 0)
    {
        ; /* 全ﾚｼﾞｽﾀが同期されるまで待機 */
    }

    /* 32.768kHz外部ｸﾘｽﾀﾙ用発振器 (XOSC32K) */
    RTC.CLKSEL = RTC_CLKSEL_TOSC32K_gc;

    /* ﾃﾞﾊﾞｯｸﾞで走行: 許可 */
    RTC.DBGCTRL = RTC_DBGRUN_bm;

    RTC.PITINTCTRL = RTC_PI_bm;                 /* 周期的割り込み: 許可 */

    RTC.PITCTRLA = RTC_PERIOD_CYC32768_gc       /* 32768 RTCｸﾛｯｸ周期 */
                 | RTC_PITEN_bm;                /* 許可: 許可 */
}

void LED0_init(void)
{
    /* High (OFF)設定 */
    PORTB.OUT |= PIN5_bm;
    /* 出力設定 */
    PORTB.DIR |= PIN5_bm;
}

void LED0_toggle(void)
{
    PORTB.IN |= PIN5_bm;
}

ISR(RTC_PIT_vect)
{
    /* '1'書き込みによって割り込み要求ﾌﾗｸﾞ解除(0) */
    RTC.PITINTFLAGS = RTC_PI_bm;

    LED0_toggle();
}

int main(void)
{
```

例7-2 (続き). RTC周期的割り込みｺｰﾄﾞ例

```
    LED0_init();
    RTC_init();

    /* 全体割り込み許可 */
    sei();

    while (1)
    {
    }
}
```

例7-3. 休止からのRTC PIT起き上がりｺｰﾄﾞ例

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>

void RTC_init(void);
void LED0_init(void);
void LED0_toggle(void);
void SLPCTRL_init(void);

void RTC_init(void)
{
    uint8_t temp;

    /* 32.768kHz発振器初期化 */
    /* 発振器禁止 */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_ENABLE_bm;
    /* 保護されたﾚｼﾞｽﾀへの書き込み許可 */
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.XOSC32KCTRLA = temp;

    while (CLKCTRL.MCLKSTATUS & CLKCTRL_XOSC32KS_bm)
    {
        ; /* XOSC32KSが0になるまで待機 */
    }

    /* SEL=0 (外部ｸﾘｽﾀﾙ使用) */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp &= ~CLKCTRL_SEL_bm;
    /* 保護されたﾚｼﾞｽﾀへの書き込み許可 */
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.XOSC32KCTRLA = temp;

    /* 発振器許可 */
    temp = CLKCTRL.XOSC32KCTRLA;
    temp |= CLKCTRL_ENABLE_bm;
    /* 保護されたﾚｼﾞｽﾀへの書き込み許可 */
    CPU_CCP = CCP_IOREG_gc;
    CLKCTRL.XOSC32KCTRLA = temp;

    /* RTC初期化 */
    while (RTC.STATUS > 0)
    {
        ; /* 全ﾚｼﾞｽﾀが同期されるまで待機 */
    }

    /* 32.768kHz外部ｸﾘｽﾀﾙ用発振器 (XOSC32K) */
```

例7-3 (続き). 休止からのRTC PIT起き上がりｺｰﾄﾞ例

```
    RTC.CLKSEL = RTC_CLKSEL_TOSC32K_gc;

    /* ﾃﾞﾊﾞｯｸﾞで走行: 許可 */
    RTC.DBGCTRL = RTC_DBGRUN_bm;

    RTC.PITINTCTRL = RTC_PI_bm;                 /* 周期的割り込み: 許可 */

    RTC.PITCTRLA = RTC_PERIOD_CYC32768_gc       /* 32768 RTCｸﾛｯｸ周期 */
                 | RTC_PITEN_bm;                /* 許可: 許可 */
}

void LED0_init(void)
{
    /* High (OFF)設定 */
    PORTB.OUT |= PIN5_bm;
    /* 出力設定 */
    PORTB.DIR |= PIN5_bm;
}

void LED0_toggle(void)
{
    PORTB.IN |= PIN5_bm;
}

ISR(RTC_PIT_vect)
{
    /* '1'書き込みによって割り込み要求ﾌﾗｸﾞ解除(0) */
    RTC.PITINTFLAGS = RTC_PI_bm;

    LED0_toggle();
}

void SLPCTRL_init(void)
{
    SLPCTRL.CTRLA |= SLPCTRL_SMODE_PDOWN_gc;
    SLPCTRL.CTRLA |= SLPCTRL_SEN_bm;
}

int main(void)
{
    LED0_init();
    RTC_init();
    SLPCTRL_init();

    /* 全体割り込み許可 */
    sei();

    while (1)
    {
        /* CPUを休止に置きます。 */
        sleep_cpu();

        /* PIT割り込みがCPUを起こします。 */
    }
}
```

Microchipウェブ サイト

Microchipはhttp://www.microchip.com/で当社のウェブ サイト経由でのオンライン支援を提供します。このウェブ 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 ブラウザを用いてアクセスすることができ、ウェブ サイトは以下の情報を含みます。

- **製品支援** – データシートと障害情報、応用記述と試供プログラム、設計資源、使用者の手引きとハードウェア支援資料、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、オンライン検討グループ、Microchip相談役プログラム員一覧
- **Microshipの事業** – 製品選択器と注文の手引き、最新Microchip報道発表、セミナーとイベントの一覧、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ビス

Microchipのお客様通知サービスは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のウェブ 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でのウェブ サイトを通して利用できます。

Microchipデバイス コード保護機能

Microchipデバイスでの以下のコード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デ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。
- コード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のデ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はそれらのコード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ドの安全を保証することはできません。コード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。

コード保護は常に進化しています。Microchipは当社製品のコード保護機能を継続的に改善することを約束します。Microchipのコード保護機能を破る試みはデジタル 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

デバ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デバ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。

日本語© *HERO* 2019.

本技術概説はMicrochipのTB3213技術概説(DS90003213A-2019年1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 - ｳﾞｪﾙｽ** 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: 47-72884388 **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