タイマ/カウンタF型(TCF)での開始に際して

TB3341

序説

著者: Bogdan-Alexandru Mariniuc Microchip Technology Inc.

AVR® EB系統ﾃﾞﾊﾞｲｽは応用の範囲を網羅するためにそれらを許す強力な計時器を含みます。ﾀｲﾏ/ｶｳﾝﾀF型(TCF)の機能は周波数と波形の生成を含みます。

この計時器は非同期で、80MHzまでの外部ｸﾛｯｸ元に接続することができます。他の計時器と異なる主な特性の1つは数値制御発振器(NCO:Numerical Controller Oscillator)動作です。

この技術概説はNCO動作と24ﾋﾞｯﾄ分解能のようなこの計時器の独特な機能を強調して読者にTCF動作形態を習熟させます。機能のより良い理解のため、「参照」章でﾃﾞｰﾀｼｰﾄを調べてください。

この文書は以下のような2つの特別な使用事例を網羅します。

- **2つの一定ON時間PWM信号を生成**

  一定ON時間の結果を作成する可変ﾃﾞｭｰﾃｨ ｻｲｸﾙを持つ2つのPWM信号を生成するために計時器をNCOﾊﾟﾙｽ周波数動作で初期化

- **2つの可変周波数信号を生成**

  2つの可変周波数信号を生成するために計時器をNCO固定ﾃﾞｭｰﾃｨ ｻｲｸﾙ動作で初期化

注: この文書で記述した各使用事例に対して、AVR16EB32で開発した素の物とAVR16EB32で開発したMPLAB®ｺｰﾄﾞ構成部(MCC)で生成した物の2つのｺｰﾄﾞ例があります。

この技術概説で記述したのと同じ機能性を持つAVR16EB32用素のｺｰﾄﾞ例は以下のここで見つけてください。

MPLAB DISCOVERでｺｰﾄﾞ例を見るにはｸﾘｯｸしてください。

この技術概説で記述したのと同じ機能性を持つAVR16EB32用MCC Melody生成ｺｰﾄﾞ例は以下のここで見つけてください。

MPLAB DISCOVERでｺｰﾄﾞ例を見るにはｸﾘｯｸしてください。

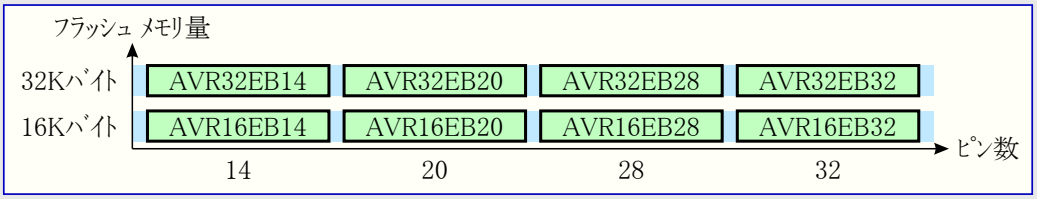
目次

1. 関連ﾃﾞﾊﾞｲｽ

本章はこの文書に関連するﾃﾞﾊﾞｲｽを一覧にします。下図はﾋﾟﾝ数の変種とﾒﾓﾘ量を展開して各種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、垂直方向移植はｺｰﾄﾞ変更なしで可能です。AVR EB系ﾃﾞﾊﾞｲｽでの下方移植はいくつかの周辺機能でのより少ない利用可能実体のためにｺｰﾄﾞ変更を必要とするかもしれません。
- 左への水平方向移植はﾋﾟﾝ数と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を持ちます。

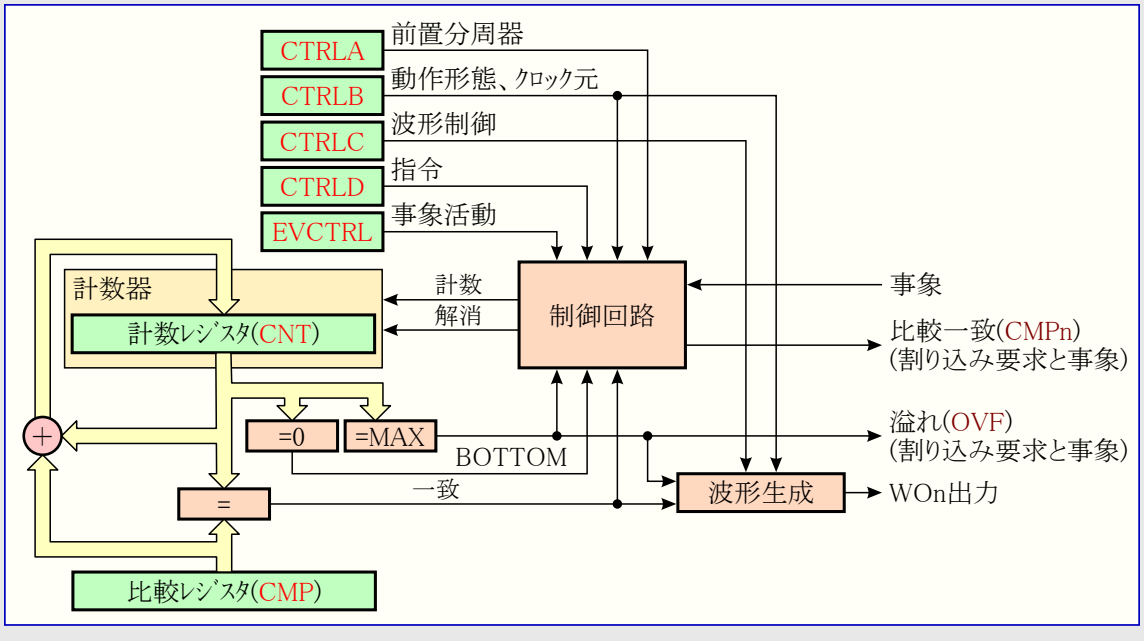
図1-1. AVR® EB系



2. 概要

TCFは基本計数器と、周波数生成、NCOﾊﾟﾙｽ周波数、NCO固定ﾃﾞｭｰﾃｨ ｻｲｸﾙ、8ﾋﾞｯﾄ ﾊﾟﾙｽ幅変調(PWM:Pulse-Width Modulation)のような各種波形生成動作に設定することができる制御論理回路から成ります。他のﾀｲﾏ/ｶｳﾝﾀ単位部と同様に、計時器溢れや比較一致のような特定条件に基づいて割り込みや事象を生成することができます。

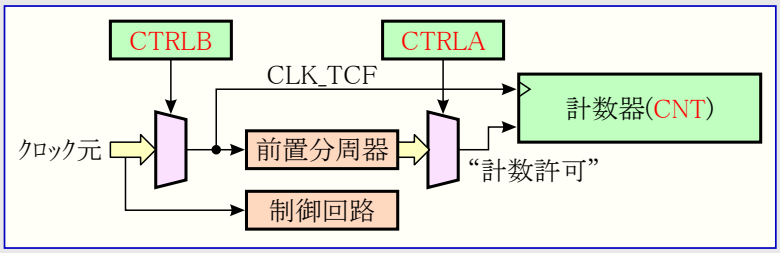
図2-1. 構成図



ﾀｲﾏ/ｶｳﾝﾀは周辺機能ｸﾛｯｸ(CLK_PER)、内部発振器、事象ｼｽﾃﾑ(EVSYS)を含む複数のｸﾛｯｸ元からｸﾛｯｸ駆動することができます。

制御B(TCFn.CTRLB)ﾚｼﾞｽﾀのｸﾛｯｸ選択(CLKSEL)ﾋﾞｯﾄ領域は前置分周器用入力として使われるｸﾛｯｸ元の1つを選びます。選んだｸﾛｯｸ元は7ﾋﾞｯﾄ前置分周器を使って最大128分周することができます。

図2-2. ｸﾛｯｸ選択



周波数生成動作でTCF使用

周波数波形生成動作ではCMPﾚｼﾞｽﾀが周期時間(T)を制御します。対応する波形出力はCNTとCMPのﾚｼﾞｽﾀ間の各比較一致で起動されます。この機能はTCAとTCBのﾀｲﾏ/ｶｳﾝﾀが持つものと同じですが、TCFは16ﾋﾞｯﾄ幅に変わって24ﾋﾞｯﾄの広い計数ﾚｼﾞｽﾀを持ちます。

8ﾋﾞｯﾄPWM動作でTCF使用

TCFは下位16ﾋﾞｯﾄ比較ﾚｼﾞｽﾀでのﾚｼﾞｽﾀ対(CMP0とCMP1)が独立した比較ﾚｼﾞｽﾀとして使われる8ﾋﾞｯﾄPWM動作で動くように構成設定することができます。8ﾋﾞｯﾄPWM機能はTCFへの有用な追加です。TCFの8ﾋﾞｯﾄPWM出力は単一傾斜だけです。

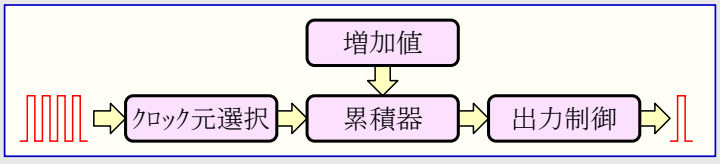
NCOﾊﾟﾙｽ周波数動作でTCF使用

NCO動作は出力信号を作成するのに累積器の溢れを使います。

累積器溢れは単一ｸﾛｯｸ ﾊﾟﾙｽや後置分周器増加よりもむしろ調整可能な増加値によって制御され、分割分解能が制限された前置分周器/後置分周器の分周値で変わらないため、簡単な計時器駆動計数器に対する優位性を提供します。NCOは固定のﾃﾞｭｰﾃｨｻｲｸﾙで周波数精度と高い分解能を必要とする応用に対して最も有用です。

NCOは固定値を繰り返し累積器に加算することによって動きます。加算は入力ｸﾛｯｸ速度で起きます。累積器は生のNCO出力の周期的なｷｬﾘｰで溢れます。これは最大累積器値に対する追加する値の比率によって入力ｸﾛｯｸを減らします。

図2-3. NCO論理回路



NCO出力はﾊﾟﾙｽを伸長するまたはﾌﾘｯﾌﾟﾌﾛｯﾌﾟを交互切り替えすることによって変更することができます。その後、変更したNCO出力は他の周辺機能に分配され、任意選択で入出力ﾋﾟﾝに出力します。累積器溢れは割り込みを生成することもできます。NCO周期は平均周波数を作成するのに離散段階で変更します。この出力は不確実性を低減するのにNCO出力を平均化する受信回路の能力に依存します。

波形周波数(f_{FRQ})は**式1.**によって定義されます。

注: 前置分周器(N)はこの動作形態で禁止されます。

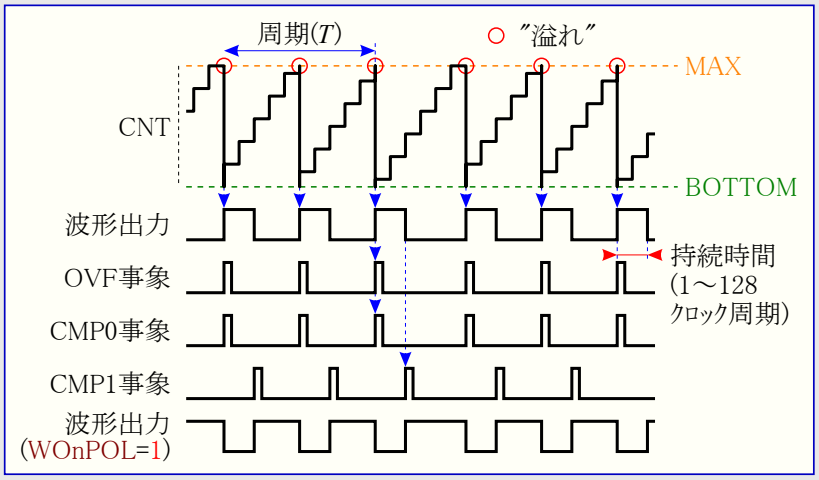
式1. FRQ周波数

$$f_{FRQ}[Hz] = \frac{TCF_{clock}[Hz] \times Increment}{2^{SIZE\_CNT}}$$

この動作形態では溢れ事象直後のｸﾛｯｸ上昇端で出力が活性になり、制御C(TCFn.CTRLC)ﾚｼﾞｽﾀの波形生成ﾊﾟﾙｽ長(WGPULSE)ﾋﾞｯﾄ領域によって決められる1～128ｸﾛｯｸ周期後に不活性になり、定期的な波形出力を与えます。

比較一致0(CMP0)事象は波形ﾊﾟﾙｽの開始毎に生成される一方で、比較一致1(CMP1)事象はﾊﾟﾙｽ終了時に生成されます。割り込みﾌﾗｸﾞはﾊﾟﾙｽ事象と同じ時に設定されます。

図2-4. NCOﾊﾟﾙｽ周波数波形生成



NCO固定ﾃﾞｭｰﾃｨ ｻｲｸﾙ周波数生成動作でTCF使用

NCO固定ﾃﾞｭｰﾃｨ ｻｲｸﾙ周波数波形生成動作ではTCFがNCO PWM動作のように動きますが、出力は累積器溢れの時毎に交互切り替えされます。

増加値が一定に留まるとすれば、これはﾊﾟﾙｽ周波数動作に対して半分の周波数で50%ﾃﾞｭｰﾃｨ ｻｲｸﾙを提供します。

CMP0とCMP1の事象は溢れで交互に生成されます。

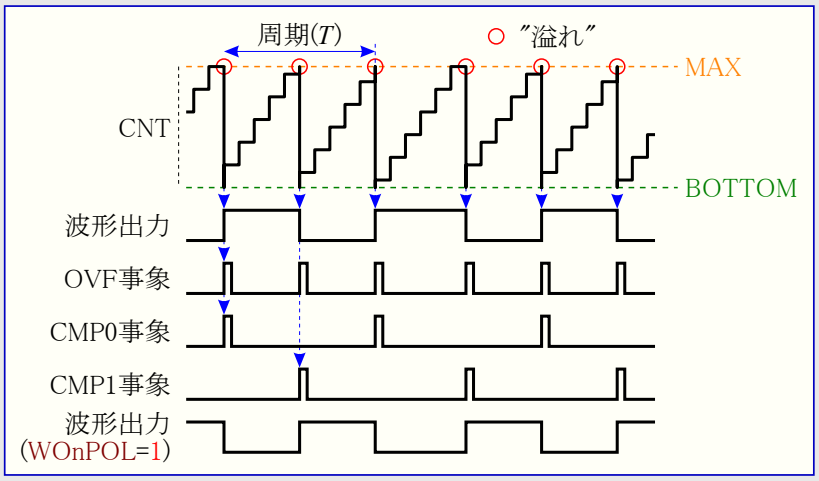
割り込みﾌﾗｸﾞは**式2.**で示されるようにﾊﾟﾙｽ事象と同じ時に設定されます。

注: Nは前置分周器を表します。

式2. FRQ周波数

$$f_{FRQ}[Hz] = \frac{TCF_{clock}[Hz] \times Increment}{2 \times N \times 2^{SIZE\_CNT}}$$

図25-6. NCO固定ﾃﾞｭｰﾃｨ ｻｲｸﾙ周波数波形生成



3. 可変ﾃﾞｭｰﾃｨ ｻｲｸﾙでの2つの固定周波数PWM信号生成

使用事例説明: 波形のﾃﾞｭｰﾃｨ ｻｲｸﾙを利用可能な全ての段階に対して増す間に固定周波数で比較ﾚｼﾞｽﾀでの溢れ事象を生成するようにTCFを構成設定してください。

結果: TCFは固定周波数でﾁｬﾈﾙ0とﾁｬﾈﾙ1での出力を生成すると同時に波形のﾃﾞｭｰﾃｨ ｻｲｸﾙが増します。

組み込み素の実装

1. **出力信号構成設定**

波形出力0と波形出力1を許可してください。

図3-1. CTRLCﾚｼﾞｽﾀ

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
		WGPULSE2~0			WO1POL	WO0POL	WO1EN	WO0EN
ｱｸｾｽ種別	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

CTRLCﾚｼﾞｽﾀは2重緩衝されます。CTRLCﾚｼﾞｽﾀが書かれる時毎に状態ﾚｼﾞｽﾀでCTRLCBUSYﾋﾞｯﾄが解除(0)されるのを繰り返しで待つことが推奨されます。uint8_t型の引数を取り、voidを返すTCF0_OutputsSetと呼ばれる関数を作成してください。

```
void TCF0_OutputsSet(uint8_t value)
{
    while((TCF0.STATUS & TCF_CTRLABUSY_bm) != 0){};
    TCF0.CTRLC = value;
}
```

2. **TCFｸﾛｯｸ構成設定**

AVR16EB32基板は既定で3.33MHzのｼｽﾃﾑ ｸﾛｯｸで走行します。TCF用に20MHzで走行するためにｼｽﾃﾑ ｸﾛｯｸ前置分周器が禁止されなければなりません。

void型の引数を取り、voidを返す関数を作成してください。

```
void CLOCK_Initialize(void)
{
    _PROTECTED_WRITE(CLKCTRL.MCLKCTRLB, 0x0);
}
```

図3-2. CTRLBﾚｼﾞｽﾀ

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	CMP1EV	CMP0EV	CLKSEL2~0			WGMODE2~0		
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	0	0	0	0	0	0	0	0

図3-3. ｸﾛｯｸ選択ﾋﾞｯﾄ遮蔽

● **ﾋﾞｯﾄ5~3 – CLKSEL2~0 : ｸﾛｯｸ選択** (Clock Select)

このﾋﾞｯﾄ領域はﾀｲﾏ/ｶｳﾝﾀ ｸﾛｯｸ元を制御し、ﾀｲﾏ/ｶｳﾝﾀが許可されている間変更することができません。

値	000	001	010	011	101	その他
名称	CLKPER	EVENT	OSCHF	OSC32K	PLL	–
説明	周辺機能ｸﾛｯｸ	事象端	内部高周波数発振器	内部32.768KHz発振器	位相固定化閉路	(予約)

TCFはそれから選ぶ様々なｸﾛｯｸ任意選択を提供します。この応用については20MHzで走行する周辺機能ｸﾛｯｸである既定の任意選択を選んでください。CTRLBﾚｼﾞｽﾀはｸﾛｯｸ任意選択を制御します。

TCF_CLKSEL_t型の引数を取り、voidを返す関数を作成してください。

```
void TCF0_ClockSet(TCF_CLKSEL_t config)
{
    TCF0.CTRLB |= config;
}
```

前置分周器はこの動作形態で禁止されます。

3. 波形生成動作形態構成設定

計時器が動く動作形態はCTRLBﾚｼﾞｽﾀを使って選ばれます。NCOPFが選ばれます。

図3-4. CTRLBﾚｼﾞｽﾀ

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	CMP1EV	CMP0EV	CLKSEL2~0			WGMODE2~0		
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	0	0	0	0	0	0	0	0

図3-5. 波形生成動作形態ﾋﾞｯﾄ遮蔽

● ﾋﾞｯﾄ2~0 – WGMODE2~0 : **波形生成動作** (Waveform Generation Mode)

このﾋﾞｯﾄ領域は波形生成動作を選びます。

値	0 0 0	0 0 1	0 1 0	1 1 1	その他
名称	FREQ	NCOPF	NCOFDC	PWM8	–
説明	周波数	数値制御発振器ﾊﾟﾙｽ周波数	数値制御発振器固定ﾃﾞｭｰﾃｨ ｻｲｸﾙ	8ﾋﾞｯﾄPWM	(予約)

TCF_WGMODE_t型の引数を取り、voidを返す関数を作成してください。

```
void TCF0_ModeSet(TCF_WGMODE_t mode)
{
    TCF0.CTRLB |= mode;
}
```

4. 周波数設定

TCFｸﾛｯｸ構成設定定後、CMPﾚｼﾞｽﾀを使って計時器が動く周波数(=125kHz)を選んでください。

図3-6. CMPﾚｼﾞｽﾀ

+$14	CMP	7～0	CMP7~0
+$15		15～8	CMP15~8
+$16		23～16	CMP23~16
+$17		31～24	

式1.を使い、増加値(Increment)は次のように計算することができます。

$$\text{Increment} = \frac{f_{FRQ}[\text{Hz}] \times 2^{\text{SIZE\_CNT}}}{\text{TCF}_{clock}[\text{Hz}]}$$

望む値を使い、次のようになります。

$$\text{Increment} = \frac{125{,}000 \times 16{,}777{,}216}{20{,}000{,}000} = 104{,}857.6$$

16進数の結果は0x0001999Aです。

上の式は次の式に変換することができます。

```
#define TCF0_NCOPL_HZ_TO_INCREMENT(HZ, F_CLOCK)    (uint32_t)(((float)(HZ) * 16777216.0) /
                                                    (float)(F_CLOCK) + 0.5)
```

CMPﾚｼﾞｽﾀは2重緩衝されます。CMPﾚｼﾞｽﾀが書かれる時毎に状態ﾚｼﾞｽﾀでCMP0BUSYﾋﾞｯﾄが解除(0)されるのを繰り返しで待つことが推奨されます。

```
void TCF0_CompareSet(uint32_t value)
{
    while((TCF0.STATUS & TCF_CMP0BUSY_bm) != 0){};
    TCF0.CMP = value;
}
```

```
void TCF0_CounterSet(uint32_t value)
{
    while((TCF0.STATUS & TCF_CNTBUSY_bm) != 0){};
    TCF0.CNT0 = (uint8_t)value;
}
```

5. 波形生成ﾊﾟﾙｽ長構成設定

図3-7. CTRLCﾚｼﾞｽﾀ

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
		WGPULSE2~0			WO1POL	WO0POL	WO1EN	WO0EN
アクセス種別	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
リセット値	0	0	0	0	0	0	0	0

図3-8. 波形生成ﾊﾟﾙｽ長ﾋﾞｯﾄ遮蔽

● **ﾋﾞｯﾄ6~4 – WGPULSE2~0 : 波形生成ﾊﾟﾙｽ長** (Waveform Generarion Pulse Length)

このﾋﾞｯﾄ領域はﾊﾟﾙｽ周波数動作で生成された波形のHigh時間を制御します。

値	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
名称	CLK1	CLK2	CLK4	CLK8	CLK16	CLK32	CLK64	CLK128
説明 (CLK_TCF周期数)	1	2	4	8	16	32	64	128

ﾊﾟﾙｽ長を50nsである単一ｸﾛｯｸ増加に設定してください。計時器が20MHzで動いているので1ｸﾛｯｸ周期は50nsかかります。1を20MHzでの除算は50nsと等価です。

```
void TCF0_NCO_PulseLengthSet(TCF_WGPULSE_t config)
{
    uint8_t temp;
    while((TCF0.STATUS & TCF_CTRLCBUSY_bm) != 0){};
    temp = (TCF0.CTRLC & ~TCF_WGPULSE_gm) | (config & TCF_WGPULSE_gm);
    TCF0.CTRLC = temp;
}
```

6. 計時器の開始と停止

図3-9. CTRLAﾚｼﾞｽﾀ

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	RUNSTDBY					PRESC2~0		ENABLE
アクセス種別	R/W	R	R	R	R/W	R/W	R/W	R/W
リセット値	0	0	0	0	0	0	0	0

```
void TCF0_Start(void)
{
    while((TCF0.STATUS & TCF_CTRLABUSY_bm) != 0){};
    TCF0.CTRLA |=  TCF_ENABLE_bm;
}
void TCF0_Stop(void)
{
    while((TCF0.STATUS & TCF_CTRLABUSY_bm) != 0){};
    TCF0.CTRLA &= ~TCF_ENABLE_bm;
}
```

TCFは以下の設定で初期化されます。

- 両出力を許可
- 周辺機能ｸﾛｯｸを選択
- NCOﾊﾟﾙｽ長動作を設定
- 125kHzに周波数を設定
- 1ｸﾛｯｸ周期の波形生成ﾊﾟﾙｽ長に設定
- 計時器を0から計時開始に設定

void型の引数を取り、voidを返す関数を作成してください。

```
void TCF0_Initialize(void)
{
    TCF0_OutputsSet(TCF_WO0EN_bm | TCF_WO1EN_bm);
    TCF0_ClockSet(TCF_CLKSEL_CLKPER_gc);
    TCF0_ModeSet(TCF_WGMODE_NCOPF_gc);
    TCF0_CompareSet(TCF0_NCOPL_HZ_TO_INCREMENT(125000, 20000000));
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK1_gc);
    TCF0_CounterSet(0);
}
```

util/delay.hﾍｯﾀﾞ ﾌｧｲﾙはF_CPU周波数を定義するのに必要とします。

```
#define F_CPU 20000000UL
#include <util/delay.h>                          /* util/delay.hﾍｯﾀﾞ ﾌｧｲﾙをｲﾝｸﾙｰﾄﾞ */
```

void型の引数を取り、voidを返す関数を作成してください。変更間に遅延を持つﾊﾟﾙｽ長変更にはNCO_Pulse_Length_Demo関数を使ってください。

```
void NCO_Pulse_Length_Demo(void)
{
    /* TCFを0から計数開始に構成設定 */
    TCF0_CounterSet(0);

    /* TCFを許可 */
    TCF0_Start();

    /* 20µs間遅延 */
    _delay_us(20);

    /* ﾊﾟﾙｽ長を2ｸﾛｯｸ周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK2_gc);

    /* 20µs間遅延 */
    _delay_us(20);

    /* ﾊﾟﾙｽ長を4ｸﾛｯｸ周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK4_gc);

    /* 20µs間遅延 */
    _delay_us(20);

    /* ﾊﾟﾙｽ長を8ｸﾛｯｸ周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK8_gc);

    /* 20µs間遅延 */
    _delay_us(20);

    /* ﾊﾟﾙｽ長を16ｸﾛｯｸ周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK16_gc);

    /* 20µs間遅延 */
    _delay_us(20);

    /* ﾊﾟﾙｽ長を32ｸﾛｯｸ周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK32_gc);

    /* 20µs間遅延 */
    _delay_us(20);

    /* ﾊﾟﾙｽ長を64ｸﾛｯｸ周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK64_gc);
```

```
    /* 20µs間遅延 */
    _delay_us(20);

    /* ﾊﾟﾙｽ長を128ｸﾛｯｸ周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK128_gc);

    /* 25µs間遅延 */
    _delay_us(25);

    /* 計時器を停止 */
    TCF0_Stop();

    /* ﾊﾟﾙｽ長を1ｸﾛｯｸ周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK1_gc);
}
```

main関数はこのように見えます。

```
void main(void)
{
    CLOCK_Initialize();
    TCF0_Initialize();
    while(1)
    {
        NCO_Pulse_Length_Demo();
        _delay_ms(20);
    }
}
```

MCC Melody実装

MPLABｺｰﾄﾞ構成部、MCC Melody(MCC Classicは不支援)を使ってこのﾌﾟﾛｼﾞｪｸﾄを生成するには次の手順に従ってください。

1. 新しいAVR16EB32用MPLAB® X IDEﾌﾟﾛｼﾞｪｸﾄを作成してください。
2. ﾂｰﾙﾊﾞｰからMCCを開き、そこでMCCﾌﾟﾗｸﾞｲﾝ ｲﾝｽﾄｰﾙでのより多くの情報を見つけてください。
3. MCC Content Manager(MCC内容管理)ｳｨｻﾞｰﾄﾞでMCC Melodyを選び、その後にFinishをｸﾘｯｸしてください。
4. Project Resources(ﾌﾟﾛｼﾞｪｸﾄ資源)からSystem(ｼｽﾃﾑ)へ行き、CLKCTRL窓をｸﾘｯｸし、前置分周器を禁止してください。
5. Device Resources(ﾃﾞﾊﾞｲｽ資源)からDrivers(ﾄﾞﾗｲﾊﾞ)へ行き、Timer(計時器)窓をｸﾘｯｸし、TCF単位部を追加し、その後に以下の構成設定を行ってください。
   - Clock Divider(ｸﾛｯｸ分周器): (既定で)System clock(ｼｽﾃﾑ ｸﾛｯｸ)、(分周器は1 - System clockです)。
   - Waveform Generation Mode(波形生成動作形態) : NCO pulswe-Length mode(NCOﾊﾟﾙｽ長動作)
   - Waveform Generation Pulse Length(波形生成ﾊﾟﾙｽ長) : 1 Clock Period(1ｸﾛｯｸ周期)
   - Requested Period[s](要求周期) : 0.000008
   - Waveform Output n(波形出力n) : 波形出力0と波形出力1に対するEnable(許可)列で枠をﾁｪｯｸしてください。
6. Pin Grid View(ﾋﾟﾝ格子表示)でPA0とPA1を選んでください。Enable(許可)列での枠とWaveform Output n(波形出力n)がﾁｪｯｸされていると、ﾋﾟﾝも固定化されます。PORTを変更するには、Pin Grid Viewで別のPORTからﾋﾟﾝをｸﾘｯｸしてください。
7. Project Resources(ﾌﾟﾛｼﾞｪｸﾄ資源)ｳｨﾝﾄﾞｳで、指定したﾄﾞﾗｲﾊﾞと構成設定の全てをMCCが生成するようにGenerate(生成)釦をｸﾘｯｸしてください。
8. 以下のようにmain.cﾌｧｲﾙを編集してください。

   util/delay.hﾍｯﾀﾞ ﾌｧｲﾙをｲﾝｸﾙｰﾄﾞしてください。

```
#include <util/delay.h>
```

   NCO_Pulse_Length_Demoと呼ばれる関数を作成してください。ﾊﾟﾙｽ長を変更するのにTCF0_NCO_PulseLengthSet関数を使ってください。

   このｺｰﾄﾞは次のとおりです。

```
void NCO_Pulse_Length_Demo(void)
{
    /* TCFを0から計数開始に構成設定 */
    TCF0_CounterSet(0);
```

```
    /* TCFを許可 */
    TCF0_Start();

    /* 20µs間遅延 */
    _delay_us(20);

    /* パルス長を2クロック周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK2_gc);

    /* 20µs間遅延 */
    _delay_us(20);

    /* パルス長を4クロック周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK4_gc);

    /* 20µs間遅延 */
    _delay_us(20);

    /* パルス長を8クロック周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK8_gc);

    /* 20µs間遅延 */
    _delay_us(20);

    /* パルス長を16クロック周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK16_gc);

    /* 20µs間遅延 */
    _delay_us(20);

    /* パルス長を32クロック周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK32_gc);

    /* 20µs間遅延 */
    _delay_us(20);

    /* パルス長を64クロック周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK64_gc);

    /* 20µs間遅延 */
    _delay_us(20);

    /* パルス長を128クロック周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK128_gc);

    /* 25µs間遅延 */
    _delay_us(25);

    /* 計時器を停止 */
    TCF0_Stop();

    /* パルス長を1クロック周期に構成設定 */
    TCF0_NCO_PulseLengthSet(TCF_WGPULSE_CLK1_gc);
}
```

9. main関数はこのように見えます。

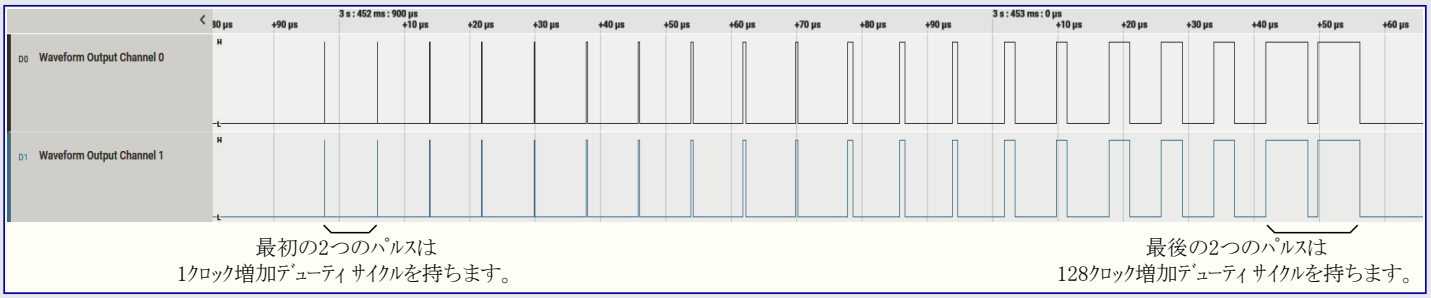
```
int main(void)
{
    SYSTEM_Initialize();

    while(1)
```

```
    {
        NCO_Pulse_Length_Demo();
        _delay_ms(20);
    }
}
```

10. ﾌﾟﾛｼﾞｪｸﾄを書き込んで実行してください。
11. 結果:

TCFは1～128ｸﾛｯｸ周期の範囲の可変持続時間で125kHzの周波数のPWMを持つ2つの同じ信号を生成します。この場合、1ｸﾛｯｸ周期は50nsかかります。

図3-10. 結果



MPLAB DISCOVERでｺｰﾄﾞ例を見るにはｸﾘｯｸしてください。

4. NCO固定ﾃﾞｭｰﾃｨ ｻｲｸﾙ波形生成で2つの可変周波数信号生成

使用事例説明: 10Hz～100kHzの周波数範囲で比較ﾚｼﾞｽﾀでの溢れ事象を生成するようにTCFを構成設定してください。

結果: TCFは累積器溢れの時毎に交互切り替えする周波数の範囲でﾁｬﾈﾙ0とﾁｬﾈﾙ1での出力を生成します。

組み込み素の実装

1. **出力信号構成設定**

波形出力0と波形出力1を許可してください。

図4-1. CTRLCﾚｼﾞｽﾀ

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
		WGPULSE2~0			WO1POL	WO0POL	WO1EN	WO0EN
ｱｸｾｽ種別	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

CTRLCﾚｼﾞｽﾀは2重緩衝されます。CTRLCﾚｼﾞｽﾀが書かれる時毎に状態ﾚｼﾞｽﾀでCTRLCBUSYﾋﾞｯﾄが解除(0)されるのを繰り返して待つことが推奨されます。uint8_t型の引数を取り、voidを返すTCF0_OutputsSetと呼ばれる関数を作成してください。

```
void TCF0_OutputsSet(uint8_t value)
{
    while((TCF0.STATUS & TCF_CTRLABUSY_bm) != 0){};
    TCF0.CTRLC = value;
}
```

2. **TCFｸﾛｯｸ構成設定**

AVR16EB32基板は既定で3.33MHzのｼｽﾃﾑ ｸﾛｯｸで走行します。TCF用に20MHzで走行するためにｼｽﾃﾑ ｸﾛｯｸ前置分周器が禁止されなければなりません。

void型の引数を取り、voidを返す関数を作成してください。

```
void CLOCK_Initialize(void)
{
    _PROTECTED_WRITE(CLKCTRL.MCLKCTRLB, 0x0);
}
```

図4-2. CTRLBﾚｼﾞｽﾀ

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	CMP1EV	CMP0EV	CLKSEL2~0			WGMODE2~0		
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	0	0	0	0	0	0	0	0

図4-3. ｸﾛｯｸ選択ﾋﾞｯﾄ遮蔽

● **ﾋﾞｯﾄ5~3 – CLKSEL2~0 : ｸﾛｯｸ選択** (Clock Select)

このﾋﾞｯﾄ領域はﾀｲﾏ/ｶｳﾝﾀ ｸﾛｯｸ元を制御し、ﾀｲﾏ/ｶｳﾝﾀが許可されている間変更することができません。

値	0 0 0	0 0 1	0 1 0	0 1 1	1 0 1	その他
名称	CLKPER	EVENT	OSCHF	OSC32K	PLL	–
説明	周辺機能ｸﾛｯｸ	事象端	内部高周波数発振器	内部32.768KHz発振器	位相固定化閉路	(予約)

TCFはそれから選ぶ様々なｸﾛｯｸ任意選択を提供します。この応用については20MHzで走行する周辺機能ｸﾛｯｸである既定の任意選択を選んでください。CTRLBﾚｼﾞｽﾀはｸﾛｯｸ任意選択を制御します。

TCF_CLKSEL_t型の引数を取り、voidを返す関数を作成してください。

```
void TCF0_ClockSet(TCF_CLKSEL_t config)
{
    TCF0.CTRLB |= config;
}
```

3. 前置分周器選択

図4-4. CTRLAﾚｼﾞｽﾀ

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	RUNSTDBY					PRESC2~0		ENABLE
アクセス種別	R/W	R	R	R	R/W	R/W	R/W	R/W
リセット値	0	0	0	0	0	0	0	0

CTRLAﾚｼﾞｽﾀは2重緩衝されます。CTRLAﾚｼﾞｽﾀが書かれる時毎に状態ﾚｼﾞｽﾀでCTRLABUSYﾋﾞｯﾄが解除(0)されるのを繰り返して待つことが推奨されます。TCF_PRESC_t型の引数を取り、voidを返す関数を作成してください。

```
void TCF0_PrescalerSet(TCF_PRESC_t config)
{
    while((TCF0.STATUS & TCF_CTRLABUSY_bm) != 0){};
    TCF0.CTRLA  |= config;
}
```

4. 波形生成動作形態構成設定

計時器が動く動作形態はCTRLBﾚｼﾞｽﾀを使って選ばれます。NCOFDCが選ばれます。

図4-5. CTRLBﾚｼﾞｽﾀ

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	CMP1EV	CMP0EV		CLKSEL2~0			WGMODE2~0	
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	0	0	0	0	0	0	0	0

図4-6. 波形生成動作形態ﾋﾞｯﾄ遮蔽

● ﾋﾞｯﾄ2~0 – WGMODE2~0：波形生成動作 (Waveform Generation Mode)

このﾋﾞｯﾄ領域は波形生成動作を選びます。

値	0 0 0	0 0 1	0 1 0	1 1 1	その他
名称	FREQ	NCOPF	NCOFDC	PWM8	–
説明	周波数	数値制御発振器ﾊﾟﾙｽ周波数	数値制御発振器固定ﾃﾞｭｰﾃｨ ｻｲｸﾙ	8ﾋﾞｯﾄPWM	(予約)

TCF_WGMODE_t型の引数を取り、voidを返す関数を作成してください。

```
void TCF0_ModeSet(TCF_WGMODE_t mode)
{
    TCF0.CTRLB  |= mode;
}
```

5. CMPﾚｼﾞｽﾀ設定

式2.を使い、増加値(Increment)は次のように計算することができます。

$$\text{Increment} = \frac{f_{FRQ}[\text{Hz}] \times 2^{\text{SIZE\_CNT}} \times N}{\text{TCF}_{\text{clock}}[\text{Hz}]}$$

望む値を使い、次のようになります。

$$\text{Increment} = \frac{10 \times 16{,}777{,}216 \times 2 \times 1}{20{,}000{,}000} = 16.777216$$

結果は16進数で0x00000011です。

上の式は次の式に変換することができます。

```
#define TCF0_NCOFD_HZ_TO_INCREMENT(HZ, F_CLOCK, TCF0_PRESCALER)    (uint32_t)(((float)(HZ)
                                    * 33554432.0 * (TCF0_PRESCALER)) / ((float)(F_CLOCK)) + 0.5)
```

計時器が開始される前にCMPが$11の値で書かれます。

図4-7. CMPﾚｼﾞｽﾀ

+$14	CMP	7～0	CMP7～0
+$15		15～8	CMP15～8
+$16		23～16	CMP23～16
+$17		31～24	

CMPﾚｼﾞｽﾀは2重緩衝されます。CMPﾚｼﾞｽﾀが書かれる時毎に状態ﾚｼﾞｽﾀでCMP0BUSYﾋﾞｯﾄが解除(0)されるのを繰り返しで待つことが推奨されます。

uint32_t型の引数を取り、voidを返す関数を作成してください。

```
void TCF0_CompareSet(uint32_t value)
{
    while((TCF0.STATUS & TCF_CMP0BUSY_bm) != 0){};
    TCF0.CMP = value;
}
```

CNTﾚｼﾞｽﾀは2重緩衝されます。CNTﾚｼﾞｽﾀが書かれる時毎に状態ﾚｼﾞｽﾀでCNTBUSYﾋﾞｯﾄが解除(0)されるのを繰り返しで待つことが推奨されます。

uint32_t型の引数を取り、voidを返す関数を作成してください。

```
void TCF0_CounterSet(uint32_t value)
{
    while((TCF0.STATUS & TCF_CNTBUSY_bm) != 0){};
    TCF0.CNT0 = (uint8_t)value;
}
```

6. 計時器の開始と停止

図4-8. CTRLAﾚｼﾞｽﾀ

ﾋﾞｯﾄ	7	6	5	4	3	2	1	0
	RUNSTDBY				PRESC2~0			ENABLE
ｱｸｾｽ種別	R/W	R	R	R	R/W	R/W	R/W	R/W
ﾘｾｯﾄ値	0	0	0	0	0	0	0	0

CTRLAﾚｼﾞｽﾀは2重緩衝されます。CTRLAﾚｼﾞｽﾀが書かれる時毎に状態ﾚｼﾞｽﾀでCTRLAUSYﾋﾞｯﾄが解除(0)されるのを繰り返しで待つことが推奨されます。

void型の引数を取り、voidを返す関数を作成してください。

```
void TCF0_Start(void)
{
    while((TCF0.STATUS & TCF_CTRLABUSY_bm) != 0){};
    TCF0.CTRLA |=  TCF_ENABLE_bm;
}
```

void型の引数を取り、voidを返す関数を作成してください。

```
void TCF0_Stop(void)
{
    while((TCF0.STATUS & TCF_CTRLABUSY_bm) != 0){};
    TCF0.CTRLA &= ~TCF_ENABLE_bm;
}
```

TCFは以下の設定で初期化されます。

- 両出力を許可
- 周辺機能ｸﾛｯｸを選択
- NCO固定ﾃﾞｭｰﾃｨ ｻｲｸﾙ動作を設定
- 1ｸﾛｯｸ周期の波形生成ﾊﾟﾙｽ長に設定
- 10Hzの周波数に設定
- 計時器を0から計時開始に設定

void型の引数を取り、voidを返す関数を作成してください。

```
void TCF0_Initialize(void)
{
    TCF0_OutputsSet(TCF_WO0EN_bm | TCF_WO1EN_bm);
    TCF0_ClockSet(TCF_CLKSEL_CLKPER_gc);
    TCF0_PrescalerSet(TCF_PRESC_DIV1_gc);
    TCF0_ModeSet(TCF_WGMODE_NCOFDC_gc);
    TCF0_CounterSet(0);
    TCF0_CompareSet(TCF0_NCOFD_HZ_TO_INCREMENT(10, 20000000, 1));
}
```

util/delay.hﾍｯﾀﾞ ﾌｧｲﾙはF_CPU周波数を定義するのに必要とします。

```
#define F_CPU 20000000UL
#include <util/delay.h>                    /* util/delay.hﾍｯﾀﾞ ﾌｧｲﾙをｲﾝｸﾙｰﾄﾞ */
```

void型の引数を取り、voidを返す関数を作成してください。周波数を変更するのにTCF0_CompareSet関数を使ってください。

```
void NCO_Fixed_DutyCycle_Demo(void)
{
    /* TCFを0から計数開始に構成設定 */
    TCF0_CounterSet(0);

    /* TCFを許可 */
    TCF0_Start();

    /* 600ms間遅延 */
    _delay_ms(600);

    /* CMPﾚｼﾞｽﾀを100Hzの周波数に設定 */
    TCF0_CompareSet(TCF0_NCOFD_HZ_TO_INCREMENT(100, 20000000, 1));

    /* 60ms間遅延 */
    _delay_ms(60);

    /* CMPﾚｼﾞｽﾀを1kHzの周波数に設定 */
    TCF0_CompareSet(TCF0_NCOFD_HZ_TO_INCREMENT(1000, 20000000, 1));

    /* 6ms間遅延 */
    _delay_ms(6);

    /* CMPﾚｼﾞｽﾀを10kHzの周波数に設定 */
    TCF0_CompareSet(TCF0_NCOFD_HZ_TO_INCREMENT(10000, 20000000, 1));

    /* 600µs間遅延 */
    _delay_us(600);

    /* CMPﾚｼﾞｽﾀを100kHzの周波数に設定 */
    TCF0_CompareSet(TCF0_NCOFD_HZ_TO_INCREMENT(100000, 20000000, 1));

    /* 60µs間遅延 */
    _delay_us(60);

    /* TCFを停止 */
    TCF0_Stop();

    /* CMPﾚｼﾞｽﾀを10Hzの周波数に設定 */
    TCF0_CompareSet(TCF0_NCOFD_HZ_TO_INCREMENT(10, 20000000, 1));
}
```

main関数はこのように見えます。

```
void main(void)
{
    CLOCK_Initialize();
    TCF0_Initialize();
    while(1)
    {
        NCO_Fixed_DutyCycle_Demo();
        _delay_ms(1000);
    }
}
```

MCC Melody実装

MPLABｺｰﾄﾞ構成部、MCC Melody(MCC Classicは不支援)を使ってこのﾌﾟﾛｼﾞｪｸﾄを生成するには次の手順に従ってください。

1. 新しいAVR16EB32用MPLAB® X IDEﾌﾟﾛｼﾞｪｸﾄを作成してください。
2. ﾂｰﾙﾊﾞｰからMCCを開き、そこでMCCﾌﾟﾗｸﾞｲﾝ ｲﾝｽﾄｰﾙでのより多くの情報を見つけてください。
3. MCC Content Manager(MCC内容管理)ｳｨｻﾞｰﾄﾞでMCC Melodyを選び、その後にFinishをｸﾘｯｸしてください。
4. Project Resources(ﾌﾟﾛｼﾞｪｸﾄ資源)からSystem(ｼｽﾃﾑ)へ行き、CLKCTRL窓をｸﾘｯｸし、前置分周器を禁止してください。
5. Device Resources(ﾃﾞﾊﾞｲｽ資源)からDrivers(ﾄﾞﾗｲﾊﾞ)へ行き、Timer(計時器)窓をｸﾘｯｸし、TCF単位部を追加し、その後に以下の構成設定を行ってください。
   - Clock Divider(ｸﾛｯｸ分周器): (既定で)System clock(ｼｽﾃﾑ ｸﾛｯｸ)、(分周器は1 - System clockです)。
   - Waveform Generation Mode(波形生成動作形態) : NCO Fixed Duty-Cycle mode(NCO固定ﾃﾞｭｰﾃｨ ｻｲｸﾙ動作)
   - Requested Period[s](要求周期) : 0.1
   - Waveform Output n(波形出力n) : 波形出力0と波形出力1に対するEnable(許可)列で枠をﾁｪｯｸしてください。
6. Pin Grid View(ﾋﾟﾝ格子表示)でPA0とPA1を選んでください。Enable(許可)列での枠とWaveform Output n(波形出力n)がﾁｪｯｸされていると、ﾋﾟﾝも固定化されます。PORTを変更するには、Pin Grid Viewで別のPORTからﾋﾟﾝをｸﾘｯｸしてください。
7. Project Resources(ﾌﾟﾛｼﾞｪｸﾄ資源)ｳｨﾝﾄﾞｳで、指定したﾄﾞﾗｲﾊﾞと構成設定の全てをMCCが生成するようにGenerate(生成)釦をｸﾘｯｸしてください。
8. 以下のようにmain.cﾌｧｲﾙを編集してください。

   util/delay.hﾍｯﾀﾞ ﾌｧｲﾙをｲﾝｸﾙｰﾄﾞしてください。

   ```
   #include <util/delay.h>
   ```

   NCO_Fixed_DutyCycle_Demoと呼ばれる関数を作成してください。周波数を変更するのにTCF0_NCO_CompareSet関数を使ってください。

   このｺｰﾄﾞは次のとおりです。

   ```
   void NCO_Fixed_DutyCycle_Demo(void)
   {
       /* TCFを0から計数開始に構成設定 */
       TCF0_CounterSet(0);

       /* TCFを許可 */
       TCF0_Start();

       /* 600ms間遅延 */
       _delay_ms(600);

       /* CMPﾚｼﾞｽﾀを100Hzの周波数に設定 */
       TCF0_CompareSet(TCF0_NCOFD_HZ_TO_INCREMENT(100, 20000000, 1));

       /* 60ms間遅延 */
       _delay_ms(60);

       /* CMPﾚｼﾞｽﾀを1kHzの周波数に設定 */
       TCF0_CompareSet(TCF0_NCOFD_HZ_TO_INCREMENT(1000, 20000000, 1));

       /* 6ms間遅延 */
       _delay_ms(6);
   ```

```
    /* CMPﾚｼﾞｽﾀを10kHzの周波数に設定 */
    TCF0_CompareSet(TCF0_NCOFD_HZ_TO_INCREMENT(10000, 20000000, 1));

    /* 600µs間遅延 */
    _delay_us(600);

    /* CMPﾚｼﾞｽﾀを100kHzの周波数に設定 */
    TCF0_CompareSet(TCF0_NCOFD_HZ_TO_INCREMENT(100000, 20000000, 1));

    /* 60µs間遅延 */
    _delay_us(60);

    /* TCFを停止 */
    TCF0_Stop();

    /* CMPﾚｼﾞｽﾀを10Hzの周波数に設定 */
    TCF0_CompareSet(TCF0_NCOFD_HZ_TO_INCREMENT(10, 20000000, 1));
}
```

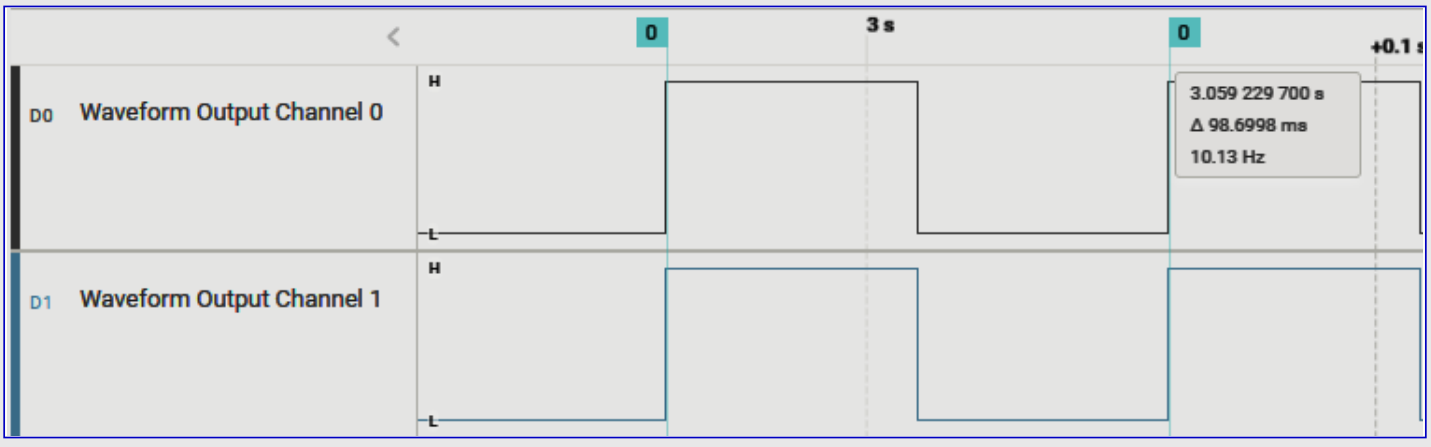
9. main.cﾌｧｲﾙはこのように見えます。

```
int main(void)
{
    SYSTEM_Initialize();

    while(1)
    {
        NCO_Fixed_DutyCycle_Demo();
        _delay_ms(1000);
    }
}
```

10. ﾌﾟﾛｼﾞｪｸﾄを書き込んで実行してください。

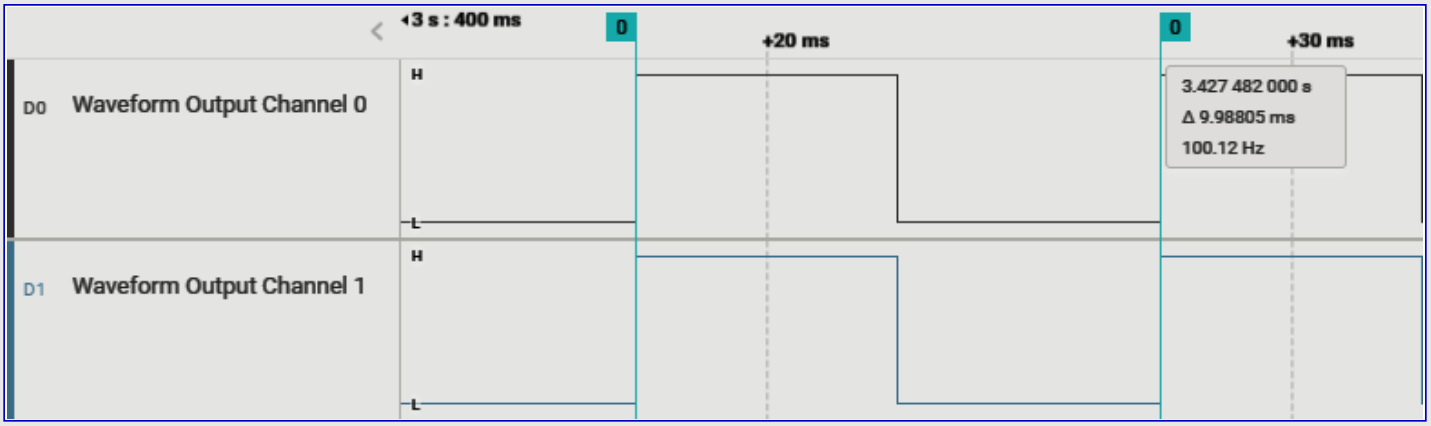
結果1: 10Hzの周波数と50%ﾃﾞｭｰﾃｨ ｻｲｸﾙで2つの同じ信号が生成されます。

図4-9. 結果



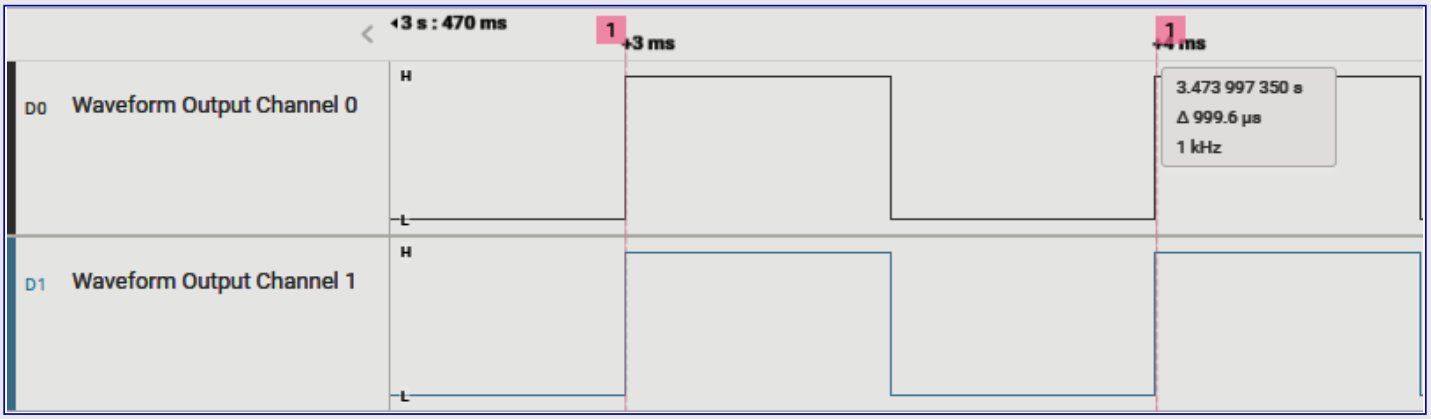
結果2: 100Hzの周波数と50%デューティ サイクルで2つの同じ信号が生成されます。

図4-10. 結果



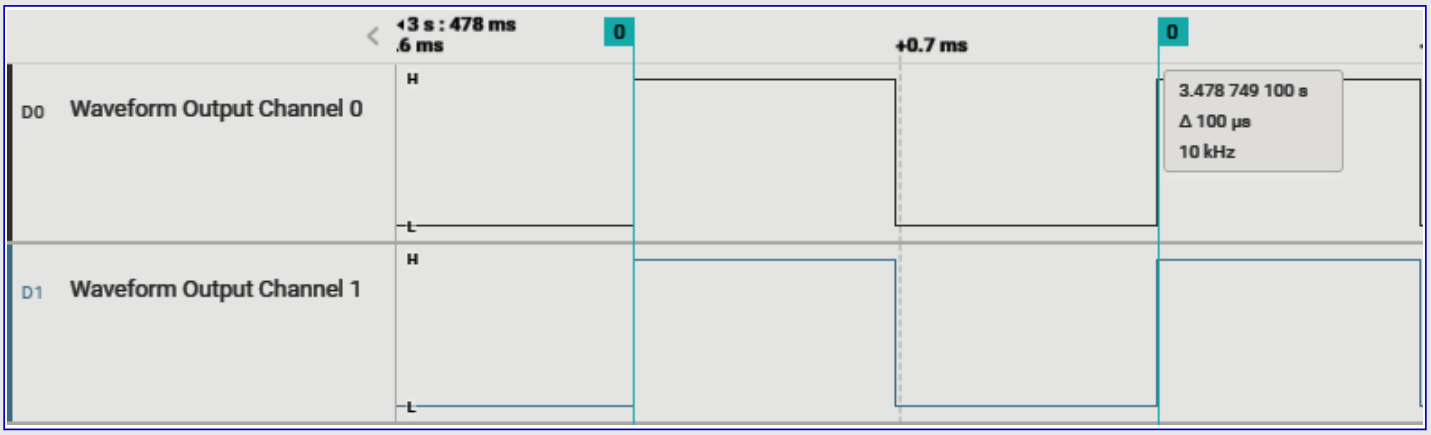
結果3: 1kHzの周波数と50%デューティ サイクルで2つの同じ信号が生成されます。

図4-11. 結果



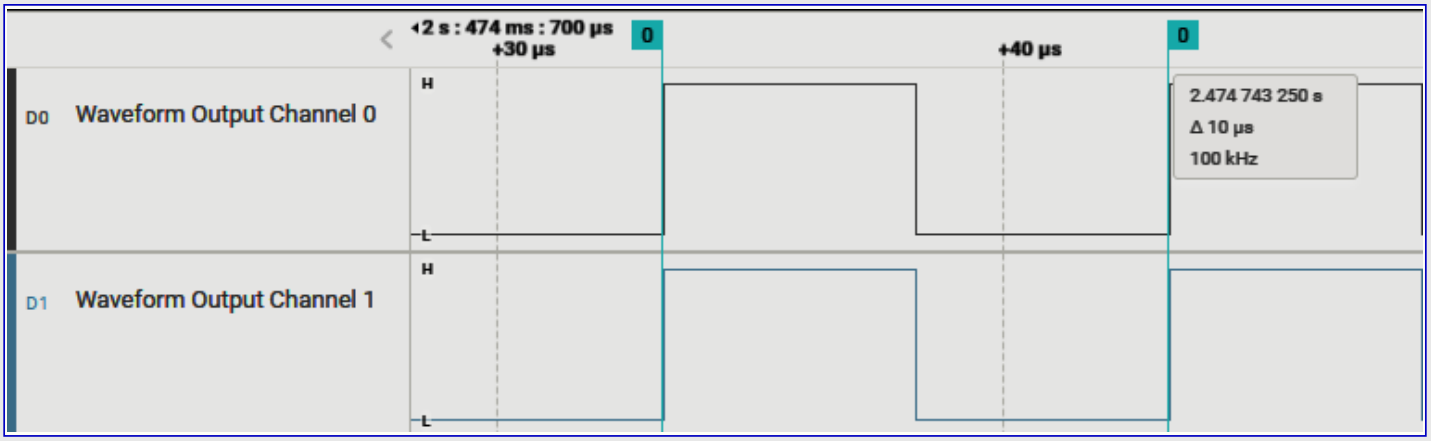
結果4: 10kHzの周波数と50%デューティ サイクルで2つの同じ信号が生成されます。

図4-12. 結果



結果2: 100kHzの周波数と50%ﾃﾞｭｰﾃｨ ｻｲｸﾙで2つの同じ信号が生成されます。

図4-13. 結果



MPLAB DISCOVERでｺｰﾄﾞ例を見るにはｸﾘｯｸしてください。

5. 参照

TCF動作形態についてより多くの情報に関しては以下のﾘﾝｸを使ってください。
これらのﾘﾝｸは不正確です。これらはｳｪﾌﾞ頁が一度実際に存在したら更新されなければなりません。

1. AVR16EB32製品頁
2. AVR16EB32 Curiosity Nano評価ｷｯﾄ
3. AVR16EB16/20/28/32 AVR® EB系統ﾃﾞｰﾀｼｰﾄ

6. 改訂履歴

文書改訂	日付	注釈
A	2023年10月	初版文書公開

Microchip情報

Microchipｳｪﾌﾞ ｻｲﾄ

Microchipは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のに使われます。利用可能な情報のいくつかは以下を含みます。

- **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設計協力課程会員一覧
- **Microchip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

製品変更通知ｻｰﾋﾞｽ

Microchipの製品変更通知ｻｰﾋﾞｽは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www.microchip.com/pcnへ行っ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
- 組み込み解決技術者(ESE:Embedded Solutions 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ESE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製品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は動作仕様内で意図した方法と通常条件下で使われる時に、その製品系統が安全であると考えます。
- Microchipはその知的所有権を尊重し、積極的に保護します。Microchip製品のｺｰﾄﾞ保護機能を侵害する試みは固く禁じられ、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。
- Microchipや他のどの半導体製造業者もそれのｺｰﾄﾞの安全を保証することはできません。ｺｰﾄﾞ保護は製品が"破ることができない"ことを当社が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。

法的通知

この刊行物と契約での情報は設計、試験、応用とのMicrochip製品の統合を含め、Microchip製品でだけ使えます。他の何れの方法でのこの情報の使用はこれらの条件に違反します。ﾃﾞﾊﾞｲｽ応用などに関す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。追加支援については最寄りのMicrochip営業所にお問い合わせ頂くか、www.microchip.com/en-us/support/design-help/client-support-servicesで追加支援を得てください。

この情報はMicrochipによって「現状そのまま」で提供されます。Microchipは非侵害、商品性、特定目的に対する適合性の何れの黙示的保証やその条件、品質、性能に関する保証を含め、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。

如何なる場合においても、Microchipは情報またはその使用に関連するあらゆる種類の間接的、特別的、懲罰的、偶発的または結果的な損失、損害、費用または経費に対して責任を負わないものとします。法律で認められている最大限の範囲で、情報またはその使用に関連する全ての請求に対するMicrochipの全責任は、もしあれば、情報のためにMicrochipへ直接支払った料金を超えないものとします。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

品質管理ｼｽﾃﾑ

Microchipの品質管理ｼｽﾃﾑに関する情報についてはwww.microchip.com/qualityを訪ねてください。

日本語© *HERO* 2024.

本技術概説はMicrochipのTB3341技術概説(DS90003341A-2023年10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: www.microchip.com/support ｳｪﾌﾞ ｱﾄﾞﾚｽ: www.microchip.com	**ｵｰｽﾄﾗﾘｱ - ｼﾄﾞﾆｰ** Tel: 61-2-9868-6733	**ｲﾝﾄﾞ - ﾊﾝｶﾞﾛｰﾙ** Tel: 91-80-3090-4444	**ｵｰｽﾄﾘｱ - ｳﾞｪﾙｽ** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393
ｱﾄﾗﾝﾀ Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455	**中国 - 北京** Tel: 86-10-8569-7000	**ｲﾝﾄﾞ - ﾆｭｰﾃﾞﾘｰ** Tel: 91-11-4160-8631	**ﾃﾞﾝﾏｰｸ - ｺﾍﾟﾝﾊｰｹﾞﾝ** Tel: 45-4485-5910 Fax: 45-4485-2829
ｵｰｽﾁﾝ TX Tel: 512-257-3370	**中国 - 成都** Tel: 86-28-8665-5511	**ｲﾝﾄﾞ - ﾌﾟﾈｰ** Tel: 91-20-4121-0141	**ﾌｨﾝﾗﾝﾄﾞ - ｴｽﾎﾟｰ** Tel: 358-9-4520-820
ﾎﾞｽﾄﾝ Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088	**中国 - 重慶** Tel: 86-23-8980-9588	**日本 - 大阪** Tel: 81-6-6152-7160	**ﾌﾗﾝｽ - ﾊﾟﾘ** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79
ｼｶｺﾞ Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075	**中国 - 東莞** Tel: 86-769-8702-9880	**日本 - 東京** Tel: 81-3-6880-3770	**ﾄﾞｲﾂ - ｶﾞﾙﾋﾝｸﾞ** Tel: 49-8931-9700
ﾀﾞﾗｽ Addison, TX Tel: 972-818-7423 Fax: 972-818-2924	**中国 - 広州** Tel: 86-20-8755-8029	**韓国 - 大邱** Tel: 82-53-744-4301	**ﾄﾞｲﾂ - ﾊｰﾝ** Tel: 49-2129-3766400
ﾃﾞﾄﾛｲﾄ Novi, MI Tel: 248-848-4000	**中国 - 杭州** Tel: 86-571-8792-8115	**韓国 - ソウル** Tel: 82-2-554-7200	**ﾄﾞｲﾂ - ﾊｲﾙﾌﾞﾛﾝ** Tel: 49-7131-72400
ﾋｭｰｽﾄﾝ TX Tel: 281-894-5983	**中国 - 香港特別行政区** Tel: 852-2943-5100	**ﾏﾚｰｼｱ - ｸｱﾗﾙﾝﾌﾟｰﾙ** Tel: 60-3-7651-7906	**ﾄﾞｲﾂ - ｶｰﾙｽﾙｰｴ** Tel: 49-721-625370
ｲﾝﾃﾞｱﾅﾎﾟﾘｽ Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380	**中国 - 南京** Tel: 86-25-8473-2460	**ﾏﾚｰｼｱ - ﾍﾟﾅﾝ** Tel: 60-4-227-8870	**ﾄﾞｲﾂ - ﾐｭﾝﾍﾝ** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44
ﾛｻﾝｾﾞﾙｽ Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800	**中国 - 青島** Tel: 86-532-8502-7355	**ﾌｨﾘﾋﾟﾝ - ﾏﾆﾗ** Tel: 63-2-634-9065	**ﾄﾞｲﾂ - ﾛｰｾﾞﾝﾊｲﾑ** Tel: 49-8031-354-560
ﾛｰﾘｰ NC Tel: 919-844-7510	**中国 - 上海** Tel: 86-21-3326-8000	**ｼﾝｶﾞﾎﾟｰﾙ** Tel: 65-6334-8870	**ｲｽﾗｴﾙ - ﾗｰﾅﾅ** Tel: 972-9-744-7705
ﾆｭｰﾖｰｸ NY Tel: 631-435-6000	**中国 - 瀋陽** Tel: 86-24-2334-2829	**台湾 - 新竹** Tel: 886-3-577-8366	**ｲﾀﾘｱ - ﾐﾗﾉ** Tel: 39-0331-742611 Fax: 39-0331-466781
ｻﾝﾎｾ CA Tel: 408-735-9110 Tel: 408-436-4270	**中国 - 深圳** Tel: 86-755-8864-2200	**台湾 - 高雄** Tel: 886-7-213-7830	**ｲﾀﾘｱ - ﾊﾟﾄﾞﾊﾞ** Tel: 39-049-7625286
ｶﾅﾀﾞ - ﾄﾛﾝﾄ Tel: 905-695-1980 Fax: 905-695-2078	**中国 - 蘇州** Tel: 86-186-6233-1526	**台湾 - 台北** Tel: 886-2-2508-8600	**ｵﾗﾝﾀﾞ - ﾃﾞﾙｰﾈﾝ** Tel: 31-416-690399 Fax: 31-416-690340
	中国 - 武漢 Tel: 86-27-5980-5300	**ﾀｲ - ﾊﾞﾝｺｸ** Tel: 66-2-694-1351	**ﾉﾙｳｪｰ - ﾄﾛﾝﾊｲﾑ** Tel: 47-72884388
	中国 - 西安 Tel: 86-29-8833-7252	**ﾍﾞﾄﾅﾑ - ﾎｰﾁﾐﾝ** Tel: 84-28-5448-2100	**ﾎﾟｰﾗﾝﾄﾞ - ﾜﾙｼｬﾜ** Tel: 48-22-3325737
	中国 - 廈門 Tel: 86-592-2388138		**ﾙｰﾏﾆｱ - ﾌﾞｶﾚｽﾄ** Tel: 40-21-407-87-50
	中国 - 珠海 Tel: 86-756-3210040		**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