Atmel AVR 8ビット マイクロ コントローラ

AVR130 : AVRタイマ/カウンタの初期設定と使用法

応用記述

序説

この応用記述はAtmel® AVR®の各種タイマ/カウンタの使用法を記述します。この資料の意図はタイマ/カウンタの一般概要を与え、それらの可能性を示し、それらを設定する方法を説明することです。コード例はこれをより明らかにし、他の応用に対する手引きとして使うことができます。タイマ/カウンタを説明するのにこの資料内の例としてAtmel ATmega328PBが使われます。

タイマ/カウンタの一般概要から始まる、各種の例はタイマ/カウンタがどう動くかと、それらがどう設定されるかを示すでしょう。経験済みの使用者は「**タイマ/カウンタの初期設定**」章で直接始められます。最終項はPWM動作の短い記述です。これら全ての例に対するCとアセンブリのコードを含むzipファイルがこの応用記述と共に入手可能です。それはAtmelのウェブサイトからダウンロードすることができます。

追加情報は特定デバイスに対するデータシートと応用記述で入手可能です。

特徴

- タイマ/カウンタ事象の説明
- タイマ/カウンタ事象の通知
- クロック任意選択
- 様々なタイマ/カウンタ用コード例
  - 溢れ割り込み
  - 捕獲入力割り込み
  - 非同期動作
  - 比較一致割り込み
- PWMの基本

Rev. 2505B-03/2016, 2505BJ1-02/2021

目次

序説 1
特徴 1
1. 概要 3
1.1. ﾀｲﾏ/ｶｳﾝﾀについて 3
1.2. ﾀｲﾏ/ｶｳﾝﾀ事象 3
2. ﾀｲﾏ/ｶｳﾝﾀ事象の通知 3
3. ｸﾛｯｸ任意選択 4
4. ﾀｲﾏ/ｶｳﾝﾀの初期設定 6
4.1. 例1 : ﾀｲﾏ/ｶｳﾝﾀ0 溢れ割り込み 6
4.2. 例2 : ﾀｲﾏ/ｶｳﾝﾀ1 捕獲割り込み 7
4.3. 例3 : ﾀｲﾏ/ｶｳﾝﾀ2 非同期動作 8
5. PWMの基本 9
5.1. 例4 : ﾀｲﾏ/ｶｳﾝﾀ2 PWM生成 10
6. 更なる読み物 10
7. 改訂履歴 10

1. 概要

原理的にﾀｲﾏ/ｶｳﾝﾀは簡単なｶｳﾝﾀ(計数器)です。これの優位性は入力ｸﾛｯｸとﾀｲﾏ/ｶｳﾝﾀの動作がﾌﾟﾛｸﾞﾗﾑ実行と独立していることです。決定するｸﾛｯｸはﾀｲﾏ/ｶｳﾝﾀの入力周波数を考慮に入れて経過した周期を計数することによって時間の測定を可能にします。

1.1. ﾀｲﾏ/ｶｳﾝﾀについて

一般的に、megaAVR®は2つの8ﾋﾞｯﾄ ﾀｲﾏ/ｶｳﾝﾀと1つの16ﾋﾞｯﾄ ﾀｲﾏ/ｶｳﾝﾀを持ちます。16ﾋﾞｯﾄ分解能を持つﾀｲﾏ/ｶｳﾝﾀは8ﾋﾞｯﾄ分解能を持つﾀｲﾏ/ｶｳﾝﾀよりも使うのに確かにより柔軟です。けれども、「大きいことは良いことだ」と言うのはﾏｲｸﾛ ｺﾝﾄﾛｰﾗの世界に対して必然的に適用されません。多くの応用に関して、それは8ﾋﾞｯﾄ分解能を持つことで足ります。より高い分解能の使用はﾌﾟﾛｸﾞﾗﾑのより大きな、処理時間の犠牲と速度最適化ｺｰﾄﾞで避けられるべき、ｵｰﾊﾞｰﾍｯﾄﾞを意味します。それはより高いﾃﾞﾊﾞｲｽ費用も意味します。

AVRﾀｲﾏ/ｶｳﾝﾀの柔軟性のため、それらは様々な目的に使うことができます。ﾀｲﾏ/ｶｳﾝﾀ数は独立した構成設定の量を決めます。以降で各種設定任意選択がもっと厳密に記述されるでしょう。

1.2. ﾀｲﾏ/ｶｳﾝﾀ事象

AVRのﾀｲﾏ/ｶｳﾝﾀは各種事象監視に指定できます。TIFRﾚｼﾞｽﾀ内の状態ﾌﾗｸﾞは事象が起こったかを示します。ATmega328PBは8ﾋﾞｯﾄ ﾀｲﾏ/ｶｳﾝﾀのTC0とTC2に対して3つまでの事象を監視するように構成設定することができます。また3つの16ﾋﾞｯﾄ ﾀｲﾏ/ｶｳﾝﾀ(TC1,TC3,TC4)も持ち、それらの各々は4つの事象を支援します。

ﾀｲﾏ/ｶｳﾝﾀ溢れ

ﾀｲﾏ/ｶｳﾝﾀ溢れはｶｳﾝﾀ(計数器)がそれの最大値まで計数され、次のﾀｲﾏ/ｶｳﾝﾀ ｸﾛｯｸ周期で0にﾘｾｯﾄされたことを示します。ﾀｲﾏ/ｶｳﾝﾀの分解能はそのﾀｲﾏ/ｶｳﾝﾀの最大値を決めます。ATmega328PBには、8ﾋﾞｯﾄ分解能の2つのﾀｲﾏ/ｶｳﾝﾀと16ﾋﾞｯﾄ分解能の3つのﾀｲﾏ/ｶｳﾝﾀがあります。ﾀｲﾏ/ｶｳﾝﾀは**式1.**によって計算できる最大値まで計数を行えます。ここで*Res*はﾋﾞｯﾄでの分解能です。

$$MaxVal = 2^{Res} - 1 \quad \cdots\cdots (式1.)$$

ﾀｲﾏ/ｶｳﾝﾀ溢れ事象はﾀｲﾏ/ｶｳﾝﾀ割り込み要求ﾌﾗｸﾞ ﾚｼﾞｽﾀ(TIFRn)内でﾀｲﾏ/ｶｳﾝﾀ溢れ割り込み要求(TOVn)ﾌﾗｸﾞの設定(1)を引き起こします。

比較一致

ﾀｲﾏ/ｶｳﾝﾀ溢れ監視で不充分な場合、比較一致割り込みを使うことができます。比較ﾚｼﾞｽﾀ(OCRnx)はﾀｲﾏ/ｶｳﾝﾀ(TCNTn)が全てのﾀｲﾏ/ｶｳﾝﾀ周期に対して検査される値(0～*MaxVal*)を格納することができます。ﾀｲﾏ/ｶｳﾝﾀがこの比較値に達すると、TIFRn内の比較出力ﾌﾗｸﾞ(OCFnx)が設定(1)されます。ﾀｲﾏ/ｶｳﾝﾀは比較一致でﾀｲﾏ/ｶｳﾝﾀを0に解除するように設定することができます。

関連する出力ﾋﾟﾝは比較一致で自動的に、設定(1)、解除(0)、切り替え(1/0)するように設定できます。この特性は各種周波数の方形波信号を生成するのに大変有用です。これはD/A変換器の実装を可能にする、広範囲な可能性を提供します。PWM動作は波形生成用にも良く仕立てられた特別な動作です。詳細についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄをご覧ください。

捕獲入力

AVRのﾀｲﾏ/ｶｳﾝﾀは捕獲入力事象を起動するための入力ﾋﾟﾝを持ちます。そのようなﾋﾟﾝでの信号変化はﾀｲﾏ/ｶｳﾝﾀ値の読みと捕獲ﾚｼﾞｽﾀ(ICRn)への保存を引き起こします。同時にTIFRn内で捕獲割り込み要求ﾌﾗｸﾞ(ICFn)が設定(1)されます。これは外部ﾊﾟﾙｽの幅を測定するのに有用です。

2. ﾀｲﾏ/ｶｳﾝﾀ事象の通知

ﾀｲﾏ/ｶｳﾝﾀはﾌﾟﾛｸﾞﾗﾑ実行と個別に動作します。各ﾀｲﾏ/ｶｳﾝﾀ事象に対して、ﾀｲﾏ/ｶｳﾝﾀ割り込み要求ﾌﾗｸﾞ ﾚｼﾞｽﾀ(TIFRn)内に対応する状態ﾌﾗｸﾞがあります。ﾀｲﾏ/ｶｳﾝﾀ事象の発生は対応する動作の実行のためにﾌﾟﾛｾｯｻへの通知が必要です。これは起こった事象の状態ﾌﾗｸﾞを設定(1)することによって行われます。

ﾀｲﾏ/ｶｳﾝﾀ事象の監視とそれらへの応答には3つの異なる方法があります。

1. 状態ﾌﾗｸﾞの継続的なﾎﾟｰﾘﾝｸﾞ ･･･ 割り込み要求ﾌﾗｸﾞと対応するｺｰﾄﾞの実行
2. ﾌﾟﾛｸﾞﾗﾑの流れ中断と割り込み処理ﾙｰﾁﾝ(ISR)の実行
3. 自動的な出力ﾋﾟﾝのﾚﾍﾞﾙ変更

割り込み要求ﾌﾗｸﾞのﾎﾟｰﾘﾝｸﾞ

この方式はﾌﾟﾛｾｯｻが対応する割り込み要求ﾌﾗｸﾞの設定(1)によってﾀｲﾏ/ｶｳﾝﾀ事象に注意を払うのに使います。主ﾌﾟﾛｸﾞﾗﾑはそれらの事象が起こったかを見るために、頻繁にそれらのﾌﾗｸﾞの状態を調べ得ます。これには追加の処理時間を費やす、幾許かのﾌﾟﾛｸﾞﾗﾑの間接負担が必要です。この方法の優位点は隙間のない繰り返しが使われる時の短い応答時間です。

ﾀｲﾏ/ｶｳﾝﾀ0用のｱｾﾝﾌﾞﾘ言語実装は以下のｺｰﾄﾞ例のように思えます。これら3行ｺｰﾄﾞは、それらが頻繁に実行されるように、主繰り返しで配置されなければなりません。

```
LOOP:   IN      R16, TIFR0          ; TIFR0値をR16に取得
        SBRS    R16, TOV0           ; TIFR0値のTOV0=0なら、次命令ｽｷｯﾌﾟ
        RJMP    LOOP                ; ﾀｲﾏ/ｶｳﾝﾀ0溢れ未発生で、LOOP:からの無限繰り返し
                                    ; ここからﾀｲﾏ/ｶｳﾝﾀ0溢れ事象処理ｺｰﾄﾞ開始
```

割り込み制御通知

AVRはﾀｲﾏ/ｶｳﾝﾀ事象が起きた(TIFRn内の対応する割り込み要求ﾌﾗｸﾞが設定(1)される)場合に割り込みを実行する設定にできます。通常、ﾌﾟﾛｸﾞﾗﾑ実行が(殆ど)直ちに割り込まれ、ﾌﾟﾛｾｯｻは割り込み処理ﾙｰﾁﾝのｺｰﾄﾞを実行するでしょう。割り込み要求ﾌﾗｸﾞのﾎﾟｰﾘﾝｸﾞに比べて有利なのは主繰り返しでの間接負荷がないことです。これは処理時間を短縮します。「**ﾀｲﾏ/ｶｳﾝﾀの初期設定**」章はこれが実装され得る方法の数例を示します。

ﾀｲﾏ/ｶｳﾝﾀ割り込みはﾀｲﾏ/ｶｳﾝﾀ割り込み許可(TIMSKn)ﾚｼﾞｽﾀ内の対応ﾋﾞｯﾄの設定(1)によって許可されます。次の例はﾀｲﾏ/ｶｳﾝﾀ2のﾁｬﾈﾙAでの比較一致割り込みを許可する方法を示します。

```
LDI     R16, 1<<OCIE2A          ; OCIE2Aのみ1のﾋﾞｯﾄ値取得
STS     TIMSK2, R16             ; ﾀｲﾏ/ｶｳﾝﾀ2 比較A一致割り込み許可
SEI                             ; 全体割り込み(SREG:Iﾋﾞｯﾄ=1)
```

事象での自動反応

本ﾃﾞﾊﾞｲｽのﾀｲﾏ/ｶｳﾝﾀはｺｰﾄﾞ実行の必要なしに純粋にﾊｰﾄﾞｳｪｱを基本としたﾀｲﾏ/ｶｳﾝﾀ割り込み事象での反応の実現性を支援します。関連する出力ﾋﾟﾝは比較一致で自動的に、設定(1)、解除(0)、切り替え(1/0)するように設定できます。他の2つの方法と対比して、これは通常のｺｰﾄﾞ実行と並行して起き、処理時間を必要としません。

後続するｺｰﾄﾞ例は比較値設定とﾋﾟﾝ交互出力許可の方法を示します。ﾀｲﾏ/ｶｳﾝﾀ2の構成設定は次のようにすることができます。

```
LDI     R16, (1<<COM2A0)|(1<<WGM21)|(1<<WGM20)  ; OC2A交互出力指定値取得
STS     TCCR2A, R16             ; 比較一致でOC2A交互出力,高速PWM動作
LDI     R16, 32                 ; 比較値取得
STS     OCR2A, R16              ; 比較値を32に設定
```

ﾋﾟﾝ切り替え出力許可は、それを出力ﾋﾟﾝとするために、OCnxに対応するﾃﾞｰﾀ方向ﾚｼﾞｽﾀのﾋﾞｯﾄが設定(1)されなければなりません。

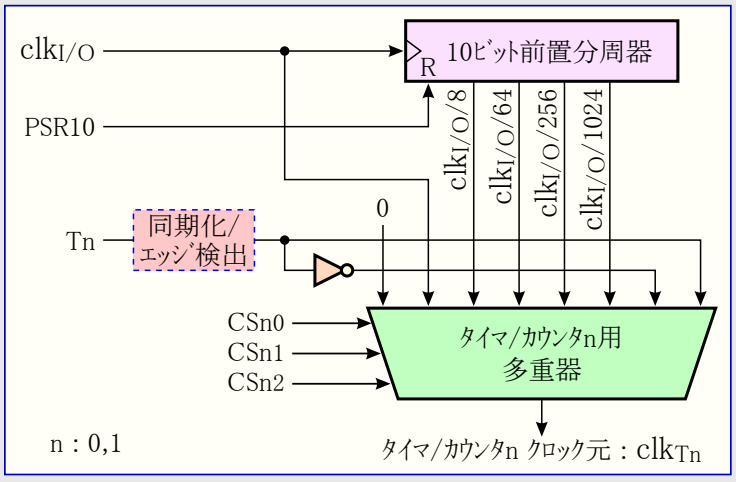
3. ｸﾛｯｸ任意選択

AVRﾀｲﾏ/ｶｳﾝﾀのｸﾛｯｸ部は多重器に接続された前置分周器から成ります。前置分器はｸﾛｯｸ除算器と見做されます。一般的に、これは異なる計数段での各出力信号を持つｶｳﾝﾀ(計数器)として実装されます。ATmega328PBの場合では、入力ｸﾛｯｸを4つ(ﾀｲﾏ/ｶｳﾝﾀ2の場合は6つ)の異なる前置分周されたｸﾛｯｸに分周するのに10ﾋﾞｯﾄ計数器が使われます。多重器はﾀｲﾏ/ｶｳﾝﾀに対して入力信号として使うために、前置分周されたどのｸﾛｯｸ信号を選択するのかに使われます。代わりに、多重器は前置分周器を迂回してﾀｲﾏ/ｶｳﾝﾀ用の入力として使われる外部ﾋﾟﾝに構成設定することができます。

ﾃﾞﾊﾞｲｽには利用可能な2つの前置分周器部があります。ﾀｲﾏ/ｶｳﾝﾀ0とﾀｲﾏ/ｶｳﾝﾀ1は同期ﾀｲﾏ/ｶｳﾝﾀで、入力元としてｼｽﾃﾑ(CPU)ｸﾛｯｸを使い、それらは(各ﾀｲﾏ/ｶｳﾝﾀが対応するCSnxﾋﾞｯﾄを使って独立して構成設定される限り)両方が同じ前置分周器部を使うことができます。けれども、非同期ｸﾛｯｸ駆動されるﾀｲﾏ/ｶｳﾝﾀ2はｼｽﾃﾑ ｸﾛｯｸから独立するために自身の前置分周器を必要とします。

右図はﾀｲﾏ/ｶｳﾝﾀ0と1に対する前置分周と構成設定部を示します。ﾃﾞｰﾀｼｰﾄは全ての前置分周器と多重器を示すもっと詳細な図を含みます。可能なｸﾛｯｸ設定の概要は**表3-1. ｸﾛｯｸ設定の概要**で与えられます。以降の項で、これらの設定がもっと明快に記述されるでしょう。

図3-1. ﾀｲﾏ/ｶｳﾝﾀ0と1の前置分周器



注: 1. 前置分周器は動作の間、継続して走行します。ﾀｲﾏ/ｶｳﾝﾀが非常に正確に計数しなければならない場合、前置分周器が0から計数を始めるのを保証しなければなりません。

2. 共有された前置分周器を持つﾃﾞﾊﾞｲｽで、前置分周器ﾘｾｯﾄの実行は接続されたﾀｲﾏ/ｶｳﾝﾀ全てに影響を及ぼすでしょう。

システム クロックによるクロック駆動

この場合はシステム クロックが前置分周器に対する入力信号として使用されます。例え前置分周された値がシステム クロックの代わりに選ばれたとしても、そのクロックはシステム クロックを基にされます。従って、タイマ/カウンタ クロックはシステム クロックに同期します。ATmega328PBの5つタイマ/カウンタ全てと他のAVRデバイスの殆どがこの任意選択を支援します。システム クロックの高い周波数のため、小さい時間構成が実装または監視できます。

タイマ/カウンタ溢れ周波数はタイマ/カウンタが網羅する時間構成の大きさの良い指示点です。**式2.**は、タイマ/カウンタ溢れ周波数(*TOV*$_{CK}$)、タイマ/カウンタの最大値(*MaxVal*)、システム クロック(*CK*)、前置分周器の分周係数(*PVal*)間の相互関係を示します。

表3-1. クロック設定の概要

TCCRnx			同期タイマ/カウンタ0,1 (P$_{CK}$=CK)	非同期タイマ/カウンタ2 (P$_{CK}$2はAS2ビット依存)
ビット2	ビット1	ビット0		
CSn2	CSn1	CSn0	TCK0,1	TCK2
0	0	0	0 (停止)	0 (停止)
0	0	1	CK	P$_{CK2}$(システム クロック/非同期クロック)
0	1	0	CK/8	P$_{CK2}$/8
0	1	1	CK/64	P$_{CK2}$/32
1	0	0	CK/256	P$_{CK2}$/64
1	0	1	CK/1024	P$_{CK2}$/128
1	1	0	外部Tnピンの下降端	P$_{CK2}$/256
1	1	1	外部Tnピンの上昇端	P$_{CK2}$/1024

$$TOV_{CK} = \frac{f_{CK}}{MaxVal} = \frac{(P_{CKn} \div PVal)}{MaxVal} = \frac{P_{CKn}}{(PVal \times MaxVal)} \quad \cdots (式2.)$$

CPUはf_{CPU}=1MHzで走行し、タイマ/カウンタの分解能は8ビット(*MaxVal*=256)と仮定してください。分周値64がTCK=1MHz/64でのクロック駆動をタイマ/カウンタにさせ、故に秒当たり約61回の溢れになります。正確な計算内容については**式2.**をご覧ください。

秒当たり61回の溢れ事象を得ることは16ms毎に溢れが起こることを意味します。最大前置分周値は約262ms毎にタイマ/ カウンタに溢れを生成し、一方最小前置分周値は256µs毎にタイマ/カウンタに溢れを生成するでしょう。

多くの場合では設定を決めるために各種の手段が使用されます。応用の必要条件がタイマ/カウンタ溢れ周波数を特定するでしょう。これを基にして、タイマ/カウンタ分解能と共にCPUのクロック周波数を与え、前置分周器設定は次式に従って計算されるでしょう。

$$PVal = \frac{P_{CKn}}{(\mathrm{TOV} \times MaxVal)} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (式3.)$$

タイマ/カウンタ0用のアセンブリ言語実装は後続するコード例のように思われます。これらの行はタイマ/カウンタ0制御レジスタB(TCCR0B)で前置分周値をクロック分周係数1024に設定します。

```
LDI     R16, (1<<CS02)|(1<<CS00)        ; 1024分周指定値取得
OUT     TCCR0B, R16                     ; タイマ/カウンタ0 クロック=システム クロック,1024分周で始動
```

非同期クロックによるクロック駆動

この任意選択を支援しない他の2種類のタイマ/カウンタと比べ、ATmega328PBのタイマ/カウンタ2は非同期外部クロックによってクロック駆動ができます。この目的に対してはクリスタル発振子またはセラミック振動子がTOSC1ピンとTOSC2ピンを経由して内部発振器に接続することができます。この発振器は時計用の32.768kHzクリスタル用に最適化されています。この周波数は実時間時計(RTC)の実装用に適合されています。より多くの情報については「AVR134:非同期計時器を使う実時間時計(RTC)」をご覧ください。分離したクロックの主な優位点はシステム クロックと無関係なことです。これは高い処理周波数でのデバイス走行の一方で、正確なタイミングに最適化された周波数での外部クロックによってクロック駆動されるタイマ/カウンタを可能にします。付加的なパワーセーブ動作支援はデバイスを休止形態にする一方で、非同期タイマ/カウンタの勤めを許します。

非同期動作はいくつかの追加の考慮が必要です。タイマ/カウンタ2のクロック駆動が非同期なため、タイマ/カウンタ事象はCPUに同期されなければなりません。これには少なくてもシステム クロックの1/4よりも低いタイマ/カウンタ クロック周波数が必要です。他方、同期と非同期の入出力間での衝突は避けられなければなりません。これは一時レジスタの使用によって行われます。状態ビットは構成設定レジスタの更新が処理中の時を合図します。詳細についてはデータシートの非同期状態レジスタ(ASSR)の内容をご覧ください。

TOV$_{CK}$は**式2.**に従って計算されますが、システム クロックの代わりに、この発振器周波数の使用によります。タイマ/カウンタ制御レジスタB(TCCR2B)の設定は**表3-1. クロック設定の概要**で与えられます。前置分周器入力クロックP$_{CK2}$はASSR内のAS2ビットの作用(結果)です。このビットが解除(0)されると、タイマ/カウンタは入力周波数としてシステム クロックでの同期動作で走行します。このビットが設定(1)されると、TOSC1ピンとTOSC2ピンの非同期クロック信号が前置分周器の入力信号として使用されます。

タイマ/カウンタ2用のアセンブリ言語実装は後続するコード例のように思われます。これら2行はTCCR2B内の前置分周値をクロック分周係数1024に設定します(**表3-1. クロック設定の概要**をご覧ください)。

```
LDI     R16, (1<<CS22)|(1<<CS21)|(1<<CS20)  ; 1024分周指定値取得
STS     TCCR2B, R16                         ; タイマ/カウンタ2 クロック=システム クロック,1024分周で始動
```

外部クロック駆動

外部クロック駆動はタイマ/カウンタ0とタイマ/カウンタ1でだけ支援されます。この動作種別はタイマ/カウンタ クロック信号として広い範囲の外部信号の使用を許します。これはCPUがピンの状態を検出し、外部クロック信号が検出された場合にシステム クロックと同期してタイマ/カウンタをクロック駆動することを意味する、同期クロック駆動です。T0/T1ピンはピン同期化論理回路によって毎システム クロック周期に1度採取されます。同期(採取)された信号はその後にエッジ(端)検出器を通して渡されます。このクロック駆動任意選択はTCCRnxで選択され、CSn2,CSn1,CSn0ビットの設定は**表3-1. クロック設定の概要**で得られます。

タイマ/カウンタ0用のアセンブリ言語実装は以下のコード例のように思われます。これらの行は活性(有効)なクロック端として上昇端でのタイマ/カウンタ クロック用の入力ピンとしてT0ピンを設定します。

```
LDI     R16, (1<<CS02) | (1<<CS01) | (1<<CS00)     ; T0ピン上昇端外部クロック指定値取得
OUT     TCCR0B, R16                                ; タイマ/カウンタ0 クロック=外部T0ピン,上昇端で始動
```

注: T0ピンがポートBのデータ方向レジスタ(DDRB)で入力ピンであるのを確実にすることが重要です。この方向レジスタの設定は、AVRでソフトウェア クロック駆動のタイマ/カウンタ実装を許すためにも、タイマ/カウンタ設定によって上書きされません。T0とT1は既定で入力です。

タイマ/カウンタの停止方法

計数(状態)からタイマ/カウンタを停止するのは簡単です。TCCRnx内の前置分周値としての0値(CSn2～0=000)が対応するタイマ/カウンタを停止します(**表3-1. クロック設定の概要**をご覧ください)。けれども、前置分周器は未だ走行していることを覚えて置いてください。

タイマ/カウンタ0用のアセンブリ言語実装は以下のコード例のように思われます。

```
CLR     R16                                        ; 0値取得
OUT     TCCR0B, R16                                ; タイマ/カウンタ0停止(TCCR0B=$00)
```

注: 他のTCCRnxはクロック選択(CSnx)ビット以外の構成設定ビットを含むかもしれません。上の命令行はそれらのビットを解除(0)します。それらのビットが設定(1)されている場合、これは避けられなければなりません。以下のコード断片で示されるように、それには1つの追加コード行を費やします。

```
IN      R16, TCCR0B                                ; TCCR0B現在値取得
ANDI    R16, ~((1<<CS02) | (1<<CS01) | (1<<CS00))  ; CS02,CS01,CS00解除(=000)
OUT     TCCR0B, R16                                ; タイマ/カウンタ0停止(CS02～0=000),他ビット影響なし
```

4. タイマ/カウンタの初期設定

本章は3つの異なるタイマ/カウンタの初期設定法に関する具体的な例を示します。加えて、データシートと本資料の最終章で一覧される応用記述が読まれるべきです。特にATmega328PB以外の他のデバイスに設定を変形する時に。

割り込みの使用はタイマ/カウンタ事象に反応するための最も一般的な方法です。以降で記述される例は割り込みを使います。

これらのタイマ/カウンタの各種機能と無関係に、それらの全ては2つの共通するものを持ちます。タイマ/カウンタはクロック元選択によって開始されなければならず、割り込みが使われる場合はそれらが許可されなければなりません。

共用レジスタ

同じレジスタが主コードと割り込み処理ルーチン(ISR)で使われる場合、それらのレジスタはISRの始めで保存され、ISRの終りで回復されなければなりません。32個全てのレジスタが応用で必要とされないなら、主コードとISRで独立したレジスタを使うことによって、保存/回復操作を避けることができます。

割り込み処理によってこれが自動的に行われないため、ステータス レジスタ(SREG)の保存を覚えて置くことも非常に重要です。

注: Cコンパイラはこれを自動的に取り扱います。アセンブリ言語が使われる場合、PUSHとPOPの命令を使うことによって手動で行わなければなりません。

4.1. 例1 : タイマ/カウンタ0 溢れ割り込み

8ビット タイマ/カウンタ0は同期タイマ/カウンタです。これはシステム クロック、前置分周したシステム クロック、またはシステム クロックで同期された外部クロックによってクロック駆動されることを意味します(これについての詳細に関しては「**クロック任意選択**」章をご覧ください)。このタイマ/カウンタは最も少ない3つの(構成設定用レジスタの)複合体です。タイマ/カウンタの走行を得るために、数設定が行われなければならないだけです。

以下の例はタイマ/カウンタ0がタイマ/カウンタ0溢れ割り込み生成にどう使われ得るかを示します。割り込み毎にポートBのPB5ピンが(1/0)切り替え(出力)されるでしょう。これを観測するのに、STK600開発基板またはATmega328PB Xplained Miniを使うことができます。STK600では、PB5がLEDに接続されなければなりません。LEDは次式によって決められる周波数(f_{LED})で点滅するでしょう。

$$f_{LED} = \frac{f_{CK}}{2 \times MaxVal} = \frac{(CK \div PVal)}{2 \times MaxVal} = \frac{CK}{2 \times (PVal \times MaxVal)}$$

8ビット タイマ/カウンタから成るシステム(*MaxVal*=256)と*Pval*=1024の前置分周値によって分周される*CK*=1MHzのシステム クロックは、LEDを概ね3.8Hzの周波数(f_{LED})で点滅させます。

次の初期化ルーチンはこのようなシステムの設定方法を示します。

```
init_Ex1:       LDI     R16, (1<<CS02)|(1<<CS00)        ; 1024分周指定値取得
                OUT     TCCR0B, R16                     ; タイマ/カウンタ0 クロック=I/Oクロック/1024で始動
                LDI     R16, 1<<TOV0                    ; TOV0のみ1値取得
                OUT     TIFR0, R16                      ; TOV0=0,保留割り込み解除
                LDI     R16, 1<<TOIE0                   ; TOIE0のみ1値取得
                STS     TIMSK0, R16                     ; タイマ/カウンタ0溢れ割り込み許可
                RET                                     ; 呼び出し元へ復帰
```

対応するC言語ｺｰﾄﾞはこのように思います。

```
void init_Ex1 (void)
{
   TCCR0B = (1<<CS02)|(1<<CS00);                        // タイマ/カウンタ0 クロック=I/Oクロック/1024で始動
   TIFR0  = 1<<TOV0;                                    // TOV0=0,保留割り込み解除
   TIMSK0 = 1<<TOIE0;                                   // タイマ/カウンタ0溢れ割り込み許可
}
```

次の段階では割り込み処理ルーチンが実装されなければなりません。このルーチンは毎回のタイマ/カウンタ0溢れ割り込みで実行されます。この例内のルーチンの目的はPB5を交互切り替えすることです。

```
ISR_TOV0:       PUSH    R16                             ; R16値ｽﾀｯｸ保存
                IN      R16, SREG                       ; 現SREG値複写/保存
                RCALL   TOGGLEPIN                       ; PB5値論理反転
                OUT     SREG, R16                       ; SREG値回復
                POP     R16                             ; ｽﾀｯｸからR16値回復
                RETI                                    ; 割り込み位置へ復帰

TOGGLEPIN:      SBIC    PORTB, PORTB5                   ; PB5=0でｽｷｯﾌﾟ
                RJMP    CLEARPIN                        ; PB5=1で0設定へ分岐
                SBI     PORTB, PORTB5                   ; PB5=1設定
                RJMP    RET1                            ; 復帰へ分岐
CLEARPIN:       CBI     PORTB, PORTB5                   ; PB5=0設定
RET1:           RET                                     ; 呼び出し元へ復帰
```

対応するC言語ｺｰﾄﾞは次のとおりです。

```
ISR (TIMER0_OVF0_vect)
{
        PORTB ^= (1<<USER_LED);                         // タイマ/カウンタ0溢れでPB5出力1/0交互切り替え
}
```

4.2. 例2：ﾀｲﾏ/ｶｳﾝﾀ1 捕獲割り込み

16ﾋﾞｯﾄ ﾀｲﾏ/ｶｳﾝﾀ1は同期ﾀｲﾏ/ｶｳﾝﾀです。これはｼｽﾃﾑ ｸﾛｯｸ、前置分周したｼｽﾃﾑ ｸﾛｯｸ、またはｼｽﾃﾑ ｸﾛｯｸで同期された外部ｸﾛｯｸによってｸﾛｯｸ駆動されることを意味します。ﾀｲﾏ/ｶｳﾝﾀ1の16ﾋﾞｯﾄ ﾚｼﾞｽﾀが同時に読み書きされるのを保証するために一時ﾚｼﾞｽﾀ(TEMP)が使われます。これは特別な順番でこれらのﾚｼﾞｽﾀに入出力することを必要とします。詳細については「AVR072:16ﾋﾞｯﾄI/Oﾚｼﾞｽﾀの入出力」応用記述とﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄをご覧ください。ﾚｼﾞｽﾀを入出力する正しい方法は右表で示されます。

表4-1. 16ﾋﾞｯﾄ ﾚｼﾞｽﾀの入出力

操作	1回目入出力	2回目入出力
読み込み	下位ﾊﾞｲﾄ	上位ﾊﾞｲﾄ
書き込み	上位ﾊﾞｲﾄ	下位ﾊﾞｲﾄ

これに従って、16ﾋﾞｯﾄ ﾚｼﾞｽﾀの読み込み操作はこのように思われます。

```
                LDS     R16, TCNT1L                     ; タイマ/カウンタ1下位ﾊﾞｲﾄ値取得
                LDS     R17, TCNT1H                     ; タイマ/カウンタ1上位ﾊﾞｲﾄ値取得
```

このﾚｼﾞｽﾀへの書き込みは逆順でﾚｼﾞｽﾀをｱｸｾｽしなければなりません。

```
                STS     TCNT1H, R17                     ; タイマ/カウンタ1上位ﾊﾞｲﾄ値設定
                STS     TCNT1L, R16                     ; タイマ/カウンタ1下位ﾊﾞｲﾄ値設定
```

Cｺﾝﾊﾟｲﾗは16ﾋﾞｯﾄI/O読み書き操作を正しい順で取り扱います。

この例は捕獲入力の事象と割り込みの非常に簡単な使用法の実装を示します。PB0ﾎﾟｰﾄ ﾋﾟﾝが捕獲入力(ICP1)ﾋﾟﾝです。このﾋﾟﾝの値が変化した場合に、捕獲が起動され、16ﾋﾞｯﾄ計数器(TCNT1)の値が捕獲ﾚｼﾞｽﾀ(ICR1)に書かれます。

外部信号のﾃﾞｭｰﾃｨ ｻｲｸﾙの測定は起動端が各捕獲の後で変更されることが必要です。端検出の変更はICR1ﾚｼﾞｽﾀが読まれた後で可能な限り速く行われなければなりません。これは次の逆端が起こる前に検知構成設定での変更が行われるのを確実にすることです。端の変更後、捕獲割り込み要求(ICF1)ﾌﾗｸﾞはｿﾌﾄｳｪｱ(I/Oﾋﾞｯﾄ位置へ論理1を書くこと)によって解除(0)されなければなりません。周波数測定だけについて、(割り込み処理部が使われる場合、)ICF1ﾌﾗｸﾞの解除(0)は必要とされません。

(訳注) ATmega328PBのﾃﾞｰﾀｼｰﾄに於けるﾀｲﾏ/ｶｳﾝﾀ1/3/4の各ﾚｼﾞｽﾀのﾋﾞｯﾄ/ﾌﾗｸﾞ名は末尾に1/3/4が付きませんが、本書では「ﾀｲﾏ/ｶｳﾝﾀ1の」を明確にするために末尾に1を付加しています(例:ICF⇒ICF1)。他の殆どのﾃﾞﾊﾞｲｽは末尾に数値が付きます。

以下の初期化ルーチンはこのようなシステムの設定法を示します。

```
init_Ex2:       LDI     R16, (1<<ICNC1)|(1<<CS11)|(1<<CS10)   ; ICP1雑音消去器許可,64分周指定値取得
                STS     TCCR1B, R16             ; タイマ/カウンタ1 ICP1雑音消去器許可,クロック=I/Oクロック/64で始動
                LDI     R16, 1<<ICF1            ; ICF1のみ1値取得
                OUT     TIFR1, R16              ; ICF1=0,保留割り込み解除
                LDI     R16, 1<<ICIE1           ; ICIE1のみ1値取得
                STS     TIMSK1, R16             ; タイマ/カウンタ1捕獲割り込み許可
                CBI     DDRB, PPRTB0            ; PB0/ICP1入力設定
                RET                             ; 呼び出し元へ復帰
```

対応するC言語コードはこのように思われます。

```
void init_Ex2 (void)
{
   TCCR1B = (1<<ICNC1)|(1<<CS11)|(1<<CS10);   // タイマ/カウンタ1 ICP1雑音消去器許可,クロック=I/Oクロック/64で始動
   TIFR   = 1<<ICF1;                          // ICF1=0,保留割り込み解除
   TIMSK  = 1<<ICIE1;                         // タイマ/カウンタ1捕獲割り込み許可
}
```

次の段階では割り込み処理ルーチン(ISR)が実装されなければなりません。このルーチンは毎回の捕獲入力事象で実行されます。ISRは各捕獲事象でPB5を交互切り替えし、次の測定の前にTCNT1レジスタを解除します。

```
ISR_TIM1_CAPT:  PUSH    R16                     ; R16値スタック保存
                IN      R16, SREG               ; 現SREG値複写
                PUSH    R16                     ; 現SREG値スタック保存
                CLR     R16                     ; 0値取得
                STS     TCNT1H, R16             ; 一時レジスタ=0
                STS     TCNT1L, R16             ; TCNT1L=0,一時レジスタ⇒TCNT1H=0、再始動
                RCALL   TOGGLEPIN               ; PB5値論理反転
                POP     R16                     ; スタックからSREG値取得
                OUT     SREG, R16               ; SREG値回復
                POP     R16                     ; スタックからR16値回復
                RETI                            ; 割り込み位置へ復帰
```

対応するC言語コードはこのように思います。

```
ISR (TIMER1_CAPT1_vect)
{
   TCNT1  = 0;                                 // タイマ/カウンタ1(TCNT1)=0、再始動
   PORTB ^= (1<<USER_LED);                     // タイマ/カウンタ1捕獲発生でPB5出力1/0交互切り替え
}
```

注: この実装にはタイマ/カウンタ溢れが検知されない、1つの不利な点があります。故に、波形のデューティ サイクルが16ビット計数器で賄われ得るよりも長い周期を持つ場合、次端発生前に溢れが起きるでしょう。捕獲が実行される前にタイマ/カウンタ溢れが起こったかを検知するのに、タイマ/カウンタ溢れISRで設定される全域変数を使うことができます。この変数が設定されていれば、実効捕獲値は($FFFF+ICR1の内容)です。

4.3. 例3 : タイマ/カウンタ2 非同期動作

タイマ/カウンタ2はタイマ/カウンタ0とタイマ/カウンタ1のように同期動作で使えます。加えて、非同期動作を使うことができます。詳細については「非同期クロックによるクロック駆動」またはデータシートで非同期クロック駆動の記述をご覧ください。

例 – タイマ/カウンタ2 比較一致割り込み

この例はタイマ/カウンタ2の比較一致割り込みの使用方法を示します。このタイマ/カウンタは比較一致事象が毎秒発生するように構成設定されます。この機能はRTCの実装に使われ得ます。けれども、この例ではPB5が0.5Hzの周波数で点滅するように、毎回の比較一致事象でポート ピンが反転されます。

直前の例と同様に、PB5はLEDに接続されなければなりません。加えて、32.768kHzクリスタル発振子がポートBのTOSC1/PB6とTOSC2/PB7のピンに配置されなければなりません。

タイマ/カウンタの設定は式2.に従って計算することができます。タイマ/カウンタの最大値(*MaxVal*)として、代わりにOCR2の値が使われなければなりません。前置分周器クロック(*PCKn*)はこの場合、時計用(32.768kHz)クリスタル発振子のクロック信号(f_{OSCCK})です。TOV_{CK}は応用によって1秒に指定される溢れ周波数です。この関連の数学的記述は次式によって示されます。

$$1 = TOV_{CK} = \frac{f_{OSCCK}}{PVal \times \mathrm{OCR2}} = \frac{32.768\mathrm{kHz}}{PVal \times \mathrm{OCR2}}$$

2つのタイマ/カウンタ2比較一致事象間で1秒の遅延時間を得るために、1024の前置分周値が対応する32のOCR2値に加えて選択されます。

以下の初期化ルーチンはこのようなシステムの設定法を示します。

```
init_Ex3:       LDI     R16, 1<<AS2                                 ; AS2のみ1値取得
                STS     ASSR, R16                                   ; 非同期動作許可
                LDI     R16, (1<<COM2A0)|(1<<WGM21)                 ; CTC動作,OC2A交互指定値取得
                STS     TCCR2A, R16                                 ; CTC動作,比較一致でOC2A交互
                LDI     R16, (1<<CS22)|(1<<CS21)|(1<<CS20)          ; 1024分周指定値取得
                STS     TCCR2B, R16                                 ; 1024分周(=32768Hz/1024)で始動
                LDI     R16, 32                                     ; 値32取得
                STS     OCR2A, R16                                  ; 比較値32設定
loop:           LDS     R16, ASSR                                   ; 現ASSR値取得 (注)
                ANDI    R16, (1<<OCR2AUB)|(1<<OCR2BUB)|(1<<TCR2AUB)|(1<<TCR2AUB)|(1<<TCN2UB)
                                                                    ; 更新中検査
                BRNE    loop                                        ; TC2ﾚｼﾞｽﾀ更新完了まで待機
;
                LDI     R16, (1<<TOV2)|(1<<OCF2A)|(1<<OCF2B)        ; TOV2,OCF2A,OCF2Bのみ1値取得
                OUT     TIFR2, R16                                  ; 上記保留割り込み解除(0)
                LDI     R16, 1<<OCIE2A                              ; OCIE2Aのみ1値取得
                STS     TIMSK2, R16                                 ; ﾀｲﾏ/ｶｳﾝﾀ2比較A一致割り込み許可
                SBI     DDRB, PORTB3                                ; PB3/OC2Aﾋﾟﾝを出力に設定
                RET                                                 ; 呼び出し元へ復帰
```

注: loop:からの3行の処理に関してはATmega328PBﾃﾞｰﾀｼｰﾄの「**ﾀｲﾏ/ｶｳﾝﾀ2の非同期動作**」項を参照してください。

対応するC言語ｺｰﾄﾞは次のとおりで有り得ます。

```
void init_Ex3 (void)
{
   ASSR   = 1<<AS2;                                                 // 非同期動作許可
   TCCR2A = (1<<COM2A0)|(1<<WGM21);                                 // CTC動作,比較一致でOC2A交互
   TCCR2B = (1<<CS22)|(1<<CS21)|(1<<CS20);                          // 1024分周(=32768Hz/1024)で始動
   OCR2A  = 32;                                                     // 比較値32設定
   while (ASSR & ((1<<OCR2AUB)|(1<<OCR2BUB)|(1<<TCR2AUB)|(1<<TCR2AUB)|(1<<TCN2UB)));
                                                                    // TC2ﾚｼﾞｽﾀ更新完了まで待機 (注)
   TIFR2  = 1<<(1<<TOV2)|(1<<OCF2A)|(1<<OCF2B);                     // 保留割り込み解除(0)
   TIMSK2 = 1<<OCIE2A;                                              // ﾀｲﾏ/ｶｳﾝﾀ2比較A一致割り込み許可
   DDRB   |= (1<<OC2A_PIN);                                         // PB3/OC2Aﾋﾟﾝを出力に設定
}
```

注: ATmega328PBﾃﾞｰﾀｼｰﾄの「**ﾀｲﾏ/ｶｳﾝﾀ2の非同期動作**」項を参照してください。

次の段階では割り込み処理ルーチンが実装されなければなりません。このルーチンは毎回の比較一致事象で実行されます。この例での目的はPB5に接続されたLEDを交互ON/OFFすることです。

```
ISR_OCIE2A:     PUSH    R16                                         ; R16値ｽﾀｯｸ保存
                IN      R16, SREG                                   ; 現SREG値複写/保存
                RCALL   TOGGLEPIN                                   ; PB5値論理反転
                OUT     SREG, R16                                   ; SREG値回復
                POP     R16                                         ; ｽﾀｯｸからR16値回復
                RETI                                                ; 割り込み位置へ復帰
```

対応するC言語ｺｰﾄﾞは次のとおりで有り得ます。

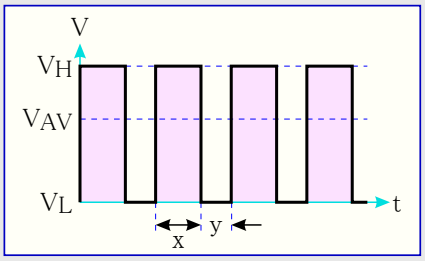
```
ISR (TIMER2_COMPA_vect)
{
  PORTB ^= (1<<USER_LED);                                           // ﾀｲﾏ/ｶｳﾝﾀ2比較A一致でPB5出力1/0交互切り替え
}
```

5. PWMの基本

PWMはPulse Width Modulation(ﾊﾟﾙｽ幅変調)の省略形です。この動作では、ﾀｲﾏ/ｶｳﾝﾀが昇降ｶｳﾝﾀ(計数器)として働きます。これは計数器が最大値へ上昇計数し、その後に0へ解除されることを意味します。PWMの利点はﾃﾞｭｰﾃｨ ｻｲｸﾙ関係が一貫した位相方向で変更できることです。

PWMが比較出力(OCnx)ﾋﾟﾝ交互切り替えに設定されると、このﾋﾟﾝでの信号は右図で示されるように見れます。

図5-1. PWMの出力信号



VH : Highﾚﾍﾞﾙ出力電圧
VL : Lowﾚﾍﾞﾙ出力電圧
VAV : 平均出力電圧ﾚﾍﾞﾙ
x : Highﾚﾍﾞﾙ ﾃﾞｭｰﾃｨ ｻｲｸﾙ
y : Lowﾚﾍﾞﾙ ﾃﾞｭｰﾃｨ ｻｲｸﾙ

相対的に高い速度のPWMと組み合わされた出力ﾋﾟﾝでの低域通過濾波器(ﾛｰﾊﾟｽ ﾌｨﾙﾀ)が出力信号として方形波に代わって一定電圧ﾚﾍﾞﾙにさせるでしょう。**式4.**は、この電圧ﾚﾍﾞﾙの計算方法を示します。

$$V_{AV} = \frac{(V_H \times x + V_L \times y)}{(x + y)} \quad \text{(式4.)}$$

ここで、 $x = OCRnx \times 2$

$y = (MaxVal - OCRnx)$

$$V_{AV} = \frac{(V_H \times OCRnx + V_L \times (MaxVal - OCRnx))}{MaxVal} \quad \text{(式5.)}$$

この方法がVCCとGND間の電圧ﾚﾍﾞﾙを生成することをﾀｲﾏ/ｶｳﾝﾀに許す事実は、D/A変換器がPWMを使って実装され得ることを意味します。これについての詳細は「**AVR314:DTMF送信機**」と「**AVR335:AVRと直列DataFlashでのﾃﾞｼﾞﾀﾙ音響記録器**」の応用記述で記述されます。

5.1. 例4：ﾀｲﾏ/ｶｳﾝﾀ2 PWM生成

この例はPWMの出力(PB3/OC2A)ﾋﾟﾝでVCCとGND間の電圧生成方法を示します。これを観測するために、PB3がLEDに接続されるべきです。この出力信号は1/8が7/8(OCR2A=$E0)になるﾃﾞｭｰﾃｨ関係を持ちます。

以下の初期化ﾙｰﾁﾝはこのようなｼｽﾃﾑの設定法を示します。

```
init_Ex4:       LDI     R16, (1<<COM2A1)|(1<<WGM21)|(1<<WGM20)    ; 非反転8ﾋﾞｯﾄ高速PWM動作指定値取得
                STS     TCCR2A, R16                               ; 非反転8ﾋﾞｯﾄ高速PWM動作設定
                LDI     R16, (1<<CS20)                            ; ﾀｲﾏ/ｶｳﾝﾀ ｸﾛｯｸ=I/Oｸﾛｯｸ指定値取得
                STS     TCCR2B, R16                               ; 前置分周なしでTC2始動
                LDI     R16, $E0                                  ; 1/8ﾃﾞｭｰﾃｨ値取得
                STS     OCR2A, R16                                ; 1/8ﾃﾞｭｰﾃｨ(比較)値設定
                SBI     DDRB, PORTB3                              ; PB3/OC2Aﾋﾟﾝを出力に設定
                RET                                               ; 呼び出し元へ復帰
```

対応するC言語ｺｰﾄﾞはこのように思います。

```
void init_Ex4 (void)
{
    TCCR2A = (1<<COM2A1)|(1<<WGM21)|(1<<WGM20);                   // 非反転8ﾋﾞｯﾄ高速PWM動作設定
    TCCR2B = (1<<CS20);                                           // 前置分周なしでTC2始動
    OCR2A  = 0xE0;                                                // 1/8ﾃﾞｭｰﾃｨ(比較)値設定
    DDRB   |= (1<<OC2A_PIN);                                      // PB3/OC2Aﾋﾟﾝを出力に設定
}
```

6. 更なる読み物

1. AVR072：16ﾋﾞｯﾄI/Oﾚｼﾞｽﾀの入出力
2. AVR131：AVRの高速なPWMの使用法
3. AVR134：非同期計時器を使う実時間時計(RTC)
4. AVR314：DTMF送信機
5. AVR335：AVR®と直列DataFlash®でのﾃﾞｼﾞﾀﾙ音響記録器

7. 改訂履歴

文書改訂	日付	注釈
2505A	2002年2月	初版文書公開
2505B	2016年3月	より新しいﾃﾞﾊﾞｲｽ用に更新

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA TEL:(+1)(408) 441-0311 FAX: (+1)(408) 436-4200 | www.atmel.com

---

本応用記述はAtmelのAVR130応用記述(Rev.2505B-03/2016)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。