RH850/U2A-EVA Group

GTM アプリケーションノート

要旨

本アプリケーションノートでは、ルネサスエレクトロニクスの自動車向けシングルチップマイクロコンピュータ RH850/U2A シリーズ(以降、U2A と称す)における Generic Timer Module(GTM)機能について説明します。

本資料およびプログラムは、RH850/U2A 搭載機能の理解促進を意図するものであり、量産設計を対象とするものではありません。

また、最新のマニュアル、正誤表、テクニカルアップデートや、開発環境の更新を反映しておりません。該当機能を使用される場合には、本プログラムは参考として扱い、最新のドキュメントや開発環境にて、お客様の責任において行ってください。

対象デバイス

- RH850/U2A-EVA Group

対象統合開発環境

CS+(ルネサスエレクトロニクス社製)

バージョン　　：V8.07.00

デバイスファイル　：DR7F702300.DVF

DR7F702301.DVF

DR7F702302.DVF

参照文書

RH850/U2A-EVA ユーザーズマニュアル ハードウェア編

デバイスの機能詳細及び電気的特性に関してはユーザーズマニュアル ハードウェア編に記載します。

本アプリケーションノートは以下のマニュアルを参照し作成しております。

・ RH850/U2A-EVA User’s Manual (Rev1.00): R01UH0864EJ0120

目次

1. 適用

GTM には多数の機能がありますが、本アプリケーションノートでは RH850/U2A に実装されている機能についてのみ説明します。なお、次の機能については RH850/U2A には実装されておりませんので本アプリケーションノートでは触れません。

TOM

BRC

F2A

DPLL

MAP

SPE

AFD

FIFO

AXI Master/Slave

2. 概要

2.1 概要

GTM は、Generic Timer Module の略称で、Robert Bosch GmbH 社がライセンスしているタイマ IP です。

GTM は信号入力機能(TIM)、クロックおよびタイムスタンプ機能(CMU および TBU)、信号出力機能(ATOM)等を持ち、さらにプログラミング可能なシーケンサ(MCS)と、各機能間でデータの交換を可能とするデータルーティング機能(ARU)により構成されています。信号の入出力処理は直接マイコンにて制御可能ですが、ARU や MCS を使用すれば、簡単な処理であればマイコンが介在せずに信号キャプチャや信号出力の制御が可能です。

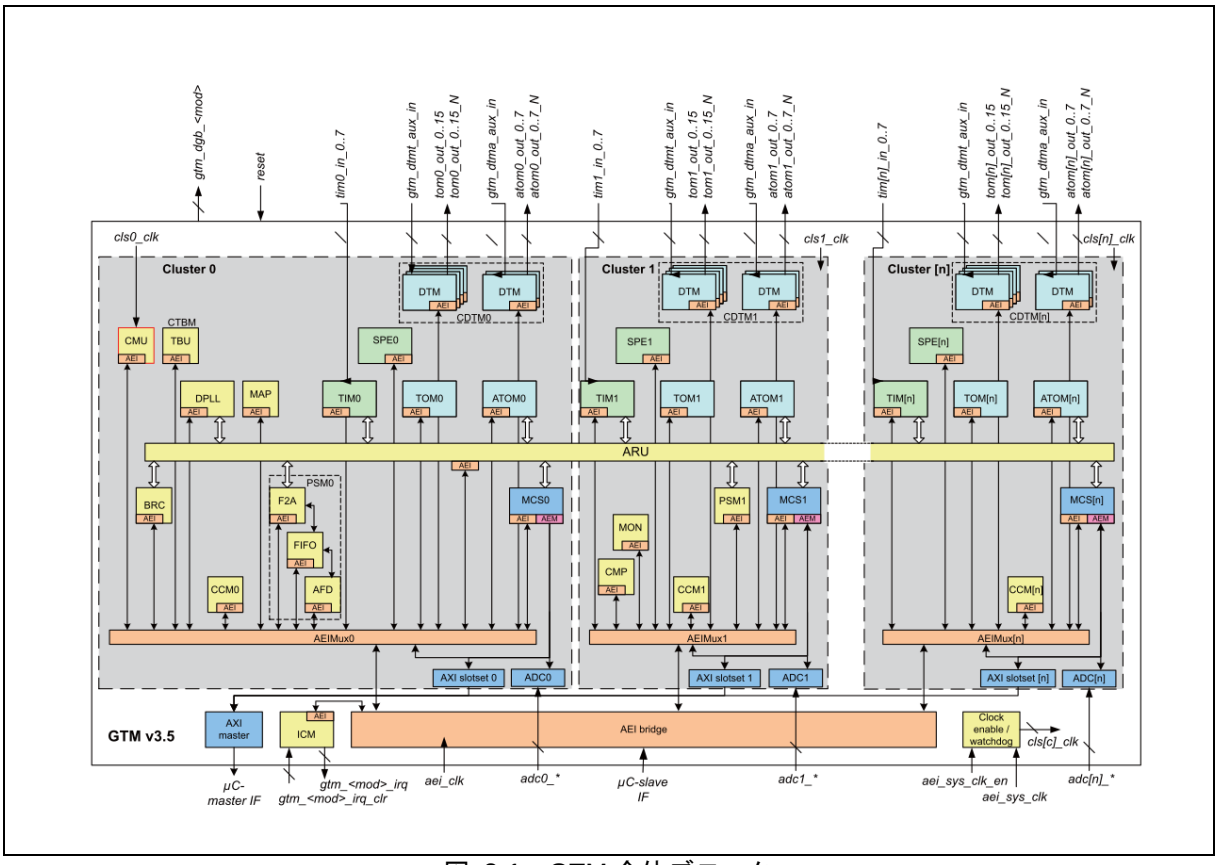


図 2-1 GTM 全体ブロック

(1) ARU(Advanced Routing Unit)は、GTM-IP の中核となる要素で、接続されている各サブモジュール間のデータルーティングを実現します。

(2) TIM(Timer Input Module)は、入力信号を測定する機能を持ち、エッジカウント、PWM 測定、信号時間測定等が可能です。

(3) ATOM(ARU-Connected Timer Output Module)は、信号を出力する機能を持ち、PWM 波形や High／Low 状態を指定された期間保持した信号等が出力可能です。

(4) DTM(Dead Time Module)は、ATOM より受信した信号に対して、デッドタイムの付加した信号の出力等が可能です。

(5) CMU(Clock Management Unit)は、外部より供給されたクロックを任意に分周し、各サブモジュールへ供給します。

(6) CCM(Cluster Configuration Module)は、CCM は各クラスタのサブモジュールで使用するクロックを設定可能です。

(7) TBU(Time Base Unit)は、CMU より供給されたクロックをもとにカウントを行い、タイムスタンプとして、TIM や ATOM 等のサブモジュールへ供給します。

(8) MCS(Multi Channel Sequencer)は、RISC のように四則演算や論理演算などが可能で、ARU を経由し、TIM からの測定結果の受け取りや、ATOM の信号出力の制御等が可能です。

(9) AEIMux(AE-Interface)は、各サブモジュールと外部バス(マイコン)とのインタフェースで、外部バスより各サブモジュールの初期化や各サブモジュールのレジスタの読み書きを可能にします。

(10) ICM(Interrupt Concentrator Module)は、各サブモジュールの割り込みを束ねて、外部モジュール(マイコン)への割り込み信号を生成します。

(11) CMP(Output Compare Unit)は、ATOM 出力信号(ただし、0～最大 2 までの 3 モジュール)にて出力チャネルを 2 本冗長的に使用した場合に、その 2 本の組み合わせで信号を比較します。

(12) MON(Monitor Unit)は、CMU のクロック出力や CMP の比較結果、ARU のラウンドトリップ時間、および MCS の動作状況を監視します。

2.2 用語集

本章以降で常用している用語や、略語に対する解説を表 2-1 に示します。

表 2-1 用語集

用語	意味
クラスタ	GTM の複数のモジュールで構成される機能ブロックです。 詳細は U2A HW UM 38.13 Cluster Configuration Module (CCM)を参照してください。
モジュール	TIM や ATOM、DTM 等、内部に複数のユニットやモジュールを持つ機能ブロックです。 なお、文章中では TIM や ATOM は GTM 内のモジュールという意味でサブモジュールという単語を使用しています。
ユニット	CMU や TBU、ARU 等、単体で機能する機能ブロックです。
サブユニット	TIM や ATOM、DTM 等のサブモジュール内部にて、単体で機能する機能ブロックです。

2.3 レジスタの表記等について

GTM はモジュールの特性上、マイコンによって内部の各ユニット数や各サブモジュール数、各サブモジュールの持つチャネル数などが異なる場合があります。また、GTM モジュールそのものが複数搭載されている場合もあります。このため、文中で表現するレジスタ名称は次のルールで表記します。

GTM[g].サブモジュールまたはユニット名[i]_CH[x]_レジスタ名[y]

- [g] は搭載される GTM モジュール番号です。
  ([g] は 0 以上、上限はマイコンおよび端子数により異なります。)
- [i] は搭載されるサブモジュール番号、またはユニット番号です。
  ([i] は 0 以上、上限はマイコンおよび端子数により異なります。)
- [x] は搭載されるサブモジュールが持つチャネル番号です。
  ([x] は 0 以上、上限はマイコンおよび端子数により異なります。)
  チャネルが存在しない場合、この部分は省略します。
- [y] は搭載されるサブモジュールが持つレジスタの番号です
  ([y] は 0 以上、上限はサブモジュールにより異なります。)
  レジスタが存在しない場合、この部分はレジスタ名も含めて省略します。

DTM サブモジュールはモジュール番号のナンバリングが特殊なため、以下のように表記します。

GTM[g].CDTM[i]_DTM[j]_レジスタ名

- [g] は搭載される GTM モジュール番号です。
  ([g] は 0 以上、上限はマイコンおよび端子数により異なります。)
- [i] は搭載される DTM サブモジュールの番号です。
  ([i] は 0 以上、上限はマイコンおよび端子数により異なります。)
- [j]は搭載される DTM サブユニットの番号です。
  ([j]は 4 or 5)

代表的なサブモジュール、またはユニットのレジスタの表記例を以下に記述します。

- GTM0、MCS1、チャネル番号 2 の汎用レジスタ R5 の場合

  GTM0.MCS1_CH2_R5

- GTM0、TIM0、チャネル 1 の CTRL レジスタの場合

  GTM0.TIM0_CH1_CTRL

- GTM0 の CMU_CLK_EN レジスタの場合

  GTM0.CMU_CLK_EN

また、信号線の表記についても、レジスタ名称と同様のルールで表記します。

- CMU_CLK のチャネル 2 の場合

  CMU_CLK2

- TIM1、チャネル 2 の NEWVAL 割り込みの場合

  TIM1_NEWVAL2_IRQ

各サブモジュールや各ユニットのレジスタ詳細等についてはユーザーズマニュアルを参照してください。

3. クロックおよびタイムスタンプ機能

GTM は、各サブモジュールへクロックを供給する機能と、クロック供給を受けてカウントしたタイムスタンプ値を、各サブモジュールへ供給する機能が搭載されています。(U2A では、タイムスタンプを供給するのは、TIM、ATOM、および MCS のみとなります。

3.1 動作クロック

U2A のリセット解除後、GTM の動作クロックが停止しているため U2A 搭載の STBC (Stanby Controller)機能により GTM に動作クロックを供給します。

GTM に動作クロックを供給するプログラムコード例を以下に示します。

```
SYSCTRL.MSRKCPROT.UINT32 = 0xA5A5A501;          // writing protection is invalid
SYSCTRL.MSR_GTM.BIT.MS_GTM_0 = 0;               // GTM0 is operating
while(SYSCTRL.MSR_GTM.BIT.MS_GTM_0 != 0);
SYSCTRL.MSRKCPROT.UINT32 = 0xA5A5A500;          // writing protection is valid
```

図 3-1 STBC 設定プログラムコード例

3.2 クラスタクロック

クラスタ k で使用するクラスタクロック (CLS[k]_CLK) を分周します。GTM[g].GTM_CLS_CLK_CFG レジスタの CLS[k]_CLK_DIV ビット設定により、クラスタクロックは Cluster k Clock Divider で 1 分周または 2 分周するか、停止することが可能です。

表 3-1 GTM[g].GTM_CLS_CLK_CFG 設定項目一覧

ビット名	説明
CLK[k]_CLK_DIV	Cluster k Clock Divider 00_B：クラスタ無効(クロック停止) 01_B：クラスタ有効(クロック 1 分周) 10_B：クラスタ有効(クロック 2 分周) 11_B：設定禁止 CLS0_CLK_DIV は CMU のプライマリ入力クロックです。 CLS0_CLK_DIV=10_Bに設定した場合、クラスタ 1～3 の CMU クロック周波数の最大はクラスタ 0 の CMU クロック周波数の最大に制限されます。

クラスタ 0～3 のクラスタクロックを GTM の動作クロックの 2 分周に設定するプログラムコード例を以下に示します。

```
GTM0.GTM_CTRL.BIT.RF_PROT = 0;                  // writing protection is invalid
GTM0.GTM_CLS_CLK_CFG.UINT32 = 0x000000AA; //CLS0_CLK = MainClock/2
                                                //CLS1_CLK = MainClock/2
                                                //CLS2_CLK = MainClock/2
                                                //CLS3_CLK = MainClock/2
while(GTM0.GTM_CLS_CLK_CFG.UINT32 != 0x000000AA);
GTM0.GTM_CTRL.BIT.RF_PROT = 1;                  // writing protection is valid
```

図 3-2 クラスタクロック設定プログラムコード例

3.3 Clock Management Unit (CMU)

CMU は、GTM0.GTM_CLS_CLK_CFG レジスタの CLS0_CLK_DIV ビットで定義されるクラスタ 0 クロック(CLS0_CLK)より GTM 内で使用するクロックを生成します。CFGU は 8 チャネルのクロック(CMU_CLK[k] k=0-7)を生成し、TIM、ATOM、TBU、MON、DTM へ供給されます。EGU は 3 つのクロックを生成し、CMU_ECLK[z]端子(z=0-2)に表示します。また EGU は CMU_CLK8 を生成し CCM へ供給します。

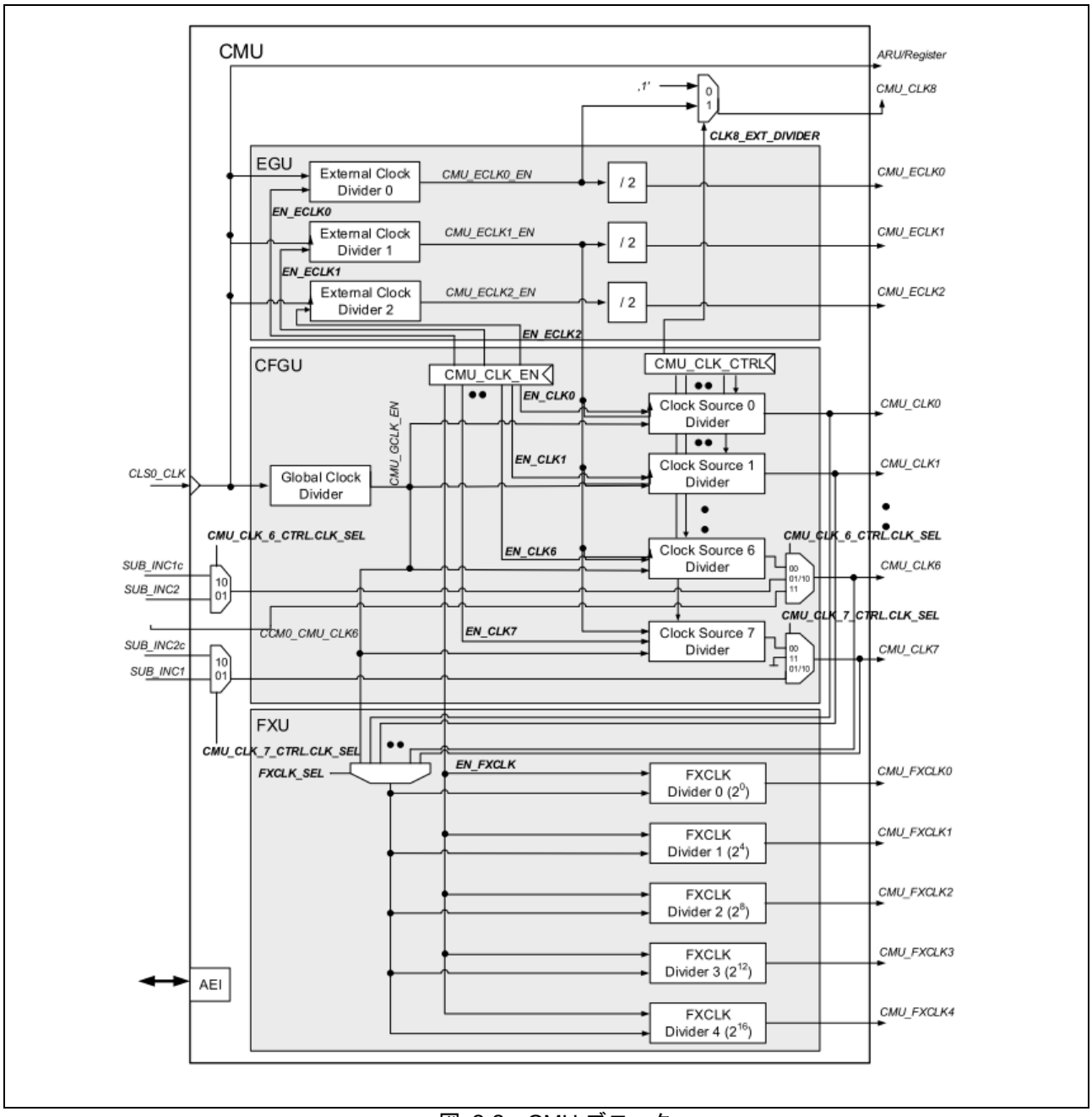


図 3-3 CMU ブロック

※ U2A は DPLL 非搭載のため SUB_INT1c,SUB_INTC2,SUB_INTC2c,SUB_INT1 は使用できません。

※ U2A は TOM 非搭載のため FXU は使用できません。

(1) Global Clock Divider
各 Clock Source Divider へ供給するクロックを生成します。計算式は以下のとおりです。

CMU_GCLK_EN = CLS0_CLK / N
N = CMU_GCLK_NUM / CMU_GCLK_DEN

CMU_GCLK_NUM が CMU_GCLK_DEN より小さい(N が 1 未満になる)場合、または CMU_GCLK_NUM か CMU_GCLK_DEN のいずれかが 0 の場合は、CMU のハードウェアにより、CMU_GCLK_NUM と CMU_GCLK_DEN に 1 が設定されます。

CLS0_CLK にて供給されるクロックは U2A では最大 160MHz です。
CMU_GCLK_NUM は GTM[g].CMU_GCLK_NUM レジスタにて、CMU_GCLK_DEN は GTM[g].CMU_GCLK_DEN レジスタにて設定可能です。

例えば、CLS0_CLK=80[MHz]、CMU_GCLK_NUM=2、CMU_GCLK_DEN=1 の場合、計算式は以下のようになり、CMU_GCLK_EN=40[MHz]となります。

N = CMU_GCLK_NUM / CMU_GCLK_DEN = 2 / 1 = 2
CMU_GCLK_EN = CLS0_CLK / N = 80 / 2 = 40

(2) Clock Source x Divider
Global Clock Divider より供給されたクロックを分周します。計算式は以下のとおりです。

- CMU_CLK[x] (x =0-5,7)

CMU_CLK[x]は Clock Source x Divider をソースクロックに選択することができます。

GTM[g].CMU_CLK_CTRL .CLK[x]_EXT_DIVIDER ビット	CMU_CLK[x]
0	CMU_GCLK_EN / (CLK_CNT[x] + 1)
1	CMU_ECLK1_EN / (CLK_CNT[x] + 1)

CLK_CNT[x] は、CMU_CLK[x] のクロック分周のカウント値です。
GTM[g].CMU_CLK_[x]_CTRL.CLK_CNT ビットにて設定可能です。

例えば、チャネル番号=2、CMU_CLK_CTRL.CLK2_EXT_DIVIDER ビット=0、CMU_GCLK_EN=40[MHz]、CLK_CNT2=3、の場合、計算式は以下のようになり、CMU_CLK2=10[MHz]となります。

CMU_CLK2 = CMU_GCLK_EN / (CLK_CNT2 + 1) = 40 / (3 + 1) = 10

- CMU_CLK6

CMU_CLK6 は Clock Source6 Diveider または CCM0 経由で CMU_CLK7 をソースクロックに選択することができます。

CMU_CLK_CTRL .CLK6_EXT_DIVIDER ビット	CMU_CLK_6_CTRL .CLKSEL ビット	CMU_CLK6
0	00	CMU_GCLK_EN / (CLK_CNT6 + 1)
1		CMU_ECLK1_EN / (CLK_CNT6 + 1)
-	11	CCM0_CMU_CLK6 (=CMU_CLK7)

CLK_CNT6 は、CMU_CLK6 のクロック分周のカウント値です。
GTM[g].CMU_CLK_6_CTRL.CLK_CNT ビットにて設定可能です。

(3) CMU_CLK8
CMU_CLK8 は CLS0_CLK または External Clock Divider0 をソースクロックに選択することができます。

CMU_CLK_CTRL .CLK8_EXT_DIVIDER ビット	CMU_CLK8
00	CLS0_CLK
11	CMU_ECLK0

(4) External Clock Divider z (z=0-2)
CMU_ECLK[z]、CMU_CLK8 へ供給するクロックを生成します。計算式は以下のとおりです。

CMU_ECLK[z]_EN = CLS0_CLK / N
N = CMU_ECLK_[z]_NUM / CMU_ECLK_[z]_DEN

CMU_ECLK_[z]_NUM が CMU_GCLK_[z]_DEN より小さい(N が 1 未満になる)場合、または CMU_ECLK_[z]_NUM が CMU_GCLK_[z]_DEN のいずれかが 0 の場合は、CMU のハードウェアにより、CMU_ECLK_[z]_NUM と CMU_GCLK_[z]_DEN に 1 が設定されます。

CMU_ECLK_[z]_NUM は GTM[g].CMU_ECLK_[z]_NUM. ECLK_NUM ビット、
CMU_ECLK_[z]_DEN は GTM[g].CMU_ECLK_[z]_DEN. ECLK_DEN ビットによって設定可能です。

例えば、チャネル番号 2、CLS0_CLK=80[MHz]、CMU_ECLK_2_NUM=2、CMU_ECLK_2_DEN=1 の場合、計算式は以下のようになり、CMU_ECLK2_EN=40[MHz]となります。

N = CMU_ECLK_2_NUM / CMU_ECLK_2_DEN = 2 / 1 = 2
CMU_ECLK_2_EN = CLS0_CLK / N = 80 / 2 = 40

(5) CMU_CLK_EN
各 Clock Source Divider に対してクロック供給の有効／無効を設定します。GTM[g].CMU_CLK_EN レジスタの EN_CLK[x] (x=0-7)ビット、EN_ECLK[z] (z=0-2)ビットにて設定可能です。
(1)の設定を行う場合、本設定値にて全チャネルの設定を無効にしてください。
(2)の設定を行う場合、本設定値にて対象チャネルの CMU_CLK[x]と ECLK1 の設定を無効にしてください。

各レジスタの詳細は、ユーザーズマニュアルを参照してください。

各サブモジュールにて使用するクロックチャネルは、任意のチャネルが選択可能です。ただし、次に示すサブユニットやサブモジュールはクロックチャネル選択に制限があり、一部のクロックチャネルのみが選択可能です。

- TIM の FLT[x]
  CMU_CLK0、CMU_CLK1、CMU_CLK6、CMU_CLK7 のみ選択可能です。
- DTM
  CMU_CLK0、CMU_CLK1、CMU_CLK2 のみ選択可能です。

CMU 設定のプログラムコード例を以下に示します。

```
/* setting all CMU channel to invalid */
GTM0.CMU_CLK_EN.UINT32 = 0x00155555;
while(GTM0.CMU_CLK_EN.UINT32 != 0) {
      // wait until CMU_CLK_EN becomes 0x0 (All CMU channel is invalid)
};

/* setting Global Clock Divider (divide cls0_clk by 2 as CMU_GCLK_EN) */
GTM0.CMU_GCLK_NUM.BIT.GCLK_NUM = 2;
GTM0.CMU_GCLK_DEN.BIT.GCLK_DEN = 1;

/* setting Clock Source x Divider */
GTM0.CMU_CLK_0_CTRL.BIT.CLK_CNT = 0; // divide CMU_GCLK_EN by 1 as CMU_CLK0
GTM0.CMU_CLK_1_CTRL.BIT.CLK_CNT = 1; // divide CMU_GCLK_EN by 2 as CMU_CLK1
GTM0.CMU_CLK_2_CTRL.BIT.CLK_CNT = 2; // divide CMU_GCLK_EN by 3 as CMU_CLK2
GTM0.CMU_CLK_3_CTRL.BIT.CLK_CNT = 3; // divide CMU_GCLK_EN by 4 as CMU_CLK3
GTM0.CMU_CLK_4_CTRL.BIT.CLK_CNT = 4; // divide CMU_GCLK_EN by 5 as CMU_CLK4
GTM0.CMU_CLK_5_CTRL.BIT.CLK_CNT = 5; // divide CMU_GCLK_EN by 6 as CMU_CLK5
GTM0.CMU_CLK_6_CTRL.BIT.CLK_CNT = 6; // divide CMU_GCLK_EN by 7 as CMU_CLK6
GTM0.CMU_CLK_7_CTRL.BIT.CLK_CNT = 7; // divide CMU_GCLK_EN by 8 as CMU_CLK7

GTM0.CMU_CLK_CTRL.UINT32 = 0x00000000; // select CMU_GCLK_EN as source clock of
                                       // Clock Source x Divider (x=0-5,7)
                                       // select cls0_clk as CMU_CLK8
/* setting External Clock Divider */
GTM0.CMU_ECLK_0_NUM.BIT.ECLK_NUM = 1; // divide cls0_clk by 1 as CMU_ECLK0
GTM0.CMU_ECLK_0_DEN.BIT.ECLK_DEN = 1;

GTM0.CMU_ECLK_1_NUM.BIT.ECLK_NUM = 2; // divide cls0_clk by 2 as CMU_ECLK1
GTM0.CMU_ECLK_1_DEN.BIT.ECLK_DEN = 1;

GTM0.CMU_ECLK_2_NUM.BIT.ECLK_NUM = 3; // divide cls0_clk by 3 as CMU_ECLK2
GTM0.CMU_ECLK_2_DEN.BIT.ECLK_DEN = 1;

/* setting All CMU channel to valid */
GTM0.CMU_CLK_EN.UINT32 = 0x002AAAAA;
while(GTM0.CMU_CLK_EN.UINT32 != 0x003FFFFF) {
      // wait until CMU_CLK_EN becomes 0x003FFFFF (All CMU channel is valid)
};
```

図 3-4 CMU 設定プログラムコード例

3.4 Cluster Configuration Module (CCM)

CCM は各クラスタのサブモジュールで使用するクロックを設定可能です。

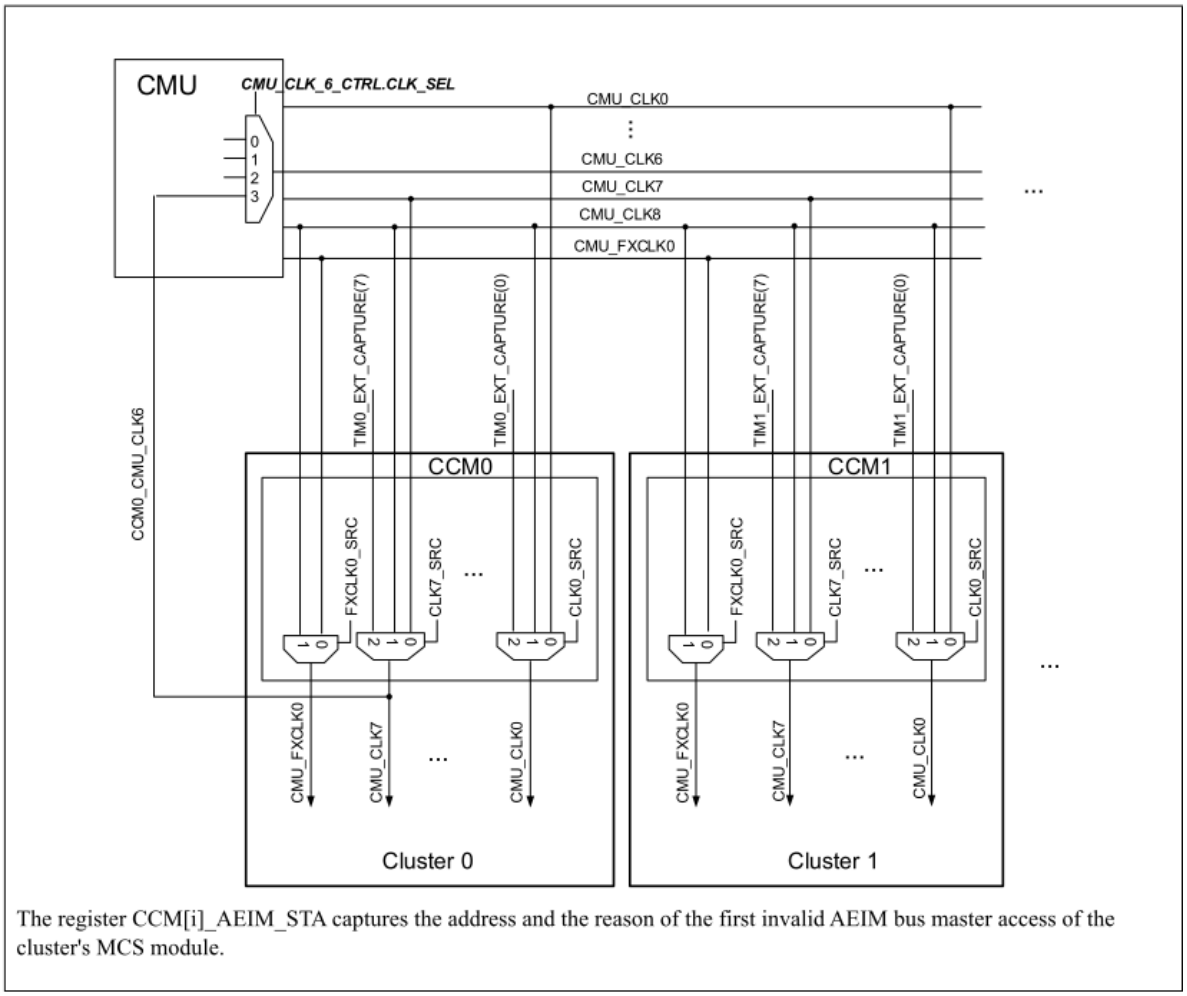


図 3-5 クラスタクロック信号接続

※U2A では TOM 非搭載のため CMU_FXCLK0 は使用できません。

※U2A では DPLL 非搭載のため CMU_CLK_6_CTRL.CLK_SEL ビット=1、2 は使用できません。

(1) CMU_CLK[x]
クラスタ i の CMU_CLK[x]クロックです。GTM[g].CCM[i]_CMU_CLK_CFG レジスタの CLK[x]_SRC ビットにて以下のいずれかから選択可能です。

- CMU の CMU_CLK[x]
- CMU の CMU_CLK8
- TIM[i]_EXT_CAPTURE[x]

その他の機能の詳細は、ユーザーズマニュアルを参照してください。

3.5 Time Base Unit (TBU)

TBU は CMU より供給されたクロックをもとにタイムスタンプのカウントを行い、GTM 全体の共通時間カウンタ値として、タイムスタンプを各サブモジュール(ATOM、TIM、MCS 等)へ提供します。U2A の場合、TBU は 3 チャネルで構成され、TBU チャネル 0 は 27 ビットのカウンタ、TBU チャネル 1、2 は 24 ビットのカウンタとなります。

TBU は 24 ビットの信号にて各サブモジュールへタイムスタンプを供給するため、TBU チャネル 0 の出力については、27 ビットカウンタの下位 24 ビット出力、または上位 24 ビット出力を選択する必要があります。

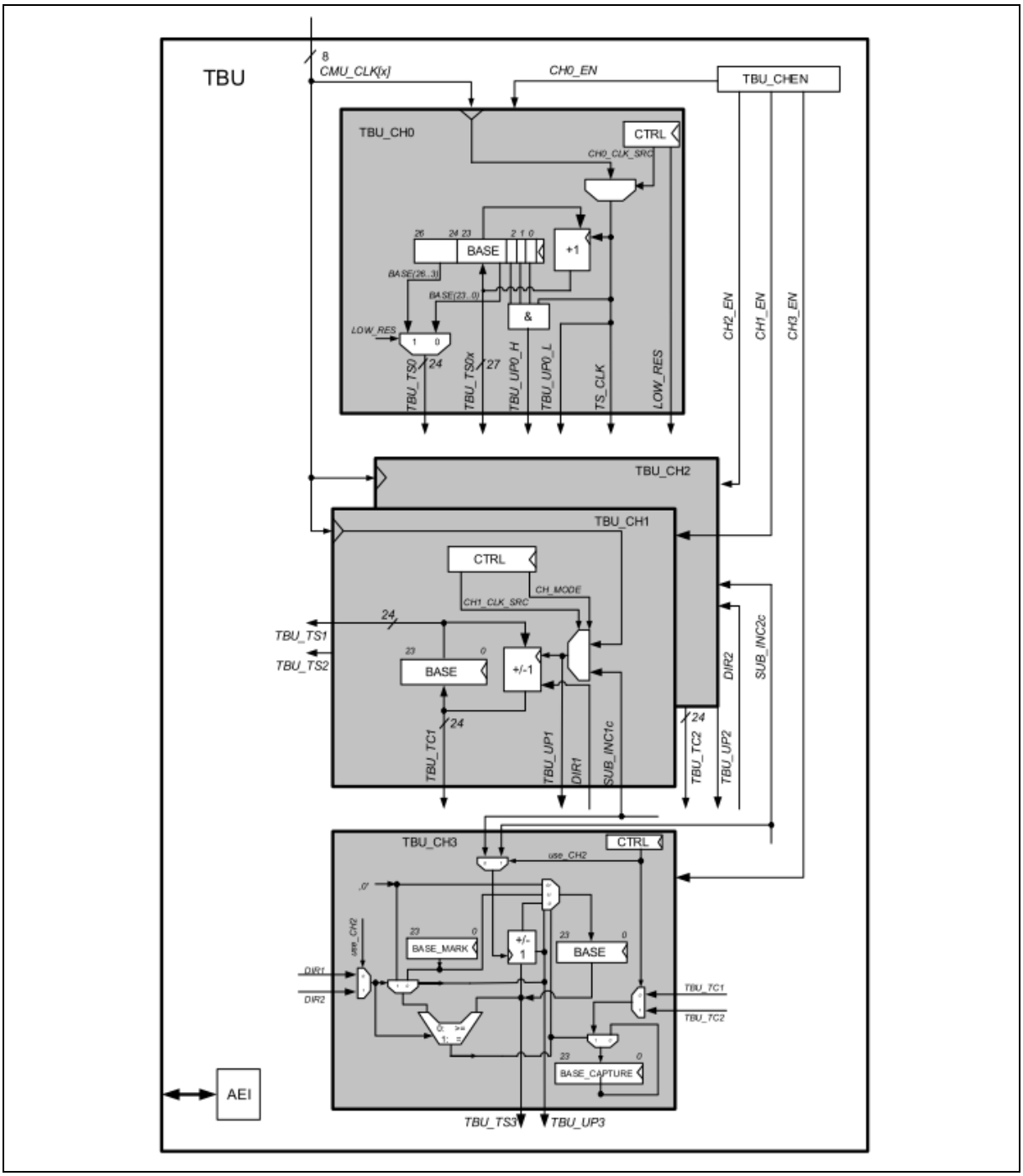


図 3-6 TBU ブロック

※ U2A は DPLL 非搭載のため SUB_INT1c、SUB_INTC2c、DIR1、DIR2 は使用できません。このため TBU CH1、2 の Forward / Backward Count Mode、TBU_CH3 は使用できません。

(1) TBU_CHEN
TBU のチャネル制御レジスタです。TBU の各チャネルの有効／無効を設定します。
GTM[g].TBU_CHEN レジスタにて設定可能です。
(2)～(4)の設定を行う場合、本設定値にて対象チャネルの設定を無効にしてください。

(2) TBU_CH0_CTRL
TBU チャネル 0 の制御レジスタです。TBU チャネル 0 にて使用するクロックチャネル、カウンタ値の出力方法(上位 24 ビット出力または下位 24 ビット出力)を設定します。
GTM[g].TBU_CH0_CTRL レジスタにて設定可能です。

(3) TBU_CH1_CTRL、TBU_CH2_CTRL
TBU チャネル 1、2 の制御レジスタです。TBU チャネル 1、2 にて使用するクロックチャネルを設定します。GTM[g].TBU_CH1_CTRL、GTM[g].TBU_CH2_CTRL にて設定可能です。

(4) BASE
各 TBU チャネルのカウンタです。本値をタイムスタンプとして各サブモジュールに提供します。
TBU チャネル 0 は GTM[g].TBU_CH0_BASE にて、TBU チャネル 1 は GTM[g].TBU_CH1_BASE にて、TBU チャネル 2 は GTM[g].TBU_CH2_BASE にて初期値を設定することも可能です。

各レジスタの詳細は、ユーザーズマニュアルを参照してください。

TBU チャネルを停止する場合は、ATOM サブモジュール、および TIM サブモジュールより参照されていないことを確認した後に停止してください。ATOM の場合、タイムスタンプを使用した比較等を行う場合もありますので、意図しない信号が出力される可能性があります。

TBU 設定のプログラムコード例を以下に示します。

```
GTM0.TBU_CHEN.UINT32 = 0x15;            // All TBU channels are invalid
while(GTM0.TBU_CHEN.UINT32 != 0){
   // wait until CMU_CLK_EN becomes 0x0 (All TBU channels are invalid)
};

GTM0.TBU_CH0_CTRL.BIT.CH_CLK_SRC = 0;   // select CMU_CLK0 as TBU channel 0 clock
                                        // select lower counter bit (24bit)
GTM0.TBU_CH1_CTRL.BIT.CH_CLK_SRC = 1;   // select CMU_CLK1 as TBU channel 1 clock
GTM0.TBU_CH2_CTRL.BIT.CH_CLK_SRC = 2;   // select CMU_CLK2 as TBU channel 2 clock

GTM0.TBU_CH0_BASE.UINT32 = 0x0;     // initialize the counter value of TBU channel 0
GTM0.TBU_CH1_BASE.UINT32 = 0x0;     // initialize the counter value of TBU channel 1
GTM0.TBU_CH2_BASE.UINT32 = 0x0;     // initialize the counter value of TBU channel 2

GTM0.TBU_CHEN.UINT32 = 0x2a;          // All TBU channels are valid
while( GTM0.TBU_CHEN.UINT32 != 0x3f ){
   //wait until TBU_CHEN becomes 0x3f (All TBU channels are valid)
};
```

図 3-7 TBU 設定プログラムコード例

4. ARU 転送

ARU はサブモジュール間のデータルーティングを行います。基本的には MCS、ATOM、TIM 間のデータ転送が対象となります。ただし、TIM はデータ送信のみとなり、ARU を経由して他のサブモジュールからデータを受信することはできません。

ARU は内部にバッファを持っており、あらかじめサブモジュールのチャネルごとに領域を割り当てています。データ転送元は割り当てられたバッファにデータを書き込むことによりデータ送信を行い、データ転送先は任意のバッファからデータを読み込むことによりデータ受信を行います。データは受信されると、バッファ内から削除されます。

バッファの割り当て(書き込み先アドレス)の詳細はユーザーズマニュアルを参照してください。

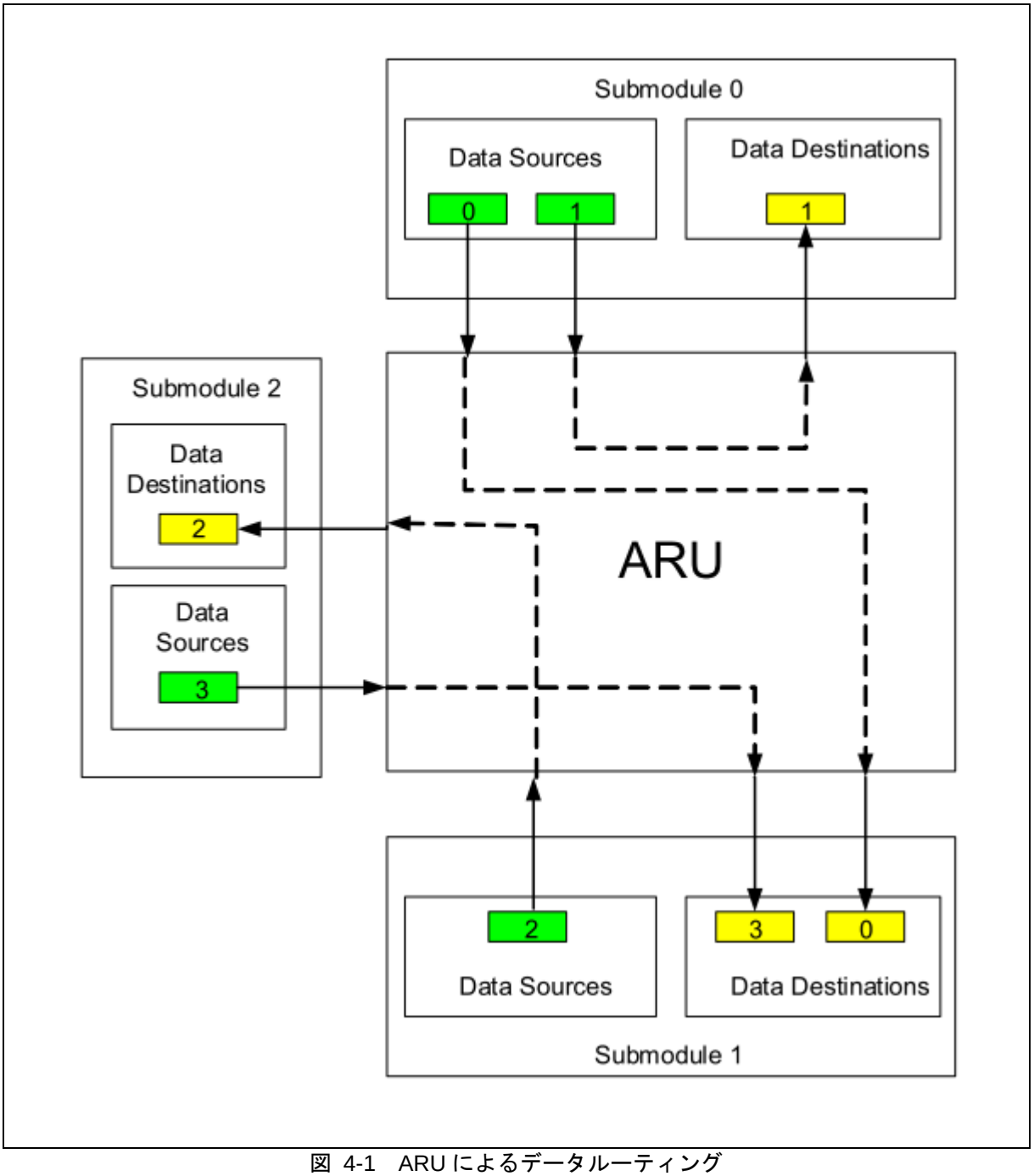


図 4-1 ARU によるデータルーティング

ARU のデータ受信はラウンドロビン方式で行われ、転送先ごとにデータ受信が 1 回ずつ順番に実行されます。

ARU では、転送処理によるラウンドトリップ時間が発生します。

ラウンドトリップ最大時間の計算式は以下のとおりです。

ラウンドトリップ最大時間 = 1 / SYS_CLK * (N + 1)

N：ARU から各サブモジュールへのチャネル数(データ転送先となりうるチャネル数)

例えば、SYS_CLK=80[MHz]、ARU から各サブモジュールへ接続されるチャネル数=21 の場合、以下のようになり、ラウンドトリップ最大時間=0.000000275[s]= 275[ns]となります。

ラウンドトリップ最大時間 = 1 / SYS_CLK * (N + 1) = 1 / 80,000,000 * (21 + 1) = 0.000000275

U2A に実装される GTM-IP 358 は、ARU は 2 ポート実装されています。GTM-IP 358 にて、ARU より各サブモジュールへ接続されるチャネル数は次の表のとおりです。

表 4-1 GTM-IP 358 における ATOM、MCS の ARU ポート接続

サブモジュール名	ARU ポート 0	ARU ポート 1
ATOM0	8	-
ATOM1	-	8
ATOM2	8	-
ATOM3	-	8
MCS0	8	-
MCS1	-	8
MSC2	8	-
MCS3	-	8

データ送信は任意のタイミングで行うことが出来るため、各サブモジュールから ARU へ接続されるチャネルについては、ARU ラウンドトリップ時間に影響はありません。

4.1 データフォーマット

ARU で送受信されるデータは、53bit 単位です。

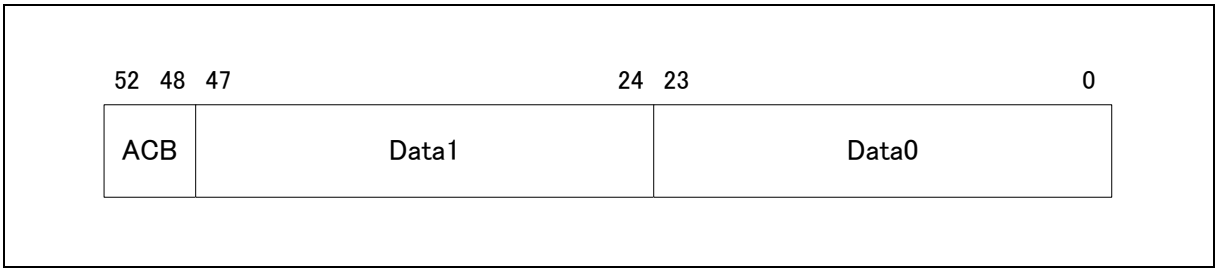


図 4-2 ARU データワードフォーマット

ACB、Data0、Data1 は、データ転送元やデータ転送先によって格納される情報は異なります。

例えば、TIM の場合は、ACB にはエラーの発生状況等が、Data0 と Data1 には測定結果が格納されます。

4.2 ブロッキングモード

ARU でのデータ転送では、ブロッキングモードとして、ブロッキングとノンブロッキングが存在します。

ブロッキングでは、データ転送元は任意のタイミングで ARU へデータを送信し、データ転送先は任意のタイミングでデータを受信します。このとき、データ転送元は、ARU にデータを送信してからデータ転送先がデータを受信するまで、次のデータを送信することは出来ません。

ノンブロッキングでは、データ転送元は任意のタイミングで ARU へデータを送信しますが、データ転送先は送信されたデータをすぐに受信します。そのため、データ転送元はすぐに次のデータを送信することが出来ます。

ATOM では、特定のイベントが発生した際のデータのみが出力に反映されます。ブロッキングの場合、特定のイベントが発生した時にデータ受信を行うため、基本的に受信したデータは出力に反映されます。しかし、ノンブロッキングの場合、データ転送元がデータを送信したタイミングでデータ受信を行うため、特定のイベントが発生した時点における最後に受信したデータ以外は、読み捨てられることとなり、出力に反映されません。

MCS では、ARU へのアクセス命令により、データ受信についてのブロッキング、ノンブロッキングが指定できます。ノンブロッキングを指定すると、データが格納されていない場合は、参照失敗として命令が終了します。

5. 信号入力機能

5.1 構成

5.1.1 全体構成

TIM(Timer Input Module)は、GTM への入力信号をキャプチャする機能を持ち、波形計測や外部信号の変化の検出、マイコン外部の外付けデバイスから簡単なデータを受信すること等が可能です。どのようなデータを測定するのかはチャネルモードを切り替えることにより行います。また、フィルタ機能を搭載しており、信号のグリッチの検出／除去を行ってからキャプチャを行うことが出来ます。

測定／検出されたデータ(カウンタ値やタイムスタンプ値等)は ARU を介し、各サブモジュールへ転送する事が可能です。

TIM は、1 モジュールにつき最大 8 チャネル(注1)搭載が可能です。

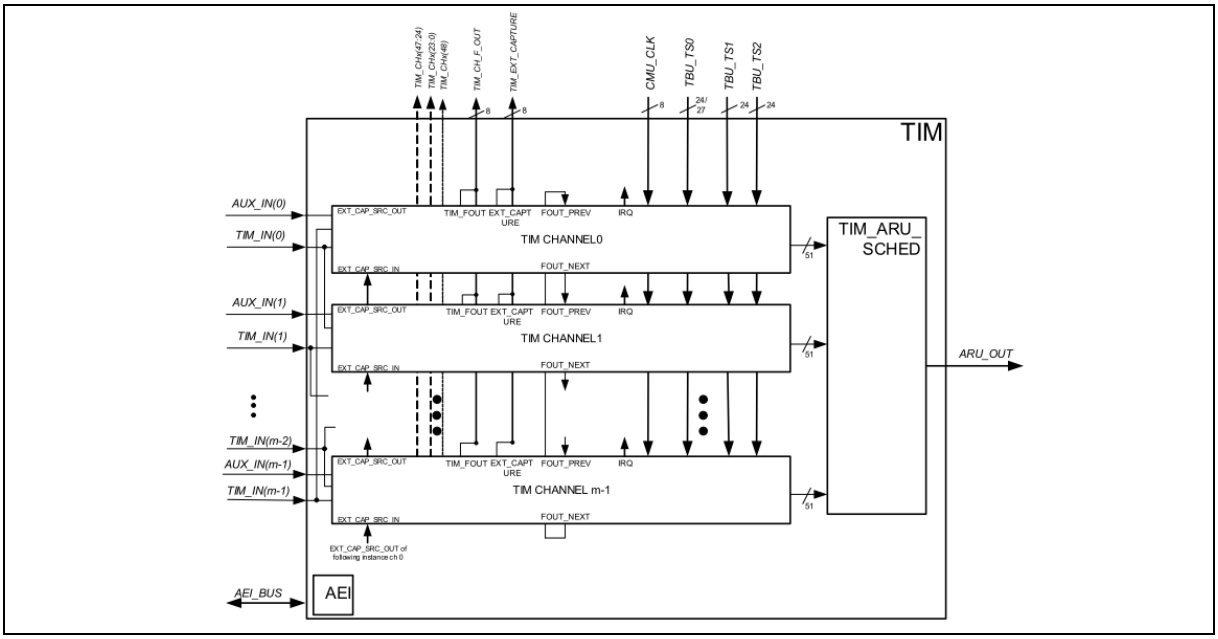


図 5-1 TIM ブロック

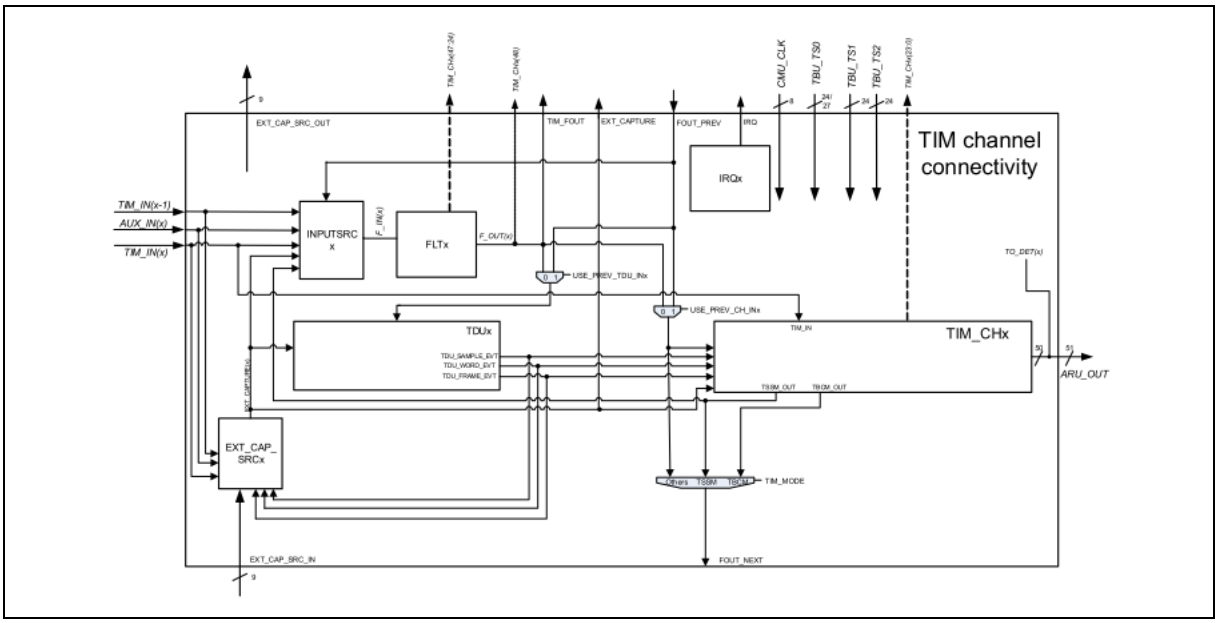


図 5-2 TIM チャネル内部接続

(注 1) 使用可能なチャネル数は、マイコンおよび端子数により異なります。

(1) INPUT_SRC[x]
タイマチャネルへの入力信号を選択します。

(2) FLT[x]
フィルタです。設定によりグリッチの検出／除去を実施します。

(3) TDU[x]
フィルタを経由した入力信号のタイムアウトを監視／検出します。

(4) EXTCAPSRC[x]
入力信号をキャプチャするためのトリガを選択します。

(5) TIM_CH[x]
波形の測定を実施します。詳細は「5.1.2 タイマチャネル部」を参照してください。

(6) TIM_ARU_SCHED
TIM_CH[x]の測定結果を ARU へ転送します。

5.1.2 タイマチャネル部

設定されたチャネルモードにて入力信号を測定します。測定結果は GPR0 レジスタ、および GPR1 レジスタへ格納されます。

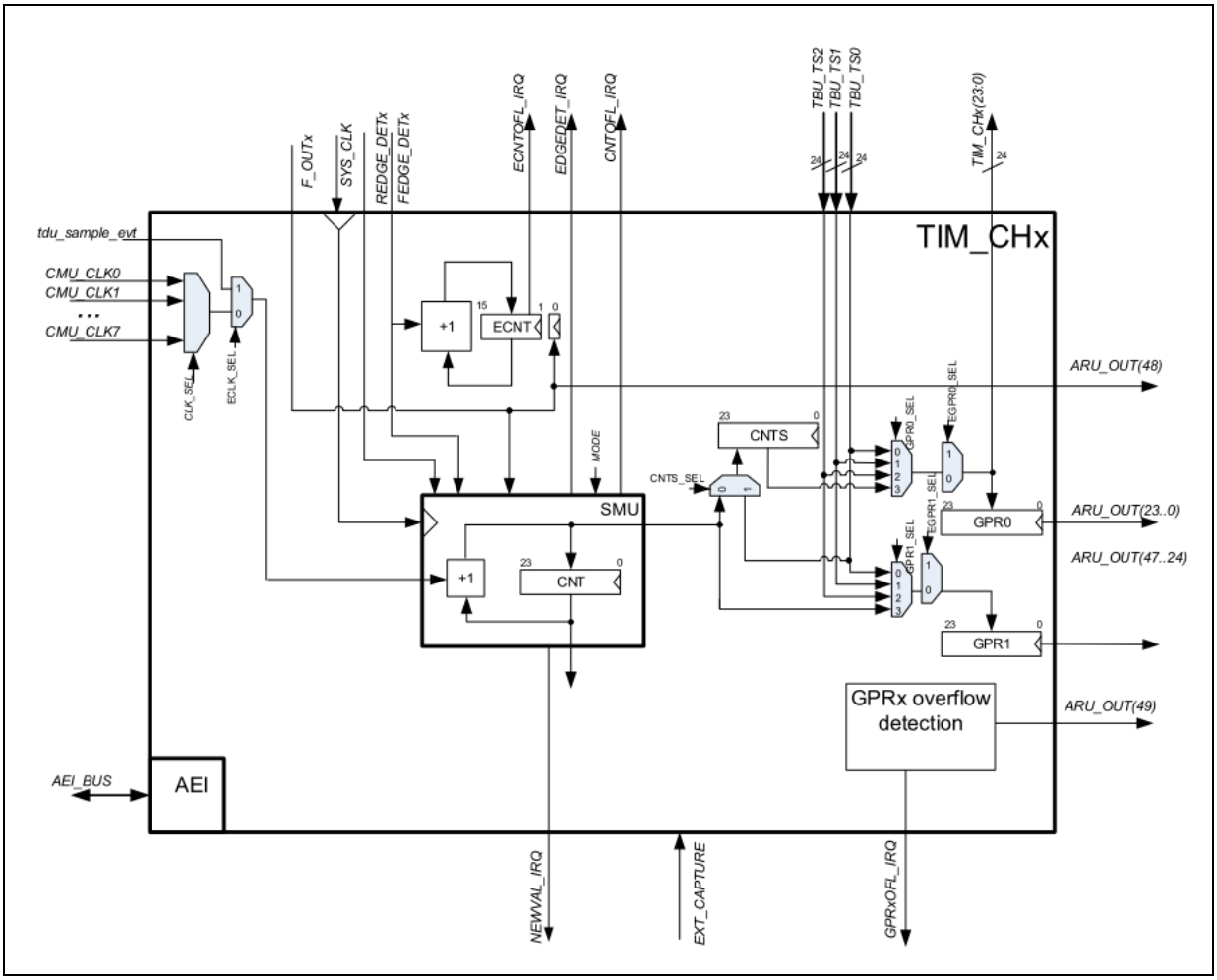


図 5-3 TIM チャネルブロック

(1) ECNT

16 ビットのエッジカウンタです。

上位 15 ビットは立ち下がりエッジをカウントし、下位 1 ビットは現在の信号レベルとなります。

従って、ロウレベルの状態でカウントをスタートした場合は 0 からカウントを開始しますが、ハイレベルの状態でカウントをスタートした場合は 1 からカウントが開始されます。

カウンタ値は GTM[g].TIM[i]_CH[x]_ECNT レジスタにて参照可能ですが、本レジスタを参照した場合、ECNT は自動的にクリアされます。また、GTM[g].TIM[i]_CH[x]_ECNT レジスタの下位 8 ビットは GTM[g].TIM[i]_CH[x]_CNTS レジスタ、GTM[g].TIM[i]_CH[x]_GPR0 レジスタ、および GTM[g].TIM[i]_CH[x]_GPR1 レジスタのビット 24～31 にて参照可能です。

カウンタがオーバフローした場合、TIM[i]_ECNTOFL[x]_IRQ 信号を生成します。

(2) SMU
信号測定ユニットです。指定されたチャネルモードにて信号を測定します。測定結果は GPR0 レジスタと GPR1 レジスタに格納します。測定結果を格納する度に TIM[i]_NEWVAL[x]_IRQ 信号を生成します。

(3) CNT
SMU 内のカウンタです。指定されたチャネルモードにより、エッジ数や信号継続時間等のカウントを行います。GTM[g].TIM[i]_CH[x]_CNT レジスタにて参照可能です。
カウンタがオーバフローした場合、TIM[i]_CNTOFL[x]_IRQ 信号を生成します。

(4) CNTS
CNT のシャドウレジスタです。GTM[g].TIM[i]_CH[x]_CNTS レジスタにて参照可能です。
使用用途は各チャネルモードにより異なります。

(5) GPRx Overflow Detection
GPR0 レジスタ、および GPR1 レジスタの監視ユニットです。各レジスタが読み出されることなく測定結果が更新された場合、TIM[i]_GPROFL[x]_IRQ 信号を生成します。
また、ARU 転送が有効な場合、ARU データの ACB の bit1 が設定されます。

(6) CTRL
以下の設定を実施します：
- 入力信号タイムアウト設定
- 入力フィルタ設定
- GPR0 レジスタへの格納内容の切り替え
  (測定結果、タイムスタンプ、エッジ数、チャネルの信号状態等)
- GPR1 レジスタへの格納内容の切り替え
  (測定結果、タイムスタンプ、エッジ数、チャネルの信号状態等)
- ARU への GPR0、および GPR1 レジスタ内容の転送の有効／無効の設定

その他の設定項目もあります。詳細はユーザーズマニュアルを参照してください。
GTM[g].TIM[i]_CH[x]_CTRL レジスタにて参照可能です。

(7) GPR0
(6)の GPR0 レジスタへの格納内容の切り替えで設定した内容が格納されます。格納内容は GTM[g].TIM[i]_CH[x]_GPR0 レジスタにて参照可能です。
また、ARU 転送が有効な場合、本レジスタの内容は ARU データの Data0 として ARU へ転送されます。

(8) GPR1
(6)の GPR1 レジスタへの格納内容の切り替えで設定した内容が格納されます。格納内容は GTM[g].TIM[i]_CH[x]_GPR1 レジスタにて参照可能です。
また、ARU 転送が有効な場合、本レジスタの内容は ARU データの Data1 として ARU へ転送されます。

各レジスタの詳細は、ユーザーズマニュアルを参照してください。

5.2 チャネルモード

TIM のチャネルモードの基本的な設定方法を説明します。

5.2.1 TIM PWM Measurement Mode (TPWM)

PWM 波形を測定し、周期カウンタ値とデューティカウンタ値を取得します。

測定を行う場合、GTM[g].TIM[i]_CH[x]_CTRL レジスタの表 5-1 の項目の設定が必要です。

表 5-1 TPWM における GTM[g].TIM[i]_CH[x]_CTRL レジスタ設定項目一覧

ビット名	説明
CLK_SEL	測定で使用する CMU_CLK のチャネル番号を設定します。 0～7：CMU チャネル番号
DSL	PWM 信号のデューティの信号レベルを設定します。 0：PWM のデューティは Low 1：PWM のデューティは High
TIM_MODE	チャネルモードとして TPWM を設定します。 0：TIM PWM Measurement Mode (TPWM)
EGPR0_SEL	GPR0 レジスタの格納内容を設定します。 CNTS(デューティカウンタ値)の内容を転送する場合は以下設定です。 EGPR0_SEL=0 GPR0_SEL=11
GPR0_SEL	
EGPR1_SEL	GPR1 レジスタの格納内容を設定します。 CNT(周期カウンタ値)の内容を転送する場合は以下設定です。 EGPR1_SEL=0 GPR1_SEL=11
GPR1_SEL	
TIM_EN	チャネルを有効にします。 1：Enabled

(1) SMU の CNT カウンタの動作
上記 CLK_SEL にて指定された CMU_CLK[x]のクロックにてカウントを開始し、入力信号を測定します。CNT カウンタは入力信号(PWM 信号)が 1 周期すると 0 クリアし、再度カウントを開始します。このとき、TIM[i]_NEWVAL[x]_IRQ 信号が生成されます。
例として、DSL ビットが 1 に設定された場合、入力信号で立ち上がりエッジが発生した際にカウントを開始し、次に入力信号が立ち上がりエッジになった際に、カウンタを 0 クリアしてからカウント再開となります。

(2) CNT より CNTS への転送タイミング
CNT カウンタ値は、デューティの終了時に CNTS へ転送されます。
例として、DSL ビットが 1 に設定された場合、入力信号で立ち下がりエッジが発生した際に CNT カウンタ値は CNTS へ転送されます。

(3) GPR0、GPR1 への転送タイミング
周期の終了時に、EGPR0_SEL、GPR0_SEL、EGPR1_SEL、GPR1_SEL で指定した内容が GPR0、GPR1 へ転送されます。

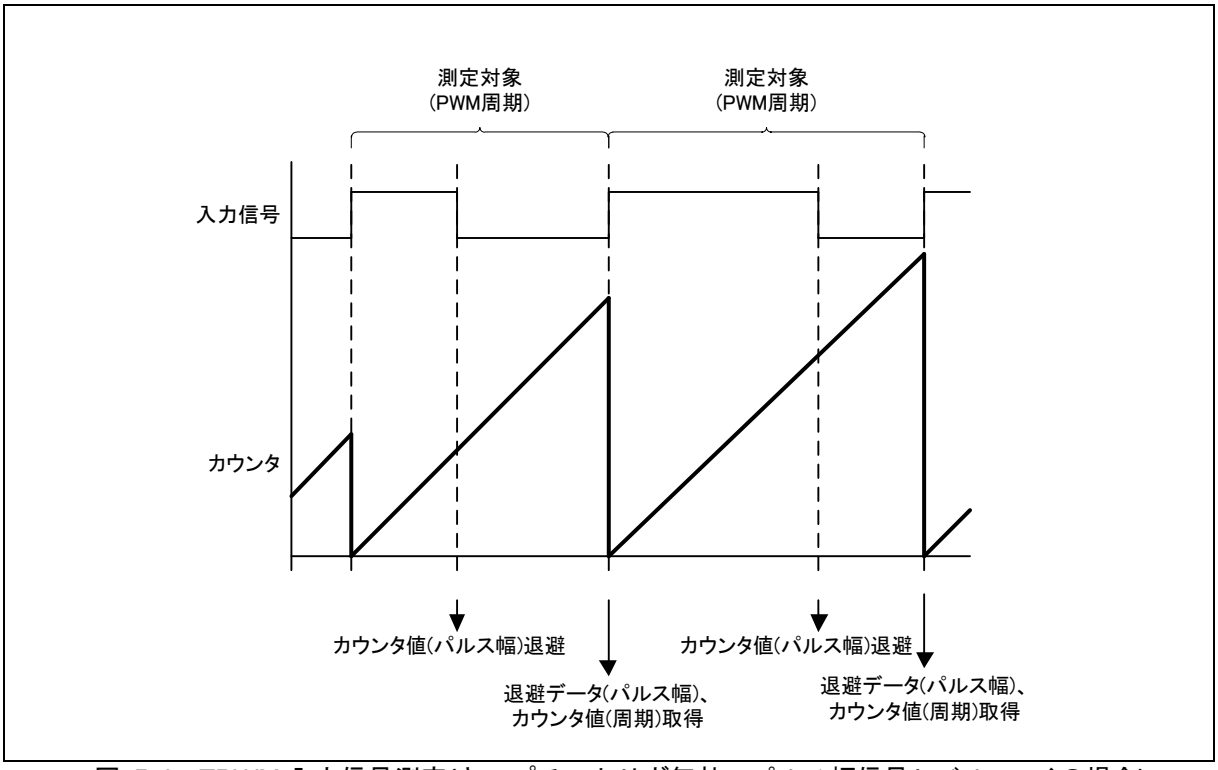


図 5-4 TPWM 入力信号測定(キャプチャトリガ無効、パルス幅信号レベル=ハイの場合)

(4) キャプチャトリガ有効 (EXT_CAP_EN=1)に設定し、EGPR0_SEL、GPR0_SEL で CNTS を転送する設定にすると、キャプチャトリガ検出時にパルス幅を取得することができます。

- ISL=0 の場合、デューティ終了時、CNT 値は CNTS に転送されるため、キャプチャトリガ検出時は直前のパルス幅を取得します。
- ISL=1 の場合、デューティ開始時、CNT 値は CNTS へ転送されるため、キャプチャトリガ検出時は直前の周期のパルス幅を取得します。

デューティ開始時、カウントのクリアと開始、デューティ終了時、カウントを停止するため周期は取得できません。

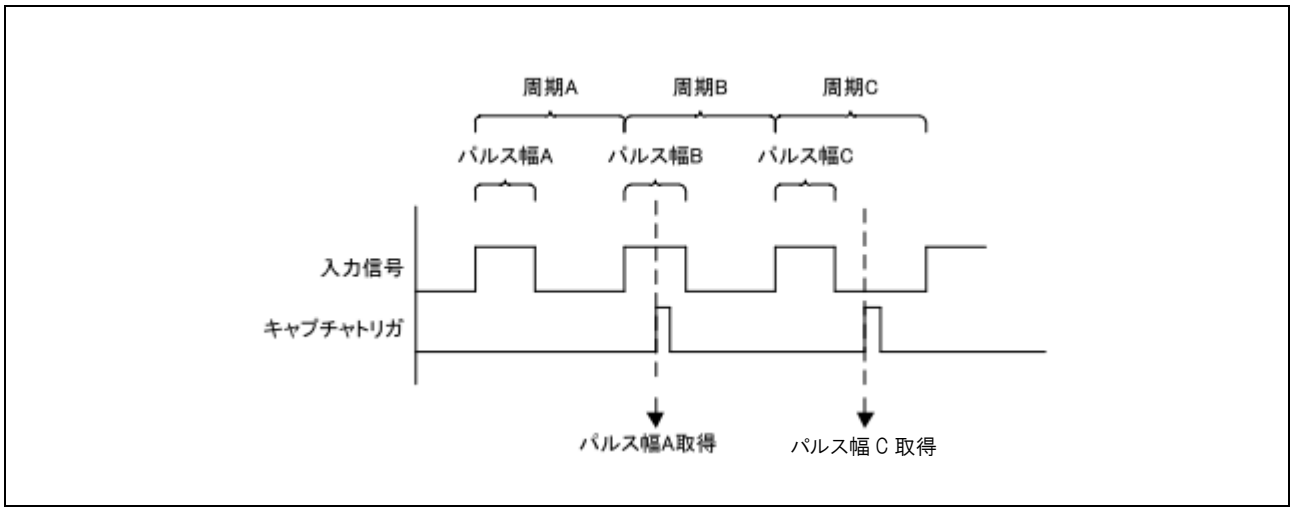


図 5-5 TPWM 入力信号測定(キャプチャトリガ有効、パルス幅信号レベル=ハイ、ISL=0)

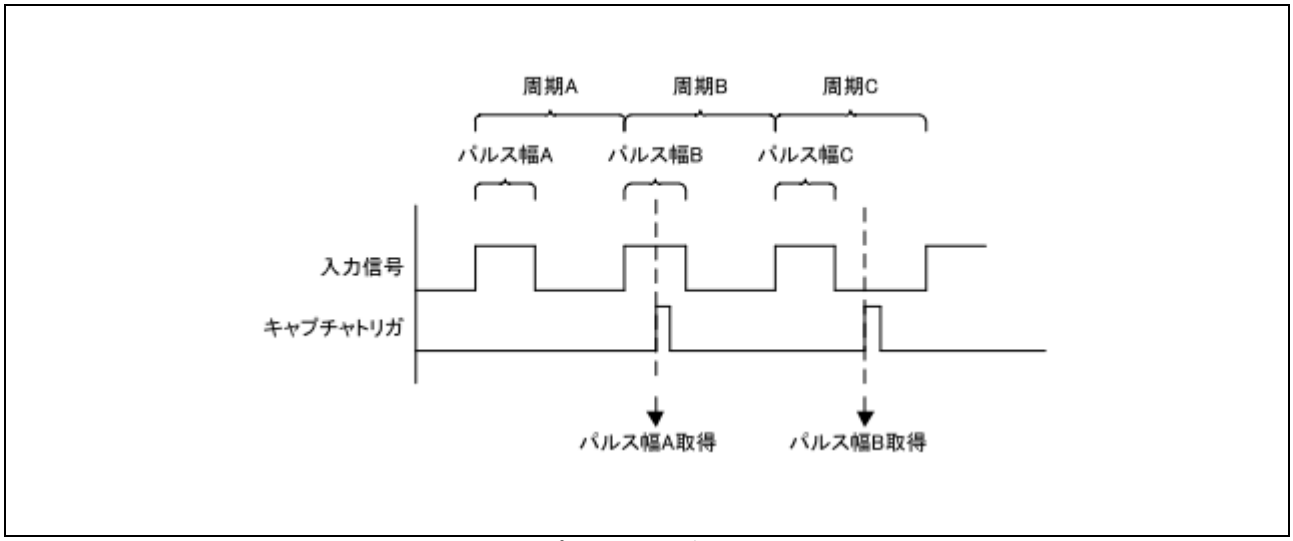


図 5-6 TPWM 入力信号測定(キャプチャトリガ有効、パルス幅信号レベル=ハイ、ISL=1)

キャプチャトリガについては 5.5 キャプチャトリガ選択を参照してください。

5.2.2 TIM Pulse Integration Mode (TPIM)

入力信号レベルの High または Low の時間を測定します。信号レベルが変化する場合、指定レベルの合計時間を測定します。

例えば、入力信号の High レベルの時間の測定であれば、波形の High レベルの合計時間を測定します。

測定を行う場合、GTM[g].TIM[i]_CH[x]_CTRL レジスタの表 5-2 の項目の設定が必要です。

表 5-2 TPIM における GTM[g].TIM[i]_CH[x]_CTRL レジスタ設定項目一覧

ビット名	説明
CLK_SEL	測定で使用する CMU_CLK のチャネル番号を設定します。 0～7：CMU_CLK チャネル番号
DSL	測定対象の信号レベルを設定します。 0：Low レベル信号を測定 1：High レベル信号を測定
TIM_MODE	チャネルモードとして TPIM を設定します。 1：TIM Pulse Integration Mode (TPIM)
EGPR0_SEL	GPR0 レジスタの格納内容を設定します。 CNTS(前回の測定時間)の内容を転送する場合は以下設定をします。 EGPR0_SEL=0 GPR0_SEL=11
GPR0_SEL	
EGPR1_SEL	GPR1 レジスタの格納内容を設定します。 CNT(転送時間)の内容を転送する場合は以下設定をします。 EGPR1_SEL=0 GPR1_SEL=11
GPR1_SEL	
TIM_EN	タイマチャネルを有効にします。 1：Enabled

(1) SMU の CNT カウンタの動作
上記の設定で指定された CMU_CLK[x]のクロックにてカウンタを作動し、入力信号を測定します。入力信号レベルが測定対象レベルではない場合はカウントを停止し、入力信号レベルが測定対象レベルとなった場合はカウントを継続します。
例として、DSL ビットが 1 に設定されている場合、入力信号が High となった際にはカウントを開始、入力信号が Low となった際にはカウントを停止、次に再度入力信号が High となった際にはカウントを再開する動作となります。

(2) ECNT カウンタの動作
ECNT は入力信号の波形エッジをカウントします。詳細は 5.1.2 の(1)を参照してください。

(3) CNT より CNTS への転送タイミング
CNT カウンタの停止時、CNT は CNTS へ転送されます。

(4) GPR0、GPR1 への転送タイミング
CNT カウンタの停止時に、EGPR0_SEL、GPR0_SEL、EGPR1_SEL、GPR1_SEL で指定した内容が GPR0、GPR1 へ転送されます。

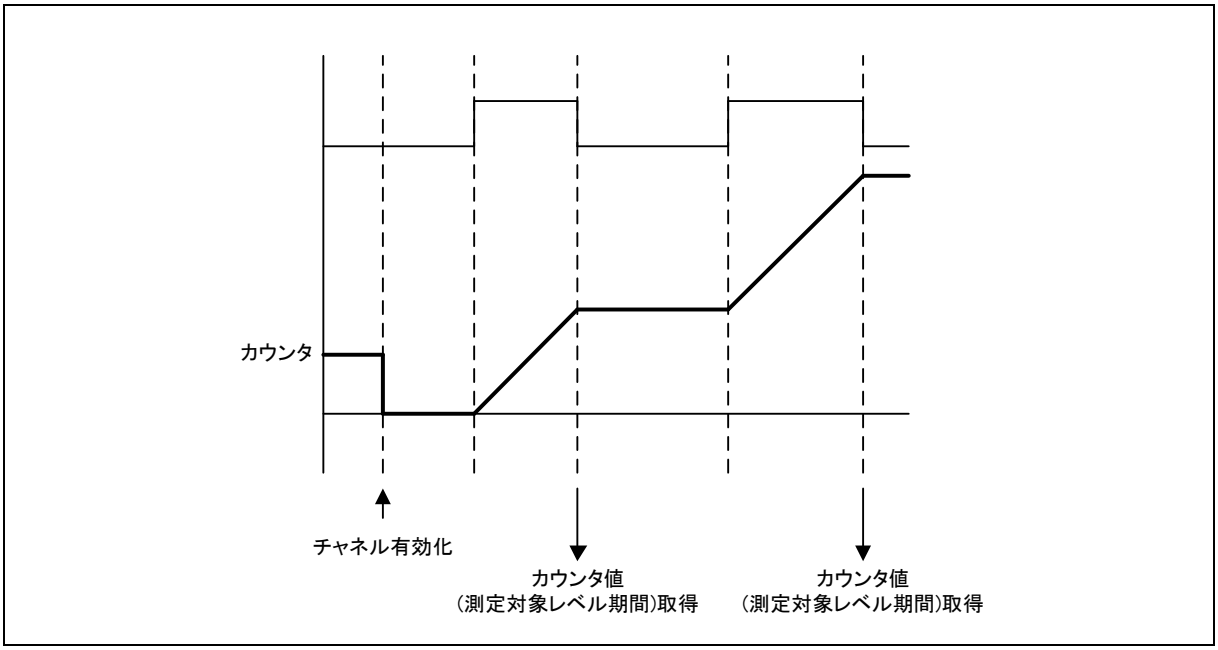


図 5-7 TPIM 入力信号測定(キャプチャトリガ無効、測定対象=ハイレベルの場合)

(5) キャプチャトリガを有効(EXT_CAP_EN=1)に設定すると、カウンタ停止時とキャプチャトリガ検出時、EGPR0_SEL、GPR0_SEL、EGPR1_SEL、GPR1_SEL で指定した内容が GPR0、GPR1 へ転送されます。またキャプチャトリガ検出時、カウンタはクリアされます。

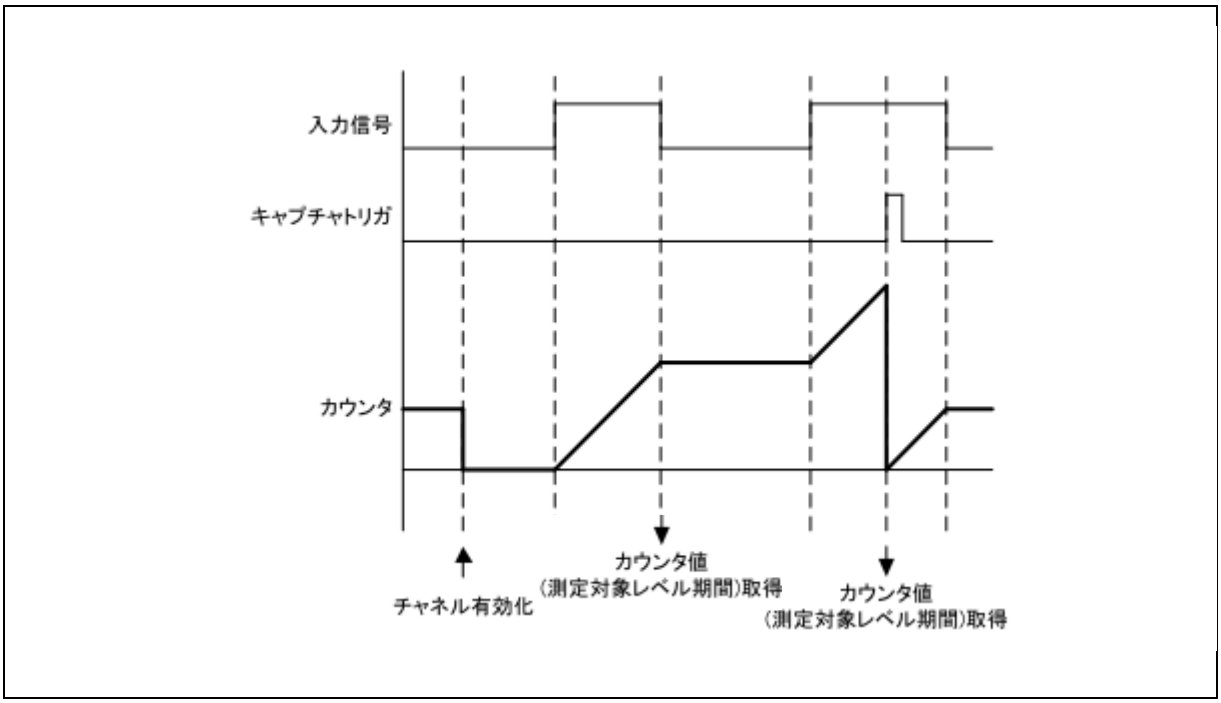


図 5-8 TPIM 入力信号測定(キャプチャトリガ有効、測定対象=ハイレベルの場合)

5.2.3 TIM Input Event Mode (TIEM)

入力信号のエッジ検出時のタイムスタンプの取得やエッジ検出をカウントします。立ち上がりエッジのみ／立ち下がりエッジのみ／両エッジの検出が可能です。

測定を行う場合、GTM[g].TIM[i]_CH[x]_CTRL レジスタの表 5-3 の項目の設定が必要です。

表 5-3 TIEM における GTM[g].TIM[i]_CH[x]_CTRL レジスタ設定項目一覧

ビット名	説明
DSL	検出対象のエッジを設定します。 ISL ビットが 0 の場合のみ有効です。 0：立ち下がりエッジ指定 1：立ち上がりエッジ指定
ISL	検出対象のエッジを設定します。 0：片エッジ指定 DSL ビットで指定したエッジが検出対象となります。 1：両エッジ指定 DSL ビットの設定は無効となります。
TIM_MODE	チャネルモードとして TIEM を設定します。 2：TIM Input Event Mode (TIEM)
EGPR0_SEL	GPR0 レジスタの格納内容を設定します。 TBU チャネル(TBU チャネル 0 のタイムスタンプ値)の内容を転送する場合は以下設定をします。 EGPR0_SEL=0 GPR0_SEL=00
GPR0_SEL	
EGPR1_SEL	GPR1 レジスタの格納内容を設定します。 CNT(エッジカウンタ)の内容を転送する場合は以下設定をします。 EGPR1_SEL=0 GPR1_SEL=11
GPR1_SEL	
TIM_EN	チャネルを有効にします。 1：Enabled

(1) SMU の CNT カウンタの動作
上記にて設定された DSL ビット、ISL ビット等により、入力信号の波形エッジでカウントを行います。また、カウント時に TIM[i]_NEWVAL[x]_IRQ 信号が生成されます。
本チャネルモードでの CNT カウンタは、ECNT とは異なり、開始時の信号レベルに関わらず 0 から開始します。

(2) GPR0、GPR1 への転送タイミング
CNT カウンタのカウント時に、EGPR0_SEL、GPR0_SEL、EGPR1_SEL、GPR1_SEL で指定した内容が GPR0、GPR1 へ転送されます。

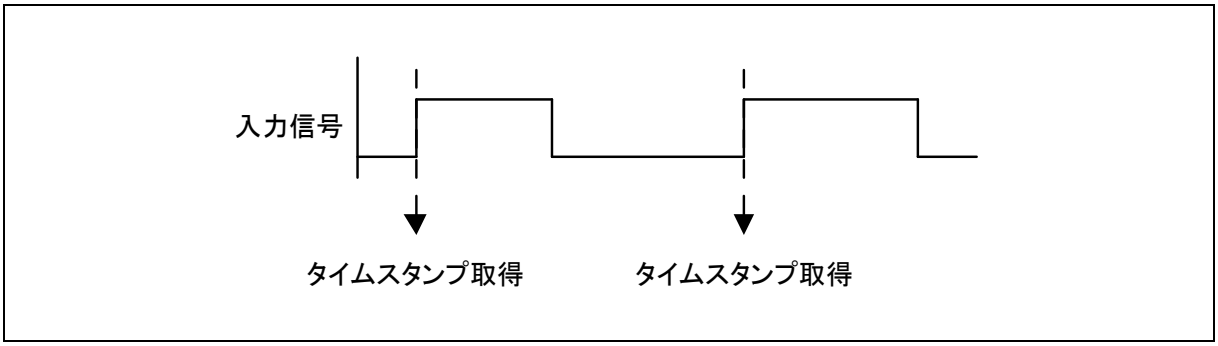


図 5-9 TIEM 入力信号測定(キャプチャトリガ無効、検出対象エッジ=立ち上がりエッジのみの場合)

(3) キャプチャトリガを有効(EXT_CAP_EN=1)に設定すると、キャプチャトリガを検出した時点において、下記条件を満たす場合にタイムスタンプ、エッジカウンタを取得します。

・検出対象エッジが両エッジの場合
・検出対象エッジが立ち上がりエッジのみ、かつ入力信号がハイレベル
・検出対象エッジが立ち下がりエッジのみ、かつ入力信号がロウレベル

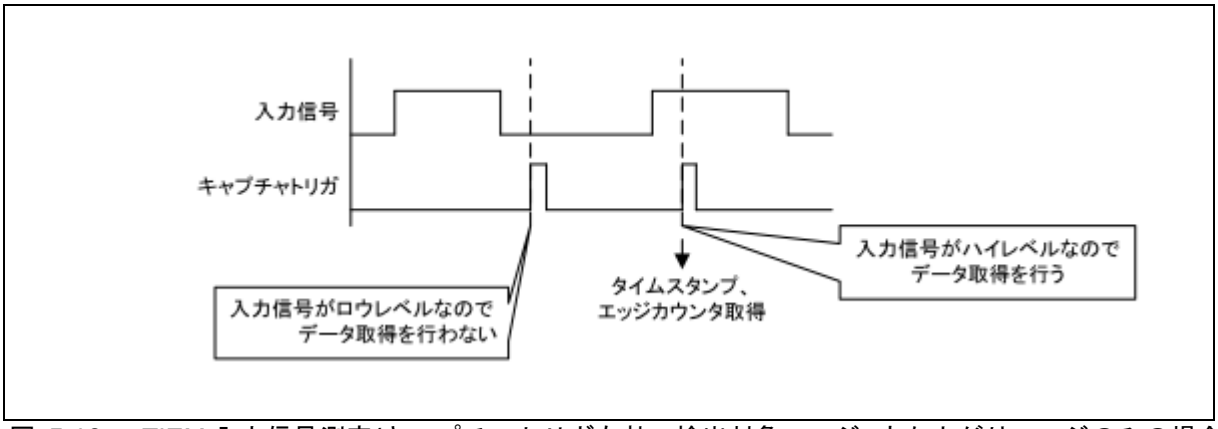


図 5-10 TIEM 入力信号測定(キャプチャトリガ有効、検出対象エッジ=立ち上がりエッジのみの場合)

5.2.4 TIM Input Prescaler Mode (TIPM)

入力信号を分周してエッジ検出時のタイムスタンプの取得やエッジ検出をカウントします。立ち上がりエッジのみ／立ち下がりエッジのみ／両エッジの検出が可能です。本モードではマイコンや MCS で処理が難しい高速信号の測定が可能です。

測定を行う場合、GTM[g].TIM[i]_CH[x]_CTRL レジスタの表 5-4 の項目の設定が必要です。

表 5-4 TIPM における GTM[g].TIM[i]_CH[x]_CTRL レジスタ設定項目一覧

ビット名	説明
DSL	検出対象のエッジを設定します。 ISL ビットが 0 の場合のみ有効です。 0：立ち下がりエッジ指定 1：立ち上がりエッジ指定
ISL	検出対象のエッジを設定します。 0：片エッジ指定 DSL ビットで指定したエッジが検出対象となります。 1：両エッジ指定 DSL ビットの設定は無効となります。
TIM_MODE	チャネルモードとして TIPM を設定します。 3：TIM Input Prescaler Mode (TIPM)
EGPR0_SEL	GPR0 レジスタの格納内容を設定します。 TBU チャネル 0(TBU チャネル 0 のタイムスタンプ値)の内容を転送する場合は以下設定をします。 EGPR0_SEL=0 GPR0_SEL=00
GPR0_SEL	
EGPR1_SEL	GPR1 レジスタの格納内容を設定します。 ECNT(エッジカウンタ)の内容を転送する場合は以下設定をします。 EGPR1_SEL=1 GPR1_SEL=00
GPR1_SEL	
TIM_EN	チャネルを有効にします。 1：Enabled

前述の表とは別に GTM[g].TIM[i]_CH[x]_CNTS レジスタへ入力信号の分周値を設定します。分周値は設定値+1 となります。

例えば、2 分周とする場合は、GTM[g].TIM[i]_CH[x]_CNTS レジスタへ 1 を設定します。

(1) SMU の CNT カウンタの動作
DSL ビット、ISL ビット等により指定された入力信号の波形エッジでカウントを行います。また、CNT カウンタ値が分周値(CNTS 設定値+1)と一致した場合、TIM[i]_NEWVAL[x]_IRQ 信号が生成されます。

(2) ECNT カウンタの動作
ECNT は入力信号の波形エッジをカウントします。詳細は 5.1.2 の(1)を参照してください。

(3) GPR0、GPR1 への転送タイミング
CNT カウンタ値と CNTS 設定値が一致時に、EGPR0_SEL、GPR0_SEL、EGPR1_SEL、GPR1_SEL で指定した内容が GPR0、GPR1 へ転送されます。

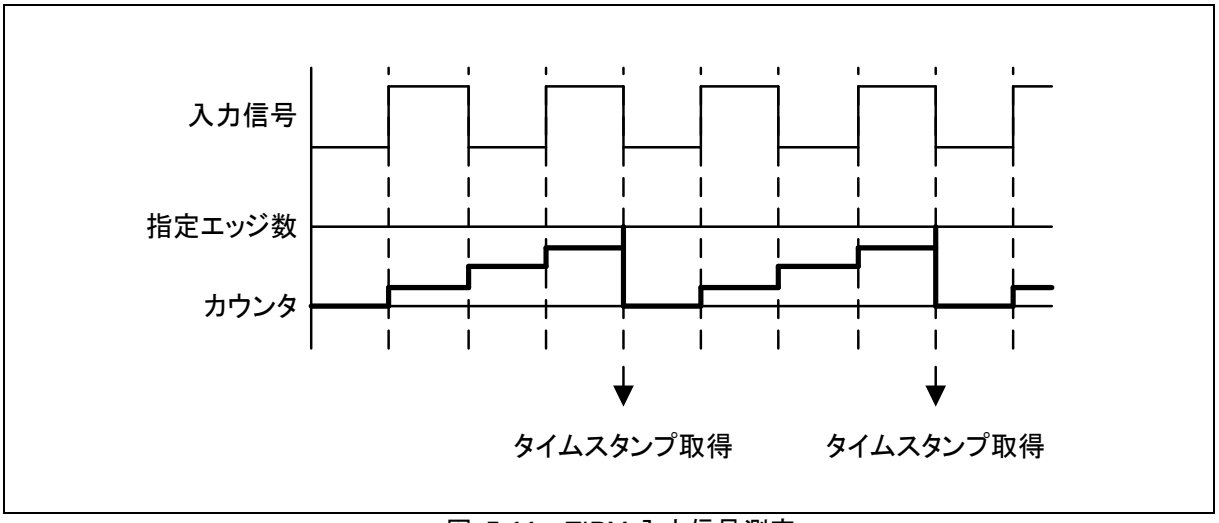


図 5-11 TIPM 入力信号測定
(キャプチャトリガ無効、検出対象エッジ=両エッジ、指定エッジ数=4 の場合)

(4) キャプチャトリガを有効(EXT_CAP_EN=1)に設定すると、入力信号のエッジではなく、キャプチャトリガを検出した時点において下記条件を満たす場合にカウントを行い、カウントが指定した数に達した場合にタイムスタンプを取得します。

・検出対象エッジが両エッジ
・検出対象エッジが立ち上がりエッジのみ、かつ入力信号がハイレベル
・検出対象エッジが立ち下がりエッジのみ、かつ入力信号がロウレベル

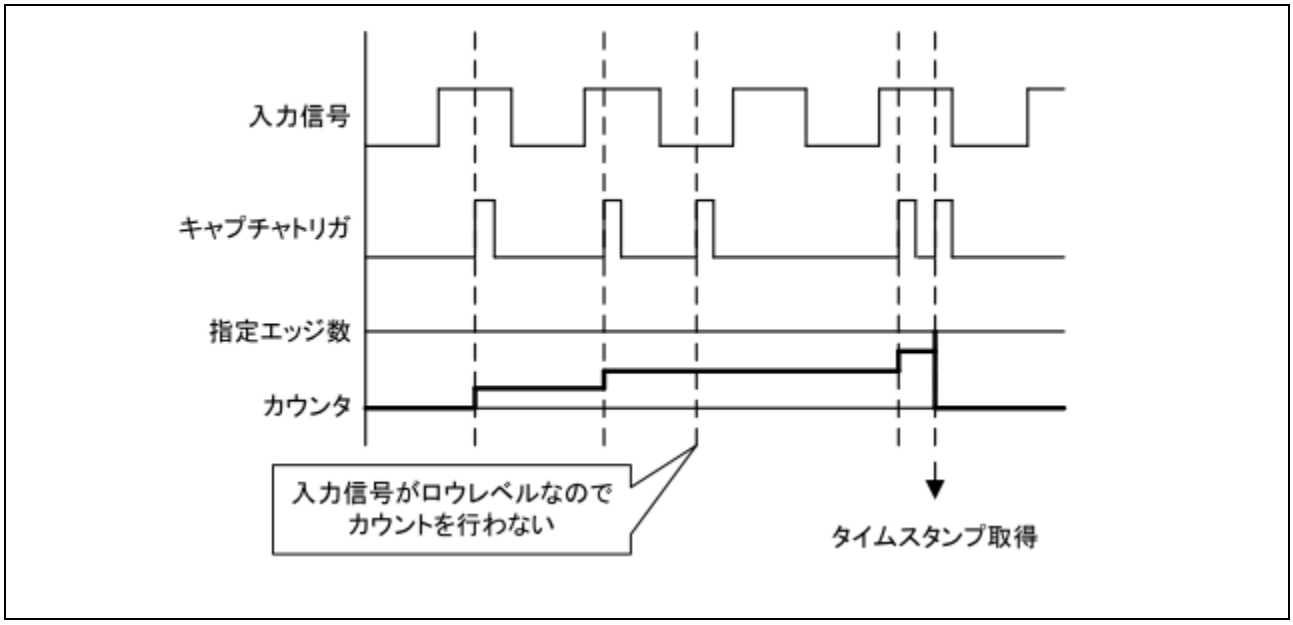


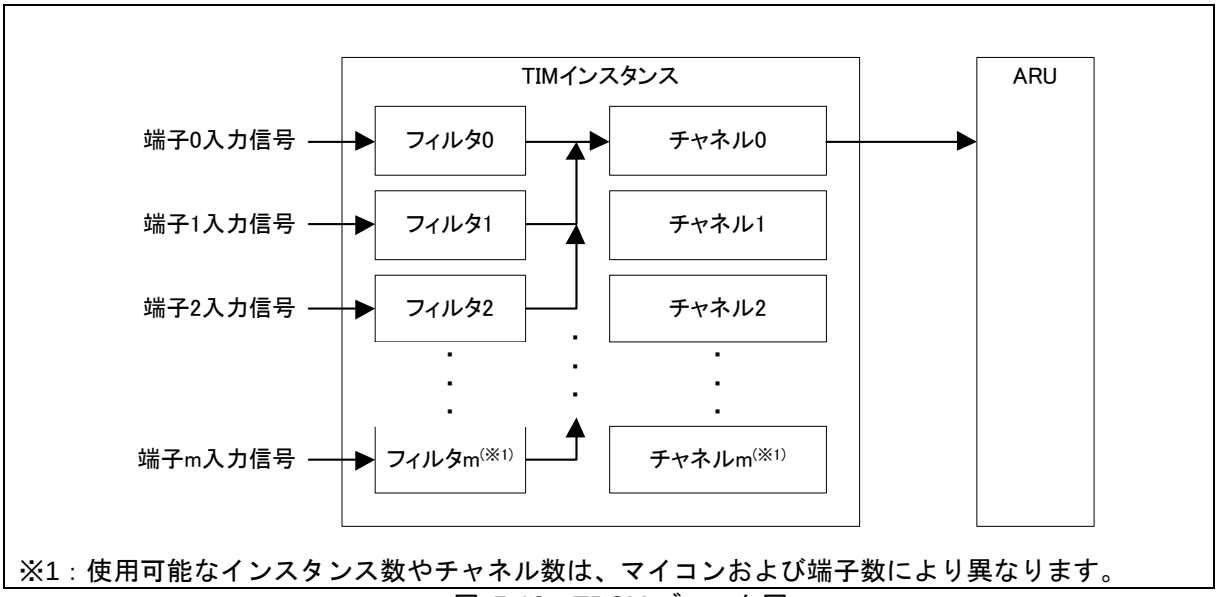
図 5-12 TIPM 入力信号測定
(キャプチャトリガ有効、検出対象エッジ=立ち上がりエッジのみ、指定エッジ数=4 の場合)

5.2.5 TIM Bit Compression Mode (TBCM)

TIM が持つ全チャネルを TIM のチャネル 0 のみでパラレルに測定するモードです。外部デバイスからの通信受信等を行うことも可能です。

入力フィルタ(FLT[x])は各チャネル有効ですが、個別に設定する必要があります。

測定を行う場合、GTM[g].TIM[i]_CH[x]_CTRL レジスタの表 5-5 の項目の設定が必要です。



※1：使用可能なインスタンス数やチャネル数は、マイコンおよび端子数により異なります。

図 5-13 TBCM ブロック図

表 5-5 TBCM における GTM[g].TIM[i]_CH[x]_CTRL レジスタ設定項目一覧

ビット名	説明
TIM_MODE	チャネルモードとして TBCM を設定します。 4：Bit Compression Mode (TBCM)
EGPR0_SEL	GPR0 レジスタの格納内容を設定します。 TBU チャネル 0(TBU チャネル 0 のタイムスタンプ値)の内容を転送する場合は以下設定をします。 EGPR0_SEL=0 GPR0_SEL=00
GPR0_SEL	
TIM_EN	タイマチャネルを有効にします。 1：Enabled

また、GTM[g].TIM[i]_CH[x]_CNTS レジスタへ入力信号の測定条件イベントをビットマップで設定します。

表 5-6 TPIM GTM[g].TIM[i]_CH[x]_CNTS レジスタ設定項目

bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
チャネル	未使用															
イベント																
bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
チャネル	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
イベント	立ち下がりエッジ								立ち上がりエッジ							

測定条件イベントは OR 設定です。ビット 0 とビット 8 の両ビットが設定された場合、チャネル 0 の入力信号の両エッジで信号を測定します。イベント成立時 TIM[i]_NEWVAL[x]_IRQ 信号が生成されます。

(1) ECNT カウンタの動作
ECNT は測定条件イベントの成立回数を表します。

(2) GPR0 への転送タイミング
測定条件イベントが成立時に、EGPR0_SEL、GPR0_SEL で指定した内容が GPR0 へ転送されます。

(3) GPR1 の内容
測定条件イベントが成立した場合に、全チャネル分の信号状態が、チャネル番号に応じたビット番号(チャネル 0 ではビット 0、チャネル 1 ではビット 1、…チャネル x ではビット x)へ転送されます。

(4) キャプチャトリガを有効(EXT_CAP_EN=1)に設定すると、キャプチャトリガを検出した時点において、下記条件を満たす場合にタイムスタンプと全フィルタ出力信号の信号レベルを取得することができます。

・ISL=1
・ISL=0 かつ DSL=1 かつ入力信号がハイレベル
・ISL=0 かつ DSL=0 かつ入力信号がロウレベル

5.2.6 TIM Gated Periodic Sampling Mode (TGPS)

指定された入力信号レベルの合計期間が指定クロック数となった時に TIM[i]_NEWVAL[x]_IRQ 信号を生成します。

測定を行う場合、GTM[g].TIM[i]_CH[x]_CTRL レジスタの表 5-7 の項目の設定が必要です。

表 5-7 TGPS における GTM[g].TIM[i]_CH[x]_CTRL レジスタ設定項目一覧

ビット名	説明
CLK_SEL	測定で使用する CMU_CLK のチャネル番号を設定します。 0～7：CMU_CLK チャネル番号
DSL	カウント対象信号レベルを設定します。 ISL ビットが 0 の場合のみ有効です。 0：Low レベル指定 1：High レベル指定
ISL	カウント対象信号レベルを設定します。 0：片レベル指定 DSL ビットで指定したレベルが対象となります。 1：両レベル指定 DSL ビットの設定は無効となります。
TIM_MODE	チャネルモードとして TGPS を設定します。 5：Gated Periodic Sampling Mode (TGPS)
EGPR0_SEL	GPR0 レジスタの格納内容を設定します。 TBU チャネル 0(TBU チャネル 0 のタイムスタンプ値)の内容を転送する場合は以下設定をします。 EGPR0_SEL=0 GPR0_SEL=00
GPR0_SEL	
EGPR1_SEL	GPR1 レジスタの格納内容を設定します。 ECNT(エッジカウンタ)の内容を転送する場合は以下設定をします。 EGPR1_SEL=1 GPR1_SEL=00
GPR1_SEL	
TIM_EN	タイマチャネルを有効にします。 1: Enabled

前述の表とは別に、GTM[g].TIM[i]_CH[x]_CNTS レジスタへ、TIM[i]_NEWVAL[x]_IRQ を生成するまでのクロック数-1 を設定します。例えば、CMU_CLK のクロック数を 4 に設定する場合は、本レジスタへ 3 を設定します。

本動作モードでは、GTM[g].TIM[i]_CH[x]_GPR1 レジスタは GTM[g].TIM[i]_CH[x]_CNTS レジスタのシャドゥレジスタとして機能します。この機能により、次に GTM[g].TIM[i]_CH[x]_CNTS レジスタに設定する値を GTM[g].TIM[i]_CH[x]_GPR1 レジスタへ設定しておき、SMU の CNT のカウンタ値と指定クロック数が一致した場合に、GTM[g].TIM[i]_CH[x]_GPR1 レジスタの値を GTM[g].TIM[i]_CH[x]_CNTS レジスタへ自動的にロードできます。

(1) SMU の CNT カウンタの動作
指定された CMU_CLK[x]のクロックにてカウントを行います。
ただし、1 クロックごとに入力信号レベルが ISL や DSL にて設定したカウント対象の信号レベルに等しいかチェックし、等しい場合のみカウントアップを行います。
カウンタ値が指定クロック数(GTM[g].TIM[i]_CH[x]_CNTS レジスタにて設定されたクロック数+1)と一致した場合、TIM[i]_NEWVAL[x]_IRQ 信号が生成され、その後、CNT は 0 にクリアされます。

(2) ECNT カウンタの動作
ECNT は入力信号の波形エッジをカウントします。詳細は 5.1.2 の(1)を参照してください。

(3) GPR0、GPR1 への転送タイミング
CNT カウンタ値と CNTS 設定値が一致時に、EGPR0_SEL、GPR0_SEL、EGPR1_SEL、GPR1_SEL で指定した内容が GPR0、GPR1 へ転送されます。

次の図は入力信号の両エッジを測定周期毎にカウントするタイミングチャートです。図中のカウンタは ECNT カウントの動作を示します。

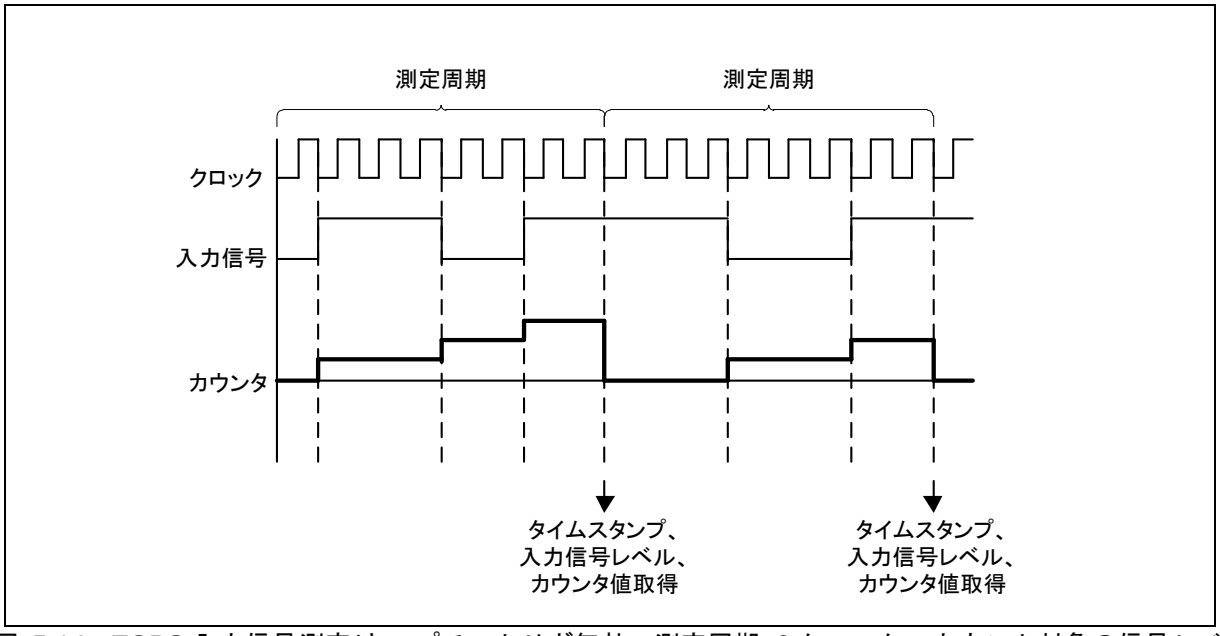


図 5-14 TGPS 入力信号測定(キャプチャトリガ無効、測定周期=8 クロック、カウント対象の信号レベルは両レベル(ISL=1)、ECNT レジスタを測定周期でリセット(ECNT_RESET=1)の場合)

(4) キャプチャトリガを設定すると、キャプチャトリガを検出した時点のタイムスタンプと入力信号レベル、周期開始からのエッジカウンタ値を取得することができます。このとき、周期のカウンタはリセットされます。図中のカウンタは ECNT カウンタの動作を示します。

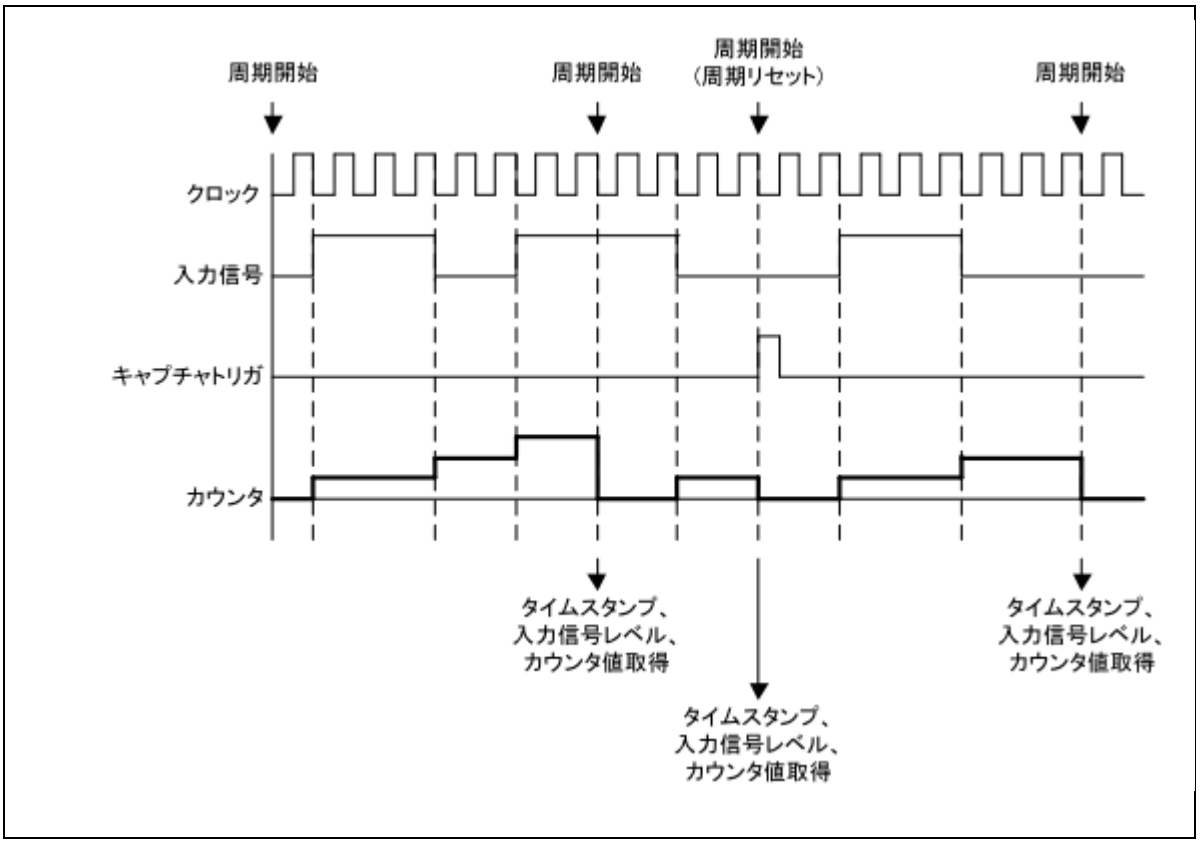


図 5-15 TGPS 入力信号測定(キャプチャトリガ有効、測定周期=8 クロック、カウント対象の信号レベルは両レベル(ISL=1)、ECNT レジスタを測定周期でリセット(ECNT_RESET=1)の場合)

5.2.7 TIM Serial Shift Mode (TSSM)

CNT レジスタを使用して、シリアル信号を受信します。シフトクロック(shift_clock)によってシリアル受信のレートが決まります。CNTS[7:0]で指定されたビット数受信後、TIM[i]_NEWVAL[x]_IRQ が通知されます。

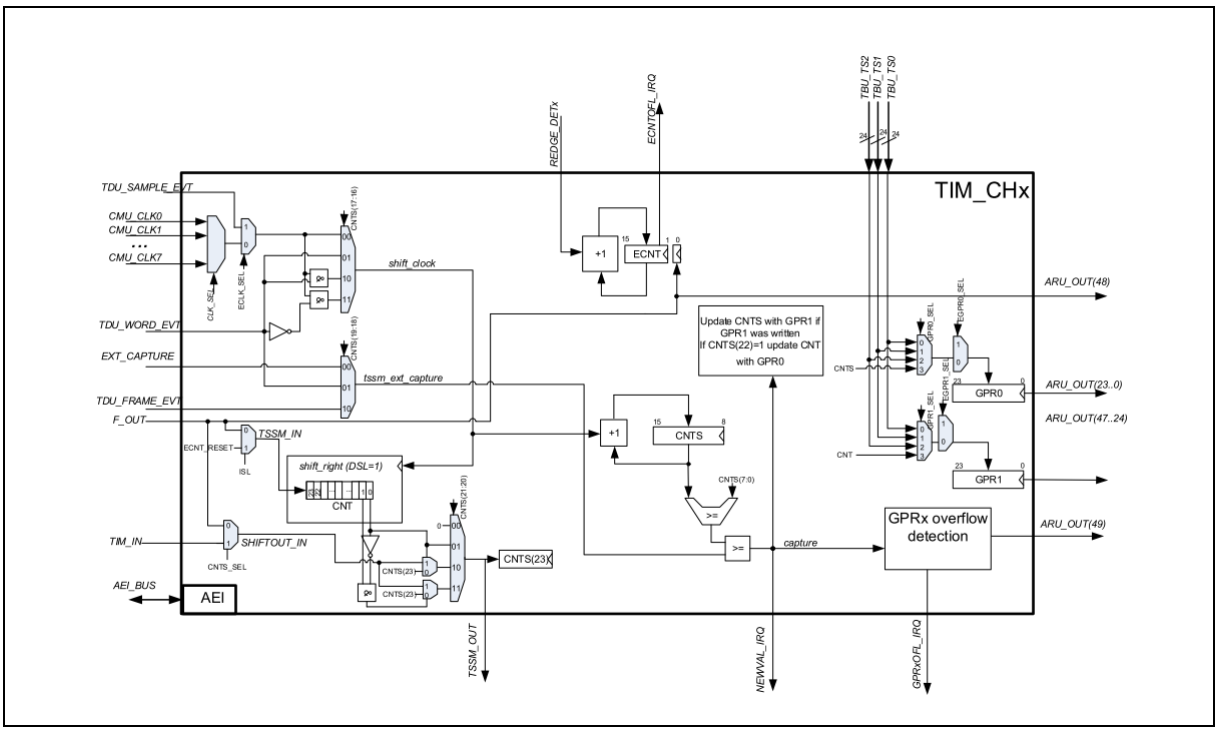


図 5-16 TIM Serial Shift Mode

測定する場合、以下の設定が必要です。

表 5-8 TSSM における GTM[g].TIM[i]_CH[x]_CTRL 設定項目一覧

ビット名	説明
CLK_SEL	シフトクロックを設定します(表 5-10 TSSM シフトクロック設定項目参照)。
DSL	シフトレジスタのシフト方向を設定します。 0：左シフト 1：右シフト
ISL	入力信号(TSSM_IN)を設定します。 0：F_OUTx 1：ECNT_RESET
TIM_MODE	チャネルモードとして TSSM を設定します。 6：Serial Shift Mode (TSSM)
EGPR0_SEL	GPR0 レジスタの格納内容を設定します。 TBU チャネル 0(TBU チャネル 0 のタイムスタンプ値)の内容を転送する場合は以下設定をします。
GPR0_SEL	EGPR0_SEL=0 GPR0_SEL=00

EGPR1_SEL	GPR1 レジスタの格納内容を設定します。 CNT レジスタの内容を転送する場合は以下設定をします。 EGPR1_SEL=0 GPR1_SEL=11
GPR1_SEL	
TIM_EN	タイマチャネルを有効にします。 1: Enabled

表 5-9 TSSM GTM[g].TIM[i]_CH[x]_CNTS レジスタ設定項目

ビット名	説明
CNTS [23]	TSSM_OUT が表示されます。
CNTS [22]	0: GPR0 は CNT のシャドウレジスタとして動作しない- 1: GPR0 は CNT のシャドウレジスタとして動作する
CNTS [21:20]	TSSM_OUT 生成方法を設定します。
CNTS [19:18]	キャプチャトリガを設定します。 00_B : EXT_CAP_SRC によるソース選択が使用されます （表 5-15 選択可能なキャプチャトリガー覧）。 01_B : tdu_word_evt がソースとして使用されます。 10_B : tdu_frame_evt がソースとして使用されます。 11_B : 予約
CNTS [17:16]	シフトクロックを設定します(表 5-10 TSSM シフトクロック設定項目参照)。
CNTS [15:8]	シフトカウンタです。
CNTS [7:0]	シフト回数(受信ビット数)を設定します。

表 5-10 TSSM シフトクロック設定項目

CNTS[17:16]	ECLK_SEL	CLK_SEL	シフトクロック
00	0	x	CMU_CLK[x] (x = 0-7)
	1	-	tdu_sample_evt
01	-	-	tdu_word_evt
10	0	x	CMU_CLK[x]を使用し、tdu_word_evt でゲートされます。tdu_word_evt=0 の場合、シフトクロックは 0 になります
	1	-	tdu_sample_evt を使用し、tdu_word_evt でゲートされます。tdu_word_evt=0 の場合、シフトクロックは 0 になります
11	0	x	CMU_CLK[x]を使用し、tdu_word_evt でゲートされます。tdu_word_evt=0 の場合、シフトクロックは 0 になります
	1	-	tdu_sample_evt を使用し、tdu_word_evt でゲートされます。tdu_word_evt=1 の場合、シフトクロックは 0 になります

TSSM モードは GTM[g].TIM[i]_CH[x]_CTRL レジスタの ISL、DSL ビットにより動作が異なります。

表 5-11 TSSM 動作一覧 (キャプチャトリガ無効)

シフトクロック	ISL	DSL	動作
0	-	-	なし
1	0	0	(1) CNT を左シフト ・CNT [23: 1] = CNT [22: 0] ・CNT [0] = TSSM_INx (2) CNTS [15: 8] ≥ CNTS [7: 0]の場合、 ・キャプチャ ・NEWVAL_IRQ 発行 ・CNTS [15: 8] = 0 (3) CNTS [15: 8] < CNTS [7: 0]の場合、CNTS [15: 8] + +
1	0	1	(1) CNT を右シフト CNT [22: 0] = CNT [23: 1] CNT [23] = TSSM_INx (2) CNTS [15: 8]≧CNTS [7: 0]の場合、 ・キャプチャ ・NEWVAL_IRQ 発行 ・CNTS [15: 8] = 0 (3) CNTS [15: 8] < CNTS [7: 0]の場合、CNTS [15: 8] + +
1	1	0	(1) CNT を左シフト ・CNT [23: 1] = CNT [22: 0]; ・CNT [0] = TSSM_INx (2) CNTS [15: 8] ≥ CNTS [7: 0]の場合、 ・キャプチャ ・NEWVAL_IRQ 発行 ・CNTS [15: 8] = 0 ・CNT [23: 0] = ECNT_RESET (3) CNTS [15: 8] < CNTS [7: 0]の場合、CNTS [15: 8] + +
1	1	1	(1) CNT を右シフト ・CNT [22: 0] = CNT [23: 1]; ・CNT [23] = TSSM_INx (2) CNTS[15: 8] ≥ CNTS [7: 0] の場合、 ・キャプチャ ・NEWVAL_IRQ 発行; ・CNTS [15: 8] = 0 ・CNT [23: 0] = ECNT_RESET (3) CNTS [15: 8] < CNTS [7: 0]の場合、CNTS [15: 8] + +

本動作モードでは、GTM[g].TIM[i]_CH[x]_GPR1 レジスタは GTM[g].TIM[i]_CH[x]_CNTS レジスタのシャドゥレジスタとして機能します。この機能により、次に GTM[g].TIM[i]_CH[x]_CNTS レジスタに設定する値を GTM[g].TIM[i]_CH[x]_GPR1 レジスタへ設定しておき、CNTS[15:8]のカウンタ値≧CNTS[7:0]が成立した場合に、GTM[g].TIM[i]_CH[x]_GPR1 レジスタの値を GTM[g].TIM[i]_CH[x]_CNTS レジスタへ自動的にロードできます。

つまり、前回キャプチャされた値が GTM[g].TIM[i]_CH[x]_GPR1 レジスタから CPU によって読み出され、その後、次のサンプリング期間(実際のサンプリング期間の後のもの)のサンプリングのために新しいビット数を書き込むことができます。

(1) ECNT カウンタの動作
ECNT は入力信号の波形エッジをカウントします。詳細は 5.1.2 の(1)を参照してください。

(2) GPR0、GPR1 への転送タイミング
CNTS[15:8]≧CNTS[7:0]が成立時に、EGPR0_SEL、GPR0_SEL、EGPR1_SEL、GPR1_SEL で指定した内容が GPR0、GPR1 へ転送されます。

キャプチャトリガ有効 (EXT_CAP_EN = 1)を設定すると、キャプチャトリガを検出した時点においてGPRx をキャプチャし、 ISL に応じてカウンタ CNT をリセットし、NEWVAL_IRQ を発行します。TSSM モードの TIM キャプチャトリガとして使用されるソースは、CNTS[19:18]で設定します。

表 5-12 TSSM 動作一覧 (キャプチャトリガ有効)

シフトクロック	tssm_ext_capture	ISL	DSL	動作
0	1	1	-	・キャプチャ ・NEWVAL_IRQ 発行 ・CNTS [15: 8] = 0 ・CNT[23:0]=ECNT_RESET
0	1	0	-	・キャプチャ ・NEWVAL_IRQ 発行 ・CNTS [15: 8] = 0
1	1	1	0	CNT を左シフト ・CNT [23: 1] = CNT [22: 0] ・CNT [0] = TSSM_INx ・キャプチャ ・NEWVAL_IRQ 発行 ・CNTS [15: 8] = 0 ・CNT[23:0]=ECNT_RESET
1	1	1	1	CNT を右シフト ・CNT [22: 0] = CNT [23: 1] ・CNT [23] = TSSM_INx ・キャプチャ ・NEWVAL_IRQ 発行 ・CNTS [15: 8] = 0 ・CNT[23:0]=ECNT_RESET
1	1	0	0	CNT を左シフト ・CNT [23: 1] = CNT [22: 0]; ・CNT [0] = TSSM_INx ・キャプチャ ・NEWVAL_IRQ 発行 ・CNTS [15: 8] = 0
1	1	0	1	CNT を右シフト ・CNT [22: 0] = CNT [23: 1]; ・CNT [23] = TSSM_INx ・キャプチャ ・NEWVAL_IRQ 発行; ・CNTS [15: 8] = 0

5.3 入力フィルタ

FLT は、TIM への入力信号に対しフィルタリングを行うサブユニットです。FLT にはフィルタモードがあり、入力信号の立ち上がり時、立ち下がり時のそれぞれに対して、異なるフィルタモードを設定することが可能です。

フィルタモードは、Immediate edge propagation mode、Individual de-glitch mode の 2 種類があり、Individual de-glitch mode はさらに Up/Down Counter、Hold Counter、Reset Counter の 3 種類のフィルタカウンタモードが存在します。

入力フィルタを使用する場合、GTM[g].TIM[i]_CH[x]_CTRL レジスタの表 5-13 の項目の設定が必要です。

表 5-13 入力フィルタにおける GTM[g].TIM[i]_CH[x]_CTRL レジスタ 設定項目一覧

<table>
<tr><th>ビット名</th><th>説明</th></tr>
<tr><td>FLT_EN</td><td>フィルタ機能の有効／無効を設定します。
0：フィルタ無効
フィルタ設定はリセットされます。
1：フィルタ有効</td></tr>
<tr><td>FLT_CNT_FRQ</td><td>フィルタユニットへ供給するクロックを選択します。
0：CMU_CLK0
1：CMU_CLK1
2：CMU_CLK6
3：CMU_CLK7</td></tr>
<tr><td>FLT_MODE_FE
FLT_CTR_FE</td><td>立ち下がり時のフィルタモードを設定します。
<table>
<tr><th>FLT_MODE_FE</th><th>EFLT_CTR_FE</th><th>FLT_CTR_FE</th><th>フィルタカウンタモード</th></tr>
<tr><td rowspan="2">0</td><td>0</td><td rowspan="2">-</td><td>Immediate edge propagation mode</td></tr>
<tr><td>1</td><td>予約</td></tr>
<tr><td rowspan="4">1</td><td rowspan="2">0</td><td>0</td><td>Individual de-glitch mode (up-down counter)</td></tr>
<tr><td>1</td><td>Individual de-glitch mode (hold counter)</td></tr>
<tr><td rowspan="2">1</td><td>0</td><td>Individual de-glitch mode (reset counter)</td></tr>
<tr><td>1</td><td>予約</td></tr>
</table></td></tr>
<tr><td>FLT_MODE_RE
FLT_CTR_RE</td><td>立ち上がり時のフィルタモードを設定します。
<table>
<tr><th>FLT_MODE_RE</th><th>EFLT_CTR_RE</th><th>FLT_CTR_RE</th><th>フィルタカウンタモード</th></tr>
<tr><td rowspan="2">0</td><td>0</td><td rowspan="2">-</td><td>Immediate edge propagation mode</td></tr>
<tr><td>1</td><td>予約</td></tr>
<tr><td rowspan="4">1</td><td rowspan="2">0</td><td>0</td><td>Individual de-glitch mode (up-down counter)</td></tr>
<tr><td>1</td><td>Individual de-glitch mode (hold counter)</td></tr>
<tr><td rowspan="2">1</td><td>0</td><td>Individual de-glitch mode (reset counter)</td></tr>
<tr><td>1</td><td>予約</td></tr>
</table></td></tr>
</table>

また、入力フィルタの時間(クロック数)は以下のレジスタにて設定します。

(1) GTM[g].TIM[i]_CH[x]_FLT_RE レジスタ
立ち上がり時のフィルタカウンタ値を設定します。フィルタカウンタ値は設定値+1 となります。
例えば、フィルタカウンタ値を 4 に設定する場合は、本レジスタへ 3 を設定します。

(2) GTM[g].TIM[i]_CH[x]_FLT_FE レジスタ
立ち下がり時のフィルタカウンタ値を設定します。フィルタカウンタ値は設定値+1 となります。
例えば、フィルタカウンタ値を 4 に設定する場合は、本レジスタへ 3 を設定します。

各レジスタの設定の詳細は、ユーザーズマニュアルを参照してください。

5.3.1 Immediate edge propagation

入力信号において、エッジを検出後、即座に一定時間信号レベルを保持するモードです。
立ち上がりエッジの場合、立ち上がり時のフィルタカウンタ値の間 High レベルを保持し、立ち下がりエッジの場合、立ち下がり時のフィルタカウンタ値の間 Low レベルを保持します。

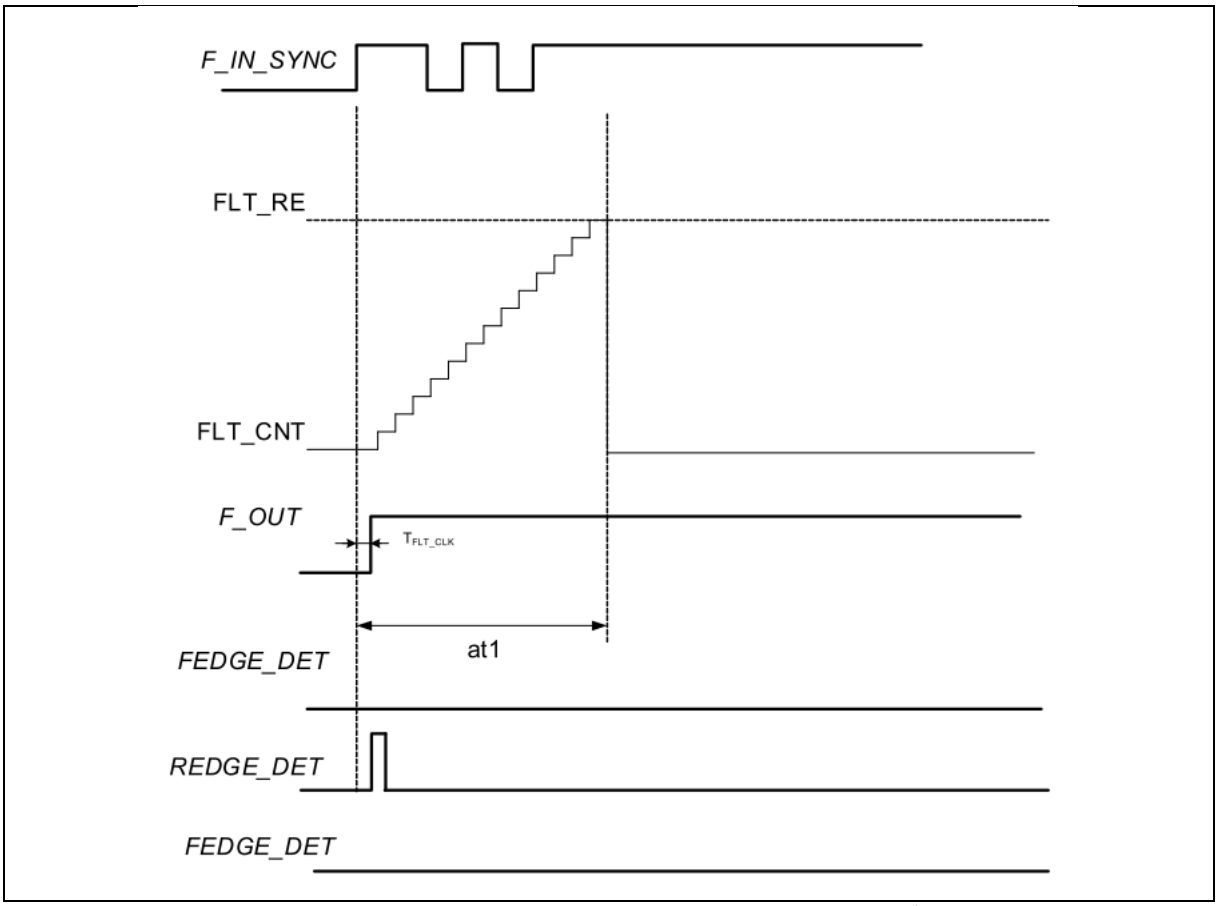


図 5-17 Immediate edge propagation タイミング

このフィルタモードでは、フィルタカウント中に入力信号の変化(グリッチ)を検出した場合、TIM[i]_GLITCHDET[x]_IRQ 信号を生成します。TIM[i]_GLITCHDET[x]_IRQ については 5.7.1 を参照してください。

5.3.2 Individual de-glitch time (up/down counter)

エッジ検出後、指定された期間、入力信号レベルが同じ場合に出力信号レベルを変化させます。入力信号がフィルタカウント中にレベル反転した場合、ダウンカウントを実施してフィルタリング時間を延長します。

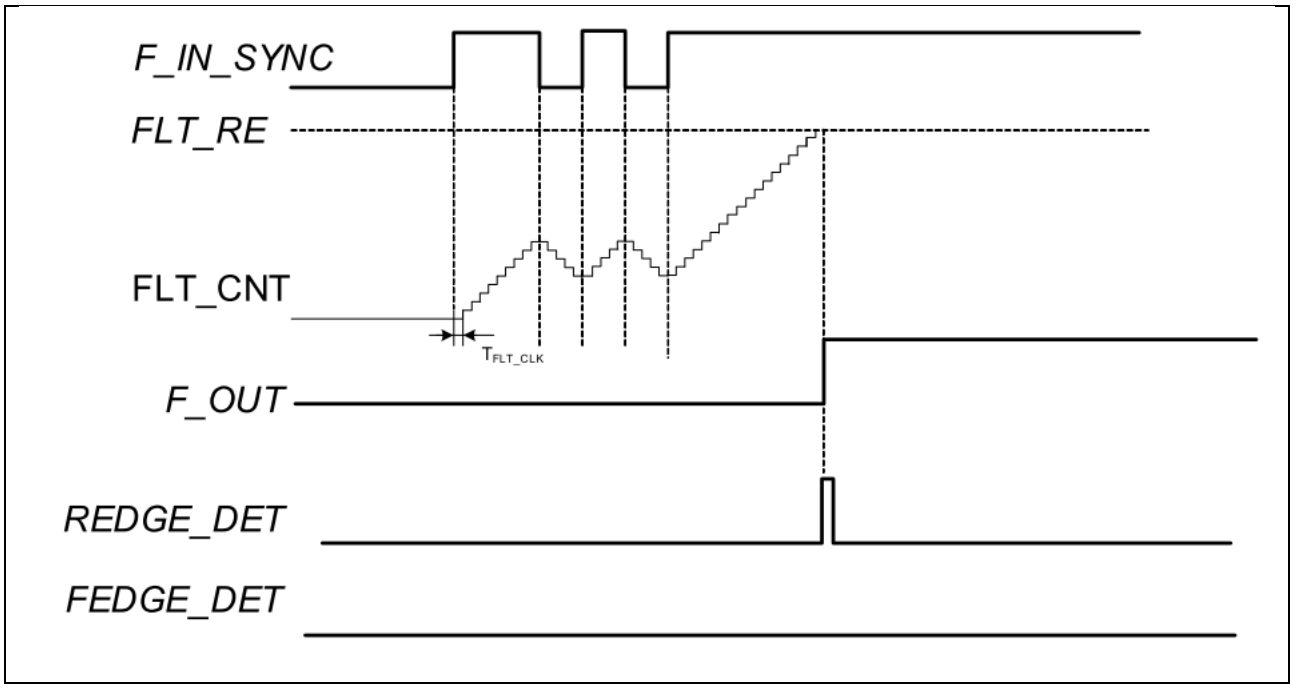


図 5-18 Individual de-glitch time (up/down counter)タイミング

このフィルタモードでは、フィルタカウント中に入力信号の変化(グリッチ)を検出した場合、TIM[i]_GLITCHDET[x]_IRQ 信号を生成します。TIM[i]_GLITCHDET[x]_IRQ については 5.7.1 を参照してください。

5.3.3 Individual de-glitch time (hold counter)

エッジ検出後、指定された期間中、入力信号レベルが同じ場合に出力信号レベルを変化させます。入力信号がフィルタカウント中にレベル反転した場合、カウントを停止してフィルタリング時間を延長します。

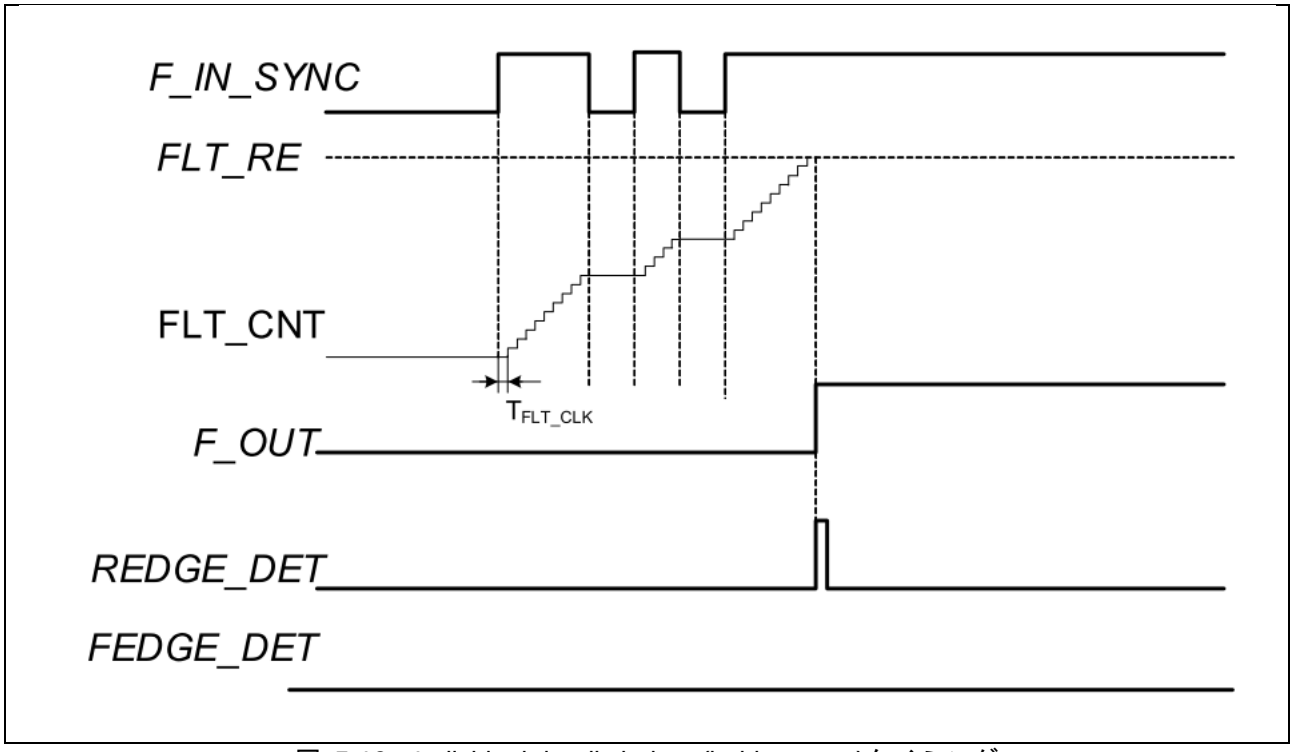


図 5-19 Individual de-glitch time (hold counter)タイミング

このフィルタモードでは、フィルタカウント中に入力信号の変化(グリッチ)を検出した場合、TIM[i]_GLITCHDET[x]_IRQ 信号を生成します。TIM[i]_GLITCHDET[x]_IRQ については 5.7.1 を参照してください。

5.3.4 Individual de-glitch mode (reset counter)

エッジ検出後、指定された期間、入力信号レベルが同じ場合に出力信号レベルを変化させます。入力信号がフィルタカウント中にレベル反転した場合、カウントをリセットしてフィルタ時間を延長します。

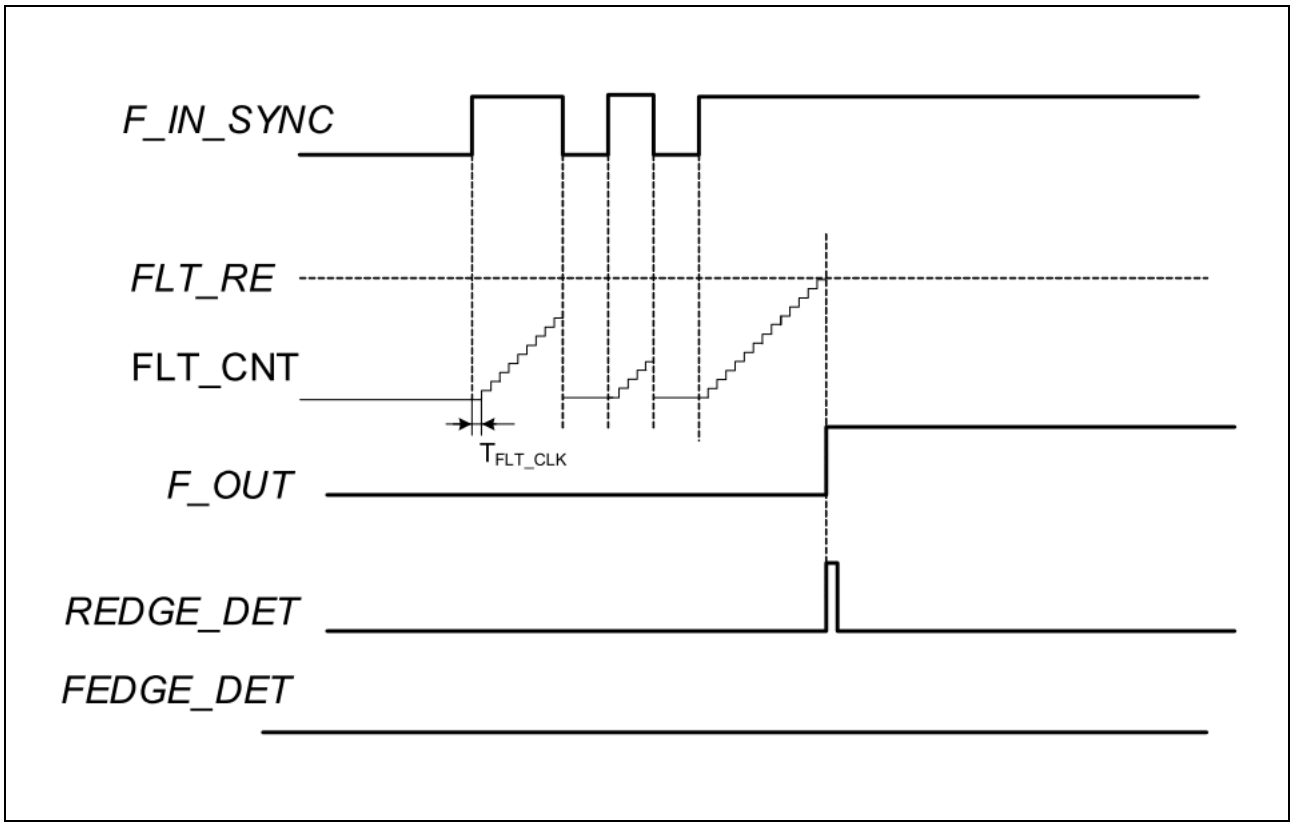


図 5-20 Individual de-glitch time (reset counter)タイミング

このフィルタモードでは、フィルタカウント中に入力信号の変化(グリッチ)を検出した場合、TIM[i]_GLITCHDET[x]_IRQ 信号を生成します。TIM[i]_GLITCHDET[x]_IRQ については 5.7.1 を参照してください。

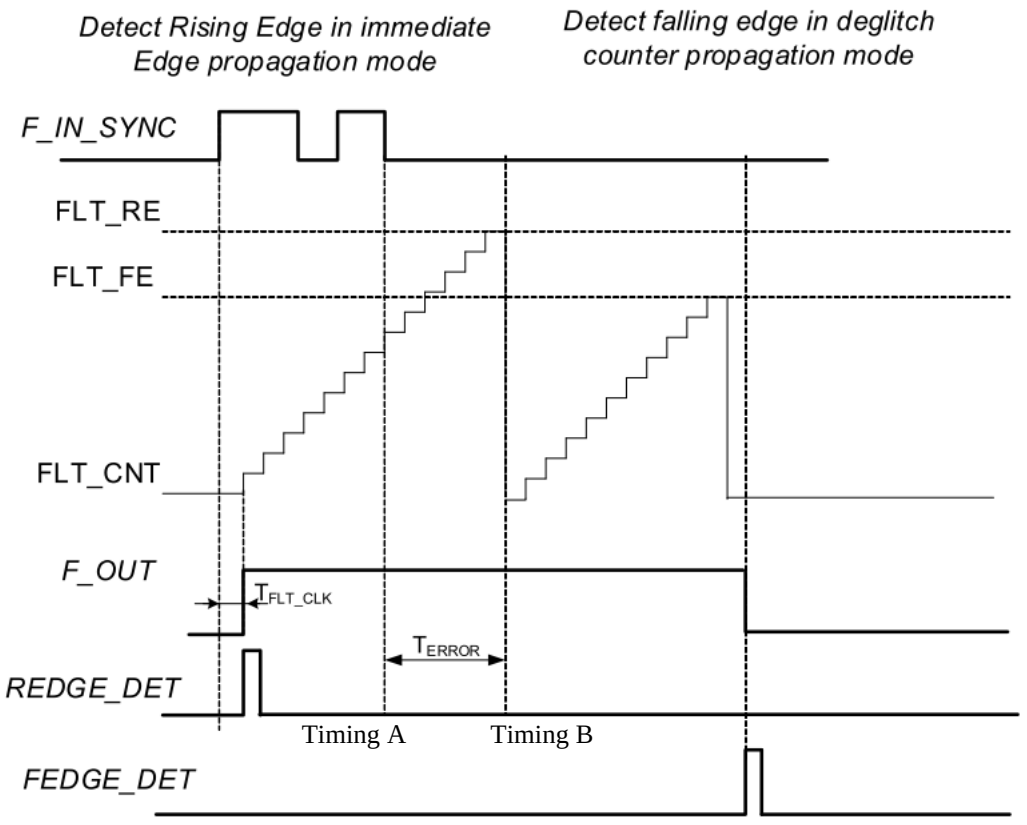
5.3.5 フィルタ使用時の注意点

フィルタ機能を使用する場合は次の点に注意してください。

(1) フィルタの出力信号は、フィルタ機能に供給されているクロックの 1 クロック分遅延して出力されます。

(2) フィルタ機能は、立ち上がり時のみ、または立ち下がり時のみの設定は不可です。フィルタ機能を有効にした場合、両エッジに対するフィルタモードやフィルタカウンタ値の設定を行ってください。

(3) フィルタカウンタ値の設定は適切に行ってください。フィルタへの入力信号周波数に対して過剰な値を設定した場合、次の図のようにフィルタ出力される信号が意図しない状態となります。
(例えば、図 5-21 のように T_{ERROR} の期間信号が延長されます)。



- Timing A で入力信号が Low レベルで安定しています。
- 立ち上がり時のフィルタカウントが終了しないため、立ち下がり時のフィルタカウントが開始できません。
- 立ち下がり時のフィルタカウントが開始できるのは、立ち上がり時のフィルタカウントが終了した後です。
- この結果 T_{ERROR} の間信号がハイレベルとなります。

図 5-21 フィルタカウンタ値が不適切な場合

5.4 信号入力選択機能

TIM へ入力される信号を選択する機能です。選択可能な信号は以下のとおりです。

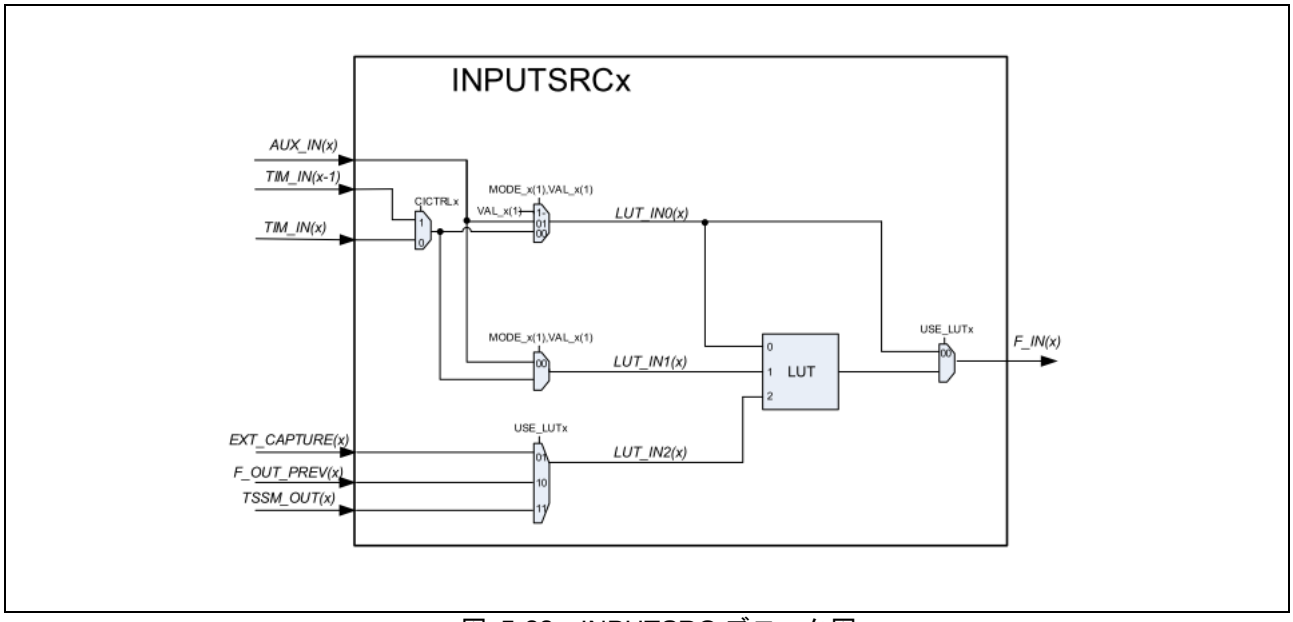


図 5-22 INPUTSRC ブロック図

表 5-14 選択可能な信号入力一覧

信号名	説明
TIM_IN[x]	対象チャネルの入力端子の信号。
TIM_IN[x-1]	対象チャネルより一つ前のチャネルの入力端子の信号。 対象チャネルがチャネル 1 の場合、チャネル 0 となります。 また、対象チャネルが 0 の場合、当該 TIM サブモジュールの最大チャネルとなります。
AUX_IN[x]	同サブモジュール番号かつ同チャネル番号の DTM 正相または一つ後のチャネルの DTM 逆相の出力信号。 また、対象チャネルが最大チャネルの場合、一つ後のチャネルは当該 TIM サブモジュールのチャネル 0 となります。
VAL_[x](1)	レジスタによる直値指定。
LUT	LUT_IN0(x)、LUT_IN1(x)、LUT_IN2(x)を入力に 8 ビットのルックアップテーブルによって生成される信号。 GTM[g].TIM[i]_CH[x]_TDUC レジスタの TO_CNT2 ビットの内容で定義されます。lookup_table_index は LUT_IN2(x) & LUT_IN1(x) &LUT_IN0(x)によって定義され、F_IN(x)は TO_CNT2[lookup_table_index]によって生成されます。

GTM[g].TIM[i]_CH[x]_CTRL レジスタの CICTRL ビット、GTM[g].TIM[i]_IN_SRC レジスタの各 MODE_[x]ビット、および VAL_[x]ビット、GTM[g].TIM[i]_CH[x]_ECTRL レジスタの USE_LUT ビットにて設定可能です。

リセット時、選択される信号入力は TIM_IN[x]です。

5.5 キャプチャトリガ選択

TIM へ入力される信号のキャプチャトリガを選択する機能です。キャプチャトリガを設定することで、隣接するタイマチャネル等と連携してキャプチャを行うことが出来ます。

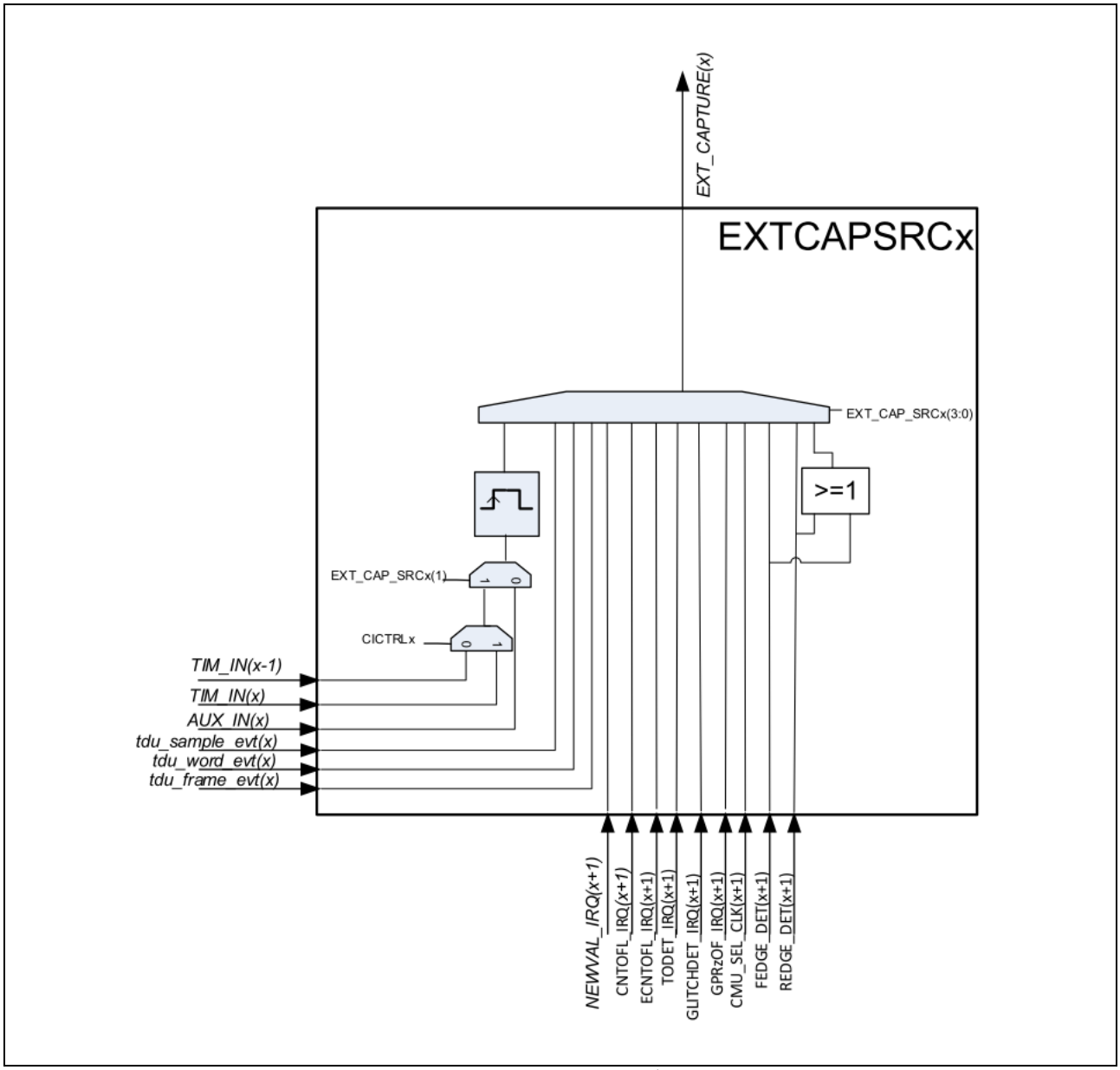


図 5-23 EXTCAPSRC ブロック図

選択可能なトリガは以下のとおりです。

表 5-15 選択可能なキャプチャトリガー覧

信号名	説明
TIM_IN[x]	対象チャネルの入力端子の信号の立ち上がりエッジ。
TIM_IN[x-1]	対象チャネルより一つ前のチャネルの入力端子の信号の立ち上がりエッジ。対象チャネルが 0 の場合、一つ前のチャネルは当該サブモジュールの最大チャネルとなります。
AUX_IN[x]	同サブモジュール番号かつ同チャネル番号の DTM 正相または一つ後のチャネルの DTM 逆相の出力信号の立ち上がりエッジ。 また、対象チャネルが最大チャネルの場合、一つ後のチャネルは当該 TIM サブモジュールのチャネル 0 となります。
tdu_sample_evt(x)	対象チャネルの TDU の出力信号。
tdu_word_evt(x)	対象チャネルの TDU の出力信号。
tdu_frame_evt(x)	対象チャネルの TDU の出力信号。
NEWVAL[x+1]_IRQ(※1)	対象チャネルより一つ後のチャネルの TIM_NEWVAL_IRQ 信号(※2)の立ち上がりエッジ。
ECNTOFL[x+1]_IRQ(※1)	対象チャネルより一つ後のチャネルの TIM_ECNTOFL_IRQ 信号(※2)の立ち上がりエッジ。
CNTOFL[x+1]_IRQ(※1)	対象チャネルより一つ後のチャネルの TIM_CNTOFL_IRQ 信号(※2)の立ち上がりエッジ。
GPROFL[x+1]_IRQ(※1)	対象チャネルより一つ後のチャネルの TIM_GPROFL_IRQ 信号(※2)の立ち上がりエッジ。
TODET[x+1]_IRQ(※1)	対象チャネルより一つ後のチャネルの TIM_TODET_IRQ 信号(※2)の立ち上がりエッジ。
GLITCHDET[x+1]_IRQ(※1)	対象チャネルより一つ後のチャネルの TIM_GLITCHDET_IRQ 信号(※2)の立ち上がりエッジ。
CMU_CLK[x+1](※1)	対象チャネルより一つ後のチャネルに入力されるクロックの立ち上がりエッジ。
FEDGE_DET[x+1](※1)	対象チャネルより一つ後のチャネルのフィルタ出力信号の立ち下がりエッジ。
REDGE_DET[x+1](※1)	対象チャネルより一つ後のチャネルのフィルタ出力信号の立ち上がりエッジ。

※1：対象チャネル[x]が最大(8 チャネル存在する場合はチャネル 7)の場合、[x+1]は 0 になります。

※2：各 IRQ 信号の詳細は、5.7.1 を参照してください。

これらのトリガは TIM[i]_EXT_CAPTURE[x]信号として外部サブモジュール(ATOM や MCS)へ通知されます。TIM チャネル x へのキャプチャトリガは GTM[g].TIM[i]_CH[x]_CTRL レジスタの EXT_CAP_EN ビットを設定することで有効となります。

GTM[g].TIM[i]_CH[x]_CTRL レジスタの CICTRL ビットと GTM[g].TIM[i]_CH[x]_ECTRL レジスタの EXT_CAP_SRC ビットにてキャプチャトリガ信号を設定します。

リセット後、キャプチャトリガは NEWVAL[x+1]_IRQ が選択され、TIM チャネル x へのキャプチャトリガは無効になります。

5.6 その他の制御

5.6.1 タイムアウト設定

入力信号のタイムアウト設定は、GTM[g].TIM[i]_CH[x]_CTRL レジスタの TOCTRL ビットへ設定します。なお、リセット後は無効となります。本ビットにて立ち上がりエッジのみ／立ち下がりエッジのみ／両エッジのいずれかをタイムアウト監視対象とするかを設定します。

タイムアウトのクロックチャネルの選択、およびタイムアウト値は GTM[g].TIM[i]_CH[x]_TDUV レジスタにて設定します。

タイムアウトが発生した場合、TIM[i]_TODET[x]_IRQ 信号を生成します。TIM[i]_TODET[x]_IRQ については 5.7.1 を参照してください。

また、タイムアウトのカウントの状況は、GTM[g].TIM[i]_CH[x]_TDUC レジスタにて参照可能です。

5.6.2 測定継続設定

ワンショット測定、または連続測定を選択します。GTM[g].TIM[i]_CH[x]_CTRL レジスタの OSM ビットを設定します。また、リセット後は連続測定となります。

なお、ワンショット測定の場合、測定(GPR0 や GPR1 にデータを格納)が 1 回終了すると、GTM[g].TIM[i]_CH[x]_CTRL レジスタの TIM_EN ビットは自動的にクリアされ、チャネルが無効となります。

5.6.3 測定データの選択

GTM[g].TIM[i]_CH[x]_CTRL レジスタの EGPR0_SEL ビット、EGPR1_SEL ビット、GPR0_SEL ビット、および GPR1_SEL ビットを設定することで、GPR0 レジスタ、および GPR1 レジスタへ格納するデータを選択することができます。選択が可能なデータは TBU チャネル 0～2 のタイムスタンプ、CNT レジスタの内容、CNTS レジスタの内容、ECNT レジスタの内容、現在の信号状態です。ARU を使用する場合は GPR0、GPR1 のデータのみが転送されるため、適切なデータが格納されるように設定する必要があります。

5.6.4 即時測定開始設定

TPWM モードと TPIM モードは GTM[g].TIM[i]_CH[x]_ECTRL レジスタの IMM_START ビットを設定することで TIM 有効(GTM[g].TIM[i]_CH[x]_CTRL レジスタの TIM_EN1 ビット)と同時に測定を開始するかどうか選択できます。

5.6.5 キャプチャタイミングのスワップ設定

TPWM モードと TPIM モードは GTM[g].TIM[i]_CH[x]_ECTRL レジスタの SWAP_CAPTURE ビットを設定することで CNTS と GPR1 のキャプチャタイミングを入れ替えることができます。

5.6.6 ARU への転送制御

GPR0 および GPR1 と各種ステータスの ARU への転送有無を設定します。GTM[g].TIM[i]_CH[x]_CTRL レジスタの ARU_EN ビットを設定することで転送が有効になります。リセット後は ARU への転送は無効となります。

5.7 測定結果等の通知

TIM からマイコンや他のサブモジュールに対して、測定結果や状態を通知する手段を説明します。

5.7.1 割り込み通知

TIM よりマイコンへ通知される割り込みは以下のとおりです。

表 5-16 TIM の割り込み通知一覧

IRQ	説明
TIM[i]_NEWVAL[x]_IRQ	新規測定データ取得通知。 GPR0、GPR1 へデータ格納が格納された場合に通知されます。
TIM[i]_ECNTOFL[x]_IRQ	エッジカウンタのオーバフロー通知。 ECNT のカウンタがオーバフローした場合に通知されます。
TIM[i]_CNTOFL[x]_IRQ	カウンタ(CNT)のオーバフロー通知。 CNT のカウンタがオーバフローした場合に通知されます。
TIM[i]_GPROFL[x]_IRQ	GPR0／GPR1 オーバライト通知。 GPR0、GPR1 に格納されたデータが、マイコンまたは ARU により読み込まれる前に、GPR0、GPR1 へ次のデータが格納された場合に通知されます。
TIM[i]_TODET[x]_IRQ	タイムアウト通知。 タイムアウトが発生した場合に通知されます。
TIM[i]_GLITCHDET[x]_IRQ	グリッチ検出通知。 入力フィルタにてグリッチを検出した場合に通知されます。

割り込みを有効にするためには、GTM[g].TIM[i]_CH[x]_IRQ_MODE レジスタへ適切な割り込みモードを設定し、GTM[g].TIM[i]_CH[x]_IRQ_EN レジスタおよび GTM[g].TIM[i]_CH[x]_EIRQ_EN レジスタにて各割り込みの有効／無効を設定してください。

また、GTM[g].TIM[i]_CH[x]_NOTIFY レジスタにて、各割り込みの要因クリアを行うことが出来ます。

5.7.2 ARU への転送

ARU 経由で TIM のデータを受信する場合は、対応する ARU アドレスを指定してください。

GTM-IP 358 において、TIM の ARU アドレス(TIM に割り当てられた ARU バッファのインデックス)は次の表のとおりです。

表 5-17 TIM の各チャネルの ARU アドレス

TIM チャネル番号	ARU アドレス			
	TIM0	TIM1	TIM2	TIM3
0	$(01)_{16}$	$(09)_{16}$	$(11)_{16}$	$(19)_{16}$
1	$(02)_{16}$	$(0a)_{16}$	$(12)_{16}$	$(1a)_{16}$
2	$(03)_{16}$	$(0b)_{16}$	$(13)_{16}$	$(1b)_{16}$
3	$(04)_{16}$	$(0c)_{16}$	$(14)_{16}$	$(1c)_{16}$
4	$(05)_{16}$	$(0d)_{16}$	$(15)_{16}$	$(1d)_{16}$
5	$(06)_{16}$	$(0e)_{16}$	$(16)_{16}$	$(1e)_{16}$
6	$(07)_{16}$	$(0f)_{16}$	$(17)_{16}$	$(1f)_{16}$
7	$(08)_{16}$	$(10)_{16}$	$(18)_{16}$	$(20)_{16}$

TIM より ARU へ転送されるデータのフォーマットは以下のとおりです。

表 5-18 送信 ARU データフォーマット

ビット	ビット名	説明
52	ACB	未使用
51		未使用
50		タイムアウト発生有無が設定されます。 0：タイムアウト未発生 1：タイムアウト発生
49		GPR0／GPR1 オーバライト発生有無が設定されます。 0：オーバライト未発生 1：オーバライト発生
48		FLT の出力信号レベルが設定されます。 0：Low 1：High
47～24	Data1	GPR1 レジスタの内容
23～0	Data0	GPR0 レジスタの内容

5.8 使用例

信号入力処理例を下記にて説明します。なお、割り込み処理、および割り込みルーチンの割り込みベクタテーブルへの登録方法については、本アプリケーションノートでは説明しません。なお、このソースコードでは CMU や TBU の初期設定については省略しています。

5.8.1 PWM 信号測定例 (TPWM)

GTM0 の TIM0 のチャネル 0 にて PWM 信号を測定し、デューティと周期を取得する場合のコードの設定例は以下のとおりです。

設定例：

```
static unsigned long g_u32Duty;
static unsigned long g_u32Cycle;

void gtm0_tim0_ch0_main(void)
{
    GTM0.TIM0_CH0_CTRL.UINT32 = 0x00002f00; //TIM channel mode is TPWM
                                            //CMU clock source is CMU_CLK0
                                            //Duty is high level
                                            //GPR1 stores CNT register
                                            //GPR0 stores CNTS register
                                            //Channel is invalid
    while(GTM0.TIM0_CH0_CTRL.UINT32 != 0x00002f00);
    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x0;      //Interrupt is invalid
    GTM0.TIM0_CH0_IRQ_MODE.UINT32 = 0x2;    //Interrupt mode (Pulse-Notify)
    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x1;      //NEWVAL_IRQ interrupt is valid
    GTM0.TIM0_CH0_CTRL.UINT32 |= 0x1;       //Channel is valid
}
void irq_gtm0_tim0_ch0(void)
{
    g_u32Cycle = GTM0.TIM0_CH0_GPR1.BIT.GPR1;  // get GPR1(cycle of PWM)
    g_u32Duty = GTM0.TIM0_CH0_GPR0.BIT.GPR0;   // get GPR0(duty of PWM)
    GTM0.TIM0_CH0_IRQ_NOTIFY.UINT32 = 0x1;     // clear NEWVAL_IRQ
}
```

図 5-24 TIM PWM 信号測定プログラムコード例

5.8.2 パルス入力期間測定例 (TPIM)

GTM0 の TIM0 のチャネル 0 にて入力信号の測定レベルを設定し、入力信号の High レベル期間を取得する場合のコードの設定例は以下のとおりです。

設定例：

```
static unsigned long g_u32PreHighLevelPeriod;
static unsigned long g_u32HighLevelPeriod;

void gtm0_tim0_ch0_main(void)
{
    GTM0.TIM0_CH0_CTRL.UINT32 = 0x00002f02; //TIM channel mode is TPIM
                                            //CMU clock source is CMU_CLK0
                                            //High level measurement
                                            //GPR1 register stores CNT register
                                            //GPR0 register stores CNTS register
                                            //Channel is invalid
    while(GTM0.TIM0_CH0_CTRL.UINT32 != 0x00002f02);
    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x0;      //Interrupt is invalid
    GTM0.TIM0_CH0_IRQ_MODE.UINT32 = 0x2;    //Interrupt mode (Pulse-Notify)
    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x1;      //NEWVAL_IRQ interrupt is valid
    GTM0.TIM0_CH0_CTRL.UINT32 |= 0x1;       //Channel is valid
}

void irq_gtm0_tim0_ch0(void)
{
    g_u32HighLevelPeriod = GTM0.TIM0_CH0_GPR1.BIT.GPR1; //get GPR1(high level total period)
    g_u32PreHighLevelPeriod = GTM0.TIM0_CH0_GPR0.BIT.GPR0; //get GPR0(previous high level total period)
    GTM0.TIM0_CH0_IRQ_NOTIFY.UINT32 = 0x1; //clear NEWVAL_IRQ
}
```

図 5-25 TIM 入力期間 測定プログラムコード例

5.8.3 入力イベント測定例 (TIEM)

GTM0 の TIM0 のチャネル 0 にて入力信号のエッジ検出を設定し、立ち上がりエッジ検出のタイムスタンプ、およびエッジ数を取得する場合のコードの設定例は以下のとおりです。

設定例：

```
static unsigned long g_u32EdgeCount;
static unsigned long g_u32TimeStamp;

void gtm0_tim0_ch0_main( void )
{
    GTM0.TIM0_CH0_CTRL.UINT32 = 0x0002c04;  //TIM channel mode is TIEM
                                            //CMU clock source is CMU_CLK0
                                            //Detect edge is riging edge
                                            //GPR1 register stores CNT register
                                            //GPR0 register stores TBU channel 0
                                            //Channel is invalid
    while(GTM0.TIM0_CH0_CTRL.UINT32 != 0x0002c04);
    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x0;      //Interrupt is invalid
    GTM0.TIM0_CH0_IRQ_MODE.UINT32 = 0x2;    //Interrupt mode (Pulse-Notify)
    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x1;      //NEWVAL_IRQ interrupt is valid
    GTM0.TIM0_CH0_CTRL.UINT32 |= 0x1;       //Channel is valid }

void irq_gtm0_tim0_ch0(void)
{
    g_u32EdgeCount = GTM0.TIM0_CH0_GPR1.BIT.GPR1; //get GPR1(EdgeCount)
    g_u32TimeStamp = GTM0.TIM0_CH0_GPR0.BIT.GPR0; //get GPR0(timestamp)
    GTM0.TIM0_CH0_IRQ_NOTIFY.UINT32 = 0x1;        //clear NEWVAL_IRQ
}
```

図 5-26 TIM 入力位置 測定プログラムコード例

5.8.4 プリスケールエッジ信号測定例 (TIPM)

GTM0 の TIM0 のチャネル 0 にてプリスケールを設定し、入力信号両エッジのタイムスタンプ、および入力信号のエッジ数を取得する場合のコードの設定例は以下のとおりです。

設定例：

```
#define EDGE_COUNT_MAX 10 //edge count times

static unsigned long g_u32EdgeCount;
static volatile unsigned long g_u32TimeStamp;

void gtm0_tim0_ch0_main(void)
{
    GTM0.TIM0_CH0_CTRL.UINT32 = 0x20004006; //TIM channel mode is TIPM
                                            //CMU clock source is CMU_CLK0
                                            //Detect edge is both edge
                                            //GPR1 register stores ECNT register
                                            //GPR0 register stores TBU channel 0
                                            //Channel is invalid
    while(GTM0.TIM0_CH0_CTRL.UINT32 != 0x20004006);
    GTM0.TIM0_CH0_CNTS.UINT32 = (EDGE_COUNT_MAX - 1); //Max edge count times for dividing frequency

    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x0;      //Interrupt is invalid
    GTM0.TIM0_CH0_IRQ_MODE.UINT32 = 0x2;    //Interrupt mode (Pulse-Notify)
    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x1;      //NEWVAL_IRQ interrupt is valid
    GTM0.TIM0_CH0_CTRL.UINT32 |= 0x1;       //Channel is valid
}

void irq_gtm0_tim0_ch0(void)
{
    g_u32EdgeCount = GTM0.TIM0_CH0_GPR1.BIT.GPR1; //get GPR1(edge count)
    g_u32TimeStamp = GTM0.TIM0_CH0_GPR0.BIT.GPR0; //get GPR0(time stamp)
    GTM0.TIM0_CH0_IRQ_NOTIFY.UINT32 = 0x1;        //clear NEWVAL_IRQ
}
```

図 5-27 TIM プリスケールエッジ信号 測定プログラムコード例

5.8.5 パラレル信号測定例 (TBCM)

GTM0 の TIM0 にて入力信号のエッジ検出を設定し、入力信号両エッジのタイムスタンプ、および TIM0 がもつ全チャネルの入力信号レベルをパラレルに取得する場合のコードの設定例は以下のとおりです。

設定例：

```
static unsigned long g_u32Level;
static volatile unsigned long g_u32TimeStamp;

void gtm0_tim0_ch0_main(void)
{
    GTM0.TIM0_CH0_CTRL.UINT32 = 0x00000008; //TIM channel mode is TIPM
                                            //CMU clock source is CMU_CLK0
                                            //Detect edge is both edge
                                            //GPR1 register stores ECNT register
                                            //GPR0 register stores TBU channel 0
                                            //Channel is invalid
    while(GTM0.TIM0_CH0_CTRL.UINT32 != 0x00000008);
    GTM0.TIM0_CH0_CNTS.UINT32 = 0xFFFF; //detect both edges of TIM all channel

    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x0;      //Interrupt is invalid
    GTM0.TIM0_CH0_IRQ_MODE.UINT32 = 0x2;    //Interrupt mode (Pulse-Notify)
    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x1;      //NEWVAL_IRQ interrupt is valid
    GTM0.TIM0_CH0_CTRL.UINT32 |= 0x1;       //Channel is valid
}

void irq_gtm0_tim0_ch0( void )
{
    g_u32Level = GTM0.TIM0_CH0_GPR1.BIT.GPR1; //get GPR1(level)
    g_u32TimeStamp = GTM0.TIM0_CH0_GPR0.BIT.GPR0; //get GPR0(time stamp)
    GTM0.TIM0_CH0_IRQ_NOTIFY.UINT32 = 0x1;   //clear NEWVAL_IRQ
}
```

図 5-28 TIM パラレル信号 測定プログラムコード例

5.8.6 ゲート期間サンプリング測定例 (TGPS)

GTM0 の TIM0 のチャネル 0 にて測定対象のレベル設定し、入力信号両レベルの合計期間が指定クロック数となった時のタイムスタンプとエッジ数を取得します。

設定例：

```
#define TIM_CAPTURE_CYCLE 1000

static unsigned long g_u32EdgeCount;
static volatile unsigned long g_u32TimeStamp;

void gtm0_tim0_ch0_main(void)
{
    GTM0.TIM0_CH0_CTRL.UINT32 = 0x2000c00a; //TIM channel mode is TGPS
                                            //CMU clock source is CMU_CLK0
                                            //ECNT counter is reset with periodic sampling
                                            //Detect edge is both edge
                                            //GPR1 register stores ECNT register
                                            //GPR0 register stores TBU channel 0
                                            //Channel is invalid
    while(GTM0.TIM0_CH0_CTRL.UINT32 != 0x2000c00a);
    GTM0.TIM0_CH0_CNTS.UINT32 = TIM_CAPTURE_CYCLE - 1;
    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x0;      //Interrupt is invalid
    GTM0.TIM0_CH0_IRQ_MODE.UINT32 = 0x2;    //Interrupt mode (Pulse-Notify)
    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x1;      //NEWVAL_IRQ interrupt is valid
    GTM0.TIM0_CH0_CTRL.UINT32 |= 0x1;       //Channel is valid
}

void irq_gtm0_tim0_ch0(void)
{
    g_u32EdgeCount = GTM0.TIM0_CH0_GPR1.BIT.GPR1; //get GPR1(edge count)
    g_u32TimeStamp = GTM0.TIM0_CH0_GPR0.BIT.GPR0; //get GPR0(time stamp)
    GTM0.TIM0_CH0_IRQ_NOTIFY.UINT32 = 0x1;        //clear NEWVAL_IRQ
}
```

図 5-29 TIM ゲート期間サンプリング 測定プログラムコード例

5.8.7 シリアル信号測定例 (TSSM)

GTM0 の TIM0 のチャネル 0 にて入力信号のサンプリングクロックを CMU_CLK0、サンプリング数を 8 ビット、シフト方向を右シフトに設定し、8 ビットのシリアルデータとタイムスタンプを取得します。

設定例：

```
#define SHIFT_BIT_NUM 8

static unsigned long g_u32Data;
static volatile unsigned long g_u32TimeStamp;

void gtm0_tim0_ch0_main(void)
{
    GTM0.TIM0_CH0_CTRL.UINT32 = 0x00002c0c; //TIM channel mode is TSSM
                                            //CMU clock source is CMU_CLK0
                                            //Shift direction is right
                                            //GPR1 register stores CNT register
                                            //GPR0 register stores TBU channel 0
                                            //Channel is invalid
    while(GTM0.TIM0_CH0_CTRL.UINT32 != 0x00002c0c);
    GTM0.TIM0_CH0_CNTS.UINT32 = SHIFT_BIT_NUM - 1; //CNTS[7:0] is bit-shift times
    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x0;      //Interrupt is invalid
    GTM0.TIM0_CH0_IRQ_MODE.UINT32 = 0x2;    //Interrupt mode (Pulse-Notify)
    GTM0.TIM0_CH0_IRQ_EN.UINT32 = 0x1;      //NEWVAL_IRQ interrupt is valid
    GTM0.TIM0_CH0_CTRL.UINT32 |= 0x1;       //Channel is valid

}

void irq_gtm0_tim0_ch0(void)
{
    g_u32Data = GTM0.TIM0_CH0_GPR1.BIT.GPR1; //get GPR1(data)
    g_u32TimeStamp = GTM0.TIM0_CH0_GPR0.BIT.GPR0; //get GPR0(time stamp)
    GTM0.TIM0_CH0_IRQ_NOTIFY.UINT32 = 0x1;   //clear NEWVAL_IRQ
}
```

図 5-30 TIM シリアル信号 測定プログラムコード例

6. 信号出力機能

ATOM(ARU-Connected Timer Output Module)は、ARU を経由して制御を行い、信号を出力するサブモジュールです。マイコンより直接制御をすることも可能です。ATOM は、1 つのサブモジュールにつき最大 8 チャネル(注 1)の信号出力が可能です。

6.1 構成

6.1.1 全体構成

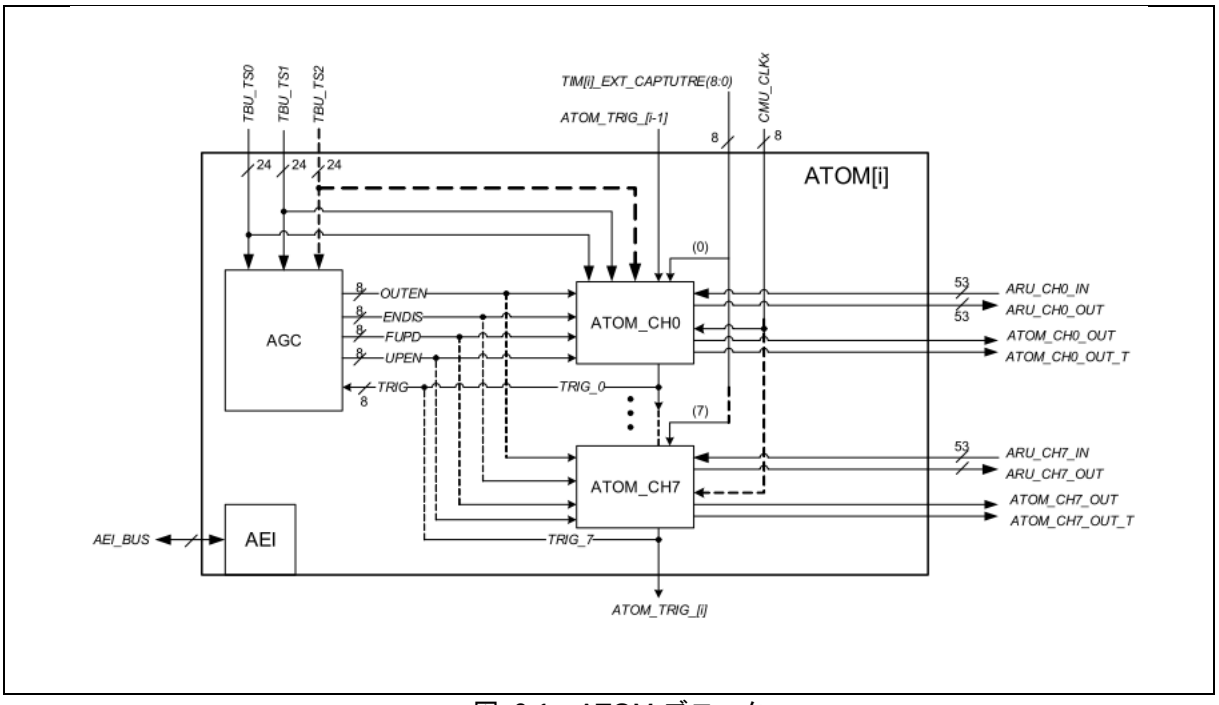


図 6-1 ATOM ブロック

(1) AGC (ATOM Global Control)
各チャネルの有効／無効や信号出力の有効／無効等を行います。内部／外部イベントに同期して、複数の処理や複数チャネルの処理を同時に行うことも出来ます。

(2) ATOM_CH[x] (ATOM Channel [x])
信号出力チャネルです。ARU インタフェースやコンペアマッチユニットなどがあります。

(注 1)：使用可能なチャネル数は、マイコンおよび端子数により異なります。

6.1.2 ATOM チャネルの構造

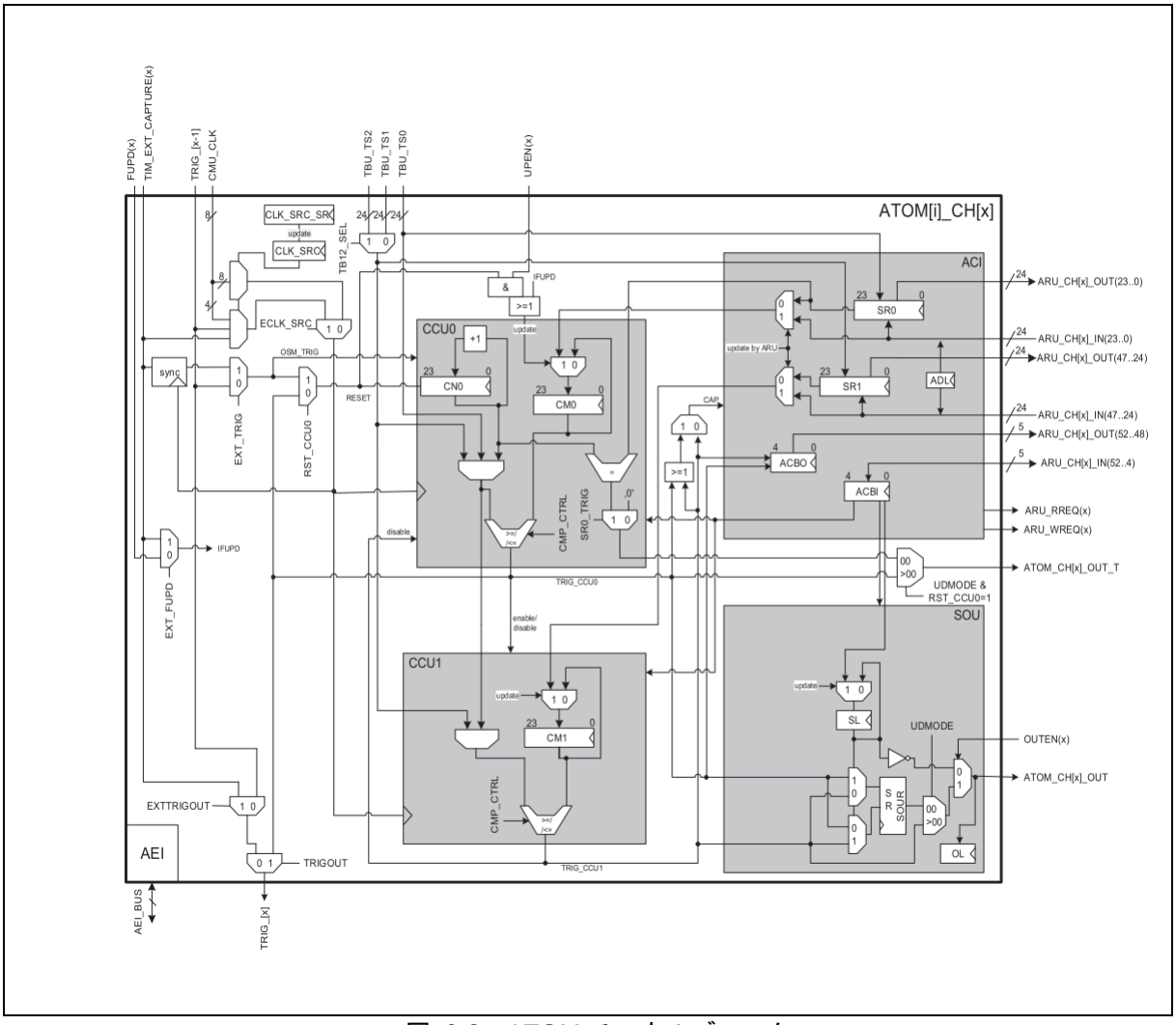


図 6-2 ATOM チャネルブロック

(1) CCU0 (Compare Unit 0)
コンペアユニット 0 です。チャネルモードにより、CM0 レジスタに設定された値と、TBU チャネルのタイムスタンプまたは CN0 カウンタを比較します。
CM0 レジスタへのデータの格納方法は、直接設定する方法と、ACI の SR0 レジスタを経由する方法があります。
CN0 カウンタは、指定した CMU クロックごとにインクリメントカウントを行うカウンタです。
CN0 ≧ CM0 になるまでカウントされます。
CM0 レジスタは GTM[g].ATOM[i]_CH[x]_CM0 レジスタ、CN0 カウンタは GTM[g].ATOM[i]_CH[x]_CN0 レジスタにて、それぞれ設定可能です。

(2) CCU1 (Compare Unit 1)
コンペアユニット 1 です。チャネルモードにより、CM1 レジスタに設定された値と、TBU チャネルのタイムスタンプまたは CN0 レジスタを比較します。
CM1 レジスタへのデータの格納方法は、直接設定する方法と、ACI の SR1 レジスタを経由する方法があります。
CM1 レジスタは GTM[g].ATOM[i]_CH[x]_CM1 レジスタにて設定可能です。

(3) ACI (ARU Communication Interface)
ARU とのインタフェースユニットです。CM0 レジスタのシャドウレジスタである SR0 レジスタ、および CM1 レジスタのシャドウレジスタである SR1 レジスタ、受信した ARU データの ACB ビットを保存する ACBI レジスタ、送信する ARU データの ACB ビットへ設定する内容を保持する ACBO レジスタがあります。
SR0 レジスタは GTM[g].ATOM[i]_CH[x]_SR0、SR1 レジスタは GTM[g].ATOM[i]_CH[x]_SR1 レジスタにて、それぞれ設定可能です。また ACBI レジスタ、および ACBO レジスタは GTM[g].ATOM[i]_CH[x]_STAT レジスタの ACBI ビット、および ACBO ビットにて参照可能です。

(4) SOU (Signal Output Unit)
信号出力ユニットです。AGC、CCU0、CCU1、および ACI の設定により出力信号レベルの High／Low、信号出力の有効／無効を制御します。
SL は信号出力時の信号レベルを設定する機能を持ち、GTM[g].ATOM[i]_CH[x]_CTRL レジスタの SL ビットにて設定可能です。

6.1.3 ATOM Global Control (AGC)

ATOM サブモジュールには、内部／外部イベントと同期してチャネルの制御を行う際に使用する、ATOM Global Control(AGC)が搭載されています。AGC は最大 8 チャネル(注 1)制御可能です。本ユニットは、以下の制御を実施します。

(1) ATOM チャネル機能(ATOM[i]_CH[x])の有効／無効トリガ設定
GTM[g].ATOM[i]_AGC_ENDIS_CTRL レジスタにて設定します。
GTM[g].ATOM[i]_AGC_ENDIS_CTRL レジスタに設定された値が反映されたかどうかは、
GTM[g].ATOM[i]_AGC_ENDIS_STAT レジスタにて確認します。
有効に設定した場合、以下要因で各レジスタの更新が実行されます。
1. マイコンによる出力トリガ
2. TBU のタイムスタンプ値による出力トリガ
3. GTM 内部信号による出力トリガ

要因については(5)、(6)、(7)の各項目を参照してください。

(2) ATOM チャネル機能(ATOM[i]_CH[x])の有効／無効制御
GTM[g].ATOM[i]_AGC_ENDIS_STAT レジスタにて設定します。
GTM[g].ATOM[i]_AGC_ENDIS_CTRL レジスタとは異なり、直接状態を制御します。
チャネル機能が無効になった場合、CCU0 の CN0 カウンタを停止し、信号出力状態は
GTM[g].ATOM[i]_CH[x]_CTRL レジスタの SL ビットの設定値を反転した状態となります。
チャネル機能が有効になった場合、CCU0 の CN0 カウンタは CN0 の現在の値からカウントを開始します。

(3) ATOM チャネル信号出力の有効／無効トリガ設定
GTM[g].ATOM[i]_AGC_OUTEN_CTRL レジスタにて設定します。
GTM[g].ATOM[i]_AGC_OUTEN_CTRL レジスタに設定された値が反映されたかどうかは、
GTM[g].ATOM[i]_AGC_OUTEN_STAT レジスタにて確認します。
有効に設定した場合、以下要因で各レジスタの更新が実行されます。
1. マイコンによる出力トリガ
2. TBU のタイムスタンプ値による出力トリガ
3. GTM 内部信号による出力トリガ

要因については(5)、(6)、(7)の各項目を参照してください。

(4) ATOM チャネル信号出力の有効／無効制御
GTM[g].ATOM[i]_AGC_OUTEN_STAT レジスタにて設定します。
信号出力が無効になった場合、信号出力状態は GTM[g].ATOM[i]_CH[x]_CTRL レジスタの SL ビットの設定値を反転した状態となります。

(5) マイコンによる出力トリガ設定
GTM[g].ATOM[i]_AGC_GLB_CTRL の HOST_TRIG ビットを設定します。本ビットは自動的にリセットされます。

(6) TBU のタイムスタンプ値による出力トリガ設定
GTM[g].ATOM[i]_AGC_ACT_TB レジスタにて設定します。

(7) GTM 内部信号による出力トリガ設定
ATOM[i]_TRIG_[x]による出力トリガを設定します。GTM[g].ATOM[i]_AGC_INT_TRIG レジスタにて設定します。

(注 1) 使用可能なチャネル数は、マイコンおよび端子数により異なります。

(8) CCU0 の CM0、CCU1 の CM1 および CLK_SRC の更新有効／無効制御
SR0 レジスタによる CM0 レジスタの更新、SR1 レジスタによる CM1 レジスタの更新、CLK_SRC_SR による CLK_SRC レジスタの更新を制御します。
GTM[g].ATOM[i]_AGC_GLB_CTRL レジスタの UPEN_CTRL[x]ビットにて更新有効／無効を設定します。有効に設定した場合、以下要因で各レジスタの更新が実行されます。
1. マイコンによる出力トリガ
2. TBU のタイムスタンプ値による出力トリガ
3. GTM 内部信号による出力トリガ
要因については(5)、(6)、(7)の各項目を参照してください。

(9) CN0 カウンタリセット強制実施、ATOM チャネル(CLK_SRC、CM0、CM1 など)の強制更新設定
CN0 カウンタリセットの強制実施や CLK_SRC、CM0 レジスタ、CM1 レジスタの強制更新を設定します。GTM[g].ATOM[i]_AGC_FUPD_CTRL レジスタの RSTCN0_CH[x]ビットにて CN0 カウンタのリセットの強制実施を、同レジスタの FUPD_CTRL[x]ビットにて ATOM チャネルの強制更新を設定します。設定後、以下要因で強制実施、強制更新が実行されます。
1. マイコンによる出力トリガ
2. TBU のタイムスタンプ値による出力トリガ
3. GTM 内部信号による出力トリガ
要因については(5)、(6)、(7)の各項目を参照してください。

これらの設定は、リセット後にすべて無効となります。したがって、ATOM より信号出力を行う場合、必要に応じて各種設定を行う必要があります。

6.2 チャネルモード

ATOM サブモジュールを使用した信号出力方法を説明します。本章では、出力制御設定はマイコンによる方法のみを説明します。ARU を使用した出力方法については 6.5 を参照してください。

各チャネルモードの初期設定は、チャネル機能を無効にした状態で実施してください(チャネル機能無効についての詳細は 6.1.3 を参照してください)。

6.2.1 ATOM Signal Output Mode Immediate (SOMI)

指定された状態で信号出力を行います。

表 6-1 SOMI における GTM[g].ATOM[i]_CH[x]_CTRL レジスタ設定項目一覧

ビット名	説明
SOMB	チャネルモードを SOMB とするかを設定します。 0：チャネルモードを SOMB 以外とします。 チャネルモードは MODE にて指定します。
SL	出力信号を設定します。 実際の出力信号レベルは表 6-2 を参照してください。
ACB(bit 0)	出力信号を設定します。 実際の出力信号レベルは表 6-2 を参照してください。
MODE	チャネルモードを設定します。 0：ATOM Signal Output Mode Immediate (SOMI)

表 6-2 SOMI 出力信号状態

SL	ACB(bit0)	信号の出力状態
0	0	信号出力は High となります。 (SL ビットで指定した信号レベルを反転して出力します。)
0	1	信号出力は Low となります。 (SL ビットで指定した信号レベルを出力します。)
1	0	信号出力は Low となります。 (SL ビットで指定した信号レベルを反転して出力します。)
1	1	信号出力は High となります。 (SL ビットで指定した信号レベルを出力します。)

6.2.2 ATOM Signal Output Mode Compare (SOMC)

TBU タイムスタンプ値と、コンペアマッチユニットのコンペアレジスタ(CM0 レジスタおよび CM1 レジスタ)にて指定されたタイムスタンプ値がコンペアマッチした場合に信号出力を制御します。コンペアマッチイベント発生時、イベントが発生したタイムスタンプがシャドウレジスタ(SR0 レジスタおよび SR1 レジスタ)へ設定されます。

このチャネルモードでは、CCU0 にてコンペアマッチが発生した場合には ATOM[i]_CCU0TC[x]_IRQ、CCU1 にてコンペアマッチが発生した場合には ATOM[i]_CCU1TC[x]_IRQ が通知されます。(詳細は 6.4.1 を参照してください。)

表 6-3 SOMC における GTM[g].ATOM[i]_CH[x]_CTRL レジスタ設定項目一覧

ビット名	説明
SOMB	チャネルモードを SOMB とするかを設定します。 0：チャネルモードを SOMB 以外とします。 チャネルモードは MODE にて指定します。
TRIGOUT	詳細は 6.3 信号出力トリガを参照ください。
EXTTRIGOUT	
SL	コンペアマッチ発生時の出力信号を設定します。 詳細は表 6-4、表 6-5 を参照してください。
CMP_CTRL	CM0 レジスタおよび CM1 レジスタの比較方法を設定します。 0：TBU のタイムスタンプ値 ≧ CM0 または CM1 1：TBU のタイムスタンプ値 ≦ CM0 または CM1
ACB(bit2～bit4)	コンペア動作モードを設定します。 詳細は表 6-6 を参照してください。
ACB(bit0～bit1)	コンペアマッチ発生時の出力信号を設定します。 詳細は表 6-4、表 6-5 を参照してください。
TB12_SEL	CM1 レジスタと比較するタイムスタンプを設定します。 0: TBU チャネル 1 のタイムスタンプ値 1: TBU チャネル 2 のタイムスタンプ値
MODE	チャネルモードを設定します。 1：ATOM Signal Output Mode Compare (SOMC)

表 6-4 SOMC コンペアマッチイベント発生時の出力信号(ACB(bit2～bit4)が$(001)_2$ 以外の場合)

SL	ACB		信号の出力状態
	bit1	bit0	
0	0	0	信号出力は変化しません。
0	0	1	信号出力を High にします。
0	1	0	信号出力を Low にします。
0	1	1	信号出力を反転します。
1	0	0	信号出力は変化しません。
1	0	1	信号出力を Low にします。
1	1	0	信号出力を High にします。
1	1	1	信号出力を反転します。

表 6-5 SOMC コンペアマッチイベント発生時の出力信号(ACB(bit2～bit4)が$(001)_2$の場合)

ACB		SL	CCU(※1)		信号の出力状態
bit1	bit0		CCU0 match	CCU1 match	
0	0	0	0	1	信号出力は変化しません。
			1	0	信号出力を反転します。
			1	1	信号出力は変化しません。
0	1	0	0	1	信号出力を Low にします。
			1	0	信号出力を High にします。
			1	1	信号出力を Low にします。
1	0	0	0	1	信号出力を High にします。
			1	0	信号出力を Low にします。
			1	1	信号出力を High にします。
1	1	0	0	1	信号出力を反転します。
			1	0	信号出力は変化しません。
			1	1	信号出力を反転します。
0	0	1	0	1	信号出力は変化しません。
			1	0	信号出力を反転します。
			1	1	信号出力は変化しません。
0	1	1	0	1	信号出力を High にします。
			1	0	信号出力を Low にします。
			1	1	信号出力を High にします。
1	0	1	0	1	信号出力を Low にします。
			1	0	信号出力を High にします。
			1	1	信号出力を Low にします。
1	1	1	0	1	信号出力を反転します。
			1	0	信号出力は変化しません。
			1	1	信号出力を反転します。

※1：CCU0 と CCU1 で同時にコンペアマッチが発生した場合、
CCU0 match と CCU1 match が両方とも 1 となります。

ACB ビット(bit2～bit4)にて設定可能な CCU0、および CCU1 の制御方法は次の表のとおりです。

表 6-6 SOMC コンペア動作モード

ACB(bit2～bit4)			CCU0, CCU1 の制御方法
bit4	bit3	bit2	
0	0	0	CCU1 と CCU0 での比較作業はパラレルに行います。どちらかのコンペアユニットでコンペアマッチとなった場合、残りのコンペアユニットは無効となります。 コンペアマッチした場合、信号レベルは表 6-4 に従い変化します。 CCU0 は TBU チャネル 0 のタイムスタンプ、CCU1 は TBU チャネル 1 または 2 のタイムスタンプと比較します。
0	0	1	CCU1 と CCU0 での比較作業はパラレルに行います。どちらかのコンペアユニットでコンペアマッチとなった場合、残りのコンペアユニットは無効となります。 コンペアマッチした場合、信号レベルは表 6-5 に従い変化します。 CCU0 は TBU チャネル 0 のタイムスタンプと、CCU1 は TBU チャネル 1 または 2 のタイムスタンプと比較します。
0	1	0	CCU0 のみで比較を行います。 コンペアマッチした場合、信号レベルは表 6-4 に従い変化します。 CCU0 は TBU チャネル 0 のタイムスタンプと比較します。
0	1	1	CCU1 のみで比較を行います。 コンペアマッチした場合、信号レベルは表 6-4 に従い変化します。 CCU1 は TBU チャネル 1 または 2 のタイムスタンプと比較します。
1	0	0	CCU0 で比較後、さらに CCU1 で比較します。 CCU0 でコンペアマッチした場合、信号レベルは表 6-4 に従い変化します。続けて CCU1 でコンペアマッチした場合、信号レベルを反転します。 CCU0、CCU1 ともに、TBU チャネル 0 のタイムスタンプと比較します。
1	0	1	CCU0 で比較後、さらに CCU1 で比較します。 CCU0 でコンペアマッチした場合、信号レベルは表 6-4 に従い変化します。続けて CCU1 でコンペアマッチした場合、信号レベルを反転します。 CCU0、CCU1 ともに、TBU チャネル 1 または 2 のタイムスタンプと比較します。
1	1	0	CCU0 で比較後、さらに CCU1 で比較します。 CCU0 でコンペアマッチしても信号は変化しません。続けて CCU1 でコンペアマッチした場合、信号レベルは表 6-4 に従い変化します。 CCU0 は TBU チャネル 0 のタイムスタンプと、CCU1 は TBU チャネル 1 または 2 のタイムスタンプと比較します。
1	1	1	ARU_EN にかかわらず保留中の比較をキャンセルします。

比較するタイムスタンプ値を設定するため、以下のレジスタを設定します。

表 6-7 SOMC GTM[g].ATOM[i]_CH[x]_CTRL レジスタ以外のレジスタ設定項目

レジスタ名	マイコン設定レジスタ	説明
CM0	GTM[g].ATOM[i]_CH[x]_CM0	CCU0 の比較タイムスタンプ値です。
CM1	GTM[g].ATOM[i]_CH[x]_CM1	CCU1 の比較タイムスタンプ値です。

CCU0 にてコンペアマッチが発生すると SR0 レジスタにコンペアマッチ発生時のタイムスタンプ値が転送され、さらに ACBO の bit3 が設定されます。同様に、CCU1 にてコンペアマッチが発生すると SR1 レジスタにコンペアマッチ発生時のタイムスタンプ値が転送され、さらに ACBO の bit4 が設定されます。

SR0 レジスタは GTM[g].ATOM[i]_CH[x]_SR0、SR1 レジスタは GTM[g].ATOM[i]_CH[x]_SR1、ACBO レジスタは GTM[g].ATOM[i]_CH[x]_STAT レジスタの ACBO ビットにて参照可能です。

コンペアマッチイベント発生時、CM0 レジスタ、および CM1 レジスタを更新する場合は、いくつかの制限があります。ACB ビット(bit2～bit4)にて設定される各コンペアモードの CM0 レジスタ、および CM1 レジスタの更新条件は次の表 6-8 のとおりです。

表 6-8 SOMC ACB ビット(bit2～bit4)の各モードでの CM0 レジスタおよび CM1 レジスタ更新条件

ACB			CM0 レジスタ、CM1 レジスタの更新条件
bit4	bit3	bit2	
0	0	0	(1) SR0 レジスタにタイムスタンプが転送されている場合、SR0 レジスタを参照する(読み出す)必要があります。 (2) SR1 レジスタにタイムスタンプが転送されている場合、SR1 レジスタを参照する(読み出す)必要があります。
0	0	1	
0	1	0	
0	1	1	
1	0	0	(1) SR0 レジスタにタイムスタンプが転送されている場合、SR0 レジスタを参照する(読み出す)必要があります。 (2) SR1 レジスタにタイムスタンプが転送されている場合、SR1 レジスタを参照する(読み出す)必要があります。 (3) CCU0 でコンペアマッチが発生してから CCU1 でコンペアマッチが発生するまでの間は、CM0 レジスタ、CM1 レジスタを更新できません。
1	0	1	
1	1	0	

このチャネルモードでは、CCU0 にてコンペアマッチイベントが発生した場合は、ATOM[i]_CCU0TC[x]_IRQ が、CCU1 でコンペアマッチイベントが発生した場合は、ATOM[i]_CCU1TC[x]_IRQ が通知されます。

6.2.3 ATOM Signal Output Mode PWM (SOMP)

PWM 信号出力モードです。

このチャネルモードではデューティの終了(CN0≧CM1)時に ATOM[i]_CCU1TC[x]_IRQ、周期の終了(CN0≧CM0)時に ATOM[i]_CCU0TC[x]_IRQ が通知されます(詳細は 6.4.1 を参照してください)。

表 6-9 SOMP GTM[g].ATOM[i]_CH[x]_CTRL レジスタ設定項目

ビット名	説明
SOMB	チャネルモードを SOMB とするかを設定します。 0：チャネルモードを SOMB 以外とします。 チャネルモードは MODE にて指定します。
OSM	ワンショットモードを設定します。 0：無効。連続出力します。 1：有効。PWM 信号を 1 周期出力後、停止します。
TRIGOUT EXTTRIGOUT	詳細は 6. 3 信号出力トリガを参照してください。
EXT_TRIG OSTM_TRIG RESET_CCU0	表 6-11 SOMP モード動作（ワンショットモード無効）、 表 6-12 ワンショットモード動作を参照してください。
UPMODE	CN0 のカウント方向を設定します。 00b：アップカウント 01b：アップ／ダウンカウント 谷(CN0=0)時、CM0 と CM1 更新 10b：アップ／ダウンカウント 山(CN0=CM0)時、CM0 と CM1 更新 11b：アップ／ダウンカウント 谷(CN0=0)または山(CN0=CM0)時、CM0 と CM1 更新
ECLK_SRC	CLK_SRC_SR を参照してください。
CLK_SRC_SR	カウントクロックソースを設定します。 ● ECLK_SRC=0 の場合 0～7：CMU_CLK チャネル番号 ● ECLK_SRC=1 の場合 000b: CMU_CLK0 001b: CMU_CLK1 010b: CMU_CLK2 011b: Reserved 100b: Clock stopped 101b: TRIG[x–1] 110b: TIM_EXT_CAPTURE[x] 111b: CMU_CLK7

	信号出力中に本ビットを設定した場合、PWM の周期終了時または強制更新により、設定した CMU_CLK チャネルに切り替わります。 PWM の周期終了時に更新する場合、GTM[g]ATOM[i]AGCGLBCTRL の UPEN_CTRL[x]ビットを設定する必要があります。 カウントクロックソースは ARU 転送無効(ARU_EN=0)の場合、CLK_SRC_SR の値によって更新され、ARU 転送有効(ARU_EN=1)の場合、ARU 経由で受信した CLK_SRC の値によって更新されます。
SL	PWM 出力開始時(デューティ部)の信号レベルを設定します。 0：出力開始時信号(デューティ部)を Low とします。 1：出力開始時信号(デューティ部)を High とします。
BITREV	Pulse Count Modulation(PCM)モードの有効／無効を設定します。 0：PCM モードを無効にします。 1：PCM モードを有効にします。 PCM モードを有効にすることが可能なチャネルはマイコンにより異なります。(※1)
MODE	チャネルモードを設定します。 2：ATOM Signal Output Mode PWM (SOMP)

※1：GTM-IP 358 では ATOM チャネル 1,3,5,7 のみ有効です。

表 6-10 SOMP GTM[g].ATOM[i]_CH[x]_CTRL レジスタ以外のレジスタ設定項目

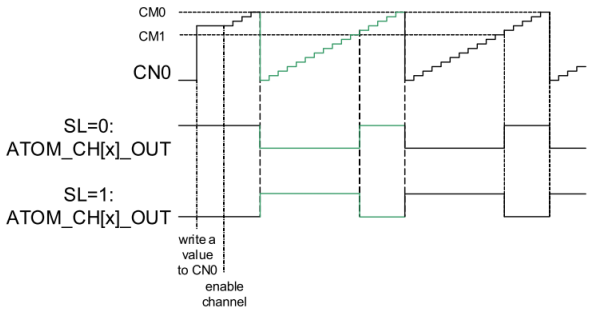
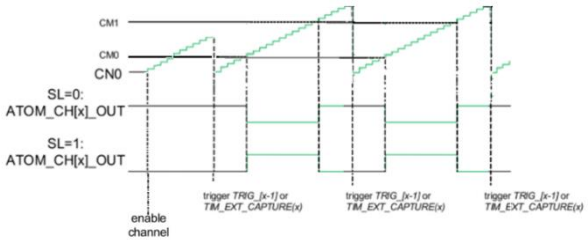
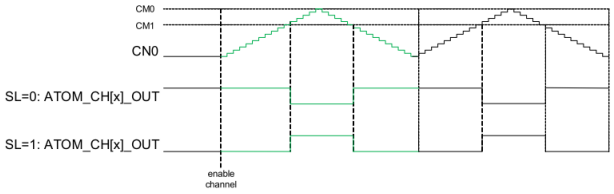
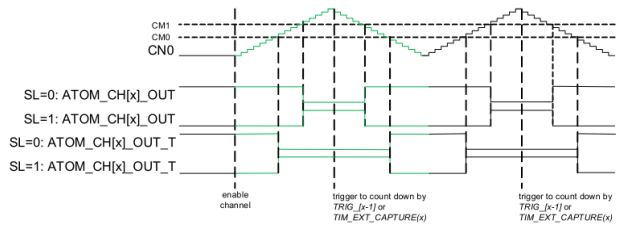
レジスタ名	マイコン設定レジスタ	説明
CN0	GTM[g].ATOM[i]_CH[x]_CN0	インクリメントカウンタの初期値を設定します。
CM0	GTM[g].ATOM[i]_CH[x]_CM0	PWM 周期クロック数を指定します。
CM1	GTM[g].ATOM[i]_CH[x]_CM1	PWM デューティクロック数を指定します。
SR0	GTM[g].ATOM[i]_CH[x]_SR0	CM0 レジスタのシャドウレジスタです。
SR1	GTM[g].ATOM[i]_CH[x]_SR1	CM1 レジスタのシャドウレジスタです。

ワンショットモードを無効にした場合の CN0 カウンタの動作は以下のとおりです。ワンショットモードを有効にした場合の動作については、6.2.3.2 を参照してください。

(1) チャネル有効直後
チャネルの有効直後に、CN0 カウンタは現在の値からカウント動作を実行します。比較対象は CM0 レジスタのカウンタ値で、CN0 カウンタ値≧CM0 レジスタ値の場合に CN0 カウンタは CCU0 によって 0 リセットされ、出力信号は SL ビットで設定した状態となります。
チャネル有効直後の CN0 カウンタの設定値は、信号出力開始までのカウンタとして使用可能です。CN0 レジスタに 0 を設定した場合は PWM 1 周期分信号出力を待つこととなり、CN0 レジスタに CM0 レジスタと同じ値を設定した場合にチャネル有効直後に PWM 信号を出力します。

(2) 信号出力中
CM1 レジスタ、および CM0 レジスタと比較を行い、CM1 レジスタとマッチした場合には信号出力を反転した状態(SL ビットに指定した状態と逆)となり、CM0 レジスタとマッチした場合には信号出力を SL ビットに指定した状態で出力します。

表 6-11 SOMP モード動作（ワンショットモード無効）

UDMODE	OSM	OSTM_TRIG	RST_CCU0	動作
0	0	0	0	
			1	
0 以外			0	
			1	

6.2.3.1 PWM 信号データ更新方法

PWM 信号送出中のデータ更新方法として次の方法が存在します。

(1) 非同期更新

CM0 レジスタ、CM1 レジスタ、CLK_SRC レジスタを、CN0 カウンタがカウント継続中に直接更新します。
この更新方法は、マイコンより直接 CM0 レジスタや CM1 レジスタを書き換えるか、SR0 レジスタおよび SR1 レジスタへ値を設定した状態で強制更新を行った場合に行われます。強制更新については 6.1.3 を参照してください。

a) 現在の信号状態がデューティ期間中(更新前 CM1 レジスタ値 ＞CN0 カウンタ)の場合、出力中の PWM 信号のデューティ期間に即座に反映されます。
b) ただし「a)」の条件で、更新後 CM1 レジスタ値＜CN0 カウンタの場合、即座にデューティが終了します。そして次の PWM 周期から更新後 CM1 レジスタ値が反映されます。
c) 現在の信号状態がデューティ期間終了後(更新前 CM1 レジスタ値 ＜CN0 カウンタ)の場合、次の PWM 周期から更新後 CM1 レジスタ値が反映されます。

この更新方法において、信号出力にスパイクノイズが出力されないことは保証されています。

(2) 同期更新

AGC にて更新を有効に設定している場合、SR0 レジスタにて CM0 レジスタを、SR1 レジスタにて CM1 レジスタを、CLK_SRC_SR レジスタにて CLK_SRC レジスタを更新します。更新タイミングは PWM 周期終了時(CN0 カウンタ値≧CM1 レジスタの条件が成立し、CN0 カウンタがリセットされた際)に反映されます。AGC の更新設定についての詳細は 6.1.3 を参照してください。

6.2.3.2 ワンショットモード

PWM 信号を単発(1 周期分)で出力するモードです。

本動作モードでは、チャネルを有効にしても CN0 カウンタのカウントは実行されず、そのままでは信号出力は開始しません。チャネルを有効にした後に、マイコンより CN0 カウンタへ値を設定する必要があります。

CN0 レジスタへ値を設定すると CN0 カウンタがカウントを開始します。CN0 カウンタ値≧CM0 レジスタ値の場合、CCU0 によってリセットされて 0 に戻り、出力信号は SL ビットで設定した状態になります。この時 SR0 レジスタと SR1 レジスタの内容で CM0 レジスタおよび CM1 レジスタの更新を避けたい場合、AGC の更新を無効にしておきます。

CN0 カウンタ値≧CM1 レジスタ値の場合、出力信号を反転します。そして CN0 カウンタ値≧CM0 レジスタ値となった場合、CN0 カウンタはカウントを停止し、信号出力を停止します。

信号出力が停止しても、チャネルは有効なままですので、再度 CN0 レジスタに値を設定することにより、信号出力が行われます。

信号出力中(すなわち CN0 カウンタがカウント中)に CN0 カウンタを書き換えた場合、次のようにフェーズ毎に動作が変わります。

フェーズ 1：PWM 信号出力前(信号出力が SL ビットで設定された状態に変更される前)

CN0 カウンタを設定した直後で CN0 カウンタ値＜CM0 レジスタ設定値の場合です。

書き換え後、CN0 カウンタが CM0 レジスタ値以上になるまで PWM 信号の出力が遅延します。

フェーズ 2：PWM デューティ期間中(信号出力が SL ビットで設定された状態で出力中)

CN0 カウンタ値≧CM0 レジスタ設定値が成立し CN0 カウンタが CCU0 によって 0 で初期化され、CM1 レジスタの設定値と比較中の場合です。

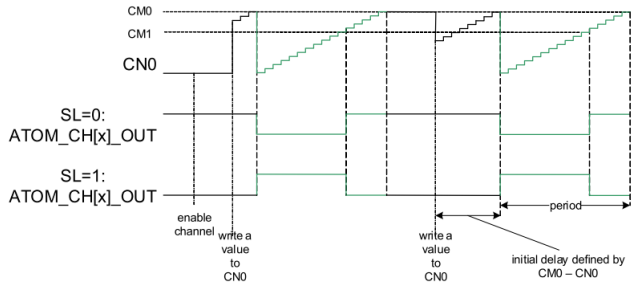
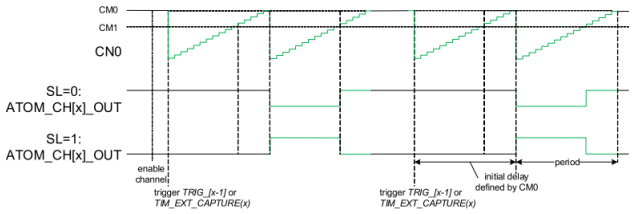
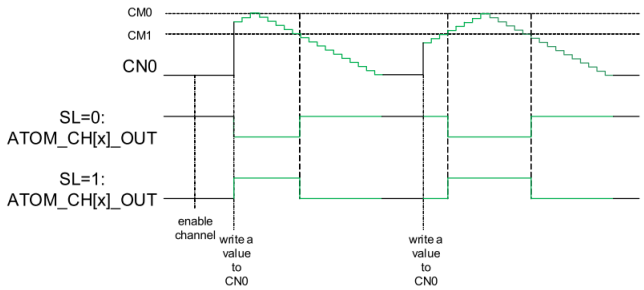
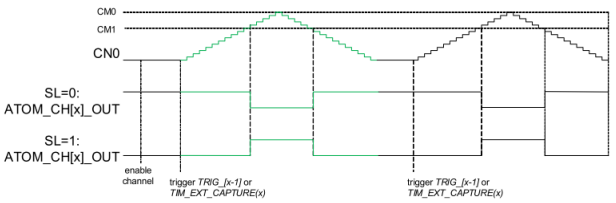
書き換え後、CN0 カウンタが CM1 レジスタ値になるまでデューティが継続します。

フェーズ 3：PWM デューティ期間終了後から周期終了前
(信号出力が SL ビットで設定された状態の逆状態で出力中)

PWM デューティ期間後より PWM の出力終了近くの場合です。

書き換え後、PWM 信号は再出力となります。CN0 カウンタが CM0 レジスタ値以上になったら、PWM 信号が出力されます。

表 6-12 ワンショットモード動作

UDMODE	OSM	OSTM_TRIG	RST_CCU0	動作
0	1	0	0	
		1		
0 以外		0		
		1		

6.2.3.3 PCM モード(Pulse Count Modulation)

ATOM の一部チャネルではパルスカウント変調による出力が可能です。PWM ではパルス幅のデューティ比を設定し、信号の ON 時間と OFF 時間を決定して信号を出力しますが、パルスカウント変調では、デューティ比を元に、入力クロックと同期したパルス数として分散して出力します。

BITREV ビットを設定することで PCM モードとなりパルスカウント変調による出力が可能になります。

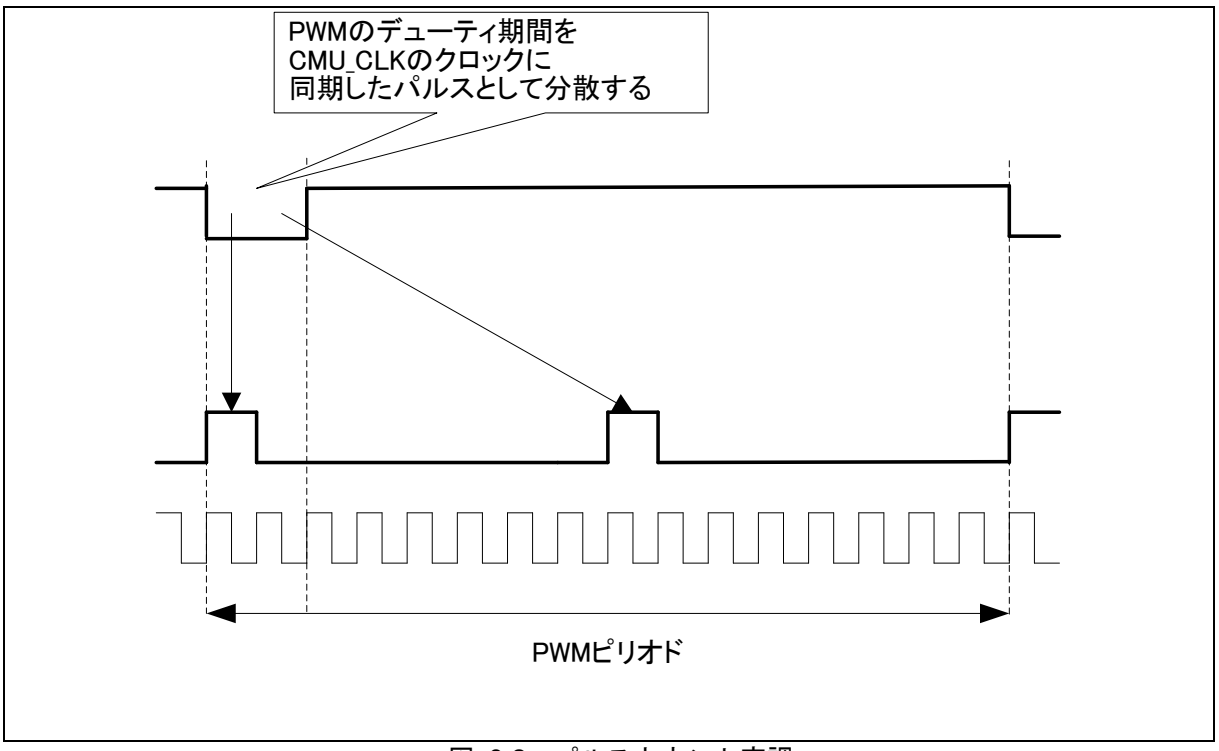


図 6-3 パルスカウント変調

本動作モードでは、CN0 カウンタのカウントを実行後、CN0 カウンタの MSB より LSB のビット配列を LSB より MSB へ入れ替えます。例えば、CN0 カウンタが$(000001)_{16}$の場合は$(800000)_{16}$となり、CN0 カウンタが$(0x000004)_{16}$の場合は$(200000)_{16}$となります。

入れ替えた CN0 カウンタ値 ≧ CM1 レジスタのデューティ比であれば SL ビットの反転で信号を出力し、そうでなければ SL ビットの設定で信号を出力します。

本動作モードでは CM0 レジスタは通常、設定可能な最大値$((FFFFFF)_{16})$を設定しますが、CM1 レジスタのデューティ比の更新時間を低減するため、最大値よりも小さい値を CM0 レジスタへ設定することも可能です。この場合、基本的に最大値に 1 を足した$(1000000)_{16}$を右方向へビットシフトした値$((800000)_{16}$、$(400000)_{16}$、$(20000)_{16}$等)が使用可能です。

ただし、CM0 レジスタに$(FFFFFF)_{16}$以外を設定する場合、CM1 レジスタに設定されるデューティは次のように計算する必要があります。

(1) PWM 周期クロック数として、CM0 レジスタへ設定する値を算出します。
(2) (1)で算出した値の$(1000000)_{16}$に対するビットシフト数を算出します。
例えば、PWM 周期クロック数として$(10000)_{16}$を算出した場合は、$(1000000)_{16}$に対して 8 ビットシフトとなります。
(3) PWM デューティクロック数を算出します。
例えば、PWM 周期クロック数が$(10000)_{16}$、デューティ比 50%の場合は、PWM デューティクロック数は$(8000)_{16}$となります。
(4) (3)で算出した値に対して、(2)で算出したビットシフト数分左シフトし、CM1 レジスタへ設定する値を算出します。
例えば、PWM デューティクロック数が$(8000)_{16}$、ビットシフト数が 8 の場合は、CM1 レジスタに設定する値は$(800000)_{16}$となります。

本動作モードでは、出力されるパルス毎に ATOM[i]_CCU0TC[x]_IRQ が通知されます。

パルスカウント変調にて CM0 レジスタ、CM1 レジスタ、および CN0 カウンタを 4 ビットサイズで仮定した場合、ビット列を入れ替えた CN0 カウンタによって、次の表 6-13 のようにパルスが均等に分散されます。また出力されるパルス数より「ディーティクロック数 = 出力するパルス数」という見方も可能です。

表 6-13 パルスカウント変調でのパルス分散状況
(CM0 レジスタ=15、SL ビット=0、各レジスタは 4 ビット長と仮定した場合)

CN0 カウンタ (bitreversed)	CM1 レジスタ設定値(デューティクロック数)														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
4	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
2	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
6	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
5	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
3	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
11	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
7	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.2.4 ATOM Signal Output Mode Serial (SOMS)

CM1 レジスタを使用してシリアル信号を出力します。設定された CMU_CLK[x]の周波数によってシリアル出力のレートが決まります。指定ビット数送出後、ATOM[i]_CCU0TC0_IRQ が通知されます。

ATOM チャネルを複数チャネル使用すれば、SPI 等のシリアルインターフェースも実現可能です。

表 6-14 SOMS GTM[g].ATOM[i]_CH[x]_CTRL レジスタ設定項目

ビット名	説明
SOMB	チャネルモードを SOMB とするかを設定します。 0：チャネルモードを SOMB 以外とします。 チャネルモードは MODE にて指定します。
OSM	ワンショットモードを設定します。 0：無効 連続出力します。 1：有効 指定ビット数送出後、停止します。
ECLK_SRC	CLK_SRC_SR を参照してください。
CLK_SRC_SR	カウントクロックソースを設定します。 ● ECLK_SRC=0 の場合 0～7：CMU_CLK チャネル番号 ● ECLK_SRC=1 の場合 000b: CMU_CLK0 001b: CMU_CLK1 010b: CMU_CLK2 011b: Reserved 100b: Clock stopped 101b: TRIG[x–1] 110b: TIM_EXT_CAPTURE[x] 111b: CMU_CLK7
SL	シリアル出力開始前の信号レベルを設定します。 0：信号を High とします。 1：信号を Low とします。
DSO	CM1 レジスタのモードを設定します。 0: CM1 は 24 ビットシフトレジスタとして使用します。 1: CM1 は 2 つの 12 ビットシフトレジスタに分割します。
ACB(bit0)	CM1 レジスタのシフト方向を設定します。 0：右シフト(LSB より MSB の順番で出力) 1：左シフト(MSB より LSB の順番で出力)
MODE	チャネルモードを設定します。 3：ATOM Signal Output Mode Serial (SOMS)

表 6-15 SOMS 他のレジスタ設定項目

レジスタ名	マイコン設定レジスタ	説明
CN0	GTM[g].ATOM[i]_CH[x]_CN0	シフトカウンタです。 ワンショットモードの場合に、チャネル停止後、0 を設定すると、再度ビットデータの出力を開始します。
CM0	GTM[g].ATOM[i]_CH[x]_CM0	シフト回数。実際に出力するビット数-1 を設定します。
CM1	GTM[g].ATOM[i]_CH[x]_CM1	出力するビットデータを設定します。
SR0	GTM[g].ATOM[i]_CH[x]_SR0	CM0 レジスタのシャドウレジスタです。
SR1	GTM[g].ATOM[i]_CH[x]_SR1	CM1 レジスタのシャドウレジスタです。

表 6-16 SOMS 動作一覧

OSM	AGC による更新(※1)	動作
0	無効	(1) CM1 レジスタの内容で、CM0 レジスタで指定したビット数分ビットデータを出力します。 (2) (1)を繰り返します。
1	無効	(1) CM1 レジスタの内容で、CM0 レジスタで指定したビット数分ビットデータを出力します。 (2) 停止時に CN0 カウンタへ 0 を設定した場合、CM1 レジスタの内容で、CM0 レジスタで指定したビット数分ビットデータを出力します。 (3) (2)を繰り返します。
0	有効	(1) SR0 レジスタの内容が CM0 レジスタへ、SR1 レジスタの内容が CM1 レジスタへ転送されます。 (2) CM1 レジスタの内容で、CM0 レジスタで指定したビット数分ビットデータを出力します。 (3) (1)、(2)の内容を繰り返します。
1	有効	(1) SR0 レジスタの内容が CM0 レジスタへ、SR1 レジスタの内容が CM1 レジスタへ転送されます。 (2) CM1 レジスタの内容で、CM0 レジスタで指定したビット数分ビットデータを出力します。 (3) CN0 カウンタへ 0 を設定した場合、SR0 レジスタの内容が CM0 レジスタへ、SR1 レジスタの内容が CM1 レジスタへ転送されます。 (4) (2)、(3)の内容を繰り返します。

※1：GTM[g].ATOM[i]_AGC_GLB_CTRL レジスタの UPEN_CTRL[x]ビットの設定です。

6.2.5 ATOM Signal Output Mode Buffered Compare (SOMB)

TBU タイムスタンプ値と、コンペアマッチユニット(CCU0 および CCU1)のコンペアレジスタ(CM0 レジスタ、および CM1 レジスタ)に指定されたタイムスタンプ値とコンペアマッチした場合、信号出力を制御します。ATOM Signal Output Mode Compare (SOMC)と異なり、コンペアマッチ発生時のタイムスタンプ値はシャドウレジスタ(SR0 レジスタ、および SR1 レジスタ)へ設定されません。

このチャネルモードでは、CCU0 にてコンペアマッチが発生した場合には ATOM[i]_CCU0TC[x]_IRQ、CCU1 にてコンペアマッチが発生した場合には ATOM[i]_CCU1TC[x]_IRQ が通知されます。詳細は 6.4.1 を参照してください。

表 6-17 SOMB GTM[g].ATOM[i]_CH[x]_CTRL レジスタ設定項目

ビット名	説明
SOMB	チャネルモードを SOMB とするかを設定します。 1：チャネルモードを SOMB とします。
TRIGOUT	詳細は 6.3 信号出力トリガを参照ください。
EXTTRIGOUT	
SL	初期出力信号を設定します。 0：初期出力信号を Low とします。 1：初期出力信号を High とします。
CMP_CTRL	CM0 レジスタおよび CM1 レジスタの比較方法を選択します。 0：TBU のタイムスタンプ値 ≧ CM0 または CM1 1：TBU のタイムスタンプ値 ≦ CM0 または CM1
ACB(bit2～bit4)	コンペア動作モードを設定します。 詳細は表 6-19 を参照してください。
ACB(bit0～bit1)	コンペアマッチ発生時の出力信号を設定します。 詳細は表 6-18 を参照してください。
TB12_SEL	CM1 レジスタと比較するタイムスタンプを設定します。 0: TBU チャネル 1 のタイムスタンプ値 1: TBU チャネル 2 のタイムスタンプ値
MODE	SOMB モードでは使用しません。

GTM[g].ATOM[i]_CH[x]_CTRL レジスタの ACB ビット(bit0～bit1)へコンペアマッチイベント発生時の信号の出力状態を設定します。信号出力は以下の**表 6-18** のように SL ビットの影響を受けます。

表 6-18 SOMB コンペアマッチイベント発生時の出力信号

SL	ACB		信号の出力状態
	bit1	bit0	
0	0	0	信号出力は変化しません。
0	0	1	信号出力を High にします。
0	1	0	信号出力を Low にします。
0	1	1	信号出力を反転します。
1	0	0	信号出力は変化しません。
1	0	1	信号出力を Low にします。
1	1	0	信号出力を High にします。
1	1	1	信号出力を反転します。

ACB ビット(bit2～bit4)にて設定可能なコンペアモードは次の表のとおりです。

表 6-19 SOMB コンペア動作モード

ACB			CCU0, CCU1 の制御方法
bit4	bit3	bit2	
0	0	0	未使用です。
0	0	1	未使用です。
0	1	0	CCU0 のみで比較を行います。 コンペアマッチした場合、信号レベルは表 6-18 に従い変化します。 CCU0 は TBU チャネル 0 のタイムスタンプと比較します。
0	1	1	CCU1 のみで比較を行います。 コンペアマッチした場合、信号レベルは表 6-18 に従い変化します。 CCU1 は TBU チャネル 1 または 2 のタイムスタンプと比較します。
1	0	0	CCU0 で比較後、さらに CCU1 で比較します。 CCU0 でコンペアマッチした場合、信号レベルは表 6-18 に従い変化します。続けて CCU1 でコンペアマッチした場合、信号レベルを反転します。 CCU0、CCU1 ともに、TBU チャネル 0 のタイムスタンプと比較します。
1	0	1	CCU0 で比較後、さらに CCU1 で比較します。 CCU0 でコンペアマッチした場合、信号レベルは表 6-18 に従い変化します。続けて CCU1 でコンペアマッチした場合、信号レベルを反転します。 CCU0、CCU1 ともに、TBU チャネル 1 または 2 のタイムスタンプと比較します。
1	1	0	CCU0 で比較後、さらに CCU1 で比較します。 CCU0 でコンペアマッチしても信号は変化しません。続けて CCU1 でコンペアマッチした場合、信号レベルは表 6-18 に従い変化します。 CCU0 は TBU チャネル 0 のタイムスタンプと、CCU1 は TBU チャネル 1 または 2 のタイムスタンプと比較します。
1	1	1	比較動作をキャンセルします。

比較するタイムスタンプ値を設定するため、以下のレジスタを設定します。

表 6-20 SOMB GTM[g].ATOM[i]_CH[x]_CTRL レジスタ以外のレジスタ設定項目

レジスタ名	マイコン設定レジスタ	説明
CM0	GTM[g].ATOM[i]_CH[x]_CM0	CCU0 の比較タイムスタンプ値です。
CM1	GTM[g].ATOM[i]_CH [x]_CM1	CCU1 の比較タイムスタンプ値です。
SR0	GTM[g].ATOM[i]_CH[x]_SR0	CM0 レジスタのシャドウレジスタです。
SR1	GTM[g].ATOM[i]_CH[x]_SR1	CM1 レジスタのシャドウレジスタです。

本チャネルモードは、他のチャネルモードとは異なり、マイコンの制御時に GTM[g].ATOM[i]_CH[x]_STAT レジスタの ACBI ビットが、比較時の動作モードや信号出力制御の実行レジスタの役割を果たします。したがって、GTM[g].ATOM[i]_CH[x]_STAT レジスタの ACBI ビットを更新する場合、GTM[g].ATOM[i]_CH[x]_CTRL レジスタの ACB ビット(bit4, bit3, bit2)へ比較時の動作モードを、ACB ビット(bit1, bit0)へ比較時の信号出力状態を設定し、AGC による更新を有効にして更新する必要があります。AGC による更新の詳細は 6.1.3 を参照してください。

ACBI ビット、CM0 レジスタ、CM1 レジスタの反映後、即座に比較を実行します。

6.3 信号出力トリガ

トリガは独立した信号線で、トリガを使用した場合には、他のチャネルや TIM サブモジュール、ATOM サブモジュールと連携して信号出力開始タイミングを制御することが可能となります。

表 6-21 選択可能な信号出力開始トリガ信号一覧

信号名	説明
ATOM[i]_TRIG_[x]	ATOM 内部でトリガを発生させます。 本トリガを使用する場合、GTM[g].ATOM[i]_AGC_INT_TRIG レジスタの INT_TRIG[x] ビットを設定する必要があります。 本トリガのソースとなる信号については、表 6-22 を参照してください。
TBU_CH[x]	TBU のタイムスタンプ値が指定した値になった場合にトリガを発生させます。 ATOM サブモジュールの全チャネルに影響します。 タイムスタンプ値は GTM[g].ATOM[i]_AGC_ACT_TB レジスタにて設定を行います。
HOST_TRIG	CPU から任意のタイミングでトリガを発生させます。 ATOM サブモジュールの全チャネルに影響します。 GTM[g].ATOM[i]_AGC_GLB_CTRL レジスタの HOST_TRIG ビットにて設定を行います。

ATOM サブモジュールのチャネルより出力可能なトリガ信号(ATOM[i]_TRIG_[x])の供給元として次の信号のいずれかを選択できます。

表 6-22 ATOM サブモジュールのチャネルトリガソース信号一覧

トリガ信号ソース名	説明
TIM[i]_EXT_CAPTURE[x]	TIM サブモジュールの同サブモジュール番号の同チャネル番号のキャプチャトリガです。 キャプチャトリガについては、5.5 を参照してください。
ATOM[i]_TRIG_[x-1]	対象チャネルの一つ前のチャネルトリガです。 対象チャネル番号が 0 の場合、1 つ前の ATOM サブモジュールの最大チャネル番号のチャネルトリガを使用します。 また、ATOM0 サブモジュールのチャネル 0 は ATOM0 サブモジュールの最大チャネル番号のチャネル信号を使用します。
TRIG_CCU0	本チャネルの CCU0 のコンペアマッチイベントです。

チャネルトリガソースは、GTM[g].ATOM[i]_CH[x]_CTRL レジスタの EXTTRIGOUT ビット、TRIGOUT ビットにより選択します。各チャネルトリガソースの選択方法は以下のようになります。

表 6-23 チャネルトリガソース選択方法

トリガ信号ソース名	EXTTRIGOUT	TRIGOUT
TIM[i]_EXT_CAPTURE[x]	1	0
ATOM[i]_TRIG_[x-1]	0	0
TRIG_CCU0	-	1

6.4 出力結果の表示

6.4.1 割り込み通知

ATOM は、CCU0 や CCU1 にてコンペアマッチが発生すると、ATOM[i]_CCU0TC[x]_IRQ または ATOM[i]_CCU1TC[x]_IRQ の割り込みが発生します。

これらの割り込みを有効とする場合、GTM[g].ATOM[i]_CH[x]_IRQ_EN レジスタを設定します。また、対応する割り込みが発生後、GTM[g].ATOM[i]_CH[x]_IRQ_NOTIFY レジスタの対応する割り込みのビットを設定することで、割り込み要因をクリアすることが出来ます。

6.5 ARU との転送

ATOM では ARU にて他のサブモジュール(おもに MCS)とのデータ交換が可能です。

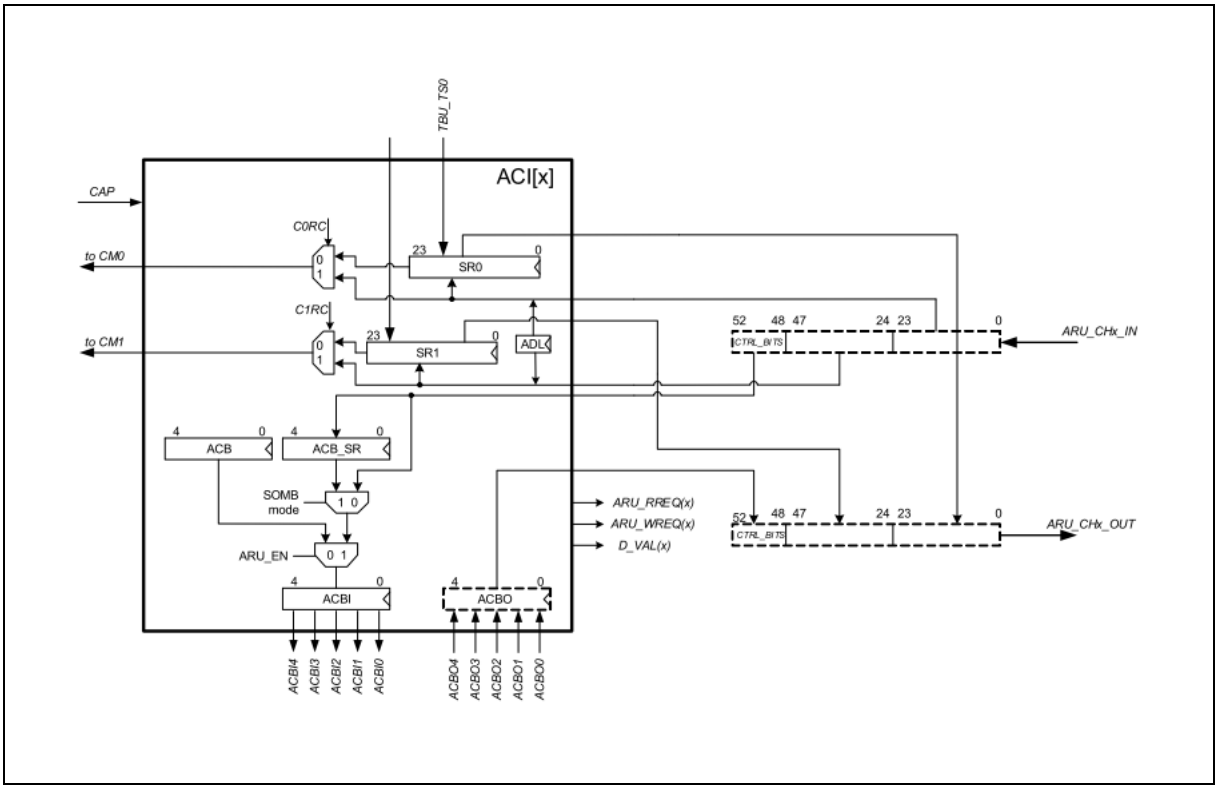


図 6-4 ATOM ACI ブロック

転送するデータについては各チャネルモードにより異なります。

ARU 転送を有効にするには GTM[g].ATOM[i]_CH[x]_CTRL レジスタの ARU_EN ビットを設定します。

(1) ATOM から ARU
ARU 経由で ATOM のデータを受信する場合は、対応する ARU アドレスを指定してください。
GTM-IP 358 における ATOM の ARU アドレス(ATOM に割り当てられた ARU バッファのインデックス)は次の表のとおりです。

表 6-24 ATOM の各チャネルの ARU アドレス

ATOM チャネル番号	ARU アドレス			
	ATOM0	ATOM1	ATOM2	ATOM3
0	$(11f)_{16}$	$(127)_{16}$	$(12f)_{16}$	$(137)_{16}$
1	$(120)_{16}$	$(128)_{16}$	$(130)_{16}$	$(138)_{16}$
2	$(121)_{16}$	$(129)_{16}$	$(131)_{16}$	$(139)_{16}$
3	$(122)_{16}$	$(12a)_{16}$	$(132)_{16}$	$(13a)_{16}$
4	$(123)_{16}$	$(12b)_{16}$	$(133)_{16}$	$(13b)_{16}$
5	$(124)_{16}$	$(12c)_{16}$	$(134)_{16}$	$(13c)_{16}$
6	$(125)_{16}$	$(12d)_{16}$	$(135)_{16}$	$(13d)_{16}$
7	$(126)_{16}$	$(12e)_{16}$	$(136)_{16}$	$(13e)_{16}$

(2) ARU から ATOM
ARU から ATOM への転送は、次のレジスタへ ARU アドレスを設定します。

GTM[g].ATOM[i]_CH[x]_RDADDR

例として、MCS0 のチャネル 0(ARU アドレス=$(077)_{16}$)のデータを ATOM1 のチャネル 0 で受信する場合、次のように設定します。

GTM0.ATOM1_CH0_RDADDR = 0x77;

本レジスタは、チャネル機能が無効の場合のみ設定可能です。チャネル機能の有効／無効の設定については 6.1.3 を参照してください。

6.5.1 ATOM Signal Output Mode Immediate (SOMI)

GTM[g].ATOM[i]_CH[x]_CTRL レジスタの SL ビットと ARU データの ACB(bit0)にて信号の出力状態を設定します。

表 6-25 SOMI(ARU) 出力信号状態

SL	ACB(bit0)	信号の出力状態
0	0	SL ビットで指定した信号レベルを反転して出力します。 (High レベルを出力します。)
0	1	Low レベルを出力します。
1	0	SL ビットで指定した信号レベルを反転して出力します。 (Low レベルを出力します。)
1	1	High レベルを出力します。

ARU 経由で他のサブモジュールより本チャネルモードにて出力信号を操作する場合、ARU のラウンドトリップ時間によるジッターを考慮してください。

6.5.2 ATOM Signal Output Mode Compare (SOMC)

本チャネルモードでは、他のチャネルモードとは異なり、ARU 転送の場合、CM0 レジスタおよび CM1 レジスタが ARU から受信したデータで直接更新されます。

ARU に関連する GTM[g].ATOM[i]_CH[x]_CTRL レジスタの設定項目は以下のとおりです。

表 6-26 SOMC GTM[g].ATOM[i]_CH[x]_CTRL レジスタ ARU 関連設定項目

ビット名	説明
ABM	ARU ブロッキングモードを設定します。 0：ノンブロッキング 1：ブロッキング
SLA	CCU0 コンペアマッチ発生時の ARU へのタイムスタンプ転送有無を設定します。 コンペア動作モード設定(ARU データの ACB(bit2～bit4))が $(100)_2$、$(101)_2$、$(110)_2$ のいずれかの場合のみ有効です。 0：転送しません。 1：転送します。
WR_REQ	マイコンによる強制更新の有効／無効を設定します。 0：強制更新を無効にします。 1：強制更新を有効にします。 詳細は 6.5.2.1 を参照してください。

ARU と ATOM 間のデータフォーマットは以下のとおりです。

表 6-27 SOMC 受信 ARU データフォーマット

ビット	ビット名	説明
50～52	ACB	コンペア動作モードが設定されます。 詳細は表 6-6 を参照してください。
48～49		コンペアマッチ時の信号出力設定が設定されます。 詳細は表 6-4、表 6-5 を参照してください。
24～47	Data1	CM1 レジスタへ設定するデータが設定されます。
0～23	Data0	CM0 レジスタへ設定するデータが設定されます。

ARU データの ACB(bit2～bit4)へ$(111)_2$が設定された場合、GTM[g].ATOM[i]_CH[x]_RDADDR レジスタの RDADDR1 ビットにて指定された ARU アドレスから ARU データを受信します。ARU の読み出し後、GTM[g].ATOM[i]_CH[x]_RDADDR レジスタの RDADDR0 ビットにて指定されたアドレスに戻ります。

ARU から ATOM への ARU データの ACB の内容は、GTM[g].ATOM[i]_CH[x]_STAT レジスタの DV ビットが設定されている場合のみ、同レジスタの ACBI ビットにて確認が可能です。

表 6-28 SOMC 送信 ARU データフォーマット

ビット	ビット名	説明
52	ACB	CCU1 でコンペアマッチ発生を示します。(※1)
51		CCU0 でコンペアマッチ発生を示します。(※1)
48～50		未使用です。
24～47	Data1	CCU1 マッチ時のタイムスタンプ値を示します。
0～23	Data0	CCU0 マッチ時のタイムスタンプ値を示します。

※1：コンペア動作モード設定が$(100)_2$、$(101)_2$、$(110)_2$のいずれか、かつ GTM[g].ATOM[i]_CH[x]_CTRL レジスタの SLA ビットが 0 の場合、ビット 52 はセットされ、ビット 51 は設定されません。

ATOM から ARU への ARU データの ACB の内容は、GTM[g].ATOM[i]_CH[x]_STAT レジスタの ACBO ビットにて確認が可能です。

6.5.2.1 ARU 転送時のマイコンによる強制設定

ARU 転送の有効時に、SR0 レジスタや SR1 レジスタを経由して、コンペア値の強制設定が可能です。
強制設定を行う際は、AGC の強制更新設定(詳細は 6.1.3 を参照)を設定します。その後、GTM[g].ATOM[i]_CH[x]_CTRL レジスタの WR_REQ が設定された状態で強制更新のトリガが発生すると、SR0 レジスタや SR1 レジスタの内容が CM0 レジスタや CM1 レジスタに転送されます。
SR0 レジスタ、SR1 レジスタに未参照のデータがある場合は、マイコンや ARU にてデータを読み出さない限り、WR_REQ ビットを設定することはできません。また、コンペア動作モード(ACB ビット(bit2～bit4))が$(100)_2$、$(101)_2$、$(110)_2$のいずれかの場合、

- EUPM=0 の場合、CCU0 のコンペアマッチ発生以降は、CCU1 のコンペアマッチ発生後に SR0 レジスタ、および SR1 レジスタを読み出すまで WR_REQ ビットを設定することはできません。
- EUPM=1 の場合、CCU1 のコンペアマッチ発生以降は、SR0 レジスタ、および SR1 レジスタを読み出すまでは WR_REQ ビットを設定することはできません。

更新の成功／失敗は、GTM[g].ATOM[i]_CH[x]_STAT レジスタの WRF ビットより確認が可能です。
また、WR_REQ ビットが設定されている間は、ARU からの受信を行いません。

6.5.3 ATOM Signal Output Mode PWM (SOMP)

SOMP モードでデータ更新を行う場合、データは ARU より ACI サブユニットの SR0 レジスタや SR1 レジスタを経由して、CM0 レジスタおよび CM1 レジスタへ転送されます。このため、AGC による更新を無効にしたままでは SR0 レジスタや SR1 レジスタの内容が転送されません。

データ更新を行う場合、対象チャネルの GTM[g].ATOM[i]_AGC_GLB_CTRL レジスタの UPEN_CTRL[x] ビットを設定し、AGC による更新を有効としてください。

ARU 転送有効時の設定項目および ARU データフォーマットは以下のとおりです。

表 6-29 SOMP GTM[g].ATOM[i]_CH[x]_CTRL レジスタ ARU 関係設定項目

ビット名	説明
CLK_SRC_SR	カウントクロックソースは ARU 転送無効(ARU_EN=0)の場合、CLK_SRC_SR の値によって更新され、ARU 転送有効(ARU_EN=1)の場合、ARU 経由で受信した CLK_SRC の値によって更新されます。
ADL	ARU からのデータ転送対象を設定します。 0：SR0 レジスタと SR1 レジスタへデータを転送します。 1：SR0 レジスタへデータを転送します。 2：SR1 レジスタへデータを転送します。
ARU_EN	0：ARU によるデータ転送無効 1：ARU によるデータ転送有効

表 6-30 SOMP 受信 ARU データフォーマット

ビット	ビット名	説明
50～52	ACB	CMU_CLK チャネル番号が設定されます。
48～49		未使用です。
24～47	Data1	SR1 レジスタへ転送するデータが設定されます。 設定値が PWM ディーティクロック数となります。
0～23	Data0	SR0 レジスタへ転送するデータが設定されます。 設定値が PWM 周期クロック数となります。

ARU 転送有効時のデータ更新方法として、次の方法が存在します。

(1) 非同期更新
ARU 無効時と同様です。詳細は「6.2.3.1PWM 信号データ更新方法」を参照してください。
ただし、同期更新のために ARU からデータを読み込んでいる最中は、非同期更新は行うことが出来ません。

(2) 同期更新
ARU からデータを受信し、CM0 レジスタや CM1 レジスタ、CLK_SRC を更新します。
この更新は PWM 信号の周期期間終了時に行われます。

ARU 転送における注意事項は以下のとおりです。

(1) PWM 周期として、ARU ラウンドトリップ時間の最悪値より小さい値を設定しないでください。
ARU データに設定された PWM 周期が ARU ラウンドトリップ時間の最悪値より小さい場合、次のデータ同期更新が正常に出来ない可能性があります。

(2) ワンショットモードにて、1 周期出力後に停止を選択した場合には、CM0 レジスタや CM1 レジスタが更新されても、PWM 出力は開始しません。出力を開始するには、マイコンにて CN0 に 0 を設定する必要があります。

6.5.4 ATOM Signal Output Mode Serial (SOMS)

SOMS モードでは、ARU のデータは ACI サブユニットの SR0 レジスタや SR1 レジスタを経由して CM0 レジスタ、および CM1 レジスタへ転送されます。このため、AGC による更新を無効にしたままでは SR0 レジスタや SR1 レジスタの内容が転送されません。対象チャネルの GTM[g].ATOM[i]_AGC_GLB_CTRL レジスタの UPEN_CTRL[x]ビットを設定して AGC による更新を有効にしてください。

表 6-31 SOMS 受信 ARU データフォーマット

ビット	ビット名	説明
49～52	ACB	未使用です。
48		CM1 レジスタのシフト方向が設定されます。 0：右シフト(ビットデータは LSB より MSB の順番) 1：左シフト(ビットデータは MSB より LSB の順番)
47～24	Data1	SR1 レジスタへ転送されるデータ(ビットデータ)が設定されます。
23～0	Data0	SR0 レジスタへ転送されるデータ(ビットデータのシフト数)が設定されます。

SOMS モードでは、GTM[g].ATOM[i]_CH[x]_CTRL レジスタの OSM ビットの設定により動作が異なります。

表 6-32 SOMS 動作一覧

OSM	動作
0	CM0 レジスタのシフト回数分ビットデータが出力される(CN0 ≧ CM0)と、SR0 レジスタと SR1 レジスタが更新されているかに関わらず、SR0 レジスタにて CM0 レジスタを、SR1 レジスタにて CM1 レジスタを更新してビットデータを継続して出力します。 したがって、SR0 レジスタ、SR1 レジスタが更新されないまま、CM0 レジスタのシフト回数分ビットデータが出力されると、同じデータを出力することとなります。
1	CM0 レジスタのシフト回数分ビットデータが出力される(CN0 ≧ CM0)と、SR0 レジスタ、SR1 レジスタが更新されている(ARU データを読み込んだ)場合のみ、SR0 レジスタにて CM0 レジスタを、SR1 レジスタにて CM1 レジスタを更新してビットデータを継続して出力します。 SR0 レジスタと SR1 レジスタが更新されていない場合は、出力を停止します。その後、SR0 レジスタ、SR1 レジスタが更新されると、CM0 レジスタと CM1 レジスタを更新し、出力を再開します。

6.5.5 ATOM Signal Output Mode Buffered Compare (SOMB)

本チャネルモードでは、ARU 転送により転送されたデータ内容にて SR0 レジスタ、SR1 レジスタ、および ACB_SR レジスタ(内部レジスタ)を更新します。

ARU に関連する設定項目は以下のとおりです。

表 6-33 SOMB GTM[g].ATOM[i]_CH[x]_CTRL レジスタ ARU 関連設定項目

ビット名	説明
ABM	ARU ブロッキングモードを設定します。 0：ノンブロッキング 1：ブロッキング
WR_REQ	マイコンによる強制更新の有効／無効を設定します。 0：強制更新を無効にします。 1：強制更新を有効にします。 詳細は 6.5.5.1 を参照してください。

ARU と ATOM 間でのデータフォーマットは以下のとおりです。

表 6-34 SOMC 受信 ARU データフォーマット

ビット		説明
50～52	ACB	コンペア動作モードが設定されます。 詳細は表 6-19 を参照してください。
48～49		コンペアマッチ時の信号出力設定が設定されます。 詳細は表 6-18 を参照してください。
24～47	Data1	CM1 レジスタへ設定するデータが設定されます。
0～23	Data0	CM0 レジスタへ設定するデータが設定されます。

受信 ARU データの Data0 および Data1 は SR0 レジスタおよび SR1 レジスタへ転送されるため、AGC による更新(詳細は 6.1.3 を参照)を有効にしないと CM0 レジスタや CM1 レジスタは更新されません。

GTM[g].ATOM[i]_CH[x]_STAT レジスタの DV ビットが設定されていた場合、現在のコンペア動作モードや信号出力設定値が、同レジスタの ACBI ビットで確認できます。

コンペア動作モードで設定したコンペアが全て終了時に、SR0 レジスタの内容で CM0 レジスタを、SR1 レジスタの内容で CM1 レジスタを、ACB_SR レジスタの内容で ACI の ACBI を更新し、コンペアを再開します。ただし、SR0 レジスタ、および SR1 レジスタが更新されていない(ARU データを受信していない)場合、CCU0／CCU1 は ARU からデータが転送されるまで待ちます。ARU 経由で SR0 レジスタ、SR1 レジスタ、ACB_SR レジスタが更新されると、CM0 レジスタ、CM1 レジスタ、ACB を更新し、コンペアを再開します。

6.5.5.1 ARU 転送時のマイコンによる強制設定

ARU 転送の有効時に、SR0 レジスタや SR1 レジスタを経由して、コンペア値の強制設定が可能です。
強制設定を行う際は、AGC の強制更新設定(詳細は 6.1.3 を参照) を設定します。その後、GTM[g].ATOM[i]_CH[x]_CTRL レジスタの WR_REQ が設定された状態で強制更新のトリガが発生すると、SR0 レジスタや SR1 レジスタの内容が CM0 レジスタや CM1 レジスタに転送されます。
SR0 レジスタ、SR1 レジスタに未参照のデータがある場合は、マイコンや ARU にてデータを読み出さない限り、WR_REQ ビットを設定することはできません。また、コンペア動作モード(ACB ビット(bit2～bit4))が$(100)_2$、$(101)_2$、$(110)_2$のいずれか、かつ EUPM=0 の場合、CCU1 のコンペアマッチ発生までは WR_REQ ビットを設定することはできません。強制更新後、WR_REQ ビットは自動的にリセットされます。
更新の成功／失敗は、GTM[g].ATOM[i]_CH[x]_STAT レジスタの WRF ビットより確認が可能です。
また、WR_REQ ビットが設定されている間は、ARU からの受信を行いません。

6.6 デッドタイム付加機能

DTM(Dead Time Module)は、入力された PWM 信号に対してデッドタイムを付加した信号を生成するサブモジュールです。

DTM サブモジュールのチャネルは 4 チャネルあり、ATOM サブモジュールから出力された信号を入力信号とします。

i を ATOM サブモジュール番号とした場合、ATOM[i]のチャネル番号 0～7 が接続する DTM サブモジュール CDTM[i]_DTM[j]は以下のようになります。

- ATOM[i]のチャネル番号 0～3 が接続する DTM サブモジュールは CDTM[i]_DTM4
- ATOM[i]のチャネル番号 4～7 が接続する DTM サブモジュールは CDTM[i]_DTM5

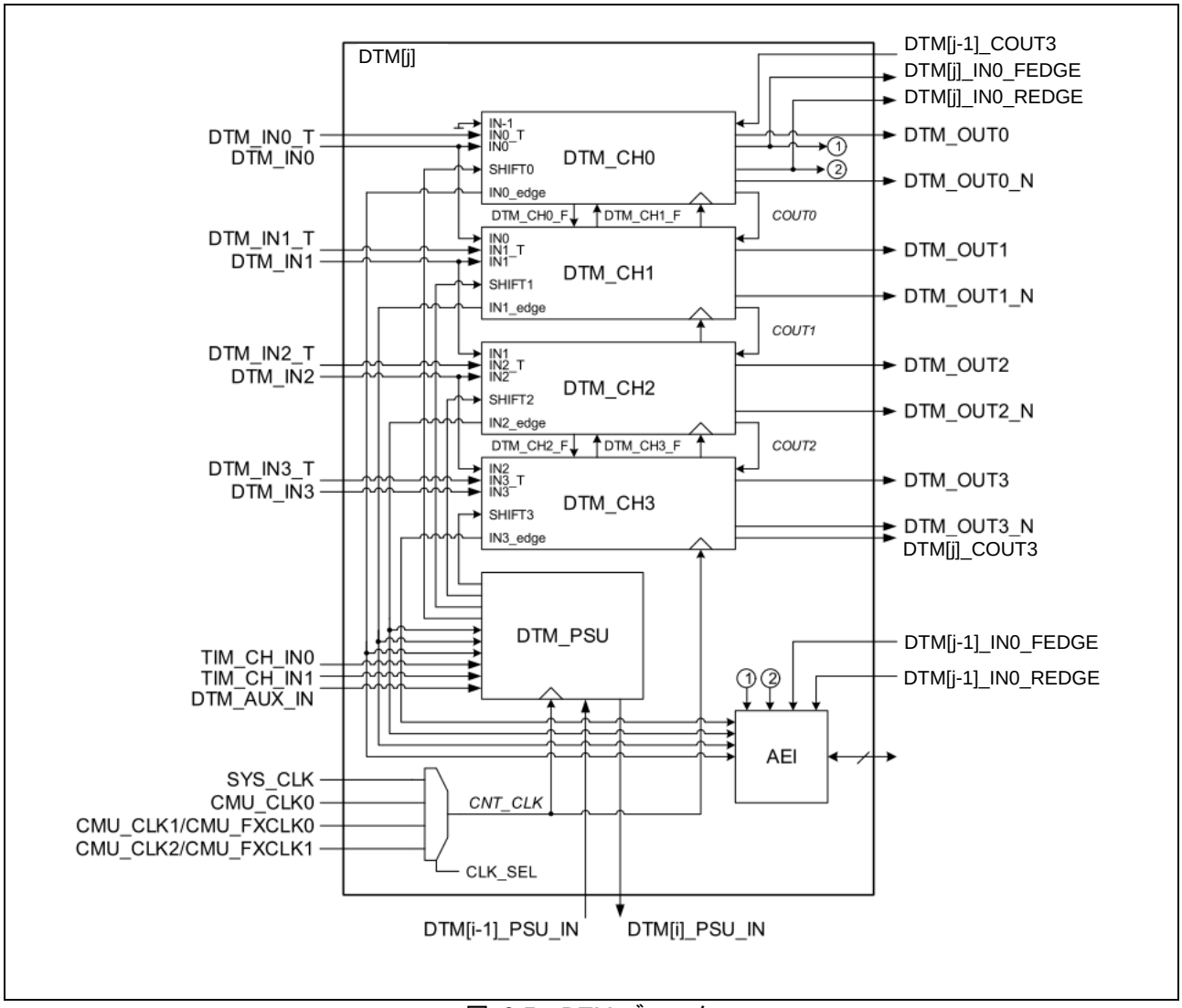


図 6-5 DTM ブロック

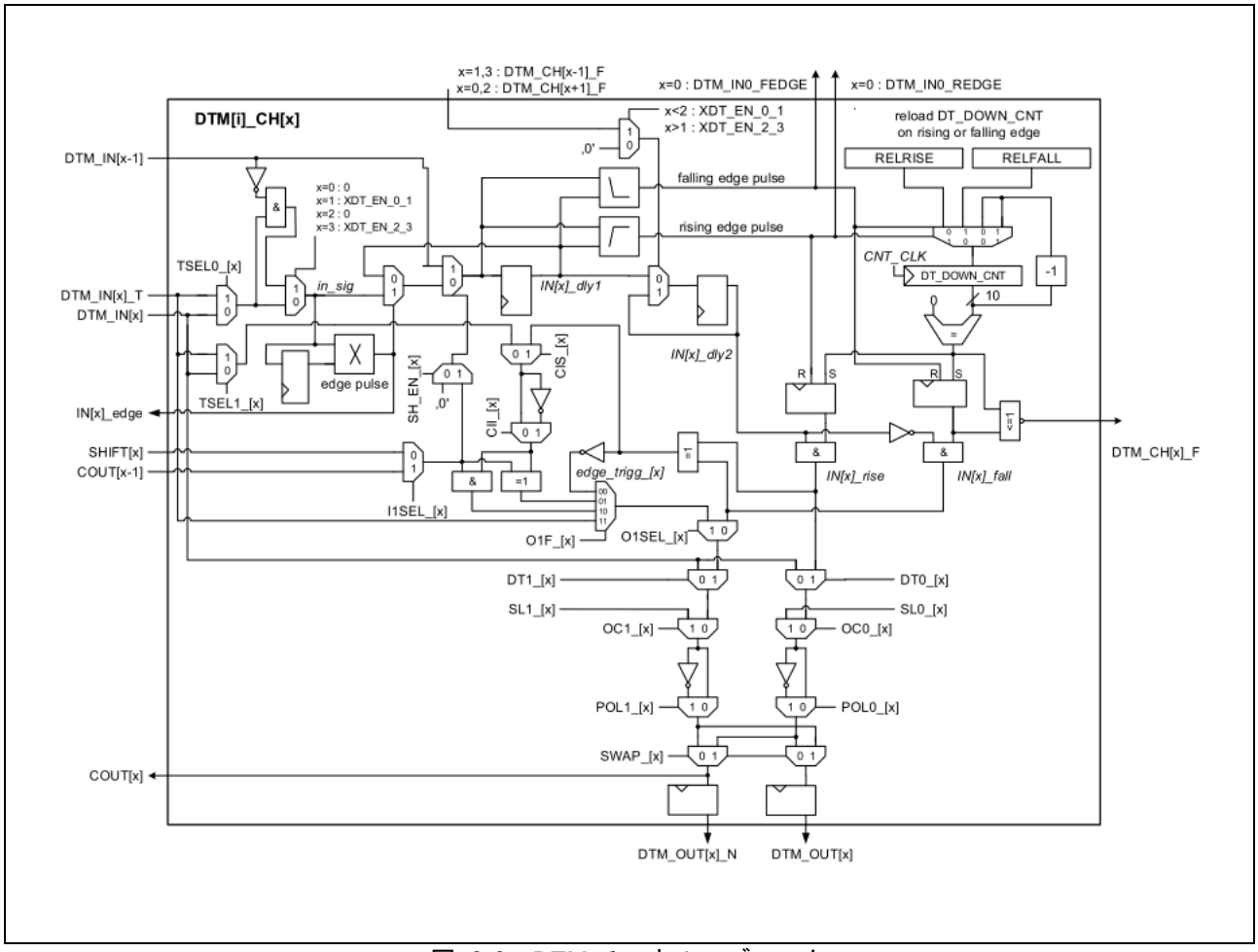


図 6-6 DTM チャネル x ブロック

表 6-35 DTM の主要な信号名一覧

信号名	説明
DTM_IN[x]	DTM サブモジュールに入力される信号を示します。 ATOM サブモジュールの対応するチャネルの出力信号です。
TIM_CH_IN0, TIM_CH_IN1	TIM サブモジュールの FLT(入力フィルタ)の出力です。 どの FLT の出力が TIM_CH_IN0、TIM_CH_IN1 に入力されるかは U2A HW UM を参照ください。
DTM_AUX_IN(※1)	外部からの入力です。 DTM サブモジュールごとに別の信号を入力できます。
SHIFT[x]	フェーズシフト制御を行うための信号です。 DTM サブモジュールごとに 1 チャネル分のみ使用できます。
IN[x]_edge	DTM_IN[x]の立ち上がり時、および立ち下がり時に出力されるパルス信号です。
COUT[x]	出力する逆相信号です。
DTM[j]_PSU_IN	TIM_CH_IN、DTM_AUX_IN のどちらかの信号が DTM_PSU よりそのまま出力されます。
DTM_OUT[x]	出力する正相信号です。
DTM_OUT[x]_N	出力する逆相信号です。

※1：U2A では使用できません。

(1) DTM_CH[x] (DTM Channel [x]) は、入力された信号に対して、デッドタイムの付加、信号の極性設定、正相と逆相信号のスワップ等を行った信号を生成します。

(2) DTM_PSU (DTM Phase Shift Unit) は、DTM[j-1]_PSU_IN、TIM_CH_IN、DTM_AUX_IN のいずれかの入力信号をトリガとしてシフト用の信号を生成します。
DTM_CH[x]は、シフト信号が Low の間は DTM_IN[x]を入力信号としますが、シフト信号が High の間は DTM_IN[x-1]を入力信号とします。シフト用信号は 4 チャネル中 1 チャネルのみにしか入力できません。

(3) デッドタイムを付加する場合、DTM_IN[x]より入力された信号は、DTM_OUT[x]へ出力されるまでに、DTM で使用するクロックの 3 クロック遅延して出力されます。また、デッドタイムを付加しない場合でも、1 クロック分遅延して出力されます。

6.6.1 スタンダードモード

ATOM サブモジュールの出力から正相と逆相の信号を生成し、デッドタイムを付加します。

表 6-36 スタンダードモード GTM[g].CDTM[i]_DTM[j]_CTRL レジスタ設定項目

ビット名	説明
CLK_SEL	入力クロック信号を設定します。 0：SYS_CLK 1：CMU_CLK0 2：CMU_CLK1 3：CMU_CLK2

表 6-37 スタンダードモード GTM[g].CDTM[i]_DTM[j]_CH_CTRL1 レジスタ設定項目

ビット名	説明
SWAP_[x]	正相／逆相入れ替えを設定します。 0：入れ替えません。 1：入れ替えます。 入れ替えた場合、DTM_OUT[x]から逆相信号を、 DTM_OUT[x]_N から正相信号を出力します。

表 6-38 スタンダードモード GTM[g].CDTM[i]_DTM[j]_CH_CTRL2 レジスタ設定項目

ビット名	説明
POL0_[x]	正相信号出力を反転するか設定します。 0：出力信号を反転しません。 1：出力信号を反転します。
OC0_[x]	正相信号出力内容を設定します。 0：DT0_[x]で定義した処理を行った信号を出力します。 1：SL0_[x]で定義した信号状態で出力します。
SL0_[x]	正相固定信号出力状態を設定します。 0：信号レベル Low 1：信号レベル High OC0_[x] = 1 の場合のみ、本ビットの設定は有効です。
DT0_[x]	正相信号デッドタイム付加有無を設定します。 0：デッドタイムを付加しません。 1：デッドタイムを付加します。
POL1_[x]	逆相信号出力を反転するか設定します。 0：出力信号を反転しません。 1：出力信号を反転します。
OC1_[x]	逆相信号出力内容を設定します。 0：DT1_[x]で定義した処理を行った信号を出力します。 1：SL1_[x]で定義した信号状態で出力します。
SL1_[x]	逆相固定信号出力状態を設定します。 0：信号レベル Low 1：信号レベル High OC1_[x] = 1 の場合のみ、本ビットの設定は有効です。
DT1_[x]	逆相信号デッドタイム付加有無を設定します。 0：デッドタイムを付加しません。 1：デッドタイムを付加します。

表 6-39 標準的なデッドタイム GTM[g].CDTM[i]_DTM[j]_CH[x]_DTV レジスタ設定項目

ビット名	説明
RELFALL	立ち下がり時のデッドタイムクロック数を設定します。
RELRISE	立ち上がり時のデッドタイムクロック数を設定します。

デッドタイムを挿入することにより、正相／逆相信号は以下のようになります。

- 入力信号の立ち上がり時

  入力信号の立ち上がりから 3 クロック遅延して逆相信号にて立ち下がりを発生させます。

  逆相信号の立ち下がりから RELRISE で設定したクロック数分遅延して正相信号にて立ち上がりを発生させます。

- 入力信号の立ち下がり時

  入力信号の立ち下がりから 3 クロック遅延して正相信号にて立ち下がりを発生させます。

  正相信号の立ち下がりから RELFALL で設定したクロック数分遅延して逆相信号にて立ち上がりを発生させます。

上記処理は、DT0_[x]と DT1_[x]の設定に関わらず常に行われます。そのため、正相(DT0_[x])と逆相(DT1_[x])のどちらか一方のみでデッドタイム付与を設定した場合でも、デッドタイム付与を設定した相の出力信号は、正相と逆相の両方でデッドタイム付与を設定した場合の信号と同じになります。

6.6.2 クロスチャネルモード

2 つの隣接する DTM チャネルの入力信号を、正相信号と逆相信号とみなし、デッドタイムを付加した信号を生成します。ペアとして指定できる DTM の入力チャネル番号は 2*x と 2*x+1(x=0 以上の数値)です。DTM 入力 DTM_IN[2*x]と DTM_IN[2*x+1] が High レベル の場合、DTM_IN[2*x+1] はチャネル入力時、直ちに Low レベルになります。

表 6-40 クロスチャネルモードで使用する主な信号名一覧

信号名	説明
DTM_IN[2*x]	DTM サブモジュールに入力される正相信号を示します。 ATOM サブモジュールの対応するチャネルの出力信号です。
DTM_IN[2*x+1]	DTM サブモジュールに入力される逆相信号を示します。 ATOM サブモジュールの対応するチャネルの出力信号です。
DTM_OUT[2*x]	出力される正相信号です。
DTM_OUT[2*x+1]	出力される逆相信号です。

GTM[g].CDTM[i]_DTM[j]_CTRL レジスタ設定、および GTM[g].CDTM[i]_DTM[j]_CH_CTRL2 レジスタ設定は、**表 6-38** と同様です。ただし、GTM[g].CDTM[i]_DTM[j]_CH_CTRL2 レジスタはチャネル番号 2*x とチャネル番号 2*x+1 の両方を設定してください。例えば、x=0 の場合、チャネル 0 とチャネル 1 を使用するため、チャネル 0 とチャネル 1 の設定を行ってください。

表 6-41 クロスチャネルモード GTM[g].CDTM[i]_DTM[j]_CH_CTRL1 レジスタ設定項目

ビット名	説明
XDT_EN_[2*x]_[2*x+1]	クロスチャネルモードの設定をします。 0：スタンダードモード。 1：クロスチャネルモード。
SWAP_[x]	正相／逆相入れ替えを設定します。 0：入れ替えません。 1：入れ替えます。 入れ替えた場合、DTM_OUT[x]から逆相信号を、 DTM_OUT[x]_N から正相信号を出力します。

デッドタイムの設定は各入力チャネルの RELFALL ビットのみを設定することで信号の両サイドのデッドタイムを指定します。

表 6-42 クロスチャネルモード GTM[g].CDTM[i]_DTM[j]_CH[2*x]_DTV レジスタ設定項目

ビット名	説明
RELFALL	正相信号の立ち下がり時のデッドタイムクロック数を設定します。
RELRISE	使用しません。

表 6-43 クロスチャネルモード GTM[g].CDTM[i]_DTM[j]_CH[2*x+1]_DTV レジスタ設定項目

ビット名	説明
RELFALL	逆相信号の立ち下がり時のデッドタイムクロック数を設定します。
RELRISE	使用しません。

6.6.3 フェーズシフト制御

選択した DTM チャネルに対して PWM 信号のフェーズシフト制御を行います。

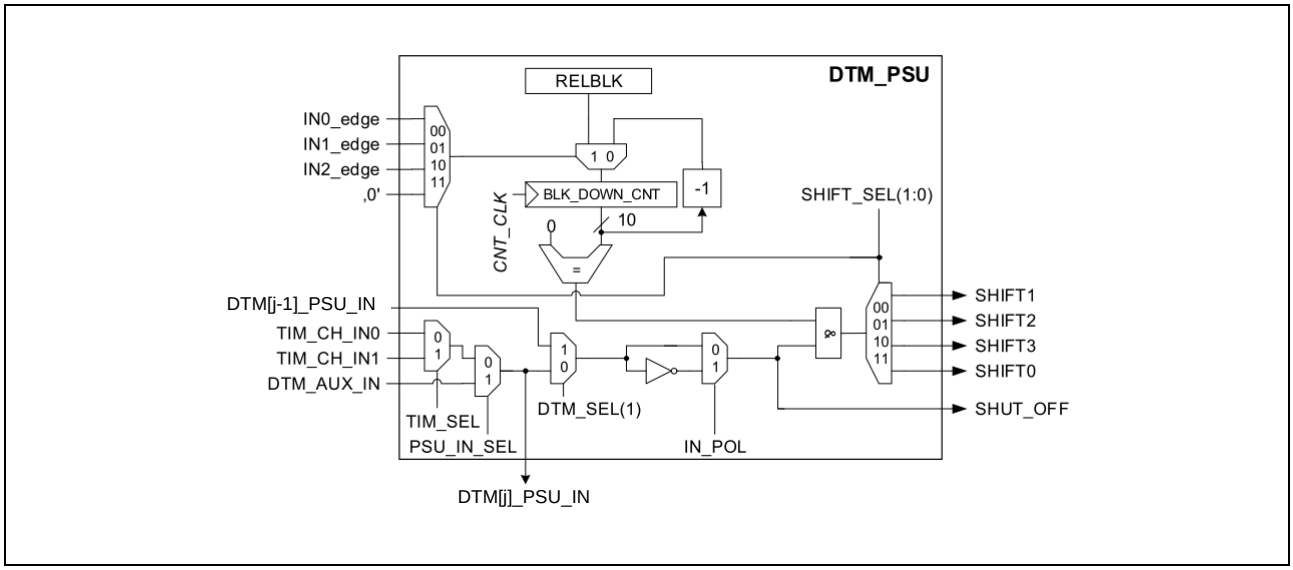


図 6-7 フェーズシフトユニット

表 6-44 フェーズシフト制御時 GTM[g].CDTM[i]_DTM[j]_PS_CTRL レジスタ設定項目

ビット名	説明
SHIFT_SEL	SHIFT[x]信号を接続する DTM チャネル[x]と、DTM_PSU に入力する IN[x-1]_edge 信号を選択します。 SHIFT_SEL / DTM チャネル[x] / IN[x-1]_edge 0 / DTM チャネル 1 / IN0_EDGE 1 / DTM チャネル 2 / IN1_EDGE 2 / DTM チャネル 3 / IN2_EDGE 3 / DTM チャネル 0 / Low 固定
RELBLK	SHIFT[x]信号の出力停止クロック数を設定します。 DTM_PSU に IN[x-1]_edge が入力されてから、設定クロック数の間、SHIFT[x]の出力を停止(Low 固定出力)します。
PSU_IN_SEL, TIM_SEL	DTM_PSU への入力信号(DTM[j]_PSU_IN)を指定します。 PSU_IN_SEL / TIM_SEL / DTM[i]_PSU_IN 0 / 0 / TIM_CH_IN0 0 / 1 / TIM_CH_IN1 1 / - / DTM_AUX_IN※1
IN_POL	PSU への入力信号のレベル反転設定を指定します。 対象となる信号は、GTM[g].CDTM[i]_DTM[j]_CTRL レジスタの DTM_SEL(bit1)で指定した信号となります。 0：入力信号を反転しない 1：入力信号を反転する

※1：DTM_AUX は U2A では Low 固定であるため使用できません。

表 6-45 フェーズシフト制御時 GTM[g].CDTM[i]_DTM[j]_CTRL レジスタ設定項目

ビット名	説明
CLK_SEL	入力クロック信号を設定します。 0：SYS_CLK 1：CMU_CLK0 2：CMU_CLK1 3：CMU_CLK2
DTM_SEL(bit1)	PSU への入力信号を指定します。 0：GTM[g].CDTM[i]_DTM[j]_PS_CTRL レジスタの PSU_IN_SEL で指定した信号 1：DTM[j-1]_PSU_IN 信号(※1)

※1：CDTM[i]_DTM[j]の DTM[j-1]_PSU_IN 信号は U2A では以下のように接続されています。

CDTM[i]_DTM[j]の DTM[j-1]_PSU_IN	接続先
CDTM0 DTM[4-1]_PSU_IN	Low 固定
CDTM0 DTM[5-1]_PSU_IN	CDTM0_DTM4_PSU_IN
CDTM1 DTM[4-1]_PSU_IN	CDTM0_DTM5_PSU_IN
CDTM1 DTM[5-1]_PSU_IN	CDTM1_DTM4_PSU_IN
CDTM2 DTM[4-1]_PSU_IN	CDTM1_DTM5_PSU_IN
CDTM2 DTM[5-1]_PSU_IN	CDTM2_DTM4_PSU_IN
CDTM3 DTM[4-1]_PSU_IN	CDTM2_DTM5_PSU_IN
CDTM3 DTM[5-1]_PSU_IN	CDTM3_DTM4_PSU_IN

表 6-46 フェーズシフト制御時 GTM[g].CDTM[i]_DTM[j]_CH_CTRL1 レジスタ設定項目

ビット名	説明
I1SEL_[x]	フェーズシフト機能で使用する信号を設定します。 0：SHIFT[x] 1：COUT[x-1](※1)
SH_EN_[x]	フェーズシフト機能の使用有無を設定します。 0：フェーズシフト機能を使用しません。 1：フェーズシフト機能を使用します。

※1：DTM チャネル x の DTM[j]_COUT[x-1]信号は U2A では以下のように接続されています。

DTM チャネル x の DTM[j]_ COUT[x-1]	接続先
DTM4_COUT[0-1]	未使用
DTM4_COUT[1-1]	DTM4_COUT0
DTM4_COUT[2-1]	DTM4_COUT1
DTM4_COUT[3-1]	DTM4_COUT2
DTM5_COUT[0-1]	DTM4_COUT3
DTM5_COUT[1-1]	DTM5_COUT0
DTM5_COUT[2-1]	DTM5_COUT1
DTM5_COUT[3-1]	DTM5_COUT2

DTM チャネル[x]の信号設定、およびデッドタイムの設定については 6.6.1 を参照してください。

DTM_PSU は、GTM[g].CDTM[i]_DTM[j]_PS_CTRL レジスタの TIM_SEL、PSU_IN_SEL、IN_POL ビット、GTM[g].CDTM[i]_DTM[j]_CTRL レジスタの DTM_SEL(bit1)ビットで指定した信号を SHIFT[x]信号として出力します。ただし、SHIFT1～SHIFT3 の場合のみ、DTM_PSU に IN[x]_edge が入力されると RELBLK で指定したクロック数の間、SHIFT[x]の出力を停止(Low 固定出力)します。

DTM チャネル[x]は、GTM[g].CDTM[i]_DTM[j]_CH_CTRL1 レジスタの SH_EN_[x]ビットにてフェーズシフト機能を使用する設定としている場合、GTM[g].CDTM[i]_DTM[j]_CH_CTRL1 レジスタの I1SEL_[x]ビットにて選択された信号が High の間、DTM チャネル[x]への入力信号として DTM[j]_IN[x-1]を使用します。

6.6.4 信号結合

DTM への入力信号(DTM[j]_IN[x])を他の信号と論理演算的に結合し、信号出力を制御します。

表 6-47 信号結合制御時 GTM[g].CDTM[i]_DTM[j]_CH_CTRL1 レジスタ設定項目

ビット名	説明
I1SEL_[x]	入力信号(DTM[j]_IN[x])と結合する信号を設定します。 0：SHIFT[x](※1) 1：COUT[x-1](※2)
O1SEL_[x]	逆相信号の出力内容を設定します。 0：デッドタイムを付加した逆相信号を出力します。 1：O1F_[x]で選択した処理を行った信号を出力します。 本設定では、出力信号に対してデッドタイムは付与されません。
O1F_[x]	信号処理の内容を設定します。 0：DTM[j]_IN[x]と I1SEL_[x]で指定した信号の信号結合は行いません。 (デッドタイムを付与した正相信号と逆相信号を XNOR 演算します。) 1：DTM[j]_IN[x]と I1SEL_[x]で指定した信号を XOR 演算します。 2：DTM[j]_IN[x]と I1SEL_[x]で指定した信号を AND 演算します。 3：DTM[j]_IN[x]_T を出力します。

※1：SHIFT[x]の詳細は、6.6.3 を参照ください。

※2：DTM チャネル x の DTM[j]_COUT[x-1]信号は U2A では以下のように接続されています。

DTM チャネル x の DTM[j]_ COUT[x-1]	接続先
DTM4_COUT[0-1]	未使用
DTM4_COUT[1-1]	DTM4_COUT0
DTM4_COUT[2-1]	DTM4_COUT1
DTM4_COUT[3-1]	DTM4_COUT2
DTM5_COUT[0-1]	DTM4_COUT3
DTM5_COUT[1-1]	DTM5_COUT0
DTM5_COUT[2-1]	DTM5_COUT1
DTM5_COUT[3-1]	DTM5_COUT2

レジスタの設定により以下の論理演算での信号結合が可能です。

SHIFT[x]信号の選択設定である TIM_SEL、PSU_IN_SEL、DTM_SEL(bit1)ビットの設定、およびその信号レベルの設定である IN_POL の設定については表 6-44 を参照してください。また POL1_[x]ビットの設定については表 6-38 を参照してください。

表 6-48 信号結合制御

O1F_[x]	POL1_[x]	IN_POL	論理演算
1	0	0	XOR
2	0	0	AND
1	1	0	XNOR
2	1	0	NAND
1	1	1	XNOR
2	1	1	OR
1	0	1	XOR
2	0	1	NOR

6.6.5 出力シャットオフ機能

SHUT_OFF 信号は GTM[g].CDTM[i]_DTM[j]_PS_CTRL レジスタの TIM_SEL、PSU_IN_SEL、IN_POL ビット、GTM[g].CDTM[i]_DTM[j]_CTRL レジスタの DTM_SEL(bit1)ビットにより決まります。SHUT_OFF 信号が High となると、シャットオフ状態となり、DTM の信号出力を GTM[g].CDTM[i]_DTM[j]CH_CTRL2_SR レジスタで設定された状態へ切り替えます。シャットオフ解除方法を実行することによりシャットオフ状態が解除されると、GTM[g].CDTM[i]_DTM[j]_CH_CTRL2 レジスタで設定された状態へ戻ります。

GTM[g].CDTM[i]_DTM[j]_CTRL レジスタの UPD_MODE ビットに 1、2、3 以外の値を設定した場合、本機能は無効となります。

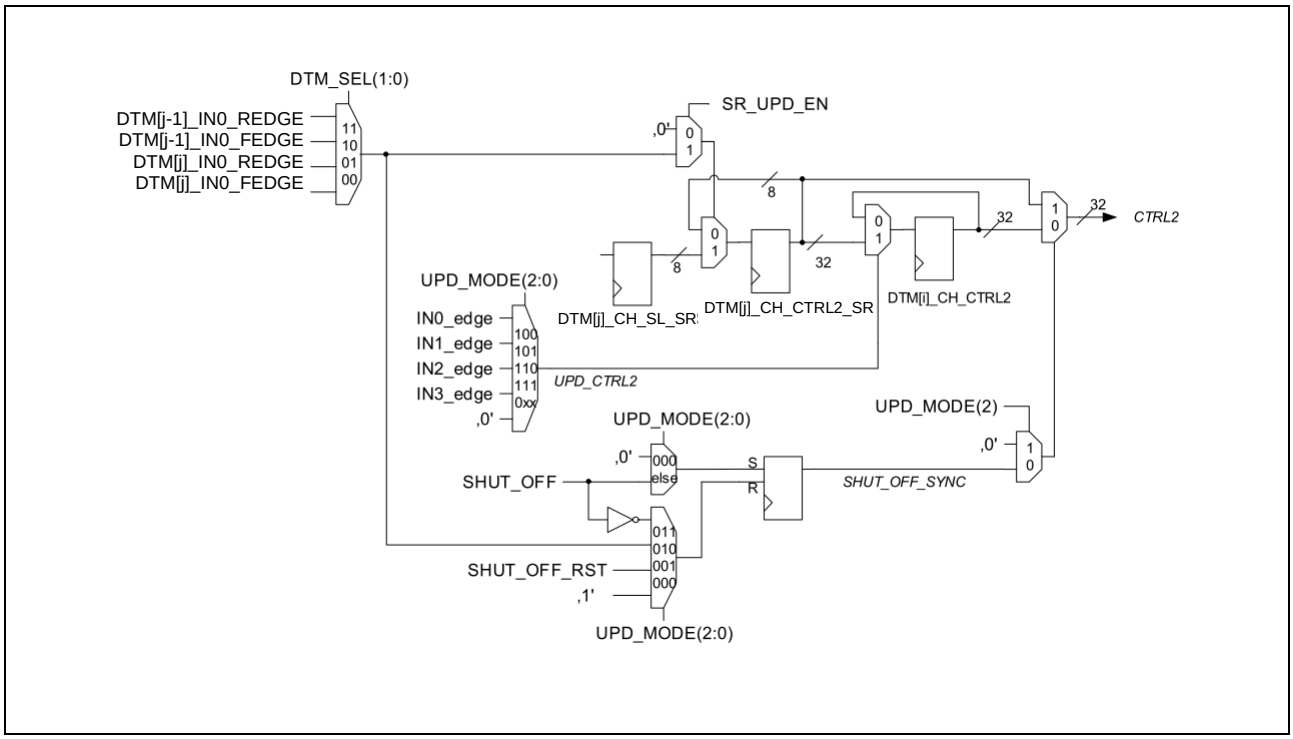


図 6-8 GTM[g].CDTM[i]_DTM[j]_CH_CTRL2 レジスタの同期更新

表 6-49 出力シャットオフ機能時 GTM[g].CDTM[i]_DTM[j]_CTRL レジスタ設定項目

ビット名	説明
CLK_SEL	入力クロック信号を設定する。 0：SYS_CLK 1：CMU_CLK0 2：CMU_CLK1 3：CMU_CLK2
DTM_SEL	SHUT_OFF 信号の元となる信号と、レジスタ更新トリガエッジを設定します。 SHUT_OFF 信号の元となる信号： 0：GTM[g].CDTM[i]_DTM[j]PS_CTRL レジスタの PSU_IN_SEL ビットで指定した信号 1：GTM[g].CDTM[i]_DTM[j]_PS_CTRL レジスタの PSU_IN_SEL ビットで指定した信号 2：DTM[j-1]_PSU_IN 信号(※1) 3：DTM[j-1]_PSU_IN 信号(※1) レジスタ更新トリガエッジ： 0：DTM[j]_IN0 立ち下がりエッジ 1：DTM[j]_IN0 立ち上がりエッジ 2：DTM[j-1]_IN0 立ち下がりエッジ 3：DTM[j-1]_IN0 立ち上がりエッジ
UPD_MODE	シャットオフ状態解除方法を設定します。 1：本レジスタの SHUT_OFF_RST ビットを設定することでシャットオフ状態を解除します。 2：本レジスタの DTM_SEL で設定する信号エッジでシャットオフ状態を解除します。 3：SHUT_OFF 信号が Low 状態でシャットオフ状態を解除します。
SR_UPD_EN	GTM[g].CDTM[i]_DTM[j]_CH_CTRL2_SR レジスタの SL0_[x]および SL1_[x]ビットを GTM[g].CDTM[i]_DTM[j]_CH_SR レジスタの内容で更新するかを設定します。 更新トリガは本レジスタの DTM_SEL ビットで設定する信号エッジです。 0：更新しません。 1：更新します。

※1：CDTM[i]_DTM[j]の DTM[j-1]_PSU_IN 信号は U2A では以下のように接続されています。

CDTM[i]_DTM[j]の DTM[j-1]_PSU_IN	接続先
CDTM0 DTM[4-1]_PSU_IN	Low 固定
CDTM0 DTM[5-1]_PSU_IN	CDTM0_DTM4_PSU_IN
CDTM1 DTM[4-1]_PSU_IN	CDTM0_DTM5_PSU_IN
CDTM1 DTM[5-1]_PSU_IN	CDTM1_DTM4_PSU_IN
CDTM2 DTM[4-1]_PSU_IN	CDTM1_DTM5_PSU_IN
CDTM2 DTM[5-1]_PSU_IN	CDTM2_DTM4_PSU_IN
CDTM3 DTM[4-1]_PSU_IN	CDTM2_DTM5_PSU_IN
CDTM3 DTM[5-1]_PSU_IN	CDTM3_DTM4_PSU_IN

表 6-50　出力シャットオフ機能時 GTM[g].CDTM[i]_DTM[j]_CH_SR レジスタ設定項目

ビット名	説明
SL1_[x]	GTM[g].CDTM[i]_DTM[j]_CH_CTRL2_SR レジスタの SL1_[x]ビットのシャドウレジスタ
SL0_[x]	GTM[g].CDTM[i]_DTM[j]_CH_CTRL2_SR レジスタの SL0_[x]ビットのシャドウレジスタ

シャットオフ発生時の信号の状態を設定する GTM[g].CDTM[i]_DTM[j]_CH_CTRL2_SR レジスタの設定内容は、GTM[g].CDTM[i]_DTM[j]_CH_CTRL2 レジスタと同様です。設定内容については**表 6-38** を参照してください。

シャットオフ条件(SHUT_OFF 信号が High)と GTM[g].CDTM[i]_DTM[j]_CTRL レジスタの UPD_MODE ビットで指定するシャットオフ解除条件が競合する場合、シャットオフ解除が優先されます。

また、シャットオフ条件(SHUT_OFF 信号が High)を満たしたまま、シャットオフ状態が解除されると再度シャットオフ状態となります。例えば、シャットオフ解除条件を DTM_SEL で設定する信号エッジとした場合、SHUT_OFF 信号が High のまま、対象の信号エッジが発生しても一時的にシャットオフが解除されますが、すぐにシャットオフ状態に戻ります。

GTM[g].CDTM[i]_DTM[j]_CH_CTRL2_SR レジスタの SL ビット群(SL0_[x]および SL1_[x])には、シャドウレジスタとして GTM[g].CDTM[i]_DTM[j]_CH_SR レジスタが存在します。GTM[g].CDTM[i]_DTM[j]_CTRL レジスタの SR_UPD_EN ビットを有効とした場合、GTM[g].CDTM[i]_DTM[j]_CTRL レジスタの DTM_SEL ビットにて選択された信号エッジをトリガとし、GTM[g].CDTM[i]_DTM[j]_CH_SR レジスタの SL1_[x]、SL0_[x]ビットにより GTM[g].CDTM[i]_DTM[j]_CH_CTRL2_SR レジスタの SL1_[x]、SL0_[x]ビットが更新されます。

6.7 使用例

信号出力処理例を下記にて説明します。なお、割り込み処理、および割り込みルーチンの割り込みベクタテーブルへの登録方法については、本アプリケーションノートでは説明しません。なお、このソースコードでは CMU や TBU の初期設定については省略しています。

6.7.1 SOMI 信号出力例

ATOM0 のチャネル 0 にて、チャネルモード SOMI を使用して指定レベルの信号を出力します。設定例では、SL ビットで ATOM0 のチャネル 0 の出力レベルを設定します。

設定例：

```
void gtm0_atom0_ch0_main(unsigned long level)
{
    /* ATOM0 all channel is invalid  */
    GTM0.ATOM0_AGC_OUTEN_STAT.UINT32 = 0x00005555;
    GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 = 0x00005555;
    while(GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 != 0x00000000);

    /* ATOM0 channel mode PWM */
    //CH0
    GTM0.ATOM0_CH0_CTRL.BIT.SL  = level;
    GTM0.ATOM0_CH0_CTRL.BIT.ACB  = 1;

    //AGC
    GTM0.ATOM0_AGC_ENDIS_CTRL.BIT.ENDIS_CTRL0 = 2; //Enable channel on an update trigger
    GTM0.ATOM0_AGC_OUTEN_CTRL.BIT.OUTEN_CTRL0 = 2; //Enable channel output on an update trigger
    GTM0.ATOM0_AGC_GLB_CTRL.UINT32 = 0x1; //Trigger forced updating
    while( 0x3 != GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 );
}
```

図 6-9 ATOM SOMI 出力プログラムコード例

6.7.2 SOMC 信号出力例

ATOM0 のチャネル 0 にて、チャネルモード SOMC を使用して PWM 信号を出力します。設定例では、CCU1 コンペアマッチイベント割り込み発生時に配列に設定しているデューティ値と周期値で、ATOM0 のチャネル 0 の出力を更新します。

設定例：

```
#define MAX_TIMESTAMP 0x00ffffff /* Max timstamp value */
#define COUNT_DIM_MAX 5 /* Max index of duty and period */

static unsigned long g_u32Count = 0;
static unsigned long g_u32TimeStamp;
static volatile unsigned long g_u32Dummy = 0;

static unsigned long g_u32Duty[COUNT_DIM_MAX ] = {0x1000, 0x1000, 0x1000, 0x1000, 0x1000};
static unsigned long g_u32Period[COUNT_DIM_MAX] = {0x2000, 0x2000, 0x2000, 0x2000, 0x2000};

void gtm0_atom0_ch0_main(void)
{
    /* ATOM0 All channel is invalid */
    GTM0.ATOM0_AGC_OUTEN_STAT.UINT32 = 0x00005555;
    GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 = 0x00005555;
    while( 0x0 != GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 );

    GTM0.ATOM0_CH0_CTRL.UINT32 = 0x00000931; //ATOM channel mode is SOMC
                                             //When CCU0 compare mutch, ATOM output is low level
                                             //When CCU1 compare mutch, ATOM output is high level
                                             //First level of ATOM output is high level
                                             //Use the lower 24 bit of TBU channel 0
    GTM0.ATOM0_CH0_IRQ_EN.UINT32 = 0x2;      //CCU1TC_IRQ interrupt is valid

    /* Get current timestamp */
    g_u32TimeStamp = GTM0.TBU_CH0_BASE.UINT32 & MAX_TIMESTAMP;

    /* Set duty */
    g_u32TimeStamp += g_u32Duty[0];
    GTM0.ATOM0_CH0_CM0.UINT32 = g_u32TimeStamp & MAX_TIMESTAMP;
    GTM0.ATOM0_CH0_SR0.UINT32 = GTM0.ATOM0_CH0_CM0.UINT32;

    /* Set cycle */
    g_u32TimeStamp += g_u32Period[0];
    GTM0.ATOM0_CH0_CM1.UINT32 = g_u32TimeStamp & MAX_TIMESTAMP;
    GTM0.ATOM0_CH0_SR1.UINT32 = GTM0.ATOM0_CH0_CM1.UINT32;

    /* Forced updating is valid */
    GTM0.ATOM0_AGC_FUPD_CTRL.UINT32 = 0x2;

    /* Channel is valid(Actually, channel is valid when forced updating) */
    GTM0.ATOM0_AGC_OUTEN_CTRL.UINT32 = 0x2;
    GTM0.ATOM0_AGC_ENDIS_CTRL.UINT32 = 0x2;
    GTM0.ATOM0_AGC_GLB_CTRL.UINT32 = 0x1; //Trigger forced updating
    while( 0x3 != GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 );
}
```

図 6-10 ATOM SOMC 出力プログラムコード例(1/2)

```
void irq_gtm0_atom0_ch0(void)
{
    GTM0.ATOM0_CH0_IRQ_NOTIFY.UINT32 = 0x2; //CCU1TC_IRQ interrupt status is clear
    g_u32Dummy = GTM0.ATOM0_CH0_SR0.UINT32; //Get timestamp of CCU0 compare mutch
    g_u32Dummy = GTM0.ATOM0_CH0_SR1.UINT32; //Get timestamp of CCU1 compare mutch

    /* Counter increment  */
    g_u32Count++;
    if( g_u32Count >= COUNT_DIM_MAX ) {
        g_u32Count = 0;
    }
    /* Set the next duty  */
    g_u32TimeStamp += g_u32Duty[g_u32Count];
    GTM0.ATOM0_CH0_CM0.UINT32 = g_u32TimeStamp & MAX_TIMESTAMP;
    /* Set the next cycle  */
    g_u32TimeStamp += g_u32Period[g_u32Count];
    GTM0.ATOM0_CH0_CM1.UINT32 = g_u32TimeStamp & MAX_TIMESTAMP;
}
```

図 6-11　ATOM SOMC 出力プログラムコード例(2/2)

6.7.3 SOMP 信号出力例

ATOM0 のチャネル 0 にて、チャネルモード SOMP を使用して PWM 信号を出力します。設定例では、次に出力する PWM 信号の周期 SR0、デューティを SR1 レジスタに設定し、CCU1 コンペアマッチイベント割り込み発生時に更新します。

設定例：

```
#define DUTY  100
#define CYCLE 200

void gtm0_atom0_ch0_main(void)
{
    /* ATOM0 all channel is invalid  */
    GTM0.ATOM0_AGC_OUTEN_STAT.UINT32 = 0x00005555;
    GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 = 0x00005555;
    while(GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 != 0x00000000);

    /* ATOM0 channel mode PWM */
    //CH0
    GTM0.ATOM0_CH0_CTRL.UINT32 = 0x00000802;  //ATOM channel mode is SOMP
                                              //CMU_CLK0 selected
                                              //Duty is High level in SOMP mode
    GTM0.ATOM0_CH0_IRQ_EN.UINT32 = 0x2;       //CCU1TC_IRQ interrupt is valid

    GTM0.ATOM0_CH0_CM1.UINT32 = DUTY;   //duty
    GTM0.ATOM0_CH0_CM0.UINT32 = CYCLE;  //cycle
    GTM0.ATOM0_CH0_SR1.UINT32 = DUTY;   //shadow resister of CM1
    GTM0.ATOM0_CH0_SR0.UINT32 = CYCLE;  //shadow resister of CM0
    GTM0.ATOM0_CH0_CN0.UINT32 = CYCLE;  //counter

    //AGC
    GTM0.ATOM0_AGC_GLB_CTRL.BIT.UPEN_CTRL0 = 2;     //ATOM channel enable update of register CM0,
                                                    //CM1 and CLK_SRC from SR0, SR1 and CLK_SRC_SR.
    GTM0.ATOM0_AGC_ENDIS_CTRL.BIT.ENDIS_CTRL0 = 2; //Enable channel on an update trigger
    GTM0.ATOM0_AGC_OUTEN_CTRL.BIT.OUTEN_CTRL0 = 2; //Enable channel output on an update trigger
    GTM0.ATOM0_AGC_FUPD_CTRL.BIT.FUPD_CTRL0 = 2;   //Force update enabled
    GTM0.ATOM0_AGC_GLB_CTRL.UINT32 = 0x1;          //Trigger forced updating
    while( 0x3 != GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 );
}

void irq_gtm0_atom0_ch0(void)
{
    GTM0.ATOM0_CH0_IRQ_NOTIFY.UINT32 = 0x2; //CCU1TC_IRQ interrupt status is clear

    GTM0.ATOM0_CH0_SR1.UINT32 = DUTY;
    GTM0.ATOM0_CH0_SR0.UINT32 = CYCLE;
}
```

図 6-12 ATOM SOMP 出力プログラムコード例

6.7.4 SOMS 信号出力例

ATOM0 のチャネル 0 にて、チャネルモード SOMS を使用してシリアル信号を出力します。設定例では、次に出力するシリアル信号データを SR0、シリアル信号データのビット数を SR1 レジスタ、シフト方向は右シフトに設定し、CCU0 コンペアマッチイベント割り込み発生時に更新します。

設定例：

```
#define SHIFT_BIT_NUM 8
#define SEND_DATA 0xA5

void gtm0_atom0_ch0_main(void)
{
    /* ATOM0 all channel is invalid  */
    GTM0.ATOM0_AGC_OUTEN_STAT.UINT32 = 0x00005555;
    GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 = 0x00005555;
    while(GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 != 0x00000000);

    //CH0
    GTM0.ATOM0_CH0_CTRL.UINT32 = 0x00000803;       //ATOM channel mode is SOMS
                                                   //CLK_SRC_SR set 1 selected
                                                   //CMU_CLK0 selected
                                                   //Initial level is Low (!SL) in SOMS mode
                                                   //CM1 is used as a 24 bit shift register
                                                   //Right shift of data is started from bit 0 of CM1
    GTM0.ATOM0_CH0_IRQ_EN.UINT32 = 0x1;            //CCU0TC_IRQ interrupt is valid

    GTM0.ATOM0_CH0_CM1.UINT32 = SEND_DATA;         //shift output data
    GTM0.ATOM0_CH0_CM0.UINT32 = SHIFT_BIT_NUM;     //shift number
    GTM0.ATOM0_CH0_SR1.UINT32 = SEND_DATA;         //shadow resister of CM1
    GTM0.ATOM0_CH0_SR0.UINT32 = SHIFT_BIT_NUM - 1; //shadow resister of CM0

    //AGC
    GTM0.ATOM0_AGC_GLB_CTRL.BIT.UPEN_CTRL0 = 2;    //ATOM channel enable update of register CM0,
                                                   //CM1 and CLK_SRC from SR0, SR1 and CLK_SRC_SR.
    GTM0.ATOM0_AGC_ENDIS_CTRL.BIT.ENDIS_CTRL0 = 2; //Enable channel on an update trigger
    GTM0.ATOM0_AGC_OUTEN_CTRL.BIT.OUTEN_CTRL0 = 2; //Enable channel output on an update trigger
    GTM0.ATOM0_AGC_FUPD_CTRL.BIT.FUPD_CTRL0 = 2;   //Force update enabled
    GTM0.ATOM0_AGC_GLB_CTRL.UINT32 = 0x1;          //Trigger forced updating
    while( 0x3 != GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 );
}

void irq_gtm0_atom0_ch0(void)
{
    GTM0.ATOM0_CH0_IRQ_NOTIFY.UINT32 = 0x1; //CCU0TC_IRQ interrupt status is clear

    GTM0.ATOM0_CH0_SR1.UINT32 = SEND_DATA;
    GTM0.ATOM0_CH0_SR0.UINT32 = SHIFT_BIT_NUM - 1;
}
```

図 6-13 ATOM SOMS 出力プログラムコード例

6.7.5 SOMB 信号出力例

ATOM0 のチャネル 0 にて、チャネルモード SOMB を使用して PWM 信号を出力します。設定例では、CCU1 コンペアマッチイベント割り込み発生時に配列に設定しているデューティ値と周期値で、ATOM0 のチャネル 0 の出力を更新します。

設定例：

```
#define MAX_TIMESTAMP 0x00ffffff /* Max timstamp value */
#define COUNT_DIM_MAX 5 /* Max index of duty and period */

static unsigned long g_u32Count = 0;
static unsigned long g_u32TimeStamp;

static unsigned long g_u32Duty[COUNT_DIM_MAX ] = {0x1000, 0x1000, 0x1000, 0x1000, 0x1000};
static unsigned long g_u32Period[COUNT_DIM_MAX] = {0x2000, 0x2000, 0x2000, 0x2000, 0x2000};

void gtm0_atom0_ch0_main(void)
{
    /* ATOM0 All channel is invalid */
    GTM0.ATOM0_AGC_OUTEN_STAT.UINT32 = 0x00005555;
    GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 = 0x00005555;
    while( 0x0 != GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 );

    GTM0.ATOM0_CH0_CTRL.UINT32 = 0x40000930; //ATOM channel mode is SOMB
                                             //When CCU0 compare mutch, ATOM output is low level
                                             //When CCU1 compare mutch, ATOM output is high level
                                             //First level of ATOM output is high level
                                             //Use the lower 24 bit of TBU channel 0
    GTM0.ATOM0_CH0_IRQ_EN.UINT32 = 0x2;      //CCU1TC_IRQ interrupt is valid

    /* Get current timestamp */
    g_u32TimeStamp = GTM0.TBU_CH0_BASE.UINT32 & MAX_TIMESTAMP;

    /* Set duty */
    g_u32TimeStamp += g_u32Duty[0];
    GTM0.ATOM0_CH0_CM0.UINT32 = g_u32TimeStamp & MAX_TIMESTAMP;
    GTM0.ATOM0_CH0_SR0.UINT32 = GTM0.ATOM0_CH0_CM0.UINT32;

    /* Set cycle */
    g_u32TimeStamp += g_u32Period[0];
    GTM0.ATOM0_CH0_CM1.UINT32 = g_u32TimeStamp & MAX_TIMESTAMP;
    GTM0.ATOM0_CH0_SR1.UINT32 = GTM0.ATOM0_CH0_CM1.UINT32;

    /* Forced updating is valid */
    GTM0.ATOM0_AGC_FUPD_CTRL.UINT32 = 0x2;

    /* Channel is valid(Actually, channel is valid when forced updating) */
    GTM0.ATOM0_AGC_OUTEN_CTRL.UINT32 = 0x2;
    GTM0.ATOM0_AGC_ENDIS_CTRL.UINT32 = 0x2;
    GTM0.ATOM0_AGC_GLB_CTRL.UINT32 = 0x1; //Trigger forced updating
    while( 0x3 != GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 );
}
```

図 6-14 ATOM SOMB 出力プログラムコード例(1/2)

```
void irq_gtm0_atom0_ch0(void)
{
    GTM0.ATOM0_CH0_IRQ_NOTIFY.UINT32 = 0x2; //CCU1TC_IRQ interrupt status is clear

    /* Counter increment  */
    g_u32Count++;
    if( g_u32Count >= COUNT_DIM_MAX ) {
        g_u32Count = 0;
    }
    /* Set the next duty  */
    g_u32TimeStamp += g_u32Duty[g_u32Count];
    GTM0.ATOM0_CH0_CM0.UINT32 = g_u32TimeStamp & MAX_TIMESTAMP;
    /* Set the next cycle  */
    g_u32TimeStamp += g_u32Period[g_u32Count];
    GTM0.ATOM0_CH0_CM1.UINT32 = g_u32TimeStamp & MAX_TIMESTAMP;
}
```

図 6-15　ATOM SOMB 出力プログラムコード例(2/2)

7. シーケンサ機能

RISC のような CPU が搭載されたデータ制御モジュールにて、ARU を経由してのデータのやり取りや制御を行うことが可能です。ただし、各サブモジュールの初期化については、あらかじめマイコンで実施する必要があります。また、MCS は AEI バスマスタインタフェース経由で GTM のサブモジュールのレジスタにアクセスすることが可能です。また MCS の ADC インタフェース経由で U2A に搭載された AD コンバータの変換結果をリードすることが可能です。

7.1 構成

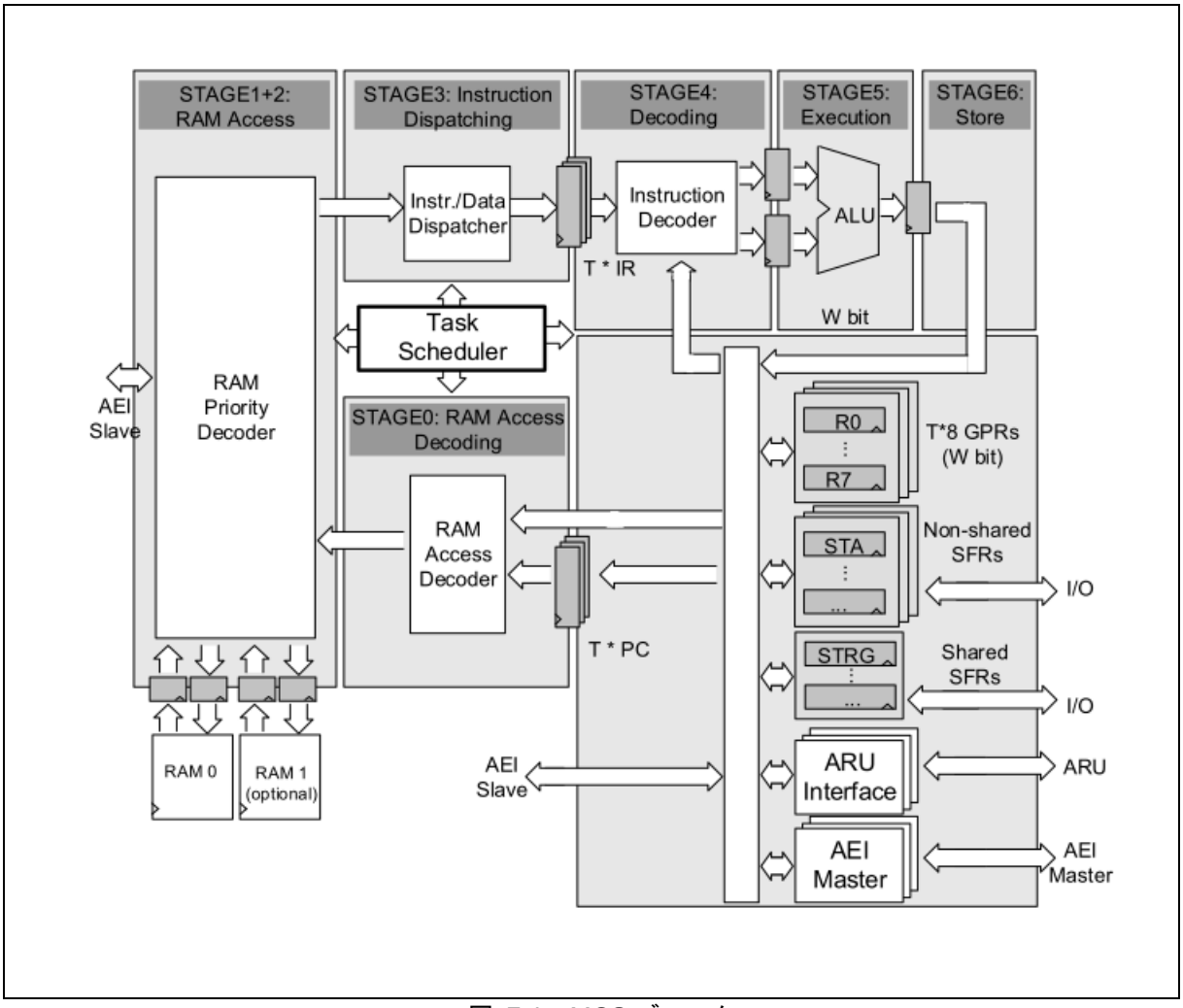


図 7-1 MCS ブロック

・MCS に搭載されているチャネル数は 8 です。

・MCS のレジスタアクセスは 32 ビットです。

・RAM のデータは 32 ビット単位です。

・プログラム実行パイプラインの、ステージ数は 7 です。

MCS の内部レジスタは以下のとおりです。

(1) CTRG (Clear Trigger Bit Register)
クリアトリガビットレジスタです。本レジスタは 24 ビットです。
1 を設定したビットに対応する STRG のビットがクリアされます。
EN_TIM_FOUT を設定することで TIM チャネルの F_OUT 信号をリードすることもできます。

本レジスタは各チャネルで共通で使用されます。

(2) STRG (Set Trigger Bit Register)
トリガビットレジスタです。本レジスタは 24 ビットです。
マイコンおよび各チャネル間で同期を取る場合等に使用します。
このトリガビットをクリアしたい場合、CTRG レジスタにて対応するビットを設定してクリアしてください。
本レジスタは各チャネルで共通で使用されます。

(3) MHB (Memory High Byte Register)
メモリハイバイトレジスタです。本レジスタは 8 ビットです。
RAM からデータを読み出す場合、下位 24 ビットは指定レジスタへ転送され、上位 8 ビットは本レジスタに転送されます。RAM へデータを書き込む場合、下位 24 ビットは指定レジスタから転送され、上位 8 ビットは本レジスタから転送されます。
本レジスタは各チャネルで独立して用意されます。

(4) ACB (ARU Control Bit Register)
ARU コントロールビットレジスタです。本レジスタは 5 ビットです。
ARU からのデータ受信時に ARU データ上位 5 ビットが本レジスタへ転送され、ARU へのデータ送信時に本レジスタから ARU データ上位 5 ビットへ転送されます。
本レジスタは各チャネルで独立して用意されます。

(5) STA (Status Register)
ステータスレジスタです。演算結果フラグやノンブロッキング用の ARU 転送成否フラグ等があります。また、IRQ ビットを設定することにより、MCS[i]_MCS[x]_IRQ 割り込みを発生させることもできます。
本レジスタは各チャネルにて独立して用意されます。

(6) R0～R7
汎用レジスタ群です。本レジスタは 24 ビットレジスタです。
R4～R7 レジスタは MCS の命令セットにより暗黙的に使用されます。

Rx	説明
R4	乗算命令(MULU、MULS)の上位乗算結果の格納先として使用。 除算命令(DIVU、DIVS)の剰余の格納先として使用。
R5	メモリのリード命令(MRDIO)、ライト命令(MWRIO)のオフセットレジスタとして使用。
R6	ウェイト命令(WURMX、WURCX)のマスクレジスタ、 分岐 I 命令(JMPI, JBSI, JBCI, and CALLI.)のジャンプ先アドレス、 ARU データにアクセスする際のアドレスレジスとして使用。
R7	スタックポインタとして使用。

本レジスタは各チャネルで独立して用意されます。

(7) RS0～RS7 (Mirror of succeeding channels register R[y])
RS0～RS7 レジスタは一つ後のチャネルの R0～R7 レジスタのミラーです。MSC チャネル 7 の RS0～R7 は MSC チャネル 0 の R0～R7 レジスタのミラーです。

(8) TBU_TS0 (TBU Timestamp TS0 Register)
TBU チャネル 0 のタイムスタンプです。24 ビットの読み出し専用レジスタです。

(9) TBU_TS1 (TBU Timestamp TS1 Register)
TBU チャネル 1 のタイムスタンプです。24 ビットの読み出し専用レジスタです。

(10) TBU_TS1 (TBU Timestamp TS2 Register)
TBU チャネル 2 のタイムスタンプです。24 ビットの読み出し専用レジスタです。

(11) GMI0 (GTM module interrupt 0 register)
ATOM または TIM チャネル 0-7 の割り込み要求がリードできます。1 を設定したビットに対応する ATOM または TIM の割り込み要求をクリアできます。

(12) GMI1 (GTM module interrupt 1 register)
MCS0 と MCS[i+1]のチャネル 0～7 の割り込み要求がリードできます。1 を設定したビットに対応する割り込み要求はクリアされます。

各プログラムのスケジューリングについては以下の 4 つのモードがあります。

(1) Round Robin Scheduling
起動するチャネルとマイコンの MCS-RAM へのアクセスに対し、それぞれ均等に処理時間を割り振ります。

(2) Accelerated Scheduling
起動したチャネルの中で、実行可能な処理に対して処理時間を割り振ります。処理順はハードウェアに依存します。ARU ブロッキング等によるサスペンド状態の処理については処理時間を割り振りません。

(3) Single Prioritization Scheduling
指定された 1 つのチャネルの処理に対して、優先的に処理時間を割り振ります。指定されたチャネルの処理がサスペンド状態に遷移した場合に、スケジューリング動作は Accelerated Scheduling となり、他のプログラムが実行可能となります。
指定したチャネル番号のチャネルが存在しない場合は、マイコンからの MCS-RAM へのアクセスが優先となります。

(4) Multiple Prioritization Scheduling
指定された範囲のチャネルの処理に対して、優先的に処理時間を割り振ります。指定された範囲内のチャネルの中でも、チャネル番号が小さいチャネルの処理の優先順位が高くなります。指定された範囲のチャネルの処理がすべてサスペンド状態に遷移した場合に、スケジューリング動作は Accelerated Scheduling となり、他のプログラムが実行可能となります。

7.2 レジスタ設定

MCS のコードを実行するために、以下のレジスタ群を設定する必要があります。

表 7-1 GTM[g].MCS[i]_CTRL_STAT レジスタ設定項目

ビット名	説明
SCD_MODE	スケジューリングモードを設定します。 0：Accelerated Scheduling 1：Round Robin Scheduling 2：Single Priority Scheduling 3：Multiple Priority Scheduling
SCD_CH	SCD_MODE=0(Accelerated Scheduling)設定時： 未使用です。 SCD_MODE=1(Round Robin Scheduling)設定時： Round Robin Scheduling の対象とする最大の MCS チャネル番号を設定します。 チャネル番号が 0～SCD_CH の MCS チャネルが昇順にパイプラインに割り当てられます。 SCD_CH がパイプラインに割り当てられた後、CPU の RAM アクセスを有効にするために空のサイクルがスケジュールされます。 SCD_CH が MCS に搭載されている最大チャネル番号より大きい場合、SCD_CH は最大チャネル番号-1 と見なされます。 SCD_MODE=2(Single Priority Scheduling)設定時： 優先的に処理するチャネル番号を指定します。 存在しないチャネル番号を設定した場合は、 マイコンから MCS-RAM へのアクセスを優先します。 SCD_MODE=3(Multiple Priority Scheduling)設定時： 優先的に処理するチャネル番号の最大値を設定します。 チャネル番号 0～設定値のチャネルが優先されます。 さらに、優先チャネルの中でも 番号が小さい(0 に近い)チャネルが優先されます。 存在しないチャネル番号を設定した場合は、 全チャネルが優先となります。

事前にアセンブルした MCS プログラムを、MCS-RAM の 0 番地より任意のサイズ転送後、GTM[g].MCS[i]_CH[x]_CTRL レジスタの EN ビットを設定してください。アセンブルの詳細は 7.4 を参照してください。

また、GTM[g].MCS[i]_CH[x]_CTRL レジスタの EN ビットを設定する前に、GTM[g].MCS[i]_CH[x]_PC レジスタに MCS-RAM のアドレス(MCS-RAM の先頭からの相対的なアドレス)を設定することにより任意のアドレスからコードを実行できます。

7.3 CPU とのインタフェース

次の方法を用いることでマイコンと同期を取ることが可能です。

7.3.1 割り込み通知

表 7-2 MCS の割り込み通知一覧

IRQ	説明
MCS[i]_ERR_IRQ[x]	以下の場合、本割り込みが通知されます。 ・ MCS プログラムで STA レジスタの ERR ビットをセット ・ メモリ ECC エラー発生 ・ 無効オペランド指定エラー発生 ・ メモリオーバフロー発生 ・ 0 で除算(DIVU or DIVS 命令) ・ MCS チャネルが GTM[g].MCS[i]_REG_PROT レジスタでライト保護された GPR にライト ・ MCS チャネルが CCM サブモジュールのアドレス範囲保護(ARP)によって保護された範囲のメモリにライト ・ MCS チャネルが GTM[g].MCS_[i]_CTRL_STAT レジスタの HLT_AEIM_ERR ビット=1 中に無効な AEI バスマスタアクセス
MCS[i]_STK_ERR_IRQ[x]	スタックのオーバフロー／アンダフロー発生時にマイコンへ通知されます。
MCS[i]_IRQ[x]	MCS プログラムで STA レジスタの IRQ ビット設定時にマイコンへ割り込みが通知されます。

これらの割り込みを有効にしたい場合は GTM[g].MCS[i]_CH[x]_IRQ_EN レジスタの設定をしてください。また割り込みが発生した場合、GTM[g].MCS[i]_CH[x]_IRQ_NOTIFY レジスタの対応する割り込みのビットを設定することで割り込み要因をクリアすることができます。MCS[i]_IRQ[x]割り込みの場合、マイコン側で GTM[g].MCS[i]_CH[x]_IRQ_NOTIFY レジスタにて割り込み要因をクリアした場合、自動的に STA レジスタの IRQ ビットもクリアされます。MCS[i]_ERR_IRQ[x]割り込みの場合、MCS プログラムで STA レジスタの ERR ビットを設定した場合でも通知されるため、割り込みが発生した際に GTM[g].MCS[i]_CTRL_STAT レジスタの ERR_SRC_ID ビットでエラー要因を確認してください。

7.3.2 イベント通知

STRG レジスタを使用することで、他の MCS チャネルやマイコンと同期をとることが可能です。

例えば、MCS チャネル 0 の特定の処理が行われるまでマイコンを待機させる場合、以下のように記述します。

マイコン側：

```
While( (GTM0.MCS0_STRG & 1) == 0 );       // STRG の bit0 に 1 が設定されるまで待機
```

MCS プログラム側：

```
movl STRG $1                                ; STRG の bit0 に 1 を設定
```

7.3.3 ARU との転送

ARU 経由で MCS のデータを受信する場合は、対応する ARU アドレスを指定してください。

MCS はサブモジュールごとに複数の ARU バッファが割り当てられています。GTM-IP 358 において、MCS の ARU アドレス(MCS に割り当てられた ARU バッファのインデックス)は次の表のとおりです。

表 7-3 MCS の各サブモジュールの ARU アドレス

ARU チャネル	ARU アドレス			
	MCS0	MCS1	MCS2	MCS3
0	$(77)_{16}$	$(8f)_{16}$	$(a7)_{16}$	$(bf)_{16}$
1	$(78)_{16}$	$(90)_{16}$	$(a8)_{16}$	$(c0)_{16}$
2	$(79)_{16}$	$(91)_{16}$	$(a9)_{16}$	$(c1)_{16}$
3	$(7a)_{16}$	$(92)_{16}$	$(aa)_{16}$	$(c2)_{16}$
4	$(7b)_{16}$	$(93)_{16}$	$(ab)_{16}$	$(c3)_{16}$
5	$(7c)_{16}$	$(94)_{16}$	$(ac)_{16}$	$(c4)_{16}$
6	$(7d)_{16}$	$(95)_{16}$	$(ad)_{16}$	$(c5)_{16}$
7	$(7e)_{16}$	$(96)_{16}$	$(ae)_{16}$	$(c6)_{16}$
8	$(7f)_{16}$	$(97)_{16}$	$(af)_{16}$	$(c7)_{16}$
9	$(80)_{16}$	$(98)_{16}$	$(b0)_{16}$	$(c8)_{16}$
10	$(81)_{16}$	$(99)_{16}$	$(b1)_{16}$	$(c9)_{16}$
11	$(82)_{16}$	$(9a)_{16}$	$(b2)_{16}$	$(ca)_{16}$
12	$(83)_{16}$	$(9b)_{16}$	$(b3)_{16}$	$(cb)_{16}$
13	$(84)_{16}$	$(9c)_{16}$	$(b4)_{16}$	$(cc)_{16}$
14	$(85)_{16}$	$(9d)_{16}$	$(b5)_{16}$	$(cd)_{16}$
15	$(86)_{16}$	$(9e)_{16}$	$(b6)_{16}$	$(ce)_{16}$
16	$(87)_{16}$	$(9f)_{16}$	$(b7)_{16}$	$(cf)_{16}$
17	$(88)_{16}$	$(a0)_{16}$	$(b8)_{16}$	$(d0)_{16}$
18	$(89)_{16}$	$(a1)_{16}$	$(b9)_{16}$	$(d1)_{16}$
19	$(8a)_{16}$	$(a2)_{16}$	$(ba)_{16}$	$(d2)_{16}$
20	$(8b)_{16}$	$(a3)_{16}$	$(bb)_{16}$	$(d3)_{16}$
21	$(8c)_{16}$	$(a4)_{16}$	$(bc)_{16}$	$(d4)_{16}$
22	$(8d)_{16}$	$(a5)_{16}$	$(bd)_{16}$	$(d5)_{16}$
23	$(8e)_{16}$	$(a6)_{16}$	$(be)_{16}$	$(d6)_{16}$

MCS より ARU へデータを送信したい場合、MCS プログラムでは次のように記述します。

MCS プログラム側：

```
; 変数設定
WRITE_ACB:      .var 0x000008
WRITE_DATA0:    .var 0x0000a0
WRITE_DATA1:    .var 0x000064

mrd R1 WRITE_ACB                ; WRITE_ACB の値を R1 レジスタに設定
mov ACB R1                      ; R1 レジスタの値を ACB に設定
mrd R1 WRITE_DATA0              ; WRITE_DATA0 の値を R1 レジスタに設定
mrd R2 WRITE_DATA1;             ; WRITE_DATA1 の値を R2 レジスタに設定
awr R1 R2 $0                    ; ARU チャネル 0 に ARU データを書き込み
```

awr 命令により、Data0 に第一オペランド、Data1 に第二オペランド、ACB に ACB レジスタの内容を持つ ARU データが転送されます。

書き込み先の ARU チャネル番号と ARU アドレスの関係については表 7-3 を参照してください。

またニーモニックについてはユーザーズマニュアルを参照してください。

ATOM0 の ARU データ(ARU アドレスが$(11f)_{16}$)を MCS にて受信したい場合、次のようにコードを記述します。ATOM の ARU アドレスについては表 6-24 を参照してください。

MCS プログラム側：

```
ard R0 R1 $11f
```

ard 命令により、ARU アドレスが$(11f)_{16}$の ATOM0 からの ARU データを受信し、R0 に Data0、R1 に Data1、ACB レジスタに ACB の内容がそれぞれ格納されます。

7.4 アセンブル

Bosch 社提供のアセンブラによるアセンブル方法について説明します。

MCS ソースコードを asm-mcs.exe のあるフォルダへコピーし、コマンドプロンプトを開き、asm_mcs.exe のあるフォルダへ cd コマンドで切り替えてください。

フォルダの切り替え後、以下のコマンドを入力してください。

```
asm-mcs -o objfile.h -odef objdef.h src.mcs -arch mcs24-3
```

objfile.h ：生成するオブジェクトファイル名

オブジェクトコードが記述されるファイル名を指定します。

objdef.h ：生成するラベルファイル名

生成したオブジェクトに対応するラベルが記述されるファイル名を指定します。

src.mcs ：MCS ソースコードファイル名

アセンブル対象の MCS ソースコードを指定します。

MCSのサンプルコードを以下に示します。

```
.org $0
jmp tsk0_init

; reserve the stack area.
.org $20
tsk0_stack:

; reserve the variable area.
.org $100
; declare the value for ACB.
ATOM0_ACB:   .var 0x000008 ; ATOM0 ACB (CMU_CLK2)

; declare the memory area which MCS share with the microcomputer.
; the microcomputer sets the value in this area.
; MCS transfers the value of this area to ATOM0.
ATOM0_CH0_SR0: .var 0x000000 ; ATOM0_CH0_SR0
ATOM0_CH0_SR1: .var 0x000000 ; ATOM0_CH0_SR1
ATOM0_CH1_SR0: .var 0x000000 ; ATOM0_CH1_SR0
ATOM0_CH1_SR1: .var 0x000000 ; ATOM0_CH1_SR1
ATOM0_CH2_SR0: .var 0x000000 ; ATOM0_CH2_SR0
ATOM0_CH2_SR1: .var 0x000000 ; ATOM0_CH2_SR1

; main process
tsk0_init:
  movl   R7  (tsk0_stack -4) ; set stack pointer.
  mrd    R1  ATOM0_ACB       ; set ATOM0_ACB in R1 register.
  mov    ACB  R1             ; set R1 register in ACB register.

tsk0_pwm_loop:
  movl   STRG $1             ; set bit0 of STRG register.
  movl   R0 $2               ; set bit1 of R0 register.
  wurm   R0 STRG $2          ; wait until the microcomputer sets bit1 of STRG register.

  orl    CTRG $3             ; clear both bit0 and bit1 of STRG register by CTRG register.

  mrd    R1  ATOM0_CH0_SR0   ; set the value of ATOM0_CH0_SR0 (PWM's period count clock) in R1 register.
  mrd    R2  ATOM0_CH0_SR1   ; set the value of ATOM0_CH0_SR1 (PWM's duty count clock) in R2 register.
  awr    R1 R2 $0            ; transfer R1, R2, and ACB register as ARU data to ARU CH0.


  mrd    R1  ATOM0_CH1_SR0   ; set the value of ATOM0_CH1_SR0 (PWM's period count clock) in R1 register.
  mrd    R2  ATOM0_CH1_SR1   ; set the value of ATOM0_CH1_SR1 (PWM's duty count clock) in R2 register.
  awr    R1 R2 $1            ; transfer R1, R2, and ACB register as ARU data to ARU CH1.


  mrd    R1  ATOM0_CH2_SR0   ; set the value of ATOM0_CH2_SR0 (PWM's period count clock) in R1 register.
  mrd    R2  ATOM0_CH2_SR1   ; set the value of ATOM0_CH2_SR1 (PWM's duty count clock) in R2 register.
  awr    R1 R2 $2            ; transfer R1, R2, and ACB register as ARU data to ARU CH2.


  movl   STA 0x000003        ; trigger MCS_IRQ interrupt.
  jmp    tsk0_pwm_loop

tsk0_done:
  movl   STA 0x000000        ; disable MCS channel.
```

図 7-2 MCS プログラムソースコード例

アセンブルにより生成されるファイルのサンプルを以下に示します。

```
/* generated by MCS-Assembler tool ASM-MCS version 0.9.2 */
/* developed by Robert Bosch GmbH, Germany */
/* target architecture : mcs24-3 */

unsigned long mcs0_mem[90] = {
    0xE000011C,
    0x00000000,
    0x00000000,
    0x00000000,
    0x00000000,
    0x00000000,
    0x00000000,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    省略
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    0x00000000,
    0x00000000,
    0x00000000,
    0x00000008,
    0x00000000,
    0x00000000,
    0x00000000,
    0x00000000,
    0x00000000,
    0x00000000,
    0x1700001C,
    0xA1010100,
    0xA9100000,
    0x1B000001,
    0x10000002,
    0xF0B00002,
    0x5A000003,
    0xA1010104,
    0xA2010108,
    0xB1210000,
    0xA101010C,
    0xA2010110,
    0xB1210001,
    0xA1010114,
    0xA2010118,
    0xB1210002,
    0x18000003,
    0xE0000128,
    0x18000000
};
```

図 7-3 MCS オブジェクトファイル例

```
/* generated by MCS-Assembler tool ASM-MCS version 0.9.2 */
/* developed by Robert Bosch GmbH, Germany */
/* target architecture : mcs24-3 */

#ifndef OBJDEF_H_
#define OBJDEF_H_

#define OFFSET_MCS0_MEM     (   0) /* byte address offset for assembled code in array C-array
'mcs0_mem' */
#define SIZE_MCS0_MEM       ( 360) /* code size in bytes of assembled code in C-array 'mcs0_mem'
*/

#define LABEL_MCS0_MEM_ATOM0_CH2_SR0            (  69) /* Index into C-array 'mcs0_mem' for
assembler label 'ATOM0_CH2_SR0' */
#define LABEL_MCS0_MEM_ATOM0_ACB                (  64) /* Index into C-array 'mcs0_mem' for
assembler label 'ATOM0_ACB' */
#define LABEL_MCS0_MEM_TSK0_STACK               (   8) /* Index into C-array 'mcs0_mem' for
assembler label 'TSK0_STACK' */
#define LABEL_MCS0_MEM_TSK0_INIT                (  71) /* Index into C-array 'mcs0_mem' for
assembler label 'TSK0_INIT' */
#define LABEL_MCS0_MEM_ATOM0_CH0_SR0            (  65) /* Index into C-array 'mcs0_mem' for
assembler label 'ATOM0_CH0_SR0' */
#define LABEL_MCS0_MEM_ATOM0_CH0_SR1            (  66) /* Index into C-array 'mcs0_mem' for
assembler label 'ATOM0_CH0_SR1' */
#define LABEL_MCS0_MEM_ATOM0_CH1_SR0            (  67) /* Index into C-array 'mcs0_mem' for
assembler label 'ATOM0_CH1_SR0' */
#define LABEL_MCS0_MEM_ATOM0_CH1_SR1            (  68) /* Index into C-array 'mcs0_mem' for
assembler label 'ATOM0_CH1_SR1' */
#define LABEL_MCS0_MEM_ATOM0_CH2_SR1            (  70) /* Index into C-array 'mcs0_mem' for
assembler label 'ATOM0_CH2_SR1' */
#define LABEL_MCS0_MEM_TSK0_PWM_LOOP            (  74) /* Index into C-array 'mcs0_mem' for
assembler label 'TSK0_PWM_LOOP' */
#define LABEL_MCS0_MEM_TSK0_DONE                (  89) /* Index into C-array 'mcs0_mem' for
assembler label 'TSK0_DONE' */

#endif
```

図 7-4　MCS ラベルファイル例

これらの生成ファイルを U2A のプロジェクトに組み込むことで MCS プログラムが実行可能です。

7.5 実行方法

「7.4 アセンブル」にてアセンブルを行った MCS プログラムの実行方法は以下のとおりです。なお、このソースコードでは CMU や TBU の初期設定については省略しています。

```
#include "iodefine.h"

#include "objfile.h"        // MCS object file
#include "objdef.h"         // MCS label file

#define bool _Bool
#define size_t unsigned long
#define true 1
#define false 0

#define MCS0RAM   0xFF638000 //MCS0 RAM start address
#define MCS0WRADDR0  0x077   //MCS0 CH0 ARU address
#define MCS0WRADDR1  0x078   //MCS0 CH1 ARU address
#define MCS0WRADDR2  0x079   //MCS0 CH2 ARU address

#define PWM_CYCLE  120 //PWM cycle. //This count clock is CMU_CLK2,
                              //selected by ACB of ARU data which MCS sends to each ATOM channels.
#define DEAD_TIME_FALLING 200 //Dead Time count.
                              //This count clock is SYS_CLK,
                              //selected by CDTM[i]_DTM[j]_CTRL.CLK_SEL.
#define DEAD_TIME_RISING 180  //Dead Time count.
                              //This count clock is SYS_CLK,
                              //selected by CDTM[i]_DTM[j]_CTRL.CLK_SEL.

bool SetupAtom(unsigned long u32CN00, unsigned long u32CN01, unsigned long u32CN02);
bool EnableAtom( void );
bool SetSimpleDTM4(unsigned short u16Fall, unsigned short u16Rise);
bool SetPWM( unsigned long *mcs_ptr, unsigned long *data_ptr );
unsigned long *LoadMcs(unsigned long *u32McsPtr, size_t size);

static void gtm0_mcs0_ch0_main(void);
static void irq_gtm0_mcs0_ch0(void);
```

図 7-5 MCS プログラム使用例(1/4)

```
void gtm0_mcs0_ch0_main( void )
{
    unsigned long *mcs_ptr;
    unsigned long u32PwmData[6] = {
        PWM_CYCLE,      // ATOM CH0 cycle
        PWM_CYCLE / 2, // ATOM CH0 duty
        PWM_CYCLE,      // ATOM CH1 cycle
        PWM_CYCLE / 2, // ATOM CH1 duty
        PWM_CYCLE,      // ATOM CH2 cycle
        PWM_CYCLE / 2  // ATOM CH2 duty
    };

    mcs_ptr = LoadMcs( mcs0_mem, SIZE_MCS0_MEM );  // MCS object transfer

    GTM0.MCS0_CH0_IRQ_EN.UINT32 = 0x1;             // MCS0 CH0 MCS_IRQ is valid
    GTM0.MCS0_CH0_CTRL.UINT32 |= 0x1;              // MCS0 CH0 is valid

    SetSimpleDTM4(DEAD_TIME_FALLING, DEAD_TIME_RISING);
                           //  for CDTM0_DTM4 CH0 to CH2, setting
                           //   riging edge deadtime  = 200 clock count
                           //   falling edge deadtime = 180 clock count

    // ATOM setting
    //  initial setting counter of CH 0 to CH2.
    SetupAtom(PWM_CYCLE, PWM_CYCLE, PWM_CYCLE);

    EnableAtom();          // ATOM is enable,
                           // thereafter, setting cycle and duty of CH0 to CH2.

    while(false == SetPWM(mcs_ptr, u32PwmData)) {
    // waiting for synchronization with MCS0 CH0.
    }

    // If you want to change cycle and duty, plaese call SetPWM function.

}

// processing interrupt from MCS0 CH0.
void irq_gtm0_mcs0_ch0(void)
{
    GTM0.MCS0_CH0_IRQ_NOTIFY.UINT32 |= 0x01; // Clear MCS_IRQ.
    GTM0.ATOM0_AGC_GLB_CTRL.UINT32 = 0x002a0000; // update by shadow registers is valid.
}

// transfering MCS program to MCS RAM.
unsigned long *LoadMcs(unsigned long *u32McsPtr, size_t nSize)
{
    unsigned long *mcs_ptr;
    unsigned int i;

    mcs_ptr = (unsigned long *)MCSORAM; // get start address MCSORAM.
    for ( i = 0; i < nSize /4; i++ ){
        mcs_ptr[i] = u32McsPtr[i];
    }
    return mcs_ptr;
}
```

図 7-6　MCS プログラム使用例(2/4)

```
// setting DTM
bool SetSimpleDTM4(unsigned short u16Fall, unsigned short u16Rise)
{
    GTM0.CDTM0_DTM4_CTRL.UINT32 = 0x00000000; //DTM count clock is SYS_SLK
    GTM0.CDTM0_DTM4_CH_CTRL2.UINT32 = 0x888888;            // CH0 to CH2 of DTM4 is valid
    GTM0.CDTM0_DTM4_CH0_DTV.UINT32 = ((u16Fall << 16) | (u16Rise)); //  setting CH0 deadtime clock
    GTM0.CDTM0_DTM4_CH1_DTV.UINT32 = ((u16Fall << 16) | (u16Rise)); //  setting CH1 deadtime clock
    GTM0.CDTM0_DTM4_CH2_DTV.UINT32 = ((u16Fall << 16) | (u16Rise)); //  setting CH2 deadtime clock
    return true;
}

// setting cycle and duty
bool SetPWM( unsigned long *mcs_ptr, unsigned long *data_ptr )
{
    if( (GTM0.MCS0_STRG.UINT32 & 1) == 0 ){
        // if MCS0 CH0 doesn't wait for syncronization, false is returned.
        return false;
    }

    // setting cycle and duty of CH0 to CH2 in specific area of MCS-RAM.
    mcs_ptr[LABEL_MCS0_MEM_ATOM0_CH0_SR0] = *data_ptr;
    mcs_ptr[LABEL_MCS0_MEM_ATOM0_CH0_SR1] = *(data_ptr + 1);
    mcs_ptr[LABEL_MCS0_MEM_ATOM0_CH1_SR0] = *(data_ptr + 2);
    mcs_ptr[LABEL_MCS0_MEM_ATOM0_CH1_SR1] = *(data_ptr + 3);
    mcs_ptr[LABEL_MCS0_MEM_ATOM0_CH2_SR0] = *(data_ptr + 4);
    mcs_ptr[LABEL_MCS0_MEM_ATOM0_CH2_SR1] = *(data_ptr + 5);
    GTM0.MCS0_STRG.UINT32 |= 0x02; // release waiting for synchronization with MCS0 CH0.
    return true;
}
```

図 7-7　MCS プログラム使用例(3/4)

```
// ATOM0 Initial setting
bool SetupAtom(unsigned long u32CN00, unsigned long u32CN01, unsigned long u32CN02)
{
    // ATOM0 CH0 to CH3 is invalid.
    GTM0.ATOM0_AGC_OUTEN_STAT.UINT32 = 0x15;
    GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 = 0x15;
    while( 0x0 != GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 );

    // setting ARU address that ATOM0 reads ARU data from.
    // assigning ATOM CH0-2 to MCS0 CH0-2.
    GTM0.ATOM0_CH0_RDADDR.UINT32 = MCS0WRADDR0;
    GTM0.ATOM0_CH1_RDADDR.UINT32 = MCS0WRADDR1;
    GTM0.ATOM0_CH2_RDADDR.UINT32 = MCS0WRADDR2;

    // channel mode = SOMP and ARU valid.
    GTM0.ATOM0_CH0_CTRL.UINT32 = 0xa;
    GTM0.ATOM0_CH1_CTRL.UINT32 = 0xa;
    GTM0.ATOM0_CH2_CTRL.UINT32 = 0xa;

    // initialize counter
    GTM0.ATOM0_CH0_CN0.UINT32 = u32CN00;
    GTM0.ATOM0_CH1_CN0.UINT32 = u32CN01;
    GTM0.ATOM0_CH2_CN0.UINT32 = u32CN02;

    return true;
}

// ATOM0 is valid.
bool EnableAtom( void )
{
    /* force ATOM0 CH0 to 2 to valid.  */
    GTM0.ATOM0_AGC_FUPD_CTRL.UINT32 = 0x2a;

    /* when forced update is requested, ATOM0 CH0-2 will validate. */
    GTM0.ATOM0_AGC_OUTEN_CTRL.UINT32 = 0x2a;
    GTM0.ATOM0_AGC_ENDIS_CTRL.UINT32 = 0x2a;

    GTM0.ATOM0_AGC_GLB_CTRL.UINT32 = 0x1; // request force-update

    while( 0x3f != GTM0.ATOM0_AGC_ENDIS_STAT.UINT32 ); //waiting that ATOM CH0-2 validate.

    return true;
}
```

図 7-8　MCS プログラム使用例(4/4)

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
0.10	2019.08.30	全項	初版作成。
0.50	2019.10.18	全項	
0.70	2020.03.31	68,109	図 6-2、図 6-5 を修正
		73,79,91	表 6-3、表 6-9、表 6-17 の設定項目追加。
		103,108	6.5.2.1、6.5.5.1 のビット名誤記を修正。
		125	6.7.1 の説明を修正。
		125,128,129	図 6-9、図 6-12、図 6-13 の ATOM チャネル開始処理追加。
		131	図 6-15 の不要な処理削除。
1.00	2020.10.06	1	要旨の対象デバイスを修正。
1.10	2022.04.01	1	対象統合開発環境のデバイスファイルを追加。
		141 142	図 7-2、図 7-3 を更新

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。