TRAVEO™ T2G ファミリのトリガマルチプレクサ使用方法

本書について

適用範囲と目的

このアプリケーションノートは TRAVEO™ T2G ファミリのマルチプレクサについて説明します。トリガ信号をソースペリフェラルからデスティネーションに接続する方法について説明します。

関連製品ファミリ

TRAVEO™ T2G ファミリ

目次

1 はじめに

TRAVEO™ T2G デバイスのすべてのペリフェラルは、トリガ信号によって相互接続されます。トリガ信号は、ペリフェラルのイベントの発生や異なる状態への遷移を通知する手段です。トリガ信号は、他のペリフェラルでアクションを開始するために使用します。例えば、トリガは DMA を介したデータ転送 (TCPWM タイマによる P-DMA 転送のトリガを参照)、SAR ADC での変換 (単一の TCPWM タイマによる 3 つの SAR での同時 ADC 変換および TCPWM タイマによる ADC 変換のトリガを参照)、またはタイマを開始 (SW トリガによる TCPWM タイマの同時開始を参照) できます。トリガマルチプレクサは、これらのトリガ信号をソースペリフェラルから目的のデスティネーションに接続するように設計されたシンプルなマルチプレクサです。

このアプリケーションノートでは、ソースペリフェラルから目的のデスティネーションへのトリガ接続をする設定方法について説明します。

このアプリケーションノートで使用されている機能と用語の詳細については、アーキテクチャ テクニカル リファレンス マニュアル (TRM) の「Trigger Multiplexer」章を参照してください。

2 トリガマルチプレクサの概要

トリガ入力は、ソースペリフェラルからの出力信号です。トリガ出力は通常、デスティネーションペリフェラルへの入力信号です。トリガマルチプレクサは、トリガ入力とトリガ出力を相互接続します。

トリガマルチプレクサは 2 つグループタイプがあります。

- マルチプレクサベースのグループタイプ (グループトリガ)
  - 1 つペリフェラルの入力トリガを複数のペリフェラルの出力トリガに接続します
- 1-to-1 ベースのグループタイプ (1-to-1 トリガ)
  - 1 つペリフェラル入力トリガを特定の出力トリガに接続します

各グループタイプは、複数のトリガグループ (最大 16) で構成されます。各グループタイプは、特定のペリフェラルのトリガ入力に関連付けられています。図 1 にトリガマルチプレクサのブロックダイヤグラムを示します。

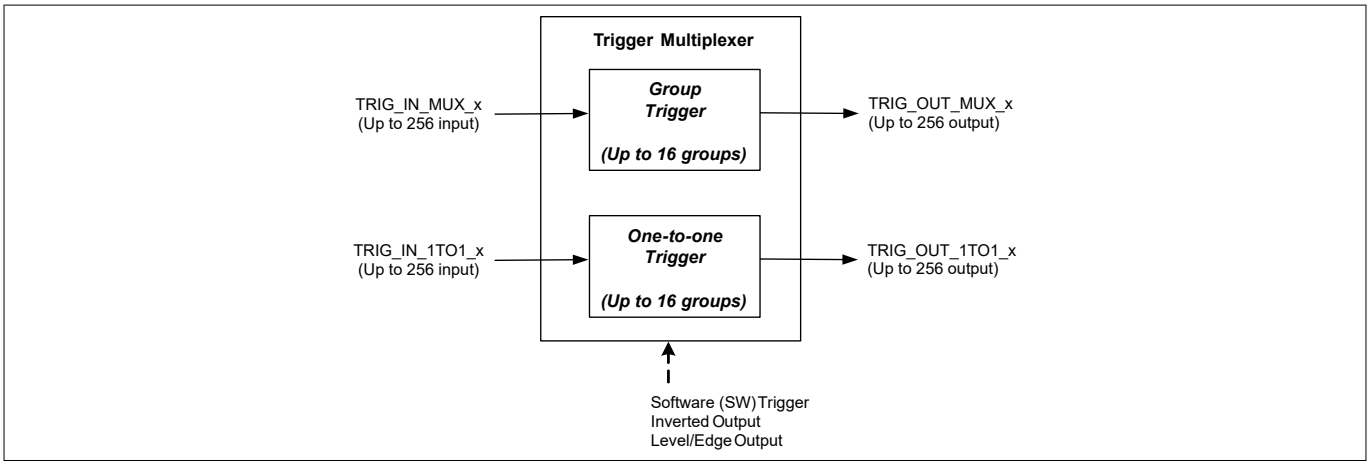

図 1 トリガマルチプレクサのブロックダイヤグラム

トリガマルチプレクサには次の入力信号と出力信号があります。

- TRIG_IN_MUX_x は、グループトリガの入力トリガです。最大 256 の入力信号があります。
- TRIG_IN_1TO1_x は、1-to-1 トリガの入力トリガです。最大 256 の入力信号があります。
- TRIG_OUT_MUX_x は、グループトリガの出力トリガです。最大 256 の出力信号があります。
- TRIG_OUT_1TO1_x は、1-to-1 トリガの出力トリガです。最大 256 の出力信号があります。

トリガマルチプレクサには SW トリガ, 反転出力トリガ, およびレベルまたはエッジセンシティブトリガの機能があります。

SW トリガは SW によって開始され、トリガマルチプレクサで任意の信号を起動できます。反転出力は、出力信号の極性を指定します。レベルトリガまたはエッジトリガは、出力トリガをレベルセンシティブトリガまたはエッジセンシティブトリガとして扱うかどうかを指定します。

サフィックス「x」は、ペリフェラルブロックの名前を表します。表 1 は相互接続可能なグループトリガの出力トリガと入力トリガ、および TRAVEO™ T2G デバイスシリーズで接続の可否を示します。ただし、一部の組合せは、一部のユニットおよびチャネル番号にルーティングできません。各ペリフェラルのユニットとチャネルの詳細については、データシートを参照してください。

表 1 シリーズによるグループトリガの出力トリガと入力トリガ間の接続

TRIG_OUT_MUX_x	TRIG_IN_MUX_x	CYT2B7	CYT2B9	CYT4BF	CYT4DN
PDMA	PDMA	✓	✓	✓	✓
	MDMA	✓	✓	✓	✓
	FAULT	✓	✓	✓	✓

(続く)

2 トリガマルチプレクサの概要

表 1 (続き) シリーズによるグループトリガの出力トリガと入力トリガ間の接続

TRIG_OUT_MUX_x	TRIG_IN_MUX_x	CYT2B7	CYT2B9	CYT4BF	CYT4DN
	CTI	✓	✓	✓	✓
	EVTGEN	✓	✓	✓	✓
	HSIOM	✓	✓	✓	✓
	TCPWM	✓	✓	✓	✓
	CAN0_TT	✓	✓	✓	
	CAN1_TT	✓	✓	✓	
	PASS_GEN	✓	✓	✓	✓
	FLEXRAY_TT			✓	
MDMA	MDMA	✓	✓		
	TCPWM			✓	✓
TCPWM	TCPWM	✓	✓	✓	✓
	CAN_TT	✓	✓		✓
	PDMA	✓	✓	✓	✓
	MDMA	✓	✓	✓	✓
	CTI	✓	✓	✓	✓
	FAULT	✓	✓	✓	✓
	PASS_GEN	✓	✓	✓	✓
	HSIOM	✓	✓	✓	✓
	SCB	✓	✓	✓	✓
	SCB_I2C_SCL	✓	✓	✓	✓
	CAN_DBG	✓	✓	✓	✓
	CAN_FIFO	✓	✓	✓	✓
	CXPI		✓		✓
	EVTGEN	✓	✓	✓	✓
	SMIF			✓	✓
	I2S			✓	✓
	TDM				✓
	SG				✓
	PWM				✓
	MIXER				✓
	AUDIODAC				✓
	FLEXRAY_TT			✓	
	FLEXRAY_IBUF			✓	

(続く)

表 1 (続き) シリーズによるグループトリガの出力トリガと入力トリガ間の接続

TRIG_OUT_MUX_x	TRIG_IN_MUX_x	CYT2B7	CYT2B9	CYT4BF	CYT4DN
	FLEXRAY_OBUF			✓	
PASS_GEN	PDMA	✓	✓	✓	✓
	CTI	✓	✓		
	FAULT	✓	✓		
	EVTGEN	✓	✓	✓	✓
	PASS_GEN	✓	✓		
	HSIOM	✓	✓	✓	✓
	TCPWM	✓	✓	✓	✓
CAN_TT	CAN_TT	✓	✓	✓	✓
	FLEXRAY_TT			✓	
FLEXRAY_TT	CAN_TT			✓	
	FLEXRAY_TT			✓	
HSIOM PERI_DEBUG_FREEZE PASS_DEBUG_FREEZE SRSS_WDT_DEBUG_ FREEZE SRSS_MCWDT_DEBUG_FRE EZE TCPWM_DEBUG_ FREEZE	TR_GROUP[i]_OUT PUT	✓	✓	✓	✓
I2S_DEBUG_FREEZE TDM_DEBUG_FREEZE SG_DEBUG_FREEZE PWM_DEBUG_FREEZE PWM_DEBUG_FREEZE MIXER_DEBUG_FREEZE AUDIODAC_DEBUG_ FREEZE	TR_GROUP[i]_OUT PUT				✓
TR_GROUP[i]_INPUT	PDMA	✓	✓	✓	✓
	SCB	✓	✓	✓	✓
	SCB_I2C_SCL	✓	✓	✓	✓
	CAN_DBG	✓	✓	✓	✓
	CAN_FIFO	✓	✓	✓	✓
	CAN_TT	✓	✓	✓	✓
	CTI	✓	✓	✓	✓

(続く)

表 1 (続き) シリーズによるグループトリガの出力トリガと入力トリガ間の接続

TRIG_OUT_MUX_x	TRIG_IN_MUX_x	CYT2B7	CYT2B9	CYT4BF	CYT4DN
	FAULT	✓	✓	✓	✓
	TCPWM	✓	✓	✓	✓
	MDMA	✓	✓	✓	✓
	PASS_GEN	✓	✓	✓	✓
	EVTGEN	✓	✓	✓	✓
	CXPI		✓		✓
	SMIF			✓	✓
	FLEXRAY_TT			✓	
	FLEXRAY_IBUF			✓	
	FLEXRAY_OBUF			✓	
	I2S			✓	✓
	HSIOM			✓	✓
	TDM				✓
	SG				✓
	PWM				✓
	AUDIODAC				✓
	MIXER				✓

1-to-1 のグループトリガでは、1 つの入力トリガが特定の出力トリガに直接接続されます。1-to-1 のトリガの接続については、データシートを参照してください。

2.1 グループトリガ

グループトリガの場合、入力トリガ TRIG_IN_MUX_x が各出力トリガ TRIG_OUT_MUX_x に対して選択されます。グループ「i」のすべての出力トリガが同じ入力トリガを共有することに注意してください。図 2 にグループトリガのブロックダイヤグラムを示します。

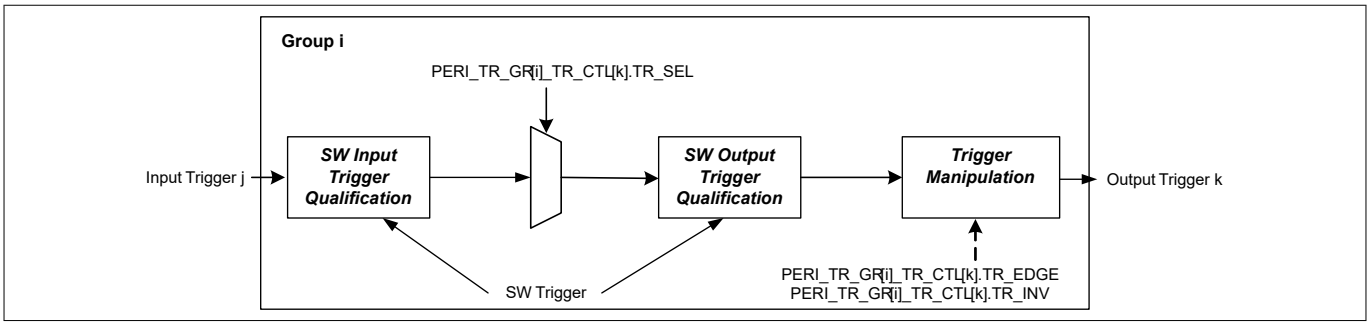

図 2 グループトリガ

ソースペリフェラルからの複数の出力信号から 1 つが選択されます。トリガグループ内の特定の出力トリガについては、PERI_TR_GR[i]_TR_CTL[k]レジスタを介して入力トリガを指定できます。ここで、サフィックス「i」と「k」は、それぞれトリガグループ番号と出力トリガ番号を示します。入力トリガ番号は、

2 トリガマルチプレクサの概要

PERI_TR_GR[i]_TR_CTL[k].TR_SEL ＝ j で指定できます。ここでは、グループトリガのいくつかの例について説明します。

例 1:

この例は、SCB RX パルスを受信したときの 16 ビット TCPWM カウンタ開始のトリガを示します。このトリガは、マルチプレクサ (MUX) グループ 5 に含まれます。この例は、トリガグループ, 入力トリガ, および出力トリガを提供する CYT2B7 シリーズのデータシートからの抜粋です。表 2 と 表 3 に、CYT2B7 シリーズのデータシートから抜粋した MUX グループ 5 のトリガ出力とトリガ入力を示します。

表 2　CYT2B7 シリーズの MUX グループ 5 のトリガ出力

出力 (「k」)	トリガラベル	説明
MUX グループ 5: TCPWM_IN (TCPWM0 トリガマルチプレクサ)		
0:10	TCPWM_ALL_CNT_TR_IN	TCPWM0 へのトリガ

表 3　CYT2B7 シリーズの MUX グループ 5 のトリガ入力

入力 (「j」)	トリガラベル	説明
MUX グループ 5: TCPWM_IN (TCPWM0 トリガマルチプレクサ)		
1:16	PDMA0_TR_OUT[0:15]	汎用 P-DMA0 トリガ
17:24	PDMA1_TR_OUT[0:7]	汎用 P-DMA1 トリガ
25:28	MDMA_TR_OUT[0:3]	M-DMA0 トリガ
29:30	CTI_TR_OUT[0:1]	トレースイベント
31:34	FAULT_TR_OUT[0:3]	フォルトイベント
35:40	PASS_GEN_TR_OUT[0:5]	PASS SAR イベント
41:72	HSIOM_IO_INPUT[0:31]	I/O 入力
73	SCB_TX_TR_OUT[0]	SCB0 TX トリガ
74	**SCB_RX_TR_OUT[0]**	**SCB0 RX トリガ**
75	SCB_I2C_SCL_TR_OUT[0]	SCB0 I2C トリガ
76	SCB_TX_TR_OUT[1]	SCB1 TX トリガ
77	SCB_RX_TR_OUT[1]	SCB1 RX トリガ
78	SCB_I2C_SCL_TR_OUT[1]	SCB1 I2C トリガ
79	SCB_TX_TR_OUT[2]	SCB2 TX トリガ
80	SCB_RX_TR_OUT[2]	SCB2 RX トリガ
81	SCB_I2C_SCL_TR_OUT[2]	SCB2 I2C トリガ
82	SCB_TX_TR_OUT[3]	SCB3 TX トリガ
83	SCB_RX_TR_OUT[3]	SCB3 RX トリガ
84	SCB_I2C_SCL_TR_OUT[3]	SCB3 I2C トリガ
85	SCB_TX_TR_OUT[4]	SCB4 TX トリガ
86	SCB_RX_TR_OUT[4]	SCB4 RX トリガ
87	SCB_I2C_SCL_TR_OUT[4]	SCB4 I2C トリガ

(続く)

表 3 (続き) CYT2B7 シリーズの MUX グループ 5 のトリガ入力

入力 (「j」)	トリガラベル	説明
88	SCB_TX_TR_OUT[5]	SCB5 TX トリガ
89	SCB_RX_TR_OUT[5]	SCB5 RX トリガ
90	SCB_I2C_SCL_TR_OUT[5]	SCB5 I2C トリガ
91	SCB_TX_TR_OUT[6]	SCB6 TX トリガ
92	SCB_RX_TR_OUT[6]	SCB6 RX トリガ
93	SCB_I2C_SCL_TR_OUT[6]	SCB6 I2C トリガ
94	SCB_TX_TR_OUT[7]	SCB7 TX トリガ
95	SCB_RX_TR_OUT[7]	SCB7 RX トリガ
96	SCB_I2C_SCL_TR_OUT[7]	SCB7 I2C トリガ
97:99	CAN0_DBG_TR_OUT[0:2]	CAN0 M-DMA0 イベント
100:102	CAN0_FIFO0_TR_OUT[0:2]	CAN0 FIFO0 イベント
103:105	CAN0_FIFO1_TR_OUT[0:2]	CAN0 FIFO1 イベント
106:108	CAN1_DBG_TR_OUT[0:2]	CAN1 M-DMA0 イベント
109:111	CAN1_FIFO0_TR_OUT[0:2]	CAN1 FIFO0 イベント
112:114	CAN1_FIFO1_TR_OUT[0:2]	CAN1 FIFO1 イベント
115:122	EVTGEN_TR_OUT[3:10]	EVTGEN トリガ

図 3 に、入力トリガと出力トリガの構造を示します。

2 トリガマルチプレクサの概要

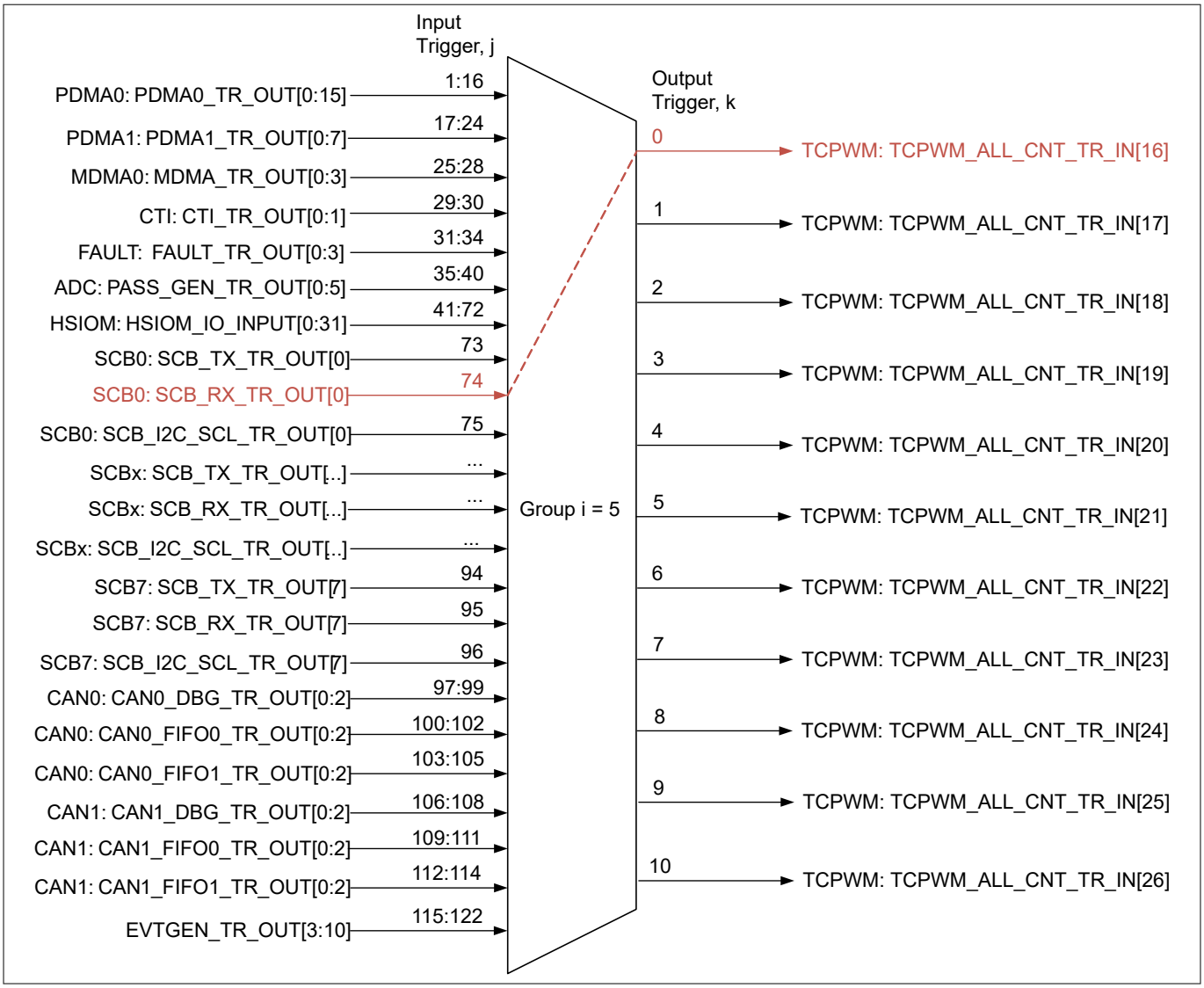

図 3　　CYT2B7 シリーズの 16 ビット TCPWM と SCB0 RX 間のトリガ接続

次に、SCB0 RX と 16 ビット TCPWM チャネル 0 を接続する方法について説明します。

1. 設定レジスタの選択
   MUX グループ 5 (i ＝ 5)
   出力トリガ TCPWM_ALL_CNT_TR_IN[16] (k ＝ 0)
   したがって、レジスタは PERI_TR_GR**5**_TR_CTL**0** として指定されます。
2. 入力トリガの選択
   入力トリガ SCB_RX_TR_OUT[0] (j ＝ 74)
   構成は PERI_TR_GR5_TR_CTL0.TR_SEL = **74** になります。

例 2:

ADC チャネルでの SAR ADC ユニットのアクティブ化は、イベントジェネレータによってトリガできます。汎用トリガ入力は ADC チャネル間で共有されます。5 つの汎用トリガの中で 1 つを任意の ADC チャネルに接続できます。SAR ADC のグループトリガは、通常、複数の SAR ADC ユニットのトリガ制御を行います。SAR ADC の 1-to-1 トリガは、通常、SAR ADC ユニット内の ADC 論理チャネルのトリガ制御に役立ちます。SAR ADC 汎用トリガ入力の詳細については、アーキテクチャ TRM の「SAR ADC」章を参照してください。

2 トリガマルチプレクサの概要

例えば、CYT2B7 シリーズでは、入力トリガイベントジェネレータ 0 が SAR の汎用トリガ 0 に接続されます。このトリガ接続は、MUX グループ 6 に含まれます。この例は、表 4 および表 5 に基づいています。これらは、トリガグループ, 入力トリガ, および出力トリガを提供する CYT2B7 シリーズのデータシートからの抜粋です。

表 4　　CYT2B7 シリーズの MUX グループ 6 のトリガ出力

出力 (「k」)	トリガラベル	説明
MUX グループ 6: PASS (PASS SAR トリガマルチプレクサ)		
0:11	PASS_GEN_TR_IN[0:11]	SAR ADC への汎用トリガ

表 5　　CYT2B7 シリーズの MUX グループ 6 のトリガ入力

入力 (「j」)	トリガラベル	説明
MUX グループ 6: PASS (PASS SAR トリガマルチプレクサ)		
1:16	PDMA0_TR_OUT[0:15]	汎用 P-DMA0 トリガ
17:18	CTI_TR_OUT[0:1]	トレースイベント
19:22	FAULT_TR_OUT[0:3]	フォルトイベント
23:25	**EVTGEN_TR_OUT[0:2]**	**EVTGEN トリガ**
26:31	PASS_GEN_TR_OUT[0:5]	PASS SAR 完了信号
32:63	HSIOM_IO_INPUT[0:31]	I/O 入力
64:67	TCPWM_32_TR_OUT1[0:3]	32 ビット TCPWM0 カウンタ
68:79	TCPWM_16M_TR_OUT1[0:11]	16 ビットモータ拡張 TCPWM0 カウンタ

各シリーズのトリガグループ入力およびトリガグループ出力の表については、デバイスのデータシートを参照してください。

図 4 に、入力トリガと出力トリガの構造を示します。

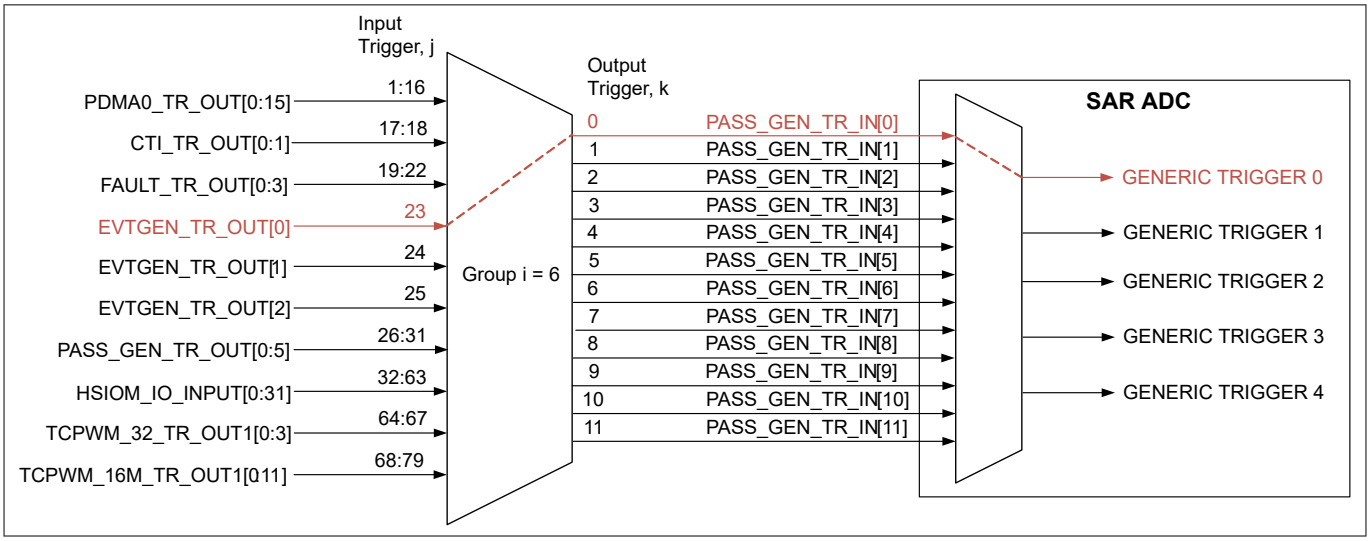

図 4　　CYT2B7 シリーズの ADC における TCPWM 16 ビットモータと汎用トリガ 2 間のトリガ接続

次に、トリガソースのイベントジェネレータ 0 が SAR0 の汎用トリガ 0 に接続する方法を説明します。

1. 設定レジスタの選択
   MUX グループ 6 (i ＝ 6)
   出力トリガ PASS_GEN_TR_IN[0] (k ＝ 0)

したがって、レジスタは PERI_TR_GR**6**_TR_CTL**0** として指定されます。

2. 入力トリガの選択

入力トリガ EVTGEN_TR_OUT[0] (j ＝ 23)

設定は PERI_TR_GR6_TR_CTL0.TR_SEL = **23** になります。

3. 汎用トリガの選択

汎用トリガ SAR_TR_IN_SEL0.IN0_SEL = **0**。

表 6 に、反転出力トリガとレベルまたはエッジセンシティブトリガを説明するグループトリガのビットレジスタを示します。詳細については、レジスタ TRM を参照してください。

表 6　　グループトリガ制御ビットレジスタ

ビットレジスタ	説明
PERI_TR_GR[i]_TR_CTL[k].TR_INV	出力トリガを反転するかどうかを指定します。 ‘0’: 反転しない ‘1’: 反転する
PERI_TR_GR[i]_TR_CTL[k].TR_EDGE	（反転）出力トリガをレベルセンシティブトリガまたはエッジセンシティブトリガとして扱うかどうかを指定します。 ‘0’: レベルセンシティブ ‘1’: エッジセンシティブトリガ

2.2　1-to-1 トリガ

1-to-1 のトリガの場合、入力トリガ TRIG_IN_1TO1_x が出力トリガ TRIG_OUT_1TO1_x に接続されます。1-to-1 のグループには、入力トリガを無効にする AND ゲート機能があります。図 5 に、1-to-1 のトリガのブロックダイヤグラムを示します。

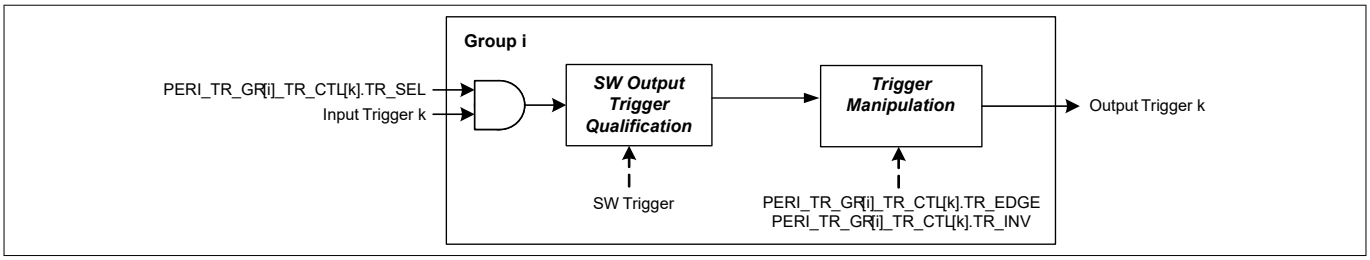

図 5　　1-to-1 トリガ

1-to-1 のグループトリガは、PERI_TR_1TO1_GR[i]_TR_CTL[k]レジスタで指定できます。ここで、サフィックス「i」と「k」は、それぞれトリガグループ番号と出力トリガ番号を示します。

PERI_TR_1TO1_GR[i]_TR_CTL[k].TR_SEL は、入力トリガの有効化/無効化を制御するために使用されます。「1」に設定すると、入力トリガが有効になります。「0」に設定すると、入力トリガが無効になり、一定の信号レベル「0」に設定されます。

ここでは、1-to-1 トリガの例について説明します。

例:

これは、SCB UART チャネル 3 によって P-DMA1 チャネル 15 でデータ送信を起動する例です。このトリガ接続は、MUX グループ 8 に含まれます。この例は、トリガグループと入力トリガを提供する CYT2B7 シリーズのデータシートから抜粋した表 7 に基づいています。

2 トリガマルチプレクサの概要

表 7　　CYT2B7 シリーズの 1-to-1 MUX グループ 8 のトリガ

入力トリガ (「k」)	トリガ入力	トリガ出力
MUX グループ 8: SCB から P-DMA1 へのトリガ		
0	SCB0_TX_TR_OUT	PDMA1_TR_IN[8]
1	SCB0_RX_TR_OUT	PDMA1_TR_IN[9]
2	SCB1_TX_TR_OUT	PDMA1_TR_IN[10]
3	SCB1_RX_TR_OUT	PDMA1_TR_IN[11]
4	SCB2_TX_TR_OUT	PDMA1_TR_IN[12]
5	SCB2_RX_TR_OUT	PDMA1_TR_IN13]
6	SCB3_TX_TR_OUT	PDMA1_TR_IN[14]
7	**SCB3_RX_TR_OUT**	**PDMA1_TR_IN[15]**
8	SCB4_TX_TR_OUT	PDMA1_TR_IN[16]
9	SCB4_RX_TR_OUT	PDMA1_TR_IN[17]
10	SCB5_TX_TR_OUT	PDMA1_TR_IN[18]
11	SCB5_RX_TR_OUT	PDMA1_TR_IN[19]
12	SCB6_TX_TR_OUT	PDMA1_TR_IN[20]
13	SCB6_RX_TR_OUT	PDMA1_TR_IN[21]
14	SCB7_TX_TR_OUT	PDMA1_TR_IN[22]
15	SCB7_RX_TR_OUT	PDMA1_TR_IN[23]

各シリーズのトリガ 1-to-1 の表については、デバイスのデータシートを参照してください。

以下では、SCB チャネル 3 を P-DMA1 チャネル 15 に接続する方法について説明します。1-to-1 のグループトリガでは、1 つの入力トリガが特定の出力トリガに直接接続されます。したがって、出力トリガを示すには、入力トリガ番号のみを指定するだけで十分です。

1. 設定レジスタの選択
   MUX グループ 8 (i ＝ 8)
   入力トリガ SCB3_RX_TR_OUT が出力トリガ PDMA1_TR_IN[15] (k ＝ 7) に直接接続します。
   したがって、レジスタは PERI_TR_1TO1_GR**8**_TR_CTL**7** として指定されます。
2. 入力トリガの設定: 基本的に、1-to-1 トリガの入力トリガが有効になります。設定は PERI_TR_1TO1_GR8_TR_CTL7.TR_SEL = **1** になります。

表 8 に、反転出力トリガとレベルまたはエッジセンシティブトリガを説明する 1-to-1 トリガのビットレジスタを示します。詳細については、レジスタ TRM を参照してください。

表 8　　1-to-1 トリガ制御ビットレジスタ

ビットレジスタ	説明
PERI_TR_1TO1_GR[i]_TR_CTL[k].TR_INV	出力トリガを反転するかどうかを指定します。 '0' : 反転しない '1' : 反転する

(続く)

表 8 (続き) 1-to-1 トリガ制御ビットレジスタ

ビットレジスタ	説明
PERI_TR_1TO1_GR[i]_TR_CTL[k].TR_EDGE	(反転)出力トリガをレベルセンシティブトリガまたはエッジセンシティブトリガとして扱うかどうかを指定します。 '0':レベルセンシティブ '1':エッジセンシティブトリガ

2.3 ソフトウェアトリガ

各グループはソフトウェアトリガをサポートします。グループトリガの場合、SW トリガには入力トリガまたは出力トリガのいずれかを選択できます。1-to-1 のトリガの場合、SW トリガとして出力トリガを選択できます。図 6 に、SW トリガを使用したグループトリガのブロックダイヤグラムを示します。図 7 に、1-to-1 トリガと SW トリガのブロックダイヤグラムを示します。

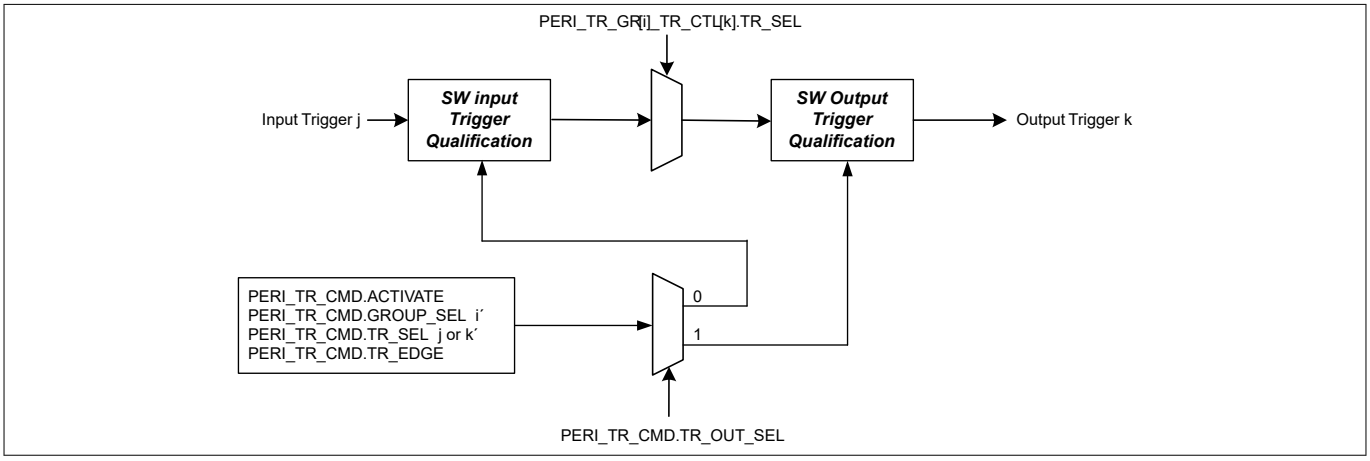

図 6 グループトリガの SW トリガ

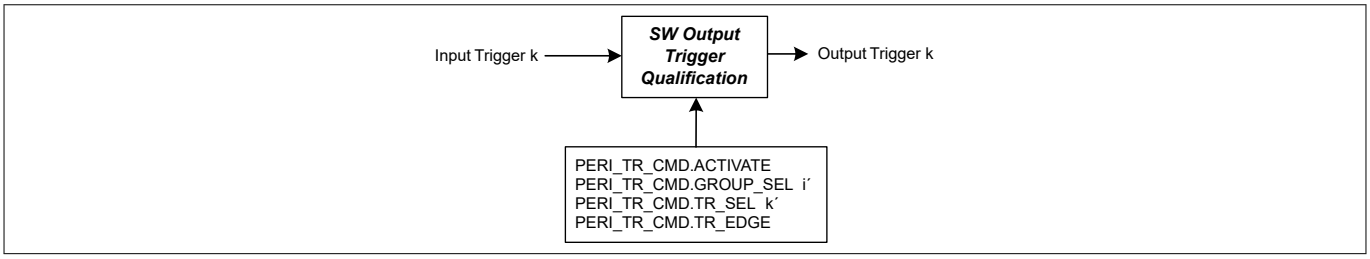

図 7 1-to-1 の SW トリガ

ソフトウェアによるトリガのアクティブ化は、PERI_TR_CMD を介して行えます。このレジスタは、SW によるトリガやデバッグに便利です。グループトリガの場合、アクティブ化されたトリガを入力トリガ (= j) または出力トリガ (= k) として指定するには、PERI_TR_CMD.OUT_SEL を構成します。「0」に設定すると、入力トリガが選択されます。「1」に設定すると、出力トリガが選択されます。この構成は 1-to-1 のトリガには使用できないことに注意してください。起動されるトリガ番号は、PERI_TR_CMD.TR_SEL=j または k で指定できます。トリガグループ番号は、PERI_TR_CMD.GROUP_SEL=i で指定できます。アクティブ化されたトリガのレベルまたはエッジセンシティブトリガは、PERI_TR_CMD.TR_EDGE で指定できます。PERI_TR_CMD.ACTIVATE を「1」に設定すると、トリガをアクティブ化できます。PERI_TR_CMD.ACTIVATE は、他の PERI_TR_CMD フィールドと一緒に設定できません。ここでは、ソフトウェアトリガの例について説明します。

例:

2 トリガマルチプレクサの概要

これは、P-DMA0 チャネル 0 でデータ転送を起動する例です。このトリガ接続は、グループトリガの MUX グループ 0 に含まれ、アクティブ化されたトリガとして出力トリガを指定します。表 9 に PERI_TR_CMD.GROUP_SEL (= i) および PERI_TR_CMD.TR_SEL (= k) に設定する必要がある値を示します。CYT2B7 シリーズのデータシートから抜粋した表 9 に、トリガグループと出力トリガを示します。

表 9　　CYT2B7 シリーズの MUX グループ 0 のトリガ出力

出力 (「k」)	トリガラベル	説明
MUX グループ 0: PDMA0_TR (P-DMA0 トリガマルチプレクサ)		
0:7	PDMA0_TR_IN[0:7]	P-DMA0[0:7]へのトリガ

各シリーズのトリガグループ出力の表については、デバイスのデータシートを参照してください。
図 8 に、SW による P-DMA0 でのデータ転送のトリガを示します。

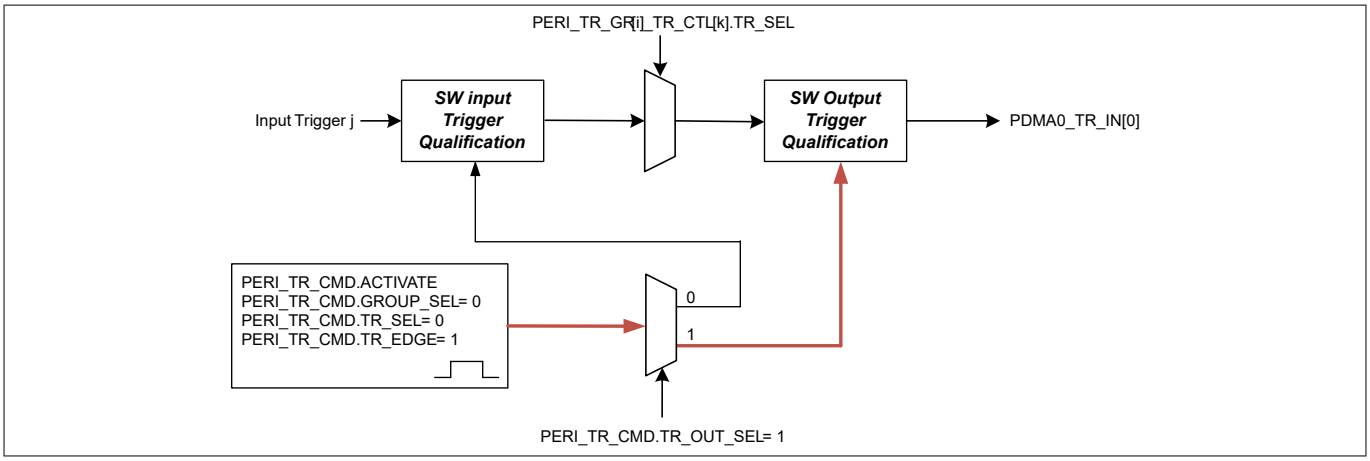

図 8　　SW による P-DMA0 データ転送の開始

次に、P-DMA0 でデータ転送を開始する方法について説明します。

1. アクティブ化されたトリガの選択
   PERI_TR_CMD.TR_SEL = **0**: 出力トリガ PDMA0_TR_IN[0] (k = 0)
   PERI_TR_CMD.GROUP_SEL = **0**: グループトリガの MUX グループ 0 (i = 0)
   PERI_TR_CMD.TR_EDGE = **1**: エッジセンシティブトリガ
2. アクティブ化されたトリガの指定
   PERI_TR_CMD.OUT_SEL = **1**: 出力トリガ
3. SW トリガをアクティブ化します。
   PERI_TR_CMD.ACTIVATE = **1**

各シリーズのトリガグループ入力およびトリガグループ出力の詳細については、デバイスのデータシートを参照してください。

3 アプリケーション

ここでは、グループトリガ, 1-to-1 トリガ, および SW トリガの使用例について説明します。

- TCPWM タイマによる P-DMA 転送のトリガ: グループトリガの使用例
- 単一の TCPWM タイマによる 3 つの SAR での同時 ADC 変換: グループトリガの使用例
- TCPWM タイマによる ADC 変換のトリガ: 1-to-1 の使用例
- SW トリガによる TCPWM タイマの同時開始: SW トリガの使用例

3.1 TCPWM タイマによる P-DMA 転送のトリガ

ここでは、グループトリガの入力トリガが P-DMA0 チャネルのデータ転送を開始する方法について説明します。

3.1.1 TCPWM タイマによる P-DMA 転送のトリガの使用例の説明

図 9 に、CYT2B7 シリーズのグループトリガのアプリケーション例を示します。P-DMA0 チャネル 9 のデータ転送は、グループトリガマルチプレクサグループ 3 の TCPWM 16 ビットチャネル 0 によってトリガされます。TCPWM カウンタのオーバーフローイベントは、P-DMA0 チャネル 9 をトリガして、SRAM のソースバッファから SRAM のデスティネーションバッファへのデータ転送を開始します。図 10 に動作を示します。TCPWM および DMA コンポーネントの詳細については、アーキテクチャ TRM の「TCPWM」および「Direct Memory Access」章を参照してください。

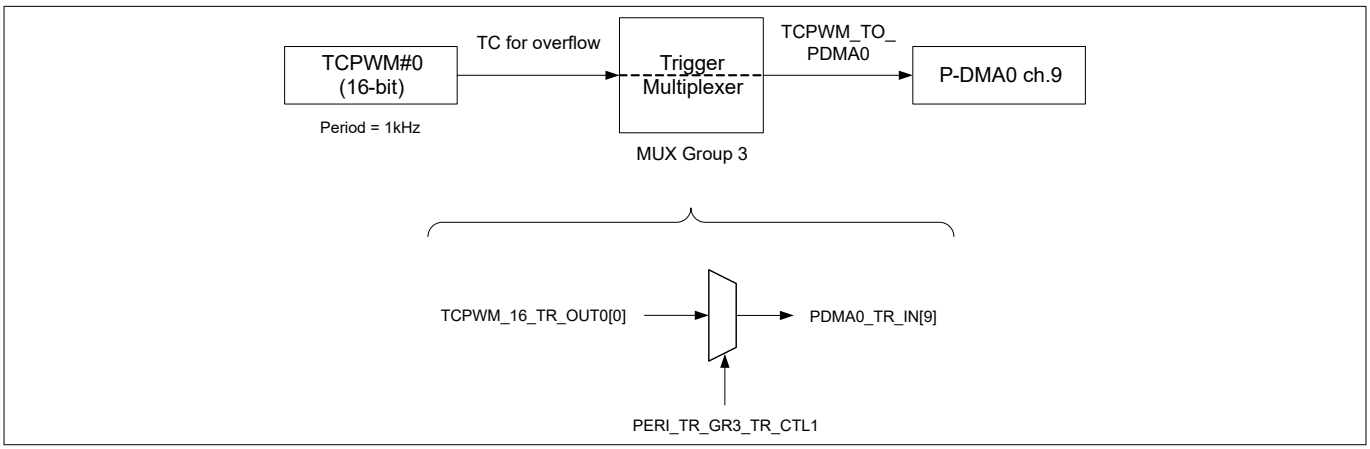

図 9 CYT2B7 シリーズのグループトリガの例

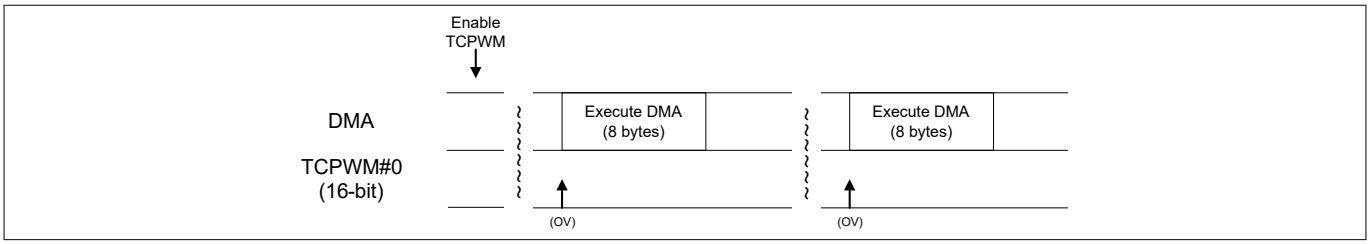

図 10 動作例: TCPWM カウンタのオーバーフローによる P-DMA 転送のトリガ

このアプリケーションを実装するには、トリガを設定する必要があります。図 11 に、トリガの設定手順を示します。図 11 は、TCPWM カウンタと P-DMA0 チャネルの設定も示します。

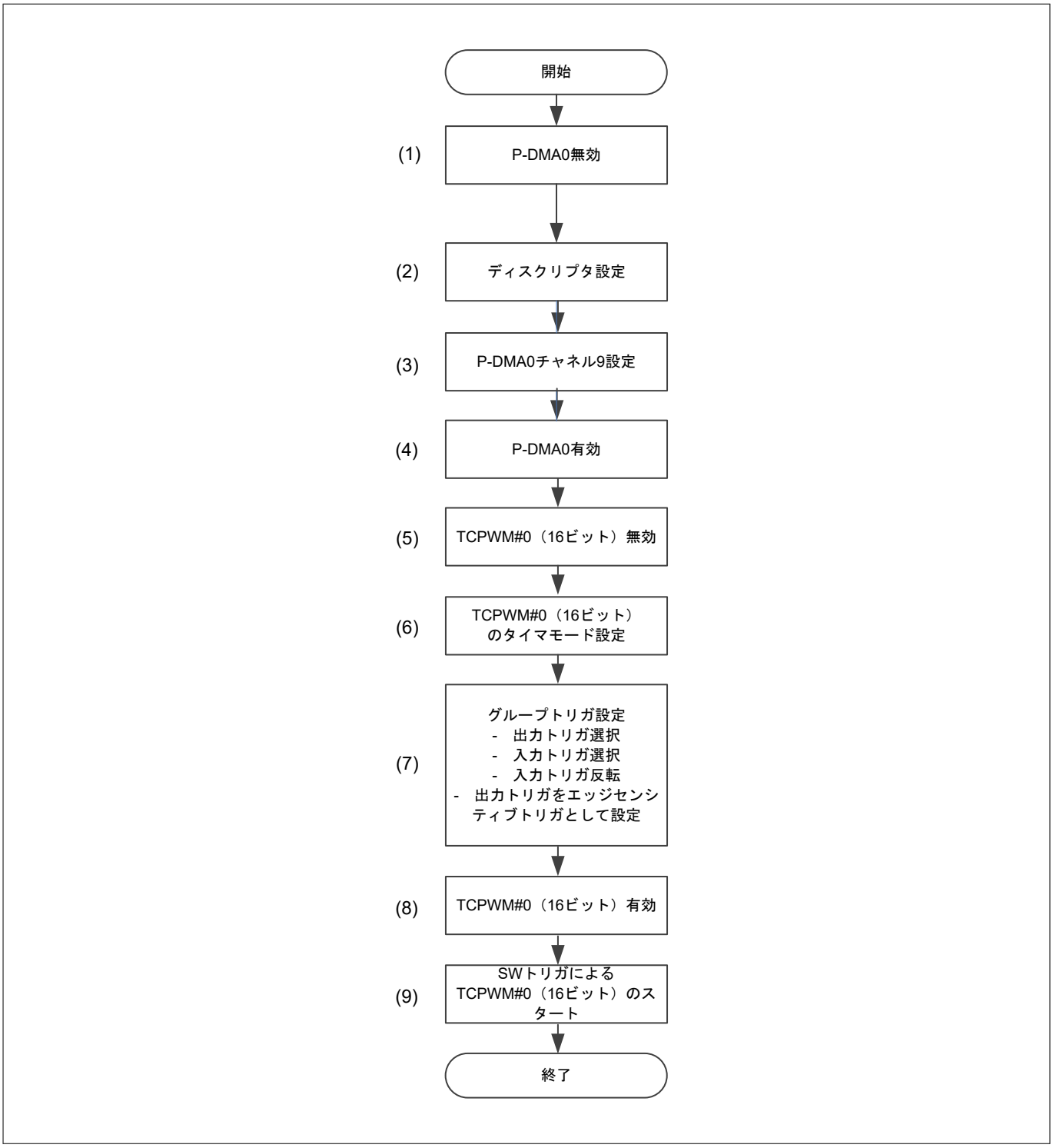

図 11　　CYT2B7 シリーズのグループトリガの設定例

ステップ 7 のグループトリガの出力トリガと入力トリガの構成は次のとおりです。

トリガマルチプレクサの設定

表 10 と表 11 に、設定する値を示します。これらの表は、トリガグループ, 入力トリガ, および出力トリガを提供する CYT2B7 シリーズデータシートからの抜粋です。

表 10　　CYT2B7 シリーズの MUX グループ 3 のトリガ出力

出力 (「k」)	トリガラベル	説明
MUX グループ 3: TCPWM_TO_PDMA0 (TCPWM0 から P-DMA0 へのトリガマルチプレクサ)		
0:7	PDMA0_TR_IN[8:15]	P-DMA0[8:15]へのトリガ

表 11　　CYT2B7 シリーズの MUX グループ 3 のトリガ入力

入力 (「j」)	トリガラベル	説明
MUX グループ 3: TCPWM_TO_PDMA0 (TCPWM0 から P-DMA0 へのトリガマルチプレクサ)		
1:4	TCPWM_32_TR_OUT0[0:3]	32 ビット TCPWM0 カウンタ
5:16	TCPWM_16M_TR_OUT0[0:11]	16 ビットモータ拡張 TCPWM0 カウンタ
17:79	**TCPWM_16_TR_OUT0[0:62]**	**16 ビット TCPWM0 カウンタ**
80:82	CAN0_TT_TR_OUT[0:2]	CAN0 TT 同期出力
83:85	CAN1_TT_TR_OUT[0:2]	CAN1 TT 同期出力

図 12 に、入力トリガと出力トリガの構造を示します。

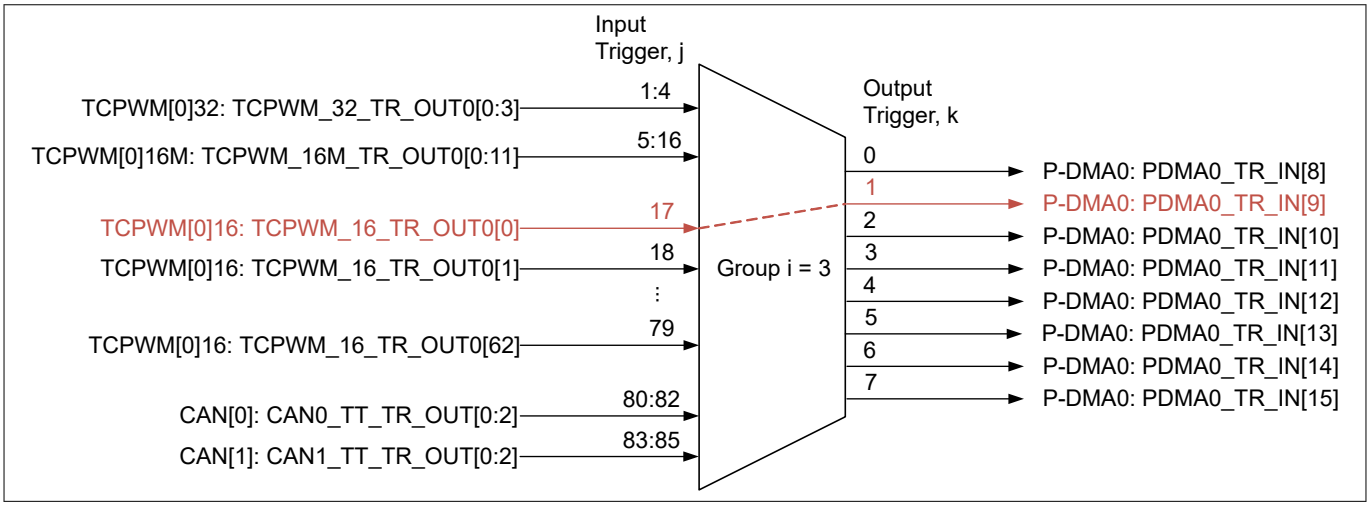

図 12　　CYT2B7 シリーズの 16 ビット TCPWM と P-DMA0 間のトリガ接続

次に、P-DMA0 チャネル 9 と 16 ビット TCPWM チャネル 0 を接続する方法について説明します。

1. 設定レジスタの選択
   MUX グループ 3 (i ＝ 3)
   出力トリガ PDMA0_TR_IN[9] (k ＝ 1)
   したがって、レジスタは PERI_TR_GR**3**_TR_CTL**1** として指定されます。
2. 入力トリガの選択
   入力トリガ TCPWM_16_TR_OUT0[0] (j ＝ 17)

構成は PERI_TR_GR3_TR_CTL1.TR_SEL = **17** になります。各シリーズのトリガマルチプレクサ図を参照する必要があります。これは、正しいマルチプレクサグループと正確な入力および出力トリガが選択されていることを確認するためです。他の MUX グループおよび他のシリーズのグループトリガのトリガの設定については、デバイスのデータシートを参照してください。

3.1.2 TCPWM タイマによる P-DMA 転送のトリガのサンプルプログラム

Code Listing 1 は、図 11 のサンプルプログラムを示します。このアプリケーションノートのプログラムコードは、サンプルドライバライブラリ (SDL) の一部です。SDL については、その他の参考資料を参照してください。

Code Listing 1 TCPWM タイマによる P-DMA 転送のトリガの例

```
void Cy_PDMA_Disable(volatile stc_DW_t *pstcPDMA)
{
  pstcPDMA->unCTL.stcField.u1ENABLED  =       0;    //(1) Disable P-DMA0
}

cy_en_pdma_status_t Cy_PDMA_Descr_Init(cy_stc_pdma_descr_t* descriptor, const
cy_stc_pdma_descr_config_t* config)
{
  cy_en_pdma_status_t retVal = CY_PDMA_ERR_UNC;     //(2) Configure Descriptor

  if ((descriptor != NULL) && (config != NULL))
  {
    descriptor->unPDMA_DESCR_CTL.stcField.u2WAIT_FOR_DEACT          =       config-
>deact;
    descriptor->unPDMA_DESCR_CTL.stcField.u2INTR_TYPE               =       config-
>intrType;
    descriptor->unPDMA_DESCR_CTL.stcField.u2TR_OUT_TYPE             =       config-
>trigoutType;
    descriptor->unPDMA_DESCR_CTL.stcField.u2TR_IN_TYPE              =       config-
>triginType;
    descriptor->unPDMA_DESCR_CTL.stcField.u1SRC_TRANSFER_SIZE       =       config-
>srcTxfrSize;
    descriptor->unPDMA_DESCR_CTL.stcField.u1DST_TRANSFER_SIZE       =       config-
>destTxfrSize;
    descriptor->unPDMA_DESCR_CTL.stcField.u1CH_DISABLE              =       config-
>chStateAtCmplt;
    descriptor->unPDMA_DESCR_CTL.stcField.u2DATA_SIZE               =       config-
>dataSize;
    descriptor->unPDMA_DESCR_CTL.stcField.u2DESCR_TYPE              =       config-
>descrType;

    descriptor->u32PDMA_DESCR_SRC                                   =       (uint32_t)config-
>srcAddr;
    descriptor->u32PDMA_DESCR_DST                                   =       (uint32_t)config-
>destAddr;

    switch(config->descrType)
    {
      case (uint32_t)CY_PDMA_1D_TRANSFER:
      {
        descriptor->unPDMA_DESCR_X_CTL.stcField.u12SRC_X_INCR   =       (uint32_t)config-
>srcXincr;
        descriptor->unPDMA_DESCR_X_CTL.stcField.u12DST_X_INCR   =       (uint32_t)config-
>destXincr;
        descriptor->unPDMA_DESCR_X_CTL.stcField.u8X_COUNT       =       (uint32_t)((config-
>xCount) - 1u);
        descriptor->unPDMA_DESCR_Y_CTL.u32Register              =       (uint32_t)config-
>descrNext;
        break;
      }
    }
```

```
    retVal = CY_PDMA_SUCCESS;
  }
  return retVal;
}

cy_en_pdma_status_t Cy_PDMA_Chnl_Init(volatile stc_DW_t *pstcPDMA, uint32_t chNum, const
cy_stc_pdma_chnl_config_t* chnlConfig)
{
  cy_en_pdma_status_t retVal = CY_PDMA_ERR_UNC;

  if ((pstcPDMA != NULL) && (chnlConfig != NULL))    //(3) Configure P-DMA0 Channel 9
  {
    /* Set current descriptor */
    pstcPDMA->CH_STRUCT[chNum].unCH_CURR_PTR.u32Register           =  (uint32_t)chnlConfig-
>PDMA_Descriptor;
    /* Set if the channel is preemtable */
    pstcPDMA->CH_STRUCT[chNum].unCH_CTL.stcField.u1PREEMPTABLE     =  chnlConfig->preemptable;
    /* Set channel priority */
    pstcPDMA->CH_STRUCT[chNum].unCH_CTL.stcField.u2PRIO            =  chnlConfig->priority;
    /* Set enabled status */
    pstcPDMA->CH_STRUCT[chNum].unCH_CTL.stcField.u1ENABLED         =  chnlConfig->enable;
    retVal = CY_PDMA_SUCCESS;
  }
  return (retVal);
}

void Cy_PDMA_Enable(volatile stc_DW_t *pstcPDMA)
{
  pstcPDMA->unCTL.stcField.u1ENABLED  =       1;    //(4) Enable P-DMA0
}

void Cy_Tcpwm_Counter_Disable(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
  ptscTCPWM->unCTRL.stcField.u1ENABLED = 0x00;    //(5) Disable TCPWM#0 (16-bit)
}

uint32_t Cy_Tcpwm_Counter_Init(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM,
cy_stc_tcpwm_counter_config_t const *config)
{
  uint32_t status = CY_RET_BAD_PARAM;

  if ((NULL != ptscTCPWM) && (NULL != config))
  {
    ptscTCPWM->unCTRL.stcField.u1ONE_SHOT = config->runMode;    //(6) Configure TCPWM#0 (16-
bit) for Timer Mode

    ptscTCPWM->unCTRL.stcField.u2UP_DOWN_MODE = config-> countDirection;
    ptscTCPWM->unCTRL.stcField.u3MODE = config->CompareOrCapture;
    ptscTCPWM->unCTRL.stcField.u1DBG_FREEZE_EN = config->debug_pause;
    ptscTCPWM->unCTRL.stcField.u1AUTO_RELOAD_CC0 = config->enableCompare0Swap;
    ptscTCPWM->unDT.stcField.u8DT_LINE_OUT_L = config->clockPrescaler;

    if (CY_TCPWM_COUNTER_COUNT_UP == config->runMode)
```

```
    {
      ptscTCPWM->unCOUNTER.u32Register = CY_TCPWM_CNT_UP_INIT_VAL;
    }

    ptscTCPWM->unCC0.u32Register = config->compare0;
    ptscTCPWM->unCC0_BUFF.u32Register = config->compare0_buff;
    ptscTCPWM->unPERIOD.u32Register = config->period;
    ptscTCPWM->unTR_IN_SEL0.stcField.u8CAPTURE0_SEL = config->capture0Input;
    ptscTCPWM->unTR_IN_SEL0.stcField.u8RELOAD_SEL = config->reloadInput;
    ptscTCPWM->unTR_IN_SEL0.stcField.u8STOP_SEL = config->stopInput;
    ptscTCPWM->unTR_IN_SEL0.stcField.u8COUNT_SEL = config->countInput;
    ptscTCPWM->unTR_IN_SEL1.stcField.u8START_SEL = config->startInput;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2CAPTURE0_EDGE = config->capture0InputMode;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2RELOAD_EDGE = config->reloadInputMode;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2START_EDGE = config->startInputMode;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2STOP_EDGE = config->stopInputMode;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2COUNT_EDGE = config->countInputMode;
    ptscTCPWM->unTR_OUT_SEL.stcField.u3OUT0 = config->trigger1;
    ptscTCPWM->unTR_OUT_SEL.stcField.u3OUT1 = config->trigger2;
    ptscTCPWM->unINTR_MASK.u32Register = config->interruptSources;
    ptscTCPWM->unCTRL.stcField.u1AUTO_RELOAD_CC1 = config->enableCompare1Swap;
    ptscTCPWM->unCC1.u32Register = config->compare1;
    ptscTCPWM->unCC1_BUFF.u32Register = config->compare1_buff;
    ptscTCPWM->unTR_IN_SEL1.stcField.u8CAPTURE1_SEL = config->capture1Input;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2CAPTURE1_EDGE = config->capture1InputMode;

    status = CY_RET_SUCCESS;
  }
  return(status);
}

cy_en_trigmux_status_t Cy_TrigMux_Connect(uint32_t inTrig, uint32_t outTrig, uint32_t invert,
en_trig_type_t trigType, uint32_t dbg_frz_en)
{
  volatile    stc_PERI_TR_GR_TR_CTL_field_t*  pTR_CTL;    //(7) Configure Group Trigger
              cy_en_trigmux_status_t          retVal = CY_TRIGMUX_BAD_PARAM;
  if ((inTrig & CY_TR_GROUP_MASK) == (outTrig & CY_TR_GROUP_MASK))
  {
    pTR_CTL             =   &(PERI->TR_GR[(outTrig & CY_TR_GROUP_MASK) >>
CY_TR_GROUP_SHIFT].unTR_CTL[outTrig & CY_TR_MASK].stcField); //Select output trigger
TRIG_OUT_MUX_3_PDMA0_TR_IN9
    pTR_CTL->u8TR_SEL   =   inTrig;   //Select input trigger TRIG_IN_MUX_3_TCPWM_16_TR_OUT00
    pTR_CTL->u1TR_INV   =   invert;   //Invert input trigger
    pTR_CTL->u1TR_EDGE  =   trigType; //Select edge sensitive trigger as output trigger type
    retVal              =   CY_TRIGMUX_SUCCESS;
  }
  return retVal;
}

void Cy_Tcpwm_Counter_Enable(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
  ptscTCPWM->unCTRL.stcField.u1ENABLED = 0x1;    //(8) Enable TCPWM#0 (16-bit)
}
```

```
void Cy_Tcpwm_TriggerStart(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
  ptscTCPWM->unTR_CMD.stcField.u1START = 0x01;    //(9) Trigger a SW Start on TCPWM#0 (16-bit)
}
```

注: プログラムコードのグレーアウトされたセクションは、このアプリケーションノートでは説明されません。詳細については、アーキテクチャ TRM を参照してください。

3.2 単一の TCPWM タイマによる 3 つの SAR での同時 ADC 変換

ここでは、グループトリガの入力トリガが 3 つの SAR ユニットで ADC チャネル変換を開始する方法について説明します。

3.2.1 単一の TCPWM タイマによる 3 つの SAR での同時 ADC 変換のユースケースの説明

ADC チャネルでの複数の SAR ADC ユニットのアクティブ化は、単一の TCPWM タイマによって同時にトリガできます。汎用トリガ入力は ADC チャネル間で共有されます。5 つの汎用トリガの中で 1 つを任意の ADC チャネルに接続できます。図 13 に、アプリケーションの例を示します。SAR0, SAR1, および SAR2 の ADC チャネル変換は、グループトリガのマルチプレクサグループ 6 の TCPWM 16 ビットモータ制御カウンタによってトリガされます。TCPWM カウンタのコンペアマッチ 0 は、汎用トリガに接続されている SAR ADC へのトリガを生成します。汎用トリガは、SAR0, SAR1, および SAR2 の ADC チャネル 3 に接続され、同時変換を開始します。

図 14 に動作を示します。TCPWM および SAR ADC の詳細については、アーキテクチャ TRM の「TCPWM」および「SAR ADC」章を参照してください。

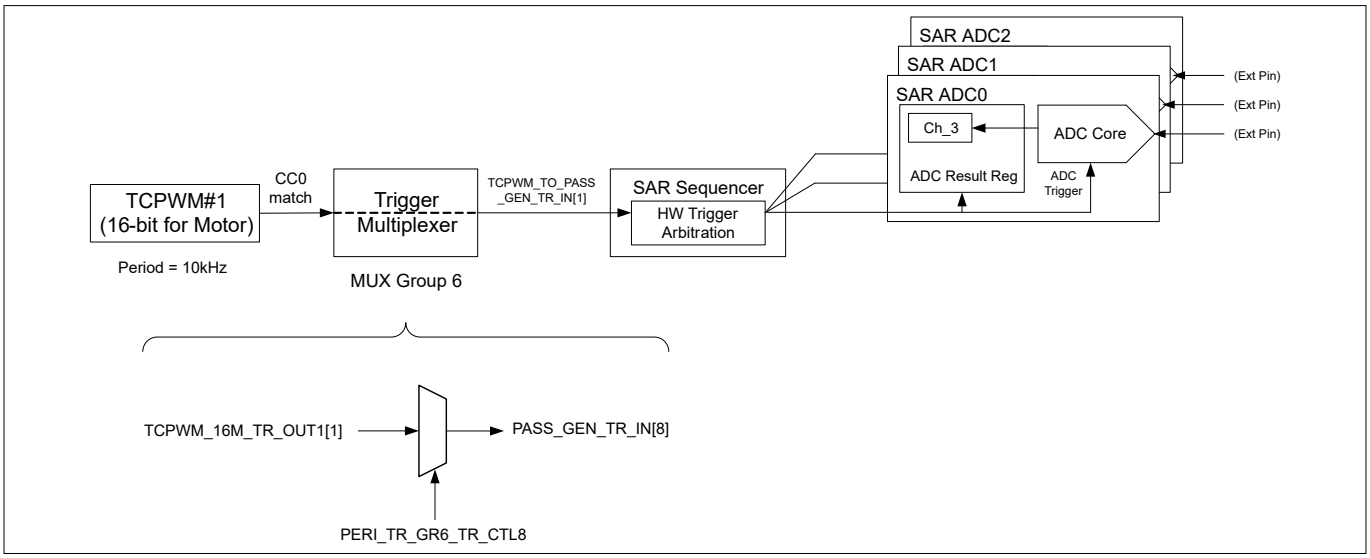

図 13 CYT2B7 シリーズのグループトリガおよび ADC 汎用トリガ入力の例

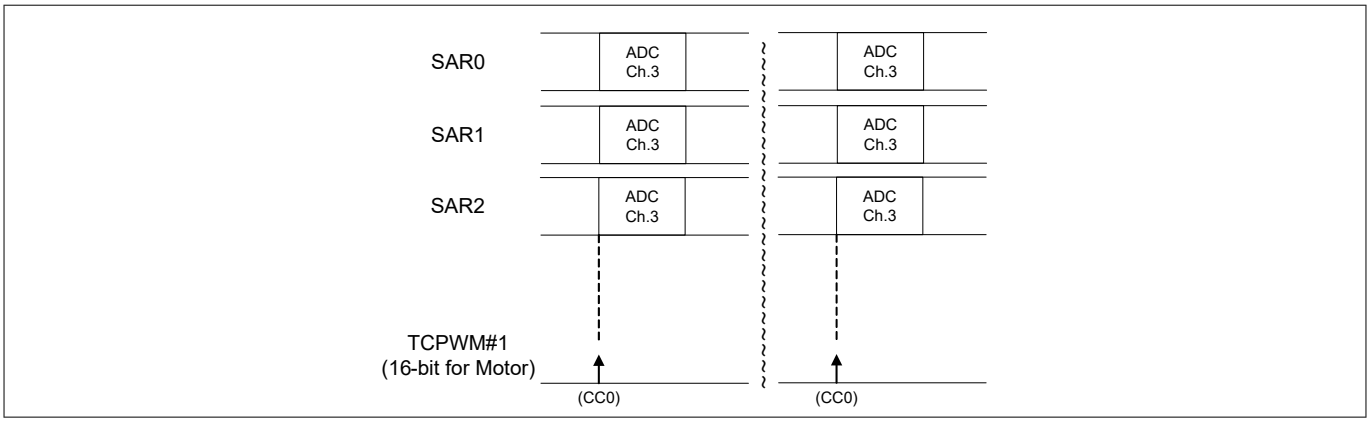

図 14　　動作例: 単一の TCPWM タイマによる 3 つの SAR での同時 ADC 変換

このアプリケーションを実装するには、トリガを設定する必要があります。図 15 に、トリガの設定手順を示します。

3 アプリケーション

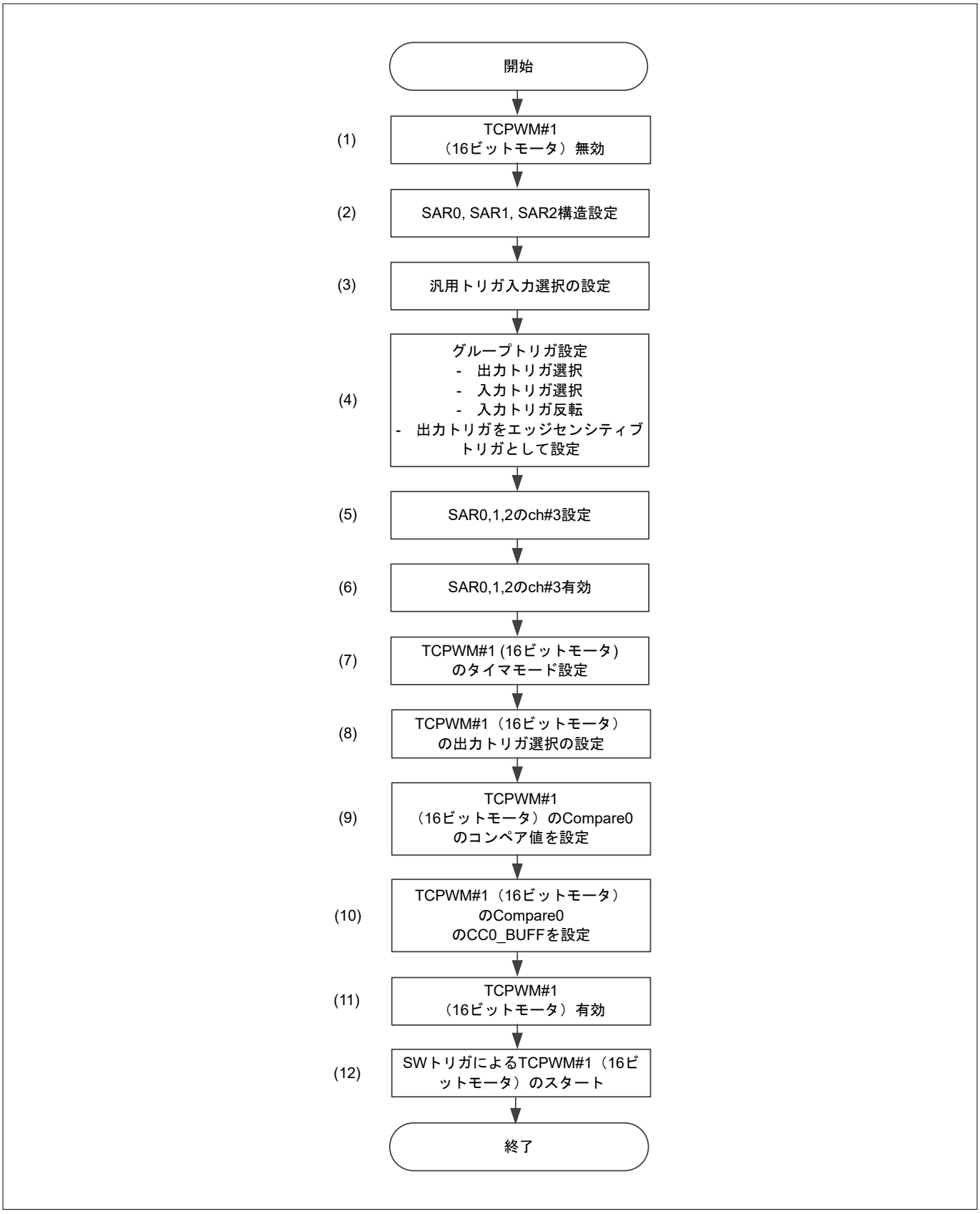

図 15　　CYT2B7 シリーズの ADC 汎用トリガ入力とグループトリガの設定

ステップ (4) のグループトリガの出力トリガと入力トリガの設定は以下のとおりです。

トリガマルチプレクサの設定

3 アプリケーション

表 12 と表 13 に、設定する値を示します。これらの表は、トリガグループ, 入力トリガ, および出力トリガを提供する CYT2B7 シリーズデータシートからの抜粋です。

表 12 CYT2B7 シリーズの MUX グループ 6 のトリガ出力

出力 (「k」)	トリガラベル	説明
MUX グループ 6: PASS (PASS SAR トリガマルチプレクサ)		
0:11	PASS_GEN_TR_IN[0:11]	SAR ADC への汎用トリガ

表 13 CYT2B7 シリーズの MUX グループ 6 のトリガ入力

入力 (「j」)	トリガラベル	説明
MUX グループ 6: PASS (PASS SAR トリガマルチプレクサ)		
1:16	PDMA0_TR_OUT[0:15]	汎用 P-DMA0 トリガ
17:18	CTI_TR_OUT[0:1]	トレースイベント
19:22	FAULT_TR_OUT[0:3]	フォルトイベント
23:25	EVTGEN_TR_OUT[0:2]	EVTGEN トリガ
26:31	PASS_GEN_TR_OUT[0:5]	PASS SAR 完了信号
32:63	HSIOM_IO_INPUT[0:31]	I/O 入力
64:67	TCPWM_32_TR_OUT1[0:3]	32 ビット TCPWM0 カウンタ
68:79	**TCPWM_16M_TR_OUT1[0:11]**	**16 ビットモータ拡張 TCPWM0 カウンタ**

各シリーズのトリガグループ入力およびトリガグループ出力の表については、デバイスのデータシートを参照してください。

図 16 に、入力トリガと出力トリガの構造を示します。

3 アプリケーション

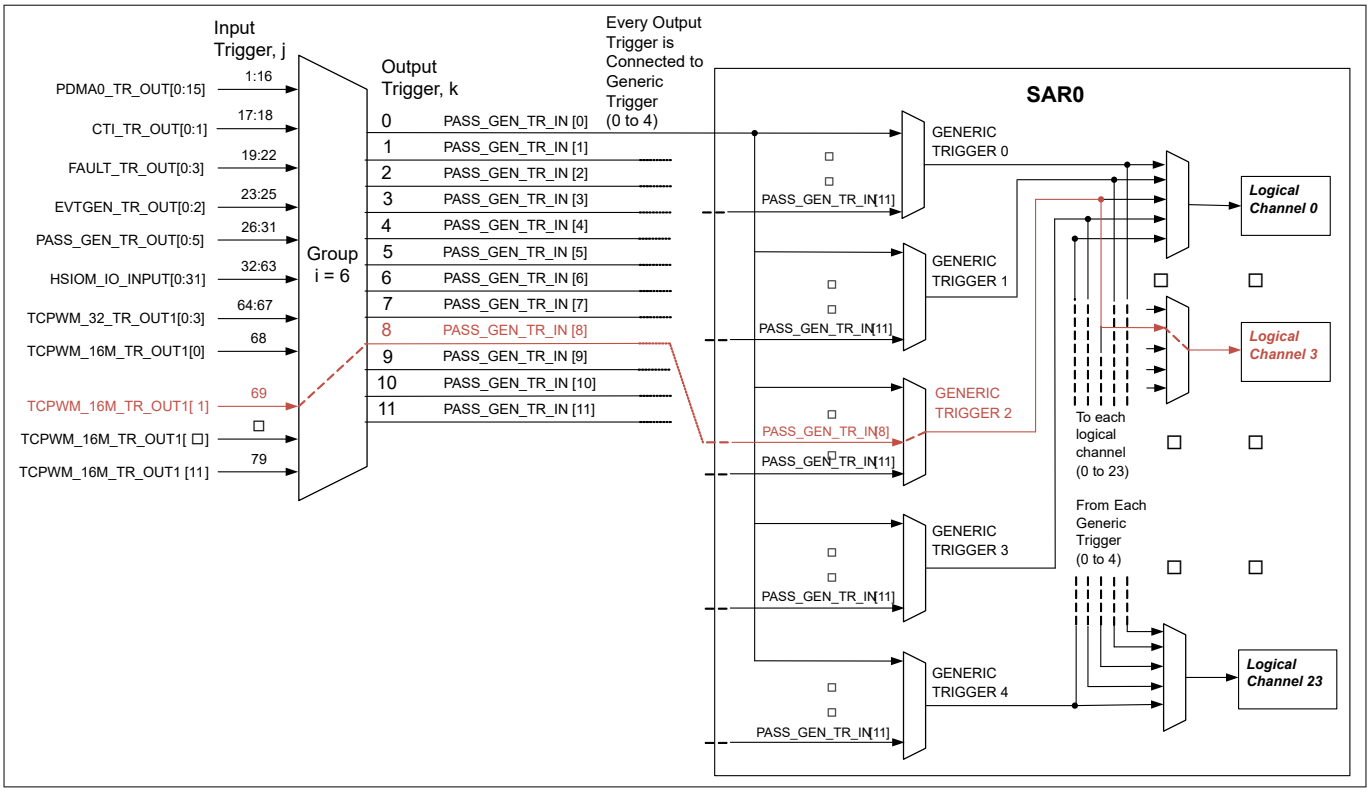

図 16　　TCPWM 16 ビットモータと CYT2B7 シリーズ SAR0 ユニットの汎用トリガ 2 間のトリガ接続

次に、トリガソース TCPWM 16 ビットモータ制御カウンタが各 SAR の汎用トリガ 2 に接続する方法を説明します。

1. 設定レジスタの選択
   MUX グループ 6 (i ＝ 6)
   出力トリガ、汎用トリガ 8 または PASS_GEN_TR_IN[8] (k ＝ 8)
   したがって、レジスタは PERI_TR_GR**6**_TR_CTL**8** として指定されます。
2. 入力トリガの選択
   入力トリガ TCPWM_16M_TR_OUT1[1] (j ＝ 69)
   構成は PERI_TR_GR6_TR_CTL8.TR_SEL = **69** になります。
3. 汎用トリガ 8 は、対応する SAR ユニットの SAR 汎用トリガ入力 2、IN2 に転送されます。
   - SAR0 の場合、構成は
     - SAR_TR_IN_SEL**0.**IN**2**_SEL = 8
   - SAR1 の場合、構成は
     - SAR_TR_IN_SEL**1.**IN**2**_SEL = 8
   - SAR2 の場合、構成は
     - SAR_TR_IN_SEL**2.**IN**2**_SEL = 8
4. IN2 は、対応する SAR ユニットのチャネル 3 をトリガします
   - SAR0 の場合、SAR トリガ選択の構成は
     - PASS0_SAR0_CH**3**_TR_CTL.SEL = 4 (この値は SAR 汎用トリガ入力 2 を表します)
   - SAR1 の場合、SAR トリガ選択の構成は
     - PASS0_SAR1_CH**3**_TR_CTL.SEL = 4
   - SAR2 の場合、SAR トリガ選択の構成は
     - PASS0_SAR2_CH**3**_TR_CTL.SEL = 4

グループトリガのトリガ設定については、デバイスのデータシートを参照してください。

3.2.2 単一の TCPWM タイマによる 3 つの SAR での同時 ADC 変換のサンプルプログラム

Code Listing 2 は、図 15 のサンプルプログラムを示します。

Code Listing 2 単一の TCPWM タイマによる 3 つの SAR での同時 ADC 変換の例

```
void Cy_Tcpwm_Counter_Disable(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
  ptscTCPWM->unCTRL.stcField.u1ENABLED = 0x00;     //(1) Disable TCPWM#1 (16-bit for Motor)
}

cy_en_adc_status_t Cy_Adc_Init(volatile stc_PASS_SAR_t * base, const cy_stc_adc_config_t *
config)
{
  cy_en_adc_status_t ret = CY_ADC_SUCCESS;
  un_PASS_SAR_CTL_t unSarCtl = { 0 };
  if (NULL != config)
  {
    /* CTL register setting */
    //(2) Configure SAR0, SAR1, SAR2 Structure
    base->unPRECOND_CTL.stcField.u4PRECOND_TIME = config->preconditionTime;

    /* CTL register setting */
    unSarCtl.stcField.u8PWRUP_TIME = config->powerupTime;
    unSarCtl.stcField.u1IDLE_PWRDWN = config->enableIdlePowerDown ? 1u : 0;
    unSarCtl.stcField.u1MSB_STRETCH = config->msbStretchMode;
    unSarCtl.stcField.u1HALF_LSB = config->enableHalfLsbConv ? 1u : 0;
    unSarCtl.stcField.u1SARMUX_EN = config->sarMuxEnable ? 1u : 0;
    unSarCtl.stcField.u1ADC_EN = config->adcEnable ? 1u : 0;
    unSarCtl.stcField.u1ENABLED = config->sarIpEnable ? 1u : 0;
    base->unCTL.u32Register = unSarCtl.u32Register;
  }
  else
  {
    ret = CY_ADC_BAD_PARAM;
  }
  return ret;
}

cy_en_adc_status_t Cy_Adc_SetGenericTriggerInput(volatile stc_PASS_EPASS_MMIO_t * base, uint8_t
numOfAdc, uint8_t triggerInputNumber, uint8_t genericTriggerValue)
{
  cy_en_adc_status_t ret = CY_ADC_SUCCESS;
  {
    switch(triggerInputNumber)     //(3) Configure Generic Trigger Input Selection
    {
      case 2:
      base->unSAR_TR_IN_SEL[numOfAdc].stcField.u4IN2_SEL = genericTriggerValue; //Select
GENERIC TRIGGER 2
      break;
    }
  }
  return ret;
}

cy_en_trigmux_status_t Cy_TrigMux_Connect(uint32_t inTrig, uint32_t outTrig, uint32_t invert,
en_trig_type_t trigType, uint32_t dbg_frz_en)
```

```
{
  volatile    stc_PERI_TR_GR_TR_CTL_field_t*  pTR_CTL;      //(4) Configure Group Trigger
              cy_en_trigmux_status_t          retVal = CY_TRIGMUX_BAD_PARAM;
  if ((inTrig & CY_TR_GROUP_MASK) == (outTrig & CY_TR_GROUP_MASK))
  {
    pTR_CTL             =   &(PERI->TR_GR[(outTrig & CY_TR_GROUP_MASK) >>
CY_TR_GROUP_SHIFT].unTR_CTL[outTrig & CY_TR_MASK].stcField); //Select output trigger
TRIG_OUT_MUX_6_PASS_GEN_TR_IN8
    pTR_CTL->u8TR_SEL   =   inTrig;   //Select input trigger TRIG_IN_MUX_6_TCPWM_16M_TR_OUT11
    pTR_CTL->u1TR_INV   =   invert;   //Invert input trigger
    pTR_CTL->u1TR_EDGE  =   trigType; // Select edge sensitive trigger as output trigger type
    retVal              =   CY_TRIGMUX_SUCCESS;
  }
  return retVal;
}

cy_en_adc_status_t Cy_Adc_Channel_Init(volatile stc_PASS_SAR_CH_t * base, const
cy_stc_adc_channel_config_t * config)
{
  cy_en_adc_status_t ret = CY_ADC_SUCCESS;
  un_PASS_SAR_CH_TR_CTL_t unTrCtl = { 0 };     //(5) Configure ch#3 of SAR0,1,2
  un_PASS_SAR_CH_SAMPLE_CTL_t unSampleCtl = { 0 };
  un_PASS_SAR_CH_POST_CTL_t unPostCtl = { 0 };
  un_PASS_SAR_CH_RANGE_CTL_t unRangeCtl = { 0 };
  un_PASS_SAR_CH_INTR_t unIntr = { 0 };

  if (NULL != config)
  {
    /* At first disable the channel */
    {
      base->unENABLE.stcField.u1CHAN_EN = 0u;
    }
    /* Clear whole interrupt flags */
    unIntr.stcField.u1CH_OVERFLOW = 1u;
    unIntr.stcField.u1CH_PULSE = 1u;
    unIntr.stcField.u1CH_RANGE = 1u;
    unIntr.stcField.u1GRP_CANCELLED = 1u;
    unIntr.stcField.u1GRP_DONE = 1u;
    unIntr.stcField.u1GRP_OVERFLOW = 1u;
    base->unINTR.u32Register = unIntr.u32Register;

    unTrCtl.stcField.u3SEL = config->triggerSelection;
    unTrCtl.stcField.u3PRIO = config->channelPriority;
    unTrCtl.stcField.u2PREEMPT_TYPE = config->preenptionType;
    unTrCtl.stcField.u1GROUP_END = config->isGroupEnd ? 1u : 0u;
    unTrCtl.stcField.u1DONE_LEVEL = config->doneLevel ? 1u : 0u;
    base->unTR_CTL.u32Register = unTrCtl.u32Register;

    unSampleCtl.stcField.u6PIN_ADDR = config->pinAddress;
    unSampleCtl.stcField.u2PORT_ADDR = config->portAddress;
    unSampleCtl.stcField.u3EXT_MUX_SEL = config->extMuxSelect;
    unSampleCtl.stcField.u1EXT_MUX_EN = config->extMuxEnable ? 1u : 0u;
    unSampleCtl.stcField.u2PRECOND_MODE = config->preconditionMode;
```

```
    unSampleCtl.stcField.u2OVERLAP_DIAG = config->overlapDiagMode;
    unSampleCtl.stcField.u12SAMPLE_TIME = config->sampleTime;
    unSampleCtl.stcField.u1ALT_CAL = config->calibrationValueSelect;
    base->unSAMPLE_CTL.u32Register = unSampleCtl.u32Register;

    unPostCtl.stcField.u3POST_PROC = config->postProcessingMode;
    unPostCtl.stcField.u1LEFT_ALIGN = config->resultAlignment;
    unPostCtl.stcField.u1SIGN_EXT = config->signExtention;
    unPostCtl.stcField.u8AVG_CNT = config->averageCount;
    unPostCtl.stcField.u5SHIFT_R = config->rightShift;
    unPostCtl.stcField.u2RANGE_MODE = config->rangeDetectionMode;
    base->unPOST_CTL.u32Register = unPostCtl.u32Register;

    unRangeCtl.stcField.u16RANGE_LO = config->rangeDetectionLoThreshold;
    unRangeCtl.stcField.u16RANGE_HI = config->rangeDetectionHiThreshold;
    base->unRANGE_CTL.u32Register = unRangeCtl.u32Register;

    {
    cy_en_adc_status_t ret = CY_ADC_SUCCESS;
    un_PASS_SAR_CH_INTR_MASK_t unMask = { 0 };
    if (NULL != mask)
    {
      unMask.stcField.u1CH_OVERFLOW_MASK = mask->chOverflow ? 1u : 0u;
      unMask.stcField.u1CH_PULSE_MASK = mask->chPulse ? 1u : 0u;
      unMask.stcField.u1CH_RANGE_MASK = mask->chRange ? 1u : 0u;
      unMask.stcField.u1GRP_CANCELLED_MASK = mask->grpCancelled ? 1u : 0u;
      unMask.stcField.u1GRP_DONE_MASK = mask->grpDone ? 1u : 0u;
      unMask.stcField.u1GRP_OVERFLOW_MASK = mask->grpOverflow ? 1u : 0u;
      base->unINTR_MASK.u32Register = unMask.u32Register;
    }
    return ret;
  }
  return ret;
}

void Cy_Adc_Channel_Enable(volatile stc_PASS_SAR_CH_t * base)
{
  base->unENABLE.stcField.u1CHAN_EN = 1u;     //(6) Enable ch#3 of SAR0,1,2
}

uint32_t Cy_Tcpwm_Counter_Init(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM,
cy_stc_tcpwm_counter_config_t const *config)
{
  uint32_t status = CY_RET_BAD_PARAM;     //(7) Configure TCPWM#1 (16-bit for Motor) for Timer
Mode

  if (config->trigger1 > 0x04 || config->trigger2 > 0x04)
  {
    return status;
  }


  if ((NULL != ptscTCPWM) && (NULL != config))
```

```
  {
    ptscTCPWM->unCTRL.stcField.u1ONE_SHOT = config->runMode;
    ptscTCPWM->unCTRL.stcField.u2UP_DOWN_MODE = config-> countDirection;
    ptscTCPWM->unCTRL.stcField.u3MODE = config->CompareOrCapture;
    ptscTCPWM->unCTRL.stcField.u1DBG_FREEZE_EN = config->debug_pause;
    ptscTCPWM->unCTRL.stcField.u1AUTO_RELOAD_CC0 = config->enableCompare0Swap;
    ptscTCPWM->unDT.stcField.u8DT_LINE_OUT_L = config->clockPrescaler;

    if (CY_TCPWM_COUNTER_COUNT_UP == config->runMode)
    {
      ptscTCPWM->unCOUNTER.u32Register = CY_TCPWM_CNT_UP_INIT_VAL;
    }
    else if (CY_TCPWM_COUNTER_COUNT_DOWN == config->runMode)
    {
      ptscTCPWM->unCOUNTER.u32Register = config->period;
    }
    else
    {
      ptscTCPWM->unCOUNTER.u32Register = CY_TCPWM_CNT_UP_DOWN_INIT_VAL;
    }

    ptscTCPWM->unCC0.u32Register = config->compare0;
    ptscTCPWM->unCC0_BUFF.u32Register = config->compare0_buff;
    ptscTCPWM->unPERIOD.u32Register = config->period;
    ptscTCPWM->unTR_IN_SEL0.stcField.u8CAPTURE0_SEL = config->capture0Input;
    ptscTCPWM->unTR_IN_SEL0.stcField.u8RELOAD_SEL = config->reloadInput;
    ptscTCPWM->unTR_IN_SEL0.stcField.u8STOP_SEL = config->stopInput;
    ptscTCPWM->unTR_IN_SEL0.stcField.u8COUNT_SEL = config->countInput;
    ptscTCPWM->unTR_IN_SEL1.stcField.u8START_SEL = config->startInput;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2CAPTURE0_EDGE = config->capture0InputMode;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2RELOAD_EDGE = config->reloadInputMode;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2START_EDGE = config->startInputMode;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2STOP_EDGE = config->stopInputMode;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2COUNT_EDGE = config->countInputMode;
    ptscTCPWM->unTR_OUT_SEL.stcField.u3OUT0 = config->trigger1;
    ptscTCPWM->unTR_OUT_SEL.stcField.u3OUT1 = config->trigger2;
    ptscTCPWM->unINTR_MASK.u32Register = config->interruptSources;
    ptscTCPWM->unCTRL.stcField.u1AUTO_RELOAD_CC1 = config->enableCompare1Swap;
    ptscTCPWM->unCC1.u32Register = config->compare1;
    ptscTCPWM->unCC1_BUFF.u32Register = config->compare1_buff;
    ptscTCPWM->unTR_IN_SEL1.stcField.u8CAPTURE1_SEL = config->capture1Input;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2CAPTURE1_EDGE = config->capture1InputMode;

    status = CY_RET_SUCCESS;
  }

  return(status);
}

  TCPWM0_GRP1_CNT1->unTR_OUT_SEL.stcField.u3OUT1 = CY_TCPWM_COUNTER_CC0_MATCH;     //(8)
Configure Output Trigger Selection for TCPWM#1 (16-bit for Motor)

void Cy_Tcpwm_Counter_SetCompare0(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM,  uint32_t compare0)
```

```
{
  ptscTCPWM->unCC0.u32Register = compare0;     //(9) Set Compare Value of Compare0 for TCPWM#1
(16-bit for Motor)
}

void Cy_Tcpwm_Counter_SetCompare0_Buff(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM,  uint32_t
compare1)
{
  ptscTCPWM->unCC0_BUFF.u32Register = compare1;     //(10) Set Compare Value of CC0_BUFF for
TCPWM#1 (16-bit for Motor)
}

void Cy_Tcpwm_Counter_Enable(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
  ptscTCPWM->unCTRL.stcField.u1ENABLED = 0x1;     //(11) Enable TCPWM#1 (16-bit for Motor)
}

void Cy_Tcpwm_TriggerStart(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
  ptscTCPWM->unTR_CMD.stcField.u1START = 0x01;     //(12) Trigger a SW Start on TCPWM#1 (16-bit
for Motor)
}
```

注: プログラムコードのグレーアウトされたセクションは、このアプリケーションノートでは説明されません。詳細については、アーキテクチャ TRM を参照してください。

3.3 TCPWM タイマによる ADC 変換のトリガ

ここでは、1-to-1 のグループトリガの入力トリガが ADC 変換を開始する方法について説明します。

3.3.1 TCPWM タイマによる ADC 変換のトリガのユースケースの説明

図 17 に、CYT2B7 シリーズでの 1-to-1 トリガのアプリケーションの例を示します。SAR0 の ADC 変換は、1-to-1 トリガのマルチプレクサグループ 1 の TCPWM 16 ビットカウンタによってトリガされます。TCPWM カウンタのチャネル 0 とチャネル 1 のコンペアマッチは、それぞれ SAR0 のチャネル 4 と 5 で変換をトリガします。図 18 に動作を示します。TCPWM および SAR ADC の詳細については、アーキテクチャ TRM の「TCPWM」および「SAR ADC」章を参照してください。

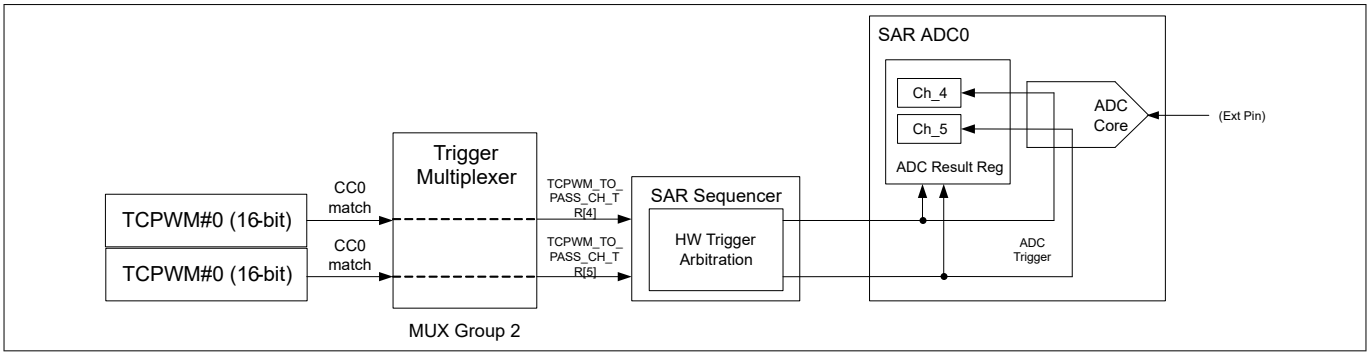

図 17 CYT2B7 シリーズの 1-to-1 トリガの例

3 アプリケーション

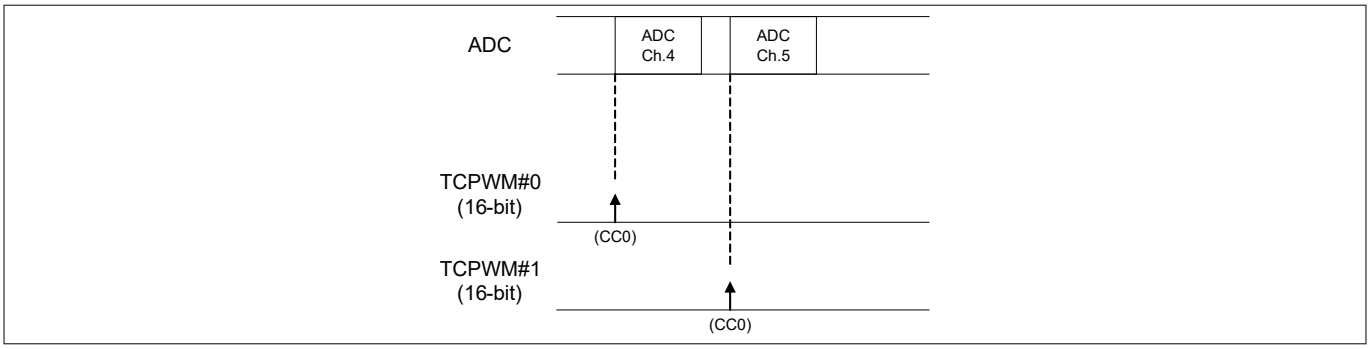

図 18　　動作例: TCPWM タイマによる ADC 変換のトリガ

このアプリケーションを実装するには、トリガを設定する必要があります。図 19 に、トリガの設定手順を示します。図 19 に、TCPWM カウンタと SAR0 構造の設定も示します。

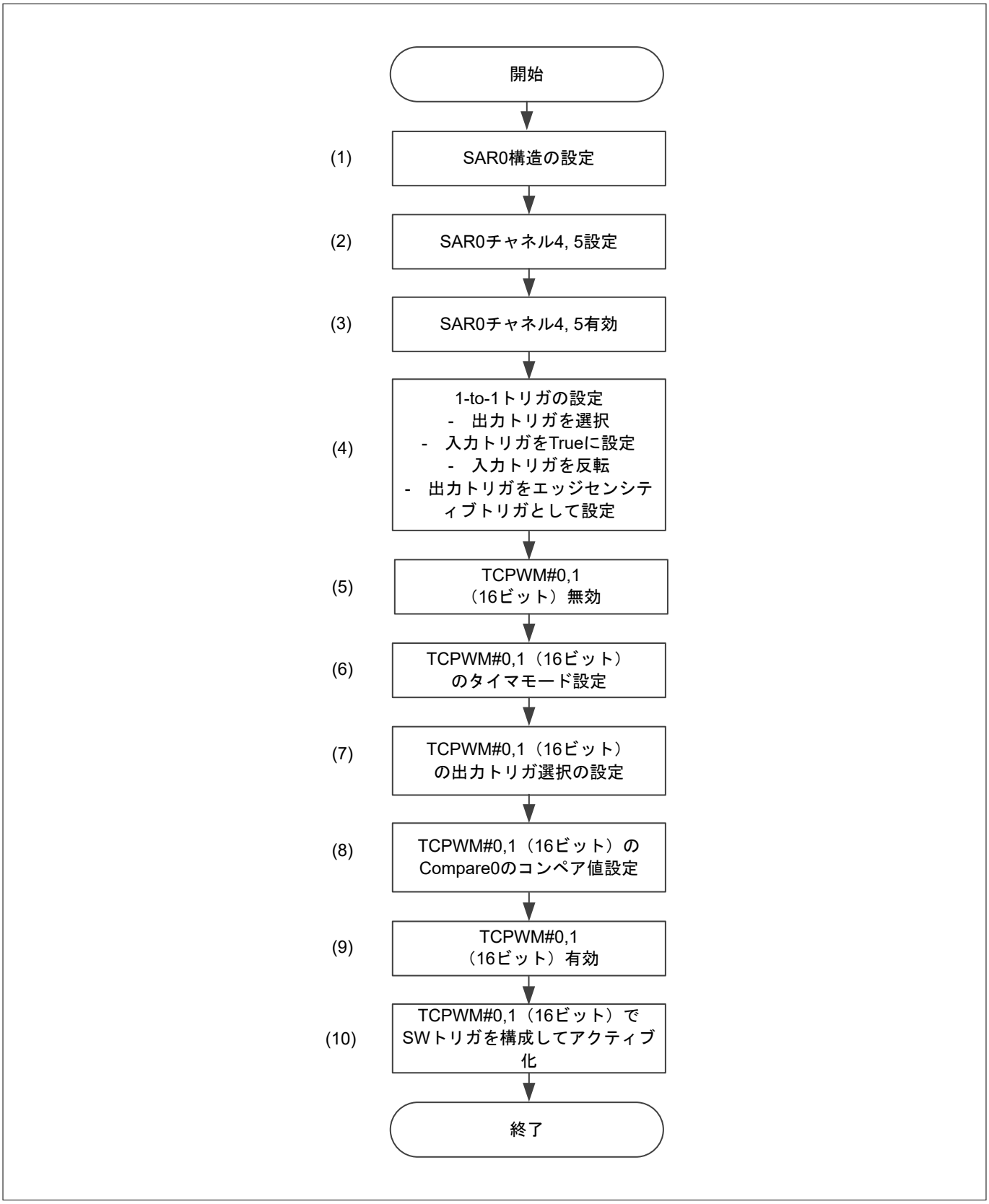

図 19　　CYT2B7 シリーズの 1-to-1 トリガの設定

ステップ (4) の 1-to-1 トリガの出力トリガと入力トリガの構成は次のとおりです。

トリガマルチプレクサの設定

表 14 に設定する値を示します。この表は、トリガグループと入力トリガを提供する CYT2B7 シリーズデータシートからの抜粋です。

表 14　　CYT2B7 シリーズの 1-to-1 トリガの MUX グループ 1

入力トリガ k	トリガ入力	トリガ出力
MUX グループ 1: TCPWM0 から PASS SARx への直接接続		
0	TCPWM0_16M_TR_OUT1[0]	PASS0_CH_TR_IN[0]
1	TCPWM0_16M_TR_OUT1[1]	PASS0_CH_TR_IN[1]
2	TCPWM0_16M_TR_OUT1[2]	PASS0_CH_TR_IN[2]
3	TCPWM0_16M_TR_OUT1[3]	PASS0_CH_TR_IN[3]
4:23	**TCPWM0_16_TR_OUT1[0:19]**	**PASS0_CH_TR_IN[4:23]**
24	TCPWM0_16M_TR_OUT1[4]	PASS0_CH_TR_IN[32]
25	TCPWM0_16M_TR_OUT1[5]	PASS0_CH_TR_IN[33]
26	TCPWM0_16M_TR_OUT1[6]	PASS0_CH_TR_IN[34]
27	TCPWM0_16M_TR_OUT1[7]	PASS0_CH_TR_IN[35]
28:55	TCPWM0_16_TR_OUT1[20:47]	PASS0_CH_TR_IN[36:63]
56	TCPWM0_16M_TR_OUT1[8]	PASS0_CH_TR_IN[64]
57	TCPWM0_16M_TR_OUT1[9]	PASS0_CH_TR_IN[65]
58	TCPWM0_16M_TR_OUT1[10]	PASS0_CH_TR_IN[66]
59	TCPWM0_16M_TR_OUT1[11]	PASS0_CH_TR_IN[67]
60:63	TCPWM0_16_TR_OUT1[48:51]	PASS0_CH_TR_IN[68:71]

次に、SAR0 チャネル 4 およびチャネル 5 を 16 ビット TCPWM チャネル 0 に接続する方法について説明します。1-to-1 のグループトリガでは、1 つの入力トリガが特定の出力トリガに直接接続されます。したがって、出力トリガを示すには、入力トリガ番号のみを指定するだけで十分です。

1. 設定レジスタの選択
   MUX グループ 1 (i ＝ 1)
   入力トリガ TCPWM0_16_TR_OUT1[0]は、直接出力トリガ PASS0_CH_TR_IN[4] (k ＝ 4) に接続します。
   入力トリガ TCPWM0_16_TR_OUT1[1] は、直接出力トリガ PASS0_CH_TR_IN[5] (k ＝ 5) に接続します。
   したがって、レジスタはそれぞれ PERI_TR_1TO1_GR**1**_TR_CTL**4** および PERI_TR_1TO1_GR**1**_TR_CTL**5** として指定されます。
2. 入力トリガの設定
   基本的に、1-to-1 トリガの入力トリガが有効になります。
   構成は PERI_TR_1TO1_GR1_TR_CTL4.TR_SEL = **1** および
   PERI_TR_1TO1_GR1_TR_CTL5.TR_SEL = **1** になります。

トリガ入力とトリガ出力の詳細については、デバイスのデータシートを参照してください。

3.3.2　TCPWM タイマによる ADC 変換をトリガするサンプルプログラム

Code Listing 3 は、図 19 のサンプルプログラムを示します。

Code Listing 3 TCPWM タイマによる ADC 変換のトリガの例

```
cy_en_adc_status_t Cy_Adc_Init(volatile stc_PASS_SAR_t * base, const cy_stc_adc_config_t *
config)
{
  cy_en_adc_status_t ret = CY_ADC_SUCCESS;    //(1) Configure SAR0 Structure
  un_PASS_SAR_CTL_t unSarCtl = { 0 };
  if (NULL != config)
  {
    /* CTL register setting */
    base->unPRECOND_CTL.stcField.u4PRECOND_TIME = config->preconditionTime;

    /* CTL register setting */
    unSarCtl.stcField.u8PWRUP_TIME = config->powerupTime;
    unSarCtl.stcField.u1IDLE_PWRDWN = config->enableIdlePowerDown ? 1u : 0;
    unSarCtl.stcField.u1MSB_STRETCH = config->msbStretchMode;
    unSarCtl.stcField.u1HALF_LSB = config->enableHalfLsbConv ? 1u : 0;
    unSarCtl.stcField.u1SARMUX_EN = config->sarMuxEnable ? 1u : 0;
    unSarCtl.stcField.u1ADC_EN = config->adcEnable ? 1u : 0;
    unSarCtl.stcField.u1ENABLED = config->sarIpEnable ? 1u : 0;
    base->unCTL.u32Register = unSarCtl.u32Register;
  }
  return ret;
}

cy_en_adc_status_t Cy_Adc_Channel_Init(volatile stc_PASS_SAR_CH_t * base, const
cy_stc_adc_channel_config_t * config)

{
  cy_en_adc_status_t ret = CY_ADC_SUCCESS;    //(2) Configure SAR0 Channel 4, 5
  un_PASS_SAR_CH_TR_CTL_t unTrCtl = { 0 };
  un_PASS_SAR_CH_SAMPLE_CTL_t unSampleCtl = { 0 };
  un_PASS_SAR_CH_POST_CTL_t unPostCtl = { 0 };
  un_PASS_SAR_CH_RANGE_CTL_t unRangeCtl = { 0 };
  un_PASS_SAR_CH_INTR_t unIntr = { 0 };

  if (NULL != config)
  {
    /* At first disable the channel */
    base->unENABLE.stcField.u1CHAN_EN = 0u;

    /* Clear whole interrupt flags */
    unIntr.stcField.u1CH_OVERFLOW = 1u;
    unIntr.stcField.u1CH_PULSE = 1u;
    unIntr.stcField.u1CH_RANGE = 1u;
    unIntr.stcField.u1GRP_CANCELLED = 1u;
    unIntr.stcField.u1GRP_DONE = 1u;
    unIntr.stcField.u1GRP_OVERFLOW = 1u;
    base->unINTR.u32Register = unIntr.u32Register;
    unTrCtl.stcField.u3SEL = config->triggerSelection;
    unTrCtl.stcField.u3PRIO = config->channelPriority;
    unTrCtl.stcField.u2PREEMPT_TYPE = config->preenptionType;
    unTrCtl.stcField.u1GROUP_END = config->isGroupEnd ? 1u : 0u;
```

```
    unTrCtl.stcField.u1DONE_LEVEL = config->doneLevel ? 1u : 0u;
    base->unTR_CTL.u32Register = unTrCtl.u32Register;

    unSampleCtl.stcField.u6PIN_ADDR = config->pinAddress;
    unSampleCtl.stcField.u2PORT_ADDR = config->portAddress;
    unSampleCtl.stcField.u3EXT_MUX_SEL = config->extMuxSelect;
    unSampleCtl.stcField.u1EXT_MUX_EN = config->extMuxEnable ? 1u : 0u;
    unSampleCtl.stcField.u2PRECOND_MODE = config->preconditionMode;
    unSampleCtl.stcField.u2OVERLAP_DIAG = config->overlapDiagMode;
    unSampleCtl.stcField.u12SAMPLE_TIME = config->sampleTime;
    unSampleCtl.stcField.u1ALT_CAL = config->calibrationValueSelect;
    base->unSAMPLE_CTL.u32Register = unSampleCtl.u32Register;

    unPostCtl.stcField.u3POST_PROC = config->postProcessingMode;
    unPostCtl.stcField.u1LEFT_ALIGN = config->resultAlignment;
    unPostCtl.stcField.u1SIGN_EXT = config->signExtention;
    unPostCtl.stcField.u8AVG_CNT = config->averageCount;
    unPostCtl.stcField.u5SHIFT_R = config->rightShift;
    unPostCtl.stcField.u2RANGE_MODE = config->rangeDetectionMode;
    base->unPOST_CTL.u32Register = unPostCtl.u32Register;

    unRangeCtl.stcField.u16RANGE_LO = config->rangeDetectionLoThreshold;
    unRangeCtl.stcField.u16RANGE_HI = config->rangeDetectionHiThreshold;
    base->unRANGE_CTL.u32Register = unRangeCtl.u32Register;

    {
    cy_en_adc_status_t ret = CY_ADC_SUCCESS;
    un_PASS_SAR_CH_INTR_MASK_t unMask = { 0 };
    if (NULL != mask)
    {
      unMask.stcField.u1CH_OVERFLOW_MASK = mask->chOverflow ? 1u : 0u;
      unMask.stcField.u1CH_PULSE_MASK = mask->chPulse ? 1u : 0u;
      unMask.stcField.u1CH_RANGE_MASK = mask->chRange ? 1u : 0u;
      unMask.stcField.u1GRP_CANCELLED_MASK = mask->grpCancelled ? 1u : 0u;
      unMask.stcField.u1GRP_DONE_MASK = mask->grpDone ? 1u : 0u;
      unMask.stcField.u1GRP_OVERFLOW_MASK = mask->grpOverflow ? 1u : 0u;
      base->unINTR_MASK.u32Register = unMask.u32Register;
    }
    return ret;
  }
  return ret;
}

void Cy_Adc_Channel_Enable(volatile stc_PASS_SAR_CH_t * base)
{
  base->unENABLE.stcField.u1CHAN_EN = 1u;     //(3) Enable SAR0 Channel 4, 5
}

void Cy_TrigMux_Connect1To1T(uint32_t outTrig, uint32_t invert, en_trig_type_t trigType,
uint32_t dbg_frz_en)
{
  volatile    stc_PERI_TR_1TO1_GR_TR_CTL_field_t*  pTR_CTL;
  pTR_CTL             =   &(PERI->TR_1TO1_GR[(outTrig & CY_TR_GROUP_MASK) >>
```

```
  CY_TR_GROUP_SHIFT].unTR_CTL[outTrig & CY_TR_MASK].stcField); //Select output trigger
TRIG_OUT_1TO1_1_TCPWM_TO_PASS_CH_TR4    //(4) Configure One-to-One Trigger
  pTR_CTL->u1TR_SEL    =    1; //Set input trigger as true
  pTR_CTL->u1TR_INV    =    invert; //Invert input trigger
  pTR_CTL->u1TR_EDGE   =    trigType; //Select edge sensitive as trigger type
}

void Cy_Tcpwm_Counter_Disable(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
  ptscTCPWM->unCTRL.stcField.u1ENABLED = 0x00;      //(5) Disable TCPWM#0,1 (16-bit)
}

uint32_t Cy_Tcpwm_Counter_Init(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM,
cy_stc_tcpwm_counter_config_t const *config)
{
  uint32_t status = CY_RET_BAD_PARAM;

  if ((NULL != ptscTCPWM) && (NULL != config))     //(6) Configure TCPWM#0,1 (16-bit) for Timer
Mode
  {
    ptscTCPWM->unCTRL.stcField.u1ONE_SHOT = config->runMode;
    ptscTCPWM->unCTRL.stcField.u2UP_DOWN_MODE = config-> countDirection;
    ptscTCPWM->unCTRL.stcField.u3MODE = config->CompareOrCapture;
    ptscTCPWM->unCTRL.stcField.u1DBG_FREEZE_EN = config->debug_pause;
    ptscTCPWM->unCTRL.stcField.u1AUTO_RELOAD_CC0 = config->enableCompare0Swap;
    ptscTCPWM->unDT.stcField.u8DT_LINE_OUT_L = config->clockPrescaler;

    if (CY_TCPWM_COUNTER_COUNT_UP == config->runMode)
    {
      ptscTCPWM->unCOUNTER.u32Register = CY_TCPWM_CNT_UP_INIT_VAL;
    }

    ptscTCPWM->unCC0.u32Register = config->compare0;
    ptscTCPWM->unCC0_BUFF.u32Register = config->compare0_buff;
    ptscTCPWM->unPERIOD.u32Register = config->period;
    ptscTCPWM->unTR_IN_SEL0.stcField.u8CAPTURE0_SEL = config->capture0Input;
    ptscTCPWM->unTR_IN_SEL0.stcField.u8RELOAD_SEL = config->reloadInput;
    ptscTCPWM->unTR_IN_SEL0.stcField.u8STOP_SEL = config->stopInput;
    ptscTCPWM->unTR_IN_SEL0.stcField.u8COUNT_SEL = config->countInput;
    ptscTCPWM->unTR_IN_SEL1.stcField.u8START_SEL = config->startInput;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2CAPTURE0_EDGE = config->capture0InputMode;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2RELOAD_EDGE = config->reloadInputMode;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2START_EDGE = config->startInputMode;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2STOP_EDGE = config->stopInputMode;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2COUNT_EDGE = config->countInputMode;
    ptscTCPWM->unTR_OUT_SEL.stcField.u3OUT0 = config->trigger1;
    ptscTCPWM->unTR_OUT_SEL.stcField.u3OUT1 = config->trigger2;
    ptscTCPWM->unINTR_MASK.u32Register = config->interruptSources;
    ptscTCPWM->unCTRL.stcField.u1AUTO_RELOAD_CC1 = config->enableCompare1Swap;
    ptscTCPWM->unCC1.u32Register = config->compare1;
    ptscTCPWM->unCC1_BUFF.u32Register = config->compare1_buff;
    ptscTCPWM->unTR_IN_SEL1.stcField.u8CAPTURE1_SEL = config->capture1Input;
    ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2CAPTURE1_EDGE = config->capture1InputMode;
```

```
    status = CY_RET_SUCCESS;
  }
  return(status);
}

void Cy_Tcpwm_Counter_SetTROUT(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
  ptscTCPWM->unTR_OUT_SEL.stcField.u3OUT0 = 2;     //(7) Configure Output Trigger Selection for
TCPWM#0,1 (16-bit)
  ptscTCPWM->unTR_OUT_SEL.stcField.u3OUT1 = 3;
}

void Cy_Tcpwm_Counter_SetCompare0(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM,  uint32_t compare0)
{
  ptscTCPWM->unCC0.u32Register = compare0;
}
//(8) Set Compare Value of Compare0 for TCPWM#0,1 (16-bit)
void Cy_Tcpwm_Counter_Enable(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
  ptscTCPWM->unCTRL.stcField.u1ENABLED = 0x1;     //(9) Enable TCPWM#0,1 (16-bit)
}

cy_en_trigmux_status_t Cy_TrigMux_SwTrigger(uint32_t trigLine, en_trig_type_t trigType,
uint32_t outSel)
{
  cy_en_trigmux_status_t retVal = CY_TRIGMUX_INVALID_STATE;

  if (PERI->unTR_CMD.stcField.u1ACTIVATE == 0)
  {
    PERI->unTR_CMD.stcField.u8TR_SEL     = (trigLine & CY_TR_MASK) >> CY_TR_SHIFT; // Select
activated trigger (0)
    PERI->unTR_CMD.stcField.u5GROUP_SEL  = (trigLine & CY_TR_GROUP_MASK) >>
CY_TR_GROUP_SHIFT; // Select trigger
    group (4)
    PERI->unTR_CMD.stcField.u1TR_EDGE    = trigType; // Select edge sensitive trigger as
trigger type
    PERI->unTR_CMD.stcField.u1OUT_SEL    = outSel; // Select activated trigger as output
trigger
    PERI->unTR_CMD.stcField.u1ACTIVATE   = 1; // Activate trigger     //(10) Configure and
Activate SW Trigger on TCPWM#0,1 (16-bit)
    retVal = CY_TRIGMUX_SUCCESS;
  }
  return retVal;
}
```

注: プログラムコードのグレーアウトされたセクションは、このアプリケーションノートでは説明されません。詳細については、アーキテクチャ TRM を参照してください。

3.4 SW トリガによる TCPWM タイマの同時開始

ここでは、SW トリガが複数の TCPWM カウンタを同時に開始する方法について説明します。

3.4.1 SW トリガによる TCPWM タイマの同時開始のユースケースの説明

図 20 に、CYT2B7 シリーズでの SW トリガの適用例を示します。120 度整流制御では、3 つの TCPWM 16 ビットモータ制御カウンタ cc チャネル 0, 1, および 2 が SW によって同時にトリガされます。

この場合、チャネル 0, チャネル 1, およびチャネル 2 は、それぞれ U 相, V 相, および W 相を表します。図 21 に動作を示します。トリガ MUX グループ 4 と TCPWM 16 ビットモータ制御カウンタ間の接続を図 22 に示します。TCPWM の詳細については、アーキテクチャ TRM の「TCPWM」の章を参照してください。

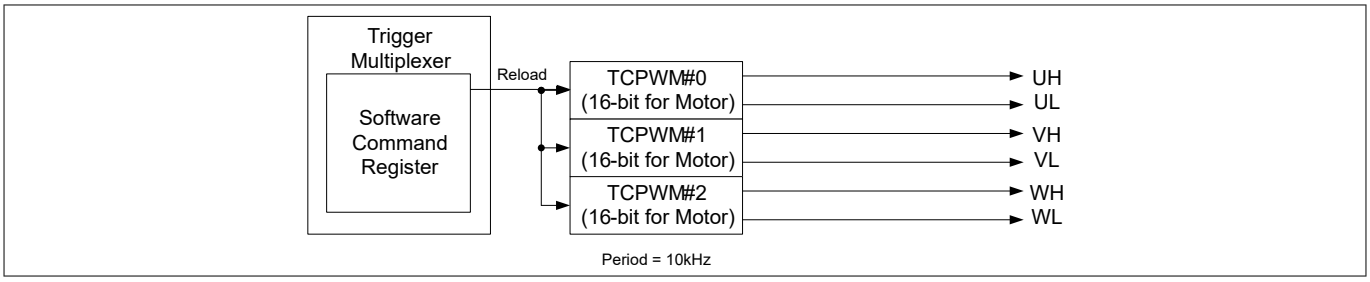

図 20 CYT2B7 シリーズの SW トリガの例

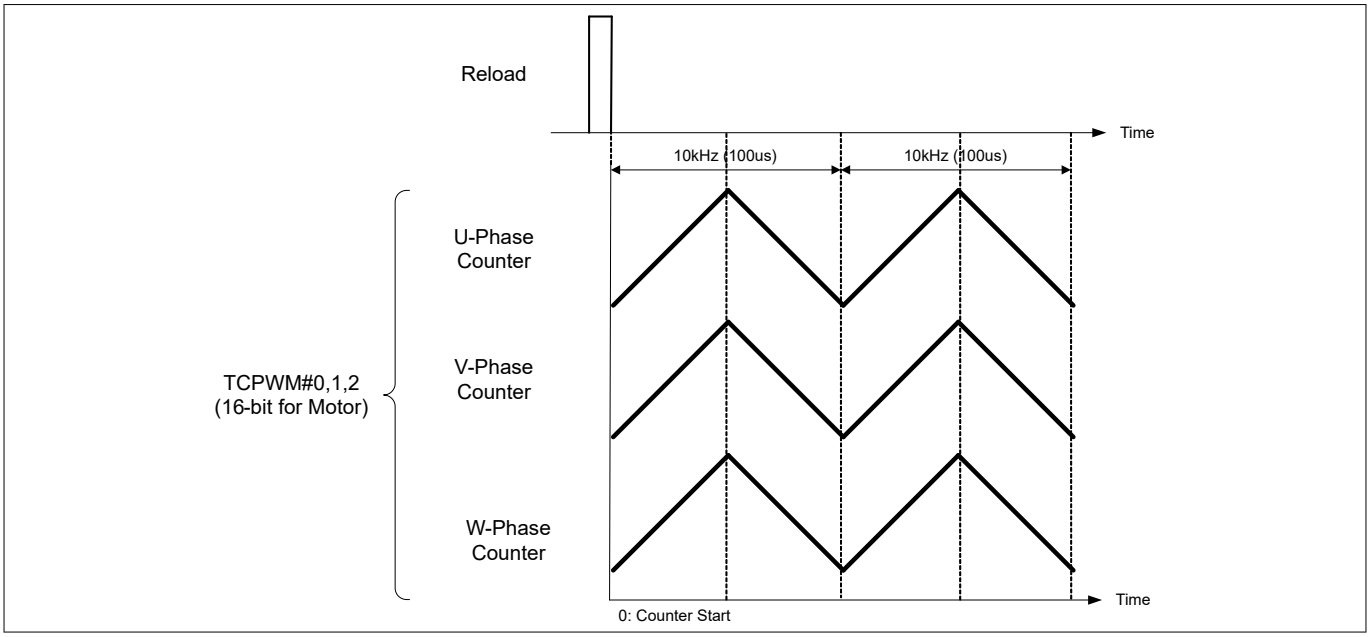

図 21 動作例: SW トリガによる TCPWM タイマの同時開始

3 アプリケーション

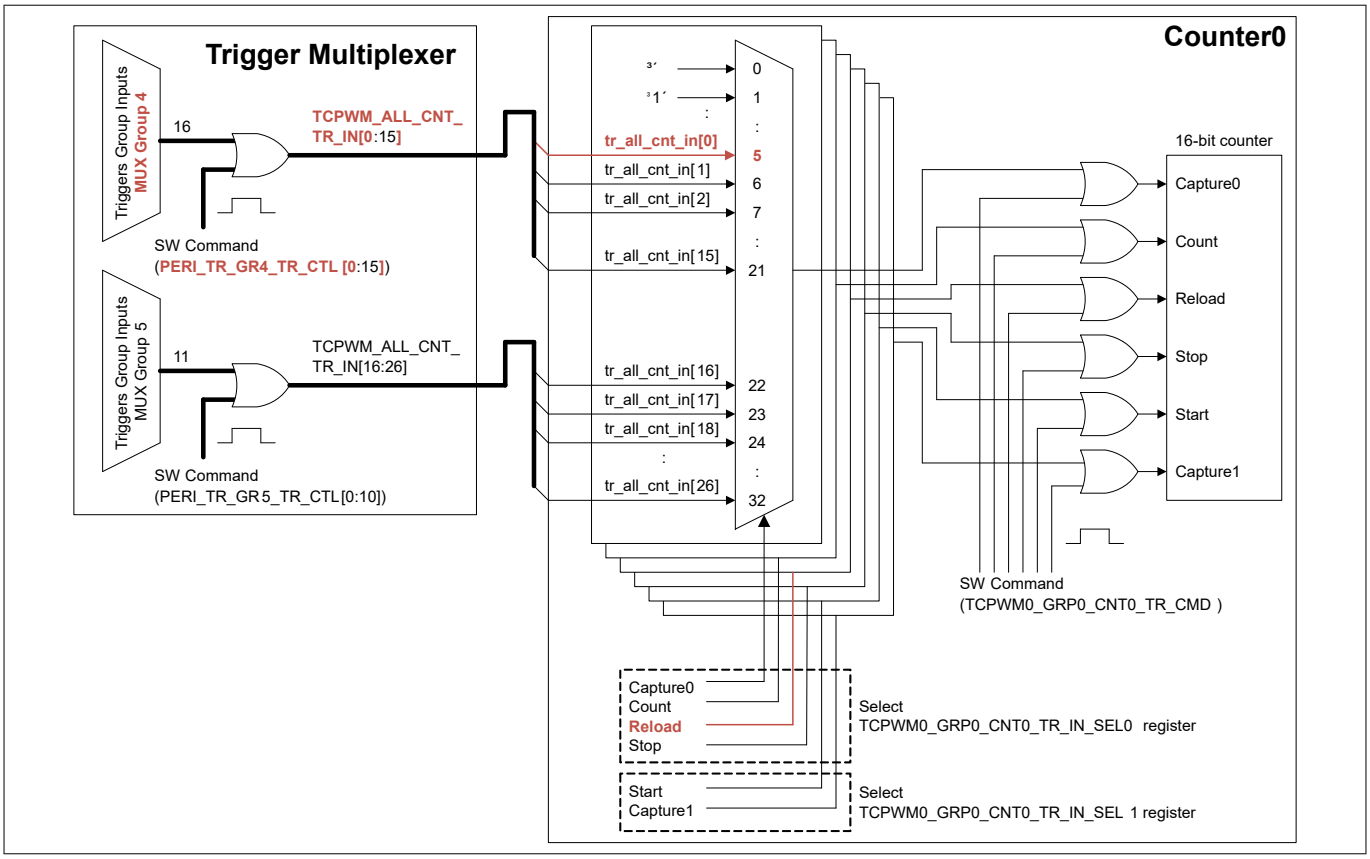

図 22　　MUX グループ 4 と TCPWM 16 ビットモータ制御カウンタ間の接続

このアプリケーションを実装するには、トリガを設定する必要があります。図 23 にトリガの設定手順を示します。

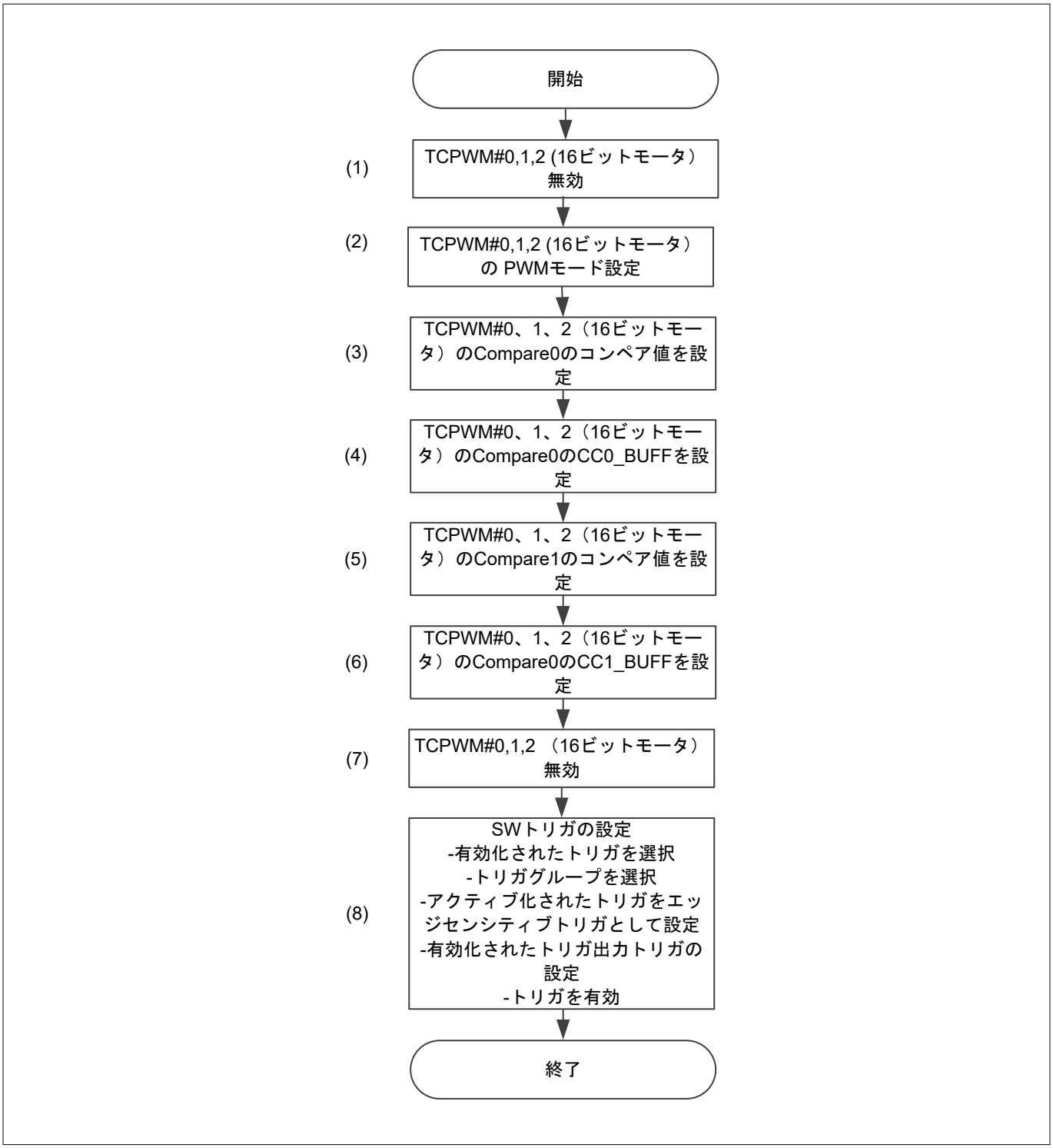

図 23　　CYT2B7 シリーズの 1-to-1 トリガの設定

ステップ (8) の SW トリガの設定は以下のとおりです。

トリガマルチプレクサの設定

表 15 に、PERI_TR_CMD.GROUP_SEL (=i) および PERI_TR_CMD.TR_SEL (=k) に設定する値を示します。この表は、トリガグループと出力トリガを提供する CYT2B7 シリーズデータシートからの抜粋です。

表 15　　CYT2B7 シリーズの MUX グループ 4 のトリガ出力

出力 (「k」)	トリガラベル	説明
MUX グループ 4: TCPWM_OUT (TCPWM0 から P-DMA0 へのトリガマルチプレクサ)		
0:15	TCPWM_ALL_CNT_TR_IN[0:15]	全カウンタのトリガ入力

各シリーズのトリガグループ出力の表については、デバイスのデータシートを参照してください。
図 24 に、TCPWM カウンタをトリガする SW を示します。

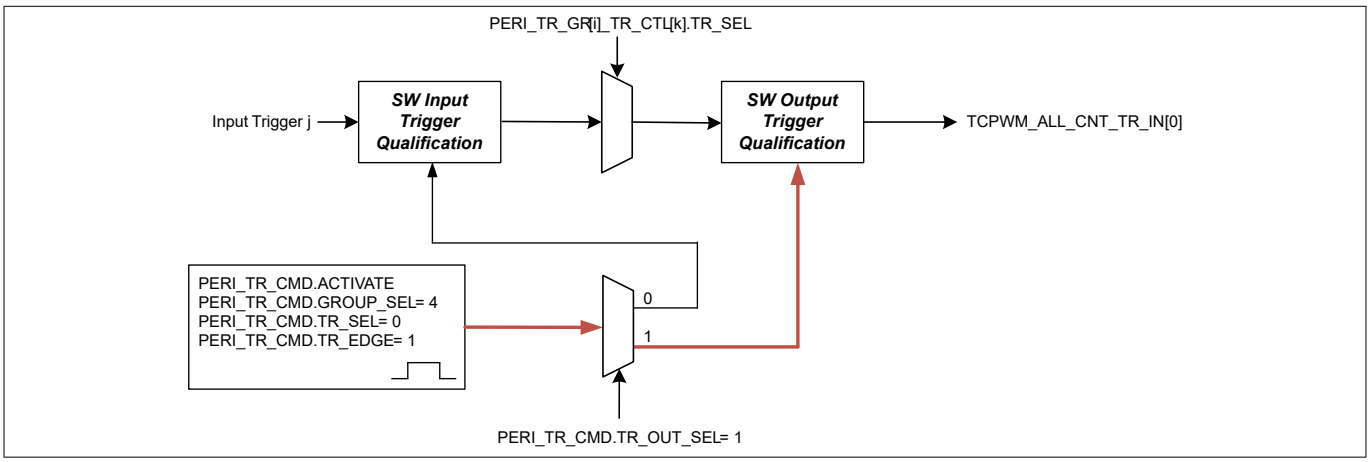

図 24　　TCPWM カウンタをトリガする SW

次に、TCPWM カウンタのタイマを開始する方法について説明します。

1. アクティブ化されたトリガの選択
   PERI_TR_CMD.TR_SEL = **0**: 出力トリガ TCPWM_ALL_CNT_TR_IN[0] (k ＝ 0)
   PERI_TR_CMD.GROUP_SEL = **4**: グループトリガの MUX グループ 4 (i ＝ 4)
   PERI_TR_CMD.TR_EDGE = **1**: エッジセンシティブトリガ
2. アクティブ化されたトリガの指定
   PERI_TR_CMD.OUT_SEL = **1**: 出力トリガ
3. SW トリガをアクティブ化します。
   PERI_TR_CMD.ACTIVATE = **1**

3.4.2　　SW トリガによる TCPWM タイマの同時開始のサンプルプログラム

Code Listing 4 は、図 23 のサンプルプログラムを示します。

Code Listing 4 SW トリガによる TCPWM タイマの同時開始の例

```
void Cy_Tcpwm_Counter_Disable(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
  ptscTCPWM->unCTRL.stcField.u1ENABLED = 0x00;      //(1) Disable TCPWM#0,1,2 (16-bit for Motor)
}

uint32_t Cy_Tcpwm_Pwm_Init(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM, cy_stc_tcpwm_pwm_config_t
const *config)
{
  uint32_t status = CY_RET_BAD_PARAM;

  if ((NULL != ptscTCPWM) && (NULL != config))
  {

    {
      ptscTCPWM->unCTRL.stcField.u1ONE_SHOT = config->runMode;      //(2) Configure TCPWM#0,1,2
(16-bit for Motor) for PWM Mode
      ptscTCPWM->unCTRL.stcField.u2UP_DOWN_MODE = config-> countDirection;
      ptscTCPWM->unCTRL.stcField.u3MODE = config->pwmMode;
      ptscTCPWM->unCTRL.stcField.u1DBG_FREEZE_EN = config->debug_pause;
      ptscTCPWM->unCTRL.stcField.u1AUTO_RELOAD_CC0 = config->enableCompare0Swap;
      ptscTCPWM->unCTRL.stcField.u1AUTO_RELOAD_PERIOD = config->enablePeriodSwap;
      ptscTCPWM->unCTRL.stcField.u1AUTO_RELOAD_LINE_SEL = config->enableLineSelSwap;
      ptscTCPWM->unCTRL.stcField.u1PWM_SYNC_KILL = config->killMode;
      ptscTCPWM->unCTRL.stcField.u1PWM_STOP_ON_KILL = (config->killMode >> 1);
    }
    if(config->pwmMode == CY_TCPWM_PWM_MODE_DEADTIME)
    {
      ptscTCPWM->unDT.stcField.u8DT_LINE_OUT_L = config->deadTime;
    }
    if (CY_TCPWM_COUNTER_COUNT_UP == config->runMode)
    {
      ptscTCPWM->unCOUNTER.u32Register = CY_TCPWM_CNT_UP_INIT_VAL;
    }
    ptscTCPWM->unCC0.u32Register = config->compare0;
    ptscTCPWM->unCC0_BUFF.u32Register = config->compare0_buff;
    ptscTCPWM->unPERIOD.u32Register = config->period;
    ptscTCPWM->unPERIOD_BUFF.u32Register = config->period_buff;
    {
      ptscTCPWM->unTR_IN_SEL0.stcField.u8CAPTURE0_SEL = config->switchInput;
      ptscTCPWM->unTR_IN_SEL0.stcField.u8RELOAD_SEL = config->reloadInput;
      ptscTCPWM->unTR_IN_SEL0.stcField.u8STOP_SEL = config->kill0Input;
      ptscTCPWM->unTR_IN_SEL0.stcField.u8COUNT_SEL = config->countInput;
    }

    ptscTCPWM->unTR_IN_SEL1.stcField.u8START_SEL = config->startInput;

    {
      ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2CAPTURE0_EDGE = config->switchInputMode;
      ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2RELOAD_EDGE = config->reloadInputMode;
      ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2START_EDGE = config->startInputMode;
      ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2STOP_EDGE = config->kill0InputMode;
```

```
        ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2COUNT_EDGE = config->countInputMode;
    }

    ptscTCPWM->unINTR_MASK.u32Register = config->interruptSources;

    ptscTCPWM->unTR_PWM_CTRL.stcField.u2CC0_MATCH_MODE =  config->Cc0MatchMode;
    ptscTCPWM->unTR_PWM_CTRL.stcField.u2OVERFLOW_MODE = config->OverflowMode;
    ptscTCPWM->unTR_PWM_CTRL.stcField.u2UNDERFLOW_MODE = config->UnderflowMode;

    #if defined (tviibe1m) || defined (tviibe2m) || defined (tviibh4m)
    if( (ptscTCPWM == TCPWM0_GRP1_CNT0) || (ptscTCPWM == TCPWM0_GRP1_CNT1) || (ptscTCPWM ==
TCPWM0_GRP1_CNT2) ||
    (ptscTCPWM == TCPWM0_GRP1_CNT3) || (ptscTCPWM == TCPWM0_GRP1_CNT4) || (ptscTCPWM ==
TCPWM0_GRP1_CNT5) ||
    (ptscTCPWM == TCPWM0_GRP1_CNT6) || (ptscTCPWM == TCPWM0_GRP1_CNT7) || (ptscTCPWM ==
TCPWM0_GRP1_CNT8) ||
    (ptscTCPWM == TCPWM0_GRP1_CNT9) || (ptscTCPWM == TCPWM0_GRP1_CNT10) || (ptscTCPWM ==
TCPWM0_GRP1_CNT11) )

    {
      ptscTCPWM->unCTRL.stcField.u1AUTO_RELOAD_CC1 = config->enableCompare1Swap;
      ptscTCPWM->unTR_IN_SEL1.stcField.u8CAPTURE1_SEL = config->kill1Input;
      ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2CAPTURE1_EDGE = config->kill1InputMode;
      ptscTCPWM->unTR_PWM_CTRL.stcField.u2CC1_MATCH_MODE =  config->Cc1MatchMode;

      ptscTCPWM->unDT.stcField.u16DT_LINE_COMPL_OUT = config->deadTimeComp;
      ptscTCPWM->unDT.stcField.u8DT_LINE_OUT_H = (config->deadTime >> 8);
    }
    status = CY_RET_SUCCESS;

  }
  return(status);
}

void Cy_Tcpwm_Counter_SetCompare0(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM,  uint32_t
compare0)     //(3) Set Compare Value of Compare0 for TCPWM#0,1,2 (16-bit for Motor)
{
  ptscTCPWM->unCC0.u32Register = compare0;
}

void Cy_Tcpwm_Counter_SetCompare0_Buff(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM,  uint32_t
compare1)     //(4) Set Compare Value of CC0_BUFF for TCPWM#0,1,2 (16-bit for Motor)
{
  ptscTCPWM->unCC0_BUFF.u32Register = compare1;
}

void Cy_Tcpwm_Counter_SetCompare1(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM,  uint32_t
compare0)     //(5) Set Compare Value of Compare1 for TCPWM#0,1,2 (16-bit for Motor)
{
  ptscTCPWM->unCC1.u32Register = compare0;
}

void Cy_Tcpwm_Counter_SetCompare1_Buff(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM,  uint32_t
```

```
compare1)     //(6) Set Compare Value of CC1_BUFF for TCPWM#0,1,2 (16-bit for Motor)
{
  ptscTCPWM->unCC1_BUFF.u32Register = compare1;
}

void Cy_Tcpwm_Pwm_Enable(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)     //(7) Enable TCPWM#0,1,2
(16-bit for Motor)
{
  ptscTCPWM->unCTRL.stcField.u1ENABLED = 0x1;
}

cy_en_trigmux_status_t Cy_TrigMux_SwTrigger(uint32_t trigLine, en_trig_type_t trigType,
uint32_t outSel)
{
  cy_en_trigmux_status_t retVal = CY_TRIGMUX_INVALID_STATE;

  if (PERI->unTR_CMD.stcField.u1ACTIVATE == 0)
  {
    PERI->unTR_CMD.stcField.u8TR_SEL     = (trigLine & CY_TR_MASK) >> CY_TR_SHIFT; // Select
activated trigger (0)
    PERI->unTR_CMD.stcField.u5GROUP_SEL  = (trigLine & CY_TR_GROUP_MASK) >>
CY_TR_GROUP_SHIFT; // Select trigger group (4)
    PERI->unTR_CMD.stcField.u1TR_EDGE    = trigType; // Select edge sensitive trigger as
trigger type
    PERI->unTR_CMD.stcField.u1OUT_SEL    = outSel; // Select activated trigger as output
trigger
    PERI->unTR_CMD.stcField.u1ACTIVATE   = 1; // Activate trigger     //(8) Configure SW Trigger

    retVal = CY_TRIGMUX_SUCCESS;
  }
  return retVal;
}
```

注: *プログラムコードのグレーアウトされたセクションは、このアプリケーションノートでは説明されません。詳細については、アーキテクチャ TRM を参照してください。*

4 用語集

表 16 用語集

用語	説明
MUX	Multiplexer (マルチプレクサ)
OV	Overflow (オーバーフロー)
P-DMA	Peripheral Direct Memory Access (ペリフェラルダイレクトメモリアクセス)
SAR ADC	Successive Approximation Register Analog-to-Digital Converter (逐次比較型アナログ デジタル コンバータ)
SCB	Serial Communications Block (シリアル通信ブロック)
SW	Software (ソフトウェア)
TCPWM	Timer, Counter, and Pulse Width Modulator (タイマ/カウンタ/パルス幅変調器)
UH	U-phase PWM waveform output (U 相 PWM 波形出力)
UL	U-phase PWM complementary waveform output (U 相 PWM 相補波形出力)
VH	V-phase PWM waveform output (V 相 PWM 波形出力)
VL	V-phase PWM complementary waveform output (V 相 PWM 相補波形出力)
WH	W-phase PWM waveform output (W 相 PWM 波形出力)
WL	W-phase PWM complementary waveform output (W 相 PWM 相補波形出力)

5 関連ドキュメント

以下は TRAVEO™ T2G ファミリのデータシートおよびテクニカルリファレンスマニュアルです。これらドキュメントの入手についてはテクニカルサポートに連絡してください。

- デバイスデータシート
  - CYT2B7 datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family
  - CYT2B9 datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family
  - CYT4BF datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family
  - CYT6BJ datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family (Doc No. 002-33466)
  - CYT4DN datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family (Doc No. 002-24601)
  - CYT3BB/4BB datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family
  - CYT3DL datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family (Doc No. 002-27763)
  - CYT4EN datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family (Doc No. 002-30842)
  - CYT2CL datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family (Doc No. 002-32508)
- ボディコントローラ Entry ファミリ
  - TRAVEO™ T2G automotive body controller entry family architecture technical reference manual (TRM)
  - TRAVEO™ T2G automotive body controller entry registers technical reference manual (TRM) for CYT2B7
  - TRAVEO™ T2G automotive body controller entry registers technical reference manual (TRM) for CYT2B9
- ボディコントローラ High ファミリ
  - TRAVEO™ T2G automotive body controller high family architecture technical reference manual (TRM)
  - TRAVEO™ T2G Automotive body controller high registers technical reference manual (TRM) for CYT4BF
  - TRAVEO™ T2G automotive body controller high registers technical reference manual (TRM) for CYT3BB/4BB
  - TRAVEO™ T2G automotive body controller high registers technical reference manual (TRM) for CYT6BJ (Doc No. 002-36068)
- クラスタ 2D ファミリ
  - TRAVEO™ T2G automotive cluster 2D family architecture technical reference manual (TRM) (Doc No. 002-25800)
  - TRAVEO™ T2G automotive cluster 2D registers technical reference manual (TRM) for CYT4DN (Doc No. 002-25923)
  - TRAVEO™ T2G automotive cluster 2D registers technical reference manual (TRM) for CYT3DL (Doc No. 002-29584)
- クラスタ Entry ファミリ
  - TRAVEO™ T2G automotive cluster entry family architecture technical reference manual (TRM) (Doc No. 002-33175)
  - TRAVEO™ T2G automotive cluster entry registers technical reference manual (TRM) (Doc No. 002-33404)

6 その他の参考資料

インフィニオンは、さまざまな周辺機器にアクセスするためのサンプルソフトウェアとして、スタートアップを含むサンプルドライバライブラリ (SDL) を提供します。SDL は、公式の AUTOSAR 製品でカバーされていないドライバーのために、顧客へのリファレンスとしても機能します。SDL は、自動車用 SW 開発プロセスを使用して開発されていないため、生産目的では使用できません。このアプリケーションノートのプログラムコードは、SDL の一部です。SDL を入手するには、テクニカルサポートに連絡してください。

改訂履歴

版数	発行日	変更内容
**	2020-09-23	このドキュメントは英語版 002-28104 Rev. **を翻訳した日本語版 002-31033 Rev.**です。
英語版*A	-	この版は英語版のみです。英語版の改訂内容: Updated to Infineon template.
英語版*B	-	この版は英語版のみです。英語版の改訂内容: Fixed trigger input to SCB0 RX.
*A	2023-07-25	このドキュメントは英語版 002-28104 Rev. *C を翻訳した日本語版 002-31033 Rev.*A です。英語版の改訂内容: Fixed footer.
*B	2024-07-03	このドキュメントは英語版 002-28104 Rev. *D を翻訳した日本語版 002-31033 Rev.*B です。英語版の改訂内容: Template update; no content updated.

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2024-07-03

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-nds1681443925270

重要事項

本手引書に記載された情報は、本製品の使用に関する手引きとして提供されるものであり、いかなる場合も、本製品における特定の機能性能や品質について保証するものではありません。本製品の使用の前に、当該手引書の受領者は実際の使用環境の下であらゆる本製品の機能及びその他本手引書に記された一切の技術的情報について確認する義務が有ります。インフィニオンテクノロジーズはここに当該手引書内で記される情報につき、第三者の知的所有権の不侵害の保証を含むがこれに限らず、あらゆる種類の一切の保証および責任を否定いたします。

本文書に含まれるデータは、技術的訓練を受けた従業員のみを対象としています。本製品の対象用途への適合性、およびこれら用途に関連して本文書に記載された製品情報の完全性についての評価は、お客様の技術部門の責任にて実施してください。

警告事項

技術的要件に伴い、製品には危険物質が含まれる可能性があります。当該種別の詳細については、インフィニオンの最寄りの営業所までお問い合わせください。

インフィニオンの正式代表者が署名した書面を通じ、インフィニオンによる明示の承認が存在する場合を除き、インフィニオンの製品は、当該製品の障害またはその使用に関する一切の結果が、合理的に人的傷害を招く恐れのある一切の用途に使用することはできないこと予めご了承ください。