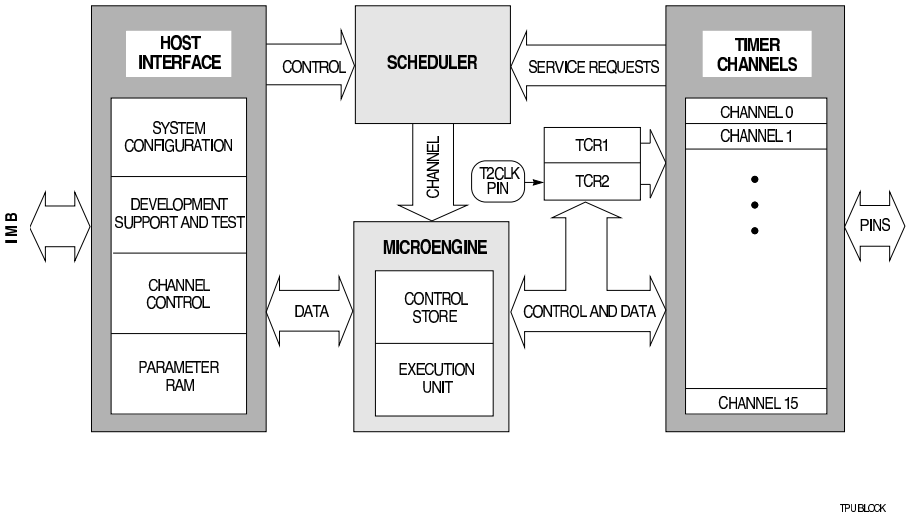
Programowanie funkcji TPU w mikrokodzie

(na podstawie “TPU microcoding for beginners”, Dyson A., Bannoura M., AMT, 1999)

Marek Wnuk
<marek.wnuk@pwr.wroc.pl>

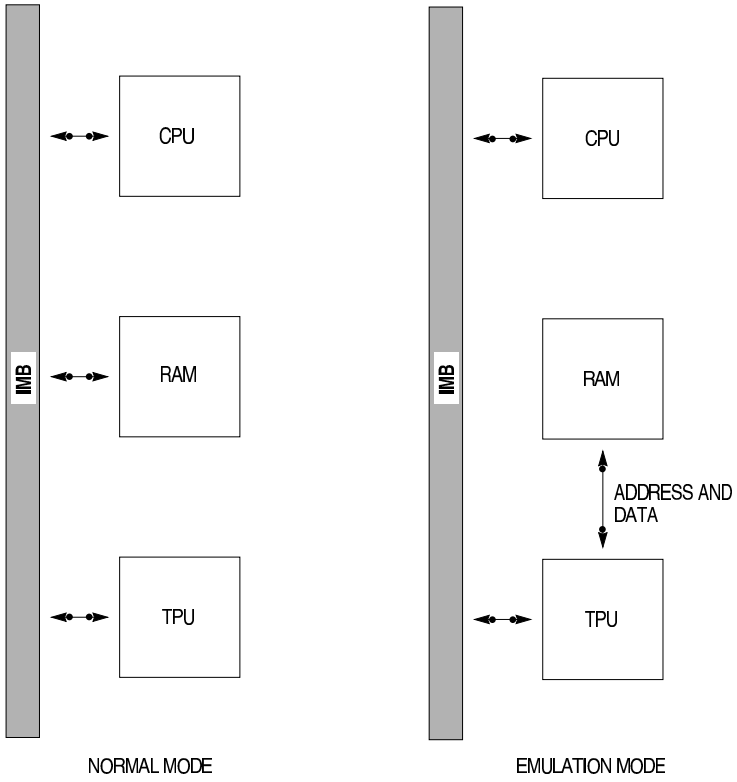
13 grudnia 2009

1 Wprowadzenie



Rysunek 1: Struktura bloku TPU

- Standardowe układy czasowe mikrokontrolerów (*timer-y*) w znacznym stopniu obciążają CPU.
- TPU (*Time Processor Unit*) jest sam w sobie procesorem, nie obciąża CPU obsługą funkcji czasowych (rys. 1).
- TPU jest becnie dostępny w kilku rodzinach mikrokontrolerów firmy Motorola (MC68HC16, MC68300, MPC500).
- TPU zawiera w *CONTROL STORE* zaprogramowany fabrycznie (maską) ROM z mikrokodem podstawowych funkcji czasowych (dla 68332 są dostępne dwie wersje maski : A (*Automotive*) i G (*General purpose*).

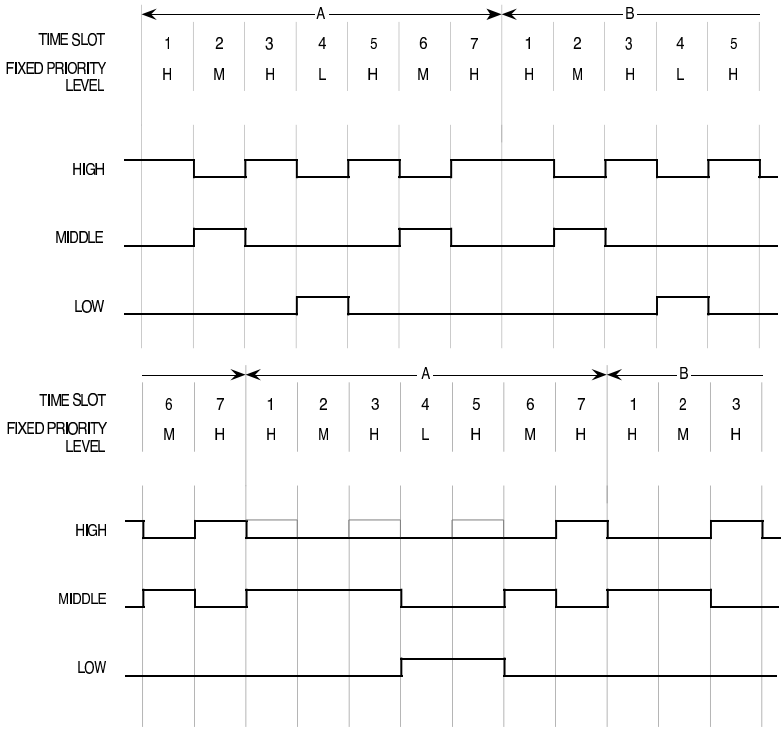


Rysunek 2: Emulacja mikrokodu w TPU

- Tryb emulacji TPU pozwala załadować własny mikrokod do pamięci RAM i używać go jako *CONTROL STORE* zamiast fabrycznego ROM.
- W celu użycia trybu emulacji należy wlączyć pamięć RAM, zapełnić ją mikrokodem, a następnie ustawić bit EMU w rejestrze sterującym TPU (rys 2).
- Użycie trybu emulacji nie spowalnia pracy TPU.
- Przy tworzeniu własnych funkcji w mikrokodzie należy zadbać o unikalność nazw (etykiet), by uniknąć problemów przy łączeniu z istniejącymi funkcjami.

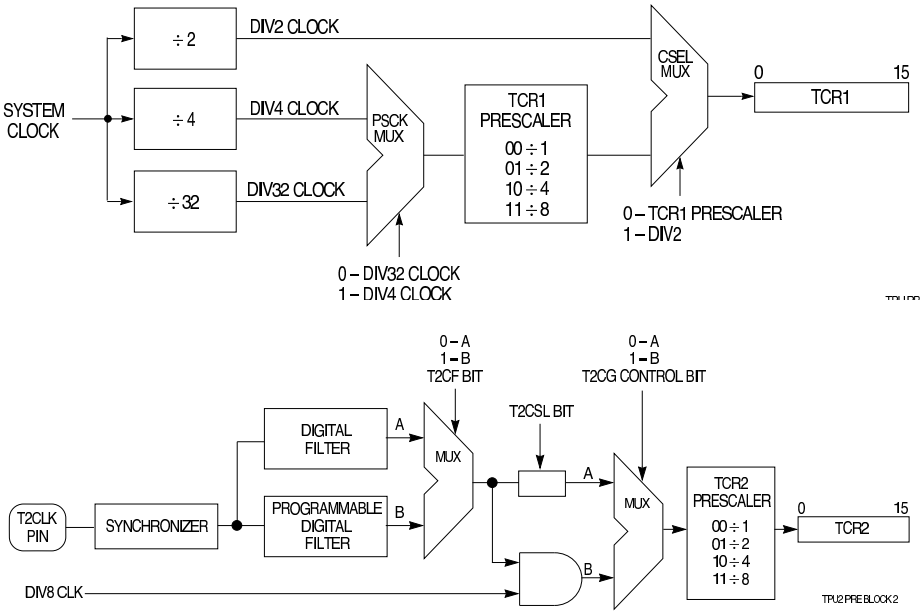
2 Właściwości

- TPU ma 16 kanałów. W każdym kanale można uruchomić dowolną funkcję czasową.
- TPU używa podziału czasu (*time slicing*) do równoczesnej obsługi wielu funkcji czasowych. Procesor TPU (*execution unit*) w każdym odcinku czasu wykonuje procedurę obsługującą jeden stan jednego kanału.
- Każdemu kanałowi można przypisać jeden z trzech poziomów priorytetu (wysoki, średni i niski). Kanały o wyższym priorytecie otrzymują częściej odcinki czasu procesora TPU (rys. 3).



Rysunek 3: Obsługa priorytetów w TPU

- W ramach grupy kanałów o równym priorytecie pierwszeństwo mają kanały o niższym numerze. Zasada rotacji priorytetu (*round robin*) zapewnia obsłużenie wszystkich kanałów.
- Pomiędzy odcinkami czasu przeznaczonymi na obsługę kanałów (*time slots*) występuje przerwa (10 taktów CPU) na przełączenie kontekstu procesora TPU (załadowanie informacji o kanale wybranym do obsługi).
- Każdy kanał ma swoje wyprowadzenie (*pin*) oraz rejestr zdarzeń (*event register*). Zdarzenia zrównania licznika (*match*) i zmiany sygnału wejściowego (*transition*) mogą być ustawiane niezależnie dla każdego kanału.
- W TPU są dostępne dwa zegary odniesienia (TCR1 i TCR2). Pozwala to na zaprogramowanie dwóch różnych prędkości działania przez CPU w celu obsłużenia zarówno szybkich, jak i wolnych zdarzeń (rys. 4).
- W TPU1 “przyszłość” jest zdefiniowana jako odcinek czasu od bieżącego stanu TCR do TCR+$8000. “Przeszłość” oznacza odcinek od $8001 do $FFFF w stosunku do bieżącego stanu TCR.
- W TPU2 i TPU3 można w każdym kanale wybrać liczenie czasu bez znaku (“przyszłość” od 0 do $FFFF od bieżącego stanu TCR), co odpowiada możliwości wyboru porównania typu *greater-than-or-equal-to* lub *equal-to*.



Rysunek 4: Układy generacji podstaw czasu dla TPU

- Liczba $8000 jest oznaczona w składni mikrokodu TPU ***max***, jako największy zakres "przyszłości" od bieżącego stanu TCR przy zastosowaniu porównania *greater-than-or-equal-to*.

3 Interfejs procesora nadrzędnego

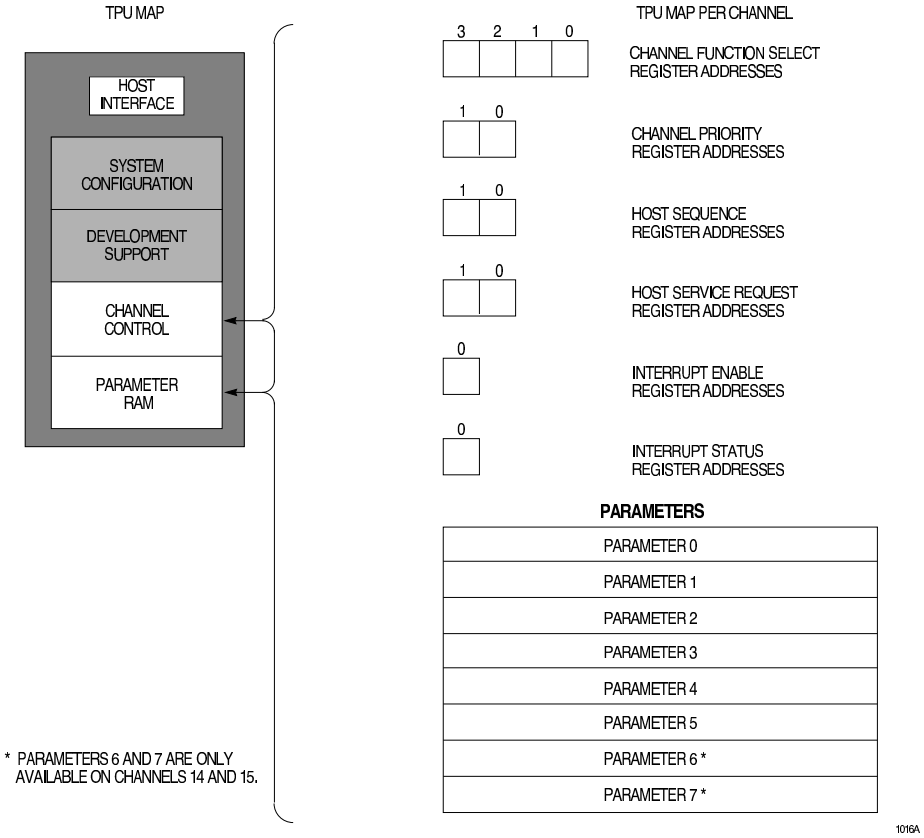
TPUMCR — TPU Module Configuration Register **$###E00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	TCR1P[1:0]		TCR2P[1:0]		EMU[1]	T2CG	STF	SUPV	PSCK	TPU2[2]	T2CSL[3]	IARB[3:0]			
RESET:															
0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

NOTES:
1. On TPU2, this bit is set or cleared according to the shadow bit for bit four of the flash EEPROM module configuration (FEEMCR) register.
2. After reset, the TPU2 enable (TPU2) bit is zero if TPU module is present. In this case, the bit cannot be modified. If the TPU2 module is present, the TPU2 enable bit is one after reset.
3. After reset, TCR2 counter clock edge (T2CSL) bit is zero if the TPU module is present. In this case, the bit cannot be modified. If the TPU2 module is present, this bit is zero after reset and can be modified.

Rysunek 5: Główny rejestr konfiguracyjny TPU

- Interfejs procesora nadrzędnego składa się z rejestrów TPU i pamięci RAM przeznaczonej na parametry funkcji uruchamianych w kanałach (*Parameter RAM*). Są one dostępne zarówno dla procesora TPU jak i dla jednostki nadrzędnej (CPU).



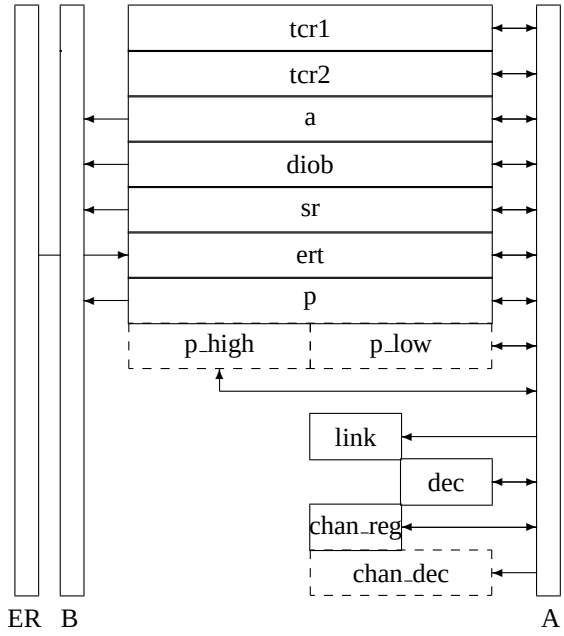
Rysunek 6: Sterowanie kanałem TPU

- Wszystkie rejestry TPU (oprócz statusu przerwań) muszą być adresowane przez CPU jako 16-bitowe słowa.
- Aby zainicjalizować pracę TPU należy najpierw skonfigurować TPU jako część mikrokontrolera (rys. 5), a następnie skonfigurować poszczególne kanały TPU (rys. 6).
- Podstawowe kroki konfigurowania TPU:

  1. Określenie podstawowych własności TPU w głównym rejestrze konfiguracyjnym TPUMCR (*TPU master configuration*).
  2. Ustalenie wektorów i poziomów obsługi przerwań w rejestrze konfiguracji przerwań TICR (*interrupt configuration*).
  3. Zezwolenie na zgłaszanie przerwań przez poszczególne kanały w rejestrze zezwoleń CIER (*channel interrupt enable*).
  4. Ustawienie wybranych funkcji w rejestrach wyboru funkcji kanałowych CFSR (*channel function select*).
  5. Wybranie opcji dla funkcji kanałowych w rejestrach HSQR (*host sequence*).

6. Wpisanie odpowiednich parametrów do pamięci RAM kanałów (*parameter RAM*).
7. Ustawienie żądania obsługi kanałów w HSRR (*host service request*).
8. Ustawienie priorytetów kanałów w CPR (*channel priority*).
9. Sprawdzenie przyjęcia przez TPU żądania obsługi kanałów (sygnalizowane wyzerowaniem odpowiedniego pola w rejestrze HSRR).

4 Przegląd mikrokodu TPU



Rysunek 7: Struktura wewnętrzna procesora TPU

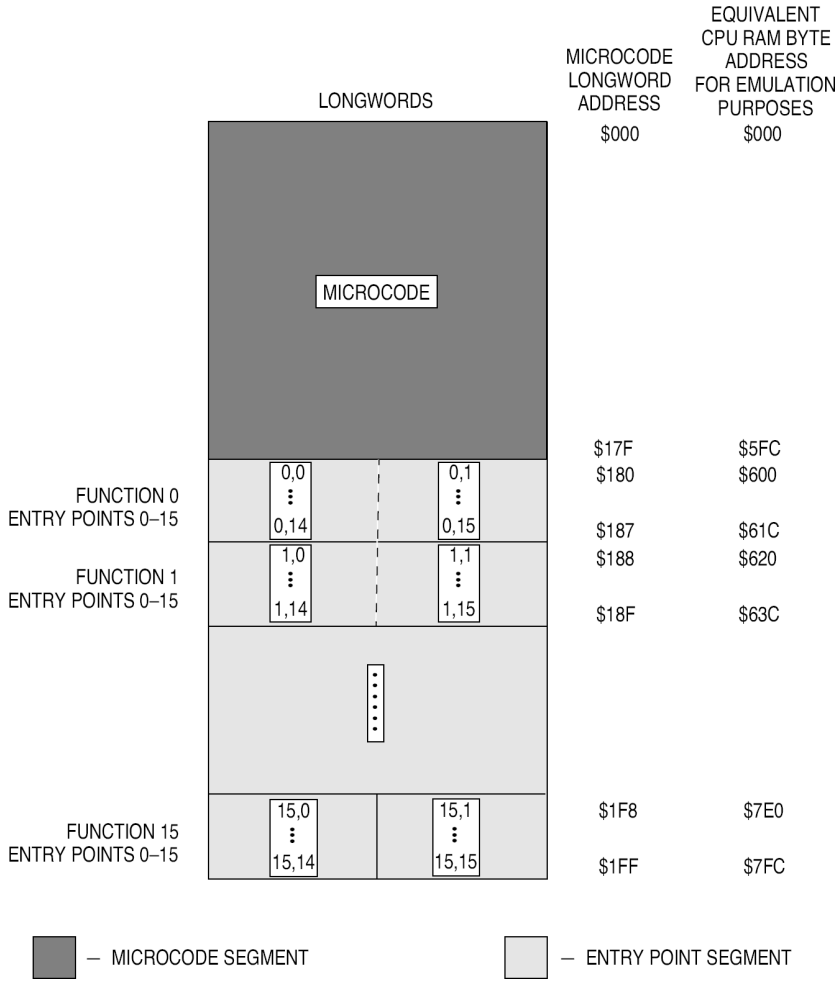
- Sprzęt TPU jest sterowany przez mikrokod.
- Mikroinstrukcja jest zbudowana z podkomend (*subcommands*), a podkomenda jest złożona z pól (*fields*).
- Mikroinstrukcje są oddzielone kropkami.
- Podkomendy są oddzielone średnikami.
- Pola są oddzielone przecinkami.
- Istnieją cztery rodzaje podkomend:
  - sterowania kanałem (*channel control*) oznaczone `chan`,

  - procesora TPU (*execution unit*) oznaczone `au`,
  - pamięci RAM oznaczone `ram`,
  - sterowania wykonaniem mikrokodu (*microengine/sequencing*) bez oznaczenia typu.

- Wszystkie mikroinstrukcje są 32-bitowe.
- Mikroinstrukcje są wykonywane w dwóch taktach CPU podzielonych na cztery stany: T1, T2, T3 i T4.
- Istnieje 5 formatów mikroinstrukcji.
- Rejestry procesora TPU są wspólne dla wszystkich kanałów. Dane muszą być przechowywane w parametrach w pamięci RAM.
- Rejestry TPU to: `tcr1, tcr2, a, diob, sr, ert, p, p_high, p_low, link, dec, chan_reg, chan_dec` (rys. 7).

5 Punkty startowe obsługi stanów

- Punkt startowy (*entry point*) zawiera adres w pamięci mikrokodu, pod którym rozpoczyna się procedura obsługi stanu danej funkcji czasowej.
- Punkty startowe są zebrane w tablicy (256 elementów, po 16 punktów startowych dla każdej funkcji czasowej) (rys. 8).
- Punkt startowy jest wybierany podczas przełączania kontekstu procesora TPU (*time slot transition*).
- Wybór punktu startowego jest określony przez następujące warunki:
  - kanał wybrany do obsługi,
  - funkcja wykonywana w wybranym kanale,
  - typ żądania w wybranym kanale,
  - stan wyprowadzenia wybranego kanału (*pin state*),
  - wartośc znacznika `flag0` w wybranym kanale.
- Typy żądań w kanale to (rys. 9):
  - żądanie od jednostki nadrzędnej (*host request*),
  - żądanie sprzętowe (*match/transition*),
  - żądanie od innego kanału TPU (*link*).
- Żądania od jednostki nadrzędnej mają wyższy priorytet niż pozostałe.
- Jeśli w funkcji nie wykorzystano wszystkich dostępnych punktów startowych, należy nieużywane punkty skierować do procedur pustych (`end_of_link`, `end_of_phase` - w zależności od tego, czy otrzymują żądania od innych kanałów).
- Trzy powszechne błędy przy definiowaniu punktów startowych:



Rysunek 8: Pamięć mikrokodu TPU

1. brak `x` dla żądań jednostki nadrzędnej (*host service*),
2. brak `0` dla `hsr0` i `hsr1` przy żądaniach `m/tsr` i `lsr`,
3. umieszczenie warunku w więcej niż jednym punkcie startowym (musi być dokładnie 16 punktów startowych dla każdej funkcji).

6 Dostęp do pamięci RAM

- Jednostka nadrzędna (CPU) i procesor TPU mogą się komunikować przez pamięć parametrów kanałowych (*parameter RAM*).
- W TPU1 kanały od 0 do 13 mają po 6 parametrów (słów 16-bitowych), a kanały 14 i 15 – po 8. W TPU2 i TPU3 wszystkie kanały mają po 8 parametrów.
- Ograniczenie ilości parametrów tworzonej funkcji do sześciu pozwala ją wykorzystać w dowolnym kanale dowolnego TPU.

Entry Points	Host Service Request (HSR)	Link Service Request (LSR)	Match/ Transition SvcReq (M/TSR)	Pin State	Channel Flag 0
0	01	x	x	0	x
1	01	x	x	1	x
2	10	x	x	x	x
3	11	x	x	x	x
4	00	0	1	0	0
5	00	0	1	0	1
6	00	0	1	1	0
7	00	0	1	1	1
8	00	1	0	0	0
9	00	1	0	0	1
10	00	1	0	1	0
11	00	1	0	1	1
12	00	1	1	0	0
13	00	1	1	0	1
14	00	1	1	1	0
15	00	1	1	1	1

Rysunek 9: Punkty startowe i stany funkcji TPU

- Dostęp do RAM jest dozwolony tylko przez slowa 16-bitowe.
- Tryby adresowania RAM:
  - względne w kanale (*channel relative*):

    `ram p <- prm0`

    – załadowanie parametru 0 z bieżącego kanału (wskazanego przez `chan_reg`) `do rejestru p)`,
  - `bezpośrednie (absolutne` (*direct*):

    `ram p <- (14,5)`

    lub

    `ram p <- \$EA`

    – załadowanie parametru 5 z kanału 14,
  - pośrednie przez rejestr `diob` (*indirect*):

    `ram p <- by_diob`

    – załadowanie parametru wskazywanego przez rejestr `diob`.
- Wynik operacji arytmetycznej jest zapisywany w fazie T3 mikrocyklu. Przekazywanie danych pomiędzy RAM a rejestrami odbywa się w fazie T4.

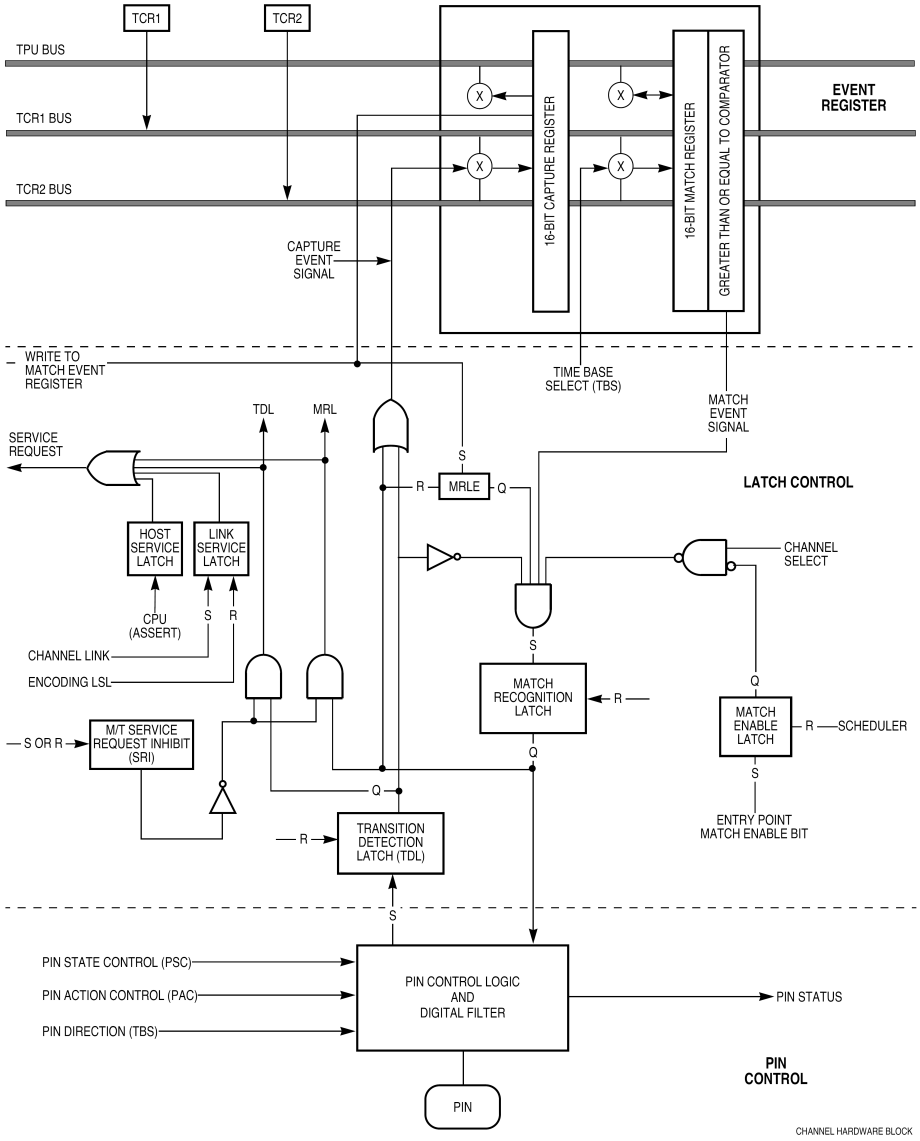
- Rejestr `diob` musi być ustawiony w mikroinstrukcji wcześniejszej niż ta, w której jest użyty do adresowania pośredniego.
- Dla zachowania spójności danych 32-bitowych CPU musi czytać/pisać parametry w jednej istrukcji, jako liczby typu *long*.
- Procesor TPU powinien używać dwóch sąsiadujących instrukcji z 16-bitowym dostępem poprzedzonych mikroinstrukcją bez dostępu do RAM.

7 Sterowanie kanałem

TBS	PAC	PSC	Input	Output
		00 01 10 11		pin:=pac pin:=high pin:=low nil
	000 001 010 011 1xx		pac:=no_detect pac:= low_high pac:=high_low pac:=any_transition nil	pac:=no_change pac:=high pac:=low pac:=toggle nil
0000 0001 0010 0011			tbs:=in_m1_c1 tbs:=in_m1_c2 tbs:=in_m2_c1 tbs:=in_m2_c2	
0100 0101 0110 0111				tbs:=out_m1_c1 tbs:=out_m1_c2 tbs:=out_m2_c1 tbs:=out_m2_c2
1xxx			no change	no change

Rysunek 10: Standardowe sterowanie kanałem TPU

- Każdy kanał jest wyposażony w rejestr zdarzeń zawierający rejestr porównywania (`mer` - *match enable register*), rejestr zatrzaskowy (*capture register*) i komparator.
- Dostęp do rejestru zdarzeń jest możliwy tylko za pośrednictwem tymczasowego rejestru zdarzeń (`ert` - *event registr temporary*):
  - aby wpisać wartość do `mer` należy ją wpisać do `ert` i użyć podkomendy `chan write_mer`,
  - wartość z rejestru zatrzaskowego jest przepisywana do `ert` w czasie przełączania kontekstu procesora TPU (*slot transition*),
  - podkomenda `chan read_mer` ładuje do `ert` zawartość `mer`, dzięki czemu można ją odczytać.



Rysunek 11: Struktura sprzętowa kanału TPU

- TBS (*time base select*), PAC (*pin action control*) i PSC (*pin state control*) są polami bitowymi, które pozwalają sterować zachowaniem wyprowadzenia kanału:

```
chan tbs := out_m1_c1, (*wyjscie, komparator tcr1, zatrzask tcr2*)
     pin := high,      (*stan poczatkowy wyjscia wysoki*)
     pac := low.       (*zrownanie powoduje stan niski wyjscia*)
```

- Podkomenda `chan config := p` pozwala zadać łącznie TBS, PAC i PSC przez parametr (rys. 10). Mikrokod ładuje parametr do `p` i następnie wykonuje podkomendę

```
chan config := p.
```

- Podkomenda `chan enable_mtsr` zezwala na zgłaszanie żądań od `mer` w kanale, a `chan disable_mtsr` – zabrania (rys. 11).

- Zdarzenie wykrycia zbocza na wejściu (*transition event*):

  – zdarzenie jest rejestrowane w chwili wykrycia zaprogramowanego w PAC zbocza sygnału wejściowego;

  – czas z wybranego w TBS zegara jest zatrzaskiwany w rejestrze i ustawiany jest przerzutnik TDL (*transition detection latch*); do czasu skasowania TDL przez mikrokod nie zmienia sie zatrzaśnięta wartość;

  – jeśli jest zezwolenie na żądanie obsługi (`chan enable_mtsr`), wystawiane jest żądanie obsługi `m/tsr`;

  – TDL blokuje ustawianie przerzutnika komparatora (MRL);

  – w procedurze obsługi zdarzenia zrównania (*match event*) należy zgasić TDL podkomendą `chan neg_tdl`.

- Zdarzenie zrównania komparatora z TCR (*match event*):

  – zdarzenie zrównania (*match event*) uaktywnia się każdorazowo poprzez wpisanie do `mer` zadanej wartości czasu (za pośrednictwem `ert`)

  ```
  chan write\_mer;
  ```

  – w chwili zrównania się wartości `mer` i wybranego zegara TCR następuje ustawienie przerzutnika MRL (o ile inne warunki na to pozwalają – patrz: ustawiony TDL);

  – jeśli jest zezwolenie na żądanie obsługi (`chan enable_mtsr`), jest wystawiane żądanie obsługi `m/tsr`;

  – zdarzenie zrównania powoduje również zatrzaśnięcie wybranego zegara TCR;

  – zaleca się zakazanie dostrzegania zdarzenia zrównania na czas obsługi kanału poprzez komendę `disable_match` w punkcie startowym.

- Przekazanie żądania do innego kanału (*link to a channel*) wymaga wydania podkomendy `au link := numer_kanalu`. Procedura obsługująca to żądanie powinna zawierać podkomendę `chan neg_lsl`.

- Zaleca się inicjalizowanie sprzętu w kanałach przy pomocy mikrokodu, nie polegając na sprzętowym resecie. Pozwala to na poprawną pracę funkcji niezależnie od warunków startowych.
- TDL blokuje ustawianie MRL, co pozwala mierzyć długie impulsy. Jeśli zarówno TDL jak i MRL są ustawione, to MRL było ustawione przed TDL i zatrzask zawiera czas wystąpienia zbocza.
- Znaczniki `flag0`, `flag1` (oraz `flag2` w TPU2 i TPU3) mogą być używane dowolnie. Dodatkowo, `flag0` jest warunkiem wybierającym punkt startowy funkcji czasowej.
- Podkomenda `chan cir` ustawia bit odpowiadający bieżącemu kanałowi w rejestrze zgłaszania przerwań CISR (*channel interrupt status*).

8 Jednostka arytmetyczna

- Ogólna postać składni podkomendy arytmetycznej w formacie jeden i dwa:

  ```
  au destination := A bus source + B bus source.
  ```

  Dopuszczalne rejestry:

  - wynik (*destination*): `tcr1, tcr2, a, diob, sr, ert, p, p_high, p_low, link, chan_reg, chan_dec` (można również użyć `nil` – brak zapisu wyniku),
  - pierwszy argument (*A bus source*): `tcr1, tcr2, a, diob, sr, ert, p, p_high, p_low, link, chan_reg`,
  - drugi argument (*B bus source*): `a, diob, sr, p`.
- Efektem ubocznym podkomendy `au read_mer` jest wystawienie zera na magistrali A procesora TPU.
- W formatach 1, 2 i 5 możliwe jest jednokrotne przesunięcie w lewo (`:=<<`), przesunięcie w prawo (`:=>>`) lub obrót w prawo (`:=R>`).
- Sumator jest 16-bitowy, krótsze słowa (8- i 4-bitowe są uzupełniane zerami.
- Zależności czasowe:
  - T1 – dane są wystawiane na magistrale A i B,
  - T2 – dane są zatrzaskiwane i wykonuje się operacja,
  - T3 – dane są zapisywane w rejestrze wynikowym,
  - T4 – możliwy jest dostęp (zapis/odczyt) do pamięci RAM.
- W formatach 1 i 2 można używać wbudowanych stałych: `0, 1, -1, max($8000)`:

```
au dest := 0.
au dest := 1.}
au dest := -1.}
au dest := max.}
au dest := Asrce.}
au dest := Asrce+1.}
au dest := Asrce-1.}
au dest := Asrce+max.}
au dest := Asrce+Bsrce.}
au dest := Asrce+Bsrce+1.}
au dest := Asrce-Bsrce-1.}
au dest :+ Asrce-Bsrce.}
```

- Dzięki specjalnemu połączeniu rejestru `sr` i rejestru przesuwającego jednostki arytmetycznej procesora TPU, w formacie 5 można dokonywać 32-bitowych przesunięć: `au diob :=>> diob,shift.`
- Kody warunkowe (*condition codes*)są zatrzaskiwane wyłączneie podkomendą `ccl`w formatach 1 i 5: `au ert := ert+diob,ccl.`
- Dane bezpośrednie (*immediate data*) mogą być wystawiane na magistralę B w formacie 5 (używa się znaku # do ich oznaczenia). Dopuszczalna składnia:

```
au dest := Asrce.
au dest := Asrce + IMM byte.
au dest := IMM byte.
```

- Przy ładowaniu rejestrów `link, chan_reg` dane są przekazywane na liniach 7:4 (np. `au link := #$50`).
- Zmiana zawartości rejestru `chan_reg` pozwala na sterowanie innym kanałem.
- Rejestr `dec` jest ładowany wartością `$F` przy każdym przełączeniu kontekstu (może być użyty przez komendy `repeat, dec_return`).
- Specjalne rozwiązanie sprzętowe pozwala mnożyć dwie liczby 16-bitowe z 32-bitowym wynikiem. Należy do rejestrów `reg1, reg2` załadować argumenty, a następnie uruchomić pętlę:

```
au dec := #15.
repeat;
au reg1 :=>> reg1 + reg2,shift.
```

9 Sterowanie wykonaniem mikrokodu

- Licznik rozkazów mikrokodu steruje wykonaniem kolejnych rozkazów w TPU.
- Jednorozkazowa kolejka pobieranych rozkazów (*prefetch queue*) przechowuje następną instrukcję do wykonania. Opcja `flush` opróżnia kolejkę, `no_flush` powoduje wykonanie zawartej w niej instrukcji przed wykonaniem skoku.

- Opcja `no_flush` jest efektywniejsza i w miarę możliwości powinna być używana (jest domyślna).
- W czasie wywołania podprogramu adres powrotu jest przechowywany w rejestrze RAR (*return address register*). Podprogramy nie mogą być zagnieżdżane.
- Komendy sterujące przebiegiem programu to: `goto, if ... then goto, call, return, repeat, end.`
- Przykłady:

```
goto LABEL1,flush.
goto LABEL1,no_flush.
goto LABEL1.                (*no_flush domyslne*)
call SUB1,flush.
call SUB1,no_flush.
call SUB1.                  (*no_flush domyslne*)
call SUB1,flush;dec_return.
return,flush.
return,no_flush.
return.                     (*no_flush domyslne*)
repeat;
au p:=<<p.
if N=1 then goto LABEL1,flush.
if C=true then goto LABEL1,no_flush.
if PSL=0 then goto LABEL1.  (*no_flush domyslne*)
end.
```

10 Przykład mikrokodu funkcji TPU

```
(*****************************************************************)
(* Function:    SQW -   RECTANGULAR WAVE                         *)
(*                                                               *)
(* Creation Date: 03//Mar//92                 From: NEW          *)
(* Author:  Amy Dyson                                            *)
(* Description:                                                  *)
(* ------------                                                  *)
(* SQW produces a continuous squarer wave after initialization.  *)
(* The user chooses the high time by writing the                 *)
(* parameter HIGH_TIME in ram.                                   *)
(* HIGH_TIME must be between $0000-$8000.                        *)
(*                                                               *)
(* Updates:   By:   Modification:                                *)
(* --------   ---   ------------                                 *)
(* 11//Apr//93   JL   Convert to new syntax                      *)
(* 05//Jan//99   AD   Update to Tpumasm 5.0                      *)
(*---------------------------------------------------------------*)
(* Standard Exits Used:-   End_Of_Phase: N      End_Of_Link: Y   *)
(*                                                               *)
(* External Files included: NONE                                 *)
```

```
(*                                                                  *)
(* CODE SIZE excluding standard exits = 11 LONG WORDS               *)
(*------------------------------------------------------------------*)
(**********              This Revision:  REV B                ****)
(**********   LAST MODIFIED: 11//Apr//93     BY: Jeff Loeliger ***)
(*                                                                  *)
(********************************************************************)
(*()()()()()()()()()()() DATA STRUCTURE ()()()()()()()()()()()()()(*)
(*                                                                  *)
(* name:               Written By:            Location  Bits:     *)
(* -----               -----------            --------------     *)
(* HIGH_TIME_SQW        CPU                 Parameter0  0..15     *)
(*                     High time of period. HIGH_TIME_SQW cannot  *)
(*                     be greater than $8000.                     *)
(*                                                                *)
(* hsr1   hsr0         Action                                     *)
(* ----   ----         ------                                     *)
(*  1       1          Initialize continuous square wave          *)
(*                                                                *)
(*                                                                *)
(* Links Accepted: NO              Links Generated: NO            *)
(*                                                                *)
(* Interrupts Generated After:     No interrupts generated        *)
(*                                                                *)
(*()()()()()()()()()()()()()()()()()()()()()()()()()()()()()()()()(*)

%macro HIGH_TIME_SQW   'prm0'.

%entry name = INIT_SQW; start_address *; disable_match;
cond hsr1=1, hsr0=1;
ram p <- @HIGH_TIME_SQW.
        chan            TBS := out_m1_c1,
                        PAC := toggle,
                        PIN := high,
                        enable_mtsr.
        au              ert := tcr1 + p;
        chan            write_mer,
                        neg_mrl, neg_tdl, neg_lsl;
                        end.

%entry name = TOGGLE_SQW; start_address *; disable_match;
cond hsr1=0, hsr0=0, m//tsr=1;
ram p <- @HIGH_TIME_SQW.

        au              ert := ert + p;
        chan            write_mer,
                        neg_mrl, neg_tdl, neg_lsl;
                        end.

(****************************************************************)
```

```
(* UNUSED ENTRIES  - execute an end                                  *)
(*********************************************************************)
%entry  name = MESSAGE_UNUSED;
        start_address END_OF_PHASE;
        cond hsr1 = 0, hsr0 = 1.

%entry  name = MESSAGE_UNUSED;
        start_address END_OF_PHASE;
        cond hsr1 = 1, hsr0 = 0.

%entry  name = MESSAGE_UNUSED;
        start_address END_OF_LINK;
        cond hsr1 = 0,hsr0 = 0,m//tsr = 0,lsr =1.
```

11 Szacowanie opóźnień

- Niezawodne systemy powinny być projektowane tak, by mogły poprawnie pracować w najgorszych z możliwych warunków.
- Najgorsze możliwe opóźnienie (*worst-case latency*) dla kanału jest to najdłuższy możliwy interwał pomiędzy uruchomieniem dwóch stanów funkcji w tym kanale.
- W celu znalezienia najgorszego możliwego opóźnienia dla kanału należy wyliczyć pierwsze przybliżenie oparte na deterministycznych wzorach (które jest zazwyczaj znacznie gorsze od rzeczywistego opóźnienia).
- Jeśli otrzymany wynik jest zadowalający, można na tym poprzestać, mając pewność, że system ma wystarczającą wydajność.
- W przeciwnym przypadku należy wykonać drugi krok, wykorzystując specyficzne dane dla systemu w celu znalezienia realnego przybliżenia najgorszego przypadku.

12 TPU2

- Ulepszenia sprzętowe:
  - TPU2 obsługuje do czterech 2K banków pamięci mikrokodu i punktów startowych funkcji czasowych.
  - Każdy kanał ma 8 parametrów w RAM.
  - Ulepszono wybór częstotliwości i filtrację zegarów TCR1 i TCR2.
  - Wprowadzono programowy reset by umożliwić rekonfigurację TPU.
  - Wprowadzono opcję zablokowania wyprowadzeń TPU.
  - Zachowano kompatybilność z TPU1.
- Ulepszenia programowe:
  - Dodano nowy znacznik (`flag2`).

  - Wprowadzono rozkaz skoku w zależności od stanu wyprowadzenia.
  - Wprowadzono warunek równości do komparatora kanału.
  - TPU2 pozwala na zgaszenie TDL i MRL w polu TBS.

13 TPU3

- Wprowadzono wstępny podział źródła zegara TCR1 przez 4..64.
- Wprowadzono wstępny podział źródła zegara TCR2 przez 1 lub 2.
- Wprowadzono możliwość zezwolenia na wielokrotny wpis pól dzielników zegara w rejestrach konfiguracyjnych TPU.