Zentrallabor für Elektronik

Berichte zum

Datenerfassungssystem

für physikalische Experimente

PCI

SYNCHRONISATIONS MODULE

W. Erven, P. Wüstner, A. Ackens, K. Zwoll

Forschungszentrum Jülich
Zentrallabor für Elektronik

Abteilung Experimentsteuerung und Kommunikation

IB-KFA-ZEL 500896
25. Oktober 1996 (Version 1)

Version 2
Letzte Bearbeitung: 19. November 1997

Inhaltsverzeichnis

1 Übersicht

Das Synchronisations-Module ist ein PCI Module, daß für den Einsatz in Standard PC's gedacht ist. Es soll dafür sorgen, daß alle Rechner in einem Data Acquistion System zum gleichen Zeitpunkt die Datenerfassung starten und die Daten einer eindeutigen Eventnummer zuordnen. Die einzelnen Module werden ringförmig miteinander verbunden. Jeder Abschnitt kann wahlweise durch ein Twisted Pair Flachband-Kabel oder durch einen Lichtleiter (OPTO BUS von MOTOROLA) verbunden werden. Auf dem Flachband-Kabel werden die Signale in LVDS Technologie übertragen.

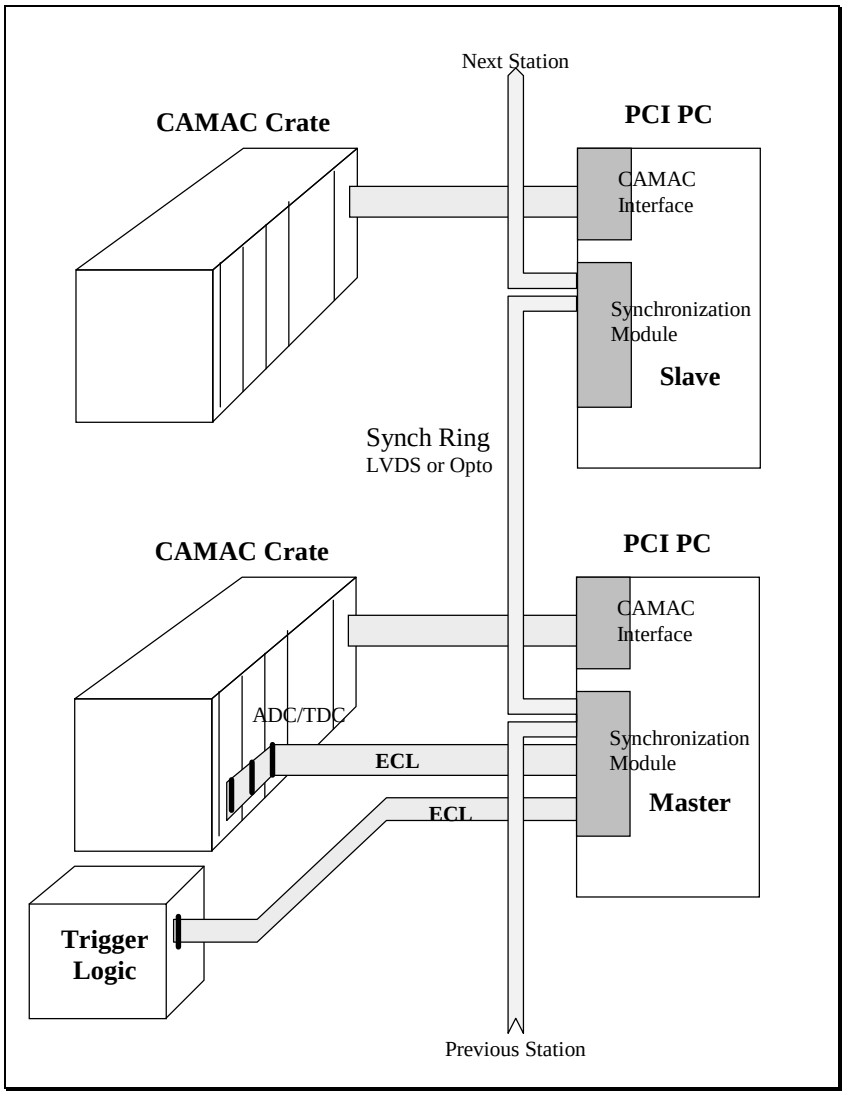


Abbildung 1 Data Acquistion System

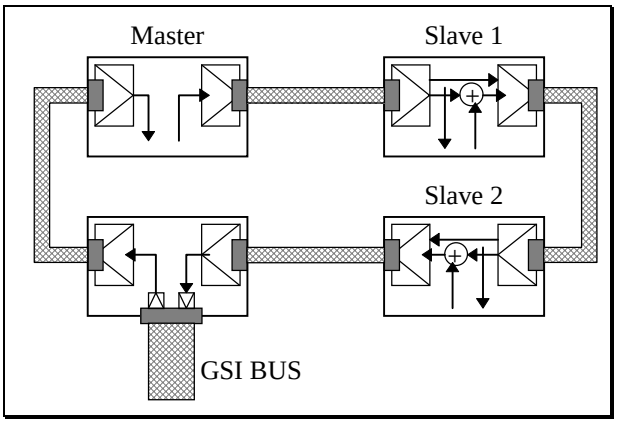
Ein (und nur ein) Module muß als Master konfiguriert werden. Dieses Module empfängt bis zu 4 Trigger-Signale sowie ein VETO und ein FASTCLEAR Signal als differential ECL. Ein Trigger Acceptance Time Window und ein FASTCLEAR Acceptance Time Window können durch Software eingestellt werden. Der vierte Trigger kann so eingestellt werden, daß er gespeichert wird, falls keine Trigger zugelassen sind (Total Dead Time). Beim nächsten Start wird er dann sofort aktiviert. Für die Statistik steht ein Counter für die abgewiesenen Trigger Signale (Rejected Trigger) zur Verfügung und ein FIFO Register, das die Abstände der Tigger Signale liefert. Die Impulsweiten der ECL Ausgangs-Signale (4 Trigger, 1 Master Trigger) sind durch Software einstellbar.

Neben den genannten Signalen, die nur der Master zur Verfügung stellt, gibt es auf dem Master und Slave Module 3 ECL Auxilliary Input Signale und 3 ECL Output Signale, die von der Software frei verwendbar sind.

Damit die GSI "Trigger and Synchronisation" Module weiter benutzt werden können, ist ein Adapter vorgesehen, der im Ring integriert ist und einen Anschluß für den GSI Trigger Bus bereitstellt (siehe **Abbildung 2**).

2 Hardware

Die **Abbildung 2** zeigt die Ringbeschaltung der Synchronisations-Module. Am Master Module ist der Ring unterbrochen. Signale wie die Trigger Signale werden vom Master eingespeist und von jeder weiteren Station durchgereicht. Auf den Signalen SI (Subevent Invalid) und TDT (Total Dead Time) "odert" jede Slave Station ihr eigenes Signal.



Die Ankopplung zum GSI Bus ist eine passive Karte, die durch geeignete Logik die Signale für den GSI BUS erzeugt.

Abbildung 2 Ring-Schaltung

Bei Verwendung eines Leichtleiters muß auf jeder Karte, wo der Eingang ein Lichtleiter ist, ein Jumper gesteckt werden (**J1** in **Abbildung 3**). Die ECL Steckerbelegung ist nochmals in **Abbildung 5** dargestellt.

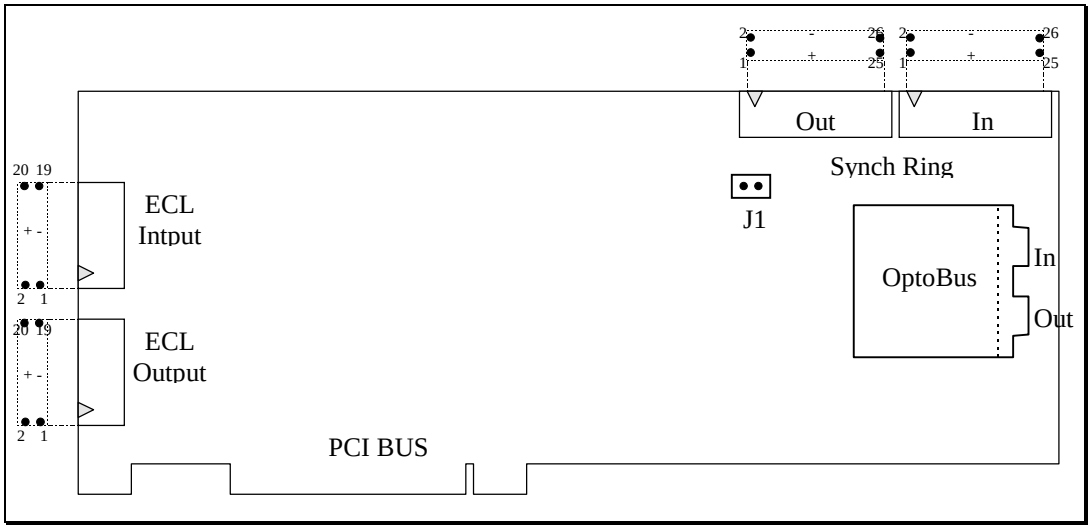


Abbildung 3 PCI Synchronisation Module

Das Blockschaltbild der Hardware ist in der folgenden Abbildung dargestellt. Sie ist in ECL Technik aufgebaut, um möglichst kurze Durchlaufzeiten zu erreichen. Die Verzögerung vom ECL Input Trigger bis zum ECL Output Trigger Impuls liegt bei 3ns. Alle umrandeten Labels führen zum MACH CPLD Baustein und sind teilweise im Control/Status Register für die Software zugänglich. Die in der Nähe der Labels angegebenen Namen sind die Signal Namen, wie sie im MACH Source File stehen.

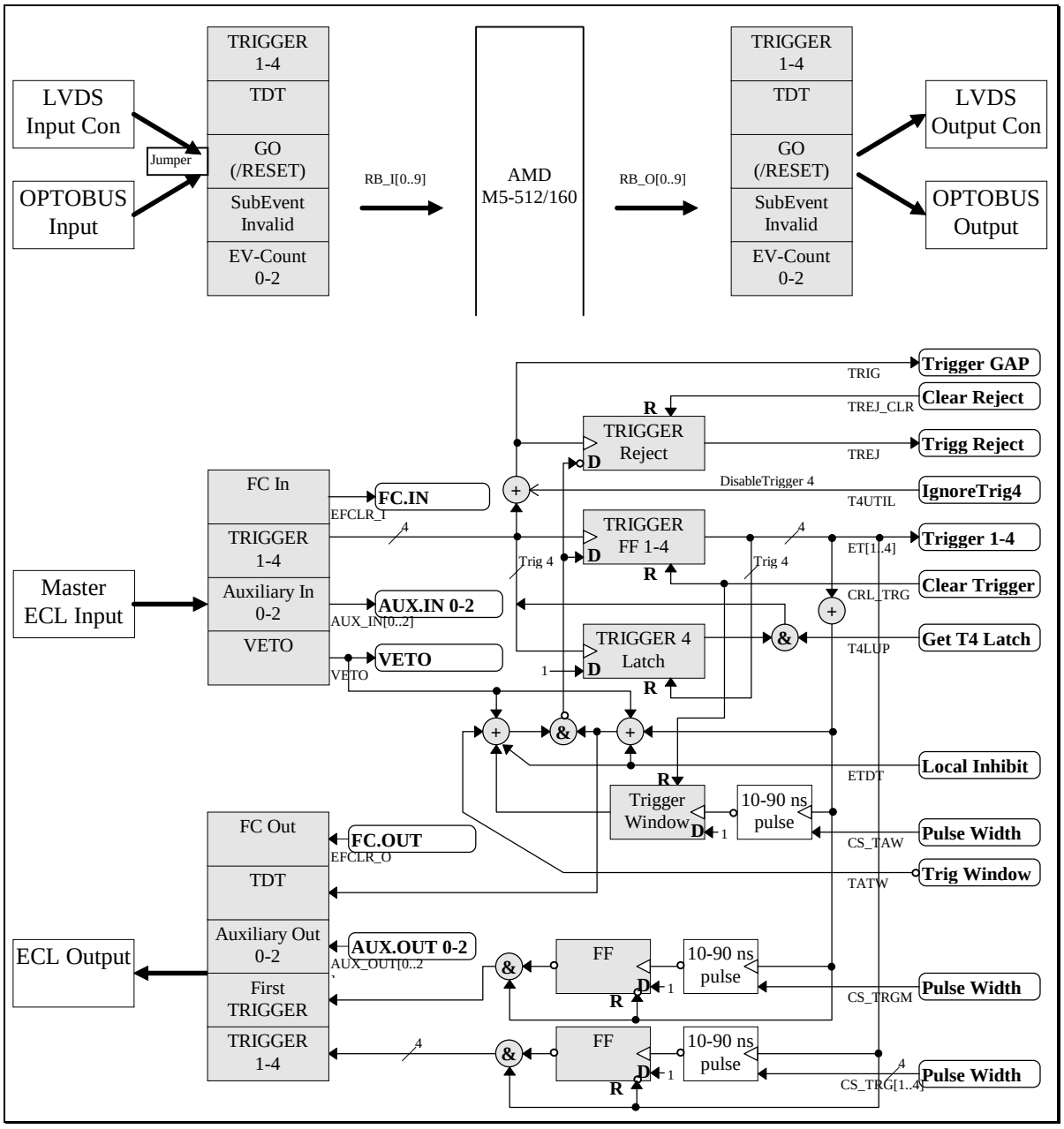
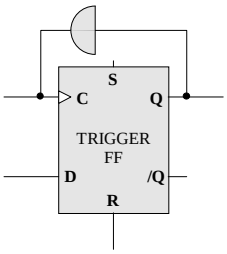


Abbildung 4 Schaltungs-Übersicht

Anmerkung:

Die FF's sind so geschaltet, daß sich der Zustand nicht mehr ändert, wenn das FF gesetzt ist und ein weiterer Trigger Impuls eingeht. Durch die Rückführung des Q Ausgangs über einem Gatter zum Clock Eingang wird der Clock Eingang auf *high* Potential gehalten (*wired or*).



2.1 Signal Belegung

Für die Bezeichnungen der Steckersignale werden folgende vorangestellte Kennbuchstaben benutzt:

I_ Signale vom ECL Input Stecker

O_ Signale am ECL Output Stecker

BI_ Signale vom Ring-Bus Input Stecker

BO_ Signale am Ring-Bus Output Stecker

Der GSI Trigger Bus wurde durch einen Synchronisations-Ring ersetzt. Das hat den Vorteil, daß die neuere LVDS Technologie oder der MOTOROLA OptoBus eingesetzt werden können. Die Ringstruktur ist notwendig, weil es "wired or" Signale gibt, die der Master erkennen muß (***SI*** und ***TDT***). Der GSI Bus kann aber weiter durch ein spezielles Module angekoppelt werden (siehe **Abbildung 2**).

Die maximale Anzahl der Signale auf dem Synchronisations-Ring ist durch den OptoBus auf 10 begrenzt. Die Anzahl der Eventcounter-Bits wurde deshalb von 4 auf 3 verringert. Bei der Ankopplung an den GSI Bus wird das fehlende Bit regeneriert.

Auf dem Synchronisations-Ring werden folgende Signale verwendet.

+	-			**Source**	**Destination**
1 3 5 7	2 4 6 8	B_T1 B_T2 B_T3 B_T4	Trigger Status Signals	Master	alle Slave’s
9	10	B_TDT+	Total Dead Time	alle Slave’s, wired or	Master
11	12	B_GO	Start mit Subevent 0	Master	alle Slave’s
13	14	B_SI++	Subevent Invalid	alle	alle
15 17 19	16 18 20	B_EC0 B_EC1 B_EC2	Event Counter, 3 LS Bit	Master	alle Slave’s
21 23 25	22 24 26	GND GND GND			

Tabelle 1 Signale auf dem Synchronisations-Ring

Folgende Tabelle zeigt die Belegung auf dem ECL Input Stecker. Die Eingänge sind ECL differential und mit 51Ω nach V_{BB} terminiert.

-	+			**Destination**
1 3 5 7	2 4 6 8	I_T1 I_T2 I_T3 I_T4	Trigger	Master
9 11 13	10 12 14	I_AUX0 I_AUX1 I_AUX2	Auxiliary Input	Alle
15	16	I_VETO		Master
17	18	I_FC	Fast Clear	Master
19	20	frei		

Tabelle 2 ECL Eingangssignale

Die Belegung des ECL Output Steckers ist in der nächsten Tabelle dargestellt. Die Ausgänge sind ECL differential und mit 510Ω nach -5V terminiert.

-	+			**Source**
1	2	O_TDT	Total Dead Time	Master
3 5 7	4 6 8	O_AUX0 O_AUX1 O_AUX2	Auxiliary Output	Alle (siehe CR Register)
9 11 13 15	10 12 14 16	O_T1 O_T2 O_T3 O_T4	Trigger Pulse	nur Master
17	18	O_MT	Main Trigger Pulse	nur Master
19	20	O_FC	Fast Clear Pulse	nur Master

Tabelle 3 ECL Ausgangssignale (für DTC's und ADC's)

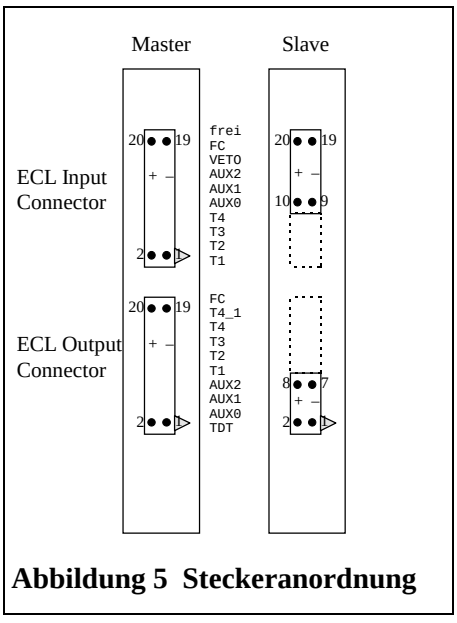


Abbildung 5 Steckeranordnung

In der nebenstehenden Abbildung ist die Steckeranordnung mit Signalbelegung dargestellt, wenn man auf die Rückseite des PCI Interfaces schaut.

2.2 Zeitdiagramme

In den folgenden Zeitdiagrammen sind im wesentlichen Signale dargestellt, die auf den ECL und Ring-Bus Anschlüssen anstehen. Dabei werden folgende vorangestellte Kennbuchstaben benutzt:

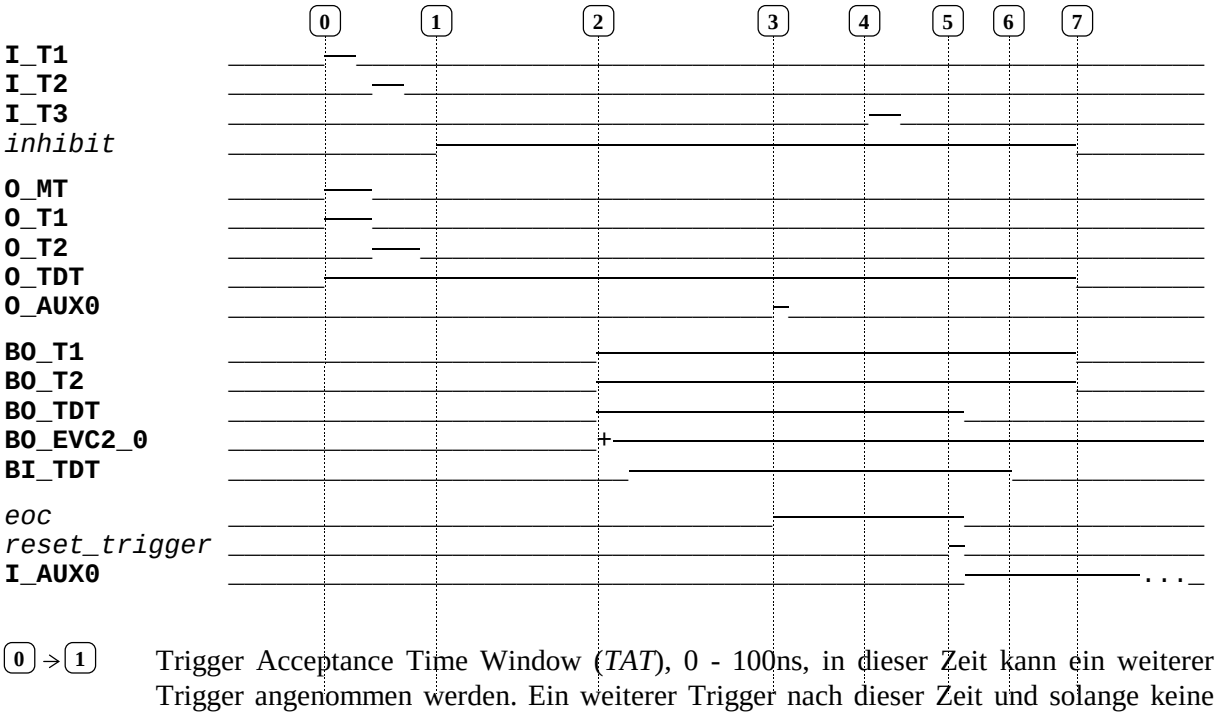
I_ Signale vom ECL Input Stecker

O_ Signale am ECL Output Stecker

BI_ Signale vom Ring-Bus Input Stecker

BO_ Signale am Ring-Bus Output Stecker

Master Timing



0 → 1 Trigger Acceptance Time Window (*TAT*), 0 - 100ns, in dieser Zeit kann ein weiterer Trigger angenommen werden. Ein weiterer Trigger nach dieser Zeit und solange keine neue Freigabe erfolgt ist, zählt den Reject Counter hoch.

0 → 2 Fast Clear Acceptance Time Window (*FCAT*), 0 - 25µs, in dieser Zeit wird ein Fast Clear Impuls akzeptiert (siehe nächstes Diagramm).

2 zu diesem Zeitpunkt legt der Master die Trigger Signale mit 3 Bit des Event Counters auf den Ring Bus. D.h, die Slave Stationen erfahren erst zu diesem Zeitpunkt, wenn *FCAT* abgelaufen ist, vom neuen Trigger und müssen diese Zeit von der Conversion Time abziehen.

0 → 3 Conversion Time, diese Zeit muß größer als die eingstellte *FCAT* sein. Nach diesem Zeitpunkt ist der End Of Conversion Timer abgelaufen und es wird das *eoc* Signal gesetzt. Am ECL Ausgang wird ein Impuls von 100ns auf **O_AUX0** ausgegeben.

4 wenn ein Trigger Signal (**I_T1**-**I_T4**) im gesperrten Zustand (*inhibit* gesetzt) erscheint, wird der Reject Counter erhöht. Ein Sonderfall stellt **I_T4** dar. Dieses Signal kann gespeichert werden, bis *inhibit* zurückgesetzt wird.

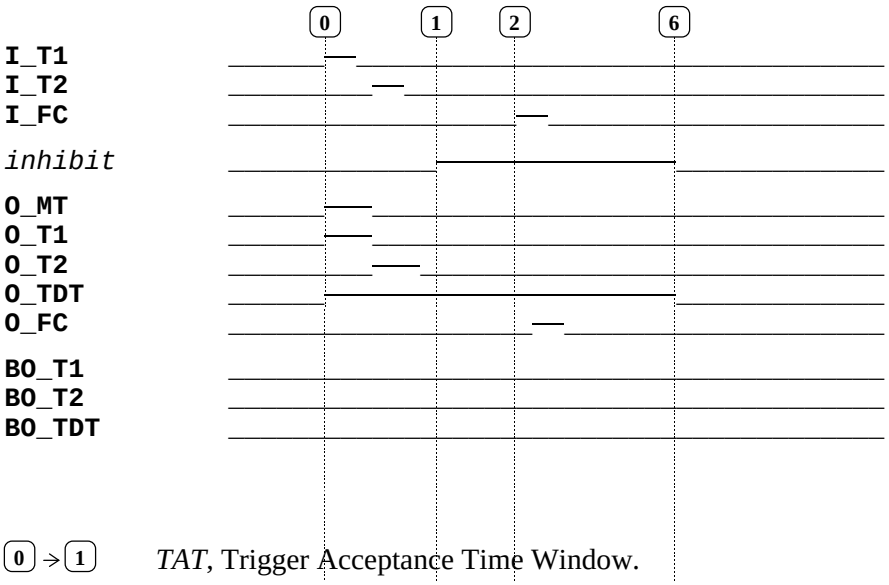
5 zu diesem Zeitpunkt wird von der Software das lokale TDT und das *eoc* Signal zurückgesetzt. Diese Signale können auch durch die positive Flanke des ECL Inputsignals **I_AUX0** zurückgesetzt werden.

6 zu diesem Zeitpunkt geht das Bus Summensignal **BI_TDT** zurück. Alle Stationen haben das ReadOut beendet.

6 → 7 wenn das lokale TDT Signal von allen Stationen zurückgenommen ist (d.h. **BI_TDT** geht zurück), werden die Trigger Signale zurückgesetzt. Wenn jedoch das ***GSI*** Bit im CSR gesetzt ist, werden die Trigger Signale erst nach einer Zeit von 1µs zurückgesetzt. Diese Zeit wird von den GSI Modulen verlangt.

7 die Trigger Signale werden vom Bus zurückgenommen, das ECL Output Signal O_TDT wird zurückgesetzt und die Triggerfreigabe wird wieder gesetzt. Der Punkt 7 fällt mit Punkt 6 zusammen, wenn das GSI Bit nicht gesetzt ist.

Master Timing mit Fast Clear



0 → 1 *TAT*, Trigger Acceptance Time Window.

2 zu diesem Zeitpunkt wird ein Fast Clear Signal erkannt. Der Eingangsimpuls wird auf den ECL Output **O_FC** gelegt. Da noch keine Signale auf den Ring Bus gelegt wurden, merken die Slave Stationen nichts von diesem Ereignis.

2 → 6 nach dem Fast Clear Signal wird nach einer weiteren Sicherheitszeit von 1,2 µs das **O_TDT** Signal zurückgenommen und .die Triggerfreigabe wieder gesetzt.

3 Firmware/PCI

Die gesamte Ablaufsteuerung ist in einem M5-512/160 Baustein von AMD untergebracht. Dieser Baustein enthält im wesentlichen das PCI Pass-Thru Timing, drei Zähler und zahlreiche Register.

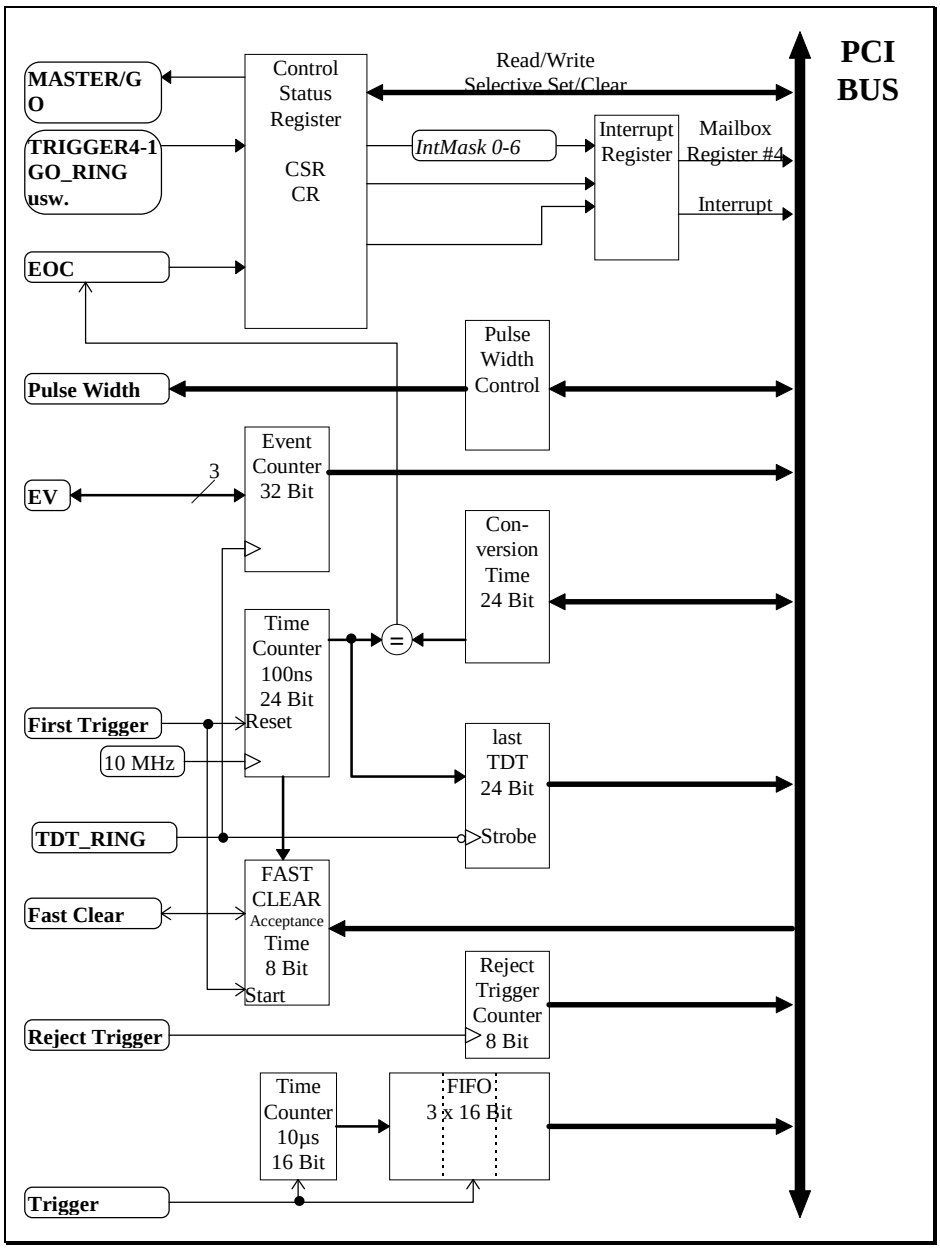


Abbildung 6 MACH Blockschaltung

3.1 Device Register

Im MACH CPLD Baustein sind zehn Register Funktionen implementiert. Diese sind nur im 32 Bit Format über den PCI Bus zugänglich. Die zugeordneten Adressen sind in folgender Tabelle aufgeführt. Die Offset Adressen wurden so gewählt, daß sie sich leicht im MACH Baustein realisieren lassen.

Die Adressen sind nicht voll dekodiert, d.h., daß beim Lesen einer nicht aufgeführten Adresse irgendein Wert gelesen wird.

Offset	Name	Read	Write	Reset
0x00	CSR	Control and Status	Clear Interrupt, Reset Trigger	1)
0x04	CR	Control and Status (wie CSR)	Selective Set/Clear Control Bit's	1)
0x08	CVTR	Conversion Time	Conversion Time, 24 Bit	---
0x0C	PUWR	0x00000000	Pulse Width Control, 8 Bit	0x00
0x10	TMC	Time Counter, 24 Bit	---	---
0x14	TDTR	Total Dead Time, 24 Bit	---	2)
0x18	FCAT	Fast Clear Acceptance Time, 8 Bit	Fast Clear Acceptance Time, 8 Bit	0x00
0x20	EVC	Event Counter, 32 Bit	Master Reset	3)
0x24	REJE	Rejected Trigger, 8 Bit	---	0x00
0x28	TGAP	Last Trigger Gaps, 3 x 16 Bit	---	4)

1) Alle Control Bit's (B0-B15) werden mit *Master Reset* zurückgesetzt. Eine Ausnahme bildet jedoch das Bit 0 (***MASTER***), welches nur durch das PCI Reset (Power Up) zurückgesetzt wird.

2) Dieses Register wird mit *Master Reset* zurückgesetzt, jedoch wird es sofort wieder geladen (mit dem Inhalt von *TMC*), wenn **BI_TDT** vom Ring-Bus mit Master Reset zurückgesetzt wurde.

3) Der Event Counter wird auf -1 gesetzt, wenn **BI_GO** vom Ring-Bus zurückgesetzt wird.

4) Es werden nur die Status-Flags zurückgesetzt.

3.1.1 Control/Status Register, CSR

Das Control/Status Register ist mit der Adresse **0x00** zugänglich (siehe auch **3.1.2**). Die ersten 16 Bit sind Control Bit's und können im *CR* selective gesetzt bzw. zurückgesetzt werden.

BIT	Acc	Name	Bemerkung
0 00000001	RO	***MASTER***	Freigabe der ECL Logik und Steuerung vom Ring BUS.
1 00000002	RO	***GO***	**Master:** Freigabe der Trigger Eingänge und des Event Counters. **Slave:** Freigabe des Conversion Timers und TDT Generierung.
2 00000004	RO	***T4LATCH***	**Master:** Trigger 4 wird zwischengespeichert.
3 00000008	RO	***TAW_ENA***	**Master:** enable Trigger Acceptance Window.
6-4 00000070	RO	***AUX_OUT0-2***	Auxiliary Output. Am ***O_AUX0*** wird auch ein Impuls von 100ns Länge ausgegeben, wenn des EOC Bit gesetzt wird.
7 00000080	RO	***AUX0_RST_TRIG***	**Master:** ***RESET_TRIGGER*** durch ***AUX_IN0*** (Vorderflanke).
10-8 11 12 13 14 00003F00	RO	***Int_Enable2-0*** ***Int_Enable3*** ***Int_Enable4*** ***Int_Enable5*** ***Int_Enable6***	Interrupt enable für: ***AUX_IN2-0*** Input (Vorderflanken) ***EOC*** ***SI_RING*** ***TRIGGER*** ***GAP_DATA***
15 00008000	RO	***GSI***	**Master**: GSI Mode, 1µs delay nach Rücksetzen von ***TDT_RING***.
19-16 000F0000	RO	***TRIGGER4-1***	Trigger Signale
20 00100000	RO	***GO_RING***	***GO*** Signal vom Ring Bus, *low* bewirkt ein Reset für alle EVC.
21 00200000	RO	***VETO***	ECL Eingang, **Master**: verhindert die Triggerfreigabe.
22 00400000	RO	***SI***	Subevent Invalid, eigener Vergleich mit dem Ring Bus.
23 00800000	RO	***INH***	lokales Inhibit (setzt ***O_TDT*** Output)
26-24 07000000	RO CLR	***AUX_IN2-0*** ***AUX_FF2-0***	RD: ECL Auxiliary Input (siehe Interrupt). WR: setzt das entsprechende Auxiliary Flip Flop zurück
27 08000000	RO	***EOC***	End Of Conversion
28 10000000	RO	***SI_RING***	Subevent Invalid Zustand vom Ring Bus.
29 20000000	RO CLR	0	RD: nicht belegt WR setzt das ***TRIGGER*** Interrupt Signal zurück
30 40000000	RO	***GAP_DATA***	im GAP FIFO stehen Daten.
31 80000000	RO CLR	***TDT_RING*** ***RESET_TRIGGER***	RD: TDT Input vom Ring Bus WR: setzt ***EOC*** zurück und gibt einen ***RESET_TRIGGER*** Impuls and die Hardware aus.

Alle Bit's sind nur lesbar. Die ersten 16 Bit's sind Control Bit's und und können mit dem Set/Clear Register (**CR**) selective gesetzt bzw. zurückgesetzt werden. Die Control Bits (außer ***MASTER***) werden durch ***SYSRESET*** (PCI RST, AMCC Command) und ***MASTER_RESET*** (siehe **3.1.8**) zurückgesetzt. Das ***MASTER*** Bit wird nur durch ***SYSRESET*** zurückgesetzt. Die in der Spalte **Acc** (Access) mit **CLR** gekennzeichneten Bit sind auch schreibbar und bewirken einen Clear Impuls.

Weitere Erläuterungen

MASTER wird durch ***MASTER_RESET*** nicht beeinflußt! Wenn das ***MASTER*** Bit in allen Stationen zurückgesetzt ist, ist der Ring Bus undefiniert (in sich geschlossen). In enem Slave Synchronisations-Modul (ECL Teil ist nicht bestückt) kann dieses Bit nicht gesetzt werden.

GO ***MASTER:*** gibt damit den Ring-Bus frei. Wenn dieses Bit nicht gesetzt ist, wird auf allen Stationen der Event Counter auf -1 vorgesetzt. Die erste Triggerfreigabe erfolgt mit einem ***RESET_TRIGGER*** Kommando.

SLAVE: wenn dieses Bit nicht gesetzt ist, bleibt der *Slave* passive. Der Eventcounter wird zwar inkrementiert, aber es wird kein EOC und damit kein TDT für den Ring-Bus generiert. Nur wenn dieses Bit gesetzt ist, ist der *Slave* in der Trigger-Synchronisation einbezogen.

AUX_OUT0 das ECL Signal ***O_OUT0*** ist das logische "oder" von diesem Bit und einem Impuls von 100ns Länge, der beim Ablauf der End Of Conversion Time ausgegeben wird. D.h. dieser Impuls kann nicht ausgegeben werden, wenn das ***AUX_OUT0*** Bit gesetzt ist.

AUX0_RST_TRIG wenn dieses Bit gesetzt ist, wird durch die positive Flanke am Eingang von Auxiliary 0 ein ***RESET_TRIGGER*** Impuls erzeugt, der die gleiche Auswirkung hat, wie das Setzen des Bit's ***RESET_TRIGGER*** im Status Register (**CSR**).

In den beiden folgenden Abbildungen ist der Anschluß des Ring Busses einmal für den Master und einmal für den Slave dargestellt. Hier sind auch die Signale des Control/Status Registers (CSR) mit gestrichelten Umrandungen dargestellt. Die vollumrandeten Kästchen sind interne Signale. Das Signal ***FCAT*** ist der Fast Clear Acceptance Zustand. Während dieser Zeit werden nach keine Signale vom Master auf den Ring Bus gegeben.

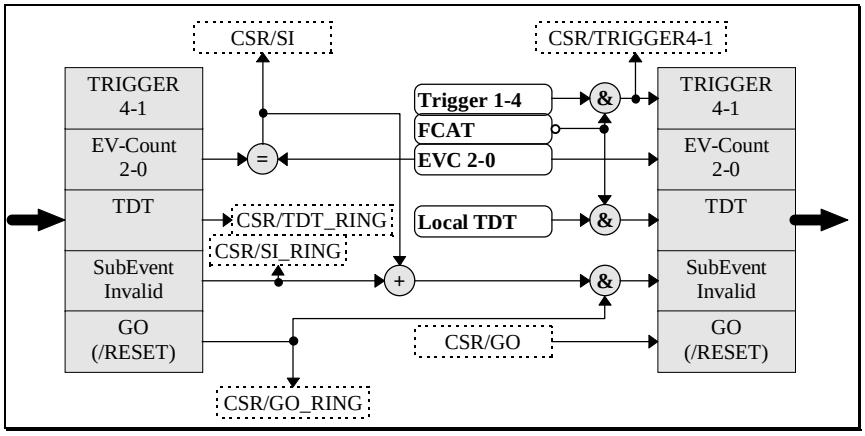


Abbildung 7 Ring Bus am Master

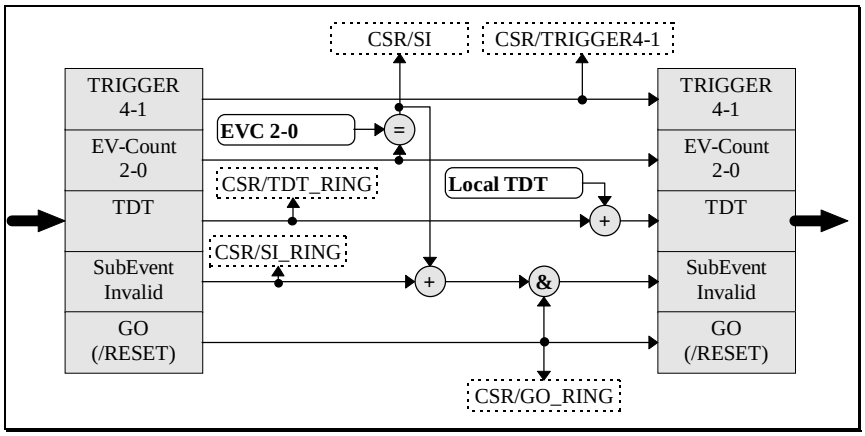


Abbildung 8 Ring Bus am Slave

3.1.2 Control Register, CR

Dieses Register ist mit der Adresse **0x04** zugänglich. Beim Lesen erhält man den Inhalt des Control/Status Registers wie in **3.1.1** beschrieben. Beim Schreiben bleiben alle Bits, die null sind, ohne Wirkung. Gesetzte Bits ermöglichen das selektive Setzen bzw. Zurücksetzen der Control Bit's im Control/Status Register (Bit 0-15). Das low Word, Bit 0-15, wird zum Setzen und der high Word, Bit 15-31, wird zum Zurücksetzen der entsprechenden Bit's benutzt.

BIT	access	Signal	Bemerkung
0 00000001	set	***MASTER***	Freigabe der ECL Logik und Steuerung vom Ring BUS.
1 00000002	set	***GO***	**Master:** Freigabe der Trigger Eingänge und des Event Counters.
2 00000004	set	***T4LATCH***	**Master:** Trigger 4 wird zwischengespeichert.
3 00000008		***TAW_ENA***	**Master:** enable Trigger Acceptance Window.
6-4 00000070	set	***AUX.OUT0-2***	Auxiliary Output.
7 00000080	set	***AUX0_RESTRIG***	`Master:` ***RESET_TRIGGER*** durch ***AUX_IN0*** (Vorderflanke).
10-8 11 12 13 14 00003F00	set	***Int_Enable2-0*** ***...*** ***Int_Enable6***	Interrupt enable für: ***AUX_IN2-0*** Input (Vorderflanken) ***EOC*** ***SI_RING*** ***TRIGGER*** ***GAP_DATA***
15 00008000	set	***GSI***	**Master**: GSI Mode, 1µs delay nach Rücksetzen von ***TDT_RING***.
16 00010000	clear	***MASTER***	**wie oben**
17 00020000	clear	***GO***	
18 00040000	clear	***T4LATCH***	
19 00080000	clear	***TAW_ENA***	
22-20 00700000	clear	***AUX.OUT0-2***	
23 0080000	clear	***AUX0_RESTRIG***	
29-24 3F000000	clear	***Int_Enable5-0***	
30 40000000		not used	
31 80000000	clear	***GSI***	

Wenn für ein Signal das *set* und das *clear Bit* gesetzt sind, nimmt das Signal den invertierten Wert an.

3.1.3 Conversion Time, CVTR

Dieses Register ist mit der Adresse **0x08** zugänglich und ist schreib- und lesbar. Das Conversion Time Register ist ein 24 Bit Register und wird mit dem Time Counter verglichen. Wenn ein Trigger erkannt wurde und der eingestellte Wert erreicht ist, wird das ***EOC*** Flag (End Of Conversion) gesetzt. Das ***EOC*** Flag wird neben *Master Reset* durch ***RESET_TRIGGER*** zurückgesetzt. Auf die Hardware hat dieses Signal keinen Einfluß. Die Zeit ist gleich Registerwert minus eins mal 100 ns. Alle Slave Module müssen die Fast Clear Acceptance Time abziehen, weil die Trigger Signale erst nach dieser Zeit auf den Ring Bus gelegt werden. Das Register wird nicht durch *Master Reset* zurückgesetzt.

3.1.4 Pulswidth Control Register, PUWR

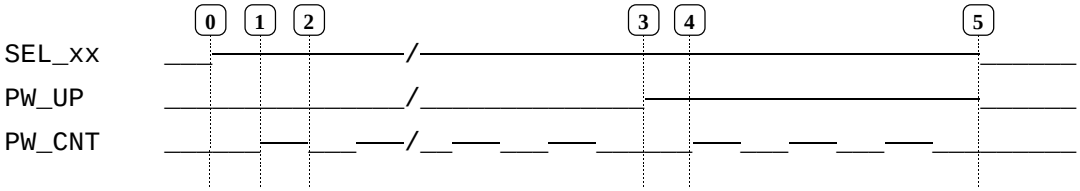
Dieses Register ist mit der Adresse **0x0C** zugänglich. Es ermöglicht die Einstellung der Impulsweiten der ECL Ausgänge und der Trigger Acceptance Time. Das Register wird durch ***MASTER_RESET*** auf null gesetzt.

BIT	access	Signal	Bemerkung
0 00000001	WO	***SEL_T1W***	Select Trigger 1 Width
1 00000002	WO	***SEL_T2W***	
2 00000004	WO	***SEL_T3W***	
3 00000008	WO	***SEL_T4W***	
4 00000010	WO	***SEL_TMW***	Select Main Trigger Width
5 0000020	WO	***SEL_TAT***	Select Trigger Acceptance Time
6 00000040	WO	***PW_UP***	Up/Down Count Direction
7 00000080	WO	***PW_CNTC***	Count

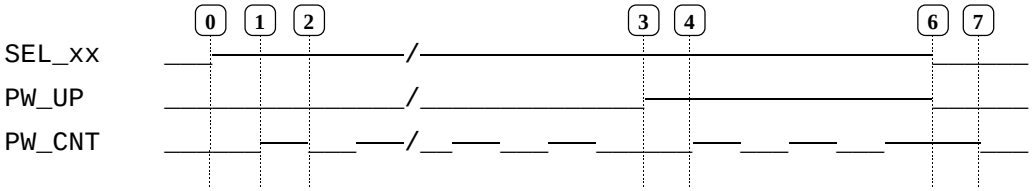
Die Mono FlipFlops sind mit digitalen Potentiometern bestückt, um die Impulsweiten einzustellen. Die Potentiometer haben 32 Positionen, die seriell eingestellt werden können. Wenn die aktuelle Position nicht bekannt ist, muß zuerst die Null Position durch 32 Down Schritte eingestellt werden. Das gewünschte Potentiometer wird durch das entsprechende Select Bit angewählt, wobei auch mehrere Potentiometer gleichzeitig angewählt werden können. Die Länge und der Abstand der ***PW_CNT*** Impulse muß größer als 1 µs sein und nach Änderung der Richtung ***PW_UP*** ist eine Wartezeit von 3 µs nötig, bevor der erste ***PW_CNT*** Impuls ausgegeben werden kann. Diese Zeiten können durch Auslesen des Time Counters gewährleistet werden. Der Time Counter wird durch jeden Schreibzugriff auf 0 gesetzt. Da der Timer Counter mit 0,1 µs zählt, muß der Wert größer als 30 sein, wenn eine Wartezeit von 3 µs notwendig ist.

Das folgende Zeitdiagramm zeigt den Ablauf, um ein digitales Poti neu einzustellen und zu speichern. Es können auch mehrere Potis gleichzeitig verändert werden, wenn die entsprechenden Select Bit's gesetzt werden.

1) Einstellen mit Speicherung



1) Einstellen ohne Speicherung



(0) hier beginnt das Zurückfahren der Potis.

(1) nachdem der TMC größer als 30 ist, wird ***PW_CNT*** für den ersten Count Impuls gesetzt.

(2) nachdem der TMC größer als 30 ist, wird das ***PW_CNT*** wieder auf 0 gesetzt. Die beiden letzten Schritte werden 30 mal wiederholt, um die Postion Null zuerreichen.

(3) nachdem der letzte ***PW_CNT*** auf 0 gesetzt ist, wird wieder nach einer Wartezeit von TMC gleich 30 die Richtung geändert.

(4) hier beginnt nach der bekannten Wartezeit das Hochfahren der Potis auf die gewünschte Position.

(5) hier werden nach der letzten Wartezeit alle Signale zurückgesetzt. Die eingestellte Position wird damit in einem NV Memory gespeichert. Nach einem Power Up wird dieser Wert wieder automatisch eingestellt. Bevor dasselbe Poti wieder neu eingestellt werden kann, ist eine Wartezeit von TMC größer oder gleich 200000, das entspricht 20ms, nötig.

Wenn der neu eingestellte Wert nicht im NV Memory gespeichert werden soll, muß nach dem Setzen das letzten ***PW_CNT*** Schrittes anders vorgegangen werden:

(5) dazu wird nach dem letzten Count das ***PW_CNT*** Bit nicht zurückgesetzt, dafür aber das (oder die) Select Bit.

(6) am Ende werden alle Bit's zurückgesetzt. Danach ist eine Wartezeit von 20ms nötig, wenn das gleiche Poti wieder neu eingestellt werden soll.

Position	Zeit/ns
0	15
10	17
15	20
20	25
22	29
24	32
25	36
26	43
27	54-56
28	115-123
29	128-137
31	128-137

Durch die Verwendung einer einfachen Schaltung ist die Einstellung der Zeiten in keiner Weise linear. Nach oben hin treten zudem Streuungen zwischen den verschiedenen Potis auf. Die Zuordnung zwischen Poti-Position und Zeit ist in nebenstehender Tabelle dargestellt.

3.1.5 Time Counter, TMC

Dieses Register ist mit der Adresse **0x10** zugänglich. Eine Schreibfunktion hat keine Auswirkung. Der Time Counter ist ein 24 Bit Zähler, der mit 10 MHz (0,1 µs) zählt. Er läuft frei und wird unter anderem durch das erste Trigger Signal zurückgesetzt. Der Überlauf erfolgt nach 1,6 sec. Der Zählerstand kann zu jeder Zeit über den PCI Bus gelesen werden. Durch eine besondere Maßnahme ist gewährleistet, daß kein undefinierter Wert während des Zählens gelesen werden kann. Der Time Counter ist die Referenz für die Fast Clear Acceptance Time (FCAT), für die Conversion Time (CVT) und für die Länge der Total Dead Time (TDT). Am Ende der TDT wird er zurückgesetzt, um gegebenenfalls die Wartezeit für die GSI Kompatibilität einzuhalten. Er kann auch für die Einhaltung der Wartezeiten beim Einstellen der digitalen Potis verwendet werden. Dazu wird er bei jedem Schreibzugriff auf das *PUWR* Register zurückgesetzt.

3.1.6 Total Dead Time Register, TDTR

Dieses Register ist mit der Adresse **0x14** zugänglich. Eine Schreibfunktion hat keine Auswirkung. Der TDT Wert ist die Zeit vom ersten Trigger bis das Signal ***TDT.BUS*** (am Trigger Bus) zurückgenommen wird. D.h., der Wert wird mit der Rückflanke von ***TDT.BUS*** vom Time Counter übernommen. Durch eine besondere Maßnahme ist gewährleistet, daß kein undefinierter Wert während der Übernahme gelesen wird. Ein geeigneter Zeitpunkt für das Auslesen des Registers ist z.B. der ***EOC*** Interrupt des folgenden Events. Das Register wird durch ***MASTER_RESET*** auf null gesetzt. Es ist jedoch möglich, das beim *MASTER* nach einem ***MASTER_RESET*** ein neuer Wert in diesem Register steht, wenn gleichzeitig das lokale TDT Signal zurückgesetzt wird.

3.1.7 Fast Clear Acceptance Time Register, FCAT

Dieses Register ist mit der Adresse **0x18** zugänglich. Es enthält die Fast Clear Acceptance Time. Das ist der Zeitraum nach dem ersten Trigger, wo ein Fast Clear durch das Eingangs-Signal ***I_FC*** möglich ist. Die Registerbreite ist 8 Bit und die Zeiteinheit ist 100ns. Das entspricht einer maximaler Zeit von 25,5µs. Das Register wird durch ***MASTER_RESET*** auf null gesetzt. Wenn ein Fast Clear Impuls ausgegeben ist, werden erst nach einer Sicherheitszeit von 1,2µs die Trigger Eingänge neu freigegeben.

3.1.8 Event Counter, EVC

Dieses Register ist mit der Adresse **0x20** zugänglich. Bei einem Schreibzugriff mit beliebigem Datenwert wird ein ***MASTER_RESET*** ausgeführt. Dabei werden im Control/Status Register (CSR) alle Control Signale außer ***MASTER*** zurückgesetzt. Weiter wird das Auxiliary Interrupt Register auf Null gesetzt. Durch den *MASTER* werden durch das Zurücksetzen des ***GO*** Bit's über den Ring-Bus alle Event Counter auf -1 (0xFFFFFFFF) gesetzt.

Der Event Counter ist ein 32 Bit Zähler. Er wird durch den *low* Zustand des Ring-Bus Signales ***GO_RING*** auf -1 vorgesetzt, damit der erste Trigger mit dem Wert Null beginnt. Er wird beim *MASTER* nach Ablauf der FCAT inkrementiert. Das *MASTER* legt die unteren 3 Bit auf den Ring-Bus. Bei den SLAVE Modulen wird der EVC durch die Vorderflanke des ersten Trigger's (lokales TDT) inkrementiert.

Die unteren 3 Bit's des Event Counters werden mit den entsprechenden Bit's, die auf dem Ring-Bus liegen, verglichen. Der Vergleich findet mit der Vorderflanke des ***EOC*** (siehe **3.1.3**) Signals statt. Wenn der Vergleich negativ ist, wird das Flag ***SI*** (Subevent Invalid) gesetzt und auf den Ring-Bus gelegt. Das ***SI*** Flag auf dem Ring-Bus kann nur durch den Master zurückgesetzt werden, indem er das ***GO*** Bit zurücksetzt.

Auch der *MASTER* vergleicht den EVC vom Eingang des Ring-Busses. Wenn er das ***SI*** Flag im CSR setzt, kann nur der Ring-Bus defekt sein.

3.1.9 Reject Trigger Count Register, REJC)

Dieses Register ist mit der Adresse **0x24** zugänglich. Es enthält die Anzahl Trigger, die nicht registriert wurden, weil das lokale Inhibit gesetzt war. Aus Spargründen hat dieses Register nur eine Breite von 8 Bit. Beim Wert von 0xFF zählt das Register nicht mehr weiter. Durch das Auslesen wird dieses Register zurückgesetzt, ohne daß ein Inpuls verloren gehen kann. Die Software kann somit die Werte bei jedem Event einfach akkumulieren. Das Register wird nur durch Lesen zurückgesetzt (nicht mit ***MASTER_RESET***).

Die Erkennung der Rejected Trigger hat eine kleine Unsicherheit, wenn das TDT Signal genau dann zurückgesetzt wird, wenn ein Trigger Impuls empfangen wird. Dieser Zeitraum liegt bei 1ns. In diesem Zeitraum kann es passieren, daß ein Trigger als Trigger und Rejected Trigger erkannt wird. Wenn Trigger 4 Utilization gesetzt ist (***T4LATCH***), wird Trigger 4 nicht berücksichtigt.

3.1.10 Trigger Gap FIFO, TGAP

Dieses FIFO besteht aus 3 Registern und ist mit der Adresse **0x28** zugänglich. Hier werden die Zeitabstände zwischen den einzelnen Triggern erfaßt. Die Register haben eine Breite von 16 Bit mit einer Zeiteinheit von 100ns. Wenn der Zeitabstand größer als 6,5535ms ist (0xFFFF), bleibt dieser Wert stehen. Beim Auslesen wird in Bit 31/30/29 der FIFO Zustand angezeigt. Eine Schreibfunktion hat keine Auswirkung. Der FIFO Flags werden durch ***MASTER_RESET*** auf null gesetzt. Wenn Trigger 4 Utilization gesetzt ist (***T4LATCH***), wird Trigger 4 nicht berücksichtigt.

BIT	access	Signal	Bemerkung
15-0 0000FFFF	RO	***TRG_GAP***	Trigger-Abstand in 10µs Einheiten
29-16 03FF0000	RO	null	
29 20000000	RO	***OVERRUN***	wenn gesetzt, sind Werte verlorengegangen.
30 40000000	RO	***MORE***	wenn gesetzt, ist mindestens noch ein GAP Wert vorhanden.
31 80000000	WO	***NO_GAP***	es ist kein GAP Wert vorhanden, Inhalt ist ungültig.

3.2 Interrupt Utility

Folgende Signale und Flags können einen PCI Interrupt auslösen.

- Vorderflanke der drei Auxiliary Signale vom ECL Input.
- ***EOC*** (End Of Conversion) Flag.
- ***SI_RING*** (Subevent Invalid, Trigger Ring) Flag.
- ***TRIGGER*** Trigger Valid, beim Master nach Ablauf von *FCAT*
- ***GAP_DATA*** Daten im Trigger GAP FIFO

Die Auxiliary Input Signale (siehe Status Register) setzen mit der Vorderflanke ein Flip Flop (FF). Diese können selektiv zurückgesetzt werden (siehe **3.1.2**). Die Signale, die einen Interrupt auslösen können, sind unabhängig von der Interrupt Mask im AMCC "Incoming Mailbox #4" lesbar.

BIT	Name	Bemerkung
23-0 00FFFFFF	not used	sind beim Lesen null.
26-24 07000000	***AUXIN_FF2-0***	Auxiliary Input FF, selektiv rücksetzbar (siehe **3.1.1**)
27 08000000	***EOC***	identisch mit ***EOC*** im Status Register
28 10000000	***SI_RING***	identisch mit ***SI_RING*** im Status Register
29 20000000	***TRIG_VALID***	ist slektiv rücksetzbar (siehe **3.1.1**)
31-30 C0000000	not used	sind beim Lesen null.

Tabelle 4 Incoming Mailbox #4

Der PCI Interrupt für "Incoming Mailbox #4, Byte 3" wird nur für diejenigen Signale erzeugt, die durch das Interrupt Mask Register (im Control Register) zugelassen sind. Ein Interrupt wird nicht erzeugt, wenn ein Interrupt enable Signal gesetzt wird und das Interrupt Source Signal bereits gesetzt ist (Edge Triggered). Nachdem man einen Interrupt enabled hat, sollte man deshalb dafür sorgen, daß die Interrupt Source Signale zurückgesetzt werden.

Wenn ein Interrupt Handler einen Interrupt erkennt, sollte er nach Möglichkeit die entsprechende Interrupt Quelle zurücksetzen, damit beim nächsten Interrupt diese Quelle nicht nocheinmal bearbeitet wird. Folgende Interrupt Quellen können nicht unabhängig zurückgesetzt werden:

EOC das End Of Conversion wird durch das ***RESET_TRIGGER*** Signal zurück gesetzt (siehe **3.1.1**). Damit wird auch der nächste Trigger freigegeben.

SI_RING Subevent Invalid, dieses Signal kann nur durch den MASTER zurckgesetzt werden, indem er das ***GO*** Signal zurücksetzt.

GAP_DATA dieses Bit erscheint nur im Control/Status Register. Es wird dadurch zurückgesetzt, indem das *TGAP* FIFO ausgelesen wird (siehe **3.1.10** Seite **18**).

3.3 PCI Bus

Das Trigger Module hat als PCI Device folgende Kennungen:

Vendor Identification 0x10E8

Device Identification 0x812F

Revision Identification 0x01

Base Class Code 0x09 (Input Devices)

Sub Class Code 0x80 (no standard Input Device)

Mit der PCI Bios Funktion *FIND_PCI_DEVICE* können die *Bus* und *Device* Nummern des Modules ermittelt werden (siehe PCI BIOS SPECIFICATION).

3.3.1 PCI Bus Mapping

Das Trigger Module hat am PCI Bus 2 Adress-Regionen, die beim Power Up vom PCI BIOS initialisiert werden. Sie können grundsätzlich im Memory- oder IO-Space liegen, wobei der IO Space kleiner oder gleich 256 Byte sein muß. Hier wurde für alle Regionen der Memory-Space gewählt, weil der IO-Space nicht auf allen Rechnern zur Verfügung steht. Die Basis Adressen dieser Bereiche können aus dem sogenannten „PCI Configuration Header“ entnommen werden (durch Aufruf des PCI BIOS).

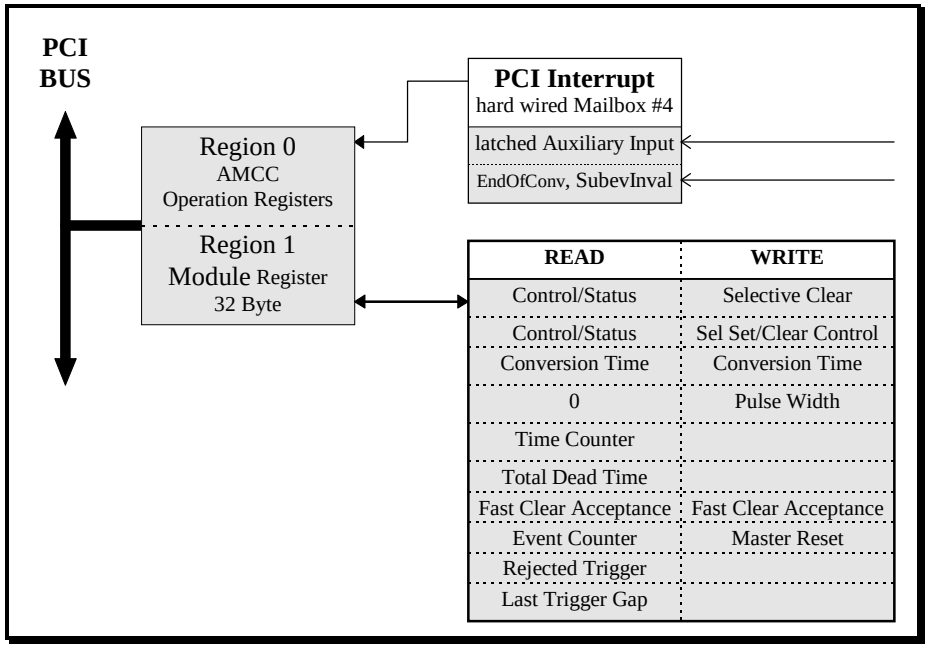


Abbildung 9

3.3.2 Region 0

Dieser Bereich hat eine Größe von 64 Byte. Über diesen Bereich sind alle Register des AMCC's zugänglich. Diese Register steuern im wesentlichen den Interrupt Mechanismus. Dazu wird nur die "Incoming Mailbox #4" benutzt, um einen PCI Interrupt zu erzeugen.

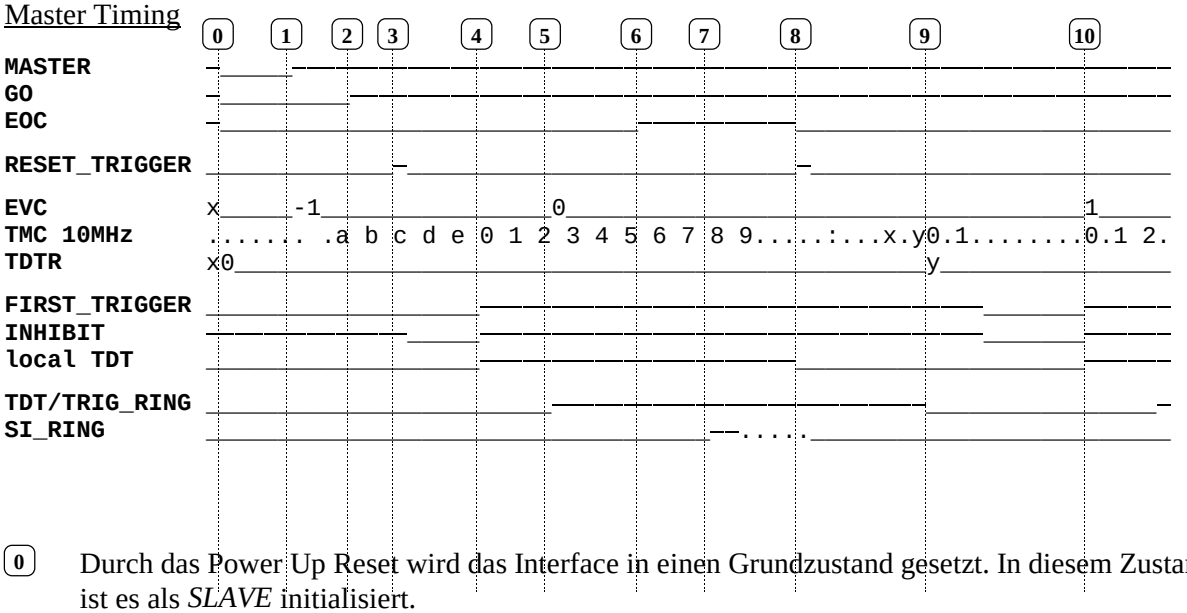
3.3.3 Region 1

In dieser Region befinden sich alle Register des Trigger Module's.

4 Programmierungshinweise

4.1 Timing

Im folgenden ist ein Beispiel für den zeitlichen Ablauf dargestellt.



(0) Durch das Power Up Reset wird das Interface in einen Grundzustand gesetzt. In diesem Zustand ist es als *SLAVE* initialisiert.

(1) Als erstes wird das ***MASTER*** Flag gesetzt. Danach können die Pulsweiten eingestellt werden und das ***CVTR***/***FCAR*** Register geladen werden.

(2) Das ***GO*** Signal wird zusammen mit anderen Control Bit's gesetzt. Es hat bis hierhin alle Event Counter (EVC) auf -1 gehalten. Das GO Signal am Ende des RING BUSses (***GO_RING*** im Statusregister) muß den selben Zustand haben. Der Trigger Input ist noch nicht freigegeben.

(3) Durch das ***RESET_TRIGGER*** Kommando wird der Trigger Eingang freigegeben.

(4) Der erste Trigger wird erkannt. Der 10 MHz Timer wird zurückgesetzt. Der Event Counter wird inkrementiert. Die unteren 3 Bit liegen auf dem Trigger Bus. Wenn eine Fast Clear Acceptance Time (FACT) eingestellt ist, werden noch keine Signale auf den RING BUS gelegt.

(5) Die FACT ist abgelaufen. Der Event Counter wird inkrementiert. Die unteren 3 Bit liegen auf dem Trigger Bus. Die Trigger Signale werden zusammen mit dem lokalen TDT auf den RING BUS gelegt.

(6) Der 10 MHz Timer erreicht die Conversion Time (***CVTR***). Das ***EOC*** Flag wird gesetzt, welches einen Interrupt auslösen kann. Das ReadOut kann beginnen.

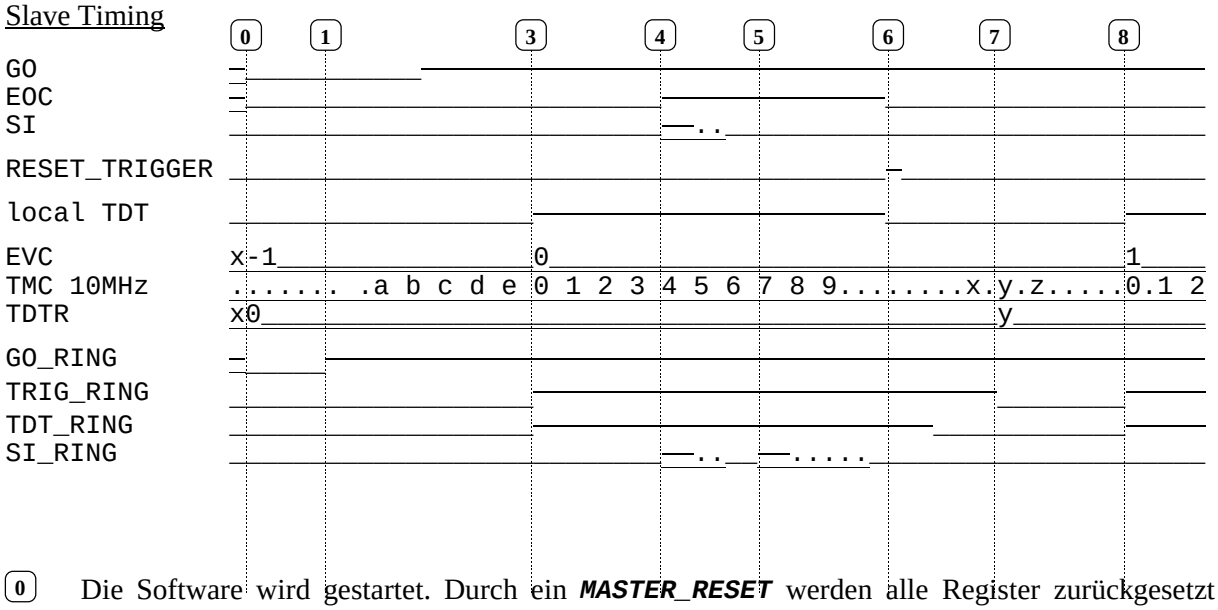
(7) Falls ein SLAVE eine Abweichung des eigenen Event Counters erkennt (Vergleich mit den unteren 3 Bit's auf dem RING BUS), setzt er das ***SI*** Signal auf den RING BUS (***SI_RING)***. Dieses Signal kann einen Interrupt auslösen und eine Fehlerbehandlung starten. Es kann nur durch das Zurücksetzen von ***GO*** gelöscht werden.

(8) Wenn alle Daten ausgelesen sind, werden durch das ***RESET_TRIGGER*** Kommando die internen Flags EOC und TDT zurückgesetzt.

(9) Wenn der letzte Slave sein TDT Signal zurückgesetzt hat, geht auch das ***TDT_RING*** Signal zurück. Mit der Rückflanke von ***TDT_RING*** wird der 10 MHz Timer in das ***TDTR*** übernommen. Wenn das ***GSI*** Bit gesetzt ist, wird nach 1 µs gewartet, ehe die Trigger Signale vom RING BUS genommen werden und der Trigger Eingang wieder freigegeben wird.

(10) Der nächste Trigger wird erkannt.

Slave Timing



0 Die Software wird gestartet. Durch ein ***MASTER_RESET*** werden alle Register zurückgesetzt (***CSR*** und ***TDTR***). Das ***CVTR*** muß gesetzt werden. Es ist darauf zu achten, das die ***FCAT*** des Masters vorher abgezogen wird.

1 Der Master setzt das ***GO*** Signal, welches hier als ***GO_RING*** im Status Register angezeigt wird. .

3 Der erste Trigger wird erkannt. Der 10 MHz Timer wird zurückgesetzt. Der Event Counter wird inkrementiert.

4 Der 10 MHz Timer erreicht die Conversion Time (***CVTR***). Das ***EOC*** Flag wird gesetzt, welches einen Interrupt auslösen kann. Gleichzeitig werden die unteren 3 Bit's des EVC mit den 3 Bit auf dem RING BUS verglichen. Bei einer Abweichung wird das ***SI*** Flag im Status Register und auf den RING BUS gesetzt.

5 Falls ein anderer SLAVE eine Abweichung des eigenen Event Counters erkennt, erscheint das ***SI_RING*** Signal auf dem Trigger Bus. Dieses Signal kann einen Interrupt auslösen und eine Fehlerbehandlung starten.

6 Wenn alle Daten ausgelesen sind, werden die Flags EOC und TDT durch das Kommando ***RESET_TRIGGER*** zurückgesetzt. Das Signal TDT_RING ist nur das Summen Signal der vorhergehenden Stationen und deshalb nicht als gesamt TDT zu verwerten.

7 Wenn der letzte Slave sein TDT Signal zurückgesetzt hat, werden vom Master die Trigger Signale vom RING BUS genommen.

8 Der nächste Trigger wird erkannt.

Die folgenden Beispiele in C sind nicht vollständig. Sie sollen nur auszugsweise kurze Hinweise auf die Programmierung des Synchronisation Modules geben.

```
// Vendor/Device ID

#define AMCC_VENDOR_ID  0x10E8
#define SYNC_DEVICE_ID  0x812F   // Sync/Trigger Module

/* PCI Bus Operation Register (see AMCC Manual) */

typedef struct {
   DWORD omb[4],     // Outgoing Mailbox Register
         imb[4],     // Incoming Mailbox Register
         fifo,       // FIFO Register (bidirectional)
         mwar,       // Master Write Address Register, a0/a1 wired as zeros
         mwtc,       // Master Write Transfer Count Register, in bytes
         mrar,       // Master Read Address Register, a0/a1 wired as zeros
         mrtc,       // Master Read Transfer Count Register, in bytes
         mbef,       // Mailbox Empty/Full Status
         intcsr,     // Interrupt Control/Status Register
         mcsr;       // Bas Master Control/Status Register
} AMCC_REGS;

// Bit's of the Bas Master Control/Status Register

#define MBX_RST      0x08000000L    // Mailbox Flags Reset
#define AD2FIFO_RST  0x04000000L    // Add-on to PCI FIFO Status Flags Reset
#define FIFO2AD_RST  0x02000000L    // PCI to Add-on FIFO Status Flags Reset
#define ADDON_RST    0x01000000L    // Add-on Reset


/* Device Register */

typedef struct {
   DWORD sm_csr,     // Control/Status Register
         sm_cr,      // Control Register
         sm_cvt,     // Conversion Time
         sm_puwi,    // Pulse Width Control
         sm_tmc,     // Time Counter
         sm_tdt,     // Total Dead Time
         sm_fcat,    // Fast Clear Acceptance Time
         sm_free0,
         sm_evc,     // Event Counter
         sm_rej,     // Reject Counter
         sm_gap;     // Trigger GAP FIFO
} SYNC_REGS;


// Bit's of the Control/Status Register

#define ST_MASTER    0x00000001L
#define ST_GO        0x00000002L
#define ST_T4_UTIL   0x00000004L
#define ST_TAW_ENA   0x00000008L
#define ST_AUX_MASK  0x00000070L
#define ST_AUX0_RST  0x00000080L
#define ST_TRIGGER   0x000F0000L
#define ST_GO_RING   0x00100000L
#define ST_SI        0x00400000L
#define ST_INH       0x00800000L
#define ST_EOC       0x08000000L
#define ST_SI_RING   0x10000000L
#define ST_GAP_DATA  0x40000000L
#define ST_TDT_RING  0x80000000L

#define SC_RST_TRG   0x80000000L

// Bit's of the GAP Register

#define GAP_OVERRUN  0x20000000L
#define GAP_MORE     0x40000000L
#define GAP_NO_GAP   0x80000000L


void *map_region(int vendor_id, int device_id, int index, int region);

AMCC_REGS   *amcc_regs;
SYNC_REGS   *sync_regs;
```

Initialisierung

```
amcc_regs=map_region(AMCC_VENDOR_ID, SYNC_DEVICE_ID, 0, 0);
sync_regs=map_region(AMCC_VENDOR_ID, SYNC_DEVICE_ID, 0, 1);

   amcc_regs->mcsr=ADDON_RST;    /* activate SYSRESET */
   amcc_regs->mcsr=0;
```

SYSRESET hat die gleiche Wirkung wie das folgende ***MASTER_RESET***. Das ***SYSRESET*** wird auch durch das PCI RST (z.B. Power Up) kurzzeitig aktiviert.

```
sync_regs->sm_cr  =ST_MASTER;       // Set MASTER
sync_regs->sm_evc =0;               // Master Reset
sync_regs->sm_evc =0;               // Master Reset
```

Es kann sinnvoll sein, das ***MASTER_RESET*** zweimal auszuführen, wenn man sicher sein will, daß auch das TDT Register auf null gesetzt wird. Wenn beim ersten ***MASTER_RESET*** auch das Signal ***TDT_BUS*** zurückgesetzt wird, wird mit der Rückflanke das TDT Register sofort wieder gesetzt.

```
   region.amcc_regs->intcsr=0x1F00;       // Mailbox #4, Byte 3 Interrupt
   sync_regs->sm_csr=0x7F000000L;         // clear any pending interrupt

   sync_regs->sm_cr  =0x0800 |ST_GO;      // enable EOC Int, set GO
   sync_regs->sm_csr =SC_RST_TRG;         // Reset Trigger
```

Interrupt Routine

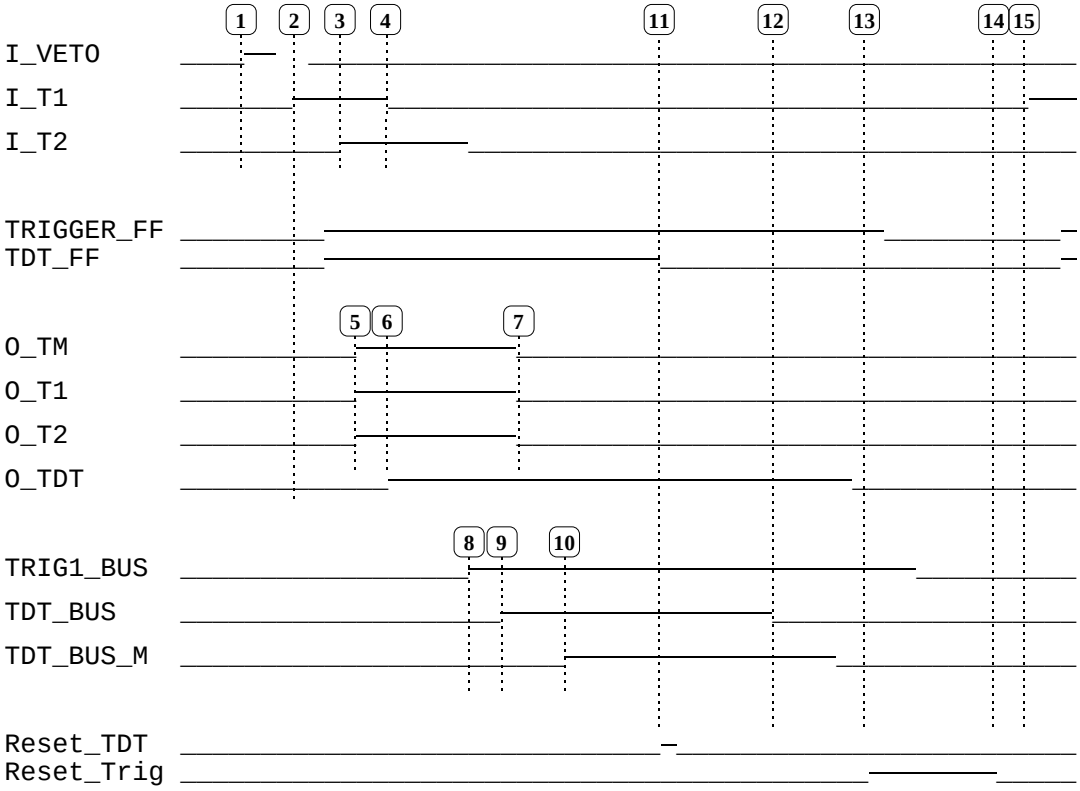
Das folgende Beispiel gilt für einen AT-PC, wobei der PCI Interrupt Request zwischen INT8 und INT15 liegt.

```
//
//============================================================================
//                              Interrupt Handler                            =
//============================================================================
//
//--------------------------- pci_isr ----------------------------------------
//
void interrupt pci_isr(void)
{
   u_long   intmask;
   u_long   intreq;
   u_int    ux;

   ux=0; intmask=sync_regs->sm_csr <<16;
   intreq=amcc_regs->imb[3] &intmask;

   do {
      sync_regs->sm_csr=intreq;
      if (intreq &0x01000000L) int_src[0]++;
      if (intreq &0x02000000L) int_src[1]++;
      if (intreq &0x04000000L) int_src[2]++;
      if (intreq &ST_EOC) {
         int_src[3]++;
         sync_regs->sm_csr =SC_RST_TRG;
      }

      if (intreq &ST_SI_RING) int_src[4]++;
      if (intreq &0x20000000L) int_src[5]++;
      if ((intmask &0x40000000L) && (sync_regs->sm_csr &ST_GAP_DATA)) {
         do int_src[6]++;
         while ((sync_regs->sm_gap &GAP_MORE) && (++ux < 10));
      }

      int_cnt++;
      intreq=amcc_regs->imb[3] &intmask;
   } while ((intreq &~ST_SI_RING) && (++ux < 10));

   amcc_regs->intcsr=amcc_regs->intcsr;
   if (IntCtrl == 0xA1) outportb(0xA0, 0x20);   // End Of Interrupt
   outportb(0x20, 0x20);
   pci_req=1;
}
```

5 Timing der ersten Version

5.1 Master

Alle Signale sind am Stecker gemessen.



$t_{1\text{-}2}$ >1ns, setup time für VETO gegenüber Trigger Signalen. Die Trigger Eingänge sind blockiert, wenn das VETO Signal vorher anliegt.

$t_{2\text{-}3}$ >1ns, setup time für den ersten Trigger gegenüber folgenden Trigger Signalen. Die restlichen Trigger Eingänge sind nach dieser Zeit blockiert, wenn das erste Trigger Signal anliegt.

$t_{2\text{-}4}$ >2ns, Mindestbreite der Trigger Signale.

$t_{2\text{-}5}$ ca. 6ns, Verzögerung der Trigger Signale bis zum ECL Output.

$t_{2\text{-}6}$ ca. 6ns, Verzögerung des TDT Signals bis zum ECL Output gegenüber dem ersten Trigger Impuls.

$t_{5\text{-}7}$ ca. 40ns, Breite der Trigger Signale am ECL Output.

$t_{2\text{-}8}$ ca. 12ns, Verzögerung der Trigger Signale bis zum Trigger Bus.

$t_{2\text{-}9}$ ca. 13ns, Verzögerung des TDT Signals bis zum Trigger Bus.

$t_{2\text{-}10}$ ca. 20ns, Verzögerung des TDT_BUS Signals bis zum MACH Baustein (TDT wird hier zum Trigger Bus gesendet und dann vom Trigger Bus empfangen).

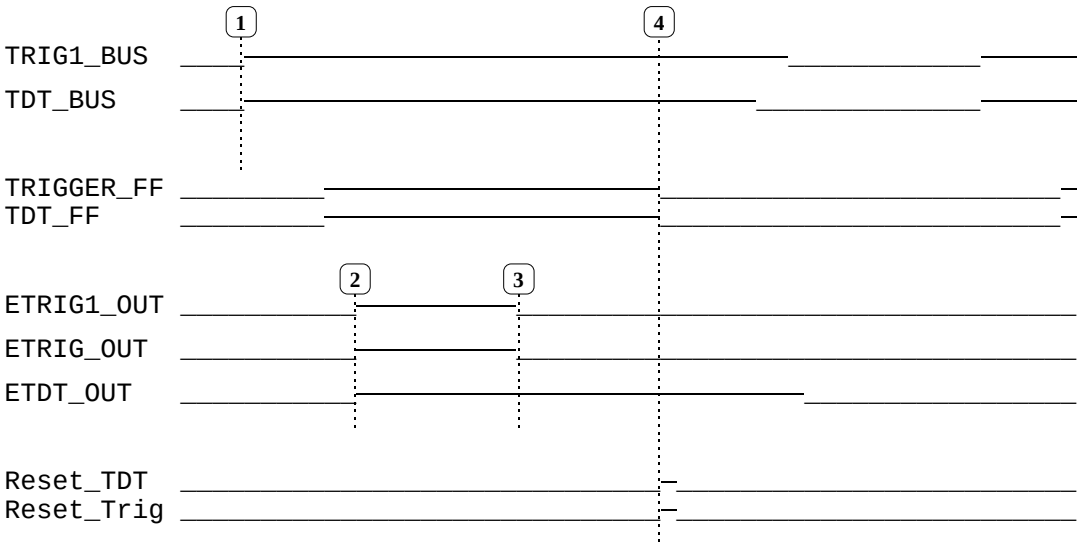
t_{11} hier wird durch Software das TDT FF zurückgesetzt (der Ausgang liegt auf dem Trigger Bus).

$t_{12\text{-}13}$ ca. 20ns, nach dem Verschwinden des TDT Signals vom Trigger Bus wird ein Reset Signal für Trigger FF's erzeugt.

$t_{13\text{-}14}$ ca. 1-2µs, das Reset Signal ist so lang, daß die GSI Trigger Module wieder bereit sind, weitere Trigger Signale zu akzeptieren.

t_{15} ab diesem Zeitpunkt können neue Trigger erkannt werden.

5.2 Slave



Hier liegen noch keine Meßergebnisse vor. Die Werte sind vorerst geschätzt.

$t_{1\text{-}2}$ ca. 18ns, Verzögerung vom Trigger Bus bis zum ECL Output.

$t_{2\text{-}3}$ ca. 40ns, Breite der Trigger Signale am ECL Output.

t_4 hier werden durch Software das TDT FF und die Trigger FF's zurückgesetzt.