# TIP - CFG

## Colour Frame Grabber

Technische Documentation

Version 1.0 - Februar 1993



Die vorliegende Dokumentation bezieht sich auf die Revision 1.1 des TIP-CFG.

# Inhaltsverzeichnis

# 1. Einleitung

Das TIP-CFG ist ein transputerbasierter Hochleistungs - Farb - Frame - Grabber mit TIP - Bus - Interface. Der Transputerknoten ist mit einem T805/30MHz, sowie 4 MByte DRAM ausgerüstet. Die gesamte Videologik läßt sich mithilfe des Transputers steuern, dadurch entfällt das lästige Umstecken von Jumpern.

Das Videointerface des TIP-CFG ist so gestaltet, daß es mit nahezu jeder R/G/B-Video-Quelle klarkommen müßte. Das TIP-CFG läßt sich entweder extern mit den Synchronisationssignalen der Kamera steuern, wenn dies nicht erwünscht ist, oder aber die Kamera keine Synchronisationssignale erzeugen kann, so kann auch das TIP-CFG die Generierung der Synchronisationssignale übernehmen. Hierfür sind ein programmierbarer Pixel - Clock - Generator und ein ebenfalls programmierbarer Video - Timing - Generator vorhanden. Damit ist es möglich, mit jedem gewünschten Video - Timing zu arbeiten.

Für jeden Farbkanal ist ein 8 - bit A/D - Wandler vorhanden. Die maximale Abtastrate der Wandler beträgt 20 MSamples/s. Die Offset- und Gain - Werte der A/D - Wandler können durch Software eingestellt werden. Optional kann vor die Wandler noch ein Anti-Aliasing - Filter geschaltet werden.

Auf dem Board sind 1,5 MByte Videospeicher vorhanden, der in Worten mit 24 Bit organisiert ist. Das Videointerface kann das VRAM in zwei verschiedenen Modi adressieren. Im Zeilenmodus werden die Daten jeder Videozeile in einer eigenen Speicherzeile abgelegt. Dabei werden die Daten für das ungerade- und das gerade Halbbild bereits richtig im Speicher zusammengefügt. Die horizontale Auflösung ist dabei jedoch auf die Länge einer Speicherzeile begrenzt (zur Zeit 512 Pixel). Im konsekutiven Adressierungsmodus werden die Daten der Reihe nach in den Videospeicher geschrieben. Dadurch kommen die Daten der beiden Halbbilder übereinander im Speicher zu liegen. Das Zusammenmischen der Bilddaten zu einem Vollbild wird hier mittels der Busübertragung vollzogen. Die horizontale Auflösung des Bildes ist hierbei nicht begrenzt.

In erster Näherung läßt sich der Framegrabber in vier Blöcke einteilen:

- Videospeicher
- TIP-Bus-Controller
- Videointerface
- Transputerknoten

Der Videospeicher stellt bei den obigen Blöcken ein zentrales Element dar, weil die anderen Blöcke alle darauf zugreifen. Das Videointerface muß die digitalisierten Daten in den Videospeicher schreiben, der Tip-Bus-Controller liest sie zur Weiterverarbeitung aus. Der Transputer kann natürlich ebenfalls auf den Videospeicher zugreifen. Wegen der zentralen Bedeutung des Videospeichers wird dieser deshalb zuerst beschrieben.

# 2. Der Videospeicher

Das größte Problem des Videospeichers sind die großen Datenmengen, die gleichzeitig gespeichert und ausgelesen werden müssen. Das Videointerface kann bis zu 20 MSamples/s an Videodaten liefern. Der TIP-Bus andererseits hat eine Übertragungskapazität von 100 MByte/s. Damit der Videospeicher dies alles verkraftet, dürfen sich die Datenströme gegenseitig möglichst nicht behindern. Häufig wird dies durch ein Konzept gelöst, das double-buffering heißt. Dabei ist der Videospeicher in zwei Bänke eingeteilt. Während in die eine Bank das neue Bild gespeichert wird, wird aus der anderen Bank das vorige Bild ausgelesen. In industriellen Echtzeitanwendungen scheidet diese Möglichkeit jedoch aus, weil die Totzeiten zwischen Bildaufnahme und Bildweiterverarbeitung zu lange werden. Bei dem hier beschriebenen Framegrabber wird dieses Problem durch Verwendung von speziellen Videospeichern gelöst. Diese Bausteine besitzen neben einem parallelen Port zwei getrennte serielle Ports. Der Transputer greift hiebei auf die parallele Seite zu, während je eine serielle Seite für TIP-Bus-Interface und für das Videointerface vorbehalten sind.

In ihrem Kern bestehen die Triple Port VRAMs aus einem Array von 512*256*8 Speicherzellen. Durch Anlegen einer 9-Bit Zeilenadresse und einer 8-Bit Spaltenadresse kann auf jede Speicherzelle wahlfrei zugegriffen werden. Mit diesen Bausteinen wird nun der Videospeicher gemäß Bild 1 aufgebaut.

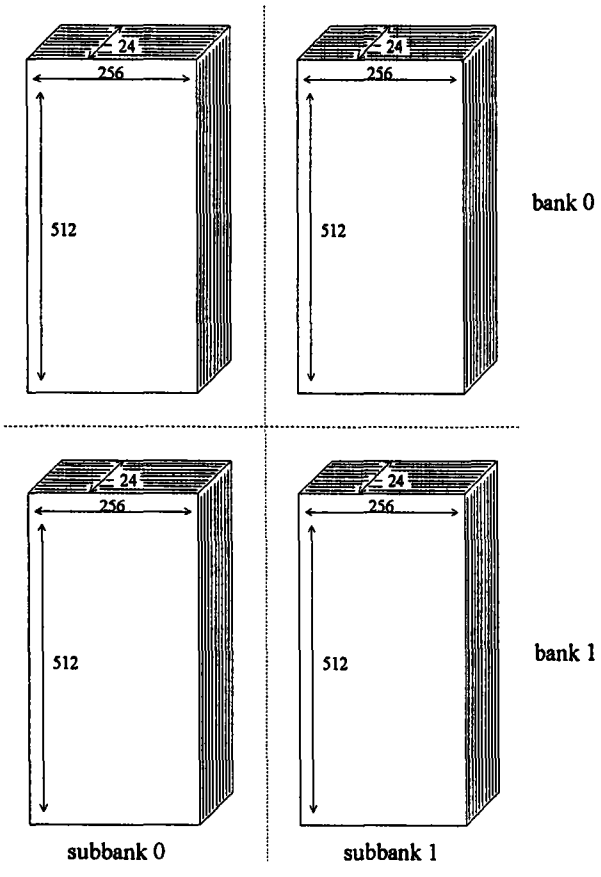



*Bild 1: Organisation des Videospeichers*

Durch Parallelschaltung von drei Speicherbausteinen wird eine Datenwortbreite von 24 Bit erreicht. Da jeder Farbwert der Kamera mit 8 Bit quantisiert wird, reicht ein Datenwort genau zur Speicherung eines Pixels. In Bild 2 wird gezeigt, wie die verschiedenen Farbanteile eines Pixels in einem Datenwort des Videospeichers abgelegt werden.

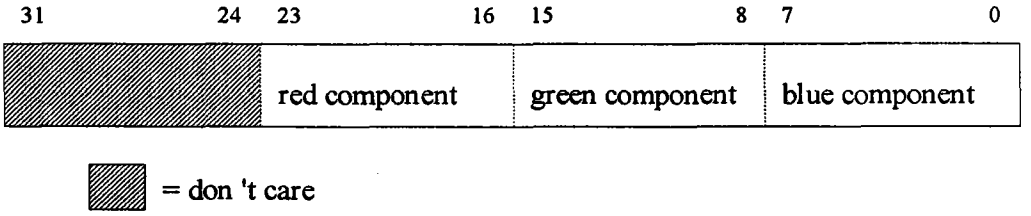



*Bild 2: Organisation eines Datenwortes im Videospeicher*

Der Transputer, der natürlich weiterhin 32-Bit-Zugriffe fährt, erhält für das obere Byte einen Zufallswert zugewiesen. In horizontaler Richtung ist der Gesamtspeicher in zwei Subbänke unterteilt. Damit wird insgesamt eine Zeilenbreite des Videospeichers von 512 Pixeln erreicht. In der vertikalen Dimension ist der Videospeicher in zwei Bänken, von je 512 Zeilen, organisiert. Damit ergibt sich eine Gesamtspeichergröße von 512*1024*24 Bit, oder 1,5 MByte. Die Adressleitungen des Parallelports wurden so geschickt geschaltet, daß aus der Sicht des Transputers auf eine Speicherzeile von Bank 0 immer eine Speicherzeile von Bank 1 folgt. Will der Transputer also linear den gesamten Speicher mit einem bestimmten Wert vorbesetzen, so beschreibt er zuerst die erste Zeile von Bank 0, anschließend die erste Zeile von Bank 1, dann die zweite Zeile von Bank 0, usw. . .

Das Videointerface kann in zwei verschiedenen Modi auf den Speicher zugreifen:

- Konsekutiver Modus
- Zeilenmodus.

Beim konsekutiven Modus werden die Daten, aus der Sicht des Transputers, in konsekutiv aufeinanderfolgenden Speicherzellen abgelegt. Bezüglich Bild 1 bedeutet dies, daß immer zeilenweise zwischen Bank 0 und Bank 1 gewechselt wird. Im konsekutiven Modus wird die verfügbare Speicherkapazität optimal ausgenutzt. Da die Daten jedoch in zwei Halbbildern angeliefert werden, stehen die Videodaten der beiden Halbbilder untereinander im Speicher. Dies verkompliziert natürlich die Adressierung beim Auslesen der Bildinformation.

Um diesen Nachteil zu beheben, wurde der Zeilenmodus geschaffen. Der Zeilenmodus funktioniert nur dann, wenn eine Zeile des Bildes nicht größer als eine Bildspeicherzeile ist. Bei diesem Modus wird mit jedem HSYNC, der ja den Beginn einer Bildzeile kennzeichnet, am Anfang einer neuen Speicherzeile zu schreiben begonnen. Dabei wird nicht etwa nach jedem HSYNC die Bank gewechselt, sondern es werden die Bildzeilen des ungeraden Halbbildes in Bank 0 und die geraden Bildzeilen in Bank 1 geschrieben. Aus der Sicht des Transputers, bzw des Businterfaces werden dadurch aber die Halbbilder richtig zusammengemischt, weil diese ja immer abwechselnd auf die beiden Bänke zugreifen. Da eine horizontale Auflösung von 512 Pixeln etwas knapp werden könnte, ist bereits eine Version mit dem doppelten Speicherausbau in Vorbereitung. Damit steht dann ein Videospeicher von 1024*1024*24 Bit zur Verfügung.

# 3. Der TIP-Bus-Controller

Alle Module der TIP-Serie hängen an einem gemeinsamen parallen Datenbus, über den sie Informationen austauschen. Die Datenbusbreite beträgt 32 Bit, dabei werden die Daten mit 25 MHz Taktfrequenz übertragen. Ein Modul kann dabei drei verschiedene Zustände, bezüglich der Buskommunikation, annehmen:

- Master, das Modul sendet Daten,
- Slave, das Modul empfängt Daten,
- Inaktiv, das Modul beteiligt sich nicht an der Datenkommunikation.

Der TIP-Bus wird nach einem Kanalkonzept verwaltet. Als Kanal wird dabei eine unindirektionale Verbindung zwischen einem Master und einem Slave, oder mehrerer Slaves, bezeichnet. Damit lassen sich drei prinzipielle Kommunikations-Topologien realisieren:

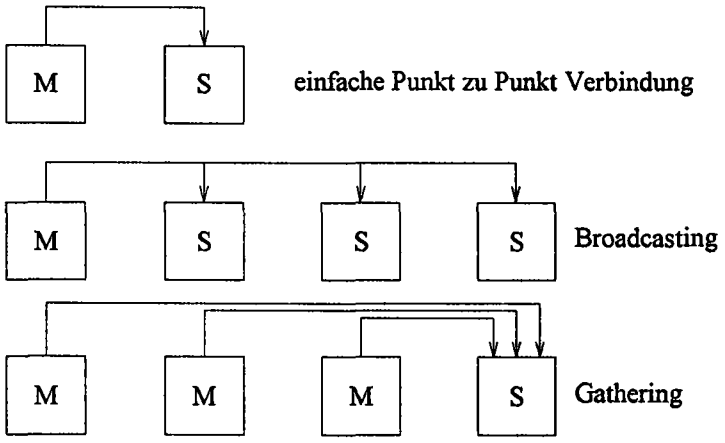



*Bild 3: Kommunikations-Topologien beim TIP-Bus*

Die einfache Punkt zu Punkt Verbindung kann zum Beispiel zwischen Framegrabber und Grafikkarte verwendet werden. Im Broadcasting-Mode sendet der Master ein Bild an mehrere Slaves gleichzeitig. Jeder Slave kann dann einen Teil des Bildes verarbeiten, zum Beispiel eine Farbraumkonvertierung durchführen. Mittels des Gathering-Modes, können dann die verschiedenen Teilbilder wieder zusammengefügt werden. Dabei wird von jedem Master jeweils ein Kanal zu einem gemeinsamen Slave definiert.

Die zeitliche Verwaltung der verschiedenen Kanäle geschieht nach einem Zeitscheibenverfahren. Die Zeitscheibe wird dabei in mehrere Datentransfer-Slots unterteilt. Jedem Slot wird dabei ein Kanal zugeteilt. Ein Kanal kann mehrere Slots belegen, die dabei nicht aufeinander folgen müssen. Die Zeitscheibe, mit ihrer Slotaufteilung ist auf allen Modulen identisch. Alle Zeitscheiben werden synchron getaktet. Ein Slot ist in verschiedene Blöcke unterteilt, die unterschiedliche Größen aufweisen dürfen. Ein Block definiert dabei einen zusammenhängenden Speicherbereich im Videospeicher. Der TIP-Bus kann also nur Daten zwischen den Videospeichern der einzelnen Module austauschen, nicht etwa zwischen den Arbeitsspeichern der Transputer.

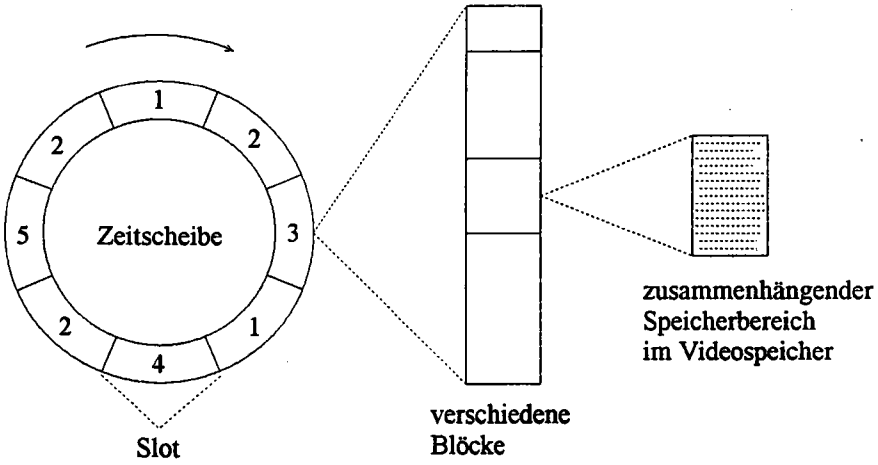



*Bild 4: Verwaltung der Kanäle nach dem Zeitscheibenverfahren*

Zur hardwaremäßigen Realisierung werden drei Listen verwendet, die über Zeiger miteinander verknüpft sind. Diese Listen heißen:

- Channel Number RAM
- Parameter RAM
- VRAM Address RAM.

Das Channel Number RAM ist eine lineare Liste, die globale Gültigkeit besitzt, d. h. diese Listen sind auf allen Modulen gleich. Jeder Listeneintrag besteht aus einem 32-Bit-Wort. In der Liste werden die Kanalnummern in der Reihenfolge, wie sie auf der Zeitscheibe liegen, eingetragen. Dafür stehen pro Eintrag 8 Bit zur Verfügung. Ein Bit ist zur Kennzeichnung des Endes der Liste reserviert, die restlichen Datenbits werden nicht genutzt. Die Kanalnummer des Listeneintrages wird zur Adressierung des Parameter RAMs benutzt.

Im Parameter RAM werden die Eigenschaften eines Kanales näher beschrieben. Für jeden Kanal gibt es demnach einen Eintrag, der aus 8 32-Bit-Worten besteht. Hier wird z. B. festgelegt, ob ein Modul für einen Kanal als Master oder als Slave agiert, oder ob es gar inaktiv ist. Daraus folgt, daß diese Liste nur lokale Gültigkeit besitzen kann. Außerdem werden in der Liste zwei Zeiger zur Adressierung des VRAM-Address-RAM, und einige Parameter zur Datenvalidierung und Synchronisation (siehe hierzu Tipset Dokumentation) deklariert. Die beiden Zeiger heißen VRAM_Start und VRAM_Temp.

Im VRAM Address RAM steht zu jedem Block eines Kanales ein Eintrag. Die verschiedenen Übertragungsblöcke, die für einen Kanal der Reihe nach abgearbeitet werden sollen, müssen nacheinander in dieser Liste beschrieben werden. Die Einträge in diese Liste sind je nach Speicherorganisation der verschiedenen TIP-Produkte etwas unterschiedlich aufgebaut. Bild 5 zeigt den Aufbau eines Eintrages in das VRAM-Address-RAM für den Farbframegrabber.

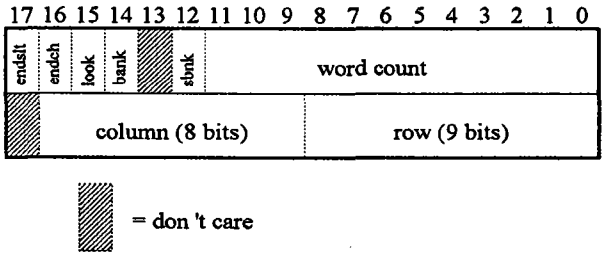



*Bild 5: Aufbau eines Eintrages im VRAM Address RAM*

Die Art und Weise, wie hier die Startadresse eines Blocks angegeben wird, spiegelt die Organisationsstruktur des Videospeichers wieder. Die Adresse wird definiert, durch die Angabe von Row-Adresse, Column-Adresse, Bank und Subbank (siehe hierzu auch Bild 1 auf Seite 2). Aus diesen Angaben wird folgendermaßen die VRAM-Adresse zusammengesetzt:

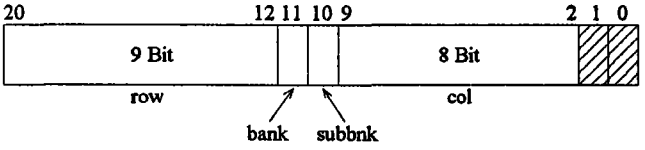



*Bild 6: Adressbildung zur VRAM-Adressierung*

Die Länge eines Datenblockes wird durch das word count Feld festgelegt. Da hierfür 12 Bit vorhanden sind, kann die maximale Blockgröße $2^{24}$ Worte , bzw. 16 kByte betragen. Mit dem look-Bit wird die, bereits oben erwähnte, Datenvalidierung veranlaßt.

Wird der erste Block eines Kanales übertragen, so wird der zugehörige Eintrag im VRAM Address RAM, mit dem Zeiger VRAM_Start, aus dem Parameter RAM angewählt. Bei jedem Blockwechsel wird nun durch einen Zähler die Adresse des VRAM-Address-RAMs erhöht, sodaß damit automatisch zum nächsten Eintrag übergegangen wird. Der letzte Block eines Slots wird durch das Bit endslt im VRAM-Address-RAM markiert. Da ein Kanal sich jedoch über mehrere Slots erstrecken kann besteht die Notwendigkeit, sich die letzte Adresse des VRAM-Address-RAMs zu merken, damit im nächsten Slot des Kanales mit dieser fortgefahren werden kann. Diese Adresse wird deshalb im Zeiger VRAM_Temp des Parameter-RAMs zwischengespeichert. Wird in einem der nächsten Slots der Kanal wieder aktiv, so beginnt die Adressierung des VRAM-Address-RAMs mit VRAM_Temp, anstatt mit VRAM_Start, beim allerersten Block. Beim letzten Slot, den ein Kanal auf einer Zeitscheibe belegt, ist für den letzten Block zusätzlich zum endslt-Bit, noch das endch-Bit gesetzt. Bei der nächsten Aktivierung des Kanals, wird deshalb die Adressierung des VRAM-Address-RAMs wieder mit VRAM_Start begonnen.

### 3.1. Das Go Register ($0000 1000)

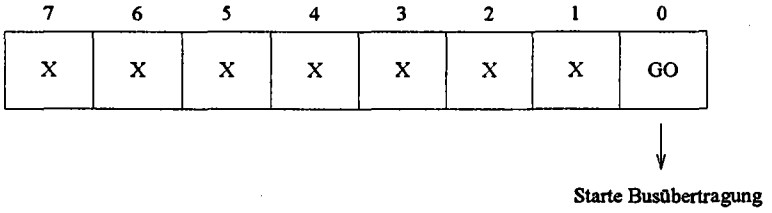



*Bild 7: Bit-Belegung des Go-Registers*

Wird in das GO-Bit eine "1" geschrieben, so wird die Übertragung auf dem TIP-Bus gestartet. Dies wird nach der Initialisierung der Tabellen gemacht. Das GO-Bit ist rücklesbar.

### 3.2. Das Bus Reset Register ($0000 0300)

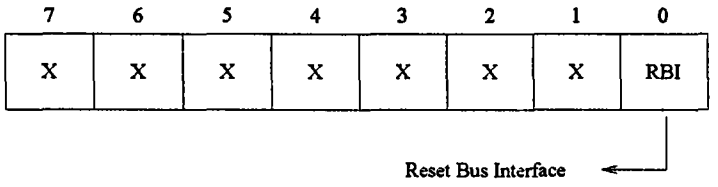



*Bild 8: Bit-Belegung des Bus Reset Registers*

Wird in dieses Register in Bit 0 eine "1" geschrieben, so wird der Bus Overall Controller des TIP-Buses in einen definierten Zustand gebracht. Das Bit braucht nicht zurückgesetzt werde. Im Gegensatz zum Link Reset Register gibt es hier keinen Schutz gegen zufälliges Beschreiben des Registers.

# 4. Die Transputersektion

In Bild 9 sind die wesentlichsten Elemente des Transputerknotens skizziert. Kernstück des Transputerknotens ist ein T805-Transputer der mit maximal 30 MHz getaktet werden kann. Der Rechnerknoten ist mit maximal 4 MByte DRAM ausrüstbar. Die Einstellung der Taktrate für den Transputer, die Link-Geschwindigkeit, sowie die Anzahl der Zyklen für den DRAM-Zugriff, sind per Jumper einstellbar.

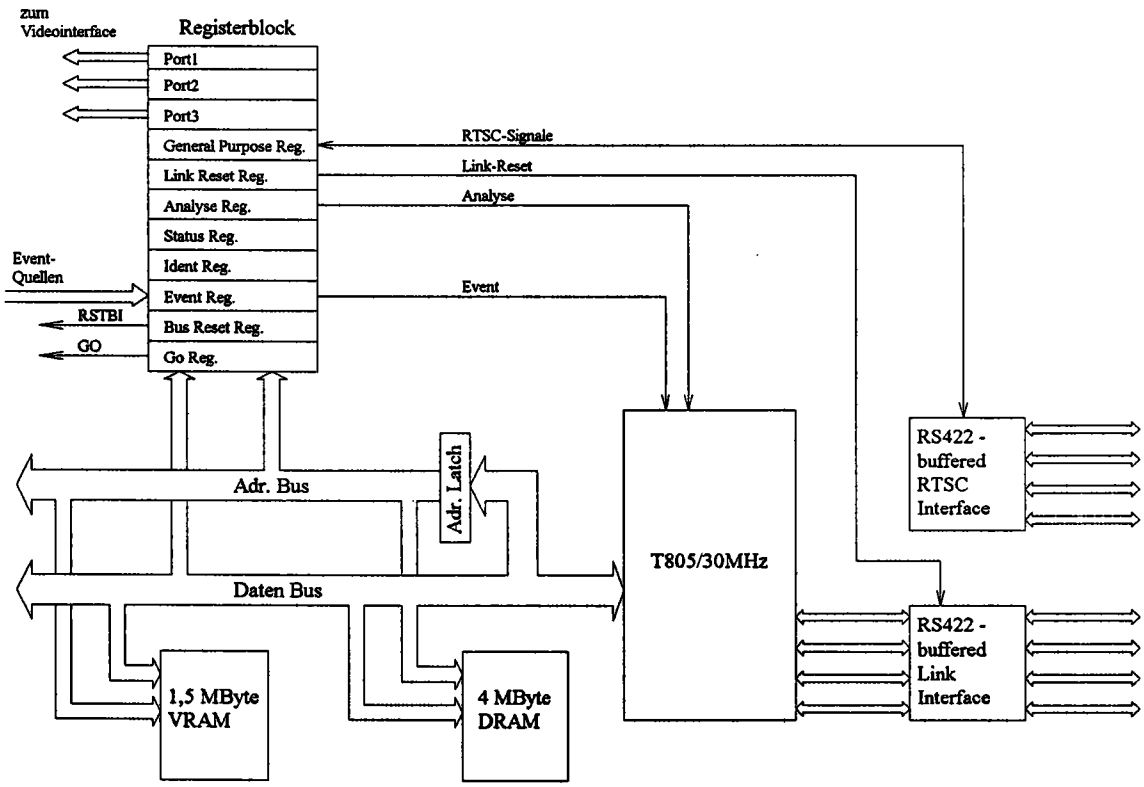



*Bild 9: Blockdiagramm des Transputerknotens*

In dem Registerblock in obiger Zeichnung sind alle Einzelregister zusammengefaßt. Die genaue Funktion dieser Register wird im nächsten Kapitel behandelt. Neben dem Link-Interface gibt es noch das RTSC-Interface (Real Time Sync. Control). Dies umfaßt eine Gruppe von vier Synchronisationssignalen, mit denen alle Module über eine Daisy-Chain verbunden sind. Den vier Kanälen sind dabei folgende Funktionen zugeordnet:

RTSC0: Dieser Kanal dient dazu, die Übertragung des TIP-Buses anzuhalten, nachdem dieser einen Synchronisationsslot erreicht hat. Wird diese Leitung von einem oder mehreren Slavemodulen aktiviert, so wird die Busübertragung beim erreichen eines Synchronisationsslots angehalten. Erst wenn alle Slave-Module diesen Kanal deaktiviert haben, wird die Busübertragung fortgesetzt.

RTSC1: Die Benutzung dieses Kanals ist dem Anwender überlassen. Wenn dieser Kanal von einem Modul aktiviert wird, so ist dies auf allen anderen Modulen durch ein Bit im General Purpose Register erkennbar.

RTSC2: Dieser Kanal wird von einem Modul aktiviert, wenn ein Fehler auftritt. Auf den anderen Modulen kann daraufhin ein Event generiert werden. Äußerlich ist das Modul, auf dem der Fehler auftrat, daran zu erkennen, daß die rote Fehler-LED angeschaltet wird.

RTSC3: Dieser Kanal dient zur synchronen Initialisierung aller Module. Wird er von einem Modul aktiviert, so wird auf allen anderen Modulen ein Reset ausgelöst.

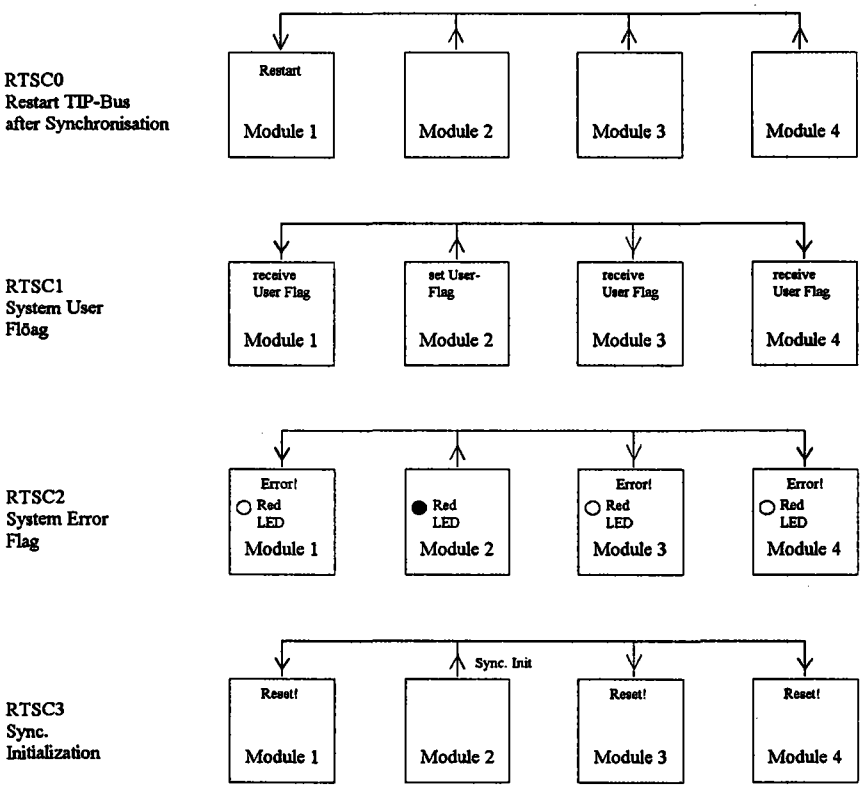



*Bild 10: Funktionsprinzip der RTSC-Kanäle*

## 4.1. Festlegung der Transputer-Parameter durch Jumper

Die meisten Konfigurationen des Farbframegrabbers können durch Software eingestellt werden. Die Betriebsparameter des Transputers müssen jedoch schon zum Zeitpunkt des Bootvorganges korrekt eingestellt sein. Diese werden deshalb durch Jumper eingestellt. Da diese Jumper jedoch einmal eingestellt werden und danach in der Regel nie mehr verändert werden, stellt dies keinen Nachteil dar. Das Jumperfeld befindet sich direkt über dem Transputer und ist in drei Gruppen unterteilt.

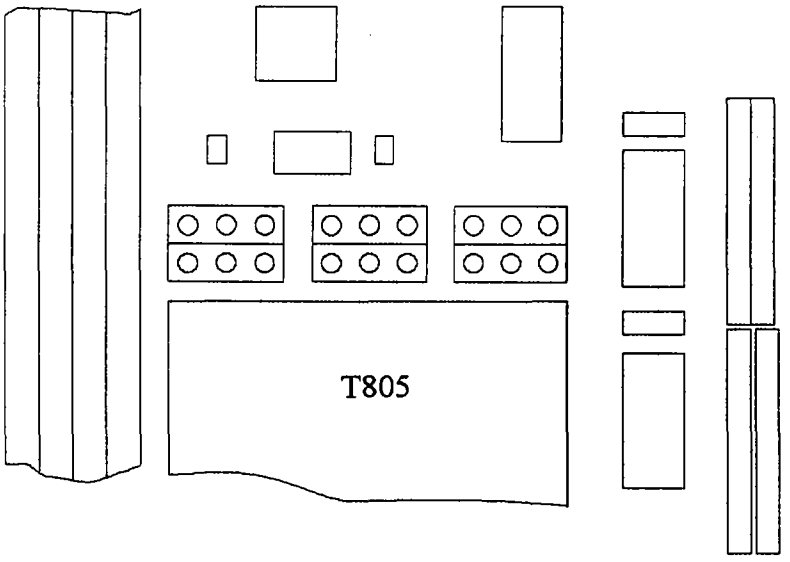



*Bild 11: Lage des Jumperfeldes zur Einstellung der Betriebsparameter des Transputers*

Für den Prozessortakt können drei verschiedenen Frequenzen gewählt werden. Hier ist die Frequenz gemäß dem Aufdruck auf dem Transputer zu wählen.

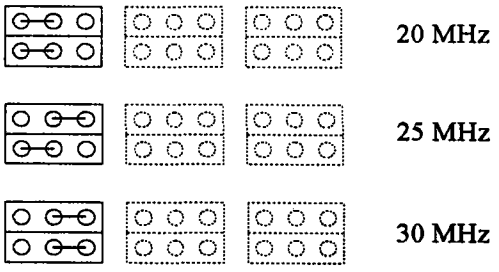



*Bild 12: Einstellung des Prozessortaktes*

Abhängig vom Prozessortakt und der Zugriffsgeschwindigkeit der DRAMs ist die Einstellung der Prozessorzyklen, die beim DRAM-Zugriff verwendet werden. Bei den heute verwendeten DRAMs mit 60 ns Zugriffszeit, genügt eine Einstellung von 4 Prozessorzyklen. Ein externer Speicherzugriff dauert damit 133 ns.

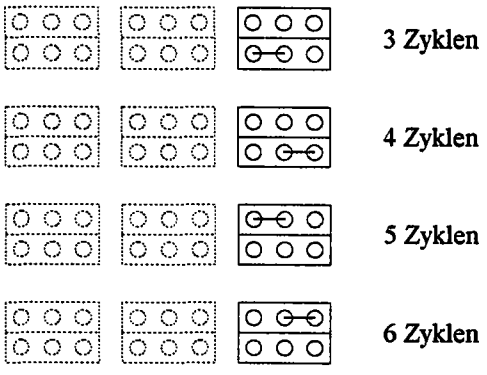
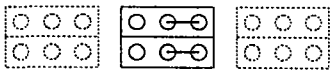
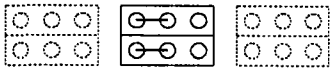



*Bild 13: Einstellung der Prozessorzyklen für den externen Speicherzugriff*

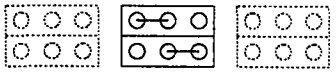
Der letzte per Jumper einstellbare Parameter ist die Linkgeschwindigkeit. Hier kann zwischen vier Möglichkeiten gewählt werden. Einmal können alle Links mit einer einheitlichen Übertragungsrate von 10 oder 20 MByte/Sekunde betrieben werden. Außerdem kann aber Link0 mit einer langsameren, bzw. schnelleren Übertragungsrate als die restlichen Links betrieben werden.



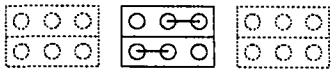
Alle Links: 20 MB/s



Alle Links: 10 MB/s



Link 0: 20 MB/s
Links 1,2,3: 10 MB/s



Link 0: 10 MB/s
Links 1,2,3: 20 MB/s

*Bild 14: Einstellung der Linkgeschwindigkeit*

## 4.2. Übersicht über die Adressenbelegung des Farbframegrabbers

Im folgenden wird ein Überblick über die Belegung des Adressraumes des Framegrabbers gegeben. Sämtliche I/O-Devices sind memory-mapped, auf diese kann durch ganz normale Speicheroperationen zugegriffen werden. Bei jedem Zugriff auf ein I/O-Gerät werden automatisch vier Wait-States eingefügt. Dadurch dauert ein I/O-Zugriff 230 ns. Da diese aber nur in der Initialisierungsphase notwendig sind, wirkt sich der relativ langsame Zugriff nicht auf die Performance des Framegrabbers aus.

**Register der Transputer-Sektion:**

| Adresse | Registername | Breite/Bit | r/w |
|---|---|---|---|
| #0000 0080 | Status Register | 8 | r |
| #0000 0084 | Ident Register | 8 | r |
| #0000 00C0 | Link Reset Register | 8 | w |
| #0000 0180 | General Purpose Register | 16 | r/w |
| #0000 01C0 | Event Register | 16 | r/w |
| #0000 0200 | Analyse Register | 8 | w |

**Adressbereich von DRAM und VRAM**

| Adresse | |
|---|---|
| #8000 0000 | DRAM Startadresse |
| #803F FFFF | DRAM Endadresse |
| #9000 0000 | VRAM Startadresse |
| #901F FFFF | VRAM Endadresse |

**Register des Videointerfaces**

| Adresse | Registername | Breite/Bit | r/w |
|---|---|---|---|
| #0000 0400 | Port Register 0 | 16 | r/w |
| #0000 0440 | Port Register 1 | 8 | r/w |
| #0000 0680 | Port Register 2 | 8 | r/w |
| #0000 0700 | Image Start Register | 8 | w |
| #0000 0480 | programmierbarer Videotakt-Generator | 8 | w |
| #0000 0500 | Zeilenauswahl, Anzahl Zeilen in vert. Austastlücke | 8 | w |
| #0000 0504 | Zeilenauswahl, Anzahl der sichbaren Zeilen | 8 | w |
| #0000 050C | Zeilenauswahl, Controlregister | 8 | w |
| #0000 0540 | Video-Synchronisationssignal-Generator | 16 | w |
| #0000 0580 | Digitalisierung, Command Register | 8 | r/w |
| #0000 0584 | Digitalisierung, obere Referenz für Blau | 8 | r/w |
| #0000 0588 | Digitalisierung, untere Referenz für Blau | 8 | r/w |
| #0000 058C | Digitalisierung, obere Referenz für Grün | 8 | r/w |
| #0000 0590 | Digitalisierung, untere Referenz für Grün | 8 | r/w |
| #0000 0594 | Digitalisierung, obere Referenz für Rot | 8 | r/w |
| #0000 0598 | Digitalisierung, untere Referenz für Rot | 8 | r/w |
| #0000 05C0 | Taktsynchronisierung, Address Register | 8 | r/w |
| #0000 05C4 | Taktsynchronisierung, Datenport | 8 | r/w |

**TIP-Bus Control Sektion**

| Adresse | Registername | Breite/Bit | r/w |
|---|---|---|---|
| #0000 0300 | Bus Reset Register | 8 | w |
| #0000 1000 | Go Register | 8 | w |
| #0002 0000 | Channel RAM Startadresse | 32 | r/w |
| #0003 0000 | Parameter RAM Startadresse | 32 | r/w |
| #0004 0000 | VRAM Address RAM Startadresse | 32 | r/w |

### 4.3. Das Status Register ($0000 0080)

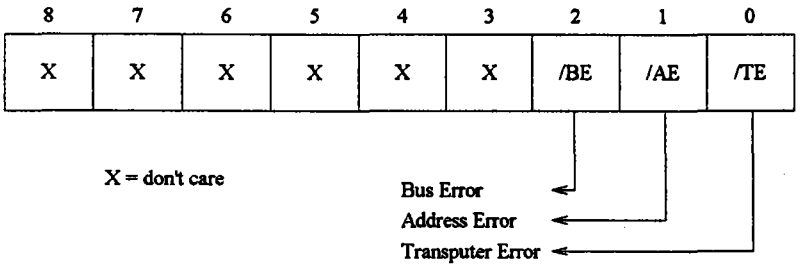



*Bild 15: Bit-Belegung des Status Registers*

Im Status Register werden Fehlerzustände des Framegrabbers angezeigt. Sämtliche Status-Bits sind low-aktiv, d. h. im Falle eines Fehlers wird für das dementsprechende Bit eine "0" gelesen. Wird das Status Register vom Transputer gelesen, so werden danach automatisch alle Bits auf "1" zurückgesetzt. Sollte der Fehler danach immer noch aktiv sein, so wird er erneut ins Status Register geschrieben. Im Falle eines Fehlers kann ein Event ausgelöst werden, wenn dies im Event Register spezifiziert wird. Außerdem wird der Real Time Sync. Kanal 2 aktiviert. Folgende Fehlerzustände werden im Status Register markiert:

/TE: Transputer Error; das Error-Flag des Transputers wurde gesetzt.

/AE: Address Error; der Transputer hat auf eine Speicherstelle zugegriffen, an der weder DRAM, noch VRAM oder ein I/O-Device liegt.

/BE: Bus Interface Error; bei der Synchronisation des TIP-Buses ist ein Fehler aufgetreten.

### 4.4. Das Ident Register ($0000 0084)

Beim Lesen des Ident Registers wird eine Typ-Identifikation zurückgegeben. Dabei handelt es sich nicht etwa um eine Seriennummer, es wird vielmehr der Hardware-Typ, insbesondere die Versionsnummmer, angegeben. Eine Initialisierungs-Software kann so auf einfache Weise die Hardware-Ressourcen in einem System feststellen. Außerdem kann bei der Initialisierung auf kleine Hardwareunterschiede zwischen den einzelnen Versionen eines Produktes Rücksicht genommen werden. Für das TIP-CFG sind dabei folgende Identifikations-Daten reserviert:

1011 0000: für CFG Rev.: 1.1, ohne Hardware Color Space Converter
1011 0001: für CFG Rev.: 1.2, ohne Hardware Color Space Converter
1011 1000: für CFG Rev.: 1.2, mit Color Space Converter

Aus Platzgründen wurde das Identregister im Statusregister mit untergebracht. Dadurch kann der Ident-Code nur über das Status Register gelesen werden. Wird das Ident Register gelesen, so erhält man zunächst nicht die gewünschte Information, vielmehr wird diese in das Statusregister geladen. Beim nächsten Lesezugriff auf das Status Register, wird dann der gewünschte Identifikations-Code gelesen. Zu Beachten ist dabei, daß der Identifikations-Code im Statusregister einen Fehler-Event auslöst, wenn ein Bit zu "0" gesetzt ist. Deshalb muß diese Eventquelle vor der ganzen Prozedur im Event Register ausmarkiert werden. Die folgende Programmsequenz verdeutlicht nocheinmal, wie auf die Ident-Information zugegriffen wird:

```
#define STAT_REG  (int *)0x00000080
#define ID_REG    (int *)0x00000084

getID()
{
  int id_code;

  disable_event(ERR);          /* maskiere Event-Quelle error */
  id_code = *ID_REG;           /* lade ID-Information ins Status Reg. */
  id_code = *STAT_REG;         /* lese ID-Information */
  enable_event(error);         /* aktiviere Event-Quelle error */
  return(id_code);
}
```

### 4.5. Das Link Reset Register ($0000 00C0)

Eine Linkverbindung enthält außer den eigentlichen bidirektionalen Datenleitungen noch bidirektionale Resetleitungen. Damit ist es jedem Transputer möglich, einen oder mehrere seiner vier Nachbartransputer zurückzusetzen. Gesteuert wird dieser Vorgang über das Reset Register. Da ein Reset einen schweren Eingriff in das Transputernetzwerk darstellt, ist dieses Register gegen "versehentliches" Schreiben geschützt. Damit ist es ziemlich unwahrscheinlich, daß ein abgestürztes Programm zufällig noch seine Nachbarn mitreißt. Um einen Reset auszulösen, muß das Register erst entriegelt werden. Dies geschieht, indem nacheinander die Werte 0, 1, 2 und 3 an die Adresse dieses Registers geschrieben werden. Beim nächsten Schreibzugriff auf dieses Register, wird mit den vier niederwertigsten Datenbits die Linknummer spezifiziert, über die ein Reset ausgelöst werden soll. Der Reset bleibt dann solange aktiv, bis eine 0 in das Reset Register ge-

schrieben wird. Bei der Entriegelungssequenz ist darauf zu achten, daß zwischendurch nicht auf eine andere Speicherzelle zugegriffen werden darf. Damit würde der Entriegelungsvorgang sofort unterbrochen werden. Die erforderliche Programmsequenz für einen Link-Reset sieht also folgendermaßen aus:

```
#define LRES_REG  (int *)0x000000C0

res_link (int link_numb)        /* gültige Linknummern sind 0..3 */
{
  *LRES_REG = 0;                /* schreibe Entriegelungssequenz */
  *LRES_REG = 1;
  *LRES_REG = 2;
  *LRES_REG = 3;
  *LRES_REG = 1<<link_numb;     /* aktiviere Resetleitung */
  wait(128);                    /* warte 128 µs */
  *LRES_REG = 0;                /* setze Resetleitung zurück */
}
```

## 4.6. Das General Purpose Register ($0000 0180)

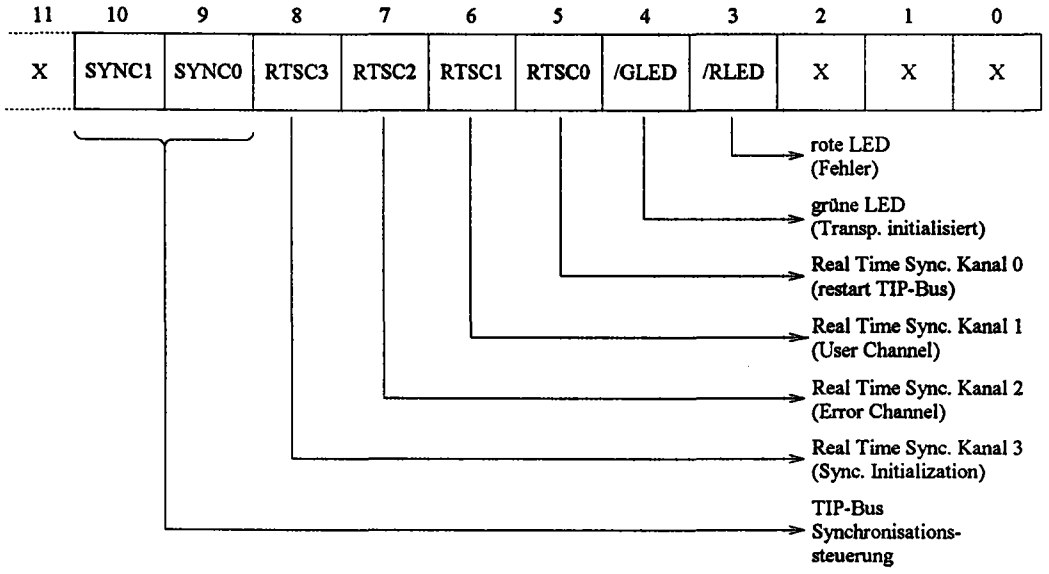



*Bild 16: Bit-Belegung des General Purpose Registers*

Das General Purpose Register ist ebenfalls gegen zufälliges Beschreiben geschützt. Hier müssen, ähnlich wie beim Link Reset Register, die Werte 4, 5, 6 und 7 an die Adresse des General Purpose Registers geschrieben werden, um dieses zu entriegeln. Zum Lesen des Registers ist diese Entriegelungssequenz nicht nötig. Die einzelnen Bits in diesem Register haben dabei folgende Bedeutung:

/RLED: Die rote LED auf dem Framegrabber wird angeschaltet, wenn der Transputer durch einen Reset zurückgesetzt wird, oder wenn ein Fehler auftritt. Sind diese Signale nicht mehr aktiv, so kann die LED durch einschreiben einer "1", in dieses Bit, gelöscht werden. Der Zustand der LED ist über dieses Bit rücklesbar. Aus Platzgründen mußte diese LED leider als SMD-Version auf

die Rückseite des Transputerknotens plaziert werden. Sie ist daher eigentlich nur noch zur Inbetriebnahme zu gebrauchen.

/GLED: Mit diesem Bit kann die grüne LED auf dem Framegrabber gesteuert werden. Durch einschreiben einer "0" wird die LED eingeschaltet. Der Zustand der LED ist über dieses Bit ebenfalls rücklesbar. Diese LED ist eigentlich dafür gedacht, nach dem Boot-Vorgang vom Transputer eingeschaltet zu werden. Dadurch kann erkannt werden, daß der Transputer richtig initialisiert wurde.

RTSC0: Mit diesem Bit kann der Real Time Sync. Kanal 0 aktiviert, bzw. dessen Zustand gelesen werden. Mit diesem Kanal kann die TIP-Bus-Übertragung, in einem Synchronisations-Slots von einem Slave aus angehalten werden, indem der Kanal von diesem aktiviert wird. Wird der Kanal danach wieder deaktiviert, so wird die Busübertragung fortgesetzt.

RTSC1: Hiermit kann der Real Time Sync. Kanal 1 aktiviert, bzw. dessen Zustand gelesen werden. Der Zweck dieses Kanals ist dem Anwender überlassen.

RTSC2: Dieses Bit kann nur gelesen werden, es gibt den Zustand des Real Time Sync. Kanales 2 an. Dieser wird automatisch aktiviert, wenn auf einem Modul ein Fehler auftritt. Der Kanal kann durch einen Schreibvorgang auf dieses Bit nicht aktiviert werden.

RTSC3: Mit diesem Bit kann der Real Time Sync. Kanal 3 aktiviert werden. In diesem Fall wird auf allen anderen Modulen ein Reset ausgelöst, solange der Kanal aktiv ist. Theoretisch kann der Zustand dieses Kanales auch gelesen werden. Praktisch bereitet dies dem Transputer jedoch gewisse Probleme, wenn er gerade einen Reset übergebraten bekommt.

SYNC0/1: Diese beiden Bits dienen zur Synchronisationssteuerung des TIP-Buses. Da die Zeitscheiben auf allen Modulen synchron laufen müssen, jedoch lokal von Oszillatoren getaktet werden, müssen diese ab und zu synchronisiert werden. Dies wird durch einfügen eines Synchronisationsslots erreicht, der per Konvention immer die Kanalnummer 0 trägt. Am Ende eines solchen Slots kann die Übertragung des TIP-Buses angehalten werden. Es gibt dann 2 Möglichkeiten die Übertragung fortzufahren:

- durch Aktivierung der TIP-Bus-Leitung CLKTK von einem Slave,
- oder durch Deaktivierung von RTSC0 durch einen Slave.

Mit den beiden Bits kann nun ein Master festlegen, wie er sich von einem Slave synchronisieren lassen will, bzw. ein Slave kann damit die CLKTK-Leitung aktivieren:

| SYNC1 | SYNC0 | |
|---|---|---|
| 0 | 0 | Hiermit legt der Master fest, daß die Übertragung am Ende eines Synchronisationsslotes nicht angehalten wird. Für einen Slave hat diese Bitkombination keine Bedeutung. |
| 0 | 1 | Hiermit legt der Master fest, daß er die Bus-Übertragung nach der Synchronisation wieder startet, wenn RTSC0 nicht aktiv ist. Für einen Slave hat diese Bitkombination keine Bedeutung. |
| 1 | 0 | Hiermit legt der Master fest, daß er die Bus-Übertragung nach der Synchronisation mit CLKTK wieder startet. Für einen Slave hat diese Bitkombination keine Bedeutung. |
| 1 | 1 | Mit dieser Bitkombination aktiviert ein Slave die CLKTK-Leitung des TIP-Buses. |

## 4.7. Das Event Register ($0000 01C0)

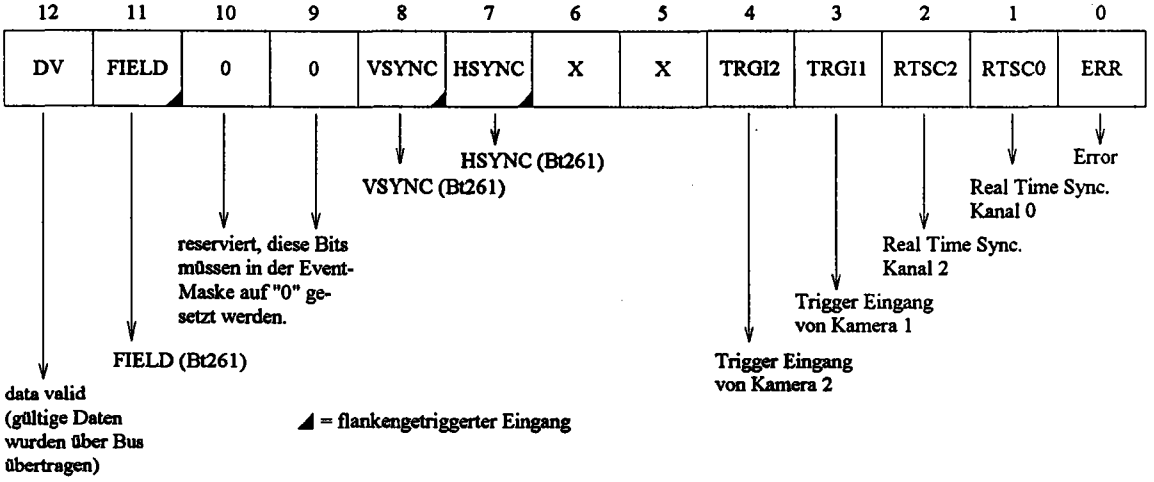



*Bild 17: Bit-Belegung des Event Registers*

Das Event Register dient zur Steuerung der Event Logik. Es gibt 9 verschiedene Eventquellen. Welche dieser Quellen tatsächlich einen Event im Transputer auslösen darf, kann der Anwender bestimmen. Dazu muß er eine Maske in das Event Register schreiben. Ist in dieser Maske ein Bit auf "1" gesetzt, so kann die zugehörige Event - Quelle einen Event im Transputer auslösen. Andererseits wird der Event von der Event-Logik unterdrückt. In dieser Maske müssen die Bits 9 und 10 immer auf "0" gesetzt werden. Die Event-Maske kann nicht zurückgelesen werden. Ist nun ein Event ausgelöst worden, so muß der Event-Handler feststellen, von welcher Quelle dieser herrührt. Dies geschieht, indem er das Event Register liest. Das Bit der Event-Quelle wird dabei mit einer "1" markiert. Durch erneutes Schreiben der Event-Maske wird dieses Bit wieder

zurückgesetzt. Bei den Event-Quellen muß unterschieden werden, ob diese die Event-Logik statisch triggern, oder ob der Event-Eingang flankengetriggert ist. Aus Platzgründen konnten nur drei Eventquellen flankengetriggert realisiert werden. Diese sind im obigen Bild durch ein schwarzes Eck gekennzeichnet. Die statisch getriggerten Event-Quellen haben den Nachteil, daß der Event-Pin des Transputers erneut aktiviert wird, nachdem der Event-Handler die Maske in das Event Register zurückgeschrieben hat, wenn die Quelle noch aktiv ist. Bei flankengetriggerten Event-Quellen dagegen, unterdrückt die Event-Logik eine erneute Aktivierung des Transputer-Event-Pins solange, bis die Event-Quelle deaktiviert ist. Zwischen folgenden Event-Quellen kann im Event Register unterschieden werden:

ERR: Ein Fehler ist auf dem Modul aufgetreten. Über die Art des Fehlers gibt das Status Register Auskunft.

RTSC0: Der Real Time Sync. Kanal 0 wurde aktiviert, d. h. die Bus-Übertragung wird am Ende eines Synchronisations-Slots angehalten.

RTSC2: Der Real Time Sync. Kanal 2 wurde aktiviert, d. h. irgendwo im Transputersystem ist auf einem Modul ein Fehler aufgetreten.

TRGI1: Der Triggereingang des Kameraanschlusses 1 wurde aktiviert.

TRGI2: Der Triggereingang des Kameraanschlusses 2 wurde aktiviert.

HSYNC: Bei der Digitalisierung wurde das Ende einer Bildzeile erreicht.

VSYNC: Bei der Digitalisierung wurde das Ende eines (Halb-) Bildes erreicht.

FIELD: FIELD kennzeichnet das Halbbild, welches gerade digitalisiert wird. FIELD = "1" bedeutet hierbei das ungerade Halbbild, d. h. wenn ein neues Vollbild begonnen wird, kann ein Event ausgelöst werden.

DV: Über den TIP-Bus wurde ein Block mit gültigen Daten übertragen (siehe Tipset Dokumentation).

### 4.8. Das Analyse Register ($0000 0200)

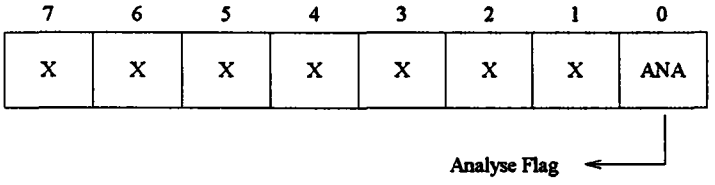



*Bild 18: Bit-Belegung des Analyse Registers*

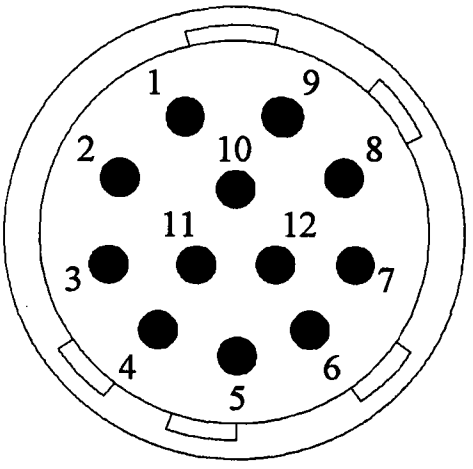
Vom Analyse Register wird nur das niederwertigste Bit genutzt. Wird in dieses Bit eine "1" geschrieben, so wird beim nächsten Auftreten eines Transputer-Fehlers das Analyse-

Signal des Transputers aktiviert. Wird danach ein Link Reset von außen ausgelöst, um den Transputer zu debuggen, so wird dieses Bit automatisch wieder gelöscht.

# 5. Das Videointerface

## 5.1. Die Kameranschlüsse

Der Framegrabber besitzt zwei Kameraanschlüsse, zwischen diesen beiden Anschlüssen kann per Software umgeschaltet werden. Neben den eigentlichen Bildsignalen, liegen auf den Kameraanschlüssen noch die Signale HSYNC, VSYNC, Trigger und der Pixeltakt. Außerdem sind noch Pins zur Spannungsversorgung, wie +12V, -12V, AGND (Bezugspotential für analoge Signale) und DGND (Bezugspotential für digitale Signale) vorhanden. Bild 19 zeigt die Signalbelegung des Kamerasteckers:




1 - 12V
2 + 12V
3 + 12V
4 AGND
5 Green Video in
6 Red Video in
7 DGND
8 VSYNC
9 PCLK
10 HSYNC
11 Blue Video in
12 Trigger

*Bild 19: Signalbelegung des Kamerasteckers (Sicht auf Frontblende)*

Mittels der Synchronisationssignale werden Kamera und Framegrabber synchronisiert. Dabei gibt es zwei verschiedene Möglichkeiten, wie das geschehen kann. So kann die Kamera die Synchronisationssignale liefern und der Framegrabber richtet die Abtastung des Videosignals daran aus. Andererseits gibt es jedoch auch Kameras, die die Synchronisationssignale vorgegeben haben wollen, um darauf ihr Videosignal auszurichten. Das bedeutet aber, daß die Leitungen mit den Synchronisationssignalen einmal als Eingang, und das andere Mal als Ausgang arbeiten. Dies macht das Kamerainterface natürlich aufwendig. Allerdings ist durch diesen Aufwand das TIP-CFG auch in der Lage jede erdenkliche Videoquelle zu bedienen. Bild 20 zeigt das Blockschaltbild der Kameraschnittstelle.

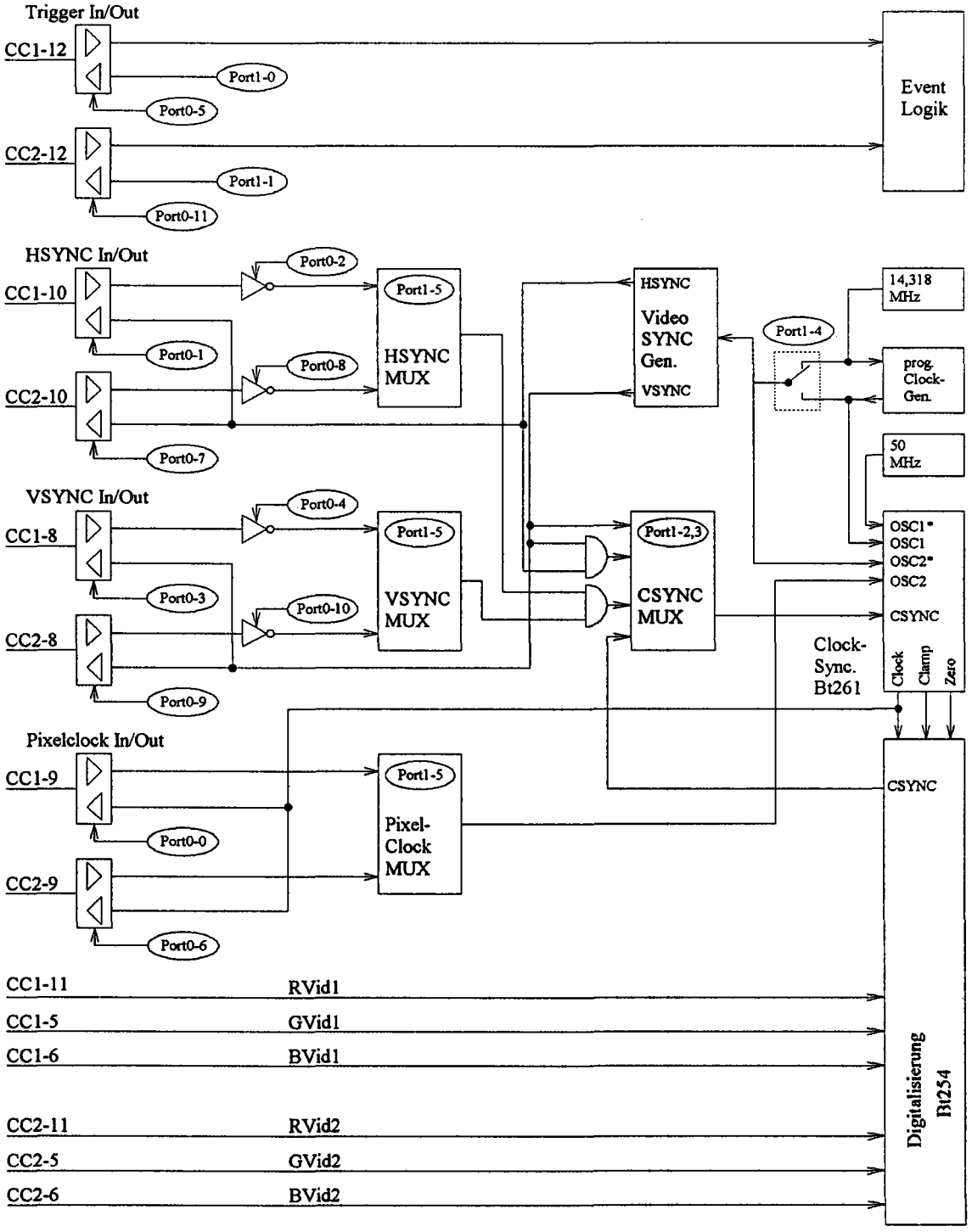



*Bild 20: Kameraschnittstelle des TIP-CFG*

Die ganze Steuerung der Kameraschnittstelle geschieht mittels zweier Register, die Port0 und Port1 genannt werden. Die Ellipsen, im obigen Bild, sollen die Ausgänge dieser Register andeuten. Die Zahl hinter dem Registernamen spezifiziert dabei die Bitstelle im Register. Alle Signalleitungen, außer den Videoleitungen, können als Eingänge oder als Ausgänge geschaltet werden. Nachdem die Versorgungsspannung für den Framegrabber eingeschaltet wurde, werden die Register automatisch so gesetzt, daß alle Anschlußlei-

tungen sich als Eingänge verhalten. Dies gewährleistet, daß in diesem Moment keine Schäden an Kamera, bzw. Framegrabber auftreten können.

Die Triggereingänge werden direkt zur Event-Logik geleitet. Die Möglichkeit, ein Trigger-Signal auszuwerten, ist gerade für die industrielle Bildverarbeitung wichtig. Dieses Signal wird dort z. B. von einer Lichtschranke an einem Fließband aktiviert, wenn ein Objekt die Schranke unterbricht. Der Framegrabber nimmt dann ein Bild von dem Objekt auf, daß dann über den TIP-Bus zur Weiterverarbeitung an andere Rechnerkomponenten geschickt wird. Die Triggerausgänge werden direkt von zwei Port-Bits beeinflußt.

Zwischen den beiden HSYNC-, VSYNC- oder Pixel-Clock-Eingängen der Kameraanschlüsse, wird mit einem Multiplexer gewählt. Die HSYNC- und VSYNC-Eingänge können zusätzlich noch invertiert werden, wenn dies notwendig ist. Hinter den Invertern müssen diese Signale allerdings aktiv-low sein. Die beiden Signale werden zu einem gemeinsamen CSYNC-Signal verknüpft und auf den Eingang eines weiteren Multiplexers gegeben. Werden die HSYNC- und VSYNC-Leitungen als Ausgang benutzt, so werden diese Signale vom Video-Synchronisationssignal-Generator erzeugt. Realisiert wird dieser mit dem integrierten Baustein LM1882. Die Synchronisationssignale werden dort mittels Zähler von einem Eingangstakt abgeleitet. Die so erzeugten HSYNC- und VSYNC-Signale werden ebenfalls verknüpft und auf den CSYNC-Multiplexer gegeben. Ein weiteres CSYNC-Signal kommt von der Digitalisierung. Oft sind die Synchronisationssignale auf das Videosignal aufmoduliert. In diesem Fall werden diese im Digitalisierungsblock extrahiert.

Das CSYNC-Signal, das über den Multiplexer ausgewählt wird, gelangt zur Takt-Synchronisierung. Dort wird der Arbeitstakt für die A/D-Wandler aus mehreren möglichen Eingangstakten erzeugt. Dabei wird der erzeugte Takt auf die fallende Flanke des CSYNC-Signales synchronisiert. Hierzu wird der integrierte Baustein Bt261 von Brooktree verwendet. Als Eingangstakte stehen vier verschiedene Quellen zur Verfügung. Vom Kamerainterface kann ein Pixel-Takt gewählt werden, falls dieser von der Kamera zur Verfügung gestellt wird. Außerdem stehen zwei Quarzoszillatoren, der eine mit 50 MHz, der andere mit 14,318 MHz zur Disposition. Der 14,318 MHz-Oszillator wurde deshalb hinzugenommen, weil diese Frequenz oft als Pixeltakt für Standard-Videoanwendungen verwendet wird. Als vierte Quelle kann ein programmierbarer Taktgenerator gewählt werden. Dieser Generator erzeugt aus dem 14,318 MHz Signal mittels einer PLL 32 verschiedene Taktfrequenzen, bis hin zu 80 MHz. Der Eingangstakt wird innerhalb des Bt261 auf die gewünschte Pixeltakt-Frequenz heruntergeteilt, wobei der Pixeltakt bei jeder fallenden Flanke von CSYNC resynchronisiert wird. Damit hierbei der Pixeljitter nicht zu groß wird, sollte eine möglichst hohe Eingangsfrequenz gewählt werden. Der so erzeugte Pixeltakt kann auch an die Kamera ausgegeben werden.

## 5.2. Die Port Register

Das Port Register 0 dient dazu, die Richtung des Signalflusses auf den HSYNC-, VSYNC-, Trigger- und Pixel Clock-Leitungen der Kameraschnittstelle festzulegen. Außerdem können mittels dieses Registers die Eingangssignale HSYNC und VSYNC invertiert werden (siehe hierzu auch Bild 20 auf Seite 21).

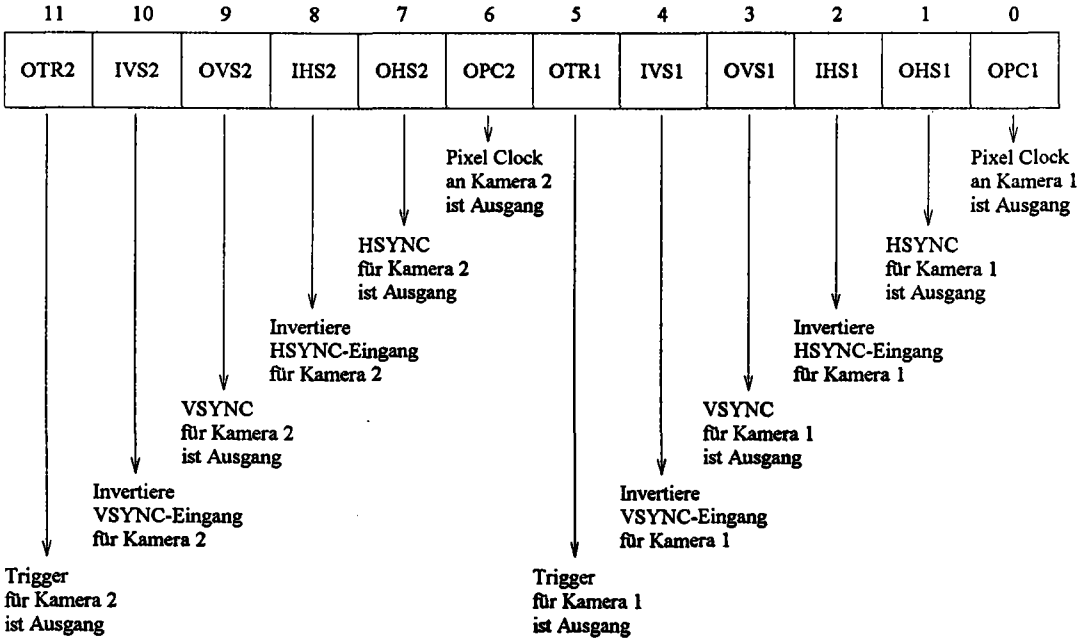



*Bild 21: Bit-Belegung des Port Registers 0* [1)]

Mit dem Port Register 1 werden die in Bild 20 gezeigten Multiplexer und Schalter gesteuert. Außerdem befinden sich hier zwei Bits mit denen die Trigger-Ausgänge der Kameraanschlüsse aktiviert werden können.

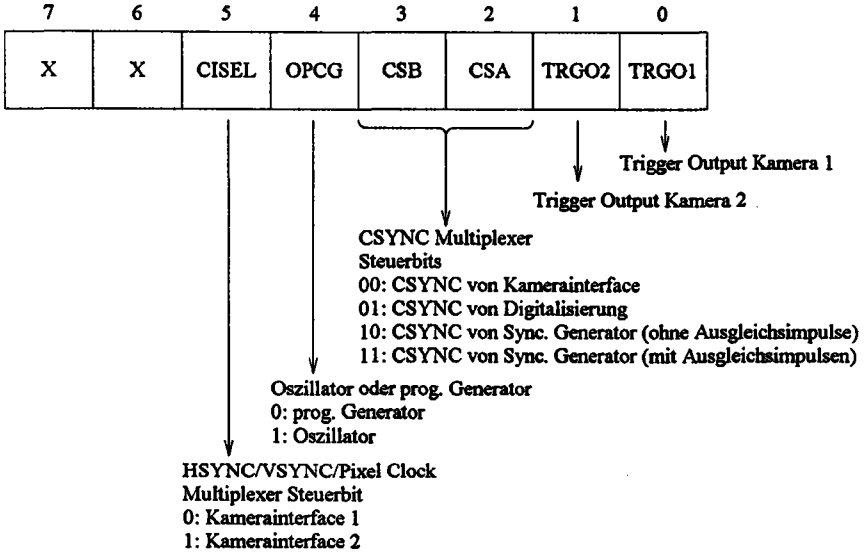



*Bild 22: Bit-Belegung des Port Registers 1*

Im Port Register 2 werden die Informationen festgehalten, die der DMA-Controller des Videointerfaces für die Arbeit benötigt. Mit dem MODE-Bit wird unterschieden, ob die

---

[1)] Durch einen Fehler im PCB-Design auf allen TIP-CFG der Revisionsnummer 1.1 läßt sich Bit 11 im Port Register 0 nicht setzen. Dadurch ist für den Kamerastecker 2 die Triggerleitung nur als Eingang zu konfigurieren.

Pixel im konsekutiven Modus oder im zeilenorientierten Modus in das VRAM geschrieben werden. Wird das VDDEN-Bit auf "0" gesetzt, so wird der Datenstrom ins VRAM unterbrochen. Dieses Bit dient also dazu, auf einfache Weise die Digitalisierung zu stoppen. Die A/D-Wandler arbeiten dabei jedoch weiter, die generierten Daten werden nur nicht mehr gespeichert. Mit dem RVI-Bit wird der DMA-Controller des Videointerfaces in einen definierten Zustand gebracht. Dazu muß dieses Bit auf "1" gesetzt werden. Dies sollte immer gemacht werden, nachdem der Framegrabber frisch an Spannung gelegt wurde. Für den eigentlichen Betrieb muß dieses Bit zu "0" gesetzt werden. Das VMEM-Bit kann nur gelesen werden. Es gibt Auskunft darüber, wieviel Videospeicher vorhanden ist.

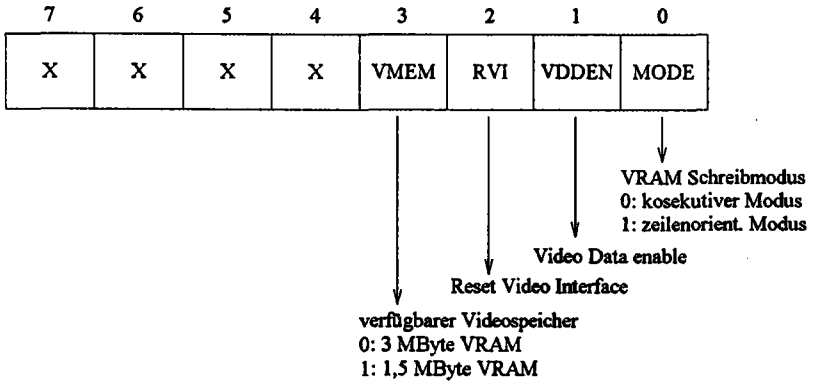



*Bild 23: Bit-Belegung des Port Registers 2*

## 5.3. Das Image Start Register

Mit dem Image Start Register wird festgelegt, ab welcher Speicherzeile ein digitalisiertes Bild abgelegt wird. Da der Videospeicher immer aus 1.024 Zeilen aufgebaut ist, würde hier normalerweise ein 10-Bit-Register benötigt. Da solche Register aber üblicherweise eine Breite von 8 oder 9 Bit haben, wären dafür zwei Bauelemente nötig gewesen. Aus Platzgründen wurde deshalb zu Gunsten eines 8-Bit-Registers entschieden. Dadurch kann für den Bildanfang nur noch jede vierte Speicherzeile festgelegt werden. In dieses Register ist deshalb die gewünschte Bildzeile, dividiert durch 4 zu programmieren.

## 5.4. Die Digitalisierung

Kernstück des Videointerfaces ist die Digitalisierung, die in Bild 24 im Blockschaltbild dargestellt ist. Mittels eines Analog-Multiplexers werden die Videosignale von den beiden Kameraanschlüssen ausgewählt. Außerdem wird mit einem weiteren Multiplexer ein Analogsignal ausgewählt, aus dem Synchronisationssignale extrahiert werden können. Hierfür kann eine Schwelle angegeben werden die angibt, wieweit der Signalpegel des Videosignals unter dem Schwarzpegel liegen muß, damit ein Synchronisationssignal erkannt wird. Das erkannte Synchronisatiossignal wird in einen TTL-Pegel umgewandelt und zum oben erwähnten CSYNC-Multiplexer geleitet.

Bevor die Videosignale digitalisiert werden, können sie noch mit einem Tiefpaß in ihrer Bandbreite begrenzt werden. Die Tiefpässe haben eine Grenzfrequenz von etwa 8 MHz und sind eigentlich nur für den Extremfall gedacht, wenn die Videosignale durch hochfrequente Anteile, z. B. von der Pixel-Clock verschmutzt sind. Da ein Videosignal in der Regel nur eine Bandbreite von 5 MHz besitzt, sind die Tiefpässe bei Abtastraten von typischerweise 15 MHz normalerweise nicht nötig. Daher sind die Tiefpässe auch nicht durch Software einkoppelbar. Um die Tiefpässe zu benutzen müssen einige Bauteile auf dem Framegrabber umgelötet werden.

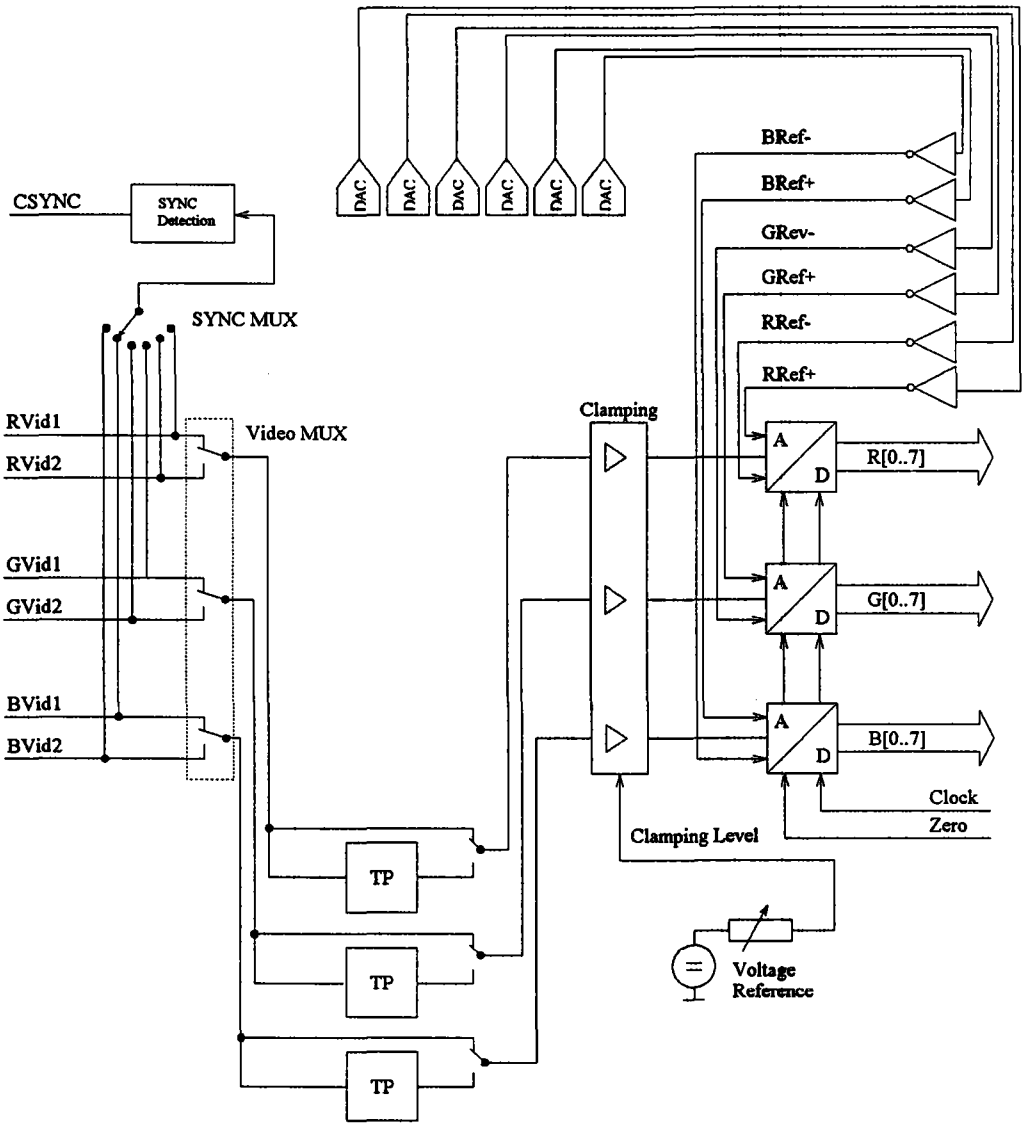



*Bild 24: Blockschaltbild der Digitalisierung*

Für die Ankopplung der Videosignale gibt es theoretisch zwei Möglichkeiten. Einmal können die Signale gleichstrommäßig (DC-Kopplung) angekoppelt werden. Dabei fängt man sich in den verschiedenen Verstärkerstufen der Bauelemente aber einen Gleichspannungs-Offset ein, der nicht in den Griff zu bekommen ist. Deshalb werden bei Videoanwendungen die verschiedenen Signalstufen über Kondensatoren gekoppelt; diese Vorgehensweise wird AC-Kopplung genannt. Die Koppelkondensatoren führen aber dazu, daß

der Gleichspannungspegel eines Videosignals abgeschnitten wird. Dadurch schwimmt hinter dem Koppelkondensator der Bezugspegel des Videosignals, d. h. vor der eigentlichen Digitalisierung muß der Gleichspannungspegel wiederhergestellt werden. Dies wird in der Clamping-Stufe erreicht.

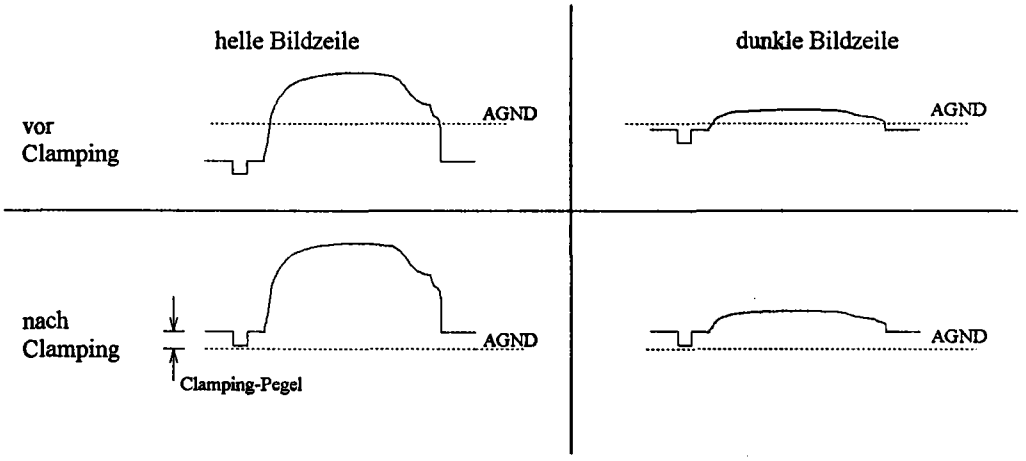



*Bild 25: Wiederherstellung des Gleichspannungspegels eines Videosignals durch Clamping*

In Bild 25 wird der Einfluß der Clampingstufe auf das Videosignal dargestellt. Hierzu wird in der linken Bildhälfte das Videosignal einer überwiegend sehr hellen Bildzeile, dem Videosignal einer sehr dunklen Bildzeile, in der rechten Bildhälfte gegenübergestellt. Vor dem Clamping haben die Videosignale einen Bezugspegel, der ihrem mittleren Gleichspannungswert entspricht. Dadurch wird natürlich der Schwarzpegel der hellen Bildzeile, bezüglich des Bezugspegels AGND sehr viel niedriger, als der dementsprechende Schwarzpegel der dunklen Bildzeile. Deshalb wird in der Clamping-Stufe der Pegel des Videosignals auf den Clamping-Pegel justiert. Dazu wird der Clamping-Stufe über den Clamp-Anschluß der Zeitpunkt übermittelt, wann dies geschehen soll. In der Regel wird hierfür ein Zeitpunkt innerhalb der front porch, bzw. der back porch gewählt. Der Clamping-Pegel entsprich in diesem Fall dem Schwarzpegel des Videosignales.

Nachdem der Gleichspannungspegel der Videosignale justiert wurde, können diese quantisiert werden. Die hierfür zur Verfügung stehenden A/D-Wandler weisen eine lineare Quantisierungskennlinie auf, jeder Analogwert wird in einen 8-Bit-Digitalwert gewandelt. Jeder A/D-Wandler bekommt eine untere und eine obere Referenzspannung zugewiesen. Ein Analogsignal, daß kleiner oder gleich dem unteren Referenzwert ist, wird in den Wert $00 gewandelt. Ist das Eingangssignal größer oder gleich dem oberen Referenzwert, so wird dafür der Wert $FF ausgegeben. Zwischen diesen beiden Pegeln werden die digitalen Werte linear aufgeteilt. Die Referenzwerte können per Software eingestellt werden. Hierfür stehen sechs D/A-Wandler zur Verfügung, die einen, zum digitalen Wert, proportionalen Strom liefern. Über die Operationsverstärker wird der "Stromwert" in eine Spannung umgewandelt.

Dadurch, daß alle sechs Referenzen unabhängig voneinander eingestellt werden können, läßt sich hiermit ein einfacher Weißabgleich des Framegrabbers realisieren. Dazu muß nur ein Bild aufgenommen werden, das dem gewünschten Referenzweiß entspricht. Anschließend werden die Referenzen solange geändert, bis man drei gleichgroße Farbwerte erhält. Die unteren Referenzen werden hierzu alle fest auf den Schwarzpegel eingestellt.

Ist nun zum Beispiel der Farbwert für Blau zu hoch, d. h. das Bild ist blaustichig, so gibt es zwei Möglichkeiten dies über die Referenzen zu beheben. Einmal kann der obere Referenzwert für Blau erhöht werden, damit sinkt der digitale Farbwert bei gleichbleibendem Videosignal. Andererseits können auch die oberen Referenzen für Rot und Grün gesenkt werden, dadurch werden die entsprechenden Farbwerte verstärkt.

Neben dem Takt-Signal benötigen die in Bild 24 gezeigten A/D-Wandler noch ein Zero-Signal. Wird dieses Zero-Signal aktiviert, so werden die internen Komparatoren der Wandler neu abgeglichen. Dies sollte in jeder horizontalen Austastlücke geschehen, damit die Wandler die angegebene Linearität von max ±1 LSB einhalten.

Für die Digitalisierung wird der hochintegrierte Baustein Bt254 von Brooktree verwendet. Dieser enthält die, in Bild 24 gezeigten, A/D-Wandler, Multiplexer und D/A-Wandler zur Referenzerzeugung. Während der Initialisierungsphase des Framegrabbers müssen 7 Register dieses Bauelementes programmiert werden.

Mittels des Command Registers werden die Multiplexer, das Datenformat der digitalen Ausgänge und die Logik zur Extrahierung der Synchronisationssignale gesteuert.

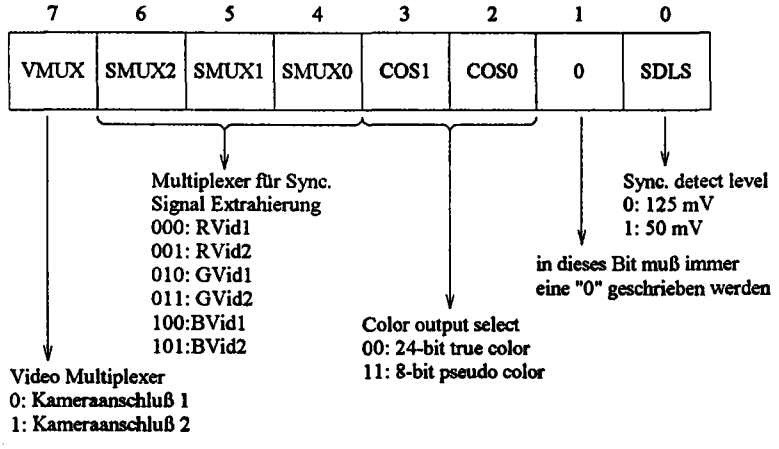



*Bild 26: Bit-Belegung des Command Registers der Digitalisierungslogik*

Mit dem SDLS-Bit läßt sich festlegen, wie weit der Pegel des Videosignals unterhalb des Schwarzpegels gefallen sein muß, damit dieser als Synchronisationsimpuls erkannt wird. Die ersten Tests des Framegrabbers haben hier gezeigt, daß der 50 mV-Wert nicht ausreicht. Einschwingvorgänge, nach der fallenden Flanke des Synchronisationsimpulses, führen hier zu Fehlinterpretationen der Extraktionslogik. In dieses Bit sollte deshalb immer eine "0" programmiert werden. Die Synchronisationssignale dürfen jedem der 6 Videosignale überlagert sein. Die Auswahl wird hierbei mit den Bits SMUX0-SMUX2 getroffen. Mit dem VMUX-Bit wird der Kameraanschluß ausgewählt, dessen Videosignale digitalisiert werden sollen. Mit den COS0- und COS1-Bits wird das Datenformat der digitalen Ausgänge festgelegt. Von den vier verschiedenen Möglichkeiten, die der Baustein hierbei bietet, sind allerdings nur zwei sinnvoll für den Framegrabber. Dies ist zum einen der 24-Bit True-Color-Modus, bei dem jeder Farbwert mit 8 Bit codiert wird. Mit dem 8-Bit-Pseudo-Color-Modus läßt sich auch das Videosignal einer Schwarz/weiß-Kamera sinnvoll bearbeiten. In diesem Mode wird nur der Grün-Kanal digitalisiert, die Daten werden aber gleichzeitig auf R[0..7], G[0..7] und B[0..7] ausgegeben. Dadurch wird das so erzeugte Bild auf einer True-Color-Grafikkarte mit den richtigen Grauwerten

wiedergegeben. Allerdings wird hierbei natürlich eine enorme Verschwendung an Speicherplatz betrieben.

Die restlichen sechs Register des Bt254 dienen zur Einstellung der oberen, bzw. unteren Referenzspannungen für A/D-Wandler. Die Referenzspannungen berechnen sich dabei nach folgender Formel:

$$U_{REF} = < REF\_REG > \cdot 4,7mV$$

Damit lassen sich die Referenzspannungen zwischen 0 V und 1,2 V, in Schritten von 4,7 mV, programmieren. Um den Framegrabber an einer einfachen 5 V-Spannungsversorgung betreiben zu können, werden die Operationsverstärker nur mit +5 V versorgt. Dadurch schaffen die Verstärkerausgänge es leider nicht, die Referenzspannung wirklich bis auf 0 V herunter zu treiben. Die untere Grenze für die gewählten Operationsverstärker liegt bei etwa 280 mV. Da der Clamping-Pegel aber zwischen 0 V und 1,25 V eingestellt werden kann, stellt dies keinen Nachteil dar. Mittels des Clamping-Pegels wird der Schwarzwert des Videosignals einfach auf 300 mV justiert, damit machen Referenzwerte unter 300 mV sowieso keinen Sinn mehr. **Der Clamping-Pegel wird vor der Auslieferung des TIP-CFG auf 300 mV eingestellt.**

## 5.5. Die Taktsynchronisierung

Die Taktsynchronisierung dient dazu, den Arbeitstakt für die A/D-Wandler mit dem Videosignal zu synchronisieren. Dazu wird der hochintegrierte Baustein Bt261 von Brooktree verwendet. Außerdem erzeugt der Bt261 noch einige Signale zur Steuerung des DMA-Controllers. Der Baustein bietet stolze 30 Register zur Programmierung an. Vom Transputer wird auf diese Register zugegriffen, indem zuerst das Address Register des Bt261 mit der gewünschten Registeradresse geladen wird. Anschließend kann über den Daten Port des Bausteines das spezifizierte Register gelesen oder geschrieben werden. Nach jedem Zugriff auf den Daten Port wird das Address Register automatisch inkrementiert. Dadurch können bei der Initialisierung des Bausteins alle Register der Reihe nach beschrieben werden, ohne immer erneut auf das Address Register zugreifen zu müssen. Die folgende Tabelle gibt einen Überblick über die verschiedenen Register des Bt261:

| Register-Nummer | Register-Name |
|---|---|
| 0 | Command Register 0 |
| 1 | Command Register 1 |
| 2 | Command Register 2 |
| 3 | Command Register 3 |
| 4 | VSYNC Sample Register |
| 5 | OSC Count Low Register |
| 6 | OSC Count High Register |
| 7 | Status Register (read only!) |
| 8, 9 | HSYNC Start Register low, high |

| | |
|---|---|
| 10, 11 | HSYNC Stop Register low, high |
| 12, 13 | CLAMP Start Register low, high |
| 14, 15 | CLAMP Stop Register low, high |
| 16, 17 | ZERO Start Register low, high |
| 18, 19 | ZERO Stop Register low, high |
| 20, 21 | FIELD Gate Start Register low, high |
| 22, 23 | FIELD Gate Stop Register low, high |
| 24, 25 | NOISE Gate Start Register low, high |
| 26, 27 | NOISE Gate Stop Register low, high |
| 28, 29 | HCOUNT Register |

Die Taktsynchronisierung kann mit dem Bt261 auf zwei verschiedene Weisen realisiert werden. Einmal kann mit einem externen VCO und mit dem sich auf dem Chip befindenden Phasen-Komparator ein Phasenregelkreis realisiert werden, der die Pixel-Clock-Phase auf die fallende Flanke von CSYNC synchronisiert. Alternativ dazu kann ein von außen angelegter Takt über Zähler heruntergeteilt werden und zu einem Pixel-Clock-Signal verarbeitet werden. Dieses Signal wird jeweils am Anfang einer Zeile, auf die fallende Flanke von CSYNC synchronisiert. Da der Eingangstakt aber, im Gegensatz zu der PLL-Lösung, fest ist, erhält man hier einen Jitter in der Pixel-Clock. Dieser ist um so geringer, je höher die Frequenz des Eingangstaktes ist, aus der die Pixel-Clock abgeleitet wird. Da PLLs aber in Theorie und Praxis recht kompliziert sind, wurde hier die zweite Alternative verwendet.

Bei der fallenden Flanke von CSYNC wird jeweils ein Zähler gestartet, der mit jedem Pixel-Takt inkrementiert wird und bis zum Inhalt des HCOUNT Registers zählt. In diesem wird die Länge einer Bildzeile in Pixel-Clock-Takten festgeschrieben. Die Lage der Steuersignale, die der Bt261 generiert, wird dann immer durch den Start- und den Stopzählerstand beschrieben, zwischen denen das Signal aktiv sein soll. Da die zugehörigen Register in der Regel 12 Bit umfassen, der Bt261 aber nur ein 8-Bit-Prozessorinterface aufweist, gibt es immer ein Register für das Low- und für das High-Byte. Werden neue Werte in solche Register geschrieben, so muß dies immer in der Reihenfolge Low-Byte, High-Byte geschehen, weil der Bt261 die Werte erst mit dem High-Byte in die internen Register übernimmt.

In den 30 Registern des Bt261 müssen natürlich auch viele Angaben gemacht werden, die sich auf den nichtverwendeten PLL-Betrieb beziehen. Bei der folgenden Registerbeschreibung werden diese Angaben nicht erläutert, da dies nur eine unnötige Belastung für den Leser wäre. Stattdessen werden kommentarlos die Werte genannt, die an diesen Stellen für den Framegrabber eingetragen werden müssen.

**Command Register 0:**

Mit diesem Register kann der Eingangstakt gewählt werden, aus dem die Pixel-Clock erzeugt wird (siehe auch Bild 20 auf Seite 21). Das CPTR-Bit in diesem Register dient zur Steuerung der Capture-Leitung des Bausteins. Die Digitalisierung arbeitet nur dann, wenn diese Leitung aktiv ist. Das Capture-Bit wird dazu verwendet, ein einzelnes Bild zu digitalisieren. Dazu wird im Interlaced-Betrieb das Capture-Bit vor der steigenden Flanke von Field gesetzt. Wird ein neues ungerades Halbbild von der Kamera

ausgegeben, so wechselt die Field-Leitung von "0" auf "1". Wurde vorher das Capture-Bit gesetzt, so wird in diesem Moment auch die Capture-Leitung aktiviert, d. h. die Digitalisierung beginnt. Wird daraufhin das Capture-Bit vom Transputer zu "0" gesetzt, so bleibt die Capture-Leitung dennoch bis zur nächsten steigenden Flanke von Field gesetzt. Damit wird aber genau ein ungerades und ein gerades Halbbild, also ein Vollbild digitalisiert. Soll eine ständige Digitalisierung durchgeführt werden, so darf das Capture-Bit nicht zurückgesetzt werden. Die steigende Flanke der Field-Leitung bekommt der Transputer mit, indem durch Field ein Event ausgelöst werden kann. Im noninterlaced Modus wird die Capture-Leitung mit VSYNC synchronisiert. Der obig geschilderte Vorgang funktioniert dann genauso, mit dem einzigen Unterschied, daß statt des Field-Signales das VSYNC-Signal bewertet wird.

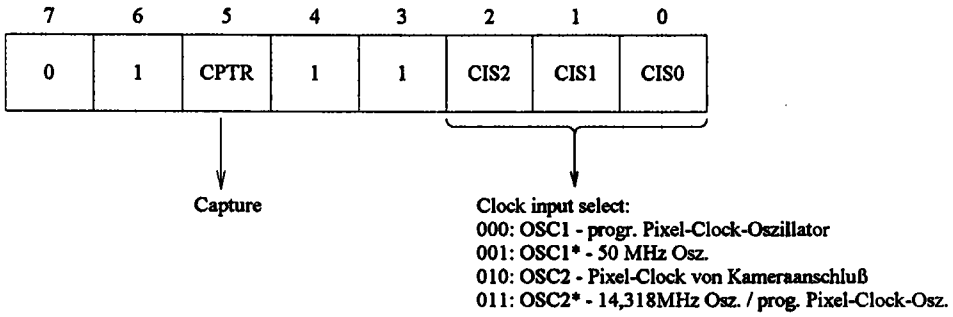



*Bild 27: Bit-Belegung des Command Registers 0 vom Bt261*

## Command Register 1

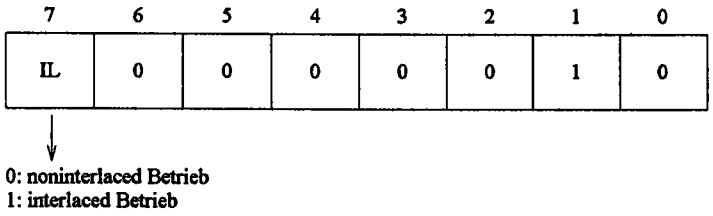



*Bild 28: Bit-Belegung des Command Registers 1 vom Bt261*

In diesem Register ist nur das IL-Bit von Interesse, mit welchem unterschieden wird, ob das Bild im interlaced Betrieb von der Kamera geliefert wird oder nicht. Der Bt261 benötigt diese Angabe für die oben erläuterte Behandlung der Capture-Leitung.

**Command Register 2**

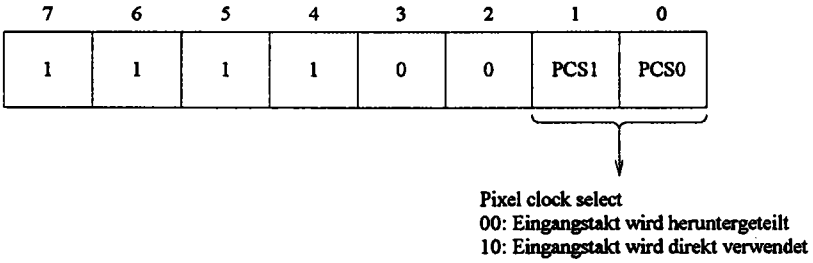



*Bild 29: Bit-Belegung des Command Registers 2 vom Bt261*

Hier wird mit den Bits PCS1 und PCS0 festgelegt, ob der mit Command Register 0 gewählte Eingangstakt, zur Erzeugung der Pixel-Clock, heruntergeteilt werden soll oder nicht. Wird der Pixel-Takt von der Kamera, bzw. der 14,318 MHz-Oszillator verwendet, so kann dieser Takt natürlich nicht mehr heruntergeteilt werden. In diesem Fall wird der Inhalt des OSC Count Low - und des OSC Count High Registers vom Bt261 ignoriert. Es sei hier nocheinmal daraufhingewiesen, daß aus Gründen des auftretenden Pixel Jitters möglichst eine hoher Eigangstakt gewählt werden sollte, der heruntergeteilt wird. Die Option den Eingangstakt direkt zu verwenden sollte deshalb möglichst nicht verwendet werden.

**Command Register 3**

Dieses Register enthält nur Angaben, die für den PLL-Betrieb von Bedeutung sind. Bei der Initialisierung sollte in dieses Register immer der Wert 0 geschrieben werden.

**VSYNC Sample Register**

Dieses Register wird dazu verwendet, um aus dem CSYNC-Signal ein VSYNC-Signal herauszufiltern. Dazu wird das CSYNC-Signal zu einem bestimmten Zeitpunkt, bei dem kein HSYNC vorhanden sein darf, auf logisch "0" untersucht. In diesem Fall wird die VSYNC-Leitung für die ganze Zeile, bis zum nächsten Sample-Zeitpunkt aktiviert. Mit diesem Register wird die Lage dieses Untersuchungszeitpunktes festgelegt, indem hier die Anzahl der Pixel-Clock-Takte nach der fallenden Flanke von CSYNC eingetragen wird, bei der dies geschehen soll. Der Wert muß natürlich größer sein als die maximale Länge eines HSYNC-Impulses, sonst könnte nicht zwischen HSYNC und VSYNC unterschieden werden. Brooktree empfiehlt hier einen Wert von 1/4 HCOUNT.

**OSC Count Low - und OSC Count High Register**

Mit diesen Registern wird festgelegt, wie der Eingangstakt heruntergeteilt wird, um das Pixel-Clock-Signal zu erzeugen. Dabei wird für die Low-Phase und für die High-Phase der Pixel-Clock getrennt festgelegt, wieviele Taktimpulse des Eingangstaktes diese umfassen sollen. Dabei ist zu beachten, daß hierbei sowohl die steigenden Flanken, als auch die fallenden Flanken des Eingangstaktes berücksichtigt werden. In Bild 30 wird dies an einem Beispiel verdeutlicht:

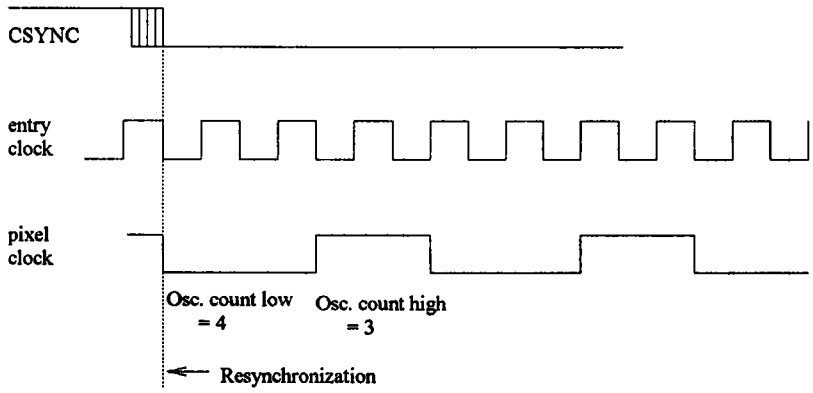



*Bild 30: Erzeugung und Synchronisierung des Pixel-Taktes*

Hier wurde für Osc. Count Low der Wert 4 und für Osc. Count High der Wert 3 gewählt. Dadurch umfaßt der Low-Pegel der erzeugten Pixel-Clock 4/2 Periodendauern des Eingangstaktes und der High-Pegel 3/2 Periodendauern. Der Eingangstakt wird also durch den Wert 3,5 geteilt. Die fallende Flanke der Pixel-Clock wird mit der fallenden Flanke von CSYNC synchronisiert. Dabei entsteht, wie in obigem Bild angedeutet, ein maximaler Fehler von der halben Periodendauer des Eingangstaktes. Dieser Jitter ist daher um so geringer, je höher die Frequenz des Eingangstaktes ist. Wird der Eingangstakt direkt verwendet, also nicht heruntergeteilt, so ist dieser Jitter an senkrechten Bildkanten deutlich zu beobachten. Diese Kanten sind dann nicht mehr gerade, sondern verlaufen etwas zick-zack-förmig. In die beiden Register können Werte zwischen 2 und 15 eingetragen werden. Zu beachten ist hierbei, daß jede Phase der erzeugten Pixel-Clock eine Mindestdauer von 20 ns aufweisen muß. Andernfalls arbeiten die A/D-Wandler nicht mehr korrekt.

**Status Register**

Der Inhalt des Status Registers ist nur für den PLL-Betrieb von Interesse. In der Initialisierungsphase kann hier ein beliebiger Wert eingeschrieben werden, um den Adresszähler des Bt261 zu inkrementieren.

**HSYNC Start- und HSYNC Stop Register**

Mit diesen beiden 16-Bit Registern wird die Lage des HSYNC-Impulses festgelegt, der aus dem CSYNC-Signal extrahiert wird. Dieses Signal wird für die Steuerung des DMA-Controllers benötigt. Dieses Signal ist, im Gegensatz zum Kamera-HSYNC[2], aktiv high und dient zur Auswahl des Zeitbereiches einer Videozeile, indem das Videosignal digitalisiert wird. Eine Digitalisierung findet nur statt, wenn dieses Signal einen High-Pegel aufweist. Die HSYNC Start Register geben die Anzahl der Pixel-Clock-Impulse nach der fallenden Flanke von CSYNC an, bei der die steigende Flanke des HSYNC-Signals generiert wird. Mit den HSYNC Stop Registern wird in analoger Weise die fallende Flanke des HSYNC-Signals spezifiziert. In Bild 31 wird nocheinmal erläutert, wie mit diesen beiden Registern der aktive Teil einer Bildzeile ausgewählt werden kann.

---

[2]Der Name HSYNC für dieses Signal ist eigentlich etwas irreführend. In Wirklichkeit wird hier der aktive Teil der Videozeile spezifiziert.

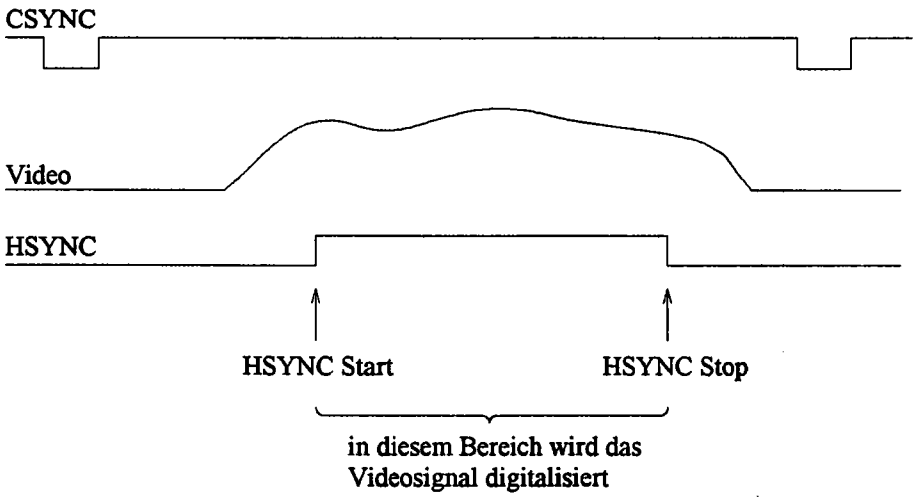



*Bild 31: Auswahl des aktiven Bereiches einer Videozeile mittels der HSYNC Start - und HSYNC Stop Register*

**CLAMP Start- und CLAMP Stop Register**

Hiermit wird das Clamping-Intervall für die Digitalisierung festgelegt. Beim Clamping wird der Schwarz-Pegel des Videosignales auf den Clamping-Pegel justiert. Als Startwert wird von Brooktree ein Zeitpunkt von 500 ns nach der steigenden Flanke von CSYNC empfohlen. Für die Dauer des Clamping-Intervalls wird 2 µs angegeben.

**ZERO Start - und ZERO Stop Register**

Während das ZERO-Signal aktiv ist, werden die internen Komparatoren der A/D-Wandler neu abgeglichen. Das Zero-Signal muß daher in jeder horizontalen Austastlücke aktiviert werden, damit der Linearitätsfehler der A/D-Wandler nicht über den spezifizierten Wert von ±1 LSB wächst. Die Anzahl der Pixel-Clock-Takte, für die das Signal aktiviert wird, ist dabei bedeutungslos.

**FIELD Gate Start - und FIELD Gate Stop Register**

Die Angaben in diesen Registern werden verwendet, um das FIELD-Signal zu erzeugen, mit dem die beiden Halbbilder im Interlaced-Mode unterschieden werden. Die Halbbilder können durch Beobachtung des CSYNC-Signals während der Zeilenmitte unterschieden werden. Da sich am Ende eines ungeraden Halbbildes nur eine halbe Bildzeile befindet, wird das CSYNC-Signal in der Mitte einer "normalen" Bildzeile aktiv. Daraus läßt sich ableiten, daß das folgende Halbbild das gerade sein muß. Mit den FIELD Gate Registern wird der Bereich innerhalb einer Bildzeile festgelegt, in dem das CSYNC-Signal daraufhin beobachtet wird. Brooktree empfiehlt hier einen Startwert von 1/4 HCOUNT und einen Stopwert von 3/4 HCOUNT. Dies erzeugt eine steigende FIELD-Flanke am Anfang eines ungeraden Halbbildes.

**NOISE Gate Start - und NOISE Gate Stop Register**

Während der Vertikal-Austastlücken eines Videosignals, können sich im CSYNC-Signal, in der Zeilenmitte, sogenannte Ausgleichsimpulse befinden. Diese Impulse sind dafür gedacht, daß eine PLL zur Regenerierung der HSYNC-Impulse während der langen Zeit der Vertikal-Austastlücke nicht ausrastet. Diese Austastimpulse müssen aus dem CSYNC-Signal herausgefiltert werden, weil sonst der oben beschriebene Mechanismus zur Erzeugung des FIELD-Signales nicht mehr korrekt arbeiten kann. Dazu dient das sogenannte Noise-Gate im Bt261. Ist das Noise-Gate geschlossen, so werden Impulsflanken im CSYNC-Signal ignoriert. Für den Startwert wird hier von Brooktree ein Wert von HCOUNT/2-2,5 µs empfohlen, für den Stopwert ein Wert von >HCOUNT/2.

**HCOUNT Register**

In diese Register muß die Länge einer Videozeile, ausgedrückt durch die Anzahl von Pixel-Clocks, geschrieben werden.

## 5.6. Die Zeilenbegrenzung

Hier wird der vertikale Bereich eines Bildes spezifiziert, der digitalisiert werden soll. Die Zeilenbegrenzung wird mit zwei Timern realisiert, die im One-Shot-Mode betrieben werden. In den ersten Timer wird dabei die Anzahl der Bildzeilen am Bildanfang geschrieben, die nicht digitalisiert werden sollen. Mittels des zweiten Timers wird die Anzahl der Bildzeilen, die digitalisiert werden sollen, spezifiziert. Für den Interlaced-Betrieb ist zu beachten, daß die Zeilenanzahl sich hier jeweils auf ein Halbbild bezieht. Wird in den zweiten Timer beispielsweise ein Wert von 256 geschrieben, so werden 256 Zeilen vom ungeraden Halbbild und 256 Zeilen vom geraden Halbbild, also insgesamt 512 Zeilen des Vollbildes digitalisiert. Bevor die Werte in die Timer geschrieben werden, muß jeweils vorher ein Timer-Control-Word geschrieben werden. Der folgende Auszug aus einem Listing zeigt, wie die Timer initialisiert werden müssen:

```
#define TIM0_REG        (int *)0x00000500
#define TIM1_REG        (int *)0x00000504
#define TIM_CWD         (int *)0x0000050C

/* specify numer of vertical blank lines in timer 0 */
*TIM_CWD    = 0x32;           /* write control word for timer 0 */
*TIM0_REG   = vert_blank_LSB; /* write LSB of timer 0 */
*TIM0_REG   = vert_blank_MSB; /* write MSB of timer 0 */

/* specify number of visible lines in timer 1 */
*TIM_CWD    = 0x72;           /* write control word for timer 1 */
*TIM1_REG   = vi_lines_LSB;   /* write LSB of timer 1 */
*TIM1_REG   = vi_lines_MSB;   /* write MSB of timer 1 */
```

## 5.7. Der programmierbare Pixel Clock Generator ($0000 0480)

Der programmierbare Pixel Clock Generator erzeugt aus einer 14,318 MHz Grundfrequenz verschiedene, höhere Taktraten, die zur Erzeugung der Pixel-Clock, mittels des Bt261, verwendet werden können. Die Auswahl einer bestimmten Frequenz geschieht durch einen Schreibzugriff auf die Adresse des Clock-Generators. Dabei sind nur die fünf niederwertigen Datenbit von Bedeutung. Es ist nicht möglich, dieses Datum zurückzulesen. Die einstellbaren Frequenzen (in MHz) können aus der folgenden Tabelle entnommen werden:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 25,175 | 8 | 25,175 | 16 | 25,175 | 24 | 65,000 |
| 1 | 28,322 | 9 | 28,322 | 17 | 44,900 | 25 | 72,000 |
| 2 | 40,000 | 10 | 80,000 | 18 | 28,322 | 26 | 74,000 |
| 3 | 32,500 | 11 | 32,500 | 19 | 38,000 | 27 | 76,000 |
| 4 | 50,350 | 12 | 50,350 | 20 | 40,000 | 28 | 78,000 |
| 5 | 65,000 | 13 | 65,000 | 21 | 46,000 | 29 | 80,000 |
| 6 | 38,000 | 14 | 76,000 | 22 | 48,000 | 30 | 100,000 |
| 7 | 44,900 | 15 | 44,900 | 23 | 60,000 | 31 | 110,000 |

Der programmierbare Pixel-Clock-Generator wurde mit dem Baustein ICS1394 realisiert. In der Basis-Library befindet sich deshalb die Routine Init1394() um diesen zu Initialisieren.

## 5.8. Der Video-Synchronisationssignal-Generator ($0000 0540)

### 5.8.1. Adressierungsarten des Bausteines

Mit diesem Generator kann ein komplettes Videotiming für die Synchronisationssignale generiert werden. Damit kann dann eine Kamera extern synchronisiert werden, falls diese die Synchronisationssignale nicht selbst liefern kann. Programmiert wird das Timing mittels der neunzehn 12-Bit-Register des Generators. Aus dem Vergleich zwischen den

Registerinhalten und den Zählerständen der Zähler, für die horizontalen und vertikalen Bildimpulse, werden die Signale generiert.

Zugegriffen wird auf den Generator über eine einzige Adresse. Da der Baustein nur einen 8-Bit-Datenbus besitzt, muß das Low-Byte und das High-Byte eines Registers getrennt geschrieben werden. Außerdem muß vorher die Registeradresse spezifiziert werden. Beim Zugriff auf den Baustein muß zwischen 2 verschiedenen Modi unterschieden werden, die durch die Daten-Bits 8 und 9 spezifiziert werden.

**Manueller Modus:**

Hier wird zuerst die Nummer des Registers, danach das eigentliche Datum als Low-/High-Bytefolge geschrieben. Dabei müssen die Datenbits 8 und 9 folgendermaßen gesetzt werden.

| Zugriff | Bit 9 | Bit 8 |
|---|---|---|
| Registernummer | 0 | 0 |
| Low-Byte des Datums | 0 | 1 |
| High-Byte des Datums | 1 | 1 |

Der manuelle Modus des LM 1882 wird durch die Funktion ManuInit1882() aus der Basis-Library unterstützt.

**Automatischer Modus:**

Dieser Modus ist dafür bestimmt, mehrere Register hintereinander zu beschreiben. Dazu wird zuerst die Anfangsregisternummer in den Baustein geschrieben. Anschließend können mehrere Register, wieder in Low-/High-Byte-Folge geschrieben werden, ohne eine neue Registernummer anzugeben. Dabei müssen die Datenbits 8 und 9 folgendermaßen gesetzt werden:

| Zugriff | Bit 9 | Bit 8 |
|---|---|---|
| Anfangs-Registernummer | 1 | 0 |
| Register-Datum | 1 | 1 |

Der automatische Modus des LM 1882 wird durch die Funktion AutoInit1882() aus der Basis-Library unterstützt.

### 5.8.2. Registerbeschreibung des Baussteines

Alle Register des LM 1882 haben eine Breite von 12 Bit. Der Beginn und das Ende der HSYNC- und VSYNC-Impulse werden durch Start- und Endzählerstände festgelegt. Beim Horizontaltiming beziehen sich die Zählerstände auf Pixel-Clock-Impules, während beim Vertikaltimiming Bildzeilen gezählt werden. Die folgende Tabelle gibt einen Überblick über die Register des LM 1882:

| Registernummer | Registerbeschreibung |
|---|---|
| 0 | Status Register |
| 1 | Horizontaltiming, front porch |
| 2 | Horizontaltiming, HSYNC-Impuls Ende |
| 3 | Horizontaltiming, hor. blanking Intervall **(nicht nutzbar !)** |
| 4 | Horizontaltiming, Länge einer Videozeile |
| 5 | Vertikaltiming, front porch |
| 6 | Vertikaltiming, VSYNC-Impuls Ende |
| 7 | Vertikaltiming, vert. blanking Intervall |
| 8 | Vertikaltiming, Zeilenanzahl pro Vollbild |
| 9 | Equalization-Impuls Ende |
| 10 | Serration-Impuls Ende |
| 11 | Equalization/Serration-Impuls Startzeitpunkt im Vertikaltiming |
| 12 | Equalization/Serration-Impuls Endezeitpunkt im Vertikaltiming |
| 13 | Vertikal Interrupt, aktive Zeit **(nicht nutzbar !)** |
| 14 | Vertikal Interrupt, inaktive Zeit **(nicht nutzbar !)** |
| 15 | Horizontal-Cursorposition Startzeitpunkt **(nicht nutzbar !)** |
| 16 | Horizontal-Cursorposition Endezeitpunkt **(nicht nutzbar !)** |
| 17 | Vertikal-Cursorposition Startzeitpunkt **(nicht nutzbar !)** |
| 18 | Vertikal-Cursorposition Endezeitpunkt **(nicht nutzbar !)** |
| 19-21 | ungenutzt |
| 22 | Restart Vektor |
| 23 | Clear Vektor |

### 5.8.2.1. Das Statusregister

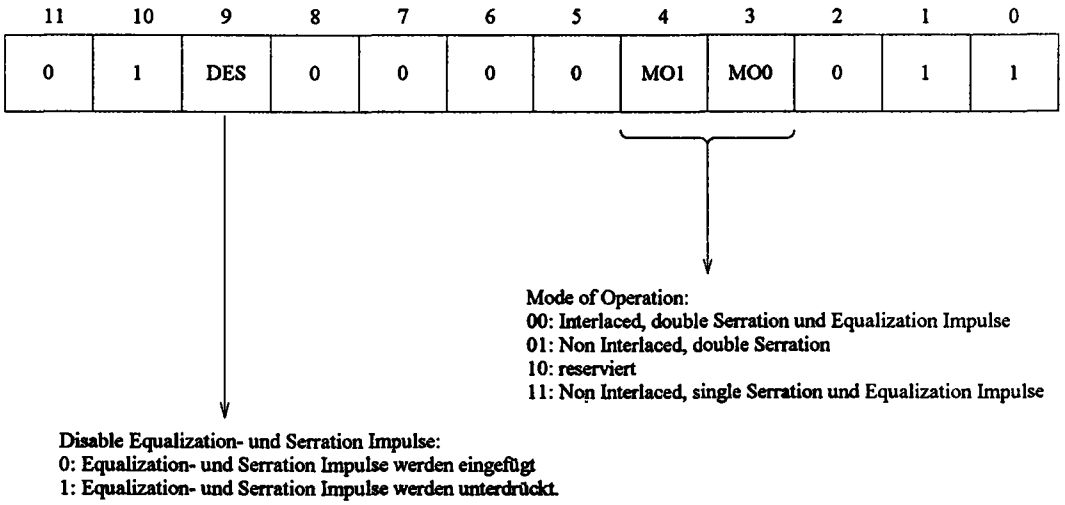



*Bild 32: Bit-Belegung des Status Registers des LM1882*

Die meisten Bits im Status Register sind durch die Einbindung des LM1882 in die Hardware fest vorgegeben. Für diese Bits wurden im obigen Bild die Werte eingetragen, die Bedeutung dieser Bits ist für die Funktion des LM1882 im TIP-CFG irrelevant und wird deshalb auch nicht diskutiert. Mit den MO0- und MO1-Bits wird zwischen Interlaced- und Non Interlaced-Betrieb unterschieden. Außerdem kann hier für den Non Interlaced-Modus zwischen single- und double Equalization/Serration-Impulsen unterschieden werden. Ob diese Impulse tatsächlich in das Timing eingefügt werden, wird mit dem DES-Bit entschieden.

### 5.8.2.2. Berechnung des Horizontaltimings

Die Berechnung des Horizontaltimings wird in Bild 33 veranschaulicht:

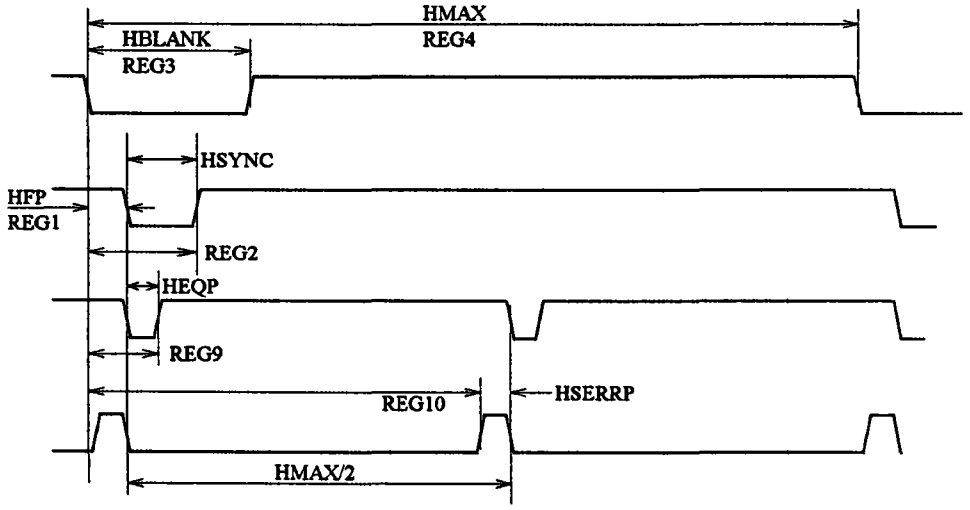



*Bild 33: Berechnung des Horizontaltimings beim LM1882*

Länge einer Bildzeile (HMAX): $REG4 * \tau_{PCLK}$

HSYNC-Dauer: $(REG2 - REG1) * \tau_{PCLK}$

HSYNC-front porch (HFP): $(REG1 - 1) * \tau_{PCLK}$

Equalization-Impulsweite: $(REG9 - REG1) * \tau_{PCLK}$

Serration-Impulsweite: $\left(REG4/n + REG1 - REG10\right) * \tau_{PCLK}$

wobei n=1 für MO1=1, MO0=1 im Statusregister,
n=2 für MO1=0, MO0=1 im Statusregister,
n=2 für MO1=0, MO0=0 im Statusregister

### 5.8.2.3. Berechnung des Vertikaltimings

Bild 34 verdeutlicht die Berechnung des VSYNC-Timings:

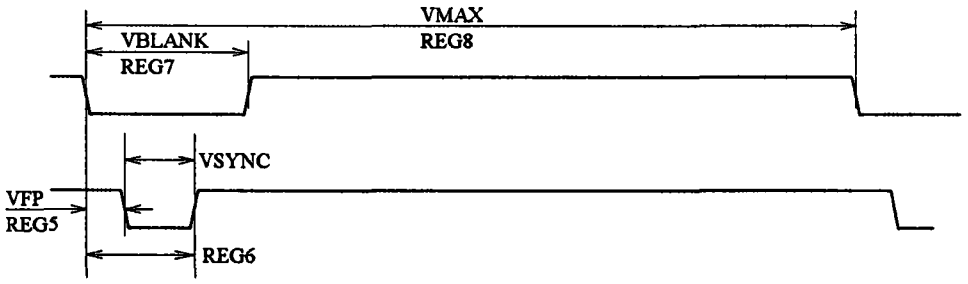



*Bild 34: Berechnung des Vertikaltimings beim LM1882*

Dauer eines Vollbildes: $REG8 * \tau_{Zeile}$

Dauer eines Halbbildes: $REG8 * \tau_{Zeile} / n$

VSYNC-Länge: $(REG6 - REG5) * \tau_{Zeile} / n$

Vertikaltiming front porch: $(REG5 - 1) * \tau_{Zeile} / n$

wobei n=1 für Non Interlaced-Betrieb,
n=2 für Interlaced-Betrieb

### 5.8.2.4. Der Restart Vektor

Bei Zugriff auf diese Registeradresse werden die internen Zähler des LM1882 zurückgesetzt und danach neu gestartet. Ein Restart sollte immer durchgeführt werden,

wenn Registerinhalte verändert wurden. Der Restart-Mechanismus wird durch die Funktion Restart1882() aus der Basis-Library unterstützt.

#### 5.8.2.5. Der Clear Vektor

Bei Zugriff auf diese Registeradresse werden alle Registerinhalte des LM1882 zurückgesetzt. Dies wird durch die Library-Funktion Clear1882() unterstützt.

### 5.9. Programmierungsbeispiel für das Videointerface

Als Kamera für dieses Beispiel diene die XC-711P der Firma Sony. Die Kamera soll am Kameraanschluß1 betrieben werden. Die Synchronisationssignale sind bei dieser Kamera nicht auf einem Videosignal überlagert, sondern werden auf getrennten HSYNC- und VSYNC-Leitungen ausgegeben. Diese Signale sind low-aktiv, müssen also nicht invertiert werden. Daraus folgt, daß das Port Register 0 mit dem Wert $00[3] initialisiert werden muß. Da die Synchronisationssignale dem Kamerainterface 1 entnommen werden, folgt für Port Register 1 ebenfalls der Wert $00. Das VRAM soll im zeilenorientierten Modus beschrieben werden, deshalb wird das Port Register 2 mit $05 initialisiert. Wird nach der Initialisierung aller Register die Digitalisierung gestartet, so muß in diesem Register noch das VDDEN-Bit gesetzt und das RVI-Bit gelöscht werden.

Im Command Register der Digitalisierung muß der Video-Multiplexer und das Datenformat der digitalen Ausgänge gewählt werden. Für die obigen Vorgaben muß hier deshalb der Wert $00 eingetragen werden. Für die untere Referenz soll ein Wert von 300 mV vorgegeben werden. Dies entspricht einem Digitalwert von 64*4,7 mV = $40. Für die obere Referenz wird 1 V genommen, wodurch in die entsprechenden DACs der Wert 213*4,7 mV = $D5 geschrieben werden muß.

Für die Programmierung des Bt261 muß das Timing der Video-Signale bekannt sein. Dies ist dem Handbuch der Kamera zu entnehmen. Für die XC-711P werden von Sony folgende Werte angegeben:

| Pixel-Clock-Frequenz | 14,1875 MHz | |
|---|---|---|
| front porch | 1,55 µs | entspricht 22 Pixel-Clock-Perioden |
| HSYNC | 4,65 µs | entspricht 66 Pixel-Clock-Perioden |
| back porch | 5,71 µs | entspricht 81 Pixel-Clock-Perioden |
| Videoinformation | 52,09 µs | entspricht 739 Pixel-Clock-Perioden |
| Gesamtzeile | 64 µs | entspricht 908 Pixel-Clock-Perioden |

Der Arbeitstakt für die A/D-Wandler soll aus dem 50 MHz-Oszillator abgeleitet werden. Damit muß in das Command Register 0 des Bt261 der Wert $59 geschrieben werden. Nach der Initialisierung muß in diesem Register noch das CPTR-Bit gesetzt werden. Die Kamera arbeitet im Interlaced-Betrieb, wodurch sich für das Command Register 1 der Wert $82 ergibt. Da der 50 MHz-Eingangstakt heruntergeteilt werden muß, wird das Command Register 2 mit $F0 initialisiert. In das Command Register 3 wird immer der Wert $00 geschrieben. In das VSYNC Sample Register wird ein Wert eingetragen, der deutlich größer als die maximale Länge von HSYNC ist, z. B $80.

---

[3]Im folgenden werden alle Initialisierungswerte der Register in Hexadezimal-Darstellung angegeben. Zur Kennzeichnung dieser Darstellung wird den Werten jeweils das $-Zeichen vorangestellt.

Der 50 MHz-Eingangstakt muß durch den Faktor 3,524 geteilt werden, um die gewünschte Pixel-Clock-Rate von 14,1875 MHz zu erhalten. Hier muß auf einen Faktor von 3,5 gerundet werden, weil das Teilverhältnis immer nur halbe Taktperioden des Eingangstaktes umfassen kann. Der Faktor 3,5 entspricht damit 7 Halbperioden des Eingangstaktes, d. h die Summe der Inhalte der OSC Count Low - und OSC Count High Register muß 7 ergeben. In diesem Beispiel wird deshalb für das OSC Count Low Register der Wert 4, und für das OSC Count High Register der Wert 3 gewählt.

Von einer Bildzeile sollen jeweils die ersten 512 Pixel digitalisiert werden. Da die Videoinformation einer Zeile 147 Pixel-Clock-Perioden nach der fallenden Flanke von HSYNC beginnt, muß das HSYNC Start Register mit $093 und das HSYNC Stop Register mit $293 initialisiert werden. Das Clamping-Intervall wird mit CLAMP Start = $050 und CLAMP Stop = $080 in die back porch gelegt. Die gleichen Werte werden auch für das ZERO-Intervall verwendet. Das FIELD-Gate wird durch 1/4HCOUNT und 3/4 HCOUNT begrenzt. Damit ergeben sich für das FIELD Gate Start Register: 0,25*908 = 227 = $0E3 und für das FIELD Gate Stop Register: 0,75*908 = 681 = $2A9. Der Noise Gate Start Wert wird auf HCOUNT/2-2,5 µs = 908/2-35 = $1A3 gesetzt. Für den dementsprechenden Stop Wert empfiehlt Brooktree einen Wert >HCOUNT/2, hier wird der Wert HCOUNT/2+2,5 µs = 908/2+35 = $1E9 verwendet.

# 6. Die Basis-Software

## 6.1. Überblick

Die hier beschriebene Library dient dazu, die Erstellung von Software für das Tip-CFG zu erleichtern. Die Funktionen dieser Library bilden ein Interface zwischen High Level-Routinen und der Hardware.

Auf dem TIP-CFG befinden sich einige Hardware-Bausteine, deren Register nicht rücklesbar sind. Deshalb wird in cfg.h die Struktur Camera definiert, in der die Inhalte aller Register gespeichert sind. Falls ein Registerwert geändert werden soll, so sollte zuerst der zugehörige Wert in der Struktur geändert werden. Danach wird die Library-Funktion aufgerufen, die den Registerwert des Bausteines ändert. Dadurch wird erreicht, daß Software immer alle Registerinhalte ermitteln kann, selbst wenn die Register eines Bausteines nicht zurückgelesen werden können. Außerdem entnehmen die Initialisierungsroutinen, z. B. CamInit(), die Initialisierungs-Daten aus einer solchen Datenstruktur. Dazu muß der Anwender den, in cfg.c deklarierten, globalen Pointer `struct Camera *camera` mit der Adresse der Datenstruktur laden.

Die verschiedenen Funktionen der Library können in zwei Gruppen unterteilt werden. Zum einen gibt es die low-level-Funktionen, die die Inhalte einzelner Register eines Bausteines modifizieren. Andererseits werden diese Routinen wieder von Funktionen aufgerufen, die den ganzen Baustein komplett initialisieren.

Um ein komfortables Arbeiten mit der Library zu ermöglichen wurden Funktionen implementiert, die:

- das TIP-CFG auf einmal initialisieren,
- die Digitalisierung eines Einzelbildes ermöglichen,
- die Digitalisierung starten,
- die Digitalisierung stoppen,
- die Video-Eventquellen HSYNC, VSYNC und FIELD verwalten.

Zur TIP-CFG Basis-Software gehören folgende Dateien:

**cfg.c** Die Quellcode-Datei in der alle Funktionen der Basis-Software enthalten sind.

**cfg.h** Header-Datei in der alle CFG - spezifischen Hardware-Adressen und Daten-Strukturen definiert sind.

**tipbasis.h** In dieser Header-Datei werden all die Dinge spezifiziert, die auf allen TIP-Modulen identisch sind, z. B. die Startadressen der Tabellen, die Adresse des Event-Registers, usw. .

**cfgdef.h** In dieser Header-Datei sind die Initialisierungsdaten einer Standard-CCIR-Kamera spezifiziert. Die Daten werden hierbei in eine Datenstruktur vom Typ Camera, mit dem Namen default_cam geschrieben. Diese Datei ist ein Beispiel dafür, wie Initialisierungs-Dateien manuell geschrieben werden können. Es ist außerdem ein Tool mit dem Namen $D^3T$ verfügbar, welches das Erstellen von Initialisierungsdateien erleichtert.

**cfg.lib** Die compilierte Basis-Library.

**makefile** Dieses Makefile dient dazu, die Basis-Library neu zu compilieren.

## 6.2. Library Beschreibung

### 6.2.1. Globale Daten

Die Basis-Software benutzt einen globalen Pointer auf eine Kamera-Datenstruktur, die in cfg.h deklariert wird. Mit diesem Pointer werden die Initialisierungsdaten, die von den Initialisierungsroutinen benötigt werden, spezifiziert. Dadurch ist es möglich, mehrere Kamerastrukturen gleichzeitig im Speicher zu halten. Soll das TIP-CFG für eine bestimmte Kamera initialisiert werden, so muß dem Pointer nur die Adresse der gewünschten Kamerastruktur zugewiesen werden. Anschließend wird CamInit() aufgerufen, das die gesamte Initialisierung durchführt. Sollen beispielsweise die die Initialisierungsdaten aus cfgdef.h benutzt werden, so hat der Beginn eines Anwenderprogrammes folgendermaßen auszusehen:

```
      .
      .
      .
#include <cfg.h>
#include "cfgdef.h"
      .
      .
      .
extern struct Camera *camera

      .
      .
int main(void)
{
   int result;
       .
       .
   camera = &default_cam;      /* set pointer to desired camera  */
                               /* structure                      */
   result = CamInit();         /* Init TIP-CFG */
       .
       .
       .
```

Der Aufbau der Datenstruktur Camera ist der Header-Datei cfg.h zu entnehmen.

### 6.2.2. Funktionsbeschreibung

## Init1394()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | Init1394(value) | |
| Parameter: | BYTE value | Der Wert value bestimmt dabei die Frequenz des programmierbaren Pixel-Clock-Generators. Dabei sind nur die 5 niederwertigsten Bit von Bedeutung (siehe auch Seite 35) |
| Rückgabewert: | keiner | |

**Beschreibung:**

Mit dieser Funktion wird die Frequenz des programmierbaren Pixel-Clock-Oszillators ICS 1394 festgelegt. Eine Tabelle der möglichen Frequenzen is auf Seite 35 abgebildet.

## AutoInit1882()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = AutoInit1882() | |
| Parameter: | keine | |
| Rückgabewert: | int result | NO_ERROR<br>PARAM_INVALID, falls die Datenstruktur Camera nicht initialisiert war. |

**Beschreibung:**

AutoInit1882() initialisiert den programmierbaren Sync.-Generator im automatischen Modus (siehe Seite 36). Dabei werden die Initialisierungsdaten aus der Datenstruktur entnommen, auf die der globale Pointer *camera zeigt.

## ManuInit1882()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = ManuInit1882() | |
| Parameter: | keine | |
| Rückgabewert: | int result | NO_ERROR<br>PARAM_INVALID, falls die Datenstruktur Camera nicht initialisiert war. |

**Beschreibung:**

ManuInit1882() initialisiert den programmierbaren Sync.-Generator im manuellen Modus (siehe Seite 36). Dabei werden die Initialisierungsdaten aus der Datenstruktur entnommen, auf die der globale Pointer *camera zeigt.

## Write1882()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | Write1882(number, value) | |
| Parameter: | int number | Hiermit wird die Nummer des Registers angegeben, das modifiziert werden soll. |
| | int value | Gibt den neuen Wert des mit number spezifizierten Registers an. |
| Rückgabewert: | keiner | |

## Clear1882()

| | |
|---|---|
| benötigte Header-Datei: | #include <cfg.h> |
| Aufruf: | Clear1882() |
| Parameter: | keine |
| Rückgabewert: | keiner |

**Beschreibung:**

Mit dieser Funktion werden alle Register des LM1882 zurückgesetzt.

## Restart1882()

| | |
|---|---|
| benötigte Header-Datei: | #include <cfg.h> |
| Aufruf: | Restart1882() |
| Parameter: | keine |
| Rückgabewert: | keiner |

**Beschreibung**

Startet den programmierbaren Sync.-Generator neu. Diese Funktion sollte immer aufgerufen werden, nachdem Registerwerte dieses Bausteines modifiziert wurden.

## Init254()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = Init254() | |
| Parameter: | keine | |
| Rückgabewert: | int result | NO_ERROR<br>PARAM_INVALID, falls die Datenstruktur Camera nicht initialisiert war.<br>C_REG_FAILED, falls der Zugriff auf das Command Register nicht erfolgreich war (siehe Seite 27).<br>IOUT_RED_FAILED<br>IOUT_GREEN_FAILED<br>IOUT_BLUE_FAILED, falls der Zugriff auf eines der Referenzregister nicht erfolgreich war (siehe Seite 28). Dabei wird nicht unterschieden zwischen oberer- und unterer Referenz eines Farbkanales. |

**Beschreibung:**

Diese Funktion initialisiert den Bt254 vollständig. Dabei werden die Initialisierungsdaten aus der Datenstruktur entnommen, auf die der globale Pointer *camera zeigt.

## Write254()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = Write254(addr, value) | |
| Parameter: | BYTE *addr | Spezifiziert die Adresse des Registers, dessen Inhalt modifiziert werden soll. |
| | BYTE value | Gibt den neuen Registerinhalt an |
| Rückgabewert: | BYTE result | Der Wert, der nach dem Schreibzugriff von diesem Register zurückgelesen wurde. |

**Beschreibung:**

Mit dieser Funktion wird ein einzelnes Register des Bt254 geschrieben und anschließend zurückgelesen.

## Read254()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = Read254(addr) | |
| Parameter: | BYTE *addr | Spezifiziert die Adresse des Registers, dessen Inhalt modifiziert werden soll. |
| Rückgabewert: | BYTE result | Der Wert, der von diesem Register gelesen wird. |

**Beschreibung:**

Diese Funktion liest ein einzelnes Register des Bt254.

## WriteIOUT254()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = WriteIOUT254(color, gain, offset) | |
| Parameter: | int color | spezifiziert den Farbkanal, für den die Referenzen geändertwerden sollen. Die Farbkanäle sind in cfg.h als RED, GREEN und BLUE definiert. |
| | BYTE offset | gibt den neuen Wert für die untere Referenz des mit color spezifizierten Farbkanales an. |
| | BYTE gain | gibt den neuen Wert für die obere Referenz des mit colorspezifizierten Farbkanales an. |
| Rückgabewert: | int result | NO_ERROR<br>CONSISTENCY_CHECK_FAIL, falls beim Beschreiben der Register ein Fehler aufgetreten ist. |

**Beschreibung:**

Diese Funktion setzt die Werte für die obere- und die untere Referenz eines Farbkanales neu (siehe auch Seite 28).

## ReadIOUT254()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = ReadIOUT254(addr) | |
| Parameter: | BYTE *addr | Spezifiziert die Adresse des Registers, dessen Inhalt gelesen werden soll. |
| Rückgabewert: | BYTE result | Inhalt des durch addr spezifizierten Referenzregisters. |

**Beschreibung:**

Mittels dieser Funktion kann der Inhalt eines der sechs Referenzregister gelesen werden.

## Init261()

| | |
|---|---|
| benötigte Header-Datei: | #include <cfg.h> |
| Aufruf: | Init261() |
| Parameter: | keine |
| Rückgabewert: | keiner |

**Beschreibung:**

Diese Funktion initialisiert den Bt261 vollständig. Dabei werden die Initialisierungsdaten aus der Datenstruktur entnommen, auf die der globale Pointer *camera zeigt.

## Write261()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = Write261(addr, value) | |
| Parameter: | int addr | Registernummer des Bt261, die modifiziert werden soll (siehe Seite 28). Dabei können die 12-Bit-Register des Bt261 mit einem Funktionsaufruf modifiziert werden. Hierzu wird als Registernummer die Nummer des Low-Registers übergeben. |
| | int value | gibt den Wert an, der in das durch addr spezifizierte Register geschrieben wird. |
| Rückgabewert: | int result | NO_ERROR<br>WRITE_STATUS, falls versucht wurde das Status-Register (read only!) zu beschreiben. |

**Beschreibung:**

Mit dieser Funktion wird ein einzelnes Register des Bt261 beschrieben. Dabei wird ein 12-Bit-Register, z.B. HCOUNT-Low und HCOUNT-High, wie ein einzelnes Register behandelt. Hierfür muß für solches Register die Nummer des Low-Registers angegeben werden. Wird für ein 12-Bit-Register die Nummer des High-Registers angegeben, so wird auch nur dieses von der Funktion modifiziert.

## Read261()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = Read261(addr) | |
| Parameter: | int addr | Regeisternummer des Bt261, die gelesen werden soll (siehe Seite 28). Für die Behandlung der 12-Bit-Register gelten die gleichen Regeln, wie sie bei der Funktion Write261() beschrieben sind. |
| Rückgabewert: | int result | Inhalt des durch addr spezifizierten Registers. |

**Beschreibung:**

Liest den Wert eines einzelnen Registers, bzw Registerpaares, des Bt261. Für die Behandlung der 12-Bit-Register gelten die gleichen Regeln, wie sie bei der Funktion Write261() beschrieben sind.

## Init82c54()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = Init82c54() | |
| Parameter: | keine | |
| Rückgabewert: | int result | NO_ERROR<br>PARAM_INVALID, falls die Datenstruktur Camera nicht initialisiert war. |

**Beschreibung:**

Diese Funktion initialisiert die Timer, die für die Zeilenauswahl zuständig sind, vollständig (siehe Seite 34). Dabei werden die Initialisierungsdaten aus der Datenstruktur entnommen, auf die der globale Pointer *camera zeigt.

## Read82c54Val()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | value = Read82c54VAL(counter, status) | |
| Parameter: | int counter | Die Nummer des Timers, der gelesen werden soll. Hier muß für den Timer, der die nicht zu digitalisierenden Zeilen am Bildanfang spezifiziert der Wert 0 angegeben werden. Für den Timer, der die Bildhöhe festlegt muß der Wert 1 angegeben werden. (siehe Seite 34). |
| | BYTE status | Ein Statuswert, der über die Funktion Read82c54Status() gewonnen wird. |
| Rückgabewert: | int value | Der aktuelle Timerwert (16 Bit)<br>WRONG_COUNTER, falls eine falsche Timernummer angegeben wurde. |

**Beschreibung:**

Mit dieser Funktion lassen sich die aktuellen Timerwerte ermitteln, die für die Zeilenauswahl zuständig sind (siehe Seite 34). Bei den ermittelten Werten handelt es sich nicht um die Initialisierungswerte, sondern um die aktuellen Zählerstände der Timer beim Funktionsaufruf. Damit läßt sich feststellen, welche Bildzeile gerade digitalisiert wird. Die Initialisierungswerte der Timer lassen sich nicht zurücklesen. Der Parameter status wird über die Funktion Read82c54Status() gewonnen.

## Read82c54Status()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | status = Read82c54Status(counter) | |
| Parameter: | int counter | Die Nummer des Timers, für den das Statuswort ermittelt werden soll. Hier muß für den Timer, der die nicht zu digitalisierenden Zeilen am Bildanfang spezifiziert der Wert 0 angegeben werden. Für den Timer, der die Bildhöhe festlegt muß der Wert 1 angegeben werden. (siehe Seite 34). |
| Rückgabewert: | BYTE status | Das Statuswort (BYTE) WRONG_COUNTER, falls eine falsche Timernummer angegeben wurde. |

**Beschreibung:**

Diese Funktion ermittelt den Status eines Timers. Das Statuswort wird als Parameter für die Funktion Read82c54Val() benötigt.

## InitPort()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = InitPort(port, value) | |
| Parameter: | int *port<br>int value | Adresse des Port-Registers<br>Neuer Wert, der in das Port-Register geschrieben werden soll. |
| Rückgabewert: | int result | Der Inhalt des Port-Registers, der nach dem Schreibvorgang zurückgelesen wurde. |

**Beschreibung:**

Mit dieser Funktion wird der Inhalt eines Port-Registers (siehe Seite 22) modifiziert. Beim Rückgabewert werden die ungenutzten Bits eines Registers zu "0" gesetzt.

## StartDigit()

benötigte Header-Datei: #include <cfg.h>

Aufruf: StartDigit()

Parameter: keine

Rückgabewert: keiner

**Beschreibung:**

Mit StartDigit() wird die Digitalisierung gestartet, indem das CPTR-Bit (siehe Seite 29) im Bt261 gesetzt wird. Die Digitalisierung wird damit synchron mit dem Beginn des nächsten Vollbildes gestartet.

## StopDigit()

benötigte Header-Datei: #include <cfg.h>

Aufruf: StopDigit()

Parameter: keine

Rückgabewert: keiner

**Beschreibung:**

Mit StopDigit() wird die Digitalisierung angehalten, indem das CPTR-Bit im Bt261 rückgesetzt wird. Die Digitalisierung wird damit synchron mit dem Ende des gerade digitalisierten Vollbildes beendet.

## Wait4HSync()

benötigte Header-Datei: #include <cfg.h>

Aufruf: Wait4HSync()

Parameter: keine

Rückgabewert: keiner

**Beschreibung:**

Diese Funktion wartet, bis ein HSYNC aufgetreten ist.

## Wait4VSync()

benötigte Header-Datei: #include <cfg.h>

Aufruf: Wait4VSync()

Parameter: keine

Rückgabewert: keiner

**Beschreibung:**

Diese Funktion wartet, bis ein VSYNC aufgetreten ist.

## Wait4Field()

benötigte Header-Datei: #include <cfg.h>

Aufruf: Wait4Field()

Parameter: keine

Rückgabewert: keiner

**Beschreibung:**

Diese Funktion wartet, bis ein neues Vollbild beginnt.

## GrabImage()

| | |
|---|---|
| benötigte Header-Datei: | #include <cfg.h> |
| Aufruf: | GrabImage() |
| Parameter: | keine |
| Rückgabewert: | keiner |

**Beschreibung:**

Diese Funktion digitalisiert genau ein Bild.

## SelImageAddress()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | SelImageAddress(addr) | |
| Parameter: | int addr | Startzeile des digitalisierten Bildes im Bildspeicher / 4 (siehe Seite 24). |
| Rückgabewert: | keiner | |

**Beschreibung:**

Mit dieser Funktion wird die Zeile im Bildschirmspeicher festgelegt, ab der das digitalisierte Bild abgelegt wird. Hierbei ist zu beachten, daß der Bildanfang nur in jede durch 4 teilbare Zeile gelegt werden kann, d. h. die mit addr übergebene Zeilennummer wird noch mit dem Faktor 4 multipliziert.

# SetCamPort()

| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = SetCamPort(cam_port,mask) | |
| Parameter: | int cam_port | Der Kameraanschluß der konfiguriert werden soll. |
| | int mask | Konfigurationsdatum für den spezifizierten Kameraanschluß. |
| Rückgabewert: | int result | NO_ERROR<br>INVALID_CAMPORT, falls für den Parameter cam_port ein nicht existierender Kameraanschluß angegeben wurde. |

**Beschreibung:**

Mit dieser Funktion wird die Datenrichtung der Synchronisationsleitungen eines Kameraanschlusses festgelegt (siehe Seite 20). Mit dem Parameter cam_port wird dabei zwischen dem oberen- und dem unteren Kameraanschluß unterschieden. Hierfür sind in cfg.c folgende Makros definiert:

| | |
|---|---|
| CAM_PORT_0 | oberer Kameraanschluß, |
| CAM_PORT_1 | unterer Kameraanschluß. |

Mit dem Parameter mask kann nun für jede Leitung festgelegt werden, ob diese als Eingang oder als Ausgang geschaltet werden soll. Für diesen Parameter sind folgende Makros definiert:

| | |
|---|---|
| PCLK_OUT | Pixelclock-Leitung als Ausgang konfigurieren, |
| PCLK_IN | Pixelclock-Leitung als Eingang konfigurieren, |
| HSYNC_OUT | HSYNC-Leitung als Ausgang konfigurieren, |
| HSYNC_IN | HSYNC-Leitung als Eingang konfigurieren, |
| INVERT_H | HSYNC-Leitung invertieren, |
| VSYNC_OUT | VSYNC-Leitung als Ausgang konfigurieren, |
| VSYNC_IN | VSYNC-Leitung als Eingang konfigurieren, |
| INVERT_V | VSYNC-Leitung invertieren, |
| TRIG_OUT | Trigger-Leitung als Ausgang konfigurieren, |
| TRIG_IN | Trigger-Leitung als Eingang konfigurieren. |

Der Parameter mask wird durch eine Oder-Verknüpfung mehrerer dieser Makros gebildet.

# CamInit()

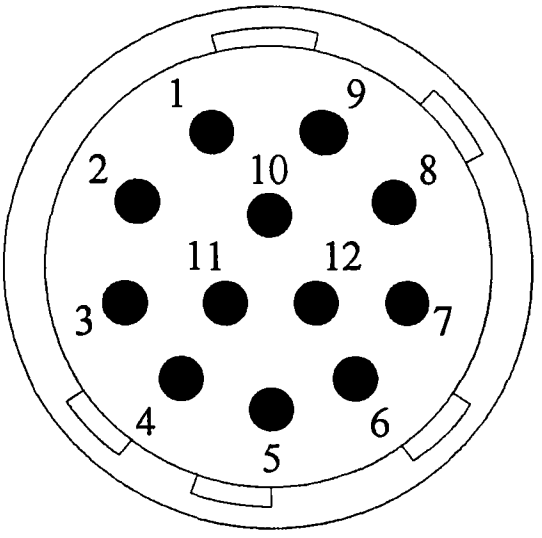
| | | |
|---|---|---|
| benötigte Header-Datei: | #include <cfg.h> | |
| Aufruf: | result = CamInit() | |
| Parameter: | keine | |
| Rückgabewert: | int result | NO_ERROR |
| | | LM1882INIT_FAILED, falls die Initialisierung des prog. Pixelclock-Generators fehlgeschlagen ist. |
| | | BT254INIT_FAILED, falls die Initialisierung des Bt254 fehlgeschlagen ist. |
| | | Bt261INIT_FAILED, falls die Initialisierung des Bt261 fehlgeschlagen ist. |
| | | TIMERINIT_FAILED, falls die Initialisierung der Timer fehlgeschlagen ist. |

**Beschreibung:**

Diese Funktion initialisiert die gesamte Videosektion des TIP-CFG vollständig. Dabei werden die Initialisierungsdaten aus der Datenstruktur entnommen, auf die der globale Pointer *camera zeigt. Die Digitalisierung wird von dieser Funktion nicht gestartet. Hierfür stehen dem Anwender die Funktionen StartDigit() oder GrabImage() zur Verfügung.

# Anhang

### A) Anschlußbelegung des Kamerasteckers




1 - 12V
2 + 12V
3 + 12V
4 AGND
5 Green Video in
6 Red Video in
7 DGND
8 VSYNC
9 PCLK
10 HSYNC
11 Blue Video in
12 Trigger

*Bild 35: Anschlußbelegung des Kamerasteckers*

Obige Anschlußbelegung zeigt die Ansicht auf die Frontblende, bzw. auf dieLötkontakte des Steckers. Der Stecker ist lieferbar von der Fa. Hirose und hat dort die Typenbezeichnung HR10A-10P-12S.

**B) Frontblendenelemente**

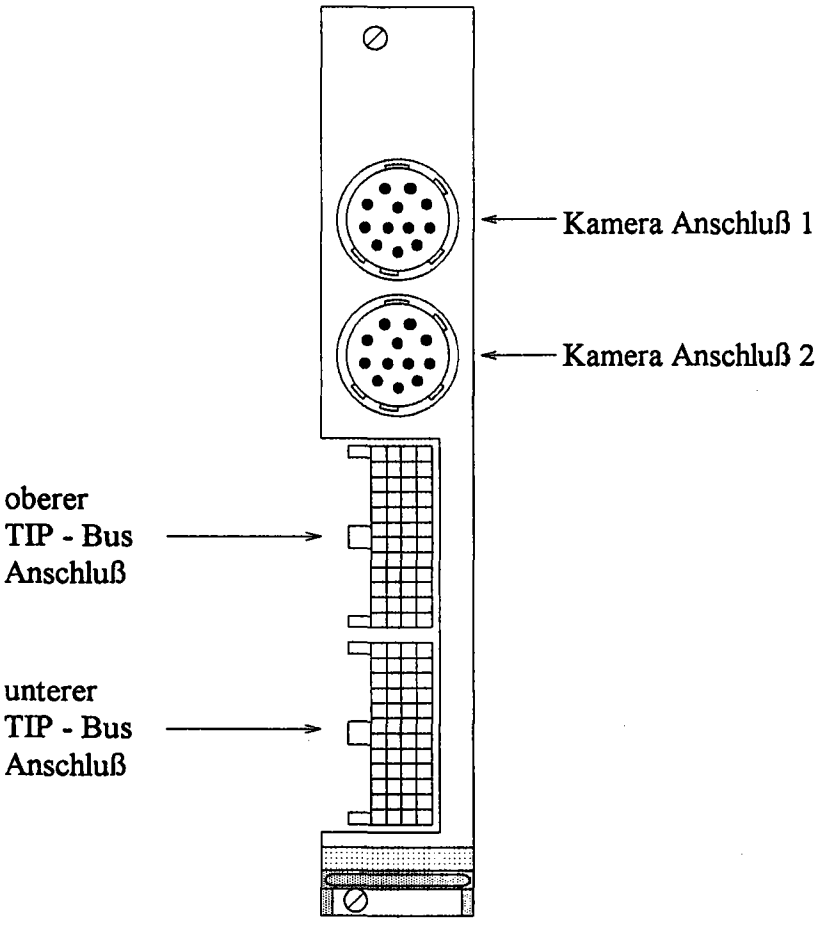



*Bild 36: Fronblendenelemente des TIP-CFG*

## C) Belegung der VG96-Leiste

| | C | B | A |
|---|---|---|---|
| 1 | Reset 0 out + | Reset 1 out + | Reset 0 out - |
| 2 | Link 0 out + | Reset 1 out - | Link 0 out - |
| 3 | GND | Link 1 out + | GND |
| 4 | Link 0 in - | link 1 out - | Link 0 in + |
| 5 | Reset 0 in - | Link 1 in - | Reset 0 in + |
| 6 | Link 1 in + | Reset 1 in - | Reset 1 in + |
| 7 | Reset 2 out + | Reset 3 out + | Reset 2 out - |
| 8 | Link 2 out + | Reset 3 out - | Link 2 out - |
| 9 | GND | Link 3 out + | GND |
| 10 | Link 2 in - | Link 3 out - | Link 2 in + |
| 11 | Reset 2 in - | Link 3 in - | Reset 2 in + |
| 12 | Link 3 in + | Reset 3 in - | Reset 3 in + |
| 13 | RTSC 2 out + | RTSC 2 out + | RTSC 2 out - |
| 14 | RTSC 0 out + | RTSC 2 out - | RTSC 0 out - |
| 15 | GND | RTSC 0 out + | GND |
| 16 | RTSC 0 in - | RTSC 0 out - | RTSC 0 in + |
| 17 | RTSC 2 in - | RTSC 0 in - | RTSC 2 in + |
| 18 | RTSC 0 in + | RTSC 2 in - | RTSC 2 in + |
| 19 | RTSC 3 out + | RTSC 3 out + | RTSC 3 out - |
| 20 | RTSC 1 out + | RTSC 3 out - | RTSC 1 out - |
| 21 | GND | RTSC 1 out + | GND |
| 22 | RTSC 1 in - | RTSC 1 out - | RTSC 1 in + |
| 23 | RTSC 3 in - | RTSC 1 in - | RTSC 3 in + |
| 24 | RTSC 1 in + | RTSC 3 in + | Master Reset |
| 25 | NC | RTSC 3 in - | NC |
| 26 | NC | LSP | NC |
| 27 | +5V | +5V | +5V |
| 28 | +5V | +5V | +5V |
| 29 | +5V | +5V | +5V |
| 30 | GND | GND | GND |
| 31 | GND | GND | GND |
| 32 | GND | GND | GND |

## D) Belegung der 8-Link-Backplane

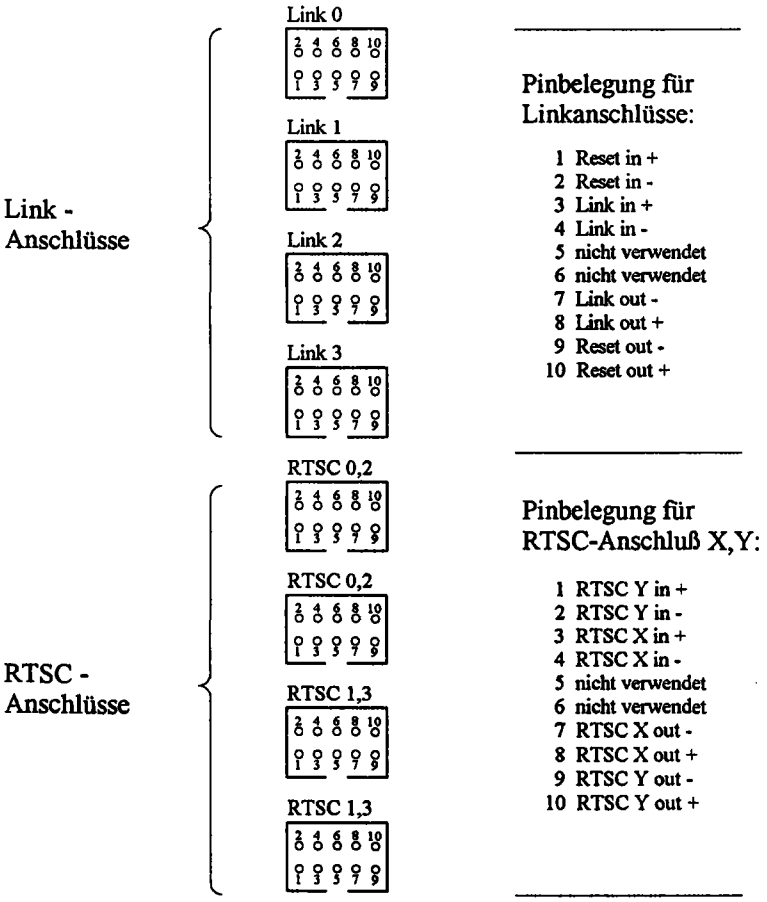



*Bild 37: Belegung der 8-Link-Backplane*

## E) Verschaltung der RTSC-Kanäle auf der 8-Link-Backplane

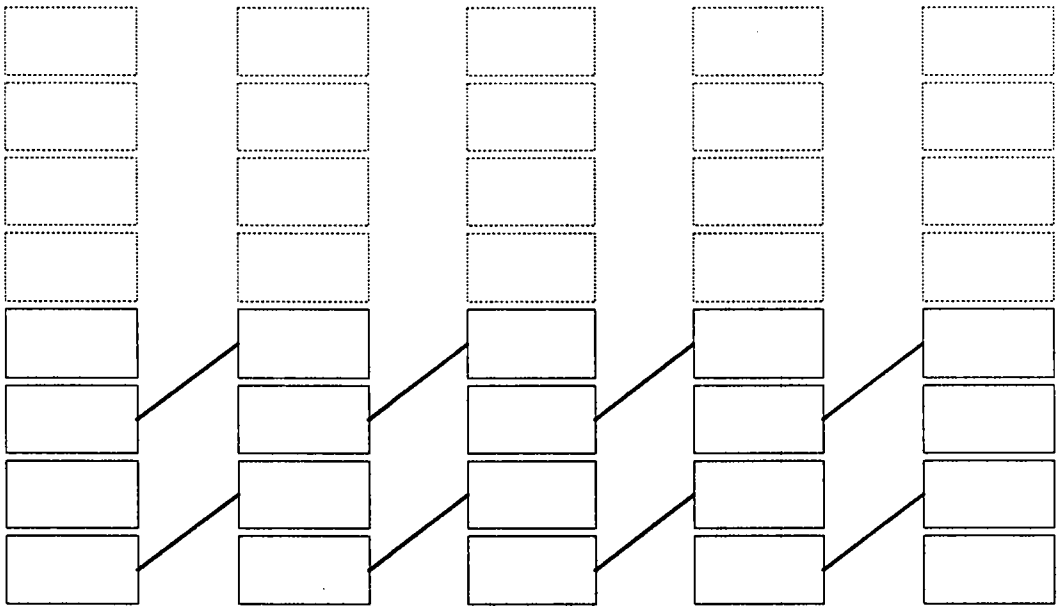


*Bild 38: Verschaltung der RTSC-Kanäle auf der 8-Link-Backplane*

## F) Technische Grunddaten

- Spannungsversorgung: +5V ±5%
- Stromaufnahme: 8A maximal
- Maximale Last an +12V-Versorgung der Kameraanschlüsse: 600 mA
- Maximale Last an -12V-Versorgung der Kameraanschlüsse: 300 mA
- Triggerleitung Ausgangspegel: $V_{OH}$ = 3,4 V (TYP) / 2,7V (MIN)<br>$V_{OL}$ = 0,55 V (MAX)
- Triggerleitung Eingangspegel: $V_{IH}$ = 2,0 V (MIN)<br>$V_{IL}$ = 0,8V (MAX)
- Triggerleitung Belastbarkeit: 24 mA (MAX)
- HSYNC / VSYNC Ausgangspegel: $V_{OH}$ = 3,4 V (TYP) / 2,7V (MIN)<br>$V_{OL}$ = 0,55 V (MAX)
- HSYNC / VSYNC Eingangspegel: $V_{IH}$ = 2,0 V (MIN)<br>$V_{IL}$ = 0,8V (MAX)
- HSYNC / VSYNC Belastbarkeit: 24 mA (MAX)
- zulässiger Temperaturbereich: 0 .. 70 °C
- zulässige Luftfeuchtigkeit: 0 .. 80% (nicht kondensierend)
- maximale Schockbelastung: 30 g
- maximale Vibrationsbelastung: 3g / 100 Hz

# Stichwortverzeichnis

## W



## Z