KFA

KERNFORSCHUNGSANLAGE JÜLICH GmbH

Zentrallabor für Elektronik

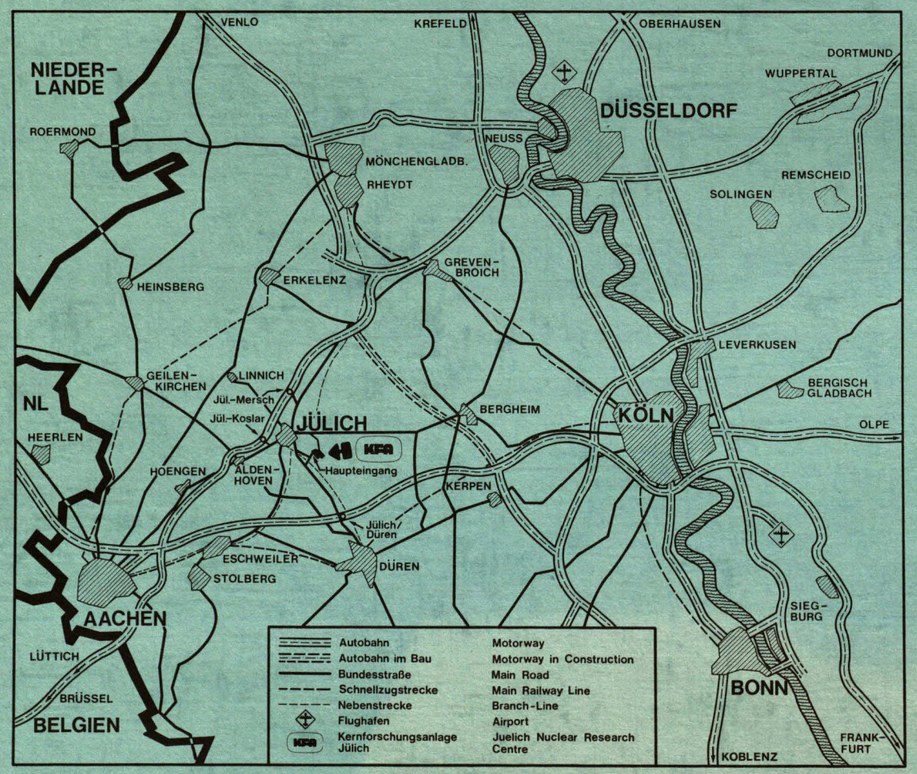
Ein Prozeßdatenübertragungssystem mit standardisierten Komponenten

von

W. Erven, K. Zwoll

Dieser Bericht veröffentlicht u. a. Ergebnisse von Forschungsvorhaben, die vom Bundesministerium für Forschung und Technologie im Rahmen des 2. DV-Programms (Projekt Prozeßlenkung mit DV-Anlagen, KFK Karlsruhe) gefördert wurden.

Jül - 1668
August 1980
ISSN 0366-0885



Als Manuskript gedruckt

Berichte der Kernforschungsanlage Jülich - Nr. 1668

Zentrallabor für Elektronik Jül - 1668

Zu beziehen durch: ZENTRALBIBLIOTHEK der Kernforschungsanlage Jülich GmbH

Postfach 1913 · D-5170 Jülich (Bundesrepublik Deutschland)

Telefon: 02461/611 · Telex: 833556 kfa d

Ein Prozeßdatenübertragungssystem mit standardisierten Komponenten

von

W. Erven, K. Zwoll

Dieser Bericht veröffentlicht u. a. Ergebnisse von Forschungsvorhaben, die vom Bundesministerium für Forschung und Technologie im Rahmen des 2. DV-Programms (Projekt Prozeßlenkung mit DV-Anlagen, KFK Karlsruhe) gefördert wurden.

Inhaltsverzeichnis

Zusammenfassung

Der Bericht beschreibt ein im Zentrallabor für Elektronik der KFA Jülich durchgeführtes und von der Bundesregierung im Rahmen des Projektes "Prozeßlenkung mit DV-Anlagen" gefördertes Entwicklungsvorhaben. Auf der Basis des CAMAC Standards /1,2/ wurde dabei ein Prozeßdatenübertragungssystem mit Anschluß an ein Echtzeitbetriebssystem und Prozess-FORTRAN implementiert, das den besonders hohen Sicherheitsanforderungen industrieller Anwender entspricht. Das zunächst für die direkte Steuerung von Werkzeugmaschinen (DNC) konzipierte System /3/ ist generell zur Steuerung auch anderer räumlich ausgedehnter Prozeßdatenverarbeitungsanlagen geeignet. Der Bericht befaßt sich neben einer kurzen Darstellung des Gesamtsystems schwerpunktmäßig mit der Beschreibung der nicht auf dem Markt verfügbaren Systemkonponenten, die im Rahmen des Vorhabens entwickelt wurden.

1. Einführung

Bei räumlich ausgedehnten und komplexen DNC-Fertigungsanlagen ist das Datenübertragungssystem zwischen den einzelnen Steuerungen und dem Fertigungsrechner eine der wichtigsten Systemkomponenten. Die Sicherheit und Leistungsfähigkeit des Gesamtsystems werden hierdurch entscheidend mitbestimmt.

Aus heutiger Sicht können die wirtschaftlichen Forderungen, bei DV-Systemen eine möglichst lineare Abhängigkeit der Kosten vom Ausbaugrad zu erreichen, bezüglich der Datenübertragung nur durch die Einführung von Bussystemen erfüllt werden. Dies gilt vor allem für räumlich ausgedehnte komplexe Prozesse, z.B. DNC-Systeme, bei denen die herkömmliche sternförmige Systemstruktur einen unverhältnismäßig großen zentralen Rangieraufwand und erhebliche Kabel- und Kabelverlegungskosten zur Folge hat und

somit wirtschaftliche Lösungen erschwert.

Für die Datenübertragung stehen generell serielle oder parallele Verfahren zur Auswahl. Besonderen Vorzug muß den seriellen DÜ-Systemen gegeben werden, weil sie im Hinblick auf Verkabelung, Wartung und Service erhebliche Vorteile aufweisen, auch wenn man die höheren Kostenanteile für elektronische Geräte an den Geräteanschlußstellen mit berücksichtigt. Da man eine serielle Übertragungsstrecke leicht auf einen Ersatzkanal umschalten kann, läßt sich mit vertretbarem Aufwand ein betriebszuverlässiges Datenübertragungssystem aufbauen. Das Problem der rechner- und prozeßseitigen Schnittstellen ist damit jedoch noch nicht gelöst.

Die direkte Ankopplung einzelner Geräte oder elektronischer Baugruppen an das zentrale Bussystem läßt sich nicht ohne einen gewissen Aufwand realisieren. Es erscheint zweckmäßiger, zusammengehörige Baueinheiten in Unterstationen zunächst auf einem untergelagerten parallelen Bussystem aufzuschalten und über einen gemeinsamen Buskoppler an das Datenübertragungssystem anzuschließen.

2. CAMAC-serielle Ringleitung

Mit der CAMAC-seriellen Ringleitung steht ein Datenübertragungssystem zur Verfügung, das dem Anwender die üblichen Vorteile einer standardisierten Ausführung bietet: Unabhängigkeit vom Hersteller, Flexibilität, Transparenz des Gesamtsystems und Service-Erleichterungen. Die Implementation der CAMAC-seriellen Ringleitung ist inzwischen so weit fortgeschritten, daß viele der notwendigen Hardware-Elemente auf dem Markt verfügbar sind und damit praktische eine "Datensteckdose" verwirklicht worden ist /1, 2, 5/.

Die erforderliche Leitungslänge, die in Fertigungsbetrieben oft zwischen einem und zwei Kilometern liegen kann, und die Datenübertragungsrate von etwa 150 kbits/s (max. Leistung der CAMAC-seriellen Ringleitung 5 Mio Bit/s) sind wesentliche Anforderungen eines DNC-Systems, die durch die Leistungsfähigkeit der CAMAC-seriellen Ringleitung voll abgedeckt sind.

Das serielle CAMAC-System ist so aufgebaut, daß ein geschlossener Nachrichtenweg über eine bitserielle Ringleitung zwischen dem Ausgang und Eingang eines Treibers geführt wird (siehe Abb. 1), der direkt mit dem angeschlossenen Prozeßrechner verkehrt. An beliebigen Stellen dieser Datenleitung können bis zu 62 CAMAC-Überrahmen (CAMAC-Crates) angeschlossen werden, die ihrerseits eine zentrale Steuerstation (Serial Crate Controller) sowie Platz für bis zu 23 anwenderspezifische Moduln (Einschubeinheiten) enthalten. Dabei erfolgt der Datenverkehr innerhalb eines CAMAC-Überrahmens über ein paralleles Bussystem, das als ein Subsystem angesehen werden kann /1/.

Ein geeignetes prozedurales Verfahren beim Betrieb der CAMAC-Ringleitung stellt sicher, daß auch spontane Nachrichten (Alarme) einer Unterstation - z.B. Anforderungen auf NC-Steuerdatenzuteilung an die Zentrale - in den Informationsfluß auf der Ringleitung eingeblendet werden können. Da bei einem räumlich ausgedehnten Datenübertragungssystem die Wahrscheinlichkeit für das Auftreten einer Störung nicht mehr vernachlässigbar ist, werden bei den Übertragungsprozeduren entsprechende Vorkehrungen in Form von Codesicherungen getroffen. Ein erkannter Fehler wird in der Regel von dem Ringleitungstreiber durch Wiederholung der Nachricht korrigiert. Erkennt dieser Defekte, die durch dieses Verfahren nicht umgangen werden können (z.B. Ausfall einer kompletten Unterstation oder Leitungsbruch), so ist es bei Verwendung eines Doppelringes möglich, die fehlerhafte Systemkomponente durch Umkonfigurierung abzuschalten (siehe Abb. 2).

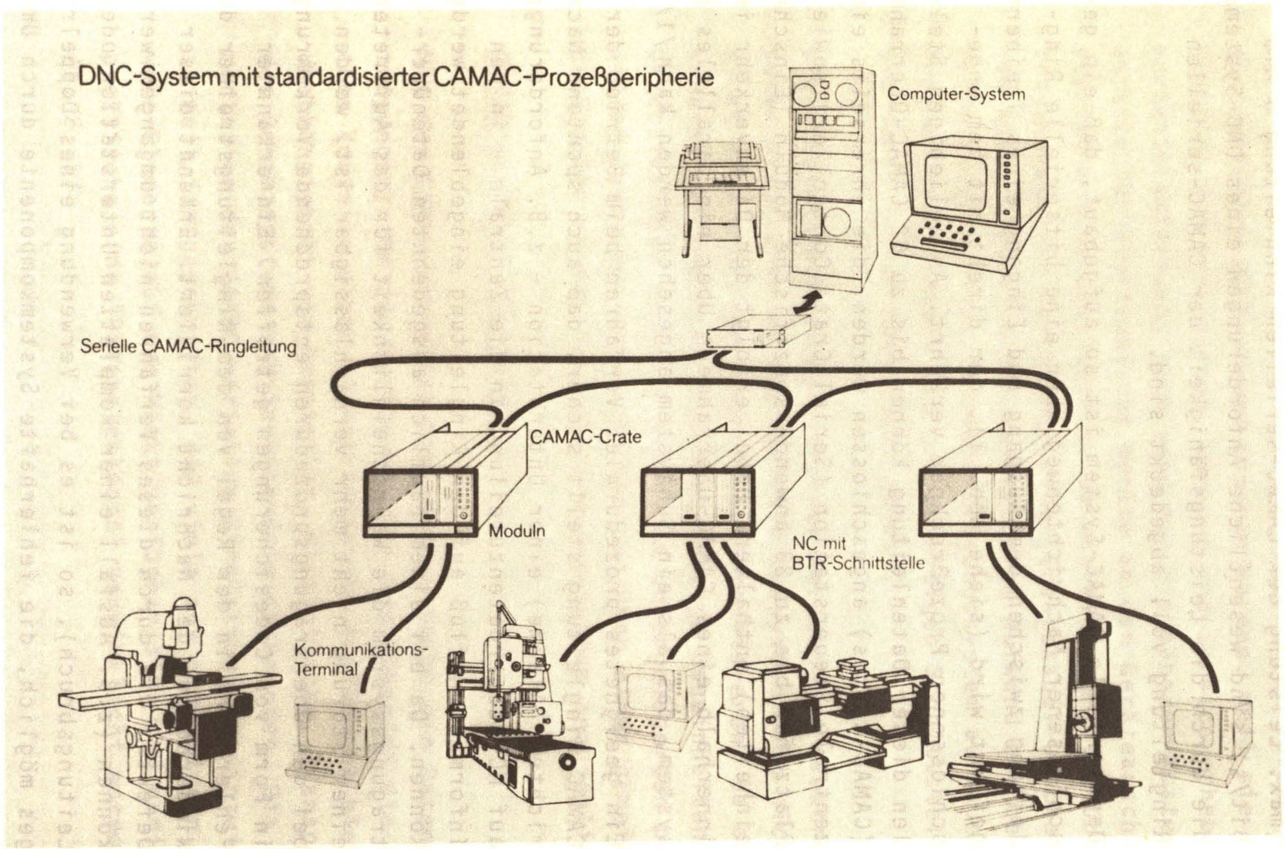


Abb. 1 CAMAC Prozeßperipherie in einem DNC-System

Die Funktionen des Ringleitungstreibers sind in Hard- und Softwareanteile untergliedert. Dabei dient ein Elektronikeinschub als Teil der Rechnerperipherie der Realisierung sehr zeitkritischer Prozeduranteile, während ein Assemblerprogramm im Fertigungsrechner u.a. die Ankopplung an das Betriebssystem übernimmt. Einzelheiten werden bei der Behandlung des CAMAC Ringleitungstreibers beschrieben.

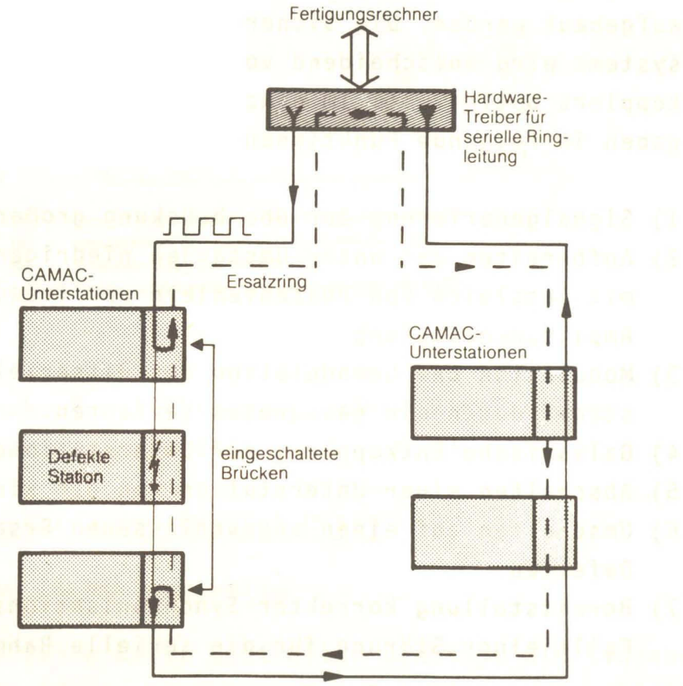
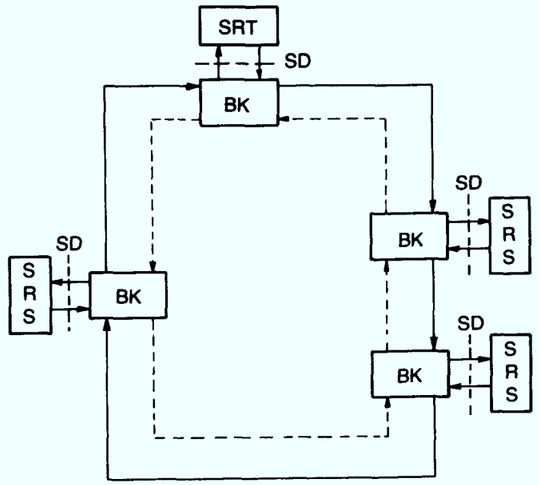


Abb. 2 Umkonfigurierung einer Ringleitung über einen Ersatzring zur Abschaltung defekter Systemkomponenten

Das CAMAC Ringleitungssystem läßt sich ebenso wie andere moderne Prozeßdatenübertragungssysteme (vergl. Abb. 3) strukturieren. Dabei ist der eigentliche Datenübertragungspfad nicht Gegenstand des CAMAC Standards, sondern nur die mechanischen, elektrischen und prozeduralen Festlegungen des "defined Ports" der seriellen Rahmensteuerung (Seriell Crate Controller, siehe /2/). Die Ankopplung an die Datenübertragungsleitung geschieht mit Buskoppler (Loop Control-Unit), die in der Regel als Einschübe des CAMAC Rahmens aufgebaut werden. Die Sicherheit des Datenübertragungssystems wird entscheidend von der Ausführung dieses Buskopplers beeinflußt. Im einzelnen lassen sich seine Aufgaben in folgende Funktionen gliedern:

1) Signalgenerierung zur Überbrückung großer Entfernungen
2) Aufbereiten der unter Umständen niedrigen Eingangspegel mit Ausgleich von Phasenfehlern und frequenzabhängiger Amplitudendämpfung
3) Modulation und Demodulation des Bitseriellen Datenstroms durch ein geeignetes Verfahren.
4) Galvanische Entkopplung der Unterstationen voneinander
5) Abschalten einer Unterstation von der Ringleitung
6) Umschalten auf einen angeschlossenen Ersatzring bei Defekten
7) Bereitstellung korrekter Synchronisationssignale im Falle einer Störung für die serielle Rahmensteuerung

Während der Entwicklungsarbeit wurden einige auf dem Markt angebotene Buskoppler auf ihre Eignung untersucht und auf Basis eines käuflichen Einschubs (Loop Control Unit, Kinetics Systems) eine Weiterentwicklung vorgenommen, die insbesondere auf das Fehlerbehandlungsprogramm (Error Recovery) des CAMAC Ringleitungstreibers abgestimmt wurde.



SRT = Serieller Ringleitungstreiber (Serial Highway Driver SHD)
SD = Standardisierte Datenschnittstelle (Defined Port, DP)
BK = Buskoppler (Loop Control Unit, LCU)
SRS = Serielle Rahmensteuerung (Serial Crate Controller, SCC)

Abb. 3 Struktur des CAMAC-Ringleitungs-systems

2.1 Buskoppler (Loop Control Unit)

2.1.1 Allgemeine Wirkungsweise

Der Serielle Crate Controller benötigt als Eingabe-signale Clock und Daten. Das Datenformat ist CAMAC Standard und in Abb. 4 dargestellt.

Die Information wird seriell und wortweise übertragen. Jedes Wort besteht aus 10 Bit und beginnt mit einem Start-Bit. Danach folgen 6 Bit Nettodaten und ein Delimiter-Bit. Das Delimiter-Bit ist solange Null, wie Datenwörter gesendet werden. Am Schluß folgt ein Parity-Bit und ein Stop-Bit.

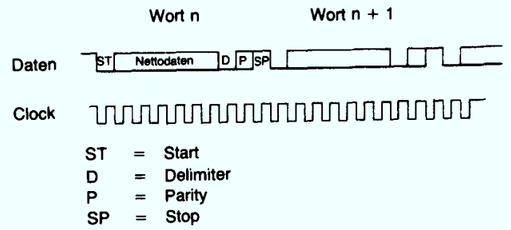


Abb. 4: Dual Port Signale

Bei der direkten Übertragung der beiden Signale werden hohe Anforderungen an die Übertragungsleitung in Bezug auf Störung und Dämpfung gestellt. Störungsfreiheit ist deshalb wichtig, weil bei Störung von Start-, Stop oder Delimiter-Bit die Synchronisierung verloren geht. Dadurch ist die Rahmensteuerung (SCC) nicht mehr in der Lage eine Antwort (Reply Message) abzusenden und dies im Statusregister (Delated Error Bit) entsprechend zu setzen. In diesem Fall kann man deshalb nicht mehr feststellen, ob eine Nachricht (Message) von der Rahmensteuerung richtig erkannt worden ist.

Das Datensignal (NRZ) hat ein Frequenzspektrum von Ø bis zur maximalen Übertragungsrate. Die Übertragungsstrecke darf deshalb für die maximale Übertragungsrate (Clockfrequenz) keine größere Dämpfung als 3 dB aufweisen, weil sonst Störungen durch Phasenfehler auftreten.

Der Buskoppler trägt wesentlich zu einer sicheren Datenübertragung durch folgende Eigenschaften bei:

1) Biphase Modulation

   Mit der biphase Modulation wird aus Clock und Daten ein einziges selbsttackendes Signal erzeugt (siehe Abbildung 5). Dadurch, daß das biphase Signal keinen Gleichspannungsanteil hat, können Ein- und Ausgänge über Transformatoren galvanisch entkoppelt werden. Die Frequenz verändert sich nur noch im Verhältnis

von 1:2, wodurch die frequenzabhängige Kabeldämpfung unkritisch wird. Die Erfahrung hat gezeigt, daß mit der biphase Modulation ca. die fünffache Übertragungsgeschwindigkeit oder die fünffache Entfernung gegenüber der Standard-Übertragung erreicht werden kann. Der Bus-Koppler ist für eine Übertragungsrate von o,1 bis 5 Mbit pro Sekunde ausgelegt.

Die Rückgewinnung (Demodulation) erfolgt mit einem Quarzstabilisierten Clockgenerator, der mit dem Eingangssignal synchronisiert wird. Dadurch wird gewährleistet, daß die Clock auch dann noch für die serielle Rahmensteuerung zur Verfügung steht, wenn das Eingangssignal stark gestört wird oder ganz ausfällt.

2) Regenerierung der Start/Stop-Signale

Damit die Rahmensteuerung im Störungsfall nicht die Synchronisation verliert, werden im Buskoppler die Start- und Stopbits regeneriert. Die Wort-Synchronisierung geschieht mit einem Warte-Wort (Wait Pattern) welches ständig zwischen den Übertragungsphasen ausgesendet wird und im Datenfluß nicht vorkommen kann.

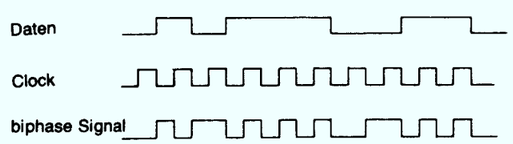


Abb. 5: Biphase Modulation

3) Fehlererkennung

In dem Buskoppler wurde eine Fehlererkennungsschaltung implementiert, die, sobald ein Fehler auftritt, für eine bestimmte Zeit ein definiertes Datensignal erzeugt. Dabei wird das Grenzbit (delimiter bit) zurücksetzt. Die Rahmensteuerung ist dadurch in der Lage, eine eventuell angelaufene Funktion zu beenden und eine Antwort (Reply) auszusenden. Das Fehlersignal ist so gewählt, daß die Rahmensteuerung (am Ende der Übertragungsstrecke auch der serielle Ringleitungstreiber) einen Parityfehler erkennt, wodurch auf jeden Fall die Error-Recovery (siehe 4.2.5) aufgerufen wird.

4) Abschaltung einer gestörten Leitung

Wenn die Fehlererkennung anspricht, wird sofort die Übertragungsstrecke, auf der der Fehler auftrat, abgetet und gleichzeitig auf eine Ersatzverbindung umgeschaltet. Damit wird vermieden, daß eine Störungsfolge (Bündelstörung) auftritt, die durch Abschalten eines Buskopplers (Abschalten der Rahmensteuerung) oder durch einen intermittierenden Kontakt entstehen kann. Eine abgeschaltete Verbindung kann nur unter Programmkontrolle (Error Recovery) wieder aufgebaut werden.

5) Bypass und Loop Collapse

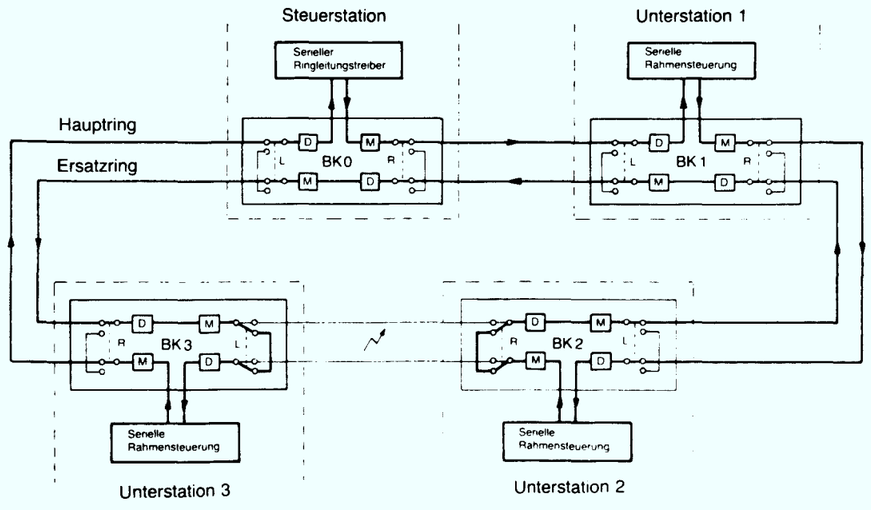
Fällt im Buskoppler die Spannungsversorgung aus, wird die Ringleitung durch Relais, die in diesem Fall abfallen, vom Buskoppler abgeschaltet und überbrückt. Durch den in der Rahmensteuerung vorgesehenen Bypass Mode kann das Ringleitungssignal im Buskoppler auch unter Programmkontrolle durchgeschaltet werden. Dadurch kann die Rahmensteuerung zwar das Signal weiter empfangen, aber keine Botschaft aussenden.

Ebenso kann die Ringleitung auf der rechten Seite (Right Hand Collaps), gesehen vom Buskoppler oder auf

der linken Seite (Left Hand Collapse) abgeschaltet werden, wobei auf eine Ersatzverbindung (Bypass Loop) umgeschaltet wird. Durch diese Möglichkeit läßt sich z.B. ein Teil der Ringleitung unter Programmkontrolle ganz abschalten (Creation of an Island), wenn z.B. Reparaturen oder Umbauten an einem Teil vorgenommen werden müssen.

6) Ersatzring (Bypass Loop)

Die Buskoppler werden durch einen Doppelring miteinander verbunden, wobei der Hauptring im Normalfall die Übertragung übernimmt und der andere als Ersatzring in entgegengesetzter Richtung zum Hauptring betrieben werden kann, um eine Unterbrechung des Hauptringes zu überbrücken. Der Umschaltmechanismus geht aus Abb. 6 hervor.



BK = Buskoppler
M = biphase Modulator
D = biphase Demodulator
R = rechtsseitiger Umleitungsschalter (Right Hand Collapse)
L = linksseitiger Umleitungsschalter (Left Hand Collapse)

Abb. 6: Wirkungsweise des Doppelringes bei einer defekten Leitung

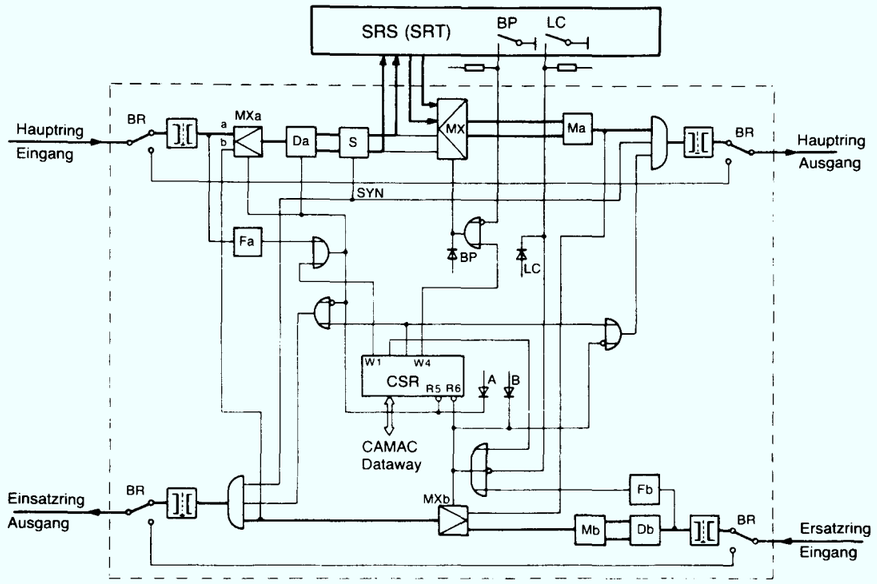
Wird z.B. der Hauptring zwischen BK2 und BK3 gestört, wird zuerst BK3 die Störung erkennen und schaltet automatisch den Schalter L um. Damit wird auch der Ersatzringzwischen BK3 und BK2 abgeschaltet. BK2 erkennt die Störung auf dem Ersatzring und schaltet den Schalter R um. Das Signal erreicht nun Buskoppler BK2 weiter über den Hauptring, wird dort über den Schalter R in den Ersatzring eingespeist und erreicht über den Ersatzring BK3, wo das Signal über Schalter L wieder in den Hauptring eingespeist wird.

2.2 Blockschaltbild

Das Blockschaltbild des Busskopplers ist in Abb. 7 dargestellt. Der Verlauf der Signale im Normalzustand ist hervorgehoben. Ein- und Ausgänge führen über Bypass Relais PR, die abfallen, sobald die Spannungsversorgung des Moduls zusammenbricht. Ein- und Ausgang der Ringleitung sind dann direkt verbunden. Trafokoppler mit elektrostatischer Abschirmung verhindern das Einstreuen von Störungen in das Modul und erlauben eine galvanische Entkopplung der einzelnen Unterstationen voneinander.

Beide Eingangssignale werden jeweils mit einer Fehlererkennungsschaltung überwacht, die sofort ein Signal setzt, wenn das Eingangssignal nicht innerhalb von zwei Clockperioden eine vorwählbare Schwelle übersteigt.

Das Fehlersignal wird etwa 10ms nach Erkennen des letzten Fehlers zurückgesetzt. Das Control/Status Register (CSR) besitzt vier Read/Write Signale (W1-W4) und zwei Read only Signale (R5, R6). Im Normalzustand sind alle Read/Write Signale zurückgesetzt.



BR	=	Bypass-Relay	CSR	=	Control-Status-Register
MX	=	Multiplexer	BP	=	Bypass
D	=	biphase Demodulator	LC	=	Loop Collapse
M	=	biphase Modulator	A	=	Hauptring-Eingang aktiv
S	=	Wort-Synchronisation	B	=	Ersatzring-Eingang aktiv
F	=	Fehlererkennung			

Abb. 7: Blockschaltbild des Buskopplers

Wenn das Fehlersignal Fa und das Statussignal W1 null sind, ist der Multiplexer MXa auf das Eingangssignal a vom Haupteingang geschaltet. Sobald jedoch das Fehlersignal Fa oder das Statussignal W1 (Left Hand Collaps) gesetzt wird, schaltet der Multiplexer MXa auf b um. Diese Eingangsleitung ist normalerweise das Eingangssignal des Ersatzrings.

Das vom Multiplexer MXa ausgewählte Signal wird demoduliert und als Clock und Daten der Synchronisationsschaltung zugeführt. Der Demodulator setzt mit der positiven Flanke des Fehlersignals, das den Multiplexer MXa umschaltet, die Ausgangsdaten für die Zeit von 16 Worten null (maximale Länge einer Botschaft, Message). Damit wird das Delimiter Bit zurückgesetzt und ein Parity-Fehler erzeugt. Die angeschlossene Einheit

ist damit in der Lage, einen Fehler richtig zu erkennen und zu verarbeiten. Die Synchronisationsschaltung synchronisiert sich auf die Warte-Wörter (Wait-Pattern), die zwischen jeder Message gesendet werden und setzt ein Ausgangssignal SYN, sobald ein Warte-Wort erkannt wird. Das Signal SYN wird zurückgesetzt, wenn für längere Zeit (ca. 250 ms) keine Warte-Wörter erkannt werden. In der Synchronisationsschaltung werden Start- und Stop Bit regeneriert, bevor die Daten zu einem Multiplexer und zur angeschlossenen Rahmensteuerung bzw. zum angeschlossenen Ringleitungstreiber weitergeleitet werden.

Mit dem Control Signal BP (Bypass) der Rahmensteuerung oder mit dem Statussignal W4 kann der Multiplexer MX auf das Ausgangssignal der Synchronisierung geschaltet werden, wodurch die angeschlossene Einheit zwar Daten empfangen, aber keine aussenden kann. Der Zustand des Multiplexers wird mit der Leuchtdiode BP (Bypass) angezeigt.

Das Ausgangssignal des Multiplexer MX wird im Modulator Ma in ein biphase Signal umgewandelt und im Normalbetrieb über Trafokopppler in den Hauptring eingespeist.

Das Eingangssignal vom Ersatzring wird demoduliert und wieder moduliert, um Phasenfehler zu beheben. Danach wird das Signal dem Multiplexer MXb zugeführt, der mit dem Fehlersignal von Fb oder dem Statussignal W2 oder dem Control Signal LC (Loop Collapse) der angeschlossenenen Rahmensteuerung auf das eigentliche Ausgangssignal für den Hauptring umschalten kann (Right Hand Collapse). Das Ausgangssignal des Multiplexers wird normalerweise wieder über einen Trafokoppler in den Ersatzring eingespeist.

Die Signalausgänge für den Haupt- und Ersatzring werden abgeschaltet, wenn keine Synchronisationszeichen mehr

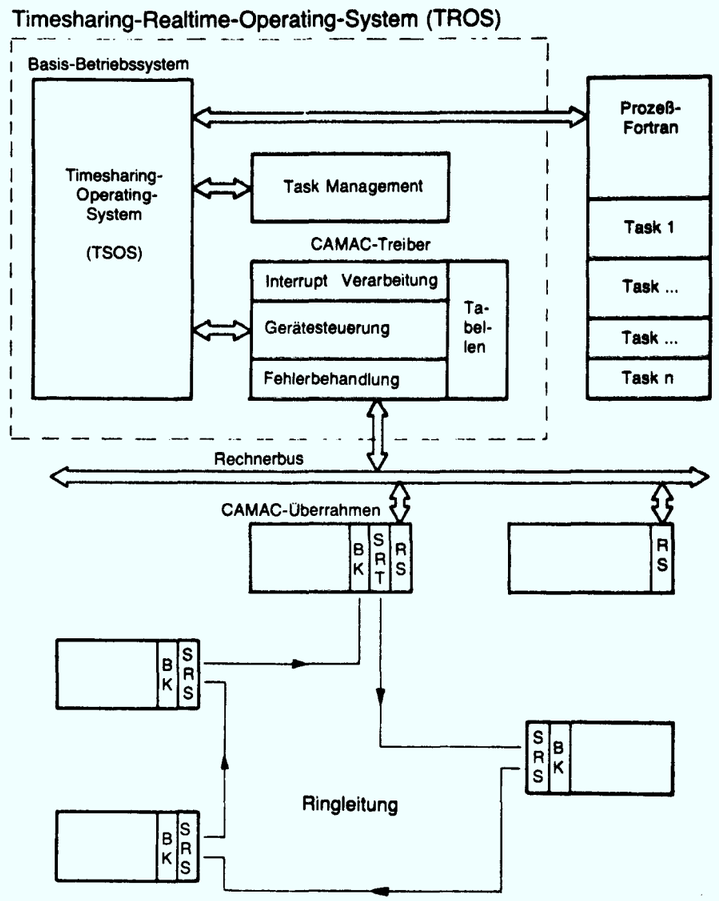
erkannt werden und das Signal SYN zurückgesetzt wird. Der Ausgang zum Hauptring wird zusätzlich abgeschaltet, wenn das Eingangssignal vom Ersatzring gestört ist (Multiplexer MXb umgeschaltet) und der Ausgang zum Ersatzring, wenn das Eingangssignal vom Hauptring gestört ist (Multiplexer MXa umgeschaltet). Ob das Eingangssignal vom Hauptring bzw. vom Bypass Ring vorhanden ist, wird mit den Leuchtdioden A bzw. B angezeigt und kann über R5 und R6 des Controll/Statutsregisters ausgelesen werden.

Das automatische Abschalten der Ausgänge kann mit dem Statussignal W3 verhindert werden, solange das Synchronisationssignal SYN gesetzt ist, um eine Verbindung aufzubauen.

3. CAMAC Ringleitungstreiber

Das Echtzeitverhalten des Prozeßrechnersystems und die erreichbare Datenübertragungsgeschwindigkeit von und zur Prozeßperipherie wird in entscheidendem Maße vom CAMAC-Ringleitungstreiber bestimmt. Darüber hinaus beeinflußt der Treiber wesentlich die Zuverlässigkeit der Datenübertragung auf der seriellen Ringleitung durch entsprechende Strategien, die im Fehlerfall eingeleitet werden.

Da in der Regel keines der zur Verfügung stehenden Echtzeitbetriebssysteme auf dem Markt befindlicher Minicomputer den Anschluß der CAMAC-seriellen Ringleitung a priori unterstützt, wurde der Treiber als eine prozeßnahe Erweiterung des Betriebssystems im Rahmen der Entwicklungsarbeiten erstmalig implementiert. Seine effektive Auslegung hinsichtlich einer optimalen Ankopplung an das Betriebssystem bei gleichzeitiger Ausnutzung der physikalischen Möglichkeiten der CAMAC-seriellen Ringleitung beeinflussen entscheidend die Echtzeiteigenschaften des Systems. Generell können Treiberfunktionen sowohl in Hardware als in Software realisiert werden.



RS = Rahmensteuerung (Crate-Controller, CC)
SRT = Serieller Ringleitungstreiber (Serial Highway Driver, SHD)
BK = Buskoppler (Loop Control Unit, LCU)
SRS = Serielle Rahmensteuerung (Serial Crate Controller, SCC)

Abb. 8 Gesamtsystem

Dabei bringt die reine Softwarerealisierung eine nicht mehr zu vernachlässigende zeitliche Belastung des verwendeten Betriebssystems mit sich, woraus sich zwangsläufig ein niedriger Datendurchsatz zum Prozeß ergibt. Untersuchungen haben gezeigt, daß die dabei erreichbaren Datenübertragungsraten nur für minimal ausgebaute DNC-Systeme ausreichen (vgl. /2/).

Deshalb wurde der Treiber mit einem verteilten Aufwand an Hard- und Software realisiert. Als Hardware-Treiberkomponente wurde der Serial-Highway-Driver von Kinetic-Systems verwendet, der im wesentlichen die parallel-serien bzw. serien-parallel Wandlung der Nachrichten und Antworten, die Abwicklung der standardisierten Ringleitungsprozeduren und die Erkennung von Übertragungsfehlern übernimmt. Er ist als CAMAC Einschub aufgebaut und befindet sich in einem Überrahmen, der rechnernah über eine geeignete Rahmensteuerung mit dem internen Rechnerbus verbunden ist (siehe Abb. 8). Der im Fertigungsrechner implementierte CAMAC-Treiber in Form optimierter Assemblerprogramme ist dem verwendeten Echtzeitbetriebssystem angelagert und übernimmt Funktionen der globalen Gerätesteuerung und der Interrupt-Verwaltung für die gesamte CAMAC-Prozeßperipherie. Ein Fehlerbehandlungsprogramm garantiert zusammen mit den dafür vorgesehenen Hardwaremöglichkeiten die fehlerfreie Datenübertragung.

Die Interrupt- bzw. LAM-Verarbeitung reagiert auf asynchrone oder synchrone Ereignismeldungen der Prozeßperipherie zweischichtig. Zunächst wird in einer Primärreaktion die das Ereignis auslösende Prozeßendstelle identifiziert und entsprechend beeinflußt, weiterhin kann der Systemzustand verwaltungstechnisch aktualisiert werden, indem eine dem Alarm zugeordnete Bool'sche Variable entsprechend verändert wird, die von der zugehörigen Task abgetestet werden kann. Die LAM-Bearbeitung ist bei alarmsynchronisierten Blocktransfers als Primärreaktion enthalten. Es handelt sich dabei jedoch nur um kurze

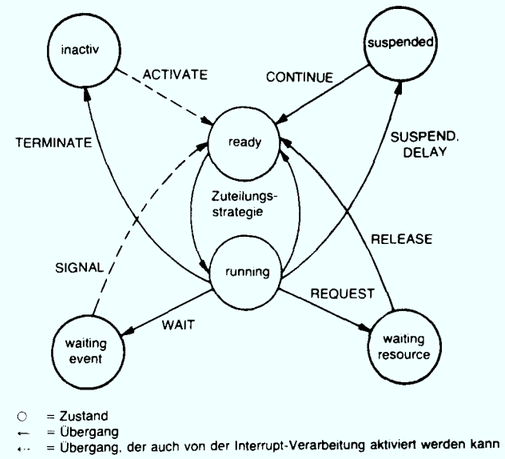


Abb. 9: Zustandsschema von Prozessen (Tasks)

immer gleiche Programmabläufe.

Die eigentliche Bearbeitung des Alarms erfolgt in der Regel in einer prioritätsgesteuerten Sekundärreaktion, die von der LAM-Verarbeitung aktiviert wird und im einzelnen folgende verschiedene Möglichkeiten beinhalten kann (siehe Abb. 9).

1) Ein Prozeß wird vom Zustand "inaktiv" in den Zustand "ready" überführt (vgl. /3, 8/), d.h. die Task ist aktiviert. Die zum Ablauf erforderlichen Betriebsmittel wie Speicherplatz, Ein-/Ausgabegeräte sind jedoch vom Task-Management noch nicht oder nur teilweise zugeteilt.

2) Ein Prozeß geht vom Zustand "waiting event" aufgrund des eintreffenden Alarms in den Zustand "ready".

3) Ein gerade laufender Prozeß wird verzögert und die Unterbrechungsursache von einem dem Treiber angelagerten Unterprogramm bearbeitet, das z.B. Funktionen der "Error Recovery" übernimmt.

Der Verkehr der einzelnen User-Tasks mit der LAM-Bearbeitung geschieht mittels Event-Nummern, die über entsprechende Tabellen mit der Ereignis setzenden Station verknüpft sind.

Die Vielzahl der möglichen Prozeßendstellen (CAMAC Einschübe) mit stark variierbarer Funktionsansprache und der sehr komfortablen standardisierten Blocktransfermechanismen / 4 / in einem CAMAC System bedingen eine hohe Komplexität der CAMAC-Gerätesteuerung (vgl. auch /6, 7/). Um dennoch Handhabbarkeit und Transparenz der Ein-/Ausgabe-Befehle zu garantieren, können Treiberaufrufe über Prozeßvariable und logische Modulnummern (bzw. Ausgabekanäle) erfolgen. Diese werden mit den tatsächlichen Hardwareadressen und Funktionen zur Ausführungszeit im Treiber mit Hilfe umfangreicher Tabellen verknüpft. Diese Tabellen können beim Systemaufbau vom Bediener generiert werden; dabei wird er von entsprechenden Programmhilfsmitteln (Deskriptorprogrammen) unterstützt. Ein-/Ausgabeoperationen können direkt ausgeführt werden, wobei die aufrufende Task bis Ende der Ausführung vom Treiber verzögert wird oder aber im Falle komplexer Aufrufe (z.B. beim langsamen Blocktransfer) autonom vom Treiber ausgeführt werden, wobei die aufrufende Task weiterlaufen kann und die Beendigung der Operationen über Events signalisiert wird.

Im Prozeßfortran erscheinen die Treiber-Aufrufe als "Call"-Statements, deren Parameteraufbereitung von kurzen Assembler-Programmen übernommen wird.

Das Fehlerbehandlungsprogramm (Error Recovery) stützt sich sehr eng auf die im Treiber, Buskoppler und seriellen Crate Controller implementierten Fehlererkennungsmechanismen. Dabei werden sowohl durch Störungen verursachte Datenfehler (z.B. Parityfehler) wie der Ausfall einzelner Systemkomponenten behandelt (z.B. Ausfall einer Unterstation durch Netzabschaltung).

4. Treiber Komponenten

Die verschiedenen Komponenten des Treibers sind im Bild 10 dargestellt. Der Treiber gliedert sich in einem umfangreichen Tabellenteil und in verschiedene Funktionsblöcke. Die Längen der Tabellen können bei der Betriebssystemgenerierung angegeben werden.

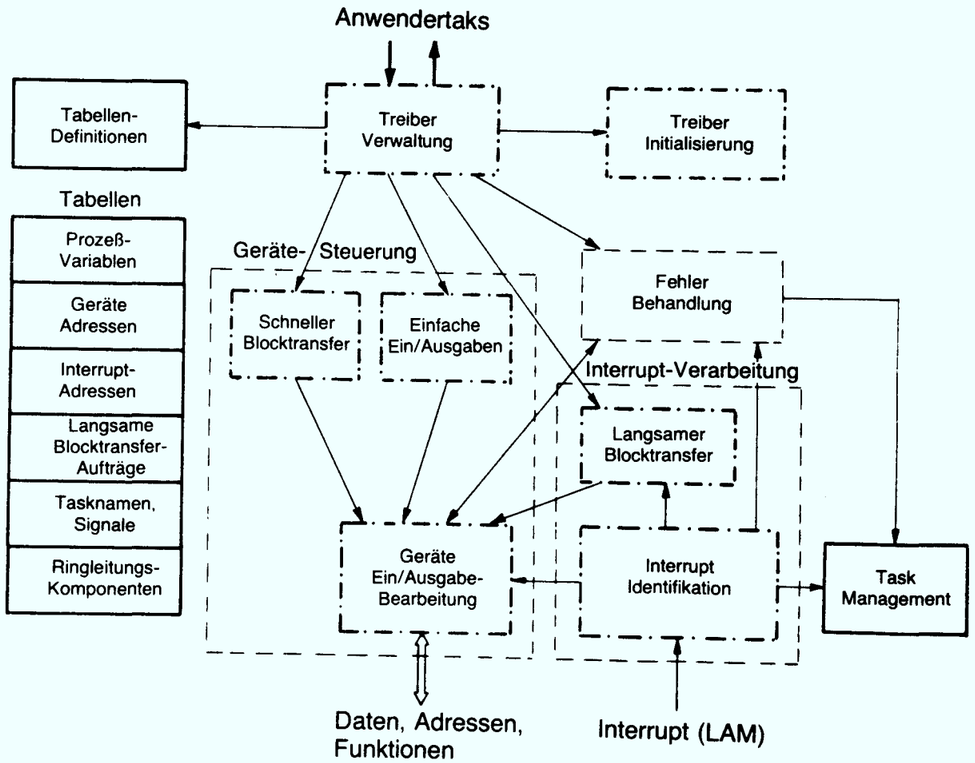


Abb. 10 Komponenten des CAMAC-Treibers

4.1 Tabellen

Für den Einsatz in einem Realtime System muß der CAMAC-Treiber eine genaue Beschreibung der Prozeßperipherie in Form von Tabellen besitzen, um mit wenigen übersichtlichen Kenngrößen auch komplexere Funktionen ausführen zu können, Interrupts zu bedienen, Task's zu starten, Interrupt-synchrone Blocktransfers (langsame Blocktransfers) autonom abzuwickeln und Störungen auf der CAMAC-Ringleitung zu beheben.

4.1.1 Prozeßvariablen (PV-Tabelle)

In ihr sind CAMAC-Funktionen, Moduladressen, Kenngrößen für Datenformat, Blocktransfer usw. Prozeß-Variablen zugeordnet. Bei Ein/Ausgaben oder Interrupt-Definitionen bezieht man sich auf diese Prozeß-Variablen.

4.1.2 Geräteadressen und Modulgruppen (MG-Tabelle)

In der MG-Tabelle werden Crate-, Stationsnummer und eventuell eine Priorität (für die Übergabe von Interrupt-Adressen) logischen Modulgruppen-Nummern zugeordnet. In einer Modulgruppe können max. 3 verschiedene Modultypen definiert werden, wobei die Nummer des Modultyps in der Prozeßvariablen definiert ist. PV- und MG-Nummer zusammen bestimmen die genaue Adresse und Funktion einer CAMAC-Operation.

4.1.3 Interrupt Adressen (LAM-Tabelle)

Alle Interrupts (LAMs), die in einem System auftreten können, werden mit einer PV- und MG-Nummer beschrieben und unter einer Event-Nummer in die LAM-Tabelle eingetragen. Beim Start einer Task durch einen Interrupt wird der Task die MG-Nummer übergeben. Dadurch kann eine Task, die von mehreren Interrupts gestartet werden kann, mit der MG-Nummer direkt das richtige Modul bedienen.

4.1.4 Tasknamen und Signale (Event Tabelle)
In der Event-Tabelle kann eine Task oder eine Signal-Nummer mit einer Event-Nummer verknüpft werden. Wenn ein in der LAM-Tabelle definierter Interrupt bedient worden ist, wird die in der Event-Tabelle zu der entsprechenden Event-Nummer definierte Task gestartet, d.h. von Zustand "inaktiv" in den Zustand "ready" überführt. Anstelle einer Task kann in der Event-Tabelle auch eine Signal-Nummer definiert werden, die dann nach dem Bedienen eines Interrupts gesetzt wird. Eine Task kann damit im Zustand "waiting event" auf einen Interrupt warten.

4.1.5 Langsame Blocktransfer-Aufträge (Kanal-Tabelle)
Alle Aufträge für langsame (interruptsynchrone) Blocktransfers werden mit einer Kanal-Nummer in die Kanal-Tabelle eingetragen. Der Datentransfer und der synchronisierende Interrupt werden jeweils von einer Prozeß-Variablen mit der entsprechenden Kanal-Nummer beschrieben.

4.1.6 Ringleitungskomponenten-Tabelle
In diese Tabelle werden die Nummern aller Crate-Controller sowie die Stationsnummern der zugehörigen Buskoppler, die sich im CAMAC-Ring befinden, in bestimmter Reihenfolge eingetragen. Mit dieser Tabelle ist die Error-Recovery in der Lage, den CAMAC-Ring zu überprüfen und gegebenenfalls Verbindungen neu aufzubauen oder Umleitungen zu schalten.

4.2 Funktionsblöcke

4.2.1 Treiberverwaltung
Die Treiberverwaltung überwacht die Belegung des Treibers. Eine Task, die den Treiber aufruft, wird in eine Warteschlange eingetragen, wenn der Treiber bereits durch eine andere Task oder durch die Interrupt-Verarbeitung belegt ist.

Der Treiber wird grundsätzlich bis zum Ende eines Auftrags von einer Task belegt, wobei die Task verzögert wird. Eine Ausnahme bildet jedoch der langsame interruptsynchronisierte Blocktransfer, der den Treiber nur zyklisch durch die Interrupt-Verarbeitung belegt, während die auftragsgebende Task weiterläuft. Eine Task mit einem langsamen Blocktransfer-Auftrag wird suspendiert, wenn für das entsprechende Modul noch ein langsamer Blocktransfer läuft, bis dieser beendet ist.

4.2.2 Treiber Initialisierung

Über die Initialisierungsfunktion des Treibers können alle Tabellen gelöscht werden, die Ringleitung initialisiert werden oder die Error-Recovery gestartet werden, die den Ring testet und das Ergebnis an die aufrufende Task übergibt. Die aufrufende Task wird solange verzögert, bis der Treiber den Auftrag beendet hat.

4.2.3 Geräte Steuerung

Die Geräte Steuerung verwaltet die Kommunikation mit der Prozeß-Endstelle. Dazu gehören datenlose Befehle wie auch einfache Datenbefehle und der schnelle Blocktransfer. Steuerfunktionen und Endadressen bestimmt die Gerätesteuerung mittels Prozeß-Variablen und Geräte-Nummern aus den Tabellen. Die Geräte-Ein/Ausgabe-Bearbeitung überwacht die Rückmeldung der Geräte und startet im Fehlerfall die Error Recovery. Mit dem Ergebnis der Error Recovery versucht die Gerätesteuerung jeden Übertragungsfehler zu beheben.

4.2.4 Interrupt-Verarbeitung

Die Interrupt-Verarbeitung hat die Aufgaben, jeden Alarm (LAM) zu identifizieren und in Tabellen festgelegte Funktionen zu veranlassen. Dazu gehört der langsame Blocktransfer, Meldungen zum Task-Management und Lokalisierung von sporadischen Fehlern mit Hilfe des Fehlerbehandlungsprogramms.

Das Flußdiagramm der Interrupt-Verarbeitung ist in Abb. 11 dargestellt. Alle Störungen (Parityfehler) auf der Ringleitung starten über einen speziellen Interrupt die Interruptverarbeitung. Diese lokalisiert und behebt mit Hilfe der Error Recovery den Fehler und übergibt bestimmte Fehlerinformationen an eine Fehlerbearbeitungstask, falls eine solche definiert worden ist.

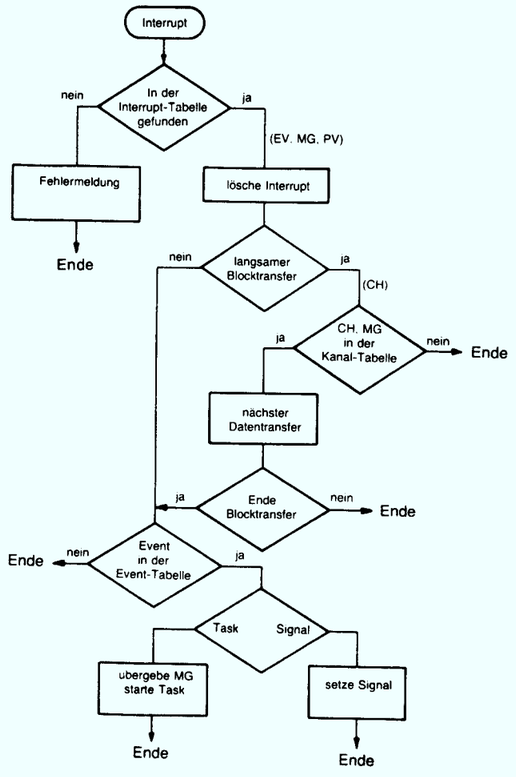


Abb. 11: Flußdiagramm der Interruptverarbeitung

Alle eingehenden Interrupts werden zunächst in der LAM-Tabelle identifiziert und mit der dort gefundenen Prozeß-Variablen gelöscht. Danach wird festgestellt, ob in der Prozeß-Variablen des Interruptes eine Kanal-Nummer angegeben ist. Wenn nicht, wird die in der LAM-Tabelle gefundene Event-Nummer in der Event-Tabelle gesucht. Ist in der Event-Tabelle eine Task definiert, wird zuerst die aus der LAM-Tabelle hervorgegangene MG-Nummer mit der

Kennung "Event-Nr" in einen FIFO Datenbuffer für die Intertaskkommunikation übergeben und danach die Task mit der angegebenen Priorität gestartet, anderenfalls wird nur die gefundene Signal-Nummer gesetzt.

Geht aus der Prozeßvariablen eine Kanal-Nummer hervor, wird diese zusammen mit der MG-Nummer in der Channel-Tabelle gesucht, wo die langsamen Blocktransferaufträge gespeichert sind. Wird dort ein Auftrag mit den entsprechenden Nummern gefunden, wird das nächste Datenzeichen übertragen und geprüft, ob damit der Blocktransfer beendet ist. Wenn ja, wird die Event-Nummer wie bereits beschrieben, in der Event-Tabelle gesucht und die entsprechende Task gestartet bzw. das entsprechende Signal gesetzt.

Wird ein Interrupt nicht in der LAM-Tabelle gefunden oder bei der Interrupt-Behandlung ein Fehler festgestellt, erfolgt eine Fehlermeldung über eine Fehlertask, die mit einer speziellen Event-Nummer in die Event-Tabelle eingetragen sein muß.

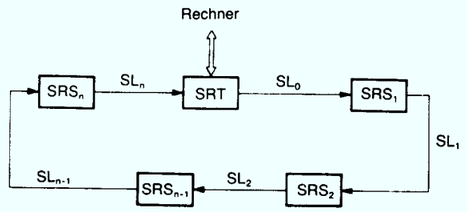
4.2.5 Fehlerbehandlungsprogramm

Das Fehlerbehandlungsprogramm hat in erster Linie die Aufgabe, den aktuellen Zustand der Ringleitung in einer Tabelle zu führen, die Ringleitung zu initialisieren, im Fehlerfall zu überprüfen und gegebenenfalls neue Verbindungen aufzubauen. Die Fehlerbehandlung kann sowohl vom CAMAC-Treiber selbst (Geräte-Ein/Ausgabe-Bearbeitung und Interrupt-Verwaltung), als auch von einer Task aufgerufen werden.

Als Eingangsparameter kann die Stationsnummer einer seriellen Rahmensteuerung (Crate Nummer) übergeben werden, die von dem Fehlerprogramm gesondert behandelt wird. D.h. sobald zu dieser Rahmensteuerung eine Verbindung besteht, wird ein Datenfeld, welches die Daten der letzten Leseoperation speichert (Previous Read Field), ausgelesen,

um Information über den letzten CAMAC-Befehl (zuletzt gelesenes Datenwort, letzter Q-Response und Delated Error Bit) sicherzustellen /2,5/. Diese Informationen sind in erster Linie für die Geräte-Ein/Ausgabe-Bearbeitung wichtig, die diese Informtion benötigt, um zu entscheiden, ob der letzte CAMAC Befehl wiederholt werden muß.

Abb. 13 zeigt das Flußdiagramm der Error Recovery. Um den CAMAC-Ring Crate für Crate mit und gegen den Uhrzeigersinn prüfen zu können, werden Crates und Leitungen wie in Abb.12 logisch numeriert.



SRT = Serieller Ringleitungstreiber
SRS = Serielle Rahmensteuerung
SL = Serielle Leitung

Abb. 12: Logische Numerierung der Ringkomponenten

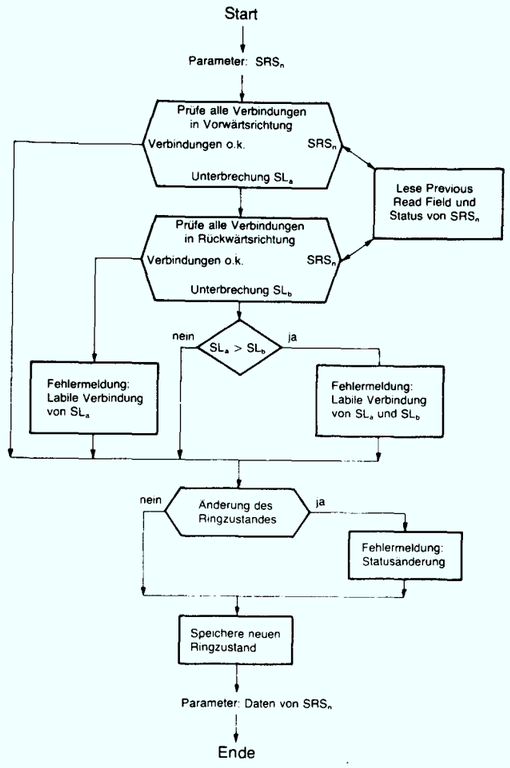


Abb. 13 Flußdiagramm des Fehlerbehandlungs-rogramms

Wie in 2.1 beschrieben, wird eine Verbindung automatisch abgeschaltet, wenn eine größere Störung auftritt. Die Störung erzeugt einen Interrupt zum Rechner, wo sie von der Interrupt-Verarbeitung wahrgenommen wird, welche daraufhin das Fehlerbehandlungsprogramm aufruft.

Das Fehlerprogramm prüft jede Verbindung, beginnend mit SLØ und versucht sie wieder herzustellen, wenn sie unterbrochen ist. Sobald sich eine Verbindung nicht herstellen läßt, merkt sich das Fehlerprogramm die defekte Verbindungsnummer und versucht den Ring von rückwärts her beginnend mit SLn aufzubauen. Im Normalfall wird sich die gleiche Verbindung als fehlerhaft herausstellen wie im vorhergehenden Versuch. Wenn sich jedoch im zweiten Versuch alle Verbindungen aufbauen lassen, oder

eine gestörte Verbindungsnummer festgestellt wird, die kleiner als die im ersten Versuch ist, deutet dies auf eine oder mehrere labile Verbindungen hin. Diese labilen Verbindungen werden einer Fehlertask gemeldet.

Anschließend vergleicht das Fehlerprogramm den neuen Ringzustand mit dem alten, der in einer Tabelle festgehalten wurde. Dabei wird festgestellt, ob die gleichen Crates on line oder off line sind und ob die gleichen Verbindungen gestört sind. Abweichungen werden wieder der Fehlertask gemeldet. Am Ende wird der neue Ringzustand in der Ringleitungs-Tabelle gespeichert.

Bei der Störung einer Botschaft (Message) ist es wichtig, Informationen über diese Botschaft von der adressierten Rahmensteuerung zu erhalten. Da dies unmittelbar mit dem nächsten Befehl für die entsprechende Rahmensteuerung geschehen muß, kann dem Fehlerprogramm die entsprechende Nummer der Rahmensteuerung übergeben werden. Bevor das Fehlerprogramm die entsprechende Rahmensteuerung adressiert, wird zuerst das Previous Read Field ausgelesen und die Daten mit Rückmeldung dem aufrufenden Programm übergegeben.

5. Schnittstelle zum Prozeß-Fortran

Es folgt zunächst eine zusammenfassende Aufstellung aller Fortran-Aufrufe und anschließend eine anwendungsbezogene Erläuterung der verschiedenen Treiber-Aufrufe anhand eines Beispiels.

5.1 Zusammenstellung der Fortran-Aufrufe

Alle Treiberaufrufe sind als Unterprogramm-Aufrufe realisiert. Dadurch wird das Prozeß-Fortran nur erweitert, nicht aber im syntaktischen Kern verändert (vgl. /6/).

5.1.1 Treiber-Initialisierung

Durch die Treiber-Initialisierung werden zunächst einmal alle Belegungsflags zurückgesetzt und der serielle Ringleitungstreiber (Serial Highway Driver, SHD) für den Betrieb der seriellen Ringleitung initialisiert. Diese Grundinitialisierung läuft automatisch bei jedem Neuladen des Betriebssystems ab. Im Prozeß-Fortran werden die Grundinitialisierung und weitere Initialisierungsfunktionen aufgerufen mit:

```
CALL INIT  (K, SCC)

K = Kennung (Integer)
     1  = "Z" in der Steuerstation (Master Crate)
          generieren
     2  = "C" in der Steuerstation generieren
     4  = alle Tabellen löschen
     8  = Ringleitungskonfiguration übergeben
          und Fehlerbehandlungsprogramm anstoßen
          (Rahmensteuerungen initialisieren)
    16  = Fehlerbehandlungsprogramm starten
SCC=Tabelle für Ringleitungskonfiguration
    (Integerfeld)
```

Mit der Kennung K gibt man die gewünschte Initialisierungsfunktion an. Sollen mehrere Funktionen durchgeführt werden, so sind die entsprechenden Kennungen zu addieren.

Da die Funktionen "Z" (Initialisieren) und "C"(Löschen) in der Steuerstation auch die Register des seriellen Ringleitungstreibers (SRT) löschen, dürfen diese Funktionen für die Steuerstation nur mit dem INIT Aufruf durchgeführt werden. Dabei wird der SRT anschließend wieder richtig initialisiert.

Mit der Kennung K = 4 werden alle Treiber-Tabellen gelöscht. Für die Kennung K = 8 muß im Integerfeld SCC eine Tabelle mit allen Rahmensteuerungen und Buskopplern in der

seriellen Ringleitung in physikalischer Reihenfolge übergeben werden. Das Ende der Tabelle wird durch eine Null gekennzeichnet. Ist der CAMAC-Ring nicht mit Buskopplern ausgestattet, muß eine leere Tabelle, d.h. erstes Element ist Null, übergeben werden. Bei der Kennung 8 und 16 wird am Ende die SCC-Tabelle zurückgegeben, die für jede Station eine Kennung enthält, die den Zustand der Station beschreibt.

5.1.2 Prozeß-Variablen Definition

Eine Prozeß-Variable vereinigt in sich zahlreiche Parameter, die Adresse und Funktion einer CAMAC-Operation beschreiben. Sie wird mit folgendem Aufruf definiert:

```
CALL PVDEF (PV, C, N, A, F, DM, BM, I, ST, KN, SM)
```

PV	= Prozeß-Variable	Real
C	= Crate-Nummer	Integer
N	= Stations-Nummer	Integer
A	= Subadresse	Integer
F	= Funktion	Integer
DM	= Datenmode	Integer
BM	= Blockmode	Integer
I	= LAM Bit-Nummer	Integer
ST	= Stopcode	Integer
KN	= Kanal-Nummer	Integer
SM	= Sondermode	Integer

Fehlende Parameter werden Null gesetzt, es müssen aber mindestens die ersten fünf Parameter angegeben werden. Die einzelnen Parameter haben folgende Bedeutung:

1) Prozeß-Variable (PV)
   Vor dem Aufruf muß in der Prozeß-Variablen ein vierstelliger Name abgelegt sein, der normalerweise mit einem "DATA" Statement zugewiesen wird. Mit dem Aufruf werden die Parameter mit dem Namen in der PV Tabelle abgelegt und der Name durch eine interne Arbeitsnummer ersetzt.

   Durch die Arbeitsnummer kann bei weiteren Aufrufen direkt auf die Parameter zugegriffen werden und mit dem Namen können auch andere Programme, die die Ar-

beitsnummer nicht kennen, auf die Parameter zugreifen, indem der Name in der PV-Tabelle gesucht wird.

2) Crate-Nummer (C)
Die Crate-Nummern 1 bis 62 sind für Unterstationen (Crates) in der Ringleitung vorgesehen. Die Crate-Nummer wird an der entsprechenden Rahmensteuerung (Serial Crate Controller) eingestellt. Die Crates, die direkt am Rechnerbus angeschlossen sind (maximal 4), haben die Nummern -1 bis -4. Ist die Crate-Nummer gleich Ø, so werden die echten Crate- und Stationsnummern in der Modulgruppentabelle eingetragen.

3) Stationsnummer (N)
Die Stationsnummer legt den Steckerplatz eines Moduls im Überrahmen (Crate) fest. Ist aber die Crate Nummer mit Null angegeben, bestimmt N die Modulnummer (0,1 oder 2) mit der Crate- und Stationsnummer in die Modulgruppentabelle eingetragen werden.

4) Subadresse (A) und Funktion (F) bestimmen die auszuführende Funktion in einem Modul und sind den entsprechenden Modulunterlagen zu entnehmen.

5) Datenmode (DM)
Der Datenmode legt fest, wie das 24 bit (3Byte) lange Datenwort einer CAMAC Operation interpretiert werden soll.

DM = Ø: 3 Byte Integer mit entsprechender Type-Umwandlung
= 1: 1 Byte " " " "
= 2: 2 Byte " " " "
= 4: 3 Byte Transparent (ohne Type-Beachtung)
= 5: 1 Byte " "
= 6: 2 Byte " "
=-1: kein Standard LAM (Interruptbearbeitung)

Bei der Ausgabe von weniger als 3 Byte wird der Rest Null gesetzt und bei der typenabhängigen Eingabe wird das Vorzeichenbit auf 3 Byte ausgedehnt. DM = -1 besagt, daß es sich bei LAM Prozeduren um ein nicht Standard LAM handelt.

6) Blockmode (SM)
Nach dem Vorschlag in /4/ wird ein Blocktransfer durch drei Kenngrößen beschrieben, die mit folgenden Wichtungen den Blocktransfermode ergeben:

BM = K1 + 4 K2 + 16 K3

K1 bestimmt die Adressfortschaltung, wobei aber nur eine Art realisiert ist:

K1 = 1: (U) Uni-Address

K2 bestimmt die Synchronisierung des Blocktransfers:

K2 = 2: (C) Controller
= 2: (Q) Q-Response
= 3: (L) LAM-Signal

K3 legt das Endekriterium fest:

K3 = 1: (C) Channel Word Count
= 2: Stop on Code
= 3: (S) Q = Ø on Last + 1
= 4: (W) Q = Ø on Last
= 5: (L) LAM-Signal

Kennung K3 = 2 ist nicht aus dem Vorschlag /4/ entnommen und bedeutet, daß der Blocktransfer beendet wird, wenn der mit dem Parameter ST angegebene Wert übertragen wurde.

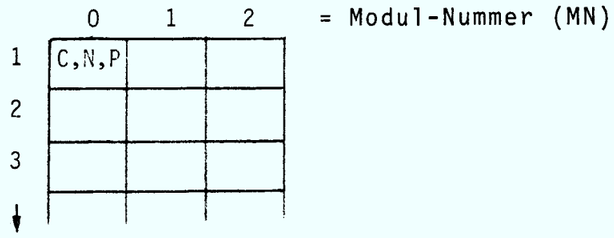
7) LAM Bit (I)
Wird ein LAM (Interrupt) durch ein bestimmtes Bit in einem Register ausgelöst, muß bei einer LAM-Definition die Bit-Nummer (1-24) mit diesem Parameter angegeben werden.

8) Kanal-Nummer (KN)
Bei einem langsamen (LAM-synchronen) Blocktransfer gibt es zwei Prozeß-Variablen, die den Blocktransfer beschreiben (eine für das synchronisierende LAM und eine für die Datenfunktion). Durch eine eindeutige Kanal-Nummer in beiden Prozeß-Variablen werden diese einander zugeordnet. Unter dieser Kanal-Nummer und einer Modulgruppen-Nummer werden die Parameter des langsamen Blocktransfers in der Kanal-Tabelle eingetragen.

9) Sondermode (SM)
Dieser Parameter dient dazu, um im Treiber eine Sonderbehandlung von speziellen CAMAC-Modulen durchzuführen. Als standard Sonder-Mode ist der Mode 1 im Treiber vorhanden, der das Zurücksenden der eingehenden Zeichen in einem Dialog Modul (TTY) bewirkt, wodurch ein fullduplex Betrieb erreicht wird.

5.1.3 Modulgruppendefinition

Crate-Nummer und Stations-Nummer können anstatt in der Prozeß-Variablen in der Modulgruppen-Tabelle definiert werden. Die MG-Tabelle ist eine zweidimensionale Tabelle, wo die Eintragsplätze durch die Modul-Nummer und durch die Modulgruppen (MG)-Nummer bestimmt werden.



Modul-Gruppen-Nummer (MG)

Die Modul-Nummer ist in der PV definiert und bestimmt die Modul-Type. z.B.

0 = BTR (Behind Tape Reader Modul)
1 = SAM (Serial Asynchron Modul)

Die MG-Nummer bestimmt zusammen mit einer Prozeß-Variablen die vollständigen Prozeß Parameter.

Die MG-Tabelle bringt folgende Vorteile:

1. Sind in einem CAMAC-System mehrere gleichartige Module vorhanden, werden die Prozeß-Variablen eines Moduls nur einmal definiert und Crate- und Stationsnummer der einzelnen Module in einer bestimmten Spalte der MG-Tabelle eingetragen.

2. Mit einer einzigen LAM-Definition kann der LAM-Handler alle in einer Spalte der MG-Tabelle definierten Module bedienen und die gefundene MG-Nummer an die zu startenden Task übergeben. Die Übergabe der MG-Nummer kann mit einer Priorität erfolgen, die zu diesem Zweck bei der MG-Definition angegeben wird.

   Die MG-Definition erfolgt mit dem Aufruf:

   ```
   CALL MGDEF (MG, MN, C, N, P)

   MG  = Modulgruppen-Nummer    Integer
   MN  = Modul-Nummer           Integer
   C   = Crate-Nummer           Integer
   N   = Stations-Nummer        Integer
   P   = Priorität (wahlweise)  Integer
   ```

5.1.4 Definiton eines LAM's (Interrupt)

Mit einer LAM-Definition wird ein LAM mit einer Event-Nummer verknüpft. Das LAM wird durch eine Prozeß-Variable und eine MG-Nummer genau definiert. Wird die MG-Nummer nicht oder mit Null angegeben, wird dem LAM-Handler das Auffinden der MG-Nummer überlassen, welche an die zu startenden Task übergeben wird.

Die LAM-Definition erfolgt mit folgendem Aufruf:

```
CALL LAMDEF  (EV, PV, MG)

EV  =  Event-Nummer          Integer
PV  =  Prozeß-Variable       Real
MG  =  Modulgruppen-Nummer   Integer
```

Eine LAM-Definition wird für zwei Zwecke verwendet.
Einmal, um einen langsamen Blocktransfer zu synchronisieren, in diesem Fall ist in PV eine Kanal-Nummer definiert; und zum anderen, um eine Task aufgrund eines LAM's zu starten. Im Normalfall wird der LAM-Handler das LAM mit der Prozeß-Variablen löschen oder abschalten. Ist aber in der Prozeß-Variablen DM mit -1 angegeben, schaltet der LAM-Handler das LAM in der entsprechenden Rahmensteuerung (Crate-Controller) ab. Die danach gestartete Task muß dann in der entsprechenden Rahmensteuerung das LAM wieder zulassen, nachdem die LAM-Quelle gelöscht oder abgeschaltet wurde.

Eine LAM-Definition kann mit dem Aufruf

```
CALL LAMDEF (EV)
```

gelöscht werden.

5.1.5 Zulassen und Sperren von LAM's im Crate-Controller

Alle LAM's in einer Station (Crate) werden über der entsprechenden Rahmensteuerung gemeldet. Mit folgenden Aufrufen kann man eine Rahmensteuerung veranlassen, LAM's zuzulassen oder zu sperren.

```
Zulassen:  CALL CTRLDE (C)
           CALL CTRLDE (PV,MG)

Sperren:   CALL CTRLDD (C)
           CALL CTRLDD (PV,MG)
           C = Crate - Nummer          Integer
          PV = Prozeß-Variable         Real
          MG = Modul-Gruppen-Nummer    Integer
```

Bei der Angabe von PV und MG anstelle von C bestimmt der CAMAC-Treiber die richtige Crate-Nummer aus PV und MG.

5.1.6 Taskdefinitionen

Nach Beendigung eines langsamen Blocktransfers oder nach Erkennen eines LAM's startet der LAM-Handler mit Hilfe der Event-Nummer die Task, die mit folgendem Aufruf mit der Event-Nummer verknüpft wurde:

```
CALL CON  (T, EV, M, P)

    T = Task-name          Double Precision
   EV = Event-Nummer       Integer
    M = Fehlervariable     Integer
    P = Priorität          Integer
```

Mit dem Aufruf

```
CALL UNCON  (T, EV, M)
```

kann eine Taskdefinition aufgehoben werden. Bevor eine Task gestartet wird, macht der LAM-Handler einen Eintrag in den FIFO-Buffer mit folgendem äquivalentem Aufruf (siehe 5.1.8):

```
CALL PARA (1, EV, MG, Ø, P)
```

Die gestartete Task muß dann mit dem Aufruf

```
CALL PARA (2, EV, MG)
```

den Parameter MG übernehmen.

5.1.7 Verbinden eines Signals mit einer Event-Nummer

Anstelle einer Task kann auch eine Signal-Nummer mit einer Event-Nummer mit folgendem Aufruf verbunden werden:

```
CALL CONSI (SI, EV)
SI = Signal-Nummer   Integer
EV = Event-Nummer    Integer
```

Sobald das Event EV eintritt (LAM oder Ende eines langsamen Blocktransfers), wird das Signal SI gesetzt.
Durch die Prozeß-Fortran-Aufrufe "CALL WSIG" (Warten auf Signal) und "CALL TSIG" (Teste Signal) kann der Ablauf einer Task von einem Signal gesteuert werden.

5.1.8 Intertask-Kommunikation

Im CAMAC-Treiber ist ein FIFO-Buffer implementiert, über dem Parameter zwischen verschiedenen Task ausgetauscht werden können. Der FIFO-Buffer hat beliebig viele Datenkanäle, die mit einer Kennung spezifiziert werden.

Die Parameter in einem Datenkanal werden grundsätzlich in der gleichen Reihenfolge ausgegeben, wie sie eingegeben wurden. Sie können aber auch beim Eingeben mit einer Priorität versehen werden, so daß beim Ausgeben die Parameter mit der höchsten Priorität zuerst ausgegeben werden. Die Ausgabe erfolgt also nach dem Kriterium: höchste Priorität und davon der älteste Eintrag. Folgende FIFO-Funktionen sind möglich:

Eingabe eines Parameterpaares

```
CALL PARA (1, K, P1, P2, P)
```

Ausgabe eines Parameterpaares

```
CALL PARA (2, K, P1, P2)
```

Löschen eines Datenkanals

```
CALL PARA (3, K)
```

Löschen des ganzen FIFO-Buffers

```
CALL PARA (4)
```

K =	Kanal-Kennung	Integer
P1,P2=	Parameterpaar	Integer
P =	Priorität (o-7)	Integer

5.1.9 Ein/Ausgabe-Aufrufe

Alle CAMAC-Operationen werden mit dem Aufruf "CALL MOVE" ausgeführt, wobei mit den ersten beiden Parametern PV und MG die CAMAC-Operation genau beschrieben wird. Das Programm wird, außer bei LAM-synchronen Blocktransfers, dabei so lange angehalten, bzw. suspendiert, bis die CAMAC-Operationen beendet ist. Beim LAM-synchronen (langsamen) Blocktransfer wird der Auftrag in die Kanal-Tabelle eingetragen und das Programm läuft danach sofort weiter. Im letzten Parameter wird bei einfachen CAMAC-Operationen eine Fehlererkennung übergeben und beim Blocktransfer dazu die aktuelle Blocklängenangabe. Die Anzahl der Parameter in dem Aufruf und die Art des Treiber-Aufrufes ist abhängig vom F-Code und Blocktransfermode (BM) in der Prozeß-Variablen. Dazu ergeben sich folgende Fälle:

1) Datenlose CAMAC-Operation
   Dabei erfüllt F die Bedingung:
   8 = F 16 oder 24 = F 32
   und der Aufruf hat die Form:
   CALL MOVE (PV, MG, RPL)

   RPL = Reply, bzw. Fehlererkennung (Integer)
   RPL = Ø CAMAC Operation in Ordnung
   = 1 Vom CAMAC-Modul wurde kein Q gesendet
   = 2 Vom CAMAC-Modul wurde kein X gesendet
   = 3 Vom CAMAC-Modul wurde weder X noch Q gesendet
   3 Bei der Übertragung auf der Ringleitung trat eine Störung auf.

2) Einfache Read/Write-Operation

Hierbei wird ein Datenwort von 24 Bit übertragen, wobei eine Umwandlung laut dem Datenmode DM in der Prozeß-Variablen erfolgt. Der Blocktransfermode BM in der Prozeß-Variablen muß mit Null angegeben sein und der F-Code muß folgende Bedingungen erfüllen:

Ø <= F < 8 für Read-Operationen und
16 <= F<24 für Write-Operationen

Der Aufruf hat die Form:

CALL MOVE (PV,MG,DAT,RPL)

Mit der Variablen DAT wird das Datenwort übergeben bzw. übernommen und RPL hat die gleiche Bedeutung wie im vorhergehenden Fall.

3) Schneller (nicht LAM-synchroner) Blocktransfer

Hierbei wird ein Datenblock übertragen, der sich aus einzelnen Read- oder Write-Operationen zusammensetzt. Der Blocktransfermode BM muß in der Prozeß-Variablen angegeben sein, wobei die zweite Kennung ungleich drei ist (LAM-Synchronisierung). Für F gilt die gleiche Bedingung wie für einfache Read/Write-Operationen. Der Aufruf hat die Form:

CALL MOVE (PV,MG,DAT,L,RPV).

DAT gibt hierbei das erste Element eines Datenfeldes an und L (Integer) die maximale Anzahl der einzelnen Datenoperationen, aus denen sich der Blocktransfer zusammensetzt.
RPV ist ein Interfeld von 2 Elementen, wobei im ersten Element die Fehlererkennung der letzten Datenoperation übergeben wird und im zweiten Element die Anzahl der tatsächlichen Datenoperationen, mit der der Blocktransfer beendet wurde.

Beispiel für einen Blocktransfer

```
      INTEGER D(128), R(2)
      DATA BLOCK /'BLCK'/
      CALL PVDEF (BLOCK,-1,3,0,16,1,21)
      CALL MOVE (BLOCK,Ø,D,128,R)
      IF (R(1).EQ.Ø) WRITE (Ø,10)
      IF (R(1).NE.Ø) WRITE (0,12)R
      STOP
   10 FORMAT (' UEBERTRAGUNG O.K.')
   12 FORMAT (' FEHLERKENNUNG: ',I3,'BLOCK LAENGE',I4)
      END
```

Es wird mit der Funktion F = 16 das Integerfeld D an ein Modul ausgegeben, welches sich im Rechnercrate mit der Nummer -1 auf dem Steckerplatz Nr. 3 befindet. Der Datenmode DM = 1 bewirkt, daß nur das niederwertige Byte eines Integer Wortes ausgegeben wird. Da ein CAMAC Datenwort aber immer 3 Byte lang ist, werden die 2 höherwertigen Bytes null gesetzt. Der Blockmode BM = 21 = 1 + 4 + 16 bewirkt, daß die Daten mit maximaler Geschwindigkeit ausgegeben werden, bis die vorgegebene Anzahl von 128 Übertragungen erreicht ist. Tritt während der Übertragung ein Fehler auf (z.B. kein X - Response), wird die Übertragung abgebrochen, wobei in R(1) die Fehlerkennung und in R(2) die Anzahl der fehlerfreien Übertragungen übergeben werden.

4) LAM-synchroner (langsamer) Blocktransfer

Ein LAM synchroner Blocktransferauftrag wird in die Kanal-Tabelle eingetragen und anschließend das Programm fortgesetzt. Die Fehlerkennung wird dabei zunächst auf -1 gesetzt. Am Ende des Blocktransfers wird in den Replyvektor die aktuelle Fehlerkennung und Blocklänge übergeben.

Ist bereits ein Blocktransferauftrag mit der gleichen Kanal- und MG-Nummer vorhanden, wird das aufrufende Programm in einen Wartezustand gesetzt, bis der alte Auftrag beendet ist. Dabei tritt eine Zeitüberwachung in Aktion, die jede Sekunde prüft, ob für den alten Auftrag noch Daten transferriert werden. Wenn nicht, wird der alte Auftrag abgebrochen und als aktuelle Fehlerkennung eine -2 in den Replyvektor übergeben.

Ein Programm kann mit dem Aufruf

```
CALL CHANEL (0, PV, MG)
```

auf die Beendigung eines LAM-synchronen Blocktransfers warten, der mit

```
CALL MOVE (PV, MG, ...)
```

angestoßen wurde. Dabei tritt wieder eine Zeitüberwachung in Aktion, die den Auftrag gegebenenfalls abbricht. Mit dem Aufruf

```
CAll CHANEL (1, PV, MG)
```

kann der entsprechende Blocktransfer abgebrochen werden, wobei wie beim Abbruch durch Zeitüberwachung als Fehlerkennung eine -2 in den Replyvektor übergeben wird.

5.1.10 Ein/Ausgabe-Aufrufe ohne Prozeß-Variable

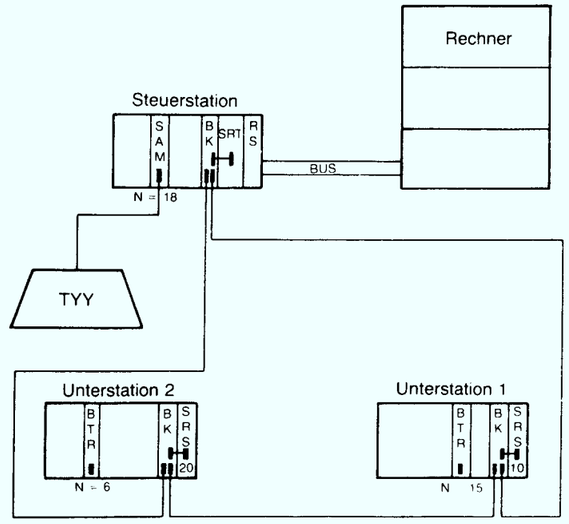
Einfache CAMAC-Operationen können mit dem Aufruf

```
CALL CAMAC (C,N,A,F,RPL)
```

für Datenlose Funktionen und mit dem Aufruf

```
CALL CAMAC (C,N,A,F,DAT,RPL)
```

für Read/Write Funktionen ohne Bezug auf Treibertabellen ausgeführt werden. Die normalerweise in der Prozeß-Variablen definierten Größen C,N,A und F werden dabei direkt angegeben. Die anderen Parameter der Prozeß-Variablen sind dabei als Null anzusehen. Die Variable DAT kann von beliebigem Typ sein. In RPL wird wie bei dem "CALL MOVE"-Aufruf eine Fehlerkennung zurückgegeben.



RS = Rahmensteuerung, angeschlossen am Rechnerbus
SRT = Serieller Ringleitungstreiber
BK = Buskoppler
SRS = Serielle Rahmensteuerung
SAM = Serielles Asynchrones Modul (TTY-Schnittstelle)
BTR = Lochstreifenleser-Ersatz (Behind Tape Reader)

Abb. 14: Aufbau für Programmbeispiel

5.2 Programmbeispiel

Am folgenden Programmbeispiel sollen die einzelnen Treiberaufrufe nochmals erläutert werden. Es wird dazu folgende Hardware-Konfiguration nach Abb. 14 angenommen:

Es ist ein CAMAC-Ring mit zwei Crates aufgebaut, wobei der erste Crate-Controller (SCC) auf die Nummer 10 und der zweite auf die Nummer 20 eingestellt ist. Im Master-Crate befindet sich auf Station 18 ein SAM-Modul /9/, wo ein TTY-Terminal angeschlossen ist. Jeweils ein BTR-Modul /10/ befindet sich in der ersten Unterstation auf dem Steckerplatz 15 und in der zweiten Unterstation auf dem Steckerplatz 6. Dieses Modul enthält im wesentlichen einen FIFO-Puffer zur Zwischenspeicherung von NC-Steuerdaten.

Das Testprogramm soll dazu dienen, wahlweise ein BTR-Modul auf Test zu schalten, wobei das BTR-Modul laufend Daten ausgibt und entsprechend mit Daten im Blocktransfermode versorgt wird. Nach dem Programstart meldet sich das Testprogramm auf der TTY-Konsole mit

TEST BTR NR:

und verlangt die Eingabe einer Nummer, wobei das BTR-Modul in der ersten Unterstation die Nummer 1 und das andere die Nummer 2 hat. Nach Eingabe der Nummer prüft das Programm, ob diese richtig ist und gibt entweder eine Fehlermeldung aus oder initialisiert das entsprechende BTR-Modul für den Test.

Das Gesamtprogramm besteht aus einem Wurzelprogramm und drei Tasks. Das Wurzelprogramm wird an der System-Konsole gestartet und geht in einen Wartezustand, nachdem es die Initialisierungs-Task gestartet hat. Das Wurzelprogramm muß aktiv bleiben, weil mit der Beendigung eines Hauptprogramms alle Tasks gestoppt werden und der Monitor des Betriebssystems sich zurückmeldet.

Die Initialisierungs-Task TINIT initialisiert den CAMAC-Ring und definiert alle Einträge für die CAMAC-Treiber-Tabelle. Am Ende started sie die Dialog-Task TDIAL, die dann das SAM-Modul initialisiert und einen Leseauftrag (LAM synchroner Blocktransfer) für das SAM-Modul übergibt.

Nach Beendigung des Leseauftrags wird die Dialog-Task vom CAMAC-Treiber gestartet, wobei ein Parameterpaar übergeben wird. Der erste Parameter ist dabei die korrespondierende MG-Nummer vom SAM-Modul und der zweite Parameter eine Null.

Nachdem die Dialog-Task eine richtige BTR-Nummer erhalten hat, initialisiert sie das entsprechende BTR-Modul für den Testbetrieb.

Sobald das LAM im BTR-Modul zugelassen ist, startet dieses die Nachfüll-Task TBTR. Diese Task gibt dann einen Datenbuffer von 128 Byte, der mit einer Rampe initialisiert wurde, an das BTR-Modul. Wenn das FIFO im BTR-Modul weniger als halbvoll ist, setzt das BTR-Modul ein LAM, worauf die Nachfüll-Task den nächsten Block ausgibt.
Der gemeinsame residente COMMON-Bereich für alle Task's ist der PVACOM mit den Prozeß-Variablen und der DIACOM mit den Input-Buffer und Reply-Vektoren für den Dialog.

Alle residenten COMMON-Bereiche werden vor dem Übersetzen der Programme mit Namen und Länge dem Übersetzungs-System bekannt gegeben.

5.3 Programmlisten

5.3.1 Wurzelprogramm

```
C     PROGRAMM-NAME: WURZEL
C
C     PROGRAMM-BESCHREIBUNG:
C
C     DAS WURZEL-PROGRAMM MUSS ALS HAUPTPROGRAMM IM HINTER-
C     GRUND LAUFEN, WEIL OHNE HAUPTPROGRAMM AUCH KEINE TASK
C     LAUFEN KANN.
C
      DOUBLE PRECISION TINIT
C
C-----STARTE DIE TASK "TINIT", DER NAME MUSS MIT EINER DOUBLE
C     PRECISION VARIBLEN OBERGEBEN WERDEN.
C
      DATA TINIT /'TINIT'/
      CALL START (TINIT,0,0,M,10)
C
C-----GEHE IN EINEN WARTEZUSTAND MIT
C     "RESET SIGNAL 40" UND "WAIT SIGNAL 40".
C     DAS SIGNAL 40 WIRD NIRGENDWO ANDERS VERWENDET UND DAMIT
C     BLEIBT DAS HAUPTPROGRAMM IN EINEM PERMANENTEN WARTEZU-
C     STAND, OHNE RECHENZEIT ZU VERBRAUCHEN.
C
      CALL RSIG (40)
      CALL WSIG (40)
C
      END
```

5.3.2 Initialisierungstask

```
C     TASK-NAME: TINIT
C     STATUS:    TRANSIENT
C     PRIORITÄT: 10
C
C     TASK-BESCHREIBUNG:
C
C     DIESE TASK INITIALISIERT DEN CAMAC-RING UND ALLE TABELLEN
C     IM CAMAC-TREIBER.
C
      INTEGER SCC (10)
      DOUBLE PRECISION TDIAL, TBTR
C
C-----ALLE PROZESS-VARIABLEN STEHEN IM RESIDENTEN COMMON
C     /PVACOM/ UND SIND SOMIT VON ALLEN TASK'S ZUGÄNGLICH.
C
      COMMON /PVACOM/ BWST,BWBL,BDLA,BEOU,BELA,BETE,BEBL,
     1                TRBL,TERE,TWBL,TETR,TSMR,TCLT,TCLR
C
C-----ZUWEISUNG DER PROZESS-VARIABLEN NAMEN.
C     DER NAME MUSS 4 ZEICHEN LANG SEIN, BRAUCHT ABER NICHT MIT
C     DEM SYNTAKTISCHEN NAMEN DER PROZESS-VARIABLEN ÜBEREINZU-
C     STIMMEN.
C
      DATA BWST,BWBL,BDLA,BEOU/'BWST','BWBL','BDLA','BEOU'/
      DATA BELA,BETE,BEBL     /'BELA','BETE','BEBL'/
      DATA TRBL,TERE,TWBL,TETR/'TRBL','TERE','TWBL','TETR'/
      DATA TSMR,TCLT,TCLR     /'TSMA','TCLT','TCLR'/
C
C-----ZUWEISUNG DER TASK-NAMEN.
C
      DATA TDIAL,TBTR        /'TDIAL','TBTR'/
C
C-----ERSTELLUNG DER SCC-TABELLE.
C     ALLE SCC-CONTROLLER MÜSSEN IN PHYSIKALISCHER REIHENFOLGE
C     ANGEGEBEN WERDEN UND DIE SCC-NUMMERN MÜSSEN NUMERISCH AUF-
C     STEIGEND SEIN. DAS ENDE WIRD DURCH NULL GEKENNZEICHNET.
```

```
C
      DATA SCC(1),SCC(2),SCC(3)     /10,20,0/
C
C-----GENERIERE EIN "Z" IM MASTER CRATE (1), LÖSCHE ALLE TREI-
C     BERTABELLEN (4), ÜBERGEBE DIE SCC-TABELLE UND STOSSE DIE
C     ERROR-RECOVERY AN (8). 1 + 4 + 8 = 13 = KENNUNG FÜR
C     "CALL INIT".
C
      CALL INIT (13,SCC)
C
C-----DER CAMAC-TREIBER GIBT DIE SCC-TABELLE ZURÜCK UND ADDIERT
C     EINE 128 AUF ALLE CRATE NUMMERN, DIE VON DER ERROR RECO-
C     VERY ALS IN ORDNUNG BEFUNDEN WURDEN.
C
      DO 200 I = 1,10
      IC=SCC(I)
C??   ENDE DER SCC-TABELLE
      IF (IC.EQ.0) GOTO 220
C??   IST DER SCC O.K.
      IF (IC.GE.128) GOTO 120
      WRITE (0,10) IC
   10 FORMAT (' CRATE NR:',13,'NICHT IM RING')
      GOTO 200
C
  120 IC = IC - 128
C-----SETZE BIT 9 (DEMAND ENABLE) UND BIT 1 (INITIATES DATAWAY
C     Z) IM STATUS-REGISTER DES SERIAL CRATE CONTROLLERS.
      CALL CAMAC (IC,30,0,17,257)
  200 CONTINUE
C
C-----DEFINIERE DIE PROZESS-VARIABLEN FÜR DAS BTR-MODUL
C
C     BWST = WRITE STATUS-REGISTER BELA = ENABLE LAM
C     BWBL = WRITE DATA BLOCK      BETE = ENABLE TEST
C     BDLA = DISABLE LAM           BEBL = ENABLE BLOCK REQUEST
C     BEOU = ENABLE OUTPUT
C
```

```
  220 CALL PVDEF (BWST,0,0,0,17)
      CALL PVDEF (BWBL,0,0,0,16,6,21)
      CALL PVDEF (BDLA,0,0,1,24)
      CALL PVDEF (BEOU,0,0,0,26)
      CALL PVDEF (BELA,0,0,1,26)
      CALL PVDEF (BETE,0,0,2,26)
      CALL PVDEF (BEBL,0,0,3,26)
C
C-----DEFINIERE DIE PROZESS-VARIABLEN FÜR DAS SAM-MODUL
C     (KOMMUNIKATIONS-MODUL).
C
C     TRBL = READ DATA BLOCK     TSMR = SETZE LAM MASK-REGISTER
C     TERE = ENABLE RECEIVER     TCLR = CLEAR LAM RECEIVER
C     TWBL = WRITE DATA BLOCK    TCLT = CLEAR LAM TRANSMITTER
C     TETR = ENABLE TRANSMITTER
C
      CALL PVDEF (TRBL,-1,18,0,0,5,45,0,13,1,1)
      CALL PVDEF (TERE,-1,18,0,26)
      CALL PVDEF (TWBL,-1,18,1,16,5,29,0,0,2)
      CALL PVDEF (TETR,-1,18,1,26)
      CALL PVDEF (TSMR,-1,18,13,19,1)
      CALL PVDEF (TCLT,-1,18,12,23,0,0,1,0,2)
      CALL PVDEF (TCLR,-1,18,12,23,0,0,2,1,1)
C
C-----DEFINIERE CRATE- UND STATIONS-NUMMERN DER BTR-MODULE.
C
      CALL MGDEF (1,0,10,15)
      CALL MGDEF (2,0,20,6)
C
C-----DEFINIERE DIE LAM'S FÜR DIE SYNCHRONISIERUNG DER EIN-
      UND AUSGABE DER DIALOG-DATEN UND DAS LAM DES BTR-MODULS.
C
      CALL LAMDEF (128,TCLT,0)
      CALL LAMDEF (129,TCLR,0)
      CALL LAMDEF (130,BDLA,0)
C
C-----DEFINIERE DIE TASK FÜR DEN DIALOG, DIE GESTARTET WERDEN
C     SOLL, WENN EINE EINGABE BEENDET IST UND DIE TASK ZUM
C     NACHFÜLLEN DES BTR-MODULS.
```

```
C
      CALL CON (TDIAL,129,M,20)
      CALL CON (TBTR,130,M,30)
C
C
C-----STARTE DIE DIALOG-TASK, DAMIT DIESE SICH AUF DEM TERMINAL
C     MELDEN UND EINEN LESEAUFTRAG AM TREIBER ABGEBEN KANN.
C
      CALL PARA (1,129,0,1)
      CALL START (TDIAL,0,0,M,10)
      CALL STOP
C
      END
```

5.3.3 Dialogtask

```
C     TASK-NAME: TDIAL
C     STAUS:     TRANSIENT
C     PRIORITÄT: 20
C
C     TASK-BESCHREIBUNG:
C
C     DIESE TASK INITIALISIERT DAS SAM-MODUL, STEUERT DEN DIA-
C     LOG UND SCHALTET DAS VERLANGTE BTR-MODUL AUF TEST.
C
      LOGICAL INBUF, L1
      DOUBLE PRECISION MESS (6)
C
      COMMON /PVACOM/ BWST,BWBL,BDLA,BEOU,BELA,BETE,BEBL,
     1                BRBL,TERE,TWBL,TETR,TSMR,TCLT,TCLR
C
C-----DER COMMON /DIACOM/ MUSS WIE DER /PVACOM/ RESIDENT SEIN,
C     WEIL DIE TASK TRANSIENT IST UND LOKALE BUFFER BEI JEDEM
C     START ODER VON ANDEREN TASK, DIE IM GLEICHEN LAUFBEREICH
C     ARBEITEN, ÜBERSCHRIEBEN WERDEN. DIE GRÖSSE DES /DIACOM/
C     IST FÜR EINE ERWEITERUNG BIS AUF 10 TERMINALS AUSGELEGT.
C
      COMMON/DIACOM/INBUF (5,10), IRPV (2,10)
C
C-----L1 (LOGICAL) UND I1 (INTEGER) WERDEN FÜR DIE TEXTVERAR-
C     BEITUNG BENUTZT.
C
      EQUIVALENCE (L1,I1)
C
C-----ZUWEISUNG DER DIALOG-MELDUNGEN.
C
      DATA MESS(1), MESS(2) /'TEST BTR', 'NR:'/
      DATA MESS(3), MESS(4) /'EINGABE ', 'FEHLER?'/
      DATA MESS(5), MESS(6) /'BTR DEFE', 'KT'/
C
C-----CODE FÜR CR (13) LF (10) = 10  2560 + 13 (2 Byte)
C
```

```
      ICRLF = 2573
C
C-----ÜBERNEHME PARAMETER AUS DEM FIFO-BUFFER, DER ZWEITE
C     PARAMETER (K) IST GLEICH NULL, WENN DIE TASK VOM CAMAC,
C     TREIBER GESTARTET WURDE. DER ERSTE PARAMETER IST HIER
C     OHNE BEDEUTUNG, WEIL CRATE- UND STATIONS-NUMMER DES
C     SAM-MODULS EINDEUTIG IN DER PROZESS-VARIABLEN DEFINIERT
C     SIND UND NICHT IN DER MG-TABELLE.
C
      CALL PARA (2,129,MG,K)
C??   START VOM CAMAC-TREIBER ??
      IF (K.EQ.0) GOTO 120
C
C-----START VON DER TASK "TINIT". INITIALISIERE DAS SAM-MODUL.
C
      CALL MOVE (TERE,0,IR)
C??   RICHTIGEN REPLY VOM SAM-MODUL ??
      IF (IR.EQ.1) GOTO 110
      WRITE (0,20)
   20 FORMAT (' SAM MODUL IST DEFEKT')
      STOP
C
  110 CALL MOVE (TETR)
      CALL MOVE (TSMR,0,3)
      GOTO 300
C
C-----START VOM CAMAC TREIBER
C
C??   KEIN ÜBERTRAGUNGS-FEHLER??
  120 IF (IRPV(1,1).EQ.0) GOTO 130
      WRITE (0,10)
   10 FORMAT (' ÜBERTRAGUNGS-FEHLER')
      STOP
C
C-----UNTERSUCHE DEN EINGABE-STRING UND BESTIMME DIE BTR-
C     NUMMER (MA).
C
  130 MA = 0
```

```
C      LÄNGE DES EINGABE-STRINGS
       J = IRPV (2,1)
C
       DO 200 I = 1,J
       L1 = INBUF (I,1)
C   ?? CR (13) ? ?
       IF (I1.EQ.13) GOTO 220
C   ?? KEINE ZIFFER (48-57) ? ?
       IF (I1.LT.48 .OR. I1.GE.58) GOTO 230
       MA = 10  MA + I1 - 48
  200 CONTINUE
C
C   ?? MA IM RICHTIGEN BEREICH (1-10) ??
  220 IF (MA.GE.1 .AND. MA.LE.10) GOTO 240
C      GEBE FEHLERMELDUNG "EINGABE FEHLER" AUS
  230 CALL MOVE (TWBL,0,ICRLF,2)
       CALL MOVE (TWBL,0,MESS(3),15)
       GOTO 300
C
C-----INITIALISIERE DAS BETREFFENDE BTR-MODUL FÜR TESTBETRIEB
C
  240 CALL MOVE (BWST,MA,10,IR)
C   ?? RICHTIGE RÜCKMELDUNG VOM BTR-MODUL ??
       IF (IR.EQ.0) GOTO 260
C      GEBE FEHLERMELDUNG "BTR DEFEKT" AUS
       CALL MOVE (TWBL,0,ICRLF,2)
       CALL MOVE (TWBL,0,MESS(5),1o)
       GOTO 300
C
  260 CALL MOVE (BELA,MA)
       CALL MOVE (BEOU,MA)
       CALL MOVE (BETE,MA)
       CALL MOVE (BEBL,MA)
C
C      GEBE BEREITMELDUNG "TEST BTR NR: " AUS
  300 CALL MOVE (TWBL,0,OCRLF,2)
       CALL MOVE (TWBL,0,MESS(1),13)
       CALL MOVE (TRBL,0,INBUF,3,IRPV(1,1))
```

```
C
C-----WARTE BIS LETZTE AUSGABE BEENDET IST. DER AUSGABE-BUFFER
C     KÖNNTE VON EINER ANDEREN TASK, DIE IM SELBEN LAUFBEREICH
C     LÄUFT, ÜBERSCHRIEBEN WERDEN!
C
      CALL CHANEL (0,TWBL)
      CALL STOP
C
      END
```

5.3.4 BTR-Nachfülltask

```
C     TASK-NAME  : TBTR
C     STATUS:    : RESIDENT
C     PRIORITÄT  : 30
C
C     TASK-BESCHREIBUNG:
C
C     DIESE TASK WIRD VOM BTR-MODUL GESTARTET UND GIBT AN DAS
C     BTR-MODUL EINE BYTE-DATENRAMPE AUS.
C
      INTEGER BUF (64)
C
      COMMON /PVACOM/ BWST,BWBL,BDLA,BEOU,BELA,BETE,BEBL,
     1                TRBL,TERE,TWBL,TETR,TSMR,TCLT,TCLR
C
C-----GENERIERE EINE BYTE-DATENRAMPE VON 0 - 127
C
      DO 200 I = 1,64
      BUF (I) = 514*(I - 1) + 256
  200 CONTINUE
C
C-----ÜBERNEHME DIE TASK-PARAMETER AUS DEM FIFO-BUFFER. DA
C     CRATE- UND STATIONS-NUMMERN DER BTR-MODULE IN DER MG-
C     TABELLE DEFINIERT SIND, ÜBERGIBT DER CAMAC-TREIBER IM
C     ERSTEN PARAMETER DIE ENTSPRECHENDE MODULGRUPPEN-NUMMER
C     (MA).
C
      CALL PARA (2,130,MA)
      CALL MOVE (BWBL,MA,BUF,64)
C
C-----WENN DER TREIBER EIN LAM ERKENNT? DISABLED ER DIESES AUTO-
C     MATISCH, DAS JETZT WIEDER ENABLED WERDEN MUSS.
C
      CALL MOVE (BELA,MA)
      CALL STOP
C
      END
```

5.4 Kurzbeschreibung der CAMAC-Module

Im Programmbeispiel wurden verschiedene CAMAC-Module verwendet. Um den Programmablauf besser zu verstehen, folgt eine kurze Beschreibung der verwendeten Modulfunktionen und der wichtigsten Registerbelegungen.

5.4.1 Serial Crate Controller (Type 3952 von Kinetic)

Befehlsliste

F(1) . A(Ø) Statusregister lesen
F(17). A(Ø) Statusregister schreiben
F(19). A(Ø) Statusregister selektiv setzen
F(23). A(Ø) Statusregister selektiv löschen

Bitbelegung vom Statusregister

Bit 1 Initialisieren (Z)
Bit 2 Löschen (C)
Bit 5 DSX, letzter X-Response
Bit 6 DSQ, letzter Q-Response
Bit 9 Interrupt zulassen
Bit 12 Bypass
Bit 13 Datenweg abgeschaltet

5.4.2 SAM 2, Serielles-Asynchrones CAMAC-Modul

Befehlsliste

F(Ø)	A(Ø)	Empfänger, Zeichen lesen
F(24)	A(Ø)	Empfänger sperren
F(26)	A(Ø)	Empfänger freigeben
F(16)	A(1)	Sender, Zeichen ausgeben
F(24)	A(1)	Sender sperren
F(26)	A(1)	Sender freigeben
F(19)	A(13)	LAM-Maskenregister selektiv setzen
F(23)	A(13)	LAM-Maskenregister selektiv löschen
F(23)	A(12)	LAM-Register selektiv löschen

Bitbelegung vom LAM-Register

Bit 1	Sendebuffer leer
Bit 2	Empfängerbuffer voll

Bitbelegung vom LAM-Maskenregister

Bit 1	LAM-Bit 1 freigeben
Bit 2	LAM-Bit 2 freigeben