KERNFORSCHUNGSZENTRUM

KARLSRUHE

Juni 1971

KFK 1394

Datenverarbeitungszentrale

Institut für Datenverarbeitung in der Technik

TCP – Terminal Control Program – ein Teilnehmersystem für das IBM System/360

R. Bader, G. Fleck, V. Haase, E. Holler, S. Müller, H. Santo,
M. Schayegh, F. Sell, W. Strolz, U. Ullrich, K. Weiß

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.

KARLSRUHE

GESELLSCHAFT FÜR KERNFORSCHUNG M.B.H.
KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

Juni 1971 KfK 1394

Datenverarbeitungszentrale
und
Institut für Datenverarbeitung in
der Technik

TCP
Terminal Control Program -
ein Teilnehmersystem für
das IBM System/360

von

R.Bader, G.Fleck, V.Haase, E.Holler, S.Müller, H.Santo,
M.Schayegh, F.Sell, W.Strolz, U.Ullrich, K.Weiß

Gesellschaft für Kernforschung m.b.H., Karlsruhe

Kurzfassung

Im Kernforschungszentrum Karlsruhe wurde das Teilnehmersystem TCP entwickelt und auf der Datenverarbeitungsanlage IBM/360-65 implementiert. Die von TCP unterstützten Terminals bestehen aus der Datenstation SIEMENS TRANSDATA 8525-101 und selbst entwickelten Datensichtgeräten.

Die TCP-Software bietet dem Terminal-Benutzer mittels einer dialogartigen Kommando-Sprache fünf Funktionskategorien: Auskunftsdienste, Datenmanipulation, Starten von Jobs, Starten von benutzerspezifischen Programmfunktionen sowie alphanumerische und graphische Ausgabe auf den Sichtgeräten.

Der Bericht gibt einen Überblick über den inneren Aufbau von TCP. Die modular aufgebauten Systemkomponenten Initialisierung, Systemüberwachung, Ein- und Ausgabesteuerung, Kommandoentschlüsslung, Bibliotheksverwaltung, TCP-Jobmanagement und TCP-Graphik werden in der Form eines "Program Logic Manual" eingehend beschrieben.

Abstract

The time-sharing-system TCP (Terminal Control Program) was developped at the Karlsruhe Nuclear-Research Center and was implemented on the central computer IBM/360-65. The terminals supported by TCP consist of the data stations SIEMENS TRANSDATA 8525-101 and of own made displays.

The TCP software offers five functional classes to the terminal user via a dialogue - based command language: general information service, conversational file handling, remote job entry, user-specific program functions, alphanumeric and graphic display.

This publication provides a surview about the internal structure of TCP. The modular system components: Initialisation, System Supervision, I/O-Control, Command Interpreter, Library Management, TCP-Job-Management and TCP-Graphic are described in detail in the form of a "Program Logic Manual".

Inhalt:

		Seite
I.	Systemübersicht (E. Holler)	1
	1. Aufgaben und Funktionen von TCP	1
	2. Anschluß von TCP an das Betriebssystem der IBM /360-65	2
	3. Die TCP-Komponenten und ihre Funktionen	4
	3.1. Die Startphase	4
	3.2. TCP - Betriebsphase	7
	3.3. TCP - Abschaltphase	10
II.	Initialisierung (W. Strolz, K. Weiß)	11
	1. TCP1BEGN	11
	2. TCP1INIT	12
	2.1. CSECT TCP1INIT	13
	2.2. CSECT TCP1TSUP	14
	3. TCP1BASE	17
	4. Initialisierung der DCB's für die Bibliotheken und die Spool-Datasets	19
	5. TCP1DSUP	19
	5.1. TCP1DTAB	24
III.	Systemüberwachung (F. Sell)	24
	1. Aktivitäten des OS	24
	2. Aktivitäten des TCP	26
	2.1. TCP1BASE, CSECT TCP1OPCP (Operator Command Processing)	26
	2.2. Nachrichten des Operateurs an TCP-Benutzer	27
	2.3. Überprüfen, welche Benutzer aktiv sind	29
	2.4. Ablaufsteuerung des TCP	29

Seite

3. Querverbindungen zwischen OS und TCP 31
3.1. Anwendung in TCPWTR 31
3.2. Anwendung in Assembler-Jobs 32
3.3. Anwendung in FORTRAN-Jobs 32
4. Benutzerprogramm für Kommunikation von Terminalbenutzern mit der Systemkonsole 32
5. Überwachung der Terminalsitzungen 33

IV. Ein-und Ausgabesteuerung der Datenstationen (E. Holler, S. Müller, H. Santo) 35

1. TCP1WAIT 36
2. OUTSPOOL/ENDPUT 41
2.1. OUTSPOOL 44
2.2. ENDPUT 53
3. TCP1SPOL 54
4. TCP1TSDN 82

V. Kommandoentschlüsselung (V. Haase, U. Ullrich) 84

1. PROC 84
2. STEFAN 86
3. MESSAGE 88
4. ATER 88

VI. Bibliotheksverwaltung (S. Müller, H. Santo, M. Schayegh) 89

1. TCP1ERK 89
2. TCP1ZUS 92
3. TCP1LIE2 94
4. TCP1AEN2 96
4.1. TCP1INP1 99
5. TCP1MODR 99

Seite

6. TCP1SCH 106
7. TCP1TS 107
8. TCP1LGES 108
9. TCP1LREC 109
10. TCP1LOUT 116
11. TCP1KOP 119
12. TCP1DMOD 121
13. TCP1HOL 123
14. TCP1BC20 (KETTE) 125
15. COMPRESS 127

VII. TCP-Jobmanagement (H. Santo, F. Sell) 136

1. Möglichkeiten 136
2. Bearbeitung durch das OS 136
   2.1. Einleseprozedur TCPRDR 136
   2.2. Ausgabeprozedur TCPWTR 137
3. Programm für das Starten von Benutzerprogrammen 143
4. Blitzstart 143

VIII. TCP-Graphik (R. Bader, G. Fleck,
M. Schayegh, W. Strolz) 146

1. DKP 146
2. TCPTAUS 148
   2.1. CANCANAS 151
3. TCPKAUS 152
   3.1. DKPUM 153
   3.2. CANCAN 157
   3.3. DKPV1 159
   3.4. DKPV2 164
   3.5. DKPV3 165
4. TCPPAUS 165
   4.1. PLOTA1, PLOTA2, PLOTA3 168
5. TCPDAUS 172

Seite

Anhang A. Beschreibung des Terminals
(G. Fleck, W. Strolz) 175

A.1. Die Dialogstation
SIEMENS-TRANSDATA
8525-101 175

A.2. Das Display-System
DASIKA /360-B 177

Anhang B. Modul-Kurzbeschreibung (W. Strolz) 180

Anhang C. TCP-Makros und-Kontrollblöcke (W. Strolz) 227

Anhang D. Spezielle Blitzstartprogramme (H. Santo) 260
D.1. XIAK12 261
D.2. XIAK24 267
D.3. XCAE 268

Anhang E. TCP unter ASP (W. Strolz) 274

Anhang F. Abbildungen 281

Anhang G. Flußdiagramme 292

Anhang H. Literaturstellen 408

I. Systemübersicht

1. Aufgaben und Funktionen von TCP (Terminal Control Program)

(E. Holler)

TCP ist ein für das System IBM/360 konzipiertes Teilnehmersystem /1,2/, das auf einer Rechenanlage IBM/360-65 implementiert ist. Als Teilnehmersystem hat TCP die Aufgabe, einer Vielzahl von Benutzern gleichzeitig einen möglichst großen Anteil der im System/360 überhaupt existierenden Möglichkeiten /3/ zur Verfügung zu stellen. Der direkte Zugriff der Teilnehmer zu den Dienstleistungen des zentralen Großrechners wird durch eine größere Anzahl flexibel ausgestatteter Datenstationen realisiert. Die Datenstationen sind standardmäßig mit Tastatur, Drucker und einem Bildschirmgerät mit Speicherröhre ausgerüstet; zusätzlich gibt es die Möglichkeit der Lochstreifen Ein- und Ausgabe. Die Verbindung der Datenstationen mit der Rechenzentrale erfolgt über ein Netz festinstallierter Standleitungen. Die Steuerung der Kommunikation zwischen Teilnehmern und zentralem System über die Datenstationen ist eine der Grundfunktionen von TCP. Die Vielfalt der elementaren Systemdienste, die das System/360 bietet, wurden innerhalb des TCP zum Aufbau von Programmkomplexen verwendet, deren Leistungen den Benutzern von TCP über eine fünf Funktionskategorien umfassende Kommandosprache zur Verfügung gestellt werden. Diese Funktionskategorien sind im einzelnen:

a) Allgemeine Dienste, wie Auskünfte über die Formulierung und die Funktionen von Kommandos, Zustand der Bibliothek und die Abfrage sonstiger Teilnehmerinformationen.

b) Manipulation von Programmen und Daten in den Teilnehmerbibliotheken.

c) Starten von Programmen, die sich in der Privatbibliothek der Teilnehmer befinden vom Terminal aus im Jobverarbeitungsverfahren sowie die Anforderung von Programmresultaten.

d) Alphanumerische und graphische Ausgabe auf den Sichtgeräten.

e) Auslösung spezieller Programmfunktionen für bestimmte Teilnehmerkreise. Die entsprechenden Programme werden on-line bearbeitet. Nahezu beliebige Spezialfunktionen können so bei Bedarf implementiert werden.

Zu den Kommandos für die Auslösung der oben bezeichneten Funktionen kommen noch Befehle für die Kommunikationssteuerung (Eröffnung und Beendigung von Terminalsitzungen, Selektion von Geräten an den Datenstationen) hinzu. Die Erteilung von Aufträgen vom Teilnehmer an das zentrale System erfolgt über die funktionsauslösenden Kommandos innerhalb von Kommunikationszyklen, von denen sich beliebig viele zu einer Terminalsitzung des Teilnehmers aneinanderreihen können. Die Terminalsitzungen selbst werden durch ein "Logon" - Kommando unter Angabe einer spezifizierten Teilnehmeridentifikation eröffnet und durch ein "Logoff"-Kommando beendet. Während der Dauer einer Terminalsitzung wird vom TCP die Belastung des Systems durch den speziellen Benutzer erfaßt und damit einer statistischen Erfassung zugänglich gemacht.

2. Anschluß von TCP an das Betriebssystem der IBM /360/65 (E. Holler)

TCP ist für Multiprogramming-Umgebung des Betriebssystems OS-MVT konzipiert /4/. Das Multitasking-Konzept des MVT ermöglicht die Aufteilung des TCP-Softwaresystems in funktionell verschiedene Programmeinheiten, die unter der Kontrolle je einer Task stehen. TCP benutzt die Systemfunktionen im Taskmanagement des MVT, um dynamisch Tasks zu kreieren (etwa zur Bearbeitung spezieller Teilnehmeraufträge) und wieder zu beenden. Die durch den Task-Supervisor des MVT gegebenen Möglichkeiten der Task-Synchronisation (WAIT-POST) werden vom TCP zur Kommunikation zwischen verschiedenen Tasks und zur Steuerung von Programmabläufen durch externe Ereignisse (Events), etwa durch Ein- oder Ausgabevorgänge bedingt, angewendet. Asynchrone Exit-Routinen und die automatische Terminierung von Tasks in der unteren Taskhierarchie bei fehlerhaftem Programmverlauf tragen zu einem verbesserten Störverhalten der TCP-Software bei.

Mit Hilfe der Hauptspeicher-Inhaltsverwaltung (Contents Supervision) des MVT ist die dynamische Zuladung und Aktivierung von Programmen unter der Kontrolle existierender oder neu zu kreierender Tasks möglich. Selten oder nur in Ausnahmesituationen benötigte Programmteile müssen so nicht ständig resident gehalten werden, solange TCP aktiv ist, sondern können bei Bedarf geladen werden.

Die dynamische Zuordnung von Arbeitsspeicherbereichen innerhalb der TCP-Tasks bedient sich der Kernspeicherverwaltung (Main Storage Supervision) des OS-MVT.Arbeitsbereiche im Hauptspeicher werden für Ein- und Ausgabepufferung, Datenmanipulation und die Anlage von Kontrollblöcken und Parameterfeldern benötigt.

Schließlich sei noch die Zeitüberwachung im OS-MVT erwähnt, die vom TCP zur Erfassung der Teilnehmer-Benutzungszeiten eingesetzt wird.

TCP kann innerhalb der Umgebung des OS-MVT als Job unter Kontrolle einer sogenannten Job-Step-Task betrieben werden und unterscheidet sich in diesem Falle nicht von den übrigen Benutzerprogrammen. Um jedoch eine Kommunikation zwischen beliebigen anderen Tasks und den TCP-Tasks zu ermöglichen, muß TCP unter der Kontrolle einer System-Task aktiv sein. Denn für jede System-Task wird vom Master Scheduler des OS-MVT ein sog. Command Scheduling Control Block (CSCB) in die CSCB-Chain des Systems im Supervisor-Queue-Space (residenter Arbeitsbereich im Hauptspeicher, der nur vom Supervisor verwendet wird) eingereiht. Diese CSCB's sind Kommunikationselemente, die über die Communication-Vector-Table (CVT) des Systems zugänglich sind und Zeiger zu den für eine Inter-Task-Verständigung notwendigen Ereignisvariablen und Nachrichtenpuffern beinhalten. Über den CSCB wird damit, wenn TCP als System-Task aktiviert wird, eine Kommunikation zwischen System-Operator und TCP sowie zwischen beliebigen anderen Tasks im System und TCP möglich. Die Aktivierung von TCP als System Task erfolgt über den Operator-Start-Command an den Master-Scheduler des OS-MVT.

Beim Start von TCP als Job oder System-Task erfolgt die Festlegung der jeweils aktuellen Konfiguration (Zahl der unterstützten Terminals usw.) über eine JCL-Prozedur. Über Prozedurparameter (EXEC-Parameter) kann außerdem Einfluß auf die Zahl der im Pufferbereich für die Terminal-Ein- und Ausgabe zur Verfügung stehenden Puffer und deren Größe genommen werden.

Das Datenmanagement innerhalb des TCP wird mit Hilfe der vom Betriebssystem bereitgestellten Zugriffsmethoden vorgenommen: die sequentiellen Zugriffsmethoden BPAM, BSAM und QSAM finden bei Dateimanipulationen Anwendung während die Ein- und Ausgabe an den Datenstationen über die zusätzlich an die Siemensterminals 8525 angepaßte Basic Telecommunications Access Method (BTAM) erfolgt.

TCP ermöglicht von den Datenstationen aus den Start von Programmen und deren Bearbeitung im Jobverarbeitungsverfahren. Die für das Jobverarbeitungsverfahren benötigten Steuerinformationen (Job Control Language oder JCL) sind dazu vom Teilnehmer in seiner Bibliothek bereitzustellen. Die Einreihung des Jobs in die System-Eingangswarteschlange wird von einer Systemroutine (Reader-Interpreter), die bei Bedarf unter der Kontrolle einer Systemtask aktiviert wird, vorgenommen. Dazu liest diese Routine die bereitgestellte Steuerinformation, interpretiert sie, und baut in der Eingangswarteschlange entsprechende Kontrollblöcke auf. Eine weitere Systemroutine, ein sogenannter Output-Writer), hat die Aufgabe, nach erfolgter Jobbearbeitung die Programmresultate für den Teilnehmer, der den Job gestartet hat, bereitzustellen und ihn entsprechend zu informieren.

3. Die TCP-Komponenten und ihre Funktionen

3.1. Die Startphase (G. Fleck, E. Holler, W. Strolz)

Die Startphase im TCP dient dem Aufbau der während der Betriebsphase benötigten Kontrollblöcke, des Pufferbereichs (Bufferpool), dem Test der Betriebsbereitschaft der Datenstationen und der Aktivierung der für die Überwachung der Ein- und Ausgabevorgänge und die Steuerung der Kommandobearbeitung verantwortlichen Tasks.

Weiter werden während des Initialisierungsvorganges die von TCP ständig benötigten Bibliotheken auf den Plattenspeichern eröffnet. Zu den Kontrollblöcken, die im TCP benötigt werden, gehören die Blöcke des Datamanagement (DCB, IOB, DECB, DEB), ferner Event-Control-Blöcke zur Ereigniserfassung, ein Buffer-Control-Block zur Überwachung des Buffer-Pools und spezielle TCP-Kontrollblöcke. Jeder Datenstation, die während des zu initialisierenden TCP-Betriebs bedient werden soll, ist ein Kontrollblock, die sogenannte Linebox, zugeordnet, in der der jeweilige Status der speziellen Terminalaktivität überwacht wird. Die Linebox beinhaltet außerdem den DCB für die Teilnehmerbibliothek und den DECB zur Steuerung der Terminal-Ein- und Ausgabe. Zentrale Statusinformationen, Zeiger zu Adresslisten von Ereignisvariablen (ECB-Listen), die Adressen von Eingangspunkten zu zentralen TCP-Routinen, Selektionstabellen für die Datenstation usw. sind zur Konbox zusammengefaßt, einem Kontrollblock der im TCP nur einmal vorhanden ist und von allen Systemteilen im TCP über geeignete Parameterversorgung erreicht werden kann. Ein weiterer spezieller Kontrollblock des TCP ist die Displaybox, in der alle Kontrollinformationen zur Steuerung der Displayausgabe zusammengefaßt sind. Für jede Displaykontrolleinheit wird während der Initialisierung eine Displaybox aufgebaut.

Die Startphase beginnt mit der Aktivierung des Moduls TCP1BEGN, der unter der Kontrolle der obersten Task in der TCP-Taskhierarchie steht **Bild 1.** In diesem Modul wird die Task kreiert, die für den eigentlichen Initialisierungsvorgang und später für die Überwachung der Ein- und Ausgabevorgänge an den Terminals verantwortlich ist. TCP1BEGN selbst startet noch den TCPWRITER und wartet dann bis zur Deaktivierung der TCP-Taskhierarchie oder bis der Operateur mit einer speziellen Nachricht die Beendigung von TCP1BEGN erwirkt.

Die neu kreierte Subtask, im folgenden kurz als Task 2 bezeichnet, sorgt unter Verwendung von Modul TCP1INIT für den Aufbau der Lineboxes und die Eröffnung der I/O-Kontrollblöcke für die in der Start-JCL-Prozedur spezifizierten Terminals. Es folgt die Ausgabe einer Testnachricht an die Datenstationen; durch die Testnachricht werden die Terminalbenutzer gleichzeitig über den Start von TCP informiert. Damit der Initialisierungsprozess weiter fortgesetzt werden kann, muß mindestens ein Terminal funktionsbereit sein. Im anderen Falle wird sofort die Terminierung von TCP eingeleitet. Terminals, die nicht bereit sind, werden in der Linebox entsprechend markiert.

Nach erfolgreichem Ablauf wird TCP1INIT von Modul TCP1BASE überlagert; innerhalb von TCP1BASE wird in der gleichnamigen Controlsection die Initialisierung weitergeführt. Die von TCP1INIT aufgebaute Linebox-Chain wird mit der Konbox verkettet und ein Bufferpool gemäß den von TCP1BEGN weitergeleiteten Parametern aufgebaut. Es folgt der Aufbau der Listen mit den Adressen von Ereignisvariablen, über die während der Betriebsphase des TCP die TCP-Tasks für die Ein- und Ausgabesteuerung und die Kommandobearbeitung aktiviert werden. Der nächste Schritt in TCP1BASE ist die Eröffnung der ständig benötigten TCP-Bibliotheken durch einen dynamischen Aufruf von TCP1LIBN. Schließlich wird die Task kreiert, die für die Überwachung der Kommandoentschlüsselung und -bearbeitung verantwortlich ist. Es existieren nun insgesamt drei Tasks im TCP. Task 2 beendet nach der Kreierung von Task 3 ihre Initialisierungstätigkeit durch Übergabe der Kontrolle an die Controlsection TCP1WAIT innerhalb des Moduls TCP1BASE. Über TCP1WAIT steuert Task 2 die Terminal-Ein- und Ausgabe. Unter der Kontrolle der dritten Task wird nun in Modul TCPOPROC für jedes Terminal ein Puffer aquiriert und in den Puffern eine Nachricht für die Ausgabe an den Datenstationen bereitgestellt. Diese Nachricht zeigt den Teilnehmern den Bereitschaftszustand von TCP an. Durch Setzen der den funktionsbereiten Datenstationen zugeordneten Ereignisvariablen wird Task 2 von Task 3 mit der Ausgabe der Nachrichten an den Datenstationen beauftragt.

Für die Steuerung der Sichtgeräteausgabe an den Datenstationen ist eine vierte Task verantwortlich, die von Task 3 im Modul TCPOPROC vor Initialisierung der ersten Nachrichtenausgabe kreiert wird. Die für die Sichtgerätesteuerung notwendige Initialisierungsprozedur wird im Modul TCP1DSUP vorgenommen.

TCP1DSUP bereitet die Ausgabe von Daten an alle angeschlossenen Displaygeräte vor. Dazu wird zunächst die Anzahl der angeschlossenen Display Control Units (CU) festgestellt. Für jede dieser Control Units wird eine Displaybox (DBX) eröffnet, in der die E/A-Kontrollblöcke und das Kanalprogramm enthalten sind. Die KONBOX (KBX) zeigt auf den Anfang der DBX-Kette, die letzte DBX zeigt wieder auf die KBX. An Hand einer Tabelle der TCP1DTAB wird in jeder LINEBOX (LBX), zu der ein Displaygerät gehört, die Display-Adresse eingetragen. Die Startphase wird abgeschlossen mit der Ausgabe eines Testbildes an alle Sichtgeräte. Tritt bei der Bildausgabe ein Fehler auf, so werden alle LBX-Einträge der betreffenden CU wieder gelöscht. Ist kein CU "betriebsbereit", so erfolgt der TCP-Start ohne Display Option. Nach erfolgreichem Ablauf von TCP1DSUP geht die Kontrolle an das Display-Kontrollprogramm (DKP), das den laufenden Display-Betrieb überwacht.

3.2. TCP Betriebsphase (G. Fleck, V. Haase, W. Strolz)

Nach Abschluß der Initialisierung befinden sich alle dauernd in TCP vorhandenen Tasks (TCP1BASE, TCPOPROC, DKP) im Wartezustand. Ebenso wartet der TCPWTR auf Aufträge. Resident sind die Moduln TCP1BASE, TCPOPROC und DKP.

Wird an irgendeinem Terminal ein Benutzer aktiv, beginnt ein Kommunikationszyklus / 5 /, der wie folgt abläuft (im allgemeinen Fall sind bei TCP mehrere sich überlappende und asynchron zueinander liegende Kommunikationszyklen aktiv).

In der Zeit zwischen dem Drücken der "Anruf-Taste" und dem Betätigen der Taste "ETX" werden vom Modul TCP1BASE (CSECT = TCP1WAIT) Zeichen in einen Puffer des Pools übernommen (Auftragseingabe). Mit "ETX" ist die Eingabe beendet, ein POST auf den INECB der betreffenden LINEBOX verständigt TCPOPROC vom Vorliegen eines Auftrages.

TCPOPROC stellt mittels des Entschlüsslers (CSECT STEFAN) die Art des Auftrages fest und läßt ihn mittels eines Funktionsmodus (z.B. TCP1ZUS, TCP1KOP usw.), der geladen und "attached" wird, ausführen. (Nach Ausführung des "ATTACH" ist die Steuercsect PROC für den nächsten Auftrag frei). Nach Ausführung wird die Task-EXIT-Routine ATER angesprungen, in der die Ausführung überprüft und mittels der CSECT MESSAGE in einem dynamischen Puffer eine Antwort zur Ausgabe auf das Terminal zusammengestellt wird. POST auf den OUTECB verständigt TCP1WAIT vom Vorliegen einer Ausgabe. TCP1WAIT führt diese sodann aus. Bei der Ein- und Ausgabe von "Daten", die nicht interpretiert, sondern nur transportiert werden müssen, werden von TCP1BASE der Modul TCP1SPOL für den Transport Eingabepuffer Platte und die CSECT OUTSPOOL und ENDPUT in umgekehrter Richtung verwendet. Die Puffer für die Terminal E/A werden dynamisch verwaltet (bei der Eingabe durch eine PCI-Appendage gesteuert). Dieser allgemeine Ablaufplan wird bei einigen Sonderfunktionen durchbrochen:

Eröffnung und Abschluß einer Terminalsitzung (Modul TCP1ACNT) erfolgt synchron (LINK) im Ablauf von TCPOPROC. Der Start von Programmen (Übergabe an den Scheduler) erfolgt durch einen SVC 34 (= lies Kommando: STARTE TCPREADER). Der TCPREADER bringt Job Control Language in die SYS1.JOBQUE.

Die Kommunikation mit der Systemkonsole geschieht in der Richtung Benutzer Konsole durch WTO-Makros, in der Richtung Konsole Benutzer durch die Routine TCP1OPCP, die beim "Operatorkommando" # über einen Fehlerausgang im Operatorkommandoentschlüßler (IGC0503D) erreicht wird.

Etwaige laufende Ausgabe von Systemnachrichten wird mittels TCP1HALT abgebrochen (HIO). Auch Nachrichten anderer Jobs im System (z.B. TCPWTR) werden über diesen Weg verarbeitet, nachdem TCP über seinen CSCB gefunden wurde.

Displayausgabe wird über einen speziellen Kommunikationszyklus durchgeführt, der zwischen TCPØPROC und der Displaytask DKP abläuft. Das Displaykontrollprogramm wird mit POST DISPECB gestartet und meldet die Beendigung mittels POST DISPECB2 an TCPOPROC zurück.

Die Ablaufsteuerung der Display-Programme erfolgt durch das Display-Kontroll-Programm (DKP). Die anfallenden Aufgaben werden sequentiell abgearbeitet, d.h. alle Display-Programme sind "serially reusable".
Um einen Displaybefehl durchzuführen, werden die ausführenden Programme nacheinander gelinkt. Es sind dies:

1. Für das Einlesen und die Ausgabe von Text, von vorbereiteten Testbildern und zum Löschen des Bildschirmes
   TCPTAUS
2. Für graphische Bildaufbereitung und Ausgabe
   a) von Kurven und Punkten im linearen Maßstab
      TCPKAUS DKPUM
      DKPV1
   b) von Diagrammen und Skizzen
      TCPKAUS DKPUM
      DKPV2
   c) von Kurven und Punkten im logarithmischen Maßstab
      TCPKAUS DKPUM
      DKPV3
3. Für Plotausgaben am Display
   TCPPAUS PLOTA1
           PLOTA2
           PLOTA3

DKP stellt diesen Programmen einen Arbeitsspeicher von 16 K Bytes zur Verfügung. Die Kommunikation der Programme untereinander erfolgt über die KBX und die LBX. Der zuletzt ausgeführte Displaybefehl steht in der zugehörigen LBX und kann in einfacher Weise wiederholt werden. Am Ende jeder Bildaufbereitung wird von dem Assembler-Unterprogramm CANCAN eines der Kanalprogramme gestartet, die von TCP1DSUP aufgebaut wurden. Ist die letzte Bildausgabe zu Ende, so wird auf den nächsten Displaybefehl gewartet.

3.3. TCP - Abschaltphase (V. Haase)

Der Operator an der Systemkonsole kann TCP (mittels #-Kommando) in verschiedene Betriebszustände versetzen: LOGIN-Modus mit Protokollierung aller Logon/Logoff Aktivitäten, HOLD-Modus, in dem die Kommandos EIN (Eröffnen von Terminalsitzungen) und LIES (Eingabe) verboten sind. Intern wird dies durch Statusbits in der KONBOX angezeigt, die vom Entschlüßler (STEFAN) und vom Logon-Modul (TCP1ACNT) geprüft werden. Auf die gleiche Weise versetzt der Operator TCP in den SHUTDOWN-Zustand. Während jedes Kommunikationszyklus prüft TCPOPROC (PROC) das SHUT-bit in der KONBOX. Wenn es gesetzt ist, wird DKP mittels POST auf den DISPECB, POSTCODE = 0 davon verständigt. Nach RETURN des DKP wird es"detached"und TCPOPROC wird beendet. Diese Beendigung wird von TCP1BASE festgestellt, die Task TCPOPROC"detached"und die Kontrolle an den Modul TCP1TSDN via LINK übergeben. TCP1TSDN beendet etwaige laufende E/A-Vorgänge (HIO), schließt die Spool (Puffer) Datasets und veranlaßt die Ausgabe der E/A-Fehlerstatistik an der Konsole (Line Error Print). Nach der Beendigung dieses Moduls wird auch TCP1BASE beendet. Über TCP1BEGN erfolgt schließlich die Rückkehr zum OS.

II. Initialisierung

1. TCP1BEGN (W. Strolz)

TCP1BEGN stellt den ersten Initialisierungsmodul von TCP dar.
Es wird von OS-Routinen durch ATTACH aufgerufen.

FUNKTION

TCP1BEGN gibt ein OPEN auf den TCP.LINKLIB-DCB, setzt den Protection Key auf 1, falls TCP als Job läuft und ruft TCP1INIT mit ATTACH auf. TCP1BEGN wird beendet durch:

Operator-Reply 'ENDTCP' oder
durch Beendigung von TCP1BASE

Zuvor wird der TCB der aufgerufenden Task TCP1INIT mit DETACH aufgeräumt und der von TCP1BASE gestartete TCPWTR gestoppt. Anschließend RETURN zum OS-Tasksupervisor.

EINGANGSPARAMETER

R1: Zeigt auf ein Feld im Command Scheduling Control Block (CSCB).
Dieses Feld enthält den "Pointer To Procedure EXEC Statement PARM Field". Besteht das linke Bit des 1. Bytes dieses Feldes aus einer Null, dann gehört der CSCB zu einer SYSTEM TASK /6/ .

PROGRAMMABLAUF

TCP1BEGN rettet zunächst den vom System übergebenen Parameter-Pointer (Reg 1). Wurde TCP als System Task gestartet, dann wird mittelts SVC 254 der Protection Key auf 1 gesetzt, um die Gewähr zu haben, daß TCP aufgrund eines eventuellen Programmfehlers

NOPKSET nicht OS-Teile überspeichern kann. Nachdem über die Communication Vector Table (CVT) der Pointer zum SYS1.SVCLIB-DCB gefunden wurde, lädt TCP1BEGN die folgenden BTAM-Routinen:

IGGO19 MS Buffer Management Routine (BTAM)
IGGO19 MB Channel End: Abnormal End Appendage (BTAM)
IGGO19 MC PCI - Appendage (BTAM)

Nach einem OPEN auf den TCP.LINKLIB-DCB erfolgt ein ATTACH auf TCP1INIT. Dabei werden Reg 1 (Parameter), die Adresse des Task-ECB und der Pointer zum TCP.LINKLIB-DCB übergeben. Die

LAB1 TCB-Adresse der neu kreierten Task wird gerettet. ENDECB wird auf Null gesetzt und die WTOR-Message "TO END TCP..." wird abgesetzt. Anschließend wird gewartet bis der Reply "ENDTCP" kommt oder bis der Task-ECB bei einem Fehler in der kreierten Task durch das OS Control Programm gepostet wird. Wurde ein Reply gegeben, der nicht korrekt ist, dann wird bei LAB1 das WTOR wiederholt. In allen anderen Fällen wird der TCB der

LAB2 kreierten Task mit DETACH abgeräumt und der TCPWTR mittels SVC 253 gestoppt. Die Kontrolle wird dann mit RETURN an das OS zurückgegeben.

LITERATUR Die von TCP1BEGN geladenen BTAM-Routinen IGGO19MS, IGGO19 MB und IGGO19 MC sind beschrieben und mit Flowcharts dokumentiert unter: / 7 / .

2. TCP1INIT (W. Strolz)

CSECT'S TCP1INIT
TCP1TSUP

ENTRY TCP1INIT via ATTACH von TCP1BEGN

EINGANGSPARAMETER
R1 genau wie zu Beginn von TCP1BEGN
R2 (RWK) zeigt auf offenen TCP.LINKLIB-DCB
(Library-DCB)

EXIT Bei Fehler - RETURN zum OS mit Completion Code 4
Normal - mit XCTL zu TCP1BASE; SUCB - und Library-DCB-Adresse werden übergeben.

FUNKTION

Die CSECT TCP1INIT baut den STARTUP CONTROL BLOCK (SUCB) auf, gibt den Command Input Buffer (CIB) in SP 255 zurück und bearbeitet die Eingabeparameter
(Pufferanzahl und - länge).
Die CSECT TCP1TSUP baut die I/O-Blöcke und die Lineboxes für die Terminals auf. Die in den DD-Statements beschriebenen Terminals werden durch Absenden einer Start-Nachricht getestet.
Dann XCTL zu TCP1BASE.

2.1. CSECT TCP1INIT

Zunächst holt TCP1INIT durch GETMAIN Speicherplatz für den SUCB und füllt den geholten Platz mit hexadezimalen Nullen. Die via Reg 1 übergebene CSCB-Adresse wird in den SUCB eingetragen. Sind EXEC-Parameter vorhanden (Normalfall) dann werden sie ebenfalls in den SUCB eingebracht.

FCIB — Ist TCP als System Task gestartet, dann werden Adresse und Länge des CIB festgestellt und der CIB mit FREEMAIN an den SP 255 zurückgegeben. Anschließend wird im Command Scheduling Contr(Block (CSCB) der CIB-Pointer auf Null gesetzt und mittels SVC 252 der Protection Key 1 eingesetzt.

CALLT — Die CSECT TCP1TSUP wird mittels CALL angesprungen.

RPOINT
ERETURN
PARM
XCTL — Nach der Rückkehr aus TCP1TSUP wird geprüft ob TCP1TSUP erfolgreich durchgelaufen wurde. Ist das nicht der Fall, erfolgt RETURN zum OS mit Completion Code 4. War TCP1TSUP dagegen in Ordnung, dann werden die Parameter Pufferanzahl und Pufferlänge in binärer Form in den SUCB eingefügt. Abschließend wird die Kontrolle an TCP1BASE übergeben mit den Parametern: SUCB - und Library-DCB-Adresse.

2.2. CSECT TCP1TSUP

Nach dem Retten der Adresse des Library-DCB beginnt der TIOT-Scan.

FTERM
WRAPUP
BDCBD
CUCBS
CUCBSC
DDEBCP
IGBINIT
GETLBX

Die Task Input Output Table (TIOT) wird nach DD - Statements mit dem DD-Name - Anfang "TSI" untersucht. Ist kein TIOT-Entry dieser Art vorhanden wird dies mittels WTO TCP30 dem Operator mitgeteilt und schließlich die Task mit ABEND 200 beendet. Findet man dagegen ein Statement mit DD-Name TSI,.., dann wird die Device Type Information geholt. Gelingt das nicht, dann wird dies in Reg 15 angezeigt und mittels WTO TCP 31 dem Operator bekannt gemacht. Anschließend ebenfalls ABEND 200. Ist dagegen Dev. Type Information vorhanden, dann beginnt der eigentliche TIOT-Scan, der den Zweck hat, sämtliche Terminal-UCB-Adressen und die Anzahl dieser UCB's zu ermitteln. Zum Verständnis dieses Scans ist es ratsam, sich den Aufbau der TIOT anzusehen / 8 /. Ist überhaupt kein Terminal - UCB zu finden, dann wird nach NULLUCB gesprungen und dort wie bei WRAPUP verfahren. Anschließend wird für je 2 DCB'S und DEB'S Kernspeicher geholt: für den SDCB (Simple Buffering DCB) und DDCB (Dynamic Buffering DCB) sowie für SDEB und DDEB. Anschließend werden diese Blöcke teilweise initialisiert. Nach einem GETMAIN für sämtliche Line-IOB's werden diese ebenfalls initialisiert. Genauso werden dann die Lineboxes (LBX) angelegt und initialisiert. Jede LBX enthält einen Pointer, der zur nächsten weist. Zu diesem Zeitpunkt der Initialisierung kann allerdings die Pointerkette noch nicht geschlossen werden, da die Konbox, die in diese Kette eingereiht werden soll, noch nicht existiert.

Die Adressen der 1. und der letzten LBX werden daher in den SUCB eingebracht. Die Line Error Blocks werden nun angelegt,

BLERB — initialisiert und mit SDCB und DDCB über Pointer verbunden.

BLERBA — Nachdem nun die für I/O-Vorgänge wichtigsten Blöcke angelegt sind, wird die Ausgabe der SAD- & ENABLE-Commands vorbereitet. Der SAD-Command schaltet einer Terminal-Line innerhalb der 2702 Transmission Control Unit ein bestimmtes Terminal Control Feature zu (bei TCP: SADONE-Command). Der ENABLE-Command bereitet die 2702 auf den Empfang weiterer Channel Commands vor.

SADENABL — Dazu werden die SAD- & ENABLE - Commands in die entsprechenden IOB's eingebracht und schließlich mit EXCP ausgegeben. Dann wird der Erfolg dieser I/O-Initialisierung abgewartet und schließlich wird dieser Vorgang für alle Lines wiederholt. Ist der Initialisierungsversuch bei allen Lines mißlungen,

ABOPEND — dann wird das WTO TCP 29 ausgegeben und die Initialisierung

LADEN — mit ABEND 200 abgebrochen. Wurde die Initialisierung von SAD- & ENABLE wenigstens teilweise erfolgreich abgeschlossen, dann wird mit BLDL das Laden von folgenden BTAM-Routinen vorbereitet und diese werden schließlich mit LOAD auch geladen.

IGGO19 MB - Channel End Appendage
IGGO19 MC - PCI Appendage
IGGO19 MS - Buffer Routine
IGGO19 MA - Read/Write Routine
IGGO19 MO - Device I/O Module (geänderter Modul für SIEMENS 8525)

ABTERM — Verläuft BLDL nicht erfolgreich, dann wird die TCP-Initialisierung mit ABEND 100 abgebrochen. Nach erfolgreichem Laden werden die Übersetzungstabellen EBCDIC → ISO7 (CCITT-Nr.5) und ISO7 → EBCDIC angelegt.

TIMEM — Mit dem Macro TIME wird Datum und Uhrzeit ermittelt, entschlüsselt und in die TCP-Start-Nachricht eingefügt. Die TIMEM-Routine ist so ausgelegt, daß sie bis zum Jahre 2000 funktioniert. Schaltjahre werden - sogar der Spezialfall im Jahre 2000 - berücksichtigt.

Im Anschluß an die TIME-Routine wird gewartet, um den Hardware-Einrichtungen in 2702 und SIEMENS-Terminals Zeit zu lassen, die SAD - & ENABLE- Commands auszuführen. Zur Vorbereitung der TCP-Start-Nachricht (Test-Message) wird diese Nachricht nach ISO7 übersetzt, Kontrollblöcke initialisiert und eine ECB-Liste angelegt, die die Adressen der Terminal ECB's (DECB's) enthält.

PRINT
SEXCP

Die bereits teilweise initialisierten Kontrollblöcke werden vervollständigt und schließlich die TCP-Start-Nachricht auf das 1. Terminal unter Verwendung der R/W-Routine ausgegeben. Wurde die R/W - Routine erfolgreich beendet, dann wird geprüft, ob es

POPENB
POPENC

sich um einen Error Retry gehandelt hat. Ist das der Fall, wird der Error Count reduziert und nach TWAIT gesprungen. Andernfalls wird die ECB-Adresse in die ECB-Liste eingefügt und ab-

GOIO

geprüft, ob es sich um die letzte LBX handelt. Ist das nicht der Fall, so wird der Vorgang für die nächste Line bei PRINT wiederholt. Sind alle Lines bedient, dann wird bei GOWAIT der letzte Entry in die ECB-Liste gekennzeichnet und dann nach TWAIT gesprungen. War dagegen nach SEXCP die R/W-Routine nicht erfor -

POPENA

derlich, so wird versucht, diese Line mit LOPEN nachträglich zu initialisieren. Schlägt auch dieser Versuch fehl, dann wird das

POPENA1

WTO TCP 32 ausgegeben und die Leitung als LINE DOWN gekennzeichnet, so daß sie nicht mehr angesteuert wird. Sind keine aktiven Leitungen mehr vorhanden, dann RETURN nach TCP1INIT mit Completion Code 4 (ERETOUR), andernfalls Sprung nach GOIO.

TWAIT
ECBTEST
ECB7F
NAKTEST

Hier wird die Beendigung des Schreibvorgangs abgewartet und der Completion Code getestet; ist dieser in Ordnung, dann werden die DECFLAGS geprüft. Sind sie ungleich Null erfolgt Sprung nach LDOWN. Andernfalls wird geprüft, ob die Selektion der Terminal-Komponenten erfolgreich war. War sie nicht erfolgreich und war auch keine negative Quittung zurückgekommen, dann ebenfalls Sprung nach LDOWN; auf eine negative Quittung (NAK) wird WTO

INTREQ

TCP 34 angegeben und nach NEXT gesprungen. Bei korrekter Selektion und positiver Quittung auf Textausgabe (ACK) ebenfalls Sprung nach NEXT.

RETRY — Andernfalls ist ein Retry zu machen, so daß nach Vorbereitung des Retry's zu SEXCP gesprungen wird. Sind dagegen alle 3 vorgesehenen Retries gemacht, dann erfolgt Sprung nach LDOWN.

TIMEOUT — Wurde bei ECBTEST festgestellt, daß der Schreibvorgang nicht korrekt zu Ende kam, wird eine Reihe von Einzelprüfungen durchlaufen, die im Falle eines Timeout zu einer Prüfung der Selektions (Adresse-) - Quittung führen; in allen anderen Fällen Sprung nach LDOWN. Ist die Selektionsquittung negativ, dann Sprung nach INTREQ, sonst nach LDOWN. Trat ein Timeout auf beim Lesen

ERRWTO — der Quittung auf "Write Address", dann wird Local Mode angezeigt, das WTO TCP 33 ausgegeben und nach NEXT gesprungen.

LDOWN — Line Down wird angezeigt, das WTO TCP 35 ausgegeben und die

ERETOUR — Zahl der aktiven Lines um eins reduziert. Sind keine aktiven Lines mehr vorhanden, dann RETURN nach TCPINIT mit Completion Code 4.

NEXT — Andernfalls werden die DECB-Fields auf Null gesetzt und, falls noch nicht alle I/O-Vorgänge abgeschlossen und überprüft sind, nach TWAIT gesprungen. Sind alle I/O's beendet, wird die ECB-

SCR — Liste abgeräumt und mit Completion Code Null nach TCP1INIT zurückgesprungen.

HINWEIS

Um die Initialisierungsphase zu verstehen, sollte man sich mit BTAM / 7 / und mit der speziellen Übertragungsprozedur für die SIEMENS TRANSDATA 8525-101 / 9 / befassen.

3. Modul TCP1BASE (W. Strolz)

CSECT — TCP1BASE

ENTRY — via XCTL von TCP1INIT
Entrypointer TCP1BASE

EINGANGSPARAMETER

R1: Start Up Control Block (SUCB) und sämtliche Blöcke auf die Felder des SUCB verweisen.

EXIT — Branch zur CSECT TCP1WAIT im Modul TCP1BASE
Bei Fehler: RETURN zum OS mit CC4

FUNKTION

KONBOX - und LINEBOX - Initialisierung wird durch Vervollständigung der Kettung beendet. Bufferpool und ECB-Listen werden aufgebaut. Der SUCB wird abgeräumt, TCP1LIBN mit LINK aufgerufen zur Library-Initialisierung und der Modul TCPOPROC mit ATTACH ins Leben gerufen. Der TCPWTR wird gestartet und schließlich Branch nach TCP1WAIT.

INTERNE LOGIK

TCP1BASE Von TCP1INIT wird mit XCTL die Kontrolle an TCP1BASE weitergegeben. Aus den im SUCB übergebenen EXEC-Parametern wird die Größe des Bufferpools berechnet und schließlich wird der Bufferpool mit BUILD aufgebaut.Die Bufferpool-Control-Block-Adresse wird in die Terminal DCB's eingetragen. Die KONBOX wird vervollständigt und über Pointer mit den LINEBOXES zu einem TCP-Kontroll-Block-Ring zusammengeschlossen.
Die Größe von ECBLISTIN und ECBLISTOUT wird berechnet und beide Listen werden dann aufgebaut. Danach wird ECBLISTOUT initialisiert. Dazu wird zuerst die Operator-ECB-Adresse in ECBLISTOUT

NOCSCB eingetragen (gefunden mittels CSCB-Scan). Läuft TCP dagegen als Job, wird statt dessen eine Dummy-ECB-Adresse eingetragen.

NEXTLBX1 Die Adresse von OUTECB's, DECB's

LASTLBX1 und vom TASKECB werden in die ECBLISTOUT eingesetzt und die Liste verdoppelt. Entsprechend werden die Adressen der INECB's und DISPLAY-ECB's in ECBLISTIN eingetragen und die Liste anschließend verdoppelt.

FREESUCB Der SUCB wird abgeräumt und danach erfolgt LINK zu TCP1LIBN wo Library-DCB's aufgebaut und Spool-DS-DCB's mit OPEN geöffnet werden. War der RETURN CODE ungleich Null, erfolgt RETURN zum OS-Task-Management mit Completion Code 4.

ATPROC War TCP1LIBN dagegen erfolgreich, dann wird mit ATTACH die Task TCPOPROC aufgerufen und deren TCP-Adresse in die Konbox eingetragen. Danach wird TCP1SPOL geladen und im CSCB wird TCP als aktiv gekennzeichnet. Danach wird der TCPWTR gestar-

WAITROUT tet und der SNAPDCB geöffnet. Abschließend BRANCH nach TCP1WAIT .

4. Initialisierung der DCB's für die Bibliotheken und die Spool-Datasets (V. Haase)

Modul: TCP1LIBN

ENTRY via LINK von TCP1BASE

EXIT RETURN zu TCP1BASE

5. TCP1DSUP (G. Fleck)

TCP1DSUP ist ein Assembler Programm, das zur Initialisierung des graphischen Supports benötigt wird. Es wird aufgerufen via ATTACH von TCPOPROC und übergibt nach normalem **Ablauf** die Kontrolle an DKP via XCTL.

FUNKTION

TCP1DSUP stellt an Hand der DD-Statements fest, wieviel CONTROL UNITS (CU's) für Display-Ausgaben vorhanden sind. Für jede CU wird eine DISPLAYBOX (DBX) erstellt, in der die E/A-Kontrollblöcke und das Kanalprogramm enthalten sind. Die Tabelle TCP1DTAB gibt Auskunft über die Zugehörigkeit von Terminal zu CU und Display-Gerät. Die Zuordnung erfolgt in der LINEBOX (LBX). Ein Testbild wird an alle Sichtgeräte ausgegeben.

EINGANGSPARAMETER

1. DD-Karten mit den Adressen in der Control Units
2. TCP1DTAB: Tabelle der Displaybenutzer
3. R1 enthält die Adresse der KONBOX
4. Dataset mit den Daten des Testbildes DSNAME = IDD

AUSGANGSPARAMETER

1. R1 enthält die Adresse einer PARMLIST, die die KONBOX-Adresse enthält.
2. DBX für jede CU
3. ECBLIST
4. DEB für jede CU

PROGRAMMABLAUF

TCP1DSUP Nach dem 'Housekeeping' wird die KBX-Adresse in das KBXREG geladen. Mit dem Makro EXTRACT wird die Adresse der TASK INPUT/OUTPUT TABLE (TIOT) gesucht und in das Register

ITERM RTIOT gebracht. Nun wird in der TIOT nach einem DDNAME gesucht, der mit 'DIS' beginnt. Wird kein solcher gefunden , so wird nach ERROR2 verzweigt. Mit dem Makro DEVTYPE wird das DD-Statement untersucht. Wird ein DEVICE-TYPE- ERROR festgestellt oder ein DUMMY DD-Statement entdeckt, so wird nach ERROR3 verzweigt. Nun

CUCBSA werden die UCB-Adressen gesucht und nacheinander nach USBADS gespeichert. Ein Zähler stellt fest wieviel solcher UCB's für

CUCBSC Display-Anschlüsse vorhanden sind. Wurde keine UCB-Adresse gefunden, so wird das Programm bei NULLUCB fortgesetzt. Im Normalfall wird jetzt Kernspeicher angefordert für die erste DBX. Ein Zeiger zu dieser DBX wird in der KBX abgelegt. Dann wird der laufende

DBXINIT Speicherschutzschlüssel gerettet und mittels der SVC 255 auf Null gesetzt. Dies ist nötig, um einen Data Extent Block (DEB) im Subpool 252 anzulegen. Nach der OPEN-Routine für den DEB wird durch Aufruf des SVC 252 der Speicherschlüssel wieder zurückgesetzt. Anschließend wird der Input/Output Block (IOB) eröffnet. Ist eine weitere CU vorhanden, so wird Kernspeicher für die nächste DBX angefordert, ein Zeiger zur neuen DBX in der vorangegangenen DBX abgespeichert und nach DBXINIT verzweigt.

LASTDBX Die letzte DBX zeigt zur KBX. Danach wird ein Datenfeld aufgebaut, das zum Löschen der Bildschirme benötigt wird. Die Display-Tabelle TCP1DTAB wird geladen. Nun wird für die jeweils

TABLEUSE nächste CU in der Tabelle eine DBX gesucht.

SCANO — Ist keine zugehörige DBX vorhanden, so wird das Programm bei ERROR1 fortgesetzt.

UAFOUND — Ist die zugehörige DBX gefunden, so wird nach IOBCCW das Modell eines Kanalprogramms übertragen und mit aktuellen Adressen versehen.

SCAN2 — In der LINEBOX (LBX) werden alle zu dieser Unit gehörenden Unit-Adressen und Display-Adressen eingetragen. Ist das Ende der Display-Tabelle noch nicht erreicht, so wird nach

TABEND — SCANO verzweigt. Es wird Speicher angefordert für den Auf-

LOOP3 — bau der ECB-Liste. In dieser Liste werden die Adressen der ECB's gebracht, die zu dieser CU gehören. Die Daten des

ELISTEND — vorbereiteten Testbildes werden eingelesen. Das Kanalpro-

CCWSTART — gramm wird vervollständigt, d.h. Datenadresse und Count kommen ins Write-CCW. Das IOBSIOCC zeigt auf das Lösch-Kanalprogramm. Lösche-Byte und Control-Byte enthalten die Adresse des ersten Displaygerätes der CU. Das Displaygerät wird durch die ersten 3 Bits im Lösch-Byte bzw. Control-

CCWLP1 — Byte adressiert. Die Anzahl der zur CU gehörenden Display-geräte wird geladen. Die Adresse des aktuellen Displaygerätes wird in das Lösch-Byte und in das Control-Byte gebracht. Das Kanalprogramm wird gestartet. Bei einem Ausgabe-

LOOPW1

CCWEXCP — fehler wird das Programm bei RETRY fortgesetzt. Sind weitere

GOON — Displaygeräte an derselben CU, so wird die Adresse des nächsten errechnet und nach LOOPW1 verzweigt. Wurde gerade gelöscht, so wird die Adresse im IOBSIOCC auf das Ausgabe-

STARTX — Kanalprogramm gesetzt und nach CCWLP1 verzweigt. Nunmehr wurde an alle Sichtgeräte der laufenden CU ein Testbild ausgegeben. Jetzt wird geprüft, ob weitere CU's vorhanden sind. Ist dies der Fall,

STARTEND — so wird das Programm ab CCWSTART wiederholt. Nachdem an sämtliche Displaygeräte ein Testbild ausgegeben wurde, wird eine Parameterliste (PARMLIST) im Subpool 1 angelegt. Sie dient zur Übergabe der KBX- und LBX-Adresse an PL/1 Programme.

Die benötigten Arbeitsspeicher werden freigegeben und die Kontrolle geht via XCTL an das Display-Kontroll-Programm (DKP).

A54 Entsprechend der in RO stehenden Länge wird Kernspeicher im Subpool 1 zur Verfügung gestellt und gelöscht. Das Programm wird nach dem aufrufenden Statement fortgesetzt. Die Adresse

ERROR1 der Fehlernachricht MSG40 wird nach R1 geladen und zur Fehler-

ERRORTYPE Ausgabe nach ERRORTYPE gesprungen. Es erfolgt die Ausgabe der

ERETOUR Fehlernachricht. Freigabe der TCP1DTAB und RETURN. Es erfolgt

ERROR2 kein Display-Start. Eintrag der MSG41 bzw. der MSG42 und

ERROR3 Sprung nach ERRORTYPE. Ausgabe von MSG43. Die Einträge der

ERROR4 laufenden CU werden gelöscht. Ist eine weitere CU vorhanden, so wird das Programm bei STARTX fortgesetzt, sonst wird das

LBLOOP Programm über ERRORTYPE beendet.

RETRY Nach dem dritten Fehlversuch wird nach ERROR4 verzweigt. Die Fehler-Flags im IOB werden zurückgesetzt, es erfolgt der nächste Versuch bei CCWEXCP.

Bild 2 Display Zuordnungstabelle

HEX	DEZ	
0	0	Anzahl aller Displays
4	4	Anzahl der Displ. an 1.CU | Terminal-Adresse
8	8	Displ.-Adresse (0) | CU - Adresse (0 D0)
C	12	00 | Terminal - Adresse
10	16	Displ-Adresse (1) | CU - Adresse (0 D0)
		Anz.der Displ.an 2. CU | Terminal - Adresse
		Displ. Adresse | CU - Adresse (0E0)
		80 | Terminal - Adresse
		Displ. Adresse | CU - Adresse (0E0)

5.1 Modul TCP1DTAB (W. Strolz)

Display - Zuordnungstabelle

CSECT TCP1DTAB

AUFRUF

Von TCP1DSUP mit LOAD

FUNKTION

Die Display - Tabelle TCP1DTAB ordnet einer Line-Adresse eine Display-Adresse zu.
Die Displays müssen fortlaufend numeriert sein. Es dürfen keine Displaynummern fehlen.
Jeder Tabellenabschnitt, der einer Control - Unit zugeordnet ist, beginnt mit der Anzahl der Displays, die zu dieser CU gehören.
Das Listenende wird durch X'80' im ersten Byte des letzten Eintrages angezeigt. (Bild 2)

III. Systemüberwachung

1. Aktivitäten des OS (F. Sell)

Der Operateur hat die Möglichkeit, von der Konsolschreibmaschine aus das TCP in seinem Ablauf zu steuern, sich über die gegenwärtigen TCP-Aktivitäten zu unterrichten und an die Benutzer Nachrichten abzusenden.

Dazu werden Teile des OS und die Systemüberwachung des TCP benützt. Die Operateur-Kommandos für das TCP sind mit dem Schlüsselwort 'SEND' oder als Abkürzung mit # zu beginnen. Die Annahme der Kommandos erfolgt über die zum SVC34 gehörigen Moduls, die durch EOB von der Konsole aktiviert werden (CONSOLE INTERCEPT LOGIC).

Durch die Angabe der TCP-Kennung 'SEND' oder '#' am Anfang eines Kommandos wird nach der Abprüfung auf gültige OS-Kommandos der Fehlerausgang aktiviert. Der Fehler-Modul IGC0503D des OS wurde durch einen gleichnamigen Modul ersetzt, der die Abfrage auf TCP-Kommandos 'SEND', bzw. '#' enthält (Bild 13a, b, c,). Wird diese Abfrage nicht erfüllt, so wird die Kontrolle an den auf IGC9503D umbenannten OS-Fehler-Modul zurückgegeben.
Wird **jedoch** ein TCP-Kommando erkannt, so wird die Kontrolle an den Modul CRB1503D weitergegeben.

CRB1503D

Der Modul CRB1503D ist so organisiert, daß er für die Systeme CRBE und TCP arbeitet (für TCP jedoch nur, wenn CRBE nicht aktiv ist).
Die vom Operateur eingegebene Nachricht wird auf die Maximallänge abgeprüft (70 Zeichen) und in den Command Input Buffer (CIB) gebracht. Nach Aufsuchen des Task-Control-Blocks (TCB) wird der CIB an den Command Scheduling Control Block (CSCB) der Task angehängt. Durch ein POST auf den Operator-Event-Control-Block (ECB) wird das Eintreten dieses Ereignisses festgehalten; die Beantwortung des eingetretenen Ereignisses obliegt der angesprochenen Task TCP.
Folgende Fehler werden erkannt und durch WTO dem Operateur angezeigt:

- kein Parameter nach dem Schlüsselwort
- TCP ist nicht aktiv
- Die Nachricht an das TCP ist zu lang.

Die Kontrolle wird an das OS zurückgegeben.

2. Aktivitäten des TCP (F. Sell)

Nach einer Initialisierungsphase (siehe II.) läuft das TCP in eine Wait-Routine (Modul TCP1BASE, CSECT TCP1WAIT), in der auf das Eintreten verschiedener Ereignisse gewartet wird. Das Ereignis meldet sich über den entsprechenden ECB. Wurde vom OS

ANALYZE der Operateur-ECB gesetzt, so wird dies im TCP1WAIT erkannt und die Kontrolle an die Auswerte-Routine weitergegeben (Modul TCP1BASE, CSECT TCP1OPCP).

2.1. TCP1BASE, CSECT TCP1OPCP (OPERATOR-COMMAND-PROCESSING)

GOON Nach Laden des Basisregisters wird über den CSCB geprüft, ob ein CIB vorhanden ist. Falls nicht, wird die Kontrolle an TCP1WAIT zurückgegeben. Die Nachricht wird aus dem CIB in einen Puffer gebracht und der CIB aus der Kette entfernt.

Die Nachricht im Buffer besteht aus dem Aktionswort und den

OPCTL Parametern. Das erste Zeichen des Aktionswortes bestimmt das auszuführende Programm. Ist das Zeichen undefiniert, so wird nach einer Fehlermeldung (WTO) an den Operateur der nächste CIB der Kette bearbeitet.

Die Liste der auszuführenden Programme enthält folgende Aktionen:

Nachricht an:

a) alle aktiven Terminals
b) an einen bestimmten Benutzer mit und ohne Bestätigung für den Operateur
c) an alle aktiven Terminals und auf den Nachrichten-Datensatz M der Bibliothek TCP.QQQ.USER

Überprüfen, welche Benutzer aktiv sind.

Ablaufsteuerung des TCP:

a) Einschalten, Einlesen und Programmstart verbieten und erlauben
b) Ein- und Ausschalten der Benutzer auf der Konsolschreibmaschine anzeigen oder nicht anzeigen

c) Das TCP normal oder für Testzwecke mit einem Dump beenden
d) Ein Terminal logisch anschalten
Die Liste ist erweiterbar.

2.2. Nachrichten des Operateurs an TCP-Benutzer

2.2.1. Ausgeben der Nachricht an den eingeschalteten Terminals

Dieser Dienst wird durch das Aktionswort BROADCAST angefordert.

OPBDCST Die Syntax der Nachricht wird geprüft, der Benachrichtigungstext isoliert und am Anfang mit zusätzlicher Information (Uhrzeit, Nr. der Nachricht), am Ende mit Steuerzeichen (LF, CR, SYN) versehen. Über die KONBOX findet man die Kette der LINEBOXes.

OPCM3 Ist das zur LINEBOX gehörige Terminal eingeschaltet, so wird (wenn nötig nach Übersetzung der Nachricht aus dem EBCDI-Code in den ISO7-Code) der Nachrichten-Buffer an das Ende einer bereits vorhandenen Nachrichten-Bufferkette angefügt. Damit diese Nachrichten-Bufferkette sobald wie möglich ausgegeben wird, wird die Kette der Kanalkommandos unterbrochen, falls am Terminal auf eine Eingabe gewartet wird (PREPARE-Kommando). Ausnahme bei Nachrichten, die mit TCP14 beginnen; Spezialfall für TCPWTR. Die Unterbrechung der Kommandokette geschieht in TCP1BASE, Csect TCP1HALT; nach Abbrechen des PREPARE-Kommandos und Initialisierung der Ausgabe wird die Kontrolle an die Absprungstelle zurückgegeben. Sind noch weitere Terminals eingeschaltet, so werden sie der Reihe nach auf die gleiche Art bedient; dazu wird jeweils ein Buffer geholt, die Nachricht in diesen Buffer gebracht und der Buffer an die Bufferkette des Terminals angekettet. Nachdem die Nachricht an alle eingeschalteten Terminals weitergegeben wurde, wird die Kontrolle an den Anfang der CSECT TCP1OPCP gegeben, um weitere Kommandos zu bearbeiten.

2.2.2. Nachricht an einen bestimmten Benutzer

Dieser Dienst wird durch das Aktionswort ID angefordert. Es wird die Syntax der Benutzeridentifikation geprüft und die Benutzeridentifikation abgespeichert. Danach wird die Syntax der Nachricht geprüft, der Benachrichtigungstext isoliert, am Anfang mit Zusatzinformation (Uhrzeit, Nr. der Nachricht), am Ende mit Steuerzeichen (LF, CR, SYN) versehen. Bei der Benachrichtigung ohne Quittung an den Operateur unterbleibt die Quittungsausgabe.
Die Kette der LINEBOXës wird daraufhin untersucht, ob sich der Benutzer mit der angegebenen Identifikation an einem Terminal eingeschaltet hat. Wird die Identifikation gefunden, so wird die Nachricht an dem gefundenen Terminal (wie unter 2.2.) beschrieben) ausgegeben (OPCM3). Ist der Benutzer jedoch nicht eingeschaltet, so wird die Nachricht auf folgende Art in seine Bibliothek in das Member M geschrieben: Mit dem Makro LOCATE wird das Volume ermittelt, auf dem die Bibliothek mit dem Namen

GUT

TCP.XXX.USER laut Katalog existieren sollte (XXX = Benutzeridentifikation). Zur Überprüfung wird mit dem Makro OBTAIN auf

BESSER

dem gefundenen Volume die Bibliothek gesucht. Nach Eröffnen von Ein- und Ausgabe Datensätze wird mit dem Makro FIND überprüft, ob ein Member M in der Bibliothek bereits existiert. Existiert das Member M noch nicht, so wird es eingerichtet (Makros: NOTE, STOW) und die mit dem Datum versehene Nachricht im EBCDI-Code in das Member M geschrieben. Existiert jedoch das Member M schon, so werden die bereits eingeschriebenen Nachrichten ausgelesen und zusammen mit der neuen Nachricht auf den freien Bereich der Datei neu geschrieben. Danach werden die Ein- und Ausgabe-Datensätze geschlossen und zur Freigabe des

OUT2A

Arbeitsspeichers nach OUT2A gesprungen. Stehen noch weitere Kommandos des Operateurs an, so werden diese bearbeitet, sonst wird die Kontrolle an TCP1WAIT zurückgegeben.

Das Aktionswort W (= Without Message to Operator) ruft den gleichen Dienst auf, jedoch bekommt der Operateur keine Quittung. Zweck dieses Dienstes ist es, von anderen Programmen (Benutzer, TCPWRITER usw.) Nachrichten gezielt an einzelne Benutzer abzusetzen, ohne daß der Operateur durch Quittungen an der Konsole verwirrt wird.
Anmerkung: Bei den Systemmakros werden Returncodes nur insofern ausgewertet, wie es zur Steuerung des Ablaufes notwendig ist; Fehlercodes werden nicht gesetzt.

2.2.3. Nachricht an alle eingeschalteten Terminals und auf das Member M der allgemeinen Bibliothek TCP.QQQ.USER

Der durch das Aktionswort ALL angeforderte Dienst ist eine Kombination der Dienste von BROADCAST und ID (Punkt 2.2. und 2.3.). Die Nachricht des Operateurs wird an allen eingeschalteten Terminals mit Uhrzeit versehen ausgegeben und außerdem zum späteren Abruf auf das Member M der allgemein zugänglichen Bibliothek TCP.QQQ.USER mit Datum versehen, abgespeichert.

2.3. Überprüfen, welche Benutzer aktiv sind

Dieser Dienst wird durch das Aktionswort USER angefordert. Die Kette der LINEBOXes wird daraufhin untersucht, welcher Benutzer sich an welchem Terminal eingeschaltet hat, und der Operateur wird davon in Kenntnis gesetzt. Nach Beendigung dieses Dienstes wird überprüft, ob noch weitere Kommandos auszuführen sind, ansonsten wird die Kontrolle an TCP1WAIT zurückgegeben.

2.4. Ablaufsteuerung des TCP

Die Ablaufsteuerung erfolgt, indem durch das Kommando des Operateurs ein Flag in der KONBOX gesetzt wird, das vom TCP im geeigneten Zeitpunkt abgefragt und interpretiert wird (Ausnahme: Dump zu Testzwecken).

2.4.1. Verbot des Einschaltens, Einlesens und Startens von Programmen, bzw. Aufheben des Verbots

Diese Dienste werden durch die Aktionswörter HOLD, bzw. RELEASE angefordert.
Das Aktionswort HOLD setzt in KONBOX ein Flag, das von TCPOPROC interpretiert wird. Ist das Flag gesetzt, so ist es Benutzern nicht möglich, sich einzuschalten, einen Leseauftrag abzusetzen oder ein Programm zu starten. Die Benutzer werden durch das Kommando des Operateurs informiert. Zweck dieses Kommandos ist es, das Abschalten des TCP rechtzeitig vorzubereiten oder das System vor Überlastung durch abgesetzte Jobs zu bewahren.
Das Aktionswort RELEASE setzt das Hold-Flag in der KONBOX zurück; damit ist es nach der Interpretation von TCPOPROC wieder erlaubt, daß Benutzer sich einschalten, einlesen und Programme starten.

2.4.2. Benachrichtigung des Operateurs über Ein- und Ausschalten von Benutzern und Aufheben dieses Modus

Diese Dienste werden durch die Aktionsworte LOG bzw. NOLOG angefordert.
Das Aktionswort LOG setzt in KONBOX ein Flag, das von TCPOPROC interpretiert wird. Ist das Flag gesetzt, so wird der Operateur vom Ein- und Ausschalten der Benutzer automatisch durch einen Ausdruck an der Konsole unterrichtet.
Das Aktionswort NOLOG setzt das Flag in KONBOX zurück, der Ausdruck an der Konsole beim Ein- und Ausschalten von TCP-Benutzern unterbleibt.

2.4.3. Beendigung des TCP normal oder für Testzwecke mit Dump

Diese Dienste werden durch die Aktionsworte SHUTDOWN, bzw. DUMP angefordert.
Das Aktionswort SHUTDOWN setzt ein Flag KONBOX, das zur geeigneten Zeit von TCPOPROC interpretiert wird. Das TCP wird daraufhin geordnet beendet, d.h. es werden noch Aufräumungsarbeiten durchgeführt (z.B. Freigabe von Buffern, Schließen von Datensätzen). Das Aktionswort DUMP erzwingt sofort über das Makro ABEND eine Beendigung des TCP mit einem Dump für Testzwecke.

2.4.4. Logisches Einschalten eines Terminals

Dieser Dienst wird durch das Aktionswort EIN angefordert.
Ist ein Terminal elektrisch eingeschaltet, so kann es dennoch manchmal dem TCP nicht gemeldet sein, z.B. durch falsche Grundstellung des Terminals bei der TCP-Initialisierung, durch Übertragungsfehler und Abbrechen der Verbindung. Der Operateur kann durch das Kommando EIN das Terminal logisch anschalten.

3. Querverbindungen zwischen OS und TCP (F. Sell)

Da der Ablauf der Kommandos durch den Operateur identisch ist mit dem Aufruf SVC34, können die dem Operateur zur Verfügung stehenden Möglichkeiten zur Systemüberwachung auch von jedem beliebigen Programm gestartet werden. An Anwendungsfällen sind dabei jedoch nur die Benachrichtigung bestimmter Benutzer sinnvoll; diese Aktion (siehe III.2.2.2.) wird in folgenden Fällen benutzt:

3.1. Anwendung in TCPWTR

Das Programm TCPWTR hat die Aufgabe, den TCP-Benutzer über Fortgang und Ausgang von Jobs zu informieren. Dazu werden durch TCPWTR Nachrichten zusammengestellt, die über den SVC34 an das TCP und über dessen Systemüberwachung an den spezifizierten Benutzer (bestimmt durch die Rechenzentrumsnummer) abgesetzt werden.

3.2. Anwendung in Assembler-Jobs

Zur Überwachung des Ablaufes von Assembler-Jobs existiert ein Makro-Befehl WTT, (Write to Terminal), der den angegebenen Text dem Benutzer des Jobs (bestimmt durch die Rechenzentrumsnummer) in der Reihenfolge der Makroaufrufe mitteilt. Die Nachricht wird nach 2.2.2 entweder an dem Terminal direkt ausgeschrieben, an dem der Benutzer sich eingeschaltet hat, oder auf die Bibliothek des Benutzers geschrieben (Wirkung ähnlich wie WTO).

3.3. Anwendung in Fortran-Jobs

Zur Überwachung des Ablaufs von Fortran-Jobs existiert ein For - tran-Unterprogramm WTT, das angebbaren Text in der Reihenfolge des Aufrufs an den durch die Rechenzentrums-Nummer festgelegten Benutzer absetzt.

4. Benutzerprogramm für Kommunikation von Terminalbenutzern mit der Systemkonsole (V. Haase)

Modul: TCP1SEN

TCP1 SEN wird von TCPOPROC attached, wenn das Kommando NACHRICHT, Text gegeben wird.
In einem dynamisch angelegten Speicherbereich (mit ALEPH) werden der String TCPO2, die Terminaladresse (LINEAD aus der Linebox), die Benutzerkennung (USERID aus der Linebox) und der eingegebene Text (Adresse in WADR1) übertragen. Mit SVC 35 (WTO) wird der Text an der Systemkonsole ausgegeben. Zuletzt werden die beiden Textpuffer (auch bei der WADR1!) zurückgegeben.

5. Überwachung der Terminalsitzungen (U. Ullrich)

TCP1ACNT

Dieser Modul wird bei LOGON (EIN) und LOGOFF (AUS) von TCPOPROC durch LINK aktiviert.

FUNKTION

EIN-Routine überprüft Passwort und Benutzerbibliothek; OPEN PARTDCB und PARTDCO.
AUS-Routine erstellt und schreibt den ACCOUNT RECORD; CLOSE PARTDCB und PARTDCO; POST auf OUTECB.

EINGANGSPARAMETER

Linebox

AUSGANGSPARAMETER

LCC12 (= X'06', wenn TCP1ACNT in Ordnung)

PROGRAMMABLAUF

Nach Makro ALEPH und Laden der Konbox- und Lineboxregister wird ein Ausgang INTERR für alle evtl. auftretenden Interrupts sowie für Fehler, die bei der Ausführung eines Makros auftreten, festgesetzt (Makro: SPIE). Dort wird LCC12 auf X'FF01' gesetzt und zum Ende gegangen.

Abfrage ob Befehl EIN oder AUS; wenn AUS Sprung nach AUSROUT.

EINROUT Der Zeitzähler wird angestoßen (Makro STIMER); die Ausgangszeit (MAXZEIT) beträgt 104 sec. Das Benutzer-Passwort (USERID) wird im Dataset TCP.XXX.USER eingefügt und der zugehörige VOLUME LISTBLOCK, der Information über den DCB enthält, wird mit LOCATE nach WKAR gelesen.

OK1 — Es folgt Prüfung der USERID. Wenn die USERID QQQ (zu jeder Zeit jedem Benutzer verfügbar) benutzt wird, geht das Programm bei OK10 weiter. Sonst wird die Linebox-Kette nach der aktuellen

COMP — USERID durchsucht. Wenn diese gefunden ist, wird der Inhalt von

EQUAL — KBXREG mit dem Inhalt des Lineboxregisters verglichen, das zu der Linebox gehört, welcher auch die USERID zugeordnet ist. Stimmen die Inhalte nicht überein, geht das Programm zum Fehlerausgang NONOME, wo die USERID neutralisiert und LCC12 X'FF02' gesetzt wird. Die Lineboxkette wird bis an den Anfang weiter-

OK10 — gerückt. Die Volume Serial Number wird aus dem Volume List Block nach VOLN und VOLSER geholt; mit Hilfe des Makro OBTAIN wird

OK2 — der DSCB für TCP.XXX.USER nach WKAR geladen. Die Adressen der Exit-List sowie VOL SER NR werden in die Input- und Output DCBs (PARTDCB und PARTDCO) eingetragen und Job File Control Block

OL — (JFCB) für PARTDCB gelesen. Anschließend wird DSN (TCP.XXX.User) in den Benutzer-Program-Data-Set (WKAR) geschrieben und Record Format im PARTDCO wird mit U (undefined length) angegeben. OPEN PARTDCB und PARTDCO. Die Terminal Flagge (TFLG) wird auf INPUT,

AUSROUT — EIN, COMMAND gesetzt. Sprung nach WTOTEST. Der Zeitzähler wird angestoßen, die Ausgangszeit steht in LBXTIME. Display-Status

CLOSE — wird gelöscht. Danach CLOSE PARTDCB und PARTDCO. TFLG zeigt an, daß AUS der letzte Befehl war und daß Befehlseingabe erwartet

WTOTEST — wird. Wenn Status angibt: NO-LOG-Modus, darf TCP keine Nachricht an den Operateur senden, andernfalls erhält der Operateur an der Konsole die Nachricht: "TCP12XXX hat OXX EIN (AUS)- GESCHALTET".

TIMEM — Mit Makro TIME werden Uhrzeit und Datum geholt, dann entpackt

JAHR — und zur Ausgabe vorbereitet. Dann wird ein Puffer angefordert (Makro REQBUF), in dem die EIN- oder AUS-Nachricht (MESS) an den Benutzer geschrieben werden soll. Die Anfangsadresse des Puffers wird in BUFADS gespeichert. Datum und Uhrzeit werden in MESS ein-

MVEND — getragen, MESS wird in den Ausgabepuffer geschrieben und mit Hilfe der Output-translate-table (TTOUT) für die Ausgabe am Terminal übersetzt. Wenn Befehl 'EIN' ist, erfolgt hier Sprung nach END.

ACC Der Zeitzähler wird angehalten (TTIMER CANCEL); die CPU-Zeit, die der Benutzer von 'EIN' bis 'AUS' benötigt hat, wird errechnet und im ACCREC (ACCCPU) und in MESS eingetragen. Außerdem werden Uhrzeit, Datum, Passwort, Hardwareadresse des Terminals (LINEAD) sowie die Benutzerkennummer , die aus USERID mit Hilfe der ACCTAB zusammengebaut wurde, in den ACCREC gespeichert. Die CPU-Zeit wird von MESS in den Ausgabepuffer übertragen und übersetzt. Dann wird der ACCREC geschrieben (Makro SMFWTM). Wenn Logoff erzwungen (SYSTEMFEHLER), Sprung nach END3. Sonst wird MESSAGE mit CALL aufgerufen. Nach Rücksprung in TCP1ACNT werden die Linebox-Parameterfelder gelöscht und die Geräteselektion am Terminal für Input und Output neu bestimmt.

END Dann POST OUTECB und Sprung nach END3. Zeitzähler wird gestoppt, die Restzeit (Ausgangszeit minus gebrauchte Zeit) in LBXTIME eingetragen.

END INTERRUPT EXIT wird gelöscht. Programm endet mit 'HE'.

IV. Ein- und Ausgabesteuerung der Datenstationen (E. Holler)

Für die Überwachung der Ein- und Ausgabe an den Datenstationen ist eine eigene Task eingesetzt. Die Steuerung der Ein- und Ausgabe erfolgt dort über die Controlsection TCP1WAIT innerhalb des Moduls TCP1BASE. Die Behandlung der verschiedenen Formen der Ein- und Ausgabe sind unter Verwendung mehrerer Hilfsroutinen vorgenommen: determinierte E/A-Prozesse werden von TCP1WAIT unter Zuhilfenahme der Controlsections OUTSPOOL, ENDPUT und TCP1HALT im Modul TCP1BASE bearbeitet, während für die Dateneingabe mit undefinierter Blocklänge die notwendige schritthaltende Bearbeitung (Umformatieren und Zwischenspeichern) durch dynamische Aktivierung des Moduls TCP1SPOL unter Kontrolle einer eigenen Task erzielt wird. TCP1WAIT kreiert für jeden dieser Eingabeprozesse je eine Task, die von TCP1WAIT nach Erfüllung ihrer Funktion wieder vernichtet wird. Die Initialisierung der einzelnen Ein- und Ausgaben von TCP1WAIT aus wird über einen Satz von Event-Control-Blöcken von Modul TCPOPROC, der unter Kontrolle der Kommandointerpretations- und -bearbeitungstasks arbeitet, gelenkt; umgekehrt wird TCPOPROC von TCP1WAIT über die einzelnen Terminals zugeteilte Event-Control-Blöcke dann informiert, wenn zu interpretierende oder zu bearbeitende Teilnehmerkommandos eingelesen worden sind.

1. TCP1WAIT (E. Holler)

TCP1WAIT ist eine Controlsection innerhalb des Moduls TCPBASE und wird nach Beendigung der TCP-Initialisierungsprozedur über einen Sprung von Controlsection TCP1BASE aus aktiviert.

FUNKTION

Steuerung der Terminal Ein- und Ausgabe, Analyse von E/A-Fehlern und Korrekturversuch, Weiterleitung eingegebener Commandos an TCPOPROC, Einleitung der Behandlung von System-Operator-Kommandos sowie Terminierung von TCP durch dynamischen Aufruf von TCP1TSDN.

EINGANGSPARAMETER

Register 6 muß auf die Konbox zeigen.

AUSGANGSPARAMETER

a) beim Aufruf von OPCP und TCP1TSDN zeigt Register 6 zur Konbox
b) beim Aufruf von OUTSPOOL und ENDPUT zeigt Register 6 auf die Konbox und Register 12 zur aktuellen Linebox
c) bei Aktivierung von TCP1SPOL zeigt Register 1 auf eine Parameterliste, die die Adresse der aktuellen Linebox und der Konbox enthält.

PROGRAMMABLAUF

TCP1WAIT Nachdem der aktuelle Anfang der ECB-Liste für das spätere TWMT ermittelt und die zugehörige Task auf Key 0 und Supervisorstatus gebracht ist, wird die Task durch TWAIT in den Wartezustand versetzt. Der Tasksupervisor aktiviert die Task erst dann wieder, wenn a) ein Operator-Kommando (Systemnachricht) zu bearbeiten ist, oder b) die Entschlüßler-Task ihre Tätigkeit beendet hat, oder c) der Entschlüßler TCPOPROC einen E/A-Vorgang aktivieren will, oder d) eine Terminal Ein- oder Ausgabe beendet ist.

TWAIT

LISTSHI — Ist eines der aufgezählten Ereignisse eingetreten, dann wird der aktuelle Beginn der ECB-Liste zyklisch um einen Eintrag verschoben (damit nicht immer die gleiche ECB-Adresse am Anfang der Liste steht und der zugehörige Event dadurch bevorzugt bedient wird). Die Task wird wieder auf der ursprünglichen Key

ANALYZE — und in den Problemstatus gebracht. Es folgt nun die Ereignisanalyse, die feststellt, welches der oben klassifizierten Ereignisse eingetreten ist. Folgende Aktionen werden in den einzelnen Fällen eingeleitet:

Ist eine Systemnachricht oder ein Operatorkommando zu analysieren

OPPOST — und zu bearbeiten, so geht die Kontrolle per Sprung an die

TASKCOMP — Controlsection TCP1OPCP, bei Beendigung der Entschlüßler Task (TCPOPROC) wird nach dem DETACH auf diese Task über ein LINK TCP1TSDN aktiviert. Mit einem Return beendet Modul TCP1BASE in diesem Falle seine Tätigkeit.

Der Fall, daß der Entschlüßler TCPOPROC eine Ereignisvariable

SCHLPOST — gesetzt hat, die einem speziellen Terminal zugeordnet ist, ist komplizierter zu behandeln. Zunächst wird die dem betroffenen

SANALYZE — Terminal zugeordnete Linebox ermittelt. Wird anhand der ACTION-

ACTTEST — Bits in der Linebox festgestellt, daß gerade die Ausgabe einer Operatornachricht am Terminal vorgenommen wird, dann muß der von TCPOPROC gewünschte E/A-Vorgang zurückgestellt werden, bis die laufende Ausgabe beendet ist. Nach Setzen des Flags SCHLPD im ACTION-Byte der Linebox folgt daher die Rückkehr nach TCP1WAIT. Ist die Datenstation jedoch frei, wird der IOB des Terminals

GEMOB — ermittelt und die von TCPOPROC in den TFLG-Flagbits der Linebox angezeigte Aktion eingeleitet. Die Ausgabe einer Nachricht, die von TCPOPROC erstellt worden war, wird nach entsprechender Initialisierung der I/O-Kontrollblöcke des Terminals über die

DYNOUT — Subroutinen DYNOUT und IOCENTER mittels EXCP eingeleitet.

MQU — Eventuell für das Terminal vorhandene Operator-Nachrichten werden zuvor mit der Entschlüßlernachricht verkettet und mitausgegeben. Eine von TCPOPROC gewünschte Ausgabe direkt aus einem auf Platte gespeicherten Datenbestand wird unter Zuhilfenahme von Controlsection OUTSPOOL in Blöcken zu je 10 Records vorbereitet.

Die dazu benötigte Zahl von Puffern wird von OUTSPOOL dem gemeinsamen Pool entnommen. Die von OUTSPOOL vorbereitete Ausgabe

IOCENTER

besorgt auch in diesem Falle die Subroutine DYNOUT. Die Routine IOSENTER wird für alle E/A-Vorgänge zum Start der jeweiligen Kanalprogramme eingesetzt. Welches Kanalprogramm erstellt wird, entscheidet sich auf Grund des im Feld DECTYPE in der Linebox abgelegten Codes. Die BTAM-Read-Write-Routine IGGO19MA, zu der von IOCENTER aus verzweigt wird, benutzt diesen Code, um aus dem Device-I/O-Module, der speziell auf die Siemens-Datenstationen zurechtgeschnitten wurde, das gewünschte Kanalprogramm zusammenzustellen. Bei Rückkehr von IGGO19MA nach IOCENTER wird geprüft, ob der Start der Ein- und Ausgabe geglückt ist. Nach erfolgreichem Start erfolgt entweder die Rückkehr nach TCP1WAIT

IOP

in den Bereitschaftszustand oder nach TCP1OPCP falls die Behandlung weiterer Systemnachrichten bzw. Operatorkommandos notwendig ist.

Die gleiche Ereignisvariable (OUTECB), die von TCPOPROC zur Einleitung von E/A-Vorgängen am Terminal gesetzt wird, wird auch zur Anzeige der Terminierung der für die Kontrolle der dynamischen Dateneingabe verantwortlichen Task (Modul TCP1SPOL) benutzt. Die Unterscheidung erfolgt auf Grund der im Action-Byte

SPLTEND

in der Linebox gesetzten Bits. Wurde Modul TCP1SPOL terminiert, erfolgt ein DETACH auf die zugehörige Task, falls die Beendigung nicht von einem Timeout an der Datenstation herrührte. Bei Timeout wird die Task nicht beendet, sondern das Kanalprogramm über RDCDYN neu gestartet. Nach dem DETACH ergibt die Analyse des Completioncodes im OUTECB, ob die Task TCP1SPOL normal verlaufen ist oder nicht. Ein von Null verschiedener Code bewirkt einen Stop der Eingabe durch Zurücksetzen der PCI Bits im Kanalprogramm. Wird in der Linebox angezeigt, daß das Kanalprogramm

SPLTEND1

bereits beendet ist, dann wird bei von Null verschiedenem

POSMW

Completion-Code eine Fehlernachricht für die Ausgabe am Terminal

TOMESS

zusammengestellt, in einen Puffer aus dem gemeinsamen Pool gebracht, mit eventuell von TCP1SPOL bereitgestellten Fehlermeldungen verkettet und über Routine TOMESS unter Benutzung von DYNOUT am Terminal ausgegeben.

Bei normalem Verlauf von TCP1SPOL wird TCPOPROC vor dem Rücksprung nach TCP1WAIT durch die Ereignisvariable INECB (POST auf INECB) über die durchgeführte Dateneingabe informiert. Bei noch nicht beendetem Kanalprogramm muß vor der Einleitung der eben beschriebenen Aktionen erst dessen Beendigung angezeigt werden; daher ist eine Rückkehr in den Bereitschaftszustand notwendig (TCP1WAIT). Die letzte der zu behandelnden Bedingungen, unter denen TCP1WAIT aus dem Wartezustand heraus aktiviert wird, ist die Beendigung eines E/A-Vorgangs an einer Datenstation. Dies kann zu drei verschiedenen Arten von Aktionen innerhalb von TCP1WAIT führen:

IOPOST
IOCOMP

a) Ist der von TCPOPROC initialisierte E/A-Vorgang fehlerfrei beendet, so muß über die Ereignisvariable INECB die entsprechende Rückmeldung an TCPOPROC erfolgen.

b) Wenn durch die beendete E/A-Operation nur ein Teil des von TCPOPROC gewünschten E/A-Vorgangs bearbeitet wurde, so ist die nächste E/A-Operation einzuleiten (in der Regel folgt z.B. im Dialog auf eine Ausgabe eine Eingabe).

c) Ist eine E/A-Operation nicht fehlerfrei verlaufen, so muß untersucht werden, ob der Fehler durch eine Wiederholung des E/A-Vorgangs korrigiert werden kann, ob der Fehler vernachlässigt werden kann oder ob der Fehler so schwerwiegend ist, daß die betroffene Datenstation als nicht mehr bedienbar (LINEDOWN) eingestuft werden muß.
Zunächst wird mittels Completion-Code Analyse nach formal fehlerfreiem Verlauf und formal fehlerhaftem Verlauf der beendeten E/A-Operation differenziert.

ECB7F
NAKTEST
FLAG08
EOTTEST

Completion Code 7F kennzeichnet einen formal fehlerfreien Verlauf; ein Fehler kann jedoch trotzdem durch eine negative Response (NAK) vom Terminal auf eine Ausgabe hin, durch die nicht rechtzeitige Bereitstellung von Eingabepuffern bei der Dateneingabe, oder durch vorzeitigen Abbruch einer Eingabe durch Fehler am Terminal entstanden sein.

Nach Aussonderung dieser Fehlermöglichkeiten kann der Verlauf der beendeten I/O-Operation als fehlerfrei bezeichnet werden. Der Typ der beendeten I/O-Operation (in DECTYPE: 00, 01, 02, 03, 04, 05, 08, 0A, 0B) bestimmt die nun folgenden Aktionen (die Label der entsprechenden Routinen sind mit OKxx gekennzeichnet, für xx ist der I/O Type einzusetzen). Entweder wird eine weitere I/O-Operation initialisiert, die logisch aus der vorangegangenenen resultiert, oder TCPOPROC verständigt, daß der geforderte I/O-Vorgang beendet ist (Kommando eingelesen, Daten eingelesen oder auch nur Nachrichten oder Daten ausgegeben).

ECB41

Completion Code 41 kennzeichnet einen formal fehlerhaften Verlauf einer I/O-Operation. Nur in wenigen Fällen kann die beendete I/O-Operation trotzdem als fehlerfrei eingestuft werden: dies geschieht bei einem Timeout oder Datacheck auf einen Read Response-oder Read EOT-Channelcommand hin. Die eigentliche Fehleranalyse läuft nach einem Prioritätsplan ab

LDOWN

(siehe /10/). Nicht korrigierbare Fehler führen zum Außerbetriebsetzen der betroffenen Datenstation (LINEDOWN). Für Fehler, die als korrigierbar angenommen werden, wird die betroffene I/O-Operation (bei Eingabe nach Ausdrucken einer entsprechenden Fehlernachricht am Terminal) bis zu zehnmal wieder-

LPERMERR

holt; tritt der Fehler dann immer noch auf, so wird er ebenfalls als nicht korrigierbar eingestuft. Der System-Operator wird durch eine Nachricht informiert; durch Setzen der Ereignisvariablen (POST INECB) wird TCPOPROC von der Außerbetriebsetzung der Datenstation informiert. Mit I/O-Operationen, deren Beendigung als formal fehlerfrei angezeigt worden war, die aber (im ECB7F) dennoch als fehlerfrei erkannt wurden, wird in der gleichen Weise verfahren wie im Falle von korrigierbaren Feh-

REMOTE

lern. Kommt es beim Lesen der Response auf die Selektion einer Terminalkomponente (nur bei Ausgabe) zu einem Timeout, so wird die betreffende Station als im Lokalbetrieb befindlich

TERMINM

markiert. Über ein Prepare-I/O-Kommando kann, wie im Falle einer kurzzeitigen Fehlerfunktion einer Terminalkomponente, die erneute Betriebsbereitschaft des Terminals festgestellt werden.

EOTRESPN Vom Terminal an einer Datenstation kann durch Betätigung der EOT (Aufruf-) Taste während einer laufenden Ausgabe künstlich eine Fehlerbedingung erzeugt werden (Timeout oder Unit Exception) die zum vorzeitigen Abbruch längerer Ausgaben aus Bibliotheken führt. In diesem Fall wird von ECB41 aus die Routine ENDPUT aktiviert, die ein CLOSE auf den Dataset veranlaßt, von dem aus die Übertragung auf die Datenstation über Routine OUTSPOOL vorgenommen worden war.

2. OUTSPOOL / ENDPUT (S. Müller)

FUNKTION

Die beiden CSECTs OUTSPOOL und ENDPUT sind in dem Modul TCP1BASE enthalten. OUTSPOOL wird aufgerufen von der CSECT TCP1WAIT, die ebenfalls im Loadmodule TCP1BASE ist. Die Aufgabe der CSECT OUTSPOOL ist die Übertragung des Inhaltes des SPOOL-DS (Ausgabe DS), des PL/1-Ausgabe-DS (TCP.Terminal Nr. PLOUTPU) oder eines Ausgabe-DS der SYSOUT = X - Klasse in Ausgabepuffer. Sind eine bestimmte Zahl von Records in Puffer geschrieben, wird nach DYNOUT gesprungen. Hier werden die Puffer an das Terminal ausgegeben und anschließend in die CSECT OUTSPOOL zurückgesprungen. Ist der DS ganz ausgegeben, wird nicht in die CSECT OUTSPOOL sondern in die CSECT ENDPUT gesprungen. Nach ENDPUT wird auch aus der CSECT OUTSPOOL gesprungen, wenn die Ausgabe aus Fehlergünden abgebrochen werden muß. Die CSECT ENDPUT ist die abschließende Routine für die CSECT OUTSPOOL. Ist die CSECT ENDPUT beendet wird die Kontrolle der CSECT TCP1WAIT zurückgegeben.

EINGANGS-UND AUSGANGSPARAMETER

Eingangsparameter für OUTSPOOL
Adresse der KONBOX in Reg. 12, wird umgeladen nach Reg.5.
Adresse der LINEBOX in Reg. 6, wird umgeladen nach Reg.3.
Folgende Felder aus der KONBOX werden benötigt:

TTOUT Adresse der Translatetabelle (Output) zum Übersetzen der Ausgabe

DDCB DCB-Adresse für 'Line group' (Puffer DCB)

Aus der LINEBOX werden verwendet:

ACTION+1 'OOOOOOOX' SWITCHBIT

0 CSECT OUTSPOOL wird für die Ausgabe zum erstenmal angesprungen

1 nicht zum erstenmal.

'XOOOOOXO'

0 0 Der Output steht auf den SPOOL DS, Bibliotheks-Output.

0 1 Es handelt sich um ein Output der Art SYSOUT=X (System-Output)

1 1 Es handelt sich um den Output eines mit 'PL1START' gestarteten PL/1-Programmes.

LAMEMBN bei Programm-Output steht in LAMEMBN der 3. Level des Output-DS-Namens.

LINEAD DCB des SPOOL DS; er wird verwendet, wenn der Output auf den SPOOL DS steht.

SPDSPLAD Anfangsadresse des SPOOL DS auf der Platte.

USERID Benutzerkennzeichen

WORKAASP Adresse der WORKAREA. Sie wird beim ersten Durchlauf eingeschrieben.

Ausgangsparameter an DYNOUT aus der LINEBOX:

ACTION+1 'OOOOOXOO' EODAD BIT

0 Ausgabe noch nicht beendet

1 Ausgabe beendet

BUFADS Adresse des 1.Puffers oder 0 wenn kein Puffer gefüllt ist. Das 1. Byte der Adresse enthält die benutzte Länge in Bytes des letzen Puffers.

Ausgangsparameter an CSECT ENDPUT ist gleich Eingangsparameter für ENDPUT

Adresse KONBOX in Reg 5

Adresse LINEBOX in Reg 3

In/Aus der LINEBOX:

WORKAASP	Adresse der WORKAREA. Die WORKAREA ist bei mehreren Durchläufen von OUTSPOOL immer dieselbe. Sie wird in ENDPUT freigegeben.	
ERRSTBIT	Steuert die Funktion von ENDPUT zusammen mit ACTION+1	
ERRSTBIT		
10000000	Keine Übertragung, nur Fehlercodes gilt nach ENDPUT ohne vorher nach DYNOUT zu gehen	KÜ BIT
01000000	PL/1-OUTPUT	PL BIT
00001000	UNCATALOG ausführen	UC "
00000100	CLOSE	CL "
00000010	SCRATCH und UNCATALOG ausführen	CS "
00000001	Nur Nachricht, kein CLOSE und bei Output kein SCRATCH ausführen	NA "
		Diese Abkürz. der Besch. verwendet
DCBADR	Adresse der verwendeten DCBs; SEQDCB aus LINEBOX bei Bibliotheks-Output oder eigener DCB bei anderem Output.	
DSN	DS-Name	
VOLIST	DEVICE Code und VOLSER Nr.	
LCC12	'00' oder Fehlercode	

Ausgangsparameter aus LINEBOX:

ACTION+1	'00000XXX'	X auf 0 gesetzt
INECB	gepostet	
LCC12	X'00' oder Fehlercodes.	

PROGRAMMBESCHREIBUNG

2.1. OUTSPOOL

Die CSECT OUTSPOOL beginnt mit einem SAVE-Macro und der Definierung des Basisregisters (BASREG1). KBXREG und LBXREG, die Adress-Reg. für die KONBOX und die LINEBOX werden anschließend geladen. Nun wird gefragt, ob das SWITCHBIT in ACTION+1 gesetzt ist. Ist es gesetzt, bedeutet das, daß die CSECT OUTSPOOL bei der Ausgabe nicht zum erstenmal aufgerufen wurde. Darum wird jetzt Reg 13 mit der Adresse der WORKAREA aus WORKAASP geladen und das Programm bei HOLEBUFF fortgesetzt.
Ist das SWITCHBIT nicht gesetzt, wird die CSECT OUTSPOOL für

FIRST eine Ausgabe zum erstenmal aufgerufen. Zuerst wird das SWITCHBIT gesetzt. Dann fragt man ob eine Ausgabe von der Bibliothek oder von einem Programm (SYSOUT=X oder PL/1-Output) behandelt werden soll. Davon abhängig wird (GETMAIN) der Platz für die WORKAREA besorgt. Anschließend werden die SAVEAREA-Adressen gespeichert, die WORKAREA mit USING adressierbar gemacht und die Anfangsadresse der WORKAREA nach WORKSPAASP in die LINEBOX gespeichert. Nun wird in ACTION+1 das EODADBIT und das Byte ERRSTBIT zu 0 gesetzt. Weiter wird in LRECLAR die LRECL=80 und in DCBADR die Adresse des SEQDCBs eingetragen.
Nun wird wieder geprüft um was für eine Ausgabe es sich handelt (ACTION+1).
Bei einer Bibliotheks-Ausgabe wird in der SEQDCB die Adresse der EODAD- und der SYNAD-Routine eingetragen, die READ-Listform aufgebaut und ein OINT auf den SPOOL DS (Ausgabe DS) ausgeführt. Dann wird nach RECHBUFF gesprungen.
Bei einem Programm-Output wird nach der Prüfung nach

OUTPUTSY gesprungen. Hier werden die nötigen Vorbereitungen getroffen, um den Data Set lesen zu können, auf dem der Progr.Output steht.

Zuerst wird der mit dem Namen OUTDCB generierte DCB in die WORKAREA geschoben. Mit dieser Kopie, genannt SPOOLDCB, wird in diesem Programm gearbeitet. Dann wird, abhängig von der Output-Art, der Data Set Name zusammengestellt. Vorher wird aber das DSN-Feld mit BLANK überschrieben.
DSN = TCP.XXX.#NNNNNNN
Es bedeutet bei Programm-Output (SYSOUT= X)
XXX=USERID, #NNNNNNN=LAMEMBEN
Anschließend wird die CAMLIST für LOCATE aufgebaut. Mit dem Macro LOCATE wird der 'Volume Liste Block' in WAREA gelesen. Aus ihm erfährt man, auf welches 'Volume' der DS geschrieben wurde. Tritt bei LOCATE ein Fehler auf, so wird er in der

OK1 'Fehlerroutine 1' verarbeitet.

Die VOLSER-Nr. und der DEVICE-Code werden aus dem 'Volume-List-Block' geholt und für den späteren Gebrauch in VOLUME und VOLIST gespeichert.
Nun wird die CAMLIST für das Macro OBTAIN aufgebaut und das Macro ausgeführt. Man liest mit ihm den DSCB des gesuchten DS von der VTOC der mit LOCATE bestimmten Platte in das Feld WAREA. Fehler, die bei OBTAIN auftreten werden ebenfalls in der 'Fehler-routine 1' behandelt.

OK2 Sind keine Fehler aufgetreten, so werden folgende Felder des SPOOLDCBs mit den Angaben des DSCB ergänzt:
DSORG, RECFM, BLKSITZE, LRECL. In der anschließenden Abfrage wird geprüft, ob die BLKSIZE = 0 ist. Wenn ja, wird im ERRSTBIT das KÜ-Bit gesetzt, im LCC12 ein Fehlercode gesetzt und nach LETZT gesprungen. In diesem Fall handelt es sich um einen ungeschrie= benen DS.
Ist BLKSIZE > 0 so wird bei

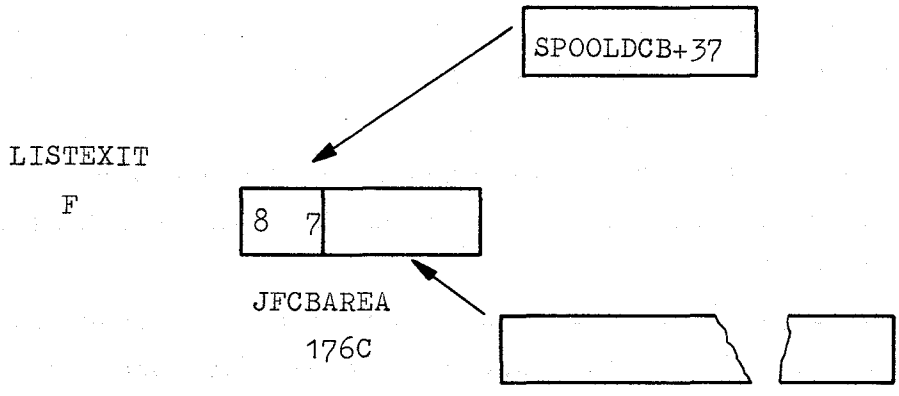
COLRECL fortgefahren. In der folgenden Abfrage wird nachgeschaut, ob LRECL=0 ist. Wenn ja, wird in LRECLAR die BLKSIZE gespeichert und nach MOVEVOC gesprungen. Wenn nein, wird bei

MOVLRECL in LRECLAR die LRECL gebracht.

Der Wert,der jetzt in LRECLAR steht, bezeichnet jetzt eine Zeilenlänge des DS, der ausgegeben werden soll. Diese Länge wird später im Programm für eine richtige Formatierung der Ausgabe-

MOVEVOC

zeilen benötigt. In den SPOOLDCB wird nun die VOLUME SER Nr. eingetragen. Für das folgende Systemmacro RDJFCB muß in den DCB eine EXITLIST eingetragen werden. Sie wird jetzt aufgebaut. Sie enthält die Adresse des Feldes JFCBAREA, in das der JFCB eingelesen werden soll und im ersten Byte den Code X'87'. Die Adresse der Exitliste LISTEXIT wird im SPOOLDCB gespeichert.



Anschließend wird die Listform für das RDJFCB aufgestellt und das RDJFCB ausgeführt.
Damit wird der JFCB in die JFCBAREA gelesen. In diese Kopie werden folgende Angaben eingetragen:
DSNAME, DSORG, RECFM, BLKSIZE, LRECL.
Die Adresse des SPOOLDCB wird für den späteren Gebrauch nach DCBADR gespeichert.
Es folgt ein Macro OPEN TYPE = J
Nun ist es möglich, den gesuchten DS, dessen DSN nicht im DD-Statement steht, zu lesen.
Hier ist die Vorbereitung für das Lesen eines Programm-Output-DS zu Ende.

Der Programmablauf für die Ausgabe der beiden Output-Arten trifft sich nun wieder in

RECHBUFF Zur Zeit können nur Zeilen mit bis zu 107 Zeichen ausgegeben werden. Sollen längere Zeilen ausgegeben werden, so werden sie nach 107 Zeichen abgebrochen und der Rest in eine neue Zeile geschrieben. Darum wird die Adresse (RECAREA+107) ausgerechnet und nach RECAD107 gespeichert. RECAREA ist das Feld in das später immer ein Record eingelesen wird.
Jetzt wird die Zahl der Puffer berechnet, die gebraucht werden, um jeweils 10 Records auszugeben. Dazu wird aus den Puffer-DCB die Pufferlänge geholt. Nun kann die Zahl der Puffer berechnet werden.
LRECL $\leq$ 107
$Pufferzahl_x$ = (LRECL+ST2L) .10/(Pufferlänge-4)
LRECL $>$ 107
$Pufferzahl_x$ = (LRECL+2.ST2L) .10/(Pufferlänge-4)
Rest: Pufferzahl = $Pufferzahl_x$+1
kein Rest Pufferzahl = $Pufferzahl_x$
ST2L = 7
LRECL aus LRECLAR.
Die Zahl der benötigten Puffer wird nach BUFFERZA gespeichert, so daß sie für den nächsten Durchlauf von OUTSPOOL für dieselbe Ausgabe nicht neu berechnet werden muß.
Nun folgt die Marke

HOLEBUFF Hier beginnt der Teil des Programmes, der für alle Durchläufe gilt
Mit REQBUF werden die benötigten Puffer geholt.
Sind zu wenig Puffer oder keine Puffer da oder treten Fehler auf, so wird der erste Fall in der Fehlerroutine 2, die anderen in der Fehlerroutine 3 bearbeitet. War REQBUF erfolgreich geht es weiter bei

OK3 Hier speichert man die Adresse des ersten Puffers nach BUFADS in der LINEBOX und nach ADJETZBU in die WORKAREA. Die Adresse des nächsten Puffers wird ADNEXTBU gespeichert.

Diese Speicherung der Pufferadressen, später auch die des zuletzt gefüllten Puffers nach ADOLDBU dient zum einfachen Arbeiten in der Pufferkette. Weiter lädt man das Reg. BUFENDR, das auf das Ende des Puffers zeigt. Das Reg. BUFZEIGR, das bis jetzt auf den Anfang des Puffers zeigte, wird um 4 erhöht und zeigt so auf das erste freie Byte im Puffer. Beim Füllen des Puffers zeigt dieses Reg. immer die Stelle, in die Zeichen in den Puffer geschrieben werden können. Die ersten 4 Byte jedes Puffers enthalten die Adresse des nächsten Puffers. Im letzten Puffer steht an dieser Stelle ein F'0'.
Der RECCOUNT wird nun zu 0 gesetzt. In diesem Feld wird die Zahl der eingelesenen Records gezählt.
Es folgt der Einlese-und Übersetzungsteil mit

RECEIN Zuerst wird geprüft, ob schon 10 Records eingelesen sind. Wenn ja, geht es zu ENDPOINT. Sonst wird die DCB Adresse aus DCBADR geladen und gefragt, ob die Ausgabe ein Bibliotheks-Output oder ein Programm-Output behandeln soll. Bei einem Bibliotheks-Output geht es nach EINREAD. Sonst wird bei

RECGET fortgefahren. Mit dem Macro GET wird hier ein Record vom Progr. Output DS in RECAREA gelesen. Dabei wird abhängig vom RECFM (Recordformat) eine der folgenden Operationen ausgeführt.

RECFM = U: Vor dem Einlesen wird das Feld RECAREA mit Leerzeichen aufgefüllt, nach dem Einlesen geht es nach EINEND

RECFM = UM: Wenn nur ein Steuerzeichen eingelesen wird, (der Record ist leer) d.h. im ersten Byte von RECAREA (erstes Zeichen des Records) '000000X0' X = 1 ist, so wird nach RECGET gesprungen und der nächste Record wird eingelesen. Sonst wird das erste Byte mit einem Leerzeichen überschrieben und nach EINEND gesprungen.

RECFM = UA: Erstes Byte durch ein Leerzeichen ersetzen und nach EINEND springen.

RECFM = F: Springe nach EINEND

RECFM = FM: wie UM

RECFM = FA: wie UA

RECFM = V Das erste und zweite Byte enthält die wirkliche Recordlänge (LRECL). Sie wird in LRECLAR gespeichert. Dann werden die ersten vier Zeichen des Records durch Leerzeichen ersetzt und nach EINEND gesprungen.

RECFM = VA Wie bei V, aber die ersten fünf Zeichen mit Leerzeichen überschreiben.

RECFM = VM Ist der Record nicht leer, wie bei VA, ist er leer, springe nach RECGET.

Ist ein Record eingelesen und die entsprechende Operation ausgeführt wird das Programm bei EINEND fortgesetzt.

EINREAD Mit READ wird hier bei Bibliotheks-Output ein Record eingelesen und mit dem Macro CHECK der Vorgang geprüft. Bei beiden Einlesearten (GET und READ) wird bei einem Fehler nach SEOIERR in die Fehlerroutine 5 gesprungen.

EINEND Ist der Data Set zu Ende gelesen, springt man ebenfalls in beiden Fällen nach DATAEND. Die Register, welche bei der Übertragung des Records in die Puffer verwendet werden, werden jetzt geladen. Anschließend wird der Record so übersetzt, daß er durch das Terminal ausgedruckt werden kann. Der RECCOUNT wird um 1 erhöht. Ist die Recordlänge größer 107, so wird jetzt bei V107GR fortgefahren. Sonst geht es nach N107KGN.

V107GR Hier werden die ersten 107 Bytes des Records übertragen. Über ein EX MVC werden immer so viele Bytes übertragen, wie es der noch freie Platz im gerade benützten Puffer oder die noch zu übertragenden Bytes im Record (bis 107) zulassen. Übertragen wird immer die kleinere Anzahl der beiden Möglichkeiten. Sind schon 107 Bytes übertragen, wird in die Steuerzeichen-Routine, ROUTSTZ, gesprungen. In ihm wird das 7 Byte lange Steuerzeichen eingesetzt. Das Steuerzeichen besteht aus:
Zeilenvorschub, fünf Synchronisationszeichen, Wagenrücklauf.

Als Rücksprungadresse wird in Reg. 15 A (N107KGN) geladen. Auf diese Marke wird nach Beendigung der Steuerzeichen-Routine gesprungen. Werden weniger als die ersten 107 Bytes des Records übertragen, so ist der gerade verwendete Puffer voll. Es geht in die 'Buffer Routine', die den nächsten Puffer verfügbar macht. Man springt nach NEXTBUFF und lädt als Rücksprung-Adresse ins Reg. 15: A (V107GR).

N107KGN Hier werden Record, die kleiner gleich 107Bytes lang sind und der Rest der größer als 107 langen Records übertragen. Die Übertragungsmethode ist gleich wie bei V107GR. Sind alle Bytes des zu übertragenden Records in einen Puffer geschrieben, so wird in die Steuerzeichen-Routine gesprungen (ROUTSTZ). Als Rücksprungadresse lädt man ins Reg. 15: A (RECEIN). Konnten nicht alle Zeichen übertragen werden, ist der Puffer voll, und es geht in die Routine NEXTBUFF mit der Rücksprungadresse A (N107KGN) im Reg. 15.

NEXTBUFF Das BUFFZEIGR Reg. wird mit der Adresse des nächsten freien Puffers geladen. Die Felder ADOLDBU (Adresse des letzten Puffers) ADJET2BU (Adresse des neuen Puffers) ADNEXTBU (Adresse des nächsten Puffers) werden auf den neuen Stand gebracht. Das BUFFENDR wird neu geladen, es zeigt auf das Ende des neuen Puffers. Zum Schluß wird das BUFFZEIGR Reg. um 4 erhöht, so daß es auf das erste freie Byte im Puffer zeigt.
Der Rücksprung erfolgt über Reg. 15 an die vorbestimmte Stelle.

ROUTSTR In dieser Routine wird das schon beschriebene Steuerzeichen eingesetzt.
Zuerst wird das Rücksprungregister (15) und die Record-Address-Register gespeichert. Es folgt eine Prüfung, ob der gerade verwendete Puffer voll ist. Wenn nein, wird nach BUFNF gesprungen. Sonst folgt

BUFFULL von hier wird nach NEXTBUFF gesprungen, mit der Rücksprungadresse in Reg. 15: A (BUFNF)

BUFNF Ist noch Platz im Puffer oder wurde ein neuer Puffer verfügbar gemacht, so wird jetzt das Steuerzeichen in den Puffer geschrieben.

Dies geschieht in der gleichen Art wie das Übertragen der Records. Konnte das Steuerzeichen nicht ganz übertragen werden, so wird nach BUFFULL gesprungen, um den nächsten Puffer zu holen. Ist das ganze Steuerzeichen übertragen worden, wird geprüft, ob der Puffer gerade voll wurde. Wenn ja, geht es nach BUFVOST. Sonst folgt jetzt

STEND — Die geretteten Register werden zurückgeladen und über Reg. 15 zurückgesprungen.

BUFVOST — Es wird geprüft, ob das Steuerzeichen gerade den letzten Puffer gefüllt hat. Ist dies der Fall, geht es nach STEND sonst nach NEXTBUFF mit der Rücksprungadresse A(STEND) im Reg. 15.

DATAEND — Auf diese Marke wird gesprungen, wenn der Data Set zu Ende gelesen ist oder wenn ein I/O Fehler beim Lesen auftrat. Nach ACTION+1 wird das EODADBIT gesetzt und nach ERRSTBIT das SC und CL Bit.

ENDPOINT — Auf diesen Punkt des Programmes springt man, wenn 10 Records in die Pufferkette geschrieben wurden.
In der jetzt folgenden Routine wird berechnet ob und wieviel Puffer gebraucht wurde. Die Zahl der Bytes, die in den letzten gebrauchten Puffer geschrieben wurde, wird in das erste Byte von BUFADS geschrieben. Das Adressfeld des letzten Puffers wird zu 0 gesetzt (F'0').

REBU, POINTPO — Es wird geprüft, ob noch Puffer freizugeben sind. Wenn ja, werden die nicht gefüllten Puffer mit RECBUF freigegeben. Treten bei RECBUF keine Fehler auf, wird das Programm bei LETZT fortgesetzt. Fehler werden in der Fehler-Routine 4 behandelt. Wurden alle Puffer gefüllt wird von der Prüfung nach LETZT gesprungen.

LETZT — Abfrage, ob eine Übertragung stattfand.
Wenn ja, wird nach DYNOUT in der CSECT TCP1WAIT gesprungen.
Sonst sind Fehler aufgetreten oder keine Übertragung und es geht über

LETZTFEH — zu CSECT: ENDOUT

Fehlerroutinen.

Fehlerroutine 1

Alle Fehlerausgänge von LOCATE und OBTAIN werden hier verarbeitet. Eine entsprechende Fehlernachricht wird übersetzt und in einen vorher mit RGQBUF gehaltenen Puffer geschrieben. In ACTION+1 wird das EODAD Bit und im ERRSTBIT das NA-Bit gesetzt. Steht kein Puffer zur Verfügung oder sind bei REQBUF Fehler aufgetreten,so wird ein Fehlercode nach LCC12 gesetzt und das KÜ-Bit gesetzt. In diesem Fall kann die Art des Fehlers bei OBTAIN und LOCATE nicht mehr bestimmt werden. In beiden Fällen wird nach LETZT gesprungen.

Fehlerroutine 2

Hier wird weitergearbeitet, wenn durch REQBUF zu wenig bereitgestellt werden konnte.
In ACTION+1 wird das EODAD-Bit gesetzt und in ERRSTBIT das CL-Bit. In den ersten Puffer wird die Fehlernachricht geschrieben. Dann werden die Vorbereitungen zur Freigabe der restlichen Puffer getroffen und nach POINTPO gesprungen.

Fehlerroutine 3

Sie bearbeitet die anderen Fehler, die bei REQBUF vorkommen können.
Entsprechende Fehlercodes werden nach LCC12 geschrieben. In ACTION+1 wird das EODADBIT und in das ERRSTBIT das KÜ- und CL-Bit gesetzt. BUFADS wird zu 0 (F'0') gesetzt und nach LETZT gesprungen.

Fehlerroutine 4

Diese Routine behandelt die Fehler von RELBUF. In LCC12 wird ein Fehlercode gesetzt und dann nach LETZT gesprungen.

Fehlerroutine 5

Sie behandelt die Fehler, die bei GET und READ auftreten.
Ein Fehlercode wird nach LCC12 gesetzt und bei READ im SEQDCB die Fehlerindikatoren zurückgesetzt.
Zum Schluß wird nach DATAEND gesprungen.

2.2. ENDPUT

Die CSECT ENDPUT wird aufgerufen, wenn der letzte Puffer bei EODAD ausgegeben ist oder wenn keine Übertragung stattfand (Fehler).
Die aufrufende CSECT ist TCP1WAIT oder OUTSPOOL, wenn keine Übertragung stattfand.
Abhängig von der Art des Output's, Bibliotheks- oder Programm-Output, und von den im ERRSTBIT gesetzten Bits, werden verschiedene Operationen durchgeführt. Bei Bibliotheks-Output wird ein CLOSE TYPE=T auf den SPOOL-DS gemacht und nach ENDZ gesprungen. Bei Programm-Output werden die Operationen durch das ERRSTBIT gesteuert. Es wird ausgeführt bei:

'01000000'	CLOSE und FREEPOOL	springe nach ENDZ
'00001000'	Nur CATLG (uncatalog)	" " ENDZ
'00000001'	Keine Operationen, es wurde keine Ausgabe sondern nur Nachrichten ausgegeben	" " ENDZ
'000001X0'	CLOSE und FREEPOOL und wenn X=1 auch SCRATCH u. CATLG (uncatalog)	" " ENDZ

Treten bei SCRATCH oder CATLG Fehler auf, so wird nach LCC12 ein Fehlercode geschrieben und nach ENDZ gesprungen.

ENDZ Hier wird der INECB gepostet. Dies ist das Zeichen dafür, daß die Ausgabe beendet ist.
Zum Schluß wird die WORKAREA mit FREEMAIN freigegeben. In ACTION+1 werden die letzten 3 Bits das EODAD-, PL/1-Output- und SWITCH-Bit gelöscht.
Die Register werden zurückgeladen und über BR15 nach TCP1WAIT zurückgesprungen.

3. TCP1SPOL: (H. Santo)

Zusammenfassung:

TCP1SPOL ist ein Bestandteil des Terminal Control Programmes (TCP).
Seine Funktion innerhalb des TCP siehe Blockdiagramm.
Das Programm arbeitet parallel zum E/A-Vorgang und wird vom Modul TCP1WAIT temporär initialisiert. Es übernimmt folgende Aufgaben:
Analyse und Umschlüsselung von Lochstreifencodes,
Bearbeiten und Formatieren der in dynamischen Puffern übergebenen Daten,
Blockung und Transport der Daten auf einen sequentiell organisierten Zwischenspeicher (Spool Data Set),
Prüfung auf fehlerfreien Abschluß des Einlesevorganges (TIMEOUT, ETX,EOT),
Prüfung auf Sonderzeichen zur Steuerung der Blockung und Korrektur,
Rückgabe der bearbeiteten Puffer an den gemeinsamen Pufferbereich der Terminals,
Anfordern und Füllen dynamisch geketteter Fehlerpuffer zur Ausgabe von als fehlerhaft erkannten Eingaben,
Synchronisationsaufgaben mit dem Modul TCP1WAIT bei TIMEOUT.

Aufruf:

Der Aufruf des Programmes TCP1SPOL erfolgt über den residenten Modul TCP1WAIT. Das Programm TCP1SPOL steht als Load Modul auf der Bibliothek TCP1.LINKLIB und wird über ein ATTACH vom residenten Modul TCP1WAIT aufgerufen, wenn der Benutzer Lesen oder Modifizieren von Daten verlangt, d.h. durch Eingabe der Befehle am Terminal

LIES mit Geräteangabe LOCHSTREIFEN
LIES mit Geräteangabe FERNSCHREIBER

oder dem Befehl

MODIFIZIERE

Da einer leistungsfähigen DVA relativ langsame Außenstationen gegenüber stehen (Lochstreifenleser 200 Baud) kann die Abarbeitung der Puffer mit der Vergabe der Puffer für E/A-Vorgänge Schritt halten, so daß immer Puffer im Pufferpool für die Eingabe vorhanden sind.

Mögliche Codes:

Die Eingabe der Daten erfolgt entweder über Tastatur oder Lochstreifenleser, wobei die interne Darstellung und die Bearbeitung der Daten unabhängig vom Eingabemedium (Lochstreifenleser oder Tastatur) bleibt.
Unter folgenden Codes kann der Benutzer durch Angabe eines Parameters auswählen:

ASCII (CCITT Nr. 5, ISO7) gerader Parität
PTTC (BCD)
CCITT Nr. 2 (Fernschreibcode)KA5

Bei Eingabe über die Tastatur wird CCITT Nr. 5 erzeugt.
Da auf benutzerfreundliche Implementierung Wert gelegt wurde, wird auf Eingabe eines Vorstreifens zur Definition der Übersetzungstabellen verzichtet und über Parameterangabe (s.o.) unter vorher aufgebauten Codetabellen ausgewählt.
Die Benutzung anderer Codes kann auf folgende Weise ermöglicht werden:
Durch Angabe des Parameters BIN wird bewirkt, daß die Codezeichen als Rohdaten auf den sequentiellen SPOOL-Datensatz transportiert und nicht formatiert werden.
Zur weiteren Aufbereitung spezieller Lochstreifencodes stehen Spezialroutinen in Form von Blitzstartmoduln zur Verfügung.
Bis jetzt können folgende Codes eingelesen, übersetzt und formatgerecht aufbereitet werden (Übersetzung in INTERN-Code und Blockung):
CAE-Code, Loadformatlochstreifen von der Midas-Anlage erzeugt, in 12 und 24 bit Modus. /11/
Näheres siehe weitere Beschreibungen.

Einschränkungen:

Durch die Hardware bedingt können die nachfolgenden Codezeichen nicht als Datenzeichen verwendet werden.
Sie entsprechen im Code CCITT Nr. 5 den Steuerzeichen:

ENQ	0000	0101
ETX	0000	0011
ACK	0000	0110

Da Steuer- und Datenübertragung nicht getrennt sind (halbduplex Betrieb mit der Controleinheit Multiplexor IBM 2702), sind in allen anderen Codes diese Bitmuster zur Darstellung von Datenzeichen verboten (quasitransparent).
Das Zeichen

NUL 0000 0000

darf bei allen Codes außer CCITT Nr. 5 nur als Blockendezeichen verwendet werden (bewirkt ETX an der 2702). Um trotzdem das Einlesen von 5-Kanalcodes zu gestatten, die das Zeichen NUL als gültiges Datenzeichen benutzen, wird bei 5 Kanal-Codes mit weniger als 7 Kanälen (bei schmälerem Lochstreifenpapier) von der Datenstation das 7. Bit automatisch bei allen eingegebenen Zeichen gesetzt, d.h. das Zeichen

NUL 0000 0000

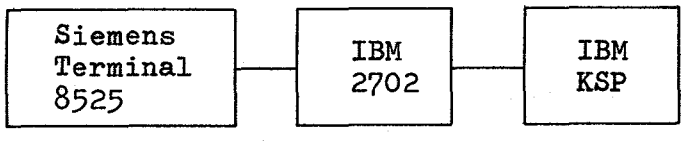
verläßt die Datenstation als

NUL* 0100 0000

und wird dann nicht als ETX von der 2702 gewertet.
Da die verwendeten Steuerzeichen, wie der an der Datenstation bei Eingabe von Daten über die Tastatur erzeugte Code, gerade Parität besitzen, ist volltransparentes Einlesen aller Codes ungerader Parität möglich und auch solcher Codes gerader Parität, die diese vorher erwähnten Steuerzeichen nicht als gültige Datenzeichen enthalten.

Übersetzen in INTERN-Code (EBCDIC)

Der Codewert eines Datenspeichers im Kernspeicher entspricht nicht der genormten Darstellung für IBM Kernspeicher, vielmehr sind die Wertigkeiten der einzelnen Bits in ihrer Reihenfolge vertauscht (Spiegelung).



Beispiel: ETX B'0000 0011' ⟶ B'1100 0000'
Dieser "Siegeleffekt" muß bei den Übersetzungstabellen bei allen Codes, auch bei quasitransparentem Code (PARAMETER BIN) berücksichtigt werden.
Zusätzlich ist bei allen Codes außer CCITT Nr. 5 noch folgendes zu beachten:
Die Steuerfunktionszeichen im CCITT Nr. 5 Code EØT, NUL, NAK werden nicht mit ihren Codewerten übertragen, sondern entsprechend in der internen Darstellung den Zeichen

EØT ⟶ NUL
NUL ⟶ ETX } Bedeutung in CCITT Nr. 5
NAK ⟶ ACK

Aufbau der Übersetzungstabellen

CCITT Nr. 5 bzw. ISO-7 Code

Die Übersetzungstabellen für Eingabe und Ausgabe in CCITT Nr. 5 Code (ISO-7 Code) sind als Macro auf der TCP.MACLIB unter dem Namen SIETRAB zu finden. Soll die Tabelle geändert werden, so muß der Modul TCP1BASE neu assembliert und als Loadmodul auf TCP.LINKLIB gebracht werden, da dieser Modul die Expansion des Macros enthält und die Adresse der Übersetzungstabelle für die Wandlung von CCITT Nr. 5 in EBCDIC nach TTIN und die Adresse der Übersetzungstabelle für die Ausgabe nach TTØUT der KONBØX bringt.
Allen Zeichen, die nur in ISO7 oder EBCDIC oder in keinem der beiden Codes definiert sind, wird bei der Eingabe die hexadezimale Wertigkeit 3F (= SUB in EBCDIC) zugeordnet. Dadurch ergibt sich eine beschränkte Prüfmöglichkeit. Da TCP1SPØL auf Sonderzeichen SUB (X'3F') prüft, wird immer dann ein Fehler festgestellt, wenn aus einem zugelassenen Codezeichen ein im EBCDIC-Code nicht definiertes Codezeichen entsteht. Bei der Tabelle TTØUT, welche die Rückübersetzung von EBCDIC in CCITT Nr. 5 beschreibt wird allen in CCITT Nr. 5 nicht definierten Codezeichen X'44' zugeordnet, was dem Symbol an der Datenstation entspricht.

Abweichend von der Norm entspricht die Bedeutung des Zeichens DC4 im ISO-Code dem Zeichen NL im EBCDIC-Code. Die beiden für die PL/1-Ein- und Ausgabe benötigten Zeichendarstellungen für 'not' und 'or' werden wie folgt dargestellt.
Den Zeichen ^ und ! der Siemenstastatur entsprechen in maschineninterner Darstellung die Zeichen ¬ und |.

BCD - Codetabelle

In BCD - Code oder in EBCDIC - Code nicht definierte Zeichen werden als X'3F' dargestellt.
Zusätzlich zum Zeichenvorrat der BCD-Zeichen können noch die ISO7-Steuerzeichen STX, EOT, ETX vorkommen. Die Übersetzung dieser Steuerzeichen in den Intern-Code wird wie nachfolgend beschrieben vorgenommen

STX	X'41'	⟶ X'02'
EOT	X'21'	⟶ X'37'
ETX	X'C0'	⟶ X'03'

STX darf nur im zuerst angeforderten Puffer als 1. Zeichen vorkommen.

CCITT Nr. 2

Besonderheit: 5 - Bit - Code
Doppeldeutigkeit der Codezeichen
Die Codezeichen Buchstabenumschaltung und Ziffernumschaltung bestimmen die gültige Bedeutung des nachfolgenden Zeichens bzw. der nachfolgenden Zeichenkette.
Die Übersetzung erfolgt mit Hilfe zweier Tabellen, die über die Umschaltzeichen ausgewählt werden.
Beide notwendigen Tabellen sind zu einer Tabelle zusammengefaßt, und sind so ineinander verschachtelt, daß sich Zeichen verschiedener Bedeutung nicht überlagern.
Die Tabellenanfangsadresse der Teiltabelle, welche die Bedeutung der Zeichen bestimmt, wenn Ziffernumschaltung vorhergeht, liegt um ein Displacement von 2 zur Tabellenanfangsadresse verschoben. (Hammingdistanz 2)
Innerhalb jeder Tabelle können außer den in CCITT definierten Codezeichen die ISO-Steuerzeichen EOT und ETX vorkommen. Siehe bei BCD-Code.

BINÄR Tabelle

Bei allen 2^8 möglichen Zeichen werden die Wertigkeiten der einzelnen Bits vertauscht. (Spiegeleffekt der 2702). Zusätzlich werden die ISO7-Steuerzeichen EOT und ETX berücksichtigt. Sie dienen als Endezeichen für den eingegebenen Datenblock und stellen die einzigen vom Programm geprüften Sonderzeichen dar.
Wird mit anderen als den oben definierten Code gearbeitet, so muß als Parameterangabe für den Code beim Befehl LIES BIN angegeben werden. Dadurch wird bewirkt, daß die über den Lochstreifenleser eingelesenen Codezeichen als Rohdaten auf den sequentiell organisierten Zwischenspeicher (Spool-Datensatz) transportiert werden (in Blöcken von 80 Byte) und von dort über die entsprechende Schreiberoutine auf der benutzereigenen Bibliothek unter dem vom Benutzer spezifizierten Membernamen aufzufinden sind.
Beispiel eines vom Terminalbenutzer eingegebenen Kommandos, bei welchem vom Programm TCP1SPOL die Tabelle BINART ausgewählt wird.

L, LOCH, DATA, NUM, BIN

Die Umschlüsselung der Daten in den internen Maschinencode (EBCDIC) kann über spezielle Blitzstartroutinen erfolgen.

Eingabe - Formate

Für die Eingabedaten sind Formate zulässig. Durch Angabe der entsprechenden Parameter beim TCP-Befehl LIES wird ein Format vom Benutzer ausgewählt und über die Lineboxparameter WADR4 und WCODE4 durch den Entschlüssler dem Programm TCP1SPOL übermittelt.

Format 1: ONU
WCODE4 = X'02'
BCODE4 = X'07'
Es werden bis zu 80 eingegebene Zeichen pro Zeile abgelegt.
(Anwendungsbeispiel: Meßdateneingabe)

Format 2: NUM

BCODE4 = X'07'

WCODE4 = X'02'

Die über Lochstreifenleser oder über die Tastatur eingegebenen Daten werden auf Kartenformat gebracht (maximal 80 Zeichen/Zeile), davon sind bis zu 72 Byte Informationszeichen, die letzten 8 Byte (Zeichen 73 - 80) enthalten die Zeilennummer, die mit 10 beginnt und automatisch in Inkrementen von 10 weitergezählt wird. (Anwendungsbeispiel: Programmeingabe mit Zeilenkennung)

Format 3: MODIFIZIEREN von Daten wird verlangt

BCODE4 ≠X'07'

Die Anzahl der Daten pro Zeile ist maximal 80, wobei die Numerierung an beliebiger Stelle innerhalb der 80 Byte steht. Sie wird gekennzeichnet durch das Zeichen und kann bis zu 8 Zeichen umfassen.

Beispiel:

MOD,..... eingegebener Befehl am Terminal

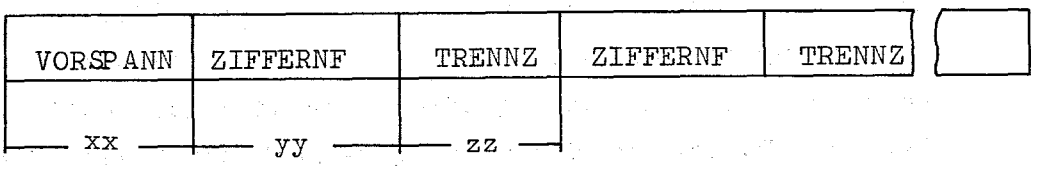
Eingabedaten am Terminal:

SYSIN DD *≠1020

Das formatgerechte Einfügen der Numerierung in die Spalten 73 - 80 übernimmt das Unterprogramm FORMI.

Format 4: Formatangabe beim Befehl LIES

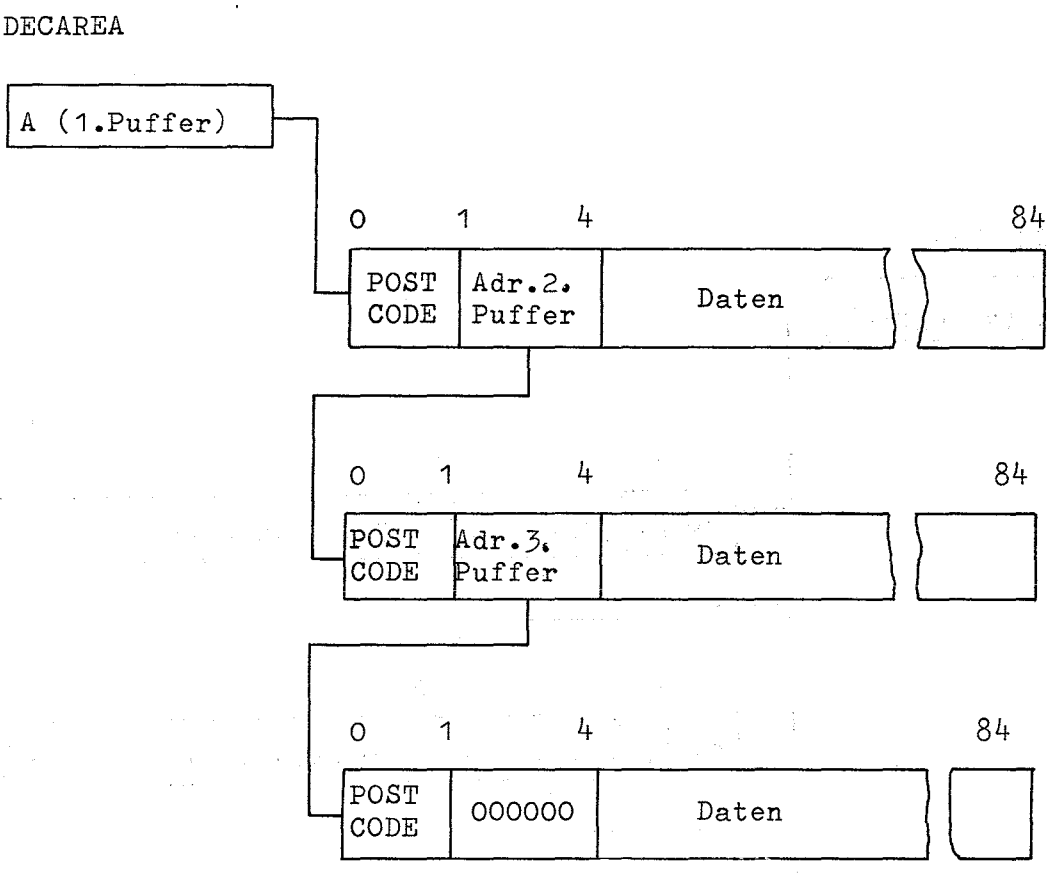
Format 4 wird verwendet zur Formatprüfung von Eingabedaten (Beisp.: Meßdaten von Spektren auf Lochstreifen), die aus einer periodischen Folge von Ziffern und Trennzeichen bestehen. Ein nicht zyklisch zu interpretierender Zeichensatz bis zu 99 Zeichen kann den zyklisch sich wiederholenden Eingabedaten vorhergehen.



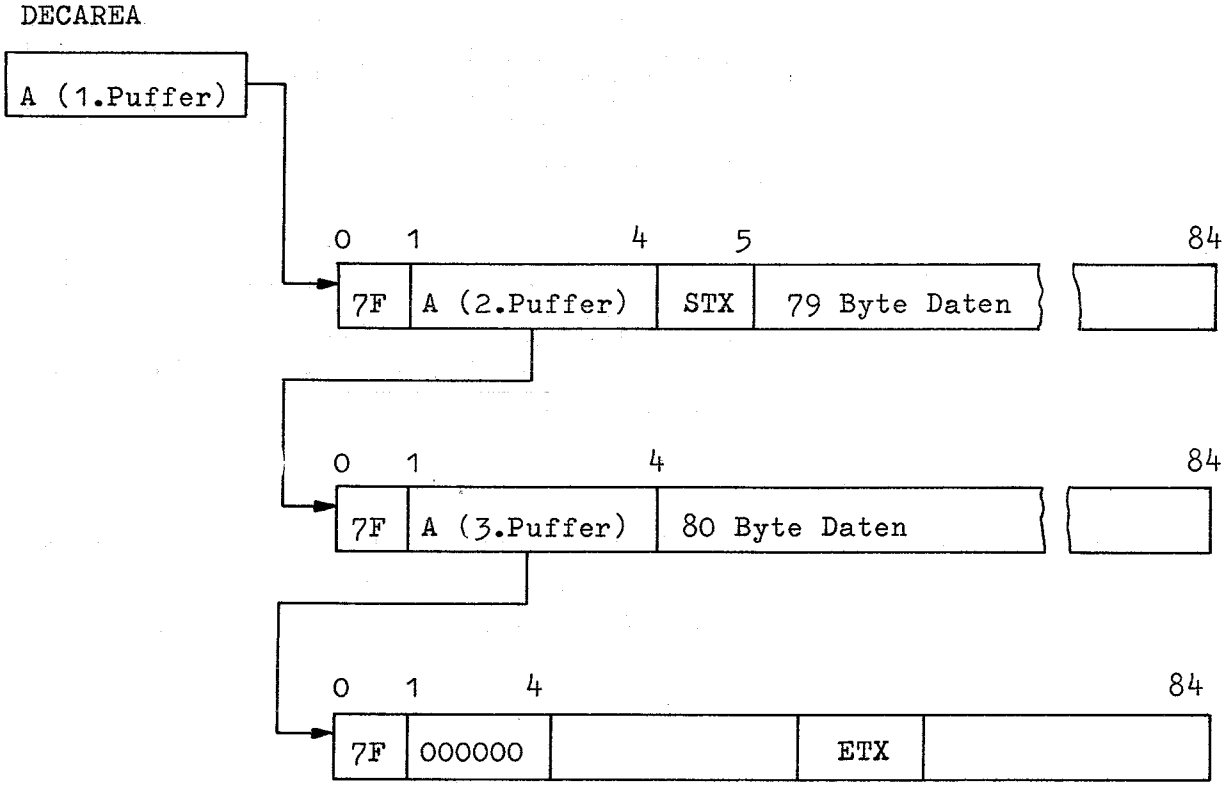
Format 4 beinhaltet zusätzlich entweder Format 1 oder Format 2.

Dateneingabe

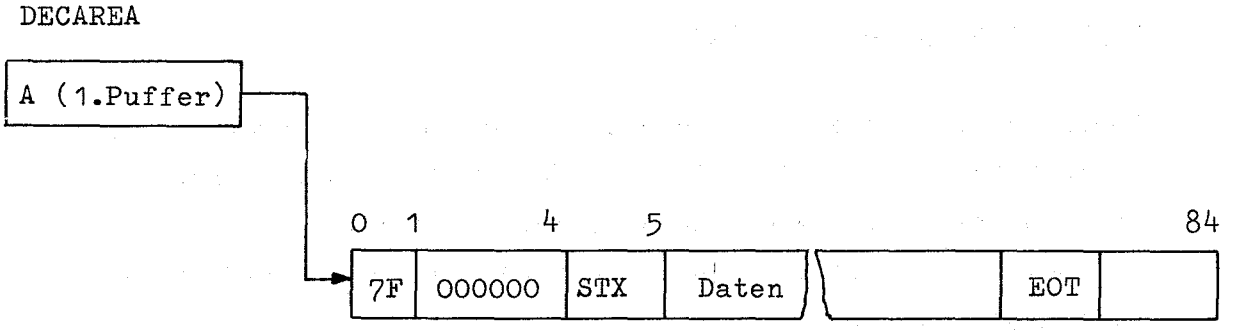
Die Übergabe der Daten an das Programm TCP1SPOL erfolgt in dyn. geketteten Puffern. Die Adresse des 1. Puffers der Kette steht unter der Linebox Adresse DECAREA.



Byte 1 - 4 eines jeden Puffers enthält eine Adresse, die auf den Anfang des nächsten Puffers zeigt. Der letzte Puffer der Kette enthält als Kettungsadresse hexadezimale Nullen. Das 1. Byte des Puffers wird als Ereignisindikator (Event Control Block) benützt. Ist der Puffer gefüllt, so wird der Indikator (POSTCODE) auf X'7F' gesetzt, wenn der Einlesevorgang exakt abgelaufen ist. Traten I/O-Fehler auf, so wird der POSTCODE X'41' gesetzt.
Der Datenbereich der gültigen Pufferkette beginnt mit STX und endet bei fehlerfreiem Abschluß der Eingabe mit ETX.

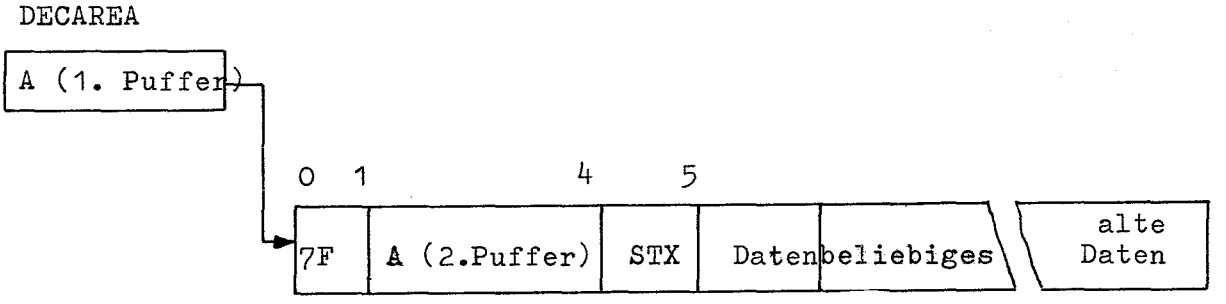


Traten Fehler bei der Informationseingabe auf, (d.h. Parity-Fehler wurden vom Terminal festgestellt, oder Terminal ist nicht mehr betriebssicher), so endet der Informationsblock mit EOT.



Läßt der Terminalbenutzer eine zu große Pause (>25 sec) bei der Eingabe von Daten verstreichen, oder wird das Zeichen ETX von der DVA nicht erkannt, so wird der Informationsblock weder mit ETX noch mit EØT beendet (kein Endesymbol)

→ TIMEØUT



Timeout wird über den Parameter DECSENSØ der Linebox angezeigt. DECCØUNT (Residual count) gibt die Anzahl der noch freien Bytes im zuletzt angeforderten Puffer an. DECSENSØ und DECCØUNT sind Parameter eines Data-Event-Control-Blockes der Zugriffsmethode BTAM, und werden über die LINEBOX für das Programm TCP1SPOL zur Programmablaufsteuerung verfügbar.

Übersetzen der Eingabedaten

Übersetzt werden die einlaufenden Daten in den Eingabepuffer pufferweise (80 Byte), mit Ausnahme von CCITT Nr. 2 (Fernschreibcode), der zeichenweise übersetzt wird.
Übersetzen geschieht mit Hilfe der oben aufgestellten Übersetzungstabellen.

Prüfen der Eingabedaten

Die Eingabedaten enthalten außer den Informationszeichen, die den gewünschten Text ergeben, Formatsteuerzeichen, die extern auf der Eingabeseite zur Steuerung des Datenschreibers (oder entsprechend vom Lochstreifenleser) benützt werden. Sinngemäß müssen diese Formatsteuerzeichen zur Bildung des Kartenformates (80 Byte-Records) auf den internen Einheiten beitragen.

Sonderzeichen

An Sonderzeichen werden behandelt:
a) DELETE
b) SUB
c) CANCEL
d) BSP

Die oben aufgeführten Sonderzeichen sind zusätzlich zu den Endezeichen und den Formatsteuerzeichen in der Translate-und Test-Tabelle berücksichtigt.
Da erst nach dem Übersetzen auf die Sonderzeichen geprüft wird, liegen bei allen Codes diese Zeichen in derselben Intern-Darstellung (EBCDIC-Code) vor. Das hat den Vorteil, daß die Prüftabellen für die Codes: CCITTNr.2, CCITTNr.5 und PTTC identisch sind*.

Aufbau der Prüftabellen ("Translate- und Testtabellen")
bei nichtformatierter Eingabe: bei Codeangabe BIN.
Tabelle besteht aus 256 Byte Bereich. Jedem Sonderzeichen wird an der Stelle, die seiner Wertigkeit in der Tabelle entspricht ein Wert ungleich X'00' zugeordnet.
Die zu behandelnden Sonderzeichen besitzen folgende Werte

Zeichen	in Tabelle eingetragener Hexadezimalwert	Increment (dezimal) bezogen auf Tabellenanfang
SUB	04	63
DELETE	08	7
LINEFEED	08	37
ETX	0C	3
CR	10	13
NL	10	21
CANCEL	14	24
(BACKSPACE)	18	

Beim Einlesen unter Angabe des Parameters BIN genügt es, auf die Begrenzungszeichen (EOT und ETX) und TIMEOUT zu prüfen. EOT wurde, um Mehrdeutigkeit der Zeichen zu vermeiden, in X'00' übersetzt. Das Zeichen NUL (X'00') wird bei nicht CCITTNr.5 Codes nie als gültiges Codezeichen vorkommen, da dem im Kernspeicher ankommenden Zeichen NUL das Datenzeichen EOT (Wertigkeit in CCITTNr.5 = X'37') entspricht. Im Zeichenvorrat für BIN-Zeichen darf die Codedarstellung für ETX nicht enthalten sein, so daß ETX dem Hexadezimalwert X'03' entsprechen kann.
Für die Prüftabelle ergibt sich daraus folgende Darstellung:

Zeichen	Hexadezimalwert in der Tabelle	Increment bezogen auf Tabellenanfang TABKEINF (dezimal)
EOT	18	0
ETX	0C	3

TABKEINF Da die Prüftabellen für quasitransparenten Code und die weiteren zugelassenen Codes nicht identisch sind, werden die Tabellen dynamisch zur Laufzeit aufgebaut (TABKEINF).

TABTESTZ Bei formatierter Eingabe (Eingabe bestehend aus Vorspann periodisch aufeindanderfolgenden Zifferngruppen mit Trennzeichen) wird die Gültigkeit der Zeichen innerhalb der Zifferngruppe ebenfalls mit Testtabellen geprüft (TABTESTZ). Die zugelassenen Ziffern (0-9) und die Zeichen + und - erhalten den Wert X'00'. Auch innerhalb der Zifferngruppe können alle oben erwähnten Sonderzeichen vorkommen. Diese erhalten Werte zugeordnet, die denen der Prüftabelle für ASCII-Codes identisch sind.

Ausgabe der Daten auf den Spooldatensatz

Die Ausgabe der Daten auf den Spool-Daten-Satz des speziellen Benutzers erfolgt im Wechselpufferbetrieb. Die Ausgabepuffer umfassen 80 Zeichen. (Kartenformat!)

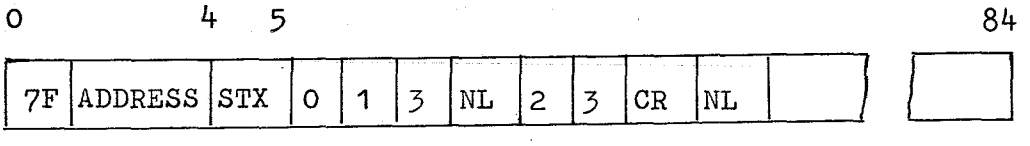
Wirkung der Formatsteuerzeichen bei CCITT Nr. 5, CCITT Nr. 2 und PTTC - Codes auf das Programm TCP1SPOL

NL CR / LF	Bedeutung für den Datenschreiber Wagenrücklauf + Zeilenvorschub

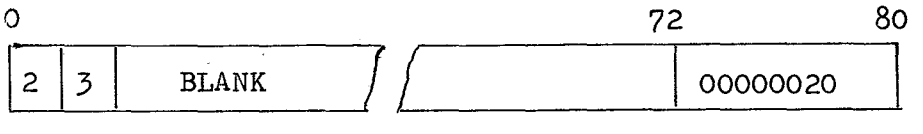
Die interne Darstellung im Recordformat (80 Byte) wird über die Steuerzeichen NL bzw. LF/CR gesteuert (Bsl. a). Bei Dateneingabe, die mehr als 80 Zeichen bei Format 1 und länger als 72 Zeichen bei FORMAT 2 sind, erfolgt die Darstellung eines neuen Records automatisch nach dem 72. (Bsp. b) bzw. 80. Zeichen (Bsp. c). Die Endezeichen STX und EOT üben ebenfalls Formatsteuerfunktionen aus. (Bsp. d)

Bsp. a)

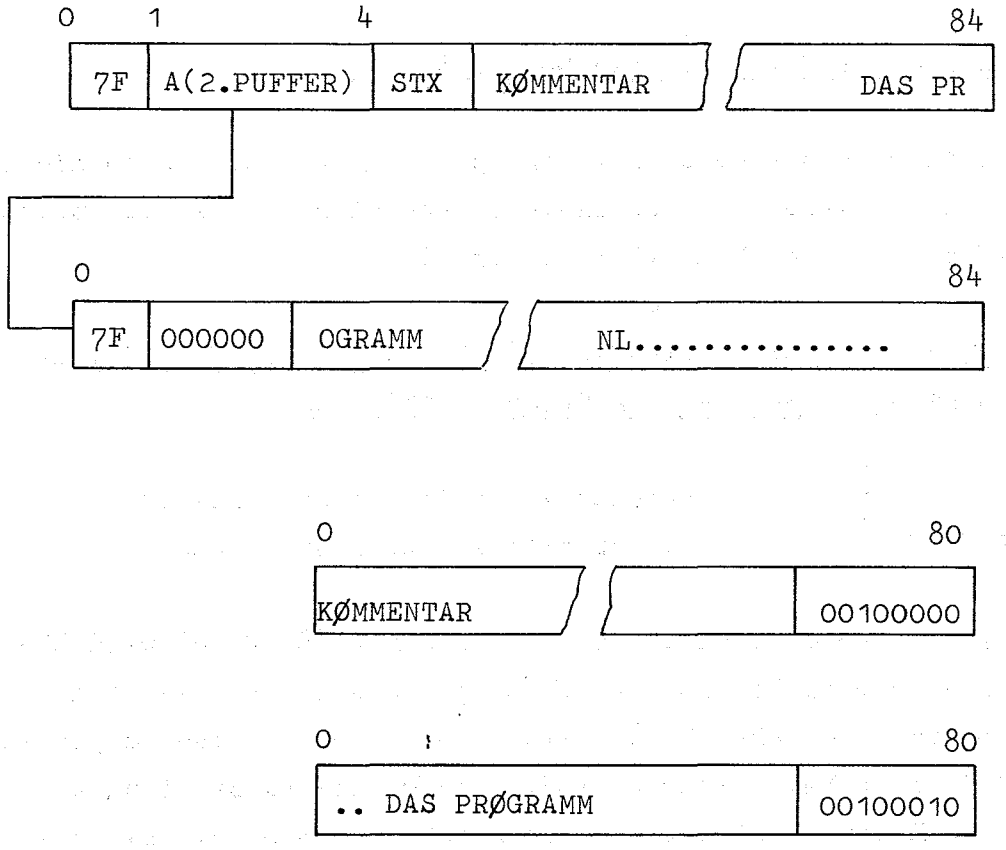
Eingabepuffer: Vom Terminaloperateur wird Numerierung verlangt.



Ausgabepuffer:

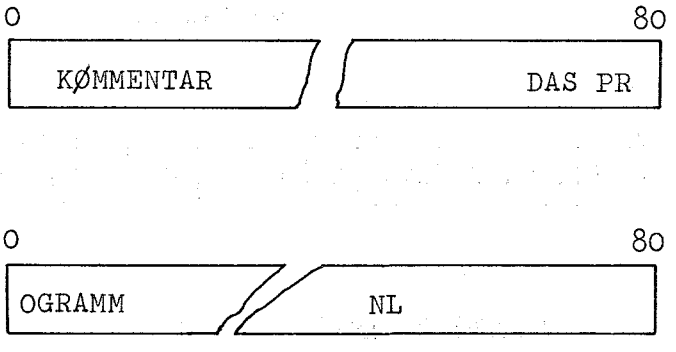


Bsp. b)



Bsp. c)

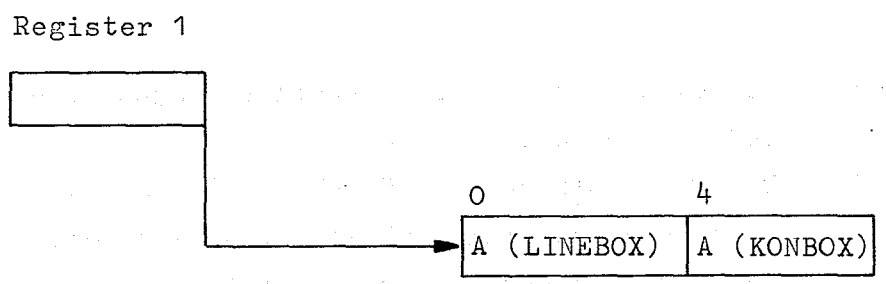
Eingabepuffer wie unter **Bsp.** c)



Alle an der Datenstation eingegebenen Daten der oben angegebenen Codes mit Ausnahme von quasitransparenten Codes (Code=BIN) werden in den EBCDIC-Code umgewandelt. Nur dann stehen dem Benutzer alle von TCP gebotenen Möglichkeiten für Dateimanipulationen und Programmübergabe in die Eingabewarteschlange der Stapelverarbeitung zur Verfügung.

EINGANGSPARAMETER

Beim Aufruf von TCP1SPØL zeigt Register 1 auf eine Parameterliste, welche die Adresse der Konbox und der aktuellen Linebox enthält.



Aus der KØNBØX werden folgende Parameter entnommen:

TTIN Adresse der Übersetzungstabelle
ASCIINr. 5 → EBCDIC

TTØUT Adresse der Übersetzungstabelle
EBCDIC → ASCII Nr. 5

DDCB DCB-Adresse für den dynamischen Daten-Control-Block der Siemens-Leitungsgruppe.

Eingangsparameter aus der Linebox entnommen:

SEQDCB — Daten-Kontroll-Block zur Beschreibung des Spool-Datensatzes

SPDSPLAD — Anfangsadresse (relative Spuradresse) des Zwischenspeicherbereiches (SPOOL-DATENSATZ) für das zugeordnete Terminal.

DECAREA — Anfangsadresse der Pufferkette, welche die eingelesenen Daten enthält.

DECSENSØ — Sense-Information der Kontroll-Einheit
DECSENSO = X'01' ≙ TIMEØUT

DECCØUNT — Residual-Byte-Count für Puffer bzw. CCW

LBXTIME — CPU-Zeit pro Terminalsitzung

BCODE

X'10' — Der extern vom Terminalbenutzer eingegebene Befehl war MODIFIZIEREN

X'07' — Der Terminalbenutzer wünscht Einlesen von Daten von externen Einheiten (Tastatur oder Lochstreifenleser) in seine Bibliothek

WCODE4

X'01' — Formatfreie Eingabe

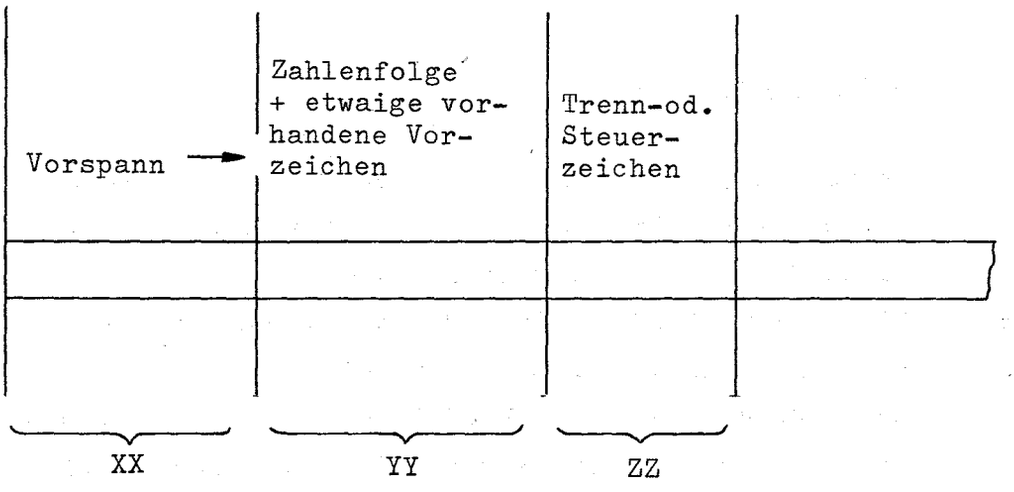
X'02'

X'03' — Eingabe mit Formatbeschreibung

X'04'

WADR4: Bei formatierter Eingabe gibt WADR4 die Adresse eines Formatbeschreibungsfeldes an.

Bei Eingabe mit Formatbeschreibung handelt es sich um Daten folgender Struktur:



XX = Anzahl der Zeichen im Vorspann 100

YY = Anzahl der Zeichen der Zifferngruppe

ZZ = Anzahl der Sonderzeichen (Trenn- oder Steuerzeichen)

Beispiel einer Eingabe:

```
MESSUNG VOM 1.7.1969   U   U
                      LF  CR
+1 3 7 9 5 U U 2 5 3 6 7 U U
         LF CR         LF CR
```

Die Formatangabe des Benutzers beim Befehl "LIES" sieht dann wie folgt aus:

```
22.06.02
XX.YY.ZZ
```

WCODE5: Kennzeichnet den vom Benutzer verwendeten Eingabecode:

X'01'	ASSCII-Code
X'02'	PTTC-Code
X'03'	BIN (quasitransparenter Modus)
X'04'	CCITT Nr. 2

AUSGANGSPARAMETER

REQBUF Enthält die Anfangsadresse einer dynamisch geketteten Pufferkette, bestehend aus maximal 5 Puffern, wenn vom Programm TCP1SPOL Fehler festgestellt wurden.

OUTECB Wird bei TIMEOUT von TCP1SPOL zur Synchronisation mit dem residenten Modul TCP1WAIT verwendet.

DECAREA Bei TIMEOUT wird von TCP1SPOL in DECAREA die Adresse eines neu angeforderten Puffers aus dem gemeinsamen Pufferbereich an die Controlsection TCP1WAIT des Moduls TCPBASE übermittelt.

REGISTER15 enthält entweder 0 bei fehlerfreiem Abschluß der Spooltask oder Returncodes.

Detaillierte Beschreibung:

Das Programm TCP1SPOL trägt das Attribut reentrant. Deshalb müssen alle Indicatoren, Zähler, Adressen und Ausgabebereiche in einem Arbeitsspeicher (WORKAREA) abgelegt werden, der für jeden Programmdurchlauf neu angefordert wird. Macros, die bei der Expansion Listen erzeugen, müssen in Listen- und Ausführungsformen verwendet werden. Bei der Einordnung der Karten ist darauf zu achten, daß die Adressen der Ausgabebereiche OUT1, OUT2 und NEUFELD ein Displacement von 128 besitzen, da die Auswahl der Ausgabebereiche für Wechselpufferbetrieb hier durch Negieren des 8.Bits der Adresse NEUFELD erfolgt.

TCP1SPOL Das Laden der Basisregister für den Programmbereich und den Arbeitsbereich, Aufbau und Verkettung der Bereiche, in denen die Registerinhalte abgespeichert werden, besorgt das Macro ALEPH.

Bei der Initialisierung werden die Ausgabebereiche (OUT1, NEUFELD, OUT2) mit Leerzeichen gefüllt und die Indikatoren OUT1+84, OUT2+84, NEUFELD+84, die fehlerhafte Puffer kennzeichnen mit X'00' überschrieben, ebenso die Indikatoren für aufgetretenes TIMEOUT (SENS@) und PUMERK.

Das Gerüst der Prüftabelle TABKEINF wird aufgebaut (256 Byte-X'00'). Die Adressen der LINEBOX und KØNBØX und der Übersetzungstabellen für ISØ7 → EBCDIC (TTIN) und umgekehrt (TTØUT) und die Parameter WCODE4, WCODE5 und BCØDE werden in den Arbeitsspeicherbereich übernommen. In den in der Linebox aufgebauten Datensatz werden die Adressen der Fehlerroutinen eingetragen, welche bei Datensatzende (EODAD) und bei I/O Fehlern (SYNAD) angesprungen werden. Die Anzahl der Bytes im Ausgabepuffer wird auf 72 festgelegt (LG = 72) und über WCODE5 bestimmt, ob quasitransparent eingelesene Daten vorliegen (WCODE5=X'03') oder Daten eines definierten Codes. Bei der Codeangabe CODE=BIN wird eine Prüftabelle aufgebaut, die nur EOT und ETX als zu prüfende Zeichen enthält.

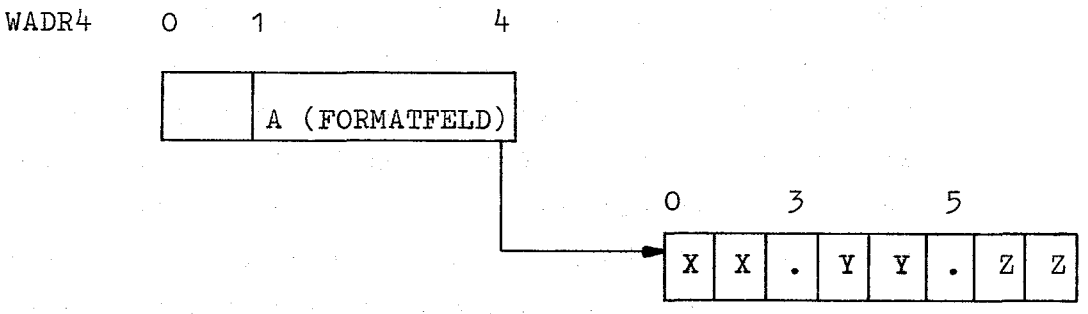
TRITSCH Aufbau der Test-Tabelle für ASCII und PTTC-Codes wie zu Beginn beschrieben. Wird Modifizieren von Benutzereinträgen verlangt (FBCODE = X'10'), so liegt der Eingabecode fest. (ASCII) Es handelt sich um Format 3. Bearbeitung bei NO/FØR.

CODEPR Der Inhalt von WCODE5 bestimmt die zu verwendete Übersetzungstabelle. Der Parameter TABELLE enthält die Adresse der ASCII Übersetzungstabelle. Für nicht ISOCodes wird TABELLE mit der Anfangsadresse der zugehörigen Übersetzungstabelle überschrieben.

X'01'	CCITTNr. 5	TABELLE	TTIN
X'02'	PTTC	TABELLE	BCDT
X'03'	BIN	TABELLE	BINART
X'04'	CCITTNr.2	TABELLE	KA5TAB

Die Tabellen stehen in einer weiteren CSECT TABLE, so daß für ihre Adressierung ein zusätzliches Basisregister (R12) geladen und wieder freigegeben werden muß.

ANFANGA Bei Eingabe mit Formatbeschreibung bestimmen die Anzahl der Ziffern (YY) und die Anzahl der Zwischenräume (ZZ)(Adresse des Formatfeldes aus WADR4 oder LINEBOX) des Formatbeschreibungsfeldes die Anzahl auszugebender relevanter Zeichen pro Puffer (bzw. Record).



XX, YY und ZZ sind Dezimalzahlen in gezonter Darstellung mit nachfolgender Bedeutung

XX = Anzahl der Zeichen im Vorspann <100

YY = Anzahl der Zeichen der Zifferngruppe

ZZ = Anzahl der Sonderzeichen (Trenn- und Steuerzeichen)

Für die Periodenzahl pro Zeile gilt:

A = F / (ZZ+YY) für (ZZ+YY) = 80 bzw. 72

F = 80 für Format 1

F = 72 für Format 2

Die Anzahl der Bytespro Zeile bzw. pro Puffer (ANZAHL)

$$\left[A/(ZZ+YY)\right] \cdot (ZZ+YY)$$

Die obige Festlegung garantiert eine einheitliche Darstellung der Ziffernfolgen pro Zeile.

EM Anzahl der Ziffern (binär)

EN Anzahl der Zwischenräume (binär)

ANZAHL " der Perioden pro Zeile (bzw. Puffer) (binär)

NØFØR Die Adresse des 1. Eingabepuffers ist über DECAREA der LINEBOX zugänglich. Das Macro POINT legt die relative Anfangsadresse auf dem Spooldatensatz für die aufzuzeichnenden Daten fest.

CNE Für BCODE = X'10' und WCODE4 = X'02', d.h. für Daten ohne **Numerierung** wird LG mit 80 überschrieben.

CE Listform für WRITE wird im Arbeitsspeicher abgelegt und verschiedene Indikatoren gelöscht. (X'00')

PUVORHAN
WAITT
NØTEINS
NACHTIMØ
NEUBASA
ETXV
EOT
CANCELT
LINECRV
DELETEV
SUB
NU

Das Feld RECBUFAD und FEHLBUF muß überschrieben werden, damit von TCP1WAIT nicht versucht wird, eine Ausgabe (RECBUFAD) auszustoßen, obwohl keine Puffer vorhanden sind. Das Programm wartet bis der 1. Eingabepuffer gefüllt ist. Die Adresse der zuletzt eingelesenen Bytes im 1. Eingabepuffer wird bestimmt und nach ENDINBUF abgelegt. Hat der Benutzer zwischen der Eingabe zweier Zeichen mehr als 28 sec verstreichen lassen, dann trat TIMEPUT auf. (DECSENSO = X'01'). Ist kein Zeichen im Eingabepuffer vorhanden, so wird der POST-Code gelöscht und über den OUTECB TCP1WAIT benachrichtigt, der Indicator, der TIMEOUT anzeigt (SENSØ) gelöscht, der Timeoutzähler erhöht und erneut auf Füllen des Eingabepuffers gewartet. Sind zu übersetzende Zeichen vorhanden →NACHTIMØ. Die Adresse des 2. Eingabepuffers wird gerettet (INBUFFER). Die Adresse des Ausgabefeldes (NEUFELD) und der Füllstand des Ausgabepuffers wird notiert. Ist mehr als ein Zeichen übertragen, dann wird der Eingabepuffer übersetzt. Folgt ETX auf STX so wird das Programm beendet →ENDAD. Der 1. Programmteil wird nur einmal durchlaufen, so daß das Basisregister wieder neu mit einer höheren Adresse geladen werden kann. Ein Verteiler bestimmt die Routinen für die weitere Bearbeitung. Prüfen des Vorspanns für Daten mit Formatangabe. Im Vorspann selbst können wieder sämtliche zugelassenen Sonderzeichen einschließlich der Steuerzeichen vorkommen. Die Zeichen des Vorspanns werden zeichenweise geprüft u. entsprechend ihrer Bedeutung bearbeitet.

ETX beendet die Spooltask.

Returncode in Reg.15 (402), Eingabepuffer und Fehlerpuffer zurückgeben (CLOSE, RETURN).

Die in den Ausgabepuffer übertragenen Zeichen werden bis zum vorhergehenden LINEFEED bzw. DC4. getilgt.

LINEFEED wurde erkannt→NUMMERN Zeichenroutine.

Das Zeichen wird übergangen.

Das Zeichen * wird eingefügt.

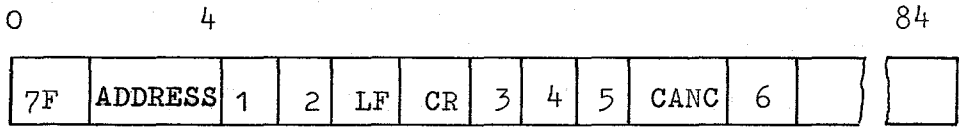
Ist der Vorspann bearbeitet und schließt mit LF/CR bzw. NL ab, so wird mit der Prüfung der Ziffernfolgen begonnen → HAUPTPR.

Nicht mit LF, CR abschließender Vorspann wird mit neuer Zeile beendet → NUMMERNZ → HAUPTPR

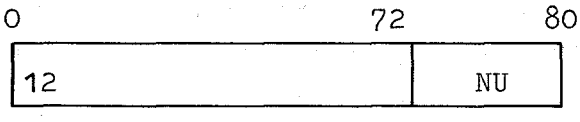
VERGL Ist der Ausgabepuffer voll bearbeitet (72 oder 80 Zeichen), so wird, auch wenn kein LF/CR folgt, automatisch dieselbe

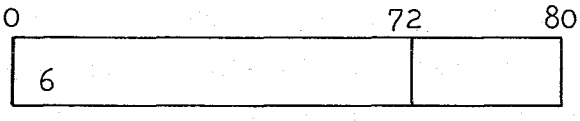
CANZELZH Funktion ausgeführt wie bei NL.Die in den letzten Ausgabepuffer übertragenen Zeichen werden ignoriert. Nach dem Zeichen CANCEL wird formatierte Eingabe beginnend mit Ziffern erwartet.

Eingabepuffer



Ausgabepuffer (Kartenformat)





Es ist möglich, daß die zu prüfenden Ziffern in verschiedenen

E Puffern stehen, deshalb wird zu Beginn geprüft, ob die zu testenden Zeichen vollständig im adressierten Puffer enthalten sind. Wenn dies der Fall ist, werden über die Routine EMM die ent-

EMM sprechenden Programmteile zur Behandlung eventuell vorkommender Sonderzeichen ausgewählt. (SUBZ, DELETEZ, ETXZH, LINEFETZ, CANCELZ, ENDEOT). Sind keine Sonderzeichen innerhalb der Zifferngruppe vorhanden, so werden die Ziffern in den Programmausgabepuffer übertragen und die Adressen auf den neuesten Stand gebracht und mit der Prüfung

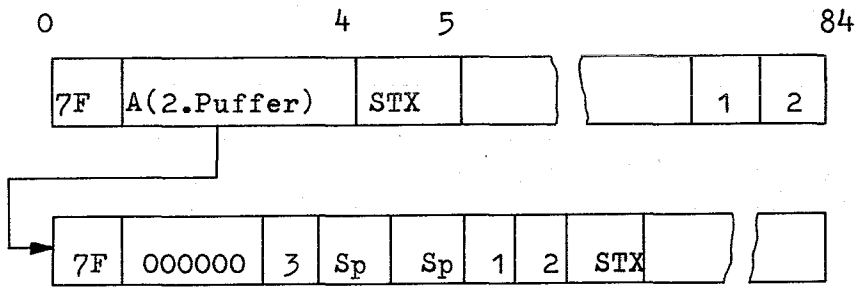
SPACEGIL das Zeichen im Zwischenraum begonnen.

Der Datenanfall in dynamisch geketteten Puffern macht die Abfrage auf Pufferende während der Prüfung auf Sonderzeichen notwendig. Ebenso muß laufend auf Ende des Ausgabepuffers geprüft werden und bei vollem Ausgabepuffer Schreiben auf den sequentiellen Spooldatensatz angestoßen werden. Das Ausschreiben und Prüfen auf fehlerfreien Ablauf des I/O-Vorgangs übernimmt die Routine NUMMERNZ. Das Bereitstellen der neuen Pufferanfangsadresse besorgt der Programmteil NEUBUF. Selbstverständlich müssen alle möglichen vorkommenden Fehlerkombinationen berücksichtigt werden (Es sollen jedoch nicht alle aufgezeigt werden.)

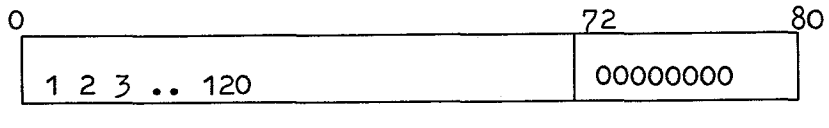
HAUPTPR Die Prüfung der Eingabedaten, die mit der Prüfung der Zifernfolge beginnt, wird ebenfalls über Tabellen innerhalb des Ziffernbereichs durchgeführt. Zugelassen sind alle Ziffern zusätzlich den Vorzeichen + und - . Im Ziffernbereich vorkommende Sonderzeichen werden wie folgt behandelt. ETXZH bedeutet Ende der Eingabe. Die fehlende Anzahl der erwarteten Ziffern (festgelegt über die Formatangabe des Benutzers) wird durch zusätzliche Nullen im Ausgabepuffer ergänzt. Der Ausgabepuffer wird als fehlerhaft gekennzeichnet →NUMMERNZ →END.

Bsp.: Formatangabe : YY = 3, ZZ = 2

Eingabepuffer



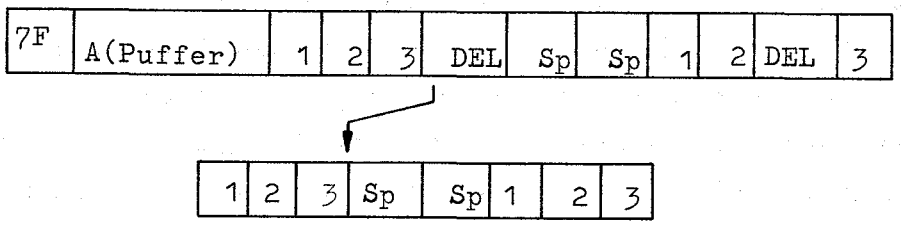
Ausgabepuffer



eingefügt

EOT — Returncode in Register 15 (402) übergeben. Eingabepuffer und eventuell angeforderte Fehlerpuffer (502) zurückgeben → TCLØSE

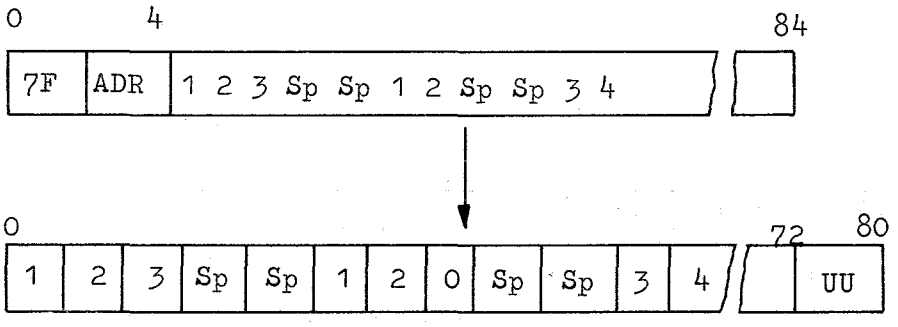
DELETEZH — DELETE als Sonderzeichen erkannt. Das Zeichen im Eingabepuffer wird ignoriert. Die Adresse im Eingabepuffer um die Anzahl der übertragenen Zeichen erhöht.



CR wird wie Delete behandelt.

SPACEZH — Die fehlenden Ziffern werden durch Nullen ergänzt

SUBZH — und der Ausgabepuffer als fehlerhaft gekennzeichnet.



Durch Einschieben von Nullen erreicht man z.B. bei statistischen Meßreihen noch gültige Datenfolgen mit gleichem Format, obwohl die Darstellung einer Messung falsch war.
Der Benutzer bekommt eine Benachrichtigung mit zusätzlicher Ausgabe der als fehlerhaft erkannten Eingabedaten am Terminal.

Bearbeitung der Daten mit der Darstellung in FØRMAT1 (ONU) oder Format 2 (NUM)

TCP - Befehl:

LIES,....., ART = NUM
oder
LIES,....., ART = ØNU

KEINF
YEND

Die Prüfung der eingelesenen Zeichen auf Sonderzeichen erfolgt über eine Prüftabelle. Der Aufbau der Tabelle wurde oben beschrieben. Wurden im geprüften Puffer keine Sonderzeichen festgestellt, so wird ein Teil der Zeichen in den Programmausgabepuffer übertragen, und zwar so viele, wie zum Füllen des Ausgabepuffers genügen, ein neuer Ausgabepuffer angefordert und die restlichen Zeichen ebenfalls in den Puffer übertragen.
Die Auswahl der detaillierten Bearbeitungsprogramme zur Behandlung der Sonderzeichen übernimmt ein Verteiler, der aufgrund des gefundenen Tabelleninhaltes der Testtabelle KEINFØRM seine Auswahl trifft.
Es sind dies die Routinen SUBK, DELETEK, LINEFETK, CANCELK, ETXK, welche die Bearbeitung der Daten bei gefundenen Sonderzeichen übernehmen.

Nummernzeichen - Routine

Die Aufgaben, welche die Routine NUMMERNZ übernimmt sind folgender Art:

Ausgabe der bearbeiteten Daten im Wechselpufferbetrieb auf den sequentiell organisierten Plattenspeicher (SPOOL-DATEN-SATZ). Anfordern von dynamisch geketteten Puffern, wenn Fehler bei Zeichen im zuletzt zu übertragenden Puffer erkannt werden.

TCP1TAB
NUMMERNZ
CFORMI

Die CSECT TCP1TAB wird aufgerufen. Dort wird auf die Tabulatorsteuerzeichen TAB geprüft und die entsprechende formatgerechte Anordnung der Zeichen im Ausgabepuffer durchgeführt. Wurde vom Benutzer der Befehl "MODIFIZIEREN" verlangt und die zu ersetzenden neuen Daten mit # und nachfolgender Zeilennumerierung eingegeben, so bringt das Unterprogramm FORMI die Numerierung in die Spalta 72-80 des Ausgabepuffers und löscht das Zeichen #. Wurde bei der Eingabe Numerierung verlangt, so wird in die Spalten 72-80 die Zeilennummer beginnend beim 1. Puffer mit 10 (mit Increment 10) als ungepackte Dezimalzahl eingefügt.

KEINNU
WRET

Beim 2. und jedem weiteren Aufruf wird das korrekte Ausschreiben des zuletzt ausgegebenen Puffers geprüft. Beim 1. Durchlauf umgeht man diese Prüfung. Der zuletzt eingelesene Puffer wird in den spezifischen Benutzer zugeordneten Zwischenspeicherbereich (Spool-Datensatz) ausgeschrieben. Enthält der Puffer das Zeichen SUB oder Zeichen die nicht definierten Codezeichen entsprechen, so wurde der Ausgabepuffer als fehlerhaft gekennzeichnet. Es erfolgt beim 1. fehlerhaften Puffer eine Anforderung von 5 dynamisch geketteten Puffern aus dem Pufferbereich, die mit dem Inhalt des als falsch erkannten Puffers gefüllt werden durch zusätzliches Einfügen von Zeilentransport- und Synchronisationszeichen. Sind alle 5 Fehlerpuffer gefüllt, so wird ein Merker gesetzt und weiterbearbeitet, d.h. die Adresse des freien Ausgabepuffers dem aufrufenden Programmteil übergeben und der Inhalt des neuen Puffers und die Kennzeichnung für fehlerhafte Puffer gelöscht.

Sind Fehlerpuffer vorhanden, d.h. RECBUFAD 0, so wird aufgrund des Returncodes in Register 15 von TCP1WAIT die Ausgabe der Puffer an den Terminalbenutzer vorgenommen.

Teilprogramm: NEUBUF

Die NEUBUF - Routine übernimmt folgende Teilaufgaben:
Rückgabe bearbeiteter Puffer an den dynamisch organisierten Pufferbereich.
Warten auf Füllen des nachfolgenden Puffers.
Übersetzen des Pufferinhaltes in den Interncode (EBCDIC)
Synchronisation mit der CSECT TCP1WAIT.

Beschreibung

NEUBUF

Der zuletzt bearbeitete Puffer wird dem dynamisch organisierten Pufferbereich zurückgegeben. Der Parameter DECONLTT des Daten-Event-Control-Blockes, der in der LINEBØX geführt wird, bestimmt die Anzahl der von TCP-Programmen benutzten dynamisch geketteten Puffer. Die Maximalzahl der Puffer im definierten Pufferbereich beträgt 255. Wird DECONLTT = X'FF', so ist ein Systemeinbruch die Folge (ab Rel. 17). Bei langen Lochstreifeneingaben ist es jedoch möglich, daß mehr als 255 Puffer benutzt werden müssen. Um ein 'Deadlock' zu vermeiden, wird der Inhalt von DECØNLTT bei jeder Pufferrückgabe, außer wenn DECØNLTT = 0 ist, um eins

SENS

erniedrigt. Trat beim Füllen des zurückgegebenen Puffers, 'TIMEØUT' auf, dann wird ein Puffer aus dem dynamisch organisierten Pufferbereich angefordert und die CSECT TCP1WAIT verständigt (POST ØUTECB) und ihr die Pufferadresse über die LINEBØX über-

NACHPØST

geben. Der Indikator für TIMEØUT wird gelöscht → NACHPOST. Wurde beim Füllen des zuletzt bearbeiteten Puffers kein TIMEØUT festgestellt, so wird auf Füllen des nächsten Puffers gewartet und der Indikator für die Kennzeichnung des letzten Bytes im Eingabepuffer gelöscht. Zeigt der Timeoutzähler 5 aufeinanderfolgende TIMEØUT an, so wird die Spooltask beendet. Returncode = 402 in Register 15→RECØDE.

Ist während der Eingabe der Daten TIMEØUT aufgetreten (DECSENSO = X'01'), so wird über DECCOUNT die Anzahl der in den Puffer übertragenen Zeichen bestimmt. Für die entsprechenden Bearbeitungsroutinen kennzeichnet diese Adresse das letzte relevante Byte innerhalb der Pufferkette. Enthält der neue Puffer eingelesene Daten? Wenn dies der Fall ist, wird mit der Bearbeitung der Daten bei SENSØSE begonnen. Wurden keine Daten in den Puffer übertragen, so wird der Timeoutzähler erhöht und der Code 7F im Eventkontrollblock des Puffers gesetzt, die Timeoutkennzeichnung gelöscht und TCP1WAIT über den OUTECB

SENSØSE benachrichtigt → NACHPOST. Der Timeoutindikator wird gesetzt

STUBEND und die Endadresse des zuletzt eingelesenen Byte notiert. Handelt es sich bei den eingelesenen Zeichen um CCITT Nr. 2 - Code, so werden die Codezeichen zeichenweise übersetzt.

TTSUEBER War das geprüfte Zeichen Buchstabenumschaltung, so wird die Übersetzungstabellenanfangsadresse geladen, für Ziffernumschaltung die Tabellenadresse +3,

DELEINF anstelle der Umschaltzeichen DELETE eingefügt und das nach-

FØLGBYT folgende Zeichen übersetzt. Sind alle Zeichen bearbeitet, werden die abgespeicherten Registerinhalte wieder geladen und zum auf-

NACHTRAN rufenden Programmteil zurückgekehrt. Sind die eingelesenen Zeichen keine CCITT Nr. 2 Codezeichen, so wird pufferweise über die entsprechend ausgewählte Tabelle in den Interncode EBCDIC übersetzt → NACHTRAN.

4. TCP1TSDN (W. Strolz)

CSECT TCP1TSDN

ENTRY via LINK von TCP1BASE
(CSECT PCP1WAIT)

EINGANGSPARAMETER

KONBØX via R6

EXIT RETURN nach TCP1BASE
(CSECT TCP1WAIT)

FUNKTION

TCP1TSDN beendet bei SHUTDOWN anstehende PREPARE-Commands durch HALT-I/O. Nach dem Ausdrucken des Line Error Blocks (Line Error Count) am Operator-Konsol erfolgt CLOSE auf alle SPOOL-Data Sets. Anschließend werden sämtliche geladenen BTAM-Moduln mit DELETE abgeräumt; der Subpool 1 wird vollständig freigegeben. RETURN nach TCP1BASE (TCP1WAIT).

PROGRAMMABLAUF

TCP1TSDN
HALTROUT
HALT1
HALT2
CLOSE
DEL1

Der DCB für dynamische Ein-/Ausgabe (DDCB) und die entsprechende DDEB weeden geholt. Die I/O-Blöcke werden zur Ausgabe des HALT-I/O-Commands initialisiert und der IOB und der UCB der 1. Line werden beschafft. Hat die Line den Status "BUSY" wird die Channel Program Address geholt und geprüft, ob das 1. CCW (Channel Command Word) ein PREPARE-Command darstellt. Ist das der Fall, dann werden im DCB und IOB bestimmte I/O-Flags gesetzt und schließlich mittels Macro IOHALT das Kommando HALT-I/O gegeben. War dagegen die Line nicht "BUSY" oder lag kein PREPARE vor, dann erfolgt statt dessen ein Sprung nach HALT 2. Nachdem HALT-I/O gegeben wurde, wird auf diese Weise bei HALT 1 die nächste Line behandelt. Sind schließlich alle Lines abgearbeitet, wird die im Line Error Block (LERB) (siehe BTAM System Reference Library) enthaltene Information an der Operator-Schreibmaschine ausgedruckt. Sämtliche DCB's für die Spool Data Sets (SEQDCB's) werden mit CLOSE abgeschlossen, die während der Initialisierungsphase geladenen BTAM-Moduln

IGG019MB	Ch. End-Appendage
IGG019MC	PCI-Appendage
IGG019MS	Buffer-Routine
IGG019MA	R/W-Routine
IGG/19MO	I/O-Device-Module

FREECORE mit DELETE abgeräumt und der ganze Subpool 1 freigegeben. Anschließend RETURN zu TCP1BASE, CSECT TCP1WAIT.

V. Kommandoentschlüsselung

Die Kommandoentschlüsselung und die Organisation der Kommandobearbeitung wird vom Loadmodule TCPOPROC durchgeführt.

1. PROC (v. Haase)

INIT Nach der Registerrettung für jedes angeschlossene Terminal (ACTENO) ein Puffer requiriert. Die Initialisierungsnachricht (*) wird auf das Druckwerk des Terminals ausgegeben (SELINDO = D; POST OUTECB mit '42'). Das geschieht zyklisch für alle Terminals, die nicht 'LINEDOWN' sind (alle Lineboxen sind verkettet!).

DKPST Dann erfolgt der Start des Displaymoduls (DCP1DSUP) via ATTACH, nachdem alle Kommunikationsfelder (DISPECB, DISPECB2, PAKT) ge-

LOOP1 löscht wurden. In der Hauptschleife wird der aktuelle Anfang der (zyklischen) Linebox-Liste geladen und auf ein Ereignis (=POST) von einem Terminal (Eingabe-End) oder vom Displayprogramm (Bildausgabe-Ende) gewartet. Ist das Ereignis eingetreten, wird die Linebox-Liste (ELISTIN) rotiert, auf den Konbox-Status =

W1 SHUT geprüft (wenn ja →END1) und festgestellt, woher die Mel-

NORMAL dung kam. Kommt sie von einem Terminal, wird auf 'HOLD-Modus', temporärer Fehler (a) und "Fehlerbeseitigung" (b) geprüft. Die betreffenden Statusbits werden in der LINEBOX gesetzt und Benutzerabschaltung (a) bzw. Nachinitialisierung (b) durchgeführt.

W13 Bei dieser Marke wird die Art der Eingabe geprüft und danach verzweigt. Keine Eingabe (= Ende einer Ausgabe)→ENDSP; dynamische Eingabe→DATA1; nach Kommandoeingabe wird der Entschlüßler (STEFAN) mit CALL aufgerufen. Nach dem Durchlauf des Entschlüßlers wird der Kommandopuffer zurückgegeben und auf Richtigkeit des Kommandos geprüft.

Die Weiterverarbeitung erfolgt im Fehlerfall beim Label ESMESS, sonst wird bei PROCESS fortgesetzt. Über eine Sprungliste wird je nach Befehlscode (BCODE) der Name der Verarbeitungsroutine in ein ATTACH-Statement eingesetzt (in einigen Fällen auch POST,

ATT LINK oder CALL). Nach der Kreierung der Funktionstask wird an den

DATA1 Anfang der Hauptschleife zurückgesprungen (LOOP1). Der gleiche Programmzweig wird vom Label DATA1 an durchlaufen; nachdem BCODE und WCODE2 untersucht wurden, wird über ATT eine der 3 Routinen

GOON TCP1AEN2, TCPMODR oder TCP1LIE2 aktiviert. Der Fluß nach Ende der gelinkten Accounting-Routine (TCP1ACNT), nach einer Ent-

ESMESS schlüßler-Fehlermeldung und nach Beendigung einer dynamischen

ENDSP Ausgabe führt zum Löschen nicht mehr gebrauchter LINEBOX-Felder und zu einem Aufruf der MESSAGE-Routine. Nachdem diese aus den Fehlercodes (LCC12) in der Linebox Meldungen zusammengestellt hat, wird TCP1BASE von der Beendigung der Bearbeitung der letzten

POSTEND Eingabe verständigt. Dann erfolgt Rücksprung zum Anfang der Hauptschleife LOOP1 (bei Displayfehler zum Display-Restart DKPST).
Wird bei W1 festgestellt, daß eine Displayrückmeldung vorliegt, so wird nach der Fehlerprüfung (OK, temporärer oder Dauerfehler) auf das Vorliegen weiterer Displayaufträge abgefragt; wenn ja wird der DISPECB mit dem nächsten Eintrag (LBX-Adresse) im Display-Kontrollblock (DISPCB) gepostet. Wenn nein, werden DISPCB und die Zeiger auf den aktuellen und den nächsten Displaywunsch (PAKT, PNEXT) zurückgesetzt. In beiden Fällen erfolgt ein Sprung zur Schlußbehandlung (GOON).
Wird bei PROCESS eine Displayanforderung festgestellt, wird diese (LBX-Adresse) in DISPCB eingetragen. Wenn kein anderer Displayauftrag zur Zeit aktiv ist, wird DISPECB gepostet (Code = LBX-Adresse).

2. STEFAN (U. Ullrich)

STEFAN führt die eigentliche Kommandoentschlüsselung durch. Die Adresse des Kommandotextes entnimmt das Programm dem Feld BUFADS in der Linebox (LBX-Adresse in R3).

ANF — Geprüft wird, ob die Zeile gecancelt wurde; tritt einmal das 'CANCEL'-Zeichen auf, wird der Text-ANFANG dahinter verlegt, 2 x CANCEL führt zum Ausgang EX1 (Kommando wiederholen).

BITTEST — Nach der Prüfung, ob Befehls- oder Parametereingabe vorliegt (TFLG = X'02' = Befehl), wird das Befehls-Schlüsselwort und das Zeichen nach dem Befehl geprüft. Auftreten von ETX wird

BEFEHL — in PFLA (= X'80') notiert. Die Felder in der Linebox für Namen variabler Länge (PARNAM, bei Displaybefehlen DDSMN und DSEITE) werden gelöscht, auf Hold-Modus (BFLG = X'40') und der Logon-

NORCOM — Zustand (TFLG = X'80') geprüft. Aus der BLIST werden die Anzahl

PARSPR — der Parameter und der Anzeiger, ob Dateneingabe folgt, übernommen. Nach der Prüfung, ob die Parametereingabe mit oder ohne Schlüsselwort erfolgt (z.B. GERAET = LOCHSTR,...bzw. LOCHSTR,...)

PAR — werden die Parameter untersucht. Bei 'Keyword'-Parametern werden

WERSUCH — die ersten 3 Zeichen des Parameternamens mit der PARLIST verglichen; dann der Wert des Parameters mit der WLIST; bei freiem

KOPROT — Namen (z.B. Member-Name) wird der Name entweder nach LAMEMBN

bzw. KOP — (Label: UEBERT) oder in einen Bereich von PARNAM (Pointer AD) übertragen (Label EIN1). Gefundene Werte-Keywords führen zum

KOMMA — Eintrag des WCODE in die LBX; die Namen-Pointer kommen nach WADR.

WEITER2 — Hier werden 'Positional' Parameter untersucht. Die Schlußbehandlung einer Parameterprüfung setzt den Parameterzeiger (PARZG)

PRUEF2 — auf den nächsten Platz (modulo 5). Ist ETX erreicht, wird die Vollständigkeit des Kommandos geprüft. Die Ausgänge des Programms sind NORMEND (alles O.K.), EX1 (Kommando wiederholen), EX2 (Parameter nachfordern), EX3 (Logon vergessen), EX4 (Logon verboten), EX5 (Dateneingabe verboten), EX6 (RJE verboten) und EX7 (kein Display angeschlossen). In den Ausgängen werden die End-Codes (LCC12) gesetzt.

EINC Das Logon-Kommando wird teilweise gesondert behandelt, da die Benutzer-Identifikation in die LBX (USERID) an eine besondere Stelle eingetragen werden muß.

Spezielle Listenformate

a) Befehlsliste BLIST (1 Eintrag = 12 Bytes)

0: Endeindikator (letzter Eintrag = X'80', sonst X'40')
1: Command-Keyword (3 Bytes)
4: Anzahl Parameter (1 Byte)
5: Adresse Parameterliste (3 Bytes) oder X'404040'
8: Command Code (BCODE) (1 Byte)
9: Dateneingabe-Indikator (1 Byte) = X'80' (Daten), X'40' (keine D.)
10: frei (2 Bytes)

b) Parameterliste PARLIST (1 Eintrag = 12 Bytes)

0: Endeindikator (1 Byte)
1: Parameter-Keyword (3 Bytes)
4: Anzahl Werte (1 Byte)
5: Adresse Werteliste (3 Bytes)
8: Parameter-Nummer (1 Byte)
9: Länge eines freien Wertenamens (=X'40' wenn nicht erlaubt)
10: Indikator, ob Name nach LAMEMBN (X'80'), sonst X'40'
11: frei (1 Byte)

c) Werteliste WLIST (1 Eintrag = 8 Bytes)

0: Endeindikator (1 Byte)
1: Werte-Keyword (3 Bytes)
4: Werte-Nummer (WCODE) (1 Byte)
5: X'40' oder die Zahl, auf die die Anzahl der Parameter dieses Kommandos geändert werden soll, wenn dieser Wert auftritt.
6: frei (2 Bytes)

3. MESSAGE (V. Haase)

'Message' ist reenterable und dient dazu, für alle von TCP festgestellten Antwortcodes (LCC12 in LBX) Antworttexte aus einer Liste herauszusuchen und in einen dynamischen Puffer (Adresse schon vorher in BUFADS) zu übertragen.
Zuerst wird festgestellt, ob der Fall: 'Accounting-Routine (TCP1ACNT) korrekt zu Ende' vorliegt (LCC12 = X'0600'); in diesem Fall wird MESSAGE übersprungen, da hier die Nachrichtenassemblierung schon von TCP1ACNT durchgeführt wurde.

SUCHEN — Andernfalls wird die Message-Liste (MLIST) nach einem passenden Code durchsucht.
Format der MLIST:
H'Code' H'Länge' CL Länge 'Text'
.....
X'FFFF' (Endezeichen)

MOVEZ — Ist der betreffende Code gefunden (bzw. das Ende der Liste erreicht), wird der zugehörige Text (bzw. eine Fehlermeldung) in den Puffer, dessen Adresse in BUFADS steht, übertragen. Der Pufferinhalt wird in USASCII-Code übersetzt, die Länge des

WEITER — Textes in das 1. Byte von BUFADS gespeichert. Dann erfolgt FREEMAIN auf Arbeitsbereiche und RETURN zum aufrufenden Programm.

4. **ATER** (V. Haase)

Diese CSECT wird angesprungen, wenn eine der von PROC attachten Funktionsroutinen zu Ende geht. Der Ablauf von PROC wird für die Dauer von ATER unterbrochen. Über die CVT wird der TCP von PROC gefunden, in seiner Save-Area sind die KONBOX- und LINEBOX-Adressen zu finden.
Nachdem die Funktionstask detached wurde, wird ein Puffer aquiriert und über die CSECT MESSAGE der dem Completion Code LCC12 (in der LBX) entsprechenden Text eingetragen.
Anschließend wird der Status der LBX geprüft; bei fehlerlosem

FL1 — Ablauf werden alle kommandorelevanten Felder der LBX (BCODE, P1-P5) gelöscht; ebenso bei irreparablen Fehlern (neue Kommandoeingabe). Muß nur ein Parameter nachverlangt werden (LCC12 = X'08XX)

SFREGEN — so werden Parameterpointer und Prompting-Code entsprechend gesetzt.

Auch etwa anstehende Folgeaufträge (z.B. Dateneingabe) werden im Status vermerkt. Alle Indikatoren für die Terminalausgabe werden gesetzt (TFLG, SELINDO, SELINDI). In jedem Fall wird am

ENDCL Ende der OUTECB des betreffenden Terminals gepostet, so daß über TCP1WAIT die Antwort auf das Terminal geschrieben werden kann. RETURN der asynchronen Routine geht zum OS.

VI. Bibliotheksverwaltung

1. TCP1ERK (M. Schayegh)

FUNKTION

Der Loadmodule TCP1ERK führt den Befehl 'ERKLAERE' aus. Im Loadmodule TCP1ERK wird dabei der Name des gewünschten Erkläretextes und der DCB des Erkläre-Data Set's aufgebaut. Mit diesem Namen und DCB in der LINEBOX, wird das Programm 'TCP1SCH' aufgerufen und so der gewünschte Text ausgegeben.

EINGANGSPARAMETER

Adresse der LINEBOX im Reg. 3: LBXREG
Adresse der KONBOX im Reg.11, wird nach KBXREG (Reg. 12) umgeladen.
Aus der LINEBOX wird hauptsächlich verwendet:

WCODE1 Nummer des Erkläretextes. Die Nummer des Textes ist der Befehlscode des zu erklärenden Befehls mit der Ausnahme von "TCP" und "BEFEHLE", mit denen das TCP-System und die Befehle erklärt werden.

PARTDCB DCB des Bibliotheks-Data Set's. Er wird während des Programmes durch einen DCB für den Erkläre-Data Set: TCP. ERKLIB ersetzt und am Ende wieder zurückgespeichert. Aus der KONBOX wird das Feld KBXLIB benötigt. Es enthält die Adresse des DCB's für die TCP. LINKLIB.

AUSGANGSPARAMETER

LCC12 X'0000' oder ein Fehlercode

Befehlscode - Erkläretextnummer

	X' '	D ' '	
EIN	01	1	ohne Erkläretexte
AUS	02	2	
ERK	03	3	
ZUS	04	4	
UMS	05	5	
NAC	06	6	
LIES	07	7	
SCH	08	8	
DIS/DTX/DZU	09	9	
MOD	0A	10	
LOE	0B	11	
KOP	0C	12	
STA	0D	13	
	0E	14	
ZLS	0F	15	
SYS	10	16	
PL1	11	17	
HOL	12	18	
BLI	13	19	
KET	14	20	
TCP	62	98	Keine Befehle
BEF	63	99	

PROGRAMMABLAUF

Der Loadmodule TCP1ERK, CSECT: ERKLAERE wird vom Loadmodule TCPOPROC aufgerufen. Die CSECT: ERKLAERE wird mit dem Macro ALEPH eingeleitet. Anschließend wird die Adresse der KONBOX in das KBXREG umgeladen.
In diesem Loadmodule wird ein DCB aufgebaut für den Data Set, auf dem die Erkläretexte stehen. Dieser DCB wird an die Stelle des PARTDCB's in die LINEBOX geschrieben. Der Standard PARTDCB wird vorher in die WORKAREA gespeichert.
Danach wird eine Listform für ein OPEN-Macro aufgebaut. Es folgt das OPEN auf den neuen PARTDCB. Somit ist der Erkläre-Data-Set eröffnet. In das Feld WCODE1 wurde vom Befehls-Entschlüssler für den zu erklärenden Befehl der Befehlscode geschrieben. Dieser Code wird nun in eine ungepackte Dezimalzahl umgewandelt und damit der Name des entsprechenden Erkläre-Members gebaut. Die Form des Namens ist:
ERTEXTXX, XX = Befehlscode als ungepackte Dez.-Zahl. Der Name des Erkläre-Members wird nun nach LAMEMBN in die LINEBOX gespeichert.
Nach WCODE 1 wird nun ein Code für 'Schreibe' (X'02') geschrieben. Nach dieser Verarbeitung wird mit dem Macro LINK der Loadmodule TCP1SCH aufgerufen. Er schreibt das gewünschte Member (Erkläretext) von der TCP.ERKLIB auf den SPOOLDS, von wo er an das Terminal ausgegeben wird.
Ist TCP1SCH zu Ende, so wird der DCB für den Erkläre-Data-Set mit CLOSE geschlossen und durch den alten PARTDCB ersetzt. Mit dem Macro HE wird nun das Programm TCP1ERK abgeschlossen.
(→ TCPØPROC (ATER))

2. TCP1ZUS (M. Schayegh)

Dieser Module wird mit dem Befehl
Z durch TCPØPROC mit ATTACH aktiviert

FUNKTION

Das Inhaltsverzeichnis der Bibliothek wird ausgedruckt

EINGANGSPARAMETER

R3 :A (LINEBOX)
USERID : Useridentifikation
VOLSER : Volume-serial-number f.Dataset

AUSGANGSPARAMETER

LCC12 : Fehlercode

PROGRAMMABLAUF

Nach Macro ALEPH wird die Adresse der SYNAD-Routine in SEQDCB geladen, Data Control Block für READ umgespeichert und der DDNAME = DD - VOLSER in DCB eingesetzt.
Das "fullword" OPCLLIST enthält : X'80', AL3 (ARTDCB)
Das "fullword" LISTEXIT enthält : X'87', AL3 (JFCBAREA)
Byte 37-40 von ARTDCB enthält : AL3 (LISTEXIT)

Das Macro RDJFCB liest den "Job File Control Block" für den spezifizierten DCB in das 176 Byte lange Feld JFCBAREA. In diesen Speicherbereich werden einige Felder wie DSNAME, DSORG, RECFM usw. überschrieben. Ein OPEN TYPE = J öffnet den Dataset und anschließend werden die Listformen für DECBS umgespeichert. Macro POINT stellt die Anfangsadresse vom sequentiellen Dataset für das nächste WRITE zur Verfügung.

WRITE Hier wird durch eine Schleife, die zweimal den Befehl WRITE durchläuft, die Überschrift

Inhalt der Bibliothek

MEMBERNAME TT R BLKZ ERZEUGT GEAEN. ------

ausgeschrieben.

Nach WRITE werden die beiden Speicherbereiche DIRFELD und

LOOPDB PRINTAR geräumt. Macro READ liest einen "Directory Block" - es

LOOPMR sind drei Directory-Einträge - in DIRFELD ein. Die Anzahl von "user data half words" wird in RI geladen, mit 2 multipliziert und 12 addiert. Damit enthält RI die Länge eines directory-Eintrages. Dann wird jeweils ein Eintrag aus DIRFELD in MEMBFELD übertragen.

Wenn das letzte Member im Directory erreicht ist, so wird das Programm beendet.

Aus MEMBFELD wird Name und TTR ins Ausgabefeld PRINTAR in ausdruckbarer Form übertragen. Dann wird die Länge des Directory-Eintrages mit 74 verglichen. Sind sie ungleich, so handelt es sich um einen Member, der durch ein UTILITY-Programm eingespielt wird. Für solche Member gibt es keine zusätzlichen directory-Einträge außer NAME und TTR. daher wird zu FBLANK gesprungen, und der Rest des Feldes durch Blank ersetzt. Handelt es sich dagegen um einen von TCP eingeschriebenen Member, so sind weitere Informationen wie Datum, Blockzahl, Code, Art,... vorhanden. Diese Informationen sind in codierter nicht ausdruckbarer Form. Sie werden nach der Umwandlung ins Ausgabefeld PRINTAR umge-

OUTWRITE speichert. WRITE schreibt den aufbereiteten Eintrag aus. Nach dem WRITE wird abgefragt, ob alle Records in einem directory-Block verarbeitet wurden. Wenn nein, dann wird nach LOOPMR gesprungen, dort ein neuer Record geholt und verarbeitet.

Ist ein Block abgearbeitet, so wird nach LOOPDB gesprungen und

ENDE ein neuer Block eingelesen. Hier beenden die Macros CLOSE und HE das Programm (⇒ ATER).

3. TCP1LIE2 (M. Schayegh)

Dieser Module wird nach dem Befehl Lies, Gerät, Name, Art, Format, Code durch TCPØPROC mit ATTACH aktiviert. Zuvor müssen die Routinen TCP1INP1 und TCP1SPOL durchlaufen und das DATA-Bit in TFLG gesetzt worden sein.

FUNKTION

Die eingegebenen Daten werden in der Bibliothek unter den gegebenen Namen abgespeichert und die spezifizierten Attribute ins Inhaltsverzeichnis eingetragen.

EINGANGSPARAMETER

1)	LAMEMBN	:	Membername
2)	WCODE3	:	ART
	1	=	JCL
	2	=	DATA
	3	=	PL1
	4	=	FORTRAN
	5	=	ASSEMBLER
	6	=	ALGOL
3)	WCODE5	:	CODE
	1	=	ISO7
	2	=	BCD
	3	=	BIN
	4	=	KA5
4)	WADR4	:	A (FORMATFELD)
5)	WCODE4	:	FORMAT
	0	=	Formatiert
	1	=	NUM
	2	=	ONU

AUSGANGSPARAMETER

LCC12	:	Fehlercode

PROGRAMMABLAUF

Nach Macro ALEPH werden die Listformen für DECB umgespeichert und SYNAD und EODAD-Adressen in den DCB gespeichert. Macro POINT stellt die Adresse des Eingabeblocks für das nächste Read zur Verfügung.

WEITER Hier erfolgt die Initialisierung und Vorbereitung von Registern.

READB Durch READ wird jeweils ein Record (80 Byte) aus dem sequentiellen Eingabestrom in das Feld AREAN übertragen, dann wird die Adresse um 80 erhöht und zu READB zurückgesprungen bis 20

INJUMP Records zu einem Block verarbeitet sind. Der gelesene Block wird dann in die Bibliothek übertragen. Falls es sich um den ersten WRITE-Befehl handelt, wird die Adresse des ersten Blocks

COUNT auf der Bibliothek, mit Macro NOTE festgehalten. Der Blockzähler wird dann um eins hochgezählt und abgefragt, ob "End of data" registriert worden ist, d.h. ob dieser Block der letzte Block ist. Wenn nein, dann wird zu READA gesprungen, um einen neuen Block zu lesen. Sonst erfolgt ein Sprung zu OVERJUMP.

LOOP1 Es ist die "End of data" - Routine. Hier wird die Länge des letzten Blocks berechnet und ein Code gesetzt und anschließend INJUMP gesprungen, um den letzten Block in die Bibliothek zu

OVERJUMP schreiben. Hier wird durch STOW im Inhaltsverzeichnis ein Ein-

DIRECONN trag für das eingefügte Member gemacht. Falls STOW nicht richtig ausgeführt wird, so werden Fehler-Codes gesetzt und das Programm beendet; bei Ein-Ausgabefehler für STOW werden einige Felder des

ENDE betreffenden DCB neu initialisiert. Ein CLOSE Type = T schließt den sequentiellen Data-Set; wenn für Formatangabe ein Speicherbereich durch GETMAIN reserviert worden ist, dann wird er hier

SCHLUSS durch FREEMAIN freigegeben. Macro HE beendet das Programm.

(→ TCPOPROC (ATER))

4. TCP1AEN2 (M. Schayegh)

Dieser Module wird nach dem Befehl MOD, NAME, Z zum TCPØPROC durch ATTACH aufgerufen. Zuvor müssen TCP1INP1 und TCP1SPOL durchlaufen und das Data-Bit in TFLG gesetzt worden sein.

FUNKTION

In dem spezifizierten Member können beliebige Zeilen hinzugefügt oder vertauscht werden.
Die Eingabezeilen werden mit einer Nummer versehen. Es muß dabei beachtet werden, daß die Nummer für die Eingabezeilen in aufsteigender Reihenfolge sein müssen.

EINGANGSPARAMETER

LAMEMBN : Membername
SPDSBLAD : Adresse des sequentiellen Datasets

AUSGANGSPARAMETER

LCC12 : Fehlercode

Schematische Darstellung

Folgende drei Puffer sind als Zwischenspeicher vorgesehen.
OLDBUFF = 1600 Byte als Eingabepuffer.
NEWBUFF = 1600 Byte als Ausgabepuffer.
NEWCOMP = 80 Byte als Eingabepuffer für sequentiellen Dataset.
Es wird aus PDS ein Block in OLDBUFF gelesen und aus SDS ein Record in NEWCOMP. Dann iwrd jeweils ein Record aus OLDBUFF herausgenommen, dessen Nummer mit der Nummer von Records in NEWBUFF verglichen und aufgrund dieses Vergleiches wird entweder von NEWCOMP oder OLDBUFF ein Record in NEWBUFF übertragen, bis NEWBUFF voll gespeichertist. Dann wird von NEWBUFF in PDS übertragen. Das große Programm ist aus einzelnen Routinen gebaut, die entweder mit einem unbedingten Sprung oder mit einem BR 9 enden. Bevor von jeder Routine zur anderen Routine gesprungen wird, wird Register 9 mit Rücksprungadresse geladen.

PROGRAMMABLAUF

Nach Macro ALEPH wird Timer gesetzt, dann werden einige Felder initialisiert, die Listform für DECB umgespeichert und "End of data" und Synad - Adresse in DCB geladen. POINT Macro Instruktion stellt die Adresse des sequentiellen Datensatzes bereit.

BLDL Macro Instruktion liest aus dem Inhaltsverzeichnis den Eintrag für das betreffende Member in BLDLLIST.

FIND — FIND Macro Instruktion stellt die Anfangsadresse des Members für das nächste READ zur Verfügung.

Im Register 9, das im Programm als Sprung-Register verwendet

RPDS — wird, wird die Adresse von RSDS1 geladen. Es wird ein Block von PDS in OLDBUFF gelesen, die Länge des gelesenen Blockes ermittelt und die Endadresse des Blocks in OLDEND gespeichert; R4 enthält A (OLDBUFF).

Dann wird abgefragt, ob die Nummer des Records im gelesenen Block mit führenden Blanks versehen ist. Ist es der Fall, so wird nach SET gesprungen, dort die führenden Blanks durch Null ersetzt und nach G04 zurückgesprungen.

G04 — BR R9 bewirkt einen Sprung nach:

RSDS1 oder M11 oder M21

Der erste Sprung ist nach RSDS1

RSDS1 — In R9 wird A (VERGL) geladen

RSDS — Es wird ein Record aus SDS in NEWCOMP gelesen, die Nummer dieses Records mit der Nummer des vorherigen Records aus SDS, die in LETZNUM gespeichert ist, verglichen. Falls sie größer oder gleich der vorherigen Nummer ist, so wird nach FEHLER1 gesprungen und dort der Fehlercode um eins hochgezählt. Dies bedeutet dann, daß bei den Eingabedaten:

1.) Die angegebenen Nummern in aufsteigender Reihenfolge sein müssen.

2.) Sich eine einmal angegebene Nummer nicht wiederholen darf.

Ist die Nummer kleiner als die vorherige Nummer, so wird sie in LETZNUM umgespeichert und dient beim nächsten Mal als vorherige Nummer.

Der Anfangswert für LETZNUM ist Null.
BR R9 bewirkt einen Sprung nach:

VERGL oder M4

Der erste Sprung ist nach VERGL.

VERGL Es wird abgefragt, ob End of data beim SDS registriert worden ist. Wenn ja, dann muß ausschließlich von OLDBUFF in NEWBUFF übertragen werden. Es wird dann nach M2 gesprungen. Sonst wird nach "End of data" beim PDS abgefragt. Ist es der Fall, so wird ein Sprung nach M4 gemacht und dort von NEWCOMP in NEWBUFF übertragen.
Ist weder bei PDS noch bei SDS "End of data" erreicht, so wird die Nummer in OLDBUFF mit der Nummer in NEWCOMP verglichen und je nachdem, ob die erste kleiner, gleich oder größer ist wird nach M2, M1 oder M3 gesprungen und dort entweder von NEWCOMP oder von OLDBUFF in NEWBUFF übertragen.

M1 bis M4 sind Routinen, bei denen eine Übertragung nach NEWBUFF erfolgt. Register 4 zeigt auf den nächst zu übertragenden Record in OLDBUFF. Jedesmal wenn ein Record von OLDBUFF- oder wenn es sich um Ersetzen eines Records handelt, von NEWCOMP - in NEWBUFF übertragen wird, wird R4 um 80 erhöht und abgefragt, ob Ende des OLDBUFF erreicht ist. Wenn ja, dann wird ein neuer Block in OLDBUFF gelesen (RPDS) und durch BR 9 nach betreffender Stelle zurückgesprungen. Sonst wird nach jeder Übertragung Register 5, das auf den gerade übertragenen Record zeigt, um 80 erhöht und wiederum abgefragt, ob das Ende von NEWBUFF erreicht ist. Wenn ja, dann wird nach WPDS gesprungen, dort der Block in PDS geschrieben, R5 auf Anfangswert, d.h. A(NEWBUFF) gesetzt und wieder zu der Stelle zurückgesprungen, von der nach WPDS gesprungen worden war. Bei den Routinen, bei denen eine Übertragung von NEWCOMP nach NEWBUFF stattfindet, wird ein Sprung nach RSDS gemacht und dort ein neuer Record eingelesen. Die Übertragungsroutinen werden solange durchlaufen, bis bei PDS bzw. bei SDS "End of data" registriert wird.

In diesem Fall wird dann nach PDSEODAD bzw. SDSEODAD gesprungen, dort ein Code gesetzt und abgefragt, ob auch "End of data" bei SDS bzw. PDS registriert worden ist. Wenn ja, dann wird nach FERTIG gesprungen, sonst durch einen Verzweigungsbefehl nach

FERTIG einer geeigneten Routine gesprungen. Hier wird die Länge des letzten Blockes in NEWBUFF bestimmt und wenn sie ungleich Null ist, dann erfolgt ein Sprung nach WPDS1; dort der letzte Block in

STOW PDS geschrieben und nach STOW zurückgesprungen. Hier wird ein Eintrag für das betreffende Member im Verzeichnis eingefügt.

END CLOSE TYPE = T und Macro HE beenden das Programm (→ATER)

4.1 TCP1INP1

Dieser Modul wird unmittelbar nach den Eingabekommandos: "LIES" und "AENDERE" von TCPØPROC attached. Die Gültigkeit der Parameter wird geprüft und das DATA-Bit in TFLG gesetzt (zur Information, daß TCP1SPOL und TCPLIE2 bzw. TCP1AEN2 gestartet werden müssen).
Rückkehr zu TCPØPROC (ATER).

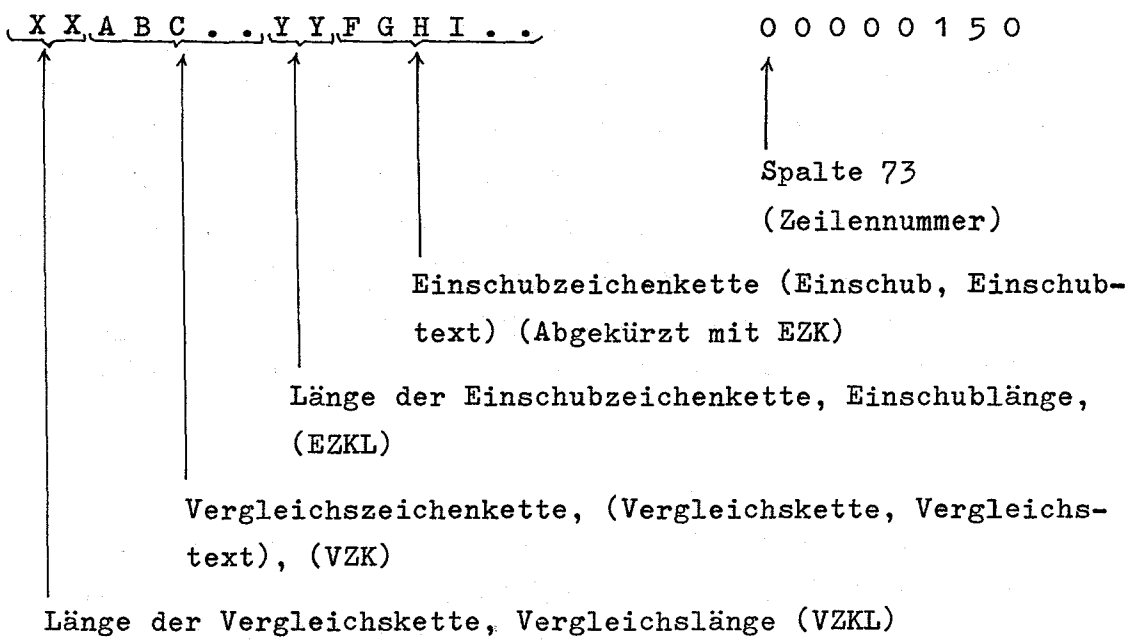
5. TCP1MODR (S. Müller)

Dieser Module wird nach dem Befehl MOD, NAME, K durch TCPØPROC mit ATTACH aktiviert.

FUNKTION

Mit diesem Programm können in einzelnen Zeilen eines Members Zeichengruppen geändert werden. Die Änderungseingabe steht auf dem SPOOLDS in Zeilenform mit einer Zeilennummer. Die Zeilennummer identifiziert die zu ändernde Zeile des Members. Das Programm liest das Member blockweise ein, ändert die entsprechenden Zeilen und schreibt das Member wieder auf die Benutzerbibliothek zurück.

Form der Änderungseingabe:



XX und YY müssen immer zweistellig geschrieben werden. Ist die Länge nur eine einstellige Zahl, so muß die Zehnerzelle mit 0 aufgefüllt werden z.B. 12/05/

Bedingung: XX > 0
YY ≥ 0

EINGANGSPARAMETER

Adresse der LINEBOX in Register 3
Benötigte Felder der LINEBOX und ihre Anfangswerte.

LAMEMBN	Name des zu ändernden Members.	
PARTDCB }	DCBs für die Benutzerbibliothek	Input
PARTDCO }		Output
SEQDCB	DCB für den SPOOL-DS	
SPDSBLAD	Anfangsadresse des SPOOL DS auf der Platte	

AUSGANGSPARAMETER

LCC12 = Fehlercode (= 0, wenn fehlerfrei)

PROGRAMMABLAUF

Das Programm wird vom Loadmodule TCPØPROC aufgerufen und springt nach TCPØPROC CSECT: ATER zurück, wenn die gewünschte Änderung durchgeführt wurde oder dabei ein Fehler aufgetreten ist. Das Programm beginnt mit dem Macro ALEPH, das die Anfangsinitialisierung durchführt und mit GETMAIN den für die WORKAREA benötigten Speicherplatz zuordnet.
Nun wird LCC12 auf einen bestimmten Anfangswert gesetzt (X'FFFF' = Programmfehler). Anschließend werden in die DCBs folgende Adressen eingetragen.

PARTDCB:	A (PARTINER)	Input Error
	A (ENDWORK)	Eodad = Datenende
SEQDCB	A (SPOCLINE)	Input Error
	A (SPOOLEND)	Datenende.

Es folgt der Aufbau der Liste für BLDL. Nun wird geprüft, ob das Member, das mit dem Programm geändert werden soll, in der Benutzerbibliothek vorhanden ist. Sollte dies nicht der Fall sein, oder ist ein Perm. I/O-Error aufgetreten, so wird über die Fehlerroutine nach ENDE gesprungen und das Programm beendet. Im anderen Falle wird das Programm bei

GO1 fortgesetzt. Hier wird in der BLDL-Liste, in der eine Kopie des Directory-Eintrages des zu ändernden Members steht, geprüft, ob das Member numerierte Zeilen hat. Wenn nein, wird ein Fehlercode in LCC12 gesetzt und nach ENDE gesprungen.
Sonst wird das Programm bei

GO11 weitergeführt. Nun werden die Listformen der benötigten Macros in die WORKAREA gebracht und bestimmte Felder initialisiert. Mit POINT wird veranlaßt, daß der SPOOLDS von seinem Anfang an gelesen wird. FIND setzt die Adresse des ersten Blockes des Members in den DCB, damit er ebenfalls von Anfang an gelesen wird.

BLOCKIN Nun wird mit READ ein Block des Members eingelesen und mit CHECK geprüft.
Wurde bei der Prüfung festgestellt, daß das Member zu Ende ist, veranlaßt die CHECK-Routine einen Sprung nach ENDWORK oder bei einen Input-Error nach PARTINEL und ENDE.
Im Normalfall hat ein Block 1600 Byte = 20 Zeilen. Da der letzte Block im allgemeinen nicht die volle Länge hat, wird nun die Blocklänge festgestellt und der Wert für das Vergleichsregister (R11) errechnet. Dieser Wert definiert beim Durchsuchen eines Blockes nach der zu ändernden Zeile das Blockende.
Nun schließt sich eine Abfrage an, die feststellt, ob die Zeilennummer des Members links bis zur Spalte 73 mit Nullen aufgefüllt ist. Wenn ja, wird nach NULLDA gesprungen. Sonst folgt bei

NORD eine Routine, die diese Nullen einsetzt. Nach ihrem Ende folgt

NULLDA mit einer Abfrage (STRBYTE), ob schon alle Änderungszeilen abgearbeitet sind, d.h. ob der SPOOLDS zu Ende (EODAD) gelesen ist. Ist dies der Fall, wird nach BLOCKOUT gesprungen. Sonst folgt die Frage (STRBYTE), ob die zuletzt eingelesene Änderungszeile schon verarbeitet ist. Ist dies der Fall, wird das Programm bei COMPARE fortgesetzt. Andernfalls wird bei

RECORDIN die nächste Änderungszeile eingelesen und mit CHECK der Einlesevorgang geprüft. Trat ein Input-Error auf, wird, nach SPOOLINE in die Fehlerroutine gesprungen und von dort geht es nach ENDE.
Ist der SPOOLDS hingegen leer, so wird bei

SPOOLEND in das STRBYTE ein Bit gesetzt das anzeigt, daß auf dem SPOOLDS keine Änderungszeilen mehr sind und man springt nach BLOCKOUT und schreibt den Block wieder in die Bibliothek. Nach dem Schreiben folgt ein Sprung nach BLOCKIN, um den nächsten Block einzulesen. Von da geht es über NULLDA wieder nach BLOCKOUT, so lange, bis das ganze Member zurückgeschrieben ist.
Ist das READ bei RECORDIN hingegen erfolgreich, wird das Programm bei

WORK1 fortgesetzt.

Als erstes steht hier eine Abfrage, ob die Zeilennummer der neuen Änderungszeile kleiner ist als die der letzten Änderungszeile. Ist sie größer oder gleich, wird nach WORK2 gesprungen. Im anderen Fall war eine Änderungszeile nicht in der richtigen Reihenfolge, der Fehlerzähler wird in

NUFE1 um eins erhöht und anschließend nach RECORDIN gesprungen.

WORK2 Das Bit, das anzeigt, daß ein Record verarbeitet ist, wird zu-

COMPARE rückgesetzt. Nun wird der Block nach der Zeilennummer der eingelesenen Zeile durchsucht. Ist eine Zeilennummer im Block gleich oder größer der Änderungszeile, so springt man nach WØRKREC. Ist die gesuchte Zeilennummer hingegen nicht im Block, wird nach

WORKREC BLOCKOUT gesprungen. Hier wird gefragt, ob die beiden Nummern gleich sind. Wenn ja geht es nach WORK3, wenn nein, nach NUFE1

WORK3 und von dort nach RECORDIN. Zuerst wird hier der Inhalt der Register abgespeichert, die zum Durchsuchen des Blockes benötigt wurden. Nun wird geprüft, ob die Ziffern, welche VZKL angeben, wirklich Ziffern sind. Wenn nicht, geht es über NUFE nach RECWEND. Sind es Ziffern, so werden sie in eine Binärzahl umgewandelt. Ist VZKL = 0, geht es nach NUFE. (VZK darf nicht die Länge 0 haben). Nun trifft man die Vorbereitungen zum Durchsuchen der gefundenen Zeile im Block auf die VZK. Dies geschieht in der

RECLOOP Schleife. Wurde sie gefunden, wird nach VORMOVE gesprungen.

VORMOVE Anderenfalls geht es über NUFE nach RECWEND. Man berechnet nun die Adresse der nächsten beiden Ziffern, welche die Länge der EZK angegeben und die Adresse der EZK. Die Ziffern werden ge-

NUFE prüft. Sind es keine Ziffern, geht es nach NUFE, sonst nach PA. Hier wird der Fehlerzähler um eins erhöht und nach RECWEND ge-

PA sprungen. Die EZKL wird in eine Binärzahl umgewandelt. Sollte diese Länge 0 sein, wird nach EILANULL gesprungen.
Nun wird berechnet ob der Einschub nicht zu groß ist.

Sollte das der Fall sein, kann die Änderung nicht ausgeführt werden und es geht nach NUFE, sonst nach

EXEC Die beiden Längenangaben werden nun verglichen

EZKL > VZKL Sprung nach LAENGER
EZKL < VZKL " " KUERZER
EZKL = VZKL weiter in der Befehlsfolge.

An die Stelle der VZK wird die EZK gesetzt und nach RECWEND

LAENGER gesprungen. Zuerst muß nun geprüft werden, ob beim Einschieben der EZK und dem davor nötigen Verschieben der restlichen Zeile hinter der Einschubstelle keine Zeichen aus der Zeile geschoben werden. Hier wird nun berechnet, wieviel Zeichen am Zeilenende vor der Zeilennummer Leerzeichen sein müssen.
In der Schleife BLANKCOM wird diese Abprüfung vorgenommen. Ist nicht die nötige Zahl Leerzeichen vorhanden, kann die Änderung nicht ausgeführt werden und es geht nach NUFE. Sonst folgt jetzt

VERSCHIEBE Berechnung der Adressen von wo nach wo der hinter der Einschubstelle liegende Teil der Zeile geschoben werden soll und seine Länge. Ist diese Länge gleich 0 wird das Verschieben übersprungen und nach NOEXMOVE gegangen. Sonst erfolgt jetzt das Verschieben über ein Zwischenfeld. Darauf folgt

NOEXMOVE jetzt wird die EZK eingesetzt. Anschließend wird nach RECWEND gesprungen.
Im Befehlsablauf des Programmes folgt nun

KUERZER Hier wird zuletzt die EZK eingesetzt. Danach wird berechnet, um wieviel Bytes der hinter der Einschubstelle liegende Teil der Zeile nach vorne geschoben werden muß, um die entstandene Lücke zu schließen.

ELNULL Berechnung der Länge der zu verschiebenden Zeichen. Ist sie 0 geht es nach NOEXMOV2, sonst werden sie jetzt verschoben.

NOEXMOV2 Es wird berechnet, wieviel Bytes am Ende der Zeile (bis Spalte 72 einschließlich) mit Leerzeichen aufgefüllt werden müssen, um
BLANKMOVE alte Zeichen zu überschreiben. Die Leerzeichen werden geschrieben
EILANULL und anschließend nach RECWEND gesprungen. Die EZKL ist Null. Man berechnet, bis wohin der hinter der Einschubstelle liegende Teil der Zeile geschoben werden muß. Und springt nach ELNULL.

RECWEND Eine Änderung ist nun durchgeführt. War in der Änderungszeile ein Fehler, wurde dieser gezählt und die Änderung nicht durchgeführt. Die Zeilennummer der eben bearbeiteten Zeile wird gespeichert, in das STRBYTE wird ein Bit gesetzt, das zeigt, daß die Änderungszeile verarbeitet wurde. Die bei WORK3 gespeicherten Registerinhalte werden zurückgeladen und nach RECORDIN gesprungen.

BLOCKOUT Es folgt die Ausgabe-Routine. Mit WRITE wird der Block auf die Bibliothek geschrieben. Nach WAIT wird geprüft, ob das Schreiben erfolgreich war. Wenn nein, geht es in die Fehlerroutine, wo unterschieden wird, ob die Bibliothek voll ist oder ein I/O-Error vorliegt. In beiden Fällen wird ein Fehlercode gesetzt und nach ENDE gesprungen. Bei erfolgreichem WRITE wird beim ersten Durchlauf mit NOTE die Adresse des Blockes auf der Platte festgestellt und gerettet. Von hier wird wieder nach BLOCKIN gesprungen, bis alle Blöcke umgeschrieben und alle Änderungen

ENDWORK durchgeführt sind. Das Member ist geändert und auf die Bibliothek zurückgeschrieben. Nun muß der Directory-Eintrag geändert werden. Dazu wird ein STOW R mit allen Einträgen gemacht. Ist das Directory voll oder ist ein Fehler aufgetreten, geht es über die Fehlerroutine nach ENDE, sonst nach

BESSER Hier prüft man den Fehlerzähler, ob Fehler gemacht wurden. Wenn ja, geht es in eine spezielle Fehlerroutine. Alle in NUFE1 und NUFE aufsummierten Fehler werden zusammengefaßt. Bis zu 4 Fehlern wird die Zahl unterschieden, bei 5 und mehr Fehlern nicht mehr.

Das Programm wird fortgesetzt bei

ENDE Es folgt ein CLOSE auf den SPOOLDS.
Das Programm wird beendet mit den Macro HE (Rückkehr zu TCPØPROC (ATER).

6. TCP1SCH (M. Schayegh)

Dieser Module wird nach dem Befehl 'Schreibe, Gerät, Name' durch TCPØPROC mit ATTACH aktiviert.

FUNKTION

Das spezifizierte Member wird am Terminal gedruckt oder in Lochstreifen gestanzt.

EINGANGSPARAMETER

LAMEMBN : Membername

AUSGANGSPARAMETER

LCC12 : Fehlercode

PROGRAMMABLAUF

Nach Macro ALEPH wird abgefragt, ob der Membername mit # anfängt. Wenn ja, dann handelt es sich um einen "SYSTEM-OUTPUT" In diesem Fall wird nach GOOTSP gesprungen, dort in ACTION+1 ein Bit gesetzt und über Macro HE das Programm beendet. Sonst werden die Adressen von EODAD- und SYNAD-Routinen für READ in PARTDCB geladen. POINT stellt die Anfangsadresse des sequentiellen Datasets für das nächste WRITE zur Verfügung. FIND stellt die Anfangsadresse des Members für das nächste READ zur Ver-

LOOPBL fügung. Durch READ wird ein Block in BUFFER1 gelesen. Register (RE) enthält nach dem READ die Länge des gelesenen Blocks minus

LOOPREC 80. Hier wird der gelesene Block, durch WRITE recordweise ausgegeben. Ist ein Record ausgegeben, so wird Register (6), das die Adresse des zuletzt ausgegebenen Record enthält, um 80 erhöht und durch BXLE ein Sprung zu LOOPREC gemacht und der nächste Record ausgegeben.

Der Inhalt von Register (RE) bestimmt, wie oft nach BXLE ein Sprung nach LOOPREC gemacht werden soll. Ist auf diese Weise ein Block verarbeitet, so wird nach LOOPBL gesprungen und der nächste Block gelesen, bis "End of data" erreicht ist.

DATAEND1 Dann wird nach ENDE gesprungen und das Programm beendet.

IOFE Kommt bei READ oder WRITE oder FIND ein Fehler vor, so werden

ENDE einige Felder des betreffenden DCB neu initialisiert. Ein CLOSE TYPE = T und Macro HE schließt das Programm. (Rückkehr zu ATER).

7. TCP1TS (M. Schayegh)

Der Module wird nach dem Befehl 'ZL, Name, AA.EE' durch TCPØPROC mit ATTACH aktiviert.

FUNKTION

Das Programm gibt denjenigen Bereich des spezifizierten Members, welcher durch Anfangs-Nr. AA und End-Nr. EE festgelegt ist, auf den Drucker aus. Anfangs- und End-Nr. sind maximal 8-stellige Zahlen und durch einen Punkt getrennt.

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

LCC12 : Fehlercode

PROGRAMMABLAUF

Nach Macro ALEPH werden EDDAD und SYNAD-Adressen in den DCB ein-

X gesetzt. Diese Routine prüft die Anfangs- und Endnummer auf Gültigkeit, füllt sie von links bis zu 8 Zeichen mit führenden Nullen und überträgt sie in die Feldern ANFNR und ENDNR. Macro FIND stellt die Anfangsadresse des Members für das nächste READ

GUT zur Verfügung. Hier wird ein Block des geforderten Members gelesen und die Endadresse des Blockes in R9 abgespeichert. Falls die Sequenz-Nr. des gelesenen Blockes mit führenden Leerzeichen versehen ist, so wird nach SET gesprungen, dort die führenden Leerzeichen durch führende Nullen ersetzt und nach BACK zurückgesprungen.

BACK
GLEICH
XYZ

Falls der erste gelesene Block verarbeitet worden ist, wird nach XYZ gesprungen, sonst je nachdem, ob AA = EE, AA > EE oder AA < EE ist., werden verschiedene Switches oder Fehlercodes gesetzt. Bei AA = EE wird solange in dem gelesenen Block gesucht, bis der erste Record (AA) gefunden wird. Dann wird nach EINZIG gesprungen, dort der Record ausgegeben und das Programm beendet. Wird der erste Record (AA) in dem gelesenen Block nicht gefunden, so wird mit dem Sprung nach GUT ein neuer Block gelesen und dann weitergesucht. Ist der erste Record gefunden, so werden die Daten recordweise ausgedruckt. Dies erfolgt solange, bis der letzte Record ausgedruckt ist. Erreicht man das Ende des gelesenen Blockes bevor der letzte Record verarbeitet ist, so wird nach GUT gesprungen, dort ein neuer Block gelesen und weiter abgearbeitet.

SET — versorgt die Anfangs- und Endnummer mit führenden Nullen.

EINZIG — Hier wird bei AA = EE ein Record ausgegeben.

END — Nach CLOSE und TTIMER wird das Programm mit Macro HE beendet. (ATER)

8. TCP1LGES (M. Schayegh)

Dieser Module wird nach dem Befehl 'LO, NAME, A' von TCPØPROC durch ATTACH aktiviert.

FUNKTION

Der Loadmodule TCP1LGES führt den Befehl 'LOESCHE, NAME = ..., BEREICH = A' aus. Das angegebene Member wird aus dem Directory der Bibliothek gestrichen.

EINGANGSPARAMETER

Adresse der LINEBOX in Register 3: LBXREG
Von der LINEBOX werden verwendet:
LAMEMBN: Name des zu streichenden Members.
PARTDCO: Output-DCB der Bibliothek.

AUSGANGSPARAMETER

LCC12: Fehlercode

PROGRAMMABLAUF

Der Loadmodule TCP1LGES wird vom Programm TCPOPROC aufgerufen. Die CSECT: STREMEMB des aufrufenden Programms beginnt mit dem Macro ALEPH.

Der Fehlercode in LCC12 wird zu X'0000' gesetzt. Anschließend wird in eine 8 Byte lange STOW-Liste STOWLIST der Membername des zu streichenden Members geschrieben. Mit dem STOW Macro (D) wird nur das gewünschte Member gestrichen. War dieser Vorgang erfolgreich, wird nach WEITER gesprungen. Wurde der angegebene Name nicht in der Directory gefunden wird nach NAMNØT und bei einem perm. I/O-Error nach PIØERRØR gesprungen. Es folgen die Marken

NAMNØT

PIØERRØR In beiden Fällen wird nach LCC12 win Fehlercode geschrieben. Im zweiten Fall werden noch im PARTDCO die Fehlerbits zurückgesetzt. Das Programm wird bei

WEITER fortgesetzt und mit dem Macro HE beendet (→ ATER)

9. TCP1LREC (S. Müller)

FUNKTION

Mit dem Loadmodule TCP1LREC wird der Befehl 'LOE, Name BEREICH = XX. YY. XXX. YYY. ...' ausgeführt. Mit diesem Befehl können innerhalb eines Members einzelne Zeilen und Zeilengruppen gestrichen werden. Je ein Zahlenpaar in der Bereichsangabe beschreibt eine zu löschende Zeilengruppe. Die Zahl der zu streichenden Gruppen ist beschränkt durch die Länge des Bereichsfeldes(40 Byte). Das zu ändernde Member wird nach den zu löschenden Zeilen durchsucht, diese werden gelöscht und das Member wird neu auf die Bibliothek geschrieben.

EINGANGSPARAMETER

Adresse der LINEBOX in Reg. 3: LBXREG

Aus der LINEBOX wird hauptsächlich verwendet:

LAMEMBN: Name des zu ändernden Members

PARTDCB, PARTDCØ: Input / Output-DCB's für den Bibliothek Data-Set.

WADR2 Adresse des Feldes in dem die Bereichsangabe steht.

AUSGANGSPARAMETER

LCC12: X'0000' oder Fehlercode

PROGRAMMABLAUF

Der Loadmodul TCP1LREC, CSECT, STRREC, wird aufgerufen vom Loadmodule TCPOPRØC. Die CSECT wird wie üblich eingeleitet mit dem Macro ALEPH.

Gleich am Programm**beginn** werden in den PARTDCB die Ausgangsadressen für EODAD und I/O-Error eingespeichert. Darauf werden die Listformen (PARTDECB und PARTØDECB) für READ und WRITE aufgebaut. Es folgt die Erstellung der Listen (BLDLLIST und STØWLIST) für das BLDL und STØW Macro. In die BLDLLIST wird aus LAMEMBN der Name des zu ändernden Members eingetragen und das Macro BLDL ausgeführt.

Ist der Name in der Directory des Bibliotheks-Data-Set vorhanden, wird nach GØ1 gesprungen, ist er nicht eingetragen nach NAMENØT oder bei einem Per. I/O-Error nach PIØER1.

NAMENØT Ist der Membername nicht im Directory, so wird hier ein Fehlercode nach LCC gesetzt und das Programm über die Marke ENDEX verlassen. Als nächste Marke folgt:

PIØER1 Hier werden die I/O-Fehler des BLDL Macros verarbeitet. Es wird ein Fehlercode nach LCC12 gesetzt, im PARTDCB die Fehlerbits zurückgesetzt und ebenfalls nach ENDE gesprungen. Es wird geprüft, ob die Daten numeriert sind. Wenn ja, geht es weiter nach GØ11, wenn nein, nach BLANKNUM. Bei unnumerierten Records ist ein Streichen von bestimmten Records nicht möglich.

GØ1

GØ11 wird angesprungen, wenn das gesuchte Member vorhanden und numeriert ist. Mit einem FIND Macro wird nun die Anfangsadresse des Members in den Input DCB (PARTDCB) eingetragen. Nun werden zwei Felder, in die später die jeweiligen Anfangs- und Endnummern der zu löschenden Zeilenbereiche eingetragen werden, zu 0 gesetzt. Nun folgt die Routine zur Formatierung und Prüfung der im Bereichsfeld angegebenen Anfangs- und Endnummern. Die Adresse dieses Feldes steht in WADR2. Zuerst wird geprüft, ob das erste Zeichen im Bereichsfeld ein Blank ist. Wenn ja, wird nach FØRMFEBL gesprungen Beginnt das Bereichsfeld mit einem Zeichen, wird in der Befehlsfolge fortgefahren und verschiedene Felder auf einen Anfangswert

FØRMLØØP gesetzt. Der Ziffernzähler wird zu 0 gesetzt und die Anfangsadresse der ersten oder nächsten Zahl im Bereichsfeld wird gespeichert.

FØRMZIPR In einer Schleife, die hier beginnt, wird jedes Zeichen geprüft, ob es eine Ziffer ist. Wird ein Zeichen gefunden, das keine Ziffer ist, wird nach FØRMNØZI gesprungen.
Ist es aber eine Ziffer, wird der Ziffernzähler um 1 erhöht, um so die Länge der Zahl festzustellen. Solange der Ziffernzähler < 9 ist, wird wieder nach FØRMZIPR gesprungen um das nächste Zeichen zu prüften. Sonst wird die Schleife verlassen und nach FØRMLZFA gesprungen; die Zahl hatte mehr als 8 Ziffern. Bei der Marke

FØRMNØZI wird das erste Zeichen, das keine Ziffer war geprüft, ob es ein Punkt (C'.') ist. Wenn ja, folgt ein Sprung nach FØRMMØVE. Sonst wird geprüft ob es ein C' ' ist. Ist es auch kein C' ' so wird nach FØRMFLZA gesprungen. In der Bereichsangabe wurde ein Fehler gefunden. Wurde aber ein C' ' gefunden, so wird in das Feld FØRMENDB ein C'*' gesetzt, als Zeichen dafür, daß die letzte Zahl in der Bereichsangabe gefunden wurde. Ist dabei aber der Ziffernzähler = 0, so besagt das, daß gar keine Zahl angegeben war und es wird nach FØRMFLZA gesprungen. (Nach einem Trennzeichen C'.' im Bereich stand ein C' '.) Wurde im bisherigen Verlauf der Routine eine Zahl gefunden, so wird bei

FØRMMØVE weitergearbeitet.

Hier wird die gefundene Zahl so mit Nullen ergänzt, daß sie 8 Ziffern hat. Man berechnet darum, wieviel Nullen man braucht um die Zahl links auf 8 Stellen aufzufüllen. Werden keine Nullen benötigt, wird nach FØRMNØMO gesprungen
Sonst werden die benötigten Nullen in das Feld STGRUPPF geschrieben. In das Feld STGRUPPF werden alle vormatierten Bereichszahlen geschrieben. Sie stehen ohne Abstand hintereinander und sind links mit Nullen auf 8 Stellen aufgefüllt.
Bei der nun folgenden Marke

FØRMNØMO wird die gefundene Zahl hinter die eventuell eingesetzten Nullen in das Feld STGRUPPF geschrieben.
Der Paarzähler: PAAR wird um eins erhöht. Ist PAAR = 2 so wird nach FØRMPAAR gesprungen. Sonst wird gefragt ob das FØRMENDB (C'*') gesetzt ist. Wenn ja, geht es nach FØRMNØPA, sonst nach

FØRMPAAR FØRMGØ1. Ein Zahlenpaar, Anfangs- und Endnummer ist gefunden worden. Ein Zähler: PAARBL wird um 1 erhöht und nach FØRMGØ1

FØRMNØPA gesprungen. Das letzte Zahlenpaar kam nicht zustande. Der Grund: die Zahl der eingegebenen Zahlen war ungerade oder in einer Zahl wurde ein Fehler gefunden, so daß ein angefangenes Zahlenpaar nicht vollendet werden konnte. Nach LCC12 wird ein Fehlercode

FØRMGØ1 gesetzt und nach FØRMGØ2 gesprungen. Ist das FØRMENDB nicht gesetzt, geht es nach FØRMLØØP um die nächste Zahl zu bearbeiten.

FØRMLZFA Ist es aber gesetzt geht es nach FØRMGØ2. Nach LCC12 wird ein Fehlercode gesetzt, wenn bei der zuletzt bearbeiteten Zahl ein Fehler auftrat. Die restlichen Zahlen im Bereich werden bei allen Fehlern nicht bearbeitet. Das Programm wird fortgesetzt bei

FØRMGØ2 mit der Abfrage, ob der PAARBL=0 ist. Bei nein wird nach FØRMGØ3 gesprungen. Sonst wird bei

FØRMFEBL weitergearbeitet. In diesem Fall ist kein Zahlenpaar zustandegekommen. Es wird ein Fehlercode gesetzt und nach ENDE gesprungen.

FØRMGØ3 ist die Fortsetzungsstelle der Routine. Alle richtigen Zahlen wurden jetzt formatiert.

Nun werden alle Zahlenpaare geprüft, ob die Anfangsnummern eines Bereiches gleich oder kleiner der Endsumme ist. Trifft diese Bedingung bei einem Zahlenpaar nicht zu, wird ein Fehlercode nach LCC12 gesetzt und der PAARBL = Paarzähler mit der Anzahl der richtigen Paare überschrieben. Die Prüfung wird abgebrochen und das Programm bei FØRMGØ5 fortgesetzt. Das erste Zahlenpaar wird in die Felder ANFRECN nach ENDRECN gebracht, die Adresse des nächsten Paares wird gespeichert und die Zahl der Paare um 1 erniedrigt. Bei

GØ2 werden Steuer-Bits zu 0 gesetzt. Nun werden die Register geladen, welche zum Durchsuchen der einzelnen Blöcke benötigt werden. Bei

LØØPREAD beginnt die Verarbeitung und Änderung des Members. Ein Block wird in das Feld: OLDBUFF gelesen. Treten dabei Fehler auf, so wird nach ERREADPA gesprungen. Ist das Member zu Ende, so geht es nach PDSENDE. Das Adressregister OAR für das Blockfeld OLDBUFF wird mit der Anfangsadresse des Feldes geladen und das Vergleichsregister OVR wird, wenn nötig, der Blocklänge des eingelesenen Blockes angepaßt. Nun werden einige Bits abgefragt. Die Bedeutung dieser Bits sei hier zusammengefaßt angegeben.

BIT2 : Ein Block, ØLDBUFF, ist verarbeitet. Es soll ein neuer eingelesen werden. Normales Durchsuchen

BIT3 : Ein zu streichender Bereich liegt nicht innerhalb eines einzelnen Blockes. Es muß ein neuer Block eingelesen werden. Anschließend soll in die Streiche-Routine zurückgesprungen werden (GØ3 BACK)

BIT10 : Alle zu streichenden Bereiche sind gestrichen. Der restliche Memberteil kann übertragen werden.

BITENDE : Data Set ist zu Ende.

Zuerst wird geprüft ob die Numerierung der Zeilen im Block 8-stellig ist, d.h., ob sie links, wenn nötig bis zur Spalte 73 einschließlich mit Nullen aufgefüllt ist. Wenn nein, wird nach SET gesprungen, die Numerierung des eingelesenen Blockes ergänzt und nach BACK1 zurückgesprungen. Ist die Numerierung vollständig wird das Programm gleich bei

BACK1 fortgesetzt.

Ist BIT3 gesetzt, geht es nach GØ3BACK. Sonst folgt die Marke

LØØPCØMP mit der Frage ob BIT10 gesetzt ist. Kann das bejaht werden, wird nach DØMØVE gesprungen. Sonst wird jetzt die laufende Recordnummer mit der Anfangsnummer eines zu streichendn Bereiches, ANFRECN verglichen. Ist die Recordnummer kleiner, geht es nach DØMØVE, ist sie gleich oder größer, geht es nach LØØPSTR. In der Befehlsliste folgt jetzt die Umschreiberoutine mit der Anfangsmarke DØMØVE. Mit einem MVC-Befehl wird ein Record vom Feld OLDBUFF in das Feld NEWBUFF geschrieben. Dann wird das Oldbuffer-Adressregister (ØAR) auf die Adresse des nächsten Records gesetzt. Ist der OLDBUFF nun leer, so wird BIT2 gesetzt. Bei

BACK2 wird das Neubuffer-Adressregister NAR (NAR) um die Länge eines Records (80) erhöht.

Ist das Feld NEWBUFF voll, so wird nach WRITEOUT gesprungen. Sonst wird bei

ENDWR1 fortgefahren mit der Frage, ob das ENDBIT gesetzt ist. Wenn ja, ist das Member ganz übertragen und es wird zur Marke WEITER gesprungen. Sonst wird gefragt, ob BIT2 gesetzt ist. Ist es nicht gesetzt, wird bei LØØPCØMP der nächste Record verarbeitet. Im anderen Fall, wird das BIT2 zurückgesetzt und bei LØØPREAD der nächste Block eingelesen.

In der Befehlsfolge folgt jetzt die Marke

WRITEOUT Mit dem Macro WRITE wird ein Block aus dem Feld NEWBUFF wieder auf dem Bibliotheks-Data-Set zurückgeschrieben. Das Neuebuffer-Adressregister wird wieder auf die Anfangsadresse von NEWBUFF zurückgesetzt und der Blockzähler um 1 erhöht. Nach einem WAIT-Macro wird geprüft ob das WRITE erfolgreich war. Wenn nein, wird in eine Fehlerroutine gesprungen, die mit der Marke ERRORWR beginnt. Sonst wird jetzt nach der Ausgabe des ersten Blockes mit dem Macro NOTE festgestellt, auf welche Adresse (TTR) dieser Block geschrieben wurde. Beim ersten und allen weiteren Blöcken wird danach nach ENDWR1 gesprungen.

In der Befehlsliste des Programmes folgt jetzt die Streiche-Routine mit der Anfangsmarke.

LØØPSTR Darin wird immer die Adresse im Oldbuffer-Adressregister OAR um eine Recordlänge (80) erhöht. Ist das Feld OLDBUFF leer, wird nach

DØSTR GØ3 gesprungen. Sonst wird die Nummer des neuen Records mit der Nummer im Feld ENDRECN verglichen. Solange die Nummer des Records gleich oder kleiner ist, wird nach LØØPSTR zurückgesprungen. Sonst wird der Durchlauf fortgesetzt mit der Frage, ob schon alle zu streichenden Bereiche gestrichen sind. Wenn nein, wird die Anfangs- und Endnummer des nächsten Bereiches nach ANFRGCN und ENDRECN gebracht und bei LØØPCØMP mit der Bearbeitung des Members

BITSET10 weitergefahren. Sind alle Bereiche gestrichen, so wird BIT10 gesetzt und ebenfalls bei LØØPCØMP fortgefahren.

GØ3BACK BIT3 wird zu Null gesetzt und nach DØSTR gesprungen.

GØ3 BIT3 wird gesetzt und nach LØØPREAD gesprungen. Hiermit ist die Streicheroutine beendet. In der Befehlsliste folgt jetzt die Marke

PDSENDE (PDS-Ende-Routine)
Das Member ist ganz eingelesen. Ist nun das Feld NEWBUFF bereits ausgegeben, kann nach WEITER gesprungen werden. Sollte dies nicht der Fall sein, so wird berechnet, wieviel Bytes noch vom Feld NEWBUFF auf die Bibliothek geschrieben werden müssen. Dann wird das BITENDE Bit gesetzt und nach WRITEØUT gesprungen.
Bei dem Namen

WEITER wird die Liste für ein STØW-Macro aufgebaut. Außer der Adresse des Members (TTR), der neuen Blockzahl und der Änderungszeit werden alle Angaben über das Member aus der BLDLIST entnommen. Mit einem STØW R wird der Directory-Eintrag des Members auf den neuesten Stand gebracht. Ist beim STØW-Macro ein Fehler aufgetreten so wird nach NØNAME, DIRFULL oder PIOER3 gesprungen, sonst wird das Programm bei ENDE abgeschlossen.
Nun folgen in der Befehlsliste einige Fehlerroutinen. Bei

NØNAME

DIRFUL wird ein Fehlercode nach LCC12 gespeichert. Bei

PIOER3 werden Fehlerbits im PARTDCØ gelöscht und ebenfalls ein Fehlercode nach LCC12 geschrieben. Nach der Marke

ERREADPA werden Lesefehler auf dem Bibliotheks Data Set behandelt und ebenfalls ein Fehlercode nach LCC12 gebracht.

Fehler, die beim Schreiben auf den Bibliothek Data Set auftreten werden

ERRØRWR behandelt. Dabei wird zwischen I/O-Error und einer vollen Bibliothek unterschieden.

In beiden Fällen wird ein Fehlercode nach LCC12 gesetzt. Bei

BLANKNUM wird ein Fehlercode gesetzt, wenn das Member das Format ONU hatte.

Nach all diesen Fehlerroutinen wird nach ENDE gesprungen. Es folgt nun mit der Marke

SET die Routine, mit der die Zeilennummer, wenn nötig, links auf 8 Stellen mit Nummern aufgefüllt wird. C' ' werden durch C'0' ersetzt. Jeweils ein Block wird bearbeitet. Am Ende wird nach BACK1 zurückgesprungen.

In der End-Routine wird bei

ENDE das Feld, in dem die Bereichsangabe stand, mit FREEMAIN freigegeben.

ENDEX Das Programm wird mit dem Macro HE beendet (→ATER)

10. TCP1LOUT (S. Müller)

FUNKTION

Das Programm TCP1LOUT, CSECT: LOEOUT löscht Output-Data-Sets, die durch den TCPWTR aufgebaut werden, wenn als Output-Klasse SYSOUT = X angegeben wurde. SYSOUT = X ist die Output-Klasse, die für Programme (JCL) angegeben wird, welche mit dem Befehl 'START' gestartet werden und eine Ausgabe der Ergebnisse auf dem Terminal erzeugen sollen. Der TCPWTR gibt den Output-Data-Sets Namen in der Form: TCP.XXX.#YYYZZNN

XXX= USERID

YYY= Bei Output-Data-Sets mit
SYSOUT = X : OUT
Bei Messages-Output mit
MSGGLASS = X : MSG

ZZ = Die letzten beiden Zeichen des JOB - Namens

NN = Nummer zwischen 00 und 99

Ist ein mit 'START' gestarteter JOB zu Ende,so wird dem Benutzer mitgeteilt, unter welchem Namen seine Ausgabe zu finden ist. Er erhält den letzten Teil des Namens = YYYZZNN. Der Benutzer kann sich den Output mit 'SCHREIBE' ausdrucken lassen oder mit 'LOESCHE' vernichten. Data-Sets, die mit 'SCHREIB' ausgegeben wurden, aber vor dem Ende der Ausgabe abgebrochen wurden, sind noch erhalten und können mit 'LÖSCHE' ebenfalls beseitigt werden. Mit TCP1LOUT kann eine ganze Output-Serie mit gleichem YYYZZ von 00 ÷ 99 vernichtet werden, gleichgültig, ob die Nummerierung durchgehend oder lückenhaft ist.

EINGANGSPARAMETER

Adresse der LINEBOX im Reg. 3 = LBXREG. Folgende Felder aus der LINEBOX werden besonders benötigt:

LAMEMBN Data-Set-Name (DSN) der Data-Set-Serie
= YYY ZZ NN (erster)

WADR2 A (BEREICHSFELD)
und in dem BEREICHSFELD die Endnummer der DSN.

AUSGANGSPARAMETER

LCC12 = X'0000' oder Fehlercode

PROGRAMMABLAUF

Der Loadmodule TCP1LOUT wird vom Module TCPOPRØC mit ATTACH aufgerufen. Er wird eingeleitet mit dem Macro ALEPH. Anschließend wird LCC12 zu X'FFFE' gesetzt. Über die Adresse in WADR2 wird jetzt die Nummer des letzten Data-Sets geholt und nach DSNENDNR gespeichert. Mit FREEMAIN wird anschließend das Übergabefeld freigegeben. Nun wird geprüft ob die letzten beiden Zeichen des Namens in LAMEMBN und die DSNENDNR Ziffern sind. Wenn nein, wird nach NDZA gesprungen, dort ein Fehlercode in LCC12 gesetzt und zur Marke ENDE gegangen. Sonst wird als nächstes geprüft, ob die Nummer, welche die letzten beiden Zeichen in LAMEMBN bildete, kleiner oder gleich der Nummer in DSNENDNR ist. Diese beiden Nummern geben die Serie der zu löschenden Data-Sets an, wobei sich die Data-Set-Namen nur durch die Nummern unterscheiden.

Ist die Nummer in LAMEMBN größer als die Nummer des letzten zu löschenden Data-Sets in DSNENDNR, so werden keine Data Sets gelöscht, sondern nach ENDKLEIN gesprungen, ein Fehlercode nach LCC12 geschrieben und das Programm bei ENDE beendet. Sind die Nummern richtig, so können jetzt die CAMLISTen für OBTAIN, LOCATE, SCRATCH und CATALOG aufgebaut werden. Der Data-Set-Name wird aus dem festen Teil: TCP, der USERID und LAMEMBN zusammen gesetzt und nach DSN geschrieben. Nun kann in die Löschschleife eingetreten werden. Sie beginnt mit

LOELOOP Diese Schleife wird mit dem Macro LOCATE begonnen. Damit liest man von Katalog einen Block, in dem Volume Nr. und der Device Code steht. Damit ist bekannt, auf welchem Volume der zu löschende Data Set steht. Ist der Data-Set-Name im Katalog nicht vorhanden, so wird nach NEXTNAME gesprungen, sonst nach

GOOBTAIN Die Volume-Nr. und der Device-Code werden für den späteren Gebrauch umgespeichert. Dann folgt das Macro OBTAIN, das aus der VTOC des gefundenem Volume's den DSCB liest. Kann der DSCB gelesen werden, steht fest, daß der Data-Set noch vorhanden ist und man ihn streichen kann. Darum springt man in diesem Fall nach GOSCRATC. Ist der Data Set bereits gelöscht, besteht sein Eintrag nur noch im Katalog und man springt nach GOUNCATL.

GOSCRATC Hier wird mit dem Macro SCRATC der DSCB des Data Sets gestrichen und der Platz, den er eingenommen hat, für den weiteren Gebrauch freigegeben. Es folgt GOUNCATL mit dem Macro CATALOG. Mit ihm wird der Katalogeintrag für den zu streichenden Data-Set gelöscht. Jedesmal, wenn die Schleife bis hierher durchlaufen ist, wurde ein Data Set gelöscht. Tritt bei einem der vier Macros ein Fehler auf so wird nach ER gesprungen, ein Fehlercode nach LCC12 geschrieben und der Programmablauf bei ENDE beendet. Ist kein Fehler aufgetreten, so folgt die Marke

NEXTNAME Die Data-Set-Nr. wird um eins erhöht und so der Name für den nächsten zu löschenden Data Set gefunden. Ist die Nummer größer als die DSNENDNR, so sind alle Data Sets gelöscht. Sonst wird die Schleife mit dem neuen Namen wieder bei LOELOOP begonnen. Sind alle Data Sets gelöscht wird LCC12 X'0000' gesetzt. Es folgt die Marke

ENDE Bei dem mit dem Macro HE das Programm beendet wird. (→ATER)

11. TCP1KOP (M. Schayegh)

Dieser Module wird nach dem Befehl 'KOP, NEUER NAME, ALTER NAME' von TCPØPROC durch ATTACH aktiviert.

FUNKTION

Mit dem Loadmodule TCP1KØP wird der Befehl 'KOP, NEU= ..., ALT = ...' ausgeführt.
Wenn der 'neue' Membername noch nicht im Inhaltsverzeichnis der Bibliothek verwendet wurde, so wird das mit dem 'alten' Name bezeichnete Member kopiert und die Kopie mit dem 'neuen' Name versehen.
Das alte Member bleibt erhalten.

EINGANGSPARAMETER

Adresse der LINEBOX in Register 3: LBXREG.
Es werden folgende Felder aus der LINEBOX verwendet.
LAMEMBN: Name des vorhandenen Members.
WADR1 : Feld, in dem der Name für die Kopie steht. (Neuer Membername, 8 Byte lang)
PARTDCB: Input
PARTDCØ: Output
} DCB's für die Bibliothek

AUSGANGSPARAMETER

LCC12 : Fehlercode

PROGRAMMABLAUF

Der Loadmodul TCP1KØP wird aufgerufen vom Programm TCPØPROC.
TCP1KØP, CSECT: COPY wird eingeleitet mit dem Macro ALEPH.
Zu Beginn werden die Synad - und Eodad-Adressen in den PARTDCB gespeichert, die BLDL- und STOW-Listen aufgebaut, sowie die Listformen für die Macro READ und WRITE erstellt. Nun wird mit dem Macro BLDL geprüft, ob der Name für die Kopie (NEU=) schon als Membername verwendet wurde.

Wenn ja oder wenn I/O-Error aufgetreten ist, werden diese Fehler bei

NAMD

und bei

PIOR

bearbeitet, ein Fehlercode gesetzt und nach

ENDE

gesprungen. Wenn der Name noch nicht im Inhaltsverzeichnis verwendet wurde, wird das Programm bei

WEIT

fortgesetzt. Man prüft nun, ob der Name des zu kopierenden Members (ALT =) im Inhaltsverzeichnis eingetragen ist. Wenn ja, geht es nach WEITER. Ist der Name nicht vorhanden oder ist ein I/O-Error aufgetreten so werden diese Fehler bei

LISTNOT

und

PIOERROR

behandelt und dann nach ENDE gesprungen.

WEITER

Nach der Vorbereitung eines Zählers wird mit FIND die Anfangsadresse des zu kopierenden Members in den PARTDCB geschrieben.

LOOP

In der jetzt folgenden Schleife wird ein Block mit READ eingelesen und mit WRITE in die Bibliothek zurückgeschrieben. Dies wiederholt sich solange bis das ganze Member kopiert ist. Nach dem Schreiben des ersten Blockes wird mit NOTE seine Adresse fortgestellt. Fehler die beim READ Macro auftreten, werden bei ERRORQ behandelt. Tritt beim Macro WRITE ein Fehler auf, wird nach ERRORQQ gesprungen. Wird bei READ das Ende des Members gefunden, wird über DATENEND nach BADILI gesprungen. Bei

BADILI

wird die STOW-Liste für die Kopie aufgebaut und mit STOW das neue Member im Inhaltsverzeichnis eingetragen. Fehler die bei STOW aufgetreten sind, werden bei

DIRCONNA

(Name schon vorhanden)

DIRFULL

(Directory voll)

PIOERR

(Perm. I/O-Error)

behandelt, ein Fehlercode nach LCC12 gesetzt und dann nach ENDE gesprungen.

ENDEX

Auf diese Marke wird nach STOW gesprungen, wenn kein Fehler auftritt. LCC12 wird zu H'0' gesetzt.

Es geht weiter bei

ENDE Das Feld, in dem der Name für die Kopie stand, wird mit FREEMAIN freigegeben und das Programm mit HE beendet. In der Programmliste folgen noch zwei Fehlerroutinen.

ERRØQ Fehlerroutine für das Einlesen (READ) und

ERRØQQ als Fehlerroutine für das Schreiben (WRITE)

Am Ende beider Routinen wird nach ENDE gesprungen.

12. TCP1DMOD (M. Schayegh)

Dieser Module wird nach dem Befehl'MOD, #D'von TCPØPROC mit ATTACH aktiviert

FUNKTION

Die zugehörige Eintragung in dem Inhaltsverzeichnis für ein bestimmtes Member wird gemäß der Angabe des Benutzers: NAME, ART, FORMAT, CODE geändert.

EINGANGSPARAMETER

R3 = A (LINEBOX)

Anfangsadresse des sequentiellen Datasets (SPDSBLAD), aus den die oben genannten vier Angaben gelesen werden.

AUSGANGSPARAMETER

LCC12 : Fehlercode

PROGRAMMABLAUF

Nach Macro ALEPH werden die Adressen von End-of-data und Synad-Routinen für SEQDCB und PARTDCB in jeweiligen Data Control Block geladen und anschließend die Listformen für den DECB umgespeichert. Macro POINT stellt die Anfangsadresse des sequentiellen Datasets für das nächste READ bereit. Macro READ liest die 4 Angaben Namen, Art, Format, Code die durch Komma getrennt sind, aus dem sequentiellen Dataset in den 80 Byte langen Speicherbereich DIRFELD ein. Die Felder STOWLIST und BLDLLIST werden geräumt.

X — Jetzt wird der Name zeichenweise in BLDLLIST übertragen, bis das erste Komma gefunden wird. Dabei wird darauf geachtet, daß nicht mehr als 8 Zeichen für Name übertragen werden. Ist die Übertragung des Namens zu Ende, so wird nach X2 gesprungen.

X2 — Macro BLDL untersucht, ob ein Member mit dem spezifizierten Namen vorhanden ist. Wenn nicht, oder im Falle eines Ein-/Ausgabefehlers, wird ein Fehlercode gesetzt und das Programm beendet.

WEITER — Register ZR wird als Blockzähler verwendet und Register RGLA mit 1600 geladen, um damit nach jedem READ die Länge des gelesenen Blocks zu errechnen. Macro FIND stellt die Adresse des gesuchten

LOOP — Members für das nächste READ zur Verfügung. Hier wird jeweils ein Block des betreffenden Members gelesen und seine Länge be-

OVER — stimmt. Der gelesene Block wird wieder in die Bibliothek geschrieben. Macro WAIT untersucht den Schreibvorgang auf eventuell auftretende Fehler. Ist WRITE fehlerlos abgelaufen, so wird abgefragt, ob es das erste WRITE ist. Wennja, dann wird die Adresse des ersten Blocks auf der Bibliothek durch NOTE festgehalten, sonst nach LOOP1 gesprungen, dort der Blockzähler um eins erhöht und nach LOOP zurückgesprungen, um den nächsten Block zu verarbeiten. Der Vorgang wiederholt sich solange, bis bei READ End of data erreicht ist. In diesem Fall wird nach BADILI gesprungen.

BADILI — Hier wird eine Liste für Macro STOW aufgebaut. In STOWLIST wird der Name eingespeichert, das Datum eingesetzt, Format aus DIRFELD umgespeichert und angegebene Art und Code von Zeichenkette in Ziffer umgewandelt und eingespeichert. Für ART steht dann in STOWLIST:

1	Statt	JCL
2	"	DATA
3	"	PL1
4	"	FORTR.
5	"	ASS
6	"	ALGOL

ENDE1 Falls das Programm fehlerlos gelaufen ist, so wird in LCC12 Null eingesetzt.

ENDE CLOSE TYPE = T schließt den sequentiellen Dataset. Mit Makro HE wird das Programm beendet (TCPØPROC (ATER))

13. TCP1HOL (M. Schayegh)

Dieser Module wird nach dem Befehl' HOLE, EIGENNAME, DSNAME, FREMDNAME' von TCPØPROC durch ATTACH aktiviert.

FUNKTION

Das Programm kopiert Members aus anderen Datasets oder aus sequentiellen Datasets unter dem Namen "EIGENNAME" in Bibliothek des Benutzers.

EINGANGSPARAMETER

LAMEMBN : Eigenname
WADR2 : A (DS Name)
WADR3 : A (Fremdname) oder Null bei seq.DS

AUSGANGSPARAMETER

LCC12 : Fehlercode

PROGRAMMABLAUF

Nach Macro ALEPH wird durch BLDL abgefragt, ob der eigene Membername schon vorhanden ist oder nicht. Ist dieser Name schon vorhanden, so wird ein Fehlercode gesetzt und das Programm beendet.

NAMEDA

WEITER Sonst wird durch LOCATE das "Volume" ermittelt, das den angegebenen Dataset enthält. Wird dies nicht gefunden, so wird ein Fehlercode gesetzt und das Programm beendet.

FEHLLOC

GUT Sonst wird durch OBTAIN der "DSCB" für den betreffenden Dataset gelesen. Aus "DSCB" wird die Blocklänge des Datasets entnommen. Ist diese genau 1600, so wird ein Sprung zu X11 gemacht. Sonst wird durch GETMAIN ein Speicherplatz mit der betreffenden Länge reserviert und deren Adresse in BUFAD geladen.

BESSER

X11 Zwecks weiterer Abfragen wird für DSORG=BSAM ein Code gesetzt.

X111 Im DCB wird DDNAME=DD - volser eingesetzt und dann die richtige DSORG aus DSCB mit der von Benutzer angegebenen DSORG verglichen und beim Widerspruch nach DSORG gesprungen, dort ein Fehlercode

X1 gesetzt und das Programm beendet. Durch RDJFCB wird der "Job file Control Block" gelesen und darin der DSNAME eingesetzt und durch OPEN TYPE = J der Dataset zum Zugriff bereitgestellt. Wenn es sich um einen sequentiellen Datasatz handelt, dann wird ein Sprung zu

X2 READ gemacht. Sonst wird durch BLDL der fremde Membername gesucht; wenn nicht gefunden, dann wird ein Fehlercode gesetzt und das

GOON Programm beendet. Durch FIND wird die Adresse des fremden Members

READ für das nachfolgende READ bereitgestellt. Es wird ein Block gelesen und seine Länge in R5 geladen. Ist BLKSIZE ≠ 1600, so wird nach Y gesprungen, dort wird der gelesene Block in einem anderen Bereich mit der Länge 1600 umgespeichert. Wenn dieser Bereich genau

WRITE 1600 Byte enthält, dann wird zu WRITE gesprungen. Hier wird der gelesene oder umgespeicherte Block in die Bibliothek geschrieben, die Adresse des ersten Blockes in der Bibliothek durch NOTE festgehalten, Blockzahl um eins erhöht und nach Y1 oder READ zurück-

EODAD gesprungen. (BR R9). Wenn bei READ "End of data" registriert wird. so wird diese Routine aktiv. Wenn BLKSIZE ≠ 1600 ist, dann wird die Länge des letzten Blockes ermittelt und mit einem Sprung zu WRITE der Block in Bibliothek geschrieben und anschließend zu

EODAD1 EODAD1 zurückgesprungen. Sonst bei BLKSIZE = 1600 durch STOW der neue Membername im "directory" eingetragen und bei erfolgreicher

ENDE Eintragung zu ENDE gesprungen. Je nachdem, ob vorher ein OPEN (TYPE=J) ausgeführt wurde oder nicht, wird CLOSE spezifiziert oder zu NURHE gesprungen

NURHE und wenn vorher durch GETMAIN ein Bereich reserviert wurde, dieser

NURHE1 Bereich mit FREEMAIN freigegeben. Macro HE beendet das Programm.

TCP1HOL: BIT-Combinationen

BIT = BXXXXXX1X' = BLKSIZE ≠ 1600
BIT = B'XXXXXXX1' = BSAM
BIT = B'11XXXXXX' = BPAM
BIT = B'XXXX11XX' = OPEN ausgeführt
BIT = B'XX11XXXX' = NOTE ausgeführt

14. TCP1BC2Ø (KETTE) (M. Schayegh)

Dieser Module wird nach dem Befehl 'KET, Name1, Name2, neuer Name' vom TCPØPROC durch ATTACH aktiviert

FUNKTION

Das Programm kettet zwei Members zu einem einzigen Member zusammen.

EINGANGSPARAMETER

WADR1 : A (Name1)
WADR2 : A (Name2)
WADR3 : A (neuer Name)

AUSGANGSPARAMETER

LCC12 : Fehlercode

PROGRAMMABLAUF

X1 Nach Macro ALEPH wird durch BLDL abgefragt, ob der "Name1" vorhanden ist. Wenn ja, dann wird aus der "directory" des ersten Members (Name1) ART+Code+FORMAT für das neue Member übernommen und in STOWLIST umgespeichert. Es wird dann abgefragt, ob das 1.Member numeriert ist. Wenn nicht, dann wird BIT = B'1XXXXXXX' gesetzt.

X11 R5 wird um 1 erhöht, damit nach dem 2.BLDL kein weiterer Rücksprung nach X1 erfolgt. Die Adresse des zweiten Namens wird in R4 geladen und es folgt ein Sprung zu X1, um den zweiten Namen auf seine Existenz in der Bibliothek zu prüfen.

X2 — Hier wird abgefragt, ob das zweite Member numeriert ist. Falls beide Member numeriert sind, dann wird BIT ='X1XXXXXX' gesetzt, damit bei dem neuen Member eine Numerierung erfolgt. Durch BLDL wird geprüft, ob der neue Membername schon vorhanden ist.

NAMEDA — Wenn ja, dann wird ein Fehlercode gesetzt und das Programm

OK — beendet. Sonst wird durch FIND die Adresse des ersten Members

READ — für das nächste READ bereitgestellt. Es wird ein Block in den Speicherbereich AREA eingelesen. R5 enthält A (AREA) und R7 enthält A (AREA+Blocklänge) und mit (BR R9 wird zu A1 oder

MOVE — MOVE gesprungen. Hier wird der gelesene Block 80 Byte-weise in einem Puffer umgespeichert. Wenn es sich bei diesem Block um einen Block aus dem zweiten Member handelt und das neue Member numeriert werden soll, so wird jeweils die Nummer des vorherigen Records um 10 erhöht und im Nummernfeld (Spalte 73-80) des letzten Records gespeichert. Der Anfangswert für diese Inkrementierung ist die Nummer vom letzten Record des ersten Members. Diese Nummer wird bei dem ersten "End of data" festgehalten. Die recordweise Übertragung von AREA nach BUFFER wiederholt sich solange, bis entweder AREA abgearbeitet oder BUFFER voll wird.
Wird zuerst AREA abgearbeitet, so wird ein Sprung zu READ gemacht, dort ein neuer Block in AREA gelesen und nach A1 zurückgesprungen. Wird zuerst BUFFER voll, so wird ein Sprung zu WRITE gemacht, dort der Inhaat von BUFFER in die Bibliothek

WRITE — übertragen und nach MOVE zurückgesprungen. Hier wird der Puffer in die Bibliothek übertragen. Die Anzahl von übertragenen Blöcken (R6) um 1 erhöht und abgefragt, ob es sich um den ersten Block handelt. Wenn ja, dann wird durch NOTE die Adresse des

FIRSTDA — ersten Blocks auf der Bibliothek festgehalten und mit (BR R9)

EODAD — nach STOW oder MOVE zurückgesprungen. Hier wird abgefragt, ob es sich um "End of data" für das zweite Member handelt. Wenn ja, dann wird die Länge des letzten Blocks ermittelt und anschließend zu WRITE gesprungen. Dort der letzte Block in die Bibliothek übertragen und nach STOW zurückgesprungen.

A6 Bezieht sich "End of data" auf das erste Member, so wird abgefragt, ob das zweite Member numeriert werden soll. Wenn ja, dann wird die Nummer vom letzten Record ermittelt und in das Feld NR
A4 abgespeichert. Sonst wird ein Sprung zu FIND gemacht, um die Adresse des zweiten Members für READ bereitzustellen.

SYNAD Diese Routine wird aktiv, wenn bei READ ein Fehler vorkommt. Hier wird ein Fehlercode gesetzt und einige Felder des zugehörigen DCB für READ zurückgesetzt, damit bei der nächsten Benutzung des DCB keine Komplikationen auftreten und anschließend das Programm be-

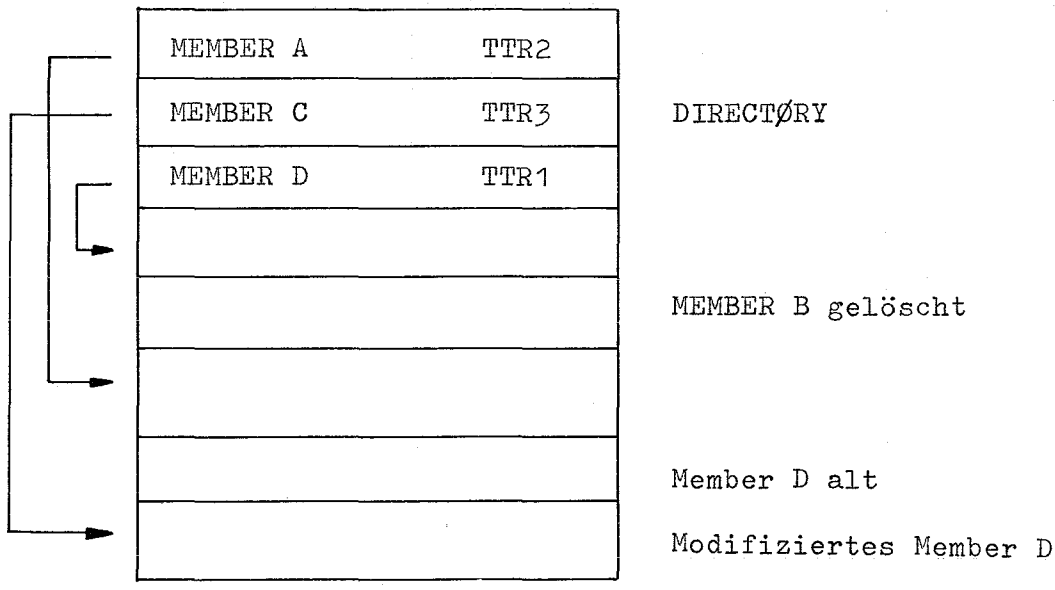
FALSCH endet. Diese Routine wird aktiv, wenn ein Fehler bei WRITE vor-

STOW kommt. Hier wird wie bei SYNAD gehandelt. Durch Macro STOW wird für das neue Member ein Eintrag in das Inhaltsverzeichnis vorgenommen.

ENDE Hier werden zwei 8-Byte Felder, deren Adressen WADR1 und WADR2 sind und in TCPOPROC (STEFAN) durch GETMAIN reserviert wurden, mit Macro FREEMAIN freigegeben. Das Macro HE beendet das Programm (←ATER).

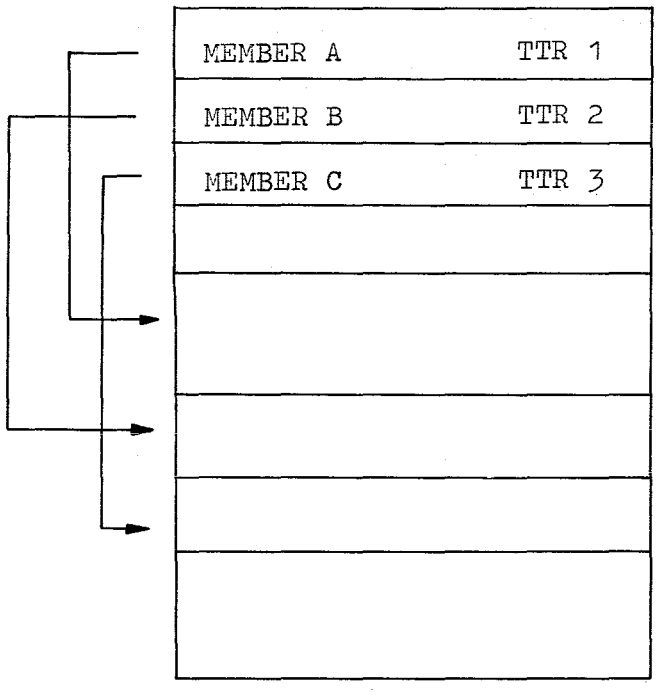
15. COMPRESS (H. Santo)

Das Programm wird benützt innerhalb des TCP zur Reorganisation der "partitioned" organisierten Datensätze der Benutzer. MODIFIZIEREN mit Parameterangabe KONTEXT oder ZEILE schreiben die ausgewählten Member nicht mehr auf denselben physikalischen Platz zurück, sondern fortlaufend an das Member mit der höchsten Spuradresse innerhalb des Datensatzes des Benutzers.



Die Membernamen im Inhaltsverzeichnis (DIRECTORY) des Datensatzes sind alphabetisch geordnet. Die eigentlichen "Member" werden in zeitlicher **Reihenfolge** nach aufsteigender Spuradresse in den Datensatz des Benutzers eingeschrieben.
Der Befehl LÖSCHE streicht den Eintrag im Inhaltsverzeichnis, schiebt das Inhaltsverzeichnis zusammen, hat jedoch keine Wirkung auf den Bibliotheksinhalt. Es entsteht eine "Lücke" im Datensatz des Benutzers. Hat ein Benutzer oft die Befehle LOESCHE bzw. MODIFIZIERE angewendet, so kann trotz geringer relevanter Information in seiner Bibliothek (Bibliotheksgröße ca 3 Cyl.), seine Bibl. voll sein.
Der Befehl 'Comprimiere' soll innerhalb des TCP eine Reorganisation des Datensatzes durchführen.
Angestrebt wird eine automatische Reorganisation, sobald beim Schreiben auf den Datensatz festgestellt wird, daß die Bibliothek voll ist. Durch Setzen eines Funktionsbits in der Linebox, welches das Programm TCPOPRØC nach jedem Funktionsaufruf testet, kann dann automatisch das Reorganisationsprogramm von TCPOPRØC aufgerufen werden.

Bibliothek nach der Reorganisation. Schema



Die Member sind in aufsteigenden Spuradressen und in alphabetischer Reihenfolge in der Benutzerbibliothek angeordnet.
Member in denen I/O-Error auftraten, werden nur teilweise, d.h. bis zum Feststellen des I/O-Errors auf die neue Bibliothek übertragen. Tritt der I/O-Error im 1. gelesenen Block auf, so wird der ganze Member gelöscht, die übrigen Member weiter bearbeitet. Es erfolgt eine Benachrichtigung des Benutzers durch Herausschreiben der Namen der Member in welchen I/O-Fehler aufgetreten waren.
Zusätzlich werden fehlerhafte Membernamen (z.B. Membernamen bestehend aus Hexadezimalen Nullen) gelöscht. Alle Fehlerausgänge innerhalb des Programmes sind so gesetzt, daß bei nicht reparablen Fehlerbedingungen der Urzustand der Bibliothek wieder hergestellt wird.
Das Programm arbeitet mit der Zugriffsmethode BPAM. Eine existierende Variation des Programmes (Programm MOVE), die für feste Blocklängen verwendbar ist, und ein MOVE inplace ohne zusätzliche Allocation durchführt, ließe sich unter Verwendung von XDAP auch für variable Blocklängen erweitern. Das Programm sortiert die Member in aufsteigenden Spuradressen und schreibt die Member in der sortierten Reihenfolge in die Bibliothek zurück.

PROGRAMMBESCHREIBUNG

Aufruf:
Aufruf des Programmes über das Macro ATTACH vom Modul TCPOPRØC der Controlsection PROC bei Eingabe des Befehls
COMPRESS
am Terminal.
Parameter bei der Befehlseingabe werden keine verlangt, da Angaben innerhalb der LINEBØX , wie die Benutzerindentifikation und der Volume-**Serial** -Name festliegen und damit auch der Datensatzname des"partioned" organisierten Benutzerdatensatzes und der Name des 2314 Plattenspeichers.

Verwendete Parameter aus der LINEBØX bzw. KØNBØX:

LINEBØX	VØLSER	: 2314-Plattenspeicher z.Zt: TCP001 bzw. TCP002
	USERID	: Benutzeridentifikation
	PAREØDAD	: Programmadresse bei der bei Auftreten von Fehlerbedingungen des Macros READ innerhalb des Moduls COMPRESS die Bearbeitung fortgesetzt wird.
	PARSYNAD	: Bei SYNAD-Fehlern des Macros READ wird bei dieser Adresse weiter gearbeitet.
	LCC12	: enthält nach Programmende X'0000' oder Fehlercodes
	BUFADS	: Adresse des 80-Byte-Puffers, der bei Auftreten von I/O Fehlern vom Modul COMPRESS angefordert wird. Der Puffer enthält die Namen der Member bei welchen I/O Fehler beim Lesen (Macro: READ) festgestellt wurden.
KØNBØX	DDCB	Adresse des DCB (dynamisch)
	Register:	
	R3	enthält die Adresse der aktuellen Linebox
	R11	enthält beim Aufruf des Moduls die Adresse der KØNBØX.

PROGRAMMABLAUF

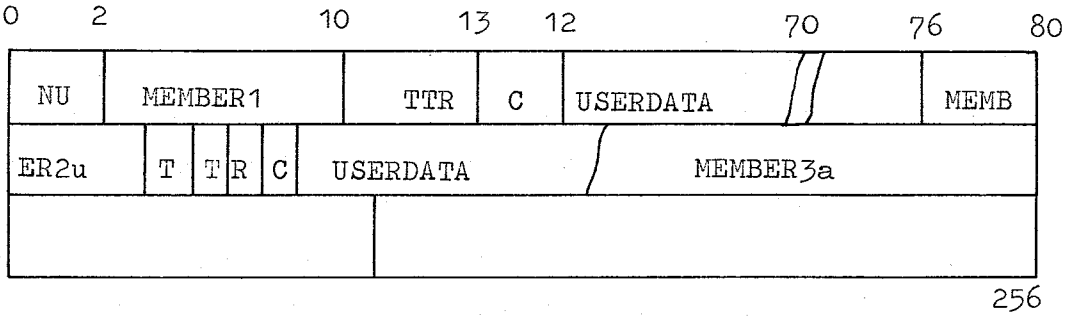
Compress

Beim Aufruf des Programmes Compress steht in Register R3 die Adresse der aktuellen Linebox und in Register R11 die Konboxadresse. SYNAD- und EODAD-Adressen werden in den PARTDCB eingetragen und in LCC12 der Linebox der Fehlercode X'FFFE' gespeichert, der immer dann unverändert bleibt, wenn das Programm Compress abnormal zu Ende geht, ohne Fehlerbedingungen analysieren zu können. Es wird eine Liste (Volume-Liste) folgender Struktur aufgebaut

VOLLIS	X'0001'	Anzahl der Volumes
	X'30C02008'	Einheit 2314
	X'TCP001'	bzw. C'TCP002' aus LINEBØX (VØLSER)
	X'0000'	

ØBTAIN Der DSCB wird eingelesen. Wird der Returncode des Macros ≠ 0, so werden die Fehlercodes weiter untersucht → ERRØBT. Die Listenform für das Macro EXTRACT wird im Arbeitsbereich des Programmes aufgebaut.

ALLØCAT Der Ausgabeparameter des Macros EXTRACT bestimmt die Adresse der Task- "Input-Output" - Tabelle. In dieser Tabelle sucht man den Eintrag für den folgenden DD-Namen:
Konvention bei TCP:
DD-NAME DD VOLSER
VOLSER ist der Inhalt des Parameters VOLSER der LINEBØX;
Wurde der entsprechende Eintrag gefunden, so wird aus ihm die UCB-Adresse entnommen und eine UCB-Liste aufgebaut. Der aufgebaute "Job-File-Control-Block" enthält nachfolgende Einträge:
DD NAMEN
Bezeichnung der Datensatzorganisation
Maximale Blocklänge
Anzahl der Kanalprogramme
Logische Satzlänge
Art der Anforderung an Speicher (Zylinder)
Anzahl der Zylinder (entnommen aus DSCB)
Größe des Inhaltsverzeichnisses (Anzahl Spuren).
Die beiden Eingabeparameter für den SVC 32 sind: in Register 1 die Adresse der UCB-Liste und in Register 0 die Adresse des selbst gebauten "Job-File-Control-Blockes."
Der SVC 32 generiert eine Bibliothek (TCP.XXX.SAVE) gleicher Größe wie die der alten Bibliothek mit dem Namen TCP.XXX.SAVE.
Ist der Returncode ≠ 0 → NØALLØCA.
Der neu generierte Datensatz (TCP.XXX.SAVE) wird geöffnet (Open INPUT Type = J) (OPEN-Fehler → NØTØPEN) das Inhaltsverzeichnis des Datensatzes TCP.XXX.USER blockweise gelesen. Die Blöcke des Inhaltsverzeichnisesses haben folgendes Aussehen.



NU = Anzahl der Byte im Block

READNEU Der Membername wird dem eingelesenen Block des Inhaltsverzeichnisses entnommen. Membernamen die aus Blanks bestehen, werden übergangen. Die übrigen Member werden blockweise auf den neu generierten Datensatz kopiert.

EODADADR Ist das Member vollständig auf den Datensatz übertragen, so wird ein Eintrag in das Inhaltsverzeichnis vorgenommen. Der Eintrag enthält außerdem Membernamen und den Benutzerdaten die neu relative Spuradresse. Sind alle Member, deren Namen in einem Block enthalten sind, auf den neuen Datensatz übertragen und im Inhaltsverzeichnis vermerkt, so wird ein neuer Block des Inhaltsverzeichnisses eingelesen. Der letzte Member ist bearbeitet. Die alte Bibliothek kann gelöscht werden.

ENDLØ (Fehler SCRATCH CATØLD)

UMBEN Die neu generierte Bibliothek wird umbenannt TCP.XXX.SAVE in TCP.XXX. USER.

CAT und katalogisiert.

CLØSEEX Ist das Programm ohne Fehlerbedingungen zu erkennen, abgelaufen, so wird der Parameter LCC12 der Linebox mit H'0' überschrieben und die geöffneten Datensätze geschlossen.

IØERRØR Wurden I/O-Fehler erkannt (beim Lesen bzw. Schreiben eines Members), so wird ein 80 Byte - Puffer aus dem dynamisch organisierten Pufferbereich angefordert.

HØLE

CLØSE
BERENDAD

Zusätzlich wird ein Fehlercode in LCC12, der mit X'06' beginnt gespeichert, um bei Programmende die Anzahl der Zeichen im angeforderten Ausgabepuffer berechnen zu können und im 1. Byte von RECBUFAD (Linebox) zu speichern. Die Behandlung der Fehlerbedingungen geht aus dem nachfolgenden Schema und dem Strukturdiagramm hervor.

Fehlerfälle - Ursachen	Returncode (LCC12)	Aktion	Zustand der Bibliothek
bei Close PARTDCØ	X'FFFE'		Bibl. im Urzustand catalogisiert
beim Entkatalogisieren			
I/O-ERRØR	H'129'	CLØSE	"
übrige Fehler	X'FF00'	CAT	"
Fehler bei OBTAIN	H'119'	IOERRØR CAT	"
Fehler DD-NAME in der Prozedur nicht gefunden	X'FF01'	CAT	"
Fehler ALLØCATIØN			
perm.I/O-ERRØR Returncode 18	H'131'	CAT	"
Returncode = 4 Dupl.Name DSCB	X'FF00'	CAT lösche TCP.xxx.SAVE	"
kein Platz in VTØC	H'130'	CAT	"
kein Platz für Alloc.	H'132'	CAT	"
Open auf TCP.xxx.SAVE	X'FF00'	ENDLØ (TCP.xxx.SAVE) CAT (TCP.xxx.USER)	"
SYNAD-Fehler lesen Directory	X'FF01'	ENDLØ (TCP.xxx.SAVE) CAT	"
EODAD-Fehler lesen Directory	H'105'	" CAT	"

Ursachen	Returncode in LCC12			Zustand der Bibliothek
Fehler FIND I/O-ERRØR	H'602'			
übrige Fehler	X'FF01'	ENDLØ (TCP.xxx.SAVE)	CAT	Bibl. im Urzustand
I/O-ERRØR bei READ Member				
α) vor 1. WRITE (Member) bei weniger als 9 fehlerhaft erkannten Member	H'602'	Mit der Bearb.bei nächstem Member beginnen vorher jedoch FLAG-Bits löschen. Membernamen in Ausgabe-Puffer übertragen		Bibl. komprimiert
ß) nach 1. Write	H'602'	Membernamen in neues Inhaltsverzeichnis und in Fehlerpuffer eintragen		"
Write I/O-Fehler	H'108'	CAT (alte Bibl.) CLØSE DCB's	ENDLO (TCP.xxx.SAVE)	Bibl. im Urzustand
STØW I/O-Fehler wie WRITE I/O-Fehler behandeln				
SCRATCH (I/O-ERRØR)	H'133'			
UMBEN NØT	H'134'			alter Zustand
CAT I/O-Fehler je nach vorhergegangenen Fehlern				

VII. TCP-Jobmanagement

1. Möglichkeiten (F. Sell)

Der TCP-Benutzer hat zwei Möglichkeiten, Jobs bearbeiten zu lassen:

- Einlesen, Starten, Ausgabe normal über die Ein- und Ausgabeschlange des OS
- als Blitzstart-Modul innerhalb des TCP.

2. Bearbeitung durch das OS (F.Sell)

Für die Bearbeitung der Jobs durch das OS wirken Einlese- und Ausgabeprozeduren mit dem TCP zusammen.

2.1. Einleseprozedur TCPRDR

Das Einlesen eines Jobs initialisiert der Benutzer vom Terminal aus durch das Kommando: ST, Name.
Name ist dabei der Name eines Moduls in der benutzereigenen Bibliothek, der einen vollständigen Job, d.h. Steuerkarten plus Programm enthalten muß. Dieser Modul muß außerdem mit dem Prädikat Job-Control-Language (JCL) im Dateiverzeichnis (Directory) versehen sein. Das Start-Kommando ruft eine Einleseprozedur TCPRDR auf. TCPRDR ist ein normaler READER des OS, der beim Aufruf mit einigen aktuellen Parametern versehen wird, z.B. **Volume-serial** number der Platte, auf dem sich die Bibliothek des Benutzers befindet. Außerdem sind einige Parameter fest angegeben, die sich auf die durch den Job produzierte Ausgabe beziehen:

- auf welche Platte die Ausgabe-Datensätze gespeichert werden sollen,
- wieviel Platz für die Ausgabe-Datensätze als Default-Wert zur Verfügung gestellt wird.

Der TCPRDR liest den angegebenen Modul und **reiht** ihn in die Eingabe-Schlange ein. Das Starten und die Ausführung des Jobs geschieht dann durch das Job-Management des OS.
Die Beendigung des Jobs wird der Ausgabeprozedur TCPWTR gemeldet, wenn Ausgabe für das Terminal verlangt und produziert worden ist.

2.2. Ausgabeprozedur TCPWTR

Die Ausgabeprozedur TCPWTR hat folgenden Zweck:
Hat ein Job Ausgabe-Datensätze erzeugt, die sich der Benutzer am Terminal ausgeben lassen möchte, (entweder über das Druckwerk oder über das Sichtgerät), so werden diese aus der allgemeinen Ausgabe-Schlange weggenommen, umbenannt und katalogisiert. Auf diese Art können sie dann über Standard-TCP-Kommandos weiter behandelt werden (ausgeben, löschen). Über die Aktivitäten des TCPWTR wird der Benutzer über das Terminal benachrichtigt (siehe III./2.2.2.).

2.2.1. Aufbau des TCPWTR

Der TCPWTR besteht im einzelnen aus den Moduln TCPWTR und TCPWTRA. Außerdem werden jeweils die Routinen TCP1LIST und die IBM-Routinen IEFQMDQQ, IEFQMRAW, IEFQMUNC, IEFQDELE und IEFQMNQ2 dazugefügt.

2.2.2. Arbeitsweise

Der TCPWTR ist als eine Systemtask organisiert; man kann ihn somit von der Konsole aus mit dem Startkommando S TCPWTR aufrufen.

2.2.2.1. Startmodul TCPWTR

Durch den Aufruf S TCPWTR wird die Kontrolle an den Start-Modul TCPWTR gegeben. Da sich der TCPWTR auf TCP-Dienste abstützt, wird überprüft, ob TCP aktiv ist. Ist das nicht der Fall, wird der TCPWTR sofort mit einem Hinweis an den Operateur beendet.

Ist das TCP aktiv, so wird die Kontrolle mit LINK an den eigentlichen TCPWTR-Modul TCPWTRA übergeben.

2.2.2.2. Modul TCPWTRA

Der Modul TCPWTRA beginnt mit einer Initialisierungsroutine. Nach Retten der allgemeinen Register und Setzen des Basisregisters wird mit dem Makro GETMAIN Arbeitsspeicher für folgende Listen besorgt: Command-Scheduling-Control-Block (CSCB), Queue-Manager-Area (QMPA), Event-Control-Block-Liste (ECB-Liste), Data-Set-Block/System-Message-Block-Buffer (DSB/SMB-Buffer), Save-Area. Die Länge der einzelnen Listen wird durch Dummy-Sections bestimmt. Die entsprechenden Register werden gesetzt. Nachdem der Platz für den Command-Input-Buffer (CIB) mit dem Start-Kommando freigegeben wurde, wird überprüft, auf welche Ausgabe-Klasse (Sysout-class) der TCPWTR arbeiten soll. Danach werden die ECBs normiert und die Listen für die Makros CATALOG, RENAME, LOCATE, OBTAIN und UNCATLG gefüllt.
Wird bei den Überprüfungen ein Fehler festgestellt, so wird die Kontrolle an die Benachrichtigungsroutine für den Operateur weitergegeben. Bei fehlerfreiem Ablauf wird die Warte-Routine (WAIT-Routine) erreicht.

2.2.2.2.2. Warteroutine

Die Warteroutine hat die Aufgabe, bis zur Beendigung eines Jobs mit einer Ausgabe für die angegebene TCP-Klasse in einem Wartezustand zu bleiben, dann jedoch die weitere Bearbeitung anzustoßen. Beim Eintritt in die Warte-Routine wird sofort untersucht, ob bereits Ausgabe-Datensätze vorhanden sind. Ist das nicht der Fall, wird mit dem Makro WAIT auf das Eintreffen des Ereignisses "Ausgabe vorhanden" gewartet. Trifft das Ereignis ein, so wird sofort untersucht, ob auch das TCP selbst aktiv ist, um seine Dienste auch durch den TCPWTR ansprechen zu können (siehe III/2.2.2.); ist TCP nicht aktiv, wird auch der TCPWTR gestoppt, damit die Ausgabe-Datensätze nicht undokumentiert umbenannt werden.

Mit der IBM-Routine IEFQMDQQ (Queue-Manager-Dequeue-Routine) wird die Ausgabe mit der höchsten Priorität von der Ausgabe-Schlange der angegebenen Klasse weggenommen und der dazugehörige Jobname in die QMPA gebracht. Zur weiteren Bearbeitung wird die Kontrolle an die Lese-Schreibe-Routine gegeben (READ- and WRITE-Routine).
Eine weitere Aufgabe der Warte-Routine ist es, das Stopp-Kommando des Operateurs für den TCPWTR zu erkennen (P TCPWTR); danach wird die Kontrolle an die Benachrichtigungsroutine weitergegeben.

2.2.2.2.3. Lese-Schreibe-Routine (READ- and WRITE-Routine)

Von der Routine IEFQMDQQ wurden Angaben über den Job in die QMPA gebracht. Aus der QMPA, Feld QMNAM wird aus dem Jobnamen die Rechenzentrums-Benutzernummer des Jobeigners ermittelt. Über die Routine TCP1LIST wird aus der Benutzernummer die TCP-Identifikation des Benutzers (3 Zeichen) ermittelt. Ist die Benutzernummer nicht für das Arbeiten mit TCP zugelassen (Kennung = CCC), so wird der Ausgabe-Datensatz auf die Klasse A umgesetzt, an die entsprechende A-Ausgabeschlange gekettet und von einem normalen OS-Output-Writer am Schnelldrucker ausgegeben.
Nun wird untersucht, was im DSB/SMB-Buffer steht; ob es ein DSB oder ein SMB oder ob der Buffer leer ist. Dieser Buffer wurde von der Routine IEFQMDQQ zum ersten Mal gefüllt. Für die weiteren Füllungen wird die IBM-Routine IEFQRAW (Queue-Manager-Read-and Write-Routine) benutzt. Die zu einem Job gehörende Information über die Ausgabe steht in einer Schlange von DSB/SMBs. Wurde bei der Job-Definition als Message-Class (MSGCLASS) die TCP-Klasse angegeben, so werden die SMBs selbst in die Ausgabeschlange für den Job geschrieben. SMBs beinhaltet z.B.: die Job-Control-Language (JCL), Hinweise auf Allocation von Datensätzen, Verbleib von Datensätzen nach Step- und Job-Ende und Accounting-Information. Für die Ausgabe-Datensätze der TCP-Klasse stehen nur ihre Namen mit Hinweisen auf ihren Speicherplatz in der Ausgabeschlange.

TCPWALL Wurde der erste SMB gefunden, so muß ein Datensatz eingerichtet werden, in den die SMBs geschrieben werden können; dies geschieht in einer Datensatz-Einrichtungsroutine (Writer Allocator). Über

FINDDD ein DD-Statement SYSMSG der Job-Control-Information des TCPWTR wird ermittelt, auf welcher Speichereinheit wieviel Platz für den Datensatz reserviert werden soll. Da der Name des einzurichtenden Datensatzes nach festen Regeln gebaut wird, (TCP.XXX.#MSGYYCT: XXX = TCP-Identifikation des Benutzers, YY = letzte beiden Zeichen des Jobnamens, CT = Zähler von 00 - 09), bei dem zur Vermeidung von gleichen Namen die beiden letzten Zeichen von 00 - 09 hochgezählt werden, muß erst mit dem Makro LOCATE überprüft werden,

TRYLOC welches der erste freie Datensatzname ist. Ist kein Name mehr frei, wird der Benutzer davon unterrichtet, daß aus Speicher-

ALLOCATE gründen seine Ausgabe der Systemnachrichten vernichtet wird. Im Normalfall wird über die IBM-Routine IEFQMRAW (Queue-Manager-Read- and Write-Routine) und dem SVC 32 (Direct Access Device Storage Allocation) der gewünschte Datensatz eingerichtet. Nachdem er mit dem ermittelten Namen mit dem Makro CATALOG katalogisiert ist und mit OPEN für die Ausgabe eines sequentiellen Datensatzes eröffnet ist, wird die Kontrolle an die Lese-Schreibe-Routine zurückgegeben Folgende mögliche Fehler werden behandelt:

CATERR - Bei der Katalogisierung: der Datensatz wird nicht katalogisiert, der Benutzer wird informiert, unter welchem Namen auf welchem Speicher der Datensatz abgelegt ist:

NOLOC - Beim Fehlen eines freien Namens: die Ausgabe wird vernichtet, der Benutzer wird informiert.

DADSMERR - Bei der Zuteilung von Speicherplatz: die Ausgabe erfolgt über die Klasse A, der Benutzer wird darüber informiert.

ABORT - Beim Suchen des DD-Statements: kein DD-Statement SYSMSG in der Job-Control-Information vorhanden, TCPWTR wird mit einer Nachricht an den Operateur beendet.

SMBPROC Die weitere Bearbeitung des SMB geschieht in der Lese-Schreibe-Routine. Der Text der System-Message wird je nach Format expandiert und über das Makro PUT auf den eingerichteten Datensatz geschrieben.

TSTINPUT
DSBPROC
LOCAT1
RENAM

Danach wird aus der Ausgabe-Schlange der nächste SMB oder DSB geholt. Ist es ein DSB, so wird über den Datensatznamen mit der IBM-Routine IEFQMRAW der Job-File-Control-Block (JFCB) eingelesen. Da der Ausgabe-Datensatz nach festen Regeln umbenannt werden soll, (TCP.XXX#OUTYYCT: XXX = TCP-Indentifikation des Benutzers, YY = letzten beiden Zeichen des Jobnamens, CT = Zähler von 00 - 09), wird mit dem Makro LOCATE der erste freie Name gesucht und mit dem Makro RENAME der Ausgabe-Datensatz umbenannt. Mit dem Makro CATALOG wird der Datensatz dann katalogisiert, damit er durch TCP-Kommandos ansprechbar ist.

Beim Suchen nach einem freien Namen wird mit dem Makro OBTAIN untersucht, ob sich der Datensatz mit diesem Namen wirklich noch auf der Platte befindet, oder ob er durch SCRATCH (= Löschen der Platte) weggenommen wurde. Ist das der Fall, so wird der Name mit dem Makro CATALOG aus dem Katalog genommen; er ist dann zur weiteren Verwendung verfügbar.

Folgende mögliche Fehler werden behandelt:

NCATOUT

- Bei Systemfehler der Katalogisierung: der Datensatz wird nicht katalogisiert, der Benutzer wird informiert, unter welchem Namen auf welchem Speicher der Datensatz abgelegt ist.
- Systemfehler bei RENAME: der Datensatz wird nicht umbenannt, an den Operateur wird ein Hinweis mit dem Returncode ausgegeben.
- Kein freier Name mehr vorhanden: der Datensatz wird nicht umbenannt und nicht katalogisiert; dem Benutzer wird neben einem Hinweis auf zuviele nicht abgeholte Ausgabe-Datensätze der Name und Speicherplatz des Datensatzes mitgeteilt.

2.2.2.2.4. Job-Ende-Routine

TCPWEOJ

Nach Abarbeitung der SMB und DSB eines Jobs werden von der Job-Ende-Routine Aufräumungsarbeiten durchgeführt. Wurde ein Datensatz für die Systemnachrichten eröffnet, so wird er nun geschlossen und der Schreibbuffer zurückgegeben.

Der Benutzer wird vom Ende seines Jobs über das Terminal verständigt. Da die SMBs und DSBs, die zu diesem Job gehören, abgearbeitet sind, wird der Hinweis auf diese SMB/DSB-Schlange durch die IBM-Routine IEFQDELE (Queue-Manager-Delete-Routine) gelöscht. Ist im bisherigen Ablauf kein Fehlerflag benutzt worden, so wird die Kontrolle zur Bearbeitung des nächsten Jobs an die Warte-Routine zurückgegeben, andernfalls wird die Fehlerbearbeitungsroutine angesprungen.
In der Job-Ende-Routine befindet sich noch die IBM-Routine IEQMNQ2 (Queue-Manager-Enqueue-Routine), die die auf die Klasse A umgesetzten Ausgabe-Datensätze wieder in die Ausgabe-Kette einreiht.

2.2.2.2.5. Fehlerbearbeitungsroutine

TCPWTO Die Fehlerbehandlung wird durch die zentrale Fehlerbearbeitungsroutine vorgenommen. Ihre Eingangsparameter sind Fehlerflags. Die Ausgabe besteht aus Nachrichten an die Operateurkonsole. Folgende Fälle werden bearbeitet:

-Ein-Ausgabefehler bei einer Schreiboperation,
-Ein-Ausgabefehler bei der Bearbeitung der System-Job-Queue,
-Beendigung des TCPWTR durch den Operateur,
-Fehlender oder falscher Parameter in der TCPWTR-Prozedur für die zu bearbeitende Ausgabeklasse,
-Fehlende DD-Karte in der TCPWTR-Prozedur für die Definition des System-Nachrichten-Speichers,
-Systemfehler; bei diesem wird näher aufgeschlüsselt, mit welchem Returncode die Operationen "Allocation", "RENAME", und "CATALOG" zu Ende gingen.

In den Fällen einer unnormalen Beendigung des TCPWTR wird die SMB/DSB-Schlange für diesen Job durch die IBM-Routine IEFQMUNC (Queue-Manager-Unchain-Routine)vom System weggenommen. (Die Ausgabe für den gerade bearbeiteten Job ist nicht vollständig!). Die Kontrolle wird über den Startmodul TCPWTR an das OS zurückgegeben.

3. Programm für das Starten von Benutzerprogrammen (V. Haase)
(Übergabe an den Stapelbetrieb des OS)

Modul TCP1STA

TCP1STA initialisiert einen READER, der Job Control Language aus einer Benutzerbibliothek in die SYS1.JOBQUE bringt. (Zugehöriges Kommando: STARTE, Membername)

Nach der Initialisierung wird abgefragt, ob der TCPRDR

TRY schon aktiv ist; wenn ja, ist Starten nicht möglich (Code 0A27).

OK Mit BLDL wird die Existenz des Members, das die JCL enthält, geprüft; in der BLDL-Liste wird überprüft,

WEITER ob die Art = JCL ist (Sonst Fehler = 113).

MOVECOM Der Operator-Befehl
S TCPRDR, 2314, Volume-Serial-Nr., DISP = SHR,
DCB = (BLKSIZE = 1600, BUFL = 1600, RECFM = FB, LRECL = 80),
DSN = TCP.Userid.USER (Member)'
wird aufgebaut, d.h. die unterstrichenen Größen eingetragen.

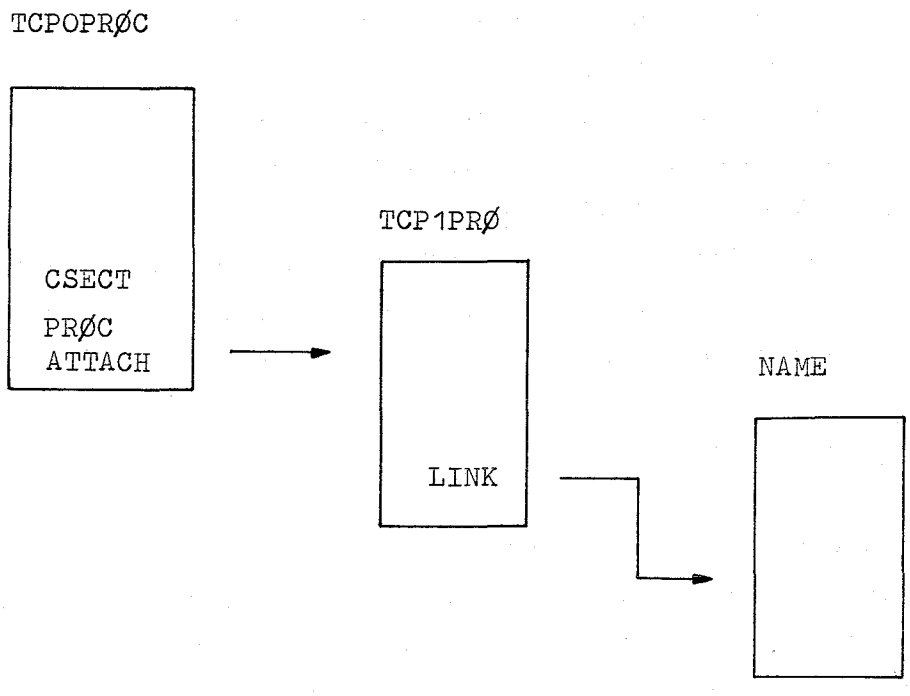
LAST Mit einem SVC wird der Befehl dem Entschlüßler für Operator-Kommandos übergeben.

4. Blitzstart (H. Santo)

Dem Terminalbenutzer wird die Möglichkeit gegeben unter Umgehung der Eingangswarteschlange der DVA die Ausführung eines speziellen Programmes zu fordern. Bei diesen Programmen kann es sich um benutzerspezifische zeitkritische Auswertungen eines Meßergebnisses handeln oder um oft gebrauchte Programme (Funktionen) eines speziellen Anwenderkreises.
Der Befehlsvorrat des TCP wird durch folgenden Befehl erweitert.

BLITZSTART, NAME = ..., PARAMETER = ...

Diese Befehlsgruppe umfaßt eine Menge von Funktionen im üblichen Sinne, d.h. die Ausführungen **verschiedenartigster** Programme wird durch einen Befehl verlangt, wobei hier erst der 2.Parameter das spezielle Funktionsprogramm kennzeichnet. Das wird auf folgende Weise erreicht:
Die CSECT PRØC des Lademoduls TCPOPRØC initialisiert das Programm TCP1SPRØ (MACRO: ATTACH), das wiederum seinerseits über den vom Benutzer angegebenen Namen (NAME = ..) den Lademodul (Member der TCP.LINKLIB) desselben Namens auswählt und ihm die eventuell angegebenen Parameter in einer Liste übergibt, deren Adresse in Register 1 steht.



Die Länge des Parameterfeldes darf 42 Byte nicht überschreiten. Innerhalb des Parameterfeldes gibt das 1. Halbwort die Zahl der nachfolgenden Byte an.

TCP1SPRØ

TCP1SPRØ wird von der Controlsection PRØC des Moduls TCP1BASE initialisiert (Macro: ATTACH)

FUNKTION

TCP1SPRØ sucht die vom Terminalbenutzer angeforderte Blitzstartroutine auf der Bibliothek (TCP.LINKLIB) und übergibt vom Benutzer spezifizierte Parameter den aufzurufenden Blitzstartmodul.

EINGANGSPARAMETER

Register 3 enthält die Adresse der aktuellen Linebox. Register 11 zeigt zur KONBOX.

Parameter aus der Linebox:

LAMENBN: Name des Blitzstartmoduls

WCØDE2 X'01' bedeutet, daß keine Eingabeparameter für den Blitzstartmodul
WADR2 zu übergeben sind. Wenn WCØDE2 = X'01' ist, enthält WADR2 die Adresse des Parameterfeldes mit den notwendigen Parametern zur Ablaufsteuerung des Blitzstartmoduls.

Parameter aus der KONBOX:

KBXLIB Adresse des DCB's der die TCP.LINKLIB beschreibt.

AUSGANGSPARAMETER

LINEBØX: LCC12 = 0 oder Fehlercodes
Register enthält Adresse des Parameterfeldes der Steuerparameter für den Blitzstartmodul

PROGRAMMABLAUF

SCØADMØD Nach Programminitialisierung wird der Parameter LCC12 mit dem Fehlercode X'FFFE' überschrieben. Mit Hilfe des Macros BLDL und vorher aufgebauter Liste wird im Inhaltsverzeichnis der Bibliothek TCP.LINKLIB nach einem Modul mit dem Namen, welcher in LAMEMBN spezifiziert wurde, gesucht.

NØNAME	Der gesuchte Modul ist nicht in der TCP.LINKLIB vorhanden. Parameter LCC12 bekommt den Wert X'10'.
ENDE	Rückkehr zum TCP1BASE.
GO1	Der vom Benutzer spezifizierte Modul wurde in der Bibliothek TCP.LINKLIB gefunden. Sind Parameter an das aufzurufende Programm zu übergeben, so wird eine Parameterliste (CALLIST) aufgebaut und Register 1 enthält (Macro LINK) die Adresse des Parameterfeldes.
NØPARAME	Es sind keine Parameter für die Steuerung des Funktionsprogrammes vorhanden oder notwendig. Aufruf des Blitzstartmoduls mit Hilfe
IØERRØR	des Macros LINK → ENDE. Beim Suchen im Inhaltsverzeichnis der Bibliothek TCP.LINKLIB sind I/O-Fehler aufgetreten. Der Fehlercode in LCC12 wird auf 110 gesetzt → ENDE.

VIII. TCP - Graphik

1. D K P (W. Strolz)

CSECT	TCP1DKP
ENTRY	via XCTL von TCP1DSUP

EINGANGSPARAMETER

R1 zeigt auf die Adresse einer Parameterliste.
Diese Parameterliste enthält
- Länge der Liste
- Konbox Adresse

EXIT	RETURN zum OS, wenn SHUTDOWN eingeleitet wird
COMMUNICATION mit TCPOPRØC	WAIT auf DISPECB POST auf DISPECB2

FUNKTION

DKP steuert den graphischen Teil von TCP. Es entschlüsselt den Typ des Display-Kommandos und delegiert die Befehlsausführung an die entsprechenden Routinen.

Da die Routinen zur graphischen Display-Ausgabe in PL1 geschrieben sind, und der PL1-Compiler keinen Reentrant Code erzeugt, muß DKP für eine Sequentialisierung der Display-Aufbereitungs- und Ausgabe Routinen sorgen. DKP stellt ein 16K-Byte-Ausgabefeld zur Verfügung, das als Lese-,Arbeits- und Ausgabepuffer dient.

PROGRAMMABLAUF

TCP1DKP DKP wird von TCP1DSUP mit XCTL aufgerufen und erhält dabei über Reg. 1 die Konbox-Adresse.
Die Adresse der Display-Ausgabe-Area wird in die Konbox eingetragen und anschließend BLDL für die Display Modules gemacht.

BLDLERR Ist BLDL nicht erfolgreich, dann wird das der Task TCPOPROC

WAIT mitgeteilt (POST DISPECB2 mit Completion Code1). In jedem Fall wird anschließend auf einen Display-Auftrag von TCPOPROC her gewartet (WAIT DISPECB). Ist schließlich DISPECB gepostet, dann wird zunächst dieser ECB auf Null gesetzt und die aktuelle Linebox-Adresse in die Parameterliste eingetragen. Ist das LBX-Reg.

RETURN Null, dann ist SHUTDOWN im Gange und DKP macht RETURN zum OS. Andernfalls beginnt die Befehlsanalyse. Bei den Kommandos DZ, DT und DA wird sofort nach TEXT gesprungen. Bei den Kommandos DO, D+ und D- erfolgt Sprung nach DO; bei DE und DD nach MODUSTST. Liegt keines dieser Kommandos vor, dann ist ein Fehler aufgetreten und

POSTROUT über ERROR1 und POSTROUT wird TCPOPROC mittels POST DISPECB2 benachrichtigt. Anschließend Sprung nach WAIT.

MODUSTST Hier wird untersucht ob einer der folgenden Modi vorliegt (WCODE2 $\geq$ 04):

A - Automatisch
AM - Automatisch mit Maßstab
G - Gesteuert
GM - Gesteuert mit Maßstab
K - Konstruiere oder Konstruiere direkt.

In diesen Fällen erfolgt Sprung nach GRAPH; sonst wird geprüft ob der Modus P (Plot, WCODE2=02) vorliegt. Ist dies der Fall, erfolgt Sprung nach PLOT, andernfalls (Text, WCODE2 = 01 oder Show,

TEXT WCODE2 = 03) wird nach TEXT gesprungen. Der Modul TCPTAUS wird via LINK angesprungen; das Reg.1 zeigt dabei auf die Adresse der Parameterliste. Ist TCPTAUS beendet, erfolgt ein Sprung nach POST-ROUT wo TCPOPROC von der Ausführung des Display-Kommandos benach-

GRAPH richtigt wird. Der Modul TCPKAUS wird via LINK angesprungen; das Reg. 1 zeigt auf die Adresse der Parameterliste. Ist TCPKAUS nicht erfolgreich gewesen, erfolgt sofort Sprung nach POSTROUT (Fehler-entschlüsselung generell in TCPOPROC). War TCPKAUS erfolgreich, dann wird geprüft ob der Konstruktionsmodus vorliegt (WCODE2 $\geq$ 08)

KONSTR In diesem Fall wird der Modul DKPV2 via LINK aufgerufen und an-

LOGA schließend erfolgt Sprung nach POSTROUT. Soll dagegen der loga-rithmische Modus verwendet werden, (WCODE4 $\geq$ 03) wird der Modul DKPV3 mit LINK angesprungen, anschließend Sprung nach POSTROUT.

V1 Bei allen anderen graphischen Modi (A-WCODE2=04, G-WCODE2=05, AM-WCODE2=06, GM-WCODE2=07) wird der Modul DKPV1 mit LINK an-gesprungen; anschließend ebenfalls Sprung nach POSTROUT.

PLOT Der Modul TCPPAUS wird via LINK angesprungen; das Reg. 1 zeigt auf die Adresse der Parameterliste. Nach Ablauf von TCPPAUS

DO erfolgt Sprung nach POSTROUT. Hier wird der Befehlscode des zu-letzt gegebenen Display Kommandos (DCBCODE und DWCODE) analysiert und die Entsprechenden Display-Aufbereitungs- und Ausgabe-Routinen bei TEXT, GRAPH oder PLOT angesprungen.

2. TCPTAUS (M. Schayegh)

Dieser Module wird nach allen Befehlen, die eine Textausgabe am Sichtgerät verlangen, von dem Module DKP durch LINK aktiviert.

FUNKTION

Bereitstellung von Text aus:
Directory, eigenem Dataset, fremdem PDS, einem sequentiellen Dataset, sowie Bereitstellung von Direkt-Text in einem Speicherbereich, für die Ausgabe auf das Sichtgerät.

EINGANGSPARAMETER

R11 = A (KONBOX)
8 (R11) = A (E/A-Feld)
R3 = A (LINEBOX)
Alle nötigen Informationen werden der LINEBOX entnommen.

AUSGANGSPARAMETER

LCC12 : Fehlercode

PROGRAMMABLAUF

Nach der Initialisierung wird bei dem Befehl DA der Bildschirm gelöscht und das Programm beendet,

TESTM bei DØ zuerst der Bildschirm gelöscht und dann in einer Routine festgestellt, welcher Befehl zuvor ausgeführt worden ist und je

CLI nachdem wird zu der entsprechenden Routine gesprungen. BCODE und WCODE werden in der LINEBOX umgespeichert und bei D+, D-, DZ zu

TEXT den Routinen VOR, ZURUECK bzw. ZUSTAND gesprungen. Hier wird entschieden, ob die Ausgabe aus eigenem Dataset, einem fremden Dataset oder einem sequentiellen Dataset erfolgen soll oder als Direkt-Text. Bei Direkt-Text werden die Zeichen aus der LINEBOX herausgeholt. Zeichenweise mit Positionsangabe ins E/A-Feld umgespeichert, ihre Länge errechnet und zur Ausgabe auf das Sichtgerät an CANCANAS

KLAMMER weitergegeben und schließlich das Programm beendet. Für die Ausgabe aus fremden PDS oder sequentiellen Dataset wird nach PDS bzw. SDS gesprungen; sonst wird versucht, aus eigenem Dataset auszugeben. Bei D+, D- wird nach LR5 gesprungen. Dort werden die Daten ein-

LA gelesen. Sonst wird die angegebene Seiten-Nr. in Binärform umgewandelt, abgespeichert und dann zum Einlesen der Daten übergegangen.

SDS
GOLOC
MVINP
CLIB
SDS1
LAR9
GETSHOW
RECHNE
GET
NOTUSER
FIND
R20
ZUSTAND

Hier wird Vorbereitung getroffen, falls aus SDS oder fremdem PDS ausgegeben werden soll. Durch Makros LOCATE und OBTAIN werden Informationen zum Aufbau und Ergänzen des DCB geholt. Eine Abfrage stellt fest, ob es sich bei dem angegebenen fremden PDS um einen TCP-Benutzer handelt oder einem anderen PDS. Im 2. Fall wird zu der Routine NOTUSER gesprungen. Hier wird der DCB ergänzt, JFCB gelesen und der Dataset eröffnet. Hier wird je nachdem, ob D+, D- oder normaler Befehl ausgeführt werden soll, gleich zur Einleseroutine gesprungen oder zuerst die Seiten-Nr. ermittelt und dann gelesen. Falls es sich um einen sequentiellen Dataset handelt, wird ein neuer DCB für den SDS aufgebaut. Er wird mit einigen Informationen aus der Routine GOLOC ergänzt. Einige Abfragen untersagen die Verarbeitung von Datasets mit LRECL > 1600 Bytes oder RECFM = U. Es wird wiederum, je nachdem ob D+, D- oder ein normaler Befehl gegeben wurde:

1.) die Seiten-Nr. ermittelt
2.) in der Routine RECHNE die Anzahl von Records errechnet, die beim Lesen ignoriert werden müssen bis man zu dem gewünschten Record gelangt, und
3.) in der Routine GET die Records in das E/A-Feld übertragen, übersetzt und auf dem Sichtgerät ausgegeben.

Wenn es sich um Show-Daten handelt, so wird die Übersetzung unterlassen. Hier werden die für Show aufbereiteten, sequentiellen Datasets eingelesen und auf dem Sichtgerät ausgegeben.

Errechnet die zu ignorierenden Daten bis zur gewünschten Seite.

Liest die sequentiell-organisierten Daten.

Hier erfolgt die Ergänzung des DCB für PDS, die nicht von TCP zur Verfügung gestellt worden sind. Stellt die Adresse des gewünschten Members zur Verfügung. Liest das Member, bis zur gewünschten Seite,übersetzt und gibt die Daten aus.

Hier wird das Inhaltsverzeichnis der Benutzerbibliothek ausgegeben.

Zuerst wird die CSECT TCP1ZUS aufgerufen, die die Daten in den Kernspeicher überträgt. Anschließend wird bestimmt, ab welcher Stelle des Speicherbereiches und mit welcher Länge die Daten ausgegeben werden sollen.
Nach der Übersetzung werden die Daten auf das Sichtgerät ausgegeben.

VOR — Hier wird die Seiten-Nr. für D+ aktualisiert.

ZURUECK — Hier wird die Seiten-Nr. für D- aktualisiert.

DØ — Die Seiten-Nr. bleibt unverändert und es erfolgt ein Sprung zu

READ — der Routine, die vor dem Befehl DØ ausgeführt wurde. Das ist die Routine zum Einlesen. Übersetzen und zur Ausgabe aus der eigenen

ENDE1 — TCP-Bibliothek. Hier wird das Programm beendet.

2.1. CANCANAS (G. Fleck)

CANCANAS ist ein Unterprogramm zu TCPTAUS, mit dessen Hilfe ein Kanalprogramm zur Textausgabe gestartet werden kann.

EINGANGSPARAMETER

a) R3 enthält die Adresse der LBX
b) R6 enthält den Löschindikator
c) R11 enthält die Adresse der KBX
d) Adresse des Ausgabefeldes (KBX + 88: KBIO)
e) Ausgabemodus (KBX + 92: KBMOD)
f) Länge des Ausgabefeldes (KBX + 93: KBIOL)

AUSGANGSPARAMETER

LCC12: Fehlercode

PROGRAMMABLAUF

Der Programmablauf von CANCANAS ist ähnlich dem in CANCAN. Auf folgende Unterschiede wird hingewiesen:
1. Die Übergabe der KBX- und LBX-Adresse ist hier einfacher, da das übergeordnete Programm ebenfalls ein Assemblerprogramm ist, und geschieht mit Hilfe von Register R3 und R11.

2. Die Übersetzung des Textes vom EBCDIC in denASCI- Code wird von TCPTAUS durchgeführt und entfällt.

3. TCP1KAUS (M. Schayegh)

Dieser Module wird nach einem Display-Befehl, der eine graphische Ausgabe zur Folge hat, von DKP durch LINK aktiviert.

FUNKTION

Bereitstellung von Daten aus verschieden organisierten Dateien für graphische Ausgabe.

EINGANGSPARAMETER

R1 = A (KONBØX)

R3 = A (LINEBØX)

AUSGANGSPARAMETER

LCC12: Fehlercode

PROGRAMMABLAUF

Nach Initialisierung des DCB und Adressberechnung werden je nachdem, ob der Befehl DØ ausgeführt werden soll oder nicht, verschiedene Felder in der LINEBOX umgespeichert.

AØ
KLAMM
KREUZ
EMEM
BLDL

Wenn Direkteingabe graphisch erfolgen soll, so werden die Daten in das E/A-Feld übertragen und ausgegeben. Wenn aus einem PDS Daten ausgegeben werden sollen, so werden der Membername und Datasetname in geeignete Felder übertragen und nach FPDS gesprungen. Bei der Ausgabe aus dem SDS wird der Datasetname in ein Feld übertragen und nach FSDS gesprungen. Bei der Ausgabe aus eigenem PDS wird in der LINEBOX ein Indikator gesetzt und SYNDD und EODAD-Adresse in dem DCB eingesetzt. Hier wird untersucht, ob der Membername vorhanden ist, ob die Daten des Members numeriert sind und bis zu welcher Spalte verarbeitet werden soll.

FIND Die Adresse des Members wird bereitgestellt.

OK Es wird ein Block gelesen und an den Module DKPUM zur Umwandlung und Ausgabe weitergegeben. Diese Prozedur wird solange wiederholt bis von DKPUM in KONBOX +8 ein Bit gesetzt wird. Damit wird das

FSDS Programm beendet. Für spätere Abfragen wird ein Switch gesetzt.

FPDS Für SDS und PDS wird der DSCB gelesen, daraus ein DCB aufgebaut und ergänzt. Der JFCB wird gelesen und ergänzt und schließlich der

FREST1 Dataset geöffnet. Bei der Ausgabe aus dem SDS wird recordweise gelesen und in das E/A-Feld übertragen, bis maximal 1600 Byte verarbeitet sind. Dann werden die Daten zur Umwandlung und Ausgabe auf den Bildschirm an den Module DKPUM weitergegeben. Dieser Vorgang wird wiederum solange wiederholt, bis vom DKPUM in KONBOX +80 ein Switch gesetzt wird. Damit wird das Programm beendet.

FEND Bei EODAD für PDS wird nach RESTEND gesprungen, für SDS wird die Länge der restlichen Daten bestimmt und falls diese Länge nicht Null ist, ein Switch gesetzt und die Daten zur Umwandlung und Ausgabe an DKPUM weitergegeben. Sonst wird gleich das Programm

DO beendet. Für den Befehl DO werden hier einige Felder in der LINEBOX umgespeichert und anschließend nach AO gesprungen.

END Hier wird das Programm beendet.

3.1. DKPUM (G. Fleck)

Das PL/1 Programm DKPUM wird von TCPKAUS via LINK aufgerufen und gibt die Kontrolle wieder an TCPKAUS zurück.

FUNKTION

DKPUM formt die von TCPKAUS eingelesenen Daten in die rechnerinterne Binary Float-Darstellung um. Für die Daten ist kein festes Format vorgeschrieben. Die einzelnen Zahlen sind lediglich durch ein oder mehrere Blanks voneinander getrennt. Das für die Daten vorgesehene Zahlenfeld kann maximal 4000 Werte aufnehmen. Die für die Bildaufbereitung nötigen Maßstabsangaben werden ebenfalls von DKPUM umgewandelt.

EINGANGSPARAMETER

a) Adresse der KONBOX (siehe Bild 4)
b) Adresse der aktuellen LINEBOX
c) Adresse des Zahlenfeldes (KBX + 9)
d) Kennzeichen für den ersten Aufruf
   (KBX + 12: KBIN = X'00')
e) Länge des eingelesenen Zahlenfeldes
   (KBX + 13: KBLEN)
f) LBX-Codes (LBX + 39, LBX + 57, LBX + 65,
   LBX + 73, LBX + 81, LBX + 90)
g) Maßstabsangaben (LBX + 492)
h) Angaben über die Form der Daten, d.h. 80 Zeichen/Zeile oder
   72 Zeichen/Zeile (KBX + 88)

AUSGABEPARAMETER

a) Kennzeichen dafür, daß Daten umgewandelt werden
   (KBX + 12: KBIN = X'80')
b) Kennzeichen für zuviele Daten
   (KBX + 8: KBEOD = X'80' sonst KBEOD = X'00')
c) Die umgewandelten Maßstabsangaben stehen im Zahlenfeld ab Z (26)
d) Die umgewandelten Daten befinden sich im Zahlenfeld ab Z (40)
e) Die Anzahl der umgewandelten Daten steht in Z (25)
f) Wichtige Steuerparameter für weitere Aufrufe stehen im Zahlenfeld Z (20) .. Z(24)
g) Fehlercode (LBX + 39: LCC12)

PROGRAMMABLAUF

DKPUM Eingangsparameter ist das Parameterfeld LISTE, das die Pointer zur KBX und zur LBX enthält. Auf die KBX und die LBX kann mit Hilfe von Based-Variablen zugriffen werden, deren Pointer den aktuellen Wert enthalten. Deshalb werden beim Eintritt in DKPUM alle benötigten Adressen errechnet und die Pointer initialisiert. Danach wird festgestellt, ob es sich um einen wiederholten Aufruf handelt (KBIN = X'80'). Die Label-Variable ILAB wird entsprechend initialisiert.

UMF Initialisierung der Pointer für das Charakter-Feld, in dem die Umwandlung der Daten erfolgt. Bei wiederholtem Aufruf

LAB(1) wird das Programm bei LAB(3) fortgesetzt. Die LBX-Codes werden lokalen Variablen zugewiesen und auf ihre Gültigkeit geprüft. Sollen Maßstabsgrößen umgewandelt werden, so werden Anfang und Ende des Maßstabsfeldes errechnet und den Kontrollvariablen zugewiesen. Die Label-Variable erhält den Wert ILAB = 2. Das

LAB(2) Programm verzweigt nach GET. Hier werden die Kontroll-Variablen zum Umformen des Datenbereiches bestimmt, wenn zum ersten Mal eingelesen wurde. Die Label-Variable, die den Programm-Ablauf steuert, erhält den Wert ILAB = 4. Ist KBLEN $<$ 80, so handelt es sich um Daten, die vom Benutzer direkt eingegeben wurden. Wurde als neunter Parameter des Maßstabsfeldes die Zahl 72 eingegeben, so werden nur 72 Zeichen als Daten interpretiert, die 8 folgenden Ziffern dienen zur Nummerierung der Daten. Ist dies nicht der Fall, so wird die Form der Daten KBIO (KBX + 88) entnommen. Das Programm wird bei GET fortgesetzt.

LAB(3) Die Kontroll-Variablen für wiederholtes Umwandeln werden bestimmt. In Z (25) steht die Anzahl der bereits umgewandelten Daten, in Z (20) bis Z (24) befinden sich die vorher schon ermittelten Steuergrößen. ILAB = 4. Es wird nach GET verzweigt.

LAB(4) Die Kontroll-Variablen werden gerettet, der Fehlercode in der LBX wird auf Null gesetzt, die Kontrolle geht wieder an TCPKAUS.

GET Die nächsten 80 Zeichen des Einlesebereiches werden adressiert. Ist das Ende erreicht, so wird das Programm dort fortgesetzt, wohin die Label-Variable ILAB zeigt.

UMF1 Es wird geprüft, ob das Zahlenfeld bereits aufgefüllt ist. Wenn ja, so wird nach EOD verzweigt. Sind die bei GET adressierten 80 Zeichen abgearbeitet, so wird das Programm ab GET wiederholt. Jetzt wird die Position des nächsten Blank gesucht. Ist dies das nächste Zeichen, so wird der Zeiger um eins erhöht und nach UMF1 gesprungen.

Sind führende Nullen vorhanden, so wird der Zeiger solange erhöht, bis sie eliminiert sind. Folgt Kommentar, so wird das Programm bei GET fortgesetzt. Folgt kein '+' so wird das folgende Programm bis UMF2 übersprungen. Ist es ein führendes '+' Zeichen, so wird der Zeiger um eins erhöht und nach UMF1 verzweigt. Ist es ein '+' vor dem Exponenten, so wird es durch Blank ersetzt.

UMF2 Es wird geprüft, ob die Zahlenlänge richtig erkannt wurde. Ist dies nicht der Fall, so wird sie korrigiert. Danach wird festgestellt, ob es sich um Floating-Point-Zahlen handelt und ob ein Exponent vorhanden ist. Steht zwischen 'E' und dem Exponenten ein Blank, so muß die Gesamtlänge der Zahl erneut festgestellt werden. Jetzt werden alle ermittelten Positionen in das richtige Verhältnis zueinander gebracht. Sind noch führende Nullen mit davorstehenden '-' vorhanden, so wird das '-' nach rechts verschoben, der Zeiger

KFNULL korrigiert und nach UMF1 verzweigt. Sind die ermittelten Positionen und Längen nicht möglich, so wird das Programm über EOD beendet. Jetzt kann die untersuchte Zahl konvertiert und in das Zahlenfeld gebracht werden. Der Zeiger UMJ1 wird um die Länge der umgewandelten Zahl erhöht. Er zeigt somit auf den dahinterliegenden Charakter String.

Wo das Programm fortgesetzt werden soll, bestimmen die LBX-Codes. Beim Automatischen Modus müssen die laufenden X-Positionen errechnet und hinzugefügt werden. Beim Logarithmischen Maßstab muß logarithmiert werden. Es bietet sich hier noch eine Vielfalt von Möglichkeiten der Datenmanipulation, die bei weitem nicht erschöpft ist. Das Programm wird für alle bis jetzt zugelassenen LBX-

EOD Codes bei UMF1 fortgesetzt. Werden mehr Zahlen angeboten als gespeichert werden können, so wird KBEOD gesetzt und das Programm

ENDE normal beendet. ENDE und RETURN zu TCPKAUS.

3.2. CANCAN (G. Fleck)

CANCAN ist ein Assembler Unterprogramm zu folgenden PL/1-Programmen: DKPV1, DKPV2, DKPV3, PLOTA1, PLOTA2 und PLOTA3. Es wird aufgerufen via CALL und es ermöglicht die Datenausgabe auf Datensichtgeräte. Die dazu nötigen Kontrollblöcke und das Kanalprogramm werden in TCP1DSUP erstellt. In CANCAN werden die aktuellen Adressen und Zähler eingesetzt und das Kanalprogramm gestartet.

EINGANGSPARAMETER

a) R1
b) Adresse des Ausgabefeldes (KBX + 88: KBIO)
c) Ausgabemodus (KBX + 92: KBMOD)
d) Länge des Ausgabefeldes (KBX + 93: KBIOL)

AUSGANGSPARAMETER

LCC12: Fehlercode

PROGRAMMABLAUF

CANCAN Es werden die Adressen der KONBOX (KBX) und der LINEBOX (LBX) aus der Parameterliste in KBXREG und LBXREG gebracht, um auf die Konbox und Linebox Zugriff zu haben. Die Adresse der benötigten

GETUA Control-Unit (CU) steht in der LBX. Dazu wird die passende Displaybox (DBX) gesucht. Ist eine solche nicht vorhanden, erfolgt eine Verzweigung zur Fehlerroutine ERROR1. Die Display-Adresse, d.h. die Adresse eines Displaygerätes einer CU, soll vor der

DBXFOUND Modus-Angabe im Steuerbyte stehen. Dazu wird die Display-Adresse positionsgerecht ins RWKC gebracht. Soll nur gelöscht werden, verzweigt das Programm zu LOE, sonst wird das Steuerbyte mit dem aktuellen Modus aufbereitet und nach STBYTE gebracht. Danach werden die Datenadresse und die Längenangabe in das WRITECCW gebracht.

Die Startadresse des Kanalprogramms muß in IOBSIOCC stehen. Ist der Modus X'08', d.h. keine Textausgabe, so wird nach EXECUTE verzweigt, ansonsten muß vorher der Text von EBCDIC in den ASCI-Code übersetzt werden.

LR
LO1 — Handelt es sich um den freipositionierbaren Modus, so muß nur jedes dritte Zeichen übersetzt werden RWKB = 3. Bei dem Fernschreibmodus werden alle Zeichen übersetzt RWKB = 1.

RT86 — Translate-Routine und Sprung nach EXECUTE.

LOE — Das Lösch-Byte wird mit der Display-Nummer versehen und nach LBYTE gebracht. Die Startadresse für das schon vorbereitete Kanalprogramm, das nur den Bildschirm löscht, wird nach IOBSIOCC

EXECUTE — gebracht. Ein Zähler für Fehlerversuche wird auf drei gesetzt.

EXCP — ECB und IOBFLAGS werden zurückgesetzt und es erfolgt der Aufruf des Kanalprogramms mit dem Makro EXCP. War die Ausgabe erfolgreich

RETRY — so wird nach RETOUR verzweigt. War es der dritte Fehlstart, so wird nach ERROR4 verzweigt. Ist der Kanal "NOT OPERATIONAL" so erfolgt ein Sprung nach ERROR5. Ansonsten wird der Zähler um eins vermindert und das Programm bei EXCP fortgesetzt.

ERROR1 — Laden Adresse der Fehlernachricht MSG48 und Sprung nach ERRORTYP.

ERROR4 — Laden Adresse der Fehlernachricht MSG48. In diese Nachricht wird die Nummer der aktuellen CU eingefügt. Im DBXECB wird "PERMANENT I/O-ERROR" angezeigt. Sprung nach ERRORTYP.

ERROR5 — Laden Adresse der Fehlernachricht MSG49. Sonst wie unter ERROR4.

ERRORTYP — WTO-Makro

ERETOUR — ABEND 123 und Ausgabe eines Dumps. Der Fehlercode in der LBX wird gesetzt (LCC12 = X'F2F0').

RETOUR — Um zu verhindern, daß während der Ausgabe des Bildes in den Ausgabepuffer neue Daten übertragen werden, wird der Timer gesetzt und gewartet. Die Wartezeit ist jedoch wesentlich kürzer als die Ausgabezeit. Danach erfolgt der Rücksprung in das aufrufende Programm.

3.3. DKPV1 (G. Fleck)

DKPV1 ist ein PL/1-Programm, das vom Display-Kontroll-Programm (DKP) via LINK aufgerufen wird. Es benötigt für die Bildausgaben das Assembler Unterprogramm CANCAN. Für den Aufruf von DKPV1 wird vorausgesetzt, daß TCPKAUS und DKPUM erfolgreich beendet wurden.

FUNKTION

Die eingelesenen Zahlen werden normiert und auf die gewünschte Bildebene umgerechnet. Sollen die Punkte durch Linien miteinander verbunden werden, so müssen mögliche Schnittpunkte auf dem Bildrand errechnet werden. Die so ermittelten Bildpunkte müssen mit Hell- bzw. Dunkelvektoren miteinander verbunden werden. Die Ausgabedaten bestehen aus einem binären Vektorfeld, dessen Ausgabe an das Sichtgerät von dem Unterprogramm CANCAN veranlaßt wird. Sind alle Daten ausgegeben, so erfolgt die Aufbereitung der Maßstabslinien. Dazu werden die erforderlichen Anfangs- und Endpunkte der Achsen und der Teilstriche errechnet, die dann wieder in Vektoren umgerechnet werden müssen. Nach der Ausgabe der Maßstabslinien wird die Beschriftung aufbereitet und zu den Sichtgeräten übertragen.

EINGANGSPARAMETER

(siehe Bild 5)

a) Adresse der KONBOX
b) Adresse der aktuellen LINEBOX
c) Adresse des Zahlenfeldes (KBX + 9: KBADD)
d) LBX-Codes (LBX + 39: LBC, LBX + 65: LBC2
LBX + 73: LBC3, LBX + 81: LBC4)

AUSGABEPARAMETER

Fehlercode (LBX + 39: LCC12)

PROGRAMMABLAUF

DKPV1 Eingabeparameter ist wie bei allen übrigen PL/1-Programmen, die zur Display-Ausstattung gehören, das Parameterfeld 'LISTE'. Es enthält die Pointer zur KBX und zur LBX. Auf alle außerhalb des Programms liegenden Daten wird mit Hilfe von Based-Variablen zugegriffen, deren Pointer die aktuellen Werte erhalten. Alle Pointer der Variablen, die auf die KBX, die LBX oder auf das Datenfeld zugreifen, werden zunächst initialisiert. Die Label-Variable ILAB, die den Programmablauf bestimmt, erhält den Wert ILAB = 1. Für den Fall, daß keine Maßstabsangaben vorhanden sind, werden die LBX-Codes LBC3 und LBC4 initialisiert und die Faktoren Z (IRZM) und Z (IRXM) erhalten den Wert 1. Ist der LBX-Code LBC = X'16', so wird zum Löschen des Bildschirmes das Assembler Unterprogramm CANCAN aufgerufen. Danach werden die Bildgrenzen festgelegt. Falls die Zahlenfolge der Daten nicht die gewünschte Form hat, werden die Daten umgeordnet.

ZXFOLGE Sind Maxima und Minima der Daten bekannt, so werden die Maßstabsfaktoren bestimmt und das Programm bei MIMAXE fortgesetzt.

MIMAX Die Maxima und Minima der Daten werden gesucht.

MIMAXE Sind Maxima und Minima gleich, so werden sie zur Maßstabsbestimmung um einen relativen Betrag geändert. Anschließend werden Normierungsfaktoren bestimmt. Die Daten werden normiert. Danach wird die Lage der Koordinaten-Achsen relativ zu den normierten Daten errechnet. In Z (IRZM) und Z (IRXM) steht der Multiplikationsfaktor in Z (IRZD) und Z (IRXD) steht das Displacement für

KM1 Bildaufbereitung. Hier wird der Abstand der Skaleneinteilung festgelegt. Im ersten Durchlauf für die X-Richtung im zweiten Durch-

KM2 lauf für die Z-Richtung. In KOXP und KOZP stehen die Werte der Skalenabstände für eine spätere Textausgabe. Es werden die Bildgrenzen für die Display-Form errechnet. Liegen die Koordinatenachsen außerhalb dieser Bildgrenzen, so wird die Lage der Maßstabslinien auf den linken bzw. unteren Bildrand festgelegt.

INIT Hier werden die für die Bildausgabe wichtigen Kontrollvariablen initialisiert. JON zeigt auf den Anfang des Ausgabepuffers. ID zeigt auf das Datenfeld, das als nächstes zu bearbeiten ist.
Das Programm wird dort fortgesetzt, wo die Label-Variable ILAB
LAB(1) hinzeigt. Es wird geprüft, ob der Ausgabepuffer voll ist. Wenn ja,
RAND so wird nach BILDENDE verzweigt. Es wird untersucht, ob die Daten eines Datenbereiches abgearbeitet sind. Ist dies der Fall, so wird ILAB = 2 gesetzt und nach BILDENDE gesprungen. In diesem Programmabschnitt soll geprüft werden, ob bei der Verbindung der neuen Koordinaten mit den vorhergegangen sich Schnittpunkte auf dem Bildrand ergeben. Sollen nur Punkte ausgegeben werden, so werden alle nicht zum Bild gehörigen Punkte ausselektiert. Die Koordinaten des zuletzt bearbeiteten Punktes stehen in Z (1) und Z (2), die des neuen Punktes in Z (3) und Z (4) . Es bedeuten:
SCHO = '00000000'B beide Punkte außerhalb
SCHO = '11111111'B beide Punkte innerhalb
SCH1 = '1111'B 1. Punkt innerhalb
SCH1 = '0000'B 1. Punkt außerhalb
Für den 2. Punkt wird entsprechend SCH2 gesetzt. SCH1 und SCH2 sind Unterdefinitionen von SCHO.
Mit dem Schalter SCH wird das Vorzeichen der Daten erfaßt, sofern es zur Hell- bzw. Dunkelsteuerung benutzt wird. Es bedeuten:
SCH = '00000000'B beide Werte positiv
SCH = '11111111'B beide Werte negativ
SCHX und SCHZ sind innerhalb von SCH definiert. Mit der Variablen DUHELL kann die Vorzeichen-Steuerung ein- bzw. ausgeschaltet werden
DUHELL < = 0 Steuerung EIN
DUHELL > 0 Steuerung AUS
Wird der erste Punkt des Datenfeldes bearbeitet (ID = IA), so wird in allen Fällen mit einem Dunkelvektor begonnen (SCHZ = '1111' B). Für die erste Verarbeitungsart (ILAB = 1, d.h. Datenausgabe) wird die Vorzeichensteuerung nicht benötigt und das Programm bei
RA1 RA2 fortgesetzt. Sonst wird bei negativen Z-bzw. X-Werten das Vorzeichen geändert und SCHZ bzw. SCHX gesetzt.

RA2 Der nächste Punkt wird nun auf die Bildebene umgerechnet. Liegt der neue Punkt außerhalb des vorgegebenen Bildrahmens, so wird SCH2 = '0000'B. Kann der neue Punkt unberücksichtigt bleiben, so wird das Programm bei BAUS fortgesetzt. Sind beide Punkte innerhalb des Bilderrahmens, so wird nach BIN verzweigt.

RGTH In den restlichen Fällen müssen die Schnittpunkte auf dem Rand ermittelt werden. Die auszugebenden Punkte werden nach Z (20) bis Z (23) gespeichert. Das Vorzeichen wird zu Hell-/Dunkelsteuerung benutzt. Liegt der Zielpunkt innerhalb, so werden seine Koordinaten nach Z (22) und Z (23) gebracht. Das Programm wird bei

BAUS VECTA fortgesetzt. Der Schalter SCH2 = '0000'B, d.h. neuer Punkt

BIN außerhalb. Der nächste Punkt wird geprüft. Der Zielpunkt wird eingetragen unter Berücksichtigung der Hell-/Dunkelsteuerung. Es wird zur Vektoraufbereitung nach VECTA verzweigt.

LAB(2) Es folgt die Aufbereitung der Maßstabslinien. Label-Variable ILAB = 3. Der Zeiger des Datenfeldes erhält den Wert IA = 40. Soll die Ausgabe von Maßstabslinien unterdrückt werden, so wird

KOVERA das Programm beendet. DUHELL = 0, d.h. Hell-/Dunkelsteuerung EIN.

KMA Die Daten für die Maßstabslinien werden aufbereitet. Das Programm

VECTA wird bei VECT fortgesetzt. Soll bei der Vektoraufbereitung zu jedem Punkt die Ordinate hellgetastet werden (Strichspektren), so werden an dieser Stelle die nötigen Zwischenpunkte in das Zahlen-

VECT feld eingefügt.Hier beginnt das Aufbereitungsprogramm für das binäre Ausgabefeld. Alle auszugebenden Punkte werden durch Hell- bzw. Dunkelvektoren miteinander verbunden. Ausgangspunkt ist P1 (0,0) in der linken unteren Ecke. Die Koordinaten des vorangegangenen Punktes werden nach VFX(1) bzw. VFZ (1) gebracht, die des Zielpunktes stehen in VFX (VN) bzw. VFZ (VN). Je nach Anzahl der nötigen Zwischenpunkte erhält VN einen Wert zwischen 2 und 36. Sind die Vorzeichen des nächsten Zahlenpaares positiv, so wird

VHD1 das Programm bei VHD3 fortgesetzt. Das Dunkelbit des nächsten

VHDO Vektors wird gesetzt. Das Vorzeichen der Ordinate wird korrigiert.

Eine Kontrollvariable für Hellpunkte wird gesetzt. Das Vorzeichen

VHD3 der Abszisse wird korrigiert. Der Abstand des neuen Punktes zum

VDX1 vorangegangenen Punkt wird ermittelt. Die Richtungsbit des ak-

VDZ2 tuellen Vektors werden gesetzt. Ist der Abstand größer als 15 Punkte, so wird nach LVECT verzweigt. Der Vektor wird aufbereitet und mit dem vorangegangenen verglichen. War dieser gleich dem

VECT2 neuen, so wird nur sein Multiplikationsfaktor VM erhöht, sonst wird der neue Vektor ins Ausgabefeld übertragen. Sollen Hellpunkte

VHELLP ausgegeben werden, so wird ein entsprechendes Bitmuster hinter dem letzten Vektor bereitgestellt. An dieser Stelle könnten beliebige andere Punktetypen eingefügt werden. Das Programm wird bei VECT

LVECT fortgesetzt. Im folgenden Programmabschnitt werden nur horizontale bzw. vertikale Vektoren aufbereitet. Nach dieser Methode werden Dunkelvektoren aufgebaut und Hellvektoren, die horizontal oder vertikal verlaufen. Die übrigen Hellvektoren werden vor dem Programm ab HVECT errechnet und in das Ausgabefeld übertragen.

LVZ Der X-Anteil des Vektors wird auf Null gesetzt. Es gibt 3 Stufen der Vektoraufbereitung:

1.) 1 <= DZ <= 31 VM = 1

LVZ3 2.) 1 <= DZ <= 7 VM =16

LVZ256 3.) 2 Vektoren mit DZ = 8 und VM = 16

Ist DX = 0 (Vertikale Gerade), so wird nach VHELLP verzweigt.

LVX Der Z-Anteil des Vektors wird auf Null gesetzt. Die drei Stufen der Vektoraufbereitung sind:

1.) 1 <= DX <= 31 und VM = 1

LVX3 2.) 1 <= DX <= 7 und VM =16

LVX256 3.) 2 Vektoren mit DX = 8 und VM = 16

Das Programm wird bei VHELLP fortgesetzt.

HVECT Die Koordinaten des Zielpunktes werden in VFX (36) und VFZ (36) gespeichert. Die Vorzeichen von DX und DZ wurden schon in den Vorzeichenbit des Vektors berücksichtigt. Zur Erreichung des Zielpunktes muß deshalb ein Pseudo-Zielpunkt ermittelt werden. Nun werden soviele Zwischenpunkte errechnet, daß alle Abstände max. 15 Punkte betragen.

Alle Zwischenpunkte werden nacheinander durch mehrere einfache Vektoren angesteuert (VM = 1). Sind aufeindanderfolgende Vektoren gleich, so wird wiederum nur der Multiplikationsfaktor VM erhöht und kein neuer Vektor in das Ausgabefeld übertragen.

HDV2 Ist der neue Vektor ungleich dem vorangegangenen oder der VM = 31, so wird der neue Vektor in das Ausgabefeld übertragen

HDV5 und VM = 1 gesetzt. Sind alle Vektoren aufbereitet, d.h. ist der Pseudopunkt erreicht, so wird das Programm bei VECT fortgesetzt.

BILDENDE Modus und Länge der Ausgabedaten werden in eine PARMLIST gebracht. Das Ausgabeprogramm CANCAN wird aufgerufen. Nach korrektem Verlauf von CANCAN wird die Kontrollvariable ID zurückgesetzt und nach

LAB(3) INIT verzweigt. Das Ausgabefeld für die Maßstabslinien ist aufbereitet. Zur Ausgabe der Daten wird nach BILDENDE verzweigt.

LAB(4) Es wird die Zeilennummer ermittelt, in der die Beschriftung

DDTEXT stehen soll. Es folgt die Aufbereitung der Textzeile und ein

COMEINF Sprung zur Ausgabe BILDENDE. Fehlercode LBCC = X'FFFE' für System-

LAB(5) fehler wird gesetzt. Der ursprüngliche Befehlscode wird wieder in

ENDE LBC eingesetzt. Ende und Return zu DKP.

3.4. DKPV2 (G. Fleck)

Das PL/1-Programm DKPV2 hat eine ähnliche Funktion wie DKPV1. Es benötigt dieselbe Ein-/Ausgabe-Parameter und benützt ebenfalls dasselbe Unterprogramm CANCAN. Verschieden sind die zu bearbeitenden Daten. Hier dienen die Vorzeichen der eingelesenen Zahlen zur Hell-/Dunkelsteuerung. Im folgenden werden nur Änderungen gegenüber DKPV1 aufgezeigt.

Änderungen im Programmablauf gegenüber DKPV1:

MIMAX Bei der Suche nach dem Maximum und Minimum werden nur die Beträge der Daten verglichen.

KM2 Es werden die Koordinaten für die Maßstabslinien am unteren und rechten Bildrand bestimmt. Koordinatenachsen innerhalb des Bildes können sich nicht ergeben, da alle Zahlen positiv gewertet werden.

3.5. DKPV3 (G. Fleck)

Das PL/1-Programm DKPV3 hat denselben Ablauf wie DKPV1. Unterschiedlich ist nur die Berechnung der Skalenabstände. Sie erfolgt im logarithmischen Maßstab. Die Daten wurden in DKPUM logarithmiert Die Ein-/Ausgabeparameter sind dieselben wie die in DKPV1.
Änderungen im Programmablauf gegenüber DKPV1:

MIMAXE Minimum bzw. Maximum des Ausgabebereiches wird bei logarithmischer Darstellung auf eine volle Zehnerpotenz ab- bzw. aufgerundet.

KMA Sollen die auszugebenden Daten in X- oder Z-Richtung logarithmisch dargestellt werden, so erfolgt die Skaleneinteilung jeweils im logarithmischen Maßstab. Bei linearer Darstellung erfolgt die übliche Skaleneinteilung wie in DKPV1.

4. TCP1PAUS (M. Schayegh)

Dieser Module wird nach einem Display-Befehl, der eine graphische Ausgabe mit PLOT-Daten zur Folge hat, von DKP durch LINK aktiviert.

FUNKTION

Bereitstellung von Daten aus verschieden organisierten Datasets.

EINGANGSPARAMETER

R11 = A (KONBOX)
R 3 = A (LINEBOX)

AUSGABEPARAMETER

LCC12 Fehlercode

PROGRAMMABLAUF

Nach der Savearea-Verkettung und Initialisierung wird bei dem Befehl DO nach DO gesprungen, sonst BCODE in LINEBOX umgespeichert.

PLUSMIN
AO

Bei D+ oder D- wird nach DØLM gesprungen. Sonst WCODE2 in der LINEBOX umgespeichert. Wenn es sich um einen PDS-handelt, der nicht zur TCP-Bibliothek gehört, so werden der Membername und Datasetname abgespeichert, zwecks späterer Abfrage in BFLG (innerhalb LINEBOX) ein Code gesetzt und nach FPDS gesprungen.

KREUZ

Wenn es sich um ein Member aus der TCP-Bibliothek handelt, so wird nach EMEM gesprungen. Sonst ist eine Ausgabe aus einem SDS verlangt. Hier wird der Datasetname umgespeichert und nach FSDS

EMEM
PLOT

gesprungen. Hier wird der Membername umgespeichert und in BFLG ein Code gesetzt. Block-Nr. und Seiten-Nr. werden in Binärform umgewandelt und in der LINEBOX abgespeichert und zu der geeigneten

LAPSYN

Routine gesprungen. SYNAD- und EODAD-Adresse werden in den DCB eingesetzt und einige Register mit Adressen und Anfangswerten

FIND

versorgt. Anfangsadresse des Members wird für READ bereitgestellt.

OK
A6

Hier wird ein Block gelesen und anschließend zur Verarbeitung nach einer anderen Routine gesprungen. Hier wird der gelesene Block recordweise abgeprüft, bis bei einem Record ein "A" als 6.Zeichen gefunden ist. Findet man dies innerhalb des Blockes nicht, so wird ein neuer Block gelesen. Es muß sooft ein solcher Record gefunden werden, bis man zu dem gewünschten Teil der Daten kommt, der durch den Befehl verlangt wurde. Kommt man beim Lesen zu "End of Data" bevor der gewünschte Teil gefunden wurde, so wird bei PEND ein Fehlercode gesetzt und das Programm beendet. Hat man den gewünschten Datenteil, so wird ein zusätzlicher Block angehängt.

LINK

Ruft die beiden PL/1-Routinen PLOTA1 und PLOTA2 zum Aufbau der Koordinaten und PLOTA3 zur Verarbeitung der Daten. Bei weiterer Datenübergabe, die jeweils 2 Blöcke übergibt, wird nur noch PLOTA3

M32

aufgerufen. Dient dem Zweck, weitere Blöcke einzulesen und an PLOTA3 weiterzugeben. Der Wert X'C5' an der Stelle 92 (R11) nach einem Rücksprung von PLOTA3 bedeutet die Einstellung der Daten-

FPDS

übergabe und Beendigung des Programmes. Makro LOCATE stellt die Serial-Nr. von der Platte zur Verfügung auf dem sich der Dataset befindet.

OBTAIN liest den DSCB des betreffenden Dataset.
Mit den Informationen aus dem DSCB wird der DCB ergänzt.
Bei dieser Ergänzung wird auch die Organisationsart des Datasets berücksichtigt. Mit RDJFCB wird der "Job file Control block" gelesen, und nach Einsetzen der DSNAME mit OPEN TYPE=J der Dataset geöffnet. Wenn die Ausgabe aus einem PDS erfolgen soll, so wird nach PARTPLOT gesprungen. Sonst wird zuerst in der Routine PLOT die Seiten-Nr. errechnet und abgespeichert und dann Daten aus

GET einem SDS in E/A-Feld transportiert. Die Logik bei der Ausgabe aus einem SDS ist genau wie bei der Ausgabe aus einem PDS.
Zunächst werden soviel Records ignoriert, bis der Datenteil (sog. A-Block) erreicht ist, welcher von dem Benutzer bei der Befehlseingabe spezifiziert wurde. Stößt man beim Lesen auf 'EODAD' bevor dieser A-Block erreicht ist, so wird ein Fehlercode gesetzt und das Programm beendet. Sonst werden die relevanten Daten bis zu 3200 Byte in das E/A-Feld übertragen und dann die Moduln PLOTA1, PLOTA2 und PLOTA3 aufgerufen. Hat PLOTA3 aufgerufen. Hat PLOTA3 bei dem Rücksprung den Code X'C5' an der Stelle 92 (R11),so erfolgt kein Datentransport mehr und das Programm wird beendet. Ist das nicht der Fall, so werden weitere 3200 Bytes in das E/A-Feld übertragen aber nur noch PLOTA3 aufgerufen. Stoßt man beim Lesen auf EODAD bevor das E/A-Feld 3200 Byte enthält, so wird an der Stelle 92 (R11) der Code X'C5' gesetzt, damit bei dem Rücksprung aus PLOTA3 das Programm beendet werden kann.

PARTPLOT Stellt die Daten aus einem beliebigen PDS für die Plotausgabe zur Verfügung. Der Ablauf ist genau wie oben, mit dem Unterschied, daß hier die Daten mit READ gelesen werden .

FEND Ist die EODAD-Routine bei der Ausgabe von Datasets, die nicht zur

DØ TCP-Bibliothek gehören. Hier werden nach dem Befehl DO einige Felder in der LINEBOX umgespeichert.

DPLM Bei D+ oder D- wird die Seiten-Nr. neu errechnet und in die

ENDE LINEBOX abgespeichert. Hier wird das Programm beendet.

4.1. PLOTA1, PLOTA2 und PLOTA3 (R. Bader)

Diese Moduln werden nach dem Befehl
'de, Name P, xx.y '
von TCPPAUS in der genannten Reihenfolge (PLOTA3 eventuell mehrmals) durch das Makro LINK aktiviert.

FUNKTION

Das Programm PLOTA1 generiert die Koordinatenachsen und die Koordinateneinteilung des auszugebenden Plots.
Der Modul PLOTA2 erzeugt die dazugehörige Beschriftung, sowie weitere gewünschte Texte innerhalb des Bildes, während PLOTA3 die Daten für eine Displayausgabe aufbereitet.

WICHTIG!

Die 3 oben genannten Programm benützen als zu verarbeitende Daten den Output des Programmes PLOTA (siehe /12/) Daher ist es nicht möglich die folgenden Ausführungen zu verstehen, ohne gründliche Kenntnisse zumindest dieser Programmbeschreibung.

EINGANGSPARAMETER

Adresse von LBX und KBX. Die Parameter werden laut IBM - Konvention übergeben. (siehe Bild 6)

PROGRAMMABLAUF

Die Logik der Programme beruht im wesentlichen darauf, jeden Programm-Kern 2-mal zu durchlaufen (x, y-Richtung) (Ausnahme PLOTA3). Dazu werden die Schalter SCHR, SCHV und die Inkremente IDR und L benutzt. Die einzelnen internen Variablen besitzen im allgemeinen dieselben Namen wie in PLOTA, so daß ihre Funktion dort beschrieben ist. Eine wesentliche Unterscheidung wird immer bei der Frage logarithmischer oder nichtlog. Plot gemacht,

In allen drei Programmen werden die fertig aufbereiteten Daten in dem Unterprogramm CANCAN ausgegeben. In PLOTA1 und PLOTA3 gibt es einen Programmteil, der mit dem Label VECT beginnt. Die genauere Beschreibung der darauf folgenden Statements befindet sich in DKPV1. Es folgt nun die Beschreibung der wesentlichen Teile der einzelnen Programme.

4.1.1. PLOTA1

A) Initialisierungsphase

Schalter und Inkremente werden für Normalplot (d.h. ohne eigene Skaleneinteilung, ohne zusätzliche Beschriftung und nicht logarithmische Darstellung) und die Aufbereitung in X-Richtung gesetzt. Nach den einzelnen Kontrollabfragen (- mehr als 3200 Byte Datenlänge - Befehlstyp DE oder DD - 'Bild löschen' korrekt ausgeführt)

NEXT folgt das Aussuchen der einzelnen Kontrollkarten (A,B,BX,BY,C,CF-Karte) und die Zuweisung der einzelnen Parameter zu den internen Variablen.

B) Verarbeitungsphase

VERARB Die eigentliche Verarbeitung beginnt mit der Festlegung von Bildgrenzen, Inkrementen usw. (wie in PLOTA genauestens beschrieben) unter Berücksichtigung der Möglichkeiten des Display-Bildschirms. Die Aufteilung des Bildschirms und des Plots geschieht wie in Bild 7.
Damit der Plot auf dem rechteckigen Bildschirm wieder quadratisch erscheint, wird der Ursprung O_D in den Ursprung O_P verschoben.
Ist der Parameter IDR von PLOTA > 1, so wird das Bild nicht in voller Länge auf dem Display gezeigt, sondern wird durch die entsprechende Befehlssteuerung (D+ oder D-) oder Parametersteuerung (Y bei Plotbefehl) jeweils um 206 Display-Bildpunkte verschoben.
Die bisher gewonnenen Parameter werden im ZFELD (16-K Feld, siehe DKP) gespeichert, damit eine Wiederberechnung in PLOTA2 und PLOTA3 entfällt.

B5 Die X-Achse und (falls im Bild) die Y-Achse werden aufbereitet und in das ZFELD ab Index 100 weggespeichert.

BK1 Koordinateneinteilung der X-Achse.

DYNULL Koordinateneinteilung der Y-Achse.

Falls die Y-Achse außerhalb des Bildausschnittes liegt, wird eine

KREUZ Pseudo-Achse bestehend aus kleinen Kreuzchen generiert.

LOGROUT Entsprechende Skalenaufbereitung für log. Plot.

NULOG Neuberechnung verschiedener Parameter, die in PLOTA2 gebraucht werden.

VECT siehe DKPV1

C) Bildausgabe

VEND Bildausgabe mit CANCAN

4.1.2. PLOTA2

A) Initialisierungsphase

Dasselbe wie in PLOTA1 (außer Entscheidung DD oder DE)

B) Verarbeitung

VERARB Die Textunterschrift wird generiert, positioniert und mit der Routine

TEX (1) CHMOD aufbereitet für Textausgabe:

TEX (2)

Z-Position	X-Position	Zeichen	Z-Position	

0 23

CHMOD

WEITER generiert obige Darstellung

KOBES Koordinatenbeschriftung
Falls eigene Skaleneinteilung gewünscht, wird die geforderte Darstellung in der Routine B7.....B4, erfaßt.

B5...B15 Die Skalenbeschriftung wird berechnet, in gewünschtem Format dargestellt, positioniert und mit CHMOD aufbereitet.
Beschränkung in der Darstellung:

Standart: E 9.2

	IW	ID
E-Format:	max IW = 9	
		max ID = 7
F-Format:	max IW = 10	
		max ID = 5
I-Format:	max IW = 10	

Überschneidungen werden geprüft (CONTRX, CONTRZ) und gegebenenfalls unterdrückt.

ZUSBES Falls zusätzliche Beschriftung gewünscht, wird diese in der folgenden Routine aufbereitet.
Beschränkung: Nur waagerechte Schrift möglich.

C) Bildausgabe

AUSG Ausgabe des Bildes mit CANCAN

4.1.3. PLOTA3

A) Initialisierungsphase

Dasselbe wie in PLOTA2
Zusätzlich: Übersetzungstabelle
Damit werden Zahlen, die in Basis 32 dargestellt sind, in Dezimalzahlen umgewandelt (notwendig, da die Koordinaten des Plots im Output von PLOTA zur Basis 32 dargestellt sind.

B) Verarbeitung

NEXT

CVERARB Die Darstellungsart der Kurve wird berechnet.
In NULB96 steht der gewünschte Punktetyp (Fall NT = 1 oder 3)
HPKTZ zählt die aufbereiteten Punkte seit dem letzten mit einem Zeichen markierten.
IPKTZ zeigt an, der wievielte Punkt immer markiert werden soll.
Die Tabelle der aufbereiteten einzelnen Punktetypen befinden sich

RAND in DKP. Umwandlung der Koordinaten in Dezimalzahlen und anschließende Überprüfung ob im Bild oder nicht. (Einzelheiten siehe DKPV1)

VECT siehe DKPV1

C) **Bildausgabe**

BILDENDE Ausgabe mit CANCAN. Falls Z(20) > 0 und DUHELL nicht gleich 2, soll das nächste Kurvenstück ohne Unterbrechung an das alte angeschlossen werden.

5. TCP1DAUS (M. Schayegh)

Dieser Module wird nach dem Befehl 'DFZ, DSN, SEITE 'von DKP durch LINK aktiviert

FUNKTION

Das Programm gibt das Directory eines beliebigen PDS auf dem Display aus.

EINGANGSPARAMETER

R11 = A (KONBOX)
8 (R11) = A (E/A-Feld)
R3 = A (LINEBOX)
DSN und SEITE werden der LINEBOX entnommen.

AUSGANGSPARAMETER

LCC12 Fehlercode

Der Module besteht aus 2 CSECTS

1) FZ 2) CANCANAS

PROGRAMMABLAUF

Nach Save Area - Verkettung und Umspeichern von Adressen wird CANCANAS aufgerufen und der Bildschirm gelöscht. Durch mehrere Abfragen wird entschieden ob der Befehl DO, oder D+/D- oder normaler Befehl vorliegt und je nachdem wird Seiten-Nr. nicht verändert oder um 1 geändert oder neu errechnet.

System-Makro LOCATE stellt die Vol.Ser.Nr. der Platte zur Verfügung, auf der sich der angegebene Dataset befindet.

OBTAIN Makro OBTAIN stellt den DSCB des betreffenden Datasets zur Verfügung.

Mit den Angaben aus DSCB wird der DCB für den betreffenden Dataset ergänzt, eine Abfrage stellt fest, ob es sich um einen PDS handelt und schließlich wird über EXLIST die Adresse eines JFCB - Feldes in den DCB gespeichert.

Makro RDJFCB liest die DD-Karte in das angegebene JFCB-Feld. Dieses Feld wird mit den Angaben im DCB ergänzt und es wird mit OPEN TYPE = J der Dataset geöffnet.

Es wird dann abgefragt, ob die Seite neu errechnet werden muß.

TMDS Hier wird die angegebene Seiten-Nr. nach Gültigkeit geprüft, in Binärform umgewandelt und in DSEITE+4 abgespeichert.

NICHTDA Im Falle eines Fehlers, z.B. wenn die Nummer nicht aus gültigen Zeichen besteht oder den Wert Null hat, wird ein Fehlercode gesetzt und das Programm beendet.

NRFALS Hier wird Fehlercode für nichtgültige Nr. gesetzt.

PLUSMIN Ändert die letztangegebene Seiten-Nr. um $\pm$ 1.

WEITER Es wird ausgerechnet, wieviel Directory-Einträge weggelassen werden sollen, bis die geforderte Seite ausgegeben werden kann. Anschließend wird ein Directory-Block, der aus mehreren Einträgen besteht gelesen.

Innerhalb des Blockes werden solange die Einträge abgearbeitet bis entweder der letzte Eintrag des gesamten Directory erreicht wird oder ein Block ganz abgearbeitet ist; wobei in diesem Fall ein neuer Block gelesen wird und die Prozedur von vorne anfängt. Gleichzeitig wird beim Abarbeiten jedes Eintrages die Zahl der auszulassenden Einträge um eins vermindert und sobald diese Zahl Null wird, werden die Einträge über einen Zwischenspeicher von 80 Byte im E/A-Feld abgelegt. Sind 20 Einträge im E/A-Feld abgelegt, so werden die Zeichen im E/A-Feld in einen für Displayausgabe gültigen Code übersetzt und durch Aufruf von CANCANAS auf den Bildschirm ausgegeben.

ENDE Mit CLOSE, TTIMER und RETURN wird das Programm beendet.

Anhang A

A. 1 Die Dialogstation SIEMENS - TRANSDATA 8525 - 101 (W. Strolz)

Eine genaue Beschreibung der SIEMENS TRANSDATA findet man unter /9/.
In der vorliegenden Übersicht wird nur das Notwendigste über den Aufbau der Station und ihren Anschluß an die DVA IBM/360-65 beschrieben.
Die TRANSDATA 8525 - 101 ist modular aus folgenden Ein/Ausgabe-Geräten aufgebaut:

Tastatur
Lochstreifenleser für:
- CCITT - Nr.5 - Code (ISO7)
  mit gerader Parität
- beliebige 8- und 7-Kanal-Codes
  mit bestimmten Einschränkungen
- Fernschreiber-Code (5-Kanal, CCITT-Nr.2)

Datenschreiber
Lochstreifenlocher (CCITT-Nr. 5-Code)
Ein/Ausgabe-Kanal zur DVA

Diese Ein/Ausgabe-Geräte werden von einer zentralen programmierbaren Steuereinheit verwaltet.
Die Dialogstation kennt zwei prinzipielle Betriebsarten:

- Leitungsbetrieb (on-line)
- Lokalbetrieb (off-line)

Im Zusammenhang mit TCP wird nur der Leitungsbetrieb verwendet. Dabei wird nur mit der Betriebsart Freie Eingabe gearbeitet. Zur Selektion der verschiedenen kombinierbaren Ein/Ausgabe-Geräte werden Programmworte folgenden Aufbaus verwendet, die von der DVA zur Station geschickt werden:

EOT - CCITT - Nr.5 - Zeichen als
Sende - oder Empfangsaufforderung
DC1 - CCITT - Nr.5 - Zeichen als
Befehlseinleitung
PS1 - Geräteadresse 1
PS2 - Geräteadresse 2
PA - Programmanweisungszeichen,
Freie Eingabe
8-Kanal-Lochstreifen ISO7
8- " " Transparensmodus
5- " " CCITT-Nr.2
ENQ - CCITT-Nr.5-Zeichen als
Programmwort-Abschluß

Programmwort:
EOT/DC1/PS1/PS2/PA/ENQ

Anschluß an die IBM/360-65

Die Dialogstationen SIEMENS TRANSDATA 8525-101 sind über 4-Draht-Halbduplex-Leitungen an die IBM 2702 Transmission Control Unit angeschlossen. Die Übertragungsgeschwindigkeit beträgt 200 Baud (ca. 18,5 Zeichen/sec.). Die Daten werden dabei blockweise übertragen. Nach jedem Textblock erfolgt ein Quittungsaustausch. Näheres über die Übertragungsprozeduren findet man in dem angeführten SIEMENS-Handbuch für die Dialogstation. (siehe Bild 8)

A.2 Das Display-System DASIKA/360-B (G. Fleck)

Zu dem Display-System DASIKA/360-B gehört eine Tektronix-Speicherröhre T611, eine Display Unit und eine Control Unit für den Anschluß an den Selektorunterkanal der IBM 2870 Kanaleinheit. Hardware und die nötige Grundsoftware sind in /13/ ausführlich beschrieben. Die wichtigsten technischen Daten werden im folgenden aufgezählt:

a) Max. Kanalgeschwindigkeit: 180 K Byte/sec.

b) Die Übertragungsgeschwindigkeit richtet sich nach der Schreibgeschwindigkeit

   1. Schreibgeschwindigkeit für Alphazeichen:
      400 µsec/Zeichen
      (abhängig von der Anzahl der darin enthaltenen Punkte)

   2. Schreibgeschwindigkeit für Vektoren:
      Dunkelvektor : 3 µsec/Punktedistanz
      Hellvektor : 40 µsec/Punktedistanz
      Helltastpunkt: 20 µsec/Punkt

Zeit für längsten Hellvektor : 20 msec
Zeit für längsten Dunkelvektor : 3 msec

c) Graphische Operationen, die in TCP zur Anwendung gelangen:

   1. Textmodus:
      5 x 7 Punkte Charakter Generator.
      Zeichenaufbereitung im ASCI-Code (CCITT-Nr. 5)

   1.1. 80 Zeichen/Zeile mit automatischer Zeilenumschaltung.
      Max. 32 Zeilen/Bild.

Control Byte:

X	X	X	1	0	0	1	0
0			3				8

Bit 1 - 3 Display-Nr.
Bit 4 - 8 zur Modus Bestimmung

Daten:	1. Byte	2.Byte	usw.
	Zeichen	Zeichen	Zeichen

1.2. Freie Positionierung auf einem wahlfreien 256 x 256 Raster. Zeichengröße wie unter 1.1.

Control Byte:

X	X	X	1	0	0	1	0

0 3 8

Daten	1. Byte	2. Byte	3. Byte	
	Z-Pos.	X-Pos.	Zeichen	Z-Pos.

0 1 2 3

2. Vektormodus in einem 512 x 512 Punkteraster.

Control Byte:

X	X	X	0	1	0	0	0

0 3 8

Daten:	1. Byte	2. Byte	3. und 4. Byte	
	Z-Anfangs Position	X-Anfangs Position	1. Vektor	2.Vektor

0 1 2 3 4

Vektor

	1. Byte								2. Byte							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

0 4 8 9 10 11 16

Bit 1 - 4 DZ = Steigung in Z-Richtung
Bit 5 - 8 DX = Steigung in X-Richtung
Bit 9 VZ = Vorzeichen in Z-Richtung (0 ≡ '+')
Bit 10 VX = Vorzeichen in X-Richtung (0 ≡ '+')
Bit 11 HD = Hell-/Dunkelsteuerung (0 ≡ hell)
Bit 12 - 16 Z = Zähler für Anzahl der Vektoren

Hinweis: der Vektor X'000F' erzeugt einen Hellpunkt.

d) Die Anzahl der zu übertragenden Daten ist im Speicherbetrieb unbegrenzt.

e) Der Bildschirm kann von Hand oder aber vom Rechner durch ein Löschsignal gelöscht werden. Das Löschen vom Rechner kann im Textmodus erfolgen:

Control Byte:

X	X	X	0	1	0	0	0
0			3				8

Daten:

1. Byte	2. Byte		60. Byte
X'93'	X'00'		X'00'
0	1	2	59 60

Anhang B: Modul-Kurzbeschreibung

Modul
TCP1BEGN TCP-Initialisierung

CSECT TCP1BEGN

ENTRY TCP1BEGN, via ATTACH von System Task Control Routine (IEEPWILI) oder INITIATOR

EXIT Return to Tasksupervisor

FUNKTION

Open auf Library-DCB für TCP.LINKLIB
Setzt Key auf 1 (TCB, PSW)
ATTACH auf TCP1INIT
WAIT auf Ende von TCP1BASE oder Reply 'ENDTCP'
Wenn einer dieser Events eintritt, dann DETACH auf Subtask-TCB und Stoppen des TCPWTR
Return nach OS-Tasksupervisor

SPEICHERBEDARF

170_{16} Bytes

MODUL TCP2BEGN

CSECT TCP2BEGN

ENTRY via ATTACH von System Task Control Routine (IEEPWILI)

EINGANGSPARAMETER

R1 zeigt auf folgendes Fullword im Command Scheduling Control Block (CSCB):
"Pointer to EXEC Statement PARM-Field"

EXIT RETURN zum OS Task Supervisor

FUNKTION

TCP2BEGN wird generell als OS System Task gestartet (TCP03) und existiert parallel zum normalen TCP, das dann immer als Job läuft. Seine wesentliche Funktion besteht darin, einen CSCB zur Verfügung zu stellen, der es gestattet, auch dann mit SEND-Commands zu arbeiten, wenn TCP als Job läuft. Die Adresse des TCP2BEGN-CSCB's wird während der Initialisierung von TCP in die Konbox eingetragen, so daß ständig auf den CSCB zugegriffen werden kann (Operator-Command ECB wichtig für OPCP).

REGION 6 K

HINWEIS

Der Modul-Name 'TCP2BEGN' muß im Member IEEVLINKT (Link Table Module) der SYS1.LINKLIB eingetragen sein, damit TCP03 als System Task laufen kann.

Modul
TCP1INIT TCP - Initialisierung

CSECT TCP1INIT , **TCP1TSUP**

ENTRY TCP1INIT VIA ATTACH von TCP1BEGN

EXIT nach TCP1BASE VIA XCTL

FUNKTION

Baut SUCB (Startupcontrolblock) auf, gibt CIB (Command Input Buffer) in SP255 zurück und bearbeitet die Eingangsparameter (Pufferzahl und Länge). Controlsection TCP1TSUP übernimmt den Aufbau der I/O-Kontrollblöcke für die Terminals und den Aufbau der Lineboxes. Es folgt ein Test der in den DD-Statements beschriebenen Terminals. Nicht betriebsbereite Datenstationen werden in der Linebox markiert (LINEDOWN). Nach erfolgtem Terminal Startup XCTL nach Modul TCP1BASE. Übergebene Parameter: SUCB- und Library-DCB-Adresse.

SPEICHERBEDARF

1158_{16} Bytes

Modul	
TCP1LIBN	TCP Spooldataset und Library - Initialisierung
CSECT	TCP1LIBN
ENTRY	VIA LINK von TCP1BASE, CsectTCP1BASE
EXIT	RETURN nach TCP1BASE

FUNKTION

PARTDCB und SEQDCB in den Lineboxes werden aufgebaut. Open für SEQDCB's, WRITE mit Dummy-Record, NOTE und CLOSE Type T. Eingangsparameter ist die KONBOX-Adresse

SPEICHERBEDARF

280_{16} Bytes

Modul	
TCP1BASE	I/O Verwaltung für Terminals, TCP-Statusänderungen, Operatorcommand Bearbeitung, Datenausgabe.
CSECT	TCP1BASE TCP1HALT TCP1OPCP OUTSPOOL ENDPUT TCP1WAIT KONBOX
ENTRY	via XCTL von TCP1INIT Entrypoint TCP1BASE
EXIT	RETURN nach OS/Taskmanagement bei SHUTDOWN

FUNKTION

a) Csect TCP1BASE

KONBOX- und LINEBOX-Initialisierung wird beendet durch Vervollständigung der Kettung.

Der Bufferpool wird aufgebaut, ebenso die ECB-Listen. FREEMAIN auf SUCB, Aufruf von TCP1LIBN zur Libraryinitialisierung und ATTACH auf TCPOPROC. Initialisierung des CSCB für SEND-Commands. Der TCPWTR wird gestartet. Branch nach TCP1WAIT. Eingangsparameter: SUCB-Adresse

b) Csect TCP1HALT

HALTIO auf I/O Vorgang.

Eingangsparameter: IOB- und DCB-Adresse

Aufruf von OPCP bei anstehendem READ (Codeo1)-Command. RETURN nach OPCP.

CSECT c) TCP1OPCP (in TCP1BASE)

ENTRY via BRANCH von TCP1WAIT

EXIT via BRANCH nach TCP1WAIT oder
via BRANCH zu DYNOUT in TCP1WAIT

FUNKTION

Analysiert die Actionwords in den SEND-Kommandos, gibt dem Operateur Nachricht zurück und führt die entsprechende Routine aus.
mögliche Actionwords:

1.) USER: meldet die aktiven Benutzer
2.) **HOLD** ; **RELEASE** ; LOGIN; NOLOG; setzt TCP in den entsprechenden Modus
3.) **SHUTDOWN**; beendet TCP
4.) EIN: schaltet ein Terminal logisch an
5.) ID: gibt Nachricht an einen bestimmten Benutzer, wenn er aktiv ist, an das entsprechende Terminal, sonst auf Member M in seiner Bibliothek
6.) BROADCAST: Nachricht an alle on-line-Terminals
7.) ALL: wie 6. dazu Nachricht auf Member M in Bibliothek TCP.QQQ.USER
8.) WIM: wie 5. jedoch ohne Rückmeldung, (wird von TCPWTR benutzt)

MACROS REQBUF, RELBUF, GETMAIN, FREEMANN, LOCATE, OBTAIN, RDJFCB, FIND, NOTE, STOW, BLDL,

SPEICHERBEDARF

PROGRAMM-DATEN: DAO_{16} BYTES

WORKAREA in SUBPOOL: $7D8_{16}$ BYTES

d) CSECT OUTSPOOL

Aufruf von TCP1WAIT für dyn. Datenausgabe. OUTSPOOL überträgt den Inhalt des SPOOL-Datasets, eines PL/1- Ausgabe-Datasets oder eines Ausgabe-Datasets der SYSOUT=X Klasse in Ausgabepuffer für die Terminalausgabe. Jeweils 10 Records des auszugebenden Datasets werden auf diese Art bearbeitet; dann erfolgt ein Sprung nach EP. DYNOUT in TCP1WAIT zur Einleitung der Ausgabe am Terminal Nach beendeter Ausgabe am Terminal erfolgt der Rücksprung nach OUTSPOOL zur Übertragung der nächsten 10 Records, es sei denn, der Terminalbenutzer hat inzwischen durch Betätigung der Anruftaste zu erkennen gegeben, daß er keine weitere Ausgabe mehr wünscht, oder in OUTSPOOL war bereits Datenende oder eine Fehlerbedingung festgestellt worden. Im letzteren Falle erfolgt der Sprung nach beendeter Ausgabe am Terminal nach Csect ENDPUT.

Eingangsparameter: Konbox-Adresse Reg. 12, LBX-Adresse Reg. 6

e) CSECT ENDPUT

Bei Bibliotheks-Ausgabe CLOSE TYPE = T auf SPOOL-DS,
bei Programm-Ausgabe SCRATCH und UNCATALOG.

POST auf INECB zur Verständigung von TCPOPROC, daß Ausgabe beendet ist.

FREEMAIN auf Workarea.

Rücksprung nach TCP1WAIT.

f) CSECT TCP1WAIT

Entry von CSECT's TCP1BASE, TCP1OPCP, OUTSPOOL und ENDPUT via BR nach EP. TCP1WAIT bzw. nach Entrypoint DYNOUT

TCP1WAIT hat die Aufgabe:

1. Die Terminal - I/O-Vorgange zu initialisieren und ihren Ablauf zu überwachen.
2. Die Datenflüsse von den Terminals nach den zugeordneten Spool-Datasets und umgekehrt zu initialisieren und zu überwachen.
3. Die Bearbeitung von Systemnachrichten (SEND Commands) zu veranlassen.

4. Die Verfügbarkeit der Leitungen (LINES) zu kontrollieren.
   Der Eingang von TCP1WAIT leitet unmittelbar zu einem TWAIT in dem das Eintreten eines der folgenden Ereignisse abgewartet wird:

1. Der Entschlüsslerteil TCPOPROC wünscht die Einleitung eines I/O-Vorganges an einer Datenstation.
2. Ein I/O-Vorgang an einer Datenstation ist beendet.
3. Vom System wurde ein SEND-Command abgesetzt.
4. Der Entschlüsslerteil TCPOPROC hat seine Tätigkeit beendet (ABEND oder SHUTDOWN)

Im Falle 1. wird der Type-Code für den zu initialisierenden I/O-Vorgang ermittelt; der I/O-Vorgang selbst wird durch die BTAM I/O-Routine (IGG019MA) angestoßen.
Im Falle 2. wird geprüft, ob der Vorgang normal beendet worden ist. Wenn ja, dann wird TCPOPROC mittels POST auf INECB verständigt. Tritt ein Fehler auf, so werden maximal 10 Wiederholungen des I/O-Vorganges zur Behebung des Fehlers versucht. Mißlingt die versuchte Korrektur oder bricht die Verbindung zur Datenstation zusammen, so wird für die entsprechende Leitung LINEDOWN an TCPOPROC gemeldet (POST auf INECB). Der Systemoperator wird verständigt. Ist für eine bestimmte Datenstation SPOOLOUT aktiv, so geht die Kontrolle nach der Ausgabe jeweils eines Blocks wieder an Spoolout bzw. nach Ausgabe des letzten Blocks an ENDPUT über. ENDPUT verständigt dann TCPOPROC über die Beendigung der Datenausgabe (POST auf INECB)
Im Falle 3. wird nach TCP1OPCP zur Analyse des Commands verzweigt. **Sind** Nachrichten an eine Datenstation zu übertragen, so erfolgt von TCP1OPCP aus direkt ein Sprung nach DYNOUT in TCP1WAIT.Bei Beendigung eines solchen I/O-Vorgangs wird TCPOPROC nicht verständigt.
Im Falle 4. geht die Kontrolle über LINK an Modul TCP1TSDN zum Abschluß des SHUTDOWN - Vorganges.

MODUL
IGG019MA BTAM R/W-Routine

CSECT IGG 019 MA

ENTRY Wird in TCP1TSUP geladen mit LOAD ,dann BRANCH

EXIT RETURN zum aufrufenden Programm

SUBROUTINES

Request/Release Buffer Routine IGG 019 MS
Device I/O-Modul IGG 019 MO

FUNKTION

FINDS IOB und Device I/O-Modu .
Bereitet Channel Program in der IOB-Channel Program Area auf und gibt EXCP.

SPEICHERBEDARF

61 C_{HEX} Bytes 1,5 K Bytes

HINWEIS

Beschreibung siehe /7/.

MODUL
IGG019MO Device I/O-Modul für SIEMENS TRANSDATA 8525/101

CSECT IGG 019 MO

ENTRY/
EXIT Wird von TCP1TSUP geladen mit LOAD

FUNKTION

Enthält 4 Tabellen:

1) Relatives Displacement zu Entries in Tabelle CHANNEL PROGRAMS und zur Tabelle CCW's (C).
2) CHANNEL PROGRAMS enthält Information zu Selektion der entsprechenden CCW's:
   a) Anzahl der CCW's pro Channel Programm
   b) Nummern der CCW's und Sonstiges
3) CCW's: Modell-CCW's für die 8525 angelehnt an die IBM 2740 CCW's
4) CHARS: spezielle Characters und Code-Kombinationen

SPEICHERBEDARF

$D0_{HEX}$ Bytes = 208_{DEZ}Bytes

HINWEIS

Dieser Modul hat die Form des Device I/O-Moduls für das Terminal IBM 2740 WITH CHECKING; er wurde aber **inhaltlich** völlig auf die SIEMENS TRANSDATA 8525/101 umgestellt. Genauere Informationen bei /7/.

Modul	
TCPOPROC	Kommandoverarbeitung
CSECTS	
PROC	Steuerungsroutine
STEFAN	Entschlüssler
MESSAGE	Nachrichten-Assembler
ATER	Funktions-Task-Exit, Detach & Cleanup.
ENTRY	von TCP1BASE via ATTACH
EXIT	RETURN to TCP1BASE, wenn SHUTDOWN in KBX gesetzt.

COMMUNICATION mit TCP1BASE (TCP1WAIT):

PROC	TWAIT auf ELISTIN
PROC bzw. ATER	POST auf OUTECB

SUBROUTINES	TCP1DSUP via ATTACH, DETACH bei Shutdown oder bei vorzeitigem Ende des DKP. Im letzten Falle Restart.
	TCP1ACNT via LINK (logon/logoff/Accounting).
	alle Routinen zur Befehlsausführung, via ATTACH von PROC, insbesondere
	Library-Management , DETACH von ATER

FUNKTION

Übernimmt Benutzerkommandos aus dem Buffer-Pool, analysiert und delegiert die Ausführung an die entsprechende Routine. Stellt Nachrichten an den Benutzer im Bufferpool zusammen.

SPEICHERBEDARF

14 K

MODUL TCPOPROC (bei ASP-Version)

CSECT STARTPRO

ENTRY Von CSECT PROC (TCPOPROC) via LINK

EINGANGSPARAMETER

LBXREG, KONBOXREG

EXIT RETURN nach PROC

Code 0 - alles OK
Code 111 - BLDL-Liste nicht vollständig gefüllt
Code 112 - Parameter I/O-Error auf USER-Library
Code 113 - Art ist keine JCL

FUNKTION

Holt mit BLDL die restliche Informationen über das Member der User-Library, das als Job zu starten ist; initialisiert das IJPWTR-Parameterfeld und ruft den IJPWTR mittels ATTACH auf. Anschließend RETURN nach PROC.

LÄNGE 244_{HEX} Bytes

MODUL TCP1SPOL

CSECTS FORMI, TCP1TAB

ENTRY von TCP1WAIT via ATTACH

EXIT via RETURN zu OS TASK Management

FUNKTION

TCP1SPOL

Übersetzen und Prüfen auf Format und Sonderzeichen der von TCP1WAIT in dyn. Puffern gelieferten Daten. Ausgabe der Daten in 80 Byte-Blöcken (evtl. num.) auf Zwischenspeicher (Magnetplatte) Prüfung auf TIMEOUT, Rückgabe bearbeiteter Puffer an den Bufferpool.

FORMI Übernimmt bei Änderungseingabe Suchen und formatgerechtes Einfügen der Zeilennumerierung.

TCP1TAB Liefert formatgerechte Schreibweise bei Eingabe von Assemblerprogrammen.

SPEICHERBEDARF

1670_{16} Bytes

MODUL	TCP1WTR auf SYS1.LINKLIB oder TCP.LINKLIB Startup-Module für TCPWTR
CSECT	TCP1WTR
ENTRY	via START TCPWTR, EXEC TCP1WTR
EXIT	via RETURN zum OS

FUNKTION

Prüft, ob TCPO aktiv ist, wenn ja, Kontrolle via LINK an TCP1WTRA auf TCP.LINKLIB

MACROS	SAVE, RETURN, OPEN, WTO, LINK

SPEICHERBEDARF

Programm + Daten 170_{16} Bytes

MODUL	TCP1WTRA	auf TCP.LINKLIB
CSECT	TCP1WTRA	
ENTRY	via LINK	von TCP1WTR
EXIT	via BRANCH ZU LINKOR	

FUNKTION

Holt Arbeitsspeicher, lädt Register für DSECTS

Abschnitt TCPWAIT: Prüft nach, ob in angegebener Klasse Output zu bearbeiten ist, sonst WAIT

TCPWALL: Besorgt Platz für SYSTEMMESSAGES, wenn MSGCLASS=X, baut Namen des Datasets und katalogisiert

TCPWIO : liest DSB/SMB's schreibt SMB's auf SYSTEMMESSAGE - DATASET, benennt DSB um , gibt Nachricht an Benutzer via SENDWIM,behandelt kritische Situationen.

TCPWTO : Fehlerauswertung, Nachricht an Operator

TCPWEOJ: schließt die Aktivität für Ausführung der Jobausgabe ab.

MACROS GETMAIN, FREEMAIN, WTO, LINK, EXTRACT, LOCATE, CATALOG, OPEN, SVC32, RENAME, OBTAIN, UNCATALOG

SUBROUTINEN TCP1LIST (auf TCP.LINKLIB), IEFQMDQQ, IEFQMRAW, IEFQMUNC, IEFQDELE, IEFQMNO2

SPEICHERBEDARF

Programm + Konstanten: $E00_{16}$ BYTES

WORKAREA in SUBPOOL 0: 590_{16} BYTES

MODUL	TCP1ACNT	Logon / Logoff und Accounting
CSECTS	LOGON	Programm
	LIST	Liste der Benutzer Codes.
ENTRY	von TCPPROC via LINK	
EXIT	nach TCPOPROC	

FUNKTION

Wird bei LOGON angesprungen, macht OPEN auf die Benutzer-Library, setzt Times in Betrieb. Wird bei Logoff angesprungen, macht CLOSE auf die Benutzer-Lib, berechnet die verbrauchte CPU-Zeit und schreibt sie ins Log.

SPEICHERBEDARF

2 K

MODUL TCP1LIST

Transformiert die RZ-Benutzernummer in die TCP-Benutzeridentifikation.

CSECT 1) LIST : enthält Übersetzungstabelle

2) TCP1LIST: enthält Zuordnungsalgorithmus

ENTRY TCP1LIST via LINK

EXIT Return to caller

MACROS SAVE, GETMAIN, FREEMAIN, RETURN

ATTRIBUT RENT

SPEICHERBEDARF

CSECT LIST	2000 Bytes
CSECT TCP1LIST	100 Bytes
	2100 Bytes

EINGABE

Adresse eines 3 Byte-Feldes in Reg.1 .

Feld: Z3 Z2 Z1

Adr.

Z3Z2Z1 = RZ-Benutzernummer in EBCDIC

AUSGABE

Im Eingabefeld steht die zugeordnete TCP-Benutzeridentifikation in EBCDIC

z.B. C1 D7 D6 APO

Die Codierung erfolgt über einen festen Algorithmus und eine Codierungstabelle. Bereich der Eingabe-RZ-Nummern: 000 - 999

Achtung: 000 gilt als nicht erlaubt!

Anwendung: Modul TCP1WTRA, Macro WTT,

F-Subroutine WTT

MODUL TCP1LIST: Codierung von RZ-Nr. in TCP-ID

	Ablauf:			Beispiel
Eingabe:	Z_0Z_3 Z_0Z_2 Z_0Z_1	Z_0	Zone	F3 F7 F9
	Adr.			
packen:	$Z_3Z_2Z_1$ +			379 C
CUB	$....B_4B_3B_2B_1$			17 B hex
2 x CUB =	.....			2F6 hex
= Displ. D				

LIST

aus List mit	+ 0		
Displacement D	+ 2		
Codezahl CZ	+ 4		
	.		
	.		
3 x CZ	+ 2F6	5_{Bin}	CZ
	2F8		

CZ_4 CZ_3 CZ_2 CZ_1	05	0000 0000 0000 0101
Tetraden		Bin

3 x CZ = 0F 0000 0000 0000 1111

mit Displacement CC (Code Character) 1.CC 2.CC 3.CC

in USITABLE

USITABLE

CCP	+ 0	C	1.CC = 0 C
	+ 1	.	2.CC = 0 C
Achtung: Freie Zuord-			3.CC = F P
nung in LIST!	.	.	
	.	.	
	+ F	P	

MODUL CØMPRESS

CSECT CØMPRESS

ENTRY via ATTACH von TCPØPRØC

EXIT via RETURN zu ØS Task Management

FUNKTION

Die Datei des gerade aktiven Terminalbenutzers wird reorganisiert. Die Member werden in aufsteigenden Spuradressen und in alphabetischer Reihenfolge in eine neue dynamisch generierte Bibliothek kopiert. Ein **Inhaltsverzeichnis** wird angelegt, wobei irrelevante Einträge des früheren Inhaltsverzeichnisses (Name mit <8XL40) unberücksichtigt bleiben.
Baut Nachricht für den Benutzer auf, wenn I/O-Fehler beim Lesen seiner Bibliothek auftreten. Setzt Fehlercodes ab und versucht bei Fehlerbedingungen den ursprünglichen Zustand der Bibliothek wieder herzustellen (siehe nähere Beschreibung des Befehls C).

SPEICHERBEDARF

$8D2_{16}$ Byte = 2258_{DEZ}

MODUL TCP1LOUT LO,# YYYYYAA,EE

CSECT LOEOUT

ENTRY via ATTACH vom TCPØPROC

EXIT Return zum TCPØPROC

EINGANGSPARAMETER

AUSGANGSPARAMETER

Returncode in LINEBOX + 27

FUNKTION

Die Ausgaben eines gestarteten Jobs mit CLASS = X, wird gestrichen. Ist AA = EE, so wird eine einzige Ausgabe gelöscht. Sonst werden alle Ausgaben, die die Endnummer AA bis EE haben gelöscht.

ATTRIBUT RENT

SPEICHERBEDARF

Hex.

CSECT 23A ≙ 570 ≈ 0.5 K Byte

DSECT 230 ≙ 560 ≈ 0.5 K Byte

MODUL TCP1ERK Erk, Name

CSECT ERKLAERE

ENTRY via ATTACH vom TCPOPROC

EXIT Return zum TCPOPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

Returncode in LINEBOX + 27 (HEX.)

FUNKTION

Das Module stellt anhand vom WCode 1 fest, welches Member aus dem Data Set "QQQ" verlangt wurde, bringt den entsprechenden Membernamen "ERTEXTnn" in LINEBOX und LINK't die Schreibe-Routine zur Ausgabe des Members auf dem Drucker.

ATTRIBUT RENT

SPEICHERBEDARF

HEX.

CSECT 162 $\hat{=}$ 354 Byte $\approx$ 0.3 K

DSECT C8 $\hat{=}$ 200 Byte $\approx$ 0.2 K

MODUL TCP1INP1

CSECT UMORG1

ENTRY via ATTACH vom TCPOPROC

EXIT Return zum TCPOPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

Returncode in LINEBOX + 27 (HEX.)

FUNKTION

Dieser Module ist eine Vorroutine für alle Befehle, die mit Dateneingabe verbunden sind. Es sind:
lies-Mod,..., z-Mod,...k-Mod, ≠ D. Der Module prüft den Membernamen ab bei "lies", prüft, ob die Parameter richtig angegeben sind und sich nicht widersprechen und **speichert schließlich** in LCC12 (LBX + 27) **einen** Returncode ein, damit eine der folgenden Nachrichten ausgegeben wird.

1. Bitte geben Sie die Daten ein.
2. Bitte geben Sie die Änderungen ein.
3. Name, Art, Format, Code

ATTRIBUT RENT

SPEICHERBEDARF

HEX.

CSECT 272 = 626 Byte 0.6 K

DSECT AO = 160 Byte

MODUL TCP1LGES LO,Name,a

CSECT STRMEMB

ENTRY via ATTACH vom TCPØPROC

EXIT Return zum TCPØPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

Returncode in LINEBOX + 27 (HEX)

FUNKTION

Der Eintrag für das angegebene Member wird aus dem Directory entfernt. Damit ist das Member nicht mehr verfügbar.

ATTRIBUT RENT

SPEICHERBEDARF

(HEX)

CSECT CA $\hat{=}$ 202 Byte $\approx$ 0.2 K

DSECT 50 $\hat{=}$ 80 Byte

MODUL TCP1AEN2 Mod, Name, Z

CSECT AENDER

ENTRY via ATTACH vom TCPØPROC

EXIT RETURN zum TCPØPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

H'0' oder Fehlercode in LINEBOX + 27 (HEX)

FUNKTION

Die an der Tastatur eingegebenen, mit aufsteigenden Nummern versehenen Zeilen ersetzen die Zeilen des angegebenen Members oder werden zwischen zwei Zeilen eingeschoben, und zwar:

1. Hat die eingegebene Zeile die gleiche Nummer wie eine Zeile im Member, so wird diese ersetzt.
2. Hat die eingegebene Zeile eine Nr. die zwischen zwei Zeilen-Nr. im Member liegt, so wird die eingegebene Zeile eingeschoben.

ATTRIBUT RENT

SPEICHERBEDARF

HEX

CSECT 578 ≙ 1400 Byte ≈ 1,4 K

DSECT E30 ≙ 3632 Byte ≈ 3,5 K

MODUL TCP1BC20 Kette,Name1,Name2,Neuname

CSECT KETTE

ENTRY via ATTACH von TCPØPROC

EXIT RETURN zum TCPØPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

H'0' oder Fehlercode in LINEBOX + 27
(HEX)

FUNKTION

Die beiden Member mit den Membernamen 'Name1' und 'Name2' werden in der angegebenen Reihenfolge zusammengekettet, und unter dem Membernamen 'Neuname' in Benutzerbibliothek abgelegt.
Ist das erste Member numeriert und in dem Inhaltsverzeichnis mit dem Attribut 'NUM' versehen, so wird das neue Member numeriert. Sonst werden die beiden Member ohne Änderung in Spalte 73-80 gekettet.

ATTRIBUT RENT

SPEICHERBEDARF

CSECT 41A (HEX) = 1050 Byte $\approx$ 1 K
DSECT DC8 (HEX) = 3528 Byte $\approx$ 3,4 K

MODUL TCP1MODR Mod,Name,K

CSECT MODREC

ENTRY via ATTACH vom TCPØPROC

EXIT RETURN zum TCPØPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

H'0' oder Fehlercode in LINEBOX + 27 (HEX)

FUNKTION

Innerhalb einer Zeile werden einzelne Zeichen oder Zeichengruppen geändert, eingeschoben oder weggenommen.

EINGABEFORM

mm xxx .. nnyyy...# Z

mm = Anzahl d. Zeichenkette nn xxx...

nn = Anzahl der Zeichenkette nn yyy...,
die xxx... ersetzen soll

Z = Zeilennummer

mm und nn müssen als zweistellige Zahlen angegeben werden. Eine einstellige Zahl wird mit om oder on angegeben. nn und mm dürfen ungleich sein.

nn > mm = einschieben vom Zeichen

nn < mm = wegnehmen vom Zeichen

Beim Einschieben muß darauf geachtet werden, daß eine Zeile nicht über Spalte 72 hinausgeht.

Es können mehrere Änderungen aufeinmal vorgenommen werden.

ATTRIBUT RENT

SPEICHERBEDARF

HEX

CSECT 66C = 1644 Byte ≈ 1,5 K

DSECT 7E0 = 2016 Byte ≈ 2 K

MODUL TCP1DMOD Mod, ## D

CSECT UTIDIR

ENTRY via ATTACH vom TCPØPROC

EXIT RETURN zum TCPØPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

Returncode in LINEBOX + 27 (HEX)

FUNKTION

Directory eines Members wird gemäß der Angaben am Terminal geändert. Folgende Parameter können geändert werden:

1. ART: ICL,dat,PL1,for,ass,alg
2. CODE:ISO,Ptt,bin,ka5
3. FORMAT:num,onu+eventuell Datenformat

ATTRIBUT RENT

SPEICHERBEDARF

CSECT 49C = 1180 Byte ≈ 1,1 K

DSECT 7C8 = 1992 Byte ≈ 1,9 K

MODUL TCP1HOL Hol,neuer Name DSN { 0 → SDS ; Membername → PDS }

CSECT HOLE

ENTRY via ATTACH vom TCPOPROC

EXIT RETURN zum TCPOPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

Returncode in LINEBOX + 27 (HEX)

FUNKTION

Der Modul bringt ein Member aus einem "Partioned Data Set" oder einen Seq. Data Set in die Benutzerbibliothek und speichert es unter dem Namen "Neuer Name" ab.
Falls die Daten aus einem Seq. Data Set geholt werden, wird im Directory ART = dat, FORMAT = ONU, CODE = ISO eingetragen. Sonst sind diese Angaben identisch mit den Angaben des Originalmembers.

ATTRIBUT RENT

SPEICHERBEDARF

	(HEX)	(DEC)	
CSECT	680	= 1664 Byte	≈ 1,6 K
DSECT	7E0	= 2016 Bytw	≈ 2 K

MODUL TCP1LREC Lo ,Name, A1.E1.A2.E2.....

CSECT STRREC

ENTRY via ATTACH vom TCPØPROC

EXIT Return zum TCPØPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

Returncode in LINEBOX + 27

FUNKTION

Es werden Zeilengruppen mit den angegebenen Anfangs- und Endnummern:

A1 E1, A2 E2 .....

aus dem Member gestrichen. Max. Stellenzahl für die Zeilenangabe "A1.E1.A2.E2".. ist 40. Die Nummern A1,E1, A2,E2.... müssen in aufsteigender Reihenfolge angegeben werden.

Ist A1 = E1, wird nur eine einzige Zeile gestrichen.

ATTRIBUT RENT

SPEICHERBEDARF

	HEX.		DEC.
CSECT	66C	=	1644 ≈ 1.6 K
DSECT	EB0	=	3760 ≈ 3.6 K

MODUL TCP1TS Zl , Name,xx.yy

CSECT TEILSCH

ENTRY via ATTACH vom TCPØPROC

EXIT Return zum TCPØPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

Returncode in LINEBOX + 27 (HEX)

FUNKTION

Von dem angegebenen Member werden alle Zeilen zwischen xx und yy einschließlich xx und yy gedruckt. Das Member muß numeriert sein und als Format das Attribut 'Num' aufweisen.

ATTRIBUT RENT

SPEICHERBEDARF

HEX.

CSECT 3F0 = 1008 Byte ≈ 1 K

DSECT 6E8 = 1768 Byte ≈ 1.7 K

MODUL TCP1LIE2 L,f,Name,Art,Format, Code

CSECT UMORG2

ENTRY via ATTACH vom TCPØPROC

EXIT Return zum TCPOPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

Returncode in LINEBOX + 27 (HEX)

FUNKTION

Die eingegebenen Daten an Tastatur oder durch Lochstreifenleser werden in Blöcken von 1600 Byte Länge in Benutzerbibliothek gebracht und anschließend durch STOW ein Directory Eintrag für das angegebene Member erzeugt.
Der erzeugte Eintrag enthält alle Parameter, die der Benutzer angegeben hat (Name, Art, Format, Code) plus Länge des Members in 1600 Byte-Blöcken und das Datum.

ATTRIBUT RENT

SPEICHERBEDARF

	HEX.		
CSECT	344	=	836 Byte ≈ 0,8 K
DSECT	720	=	1824 Byte ≈ 1,8 K

MODUL TCP1ZUS Z

CSECT DIRZUST

ENTRY via ATTACH vom TCPØPROC

EXIT Return zum TCPØPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

Returncode in LINEBOX + 27 (HEX)

FUNKTION

Inhaltsverzeichnis (Directory) der Benutzerbibliothek wird am Terminal ausgedruckt,
jeder Eintrag besteht aus folgenden Informationen:

1. Membername
2. TT R: Adr. des Members in Bibliothek
3. Länge des Members in 1600 - Byte Blöcken
4. Erzeugungsdatum
5. Änderungsdatum, wenn das Member einmal modifiziert wurde.
6. ART des Members
7. Code
8. Format

Wenn das Member nicht über Terminal eingespielt wird, so fehlen die Informationen 3 bis 8.

ATTRIBUT RENT

SPEICHERBEDARF

	HEX.		
CSECT	4C6	= 1222 Byte	≈ 1.2 K
DSECT	3B0	= 944 Byte	≈ 9 K

MODUL TCP1SCH S,f,Name

CSECT RUECKORG

ENTRY via ATTACH vom TCPØPROC

EXIT Return zum TCPØPROC

EINGANGSPARAMETER

R3 = A (LINEBOX)

AUSGANGSPARAMETER

Returncode in LINEBOX + 27

FUNKTION

Das angegebene Member wird ganz ausgedruckt

ATTRIBUT RENT

SPEICHERBEDARF

	HEX.	
CSECT	21A	= 538 Byte ≈ 0.5 K
DSECT	6B8	= 1720 Byte ≈ 1.7 K

MODUL XIAK12

ENTRY von TCP1SPRO via LINK

EXIT nach TCP1SPRO

FUNKTION

Daten (als Member eines Part. Datensatzes auf Platte) eines bestimmten Formates werden in Binärzahlen (4Byte) umgewandelt und wieder unter dem gleichen Membernamen abgespeichert.

SPEICHERBEDARF

ECE_{16} Bytes

MODUL XIAK24

ENTRY TCP1SPRO

EXIT nach TCP1SPRO

FUNKTION

Wie XIAK12 nur anderes Eingabeformat der Daten.

SPEICHERBEDARF

$3A8_{16}$ Bytes

MODUL IJPWTR Internal Job Processor Writer

CSECT IJPWTR

ENTRY Von CSECT STARTPRO (TCPOPROC) via ATTACH

EINGANGSPARAMETER

R1 zeigt auf Parameter-Liste (s. ausführliche Beschreibung des Moduls IJPWTR)

EXIT RETURN nach TCPOPROC, CSECT ATER.

FUNKTION

IJPWTR stellt das TCP-ASP Interface dar. Er liest das als Job zu startende Member der TCP-Benutzerbibliothek und schreibt es über den CTC (Channel To Channel) um es schließlich vom IJP weiter bearbeiten zu lassen. Danach gibt der IJPWTR die Kontrolle an ATER (TCPOPROC) zurück.

HINWEIS

- Der IJPWTR ist nach OS-Konvention programmiert.
- Damit der IJPWTR funktionieren kann, muß auf dem selben Processor an dem TCP läuft, gleichzeitig das ASP-DSP (Dynamic Support Program) IJP aktiv sein.
- Der ins TCP eingebaute IJPWTR wurde für die besonderen TCP-Anforderungen geändert (s.ausführliche Beschreibung).

LÄNGE $D30_{HEX}$ Bytes

MODUL TCP1DSUP Display-Startup-Modul

CSECT TCP1DSUP

ENTRY via ATTACH von TCPOPROC
CSECT PROC

EXIT Normal : XCTL auf DKP
Abnormal: RETURN zum OS

FUNKTION

TCP1DSUP initialisiert die für Display-Ausgabe notwendigen I/O-Blöcke wie DEB, IOB und die Display-Boxes DBX. Die Display-Line-Zuordnungstabelle TCP1DTAB wird geladen.
Display-Startbilddaten werden eingelesen (GET, INDATA über DDNAME = IDD); das Testbild wird an alle Displays ausgegeben, anschließend XCTL zu DKP.

SPEICHERBEDARF

$7F8_{HEX}$ Bytes ≈ 2 K Bytes

MODUL DKP Display - Kontroll - Programm

CSECT TCP1DKP

ENTRY via XCTL von TCP1DSUP

EXIT RETURN zum OS, wenn SHUTDOWN eingeleitet wird

COMMUNICATION mit TCPOPROC

WAIT auf DISPECB
POST auf DISPECB2

SUBROUTINES

TCPTAUS via LINK
TCPKAUS via LINK
TCPPAUS via LINK
DKPV1 via LINK
DKPV2 via LINK
DKPV3 via LINK

FUNKTION

Entschlüsselt den Typ des Display-Befehls und delegiert die Befehlsausführung an die entsprechende Routine. Sequentialisiert bei graphischer Ausgabe die Aufbereitungs- und Ausgaberoutinen. Stellt 16K-Ausgabefeld zur Verfügung.

SPEICHERBEDARF

$4A10_{HEX}$ Bytes $\approx$ 18,5 K Bytes

TCPTAUS (ASS)

CSECT TEXTAUS, TCP1ZUS, CANCANAS

ENTRY via LINK von DKP

EXIT RETURN zum DKP

SUBROUTINES

TCP1ZUS via CALL

CANCANAS via CALL

FUNKTION

Daten einlesen für Text- und Show-Ausgabe;
Zustand der Bibliothek

SPEICHERBEDARF

6 K Bytes

HINWEIS

Dieser Modul wird bei folgenden TCP-Befehlen benötigt:

dz

dt, name

de, name, t, nnn

dt

d-

da

de, 'TEXT', nn.xx

dd, 'TEXT', nn.xx

dd, name, t, nnn

de
, name, s

dd

TCPKAUS (ASS)

CSECT KURVAUS

ENTRY via LINK von DKP

EXIT RETURN zum DKP

SUBROUTINES

DKPUM via LINK

FUNKTION

Einlesen der Daten für graphische Displayausgaben

SPEICHERBEDARF

1,7 K Bytes

HINWEIS

Dieser Modul wird bei den folgenden TCP-Befehlen benötigt:

de , name, dd	a g am, ... gm, ... k, ...

DKPUM (PL/1)

ENTRY via LINK von TCPKAUS

EXIT RETURN zu TCPKAUS

SUBROUTINES

keine

FUNKTION

Sucht aus dem von TCPKAUS eingelesenen Feld alle Zahlenwerte aus und wandelt sie in FLOATING-POINT-Zahlen um. Beim Automatischen-Modus werden die fehlenden X-Werte erzeugt. Insgesamt können 4000 Zahlenwerte verarbeitet werden.

SPEICHERBEDARF

5748_{HEX} Bytes 22 K Bytes

HINWEIS

Dieser Modul wird bei den folgenden TCP-Befehlen benötigt:

de g
dd ,name, a
gm, ...
am, ...
k, ...

DKPV1 (PL/1)

ENTRY via LINK von DKP

EXIT RETURN zu DKP

SUBROUTINES

CANCAN via CALL

FUNKTION

Falls erforderlich wird durch einen CANCAN-Aufruf der Bildschirm gelöscht. Die eingelesenen Daten werden auf den gewünschten Maßstab umgerechnet und auf die entsprechenden Randbedingungen abgeprüft. Das auszugebende Bild wird wechselweise in zwei verschiedenen Ausgabebereichen erstellt. Während der Ausgabe aus dem einen Ausgabepuffer, wird in dem anderen ein neues Teilbild erstellt. Es erfolgt ebenfalls die Aufbereitung und Ausgabe von Maßstabslinien und Beschriftung.

SPEICHERBEDARF

$74E0_{Hex}$ Bytes 30 K Bytes

HINWEIS

Dieser Modul wird bei den folgenden TCP-Befehlen benötigt:

de
dd ,name, g
a
gm,...
am,...

DKPV2 (PL/1)

ENTRY via LINK von DKP

EXIT RETURN zu DKP

SUBROUTINES

CANCAN via CALL

FUNKTION

siehe Beschreibung DKPV1

SPEICHERBEDARF

$74F8_{HEX}$ Bytes 30 K Bytes

HINWEIS

Dieser Modul wird bei den folgenden TCP-Befehlen benötigt:

de
dd , name, k, ...

DKPV3 (PL/1)

ENTRY, EXIT, SUBROUT

wie DKPV2

FUNKTION

siehe Beschreibung DKPV1

SPEICHERBEDARF

30 K

HINWEIS

Datenausgabe erfolgt im Logarithmischen Maßstab

TCPPAUS (ASS)

CSECT PLOTAUS

ENTRY via LINK von DKP

EXIT RETURN zum DKP

SUBROUTINES

PLOTA1 via LINK

PLOTA2 via LINK

PLOTA3 via LINK

FUNKTION

Einlesen der Plot-Daten und Suchen nach der richtigen A-Karte

SPEICHERBEDARF

2 K Bytes

HINWEIS

Dieser Modul wird bei den folgenden TCP-Befehlen benötigt:

de
dd , name, p, nn.x

d+
d- nach vorangegangener Plotausgabe

PLØTA1 (PL/1)

ENTRY via LINK von TCPPAUS

EXIT RETURN zu TCPPAUS

SUBROUTINE

CANCAN

FUNKTION

Verarbeitung der A, B, C-Karten
Ausgabe der Koordinaten-Linien

SPEICHERBEDARF

6218_{HEX} Bytes 25 K Bytes

HINWEIS

Dieser Modul wird bei Plotausgabe auf das Display benötigt

```
de
dd    ,name,p,nn.x
```

PLØTA2 (PL/1)

ENTRY, EXIT, SUBROUTINE

wie oben

FUNKTION

Ausgabe der Beschriftung von Plotbildern

SPEICHERBEDARF

6420_{HEX} Bytes 26 K Bytes

PLØTA3 (PL/1)

ENTRY, EXIT, SUBROUTINE

wie oben

FUNKTION

Kurvenausgabe von Plotbildern
(D-Karten)

SPEICHERBEDARF

$5FF0_{HEX}$ Bytes 25 K Bytes

MODUL TCP1TSDN

CSECT TCP1TSDN

ENTRY via LINK von TCP1BASE (CSECT TCP1WAIT)

EXIT RETURN nach TCP1WAIT in TCP1BASE

FUNKTION

Beendigung der SHUTDOWN - PHASE. Auf anstehende PREPARE - Commands wird HALT I/O gegeben. Nach dem Ausdrucken der Line-Error-Counts am Operator-Konsol erfolgt ein CLOSE für alle SPOOL-Datasets. Anschließend DELETE für die geladenen BTAM-Routinen. RETURN nach TCP1WAIT.

SPEICHERBEDARF

190_{16} Bytes

Anhang C

TCP - Makros und - Kontrollblöcke

In den TCP-Assemblerprogrammen werden neben **den** allgemein bekannten IBM 360-Assembler-Makros (SYS1.MACLIB) folgende zusätzliche Makros verwendet, die zum Teil eigens für TCP codiert **wurden** (TCP.MACLIB):

ALEPH
COMMAND
CVT
DDBX
DEBD
DKBX
DLBX
HE
IECDSECT
IEECHAIN
IEECUCM
IEECVMUG
IEEXSA
IEFJFCBN
IEFQMNGR
IEFTIOT1
IEFUCBOB
IOBD
MVL
OPT
SIETRTAB
SUCB
WTP
WTT

Alle Makros, die speziell für TCP Codiert wurden, werden im folgendem beschrieben.

ALEPH

Dieses Makro dient zur Anfangsinitialisierung eines Programmes (CSECT) und zur Zuordnung eines Arbeitsbereiches (WØRKAREA). Für die WØRKAREA wird Register 13 als Basis-Adressregister verwendet.

FØRM

```
symbol  ALEPH  BASEREG = register    , SPN = subpool nr.
```

symbol : Ein in der Assembler Language gültiger Name.
register : Ein Register, das als Basisregister verwendet werden soll.
subpool nr: Nummer des subpools aus dem der Speicherbereich für die WØRKAREA zugewiesen werden soll.
Wird dieser Parameter weggelassen, so wird SPN = 0 angenommen.

Wird das Makro ALEPH verwendet, so muß im Programm eine DSECT für den Arbeitsbereich vereinbart sein. Die DSECT muß mit einer Doppelwortgrenze beginnen und enden. Der Anfangsname ist WØRKAREA, der Endname ist WØRKEND. Die ersten 18 F sind für die Savearea reserviert.

Form:

```
WORKAREA   DSECT
SAVEAREA   DS          9 D
    .       .
    .       .
    .       .
WØRKEND    DS          0 D
```

FUNKTION

Das Makro ALEPH veranlaßt zuerst mit dem Makro SAVE (14, 12), das Retten der Register.
Anschließend wird mit einem USING *, & BASEREG das bei BASEREG = angegebene Register als Basisadressregister verwendbar gemacht.
Weiter wird mit einem GETMAIN Makro der Speicherplatz aus dem gewünschten Subpool für die WØRKAREA zugeordnet. Die Länge der WØRKAREA wird vor der Ausführung des GETMAIN-Makros berechnet.
Nach der Speicherplatzzuordnung steht im Register 1 die Adresse des Speicherplatzes und damit auch die der Savearea.
Nun werden die Savearea-Adressen gespeichert, die Adresse der alten Savearea in die neue Savearea +4 und die der neuen in die alte Savearea +8.
Aus der alten Savearea wird der Inhalt des Registers 1 ins Register 15 gerettet.
Die Adresse der Savearea und WØRKAREA wird aus dem Register 1 in das Register 13 umgeladen und mit einem USING WØRKAREA, 13 dieses Registers auch als Adressregister für die WØRKAREA erklärt.
Zum Schluß wird Register 15 in das Register 1 geladen, so daß im Register 1 wieder derselbe Wert steht wie bei Eintritt in das Programm.
Das Makro ALEPH soll in Verbindung mit dem Makro HE verwendet werden.

Das Makro DDBX

Schreibweise:

```
name    DDBX
```

Das Makro DDBX generiert eine Display-Box-DSECT.
Jeder Display-Control-Unit ist eine DBX zugeordnet.
Sie wird im Modul TCP1DSUP aufgebaut und enthält
Display-Informationen. (Beschreibung siehe Bild 9)

Bild 9

DISPLAYBOX

DEZ	HEX	
0	0	DBXCHAIN
4	4	DBXUA
8	8	DBXECB
12	C	DBXCP
16	10	DBXCOUNT | DBXDNR
20	14	DBXCBLI
24	18	IOBFLAG1 | IOBFLAG2 | IOBSENSO | IOBSENS1
28	1C	IOBECBCC | IOBECBPT
32	20	IOBFLAG3 | IOBCSW
36	24	
40	28	IOBSIOCC | IOBSTART
44	2C	IOBDCBPT
48	30	OPCODE | IOBRESTR
52	34	IOBINCAM | IOBERRCT
56	38	IOBUCBX |
60	3C	STIND | IOBNXTPT
64	40	IOBCCW
112	70	DBXAPP
132	84	DCBTIOT | DCBMACRF
136	88	DCBIFLGS | DCBDEBAD
140	8C	DCBOFLGS |

OFFSET	LÄNGE (BYTE)	NAME	BEDEUTUNG
0	4	DBXCHAIN	Pointer zur nächsten Displaybox
4	4	DBXUA	Adresse der Control Unit
8	4	DBXECB	ECB für Displayausgabe. Wird vom I/O-Supervisor gepostet, wenn die Ausgabe zu Ende ist.
C	4	DBXCP	Pointer zum Kanalprogramm
10	2	DBXCOUNT	Anzahl aktiver Kanalprogramme
18	2	DBXDNR	Anzahl der Displays pro Control Unit
20	6C	IOB	Siehe /8/
84-28	34	DCB	Data Control Block
84	4	DCBTIOT	Offset zum DD-Eintrag in der TIOT
88	1	DCBIFLGS	IOS Error Flags
89	3	DCBDEBAD	Adresse zum DEB
90	1	DCBOFLGS	Flags for Open

Das Makro DEBD

Schreibweise:

name DEBD

Das Makro DEBD kreiert eine Data Extent Block (DEB) - DSECT für einen BTAM-DEB.
Die Bedeutung der einzelnen Felder ist in /7/ beschrieben.

Das Makro DKBX

Schreibweise:

name DKBX TYPE = DSECT / CSECT

Wird das Keyword TYPE weggelassen, dann gilt DSECT.

Funktion:

a) TYPE = CSECT
   Die Konbox wird als CSECT angelegt.

b) TYPE = DSECT
   Eine DSECT der Konbox wird angelegt.

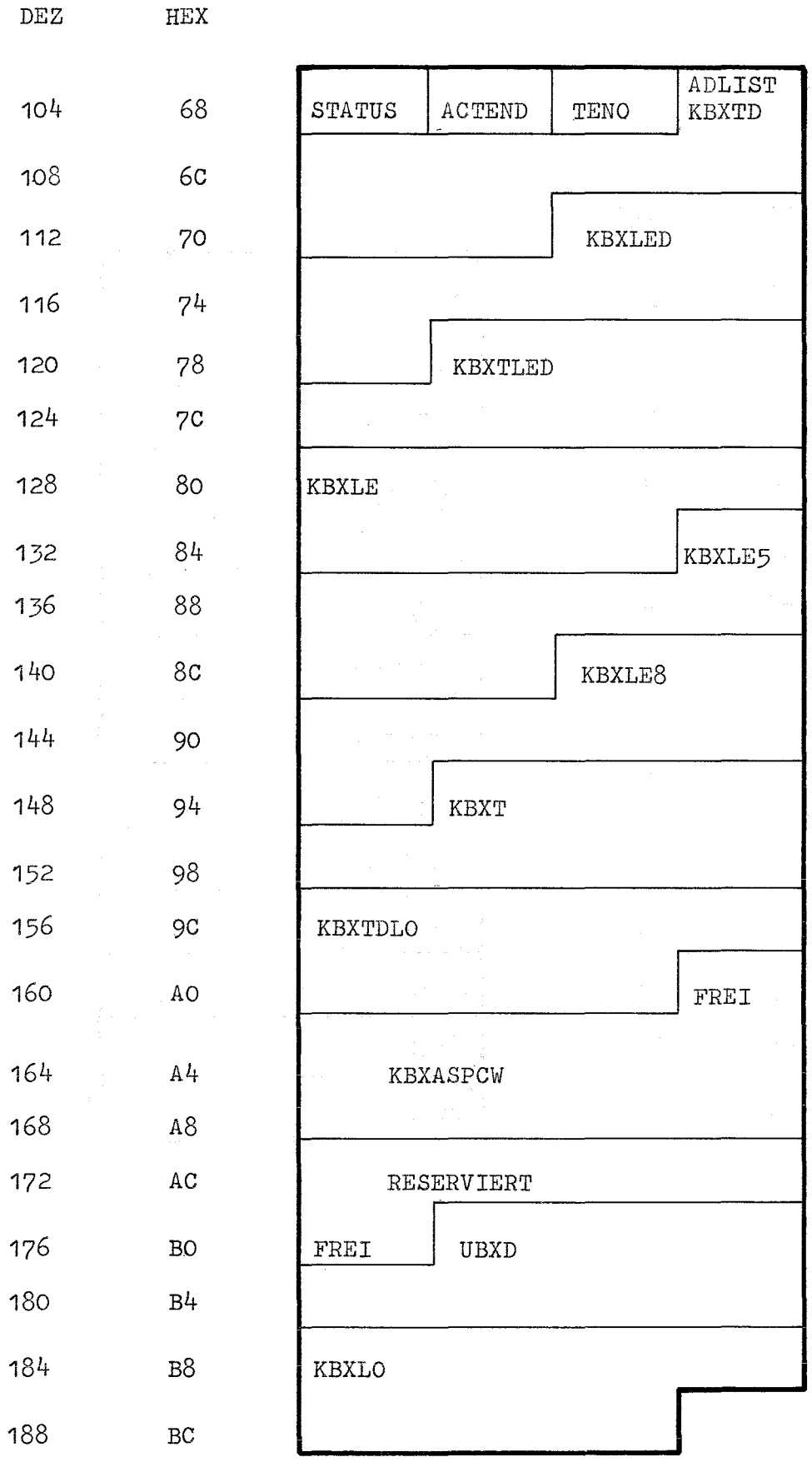
Die Konbox ist der zentrale Kontrollblock für TCP. Sie enthält Pointer zur Routinen und Kontrollblöcken und Selektionszeichen zur Steuerung der I/O-Geräte an den Datenstationen.

Aufbau und Beschreibung der Konbox siehe Bild 10.

Bild 10

DEZ	HEX	
0	0	KBXID | LB1EP
4	4	WAITEP
8	8	DISAUSFD
12	C	
16	10	OUTSPEP
20	14	ENDPUTEP
24	18	HALTEP
28	1C	DYNØUTEP
32	20	OPCM
36	24	DISPECB
40	28	DISPECB2
44	2C	DB1EP

DEZ	HEX	
48	30	DISPECBL
52	34	DBXNO
56	38	KBXDTAB
60	3C	KBXLIB
64	40	SVC255EP
68	44	SVC254EP
72	48	SVC253EP
76	4C	SVC252EP
80	50	SVC251EP
84	54	SVC250EP
88	58	A (Ausgabefeld)
92	5C	DISPL-MODUS / Länge des Ausg.Feldes
96	60	DDCB
100	64	SDCB


DEZ
HEX
104
68
108
6C
112
70
116
74
120
78
124
7C
128
80
132
84
136
88
140
8C
144
90
148
94
152
98
156
9C
160
A0
164
A4
168
A8
172
AC
176
B0
180
B4
184
B8
188
BC
STATUS
ACTEND
TENO
ADLIST
KBXTD
KBXLED
KBXTLED
KBXLE
KBXLE5
KBXLE8
KBXT
KBXTDLO
FREI
KBXASPCW
RESERVIERT
FREI
UBXD
KBXLO

DEZ	HEX	
191	BF	KBXDLO
192	C0	
196	C4	RE
200	C8	SERVIERT
204	CC	FREI / FREI
208	D0	KBXCSCB
212	D4	LISTINL / LISTOUTL
216	D8	ELISTIN
220	DC	ELISTOUT
224	E0	ELISTINA
228	E4	ELISTOTA
232	E8	TASKECB
236	ED	TTIN
240	F0	TTOUT
244	F4	TCBLOCAD

OFFSET	LÄNGE (BYTE)	NAME	BEDEUTUNG
0	1	KBXID	X'FF' :KONBOX - Markierung
1	3	LB1EP	Adresse der 1. LINEBOX
4	4	WAITEP	Adresse des Entry-Point der CSECT TCP1WAIT im Modul TCP1BASE
8	4	DISAUSFD	1.Byte : KBEOD, dient der Kennzeichnung von EOD-Situationen,die beim Lesen der Rohdaten auftreten (X'80') 2.-4.Byte: KBADD, enthält die Adresse des von DKP zur Verfügung gestellten 16K-Display-Ausgabefeldes
C	1		Dient der Kennzeichnung des 1. DKPUM-Aufrufs
D	3		Enthält die Anzahl der komprimierten Bytes zur Display-Ausgabe
10	4	OUTSPEP	Adresse der CSECT OUTSPOOL in TCP1BASE
14	4	ENDPUTEP	" " " ENDPUT " "
18	4	HALTEP	" " " TCP1HALT " "
1C	4	DYNOMTEP	" " Routine DYNOUT " "
20	4	OPCM	" " CSECT TCP1OPCP " "
24	4	DISPECB	Display-Wunsch-ECB. Von TCPOPROC gepostet, wenn Display-Ausgabe durch DKP erfolgen soll.
28	4	DISPECB2	Display-Fertig-ECB. DKP teilt durch POST auf diesen ECB dem Modul TCPOPROC das Ende der Display-Ausgabe mit.
2C	4	DB1EP	Pointer zur 1. Display-Box (DBX)
30	4	DISPECBL	Pointer zur ECB-Liste für Displays. Diese Liste enthält die Adressen von Display-I/O-ECB's und Display-Wunsch ECB.
34	2	DBXNO	Anzahl der Display-Control-Units (gesteuert durch DD-Statements)
36	2		Frei
38	4	KBXDTAB	Pointer zur Display-Terminal-Zuordnungstabelle TCP1DTAB.

OFFSET	LÄNGE (BYTE)	NAME	BEDEUTUNG
3C	4	KBXLIB	Adresse des TCP.LINKLIB-DCB (Library-DCB)
40	4	SVC255EP	Reserviert für evtl. künftigen Umbau der
44	4	SVC254EP	TCP-SVC's: Die entsprechenden
48	4	SVC253EP	SVC-Moduln sind dann in den Modul
4C	4	SVC252EP	TCP1BASE einzubauen und mittels eines
50	4	SVC251EP	SVC's der aus einem BR 15 besteht,
54	4	SVC250EP	anzuspringen.
58	4		Adresse des Ausgabefeldes, in dem die aufbereiteten Display-Daten stehen
5C	1		KBMOD: Enthält den Display-Ausgabe-Modus
5D	3		KBIOL: Länge des Ausgabefeldes, in dem die aufbereiteten Display-Daten stehen.
60	4	DDCB	Pointer zum Dynamic Buffering DCB
64	4	SDCB	" " Simple " DCB
68	1	STATUS	01 —— LOGIN —— Modus 02 —— HOLD —— Modus 04 —— SHUTDOWN
69	1	ACTENO	Anzahl der aktiven Terminals
6A	1	TENO	" aller Terminals, für die DD-Statements existieren
6B	0	ADLIST	Liste der Programmworte für die SIEMENS TRANSDATA 8525-101. Die Liste enthält mehrere Selektions-Kombinationen für die Ein-/Ausgabe-Geräte der Station. Aufbau des Programmworts: A | EOT | DC1 | PS1 | PS2 | PA | ENQ A - Anzahl der nachfolgenden Zeichen (6) EOT- CCITT-Nr.5 - Zeichen DC1- " " " PS1- Selektionszeichen PS2- " PA - Programmanweisungszeichen ENQ- CCITT-Nr.5-Zeichen

OFFSET	LÄNGE (BYTE)	NAME	BEDEUTUNG
	3		Näheres über den Aufbau des Programmwortes findet sich bei der Beschreibung der Terminal-Hardware
6B	0	Q	Quellenselektion
6B	7	KBXTD	Programmwort für die Kombination Tastatur & Drucker
72	7	KBXLED	" " " " Leser & Drucker
79	7	KBXTLED	" " " " Tastatur & Leser & Drucker
80	7	KBXLE	" " " " Leser CCITT-Nr.5, 8 Bit
87	7	KBXLE5	" " " " Leser Transparenz Mode, 5 Bit
8E	7	KBXLE8	" " " " Leser Transparenz Mode, 8 Bit
95	7	KBXT	" " " " Tastatur
9C	7	KBXTDLO	" " " " Tastatur & Drucker & Locher
A3	1		Frei
A4	8	KBXASPCW	Vom Internal Job Processor (IJP) benötigtes Control Word (nur bei der ASP-Version benötigt)
AC	4		Reserviert
B0	1		Frei
B1	0	S	Senkenselektion
B1	7	KBXD	Programmwort für die Kombination Drucker
B8	7	KBXLØ	" " " " Locher
BF	7	KBXDLO	" " " " Drucker & Locher
C6	7		Reserviert für weitere Senkenselektion
CD	3		Frei
D0	4	KBXCSCB	Adresse des CSCB-Feldes "Pointer To Exec Statement Parm Field"
D4	2	LISTINL	Länge von ECB-List-In
D6	2	LISTOUTL	" " ECB-List-OUT

OFFSET	LÄNGE (BYTE)	NAME	BEDEUTUNG
D8	4	ELISTIN	Adresse der ECB-List-In (auf die TCPOPRØC mit WAIT wartet).
DC	4	ELISTOUT	Adresse der ECB-List-Out (auf die TCP1WAIT mit WAIT wartet).
EO	4	ELISTINA	Pointer zum aktuellen ersten Eintrag in ECB-List-In
E4	4	ELISTOTA	Pointer zum aktuellen ersten Eintrag in ECB-List-Out
E8	4	TASKECB	Task-ECB für die Task TCPOPRØC
EC	4	TTIN	Pointer zur Input Translate Table
FO	4	TTOUT	" " Output " "
F4	4	TCBLOCAD	Adresse des TCB für TCPOPRØC
F8	O	KBXLENG	

Das Makro DLBX

Schreibweise:

```
name    DLBX
```

Das Makro DLBX generiert eine Linebox - DSECT.

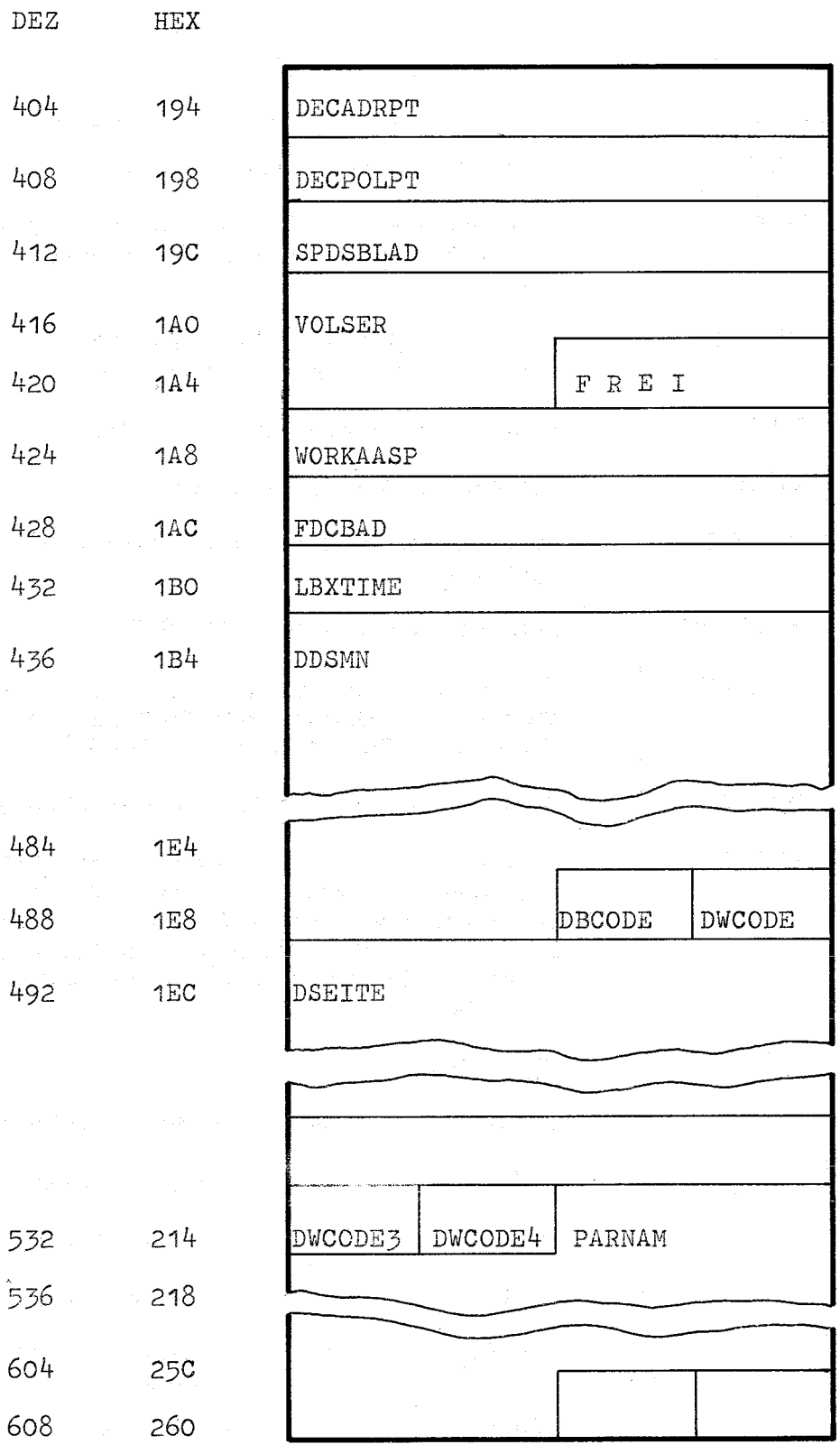
Jedem Terminal, für das in der TCP-Prozedur ein DD-Statement existiert, ist eine Linebox zugeordnet. Sie wird in der Initialisierungsphase aufgebaut und enthält DCB's sowie Informationen über den momentanen Zustand des assoziierten Terminals und über die momentane Phase des Dialogs "Benutzer-TCP".
Aufbau und Beschreibung der Linebox (LBX) siehe Bild 11.

Bild 11

LINEBOX

DEZ	HEX				
0	0	LBXCHAIN			
4	4	INECB			
8	8	OUTECB			
12	C	BUFL BUFADS			
16	10	MSBUFADS			
20	14	SPOOLTCB			
24	18	SELINDI	SELINDØ	ACTION	
28	1C	RECBUFAD			
32	20	LINEAD			
36	24	TFLG	LINENØ	PARZG	CC1
40	28	CC2	USERID		
44	2C	LAMEMBN			
48	30				
52	34	BCODE	ANZPAR	DATA	BFLG
56	38	PSODE1	WCODE1	PARF1	
60	3C	W A D R 1			
64	40	PCODE2	WCODE2	PARF2	
68	44	W A D R 2			
72	48	PCODE3	WCODE3	PARF3	
76	4C	W A D R 3			
80	50	PCODE4	WCODE4	PARF4	
84	54	W A D R 4			

DEZ	HEX				
88	58	PCODE5	WCODE5	PARF5	
92	5C	W A D R 5			
96	60	DISPLSTA	WORKTTR		
100	64	DISPNR	DISPUNIT		
104	68	DISCB			
108	6C	S E Q D C B			
112	70				
116	74				
196	C4	PARTDCB			
200	C8				
284	11C	PARTDCØ			
372	174	DECB DECSDECB			
376	178	DECTYPE	DECLNGTH		
380	17C	DECONLTT DECDCBAD			
384	180	DECAREA			
388	184	DECSENSO	DECSENS1	DECCOUNT	
392	188	DECCMCOD DECENTRY			
396	18C	DECFLAGS	DECRLN	DECRESPN	
400	190	DECTPCOD	DECERRST	DECCSWST	


DEZ
HEX
404
194
DECADRPT
408
198
DECPOLPT
412
19C
SPDSBLAD
416
1A0
VOLSER
420
1A4
F R E I
424
1A8
WORKAASP
428
1AC
FDCBAD
432
1B0
LBXTIME
436
1B4
DDSMN
484
1E4
488
1E8
DBCODE
DWCODE
492
1EC
DSEITE
532
214
DWCODE3
DWCODE4
PARNAM
536
218
604
25C
608
260

OFFSET	LÄNGE (BYTE)	NAME	BEDEUTUNG
0	4	LBXCHAIN	Pointer zur nächsten Linebox
4	4	INECB	ECB für TCPOPROC. Von TCP1BASE gepostet, falls TCPOPROC einen Auftrag ausführen soll.
8	4	OUTECB	ECB für TCPOPROC. Von TCPOPROC gepostet, wenn ein Auftrag ausgeführt ist.
C	0	BUFL	Länge des auszugebenden Puffers
C	4	BUFADS	Adresse eines Text-Puffers
10	4	MSBUFADS	Adresse eines Message-Puffers (asynchrone Messages)
14	4	SPOOLTCB	Adresse des TCB von TCP1SPOL
18	1	SELINDI	Diese Zahl addiert zu ADLIST (in Konbox) ergibt die Adresse der Terminal-Geräte-Selektion für Quellen (INPUT)
19	1	SELINDØ	Dasgleiche für OUTPUT (Senkenselektion)
1A	2	ACTION	Beide Bytes geben Informationen über den momentanen Zustand der Leitung und über den momentanen Zustand der einzelnen Tasks. Sie dienen zur Steuerung des internen Task-Ablaufs für die Bedienung der betreffenden Leitung.

1. Byte:

SCHLO	X'80'	Output für Entschlüßler
MSOUT	X'40'	Message Output
SCHLOTIN	X'20'	Input-Output für Entschl.
LOCAL	X'10'	Lokalbetrieb
SPLTASK	X'08'	Spooltask aktiv. Dynamische Eingabe
LINEDOWN	X'04'	Line Down
TEMPERR	X'02'	Behebbarer Fehler an der Leitung
ERRMSOUT	X'01'	Error Message steht zur Ausgabe an.

OFFSET	LÄNGE (BYTE)	NAME	BEDEUTUNG
			2.Byte:
			SWITCHB X'01' Outspool wird für Ausgabe angesprungen
			SYSOUT X'02' Output der Klasse X liegt vor
			EODADBIT X'04' EOD-Bedingung bei Spool-out oder Outspool
			CHAINBIT X'08' Spoolout will Fehlernachrichten am Terminal absetzen
			X'10' PL1-Start-Output
1C	4	RECBUFAD	Adresse des angeforderten Puffers für dynamisch gepuffertes Einlesen
20	4	LINEAD	Hardware-Terminal-Adresse
24	1	TFLG	TFLG enthält Informationen über den momentanen Stand des Dialogs "Benutzer-TCP".
			EIN X'80' Benutzer hat sich eingeschaltet
			SCHLPD X'40' I/O-Wunsch von Entschlüßler
			MSGPD X'20' I/O-Wunsch von Message
			INPUT X'10' Eingabe wird erwartet
			SPOOLAUT X'08' dyn.gepufferte Ausgabe erwartet
			FORMAT X'04' Formatierte Eingabe erwartet
			COMMAND X'02' Befehl wird erwartet
			SPOOLIN X'01' dyn.gep. Eingabe erwartet
25	1	LINEND	Relative Line Number
26	1	PARZG	Parameterzeiger
27	1	CC1	TCP-Fehlernachricht-Code
28	1	CC2	Interner TCP-Routinen-Completion Code
29	3	USERID	Benutzeridentifikation
2C	8	LAMEMBN	Member-Name des bearbeiteten Members

OFFSET	LÄNGE (BYTE)	NAME	BEDEUTUNG
34	1	BCODE	Befehlscode des letzten TCP-Kommandos
35	1	ANZPAR	Anzahl der zum Befehl gehörenden Parameter
36	1	DATA	X'80', wenn nach dem Befehl Daten einzulesen sind
37	1	BFLG	1.Bit: Umschalte-Befehl gegeben 2.Bit: Ein&Lies verboten
38	1	PCODE1	Nummerierung der Parameter
39	1	WCODE1	Code des zum Parameter gehörigen aktuellen Wertes
3A	1	PARF	Parameterflag; X'80', wenn Parameter korrekt eingetragen
3B	1		
3C	4	WADR1 : : :	Adresse eines Namens oder Datenfeldes, das (anstelle von Werten) die Parameterinformation enthält Sinngemäß gilt das gleiche für die Felder PCODE2 bis WADR5
60	1	DISPLSTA	Display-Status
61	3	WORKTTR	
64	1	DISPNR	Display-Nummer (Hardwareadresse)
65	3	DISPUNIT	Hardware-Adresse der Display CU
68	4	DISCB	
6C	88	SEQDCB	DCB für Spool-Data-Set. Sequentiell
C4	88	PARTDCB	DCB für User-Library Data Set. Input. Partitioned.
11C	88	PARTDCØ	DCB für User-Library Data Set. Output. Partitioned
174	0	DECB	Data Event Control Block für die Terminal Line (BTAM) (s.auch IBM/360 ØS BTAM Program Logic Manual Y30-2001)

OFFSET	LÄNGE (BYTE)	NAME	BEDEUTUNG
174	4	DECSDECB	Event Control Block für Terminal Line
178	2	DECTYPE	Operation Type. Kennzeichnet Kanalprogramm
17A	2	DECLNGTA	Buffer Length oder Area Length
17C	0	DECONLTT	Reserviert für Online Terminal Test
17C	4	DECDCBAD	Adresse des BTAM-DCB (SDCB oder DDCB)
180	4	DECAREA	Adresse des Ausgabebereiches
184	1	DECSENSO	SENSE-Information (bei I/O-Fehler)
185	1	DECSENS1	Reserviert
186	2	DECCOUNT	Residual Count
188	0	DECCMCOD	Command Code bei dem ein Error auftrat
188	4	DECENTRY	Zeigt auf das betreffende Programmwort zur Selektion der Terminal-Komponenten in der Konbox.
18C	1	DECFLAGS	Status-Flags. Nur Bit 4 ist wichtig. (Bedeutung: s. BTAM-Manual)
18D	1	DECRLN	Relative Line Number
18E	2	DECRESPN	1.Byte: Quittung auf Terminal-Selektion 2.Byte: Quittung auf Textübertragung
190	1	DECTPCOD	BTAM-Code für das aktuelle CCW
191	1	DECERRST	Beschreibung diverser I/O-Errors (s.BTAM-Manual)
192	2	DECCSWST	Status Bits vom CSW des letzten CCW
194	4	DECADRPT	
198	4	DECPOLPT	Für Retry Count verwendet
19C	4	SPDSBLAD	Adresse von Spool Data Set (Schreib)
1A0	6	VOLSER	Volume Serial Number
1A6	2		Frei

OFFSET	LÄNGE (BYTE)	NAME	BEDEUTUNG
1A8	4	WORKAASP	Adresse der Work-Area von Spoolout
1AC	4	FDCBAD	Adresse des DCB für Display-Ausgabe von fremden Members und sequ.DS.
1B0	4	LBXTIME	CPU-Zeit für eine Terminal-Sitzung wird hier gespeichert
1B4	54	DDSMN	DS-Name und Member-Name für Display
1EA	1	DBCODE	Alter Display-Befehlcode wird hier gerettet
1EB	1	DWCODE	Alter Display-Parameter-Wertcode
1EC	40	DSEITE	Seiten-Nr. für Display-Text,-Plot; Maßstab bei graph. Display; Anfangsposition bei Direkt-Text.
214	1	DWCODE3	Alter Display-Parameter-Wertcode
215	1	DWCODE4	Alter Display-Parameter-Wertcode
216	72	PARNAM	Enthält DS-Name, Zeilennummer, Formatangabe usw.
260	0	LBXENDE	

Beschreibung des Makros

HE

Mit dem Makro HE wird ein Programm beendet, das mit dem Makro ALEPH eingeleitet wurde.

Es gibt den durch das Makro ALEPH zugeordneten Speicherplatz frei und führt ein RETURN Makro aus.

Form:

```
symbol    HE    CØDE = code, SPN = subpool nr.
```

symbol:	Ein in der Assembler Language gültiger Name.
code:	Ein Returncode wie im Makro RETURN. Wird dieser Parameter nicht angegeben, so wird CØDE = 0 angenommen.
subpool nr.:	Nummer des subpools, aus welchem durch ALEPH Speicherplatz für die WØRKAREA zugeordnet wurde. Wird dieser Parameter nicht angegeben so wird SPN = 0 angenommen.

Funktion:

Im Makro HE wird die Länge der WØRKAREA berechnet, die Adresse der WØRKAREA in das Register 11 umgeladen und die Adresse der alten Savearea ins Register 13 geladen. Mit FREEMAIN wird der Speicherplatz freigegeben und mit dem Makro RETURN das Programm beendet.

Das Makro IOBD

Schreibweise

```
name    IOBD
```

Das Makro IOBD baut eine Input/output Block - DSECT für einen BTAM-IOB auf.

Die Bedeutung der einzelnen Felder dieses IOB's ist beschrieben in /7/.

Makro-Beschreibung: OPT

Das Makro OPT befindet sich auf der TCP-Maclib.

Zweck: Das Flowchart-Programm kann durch Angabe von Parametern in seinem Ablauf gesteuert werden, z.B. Art der gezeichneten Symbole. Diese Parameter werden in einer OPT-Anweisung übergeben: die OPT-Anweisung ist die erste Anweisung des Flowchart-Programms, die Parameter werden durch Lochen der Ziffern 0 bzw. 1 in bestimmten Spalten der Lochkarte angegeben.
Ein OPT-Parameter, Ziffer 1 in Spalte 24, ermöglicht, daß nur Karten vom Flowchart-Programm ausgewertet werden, die in Spalte 0 und 1 mit *$, bzw. C* beginnen. Auf diese Art ist es möglich, in Assembler-bzw. Fortranprogramme die Flowchart-Steueranweisungen an der logisch richtigen Stelle als erklärenden Kommentar einzufügen. Um bei der Assemblerprogrammierung den selben Kartenstapel sowohl durch Assembler als auch durch das Flowchart-Programm auswerten zu können, muß die als erste Anweisung erscheinende OPT-Karte dem Assembler bekannt sein. Dies wird durch die Definition eines Pseudo-Makros OPT erreicht.

Wirkung: OPT wird durch den Makro-Assembler zum erklärenden Assembler-Kommentar "*Dient zur Steuerung des Flowchart-Programms" expandiert.

Das Makro SUCB

Schreibweise:

```
name  SUCB
```

Das Makro SUCB (Start Up Control Block) wird während der TCP-Initialisierung benötigt. Es enthält während dieser Phase Informationen, die nach erfolgter Initialisierung in der Konbox bzw. in den Lineboxes gehalten werden.
(Beschreibung siehe Bild 12)

Bild 12

S U C B

Dez	HEX		
0	0	SUSIZE	
4	4	SULBXST	
8	8	SULBXEND	
12	C	SUCSCB	
16	10	SUDEBL	
20	14	SUTIOT	
24	18	SULINO	
28	1C	SUTDCB1	
32	20	SUTDCB2	
36	24	SUTTOUT	
40	28	SUTTIN	
44	2C	SUPARM SUBUFNO	
48	30	SUBUFFL	
52	34		
56	38		
60	3C		
64	40	SUBNO	SUBLENG

SUSIZE	Länge des SUCB in Bytes
SULBXST	Adresse der 1. LBX
SULBXEND	Adresse der letzten LBX
SUCSCB	Adresse des Feldes "Pointer To Exec-Statement Parm Field" im CSCB
SUDEBL	DEB - Länge
SUTIOT	Adresse des TIOT
SULINO	Anzahl der Lines, für die DD-Statements existieren
SUDCB1	Adresse des SDCB
SUDCB2	Adresse des DDCB
SUTTOUT	Pointer auf die Translate Table "EBCDIC - ISO7"
SUTTIN	Pointer auf die Translate Table "ISO7 - EBCDIC"
SUBUFNO	Anzahl der Puffer (gezonte Darst.)
SUBUFFL	Länge der Puffer (gezonte Darst.)
SUBNO	Anzahl der Puffer (gepackte Darst.)
SUBLENG	Länge der Puffer (gepackte Darst.)

Das Makro WTP

Das Makro **WTP** befindet sich auf der Platte TCPLIB in der Library TCP.MACLIB. Es kann wie ein Systemmakro verwendet werden, in dem man die TCP.MACLIB durch eine DD-Karte an die SYS1.MACLIB kettet.

Anwendungszweck:

Das Makro **WTP** ist geeignet, während der Ausführung eines Assembler-Programms zu Testzwecken und zur Information über den Programmablauf eine Nachricht auszugeben. Die Nachricht erscheint auf dem Listing des Benutzers.

Schreibweise: Es gelten die Regeln der Assembler-Programmierung

Name WTP 'Text' [,N = ein Zeichen]

Name: wird ein Name angegeben, so kann das Makro als Sprungziel oder als Referenz verwendet werden.

'Text': die Zeichenkette innerhalb der Apostrophe wird später als Nachrichtentext ausgegeben. (Ein Apostroph innerhalb des Textes muß durch zwei Apostrophe dargestellt werden.); maximale Anzahl der Zeichen = 125; es gelten die Regeln über Fortsetzungskarten.

N = ein Zeichen: kann weggelassen werden bei Programmen, die aus nur einer CSECT bestehen; besteht ein Programm aus mehreren CSECT, bzw. DSECT, so muß in jeder SECT ein anderes Zeichen verwendet werden, dieses Zeichen muß bei jedem WTP-Makro der SECT geschrieben werden.

Wirkungsweise:

Der erste Makroaufruf WTP pro Assemblerablauf erzeugt neben den eigentlichen Vorbereitungs- und Ausgabestatements außerdem die Versorgungsblöcke DCB und WORKAREA. Er belegt einen Platz (in Bytes) von 393 + Anzahl der Textzeichen. Jeder weitere Aufruf erzeugt nur Vorbereitungs- und Ausgabestatements. Die Expansion belegt einen Platz (in Bytes) von 102 Anzahl der Textzeichen.

Während der Programmausführung können die WTP's in beliebiger Reihenfolge passiert werden.

Das Makro WTP schreibt den Text in eine Liste, die zusammen mit anderen Ergebnissen und Nachrichten über die Output-Queue ausgegeben wird.
Für den Go-Step wird dazu folgendes DD-Statement benötigt:
//WTPMESS* DD SYSOUT = A;
dabei muß * dem Zeichen in N = * entsprechen;
falls N = ... fehlt, wird * = G, d.h.
//WTPMESSG ... verlangt.

Achtung:

1) Es darf nur eine CSECT, bzw. DSECT mit WTP-Makros in einem Lauf assembliert werden.
2) Besteht das Programm aus mehreren, in getrennten Läufen assemblierten Teilen, so müssen die WTP's der gleichen CSECT, bzw. DSECT mit dem gleichen N = * Zeichen spezifiziert werden.
3) Für jedes N = * Zeichen muß ein //WTPMESS* DD-Statement eingefügt werden.
4) Soll mit einem WTP-Makro verschiedener Text ausgegeben werden, so muß im WTP-Makro der Platz durch Angaben eines DUMMY-Textes genügender Länge freigehalten und der aktuelle Text während der Programmausführung auf diesen Platz gebracht werden.
   der freigehaltene Platz beginnt bei Name + 9 Bytes.
5) Das Makro WTP ist nicht reenterable.

Das Makro WTT

MACRO WTT
(Write - to - terminal)

FUNKTION
Eine in WTT angegebene Nachricht wird dem Benutzer am Terminal ausgegeben oder in das Member M seiner Bibliothek geschrieben, wenn der Befehl während des Programmablaufs ausgeführt wird. (vergl. WTO, WTP)

Voraussetzung
TCP ist aktiv; sonst keine Wirkung

Schreibweise
MARKE WTT 'NACHRICHT'

SPEICHERBEDARF pro Aufruf
82_{10} Bytes + Anzahl der Nachrichtenzeichen

Achtung
WTT ist für Testzwecke geeignet, doch sollte es nicht zu häufig und keinesfalls in unkontrollierten Schleifen verwendet werden. Stehen keine Buffer mehr zur Verfügung, werden die Nachrichten vernichtet.

JCL WTT befindet sich auf der TCP.MACLIB, es muß also die TCP.MACLIB an die SYSLIB DD SYS1. MACLIB für den Assemblierungslauf gekettet werden.

Dynamische Parameterversorgung
Sollen erst während des GO-STEPS die NACHRICHTEN-Zeichen in das Macro gebracht werden, so muß
1.) im Macroaufruf genügend Platz reserviert werden
2.) die NACHRICHT - Zeichen ab MARKE + 21_{10} abgespeichert werden.

ATTRIBUTE Serially reusable, not reenterable

FORTRAN Subroutine WTT ('TEXT# ')

Für FORTRAN steht eine Subroutine WTT ('TEXT# ') zur Verfügung, die mittels CALL aufgerufen werden kann.

FUNKTION

Der in Apostrophen angegebene TEXT wird entweder an dem Terminal ausgegeben, an dem der Benutzer aktiv ist, oder er wird auf das Member M geschrieben, wenn der Benutzer nicht aktiv ist, sobald das Unterprogramm aufgerufen wird.

Schreibweise: CALL WTT ('TEXT#')

Die Textzeichen müssen in Apostrophen stehen und mit einem# abgeschlossen werden. Wird Text nicht mit einem #abgeschlossen, so werden nur die ersten drei Zeichen der Nachricht zusammen mit einem Hinweis auf das fehlende #ausgegeben.

Achtung

Die Anzahl der TEXT - Zeichen ist auf 62 beschränkt. Werden mehr Zeichen angegeben, hat das Unterprogramm überhaupt keine Wirkung.

TCP muß aktiv sein, sonst keine Wirkung!

Der Benutzer muß eine gültige TCP-Identifikation besitzen, sonst keine Wirkung!

WTT ist als Testhilfe gedacht, doch sollte es nicht unnötig oft verwendet werden (starke Systembelastung). Um die Verwendung des WTT-Aufrufes in unkontrollierten Schleifen zu beschränken, kann jeder WTT-Aufruf höchstens 20 mal mit einer Wirkung durchlaufen werden. Mehr Durchläufe bleiben wirkungslos.

ATTRIBUTE Nicht reenterable, bedingte Reusable.

Anhang D

Spezielle Blitzstartprogramme (H. Santo)

Folgende Blitzstartprogramme werden in diesem Anhang beschrieben:

XIAK 12
XIAK 24
XCAE

D.1 Blitzstartmodul XIAK12

Der Loadmodul XIAK12 wird von der CSECT PRØC (Modul TCPOPRØC) über den Modul TCP1SPRØ dynamisch initialisiert (MACRØ LINK), wenn der Terminalbenutzer die Ausführung des Befehls

BLI, XIAK12, MEMBERN

verlangt.

Hierbei kennzeichnet MEMBERN ein Member in der "partitioned" organisierten Bibliothek (TCP.XXX.USER) des Benutzers.

Das Member besitzt die im TCP übliche Blocklänge von 1600 Byte mit einer logischen Satzlänge von 80 Byte, wobei der letzte Block kleiner als 1600 Byte sein kann.

Das Programm XIAK12 wandelt die im quasitransparenten Modus mit dem Terminalbefehl

LIES, LOCH ... mit Parameterangabe CODE = BIN

über den Lochstreifenleser in die Benutzerbibliothek eingelesenen Daten um in vorzeichenlose Dezimalzahlen. Das neue Member, das die aufbereiteten Daten enthält, bekommt denselben Namen wie das Member, mit den Rohdaten.

Es handelt sich bei den eingelesenen Rohdaten um Daten eines 8-Kanal-Lochstreifens (Ausgabe eines Vielkanalanalysators, bei welchem nur 6 Kanäle relevante Information liefern.) Die 6 relevanten Kanalinhalte der beiden aufeinanderfolgenden Sprossen ergeben entsprechend aneinandergefügt die Binärdarstellung eines Meßwertes.

Um das Einlesen an der Siemensdatenstation zu ermöglichen, ist der 8. Kanalinhalt immer gesetzt.

Das Zeichen der 2. Sprosse ohne Lochung (nur Transportloch) würde sonst als Endezeichen (NUL) interpretiert werden und den Einlesevorgang beenden.

Die 2. Sprosse ist durch Lochung des 7. Kanals gekennzeichnet.

Beispiel der Darstellung auf

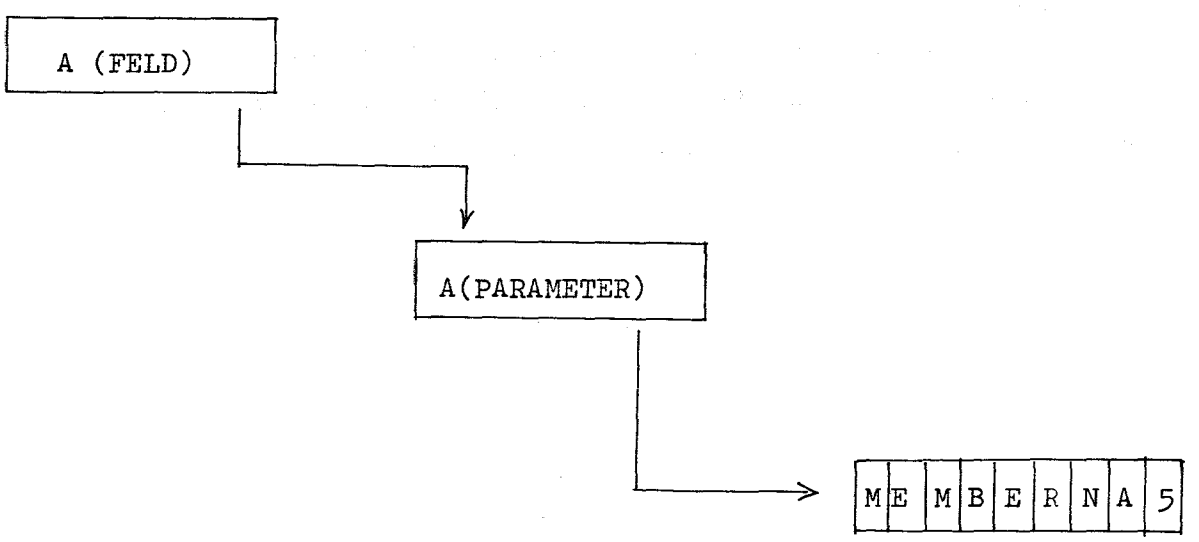
Lochstreifen	als Rohdaten	Hexadezimal-wert nach Um-wandlung	Dezimalwert
● ● ● , O O O ● ● ● O O , O O O O ● O O O , O O O ● ● ● ● ● , O O O O ●	X'C781'	X'0101'	449
	X'C017'	X'0007'	7

Die größte vorkommende Zahl ist 2^{12}, da in jeder der beiden Sprosse je 6 gültige Kanalinhalte vorhanden sind. Das entspricht einer 4-stelligen Dezimalzahl, die in gezonter Darstellung 4-Byte einnimmt. Das nachfolgende 5.Byte wird zur Trennung der Ziffern mit einem Leerzeichen aufgefüllt. Dadurch, daß einem Hexadezimalwert von 2 Byte ein dezimaler Wert mit 4 Byte + Zwischenraum entspricht, muß Lesen und Schreiben auf den Datensatzsynchron vor sich gehen.
Das Programm trägt das Attribut RENT.

EINGABEPARAMETER

Register 3: enthält die Adresse der Linebox
Register 1: zeigt auf ein Feld in welchem die Adresse eines Parameterfeldes steht, das den Membernamen enthält.

REG1



Parameter aus der Linebox:

PARTDCO Adresse des Datenkontrollblockes, der den Benutzerdatensatzes beschreibt (Ausgabe)

PARTDCB wie oben nur für (Eingabe)

USERID Benutzeridentifikation

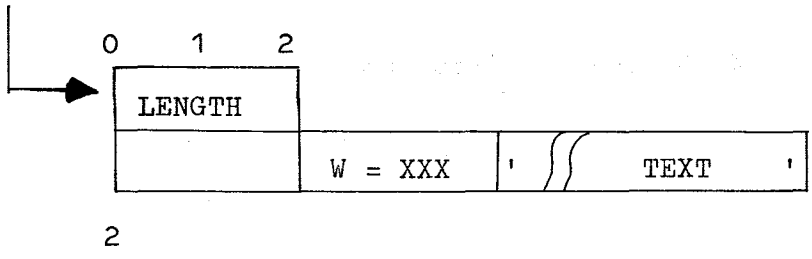
Ausgabeparameter

LINEBOX

LCC12 X'00' oder Fehlercodes

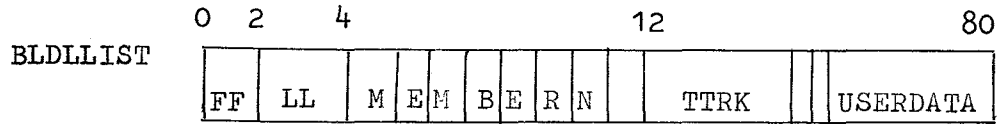
XIAK12 Bei der Initialisierung werden die Adressen der Fehlerroutinen in den PARTDCB gespeichert. Das Programm ist reentrant, deshalb müssen die MACRØS in ihren Listen- und Ausführungsformen verwendet werden. Die Listenformen werden im Arbeitsbereich des Programmes aufgebaut.
Für die Benachrichtigung des Terminaloperateurs mit Hilfe eines SVC's wird eine Liste folgender Struktur aufgebaut

Reg 1



XXX = Benutzerindenfikation = USERID
TEXT = ZZZ BEARBEITETE WØRTE
LENGTH = Länge der Nachricht in Byte

und deren Adresse in Register 1 übergeben.
Die Eingabeliste für das Macro BLDL wird im Arbeitsbereich angelegt.



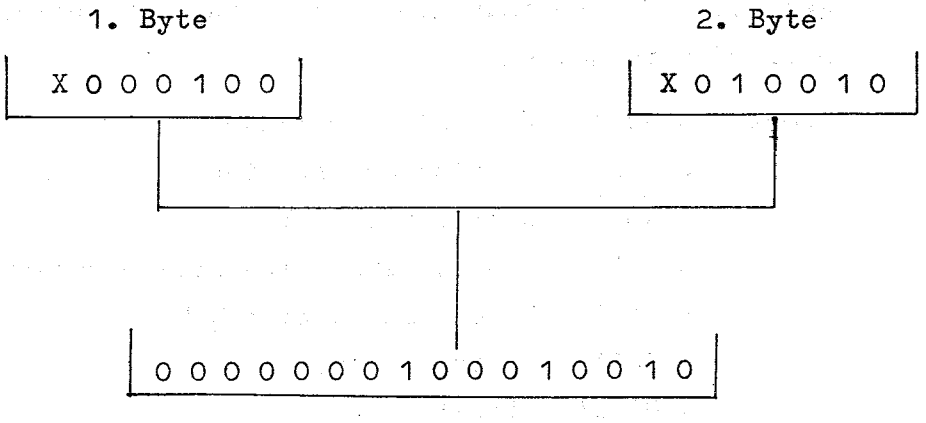
FF = Anzahl der Einträge hier 1
LL = Länge eines jeden Eintrages hier 76 =
Gesamtlänge - 4
MEMBERN wird dem Eingabeparameterfeld entnommen.

Es wird geprüft ob der Membername in der Bibliothek des Benutzers vorhanden ist. Wird der zu ändernde Member nicht gefunden, übergibt man der Linebox in LCC12 den Fehlercode H'150' um das Ausschreiben der Nachricht "Name nicht gefunden" am Terminal zu veranlassen. Bei I/O-Fehler wird LCC12 = H'110' gesetzt und die Nachricht Lesefehler Inhaltsverzeichnis am Terminal abgesetzt.

NAMEFOUN Der Membername wurde im Inhaltsverzeichnis des benutzerspezifischen Datensatzes gefunden. Für die eingelesenen Daten wird ein 1600-Byte Feld AREA im Arbeitsbereich (AREA) von 1600-Byte für die bearbeiteten, umgewandelten Daten bereitgestellt. Die Anzahl bearbeiteter Blöcke (BN), die Fehlerkennzeichnung (FEHLERK) zur Markierung aufgetretener Stanzfehler und den Switch READM, der gesetzt wird, wenn der zuletzt geschriebene Block nicht voll ist, werden gelöscht (X'00')
Der 1. Block wird eingelesen (READ in BPAM EXECUTE FORM) und auf die vollständige Ausführung der I/O Operation gewartet und geprüft. Die Anzahl gelesener Byte wird bestimmt und damit die Adresse des zuletzt eingelesenen Byte im Feld AREA (AREAEND) festgelegt.

AREAEND Es schließt sich das Bearbeiten und Prüfen der eingelesenen Daten an. Die Prüfung erstreckt sich auf Bit 7 und 8 aller Byte mit ungeraden Adressen und Bit 8 aller geraden Adressen. Das entspricht am Lochstreifen den Inhalt der Kanäle 7 und 8.

MØVEANF Ist das 7. und 8. Bit des 1. Byte nicht gesetzt, so wird dem Benutzer die Anzahl der bearbeiteten Byte ausgegeben (SVC32), so daß er auf den als fehlerhaft erkannten Meßwert schließen kann. Es kennzeichnet LCC12 mit X'0AA1' eine falsche Wortbeginnmarkierung und LCC12 = X'0AA12 = X'0AA2' zwei aufeinander folgende Wortbeginnmarkierungen und LCC12 mit X'0AA0'. 8.Bit nicht gesetzt. Die Rohdaten innerhalb des Members werden wie folgt umgewandelt.



und als Dezimalzahl (4 Byte) in das Ausgabefeld AREAA übertragen. Ein Blank zur Trennung eingefügt. Die Adresse im Ausgabefeld um ein Displacement von 5 erhöht. Die Adresse im Eingabefeld um 2

WRITEN erhöht. Ist das Ausgabefeld voll, so wird der Block anschließend an den letzten Member ausgeschrieben. Beim 1. Block die relative Spuradresse festgehalten und ein Merker gesetzt, daß der 1. Block bearbeitet wurde. (BZ = X'01'). Die Anzahl der übertragenen Blöcke (BN) um Eins erhöht. Ist FEHLERK gesetzt, so wird dem Benutzer die Anzahl der bearbeiteten Worte an seinem Terminal ausgeschrieben. Wurde ein Fehler beim 1. Wort festgestellt, so wird der Eintrag in das Inhaltsverzeichnis seiner Bibliothek übergangen und das Programm beendet.

XA Sind jedoch nicht bearbeitete Daten im Eingangspuffer vorhanden, so wird ihre Bearbeitung fortgesetzt und bei Ende des Eingabepuffers ein neuer Block eingelesen.

EODADADR Trifft man auf Ende des Ausgabedatensatzes d.h. EODAD, so wird die Anzahl der ausgegebenen Meßdaten (= 1 Kanalinhalt eines Vielkanalanalysators) über die geschriebene Anzahl voller Blöcke die Anzahl der Byte im letzten Block bestimmt, und als gezonte Dezimalzahl innerhalb des Parameterfeldes für den SVC 253 notiert.

ENDDIR In der als Output des Macros BLDL erzeugten Liste wird die neue relative Spuradresse eingetragen (der Membernamen bleibt derselbe) und im Directory, der alte Membereintrag ersetzt.

Fehlerfälle

Bei den verwendeten BPAM Macros werden die Returncodes geprüft und sinngemäß die definierten Fehlereinträge in LCC12 der LINEBOX vorgenommen.

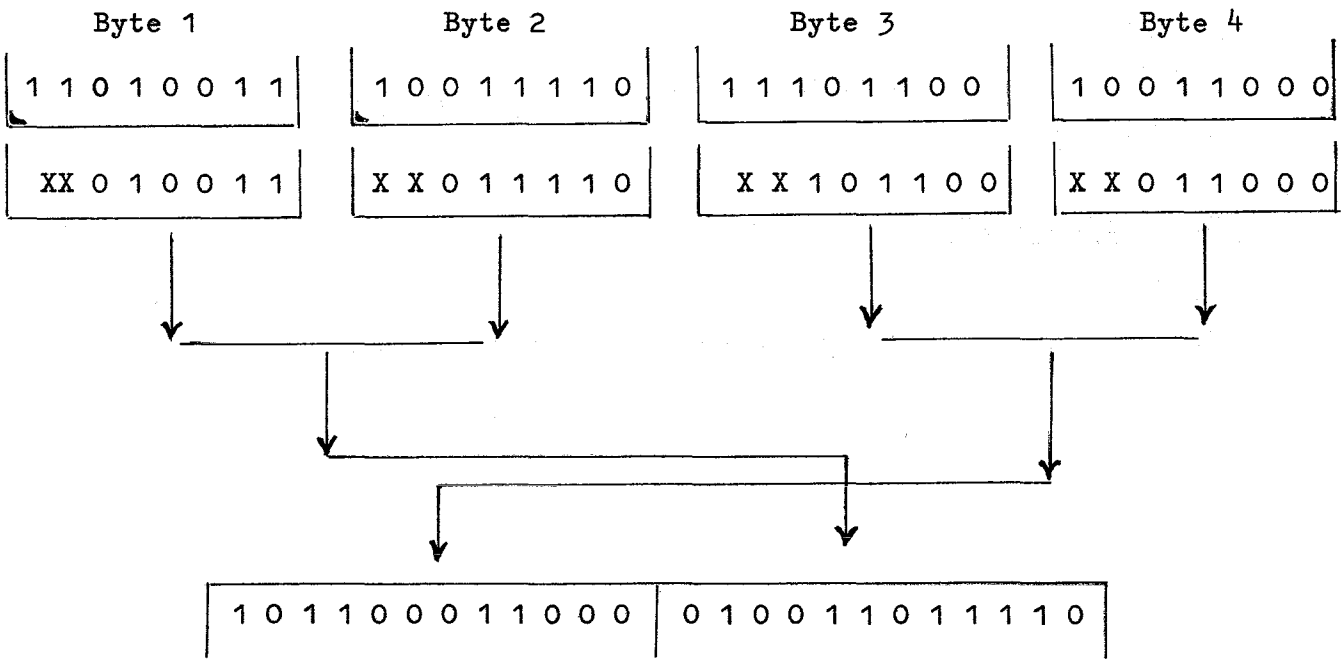
LCC12	H'150'	Name nicht gefunden
	H'110'	Lesefehler Inhaltsverzeichnis
	H'107'	Bibliothek voll
	H'106'	Schreibfehler Inhaltsverzeichnis
	H'119'	Lesefehler Bibliothek
	H'105'	Directory voll
	X'0AA0'	Stanzfehler
	X'0AA1'	Stanzfehler Wortbeginn
	X'0AA2'	Falsche Wortbeginnmarkierung

D.2 XIAK24

Der Modul XIAK24 wandelt Daten eines Loadformatlochstreifens der Midasanlage, der über den Lochstreifenleser der Siemens Datenstation im Transparenzmodus eingelesen wurde, in 24-stellige Binärzahlen um. Der Modul XIAK24 ähnelt im Aufbau dem Blitzstartmodul XIAK12. Die Daten eines vom Terminaloperateurs spezifizierten Members seiner Bibliothek werden bei Eingabe des Kommandos

BLI, XIAK24, MEMBERN

wie folgt bearbeitet:



Bit 7 und 8 eines jeden Byte enthalten keine Information. Sie dienen hier nur zur Markierung des Wortbeginns (Bit 7) bzw. sind notwendig (Bit8) um Einlesen des Lochstreifensan der Datenstation zu ermöglichen. Die umgewandelten Daten werden unter dem alten Membernamen abgelegt und der Terminaloperateur über die Anzahl bearbeiteter Worte benachrichtigt. Einzelheiten siehe Strukturdiagramm.

D.3 Blitzstartmodul XCAE

DerAufruf des Programmes XCAE erfolgt bei Eingabe des Befehles

BLI, XCAE, MEMBERN

am Terminal durch die CSECT PROC (Modul: TCPOPROC) über TCP1SPRØ

EINGABEPARAMETER:

Register	1	enthält die Adresse des Parameterfeldes	
Register	3	Adresse der LINEBØX	
LINEBØX			INPUT
PARTDCB		Datenkontrollblöcke	OUTPUT
PARTDCØ			

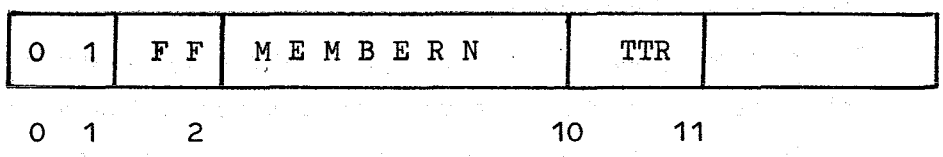
AUSGANGSPARAMETER:

LINEBØX	
LCC12	0 oder entsprechende Fehlercodes

Programmablauf:

XCAE

Der Inhalt von Register 1 adressiert ein Parameterfeld, welches den Namen eines Members enthält, auf welchen die spezielle Umwandlungsfunktion angewendet werden soll. Der Membername wird zum Aufbau einer Liste für das Macro BLDL verwendet.



Das 1. Byte gibt die Anzahl der Einträge pro Liste an hier : X'01'.
Das 2. Byte definiert die Länge eines jeden Eintrages hier: X'FF'.
Die Returncodes des Macros BLDL geben an, ob der gesuchte Member in der Bibliothek vorhanden ist (→NAMEGEF), nicht existiert (→NAMENØTF) oder I-Ø-Fehler aufgetreten sind (→IØERRF)

NAMEGEF

Über das Macro FIND (Eingabeparameter: Relative Adressliste) wird dem Kontrollprogramm mitgeteilt, daß die nachfolgende Leseoperation beim 1.Block des in der Liste spezifizierten Members beginnen soll.

READNEU

Blockweise (1600 Byte max.) wird der Member in den Arbeitsbereich des Moduls XCAE übertragen und die eingelesenen CAE-Zeichen in entsprechende EBCDIC-Zeichen übersetzt.
Die verwendete Codetabelle und die aufgestellten Übersetzungstabellen sind weiter unten aufgeführt.

MAIABF
MINABF

Über die Codezeichen für Buchstabenumschaltung bzw. Ziffernumschaltung wird die zugehörige Übersetzungstabelle ausgewählt und mit ihrer Hilfe Zeichenweise die CAE-Codezeichen in EBCDIC-Codezeichen übersetzt.

TRANSL Ist das zu übersetzende Zeichen in seiner Bedeutung 'neue Zeile', so wird der Satz (80 Byte) im Ausgabefeld beendet und wenn notwendig mit Leerzeichen aufgefüllt. Alle Zeichen, denen in der Übersetzungstabelle die Wertigkeit X'FF' zugeordnet wurde, werden ignoriert. Die übersetzten Zeichen werden in den Ausgabepuffer (1600-Byte) übertragen.

WRITE Ist der Ausgabepuffer voll oder sind alle Daten des Members bearbeitet, so werden die übersetzten und aufbereiteten Daten blockweise auf den "partitioned" organisierten Datensatz des Benutzers geschrieben.

ZEIVØRH Der letzte Block kann weniger als 1600 Zeichen enthalten, muß jedoch, wenn notwendig, vor dem Ausschreiben unter Einfügen von Leerzeichen auf ein Vielfaches der logischen Recordlänge (hier LRECL=80 Byte wegen Kartenformat) ergänzt werden.

ENDST Das neu generierte Member wird unter dem alten Membernamen in das Inhaltsverzeichnis des Datensatzes des Benutzers aufgenommen.
An möglichen Fehlercodes können der Linebox im Parameterfeld LCC12 folgende Werte übergeben werden:

107 Bibliothek voll
108 Schreibfehler Bibliothek
110 I-O Fehler (Makro: BLDL)
119 Lesefehler Bibliothek
150 Membername nicht gefunden.

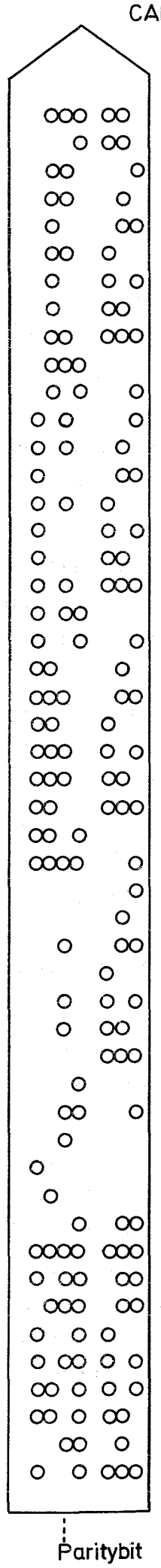
Übersetzungstabelle CAE - EBCDIC

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	7F F1	50 F2		6E F4			F7	6D F8			7C		16 16	MA7 MA7		0
7A F0			4C F3		6C F5	5A F6			F9	40 40					7F 7F	1
61 4E		C3 C3			C5 C5	C6 C6			C9 C9	7F 7F						2
	C1 C1	C2 C2		C4 C4			C7 C7	C8 C8			4B			MIN MIN		3
60			D3		D5	D6			D9			6F 5C			OD OD	4
	D1	D2		D4			D7	D8					7D 5D	7F 7F		5
		E2 E2		E4 E4			E7 E7	E8 E8					LB 4D			6
			E3 E3		E5 E5	E6 E6			E9 E9						7F 7F	7
																8
																9
																A
																B
																C
																D
																E
																F

Allen nicht zugelassenen Codezeichen wird der Hexadezimalwert 7C zugeordnet. Das entspricht dem Zeichen ". Bei der Ausgabe des Members sind somit leicht die Codeverstümmelungen zu erkennen, welche nicht zugelassene Zeichen erzeugt hatten.

CAE-LOCHSTREIFEN

FLEXOWRITER-LOCHSTREIFEN



Großschreibung	Kleinschreibung		
A	a		
B	b		
C	c		
D	d		
E	e		
F	f		
G	g		
H	h		
I	i		
J	j		
K	k		
L	l		
M	m		
N	n		
O	o		
P	p		
Q	q		
R	r		
S	s		
T	t		
U	u		
V	v		
W	w		
X	x		
Y	y		
Z	z		
1	"		
2	&		
3	<		
4	>		
5	%		
6	!		
7	‡		
8	—		
9	¢		
0	:		
-	=	01000000	X'40'
+	/		
@	•		
Feedcode			
$	#	01011011	X'5B'
.	;	00111011	X'3B'
*	?		
)	'	01011101	5D
(	,		
Tabulator			
Space			
Carriage Retum		Nicht in EBCDIC	

Anhang E

TCP unter ASP (W. Strolz)

Im 2. Halbjahr 1970 wurde im Rechenzentrum der Ges.f.Kernforschung eine IBM 360/85 installiert und über eine Channel-To-Channel-Verbindung mit der bereits vorhandenen IBM 360/65 gekoppelt. Als Kopplungssoftware wurde ASP (Associated Support Processor System) eingeführt. Es bestand daher die Notwendigkeit an das geänderte Betriebssystem anzupassen.
Soll TCP unter ASP (Associated Support Processor System) laufen, dann sind folgende Einzelheiten zu beachten:

1. TCP sollte möglichst als Job und nicht als System Task laufen, da sonst das ASP-Konzept der System-Resource-Verwaltung von TCP durchbrochen wird. Aus Effektivitätsgründen sollte TCP auf dem Local Main Processor (Support Processor) laufen.
   Läuft TCP als Job, dann funktionieren zunächst keine Operator-SEND-Commands mehr, da diese Commands über einen Command Scheduling Control Block laufen (CSCB), der einen Entry TCP... enthält. Um einen CSCB zu generieren muß man eine System Task (TCP03) aufbauen, die ihren CSCB für SEND-Commands zur Verfügung stellt, ansonsten aber keine Funktion hat, Die Verbindung mit TCP wird während der Initialisierungsphase von TCP hergestellt.
   Näheres darüber findet man bei der Beschreibung des Moduls TCP2BEGN.

2. Die Console Intercept Logic (SEND-Commands) ist bei der ASP/OS-Version anders als bei einem reinen OS-Betriebssystem. Dieser Tatsache hat man beim Einbau der für SEND-Commands benötigten Modulen CRB10503D und IGC0503D in die SYS1.SVCLIB Rechnung zu tragen.
   Der ASP-Modul IGC0503D enthält einen ASP-Command-Interpreter und einen Error Exit. Soll TCP unter ASP laufen, muß dieser Modul substituiert werden durch den SEND-Interpreter-Modul IGC0503D; der alte ASP-Command-Interpreter-Modul muß den neuen Namen IGC9503D bekommen.
   Er arbeitet dann als Error Exit für den SEND-Interpreter-Modul. Außerdem muß wie unter OS der Modul CRB1503D neu eingefügt werden. Diese Zusammenhänge werden in Bild 13a,b,c, ersichtlich gemacht.

3. Eine wesentliche Funktion von ASP besteht im Verteilen der Jobs auf die Processoren und in der Verteilung der Resources auf die in der ASP-Queue wartenden Jobs. Ein effektives Arbeiten dieses Features ist nur gewährleistet, wenn dabei alle ankommenden Jobs erfaßt werden können. Daher kann die im reinen OS-Betrieb verwendete Technik um Jobs von TCP aus in die OS-Input Queue zu bringen (Starten eines speziellen OS-READERS) nicht übernommen werden. Bei ASP ist statt dessen ein Internal Job Processor (IJP) vorgesehen, der als selbständiges ASP-Dynamic Support Program (DSP) vom Operator gerufen werden kann und dann von TCP aus Jobs an ASP weitergibt.

Dazu sind folgende Änderungen notwendig:

Der Modul TCP1STA entfällt.

Statt dessen wird die CSECT STARTPRO in den Modul TCPOPRØC eingefügt. Soll von TCP aus ein Job gestartet werden, ruft PROC (Modul TCPOPROC) via LINK die CSECT STARTPRØ auf. Dort wird die vom Internal Job Processor Writer (IJPWTR) benötigte Parameterliste mit aktuellen Daten über den zu startenden Job gefüllt und der IJPWTR mit ATTACH gerufen.
Der IJPWTR - Modul muß als TCP-Modul in der TCP.LINKLIB existieren.
Der IJPWTR wurde an einigen Stellen geändert (insbesondere um aus den TCP-Benutzerbibliotheken lesen zu können).
Das IJP - DSP muß vom Operator beim ASP-Kaltstart gerufen werden.
Strukturbeschreibung und interne Logik des IJP findet man unter /14/ und /15/.

TCP2BEGN

CSECT TCP2BEGN

ENTRY via ATTACH von System Task Control Routine
(IEEPWILI)

EINGANGSPARAMETER
R1 zeigt auf das Feld im CSCB
"Pointer to EXEC Statement PARM-Field"

EXIT RETURN zum OS Task Supervisor

FUNKTION
Stellt einen CSCB für SEND-Commands zur Verfügung, wenn TCP als Job läuft (siehe Kurzbeschreibung).

PROGRAMMABLAUF

TCP2BEGN Der Command Input Buffer (CIB) und der CIB-Pointer werden freigegeben bzw. gelöscht.

REPEAT Die Operator-Message
'DER REPLY "LCSTCP" BEENDET DIE TCP DUMMY ROUTINE'
wird durch ein WTOR abgesetzt.
TCP2BEGN wartet dann auf das Eintreffen des Replys LCSTCP. Kommt ein Reply, dann wird er auf Gültigkeit untersucht; ist er ungültig wird das WTOR bei REPEAT wiederholt, andernfalls erfolgt RETURN zum OS-Tasksupervisor.

CSECT STARTPRO von TCPOPROC bei ASP

ENTRY Von CSECT PROC via LINK

EINGANGSPARAMETER

LBXREG, KONBOXREG

EXIT RETURN nach PROC

RETURN Code 0 - alles ok

RETURN Code 111 - BLDL-Liste konnte nicht ganz gefüllt werden

RETURN Code 112 - Parameter I/O-Error auf User-Library

RETURN Code 113 - Art ist keine JCL

PROGRAMMABLAUF

STARTPRO Der Fehler-Code wird auf FFFE gesetzt und Accounting angeworfen. Nach Initialisierung der BLDL-Liste erfolgt BLDL um restliche Informationen über das Member, das als Job zu starten ist, zu

WEITER holen. Ist BLDL ordentlich abgelaufen, wird falls das Member aus Job Control Language (JCL) besteht, also einen Job darstellt,

INITPLST der Fehlercode gelöscht und das vom IJPWTR benötigte Parameterfeld mit aktuellen Informationen versehen. (Dieses Feld wurde dynamisch mit GETMAIN geholt). Dann wird der IJPWTR mit ATTACH aufgerufen. Abschließend wird die Accouting-Information in das

ENDE dafür vorgesehene Feld in der Linebox eingetragen, und nach PROC mit RETURN zurückgesprungen. Stellt das betreffende Member keinen Job dar, wird der Code 113 gesetzt und nach ENDE gesprungen.

NOFILL Falls sich die BLDL-Liste nicht vollständig füllen ließ, erfolgt nach Setzen von Code 111 ebenfalls Sprung nach ENDE.

PIOE Trat bei BLDL ein Parameter I/O-Error auf werden Code 112 und bestimmte DCB-Flags gesetzt; anschließend ebenfalls Sprung nach ENDE.

MODUL IJPWTR Internal Job Processor Writer

CSECT IJPWTR

ENTRY Von CSECT STARTPRO (TCPOPROC) via ATTACH

EINGANGSPARAMETER

R1 zwigt auf folgende Parameter-Liste:
A (VOLSER)
A (Data Set Name)
A (Member Name)
A (ASP -DSP - Name)
A (Options)
A (Communications-ECB)
A (Workarea)
Options: bei TCP wird SD spezifiziert, d.h.
serielle Abarbeitung

Communications-ECB:
Dummy. Wird bei TCP nicht verwendet.
Workarea: 2 Fullwords in der Konbox

EXIT RETURN nach TCPOPROC, CSECT ATER

FUNKTION

Der IJPWTR stellt das TCP-ASP Interface dar.
Er liest das als Job zu startende Member der TCP-Benutzerbibliothek und schreibt es über den Channel To Channel (CTC) um es schließlich vom IJP-DSP weiter bearbeiten zu lassen. Danach geht die Kontrolle zurück an TCPOPROC.

LITERATUR

Über IJP und /IJPWTR:
IBM-Application Programm GH20-0323
IBM/360
ASP
SYSTEM PROGRAMMER'S MANUAL

und:

IBM - Application Program Y20-0305
IBM/360
ASP
SYSTEM MANUAL

HINWEIS

Wird mit OPTIONS=CD (Checkpointing) gearbeitet, dann muß ein sequentieller CHECKPOINT-Data Set existieren (s.TCP-Internals)

PROGRAMMABLAUF

IJPWTR
CTCIORT2
INITCTC
CKDISK
RINJFCB
BGNIO
GETARCD
PUTARCD
CKEND1
INPEODRT

Die Option-Parameter werden untersucht, ob mit CHECKPOINTING gearbeitet wird. Ist das nicht der Fall wird aus SP100 Arbeitsspeicher geholt und die Parameter dorthin gerettet. Dann wird Direct Access Input, bzw. Tape Input, angezeigt, DCB's und JFCB initialisiert, sowie die Vollständigkeit des DD-Statements überprüft, das die Benutzerbibliothek beschreibt. Enthält DD-Statement fehlerhafte Informationen, erfolgt WTO IJPW015 und schließlich RETURN nach TCPOPROC via ARETURN mit Completion Code in Reg. 9 . War das DD-Statement in Ordnung, wird bei DA-Input der Job File Control Block gelesen. Achtung: TCP-Spezifische Codierung: Die Felder LRECL, BLKSIZE und RECFM werden nicht auf Null gesetzt, sondern direkt vom DD-Statement übernommen; dadurch erreicht man, daß der IJPWTR aus den speziell formatierten TCP-Benutzerbibliotheken lesen kann. Das IJP-DSP wird nun mittels WTOR auf Empfangsbereitschaft untersucht. Ist der IJP bereit, werden INPUT-DCB und CTC-DCB mit OPEN geöffnet und der erste Record mit GET geholt und mittels PUT über die CTC-Verbindung geschickt.
Befindet sich die Mother Task TCPOPROC im SHUTDOWN, wird dies dem IJP-DSP mittels WTOR mitgeteilt; das DSP löscht daraufhin sämtliche Records dieses Jobs. Die DCB's werden mit CLOSE geschlossen und es erfolgt RETURN (ARETURN) nach TCPOPROC. Ist dagegen TCPOPROC in Ordnung, wird solange übertragen, bis End of Data (EOD) auftritt. In diesem Fall wird das IJP-DSP gefragt ob alle Daten angekommen sind. Ist das der Fall, erfolgt RETURN nach TCPOPROC. Sind nicht alle Daten angekommen, muß die Übertragung bei BGNIO wiederholt werden.

Ist bei den OPTION-Parametern CHECKPOINTING angegeben,dann werden die Parameter zunächst auf den CKPT-Data Set gerettet. Dann wird geprüft, ob diese Task die erste IJPWTR-Task darstellt, die die rufende Task initialisiert hat. Ist das nicht der Fall, dann wird die rufende Task (TCPOPROC) mit POST über das Retten der Parameter benachrichtigt. Ist es dagegen die erste initialisierte IJPWTR-Task, dann wird der erste zu bearbeitende Record des CKPT-Data-Sets gelesen und entsprechend wie bei sequentieller Abarbeitung behandelt.

Anhang F

Abbildungen

KONTROLLFLUSS IN TCP

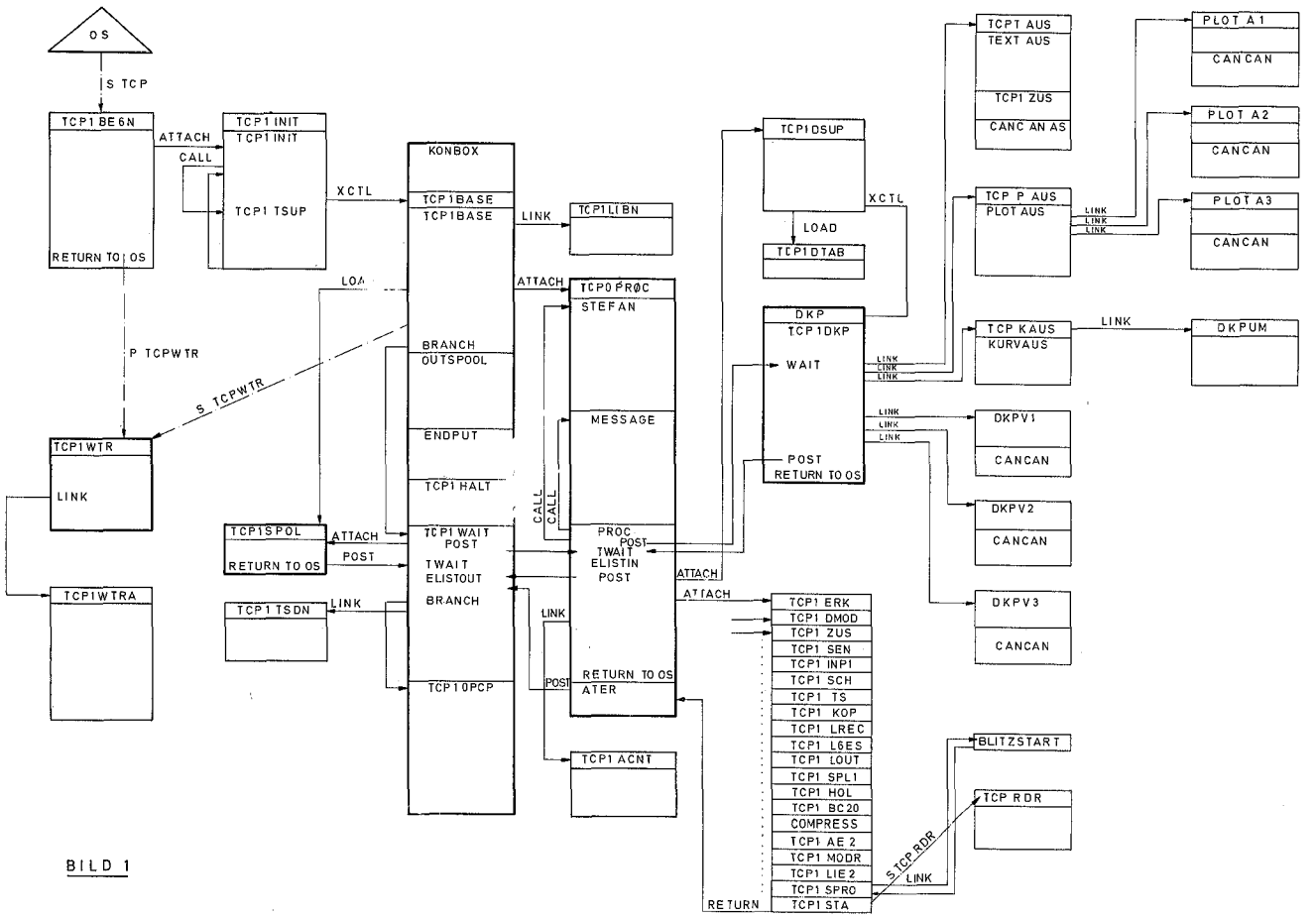
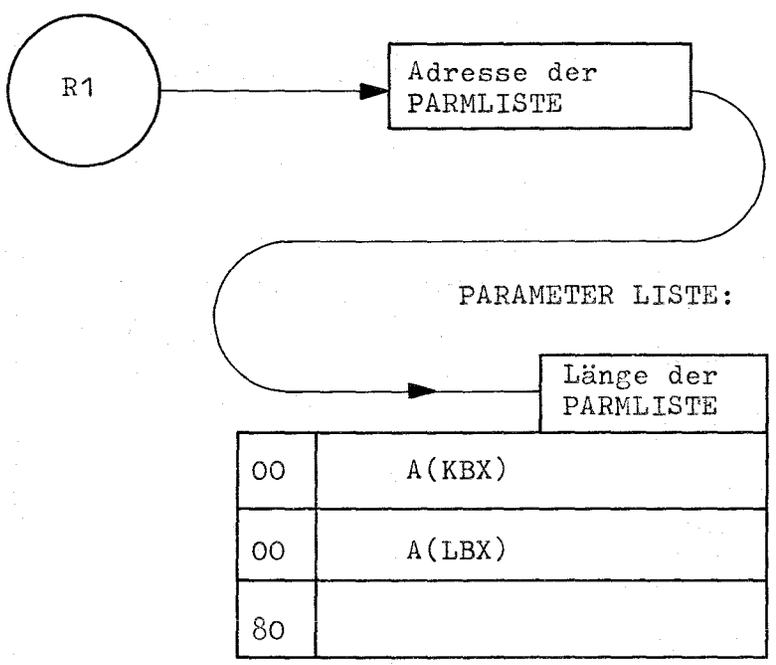


BILD 1

Bild 3

Parameter - Übergabe von TCP1DSUP nach DKP:



Die Adresse der aktuellem LBX kann erst nach der Erteilung eines Display-Auftrags von TCP0PROC an DKP eingetragen werden.

SPEICHERBELEGUNG BEI DKPUM

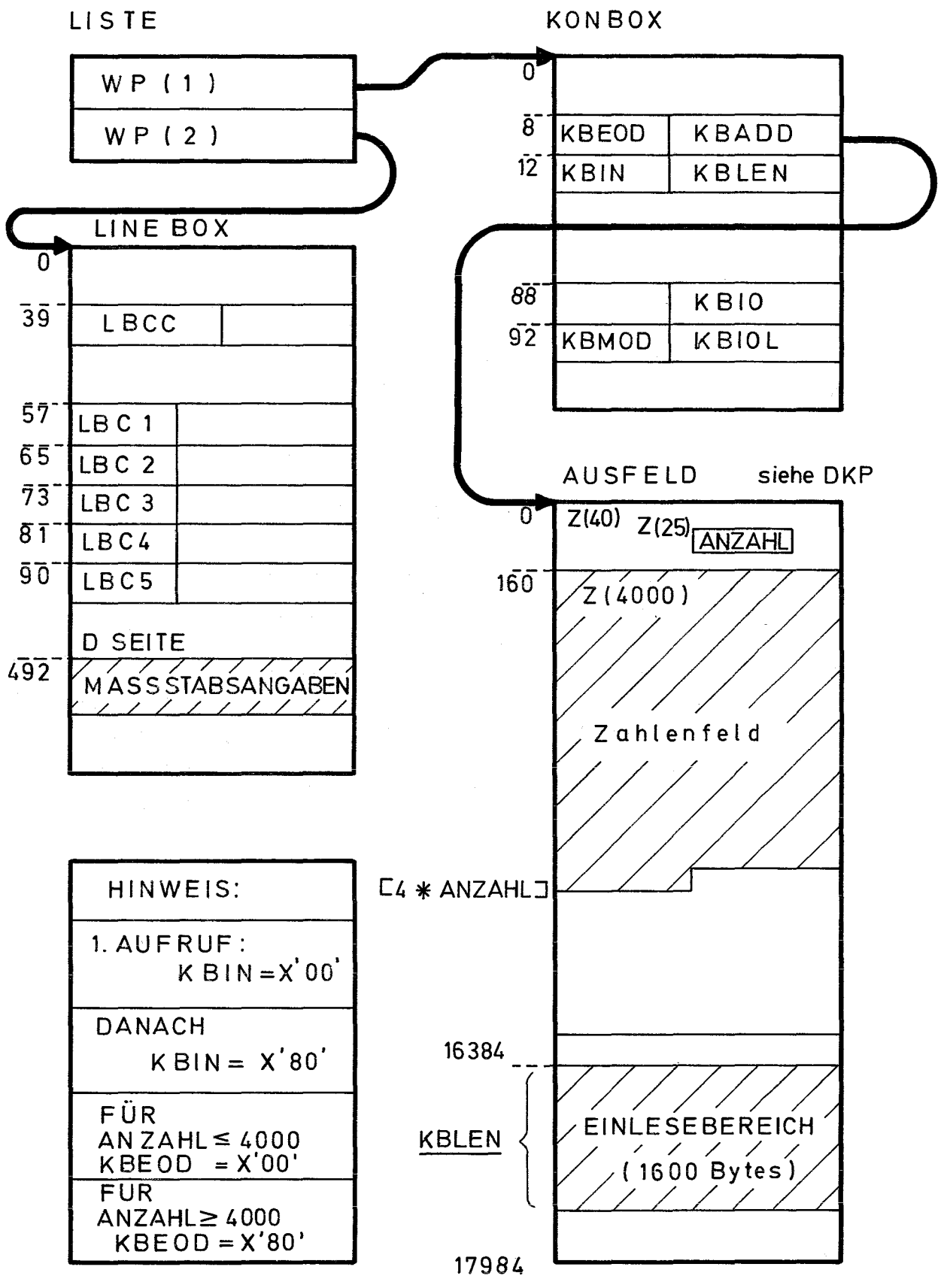


Bild 4

SPEICHERBELEGUNG BEI DKPV1÷DKPV3

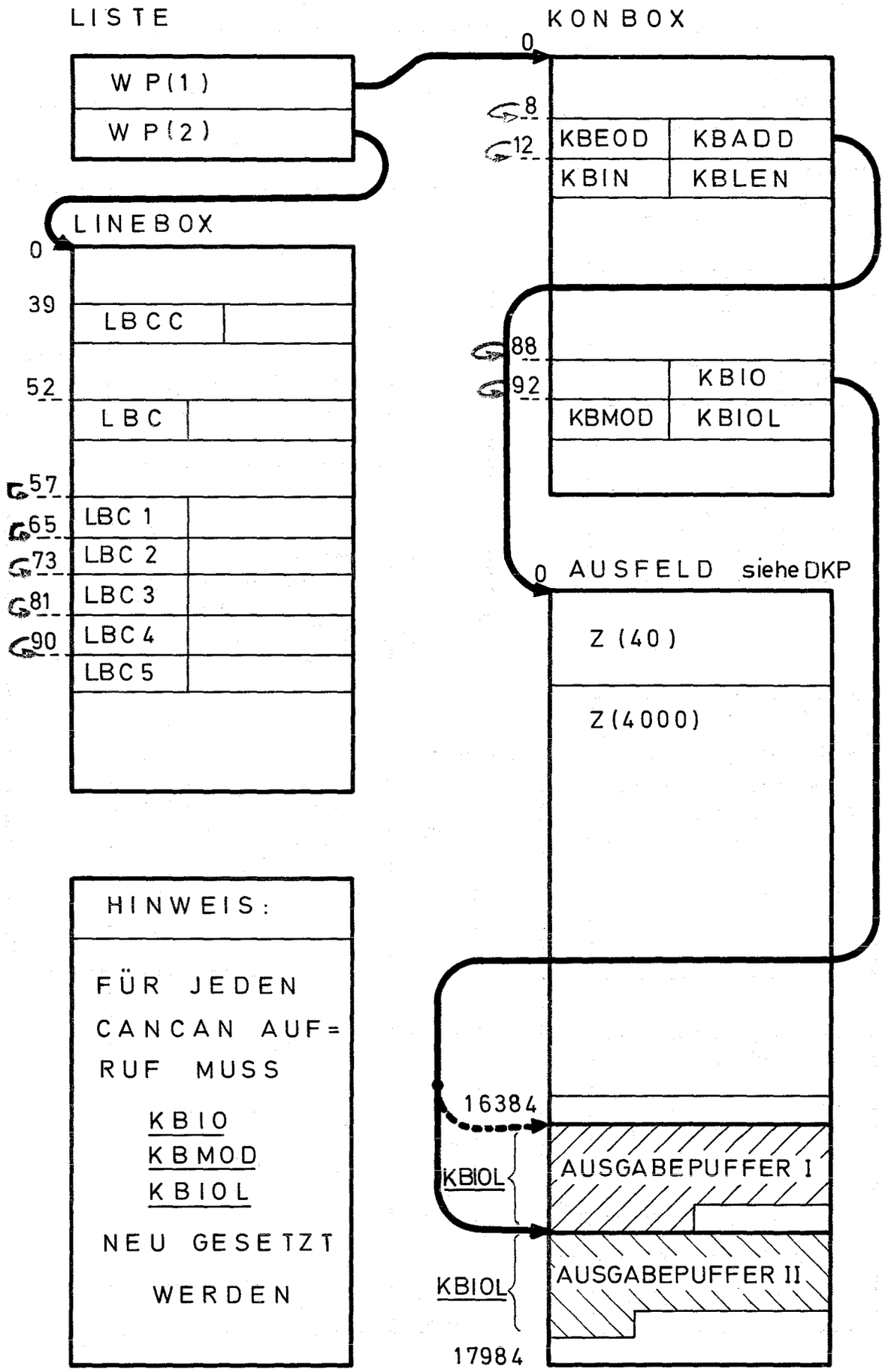


Bild 5

SPEICHERBELEGUNG BEI PLOTA 1 ··· PLOTA 3

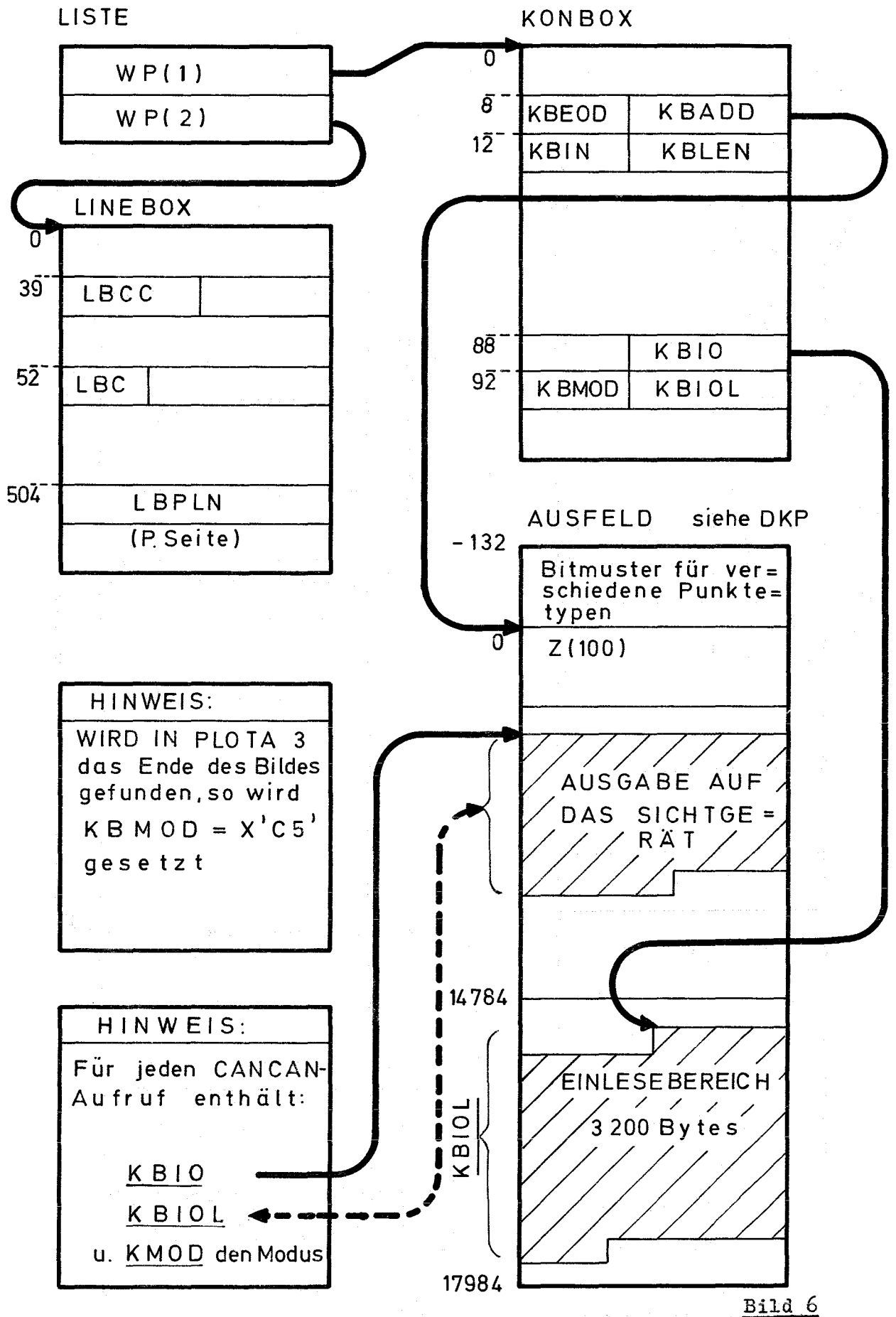


Bild 6

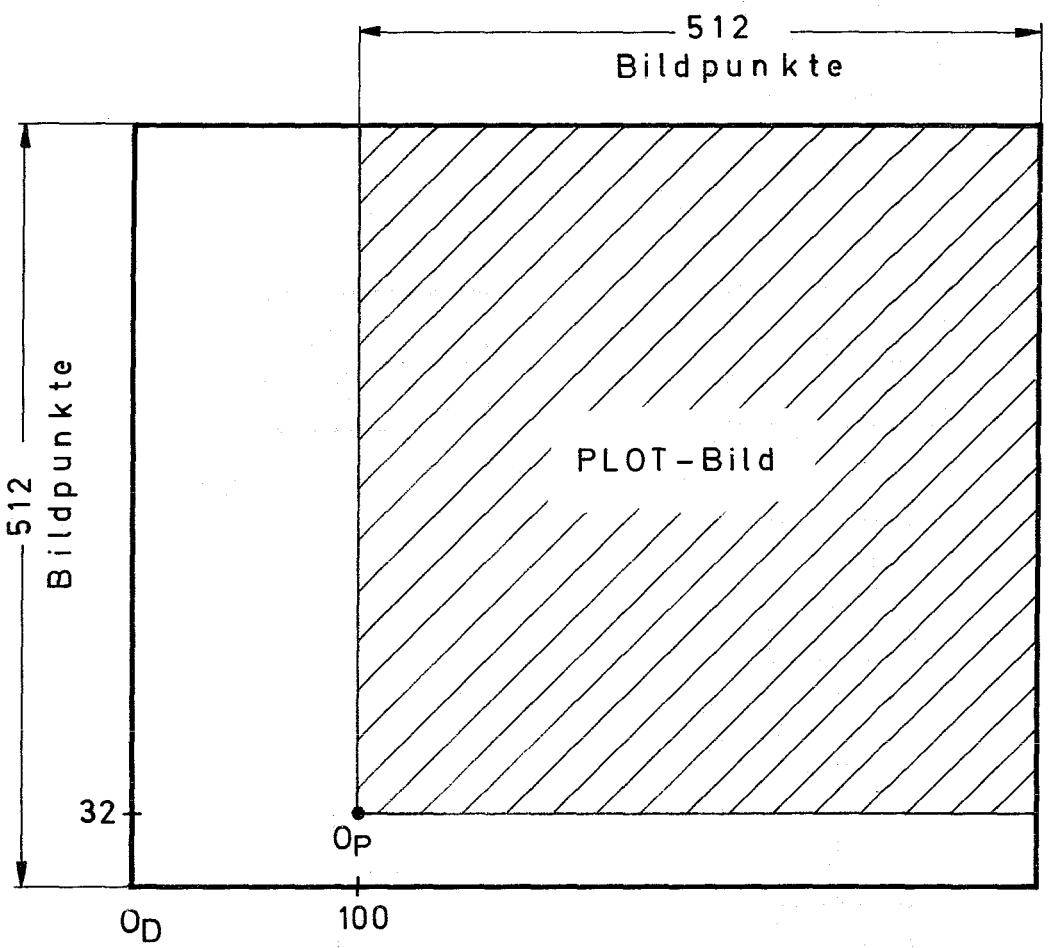
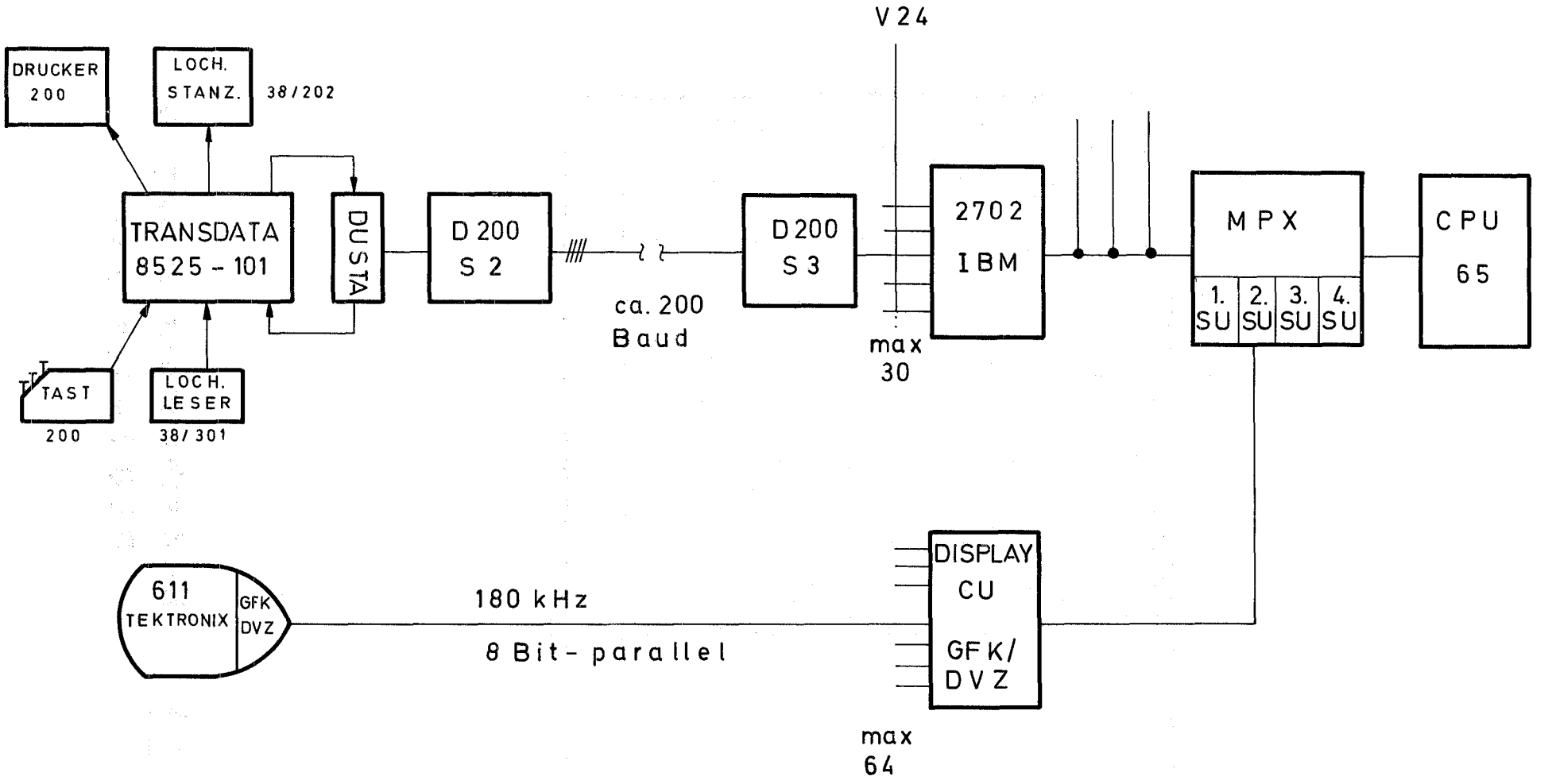


Bild 7: Bildschirmaufteilung bei PLOT-Ausgabe

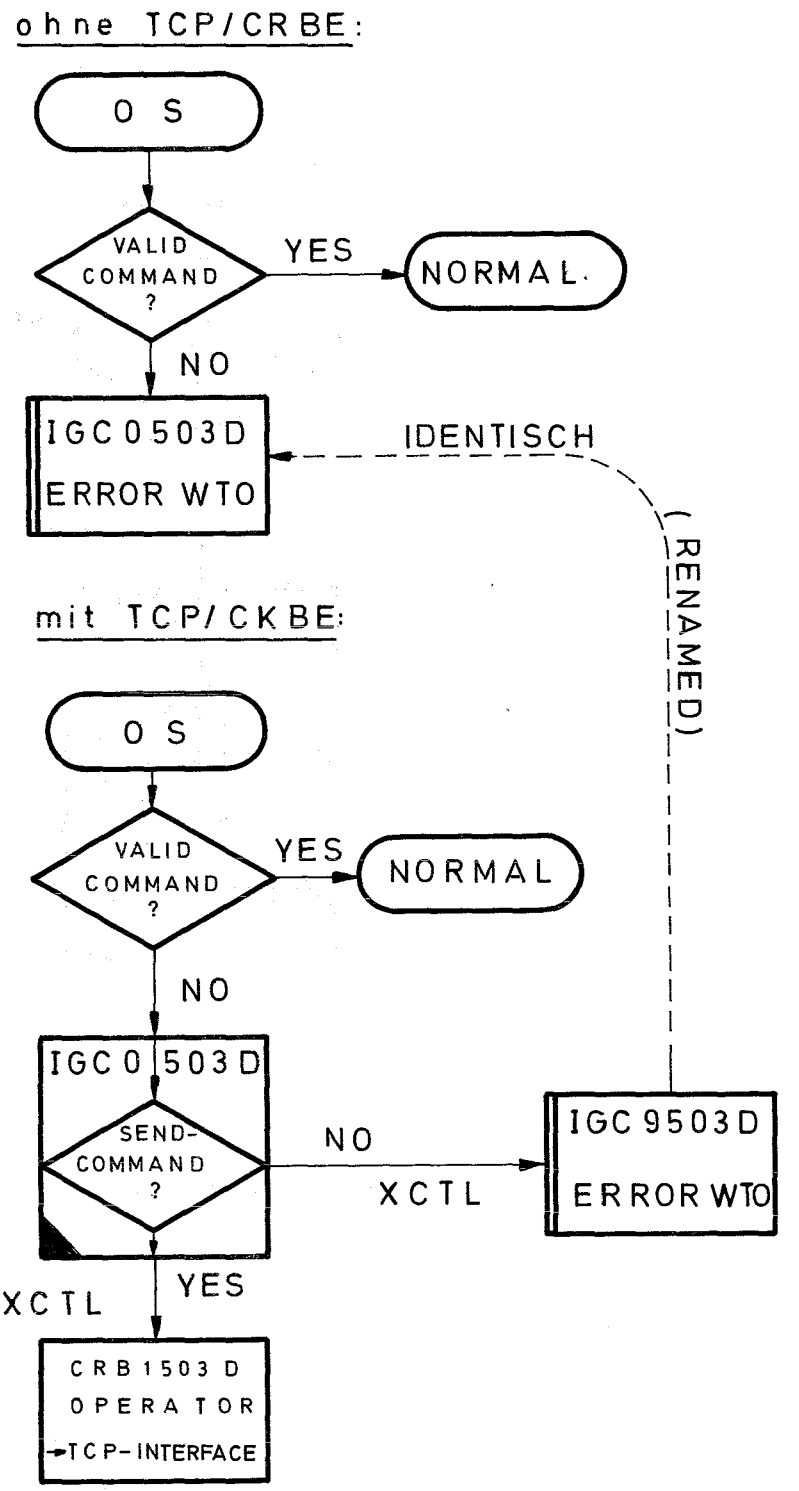


TERMINAL ANSCHLUSS FÜR SIEMENS TRANSDATA 8525 101
UND DVZ SICHTGERÄT AN DIE IBM 360/65 BEI GFK/DVZ

CONSOLE INTERCEPT LOGIC

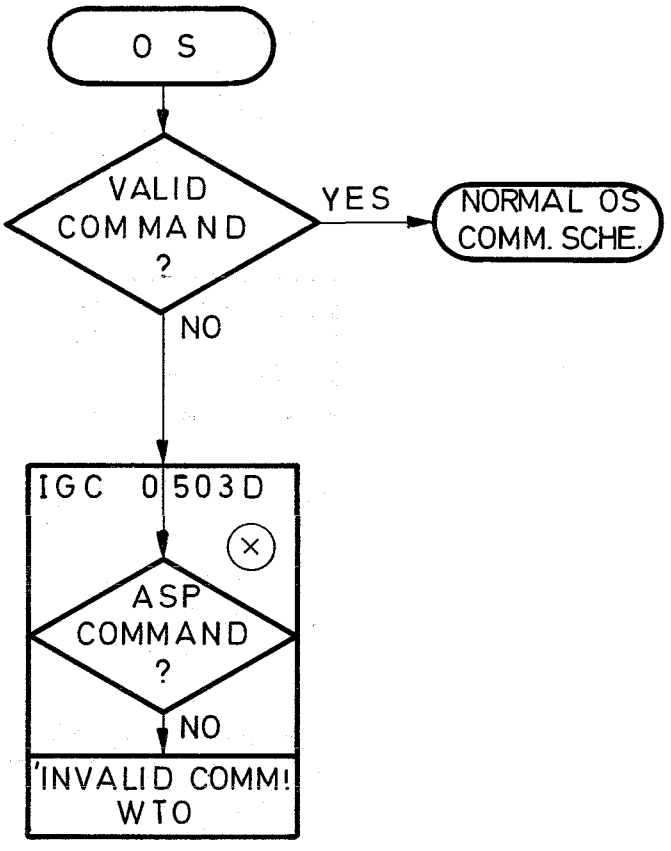
SEND – Commands

TCP/CRBE – bedingte OS – Änderung



CONSOLE INTERCEPT LOGIC BEI TCP/ASP

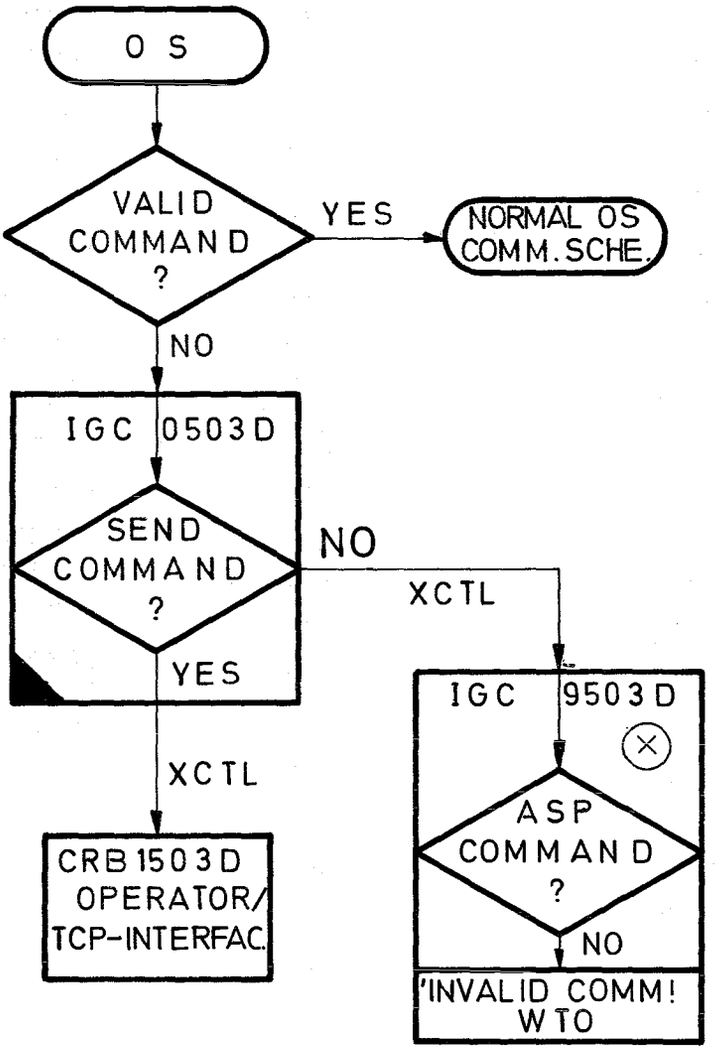
mit ASP, ohne TCP (CRBE):



⊗ siehe nächste Seite

Bild 13 c

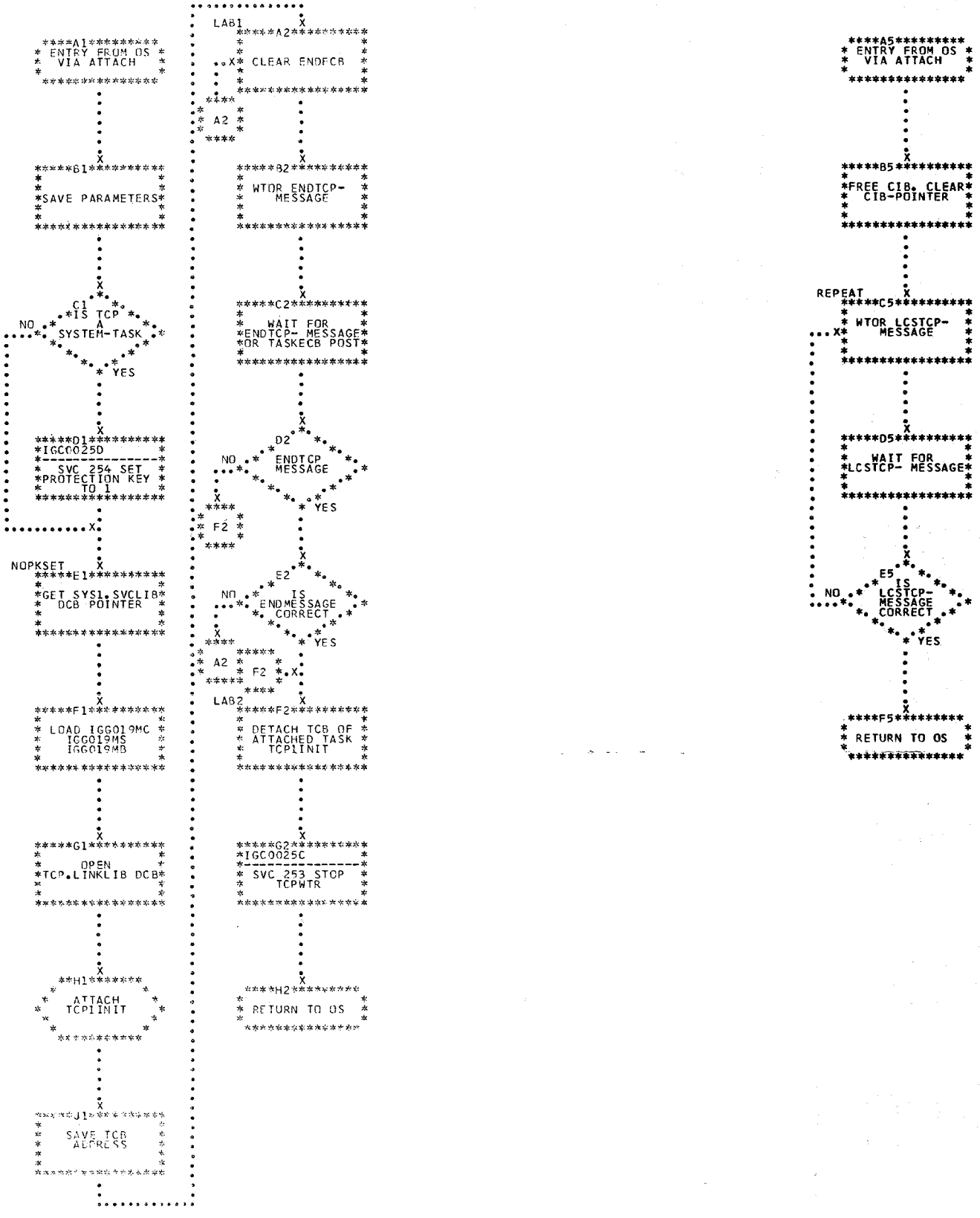
mit ASP, mit TCP (CRBE):

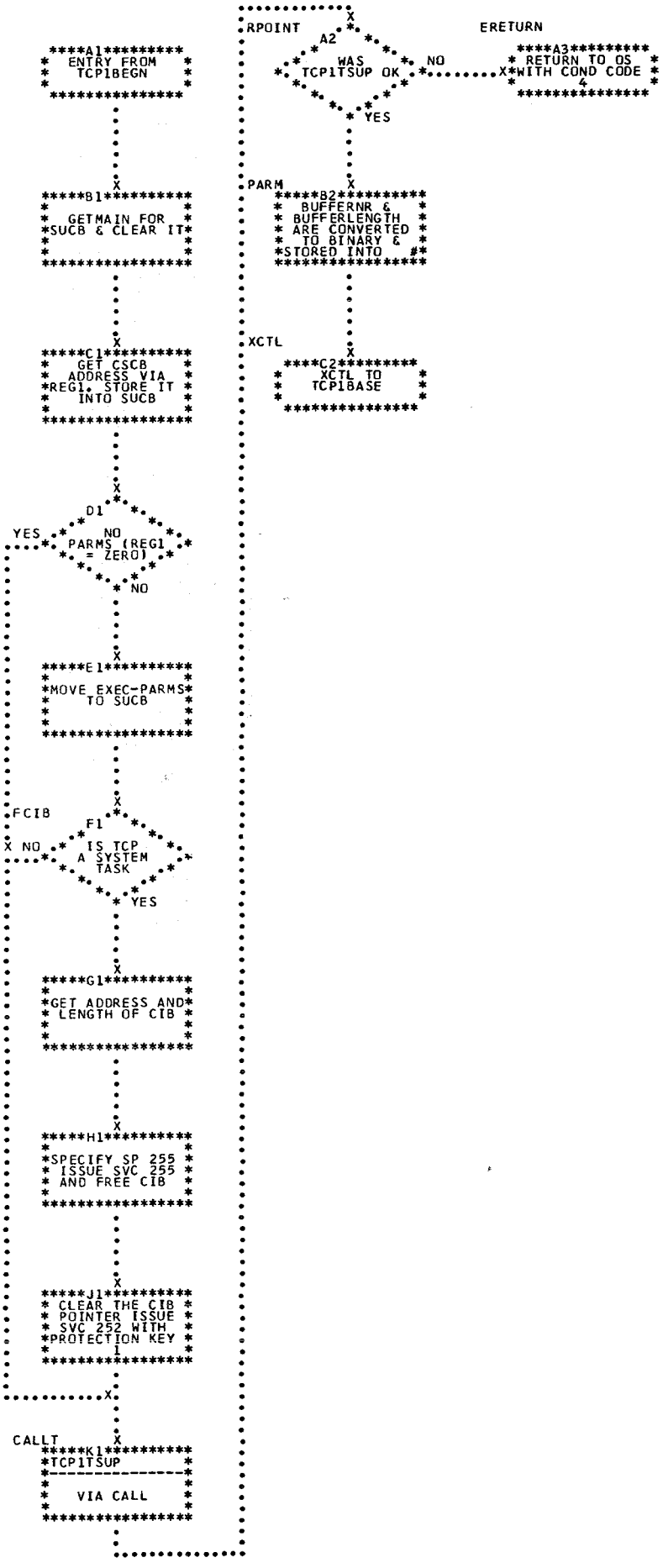


(×) beide Moduln sind identisch

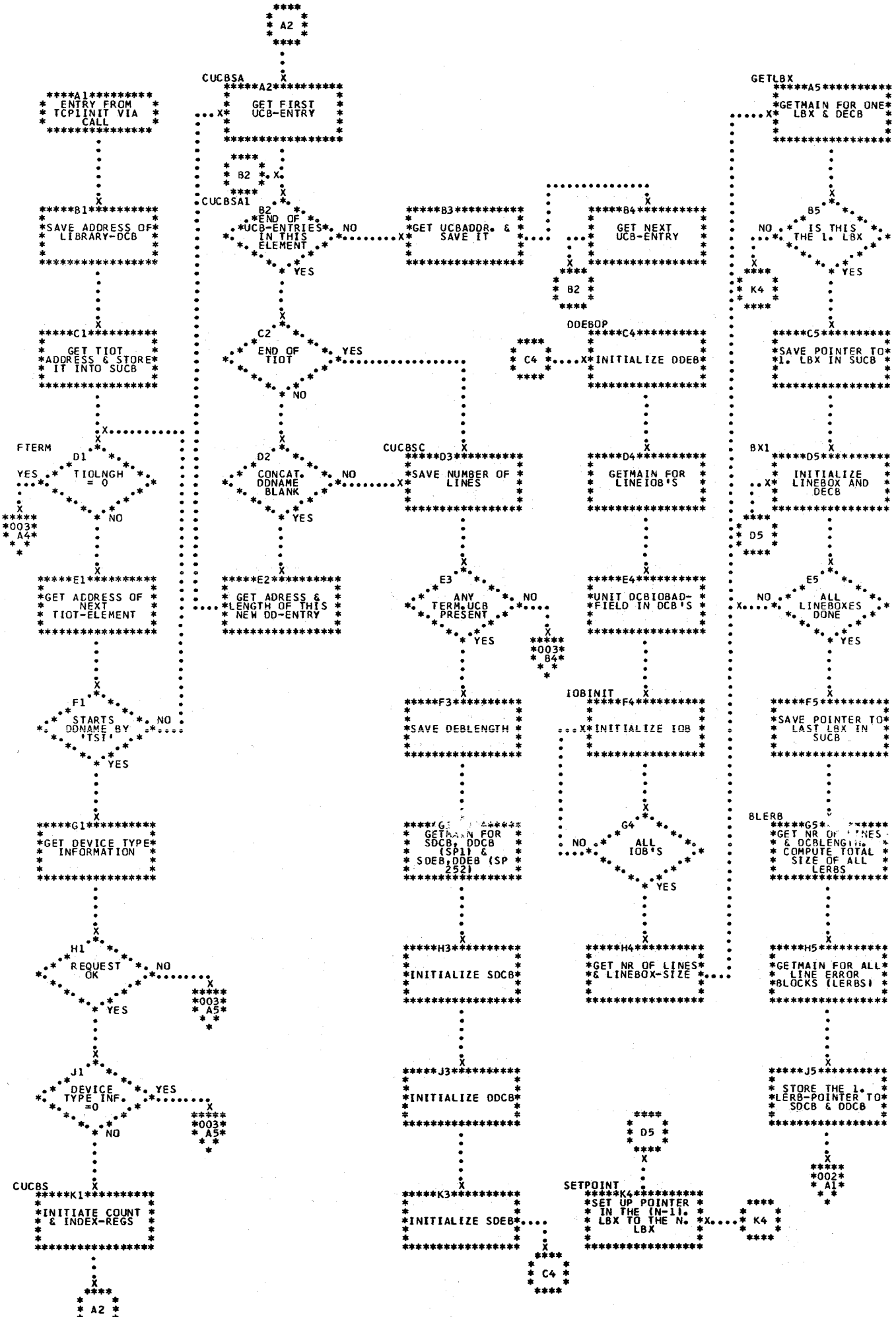
Anhang G

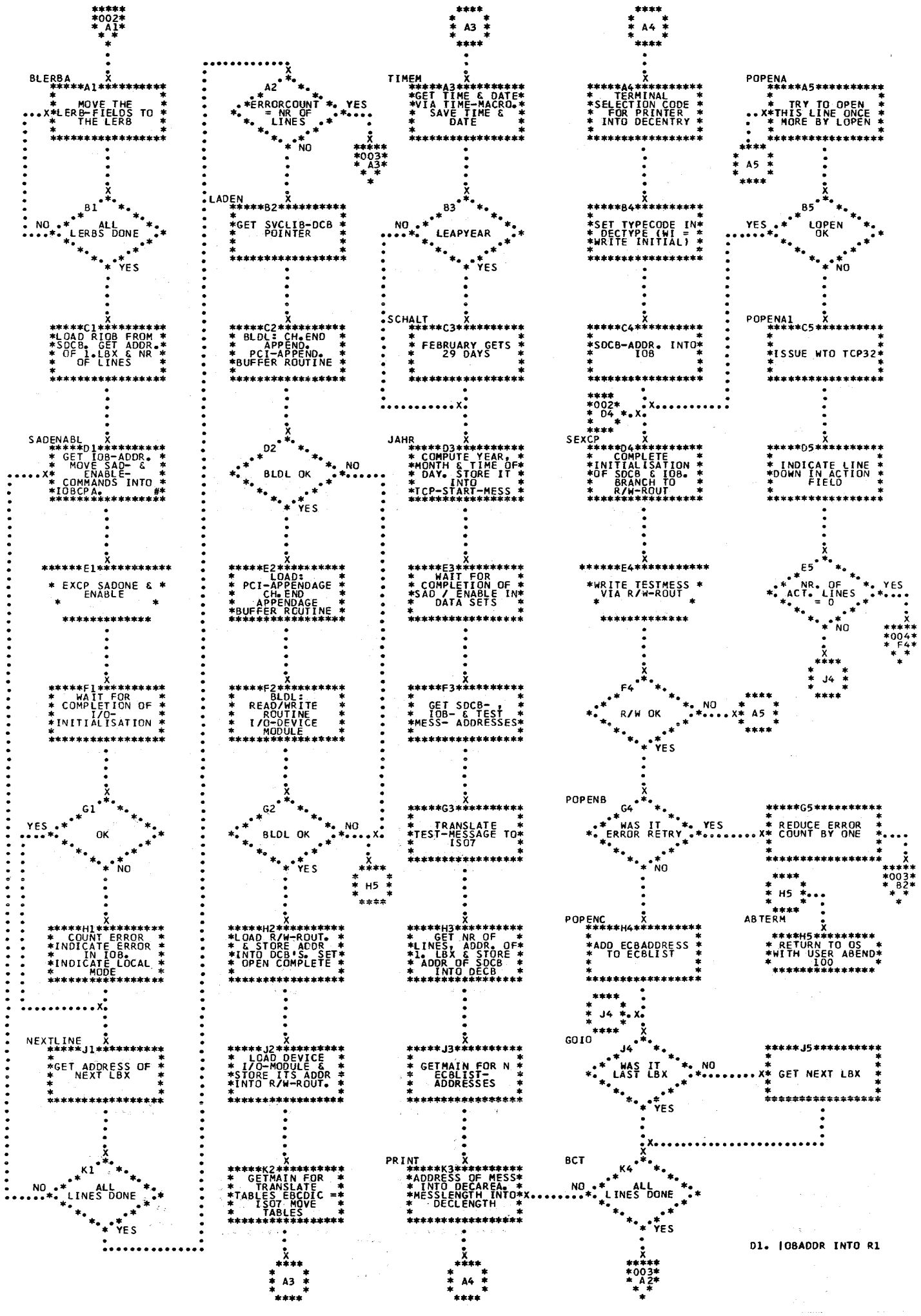
Flußdiagramme	Seite
TCP1BEGN, TCP2BEGN	293
TCP1INIT	294
TCP1BASE	299
TCP1SPOL	324
TCPOPROC	333
TCP1ACNT	339
TCP1ERK	341
TCP1ZUS	342
TCP1LIE2	343
TCP1AEN2	344
TCP1INP1	346
TCP1MODR	347
TCP1SCH	348
TCP1TS	349
TCP1LREC	350
TCP1LOUT	352
TCP1KOP	353
TCP1DMOD	354
TCP1HOL	355
TCP1BC20	356
COMPRESS	357
TCP1DSUP	361
DKP	363
TCPTAUS	366
TCPKAUS	368
DKPUM	370
DKPV1	372
CANCAN	377
TCPPAUS	378
PLOTA1	380
PLOTA2	383
PLOTA3	385
TCP1WTR	386
TCP1WTRA	387
TCP1TSDN	394
XIAK12	395
XIAK24	397
XCAE	399
TCPOPROC (STARTPRO)	401
IJPWTR	402
CONSOLE INTERCEPT LOGIC	405
TCP-SVC'S	406

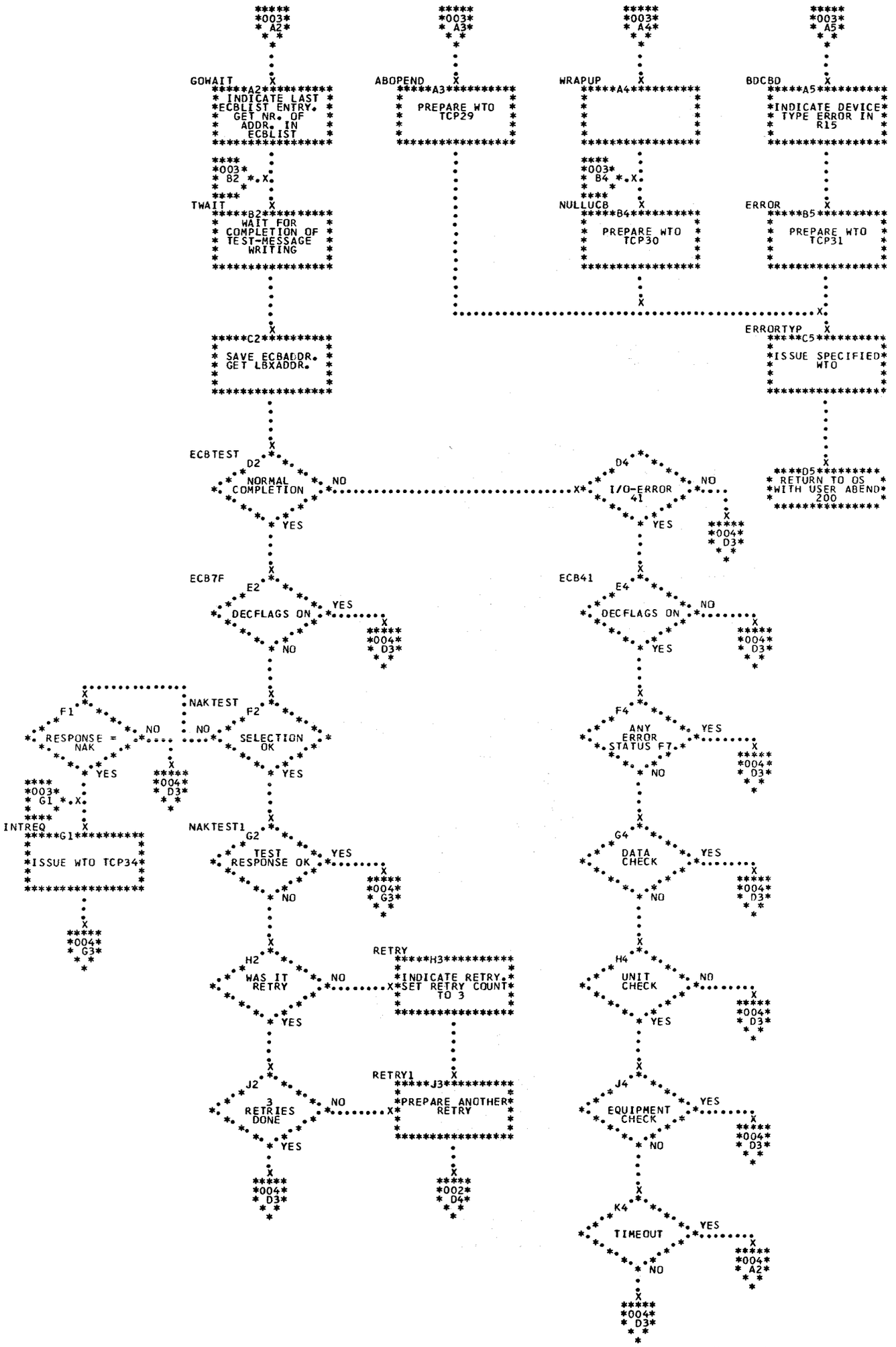

A1
ENTRY FROM OS VIA ATTACH
B1
SAVE PARAMETERS
C1
IS TCP A SYSTEM-TASK
NO
YES
D1
IGC0025D
SVC 254 SET PROTECTION KEY TO 1
NOPKSET
E1
GET SYS1.SVCLIB DCB POINTER
F1
LOAD IGG019MC IGG019MS IGG019MB
G1
OPEN TCP.LINKLIB DCB
H1
ATTACH TCP1INIT
J1
SAVE TCB ADDRESS
LAB1
A2
CLEAR ENDECB
A2
B2
WTOR ENDTCP-MESSAGE
C2
WAIT FOR ENDTCP- MESSAGE OR TASKECB POST
D2
ENDTCP MESSAGE
NO
YES
F2
E2
IS ENDMESSAGE CORRECT
NO
YES
A2
F2
LAB2
F2
DETACH TCB OF ATTACHED TASK TCP1INIT
G2
IGC0025C
SVC 253 STOP TCPWTR
H2
RETURN TO OS
A5
ENTRY FROM OS VIA ATTACH
B5
FREE CIB, CLEAR CIB-POINTER
REPEAT
C5
WTOR LCSTCP-MESSAGE
D5
WAIT FOR LCSTCP- MESSAGE
E5
IS LCSTCP-MESSAGE CORRECT
NO
YES
F5
RETURN TO OS

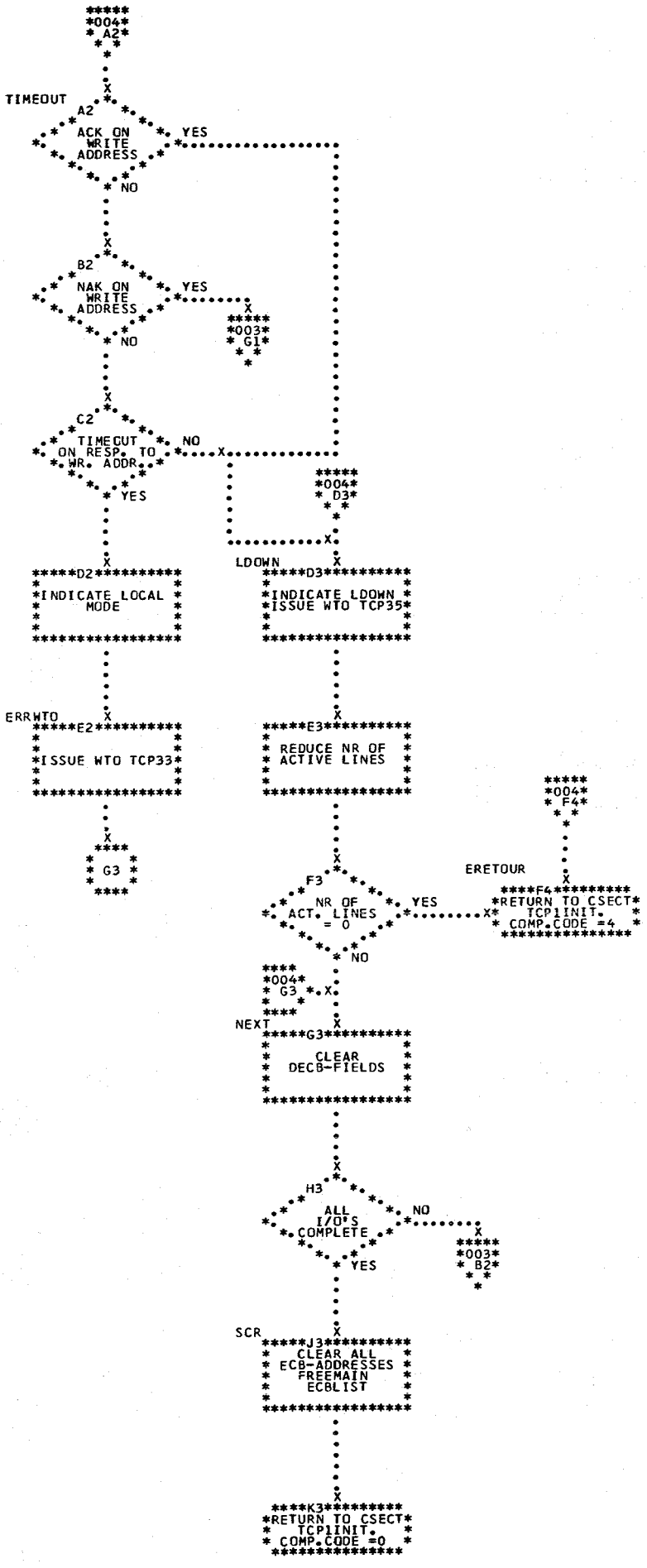

RPOINT
ERETURN
A2
WAS TCP1TSUP OK
NO
YES
A1 ENTRY FROM TCP1BEGN
A3 RETURN TO OS WITH COND CODE 4
PARM
B1 GETMAIN FOR SUCB & CLEAR IT
B2 BUFFERNR & BUFFERLENGTH ARE CONVERTED TO BINARY & STORED INTO #
XCTL
C1 GET CSCB ADDRESS VIA REG1, STORE IT INTO SUCB
C2 XCTL TO TCP1BASE
D1
YES
NO
PARMS (REG1 = ZERO)
NO
E1 MOVE EXEC-PARMS TO SUCB
FCIB
F1
X NO
IS TCP A SYSTEM TASK
YES
G1 GET ADDRESS AND LENGTH OF CIB
H1 SPECIFY SP 255 ISSUE SVC 255 AND FREE CIB
J1 CLEAR THE CIB POINTER ISSUE SVC 252 WITH PROTECTION KEY 1
CALLT
K1 TCP1TSUP
VIA CALL

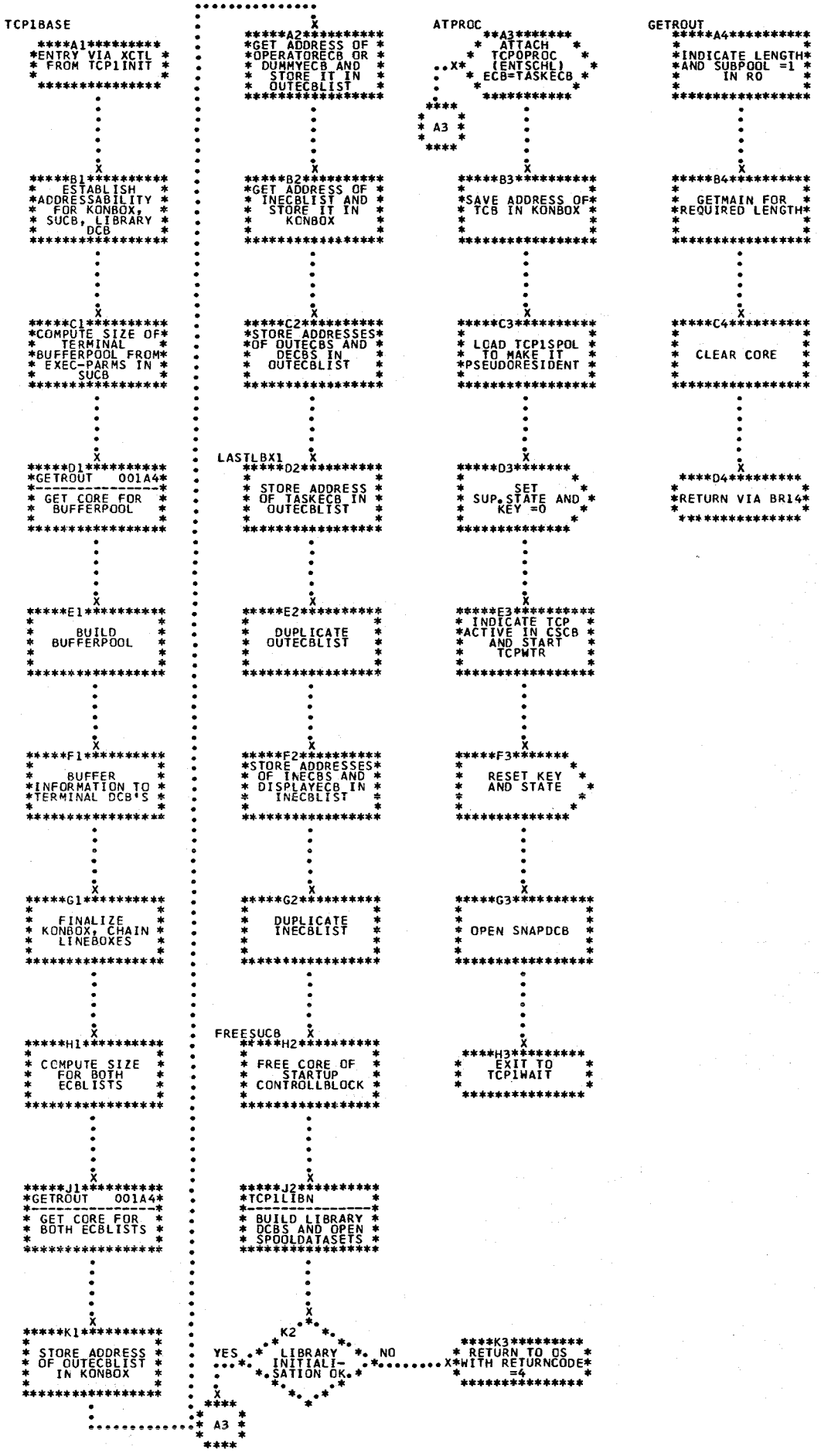
B2. SUCB

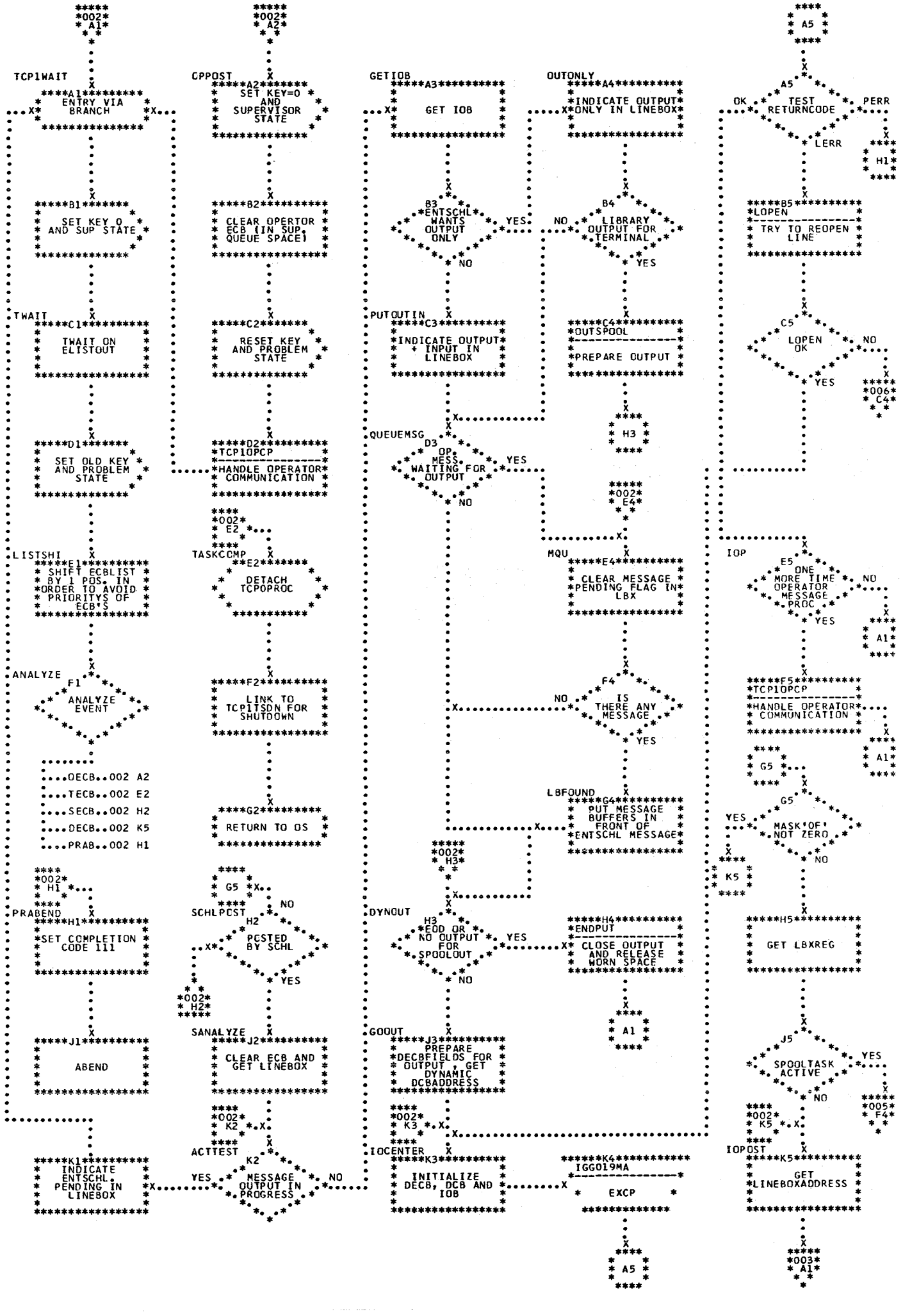

A2
A1 ENTRY FROM TCP1INIT VIA CALL
CUCBSA
A2 GET FIRST UCB-ENTRY
B2
CUCBSA1
B1 SAVE ADDRESS OF LIBRARY-DCB
B2 END OF UCB-ENTRIES IN THIS ELEMENT
NO
YES
B3 GET UCBADDR. & SAVE IT
B4 GET NEXT UCB-ENTRY
B2
C1 GET TIOT ADDRESS & STORE IT INTO SUCB
C2 END OF TIOT
YES
NO
DDEBOP
C4 INITIALIZE DDEB
C4
FTERM
D1 TIOLNGH = 0
YES
NO
003 A4
CUCBSC
D2 CONCAT. DDNAME BLANK
NO
YES
D3 SAVE NUMBER OF LINES
D4 GETMAIN FOR LINEIOB'S
E1 GET ADDRESS OF NEXT TIOT-ELEMENT
E2 GET ADRESS & LENGTH OF THIS NEW DD-ENTRY
E3 ANY TERM.UCB PRESENT
NO
YES
003 B4
E4 UNIT DCBIOBAD- FIELD IN DCB'S
F1 STARTS DDNAME BY 'TSI'
NO
YES
F3 SAVE DEBLENGTH
IOBINIT
F4 INITIALIZE IOB
G1 GET DEVICE TYPE INFORMATION
G3 GETMAIN FOR SDCB, DDCB (SP1) & SDEB,DDEB (SP 252)
G4 ALL IOB'S
NO
YES
H1 REQUEST OK
NO
YES
003 A5
H3 INITIALIZE SDCB
H4 GET NR OF LINES & LINEBOX-SIZE
J1 DEVICE TYPE INF. =0
YES
NO
003 A5
J3 INITIALIZE DDCB
D5
SETPOINT
K1 INITIATE COUNT & INDEX-REGS
K3 INITIALIZE SDEB
K4 SET UP POINTER IN THE (N-1). LBX TO THE N. LBX
K4
C4
A2
GETLBX
A5 GETMAIN FOR ONE LBX & DECB
B5 IS THIS THE 1. LBX
NO
YES
K4
C5 SAVE POINTER TO 1. LBX IN SUCB
BX1
D5 INITIALIZE LINEBOX AND DECB
D5
E5 ALL LINEBOXES DONE
NO
YES
F5 SAVE POINTER TO LAST LBX IN SUCB
BLERB
G5 GET NR OF LINES & DCBLENGTH. COMPUTE TOTAL SIZE OF ALL LERBS
H5 GETMAIN FOR ALL LINE ERROR BLOCKS (LERBS)
J5 STORE THE 1. LERB-POINTER TO SDCB & DDCB
002 A1

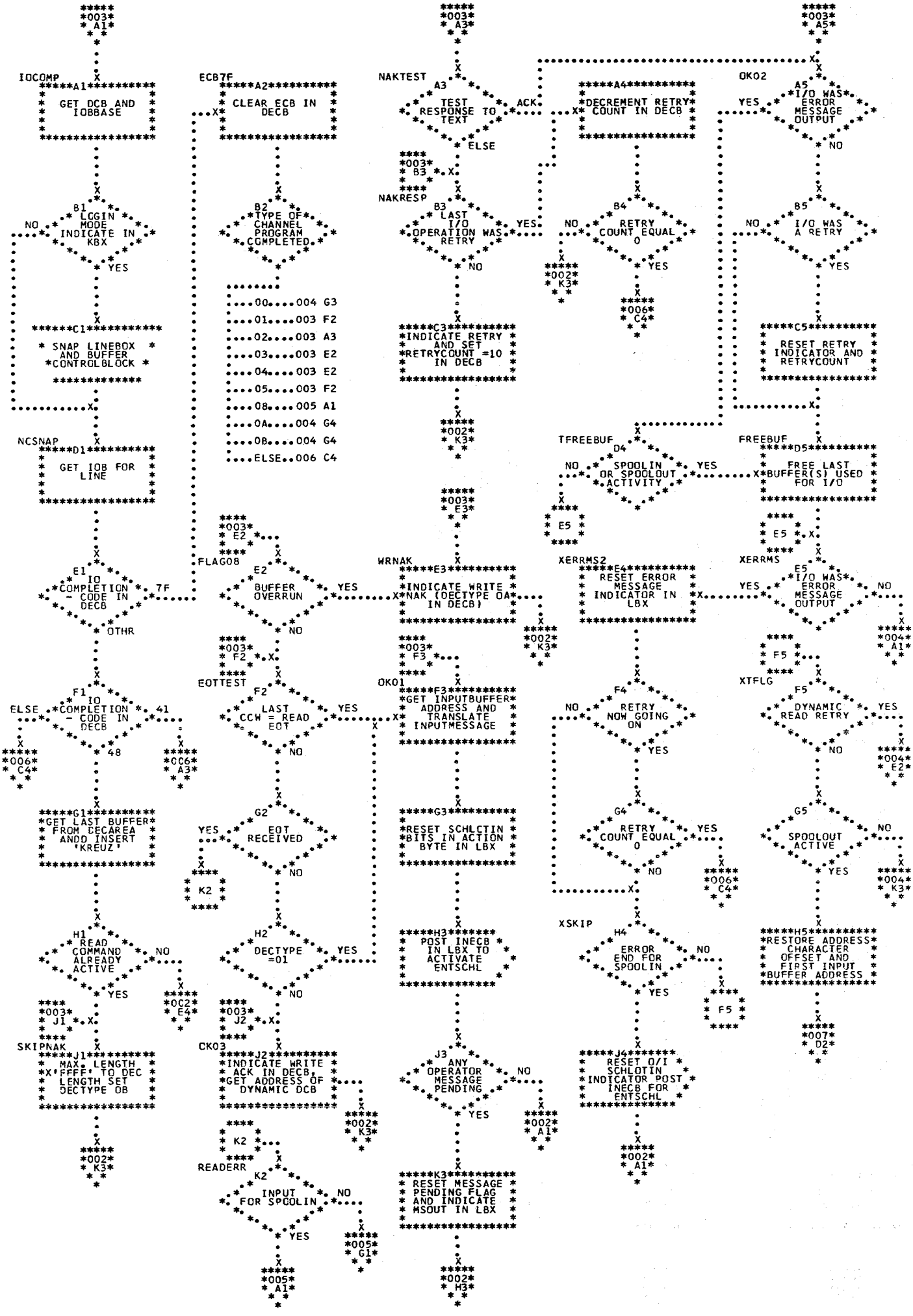

002 A1
BLERBA
A1 MOVE THE LERB-FIELDS TO THE LERB
B1 ALL LERBS DONE
NO
YES
C1 LOAD RIOB FROM SDCB. GET ADDR. OF 1.LBX & NR OF LINES
SADENABL
D1 GET IOB-ADDR. MOVE SAD- & ENABLE- COMMANDS INTO IOBCPA.
E1 EXCP SADONE & ENABLE
F1 WAIT FOR COMPLETION OF I/O-INITIALISATION
G1 OK
H1 COUNT ERROR INDICATE ERROR IN IOB. INDICATE LOCAL MODE
NEXTLINE
J1 GET ADDRESS OF NEXT LBX
K1 ALL LINES DONE
A2 ERRORCOUNT = NR OF LINES
003 A3
LADEN
B2 GET SVCLIB-DCB POINTER
C2 BLDL: CH.END APPEND. PCI-APPEND. BUFFER ROUTINE
D2 BLDL OK
E2 LOAD: BLDL: PCI-APPENDAGE CH.END APPENDAGE BUFFER ROUTINE
F2 BLDL: READ/WRITE ROUTINE I/O-DEVICE MODULE
G2 BLDL OK
H5
H2 LOAD R/W-ROUT. & STORE ADDR INTO DCB'S. SET OPEN COMPLETE
J2 LOAD DEVICE I/O-MODULE & STORE ITS ADDR INTO R/W-ROUT.
K2 GETMAIN FOR TRANSLATE TABLES EBCDIC =ISO7 MOVE TABLES
A3
A3
TIMEM
A3 GET TIME & DATE VIA TIME-MACRO. SAVE TIME & DATE
B3 LEAPYEAR
SCHALT
C3 FEBRUARY GETS 29 DAYS
JAHR
D3 COMPUTE YEAR, MONTH & TIME OF DAY. STORE IT INTO TCP-START-MESS
E3 WAIT FOR COMPLETION OF *SAD / ENABLE IN* DATA SETS
F3 GET SDCB- IOB- & TEST MESS- ADDRESSES
G3 TRANSLATE TEST-MESSAGE TO ISO7
H3 GET NR OF LINES, ADDR. OF 1. LBX & STORE ADDR OF SDCB INTO DECB
J3 GETMAIN FOR N ECBLIST-ADDRESSES
PRINT
K3 ADDRESS OF MESS INTO DECAREA. MESSLENGTH INTO DECLENGTH
A4
A4
A4 TERMINAL SELECTION CODE FOR PRINTER INTO DECENTRY
B4 SET TYPECODE IN DECTYPE (WI = WRITE INITIAL)
C4 SDCB-ADDR. INTO IOB
002 D4
SEXCP
D4 COMPLETE INITIALISATION OF SDCB & IOB. BRANCH TO R/W-ROUT
E4 WRITE TESTMESS VIA R/W-ROUT
F4 R/W OK
A5
POPENB
G4 WAS IT ERROR RETRY
H4 ADD ECBADDRESS TO ECBLIST
J4
GOIO
J4 WAS IT LAST LBX
K4 ALL LINES DONE
BCT
003 A2
POPENA
A5 TRY TO OPEN THIS LINE ONCE MORE BY LOPEN
B5 LOPEN OK
POPENA1
C5 ISSUE WTO TCP32
D5 INDICATE LINE DOWN IN ACTION FIELD
E5 NR. OF ACT. LINES = 0
004 F4
J4
G5 REDUCE ERROR COUNT BY ONE
003 B2
H5
ABTERM
H5 RETURN TO OS WITH USER ABEND IOO
J5 GET NEXT LBX
D1. IOBADDR INTO R1

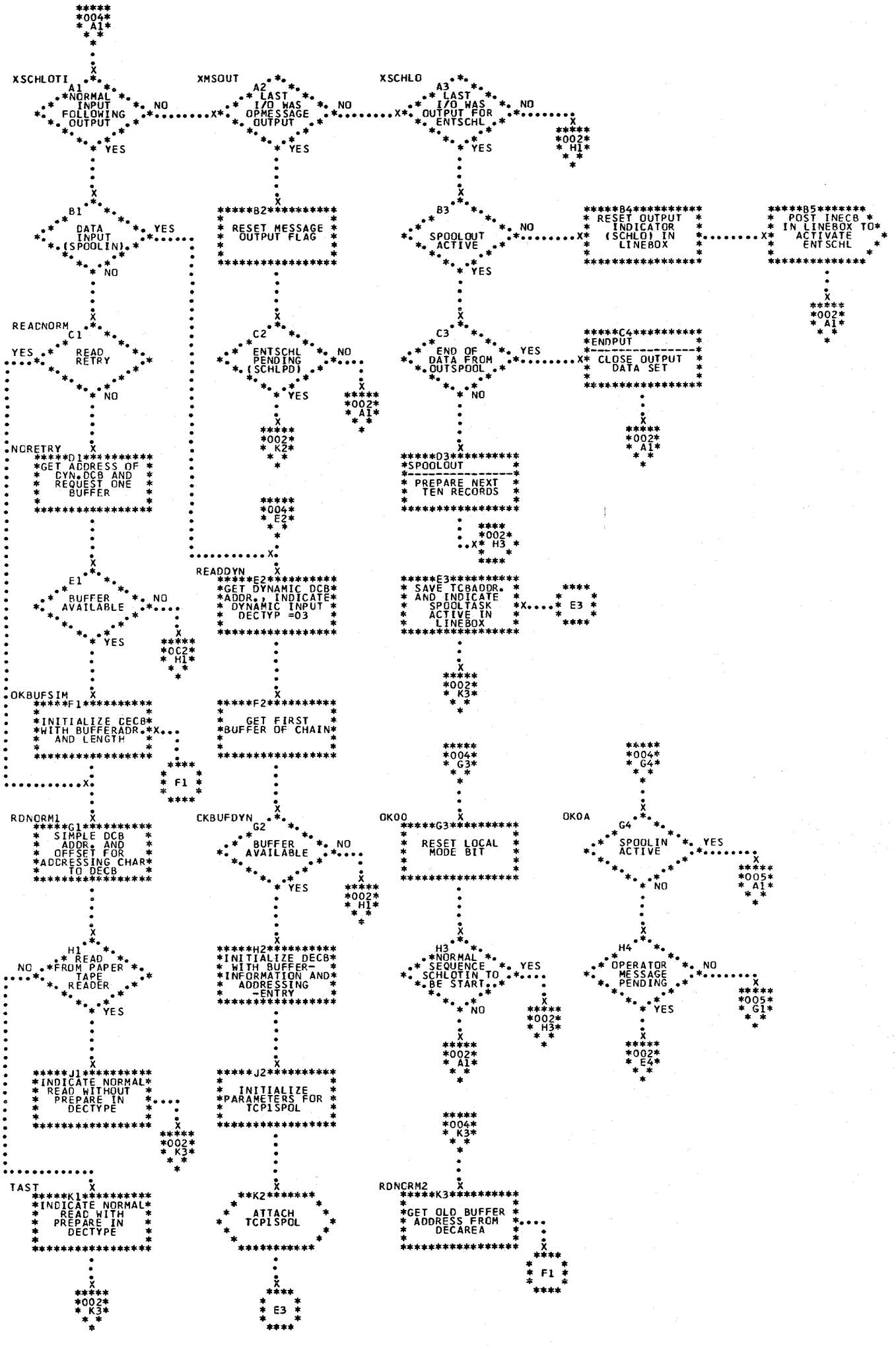

003 * A2*
GOWAIT
INDICATE LAST ECBLIST ENTRY. GET NR. OF ADDR. IN ECBLIST
003 * B2 *
TWAIT
WAIT FOR COMPLETION OF TEST-MESSAGE WRITING
SAVE ECBADDR. GET LBXADDR.
ECBTEST
D2
NORMAL COMPLETION
NO
YES
ECB7F
E2
DECFLAGS ON
YES
NO
004 * D3*
NAKTEST
F2
SELECTION OK
NO
YES
F1
RESPONSE = NAK
NO
YES
003 * G1 *
INTREQ
ISSUE WTO TCP34
004 * G3*
NAKTEST1
G2
TEST RESPONSE OK
YES
NO
004 * G3*
RETRY
H2
WAS IT RETRY
NO
YES
INDICATE RETRY. SET RETRY COUNT TO 3
RETRY1
J2
3 RETRIES DONE
NO
YES
PREPARE ANOTHER RETRY
004 * D3*
002 * D4*
003 * A3*
ABOPEND
PREPARE WTO TCP29
003 * A4*
WRAPUP
003 * B4 *
NULLUCB
PREPARE WTO TCP30
003 * A5*
BDCBD
INDICATE DEVICE TYPE ERROR IN R15
ERROR
PREPARE WTO TCP31
ERRORTYP
ISSUE SPECIFIED WTO
D5
RETURN TO OS WITH USER ABEND 200
D4
I/O-ERROR 41
NO
YES
004 * D3*
ECB41
E4
DECFLAGS ON
NO
YES
004 * D3*
F4
ANY ERROR STATUS F7
YES
NO
004 * D3*
G4
DATA CHECK
YES
NO
004 * D3*
H4
UNIT CHECK
NO
YES
004 * D3*
J4
EQUIPMENT CHECK
YES
NO
004 * D3*
K4
TIMEOUT
YES
NO
004 * A2*
004 * D3*

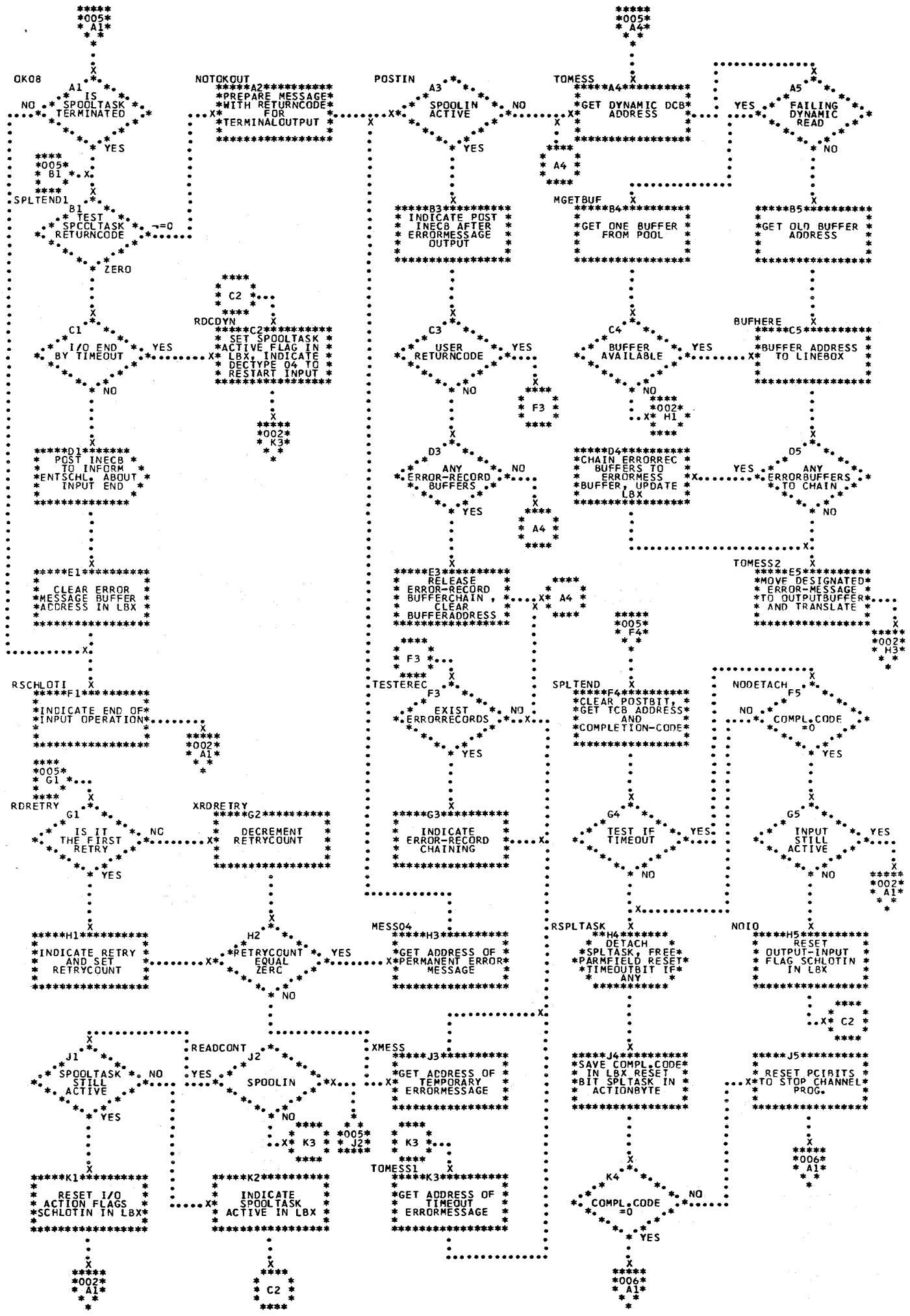

004
A2
TIMEOUT
A2
ACK ON WRITE ADDRESS
YES
NO
B2
NAK ON WRITE ADDRESS
YES
NO
003
G1
C2
TIMEOUT ON RESP. TO WR. ADDR.
NO
YES
004
D3
LDOWN
D2
INDICATE LOCAL MODE
D3
INDICATE LDOWN
ISSUE WTO TCP35
ERRWTO
E2
ISSUE WTO TCP33
E3
REDUCE NR OF ACTIVE LINES
004
F4
ERETOUR
F4
RETURN TO CSECT TCP1INIT. COMP.CODE =4
F3
NR OF ACT. LINES = 0
YES
NO
004
G3
NEXT
G3
CLEAR DECB-FIELDS
H3
ALL I/O'S COMPLETE
NO
YES
003
B2
SCR
J3
CLEAR ALL ECB-ADDRESSES FREEMAIN ECBLIST
K3
RETURN TO CSECT TCP1INIT. COMP.CODE =0

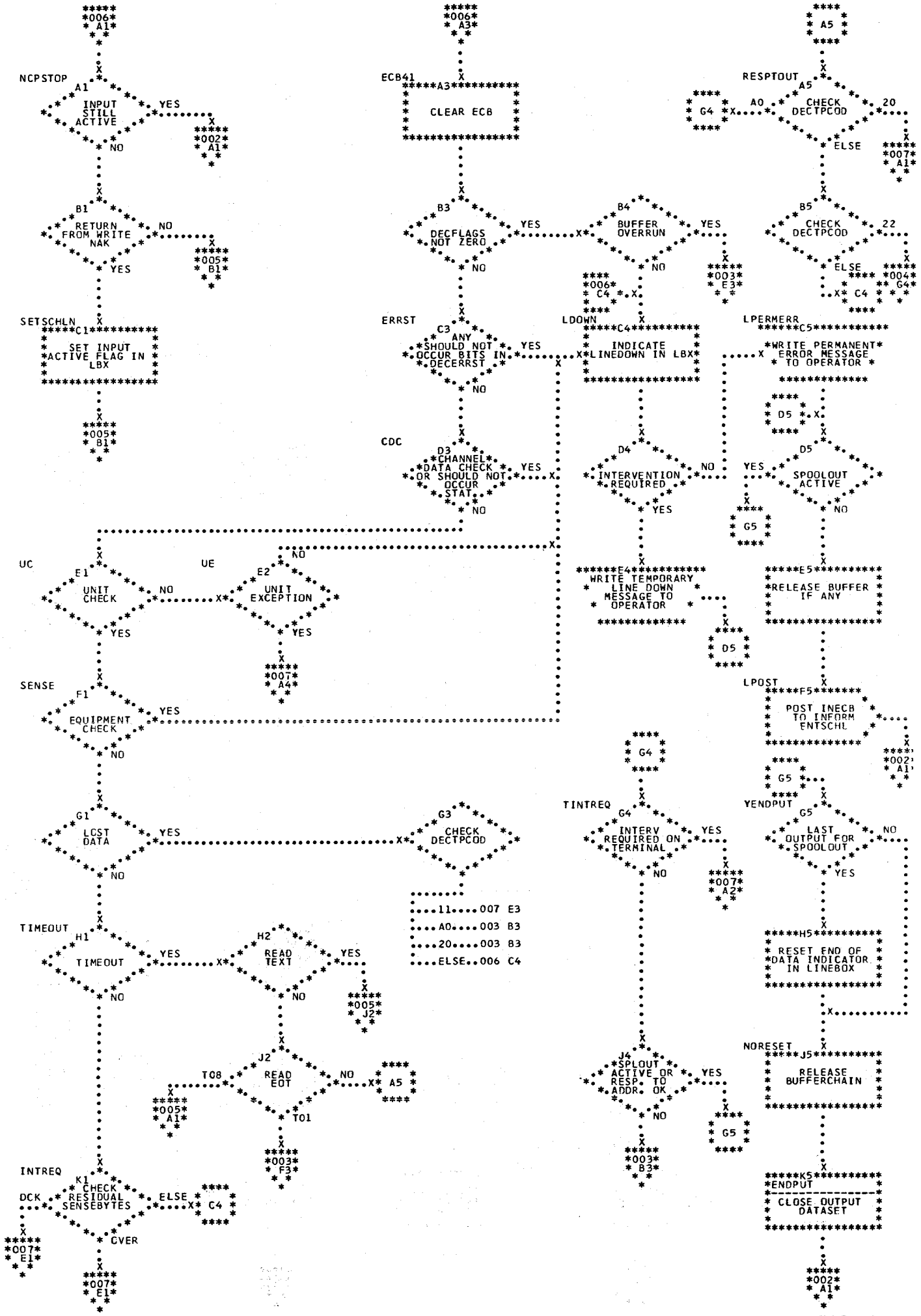

TCP1BASE
A1
ENTRY VIA XCTL FROM TCP1INIT
B1
ESTABLISH ADDRESSABILITY FOR KONBOX, SUCB, LIBRARY DCB
C1
COMPUTE SIZE OF TERMINAL BUFFERPOOL FROM EXEC-PARMS IN SUCB
D1
GETROUT 001A4
GET CORE FOR BUFFERPOOL
E1
BUILD BUFFERPOOL
F1
BUFFER INFORMATION TO TERMINAL DCB'S
G1
FINALIZE KONBOX, CHAIN LINEBOXES
H1
COMPUTE SIZE FOR BOTH ECBLISTS
J1
GETROUT 001A4
GET CORE FOR BOTH ECBLISTS
K1
STORE ADDRESS OF OUTECBLIST IN KONBOX
A2
GET ADDRESS OF OPERATORECB OR DUMMYECB AND STORE IT IN OUTECBLIST
B2
GET ADDRESS OF INECBLIST AND STORE IT IN KONBOX
C2
STORE ADDRESSES OF OUTECBS AND DECBS IN OUTECBLIST
LASTLBX1
D2
STORE ADDRESS OF TASKECB IN OUTECBLIST
E2
DUPLICATE OUTECBLIST
F2
STORE ADDRESSES OF INECBS AND DISPLAYECB IN INECBLIST
G2
DUPLICATE INECBLIST
FREESUCB
H2
FREE CORE OF STARTUP CONTROLLBLOCK
J2
TCP1LIBN
BUILD LIBRARY DCBS AND OPEN SPOOLDATASETS
K2
LIBRARY INITIALISATION OK.
YES
NO
A3
ATPROC
A3
ATTACH TCPOPROC (ENTSCHL) ECB=TASKECB
A3
B3
SAVE ADDRESS OF TCB IN KONBOX
C3
LOAD TCP1SPOL TO MAKE IT PSEUDORESIDENT
D3
SET SUP.STATE AND KEY =0
E3
INDICATE TCP ACTIVE IN CSCB AND START TCPWTR
F3
RESET KEY AND STATE
G3
OPEN SNAPDCB
H3
EXIT TO TCP1WAIT
K3
RETURN TO OS WITH RETURNCODE =4
GETROUT
A4
INDICATE LENGTH AND SUBPOOL =1 IN R0
B4
GETMAIN FOR REQUIRED LENGTH
C4
CLEAR CORE
D4
RETURN VIA BR14

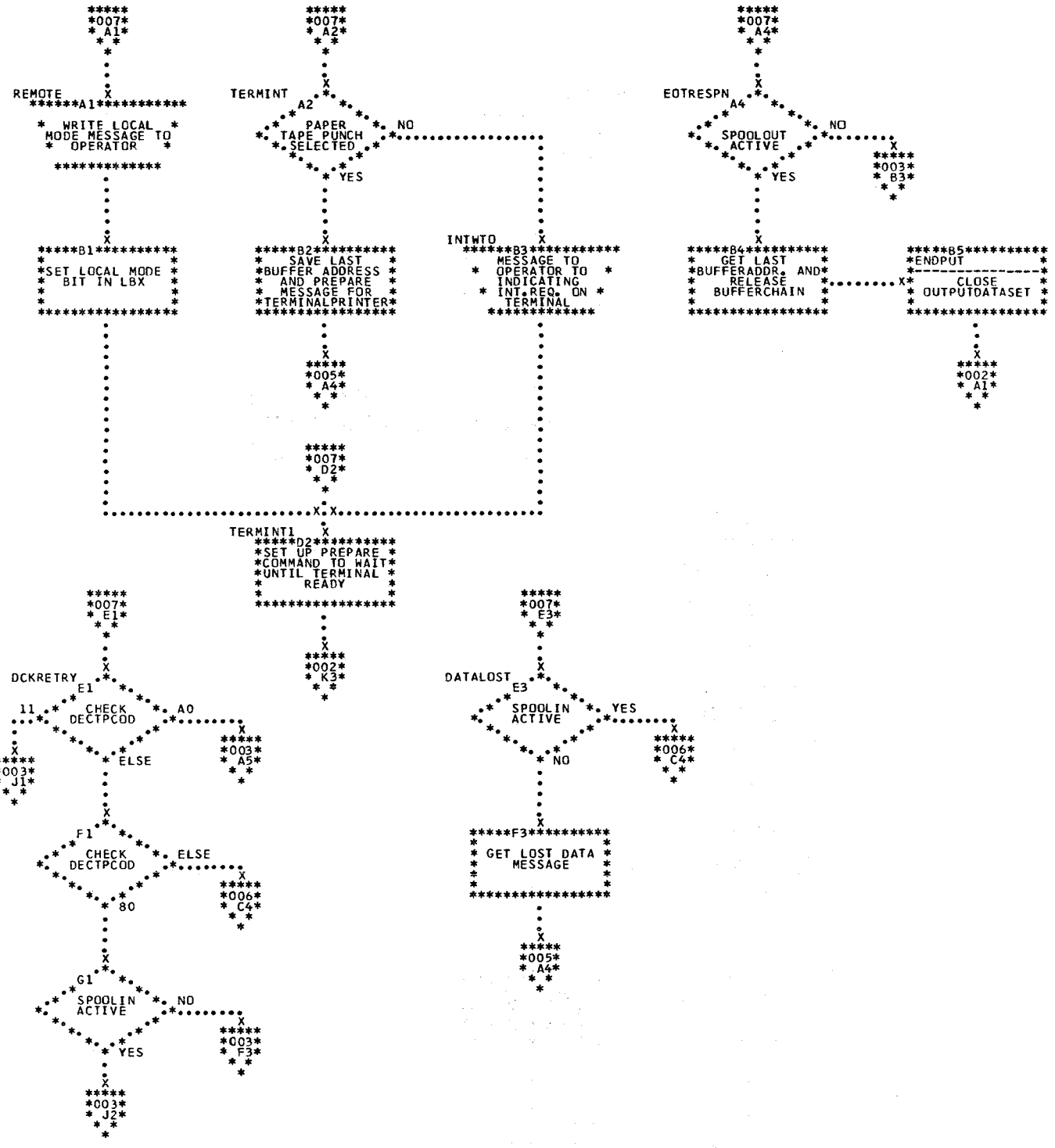

TCP1WAIT
A1 ENTRY VIA BRANCH
B1 SET KEY 0 AND SUP STATE
C1 TWAIT ON ELTSTOUT
D1 SET OLD KEY AND PROBLEM STATE
LISTSHI
E1 SHIFT ECBLIST BY 1 POS. IN ORDER TO AVOID PRIORITYS OF ECB'S
ANALYZE
F1 ANALYZE EVENT
OECB..002 A2
TECB..002 E2
SECB..002 H2
DECB..002 K5
PRAB..002 H1
002 H1
PRABEND
H1 SET COMPLETION CODE 111
J1 ABEND
K1 INDICATE ENTSCHL PENDING IN LINEBOX
CPPOST
A2 SET KEY=0 AND SUPERVISOR STATE
B2 CLEAR OPERTOR ECB (IN SUP. QUEUE SPACE)
C2 RESET KEY AND PROBLEM STATE
D2 TCP1OPCP HANDLE OPERATOR COMMUNICATION
002 E2
TASKCOMP
E2 DETACH TCPOPROC
F2 LINK TO TCP1ISDN FOR SHUTDOWN
G2 RETURN TO OS
G5
SCHLPOST
H2 POSTED BY SCHL
002 H2
SANALYZE
J2 CLEAR ECB AND GET LINEBOX
002 K2
ACTTEST
K2 MESSAGE OUTPUT IN PROGRESS
YES
NO
GETIOB
A3 GET IOB
B3 ENTSCHL WANTS OUTPUT ONLY
YES
NO
PUTOUTIN
C3 INDICATE OUTPUT + INPUT IN LINEBOX
QUEUEMSG
D3 OP. MESS. WAITING FOR OUTPUT
YES
NO
002 H3
DYNOUT
H3 EOD OR NO OUTPUT FOR SPOOLOUT
YES
NO
GOOUT
J3 PREPARE DECBFIELDS FOR OUTPUT, GET DYNAMIC DCBADDRESS
002 K3
IOCENTER
K3 INITIALIZE DECB, DCB AND IOB
OUTONLY
A4 INDICATE OUTPUT ONLY IN LINEBOX
B4 LIBRARY OUTPUT FOR TERMINAL
NO
YES
C4 OUTSPOOL PREPARE OUTPUT
H3
002 E4
MQU
E4 CLEAR MESSAGE PENDING FLAG IN LBX
F4 IS THERE ANY MESSAGE
NO
YES
LBFOUND
G4 PUT MESSAGE BUFFERS IN FRONT OF ENTSCHL MESSAGE
H4 ENDPUT CLOSE OUTPUT AND RELEASE WORN SPACE
A1
K4 IGG019MA EXCP
A5
A5
A5 TEST RETURNCODE
OK
PERR
LERR
H1
B5 LOPEN TRY TO REOPEN LINE
C5 LOPEN OK
NO
YES
006 C4
IOP
E5 ONE MORE TIME OPERATOR MESSAGE PROC
NO
YES
A1
F5 TCP1OPCP HANDLE OPERATOR COMMUNICATION
A1
G5
G5 MASK 'OF' NOT ZERO
YES
NO
K5
H5 GET LBXREG
J5 SPOOLTASK ACTIVE
YES
NO
005 F4
IOPOST
K5 GET LINEBOXADDRESS
003 A1

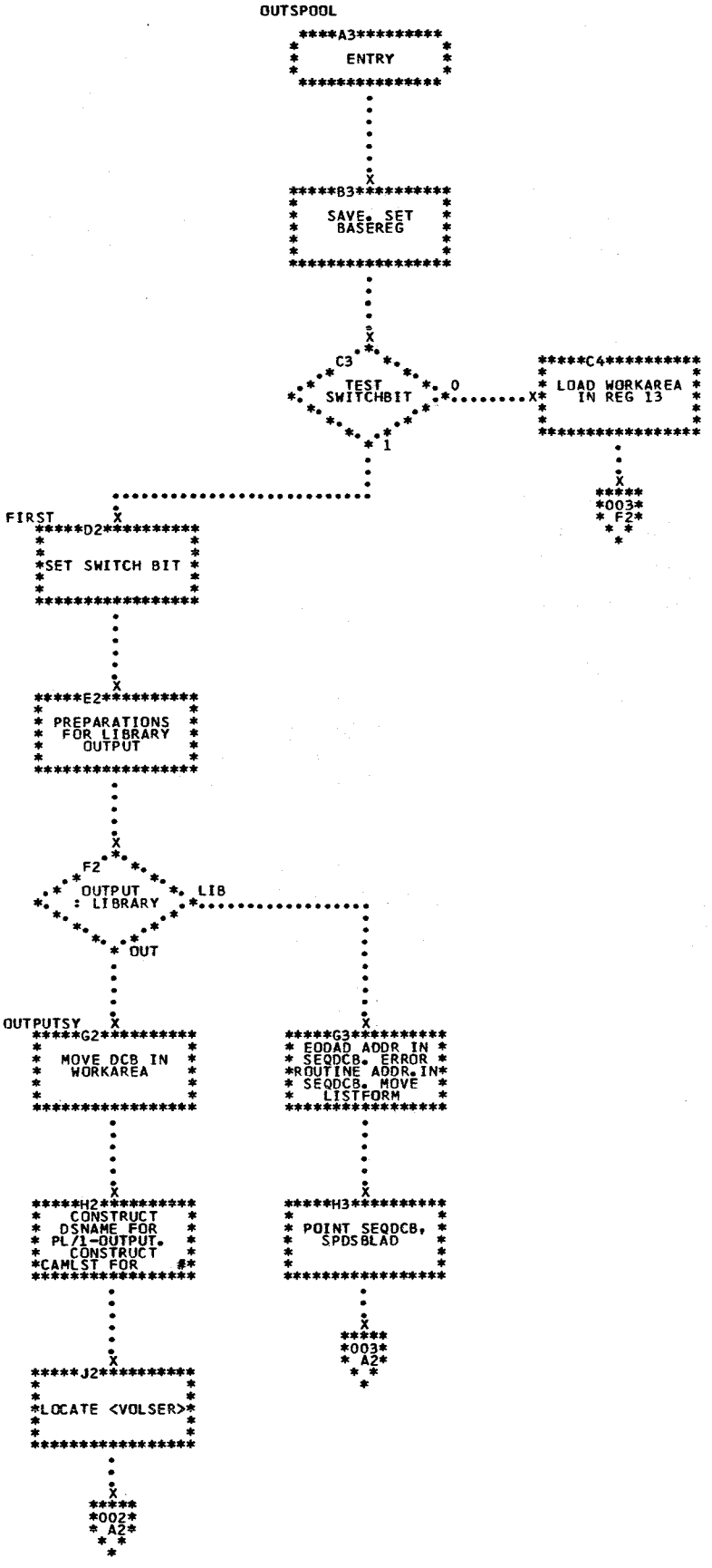

IOCOMP
A1
GET DCB AND IOBBASE
B1
LOGIN MODE INDICATE IN KBX
NO
YES
C1
SNAP LINEBOX AND BUFFER CONTROLBLOCK
NCSNAP
D1
GET IOB FOR LINE
E1
IO COMPLETION - CODE IN DECB
7F
OTHR
F1
IO COMPLETION - CODE IN DECB
41
48
G1
GET LAST BUFFER FROM DECAREA ANDD INSERT 'KREUZ'
H1
READ COMMAND ALREADY ACTIVE
SKIPNAK
J1
MAX. LENGTH X'FFFF' TO DEC LENGTH SET DECTYPE OB
ECB7F
A2
CLEAR ECB IN DECB
B2
TYPE OF CHANNEL PROGRAM COMPLETED
00....004 G3
01....003 F2
02....003 A3
03....003 E2
04....003 E2
05....003 F2
08....005 A1
0A....004 G4
0B....004 G4
ELSE..006 C4
FLAG08
E2
BUFFER OVERRUN
YES
NO
EOTTEST
F2
LAST CCW = READ EOT
G2
EOT RECEIVED
H2
DECTYPE =01
CK03
J2
INDICATE WRITE ACK IN DECB, GET ADDRESS OF DYNAMIC DCB
READERR
K2
INPUT FOR SPOOLIN
NAKTEST
A3
TEST RESPONSE TO TEXT
ACK
ELSE
NAKRESP
B3
LAST I/O OPERATION WAS RETRY
C3
INDICATE RETRY AND SET RETRYCOUNT =10 IN DECB
WRNAK
E3
INDICATE WRITE NAK (DECTYPE 0A IN DECB)
OK01
F3
GET INPUTBUFFER ADDRESS AND TRANSLATE INPUTMESSAGE
G3
RESET SCHLCTIN BITS IN ACTION BYTE IN LBX
H3
POST INECB IN LBX TO ACTIVATE ENTSCHL
J3
ANY OPERATOR MESSAGE PENDING
K3
RESET MESSAGE PENDING FLAG AND INDICATE MSOUT IN LBX
A4
DECREMENT RETRY COUNT IN DECB
B4
RETRY COUNT EQUAL 0
C4
TFREEBUF
D4
SPOOLIN OR SPOOLOUT ACTIVITY
E5
XERRMS2
E4
RESET ERROR MESSAGE INDICATOR IN LBX
F4
RETRY NOW GOING ON
G4
RETRY COUNT EQUAL 0
XSKIP
H4
ERROR END FOR SPOOLIN
J4
RESET O/I SCHLOTIN INDICATOR POST INECB FOR ENTSCHL
OK02
A5
I/O WAS ERROR MESSAGE OUTPUT
B5
I/O WAS A RETRY
C5
RESET RETRY INDICATOR AND RETRYCOUNT
FREEBUF
D5
FREE LAST BUFFER(S) USED FOR I/O
XERRMS
E5
I/O WAS ERROR MESSAGE OUTPUT
XTFLG
F5
DYNAMIC READ RETRY
G5
SPOOLOUT ACTIVE
H5
RESTORE ADDRESS CHARACTER OFFSET AND FIRST INPUT BUFFER ADDRESS


004 A1
XSCHLOTI
A1 NORMAL INPUT FOLLOWING OUTPUT
NO
YES
XMSOUT
A2 LAST I/O WAS OPMESSAGE OUTPUT
NO
YES
XSCHLO
A3 LAST I/O WAS OUTPUT FOR ENTSCHL
NO
YES
002 H1
B1 DATA INPUT (SPOOLIN)
YES
NO
B2 RESET MESSAGE OUTPUT FLAG
B3 SPOOLOUT ACTIVE
NO
YES
B4 RESET OUTPUT INDICATOR (SCHLO) IN LINEBOX
B5 POST INECB IN LINEBOX TO ACTIVATE ENTSCHL
002 A1
READNORM
C1 READ RETRY
YES
NO
C2 ENTSCHL PENDING (SCHLPD)
NO
YES
002 A1
C3 END OF DATA FROM OUTSPOOL
YES
NO
C4 ENDPUT CLOSE OUTPUT DATA SET
002 A1
002 K2
004 E2
NORETRY
D1 GET ADDRESS OF DYN.DCB AND REQUEST ONE BUFFER
D3 SPOOLOUT PREPARE NEXT TEN RECORDS
002 H3
E1 BUFFER AVAILABLE
NO
YES
002 H1
READDYN
E2 GET DYNAMIC DCB ADDR., INDICATE DYNAMIC INPUT DECTYP =03
E3 SAVE TCBADDR, AND INDICATE SPOOLTASK ACTIVE IN LINEBOX
E3
002 K3
OKBUFSIM
F1 INITIALIZE DECB WITH BUFFERADR. AND LENGTH
F1
004 G3
004 G4
RDNORM1
G1 SIMPLE DCB ADDR. AND OFFSET FOR ADDRESSING CHAR TO DECB
CKBUFDYN
G2 BUFFER AVAILABLE
NO
YES
002 H1
OKOO
G3 RESET LOCAL MODE BIT
OKOA
G4 SPOOLIN ACTIVE
YES
NO
005 A1
H1 READ FROM PAPER TAPE READER
NO
YES
H2 INITIALIZE DECB WITH BUFFER- INFORMATION AND ADDRESSING -ENTRY
H3 NORMAL SEQUENCE SCHLOTIN TO BE START
YES
NO
002 H3
H4 OPERATOR MESSAGE PENDING
NO
YES
005 G1
002 A1
002 E4
J1 INDICATE NORMAL READ WITHOUT PREPARE IN DECTYPE
J2 INITIALIZE PARAMETERS FOR TCP1SPOL
002 K3
TAST
K1 INDICATE NORMAL READ WITH PREPARE IN DECTYPE
**K2* ATTACH TCP1SPOL
RDNCRM2
K3 GET OLD BUFFER ADDRESS FROM DECAREA
F1
E3

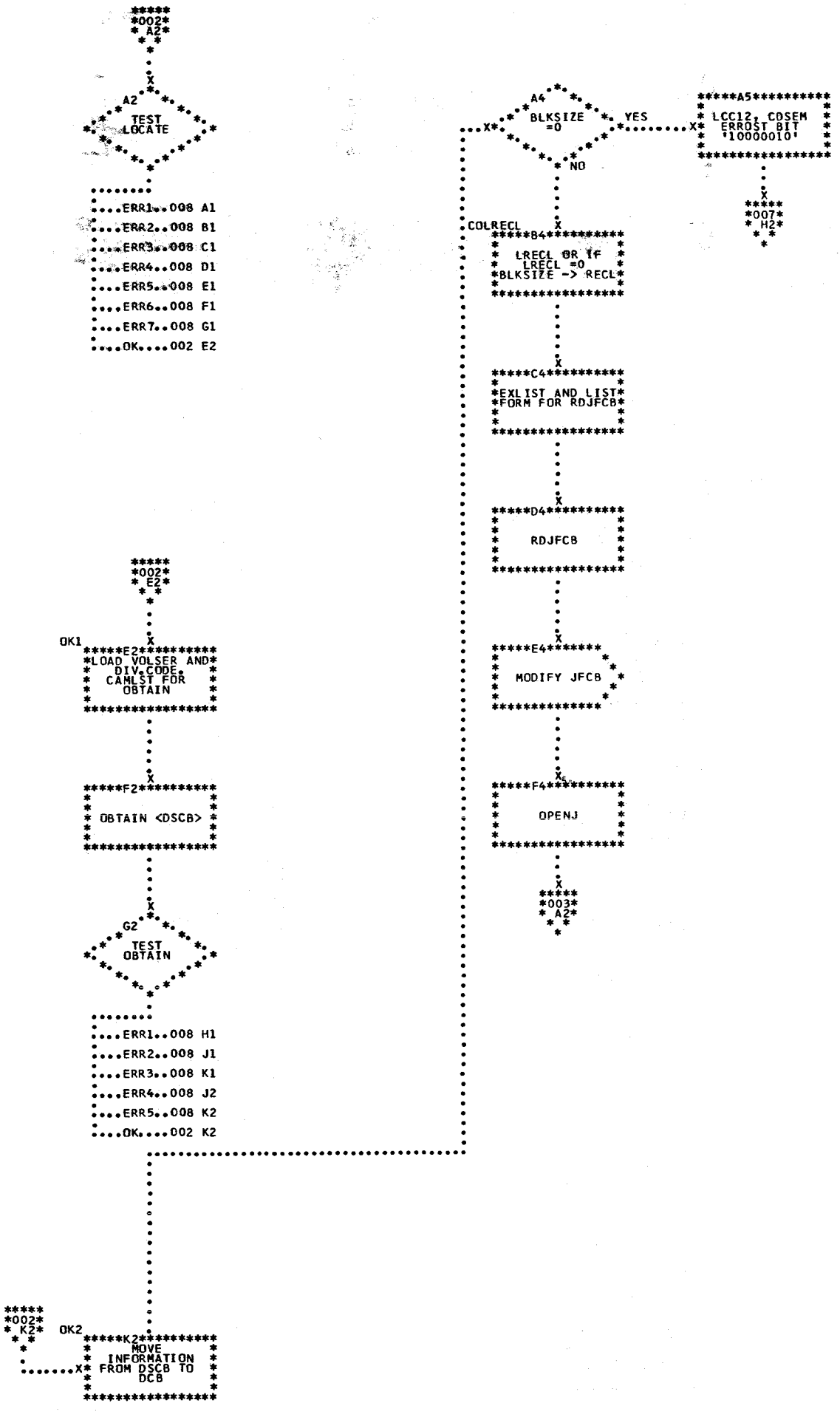

005 A1
OK08
A1 IS SPOOLTASK TERMINATED
NO
YES
005 B1
SPLTEND1
B1 TEST SPCCLTASK RETURNCODE
¬=0
ZERO
NOTOKOUT
A2 PREPARE MESSAGE WITH RETURNCODE FOR TERMINALOUTPUT
C2
RDCDYN
C2 SET SPOOLTASK ACTIVE FLAG IN LBX, INDICATE DECTYPE 04 TO RESTART INPUT
002 K3
C1 I/O END BY TIMEOUT
YES
NO
D1 POST INECB TO INFORM ENTSCHL ABOUT INPUT END
E1 CLEAR ERROR MESSAGE BUFFER ADDRESS IN LBX
RSCHLOTI
F1 INDICATE END OF INPUT OPERATION
002 A1
005 G1
RDRETRY
G1 IS IT THE FIRST RETRY
NO
YES
XRDRETRY
G2 DECREMENT RETRYCOUNT
H1 INDICATE RETRY AND SET RETRYCOUNT
H2 RETRYCOUNT EQUAL ZERC
YES
NO
J1 SPOOLTASK STILL ACTIVE
NO
YES
READCONT
J2 SPOOLIN
YES
NO
K1 RESET I/O ACTION FLAGS SCHLOTIN IN LBX
K2 INDICATE SPOOLTASK ACTIVE IN LBX
002 A1
C2
POSTIN
A3 SPOOLIN ACTIVE
NO
YES
B3 INDICATE POST INECB AFTER ERRORMESSAGE OUTPUT
C3 USER RETURNCODE
YES
NO
F3
D3 ANY ERROR-RECORD BUFFERS
NO
YES
A4
E3 RELEASE ERROR-RECORD BUFFERCHAIN , CLEAR BUFFERADDRESS
A4
TESTEREC
F3 EXIST ERRORRECORDS
NO
YES
G3 INDICATE ERROR-RECORD CHAINING
MESS04
H3 GET ADDRESS OF PERMANENT ERROR MESSAGE
XMESS
J3 GET ADDRESS OF TEMPORARY ERRORMESSAGE
F3
K3
TOMESS1
K3 GET ADDRESS OF TIMEOUT ERRORMESSAGE
005 J2
TOMESS
005 A4
A4 GET DYNAMIC DCB ADDRESS
A4
MGETBUF
B4 GET ONE BUFFER FROM POOL
C4 BUFFER AVAILABLE
YES
NO
002 H1
D4 CHAIN ERRORREC BUFFERS TO ERRORMESS BUFFER, UPDATE LBX
A5 FAILING DYNAMIC READ
YES
NO
B5 GET OLD BUFFER ADDRESS
BUFHERE
C5 BUFFER ADDRESS TO LINEBOX
D5 ANY ERRORBUFFERS TO CHAIN
YES
NO
TOMESS2
E5 MOVE DESIGNATED ERROR-MESSAGE TO OUTPUTBUFFER AND TRANSLATE
002 H3
005 F4
SPLTEND
F4 CLEAR POSTBIT, GET TCB ADDRESS AND COMPLETION-CODE
G4 TEST IF TIMEOUT
YES
NO
RSPLTASK
H4 DETACH SPLTASK, FREE PARMFIELD RESET TIMEOUTBIT IF ANY
J4 SAVE COMPL.CODE IN LBX RESET BIT SPLTASK IN ACTIONBYTE
K4 COMPL.CODE =0
NO
YES
006 A1
NODETACH
F5 COMPL.CODE =0
NO
YES
G5 INPUT STILL ACTIVE
YES
NO
002 A1
NOIO
H5 RESET OUTPUT-INPUT FLAG SCHLOTIN IN LBX
C2
J5 RESET PCIBITS TO STOP CHANNEL PROG.
006 A1

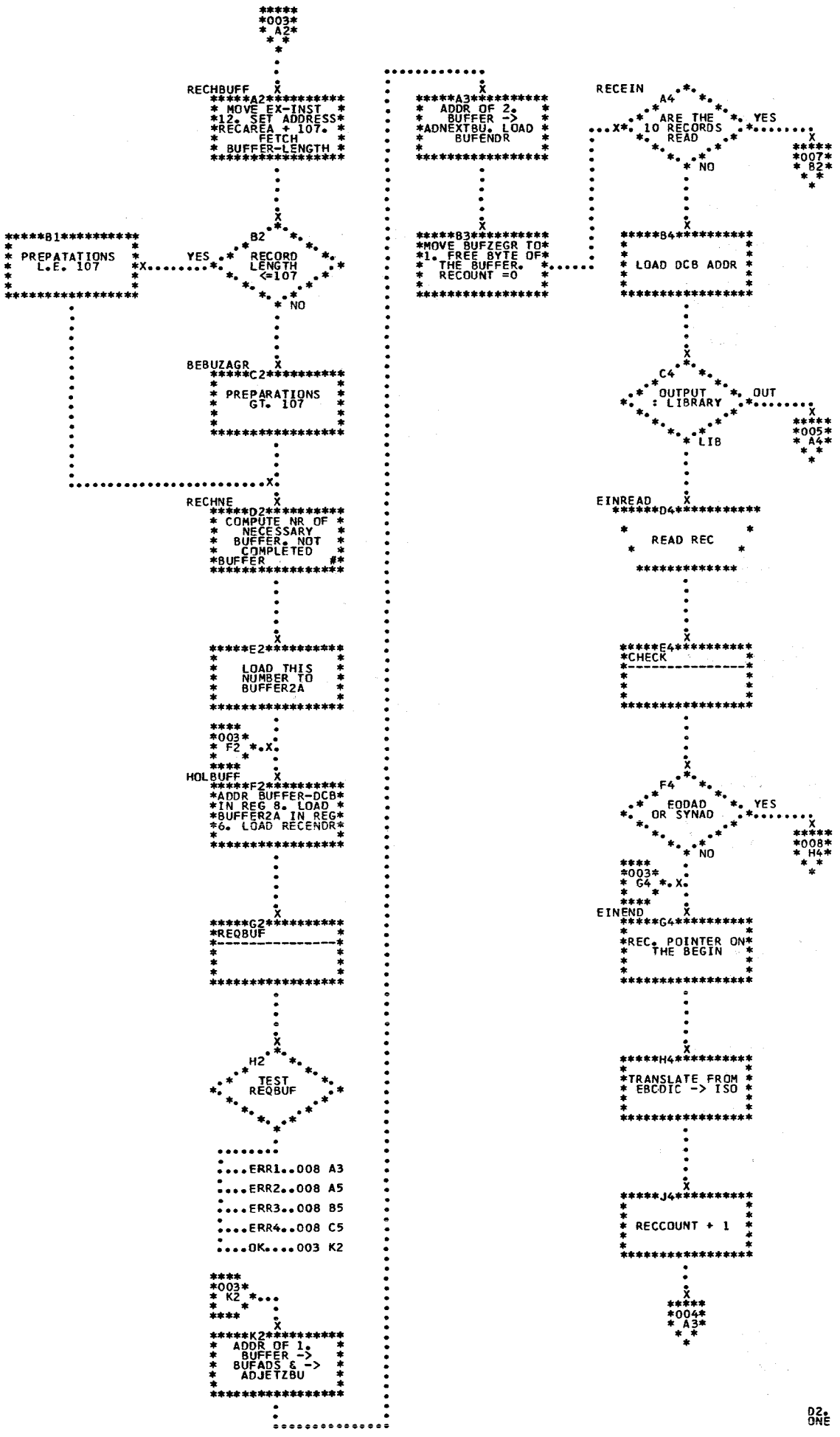

NCPSTOP
A1
INPUT STILL ACTIVE
YES
NO
B1
RETURN FROM WRITE NAK
NO
YES
SETSCHLN
C1
SET INPUT ACTIVE FLAG IN LBX
ECB41
A3
CLEAR ECB
B3
DECFLAGS NOT ZERO
YES
NO
ERRST
C3
ANY SHOULD NOT OCCUR BITS IN DECERRST
YES
NO
CDC
D3
CHANNEL DATA CHECK OR SHOULD NOT OCCUR STAT
YES
NO
B4
BUFFER OVERRUN
YES
NO
LDOWN
C4
INDICATE LINEDOWN IN LBX
D4
INTERVENTION REQUIRED
NO
YES
E4
WRITE TEMPORARY LINE DOWN MESSAGE TO OPERATOR
D5
RESPTOUT
A5
CHECK DECTPCOD
A0
20
ELSE
B5
CHECK DECTPCOD
22
ELSE
C4
LPERMERR
C5
WRITE PERMANENT ERROR MESSAGE TO OPERATOR
D5
SPOOLOUT ACTIVE
YES
NO
E5
RELEASE BUFFER IF ANY
LPOST
F5
POST INECB TO INFORM ENTSCHL
YENDPUT
G5
LAST OUTPUT FOR SPOOLOUT
NO
YES
H5
RESET END OF DATA INDICATOR IN LINEBOX
NORESET
J5
RELEASE BUFFERCHAIN
K5
ENDPUT
CLOSE OUTPUT DATASET
UC
E1
UNIT CHECK
NO
YES
UE
E2
UNIT EXCEPTION
NO
YES
SENSE
F1
EQUIPMENT CHECK
YES
NO
G1
LOST DATA
YES
NO
G3
CHECK DECTPCOD
11....007 E3
A0....003 B3
20....003 B3
ELSE..006 C4
TIMEOUT
H1
TIMEOUT
YES
NO
H2
READ TEXT
YES
NO
J2
READ EOT
NO
TO8
TO1
TINTREQ
G4
INTERV REQUIRED ON TERMINAL
YES
NO
J4
SPLOUT ACTIVE OR RESP. TO ADDR. OK
YES
NO
INTREQ
K1
DCK
CHECK RESIDUAL SENSEBYTES
ELSE
OVER
G4
G5
D5
A5
C4


007 * A1*
REMOTE
A1
* WRITE LOCAL MODE MESSAGE TO OPERATOR
007 * A2*
TERMINT
A2
PAPER TAPE PUNCH SELECTED
NO
YES
007 * A4*
EOTRESPN
A4
SPOOLOUT ACTIVE
NO
YES
003 * B3*
INTWTO
B1
*SET LOCAL MODE BIT IN LBX
B2
SAVE LAST *BUFFER ADDRESS AND PREPARE MESSAGE FOR *TERMINALPRINTER
B3
MESSAGE TO OPERATOR TO INDICATING INT.REQ. ON TERMINAL
B4
GET LAST *BUFFERADDR. AND RELEASE BUFFERCHAIN
B5
*ENDPUT
CLOSE OUTPUTDATASET
005 * A4*
002 * A1*
007 * D2*
TERMINT1
D2
*SET UP PREPARE COMMAND TO WAIT *UNTIL TERMINAL READY
002 * K3*
007 * E1*
DCKRETRY
E1
11
CHECK DECTPCOD
A0
ELSE
003 * A5*
003 * J1*
F1
CHECK DECTPCOD
ELSE
80
006 * C4*
G1
SPOOLIN ACTIVE
NO
YES
003 * F3*
003 * J2*
007 * E3*
DATALOST
E3
SPOOLIN ACTIVE
YES
NO
006 * C4*
F3
GET LOST DATA MESSAGE
005 * A4*

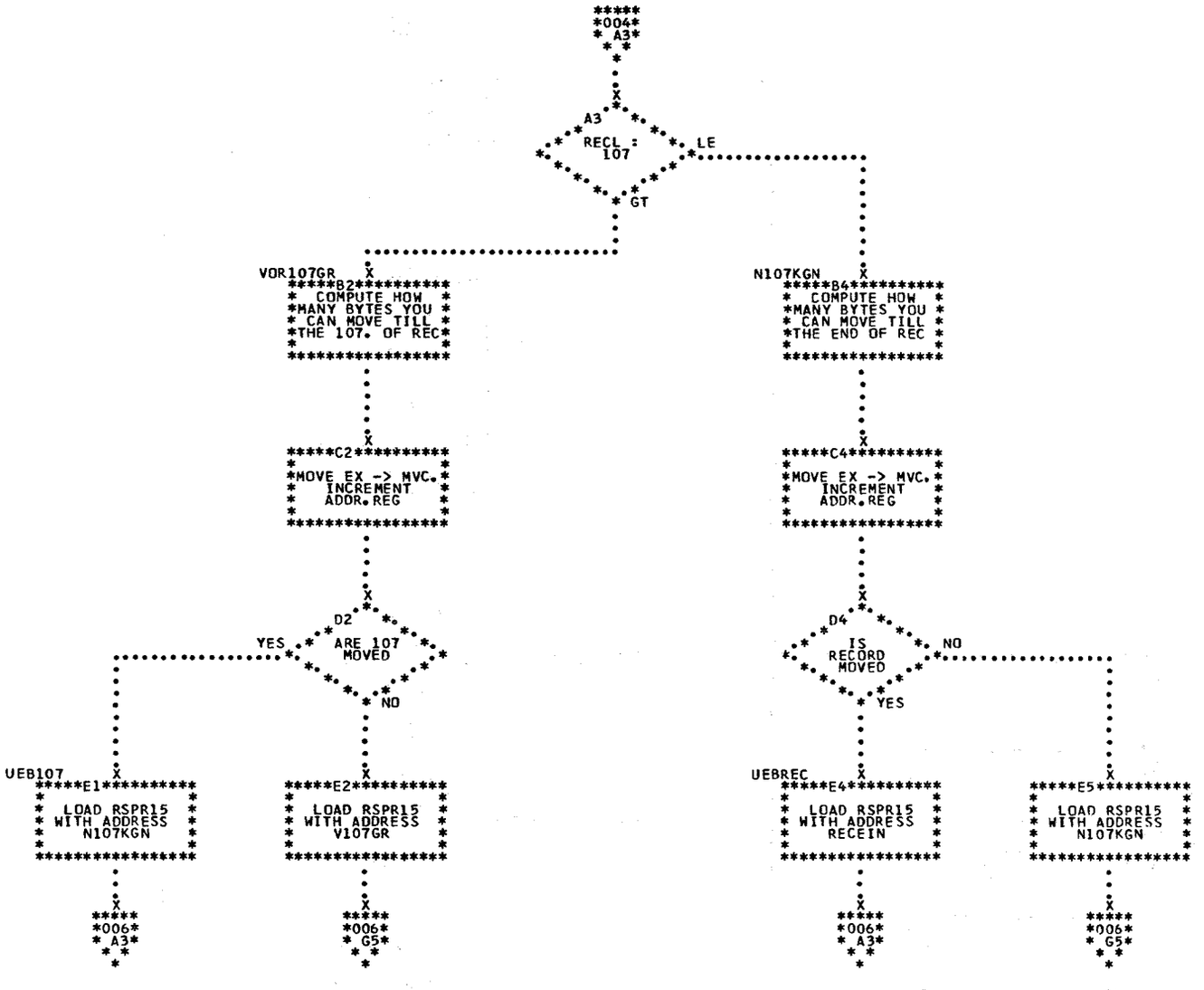

OUTSPOOL
A3
ENTRY
B3
SAVE, SET BASEREG
C3
TEST SWITCHBIT
0
1
C4
LOAD WORKAREA IN REG 13
003
F2
FIRST
D2
SET SWITCH BIT
E2
PREPARATIONS FOR LIBRARY OUTPUT
F2
OUTPUT : LIBRARY
LIB
OUT
OUTPUTSY
G2
MOVE DCB IN WORKAREA
G3
EODAD ADDR IN SEQDCB, ERROR ROUTINE ADDR, IN SEQDCB, MOVE LISTFORM
H2
CONSTRUCT DSNAME FOR PL/1-OUTPUT, CONSTRUCT CAMLST FOR #
H3
POINT SEQDCB, SPDSBLAD
003
A2
J2
LOCATE <VOLSER>
002
A2

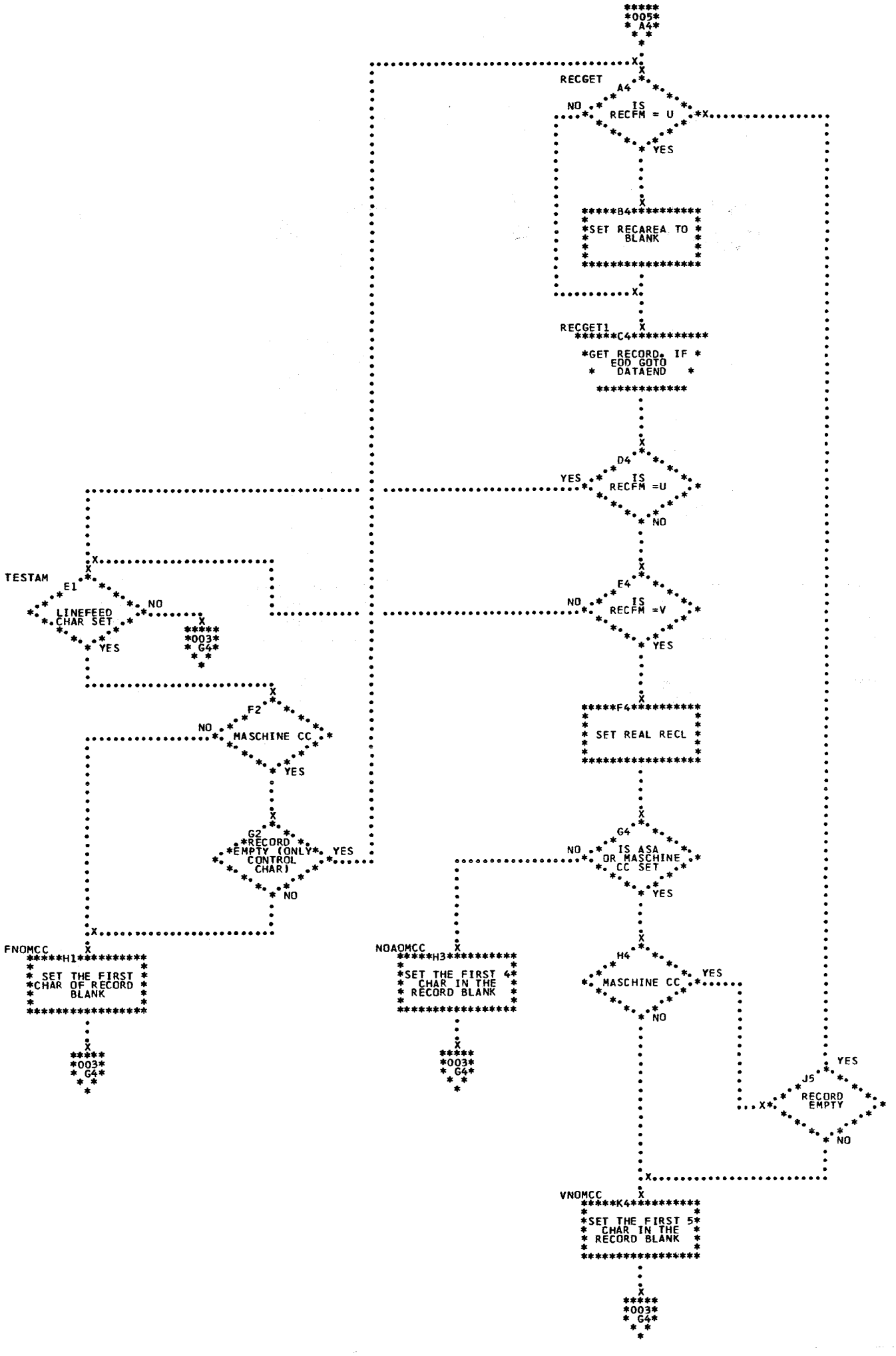
H2. LOCATE

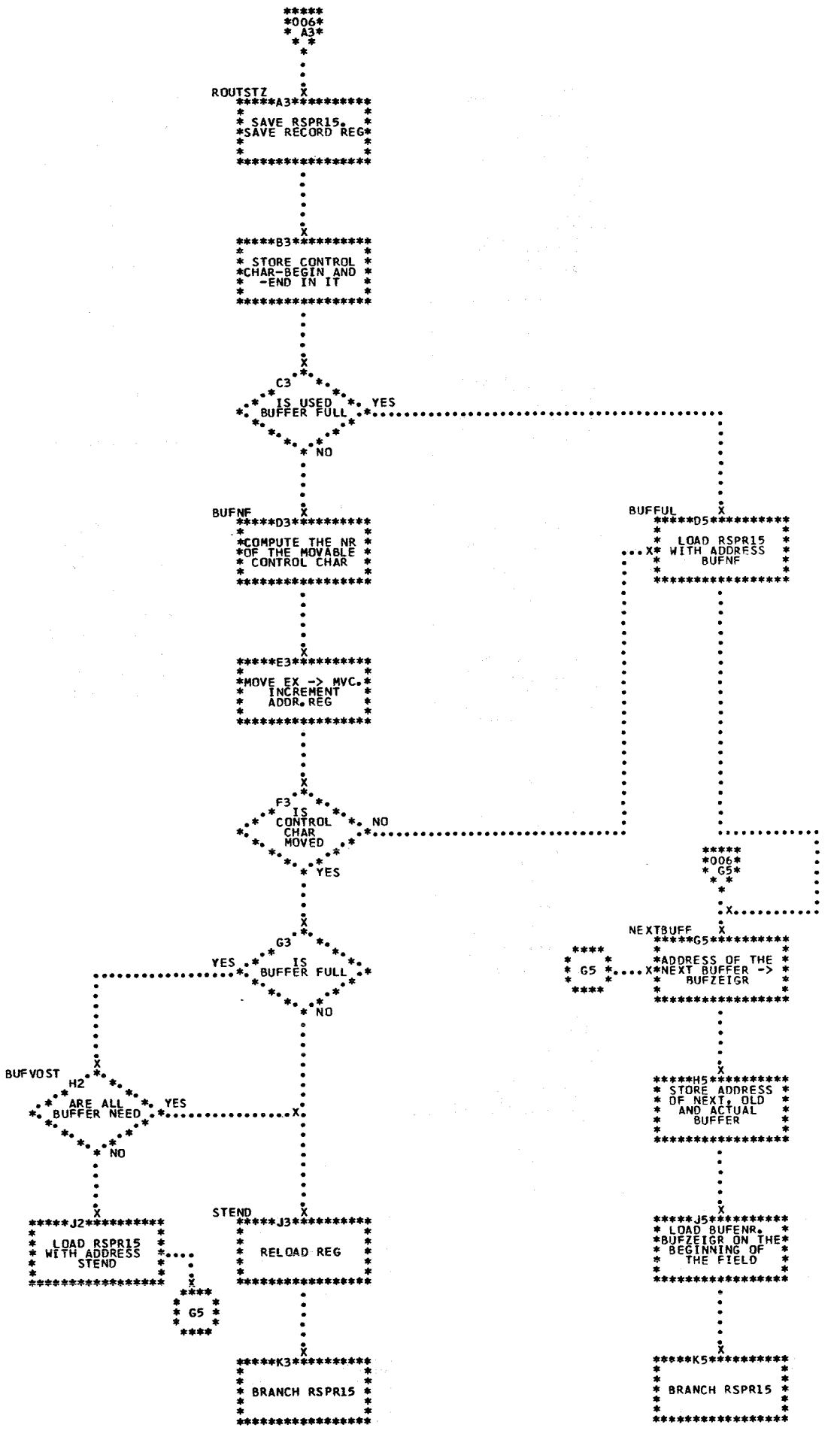

002
A2
A2
TEST LOCATE
ERR1..008 A1
ERR2..008 B1
ERR3..008 C1
ERR4..008 D1
ERR5..008 E1
ERR6..008 F1
ERR7..008 G1
OK....002 E2
A4
BLKSIZE =0
YES
NO
*****A5*****
LCC12, CDSEM
ERROST BIT
'10000010'
007
H2
COLRECL
*****B4*****
LRECL OR IF LRECL =0
BLKSIZE -> RECL
*****C4*****
EXLIST AND LIST
FORM FOR RDJFCB
*****D4*****
RDJFCB
*****E4*****
MODIFY JFCB
*****F4*****
OPENJ
003
A2
002
E2
OK1
*****E2*****
LOAD VOLSER AND
DIV.CODE.
CAMLST FOR
OBTAIN
*****F2*****
OBTAIN <DSCB>
G2
TEST OBTAIN
ERR1..008 H1
ERR2..008 J1
ERR3..008 K1
ERR4..008 J2
ERR5..008 K2
OK....002 K2
002
K2
OK2
*****K2*****
MOVE
INFORMATION
FROM DSCB TO
DCB

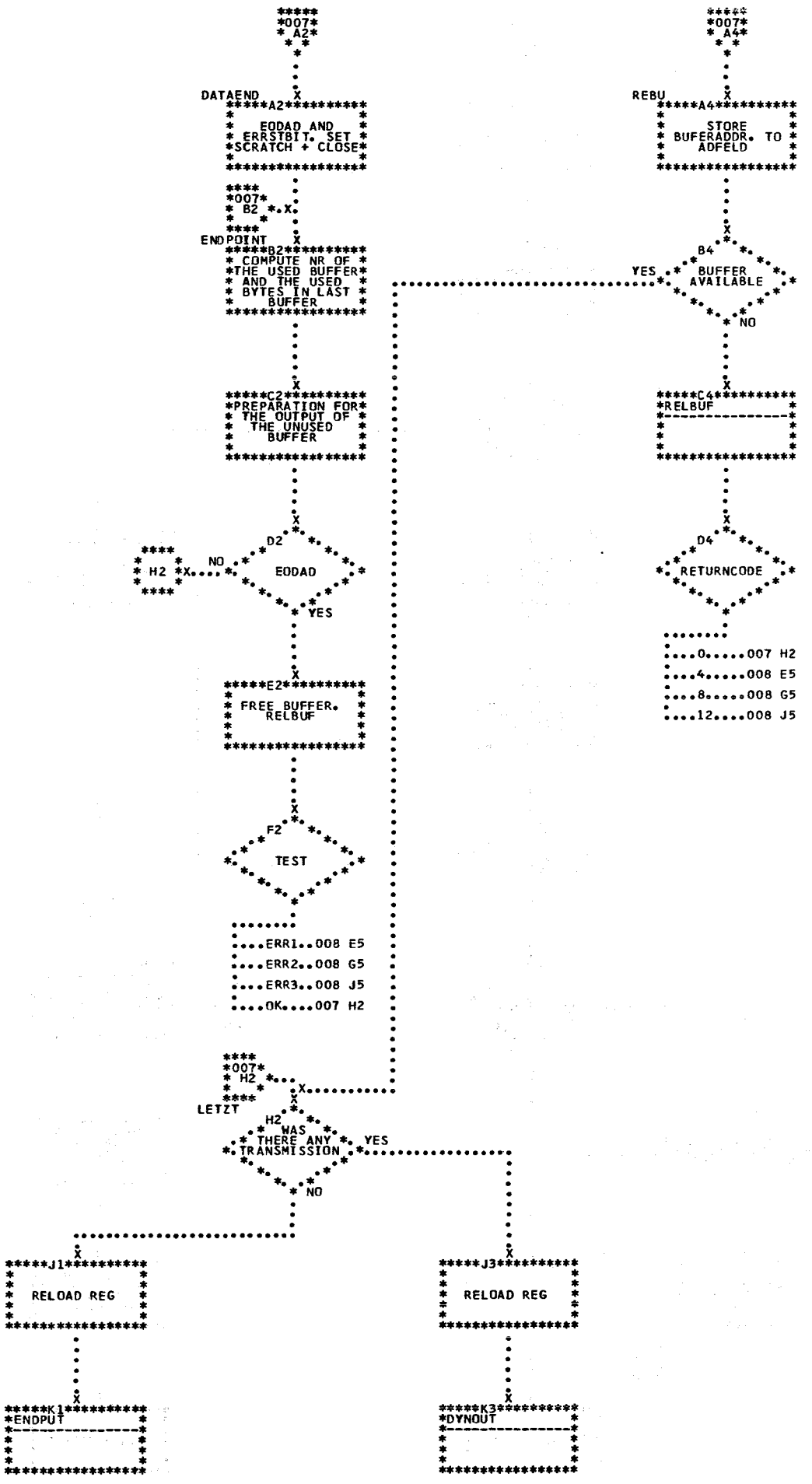


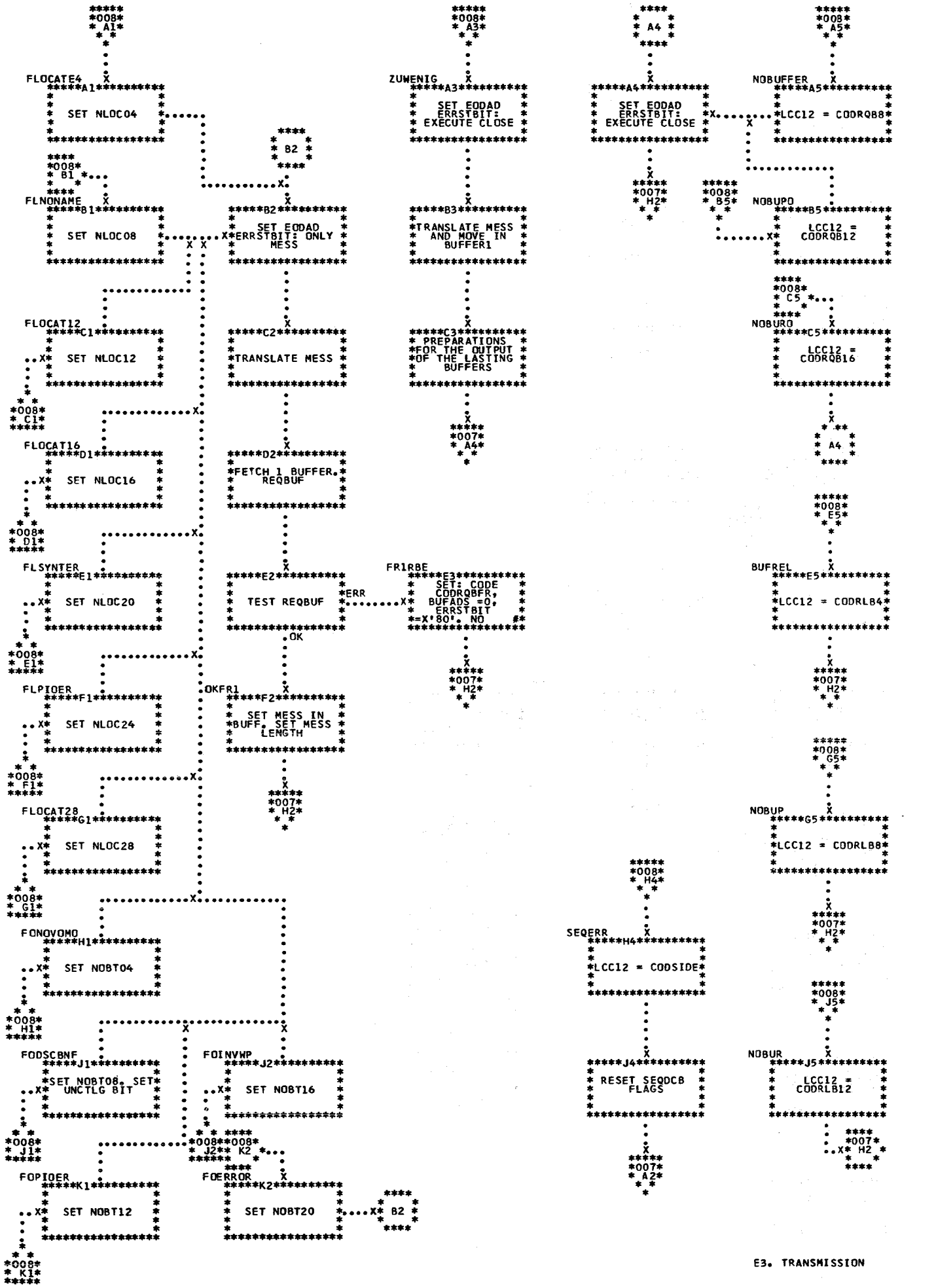
D2. IMPLICATES ANOTHER ONE


004
A3
A3
RECL : 107
LE
GT
VOR107GR
B2
COMPUTE HOW MANY BYTES YOU CAN MOVE TILL THE 107. OF REC
C2
MOVE EX -> MVC. INCREMENT ADDR. REG
D2
ARE 107 MOVED
YES
NO
N107KGN
B4
COMPUTE HOW MANY BYTES YOU CAN MOVE TILL THE END OF REC
C4
MOVE EX -> MVC. INCREMENT ADDR. REG
D4
IS RECORD MOVED
NO
YES
VEB107
E1
LOAD RSPR15 WITH ADDRESS N107KGN
E2
LOAD RSPR15 WITH ADDRESS V107GR
VEBREC
E4
LOAD RSPR15 WITH ADDRESS RECEIN
E5
LOAD RSPR15 WITH ADDRESS N107KGN
006
A3
G5


005
* A4*
RECGET
A4
IS RECFM = U
NO
YES
B4
SET RECAREA TO BLANK
RECGET1
C4
GET RECORD, IF EOD GOTO DATAEND
D4
IS RECFM =U
YES
NO
TESTAM
E1
LINEFEED CHAR SET
NO
YES
003 G4*
E4
IS RECFM =V
NO
YES
F2
MASCHINE CC
NO
YES
F4
SET REAL RECL
G2
RECORD EMPTY (ONLY CONTROL CHAR)
YES
NO
G4
IS ASA OR MASCHINE CC SET
NO
YES
FNOMCC
H1
SET THE FIRST CHAR OF RECORD BLANK
NOAOMCC
H3
SET THE FIRST 4 CHAR IN THE RECORD BLANK
H4
MASCHINE CC
YES
NO
J5
RECORD EMPTY
YES
NO
VNOMCC
K4
SET THE FIRST 5 CHAR IN THE RECORD BLANK


006
* A3*
ROUTSTZ
SAVE RECORD REG
*STORE CONTROL *CHAR-BEGIN AND -END IN IT
C3
IS USED BUFFER FULL
YES
NO
BUFNF
*COMPUTE THE NR *OF THE MOVABLE * CONTROL CHAR
*MOVE EX -> MVC. INCREMENT ADDR. REG
F3
IS CONTROL CHAR MOVED
NO
YES
G3
IS BUFFER FULL
YES
NO
BUFVOST
H2
ARE ALL BUFFER NEED
YES
NO
LOAD RSPR15 WITH ADDRESS STEND
G5
STEND
RELOAD REG
BRANCH RSPR15
BUFFUL
LOAD RSPR15 WITH ADDRESS BUFNF
006
* G5*
NEXTBUFF
*ADDRESS OF THE *NEXT BUFFER -> BUFZEIGR
G5
STORE ADDRESS OF NEXT, OLD AND ACTUAL BUFFER
LOAD BUFENR. *BUFZEIGR ON THE* BEGINNING OF THE FIELD
BRANCH RSPR15

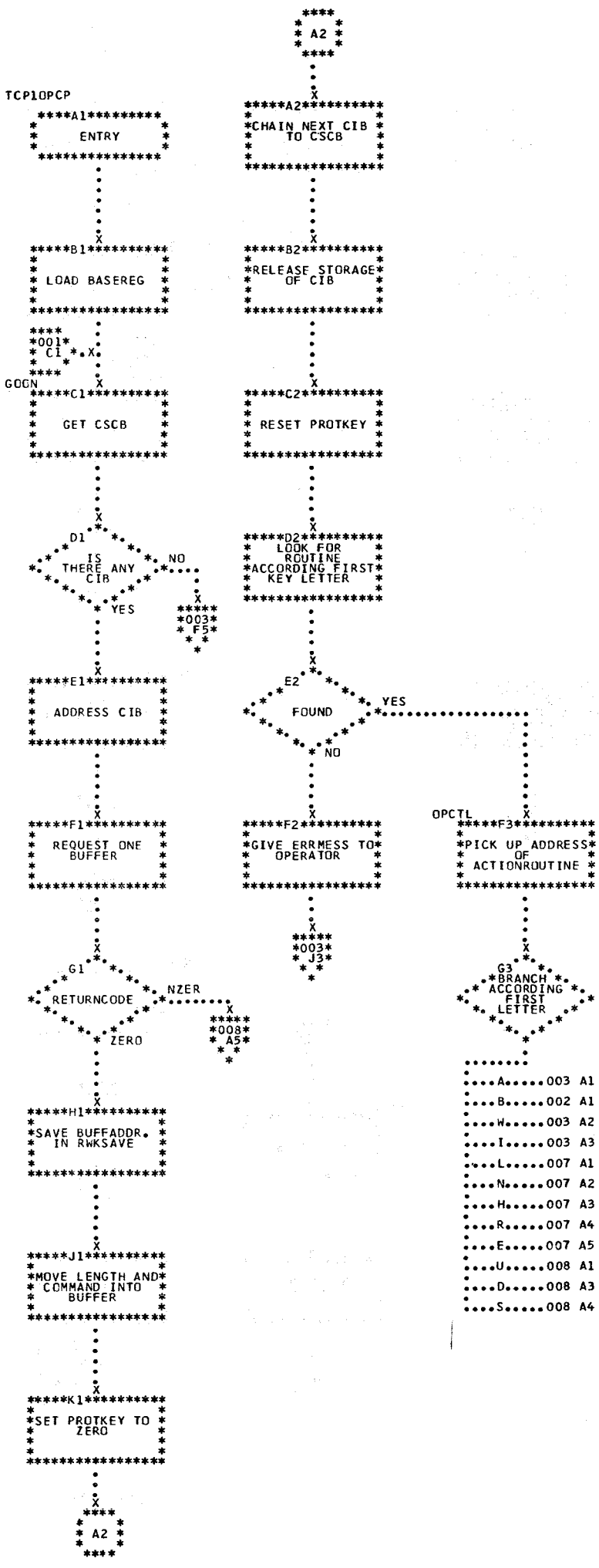

007
A2
DATAEND
A2
EODAD AND ERRSTBIT SET SCRATCH + CLOSE
007
B2
ENDPOINT
B2
COMPUTE NR OF THE USED BUFFER AND THE USED BYTES IN LAST BUFFER
C2
PREPARATION FOR THE OUTPUT OF THE UNUSED BUFFER
D2
EODAD
NO
YES
E2
FREE BUFFER. RELBUF
F2
TEST
ERR1..008 E5
ERR2..008 G5
ERR3..008 J5
OK....007 H2
007
H2
LETZT
H2
WAS THERE ANY TRANSMISSION
YES
NO
J1
RELOAD REG
K1
ENDPUT
J3
RELOAD REG
K3
DYNOUT
007
A4
REBU
A4
STORE BUFERADDR. TO ADFELD
B4
BUFFER AVAILABLE
YES
NO
C4
RELBUF
D4
RETURNCODE
0.....007 H2
4.....008 E5
8.....008 G5
12....008 J5

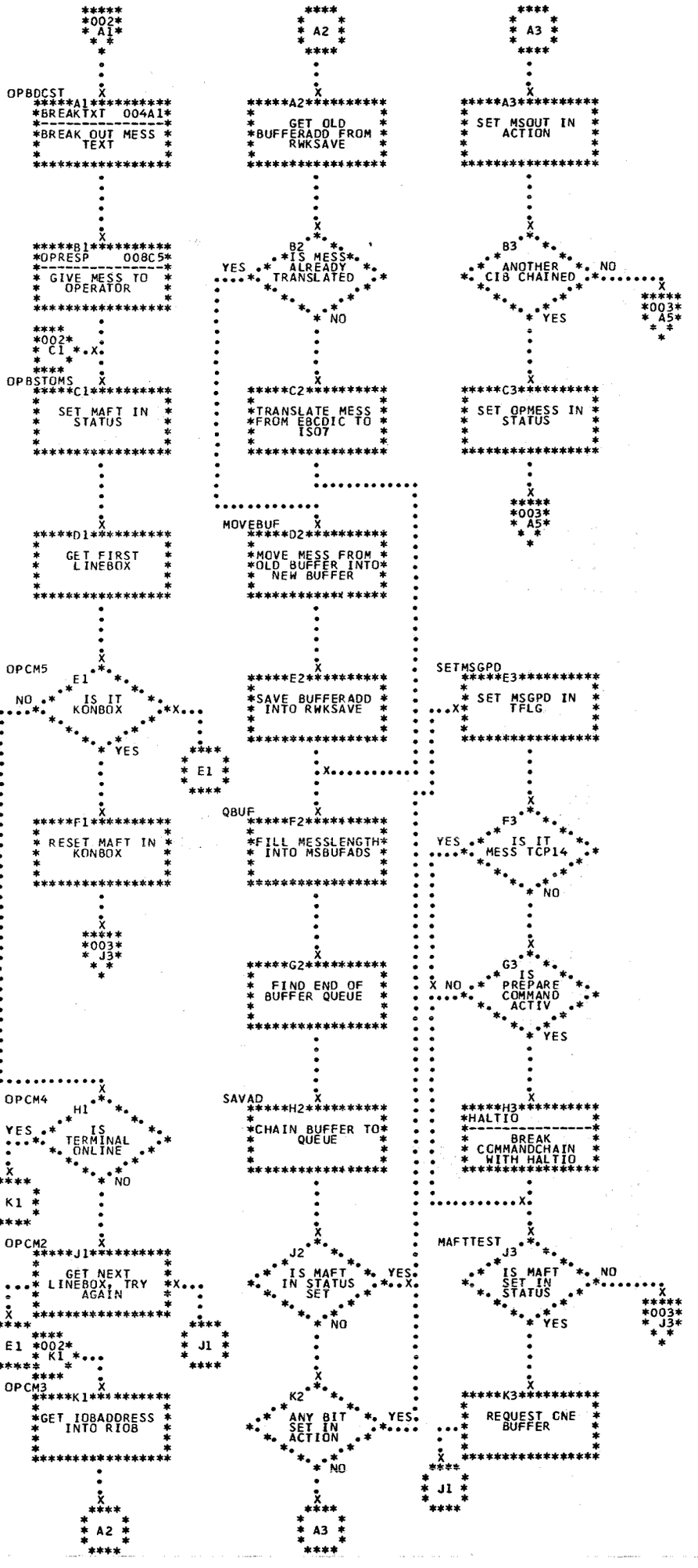

FLOCATE4
SET NLOC04
FLNONAME
SET NLOC08
FLOCAT12
SET NLOC12
FLOCAT16
SET NLOC16
FLSYNTER
SET NLOC20
FLPIOER
SET NLOC24
FLOCAT28
SET NLOC28
FONOVOMO
SET NOBT04
FODSCBNF
SET NOBT08, SET UNCTLG BIT
FOPIOER
SET NOBT12
B2
SET EODAD ERRSTBIT: ONLY MESS
TRANSLATE MESS
FETCH 1 BUFFER, REQBUF
TEST REQBUF
ERR
OK
OKFR1
SET MESS IN BUFF, SET MESS LENGTH
FOINVWP
SET NOBT16
FOERROR
SET NOBT20
B2
ZUWENIG
SET EODAD ERRSTBIT: EXECUTE CLOSE
TRANSLATE MESS AND MOVE IN BUFFER1
PREPARATIONS FOR THE OUTPUT OF THE LASTING BUFFERS
FR1RBE
SET: CODE CODRQBFR, BUFADS =0, ERRSTBIT =X'80', NO
A4
SET EODAD ERRSTBIT: EXECUTE CLOSE
H2
B5
NOBUFFER
LCC12 = CODRQB8
NOBUPO
LCC12 = CODRQB12
NOBURO
LCC12 = CODRQB16
BUFREL
LCC12 = CODRLB4
NOBUP
LCC12 = CODRLB8
SEQERR
LCC12 = CODSIDE
RESET SEQDCB FLAGS
NOBUR
LCC12 = CODRLB12
E3. TRANSMISSION

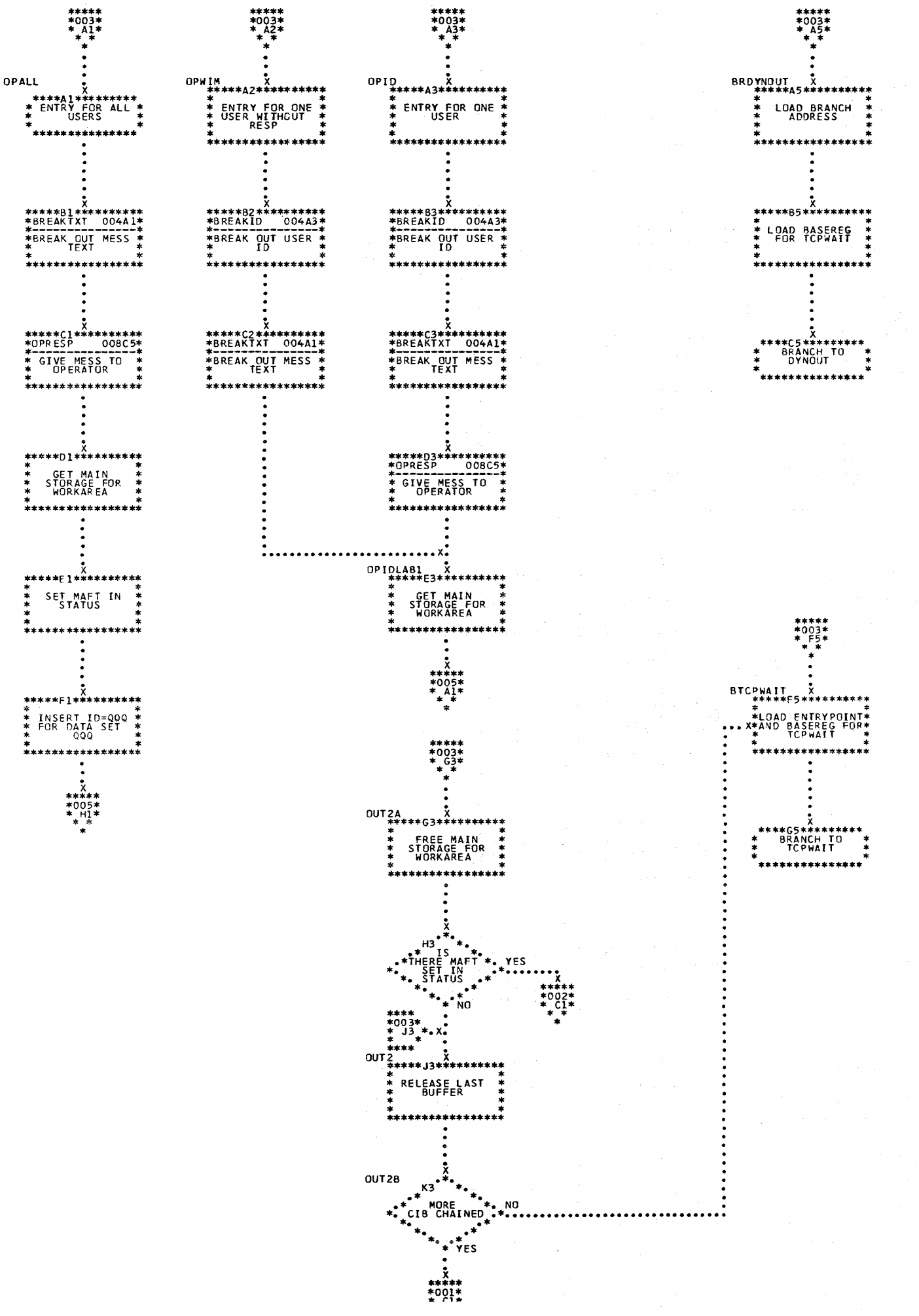
CSECT T C P 1 0 P C P IN MODULE T C P 1 B A S E

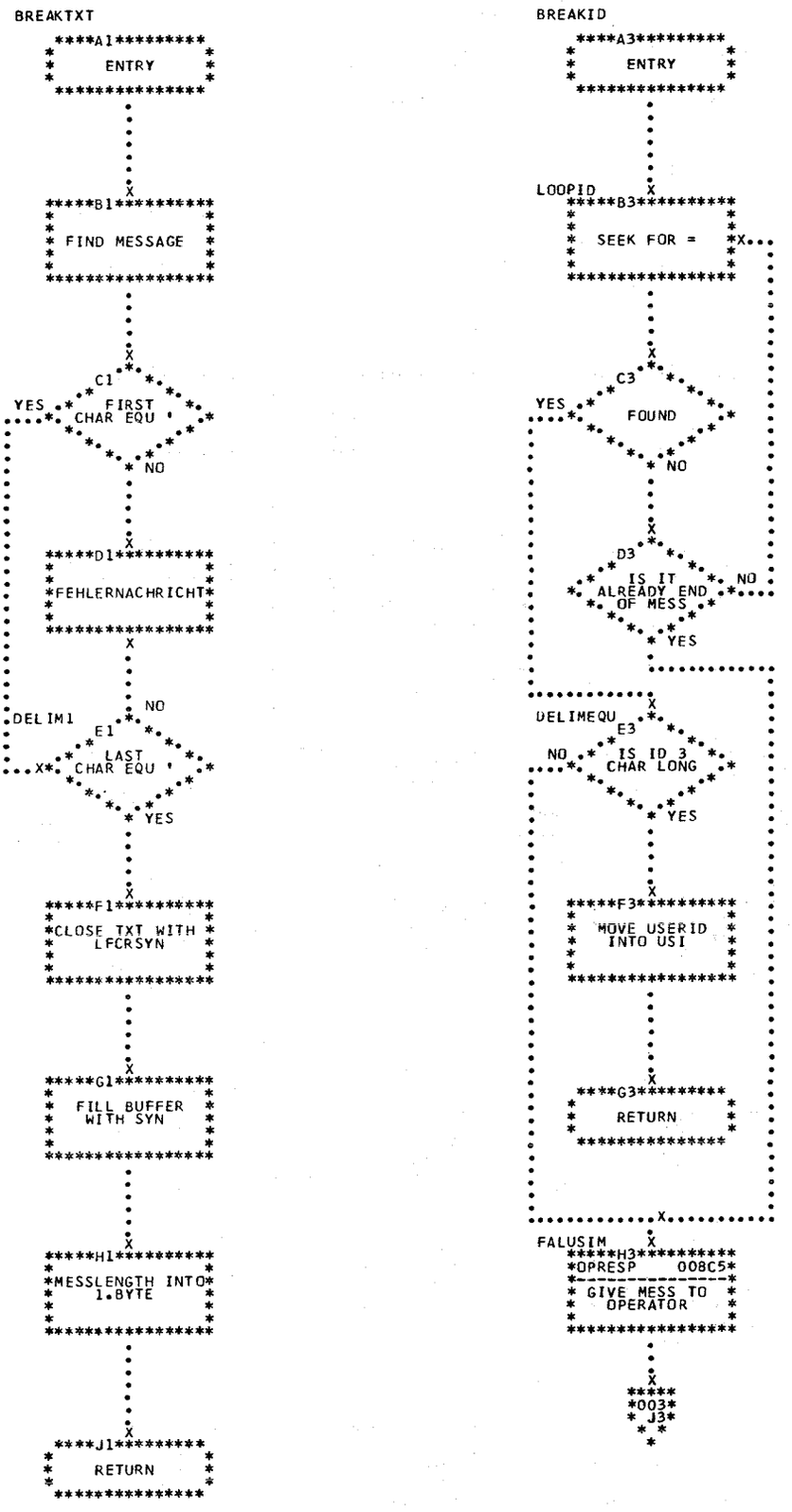
ENTRY FROM BASE, EXIT TO BASE

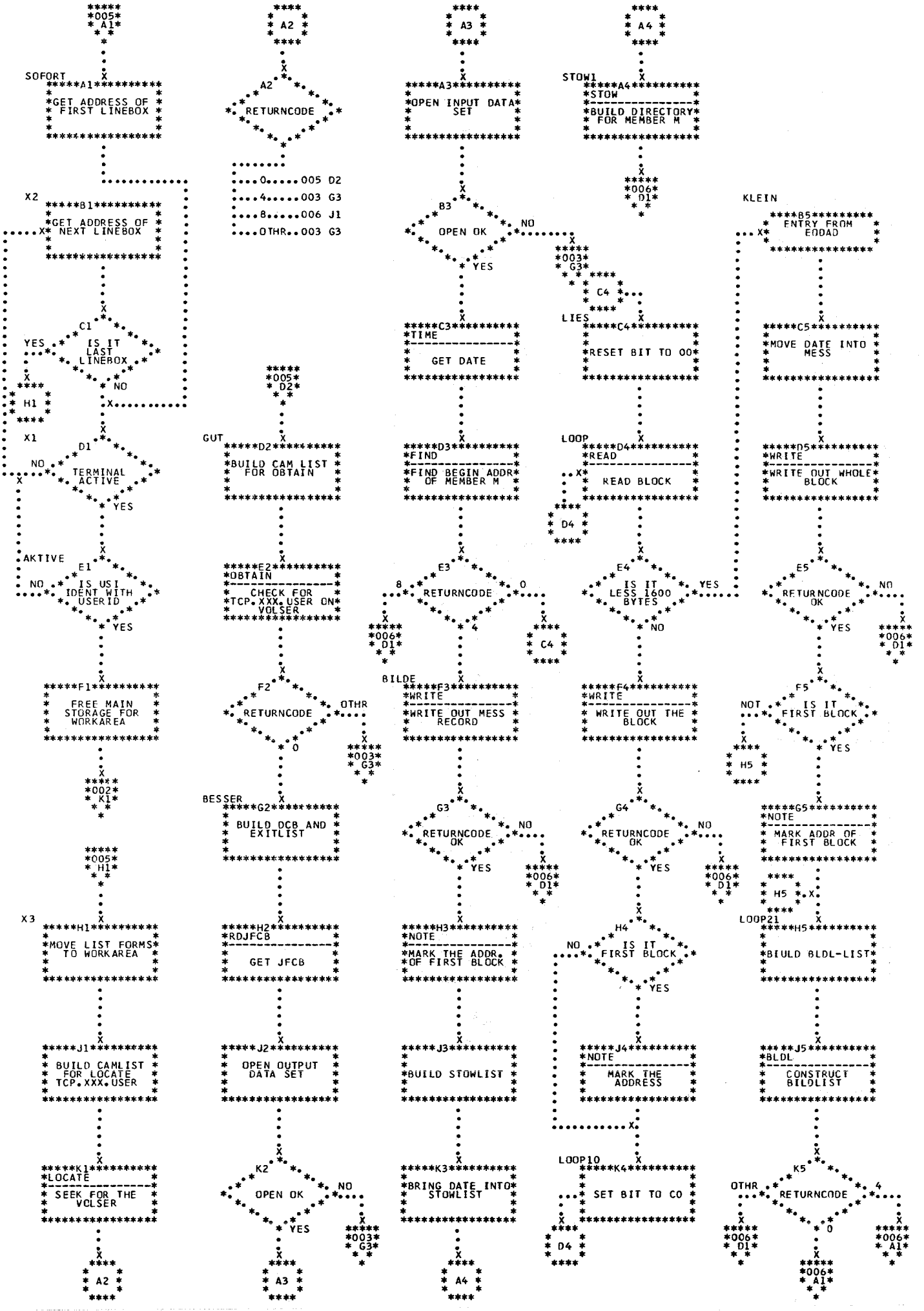
INPUT: POINTER TO CSCB, EXIT: ACCORDING TO ACTIONWORD IN CIB

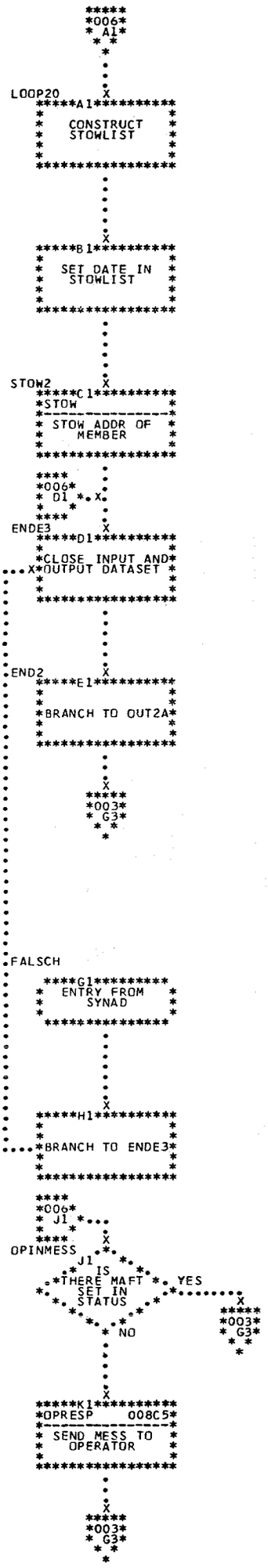



002 A1
A2
A3
OPBDCST
A1 BREAKTXT 004A1
BREAK OUT MESS TEXT
B1 OPRESP 008C5
GIVE MESS TO OPERATOR
002 C1
OPBSTOMS
C1 SET MAFT IN STATUS
D1 GET FIRST LINEBOX
OPCM5
E1 IS IT KONBOX
YES
NO
F1 RESET MAFT IN KONBOX
003 J3
OPCM4
H1 IS TERMINAL ONLINE
YES
NO
K1
OPCM2
J1 GET NEXT LINEBOX, TRY AGAIN
E1 002 K1
J1
OPCM3
K1 GET IOBADDRESS INTO RIOB
A2
MOVEBUF
D2 MOVE MESS FROM OLD BUFFER INTO NEW BUFFER
E2 SAVE BUFFERADD INTO RWKSAVE
E1
QBUF
F2 FILL MESSLENGTH INTO MSBUFADS
G2 FIND END OF BUFFER QUEUE
SAVAD
H2 CHAIN BUFFER TO QUEUE
J2 IS MAFT IN STATUS SET
YES
NO
K2 ANY BIT SET IN ACTION
YES
NO
A3
A2 GET OLD BUFFERADD FROM RWKSAVE
B2 IS MESS ALREADY TRANSLATED
YES
NO
C2 TRANSLATE MESS FROM EBCDIC TO ISO7
D2 MOVE MESS FROM OLD BUFFER INTO NEW BUFFER
E2 SAVE BUFFERADD INTO RWKSAVE
A3 SET MSOUT IN ACTION
B3 ANOTHER CIB CHAINED
NO
YES
003 A5
C3 SET OPMESS IN STATUS
003 A5
SETMSGPD
E3 SET MSGPD IN TFLG
F3 IS IT MESS TCP14
YES
NO
G3 IS PREPARE COMMAND ACTIV
NO
YES
H3 HALTIO
BREAK COMMANDCHAIN WITH HALTIO
MAFTTEST
J3 IS MAFT SET IN STATUS
NO
YES
003 J3
K3 REQUEST ONE BUFFER
J1
A3
A2

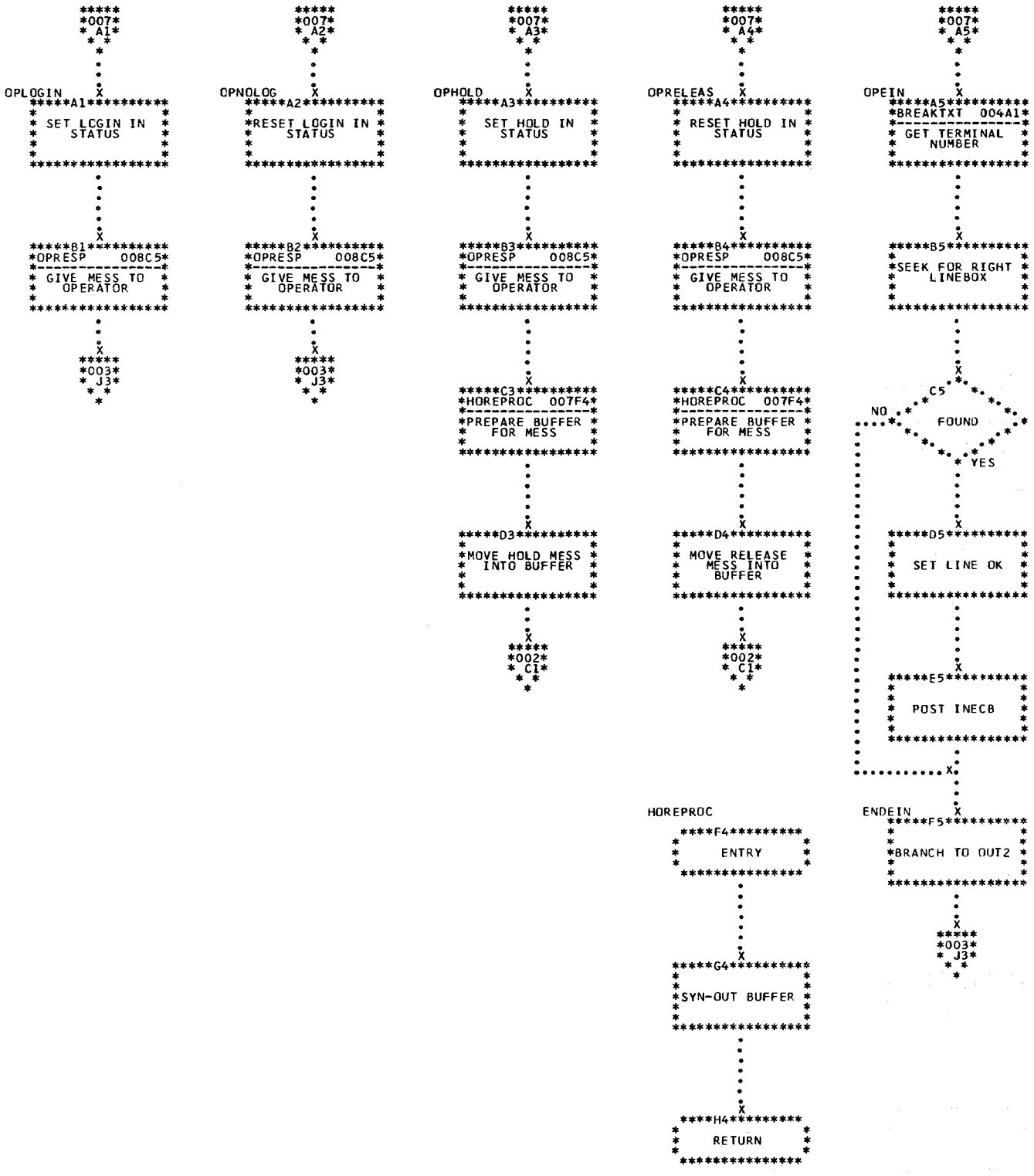

OPALL
A1
ENTRY FOR ALL USERS
B1
BREAKTXT 004A1
BREAK OUT MESS TEXT
C1
OPRESP 008C5
GIVE MESS TO OPERATOR
D1
GET MAIN STORAGE FOR WORKAREA
E1
SET MAFT IN STATUS
F1
INSERT ID=QQQ FOR DATA SET QQQ
005 H1
OPWIM
A2
ENTRY FOR ONE USER WITHOUT RESP
B2
BREAKID 004A3
BREAK OUT USER ID
C2
BREAKTXT 004A1
BREAK OUT MESS TEXT
OPID
A3
ENTRY FOR ONE USER
B3
BREAKID 004A3
BREAK OUT USER ID
C3
BREAKTXT 004A1
BREAK OUT MESS TEXT
D3
OPRESP 008C5
GIVE MESS TO OPERATOR
OPIDLA81
E3
GET MAIN STORAGE FOR WORKAREA
005 A1
003 G3
OUT2A
G3
FREE MAIN STORAGE FOR WORKAREA
H3
IS THERE MAFT SET IN STATUS
YES
NO
002 C1
003 J3
OUT2
J3
RELEASE LAST BUFFER
OUT2B
K3
MORE CIB CHAINED
NO
YES
001 C1
BRDYNOUT
A5
LOAD BRANCH ADDRESS
B5
LOAD BASEREG FOR TCPWAIT
C5
BRANCH TO DYNOUT
BTCPWAIT
F5
LOAD ENTRYPOINT AND BASEREG FOR TCPWAIT
G5
BRANCH TO TCPWAIT

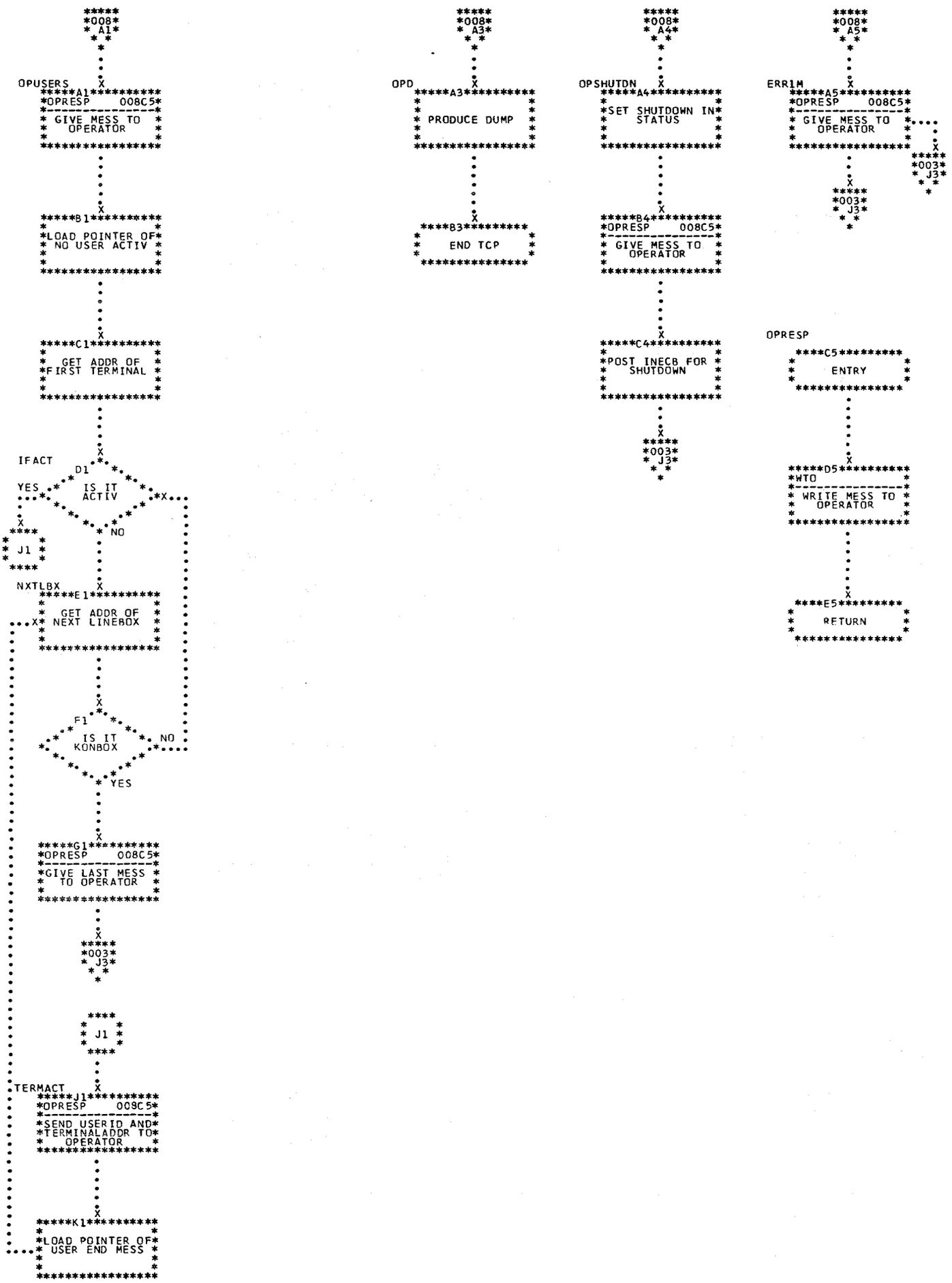

BREAKTXT
A1
ENTRY
B1
FIND MESSAGE
C1
FIRST CHAR EQU '
YES
NO
D1
FEHLERNACHRICHT
DELIM1
E1
LAST CHAR EQU '
NO
YES
F1
CLOSE TXT WITH LFCRSYN
G1
FILL BUFFER WITH SYN
H1
MESSLENGTH INTO 1.BYTE
J1
RETURN
BREAKID
A3
ENTRY
LOOPID
B3
SEEK FOR =
C3
FOUND
YES
NO
D3
IS IT ALREADY END OF MESS
NO
YES
DELIMEQU
E3
IS ID 3 CHAR LONG
NO
YES
F3
MOVE USERID INTO USI
G3
RETURN
FALUSIM
H3
OPRESP 008C5
GIVE MESS TO OPERATOR
003
J3

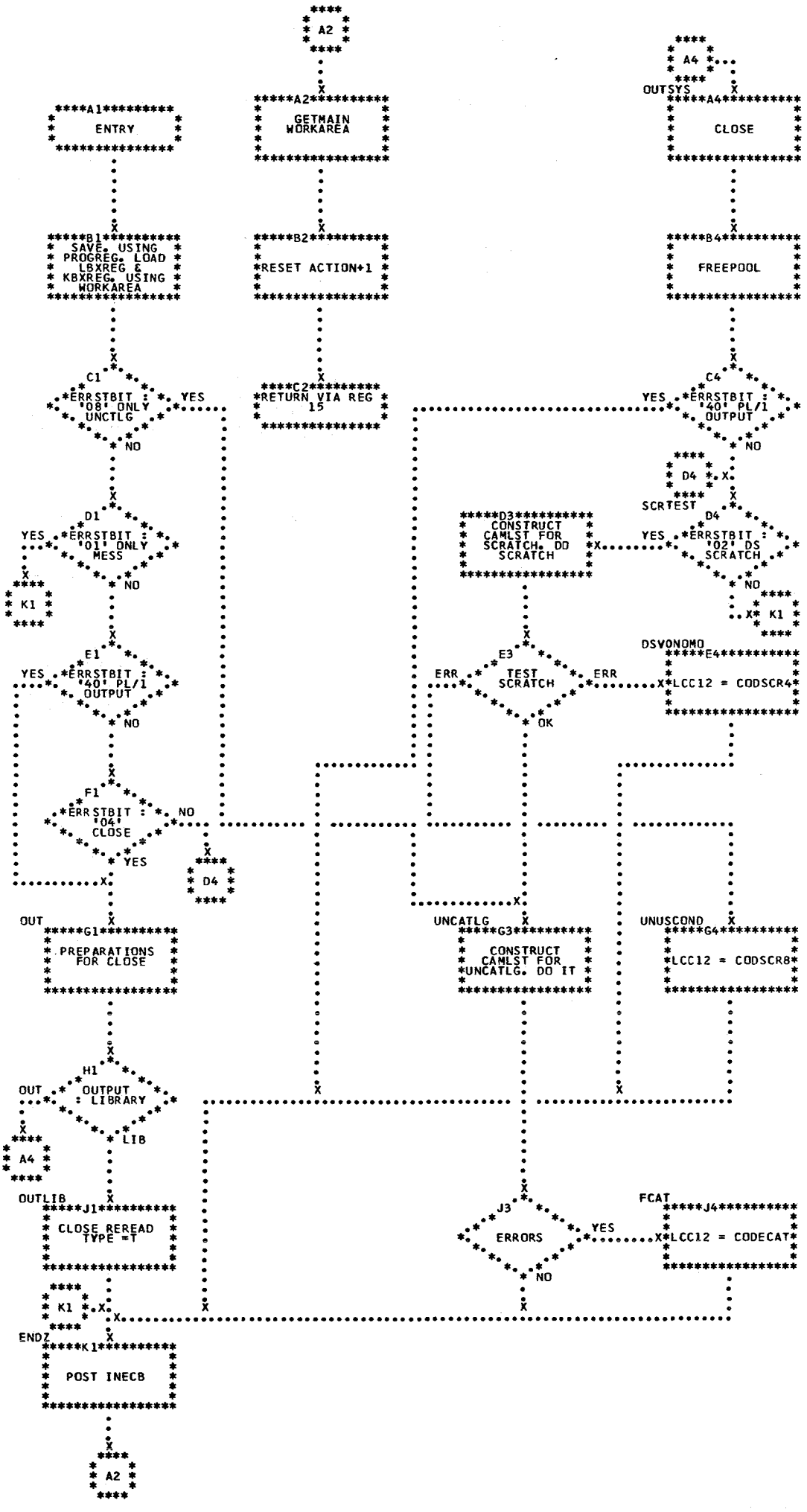

A1
SOFORT
GET ADDRESS OF FIRST LINEBOX
A2
RETURNCODE
0 005 D2
4 003 G3
8 006 J1
OTHR 003 G3
A3
OPEN INPUT DATA SET
A4
STOW1
STOW
BUILD DIRECTORY FOR MEMBER M
006 D1
KLEIN
B1
GET ADDRESS OF NEXT LINEBOX
B3
OPEN OK
NO
YES
003 G3
B5
ENTRY FROM EODAD
C1
IS IT LAST LINEBOX
YES
NO
H1
C3
TIME
GET DATE
C4
LIES
RESET BIT TO 00
C5
MOVE DATE INTO MESS
D1
TERMINAL ACTIVE
NO
YES
005 D2
D2
BUILD CAM LIST FOR OBTAIN
D3
FIND
FIND BEGIN ADDR OF MEMBER M
D4
LOOP
READ
READ BLOCK
D5
WRITE
WRITE OUT WHOLE BLOCK
AKTIVE
E1
IS USI IDENT WITH USERID
E2
OBTAIN
CHECK FOR TCP.XXX.USER ON VOLSER
E3
RETURNCODE
8
0
4
BILDE
E4
IS IT LESS 1600 BYTES
E5
RETURNCODE OK
F1
FREE MAIN STORAGE FOR WORKAREA
002 K1
F2
RETURNCODE
OTHR
0
003 G3
F3
WRITE
WRITE OUT MESS RECORD
F4
WRITE
WRITE OUT THE BLOCK
F5
NOT
IS IT FIRST BLOCK
H5
BESSER
G2
BUILD DCB AND EXITLIST
G3
RETURNCODE OK
NO
YES
006 D1
G4
RETURNCODE OK
G5
NOTE
MARK ADDR OF FIRST BLOCK
LOOP21
005 H1
X3
H1
MOVE LIST FORMS TO WORKAREA
H2
RDJFCB
GET JFCB
H3
NOTE
MARK THE ADDR. OF FIRST BLOCK
H4
NO
IS IT FIRST BLOCK
H5
BIULD BLDL-LIST
J1
BUILD CAMLIST FOR LOCATE TCP.XXX.USER
J2
OPEN OUTPUT DATA SET
J3
BUILD STOWLIST
J4
NOTE
MARK THE ADDRESS
J5
BLDL
CONSTRUCT BILDLIST
K1
LOCATE
SEEK FOR THE VOLSER
K2
OPEN OK
NO
YES
003 G3
K3
BRING DATE INTO STOWLIST
LOOP10
K4
SET BIT TO CO
D4
K5
OTHR
RETURNCODE
4
0
006 A1
A2
A3
A4


006
* A1*
LOOP20
*****A1*****
CONSTRUCT
STOWLIST
*****B1*****
SET DATE IN
STOWLIST
STOW2
*****C1*****
*STOW
STOW ADDR OF
MEMBER
006
* D1 *
ENDE3
*****D1*****
CLOSE INPUT AND
*OUTPUT DATASET *
END2
*****E1*****
BRANCH TO OUT2A
003
* G3*
FALSCH
****G1*****
ENTRY FROM
SYNAD
*****H1*****
BRANCH TO ENDE3
006
* J1 *
OPINMESS
J1
IS
THERE MAFT
YES
SET IN
STATUS
NO
003
* G3*
*****K1*****
OPRESP 008C5
SEND MESS TO
OPERATOR
003
* G3*

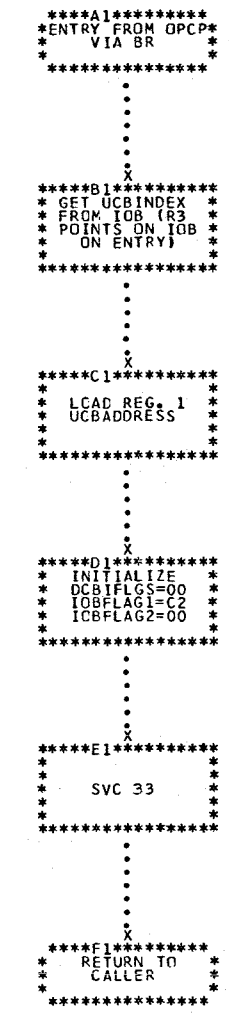
OTHER ACTIONROUTINS:OPLOGIN,OPNOLOG,OPHOLD,OPRELEAS,OPEIN,OPUSERS,OPD,OPSHUTDN

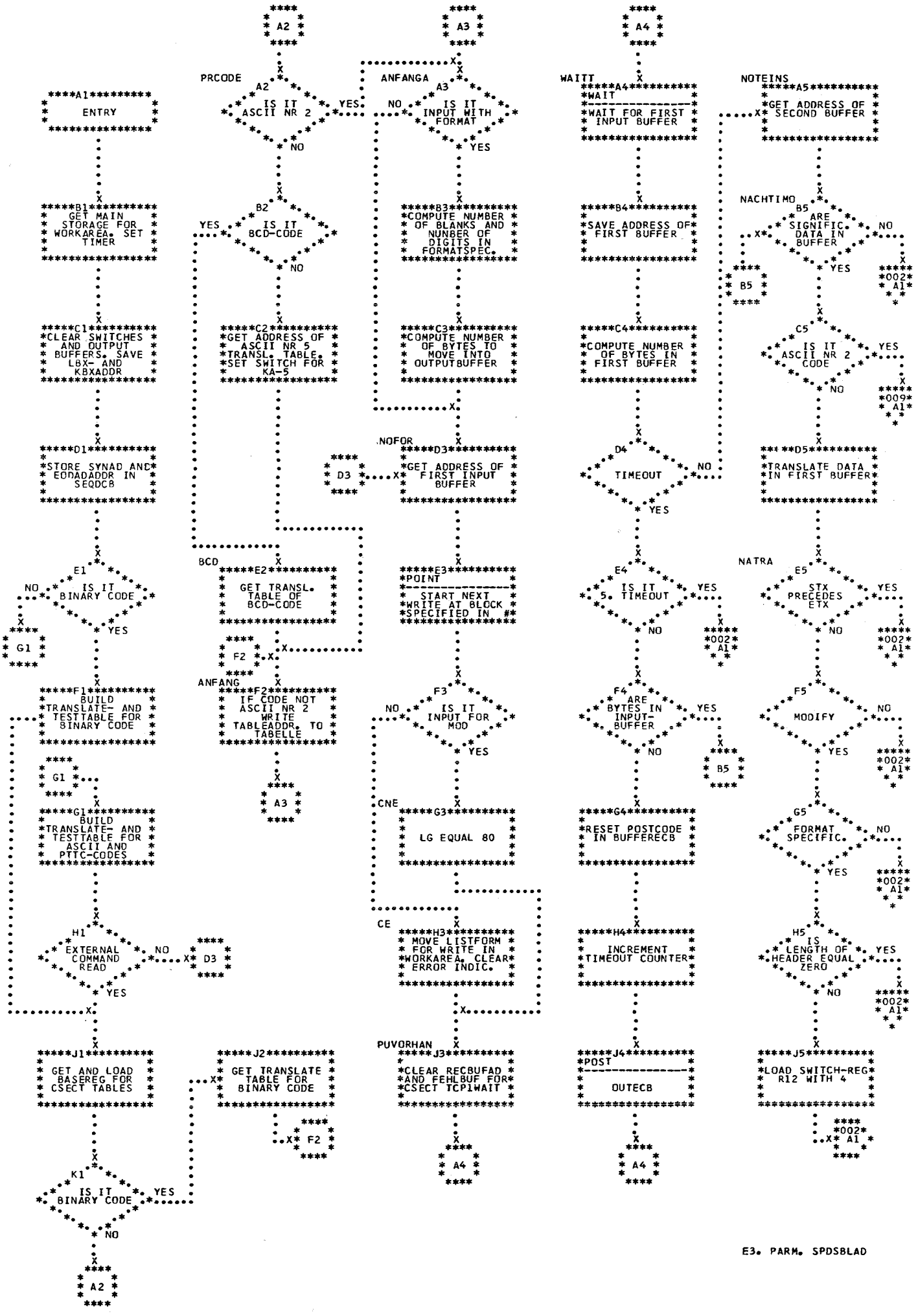



OPUSERS
A1
OPRESP 008C5
GIVE MESS TO OPERATOR
B1
LOAD POINTER OF NO USER ACTIV
C1
GET ADDR OF FIRST TERMINAL
IFACT
D1
IS IT ACTIV
YES
NO
J1
NXTLBX
E1
GET ADDR OF NEXT LINEBOX
F1
IS IT KONBOX
NO
YES
G1
OPRESP 008C5
GIVE LAST MESS TO OPERATOR
003
J3
J1
TERMACT
J1
OPRESP 008C5
SEND USERID AND TERMINALADDR TO OPERATOR
K1
LOAD POINTER OF USER END MESS
OPD
A3
PRODUCE DUMP
B3
END TCP
OPSHUTDN
A4
SET SHUTDOWN IN STATUS
B4
OPRESP 008C5
GIVE MESS TO OPERATOR
C4
POST INECB FOR SHUTDOWN
003
J3
ERR1M
A5
OPRESP 008C5
GIVE MESS TO OPERATOR
003
J3
OPRESP
C5
ENTRY
D5
WTO
WRITE MESS TO OPERATOR
E5
RETURN

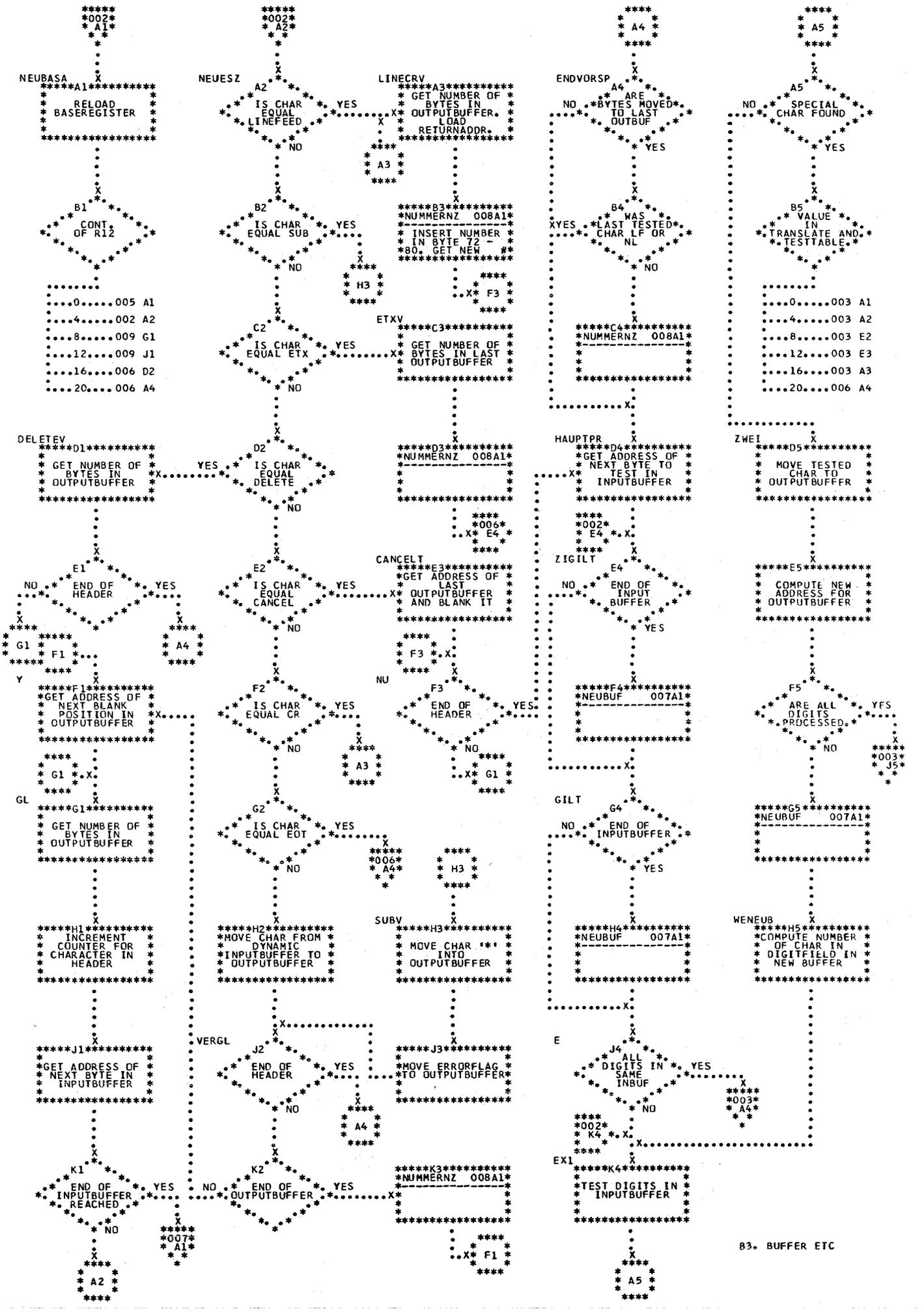

A2
A1
ENTRY
A2
GETMAIN WORKAREA
A4
OUTSYS
A4
CLOSE
B1
SAVE, USING PROGREG, LOAD LBXREG &, KBXREG, USING WORKAREA
B2
RESET ACTION+1
B4
FREEPOOL
C1
ERRSTBIT : '08' ONLY UNCTLG
YES
NO
C2
RETURN VIA REG 15
C4
ERRSTBIT : '40' PL/1 OUTPUT
YES
NO
D1
ERRSTBIT : '01' ONLY MESS
YES
NO
K1
D3
CONSTRUCT CAMLST FOR SCRATCH, DO SCRATCH
D4
SCRTEST
D4
ERRSTBIT : '02' DS SCRATCH
YES
NO
K1
E1
ERRSTBIT : '40' PL/1 OUTPUT
YES
NO
E3
TEST SCRATCH
ERR
OK
ERR
DSVONOMO
E4
LCC12 = CODSCR4
F1
ERRSTBIT : '04' CLOSE
NO
YES
D4
OUT
G1
PREPARATIONS FOR CLOSE
UNCATLG
G3
CONSTRUCT CAMLST FOR UNCATLG, DO IT
UNUSCOND
G4
LCC12 = CODSCR8
H1
OUT : OUTPUT : LIBRARY
LIB
A4
OUTLIB
J1
CLOSE REREAD TYPE =T
J3
ERRORS
YES
NO
FCAT
J4
LCC12 = CODECAT
K1
ENDZ
K1
POST INECB
A2

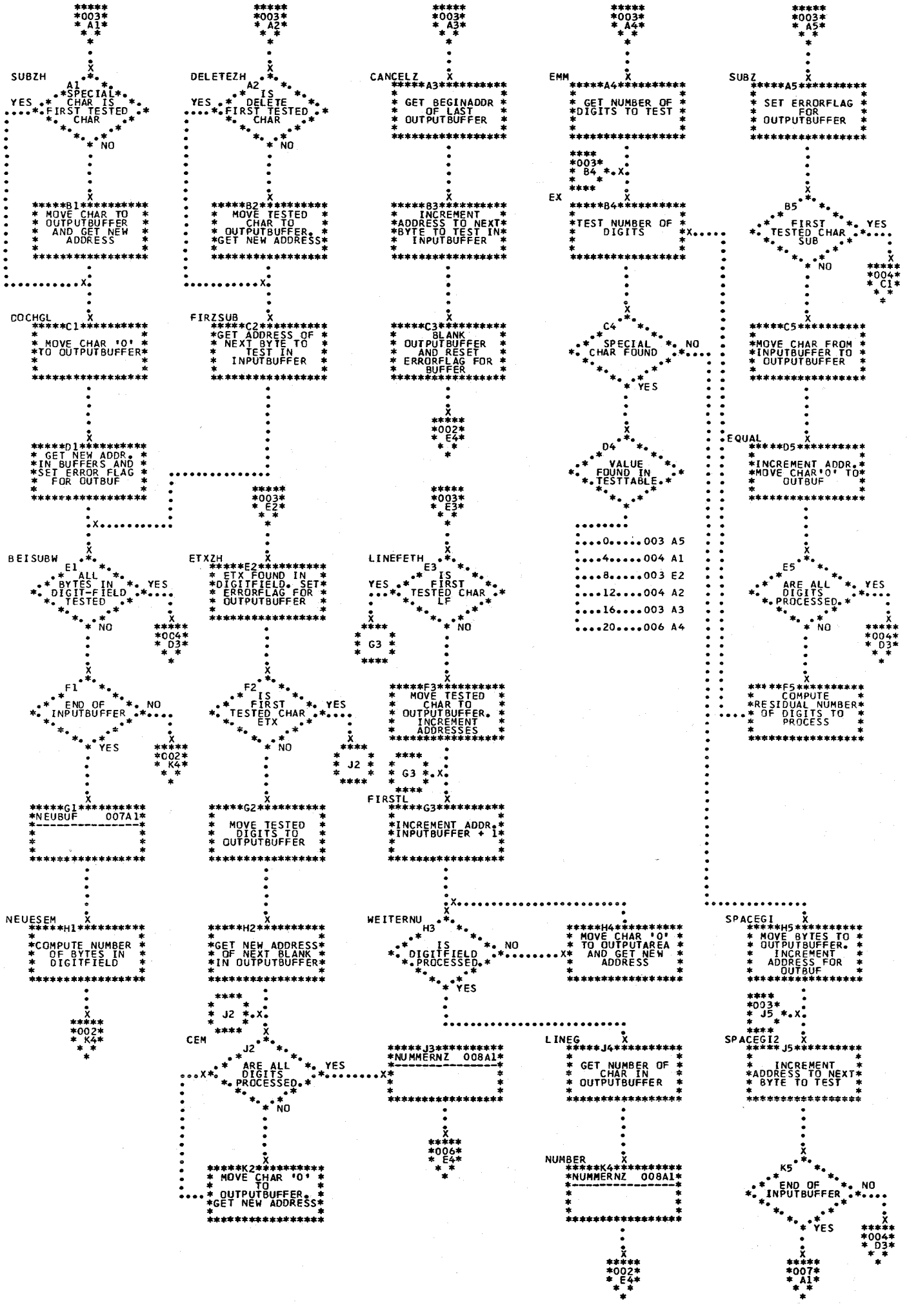
TCP1HALT

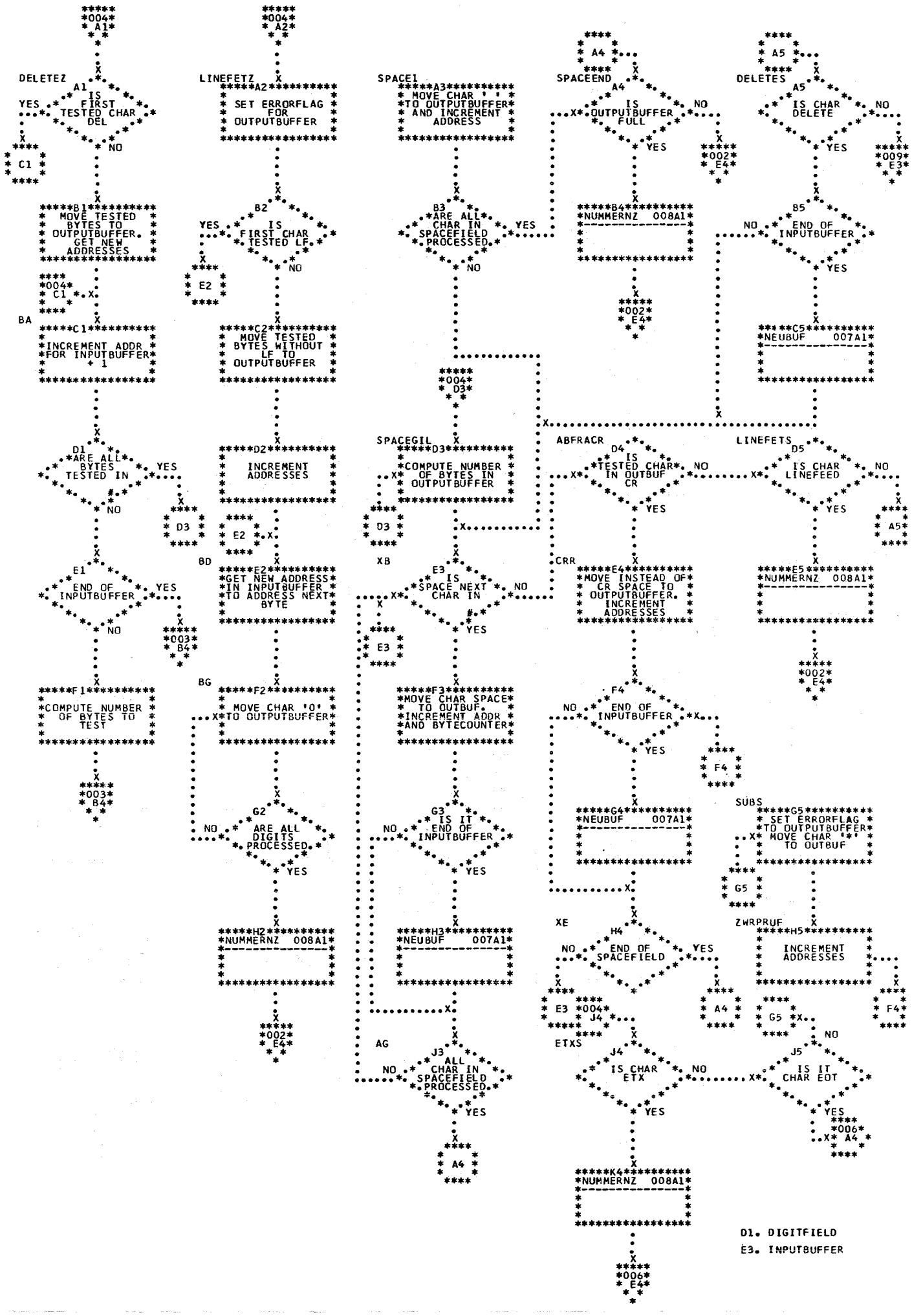

A1
ENTRY FROM OPCP
VIA BR
B1
GET UCBINDEX
FROM IOB (R3
POINTS ON IOB
ON ENTRY)
C1
LOAD REG. 1
UCBADDRESS
D1
INITIALIZE
DCBIFLGS=00
IOBFLAG1=C2
IOBFLAG2=00
E1
SVC 33
F1
RETURN TO
CALLER

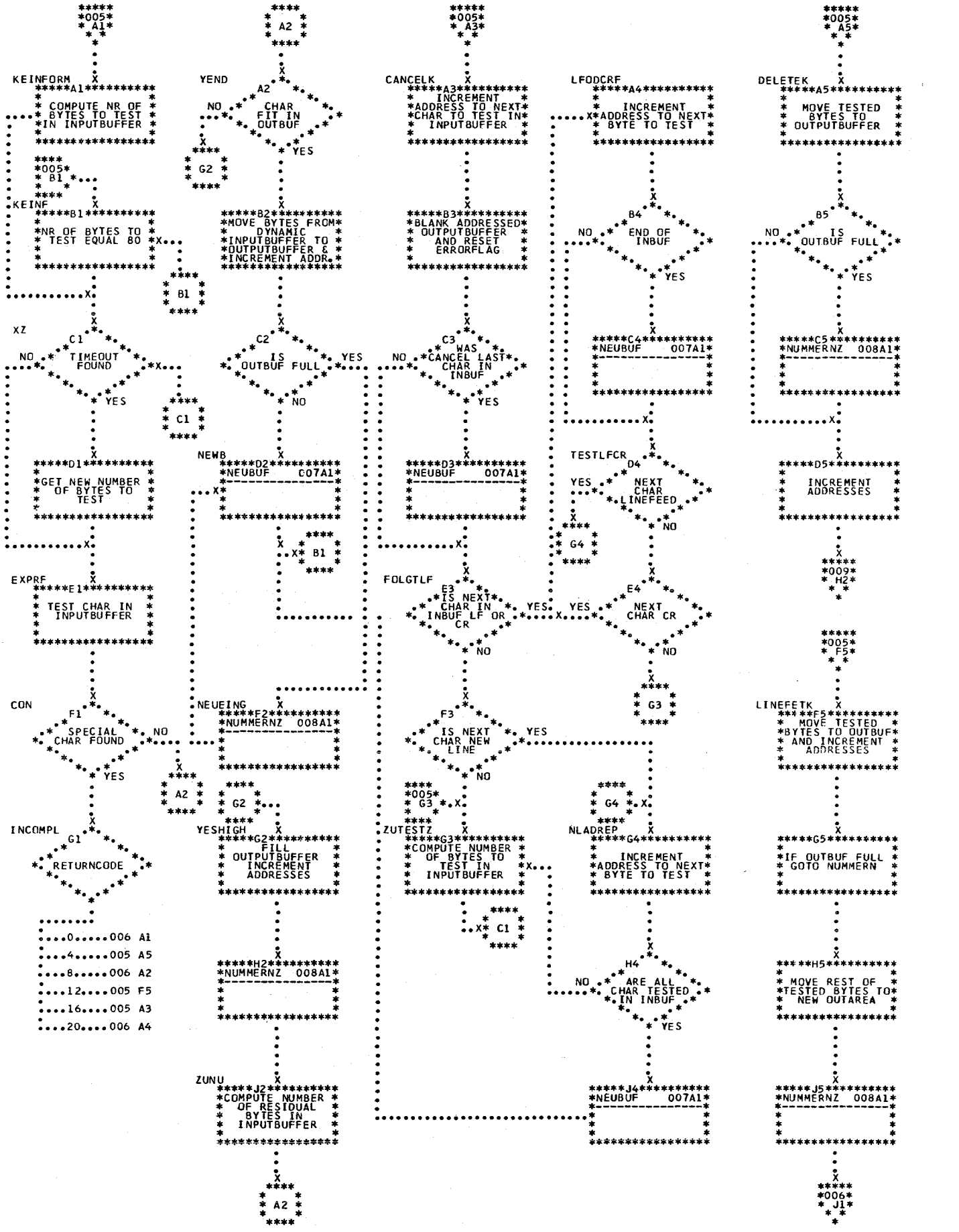


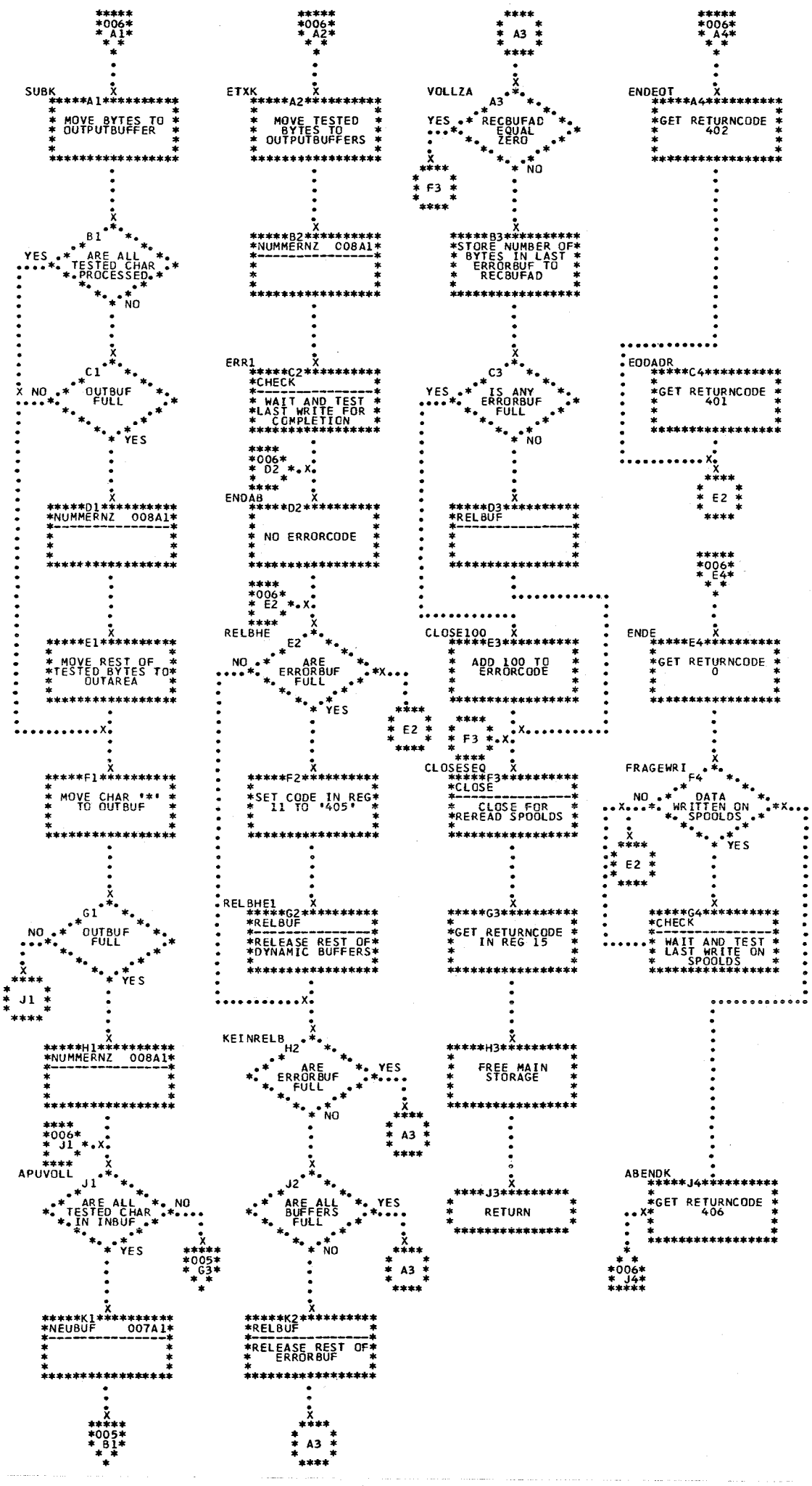
E3. PARM. SPDSBLAD

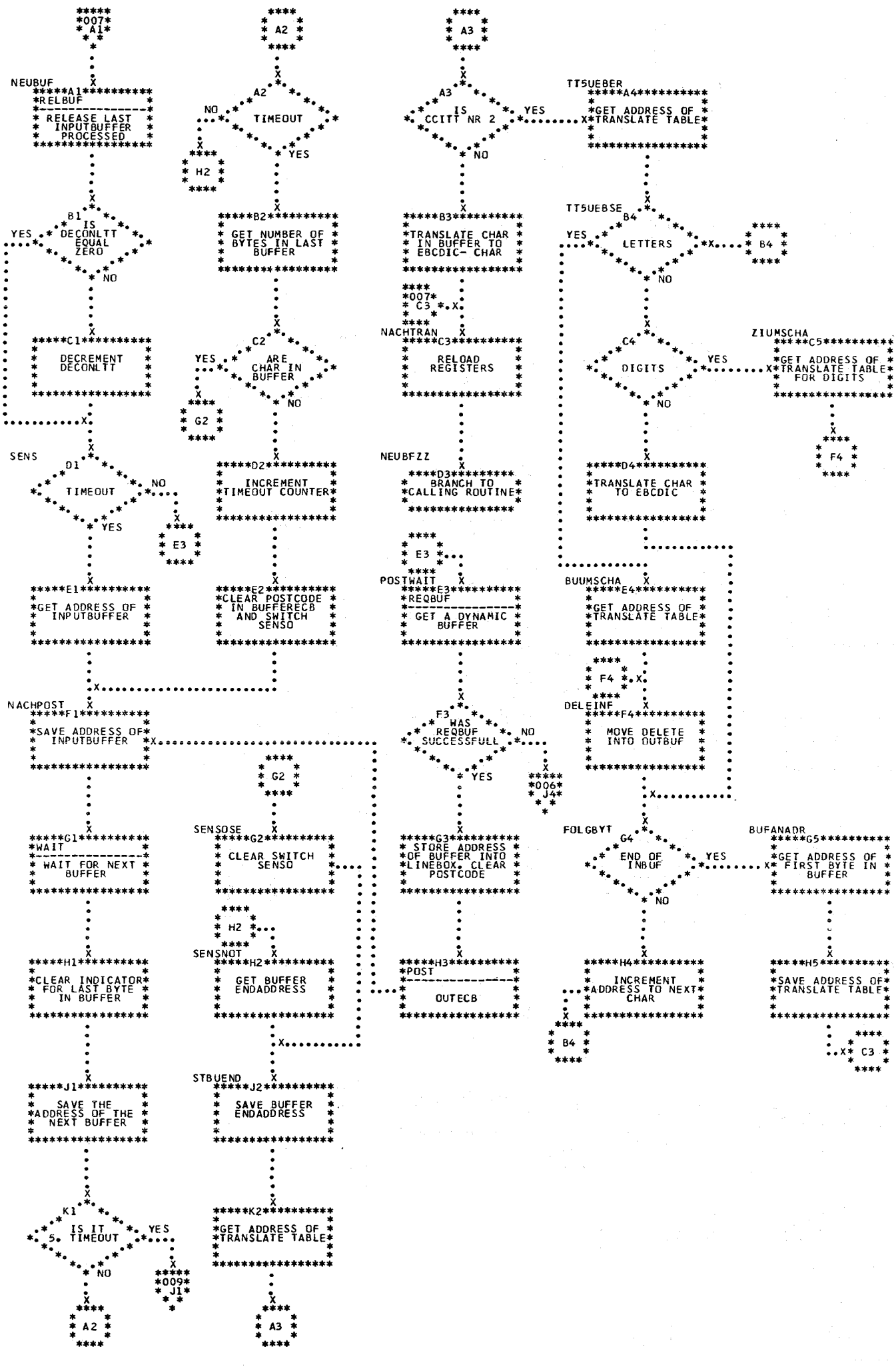

A1
RELOAD BASEREGISTER
B1
CONT. OF R12
0 005 A1
4 002 A2
8 009 G1
12 009 J1
16 006 D2
20 006 A4
A2
B2
C2
D2
E1
E2
F2
G2
H1
J1
K1
K2
A3
A4
H3
F3
E4
G4
J4
K4
A5
B5
C5
D5
E5
F5
G5
H5
J5
NEUBASA
NEUESZ
LINECRV
ETXV
DELETEV
CANCELT
ZIGILT
GILT
ENDVORSP
HAUPTPR
ZWEI
SUBV
VERGL
WENEUB
EX1
GL
NU
Y
E
IS CHAR EQUAL LINEFEED
IS CHAR EQUAL SUB
IS CHAR EQUAL ETX
IS CHAR EQUAL DELETE
END OF HEADER
IS CHAR EQUAL CANCEL
IS CHAR EQUAL CR
IS CHAR EQUAL EOT
END OF INPUTBUFFER REACHED
END OF OUTPUTBUFFER
ARE BYTES MOVED TO LAST OUTBUF
WAS LAST TESTED CHAR LF OR NL
END OF INPUT BUFFER
END OF INPUTBUFFER
ALL DIGITS IN SAME INBUF
SPECIAL CHAR FOUND
VALUE IN TRANSLATE AND TESTTABLE
ARE ALL DIGITS PROCESSED
GET NUMBER OF BYTES IN OUTPUTBUFFER
INSERT NUMBER IN BYTE 72 - 80. GET NEW
GET NUMBER OF BYTES IN LAST OUTPUTBUFFER
GET ADDRESS OF NEXT BLANK POSITION IN OUTPUTBUFFER
GET ADDRESS OF NEXT BYTE TO TEST IN INPUTBUFFER
MOVE TESTED CHAR TO OUTPUTBUFFER
COMPUTE NEW ADDRESS FOR OUTPUTBUFFER
GET ADDRESS OF LAST OUTPUTBUFFER AND BLANK IT
MOVE CHAR '*' INTO OUTPUTBUFFER
MOVE CHAR FROM DYNAMIC INPUTBUFFER TO OUTPUTBUFFER
MOVE ERRORFLAG TO OUTPUTBUFFER
TEST DIGITS IN INPUTBUFFER
COMPUTE NUMBER OF CHAR IN DIGITFIELD IN NEW BUFFER
NEUBUF 007A1
NUMMERNZ 008A1
B3. BUFFER ETC

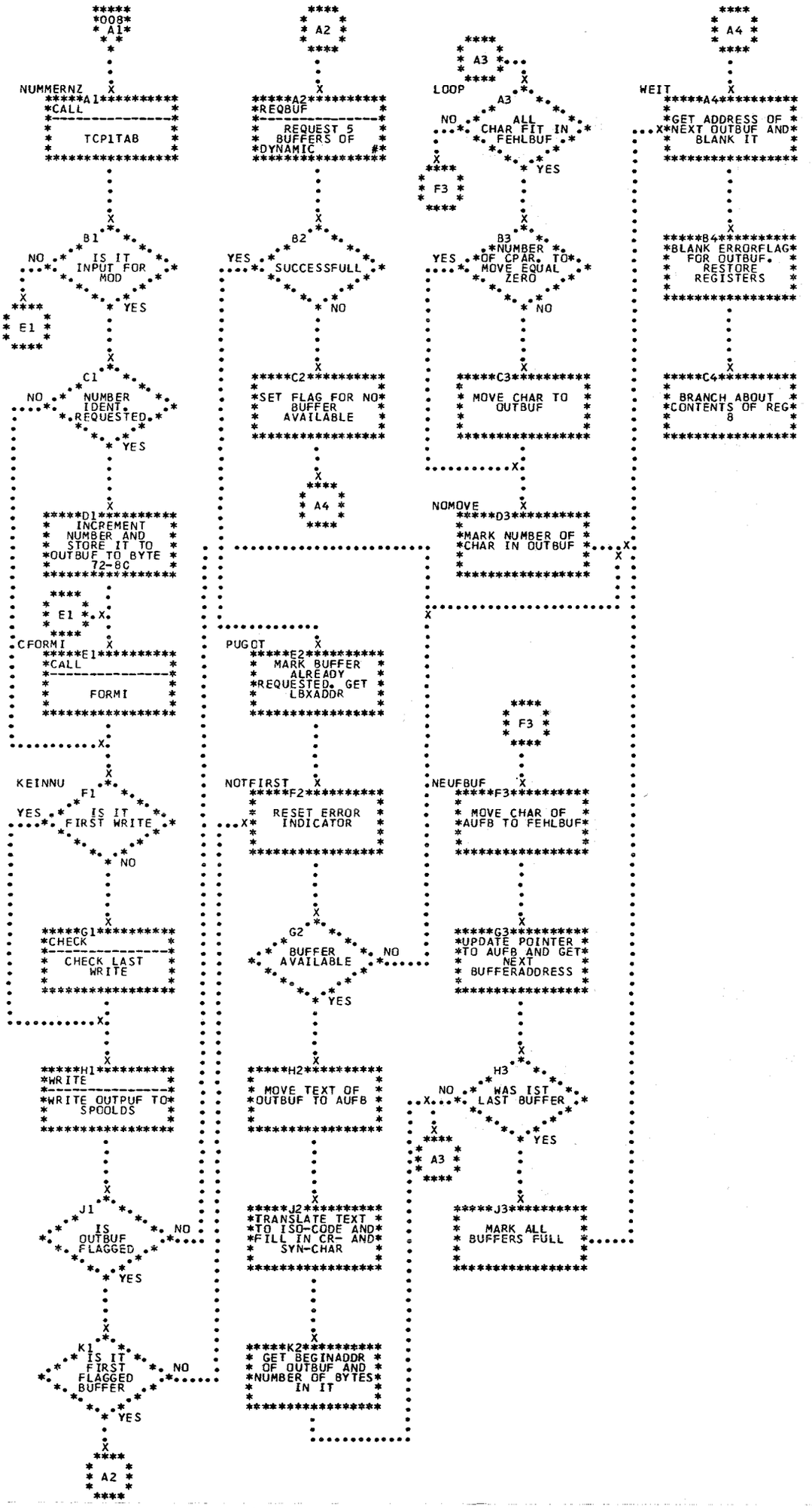

SUBZH
A1
SPECIAL CHAR IS FIRST TESTED CHAR
YES
NO
B1 MOVE CHAR TO OUTPUTBUFFER AND GET NEW ADDRESS
COCHGL
C1 MOVE CHAR '0' TO OUTPUTBUFFER
D1 GET NEW ADDR. IN BUFFERS AND SET ERROR FLAG FOR OUTBUF
BEISUBW
E1 ALL BYTES IN DIGIT-FIELD TESTED
YES
NO
004 D3
F1 END OF INPUTBUFFER
NO
YES
002 K4
G1 NEUBUF 007A1
NEUESEM
H1 COMPUTE NUMBER OF BYTES IN DIGITFIELD
002 K4
DELETEZH
A2 IS DELETE FIRST TESTED CHAR
YES
NO
B2 MOVE TESTED CHAR TO OUTPUTBUFFER, GET NEW ADDRESS
FIRZSUB
C2 GET ADDRESS OF NEXT BYTE TO TEST IN INPUTBUFFER
E2
ETXZH
E2 ETX FOUND IN DIGITFIELD, SET ERRORFLAG FOR OUTPUTBUFFER
F2 IS FIRST TESTED CHAR ETX
YES
NO
J2
G2 MOVE TESTED DIGITS TO OUTPUTBUFFER
H2 GET NEW ADDRESS OF NEXT BLANK IN OUTPUTBUFFER
CEM
J2 ARE ALL DIGITS PROCESSED
YES
NO
K2 MOVE CHAR '0' TO OUTPUTBUFFER, GET NEW ADDRESS
CANCELZ
A3 GET BEGINADDR OF LAST OUTPUTBUFFER
B3 INCREMENT ADDRESS TO NEXT BYTE TO TEST IN INPUTBUFFER
C3 BLANK OUTPUTBUFFER AND RESET ERRORFLAG FOR BUFFER
002 E4
E3
LINEFETH
E3 IS FIRST TESTED CHAR LF
YES
NO
G3
F3 MOVE TESTED CHAR TO OUTPUTBUFFER, INCREMENT ADDRESSES
G3
FIRSTL
G3 INCREMENT ADDR. INPUTBUFFER + 1
WEITERNU
H3 IS DIGITFIELD PROCESSED
NO
YES
H4 MOVE CHAR '0' TO OUTPUTAREA AND GET NEW ADDRESS
J3 NUMMERNZ 008A1
006 E4
LINEG
J4 GET NUMBER OF CHAR IN OUTPUTBUFFER
NUMBER
K4 NUMMERNZ 008A1
002 E4
EMM
A4 GET NUMBER OF DIGITS TO TEST
B4
EX
B4 TEST NUMBER OF DIGITS
C4 SPECIAL CHAR FOUND
NO
YES
D4 VALUE FOUND IN TESTTABLE
0 003 A5
4 004 A1
8 003 E2
12 004 A2
16 003 A3
20 006 A4
SUBZ
A5 SET ERRORFLAG FOR OUTPUTBUFFER
B5 FIRST TESTED CHAR SUB
YES
NO
004 C1
C5 MOVE CHAR FROM INPUTBUFFER TO OUTPUTBUFFER
EQUAL
D5 INCREMENT ADDR. MOVE CHAR '0' TO OUTBUF
E5 ARE ALL DIGITS PROCESSED
YES
NO
004 D3
F5 COMPUTE RESIDUAL NUMBER OF DIGITS TO PROCESS
SPACEGI
H5 MOVE BYTES TO OUTPUTBUFFER, INCREMENT ADDRESS FOR OUTBUF
J5
SPACEGI2
J5 INCREMENT ADDRESS TO NEXT BYTE TO TEST
K5 END OF INPUTBUFFER
NO
YES
004 D3
007 A1


DELETEZ
A1
IS FIRST TESTED CHAR DEL
YES
NO
B1
MOVE TESTED BYTES TO OUTPUTBUFFER. GET NEW ADDRESSES
C1
INCREMENT ADDR FOR INPUTBUFFER + 1
D1
ARE ALL BYTES TESTED IN
YES
NO
D3
E1
END OF INPUTBUFFER
YES
NO
003 B4
F1
COMPUTE NUMBER OF BYTES TO TEST
003 B4
BA
LINEFETZ
A2
SET ERRORFLAG FOR OUTPUTBUFFER
B2
IS FIRST CHAR TESTED LF
YES
NO
E2
C2
MOVE TESTED BYTES WITHOUT LF TO OUTPUTBUFFER
D2
INCREMENT ADDRESSES
E2
GET NEW ADDRESS IN INPUTBUFFER TO ADDRESS NEXT BYTE
BD
F2
MOVE CHAR '0' TO OUTPUTBUFFER
G2
ARE ALL DIGITS PROCESSED
NO
YES
H2
NUMMERNZ 008A1
002 E4
BG
SPACE1
A3
MOVE CHAR ' ' TO OUTPUTBUFFER AND INCREMENT ADDRESS
B3
ARE ALL CHAR IN SPACEFIELD PROCESSED
YES
NO
D3
SPACEGIL
D3
COMPUTE NUMBER OF BYTES IN OUTPUTBUFFER
D3
XB
E3
IS SPACE NEXT CHAR IN
NO
YES
E3
F3
MOVE CHAR SPACE TO OUTBUF. INCREMENT ADDR AND BYTECOUNTER
G3
IS IT END OF INPUTBUFFER
NO
YES
H3
NEUBUF 007A1
AG
J3
ALL CHAR IN SPACEFIELD PROCESSED
NO
YES
A4
SPACEEND
A4
IS OUTPUTBUFFER FULL
NO
YES
002 E4
B4
NUMMERNZ 008A1
002 E4
DELETES
A5
IS CHAR DELETE
NO
YES
009 E3
B5
END OF INPUTBUFFER
NO
YES
C5
NEUBUF 007A1
ABFRACR
D4
IS TESTED CHAR IN OUTBUF CR
NO
YES
CRR
E4
MOVE INSTEAD OF CR SPACE TO OUTPUTBUFFER. INCREMENT ADDRESSES
F4
END OF INPUTBUFFER
NO
YES
F4
G4
NEUBUF 007A1
XE
H4
END OF SPACEFIELD
NO
YES
E3 004 J4
A4
ETXS
J4
IS CHAR ETX
NO
YES
K4
NUMMERNZ 008A1
006 E4
LINEFETS
D5
IS CHAR LINEFEED
NO
YES
A5
E5
NUMMERNZ 008A1
002 E4
SUBS
G5
SET ERRORFLAG TO OUTPUTBUFFER MOVE CHAR '*' TO OUTBUF
G5
ZWRPRUF
H5
INCREMENT ADDRESSES
F4
J5
IS IT CHAR EOT
NO
YES
006 A4
D1. DIGITFIELD
E3. INPUTBUFFER

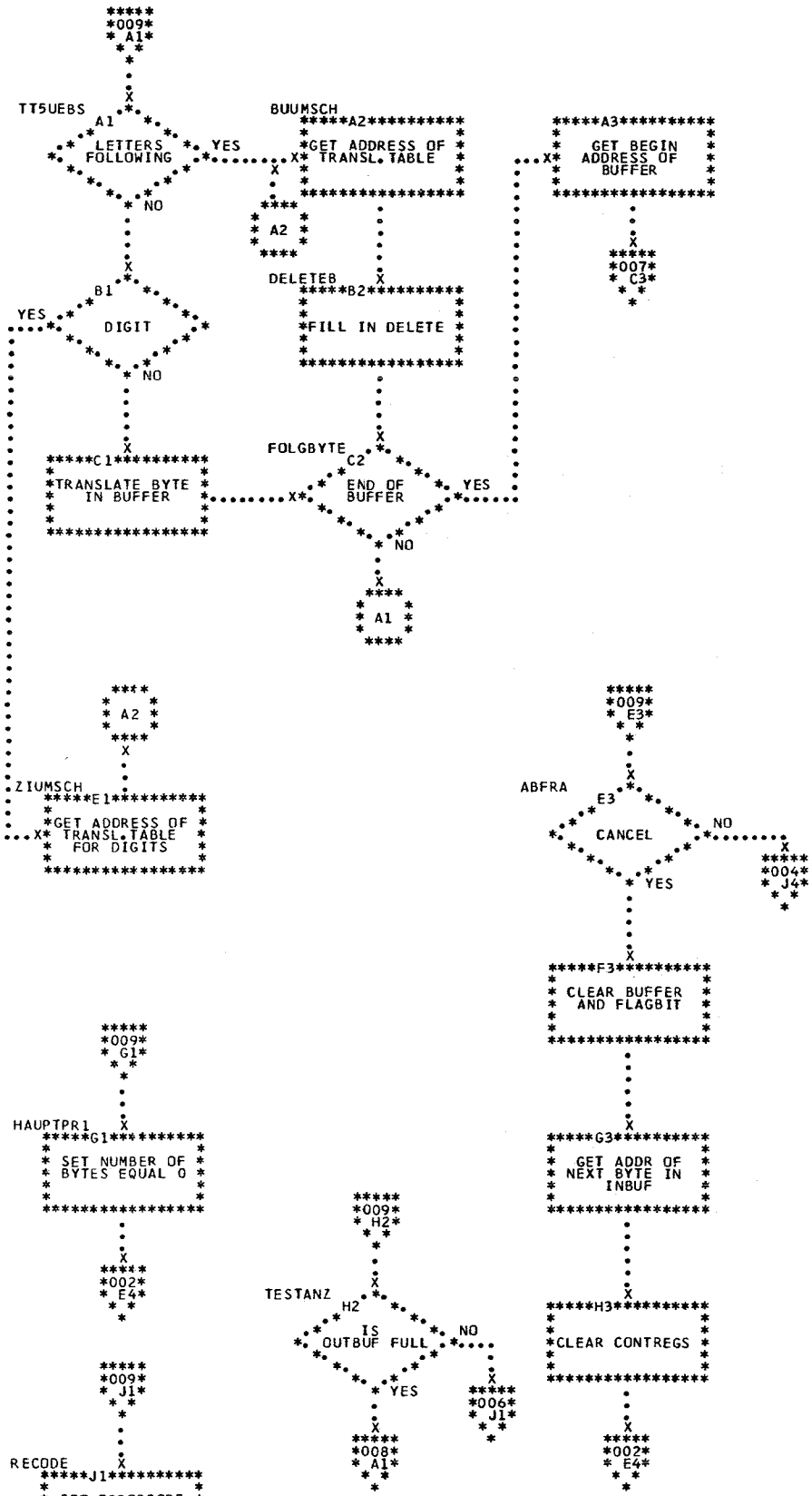

005 A1
KEINFORM
A1
COMPUTE NR OF BYTES TO TEST IN INPUTBUFFER
005 B1
KEINF
B1
NR OF BYTES TO TEST EQUAL 80
XZ
C1
TIMEOUT FOUND
NO
YES
005 C1
D1
GET NEW NUMBER OF BYTES TO TEST
EXPRF
E1
TEST CHAR IN INPUTBUFFER
CON
F1
SPECIAL CHAR FOUND
NO
YES
INCOMPL
G1
RETURNCODE
0 006 A1
4 005 A5
8 006 A2
12 005 F5
16 005 A3
20 006 A4
005 A2
YEND
A2
CHAR FIT IN OUTBUF
NO
YES
005 G2
B2
MOVE BYTES FROM DYNAMIC INPUTBUFFER TO OUTPUTBUFFER & INCREMENT ADDR.
C2
IS OUTBUF FULL
YES
NO
005 B1
NEWB
D2
NEUBUF 007A1
005 B1
NEUEING
F2
NUMMERNZ 008A1
005 A2
005 G2
YESHIGH
G2
FILL OUTPUTBUFFER INCREMENT ADDRESSES
H2
NUMMERNZ 008A1
ZUNU
J2
COMPUTE NUMBER OF RESIDUAL BYTES IN INPUTBUFFER
005 A2
005 A3
CANCELK
A3
INCREMENT ADDRESS TO NEXT CHAR TO TEST IN INPUTBUFFER
B3
BLANK ADDRESSED OUTPUTBUFFER AND RESET ERRORFLAG
C3
WAS CANCEL LAST CHAR IN INBUF
NO
YES
D3
NEUBUF 007A1
005 C1
FOLGTLF
E3
IS NEXT CHAR IN INBUF LF OR CR
YES
NO
F3
IS NEXT CHAR NEW LINE
YES
NO
005 G3
ZUTESTZ
G3
COMPUTE NUMBER OF BYTES TO TEST IN INPUTBUFFER
005 C1
005 A4
LFODCRF
A4
INCREMENT ADDRESS TO NEXT BYTE TO TEST
B4
END OF INBUF
NO
YES
C4
NEUBUF 007A1
005 G4
TESTLFCR
D4
NEXT CHAR LINEFEED
YES
NO
005 G3
E4
NEXT CHAR CR
YES
NO
005 G4
NLADREP
G4
INCREMENT ADDRESS TO NEXT BYTE TO TEST
H4
ARE ALL CHAR TESTED IN INBUF
NO
YES
J4
NEUBUF 007A1
005 A5
DELETEK
A5
MOVE TESTED BYTES TO OUTPUTBUFFER
B5
IS OUTBUF FULL
NO
YES
C5
NUMMERNZ 008A1
D5
INCREMENT ADDRESSES
009 H2
005 F5
LINEFETK
F5
MOVE TESTED BYTES TO OUTBUF AND INCREMENT ADDRESSES
G5
IF OUTBUF FULL GOTO NUMMERN
H5
MOVE REST OF TESTED BYTES TO NEW OUTAREA
J5
NUMMERNZ 008A1
006 J1

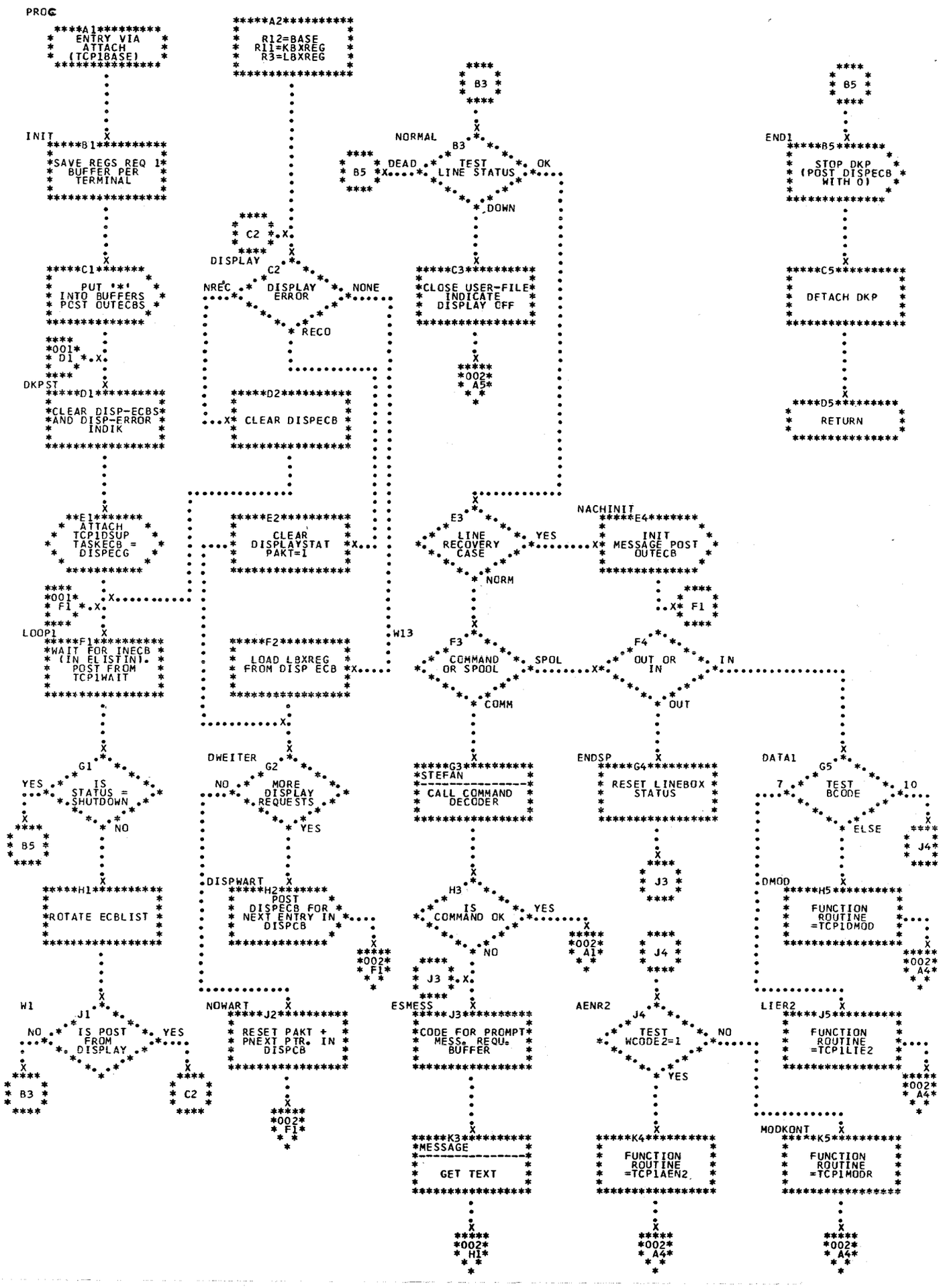

006 A1
SUBK
A1
MOVE BYTES TO OUTPUTBUFFER
B1
ARE ALL TESTED CHAR PROCESSED
YES
NO
C1
OUTBUF FULL
NO
YES
D1
NUMMERNZ 008A1
E1
MOVE REST OF TESTED BYTES TO OUTAREA
F1
MOVE CHAR '*' TO OUTBUF
G1
OUTBUF FULL
NO
YES
006 J1
APUVOLL
J1
ARE ALL TESTED CHAR IN INBUF
NO
YES
005 G3
H1
NUMMERNZ 008A1
K1
NEUBUF 007A1
005 B1
006 A2
ETXK
A2
MOVE TESTED BYTES TO OUTPUTBUFFERS
B2
NUMMERNZ 008A1
ERR1
C2
CHECK
WAIT AND TEST LAST WRITE FOR COMPLETION
006 D2
ENDAB
D2
NO ERRORCODE
006 E2
RELBHE
E2
ARE ERRORBUF FULL
NO
YES
E2
F2
SET CODE IN REG 11 TO '405'
RELBHE1
G2
RELBUF
RELEASE REST OF DYNAMIC BUFFERS
KEINRELB
H2
ARE ERRORBUF FULL
YES
NO
A3
J2
ARE ALL BUFFERS FULL
YES
NO
A3
K2
RELBUF
RELEASE REST OF ERRORBUF
A3
A3
VOLLZA
A3
RECBUFAD EQUAL ZERO
YES
NO
F3
B3
STORE NUMBER OF BYTES IN LAST ERRORBUF TO RECBUFAD
C3
IS ANY ERRORBUF FULL
YES
NO
D3
RELBUF
CLOSE100
E3
ADD 100 TO ERRORCODE
E2
F3
CLOSESEQ
F3
CLOSE
CLOSE FOR REREAD SPOOLDS
G3
GET RETURNCODE IN REG 15
H3
FREE MAIN STORAGE
J3
RETURN
006 A4
ENDEOT
A4
GET RETURNCODE 402
EODADR
C4
GET RETURNCODE 401
E2
006 E4
ENDE
E4
GET RETURNCODE 0
FRAGEWRI
F4
DATA WRITTEN ON SPOOLDS
NO
YES
E2
G4
CHECK
WAIT AND TEST LAST WRITE ON SPOOLDS
ABENDK
J4
GET RETURNCODE 406
006 J4

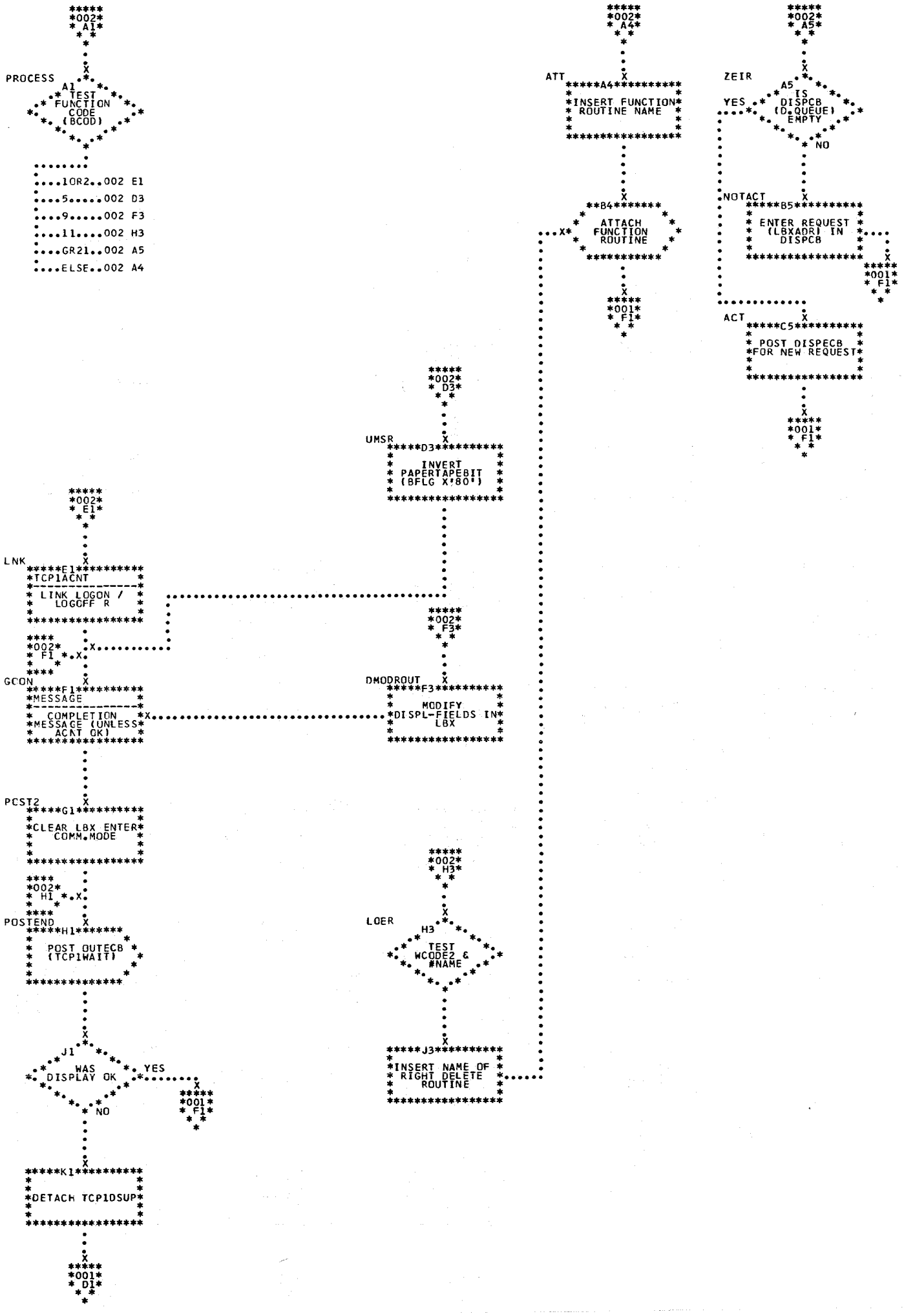

007 A1
NEUBUF
A1 RELBUF
RELEASE LAST INPUTBUFFER PROCESSED
B1 IS DECONLTT EQUAL ZERO
YES
NO
C1 DECREMENT DECONLTT
SENS
D1 TIMEOUT
NO
YES
E1 GET ADDRESS OF INPUTBUFFER
E3
A2
A2 TIMEOUT
NO
YES
H2
B2 GET NUMBER OF BYTES IN LAST BUFFER
C2 ARE CHAR IN BUFFER
YES
NO
G2
D2 INCREMENT TIMEOUT COUNTER
E2 CLEAR POSTCODE IN BUFFERECB AND SWITCH SENSO
A3
A3 IS CCITT NR 2
YES
NO
B3 TRANSLATE CHAR IN BUFFER TO EBCDIC- CHAR
007 C3
NACHTRAN
C3 RELOAD REGISTERS
NEUBFZZ
D3 BRANCH TO CALLING ROUTINE
E3
POSTWAIT
E3 REQBUF
GET A DYNAMIC BUFFER
F3 WAS REQBUF SUCCESSFULL
NO
YES
006 J4
TT5UEBER
A4 GET ADDRESS OF TRANSLATE TABLE
TT5UEBSE
B4 LETTERS
YES
NO
B4
C4 DIGITS
YES
NO
ZIUMSCHA
C5 GET ADDRESS OF TRANSLATE TABLE FOR DIGITS
F4
D4 TRANSLATE CHAR TO EBCDIC
BUUMSCHA
E4 GET ADDRESS OF TRANSLATE TABLE
F4
DELEINF
F4 MOVE DELETE INTO OUTBUF
FOLGBYT
G4 END OF INBUF
YES
NO
H4 INCREMENT ADDRESS TO NEXT CHAR
B4
BUFANADR
G5 GET ADDRESS OF FIRST BYTE IN BUFFER
H5 SAVE ADDRESS OF TRANSLATE TABLE
C3
NACHPOST
F1 SAVE ADDRESS OF INPUTBUFFER
G1 WAIT
WAIT FOR NEXT BUFFER
H1 CLEAR INDICATOR FOR LAST BYTE IN BUFFER
J1 SAVE THE ADDRESS OF THE NEXT BUFFER
K1 IS IT 5. TIMEOUT
YES
NO
009 J1
A2
G2
SENSOSE
G2 CLEAR SWITCH SENSO
H2
SENSNOT
H2 GET BUFFER ENDADDRESS
STBUEND
J2 SAVE BUFFER ENDADDRESS
K2 GET ADDRESS OF TRANSLATE TABLE
A3
G3 STORE ADDRESS OF BUFFER INTO LINEBOX, CLEAR POSTCODE
H3 POST
OUTECB

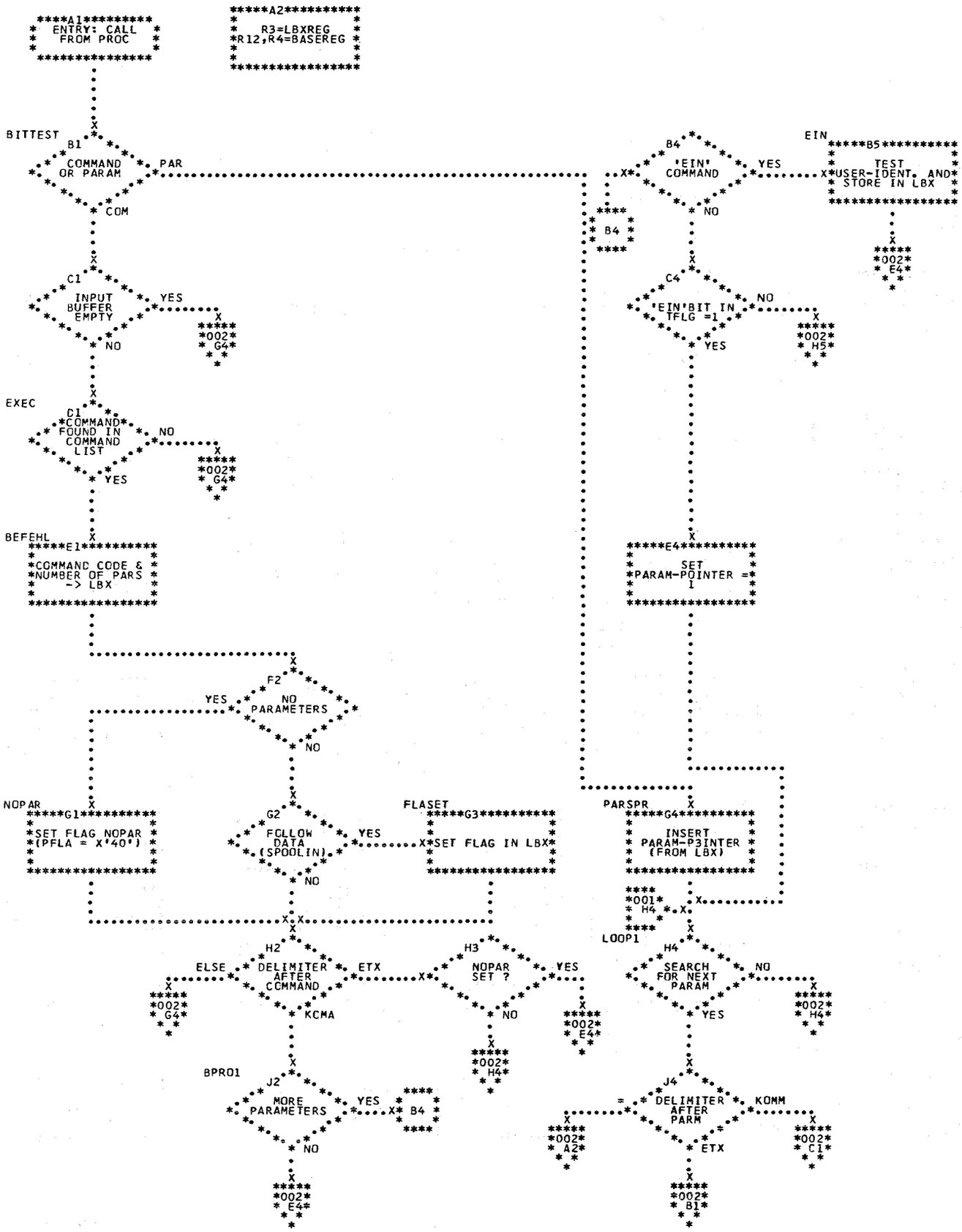


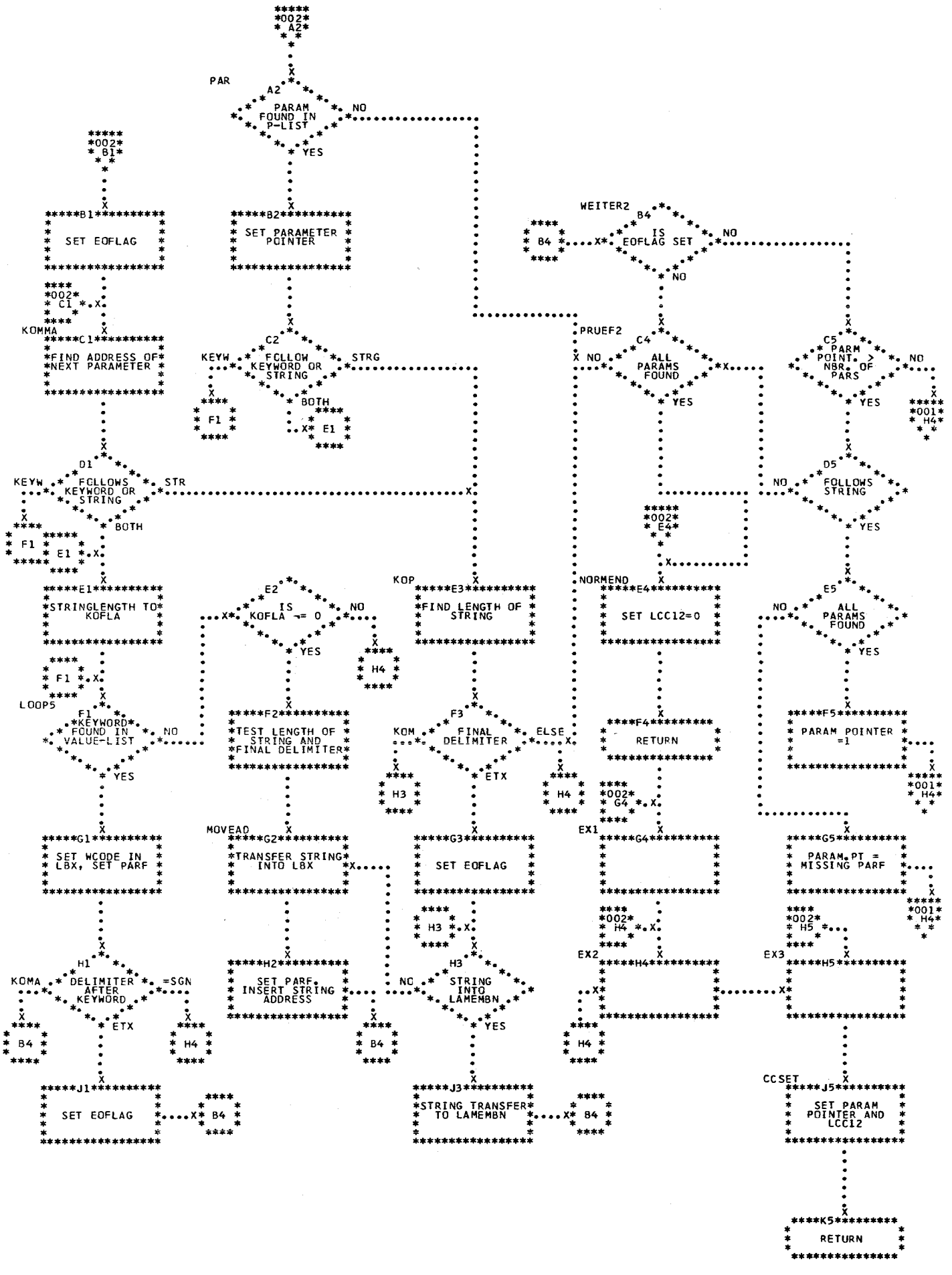
A2. BUFFERPOOL

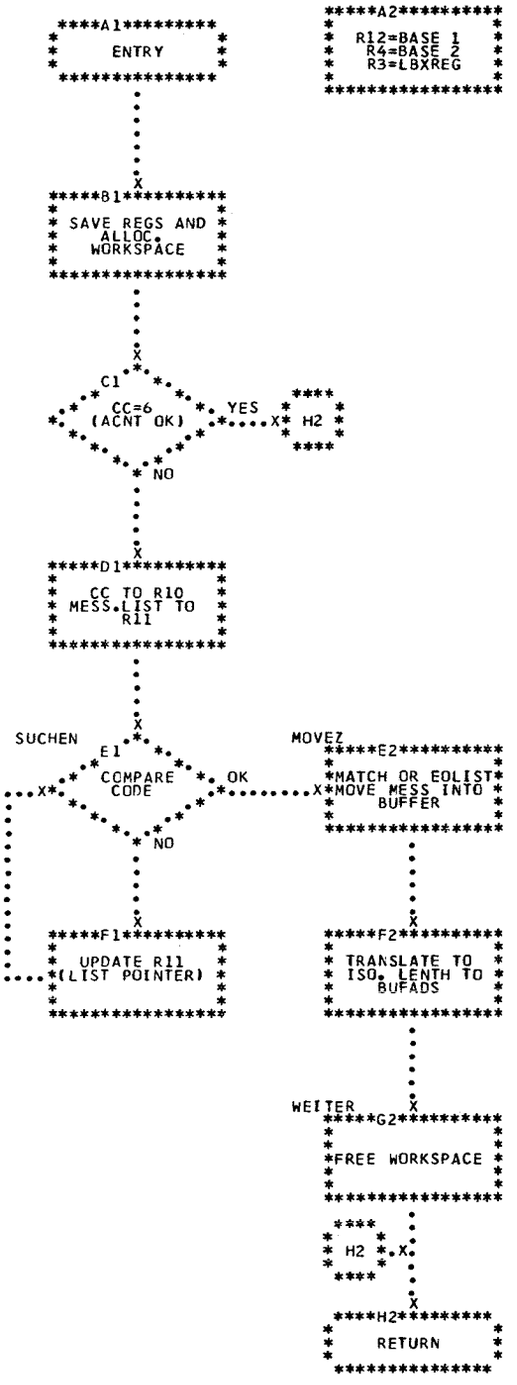

009
A1
TT5UEBS
A1
LETTERS FOLLOWING
YES
NO
BUUMSCH
A2
GET ADDRESS OF TRANSL.TABLE
A2
A3
GET BEGIN ADDRESS OF BUFFER
007
C3
DELETEB
B1
DIGIT
YES
NO
B2
FILL IN DELETE
FOLGBYTE
C2
END OF BUFFER
YES
NO
C1
TRANSLATE BYTE IN BUFFER
A1
A2
ZIUMSCH
E1
GET ADDRESS OF TRANSL.TABLE FOR DIGITS
009
E3
ABFRA
E3
CANCEL
NO
YES
004
J4
F3
CLEAR BUFFER AND FLAGBIT
009
G1
HAUPTPR1
G1
SET NUMBER OF BYTES EQUAL 0
002
E4
009
H2
TESTANZ
H2
IS OUTBUF FULL
NO
YES
006
J1
008
A1
G3
GET ADDR OF NEXT BYTE IN INBUF
H3
CLEAR CONTREGS
002
E4
009
J1
RECODE
J1
SET ERRORCODE IN R11 TO 403
006
E2

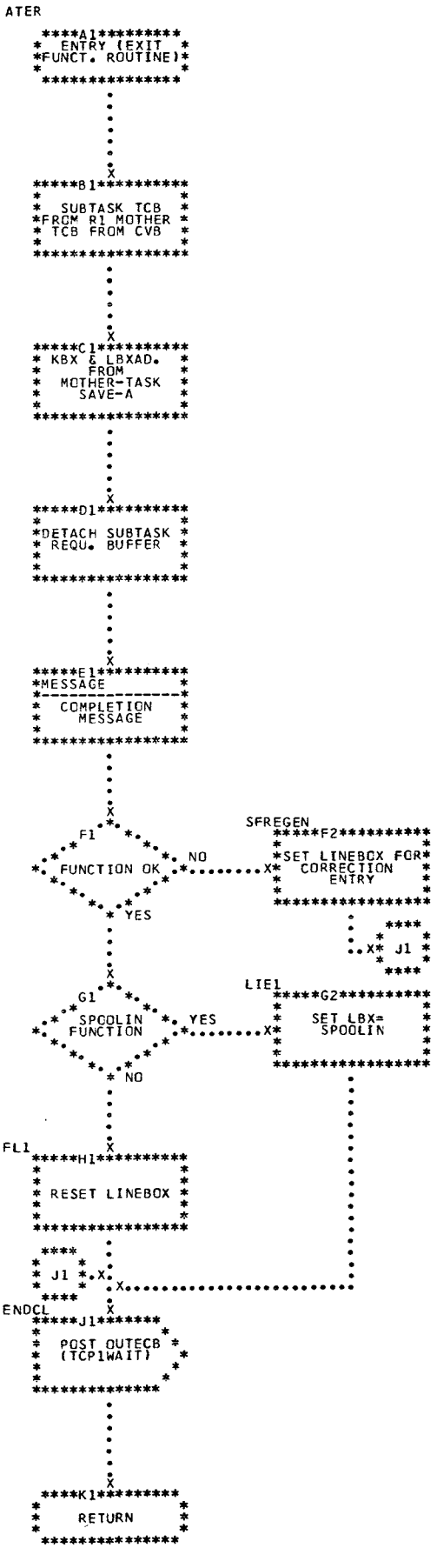

PROC
A1
ENTRY VIA ATTACH (TCP1BASE)
A2
R12=BASE R11=KBXREG R3=LBXREG
INIT
B1
SAVE REGS REQ 1 BUFFER PER TERMINAL
B3
NORMAL
DEAD LINE STATUS OK
DOWN
B5
END1
B5
STOP DKP (POST DISPECB WITH 0)
C1
PUT '*' INTO BUFFERS POST OUTECBS
C2
DISPLAY
NREC DISPLAY ERROR NONE
RECO
C3
CLOSE USER-FILE INDICATE DISPLAY OFF
C5
DETACH DKP
001 D1
DKPST
D1
CLEAR DISP-ECBS AND DISP-ERROR INDIK
D2
CLEAR DISPECB
D5
RETURN
E1
ATTACH TCP1DSUP TASKECB = DISPECG
E2
CLEAR DISPLAYSTAT PAKT=1
E3
LINE RECOVERY CASE
YES
NORM
NACHINIT
E4
INIT MESSAGE POST OUTECB
001 F1
LOOP1
F1
WAIT FOR INECB (IN ELISTIN). POST FROM TCP1WAIT
F2
LOAD LBXREG FROM DISP ECB
W13
F3
COMMAND OR SPOOL
SPOL
COMM
F4
OUT OR IN
IN
OUT
G1
IS STATUS = SHUTDOWN
YES
NO
B5
DWEITER
G2
MORE DISPLAY REQUESTS
NO
YES
G3
STEFAN
CALL COMMAND DECODER
ENDSP
G4
RESET LINEBOX STATUS
J3
DATA1
G5
TEST BCODE
7
10
ELSE
J4
H1
ROTATE ECBLIST
DISPWART
H2
POST DISPECB FOR NEXT ENTRY IN DISPCB
002 F1
H3
IS COMMAND OK
YES
NO
002 A1
J4
DMOD
H5
FUNCTION ROUTINE =TCP1DMOD
002 A4
W1
J1
IS POST FROM DISPLAY
NO
YES
B3
C2
NOWART
J2
RESET PAKT + PNEXT PTR. IN DISPCB
002 F1
ESMESS
J3
CODE FOR PROMPT MESS. REQU. BUFFER
AENR2
J4
TEST WCODE2=1
NO
YES
LIER2
J5
FUNCTION ROUTINE =TCP1LIE2
002 A4
K3
MESSAGE
GET TEXT
002 H1
K4
FUNCTION ROUTINE =TCP1AEN2
002 A4
MODKONT
K5
FUNCTION ROUTINE =TCP1MODR
002 A4

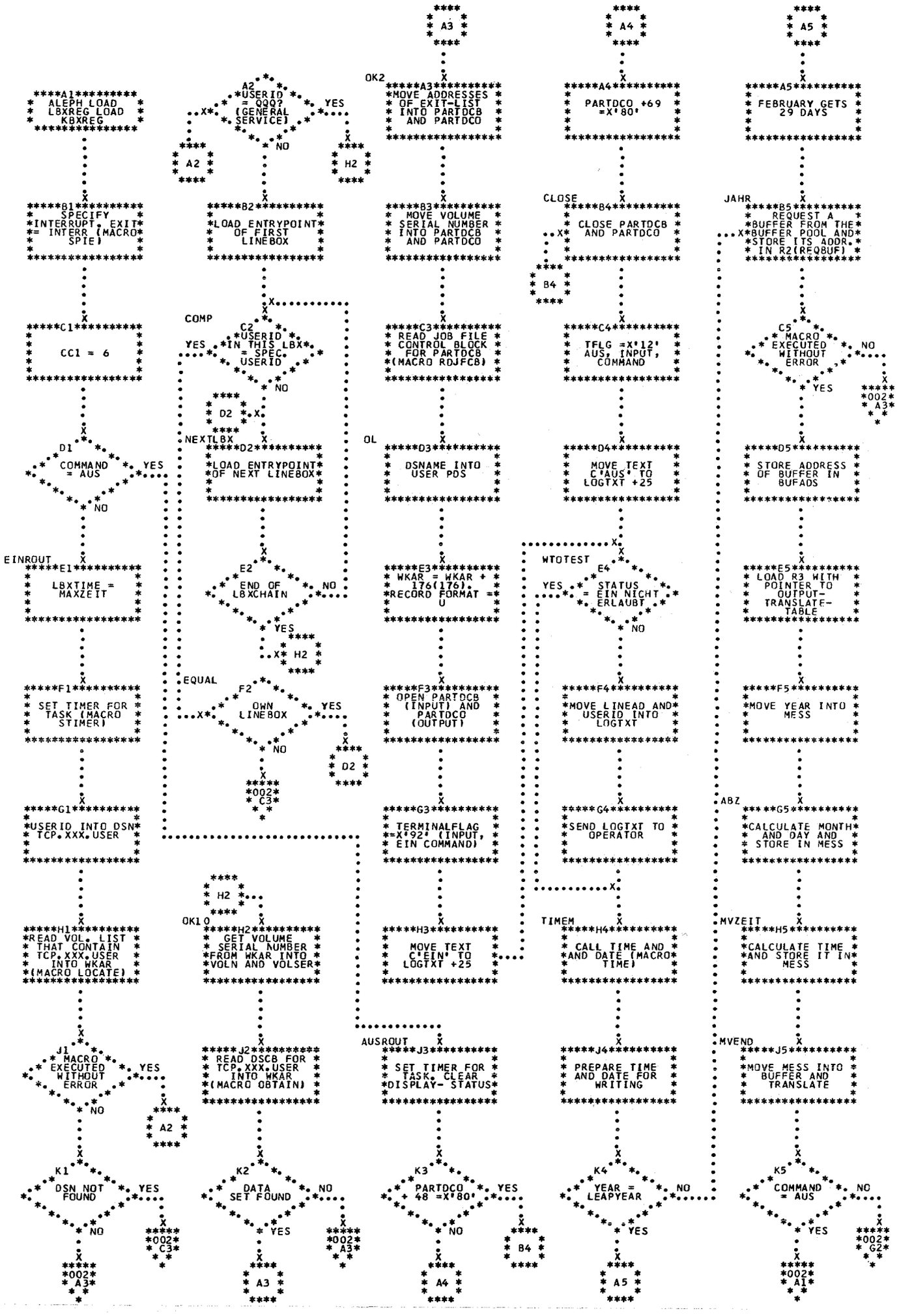

002 A1
PROCESS
A1 TEST FUNCTION CODE (BCOD)
1OR2 002 E1
5 002 D3
9 002 F3
11 002 H3
GR21 002 A5
ELSE 002 A4
002 A4
ATT
A4 INSERT FUNCTION ROUTINE NAME
B4 ATTACH FUNCTION ROUTINE
001 F1
002 A5
ZEIR
A5 IS DISPCB (D.QUEUE) EMPTY
YES
NO
NOTACT
B5 ENTER REQUEST (LBXADR) IN DISPCB
001 F1
ACT
C5 POST DISPECB FOR NEW REQUEST
001 F1
002 D3
UMSR
D3 INVERT PAPERTAPEBIT (BFLG X'80')
002 E1
LNK
E1 TCP1ACNT LINK LOGON / LOGOFF R
002 F1
GCON
F1 MESSAGE COMPLETION MESSAGE (UNLESS ACNT OK)
002 F3
DMODROUT
F3 MODIFY DISPL-FIELDS IN LBX
PCST2
G1 CLEAR LBX ENTER COMM.MODE
002 H1
POSTEND
H1 POST OUTECB (TCP1WAIT)
J1 WAS DISPLAY OK
YES
NO
001 F1
K1 DETACH TCP1DSUP
001 D1
002 H3
LOER
H3 TEST WCODE2 & #NAME
J3 INSERT NAME OF RIGHT DELETE ROUTINE

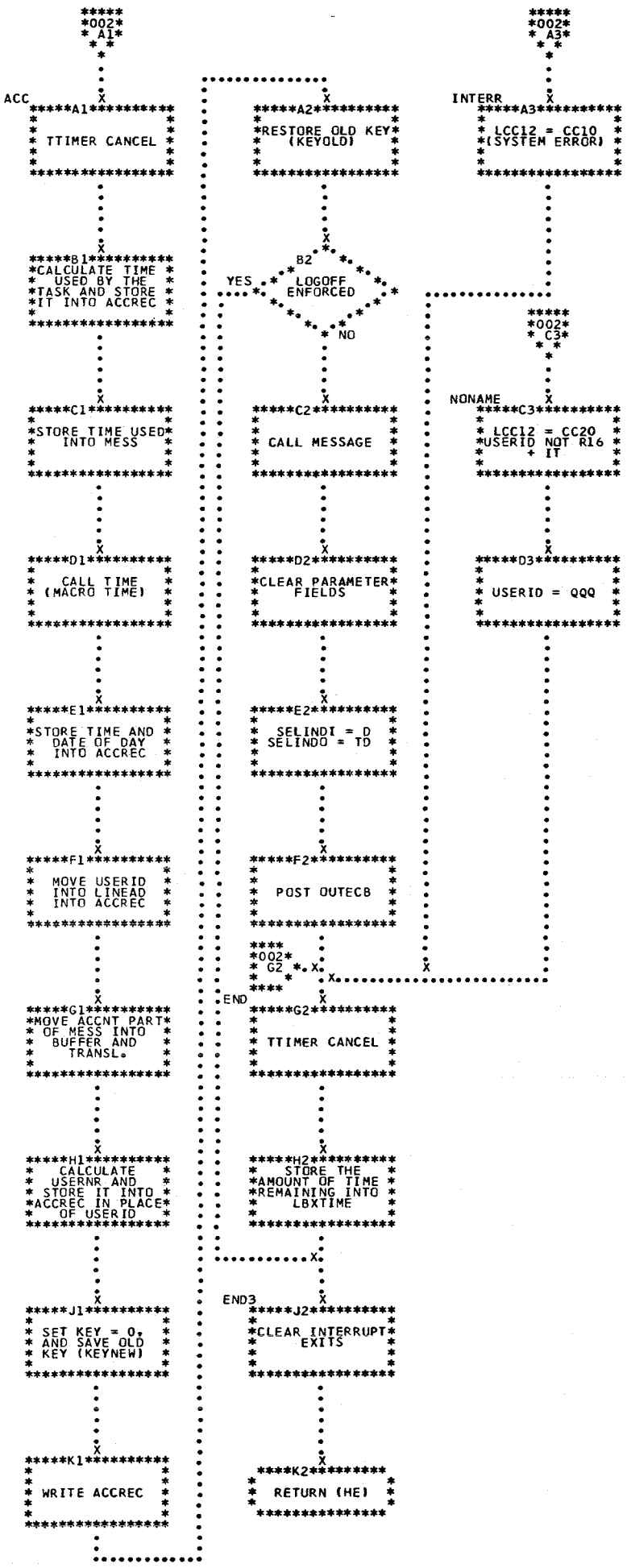

A1
ENTRY: CALL FROM PROC
A2
R3=LBXREG
R12,R4=BASEREG
BITTEST
B1
COMMAND OR PARAM
PAR
COM
C1
INPUT BUFFER EMPTY
YES
NO
002
G4
EXEC
C1
COMMAND FOUND IN COMMAND LIST
NO
YES
002
G4
BEFEHL
E1
COMMAND CODE & NUMBER OF PARS -> LBX
F2
NO PARAMETERS
YES
NO
NOPAR
G1
SET FLAG NOPAR (PFLA = X'40')
G2
FOLLOW DATA (SPOOLIN)
YES
NO
FLASET
G3
SET FLAG IN LBX
H2
DELIMITER AFTER COMMAND
ELSE
ETX
KOMA
002
G4
H3
NOPAR SET ?
YES
NO
002
E4
002
H4
BPRO1
J2
MORE PARAMETERS
YES
NO
B4
002
E4
B4
'EIN' COMMAND
YES
NO
EIN
B5
TEST USER-IDENT. AND STORE IN LBX
002
E4
C4
'EIN'BIT IN TFLG =1
NO
YES
002
H5
E4
SET PARAM-POINTER = 1
PARSPR
G4
INSERT PARAM-P3INTER (FROM LBX)
001
H4
LOOP1
H4
SEARCH FOR NEXT PARAM
NO
YES
002
H4
J4
DELIMITER AFTER PARM
KOMM
ETX
002
A2
002
C1
002
B1

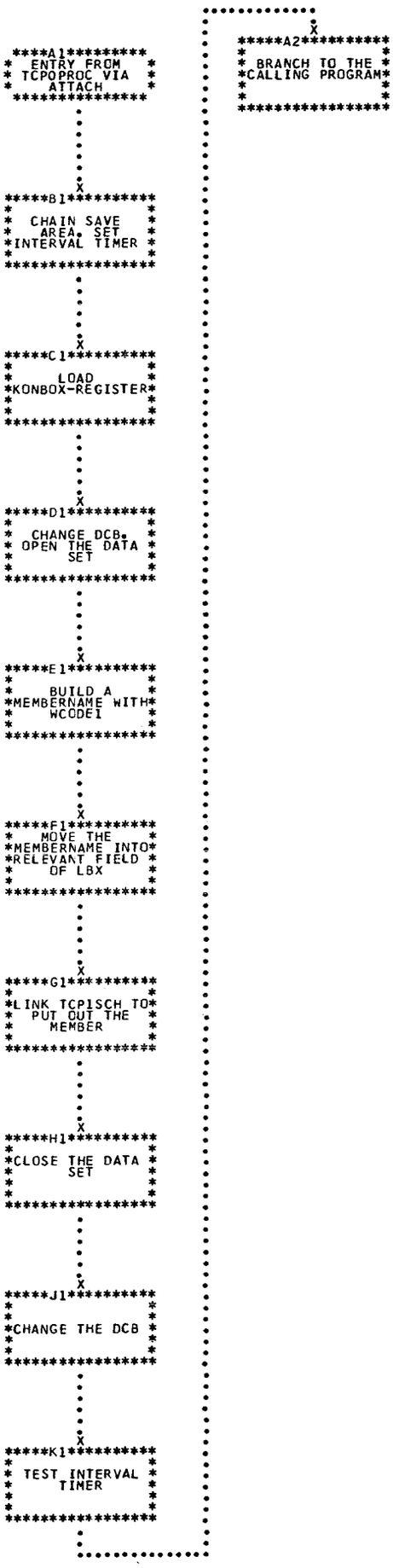

002
A2
PAR
A2
PARAM FOUND IN P-LIST
NO
YES
002
B1
B1
SET EOFLAG
B2
SET PARAMETER POINTER
002
C1
KOMMA
C1
FIND ADDRESS OF NEXT PARAMETER
C2
KEYW
FOLLOW KEYWORD OR STRING
STRG
F1
BOTH
E1
D1
KEYW
FOLLOWS KEYWORD OR STRING
STR
BOTH
F1
E1
E1
STRINGLENGTH TO KOFLA
E2
IS KOFLA ¬= 0
NO
YES
H4
LOOP5
F1
KEYWORD FOUND IN VALUE-LIST
NO
YES
F2
TEST LENGTH OF STRING AND FINAL DELIMITER
KOP
E3
FIND LENGTH OF STRING
F3
KOM
FINAL DELIMITER
ELSE
ETX
H3
H4
WEITER2
B4
IS EOFLAG SET
NO
B4
NO
PRUEF2
C4
ALL PARAMS FOUND
NO
YES
002
E4
NORMEND
E4
SET LCC12=0
F4
RETURN
002
G4
EX1
C5
PARM POINT. > NBR. OF PARS
NO
YES
001
H4
D5
NO
FOLLOWS STRING
YES
E5
NO
ALL PARAMS FOUND
YES
F5
PARAM POINTER =1
001
H4
G1
SET WCODE IN L8X, SET PARF
MOVEAD
G2
TRANSFER STRING INTO L8X
G3
SET EOFLAG
002
H3
G4
002
H4
EX2
G5
PARAM.PT = MISSING PARF
001
H4
002
H5
EX3
H1
KOMA
DELIMITER AFTER KEYWORD
=SGN
ETX
B4
H4
H2
SET PARF. INSERT STRING ADDRESS
B4
H3
NO
STRING INTO LAMEMBN
YES
H4
H4
H5
J1
SET EOFLAG
B4
J3
STRING TRANSFER TO LAMEMBN
B4
CCSET
J5
SET PARAM POINTER AND LCC12
K5
RETURN

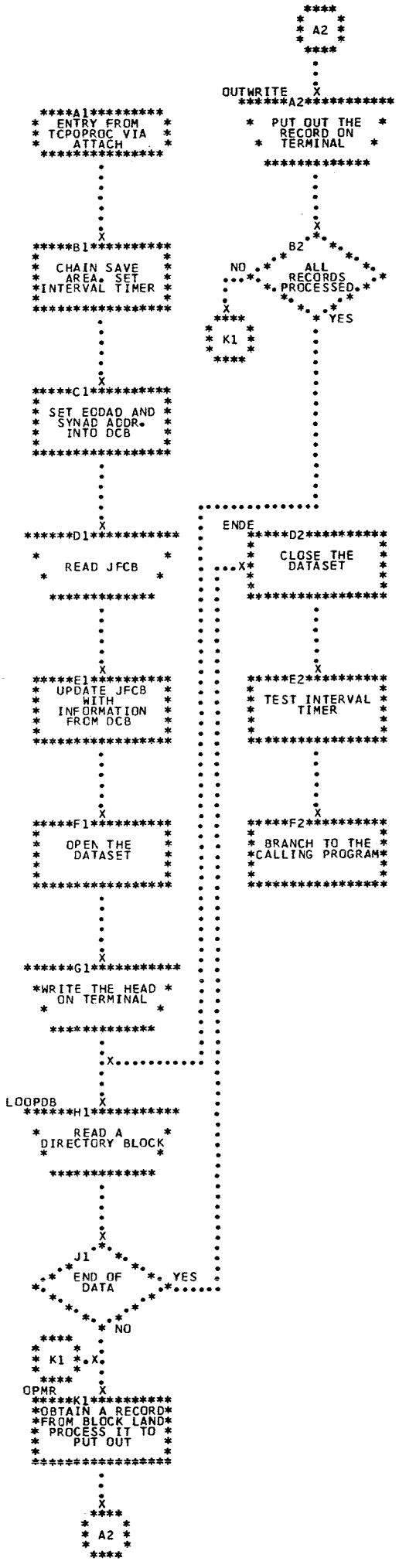

A1
ENTRY
A2
R12=BASE 1
R4=BASE 2
R3=LBXREG
B1
SAVE REGS AND ALLOC. WORKSPACE
C1
CC=6 (ACNT OK)
YES
H2
NO
D1
CC TO R10
MESS.LIST TO R11
SUCHEN
E1
COMPARE CODE
OK
NO
MOVEZ
E2
MATCH OR EOLIST
MOVE MESS INTO BUFFER
F1
UPDATE R11 (LIST POINTER)
F2
TRANSLATE TO ISO. LENTH TO BUFADS
WEITER
G2
FREE WORKSPACE
H2
RETURN

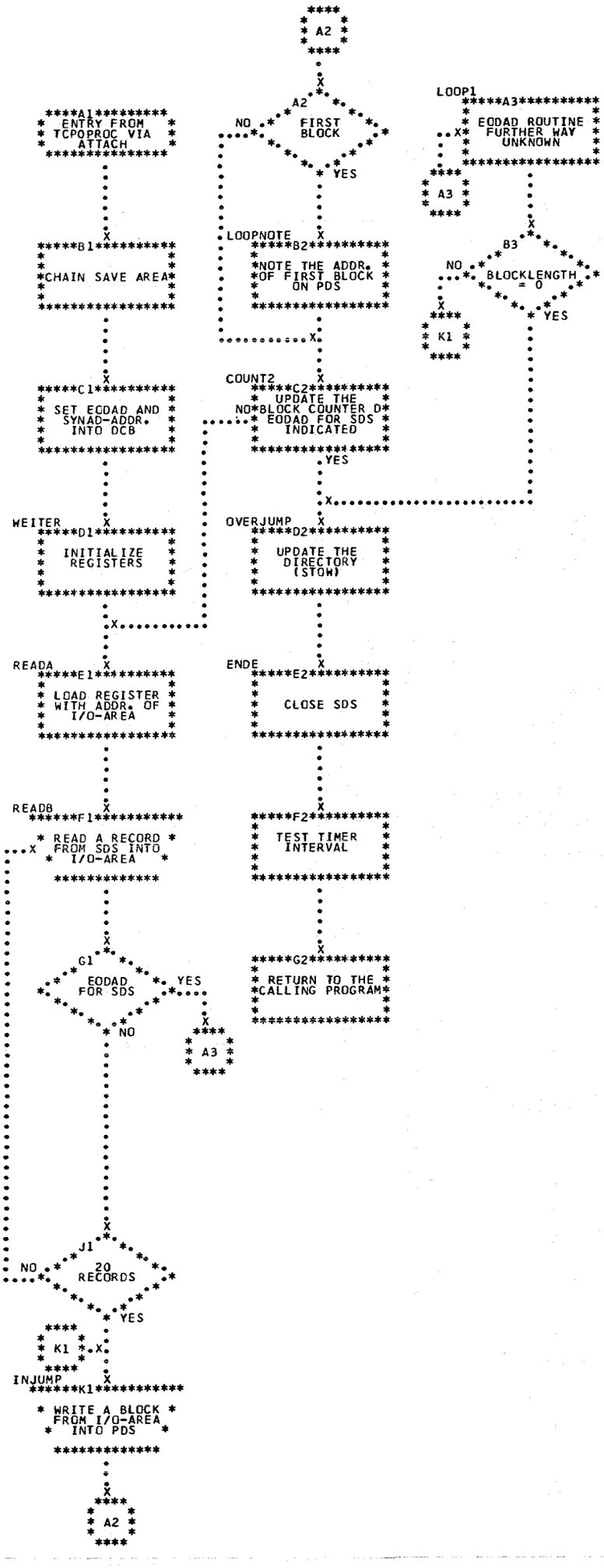

ATER
A1
ENTRY (EXIT FUNCT. ROUTINE)
B1
SUBTASK TCB FROM R1 MOTHER TCB FROM CVB
C1
KBX & LBXAD. FROM MOTHER-TASK SAVE-A
D1
DETACH SUBTASK REQU. BUFFER
E1
MESSAGE
COMPLETION MESSAGE
F1
FUNCTION OK
NO
YES
SFREGEN
F2
SET LINEBOX FOR CORRECTION ENTRY
J1
G1
SPOOLIN FUNCTION
YES
NO
LIE1
G2
SET LBX= SPOOLIN
FL1
H1
RESET LINEBOX
J1
ENDCL
J1
POST OUTECB (TCP1WAIT)
K1
RETURN

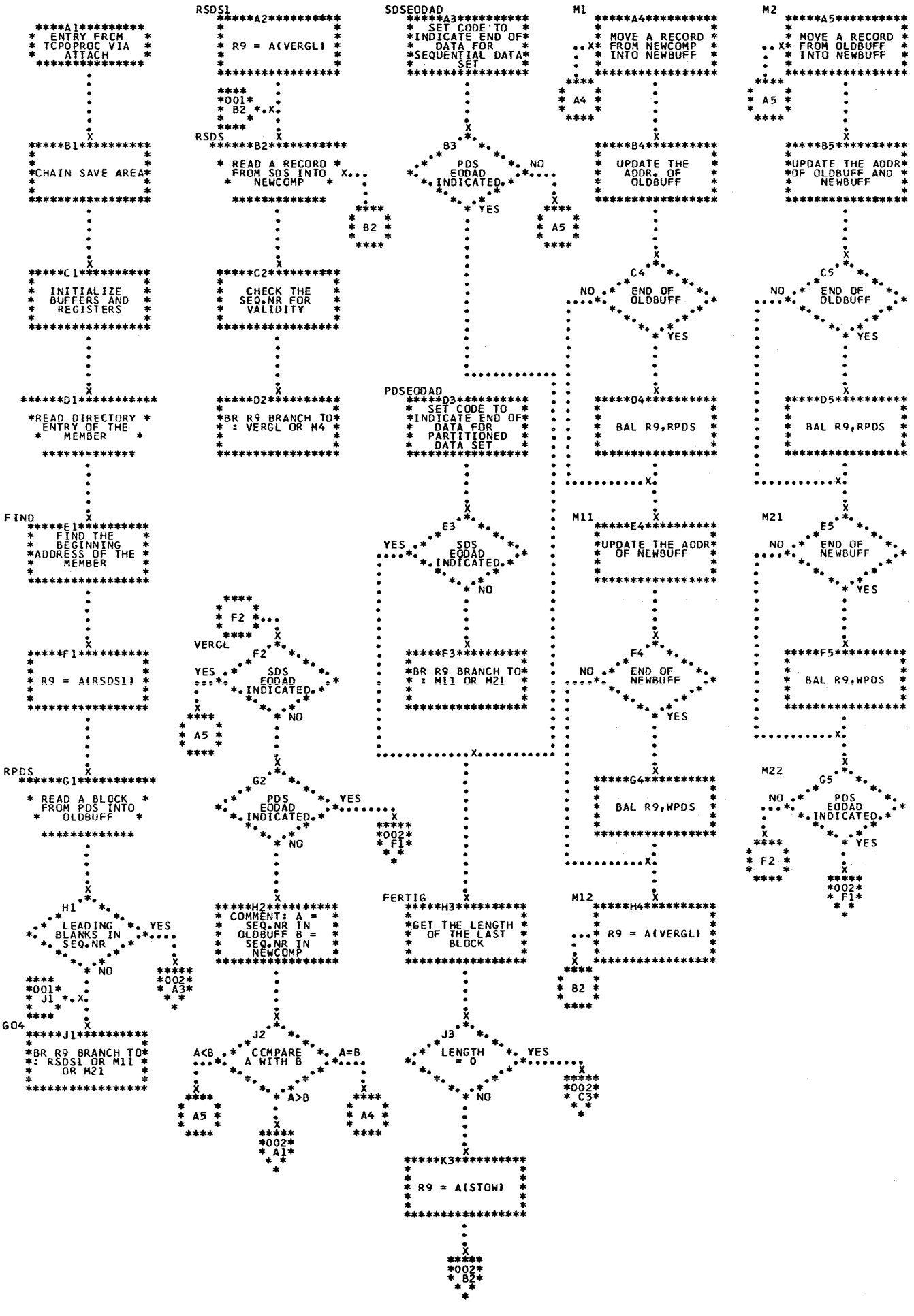

A1
ALEPH LOAD
LBXREG LOAD
KBXREG
B1
SPECIFY
INTERRUPT EXIT
= INTERR (MACRO
SPIE)
C1
CC1 = 6
D1
COMMAND = AUS
YES
NO
EINROUT
E1
LBXTIME = MAXZEIT
F1
SET TIMER FOR TASK (MACRO STIMER)
G1
USERID INTO DSN
TCP.XXX.USER
H1
READ VOL. LIST THAT CONTAIN TCP.XXX.USER INTO WKAR (MACRO LOCATE)
J1
MACRO EXECUTED WITHOUT ERROR
YES
NO
A2
K1
DSN NOT FOUND
YES
NO
002 C3*
002 A3*
A2
USERID = QQQ? (GENERAL SERVICE)
YES
NO
H2
B2
LOAD ENTRYPOINT OF FIRST LINEBOX
COMP
C2
USERID = SPEC. USERID
YES
NO
D2
NEXTLBX
D2
LOAD ENTRYPOINT OF NEXT LINEBOX
E2
END OF LBXCHAIN
NO
YES
H2
EQUAL
F2
OWN LINEBOX
YES
NO
D2
002 C3*
H2
OK1
H2
GET VOLUME SERIAL NUMBER FROM WKAR INTO VOLN AND VOLSER
J2
READ DSCB FOR TCP.XXX.USER INTO WKAR (MACRO OBTAIN)
K2
DATA SET FOUND
NO
YES
002 A3*
A3
OK2
A3
MOVE ADDRESSES OF EXIT-LIST INTO PARTDCB AND PARTDCO
B3
MOVE VOLUME SERIAL NUMBER INTO PARTDCB AND PARTDCO
C3
READ JOB FILE CONTROL BLOCK FOR PARTDCB (MACRO RDJFCB)
OL
D3
DSNAME INTO USER PDS
E3
WKAR = WKAR + 176(176). *RECORD FORMAT = U
F3
OPEN PARTDCB (INPUT) AND PARTDCO (OUTPUT)
G3
TERMINALFLAG =X'92' (INPUT, EIN COMMAND)
H3
MOVE TEXT C'EIN' TO LOGTXT +25
AUSROUT
J3
SET TIMER FOR TASK, CLEAR *DISPLAY- STATUS
K3
PARTDCO + 48 =X'80'
YES
NO
B4
A4
A4
PARTDCO +69 =X'80'
CLOSE
B4
CLOSE PARTDCB AND PARTDCO
C4
TFLG =X'12' AUS, INPUT, COMMAND
D4
MOVE TEXT C'AUS' TO LOGTXT +25
WTOTEST
E4
STATUS = EIN NICHT ERLAUBT
YES
NO
F4
MOVE LINEAD AND USERID INTO LOGTXT
G4
SEND LOGTXT TO OPERATOR
TIMEM
H4
CALL TIME AND AND DATE (MACRO TIME)
J4
PREPARE TIME AND DATE FOR WRITING
K4
YEAR = LEAPYEAR
NO
YES
A5
A5
FEBRUARY GETS 29 DAYS
JAHR
B5
REQUEST A BUFFER FROM THE BUFFER POOL AND STORE ITS ADDR. IN R2(REQBUF)
C5
MACRO EXECUTED WITHOUT ERROR
NO
YES
002 A3*
D5
STORE ADDRESS OF BUFFER IN BUFADS
E5
LOAD R3 WITH POINTER TO OUTPUT-TRANSLATE-TABLE
F5
MOVE YEAR INTO MESS
ABZ
G5
CALCULATE MONTH AND DAY AND STORE IN MESS
MVZEIT
H5
CALCULATE TIME AND STORE IT IN MESS
MVEND
J5
MOVE MESS INTO BUFFER AND TRANSLATE
K5
COMMAND = AUS
NO
YES
002 G2*
002 A1*

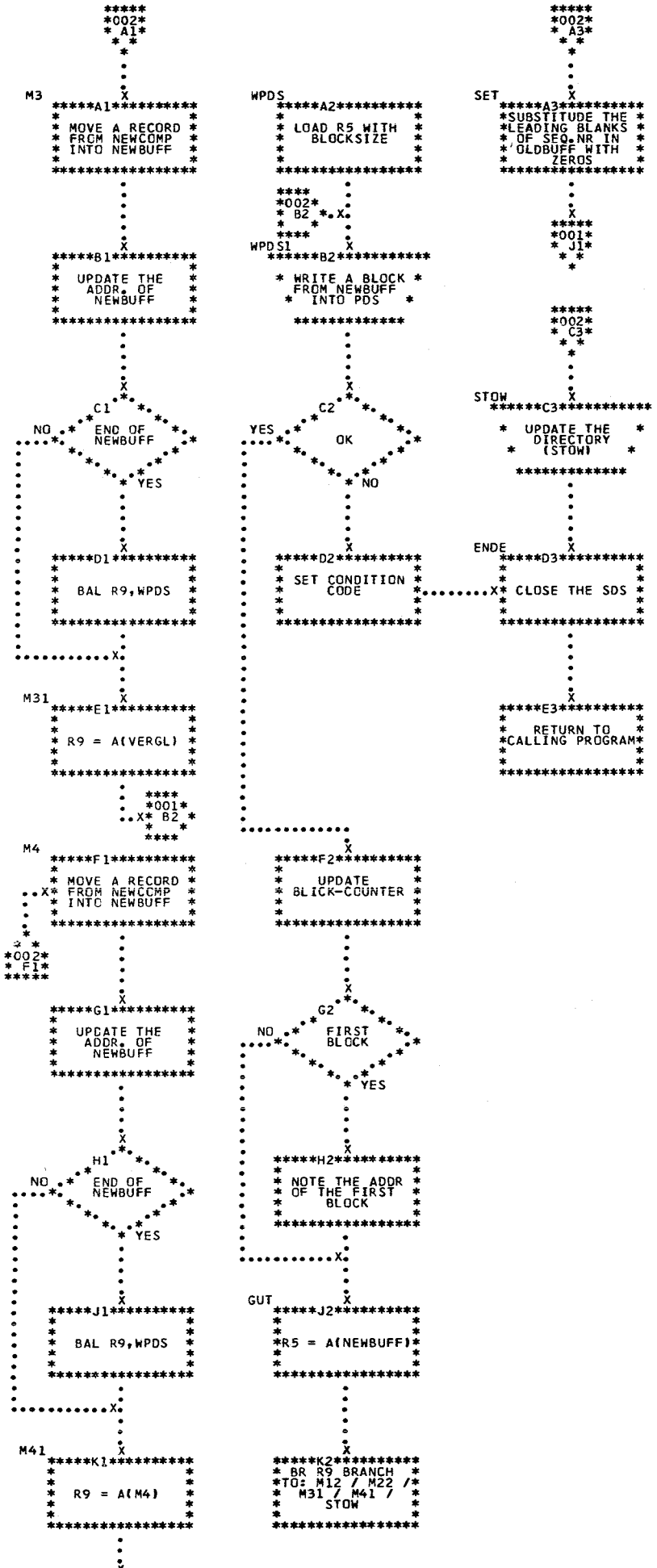

002 A1
ACC
A1
TTIMER CANCEL
B1
CALCULATE TIME USED BY THE TASK AND STORE IT INTO ACCREC
C1
STORE TIME USED INTO MESS
D1
CALL TIME (MACRO TIME)
E1
STORE TIME AND DATE OF DAY INTO ACCREC
F1
MOVE USERID INTO LINEAD INTO ACCREC
G1
MOVE ACCNT PART OF MESS INTO BUFFER AND TRANSL.
H1
CALCULATE USERNR AND STORE IT INTO ACCREC IN PLACE OF USERID
J1
SET KEY = 0, AND SAVE OLD KEY (KEYNEW)
K1
WRITE ACCREC
A2
RESTORE OLD KEY (KEYOLD)
B2
YES
LOGOFF ENFORCED
NO
C2
CALL MESSAGE
D2
CLEAR PARAMETER FIELDS
E2
SELINDI = D
SELINDO = TD
F2
POST OUTECB
002 G2
END
G2
TTIMER CANCEL
H2
STORE THE AMOUNT OF TIME REMAINING INTO LBXTIME
END3
J2
CLEAR INTERRUPT EXITS
K2
RETURN (HE)
002 A3
INTERR
A3
LCC12 = CC10 (SYSTEM ERROR)
002 C3
NONAME
C3
LCC12 = CC20 USERID NOT R16 + IT
D3
USERID = QQQ

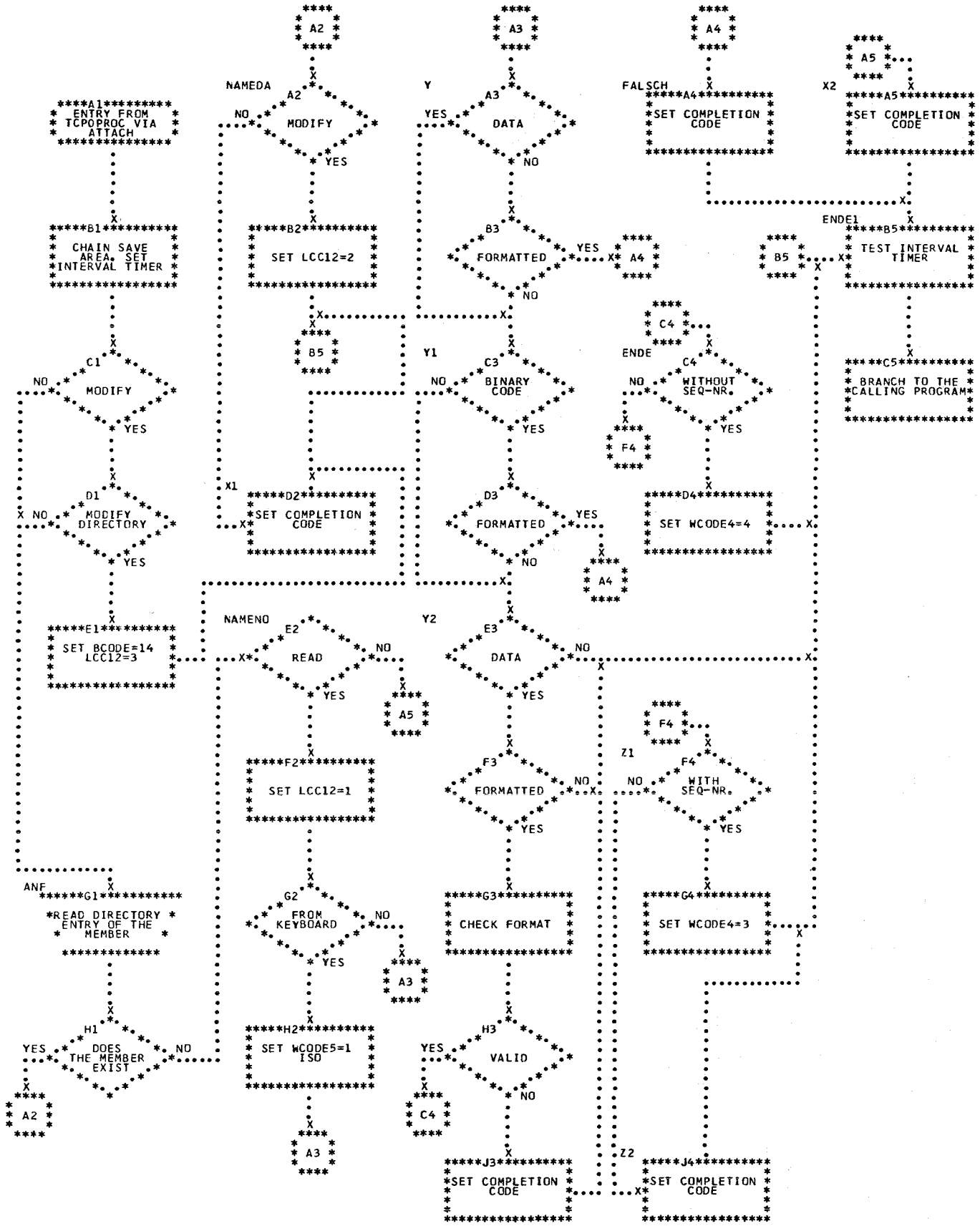

A1
ENTRY FROM TCPOPROC VIA ATTACH
B1
CHAIN SAVE AREA. SET INTERVAL TIMER
C1
LOAD KONBOX-REGISTER
D1
CHANGE DCB. OPEN THE DATA SET
E1
BUILD A MEMBERNAME WITH WCODE1
F1
MOVE THE MEMBERNAME INTO RELEVANT FIELD OF LBX
G1
LINK TCP1SCH TO PUT OUT THE MEMBER
H1
CLOSE THE DATA SET
J1
CHANGE THE DCB
K1
TEST INTERVAL TIMER
A2
BRANCH TO THE CALLING PROGRAM

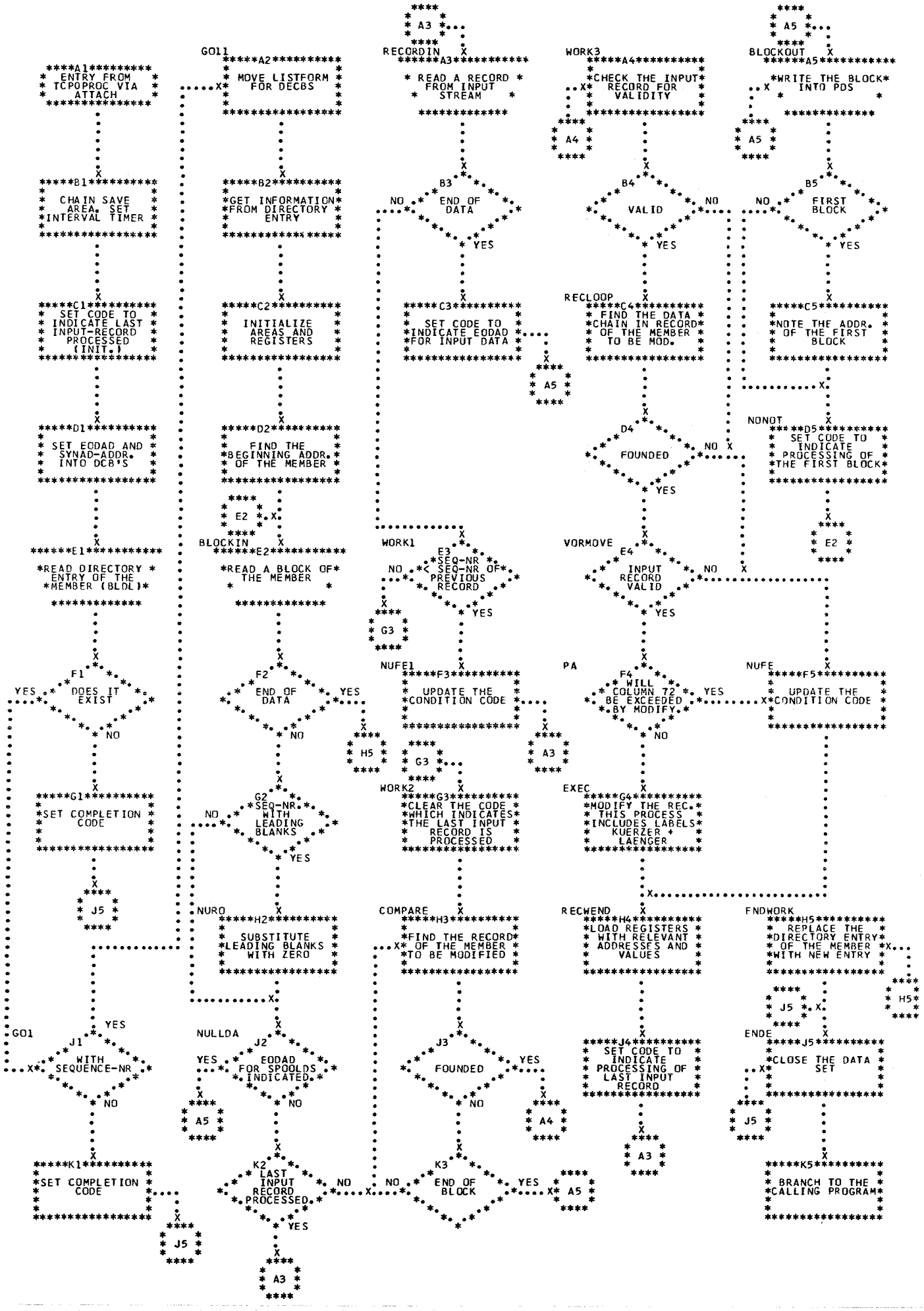

A2
OUTWRITE
A2
PUT OUT THE RECORD ON TERMINAL
A1
ENTRY FROM TCPOPROC VIA ATTACH
B2
NO
ALL RECORDS PROCESSED
YES
K1
B1
CHAIN SAVE AREA, SET INTERVAL TIMER
C1
SET EODAD AND SYNAD ADDR. INTO DCB
ENDE
D2
CLOSE THE DATASET
D1
READ JFCB
E2
TEST INTERVAL TIMER
E1
UPDATE JFCB WITH INFORMATION FROM DCB
F2
BRANCH TO THE CALLING PROGRAM
F1
OPEN THE DATASET
G1
WRITE THE HEAD ON TERMINAL
LOOPDB
H1
READ A DIRECTORY BLOCK
J1
END OF DATA
YES
NO
K1
OPMR
K1
OBTAIN A RECORD FROM BLOCK LAND PROCESS IT TO PUT OUT
A2

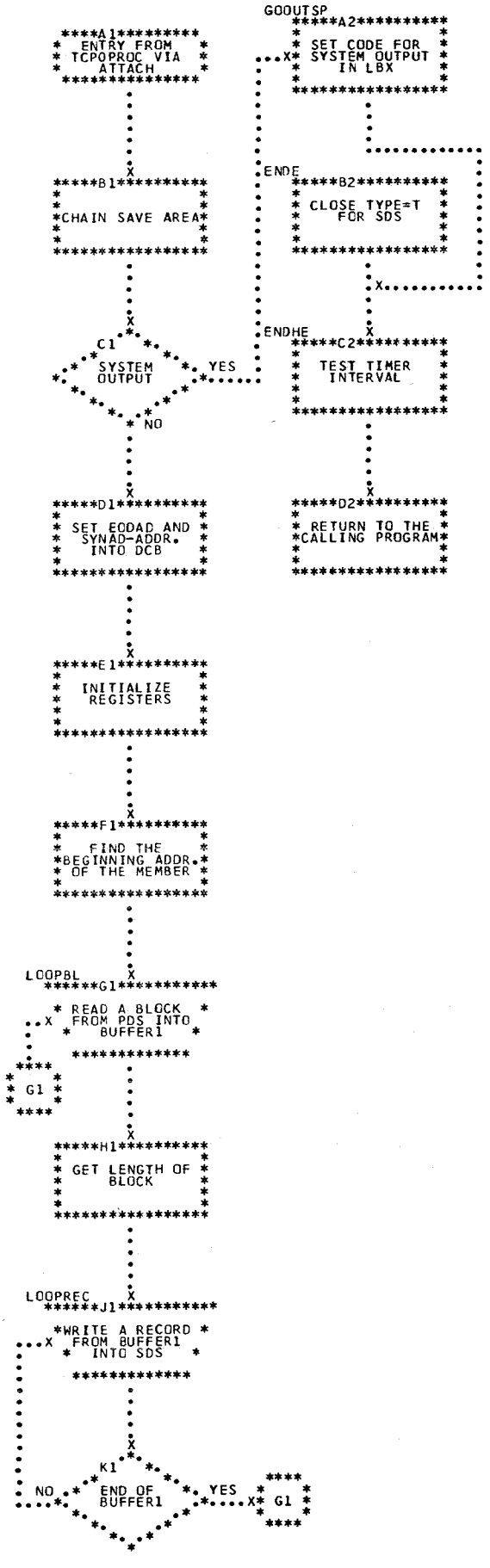

A2
FIRST BLOCK
NO
YES
A1
ENTRY FROM TCPOPROC VIA ATTACH
B1
CHAIN SAVE AREA
C1
SET EODAD AND SYNAD-ADDR. INTO DCB
D1
INITIALIZE REGISTERS
E1
LOAD REGISTER WITH ADDR. OF I/O-AREA
F1
READ A RECORD FROM SDS INTO I/O-AREA
G1
EODAD FOR SDS
J1
20 RECORDS
K1
WRITE A BLOCK FROM I/O-AREA INTO PDS
LOOPNOTE
B2
NOTE THE ADDR. OF FIRST BLOCK ON PDS
COUNT2
C2
UPDATE THE BLOCK COUNTER OF EODAD FOR SDS INDICATED
OVERJUMP
D2
UPDATE THE DIRECTORY (STOW)
ENDE
E2
CLOSE SDS
F2
TEST TIMER INTERVAL
G2
RETURN TO THE CALLING PROGRAM
LOOP1
A3
EODAD ROUTINE FURTHER WAY UNKNOWN
B3
BLOCKLENGTH = 0
INJUMP

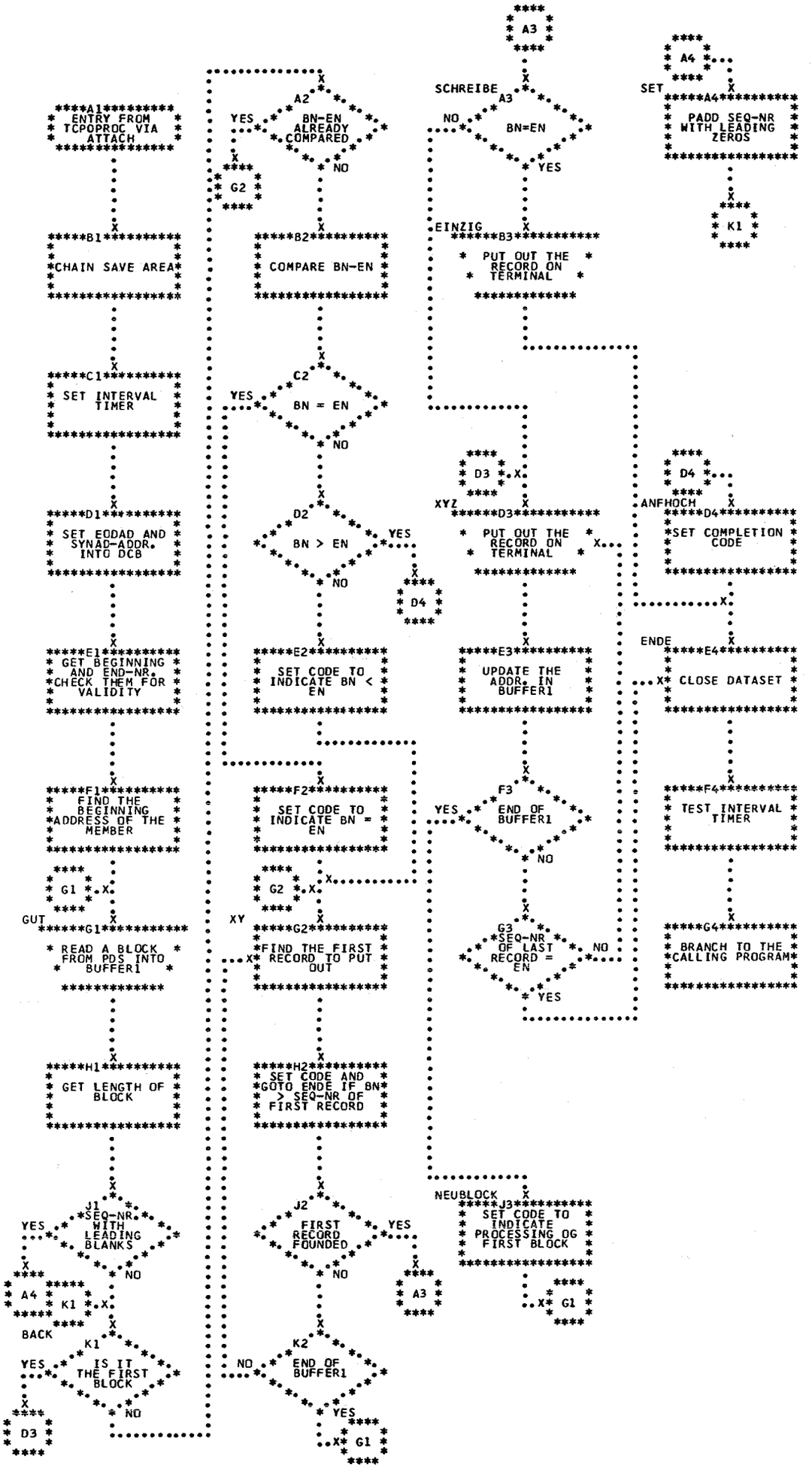

A1
ENTRY FRCM TCPOPROC VIA ATTACH
B1
CHAIN SAVE AREA
C1
INITIALIZE BUFFERS AND REGISTERS
D1
READ DIRECTORY ENTRY OF THE MEMBER
FIND
E1
FIND THE BEGINNING ADDRESS OF THE MEMBER
F1
R9 = A(RSDS1)
RPDS
G1
READ A BLOCK FROM PDS INTO OLDBUFF
H1
LEADING BLANKS IN SEQ.NR
YES
NO
001
J1
GO4
J1
BR R9 BRANCH TO: RSDS1 OR M11 OR M21
RSDS1
A2
R9 = A(VERGL)
001
B2
RSDS
B2
READ A RECORD FROM SDS INTO NEWCOMP
B2
C2
CHECK THE SEQ.NR FOR VALIDITY
D2
BR R9 BRANCH TO: VERGL OR M4
VERGL
F2
SDS EODAD INDICATED
YES
NO
A5
G2
PDS EODAD INDICATED
YES
NO
002
F1
H2
COMMENT: A = SEQ.NR IN OLDBUFF B = SEQ.NR IN NEWCOMP
J2
COMPARE A WITH B
A<B
A=B
A>B
A5
A4
002
A1
SDSEODAD
A3
SET CODE TO INDICATE END OF DATA FOR SEQUENTIAL DATA SET
B3
PDS EODAD INDICATED
NO
YES
A5
PDSEODAD
D3
SET CODE TO INDICATE END OF DATA FOR PARTITIONED DATA SET
E3
SDS EODAD INDICATED
YES
NO
F3
BR R9 BRANCH TO: M11 OR M21
FERTIG
H3
GET THE LENGTH OF THE LAST BLOCK
J3
LENGTH = 0
YES
NO
002
C3
K3
R9 = A(STOW)
002
B2
M1
A4
MOVE A RECORD FROM NEWCOMP INTO NEWBUFF
B4
UPDATE THE ADDR. OF OLDBUFF
C4
END OF OLDBUFF
NO
YES
D4
BAL R9,RPDS
M11
E4
UPDATE THE ADDR OF NEWBUFF
F4
END OF NEWBUFF
NO
YES
G4
BAL R9,WPDS
M12
H4
R9 = A(VERGL)
B2
M2
A5
MOVE A RECORD FROM OLDBUFF INTO NEWBUFF
B5
UPDATE THE ADDR OF OLDBUFF AND NEWBUFF
C5
END OF OLDBUFF
NO
YES
D5
BAL R9,RPDS
M21
E5
END OF NEWBUFF
NO
YES
F5
BAL R9,WPDS
M22
G5
PDS EODAD INDICATED
NO
YES
F2
F1


002 A1
M3
A1
MOVE A RECORD FROM NEWCOMP INTO NEWBUFF
B1
UPDATE THE ADDR. OF NEWBUFF
C1
END OF NEWBUFF
NO
YES
D1
BAL R9,WPDS
M31
E1
R9 = A(VERGL)
001 B2
M4
F1
MOVE A RECORD FROM NEWCOMP INTO NEWBUFF
002 F1
G1
UPDATE THE ADDR. OF NEWBUFF
H1
END OF NEWBUFF
NO
YES
J1
BAL R9,WPDS
M41
K1
R9 = A(M4)
001 B2
WPDS
A2
LOAD R5 WITH BLOCKSIZE
002 B2
WPDS1
B2
WRITE A BLOCK FROM NEWBUFF INTO PDS
C2
OK
YES
NO
D2
SET CONDITION CODE
F2
UPDATE BLICK-COUNTER
G2
FIRST BLOCK
NO
YES
H2
NOTE THE ADDR OF THE FIRST BLOCK
GUT
J2
R5 = A(NEWBUFFI)
K2
BR R9 BRANCH TO: M12 / M22 / M31 / M41 / STOW
002 A3
SET
A3
SUBSTITUDE THE LEADING BLANKS OF SEQ.NR IN OLDBUFF WITH ZEROS
001 J1
002 C3
STOW
C3
UPDATE THE DIRECTORY (STOW)
ENDE
D3
CLOSE THE SDS
E3
RETURN TO CALLING PROGRAM


A1
ENTRY FROM TCPOPROC VIA ATTACH
B1
CHAIN SAVE AREA, SET INTERVAL TIMER
C1
MODIFY
YES
NO
D1
MODIFY DIRECTORY
YES
NO
E1
SET BCODE=14 LCC12=3
ANF
G1
READ DIRECTORY ENTRY OF THE MEMBER
H1
DOES THE MEMBER EXIST
YES
NO
A2
NAMEDA
A2
MODIFY
NO
YES
B2
SET LCC12=2
B5
X1
D2
SET COMPLETION CODE
NAMENO
E2
READ
NO
YES
A5
F2
SET LCC12=1
G2
FROM KEYBOARD
NO
YES
A3
H2
SET WCODE5=1 ISO
A3
A3
Y
A3
DATA
YES
NO
B3
FORMATTED
YES
NO
A4
B5
Y1
C3
BINARY CODE
NO
YES
D3
FORMATTED
YES
NO
A4
Y2
E3
DATA
NO
YES
F3
FORMATTED
NO
YES
G3
CHECK FORMAT
H3
VALID
YES
NO
C4
J3
SET COMPLETION CODE
FALSCH
A4
SET COMPLETION CODE
C4
ENDE
C4
WITHOUT SEQ-NR.
NO
YES
F4
D4
SET WCODE4=4
F4
Z1
F4
WITH SEQ-NR.
NO
YES
G4
SET WCODE4=3
Z2
J4
SET COMPLETION CODE
X2
A5
SET COMPLETION CODE
ENDE1
B5
TEST INTERVAL TIMER
C5
BRANCH TO THE CALLING PROGRAM


A1
ENTRY FROM TCPOPROC VIA ATTACH
B1
CHAIN SAVE AREA, SET INTERVAL TIMER
C1
SET CODE TO INDICATE LAST INPUT-RECORD PROCESSED (INIT.)
D1
SET EODAD AND SYNAD-ADDR. INTO DCB'S
E1
READ DIRECTORY ENTRY OF THE MEMBER (BLDL)
F1
DOES IT EXIST
YES
NO
G1
SET COMPLETION CODE
J5
GO11
A2
MOVE LISTFORM FOR DECBS
B2
GET INFORMATION FROM DIRECTORY ENTRY
C2
INITIALIZE AREAS AND REGISTERS
D2
FIND THE BEGINNING ADDR. OF THE MEMBER
E2
BLOCKIN
E2
READ A BLOCK OF THE MEMBER
F2
END OF DATA
YES
NO
G2
SEQ-NR. WITH LEADING BLANKS
YES
NO
H2
SUBSTITUTE LEADING BLANKS WITH ZERO
NURO
J2
EODAD FOR SPOOLDS INDICATED
YES
NO
NULLDA
A5
J1
WITH SEQUENCE-NR
YES
NO
GO1
K1
SET COMPLETION CODE
K2
LAST INPUT RECORD PROCESSED
NO
YES
J5
A3
A3
RECORDIN
A3
READ A RECORD FROM INPUT STREAM
B3
END OF DATA
NO
YES
C3
SET CODE TO INDICATE EODAD FOR INPUT DATA
A5
WORK1
E3
SEQ-NR < SEQ-NR OF PREVIOUS RECORD
NO
YES
G3
NUFE1
F3
UPDATE THE CONDITION CODE
H5
G3
WORK2
G3
CLEAR THE CODE WHICH INDICATES THE LAST INPUT RECORD IS PROCESSED
COMPARE
H3
FIND THE RECORD OF THE MEMBER TO BE MODIFIED
J3
FOUNDED
YES
NO
A4
K3
END OF BLOCK
NO
YES
A5
A3
WORK3
A4
CHECK THE INPUT RECORD FOR VALIDITY
A4
B4
VALID
NO
YES
RECLOOP
C4
FIND THE DATA CHAIN IN RECORD OF THE MEMBER TO BE MOD.
D4
FOUNDED
NO
YES
VORMOVE
E4
INPUT RECORD VALID
NO
YES
PA
F4
WILL COLUMN 72 BE EXCEEDED BY MODIFY
YES
NO
EXEC
G4
MODIFY THE REC. THIS PROCESS INCLUDES LABELS KUERZER + LAENGER
RECWEND
H4
LOAD REGISTERS WITH RELEVANT ADDRESSES AND VALUES
J4
SET CODE TO INDICATE PROCESSING OF LAST INPUT RECORD
A3
BLOCKOUT
A5
WRITE THE BLOCK INTO PDS
A5
B5
FIRST BLOCK
NO
YES
C5
NOTE THE ADDR. OF THE FIRST BLOCK
NONOT
D5
SET CODE TO INDICATE PROCESSING OF THE FIRST BLOCK
E2
NUFE
F5
UPDATE THE CONDITION CODE
FNDWORK
H5
REPLACE THE DIRECTORY ENTRY OF THE MEMBER WITH NEW ENTRY
J5
H5
ENDE
J5
CLOSE THE DATA SET
J5
K5
BRANCH TO THE CALLING PROGRAM

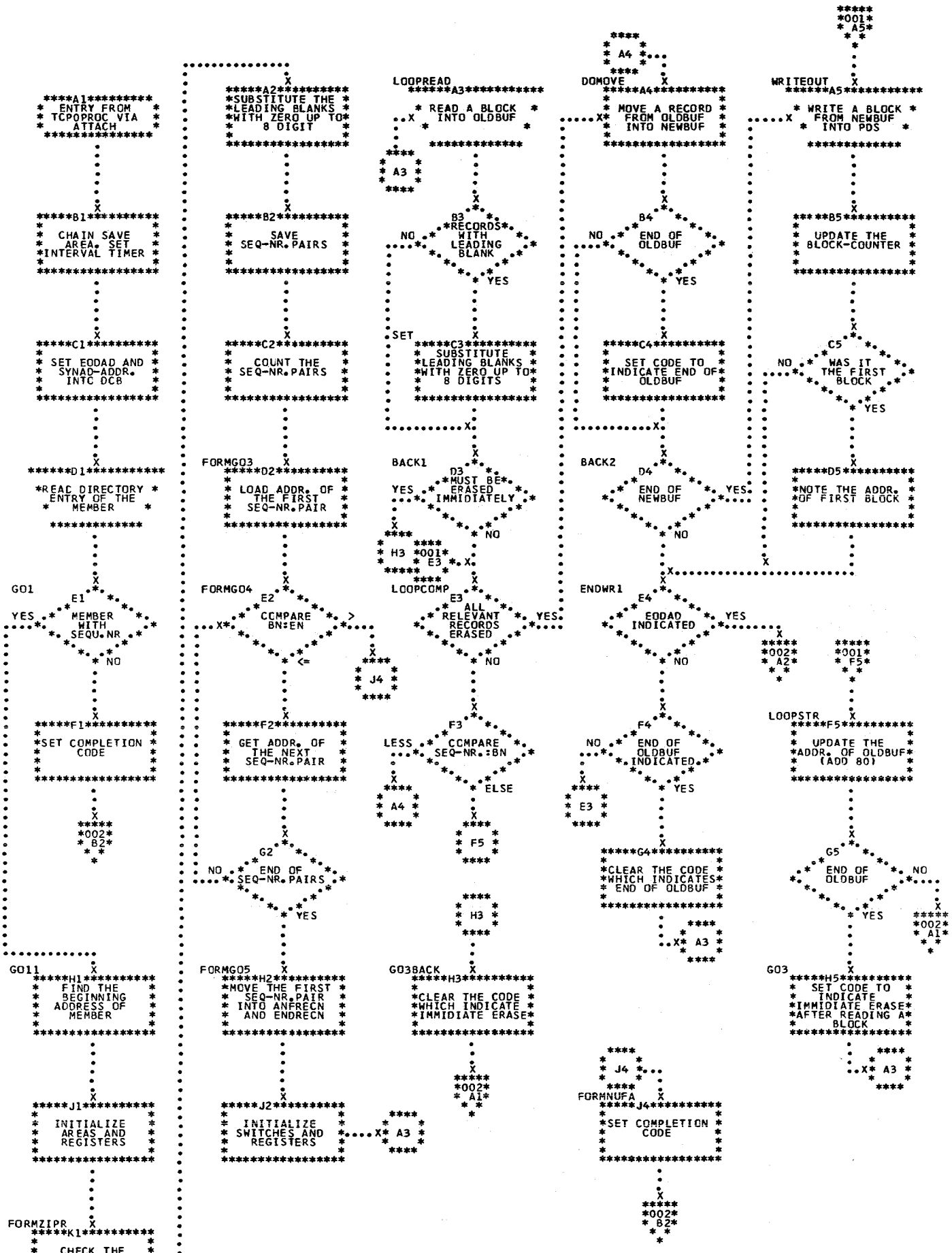

GOOUTSP
A2
SET CODE FOR SYSTEM OUTPUT IN LBX
A1
ENTRY FROM TCPOPROC VIA ATTACH
ENDE
B2
CLOSE TYPE=T FOR SDS
B1
CHAIN SAVE AREA
ENDHE
C2
TEST TIMER INTERVAL
C1
SYSTEM OUTPUT
YES
NO
D2
RETURN TO THE CALLING PROGRAM
D1
SET EODAD AND SYNAD-ADDR. INTO DCB
E1
INITIALIZE REGISTERS
F1
FIND THE BEGINNING ADDR. OF THE MEMBER
LOOPBL
G1
READ A BLOCK FROM PDS INTO BUFFER1
G1
H1
GET LENGTH OF BLOCK
LOOPREC
J1
WRITE A RECORD FROM BUFFER1 INTO SDS
K1
END OF BUFFER1
YES
NO
G1

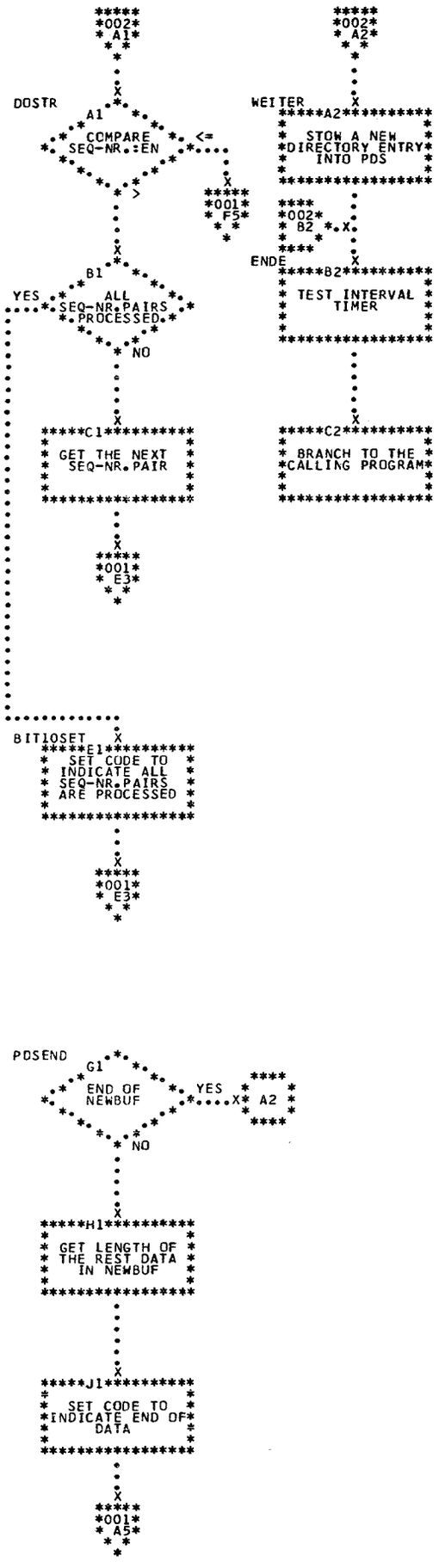

A1
ENTRY FROM TCPOPROC VIA ATTACH
B1
CHAIN SAVE AREA
C1
SET INTERVAL TIMER
D1
SET EODAD AND SYNAD-ADDR. INTO DCB
E1
GET BEGINNING AND END-NR. CHECK THEM FOR VALIDITY
F1
FIND THE BEGINNING ADDRESS OF THE MEMBER
G1
READ A BLOCK FROM PDS INTO BUFFER1
H1
GET LENGTH OF BLOCK
J1
SEQ-NR. WITH LEADING BLANKS
K1
IS IT THE FIRST BLOCK
A2
BN-EN ALREADY COMPARED
G2
B2
COMPARE BN-EN
C2
BN = EN
D2
BN > EN
D4
E2
SET CODE TO INDICATE BN < EN
F2
SET CODE TO INDICATE BN = EN
G2
G2
H2
SET CODE AND GOTO ENDE IF BN > SEQ-NR OF FIRST RECORD
J2
FIRST RECORD FOUNDED
K2
END OF BUFFER1
G1
A3
SCHREIBE
A3
BN=EN
B3
PUT OUT THE RECORD ON TERMINAL
D3
XYZ
D3
PUT OUT THE RECORD ON TERMINAL
E3
UPDATE THE ADDR. IN BUFFER1
F3
END OF BUFFER1
G3
SEQ-NR OF LAST RECORD = EN
J3
SET CODE TO INDICATE PROCESSING OG FIRST BLOCK
G1
A4
SET
A4
PADD SEQ-NR WITH LEADING ZEROS
K1
D4
ANFHOCH
D4
SET COMPLETION CODE
E4
CLOSE DATASET
F4
TEST INTERVAL TIMER
G4
BRANCH TO THE CALLING PROGRAM
YES
NO
EINZIG
NEUBLOCK
GUT
XY
BACK
ENDE

MODULE TCP1LREC

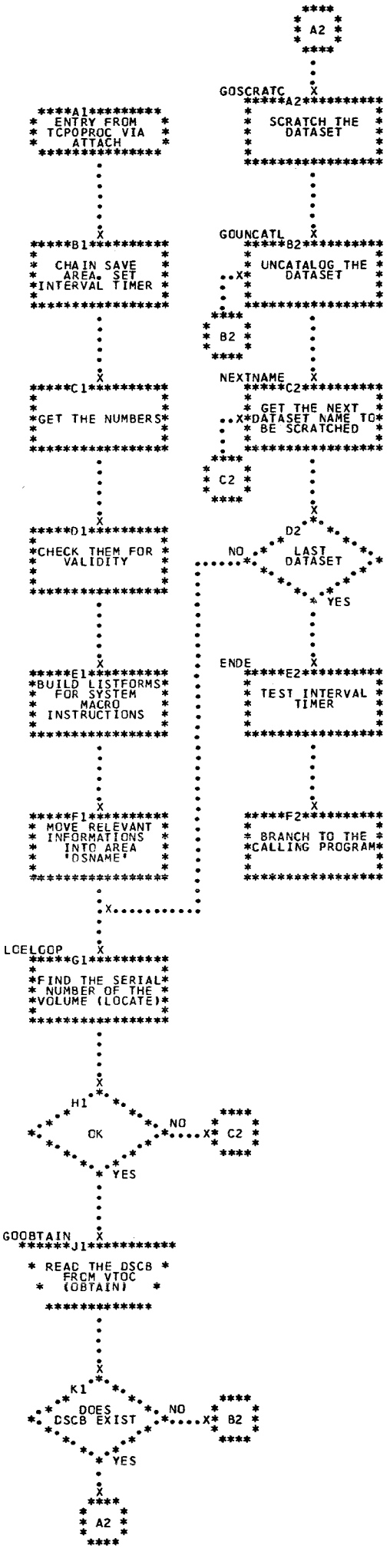
BN = BEGINNING-NR. END = END-NR.

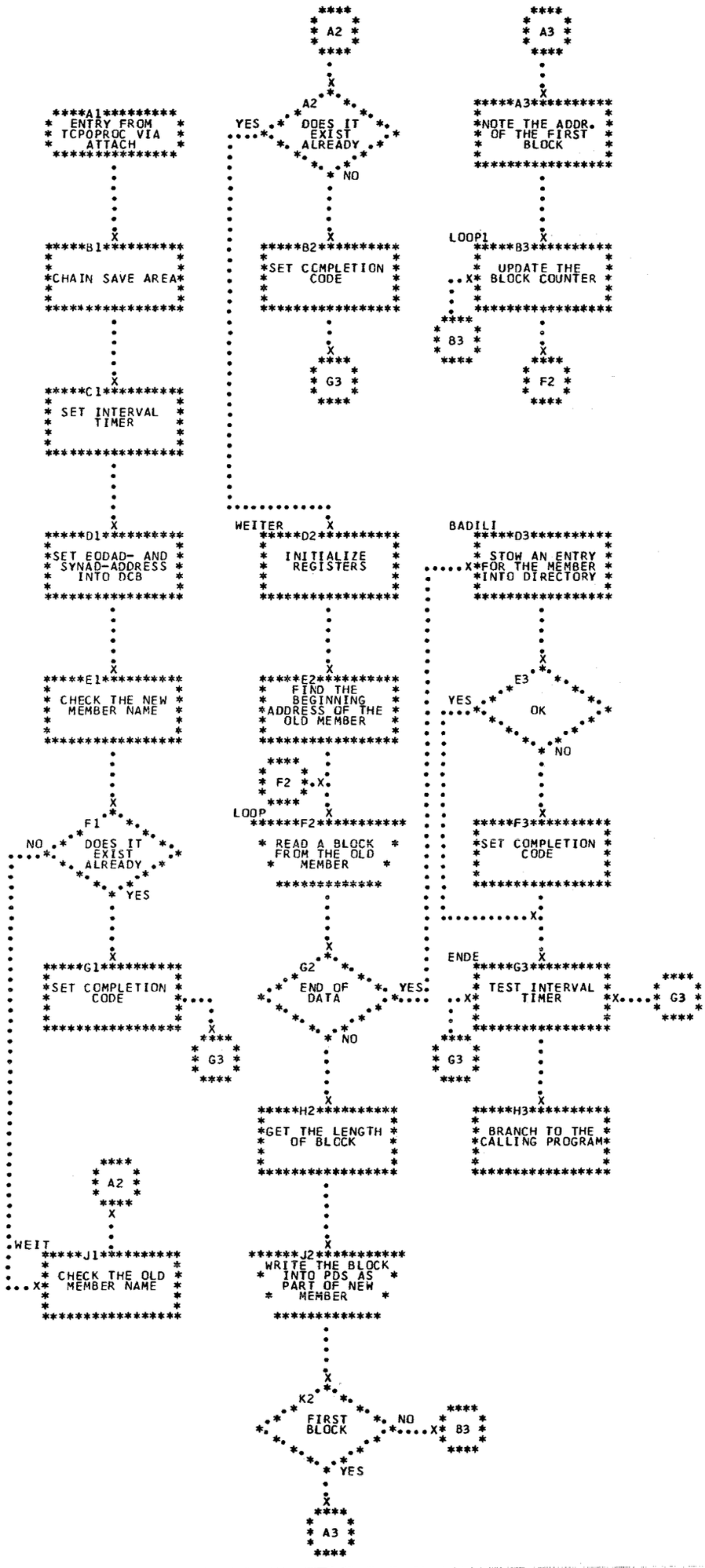
SEQ-NR. = SEQUENCENUMBER

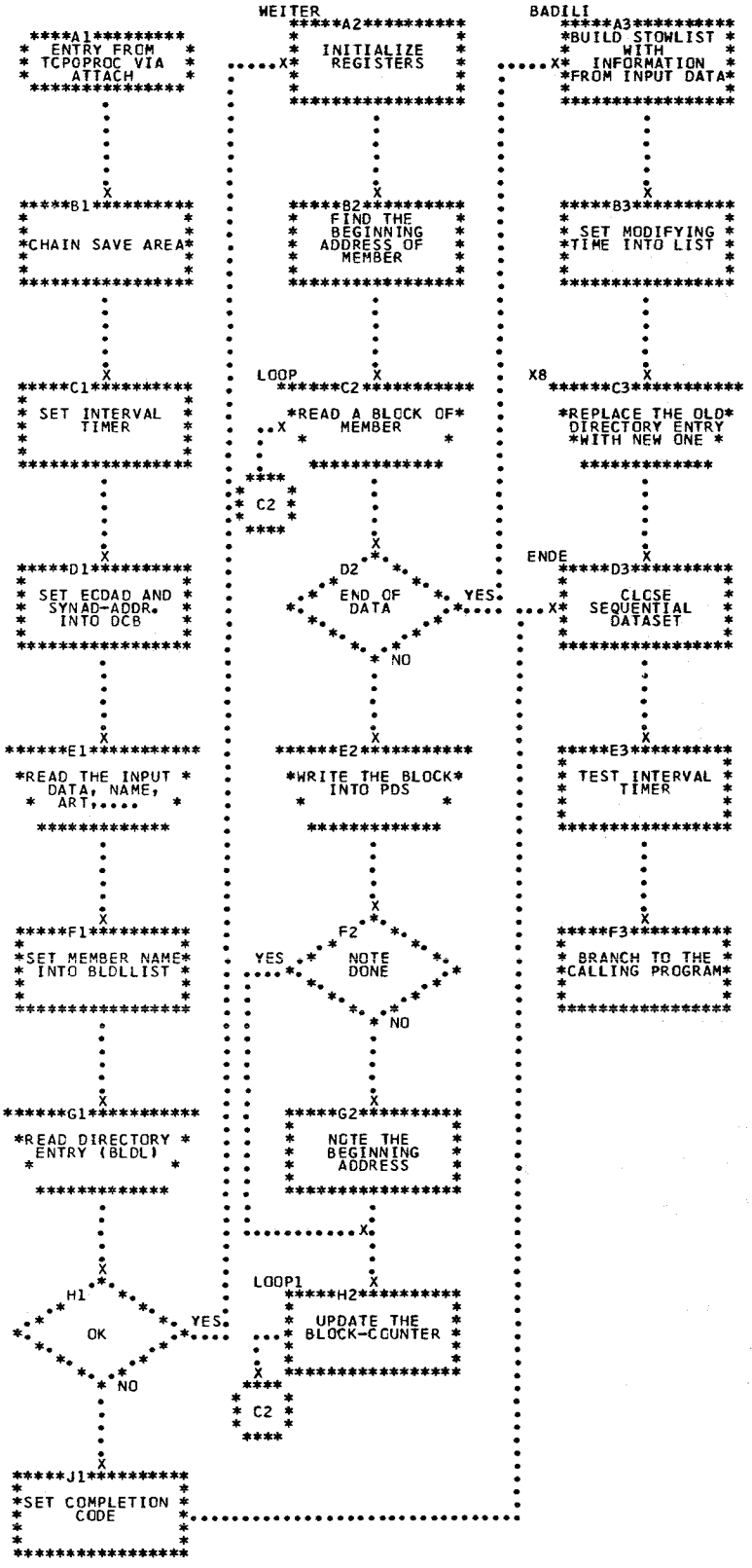


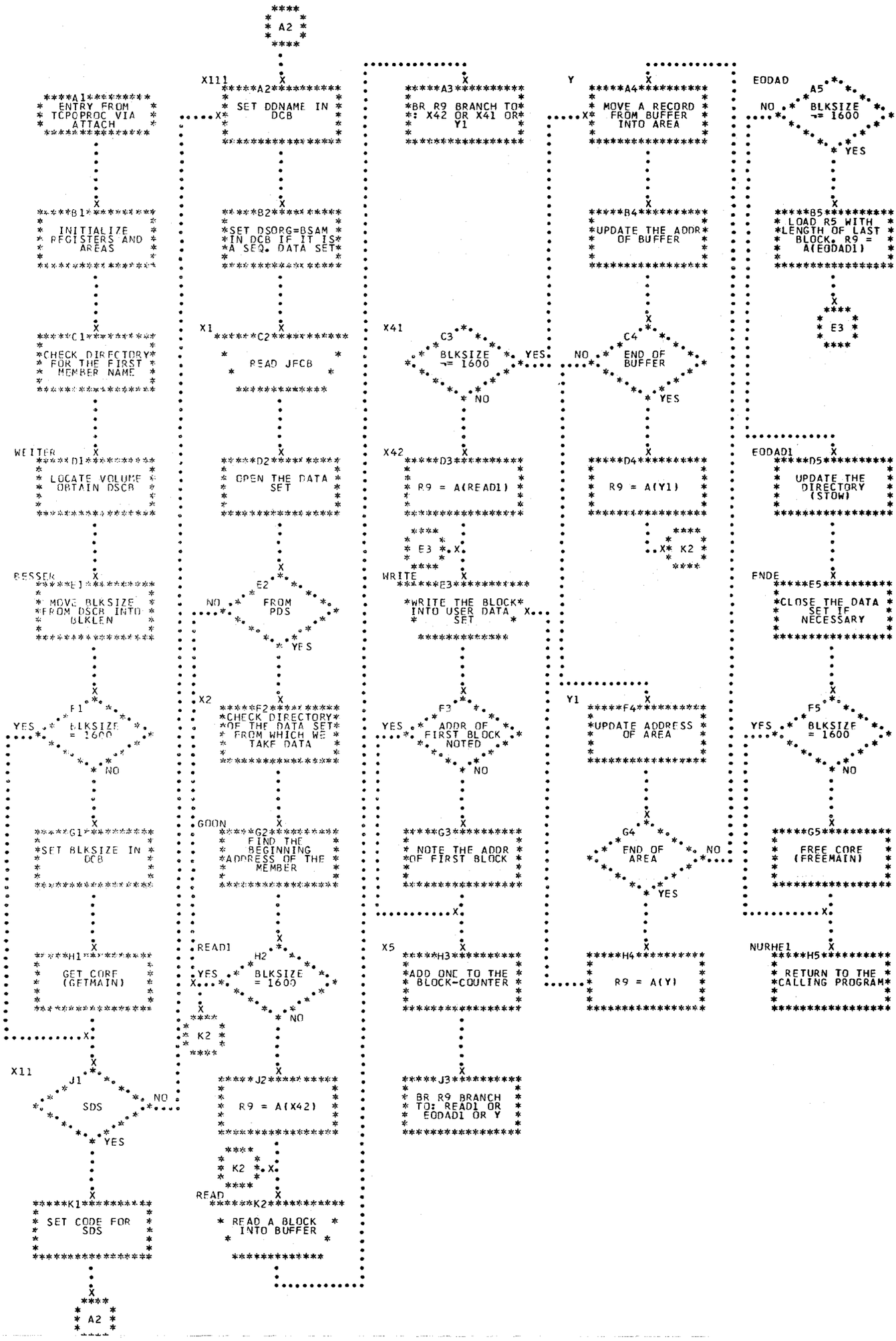

002
A1
DOSTR
COMPARE SEQ-NR.:EN
<=
>
001
F5
B1
YES
ALL SEQ-NR.PAIRS PROCESSED
NO
C1
GET THE NEXT SEQ-NR.PAIR
001
E3
BIT1OSET
E1
SET CODE TO INDICATE ALL SEQ-NR.PAIRS ARE PROCESSED
001
E3
002
A2
WEITER
A2
STOW A NEW DIRECTORY ENTRY INTO PDS
002
B2
ENDE
B2
TEST INTERVAL TIMER
C2
BRANCH TO THE CALLING PROGRAM

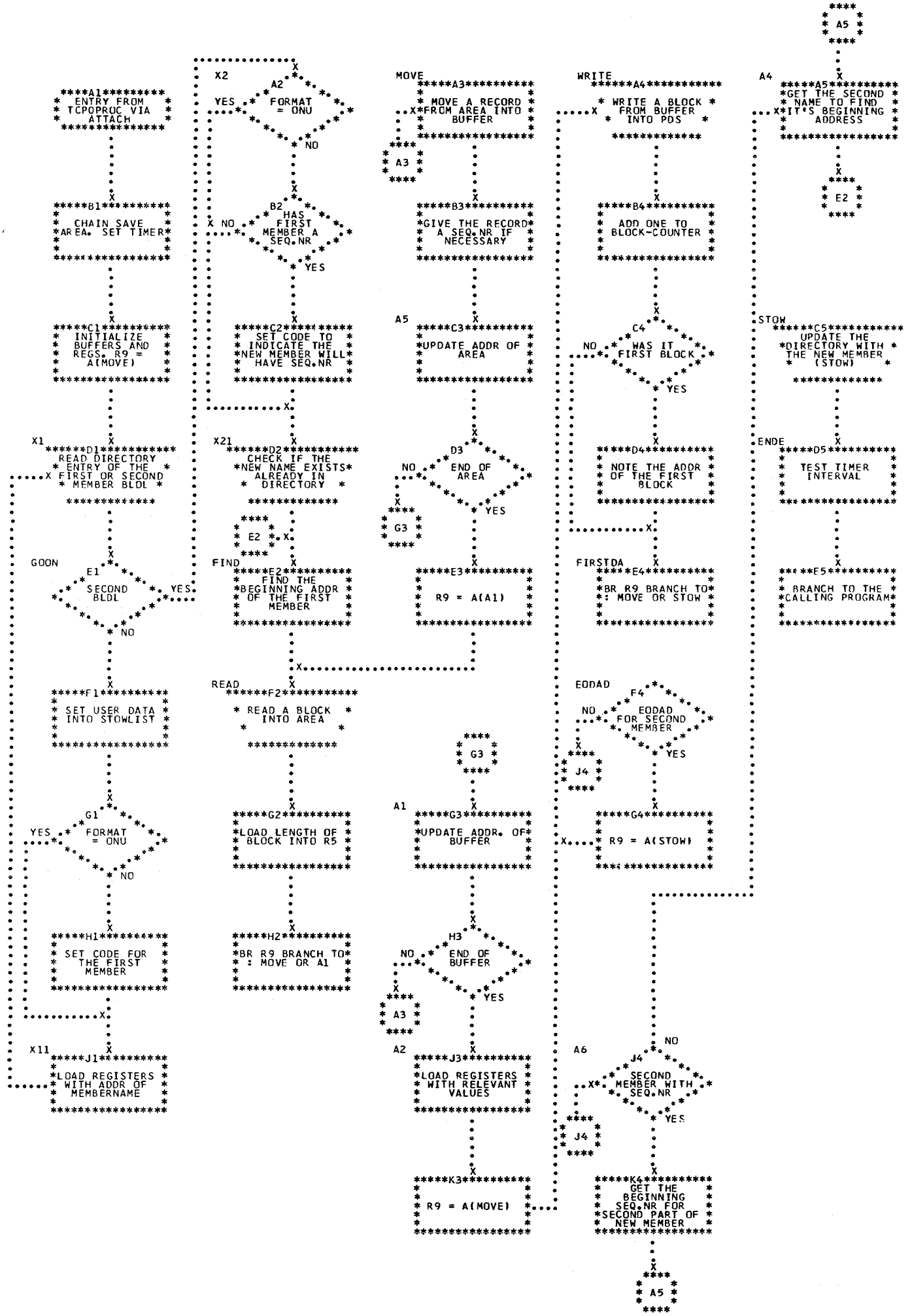
PDSEND
G1
END OF NEWBUF
YES
A2
NO
H1
GET LENGTH OF THE REST DATA IN NEWBUF
J1
SET CODE TO INDICATE END OF DATA
001
A5

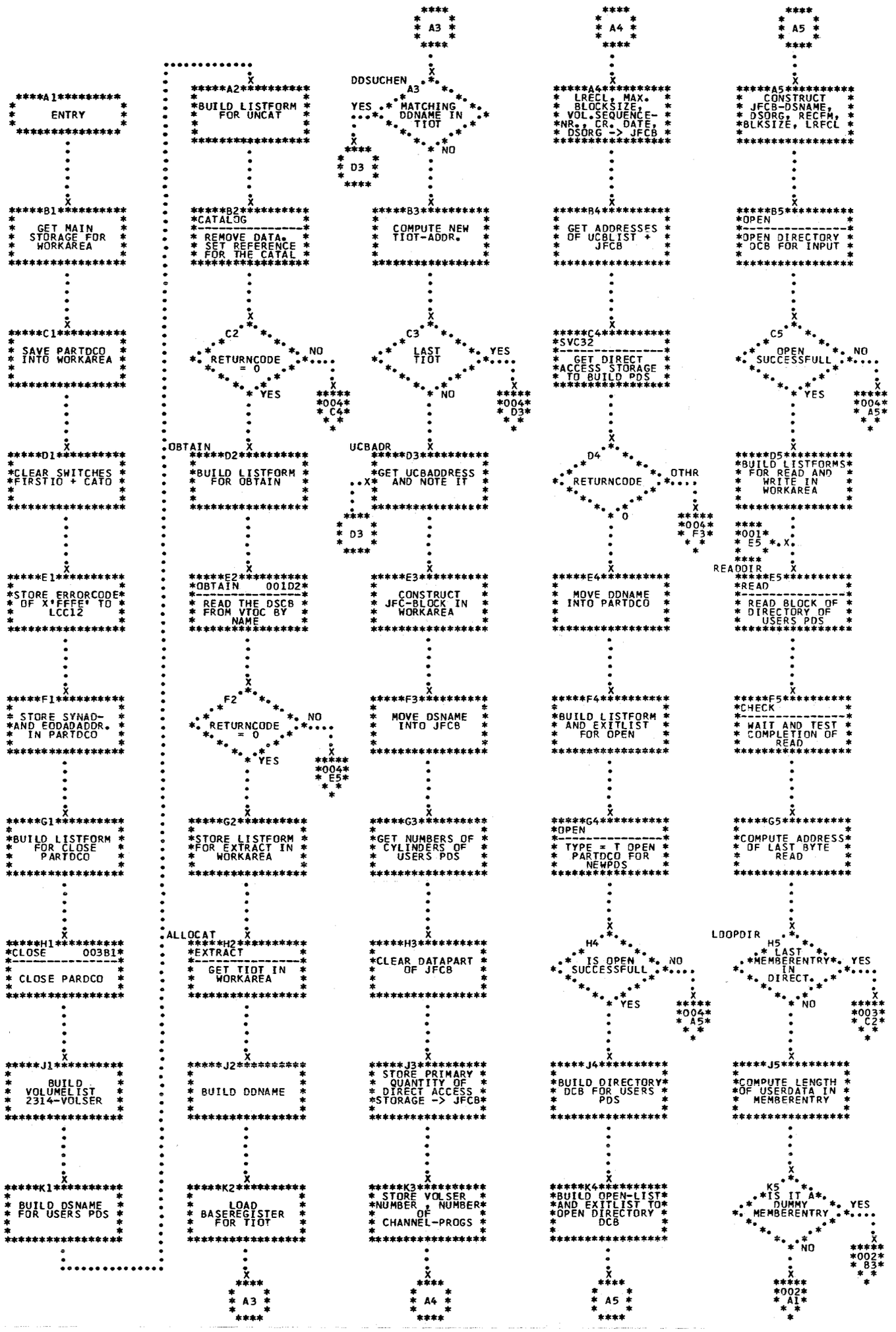

A2
GOSCRATC
A2
SCRATCH THE DATASET
A1
ENTRY FROM TCPOPROC VIA ATTACH
GOUNCATL
B2
UNCATALOG THE DATASET
B1
CHAIN SAVE AREA, SET INTERVAL TIMER
B2
NEXTNAME
C2
GET THE NEXT DATASET NAME TO BE SCRATCHED
C2
C1
GET THE NUMBERS
D1
CHECK THEM FOR VALIDITY
D2
LAST DATASET
NO
YES
ENDE
E2
TEST INTERVAL TIMER
E1
BUILD LISTFORMS FOR SYSTEM MACRO INSTRUCTIONS
F2
BRANCH TO THE CALLING PROGRAM
F1
MOVE RELEVANT INFORMATIONS INTO AREA 'DSNAME'
LOELOOP
G1
FIND THE SERIAL NUMBER OF THE VOLUME (LOCATE)
H1
OK
NO
C2
YES
GOOBTAIN
J1
READ THE DSCB FROM VTOC (OBTAIN)
K1
DOES DSCB EXIST
NO
B2
YES
A2

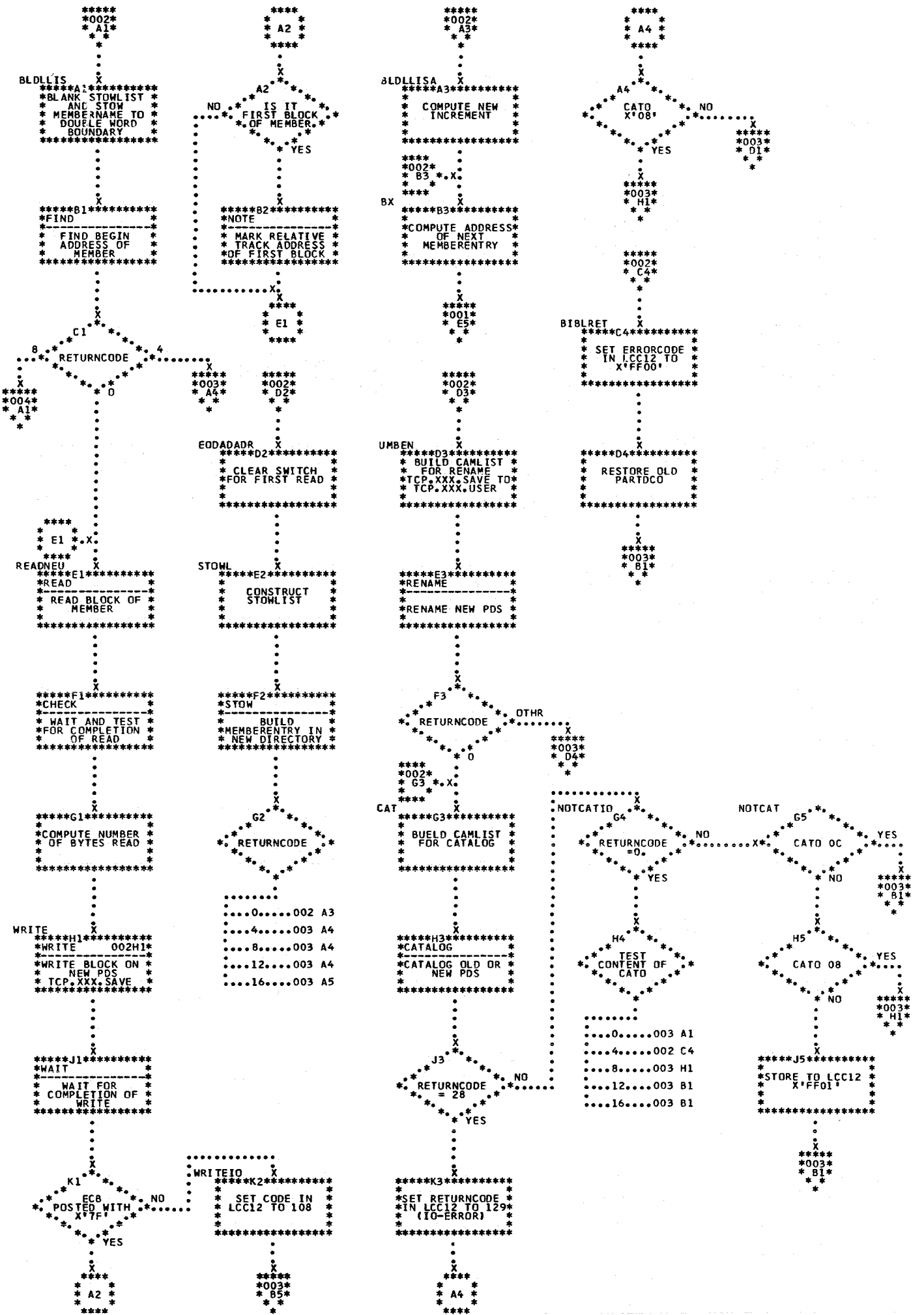

A1
ENTRY FROM TCPOPROC VIA ATTACH
B1
CHAIN SAVE AREA
C1
SET INTERVAL TIMER
D1
SET EODAD- AND SYNAD-ADDRESS INTO DCB
E1
CHECK THE NEW MEMBER NAME
F1
DOES IT EXIST ALREADY
YES
NO
G1
SET COMPLETION CODE
G3
A2
DOES IT EXIST ALREADY
YES
NO
B2
SET COMPLETION CODE
G3
WEITER
D2
INITIALIZE REGISTERS
E2
FIND THE BEGINNING ADDRESS OF THE OLD MEMBER
F2
LOOP
F2
READ A BLOCK FROM THE OLD MEMBER
G2
END OF DATA
YES
NO
H2
GET THE LENGTH OF BLOCK
A2
WEIT
J1
CHECK THE OLD MEMBER NAME
J2
WRITE THE BLOCK INTO PDS AS PART OF NEW MEMBER
K2
FIRST BLOCK
NO
B3
YES
A3
A3
NOTE THE ADDR. OF THE FIRST BLOCK
LOOP1
B3
UPDATE THE BLOCK COUNTER
B3
F2
BADILI
D3
STOW AN ENTRY FOR THE MEMBER INTO DIRECTORY
E3
OK
YES
NO
F3
SET COMPLETION CODE
ENDE
G3
TEST INTERVAL TIMER
G3
H3
BRANCH TO THE CALLING PROGRAM

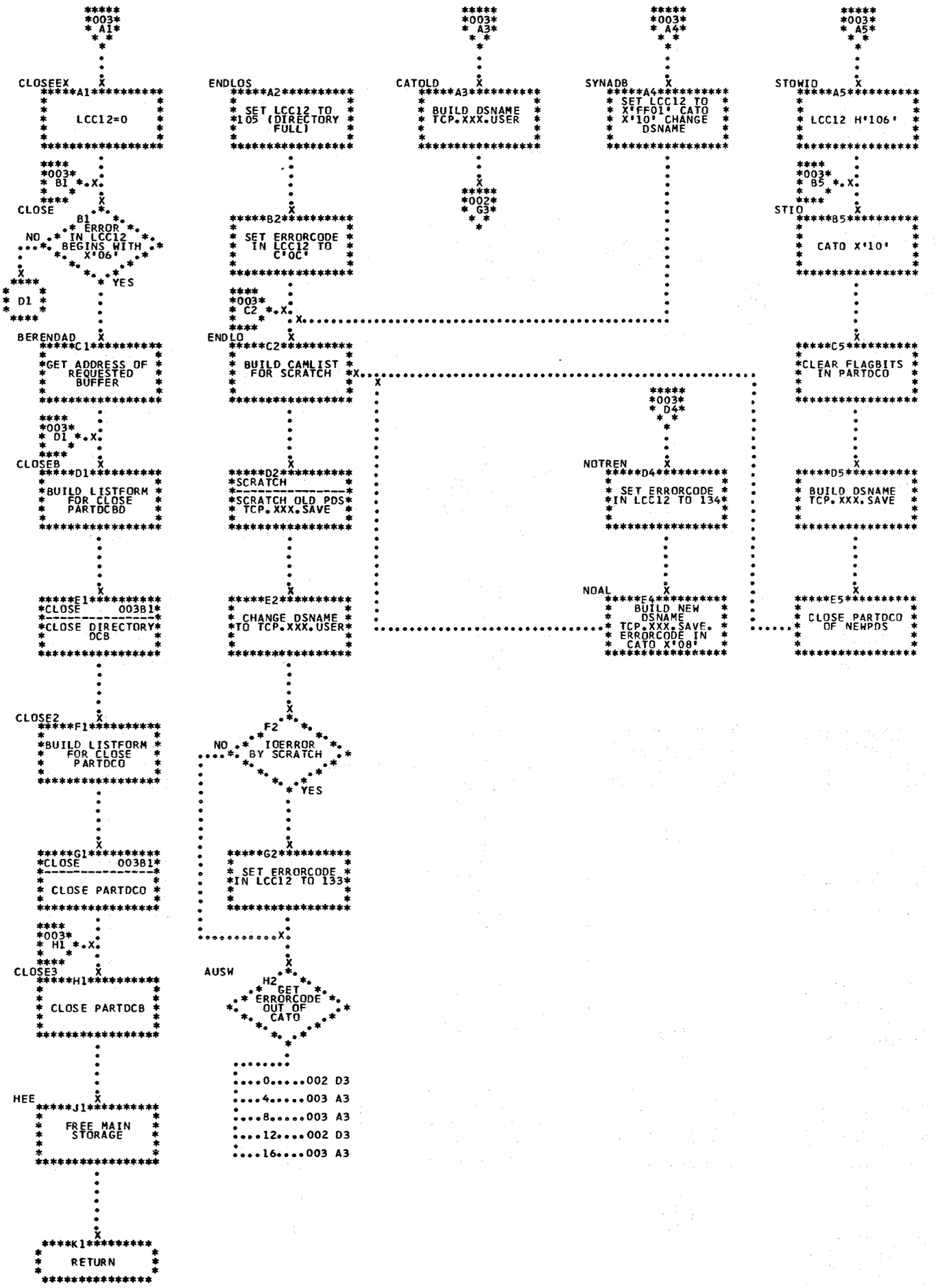

WEITER
BADILI
A1
ENTRY FROM TCPOPROC VIA ATTACH
A2
INITIALIZE REGISTERS
A3
BUILD STOWLIST WITH INFORMATION FROM INPUT DATA
B1
CHAIN SAVE AREA
B2
FIND THE BEGINNING ADDRESS OF MEMBER
B3
SET MODIFYING TIME INTO LIST
C1
SET INTERVAL TIMER
LOOP
C2
READ A BLOCK OF MEMBER
X8
C3
REPLACE THE OLD DIRECTORY ENTRY WITH NEW ONE
C2
D1
SET ECDAD AND SYNAD-ADDR. INTO DCB
D2
END OF DATA
YES
NO
ENDE
D3
CLOSE SEQUENTIAL DATASET
E1
READ THE INPUT DATA, NAME, ART,....
E2
WRITE THE BLOCK INTO PDS
E3
TEST INTERVAL TIMER
F1
SET MEMBER NAME INTO BLDLLIST
F2
NOTE DONE
YES
NO
F3
BRANCH TO THE CALLING PROGRAM
G1
READ DIRECTORY ENTRY (BLDL)
G2
NOTE THE BEGINNING ADDRESS
H1
OK
YES
NO
LOOP1
H2
UPDATE THE BLOCK-COUNTER
C2
J1
SET COMPLETION CODE

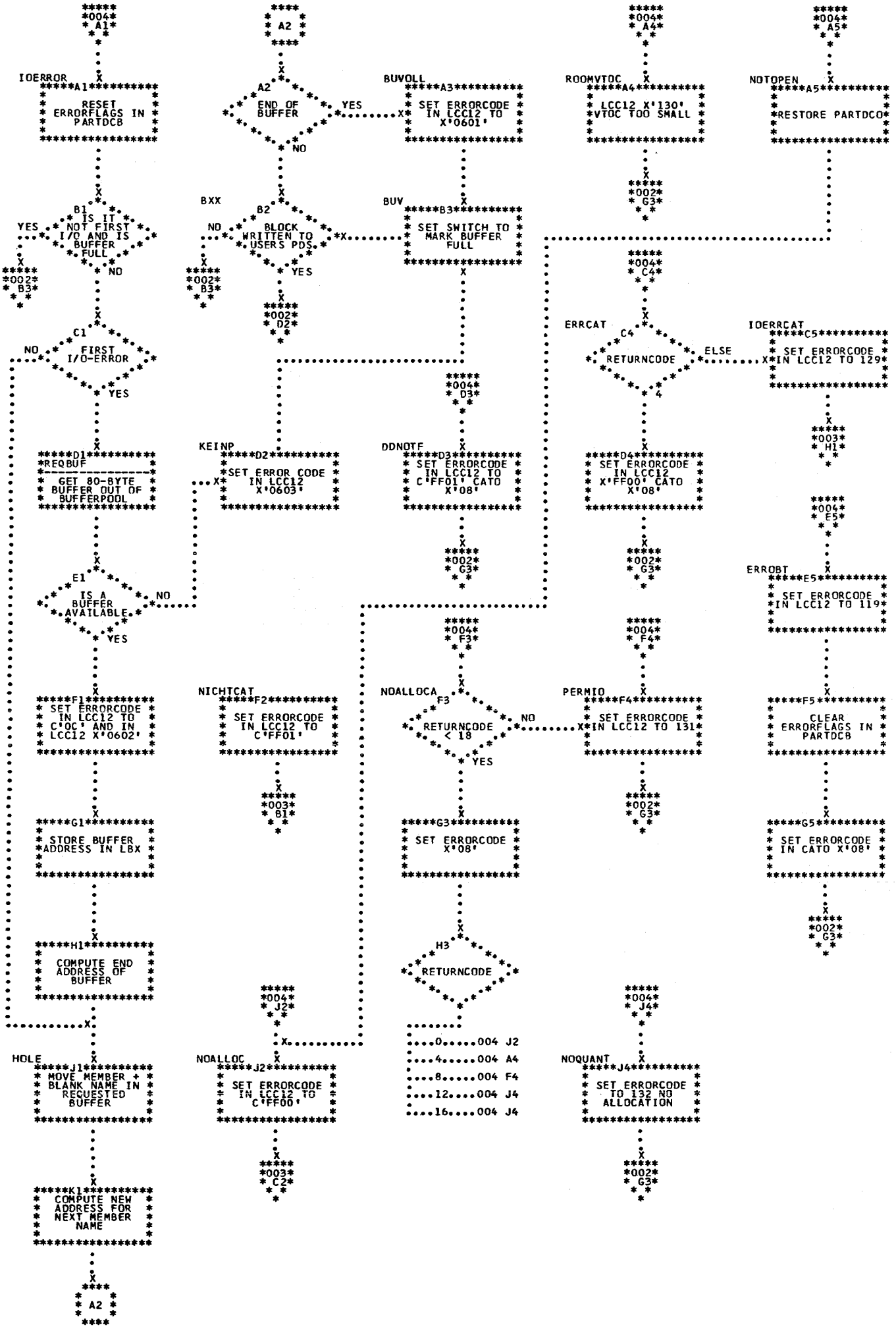

A2
A1
ENTRY FROM TCPOPROC VIA ATTACH
B1
INITIALIZE REGISTERS AND AREAS
C1
CHECK DIRECTORY FOR THE FIRST MEMBER NAME
WEITER
D1
LOCATE VOLUME OBTAIN DSCB
BESSER
E1
MOVE BLKSIZE FROM DSCB INTO BLKLEN
F1
BLKSIZE = 1600
NO
YES
G1
SET BLKSIZE IN DCB
H1
GET CORE (GETMAIN)
X11
J1
SDS
NO
YES
K1
SET CODE FOR SDS
A2
X111
A2
SET DDNAME IN DCB
B2
SET DSORG=BSAM IN DCB IF IT IS A SEQ. DATA SET
X1
C2
READ JFCB
D2
OPEN THE DATA SET
E2
FROM PDS
NO
YES
X2
F2
CHECK DIRECTORY OF THE DATA SET FROM WHICH WE TAKE DATA
GOON
G2
FIND THE BEGINNING ADDRESS OF THE MEMBER
READ1
H2
BLKSIZE = 1600
NO
K2
X11
J2
R9 = A(X42)
K2
READ
K2
READ A BLOCK INTO BUFFER
A2
A3
BR R9 BRANCH TO: X42 OR X41 OR Y1
X41
C3
BLKSIZE ¬= 1600
YES
NO
X42
D3
R9 = A(READ1)
E3
WRITE
E3
WRITE THE BLOCK INTO USER DATA SET
F3
ADDR OF FIRST BLOCK NOTED
YES
NO
G3
NOTE THE ADDR OF FIRST BLOCK
X5
H3
ADD ONE TO THE BLOCK-COUNTER
J3
BR R9 BRANCH TO: READ1 OR EODAD1 OR Y
Y
A4
MOVE A RECORD FROM BUFFER INTO AREA
B4
UPDATE THE ADDR OF BUFFER
C4
END OF BUFFER
NO
YES
D4
R9 = A(Y1)
K2
Y1
F4
UPDATE ADDRESS OF AREA
G4
END OF AREA
NO
YES
H4
R9 = A(Y)
EODAD
A5
BLKSIZE ¬= 1600
NO
YES
B5
LOAD R5 WITH LENGTH OF LAST BLOCK, R9 = A(EODAD1)
E3
EODAD1
D5
UPDATE THE DIRECTORY (STOW)
ENDE
E5
CLOSE THE DATA SET IF NECESSARY
F5
BLKSIZE = 1600
NO
YES
G5
FREE CORE (FREEMAIN)
NURHE1
H5
RETURN TO THE CALLING PROGRAM

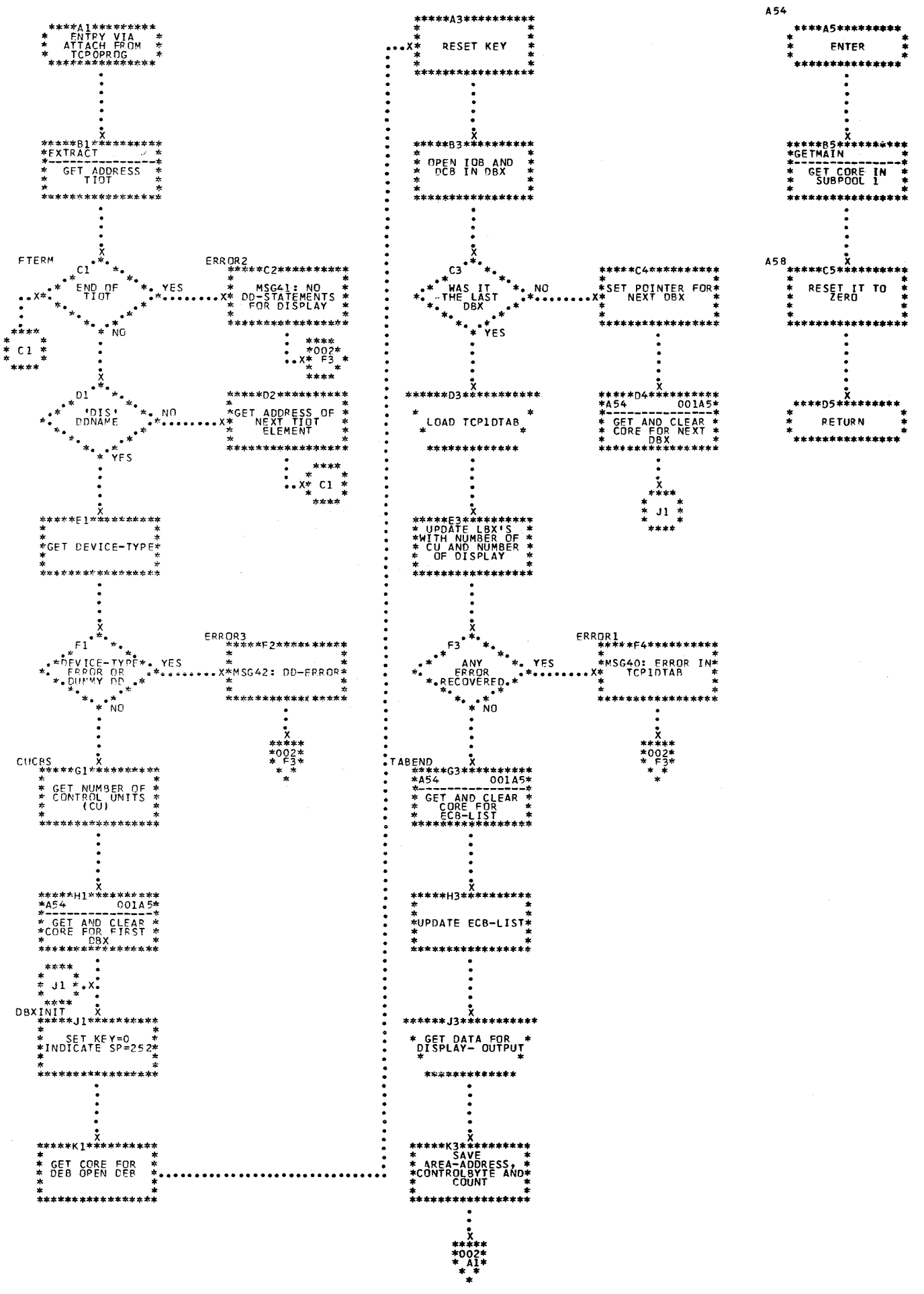

A1
ENTRY FROM TCPOPROC VIA ATTACH
B1
CHAIN SAVE AREA. SET TIMER
C1
INITIALIZE BUFFERS AND REGS. R9 = A(MOVE)
X1
D1
READ DIRECTORY ENTRY OF THE FIRST OR SECOND MEMBER BLDL
GOON
E1
SECOND BLDL
YES
NO
F1
SET USER DATA INTO STOWLIST
G1
FORMAT = ONU
YES
NO
H1
SET CODE FOR THE FIRST MEMBER
X11
J1
LOAD REGISTERS WITH ADDR OF MEMBERNAME
X2
A2
FORMAT = ONU
YES
NO
B2
HAS FIRST MEMBER A SEQ.NR
X NO
YES
C2
SET CODE TO INDICATE THE NEW MEMBER WILL HAVE SEQ.NR
X21
D2
CHECK IF THE NEW NAME EXISTS ALREADY IN DIRECTORY
E2
FIND
E2
FIND THE BEGINNING ADDR OF THE FIRST MEMBER
READ
F2
READ A BLOCK INTO AREA
G2
LOAD LENGTH OF BLOCK INTO R5
H2
BR R9 BRANCH TO : MOVE OR A1
MOVE
A3
MOVE A RECORD FROM AREA INTO BUFFER
A3
B3
GIVE THE RECORD A SEQ.NR IF NECESSARY
A5
C3
UPDATE ADDR OF AREA
D3
END OF AREA
NO
YES
G3
E3
R9 = A(A1)
A1
G3
UPDATE ADDR. OF BUFFER
H3
END OF BUFFER
NO
YES
A3
A2
J3
LOAD REGISTERS WITH RELEVANT VALUES
K3
R9 = A(MOVE)
WRITE
A4
WRITE A BLOCK FROM BUFFER INTO PDS
B4
ADD ONE TO BLOCK-COUNTER
C4
WAS IT FIRST BLOCK
NO
YES
D4
NOTE THE ADDR OF THE FIRST BLOCK
FIRSTDA
E4
BR R9 BRANCH TO : MOVE OR STOW
EODAD
F4
EODAD FOR SECOND MEMBER
NO
YES
J4
G4
R9 = A(STOW)
A6
J4
SECOND MEMBER WITH SEQ.NR
NO
YES
J4
K4
GET THE BEGINNING SEQ.NR FOR SECOND PART OF NEW MEMBER
A5
A4
A5
GET THE SECOND NAME TO FIND IT'S BEGINNING ADDRESS
E2
STOW
C5
UPDATE THE DIRECTORY WITH THE NEW MEMBER (STOW)
ENDE
D5
TEST TIMER INTERVAL
E5
BRANCH TO THE CALLING PROGRAM
A5

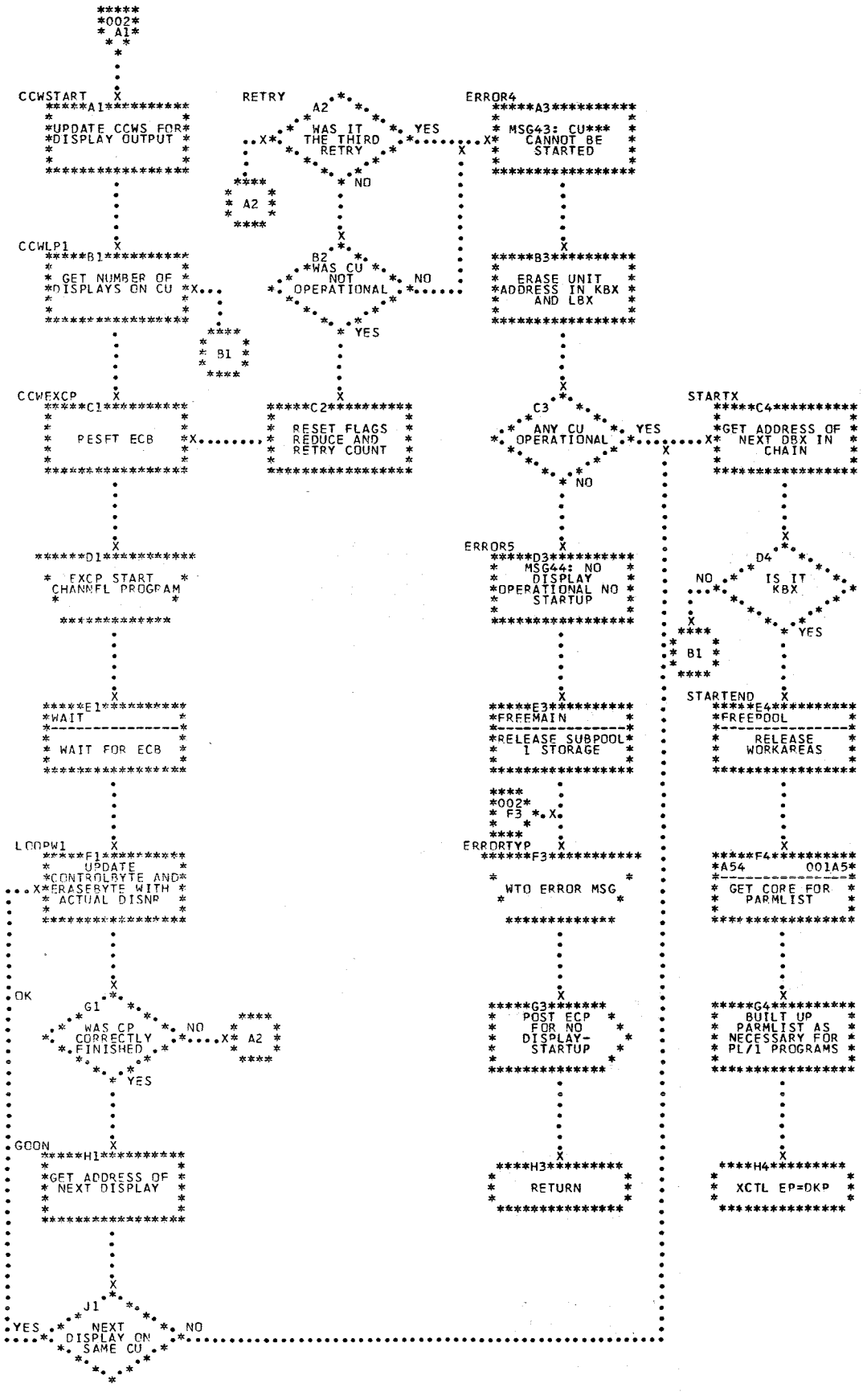

A1
ENTRY
B1
GET MAIN STORAGE FOR WORKAREA
C1
SAVE PARTDCO INTO WORKAREA
D1
CLEAR SWITCHES FIRSTIO + CATO
E1
STORE ERRORCODE OF X'FFFE' TO LCC12
F1
STORE SYNAD- AND EODADADDR. IN PARTDCO
G1
BUILD LISTFORM FOR CLOSE PARTDCO
H1
CLOSE 003B1
CLOSE PARDCO
J1
BUILD VOLUMELIST 2314-VOLSER
K1
BUILD DSNAME FOR USERS PDS
A2
BUILD LISTFORM FOR UNCAT
B2
CATALOG
REMOVE DATA. SET REFERENCE FOR THE CATAL
C2
RETURNCODE = 0
NO
004 C4
YES
OBTAIN
D2
BUILD LISTFORM FOR OBTAIN
E2
OBTAIN 001D2
READ THE DSCB FROM VTOC BY NAME
F2
RETURNCODE = 0
NO
004 E5
YES
G2
STORE LISTFORM FOR EXTRACT IN WORKAREA
ALLOCAT
H2
EXTRACT
GET TIOT IN WORKAREA
J2
BUILD DDNAME
K2
LOAD BASEREGISTER FOR TIOT
A3
DDSUCHEN
A3
MATCHING DDNAME IN TIOT
YES
NO
D3
B3
COMPUTE NEW TIOT-ADDR.
C3
LAST TIOT
YES
NO
004 D3
UCBADR
D3
GET UCBADDRESS AND NOTE IT
D3
E3
CONSTRUCT JFC-BLOCK IN WORKAREA
F3
MOVE DSNAME INTO JFCB
G3
GET NUMBERS OF CYLINDERS OF USERS PDS
H3
CLEAR DATAPART OF JFCB
J3
STORE PRIMARY QUANTITY OF DIRECT ACCESS STORAGE -> JFCB
K3
STORE VOLSER NUMBER , NUMBER OF CHANNEL-PROGS
A4
A4
LRECL, MAX. BLOCKSIZE, VOL.SEQUENCE-NR., CR. DATE, DSORG -> JFCB
B4
GET ADDRESSES OF UCBLIST + JFCB
C4
SVC32
GET DIRECT ACCESS STORAGE TO BUILD PDS
D4
RETURNCODE
OTHR
0
004 F3
E4
MOVE DDNAME INTO PARTDCO
F4
BUILD LISTFORM AND EXITLIST FOR OPEN
G4
OPEN
TYPE = T OPEN PARTDCO FOR NEWPDS
H4
IS OPEN SUCCESSFULL
NO
YES
004 A5
J4
BUILD DIRECTORY DCB FOR USERS PDS
K4
BUILD OPEN-LIST AND EXITLIST TO OPEN DIRECTORY DCB
A5
A5
CONSTRUCT JFCB-DSNAME, DSORG, RECFM, BLKSIZE, LRECL
B5
OPEN
OPEN DIRECTORY DCB FOR INPUT
C5
OPEN SUCCESSFULL
NO
YES
004 A5
D5
BUILD LISTFORMS FOR READ AND WRITE IN WORKAREA
001 E5
READDIR
E5
READ
READ BLOCK OF DIRECTORY OF USERS PDS
F5
CHECK
WAIT AND TEST COMPLETION OF READ
G5
COMPUTE ADDRESS OF LAST BYTE READ
LOOPDIR
H5
LAST MEMBERENTRY IN DIRECT.
YES
NO
003 C2
J5
COMPUTE LENGTH OF USERDATA IN MEMBERENTRY
K5
IS IT A DUMMY MEMBERENTRY
YES
NO
002 B3
002 A1


002 A1
BLDLLIS
A1
BLANK STOWLIST AND STOW MEMBERNAME TO DOUBLE WORD BOUNDARY
B1
FIND
FIND BEGIN ADDRESS OF MEMBER
C1
RETURNCODE
8
4
0
004 A1
003 A4
002 D2
EODADADR
D2
CLEAR SWITCH FOR FIRST READ
READNEU
E1
READ
READ BLOCK OF MEMBER
STOWL
E2
CONSTRUCT STOWLIST
F1
CHECK
WAIT AND TEST FOR COMPLETION OF READ
F2
STOW
BUILD MEMBERENTRY IN NEW DIRECTORY
G1
COMPUTE NUMBER OF BYTES READ
G2
RETURNCODE
0 002 A3
4 003 A4
8 003 A4
12 003 A4
16 003 A5
WRITE
H1
WRITE 002H1
WRITE BLOCK ON NEW PDS TCP.XXX.SAVE
J1
WAIT
WAIT FOR COMPLETION OF WRITE
K1
EC8
POSTED WITH X'7F'
NO
YES
WRITEIO
K2
SET CODE IN LCC12 TO 108
003 B5
002 A2
A2
IS IT FIRST BLOCK OF MEMBER
NO
YES
B2
NOTE
MARK RELATIVE TRACK ADDRESS OF FIRST BLOCK
001 E1
002 A3
BLDLLISA
A3
COMPUTE NEW INCREMENT
002 B3
BX
B3
COMPUTE ADDRESS OF NEXT MEMBERENTRY
001 E5
002 D3
UMBEN
D3
BUILD CAMLIST FOR RENAME TCP.XXX.SAVE TO TCP.XXX.USER
E3
RENAME
RENAME NEW PDS
F3
RETURNCODE
0
OTHR
003 D4
002 G3
CAT
G3
BUELD CAMLIST FOR CATALOG
H3
CATALOG
CATALOG OLD OR NEW PDS
J3
RETURNCODE = 28
NO
YES
K3
SET RETURNCODE IN LCC12 TO 129 (IO-ERROR)
003 B1
002 A4
A4
CATO X'08'
NO
YES
003 D1
003 H1
002 C4
BIBLRET
C4
SET ERRORCODE IN LCC12 TO X'FF00'
D4
RESTORE OLD PARTDCO
003 B1
NOTCATID
G4
RETURNCODE =0
NO
YES
H4
TEST CONTENT OF CATO
0 003 A1
4 002 C4
8 003 H1
12 003 B1
16 003 B1
NOTCAT
G5
CATO OC
YES
NO
003 B1
H5
CATO 08
YES
NO
003 H1
J5
STORE TO LCC12 X'FF01'
003 B1

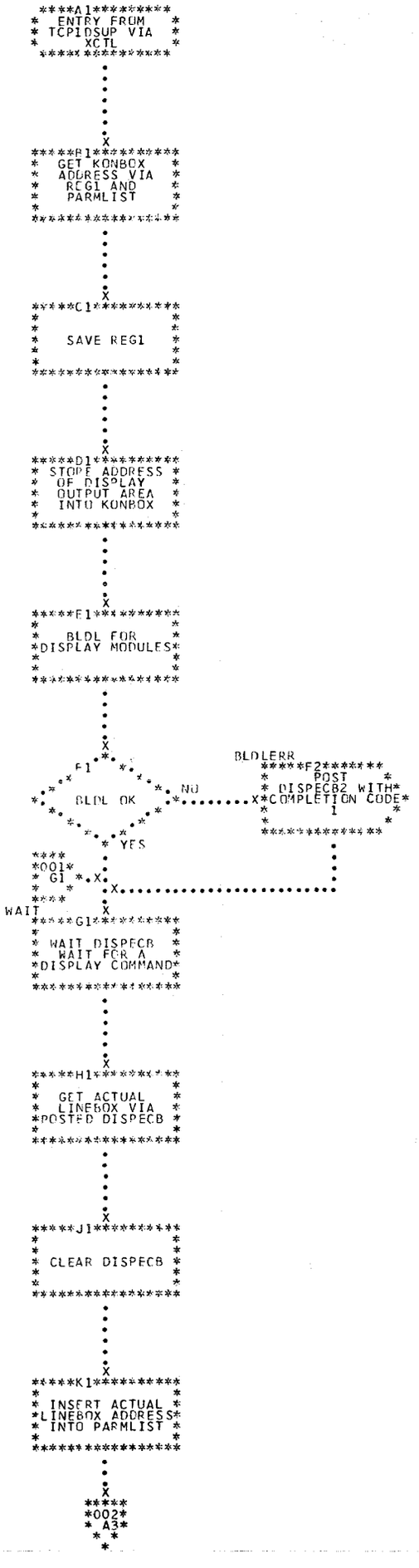

003 A1
CLOSEEX
LCC12=0
003 B1
CLOSE
B1 ERROR
NO IN LCC12 BEGINS WITH X'06'
YES
BERENDAD
GET ADDRESS OF REQUESTED BUFFER
003 D1
CLOSEB
BUILD LISTFORM FOR CLOSE PARTDCBD
CLOSE 003B1
CLOSE DIRECTORY DCB
CLOSE2
BUILD LISTFORM FOR CLOSE PARTDCO
CLOSE 003B1
CLOSE PARTDCO
003 H1
CLOSE3
CLOSE PARTDCB
HEE
FREE MAIN STORAGE
RETURN
ENDLOS
SET LCC12 TO *105 (DIRECTORY FULL)
003 C2
ENDLO
BUILD CAMLIST FOR SCRATCH
SCRATCH
SCRATCH OLD PDS TCP.XXX.SAVE
CHANGE DSNAME TO TCP.XXX.USER
F2
NO IOERROR BY SCRATCH
YES
SET ERRORCODE IN LCC12 TO 133
AUSW
H2
GET ERRORCODE OUT OF CATO
0....002 D3
4....003 A3
8....003 A3
12....002 D3
16....003 A3
CATOLD
BUILD DSNAME TCP.XXX.USER
002 G3
SYNADB
SET LCC12 TO X'FF01' CATO X'10' CHANGE DSNAME
003 D4
NOTREN
SET ERRORCODE IN LCC12 TO 134
NOAL
BUILD NEW DSNAME TCP.XXX.SAVE. ERRORCODE IN CATO X'08'
STOWID
LCC12 H'106'
003 B5
STIO
CATO X'10'
CLEAR FLAGBITS IN PARTDCO
BUILD DSNAME TCP.XXX.SAVE
CLOSE PARTDCO OF NEWPDS

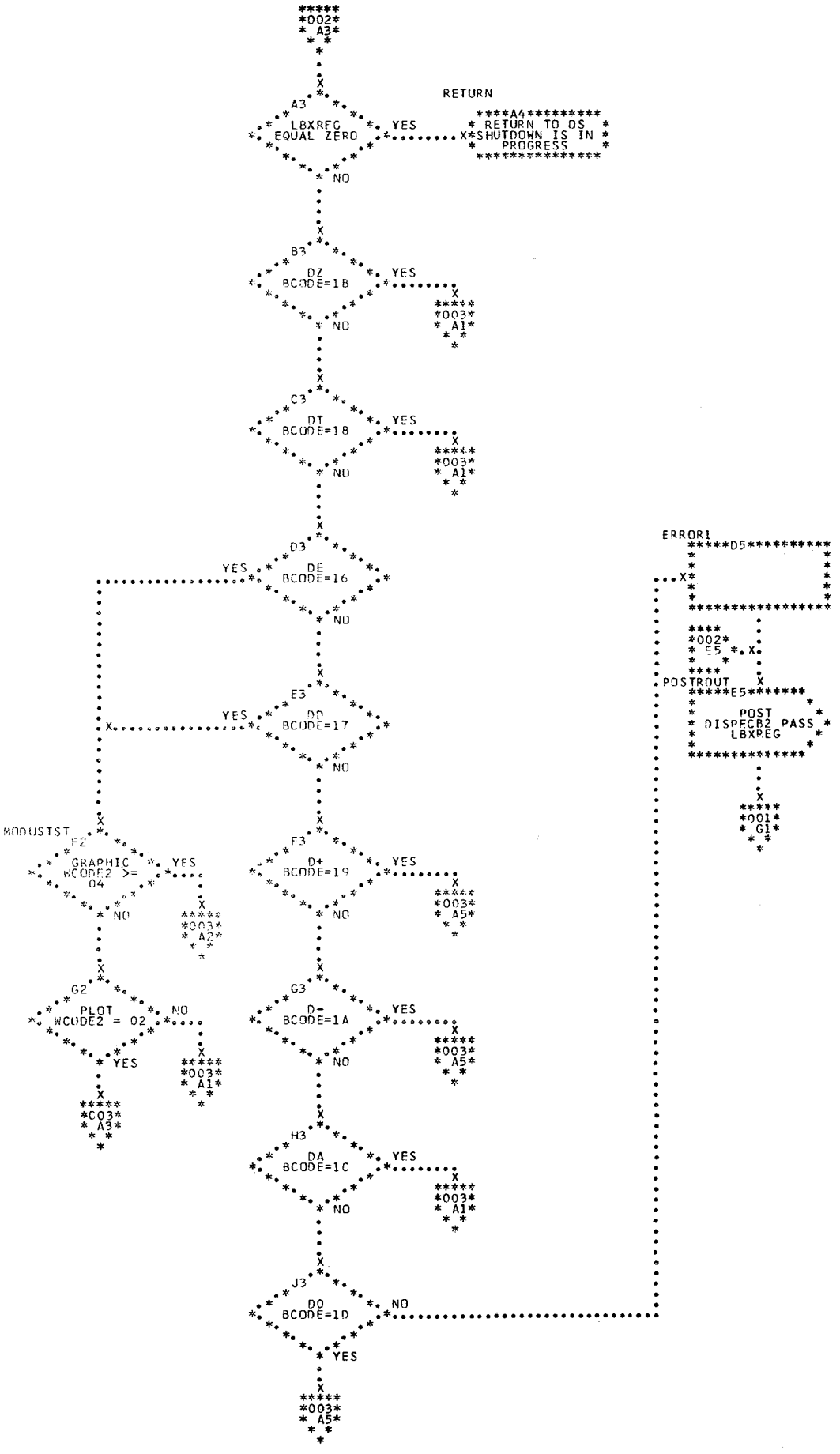

IOERROR
A1
RESET ERRORFLAGS IN PARTDCB
B1
IS IT NOT FIRST I/O AND IS BUFFER FULL
YES
NO
002 B3*
C1
FIRST I/O-ERROR
NO
YES
D1
REQBUF
GET 80-BYTE BUFFER OUT OF BUFFERPOOL
E1
IS A BUFFER AVAILABLE
NO
YES
F1
SET ERRORCODE IN LCC12 TO C'0C' AND IN LCC12 X'0602'
G1
STORE BUFFER ADDRESS IN LBX
H1
COMPUTE END ADDRESS OF BUFFER
HOLE
J1
MOVE MEMBER + BLANK NAME IN REQUESTED BUFFER
K1
COMPUTE NEW ADDRESS FOR NEXT MEMBER NAME
A2
A2
END OF BUFFER
YES
NO
BXX
B2
BLOCK WRITTEN TO USERS PDS
NO
YES
002 B3*
002 D2*
KEINP
D2
SET ERROR CODE IN LCC12 X'0603'
NICHTCAT
F2
SET ERRORCODE IN LCC12 TO C'FF01'
003 B1*
NOALLOCA
J2
SET ERRORCODE IN LCC12 TO C'FF00'
003 C2*
BUVOLL
A3
SET ERRORCODE IN LCC12 TO X'0601'
BUV
B3
SET SWITCH TO MARK BUFFER FULL
004 D3*
DDNOTF
D3
SET ERRORCODE IN LCC12 TO C'FF01' CATO X'08'
002 G3*
NOALLOCA
F3
RETURNCODE < 18
NO
YES
G3
SET ERRORCODE X'08'
H3
RETURNCODE
....0.....004 J2
....4.....004 A4
....8.....004 F4
....12....004 J4
....16....004 J4
ROOMVTOC
A4
LCC12 X'130' VTOC TOO SMALL
002 G3*
ERRCAT
C4
RETURNCODE
ELSE
4
D4
SET ERRORCODE IN LCC12 X'FF00' CATO X'08'
002 G3*
PERMIO
F4
SET ERRORCODE IN LCC12 TO 131
002 G3*
NOQUANT
J4
SET ERRORCODE TO 132 NO ALLOCATION
002 G3*
NOTOPEN
A5
RESTORE PARTDCB
004 C4*
IOERRCAT
C5
SET ERRORCODE IN LCC12 TO 129
003 H1*
004 E5*
ERROBT
E5
SET ERRORCODE IN LCC12 TO 119
F5
CLEAR ERRORFLAGS IN PARTDCB
G5
SET ERRORCODE IN CATO X'08'
002 G3*

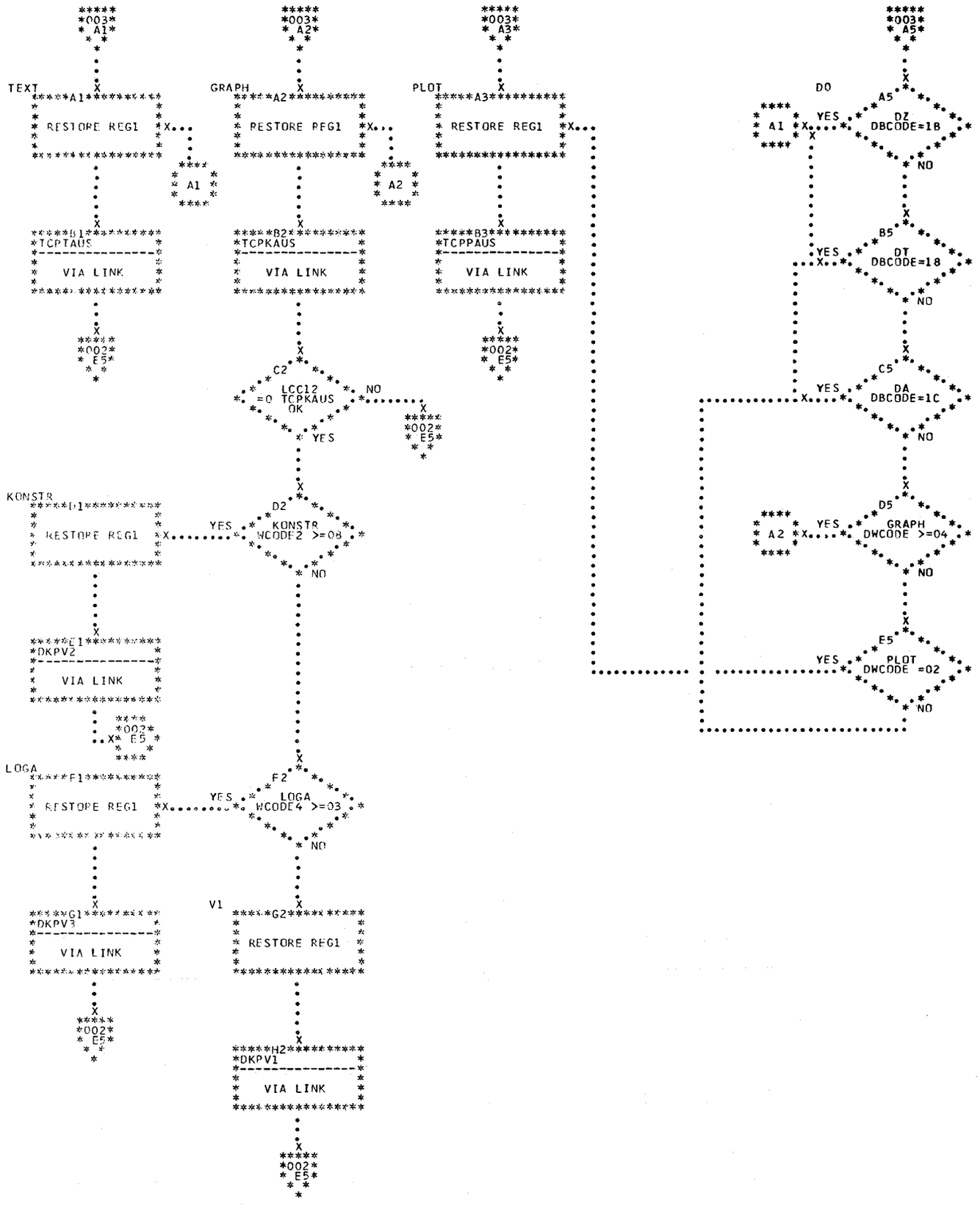

A1
ENTRY VIA ATTACH FROM TCPOPROG
B1
EXTRACT
GET ADDRESS TIOT
FTERM
C1
END OF TIOT
YES
NO
ERROR2
C2
MSG41: NO DD-STATEMENTS FOR DISPLAY
F3
C1
D1
'DIS' DDNAME
NO
YES
D2
GET ADDRESS OF NEXT TIOT ELEMENT
C1
E1
GET DEVICE-TYPE
ERROR3
F1
DEVICE-TYPE ERROR OR DUMMY DD
YES
NO
F2
MSG42: DD-ERROR
F3
CUCBS
G1
GET NUMBER OF CONTROL UNITS (CU)
H1
A54 001A5
GET AND CLEAR CORE FOR FIRST DBX
J1
DBXINIT
J1
SET KEY=0 INDICATE SP=252
K1
GET CORE FOR DEB OPEN DEB
A3
RESET KEY
B3
OPEN IOB AND DCB IN DBX
C3
WAS IT THE LAST DBX
NO
YES
C4
SET POINTER FOR NEXT DBX
D3
LOAD TCP1DTAB
D4
A54 001A5
GET AND CLEAR CORE FOR NEXT DBX
J1
E3
UPDATE LBX'S WITH NUMBER OF CU AND NUMBER OF DISPLAY
F3
ANY ERROR RECOVERED
NO
YES
ERROR1
F4
MSG40: ERROR IN TCP1DTAB
TABEND
G3
A54 001A5
GET AND CLEAR CORE FOR ECB-LIST
H3
UPDATE ECB-LIST
J3
GET DATA FOR DISPLAY- OUTPUT
K3
SAVE AREA-ADDRESS, CONTROLBYTE AND COUNT
002
AI
A54
A5
ENTER
B5
GETMAIN
GET CORE IN SUBPOOL 1
A58
C5
RESET IT TO ZERO
D5
RETURN

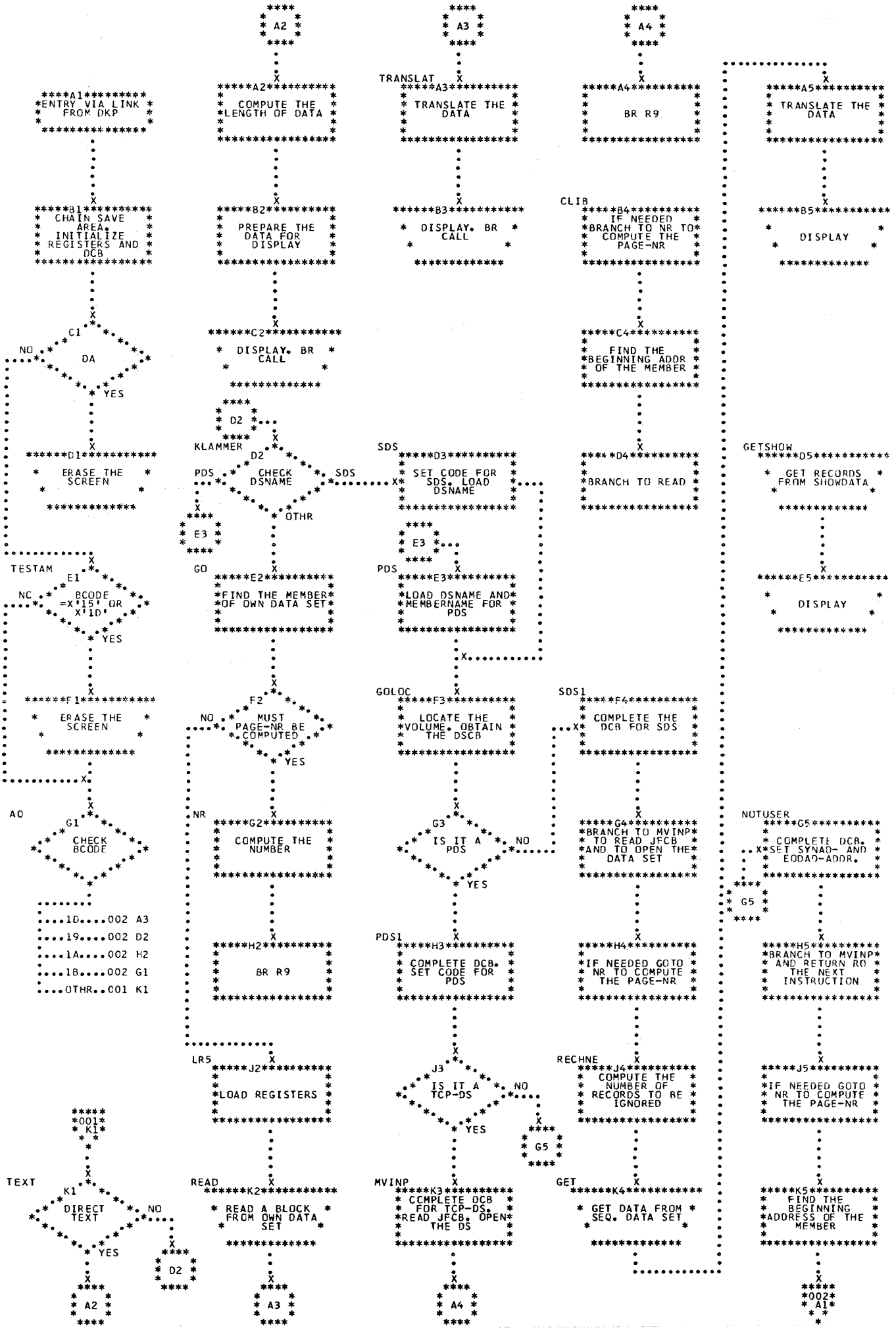

002
A1
CCWSTART
A1
UPDATE CCWS FOR DISPLAY OUTPUT
CCWLP1
B1
GET NUMBER OF DISPLAYS ON CU
CCWEXCP
C1
PESFT ECB
D1
EXCP START CHANNEL PROGRAM
E1
WAIT
WAIT FOR ECB
LOOPW1
F1
UPDATE CONTROLBYTE AND ERASEBYTE WITH ACTUAL DISNP
OK
G1
WAS CP CORRECTLY FINISHED
NO
A2
YES
GOON
H1
GET ADDRESS OF NEXT DISPLAY
J1
NEXT DISPLAY ON SAME CU
YES
NO
RETRY
A2
WAS IT THE THIRD RETRY
YES
NO
A2
B2
WAS CU NOT OPERATIONAL
NO
YES
B1
CCWEXCP
C2
RESET FLAGS REDUCE AND RETRY COUNT
ERROR4
A3
MSG43: CU*** CANNOT BE STARTED
B3
ERASE UNIT ADDRESS IN KBX AND LBX
C3
ANY CU OPERATIONAL
YES
NO
ERROR5
D3
MSG44: NO DISPLAY OPERATIONAL NO STARTUP
E3
FREEMAIN
RELEASE SUBPOOL 1 STORAGE
F3
ERRORTYP
F3
WTO ERROR MSG
G3
POST ECP FOR NO DISPLAY-STARTUP
H3
RETURN
STARTX
C4
GET ADDRESS OF NEXT DBX IN CHAIN
D4
IS IT KBX
NO
YES
B1
STARTEND
E4
FREEPOOL
RELEASE WORKAREAS
F4
A54 001A5
GET CORE FOR PARMLIST
G4
BUILT UP PARMLIST AS NECESSARY FOR PL/1 PROGRAMS
H4
XCTL EP=DKP

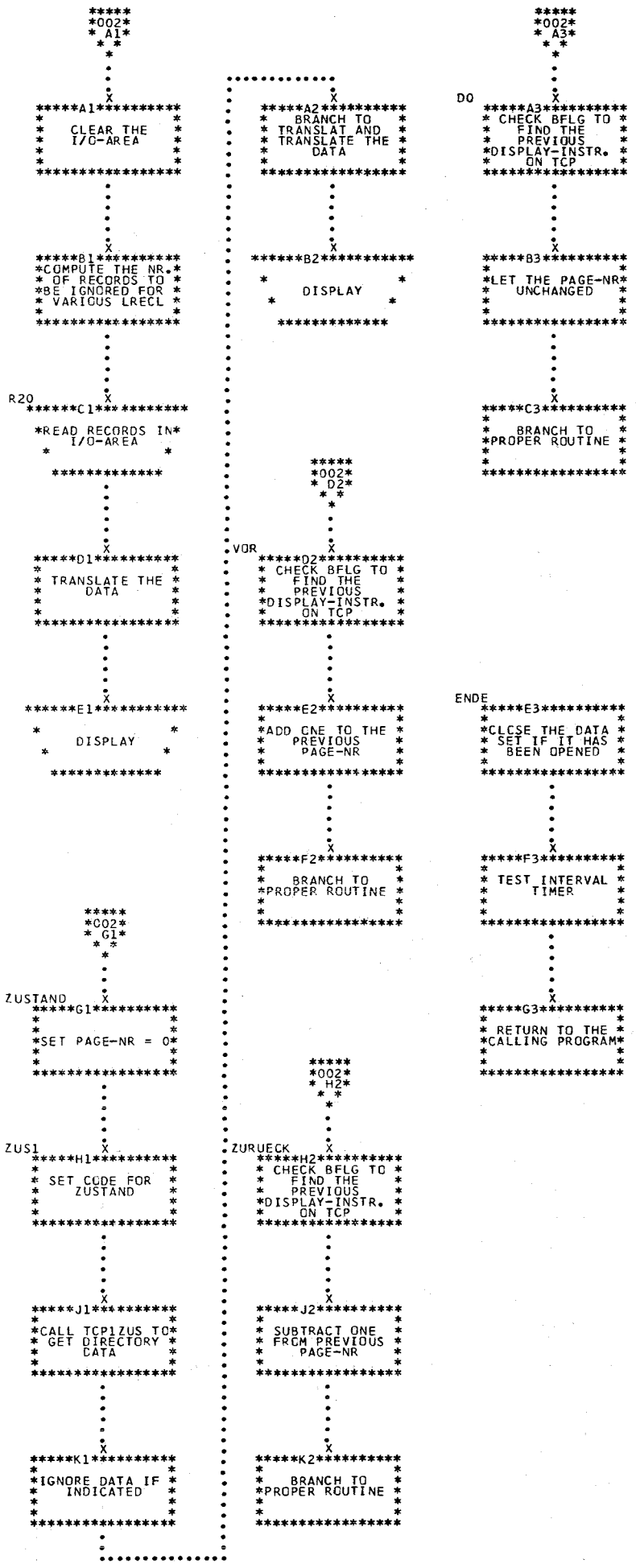
TCP1DKP

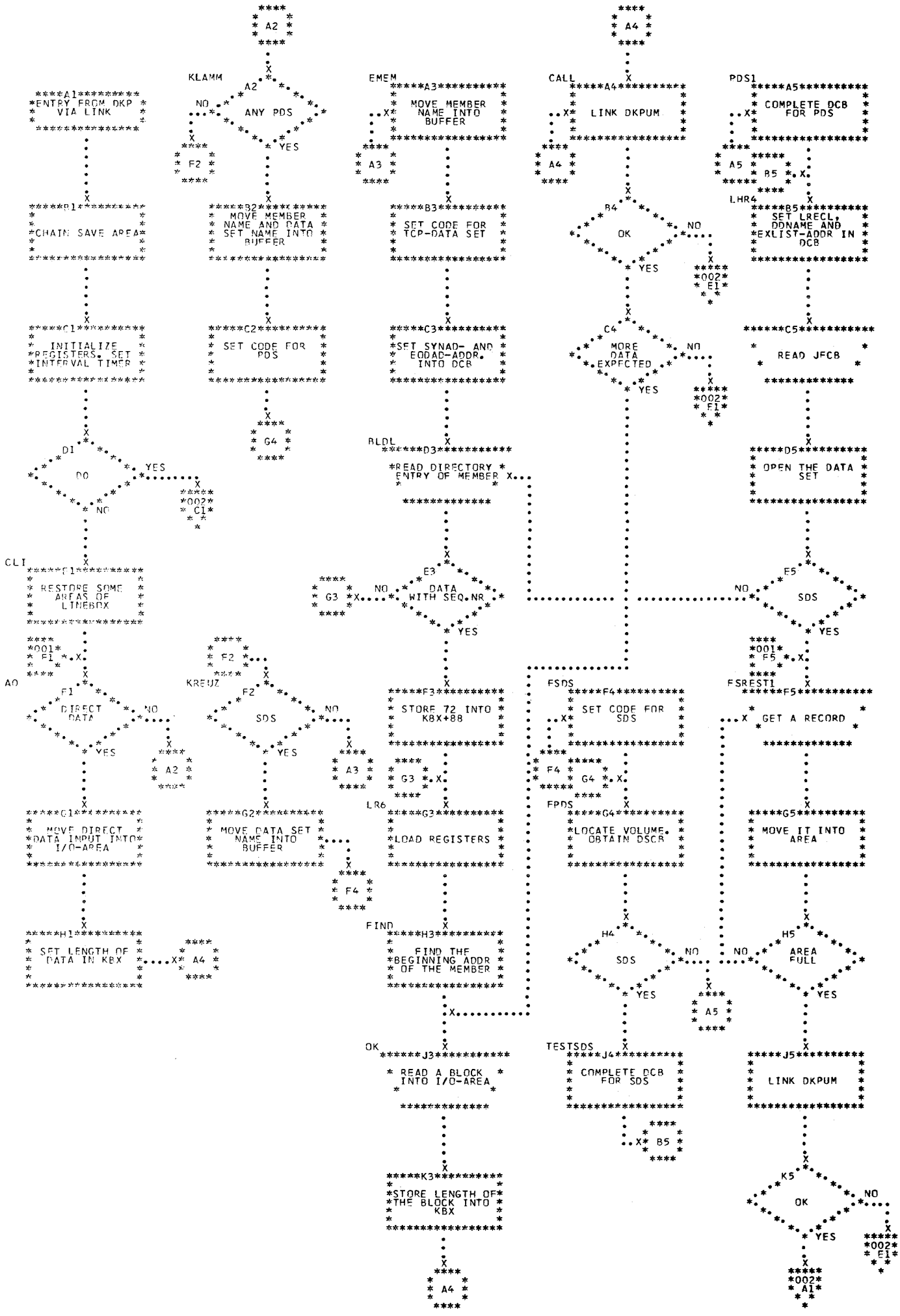

A1
ENTRY FROM TCP1DSUP VIA XCTL
R1
GET KONBOX ADDRESS VIA REG1 AND PARMLIST
C1
SAVE REG1
D1
STORE ADDRESS OF DISPLAY OUTPUT AREA INTO KONBOX
E1
BLDL FOR DISPLAY MODULES
F1
BLDL OK
NO
YES
BLDLERR
F2
POST DISPECB2 WITH COMPLETION CODE 1
WAIT
G1
WAIT DISPECB FOR A DISPLAY COMMAND
H1
GET ACTUAL LINEBOX VIA POSTED DISPECB
J1
CLEAR DISPECB
K1
INSERT ACTUAL LINEBOX ADDRESS INTO PARMLIST
002
A3

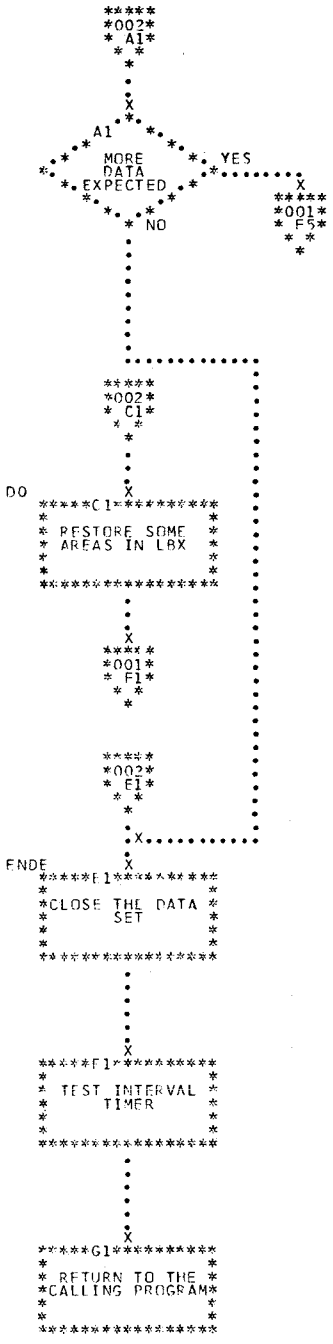

002
A3
LBXREG EQUAL ZERO
YES
RETURN
A4******
* RETURN TO OS *
SHUTDOWN IS IN PROGRESS
NO
B3
DZ BCODE=1B
YES
003
A1
NO
C3
DT BCODE=18
YES
003
A1
NO
D3
DE BCODE=16
YES
NO
E3
DD BCODE=17
YES
NO
MODUSTST
F2
GRAPHIC WCODE2 >= 04
YES
003
A2
NO
G2
PLOT WCODE2 = 02
NO
003
A1
YES
C03
A3
F3
D+ BCODE=19
YES
003
A5
NO
G3
D- BCODE=1A
YES
003
A5
NO
H3
DA BCODE=1C
YES
003
A1
NO
J3
DO BCODE=1D
NO
YES
003
A5
ERROR1
*****D5**********
002
E5
POSTROUT
*****E5*******
POST DISPECB2 PASS LBXREG
001
G1

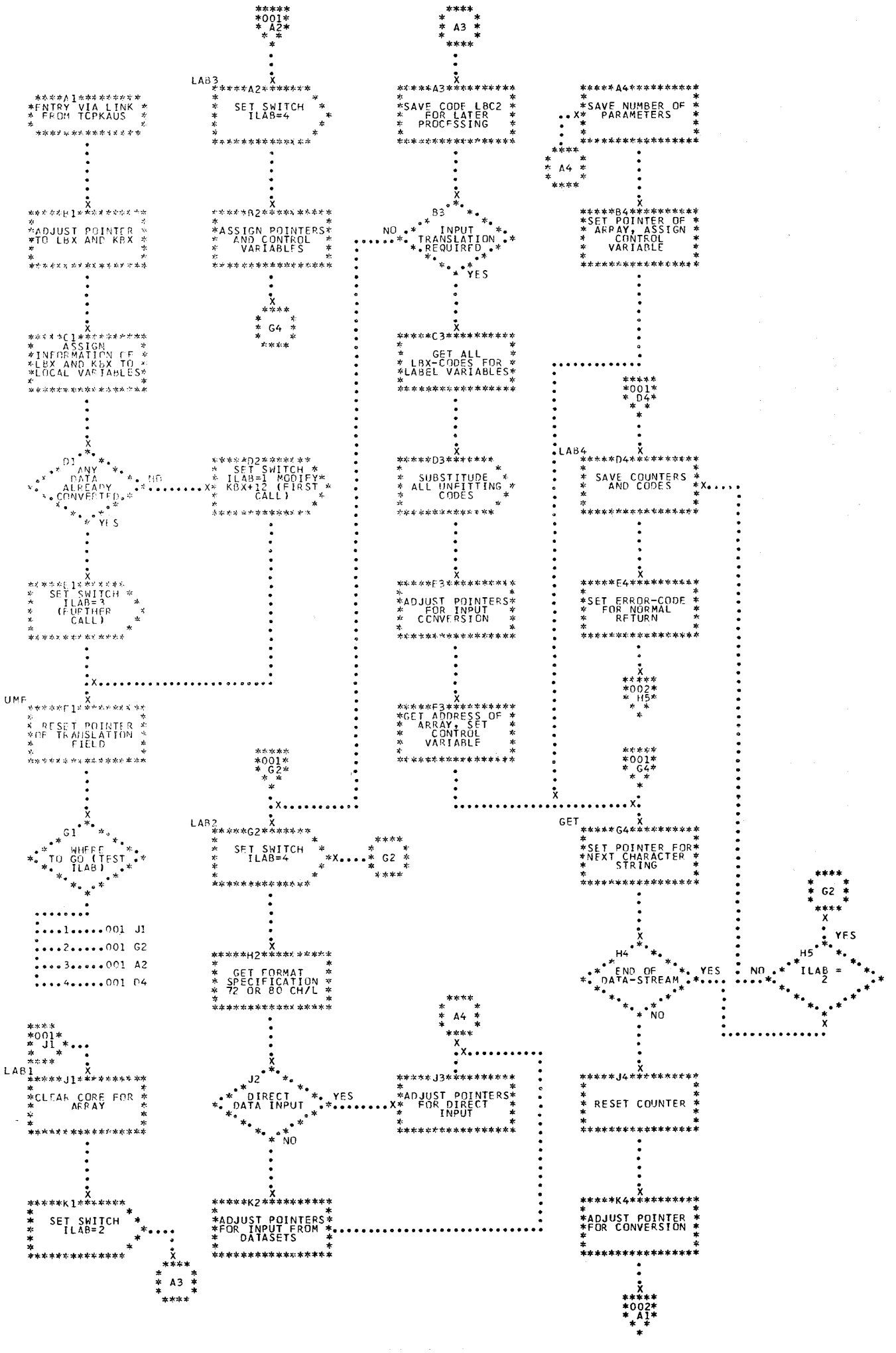

003 A1
TEXT
A1
RESTORE REG1
A1
B1
TCPTAUS
VIA LINK
002 E5
003 A2
GRAPH
A2
RESTORE REG1
A2
B2
TCPKAUS
VIA LINK
C2
LCC12 =0 TCPKAUS OK
NO
YES
002 E5
D2
KONSTR WCODE2 >=08
YES
NO
KONSTR
RESTORE REG1
E1
DKPV2
VIA LINK
002 E5
F2
LOGA WCODE4 >=03
YES
NO
LOGA
RESTORE REG1
G1
DKPV3
VIA LINK
002 E5
V1
G2
RESTORE REG1
H2
DKPV1
VIA LINK
002 E5
003 A3
PLOT
A3
RESTORE REG1
B3
TCPPAUS
VIA LINK
002 E5
003 A5
DO
A1
A5
DZ DBCODE=1B
YES
NO
B5
DT DBCODE=18
YES
NO
C5
DA DBCODE=1C
YES
NO
D5
A2
GRAPH DWCODE >=04
YES
NO
E5
PLOT DWCODE =02
YES
NO

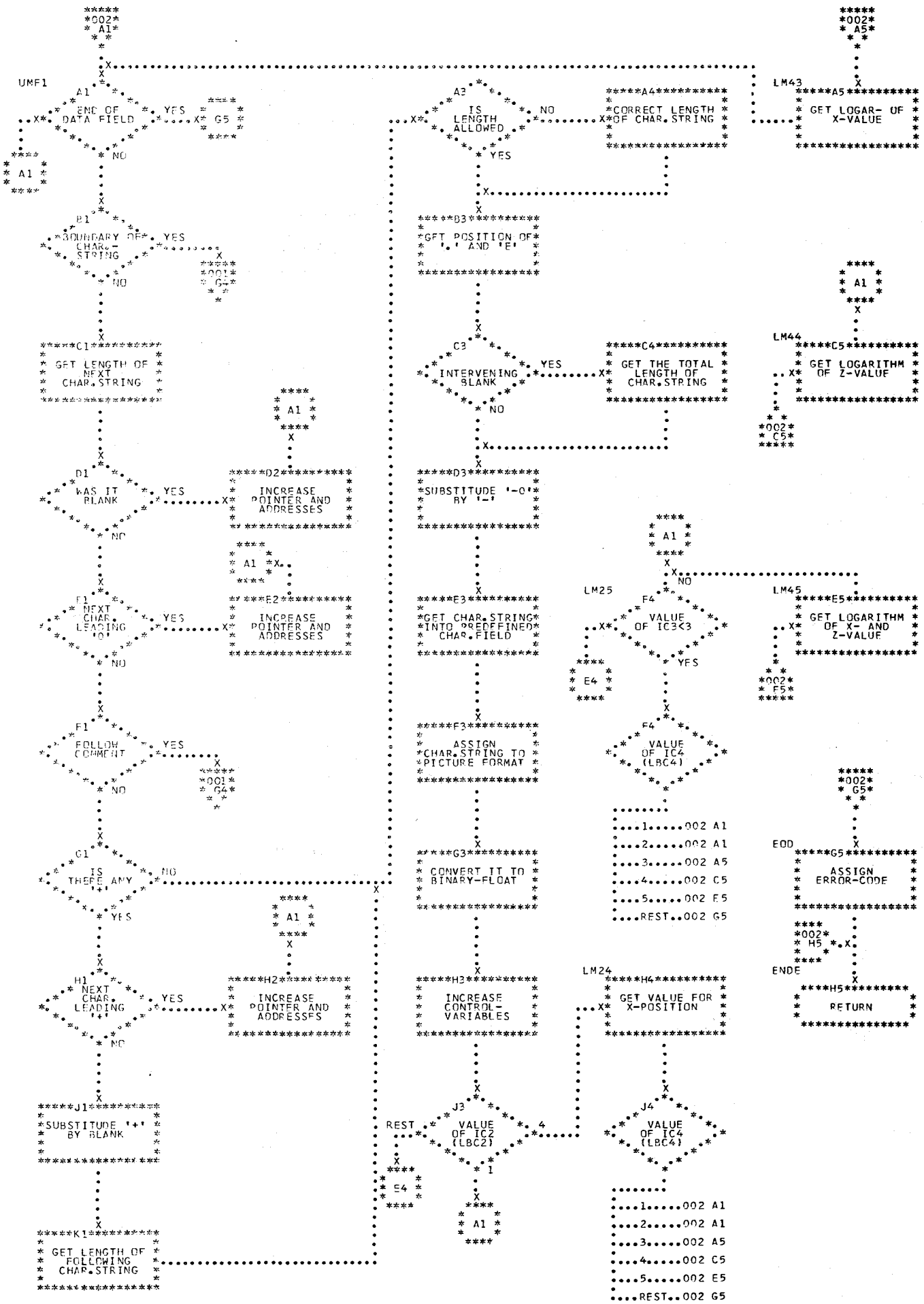

A1
ENTRY VIA LINK FROM DKP
B1
CHAIN SAVE AREA, INITIALIZE REGISTERS AND DCB
C1
DA
NO
YES
D1
ERASE THE SCREEN
TESTAM
E1
BCODE =X'15' OR X'1D'
NC
YES
F1
ERASE THE SCREEN
A0
G1
CHECK BCODE
1D....002 A3
19....002 D2
1A....002 H2
1B....002 G1
OTHR..C01 K1
A2
A2
A2
COMPUTE THE LENGTH OF DATA
B2
PREPARE THE DATA FOR DISPLAY
C2
DISPLAY. BR CALL
D2
KLAMMER
D2
CHECK DSNAME
PDS
SDS
OTHR
E3
GO
E2
FIND THE MEMBER OF OWN DATA SET
F2
MUST PAGE-NR BE COMPUTED
NO
YES
NR
G2
COMPUTE THE NUMBER
H2
BR R9
LR5
J2
LOAD REGISTERS
READ
K2
READ A BLOCK FROM OWN DATA SET
A3
A3
TRANSLAT
A3
TRANSLATE THE DATA
B3
DISPLAY. BR CALL
SDS
D3
SET CODE FOR SDS. LOAD DSNAME
E3
PDS
E3
LOAD DSNAME AND MEMBERNAME FOR PDS
GOLOC
F3
LOCATE THE VOLUME. OBTAIN THE DSCB
G3
IS IT A PDS
NO
YES
PDS1
H3
COMPLETE DCB. SET CODE FOR PDS
J3
IS IT A TCP-DS
NO
YES
G5
MVINP
K3
COMPLETE DCB FOR TCP-DS. READ JFCB. OPEN THE DS
A4
A4
A4
BR R9
CLIB
B4
IF NEEDED COMPUTE THE BRANCH TO NR TO COMPUTE THE PAGE-NR
C4
FIND THE BEGINNING ADDR OF THE MEMBER
D4
BRANCH TO READ
SDS1
F4
COMPLETE THE DCB FOR SDS
G4
BRANCH TO MVINP TO READ JFCB AND TO OPEN THE DATA SET
H4
IF NEEDED GOTO NR TO COMPUTE THE PAGE-NR
RECHNE
J4
COMPUTE THE NUMBER OF RECORDS TO BE IGNORED
GET
K4
GET DATA FROM SEQ. DATA SET
A5
A5
TRANSLATE THE DATA
B5
DISPLAY
GETSHOW
D5
GET RECORDS FROM SHOWDATA
E5
DISPLAY
NOTUSER
G5
COMPLETE DCB. SET SYNAD- AND EODAD-ADDR.
G5
H5
BRANCH TO MVINP AND RETURN RO THE NEXT INSTRUCTION
J5
IF NEEDED GOTO NR TO COMPUTE THE PAGE-NR
K5
FIND THE BEGINNING ADDRESS OF THE MEMBER
A1
TEXT
K1
DIRECT TEXT
NO
YES
D2
001
K1
002
A1

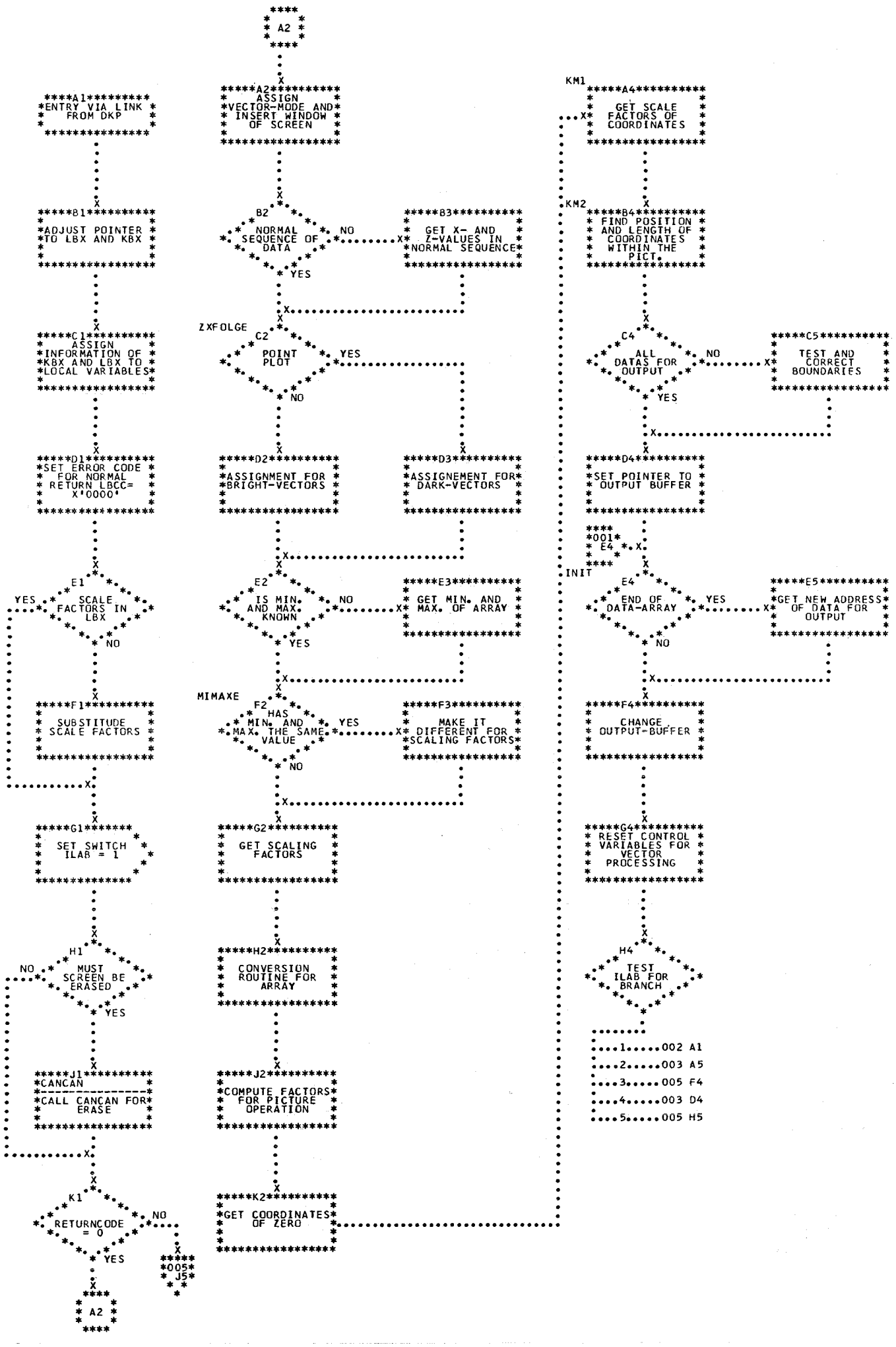

002 A1
A1 CLEAR THE I/O-AREA
B1 COMPUTE THE NR. OF RECORDS TO BE IGNORED FOR VARIOUS LRECL
R20
C1 READ RECORDS IN I/O-AREA
D1 TRANSLATE THE DATA
E1 DISPLAY
002 G1
ZUSTAND
G1 SET PAGE-NR = 0
ZUS1
H1 SET CODE FOR ZUSTAND
J1 CALL TCP1ZUS TO GET DIRECTORY DATA
K1 IGNORE DATA IF INDICATED
002 A2
A2 BRANCH TO TRANSLAT AND TRANSLATE THE DATA
B2 DISPLAY
002 D2
VOR
D2 CHECK BFLG TO FIND THE PREVIOUS DISPLAY-INSTR. ON TCP
E2 ADD ONE TO THE PREVIOUS PAGE-NR
F2 BRANCH TO PROPER ROUTINE
002 H2
ZURUECK
H2 CHECK BFLG TO FIND THE PREVIOUS DISPLAY-INSTR. ON TCP
J2 SUBTRACT ONE FROM PREVIOUS PAGE-NR
K2 BRANCH TO PROPER ROUTINE
002 A3
DO
A3 CHECK BFLG TO FIND THE PREVIOUS DISPLAY-INSTR. ON TCP
B3 LET THE PAGE-NR UNCHANGED
C3 BRANCH TO PROPER ROUTINE
ENDE
E3 CLOSE THE DATA SET IF IT HAS BEEN OPENED
F3 TEST INTERVAL TIMER
G3 RETURN TO THE CALLING PROGRAM

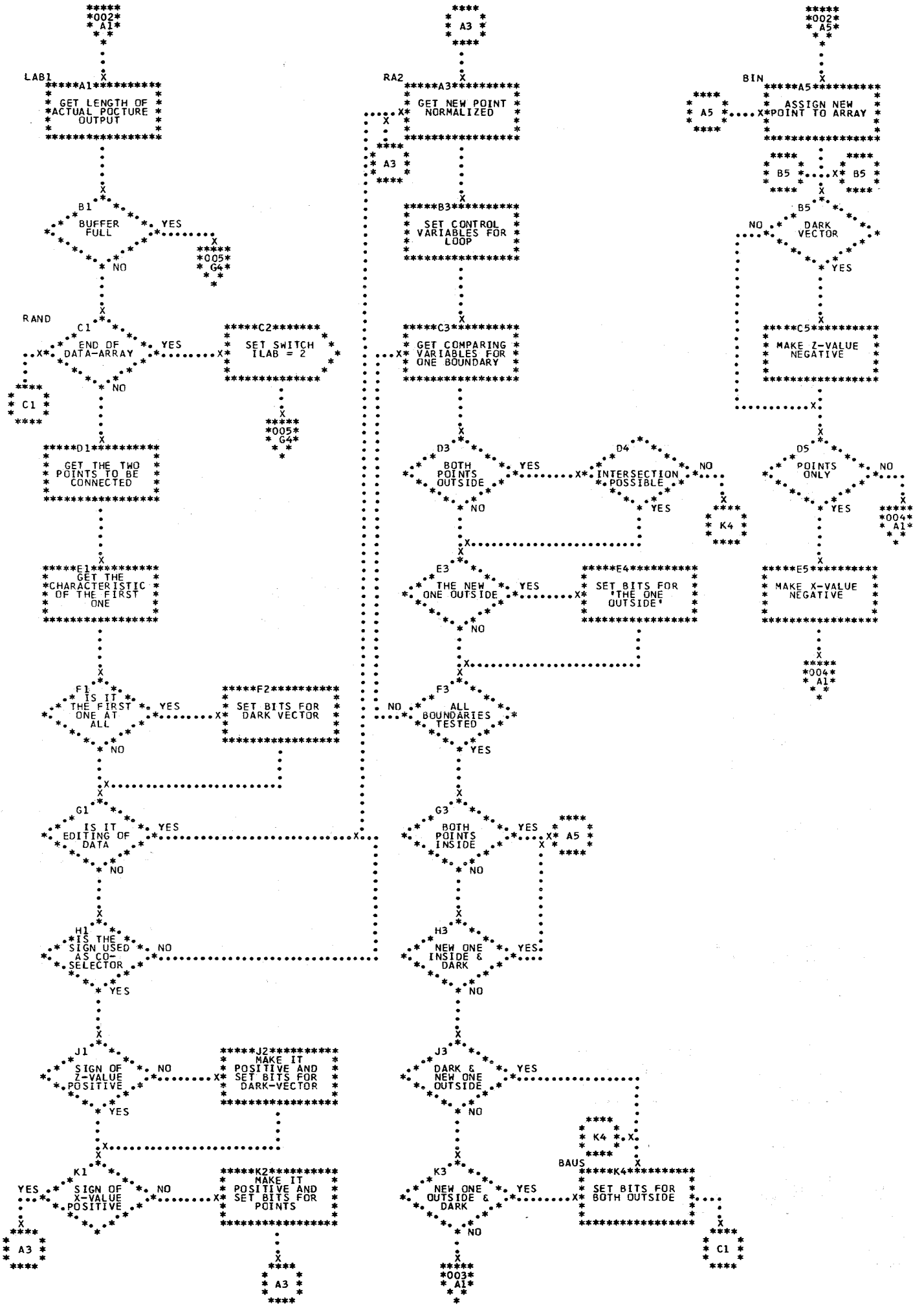

A1
ENTRY FROM DKP VIA LINK
B1
CHAIN SAVE AREA
C1
INITIALIZE REGISTERS, SET INTERVAL TIMER
D1
DO
NO
YES
002 C1
CLI
F1
RESTORE SOME AREAS OF LINEBOX
001 F1
A0
G1
MOVE DIRECT DATA INPUT INTO I/O-AREA
H1
SET LENGTH OF DATA IN KBX
A4
F1
DIRECT DATA
NO
YES
A2
KLAMM
A2
ANY PDS
NO
YES
F2
B2
MOVE MEMBER NAME AND DATA SET NAME INTO BUFFER
C2
SET CODE FOR PDS
G4
KREUZ
F2
SDS
NO
YES
A3
G2
MOVE DATA SET NAME INTO BUFFER
F4
EMEM
A3
MOVE MEMBER NAME INTO BUFFER
B3
SET CODE FOR TCP-DATA SET
C3
SET SYNAD- AND EODAD-ADDR. INTO DCB
RLDL
D3
READ DIRECTORY ENTRY OF MEMBER
E3
DATA WITH SEQ.NR.
NO
YES
G3
F3
STORE 72 INTO KBX+88
A3
G3
LR6
G3
LOAD REGISTERS
FIND
H3
FIND THE BEGINNING ADDR OF THE MEMBER
OK
J3
READ A BLOCK INTO I/O-AREA
K3
STORE LENGTH OF THE BLOCK INTO KBX
A4
A4
CALL
A4
LINK DKPUM
B4
OK
NO
YES
002 E1
C4
MORE DATA EXPECTED
NO
YES
002 E1
FSDS
F4
SET CODE FOR SDS
F4
G4
FPDS
G4
LOCATE VOLUME, OBTAIN DSCR
H4
SDS
NO
YES
A5
TESTSDS
J4
COMPLETE DCB FOR SDS
B5
PDS1
A5
COMPLETE DCB FOR PDS
A5
B5
LHR4
B5
SET LRECL, DDNAME AND EXLIST-ADDR IN DCB
C5
READ JFCB
D5
OPEN THE DATA SET
E5
SDS
NO
YES
001 E5
FSRESTL
F5
GET A RECORD
G5
MOVE IT INTO AREA
H5
AREA FULL
NO
YES
A5
J5
LINK DKPUM
K5
OK
NO
YES
002 E1
002 A1

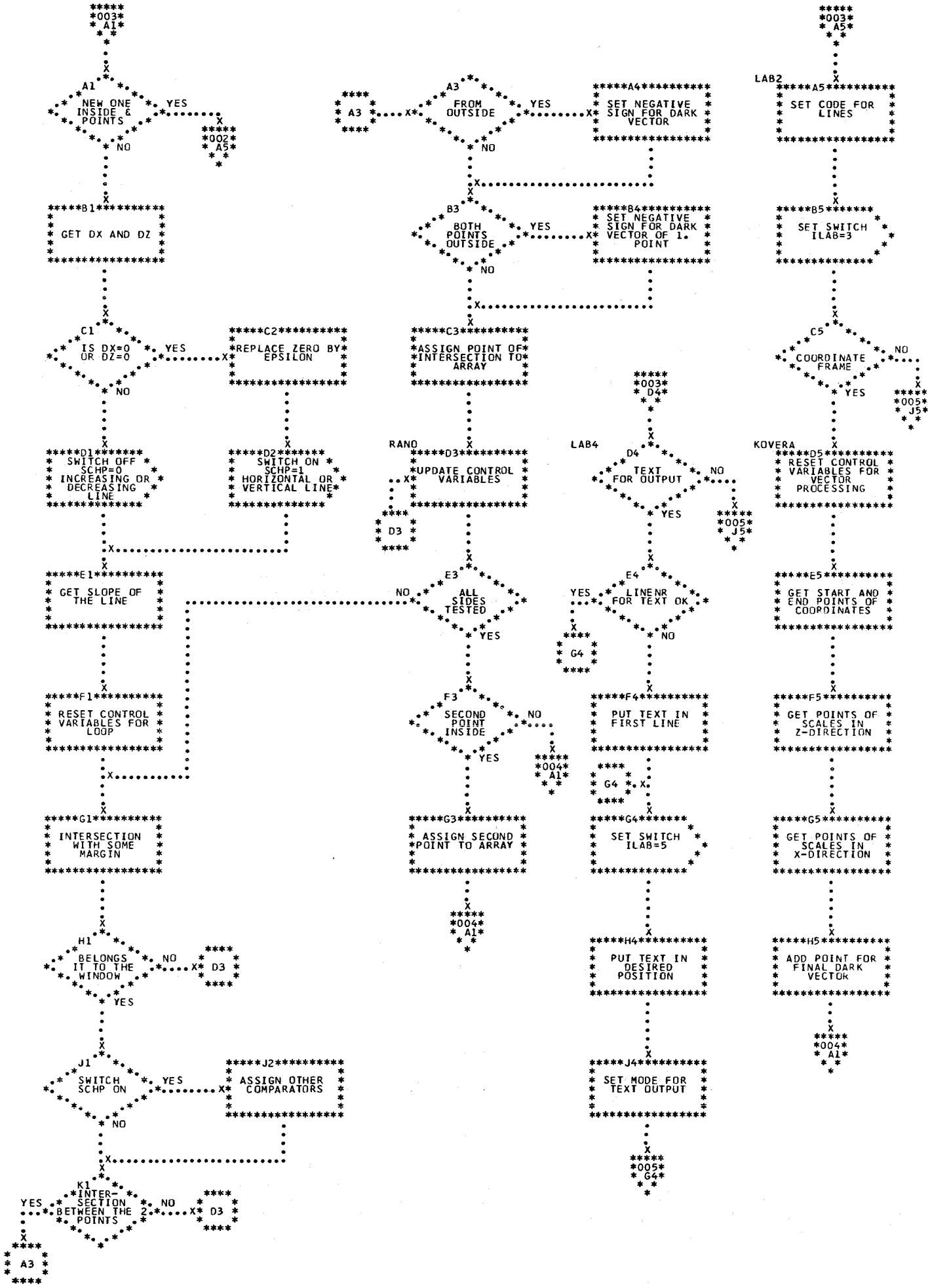

002
A1
A1
MORE DATA EXPECTED
YES
NO
001
F5
002
C1
DO
C1
RESTORE SOME AREAS IN LBX
001
F1
002
E1
X
ENDE
F1
CLOSE THE DATA SET
F1
TEST INTERVAL TIMER
G1
RETURN TO THE CALLING PROGRAM

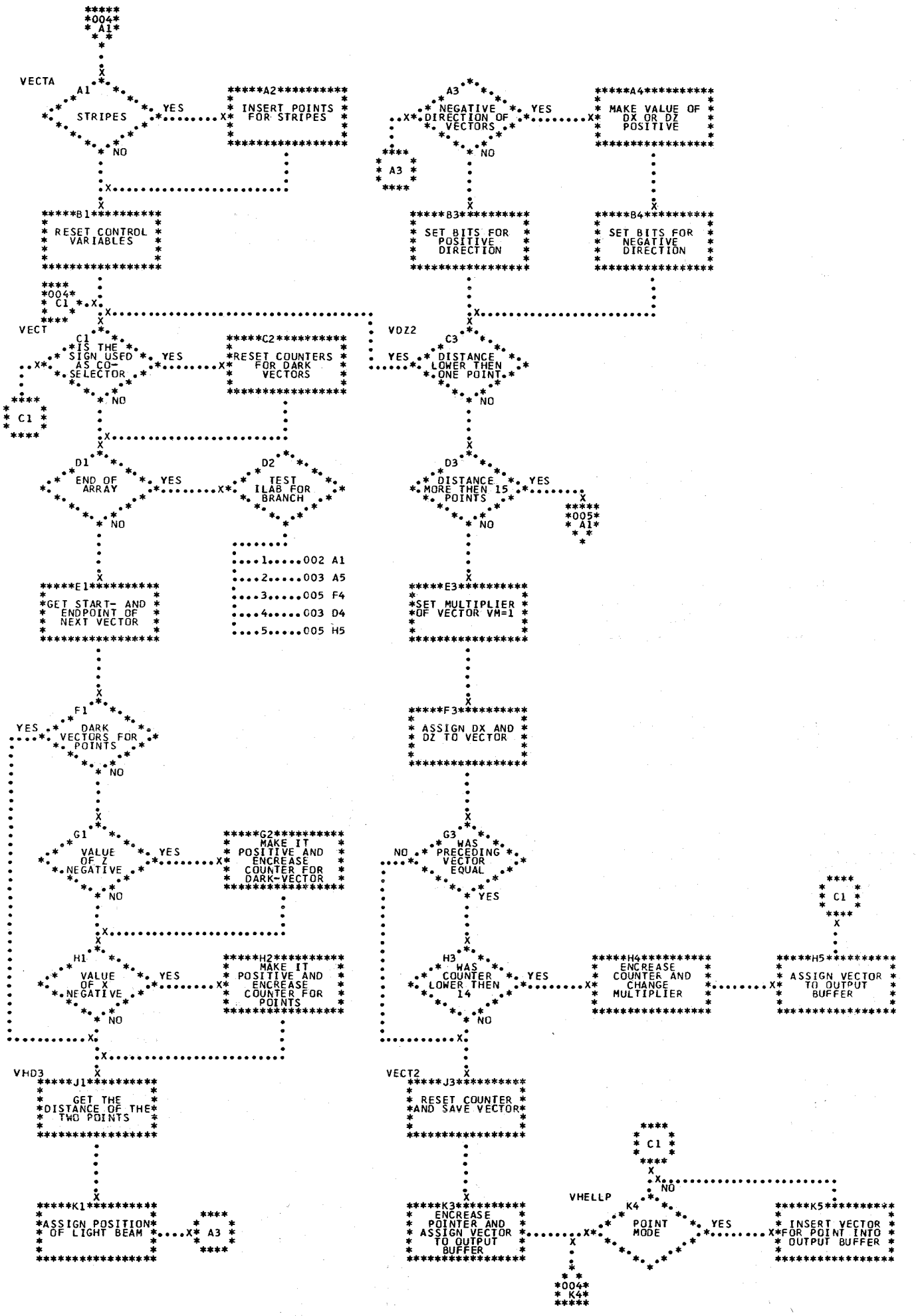

A1
ENTRY VIA LINK FROM TCPKAUS
A2
SET SWITCH ILAB=4
LAB3
A3
SAVE CODE LBC2 FOR LATER PROCESSING
A4
SAVE NUMBER OF PARAMETERS
B1
ADJUST POINTER TO LBX AND KBX
B2
ASSIGN POINTERS AND CONTROL VARIABLES
B3
INPUT TRANSLATION REQUIRED
NO
YES
B4
SET POINTER OF ARRAY, ASSIGN CONTROL VARIABLE
G4
C1
ASSIGN INFORMATION OF LBX AND KBX TO LOCAL VARIABLES
C3
GET ALL LBX-CODES FOR LABEL VARIABLES
D1
ANY DATA ALREADY CONVERTED
NO
YES
D2
SET SWITCH ILAB=1 MODIFY KBX+12 (FIRST CALL)
D3
SUBSTITUDE ALL UNFITTING CODES
D4
LAB4
SAVE COUNTERS AND CODES
E1
SET SWITCH ILAB=3 (FURTHER CALL)
E3
ADJUST POINTERS FOR INPUT CONVERSION
E4
SET ERROR-CODE FOR NORMAL RETURN
UME
F1
RESET POINTER OF TRANSLATION FIELD
F3
GET ADDRESS OF ARRAY, SET CONTROL VARIABLE
001
A2
G2
002
H5
G1
WHERE TO GO (TEST ILAB)
1 001 J1
2 001 G2
3 001 A2
4 001 D4
LAB2
G2
SET SWITCH ILAB=4
GET
G4
SET POINTER FOR NEXT CHARACTER STRING
H2
GET FORMAT SPECIFICATION 72 OR 80 CH/L
H4
END OF DATA-STREAM
YES
NO
H5
ILAB = 2
YES
NO
J1
LAB1
CLEAR CORE FOR ARRAY
J2
DIRECT DATA INPUT
YES
NO
J3
ADJUST POINTERS FOR DIRECT INPUT
J4
RESET COUNTER
K1
SET SWITCH ILAB=2
K2
ADJUST POINTERS FOR INPUT FROM DATASETS
K4
ADJUST POINTER FOR CONVERSION
AI

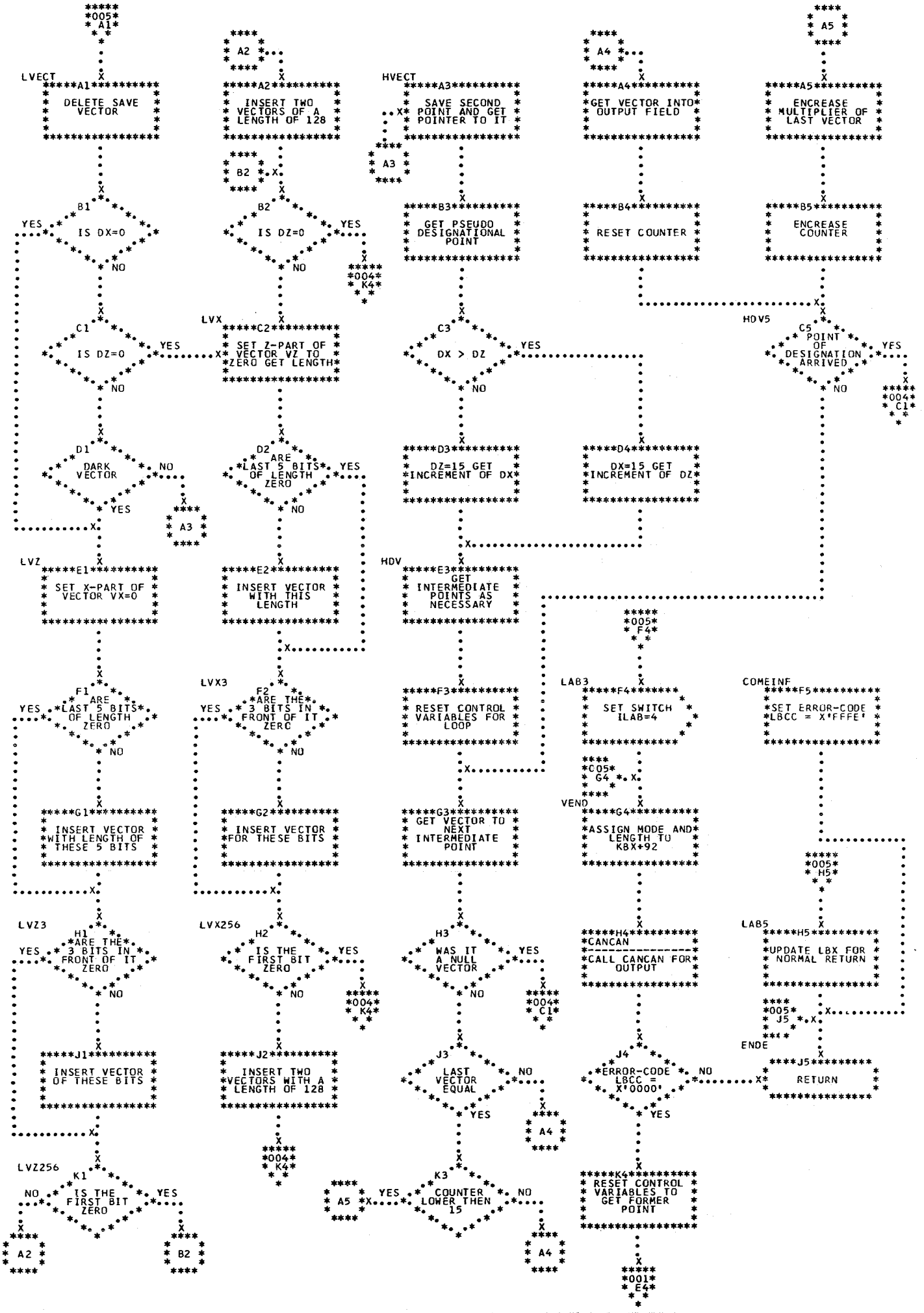

UMF1
A1 END OF DATA FIELD
YES
NO
G5
002 A1
B1 BOUNDARY OF CHAR.-STRING
001 G4
C1 GET LENGTH OF NEXT CHAR.STRING
D1 WAS IT BLANK
D2 INCREASE POINTER AND ADDRESSES
E1 NEXT CHAR. LEADING '0'
E2 INCREASE POINTER AND ADDRESSES
F1 FOLLOW COMMENT
G1 IS THERE ANY '+'
H1 NEXT CHAR. LEADING '+'
H2 INCREASE POINTER AND ADDRESSES
J1 SUBSTITUTE '+' BY BLANK
K1 GET LENGTH OF FOLLOWING CHAR.STRING
A2 IS LENGTH ALLOWED
A4 CORRECT LENGTH OF CHAR.STRING
LM43
A5 GET LOGAR- OF X-VALUE
002 A5
B3 GET POSITION OF '.' AND 'E'
C3 INTERVENING BLANK
C4 GET THE TOTAL LENGTH OF CHAR.STRING
LM44
C5 GET LOGARITHM OF Z-VALUE
002 C5
D3 SUBSTITUDE '-0' BY '-'
E3 GET CHAR.STRING INTO PREDEFINED CHAR.FIELD
F3 ASSIGN CHAR.STRING TO PICTURE FORMAT
G3 CONVERT IT TO BINARY-FLOAT
H3 INCREASE CONTROL-VARIABLES
J3 VALUE OF IC2 (LBC2)
REST
1
4
E4
LM25
F4 VALUE OF IC3<3
YES
NO
F4 VALUE OF IC4 (LBC4)
1.....002 A1
2.....002 A1
3.....002 A5
4.....002 C5
5.....002 E5
REST..002 G5
LM45
E5 GET LOGARITHM OF X- AND Z-VALUE
002 E5
002 G5
EOD
G5 ASSIGN ERROR-CODE
002 H5
ENDE
H5 RETURN
LM24
H4 GET VALUE FOR X-POSITION
J4 VALUE OF IC4 (LBC4)
1.....002 A1
2.....002 A1
3.....002 A5
4.....002 C5
5.....002 E5
REST..002 G5

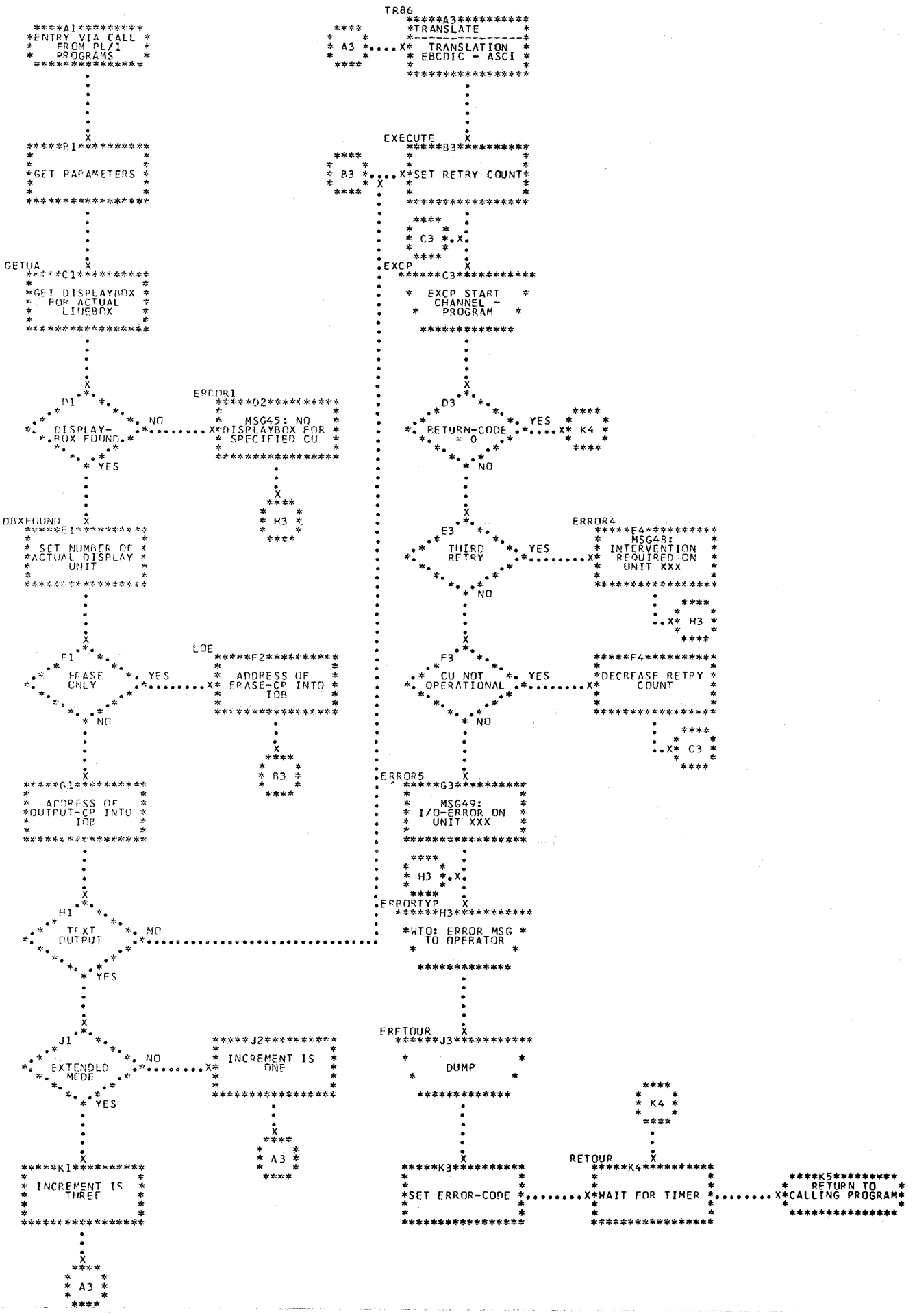

A2
A1 ENTRY VIA LINK FROM DKP
B1 ADJUST POINTER TO LBX AND KBX
C1 ASSIGN INFORMATION OF KBX AND LBX TO LOCAL VARIABLES
D1 SET ERROR CODE FOR NORMAL RETURN LBCC= X'0000'
E1 SCALE FACTORS IN LBX
YES
NO
F1 SUBSTITUDE SCALE FACTORS
G1 SET SWITCH ILAB = 1
H1 MUST SCREEN BE ERASED
J1 CANCAN CALL CANCAN FOR ERASE
K1 RETURNCODE = 0
NO
YES
A2
005 J5
A2 ASSIGN VECTOR-MODE AND INSERT WINDOW OF SCREEN
B2 NORMAL SEQUENCE OF DATA
NO
YES
B3 GET X- AND Z-VALUES IN NORMAL SEQUENCE
ZXFOLGE
C2 POINT PLOT
YES
NO
D2 ASSIGNMENT FOR BRIGHT-VECTORS
D3 ASSIGNEMENT FOR DARK-VECTORS
E2 IS MIN. AND MAX. KNOWN
NO
YES
E3 GET MIN. AND MAX. OF ARRAY
MIMAXE
F2 HAS MIN. AND MAX. THE SAME VALUE
YES
NO
F3 MAKE IT DIFFERENT FOR SCALING FACTORS
G2 GET SCALING FACTORS
H2 CONVERSION ROUTINE FOR ARRAY
J2 COMPUTE FACTORS FOR PICTURE OPERATION
K2 GET COORDINATES OF ZERO
KM1
A4 GET SCALE FACTORS OF COORDINATES
KM2
B4 FIND POSITION AND LENGTH OF COORDINATES WITHIN THE PICT.
C4 ALL DATAS FOR OUTPUT
NO
YES
C5 TEST AND CORRECT BOUNDARIES
D4 SET POINTER TO OUTPUT BUFFER
001 E4
INIT
E4 END OF DATA-ARRAY
YES
NO
E5 GET NEW ADDRESS OF DATA FOR OUTPUT
F4 CHANGE OUTPUT=BUFFER
G4 RESET CONTROL VARIABLES FOR VECTOR PROCESSING
H4 TEST ILAB FOR BRANCH
1 002 A1
2 003 A5
3 005 F4
4 003 D4
5 005 H5

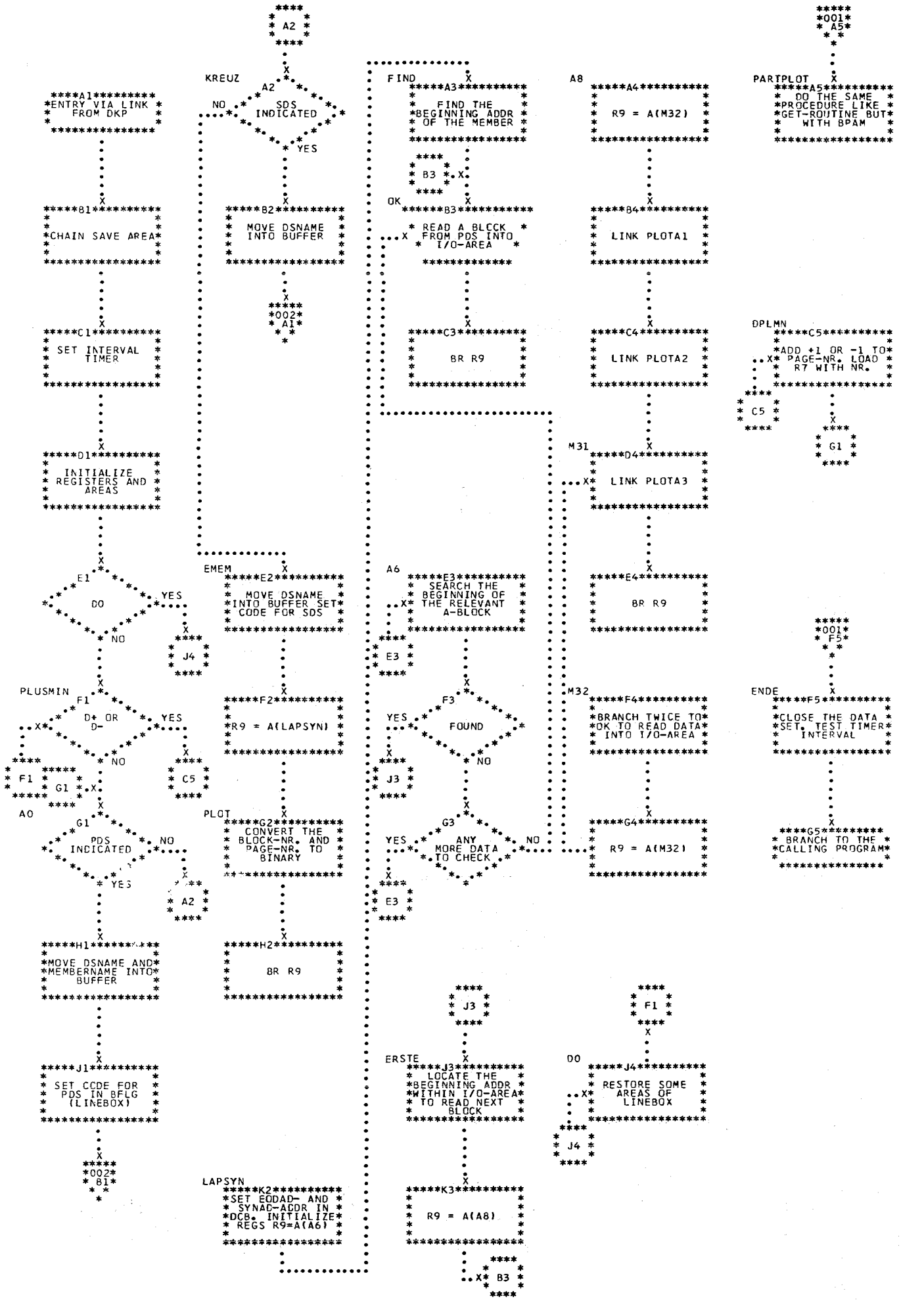

002 A1
LAB1
A1 GET LENGTH OF ACTUAL POCTURE OUTPUT
B1 BUFFER FULL
YES
NO
005 G4
RAND
C1 END OF DATA-ARRAY
C2 SET SWITCH ILAB = 2
005 G4
C1
D1 GET THE TWO POINTS TO BE CONNECTED
E1 GET THE CHARACTERISTIC OF THE FIRST ONE
F1 IS IT THE FIRST ONE AT ALL
F2 SET BITS FOR DARK VECTOR
G1 IS IT EDITING OF DATA
H1 IS THE SIGN USED AS CO-SELECTOR
J1 SIGN OF Z-VALUE POSITIVE
J2 MAKE IT POSITIVE AND SET BITS FOR DARK-VECTOR
K1 SIGN OF X-VALUE POSITIVE
K2 MAKE IT POSITIVE AND SET BITS FOR POINTS
A3
A3
RA2
A3 GET NEW POINT NORMALIZED
B3 SET CONTROL VARIABLES FOR LOOP
C3 GET COMPARING VARIABLES FOR ONE BOUNDARY
D3 BOTH POINTS OUTSIDE
D4 INTERSECTION POSSIBLE
K4
E3 THE NEW ONE OUTSIDE
E4 SET BITS FOR 'THE ONE OUTSIDE'
F3 ALL BOUNDARIES TESTED
G3 BOTH POINTS INSIDE
A5
H3 NEW ONE INSIDE & DARK
J3 DARK & NEW ONE OUTSIDE
K3 NEW ONE OUTSIDE & DARK
BAUS
K4 SET BITS FOR BOTH OUTSIDE
C1
003 A1
002 A5
BIN
A5 ASSIGN NEW POINT TO ARRAY
B5
B5 DARK VECTOR
C5 MAKE Z-VALUE NEGATIVE
D5 POINTS ONLY
004 A1
E5 MAKE X-VALUE NEGATIVE
004 A1

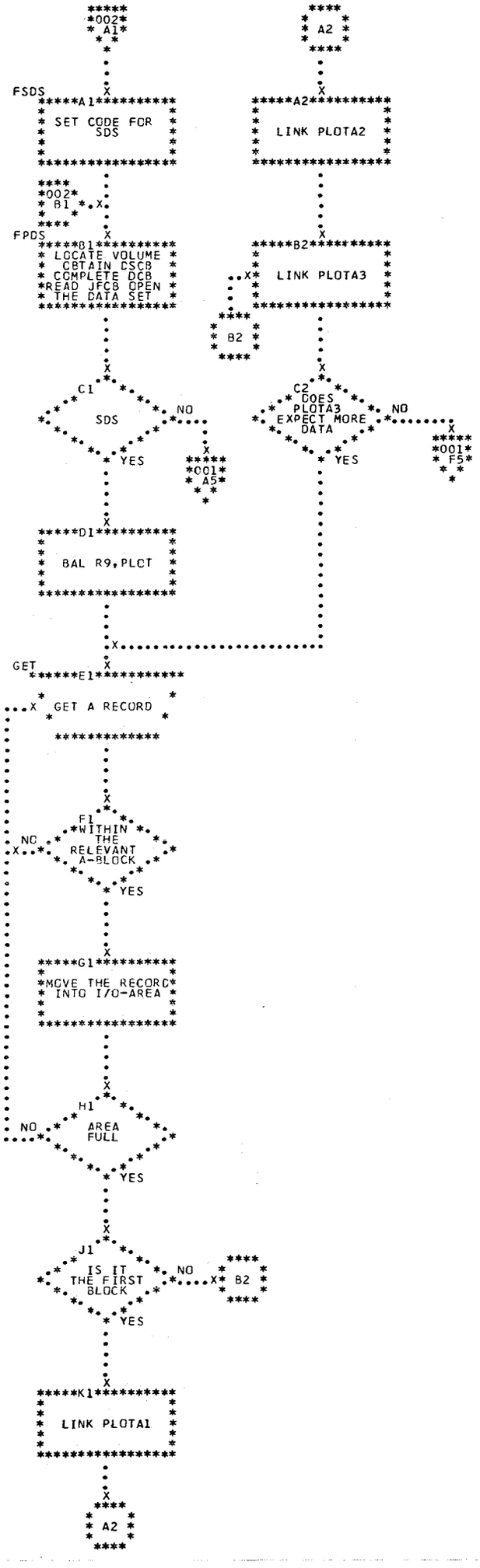

003 A1
A1 NEW ONE INSIDE & POINTS YES
NO
002 A5
B1 GET DX AND DZ
C1 IS DX=0 OR DZ=0 YES
NO
C2 REPLACE ZERO BY EPSILON
D1 SWITCH OFF SCHP=0 INCREASING OR DECREASING LINE
D2 SWITCH ON SCHP=1 HORIZONTAL OR VERTICAL LINE
E1 GET SLOPE OF THE LINE
F1 RESET CONTROL VARIABLES FOR LOOP
G1 INTERSECTION WITH SOME MARGIN
H1 BELONGS IT TO THE WINDOW NO
YES
D3
J1 SWITCH SCHP ON YES
NO
J2 ASSIGN OTHER COMPARATORS
K1 INTER-SECTION BETWEEN THE 2 POINTS YES NO
D3
A3
A3 FROM OUTSIDE YES
NO
A4 SET NEGATIVE SIGN FOR DARK VECTOR
B3 BOTH POINTS OUTSIDE YES
NO
B4 SET NEGATIVE SIGN FOR DARK VECTOR OF 1. POINT
C3 ASSIGN POINT OF INTERSECTION TO ARRAY
RANO
D3 UPDATE CONTROL VARIABLES
D3
E3 ALL SIDES TESTED NO
YES
F3 SECOND POINT INSIDE NO
YES
004 A1
G3 ASSIGN SECOND POINT TO ARRAY
004 A1
003 D4
LAB4
D4 TEXT FOR OUTPUT NO
YES
005 J5
E4 LINENR FOR TEXT OK YES
NO
G4
F4 PUT TEXT IN FIRST LINE
G4
G4 SET SWITCH ILAB=5
H4 PUT TEXT IN DESIRED POSITION
J4 SET MODE FOR TEXT OUTPUT
005 G4
003 A5
LAB2
A5 SET CODE FOR LINES
B5 SET SWITCH ILAB=3
C5 COORDINATE FRAME NO
YES
005 J5
KOVERA
D5 RESET CONTROL VARIABLES FOR VECTOR PROCESSING
E5 GET START AND END POINTS OF COORDINATES
F5 GET POINTS OF SCALES IN Z-DIRECTION
G5 GET POINTS OF SCALES IN X-DIRECTION
H5 ADD POINT FOR FINAL DARK VECTOR
004 A1

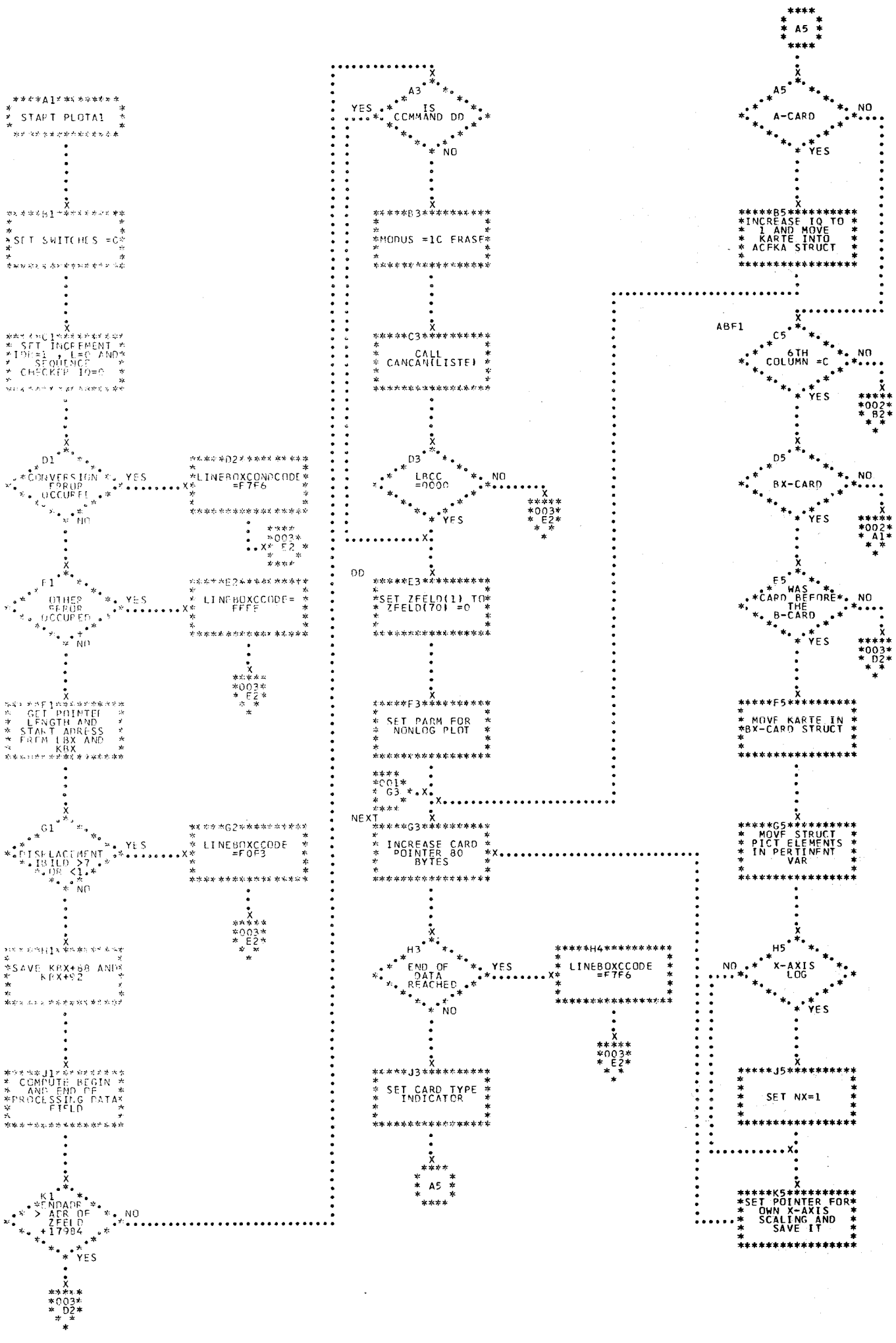

004 A1
VECTA
A1 STRIPES YES
NO
A2 INSERT POINTS FOR STRIPES
B1 RESET CONTROL VARIABLES
A3 NEGATIVE DIRECTION OF VECTORS YES
NO
A3
A4 MAKE VALUE OF DX OR DZ POSITIVE
B3 SET BITS FOR POSITIVE DIRECTION
B4 SET BITS FOR NEGATIVE DIRECTION
004 C1
VECT
C1 IS THE SIGN USED AS CO-SELECTOR YES
NO
C1
C2 RESET COUNTERS FOR DARK VECTORS
VDZ2
C3 DISTANCE LOWER THEN ONE POINT YES
NO
D1 END OF ARRAY YES
NO
D2 TEST ILAB FOR BRANCH
1.....002 A1
2.....003 A5
3.....005 F4
4.....003 D4
5.....005 H5
D3 DISTANCE MORE THEN 15 POINTS YES
NO
005 A1
E1 GET START- AND ENDPOINT OF NEXT VECTOR
E3 SET MULTIPLIER OF VECTOR VM=1
F1 DARK VECTORS FOR POINTS YES
NO
F3 ASSIGN DX AND DZ TO VECTOR
G1 VALUE OF Z NEGATIVE YES
NO
G2 MAKE IT POSITIVE AND ENCREASE COUNTER FOR DARK-VECTOR
G3 WAS PRECEDING VECTOR EQUAL NO
YES
H1 VALUE OF X NEGATIVE YES
NO
H2 MAKE IT POSITIVE AND ENCREASE COUNTER FOR POINTS
H3 WAS COUNTER LOWER THEN 14 YES
NO
H4 ENCREASE COUNTER AND CHANGE MULTIPLIER
C1
H5 ASSIGN VECTOR TO OUTPUT BUFFER
VHD3
J1 GET THE DISTANCE OF THE TWO POINTS
VECT2
J3 RESET COUNTER AND SAVE VECTOR
C1
NO
VHELLP
K4 POINT MODE YES
004 K4
K1 ASSIGN POSITION OF LIGHT BEAM
A3
K3 ENCREASE POINTER AND ASSIGN VECTOR TO OUTPUT BUFFER
K5 INSERT VECTOR FOR POINT INTO OUTPUT BUFFER

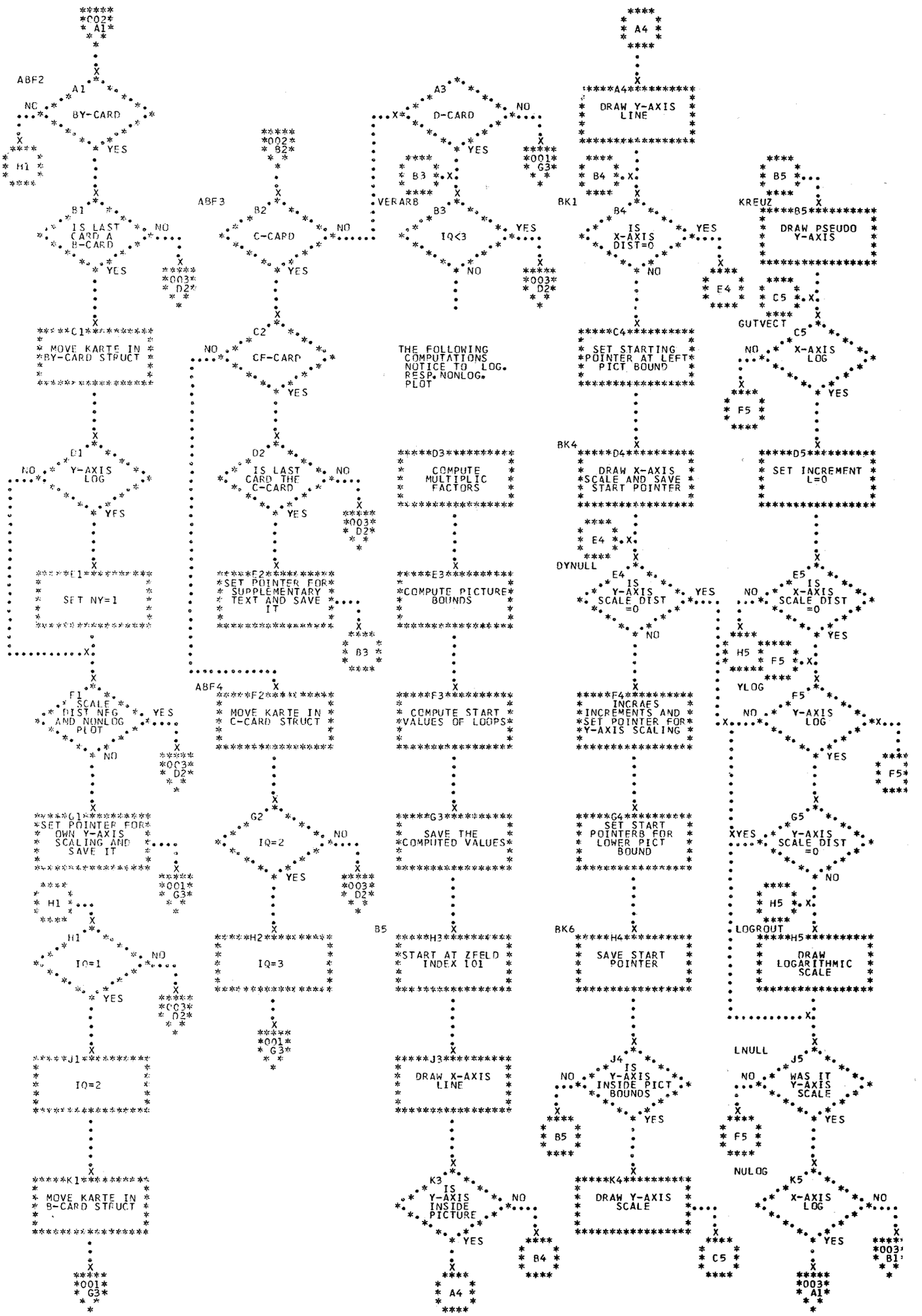

005 A1
LVECT
A1 DELETE SAVE VECTOR
B1 IS DX=0
YES
NO
C1 IS DZ=0
YES
NO
D1 DARK VECTOR
NO
YES
A3
LVZ
E1 SET X-PART OF VECTOR VX=0
F1 ARE *LAST 5 BITS* OF LENGTH ZERO
YES
NO
G1 INSERT VECTOR *WITH LENGTH OF* THESE 5 BITS
LVZ3
H1 ARE THE* 3 BITS IN * FRONT OF IT ZERO
YES
NO
J1 INSERT VECTOR OF THESE BITS
LVZ256
K1 IS THE FIRST BIT ZERO
NO
YES
A2
B2
A2
A2 INSERT TWO VECTORS OF A LENGTH OF 128
B2
B2 IS DZ=0
YES
NO
004 K4
LVX
C2 SET Z-PART OF VECTOR VZ TO *ZERO GET LENGTH*
D2 ARE *LAST 5 BITS* OF LENGTH ZERO
YES
NO
E2 INSERT VECTOR WITH THIS LENGTH
LVX3
F2 ARE THE* 3 BITS IN * FRONT OF IT ZERO
YES
NO
G2 INSERT VECTOR *FOR THESE BITS
LVX256
H2 IS THE FIRST BIT ZERO
YES
NO
004 K4
J2 INSERT TWO *VECTORS WITH A* LENGTH OF 128
004 K4
HVECT
A3 SAVE SECOND POINT AND GET POINTER TO IT
A3
B3 GET PSEUDO DESIGNATIONAL POINT
C3 DX > DZ
YES
NO
D3 DZ=15 GET *INCREMENT OF DX*
HDV
E3 GET INTERMEDIATE POINTS AS NECESSARY
F3 RESET CONTROL VARIABLES FOR LOOP
G3 GET VECTOR TO NEXT INTERMEDIATE POINT
H3 WAS IT A NULL VECTOR
YES
NO
004 C1
J3 LAST VECTOR EQUAL
NO
YES
A4
K3 COUNTER LOWER THEN 15
YES
NO
A5
A4
A4
A4 GET VECTOR INTO* OUTPUT FIELD
B4 RESET COUNTER
D4 DX=15 GET *INCREMENT OF DZ*
A5
A5 ENCREASE MULTIPLIER OF LAST VECTOR
B5 ENCREASE COUNTER
HDV5
C5 POINT OF DESIGNATION ARRIVED
YES
NO
004 C1
005 F4
LAB3
F4 SET SWITCH ILAB=4
C05 G4
VEND
G4 *ASSIGN MODE AND* LENGTH TO KBX+92
H4 *CANCAN
CALL CANCAN FOR OUTPUT
J4 *ERROR-CODE LBCC = X'0000'
NO
YES
K4 RESET CONTROL VARIABLES TO GET FORMER POINT
001 E4
COMEINF
F5 *SET ERROR-CODE *LBCC = X'FFFE'
005 H5
LAB5
H5 *UPDATE LBX FOR * NORMAL RETURN
005 J5
ENDE
J5 RETURN

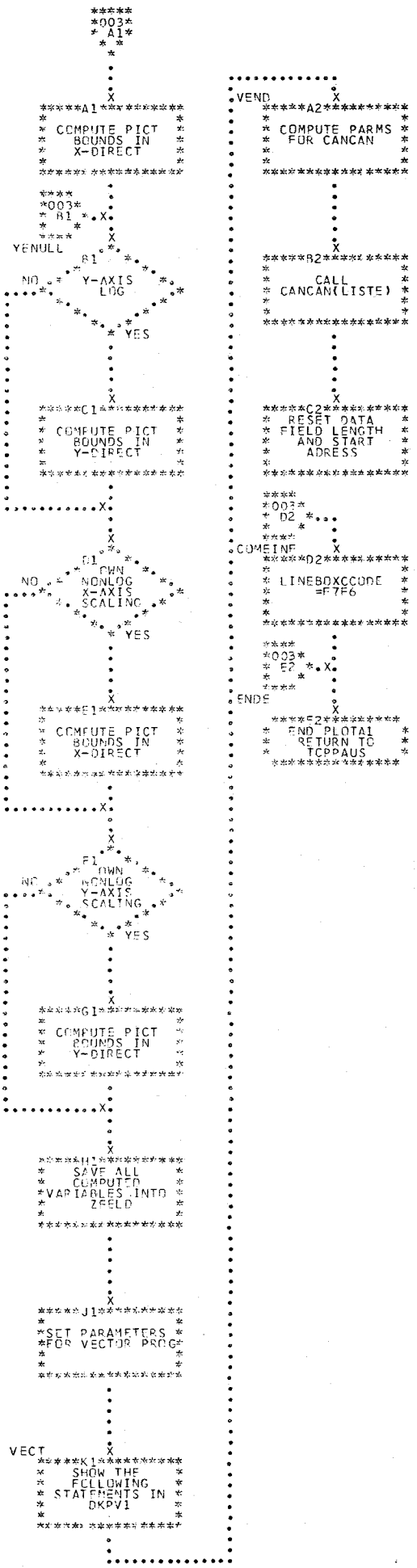

A1
ENTRY VIA CALL FROM PL/1 PROGRAMS
R1
GET PARAMETERS
GETUA
C1
GET DISPLAYBOX FOR ACTUAL LINEBOX
D1
DISPLAY-BOX FOUND
NO
YES
ERROR1
D2
MSG45: NO DISPLAYBOX FOR SPECIFIED CU
H3
DBXFOUND
E1
SET NUMBER OF ACTUAL DISPLAY UNIT
F1
ERASE ONLY
YES
NO
LOE
F2
ADDRESS OF ERASE-CP INTO IOB
B3
G1
ADDRESS OF OUTPUT-CP INTO IOB
H1
TEXT OUTPUT
NO
YES
J1
EXTENDED MODE
NO
YES
J2
INCREMENT IS ONE
A3
K1
INCREMENT IS THREE
A3
TR86
A3
TRANSLATE
TRANSLATION EBCDIC - ASCII
EXECUTE
B3
SET RETRY COUNT
C3
EXCP
C3
EXCP START CHANNEL - PROGRAM
D3
RETURN-CODE = 0
YES
K4
NO
E3
THIRD RETRY
YES
NO
ERROR4
F4
MSG48: INTERVENTION REQUIRED ON UNIT XXX
H3
F3
CU NOT OPERATIONAL
YES
NO
F4
DECREASE RETRY COUNT
C3
ERROR5
G3
MSG49: I/O-ERROR ON UNIT XXX
H3
ERPORTYP
H3
WTO: ERROR MSG TO OPERATOR
ERETOUR
J3
DUMP
K3
SET ERROR-CODE
RETOUR
K4
WAIT FOR TIMER
K5
RETURN TO CALLING PROGRAM

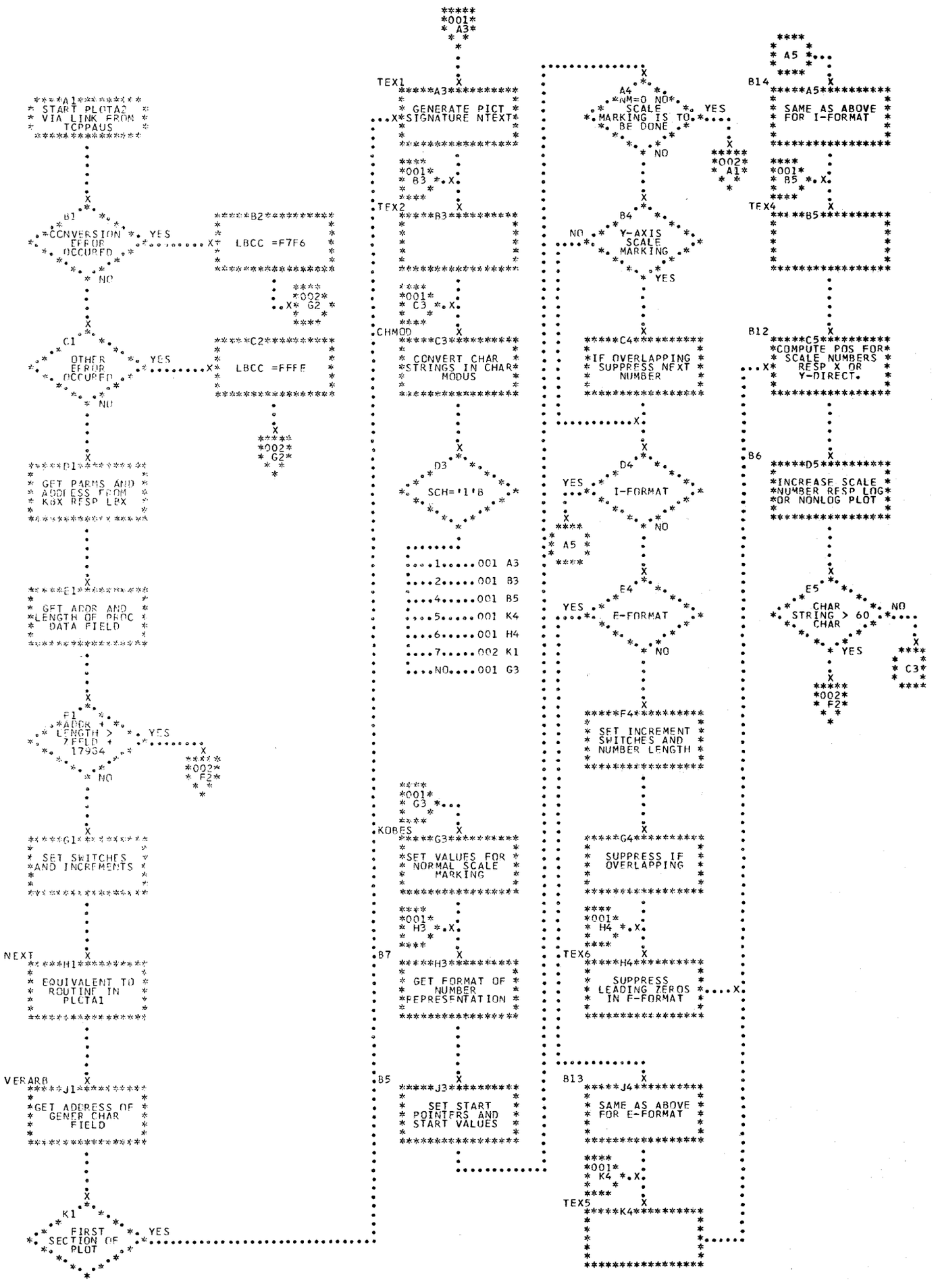

A1
ENTRY VIA LINK FROM DKP
B1
CHAIN SAVE AREA
C1
SET INTERVAL TIMER
D1
INITIALIZE REGISTERS AND AREAS
E1
DO
YES
NO
J4
PLUSMIN
F1
D+ OR D-
YES
NO
F1
G1
C5
A0
G1
PDS INDICATED
NO
YES
A2
H1
MOVE DSNAME AND MEMBERNAME INTO BUFFER
J1
SET CCDE FOR PDS IN BFLG (LINEBOX)
002
B1
KREUZ
A2
SDS INDICATED
NO
YES
B2
MOVE DSNAME INTO BUFFER
002
A1
EMEM
E2
MOVE DSNAME INTO BUFFER SET CODE FOR SDS
F2
R9 = A(LAPSYN)
PLOT
G2
CONVERT THE BLOCK-NR. AND PAGE-NR. TO BINARY
H2
BR R9
LAPSYN
K2
SET EODAD- AND SYNAD-ADDR IN DCB. INITIALIZE REGS R9=A(A6)
FIND
A3
FIND THE BEGINNING ADDR OF THE MEMBER
B3
OK
B3
READ A BLOCK FROM PDS INTO I/O-AREA
C3
BR R9
A6
E3
SEARCH THE BEGINNING OF THE RELEVANT A-BLOCK
E3
F3
FOUND
YES
NO
J3
G3
ANY MORE DATA TO CHECK
YES
NO
E3
ERSTE
J3
LOCATE THE BEGINNING ADDR WITHIN I/O-AREA TO READ NEXT BLOCK
K3
R9 = A(A8)
B3
A8
A4
R9 = A(M32)
B4
LINK PLOTA1
C4
LINK PLOTA2
M31
D4
LINK PLOTA3
E4
BR R9
M32
F4
BRANCH TWICE TO OK TO READ DATA INTO I/O-AREA
G4
R9 = A(M32)
DO
J4
RESTORE SOME AREAS OF LINEBOX
J4
F1
PARTPLOT
A5
DO THE SAME PROCEDURE LIKE GET-ROUTINE BUT WITH BPAM
001
A5
DPLMN
C5
ADD +1 OR -1 TO PAGE-NR. LOAD R7 WITH NR.
C5
G1
001
F5
ENDE
F5
CLOSE THE DATA SET. TEST TIMER INTERVAL
G5
BRANCH TO THE CALLING PROGRAM

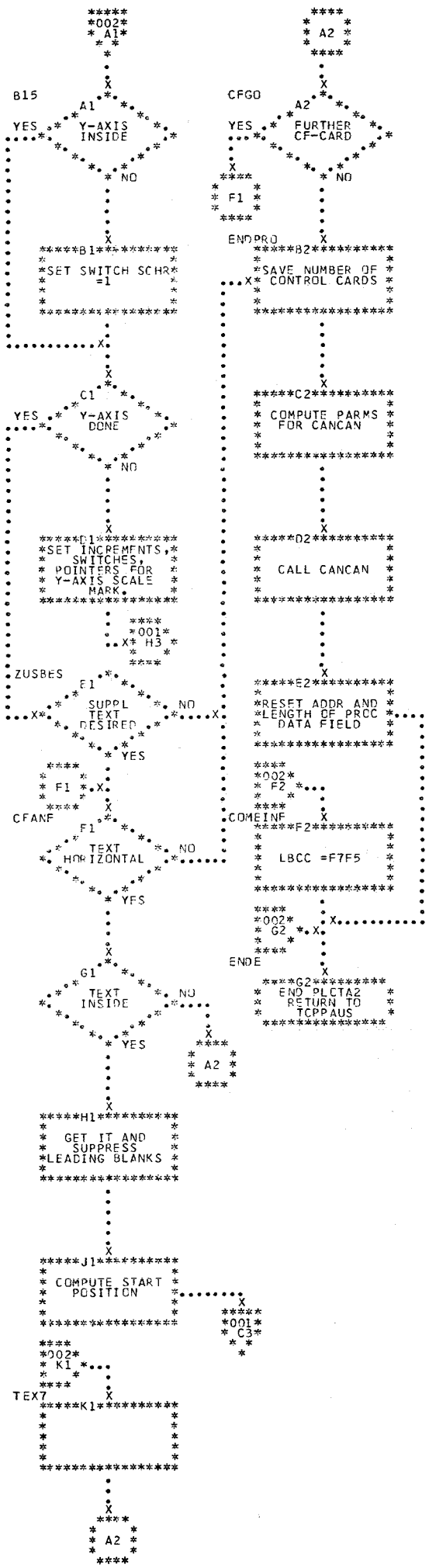

002
A1
A2
FSDS
A1
SET CODE FOR SDS
A2
LINK PLOTA2
002
B1
FPDS
B1
LOCATE VOLUME
OBTAIN DSCB
COMPLETE DCB
READ JFCB OPEN
THE DATA SET
B2
LINK PLOTA3
B2
C1
SDS
NO
YES
001
A5
C2
DOES PLOTA3 EXPECT MORE DATA
NO
YES
001
F5
D1
BAL R9,PLOT
GET
E1
GET A RECORD
F1
WITHIN THE RELEVANT A-BLOCK
NO
YES
G1
MOVE THE RECORD INTO I/O-AREA
H1
AREA FULL
NO
YES
J1
IS IT THE FIRST BLOCK
NO
B2
YES
K1
LINK PLOTA1
A2

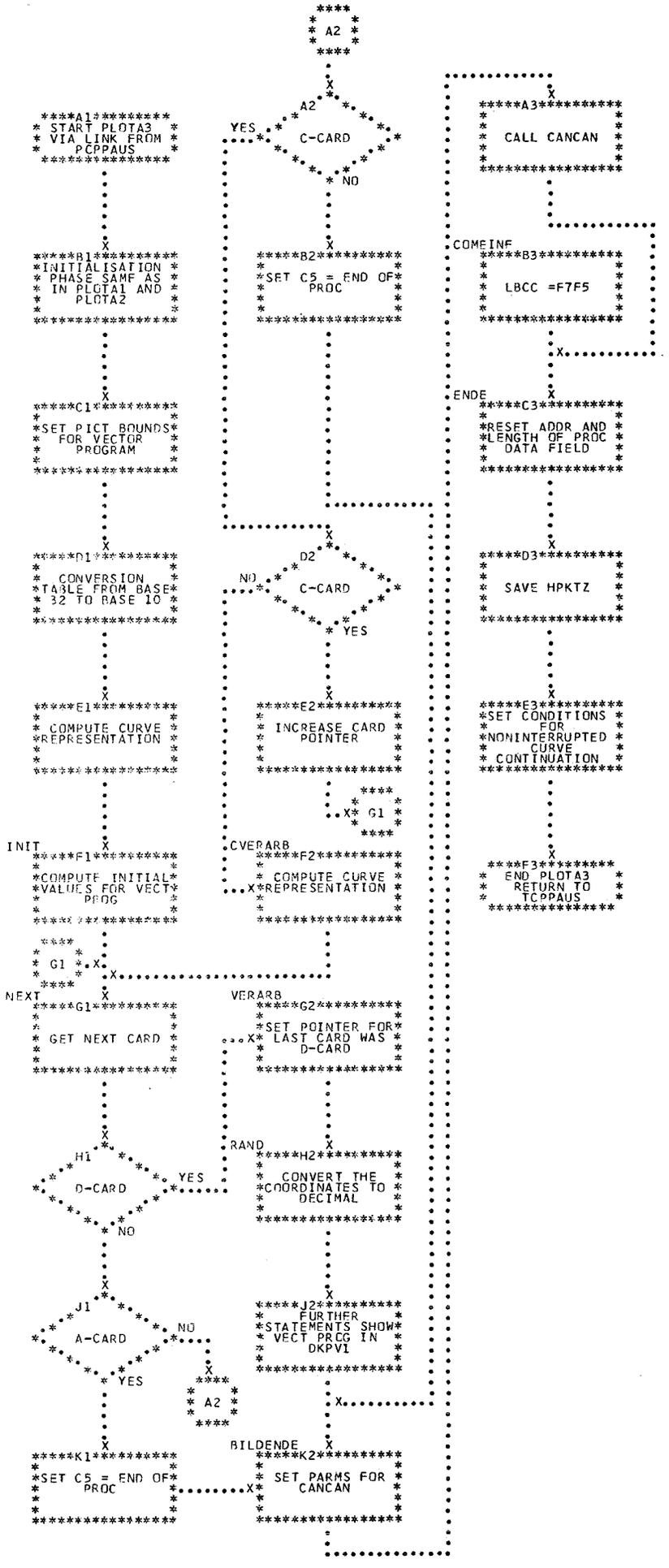

A1
START PLOTA1
B1
SET SWITCHES =0
C1
SET INCREMENT IQ1=1 , L=0 AND SEQUENCE CHECKER IQ=0
D1
CONVERSION ERROR OCCURED
NO
D2
LINEBOXCONDCODE =F7F6
003 E2
E1
OTHER ERROR OCCURED
NO
E2
LINEBOXCCODE= FFFE
003 E2
F1
GET POINTER LENGTH AND START ADRESS FROM LBX AND KBX
G1
DISPLACEMENT ISIZE >7 OR <1
NO
G2
LINEBOXCCODE =F0F3
003 E2
H1
SAVE KBX+68 AND KBX+92
J1
COMPUTE BEGIN AND END OF PROCESSING DATA FIELD
K1
ENDADR > ADR OF ZFELD +17984
NO
YES
003 D2
A3
IS COMMAND DD
NO
B3
MODUS =1C ERASE
C3
CALL CANCAN(LISTE)
D3
LBCC =0000
NO
YES
003 E2
DD
E3
SET ZFELD(1) TO ZFELD(70) =0
F3
SET PARM FOR NONLOG PLOT
001 G3
NEXT
G3
INCREASE CARD POINTER 80 BYTES
H3
END OF DATA REACHED
YES
NO
H4
LINEBOXCCODE =F7F6
003 E2
J3
SET CARD TYPE INDICATOR
A5
A5
A-CARD
NO
YES
B5
INCREASE IQ TO 1 AND MOVE KARTE INTO ACFKA STRUCT
ABF1
C5
6TH COLUMN =C
NO
YES
002 B2
D5
BX-CARD
NO
YES
002 A1
E5
WAS CARD BEFORE THE B-CARD
NO
YES
003 D2
F5
MOVE KARTE IN BX-CARD STRUCT
G5
MOVE STRUCT PICT ELEMENTS IN PERTINENT VAR
H5
X-AXIS LOG
NO
YES
J5
SET NX=1
K5
SET POINTER FOR OWN X-AXIS SCALING AND SAVE IT


002 A1
A1 BY-CARD
NO
YES
H1
B1 IS LAST CARD A B-CARD
NO
YES
003 D2
C1 MOVE KARTE IN BY-CARD STRUCT
D1 Y-AXIS LOG
NO
YES
E1 SET NY=1
F1 SCALE DIST NEG AND NONLOG PLOT
YES
NO
003 D2
G1 SET POINTER FOR OWN Y-AXIS SCALING AND SAVE IT
001 G3
H1
H1 IQ=1
NO
YES
003 D2
J1 IQ=2
K1 MOVE KARTE IN B-CARD STRUCT
001 G3
ABF2
ABF3
002 B2
B2 C-CARD
NO
YES
C2 CF-CARD
NO
YES
D2 IS LAST CARD THE C-CARD
NO
YES
003 D2
F2 SET POINTER FOR SUPPLEMENTARY TEXT AND SAVE IT
B3
ABF4
F2 MOVE KARTE IN C-CARD STRUCT
G2 IQ=2
NO
YES
001 G3
003 D2
H2 IQ=3
001 G3
A3 D-CARD
NO
YES
001 G3
B3
VERARB
B3 IQ<3
YES
NO
003 D2
THE FOLLOWING COMPUTATIONS NOTICE TO LOG. RESP. NONLOG. PLOT
D3 COMPUTE MULTIPLIC FACTORS
E3 COMPUTE PICTURE BOUNDS
F3 COMPUTE START VALUES OF LOOPS
G3 SAVE THE COMPUTED VALUES
B5
H3 START AT ZFELD INDEX I01
J3 DRAW X-AXIS LINE
K3 IS Y-AXIS INSIDE PICTURE
NO
YES
B4
A4
A4 DRAW Y-AXIS LINE
B4
BK1
B4 IS X-AXIS DIST=0
YES
NO
E4
C4 SET STARTING POINTER AT LEFT PICT BOUND
BK4
D4 DRAW X-AXIS SCALE AND SAVE START POINTER
E4
DYNULL
E4 IS Y-AXIS SCALE DIST =0
YES
NO
F4 INCRAES INCREMENTS AND SET POINTER FOR Y-AXIS SCALING
G4 SET START POINTERB FOR LOWER PICT BOUND
BK6
H4 SAVE START POINTER
J4 IS Y-AXIS INSIDE PICT BOUNDS
NO
YES
B5
K4 DRAW Y-AXIS SCALE
C5
B5
KREUZ
B5 DRAW PSEUDO Y-AXIS
C5
GUTVECT
C5 X-AXIS LOG
NO
YES
F5
D5 SET INCREMENT L=0
E5 IS X-AXIS SCALE DIST =0
NO
YES
H5
F5
YLOG
F5 Y-AXIS LOG
NO
YES
F5
G5 Y-AXIS SCALE DIST =0
YES
NO
H5
LOGROUT
H5 DRAW LOGARITHMIC SCALE
LNULL
J5 WAS IT Y-AXIS SCALE
NO
YES
F5
NULOG
K5 X-AXIS LOG
NO
YES
003 B1
003 A1


003
A1
COMPUTE PICT BOUNDS IN X-DIRECT
003
B1
YENULL
B1
NO
Y-AXIS LOG
YES
C1
COMPUTE PICT BOUNDS IN Y-DIRECT
D1
OWN
NONLOG X-AXIS SCALING
YES
E1
COMPUTE PICT BOUNDS IN X-DIRECT
F1
OWN
NONLOG Y-AXIS SCALING
YES
G1
COMPUTE PICT BOUNDS IN Y-DIRECT
H1
SAVE ALL COMPUTED VARIABLES INTO ZFELD
J1
SET PARAMETERS FOR VECTOR PROG
VECT
K1
SHOW THE FOLLOWING STATEMENTS IN DKPV1
VEND
A2
COMPUTE PARMS FOR CANCAN
B2
CALL CANCAN(LISTE)
C2
RESET DATA FIELD LENGTH AND START ADRESS
003
D2
COMEINE
D2
LINEBOXCCODE =F7F6
003
E2
ENDE
E2
END PLOTA1 RETURN TO TCPPAUS


A1
START PLOTA2 VIA LINK FROM TCPPAUS
B1
CONVERSION ERROR OCCURED
YES
NO
B2
LBCC =F7F6
002
G2
C1
OTHER ERROR OCCURED
YES
NO
C2
LBCC =FFFF
002
G2
D1
GET PARMS AND ADDRESS FROM K6X RESP LBX
E1
GET ADDR AND LENGTH OF PROC DATA FIELD
F1
ADDR + LENGTH > FIELD + 17934
YES
NO
002
F2
G1
SET SWITCHES AND INCREMENTS
NEXT
H1
EQUIVALENT TO ROUTINE IN PLOTA1
VERARB
J1
GET ADDRESS OF GENER CHAR FIELD
K1
FIRST SECTION OF PLOT
YES
001
A3
TEX1
A3
GENERATE PICT SIGNATURE NTEXT
TEX2
B3
001
B3
CHMOD
C3
CONVERT CHAR STRINGS IN CHAR MODUS
D3
SCH='1'B
1....001 A3
2....001 B3
4....001 B5
5....001 K4
6....001 H4
7....002 K1
NO....001 G3
KOBES
G3
SET VALUES FOR NORMAL SCALE MARKING
H3
B7
H3
GET FORMAT OF NUMBER REPRESENTATION
B5
J3
SET START POINTERS AND START VALUES
A4
NM=0 NO
SCALE MARKING IS TO BE DONE
YES
NO
002
A1
B4
Y-AXIS SCALE MARKING
NO
YES
C4
IF OVERLAPPING SUPPRESS NEXT NUMBER
D4
I-FORMAT
YES
NO
A5
E4
E-FORMAT
YES
NO
F4
SET INCREMENT SWITCHES AND NUMBER LENGTH
G4
SUPPRESS IF OVERLAPPING
H4
TEX6
SUPPRESS LEADING ZEROS IN F-FORMAT
B13
J4
SAME AS ABOVE FOR E-FORMAT
001
K4
TEX5
K4
A5
B14
A5
SAME AS ABOVE FOR I-FORMAT
001
B5
TEX4
B5
B12
C5
COMPUTE POS FOR SCALE NUMBERS RESP X OR Y-DIRECT.
B6
D5
INCREASE SCALE NUMBER RESP LOG OR NONLOG PLOT
E5
CHAR STRING > 60 CHAR
YES
NO
002
F2
C3

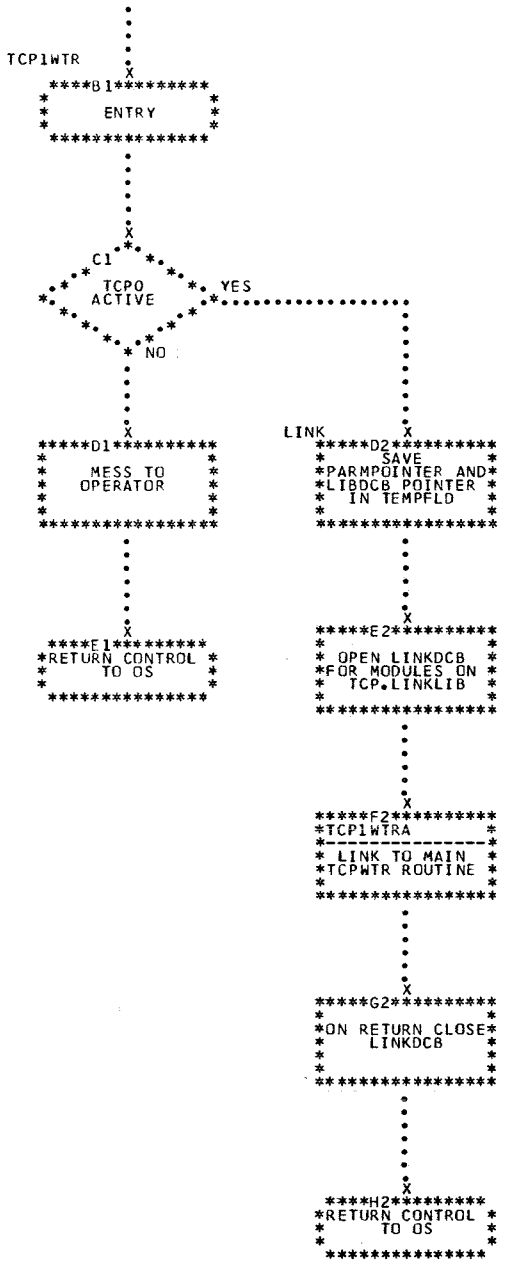

002
A1
B15
A1
YES
Y-AXIS INSIDE
NO
B1
SET SWITCH SCHR =1
C1
YES
Y-AXIS DONE
NO
D1
SET INCREMENTS, SWITCHES, POINTERS FOR Y-AXIS SCALE MARK
001
H3
ZUSBES
F1
YES
SUPPL TEXT DESIRED
NO
CFANF
F1
YES
TEXT HORIZONTAL
NO
G1
YES
TEXT INSIDE
NO
A2
H1
GET IT AND SUPPRESS LEADING BLANKS
J1
COMPUTE START POSITION
001
C3
002
K1
TEX7
K1
A2
A2
CFGO
A2
YES
FURTHER CF-CARD
NO
F1
ENDPRO
B2
SAVE NUMBER OF CONTROL CARDS
C2
COMPUTE PARMS FOR CANCAN
D2
CALL CANCAN
E2
RESET ADDR AND LENGTH OF PRCC DATA FIELD
002
F2
COMEINF
F2
LBCC =F7F5
002
G2
ENDE
G2
END PLOTA2 RETURN TO TCPPAUS


A2
A2
YES
C-CARD
NO
A1
START PLOTA3
VIA LINK FROM
PCPAUS
B1
INITIALISATION
PHASE SAME AS
IN PLOTA1 AND
PLOTA2
B2
SET C5 = END OF
PROC
A3
CALL CANCAN
COMEINF
B3
LBCC =F7F5
ENDE
C1
SET PICT BOUNDS
FOR VECTOR
PROGRAM
C3
RESET ADDR AND
LENGTH OF PROC
DATA FIELD
D1
CONVERSION
TABLE FROM BASE
32 TO BASE 10
D2
NO
C-CARD
YES
D3
SAVE HPKTZ
E1
COMPUTE CURVE
REPRESENTATION
E2
INCREASE CARD
POINTER
E3
SET CONDITIONS
FOR
NONINTERRUPTED
CURVE
CONTINUATION
G1
INIT
F1
COMPUTE INITIAL
VALUES FOR VECT
PROG
CVERARB
F2
COMPUTE CURVE
REPRESENTATION
F3
END PLOTA3
RETURN TO
TCPPAUS
G1
NEXT
GET NEXT CARD
G2
VERARB
SET POINTER FOR
LAST CARD WAS
D-CARD
H1
D-CARD
YES
NO
H2
RAND
CONVERT THE
COORDINATES TO
DECIMAL
J1
A-CARD
NO
YES
J2
FURTHER
STATEMENTS SHOW
VECT PROG IN
DKPV1
A2
K1
SET C5 = END OF
PROC
K2
BILDENDE
SET PARMS FOR
CANCAN

MODULE T C P 1 W T R ON SYS1.LINKLIB PAGE 001

STARTUP MODULE FOR TCP-OUTPUTWRITER, GETS CONTROL ON S TCPWTR,GIVES CONTROL TO TCP1WTRA ON TCP.LINKLIB APR 1, 1971

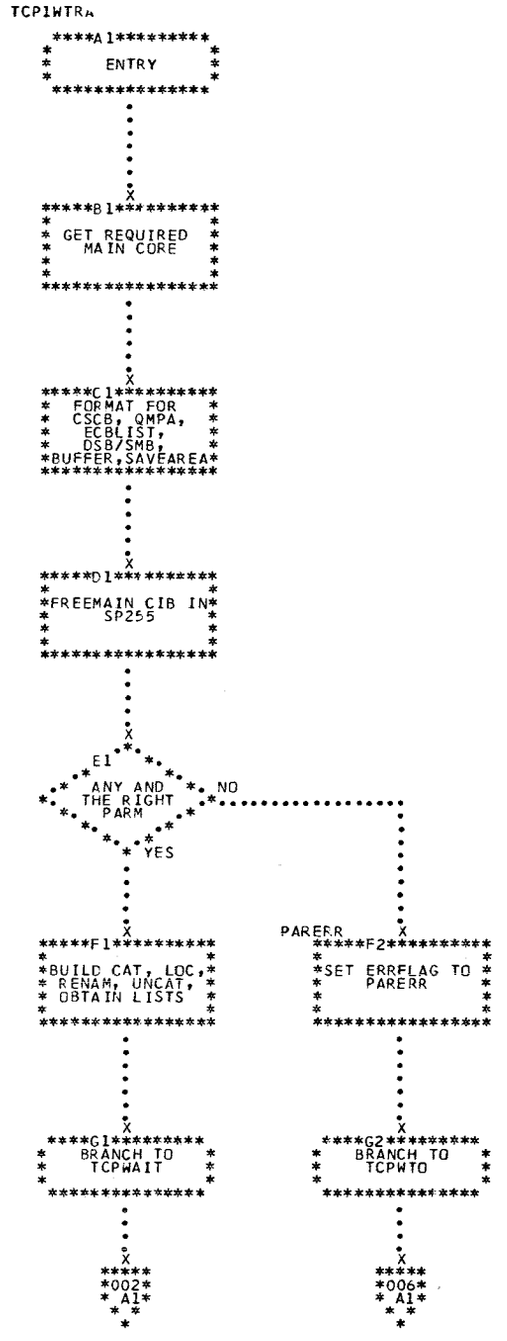
INPUT:POINTER TO CLASS PAR IN R1, OUTPUT : POINTER TO CLASSPARM AND TO LINKDCB IN TEMPFLD

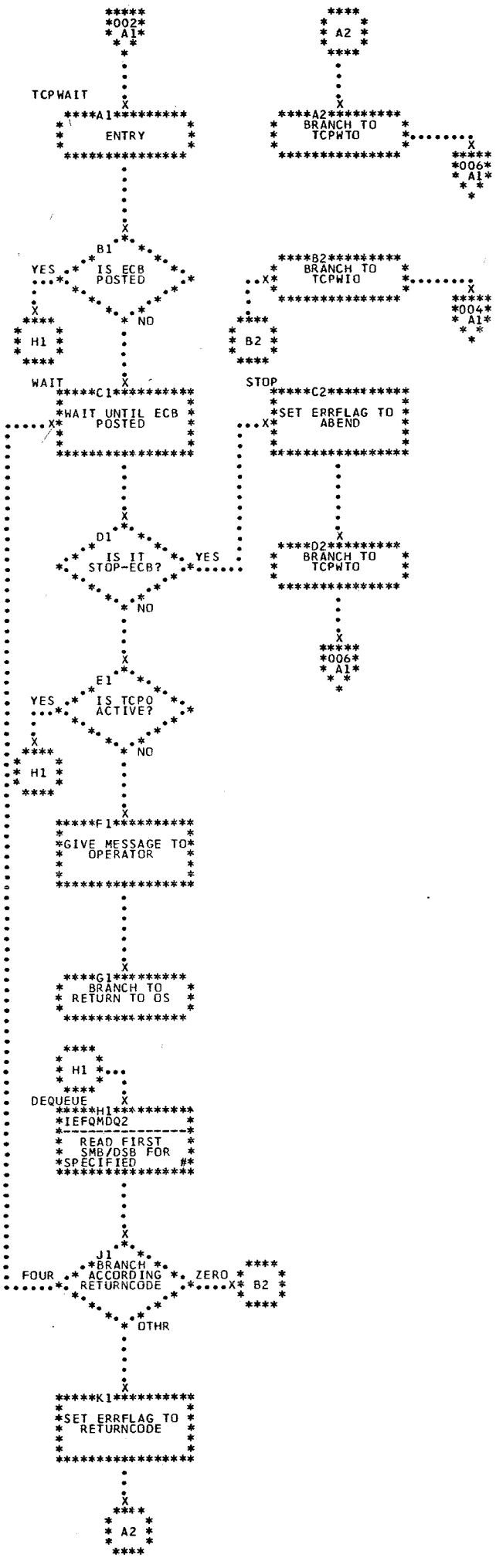


MODULE T C P W T R A ON TCP.LINKLIB

ENTRY FROM TCPWTR ON SYS1.LINKLIB VIA LINK

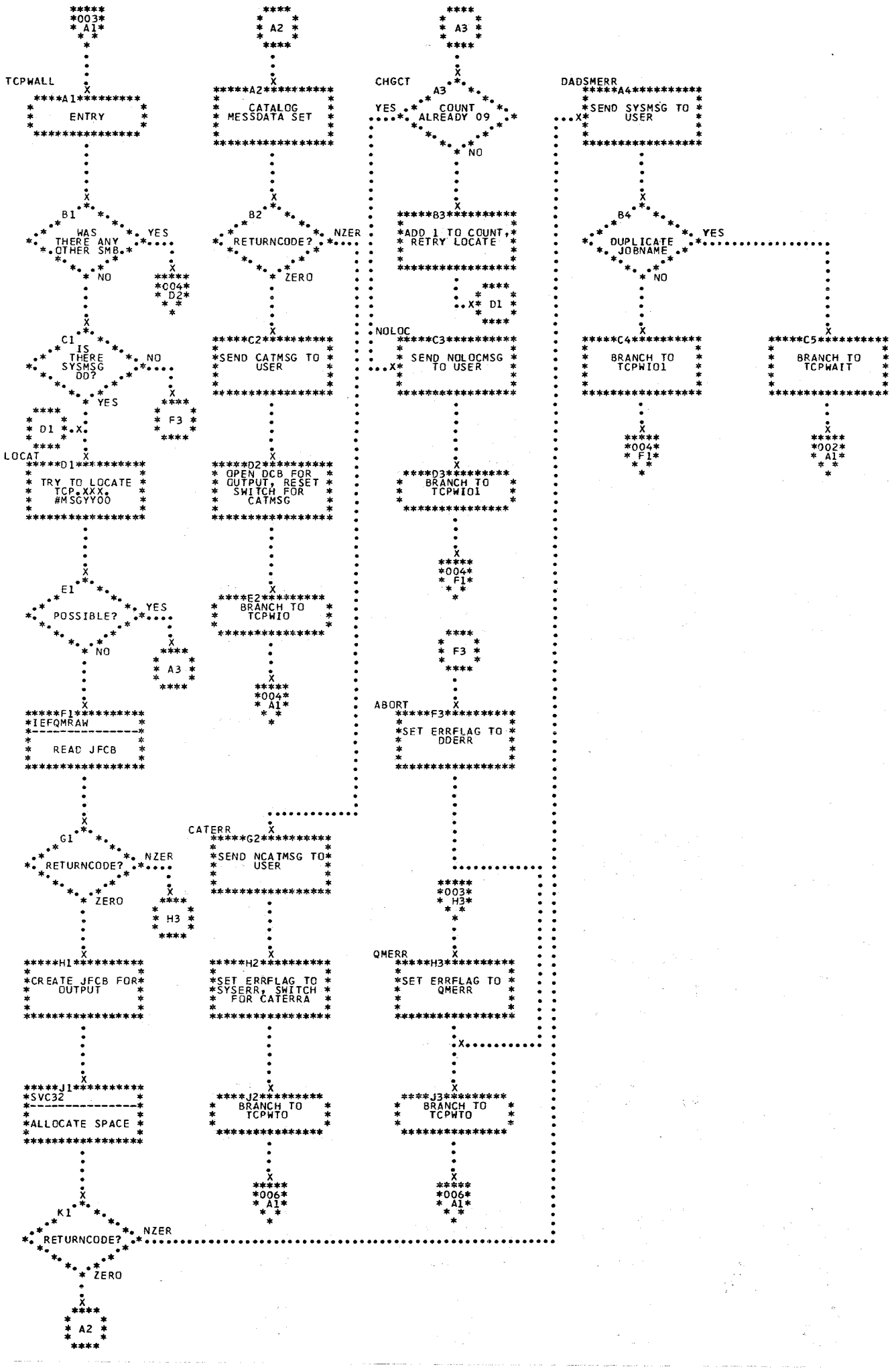
INPUT: CSCB-PARMLIST-POINTER,OUTPUT:FORMATTED CORE-AREA

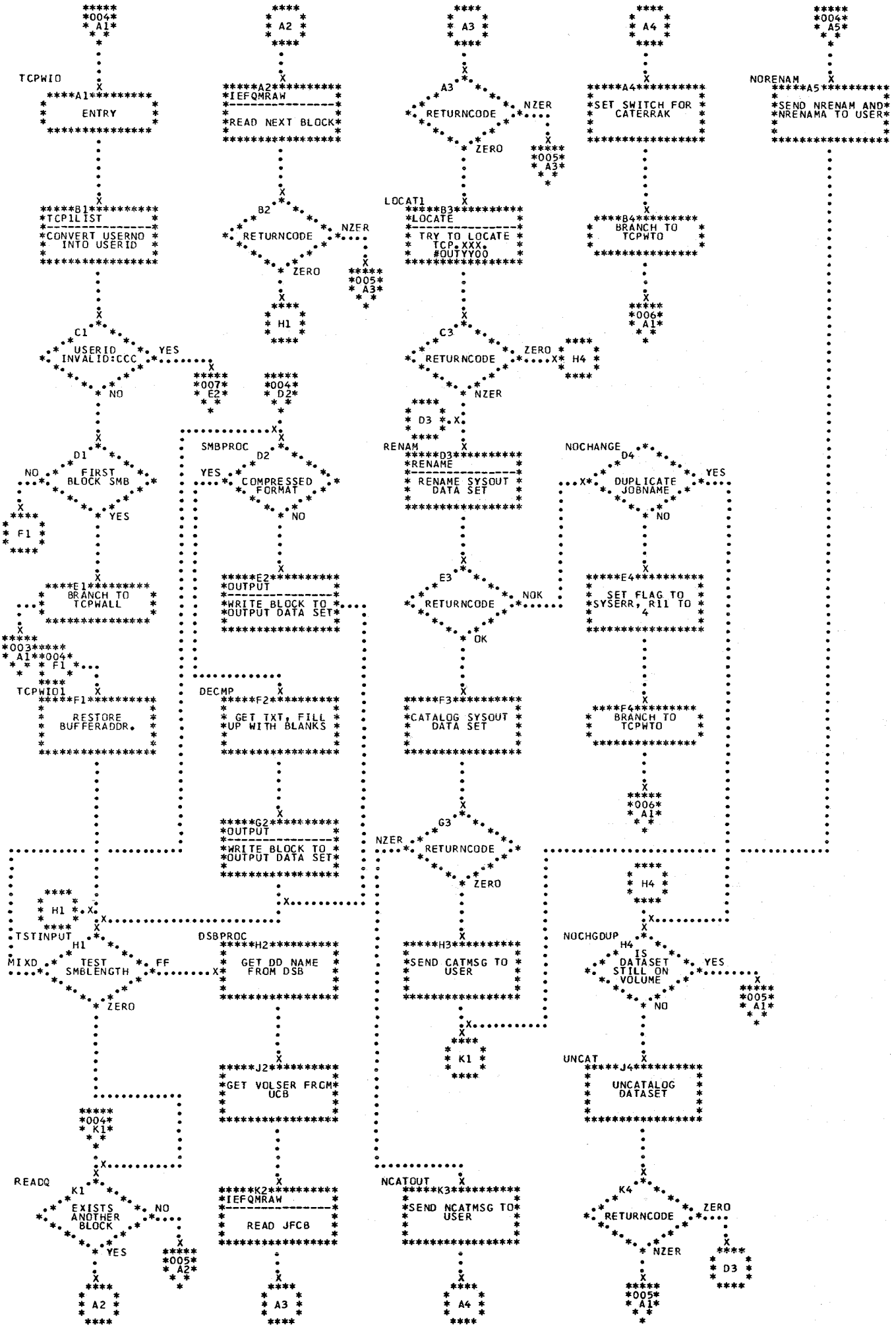




H1. OUTPUTCLASS

INPUT:PARMLIST, TIOT, JFCB FOR SYSMSG, OUTPUT:OPENED DCB, OUTPUT-BUFFER, -VTOC, PARMLIST

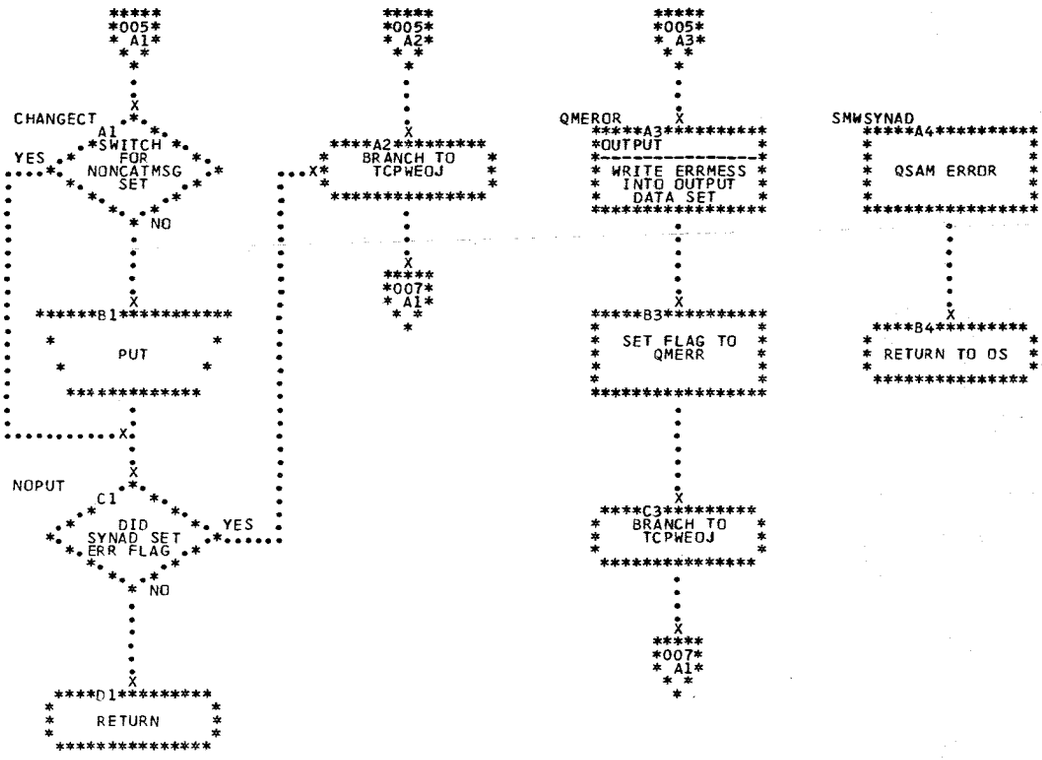



TCPWIO
ENTRY
TCP1LIST
CONVERT USERNO INTO USERID
USERID INVALID:CCC
YES
NO
FIRST BLOCK SMB
NO
YES
BRANCH TO TCPWALL
TCPWIO1
RESTORE BUFFERADDR.
TSTINPUT
TEST SMBLENGTH
FF
ZERO
READQ
EXISTS ANOTHER BLOCK
NO
YES
SMBPROC
COMPRESSED FORMAT
YES
NO
DECMP
GET TXT, FILL UP WITH BLANKS
DSBPROC
GET DD NAME FROM DSB
GET VOLSER FROM UCB
IEFQMRAW
READ JFCB
IEFQMRAW
READ NEXT BLOCK
RETURNCODE
NZER
ZERO
LOCAT1
LOCATE
TRY TO LOCATE TCP.XXX. #OUTYY00
RETURNCODE
ZERO
NZER
RENAM
RENAME
RENAME SYSOUT DATA SET
RETURNCODE
NOK
OK
CATALOG SYSOUT DATA SET
RETURNCODE
NZER
ZERO
SEND CATMSG TO USER
NCATOUT
SEND NCATMSG TO USER
NORENAM
SEND NRENAM AND NRENAMA TO USER
SET SWITCH FOR CATERRAK
BRANCH TO TCPWTO
NOCHANGE
DUPLICATE JOBNAME
YES
NO
SET FLAG TO SYSERR, R11 TO 4
BRANCH TO TCPWTO
NOCHGDUP
IS DATASET STILL ON VOLUME
YES
NO
UNCAT
UNCATALOG DATASET
RETURNCODE
ZERO
NZER
003
004
005
006
007
A1
A2
A3
A4
A5
B1
B2
B3
B4
C1
C3
D1
D2
D3
D4
E1
E2
E3
E4
F1
F2
F3
F4
G2
G3
H1
H2
H3
H4
J2
J4
K1
K2
K3
K4

MODULE T C P W T R A ON TCP.LINKLIB

T C P W I O , ENTRY FROM TCPWAIT AND TCPWIO1 FROM TCPWALL

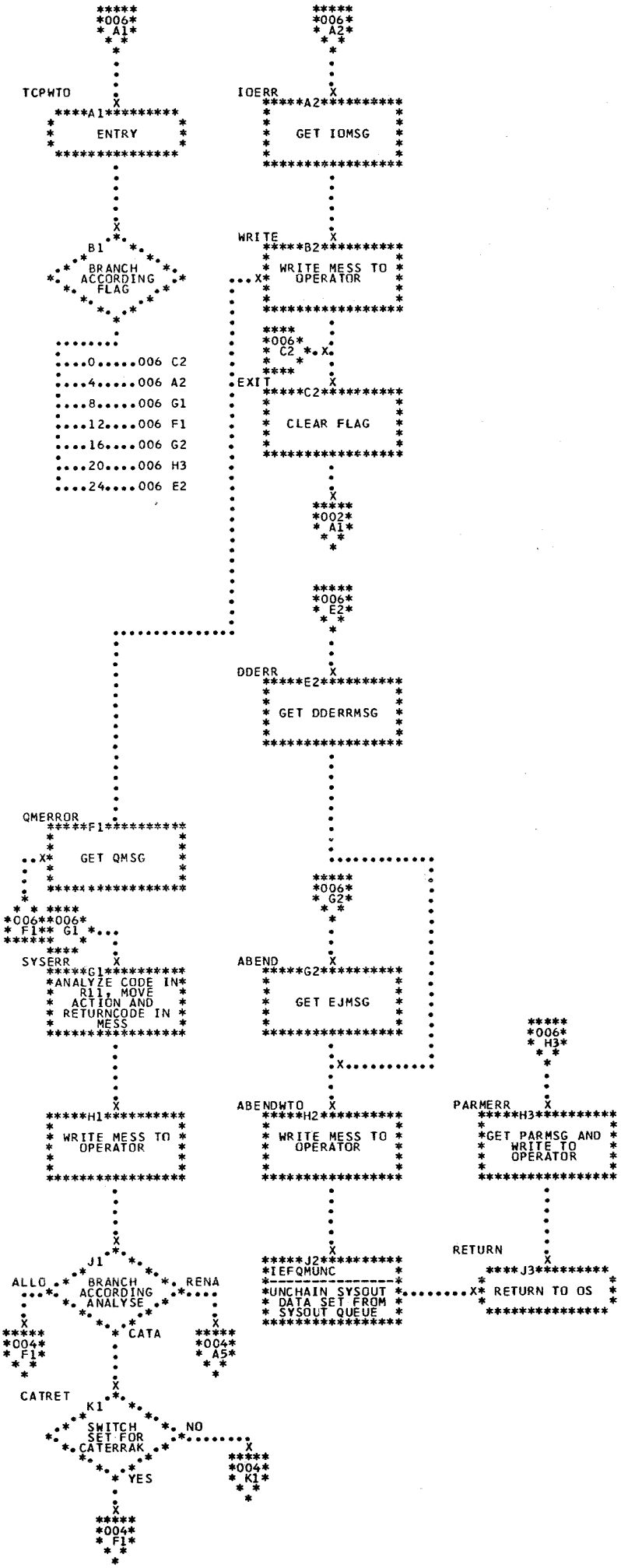
INPUT: ADDR. OF PARMLIST, OUTPUT: QSAM DATA SET

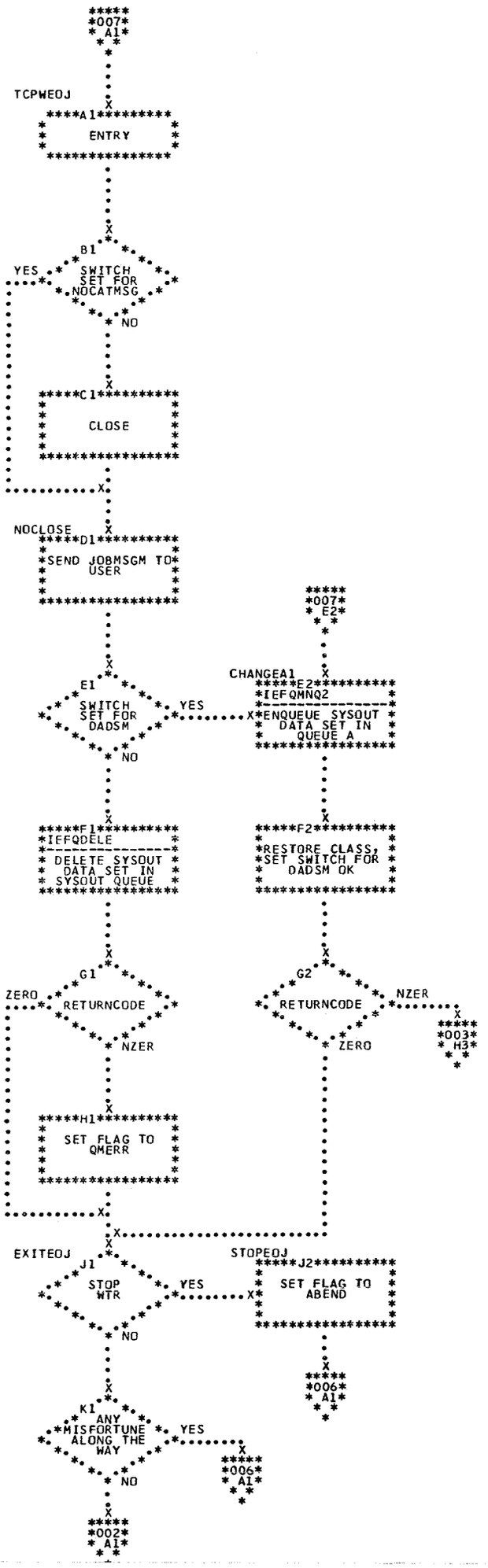


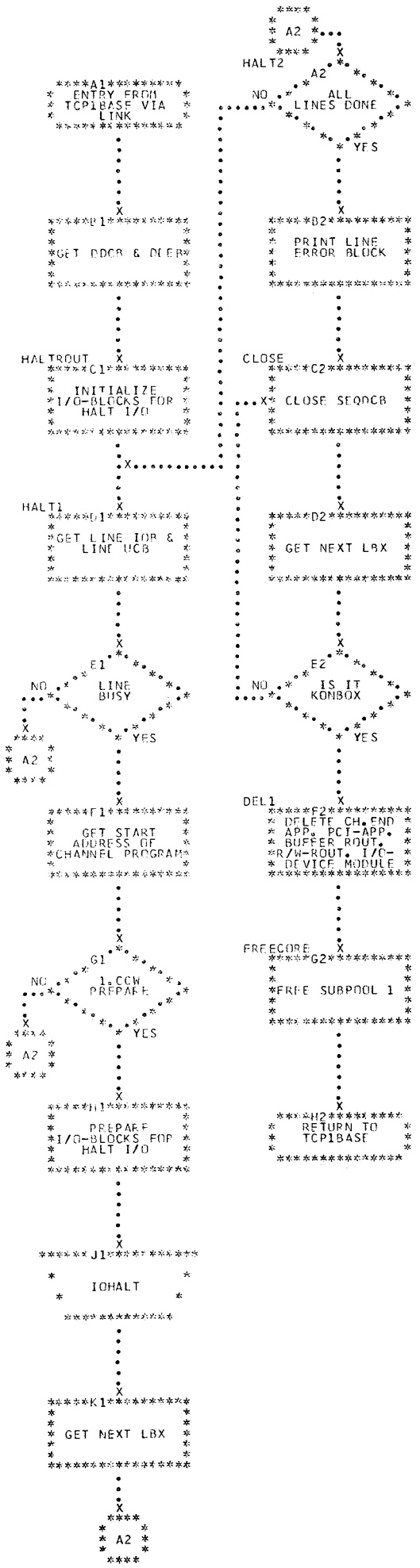
MODULE T C P W T R A ON TCP.LINKLIB

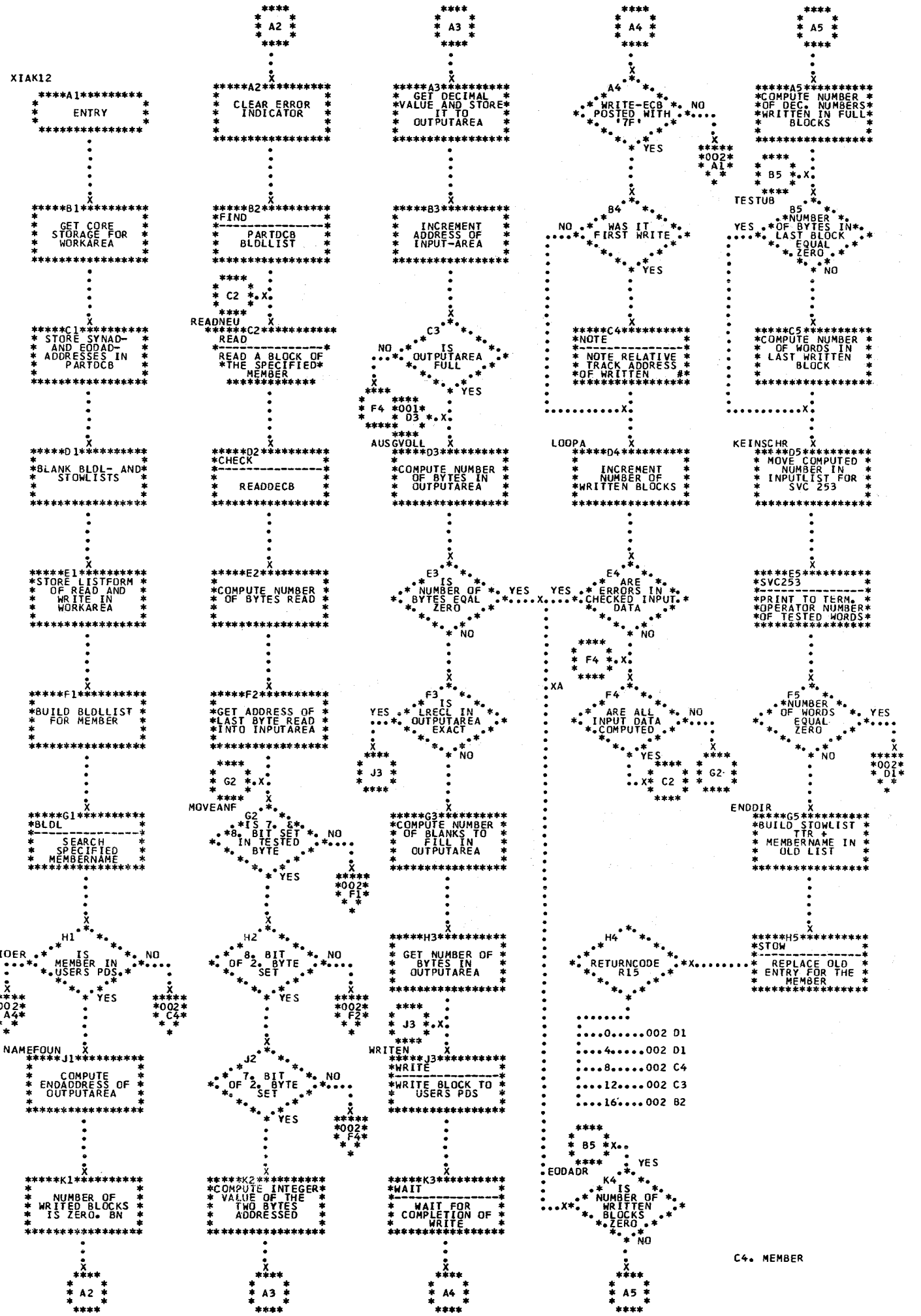
T C P W T O , ENTRY FROM ALL OTHER TCPWTR ROUTINES

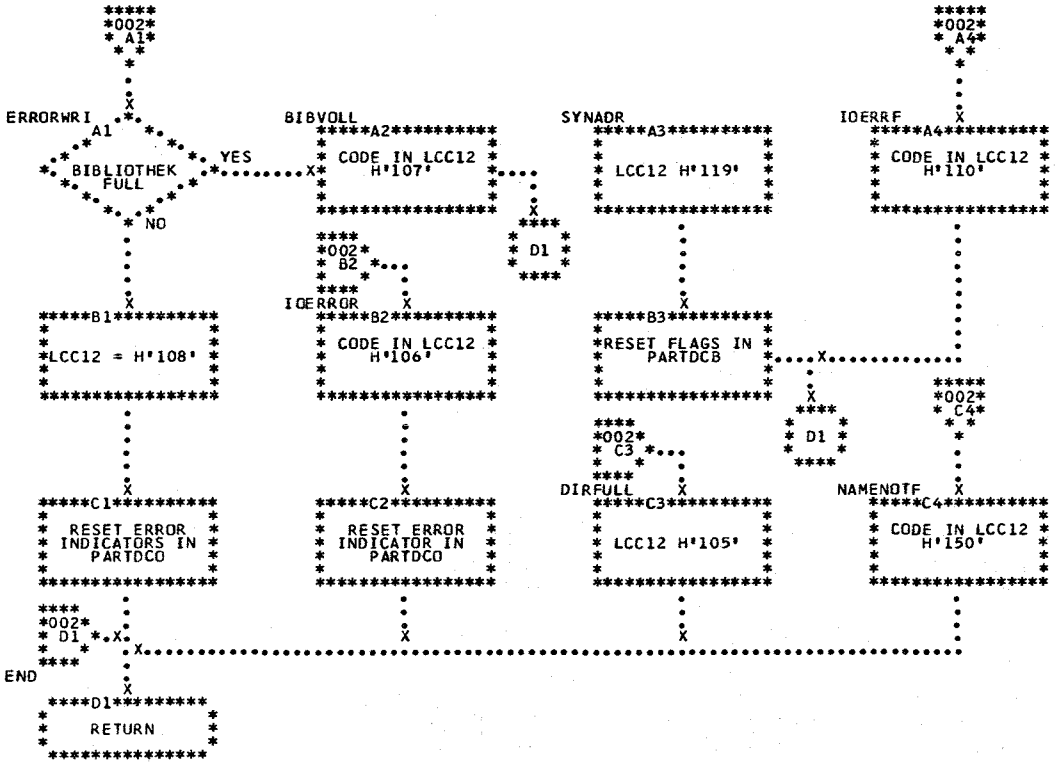
INPUT: ERRCODE IN FLAG, OUTPUT: MESS TO OPERATOR, FLAG CLEA RED

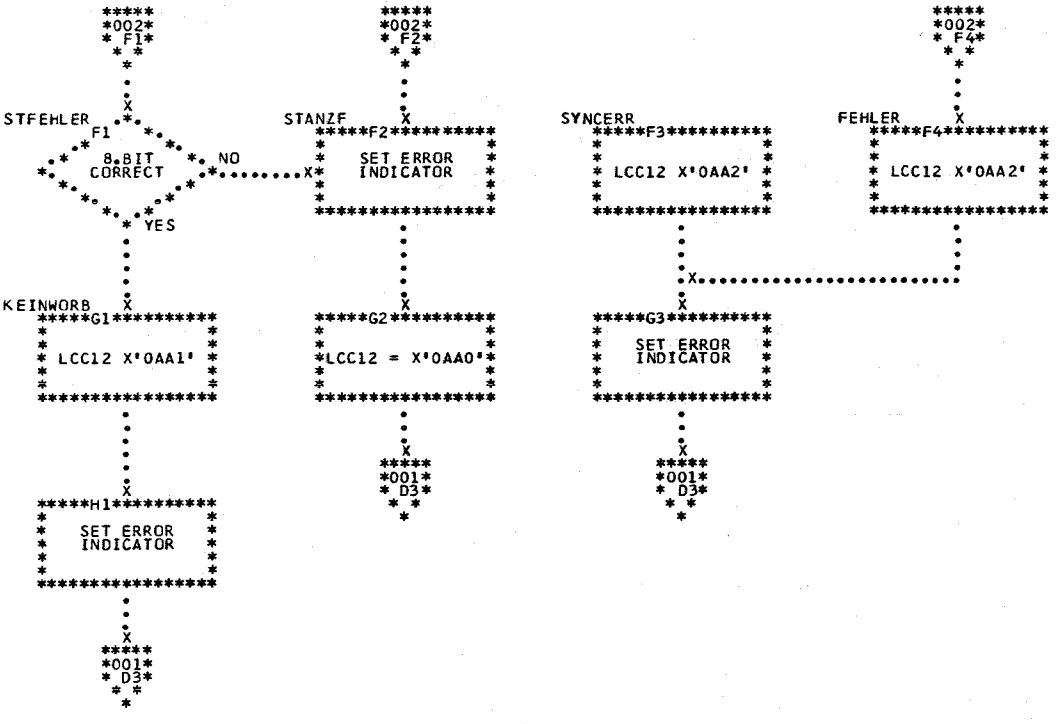


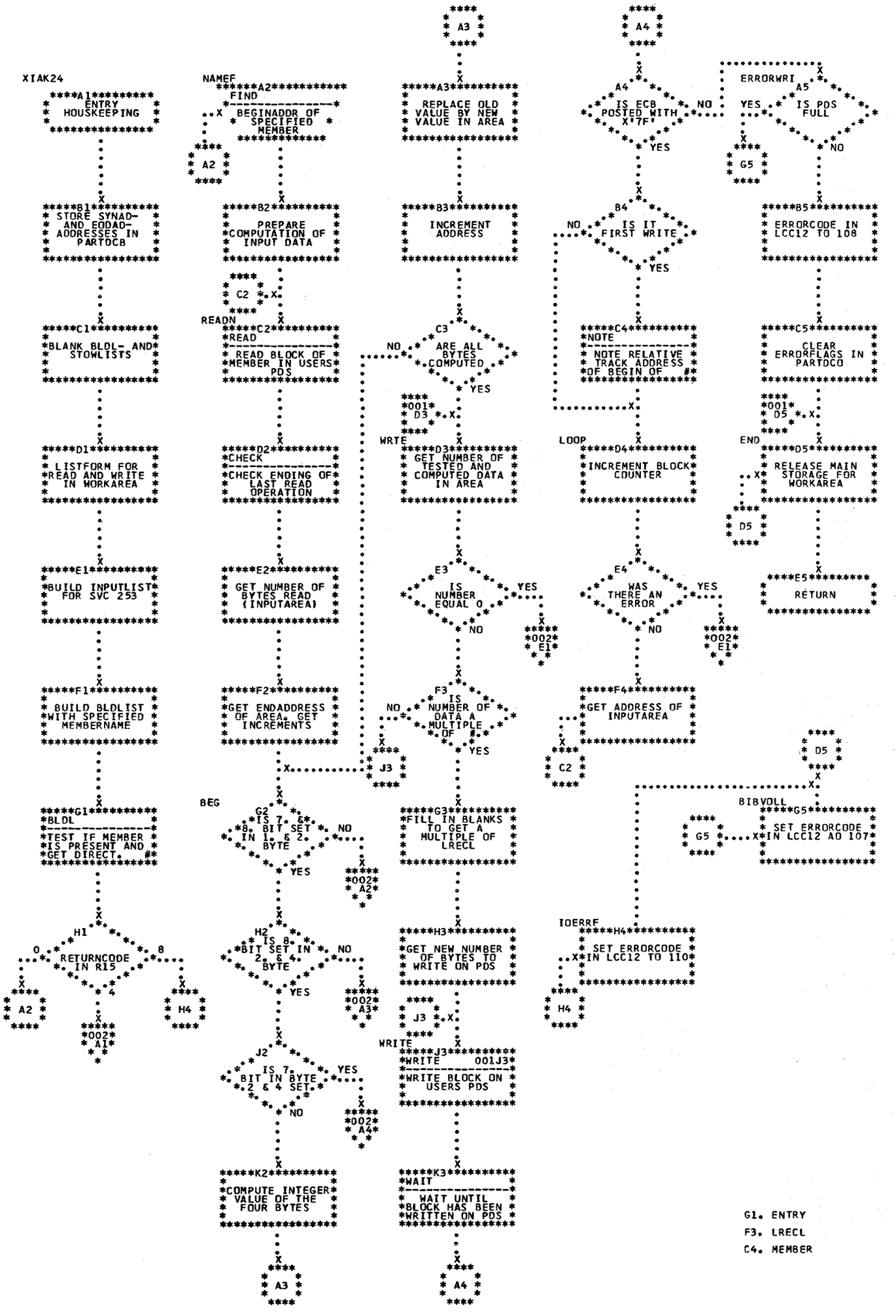

007 * A1*
TCPWEOJ
A1
ENTRY
B1
SWITCH SET FOR NOCATMSG
YES
NO
C1
CLOSE
NOCLOSE
D1
SEND JOBMSGM TO USER
E1
SWITCH SET FOR DADSM
YES
NO
F1
IEFQDELE
DELETE SYSOUT DATA SET IN SYSOUT QUEUE
G1
RETURNCODE
ZERO
NZER
H1
SET FLAG TO QMERR
007 * E2*
CHANGEA1
E2
IEFQMNQ2
ENQUEUE SYSOUT DATA SET IN QUEUE A
F2
RESTORE CLASS, SET SWITCH FOR DADSM OK
G2
RETURNCODE
NZER
ZERO
003 * H3*
EXITEOJ
J1
STOP WTR
YES
NO
K1
ANY MISFORTUNE ALONG THE WAY
YES
NO
006 * A1*
002 * A1*
STOPEOJ
J2
SET FLAG TO ABEND
006 * A1*

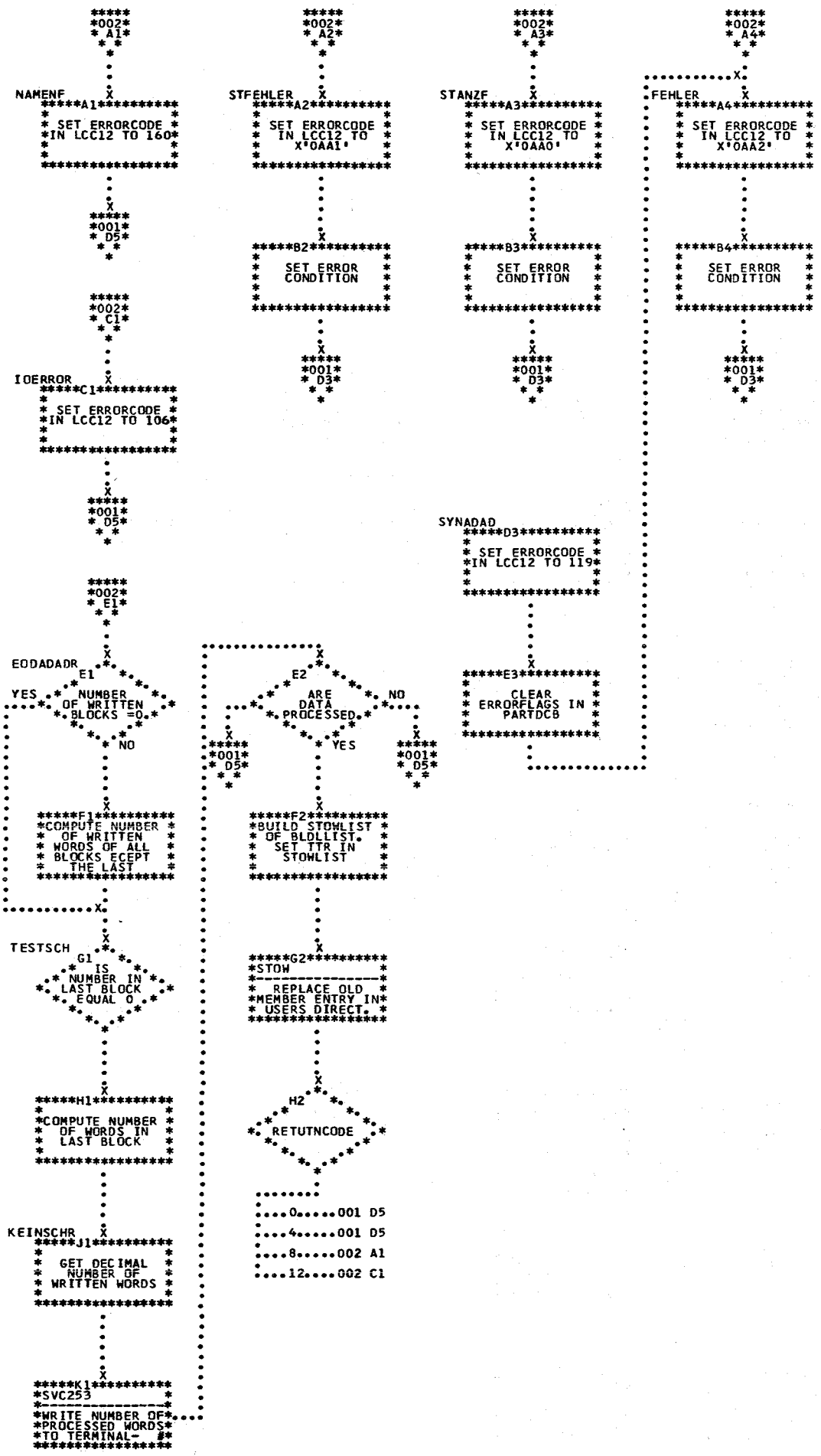

A1
ENTRY FROM TCP1BASE VIA LINK
P1
GET DDCB & DEB
HALTROUT
C1
INITIALIZE I/O-BLOCKS FOR HALT I/O
HALT1
D1
GET LINE IOB & LINE UCB
E1
LINE BUSY
NO
YES
A2
F1
GET START ADDRESS OF CHANNEL PROGRAM
G1
1.CCW PREPARE
NO
YES
A2
H1
PREPARE I/O-BLOCKS FOR HALT I/O
J1
IOHALT
K1
GET NEXT LBX
A2
HALT2
A2
NO
ALL LINES DONE
YES
D2
PRINT LINE ERROR BLOCK
CLOSE
C2
CLOSE SEQDCB
D2
GET NEXT LBX
E2
NO
IS IT KONBOX
YES
DEL1
F2
DELETE CH.END APP. PCI-APP. BUFFER ROUT. R/W-ROUT. I/O-DEVICE MODULE
FREECORE
G2
FREE SUBPOOL 1
H2
RETURN TO TCP1BASE


XIAK12
A1 ENTRY
B1 GET CORE STORAGE FOR WORKAREA
C1 STORE SYNAD- AND EODAD- ADDRESSES IN PARTDCB
D1 BLANK BLDL- AND STOWLISTS
E1 STORE LISTFORM OF READ AND WRITE IN WORKAREA
F1 BUILD BLDLLIST FOR MEMBER
G1 BLDL SEARCH SPECIFIED MEMBERNAME
H1 IS MEMBER IN USERS PDS
IOER
002 A4*
NAMEFOUN
J1 COMPUTE ENDADDRESS OF OUTPUTAREA
K1 NUMBER OF WRITED BLOCKS IS ZERO. BN
A2
A2 CLEAR ERROR INDICATOR
B2 FIND PARTDCB BLDLLIST
C2 READ READ A BLOCK OF THE SPECIFIED MEMBER
READNEU
D2 CHECK READDECB
E2 COMPUTE NUMBER OF BYTES READ
F2 GET ADDRESS OF LAST BYTE READ INTO INPUTAREA
G2 IS 7. BIT IN TESTED BYTE
002 F1*
MOVEANF
H2 8. BIT OF 2. BYTE SET
002 F2*
J2 7. BIT OF 2. BYTE SET
002 F4*
K2 COMPUTE INTEGER VALUE OF THE TWO BYTES ADDRESSED
A3
A3 GET DECIMAL VALUE AND STORE IT TO OUTPUTAREA
B3 INCREMENT ADDRESS OF INPUT-AREA
C3 IS OUTPUTAREA FULL
F4 *001* D3
AUSGVOLL
D3 COMPUTE NUMBER OF BYTES IN OUTPUTAREA
E3 IS NUMBER OF BYTES EQAL ZERO
F3 IS LRECL IN OUTPUTAREA EXACT
J3
G3 COMPUTE NUMBER OF BLANKS TO FILL IN OUTPUTAREA
H3 GET NUMBER OF BYTES IN OUTPUTAREA
J3
WRITEN
J3 WRITE WRITE BLOCK TO USERS PDS
K3 WAIT WAIT FOR COMPLETION OF WRITE
A4
A4 WRITE-ECB POSTED WITH '7F'
002 A1*
B4 WAS IT FIRST WRITE
C4 NOTE NOTE RELATIVE TRACK ADDRESS OF WRITTEN #
LOOPA
D4 INCREMENT NUMBER OF WRITTEN BLOCKS
E4 ARE ERRORS IN CHECKED INPUT DATA
XA
F4 ARE ALL INPUT DATA COMPUTED
G2
J3
H4 RETURNCODE R15
0 002 D1
4 002 D1
8 002 C4
12 002 C3
16 002 B2
B5
EODADR
K4 IS NUMBER OF WRITTEN BLOCKS ZERO
A5
A5 COMPUTE NUMBER OF DEC. NUMBERS WRITTEN IN FULL BLOCKS
B5 NUMBER OF BYTES IN LAST BLOCK EQUAL ZERO
TESTUB
C5 COMPUTE NUMBER OF WORDS IN LAST WRITTEN BLOCK
KEINSCHR
D5 MOVE COMPUTED NUMBER IN INPUTLIST FOR SVC 253
E5 SVC253 PRINT TO TERM. OPERATOR NUMBER OF TESTED WORDS
F5 NUMBER OF WORDS EQUAL ZERO
002 D1*
ENDDIR
G5 BUILD STOWLIST TTR OF MEMBERNAME IN OLD LIST
H5 STOW REPLACE OLD ENTRY FOR THE MEMBER
C4. MEMBER

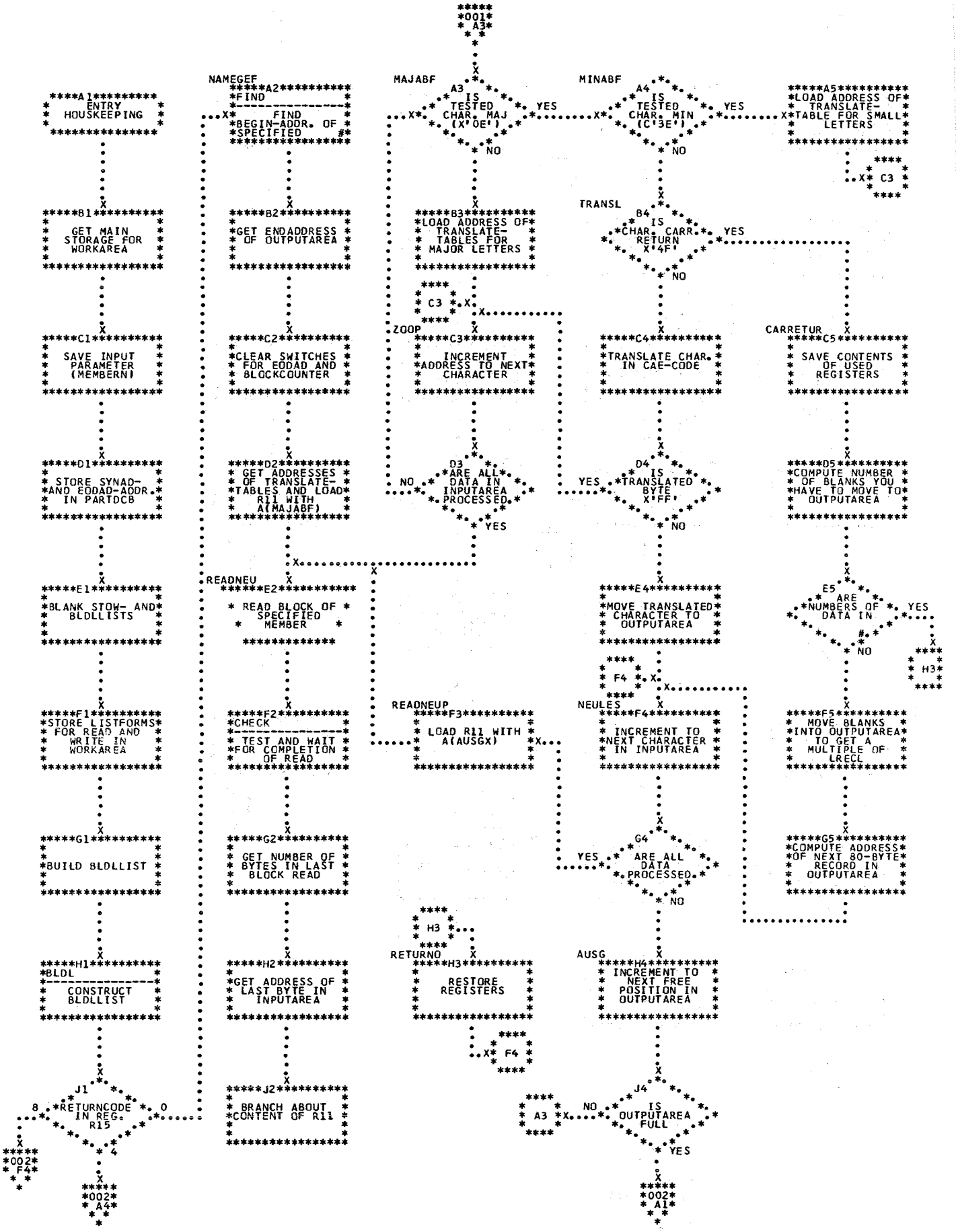

002
A1
ERRORWRI
A1
BIBLIOTHEK FULL
YES
NO
BIBVOLL
A2
CODE IN LCC12
H'107'
002
B2
IOERROR
D1
SYNADR
A3
LCC12 H'119'
IOERRF
A4
CODE IN LCC12
H'110'
B1
LCC12 = H'108'
B2
CODE IN LCC12
H'106'
B3
RESET FLAGS IN
PARTDCB
002
C4
C1
RESET ERROR
INDICATORS IN
PARTDCO
C2
RESET ERROR
INDICATOR IN
PARTDCO
002
C3
DIRFULL
C3
LCC12 H'105'
NAMENOTF
C4
CODE IN LCC12
H'150'
002
D1
END
D1
RETURN


002
F1
STFEHLER
F1
8.BIT CORRECT
NO
YES
STANZF
F2
SET ERROR INDICATOR
002
F2
SYNCERR
F3
LCC12 X'0AA2'
FEHLER
F4
LCC12 X'0AA2'
002
F4
KEINWORB
G1
LCC12 X'0AA1'
G2
LCC12 = X'0AA0'
G3
SET ERROR INDICATOR
SET ERROR INDICATOR
001
D3
001
D3
001
D3


XIAK24
A1 ENTRY HOUSKEEPING
B1 STORE SYNAD- AND EODAD- ADDRESSES IN PARTDCB
C1 BLANK BLDL- AND STOWLISTS
D1 LISTFORM FOR READ AND WRITE IN WORKAREA
E1 BUILD INPUTLIST FOR SVC 253
F1 BUILD BLDLIST WITH SPECIFIED MEMBERNAME
G1 BLDL
TEST IF MEMBER IS PRESENT AND GET DIRECT. #
H1 RETURNCODE IN R15
0
4
8
A2
H4
002 A1
NAMEF
A2 FIND
BEGINADDR OF SPECIFIED MEMBER
B2 PREPARE COMPUTATION OF INPUT DATA
C2
READN
C2 READ
READ BLOCK OF MEMBER IN USERS PDS
D2 CHECK
CHECK ENDING OF LAST READ OPERATION
E2 GET NUMBER OF BYTES READ (INPUTAREA)
F2 GET ENDADDRESS OF AREA, GET INCREMENTS
BEG
G2 IS 7 & 8 BIT SET IN 1 & 2 BYTE
YES
NO
002 A2
H2 IS 8 BIT SET IN 2 & 4 BYTE
YES
NO
002 A3
J2 IS 7 BIT IN BYTE 2 & 4 SET
YES
NO
002 A4
K2 COMPUTE INTEGER VALUE OF THE FOUR BYTES
A3
A3 REPLACE OLD VALUE BY NEW VALUE IN AREA
B3 INCREMENT ADDRESS
C3 ARE ALL BYTES COMPUTED
YES
NO
001 D3
WRTE
D3 GET NUMBER OF TESTED AND COMPUTED DATA IN AREA
E3 IS NUMBER EQUAL 0
YES
NO
002 E1
F3 IS NUMBER OF DATA A MULTIPLE OF #
YES
NO
J3
G3 FILL IN BLANKS TO GET A MULTIPLE OF LRECL
H3 GET NEW NUMBER OF BYTES TO WRITE ON PDS
J3
WRITE
J3 WRITE 001J3
WRITE BLOCK ON USERS PDS
K3 WAIT
WAIT UNTIL BLOCK HAS BEEN WRITTEN ON PDS
A4
A4 IS ECB POSTED WITH X'7F'
YES
NO
B4 IS IT FIRST WRITE
YES
NO
C4 NOTE
NOTE RELATIVE TRACK ADDRESS OF BEGIN OF #
LOOP
D4 INCREMENT BLOCK COUNTER
E4 WAS THERE AN ERROR
YES
NO
002 E1
F4 GET ADDRESS OF INPUTAREA
C2
IOERRF
H4 SET ERRORCODE IN LCC12 TO 110
H4
ERRORWRI
A5 IS PDS FULL
YES
NO
G5
B5 ERRORCODE IN LCC12 TO 108
C5 CLEAR ERRORFLAGS IN PARTDCB
END
D5 RELEASE MAIN STORAGE FOR WORKAREA
D5
E5 RETURN
D5
BIBVOLL
G5 SET ERRORCODE IN LCC12 AS 107
G5
G1. ENTRY
F3. LRECL
C4. MEMBER

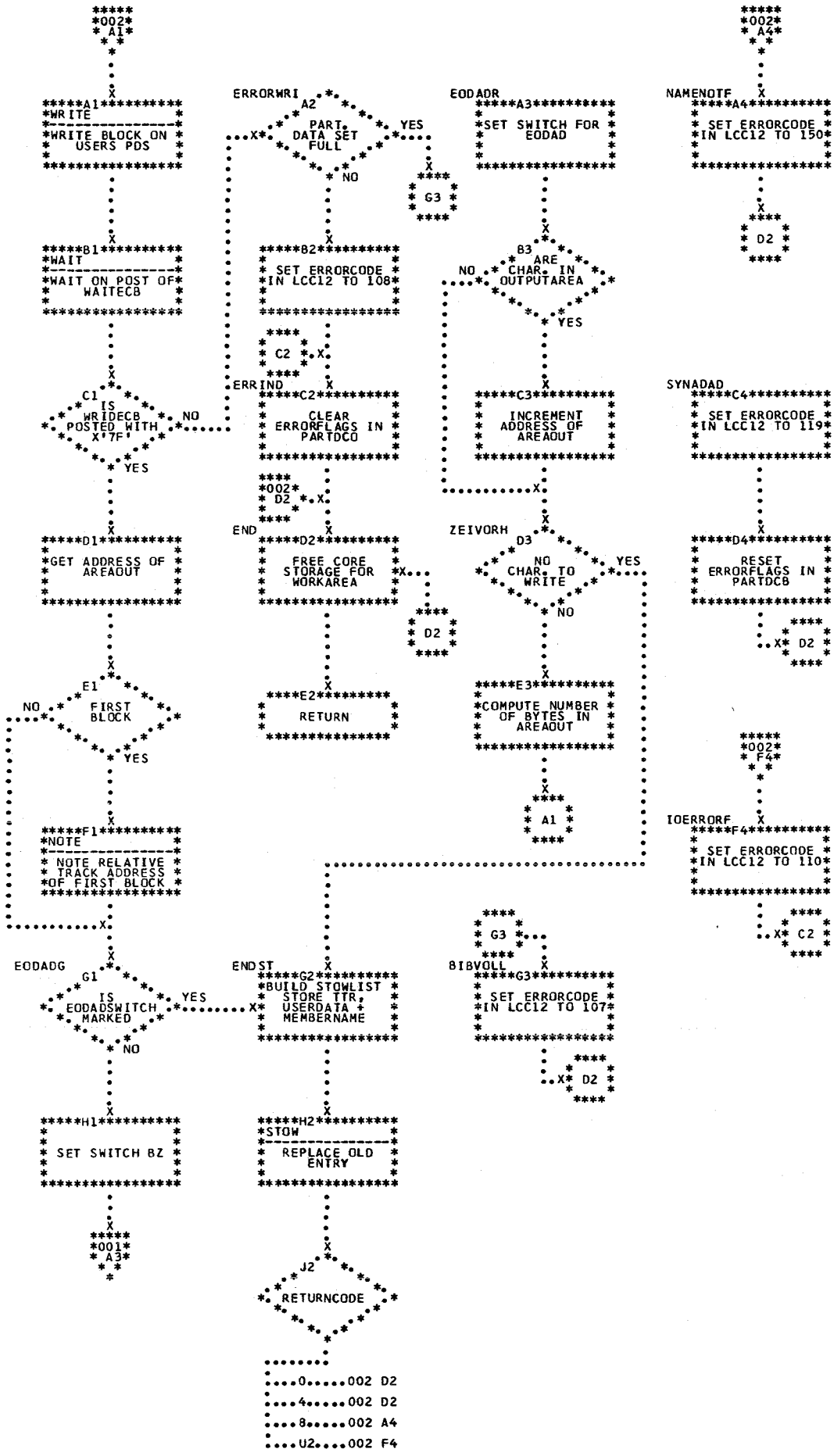

002 A1
NAMENF
A1
SET ERRORCODE IN LCC12 TO 160
001 D5
002 A2
STFEHLER
A2
SET ERRORCODE IN LCC12 TO X'0AA1'
B2
SET ERROR CONDITION
001 D3
002 A3
STANZF
A3
SET ERRORCODE IN LCC12 TO X'0AA0'
B3
SET ERROR CONDITION
001 D3
002 A4
FEHLER
A4
SET ERRORCODE IN LCC12 TO X'0AA2'
B4
SET ERROR CONDITION
001 D3
002 C1
IOERROR
C1
SET ERRORCODE IN LCC12 TO 106
001 D5
002 E1
EODADADR
E1
YES
NUMBER OF WRITTEN BLOCKS =0
NO
F1
COMPUTE NUMBER OF WRITTEN WORDS OF ALL BLOCKS ECEPT THE LAST
TESTSCH
G1
IS NUMBER IN LAST BLOCK EQUAL 0
H1
COMPUTE NUMBER OF WORDS IN LAST BLOCK
KEINSCHR
J1
GET DECIMAL NUMBER OF WRITTEN WORDS
K1
SVC253
WRITE NUMBER OF PROCESSED WORDS TO TERMINAL- ##
E2
ARE DATA PROCESSED
NO
YES
001 D5
001 D5
F2
BUILD STOWLIST OF BLDLLIST. SET TTR IN STOWLIST
G2
STOW
REPLACE OLD MEMBER ENTRY IN USERS DIRECT.
H2
RETUTNCODE
0 001 D5
4 001 D5
8 002 A1
12 002 C1
SYNADAD
D3
SET ERRORCODE IN LCC12 TO 119
E3
CLEAR ERRORFLAGS IN PARTDCB

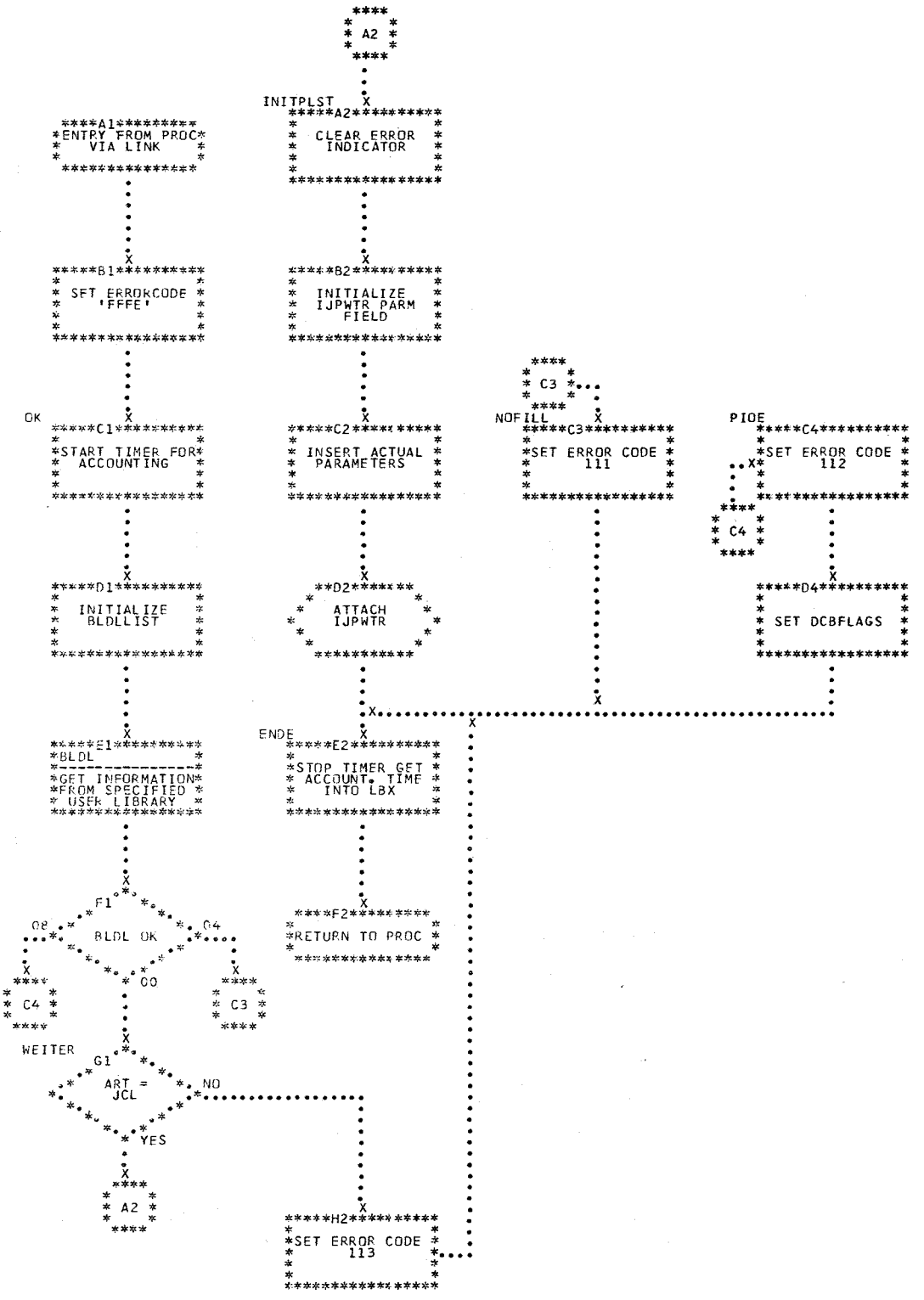
K1. OPERATOR

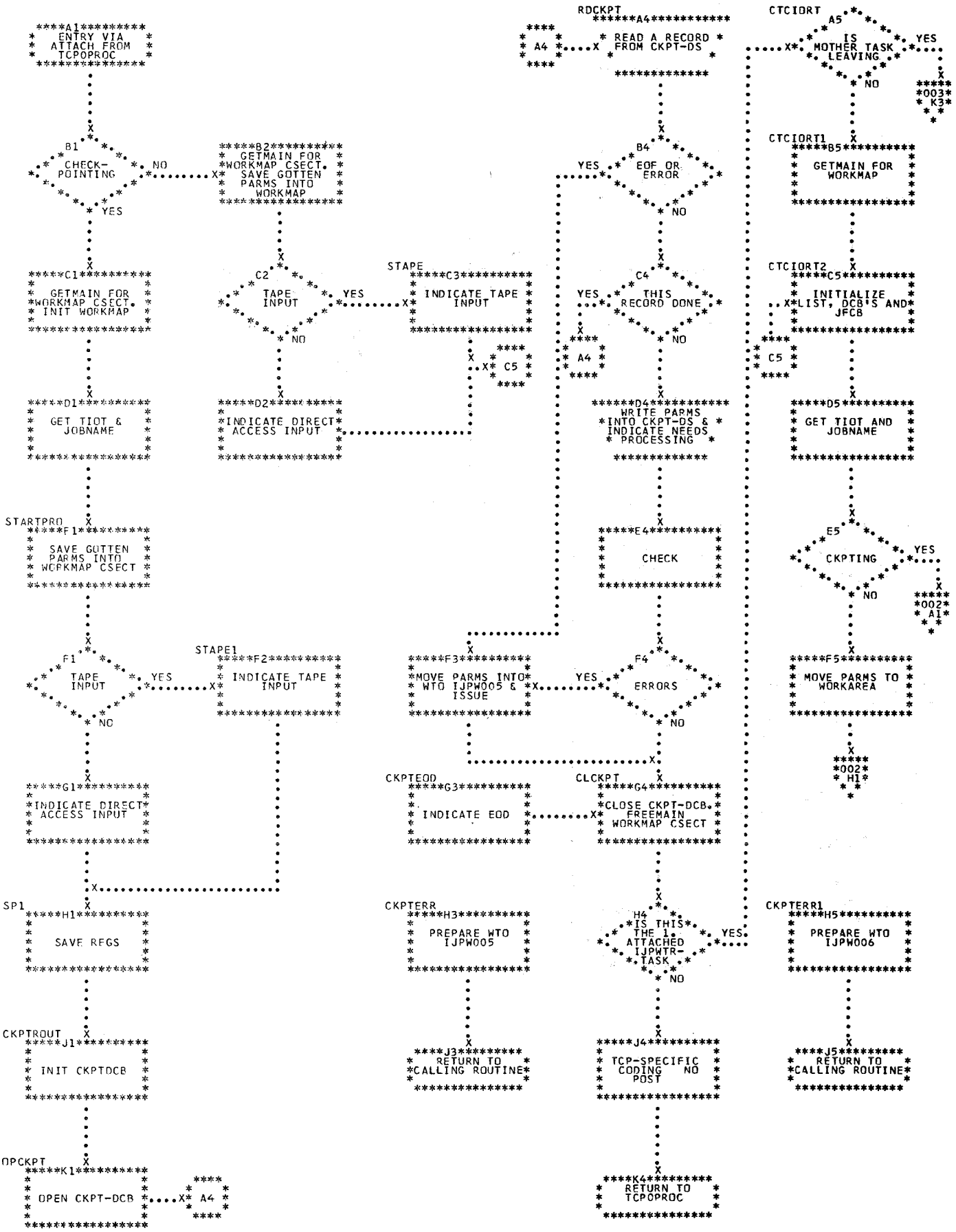


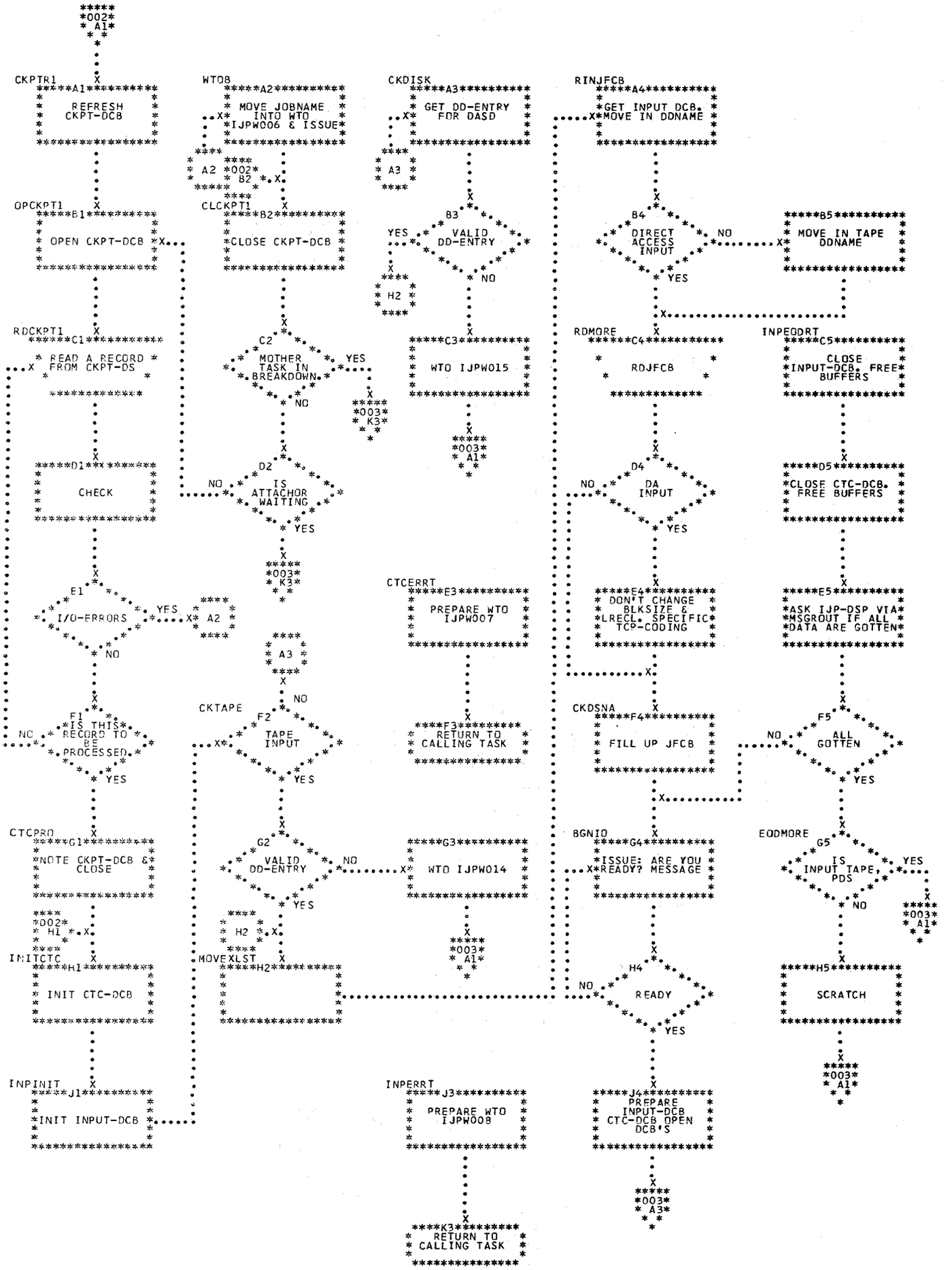
A2. MEMBER

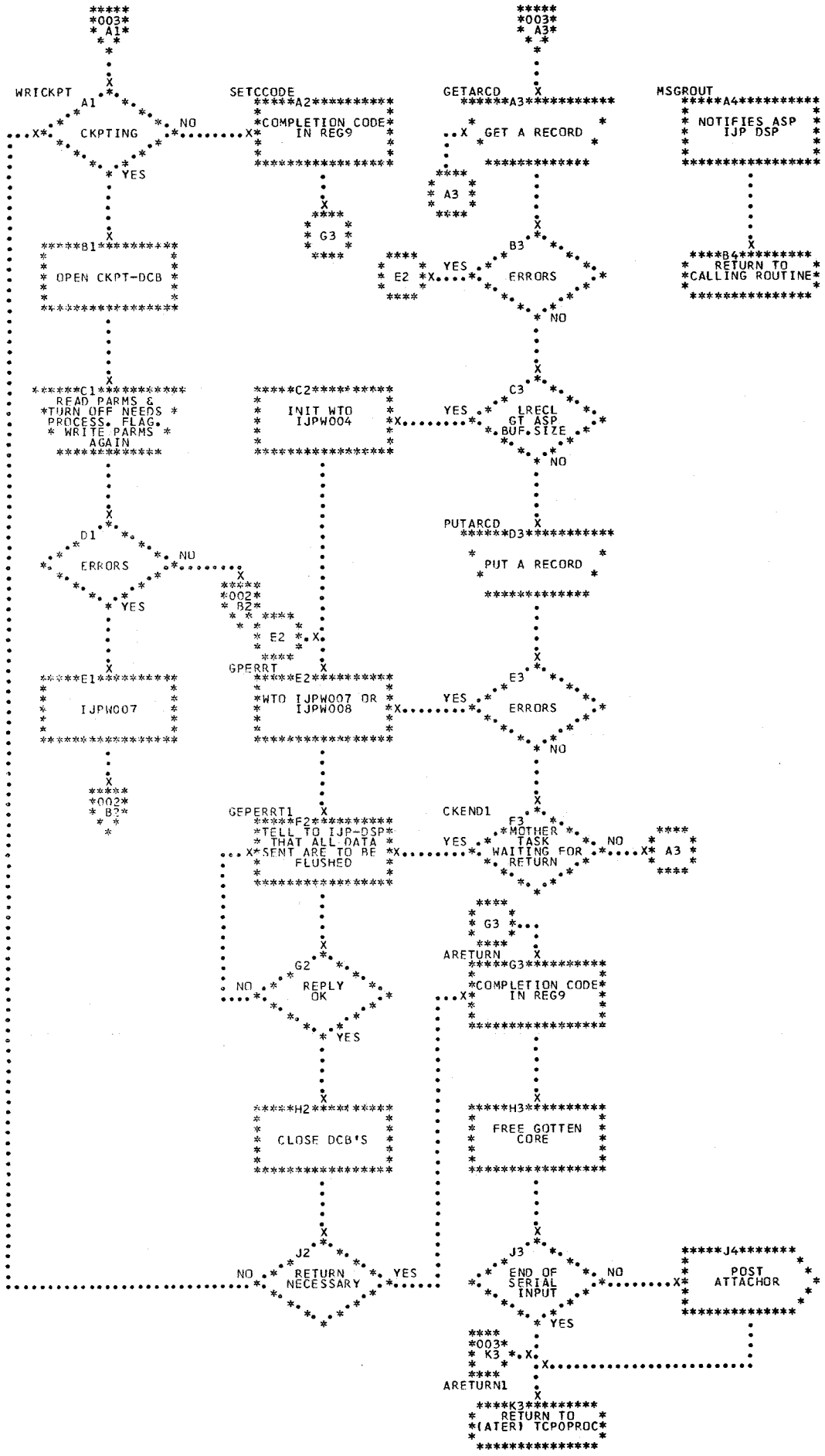
E5. OUTPUTAREA A MULTIPLE OF LRECL


002 A1
A1 WRITE
WRITE BLOCK ON USERS PDS
B1 WAIT
WAIT ON POST OF WAITECB
C1 IS WRIDECB POSTED WITH X'7F'?
NO
YES
D1 GET ADDRESS OF AREAOUT
E1 FIRST BLOCK?
NO
YES
F1 NOTE
NOTE RELATIVE TRACK ADDRESS OF FIRST BLOCK
EODADG
G1 IS EODADSWITCH MARKED?
YES
NO
H1 SET SWITCH BZ
001 A3
ERRORWRI
A2 PART. DATA SET FULL?
YES
NO
G3
B2 SET ERRORCODE IN LCC12 TO 108
C2
ERRIND
C2 CLEAR ERRORFLAGS IN PARTDCB
002 D2
END
D2 FREE CORE STORAGE FOR WORKAREA
D2
E2 RETURN
ENDST
G2 BUILD STOWLIST STORE TTR, USERDATA, MEMBERNAME
H2 STOW REPLACE OLD ENTRY
J2 RETURNCODE
0 002 D2
4 002 D2
8 002 A4
U2 002 F4
EODADR
A3 SET SWITCH FOR EODAD
B3 ARE CHAR. IN OUTPUTAREA?
NO
YES
C3 INCREMENT ADDRESS OF AREAOUT
ZEIVORH
D3 NO CHAR. TO WRITE?
YES
NO
E3 COMPUTE NUMBER OF BYTES IN AREAOUT
A1
BIBVOLL
G3 SET ERRORCODE IN LCC12 TO 107
D2
002 A4
NAMENOTF
A4 SET ERRORCODE IN LCC12 TO 150
D2
SYNADAD
C4 SET ERRORCODE IN LCC12 TO 119
D4 RESET ERRORFLAGS IN PARTDCB
D2
002 F4
IOERRORF
F4 SET ERRORCODE IN LCC12 TO 110
C2

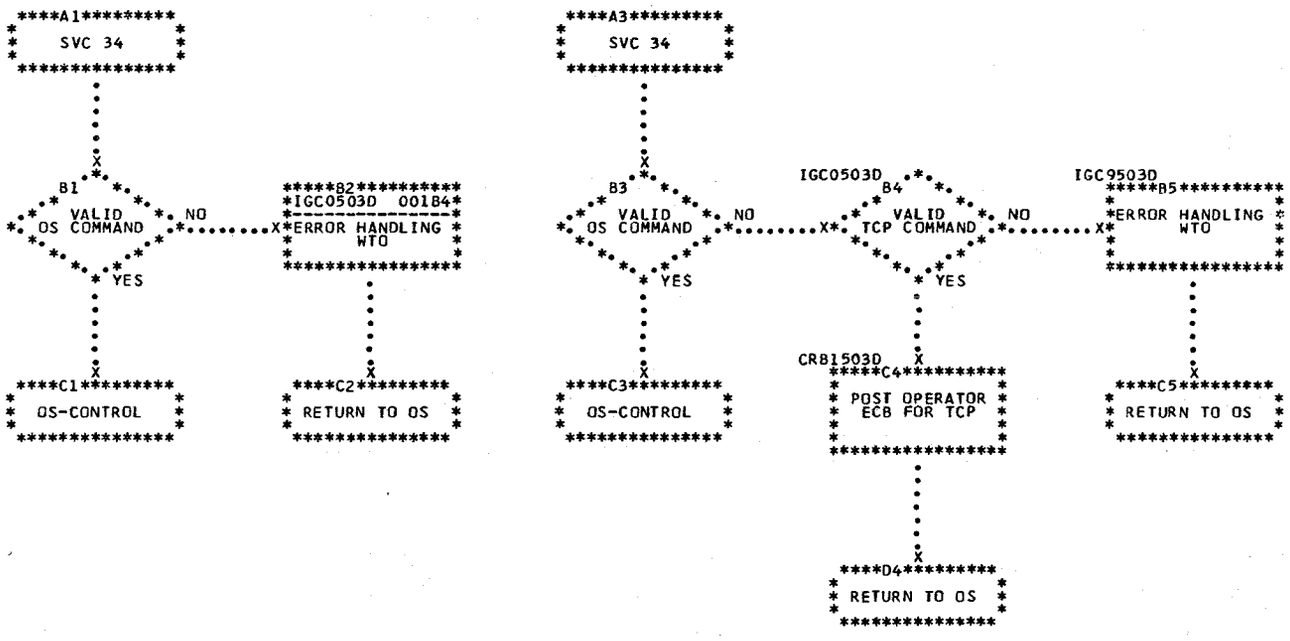

A2
INITPLST
A2
CLEAR ERROR INDICATOR
A1
ENTRY FROM PROC VIA LINK
B1
SET ERROKCODE 'FFFE'
B2
INITIALIZE IJPWTR PARM FIELD
OK
C1
START TIMER FOR ACCOUNTING
C2
INSERT ACTUAL PARAMETERS
C3
NOFILL
SET ERROR CODE 111
C4
PIOE
SET ERROR CODE 112
D1
INITIALIZE BLDLLIST
D2
ATTACH IJPWTR
D4
SET DCBFLAGS
ENDE
E1
BLDL
GET INFORMATION FROM SPECIFIED USER LIBRARY
E2
STOP TIMER GET ACCOUNT. TIME INTO LBX
F1
BLDL OK
00
04
08
F2
RETURN TO PROC
WEITER
G1
ART = JCL
NO
YES
H2
SET ERROR CODE 113

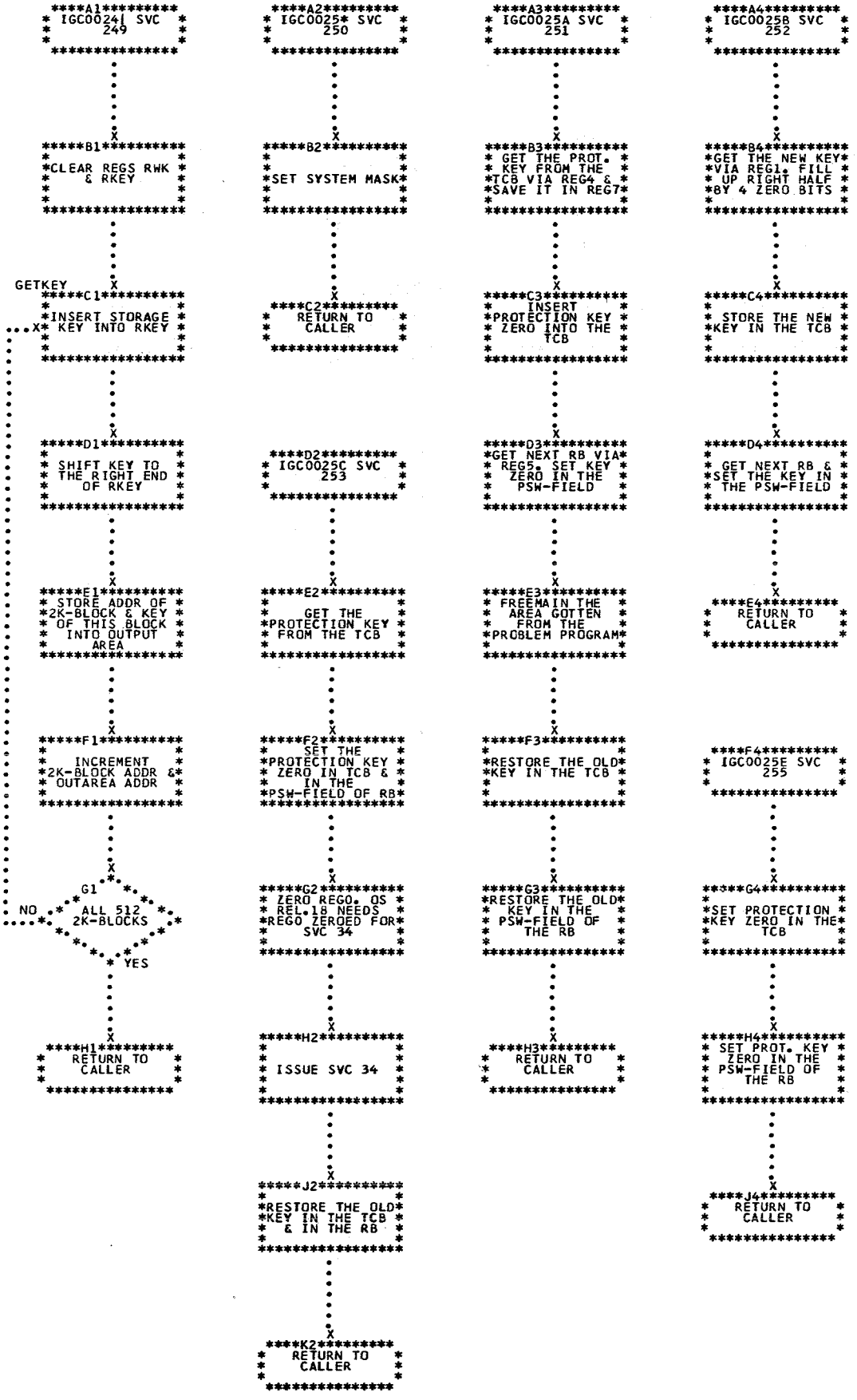

A1
ENTRY VIA ATTACH FROM TCPOPROC
B1
CHECK-POINTING
NO
YES
B2
GETMAIN FOR WORKMAP CSECT. SAVE GOTTEN PARMS INTO WORKMAP
C1
GETMAIN FOR WORKMAP CSECT. INIT WORKMAP
C2
TAPE INPUT
YES
NO
STAPE
C3
INDICATE TAPE INPUT
C5
D1
GET TIOT & JOBNAME
D2
INDICATE DIRECT ACCESS INPUT
STARTPRO
F1
SAVE GOTTEN PARMS INTO WORKMAP CSECT
F1
TAPE INPUT
YES
NO
STAPE1
F2
INDICATE TAPE INPUT
G1
INDICATE DIRECT ACCESS INPUT
SP1
H1
SAVE REGS
CKPTROUT
J1
INIT CKPTDCB
OPCKPT
K1
OPEN CKPT-DCB
A4
RDCKPT
A4
READ A RECORD FROM CKPT-DS
B4
EOF OR ERROR
YES
NO
C4
THIS RECORD DONE
YES
NO
A4
D4
WRITE PARMS INTO CKPT-DS & INDICATE NEEDS PROCESSING
E4
CHECK
F4
ERRORS
YES
NO
F3
MOVE PARMS INTO WTO IJPW005 & ISSUE
CKPTEOD
G3
INDICATE EOD
CLCKPT
G4
CLOSE CKPT-DCB FREEMAIN WORKMAP CSECT
H4
IS THIS THE 1. ATTACHED IJPWTR-TASK
YES
NO
CKPTERR
H3
PREPARE WTO IJPW005
J3
RETURN TO CALLING ROUTINE
J4
TCP-SPECIFIC CODING NO POST
K4
RETURN TO TCPOPROC
CTCIORT
A5
IS MOTHER TASK LEAVING
YES
NO
003 K3
CTCIORT1
B5
GETMAIN FOR WORKMAP
CTCIORT2
C5
INITIALIZE LIST, DCB'S AND JFCB
A4
C5
D5
GET TIOT AND JOBNAME
E5
CKPTING
YES
NO
002 A1
F5
MOVE PARMS TO WORKAREA
002 H1
CKPTERR1
H5
PREPARE WTO IJPW006
J5
RETURN TO CALLING ROUTINE

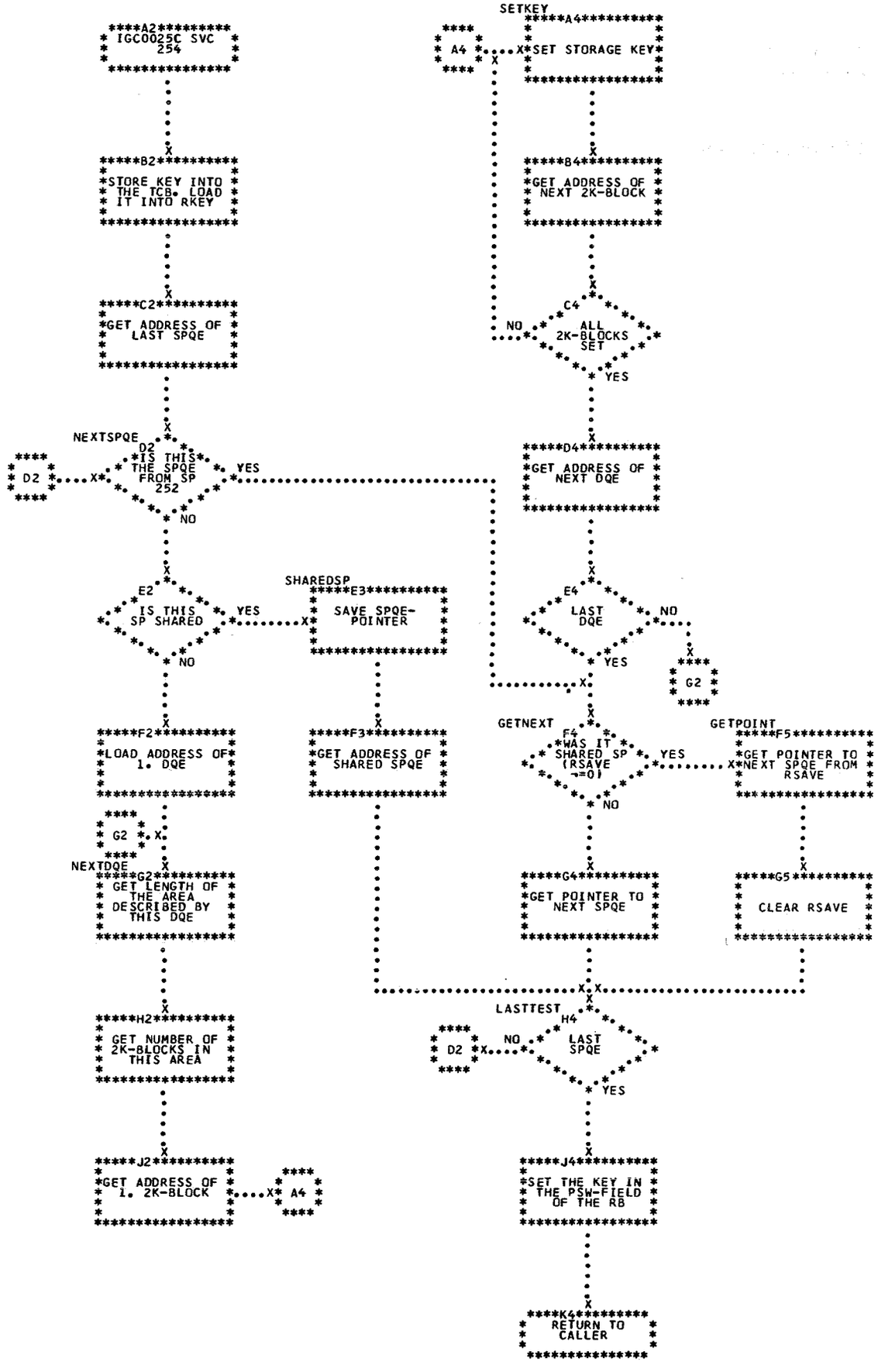

002 A1
CKPTR1
A1 REFRESH CKPT-DCB
OPCKPT1
B1 OPEN CKPT-DCB
RDCKPT1
C1 READ A RECORD FROM CKPT-DS
D1 CHECK
E1 I/O-ERRORS
YES
NO
F1 IS THIS RECORD TO BE PROCESSED
CTCPRO
G1 NOTE CKPT-DCB & CLOSE
002 H1
INITCTC
H1 INIT CTC-DCB
INPINIT
J1 INIT INPUT-DCB
WTO8
A2 MOVE JOBNAME INTO WTO IJPW006 & ISSUE
A2 *002* B2
CLCKPT1
B2 CLOSE CKPT-DCB
C2 MOTHER TASK IN BREAKDOWN
003 K3
D2 IS ATTACHOR WAITING
003 K3
A3
CKTAPE
F2 TAPE INPUT
G2 VALID DD-ENTRY
A3 H2
MOVEXLST
H2
CKDISK
A3 GET DD-ENTRY FOR DASD
B3 VALID DD-ENTRY
H2
C3 WTO IJPW015
003 A1
CTCERRT
E3 PREPARE WTO IJPW007
F3 RETURN TO CALLING TASK
G3 WTO IJPW014
003 A1
INPERRT
J3 PREPARE WTO IJPW008
K3 RETURN TO CALLING TASK
RINJFCB
A4 GET INPUT DCB MOVE IN DDNAME
B4 DIRECT ACCESS INPUT
B5 MOVE IN TAPE DDNAME
RDMORE
C4 RDJFCB
D4 DA INPUT
E4 DON'T CHANGE BLKSIZE & LRECL SPECIFIC TCP-CODING
CKDSNA
F4 FILL UP JFCB
BGNID
G4 ISSUE: ARE YOU READY? MESSAGE
H4 READY
003 A3
J4 PREPARE INPUT-DCB CTC-DCB OPEN DCB'S
003 A3
INPEODRT
C5 CLOSE INPUT-DCB, FREE BUFFERS
D5 CLOSE CTC-DCB, FREE BUFFERS
E5 ASK IJP-DSP VIA MSGROUT IF ALL DATA ARE GOTTEN
F5 ALL GOTTEN
EODMORE
G5 IS INPUT TAPE, PDS
003 A1
H5 SCRATCH
003 A1


WRICKPT
A1
CKPTING
NO
YES
B1
OPEN CKPT-DCB
C1
READ PARMS &
TURN OFF NEEDS
PROCESS. FLAG.
WRITE PARMS
AGAIN
D1
ERRORS
NO
YES
E1
IJPW007
SETCCODE
A2
COMPLETION CODE
IN REG9
G3
C2
INIT WTO
IJPW004
B2
E2
GPERRT
WTO IJPW007 OR
IJPW008
GETARCD
A3
GET A RECORD
A3
B3
E2
YES
ERRORS
NO
C3
YES
LRECL
GT ASP
BUF. SIZE
NO
PUTARCD
D3
PUT A RECORD
E3
YES
ERRORS
NO
CKEND1
F3
MOTHER
TASK
WAITING FOR
RETURN
NO
A3
G3
ARETURN
G3
COMPLETION CODE
IN REG9
H3
FREE GOTTEN
CORE
J3
END OF
SERIAL
INPUT
NO
YES
K3
ARETURN1
K3
RETURN TO
(ATER) TCPOPROC
MSGROUT
A4
NOTIFIES ASP
IJP DSP
B4
RETURN TO
CALLING ROUTINE
F2
TELL TO IJP-DSP
THAT ALL DATA
SENT ARE TO BE
FLUSHED
G2
REPLY
OK
NO
YES
H2
CLOSE DCB'S
J2
RETURN
NECESSARY
NO
YES
J4
POST
ATTACHOR

CONSOLE INTERCEPT LOGIC
WITHOUT TCP-HANDLING (LEFT SIDE)
WITH TCP-HANDLING (RIGHT SIDE)




IGC00241 SVC 249
IGC00025* SVC 250
IGC00025A SVC 251
IGC00025B SVC 252
CLEAR REGS RWK & RKEY
SET SYSTEM MASK
GET THE PROT. KEY FROM THE TCB VIA REG4 & SAVE IT IN REG7
GET THE NEW KEY VIA REG1. FILL UP RIGHT HALF BY 4 ZERO BITS
GETKEY
INSERT STORAGE KEY INTO RKEY
RETURN TO CALLER
INSERT PROTECTION KEY ZERO INTO THE TCB
STORE THE NEW KEY IN THE TCB
SHIFT KEY TO THE RIGHT END OF RKEY
IGC00025C SVC 253
GET NEXT RB VIA REG5. SET KEY ZERO IN THE PSW-FIELD
GET NEXT RB & SET THE KEY IN THE PSW-FIELD
STORE ADDR OF 2K-BLOCK & KEY OF THIS BLOCK INTO OUTPUT AREA
GET THE PROTECTION KEY FROM THE TCB
FREEMAIN THE AREA GOTTEN FROM THE PROBLEM PROGRAM
RETURN TO CALLER
INCREMENT 2K-BLOCK ADDR & OUTAREA ADDR
SET THE PROTECTION KEY ZERO IN TCB & IN THE PSW-FIELD OF RB
RESTORE THE OLD KEY IN THE TCB
IGC00025E SVC 255
G1
NO
ALL 512 2K-BLOCKS
YES
ZERO REG0. OS REL.18 NEEDS REG0 ZEROED FOR SVC 34
RESTORE THE OLD KEY IN THE PSW-FIELD OF THE RB
SET PROTECTION KEY ZERO IN THE TCB
ISSUE SVC 34
SET PROT. KEY ZERO IN THE PSW-FIELD OF THE RB
RESTORE THE OLD KEY IN THE TCB & IN THE RB


A2
IGC0025C SVC 254
B2
STORE KEY INTO THE TCB. LOAD IT INTO RKEY
C2
GET ADDRESS OF LAST SPQE
NEXTSPQE
D2
IS THIS THE SPQE FROM SP 252
YES
NO
E2
IS THIS SP SHARED
YES
NO
SHAREDSP
E3
SAVE SPQE-POINTER
F2
LOAD ADDRESS OF 1. DQE
F3
GET ADDRESS OF SHARED SPQE
NEXTDQE
G2
GET LENGTH OF THE AREA DESCRIBED BY THIS DQE
H2
GET NUMBER OF 2K-BLOCKS IN THIS AREA
J2
GET ADDRESS OF 1. 2K-BLOCK
A4
SETKEY
A4
SET STORAGE KEY
B4
GET ADDRESS OF NEXT 2K-BLOCK
C4
ALL 2K-BLOCKS SET
NO
YES
D4
GET ADDRESS OF NEXT DQE
E4
LAST DQE
NO
YES
G2
GETNEXT
F4
HAS IT SHARED SP (RSAVE -=0)
YES
NO
G4
GET POINTER TO NEXT SPQE
GETPOINT
F5
GET POINTER TO NEXT SPQE FROM RSAVE
G5
CLEAR RSAVE
LASTTEST
H4
LAST SPQE
NO
YES
D2
J4
SET THE KEY IN THE PSW-FIELD OF THE RB
K4
RETURN TO CALLER

Anhang H

Literaturstellen

Literatur:

/1/ E. Holler
The Karlsruhe Modular Terminal System.
Proceedings of the Share European Association
Meeting (SEAS XIV)Grenoble, September 17-19, 1969

/2/ V. Haase, E. Holler
TCP - Ein flexibles Teilnehmersystem für experimentelle Forschungsaufgaben. 2. Intern. Kongress "Datenverarbeitung im europäischen Raum", Baden/Wien, 29.9. - 3.10.1969
Vol. 2 (1969) S. 97-105

/3/ IBM System/360 Operating System
Concepts and Facilities

System Reference Library
Form C 28-6535

/4/ IBM System/360 Operating System MVT Supervisor

Program Logic Manual
Form Y 28-6659

/5/ E. Holler
Teilnehmersysteme und Mehrrechnersysteme

Vortrag im Rahmen der "Einführung in die Datenverarbeitung" Gesellschaft für Kernforschung Karlsruhe (Schule für Kerntechnik) Oktober 1970

/6/ IBM System/360 Operating System
MVT Job Management

Program Logic Manual
Form Y 28-6660

/7/ IBM System/360 Operating System
BTAM Basic Telecommunication Access Method

Program Logic Manual
Form Y 30-2001

/8/ IBM System/360 Operating System
System Control Blocks

System Reference Library
Form C 28-6628

/9/ SIEMENS

Siemens Transdata
Dialogstation Transdata 8525-101
Beschreibung

/10/ IBM System/360 Operating System
BTAM Basic Telecommunication Access Method
System Reference Library
Oder No. GC 30-2004

/11/ D. Jenet, G. Würz
MIDAS - Meßprogramm © 1968
unveröffentlicht

/12/ S. Heine
PLOTA
Ein verallgemeinertes Plot-Programm.
unveröffentlicht

/13/ G. Fleck, R. Merkel
Hardware - und Software-Anschluß von
KFK - Sichtgeräten
Externer Bericht in Vorbereitung

/14/ IBM System/360
Attached Support Processor System (ASP)
Version 2
System Programmers Manual
Application Program
Form GH 20-0323

/15/ IBM System/360
Attached Support Processor System (ASP)
Version 2
System Manual
Application Program
Form Y 20-0305