DFVLR

Deutsche Forschungs- und
Versuchsanstalt
für Luft- und Raumfahrt

LOGREA: Ein Programm zur formatierten Textausgabe

Thomas C. Henderson

Interner Bericht

Institut für
Nachrichtentechnik

Bl.:
IB. Nr.: NE-NT-D-37-80

LOGREA: Ein Programm zur formatierten Textausgabe

Thomas C. Henderson

Freigabe: Die Bearbeiter: Unterschriften:

Thomas C. Henderson

Der Abteilungsleiter

Der stellv. Institutsdirektor :

Der Institutsdirektor :

Dieser Bericht enthält 64 Blatt davon
____ Bilder
____ Diagramme

LOGREA: Ein Programm zur formatierten Textausgabe

Thomas C. Henderson

DFVLR Oberpfaffenhofen

August 10, 1980

Inhaltsverzeichnis

Einleitung

LOGREA ist ein Programm zur formatierten Textausgabe, das speziell fuer die DIBIAS Gruppe entwickelt wurde. Ein solches Programm hilft bei der Entwicklung und bei der Erzeugung von Berichten und Dokumentation, indem es eine schnelle Textmanipulation und Formataenderung erlaubt. Zum Beispiel: Raender, Einruecken, leere Zeilen usw. koennen einfach veraendert werden. Ausserdem, wenn der Eingabetext veraendert ist, formatiert wieder der Lauf von LOGREA automatisch den ganzen Bericht.

Dieser Bericht besteht aus 2 Teilen: erster Teil, eine Eroeterung ueber LOGREA, was den Benutzer betrifft, und zweiter Teil, eine Eroeterung ueber die Programmierungseinzelheiten. Die Eroeterung ist gueltig fuer die zwei Rechner INTERDATA 8/32 und AMDAHL. Eine LOGREA-Benutzerbeschreibung ist immer an der INTERDATA 8/32 auf der LOGREA.MAN/G Datei vorhanden. Die AMDAHL Einzelheiten sind in Anhang B erklaert.

Teil I. Wie wird LOGREA benutzt

LOGREA wurde fuer die DIBIAS-Gruppe entworfen. Es laeuft im RTOS/MTM auf dem INTERDATA 8/32 Rechner. Die MTM-Konfiguration von LOGREA ist in Bild 1 schematisch dargestellt.

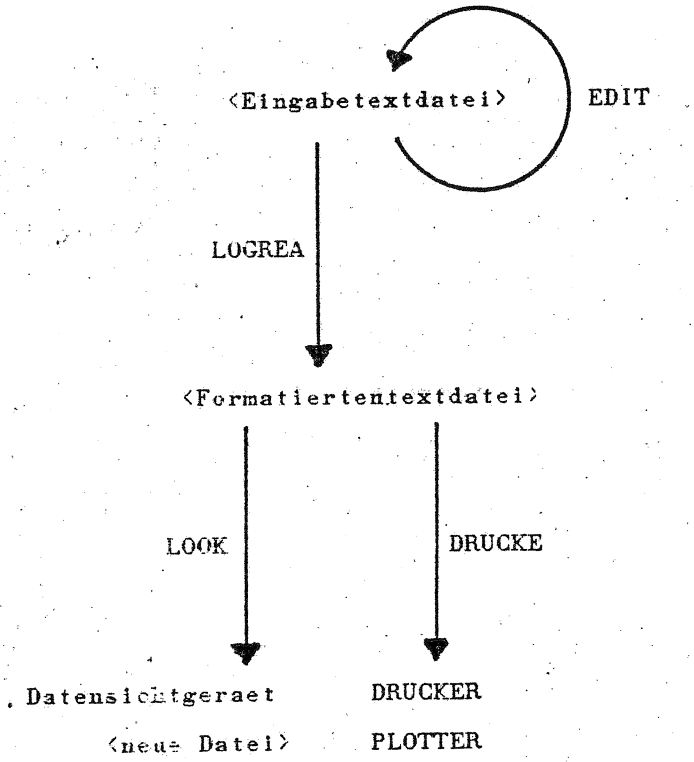


Bild 1 - Die LOGREA Konfiguration

Die <Eingabetextdatei> wird mit EDIT geschaffen oder veraendert (EDIT ist eine CSS-Datei, die den Editor in Gang setzt). Die <Eingabetextdatei> wird zeilenweise abgearbeitet und erhaelt zwei verschiedene Arten von Daten:

- LOGREA Kommandos und
- Textdaten.

Das bedeutet, dass jede Zeile entweder ein Kommando oder irgendwelche

Textdaten sind, die in der <Formatiertentextdatei> erscheinen sollen. Die LOGREA-Kommandos werden in der <Formatiertentextdatei> nicht erscheinen. Die Kommandos koennen irgendwo in der <Eingabetextdatei> eingestreut werden und kontrollieren die Erscheinung der <Formatiertentextdatei>.

Der Text

Der zu verarbeitende Text kann auch Gross- und Kleinbuchstaben enthalten. Jedoch gibt es zwei Arten von Ausnahmen. Zuerst gibt es ein Vereinbarung, so dass die Kommandos von den Textdaten unterschieden werden koennen. Die Kommandos beginnen alle mit einem .-Zeichen, und diese Steuerzeichen muss stets am Anfang der Zeile stehen. Zweitens, jede Zeile, die mit einem Blank beginnt, bedeutet ein neuer Absatz. Sonst koennen Textdaten irgendeine Zeichenkette sein. Wenn die <Eingabetextdatei> ohne Kommandos ist, gibt es Defaultwerte fuer die Raender, usw. Diese Defaultwerte sind in Anhang C gegeben.

Die Kommandos

Die Kommandos erlauben die Wahl der Raender, Einruecken, usw. Die Textdaten bestehen aus der zu verarbeitenden <Eingabetextdatei> und darin eingestreuten Kommandos fuer LOGREA, aber die Kommandos werden in der <Formatiertentextdatei> nicht erscheinen. Nur ein Kommando darf pro Eingabezeile stehen, und es muss so aussehen:

```
Spalte :  1   2 3    4 5...
          . <Kommando> <Parameter>
```

Die Kommandos beginnen alle mit einem Steuerzeichen (hier die .-Zeichen), und sie muessen stets im Spalte 1 der Zeile stehen. Alle

Kommandos bestehen aus 2 Buchstaben und sind von dem <Parameter> 1 Blank entfernt. Ein <Parameter> kann nur eine ganze Zahl oder Blanks sein. Das <Kommando> darf nur aus Kleinbuchstaben bestehen, d.h. a,b,c, ..,z. Die gueltigen Kommandos sind:

.bl n n Zeilen Vorschub. (Achtung! .sp 2 und dann .bl 2 gibt 4 leere Zeilen.)

.bo n (Defaultwert ist n=64.) Definiert die letzte Zeile am Ende der Seite.

.br Vorschub auf die naechste Zeile.

.ce Die folgende Zeile ist zwischen dem linken und rechten Rand.

.cm (Defaultwert soll die aufeinanderfolgenden Zeilen komprimieren.) Im .cm Modus arbeitet LOGREA folgendermassen: Jede Eingabezeile wird zeichenweise durchgescannt und jedes gefundene Wort (= Zeichenkette von Blankzeichen umgeben) wird in die Ausgabezeile uebertragen. Dabei wird versucht, ganze Worte auf eine Zeile zu bringen, wobei der Eingabetext komprimiert wird, d.h. ueberfluessige Blanks werden nicht mit in den Ausgabetext uebernommen.

.en Beendet die Textausgabe. Dieses Kommando kann benutzt werden, um die Datei vor dem physikalischen Ende der Datei abzustellen.

.er n (Defaultwert ist n=1.) Fehlermeldungen koennen durch dieses Kommando kontrolliert werden. Falls n=0, gibt es gar keine Spur. Falls n=1, irgendeine Fehlermeldung geht zu der <Formatiertentextdatei>, d.h. Fehlermeldungen erscheinen in der <Formatiertentextdatei> und werden die Anzahl der Ausgabezeilen zerstoeren. Falls n=999, die vollkommene Fehlerausgabe ist moeglich.

Normalerweise deuten die Fehlermeldungen an, dass der Benutzer einen Anwenderfehler gemacht hat.

.ex Setzt alle Metablanks als Metablank-Zeichen, d.h. an den Stellen wo dieses Metablank (hier @-Zeichen) erscheint, soll im Ausgabetext dieselbe @-Zeichen erscheinen (siehe Besondere Merkmale).

.fg n Vorschub n Zeilen um z.B. ein Bild Platz freizumachen. Wenn auf der betreffenden Seite keine n Zeilen uebrig bleiben, soll es auf die naechste Seite verschoben werden. Die .sp Kommando hat keine Wirkung hier. Zum Beispiel, .sp 3 und .fg 10 gibt 10 leere Zeilen.

.ju (Defaultwert soll die Worte auf dem rechten Rand buendig schreiben.) Die Eingabezeilen wird zeichenweise durchgescannt und jedes gefundene Wort (= Zeichenkette von Blankzeichen umgeben) wird in die Ausgabezeile uebertragen. Dann sind die Worte auf dem rechten Rand buendig.

.lm n (Defaultwert ist n=1.) Definiert den linken Rand, d.h. der Wert gibt die Anzahl der leeren Spalten zwischen dem physikalischen linken Rand und dem ersten Wort an.

.ml n (Defaultwert ist n=81.) Definiert die Anzahl der Zeilen in einer Seite des Ausgabetexts. Damit kann das Routing der Druckausgabe zu verschiedenen Druckern geschickt werden. Zum Beispiel, .ml 66 soll fuer den Drucker (PR:) benutzt werden. Der Defaultwert ist fuer den PLOTTER (PLT:) gueltig. Der Drucker setzt Kleinbuchstaben in Grossbuchstaben um und bedarf daher keiner besonderen Beachtung.

.nc Nicht Zeilen komprimieren. Jede Eingabezeile wird zeichenweise durchgescannt und jedes gefundene Zeichen wird in die Ausgabezeile uebertragen. Blanks werden mit in dem Ausgabetext uebernommen.

ne (Ein Metablank wird als echtes Blankzeichen uebersetzt.) Dieses Kommando definiert ein Metablank, d.h. an den Stellen wo dieses Metablank (hier @-Zeichen) erscheint, soll im Ausgabetext ein echtes Blankzeichen erscheinen.

.nj Kein buendiger rechter Rand. Das gibt einen unebenen rechten Rand, aber die Worte werden regelmaessig getrennt.

.nn Keine Seitennummer. Keine Seitennummern werden gedruckt, aber die Anzahl der Seiten wird gerechnet.

.nu n (Defaultwert ist n=1.) Setzt die Seitennummer auf n, wenn n groesser als 0 ist. Falls n=0, denn wird die Seitennummer wieder gedruckt.

.pa n (Defaultwert ist n=1.) Die Seitennummer wird auf Zeile n gedruckt. (Achtung! n muss weniger als die Anzahl der Anfangsseite sein.) Das Wort 'Page' geht der Nummer der Seite voran. Um das Wort 'Seite' zu haben, siehe das .se Kommando.

.pg Vorschub auf die naechste Seite.

.rm n (Defaultwert ist n=80.) Definiert den rechten Rand, d.h. der Wert gibt die Anzahl der leeren Spalten zwischen dem rechten (physikalischen) Rand und dem letzten Wort an.

.se n (Als Defaultwert ist das Wort 'Page' zu benutzen und n=1.) Die Seitennummer wird auf Zeile n gedruckt. Das Wort 'Seite' geht der Nummer der Seite voran.

.sp n (Defaultwert ist n=1.) Definiert die Anzahl der <carriage return> zwischen zwei Ausgabezeilen. Zum Beispiel, .sp 1 gibt

eine Ausgabezeile jede Zeile, und .sp 2 gibt eine leere Zeile zwischen zwei Ausgabezeilen.

.to n (Defaultwert ist n=1.) Definiert die Anzahl der leeren Zeilen am Anfang der Seite.

Neuer Absatz - Dieses Kommando ist das einzige, das nicht explizit ist. Jede Eingabezeile die mit einem Blank anfaengt, bedeutet ein neuer Absatz. Es beendet die laufende Ausgabezeile, erzeugt ein <carriage return> und rueckt die folgende Zeile 5 Zeichen ein.

Besondere Merkmale

Dies sind nuetzliche Parameter, aber keine Kommandos. Zum Beispiel: Unterstreichungszeichen, Metablank und besondere Zeichen. Zur Zeit gibt es:

Metablank - Manchmal ist es notwendig eine exakte Anzahl von leeren Spalten zu haben oder zwei Worte zusammenzubinden, die nicht ueber zwei Zeilen gesplittet werden sollen. Jedoch sind die Blanks in der <Eingabetextdatei> nicht gestattet, und irgendwelche anderen Zeichen muessen benutzt werden. Die @-Zeichen definiert dieses Metablank, d.h. an den Stellen, wo dieses Metablank erscheint, soll in der <Formatiertentextdatei> ein echtes Blankzeichen erscheinen. Ausserdem ist es in der <Eingabetextdatei> wie eine nicht-Blank behandelt. (Siehe Anhang A fuer Beispiele.)

Ein Beispiel

Man gewoehnt sich am besten an LOGREA indem man mit einem Beispiel arbeitet. Anhang A gibt die <Eingabetextdatei> fuer diesen Bericht. Dieser Bericht ist die <Formatiertetextdatei>.

Wie LOOK benutzt wird

Sobald die <Formatiertetextdatei> vorhanden ist, sind zahlreiche Ausgabemoeglichkeiten vorhanden. Die Druckausgabe kann von dem folgenden Kommando erzeugt werden:

DRUCKE <Formatiertetextdatei>,1,PLT:

wo <Formatiertetextdatei> den wirklichen Namen den Datei bedeutet, die den formatierten Text enthaelt. Die Ausgabe von diesem Kommando erfolgt direkt auf dem VERSATEK PLOTTER in Klein- und Grossbuchstaben. Das Kommando:

DRUCKE <Formatiertetextdatei>

setzt Kleinbuchstaben in Grossbuchstaben um und gibt die Ausgabe auf den PRINTER.

Jedoch die <Formatiertetextdatei> kann mit dem Programm LOOK auch direkt auf das Datensichtgeraet ausgegeben werden. LOOK hat zwei Parameter:

LOOK <Formatiertetextdatei>,<neue Datei>

wo <neue Datei> von dem Benutzer schon angelegt wurde (diese sollte eine 'indexed' Datei werden). Diese <neue Datei> kann dazu benutzt werden, nur ein paar Seiten von der ganzen <Formatiertentextdatei> auszuwaehlen. LOOK fragt zuerst:

ZEILEN PRO SEITE I2 (XX)

Die Antwort muss die maximale physikalische Seitenlaenge sein und muss in FORTRAN FORMAT I2 angegeben werden. Der Defaultwert ist 81, und wenn kein .ml Kommando in der <Eingabetextdatei> steht, ist diese Nummer gueltig. Dann gibt es 3 Moeglichkeiten: SEIT, AUSG oder ENDE.

SEITnnn - das Wort SEIT ermoeglicht die Ausgabe der physikalischen Seite nnn. Der Parameter nnn ist eine Nummer im FORTRAN FORMAT I3. (Die Seitennummern von LOGREA werden nicht benutzt.) Jede Seite wird auf dem Datensichtgeraet in Teilen von 22 Zeilen ausgegeben. Ein <carriage return> bringt die naechsten 22 Zeilen, usw. bis Ende der Seite. Dann gibt es wieder die 3 Moeglichkeiten: SEIT, AUSG oder ENDE. Wenn nur die naechste Seite gewuenscht ist, dann bekommt ein <carriage return> das Resultat. Wenn in der 22-Zeilen Pause abzubrechen gewuenscht wird, muss man ein 'E' geben, und es gibt wieder die 3 Moeglichkeiten. Wenn die vorhergehende Seite gewuenscht ist, muss man ein '-' geben.

AUSG - Mit AUSG kann eine ganze Seite sowohl auf dem Datensichtgeraet und als auf der <neue Datei> ausgegeben werden. Der Inhalt der <neuen Datei> kann spaeter ausgedruckt werden.

ENDE - das Wort ENDE verursacht LOOK abzubrechen.

Also, mit LOOK kann man den Inhalt der <Formatiertentextdatei> pruefen, ohne Druckausgabe zu erzeugen.

Teil II. LOGREA Programmierung Einzelheiten

§ 1

LOGREA ist ein grosses Programm, das ungefaehr 2500 FORTRAN-Befehl enthaelt. Der modularische Entwurf in Daten-Strukturen und in Funktionen hilft dabei, die Verstaendlichkeit zu verstuerzen. Das ganze Programm wird hier nicht erklaert, weil jedes Unterprogramm intern dokumentiert ist. Um das Hauptziel dieses Teils des Berichts zu erreichen, wird ein begrifflicher Ueberblick der Programmierung gegeben. Ausserdem werden die zugeordnete CSS-Datei und ihr Lauf beschrieben.

LOGREA besteht aus 51 Unterprogrammen, alle organisiert um einen COMMON Block, der die Parameter der Textformatierung enthaelt. Anhang C beschreibt den COMMON Block, /LOGVAR/. Die logische Organisation des Programms LOGREA ist in Bild 2 gezeigt.

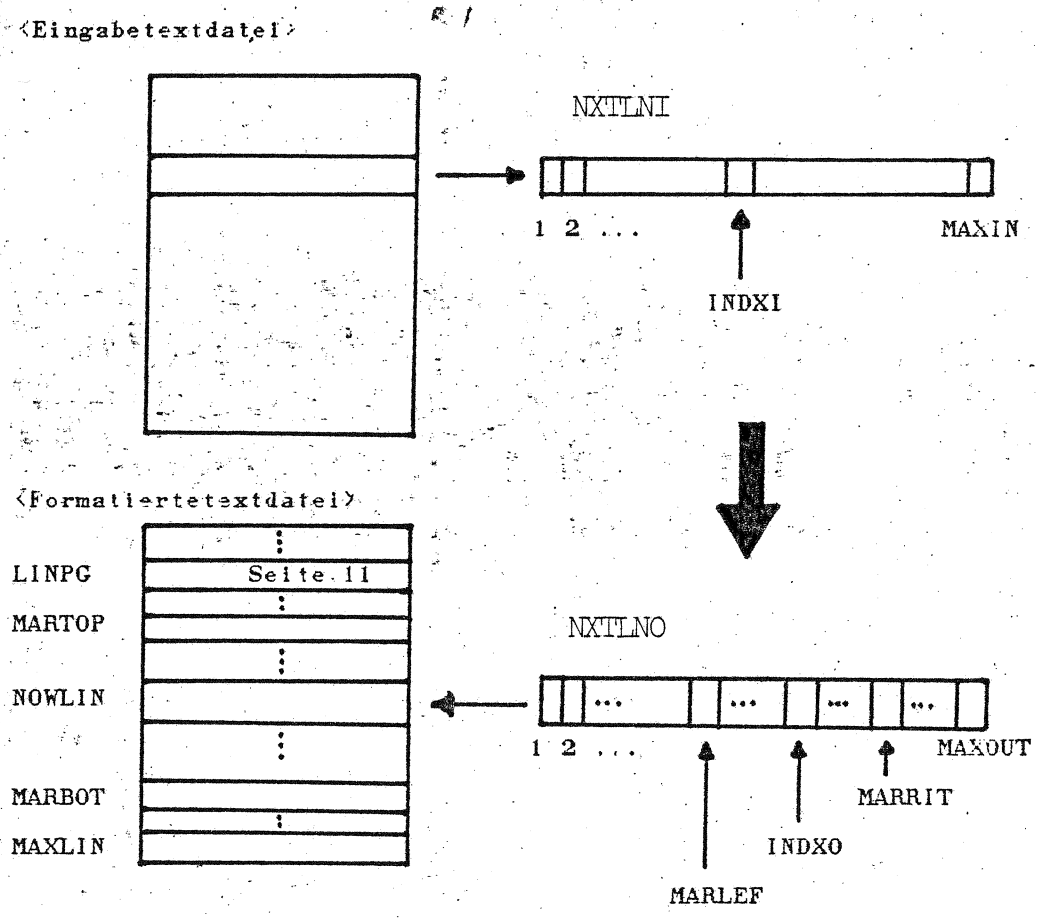


Bild 2 - Text Dateien und Puffer Organisation

Alle Eingabezeilen werden zeilenweise unveraendert in den Eingabepuffer, NXTLNI, uebernommen. NXTLNI ist zeichenweise durchgescannt und jedes gefundene Wort (= Zeichenkette von Blankzeichen umgeben) wird in den Ausgabepuffer, NXTLNO, uebertragen. Dabei wird versucht, ganze Worte auf den Puffer zubringen, wobei NXTLNO

komprimiert wird, d.h. ueberfluessige Blanks werden nicht bleiben. Wenn es keinen Platz mehr zwischen dem linken und dem rechten Rand gibt, werden die Worte auf den rechten Rand buendig geschoben (falls JUST=.TRUE.) und dann in die <Formatiertetextdatei> geschrieben.

Das Hauptprogramm, LOGREA.FTN, geht so weiter:

(1) holt die naechste Zeile von der <Eingabetextdatei>,

(2) entscheidet, ob es sich um ein Kommando oder Textdaten handelt,

(3) falls Textdaten, dann bearbeiten,

(4) falls Kommando, dann das richtige Unterprogramm verwenden.

Es ist einfach, ein neues Kommando zu impementieren durch Einschub eines neuen korrespondierenden Unterprogramms:

- setze den neuen Zeichen-Kode des Kommandos in SETSUB
- setze eine neue Verzweigung im Computed GO TO im Hauptprogramm
- schreibe das neue Unterprogramm.

Der richtige Zeichen Kode ist in Tabelle 1 beschrieben. Wenn ein neues Unterprogramm geschrieben werden soll, kann eine Standard-Prozedur benutzt werden: ein neues Kommando ohne Parameter kann direkt im COMMON-Block angegeben werden, und ein Kommando mit Parametern kann in Anlehnung von BOTMAR (das .bo Kommando) ausgefuehr werden.

Tabelle 1 - Zeichen Kode

Character	INTERDATA 8/32	AMDAHL
a	61	81
b	62	82
c	63	82
d	64	83
e	65	85
f	66	86
g	67	87
h	68	88
i	69	89
j	6A	91
k	6B	92
l	6C	93
m	6D	94
n	6E	95
o	6F	96
p	70	97
q	71	98
r	72	99
s	73	A2
t	74	A3
u	75	A4
v	76	A5
w	77	A6
x	78	A7
y	79	A8
z	7A	A9
S	53	E2
P	50	D7
.	2E	4B

LOGREA.CSS

Die LOGREA.CSS Datei ist:

```
*
* LOGREA  <FILENAME1>,<FILENAME2>
*
*   REFORMAT TEXT ON <FILENAME1> TO <FILENAME2>.
*
*     FILE USAGE      UNIT    FILE NAME
*                             (DEFAULT)
*    ------------------------------------
*
*       INPUT          5        CON:
*       OUTPUT         6        CON:
*       TRACE INPUT    7        CON:
*
LO LOGREA.TSK
$IFNU  1; AS 5,CON:  $ENDC;
$IFNN  1; AS 5, 1; $ENDC;
$IFNN  2; XDE  2; AL  2,IN,126; $ENDC;
$IFNU  2; AS 6,CON:; $ENDC;
$IFNN  2; AS 6, 2; $ENDC;
AS 7,CON:
ST
$EXIT
```

Keine <Formatiertetextdatei> muss bei dem Benutzer zugewiesen werden, weil es automatisch geht. Ausserdem wird eine Datei mit demselben Namen zerstoert. Beim Debugging, darf die <Formatiertetextdatei> nicht angegeben sein, weil dann die Fehlermeldungen auf das Datensichtgeraet (CON:) gehen werden.

Anhang A. Eingabe, von dem dieser Bericht erzeugt wurde

```
.to 5
.bo 70
.lm 10
.se 2
.sp 2
.nn
.bl 10
.ce
LOGREA: Ein Programm zur formatierten Textausgabe
.bl 6
.ce
Thomas C. Henderson
.bl 1
.ce
DFVLR Oberpfaffenhofen
.ce
August 10, 1980
.pg
.ce
Inhaltsverzeichnis
.bl 3
.nc
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@Seite
@
@@@@@Einleitung@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@1
@@@@@Teil I. Wie wird LOGREA benutzt@@@@@@@@@@@@@@@@@@@@@@@@@@@@@2
@@@@@@@@Der Text@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@3
@@@@@@@@Die Kommandos@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@3
@@@@@@@@Besondere Merkmale@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@7
@@@@@@@@Ein Beispiel@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@8
@@@@@@@@Wie LOOK benutzt wird@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@8
@@@@@Teil II. LOGREA Programmierung Einzelheiten@@@@@@@@@@@@@@@@10
@@@@@@@@Zeichen Kode@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@13
@@@@@@@@LOGREA.CSS@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@14
@@@@@Anhang A. Eingabe, von dem dieser Bericht erzeugt wurde@@@@15
@@@@@Anhang B. AMDAHL Einzelheiten@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@27
@@@@@Anhang C. /LOGVAR/ COMMON Block@@@@@@@@@@@@@@@@@@@@@@@@@@@@@28
@@@@@Anhang D. Kommandoszusammenfassung@@@@@@@@@@@@@@@@@@@@@@@@@@31
.nu 1
.bl 1
.cm
.pg
.ce
Einleitung
.bl 1
 LOGREA ist ein Programm zur formatierten Textausgabe, das speziell
fuer die DIBIAS Gruppe entwickelt wurde. Ein solches Programm hilft
bei der Entwicklung und bei der Erzeugung von Berichten und
Dokumentation, indem es eine schnelle Textmanipulation
und Formataenderung erlaubt. Zum Beispiel: Raender,
Einruecken, leere Zeilen usw. koennen einfach veraendert werden.
Ausserdem, wenn der Eingabetext veraendert ist, formatiert wieder
der Lauf von LOGREA automatisch den ganzen Bericht.
 Dieser Bericht besteht aus 2 Teilen: erster Teil, eine Eroeterung
ueber LOGREA, was den Benutzer betrifft, und zweiter Teil, eine
Eroeterung ueber die Programmierungseinzelheiten. Die Eroeterung ist
gueltig fuer die zwei Rechner INTERDATA 8/32 und AMDAHL. Eine
LOGREA-Benutzerbeschreibung ist immer an der INTERDATA 8/32 auf
der LOGREA.MAN/G Datei vorhanden. Die AMDAHL Einzelheiten sind in Anhang
B erklaert.
```

```
.pg
.ce
Teil I. Wie wird LOGREA benutzt
.bl 2
 LOGREA wurde fuer die DIBIAS-Gruppe entworfen. Es laeuft im
RTOS/MTM auf dem INTERDATA 8/32 Rechner.
Die MTM-Konfiguration von LOGREA ist in Bild 1 schematisch
dargestellt.
.bl 2
.nc
@@@@@@@@@@@@@@@@@@<Eingabetextdatei>@@@@@EDIT
@
@
@@@@@@@@@@@@@@@@@LOGREA
@
@
@@@@@@@@@@@@@@@@<Formatiertentextdatei>
@
@
@@@@@@@@@@@@@@@LOOK@@@@@@@@@@@@@@@@DRUCKE
@
@
@@@@Datensichtgeraet        DRUCKER
@@@@@@@@@@<neue Datei>       PLOTTER
.bl 1
.ce
Bild 1 - Die LOGREA Konfiguration
.bl 2
.cm
Die <Eingabetextdatei> wird mit EDIT geschaffen oder veraendert
(EDIT ist eine CSS-Datei, die den Editor in Gang setzt). Die
<Eingabetextdatei> wird zeilenweise abgearbeitet und erhaelt zwei
verschiedene Arten von Daten:
.br
@@@@@-@LOGREA Kommandos und
.br
@@@@@-@Textdaten.
.br
Das bedeutet, dass jede Zeile entweder ein Kommando oder irgendwelche
Textdaten sind, die in der <Formatiertentextdatei> erscheinen sollen.
Die LOGREA-Kommandos werden in der <Formatiertentextdatei>
nicht erscheinen. Die
Kommandos koennen irgendwo in der <Eingabetextdatei>
eingestreut werden und kontrollieren die Erscheinung der
<Formatiertentextdatei>.
.bl 2
.ce
Der Text
.bl 1
 Der zu verarbeitende Text kann auch Gross- und Kleinbuchstaben
enthalten. Jedoch gibt es zwei Arten von Ausnahmen. Zuerst
gibt es ein Vereinbarung, so dass die Kommandos von den Textdaten
unterschieden werden
koennen. Die Kommandos beginnen alle mit einem .-Zeichen,
und diese Steuerzeichen muss stets am Anfang der
Zeile stehen. Zweitens, jede Zeile, die mit einem Blank beginnt,
bedeutet ein neuer Absatz. Sonst koennen Textdaten irgendeine
Zeichenkette sein. Wenn die <Eingabetextdatei> ohne
Kommandos ist, gibt es Defaultwerte fuer die Raender, usw. Diese
Defaultwerte sind in Anhang C gegeben.
.bl 2
.ce
Die Kommandos
.bl 1
 Die Kommandos erlauben die Wahl der Raender, Einruecken, usw.
Die Textdaten bestehen aus der zu verarbeitenden
```

<Eingabetextdatei> und darin eingestreuten
Kommandos fuer LOGREA, aber die Kommandos werden in der
<Formatiertentextdatei> nicht erscheinen.
Nur ein Kommando darf pro Eingabezeile stehen, und es muss so
aussehen:
.br
.bl 1
Spalte@@:@@@1@@@@2@3@@@@@4@5...
.br
@@@@@@@@@@@@@.@<Kommando>@@<Parameter>
.br
.bl 1
Die Kommandos beginnen alle mit einem Steuerzeichen (hier
die .-Zeichen),
und sie muessen stets im Spalte 1 der Zeile stehen.
Alle Kommandos bestehen aus 2 Buchstaben und sind von dem
<Parameter> 1 Blank entfernt. Ein <Parameter> kann nur eine ganze Zahl
oder Blanks sein. Das <Kommando> darf nur aus Kleinbuchstaben bestehen,
d.h. a,b,c.....z. Die gueltigen Kommandos sind:
.bl 1
@@@@@.bl@n@@@@@@n Zeilen Vorschub. (Achtung! .sp 2 und dann .bl 2
gibt 4 leere Zeilen.)
.bl 1
@@@@@.bo@n@@@@@(Defaultwert ist n=64.) Definiert die letzte Zeile
am Ende der Seite.
.bl 1
@@@@@.br@@@@@@@@Vorschub auf die naechste Zeile.
.bl 1
@@@@@.ce@@@@@@@@Die folgende Zeile ist zwischen dem linken und
rechten Rand.
.bl 1
@@@@@.cm@@@@@@@@(Defaultwert soll die aufeinanderfolgenden Zeilen
komprimieren.) Im .cm Modus arbeitet LOGREA folgendermassen:
jede Eingabezeile wird zeichenweise durchgescannt und jedes
gefundene Wort (@=@Zeichenkette von Blankzeichen umgeben) wird in
die Ausgabezeile uebertragen. Dabei wird versucht, ganze Worte auf
eine Zeile zu bringen, wobei der Eingabetext komprimiert wird,
d.h. ueberfluessige Blanks werden nicht mit in den Ausgabetext
uebernommen.
.bl 1
@@@@@.en@@@@@@@@Beendet die Textausgabe. Dieses Kommando kann benutzt
werden, um die Datei vor dem physikalischen Ende der Datei
abzustellen.
.bl 1
@@@@@.er@n@@@@@(Defaultwert ist n=1.) Fehlermeldungen koennen durch
dieses Kommando kontrolliert werden. Falls n=0, gibt es gar keine
Spur. Falls n=1, irgendeine Fehlermeldung geht zu der
<Formatiertentextdatei>, d.h. Fehlermeldungen erscheinen in der
<Formatiertentextdatei> und werden die Anzahl der Ausgabezeilen
zerstoeren. Falls n=999, die vollkommene Fehlerausgabe ist
moeglich. Normalerweise deuten die Fehlermeldungen an, dass der
Benutzer einen Anwenderfehler gemacht hat.
.bl 1
@@@@@.ex@@@@@@@@Setzt alle Metablanks als Metablank-Zeichen, d.h. an
den Stellen wo dieses Metablank (hier @-Zeichen) erscheint, soll
.ex
im Ausgabetext dieselbe @-Zeichen erscheinen (siehe Besondere Merkmale).
.bl 1
.ne
@@@@@.fg@n@@@@@Vorschub n Zeilen um z.B. ein Bild Platz freizumachen.
Wenn auf der betreffenden Seite keine n Zeilen uebrig bleiben, soll es
auf die
naechste Seite verschoben werden. Die .sp Kommando hat keine Wirkung hier.
Zum Beispiel, .sp 3 und .fg 10 gibt 10 leere Zeilen.
.bl 1
@@@@@.ju@@@@@@@@(Defaultwert soll die Worte auf dem rechten Rand

buendig schreiben.) Die Eingabezeilen wird zeichenweise durchgescannt und jedes gefundene Wort (= Zeichenkette von Blankzeichen umgeben) wird in die Ausgabezeile uebertragen. Dann sind die Worte auf dem rechten Rand buendig.

.bl 1

@@@@@.lm@n@@@@@(Defaultwert ist n=1.) Definiert den linken Rand, d.h. der Wert gibt die Anzahl der leeren Spalten zwischen dem physikalischen linken Rand und dem ersten Wort an.

.bl 1

@@@@@.ml@n@@@@@(Defaultwert ist n=81.) Definiert die Anzahl der Zeilen in einer Seite des Ausgabetexts. Damit kann das Routing der Druckausgabe zu verschiedenen Druckern geschickt werden. Zum Beispiel, .ml 66 soll fuer den Drucker (PR:) benutzt werden. Der Defaultwert ist fuer den PLOTTER (PLT:) gueltig. Der Drucker setzt Kleinbuchstaben in Grossbuchstaben um und bedarf daher keiner besonderen Beachtung.

.bl 1

@@@@@.nc@@@@@@@@Nicht Zeilen komprimieren. Jede Eingabezeile wird zeichenweise durchgescannt und jedes gefundene Zeichen wird in die Ausgabezeile uebertragen. Blanks werden mit in dem Ausgabetext uebernommen.

.bl 1

@@@@@.ne@@@@@@@@(Ein Metablank wird als echtes Blankzeichen uebersetzt.) Dieses Kommando definiert ein Metablank, d.h. an
.ex
den Stellen wo dieses Metablank (hier @-Zeichen) erscheint, soll im Ausgabetext ein echtes Blankzeichen erscheinen.

.bl 1

.ne

@@@@@.nj@@@@@@@@Kein buendiger rechter Rand. Das gibt einen unebenen rechten Rand, aber die Worte werden regelmaessig getrennt.

.bl 1

@@@@@.nn@@@@@@@@Keine Seitennummer. Keine Seitennummern werden gedruckt, aber die Anzahl der Seiten wird gerechnet.

.bl 1

@@@@@.nu@n@@@@@@(Defaultwert ist n=1.) Setzt die Seitennummer auf n, wenn n groesser als 0 ist. Falls n=0, denn wird die Seitennummer wieder gedruckt.

.bl 1

@@@@@.pa@n@@@@@@(Defaultwert ist n=1.) Die Seitennummer wird auf Zeile n gedruckt. (Achtung! n muss weniger als die Anzahl der Anfangsseite sein.) Das Wort 'Page' geht der Nummer der Seite voran. Um das Wort 'Seite' zu haben, siehe das .se Kommando.

.bl 1

@@@@@.pg@@@@@@@@Vorschub auf die naechste Seite.

.bl 1

@@@@@.rm@n@@@@@@(Defaultwert ist n=80.) Definiert den rechten Rand, d.h. der Wert gibt die Anzahl der leeren Spalten zwischen dem rechten (physikalischen) Rand und dem letzten Wort an.

.bl 1

@@@@@.se@n@@@@@@(Als Defaultwert ist das Wort 'Page' zu benutzen und n=1.) Die Seitennummer wird auf Zeile n gedruckt. Das Wort 'Seite' geht der Nummer der Seite voran.

.bl 1

@@@@@.sp@n@@@@@@(Defaultwert ist n=1.) Definiert die Anzahl der <carriage return> zwischen zwei Ausgabezeilen. Zum Beispiel, .sp 1 gibt eine Ausgabezeile jede Zeile, und .sp 2 gibt eine leere Zeile zwischen zwei Ausgabezeilen.

.bl 1

@@@@@.to@n@@@@@@(Defaultwert ist n=1.) Definiert die Anzahl der leeren Zeilen am Anfang der Seite.

.bl 1

Neuer Absatz@-@Dieses Kommando ist das einzige, das nicht explizit ist. Jede Eingabezeile die mit einem Blank anfaengt, bedeutet ein neuer Absatz. Es beendet die laufende Ausgabezeile, erzeugt ein

```
<carriage return> und rueckt die folgende Zeile 5 Zeichen ein.
.bl 2
.ce
Besondere Merkmale
.bl 1
 Dies sind nuetzliche Parameter, aber keine Kommandos.
.ex
Zum Beispiel: Unterstreichungszeichen, Metablank und besondere
Zeichen. Zur Zeit gibt es:
.ne
.bl 1
 Metablank@-@Manchmal ist es notwendig eine exakte Anzahl von
leeren Spalten zu haben oder zwei Worte zusammenzubinden, die
.ex
nicht ueber zwei Zeilen gesplittet werden sollen. Jedoch sind die
Blanks in den <Eingabetextdatei> nicht gestattet,
und irgendwelche anderen Zeichen muessen benutzt werden. Die @-Zeichen
definiert dieses Metablank, d.h. an den Stellen, wo dieses
Metablank erscheint, soll in der <Formatiertentextdatei>
.ne
ein echtes Blankzeichen erscheinen. Ausserdem ist es in der
<Eingabetextdatei>
wie eine nicht-Blank behandelt. (Siehe Anhang A fuer Beispiele.)
.pg
.ce
Ein Beispiel
.bl 1
 Man gewoehnt sich am besten an LOGREA indem man mit einem Beispiel
arbeitet. Anhang A gibt die <Eingabetextdatei>
fuer diesen Bericht. Dieser Bericht ist die
<Formatiertetextdatei>.
.bl 2
.ce
Wie LOOK benutzt wird
.bl 1
 Sobald die <Formatiertetextdatei> vorhanden ist, sind
zahlreiche Ausgabemoeglichkeiten vorhanden. Die Druckausgabe kann von dem
folgenden Kommando erzeugt werden:
.br
@@@@@DRUCKE@<Formatiertetextdatei>,1,PLT:
.br
wo <Formatiertetextdatei> den wirklichen Namen den Datei bedeutet,
die den
formatierten Text enthaelt. Die Ausgabe von diesem Kommando
erfolgt direkt auf dem VERSATEK PLOTTER in Klein- und
Grossbuchstaben. Das Kommando:
.br
@@@@@DRUCKE@@<Formatiertetextdatei>
.br
setzt Kleinbuchstaben in Grossbuchstaben um und gibt die Ausgabe
auf den PRINTER.
 Jedoch die <Formatiertetextdatei> kann mit dem Programm
LOOK auch direkt auf das Datensichtgeraet ausgegeben werden. LOOK hat
zwei Parameter:
.br
@@@@@@@@LOOK@<Formatiertetextdatei>,<neue Datei>
.br
wo <neue Datei> von dem Benutzer schon angelegt wurde (diese sollte eine
'indexed' Datei werden). Diese <neue Datei> kann dazu benutzt werden, nur
ein paar Seiten von der ganzen <Formatiertentextdatei>
auszuwaehlen. LOOK fragt zuerst:
.br
@@@@@ZEILEN PRO SEITE  I2 (XX)
.br
Die Antwort muss die maximale physikalische Seitenlaenge
sein und muss in FORTRAN FORMAT I2 angegeben werden. Der Defaultwert
```

ist 81, und wenn kein .ml Kommando in der <Eingabetextdatei>
steht, ist diese Nummer gueltig. Dann gibt es 3 Moeglichkeiten:
SEIT, AUSG oder ENDE.
.bl 1
SEITnnn@-@das Wort SEIT ermoeglicht die Ausgabe der physikalischen
Seite nnn.
Der Parameter nnn ist eine Nummer im FORTRAN FORMAT I3. (Die
Seitennummern von LOGREA werden nicht benutzt.) Jede Seite wird auf dem
Datensichtgeraet in Teilen von 22 Zeilen ausgegeben. Ein <carriage return>
bringt die naechsten 22 Zeilen, usw. bis Ende der Seite. Dann gibt
es wieder die 3 Moeglichkeiten: SEIT, AUSG oder ENDE.
Wenn nur die naechste Seite gewuenscht ist,
dann bekommt ein <carriage return> das Resultat. Wenn in der 22-Zeilen
Pause abzubrechen gewuenscht wird, muss man ein 'E' geben, und es
gibt wieder die 3 Moeglichkeiten. Wenn die vorhergehende Seite gewuenscht
ist,
muss man ein '-' geben.
AUSG@-@Mit AUSG kann eine ganze Seite sowohl auf dem Datensichtgeraet
und als auf der <neue Datei> ausgegeben werden. Der Inhalt der
<neuen Datei>
kann spaeter ausgedruckt werden.
ENDE@-@das Wort ENDE verursacht LOOK abzubrechen.
.bl 1
Also, mit LOOK kann man den Inhalt der <Formatiertentextdatei>
pruefen, ohne Druckausgabe zu erzeugen.
.pg
.ce
Teil II. LOGREA Programmierung Einzelheiten
.bl 2
LOGREA ist ein grosses Programm, das ungefaehr 2500 FORTRAN-Befehl
enthaelt. Der modularische Entwurf in Daten-Strukturen und in
Funktionen hilft dabei, die Verstaendlichkeit zu verstuerzen. Das
ganze Programm
wird hier nicht erklaert, weil jedes Unterprogramm intern
dokumentiert ist. Um das Hauptziel dieses Teils des Berichts
zu erreichen, wird ein begrifflicher Ueberblick der Programmierung
gegeben. Ausserdem werden die zugeordnete CSS-Datei und ihr Lauf
beschrieben.
LOGREA besteht aus 51 Unterprogrammen, alle organisiert um einen
COMMON Block, der die Parameter der Textformatierung enthaelt. Anhang C
beschreibt den COMMON Block, /LOGVAR/. Die logische Organisation des
Programms LOGREA ist in Bild 2 gezeigt.
.sp 1
.pg
.nc
.bl 5
<Eingabetextdatei>
.bl 7
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@1@2@...@@@@@@@@@@@@@@@@@@@MAXIN
.bl 2
@@INDXI
.bl 7
<Formatiertetextdatei>
.bl 2
LINPG Seite 11
@
MARTOP
@
@
NOWLIN
@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@1@2@...@@@@@@@@@@@@@@@@@@@MAXOUT
@
MARBOT@@MARRIT
@
MAXLIN@@INDXO

```
@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@MARLEF
@
.bl 10
.cm
.sp 2
.ce
Bild 2 - Text Dateien und Puffer Organisation
.bl 1
 Alle Eingabezeilen werden zeilenweise unveraendert in den
Eingabepuffer, NXTLNI, uebernommen. NXTLNI ist zeichenweise
durchgescannt und jedes gefundene Wort (@=@Zeichenkette von
Blankzeichen umgeben) wird in den Ausgabepuffer, NXTLNO,
uebertragen. Dabei wird versucht, ganze Worte auf den Puffer
zubringen, wobei NXTLNO komprimiert wird, d.h. ueberfluessige
Blanks werden nicht bleiben. Wenn es keinen Platz mehr zwischen
dem linken und dem rechten Rand gibt, werden die Worte auf den
rechten Rand buendig geschoben (falls JUST=.TRUE.) und dann in die
<Formatiertetextdatei> geschrieben.
 Das Hauptprogramm, LOGREA.FTN, geht so weiter:
.br
.bl 1
@@@@@@@@(1)@holt die naechste Zeile von der <Eingabetextdatei>,
.br
@@@@@@@@(2)@entscheidet, ob es sich um ein Kommando oder Textdaten
handelt,
.br
@@@@@@@@(3)@falls Textdaten, dann bearbeiten,
.br
@@@@@@@@(4)@falls Kommando, dann das richtige Unterprogramm
verwenden.
.br
.bl 1
Es ist einfach, ein neues Kommando zu impementieren durch Einschub
eines neuen korrespondierenden Unterprogramms:
.br
.bl 1
@@@@@-@setze den neuen Zeichen-Kode des Kommandos in SETSUB
.br
@@@@@-@setze eine neue Verzweigung im Computed GO@TO im
Hauptprogramm
.br
@@@@@-@schreibe das neue Unterprogramm.
.br
.bl 1
Der richtige Zeichen Kode ist in Tabelle 1 beschrieben.
Wenn ein neues Unterprogramm geschrieben werden soll, kann eine
Standard-Prozedur benutzt werden: ein neues Kommando ohne Parameter
kann direkt im COMMON-Block angegeben werden, und ein Kommando mit
Parametern kann in Anlehnung von BOTMAR (das .bo Kommando)
ausgefuehr werden.
.pg
.sp 1
.ce
Tabelle 1 - Zeichen Kode
.bl 2
.nc
@@@@@@Character      INTERDATA 8/32      AMDAHL
      ---------      --------------      ------
@
          a               61               81
          b               62               82
          c               63               82
          d               64               83
          e               65               85
          f               66               86
```

```
g                   67                  87
h                   68                  88
i                   69                  89
j                   6A                  91
k                   6B                  92
l                   6C                  93
m                   6D                  94
n                   6E                  95
o                   6F                  96
p                   70                  97
q                   71                  98
r                   72                  99
s                   73                  A2
t                   74                  A3
u                   75                  A4
v                   76                  A5
w                   77                  A6
x                   78                  A7
y                   79                  A8
z                   7A                  A9
@
S                   53                  E2
P                   50                  D7
@
                    2E                  4B
.cm
.sp 2
.pg
.ce
LOGREA.CSS
.bl 2
 Die LOGREA.CSS Datei ist:
.br
.sp 1
.nc
*
* LOGREA  <FILENAME1>,<FILENAME2>
*
*   REFORMAT TEXT ON <FILENAME1> TO <FILENAME2>.
*
*     FILE USAGE      UNIT      FILE NAME
*                               (DEFAULT)
*    ------------------------------------
**
*       INPUT          5          CON:
*       OUTPUT         6          CON:
*       TRACE INPUT    7          CON:
*
LO LOGREA.TSK
$IFNU @1; AS 5,CON:; $ENDC;
$IFNN @1; AS 5,@1; $ENDC;
$IFNN @2; XDE @2; AL @2,IN,126; $ENDC;
$IFNU @2; AS 6,CON:; $ENDC;
$IFNN @2; AS 6,@2; $ENDC;
AS 7,CON:
ST
$EXIT
.cm
.sp 2
.bl 1
Keine <Formatiertetextdatei> muss bei dem
Benutzer zugewiesen werden, weil es automatisch geht. Ausserdem wird
eine Datei mit demselben Namen zerstoert. Beim Debugging, darf die
<Formatiertetextdatei> nicht angegeben sein, weil dann die
Fehlermeldungen auf das Datensichtgeraet (CON:) gehen werden.
.pg
```

```
.ce
Anhang A. Eingabe, von dem dieser Bericht erzeugt wurde
.bl 2
.sp 1
.nc
.ex
.to 5
.bo 70
.lm 10
.se 2
.sp 2
.nn
.bl 10
.ce
LOGREA: Ein Programm zur formatierten Textausgabe
.bl 6
.ce
Thomas C. Henderson
.bl 1
.ce
DFVLR Oberpfaffenhofen
.ce
August 10, 1980
.pg
.ce
Inhaltsverzeichnis
.bl 3
.nc
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@Seite
@
@@@@@Einleitung@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@1
@@@@@Teil I. Wie wird LOGREA benutzt@@@@@@@@@@@@@@@@@@@@@@@@@@@@@2
@@@@@@@@Der Text@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@3
@@@@@@@@Die Kommandos@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@3
@@@@@@@@Besondere Merkmale@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@7
@@@@@@@@Ein Beispiel@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@8
@@@@@@@@Wie LOOK benutzt wird@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@8
@@@@@Teil II. LOGREA Programmierung Einzelheiten@@@@@@@@@@@@@@10
@@@@@@@@Zeichen Kode@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@13
@@@@@@@@LOGREA.CSS@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@14
@@@@@Anhang A. Eingabe, von dem dieser Bericht erzeugt wurde@@@@15
@@@@@Anhang B. AMDAHL Einzelheiten@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@25
@@@@@Anhang C. /LOGVAR/ COMMON Block@@@@@@@@@@@@@@@@@@@@@@@@@@@@26
@@@@@Anhang D. Kommandoszusammenfassung@@@@@@@@@@@@@@@@@@@@@@@@@29
.nu 1
.bl 1
.cm
.pg
.ce
Einleitung
.bl 1
 LOGREA ist ein Programm zur formatierten Textausgabe, das speziell
fuer die DIBIAS Gruppe entwickelt wurde. Ein solches Programm hilft
bei der Entwicklung und bei der Erzeugung von Berichten und
Dokumentation, indem es eine schnelle Textmanipulation
und Formataenderung erlaubt. Zum Beispiel: Raender,
Einruecken, leere Zeilen usw. koennen einfach veraendert werden.
Ausserdem, wenn der Eingabetext veraendert ist, formatiert wieder
der Lauf von LOGREA automatisch den ganzen Bericht.
 Dieser Bericht besteht aus 2 Teilen: erster Teil, eine Eroeterung
ueber LOGREA, was den Benutzer betrifft, und zweiter Teil, eine
Eroeterung ueber die Programmierungseinzelheiten. Die Eroeterung ist
gueltig fuer die zwei Rechner INTERDATA 8/32 und AMDAHL. Eine
LOGREA-Benutzerbeschreibung ist immer an der INTERDATA 8/32 auf
der LOGREA.MAN/G Datei vorhanden. Die AMDAHL Einzelheiten sind in Anhang
B erklaert.
```

```
.pg
.ce
Teil I. Wie wird LOGREA benutzt
.bl 2
 LOGREA wurde fuer die DIBIAS-Gruppe entworfen. Es laeuft im
RTOS/MTM auf dem INTERDATA 8/32 Rechner.
Die MTM-Konfiguration von LOGREA ist in Bild 1 schematisch
dargestellt.
.bl 2
.nc
@@@@@@@@@@@@@@@@@<Eingabetextdatei>@@@@@EDIT
@
@
@@@@@@@@@@@@@@@@@LOGREA
@
@
@@@@@@@@@@@@@@@@<Formatiertentextdatei>
@
@
@@@@@@@@@@@@@@LOOK@@@@@@@@@@@@@@DRUCKE
@
@
@@@@Datensichtgeraet       DRUCKER
@@@@@@@@@@<neue Datei>     PLOTTER
.bl 1
.ce
Bild 1 - Die LOGREA Konfiguration
.bl 2
.cm
Die <Eingabetextdatei> wird mit EDIT geschaffen oder veraendert
(EDIT ist eine CSS-Datei, die den Editor in Gang setzt). Die
<Eingabetextdatei> wird zeilenweise abgearbeitet und erhaelt zwei
verschiedene Arten von Daten:
.br
@@@@@-@LOGREA Kommandos und
.br
@@@@@-@Textdaten.
.br
Das bedeutet, dass jede Zeile entweder ein Kommando oder irgendwelche
.sp 2
.cm
.ne
.pg
.ce
Anhang B. AMDAHL Einzelheiten
.bl 2
 LOGREA laeuft im IBM/MVS auf den AMDAHL-Rechner. Es ist moeglich
die Druckausgabe auf den Schnelldrucker im Rechnenzentrum zu
schicken. Ausserdem kann die Eingabe von LOGREA eine Textdatei sein,
die mit einem beliebigen Editor erzeugt wurde. Es ist auch moeglich,
Textdateien die mit dem INTERDATA 8/32 erzeugt wurden, auf dem
AMDAHL-Rechner zu verarbeiten. Mann muss TAPECARD@- eine CSS-Datei@-@in
MTM nutzen.
 Um LOGREA auf AMDAHL zum Laufen zu bringen:
.br
.bl 1
@@@@@EXEC 'NT15.CLIST(LOGREA)' 'eingabe,ausgabe'
.br
.bl 1
Wenn man einen AMDAHL Editor benutzt, muss er im Zeichen Modus sein,
sonst gibt es Zeilenummern zwischen Spalten 73-80 in der
Eingabedatei. In diesem Fall kann des .mi Kommando benutzt werden,
um die Spalten 73-80 zu ignorieren (d.h. es muss ein .mi 72 in
der Eingabedatei stehen).
 AMDAHL FORTRAN gibt keine logischen Funktionen AND, OR oder SHIFT.
Deshalb sind diese Funktionen auf ASSEMBLY Sprache geschrieben.
```

```
.pg
.ce
Anhang C. /LOGVAR/ COMMON Block
.bl 2
Die Defaultwerte sind in dem COMMON Block gegeben als:
.br
.bl 1
.nc
      LOGICAL JUST,NEWPAR,NEWPG,NEWLIN,NUMBR,CMPACT,KHRCON,LINMTY,
     $    GERMAN
      INTEGER*4 NXTLNI(200),NXTLNO(400),LINPG,IDUMMY(73),KHRNUM,
     $    KHRIN(10),KHROUT(10)
@
      DATA NXTSUB,INDXI,INDXO,IER,MAXIN,MAXOUT,MAXSP,MINLIN,MAXLIN,
     $    MARRIT,MARLEF,INPAR,NUMPG,MARTOP,MARBOT,MAXPG,MINPG,
     $    INDPAR,NEWPAR,NEWPG,NEWLIN,NOWLIN,JUST,NUMSP,LINPG,NUMBR,
     $    CMPACT,KHRCON,LINMTY,GERMAN,NUMKHR,KHRIN,KHROUT
     $   / 8,1,1,1,80,80,80,1,81,80,1,0,1,2,81,800,1,6,
     $     .TRUE.,.TRUE.,.TRUE.,1,.TRUE.,1,1,.TRUE.,.TRUE.,
.ex  $     .TRUE.,.TRUE.,.FALSE.,1,'@   ',9*0,'      ',9*0 /
.ne
.bl 1
.cm
Die Variablen haben die folgenden Bedeutungen:
.br
.bl 1
.no
   NXTLNI..(INTEGER*4 ARRAY)   Eingabe Puffer.
   NXTLNO..(INTEGER*4 ARRAY)   Ausgabe Puffer.
   NXTSUB..(INTEGER*4)         Integer Code, um das naechste
                               Unterprogramm zu verwenden.
   INDXI...(INTEGER*4)         Die naechste Position in NXTLNI
                               zu verarbeiten.
   INDXO...(INTEGER*4)         Die naechste Position in NXTLNO
                               zu verarbeiten.
   IER.....(INTEGER*4)         Fehler Puffer.
   MAXIN...(INTEGER*4)         Maximale Zeichenanzahl des Eingabe-
                               Datei-Puffers.
   MAXOUT..(INTEGER*4)         Maximale Spaltenauzahl des Ausgabe-
                               Datei-Puffers.
   MAXSP...(INTEGER*4)         Maximale Anzahl der leeren Zeilen
                               zwischen zwei nicht-leeren Zeilen.
   MINLIN..(INTEGER*4)         Minimale Anzahl der physikalischen
                               Zeilen.
   MAXLIN..(INTEGER*4)         Maximale physikalische Anzahl
                               von Zeilen einer Seite.
   MARRIT..(INTEGER*4)         Letzte moegliche nicht-Blank Spalte.
   MARLEF..(INTEGER*4)         Erste moegliche nicht-Blank Spalte.
   INPAR...(INTEGER*4)         Anzahl der leeren Zeilen zwischen
                               den Absaetzen.
   NUMPG...(INTEGER*4)         Gueltige Seitennummer.
   MARTOP..(INTEGER*4)         Erste moegliche nicht-leere Zeile.
   MARBOT..(INTEGER*4)         Letzte moegliche nicht-leere Zeile.
   MAXPG...(INTEGER*4)         Maximale Anzahl der Seiten.
   MINPGn.n.n.(INTEGER*4)      Minimale Anzahl der Seiten.
   INDPAR..(INTEGER*4)         Anzahl von Blanks fuer einen Absatz.
   NEWPAR..(LOGICAL)           Neuer Absatz.
   NEWPG...(LOGICAL)           Neue Seite.
   NEWLIN..(LOGICAL)           Neue Zeile.
   NOWLIN..(INTEGER*4)         Laufende Zeilenummer der
                               <Formatiertentextdatei>.
   JUST....(LOGICAL)           Rechts buendig schieben.
   NUMSP...(INTEGER*4)         Anzahl von <carriage return> bis
                               zur naechsten Zeile.
   LINPG...(INTEGER*4)         Zeilenummer in der die Seitennummer
```

```
                                steht.
   NUMBR...(LOGICAL)            Drucke Seitennummern.
   CMPACT..(LOGICAL)            Komprimiere Zeilen.
   KHRCON..(INTEGER*4)          Uebersetzen explizite Zeichen.
   LINMTY..(INTEGER*4)          Leere Zeile.
   GERMAN..(LOGICAL)            / Deutsche Wort fuer Seitennummern.
   IDUMMY..(INTEGER*4 ARRAY)    Nicht benutzt.
   NUMKHR..(INTEGER*4)          Anzahl der expliziten Zeichen.
   KHRIN...(INTEGER*4 ARRAY)    Liste von expliziten Zeichen.
   KHROUT..(INTEGER*4 ARRAY)    Liste von Zeichen die zu den
                                expliziten Zeichen gehoeren.

.cm
.bl 1
Falls eine neue Variable gebraucht wird, gibt es Platz in IDUMMY.
.pg
.ce
Anhang D. Kommandos Zusammenfassung
.bl 2
.sp 2
.nc
   .bl n   (Vorschub n Zeilen)
   .bo n   (n leere Zeilen am Ende der Seite)
   .br     (Vorschub eine Zeile)
   .ce     (zentrieren folgende Zeile)
.ex
   .cm     (Zeilen komprimieren)
   .en     (Ende Bearbeitung)
   .er n   (Fehlermeldungen)
   .ex     (die @-Zeichen werden gedruckt)
   .fg n   (n leere Zeilen fuer ein Bild)
   .ju     (Worte rechts buendig)
   .lm n   (definiert linken Rand)
   .ml n   (definiert physikalisches Maximum von Zeilen)
   .nc     (nicht Zeilen komprimieren)
   .ne     (die Blankzeichen werden fuer @-Zeichen gedruckt)
   .nj     (nicht rechts buendig)
.ne
   .nn     (nicht Seitennummern drucken)
   .nu n   (setzt Seitennummer = n)
   .pa n   (Seitennummern auf Englisch)
   .pg     (Vorschub eine Seite)
   .rm n   (definiert rechten Rand)
   .se n   (Seitennummern auf Deutsch)
   .sp n   (setzt Vorschub zwischen Zeilen)
   .to n   (n leere Zeilen am Anfang der Seite)
.cm
```

Anhang B. AMDAHL Einzelheiten

LOGREA laeuft im IBM/MVS auf den AMDAHL-Rechner. Es ist moeglich die Druckausgabe auf den Schnelldrucker im Rechnenzentrum zu schicken. Ausserdem kann die Eingabe von LOGREA eine Textdatei sein, die mit einem beliebigen Editor erzeugt wurde. Es ist auch moeglich, Textdateien die mit dem INTERDATA 8/32 erzeugt wurden, auf dem AMDAHL-Rechner zu verarbeiten. Mann muss TAPECARD - eine CSS-Datei - in MTM nutzen.

Um LOGREA auf AMDAHL zum Laufen zu bringen:

```
EXEC 'NT15.CLIST(LOGREA)' 'eingabe,ausgabe'
```

Wenn man einen AMDAHL Editor benutzt, muss er im Zeichen Modus sein, sonst gibt es Zeilennummern zwischen Spalten 73-80 in der Eingabedatei. In diesem Fall kann des .mi Kommando benutzt werden, um die Spalten 73-80 zu ignorieren (d.h. es muss ein .mi 72 in der Eingabedatei stehen).

AMDAHL FORTRAN gibt keine logischen Funktionen AND, OR oder SHIFT. Deshalb sind diese Funktionen auf ASSEMBLY Sprache geschrieben.

Anhang C. /LOGVAR/ COMMON Block

Die Defaultwerte sind in dem COMMON Block gegeben als:

```
      LOGICAL JUST,NEWPAR,NEWPG,NEWLIN,NUMBR,CMPACT,KHRCON,LINMTY,
     $   GERMAN
      INTEGER*4 NXTLNI(200),NXTLNO(400),LINPG,IDUMMY(73),KHRNUM,
     $   KHRIN(10),KHROUT(10)

      DATA NXTSUB,INDXI,INDXO,IER,MAXIN,MAXOUT,MAXSP,MINLIN,MAXLIN,
     $   MARRIT,MARLEF,INPAR,NUMPG,MARTOP,MARBOT,MAXPG,MINPG,
     $   INDPAR,NEWPAR,NEWPG,NEWLIN,NOWLIN,JUST,NUMSP,LINPG,NUMBR,
     $   CMPACT,KHRCON,LINMTY,GERMAN,NUMKHR,KHRIN,KHROUT
     $   / 8,1,1,1,80,80,80,1,81,80,1,0,1,2,81,800,1,6,
     $     .TRUE.,.TRUE.,.TRUE.,1,.TRUE.,1,1,.TRUE.,.TRUE.,
     $     .TRUE.,.TRUE.,.FALSE.,1,'@   ',9*0,'      ',9*0 /
```

Die Variablen haben die folgenden Bedeutungen:

NXTLNI..(INTEGER*4 ARRAY)	Eingabe Puffer.
NXTLNO..(INTEGER*4 ARRAY)	Ausgabe Puffer.
NXTSUB..(INTEGER*4)	Integer Code, um das naechste Unterprogramm zu verwenden.
INDXI...(INTEGER*4)	Die naechste Position in NXTLNI zu verarbeiten.
INDXO...(INTEGER*4)	Die naechste Position in NXTLNO zu verarbeiten.
IER.....(INTEGER*4)	Fehler Puffer.
MAXIN...(INTEGER*4)	Maximale Zeichenanzahl des Eingabe-Datei-Puffers.
MAXOUT..(INTEGER*4)	Maximale Spaltenanzahl des Ausgabe-Datei-Puffers.

MAXSP...(INTEGER*4)	Maximale Anzahl der leeren Zeilen zwischen zwei nicht-leeren Zeilen.
MINLIN..(INTEGER*4)	Minimale Anzahl der physikalischen Zeilen.
MAXLIN..(INTEGER*4)	Maximale physikalische Anzahl von Zeilen einer Seite.
MARRIT..(INTEGER*4)	Letzte moegliche nicht-Blank Spalte.
MARLEF..(INTEGER*4)	Erste moegliche nicht-Blank Spalte.
INPAR...(INTEGER*4)	Anzahl der leeren Zeilen zwischen den Absaetzen.
NUMPG...(INTEGER*4)	Gueltige Seitennummer.
MARTOP..(INTEGER*4)	Erste moegliche nicht-leere Zeile.
MARBOT..(INTEGER*4)	Letzte moegliche nicht-leere Zeile.
MAXPG...(INTEGER*4)	Maximale Anzahl der Seiten.
MINPGn.n.n.(INTEGER*4)	Minimale Anzahl der Seiten.
INDPAR..(INTEGER*4)	Anzahl von Blanks fuer einen Absatz.
NEWPAR..(LOGICAL)	Neuer Absatz.
NEWPG...(LOGICAL)	Neue Seite.
NEWLIN..(LOGICAL)	Neue Zeile.
NOWLIN..(INTEGER*4)	Laufende Zeilenummer der <Formatiertentextdatei>.
JUST....(LOGICAL)	Rechts buendig schieben.
NUMSP...(INTEGER*4)	Anzahl von <carriage return> bis zur naechsten Zeile.
LINPG...(INTEGER*4)	Zeilenummer in der die Seitennummer steht.
NUMBR...(LOGICAL)	Drucke Seitennummern.
CMPACT..(LOGICAL)	Komprimiere Zeilen.
KHRCON..(INTEGER*4)	Uebersetzen explizite Zeichen.
LINMTY..(INTEGER*4)	Leere Zeile.
GERMAN..(LOGICAL)	Deutsche Wort fuer Seitennummern.
IDUMMY..(INTEGER*4 ARRAY)	Nicht benutzt.
NUMKHR..(INTEGER*4)	Anzahl der expliziten Zeichen.

KHRIN...(INTEGER*4 ARRAY) Liste von expliziten Zeichen.

KHROUT..(INTEGER*4 ARRAY) Liste von Zeichen die zu den expliziten Zeichen gehoeren.

Falls eine neue Variable gebraucht wird, gibt es Platz in IDUMMY.

Anhang D. Kommandos Zusammenfassung

.bl n (Vorschub n Zeilen)

.bo n (n leere Zeilen am Ende der Seite)

.br (Vorschub eine Zeile)

.ce (zentrieren folgende Zeile)

.cm (Zeilen komprimieren)

.en (Ende Bearbeitung)

.er n (Fehlermeldungen)

.ex (die @-Zeichen werden gedruckt)

.fg n (n leere Zeilen fuer ein Bild)

.ju (worte rechts buendig)

.lm n (definiert linken Rand)

.ml n (definiert physikalisches Maximum von Zeilen)

.nc (nicht Zeilen komprimieren)

.ne (die Blankzeichen werden fuer @-Zeichen gedruckt)

.nj (nicht rechts buendig)

.nn (nicht Seitennummern drucken)

.nu n (setzt Seitennummer = n)

.pa n (Seitennummern auf Englisch)

.pg (Vorschub eine Seite)

.rm n (definiert rechten Rand)

.se n (Seitennummern auf Deutsch)

.sp n (setzt Vorschub zwischen Zeilen)

.to n (n leere Zeilen am Anfang der Seite)

LOGREA: A Text Formatter

Thomas C. Henderson

DFVLR Oberpfaffenhofen

August 10, 1980

Table of Contents

Introduction

LOGREA is a text formatter. Such a program aids in the development and production of reports and documentation by allowing rapid manipulation and formatting changes to text. For example, margins, indentation, spacing, etc. can be changed easily. Moreover, when the text is changed, running LOGREA reformats the entire report automatically.

This report is divided into two parts: Part I, a discussion of LOGREA from a user viewpoint, and Part II, a discussion of the programming details. The discussion holds for both the version implemented on the INTERDATA 8/32 and that on the AMDAHL. An up-to-date user guide to LOGREA is always available from the INTERDATA 8/32 on the file LOGREA.MAN/G. Appendix B gives details on the AMDAHL version of LOGREA.

Part I. How To Use LOGREA

LOGREA is designed for use on the INTERDATA 8/32. Familiarity with the MTM operating system is assumed. Figure 1 shows the MTM framework for using LOGREA.

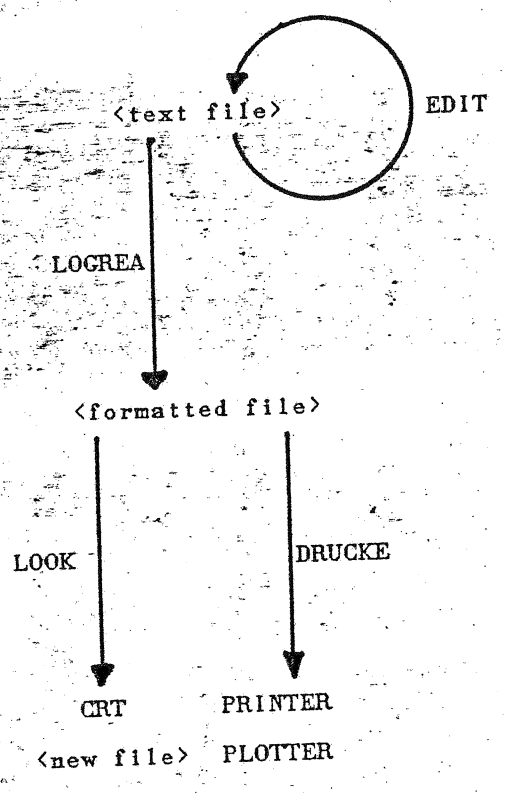


Figure 1 - MTM Framework for LOGREA

The <text file> is created or modified within EDIT (EDIT is a CSS file which starts the editor). The <text file> is a line-oriented file which contains two kinds of data:

- LOGREA commands, and
- text.

That is, each line is either a command to LOGREA or else text to appear in the <formatted file>. (The commands to LOGREA will not appear in the

<formatted file>.) The commands are interspersed throughout the <text file> and control the appearance of the <formatted file>.

The Text

The text is any sequence of characters (including upper/lower case). However, there are two exceptions to this. Since LOGREA commands can be anywhere in the file, there must be a special convention so that commands can be distinguished from user text. The convention is adopted that any line having a '.' (a period) in the first character position of the line is assumed to be a LOGREA command. The other convention is for the start of new paragraphs. Any line having a ' ' (a blank) in the first character position of the line is assumed to be the first line of a new paragraph. Otherwise, text can be any sequence of characters. If the <text file> contains only text, i.e., no LOGREA commands at all, then LOGREA will provide default values for margins, spacing, etc. (these defaults are given in Appendix C).

The Commands

The commands to LOGREA allow the user to specify margins, paragraph indentation, titles, blank lines, page numbers, etc. These commands can be interspersed throughout the text in the <text file>, but they will not appear in the <formatted file>. Each command must appear on a separate line and must have the following structure:

```
character
position:  1   2 3   4 5...
           . <command> <parameters>
```

Each command must be preceded by a period, and the period must be

located in the first column of the input line. Each <command> is a two letter sequence; <command> is separated from <parameters> by one blank, and <parameters> consists of a single integer number or blanks. The commands must all be lower case letters, i.e., a,b,c,...,z. The available commands are:

.bl n Output n blank lines. The spacing parameter (see .sp) has an effect on the number of lines. For example, double space (or .sp 2) and .bl 2 will produce 4 blank lines in the <formatted file>.

.bo n (Default n=64.) Set the bottom margin of the page to line n. This is the last line which will be printed before going to a new page.

.br Break. Output the current line and go to a new line. No justification of the broken line occurs.

.ce Center the line following this command. The following line will be centered between the left and right margins as set by the user.

.cm (Default is to compact successive lines.) Compact successive lines together from the <text file> to fill from the left margin as far as possible to the right margin of the <formatted file>. Compaction is a multi-line operation; see the .ju command for right margin justification.

.en End the text formatting. This command can be used to stop processing of a file before the physical end of file.

.er n (Default is n=1.) Error output trace can be controlled by this command. For n=0, no trace occurs at all. For n=1, any error message goes to the <formatted file>; that is, error messages appear in

the <formatted file> and will cause line counts to be wrong. For n=999, complete trace of the program is possible. The error messages usually indicate that the user has made a mistake in the <command> string, e.g., if it is misspelled.

.ex Do not perform character transformation of special characters, e.g., explicit blanks (see .ne command). This allows for '@' (the at sign) to be printed.

.fg n Skip n lines for a figure. If there are less than n lines on the current output page, go to the next page and skip n lines. Spacing does not affect the number of lines skipped. E.g., .sp 3 and .fg 10 results in 10 lines skipped.

.ju (Default is to perform justification.) This causes the words in a line to be right-justified to the right margin by putting extra blanks between words (see the .nj command).

.lm n (Default n=1.) Set the left margin to n. This is the first column into which text will be output on <formatted file>.

.ml n (Default n=81.) Set the physical line limit of the page to n lines. This allows the <formatted file> to be output to different output devices. For example, .ml 66 should be used for the PRINTER, whereas the default value gives a reasonable number for the PLOTTER (which can then be Xeroxed onto normal paper).

.nc Do not compact lines. This causes the <text file> to be copied as it appears to the <formatted file>. This is handy for a table of contents or for some types of figures.

.ne (Default is to perform explicit character conversion.) Print the related character of an explicit character. E.g., print ' '

(blank) for the at sign.

.nj Do not justify the text, but rather output each line as it has been compacted. This achieves a ragged right margin, but has regular spacing between the words (see the .ju command).

.nn No page numbers. This suppresses the printing of page numbers, although a page count is still maintained in LOGREA so that page numbering may be resumed.

.nu n (Default n=1.) Set the current page number to n, if n is greater than zero. If n=0, then resume printing page numbers; if n is less than zero, no page numbers will be printed, although a page count will still be kept in LOGREA.

.pa n (Default is n=1.) Output the page number in line n of the outout page. (Note that n should not be less than the line number of the top margin of the page.) The page number is preceded by the word 'Page'; for the word 'Seite', see the .se command.

.pg Page eject. Clear the current line and advance to a new page.

.rm n (Default is n=80.) Set the right margin to n. The last column to possibly contain a non-blank character will be column n in the <formatted file>.

.se n (Default is to put page numbers in English and in line 1.) Output the page number in line n of the output page. The page number will be preceded by the word 'Seite'. (Note that n should be less than the line number of the top line margin of the page.)

.sp n (Default n=1.) Set the interline spacing to n. For

example, .sp 1 means single space, while .sp 2 means double space.

.to n (Default n=1.) Set the top margin of the <formatted file>. The first line to be written on the <formatted file> will be line n. The lines 1 to (n-1) will be blank except for the line with the page number.

New paragraph - This is the one implicit command in LOGREA. Any line with a blank (' ') in column one is the signal to start a new paragraph. Thus, any line with a blank in column one will be put on a new line and indented (default is 5 spaces).

Special Features

Special features involve parameters that may or may not be set by the user and which do not really correspond to commands. For example, underlining, word hyphenation and special characters. The currently implemented features include:

Explicit blank - Sometimes it is necessary to be able to specify exactly the number of blanks between two consecutive words or to be able to specify that they should not be split across lines. However, blanks in the <text file> are disregarded, and some other character must be used to explicitly state that a blank is desired. The character "@" (the at sign) will be used for this purpose and will be changed to a blank just before output. Otherwise, it is treated as a non-blank character. This provides a mechanism to control the spacing between words and the indention from the left margin (see Appendix A for examples of the use of this feature).

An Example

The best way to become familiar with LOGREA is to work through an example. Appendix A gives the <text file> for this report. This report is the <formatted file>.

How To LOOK at the Result

Once the <formatted file> is available, there are several output options available. Hardcopy can be produced by the following call:

DRUCKE <formatted file>,1,PLT:

where <formatted file> means the name of the actual file which contains the formmated text.This command produces an upper/lower case copy of the <formatted file> on the VERSATEK. For an upper case only copy, use:

DRUCKE <formatted file>

which produces a copy on the printer. However, the <formatted file> can be inspected from the CRT by using the LOOK facility.

LOOK has two parameters:

LOOK <formatted file>,<new file>

where the <new file> is a file already allocated by the user (and should be an indexed type file). The <new file> can be used to store selected pages from the complete <formatted file>. In this way one can make hard copy of selected parts of the complete report. LOOK first asks:

ZEILEN PRO SEITE I2 (XX)

To this the user responds with the maximum physical page length of his <formatted file>. This number should be typed in FORTRAN FORMAT I2. The default page length for LOGREA is 81, and unless the .ml command is used in the <text file> to alter the physical page length, the value of 81 should be given to LOOK. Next LOOK allows three options: SEIT, AUSG and ENDE to be selected by the user.

SEITnnn - the word SEIT followed by an integer in FORTRAN FORMAT

I3 causes LOOK to display the physical page corresponding to that integer, i.e., LOGREA page numbers are not used. Each page is shown twenty-two (22) lines at a time, and a carriage return is required to see more of the page. When the entire page has been displayed, the user must again select one of the three options. If it is desired to simply continue on to the next page, a carriage return will suffice. If during the display of a page, the user wishes to change option or page, then at the 22 line pause, typing an 'E' (the capital letter e) will give the opportunity of selecting a new option. If one wishes to see the previous page, merely enter a '-' (minus sign).

AUSGnnn - the word AUSG followed by an integer in FORTRAN FORMAT I3 causes LOOK to show that entire page on the CRT and to copy it to the <new file>. The contents of the <new file> can be examined either in LOOK, EDIT or hard copy output.

ENDE - the word ENDE causes LOOK to stop processing.

Thus, LOOK allows the user to examine the <formatted file> without having to produce hard copy for change. (Note that it is not easy to look at the <formatted file> within EDIT since the lines are usually too long to fit on one CRT line.) LOOK can be used to detect errors quickly or to let someone else see the body of the text.

Part II. LOGREA Programming Details

LOGREA is a fairly large program consisting of about 2500 lines of FORTRAN code. Modular design in both the data structure and procedure interface has been exploited in order to aid comprehensibility. The entire code will not be examined here, since each subprogram is internally documented. The main goal to be achieved in this part of the report is to document the overall conceptual design of the system and to detail what the associated CSS files are and how they work.

LOGREA consists of 51 subprograms all organized around a COMMON block which contains the parameters of the text formatter. Appendix C contains a description of the COMMON block, /LOGVAR/. The logical organization of the text formatting system is shown in Figure 2.

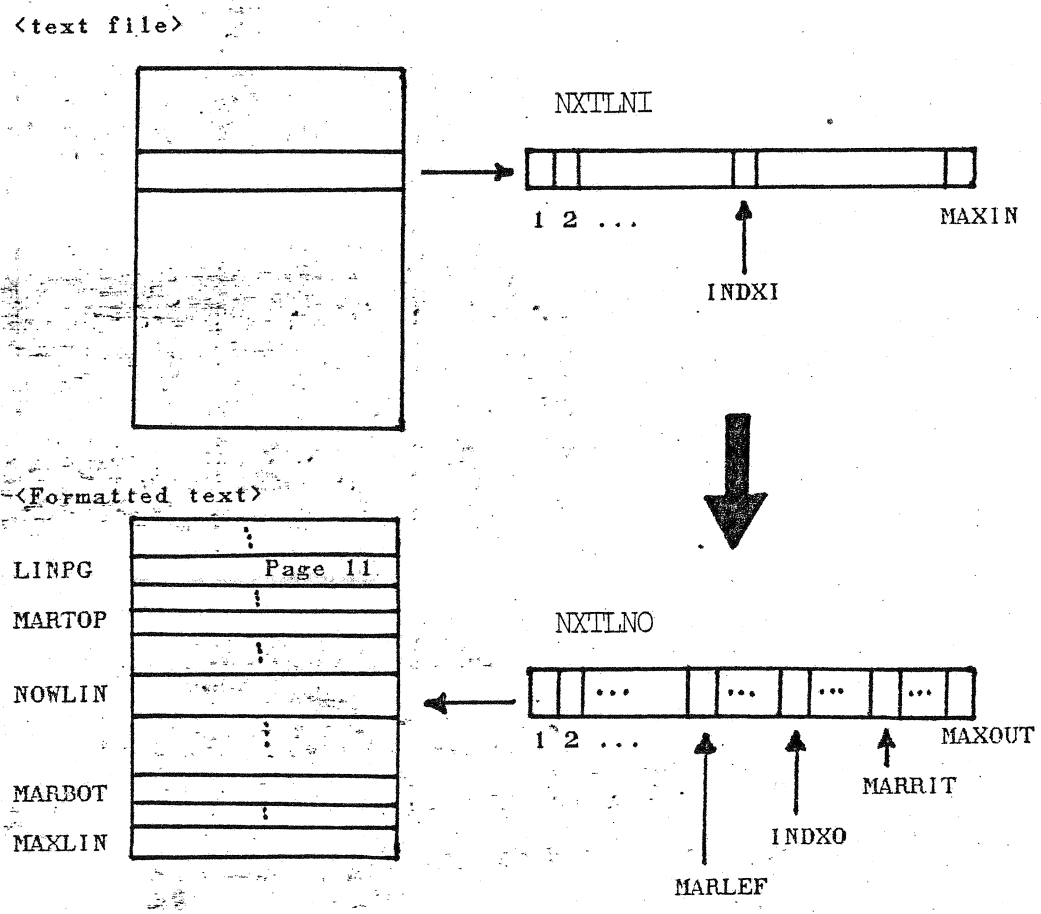


Figure 2 - Text Formatting File and Buffer Organization

Each line of the <text file> is processed individually. The line is read in to the input buffer, NXTLNI. As the line is processed, words may be inserted into the output buffer, NXTLNO. Up to the left margin of NXTLNO is blank filled, as is after the right margin. Words are copied from NXTLNI to NXTLNO until not enough room is left. Then NXTLNO is right-justified (if JUST=.TRUE.) and output.

The main program of LOGREA proceeds as follows:

(1) get next line of <text file>,

(2) decide if line is a LOGREA command or text,

(3) if text, then process to <formatted file>,

(4) if command, then branch to the appropriate subprogram which corresponds to that cammand.

In this way, a new command is easily added by:

- adding the character code in SETSUB,
- adding a branch to the computed GO TO statement in the main program, and
- writing the appropriate subprogram.

As for the character code, Table 1 shows the codes for the necessary characters. In writing the new subprogram, a standard procedure form can be used, e.g., a new command with no parameters can operate directly on the COMMON block, and one with parameters can be patterned after BOTMAR (the .bo command).

Table 1 - Character Codes

Character	INTERDATA 8/32	AMDAHL
a	61	81
b	62	82
c	63	82
d	64	83
e	65	85
f	66	86
g	67	87
h	68	88
i	69	89
j	6A	91
k	6B	92
l	6C	93
m	6D	94
n	6E	95
o	6F	96
p	70	97
q	71	98
r	72	99
s	73	A2
t	74	A3
u	75	A4
v	76	A5
w	77	A6
x	78	A7
y	79	A8
z	7A	A9
S	53	E2
P	50	D7
.	2E	4B

LOGREA.CSS

The LOGREA.CSS file is as follows:

```
*
* LOGREA  <FILENAME1>,<FILENAME2>
*
*   REFORMAT TEXT ON <FILENAME1> TO <FILENAME2>.
*
*     FILE USAGE      UNIT      FILE NAME
*                               (DEFAULT)
*    ------------------------------------
*
*       INPUT          5          CON:
*       OUTPUT         6          CON:
*       TRACE INPUT    7          CON:
*
LO LOGREA.TSK
$IFNU  1; AS 5,CON:; $ENDC;
$IFNN  1; AS 5, 1; $ENDC;
$IFNN  2; XDE  2; AL  2,IN,126; $ENDC;
$IFNU  2; AS 6,CON:; $ENDC;
$IFNN  2; AS 6, 2; $ENDC;
AS 7,CON:
ST
$EXIT
```

Thus, no <formatted file> need be allocated by the user since the CSS file will do it automatically. Note that any existant file with the output file name will be destroyed. Moreover, when debugging, no <formatted file> need be specified at all, and error debug messages will go to the CRT (i.e., CON:).

Appendix A. Input File To LOGREA To Produce This Report

```
.to 5
.bo 70
.lm 10
.pa 2
.sp 2
.nn
.bl 10
.ce
LOGREA: A Text Formatter
.bl 6
.ce
Thomas C. Henderson
.bl 1
.ce
DFVLR Oberpfaffenhofen
.ce
August 10, 1980
.pg
.ce
Table of Contents
.bl 3
.nc
```



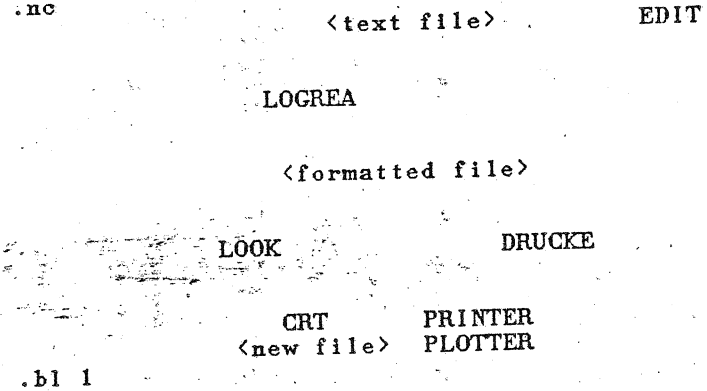
```
.nu 1
.bl 1
.cm
.pg
.ce
Introduction
.bl 1
```

LOGREA is a text formatter. Such a program aids in the development and production of reports and documentation by allowing rapid manipulation and formatting changes to text. For example, margins, indentation, spacing, etc. can be changed easily. Moreover, when the text is changed, running LOGREA reformats the entire report automatically.

This report is divided into two parts: Part I, a discussion of LOGREA from a user viewpoint, and Part II, a discussion of the programming details. The discussion holds for both the version implemented on the INTERDATA 8/32 and that on the AMDAHL. An up-to-date user guide to LOGREA is always available from the INTERDATA 8/32 on the file LOGREA.MAN/G. Appendix B gives details on the AMDAHL version of LOGREA.

```
.pg
.ce
```

Part I. How To Use LOGREA
.bl 2
LOGREA is designed for use on the INTERDATA 8/32. Familiarity with the MTM operating system is assumed. Figure 1 shows the MTM framework for using LOGREA.
.bl 2
.nc



.bl 1
.ce
Figure 1 - MTM Framework for LOGREA
.bl 2
.cm
The <text file> is created or modified within EDIT (EDIT is a CSS file which starts the editor). The <text file> is a line-oriented file which contains two kinds of data:
.br
- LOGREA commands, and
.br
- text.
.br
That is, each line is either a command to LOGREA or else text to appear in the <formatted file>. (The commands to LOGREA will not appear in the <formatted file>.) The commands are interspersed throughout the <text file> and control the appearance of the <formatted file>.
.bl 2
.ce
The Text
.bl 1
The text is any sequence of characters (including upper/lower case). However, there are two exceptions to this. Since LOGREA commands can be anywhere in the file, there must be a special convention so that commands can be distinguished from user text. The convention is adopted that any line having a '.' (a period) in the first character position of the line is assumed to be a LOGREA command. The other convention is for the start of new paragraphs. Any line having a ' ' (a blank) in the first character position of the line is assumed to be the first line of a new paragraph. Otherwise, text can be any sequence of characters. If the <text file> contains only text, i.e., no LOGREA commands at all, then LOGREA will provide default values for margins, spacing, etc. (these defaults are given in Appendix C).
.bl 2
.ce
The Commands
.bl 1
The commands to LOGREA allow the user to specify margins, paragraph indention, titles, blank lines, page numbers, etc. These commands can be interspersed throughout the text in the <text file>, but they will not appear in the <formatted file>. Each command must appear on a separate line and must have the following structure:
.br
.bl 1
character

.br
position: 1 2 3 4 5...
.br
. <command> <parameters>
.br
.bl 1
Each command must be preceded by a period, and the period must be located in the first column of the input line. Each <command> is a two letter sequence; <command> is separated from parameters> by one blank, and <parameters> consists of a single integer number or blanks. The commands must all be lower case letters, i.e., a,b,c,...,z. The available commands are:
.bl 1
.bl n Output n blank lines. The spacing parameter (see .sp) has an effect on the number of lines. For example, double space (or .sp 2) and .bl 2 will produce 4 blank lines in the <formatted file>.
.bl 1
.bo n (Default n=64.) Set the bottom margin of the page to line n. This is the last line which will be printed before going to a new page.
.bl 1
.br Break. Output the current line and go to a new line. No justification of the broken line occurs.
.bl 1
.ce Center the line following this command. The following line will be centered between the left and right margins as set by the user.
.bl 1
.cm (Default is to compact successive lines.) Compact successive lines together from the <text file> to fill from the left margin as far as possible to the right margin of the <formatted file>. Compaction is a multi-line operation; see the .ju command for right margin justification.
.bl 1
.en End the text formatting. This command can be used to stop processing of a file before the physical end of file.
.bl 1
.er n (Default is n=1.) Error output trace can be controlled by this command. For n=0, no trace occurs at all. For n=1, any error message goes to the <formatted file>; that is, error messages appear in the <formatted file> and will cause line counts to be wrong. For n=999, complete trace of the program is possible. The error messages usually indicate that the user has made a mistake in the <command> string, e.g., if it is misspelled.
.bl 1
.ex Do not perform character transformation of special characters, e.g., explicit blanks (see .ne command). This allows for
.ex
' ' (the at sign) to be printed.
.bl 1
.ne
.fg n Skip n lines for a figure. If there are less than n lines on the current output page, go to the next page and skip n lines. Spacing does not affect the number of lines skipped. E.g., .sp 3 and .fg 10 results in 10 lines skipped.
.bl 1
.ju (Default is to perform justification.) This causes the words in a line to be right-justified to the right margin by putting extra blanks between words (see the .nj command).
.bl 1
.lm n (Default n=1.) Set the left margin to n. This is the first column into which text will be output on <formatted file>.
.bl 1
.ml n (Default n=81.) Set the physical line limit of the page to n lines. This allows the <formatted file> to be output to different output devices. For example, .ml 66 should be used for

the PRINTER, whereas the default value gives a reasonable number for the PLOTTER (which can then be Xeroxed onto normal paper).
.bl 1
.nc Do not compact lines. This causes the <text file> to be copied as it appears to the <formatted file>. This is handy for a table of contents or for some types of figures.
.bl 1
.ne (Default is to perform explicit character conversion.) Print the related character of an explicit character. E.g., print ' ' (blank) for the at sign.
.bl 1
.nj Do not justify the text, but rather output each line as it has been compacted. This achieves a ragged right margin, but has regular spacing between the words (see the .ju command).
.bl 1
.nn No page numbers. This suppresses the printing of page numbers, although a page count is still maintained in LOGREA so that page numbering may be resumed.
.bl 1
.nu n (Default n=1.) Set the current page number to n, if n is greater than zero. If n=0, then resume printing page numbers; if n is less than zero, no page numbers will be printed, although a page count will still be kept in LOGREA.
.bl 1
.pa n (Default is n=1.) Output the page number in line n of the outout page. (Note that n should not be less than the line number of the top margin of the page.) The page number is preceded by the word 'Page'; for the word 'Seite', see the .se command.
.bl 1
.pg Page eject. Clear the current line and advance to a new page.
.bl 1
.rm n (Default is n=80.) Set the right margin to n. The last column to possibly contain a non-blank character will be column n in the <formatted file>.
.bl 1
.se n (Default is to put page numbers in English and in line 1.) Output the page number in line n of the output page. The page number will be preceded by the word 'Seite'. (Note that n should be less than the line number of the top line margin of the page.)
.bl 1
.sp n (Default n=1.) Set the interline spacing to n. For example, .sp 1 means single space, while .sp 2 means double space.
.bl 1
.to n (Default n=1.) Set the top margin of the <formatted file>. The first line to be written on the <formatted file> will be line n. The lines 1 to (n-1) will be blank except for the line with the page number.
.bl 1
New paragraph - This is the one implicit command in LOGREA. Any line with a blank (' ') in column one is the signal to start a new paragraph. Thus, any line with a blank in column one will be put on a new line and indented (default is 5 spaces).
.bl 2
.ce
Special Features
.bl 1
Special features involve parameters that may or may not be set by the user and which do not really correspond to commands. For example, underlining, word hyphenation and special characters. The currently implemented features include:
.bl 1
Explicit blank - Sometimes it is necessary to be able to specify exactly the number of blanks between two consecutive words or to be able to specify that they should not be split across lines. However, blanks in the <text file> are disregarded, and some other character

must be used to explicitly state that a blank is desired. The
.ex
character " " (the at sign) will be used for this purpose and will
be changed to a blank just before output. Otherwise, it is treated
.ne
as a non-blank character. This provides a mechanism to control the
spacing between words and the indention from the left margin (see
Appendix A for examples of the use of this feature).
.pg
.ce
An Example
.bl 1
The best way to become familiar with LOGREA is to work through an example. Appendix A gives the <text file> for this report. This report is the <formatted file>.
.bl 2
.ce
How To LOOK at the Result
.bl 1
Once the <formatted file> is available, there are several output options available. Hardcopy can be produced by the following call:
.br
DRUCKE <formatted file>,1,PLT:
.br
where <formatted file> means the name of the actual file which contains the formmated text.This command produces an upper/lower case copy of the <formatted file> on the VERSATEK. For an upper case only copy, use:
.br
DRUCKE <formatted file>
.br
which produces a copy on the printer. However, the <formatted file> can be inspected from the CRT by using the LOOK facility.
LOOK has two parameters:
.br
LOOK <formatted file>,<new file>
.br
where the <new file> is a file already allocated by the user (and should be an indexed type file). The <new file> can be used to store selected pages from the complete <formatted file>. In this way one can make hard copy of selected parts of the complete report. LOOK first asks:
.br
ZEILEN PRO SEITE I2 (XX)
.br
To this the user responds with the maximum physical page length of his <formatted file>. This number should be typed in FORTRAN FORMAT I2. The default page length for LOGREA is 81, and unless the .ml command is used in the <text file> to alter the physical page length, the value of 81 should be given to LOOK. Next LOOK allows three options SEIT, AUSG and ENDE to be selected by the user.
SEITnnn - the word SEIT followed by an integer in FORTRAN FORMAT I3 causes LOOK to display the physical page corresponding to that integer, LOGREA page numbers are not used. Each page is shown twenty-two (22) lines at a time, and a carriage return is required to see more of the page. When the entire page has been displayed, the user must again select one of the three options. If it is desired to simply continue on to the next page, a carriage return will suffice. If during the display of a page, the user wishes to change option or page, then at th 22 line pause, typing an 'E' (the capital letter e) will give the opportunity of selecting a new option. If one wishes to see the previous page, merely enter a '-' (minus sign).
AUSGnnn - the word AUSG followed by an integer in FORTRAN FORMAT I3 causes LOOK to show that entire page on the CRT and to copy it to the <new file>.
The contents of the <new file> can be examined either in LOOK, EDIT

or hard copy output.
ENDE - the word ENDE causes LOOK to stop processing.
.bl 1
Thus, LOOK allows the user to examine the <formatted file> without having to produce hard copy for change. (Note that it is not easy to look at the <formatted file> within EDIT since the lines are usually too long to fit on one CRT line.) LOOK can be used to detect errors quickly or to let someone else see the body of the text.
.pg
.ce
Part II. LOGREA Programming Details
.bl 2
LOGREA is a fairly large program consists of about 2500 lines of FORTRAN code. Modular design in both the data structure and procedure interface has been exploited in order to aid comprehensibility. The entire code will not be examined here, since each subprogram is internally documented. The main goal to be achieved in this part of the report is to document the overall conceptual design of the system and to detail what the associated CSS files are and how they work.
LOGREA consists of 51 subprograms all organized around a COMMON block which contains the parameters of the text formatter. Appendix C contains a description of the COMMON block, /LOGVAR/. The logical organization of the text formatting system is shown in Figure 2.
.pg
.bl 25
.ce
Figure 2 - Text Formatting File and Buffer Organization
.bl 1
Each line of the <text file> is processed individually. The line is read in to the input buffer, NXTLNI. As the line is processed, words may be inserted into the output buffer, NXTLNO. Up to the left margin of NXTLNO is blank filled, as is after the right margin. Words are copied from NXTLNI to NXTLNO until not enough room is left. Then NXTLNO is right-justified (if JUST=.TRUE.) and output.
The main program of LOGREA proceeds as follows:
.br
(1) get next line of <text file>,
.br
(2) decide if line is a LOGREA command or text,
.br
(3) if text, then process to <formatted file>,
.br
(4) if command, then branch to the appropriate subprogram which corresponds to that cammand.
.br
In this way, a new command is easily added by:
.br
- adding the character code in SETSUB,
.br
- adding a branch to the computed GO TO statement in the main program, and
.br
- writing the appropriate subprogram.
.br
As for the character code, Table 1 shows the codes for the necessary characters. In writing the new subprogram, a standard procedure form can be used, e.g., a new command with no parameters can operate directly on the COMMON block, and one with parameters can be patterned after BOTMAR (the .bo command).
.bl 1
.pg
.sp 1
.ce
Table 1 - Character Codes
.bl 2

```
.nc
```

Character	INTERDATA 8/32	AMDAHL
a	61	81
b	62	82
c	63	82
d	64	83
e	65	85
f	66	86
g	67	87
h	68	88
i	69	89
j	6A	91
k	6B	92
l	6C	93
m	6D	94
n	6E	95
o	6F	96
p	70	97
q	71	98
r	72	99
s	73	A2
t	74	A3
u	75	A4
v	76	A5
w	77	A6
x	78	A7
y	79	A8
z	7A	A9
S	53	E2
P	50	D7
.	2E	4B

```
.cm
.sp 2
.pg
.ce
LOGREA.CSS
.bl 2
 The LOGREA.CSS file is as follows:
.br
.sp 1
.nc
*
* LOGREA  <FILENAME1>,<FILENAME2>
*
*   REFORMAT TEXT ON <FILENAME1> TO <FILENAME2>.
*
*     FILE USAGE      UNIT      FILE NAME
*                               (DEFAULT)
*    ------------------------------------
*
*        INPUT          5         CON:
*        OUTPUT         6         CON:
*        TRACE INPUT    7         CON:
*
LO LOGREA.TSK
$IFNU  1; AS 5,CON:; $ENDC;
$IFNN  1; AS 5, 1; $ENDC;
$IFNN  2; XDE  2; AL  2,IN,126; $ENDC;
$IFNU  2; AS 6,CON:; $ENDC;
$IFNN  2; AS 6, 2; $ENDC;
AS 7,CON:
ST
```

```
$EXIT
.cm
.sp 2
.bl 1
Thus, no <formatted file> need be allocated by the user since the
CSS file will do it automatically. Note that any existant file with
the output file name will be destroyed. Moreover, when debugging,
no <formatted file> need be specified at all, and error debug
messages will go to the CRT (i.e., CON:).
.pg
.ce
Appendix A. Input File To LOGREA To Produce This Report
.bl 2
.sp 1
(...ad infinitum...)
.pg
.ce
Appendix B. AMDAHL Version of LOGREA
.bl 2
 LOGREA has been implemented on the AMDAHL since an upper/lower case
printer is available on that machine as is a much better text editor.
Text may be entered via the AMDAHL editor or by magnetic tape. Text
created on the INTERDATA 8/32 can be stored to magnetic tape using
TAPECARD, a CSS file available under MTM. This tape can then be
transported to the AMDAHL. In the latter case, upper/lower case
output to plain white hole-punched printer paper can be accomplished
by choosing the SPF option 6 and issuing:
.br
.bl 1
     EXEC 'NT23.CLIST(LOGREA)' 'infile,outfile'
.br
.bl 1
Alternatively, the <text file> may be edited directly on the AMDAHL,
but the file should be edited in text mode so as to avoid sequence
numbers in columns 73 to 80. If a file with sequence numbers must be
processed, a special command exists for the AMDAHL LOGREA. This
command is the .mi command, and should be used to set the input line
length to 72.
 The AMDAHL FORTRAN does not provide the logical function AND, OR
or SHIFT; therefore, these have been written especially for LOGREA in
assembly language.
.pg
.ce
Appendix C. /LOGVAR/ the COMMON Block
.bl 2
 The default values of the COMMON block are set in the block data
subprogram as:
.br
.bl 1
.nc
      LOGICAL JUST,NEWPAR,NEWPG,NEWLIN,NUMBR,CMPACT,KHRCON,LINMTY,
     $   GERMAN
      INTEGER*4 NXTLNI(200),NXTLNO(400),LINPG,IDUMMY(73),KHRNUM,
     $   KHRIN(10),KHROUT(10)

      DATA NXTSUB,INDXI,INDXO,IER,MAXIN,MAXOUT,MAXSP,MINLIN,MAXLIN,
     $   MARRIT,MARLEF,INPAR,NUMPG,MARTOP,MARBOT,MAXPG,MINPG,
     $   INDPAR,NEWPAR,NEWPG,NEWLIN,NOWLIN,JUST,NUMSP,LINPG,NUMBR,
     $   CMPACT,KHRCON,LINMTY,GERMAN,NUMKHR,KHRIN,KHROUT
     $  / 8,1,1,1,80,80,80,1,81,80,1,0,1,2,81,800,1,6,
     $     .TRUE.,.TRUE.,.TRUE.,1,.TRUE.,1,1,.TRUE.,.TRUE.,
.ex
     $     .TRUE.,.TRUE.,.FALSE.,1,'     ',9*0,'      ',9*0 /
.ne
.bl 1
.cm
```

The variables in /LOGVAR/ have the following meanings:

.br
.bl 1
.nc

NXTLNI..(INTEGER*4 ARRAY)	Input buffer.
NXTLNO..(INTEGER*4 ARRAY)	Output buffer.
NXTSUB..(INTEGER*4)	Integer code corresponding to next subroutine to execute.
INDXI...(INTEGER*4)	Index into NXTLNI for next position to process.
INDXO...(INTEGER*4)	Index into NXTLNO for next output buffer position.
IER.....(INTEGER*4)	Error flag.
MAXIN...(INTEGER*4)	Maximum number of characters to be read into NXTLNI.
MAXOUT..(INTEGER*4)	Maximum number of characters to be printed on <formatted file>.
MAXSP...(INTEGER*4)	Maximum value of line spacing.
MINLIN..(INTEGER*4)	Minimum physical line number.
MAXLIN..(INTEGER*4)	Maximum number of physical lines per page.
MARRIT..(INTEGER*4)	Pointer to right margin: last possible non-blank character of <formatted file>.
MARLEF..(INTEGER*4)	Pointer to left margin: first possible non-blank character of <formatted file>.
INPAR...(INTEGER*4)	Number of lines to space between paragraphs.
NUMPG...(INTEGER*4)	Current page number.
MARTOP..(INTEGER*4)	Pointer to top margin: first possible non-blank line of <formatted file>.
MARBOT..(INTEGER*4)	Pointer to botom margin: last possible non-blank line of <formatted file>.
MAXPG...(INTEGER*4)	Maximum number of pages of <formatted file>.
MINPG...(INTEGER*4)	Minimum page number.
INDPAR..(INTEGER*4)	Number of spaces to indent paragraph.
NEWPAR..(LOGICAL)	New paragraph switch.
NEWPG...(LOGICAL)	New page switch.
NEWLIN..(LOGICAL)	New line switch.
NOWLIN..(INTEGER*4)	Current line number being processed on <formatted file>.
JUST....(LOGICAL)	Right-justify switch.
NUMSP...(INTEGER*4)	Number of carriage returns from current line to next line.
LINPG...(INTEGER*4)	Number of the line in which the page number appears.
NUMBR...(LOGICAL)	Print page number switch.
CMPACT..(LOGICAL)	Line compact switch.
KHRCON..(LOGICAL)	Explicit character conversion switch.
LINMTY..(LOGICAL)	Empty line switch; i.e., no text has yet been inserted in the output buffer.
GERMAN..(LOGICAL)	German word for page numbers switch.
IDUMMY..(INTEGER*4 ARRAY)	Extra space for new variables.
NUMKHR..(INTEGER*4)	Number of explicit characters.
KHRIN...(INTEGER*4 ARRAY)	List of explicit characters used in <text file>.
KHROUT..(INTEGER*4 ARRAY)	List of corresponding characters to be printed in <formatted file>.

.cm
.bl 1
Any new variable needed for new LOGREA commands can be inserted into the IDUMMY array.
.pg
.ce
Appendix D. Brief Command List
.bl 2

```
.sp 2
.nc
    .bl n    (n blank lines)
    .bo n    (bottom margin line n)
    .br      (break; output current line)
    .ce      (center line following this command)
    .cm      (compact successive lines)
    .en      (end text formatting)
    .er n    (error trace switch)
    .ex      (print explicit characters)
    .fg n    (leave n lines blank for a figure)
    .ju      (right-justify margin)
    .lm n    (set left margin to column n)
    .ml n    (set physical maximum lines)
    .nc      (do not compact successive lines)
    .ne      (perform explicit character conversion)
    .nj      (do not right-justify text)
    .nn      (do not print page numbers)
    .nu n    (set page number to n and print page numbers)
    .pa n    (print page number in English in line n)
    .pg      (page eject)
    .rm n    (set right margin to column n)
    .se n    (print page number in German in line n)
    .sp n    (set line spacing to n)
    .to n    (top line margin line n)
.cm
```

Appendix B. AMDAHL Version of LOGREA

LOGREA has been implemented on the AMDAHL since an upper/lower case printer is available on that machine as is a much better text editor. Text may be entered via the AMDAHL editor or by magnetic tape. Text created on the INTERDATA 8/32 can be stored to magnetic tape using TAPECARD, a CSS file available under MTM. This tape can then be transported to the AMDAHL. In the latter case, upper/lower case output to plain white hole-punched printer paper. LOGREA can be used by choosing the SPF option 6 and issuing:

```
EXEC 'NT23.CLIST(LOGREA)' 'infile,outfile'
```

Alternatively, the <text file> may be edited directly on the AMDAHL, but the file should be edited in text mode so as to avoid sequence numbers in columns 73 to 80. If a file with sequence numbers must be processed, a special command exists for the AMDAHL LOGREA. This command is the .mi command, and should be used to set the input line length to 72.

The AMDAHL FORTRAN does not provide the logical function AND, OR or SHIFT; therefore, these have been written especially for LOGREA in assembly language.

Appendix C. /LOGVAR/ the COMMON Block

The default values of the COMMON block are set in the block data subprogram as:

```
      LOGICAL JUST,NEWPAR,NEWPG,NEWLIN,NUMBR,CMPACT,KHRCON,LINMTY,
     $   GERMAN
      INTEGER*4 NXTLNI(200),NXTLNO(400),LINPG,IDUMMY(73),KHRNUM,
     $   KHRIN(10),KHROUT(10)

      DATA NXTSUB,INDXI,INDXO,IER,MAXIN,MAXOUT,MAXSP,MINLIN,MAXLIN,
     $   MARRIT,MARLEF,INPAR,NUMPG,MARTOP,MARBOT,MAXPG,MINPG,
     $   INDPAR,NEWPAR,NEWPG,NEWLIN,NOWLIN,JUST,NUMSP,LINPG,NUMBR,
     $   CMPACT,KHRCON,LINMTY,GERMAN,NUMKHR,KHRIN,KHROUT
     $   / 8,1,1,1,80,80,80,1,81,80,1,0,1,2,81,800,1,6,
     $     .TRUE.,.TRUE.,.TRUE.,1,.TRUE.,1,1,.TRUE.,.TRUE.,
     $     .TRUE.,.TRUE.,.FALSE.,1,'@   ',9*0,'     ',9*0 /
```

The variables in /LOGVAR/ have the following meanings:

NXTLNI..(INTEGER*4 ARRAY)	Input buffer.
NXTLNO..(INTEGER*4 ARRAY)	Output buffer.
NXTSUB..(INTEGER*4)	Integer code corresponding to next subroutine to execute.
INDXI...(INTEGER*4)	Index into NXTLNI for next position to process.
INDXO...(INTEGER*4)	Index into NXTLNO for next output buffer position.
IER.....(INTEGER*4)	Error flag.
MAXIN...(INTEGER*4)	Maximum number of characters to be read into NXTLNI.
MAXOUT..(INTEGER*4)	Maximum number of characters to be

	printed on <formatted file>.
MAXSP...(INTEGER*4)	Maximum value of line spacing.
MINLIN..(INTEGER*4)	Minimum physical line number.
MAXLIN..(INTEGER*4)	Maximum number of physical lines per page.
MARRIT..(INTEGER*4)	Pointer to right margin: last possible non-blank character of <formatted file>.
MARLEF..(INTEGER*4)	Pointer to left margin: first possible non-blank character of <formatted file>.
INPAR...(INTEGER*4)	Number of lines to space between paragraphs.
NUMPG...(INTEGER*4)	Current page number.
MARTOP..(INTEGER*4)	Pointer to top margin: first possible non-blank line of <formatted file>.
MARBOT..(INTEGER*4)	Pointer to botom margin: last possible non-blank line of <formatted file>.
MAXPG...(INTEGER*4)	Maximum number of pages of <formatted file>.
MINPG...(INTEGER*4)	Minimum page number.
INDPAR..(INTEGER*4)	Number of spaces to indent paragraph.
NEWPAR..(LOGICAL)	New paragraph switch.
NEWPG...(LOGICAL)	New page switch.
NEWLIN..(LOGICAL)	New line switch.
NOWLIN..(INTEGER*4)	Current line number being processed on <formatted file>.
JUST....(LOGICAL)	Right-justify switch.
NUMSP...(INTEGER*4)	Number of carriage returns from current line to next line.
LINPG...(INTEGER*4)	Number of the line in which the page number appears.
NUMBR...(LOGICAL)	Print page number switch.
CMPACT..(LOGICAL)	Line compact switch.
KHRCON..(LOGICAL)	Explicit character conversion switch.

LINMTY..(LOGICAL)	Empty line switch; i.e., no text has yet been inserted in the output buffer.
GERMAN..(LOGICAL)	German word for page numbers switch.
IDUMMY..(INTEGER*4 ARRAY)	Extra space for new variables.
NUMKHR..(INTEGER*4)	Number of explicit characters.
KHRIN...(INTEGER*4 ARRAY)	List of explicit characters used in <text file>.
KHROUT..(INTEGER*4 ARRAY)	List of corresponding characters to be printed in <formatted file>.

Any new variable needed for new LOGREA commands can be inserted into the IDUMMY array.

Appendix D. Brief Command List

.bl n (n blank lines)

.bo n (bottom margin line n)

.br (break; output current line)

.ce (center line following this command)

.cm (compact successive lines)

.en (end text formatting)

.er n (error trace switch)

.ex (print explicit characters)

.fg n (leave n lines blank for a figure)

.ju (right-justify margin)

.lm n (set left margin to column n)

.ml n (set physical maximum lines)

.nc (do not compact successive lines)

.ne (perform explicit character conversion)

.nj (do not right-justify text)

.nn (do not print page numbers)

.nu n (set page number to n and print page numbers)

.pa n (print page number in English in line n)

.pg (page eject)

.rm n (set right margin to column n)

.se n (print page number in German in line n)

.sp n (set line spacing to n)

.to n (top line margin line n)