1. TEST03, MEMORY MARCH TEST LOW RANGE

2. Funktion des Programms

Der "Memory March Test Low Range" prueft die Funktion des Speichers im unteren Speicherbereich von der Anfangsadresse: 0:0010H bis zur Endadresse: 1000:FFFFH mit dem Marching - Algorithmus.
Der Marching - Algorithmus wird durch geeignete Wahl des Datenmusters einmal auf die Speicherschaltkreise fuer die Datenbits und zum anderen auf die fuer die Paritaetsbits angewendet.
Am Beginn der Pruefung wird der vorhandene Speicherinhalt gerettet und am Programmende wieder zurueckgeschrieben.

3. Voraussetzungen

Im Folgenden werden Speicheradressen in der Form Segment:Offset hexadezimal angegeben, wobei das Segment durch einen symbolischen Namen bezeichnet wird. Die Zuordnung von symbolischem Segmentnamen und absoluten Adressen ist aus der MP2-Liste zu entnehmen.
In speziellen Faellen (z.B. Darstellung der Fehlerausschrift oder Bereichsangaben) wird die Speicheradresse absolut mit 5 hexadezimalen Zahlen (Kennzeichnung mit H) angegeben.

3.1. Geraeteausruestung

Es ist die Grundausruestung des Rechners A 7100 erforderlich. Die Ein/Ausgabe richtet sich nach der im Monitor generierten Geraetekonfiguration.

3.2. Speicherbedarf

ca. 1,3 KB fuer TEST03.OBJ
Die absoluten Adressen der durch den Locator zugewiesenen Speicherbereiche sind anhand der MP2-Liste zu berechnen.

3.3. Test zusaetzlicher Baugruppen

TEST03 erfordert u.a. eine in den wesentlichen Teilen funktionierende ZVE einschliesslich der Interruptlogik. Insbesondere wird in der ZVE die richtige Adressierung der Speicherzellen und der Datenaustausch ueber den Systembus bzw. internen X2-Bus vorausgesetzt.

3.4. Nutzung anderer Programmodule

Der Testmodul TEST03.OBJ muss mit den Hilfsmodulen MEMCOM.OBJ, MEMLIM.OBJ, MEMBRR.OBJ, dem Leitprogrammodul LACS.OBJ und der Bibliothek LACS.LIB verbunden sein.
Es werden Unterprogramme (Routinen) aus diesen Modulen genutzt.

Aus dem Hilfsmodul MEMCOM werden folgende Routinen genutzt:
- INIT_TBL :Loeschen Fehlertabelle
- DELETE : - " - allgemeiner Merkzellen
- NMI_VEK_SAVE :Behandlung NMI-Vektor am Programmanfang
- NMI_VEK_RETURN : - " - am Programmende
- START_INIT :Start Einstellung
- DISPLAY_VERSION :Ausgabe der Versionsnummer
- TEST_PROC_LOC :Ermittlung des Programmsegmentes

Aus dem Hilfsmodul MEMLIM werden folgende Routinen genutzt:
- RANGE_MOV :Bereich auslagern

Aus dem Hilfsmodul MEMERR werden folgende Routinen genutzt:
- ERROR_MEM :allgemeines Fehlerbehandlungsprogramm
- ERROR_ZPS :spezielles Fehlerbehandlungsprogramm fuer ZPS
- NMI_OCCURRED :Behandlungsprogramm fuer Paritaetsbitfehler (NMI), die beim Programmlesen aufgetreten sind
- PARITY_OUT :Ausgabe der Paritaetsfehlerregister mit PBF
- SEARCH_PARITY_ERRORS :Suchprogramm fuer Paritaetsbitfehler

Aus LACS werden folgende Routinen genutzt:
- ASCII_CONV :Umwandlung Hex ---> ASCII
- TDMASKEDMESSAGE :von DEBUG abhaengige Zeichenausgabe
- TDDISPLAY :von DEBUG unabhaengige Zeichenausgabe

4. Ladeprozedur fuer autonomen Ablauf

Das Testprogramm wird im Verbund mit dem Leitprogramm LACS von Minidiskette gebootet (Kommandobeispiele siehe Pkt. 4.1. - 4.3.).

4.1. Laden mit Angabe des Geraetenamens

.B :Fn:TEST<CR> (n = Laufwerksnr.)
Nach Erreichen des Monitorbedienzustandes:
.B :Fn:G03<CR> (n = Laufwerksnr.)
Anschliessend verweilt das Programm im Generierhalt.

4.2. Laden ohne Angabe des Geraetenamens

.B TEST<CR>
Nach Erreichen des Monitorbedienzustandes:
.B G03<CR>
Bei letzterem Kommando werden TEST bzw. G03 auf allen Laufwerken gesucht, beginnend mit F0.
Anschliessend verweilt das Programm im Generierhalt.

4.3. Laden durch Kommandoeingabe im ACT (A 7100 - Confidence-Test)

Nach Netzeinschalten oder nach Druecken der Taste RESET besteht die verkuerzte Lademoeglichkeit von LACS durch Eingabe des Zeichens "T" innerhalb des PIC - Tests von ACT.
Nach Erreichen des Monitorbedienzustandes weiter wie oben.

5. Generierinformationen

Der Bediener hat die Moeglichkeit, die Ausgabe von Durchlauf- und Fehlermeldungen sowie allgemeine Mitteilungen ueber den Zustand des Prueflings durch Eingabe von Generierinformationen zu steuern. (siehe Punkt 5.1.). Dazu muss der Programmablauf unterbrochen werden (z.B. durch CTRL-C). Mittels des Monitorkommando-SW kann dann eine Aenderung der Generierung vorgenommen werden. Fuer spezielle Pruefzwecke koennen die verwendeten standardmaessigen Pruefmuster (MUSTER1 ... MUSTER4) im Datensegment DATA_MEMCOM veraendert werden (siehe MEMCOM-Beschreibung).

5.1. Generierung von Meldungen und Mitteilungen

TEST03 laeuft unter der Regie des Leitprogrammes LACS. Deshalb werden Meldungen ueber den Programmablauf durch die Variablen
TDERRONLY (Adr.: DATA_LACS:0000) und
TDDEBUG (Adr.: DATA_LACS:0002) gesteuert
(siehe Punkt 5.1. der Beschreibung des Leitprogrammes LACS).
Hinweise zur Interpretation der Fehlerausschriften sind in der MEMERR-Beschreibung Pkt. 8.2.4. enthalten.

5.2. Generierung spezieller Testablaeufe

Die Wirkung von DEBUG auf die Fehlerausschriften und den Fehlerhalt ist in der MEMERR-Beschreibung Pkt. 5.2. dargestellt. Der Pruefbereich ist durch das Pruefprogramm fest vorgegeben.

6. Startprozedur

6.1. Start des Testprogramms

Das Testprogramm TEST03 laeuft unter der Regie von LACS. Nach dem Laden (siehe Punkt 4.) erfolgt der Start:
- Aus dem Generierhalt durch das Monitorkommando
  .G<CR>
- Restart erfolgt durch das Monitorkommando
  .G <CODE_LACS>:40<CR>.

6.2. Bedieneraktionen

Der Bediener hat folgende Eingriffsmoeglichkeiten waehrend des Programmablaufs:
- CTRL-S : Unterbrechung der Ausgabe, Fortsetzung mit CTRL-Q.
- CTRL-Q : Fortsetzung einer unterbrochenen Ausgabe.
- CTRL-P : Hardcopy der Bildschirmausgabe auf angeschlossenen Drucker
  Erneutes CTRL-P setzt die Hardcopy zurueck.
- CTRL-C : Uebergang in den Bedienzustand des Monitors.

Die Wirkung dieser Kommandos erfolgt erst unmittelbar vor Ausgabe einer Meldung auf den Bildschirm.

7. Einstellungsmoeglichkeiten

Die Standardeinstellung fuer TEST03 lautet:
- TDERRONLY: 0 : siehe Punkt 5.1. LACS-Beschreibung
- TDDEBUG : 3 : _ " _

8. Programmbeschreibung

8.1. Programmablauf

Im 1. Durchlauf erfolgt vor dem eigentlichen Test eine von DEBUG abhaengige Ausgabe der Programmnummern, Modulnamen und der Zeichnungs- und Ausgabenummern aller beteiligten Module. Sie hat fuer TEST03 folgendes Aussehen (siehe Punkt 8.2.1.):

```
TEST03  yy-mm-dd  1.56.703004.1/67-aa-56ACnnn
MEMCOM  yy-mm-dd  1.56 703051.5/67-aa-56ACnnn
MEMLIM  yy-mm-dd  1.56.703052.3/67-aa-56ACnnn
MEMERR  yy-mm-dd  1.56.703053.1/67-aa-56ACnnn
```

dabei ist: yy-mm-dd: Datum der letzten Aenderung
aa: Ausgabenummer
56ACnnn: Nr. der Aenderungsmitteilung

Anschliessend beginnt die Abarbeitung des Testmoduls mit dem Einstellen der Anfangsbedingungen:
- Retten und Setzen der Interruptmaske fuer INT 1 (:= <BREAK>) (START_INIT)
  Im Verlaufe der Programmmabarbeitung wird vor der Auslagerung des Pruefbereiches der Interrupt mit dem Befehl CLI verboten und nach dem Ruecktransport des ausgelagerten Bereiches wieder mit STI erlaubt.
- Loeschen der Fehlertabelle (INIT_TBL)
- Loeschen der allgemeinen Merkzellen (DELETE)
- Ermittlung des Programmsegmentes (TEST_PROC_LOC)
- Ueberschreiben des NMI-Vektors (NMI_VEK_SAVE)
- Pruefbereich (0H ... 1FFFFH) auslagern nach save area: (20000H ... 3FFFFH) (RANGE_MOV)
- Umschaltung der Programmsteuerung in den ausgelagerten Be-

reich, wenn das Programm nach dem Booten im Bereich 3000H ... 1FFFFH stand.
Waehrend der Pruefung sind keine Ausgaben moeglich. Der Marching-Algorithmus beinhaltet folgende Phasen:

- B: Background:
  Speicher von der Anfangsadresse bis zur Endadresse mit dem MUSTER1 (A55A) wortweise beschreiben (Hintergrundmuster).
- R1: Read 1:
  Speicher von der Anfangsadresse bis zur Endadresse wortweise behandeln, dabei MUSTER1 (A55A) lesen, pruefen und sofort das invertierte Muster (/MUSTER1 := 5AA5) einschreiben.
- R2: Read 2:
  Wie R1, jedoch invertiertes Muster (/MUSTER1 := 5AA5) lesen, pruefen und sofort MUSTER2 (9494) einschreiben.
- R3: Read 3:
  Speicher von der Anfangsadresse bis zur Endadresse byteweise behandeln, dabei MUSTER2 (94) lesen, pruefen und sofort das invertierte Muster (/MUSTER2 := 6B) einschreiben.
- R4: Read 4:
  Wie R3, jedoch invertiertes Muster (/MUSTER2 := 6B) lesen, pruefen und sofort MUSTER1 byteweise (5A) als Speichermuster nach Testabschluss einschreiben.

Der Pruefalgorithmus wird jeweils fuer 3 Bereiche hintereinander ausgefuehrt:

10H ... 3FFH
402H ... FFFFH
10000H ... 1FFFFH

Die einzelnen Phasen (B, R1, R2, R3 und R4) werden immer ueber den gesamten Pruefbereich von 10H ... 1FFFFH zusammenhaengend abgearbeitet.
Es ergibt sich aber eine Sprungstelle des Pruefmusters an den Grenzen der o.g. 3 Unterbereiche.
Bei der Pruefung wird die Zelle mit der Wortadresse 400H ausgespart (Control-Byte ZPS).

Nach jedem Prueflesen einer Testadresse erfolgt eine Fehlerauswertung durch einen Soll-Ist-Datenvergleich und die Kontrolle der Paritaetsfehlermeldung. Paritaetsfehler (PBF) werden durch Setzen von 2 Marken (Speicherzelle, Register SI) registriert. Unmittelbar vor dem Prueflesen wird SI geloescht, so dass beim Abfragen beider Marken zwischen PBF durch Programmlesen und Prueflesen unterschieden werden kann.
PBF durch Programmlesen werden mit NMI_OCCURRED (MEMERR) behandelt. Treten Datenfehler bzw. PBF beim Prueflesen auf, wird der Fehler durch ERROR_MEM (MEMERR) behandelt, d.h. die Fehler werden gezaehlt, das Fehlerkennzeichen fuer LACS gesetzt und die Fehlertabelle aktualisiert.
Bei Paritaetsbitfehlern infolge Programmlesens werden selbstaendig weitere Versuche zur Fortsertzung des Tests ohne Ausgabe

einer Fehlermeldung ausgefuehrt (siehe MEMERR-Beschreibung Pkt. 8.2.4. Routine: NMI_OCCURRED)

Am Schluss des Programms wird der ausgelagerte Speicherbereich zurueckgespeichert, der Interrupt wieder erlaubt (STI) und wenn erforderlich die Programmsteuerung an diesen Bereich uebergeben. Damit werden wieder Fehlerausgaben ermoeglicht.
Durch Auslesen der Fehlertabelle erfolgt jetzt die DEBUG abhaengige Ausgabe der waehrend des Pruefens aufgetretenen Fehler. Der Ueberlauf der Fehlertabelle infolge zu vieler Lesefehler wird angezeigt (siehe MEMERR-Beschreibung Pkt. 8.2.4. Routine: ERROR_ZPS).
Der NMI-Vektor wird zurueckgeschrieben (NMI_VEK_RETURN), die Interruptmaske zurueckgestellet, auf Paritaetsbitfehler (PBF) (SEARCH_PARITY_ERRORS) und auf fehlerfreien Durchlauf kontrolliert (siehe Pkt. 10.2.).
Im Fehlerfall informiert das Testprogramm den Bediener durch Fehlerausschriften (beachte Punkt 5. und 9.) und geht im Standardfall durch einen INT 3 in den Monitorbedienzustand. Die Rueckkehr zum Leitprogramm erfolgt durch die Uebergabe der Fehlermeldung in AX und den Return-Befehl: (FAR-) RET.

fehlerfreier Durchlauf des Tests: AX := 1
fehlerhafter Durchlauf des Tests: AX := 0

Zum Abbruch des Testprogramms TEST03 siehe Punkt 10.

8.2. Anschlussbedingungen

8.2.1. Struktur des Quellprogramms

Der Modul TEST03 ist folgendermassen aufgebaut:
1. ¤TITLE(TEST03 Zeichnungsnummer-Ausgabenummer)
2. Programmbeschreibung nach dem Muster der Beschreibung des Leitprogrammes LACS.
3. NAME TEST03
4. Makrodefinitionen.
5. EXTRN-Erklaerungen
6. Datensegmente der Tests: TEST02 ... TEST07 und der Hilfsmodule MEMCOM, MEMLIM, MEMERR mit den Bezeichnungen:
   DATA_TEST02, DATA_TEST03, DATA_TEST04, DATA_TEST05,
   DATA_TEST06, DATA_TEST07, DATA_MEMCOM, DATA_MEMLIM,
   DATA_MEMERR, DATA_MEMINP,
   die zusammen die Gruppe: DGROUP_MEMORY bilden. Die Datensegmente (ausser DATA_TEST03) sind Pseudosegmente fuer die Ermoeglichung der Gruppenbildung. Sie enthalten entweder nichts oder die EXTRN-Erklaerungen der fuer TEST03 aus dem entsprechenden Modul benutzten Daten.
7. Codesegmente der Tests: TEST02 ... TEST07 und der Hilfsmodule MEMCOM, MEMLIM, MEMERR mit den Bezeichnungen:
   CODE_TEST02, CODE_TEST03, CODE_TEST04, CODE_TEST05,
   CODE_TEST06, CODE_TEST07, CODE_MEMCOM, CODE_MEMLIM,
   CODE_MEMERR,
   die zusammen die Gruppe: CGROUP_MEMORY bilden. Die Codesegmente (ausser CODE_TEST03) sind Pseudosegmente fuer die Ermoegli-

chung der Gruppenbildung. Sie enthalten entweder nichts oder die EXTRN-Erklaerungen der fuer TEST03 aus dem entsprechenden Modul benutzten Unterprogramme.

PUBLIC-Erklaerungen stehen in dem Segment, in dem die betreffenden Symbole vorkommen, ganz am Anfang des Segments. Im Segment CODE_TEST03 steht nach den PUBLIC-Erklaerungen folgende Zeile:

TEST03 yy-mm-dd 1.56.703004.1/67-aa-56ACnnn'

dabei ist: yy-mm-dd: Datum der letzten Aenderung
aa: Ausgabenummer
56ACnnn: Nr. der Aenderungsmitteilung

Dieser Text wird nach dem Start des Testmoduls in Abhaengigkeit von TDDEBUG (s. Pkt. 5.1.) ausgegeben.

8.2.2. Stack- und Registernutzung

Von TEST03 wird der Stack des Leitprogramms LACS genutzt, d.h. beim Start von TEST03 wird kein SS sondern nur DS initialisiert.

8.2.3. Protokollsteuerung

Die Ausgaben sind je nach den Werten, die die Variablen TDERRONLY und TDDEBUG enthalten, abgestuft unterdrueckbar (s. Pkt. 5.1. der MEMERR-Beschreibung). Es gelten die Richtlinien gemaess Punkt 8.2.3. der Beschreibung des Leitprogrammes LACS.

8.2.4. Nutzbare TEST03-Routinen

TEST03 enthaelt keine Routinen, die fuer andere Nutzer verfuegbar sind.

8.2.5. Nutzbarkeit anderer Routinen

Siehe Punkt 3.4.

9. Fehler

9.1. Fehlerausgaben

Bei fehlerhaften Testergebnissen wird in Abhaengigkeit von DEBUG (siehe Pkt.5.2. der MEMERR-Beschreibung) eine geeignete Meldung auf dem Bildschirm ausgegeben (siehe Beschreibungen der Unterprogramme im Pkt. 8.2.4. der Module MEMCOM, MEMLIM und MEMERR). Falls undefiniertes Verhalten oder 'Abstuerze' des Rechners den normalen Programmablauf verhindern oder falls keine Fehlerausschriften generiert sind, koennen die Fehlerinformation bei Spei-

cherlesefehlern der Fehlertabelle (siehe MEMERR-Beschreibung Pkt. 8.2.1.) entnommen werden. Bytedaten werden mit fuehrenden Nullen dargestellt.

Andere Fehlerausschriften beginnen mit einem eingerueckten Pfeil ===> und der Ausgabe eines Textes (siehe Beschreibungen der Unterprogramme im Pkt. 8.2.4. der Hilfsmodule MEMCOM, MEMLIM und MEMERR).
Dann wird in Abhaengigkeit von DEBUG durch einen INT 3 der Monitorbedienzustand erreicht (Standardgenerierung). Der Bediener kann mittels Monitorkommandos den Fehler eingrenzen, verdichten oder ausblenden (siehe Punkt 5.).

9.2. Fortsetzung nach Fehler

Nach der Fehlerbehandlung in "ERROR_MEM" bzw. "ERROR_ZPS" (siehe Pkt. 8.2.4. der MEMERR-Beschreibung) und der Ausgabe der Fehlermeldung z.B. in "RESULT_OUT" (siehe Pkt. 8.2.4. der MEMERR-Beschreibung) erfolgt Programmhalt im Falle der Standardgenerierung (s. Pkt. 5.2.).

```
TDERRONLY = 0
TDDEBUG   = 3
```

Die Fortsetzung der Fehlerausgabe aus der Fehlertabelle kann durch das Monitorkommando G<CR> erfolgen.

10. Verschiedenes

10.1. Programmabbruchmoeglichkeiten

Der Programmablauf kann durch die BREAK-Taste nicht unterbrochen werden (Interrupt durch CLI verboten), weil der Monitorarbeitsbereich wie Stack usw. ausgelagert und mit Pruefinformation ueberschrieben ist.
Ein geordneter Abbruch durch Eingabe von CTRL-C ist praktisch nur am Programmende vor der naechsten Ausgabe moeglich. Alle Interruptvektoren haben jetzt den fuer die Monitorbedienung notwendigen Wert.
Restart auf der Startadresse des Testmoduls: CODE_LACS:40 ist moeglich.

10.2. Kontrolle der Programmdurchfuehrung

Die einzelnen Arbeitsphasen des Pruefprogrammes (E R1, R2, R3 und R4) werden nicht angezeigt (siehe Pkt. 8.2.1.). Waehrend der Abarbeitung von TEST03 sind keine Ausgaben moeglich. Am Ende von TEST03 wird bei aufgetretenem Fehler (NUMBER_READ_ERROR # 0H) der im Programm intern gefuehrte Zykluszaehler und die Anzahl der Fehler im TEST03 ausgegeben (kumulativer Wert aus allen absolvierten Durchlaeufen von TEST03):

```
===> MEMORY MARCH TEST03 cycles:  xxxxH
     with memory errors:  xxxxH
```

Sind im aktuellen Durchlauf von TEST03 keine Fehler aufgetreten, wird die o.g. Meldung nicht ausgegeben.
Nach Abgabe der Steuerung durch den Testmodul an die Regie des Leitprogrammes LACS und die Uebergabe des fehlerfreien Pruefdurchlaufes wird durch die folgende LACS-Meldung die Durchfuehrung des Testprogramms auf dem Bildschirm bestaetigt (Standardfall):

```
TEST03: MEMORY MARCH TEST LOW RANGE                    "PASSED"
```

Fehler- und Durchlaufzaehler werden in folgender Form ausgegeben:

```
ERRORS / CYCLES:   xxxx / yyyy (xxxx,yyyy in dezimaler Form)
```

Danach beginnt ein neuer Programmzyklus.
Nach Unterbrechung des Programms kann die Durchfuehrungszahl des Testprogramms TEST03 auch aus der Zelle:

```
CONST_LACS:0036
```

ermittelt werden (Monitorkommando DW). Die Anzahl der an LACS uebergebenen Fehlermeldungen kann man der Zelle:

```
CONST_LACS:0034
```

entnehmen.
Als Fehler werden in LACS fehlerhafte Durchlaeufe des Testprogrammes gezaehlt (:= Ruecksprueng mit gesetztem Fehlerkennzeichen). Die echte Fehlerzahl wird durch das Testprogramm selbst ausgegeben oder ist der Zelle NUMBER_READ_ERROR_3 (MEMCOM) zu entnehmen.

1. TEST04, MEMORY REFRESH TEST HIGH RANGE

2. Funktion des Programms

Der "Memory Refresh Test High Range" prueft die dynamische Speicherfaehigkeit des Speichers im oberen Speicherbereich von einer Anfangsadresse (Standard: 2000:0000H) bis zur Endadresse einschliesslich der zugehoerigen Refresh - Logik. Die Endadresse wird im Standardfall automatisch ermittelt.
Der Refreshtest wird durch geeignete Wahl des Datenmusters einmal auf die Speicherschaltkreise fuer die Datenbits und zum anderen auf die fuer die Paritaetsbits angewendet.

3. Voraussetzungen

Im Folgenden werden Speicheradressen in der Form Segment:Offset hexadezimal angegeben, wobei das Segment durch einen symbolischen Namen bezeichnet wird. Die Zuordnung von symbolischem Segmentnamen und absoluten Adressen ist aus der MP2-Liste zu entnehmen.
In speziellen Faellen (z.B. Darstellung der Fehlerausschrift oder Bereichsangaben) wird die Speicheradresse absolut mit 5 hexadezimalen Zahlen (Kennzeichnung mit H) angegeben.

3.1. Geraeteausruestung

Es ist die Grundausruestung des Rechners A 7100 erforderlich. Die Ein/Ausgabe richtet sich nach der im Monitor generierten Geraetekonfiguration.

3.2. Speicherbedarf

ca. 1,3 KB fuer TEST04.OBJ
Die absoluten Adressen der durch den Locator zugewiesenen Speicherbereiche sind anhand der MP2-Liste zu berechnen.

3.3. Test zusaetzlicher Baugruppen

TEST04 erfordert u.a. eine in den wesentlichen Teilen funktionierende ZVE einschliesslich der Interruptlogik. Insbesondere wird in der ZVE die richtige Adressierung der Speicherzellen und der Datenaustausch ueber den Systembus vorausgesetzt.

3.4. Nutzung anderer Programmodule

Der Testmodul TEST04.OBJ muss mit den Hilfsmodulen MEMCOM.OBJ, MEMLIM.OBJ, MEMERR.OBJ, dem Leitprogrammodul LACS.OBJ und der Bibliothek LACS.LIB verbunden sein.
Es werden Unterprogramme (Routinen) aus diesen Modulen genutzt.

Aus dem Hilfsmodul MEMCOM werden folgende Routinen genutzt:

- INIT_TBL	:Loeschen Fehlertabelle
- DELETE	: - " - allgemeiner Merkzellen
- NMI_VEK_SAVE	:Behandlung NMI-Vektor am Programmanfang
- NMI_VEK_RETURN	: - " - am Programmende
- START_INIT	:Start Einstellung
- DISPLAY_VERSION	:Ausgabe der Versionsnummer
- TIME_5	:Zeitschleife von ca. 5 sec zur Realisierung der Wartezeit beim Refreshtest

Aus dem Hilfsmodul MEMLIM werden folgende Routinen genutzt:

- TEST_LIMIT_OPS :Behandlung der Speicherpruefgrenzen (RAM_RANGE, OPS_OR_ZPS, SCAN_END_OPS)

Aus dem Hilfsmodul MEMERR werden folgende Routinen genutzt:

- ERROR_MEM	:Fehlerbehandlungsprogramm
- NMI_OCCURRED	:Behandlungsprogramm fuer Paritaetsbitfehler (NMI), die beim Programmlesen aufgetreten sind
- SEARCH_PARITY_ERRORS	:Suchprogramm fuer Paritaetsbitfehler

Aus LACS werden folgende Routinen genutzt:

- ASCII_CONV	:Umwandlung Hex ——> ASCII
- TDMASKEDMESSAGE	:von DEBUG abhaengige Zeichenausgabe
- TDDISPLAY	:von DEBUG unabhaengige Zeichenausgabe

4. Ladeprozedur fuer autonomen Ablauf

Das Testprogramm wird im Verbund mit dem Leitprogramm LACS von Minidiskette gebootet (Kommandobeispiele siehe Pkt. 4.1. - 4.3.).

4.1. Laden mit Angabe des Geraetenamens

```
.B :Fn:TEST<CR>          (n = Laufwerksnr.)
```

Nach Erreichen des Monitorbedienzustandes:

```
.B :Fn:G04<CR>           (n = Laufwerksnr.)
```

Anschliessend verweilt das Programm im Generierhalt.

4.2. Laden ohne Angabe des Geraetenamens

```
.B TEST<CR>
```

Nach Erreichen des Monitorbedienzustandes:

```
.B G04<CR>
```

Bei letzterem Kommando werden TEST bzw. G04 auf allen Laufwerken gesucht, beginnend mit F0.
Anschliessend verweilt das Programm im Generierhalt.

4.3. Laden durch Kommandoeingabe im ACT (A 7100 - Confidence-Test)

Nach Netzeinschalten oder nach Druecken der Taste RESET besteht die verkuerzte Lademoeglichkeit von LACS durch Eingabe des Zeichens "T" innerhalb des PIC - Tests von ACT.

Nach Erreichen des Monitorbedienzustandes weiter wie oben.

5. Generierinformationen

Der Bediener hat die Moeglichkeit, die Ausgabe von Durchlauf- und Fehlermeldungen sowie allgemeine Mitteilungen ueber den Zustand des Prueflings durch Eingabe von Generierinformationen zu steuern. (siehe Punkt 5.1.). Dazu muss der Programmablauf unterbrochen werden (z.B. durch CTRL-C). Mittels des Monitorkommandos SW kann dann eine Aenderung der Generierung vorgenommen werden.
Fuer spezielle Pruefzwecke kann das verwendete standardmaessige Pruefmuster (MUSTER3) im Datensegment DATA_MEMCOM veraendert werden (siehe MEMCOM-Beschreibung).
Das in MEMLIM vorhandene Datensegment: DATA_MEMINP dient zur Eingabe der Bereichsgrenzen fuer die Pruefung des Speichers (siehe Pkt. 8.2.1. der MEMLIM-Beschreibung). Zur Fehlersuche kann der Pruefbereich damit eingeschraenkt werden.

5.1. Generierung von Meldungen und Mitteilungen

TEST04 laeuft unter der Regie des Leitprogrammes LACS. Deshalb werden Meldungen ueber den Programmablauf durch die Variablen
TDERRONLY (Adr.: DATA_LACS:0000) und
TDDEBUG (Adr.: DATA_LACS:0002) gesteuert
(siehe Punkt 5.1. der Beschreibung des Leitprogrammes LACS).
Hinweise zur Interpretation der Fehlerausschriften sind in der MEMERR-Beschreibung Pkt. 8.2.4. enthalten.

5.2. Generierung spezieller Testablaeufe

Die Wirkung von DEBUG auf die Fehlerausschriften und den Fehlerhalt ist in der MEMERR-Beschreibung Pkt. 5.2. dargestellt.
Die obere Grenze des zu pruefenden Speichers wird im Standardfall (V0 = 4) automatisch ermittelt (siehe MEMLIM-Beschreibung Pkt. 8.2.4.).
Fuer spezielle Zwecke koennen die Grenzen des Pruefbereiches im Monitorbedienzustand (z.B. im Generierhalt oder nach CTRL-C) ueber das Kommando SW im Segment DATA_MEMINP veraendert werden (siehe MEMLIM-Beschreibung Pkt. 8.2.1.).

6. Startprozedur

6.1. Start des Testprogramms

Das Testprogramm TEST04 laeuft unter der Regie von LACS. Nach dem Laden (siehe Punkt 4.) erfolgt der Start:
- Aus dem Generierhalt durch das Monitorkommando
  .G<CR>
- Restart erfolgt durch das Monitorkommando
  .G <CODE_LACS>:40<CR>.

6.2. Bedieneraktionen

Der Bediener hat folgende Eingriffsmoeglichkeiten waehrend des Programmablaufs:

- CTRL-S : Unterbrechung der Ausgabe, Fortsetzung mit CTRL-Q.
- CTRL-Q : Fortsetzung einer unterbrochenen Ausgabe.
- CTRL-P : Hardcopy der Bildschirmausgabe auf angeschlossenen Drucker
  Erneutes CTRL-P setzt die Hardcopy zurueck.
- CTRL-C : Uebergang in den Bedienzustand des Monitors.

Die Wirkung dieser Kommandos erfolgt erst unmittelbar vor Ausgabe einer Meldung auf den Bildschirm.
Der Uebergang in den Bedienzustand des Monitors ist auch moeglich durch Druecken der Taste BREAK (:= INT 1) auf der Tastatur, wobei Pkt. 10.1. zu beachten ist. Damit ist dem Bediener ein Mittel gegeben, einen Fehler im Programmablauf von TEST04 zu erzeugen. Durch Umschreiben der Istinformation in einer Speicherzelle kann z.B. ein Lesefehler provoziert werden, so dass eine Fehlerausschrift erfolgt.

7. Einstellungsmoeglichkeiten

Die Standardeinstellung fuer TEST04 lautet:

- TDERRONLY:	0	: siehe Punkt 5.1. LACS-Beschreibung
- TDDEBUG :	3	: - " -
- V0 :	4	: Pruefbereich: 2000:0000H bis zur automatisch ermittelten oberen Speichergrenze (siehe MEMLIM-Beschreibung)

8. Programmbeschreibung

8.1. Programmablauf

Im 1. Durchlauf erfolgt vor dem eigentlichen Test eine von DEBUG abhaengige Ausgabe der Programmnummern, Modulnamen und der Zeichnungs- und Ausgabenummern aller beteiligten Module. Sie hat fuer TEST04 folgendes Aussehen (siehe Punkt 8.2.1.):

```
TEST04  yy-mm-dd  1.56.703005.8/67-aa-56ACnnn
MEMCOM  yy-mm-dd  1.56.703051.5/67-aa-56ACnnn
MEMLIM  yy-mm-dd  1.56.703052.3/67-aa-56ACnnn
MEMERR  yy-mm-dd  1.56.703053.1/67-aa-56ACnnn
```

dabei ist:
- yy-mm-dd: Datum der letzten Aenderung
- aa: Ausgabenummer
- 56ACnnn: Nr. der Aenderungsmitteilung

Anschliessend beginnt die Abarbeitung des Testmoduls mit dem Einstellen der Anfangsbedingungen:

- Retten und Setzen der Interruptmaske fuer INT 1 (:= <BREAK>) (START_INIT)
- Loeschen der Fehlertabelle (INIT_TBL)

- Loeschen der allgemeinen Merkzellen (DELETE)
- Einstellen der Pruefgrenzen (TEST_LIMIT_OPS)
- Ueberschreiben des NMI-Vektors (NMI_VEK_SAVE)

Werden in den Unterprogrammen Fehler gefunden (z.B. falsch eingegebene Pruefgrenzen), wird der Monitorbedienzustand eingenommen und der Bediener zur Korrektur aufgefordert. Danach kann mit G<CR> weitergestartet werden.
Der Pruefbereich des Speichers wird viermal beschrieben und nach einer Wartezeit gelesen. Die Phasen eines solchen Zyklus werden durch die Ausgabe von Kurzzeichen in Abhaengigkeit von DEBUG angezeigt. (siehe auch Pkt. 10.2.):

Zyklus I:
- B: Background:
  Speicher von der Anfangsadresse bis zur Endadresse mit dem Schiebemuster (MUSTER3: 0001H) wortweise beschreiben (durchlaufende "1" ab Bit 0, d.h.: 0001H, 0002H, 0004H usw. auf fortlaufenden Wortadressen).
  Die Speicherschaltkreise fuer die Paritaetsbits werden dabei abwechselnd auf jeweils 8 aufeinanderfolgenden Adressen mit "0" bzw. "1" belegt.
- W: Wait:
  Nach dem Beschreiben unterbleiben fuer eine Wartezeit von ca. 5 sec jegliche Zugriffe zum Pruefbereich des Speichers.
- R: Read:
  Danach wird der Pruefbereich gelesen und der Dateninhalt geprueft.

Zyklus II:
Der Zyklus I wird mit dem invertierten Schiebemuster wiederholt (durchlaufende "0" ab Bit 0, d.h.: FFFEH, FFFDH, FFFBH usw. auf fortlaufenden Wortadressen).

Zyklus III:
Wiederholung des Zyklus I, wobei das Schiebemuster (MUSTER3) ab Bit 8 beginnt (0100H) (durchlaufende "1" ab Bit 8, d.h.: 0100H, 0200H, 0400H usw. auf fortlaufenden Wortadressen).
Dadurch wird in den Speicherschaltkreisen fuer die Paritaetsbits das zu Zyklus I invertierte Muster realisiert.

Zyklus IV:
Wiederholung von Zyklus III mit dem dazu invertierten Schiebemuster (durchlaufende "0" ab Bit 8, d.h.: FEFFH, FDFFH, FBFFH usw. auf fortlaufenden Wortadressen).

Sind bei wortweiser Abarbeitung des Algorithmus die Pruefgrenzen keine Wortgrenzen (ungerade untere Pruefgrenze und/oder gerade obere Pruefgrenze), werden am Anfang und/oder am Ende des Pruefbereiches Bytezyklen durchgefuehrt. Das ist z.B. bei der Auslistung des Speichers mit dem Monitorkommando DW oder DB zu beachten. Das Pruefmuster wird so eingeschrieben, als ob es sich in dem nicht vorhandenem bzw. nicht geprueftem Byte fortsetzen wuerde.
Nach jedem Prueflesen einer Testadresse erfolgt eine Fehlerauswertung durch einen Soll-Ist-Datenvergleich und die Kontrolle der Paritaetsfehlermeldung. Paritaetsfehler (PBF) werden durch Setzen von 2 Marken (Speicherzelle, Register SI) registriert. Unmittel-

bar vor dem Prueflesen wird SI geloescht, so dass beim Abfragen beider Marken zwischen PBF durch Programmlesen und Prueflesen unterschieden werden kann. PBF durch Programmlesen werden mit NMI_OCCURRED (MEMERR) behandelt.
Treten Datenfehler bzw. PBF beim Prueflesen auf, wird der Fehler durch ERROR_MEM (MEMERR) behandelt, d.h. die Fehler werden gezaehlt,das Fehlerkennzeichen fuer LACS gesetzt, die Fehlertabelle aktualisiert, die fehlerhaften Daten- bzw. Paritaetsbits gekennzeichnet (FAILED_BIT, FAILED_PARITY) und eine von DEBUG abhaengige Fehlermeldung ausgegeben (sieh MEMERR-Beschreibung).
Am Schluss des Programms wird der NMI-Vektor zurueckgeschrieben (NMI_VEK_RETURN), die Interruptmaske zurueckgestellt, auf Paritaetsbitfehler (PBF) (SEARCH_PARITY_ERRORS) und auf fehlerfreien Durchlauf kontrolliert (siehe Pkt. 10.2.).
Im Fehlerfall informiert das Testprogramm den Bediener durch Fehlerausschriften (beachte Punkt 5. und 9.) und geht im Standardfall durch einen INT 3 in den Monitorbedienzustand.
Die Rueckkehr zum Leitprogramm erfolgt durch die Uebergabe der Fehlermeldung in AX und den Return-Befehl: (FAR-) RET.

fehlerfreier Durchlauf des Tests: AX := 1
fehlerhafter Durchlauf des Tests: AX := 0

Zum Abbruch des Testprogramms TEST04 siehe Punkt 10.

8.2. Anschlussbedingungen

8.2.1. Struktur des Quellprogramms

Der Modul TEST04 ist folgendermassen aufgebaut:

1. $TITLE(TEST04 Zeichnungsnummer-Ausgabenummer)
2. Programmbeschreibung nach dem Muster der Beschreibung des Leitprogrammes LACS.
3. NAME TEST04
4. Makrodefinitionen.
5. EXTRN-Erklaerungen
6. Datensegmente der Tests: TEST02 ... TEST07 und der Hilfsmodule MEMCOM, MEMLIM, MEMERR mit den Bezeichnungen:
   DATA_TEST02, DATA_TEST03, DATA_TEST04, DATA_TEST05,
   DATA_TEST06, DATA_TEST07, DATA_MEMCOM, DATA_MEMLIM,
   DATA_MEMERR, DATA_MEMINP,
   die zusammen die Gruppe: DGROUP_MEMORY bilden. Die Datensegmente (ausser DATA_TEST04) sind Pseudosegmente fuer die Ermoeglichung der Gruppenbildung. Sie enthalten entweder nichts oder die EXTRN-Erklaerungen der fuer TEST04 aus dem entsprechenden Modul benutzten Daten.
7. Codesegmente der Tests: TEST02 ... TEST07 und der Hilfsmodule MEMCOM, MEMLIM, MEMERR mit den Bezeichnungen:
   CODE_TEST02, CODE_TEST03, CODE_TEST04, CODE_TEST05,
   CODE_TEST06, CODE_TEST07, CODE_MEMCOM, CODE_MEMLIM,
   CODE_MEMERR,
   die zusammen die Gruppe: CGROUP_MEMORY bilden. Die Codesegmente (ausser CODE_TEST04) sind Pseudosegmente fuer die Ermoeglichung der Gruppenbildung. Sie enthalten entweder nichts oder die EXTRN-Erklaerungen der fuer TEST04 aus dem entsprechenden

Modul benutzten Unterprogramme.

PUBLIC-Erklaerungen stehen in dem Segment, in dem die betreffenden Symbole vorkommen, ganz am Anfang des Segments. Im Segment CODE_TEST04 steht nach den PUBLIC-Erklaerungen folgende Zeile:

TEST04 yy-mm-dd 1.56.703005.8/67-aa-56ACnnn'

da' ' ist: yy-mm-dd: Datum der letzten Aenderung
aa: Ausgabenummer
56ACnnn: Nr. der Aenderungsmitteilung

Dieser Text wird nach dem Start des Testmoduls in Abhaengigkeit von TDDEBUG (s. Pkt. 5.1.) ausgegeben.

8.2.2. Stack- und Registernutzung

Von TEST04 wird der Stack des Leitprogramms LACS genutzt, d.h. beim Start von TEST04 wird kein SS sondern nur DS initialisiert.

8.2.3. Protokollsteuerung

Die Ausgaben sind je nach den Werten, die die Variablen TDERRONLY und TDDEBUG enthalten, abgestuft unterdrueckbar (s. Pkt. 5.1. der MEMERR-Beschreibung). Es gelten die Richtlinien gemaess Punkt 8.2.3. der Beschreibung des Leitprogrammes LACS.

8.2.4. Nutzbare TEST04-Routinen

TEST04 enthaelt keine Routinen, die fuer andere Nutzer verfuegbar sind.

8.2.5. Nutzbarkeit anderer Routinen

Siehe Punkt 3.4.

9. Fehler

9.1. Fehlerausgaben

Bei fehlerhaften Testergebnissen wird in Abhaengigkeit von DEBUG (siehe Pkt.5.2. der MEMERR-Beschreibung) eine geeignete Meldung auf dem Bildschirm in dem Unterprogramm (Routine) ausgegeben, in dem der Fehler festgestellt wurde (siehe Beschreibungen der Unterprogramme im Pkt. 8.2.4. der Hilfsmodule MEMCOM, MEMLIM und MEMERR).
Falls undefiniertes Verhalten oder 'Abstuerze' des Rechners den normalen Programmablauf verhindern oder falls keine Fehlerausschriften generiert sind, koennen die Fehlerinformation bei Spei-

cherlesefehlern der Fehlertabelle (siehe MEMERR-Beschreibung Pkt. 8.2.1.) oder der letzte Fehler folgenden Zellen entnommen werden:

- Istwert : AX(AL)
- Sollwert : DX(DL)
- Offset der Testadresse: BX
- Segment - " - : ES
- Begleitwort : REC_DATA
- NMI : SI (0: kein NMI, OFFH: NMI aufgetreten)

Bytedaten werden mit fuehrenden Nullen dargestellt.Sie stehen im LOW-Teil der Register AX und DX.

Andere Fehlerausschriften beginnen mit einem eingerueckten Pfeil ===> und der Ausgabe eines Textes (siehe Beschreibungen der Unterprogramme im Pkt. 8.2.4. der Hilfsmodule MEMCOM, MEMLIM und MEMERR).
Dann wird in Abhaengigkeit von DEBUG durch einen INT 3 der Monitorbedienzustand erreicht (Standardgenerierung). Der Bediener kann mittels Monitorkommandos den Fehler eingrenzen, verdichten oder ausblenden (siehe Punkt 5.).

9.2. Fortsetzung nach Fehler

Nach der Fehlerbehandlung in "ERROR_MEM" (siehe Pkt. 8.2.4. der MEMERR-Beschreibung) und der Ausgabe der Fehlermeldung z.B. in "RESULT_OUT" (siehe Pkt. 8.2.4. der MEMERR-Beschreibung) erfolgt Programmhalt im Falle der Standardgenerierung (s. Pkt. 5.2.).

TDERRONLY = 0
TDDEBUG = 3

Die Fortsetzung des Tests kann durch das Monitorkommando G<CR> erfolgen.

10. Verschiedenes

10.1. Programmabbruchmoeglichkeiten

Der Programmablauf kann durch die BREAK-Taste bedingt unterbrochen werden (siehe Punkt 6.2.). Dabei ist zu beachten, dass der NMI-Vektor fuer die Behandlug von Paritaetsbitfehlern im Testprogramm ueberschrieben ist (siehe MEMERR-Beschreibung). Weiterstart mit G<CR> ist erlaubt.
Ein geordneter Abbruch kann durch Eingabe von CTRL-C erfolgen, damit erfolgt der Uebergang in den Monitorbedienzustand vor der naechsten Ausgabe. Alle Interruptvektoren haben jetzt den fuer die Monitorbedienung notendigen Wert.
Restart auf der Startadresse des Testmoduls: CODE_LACS:40 ist moeglich.

10.2. Kontrolle der Programmdurchfuehrung

Die einzelnen Arbeitsphasen des Pruefprogrammes werden durch Kurzzeichen angezeigt (siehe Pkt. 8.2.1.):

```
...>B
...>B...>W
...>B...>W...>R
...>B...>W...>R...>B
...>B...>W...>R...>B...>W
...>B...>W...>R...>B...>W...>R
...>B...>W...>R...>B...>W...>R...>B
...>B...>W...>R...>B...>W...>R...>B...>W
...>B...>W...>R...>B...>W...>R...>B...>W...>R
...>B...>W...>R...>B...>W...>R...>B...>W...>R...>B
...>B...>W...>R...>B...>W...>R...>B...>W...>R...>B...>W
...>B...>W...>R...>B...>W...>R...>B...>W...>R...>B...>W...>R
```

Am Ende von TEST04 wird bei aufgetretenem Fehler (NUMBER_READ_ERROR # 0H) der im Programm intern gefuehrte Zykluszaehler und die Anzahl der Fehler im TEST04 ausgegeben (kumulativer Wert aus allen absolvierten Durchlaeufen von TEST04):

```
===> MEMORY REFRESH TEST04 cycles:  xxxxH
     with memory errors:  xxxxH
```

Sind im aktuellen Durchlauf von TEST04 keine Fehler aufgetreten, wird die o.g. Meldung nicht ausgegeben.
Nach Abgabe der Steuerung durch den Testmodul an die Regie des Leitprogrammes LACS und die Uebergabe des fehlerfreien Pruefdurchlaufes wird durch die folgende LACS-Meldung die Durchfuehrung des Testprogramms auf dem Bildschirm bestaetigt (Standardfall):

```
TEST04: MEMORY REFRESH TEST HIGH RANGE                    "PASSED"
```

Fehler- und Durchlaufzaehler werden in folgender Form ausgegeben:

```
ERRORS / CYCLES:   xxxx / yyyy (xxxx,yyyy in dezimaler Form)
```

Danach beginnt ein neuer Programmzyklus.
Nach Unterbrechung des Programms kann die Durchfuehrungszahl des Testprogramms TEST04 auch aus der Zelle:

```
CONST_LACS:0044
```

ermittelt werden (Monitorkommando DW). Die Anzahl der an LACS uebergebenen Fehlermeldungen kann man der Zelle:

```
CONST_LACS:0042
```

entnehmen.
Als Fehler werden in LACS fehlerhafte Durchlaeufe des Testprogrammes gezaehlt (:= Ruecksprueng mit gesetztem Fehlerkennzeichen). Die echte Fehlerzahl wird durch das Testprogramm selbst ausgegeben oder ist der Zelle NUMBER_READ_ERROR_4 (MEMCOM) zu entnehmen.

1. TEST05, MEMORY REFRESH TEST LOW RANGE

2. Funktion des Programms

Der "Memory Refresh Test Low Range" prueft die dynamische Speicherfaehigkeit des Speichers im unteren Speicherbereich von der Anfangsadresse 0:0010H bis zur Endadresse 1000:FFFFH einschliesslich der zugehoerigen Refresh - Logik.
Der Refreshtest wird durch geeignete Wahl des Datenmusters einmal auf die Speicherschaltkreise fuer die Datenbits und zum anderen auf die fuer die Paritaetsbits angewendet.
Am Beginn der Pruefung wird der vorhandene Speicherinhalt gerettet und am Programmende wieder zurueckgeschrieben.

3. Voraussetzungen

Im Folgenden werden Speicheradressen in der Form Segment:Offset hexadezimal angegeben, wobei das Segment absolut oder durch einen symbolischen Namen bezeichnet sein kann. Die Zuordnung von symbolischem Segmentnamen und absoluten Adressen ist aus der MP2-Liste zu entnehmen.
In speziellen Faellen (z.B. Darstellung der Fehlerausschrift oder Bereichsangaben) wird die Speicheradresse absolut mit 5 hexadezimalen Zahlen (Kennzeichnung mit H) angegeben.

3.1. Geraeteausruestung

Es ist die Grundausruestung des Rechners A 7100 erforderlich. Die Ein/Ausgabe richtet sich nach der im Monitor generierten Geraetekonfiguration.

3.2. Speicherbedarf

ca. 1,0 KB fuer TEST05.OBJ
Die absoluten Adressen der durch den Locator zugewiesenen Speicherbereiche sind anhand der MP2-Liste zu berechnen.

3.3. Test zusaetzlicher Baugruppen

TEST05 erfordert u.a. eine in den wesentlichen Teilen funktionierende ZVE einschliesslich der Interruptlogik. Insbesondere wird in der ZVE die richtige Adressierung der Speicherzellen und der Datenaustausch ueber den Systembus bzw. internen X2-Bus vorausgesetzt.

3.4. Nutzung anderer Programmodule

Der Testmodul TEST05.OBJ muss mit den Hilfsmodulen MEMCOM.OBJ, MEMLIM.OBJ, MEMERR.OBJ, dem Leitprogrammodul LACS.OBJ und der Bibliothek LACS.LIB verbunden sein.
Es werden Unterprogramme (Routinen) aus diesen Modulen genutzt.
Aus dem Hilfsmodul MEMCOM werden folgende Routinen genutzt:

- INIT_TBL :Loeschen Fehlertabelle
- DELETE : - " - allgemeiner Merkzellen
- NMI_VEK_SAVE :Behandlung NMI-Vektor am Programmanfang
- NMI_VEK_RETURN : - " - am Programmende
- START_INIT :Start Einstellung
- DISPLAY_VERSION :Ausgabe der Versionsnummer
- TEST_PROC_LOC :Ermittlung des Programmsegmentes
- TIME_5 :Zeitschleife von ca. 5 sec zur Realisierung der Wartezeit beim Refreshtest

Aus dem Hilfsmodul MEMLIM werden folgende Routinen genutzt:

- RANGE_MOV :Bereich auslagern

Aus dem Hilfsmodul MEMERR werden folgende Routinen genutzt:

- ERROR_MEM :allgemeines Fehlerbehandlungsprogramm
- ERROR_ZPS :spezielles Fehlerbehandlungsprogramm fuer ZPS
- NMI_OCCURRED :Behandlungsprogramm fuer Paritaetsbitfehler (NMI), die beim Programmlesen aufgetreten sind
- PARITY_OUT :Ausgabe der Paritaetsfehlerregister mit PBF
- SEARCH_PARITY_ERRORS :Suchprogramm fuer Paritaetsbitfehler

Aus LACS werden folgende Routinen genutzt:

- ASCII_CONV :Umwandlung Hex ——> ASCII
- TDMASKEDMESSAGE :von DEBUG abhaengige Zeichenausgabe
- TDDISPLAY :von DEBUG unabhaengige Zeichenausgabe

4. Ladeprozedur fuer autonomen Ablauf

Das Testprogramm wird im Verbund mit dem Leitprogramm LACS von Minidiskette gebootet (Kommandobeispiele siehe Pkt. 4.1. - 4.3.).

4.1. Laden mit Angabe des Geraetenamens

```
.B :Fn:TEST<CR>          (n = Laufwerksnr.)
```
Nach Erreichen des Monitorbedienzustandes:
```
.B :Fn:G05<CR>           (n = Laufwerksnr.)
```
Anschliessend verweilt das Programm im Generierhalt.

4.2. Laden ohne Angabe des Geraetenamens

```
.B TEST<CR>
```
Nach Erreichen des Monitorbedienzustandes:
```
.B G05<CR>
```
Bei letzterem Kommando werden TEST bzw. G05 auf allen Laufwerken

gesucht, beginnend mit F0.
Anschliessend verweilt das Programm im Generierhalt.

4.3. Laden durch Kommandoeingabe im ACT (A 7100 - Confidence-Test)

Nach Netzeinschalten oder nach Druecken der Taste RESET besteht die verkuerzte Lademoeglichkeit von LACS durch Eingabe des Zeichens "T" innerhalb des PIC - Tests von ACT.
Nach Erreichen des Monitorbedienzustandes weiter wie oben.

5. Generierinformationen

Der Bediener hat die Moeglichkeit, die Ausgabe von Durchlauf- und Fehlermeldungen sowie allgemeine Mitteilungen ueber den Zustand des Prueflings durch Eingabe von Generierinformationen zu steuern. (siehe Punkt 5.1.). Dazu muss der Programmablauf unterbrochen werden (z.B. durch CTRL-C). Mittels des Monitorkommandos SW kann dann eine Aenderung der Generierung vorgenommen werden. Fuer spezielle Pruefzwecke kann das verwendete standardmaessige Pruefmuster (MUSTER3) im Datensegment DATA_MEMCOM veraendert werden (siehe MEMCOM-Beschreibung).

5.1. Generierung von Meldungen und Mitteilungen

TEST05 laeuft unter der Regie des Leitprogrammes LACS. Deshalb werden Meldungen ueber den Programmablauf durch die Variablen
TDERRONLY (Adr.: DATA_LACS:0000) und
TDDEBUG (Adr.: DATA_LACS:0002) gesteuert
(siehe Punkt 5.1. der Beschreibung des Leitprogrammes LACS).
Hinweise zur Interpretation der Fehlerausschriften sind in der MEMERR-Beschreibung Pkt. 8.2.4. enthalten.

5.2. Generierung spezieller Testablaeufe

Die Wirkung von DEBUG auf die Fehlerausschriften und den Fehlerhalt ist in der MEMERR-Beschreibung Pkt. 5.2. dargestellt.
Der Pruefbereich ist durch das Pruefprogramm fest vorgegeben.

6. Startprozedur

6.1. Start des Testprogramms

Das Testprogramm TEST05 laeuft unter der Regie von LACS. Nach dem Laden (siehe Punkt 4.) erfolgt der Start:
- Aus dem Generierhalt durch das Monitorkommando
  .G<CR>
- Restart erfolgt durch das Monitorkommando
  .G <CODE_LACS>:40<CR>

6.2. Bedieneraktionen

Der Bediener hat folgende Eingriffsmoeglichkeiten waehrend des Programmablaufs:
- CTRL-S : Unterbrechung der Ausgabe, Fortsetzung mit CTRL-Q.
- CTRL-Q : Fortsetzung einer unterbrochenen Ausgabe.
- CTRL-P : Hardcopy der Bildschirmausgabe auf angeschlossenen Drucker
  Erneutes CTRL-P setzt die Hardcopy zurueck.
- CTRL-C : Uebergang in den Bedienzustand des Monitors.

Die Wirkung dieser Kommandos erfolgt erst unmittelbar vor Ausgabe einer Meldung auf den Bildschirm.

7. Einstellungsmoeglichkeiten

Die Standardeinstellung fuer TEST05 lautet:
- TDERRONLY: 0 : siehe Punkt 5.1. LACS-Beschreibung
- TDDEBUG : 3 : _ " _

8. Programmbeschreibung

8.1. Programmablauf

Im 1. Durchlauf erfolgt vor dem eigentlichen Test eine von DEBUG abhaengige Ausgabe der Programmnummern, Modulnamen und der Zeichnungs- und Ausgabenummern aller beteiligten Module. Sie hat fuer TEST05 folgendes Aussehen (siehe Punkt 8.2.1.):

```
TEST05  yy-mm-dd  1.56.703006.6/67-aa-56ACnnn
MEMCOM  yy-mm-dd  1.56.703051.5/67-aa-56ACnnn
MEMLIM  yy-mm-dd  1.56.703052.3/67-aa-56ACnnn
MEMERR  yy-mm-dd  1.56.703053.1/67-aa-56ACnnn
```

dabei ist:
- yy-mm-dd: Datum der letzten Aenderung
- aa: Ausgabenummer
- 56ACnnn: Nr. der Aenderungsmitteilung

Anschliessend beginnt die Abarbeitung des Testmoduls mit dem Einstellen der Anfangsbedingungen:
- Retten und Setzen der Interruptmaske fuer INT 1 (:= <BREAK>) (START_INIT)
  Im Verlaufe der Programmmabarbeitung wird vor der Auslagerung des Pruefbereiches der Interrupt mit dem Befehl CLI verboten und nach dem Ruecktransport des ausgelagerten Bereiches wieder mit STI erlaubt.
- Loeschen der Fehlertabelle (INIT_TBL)
- Loeschen der allgemeinen Merkzellen (DELETE)
- Ermittlung des Programmsegmentes (TEST_PROC_LOC)
- Ueberschreiben des NMI-Vektors (NMI_VEK_SAVE)
- Pruefbereich (0H ... 1FFFFH) auslagern nach save area: (20000H ... 3FFFFH) (RANGE_MOV)
- Umschaltung der Programmsteuerung in den ausgelagerten Be-

reich, wenn das Programm nach dem Booten im Bereich 3000H ... 1FFFFH stand.
Waehrend der Pruefung sind keine Ausgaben moeglich.
Der Pruefbereich des Speichers wird viermal beschrieben und nach einer Wartezeit gelesen. Die Phasen eines solchen Zyklus werden nachfolgend beschrieben:

Zyklus I:

- B: Background:
  Speicher von der Anfangsadresse bis zur Endadresse mit dem Schiebemuster (MUSTER3: 0001H) wortweise beschreiben (durchlaufende "1" ab Bit 0, d.h.: 0001H, 0002H, 0004H usw. auf fortlaufenden Wortadressen).
  Die Speicherschaltkreise fuer die Paritaetsbits werden dabei abwechselnd auf jeweils 8 aufeinanderfolgenden Adressen mit "0" bzw. "1" belegt.
- W: Wait:
  Nach dem Beschreiben unterbleiben fuer eine Wartezeit von ca. 5 sec jegliche Zugriffe zum Pruefbereich des Speichers.
- R: Read:
  Danach wird der Pruefbereich gelesen und der Dateninhalt geprueft.

Zyklus II:
Der Zyklus I wird mit dem invertierten Schiebemuster wiederholt (durchlaufende "0" ab Bit 0, d.h.: FFFEH, FFFDH, FFFBH usw. auf fortlaufenden Wortadressen).

Zyklus III:
Wiederholung des Zyklus I, wobei das Schiebemuster (MUSTER3) ab Bit 8 beginnt (0100H) (durchlaufende "1" ab Bit 8, d.h.: 0100H, 0200H, 0400H usw. auf fortlaufenden Wortadressen).
Dadurch wird in den Speicherschaltkreisen fuer die Paritaetsbits das zu Zyklus I invertierte Muster realisiert.

Zyklus IV:
Wiederholung von Zyklus III mit dem dazu invertierten Schiebemuster (durchlaufende "0" ab Bit 8, d.h.: FEFFH, FDFFH, FBFFH usw. auf fortlaufenden Wortadressen).

Der Pruefalgorithmus wird jeweils fuer 3 Bereiche hintereinander ausgefuehrt:

```
   10H ...   3FFH
  402H ...  FFFFH
10000H ... 1FFFFH
```

Die einzelnen Phasen (B, W und R) werden immer ueber den gesamten Pruefbereich von 10H ... 1FFFFH zusammenhaengend abgearbeitet. Es ergibt sich aber eine Sprungstelle des Pruefmusters an den Grenzen der o.g. 3 Unterbereiche.
Bei der Pruefung wird die Zelle mit der Wortadresse 400H ausgespart (Control-Byte ZPS).

Nach jedem Prueflesen einer Testadresse erfolgt eine Fehlerauswertung durch einen Soll-Ist-Datenvergleich und die Kontrolle der Paritaetsfehlermeldung.

Paritaetsfehler (PBF) werden durch Setzen von 2 Marken (Speicherzelle, Register SI) registriert. Unmittelbar vor dem Prueflesen wird SI geloescht, so dass beim Abfragen beider Marken zwischen PBF durch Programmlesen und Prueflesen unterschieden werden kann. PBF durch Programmlesen werden mit NMI_OCCURRED (MEMERR) behandelt. Treten Datenfehler bzw. PBF beim Prueflesen auf, wird der Fehler durch ERROR_MEM (MEMERR) behandelt, d.h. die Fehler werden gezaehlt,das Fehlerkennzeichen fuer LACS gesetzt und die Fehlertabelle aktualisiert.
Bei Paritaetsbitfehlern infolge Programmlesens werden selbstaendig weitere Versuche zur Fortsertzung des Tests ohne Ausgabe einer Fehlermeldung ausgefuehrt (siehe MEMERR-Beschreibung Pkt. 8.2.4. Routine: NMI_OCCURRED)
Am Schluss des Programms wird der ausgelagerte Speicherbereich zurueckgespeichert, der Interrupt wieder erlaubt (STI) und wenn erforderlich die Programmsteuerung an diesen Bereich uebergeben. Damit werden wieder Fehlerausgaben ermoeglicht.
Durch Auslesen der Fehlertabelle erfolgt jetzt die DEBUG abhaengige Ausgabe der waehrend des Pruefens aufgetretenen Fehler. Der Ueberlauf der Fehlertabelle infolge zu vieler Lesefehler wird angezeigt (siehe MEMERR-Beschreibung Pkt. 8.2.4. Routine: ERROR_ZPS).
Der NMI-Vektor wird zurueckgeschrieben (NMI_VEK_RETURN), die Interruptmaske zurueckgestellet, auf Paritaetsbitfehler (PBF) (SEARCH_PARITY_ERRORS) und auf fehlerfreien Durchlauf kontrolliert (siehe Pkt. 10.2.).
Im Fehlerfall informiert das Testprogramm den Bediener durch Fehlerausschriften (beachte Punkt 5. und 9.) und geht im Standardfall durch einen INT 3 in den Monitorbedienzustand.
Die Rueckkehr zum Leitprogramm erfolgt durch die Uebergabe der Fehlermeldung in AX und den Return-Befehl: (FAR-) RET.

fehlerfreier Durchlauf des Tests: AX := 1
fehlerhafter Durchlauf des Tests: AX := 0

Zum Abbruch des Testprogramms TEST05 siehe Punkt 10.

8.2. Anschlussbedingungen

8.2.1. Struktur des Quellprogramms

Der Modul TEST05 ist folgendermassen aufgebaut:
1. ¤TITLE(TEST05 Zeichnungsnummer-Ausgabenummer)
2. Programmbeschreibung nach dem Muster der Beschreibung des Leitprogrammes LACS.
3. NAME TEST05
4. Makrodefinitionen.
5. EXTRN-Erklaerungen
6. Datensegmente der Tests: TEST02 ... TEST07 und der Hilfsmodule MEMCOM, MEMLIM, MEMERR mit den Bezeichnungen:
   DATA_TEST02, DATA_TEST03, DATA_TEST04, DATA_TEST05,
   DATA_TEST06, DATA_TEST07, DATA_MEMCOM, DATA_MEMLIM,
   DATA_MEMERR, DATA_MEMINP,
   die zusammen die Gruppe: DGROUP_MEMORY bilden. Die Datensegmente (ausser DATA_TEST05) sind Pseudosegmente fuer die Er-

moeglichung der Gruppenbildung. Sie enthalten entweder nichts oder die EXTRN-Erklaerungen der fuer TEST05 aus dem entsprechenden Modul benutzten Daten.

7. Codesegmente der Tests: TEST02 ... TEST07 und der Hilfsmodule MEMCOM, MEMLIM, MEMERR mit den Bezeichnungen:
   CODE_TEST02, CODE_TEST03, CODE_TEST04, CODE_TEST05,
   CODE_TEST06, CODE_TEST07, CODE_MEMCOM, CODE_MEMLIM,
   CODE_MEMERR,
   die zusammen die Gruppe: CGROUP_MEMORY bilden. Die Codesegmente (ausser CODE_TEST05) sind Pseudosegmente fuer die Ermoeglichung der Gruppenbildung. Sie enthalten entweder nichts oder die EXTRN-Erklaerungen der fuer TEST05 aus dem entsprechenden Modul benutzten Unterprogramme.

PUBLIC-Erklaerungen stehen in dem Segment, in dem die betreffenden Symbole vorkommen, ganz am Anfang des Segments.
Im Segment CODE_TEST05 steht nach den PUBLIC-Erklaerungen folgende Zeile:

TEST05 yy-mm-dd 1.56.703006.6/67-aa-56ACnnn'

dabei ist: yy-mm-dd: Datum der letzten Aenderung
aa: Ausgabenummer
56ACnnn: Nr. der Aenderungsmitteilung

Dieser Text wird nach dem Start des Testmoduls in Abhaengigkeit von TDDEBUG (s. Pkt. 5.1.) ausgegeben.

8.2.2. Stack- und Registernutzung

Von TEST05 wird der Stack des Leitprogramms LACS genutzt, d.h. beim Start von TEST05 wird kein SS sondern nur DS initialisiert.

8.2.3. Protokollsteuerung

Die Ausgaben sind je nach den Werten, die die Variablen TDERRONLY und TDDEBUG enthalten, abgestuft unterdrueckbar (s. Pkt. 5.1. der MEMERR-Beschreibung). Es gelten die Richtlinien gemaess Punkt 8.2.3. der Beschreibung des Leitprogrammes LACS.

8.2.4. Nutzbare TEST05-Routinen

TEST05 enthaelt keine Routinen, die fuer andere Nutzer verfuegbar sind.

8.2.5. Nutzbarkeit anderer Routinen

Siehe Punkt 3.4.

9. Fehler

9.1. Fehlerausgaben

Bei fehlerhaften Testergebnissen wird in Abhaengigkeit von DEBUG (siehe Pkt.5.2. der MEMERR-Beschreibung) eine geeignete Meldung auf dem Bildschirm ausgegeben (siehe Beschreibungen der Unterprogramme im Pkt. 8.2.4. der Module MEMCOM, MEMLIM und MEMERR).
Falls undefiniertes Verhalten oder 'Abstuerze' des Rechners den normalen Programmablauf verhindern oder falls keine Fehlerausschriften generiert sind, koennen die Fehlerinformation bei Speicherlesefehlern der Fehlertabelle (siehe MEMERR-Beschreibung Pkt. 8.2.1.) entnommen werden.
Bytedaten werden mit fuehrenden Nullen dargestellt.

Andere Fehlerausschriften beginnen mit einem eingerueckten Pfeil ===> und der Ausgabe eines Textes (siehe Beschreibungen der Unterprogramme im Pkt. 8.2.4. der Module MEMCOM, MEMLIM und MEMERR).
Dann wird in Abhaengigkeit von DEBUG durch einen INT 3 der Monitorbedienzustand erreicht (Standardgenerierung). Der Bediener kann mittels Monitorkommandos den Fehler eingrenzen, verdichten oder ausblenden (siehe Punkt 5.).

9.2. Fortsetzung nach Fehler

Nach der Fehlerbehandlung in "ERROR_MEM" bzw. "ERROR_ZPS" (siehe Pkt. 8.2.4. der MEMERR-Beschreibung) und der Ausgabe der Fehlermeldung z.B. in "RESULT_OUT" (siehe Pkt. 8.2.4. der MEMERR-Beschreibung) erfolgt Programmhalt im Falle der Standardgenerierung (s. Pkt. 5.2.).

```
TDERRONLY = 0
TDDEBUG   = 3
```

Die Fortsetzung der Fehlerausgabe aus der Fehlertabelle kann durch das Monitorkommando G<CR> erfolgen.

10. Verschiedenes

10.1. Programmabbruchmoeglichkeiten

Der Programmablauf kann durch die BREAK-Taste nicht unterbrochen werden (Interrupt durch CLI verboten), weil der Monitorarbeitsbereich wie Stack usw. ausgelagert und mit Pruefinformation ueberschrieben ist.
Ein geordneter Abbruch durch Eingabe von CTRL-C ist praktisch nur am Programmende vor der naechsten Ausgabe moeglich. Alle Interruptvektoren haben jetzt den fuer die Monitorbedienung notwendigen Wert.
Restart auf der Startadresse des Testmoduls: CODE_LACS:40 ist moeglich.

10.2. Kontrolle der Programmdurchfuehrung

Die einzelnen Arbeitsphasen des Pruefprogrammes (B, W und R) werden nicht angezeigt (siehe Pkt. 8.2.1.). Waehrend der Abarbeitung von TEST05 sind keine Ausgaben moeglich. Am Ende von TEST05 wird bei aufgetretenem Fehler (NUMBER_READ_ERROR # 0H) der im Programm intern gefuehrte Zykluszaehler und die Anzahl der Fehler im TEST05 ausgegeben (kumulativer Wert aus allen absolvierten Durchlaeufen von TEST05):

```
===> MEMORY REFRESH TEST05 cycles:  xxxxH
     with memory errors:  xxxxH
```

Sind im aktuellen Durchlauf von TEST05 keine Fehler aufgetreten, wird die o.g. Meldung nicht ausgegeben.
Nach Abgabe der Steuerung durch den Testmodul an die Regie des Leitprogrammes LACS und die Uebergabe des fehlerfreien Pruefdurchlaufes wird durch die folgende LACS-Meldung die Durchfuehrung des Testprogramms auf dem Bildschirm bestaetigt (Standardfall):

```
TEST05: MEMORY REFRESH TEST LOW RANGE                "PASSED"
```

Fehler- und Durchlaufzaehler werden in folgender Form ausgegeben:

```
ERRORS / CYCLES:   xxxx / yyyy (xxxx,yyyy in dezimaler Form)
```

Danach beginnt ein neuer Programmzyklus.
Nach Unterbrechung des Programms kann die Durchfuehrungszahl des Testprogramms TEST05 auch aus der Zelle:

```
CONST_LACS:0052
```

ermittelt werden (Monitorkommando DW). Die Anzahl der an LACS uebergebenen Fehlermeldungen kann man der Zelle:

```
CONST_LACS:0050
```

entnehmen.
Als Fehler werden in LACS fehlerhafte Durchlaeufe des Testprogrammes gezaehlt (:= Ruecksprueenge mit gesetztem Fehlerkennzeichen). Die echte Fehlerzahl wird durch das Testprogramm selbst ausgegeben oder ist der Zelle NUMBER_READ_ERROR_5 (MEMCOM) zu entnehmen.

1. TEST06, OPS PARITY LOGIC TEST

2. Funktion des Programms

Der "OPS Parity Logic Test" prueft die Funktion der Logik fuer die Paritaetsbitpruefung des OPS. Es werden gezielt Paritaetsbitfehler erzeugt und die Reaktion ueberprueft. Dazu werden 2 Testadressen benutzt, die an der 128 KB-Grenze eines jeden OPS liegen, d.h. die letzte Wortadresse im 1. 128 KB-Bereich und die 1. Wortadresse im 2. 128 KB-Bereich. Jeder der beiden Bereiche ist einer Bank (0X oder 1X) zugeordnet.
Da hierbei die absolute Adresse eingeht, ist die Zuordnung vom Vorhandensein eines ZPS unabhaengig.

Zuordnung der OPS zu den Adressenbereichen:

mit ZPS		ohne ZPS	
ZPS	(0H .... 1FFFFH)	OPS_1	(0H ..... 3FFFFH)
OPS_1	(20000H .... 5FFFFH)	OPS_2	(40000H .... 7FFFFH)
OPS_2	(60000H .... 9FFFFH)	OPS_3	(80000H .... BFFFFH)
OPS_3	(A0000H .... DFFFFH)	OPS_4	(C0000H .... F7FFFH)

Bank 0X: 0H ... 1FFFFH, 40000H ... 5FFFFH,
80000H ... 9FFFFH, C0000H ... DFFFFH

Bank 1X: 20000H ... 3FFFFH, 60000H ... 7FFFFH,
A0000H ... BFFFFH, E0000H ... F7FFFH

(F8000H ... FFFFFH := ROM-Bereich des Monitors)

Im OPS uebernimmt ein Steuerbit (Bit 0 im dem Byte, das ueber das Paritaetsbitfehlerregister des OPS auf den moeglichen I/O-Adressen: 0H, 2H, 40H oder 42H adressierbar ist) die Steuerung der Paritaetskontrolle und der Fehlerbewertung. Das Steuerbit ist nicht lesbar.
Das auf o.g. Adressen lesbare Byte kennzeichnet den Paritaetsbitfehler und hat folgenden Aufbau:

Bedeutung der einzelnen Bits:

```
0000 1111 : 0FH := fehlerfreier Zustand
^^^^ ^^^^
|||| ||||
|||| |||0 : Fehler im geraden Byte
|||| |||1 : kein Fehler im geraden Byte
|||| ||0  : Fehler im ungeraden Byte
|||| ||1  : kein Fehler im ungeraden Byte
|||| |0   : Fehler in Bank 1X
|||| |1   : Fehler in Bank 0X
|||| 1    : hat immer den Wert "1"
0000      : haben immer den Wert "0"
```

Ein Paritaetsbitfehler im OPS wird in der ZVE als NMI wirksam. Die NMI-Fehlermeldung wird durch das Testprogramm im Register SI

registriert. Die LED-Anzeige wird nicht ausgewertet.

3. Voraussetzungen

Im Folgenden werden Speicheradressen in der Form Segment:Offset hexadezimal angegeben, wobei das Segment absolut oder durch einen symbolischen Namen bezeichnet sein kann. Die Zuordnung von symbolischem Segmentnamen und absoluten Adressen ist aus der MP2-Liste zu entnehmen.
In speziellen Faellen (z.B. Darstellung der Fehlerausschrift oder Bereichsangaben) wird die Speicheradresse absolut mit max. 5 hexadezimalen Zahlen (Kennzeichnung mit H) angegeben.

3.1. Geraeteausruestung

Es ist die Grundausruestung des Rechners A 7100 erforderlich. Die Ein/Ausgabe richtet sich nach der im Monitor generierten Geraetekonfiguration.

3.2. Speicherbedarf

ca. 10,5 KB fuer TEST06.OBJ
Die absoluten Adressen der durch den Locator zugewiesenen Speicherbereiche sind anhand der MP2-Liste zu berechnen.

3.3. Test zusaetzlicher Baugruppen

TEST06 erfordert u.a. eine in den wesentlichen Teilen funktionierende ZVE einschliesslich der NMI Behandlungslogik.

3.4. Nutzung anderer Programmodule

Der Testmodul TEST06.OBJ muss mit den Hilfsmodulen MEMCOM.OBJ, MEMLIM.OBJ, MEMERR.OBJ, dem Leitprogrammodul LACS.OBJ und der Bibliothek LACS.LIB verbunden sein.
Es werden Unterprogramme (Routinen) aus diesen Modulen genutzt.
Aus dem Hilfsmodul MEMCOM werden folgende Routinen genutzt:

- START_INIT :Start Einstellung
- NMI_VEK_RETURN :Behandlung NMI-Vektor am Programmende
- DISPLAY_VERSION :Ausgabe der Versionsnummer
- TEST_PROC_LOC :Ermittlung des Programmsegmentes

Aus dem Hilfsmodul MEMLIM werden folgende Routinen genutzt:

- TEST_LIMIT_OPS :Behandlung der Speicherpruefgrenzen (RAM_RANGE, OPS_OR_ZPS, SCAN_END_OPS)

Aus dem Hilfsmodul MEMERR werden folgende Routinen genutzt:

- NMI_OCCURRED :Behandlungsprogramm fuer Paritaetsbitfehler (NMI), die beim Programmlesen aufgetreten sind
- SEARCH_PARITY_ERRORS :Suchprogramm fuer Paritaetsbitfehler

Aus LACS werden folgende Routinen genutzt:
- ASCII_CONV :Umwandlung Hex ---> ASCII
- TDMASKEDMESSAGE :von DEBUG abhaengige Zeichenausgabe
- TDDISPLAY :von DEBUG unabhaengige Zeichenausgabe

4. Ladeprozedur fuer autonomen Ablauf

Das Testprogramm wird im Verbund mit dem Leitprogramm LACS von Minidiskette gebootet (Kommandobeispiele siehe Pkt. 4.1. - 4.3.).

4.1. Laden mit Angabe des Geraetenamens

.B :Fn:TEST<CR> (n = Laufwerksnr.)
Nach Erreichen des Monitorbedienzustandes:
.B :Fn:G06<CR> (n = Laufwerksnr.)
Anschliessend verweilt das Programm im Generierhalt.

4.2. Laden ohne Angabe des Geraetenamens

.B TEST<CR>
Nach Erreichen des Monitorbedienzustandes:
.B G06<CR>
Bei letzterem Kommando werden TEST bzw. G06 auf allen Laufwerken gesucht, beginnend mit F0. Anschliessend verweilt das Programm im Generierhalt.

4.3. Laden durch Kommandoeingabe im ACT (A 7100 - Confidence-Test)

Nach Netzeinschalten oder nach Druecken der Taste RESET besteht die verkuerzte Lademoeglichkeit von LACS durch Eingabe des Zeichens "T" innerhalb des PIC - Tests von ACT.
Nach Erreichen des Monitorbedienzustandes weiter wie oben.

5. Generierinformationen

Der Bediener hat die Moeglichkeit, die Ausgabe von Durchlauf- und Fehlermeldungen sowie allgemeine Mitteilungen ueber den Zustand des Prueflings durch Eingabe von Generierinformationen zu steuern (siehe Punkt 5.1.). Dazu muss der Programmablauf unterbrochen werden (z.B. durch CTRL-C). Mittels des Monitorkommandos SW kann dann eine Aenderung der Generierung vorgenommen werden.
Fuer spezielle Pruefzwecke koennen die verwendeten standardmaessigen Pruefmuster (MUSTER1 und MUSTER4) im Datensegment DATA_MEMCOM veraendert werden (siehe MEMCOM-Beschreibung).
Das in MEMLIM vorhandene Datensegment: DATA_MEMINP dient zur Eingabe der Bereichsgrenzen fuer die Pruefung des Speichers (siehe Pkt. 8.2.1. der MEMLIM-Beschreibung). Zur Fehlersuche kann der Pruefbereich damit eingeschraenkt werden.

5.1. Generierung von Meldungen und Mitteilungen

TEST06 laeuft unter der Regie des Leitprogrammes LACS. Deshalb werden Meldungen ueber den Programmablauf durch die Variablen
TDERRONLY (Adr.: DATA_LACS:0000) und
TDDEBUG (Adr.: DATA_LACS:0002) gesteuert
(siehe Punkt 5.1. der Beschreibung des Leitprogrammes LACS).

5.2. Generierung spezieller Testablaeufe

Die Wirkung von DEBUG auf die Fehlerausschriften und den Fehlerhalt ist in der MEMERR-Beschreibung Pkt. 5.2. dargestellt.
Die obere Grenze des zu pruefenden Speichers wird im Standardfall (V0 = 4) automatisch ermittelt (siehe MEMLIM-Beschreibung Pkt. 8.2.4.).
Fuer spezielle Zwecke koennen die Grenzen des Pruefbereiches im Monitorbedienzustand (z.B. im Generierhalt oder nach CTRL-C) ueber das Kommando SW im Segment DATA_MEMINP veraendert werden (siehe MEMLIM-Beschreibung Pkt. 8.2.1.).

6. Startprozedur

6.1. Start des Testprogramms

Das Testprogramm TEST06 laeuft unter der Regie von LACS. Nach dem Laden (siehe Punkt 4.) erfolgt der Start:
- Aus dem Generierhalt durch das Monitorkommando
  .G<CR>
- Restart erfolgt durch das Monitorkommando
  .G <CODE_LACS>:40<CR>.

6.2. Bedieneraktionen

Der Bediener hat folgende Eingriffsmoeglichkeiten waehrend des Programmablaufs:
- CTRL-S : Unterbrechung der Ausgabe, Fortsetzung mit CTRL-Q.
- CTRL-Q : Fortsetzung einer unterbrochenen Ausgabe.
- CTRL-P : Hardcopy der Bildschirmausgabe auf angeschlossenen Drucker
  Erneutes CTRL-P setzt die Hardcopy zurueck.
- CTRL-C : Uebergang in den Bedienzustand des Monitors.

Die Wirkung dieser Kommandos erfolgt erst unmittelbar vor Ausgabe einer Meldung auf den Bildschirm.
Der Uebergang in den Bedienzustand des Monitors ist auch moeglich durch Druecken der Taste BREAK (:= INT 1) auf der Tastatur, wobei Pkt. 10.1. zu beachten ist. Damit ist dem Bediener ein Mittel gegeben, einen Fehler im Programmablauf von TEST06 zu erzeugen.

7. Einstellungsmoeglichkeiten

Die Standardeinstellung fuer TEST06 lautet:

- TDERRONLY:	0	: siehe Punkt 5.1. LACS-Beschreibung
- TDDEBUG :	3	: - " -
- V0 :	4	: Pruefbereich: 2000:0000H bis zur automatisch ermittelten oberen Speichergrenze (siehe MEMLIM-Beschreibung)

8. Programmbeschreibung

8.1. Programmablauf

Im 1. Durchlauf erfolgt vor dem eigentlichen Test eine von DEBUG abhaengige Ausgabe der Programmnummern, Modulnamen und der Zeichnungs- und Ausgabenummern aller beteiligten Module. Sie hat fuer TEST06 folgendes Aussehen (siehe Punkt 8.2.1.):

```
TEST06  yy-mm-dd  1.56.703007.4/67-aa-56ACnnn
MEMCOM  yy-mm-dd  1.56.703051.5/67-aa-56ACnnn
MEMLIM  yy-mm-dd  1.56.703052.3/67-aa-56ACnnn
MEMERR  yy-mm-dd  1.56.703053.1/67-aa-56ACnnn
```

dabei ist: yy-mm-dd: Datum der letzten Aenderung
aa: Ausgabenummer
56ACnnn: Nr. der Aenderungsmitteilung

Anschliessend beginnt die Abarbeitung des Testmoduls mit dem Einstellen der Anfangsbedingungen:
- Retten und Setzen der Interruptmaske fuer INT 1 (:= <BREAK>)
- Loeschen des Fehlerzaehlers und der allgemeinen Merkzellen

Mit "OPS_OR_ZPS" wird die Konfiguration im Speicherbereich 0H ... 1FFFFH und mit "RAM_RANGE" die obere Speichergrenze abgefragt. Danach wird die Konfiguration des gesamten Speichers ermittelt und entsprechende Bits in einer Merkzelle (CONFIG_RAM) mit der Adresse: DATA_TEST06:003B gesetzt.
Aufbau CONFIG_RAM (Bedeutung der einzelnen Bits):

```
     0000 0000  0000 0000
                     ^ ^^^^
                     | ||||
                     | |||ZPS      :  1H
                     | ||1.OPS     :  2H
                     | |2.OPS      :  4H
                     | 3.OPS       :  8H
                     4.OPS         : 10H
```

Fuer die Ausgabe der Speicherzuordnung zu den Paritaetsbitfehlerregistern werden zunaechst die Speichergrenzen unter Beruecksichtigung der eingegebenen Pruefgrenzen im Eingabesegment DATA_MEMINP (MEMLIM) in den Ausgabetext eingefuegt.
Fuer die NMI Behandlungsroutine wird der vorhandene NMI-Vektor gegen einen speziellen ausgetauscht. Am Programmende wird der

urspruengliche NMI-Vektor wieder zurueckgeschrieben.
Die eigentliche Pruefung beginnt mit dem Vergleich der Anzahl der konfigurierten OPS und der Anzahl antwortender Paritaetsbitfehlerregister. Bei der Produktionspruefung (Muttermaschinenpruefung) (siehe MEMLIN-Beschreibung Pkt. 8.2.4.) wird dieser Test nicht ausgefuehrt und eine entsprechende Mitteilung ausgegeben:

```
===> Test incompletly:
     number of OPS not compared with number of
     PARITY_ERROR_REGISTERS
     (procedure located for manufacturing test)
```

Als Naechstes wird die Zuordnung der OPS zu den Paritaetbitfehlerregistern ermittelt. Dazu wird auf ausgewaehlten Testadressen eine im Vergleich zum uebrigen Speicher paritaetsbitfalsche Information eingeschrieben. Durch systematisches Lesen dieser Testadressen wird festgestellt, in welchem Paritaetsbitfehlerregister der Paritaetsfehler (PBF) registriert ist.
Die Zuordnung des die jeweilige Testadresse beinhaltenden OPS zu dem registrierenden Register wird in Merkzellen fuer die weitere Verwendung im Test eingetragen. Nach dem Zuordnungstest wird die Paritaet der Information in den Testadressen wieder dem uebrigen Speicher angeglichen.
Jetzt erfolgt die von DEBUG abhaengige Ausgabe der ermittelten Zuordnung von OPS und Paritaetsbitfehlerregistern:

```
MEMORY CONNECTIONS:
ZPS    (     0H .... 1FFFFH)
OPS_1 (xxxxxH .... xxxxxH):  xxH
OPS_2 (xxxxxH .... xxxxxH):  xxH
OPS_3 (xxxxxH .... xxxxxH):  xxH
OPS_4 (xxxxxH .... xxxxxH):  xxH
```

Die richtige Funktion der Paritaetslogik bezueglich des Testmusters (geradzahlige/ungeradzahlige Anzahl "1"), des Testwortes (Low-, High-Byte oder Wort) und der ueberprueften Paritaet (Ergaenzung auf geradzahlige/ungeradzahlige Anzahl "1") wird anschliessend getestet. Dazu werden 2 Testadressen benutzt, die an der 128 KB-Grenze des jeweiligen OPS liegen, d.h. die letzte Wortadresse im 1. 128 KB-Bereich und die 1. Wortadresse im 2. 128 KB-Bereich.

```
Testadresse 1: (20000H + ADR_DISPLACEMENT) modulo 40000H
               (256 KB)
Testadresse 2: (20000H - 2H + ADR_DISPLACEMENT) modulo 40000H
               (256 KB)
```

Dabei gilt fuer die Adressenverschiebung der Testadresse (ADR_DISPLACEMENT) folgende Zuordnung:

	ADR_DISPLACEMENT	Bank
Testadr.1:	20000H (mit ZPS)	0X
	0H (ohne ZPS)	1X
Testadr.2:	20000H (mit ZPS)	1X
	0H (ohne ZPS)	0X

Vor Beginn der Fehlerprovozierung durch zum uebrigen Speicher paritaetsbitfalsche Information in den Testzellen, wird ein normaler Leseversuch (wortweise) auf den Testzellen mit geradzahliger und ungeradzahliger Anzahl "1" bei sowohl normaler (ungerader) als auch gedrehter (gerader) Paritaet durchgefuehrt. Das entspricht der normalen Arbeitsweise des Speichers. Dabei darf kein Paritaetsbitfehler entstehen.
Der Test mit gerader Testparitaet d.h. mit zum uebrigen Speicher entgegengesetzter Paritaet erfolgt pro OPS in der Reihenfolge:

- Geradzahliges (EVEN) Muster (MUSTER1 := A55AH) fuer
  - Low-Byte auf
    - Testadresse 1
    - Testadresse 2
  - High-Byte auf
    - Testadresse 1
    - Testadresse 2
  - Wort (Low- und High-Byte) auf
    - Testadresse 1
    - Testadresse 2
- Ungeradzahliges (ODD) Muster (MUSTER4 := 4AA4H) fuer
  - Low-Byte auf
    - Testadresse 1
    - Testadresse 2
  - High-Byte auf
    - Testadresse 1
    - Testadresse 2
  - Wort (Low- und High-Byte) auf
    - Testadresse 1
    - Testadresse 2

Anschliessend erfolgt der Test mit ungerader Testparitaet.
Fuer die Provozierung von Paritaetsbitfehlern muss die Paritaet der Testzelle entgegengesetzt zum uebrigen Speicher sein. Damit beim Programmlesen nicht ungewollt PBF entstehen, muss die urspruenglich mit ungerader Paritaet eingeschriebene Information im Programmspeicher gedreht werden. Dazu wird das schon erwaehnte Steuerbit (Pkt. 2.) geaendert (gerade Paritaet := Steuerbit = 0), die NMI-Weiterleitung in der ZVE verboten, der Programmspeicher pro Zelle ausgelesen und wieder zurueckgeschrieben. Entstandene PBF werden anschliessend geloescht.
Die Pruefung mit ungerader Testparitaet umfasst pro OPS folgende Routinen:

- Geradzahliges (EVEN) Muster (MUSTER1 := A55AH) fuer
  - Wort (Low- und High-Byte) auf
    - Testadresse 1
    - Testadresse 2
- Ungeradzahliges (ODD) Muster (MUSTER4 := 4AA4H) fuer
  - Wort (Low- und High-Byte) auf
    - Testadresse 1
    - Testadresse 2

Fuer jeden provozierten Paritaetsbitfehler wird folgendes getestet:

- Loeschen des Paritaetsbitfehlerregisters
- Wirksamkeit des NMI-Verbotes
  pro Testadresse
  - Entstehen des Paritaetsbitfehlers
  - Entstehen des NMI
  - Zuordnung zum richtigen Byte
  - Zuordnung zur richtigen Bank

Am Programmende von TEST06 wird die Information im Programmspeicher auf die bereits geschilderte Weise wieder mit der standardmaessigen ungeraden Paritaet versehen.
Der NMI-Vektor wird zurueckgeschrieben (NMI_VEK_RETURN), die Interruptmaske zurueckgestellt, auf Paritaetsbitfehler (PBF) (SEARCH_PARITY_ERRORS) und auf fehlerfreien Durchlauf kontrolliert (siehe Pkt. 10.2.).
Im Falle einer Fehlfunktion nach einem provozierten Paritaetsbitfehler informiert das Testprogramm den Bediener durch Fehlerausschriften (beachte Punkt 5. und 9.) und geht im Standardfall durch einen INT 3 in den Monitorbedienzustand.
Die Rueckkehr zum Leitprogramm erfolgt durch die Uebergabe der Fehlermeldung in AX und den Return-Befehl: (FAR-) RET.

fehlerfreier Durchlauf des Tests:	AX := 1
fehlerhafter Durchlauf des Tests:	AX := 0

Zum Abbruch des Testprogramms TEST06 siehe Punkt 10

8.2. Anschlussbedingungen

8.2.1. Struktur des Quellprogramms

Der Modul TEST06 ist folgendermassen aufgebaut:
1. %TITLE(TEST06 Zeichnungsnummer-Ausgabenummer)
2. Programmbeschreibung nach dem Muster der Beschreibung des Leitprogrammes LACS.
3. NAME TEST06
4. Makrodefinitionen.
5. EXTRN-Erklaerungen
6. Datensegmente der Tests: TEST02 ... TEST07 und der Hilfsmodule MEMCOM, MEMLIM, MEMERR mit den Bezeichnungen:
   DATA_TEST02, DATA_TEST03, DATA_TEST04, DATA_TEST05,
   DATA_TEST06, DATA_TEST07, DATA_MEMCOM, DATA_MEMLIM,
   DATA_MEMERR, DATA_MEMINP,

die zusammen die Gruppe: DGROUP_MEMORY bilden. Die Datensegmente (ausser DATA_TEST06) sind Pseudosegmente fuer die Ermoeglichung der Gruppenbildung. Sie enthalten entweder nichts oder die EXTRN-Erklaerungen der fuer TEST06 aus dem entsprechenden Modul benutzten Daten.

7. Codesegmente der Tests: TEST02 ... TEST07 und der Hilfsmodule MEMCOM, MEMLIM, MEMERR mit den Bezeichnungen:
   CODE_TEST02, CODE_TEST03, CODE_TEST04, CODE_TEST05,
   CODE_TEST06, CODE_TEST07, CODE_MEMCOM, CODE_MEMLIM,
   CODE_MEMERR,
   die zusammen die Gruppe: CGROUP_MEMORY bilden. Die Codesegmente (ausser CODE_TEST06) sind Pseudosegmente fuer die Ermoeglichung der Gruppenbildung. Sie enthalten entweder nichts oder die EXTRN-Erklaerungen der fuer TEST06 aus dem entsprechenden Modul benutzten Unterprogramme.

PUBLIC-Erklaerungen stehen in dem Segment, in dem die betreffenden Symbole vorkommen, ganz am Anfang des Segments. Im Segment CODE_TEST06 steht nach den PUBLIC-Erklaerungen folgende Zeile:

TEST06 yy-mm-dd 1.56.703007.4/67-aa-56ACnnn'

dabei ist: yy-mm-dd: Datum der letzten Aenderung
aa: Ausgabenummer
56ACnnn: Nr. der Aenderungsmitteilung

Dieser Text wird nach dem Start des Testmoduls in Abhaengigkeit von TDDEBUG (s. Pkt. 5.1.) ausgegeben.

8.2.2. Stack- und Registernutzung

Von TEST06 wird der Stack des Leitprogramms LACS genutzt, d.h. beim Start von TEST06 wird kein SS sondern nur DS initialisiert.

8.2.3. Protokollsteuerung

Die Ausgaben sind je nach den Werten, die die Variablen TDERRONLY und TDDEBUG enthalten, abgestuft unterdrueckbar (s. Pkt. 5.1. der MEMERR-Beschreibung). Es gelten die Richtlinien gemaess Punkt 8.2.3. der Beschreibung des Leitprogrammes LACS.

8.2.4. Nutzbare TEST06-Routinen

TEST06 enthaelt keine Routinen, die fuer andere Nutzer verfuegbar sind.

8.2.5. Nutzbarkeit anderer Routinen

Siehe Punkt 3.4.

9. Fehler

9.1. Fehlerausgaben

Bei fehlerhaften Testergebnissen wird in Abhaengigkeit von DEBUG (siehe Pkt. 5.2. der MEMERR-Beschreibung) eine geeignete Meldung auf dem Bildschirm ausgegeben.
Die Fehlerangabe nach einem provozierten Paritaetsbitfehler enthaelt in einem Kopf folgende Informationen:

- zum getesten OPS (OPS_1 ... OPS_4) einschliesslich der zugehoerigen Paritaetsfehlerregisteradresse
- zur Testparitaet (ODD (Control_Bit := 1)/EVEN (Control_Bit := 0))
- zum Testmuster (ODD (4AA4H)/EVEN (A55AH))
- zur Bank (0/1) in der die Testadresse liegt.

```
OPS_1 (xxxxxH .... xxxxxH):  xxH
OPS_2 (xxxxxH .... xxxxxH):  xxH
OPS_3 (xxxxxH .... xxxxxH):  xxH
OPS_4 (xxxxxH .... xxxxxH):  xxH
Testparity ..............:  ODD   (Control_Bit = 1)
Testparity ..............:  EVEN  (Control_Bit = 0)
Testpattern .............:  ODD   (4AA4H)
Testpattern .............:  EVEN  (A55AH)
Bank ....................:  0
Bank ....................:  1
```

Danach folgt eine Textausgabe, die zusammen mit der obigen Angabe die aufgetretenen Fehlfunktionen charakterisiert.
Folgende Textausgaben sind moeglich:

```
ERRO_1 :
   ===> number of PARITY_ERROR_REGISTERS greater as number of
        OPS

ERRO_2 :
   ===> number of PARITY_ERROR_REGISTERS lower as number of OPS

ERRO_3 :    ===> no PARITY_ERROR_REGISTER allocated
ERRO_4 :    ===> multiple PARITY_ERROR_REGISTER allocated
ERRO_5 :    ===> PARITY ERROR detected but NMI not occurred
ERRO_6 :    ===> neither PARITY ERROR nor NMI occurred
ERRO_7 :    ===> NMI occurred but no PARITY ERROR detectable

ERRO_8 :
   ===> test for creating NMI impossible because NMI occurred
        in memory
        (:= PBF (NMI) beim Programmlesen)

ERRO_9 :    ===> PARITY ERROR not occurred for LOW  BYTE
ERRO_10:    ===> PARITY ERROR not occurred for HIGH BYTE

ERRO_11:
   ===> PARITY ERROR of LOW BYTE occurred as error of HIGH BYTE
```

```
ERRO_12:
   ===> PARITY ERROR of HIGH BYTE occurred as error of LOW BYTE

ERRO_13:
   ===> PARITY ERROR in BANK_0 occurred as PARITY ERROR in
        BANK_1

ERRO_14:
   ===> PARITY ERROR in BANK_1 occurred as PARITY ERROR in
        BANK_0

ERRO_15:   ===> BANK(0/1) characterize erroneous occurred
ERRO_16:   ===> deletion of PARITY ERROR impossible
ERRO_17:   ===> NMI occurred but NMI has been disabled
                (:= NMI Sperre in der ZVE nicht wirksam)
```

Bei Mehrfachzuordnung von Paritaetsbitfehlerregistern (PER) zu gleichen I/O-Adressen kann es zu einer Verfaelschung des Fehlerbildes kommen. Die von den verschiedenen Paritaetsbitfehlerregistern gesendeten Daten ueberlagern sich und taeuschen auf diese Weise einen anderen Fehler vor.
Falls undefiniertes Verhalten oder 'Abstuerze' des Rechners den normalen Programmablauf verhindern oder falls keine Fehlerausschriften generiert sind, koennen in folgenden Zellen Informationen ueber aufgetretene Fehlfunktionen nach einem provozierten Paritaetsbitfehler entnommen werden.

- DATA_MEMCOM:0045 : DW ERROR_RUN_PARLOC
  Enthaelt aktuelle Fehlfunktion zur Uebergabe an die Ausgaberoutine und wird am Schluss der Ausgabe geloescht.
  (Bit := "1" (gesetzt) = Fehlfunktion)
  (Bit := "0" (nicht gesetzt) = keine Fehlfunkt.)
  Bit 0: Fehlermeldung fuer ERRO_1
  Bit 1: - " - ERRO_2
  :
  :
  Bit 15: - " - ERRO_16
  Bit 0 u.1: - " - ERRO_17
  (Verwechselung mit ERRO_1 und ERRO_2 moeglich!)

- DATA_MEMCOM:0042 : DW ERROR_PARLOC_OPS
  wie ERROR_RUN_PARLOC, enthaelt aber alle Fehlfunktionen im TEST06 kumulativ

Fehlermerkzellen fuer Fehler beim Zuordnungstest PER <----> OPS_X (Initialwert: 0H). Der Aufbau entspricht ERROR_RUN_PARLOC

```
- DATA_TEST06:0000 : DW ERR_OPS_1_ALLOC
- DATA_TEST06:0002 : DW ERR_OPS_2_ALLOC
- DATA_TEST06:0004 : DW ERR_OPS_3_ALLOC
- DATA_TEST06:0006 : DW ERR_OPS_4_ALLOC
```

Merkzellen fuer die Zuordnung PER <---> OPS_X (1 - 4: bei fehlerhafter Zuordnung von bis zu 4 Paritaetsbitfehlerregistern (PER) zu einem OPS) (Initialwert: 3FH)

```
- DATA_TEST06:000E :   DB      PER_OPS_11
- DATA_TEST06:000F :   DB      PER_OPS_12
- DATA_TEST06:0010 :   DB      PER_OPS_13
- DATA_TEST06:0011 :   DB      PER_OPS_14
- DATA_TEST06:0012 :   DB      PER_OPS_21
- DATA_TEST06:0013 :   DB      PER_OPS_22
- DATA_TEST06:0014 :   DB      PER_OPS_23
- DATA_TEST06:0015 :   DB      PER_OPS_24
- DATA_TEST06:0016 :   DB      PER_OPS_31
- DATA_TEST06:0017 :   DB      PER_OPS_32
- DATA_TEST06:0018 :   DB      PER_OPS_33
- DATA_TEST06:0019 :   DB      PER_OPS_34
- DATA_TEST06:001A :   DB      PER_OPS_41
- DATA_TEST06:001B :   DB      PER_OPS_42
- DATA_TEST06:001C :   DB      PER_OPS_43
- DATA_TEST06:001D :   DB      PER_OPS_44
```

Andere Fehlerausschriften beginnen mit einem eingerueckten Pfeil ===> und der Ausgabe eines Textes (siehe Beschreibungen der Unterprogramme im Pkt. 8.2.4. der Module MEMCOM, MEMLIM und MEMERR).
Dann wird in Abhaengigkeit von DEBUG durch einen INT 3 der Monitorbedienzustand erreicht (Standardgenerierung). Der Bediener kann mittels Monitorkommandos den Fehler eingrenzen, verdichten oder ausblenden (siehe Punkt 5.).

9.2. Fortsetzung nach Fehler

Nach der Ausgabe der Fehlermeldung erfolgt Programmhalt im Falle der Standardgenerierung (s. Pkt. 5.2.).

```
TDERRONLY = 0
TDDEBUG   = 3
```

Die Fortsetzung des Tests kann durch das Monitorkommando G<CR> erfolgen.

10. Verschiedenes

10.1. Programmabbruchmoeglichkeiten

Der Programmablauf kann durch die BREAK-Taste bedingt unterbrochen werden (siehe Punkt 6.2.). Dabei ist zu beachten, dass der NMI-Vektor fuer die Behandlug von Paritaetsbitfehlern im Testprogramm ueberschrieben ist (siehe MEMERR-Beschreibung). Weiterstart mit G<CR> ist erlaubt.
Ein geordneter Abbruch kann durch Eingabe von CTRL-C erfolgen, damit erfolgt der Uebergang in den Monitorbedienzustand vor der naechsten Ausgabe. Alle Interruptvektoren haben jetzt den fuer die Monitorbedienung notendigen Wert.

Restart auf der Startadresse des Testmoduls: CODE_LACS:40 ist moeglich.

10.2. Kontrolle der Programmdurchfuehrung

Am Ende von TEST06 wird bei aufgetretenem Fehler (NUMBER_READ_ERROR # 0H) der im Programm intern gefuehrte Zyklus-zaehler und die Anzahl der Fehler im TEST06 ausgegeben (kumulativer Wert aus allen absolvierten Durchlaeufen des Tests 06):

```
===> OPS PARITY LOGIC TEST06 cycles:  xxxxH
     with memory errors:  xxxxH
```

Sind im aktuellen Durchlauf von TEST06 keine Fehler aufgetreten, wird die o.g. Meldung nicht ausgegeben.
Nach Abgabe der Steuerung durch den Testmodul an die Regie des Leitprogrammes LACS und die Uebergabe des fehlerfreien Pruef-durchlaufes wird durch die folgende LACS-Meldung die Durchfuehrung des Testprogramms auf dem Bildschirm bestaetigt (Standard-fall):

```
TEST06: OPS PARITY LOGIC TEST                    "PASSED"
```

Fehler- und Durchlaufzaehler werden in folgender Form ausgegeben:

```
 ERRORS / CYCLES:   xxxx / yyyy   (xxxx,yyyy in dezimaler Form)
```

Danach beginnt ein neuer Programmzyklus.
Nach Unterbrechung des Programms kann die Durchfuehrungszahl des Testprogramms TEST06 auch aus der Zelle:

```
CONST_LACS:0060
```

ermittelt werden (Monitorkommando DW). Die Anzahl der an LACS uebergebenen Fehlermeldungen kann man der Zelle:

```
CONST_LACS:005C
```

entnehmen.
Als Fehler werden in LACS fehlerhafte Durchlaeufe des Testprogrammes gezaehlt (:= Ruecksprueenge mit gesetztem Fehlerkennzeichen). Die echte Fehlerzahl wird durch das Testprogramm selbst ausgegeben oder ist der Zelle NUMBER_READ_ERROR_6 (MEMCOM) zu entnehmen.

1. TEST07, ZPS PARITY LOGIC TEST

2. Funktion des Programms

Der "ZPS Parity Logic Test" prueft die Funktion der Logik fuer die Paritaetsbitpruefung des ZPS. Es werden gezielt Paritaetsbitfehler erzeugt und die Reaktion ueberprueft.
Im ZPS uebernimmt das "Control-Byte" (Speicheradresse: 400H) die Steuerung der Paritaetskontrolle und der Fehlerbewertung. Das "Control-Byte" hat folgenden Aufbau:

Bedeutung der einzelnen Bits:

```
xxxx 0000
     ^^^^
     ||||
     |||0 : ungerade Paritaet
     |||1 : gerade Paritaet
     ||0  : LED - Anzeige verhindert bzw. geloescht
     ||1  : LED - Anzeige erlaubt, wenn gleichzeitig
     ||     ZPS-INTR erl.
     |0   : ZPS-INTR verhindert bzw. geloescht
     |1   : ZPS-INTR erlaubt
     0    : ZPS-NMI verhindert bzw. geloescht
     1    : ZPS-NMI erlaubt
```

Die Fehlermeldung ZPS-INTR bewirkt im PPI Port A (I/O - Adresse: C8) das Setzen folgender Bits:

```
11xx xxxx
  ^^
  ||
  |0 : PBF im HIGH - Byte
  |1 : kein PBF im HIGH - Byte
  0  : PBF im LOW - Byte
  1  : kein PBF im LOW - Byte
```

Die NMI-Fehlermeldung wird durch das Testprogramm im Register SI registriert. Die LED-Anzeige wird nicht ausgewertet.

3. Voraussetzungen

Im Folgenden werden Speicheradressen in der Form Segment:Offset hexadezimal angegeben, wobei das Segment absolut oder durch einen symbolischen Namen bezeichnet sein kann. Die Zuordnung von symbolischem Segmentnamen und absoluten Adressen ist aus der MP2-Liste zu entnehmen.
In speziellen Faellen (z.B. Darstellung der Fehlerausschrift oder Bereichsangaben) wird die Speicheradresse absolut mit max. 5 hexadezimalen Zahlen (Kennzeichnung mit H) angegeben.

3.1. Geraeteausruestung

Es ist die Grundausruestung des Rechners A 7100 mit ZPS erforderlich. Die Ein/Ausgabe richtet sich nach der im Monitor generier-

ten Geraetekonfiguration.

3.2. Speicherbedarf

ca. 2,9 KB fuer TEST07.OBJ
Die absoluten Adressen der durch den Locator zugewiesenen Speicherbereiche sind anhand der MP2-Liste zu berechnen.

3.3. Test zusaetzlicher Baugruppen

TEST07 erfordert u.a. eine in den wesentlichen Teilen funktionierende ZVE einschliesslich der Interrupt- und NMI Behandlungslogik.

3.4. Nutzung anderer Programmodule

Der Testmodul TEST07.OBJ muss mit den Hilfsmodulen MEMCOM.OBJ, MEMLIM.OBJ, MEMERR.OBJ, dem Leitprogrammodul LACS.OBJ und der Bibliothek LACS.LIB verbunden sein.
Es werden Unterprogramme (Routinen) aus diesen Modulen genutzt.
Aus dem Hilfsmodul MEMCOM werden folgende Routinen genutzt:
NMI_VEK_RETURN :Behandlung NMI-Vektor am Programmende
- RESET_PARITY :Ruecksetzen Paritaetsfehlerregister OPS, ZPS
- DISPLAY_VERSION :Ausgabe der Versionsnummer
Aus dem Hilfsmodul MEMLIM werden folgende Routinen genutzt:
- OPS_OR_ZPS :Ermittlung des von OH ... 1FFFFH konfigurierten Speichers
Aus dem Hilfsmodul MEMERR werden folgende Routinen genutzt:
- NMI_OCCURRED :Behandlungsprogramm fuer Paritaetsbitfehler (NMI), die beim Programmlesen aufgetreten sind
- SEARCH_PARITY_ERRORS :Suchprogramm fuer Paritaetsbitfehler
Aus LACS werden folgende Routinen genutzt:
- ASCII_CONV :Umwandlung Hex ——> ASCII
- TDMASKEDMESSAGE :von DEBUG abhaengige Zeichenausgabe
- TDDISPLAY :von DEBUG unabhaengige Zeichenausgabe

4. Ladeprozedur fuer autonomen Ablauf

Das Testprogramm wird im Verbund mit dem Leitprogramm LACS von Minidiskette gebootet (Kommandobeispiele siehe Pkt. 4.1. - 4.3.).

4.1. Laden mit Angabe des Geraetenamens

.B :Fn:TEST<CR> (n = Laufwerksnr.)
Nach Erreichen des Monitorbedienzustandes:
.B :Fn:G07<CR> (n = Laufwerksnr.)
Anschliessend verweilt das Programm im Generierhalt.

4.2. Laden ohne Angabe des Geraetenamens

.B TEST<CR>
Nach Erreichen des Monitorbedienzustandes:
.B G07<CR>
Bei letzterem Kommando werden TEST bzw. G07 auf allen Laufwerken gesucht, beginnend mit F0. Anschliessend verweilt das Programm im Generierhalt.

4.3. Laden durch Kommandoeingabe im ACT (A 7100 - Confidence-Test)

Nach Netzeinschalten oder nach Druecken der Taste RESET besteht die verkuerzte Lademoeglichkeit von LACS durch Eingabe des Zeichens "T" innerhalb des PIC - Tests von ACT.
Nach Erreichen des Monitorbedienzustandes weiter wie oben.

5. Generierinformationen

Der Bediener hat die Moeglichkeit, die Ausgabe von Durchlauf- und Fehlermeldungen sowie allgemeine Mitteilungen ueber den Zustand des Prueflings durch Eingabe von Generierinformationen zu steuern. (siehe Punkt 5.1.). Dazu muss der Programmablauf unterbrochen werden (z.B. durch CTRL-C). Mittels des Monitorkommandos SW kann dann eine Aenderung der Generierung vorgenommen werden. Fuer spezielle Pruefzwecke koennen die verwendeten standardmaessigen Pruefmuster (MUSTER1 und MUSTER4) im Datensegment DATA_MEMCOM veraendert werden (siehe MEMCOM-Beschreibung).

5.1. Generierung von Meldungen und Mitteilungen

TEST07 laeuft unter der Regie des Leitprogrammes LACS. Deshalb werden Meldungen ueber den Programmablauf durch die Variablen
TDERRONLY (Adr.: DATA_LACS:0000) und
TDDEBUG (Adr.: DATA_LACS:0002) gesteuert
(siehe Punkt 5.1. der Beschreibung des Leitprogrammes LACS).

5.2. Generierung spezieller Testablaeufe

Die Wirkung von DEBUG auf die Fehlerausschriften und den Fehlerhalt ist in der MEMERR-Beschreibung Pkt. 5.2. dargestellt.

6. Startprozedur

6.1. Start des Testprogramms

Das Testprogramm TEST07 laeuft unter der Regie von LACS. Nach dem Laden (siehe Punkt 4.) erfolgt der Start:
- Aus dem Generierhalt durch das Monitorkommando

.G<CR>
- Restart erfolgt durch das Monitorkommando
.G <CODE_LACS>:40<CR>.

6.2. Bedieneraktionen

Der Bediener hat folgende Eingriffsmoeglichkeiten waehrend des Programmablaufs:
- CTRL-S : Unterbrechung der Ausgabe, Fortsetzung mit CTRL-Q.
- CTRL-Q : Fortsetzung einer unterbrochenen Ausgabe.
- CTRL-P : Hardcopy der Bildschirmausgabe auf angeschlossenen Drucker
  Erneutes CTRL-P setzt die Hardcopy zurueck.
- CTRL-C : Uebergang in den Bedienzustand des Monitors.

Die Wirkung dieser Kommandos erfolgt erst unmittelbar vor Ausgabe einer Meldung auf den Bildschirm.
Der Uebergang in den Bedienzustand des Monitors ist auch moeglich durch Druecken der Taste BREAK (:= INT 1) auf der Tastatur, wobei Pkt. 10.1. zu beachten ist. Damit ist dem Bediener ein Mittel gegeben, einen Fehler im Programmablauf von TEST07 zu erzeugen.

7. Einstellungsmoeglichkeiten

Die Standardeinstellung fuer TEST07 lautet:
- TDERRONLY: 0 : siehe Punkt 5.1. LACS-Beschreibung
- TDDEBUG : 3 : - " -

8. Programmbeschreibung

8.1. Programmablauf

Im 1. Durchlauf erfolgt vor dem eigentlichen Test eine von DEBUG abhaengige Ausgabe der Programmnummern, Modulnamen und der Zeichnungs- und Ausgabenummern aller beteiligten Module. Sie hat fuer TEST07 folgendes Aussehen (siehe Punkt 8.2.1.):

```
TEST07  yy-mm-dd  1.56.703008.2/67-aa-56ACnnn
MEMCOM  yy-mm-dd  1.56.703051.5/67-aa-56ACnnn
MEMLIM  yy-mm-dd  1.56.703052.3/67-aa-56ACnnn
MEMERR  yy-mm-dd  1.56.703053.1/67-aa-56ACnnn
```

dabei ist: yy-mm-dd: Datum der letzten Aenderung
aa: Ausgabenummer
56ACnnn: Nr. der Aenderungsmitteilung

Anschliessend beginnt die Abarbeitung des Testmoduls mit dem Einstellen der Anfangsbedingungen:
- Retten und Setzen der Interruptmaske fuer INT 1 (:= <BREAK>)
- Loeschen des Fehlerzaehlers und der allgemeinen Merkzellen

Mit "OPS_OR_ZPS" wird die Konfiguration im Speicherbereich OH ...

1FFFFH ermittelt. Ist kein ZPS vorhanden, wird eine von DEBUG unabhaengige Meldung ausgegeben und der Test abgebrochen:

```
===> ZPS not exists
     Test aborted, goes to next test
```

Ist ein ZPS vorhanden, erfolgt das Retten und der Austausch des NMI-Vektors fuer die Paritaetsfehler- (NMI) Behandlung im Testprogramm.
Alle Paritaetsfehlermeldungen werden geloescht und der Loeschzustand ueberprueft. Anschliessend erfolgt das Lesen einer Zelle, bei dem kein Paritaetsfehler auftreten darf. Abweichungen fuehren unter Beachtung der DEBUG-Variablen zu einer Fehlermeldung entsprechend Pkt. 9.1.

Danach werden in 4 Durchlaeufen zu je 9 Routinen Paritaetsbitfehler in einer Speicherzelle durch Aenderung des Steuerbits im Control-Byte provoziert und die Reaktion ueberprueft.

Durchlaeufe:

Durchlauf Nr.	Paritaet im Ausgangszustand	Paritaet bei Fehlerprovokation	Paritaet im Testmuster := Anzahl "1"
1.Durchlauf	ungerade	gerade	gerade
2.Durchlauf	ungerade	gerade	ungerade
3.Durchlauf	gerade	ungerade	gerade
4.Durchlauf	gerade	ungerade	ungerade

Routinen:

Routine Nr.	Testadresse, geprueft.Teil	erlaubte Fehlermeldung
Routine 1	Low - Byte	INT
Routine 2	High - Byte	INT
Routine 3	Wort	INT
Routine 4	Low - Byte	NMI, INT
Routine 5	High - Byte	NMI, INT
Routine 6	Wort	NMI, INT
Routine 7	Low - Byte	NMI
Routine 8	High - Byte	NMI
Routine 9	Wort	NMI

Eine Routine enthaelt folgende Schritte:
- Steuerbit im Control-Byte auf Paritaet zur Fehlerprovozierung setzen, INT und NMI verbieten
- Testadresse (Low-, High-Byte oder Wort) mit Testmuster beschreiben
- Paritaetssteuerbit zuruecksetzen, INT und/oder NMI je nach Routine erlauben

- Testadresse wortweise lesen (dabei muss der erwartete Paritaetsbitfehler im Low-, High-Byte oder Wort auftreten) Fehlermeldung aus Port A (I/O - Adresse: C8) einlesen Control-Byte loeschen
- Urspruenglichen Inhalt der Testadresse zurueckschreiben
- NMI - Meldung und Fehlermeldung auswerten, bei Fehlfunktion erfolgt Fehlerausschrift (von DEBUG abhaengig) und Fehlerzaehlung
- Loeschen des Control-Byte ueberpruefen

Im Normalzustand wird im ZPS auf ungerade Paritaet ergaenzt (Bit 0 des Control-Byte := 0). Fuer den 3. und 4 Durchlauf muss das Steuerbit geaendert werden.
Aus diesem Grunde wird der gesamte ZPS gelesen und mit geaenderter Paritaet bei verbotener Paritaetsfehlerbewertung (NMI, ZPS-INTR) zurueckgeschrieben, so dass beim spaeteren Programmlesen kein Paritaetsbitfehler auftreten kann. Nach dem 4. Durchlauf wird der Normalzustand wieder hergestellt.

Am Schluss des Programms wird der NMI-Vektor zurueckgeschrieben (NMI_VEK_RETURN), die Interruptmaske zurueckgestellt, auf Paritaetsbitfehler (PBF) (SEARCH_PARITY_ERRORS) und auf fehlerfreien Durchlauf kontrolliert (siehe Pkt. 10.2.).
Im Falle einer Fehlfunktion nach einem provozierten Paritaetsbitfehler informiert das Testprogramm den Bediener durch Fehlerausschriften (beachte Punkt 5. und 9.) und geht im Standardfall durch einen INT 3 in den Monitorbedienzustand.
Die Rueckkehr zum Leitprogramm erfolgt durch die Uebergabe der Fehlermeldung in AX und den Return-Befehl: (FAR-) RET.

fehlerfreier Durchlauf des Tests: AX := 1
fehlerhafter Durchlauf des Tests: AX := 0

Zum Abbruch des Testprogramms TEST07 siehe Punkt 10.

8.2. Anschlussbedingungen

8.2.1. Struktur des Quellprogramms

Der Modul TEST07 ist folgendermassen aufgebaut:
1. ¤TITLE(TEST07 Zeichnungsnummer-Ausgabenummer)
2. Programmbeschreibung nach dem Muster der Beschreibung des Leitprogrammes LACS.
3. NAME TEST07
4. Makrodefinitionen.
5. EXTRN-Erklaerungen
6. Datensegmente der Tests: TEST02 ... TEST07 und der Hilfsmodule MEMCOM, MEMLIM, MEMERR mit den Bezeichnungen:
   DATA_TEST02, DATA_TEST03, DATA_TEST04, DATA_TEST05,
   DATA_TEST06, DATA_TEST07, DATA_MEMCOM, DATA_MEMLIM,
   DATA_MEMERR, DATA_MEMINP,
   die zusammen die Gruppe: DGROUP_MEMORY bilden. Die Datensegmente (ausser DATA_TEST07) sind Pseudosegmente fuer die Ermoeglichung der Gruppenbildung. Sie enthalten entweder nichts oder die EXTRN-Erklaerungen der fuer TEST07 aus dem entspre-

chenden Modul benutzten Daten.

7. Codesegmente der Tests: TEST02 ... TEST07 und der Hilfsmodule MEMCOM, MEMLIM, MEMERR mit den Bezeichnungen:
   CODE_TEST02, CODE_TEST03, CODE_TEST04, CODE_TEST05,
   CODE_TEST06, CODE_TEST07, CODE_MEMCOM, CODE_MEMLIM,
   CODE_MEMERR,
   die zusammen die Gruppe: CGROUP_MEMORY bilden. Die Codesegmente (ausser CODE_TEST07) sind Pseudosegmente fuer die Ermoeglichung der Gruppenbildung. Sie enthalten entweder nichts oder die EXTRN-Erklaerungen der fuer TEST07 aus dem entsprechenden Modul benutzten Unterprogramme.

PUBLIC-Erklaerungen stehen in dem Segment, in dem die betreffenden Symbole vorkommen, ganz am Anfang des Segments. Im Segment CODE_TEST07 steht nach den PUBLIC-Erklaerungen folgende Zeile:

TEST07 yy-mm-dd 1.56.703008.2/67-aa-56ACnnn'

dabei ist: yy-mm-dd: Datum der letzten Aenderung
aa: Ausgabenummer
56ACnnn: Nr. der Aenderungsmitteilung

Dieser Text wird nach dem Start des Testmoduls in Abhaengigkeit von TDDEBUG (s. Pkt. 5.1.) ausgegeben.

8.2.2. Stack- und Registernutzung

Von TEST07 wird der Stack des Leitprogramms LACS genutzt, d.h. beim Start von TEST07 wird kein SS sondern nur DS initialisiert.

8.2.3. Protokollsteuerung

Die Ausgaben sind je nach den Werten, die die Variablen TDERRONLY und TDDEBUG enthalten, abgestuft unterdrueckbar (s. Pkt. 5.1. der MEMERR-Beschreibung). Es gelten die Richtlinien gemaess Punkt 8.2.3. der Beschreibung des Leitprogrammes LACS.

8.2.4. Nutzbare TEST07-Routinen

TEST07 enthaelt keine Routinen, die fuer andere Nutzer verfuegbar sind

8.2.5. Nutzbarkeit anderer Routinen

Siehe Punkt 3.4.

9. Fehler

9.1. Fehlerausgaben

Bei fehlerhaften Testergebnissen wird in Abhaengigkeit von DEBUG (siehe Pkt. 5.2. der MEMERR-Beschreibung) eine geeignete Meldung auf dem Bildschirm ausgegeben.
Die Fehlerangabe nach einem provozierten Paritaetsbitfehler enthaelt in der 1. Zeile den Durchlauf (1. Ziffer) und die Routine (2. Ziffer) entsprechend der Programmbeschreibung Pkt. 8.1.

```
        RUN-, ROUT-COUNT: xx
```

Danach folgt eine Textausgabe, die zusammen mit der obigen Angabe die aufgetretenen Fehlfunktionen charakterisiert.
Folgende Textausgaben sind moeglich:

```
ERRZ1:    ===> ZPS not exists
                Test aborted, goes to next test
ERRZ2:
   ===> ZPS PARITY LOGIC TEST impossible because PARITY ERROR
        occurred in ZPS
        (:= PBF beim Programmlesen, dabei ist aber kein NMI
         aufgetreten)

ERRZ3:    ===> PARITY ERROR not occurred for LOW BYTE
ERRZ4:    ===> PARITY ERROR not occurred for HIGH BYTE
ERRZ5:    ===> deletion of PARITY ERROR impossible for LOW  BYTE
ERRZ6:    ===> deletion of PARITY ERROR impossible for HIGH BYTE

ERRZ7:
   ===> Test for creating NMI impossible because NMI occurred
        in memory
        (:= PBF beim Programmlesen, dabei ist gleichzeitig ein
         NMI aufgetreten)

ERRZ8:
   ===> PARITY ERROR in LOW BYTE has not been effected NMI

ERRZ9:
   ===> PARITY ERROR in HIGH BYTE has not been effected NMI

ERRZ10:
  ===> PARITY ERROR of LOW BYTE occurred as error of HIGH BYTE

ERRZ11:
  ===> PARITY ERROR of HIGH BYTE occurred as error of LOW BYTE

ERRZ12:
  ===> PARITY ERROR of LOW BYTE occurred but INT disabled

ERRZ13:
  ===> PARITY ERROR of HIGH BYTE occurred but INT disabled
```

ERRZ14: ===> occurred NMI, but NMI disabled
(:= NMI Sperre im ZPS oder in der ZVE nicht wirksam)

Falls undefiniertes Verhalten oder 'Abstuerze' des Rechners den normalen Programmablauf verhindern oder falls keine Fehlerausschriften generiert sind, koennen in folgenden Zellen Informationen ueber aufgetretene Fehlfunktionen nach einem provozierten Paritaetsbitfehler entnommen werden.

- DATA_MEMCOM:000D : DB ROUT_COUNT_1
Durchlaufzaehler entspr. Beschreibung Pkt. 8.1.
- DATA_MEMCOM:000E : DB ROUT_COUNT_2
Routinezaehler entpsr. Beschreibung Pkt. 8.1.
- DATA_MEMCOM:0045 : DW ERROR_RUN_PARLOC
Enthaelt aktuelle Fehlfunktion zur Uebergabe an die Ausgaberoutine und wird am Schluss der Ausgabe geloescht.
(Bit := "1" (gesetzt) = Fehlfunktion)
(Bit := "0" (nicht gesetzt) = keine Fehlfunkt.)
Bit 0: Fehlermeldung fuer ERRZ1
Bit 1: - " - ERRZ2
:
:
Bit 13: - " - ERRZ14
- DATA_MEMCOM:0040 : DW ERROR_PARLOC_ZPS
wie ERROR_RUN_PARLOC, enthaelt aber alle Fehlfunktionen im TEST07 kumulativ

Andere Fehlerausschriften beginnen mit einem eingerueckten Pfeil ===> und der Ausgabe eines Textes (siehe Beschreibungen der Unterprogramme im Pkt. 8.2.4. der Module MEMCOM, MEMLIM und MEMERR).
Dann wird in Abhaengigkeit von DEBUG durch einen INT 3 der Monitorbedienzustand erreicht (Standardgenerierung). Der Bediener kann mittels Monitorkommandos den Fehler eingrenzen, verdichten oder ausblenden (siehe Punkt 5.).

9.2. Fortsetzung nach Fehler

Nach der Ausgabe der Fehlermeldung erfolgt Programmhalt im Falle der Standardgenerierung (s. Pkt. 5.2.).

TDERRONLY = 0
TDDEBUG = 3

Die Fortsetzung des Tests kann durch das Monitorkommando G<CR> erfolgen.

10. Verschiedenes

10.1. Programmabbruchmoeglichkeiten

Der Programmablauf kann durch die BREAK-Taste bedingt unterbrochen werden (siehe Punkt 6.2.). Dabei ist zu beachten, dass der NMI-Vektor fuer die Behandlug von Paritaetsbitfehlern im Testprogramm ueberschrieben ist (siehe MEMERR-Beschreibung). Weiterstart mit G<CR> ist erlaubt.
Ein geordneter Abbruch kann durch Eingabe von CTRL-C erfolgen, damit erfolgt der Uebergang in den Monitorbedienzustand vor der naechsten Ausgabe. Alle Interruptvektoren haben jetzt den fuer die Monitorbedienung notendigen Wert.
Restart auf der Startadresse des Testmoduls: CODE_LACS:40 ist moeglich.

10.2. Kontrolle der Programmdurchfuehrung

Am Ende von TEST07 wird bei aufgetretenem Fehler (NUMBER_READ_ERROR # 0H) der im Programm intern gefuehrte Zykluszaehler und die Anzahl der Fehler im TEST07 ausgegeben (kumulativer Wert aus allen absolvierten Durchlaeufen von TEST07):

```
===> ZPS PARITY LOGIC TEST07 cycles:  xxxxH
     with memory errors:  xxxxH
```

Sind im aktuellen Durchlauf von TEST07 keine Fehler aufgetreten, wird die o.g. Meldung nicht ausgegeben.
Nach Abgabe der Steuerung durch den Testmodul an die Regie des Leitprogrammes LACS und die Uebergabe des fehlerfreien Pruefdurchlaufes wird durch die folgende LACS-Meldung die Durchfuehrung des Testprogramms auf dem Bildschirm bestaetigt (Standardfall):

```
TEST07: ZPS PARITY LOGIC TEST                    "PASSED"
```

Fehler- und Durchlaufzaehler werden in folgender Form ausgegeben:

```
 ERRORS / CYCLES:   xxxx / yyyy (xxxx,yyyy in dezimaler Form)
```

Danach beginnt ein neuer Programmzyklus.
Nach Unterbrechung des Programms kann die Durchfuehrungszahl des Testprogramms TEST07 auch aus der Zelle:

```
CONST_LACS:006E
```

ermittelt werden (Monitorkommando DW). Die Anzahl der an LACS uebergebenen Fehlermeldungen kann man der Zelle:

```
CONST_LACS:006C
```

entnehmen.
Als Fehler werden in LACS fehlerhafte Durchlaeufe des Testprogrammes gezaehlt (:= Ruecksprueng mit gesetztem Fehlerkennzeichen). Die echte Fehlerzahl wird durch das Testprogramm selbst ausgegeben oder ist der Zelle NUMBER_READ_ERROR_7 (MEMCOM) zu entnehmen.

1. TEST08, Byte/Word Transfer Test

2. Funktion des Programms

Der Testmodul TEST08 fuehrt die folgende Funktion aus:
Test der Datenleitungen sowie der Byte- und/oder Wort-Uebertragung zum ZPS/OPS und zurueck. In die Speicherzellen ES:0F000H, :0F001H, :0F002H, :0F003H wird das Testmuster 05AA5H eingeschrieben und gemaess Testalgorithmus byteweise/wortweise bzw. ueber Wortgrenze uebergreifend gelesen. Es folgt der SOLL-IST Vergleich. Durch die Wahl von <ES> werden die zu testenden 64KB-Speichersegmente festgelegt. Standardbelegung fuer ES: 0000H, 1000H, 2000H, 3000H, d.h. es werden die ersten vier Speicherseiten angesprochen.

3. Voraussetzungen

In der Beschreibung werden Speicheradressen hexadezimal in der Form Segment:Offset angegeben.

3.1. Geraeteausruestung

Es ist die Grundausruestung des Rechners A7100 erforderlich. Die Ein/Ausgabe richtet sich nach der im Monitor generierten Geraetekonfiguration.

3.2. Speicherbedarf

0.6 KB fuer den Testmodul TEST08.
Die absoluten Adressen der durch den Locator zugewiesenen Speicherbereiche sind anhand der MP2-Liste zu berechnen.

3.3. Test zusaetzlicher Baugruppen

Durch den Testmodul werden explizit keine zusaetzlichen Baugruppen getestet.

3.4. Nutzung anderer Programmodule

Der Testmodul TEST08.OBJ ist mit dem Bibliotheksmodul LIBRAR.OBJ, iem Leitmodul LACS.OBJ und der Bibliothek LACS.LIB verbunden.

Aus LACS.OBJ werden genutzt:

Anspruenge:	-TEST_DONE
Routinen:	-ASCII_CONV
	-NEWLINE
	-TDMASKEDMESSAGE
	-THREESPACE

4. Ladeprozedur

Das Testprogramm TEST08 wird im Verbund mit dem Leitprogramm LACS von Minidiskette gebootet (Kommandobeispiele siehe Pkt.4.1 - 4.3).

4.1. Laden mit Angabe des Geraetenamens

```
.B :Fn:TEST<CR>          (n = Laufwerksnr.)
```
Nach Erreichen des Monitorbedienzustandes:
```
.B :Fn:G08<CR>           (n = Laufwerksnr.)
```
Anschliessend verweilt das Programm im Generierhalt.

4.2. Laden ohne Angabe des Geraetenamens

```
.B TEST<CR>
```
Nach Erreichen des Monitorbedienzustandes:
```
.B G08<CR>
```
Bei letzterem Kommando werden TEST bzw. G08 auf allen Laufwerken gesucht, beginnend mit F0. Anschliessend verweilt das Programm im Generierhalt.

4.3. Laden durch Kommandoeingabe im ACT (A 7100-Confidence-Test)

Nach Netzeinschalten oder nach Druecken der Taste RESET besteht die verkuerzte Lademoeglichkeit von LACS durch Eingabe des Zeichens "T" innerhalb des PIC - Tests von ACT.

5. Generierinformationen

Der Bediener hat die Moeglichkeit, die Ausgabe von Durchlauf- und Fehlermeldungen durch Eingabe von Generierinformationen zu steuern. Dazu muss der Programmablauf unterbrochen werden (z.B. durch CTRL-C), bzw. das Programm muss sich im Generierhalt befinden (s. Pkt. 4.1./ 4.2.). Mittels des Monitorkommandos SW kann dann eine Aenderung der Generierung vorgenommen werden.

5.1. Generierung der Meldungen

TEST08 laeuft unter der Regie des Leitprogrammodules LACS. Deshalb werden Meldungen ueber den Programmablauf durch die Wort-Variablen TDERRONLY (Adr. DATA_LACS:0000) und TDDEBUG (Adr. DATA_LACS:0002) gesteuert. (s. Pkt. 5.1 der Beschreibung des LACS-Moduls)

6. Startprozedur

6.1. Start des Testprogramms

Das Testprogramm TEST08 laeuft unter der Regie von LACS. Nach dem Laden (siehe Pkt. 4.) erfolgt der Start:

- Aus dem Generierhalt durch das Monitorkommando
  .G<CR>
- Restart erfolgt durch das Monitorkommando
  .G<CODE_LACS>:40<CR>.

6.2. Bedieneraktionen

Der Bediener hat folgende Eingriffsmoeglichkeiten waehrend des Programmablaufs:

- CTRL-S Unterbrechung der Ausgabe, Fortsetzung mit CTRL-Q
- CTRL-Q Fortsetzung einer unterbrochenen Ausgabe
- CTRL-P Hardcopy der Bildschirmausgabe auf angeschlossenen Drucker, erneutes CTRL-P setzt die Hardcopy-Funktion zurueck
- CTRL-C Uebergang in den Bedienzustand des Monitors

Die Wirkung dieser Kommandos erfolgt erst unmittelbar vor einer Ausgabe einer Meldung auf den Bildschirm.

7. Einstellungsmoeglichkeiten

Gemaess Standardeinstellung wird der Test zyklisch wiederholt. Pro Zyklus wird das Testprogramm einmal abgearbeitet.

8. Programmbeschreibung

8.1. Programmablauf

Vor dem 1. Durchlauf erfolgt die Ausgabe der Programmnummern, Modulnamen und der Zeichnungs- und Ausgabenummern aller beteiligten Module. Sie hat fuer TEST08 folgendes Aussehen (siehe Punkt 8.2.1.):

```
TEST08  yy-mm-dd  1.56.703009.0/67-aa-56ACnnn
        mit       yy-mm-dd = Erstellungsdatum
                  aa       = Ausgabenummer
                  nnn      = Nr. der Aenderungsmitteilung
```

Anschliessend erfolgt der eigentliche Test. Seine Laufzeit betraegt kleiner 1 sec. Waehrend der Testdurchfuehrung koennen entsprechend den Erfordernissen die Generierinformationen umgestellt werden. Die Rueckkehr zum Leitprogrammodul erfolgt im fehlerfreien Fall durch einen RET (zum Leitprogrammodul)
AX enthaelt den Rueckkehrcode "1"

Im Fehlerfall wird der Test wie folgt beendet:
1. Ausgabe der Fehlermeldung:
   TEST08_ERROR NO. <XX> H

   mit <xx>: 0D000H - Fehler beim Schreiben eines Wortes auf ungerade Adresse und Wort-Lesen von davorliegender gerader Adresse
   0D001H - Fehler beim Wort-Lesen von naechster gerader Adresse
   0D002H - Fehler beim Byte-Schreiben auf gerader Adresse

2. Ruecksprung zum Leitprogrammodul mit
   JMP TEST_DONE
   Es erfolgt daraufhin automatisch eine Ausgabe durch die Routine DISPLAY_RESULTS. Die Parameter werden vor dem JMP in folgenden Registern uebergeben:
   - BX: Offset der Adresse der Testspeicherzelle
   - DX: Sollwert
   - AX: Istwert
   - ES: Segment der Adresse der Testspeicherzelle

Zum Abbruch des Tests s.Pkt.10.

8.2. Anschlussbedingungen

8.2.1. Struktur des Quellprogramms

Der Testmodul TEST08 ist folgendermassen aufgebaut:
1. $TITLE(TEST08 Zeichnungsnr.-Ausgabenr.-Aend.Mitt.Nr.)
2. Programmbeschreibung
3. NAME TEST08
4. Es folgen die EXTRN-und PUBLIC-Erklaerungen.
5. Es folgt das Datensegment mit der Bezeichnung DATA_TEST08.
6. Es folgt das Codesegment mit der Bezeichnung CODE_TEST08.

Das Datensegment DATA_TEST08 enthaelt die folgenden Variablen:
- T08_CS Codesegment des Testprogrammes
- T08_IP Offset der Programmzeile, in deren unmittelbarer Naehe der Fehler geortet wurde

Es folgen Variable zum Retten von Registern:
- T08_AX Enthaelt den Ist-Wert
- T08_BX Enthaelt Offset der Testzelle
- T08_CX
- T08_DX Enthaelt den Soll-Wert
- T08_SI
- T08_DI
- T08_ES Enthaelt Segment der Testzelle

Weitere Variable sind:
- T08_BAD_CODE Enthaelt den Fehlercode (s. Pkt. 8.1.
- T08_VERS_SWITCH Datenbyte, das auf 1 gesetzt die Bildschirmausgabe der Version bewirkt

- T08_ERROR Textrahmen fuer Fehlermitteilung

Im Segment CODE_TEST08 stehen am Anfang folgende Zeilen:

```
VERSION_TEST08  DB  'TEST08  '
                DB  'yy-mm-dd  '
                DB  'Zeichn.nr.-Ausg.nr.-Aend.Mitt.Nr.',0AH,0DH,0
                ORG 40H
```

dabei sind: yy-mm-dd Datum der letzten Aenderung

Dieser Text wird nach dem Start des Testmoduls in Abhaengigkeit von TDDEBUG (s.Pkt.5.1) ausgegeben. Die naechste Zeile nach dem ORG-Befehl enthaelt die Ansprungstelle des Testmoduls mit der Marke TEST08.

8.2.2. Stack- und Registernutzung

Vom Testmodul TEST08 wird der Stack des Leitprogrammoduls genutzt, d.h. beim Start von TEST08 wird kein SS sondern nur DS initialisiert.

8.2.3. Protokollsteuerung

Die Ausgaben sind je nach den Werten, die die Variablen TDERRONLY und TDDEBUG enthalten, abgestuft unterdrueckbar (s.Pkt.5.1.). Es gelten die Richtlinien gemaess Punkt 8.2.3 der LACS-Beschreibung.

8.2.4. Nutzbare TEST08-Routinen

TEST08 enthaelt keine Routinen, die fuer andere Nutzer verfuegbar sind.

8.2.5. Nutzbarkeit anderer Routinen

Siehe Punkt 3.4.

9. Fehler

9.1. Fehlerausgaben

Das Programm TEST08 gibt im Fehlerfall einen Text aus, welcher das fehlerhafte Testobjekt kennzeichnet. Dann wird der Leitprogrammodul angesprungen, welcher durch INT 3 den Monitor-Bedienzustand erreicht (Standardgenerierung). Vom Leitprogrammodul werden die Standardfehlermeldungen ausgegeben (s. LACS-Beschreibung). Der Bediener kann mittels Monitorkommandos den Fehler eingrenzen, verdichten oder ausblenden (siehe Punkt 5.).

9.2. Fortsetzung nach Fehler

Die Wiederholung des Tests erfolgt durch das Monitorkommando
G<CODE_LACS>:40<CR>

10. Verschiedenes

10.1. Programmabbruchmoeglichkeiten

Der Programmablauf kann durch die BREAK-Taste unterbrochen werden. Weiterstart mit G<CR> fuehrt zu einem Fehler. Restart auf Adresse <CODE_LACS>:40 moeglich. Ein geordneter Abbruch kann durch Eingabe von CTRL-C erfolgen, damit erfolgt der Uebergang in den Monitorbedienzustand vor der naechsten Ein/Ausgabe.

10.2. Kontrolle der Programmdurchfuehrung

Die Durchfuehrung des Testprogramms wird durch die folgende LACS-Meldung auf dem Bildschirm bestaetigt (Standardfall):
TEST08: BYTE/WORD TRANSFER TEST "PASSED"
Fehler- und Durchlaufzaehler werden in folgender Form ausgegeben:
ERRORS / CYCLES: xxxx / yyyy
Danach beginnt ein neuer Programmzyklus. Nach Unterbrechung des Programms kann die Durchfuehrungszahl des Testprogramms auch aus der Zelle
<CONST_LACS>:07C
ermittelt werden (Monitorkommando DW). Die Anzahl der aufgetretenen Fehlermeldungen kann man der Zelle
<CONST_LACS>:07A
entnehmen.

1. TEST09, CPU Test

2. Funktion des Programms

Der Testmodul TEST09 fuehrt die folgende Funktion aus:
In 109 Tests, die z.T. aus mehreren Testabschnitten bestehen, werden die Befehle des CPU-Schaltkreises K1810 WM86 getestet. Nach jedem Teiltest folgt ein SOLL-IST Vergleich des erwarteten und des durch den Testalgorithmus bestimmten IST-Wertes.
Nicht getestet werden die Befehle: IN, OUT, ESC, HLT, LOCK, WAIT.

3. Voraussetzungen

In der Beschreibung werden Speicheradressen hexadezimal in der Form Segment:Offset angegeben.

3.1. Geraeteausruestung

Es ist die Grundausruestung des Rechners A7100 erforderlich. Die Ein/Ausgabe richtet sich nach der im Monitor generierten Geraetekonfiguration.

3.2. Speicherbedarf

ca. 6,5 KB fuer den Testmodul TEST09 und 1,5 KB fuer den Modul KRUTIL. Die absoluten Adressen der durch den Locator zugewiesenen Speicherbereiche sind anhand der MP2-Liste zu berechnen

3.3. Test zusaetzlicher Baugruppen

Der Testmodul setzt einen fehlerfrei arbeitenden Arbeitsspeicher (ZPS/OPS) voraus. Weiterhin muessen Adress- und Datenbus ordnungsgemaess arbeiten.

3.4. Nutzung anderer Programmodule

Der Testmodul TEST09.OBJ ist mit dem Bibliotheksmodul LIBRAR.OBJ, dem Leitmodul LACS.OBJ ,der Bibliothek LACS.LIB und dem Hilfsmodul KRUTIL.OBJ verbunden.
Aus LACS.OBJ werden genutzt:

Anspruenge:	-RETURN_PASS
	-RETURN_FAIL
Routinen:	-ASCII_CONV
	-NEWLINE
	-TDMASKEDMESSAGE
	-THREESPACE

Aus KRUTIL.OBJ werden genutzt:
-SAVE_INTR_UT
-RESTORE_INTR_UT
-TIMER_UT
-INIT_59A_UT
-RESET_59A_UT
-FILL_WRONGSWI_UT
-SWI_INT_UT
-WRONG_SWI_UT

4. Ladeprozedur

Das Testprogramm TEST09 wird im Verbund mit dem Leitprogramm LACS von Minidiskette gebootet (Kommandobeispiele siehe Pkt.4.1 - 4.3).

4.1. Laden mit Angabe des Geraetenamens

.B :Fn:TEST<CR> (n = Laufwerksnr.)
Nach Erreichen des Monitorbedienzustandes:
.B :Fn:G09<CR> (n = Laufwerksnr.)
Anschliessend verweilt das Programm im Generierhalt.

4.2. Laden ohne Angabe des Geraetenamens

.B TEST<CR>
Nach Erreichen des Monitorbedienzustandes:
.B G09<CR>
Bei letzterem Kommando werden TEST bzw. G09 auf allen Laufwerken gesucht,beginnend mit F0. Anschliessend verweilt das Programm im Generierhalt

4.3. Laden durch Kommandoeingabe im ACT (A 7100 - Confidence - Test)

Nach Netzeinschalten oder nach Druecken der Taste RESET besteht die verkuerzte Lademoeglichkeit von LACS durch Eingabe des Zeichens "T" innerhalb des PIC - Tests von ACT.

5. Generierinformationen

Der Bediener hat die Moeglichkeit, die Ausgabe von Durchlauf- und Fehlermeldungen durch Eingabe von Generierinformationen zu steuern. Dazu muss der Programmablauf unterbrochen werden (z.B. durch CTRL-C), bzw. das Programm muss sich im Generierhalt befinden (s. Pkt. 4.1./ 4.2.). Mittels des Monitorkommandos SW kann dann eine Aenderung der Generierung vorgenommen werden.

5.1. Generierung der Meldungen

TEST09 laeuft unter der Regie des Leitprogrammodules LACS. Deshalb werden Meldungen ueber den Programmablauf durch die Wort-Variablen TDERRONLY (Adr. DATA_LACS:0000) und TDDEBUG (Adr. DATA_LACS:0002) gesteuert. (s. Pkt. 5.1 der Beschreibung des LACS-Moduls)

6. Startprozedur

6.1. Start des Testprogramms

Das Testprogramm TEST09 laeuft unter der Regie von LACS. Nach dem Laden (siehe Pkt. 4.) erfolgt der Start:

- Aus dem Generierhalt durch das Monitorkommando
  .G<CR>
- Restart erfolgt durch das Monitorkommando
  .G <CODE_LACS>:40<CR>.

6.2. Bedieneraktionen

Der Bediener hat folgende Eingriffsmoeglichkeiten waehrend des Programmablaufs:

- CTRL-S Unterbrechung der Ausgabe, Fortsetzung mit CTRL-Q
- CTRL-Q Fortsetzung einer unterbrochenen Ausgabe
- CTRL-P Hardcopy der Bildschirmausgabe auf angeschlossenen Drucker, erneutes CTRL-P setzt die Hardcopy-Funktion zurueck
- CTRL-C Uebergang in den Bedienzustand des Monitors

Die Wirkung dieser Kommandos erfolgt erst unmittelbar vor einer Ausgabe einer Meldung auf den Bildschirm.

7. Einstellungsmoeglichkeiten

Gemaess Standardeinstellung wird der Test zyklisch wiederholt. Pro Zyklus wird das Testprogramm einmal abgearbeitet.

8. Programmbeschreibung

8.1. Programmablauf

Vor dem 1. Durchlauf erfolgt die Ausgabe der Programmnummern, Modulnamen und der Zeichnungs- und Ausgabenummern aller beteiligten Module. Sie hat fuer TEST09 folgendes Aussehen (siehe Punkt 8.2.1.):

```
      TEST09  yy-mm-dd  1.56.703010.5/67-aa-56ACnnn
  mit         yy-mm-dd = Erstellungsdatum
                   aa = Ausgabenummer
```

nnn = Nr. der Aenderungsmitteilung
Anschliessend erfolgt der eigentliche Test.Seine Laufzeit betraegt kleiner 1 sec.
Waehrend der Testdurchfuehrung koennen entsprechend den Erfordernissen die Generierinformationen umgestellt werden.
Die Rueckkehr zum Bedienmodul erfolgt im fehlerfreien Fall durch einen

```
JMP     RETURN_PASS
```

Im Fehlerfall wird der Test wie folgt beendet:
1.Ausgabe der Fehlermeldung:

```
TEST09 ERROR_CODE:  <XX> H

mit <XX> : Fehlercode  (0A000H ... 0A020H
                        0B000H ... 0B05FH,
                        0C000H ... 0C037H,
                        0E000H ... 0E022H: s. Anhaenge)
```

2.Rueecksprung zum Bedienmodul mit

```
JMP     RETURN_FAIL
```

Es erfolgt daraufhin automatisch eine Ausgabe durch die Routine DISPLAY_RESULTS. Die Parameter werden vor dem JMP in folgenden Registern uebergeben:
- BX: Offset der Adresse der Testspeicherzelle
- DX: Sollwert
- AX: Istwert
- ES: Segment der Adresse der Testspeicherzelle

Zum Abbruch des Tests s.Pkt.10.

8.2. Anschlussbedingungen

8.2.1. Struktur des Quellprogramms

Der Testmodul TEST09 ist folgendermassen aufgebaut:
1. ¤TITLE(TEST09 Zeichnungsnr.-Ausgabenr.-Aend.Mitt.Nr.)
2. Programmbeschreibung
3. NAME TEST09
4. Es folgen die EXTRN-und PUBLIC-Erklaerungen.
5. Es folgt das Datensegment mit der Bezeichnung DATA_TEST09.
6. Es folgt das Codesegment mit der Bezeichnung CODE_TEST09.

Das Datensegment DATA_TEST09 enthaelt die folgenden Variablen:
- T09_CS Codesegment des Testprogrammes
- T09_IP Offset der Programmzeile, in deren unmittelbarer Naehe der Fehler geortet wurde

Es folgen Variable zum Retten von Registern:
- T09_AX Enthaelt i.a. den Ist-Wert
- T09_BX Enthaelt Offset der Testzelle bei String-Befehlen
- T09_CX
- T09_DX Enthaelt i.a. den Soll-Wert
- T09_SI
- T09_DI

- T09_ES Enthaelt Segment der Testzelle bei String-Befehlen

Weitere Variable sind:
- T09_BAD_CODE Enthaelt den Fehlercode
- T09_VERS_SWITCH Datenbyte, dass auf 1 gesetzt die Bildschirmausgabe der Version bewirkt
- T09_ERROR Textrahmen fuer Fehlermitteilung

Im Segment CODE_TEST09 stehen am Anfang folgende Zeilen:

```
VERSION_TEST09  DB  '      TEST09  '
                DB  'yy-mm-dd  '
                DB  'Zeichn.nr. Ausg.nr.-Aend.Mitt.Nr.',0AH,0DH,0
                ORG  40H
```

dabei sind:
yy-mm-dd Datum der letzten Aenderung

Dieser Text wird nach dem Start des Testmoduls in Abhaengigkeit von TDDEBUG (s.Pkt.5.1) ausgegeben. Die naechste Zeile nach dem ORG-Befehl enthaelt die Ansprungstelle des Testmoduls mit der Marke TEST09.

8.2.2. Stack- und Registernutzung

Vom Testmodul TEST09 wird der Stack des Leitprogramms genutzt, d.h. beim Start von TEST09 wird kein SS sondern nur DS initialisiert.

8.2.3. Protokollsteuerung

Die Ausgaben sind je nach den Werten, die die Variablen TDERRONLY und TDDEBUG enthalten, abgestuft unterdrueckbar (s.Pkt.5.1.). Es gelten die Richtlinien gemaess Punkt 8.2.3 der LACS-Beschreibung.

8.2.4. Nutzbare TEST09-Routinen

TEST09 enthaelt keine Routinen, die fuer andere Nutzer verfuegbar sind.

8.2.5. Nutzbarkeit anderer Routinen

Siehe Punkt 3.4.

9. Fehler

9.1. Fehlerausgaben

Das Programm TEST09 gibt im Fehlerfall einen Text aus, welcher das fehlerhafte Testobjekt kennzeichnet. Dann wird das Leitprogramm angesprungen, welches durch INT 3 den Monitor-Bedienzustand

erreicht (Standardgenerierung). Vom Leitprogramm werden die Standardfehlermeldungen ausgegeben (s. LACS-Beschreibung).
Der Bediener kann mittels Monitorkommandos den Fehler eingrenzen, verdichten oder ausblenden (siehe Punkt 5.).

9.2. Fortsetzung nach Fehler

Die Wiederholung des Tests erfolgt durch das Monitorkommando
G<CODE_LACS>:40<CR>

10. Verschiedenes

10.1. Programmabbruchmoeglichkeiten

Der Programmablauf kann durch die BREAK-Taste unterbrochen werden. Weiterstart mit G<CR> fuehrt zu einem Fehler. Restart auf Adresse <CODE_LACS>:40 moeglich. Ein geordneter Abbruch kann durch Eingabe von CTRL-C erfolgen, damit erfolgt der Uebergang in den Monitorbedienzustand vor der naechsten Ein/Ausgabe.

10.2. Kontrolle der Programmdurchfuehrung

Die Durchfuehrung des Testprogramms wird durch die folgende LACS-Meldung auf dem Bildschirm bestaetigt (Standardfall):

```
TEST09: CPU TEST                    "PASSED"
```

Fehler- und Durchlaufzaehler werden in folgender Form ausgegeben:

```
ERRORS / CYCLES:   xxxx / yyyy
```

Danach beginnt ein neuer Programmzyklus.
Nach Unterbrechung des Programms kann die Durchfuehrungszahl des Testprogramms auch aus der Zelle

```
<CONST_LACS>:08A
```

ermittelt werden (Monitorkommando DW).
Die Anzahl der aufgetretenen Fehlermeldungen kann man der Zelle

```
<CONST_LACS>:88
```

entnehmen.

Anhang A:

Liste der Fehlercode fuer CPU TEST

Fehlercode	Fehler bei Ausfuehrung des Befehls	Ansprungmarke
A000H	JNE	J_0
A001H	JE	J_2
A002H	JNP	J_1
A003H	JP	J_3
A004H	JNB	J_4
A005H	JB	J_6
A006H	JNS	J_5
A007H	JS	J_7
A008H	JNO	J_8
A009H	JO	J_A
A00AH	JNBE	J_9
A00BH	JBE	J_B
A00CH	JBE	J_D
A00DH	JBE	J_C
A00EH	JNLE	J_E
A00FH	JLE	J_10
A010H	JLE	J_F
A011H	JLE	J_11
A012H	JLE	J_13
A013H	JLE	J_12
A014H	JLE	J_15
A015H	JNLE	J_14
A016H	JNL	J_16
A017H	JL	J_18
A018H	JL	J_17
A019H	JNL	J_19
A01AH	JNB	J_1B
A01BH	JB	J_1A
A01CH	JCXZ	J_1C
A01DH	JNB,CMC	J_1D
A01EH	CMP	J_1E
A01FH	"	J_1EA
A020H	MOV	J_1EB
B000H	AND	EX_0
B001H	OR	EX_1
B002H	—	—
B003H	NOT	EX_2
B004H	NEG	EX_3
B005H	INC,DEC,LOOP	EX_4
B006H	" " "	EX_4A
B007H	" " "	EX_4B
B008H	INC,DEC,LOOPNZ, LAHF	EX_5
B009H	" " " "	EX_5A
B00AH	" " " "	EX_5B
B00BH	INC,DEC,LOOPZ, LAHF	EX_6

Fehlercode	Fehler bei Ausfuehrung des Befehls	Ansprungmarke
B00CH	" " " "	EX_6A
B00DH	" " " "	EX_6B
B00EH	" " " "	EX_6C
B00FH	LEA	EX_8
B010H	POPF	EX_9
B011H	"	EX_9
B012H	"	EX_9A
B013H	"	EX_9A
B014H	SAHF,LAHF	EX_A
B015H	—	—
B016H	RCL	EX_B
B017H	"	EX_BA
B018H	"	EX_BB
B019H	"	EX_BC
B01AH	"	EX_BD
B01BH	"	EX_BE
B01CH	ROL	EX_C
B01DH	"	EX_CA
B01EH	"	EX_CB
B01FH	"	EX_CC
B020H	SAL	EX_D
B021H	"	EX_DA
B022H	"	EX_DB
B023H	"	EX_DC
B024H	"	EX_DD
B025H	RCR	EX_E
B026H	"	EX_EA
B027H	"	EX_EB
B028H	"	EX_EC
B029H	"	EX_ED
B02AH	"	EX_EE
B02BH	ROR	EX_F
B02CH	"	EX_FA
B02DH	"	EX_FB
B02EH	"	EX_FC
B02FH	SHR	EX_10
B030H	"	EX_10A
B031H	"	EX_10B
B032H	"	EX_10C
B033H	"	EX_10D
B034H	SAR	EX_11
B035H	"	EX_11A
B036H	"	EX_11B
B037H	"	EX_11C
B038H	"	EX_11D
B039H	SBB	EX_12
B03AH	DAA	EX_13
B03BH	"	EX_13A
B03CH	"	EX_13B
B03DH	"	EX_13C
B03EH	"	EX_13D

Fehlercode	Fehler bei Ausfuehrung des Befehls		Ansprungmarke
B03FH	"		EX_13E
B040H	"		EX_13F
B041H	"		EX_13AA
B042H	DAS		EX_14
B043H	"		EX_14A
B044H	"		EX_14B
B045H	"		EX_14C
B046H	"		EX_14D
B047H	"		EX_14E
B048H	"		EX_14F
B049H	"		EX_14AA
B04AH	TEST		EX_15
B04BH	"		EX_15A
B04CH	"		EX_15B
B04DH	"		EX_15C
B04EH	"		EX_15D
B04FH	"		EX_15E
B050H	ADD		EX_16
B051H	"		EX_16A
B052H	"		EX_16B
B053H	SUB		EX_17
B054H	"		EX_17A
B055H	"		EX_17B
B056H	LODS		EX_18
B057H	"		EX_18A
B058H	"		EX_18B
B059H	"		EX_18C
B05AH	CALL/RET		EX_19
B05BH	POP/PUSH/XCHG		EX_20
B05CH	" " "		EX_20B
B05DH	JMP		EX_21
B05EH	"		EX_21B
B05FH	XOR		EX_22
C000H	AND	(B)	EX_30
C001H	"	(B)	EX_30A
C002H	OR	(B)	EX_31
C003H	NOT	(B)	EX_32
C004H	NEG	(B)	EX_33
C005H	INC/DEC/LOOP	(B)	EX_34
C006H	" " "	(B)	EX_34A
C007H	" " "	(B)	EX_34B
C008H	RCL	(B)	EX_35
C009H	"	(B)	EX_35A
C00AH	"	(B)	EX_35B
C00BH	"	(B)	EX_35C
C00CH	ROL	(B)	EX_36
C00DH	"	(B)	EX_36A
C00EH	SAL	(B)	EX_37
C00FH	"	(B)	EX_37A
C010H	"	(B)	EX_37B
C011H	RCR	(B)	EX_38

Fehlercode	Fehler bei Ausfuehrung des Befehls		Ansprungmarke
C012H	"	(B)	EX_38A
C013H	"	(B)	EX_38B
C014H	"	(B)	EX_38C
C015H	ROR	(B)	EX_39
C016H	"	(B)	EX_39A
C017H	SHR	(B)	EX_3A
C018H	"	(B)	EX_3AA
C019H	"	(B)	EX_3AB
C01AH	SAR	(B)	EX_3B
C01BH	"	(B)	EX_3BA
C01CH	"	(B)	EX_3BB
C01DH	SBB	(B)	EX_3C
C01EH	TEST	(B)	EX_3D
C01FH	"	(B)	EX_3DA
C020H	"	(B)	EX_3DB
C021H	"	(B)	EX_3DC
C022H	"	(B)	EX_3DD
C023H	"	(B)	EX_3DE
C024H	ADD	(B)	EX_3E
C025H	"	(B)	EX_3EA
C026H	"	(B)	EX_3EB
C027H	SUB	(B)	EX_3F
C028H	"	(B)	EX_3FA
C029H	"	(B)	EX_3FB
C02AH	XCHG	(B)	EX_40
C02BH	"	(B)	EX_40A
C02CH	XOR	(B)	EX_41
C02DH	"	(B)	EX_41A
C02EH	ADC	(B)	EX_42
C02FH	CMP	(B)	EX_43
C030H	"	(B)	EX_43A
C031H	MOV	(B)	EX_43B
C032H	DIV	(B)	EX_44
C033H	IDIV	(B)	EX_45
C034H	"	(B)	EX_45A
C035H	MUL	(B)	EX_46
C036H	IMUL	(B)	EX_47
C037H	"	(B)	EX_47A
E000H	AAA		CPU_0
E001H	AAD		CPU_1
E002H	AAM		CPU_2
E003H	AAS		CPU_3
E004H	ADC		CPU_4
E005H	CBW		CPU_5
E006H	CWD		CPU_6
E007H	DIV		CPU_7
E008H	IDIV		CPU_8
E009H	MUL		CPU_9
E00AH	IMUL		CPU_A
E010H	STOS/SCAS	(U)	CPU_10
E011H	MOVS/CMPS	(U)	CPU_11

Fehlercode	Fehler bei Ausfuehrung des Befehls		Ansprungmarke
E012H	CMPS	(U)	CPU_12
E013H	STOS/SCAS		CPU_13
E014H	MOVS/CMPS	(D)	CPU_14
E015H	CMPS	(D)	CPU_15
E016H	STOS/SCAS	(B)	CPU_16
E017H	MOVS/CMPS	(U,B)	CPU_17
E018H	CMPS	(U,B)	CPU_18
E019H	STOS/SCAS	(B)	CPU_19
E01AH	MOVS/CMPS	(D,B)	CPU_1A
E01BH	CMPS	(D,B)	CPU_1B
E020H	XLAT		CPU_20
E021H	LDS		CPU_21
E022H	LES		CPU_22

Anmerkung:
(U) - UP: Stringbefehl aufwaerts
(D) - DOWN: Stringbefehl abwaerts
(B) - BYTE: Byte-Verarbeitung
Ohne Kennzeichen: Word-Verarbeitung

Anhang B:

Alphabetische Befehlsliste mit Fehlercode

Fehler bei Ausfuehrung des Befehls	Fehlercode	Ansprungmarke
AAA	E000H	CPU_0
AAD	E001H	CPU_1
AAM	E002H	CPU_2
AAS	E003H	CPU_3
ADC	E004H	CPU_4
ADD	B050H	EX_16
	B051H	EX_16A
	B052H	EX_16B
AND	B000H	EX_0
CALL	B05AH	EX_19
CBW	E005H	CPU_5
CLC	A01AH	J_1B
CLD	E010H	CPU_10
	E011H	CPU_11
	E012H	CPU_12
	E013H	CPU_13
	E014H	CPU_14
	E016H	CPU_16
	E019H	CPU_19
CLI	(I)	
CMC	A01CH	J_1D
CMP	A01EH	J_1E
	A01FH	J_1EA
CMPS	E011H	CPU_11

Fehler bei Ausfuehrung des Befehls	Fehlercode	Ansprungmarke
	E012H	CPU_12
	E014H	CPU_14
	E015H	CPU_15
CWD	E006H	CPU_6
DAA	B03AH	EX_13
	B03BH	EX_13A
	B03CH	EX_13B
	B03DH	EX_13C
	B03EH	EX_13D
	B03FH	EX_13E
	B040H	EX_13F
	B041H	EX_13AA
DAS	B042H	EX_14
	B043H	EX_14A
	B044H	EX_14B
	B045H	EX_14C
	B046H	EX_14D
	B047H	EX_14E
	B048H	EX_14F
	B049H	EX_14AA
DEC	B005H	EX_4
	B006H	EX_4A
	B007H	EX_4B
	B008H	EX_5
	B009H	EX_5A
	B00AH	EX_5B
	B00BH	EX_6
	B00CH	EX_6A
	B00DH	EX_6B
	B00EH	EX_6C
DIV	E007H	CPU_7
ESC	----	----
HLT	----	----
IDIV	E008H	CPU_8
IMUL	E00AH	CPU_A
IN	----	----
INC	B005H	EX_4
	B006H	EX_4A
	B007H	EX_4B
	B008H	EX_5
	B009H	EX_5A
	B00AH	EX_5B
	B00BH	EX_6
	B00CH	EX_6A
	B00DH	EX_6B
	B00EH	EX_6C
INT	siehe TEST0C	
INTO	" "	
IRET	" "	
JA/JNBE	A00AH	J_9
JAE/JNB	A004H	J_4

Fehler bei Ausfuehrung des Befehls	Fehlercode	Ansprungmarke
	A01AH	J_1B
	A01CH	J_1C
JB/JNAE	A005H	J_6
	A01BH	J_1A
JBE/JNA	A00BH	J_B
	A00CH	J_D
	A00DH	J_C
JCXZ	A01DH	J_1C
JE/JZ	A001H	J_2
JG/JNLE	A00EH	J_E
	A015H	J_14
JGE/JNL	A016H	J_16
	A019H	J_19
JL/JNGE	A017H	J_18
	A018H	J_17
JLE/JNG	A00FH	J_10
	A010H	J_F
	A011H	J_11
	A012H	J_13
	A013H	J_12
	A014H	J_15
JMP	B05DH	EX_21
	B05EH	EX_21B
JNA/JBE	A00BH	J_B
	A00CH	J_D
	A00DH	J_C
JNAE/JB	A005H	J_6
	A01BH	J_1A
JNB/JAE	A004H	J_4
	A01AH	J_1B
	A01CH	J_1D
JNBE/JA	A00AH	J_9
JNE/JNZ	A000H	J_0
JNG/JLE	A00FH	J_10
	A010H	J_F
	A011H	J_11
	A012H	J_13
	A013H	J_12
	A014H	J_15
JNGE/JL	A017H	J_18
	A018H	J_17
JNL/JGE	A016H	J_16
	A019H	J_19
JNLE/JG	A00EH	J_E
	A015H	J_14
JNO	A008H	J_8
JNP/JPO	A002H	J_1
JNS	A006H	J_5
JNZ/JNE	A000H	J_0
JO	A009H	J_A
JP/JPE	A003H	J_3

Fehler bei Ausfuehrung des Befehls	Fehlercode	Ansprungmarke
	"	"
JPE/JE	A002H	J_1
JPO/JNP	A007H	J_7
JS	A001H	J_2
JZ/JE	B008H	EX_5
LAHF	B00BH	EX_6
	B014H	EX_A
LDS	E021H	CPU_21
LEA	B00FH	EX_8
LES	E022H	CPU_22
LOCK	———	———
LODS	B056H	EX_18
	B057H	EX_18A
	B058H	EX_18B
	B059H	EX_18C
LOOP	B005H	EX_4
	B006H	EX_4A
	B007H	EX_4B
LOOPE/LOOPZ	B00BH	EX_6
	B00CH	EX_6A
	B00DH	EX_6B
	B00EH	EX_6C
LOOPNE/LOOPNZ	B008H	EX_5
	B009H	EX_5A
	B00AH	EX_5B
MOV	A020H	J_1EB
MOVS	E011H	CPU_11
	E014H	CPU_14
MUL	E009H	CPU_9
NEG	B004H	EX_3
NOP	(I)	
NOT	B003H	EX_2
OR	B001H	EX_1
OUT	———	———
POP	B05BH	EX_20
	B05CH	EX_20B
POPF	B010H	EX_9
	B011H	EX_9
	B012H	EX_9A
	B013H	EX_9A
PUSH	B05BH	EX_20
	B05CH	EX_20B
PUSHF	B010H	EX_9
	B011H	EX_9
	B012H	EX_9A
	B013H	EX_9A
RCL	B016H	EX_B
	B017H	EX_BA
	B018H	EX_BB
	B019H	EX_BC
	B01AH	EX_BD

Fehler bei Ausfuehrung des Befehls	Fehlercode	Ansprungmarke
RCL	B01BH	EX_BE
RCR	B025H	EX_E
	B026H	EX_EA
	B027H	EX_EB
	B028H	EX_EC
	B029H	EX_ED
	B02AH	EX_EE
REPE	E010H	CPU_10
	E011H	CPU_11
	E012H	CPU_12
	E013H	CPU_13
	E014H	CPU_14
	E015H	CPU_15
	E016H	CPU_16
	E017H	CPU_17
	E018H	CPU_18
	E019H	CPU_19
	E01AH	CPU_1A
	E01BH	CPU_1B
RET	B05AH	EX_19
ROL	B01CH	EX_C
	B01DH	EX_CA
	B01EH	EX_CB
	B01FH	EX_CC
ROR	B02BH	EX_F
	B02CH	EX_FA
	B02DH	EX_FB
	B02EH	EX_FC
SAHF	B014H	EX_A
SAL/SHL	B020H	EX_D
	B021H	EX_DA
	B022H	EX_DB
	B023H	EX_DC
	B024H	EX_DD
SAR	B034H	EX_11
	B035H	EX_11A
	B036H	EX_11B
	B037H	EX_11C
	B038H	EX_11D
SBB	B039H	EX_12
SCAS	E010H	CPU_10
	E013H	CPU_13
	E019H	CPU_19
SHL/SAL	B020H	EX_D
	B021H	EX_DA
	B022H	EX_DB
	B023H	EX_DC
	B024H	EX_DD
SHR	B02FH	EX_10
	B030H	EX_10A
	B031H	EX_10B

Fehler bei Ausfuehrung des Befehls		Fehlercode	Ansprungmarke
SHR		B032H	EX_10C
		B033H	EX_10D
STC		A01BH	J_1A
STD		E010H	CPU_10
		E011H	CPU_11
		E013H	CPU_13
		E014H	CPU_14
		E015H	CPU_15
		E016H	CPU_16
		E017H	CPU_17
		E019H	CPU_19
		E01AH	CPU_1A
		E01BH	CPU_1B
STI		(I)	
STOS		E010H	CPU_10
		E013H	CPU_13
		E019H	CPU_19
SUB		B053H	EX_17
		B054H	EX_17A
		B055H	EX_17B
TEST		B04AH	EX_15
		B04BH	EX_15A
		B04CH	EX_15B
		B04DH	EX_15C
		B04EH	EX_15D
		B04FH	EX_15E
WAIT		----	----
XCHG		B05BH	EX_20
		B05CH	EX_20B
XLAT		E020H	CPU_20
XOR		B05FH	EX_22
ADD	(B)	C024H	EX_3E
		C025H	EX_3EA
		C026H	EX_3EB
AND	(B)	C000H	EX_30
		C001H	EX_30A
CMP	(B)	C02FH	EX_43
		C030H	EX_43A
CMPS	(B)	E017H	CPU_17
		E018H	CPU_18
		E01AH	CPU_1A
		E01BH	CPU_1B
DEC	(B)	C005H	EX_34
		C006H	EX_34A
		C007H	EX_34B
DIV	(B)	C032H	EX_44
IDIV	(B)	C033H	EX_45
IMUL	(B)	C036H	EX_47
		C037H	EX_47A
INC	(B)	C005H	EX_34
		C006H	EX_34A

Fehler bei Ausfuehrung des Befehls	Fehlercode	Ansprungmarke
INC (B)	C007H	EX_34B
MOV (B)	C031H	EX_43B
MOVS (B)	E017H	CPU_17
	E01AH	CPU_1A
MUL (B)	C035H	EX_46
NEG (B)	C004H	EX_33
NOT (B)	C003H	EX_32
OR (B)	C002H	EX_31
RCL (B)	C008H	EX_35
	C009H	EX_35A
	C00AH	EX_35B
	C00BH	EX_35C
RCR (B)	C011H	EX_38
	C012H	EX_38A
	C013H	EX_38B
	C014H	EX_38C
ROL (B)	C00CH	EX_36
	C00DH	EX_36A
ROR (B)	C015H	EX_39
	C016H	EX_39A
SAL/SHL (B)	C00EH	EX_37
	C00FH	EX_37A
	C010H	EX_37B
SAR (B)	C01AH	EX_3B
	C01BH	EX_3BA
	C01CH	EX_3BB
SBB (B)	C01DH	EX_3C
SCAS (B)	E016H	CPU_16
SHL/SAL (B)	C00EH	EX_37
	C00FH	EX_37A
	C010H	EX_37B
SHR (B)	C017H	EX_3A
	C018H	EX_3AA
	C019H	EX_3AB
STOS (B)	E016H	CPU_16
SUB (B)	C027H	EX_3F
	C028H	EX_3FA
	C029H	EX_3FB
TEST (B)	C01EH	EX_3D
	C01FH	EX_3DA
	C020H	EX_3DB
	C021H	EX_3DC
	C022H	EX_3DD
	C023H	EX_3DE
XCHG (B)	C02AH	EX_40
	C02BH	EX_40A
XOR (B)	C02CH	EX_41
	C02DH	EX_41A

Anmerkung:	(I):	Befehle werden indirekt in verschiedenen Abschnitten der PSU getestet
	(B) - BYTE:	Byte-Verarbeitung
	Ohne Kennzeichen:	Word-Verarbeitung

1. TEST0A, Timer Count Test

2. Funktion des Programms

Der Timer Count Test prueft die Zaehlgenauigkeit des programmierbaren Schaltkreises 8253A (Zeitgeber PIT) auf der ZVE des A 7100. Der Test besteht aus zwei Routinen.

- ROUTINE_01:
  Counter_0 wird programmiert im Modus 0 (Interrupt, wenn Zaehlerstand 0 erreicht ist). Zu einem bestimmten Zeitpunkt nach Starten von Counter_0 wird dieser gelesen (read on fly) und mit dem erwarteten Zaehlerstand verglichen. Wenn ein Fehler auftritt, werden sofort die Nummer des Counters sowie erwarteter und gelesener Zaehlerstand angezeigt.
- ROUTINE_02:
  Counter_1 wird programmiert im Modus 0 (Interrupt, wenn Zaehlerstand 0 erreicht ist). Zu einem bestimmten Zeitpunkt nach Starten von Counter_1 wird dieser gelesen (read on fly) und mit dem erwarteten Zaehlerstand verglichen. Wenn ein Fehler auftritt, werden sofort die Nummer des Counters sowie erwarteter und gelesener Zaehlerstand angezeigt.

3. Voraussetzungen

Im Folgenden werden Speicheradressen in der Form Segment:Offset angegeben. Die Zuordnung von Segmentnamen und absoluten Adressen ist aus der MP2-Liste zu entnehmen.

3.1. Geraeteausruestung

Es ist die Grundausruestung des Rechners A 7100 erforderlich. Die Ein/Ausgabe richtet sich nach der im Monitor generierten Geraetekonfiguration.

3.2. Speicherbedarf

ca. 1 KB fuer TEST0A.OBJ
Die absoluten Adressen der durch den Locator zugewiesenen Speicherbereiche sind anhand der MP2-Liste zu berechnen.

3.3. Test zusaetzlicher Baugruppen

TEST0A erfordert u.a. fehlerfrei arbeitende Speicher (ZPS oder OPS). Insbesondere wird das richtige Zeitverhalten von ZPS oder OPS geprueft.

3.4. Nutzung anderer Programmodule

Der Testmodul TESTOA.OBJ ist mit dem Bibliotheksmodul LIBRAR.OBJ, dem Leitprogrammodul LACS.OBJ und der Bibliothek LACS.LIB verbunden.

Aus LACS.OBJ werden genutzt:

Anspruenge:	- RETURN_PASS
	- RETURN_FAIL
Variable:	- TDERRONLY
	- TDDEBUG
Routinen:	- ASCII_CONV
	- TDMASKEDMESSAGE
	- TDDISPLAY

Aus LIBRAR.OBJ werden genutzt:

Routinen:	- SAVE_INTR_MON
	- REST_INTR_MON
	- RESET_59A
	- CHECK_ROUT
	- CHECK_ITERA
	- CHECK_ERROR

4. Ladeprozedur fuer autonomen Ablauf

Das Testprogramm wird im Verbund mit dem Leitprogramm LACS von Minidiskette gebootet (Kommandobeispiele siehe Punkt 4.1. - 4.3.).

4.1. Laden mit Angabe des Geraetenamens

```
.B :Fn:TEST<CR>          (n = Laufwerksnr.)
```
Nach Erreichen des Monitorbedienzustandes:
```
.B :Fn:GOA<CR>           (n = Laufwerksnr.)
```
Anschliessend verweilt das Programm im Generierhalt.

4.2. Laden ohne Angabe des Geraetenamens

```
.B TEST<CR>
```
Nach Erreichen des Monitorbedienzustandes:
```
.B GOA<CR>
```
Bei letzterem Kommando werden TEST bzw. GOA auf allen Laufwerken gesucht, beginnend mit F0.
Anschliessend verweilt das Programm im Generierhalt.

4.3. Laden durch Kommandoeingabe im ACT (A 7100 - Confidence - Test)

Nach Netzeinschalten oder nach Druecken der Taste RESET besteht die verkuerzte Lademoeglichkeit von LACS durch Eingabe des Zei-

chens "T" innerhalb des PIC - Tests von ACT.
Nach Erreichen des Monitorbedienzustandes weiter wie oben.

5. Generierinformationen

Der Bediener hat die Moeglichkeit, die Ausgabe von Durchlauf- und Fehlermeldungen sowie allgemeine Mitteilungen ueber den Zustand des Prueflings durch Eingabe von Generierinformationen zu steuern. (siehe Punkt 5.1.).
Ausserdem besteht die Moeglichkeit, zur Fehlereingrenzung, Fehlerverdichtung oder fuer Messzwecke den normalen Ablauf des Testprogrammes zu veraendern (siehe Punkt 5.2.).
Dazu muss der Programmablauf unterbrochen werden (z.B. durch CTRL-C). Mittels des Monitorkommandos SW kann dann eine Aenderung der Generierung vorgenommen werden.

5.1. Generierung von Meldungen und Mitteilungen

TESTOA laeuft unter der Regie des Leitprogrammodules LACS. Deshalb werden Meldungen ueber den Programmablauf durch die Variablen TDERRONLY (Adr. DATA_LACS:0000) und TDDEBUG (Adr. DATA_LACS:0002) gesteuert. (siehe Punkt 5.1. der Beschreibung des Leitprogrammes LACS).

5.2. Generierung spezieller Testablaeufe

Spezielle Testablaeufe werden gesteuert durch die Variablen
-ROUT_TESTOA (Adr. DATA_TESTOA:0000) - Routinesteuerwort
Bit 0 = ROUTINE_01, Bit 1 = ROUTINE_02
-ERR_MAX (Adr. DATA_TESTOA:000A) - Fehlerzaehler
Gestattet die Vorwahl einer maximalen Fehlerzahl. Das Testprogramm behaelt im Fehlerfalle solange die Regie, bis die vorgewaehlte Fehlerzahl erreicht ist. Erst dann erfolgt die Rueckkehr zum Leitprogramm LACS und damit Uebergang in den Bedienzustand des Monitors.
Voraussetzung: TDDEBUG Bit 1 gesetzt, ITERA_0A muss mindestens so gross wie ERR_MAX sein.
-ITERA_0A (Adr. DATA_TESTOA:000C) - Iterationszaehler
Gestattet die Veraenderung der Anzahl der Durchlaeufe von TESTOA, bevor die Regie an LACS abgegeben wird.
-FIRST_0A (Adr. DATA_TESTOA:000E) - Versionsausgabe
Wenn FIRST_0A=FF, dann Ausgabe der Versionsnummer, sonst nicht. (siehe Punkt 8.1.)

6. Startprozedur

6.1. Start des Testprogramms

Das Testprogramm TESTOA laeuft unter der Regie von LACS. Nach dem Laden (siehe Punkt 4.) erfolgt der Start:

- Aus dem Generierhalt durch das Monitorkommando
  .G<CR>
- Restart erfolgt durch das Monitorkommando
  .G <CODE_LACS>:40<CR>.

6.2. Bedieneraktionen

Der Bediener hat folgende Eingriffsmoeglichkeiten waehrend des Programmablaufs:
- CTRL-S = Unterbrechung der Ausgabe, Fortsetzung mit CTRL-Q.
- CTRL-Q = Fortsetzung einer unterbrochenen Ausgabe.
- CTRL-P = Hardcopy der Bildschirmausgabe auf angeschlossenen Drucker
  Erneutes CTRL-P setzt die Hardcopy zurueck.
- CTRL-C = Uebergang in den Bedienzustand des Monitors.

Die Wirkung dieser Kommandos erfolgt erst unmittelbar vor Ausgabe einer Meldung auf den Bildschirm.
Der Uebergang in den Bedienzustand des Monitors ist auch moeglich durch Druecken der Taste BREAK auf der Tastatur. Damit ist dem Bediener ein Mittel gegeben, einen Fehler im Programmablauf von TESTOA zu erzeugen. Der Fehler entsteht, wenn der Monitor-Bedienzustand mit G<CR> verlassen wird und das unterbrochene Programm wieder angesprungen wird.

7. Einstellungsmoeglichkeiten

Die Standardeinstellung fuer TESTOA lautet:

- TDERRONLY:	0	;siehe Punkt 5.1. LACS-Beschreibung
- TDDEBUG:	3	; " " " " "
- ROUT_TESTOA:	3	;2 Routinen generiert
- ERR_MAX:	1	;INT 3 nach dem ersten Fehler
- ITERA_OA:	44D	;TESTOA laeuft 44 mal,
		;dann Regie an LACS

(siehe Punkte 5. und 8.2.1. der Programmbeschreibung TESTOA)

8. Programmbeschreibung

8.1. Programmablauf

Vor dem 1. Durchlauf erfolgt die Ausgabe der Programmnummern, Modulnamen und der Zeichnungs- und Ausgabenummern aller beteiligten Module. Sie hat fuer TESTOA folgendes Aussehen (siehe Punkt 8.2.1.):

```
TESTOA  yy-mm-dd  1.56.703011.3/67-aa-56ACnnn
       (yy-mm-dd = Erstellungsdatum
        aa       = Ausgabenummer
        56ACnnn  = Nr. der Aenderungsmitteilung).
```

Anschliessend beginnt die eigentliche Pruefung. Zuerst werden die testprogrammspezifischen Einstellungen (Routinesteuerwort, Fehlerzaehler und Iterationszaehler) geprueft. Falls in irgendeiner der drei Variablen der Wert 0 gefunden wurde, wird der Monitor-

Bedienzustand angesprungen und der Bediener zur Korrektur aufgefordert. Danach kann mit G<CR> weitergestartet werden. Dann werden ROUTINE_01 und ROUTINE_02 abgearbeitet, und zwar so oft wie in ITERA_0A angewiesen ist. Die Abarbeitungszeit von TESTOA betraegt bei Fehlerfreiheit ca. 3 sec.
Die Rueckkehr zum Leitprogramm erfolgt im fehlerfreien Fall durch einen

```
JMP     RETURN_PASS
```

Im Fehlerfall informiert das Testprogramm den Bediener durch Fehlerausschriften (beachte Punkt 5. und 9.) und uebergibt die Regie an das Leitprogramm durch einen

```
JMP     RETURN_FAIL
```

Abbruch des Testprogramms TESTOA siehe Punkt 10.

8.2. Anschlussbedingungen

8.2.1. Struktur des Quellprogramms

Der Testmodul TESTOA ist folgendermassen aufgebaut:
1. TITLE(TESTOA Zeichn.nr.-Ausgabenr.-Aend.Mitt.Nr.)
2. NAME TESTOA
3. Anschliessend folgt die Programmbeschreibung nach dem Muster der vorliegenden Beschreibung des Leitprogrammes LACS.
4. Es folgen die EXTRN-Erklaerungen.
5. Es folgt das Datensegment mit der Bezeichnung DATA_TESTOA.
6. Es folgt das Codesegment mit der Bezeichnung CODE_TESTOA.
7. PUBLIC-Erklaerungen stehen in dem Segment, in dem die betreffenden Symbole vorkommen, ganz am Anfang des Segments.

Das Datensegment DATA_TESTOA enthaelt folgende Variablen:

- ROUT_TESTOA	Wort	0000	Routinesteuerwort
- ERRONLY	"	0002	Entspricht TDERRONLY aus LACS
- DEBUG	"	0004	" TDDEBUG " "
- ROUT_TOPIC	"	0006	Zeigt auf die aktuelle ROUTINE
- ERROR_REG	"	0008	Enthaelt die aktuelle Fehlerzahl
- ERR_MAX	"	000A	Fehlerzaehler
- ITERA_0A	"	000C	Iterationszaehler
- FIRST_0A	Byte	000E	Versionsausgabe
- COUNTER_0_VALUE	Wort	000F	Enthaelt Stand von COUNTER_0
- COUNTER_1_VALUE	"	0011	" " " COUNTER_1

Anschliessend folgen Texte fuer Fehlerausschriften.

Im Segment CODE_TESTOA stehen nach den PUBLIC-Erklaerungen und den Symboldefinitionen folgende Daten:

```
VERSION_TESTOA  DB  '        TESTOA  '
                DB  'yy-mm-dd  '
                DB  'Zeichn.Nr.-Ausg.nr.-Aend.Mitt.Nr.',0AH,0DH,0
                ORG  40H
```

dabei sind:
yy-mm-dd Datum der letzten Aenderung

Dieser Text wird nach dem Start des Testmoduls in Abhaengigkeit von TDDEBUG (s.Pkt.5.1.) ausgegeben. Die naechste Zeile nach dem ORG-Befehl enthaelt die Ansprungstelle des Testmoduls mit der Marke TESTOA.

8.2.2. Stack- und Registernutzung

Vom Testmodul TESTOA wird der Stack des Leitprogrammes genutzt, d.h. beim Start von TESTOA wird kein SS sondern nur DS initialisier+.

8.2.3. Protokollsteuerung

Die Ausgaben sind je nach den Werten, die die Variablen TDERRONLY und TDDEBUG enthalten, abgestuft unterdrueckbar (s.Pkt.5.1.). Es gelten die Richtlinien gemaess Punkt 8.2.3. der LACS-Beschreibung.

8.2.4. Nutzbare TESTOA-Routinen

TESTOA enthaelt keine Routinen, die fuer andere Nutzer verfuegbar sind.

8.2.5. Nutzbarkeit anderer Routinen

Siehe Punkt 3.4.

9. Fehler

9.1. Fehlerausgaben

Jede Fehlerausschrift beginnt mit einem eingerueckten Pfeil
' ===>'.
Das Programm TESTOA gibt im Fehlerfall einen Text aus, welcher das fehlerhafte Testobjekt (COUNTER_0 oder COUNTER_1) kennzeichnet. Dann wird das Leitprogramm angesprungen, welches durch INT 3 den Monitor-Bedienzustand erreicht (Standardgenerierung). Der Bediener kann mittels Monitorkommandos den Fehler eingrenzen, verdichten oder ausblenden (siehe Punkt 5.).
Falls undefiniertes Verhalten oder 'Abstuerze' des Rechners den normalen Programmablauf verhindern oder falls keine Fehlerausschriften generiert sind, kann aus den Variablen COUNTER_0_VALUE (in ROUTINE_01) und COUNTER_1_VALUE (in ROUTINE_02) der fehlerhafte Zaehlerstand des gerade gepruefteten Counters ermittelt werden (s. Punkt 8.2.1.). Aus ROUT_TOPIC ist zu entnehmen, in welcher Routine sich das Testprogramm befindet.

9.2. Fortsetzung nach Fehler

In Abhaengigkeit von ERR_MAX (s. Punkt 5.) erfolgt nach einer (Standardgenerierung) oder mehreren Fehlerausschriften der Uebergang zum Leitprogramm. Dieses erhoeht seinen Fehlerzaehler und geht zum Monitor. Die Wiederholung des Tests erfolgt durch das Monitorkommando

G<CR>

10. Verschiedenes

10.1. Programmabbruchmoeglichkeiten

Der Programmablauf kann durch die BREAK-Taste unterbrochen werden (siehe Punkt 6.2.). Weiterstart mit G<CR> fuehrt zu einem Fehler. Restart auf Adresse CODE_LACS:40 moeglich. Ein geordneter Abbruch kann durch Eingabe von CTRL-C erfolgen, damit erfolgt der Uebergang in den Monitorbedienzustand vor der naechsten Ein/Ausgabe.

10.2. Kontrolle der Programmdurchfuehrung

Die Durchfuehrung des Testprogramms wird durch die folgende LACS-Meldung auf dem Bildschirm bestaetigt (Standardfall):

(Testprogr.Nr.) (Benennung des Testprogr.) "PASSED"

Fehler- und Durchlaufzaehler werden in folgender Form ausgegeben:

ERRORS / CYCLES: xxxx / yyyy

Danach beginnt ein neuer Programmzyklus.
Nach Unterbrechung des Programms kann die Durchfuehrungszahl des Testprogramms TESTOA auch aus der Zelle

CONST_LACS:0098

ermittelt werden (Monitorkommando DW).
Die Anzahl der aufgetretenen Fehlermeldungen kann man der Zelle

CONST_LACS:0096

entnehmen.