Kurzbeschreibung ZSID/SE

Einleitung

Das Programm ZSID/SE ist ein universeller, CP/M-kompatibler Debugger, vorzugsweise zum Test von Assemblerprogrammen, der auch als Monitor geeignet ist. Gegenüber dem Programm ZSID von Zilog (auch DU) existieren folgende Einschränkungen:

- Die Nutzung von SYM-Dateien beim Debugging ist nicht möglich (keine symbolische Adressierung).
- Ergänzungsprogramme (UTL-Dateien) können in die Arbeit von ZSID/SE nicht einbezogen werden.
- HEXA-Dateien werden beim Einlesen nicht in eine abarbeitbare Form konvertiert.
- Das zu testende Programm kann nicht mit ZSID/SE gemeinsam geladen werden.

Die Erweiterungen gegenüber ZSID von Zilog sind in der folgenden Kommando-Kurzdarstellung mit * gekennzeichnet:

A	Assembler	Direktassemblierung in den Speicher
B	Backtrace	* Anzeige der letzten 128 Befehle
C	Call	Abarbeitung von Unterprogrammen bis RET
D	Display	Anzeige eines Speicherbereich
E	Equal	* Vergleich von Speicherbereichen
F	Fill	Füllen eines Speicherbereiches
G	Go	Programmstart mit max. 2 Haltepunkten (RST 38H)
		* Zeitmessung möglich (n+0,1ms)
H	Hexa-Help	Umrechnung von Hexazahlen
I	Initial	Initialisierung von Datei-Ein- und -Ausgaben
J	-Input	* Einlesen von Bytes vom E/A-Port
K	Kontrolle	* Bildung und Kontrolle von CRC-Worten
L	List	Reassemblieren von Speicherbereichen
M	Move	Verschieben von Speicherbereichen
		* ohne Beachtung von Überlappungen
N		
O	Output	* Schreiben von Bytes auf den E/A-Port
P	Passpoints	Setzen von Durchlaufpunkten (RST38H)
Q	-Find String	* Suchen eines Strings im Speicherbereich
R	Read	Einlesen einer Datei entsprechend “I“
S	Substitute	Ersetzen von Bytes im Speicher
T	Trace	Schrittbetrieb mit Zustandsanzeige
U	Untrace	Schrittbetrieb ohne Zustandsanzeige
V		
W	Write	* Ausgabe einer Datei entsprechend “I“
X	-Examine	Anzeige und Modifikation von Registern,
		* von max. 3 Testzellen und der Laufzeit
Y		
Z	Zusatz	* Zusatzkommandos (spez. Nutzerkommandos)

Kommandoformat

Jedes Kommando besteht aus dem Kennbuchstaben, einer kommandoabhängigen Anzahl von Parametern und dem Zeichen <CR> (Taste ENTER, ET1, NL usw.) als Abschluß. Der Kennbuchstabe kann durch den Präfix <-> und/oder den Suffix <w> ergänzt werden. Trennzeichen zwischen den Parametern können <space> oder <,> sein.
Die Anzahl der Zeichen <space> vor und nach dem Kennbuchstaben ist nicht relevant. Zwischen den Parametern darf nur ein Trennzeichen stehen. Der erste Parameter eines Kommandos wird mit einem kommandospezifisch aktuellen Parameter belegt, wenn statt des ersten Parameters nur das Zeichen <,> zwischen Kennbuchstaben und zweiten Parameter eingegeben wird. Das Weglassen von

Folgeparametern führt entweder zu einer Sonderfunktion des Kommandos oder zur Belegung mit internen kommandospezifischen Konstanten.
Bei der Abarbeitung der Kommandos '"Q", "R", "W" sind weitere unformatierte Eingaben notwendig.

Parameterformat

Jeder Parameter wird durch eine Folge von Nutzsymbolen und ein Trennzeichen <SEP> (<space> oder <,> , <CR>) gebildet. Zulässige Folgen von Nutzsignalen sind:
- Steuerzeichen (nur in Kommando "X")
- Text (nur in Kommando "I")
- Zahl

Für Zahlen als Kommandoparameter sind folgende Formen der Darstellung zulässig:

hh	hexadezimal, 1 Byte (2 letzte Ziffern signifikant)
hhhh	hexadezimal, 2 Bytes (4 letzte Ziffern signifikant)
+hhhh	hexadezimal, Offset zur vorhergehenden Parametereingabe
-hhhh	negative Hexazahl
#d..d	dezimal (max. 255 (1 Byte) bzw. 65535 (2 Bytes))
+#d..d	dezimal, sonst wie +hhhh
-#d..d	negative Dezimalzahl
a...a	Folge von ASCII-Zeichen
^..^	Wert des Stackinhaltes auf der Position n-1 (n-Anzahl der Zeichen ^) entspr. dem geretteten Stackpointer

Kommandofehler

Fehler bei der Eingabe der Kommandos werden unspezifiziert nur durch das Zeichen <?> und anschließenden Wiederstart des Programms angezeigt. Mögliche Fehlerursachen sind:
- unzulässiger Kommandokennbuchstabe
- kommandospezifisch unzulässige Parameteranzahl (nicht immer Fehleranzeige)
- unzulässiges Zeichen in Parametern

Die Fehleranzeige erscheint auch bei:

- Auswahl der Kommandos "A" oder "L" nachdem der Programmbereich für deren Realisierung überschrieben wurde.
- fehlerhafte CRC-Prüfung beim Kommando "K".
- Eingabe eines unzulässigen Befehls beim Kommando "A".
  (Achtung! Die Eingaben werden hier nicht vollkommen kontrolliert. Es empfiehlt sich eine anschließende Kontrolle über das Kommando "L".)

Zeichenerklärung:

AD	Adresse
AAD	Anfangsadresse
EAD	Endadresse
B	Byte
<CR>	Kursor an Zeilenanfang
<space>	Leerzeichen
<SEP>	Trennzeichen
NMI	nichtmaskierter Interrupt
h	HEXA-Ziffer
d	Dezimalziffer
a	ASCII-Zeichen
x,y	Ziffern
ttt	File-Typ

EINGABE	BEDEUTUNG/WIRKUNG
	ASSEMBLER
a AD<CR>	- mnemonische Eingabe eines Maschinenbefehls ab AD (für Tabulator kann <space> eingegeben werden)
	- Abbruch mit <CR> statt Befehlseingabe
	- Befehl auf AD wird mnemonisch angezeigt
a<CR>	- wie oben, nur hier Eingabe ab aktiver Adresse (bei Neustart ab TPA-Beginn)
	- Zahleneingaben in den zu assemblierenden Befehlen sind wie bei Parametereingaben möglich
	BACKTRACE
b<CR>	- mnemonische Anzeige der bisher mit dem Kommando “T“ abgearbeiteten Befehle (rückwärts max. 128 Befehle)
-b<CR>	- Löschen der gespeicherten Befehlsfolge
bw<CR>	- wie b<CR>, hier nur Anzeige der Adressen der bisher abgearbeiteten Befehle
===>	laufende Anzeige möglich
	CALL
c AD<CR>	- Abarbeitung eines Unterprogramms ab AD in Echtzeit bis RET (BC und DE=0)
c AD AD1<CR>	- wie oben, nur hier BC=AD1 und DE=0
c AD AD1 AD2<CR>	- wie oben, nur hier BC=AD1 und DE=AD2
	HEXA-ARITHMETIK
h xxxx yyyy<CR>	- mit eingegebenen Zahlen werden folgende Operationen durchgeführt: 1. X+Y 2. Y-X 3. X-Y
	- Anzeige der Ergebnisse in o.a. Reihenfolge
h xxxx<CR>	- eingegebene Zahl wird in folgende Werte umgewandelt: 1. Hexawert 2. negativer Wert 3. Dezimalwert 4. ASCII-Zeichen und in gleicher Reihenfolge angezeigt
	INITIALISIERUNG
i N:a...a.ttt<CR>	- Vorbereitung einer Datei-Ein-/Ausgabe durch Eingabe von Laufwerk N, Dateinamen (max. 8 Zeichen) und Dateityp (max. 3 Zeichen), wenn kein Typ angegeben ist, dann “COM“)
	- Datei wird im file-control-block (FCB) eingetragen, Vorbereitung für Kommandos “R“ und “W“
	- in Abhängigkeit von der generierten Variante können folgende Geräte bedient werden: N = A...E Diskette o.ä. I Koppelinterface IFSS aktiv J Koppelinterface IFSS passiv K Kassette KC85/1 P Koppelinterface PIO aktiv Q Koppelinterface PIO passiv S Kassette ZX-Spectrum V Koppelinterface V24 aktiv W Koppelinterface V24 passiv

	(aktiv = Bestimmung des Transferbereiches und der Zieladresse) - Laufwerksangabe kann entfallen, wenn im akt. Laufwerk geblieben wird.
i N:<CR>	- Erzeugung einer COM-Datei mit 8 Leerzeichen im Dateinamen des FCB (Beim Lesen/Schreiben über Koppelinterface ist diese Eingabe sinnvoll, da dort der Name nicht übertragen wird.)
	INPUT
j AD<CR>	- Eingabe eines Bytes über Kanal AD und Anzeige
	- mit Taste <space> wird nächstes Byte über den selben Kanal eingegeben und angezeigt
	- Abbruch mit <CR>
	DISPLAY
d AAD EAD<CR>	- Anzeige des Speicherinhaltes zwischen AAD und EAD (HEXA- und ASCII-Information)
d AD<CR>	- wie oben, nur hier ab AD
d<CR>	- wie oben, nur hier ab der aktuellen Adresse
-d...	- wie bei d, nur hier ohne ASCII-Information
dw...	- wie bei d, nur hier (1. hochwertiges Byte, 2. niederwertiges Byte)
-dw...	- Anzeige in Doppelbytedarstellung, ohne ASCII-Zeichen
===>	laufende Anzeige möglich
	EQUAL
e AAD EAD AD<CR>	- Vergleich der Speicherinhalte im Sollbereich von AAD bis EAD mit dem Istbereich ab AD
	- Anzeige nichtübereinstimmender Positionen (Adresse am Sollbereich / Byte im Sollbereich / Byte im Istbereich)
===>	laufende Anzeige möglich
	FILL
f AAD EAD hh<CR>	- Füllen des Speicherbereiches von AAD bis EAD mit dem Wert hh
	GO
g AD<CR>	- Starten eines Programms ab AD
g AD AD1<CR>	- Abarbeiten eines Programms von AD bis AD1 (Haltepunkt durch 0FFH= RST38H)
	- wenn Haltepunkt erreicht ist werden Haltepunktadresse angezeigt, der Haltepunkt gelöscht und die Registerinhalte gerettet, Anzeige dieser über Kommando “X“ möglich
g AD AD1 AD2<CR>	- Abarbeiten von AD bis AD1 oder AD2
	- bei Erreichen eines Haltepunktes werden beide gelöscht
	- wird kein Haltepunkt erreicht, können sie über das Kommando “S“ gelöscht werden
g,AD<CR>	- Programm weiter abarbeiten bis AD
g,AD1 AD2<CR>	- weiter abarbeiten bis AD1 oder AD2
g,<CR>	- vom geretteten PC-Stand weiter abarbeiten
-g...	- wie oben, nur hier mit der Zeitmessung zwischen Start und Haltepunkt, Zeitangabe erfolgt beim Kommando “X“

KONTROLLSUMME

k AAD EAD<CR> - Berechnung eines CRC (16 Bit) nach MEOS-Routine vom angegebenen Bereich, Anzeige und Speicherung des CRC-Wortes

k<CR> - Kontrollrechnung vom selben Bereich wie im letzten “K“-Kommando angegeben und Vergleich, bei Nichtübereinstimmung Fehleranzeige “?“

LIST

l AAD EAD<CR> - Anzeige der mnemonisch dargestellten Maschinenbefehle im Bereich von AAD bis EAD

l AD<CR> - wie oben, nur hier ab der angegebenen Adresse AD

l<CR> - wie oben, nur hier ab der aktuellen Adresse

===> laufende Anzeige möglich

MOVE

m AAD EAD AD<CR> - Verschieben des Speicherbereiches von AAD bis EAD auf den Bereich ab AD

- Überlappungen sind nicht zu beachten

OUTPUT

o AD B1 B2...<CR> - Ausgabe einer Bytefolge über IO-Kanal AD (max. 16 Bytes)

PASSPOINT

p AD<CR> - Setzen eines Passpoints auf Adresse AD (Passpoint ist ein Haltepunkt, der nach dem Erreichen nicht gelöscht wird.), Durchlaufzähler = 1

- bei Erreichen des Passpoints werden die Registerinhalte gerettet und angezeigt, Rückgabe an ZSID erfolgt nur, wenn Durchlaufzähler = 1 ist

p AD hh<CR> - Setzen eines Passpoints auf AD mit Eingabe der Anzahl der Durchläufe (max. 255)

p<CR> - Anzeige der Adressen aller Passpoints mit Stand der Durchlaufzähler

-p<CR> - Löschen aller Passpoints

-p AD<CR> - Löschen aller Passpoints auf AD

- Es können max. 8 Passpoints eingegeben werden!

===> laufende Anzeige möglich

SUCHEN STRING - QUEST

q AAD EAD<CR> - Suchen einer Bytefolge (als ASCII- oder HEXA-Folge) im Bereich von AAD bis EAD

- nach Anfrage “search:“ ist der gesuchte String einzugeben:
  HEXA-Folge: hh1 ... hhn<CR> ($n \leq 16$)
  ASCII-Folge: a1 ... an<CR> ($n \leq 16$)

- bei Angabe von ‘?’ oder 3F in der HEXA-Folge bzw. ? in der ASCII-Folge (nicht an 1. Position!) wird diese Position beim Vergleich übergangen

- ASCII-Folgen mit Kleinbuchstaben im Speicher sind nur durch die Eingabe ihrer HEXA-Folge auffindbar

- die Anfangsadressen aller gefundenen Strings werden angezeigt

- die letzte gefundene Adresse wird als aktuelle Adresse an Kommando “L“ übergeben

q<CR> - suchen im zuletzt gewählten Adreßbereich

READ DATEI

r AD<CR> - Lesen einer Datei, deren Name vorher über das Kommando “I“ in den FCB eingegeben wurde, auf den Bereich ab AD

r<CR> - wie oben, nur hier auf die in der Datei festgelegten Adresse

- wird die Datei über das Koppelinterface gelesen, sind Anfangs- und Endadresse der Datei im Speicher des gekoppelten Rechners anzugeben:
  from/to: AAD EAD<CR>

SUBSTITUTE

s AD<CR> - Anzeige und Möglichkeit des Ersetzens des Speicherinhaltes von AD (HEXA oder ASCII)

- mit <CR> wird die nächste Speicherzelle angewählt

- mit ,<CR> Abbruch

- mit /<CR> wird der Inhalt der vorhergehenden Speicherzelle angezeigt

sw AD<CR> - wie oben, nur hier Anzeige und Ersatz eines Doppelbytes möglich

s<CR> - wie oben, nur ab aktueller Adresse

TRACE

t<CR> - Abarbeitung **e i n e s** Befehls im RAM ab aktueller Adresse

- nach Abarbeitung erfolgt Anzeige wie bei Kommando “X“

t hhhh<CR> - Abarbeitung von hhhh Befehlen (max. 65535)

- nach jedem abgearbeiteten Befehl wird wie bei Kommando “X“ angezeigt

===> laufende Anzeige möglich

UNTRACE

u hhhh<CR> - wie bei TRACE, aber hier werden nur die Registerinhalte vor dem ersten abzuarbeitenden Befehl und die Adresse des erreichten Befehls angezeigt

- Rückverfolgung des Pfades mit dem Kommando “B“ möglich

WRITE DATEI

w AAD EAD<CR> - Schreiben einer Datei, die vorher über das Kommando “I“ definiert wurde, aus dem angegebenen Speicherbereich von AAD bis EAD auf den durch Laufwerkscode festgelegten externen Speicher

- bei Ausgabe über das Koppelinterface ist die Zieladresse im Speicher des Koppelrechners anzugeben:
  to: AD<CR>

ZUSTANDSANZEIGE

x<CR> - Anzeige der Registerinhalte, Flags, Inhalte ausgewählter Speicherzellen und Mnemo-Code des aktuellen Befehls und eventuell Zeitanzeige

xReg<CR> - Anzeige des eingegebenen Registers bzw. Flags, Veränderung durch Neueingabe möglich

- Reg = a b d h p s x y
  Register: AF BC DE HL PC STACK IX IY

- Reihenfolge und Bedeutung der Flagsymbole in der Anzeige:

```
Reg = C     Z      M      E      I
      |     |      |      |      |
    carry  zero   sign  parity halfcarry
```

x AD<CR>	- Auswahl von Speicherzellen deren Inhalt bei x<CR> angezeigt wird (max. 3 möglich), Adreßeingabe mit führender Ziffer erforderlich)
	- Es wird der zum Zeitpunkt des Kommandos “X“ aktuelle Inhalt der Speicherzelle angezeigt
-x<CR>	- Löschender Speicherzellenanzeige

	ZUSATZFUNKTION
z<CR>	- Nach dieser Eingabe kann eine Routine aus einem Block von wahlweise generierbaren Zusatzfunktionen angesprungen werden (z.B. EPROM-Programmierung)

Ergänzende Bedienhinweise

- durch Taste <space> stoppen der Anzeige
  weiterlaufen durch:
  <space> --> weiterlaufen mit aktueller Geschwindigkeit
  1 ... 9 --> weiterlaufen mit veränderter Rollgeschwindigkeit
  <CR> --> Abbruch des Kommandos
- durch Taste <CR> Abbruch des Kommandos

Routinen, die als Zusatzfunktion generiert werden können

EINGABE	**BEDEUTUNG/WIRKUNG**
	ENDBLOCKERZEUGUNG LOCHBAND
eAD<CR>	- Ausgabe des Endblockes im INTEL-HEX-Format auf dem Stanzer mit Startadresse des Programms AD
	PUNCH
pAAD EAD<CR>	- Stanzen eines Bereiches von AAD bis EAD im INTEL-HEX-Format
	READ
rAD<CR>	- Lesen von INTEL-HEX-Code-Lochbändern auf der durch die auf dem Lochband angegebene Blockadresse und AD (Summe) bestimmten Speicherbereich

Bei den folgenden Routinen erscheinen die Bedienungshinweise nach Aufruf auf dem Display:

	LESEN
l	Einlesen von ASCII-Quellcodelochbändern
	STANZEN
s	Stanzen von ASCII-Zeichen
	WRITE
w	Programmieren und Lesen von EPROMs