Klaus Kohl

Heinz Schnitter

Handbuch zum

Zilog Super8-FORTH V1.0

Wichtige Hinweise:

Alle im Handbuch angegebenen Informationen wurden mit größter Sorgfalt zusammengestellt. Trotzdem können die Autoren keine Gewähr dafür übernehmen, daß die Angaben korrekt und die verwendeten Warenbezeichnungen, Warenzeichen und Programmlistings frei von Schutzrechten Dritter ist. Da sich Fehler nie vollständig vermeiden lassen, ist die Autoren für Hinweise jederzeit dankbar.

Ing. Büro
Klaus Kohl
Pestalozzistr. 69
8905 Mering
Tel. 08233/30524

Inhaltsverzeichnis

1. Einleitung

1.1 Vorwort

Das Super8-FORTH entstand durch Zusammenarbeit zweier Mitglieder der FORTH-Gesellschaft e.V. . Das vom Dezember 1989 bis Juni 1990 entwickelte FORTH wurde im Dezember 1990 in das ROM einer Super8-Maske übernommen.

Ausgangsbasis war das Entwicklungskit mit Monitorprogramm für den Super8. Nach der Einarbeitung in die Mnemonics des Prozessors erstellte Heinz Schnitter einen sehr kompakten, vollständig in FORTH geschriebenen Assembler. Durch seine Realisierung in der Hochsprache FORTH kann er auch auf anderen Prozessoren laufen.

Auf Grund seiner Erfahrung mit Implementierungen von FORTH-Systemen auf anderen Prozessoren (RTX-2000, Z80, 8086) übernahm Klaus Kohl die Erstellung des Super8-FORTH. Mittels des oben erwähnten Assemblers und einem ebenfalls in FORTH geschriebenen Metacompiler konnte dann mit Hilfe des Monitorprogramms interaktiv eine RAM-Version des Super8-FORTH realisiert werden. Nach Überarbeitung des gesamten Speicherkonzepts und der Einbindung der von Heinz Schnitter erstellten Initialisierungssequenz konnte dann eine unabhängige ROM-Version erzeugt werden. Leider mußten wegen der Begrenzung des ROM's auf 8KByte viele Befehle der RAM-Version gestrichen werden. Trotzdem konnte durch trickreiche Programmierung bis auf den Befehl FORGET der vollständige Befehlsumfang des FORTH83-Standards beibehalten werden. Der noch fehlende Befehl und einige zusätzliche Definitionen sind im mitgelieferten File ZUSATZ.SCR enthalten und können nachgeladen werden.

Diese Beschreibung behandelt nur die Eigenschaften dieses Super8-FORTH. Es muß davon ausgegangen werden, daß der Anwender zumindest über Grundkenntnisse zum FORTH verfügt. Trotzdem kann dieses FORTH auch für Anfänger empfohlen werden, da es sich sehr streng an den FORTH83-Standard hält. Zusätzlich enthält das Kapitel 1.8 ausführliche Beispiele für das Arbeiten mit dem Super8-FORTH und zur Erstellung einer automatisch nach dem Einschalten startenden Applikation im EPROM.

Als Ergänzung zum Handbuch ist folgende Literatur zu empfehlen:

* Einführung in die Programmiersprache FORTH

  Leo Brodie
  Programmieren in FORTH
  Carl Hanser Verlag, München/Wien
  ISBN 3-446-14070-0

  Ronald Zech
  FORTH83
  Franzis-Verlag GmbH, München
  ISBN 3-7723-8621-0

* Beschreibung des FORTH83-Standard

  FORTH-83 Standard
  A Publication of the FORTH Standards Team
  FORTH Standards Team;P.O. Box 4545;Mountain View,CA 94040

* Weitere Informationen zum Prozessor Super8

  Super8 MCU - Product Specification
  Super8 MCU - Technical Manual
  Zilog, Inc.; 210 Hacienda Ave., Campbell, California 95008-6609

Kontakte zu FORTH-Programmierer

FORTH-Gesellschaft e.V.
Postfach 1110
W-8044 Unterschleißheim
Tel. 089/3173784

FORTH Interest Group
P.O.Box 8231
San Jose, CA 95155
USA

1.2 Eigenschaften des S8-FORTH

Bei der Realisierung des Super8-FORTH wurden sehr viele, über die üblichen Fähigkeiten eines FORTH-Systems hinausgehenden Eigenschaften realisiert:

* Umfang von 8KByte (damit für EPROM-Version und ROM-Maske geeignet)
* ROM-Fähig durch Auslagerung von Programmteile und Daten ins RAM
* Bis auf das fehlende FORGET vollen FORTH83-Befehlsumfang
* Fileinterface über PC-Terminalprogramm implementiert
* Enthält Befehl SAVESYSTEM zur Speicherung von (Autostart-)Applikationen
* Unterstützt einen Heap für temporäre Programme und Befehlsheader
* Einbindung eigener Routinen in den FORTH-Kern
* Umleitung von Ein-/Ausgabe und Fileinterface vorgesehen
* Fehlerbehandlung auf eigene Routinen umleitbar (Aufruf mit Fehlernummer)
* Vorbereitet für Multitasking (bis zu 12 Tasks im Registersatz realisierbar)
* Einbindung eigener Interruptbefehle (Vektoren im RAM)

Viele dieser zusätzlichen Eigenschaften sind an das volksFORTH, einer FORTH-Implementierung für C64, Atari ST, CP/M und IBM-PC angelehnt. Sie werden beim normalen Arbeiten nicht unbedingt benötigt, erleichtern aber die Realisierung eigener Anwendungen. Vor allem der Heap ist besonders dann interessant, wenn Programmteile (wie der Assembler) nur für kurze Zeit benötigt werden und danach wieder gelöscht werden sollen.

1.3 Hardware-Voraussetzungen

Die wichtigste Vorgabe war, das die Maskenversion des FORTH sofort nach dem Einstecken in das Entwicklungsboard und dem Anschließen eines Terminals zur Kommunikation bereit ist.

Folgende Hardware ist notwendig für das Arbeiten mit dem Super8-FORTH:

* Super8-Microprozessor mit FORTH im Masken-ROM (Nummer 0887520)
* Externes RAM mit mindestens 2KByte am Speicherende des Codebereiches
* Taktgenerator (Quarz) mit 19.661MHz (für 9600 Baud)
* MAX232-Treiber für die RS232-Leitungen
* Terminalprogramm mit 9600 Baud

Sehr gut geeignet für das Arbeiten mit dem Super8-FORTH ist das "Zilog Super8 Design Module", dem vom Zilog ausgelieferten Entwicklungsboard. Mit 32KByte stehen genügend Speicher für eigene Befehlsdefinitionen und Variablen zur Verfügung. Darüber hinaus ist ein freier Sockel für ein EPROM vorhanden. Im EPROM können dann weitere Befehle oder sogar vollständige Programme abgelegt werden,

die beim Start des Super8-FORTH automatisch wieder in das RAM übernommen werden.

Das folgende Schaltbild enthält die notwendige Hardware für das Arbeiten mit Super8-FORTH. Dabei wird Port 0 und 1 für die Ansteuerung des externen Speichers oder zusätzlicher Hardware herangezogen. Vom Port 3 werden die Bits 0 und 1 für die serielle Schnittstelle (UART) verwendet. Ports 2, Bit 2-7 von Port 3 und Port 4 werden vom Super8-FORTH nicht verwendet und können zusammen mit den internen Zähler oder der DMA für eigene Zwecke verwendet werden.

((Bild 1))
((Schaltbild Super8))

1.4 Software-Voraussetzungen

Da keine weitere Hardware auf dem Super8-Board erwartet werden kann, müssen alle für die Programmentwicklung notwendigen Tätigkeiten über die serielle Schnittstelle abgewickelt werden.

Solange nur interaktiv die Befehle des S8-FORTH oder der eigenen Applikationen aufgerufen werden oder Daten ausgegeben werden, kann jedes beliebige 9600Baud-Terminal verwendet werden. Falls mit diesem Terminal auch noch Programme übertragen werden sollen, so müssen die sequenziellen Files mit ausreichender Wartezeit zur Interpretation einer Zeile nach dem RETURN geschickt werden.

Wenn aber das Super8-FORTH selbst einzelne Blöcke eines Screenfiles anfordern oder die geänderten Blöcke wieder zum Terminal zurückschicken soll, so muß das Terminal die Steuercodes interpretieren. Ein zusätzliche Steuersequenz dient der Übertragung eines Bitimage vom Super8 zur Diskette oder Harddisk (Befehl SAVESYSTEM). Jede Steuersequenz beginnt mit dem ASCII-Wert $1B (ESC) mit nachfolgender Befehlsnummer und einem 16Bit-Wert mit befehlsabhängiger Bedeutung. Die genaue Beschreibung ist im Kapitel über das Fileinterface zu finden.

1.5 Beschreibung der Files

In diesem Kapitel werden die auf den beiden Disketten ausgelieferten Programme beschrieben. Normalerweise wird für die Programmentwicklung nur das richtige Terminalprogramm (T1.COM bis T4.COM) benötigt. File:S8.ASM und File:ZUSATZ.SCR dienen zum Nachladen des Assemblers und der zusätzlichen Befehle des vollständigen FORTH83-Standards. Das File File:TEST.SCR wird beim Start des Terminalprogrammes als aktuelles Arbeitsfile behandelt, muß aber nicht unbedingt vorhanden sein.

1.5.1 Arbeitsdiskette

```
README   DOC     3385  23.03.91  21.48
T_EDIT   DOC     3209   7.03.91  18.22
---VF--- ---        0   3.09.89  20.17
T1       COM    28404  23.03.91  20.10
T2       COM    28404  23.03.91  20.12
T3       COM    28404  23.03.91  20.11
T4       COM    28404  23.03.91  20.13
EDITOR   COM    21860  13.01.91  22.59
--S8_F10 ---        0   3.09.89  20.17
S8V10    SCR   114688  23.03.91  20.47
S8       ASM    33792  23.03.91  20.55
ZUSATZ   SCR    20480  23.03.91  21.17
TEST     SCR     9216  23.03.91  21.13
S8       LOG      251  23.03.91  21.27
S8       IMG     7890  23.03.91  21.25
ISR      SCR     9216  24.03.91  16.00
```

File:README.DOC enthält in Stichpunkten die wichtigsten Hinweise zum Arbeiten mit der Entwicklungsdiskette. In File:T_EDIT.DOC steht die nicht als Hilfsinformation verfügbare Tastenbelegung des integrierten Editors. Man sollte zumindest während des Einarbeitens in den Editor entweder die Liste aus dem Anhang oder einen Ausdruck dieses Files bereithalten.

Die Terminalprogramme T1.COM bis T4.COM sind bis auf die Ansteuerung der unterschiedlichen Schnittstellen identisch. Ausführliche Beschreibungen der Fähigkeiten folgen im Kapitel 1.6 und bei den einzelnen Beispiele.

File:EDITOR.COM ist ein Editor für Screenfiles, der das gesamte File in den Speicher des PC ladet und deshalb sehr schnell ist. Zusätzlich ist mit diesem Editor ein Verkürzen der Files möglich, da beim Löschen einzelner Screens der letzte Screen entfernt wird. Die Tastaturbelegung ist dabei mit dem integrierten Editor des Terminalprogramms identisch. Falls beim Start von EDITOR.COM kein Filename in der gleichen Zeile angegeben ist, wird er gesondert abgefragt. Die ID (siehe Beschreibung des Editors) kann nicht mit F9 verändert, sondern muß beim Start eingegeben werden.

Für Interessierte oder als Beispiel für ein FORTH-Programm wird das komplette Source-Listing des Super8-FORTH mitgeliefert. Neben den Informationen zur Arbeitsweise der einzelnen Befehle können auch die Anfangsadressen versteckter Befehle ermittelt und der Aufbau der vom FORTH verwendeten Speicherbereiche eingesehen werden.

Für eigene Assemblerprogramme und für das Laden der zusätzlichen Befehle wird File:S8.ASM benötigt. Dieser in FORTH geschriebene Assembler von Heinz Schnitter wird sehr ausführlich im Kapitel 3 beschrieben.

In File:ZUSATZ.SCR sind zusätzliche Befehle und die Fehlermeldungen als Text enthalten. Sie konnten aus Platzgründen nicht mehr in das Masken-ROM des Super8 übernommen werden. Wie dieses File zu laden ist, wird im Kapitel 1.8.4 erklärt. Da dazu auch der Assembler benötigt wird und die zusätzlichen Befehle eigentlich immer sinnvoll sind, dient diese Beschreibung des Ladevorgangs als Beispiel für die Erstellung eines EPROM's.

Das File File:TEST.SCR wird, falls es im gleichen Verzeichnis vorhanden ist, von den Terminalprogrammen automatisch geöffnet. Es kann also sofort mit dem Editor verändert oder einzelne Screens geladen werden. Folgende Befehle sind nach dem Laden des Befehls verfügbar:

```
Screen  1 : Ladescreen für alle Befehle
Screen  2 : COMCOUNT          - Zählen der Befehlsanzahl
Screen  3 : DUMP              - DUMP mit ASCII-Ausgabe
Screen  4 : WORDS             - Wortliste mit zusätzlichen Angaben
Screen  5 : EI                - Test der Interrupt-Fehlermeldung
Screen  6 : INIT_CO0  CO0@    - Zähler 0 starten und auslesen
Screen  7 : TABELLE           - Multiplikationstabelle ausgeben
```

Falls man sich die Arbeit zur Erstellung eines neuen Bitimage sparen, aber trotzdem über Assembler und dem vollen FORTH83-Vokabular verfügen will, so muß man nur das File S8.IMG ab der Adresse $2000 in ein EPROM brennen und z.B. auf dem Entwicklungsboard in den freien Sockel stecken. Stehen dann die Jumper richtig, so erkennt das Super8-FORTH dieses Bitimage und kopiert es an die Speicheradresse $8000. Dort beginnt normalerweise das RAM des Entwicklungssystems.

Wie aus dem Datum ermittelt werden kann, wurde gleichzeitig während der Erstellung des Bitimage auch das Logfile File:S8.LOG erstellt. Es wird aber für die Arbeitsdiskette nicht benötigt, da beim Aktivieren der Log-Funktion automatisch ein neues File erzeugt wird.

1.5.2 volksFORTH-Diskette

```
README   DOC     2979   8.03.91  11.33
---VF--- ---        0   8.03.91  10.52
READ     ME     15154   7.05.89  14.29
INSTALL  BAT      607  12.09.88  11.08
PKXARC   COM    12242  27.04.87
KERNEL   COM    15874  11.03.89  13.45
FORTH1   ARC    65995  11.03.89  13.47
FORTH2   ARC    39663  11.03.89  13.47
VOLKS4TH COM    31724  11.03.89  13.46
VOLKS4TH DOC    34494   7.05.89  14.34
--S8-T-- --         0   8.03.91  10.52
EDITOR   SCR    40960   3.09.89   9.32
INSTALL  SCR     3072   2.09.89  12.20
NECP6    SCR    13312  24.02.89  16.09
MINITERM SCR    19456   8.03.91   9.27
T        COM    38578   8.03.91  11.04
S8       IMG     8192  22.11.90  19.48
TEST     SCR     8192   8.03.91   9.28
```

Wenn für eigene Applikationen einmal mehr als nur ein Terminalprogramm benötigt wird, so muß man auch ein eigenes Programm dafür entwickeln. Deshalb wird auf dieser Diskette sowohl das PC-volksFORTH als auch Sourcen eines Terminalprogramms mitgegeben. Allerdings sind in dieser Version weder Druckerausgabe noch Logfile implementiert.

Bei dem volksFORTH handelt es sich um eine von Mitgliedern der FORTH-Gesellschaft e.V. entwickelten FORTH-Version für IBM-PC und Kompatible. Es wird mit allen Sourcen und einigen Beschreibungen geliefert. Das Handbuch zu diesem FORTH und die anderen volksFORTH-Version für C64, Atari ST und CP/M kann

beim Vertrieb der FORTH-Gesellschaft e.V. bestellt werden. Die Adresse ist im File README.DOC enthalten.

Beim Start des Terminalprogramms T.COM (für COM1 vorbereitet) wird das File TEST.SCR für die Bearbeitung geöffnet und der Interpreter des volksFORTH aktiviert. Mit dem Befehl T kann man dann in den Terminalmodus wechseln. Dieser Modus wird mit der ESC-Taste wieder beendet. Ansonsten stehen alle Befehle des volksFORTH zur Verfügung. Wie mit den Programmen auf dieser Diskette gearbeitet wird, ist im Kapitel 1.7 beschrieben.

1.6 Terminalprogramme Tx.COM

Mit File:T1.COM bis File:T4.COM erhalten Sie nicht nur ein einfaches Terminalprogramm für 9600 Baud, sondern eine komplette Arbeitsumgebung mit folgenden Fähigkeiten:

* Interruptgesteuerter Zeichenempfang (4KByte-Puffer)
* Interpreter für Steuerbefehle des Super8-FORTH
  - Übertragen von Screens in beide Richtungen
  - Speichern des Bitimage auf Diskette (Befehl SAVESYSTEM)
* Integrierter Screeneditor
* Paralleles Speichern der Super8-Ausgaben in ein Logfile
* Parallele Ausgabe auf einen Drucker
* Anzeige des Directory vom aktuellen Verzeichnis
* Aufruf des DOS-Shell aus dem Terminalmodus

Da dieses Programm ebenfalls in FORTH geschrieben wurde, hat es nur einen Umfang von 28KByte und benötigt zum Arbeiten 130KByte des PC-Speichers.

1.6.1 Terminalmodus

Beim Start des Programmes wird die entsprechende Schnittstelle (COM1 bei T1.COM) auf 9600 Baud, 8 Datenbits und 1 Stopbit ohne Handshake initialisiert und der Empfangsinterrupt auf eine eigene Routine umgeleitet. Da die empfangenen Zeichen in einen Ringpuffer für 4096 Zeichen gehen, kann parallel dazu die Ausgabe der Zeichen auf Bildschirm, Drucker und Logfile erfolgen. Eingaben von Tastatur werden bis auf die Steuerzeichen (ALT+Taste und alle Funktionstasten) direkt an das Super8-FORTH weitergereicht. Erst die vom FORTH zurückgeschichten Zeichen werden dann auf dem Bildschirm (und Drucker/Logfile) ausgegeben.

1.6.2 Befehlsinterpreter für Super8-FORTH

Empfängt das Terminalprogramm den ASCII-Wert $1B (ESC-Zeichen), so wird eine Kommando vom Super8-FORTH erwartet und interpretiert. Dazu Interpretiert das Programm das nächste empfangene Byte als Befehlsnummer und die nachfolgenden zwei Bytes als 16Bit-Parameter (zuerst Highbyte). Folgende Befehlsnummern werden behandelt:

$00 Es wird das Escape-Zeichen ($1B) ausgegeben
(16Bit-Parameter werden ignoriert)

$01 Der mit dem 16Bit-Parameter angegebene Screen wird von Diskette geholt und zum Super8 übertragen. Falls die Screennummer nicht verfügbar ist, erfolgt eine Fehlermeldung. Damit man auch auf dem Bildschirm sieht, daß etwas übertragen wird, gibt das Terminal noch ">" aus. Dieses Zeichen wird aber weder auf dem Drucker ausgegeben noch im Logfile gespeichert.

$02 Der mit dem 16Bit-Parameter angegebene Screen wird vom Super8 zur Diskette zurückgeschrieben. Auch hier wird ein Fehler ausgegeben, falls die Screennummer nicht verfügbar ist. Nur auf dem Display wird zur Information das Zeichen "<" ausgegeben.

$03 Nach Aufruf des Befehls SAVESYSTEM schickt das Super8-FORTH ein Bitimage des aktuellen Systemzustandes (Dictionary- und Task-Bereiche) zum Terminal. Die Länge wird mit dem 16Bit-Parameter übergeben. Der Name des Imagefiles kann mit der Tastenkombination ALT+I geändert werden.

Wenn das Terminal den Befehl $01 bis $03 erkannt hat, wird zuerst die Fehlernummer 0 als Bestätigung geschickt. Danach folgt dann die Datenübertragung vom Super8 zum Terminal oder umgekehrt. Sind unerlaubte Steuercodes übertragen worden, so wird die Fehlernummer 7 zurückgeschickt.

ACHTUNG: Da das erste Zeichen nach einer Befehlsübertragung als Fehlerflag behandelt wird, stören Tastatureingaben die Datenübertragung. Das Laden eines Files kann deshalb durch das Drücken einer beliebigen Taste abgebrochen werden. Der ASCII-Wert der entsprechenden Taste wird als Fehlernummer ausgegeben.

1.6.3 Befehlsinterpreter für das Terminal

Bis auf die mit ASCII-Wert 0 beginnenden Funktionstasten und der Kombination ALT+Taste werden alle Zeichen unverändert an das Super8-FORTH weitergegeben und erst dort interpretiert oder ignoriert. Mit den Tastenkombinationen ALT+Taste oder der Funktionstaste F1 können alle Funktionen des Terminals aufgerufen werden:

F1
ALT+H Anzeige der Hilfsinformation für die Tastenbelegung

ALT+F Ein neuer Filename für das aktuelle Arbeitsfile kann eingegeben werden. Als Vorgabe erscheint der alte Filename. Er wird dann übernommen, wenn sich der Cursor beim Drücken der RETURN-Taste am Anfang des Eingabefeldes befindet. Da das Super8-FORTH diese Änderung nicht erkennt, sollte danach mit EMPTY-BUFFERS der Diskpuffer des Zielsystems gelöscht werden. Im Logfile und auf dem Drucker ist die Änderung nicht sichtbar.

ALT+E Aufruf des integrierten Editors. Dies ist nur im Textmodus des PC (Modus 2, 3 und 7) und bei einem geöffneten File möglich. Die Bedienung des Editors wird im nachfolgenden Kapitel noch ausführlicher beschrieben. Auch hier muß mit EMPTY-BUFFERS der alte Inhalt des Diskpuffers gelöscht werden.

ALT+P Ein-/Ausschalten des Druckers. Nur im Display wird die Information "((Printer on))" oder "((Printer off))" ausgegeben. Da die Ausgabe parallel zum Bildschirm erfolgt, bleibt das Terminalprogramm stehen, bis der Drucker bereit ist. Es können aber trotzdem noch Zeichen vom Super8 empfangen werden.

ALT+L Beim Arbeiten mit dem Super8 können alle empfangenen Zeichen auch in einem Logfile geschrieben werden. Dadurch kann, wie beim Erstellen dieser Beschreibungen, das Arbeiten mit dem Super8 protokolliert werden. Nach Drücken von ALT+L muß der Name des Files (Vorgabe: S8.LOG) eingegeben werden. Dieses File wird dann mit Länge 0 geöffnet und alle danach vom Super8 emfangenen Zeichen darin gespeichert. Falls beim Speichern ein Fehler auftritt (Diskette voll oder nicht beschreibbar), so wird das Logfile geschlossen. Es kann aber auch durch erneute Betätigung von ALT+L geschlossen werden. In der Anzeige wird dann "((Logfile closed))" ausgegeben.

ALT+I Mit SAVESYSTEM wird ein Imagefile erzeugt. Der Name des Files ist mit S8.IMG vorgegeben, kann aber mit ALT+I geändert werden. Diese Änderung ist nur in der Anzeige sichtbar.

ALT+D Das Directory des aktuellen Verzeichnis wird bei ALT+D angezeigt. Weder auf dem Drucker noch im Logfile sind diese Ausgaben sichtbar.

ALT+C Aufruf der COMMAND-Oberfläche des Betriebssystems. Diese Funktion ist erst ab DOS 3.1 möglich und erlaubt danach die Eingabe von DOS-Befehlen. Da aber das Terminalprogramm weiterhin im Speicher steht, dürfen weder die verwendeten Files noch die aktive

Schnittstelle durch diese Befehle verändert werden. Nach der Eingabe von EXIT kehrt man wieder in das Terminalprogramm zurück.

ALT+X Terminalprogramm nach dem Schließen des aktuellen Arbeitsfile und des Logfile beenden. Zur Sicherheit wird vorher abgefragt, ob das Programm wirklich verlassen werden soll.

1.6.4 Integrierter Editor des Terminalprogrammes

Bis auf die Art des Aufrufes und die Verwendung der Funktionstaste F9 ist die Bedienung des integrierten Editor und des selbstständigen Editor EDITOR.COM identisch.

Cursorsteuerung:

^E oder Cursor_hoch	Eine Zeile höher
^X oder Cursor_tief	Eine Zeile tiefer
^S oder Cursor_links	Ein Zeichen links
^D oder Cursor_rechts	Ein Zeichen rechts
TAB	Zur nächsten 4er-Teilung
Shift+TAB	Zur vorherigen 4er-Teilung
^Q B	Zum Zeilenanfang
^Q K	Zum Zeilenende-1
^Q E	Zum Screenanfang
^Q X	Zum Screenende-1
POS1	Zum ersten Zeichen der Zeile
ENDE	Hinter das letzte Zeichen der Zeile
^POS1	Zum ersten Zeichen des Screens
^ENDE	Hinter das letzte Zeichen des Screens
Return	Zum nächsten Zeilenanfang
^R oder Bild_hoch	Zum vorhergehenden Screen
^C oder Bild_tief	Zum nächsten Screen
^K R oder ^Bild_hoch	Zum ersten Screen
^K C oder ^Bild_tief	Zum letzten Screen
^Q G	Eingabe der gewünschten Screennummer
^A	Zum zweiten File / Cursorposition umschalten

Zwischenspeicherung von Zeichen und Zeilen:

F1	Zeichen speichern und löschen
F2	Zeichen speichern, Cursor rechts
F3	Zeichen einfügen
F5	Zeile speichern und löschen
F6	Zeile speichern, Cursor in die nächste Zeile
F7	Zeile einfügen

Steuerung der Zeicheneingabe und des Löschens:

^V	Insert-Modus umschalten
Einfg	Ein Leerzeichen einfügen
^G oder Entf	Zeichen unter dem Cursor löschen
^H oder Backspace	Zeichen vor dem Cursor löschen
F8	Rest der Zeile löschen
^Y	Aktuelle Zeile entfernen
^N	Leerzeile einfügen
^K N	Leerscreen einfügen
^K Y	Aktuellen Screen entfernen
^Backspace	Nächste Zeile in den Rest dieser Zeile übernehmen
^Return	Rest dieser Zeile in die nächste Zeile

Suchen und Ersetzen:

^Q F	String suchen (u=Option für Rückwärts-Suche)
^Q A	String suchen und ersetzen (mehrere Optionen)

Speicherung:

F10	Änderungen in diesem Screen rückgängig machen
F9	Eingabe der ID-Kennung (nicht bei EDITOR.COM)
ESC	File speichern, Editor verlassen

Im Rahmen wird ein Screen mit 16 Zeilen zu 64 Zeichen dargestellt. Der Filename, die aktuelle Position (Screennummer, X- und Y-Position), die Anzahl der im Zeilen- (L:000) und Zeichenpuffer (C:000) gespeicherten Zeilen/Zeichen, ein Flag für Einfügen (I) oder Überschreiben (O) und die aktuelle Uhrzeit wird angezeigt. Mit den Cursortasten und einige, an WORDSTAR angelehnten Tastenkombinationen kann dann das File editiert werden. Statt der Blockbefehle können Zeilen oder einzelne Zeichen in einen Zwischenpuffer gebracht und von dort aus an einer beliebigen Stelle wieder eingefügt werden.

Falls das File zu kurz ist, so kann ein neuer Screen mit CTRL+K N eingefügt werden. Löschen einzelner Screens ist mit CTRL+K Y möglich. Da das File nicht verkürzt werden kann, bleibt danach der letzte Screen leer. Bei Fehler während des Editierens kehrt das Programm wieder in den Terminalmodus zurück. Es sind aber die letzten Änderungen in dem aktuellen Screen verloren. Alle anderen Daten befinden sich aber schon auf Diskette.

Beim unabhängigen Editor wird das gesamte File vollständig in den Speicher des PC's geladen und dort editiert. Deshalb können mit CTRL+K Y die Files auch verkürzt werden. Da aber vor dem Zurückschreiben des Files das alte File gelöscht wird, ist bei einem Fehler während des Speichervorganges das File verloren. Es sollte daher immer eine Kopie dieses Files vorhanden sein.

1.7 Das volksFORTH

Für Demonstationszwecke befindet sich auf der zweiten Diskette neben dem gesamten PC-volksFORTH ein für das Super8-FORTH verwendbares Terminalprogramm. Es sollte nur dann verwendet werden, wenn die auf der anderen Diskette befindlichen Terminalprogramme nicht funktionieren oder eigene Programme für den Datenaustausch entwickelt werden sollten. Da in dem im volksFORTH mitgelieferten Editor einige Eigenschaften sehr unschön sind (File wird automatisch verlängert) wurde eine geänderte Version EDITOR.SCR mit INSTALL.SCR mitgeliefert.

1.7.1 Erzeugung eines Terminalprogrammes

Sobald man das Terminalprogramm T.COM startet, stehen alle Befehle des volksFORTH und zusätzliche Terminalbefehle zur Verfügung. Davon sind aber für das normale Arbeiten nur folgende Befehle wichtig:

`MAKEFILE name`	Anlegen eines neuen Files
`n MORE`	File um n Screens (mit je 1024 Zeichen) vergrößern
`USE name`	Vorhandenes File zur Bearbeitung öffnen
`SAVESYSTEM name.com`	Neues (Terminal-)Programm abspeichern
`n L`	Screen n editieren (Ende mit ESC)

T	Start des Terminalbetriebs (Ende mit ESC)
BYE	Verlassen des FORTH-Systems

T.COM ist auf dieser Diskette auf COM1 eingestellt. Jedoch ist auch das Umstellen auf andere Schnittstellen oder Baudraten keine Affäre:

```
T                                   Alte Version starten
USE MINITERM.SCR                    Aktuelles File: Terminalprogramm
  ( File nach 2 L entsprechen verändern, ESC beendet Editor )
' MINITERM.SCR >NAME 4 - (FORGET    Terminalteil löschen
INCLUDE MINITERM.SCR                und erneut laden
USE TEST.SCR                        Aktuelles File angeben
SAVESYSTEM MEINT.COM                Programm abspeichern
BYE                                 das war's
```

In Screen 2 werden nur die Zeilen mit | am Zeilenanfang übernommen. Die mit \ markierten Zeilen werden ignoriert. Es ist auf die richtige Zuordnung der Portadresse, Interruptnummer und -ebene zu achten. Die Baudrate wird durch den Teiler 115200/n ermittelt.

1.7.2 Arbeiten mit dem Terminalprogramm

Nachdem man eine geeignete Arbeitsversion für die richtige Schnittstelle erstellt hat, kann damit fast wie mit der integrierten Arbeitsumgebung gearbeitet werden. Da bei Start zuerst der FORTH-Interpreter aktiviert wird, muß nach der Vorgabe des richtigen Files durch Eingabe des Befehls T in den Terminalmodus gewechselt werden. Mit der ESC-Taste kann man diesen Modus wieder verlassen.

1.7.2.1 Eingabe und Test eines Befehls

Mit folgender Befehlsequenz wird ein neuer Befehl mit Namen MULTI eingegeben und im Interpretermodus getestet. In diesem Beispiel wird der Befehl auf zwei Zeilen verteilt. Damit man bei der Eingabe erkennt, daß man sich noch im Kompilermodus befindet, wird kein "OK", sondern] am Anfang der nächsten Zeile ausgegeben.

```
T                       (( Terminalprogramm laden und starten ))
USE TEST.SCR            (( Testfile als aktuelles File anmelden ))
T                       (( In den Terminal-Modus gehen ))
  (( Super8-Karte anschießen und einschalten, Meldung abwarten ))
 Zilog Super8-FORTH V1.0
words                   (( Wortliste anzeigen ))
COLD  'COLD  ABORT  'ABORT  IDENT  QUIT  INTERPRET  SAVESYSTEM
... (( weitere Befehle )) ...
LEAVE  ?BRANCH  BRANCH  EXECUTE  EXIT  ok
: multi                 (( Eingabe des neuen Befehls ))
```

```
] 10 0 do cr 10 0 do j 10 * i + 3 u.r loop loop cr ; ok
```
((Der Befehl kann natürlich auch in einer Zeile eingegeben werden))
```
.s <empty> ok          (( Stack testen ))
multi                  (( Befehl aufrufen ))
  0  1  2  3  4  5  6  7  8  9
 10 11 12 13 14 15 16 17 18 19
 20 21 22 23 24 25 26 27 28 29
 30 31 32 33 34 35 36 37 38 39
 40 41 42 43 44 45 46 47 48 49
 50 51 52 53 54 55 56 57 58 59
 60 61 62 63 64 65 66 67 68 69
 70 71 72 73 74 75 76 77 78 79
 80 81 82 83 84 85 86 87 88 89
 90 91 92 93 94 95 96 97 98 99
ok
```
((ESC drücken, um den Terminalmoudus zu verlassen))
```
BYE                    (( Terminalprogramm verlassen ))
```

1.7.2.2 EPROM mit Assembler und Zusätze erstellen

Auch die im nächsten Kapitel als Beispiel angegebene Erstellung eines Bitimage ist mit diesem Terminalprogramm möglich. Es muß nur darauf geachtet werden, daß das immer verwendete File S8.IMG ausreichend groß für die Speicherung des gesamten Image ist. Ansonsten muß es mit MAKEFILE S8.IMG 16 MORE neu angelegt und auf eine meistens ausreichende Größe von 16KByte erweitert werden.

```
T                      (( Terminalprogramm starten ))
USE S8.ASM             (( Assembler-File als aktuelles File anmelden ))
T                      (( In den Terminal-Modus gehen ))
```
((Super8-Karte anschießen und einschalten, Meldung abwarten))
```
1 LOAD                 (( Assembler laden ))
```
((Die Nummer der gerade übertragenen Screens wird ausgegeben))
((Nach "ok" ESC drücken, um Terminalmodus zu verlassen))
```
USE ZUSATZ.SCR         (( Anmeldung des Zusatz-Files ))
T                      (( Wieder in den Terminalmodus gehen ))
EMPTY-BUFFERS          (( Diskpuffer auf der Karte löschen ))
1 LOAD                 (( Zusätze laden ))
```
((Die Nummer der übertragenen Screens wird wieder ausgegeben))
```
HCLEAR                 (( Entfernen überflüssiger Header vom HEAP ))
SAVESYSTEM             (( EPROM-Image erstellen ))
```
((ESC drücken, um den Terminalmoudus zu verlassen))
```
BYE                    (( Terminalprogramm verlassen ))
```

1.7.3 Tastenbelegung des Editors

Die nachfolgende Tastenbelegung des volksFORTH-Editor ist dem entsprechenden Handbuch entnommen. Sie kann jederzeit durch das Laden des Files INSTALL.SCR interaktiv geändert werden.

F1	Gibt Hilfestellung für den Editor
ESC	Verläßt Editor mit sofortiger Speicherung der Änderungen
CTRL-U	(undo) Macht die noch nicht auf Diskette zurückgeschiebenen Änderungen rückgängig
CTRL-E	Verläßt den Editor ohne sofortige Speichung
CTRL-F	(fix) Sucht das Wort rechts vom Cursor, ohne den Editor zu verlassen
CTRL-L	(showload) Lädt den Screen ab Cursorposition
CTRL-N	Fügt an Cursorposition eine neue Zeile ein
CTRL-PgDn	Splittet diese Zeile an der Cursorposition
CTRL-S	(Scr#) Legt die Nummer aktuellen Screens auf dem Stack ab
CTRL-Y	Löscht aktuelle Zeile
CTRL-PgUp	(join) Fügt nächste Zeile ab Cursorposition ein
TAB	Bewegt den Cursor einen großen TABulator vor
SHIFT TAB	Bewegt den Cursor einen kleinen TABulator zurück
F2	(suchen/ersetzen) Erwartet eine Zeichenkette, die gesucht und eine Zeichenkette, die statt dessen eingefügt werden soll Wird eine Übereinstimmung gefunden, kann man mit R (replace) die gefundene Zeichenkette ersetzen, mit <cr> den Suchvorgang abbrechen oder mit jeder anderen Taste weitersuchen. Auf diese Weise kann man die Quelltexte auch nach einer Zeichenkette durchsuchen lassen. Als Ersatz-String wird dann <CR> eingegeben.
F3	Bringt eine Zeile in den Zeilenpuffer und Löscht sie vom Screen
F5	Bringt die Kopie einer Zeile in den Zeilenpuffer
F7	Fügt die Zeile aus dem Zeilenpuffer in den Screen ein
F4	wie F3, jedoch für ein einzelnes Zeichen
F6	wie F5, jedoch für ein einzelnes Zeichen
F8	wie F7, jedoch für ein einzelnes Zeichen
F9	Vertauscht die aktuelle Datei (isfile) mit der Hintergrunddatei (fromfile). Erneutes Drücken von F9 vertauscht erneut und stellt damit den alten Zustand wieder her. Diese Funktion ist dann sinnvoll, wenn Sie eine Datei bearbeiten, sich zwischendurch aber mit CRTL-F ein Wort anzeigen lassen. Dann stellt F9 (=fswap) die alte Dateiverteilung wieder her, die sich durch das fix geändert hat.
SHIFT F9	Schaltet auf die Kommentartexte (shadowscreen) um und beim nächsten Drücken wieder zurück

F10 Legt den aktuellen Screen kurz beiseite, um ihn dann mit einem Druck auf F9 wieder bearbeiten zu können. Sollte Ihnen das volksFORTH eine der Kopierfunktionen copy oder convey mit der Meldung "TARGET BLOCK NOT EMPTY" verweigern, weil isfile und fromfile unterschiedlich sind, so sorgt F10 wieder für klare Verhältnisse.

1.8 Arbeiten mit dem S8-FORTH

Für die folgenden Beispiele wurde das Super8-Entwicklungsboard mit der Maskenversion des Super8-FORTH und einem 32KByte-RAM verwendet. Das Terminalprogramm T4.COM übernahm die Befehlsinterpretation und protollierte gleichzeitig die Ausgaben. Das erzeugte Bitimage wurde jeweils in ein 32K-EPROM gebrannt und ebenfalls auf dem Entwicklungsboard getestet.

1.8.1 Anschluß und direkte Programmeingabe

Falls man über eine Harddisk verfügt, so sollte die Arbeitsdiskette in ein eigenes Verzeichnis kopiert werden. Falls man dabei nicht genügend Platz auf der Harddisk hat, so können die nicht benötigten Terminalprogramme und das doch sehr große Sourcefile S8V10.SCR wieder gelöscht werden.

Sobald man die serielle Schnittstelle mit dem freien Port des PC verbunden und das zugehörige Terminalprogramm T1.COM bis T4.COM gestartet hat, kann das Arbeiten mit dem Super8 beginnen. In den nachfolgenden Listings sind Kommentare in doppelten Klammern eingeschlossen. Eingaben wurden zur besseren Kennung unterstrichen.
Das nachfolgende Beispiel wurde aus dem als Einführung sehr gut geeignete Buch "Programmieren in FORTH" von Leo Brodie entnommen. Obwohl es sich oft an spezielle FORTH-Versionen anlehnt (FORTH83 und LMI-FORTH) kann es wegen des FORTH83-Standards auch beim S8-FORTH verwendet werden.

((S8-Board mit Strom versorgen oder Reset drücken))

```
 Zilog Super8-FORTH V1.0
  ok                  (( Return drücken ))
15 spaces                    ok
42 emit * ok
15 spaces  42 emit  42 emit                      ** ok
: stern  42 emit ;  ok
stern * ok
cr
 ok
cr stern cr stern cr stern
```

```
*
*
* ok
: rand   cr 30 spaces ;  ok
: punkt  rand stern ;  ok
: sterne  0 DO stern LOOP ;  ok
5 sterne ***** ok
20 sterne ******************** ok
: balken  rand 5 sterne ;  ok
balken punkt balken punkt punkt
                                *****
                                *
                                *****
                                *
                                * ok
: f balken punkt balken punkt punkt cr ;  ok
f
                                *****
                                *
                                *****
                                *
                                *
 ok
xlerb "XLERB" ?  ???
```

Falls immer wieder neue Befehl eingegeben werden oder bei einem Fehler die alten Befehle neu definiert werden, wird irgendwann der Speicherplatz des Systems erschöpft oder durch fehlerhafte Befehle das FORTH zerstört sein. Dann hilft nur ein Neustart des FORTH mit dem Löschen aller im RAM befindlichen Befehle und Daten. Dieser kann entweder mit der Return-Taste oder durch Eingabe des Befehl COLD veranlaßt werden.

```
words
F  BALKEN  STERNE  PUNKT  RAND  STERN  COLD  'COLD  ABORT
... (( weitere Befehle )) ...
VDP  DP  VOC-LINK  RAMORG  LEAVE  ?BRANCH  BRANCH  EXECUTE
EXIT   ok
cold
 Zilog Super8-FORTH V1.0
words
COLD  'COLD  ABORT  'ABORT  IDENT  QUIT  INTERPRET  SAVESYSTEM
... (( weitere Befehle )) ...
LEAVE  ?BRANCH  BRANCH  EXECUTE  EXIT   ok
```

Da das S8-FORTH bei einem Fehler den Kompilermodus nicht verläßt, muß mit [der Interpretermodus erzwungen werden. Nach REVEAL ist dann der unvollständig definierte Befehl sichtbar und kann mit FORGET Name gelöscht werden. Diese Methode dient aber nur als Notausstieg, da durch einen Fehler nichts verändert wird und ein Tippfehler durch die Eingabe des richtigen Befehls behoben werden kann.

1.8.2 Laden vorhandener Programme

Beim Start des Terminalprogramms wird automatisch das auf der Diskette befindliche File TEST.SCR geöffnet. Da nach einem Reset auch der Diskpuffer des Super8-FORTH gelöscht ist, kann man mit dem Befehl LOAD einen oder mit THRU mehrere Screens aus diesem File laden. In dem nachfolgenden Beispiel wird der Screen 7 geladen. Der danach verfügbare Befehl TABELLE gibt eine Tabelle mit Multiplikationswerte aus.

((S8-Board mit Strom versorgen oder Reset drücken))

```
 Zilog Super8-FORTH V1.0
7 load  ok
tabelle
             Multiplikationstabelle:
 *  |  0 |  1 |  2 |  3 |  4 |  5 |  6 |  7 |  8 |  9 |
---|----|----|----|----|----|----|----|----|----|----|
  0|  0 |  0 |  0 |  0 |  0 |  0 |  0 |  0 |  0 |  0 |
---|----|----|----|----|----|----|----|----|----|----|
  1|  0 |  1 |  2 |  3 |  4 |  5 |  6 |  7 |  8 |  9 |
---|----|----|----|----|----|----|----|----|----|----|
  2|  0 |  2 |  4 |  6 |  8 | 10 | 12 | 14 | 16 | 18 |
---|----|----|----|----|----|----|----|----|----|----|
  3|  0 |  3 |  6 |  9 | 12 | 15 | 18 | 21 | 24 | 27 |
---|----|----|----|----|----|----|----|----|----|----|
  4|  0 |  4 |  8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 |
---|----|----|----|----|----|----|----|----|----|----|
  5|  0 |  5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 |
---|----|----|----|----|----|----|----|----|----|----|
  6|  0 |  6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 |
---|----|----|----|----|----|----|----|----|----|----|
  7|  0 |  7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 |
---|----|----|----|----|----|----|----|----|----|----|
  8|  0 |  8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 |
---|----|----|----|----|----|----|----|----|----|----|
  9|  0 |  9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 |
---|----|----|----|----|----|----|----|----|----|----| ok
```

Da man oft das gesamte File laden will, wird meistens in Screen 1 die Anweisungen zum Laden des restlichen Files untergebracht. Es genügt dann die Befehlsfolge 1 LOAD zum Laden aller Beispiele von TEST.SCR oder des gesamten Assemblers (File S8.ASM).

1.8.2 Erstellung eines ladbaren Programmes

Falls ein neues Programm geschrieben werden soll, so wird dafür ein neues File angelegt. Dies ist ebenfalls mit dem Terminalprogramm möglich, weil bei einem nicht gefundenen Filename automatisch ein neues File mit einer Länge von 3 Screens (Nummern 0..2) erzeugt wird. Alle drei Screens sind mit Leerzeichen gefüllt.

Weitere Screens können dann durch die Tastenkombination CTRL+K N beim Editieren dieses Files vor dem aktuellen Screen eingefügt werden. Da neue Befehle meistens am Fileende angehängt werden, sollte man den letzten Screen nicht für Programme, sondern nur zur Ablage zusätzlicher Informationen verwenden. Durch den Befehl \\ am Anfang dieses Screens wird die Ausführung des Kommentars verhindert.

Eine Eigenheit des FORTH ist, das der Screen 0 nicht für Programme verwendet werden kann. Deshalb enthält in den Beispielen dieser Screen Informationen wie Erstellungsdatum, Autorenname oder kurzen Bedienungshinweise.

Also steht nach dem Öffnen eines Files nur der Screen 1 für das Programm zur Verfügung. Falls es sich nur um ein Testwort handelt, ist dies sicher ausreichen. Bei längeren Programmen sollte, wie auch bei TEST.SCR , der Screen 1 nur die Anweisung zum Laden der gewünschten Befehle (z.B. DECIMAL 2 7 THRU) enthalten und die einzelnen Wörter in den eingefügten Screens 2 ... abgelegt werden. Bei der inkrementalen Programmierung wird jeder neu geschriebene Screen sofort mit n LOAD geladen und getestet. Selbst wenn das Super8-FORTH durch fehlerhafte Befehle ins Computer-Nirwana entschwindet, sind keine Daten verloren. Nach einem Reset und (natürlich nach Korrektur des Fehlers mit dem Editor) 1 LOAD kann man das Programm weiterentwickeln.

((Terminalprogramm starten))
((Nach ALT+L Filename MEINPRG.SCR eingeben, (NEW) wird angezeigt))
((Nach ALT+E kann im Editor folgendes Programm eingegeben werden))

((Screen 0))

```
\\ Ausgabe der Zahlen 0-9                          ( 08.03.91/KK )

Erstellt:       08.03.91  Klaus Kohl
System:         Super8-FORTH V1.0 mit T4.COM

                    Hinweis:
  0..16     Ausgabe der Zahlen 0 bis 16
  HEX.      Zahlen hexadezimal ausgeben
  DEC.      Zahlen dezimal ausgeben
  OCT.      Zahlen oktal ausgeben
```

((Screen 1))

```
\ Befehle zur Ausgabe der Zahlen 0..16          ( 08.03.91/KK )

 DECIMAL

  : 0..16       ( -- ) ( Zahlen 0 bis 16 ausgeben )
   17 0  DO  i .  LOOP ;

  : HEX.        ( -- ) ( Zahlen hexadezimal ausgeben )
   base push   hex        0..16  ;

  : DEC.        ( -- ) ( Zahlen dezimal ausgeben )
   base push   decimal    0..16  ;

  : OCT.        ( -- ) ( Zahlen octal ausgeben )
   base push   8 base !   0..16  ;
```

((Screen 2))

```
\\ Hinweise zum Programm                           ( 08.03.91/KK )
```

Die aktuelle Zahlenbasis ist im FORTH in der USER-Variable BASE gespeichert. Sie kann ohne Veränderung der internen Bearbeitung geändert werden, da sie nur für die Ein- und Ausgabe berücksichtigt wird.

Um nach den Befehlen HEX. bis OCT. wieder den alten Inhalt der USER-Variable herzustellen, wird der Befehl PUSH verwendet. Er speicherte Adresse und Inhalt der Variable auf dem Returnstack und schreibt ihn beim Verlassen des Befehls automatisch wieder zurück.

((Mit ESC-Taste wird dann der Editor wieder verlassen))
((Danach muß das Board aktiviert und das File geladen werden))
((Bei Fehler wieder Editor aufrufen und Screen korrigieren))
((Die Screennummer des Fehlers steht in der USER-Variable SCR))
((Vor dem Laden entweder COLD oder FORGET 0..16 eingeben))

```
 Zilog Super8-FORTH V1.0
1 load  ok
```

((Nach dem Laden folgt immer der Test der einzelnen Befehle))

```
0..16 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  ok
hex. 0 1 2 3 4 5 6 7 8 9 A B C D E F 10  ok
dec. 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  ok
oct. 0 1 2 3 4 5 6 7 10 11 12 13 14 15 16 17 20  ok
0..16 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  ok
```

1.8.4 Erzeugung eines EPROM's mit Zusatzbefehlen

Das Super8-FORTH hat die Fähigkeit, ein zwischen Adresse $2000 und dem RAM-Anfang liegendes Bitimage zu erkennen. Dieses Bitimage muß an einer 1K-Speichergrenze beginnen und kann entweder schon an der gewünschten Adresse liegen (wenn die Kompilierung mit RAM an dieser Adresse erfolgte) oder eine Kopie des RAM-Bereiches sein. Nur das Image mit der höchsten Anfangsadresse wird verwendet.

Stimmt die Anfangsadresse des Bitimage, so bleiben die Befehle dieses Image an dieser Adresse und nur die Variablen, der USER-Bereich und die Interruptvektoren werden an das Ende des RAM kopiert. Am Anfang des RAM sind nach einem Reset nur 80 Byte für die Ablage der aktuellen Systemstatus durch SAVE reserviert. Neue Befehle werden dahinter abgelegt. Bei SAVESYSTEM werden in das Bitimage nur die neuen Befehle übernommen. Da aber alle Zeiger weiterhin auf die im alten Image vorhandenen Befehle weisen, muß auch der alte Inhalt des EPROM's verfügbar sein.

Falls die Kompilierungsadresse dieses Image von der momentanen Adresse abweicht, so geht das Super8-FORTH davon aus, daß die Daten aus einem RAM stammen. Das gesamte Image wird dann wieder an diese Adresse kopiert und die ebenfalls gespeicherten Daten des Variablen-, Task- und Interruptbereiches an das Speicherende gebracht. Dadurch hat man nach einem Reset den gleichen Systemzustand wie bei SAVESYSTEM . Weitere Befehle können unmittelbar nach der Endadresse der letzten Definition abgelegt werden. Falls danach wieder SAVESYSTEM eingegeben wird, sind im Bitimage auch die alten Befehle wieder vorhanden.

Das folgende Beispiel wurde bei der Erzeugung des auf Diskette gespeicherten S8.IMG mitprotokolliert. Wenn dieses Image in ein EPROM übernommen und in ein

Super8-Board gesteckt wird, muß die Anfangsadresse des RAM's bei $8000 liegen. Man kann aber ohne Probleme auch mit anderer RAM-Anfangsadresse arbeiten. Es ist dann der hier beschriebene Ablauf nachzuvollziehen. Die Files S8.ASM und ZUSATZ.SCR müssen sich im aktuellen Verzeichnis befinden.

((Terminal starten, Reset am Super8-Board auslösen))

```
Zilog Super8-FORTH V1.0
```

((File: S8.ASM nach Drücken von ALT+F wählen))

```
empty-buffers 1 load
```

((Die >>> dienen nur zur Infomation, daß etwas geladen wird))

```
, is redefined
# is redefined
@ is redefined
BL is redefined
Length of Assembler: 5531 Bytes
 ok
```

((File: ZUSATZ.SCR nach Drücken von ALT+F wählen))

```
empty-buffers 1 load  ok
RP! is redefined
SAVESYSTEM is redefined
```

((Imagefile: S8.IMG mit ALT+I wählen))

```
hclear savesystem  ok
```

((Terminalprogramm mit ALT+X beenden))

1.8.5 Erzeugung eines Autostart-EPROM's

Die typische Anwendung des Super8 ist natürlich nicht die ständige Neuprogrammierung in FORTH. Normalerweise wird ein Programm einmal geschrieben und danach sollte beim Einschalten der Platine automatisch ein bestimmtes Programm gestartet werden. Das nachfolgende Beispiel zeigt, wie auch diese Aufgabe mit dem Super8-FORTH erledigt werden kann.

Ein kleines Programm soll die von der seriellen Schnittstelle kommenden Daten an Port 4 ausgegeben. Dazu wird zuerst der Port 4 auf Ausgang geschaltet und danach in einer Endlosschleife die von der UART kommenden Daten ausgegeben. Damit das FORTH auch tatsächlich nach dem Start dieses Programm ausführt, muß der DEFER-Befehl 'ABORT auf dieses Wort umgestellt werden. Um im Terminal zu sehen, daß tatsächlich etwas passiert, wird beim Programmstart eine Einschaltmeldung ausgegeben und jeder empfangene Wert zurück zum Terminal geschickt. Da das Programm in einen Screen paßt, sind die anderen Screens nicht ausgegeben und nur die korrigierte Version des Screen 1 abgedruckt.

((Terminalprogramm starten))
((Neues Screenfile SPOOLER.SCR anlegen und Programm eingeben))
((Hier ist vom vollständige Programm nur Screen 1 abgedruckt))
((Für Test DUP 3 - WHILE und DROP nicht in Klammern setzen))

```
\ SPOOLER
 DECIMAL
  212 CONSTANT P4_DATA          ( Datenregister )
  246 CONSTANT P4_DIR           ( Richtungsregister )
  247 CONSTANT P4_OD            ( Open-Drain Register )
 HEX
  : SPOOLER     ( -- )
   000 P4_DATA SPR!              ( Anfangswert 0 )
   000 P4_OD SPR!                ( Ausgänge als PUSH-PULL )
   000 P4_DIR SPR!               ( Alle Bits auf Ausgang )
   CR  ." Super8-FORTH: Spooler aktiv"  CR
   BEGIN   KEY  ( DUP 3 - )           ( Zeichen holen )
 ( WHILE ) DUP  P4_DATA SPR!  EMIT    ( zum Port und zurück )
   REPEAT ( DROP ) ;                  ( Endlosschleife )

 ' SPOOLER IS 'ABORT             ( für den Autostart )
```

((Dann diesen Screen Laden und Testen))

```
1 load  ok
spooler
Super8-FORTH: Spooler aktiv
dkjflaösslöflasdsdkldkl
 ok
```

((Beim ersten Versuch war die Portrichtung falsch gesetzt))
((Nach der Änderung einfach COLD aufrufen und dann nochmal laden))

```
COLD
 Zilog Super8-FORTH V1.0
1 LOAD  ok
SPOOLER
Super8-FORTH: Spooler aktiv
ABCDEFA
```

((Mit Meßgerät an Port kann die korrekte Funktion getestet werden))
((Danach die Testhilfen mit Klammern als Kommentar markieren))

```
COLD
 Zilog Super8-FORTH V1.0
1 LOAD  ok
decimal savesystem  ok
```

((Das Programm kann nun ins EPROM gebrannt werden))
((Nach dem Einschalten müßte sich der Spooler melden))

```
Super8-FORTH: Spooler aktiv
abcdezhzhzhzhzh
```

((Wird das EPROM entfernt, so hat man wieder das Super8-FORTH))

```
Zilog Super8-FORTH V1.0
```

Auch als Entwickler des Super8-FORTH ist man nicht vor Fehler sicher. Nach dem Brennen des 373 Byte langen Image in das EPROM und Einstecken ins Super8-Board meldete sich weiterhin das FORTH. Nach kurzem Grübeln ist mir wieder eingefallen, daß ich das Image an Adresse $0000 gebrannt hatte. Da aber bis $1FFF das interne ROM des Prozessors reicht, werden diese Bitimage erst ab Adresse $2000 erkannt. Da im EPROM noch reichlich Platz war, konnte durch einfaches Kopieren der Daten nach $2000 und erneutes Einsetzen ins Board das Programm erfolgreich gestartet werden.

1.9 Hinweise auf Fehler im S8-FORTH

Trotz vieler Tests baben sich Fehler in die Maskenversion des S8-FORTH V1.0 eingeschlichen. Die bis jetzt bekannten Fehler haben aber keine Auswirkung auf den FORTH-Kern und können durch Neudefinitionen oder bestimmte Verhaltensregeln behoben werden.

RP@ Der Befehl RP@ (Lesen des aktuellen Returnstackzeigers) verwendet zum Auslesen des entsprechenden Registers die falsche Adressierungsart und liefert deshalb einen verkehrten Wert. Die korrigierte Version dieses Befehls ist in ZUSATZ.SCR enthalten.

SAVESYSTEM Beim Befehl SAVESYSTEM wird der Heap nicht in das Bitimage übernommen. Falls sich beim Speichern der Heap belegt ist, wird sowohl ist die Verkettung der Befehlsnamen als auch der USER-Bereich zerstört.
Nach dem Laden von ZUSATZ.SCR ist der Befehl HCLEAR verfügbar. Er sollte vor SAVESYSTEM aufgerufen werden und Löscht den Heap vollständig.
Da beim Start des S8-FORTH sowohl der Variablenbereich als auch der Heap aus dem EPROM ins RAM übernommen wird, kann auch durch eine Neudefinition des Befehl SAVESYSTEM dieser Fehler behoben werden. Die Anzahl der zu übertragenden Daten muß um die Heapgröße erweitert und nach den Variablen der Heap (Bereich HEAP bis TASKORG @) übertragen werden.

2. Beschreibung des Super8-FORTH

2.1 Allgemeines

In diesem Kapitel werden alle Informationen angegeben, die für die Programmierung des Super8 benötigt werden. Die meisten Angaben sind für die "normale" FORTH-Programmierung nicht nötig. Nur wenn man die internen Registersätze oder Ports für eigene Zwecke benötigt, müssen Überschneidungen mit dem FORTH-Interpreter vermieden werden.

2.1.1 Verwendete Register des Super8

Das Super8-FORTH benötigt zum Arbeiten nur einen Registersatz des Prozessors zur Ablage der Zeiger und als Arbeitsregister. Dazu wird der sowieso nicht für andere Zwecke verwendbare Speicherbereich $C0-$CF des internen RAM verwendet. Jeweils zwei Bytes werden zusammengefaßt zu einem 16Bit-Register oder -Zeiger.

Offset	Bezeichnung	Zweck
0	TEMPX	Ein nicht für Zeiger verwendbares Arbeitsregister
2	FFP	Für Floatingpoint-Stack reservierter Zeiger
4	FUP	Zeiger auf den USER-Bereich dieses Tasks
6	FSP	FORTH-Datenstackzeiger
8	DX	Arbeitsregister 4
10	CX	Arbeitsregister 3
12	BX	Arbeitsregister 2 (meist als NOS verwendet)
14	AX	Arbeitsregister 1 (enthält immer TOS=Top of Stack)

Zusätzlich werden noch zwei weitere 16Bit-Register des S8 herangezogen. Diese Register müssen beim Wechsel auf andere Tasks gerettet werden. Sie sind nicht auf andere Adressen umlegbar, da damit bestimmte Prozessorbefehle verbunden sind.

Adresse	Bezeichnung	Zweck
$D8/9	FRP	FORTH-Returnstackzeiger
$DA/B	IP	FORTH-Instruktionszeiger

Darüber hinaus verwendet das Super8-FORTH nach der Initialisierung auch die serielle Schnittstelle des Prozessors. Dabei wird immer davon ausgegangen, daß die Registerbank 0 eingestellt ist.

2.1.2 Wie funktioniert der FORTH-Interpreter

Der Super8 besitzt drei Befehle, die speziell für die Ausführung einer gefädelten Programmiersprachen wie FORTH gedacht sind:

Opcode	Befehl	Anwendung
$1F	ENTER	SP<--SP-2;@SP<--IP;IP<--PC;PC<--@IP;IP<--IP+2 Die nächste Befehlsadresse wird auf dem Returnstack abgelegt und der erste Befehl des nach ENTER folgenden Unterbefehls ausgeführt.
$2F	EXIT	IP<--@SP;SP<--SP+2;PC<--@IP;IP<--IP+2 Die Rücksprungadresse wird vom Returnstack geholt und der nächste Befehl ausgeführt.
$1F	NEXT	PC<--@IP;IP<--IP+2 Der nächste Befehl wird ausgeführt.

Um die Anwendung dieser drei Befehle zu verstehen, ist es nötig, den Aufbau einiger FORTH-Befehle anzusehen. Es gibt davon nur drei Grundarten:

* Code-Definitionen (sogenannte Primitiv's)
* FORTH-Definitionen (sogenannte High-Level-Definitionen)
* CREATE-Definitionen

Beispiele:

```
CODE DUP    AX PUSHW   NEXT,     ENDCODE
: 2DUP      OVER  OVER ;
0 CONSTANT FALSE
```

In der Codedefinition von DUP folgt nach dem Befehlsheader sofort die entsprechende Assemblerdefinition. Am Schluß des Befehls wird dann durch NEXT, der nächste Befehl aus einer FORTH-Definition aufgerufen und dabei den Instruktionszeiger IP erhöht.

FORTH-Definitionen wie dem 2DUP beginnen ebenfalls mit einem Assemblerbefehl. Diese oben angesprochene Anweisung ENTER bewirkt dann, daß die nächste Adresse der aktuellen Befehlsliste auf dem Returnstack gespeichert wird und der Instruktionszeiger IP auf die nach ENTER folgende Befehlsliste zeigt. Danach wird der erste Befehl dieser Liste ausgeführt. Am Ende der Liste steht dann ein Zeiger auf die Codedefinition EXIT . EXIT enthält den gleichlautenden Assemblerbefehl, der dann die von ENTER gespeicherte Rücksprungadresse wieder holt und den nächsten Befehl in dieser Liste ausführt.

Etwas komplizierter ist die letzte Gruppe von Befehlen. Auch sie müssen mit mindestens einem Assemblerbefehl anfangen. Bei Konstanten ist dieser Befehl ein Unterprogrammsprung zu der Assemblerroutine (CON . Da Unterprogrammaufrufe automatisch die nachfolgende Adresse auf dem Returnstack ablegen, muß die Assembler-

routine diese Adresse vom Returnstack holen und den dort gespeicherten Wert zum Datenstack bringen.

2.2 Speicheraufteilung im S8-FORTH

Bei der Konzeptionierung des S8-FORTH wurde von Anfang an davon ausgegangen, daß immer ein externes RAM mit mindestens 2 KByte zur Verfügung steht. Deshalb wird auch der externe Speicher als Daten- und Returnstack verwendet.

Für den externen 64K-Speicher werden einige Ports des Super8 für die Ansteuerung benötigt. Da Ports 0/1 für Adreß- und Datenleitung und Port 2 und 3 zumindest teilweise für die serielle Schnittstelle und zusätzlichen Steuerleitungen herangezogen werden, bleibt nur Port 4 für eigene Anwendungen. Ein Beispiel dazu ist beim Arbeiten mit dem Super8 angegeben. Alle anderen Informationen über Portbelegung ist der Hardwarebeschreibung des Super8 zu entnehmen. In der Beschreibung des Initialisierungsablaufes ist ebenfalls die Einstellungen der Ports angegeben.

Beim Super8 sind die ersten 32 Bytes durch die Interrupt-Vektoren belegt. Ab der Adresse $0020 beginnt der FORTH-Kern. Die Startroutine initialisiert zuerst die Ports und die Interrupts. Nach Ermittlung des verfügbaren Speicherplatzes und des aktuellen (letzten) Bitimage werden Programmteile aus dem ROM ins RAM und die Systemparameter an das Speicherende (ab $FFB0) kopiert.

2.2.1 Masken-ROM des Super8

$0000 ... $001F	Interrupt-Vektoren zeigen auf $FFD0 ... $FFFF
$0020	JR zur Initialisierungsroutine
$0022	Nummer der FORTH-Version ($00010000 für 1.0)
$0026	Erstellungsdatum (DOS-Format: 07.06.1990 19:30)
$002A	Standardparameter (werden nach $FFB0 kopiert)
$007A	Initialisierungsroutine
...	Kern des Super8-FORTH
...	Kopie des Variablenbereiches
... $1FFF	Kopie der ersten 48 Bytes des Standardtask

Falls beim Durchsuchen des Speichers nach einem Image nichts gefunden wird, so verwendet der Kern seine eigenen Initialisierungswerte ab Adresse $002A. Da das Programm schon an der richtigen Adresse sitzt, bleibt der gesamte RAM-Bereich bis auf 80 Bytes (enhält neue Systemparameter) für Programm, Variablen und Task-Bereiche frei. Am Ende des Masken-ROM's sind noch die Variablen und die verwendeten Daten des Taskbereiches abgelegt.

2.2.2 Bitimage im EPROM

addr + $00/$01	Kennung $A55A
addr + $02 ... + $1F	Systemparameter
addr + $20 ... + $4F	Kopie der Interrupt-Vektoren vom Speicherende
addr + $50 ...	FORTH-Erweiterungen
danach	Variablenbereich und Heap
danach	Je $80 Bytes für jeden Task-Bereich

Ab jeder 1K-Adresse kann ein Bitimage im EPROM stehen. Immer das ROM-Image, mit der höchsten Speicheradresse gibt den aktuellen Systemzustand an. Die 80 Bytes mit den System-Variablen und den Interrupt-Vektoren wird an das Speicherende ab $FFB0 kopiert.

Liegt das Image schon an der richtigen Speicheradresse, so werden nur die ersten 80 Bytes des freien RAM-Speichers für den neuen Header reserviert und der aktuelle Zustand darin gespeichert. Der Wert für DP und RAMORG wird dann entsprechend geändert. Ansonsten wird (ohne Kontrolle, ob dort überhaupt RAM ist) das Programm an seine angegebene Speicheradresse kopiert und die gespeicherten Werte verwendet.

In beiden Fällen werden die nach dem Programm abgelegten Variablenwerte und die Taskbereiche (im EPROM ohne Arbeits- und Stackbereiche) in die vorgesehenen Speicheradressen am Ende des RAM's übernommen.

2.2.3 RAM-Bereich

addr + $00 ...	Kopie der Tabellen vom Speicherende
addr + $50 ... DP @ 1-	Programm
VDP @ ...	Variablen-Bereich (mit DEFER's und Vokabulare)
HDP @ ...	HEAP
LASTTASK @ ...	alle Tasks mit ihren Arbeitsbereichen und Stacks
DISKORG @ ...	Raum für einen 16Bit-Wert und 1024 Zeichen
$FFB0 ...	System-Variable wie DP, ...
$FFD0 ... $FFFF	Interrupt-Bereich (enthält 16 JP INTERROR)

Es muß beim S8-FORTH mindestens 2 KByte RAM am Ende des 64KByte-Programmspeichers existieren. Es ist eine vollständige Dekodierung der Adressen notwendig, da sonst ein viel zu großer RAM-Bereich ermittelt wird.

2.2.4 Task-Bereich

WDP@ ...	Arbeitsbereich für WORD
... PAD-1	Arbeitsbereich für Zahlenstring-Erzeugung

PAD ...	String-Arbeitsbereich
... S0 @	Datenstack (darüber 6 Bytes frei)
S0 16 -	$80 Bytes des Tasks
S0 $40 + ...	Arbeitsbereich für Zeileneingabe
... R0 @	Returnstack

Für jeden der beiden Stacks ist in dieser Version ein Raum für 32 16-Bit-Werte (64 Bytes) reserviert. Über dem Datenstack sind aus Sicherheitsgründen noch 3 Einträge freigehalten.

2.2.5 System-Variablen

$FFB0/1	Kennung $A55A für Anfang des Bereiches	
$FFB2/3	RAMORG	Anfang des RAM-Speichers (Tabelle)
$FFB4/5	VOC-LINK	Zeiger auf PFA des letzen Vokabulars
$FFB6/7	DP	Aktuelle HERE-Adresse
$FFB8/9	VDP	Anfangsadresse des Variablenbereiches
$FFBA/B	VLEN	Gesamtlänge des Variablenbereiches
$FFBC/D	HDP	Anfangsadresse des Heaps
$FFBE/F	TASKORG	Anfangsadresse des zuletzt definierten Tasks
$FFC0/1	LASTTASK	Anfangsadresse des ersten Tasks
$FFC2/3	TASK#	Registeradresse des zuletzt definierten Tasks
$FFC4/5	DISKORG	Anfangsadresse des Diskettenpuffers
$FFC6..$FFC9		Reserviert für SPR@ und SPR!
$FFCA..$FFCF		frei

Darin sind die wichtigsten Informationen des FORTH-Systems abgelegt. Sie werden zusammen mit den Interruptvektoren bei jedem SAVE oder SAVESYSTEM in den reservierten Bereich am Anfang des RAM's kopiert und damit auch in das gespeicherte Bitimage übernommen. Eine genaue Beschreibung dieser Variablen ist im Kapitel 2.6.1 angegeben

2.2.6 Interruptvektoren

$FFD0	IRQ3	reserviert
$FFD3	"	reserviert
$FFD6	"	P32
$FFD9	"	P22
$FFDC	IRQ0	P23
$FFDF	"	P33
$FFE2	IRQ2	P27 / P26 / Counter 0-Zero
$FFE5	IRQ5	P37 / P36 / COunter 1-Zero
$FFE8	IRQ6	UART (Overrun/Framing/Parity)
$FFEB	"	UART (Break/Kontrollzeichen/Wake-Up)

$FFEE	"	UART (Empfang) / P30
$FFF1	"	P20
$FFF4	IRQ1	UART (Zero Count) / P21
$FFF7	"	UART (Senden) / P31
$FFFA	IRQ4	P25 / P24 oder HS0
$FFFD	IRQ7	P35 / P34 oder HS1

Um die Interrupt-Fähigkeit des Super8 auch in der Maskenversion des S8-FORTH verwenden zu können, sind alle Adressen bei $0000 ... $001F auf eine Tabelle am Speicherende umgeleitet worden. Dort stehen dann jeweils 3 Byte zum Aufruf eigener Routinen zur Verfügung. Diese Tabelle ist im Kern mit Aufrufe einer Fehlerbehandlungsroutine belegt, die den Interrupt wieder abschaltet und eine entsprechende Meldung ausgibt.

Die Tabelle wird bei SAVESYSTEM gespeichert und beim Start wieder in das RAM kopiert. Es muß dann nur (z.B. durch Einbindung in 'COLD) der gewünschte Interrupt erlaubt werden. Ein Beispiel dazu ist im Assemblerteil angegeben.

2.3 Ablauf der Systeminitialisierung

Beim Einschalten werden folgende Tätigkeiten bis zur Interpreterschleife durchgeführt. An den mit ">" markierten Stellen kann durch Veränderung des DEFER-Vektors eigene Aktionen eingefügt werden. Bei Unklarheiten muß auf das Sourcelisting in S8V10.SCR und dem Handbuch zum Super8 verwiesen werden:

* RESET startet das Programm bei Adresse $20
* Versionsnummer und Tabelle überspringen
* Interrupt zuerst ein- und danach ausschalten
* Registerbank 0 adressieren
* Arbeitsregister $C0-$CF einstellen
* Port 0/1 für externe 64K-Adressen vorbereiten
* DMA-Handshake abschalten
* Port 2 bis 4 Initialisieren
  - Port 2 und 3 bis auf P30 auf Ausgang mit Wert 0
  - Port 4 auf Push/Pull-Eingang
* Kein schneller Interrupt; Reihenfolge IRQ0...IRQ7; alle Interrupts maskieren
* UART auf 9600 Baud/8 Bit
* externer Stack aktivieren und auf $FFB0 setzen (Ende des Diskbereiches)
* Ab $FFFF den RAM-Anfang suchen (in Register BX merken)
* Ab RAM-Anfang an jeder 1K-Grenze die Kennung $A55A suchen
  (Diese Kennung wird am Anfang jedes Bitimage abgelegt)
  (Wenn Kennung nicht gefunden wird, dann Tabelle ab $002A verwenden)
* Variablen und Interruptvektoren aus Tabelle nach $FFB0-$FFFF bringen

* Wenn Programmadresse nicht stimmt, dann EPROM in's RAM kopieren
* Variablenbereich ins RAM kopieren
* Alle Taskbereiche ins RAM kopieren
* Daten- und Returnstack auf Anfangsadressen des $C0-Tasks setzen
* IP auf Highlevel-Teil von COLD setzen und NEXT ausführen

(FORTH-Befehl COLD)
* FIRST mit $7FFF besetzen und damit leeren Diskpuffer kennzeichnen
* FORTH als einziges Vokabular eintragen (ONLYFORTH)
> DEFER-Befehl 'COLD (mit NOOP belegt) aufrufen
* ABORT aufrufen

(FORTH-Befehl ABORT)
* Datenstack löschen; Interpretermodus; Variable ?HEAD auf 0 setzen
> DEFER-Befehl 'ABORT (mit NOOP belegt) aufrufen
* Einschaltmeldung mit IDENT ausgeben
* QUIT aufrufen

(DEFER-Vektor QUIT ist auf nachfolgenden Befehl gesetzt)
* Returnstack löschen
* Endlosschleife des FORTH-Kerns starten
  - Bis zu 80 Zeichen mit EXPECT nach TIB holen
  - Eingabe mit INTERPRET interpretieren/kompilieren
  - "OK" oder "] " ausgeben (je nach STATE)

Beim Start des FORTH werden zwei DEFER-Vektoren aufgerufen. 'COLD dient zur Initialisierung spezieller Hardware oder des Interrupts und wird deshalb nur einmal beim Start des Systems benötigt. 'ABORT ist für die Vorgabe eigener Programme gedacht. Durch Veränderung können sogenannte Autostart-Programme erzeugt werden. Der DEFER-Vektor QUIT wird bei unveränderten 'COLD und 'ABORT oder nach einem Fehler aufgerufen. Dadurch hat man für die Fehleranalyse das gesamte FORTH-System zur Verfügung. Es ist auch die Vorgabe einer eigenen Fehlerbehandlungsroutine in ERRORHANDLER möglich.

2.4 Fehlerbehandlung

Wenn im S8-FORTH ein Fehler auftritt, so wird normalerweise die entsprechende Fehlernummer auf den Stack gelegt und ERROR aufgerufen. ERROR überprüft diese Nummer und führt bei einem Wert ungleich 0 die Routine aus, dessen Codefeldadresse in der USER-Variablen ERRORHANDLER gespeichert ist.

Nach dem Start des FORTH-Kerns zeigt ERRORHANDLER auf die Standardroutine (ERROR . Dieser Befehl gibt den bei WDP@ abgelegten String (meistens der

aktuelle Befehlsname) in Anführungszeichen und die Fehlernummer aus. Beim Wert $FFFF wird statt der Fehlernummer der ebenfalls auf dem Datenstack abgelegte String ausgegeben. Wurde gerade ein Screen kompiliert, so wird die Screennummer (in SCR) mit aktueller Position (in R#) gespeichert. Der Interpretermodus wieder aktiviert. Da im normalen Eingabemodus der Datenstack nicht gelöscht und auch der Modus nicht verändert wird, sind Tippfehler durch Neueingabe des Befehls behebbar. Dazu kehrt (ERROR durch Aufruf von QUIT wieder in die Interpreterschleife zurück.

Nicht über diese Fehlerbehandlungsroutine läuft ABORT . Bei ABORT wird nach dem Löschen des Datenstacks sofort QUIT ausgeführt. Was bei Aufruf von ABORT geschieht, ist bei der Systeminitialisierung nachzulesen.

NOTFOUND ist eine DEFER-Routine, die für eigene Interpretation unbekannter Befehle verwendet werden kann. Sie zeigt normalerweise auf den versteckten Befehl (NOTFOUND bei Adresse ' NOTFOUND 5 + und gibt nur "???" aus.

Beim Start des Super8-FORTH zeigen alle Interruptvektoren auf eine Fehlerbehandlungsroutine, die alle Interrupts abschaltet, die Adresse nach dem Interruptvektor auf den Datenstack legt und dann die Fehlernummer $0015 ausgibt. Bei Verwendung interruptbetriebener Ein-/Ausgaben kann es hier zu Probleme kommen. Man sollte deshalb bei jedem Aufruf von KEY oder KEY? dafür sorgen, daß der Interrupt eingeschaltet bleibt.

Fehlernummern:

	$0000	Kein Fehler (ERROR springt zurück)	
	$0001	Stackunterlauf	
(	$0002	Stacküberlauf	)
(	$0003	Returnstackunterlauf	)
(	$0004	Returnstacküberlauf	)
	$0005	Zu wenig Parameter (von ?DEPTH)	
	$0006	Name erwartet	
	$0007	COMMAND!-Fehler	
	$0008	Dictionary voll	
	$0009	Falsche Blocknummer	
	$000A	Befehl unstrukturiert	
(	$000B	Adresse zu weit entfernt	)
	$000C	DEFER nicht gesetzt (von NOTFOUND)	
(	$000D	Unbekannter Opcode	)

	$000E	Befehl darf nur kompiliert werden	
(	$000F	Kein Kurzliteral erlaubt	)
	$0010	Befehl im geschützten Bereich	
	$0011	Befehl kann nicht vergessen werden	
(	$0012	DOES> ist nicht erlaubt	)
	$0013	Befehl ist kein DEFER-Wort	
	$0014	Befehl hat kein Codefeld	
	$0015	Nicht definierter Interupt aufgetreten	
	$0016	Zu viele CONTEXT-Vokabulare	
	$FFFF	Stringadresse des Fehler liegt auf dem Stack	

Die in Klammern gesetzten Fehlernummern werden vom S8-Kern nicht verwendet. Beim Laden von ZUSATZ.SCR wird die Fehlerbehandlung auf eine Routine umgeleitet, die statt der Fehlernummer die englische Fehlermeldung ausgibt. Natürlich können durch Änderungen im File anderssprachige Fehlermeldungen erzeugt werden.

2.5 Das Fileinterface

Als das FORTH entwickelt wurde, existierten auf den Computer nur Grundkonzepte eines Massenspeichers. Falls Disketten verfügbar waren, so wurden sie als ein Medium ohne Unterscheidung einzelner Files durchgehend beschrieben. Da schon damals eine Blockgröße von 1024 Zeichen verwaltet wurden, hat man für FORTH eine Screengröße von 16 Zeilen mit je 64 Zeichen verwendet. Die Screens selbst werden von 0 an durchnummeriert.

Bis heute hat sich diese Verwaltung im FORTH83-Standard erhalten. Die nachfolgenden Befehle sind in diesem Standard erwähnt und wurde im S8-FORTH beibehalten.

`BLOCK`	Holt einen Screen vom Massenspeicher (liefert Adresse)
`BUFFER`	Liefert Adresse des gewünschten Screens (Inhalt undefiniert)
`UPDATE`	Setzt UPDATE-Flag des zuletzt gewünschten Screens
`SAVE-BUFFERS`	Schreibt den geänderten Screen zurück
`FLUSH`	Wie SAVE-BUFFERS EMPTY-BUFFERS
`LOAD`	Angegebenen Screen interpretieren/kompilieren

Darüber hinaus sind noch folgende sinnvollen Befehle für diese Gruppe in den S8-Kern aufgenommen worden.

EMPTY-BUFFERS	Löscht Screen im Speicher (ohne Zuürckschreiben)
THRU	Screen n1 bis n2 interpretieren/kompilieren
-->	Nächsten Screen interpretieren/kompilieren
R/W	Grundroutine zum Laden/Speichern eines Screens

Nun zur Realisierung dieser Befehle im S8-FORTH: Im Speicher ist unterhalb der Systemvariablen und den Interruptvektoren ein Feld mit insgesamt 1026 Bytes angelegt. Die Anfangsadresse dieses Feldes liefert FIRST . Die ersten zwei Bytes in diesem Feld werden für die aktuelle Screennummer (Bit 0-14) und dem UPDATE-Flag (Bit 15) verwendet. Die nachfolgenden 1024 Bytes enthalten die Zeichen des entsprechenden Screens.

Bei Start des FORTH oder nach EMPTY-BUFFERS steht ab FIRST der Wert $7FFF. Da normalerweise keine 32767 KByte für ein File benötigt wird, kennzeichnet $7FFF einen nicht belegten Diskpuffer. Die Werte 0-32767 ($7FFE) geben an, daß sich der entsprechende Screen im Diskpuffer befindet. Der Befehl UPDATE setzt Bit 15 in dieser Nummer. Dadurch wird das S8-FORTH veranlaßt, vor dem Laden anderer Screennummer den aktuellen Screen auf Diskette zurückzuschreiben.

Es ist im S8-FORTH darauf zu achten, daß immer nur ein Diskpuffer verfügbar ist. Ein mit UPDATE markierter Diskpuffer wird erst dann zurückgeschrieben, wenn durch BLOCK oder BUFFER ein neuer Diskpuffer angefordert oder durch SAVE-BUFFERS oder FLUSH eine Speicherung erzwungen wurde.

Da auf dem Super8-Board kein Interface für Diskette oder Harddisk vorhanden ist, erfolgt die Realisierung des Massenspeichers ebenfalls über die serielle Schnittstelle. Dazu werden neben der gewohnten Datenübertragung mit einem Escape-Zeichen ($1B) Befehle vom S8-FORTH an das Terminalprogramm geschickt. Transparent für den Anwender verarbeitet das Programm diese Befehle und überträgt Daten aus dem aktuellen Arbeitsfile zum S8-Board oder empfängt Daten/Bitimages. Damit der Anwender auch erkennt, daß die Anlage arbeitet, werden vom Terminalprogramm in der Anzeige ">" oder "<" ausgegeben. Da aber bei der Befehlsbearbeitung ein Protokoll zwischen S8-FORTH und Terminalprogramm abläuft, stören Tastendrücke während dieser Datenübertragung. Man hat aber dadurch den Vorteil, daß das Laden eines Files vorzeitig abgebrochen werden kann. Der ASCII-Wert der gedrückten Taste wird dann als Fehlernummer behandelt.

2.5.1 Zusatzbefehle über die serielle Schnittstelle

Alle Befehle für das Fileinterface werden durch den S8-Befehl COMMAND! eingeleitet. Er erwartet neben der Befehlsnummer (8 Bit) einen weiteren 16Bit-Parameter.

Falls schon ein Zeichen ansteht, so werden beide Werte vom Datenstack entfernt und nur die Fehlernummer $0007 zurückgeliefert. Ansonsten wird zuerst das Escape-Zeichen ($1B), dann Befehlsnummer und der 16Bit-Parameter (Highbyte zuerst) gesendet. Das als nächstes emfangene Zeichen wird dann als Fehlernummer behandelt und dem aufrufenden Befehl zurückgegeben.

Abhängig von der Befehlsnummer ist dann der Ablauf der nachfolgenden Datenübertragung unterschiedlich und wird im nachfolgenden Abschnitt genauer beschrieben. Im S8-Kern werden nur die Befehle $01 (schreiben), $02 (lesen) und $03 (Image schreiben) verwendet.

2.5.2 Der Befehlsinterpreter des Terminalprogrammes

Das Terminalprogramm erkennt die Befehlsnummern $00 bis $03 und $1B. ">" kennzeichnet eine Datenübertragung vom S8-FORTH zum Terminal und "<" die Übertragung vom Terminal zum S8-FORTH. Es ist hier das vollständige Protokoll abgedruckt. Bei den Befehlen $01 bis $03 wird die Übertragung von Befehlsnummer, 16Bit-Parameter und der Fehlernummer durch den S8-Befehl COMMAND! abgewickelt.

$00 Escape-Zeichen ausgeben.
>1 Escape-Zeichen $1B
>1 Befehlsnummer $00
Im Kern wird dieser Befehl nicht verwendet. Das Terminal gibt das ESC-Zeichen an die Anzeige weiter. Abweichen von den anderen Befehlen wird keine Fehlernummer zurückgeliefert.

$01 Screen vom S8-FORTH zum Terminal übertragen
>1 Escape-Zeichen $1B
>1 Befehlsnummer $01
>2 Nummer des Screens
<1 Fehlerflag (Abbruch bei Werten ungleich 0)
>1024 Screeninhalt
Die Übertragung des Screeninhalts erfolgt nur dann, wenn die Fehlernummer 0 ist. Ist das File nicht geöffnet oder die Blocknummer nicht verfügbar, so wird Fehlernummer 9 übertragen. Nach dem Emfang der Daten schreibt das Terminalprogramm diesen Screen sofort auf Diskette.

$02 Screen vom Terminal zum S8-FORTH übertragen
>1 Escape-Zeichen $1B
>1 Befehlsnummer $02
>2 Nummer des Screens
<1 Fehlerflag (Abbruch bei Werten ungeleich 0)

<1024 Screeninhalt
Hier wird der Screen von Diskette zum S8-FORTH übertragen. Falls kein File geöffnet oder die angeforderte Screennummer zu hoch ist, wird nur die Fehlernummer 9 zurückgeliefert.

$03 Bitimage vom S8-FORTH zur Diskette schreiben
>1 Escape-Zeichen $1B
>1 Befehlsnummer $03
>2 n = Länge des Datenblockes in Bytes
<1 Fehlerflag (immer 0)
>n Bitimage
Bei der Speicherung eines Bitimage vom S8-FORTH auf Diskette wird der Datenblock zuerst in das RAM des PC geschrieben. Erst nach dem vollständigen Emfang wird ein neues File angelegt und das Bitimage geschrieben. Dadurch wird verhindert, daß beim Schreiben der Daten auf Diskette einzelne Bytes verlorengehen. Der Aufbau der Daten ist im Kapitel 2.2 nachzulesen oder kann dem Befehl SAVESYSTEM des Sourcelisting entnommen werden.

$1B wie $00

2.6 Weitere Informationen zum S8-FORTH

Falls man schon mit einer FORTH83-Implementierung wie F83, volksFORTH oder LMI-FORTH gearbeitet hat, wird man auch sehr schnell mit dem S8-FORTH vertraut sein. Es gibt aber weitere Informationen, die für das Arbeiten sehr nützlich sind.

2.6.1 Bedeutung der System-Variablen

Im Speicherbereich ab $FFB0 bis $FFCF sind die wichtigsten Variablen des S8-FORTH abgelegt. Sie sind unabhängig von jedem Task und kennzeichnen die aktuelle Speicherverteilung. Es sollte möglichst nur durch den Standardtask auf diese Wert zugegriffen werden.

$FFB0/1 Kennung $A55A für Anfang eines Bitimage
Da die Kombination $A55A am Anfang eines 1KByte-Speicherblockes sehr unwahrscheinlich ist, wird diese Kennung zur Identifizierung des Bitimage herangezogen. Da am Anfang des RAM's eine Kopie des Speicherbereiches ab $FFB0 (bis $FFFF) angelegt wird, setzte man einfach die Kennung $A55A auch in $FFB0.

$FFB2/3 RAMORG
RAMORG enthält die Anfangadresse des RAM-Speichers. Bei SAVESYSTEM werden Systemvariablen und Interruptvektoren an

den RAM-Anfang kopiert und dann der verwendete RAM-Bereich übertragen. Beim Start des S8-FORTH wird durch den Wert in RAMORG erkannt, ob das Bitimage schon an der richtigen Speicheradresse steht. Falls dies nicht der Fall ist, so wird davon ausgegangen, daß es wieder in den alten Speicherbereich kopiert werden muß. Natürlich muß sich dort dann wieder RAM befinden.

$FFB4/5 VOC-LINK

In VOC-LINK wird die Parameterfeldadresse des zuletzt definierten Vokabulars aufbewahrt. Der weitere Aufbau dieser Verkettung ist im Kapitel 2.6.4 zu finden.

$FFB6/7 DP

Der Dictionary-Pointer und damit die Adresse des ersten freien Bytes nach dem Programm ist in DP gespeichert. Der Bereich von RAMORG bis DP muß durchgehend mit RAM belegt sein.

$FFB8/9 VDP

In VDP wird die Anfangsadresse des Variablenbereiches gespeichert. Da sich diese Adresse bei Veränderungen im HEAP oder bei zusätzlichen Variablen, Vokabulare oder DEFER-Befehle ändern, dürfen Variablenadressen nicht als Konstanten verwendet werden.

$FFBA/B VLEN

Die Gesamtlänge des Variablenbereiches ist in VLEN abgelegt. Bei neuen Variablen wird dieser Wert als Offset verwendet und um 2 erhöht. Gleichzeitig muß aber der gesamte Speicherbereich der Variablen nach unten verschoben und VDP entsprechend korrigiert werden. Diese Tätigkeiten werden durch die Befehle VHALLOT oder V, durchgeführt.

$FFBC/D HDP

Ein Heap dient der Ablage von später nicht mehr benötigten Befehlsnamen oder Befehle. Die Anfangsadresse dieses Heap ist in HDP gespeichert. Da der Heap von SAVESYSTEM nicht gespeichert wird, muß er vorher durch HCLEAR (in ZUSATZ.SCR) gelöscht werden.

$FFBE/F TASKORG

Die Anfangsadresse des Standardtask ist in TASKORG gespeichert. Gleichzeitig zeigt diese Adresse auch das Ende des Heap an.

$FFC0/1 LASTTASK

Die Speicheradresse des zuletzt definierten Tasks wird in LASTTASK abgelegt. Die Verkettung zu den weiteren Tasks wird aus der Adresse des in den ersten drei Bytes eines Tasks angegebenen Sprungs (JP addr) oder Vergleiches (CP r,r) bei aktiven Tasks gelesen.

$FFC2/3 TASK#

Da zu jedem Task auch ein Registersatz gehört, muß die Registeradresse des zuletzt definierten Tasks in TASK# gespeichert werden.

$FFC4/5 DISKORG

Normalerweise ist ein Arbeitsbereich von 1026 Bytes für den Diskpuffer reserviert. Durch Veränderung der entsprechenden Adresse ist es durchaus möglich, den Diskpuffer vollständig zu löschen oder zusätzlichen Speicher für andere Zwecke zu reservieren. Da aber dazu eine Verschiebung des gesamten Variablen-, Heap- und Taskbereiches notwendig ist, sollte diese Änderung durch eine spezielle Routine abgewickelt werden.

$FFC6..$FFC9Reserviert für SPR@ und SPR!

Bei der Adressierung der Portregister ist die Angabe der Portadresse im Befehl notwendig. Da dies im ROM nicht möglich ist, wird die Adresse in der Befehlssequenz LD ___,___:RET (ab $FFC6) geändert und die Routine danach angesprungen. Diese Routiene ist weder Interrupt- noch Multitaskingfähig.

$FFCA..$FFCF frei

Der Bereich ist für weitere Verwendung reserviert.

2.6.2 Ein-/Ausgabe-Vektorisierungen im S8-FORTH

Beim Start des S8-FORTH wird die UART auf 9600 Baud initialisiert. Alle empfangenen Zeichen landen zuerst in dem Hardwareregister und können dann vom FORTH-Interpreter mit KEY ausgelesen werden. Falls aber weitere Zeichen ankommen, so sind die vorhergehenden Zeichen verloren. Bei der Ausgabe wird solange gewartet, bis das letzte Zeichen vollständig ausgegeben worden ist. Während der Übertragung kann aber das FORTH-Programm weiterlaufen. Eine Besonderheit ist die Einbindung eigener Ein-/Ausgaberoutinen ohne Änderung des S8-Kerns.

Alle vom System verwendeten Eingabe laufen über die Befehle KEY? und KEY . Diese beiden Befehle holen die auszuführenden Codefeldadressen aus einer Tabelle, auf die INPUT zeigt. Im S8-Kern wurde durch den Befehl SIO_INPUT die USER-Variable INPUT auf die nicht sichtbare SIO-Eingabetabelle gesetzt.

```
INPUT  -->  SIO_KEY?
            SIO_KEY
```

Nach dem Laden von ZUSATZ.SCR stehen zwei neue Befehle zur Definition weiterer Eingabevektoren (INVECTOR:) und der Definition eines neuen INPUT-Vektors (INPUT:) zur Verfügung. Die Anwendung soll am Beispiel der im Kern definierten Eingabebefehle erläutert werden.

```
0 INVECTOR: KEY?    ( erster Vektor mit Offset 0 )
2 INVECTOR: KEY     ( zweiter Vektor mit Offset 2 )
```

```
| : SIO_KEY?   ( -- f ) ( Abfrage, ob Zeichen bereit )
   0EC SPR@  1 AND  1 = ;
| : SIO_KEY    ( -- char ) ( Warten, dann Zeichen holen )
   BEGIN  SIO_KEY?  UNTIL   0EF SPR@ ;

INPUT: SIO_INPUT   SIO_KEY?  SIO_KEY  ;
```

Der Befehl SIO_INPUT hat nur die Aufgabe, die USER-Variable INPUT auf die im Befehl folgende Tabelle mit den SIO-Befehlen umzusetzen. Die Vektoren KEY? und KEY sind nur einmal im System notwendig. Erst wenn weitere Eingabebefehle notwendig sind, kann ab Offset 4 die Tabelle vergrößert werden. Es ist aber zu beachten, daß von SIO_INPUT nur die ersten beiden Einträge abgedeckt sind.

Bei der Ausgaben läuft alles über die beiden Befehle EMIT? und EMIT . Auch sie holen die auszuführende Codefeldadresse aus einer Tabelle. Die Anfangsadresse dieser Tabelle ist in OUTPUT gespeichert. Im S8-Kern wurde durch den Befehl SIO_OUTPUT die USER-Variable OUTPUT auf die nicht sichtbare SIO-Ausgabetabelle gesetzt.

```
OUTPUT -->  SIO_EMIT?
            SIO_EMIT
```

Die Definitionen im Kern sind ähnlich wie bei der Eingabe. Auch hier deckt die Tabelle in SIO_OUTPUT nur die ersten beiden Ausgabevektoren ab. Eigenen Vektoren beginnen mit Offset 4 und müssen in dem OUTPUT: -Befehl durch weitere Wörter abgedeckt werden.

```
0 OUTVECTOR: EMIT?  ( erster Vektor mit Offset 0 )
2 OUTVECTOR: EMIT   ( zweiter Vektor mit Offset 2 )

| : SIO_EMIT?   ( -- f ) ( Abfrage, ob Zeichen ausgegeben )
   0EB SPR@  1 AND  1 = ;
| : SIO_EMIT    ( char -- ) ( Warten, dann Zeichen ausgeben )
   BEGIN  SIO_EMIT?  UNTIL   0EF SPR! ;

OUTPUT: SIO_OUTPUT   SIO_EMIT?  SIO_EMIT  ;
```

2.6.3 DEFER-Wörter in S8-FORTH

DEFER-Befehle sind Wörter, die bei ihrem Aufruf erst die Adresse der gewünschten Routine aus dem RAM lesen und dann ausführen. Dadurch hat man die Möglichkeit, die Eigenschaften des Befehls zu verändern.

Die beiden nachfolgenden DEFER-Befehle sind mit NOOP vorbelegt und dienen zum Einbau eigener Befehl in die Initialisierunssequenz des S8-Kerns.

'COLD	(... --)	Frei für Hardwareinitialisierung
'ABORT	(--)	Frei für eigene Befehle

Falls der DEFER-Vektor nicht nur zum Einbau zusätzlicher Befehle bereitsteht, habt er ein vorgegebene Routine im S8-Kern. Aus Platzgründen wurde dabei auf einen eigenen Namen verzichtet. Damit die Adresse dieser Routine auffindbar bleibt, beginnen sie immer 5 Byte hinter dem entsprechenden DEFER-Wort. Falls eigene Routinen dafür eingesetzt werden, sollte man die Bedeutung des Wortes beibehalten und auch den Stack in angegebener Weise verändern.

NOTFOUND	(addr --)	Fehlermeldung für nicht gefundene Worte
NUMBER?	(csa -- ???)	Test auf Zahlenstring
R/W	(addr blk r/w -- f)	Blockübertragung
?CAP	(addr -- addr)	Wandlung in Großschrift
CREATE	(name ; --)	Neuen Befehlsheader anlegen
INTERPRET	(--)	Eingabetext interpretieren/kompilieren
QUIT	(--)	Haupt-Eingabeschleife

Die Änderung des DEFER-Vektors ist mit IS möglich:

```
' NOOP  IS  ?CAP        ( Unterscheidung von Groß-/Kleinschrift )
' ?CAP  5 +  IS  ?CAP   ( Standard-Vektor erzeugt Großschrift )
```

2.6.4 Aufbau der Befehle des S8-FORTH

Für einen Debugger oder zum Auffinden versteckter Wörter ist die Kenntnis des Aufbau einzelner Befehle notwendig. Sie weichen bei den Definitionen vom normalen Aufbau ab, wo innerhalb eines Befehls veränderbare Daten gespeichert werden. Da dies im ROM nicht möglich ist, wird meistens ein Offset in den Variablenbereich abgelegt und erst dort der entsprechende Wert gespeichert.

Allgemeiner Aufbau eines S8-Befehls:

addr+	0/1	Linkfeldadresse	
	2	Namensfeldadresse	
		Bit 7=1	Befehl ist IMMEDIATE
		Bit 6=1	Befehl ist RESTRICT
		Bit 5=1	Befehl ist INDIRECT
		Bit 0..4	Länge des Befehlsnamens
	3..n-1	Befehlsnamen	
	n	Codefeldadresse	
	n+3	Parameterfeldadresse	

Ein Befehl besteht immer aus einem Header und dem Befehlskern. Im Header sind neben Verkettungszeiger und Befehlsname noch einige Bits für die speziellen Eigenschaften des Wortes vorgesehen. Der Befehlskern beginnt mit der sogenannten Codefeldadresse. Da im Super8 dort schon die Assemblerroutine des entsprechenden Befehls stehen muß, wird normalerweise ein Unterprogrammaufruf (CALL addr) dort abgelegt. Ab der Parameterfeldadresse folgen dann die zusätzlichen Daten für den Befehl.

Ist das Indirect-Bit im Header gesetzt (z.B. bei Befehlsname im Heap), so wird in der Codefeldadresse nur die Adresse des tatsächlichen Codefeld abgelegt. Der Befehlskern beginnt dann an einer anderen Adresse (meist im Dictionary) mit dem Codefeld.

CREATE-Definition:

addr+	0/1 ... n-1	wie oben
	n ... n+2	Unterprogrammaufruf der Routine (DIC

Bei der Verwendung von CREATE wird ein neuer Befehlsheader angelegt. Im Codefeld dieser Definition ist ein Unterprogrammaufruf untergebracht. Dieses mit (DIC bezeichnete Assemblerprogramm bringt dann die Parameterfeldadresse, also die nachfolgende Speicheradresse, auf den Datenstack und führt den nächsten Befehl aus.

FORTH-Befehl:

addr+	0/1 ... n-1	wie oben
	n	S8-Befehl ENTER (Opcode $1F)
	n+1/n+2	Codefeldadresse des ersten Befehls
	...	Weitere Befehl bis zum EXIT (von ; abgelegt)

Der Super8 hat einen speziellen Befehl für den Einsprung in die High-Level-Definition eines FORTH-Wortes. Da dieser nur ein Byte belegt, kann unmittelbar danach

die Codefeldadresse des ersten Befehls in diesem FORTH-Wort abgelegt werden. Da : zuerst mit CREATE den neuen Header anlegt, werden die beiden zusätzlichen Bytes wieder freigegeben. Der Befehl >BODY ermittelt deshalb die falsche Parameterfeldadresse und sollte hier nicht verwendet werden.

Konstanten:

addr+	0/1 ... n-1	wie oben
	n ... n+2	Unterprogrammaufruf der Routine (CON
	n+3/n+4	Wert der Konstante

Da Konstanten ihren Wert nicht verändern, kann der Wert auch im ROM abgelegt werden. Die Routine (CON bringt dann bei Aufruf dieser Konstante den Wert zum Datenstack.

Variable:

addr+	0/1 ... n-1	wie oben
	n ... n+2	Unterprogrammaufruf der Routine (VAR
	n+3/n+4	Offset in den Variablenbereich

Da im ROM der Wert einer Variable nicht verändert werden kann, mußte der Speicher dazu ins RAM verlegt werden. Im Parameterfeld einer Variable wird deshalb nur ein Offset gespeichert. Die tatsächliche Adresse wird dann beim Aufruf der Variable durch die Routine (VAR berechnet und darf deshalb nicht als Konstante verwendet werden.

USER-Variable:

addr+	0/1 ... n-1	wie oben
	n ... n+2	Unterprogrammaufruf der Routine (USER
	n+3	Offset in den USER-Bereich

Da nur 128 Bytes im USER-Bereich eines Tasks für Variable vorgesehen sind, genügt ein 8Bit-Offset zur Berechnung der gewünschten Speicheradresse. Dazu addiert die Routine (USER zum Offset die aktuelle Taskadresse.

DEFER-Definition:

	addr+	0/1 ... n-1	wie oben
		n ... n+2	Unterprogrammaufruf der Routine (DEFER
		n+3/n+4	Offset in den Variablenbereich
(		n+5 ...	Standardroutine einiger DEFER-Befehle)

Bei DEFER-Befehlen muß die Codefeldadresse der auszuführenden Routine im RAM abgelegt werden. Es wird deshalb ebenfalls nur der Offset in den Variablen-

bereich abgelegt und die Ausführung der dort angegebenen Routine durch (DEFER veranlaßt. Bei den im vorhergehenden Kapitel erwähnten DEFER-Befehlen mit vorgegebenen Routinen folgt nach dem Offset die entsprechende Definition. Dabei kann es sich sowohl um eine Code- als auch um eine FORTH-Definiton handeln.

Vokabular:

addr+	0/1 ... n-1	wie oben
	n ... n+2	Unterprogrammaufruf der Routine (VOC
	n+3/n+4	Zeiger zum vorgehenden Vokabular
	n+5/n+6	Offset in den Variablenbereich

Da keine weiteren Vokabulare vor einem existierenden Vokabular eingefügt wird, ist der Zeiger auf das vorhergehende Vokabular im Parameterfeld abgelegt. Da aber durchaus neue Befehle an ein im ROM vorhandenes Vokabular (wie FORTH) angehängt wird, mußte der Zeiger auf das zuletzt definierte Wort im Variablenbereich untergebracht werden. Im Vokabular ist deshalb der entsprechende Offset gespeichert. Die vollständige Verkettung von Vokabular und Befehl sieht wie folgt aus:

(Anfang der Vokabularverkettung beginnt mit Systemvariable VOC-LINK)		
	VOC-LINK	Zeiger auf letzte Vokabular-PFA
(Verkettung der Vokabulare)		
	pfa+ 0/1	Zeiger auf nächste Vokabular-PFA
	2/3	Offset (off) in den Variablen-Bereich
(Zeiger auf das letzte Wort eines Vokabulars liegt im Variablenbereich)		
	VDP+off+0/1	Zeiger auf Linkfeld (LFA) des erste Befehl
(Weitere Befehlsverkettung stehen am Anfang jedes Wortes)		
	lfa+ 0/1	Zeiger auf nächste Linkfeld

Hat einer der Zeiger den Wert 0, so ist das Ende der Kette erreicht. Im S8-Kern ist nur das Vokabular FORTH mit den Befehlen COLD bis EXIT implementiert.

CODE-Definitionen:

addr+	0/1 ... n-1	wie oben
	n ...	Assemblerroutine

Bei einer Assemblerdefinition sind keine Zwischenschritte notwendig. Noch im Codefeld steht der erste Assemblerbefehl. Am Ende einer Definition sollte mit NEXT, die Ausführung des nächsten FORTH-Befehls gestartet werden. Es ist darauf zu achten, daß dann im Registerpaar AX der aktuelle TOS (=Top Of Stack) und die anderen, ebenfalls von FORTH verwendeten Register sinnvolle Werte stehen.

2.6.5 Besonderheiten beim ROM-Fähigen S8-FORTH

Wie man schon aus dem Aufbau der einzelnen Befehle erkennt, kompliziert die Verwendung eines ROM die Bearbeitung von Variablen und Datenfelder.

Solange keine Änderungen in den einmal abgelegten Werten einer CREATE-Definition notwendig sind, gibt es keinen Unterschied zu Definitionen in reinen RAM-Systemen. Hier soll als Beispiel ein Definitionswort für 3D-Konstanten vorgestellt werden.

```
: 3D-CONSTANT    ( Definitionswort für 3D-Konstanten )
 CREATE  ,              ,              ,    ( n1 n2 n3 -- )
 DOES>   DUP @  SWAP 2+  DUP @  SWAP 2+  @ ; ( -- n1 n2 n3 )

0 0 0  3D-CONSTANT  NULLVECTOR
0 0 1  3D-CONSTANT  Z-VECTOR
```

Sobald aber die Daten selbst verändert werden müssen, ist die Auslagerung der Werte in den Variablenbereich notwendig. Statt die einzelnen Werte mit , direkt im Codeteil des Befehls unterzubringen, wird meist nur der Offset gespeichert und ein ausreichend großes Datenfeld im Variablenbereich reserviert. Da bei SAVESYSTEM alle Variablen gespeichert werden, ist nach dem Programmstart der alte Wert wieder verfügbar. Die nachfolgende FIELD-Definition erlaubt das Anlegen eines 2D-Datenfeldes mit automatischer Berechnung der Adresse eines einzelnen Elements. Dazu wird die Länge einer Zeile im Codeteil des Befehls gespeichert.

```
: FIELD   ( Definitionswort für 2D-Felder mit 16Bit-Einträge )
 CREATE  OVER ,  VLEN @ ,  * 2* VALLOT      ( fx fy -- )
 DOES>   DUP 2+ >R
         @ *  +  2*
         R> @ +  VLEN @  + ;                ( x y -- addr )

4 4 FIELD 4*4-MATRIX
```

CREATE und DOES> sind ein Paar von Befehlen, die zusammen ein sehr mächtiges Werkzeug zur Erstellung eigener Datentypen sind. In dem vorhergehenden Beispiel dienen sie zur Anlage eines Datenfeldes. Dabei wird bei der Verwendung von FIELD durch CREATE ein neuer Befehl mit nachfolgenden Name 4*4-Matrix angelegt und die Befehle zwischen CREATE und DOES> ausgeführt. Deshalb wird nach dem neuen Befehlsname die X-Länge des Feldes (fx) und der Offset in den Variablenbereich abgelegt. Mit VALLOT sind danach auch noch 2*X*Y Bytes für das Datenfeld reserviert. Wenn der Befehl 4*4-MATRIX aufgerufen wird, müssen die gewünschten X/Y-Koordinaten auf dem Stack liegen. Ausgeführt wird der Teil zwischen DOES> und ; , wobei DOES> noch die Parameterfeldadresse von 4*4-Matrix auf den Stack bringt. Die Adresse des Offsets wird dabei zuerst auf dem Returnstack gespeichert. Durch die Formel (y*fx+x)*2 erhält man den Offset des ge-

wünschten Feldes, zu dem dann noch Variablenoffset und die Anfangsadresse des Variablenbereiches addiert wird.

3. Super8 FORTH ASSEMBLER

3.1 Aufbau des S8-Assemblers

Der Super8™ Assembler ist ausschließlich in FORTH implementiert, um ihn einfach portieren zu können. Folgende Merkmale beschreiben die Leistungsfähigkeit des Assemblers.

- Umfang von nur 5,5K-Byte (kann temporär auf den Heap geladen werden)
- Alle Super8 Befehle
- Lokale Labels
- Tabellengesteuert
- 16-Bit Befehle als Makros
- Ausführliche Fehlerbehandlung
- Symbolische Adressierung der System, Mode und Special Register
- Symbolische Adressierung der Register der virtuellen FORTH Maschine

Die Mnemonic des Super8 FORTH Assemblers ist sehr eng an den Cross Assembler asmS8™ der Firma Zilog angelehnt. Eine Vergleichsliste (Anhang B.4) erleichtert die Umsetzung vorhandener Programme von der Zilog Notation in die FORTH Assembler Notation. Diese Vergleichsliste enthält den generierten Objectcode, auch für Makros.
Zur Kennzeichnung sind die Befehle mit einem "," versehen (z.B. LD,), während die Makros kein "," in der Mnemonic enthalten (z.B. LDCW oder PUSHW).

Lokale Label:

Die lokalen Labels erlauben selbstverständlich Vorwärtsreferrenzen und sind lokal innerhalb einer CODE....END-CODE oder PROC....END-PROC Definition. Vorgesehen sind die Labels 1$: ..... 10$:. Diese Labels sind immer 16-Bit Adressen und dürfen bei JP, CALL, und LDW, vervendet werden. END-CODE und END-PROC lösen die Vorwärtsreferenzen über eine verkettete Liste auf, die in der Labeltabelle $TAB beginnt.

Makros:

Die Makros PUSHW und POPW legen auf, bzw. entnehmen vom Parameterstack einen 16-Bit Wert. Mit LDCW wird der gemeinsame Programm-/Datenbereich des Super8 FORTH adressiert. Der Assemblerbefehl LDE, adressiert einen zusätzlichen separaten 64kB Datenbereich. Soll diese Möglichkeit genutzt werden, muß der Port 3 entsprechend initialisiert werden, um an Bit 5 das Signal /DM zu erhalten.

Zusätzliche Makros können sehr einfach hinzugefügt werden, da der vollständige Quellcode mitgeliefert wird.

Die virtuelle FORTH-Maschine verwendet symbolische Register AX, BX, CX, und DX. AX ist immer für den TOS reserviert. Weitere Register sind FUP als FORTH User Pointer und FSP als FORTH Stack Pointer. Für ein Software Floating Paket ist FFP reserviert um einen separaten Floating Stack realisieren zu können. Ein weiteres Register ist TX als temporärer Zwischenspeicher. TX kann nicht als Adressregister verwendet werden. Alle aufgeführten Register beziehen sich auf den jeweiligen Registerpointer und sind für jede Task lokal.

Adressierungsarten:

Werden die oben aufgeführten Registernamen verwendet, wird implizit eine Working Register Adresse angenommen. Am TOS Register wird als Beispiel die kompilierte Adressierungsart erläutert. Ob XS oder XL kompiliert wird, entscheidet der Assembler abhängig vom Wert der addrs.

r	AH	R14	Working Register
r	AL	R15	Working Register
Ir	@AH	@R14	Indirect Working Register
rr	AX	RR14	Working Register Pair
Irr	@AX	RR14	Indirect Working Register Pair
X	reg (AH	reg (R14	Indexed Adressing Mode
XS	addrs (AX14	addrs(RR14)	Indexed Short Offset
XL	addrs (AX14	addrs(RR14)	indexed Long Offset

3.2 Beispielprogramme mit dem S8-Assembler

Auf der Sourcediskette finden Sie das Beispielprogramm File:ISR.SCR für die Anwendung des Super8 FORTH Assemblers.
Das erste Beispiel zeigt einen Interrupt gesteuerten UART-Input mit Datenbuffer. Das Interrupt Service Programm filtert aus den eingelesenen Daten Control Q und Control S Character aus und bietet deshalb ein XON/XOFF-Softwareprotokoll für Terminal oder Meßgeräte. Dieses Programm ist vollständig in Assembler geschrieben und benützt einen Teil der Registerfile als Datenbuffer und die obersten 4 Byte der Registerfile für Zeiger Count und Flags.

Die CODE Definition INT_INIT initialisiert die Zeiger, Zähler und und die notwendigen Register für den Receive Character Interrupt. Die eigentliche Interrupt Service Routine KEY_ISR ist als PROCedure definiert. (INT_KEY holt einen Character aus dem Buffer und INT_KEY? legt den Zählerinhalt auf den Parameterstack.

INT_EMIT? liefert ein Flag auf dem Parameterstack, abhängig davon, ob das Senderegister frei ist und ob das Flag XON enthält. (INT_EMIT schreibt den TOS in das UART Senderegister.

Die Colondefinitionen INT_KEY und INT_EMIT verwenden die oben beschriebenen Definitionen. Multitasking ist bei Verwendung der neuen I/O-Worte möglich, da in den Warteschleifen PAUSE aufgerufen wird. Mit INPUT: und OUTPUT: werden zusätzliche I/O Vektoren erzeugt. Das Wort START_ISR startet den neuen I/O. CONNECT verbindet die Interrupt Service Routine KEY_ISR mit dem Vektor 20. INT_OUTPUT und INT_INPUT schalten den I/O auf die neuen Routinen und INT_INIT setzt die Startwerte.

Um zu zeigen, wie einfach die Programmierung einer Interrupt Service Routine in FORTH ist, wurde das oben beschriebene Problem ganz in Highlevel Definitionen gelöst. Das Definingwort ISR: erzeugt den Namen der Interrupt Service Routine und schaltet den Compiler ein. Die Worte die dem Namen folgen bilden den Programmteil. ;ISR schließt die Interrupt Service Routine ab. ISR: kompiliert auch einen Assembler-Vorspann der Interrupt Service Routine, der zur Laufzeit die gesamte virtuelle FORTH Maschine auf dem Stack sichert. ;ISR kompiliert den Nachspann der Interrupt Service Routine, die das Restaurieren der virtuellen FORTH Maschine vor dem Rückkehren in das Super8 FORTH übernimmt.
Sicherlich würde man das Problem eines Interrupt gesteuerten Datenbuffers, der in der Registerfile angelegt wurde, nicht in High Level Definitionen lösen, vielmehr soll dieses Programm demonstrieren, daß der Programmierer auch mit dem FORTH Interpreter/Compiler auf alle Resourcen des Super8 Zugriff hat.

ISR:	(name ; --)	Definition einer FORTH-Interruptroutine
;ISR	(--)	Ende einer FORTH-Interruptroutine
CONNECT	(isr vector --)	Interruptvektor auf ISR-Routine setzen

Index

Anhang A:
Glossar des Zilog Super8-FORTH

A.1 Beschreibung der Abkürzungen

Bezeichnung der Befehlsart

flag	Flag (0=ff=Falsch; sonst tf=Wahr)
b	Byte
n	Einfachgenauer, vorzeichenbehafteter Wert
+n	Einfachgenauer, positiver Wert oder 0
u	Einfachgenauer, vorzeichenloser Wert
w	Nicht definierter, einfachgenauer Wert
addr	Adresse
cfa	Codefeldadresse
pfa	Parameterfeldadresse
nfa	Namensfeldadresse
lfa	Linkfeldadresse
csa	Counted-String-Adresse (Adresse des Längenbytes)
d	Doppeltgenauer, vorzeichenbehafteter Wert
+d	Doppeltgenauer, positiver Wert oder 0
ud	Doppeltgenauer, vorzeichenloser Wert
wd	Nicht definierter, doppeltgenauer Wert

Bezeichnung der Befehlsart

C	Dieser Befehl darf nur kompiliert werden
I	Dieser Befehl wird auch in :-Definitionen ausgeführt
Con	Unveränderliche Konstante
U	USER-Variable (jeder Task hat eine Kopie davon)
V	Variable
S	Systemvariable (im Speicherbereich ab Adresse $FFB0)
D	Standardmäßig mit NOOP definierter DEFER-Befehl
DX	Mit festen Funktionen versehener DEFER-Befehl
	(Die Routine dieses DEFER-Befehls folgt unmittelbar danach)
I/O	Ein über INPUT oder OUTPUT vektorisierter Befehl
	(SIO-Routinen dieser Befehle folgen unmittelbar danach)
M	Bei Multitasking kann der Befehl Probleme bereiten

A.2 Nach Gruppen sortierte Befehlsliste

Kernroutinen des S8-FORTH

EXECUTE	(cfa --)		Ausführen des Befehls mit angegebener Codefeldadresse
EXIT	(-- ; R: sys --)		Aussprung aus einem FORTH-Befehl
BRANCH	(--)		Bedingungsloser Sprung (Zieladresse als Inline-Literal)
?BRANCH	(f --)		Sprung wenn f=0 (Zieladresse als Inline-Literal)
NOOP	(--)		Nur als Platzhalter, keine Aktion

Konstanten, Adressen und Variablen

FALSE	(-- 0)		Liefert False-Flag mit Wert 0
TRUE	(-- -1)		Liefert True-Flag mit Wert -1
BL	(-- $20)	Con	Liefert ASCII-Wert des Leerzeichens
S0	(-- addr)	U	Enthält Anfangsadresse des Datenstacks (absteigend)
R0	(-- addr)	U	Enthält Anfangsadresse des Returnstacks (absteigend)
STATE	(-- addr)	U	Enthält Flag, ob Text interpretiert (STATE=0) oder kompiliert wird
BLK	(-- addr)	U	Enthält die Nummer des aktuellen Screens (0=Eingabepuffer TIB)
>IN	(-- addr)	U	Enthält den Offset in den zu interpretierenden Puffer
SCR	(-- addr)	U	Enthält Screennummer des letzten Fehlers
R#	(-- addr)	U	Enthält den Offset des letzten Fehlers
SPAN	(-- addr)	U	Enthält Anzahl der Zeichen beim letzten EXPECT
#TIB	(-- addr)	U	Enthält Anzahl der Zeichen des Eingabepuffers
BASE	(-- addr)	U	Enthält aktuelle Zahlenbasis
HLD	(-- addr)	U	Enthält Adresse des vordersten Zeichen im Zahlenausgabestring
DPL	(-- addr)	U	Enthält Anzahl der nach dem Punkt folgenden Zeichen bei Zahleneingabe
INPUT	(-- addr)	U	Enthält Zeiger auf Tabelle mit aktuellen Eingaberoutinen
OUTPUT	(-- addr)	U	Enthält Zeiger auf Tabelle mit aktuellen Ausgaberoutinen
ERRORHANDLER	(-- addr)	U	Enthält CFA der aktuellen Fehlerbehandlungsroutine
TIB	(-- addr)	U	Ab der Adresse von TIB beginn der Eingabepuffer
CURRENT	(-- addr)	V	Enthält Parameterfeldadresse des aktuellen Vokabulars
CONTEXT	(-- addr)		Enthält Adresse des ersten Suchvokabulars (variabler Teil)
LAST	(-- addr)	V	Enthält Adresse der letzten Linkfeldadresse
?HEAD	(-- addr)	V	Enthält Anzahl (0-n) der headerlos zu kompilierenden Wörter
RAMORG	(-- $FFB2)	S	Enthält Anfangsadresse des RAM's
VOC-LINK	(-- $FFB4)	S	Enthält Parameterfeldadresse des zuletzt definierten Vokabulars
DP	(-- $FFB6)	S	Enthält Adresse des aktuellen Dictionary-Ende
VDP	(-- $FFB8)	S	Enthält Anfangsadresse des Variablenbereiches
VLEN	(-- $FFBA)	S	Enthält Länge des Variablenbereiches
HDP	(-- $FFBC)	S	Enthält Anfangsadresse des Heap
HERE	(-- addr)		Liefert Adresse des nächsten Dictionaryeintrages
WDP@	(-- addr)		Liefert Adresse für WORD (84 Bytes unterhalb PAD)
PAD	(-- addr)		Liefert Adresse eines Arbeitsspeichers von mindestens 84 Bytes
HEAP	(-- addr)		Liefert Anfangsadresse des Heaps
FIRST	(-- addr)		Liefert Adresse des Diskpufferanfangs
->VAR	(addr1 -- addr2)		Liest Offset aus addr1 und ermittelt Variablenadresse

Datenstackbefehle

SP@	(-- addr)		Aktuelle Stackadresse abfragen (Achtung: TOS im Register)
SP!	(addr --)		Neue Stackposition setzen (Achtung: TOS im Register)
DEPTH	(-- n)		Anzahl der Stackeinträge vor Ausführung des Befehls
DCLEAR	(... --)		Datenstack löschen
ROLL	(wo wn wm .. n -- wo wm .. wn)		Rotiert n-ten Stackwert (ab 0 gezählt) zum TOS
SWAP	(w1 w2 -- w2 w1)		Vertauschen der obersten zwei Stackeinträge
ROT	(w1 w2 w3 -- w2 w3 w1)		Rotieren der obersten drei Stackeinträge
-ROT	(w1 w2 w3 -- w3 w1 w2)		Rotiert obersten Wert nach unten
2SWAP	(d1 d2 -- d2 d1)		Vertauschen der obersten zwei 32Bit-Einträge
PICK	(wn .. w0 u -- wn .. w0 wn)		Kopiert n-ten Stackeintrag (ab 0 gezählt) zum TOS
DUP	(w -- w w)		Dupliziert w
OVER	(w1 w2 -- w1 w2 w1)		Kopiert den zweiten Stackeintrag zum TOS
TUCK	(w1 w2 -- w2 w1 w2)		Kopiert TOS unter den nächsten Stackeintrag
2DUP	(d -- d d)		Dupliziert 32Bit-Wert
?DUP	(n -- n n) oder (0 -- 0)		Dupliziert n nur, wenn der Wert ungleich 0 ist
DROP	(w --)		Entfernt w vom Datenstack
NIP	(w1 w2 -- w2)		Entferne zweiten Stackeintrag
2DROP	(w1 w2 --)		Zwei Werte von Datenstack entfernen

Returnstackbefehle

RP@	(-- addr)		Aktuelle Returnstackadresse abfragen (Achtung: Befehl fehlerhaft)
RP!	(addr --)	R	Neue Returnstackposition setzen
>R	(w -- ; R: -- w)	R	Wert w zum Returnstack bringen
R@	(-- w ; R: w -- w)		Kopiert obersten Returnstackwert zum Datenstack
RDROP	(-- ; R: w --)	R	Entfernt obersten Returnstackwert
R>	(-- w ; R: w --)	R	Übertragt obersten Returnstackwert zum Datenstack
PUSH	(addr -- ; R: -- w addr pop)	R	Variable bis zum nächsten EXIT merken

Arithmetik, Logik und Vergleiche

+	(n1 n2 -- n3)	Addiert n2 zu n1
-	(n1 n2 -- n3)	Subtrahiert n2 von n1
NEGATE	(n1 -- n2)	Zweierkomplement: n2 = 0-n1
ABS	(n1 -- n2)	Absolutwert (Achtung: bei n1=-32768 ist n2=32768=-32768)
1+	(n1 -- n2)	Addiert 1 zu n1
2+	(n1 -- n2)	Addiert 2 zu n1
1-	(n1 -- n2)	Subtrahiert 1 von n1
2-	(n1 -- n2)	Subtrahiert 2 von n1
AND	(mask1 mask2 -- mask3)	Logische AND-Verknüpfung
OR	(mask1 maks2 -- mask3)	Logische OR-Verknüpfung
XOR	(mask1 mask2 -- mask3)	Logische XOR-Verknüpfung
NOT	(w1 -- w2)	Einerkomplement: Alle Bits in w1 invertiert
2*	(w1 -- w2)	w1 um ein Bit nach links schieben (Bit0 wird 0)
2/	(n1 -- n2)	n1 arithmetisch um ein Bit nach rechts schieben (Bit 15 bleibt)
U2/	(u1 -- u2)	u1 logisch nach links schieben (Bit 15 wird 0)
FLIP	($xxyy -- $yyxx)	High- und Low-Byte des Wertes vertauschen

0<	(n -- f)	f wird -1, wenn n kleiner 0 ist
0=	(n -- f)	f wird -1, wenn n gleich 0 ist
0>	(n -- f)	f wird -1, wenn n größer 0 ist
<	(n1 n2 -- f)	f wird -1, wenn n1 kleiner n2 ist
=	(n1 n2 --)	f wird -1, wenn n1 gleich n2 ist
>	(n1 n2 -- f)	f wird -1, wenn n1 größer n2 ist
U<	(u1 u2 -- f)	f wird -1, wenn u1 kleiner u2 ist
MAX	(n1 n2 -- n1 | n2)	Der größere Wert bleibt auf dem Stack
MIN	(n1 n2 -- n1 | n2)	Der kleinere Wert bleibt auf dem Stack
UWITHIN	(u1 u2 u3 -- f)	f wird -1, wenn u2µu1<u3 ist (Differenz zu u2 mit U< testen)
CASE?	(u1 u2 -- u1 0 | -1)	Wenn u1=u2 ist, wird nur True-Flag -1 zurückgeliefert
D+	(d1 d2 -- d3)	Addiere d2 zu d1
D-	(d1 d2 -- d3)	Subtrahiere d2 von d1
DNEGATE	(d1 -- d2)	Subtrahiere d1 von 0
DABS	(d1 -- +d2)	d1 von 0 subtrahieren, fall d1 kleiner 0 ist
D<	(d1 d2 -- f)	f wird -1, wenn d1 kleiner d2 ist
UM*	(u1 u2 -- ud)	Multipliziert u1 mit u2 zum 32Bit-Ergebnis (vorzeichenlos)
M*	(n1 n2 -- d)	Multipliziert n1 mit n2 zum 32Bit-Ergebnis
*	(n1 n2 -- n3)	Multipliziert n1 mit n2 zum 16Bit-Ergebnis
UM/MOD	(ud u1 -- u2 u3)	Division von ud durch u1 liefert Quotient u3 und Rest u2 (vorzeichenlos)
M/MOD	(d n1 -- n2 n3)	Division von d durch n1 liefert Rest n2 und Quotient n3
M/	(d1 n1 -- n2)	Division von d durch n1 liefert Quotient n2
/MOD	(n1 n2 -- n3 n4)	Division von n1 durch n2 liefert Rest n3 und Quotient n4
/	(n1 n2 -- n3)	Division von n1 durch n2 mit Quotient n3
MOD	(n1 n2 -- n3)	Division von n1 durch n2 liefert Rest n3
*/MOD	(n1 n2 n3 -- n4 n5)	n1*n2/n3 ergibt Rest n4 und Quotient n5 (32Bit-Zwischenergebnis)
*/	(n1 n2 n3 -- n4)	n1*n2/n3 ergibt Quotient n4 (32Bit-Zwischenergebnis)

Speicherzugriffe

C@	(addr -- w)	Byte aus Speicheradresse addr lesen (Programmspeicher)
C!	(byte addr --)	Ein Byte in Adresse addr speichern (Programmspeicher)
@	(addr -- w)	Einen 16Bit-Wert ab Programmadresse addr lesen (Highbyte zuerst)
!	(w addr --)	16Bit-Wert ab Programmadresse addr speichern (Highbyte zuerst)
ON	(addr --)	16Bit-Variable ab Adresse addr auf -1 setzen
OFF	(addr --)	16Bit-Variable ab Adresse addr auf 0 setzen
+!	(n addr --)	Addieren des Wertes n zur 16Bit-Variable ab Adresse addr
GR@	(addr -- b)	Byte aus dem allgemeinen Registerfile lesen ($C0-$CF für S8-FORTH)
GR!	(b addr --)	Bytes in das allgemeine Registerfile schreiben ($C0-$CF für S8-FORTH)
SPR@	(addr -- b)	Byte aus einer der Portregister lesen (Bank1 wenn Adressbits 8-15 <> 0)
SPR!	(b addr --)	Byte in ein Portregister schreiben (Bank1 wenn Adressbits 8-15 <> 0)

CMOVE	(addr1 addr2 u --)		Kopiert u Bytes von addr1 nach addr2 (aufsteigend ab addr)
CMOVE>	(addr1 addr2 u --)		Kopiert u Bytes von addr1 nach addr2 (absteigend ab addr+u-1)
FILL	(addr u char --)		Füllt u Bytes ab addr mit Bytewert char
ERASE	(addr u --)		Füllt u Bytes ab addr mit Null
BLANK	(addr u --)		Füllt u Bytes ab addr mit ASCII-Wert 32 (Leerzeichen)
COUNT	(csa -- addr u)		Ermittelt Stringadresse und -länge aus der csa
-TRAILING	(addr1 +n1 -- addr1 +n2)		Schneidet Leerzeichen am Stringende ab
?CAP	(csa -- csa)	DX	DEFER-Befehl zur Umwandlung von Kleinschrift in Großschrift

Ein-/Ausgabebefehle

SIO_INPUT	(--)		INPUT so ändern, daß Eingaben von SIO kommen
SIO_OUTPUT	(--)		OUTPUT so ändern, daß Ausgaben zur SIO gehen
EMIT?	(-- f)	I/O, M	f=-1 wenn ein Zeichen ausgegeben werden kann
EMIT	(char --)	I/O, M	Zeichen mit ASCII-Wert char ausgeben
TYPE	(addr u --)	M	u Zeichen ab addr ausgeben
SPACE	(--)	M	Ein Leerzeichen ausgeben
SPACES	(n --)	M	Falls n positiv ist: n Leerzeichen ausgeben
CR	(--)	M	Zum Anfang der nächsten Zeile gehen (CR=$13 und LF=$0A ausgeben)
KEY?	(-- f)	I/O, M	f=-1 wenn ein Zeichen bereitsteht
KEY	(-- char)	I/O, M	Wartet auf ein Zeichen und liefert dessen ASCII-Wert char
STOP?	(-- f)	M	Wartet bei beliebigen Zeichen auf nächstes Zeichen, f=-1 bei CTRL+X
EXPECT	(addr +n)	M	Holtn String (maximal n Zeichen; Ende: RETURN-Taste; ändert SPAN)
DECIMAL	(--)		Setzt USER-Variable BASE auf 10 (demimale Zahlenbasis)
HEX	(--)		Setzt USER-Variable BASE auf 16 (hexadezimale Zahlenbasis)
<#	(--)		Initialisiert Zahlenausgabe (setzt HLD auf PAD)
HOLD	(char --)		Setzt Zeichen char vor den Zahlenstring (erniedrigt vorher HLD)
#	(ud1 -- ud2)		Teilt ud1 durch BASE und setzt Rest als Ziffer vor den Zahlenstring
#S	(ud1 -- 0.)		Ruft # solange auf, bis Quotient 0 ist (aber mindestens einmal)
SIGN	(n --)		Setzt bei n<0 das Minuszeichen vor den Zahlenstring
#>	(ud -- addr u)		Abschluß der Zahlenstring-Erzeugung liefert Adresse und Stringlänge
D.R	(d n --)	M	Rechtsbündige Ausgabe von d in einem Feld mit n Zeichen
D.	(d --)	M	Ausgabe von d mit nachfolgendem Leerzeichen
U.R	(u n --)	M	u rechtsbündig in einem Feld mit n Zeichen ausgeben
U.	(u --)	M	u mit nachfolgendem Leerzeichen ausgeben
.R	(n1 n2 --)	M	Ausgabe des Wertes n1 rechtsbündig in einem Feld mit n2 Zeichen
.	(n --)	M	n mit nachfolgendem Leerzeichen ausgeben
CONVERT	(ud1 addr1 -- ud2 addr2)		Akkumuliert String ab addr1+1 gemäß BASE zu ud1 (Fehler bei addr2)
NUMBER?	(csa -- addr ff | d 0> | n -1)	DX	Umwandlung des Zahlenstrings in eine 16- oder 32Bit-Zahl

Kontrollstrukturen

<MARK	(-- addr)	Liefert aktuelle Dictionaryadresse als Rücksprungziel
<RESOLVE	(addr --)	Setzt addr als Rücksprungadresse ins Dictionary
>MARK	(-- addr)	Reserviert 2 Bytes und legt dessen Adresse auf den Stack
>RESOLVE	(addr --)	Setzt aktuelle Dictionaryadresse als Sprungziel an addr

IF	(f --) (c: -- addr 1)	R,I	Anfang einer IF... [ELSE...] THEN -Struktur (Sprung bei f<>0)
ELSE	(--) (c: addr1 1 -- addr2 -1)	R,I	False-Teil einer IF...ELSE...THEN -Struktur (unbedingter Sprung)
THEN	(--) (c: addr |1| --)	R,I	Ende einer IF... [ELSE...] THEN -Struktur
BEGIN	(--) (c: -- 2 addr 2)	R,I	Anfang einer BEGIN...UNTIL/BEGIN...REPEAT -Schleife
WHILE	(f --) (c: addr1 2 -- addr2 -2 addr1 2)	R,I	Bei f=0 wird die BEGIN...REPEAT-Schleife verlassen
UNTIL	(f --) (c: 2...addr2 -2 addr1 2 --)	R,I	Ende einer BEGIN...UNTIL-Struktur (Rückkehr bei f=0)
REPEAT	(--) (c: 2...addr2 -2 addr1 2 --)	R,I	Ende einer BEGIN...REPEAT-Struktur (unbedingte Rückkehr)
?DO	(n1 n2 --) (c: -- addr 4)	R,I	Anfang einer ?DO...LOOP -Struktur (überspringen, wenn n1=n2)
DO	(n1 n2 --) (c: -- addr 4)	R,I	Anfang einer DO...LOOP -Struktur
LOOP	(--) (c: addr 4 --)	R,I	Ende einer DO...LOOP -Struktur (Index um 1 erhöhen, Test auf Ende)
+LOOP	(n --) (c: addr 4 --)	R,I	Ende einer DO...+LOOP -Struktur (Index+n, Test auf Ende)
LEAVE	(--) (c: --)	R	Verläßt sofort die innerste DO...LOOP -Struktur
I	(-- n)	R	Liefert aktuellen Wert der innersten DO...LOOP -Schleife
J	(-- n)	R	Liefert aktuellen Wert der nächst äußeren DO...LOOP -Schleife

Vokabularverwaltung

>BODY	(cfa -- pfa)	Ermittelt Parameterfeldadresse aus Codefeldadresse
>NAME	(cfa -- nfa)	Ermittelt Namensfeldadresse aus Codefeldadresse (0 bei Fehler)
>LINK	(cfa -- lfa)	Ermittelt Linkfeldadresse aus Codefeldadresse (0 bei Fehler)
BODY>	(pfa -- cfa)	Ermittelt Codefeldadresse aus Parameterfeldadresse
NAME>	(nfa -- cfa)	Ermittelt Codefeldadresse aus Namensfeldadresse
LINK>	(lfa -- cfa)	Ermittelt Codefeldadresse aus Linkfeldadresse
N>LINK	(nfa -- lfa)	Ermittelt Linkfeldadresse aus Namensfeldadresse
L>NAME	(lfa -- nfa)	Ermittelt Namensfeldadresse aus Linkfeldadresse
FORTH	(--)	Ersetzte das erste Suchvokabular durch FORTH
ONLYFORTH	(--)	FORTH wird zum einzigen Vokabular, daß durchsucht wird
ALSO	(--)	CONTEXT -Vokabular wird in den festen Teil übernommen (maximal 4)
DEFINITIONS	(--)	CONTEXT -Vokabular wird nach CURRENT übernommen
FIND	(csa -- csa 0 | cfa f)	CFA ermitteln (f<0:Immediate;|f|=2:restrict;verwendet ?CAP)
'	(name ; -- cfa)	Liefert Codefeldadresse des Befehls
[']	(-- w) (c: name ; --) R,I	Kompiliert Codefeldadresse als Literal in den Befehls

Informationen zum FORTH-System

.S	(--)	M	Gibt bis zu 10 Einträge ohne Stackveränderung aus (TOS zuerst)
DUMP	(addr u --)	M	Zeigt mindestens u Bytes ab Adressen addr an (Programmspeicher)
VOCS	(--)		Zeige alle vorhandenen Vokabulare an
ORDER	(--)		Zeigt alle CONTEXT - und das CURRENT -Vokabular an
WORDS	(--)		Zeigt alle Befehle des aktuellen CONTEXT -Vokabular an

Definition und Dictionaryverwaltung

ALLOT	(n --)	Reserviert n Adressen im Dictionary
C,	(w --)	Speichert das niederwertige Byte von w im Dictionary und erhöht DP
,	(w --)	Speichert w im Dictionary (Highbyte zuerst) und erhöht DP
$,	(char --)	Nachfolgenden String bis Zeilenende oder char kompilieren

HALLOT	(n --)	M	Reserviert n Adressen im Heap (verschiebt auch Variablenbereich)
H,	(w --)	M	Speichert w im Heap (Highbyte zuerst)
VALLOT	(n --)	M	Reserviert n Adressen im Variablenbereich (verändert VDP und VLEN)
V,	(w --)	M	Speichert w im Variablenbereich (Highbyte zuerst)
WORD	(string ; char -- addr)	M	Nächstes, mit char begrenztes Word nach WDP@ bringen
LITERAL	(-- w) (c: w --)	R,I	Kompiliert w als Inline-Literal in die aktuelle Definition
COMPILE	(--)	R	Kompiliert nächsten Befehl in die aktuelle Definition und erhöht IP
[COMPILE]	(--) (c: name ; --)	R,I	Compiliert die Codefeldadresse des Befehls in die aktuelle Definition
"	(-- csa) (c: ccc" ; --)	I	Liefert Stringadresse des nachfolgenden Strings (Interpretermodus: PAD)
."	(--) (c: ccc" ; --)	R,I,M	Kompiliert String und gibt ihn bei Befehlsausführung aus
.(	(ccc) ; --)	I,M	Ein mit) abgeschlossener String wird unmittelbar ausgeben
(	(ccc) ; --)	I	String bis zum abschließenden) ist Kommentar (nicht schachtelbar)
\	(--)	I	Rest der Zeile ist Kommentar
\\	(--)	I	Rest des Screens ist Kommentar
ASCII	(ccc ; -- char)		Liefert Wert des ersten Zeichens im nachfolgenden String
-HEADERS	(--)		Alle nachfolgenden Befehlsheader im Heap ablegen
|	(--)		Nur nächsten Befehlsheader im Heap ablegen
HEADERS	(--)		Alle nachfolgenden Befehlsheader wieder im Dictionary ablegen
HIDE	(--)		Letzten Befehl verstecken
REVEAL	(--)		Versteckten Befehl wieder sichtbar machen
IMMEDIATE	(--)		Markiert den zuletzt definierten Befehl als Immediate
RESTRICT	(--)		Markiert letztes Wort als Restrict
CREATE	(name ; --)	DX	Erzeugt einen neuen Befehl mit nachfolgenden Name
DOES>	(-- addr) (c: --)	R,I	Leitet den Ausführungsteile einer CREATE-Definition ein
:	(name ; -- 0) (-- addr)		Definition eines FORTH-Befehls (Wort wird versteckt)
]	(--)		Setzt Kompiler-Modus
;	(0 --) (addr --)	R,I	Abschluß einer FORTH-Definition (Wort wieder sichtbar)
[	(--)	I	Setzt Interpreter-Modus
VARIABLE	(name ; --)		Definition einer Variable (Wert im Variablenbereich)
CONSTANT	(name ; w --)		Definition einer Konstante (Wert im Dictionary)
VOCABULARY	(name ; --)		Definition eines Vokabulars
DEFER	(name ; --)		Definition eines DEFER-Wortes (CFA-Zeiger im Variablenbereich)
IS	(name ; cfa --)	I	DEFER-Befehl name wird auf die angegebene Codefeldadresse umleiten

Initialisierung und FORTH-Interpreter/Kompiler

COLD	(... --)		Start des FORTH wie beim Einschalten (Sprung zu $20)
'COLD	(--)	D	DEFER-Vektor für eigene Einschaltroutinen

ABORT	(... --)		Datenstack löschen, Kompiliermodus, 'COLD , Einschaltmeldung, QUIT
'ABORT	(--)	D	DEFER-Vektor für eigene Programme vor Aufruf von IDENT und QUIT
IDENT	(--)		Einschaltmeldung "Zilog Super8-FORTH V1.0" ausgeben
QUIT	(-- ; R: ? --)	DX	Schleife: Eingabe holen, interpretieren, "]" oder "ok" ausgeben
INTERPRET	(--)	DX	Diskpuffer oder TIB kompilieren oder interpretieren

Fehlerbehandlung

(ERROR	($xxxx | csa -1 --)		Standard-Fehlerbehandlung (Fehlernummer und "]" oder "ok" ausgeben)
ERROR	($xxxx | csa -1 --)		Aufruf der Fehlerbehandlung, wenn TOS ungleich Null
?ERROR	(f $xxxx --)		Aufruf von ERROR, wenn f ungleich 0
?STACK	(--)		Fehlerbehandlung, wenn Stack kleiner 0 oder größer 20 ist
?PAIRS	(w1 w2 --)		Fehlerbehandlung, wenn w1<>w2 ist (Error $000A)
INTERROR	(--)		Interrupt abschalten, Fehler $0015 ausgeben
NODEFER	(--)		Fehlernummer $000C ausgeben
NOTFOUND	(addr --)	DX	addr entfernen, Aufruf von ERROR mit Text " ??? "
ABORT"	(string"; f --)	R,I	Bei f<>0 wird ERROR mit Fehlerstring aufgerufen (Stack gelöscht)
ERROR"	(string"; f --)	R,I,M	Bei f<>0 wird ERROR mit nachfolgenden String aufgerufen

Fileverwaltung

COMMAND!	(n command -- error)		Befehl command mit 16Bit-Wert w zum Terminal, error erwarten
R/W	(addr blk r/w -- error)	DX	Block blk zur Adresse addr bringen oder auf Diskette schreiben (r/w=0)
BLOCK	(u -- addr)	M	Ladet Block u und liefert Anfangsadresse (sichert UPDATE'd Block)
BUFFER	(u -- addr)	M	Wie BLOCK, jedoch wird der Block nicht geladen
UPDATE	(--)		Markiert aktuellen Puffer als verändert (Bit 15 der Nummer wird gesetzt)
SAVE-BUFFERS	(--)	M	Ein mit UPDATE markierter Puffer wird gesichert
EMPTY-BUFFERS	(--)	M	Gibt Diskpuffer ohne Sicherung frei (füllt ihn auch mit Leerzeichen)
FLUSH	(--)	M	Führt SAVE-BUFFERS aus und gibt dann den Diskpuffer frei
LOAD	(u --)	M	Interpretiert/Kompiliert Block u (0 nicht erlaubt)
THRU	(u1 u2)	M	Interpretiert/Kompiliert Blöcke u1 bis einschließlich u2
-->	(--)	I	Interpretiert/Kompiliert nächsten Block
SAVESYSTEM	(--)	M	Aktueller Systemzustand als Bitimage zum Terminal schicken

A.3 Alphabetisch sortierte Befehlsliste

!	(w addr --)		16Bit-Wert ab Programmadresse addr speichern (Highbyte zuerst)
"	(-- csa) (c: ccc" ; --)	I	Liefert Stringadresse des nachfolgenden Strings (Interpretermodus: PAD)
#	(ud1 -- ud2)		Teilt ud1 durch BASE und setzt Rest als Ziffer vor den Zahlenstring
#>	(ud -- addr u)		Abschluß der Zahlenstring-Erzeugung liefert Adresse und Stringlänge
#S	(ud1 -- 0.)		Ruft # solange auf, bis Quotient 0 ist (aber mindestens einmal)
#TIB	(-- addr)	U	Enthält Anzahl der Zeichen des Eingabepuffers
$,	(char --)		Nachfolgenden String bis Zeilenende oder char kompilieren
'	(name ; -- cfa)		Liefert Codefeldadresse des Befehls
'ABORT	(--)	D	DEFER-Vektor für eigene Programme vor Aufruf von IDENT und QUIT
'COLD	(--)	D	DEFER-Vektor für eigene Einschaltroutinen
(	(ccc) ; --)	I	String bis zum abschließenden) ist Kommentar (nicht schachtelbar)
(ERROR	($xxxx | csa -1 --)		Standard-Fehlerbehandlung (Fehlernummer und "]" oder "ok" ausgeben)
*	(n1 n2 -- n3)		Multipliziert n1 mit n2 zum 16Bit-Ergebnis
*/	(n1 n2 n3 -- n4)		n1*n2/n3 ergibt Quotient n4 (32Bit-Zwischenergebnis)
*/MOD	(n1 n2 n3 -- n4 n5)		n1*n2/n3 ergibt Rest n4 und Quotient n5 (32Bit-Zwischenergebnis)
+	(n1 n2 -- n3)		Addiert n2 zu n1
+!	(n addr --)		Addieren des Wertes n zur 16Bit-Variable ab Adresse addr
+LOOP	(n --) (c: addr 4 --)	R,I	Ende einer DO...+LOOP -Struktur (Index+n, Test auf Ende)
,	(w --)		Speichert w im Dictionary (Highbyte zuerst) und erhöht DP
-	(n1 n2 -- n3)		Subtrahiert n2 von n1
-->	(--)	I	Interpretiert/Kompiliert nächsten Block
->VAR	(addr1 -- addr2)		Liest Offset aus addr1 und ermittelt Variablenadresse
-HEADERS	(--)		Alle nachfolgenden Befehlsheader im Heap ablegen
-ROT	(w1 w2 w3 -- w3 w1 w2)		Rotiert obersten Wert nach unten
-TRAILING	(addr1 +n1 -- addr1 +n2)		Schneidet Leerzeichen am Stringende ab
.	(n --)	M	n mit nachfolgendem Leerzeichen ausgeben
."	(--) (c: ccc" ; --)	R,I,M	Kompiliert String und gibt ihn bei Befehlsausführung aus
.(	(ccc) ; --)	I,M	Ein mit) abgeschlossener String wird unmittelbar ausgeben
.R	(n1 n2 --)	M	Ausgabe des Wertes n1 rechtsbündig in einem Feld mit n2 Zeichen
.S	(--)	M	Gibt bis zu 10 Einträge ohne Stackveränderung aus (TOS zuerst)
/	(n1 n2 -- n3)		Division von n1 durch n2 mit Quotient n3
/MOD	(n1 n2 -- n3 n4)		Division von n1 durch n2 liefert Rest n3 und Quotient n4
:	(name ; -- 0) (-- addr)		Definition eines FORTH-Befehls (Wort wird versteckt)
;	(0 --) (addr --)	R,I	Abschluß einer FORTH-Definition (Wort wieder sichtbar)
<	(n1 n2 -- f)		f wird -1, wenn n1 kleiner n2 ist
<#	(--)		Initialisiert Zahlenausgabe (setzt HLD auf PAD)
<MARK	(-- addr)		Liefert aktuelle Dictionaryadresse als Rücksprungziel
<RESOLVE	(addr --)		Setzt addr als Rücksprungadresse ins Dictionary
=	(n1 n2 --)		f wird -1, wenn n1 gleich n2 ist
>	(n1 n2 -- f)		f wird -1, wenn n1 größer n2 ist
>BODY	(cfa -- pfa)		Ermittelt Parameterfeldadresse aus Codefeldadresse
>IN	(-- addr)	U	Enthält den Offset in den zu interpretierenden Puffer
>LINK	(cfa -- lfa)		Ermittelt Linkfeldadresse aus Codefeldadresse (0 bei Fehler)
>MARK	(-- addr)		Reserviert 2 Bytes und legt dessen Adresse auf den Stack
>NAME	(cfa -- nfa)		Ermittelt Namensfeldadresse aus Codefeldadresse (0 bei Fehler)
>R	(w -- ; R: -- w)	R	Wert w zum Returnstack bringen
>RESOLVE	(addr --)		Setzt aktuelle Dictionaryadresse als Sprungziel an addr
?BRANCH	(f --)		Sprung wenn f=0 (Zieladresse als Inline-Literal)
?CAP	(csa -- csa)	DX	DEFER-Befehl zur Umwandlung von Kleinschrift in Großschrift
?DO	(n1 n2 --) (c: -- addr 4)	R,I	Anfang einer ?DO...LOOP -Struktur (überspringen, wenn n1=n2)
?DUP	(n -- n n) oder (0 -- 0)		Dupliziert n nur, wenn der Wert ungleich 0 ist
?ERROR	(f $xxxx --)		Aufruf von ERROR, wenn f ungleich 0
?HEAD	(-- addr)	V	Enthält Anzahl (0-n) der headerlos zu kompilierenden Wörter
?PAIRS	(w1 w2 --)		Fehlerbehandlung, wenn w1<>w2 ist (Error $000A)
?STACK	(--)		Fehlerbehandlung, wenn Stack kleiner 0 oder größer 20 ist
0<	(n -- f)		f wird -1, wenn n kleiner 0 ist
0=	(n -- f)		f wird -1, wenn n gleich 0 ist
0>	(n -- f)		f wird -1, wenn n größer 0 ist
1+	(n1 -- n2)		Addiert 1 zu n1
1-	(n1 -- n2)		Subtrahiert 1 von n1
2*	(w1 -- w2)		w1 um ein Bit nach links schieben (Bit0 wird 0)
2+	(n1 -- n2)		Addiert 2 zu n1

2-	(n1 -- n2)		Subtrahiert 2 von n1
2/	(n1 -- n2)		n1 arithmetisch um ein Bit nach rechts schieben (Bit 15 bleibt)
2DROP	(w1 w2 --)		Zwei Werte von Datenstack entfernen
2DUP	(d -- d d)		Dupliziert 32Bit-Wert
2SWAP	(d1 d2 -- d2 d1)		Vertauschen der obersten zwei 32Bit-Einträge
@	(addr -- w)		Einen 16Bit-Wert ab Programmadresse addr lesen (Highbyte zuerst)
ABORT	(... --)		Datenstack löschen, Kompiliermodus, 'COLD , Einschaltmeldung, QUIT
ABORT"	(string"; f --)	R,I	Bei f<>0 wird ERROR mit Fehlerstring aufgerufen (Stack gelöscht)
ABS	(n1 -- n2)		Absolutwert (Achtung: bei n1=-32768 ist n2=32768=-32768)
ALLOT	(n --)		Reserviert n Adressen im Dictionary
ALSO	(--)		CONTEXT -Vokabular wird in den festen Teil übernommen (maximal 4)
AND	(mask1 mask2 -- mask3)		Logische AND-Verknüpfung
ASCII	(ccc ; -- char)		Liefert Wert des ersten Zeichens im nachfolgenden String
BASE	(-- addr)	U	Enthält aktuelle Zahlenbasis
BEGIN	(--) (c: -- 2 addr 2)	R,I	Anfang einer BEGIN...UNTIL/BEGIN...REPEAT -Schleife
BL	(-- $20)	Con	Liefert ASCII-Wert des Leerzeichens
BLANK	(addr u --)		Füllt u Bytes ab addr mit ASCII-Wert 32 (Leerzeichen)
BLK	(-- addr)	U	Enthält die Nummer des aktuellen Screens (0=Eingabepuffer TIB)
BLOCK	(u -- addr)	M	Ladet Block u und liefert Anfangsadresse (sichert UPDATE'd Block)
BODY>	(pfa -- cfa)		Ermittelt Codefeldadresse aus Parameterfeldadresse
BRANCH	(--)		Bedingungsloser Sprung (Zieladresse als Inline-Literal)
BUFFER	(u -- addr)	M	Wie BLOCK, jedoch wird der Block nicht geladen
C!	(byte addr --)		Ein Byte in Adresse addr speichern (Programmspeicher)
C,	(w --)		Speichert das niederwertige Byte von w im Dictionary und erhöht DP
C@	(addr -- w)		Byte aus Speicheradresse addr lesen (Programmspeicher)
CASE?	(u1 u2 -- u1 0 | -1)		Wenn u1=u2 ist, wird nur True-Flag -1 zurückgeliefert
CMOVE	(addr1 addr2 u --)		Kopiert u Bytes von addr1 nach addr2 (aufsteigend ab addr)
CMOVE>	(addr1 addr2 u --)		Kopiert u Bytes von addr1 nach addr2 (absteigend ab addr+u-1)
COLD	(... --)		Start des FORTH wie beim Einschalten (Sprung zu $20)
COMMAND!	(n command -- error)		Befehl command mit 16Bit-Wert w zum Terminal, error erwarten
COMPILE	(--)	R	Kompiliert nächsten Befehl in die aktuelle Definition und erhöht IP
CONSTANT	(name ; w --)		Definition einer Konstante (Wert im Dictionary)
CONTEXT	(-- addr)		Enthält Adresse des ersten Suchvokabulars (variabler Teil)
CONVERT	(ud1 addr1 -- ud2 addr2)		Akkumuliert String ab addr1+1 gemäß BASE zu ud1 (Fehler bei addr2)
COUNT	(csa -- addr u)		Ermittelt Stringadresse und -länge aus der csa
CR	(--)	M	Zum Anfang der nächsten Zeile gehen (CR=$13 und LF=$0A ausgeben)
CREATE	(name ; --)	DX	Erzeugt einen neuen Befehl mit nachfolgenden Name
CURRENT	(-- addr)	V	Enthält Parameterfeldadresse des aktuellen Vokabulars
D+	(d1 d2 -- d3)		Addiere d2 zu d1
D-	(d1 d2 -- d3)		Subtrahiere d2 von d1
D.	(d --)	M	Ausgabe von d mit nachfolgendem Leerzeichen
D.R	(d n --)	M	Rechtsbündige Ausgabe von d in einem Feld mit n Zeichen
D<	(d1 d2 -- f)		f wird -1, wenn d1 kleiner d2 ist
DABS	(d1 -- +d2)		d1 von 0 subtrahieren, fall d1 kleiner 0 ist
DCLEAR	(... --)		Datenstack löschen
DECIMAL	(--)		Setzt USER-Variable BASE auf 10 (demimale Zahlenbasis)
DEFER	(name ; --)		Definition eines DEFER-Wortes (CFA-Zeiger im Variablenbereich)
DEFINITIONS	(--)		CONTEXT -Vokabular wird nach CURRENT übernommen
DEPTH	(-- n)		Anzahl der Stackeinträge vor Ausführung des Befehls
DNEGATE	(d1 -- d2)		Subtrahiere d1 von 0
DO	(n1 n2 --) (c: -- addr 4)	R,I	Anfang einer DO...LOOP -Struktur
DOES>	(-- addr) (c: --)	R,I	Leitet den Ausführungsteile einer CREATE-Definition ein
DP	(-- $FFB6)	S	Enthält Adresse des aktuellen Dictionary-Ende
DPL	(-- addr)	U	Enthält Anzahl der nach dem Punkt folgenden Zeichen bei Zahleneingabe
DROP	(w --)		Entfernt w vom Datenstack
DUMP	(addr u --)	M	Zeigt mindestens u Bytes ab Adressen addr an (Programmspeicher)
DUP	(w -- w w)		Dupliziert w
ELSE	(--) (c: addr1 1 -- addr2 -1)	R,I	False-Teil einer IF...ELSE...THEN -Struktur (unbedingter Sprung)
EMIT	(char --)	I/O, M	Zeichen mit ASCII-Wert char ausgeben
EMIT?	(-- f)	I/O, M	f=-1 wenn ein Zeichen ausgegeben werden kann
EMPTY-BUFFERS	(--)	M	Gibt Diskpuffer ohne Sicherung frei (füllt ihn auch mit Leerzeichen)
ERASE	(addr u --)		Füllt u Bytes ab addr mit Null
ERROR	($xxxx | csa -1 --)		Aufruf der Fehlerbehandlung, wenn TOS ungleich Null
ERROR"	(string"; f --)	R,I,M	Bei f<>0 wird ERROR mit nachfolgenden String aufgerufen

ERRORHANDLER	(-- addr)	U	Enthält CFA der aktuellen Fehlerbehandlungsroutine
EXECUTE	(cfa --)		Ausführen des Befehls mit angegebener Codefeldadresse
EXIT	(-- ; R: sys --)		Aussprung aus einem FORTH-Befehl
EXPECT	(addr +n)	M	Holtn String (maximal n Zeichen; Ende: RETURN-Taste; ändert SPAN)
FALSE	(-- 0)		Liefert False-Flag mit Wert 0
FILL	(addr u char --)		Füllt u Bytes ab addr mit Bytewert char
FIND	(csa -- csa 0 | cfa f)		CFA ermitteln (f<0:Immediate;|f|=2:restrict;verwendet ?CAP)
FIRST	(-- addr)		Liefert Adresse des Diskpufferanfangs
FLIP	($xxyy -- $yyxx)		High- und Low-Byte des Wertes vertauschen
FLUSH	(--)	M	Führt SAVE-BUFFERS aus und gibt dann den Diskpuffer frei
FORTH	(--)		Ersetzte das erste Suchvokabular durch FORTH
GR!	(b addr --)		Bytes in das allgemeine Registerfile schreiben ($C0-$CF für S8-FORTH)
GR@	(addr -- b)		Byte aus dem allgemeinen Registerfile lesen ($C0-$CF für S8-FORTH)
H,	(w --)	M	Speichert w im Heap (Highbyte zuerst)
HALLOT	(n --)	M	Reserviert n Adressen im Heap (verschiebt auch Variablenbereich)
HDP	(-- $FFBC)	S	Enthält Anfangsadresse des Heap
HEADERS	(--)		Alle nachfolgenden Befehlsheader wieder im Dictionary ablegen
HEAP	(-- addr)		Liefert Anfangsadresse des Heaps
HERE	(-- addr)		Liefert Adresse des nächsten Dictionaryeintrages
HEX	(--)		Setzt USER-Variable BASE auf 16 (hexadezimale Zahlenbasis)
HIDE	(--)		Letzten Befehl verstecken
HLD	(-- addr)	U	Enthält Adresse des vordersten Zeichen im Zahlenausgabestring
HOLD	(char --)		Setzt Zeichen char vor den Zahlenstring (erniedrigt vorher HLD)
I	(-- n)	R	Liefert aktuellen Wert der innersten DO...LOOP -Schleife
IDENT	(--)		Einschaltmeldung "Zilog Super8-FORTH V1.0" ausgeben
IF	(f --) (c: -- addr 1)	R,I	Anfang einer IF... [ELSE...] THEN -Struktur (Sprung bei f<>0)
IMMEDIATE	(--)		Markiert den zuletzt definierten Befehl als Immediate
INPUT	(-- addr)	U	Enthält Zeiger auf Tabelle mit aktuellen Eingaberoutinen
INTERPRET	(--)	DX	Diskpuffer oder TIB kompilieren oder interpretieren
INTERROR	(--)		Interrupt abschalten, Fehler $0015 ausgeben
IS	(name ; cfa --)	I	DEFER-Befehl name wird auf die angegebene Codefeldadresse umleiten
J	(-- n)	R	Liefert aktuellen Wert der nächst äußeren DO...LOOP -Schleife
KEY	(-- char)	I/O, M	Wartet auf ein Zeichen und liefert dessen ASCII-Wert char
KEY?	(-- f)	I/O, M	f=-1 wenn ein Zeichen bereitsteht
L>NAME	(lfa -- nfa)		Ermittelt Namensfeldadresse aus Linkfeldadresse
LAST	(-- addr)	V	Enthält Adresse der letzten Linkfeldadresse
LEAVE	(--) (c: --)	R	Verläßt sofort die innerste DO...LOOP -Struktur
LINK>	(lfa -- cfa)		Ermittelt Codefeldadresse aus Linkfeldadresse
LITERAL	(-- w) (c: w --)	R,I	Kompiliert w als Inline-Literal in die aktuelle Definition
LOAD	(u --)	M	Interpretiert/Kompiliert Block u (0 nicht erlaubt)
LOOP	(--) (c: addr 4 --)	R,I	Ende einer DO...LOOP -Struktur (Index um 1 erhöhen, Test auf Ende)
M*	(n1 n2 -- d)		Multipliziert n1 mit n2 zum 32Bit-Ergebnis
M/	(d1 n1 -- n2)		Division von d durch n1 liefert Quotient n2
M/MOD	(d n1 -- n2 n3)		Division von d durch n1 liefert Rest n2 und Quotient n3
MAX	(n1 n2 -- n1 | n2)		Der größere Wert bleibt auf dem Stack
MIN	(n1 n2 -- n1 | n2)		Der kleinere Wert bleibt auf dem Stack
MOD	(n1 n2 -- n3)		Division von n1 durch n2 liefert Rest n3
N>LINK	(nfa -- lfa)		Ermittelt Linkfeldadresse aus Namensfeldadresse
NAME>	(nfa -- cfa)		Ermittelt Codefeldadresse aus Namensfeldadresse
NEGATE	(n1 -- n2)		Zweierkomplement: n2 = 0-n1
NIP	(w1 w2 -- w2)		Entferne zweiten Stackeintrag
NODEFER	(--)		Fehlernummer $000C ausgeben
NOOP	(--)		Nur als Platzhalter, keine Aktion
NOT	(w1 -- w2)		Einerkomplement: Alle Bits in w1 invertiert
NOTFOUND	(addr --)	DX	addr entfernen, Aufruf von ERROR mit Text " ??? "
NUMBER?	(csa -- addr ff | d 0> | n -1)	DX	Umwandlung des Zahlenstrings in eine 16- oder 32Bit-Zahl
OFF	(addr --)		16Bit-Variable ab Adresse addr auf 0 setzen
ON	(addr --)		16Bit-Variable ab Adresse addr auf -1 setzen
ONLYFORTH	(--)		FORTH wird zum einzigen Vokabular, daß durchsucht wird
OR	(mask1 maks2 -- mask3)		Logische OR-Verknüpfung
ORDER	(--)		Zeigt alle CONTEXT - und das CURRENT -Vokabular an
OUTPUT	(-- addr)	U	Enthält Zeiger auf Tabelle mit aktuellen Ausgaberoutinen
OVER	(w1 w2 -- w1 w2 w1)		Kopiert den zweiten Stackeintrag zum TOS
PAD	(-- addr)		Liefert Adresse eines Arbeitsspeichers von mindestens 84 Bytes

PICK	(wn ... w0 u -- wn ... w0 wn)		Kopiert n-ten Stackeintrag (ab 0 gezählt) zum TOS
PUSH	(addr -- ; R: -- w addr pop)	R	Variable bis zum nächsten EXIT merken
QUIT	(-- ; R: ? --)	DX	Schleife: Eingabe holen, interpretieren, "]" oder "ok" ausgeben
R#	(-- addr)	U	Enthält den Offset des letzten Fehlers
R/W	(addr blk r/w -- error)	DX	Block blk zur Adresse addr bringen oder auf Diskette schreiben (r/w=0)
R>	(-- w ; R: w --)	R	Übertragt obersten Returnstackwert zum Datenstack
R@	(-- w ; R: w -- w)		Kopiert obersten Returnstackwert zum Datenstack
R0	(-- addr)	U	Enthält Anfangsadresse des Returnstacks (absteigend)
RAMORG	(-- $FFB2)	S	Enthält Anfangsadresse des RAM's
RDROP	(-- ; R: w --)	R	Entfernt obersten Returnstackwert
REPEAT	(--) (c: 2...addr2 -2 addr1 2 --)	R,I	Ende einer BEGIN...REPEAT-Struktur (unbedingte Rückkehr)
RESTRICT	(--)		Markiert letztes Wort als Restrict
REVEAL	(--)		Versteckten Befehl wieder sichtbar machen
ROLL	(wo wn wm .. n -- wo wm .. wn)		Rotiert n-ten Stackwert (ab 0 gezählt) zum TOS
ROT	(w1 w2 w3 -- w2 w3 w1)		Rotieren der obersten drei Stackeinträge
RP!	(addr --)	R	Neue Returnstackposition setzen
RP@	(-- addr)		Aktuelle Returnstackadresse abfragen (Achtung: Befehl fehlerhaft)
S0	(-- addr)	U	Enthält Anfangsadresse des Datenstacks (absteigend)
SAVE-BUFFERS	(--)	M	Ein mit UPDATE markierter Puffer wird gesichert
SAVESYSTEM	(--)	M	Aktueller Systemzustand als Bitimage zum Terminal schicken
SCR	(-- addr)	U	Enthält Screennummer des letzten Fehlers
SIGN	(n --)		Setzt bei n<0 das Minuszeichen vor den Zahlenstring
SIO_INPUT	(--)		INPUT so ändern, daß Eingaben von SIO kommen
SIO_OUTPUT	(--)		OUTPUT so ändern, daß Ausgaben zur SIO gehen
SP!	(addr --)		Neue Stackposition setzen (Achtung: TOS im Register)
SP@	(-- addr)		Aktuelle Stackadresse abfragen (Achtung: TOS im Register)
SPACE	(--)	M	Ein Leerzeichen ausgeben
SPACES	(n --)	M	Falls n positiv ist: n Leerzeichen ausgeben
SPAN	(-- addr)	U	Enthält Anzahl der Zeichen beim letzten EXPECT
SPR!	(b addr --)		Byte in ein Portregister schreiben (Bank1 wenn Adressbits 8-15 <> 0)
SPR@	(addr -- b)		Byte aus einer der Portregister lesen (Bank1 wenn Adressbits 8-15 <> 0)
STATE	(-- addr)	U	Enthält Flag, ob Text interpretiert (STATE=0) oder kompiliert wird
STOP?	(-- f)	M	Wartet bei beliebigen Zeichen auf nächstes Zeichen, f=-1 bei CTRL+X
SWAP	(w1 w2 -- w2 w1)		Vertauschen der obersten zwei Stackeinträge
THEN	(--) (c: addr |1| --)	R,I	Ende einer IF... [ELSE...] THEN -Struktur
THRU	(u1 u2)	M	Interpretiert/Kompiliert Blöcke u1 bis einschließlich u2
TIB	(-- addr)	U	Ab der Adresse von TIB beginn der Eingabepuffer
TRUE	(-- -1)		Liefert True-Flag mit Wert -1
TUCK	(w1 w2 -- w2 w1 w2)		Kopiert TOS unter den nächsten Stackeintrag
TYPE	(addr u --)	M	u Zeichen ab addr ausgeben
U.	(u --)	M	u mit nachfolgendem Leerzeichen ausgeben
U.R	(u n --)	M	u rechtsbündig in einem Feld mit n Zeichen ausgeben
U<	(u1 u2 -- f)		f wird -1, wenn u1 kleiner u2 ist
U2/	(u1 -- u2)		u1 logisch nach links schieben (Bit 15 wird 0)
UM*	(u1 u2 -- ud)		Multipliziert u1 mit u2 zum 32Bit-Ergebnis (vorzeichenlos)
UM/MOD	(ud u1 -- u2 u3)		Division von ud durch u1 liefert Quotient u3 und Rest u2 (vorzeichenlos)
UNTIL	(f --) (c: 2...addr2 -2 addr1 2 --)	R,I	Ende einer BEGIN...UNTIL-Struktur (Rückkehr bei f=0)
UPDATE	(--)		Markiert aktuellen Puffer als verändert (Bit 15 der Nummer wird gesetzt)
UWITHIN	(u1 u2 u3 -- f)		f wird -1, wenn u2µu1<u3 ist (Differenz zu u2 mit U< testen)
V,	(w --)	M	Speichert w im Variablenbereich (Highbyte zuerst)
VALLOT	(n --)	M	Reserviert n Adressen im Variablenbereich (verändert VDP und VLEN)
VARIABLE	(name ; --)		Definition einer Variable (Wert im Variablenbereich)
VDP	(-- $FFB8)	S	Enthält Anfangsadresse des Variablenbereiches
VLEN	(-- $FFBA)	S	Enthält Länge des Variablenbereiches
VOC-LINK	(-- $FFB4)	S	Enthält Parameterfeldadresse des zuletzt definierten Vokabulars
VOCABULARY	(name ; --)		Definition eines Vokabulars
VOCS	(--)		Zeige alle vorhandenen Vokabulare an
WDP@	(-- addr)		Liefert Adresse für WORD (84 Bytes unterhalb PAD)
WHILE	(f --) (c: addr1 2 -- addr2 -2 addr1 2)	R,I	Bei f=0 wird die BEGIN...REPEAT-Schleife verlassen
WORD	(string ; char -- addr)	M	Nächstes, mit char begrenztes Word nach WDP@ bringen
WORDS	(--)		Zeigt alle Befehle des aktuellen CONTEXT -Vokabular an
XOR	(mask1 mask2 -- mask3)		Logische XOR-Verknüpfung
[	(--)	I	Setzt Interpreter-Modus
[']	(-- w) (c: name ; --)	R,I	Kompiliert Codefeldadresse als Literal in den Befehls

[COMPILE]	(--) (c: name ; --)	R,I	Compiliert die Codefeldadresse des Befehls in die aktuelle Definition
\	(--)	I	Rest der Zeile ist Kommentar
\\	(--)	I	Rest des Screens ist Kommentar
]	(--)		Setzt Kompiler-Modus
|	(--)		Nur nächsten Befehlsheader im Heap ablegen

A.4 Neue Befehle in ZUSATZ.SCR

2VARIABLE	(name ; --)		Definition einer 32Bit-Variable (Wert im Variablenbereich)
2CONSTANT	(name ; d --)		Definition einer 32Bit-Konstante (Wert im Dictionary)
OUTVECTOR:	(name ; offset --)		Definition eines neuen Ausgabebefehls
OUTPUT:	(name ; --)		Definition eines neuen Ausgabevektors
INVECTOR:	(name ; offset --)		Definition eines neuen Eingabebefehls
INPUT:	(name ; --)		Definition eines neuen Eingabevektors
'REMOVE	(dp heap -- dp heap)		Mit NOOP besetzter DEFER-Befehl wird von REMOVE aufgerufen
REMOVE	(dp heap --)		Befehle zwischen dp und heap werden entfernt und DP/HDP gesetzt
(FORGET	(dp --)		Alle nach dp folgenden Befehle werden entfernt
FORGET	(name ; --)		Einschließlich Name werden alle folgenden Befehle entfernt
HCLEAR	(--)		Der Heap wird gelöscht (verwendet REMOVE)
EMPTY	(--)		Zustand vom letzten SAVE oder SAVESYSTEM wieder herstellen
TASKORG	(-- $FFBE)	S	Enthält Anfangsadresse des Taskbereiches
LASTTASK	(-- $FFC0)	S	Enthält Offsetadresse des zuletzt definierten Tasks
TASK#	(-- $FFC2)	S	Enthält Registeradresse des zuletzt definierten Tasks
DISKORG	(-- $FFC4)	S	Enthält Anfangsadresse des Diskbereiches
SLEN	(-- addr)	U	Enthält Gesamtlänge des Datenstacks
RLEN	(-- addr)	U	Enthält Gesamtlänge des Returnstacks
UMAX	(u1 u2 -- u1 | u2)		Der größere vorzeichenlose Werte bleibt auf dem Stack
HEAP?	(addr -- f)		f wird -1, wenn addr im Heap liegt
2OVER	(d1 d2 -- d1 d2 d1)		Kopiert zweiten 32Bit-Eintrag
2ROT	(d1 d2 d3 -- d2 d3 d1)		Rotiert oberste drei 32Bit-Einträge
D2*	(dw1 -- dw2)		dw1 um ein Bit nach links schieben (Bit0 wird 0)
D2/	(d1 -- d2)		d1 arithmetisch um ein Bit nach rechts schieben (Bit 31 bleibt)
UD2/	(ud1 -- ud2)		ud1 logisch um ein Bit nach rechts schieben (Bit 31 wird 0)
D0<	(d -- f)		f wird -1, wenn d kleiner 0 ist
D0=	(d -- f)		f wird -1, wenn d gleich 0 ist
D0>	(d -- f)		f wird -1, wenn d größer 0 ist
D=	(d1 d2 --)		f wird -1, wenn d1 gleich d2 ist
D>	(d1 d2 -- f)		f wird -1, wenn d1 größer d2 ist
DMAX	(d1 d2 -- d1 | d2)		Der größere Wert bleibt auf dem Stack
DMIN	(d1 d2 -- d1 | d2)		Der kleinere Wert bleibt auf dem Stack
EC@	(addr -- w)		Byte aus Datenspeicher lesen
EC!	(w addr --)		Byte in den Datenspeicher schreiben
E@	(addr -- w)		Einen 16Bit-Wert ab Datenadresse addr lesen (Highbyte zuerst)
E!	(w addr --)		Einen 16Bit-Wert ab Datenadresse addr schreiben (Highbyte zuerst)
SKIP	(addr1 len1 char -- addr2 len2)		Zeichen char übergehen
SCAN	(addr1 len1 char -- addr2 len2)		Nach Zeichen char suchen
PAUSE	(--)		Hauptbefehl zum Umschalten auf den nächsten Task
MULTITASK	(--)		PAUSE aktivieren
SINGLETASK	(--)		PAUSE deaktivieren
ERRORTEXT	(--)		Fehlerbehandlung mit Textausgabe (in ERRORHANDLER eingebunden)
RP@	(-- addr)		Aktuelle Returnstackadresse holen (Korrigierte Version)
SAVESYSTEM	(--)		Bitimage speichern (Korrigierte Version)

Anhang B: Tabellenteil

B.1 Befehle des Terminalprogrammes

F1 Anzeige der Hilfsinformation für die Tastenbelegung
ALT+H "

ALT+F Ein neuer Name für das aktuelle Arbeitsfile eingeben.
(Vorher mit FLUSH Diskpuffer zurückschreiben)

ALT+E Aufruf des integrierten Editors.
(Vorher mit FLUSH Diskpuffer zurückschreiben)

ALT+P Ein-/Ausschalten des Druckers.

ALT+L Ein-/Ausschalten des Logfiles
(Beim Einschalten wird Filename abgefragt)

ALT+I Vorgabe des Filename für Bitimage bei SAVESYSTEM .

ALT+D Das Directory des aktuellen Verzeichnis anzeigen.

ALT+C Aufruf der COMMAND-Oberfläche.
(Rückkehr mit Befehl EXIT , SIO und Arbeitsfile nicht ändern)

ALT+X Terminalprogramm verlassen.
(Schließt aktuelles Arbeits- und Logfile)

B.2 Tastenbelegung des Editors

Cursorsteuerung:

^E oder Cursor_hoch	Eine Zeile höher
^X oder Cursor_tief	Eine Zeile tiefer
^S oder Cursor_links	Ein Zeichen links
^D oder Cursor_rechts	Ein Zeichen rechts
TAB	Zur nächsten 4er-Teilung
Shift+TAB	Zur vorherigen 4er-Teilung
^Q B	Zum Zeilenanfang
^Q K	Zum Zeilenende-1
^Q E	Zum Screenanfang
^Q X	Zum Screenende-1
POS1	Zum ersten Zeichen der Zeile
ENDE	Hinter das letzte Zeichen der Zeile
^POS1	Zum ersten Zeichen des Screens
^ENDE	Hinter das letzte Zeichen des Screens
Return	Zum nächsten Zeilenanfang
^R oder Bild_hoch	Zum vorhergehenden Screen
^C oder Bild_tief	Zum nächsten Screen
^K R oder ^Bild_hoch	Zum ersten Screen
^K C oder ^Bild_tief	Zum letzten Screen
^Q G	Eingabe der gewünschten Screennummer
^A	Zum zweiten File / Cursorposition umschalten

Zwischenspeicherung von Zeichen und Zeilen:

F1	Zeichen speichern und löschen
F2	Zeichen speichern, Cursor rechts
F3	Zeichen einfügen
F5	Zeile speichern und löschen
F6	Zeile speichern, Cursor in die nächste Zeile
F7	Zeile einfügen

Steuerung der Zeicheneingabe und des Löschens:

^V	Insert-Modus umschalten
Einfg	Ein Leerzeichen einfügen
^G oder Entf	Zeichen unter dem Cursor löschen
^H oder Backspace	Zeichen vor dem Cursor löschen
F8	Rest der Zeile löschen
^Y	Aktuelle Zeile entfernen
^N	Leerzeile einfügen
^K N	Leerscreen einfügen
^K Y	Aktuellen Screen entfernen
^Backspace	Nächste Zeile in den Rest dieser Zeile übernehmen
^Return	Rest dieser Zeile in die nächste Zeile

Suchen und Ersetzen:

^Q F	String suchen (u=Option für Rückwärts-Suche)
^Q A	String suchen und ersetzen (mehrere Optionen)

Speicherung:

F10	Änderungen in diesem Screen rückgängig machen
F9	Eingabe der ID-Kennung (nicht bei EDITOR.COM)
ESC	File speichern, Editor verlassen

B.3 Fehlerliste

	$0000	Kein Fehler (ERROR springt zurück)	
	$0001	Stackunterlauf	
(	$0002	Stacküberlauf	)
(	$0003	Returnstackunterlauf	)
(	$0004	Returnstacküberlauf	)
	$0005	Zu wenig Parameter (von ?DEPTH)	
	$0006	Name erwartet	
	$0007	COMMAND!-Fehler	
	$0008	Dictionary voll	
	$0009	Falsche Blocknummer	
	$000A	Befehl unstrukturiert	
(	$000B	Adresse zu weit entfernt	)
	$000C	DEFER nicht gesetzt (von NOTFOUND)	
(	$000D	Unbekannter Opcode	)
	$000E	Befehl darf nur kompiliert werden	
(	$000F	Kein Kurzliteral erlaubt	)
	$0010	Befehl im geschützten Bereich	
	$0011	Befehl kann nicht vergessen werden	
(	$0012	DOES> ist nicht erlaubt	)
	$0013	Befehl ist kein DEFER-Wort	
	$0014	Befehl hat kein Codefeld	
	$0015	Nicht definierter Interupt aufgetreten	
	$0016	Zu viele CONTEXT-Vokabulare	
	$FFFF	Stringadresse des Fehler liegt auf dem Stack	

B.4 Assembler-Vergleichsliste

Object Code	Zilog asmS8 Mnemonic		FORTH asmS8 Mnemonic	
1235	adc	r3,r5	R 3 , R 5	ADC,
1335	adc	r3,@r5	R 3 , @R 5	ADC,
1440c3	adc	r3,64	R 3 , 64	ADC,
14c520	adc	32,r5	32 , R 5	ADC,
144020	adc	32,64	32 , 64	ADC,
1540c3	adc	r3,@64	R 3 , @ 64	ADC,
15c520	adc	32,@r5	32 , @R 5	ADC,
154020	adc	32,@64	32 , @ 64	ADC,
16c340	adc	r3,#64	R 3 , # 64	ADC,
162040	adc	32,#64	32 , # 64	ADC,
0235	add	r3,r5	R 3 , R 5	ADD,
0335	add	r3,@r5	R 3 , @R 5	ADD,
0440c3	add	r3,64	R 3 , 64	ADD,
04c520	add	32,r5	32 , R 5	ADD,
044020	add	32,64	32 , 64	ADD,
0540c3	add	r3,@64	R 3 , @ 64	ADD,
05c520	add	32,@r5	32 , @R 5	ADD,
054020	add	32,@64	32 , @ 64	ADD,
06c340	add	r3,#64	R 3 , # 64	ADD,
062040	add	32,#64	32 , # 64	ADD,
5235	and	r3,r5	R 3 , R 5	AND,
5335	and	r3,@r5	R 3 , @R 5	AND,
5440c3	and	r3,64	R 3 , 64	AND,
54c520	and	32,r5	32 , R 5	AND,
544020	and	32,64	32 , 64	AND,
5540c3	and	r3,@64	R 3 , @ 64	AND,
55c520	and	32,@r5	32 , @R 5	AND,
554020	and	32,@64	32 , @ 64	AND,
56c340	and	r3,#64	R 3 , # 64	AND,
562040	and	32,#64	32 , # 64	AND,
673ec5	band	r3,r5,#7	R 3 , R 5 # 7	BAND,
673e40	band	r3,64,#7	R 3 , 64 # 7	BAND,
675fc3	band	r3,#7,r5	R 3 # 7 , R 5	BAND,
675f20	band	32,#7,r5	32 # 7 , R 5	BAND,
173ec5	bcp	r3,r5,#7	R 3 , R 5 # 7	BCP,
173e40	bcp	r3,64,#7	R 3 , 64 # 7	BCP,
573e	bitc	r3,#7	R 3 , # 7	BITC,
773e	bitr	r3,#7	R 3 , # 7	BITR,
773f	bits	r3,#7	R 3 , # 7	BITS,
073ec5	bor	r3,r5,#7	R 3 , R 5 # 7	BOR,
073e40	bor	r3,64,#7	R 3 , 64 # 7	BOR,
075fc3	bor	r3,#7,r5	R 3 # 7 , R 5	BOR,
075f20	bor	32,#7,r5	32 # 7 , R 5	BOR,
Object Code	Zilog asmS8 Mnemonic		FORTH asmS8 Mnemonic	
375efd	btjrf	$,r5,#7	$, R 5 # 7	BTJRF,
375ffd	btjrt	$,r5,#7	$, R 5 # 7	BTJRT,
273ec5	bxor	r3,r5,#7	R 3 , R 5 # 7	BXOR,
273e40	bxor	r3,64,#7	R 3 , 64 # 7	BXOR,
275fc3	bxor	r3,#7,r5	R 3 # 7 , R 5	BXOR,
275f20	bxor	32,#7,r5	32 # 7 , R 5	BXOR,

```
d420          call   #32 ____________________________________# 32   CALL,
f4c2          call   @rr2 ___________________________________@RR 2   CALL,
f420          call   @32 ____________________________________@ 32   CALL,
f60040        call   64 _____________________________________64   CALL,

ef             ccf   ________________________________________      CCF,

b0c3           clr   r3 ______________________________________R 3   CLR,
b020           clr   32 _____________________________________32   CLR,
b1c3           clr   @r3 _____________________________________@R 3   CLR,
b120           clr   @32 _____________________________________@ 32   CLR,

60c3           com   r3 ______________________________________R 3   COM,
6020           com   32 _____________________________________32   COM,
61c3           com   @r3 _____________________________________@R 3   COM,
6120           com   @32 _____________________________________@ 32   COM,

a235            cp   r3,r5 ___________________________________R 3 , R 5   CP,
a335            cp   r3,@r5 __________________________________R 3 , @R 5   CP,
a440c3          cp   r3,64 ___________________________________R 3 , 64   CP,
a4c520          cp   32,r5 ___________________________________32 , R 5   CP,
a44020          cp   32,64 ___________________________________32 , 64   CP,
a540c3          cp   r3,@64 __________________________________R 3 , @ 64   CP,
a5c520          cp   32,@r5 __________________________________32 , @R 5   CP,
a54020          cp   32,@64 __________________________________32 , @ 64   CP,
a6c340          cp   r3,#64 __________________________________R 3 , # 64   CP,
a62040          cp   32,#64 __________________________________32 , # 64   CP,

d253fd       cpijne  r3,@r5,$ _____________________________R 3 , @R 5 $   CPIJNE,

c253fd        cpije  r3,@r5,$ _____________________________R 3 , @R 5 $   CPIJE,

40c3            da   r3 ______________________________________R 3   DA,
4020            da   32 _____________________________________32   DA,
41c3            da   @r3 _____________________________________@R 3   DA,
4120            da   @32 _____________________________________@ 32   DA,

00c3           dec   r3 ______________________________________R 3   DEC,
0020           dec   32 _____________________________________32   DEC,
01c3           dec   @r3 _____________________________________@R 3   DEC,
0120           dec   @32 _____________________________________@ 32   DEC,
```

Object Code	Zilog asmS8 Mnemonic		FORTH asmS8 Mnemonic

```
80c2           decw   rr2 ____________________________________RR 2   DECW,
8020           decw   32 _____________________________________32   DECW,
81c3           decw   @r3 _____________________________________@R 3   DECW,
8120           decw   @32 _____________________________________@ 32   DECW,

8f               di   ________________________________________      DI,

94c5c2         div    rr2,r5 ________________________________RR 2 , R 5   DIV,
9440c2         div    rr2,64 ________________________________RR 2 , 64   DIV,
94c520         div    32,r5 _________________________________32 , R 5   DIV,
944020         div    32,64 _________________________________32 , 64   DIV,
95c5c2         div    rr2,@r5 _______________________________RR 2 , @R 5   DIV,
9540c2         div    rr2,@64 _______________________________RR 2 , @ 64   DIV,
95c520         div    32,@r5 ________________________________32 , @R 5   DIV,
954020         div    32,@64 ________________________________32 , @ 64   DIV,
9640c2         div    rr2,#64 _______________________________RR 2 , # 64   DIV,
964020         div    32,#64 ________________________________32 , # 64   DIV,

3afe          djnz    r3,$ __________________________________R 3 , $   DJNZ,
```

```
9f              ei   ____________________________________________   EI,

1f           enter   ____________________________________________   ENTER,

2f            exit   ____________________________________________   EXIT,

3e             inc   r3 ____________________________________R 3     INC,
2020           inc   32 ____________________________________32     INC,
21c3           inc   @r3 ___________________________________@R 3    INC,
2120           inc   @32 ___________________________________@ 32    INC,

a0c2          incw   rr2 __________________________________RR 2    INCW,
a020          incw   32 ___________________________________32     INCW,
a1c3          incw   @r3 __________________________________@R 3    INCW,
a120          incw   @32 __________________________________@ 32    INCW,

bf            iret   ____________________________________________   IRET,

8d0400          jp   1024 ________________________________1024     JP,
ed0400          jp   nz,1024 _____________________________NZ , 1024 JP,
30c2            jp   @rr2 ________________________________@RR 2    JP,
3020            jp   @32 _________________________________@ 32    JP,

8bfe            jr   $ ___________________________________$        JR,
ebfe            jr   nz,$ ________________________________NZ , $    JR,

3c40            ld   r3,#64 ______________________________R 3 , # 64 LD,

38c5            ld   r3,r5 _______________________________R 3 , R 5  LD,
3840            ld   r3,64 _______________________________R 3 , 64   LD,
5920            ld   32,r5 _______________________________32 , R 5  LD,
```

Object Code	Zilog asmS8 Mnemonic		FORTH asmS8 Mnemonic

```
c735            ld   r3,@r5 ______________________________R 3 , @R 5 LD,
d735            ld   @r3,r5 ______________________________@R 3 , R 5 LD,

e44020          ld   32,64 _______________________________32 , 64   LD,

c735            ld   r3,@r5 ______________________________R 3 , @R 5 LD,

e540c3          ld   r3,@64 ______________________________R 3 , @ 64 LD,
e5c520          ld   32,@r5 ______________________________32 , @R 5 LD,
e54020          ld   32,@64 ______________________________32 , @ 64 LD,

3c40            ld   r3,#64 ______________________________R 3 , # 64 LD,
e62040          ld   32,#64 ______________________________32 , # 64 LD,
d6c340          ld   @r3,#64 _____________________________@R 3 , # 64 LD,
d62040          ld   @32,#64 _____________________________@ 32 , # 64 LD,

d735            ld   @r3,r5 ______________________________@R 3 , R 5 LD,
f540c3          ld   @r3,64 ______________________________@R 3 , 64  LD,
f5c520          ld   @32,r5 ______________________________@ 32 , R 5 LD,
f54020          ld   @32,64 ______________________________@ 32 , 64  LD,

873540          ld   r3,64(r5) ___________________________R 3 , 64 (R 5  LD,
975340          ld   64(r3),r5 ___________________________64 (R 3 , R 5  LD,

473ec5         ldb   r3,r5,#7 ___________________________R 3 , R 5 # 7  LDB,
473e40         ldb   r3,64,#7 ___________________________R 3 , 64 # 7   LDB,
475fc3         ldb   r3,#7,r5 ___________________________R 3 # 7 , R 5  LDB,
475f20         ldb   32,#7,r5 ___________________________32 # 7 , R 5  LDB,


a7340004       ldc   r3,1024(rr4) _____________________R 3 , 1024 (RR 4  LDC,
e73440         ldc   r3,64(rr4) _______________________R 3 , 64 (RR 4   LDC,
```

```
b7520004        ldc   1024(rr2),r5 _________________1024 (RR 2 , R 5   LDC,
f75240          ldc   64(rr2),r5 ___________________64 (RR 2 , R 5   LDC,
b7502000        ldc   32,r5 _______________________________32 , R 5   LDC,
a7504000        ldc   r5,64 _______________________________R 5 , 64   LDC,
c334            ldc   r3,@rr4 _____________________________R 3 , @RR 4   LDC,
d352            ldc   @rr2,r5 _____________________________@RR 2 , R 5   LDC,

e234           ldcd   r3,@rr4 _____________________________R 3 , @RR 4   LDCD,
e334           ldci   r3,@rr4 _____________________________R 3 , @RR 4   LDCI,
f252          ldcpd   @rr2,r5 _____________________________@RR 2 , R 5   LDCPD,
f352          ldcpi   @rr2,r5 _____________________________@RR 2 , R 5   LDCPI,

a7350004        lde   r3,1024(rr4) _________________R 3 , 1024 (RR 4   LDE,
e73540          lde   r3,64(rr4) ___________________R 3 , 64 (RR 4   LDE,
b7530004        lde   1024(rr2),r5 _________________1024 (RR 2 , R 5   LDE,
f75340          lde   64(rr2),r5 ___________________64 (RR 2 , R 5   LDE,
b7512000        lde   32,r5 _______________________________32 , R 5   LDE,
a7514000        lde   r5,64 _______________________________R 5 , 64   LDE,
c335            lde   r3,@rr4 _____________________________R 3 , @RR 4   LDE,
d353            lde   @rr2,r5 _____________________________@RR 2 , R 5   LDE,
```

Object Code Zilog asmS8 Mnemonic FORTH asmS8 Mnemonic

```
e235           lded   r3,@rr4 _____________________________R 3 , @RR 4   LDED,
e335           ldei   r3,@rr4 _____________________________R 3 , @RR 4   LDEI,
f253          ldepd   @rr2,r5 _____________________________@RR 2 , R 5   LDEPD,
f353          ldepi   @rr2,r5 _____________________________@RR 2 , R 5   LDEPI,

c4c4c2          ldw   rr2,rr4 _____________________________RR 2 , RR 4   LDW,
c440c2          ldw   rr2,64 ______________________________RR 2 , 64   LDW,
c4c420          ldw   32,rr4 ______________________________32 , RR 4   LDW,
c44020          ldw   32,64 _______________________________32 , 64   LDW,
c5c4c2          ldw   rr2,@r4 _____________________________RR 2 , @R 4   LDW,
c540c2          ldw   rr2,@64 _____________________________RR 2 , @ 64   LDW,
c5c420          ldw   32,@r4 ______________________________32 , @R 4   LDW,
c54020          ldw   32,@64 ______________________________32 , @ 64   LDW,

c6c20400        ldw   rr2,#1024 ___________________________RR 2 , # 1024   LDW,
c6200400        ldw   32,#1024 ____________________________32 , # 1024   LDW,

84c5c2         mult   rr2,r5 _____________________________RR 2 , R 5   MULT,
8440c2         mult   rr2,64 _____________________________RR 2 , 64   MULT,
84c520         mult   32,r5 ______________________________32 , R 5   MULT,
844020         mult   32,64 ______________________________32 , 64   MULT,
85c5c2         mult   rr2,@r5 ____________________________RR 2 , @R 5   MULT,
8540c2         mult   rr2,@64 ____________________________RR 2 , @ 64   MULT,
85c520         mult   32,@r5 _____________________________32 , @R 5   MULT,
854020         mult   32,@64 _____________________________32 , @ 64   MULT,
8640c2         mult   rr2,#64 ____________________________RR 2 , # 64   MULT,
864020         mult   32,#64 _____________________________32 , # 64   MULT,

0f             next   ____________________________________________   NEXT,

ff              nop   ____________________________________________   NOP,

4235             or   r3,r5 ______________________________R 3 , R 5   OR,
4335             or   r3,@r5 _____________________________R 3 , @R 5   OR,
4440c3           or   r3,64 ______________________________R 3 , 64   OR,
44c520           or   32,r5 ______________________________32 , R 5   OR,
444020           or   32,64 ______________________________32 , 64   OR,
4540c3           or   r3,@64 _____________________________R 3 , @ 64   OR,
45c520           or   32,@r5 _____________________________32 , @R 5   OR,
454020           or   32,@64 _____________________________32 , @ 64   OR,
46c340           or   r3,#64 _____________________________R 3 , # 64   OR,
462040           or   32,#64 _____________________________32 , # 64   OR,
```

```
50c3            pop   r3 _____________________________________R 3   POP,
5020            pop   32 ____________________________________32   POP,
51c3            pop   @r3 ___________________________________@R 3   POP,
5120            pop   @32 ___________________________________@ 32   POP,

92c5c3        popud   r3,@r5 _______________________________R 3 , @R 5   POPUD,
9240c3        popud   r3,@64 _______________________________R 3 , @ 64   POPUD,
92c520        popud   32,@r5 _______________________________32 , @R 5   POPUD,
924020        popud   32,@64 _______________________________32 , @ 64   POPUD,
```

Object Code Zilog asmS8 Mnemonic FORTH asmS8 Mnemonic

```
93c5c3        popui   r3,@r5 _______________________________R 3 , @R 5   POPUI,
9340c3        popui   r3,@64 _______________________________R 3 , @ 64   POPUI,
93c520        popui   32,@r5 _______________________________32 , @R 5   POPUI,
934020        popui   32,@64 _______________________________32 , @ 64   POPUI,

70c3           push   r3 _____________________________________R 3   PUSH,
7020           push   32 ____________________________________32   PUSH,
71c3           push   @r3 ___________________________________@R 3   PUSH,
7120           push   @32 ___________________________________@ 32   PUSH,

82c3c5       pushud   @r3,r5 _______________________________@R 3 , R 5   PUSHUD,
82c340       pushud   @r3,64 _______________________________@R 3 , 64   PUSHUD,
8220c5       pushud   @32,r5 _______________________________@ 32 , R 5   PUSHUD,
822040       pushud   @32,64 _______________________________@ 32 , 64   PUSHUD,

83c3c5       pushui   @r3,r5 _______________________________@R 3 , R 5   PUSHUI,
83c340       pushui   @r3,64 _______________________________@R 3 , 64   PUSHUI,
8320c5       pushui   @32,r5 _______________________________@ 32 , R 5   PUSHUI,
832040       pushui   @32,64 _______________________________@ 32 , 64   PUSHUI,

cf              rcf   _____________________________________________   RCF,

\ d5a5          rdr   #0a5h _________________________________# 0 A 5 H   RDR,

af              ret   _____________________________________________   RET,

90c3             rl   r3 _____________________________________R 3   RL,
9020             rl   32 ____________________________________32   RL,
91c3             rl   @r3 ___________________________________@R 3   RL,
9120             rl   @32 ___________________________________@ 32   RL,

10c3            rlc   r3 _____________________________________R 3   RLC,
1020            rlc   32 ____________________________________32   RLC,
11c3            rlc   @r3 ___________________________________@R 3   RLC,
1120            rlc   @32 ___________________________________@ 32   RLC,

e0c3             rr   r3 _____________________________________R 3   RR,
e020             rr   32 ____________________________________32   RR,
e1c3             rr   @r3 ___________________________________@R 3   RR,
e120             rr   @32 ___________________________________@ 32   RR,

c0c3            rrc   r3 _____________________________________R 3   RRC,
c020            rrc   32 ____________________________________32   RRC,
c1c3            rrc   @r3 ___________________________________@R 3   RRC,
c120            rrc   @32 ___________________________________@ 32   RRC,

4f              sb0   _____________________________________________   SB0,

5f              sb1   _____________________________________________   SB1,

3235            sbc   r3,r5 _________________________________R 3 , R 5   SBC,
3335            sbc   r3,@r5 ________________________________R 3 , @R 5   SBC,
3440c3          sbc   r3,64 _________________________________R 3 , 64   SBC,
```

Object Code Zilog asmS8 Mnemonic FORTH asmS8 Mnemonic

```
34c520          sbc   32,r5 _____________________________________32 , R 5   SBC,
344020          sbc   32,64 _____________________________________32 , 64   SBC,
3540c3          sbc   r3,@64 ___________________________________R 3 , @ 64   SBC,
35c520          sbc   32,@r5 ___________________________________32 , @R 5   SBC,
354020          sbc   32,@64 ___________________________________32 , @ 64   SBC,
36c340          sbc   r3,#64 ___________________________________R 3 , # 64   SBC,
362040          sbc   32,#64 ___________________________________32 , # 64   SBC,

df              scf   _____________________________________________________  SCF,

d0c3            sra   r3 _____________________________________________R 3   SRA,
d020            sra   32 ____________________________________________32   SRA,
d1c3            sra   @r3 ___________________________________________@R 3   SRA,
d120            sra   @32 ___________________________________________@ 32   SRA,

3180            srp   #128 __________________________________________# 128   SRP,
3181           srp1   #128 __________________________________________# 128   SRP1,
3182           srp0   #128 __________________________________________# 128   SRP0,

2235            sub   r3,r5 ___________________________________R 3 , R 5   SUB,
2335            sub   r3,@r5 __________________________________R 3 , @R 5   SUB,
2440c3          sub   r3,64 ___________________________________R 3 , 64   SUB,
24c520          sub   32,r5 ___________________________________32 , R 5   SUB,
244020          sub   32,64 ___________________________________32 , 64   SUB,
2540c3          sub   r3,@64 __________________________________R 3 , @ 64   SUB,
25c520          sub   32,@r5 __________________________________32 , @R 5   SUB,
254020          sub   32,@64 __________________________________32 , @ 64   SUB,
26c340          sub   r3,#64 __________________________________R 3 , # 64   SUB,
262040          sub   32,#64 __________________________________32 , # 64   SUB,

f0c3           swap   r3 _____________________________________________R 3   SWAP,
f020           swap   32 ____________________________________________32   SWAP,
f1c3           swap   @r3 ___________________________________________@R 3   SWAP,
f120           swap   @32 ___________________________________________@ 32   SWAP,

6235            tcm   r3,r5 ___________________________________R 3 , R 5   TCM,
6335            tcm   r3,@r5 __________________________________R 3 , @R 5   TCM,
6440c3          tcm   r3,64 ___________________________________R 3 , 64   TCM,
64c520          tcm   32,r5 ___________________________________32 , R 5   TCM,
644020          tcm   32,64 ___________________________________32 , 64   TCM,
6540c3          tcm   r3,@64 __________________________________R 3 , @ 64   TCM,
65c520          tcm   32,@r5 __________________________________32 , @R 5   TCM,
654020          tcm   32,@64 __________________________________32 , @ 64   TCM,
66c340          tcm   r3,#64 __________________________________R 3 , # 64   TCM,
662040          tcm   32,#64 __________________________________32 , # 64   TCM,

7235             tm   r3,r5 ___________________________________R 3 , R 5   TM,
7335             tm   r3,@r5 __________________________________R 3 , @R 5   TM,
7440c3           tm   r3,64 ___________________________________R 3 , 64   TM,
74c520           tm   32,r5 ___________________________________32 , R 5   TM,
744020           tm   32,64 ___________________________________32 , 64   TM,
7540c3           tm   r3,@64 __________________________________R 3 , @ 64   TM,
75c520           tm   32,@r5 __________________________________32 , @R 5   TM,
```

Object Code	Zilog asmS8 Mnemonic	FORTH asmS8 Mnemonic

```
754020           tm   32,@64 __________________________________32 , @ 64   TM,
76c340           tm   r3,#64 __________________________________R 3 , # 64   TM,
762040           tm   32,#64 __________________________________32 , # 64   TM,

b235            xor   r3,r5 ___________________________________R 3 , R 5   XOR,
b335            xor   r3,@r5 __________________________________R 3 , @R 5   XOR,
b440c3          xor   r3,64 ___________________________________R 3 , 64   XOR,
b4c520          xor   32,r5 ___________________________________32 , R 5   XOR,
b44020          xor   32,64 ___________________________________32 , 64   XOR,
b540c3          xor   r3,@64 __________________________________R 3 , @ 64   XOR,
b5c520          xor   32,@r5 __________________________________32 , @R 5   XOR,
b54020          xor   32,@64 __________________________________32 , @ 64   XOR,
```

```
b6c340          xor   r3,#64 _____________________________R 3 , # 64   XOR,
b62040          xor   32,#64 _____________________________32 , # 64   XOR,

3f              wfi   _____________________________________________   WFI,

;defined register names

38de             ld   r3,sym _____________________________R 3 , SYM   LD,
38dd             ld   r3,imr _____________________________R 3 , IMR   LD,
38dc             ld   r3,irr _____________________________R 3 , IRR   LD,
c4dac2          ldw   rr2,ip _____________________________RR 2 , IP   LDW,
38db             ld   r3,ipl _____________________________R 3 , IPL   LD,
38da             ld   r3,iph _____________________________R 3 , IPH   LD,
c4d8c2          ldw   rr2,sp _____________________________RR 2 , SP   LDW,
38d9             ld   r3,spl _____________________________R 3 , SPL   LD,
38d8             ld   r3,sph _____________________________R 3 , SPH   LD,
38d7             ld   r3,rp1 _____________________________R 3 , RP1   LD,
38d6             ld   r3,rp0 _____________________________R 3 , RP0   LD,
38d5             ld   r3,flags ___________________________R 3 , FLAGS   LD,
38d4             ld   r3,p4 _____________________________R 3 , P4   LD,
38d3             ld   r3,p3 _____________________________R 3 , P3   LD,
38d2             ld   r3,p2 _____________________________R 3 , P2   LD,
38d1             ld   r3,p1 _____________________________R 3 , P1   LD,
38d0             ld   r3,p0 _____________________________R 3 , P0   LD,

;Bank 0 Special Registers

38ff             ld   r3,ipr _____________________________R 3 , IPR   LD,
38fe             ld   r3,emt _____________________________R 3 , EMT   LD,
38fd             ld   r3,p2bip ___________________________R 3 , P2BIP   LD,
38fc             ld   r3,p2aip ___________________________R 3 , P2AIP   LD,
38fb             ld   r3,p2dm _____________________________R 3 , P2DM   LD,
38fa             ld   r3,p2cm _____________________________R 3 , P2CM   LD,
38f9             ld   r3,p2bm _____________________________R 3 , P2BM   LD,
38f8             ld   r3,p2am _____________________________R 3 , P2AM   LD,
38f7             ld   r3,p4od _____________________________R 3 , P4OD   LD,
38f6             ld   r3,p4d _____________________________R 3 , P4D   LD,
38f5             ld   r3,h1c _____________________________R 3 , H1C   LD,
38f4             ld   r3,h0c _____________________________R 3 , H0C   LD,
38f1             ld   r3,pm _____________________________R 3 , PM   LD,
38d1             ld   r3,p1 _____________________________R 3 , P1   LD,
```

Object Code Zilog asmS8 Mnemonic FORTH asmS8 Mnemonic

```
38f0             ld   r3,p0m _____________________________R 3 , P0M   LD,
38ed             ld   r3,uie _____________________________R 3 , UIE   LD,
38ec             ld   r3,urc _____________________________R 3 , URC   LD,
38eb             ld   r3,utc _____________________________R 3 , UTC   LD,
38ea             ld   r3,sio _____________________________R 3 , SIO   LD,
38e9             ld   r3,sie _____________________________R 3 , SIE   LD,
38e8             ld   r3,srcb _____________________________R 3 , SRCB   LD,
38e7             ld   r3,srca _____________________________R 3 , SRCA   LD,
38e6             ld   r3,stc _____________________________R 3 , STC   LD,
c4e4c2          ldw   rr2,c1c _____________________________RR 2 , C1C   LDW,
38e5             ld   r3,c1cl _____________________________R 3 , C1CL   LD,
38e4             ld   r3,c1ch _____________________________R 3 , C1CH   LD,
c4e2c2          ldw   rr2,c0c _____________________________RR 2 , C0C   LDW,
38e3             ld   r3,c0cl _____________________________R 3 , C0CL   LD,
38e2             ld   r3,c0ch _____________________________R 3 , C0CH   LD,
38e1             ld   r3,c1ct _____________________________R 3 , C1CT   LD,
38e0             ld   r3,c0ct _____________________________R 3 , C0CT   LD,

;Bank 1 Special Registers

38ff             ld   r3,wumsk _____________________________R 3 , WUMSK   LD,
38fe             ld   r3,wumch _____________________________R 3 , WUMCH   LD,
38fb             ld   r3,umb _____________________________R 3 , UMB   LD,
38fa             ld   r3,uma _____________________________R 3 , UMA   LD,
```

```
c4f8c2          ldw   rr2,ubg _____________________________RR 2 , UBG   LDW,
38f9             ld   r3,ubgl _____________________________R 3 , UBGL   LD,
38f8             ld   r3,ubgh _____________________________R 3 , UBGH   LD,
c4f0c2          ldw   rr2,dc ______________________________RR 2 , DC   LDW,
38f1             ld   r3,dcl ______________________________R 3 , DCL   LD,
38f0             ld   r3,dch ______________________________R 3 , DCH   LD,
c4eec2          ldw   rr2,syn _____________________________RR 2 , SYN   LDW,
38ef             ld   r3,synh _____________________________R 3 , SYNH   LD,
38ee             ld   r3,synl _____________________________R 3 , SYNL   LD,
38ed             ld   r3,smd ______________________________R 3 , SMD   LD,
38ec             ld   r3,smc ______________________________R 3 , SMC   LD,
38eb             ld   r3,smb ______________________________R 3 , SMB   LD,
38ea             ld   r3,sma ______________________________R 3 , SMA   LD,
c4e8c2          ldw   rr2,sbg _____________________________RR 2 , SBG   LDW,
38e9             ld   r3,sbgl _____________________________R 3 , SBGL   LD,
38e8             ld   r3,sbgh _____________________________R 3 , SBGH   LD,
c4e4c2          ldw   rr2,c1tc ____________________________RR 2 , C1TC   LDW,
38e5             ld   r3,c1tcl ____________________________R 3 , C1TCL   LD,
38e4             ld   r3,c1tch ____________________________R 3 , C1TCH   LD,
c4e2c2          ldw   rr2,c0tc ____________________________RR 2 , C0TC   LDW,
38e3             ld   r3,c0tcl ____________________________R 3 , C0TCL   LD,
38e2             ld   r3,c0tch ____________________________R 3 , C0TCH   LD,
38e1             ld   r3,c1m ______________________________R 3 , C1M   LD,
38e0             ld   r3,c0m ______________________________R 3 , C0M   LD,

;upper case test

38de             ld   r3,SYM ______________________________R 3 , SYM   LD,
38dd             ld   r3,IMR ______________________________R 3 , IMR   LD,
```

Object Code Zilog asmS8 Mnemonic FORTH asmS8 Mnemonic

```
38dc             ld   r3,IRR ______________________________R 3 , IRR   LD,
c4dac2          ldw   rr2,IP ______________________________RR 2 , IP   LDW,
38db             ld   r3,IPL ______________________________R 3 , IPL   LD,
38da             ld   r3,IPH ______________________________R 3 , IPH   LD,
c4d8c2          ldw   rr2,SP ______________________________RR 2 , SP   LDW,
38d9             ld   r3,SPL ______________________________R 3 , SPL   LD,
38d8             ld   r3,SPH ______________________________R 3 , SPH   LD,
38d7             ld   r3,RP1 ______________________________R 3 , RP1   LD,
38d6             ld   r3,RP0 ______________________________R 3 , RP0   LD,
38d5             ld   r3,FLAGS ____________________________R 3 , FLAGS   LD,
38d4             ld   r3,P4 _______________________________R 3 , P4   LD,
38d3             ld   r3,P3 _______________________________R 3 , P3   LD,
38d2             ld   r3,P2 _______________________________R 3 , P2   LD,
38d1             ld   r3,P1 _______________________________R 3 , P1   LD,
38d0             ld   r3,P0 _______________________________R 3 , P0   LD,

;Bank 0 Special Registers

38ff             ld   r3,IPR ______________________________R 3 , IPR   LD,
38fe             ld   r3,P2BIP ____________________________R 3 , P2BIP   LD,
38fc             ld   r3,P2AIP ____________________________R 3 , P2AIP   LD,
38fb             ld   r3,P2DM _____________________________R 3 , P2DM   LD,
38fa             ld   r3,P2CM _____________________________R 3 , P2CM   LD,
38f9             ld   r3,P2BM _____________________________R 3 , P2BM   LD,
38f8             ld   r3,P2AM _____________________________R 3 , P2AM   LD,
38f7             ld   r3,P4OD _____________________________R 3 , P4OD   LD,
38f6             ld   r3,P4D ______________________________R 3 , P4D   LD,
38f5             ld   r3,H1C ______________________________R 3 , H1C   LD,
38f4             ld   r3,H0C ______________________________R 3 , H0C   LD,
38f1             ld   r3,PM _______________________________R 3 , PM   LD,
38d1             ld   r3,P1 _______________________________R 3 , P1   LD,
38f0             ld   r3,P0M ______________________________R 3 , P0M   LD,
38ed             ld   r3,UIE ______________________________R 3 , UIE   LD,
38ec             ld   r3,URC ______________________________R 3 , URC   LD,
38eb             ld   r3,UTC ______________________________R 3 , UTC   LD,
38ea             ld   r3,SIO ______________________________R 3 , SIO   LD,
```

```
38e9              ld   r3,SIE _____________________________________R 3 , SIE   LD,
38e8              ld   r3,SRCB ____________________________________R 3 , SRCB  LD,
38e7              ld   r3,SRCA ____________________________________R 3 , SRCA  LD,
38e6              ld   r3,STC _____________________________________R 3 , STC   LD,
c4e4c2           ldw   rr2,C1C ___________________________________RR 2 , C1C   LDW,
38e5              ld   r3,C1CL ____________________________________R 3 , C1CL  LD,
38e4              ld   r3,C1CH ____________________________________R 3 , C1CH  LD,
c4e2c2           ldw   rr2,C0C ___________________________________RR 2 , C0C   LDW,
38e3              ld   r3,C0CL ____________________________________R 3 , C0CL  LD,
38e2              ld   r3,C0CH ____________________________________R 3 , C0CH  LD,
38e1              ld   r3,C1CT ____________________________________R 3 , C1CT  LD,
38e0              ld   r3,C0CT ____________________________________R 3 , C0CT  LD,

;Bank 1 Spezial Registers

38ff              ld   r3,WUMSK ___________________________________R 3 , WUMSK  LD,
38fe              ld   r3,WUMCH ___________________________________R 3 , WUMCH  LD,
```

Object Code Zilog asmS8 Mnemonic FORTH asmS8 Mnemonic

```
38fb              ld   r3,UMB _____________________________________R 3 , UMB   LD,
38fa              ld   r3,UMA _____________________________________R 3 , UMA   LD,
c4f8c2           ldw   rr2,UBG ___________________________________RR 2 , UBG   LDW,
38f9              ld   r3,UBGL ____________________________________R 3 , UBGL  LD,
38f8              ld   r3,UBGH ____________________________________R 3 , UBGH  LD,
c4f0c2           ldw   rr2,DC ____________________________________RR 2 , DC    LDW,
38f1              ld   r3,DCL _____________________________________R 3 , DCL   LD,
38f0              ld   r3,DCH _____________________________________R 3 , DCH   LD,
c4eec2           ldw   rr2,SYN ___________________________________RR 2 , SYN   LDW,
38ef              ld   r3,SYNH ____________________________________R 3 , SYNH  LD,
38ee              ld   r3,SYNL ____________________________________R 3 , SYNL  LD,
38ed              ld   r3,SMD _____________________________________R 3 , SMD   LD,
38ec              ld   r3,SMC _____________________________________R 3 , SMC   LD,
38eb              ld   r3,SMB _____________________________________R 3 , SMB   LD,
38ea              ld   r3,SMA _____________________________________R 3 , SMA   LD,
c4e8c2           ldw   rr2,SBG ___________________________________RR 2 , SBG   LDW,
38e9              ld   r3,SBGL ____________________________________R 3 , SBGL  LD,
38e8              ld   r3,SBGH ____________________________________R 3 , SBGH  LD,
c4e4c2           ldw   rr2,C1TC ___________________________________RR 2 , C1TC  LDW,
38e5              ld   r3,C1TCL ___________________________________R 3 , C1TCL LD,
38e4              ld   r3,C1TCH ___________________________________R 3 , C1TCH LD,
c4e2c2           ldw   rr2,C0TC ___________________________________RR 2 , C0TC  LDW,
38e3              ld   r3,C0TCL ___________________________________R 3 , C0TCL LD,
38e2              ld   r3,C0TCH ___________________________________R 3 , C0TCH LD,
38e1              ld   r3,C1M _____________________________________R 3 , C1M   LD,
38e0              ld   r3,C0M _____________________________________R 3 , C0M   LD,

;test for condition codes

0d0080            jp   f,128 ______________________________________F , 128   JP,

6d0080            jp   z,128 ______________________________________Z , 128   JP,
ed0080            jp   nz,128 _____________________________________NZ , 128  JP,
6d0080            jp   eq,128 _____________________________________EQ , 128  JP,
ed0080            jp   ne,128 _____________________________________NE , 128  JP,

7d0080            jp   c,128 ______________________________________C , 128   JP,
fd0080            jp   nc,128 _____________________________________NC , 128  JP,

ad0080            jp   gt,128 _____________________________________GT , 128  JP,
1d0080            jp   lt,128 _____________________________________LT , 128  JP,
9d0080            jp   ge,128 _____________________________________GE , 128  JP,
2d0080            jp   le,128 _____________________________________LE , 128  JP,

dd0080            jp   pl,128 _____________________________________PL , 128  JP,
5d0080            jp   mi,128 _____________________________________MI , 128  JP,
```

```
cd0080          jp   nov,128 _____________________________NOV , 128   JP,
4d0080          jp   ov,128 ______________________________OV , 128   JP,

Object Code     Zilog asmS8 Mnemonic                       FORTH asmS8 Mnemonic

bd0080          jp   ugt,128 _____________________________UGT , 128   JP,
7d0080          jp   ult,128 _____________________________ULT , 128   JP,
fd0080          jp   uge,128 _____________________________UGE , 128   JP,
3d0080          jp   ule,128 _____________________________ULE , 128   JP,


;test for Makros

F2F6F2E6                     ____________________________________AX   PUSHW
E3E6E3F6                     ____________________________________AX   POPW
F2D6F2C6                     ____________________________________BX   PUSHW
E3C6E3D6                     ____________________________________BX   POPW

C3ECE7FC01                   ________________________________AX , @BX   LDCW
D3ECF7FC01                   ________________________________@BX , AX   LDCW
E7EC40E7FC41                 ____________________________AX , 64 (BX   LDCW
F7EC40F7FC41                 ____________________________64 (BX , AX   LDCW
A7EC0004A7FC0104             ________________________AX , 1024 (BX   LDCW
B7EC0004B7FC0104             ________________________1024 (BX , AX   LDCW
A7E04000A7F04100             ____________________________AX , 64   LDCW
B7E04000B7F04100             ____________________________64 , AX   LDCW
```