
IDE11
Integrated Development Environment

IDE11

Version 2.3

Benutzer-Handbuch

1. Juli 1997

MCT Lange & Thamm Mikrocomputertechnik GbR
Hohe Str. 9-13 D-04107 Leipzig
Telefon: +49-(0)341-2118354
Fax: +49-(0)341-2118355
Email: mctgbr@aol.com

Inhaltsverzeichnis

1. Einführung

Die Entwicklung von (Assembler-) Programmen für Mikrocontroller vollzieht sich stets nach dem gleichen Grundmuster:

- den Quelltext formulieren
- die Quelle übersetzen (assemblieren)
- das Resultat zum Zielsystem übertragen
- Testlauf
- Fehlersuche

Jeder dieser Schritte erfordert spezielle Werkzeuge: Editor, Assembler, Monitorprogramm, Terminalemulator, Debugger etc. Der angegebene Entwicklungszyklus wird im Normalfall sehr oft wiederholt, um sich dem Ziel - ein fehlerfreies Programm - Schritt für Schritt anzunähern. Der Wechsel zwischen all den Werkzeugen verlangt viel Zeit, die Übergabe von Daten bzw. Informationen erfordert vom Benutzer Sorgfalt und ein gutes Gedächtnis. Zudem ist die Bedienung zumeist inkonsistent - ähnliche Befehle und Funktionen sind durch unterschiedlichste Tastenkombinationen bzw. Kommandozeilenschalter zu erreichen.

Zur Eleminierung dieser Nachteile hat die Menschheit "Integrierte Entwicklungsumgebungen" ersonnen (IDE - Integrated Development Environment). Im Gegensatz zu verstreuten Softwareentwicklungs-Utilities stellt eine IDE eine durchgängige Plattform für die Entwicklung einer Software zur Verfügung. Der Benutzer muß nicht mehr mit den syntaktischen Feinheiten diverser Einzelprogramme umgehen, statt dessen findet er alles notwendige wohlgeordnet und konsistent unter einem Dach vor.

IDE11 ist eine solche Integrierte Entwicklungsumgebung für die Motorola Mikrocontroller MC68HC11xx. Die vertraute Benutzeroberflächen (Borlands Turbo Vision) haben wir vereint mit einem effizienten Werkzeugkasten, speziell zugeschnitten auf die Bedürfnisse des HC11 Softwareentwicklers.

1.1. Auf einen Blick

Wie in jeder Entwicklungsumgebung ist der zentrale Punkt der Editor. IDE11 besitzt einen integrierten Multidatei-Editor, mit dessen Hilfe Sie Ihre Quelltexte schreiben bzw. modifizieren können. Dieser Editor läßt sich über einige hinlänglich bekannte Tastatur-Kommandos steuern; alternativ (oder in den meisten Fällen eine Mischung aus beidem) übernimmt die Maus Steuerfunktionen.

Der eingebaute Assembler übersetzt Ihre Quellen dann wunschgemäß in Executables in Form von S-Record Files. Treten Fehler beim Übersetzen auf, so bekommen Sie selbstverständlich eine entsprechende Meldung. Sie müssen dabei nicht erst die Listing-Datei durchstöbern, sondern bekommen die fehlerhafte Quelltextzeile gleich mit angezeigt.

Ein Kommunikationsmodul sorgt dafür, daß Ihr Programm zum Mikrocontroller transferiert werden kann, um dort zur Ausführung zu kommen. Ein serielles Schnittstellenkabel ist die einzige Verbindung zwischen PC und HC11 - der Hardwareaufwand ist also minimal. Die Kommunikation zwischen Host und Target bedarf *keiner* Unterstützung durch einen (EPROM-) Monitor auf der HC11-Seite. Die erforderlichen Routinen werden, je nach Bedarf und unterstützt durch den Bootloader des HC11, "häppchenweise" in den internen RAM des Controllers geladen.

Ausgaben Ihrer Anwendung können Sie in einem Terminalfenster verfolgen. Die Baudrate hierfür ist einstellbar. Zur Fehlersuche können Sie Fenster mit Speicherauszügen (Memory Dump) und Reassemblerlistings erzeugen. Die Informationen können auch als Datei gesichert werden. Hinzu kommt ein einfacher Singlestep-Debugger. Er benutzt das Kommunikationsmodul und den internen Reassembler, um Ihnen Abläufe im Target zu visualisieren. Schritt für Schritt können Sie so den Programmverlauf beobachten. Zum Test der Anwendung existieren Schnittstellen zu Softwaresimulatoren (TESTE68) und EPROM-Emulatoren (EPSIM).

Zentrale Bedeutung hat bei allen HC11-Derivaten das CONFIG Register. Zur Anzeige bzw. Veränderung des Inhalts dieses E^2PROM basierten Registers wurde eigens ein Programmpunkt vorgesehen.

2. Installation

2.1. Softwareinstallation

IDE11 läuft als DOS-Programm im Textmode. An den eingesetzten PC werden keine besonderen Anforderungen gestellt (siehe dazu Anhang A: Systemvoraussetzungen). Die Funktion unter Windows 3.1 ist leider nicht gewährleistet, der direkte Zugriff auf die Hardware der seriellen Schnittstelle durch IDE11 kollidiert zuweilen mit Aktivitäten des Windows-System. Beim Betrieb unter Windows 95 sind jedoch bisher keine Probleme berichtet worden.

IDE11 wird als gepackte Datei (ZIP Archiv) geliefert. Das Archiv enthält alle notwendigen Dateien sowie die folgende Verzeichnisstruktur:

```
IDE11V23.ZIP
           |
           ---BIN
           |
           ---LIB
           |
           ---SRC
```

Die Installation wird wie folgt durchgeführt: Erzeugen Sie zuerst auf der Festplatte ein Installationsverzeichnis für IDE11 und wechseln Sie in dieses Verzeichnis:

```
C:
cd \
md IDE11
cd IDE11
```

Kopieren Sie die Archivdatei in dieses Verzeichnis:

```
copy A:\IDE11V23.ZIP C:\IDE11
```

Entpacken Sie die Archivdatei mittels UNZIP bzw. PKUNZIP. Dabei wird, ausgehend vom gewählten Installationsverzeichnis (z.B. C:\IDE11), die oben gezeigte Unterverzeichnisstruktur automatisch mit angelegt:

```
UNZIP IDE11V23
```

Bemerkung: IDE11 wurde mit Info-ZIP´s Kompressionsprogramm gepackt. Die Installation erfordert UnZip um die Dateien aus dem Archiv zu extrahieren. Info-ZIP´s Software (Zip, UnZip und zugehörige Utilities) sind frei erhältlich und können im Quellcode oder als ausführbare Programme über verschiedene FTP-Server bezogen werden, u.a. über: ftp.uu.net:/pub/archiving/zip/.*

Schließlich kann die Archivdatei IDE11V23.ZIP im Installationsverzeichnis wieder gelöscht werden (Sie sollten dennoch in jedem Fall eine Sicherheitskopie der Originaldatei aufbewahren).

Im Unterverzeichnis IDE11\BIN werden die eigentlichen Systemkomponenten der IDE abgelegt, nehmen Sie dieses Verzeichnis unbedingt in den DOS-Suchpfad auf. Ergänzen Sie dazu die PATH-Anweisung (Programmsuchpfad) in der AUTOEXEC.BAT Datei Ihres Rechners, z.B.:

```
PATH=C:\DOS; . . . ;C:\IDE11\BIN
```

Das Unterverzeichnis IDE11\LIB enthält vorbereitete Bibliotheksroutinen (als Assembler-Quelltext) für den 68HC11. Sie können diese Module bei Bedarf in Ihre eigene Software einbinden.

IDE11\SRC ist das Arbeitsverzeichnis. Es enthält die Quelltexte, aber auch die Listing- und Objektdateien (Executables) werden dort abgelegt. Selbstverständlich kann IDE11 auch von einem beliebigen anderen Arbeitsverzeichnis aus aufgerufen werden. Sinnvoll ist z.B. eine Verzeichnisstruktur, die den verschiedenen in Bearbeitung befindlichen Projekten Rechnung trägt.

Am einfachsten ist der Aufruf von IDE11 über eine Batchdatei, die etwa so aussehen könnte:

```
@echo off
C:
CD \IDE11\SRC
REM === IDE11 benutzt hier COM2 ===
IDE11 -c2
```

Eine solche Batchdatei ist nicht auf der Installationsdiskette enthalten, da sie abhängig ist von den jeweiligen Anwenderpräferenzen. Es wird Ihnen wahrscheinlich keine Probleme bereiten, sich ggf. eine solche Batchdatei selber zu erstellen und an die Gegebenheiten anzupassen.

2.2. Registrierung

Nach der Installation müssen Sie noch die Seriennummer und den Registrierungskey eingeben, um die Software für Ihre Anwendung zu personalisieren. Diesen Registrierung genannten Vorgang müssen Sie nur einmalig durchführen. Nach der Eingabe werden Ihre Daten gespeichert und Ihre Seriennummer (bzw. der Benutzername) wird fortan im About-Dialogfeld angezeigt. Ohne Registrierung läuft das Programm nur als 30-Tage-Eval-Version (mit vollem Funktionsumfang).

Die Registrierung führen Sie wie folgt durch: Nach Start des Programms (Klicken Sie imAbout-Dialogfeld auf OK) erscheint im unregistrierten Zustand ein Licence-Info-Dialog. Klicken Sie auf "Register Now!", es erscheinen zwei Eingabezeilen für die Registrierungsinformationen. Geben Sie hier die Seriennummer (Ziffernkombination bzw. Benutzername) und den Registrierungskey exakt so ein, wie Sie diese Informationen erhalten haben - unter Berücksichtigung von Groß- bzw. Kleinschreibung. Zwischen den Feldern können Sie mit der Maus oder mittels TAB wechseln. Klicken Sie nach Überprüfung Ihrer Eingabe auf OK. Es erscheint nun die Mitteilung über die erfolgreiche Registrierung, ab sofort entfallen alle überflüssigen Hinweise.

Wenn Sie die Meldung "This is not a valid serial number" erhalten, steckt in einer der beiden Eingabezeilen noch ein Fehler. Ein zweiter Versuch dürfte erfolgreich sein. Wenn nicht, wenden Sie sich bitte an den Autor, um Hilfestellung zu erhalten.

Seriennummer und Registrierungskey identifizieren Ihre persönliche Programmkopie als rechtmäßig erworbene Lizenz (siehe hierzu auch die Datei LICENSE.DOC). Sie dürfen diese Informationen (insbesondere den Registrierungskey) keinesfalls Dritten zugänglich machen, dies wäre gleichbedeutend mit der Weitergabe einer Raubkopie.

Das Programm selbst hingegen können Sie gern an Kollegen und Freunde weitergeben, wenn diese IDE11 näher in Augenschein nehmen wollen. Bitte fertigen Sie Kopien ausschließlich von der Originaldiskette bzw. der unveränderten Archivdatei IDE11V23.ZIP an. Wenn Sie mit IDE11 zufrieden sind, so empfehlen Sie unser Programm bitte weiter (...wenn nicht, sagen Sie es bitte *uns*) - vielen Dank!

Selbstverständlich können Sie jederzeit IDE11 von einem Computer entfernen und das Programm stattdessen auf einem anderen Computer weiterverwenden - Sie müssen nur die Registrierungsinformation neu eingeben. Wenn Sie IDE11 gleichzeitig auf mehreren Rechnern verwenden wollen, so wenden Sie sich bitte an Ihren Händler zur Vereinbarung einer Mehrfach- bzw. Netzwerklizenz.

Der Einsatz einer (Einzelplatz-) Lizenz in einem Netzwerk ist zwar möglich, trotzdem ist die Benutzung nur auf einem Computer zulässig. Aufgrund des überschaubaren Platzbedarfs von IDE11 empfehlen wir im Normalfall die Installation auf der lokalen Festplatte.

2.3. Hardwareinstallation

Sie können IDE11 verwenden, um lediglich HC11-Assemblerfiles zu schreiben und zu übersetzen, ohne irgendeine HC11-Hardware anzuschließen. Der Vorteil von IDE11 besteht aber gerade in der Eigenschaft, echtes Silizium zu programmieren und die geschriebenen Programme darauf auszutesten. Um ein HC11-Board (Target) mit dem PC (und somit IDE11) zu verbinden, sind nur einige Leitungen anzuschließen und einige simple Regeln zu befolgen.

Zuerst zum Anschluß: PC und Target müssen über ein serielles Nullmodemkabel verbunden werden. Auf der Seite des Mikrocontrollers muß selbstverständlich ein **Pegelwandler**

(Interfacemodul) vorhanden sein, der die TTL-Signale des 68HC11 auf den V.24-Pegel für den PC (und vice versa) umsetzt. Folgende Verbindungen sind zwischen PC und Target herzustellen:

Host-PC Sub-D9F Pin-Nr.	Host Signal Name		Target Signal Name	Target Sub-D9F Pin-Nr.	Zum HC11 Port/Pin: (*)
3	TxD	<----->	RxD	2	PD0
2	RxD	<----->	TxD	3	PD1
5	GND	<----->	GND	5	VSS
4	DTR	<----->	DSR	6	RESET(#)

(*) Pegelumsetzer V.24 / TTL erforderlich
(#) Optional: Hierdurch kann der PC einen Reset des Controllers bewirken

Weitere Bedingungen sind:

- Der HC11 muß eine Taktfrequenz von 8 MHz haben
- Die beiden Mode-Pins (MODA und MODB) müssen mit L-Pegel (GND) verbunden werden, d.h. der HC11 muß nach Reset im Special Bootstrap Mode arbeiten. Die Umschaltung in den Expanded Multiplex Mode erfolgt bedarfsweise per Software (es sei denn, die Umschaltung der Betriebsart des HC11 soll vom PC über RTS ferngesteuert bzw. manuell vorgenommen werden)

2.4. Unterstützte Singleboardcomputer

IDE11 kann sich auf unterschiedliche Zielhardware einstellen, dem Programm muß lediglich mitgeteilt werden, welcher HC11-Typ zum Einsatz kommt, ob extern Speicherbausteine (RAM oder EEPROM) angeschlossen sind, und welche Methode für die Resetsteuerung und ggf. die Mode-Steuerung verwendet werden soll.

Setzen Sie eines der in der folgenden Liste aufgeführten Produkte ein, haben Sie nicht viel Arbeit - Sie müssen lediglich im Menü "Options/Target" den passenden Typ einstellen. Versuchen Sie nicht, die über "Detail" angezeigten Settings zu verändern - sie lassen sich nicht modifizieren. Eigene Kompositionen können Sie über den ersten Listeneintrag ("User defined") realisieren - nach Betätigen des "Detail"-Buttons.

ZWERG11A (mit MC68HC11A1)

Die im Handbuch des Zwerg11A beschriebene Verbindung zwischen Anschluß DSR am IF-Modul und der Resetleitung des HC11 wird als vorhanden vorausgesetzt.

ZWERG11A/E2 (mit MC68HC811E2)

Anmerkung siehe oben.

ZWERG11plus

Zwerg11plus mit 32 KB RAM und 32 KB EEPROM

IC11B

Scheckkartenmodul mit 64 KB RAM.

MOPS11

Das 68HC11A1 Board, vorgestellt in der Zeitschrift "elrad" ab Heft 3/91. Mit 64 KB RAM. Einstellung ebenso für MOPS 2.1 geeignet.

MOPS-light

Ebenfalls erschienen in "elrad" (ab Heft 2/94), jedoch ausgestattet mit der MCU 68HC11F1. Mit 32 KB RAM.

MOPS-XL

68HC11F1 extra-light - ohne externe Speicher. Sonst wie MOPS-light.

PCMCIA11

Unser Datenlogger mit PCMCIA-Interface und MCU 68HC11A1. Mit 32 KB RAM ab $8000.

MOT EVBU

Motorola M68HC11EVBU - Universal Evaluation Board, bestückt mit einem 68HC11E1 (bzw. E9)

HC11-MiniKit

Das Projekt aus der Zeitschrift "Funkamateur" ab Heft 9/94; voreingestellt ist derzeit die MCU 68HC11E1. Bei einer Bestückung mit einer anderen MCU (z.B. 68HC11A1) ist der benutzerdefinierte Typ zu verwenden!

ZSLIC11

Miniaturisierte Controllerbaugruppe mit 68HC11E1 und dem "SLIC Micro Peripheral" X68C75 von Xicor (8KB EEPROM).

Achtung: Gelegentlich setzen die Board-Hersteller auch (gänzlich oder weitgehend kompatible) Ersatztypen ein. Controller von Toshiba (TMP68HC11 statt MC68HC11) sind mit den Motorola Bausteinen bis ins letzte Bit identisch. Auch Maskentypen von Motorola oder Toshiba (erkennbar an zuweilen "kryptischen" Typbezeichnungen) stellen kein Problem dar. Immer öfter wird jedoch der HC11A8 durch den pinkompatiblen HC11E9 (bzw A1 durch E1) ersetzt - das ist Vorteilhaft für den Anwender (u.a. mehr RAM), leider "weiß" IDE11 nichts von solchen Produktmodifikationen. Da man stets auf das "User defined" Target ausweichen kann, stellt das aber kein Problem dar.

Wenn man sich damit vertraut gemacht hat, welche spezifischen Einstellungen für die jeweilige (z.B. eigene) Zielhardware benötigt wird, kann man also in jedem Fall "User defined" benutzen. Der Unterschied besteht lediglich darin, daß man im Gegensatz zu den voreingestellten Target-Optionen die einzelnen Settings des benutzerdefinierten Typs mit dem Button "Detail" nicht nur anschauen, sondern auch nach eigenen Vorstellungen verändern kann. Die Einstellungen des benutzerdefinierten Typen werden beim Verlassen der IDE gespeichert und stehen somit beim nächsten Aufruf der IDE unverändert zur Verfügung.

2.5. Aufrufoptionen

IDE11 wird mit folgender Syntax aufgerufen (Die Parameter in eckigen Klammern sind optional):

```
IDE11 [-Cx] [-In] [file...]
```

Die Parameter haben folgende Bedeutung:

`-Cx`
Verwendet COM-Port x zur Kommunikation mit der Zielhardware, mögliche Werte: x = 1 ... 4, Standard ist x = 1

`-In`
Verwendet Interrupt n für den gewählten COM-Port, mögliche Werte: n = 3, 4, 5 oder 7, Standard ist n = 4 (für COM1/COM3) bzw. n = 3 (für COM2/COM4). Dieser Parameter muß der Konfiguration (vorgenommen per Jumper bzw. BIOS-Setup) der verwendeten Schnittstellenkarte entsprechen.

`file...`
Bis zu zehn Namen von Quelldateien, welche automatisch beim Programmstart geladen werden sollen.

`-H`
Help - zeigt Informationen zu Programmaufruf und Parametern.

`-?`
Wie -H.

3. Arbeiten mit IDE11

3.1. Getting Started

Das folgende Kapitel zeigt Ihnen, wie der grundsätzliche Arbeitsablauf des Programms funktioniert. Es werden nur die wesentlichsten Funktionen genannt, damit Sie schnell in die Lage versetzt werden, eigene Versuche zu unternehmen. Eine detaillierte Beschreibung der einzelnen Menüs und Funktionen finden Sie in den anschließenden Abschnitten dieses Handbuches.

Es wird angenommen, Sie hätten die Software mit den Standardeinstellungen auf dem Laufwerk C: im Verzeichnis \IDE11 installiert. Haben Sie eine andere Einstellung gewählt, so gelten die folgenden Pfadangaben sinngemäß.

Wechseln Sie in das Verzeichnis, in dem Ihre Sourcecodes abgelegt sind (im Falle der Standardinstallation ist dies das Verzeichnis C:\IDE11\SRC):

```
C:
CD \IDE11\SRC
```

Rufen Sie das Programm auf mit:

```
IDE11
```

Startet das Programm nicht, so überprüfen Sie, ob das Verzeichnis C:\IDE11\BIN Bestandteil der PATH-Variable ist. Ergänzen Sie die AUTOEXEC.BAT Datei ggf. entsprechend.

Die Integrierte Entwicklungsumgebung bietet Ihnen zur Befehlseingabe ein Menüsystem, die Statuszeile sowie Hotkeys an. Die Verwendung dieser Elemente ist Ihnen sicherlich vertraut, eine Beschreibung der Funktion von Maus und Tastatur können wir uns an dieser Stelle sparen.

Laden Sie die Beispieldatei BLINK.A über den Menüpunkt File/Open. Geben Sie den Dateinamen ein und bestätigen Sie mit <ENTER>.

Übersetzen Sie den Quelltext mit dem Befehl Run/Assemble. Nach einem kurzen Moment sehen Sie die Meldung, daß der Assembler seine Arbeit erfolgreich abgeschlossen hat. Schließen Sie diese Mitteilung mit <ENTER> oder <ESC>. Sollte jedoch ein Fehler angezeigt werden, so beheben Sie dessen Ursache (Fehlerfenster schließen mit <ALT-F3> oder <ESC>) und übersetzen Sie den Quelltext erneut. Anstelle Run/Assemble im Menü können Sie auch den Hotkey <F9> dazu benutzen.

Das Programm bringen Sie zur Ausführung, indem Sie den Befehl Run/Execute (oder <F8>) verwenden. Vorher müssen Sie jedoch im Menüpunkt Options/Target die von Ihnen eingesetzte Zielhardware auswählen. Das übersetzte S-Record File kann dann in den Speicher der Target-CPU geladen und gestartet werden. An dieser Stelle können Fehlermeldungen vor allem durch das Ausbleiben der Betriebsspannung der Target-CPU oder eine unkorrekte serielle Verbindung zwischen PC und Target auftreten.

Der Einzelschrittbetrieb ist an zwei Voraussetzungen geknüpft: Das Programm muß schon im Speicher des Target stehen und darf nicht den internen RAM-Bereich ($0000-$00FF) belegen (dieser Bereich wird vom Talker, also der Kommunikationsroutine belegt). Wenn das zu testende

Programm in einem externen Speicherbereich abgelegt ist (d.h., wenn der HC11 im Expanded Mode arbeitet) muß vor Ausführung des Einzelschrittbetriebes die Datei OC5VEC.A übersetzt und geladen werden, um den für den Tracemode verwendeten Interruptvektor zu installieren.

Laden Sie Ihr Programm zuerst mit Run/Load in den internen EEPROM. Starten Sie danach den Einzelschrittbetrieb mit Debug/Trace, die Startadresse ist $B600. Nach jedem Programmschritt bekommen Sie den Prozessorstatus, die Belegung der Ports A bis E und die nächste Instruction angezeigt. Mit <ENTER> führen Sie den nächsten Programmschritt aus, <ESC> bricht den Singlestepbetrieb ab.

Weitere Einblicke in das "Innenleben" des HC11 ermöglichen die Befehle "Memory Dump" und "Reassemble" im Menü "Debug". Beide Befehle greifen auf den Target-Speicher zu und stellen die erhaltenen Informationen als Speicherauszug bzw. als Reassemblerlisting dar.

Der Editor gestattet Ihnen, mehrere Quelldateien gleichzeitig zu laden. Die Befehle zur Assemblierung und Programmausführung beziehen sich immer auf das oberste Fenster. Somit können Sie schnell per Mausklick zwischen mehreren Quelltexten wechseln und verschiedene Softwaremodule wechselseitig testen.

Sichern Sie geänderte Quelltexte mit File/Save oder kurz <F2>. Wenn Sie eine geänderte und noch nicht gespeicherte Datei übersetzen lassen, wird der Inhalt des Editorfensters automatisch gesichert. Handelt es sich um eine neue Datei (File/New), müssen Sie diese zuerst benennen. Die Eingabeaufforderung hierzu wird automatisch angezeigt.

Sie beenden das Programm über den Menüpunkt File/Exit oder das Tastenkürzel <ALT-X>. Geänderte Dateien, die noch nicht abgespeichert wurden, werden (nach einer Rückfrage an den Benutzer) gesichert.

3.2. Die zehn häufigsten Programmierfehler

Profis und Anfänger haben eines gemein: Sie machen beim Programmieren Fehler. Der Unterschied: professionell gemachte Fehler sind schwerer zu finden (der Autor muß es wissen, schließlich verdient er seinen Lebensunterhalt u.a. damit, solche kniffeligen Dinge aufzuspüren).

Aus den Gesprächen mit vielen IDE11 Anwendern wurde die folgende Liste mit den häufigsten Fallen extrahiert. Sie erhebt keinen Anspruch auf Vollständigkeit, schließlich ist der Mensch ständig auf der Suche (...nach neuen Fehlern...):

Stackpointer

Der Stackpointer muß unbedingt auf eine geeignete Adresse gesetzt werden. Beliebt ist das Ende des internen RAMs (Vorsicht wegen der Interruptvektoren im Special Bootstrap Mode!). Unterbleibt dies, wird das Programm höchstwahrscheinlich schon nach dem ersten Unterprogramm oder Interrupt abstürzen. Vorsicht Falle: Auch der Talker von IDE11 muß den Stackpointer setzen, um korrekt arbeiten zu können. Versäumt ein geladenes Anwenderprogramm, sich seinerseits um den Stack zu kümmern, wird das Programm u.U. dennoch laufen, da der Talker den Stackpointer an einer brauchbaren Adresse "stehen ließ". Im Standalone-Betrieb (Expanded Mode) kommt es dann "plötzlich" zum Crash.

Reset-Vektor

IDE11 startet ein Programm normalerweise über einen Sprung auf die Anfangsadresse (ausgeführt durch den Talker). Im realen Betrieb (soll heißen: im Expanded Multiplexed Mode) sucht

sich die CPU den Startpunkt über den Resetvektor bei $FFFE/FFFF. Sind Sie sicher, den Resetvektor in Ihren Programmcode eingebaut zu haben?

CONFIG-Register

Wenn der HC11 unerwartet reagiert (speziell, wenn es mit einem anderen Exemplar schon einmal funktionierte), sollte das CONFIG-Register überprüft werden. Dieses EEPROM-basierte Register hat teilweise bei den einzelnen HC11-Derivaten unterschiedliche Bedeutung, ein tiefer Blick in das passende "Technical Data" sollte weiterhelfen. Ein gängiger Wert (zumindest für 68HC11Ax/Ex) ist $0D.

Interruptvektoren

Im Special Mode liegen die Interruptvektoren an anderen Adressen als im Expanded Mode. Zudem werden für die Pseudovektoren im RAM drei Byte belegt - vor der eigentlichen Vektoradresse steht schließlich noch der JMP-Opcode. Über das Wirkprinzip dieser Vektoren, abhängig von der jeweiligen Betriebsart, sollte man sich gut belesen.

Pull-Up am SCI

Pull-Up Widerstände am TxD-Pin (und ggf. am RxD-Pin) haben oft schon Wunder bewirkt. Schließlich schaltet der Bootlader des HC11 Port D (und somit auch TxD) in den Wired-Or-Mode (vergleichbar mit einem "Open-Collector" Ausgang).

Time Protected Register

Bestimmte Steuerregister bzw. einzelne darin enthaltene Bits lassen sich - im Expanded Mode - nur während der ersten 64 Taktzyklen nach Reset ändern. Diese Register sollte man also stets gleich zu Beginn des Programms initialisieren und zum Test das Target wirklich im Expanded Mode starten (manuell oder mittels der /RTS Option von IDE11).

Stromversorgung

Man glaubt es einfach nicht, wieviele Probleme durch eine "wackelige" Stromversorgung ausgelöst werden. Messen Sie doch mal nach, ob wirklich 5V beim HC11 ankommen. Einstellbare Netzteile sollte man übrigens meiden, da man sie nicht nur einstellen, sondern auch *verstellen* kann! Außerdem produzieren manche von Ihnen äußerst ungesunde Spitzen beim Ein- bzw. Ausschalten...

Resetcontroller

Eine RC-Kombination am Reseteingang *tut es nicht*! Hier sollte stets ein Resetcontroller (MC34064, TL7705...) seinen Dienst tun.

EEPROM-Autostart

Immer wieder taucht die Frage auf: Wie kann mein Programm im internen EEPROM selbständig nach Reset starten? Die Antwort ist einfach (siehe Technical Data bzw. Motorolas Application Note AN1060 "MC68HC11 Bootstrap Mode"): Verbinden Sie TxD mit RxD - und den Pull-Up nicht vergessen. Dann schickt sich der HC11 selbst einen L-Impuls, was der Bootlader als

Sprung zum EEPROM umsetzt. Einfach RxD auf L-Pegel zulegen geht übrigens nicht, es muß schon ein Impuls sein.

Watchdog-Aktivierung

Der Watchdog wird durch das NOCOP Bit im CONFIG-Register gesteuert. Sie haben das NOCOP Bit gelöscht und es funktioniert rotzdem nicht? Dann läuft Ihr Programm wahrscheinlich im Special Bootstrap Mode, und Sie haben vergessen, das DISR Bit im TEST1 Register zurückzusetzen - dieses stellt nämlich eine zusätzliche Sperre für den Watchdog in besagtem Mode dar.

4. Menüreferenz

Um einen Menüpunkt auszuwählen, gibt man den hervorgehobenen Buchstaben des Menüpunkts bei gleichzeitigem Halten der <ALT>-Taste ein. Alternativ erreicht man das Menü auch mit <F10>; der Wechsel zwischen den Menüpunkten ist mit den Cursortasten möglich. Der gebräuchlichste Weg zur Auswahl von Menüpunkten ist jedoch ein Mausklick auf den gewünschten Befehl in der Menüleiste oder aber auf der Statuszeile (unterste Bildschirmzeile).

4.1. System

4.1.1 About

Zeigt eine Information zur Versionsnummer und den Copyright-Vermerk an. Außerdem wird die Seriennummer (bzw. Benutzer-ID) ausgegeben, bei nichtlizensierten Programmkopien erscheint hier "Unregistered Evaluation Copy".

Die About-Dialogbox erscheint automatisch bei jedem Aufruf des Programms, es sei denn, IDE11 wird mit Parametern aufgerufen.

4.2. File

4.2.1. Open

Hotkey: <F3>

Im angezeigten Dateidialog kann eine vorhandene Quelltext-Datei ausgewählt werden, um sie im Editor zu bearbeiten. Die Dateinamens-Erweiterung ist beliebig, es empfiehlt sich jedoch, an der Vorgabe "*.A" festzuhalten.

Im unteren Bereich des Dialogfensters werden Datei-Informationen (Pfadangabe, Dateilänge, Erstellungszeit) angezeigt. <ESC> bricht den Dialog ab. Es ist selbstverständlich möglich, gleichzeitig mehrere Editorfenster zu öffnen. Die Funktionen des Menüpunktes "Run" beziehen sich standardmäßig immer auf das zuoberst liegende (aktive) Editorfenster.

4.2.2. New

Erzeugt ein leeres Editorfenster mit dem Bezeichnung "Untitled". Beim ersten Speichern muß der Datei einen gültiger Dateiname zugeordnet werden.

4.2.3. Save

Hotkey: <F2>

Speichert den Inhalt des aktiven Editorfensters als Datei. Ist der Dateiname noch nicht bekannt, wird stattdessen ein "Save as" ausgeführt (siehe unten). Beim Aufruf des Assemblers wird die Quelltextdatei vorher automatisch gespeichert.

4.2.4. Save as

Speichert den Inhalt des aktiven Editorfensters als Datei und erfragt zuvor den Dateinamen. Eine Quelltextdatei für den integrierten Assembler sollte stets die Erweiterung *.A erhalten.

4.2.5. Close

Hotkey: <ALT-F3>

Schließt das zuoberst liegende Fenster (z.B. Editor-, Fehler- oder Terminalfenster). Ist der Inhalt eines Editorfensters noch nicht gespeichert, wird der Benutzer gefragt, ob der Text noch gespeichert werden soll.

4.2.6. Change dir

Wechselt das Arbeitsverzeichnis und/oder das aktuelle Laufwerk.

4.2.7. DOS shell

Lädt den Kommandointerpreter COMMAND.COM und geht damit in den DOS-Modus. Mit dem Befehl "EXIT" <ENTER> erfolgt die Rückkehr zur IDE. Die Position des

DOS-Kommandointerpreters wird über die Environment-Variable COMSPEC ermittelt. Überprüfen Sie bei Problemen ggf. diesen Eintrag (DOS-Befehl SET zeigt alle Environment-Einträge an).

Es kann nicht ausgeschlossen werden, daß verschiedene Programme die Funktionsweise der IDE beeinträchtigen können, wenn sie von diesem DOS-Shell Befehl aus aufgerufen werden. Es sollten insbesondere keine Programme aufgerufen werden, die Einfluß auf die serielle Schnittstelle nehmen.

Da IDE11 vor Aufruf des Kommandointerpreters die eigenen Programmbestandteile weitestgehend aus dem Hauptspeicher in den Extended Memory (XMS) auslagert, steht in der DOS-Shell ein Maximum freien Arbeitsspeichers zur Verfügung. Ist kein XMS-Speicher verfügbar, nutzt IDE11 Festplattenplatz zum Swappen. Diese Vorteile stehen sowohl für die DOS-Shell als auch bei Aufruf der externen Komponenten EPROM-Emulator und Software-Simulator zur Verfügung.

4.2.8. Exit

Hotkey: <ALT-X>

Beendet das Programm. Sollten noch ungesicherte Editorfenster offen sein, erfolgt eine Nachfrage, ob diese vor dem Schließen gespeichert werden sollen. Zur Auswahl stehen die Antworten YES, NO und CANCEL, wobei CANCEL den Exit-Befehl abbricht, d.h. man bleibt im Programm.

4.3. Edit

Die Befehle im Menü Edit beziehen sich stets auf das sichtbare (zuoberst liegende) Editorfenster (bzw. die geöffnete Zwischenablage).

4.3.1. Undo

Nimmt alle Änderungen seit der letzten Cursorbewegung zurück.

4.3.2. Cut

Hotkey: <SHIFT-DEL>

Entfernt einen zuvor markierten Abschnitt aus dem Text und transferiert ihn in die Zwischenablage. Von dort aus kann er mit Paste in einen anderen Quelltext oder an eine andere Stelle im gleichen Text übernommen werden.

Sie können einen Bereich auf einfache Art entweder durch Ziehen mit der Maus oder durch Betätigen der Cursortasten bei gedrückter <SHIFT>-Taste markieren.

4.3.3. Copy

Hotkey: <CTRL-INS>

Überträgt den zuvor markierten Bereich in die Zwischenablage, der Bereich bleibt jedoch im Textfenster erhalten. Mit Paste läßt sich der kopierte Bereich an einer anderen Stelle / einer anderen Datei einfügen.

4.3.4. Paste

Hotkey: <SHIFT-INS>

Fügt den zuvor in die Zwischenablage transferierten Textabschnitt an der aktuellen Cursorposition ein. Es ist auch möglich, den Inhalt der Zwischenablage manuell anzupassen. Mit Paste wird immer der in der Zwischenablage markierte Text eingefügt. Diese Markierung läßt sich wie in jedem Editorfenster durch Ziehen der Maus festlegen.

4.3.5. Show clipboard

Zeigt den Inhalt der Zwischenablage in einem Editorfenster an. Das Fenster trägt die Bezeichnung "Clipboard". Sie können den Inhalt der Zwischenablage auch speichern, müssen aber die resultierende Datei erst benennen.

4.3.6. Clear

Hotkey: <CTRL-DEL>

Löscht den zuvor markierten Textabschnitt. Das kann zwar mit dem Befehl Undo unmittelbar danach rückgängig gemacht werden, der Abschnitt wird jedoch nicht in die Zwischenablage kopiert.

4.4. Search

4.4.1. Find

Hotkey: <CTRL-Q> <F>

Sucht einen Text, ausgehend von der aktuellen Cursorposition. Die Auswahl "Case sensitive" ermöglicht die Unterscheidung von Groß- und Kleinschreibung. Die Option "Whole words only" ist zu wählen, wenn nur nach vollständigen Worten gesucht werden soll, und nicht nach Silben bzw. Bruchstücken. Um den Text von Beginn an zu durchsuchen, muß der Cursor zuvor an den Textanfang gesetzt werden.

4.4.2. Replace

Hotkey: <CTRL-Q> <A>

Replace ersetzt Textstücke durch einen alternative Ausdruck. Die Funktionsweise ist dieselbe, wie bei Find beschrieben. Dazu kommt die Möglichkeit, die Sicherheits-Nachfrage beim Ersetzen jeder gefundenen Passage abzuschalten (Prompt on replace nicht angekreuzt), sowie nicht nur den ersten, sondern alle gefundenen Übereinstimmungen ersetzen zu lassen (Replace all).

4.4.3. Search again

Hotkey: <CTRL-L>

Wiederholt die zuletzt durchgeführte Find oder Replace Operation, ausgehend von der aktuellen Cursorposition.

4.5. Run

Die meisten der unter dem Menüpunkt "Run" zusammengefaßten Funktionen benötigen als Input eine Assemblerdatei (Quelltext) bzw. eine hieraus erzeugte Objektdatei (Executable). Es wird stets auf das aktuelle (sichtbare) Editorfenster Bezug genommen. Ist kein solches Fenster vorhanden, wird ein Dateidialog angezeigt, mit dessen Hilfe ein Inputfile ausgewählt werden kann. Dadurch ist es einerseits möglich, Quelltexte zu übersetzen, ohne sie vorher in ein Editorfenster zu laden, andererseits können auch fremderzeugte S-Record Executables (zu denen kein passender Quelltext vorliegt) in das Target geladen werden.

4.5.1. Assemble

Hotkey: <F9>

Übersetzt die Datei im aktuellen (sichtbaren) Editorfenster. Zuerst wird überprüft, ob der Editorinhalt verändert wurde. Trifft dies zu, wird die Datei automatisch (!) gespeichert. Unbenannten Editorfenstern muß dabei erst ein Dateiname zugeordet werden. Anschließend wird die Quelldatei (Endung standardmäßig *.A) übersetzt und ein Objektfile gleichen Namens, jedoch mit der Endung *.S19 erzeugt. Existiert bereits ein aktueller Output (das ist der Fall, wenn die Quelldatei älter ist als die Objektdatei), tritt der Assembler nicht in Aktion. Stattdessen erscheint die Meldung "Executable is up to date". Include-Dateien werden nicht auf Veränderungen überprüft. Wurde eine Include-Datei geändert, nicht jedoch die Haupt-Quelldatei, so wird der Assembler die Übersetzung nicht neu ausführen, da er von der Änderung nichts bemerkt. In einem solchen Fall kann die Übersetzung mit dem Befehl "Rebuild" erzwungen werden.

Kommt es zu Fehlermeldungen des Assemblers, so werden die Fehlermeldungen in einem eigenen Fenster angezeigt. Jede Zeile bezeichnet einen Fehler durch Angabe der Quelldatei, der Zeilennummer und einer (textuellen) Fehlermeldung. Ein Eintrag in der Fehlerliste könnte z.B. wie folgt aussehen:

```
c:\ide11\src\blink.a L20 E Symbol (identifier) not found
```

Die Zeile sagt aus, daß in der Datei blink.a in Zeile (Line) 20 ein Fehler aufgetreten ist (Symbol / Bezeichner nicht gefunden). Der Buchstabe "E" steht für Error, also Fehler. Weitere Fehlerklassen sind "F" für fatale (schwerwiegende) Fehler und "W" für Warnungen. Das Auftreten von Warnungen schließt nicht aus, daß der Quelltext einwandfrei übersetzt werden konnte. Warnungen weisen auf Unstimmigkeiten und potentielle Fehlerquellen hin, die üblicherweise durch einen "guten Programmierstil" vermieden werden können.

Innerhalb der Fehlerliste ist eine Auswahl bzw. Bewegung mit den Cursortasten möglich. Die Beseitigung von Fehlern im Quelltext ist sehr einfach. Nach Auswahl des interessierenden Eintrags in der Fehlerliste gelangt man durch Drücken von <ENTER> direkt zur fehlerverursachenden Stelle im Quelltext. Ist die betroffene Datei gerade nicht geladen, öffnet die IDE ein neues Editorfenster mit dieser Datei.

<ESC> schließt das Fehlerfenster. Das Fehlerfenster wird auch beim nächsten Assemble-Befehl automatisch geschlossen.

Der Assembler erzeugt (bei erfolgreicher Abarbeitung) eine Objekt-Datei im Motorola S-Record Format. Einzelheiten zum Aufbau dieses Formats sind im Anhang aufgeführt.

4.5.2. Rebuild

Hotkey: <ALT-F9>

Rebuild funktioniert wie Assemble, es wird aber in jedem Falle der Assembler aufgerufen, auch wenn bereits ein aktuelles Executable existiert.

4.5.3. Load

Hotkey: <CTRL-F8>

"Load" prüft, ob das Objectfile up-to-date ist. Wenn nicht, so wird zuerst automatisch der Befehl "Assemble" ausgeführt. Anschließend wird das Objectfile in den Speicher des Target übertragen, das geladene Programm wird jedoch nicht gestartet. Ausnahme: Downloads in den internen RAM des HC11 sind nur verbunden mit einem Autostart des geladenen Programms möglich (bedingt durch die HC11-Firmware).

Wenn kein Editorfenster geöffnet ist (bzw. ein anderes Fenster "obenauf" liegt), wird ein Dateidialog angezeigt, um eine vorhandene S-Record Datei auszuwählen. Somit ist es möglich, beliebige Objektdateien im Motorola S-Record Format zu laden, selbst wenn dazu nicht der Quelltext vorliegt.

4.5.4. Execute

Hotkey: <F8>

Lädt das Objectfile (siehe "Load") und startet das Programm. Eine Meldung informiert über die Programm-Startadresse und den gewählten Betriebs-Mode des HC11. Ggf. erfolgt hierbei automatisch eine Umschaltung vom Special Bootstrap in den Expanded Mode. Dies kann entweder per Softwarebefehl (Modifikation des HPRIO Registers durch den Talker, Standardverhalten) oder durch Umschaltung des Pegels der beiden Mode-Pins des HC11 (Mode-Control im Menü "Options/Target/Detail" aktiviert) mit nachfolgendem Reset erfolgen.

4.5.5. EPROM Emulator

Es ist möglich, ein Programm an einen EPROM Emulator zu übergeben. IDE11 geht davon aus, daß der EPROM Emulator den Adreßbereich $8000 bis $FFFF abdeckt (32 KB). Zuerst wird aus der S-Record Datei ein Binärfile generiert. Dieses Binärfile trägt stets den Namen "EPSIM.ROM", es wird im aktuellen Arbeitsverzeichnis angelegt. EPROM Emulatoren werden i.d.R. mit einem Utilityprogramm zum Download von Binärdaten ausgeliefert. Zur Einbindung eines solchen Utilityprogramms in das IDE11 System muß die Datei IDE11_E!.BAT (im Verzeichnis \IDE11\BIN) angepaßt werden. Die mitgelieferte Batchdatei verwendet den EPROM Emulator EPSIM/1 von MCT und hat folgendes Aussehen:

```
@echo off
rem ------------------------------------------------
rem Usage: IDE11_E!.BAT starting_address download_file
rem ------------------------------------------------
cls
if not exist %2 goto err1
```

```
rem ------------------------------------------------
rem -d1    = Device (LPT) 1
rem -aXXXX = Offset in download_file (4 hex numbers)
rem ------------------------------------------------
epsim -d1 -a%1 %2
goto ende
:err1
echo EPSIM: File %2 not found! Abort.
pause
:ende
```

Es ist ersichtlich, daß zwei Parameter übergeben werden: die Startadresse und der Dateiname der Binärdatei. Die Startadresse kennzeichnet den Offset zur Anfangsadresse ($8000) des 32 KB EPROM-Bereiches. Liegt beispielsweise die absolute Anfangsadresse eines Anwenderprogramms bei $E000, lautet der Wert für die übergebene Startadresse $6000. IDE11 ruft die Batchdatei also wie folgt auf:

```
IDE11_E! 6000 EPSIM.ROM
```

Hieraus erzeugt die Batchdatei folgenden Aufruf für EPSIM/1:

```
EPSIM -d1 -a6000 EPSIM.ROM
```

Mit den gegebenen Informationen sollte es nicht schwerfallen, andere EPROM Emulatoren anstelle des EPSIM/1 einzubinden.

4.5.6. Simulate

Dieser Menüpunkt wird nur angezeigt, wenn IDE11 mit dem Kommandozeilenparameter "-s" gestartet wurde (siehe hierzu auch Handbuchabschnitt Aufrufoptionen)!

Die Funktion stellt eine Schnittstelle zum HC11 Softwaresimularor TESTE68 her (dieser Simulator ist nicht Bestandteil des Lieferumfangs von IDE11!). Voraussetzung für die reibungslose Funktion ist die Ausgabe einer Symboldatei während der Assemblierung, was durch Ankreuzen der Option "Write symbols" im Menü "Options/Assembler" ermöglicht wird.

IDE11 übergibt die erzeugte S-Record Datei und die Symbol- (Map-) Datei an TESTE68 über die Batchdatei IDE11_S!.BAT. Es gibt keinen Grund, diese Batchdatei zu verändern. TESTE68 sollte im DOS-Suchpfad enthalten sein. Nach Beendigung von TESTE68 erfolgt die Rückkehr zu IDE11. In der vorliegenden Version von IDE11 sollte es keine Speicherplatzprobleme bei Ausführung des Simulators mehr geben, da IDE11 vor Ausführung des Simulators in virtuellen Speicher (XMS bzw. Disk) ausgelagert wird (siehe Erläuterungen zum Menüpunkt "File/Dos shell").

4.5.7. Reset

Hotkey: <ALT-F4>

Erzeugt einen Reset-Impuls, je nach gewählten Targetoptionen als High- oder Low-Impuls. Voraussetzung dafür ist, daß eine Verbindung zwischen PC und Resetleitung des Target besteht

(Reset control *nicht* manual) , sonst führt die Funktion nur zur Aufforderung, den Resetimpuls manuell auszulösen - aber was nützen letztlich derlei Selbstgespräche...

4.5.8. Restart RAM

Schaltet in den Special Bootstrap Mode und startet ein Programm, welches sich bereits im internen RAM des HC11 befindet. Hierzu wird erst ein Resetimpuls ausgelöst und dann wird der Bootlader des HC11 verwendet, um zu Adresse $0000 zu springen (das genaue Verfahren hängt vom eingesetzten HC11 Typ ab).

4.5.9. Restart EEPROM

Schaltet in den Special Bootstrap Mode und startet ein Programm, welches sich bereits im internen EEPROM des HC11 befinden muß. Hierzu wird erst ein Resetimpuls ausgelöst und dann wird der Bootlader des HC11 verwendet, um zur Anfangsadresse des internen EEPROM zu springen. Technisch geschieht das durch Senden eines $00-Bytes an den Bootlader (funktionsgleich mit dem Verbinden von TxD- und RxD-Leitung des HC11).

4.5.10. Goto

Hotkey: <ALT-F8>

Startet ein bereits geladenes Programm auf einer vom Benutzer einzugebenden Adresse. Der Befehl wird durch den IDE11 Talker ausgeführt. Der Programmstart erfolgt wie bei Execute, nur entfällt das Assemblieren und Laden.

4.6. Debug

Die unter "Debug" zusammengefaßten Funktionen ermöglichen Einblicke in das "Innenleben" der Targethardware. Zur Kommunikation transferiert die IDE kurze Sytemroutinen (Talker) in den internen RAM des HC11. Diese Vorgehensweise ermöglicht eine weitgehende Unabhängigkeit vom konkreten Zielsystem, der Anwender sollte jedoch die Belegung des internen RAM durch den Talker beim Debugging nicht außer acht lassen.

4.6.1. Trace

Hotkey: <F7>

Startet den Einzelschrittbetrieb, beginnend bei der eingegebenen Startadresse. Es wird davon ausgegangen, daß sich das auszuführende Programm bereits im Speicher des Target befindet.

Zudem muß der Interruptvektor für Output Compare Kanal 5 (OC5) korrekt gesetzt sein. Im Special Bootstrap Mode ist das automatisch der Fall, im Expanded Mode hingegen muß der Anwender den Vektor erst in das Target laden. Dies geschieht durch Übersetzen und Laden der Datei OC5VEC.A.

Der interne RAM des HC11 scheidet als Zielspeicher aus, da er weitestgehend von der Singlestep-Routine belegt ist. Einige RAM-Bereiche sind dennoch als Platz für Benutzervariablen nutzbar, Einzelheiten der Speicherbelegung sind im Handbuchabschnitt zum Singelstep-Betrieb zu finden.

Im Trace-Fenster wird angezeigt:

- die Akkus ACCA und ACCB
- die Indexregister IX und IY
- den Programcounter PC
- den Stackpointer SP
- das Flagregister CCR mit den einzelnen Bitpositionen
- die Belegung der Ports A-E
- die aktuelle Instruction in Assembler-Notation

Der Einzelschrittbetrieb ist keine Simulation, sondern stellt Ihnen die realen Prozessorinformationen zur Verfügung. Die Portbelegung stellt die Verhältnisse dar, die der HC11 aus den Portregistern ausliest. Probleme gibt es bei der Abarbeitung interrupt-orientierter Software sowie einiger spezieller Prozessorbefehle (Einzelheiten dazu im Handbuchabschnitt zum Singlestep-Debugger). Bei Verletzung dieser Regeln bricht der Einzelschrittbetrieb mit einer Fehlermeldung ab, da die Kommunikation zwischen PC und Target-CPU nicht mehr möglich ist.

Mit <ENTER> wird der nächste Programmschritt ausgeführt, mit <ESC> wird die Funktion beendet.

4.6.2. Memory Dump

Zeigt den Inhalt des Target-Speichers in Hex- und ASCII-Darstellung an. Vorab wird die Eingabe der Startadresse verlangt. Wird die Option "Write memory dump to file" angekreuzt, wird eine Binärdatei mit dem angegebenen Namen ("Filename") erzeugt, die den Speicherauszug ab Startadresse bis inkl. Endadresse ("Last Address") aufnimmt.

Bei der Anzeige im Memory-Dump Fenster kann eine neue Startadresse durch Doppelklick mit der Maus innerhalb des Fensters, oder einfach durch Betätigen der <ENTER>-Taste vorgegeben werden. Weiterhin steht der Rollbalken am rechten Rand des Fensters zur Steuerung der Anzeigeposition zur Verfügung.

Zur Anzeige des Target-Speichers muß der anzuzeigende Bereich vom Target zur IDE transferiert werden. Da die Übertragung des gesamten 64 KB Adreßraumes zu lange dauern würde, wird nur jeweils der unmittelbar anzuzeigende Bereich transferiert. Die Daten werden jedoch durch die IDE gepuffert. Dadurch wird vermieden, Speicherbereiche, auf die bereits zugegriffen wurde, erneut zu laden. Ist ein erneutes Laden dennoch erwünscht (z.B. wenn sich inzwischen der Speicherinhalt durch Ausführung eines Anwenderprogramms oder durch Ausschalten des Target geändert hat), ist das Memory Dump Fenster zu schließen. Beim erneuten Öffnen des Fensters werden alle Puffer der IDE, welche Informationen zum Target-Speicher enthalten, gelöscht und alle Speicherinformationen aktualisiert.

4.6.3. Reassemble

Stellt den Inhalt des Target-Speichers reassembliert (d.h. in Quelltextform) dar. Vorab muß die Startadresse eingegeben werden. Auch hier ist es möglich, den Output in eine Datei umzuleiten (siehe Memory Dump), allerdings ist es hier keine Binärdatei, sondern eine Textdatei mit Formatierungen wie im Reassembler-Fenster. Die Darstellung jedes Befehls (d.h. jeder Zeile) ist gegliedert in Befehlsadresse, bis zu fünf Bytes, die den Befehl bilden, Mnemonic und schließlich eventuell vorhandene Operanden.

Eine neue Startadresse kann durch Doppelklick mit der Maus innerhalb des Fensters, oder einfach durch Betätigen der <ENTER>-Taste vorgegeben werden. Weiterhin steht der Rollbalken

am rechten Rand des Fensters zur Steuerung der Anzeigeposition zur Verfügung. Bezüglich des Zugriffs auf den Target-Speicher gelten die Ausführungen im vorangegangenen Abschnitt.

Anmerkung zum Reassembler: Grundsätzlich ist zu beachten, daß beim "Zurückblättern" durch einen Reassembler nie eindeutige Ergebnisse erzielt werden können. Ausgehend von einer bestimmten Startadresse ist (vorwärts) stets klar, wie lang ein Befehl ist und wo der nächste Befehl beginnt. Rückwärts existieren jedoch Mehrdeutigkeiten, da die Länge des vorhergehenden Befehls nicht eindeutig ermittelbar ist.

4.7. Utilities

4.7.1. Terminal

Hotkey: <F5>

Öffnet das Terminalfenster. Dieses Fenster zeigt alle vom HC11 zum PC gesendeten Zeichen an. Der Zeichenempfang ist gepuffert. Auch wenn das Terminalfenster nicht aktiv ist, werden alle eingehenden Zeichen an das Terminal übergeben. Somit erhält man - bei Aktivierung - stets eine aktuelle Anzeige, unabhängig von der Sichtbarkeit des Fensters. Nicht angezeigt werden Bytes, die der Kommunikation der IDE11 mit dem Target dienen, also z.B. Downloadsequenzen.

Tastatureingaben des Benutzers werden vom Terminal seriell zum HC11 gesendet. Das gilt selbstverständlich nur, wenn das Terminalfenster aktiv ist.

Das Übertragungsformat ist, sende- wie empfangsseitig, "8N1", das bedeutet 8 Datenbit, keine Parität, 1 Stoppbit. Die Baudrate wird PC-seitig im Menü "Options/Terminal" eingestellt. Auf

der HC11-Seite verhält es sich wie folgt: Im Special Bootstrap Mode arbeitet das SCI des HC11 mit 7812 Baud (8 MHz Quarzfrequenz als Voraussetzung). Im Expanded Mode müssen Sie das SCI Ihren Wünschen entsprechend initialisieren. Üblich sind 9600 Baud.

Das Terminalfenster kann mittels <ALT-F3> oder mit einem Mausklick auf das Schließsymbol (oben links im Rahmen des Fensters) geschlossen werden.

Ist die Option "Activate/automatic" im Menü "Options/Terminal" angekreuzt, wird das Terminalfenster automatisch nach jedem Befehl, der ein Programm auf dem HC11 startet, aktiviert. Zwischen Ende des Downloads und Aktivierung des Terminals benötigt der PC einen Augenblick, währenddessen die ersten vom HC11 innerhalb des Anwenderprogramms gesendeten Zeichen verloren gehen können. Um Zeichenverlust zu vermeiden, empfiehlt es sich, an den Anfang des Anwenderprogramms eine Verzögerung einzubauen, wie z.B. in diesem Programmstück:

```
        ORG $B600

        ldx #0
_delay  dex
        bne _delay
        ...
```

4.7.2. Config reg

Mit diesem Menüpunkt kann der Inhalt des EEPROM-basierten CONFIG-Registers der MCU abgerufen und modifiziert (programmiert) werden.

Zuerst wird der aktuelle Inhalt des CONFIG-Registers ausgelesen und (in hexadezimaler Form) angezeigt. Entscheidet sich der Anwender, das Register zu modifizieren, wird das Register bitweise aufgeschlüsselt dargestellt. Die neuen Werte für die einzelnen Registerbits können nun angekreuzt werden. An dieser Stelle ist immer noch ein Abbruch (Cancel) möglich. Wird auch die folgende Rückfrage bestätigt, programmiert IDE11 das CONFIG-Register neu.

Vor Änderung des CONFIG-Registers sollten Sie sich genau mit der Bedeutung der einzelnen Flags vertraut machen. Informationen zu diesem Punkt finden Sie im Datenbuch der verwendeten MCU, teilweise weicht die Implementierung des CONFIG-Registers bei den einzelnen HC11 Typen voneinander ab - im Zweifelsfalle sollten Sie das CONFIG-Register unverändert lassen!

Eine Anmerkung zum **68HC11F1 und 68HC811E2** : Die Bits des CONFIG-Registers dieses HC11-Typs "funktionieren" teilweise in den einzelnen Betriebsmodi unterschiedlich. Ein CONFIG-Wert, ausgelesen im Special Bootstrap Mode, kann sich im Expanded Mode ganz anders darstellen. Leider gibt es keine universelle und stets anwendbare Möglichkeit, den im Expanded Mode effektiven Wert auszulesen. IDE11 verwendet daher stets nur den Special Bootstrap Mode zum Auslesen des CONFIG-Registers. Sollte ein Programm auf dem ´F1 oder ´E2 mit IDE11 (bei Benutzung der softwaremäßigen Mode-Umschaltung) funktionieren und "Standalone" im Expanded Mode nicht mehr, dann lohnt der Versuch, das CONFIG-Register "blind" zu reprogrammieren (die ggf. angezeigte Fehlermeldung beim Verifizieren kann man aus o.a. Gründen ignorieren).

4.8. Options

4.8.1. Target

Dieser Menüpunkt dient zur Auswahl des angeschlossenen HC11-Target. Es erscheint eine Auswahlliste mit den zur Verfügung stehenden Zielsystemen, welche die Software "kennt". Neben verschiedenen vordefinierten Produkten gibt es einen benutzerdefinierten Typ.

Mit dem Schalter "Details" lassen sich die einzelnen Einstellungen kontrollieren bzw. (nur beim Typ "User defined") ändern.

Mit der Einstellung "MCU type" bestimmen Sie den verwendeten Mikrocontroller aus der HC11-Familie. Die Typen 68HC11A8/A1/A0 unterscheiden sich nur im Wert des CONFIG-Registers und (dadurch) im Vorhandensein interner ROM- und EEPROM-Bereiche. Gleiches gilt für die Reihe 68HC11E9/E1/E0. Die Dialogbox zeigt den ausgewählten CONFIG-Wert an. Mit dem Button "Verify" kann der tatsächliche Inhalt des CONFIG-Registers überprüft und ggf. verändert werden. Beim Verändern des CONFIG-Registers ist Vorsicht geboten (vergleiche hierzu Menüpunkt "Utilities/Config reg").

Unter "External Mem" sind zwei Auswahlfelder vorhanden, die die externen Speicherbereiche von $0000 bis $7FFF und von $8000 bis $FFFF definieren. In beiden Bereichen stehen jeweils "None" (kein Speicher), "RAM" und "EEPROM" zur Auswahl.

Wichtig: Ein Target wird vor dem Start eines Programms in den Expanded Mode versetzt, es sei denn, beide der o.g. Auswahlfelder stehen auf "None" (kein RAM oder EEPROM

angeschlossen). Die bei "External mem" ausgewählten Einträge entscheiden also über den Betriebsmode, den IDE11 für das Target zur Programmausführung wählt.

Wichtig für eine effiziente Arbeit mit der Software ist eine Möglichkeit, das Target vom PC aus (automatisch) rücksetzen zu können. Die Auslösung eines Reset kann durch einen L-Impuls oder einen H-Impuls über die Leitung /DTR des seriellen Schnittstelle des PC erfolgen. Im Feld "Reset ctrl" läßt sich das erforderliche Verfahren einstellen. Gibt es keine Möglichkeit, solch ein Remote-Reset auszulösen, wählen Sie "Manual" aus. Mit dieser Einstellung werden Sie bei Bedarf von der Software aufgefordert, per Hand ein Reset am Target auszulösen (Reset-Taste).

Die Einstellung im Feld "Mode ctrl" dient der Festlegung der HC11 Betriebsmode Steuerung. Die Standardeinstellung ist "None". Hierbei wird davon ausgegangen, daß das Target permanent auf Special Bootstrap Mode eingestellt ist (MODA=MODB=L). Die Umschaltung in den Expanded Mode erfolgt - bei Bedarf - per Software (Setzen des HPRIO Registers). Wenn Sie in diesem Feld jedoch "/RTS" oder "Manual" wählen, so wird eine echte Mode-Umschaltung vorgenommen. Das geschieht entweder automatisch über die /RTS-Leitung des PC oder manuell durch den Benutzer. Der Vorteil dieser echten Modeumschaltung ist die bessere Praxisnähe bei Abarbeitung des Programms. Der Programmstart erfolgt hier in der Entwicklungsphase praktisch genau wie in der Einsatzphase.

Zur Information über die Belegung des Speicheradreßraums kann über den Button "Mem-Map" ein Schema abgerufen werden. Es zeigt die Belegung des Speichers aus Sicht des HC11 in Schritten von 128 Byte. Die Bedeutung der Symbole ist in dem Schema mit angegeben.

4.8.2. Assembler

Unter "Files" kann die Erzeugung verschiedener Ausgabedateien des Assemblers beeinflußt werden. Bei Aktivierung von "Create Listing *.LST" wird ein Assemblerlisting generiert. Diese Datei mit der Endung "LST" kann u.U. recht groß werden. Aus diesem Grund, sowie zur Beschleunigung des Assemblerlaufs, sollte man diese Option i.Allg. abschalten. Eine weitere Auswahlmöglichkeit in dieser Gruppe ist "Write symbols *.MAP". Ist diese Option eingeschaltet, wird eine Symboldatei mit der Endung "MAP" angelegt. Sie enthält alle Symbole (Labels, Konstanten etc.) des Quelltextes und die zugehörigen numerischen Werte. Die Symboldatei ist eine reine Textdatei, der Aufbau ist im Anhang beschrieben.

Ist (bei "Behaviour") die Option "Ignore case" angekreuzt, ignoriert der Assembler, entgegen dem Standardverhalten, Groß- bzw. Kleinschreibung in Marken (Labels, Bezeichner). Strings sind davon selbstverständlich nicht betroffen. Für Assemblerdirektiven und Mnemonics spielt die Schreibweise ohnehin keine Rolle.

Unter "Include directory" kann ein Verzeichnis angegeben werden, in dem der Assembler nach Include-Dateien (Syntax: INCLUDE <file.ext>) suchen soll. Die Pfadangabe kann Laufwerksbezeichner enthalten und kann relativ sein. Sie sollte mit einem Backslash ("\") abgeschlossen werden. Einige gültige Beispiele:

```
\IDE11\LIB\
C:\INCLUDE\
F:\DEV\HC11\INC\
..\LIB\
```

4.8.3. Terminal

Der Schalter "Logfile" steuert die Erzeugung eines "Mitschnitts" der Terminalausgaben. Alle im Terminalfenster ausgegebenen Zeichen werden in eine Datei namens IDE11.LOG kopiert. Durch Aktivieren des Schalters wird diese Datei erzeugt, beim Deaktivieren wird sie wieder geschlossen. Jedes erneute Aktivieren löscht den bisherigen Inhalt der Datei. Die Datei wird im aktuellen Arbeitsverzeichnis angelegt.

Der Schalter "Activate" steuert das automatische Öffnen des Terminalfensters bei jedem "Execute" Befehl. Dadurch lassen sich Programme leichter beobachten, die kurz nach Programmstart bereits Ausgaben erzeugen.

In der Liste "User baudrate" kann man aus mehreren Möglichkeiten die gewünschte Schnittstellengeschwindigkeit auswählen. Diese Baudrate bezieht sich auf die vom Benutzer im (HC11-) Programm eingestellte Baudrate. Der Standardwert ist 7812 Baud. Dies ist auch die Baudrate, die IDE11 und HC11 *stets* verwenden, um Informationen auszutauschen. Diese (interne) Kommunikation wird durch die Einstellung der "User baudrate" *nicht* beeinflußt.

Der Wert für "Break timeout" kann auf einen höheren Wert gesetzt werden, wenn auf dem Target ein Resetcontroller zum Einsatz kommen, der das Resetsignal (vom PC kommend) zusätzlich verzögert. Bei Einsatz eines einfachen MC34064 Resetcontrollers reicht der Standardwert von 50 ms aus.

Die Angaben zu COM-Port, Interrupt-Nummer und Basisadresse dienen zur Information. Die angezeigten Werte repräsentieren die aktuell gültigen Werte. Die Änderung dieser Einstellungen erfolgt über Kommandozeilenparameter beim Aufruf der IDE11.

4.8.4. Desktop

Im Feld "Video Mode" ist es möglich, zwischen Farb- und Schwarzweissdarstellung sowie zwischen 25 und 43/50 Zeilen Anzeige (letzteres abhängig von der installierten PC-Grafikkarte) umzuschalten. Die Umschaltung erfolgt sofort nach Bestätigung mittels OK.

Der Wert für "Message Box Timeout" bestimmt, wie lange Meldungsfenster angezeigt werden. Die Angabe erfolgt in Sekunden. Der Wert 0 bedeutet dauernde Anzeige (bis der Benutzer <ENTER> drückt).

4.9. Window

In diesem Menü sind die üblichen Funktionen zur Beeinflussung von Fenstern versammelt. Sinnvoll sind die Funktionen insbesondere dann, wenn keine Maus zur Verfügung steht und stattdessen ausschließlich mit der Tastatur gearbeitet werden muß.

4.9.1. Size/move

Ermöglicht das Bewegen und die Änderung der Größe des aktiven (zuoberst liegenden) Fensters. Die Steuerung der Position erfolgt mit den Cursortasten. Durch gleichzeitiges Drücken der <SHIFT>-Taste wird die Größe des Fensters eingestellt. Längere Wege legt man bei gleichzeitigem Halten der <STRG>-Taste zurück. Die neue Lage des Fensters wird schließlich mit <ENTER> bestätigt, wohingegen <ESC> das Fenster zurück an die ursprüngliche Position bringt.

4.9.2. Zoom

Mittels "Zoom" wird die Größe des aktiven Fensters umgeschaltet, und zwar zwischen normaler (vom Benutzer eingestellt) und maximaler (gesamtes Desktop) Größe. Alternativ kann auch mit der Maus "gezoomt" werden. Dazu reicht ein Klick auf das Zoom-Symbol des Fensters (oben rechts im Rahmen des Fensters).

4.9.3. Tile

Ordnet alle Fenster untereinander bzw. nebeneinander an. Dadurch erspart man sich manuelles verschieben und sieht auf einen Blick, welche Fenster noch offen sind.

4.9.4. Cascade

Ordnet alle Fenster kaskadiert in einem übersichtlichen "Stapel" an.

4.9.5. Next

Hotkey: <F6>

Selbstverständlich kann man ein Fenster aktivieren, indem man es mit der Maus anklickt. Dazu muß es jedoch erst einmal, zumindest teilweise, sichtbar sein. Ist dies nicht der Fall, kann mit dem Befehl "Next" zum nächsten Fenster weitergeschaltet werden. Die Reihenfolge, in der die Fenster hervorgehoben werden, korrespondiert nicht unbedingt mit den evtl. sichtbaren Fenstern (oben rechts im Rahmen des Fensters). Ausschlaggebend ist nur die Reihenfolge, in der

die Fenster auf dem Desktop liegen. "Next" aktiviert stets das am weitesten unten befindliche Fenster.

4.9.6. Previous

Hotkey: <SHIFT-F6>

Die Funktion ist analog "Next", jedoch ist die Reihenfolge umgekehrt. "Previous" aktiviert das (nach dem aktiven Fenster) am weitesten oben befindliche Fenster.

4.9.7. Close

Hotkey: <ALT-F3>

Schließt das aktive Fenster. Funktion identisch mit Menüpunkt "File/Close".

4.10. Help

4.10.1. Manual

Hotkey: <F1>

Zeigt ein Online-Manual an, welches Informationen zur Assemblersprache und den Targeteinstellungen enthält. Die Datei namens MANUAL.DOC muß sich im selben Verzeichnis wie IDE11.EXE befinden.

5. Der integrierte Editor

Der integrierte Editor ist ein kompakter Multidateieditor für Textdateien bis maximal 64 KB Größe. Den Editorfunktionen wie z.B. Blockoperationen oder Suchen & Ersetzen liegen Wordstar-kompatible Tastenbelegungen zugrunde. Der Textcursor kann mit der Maus positioniert, und Blöcke mit der Maus markiert werden. Der Editor kann auf eine Zwischenablage (Clipboard) zurückgreifen. Über diese Zwischenablage können Blöcke ausgeschnitten und eingefügt werden, entweder innerhalb einer Datei oder zwischen zwei verschiedenen Dateien. Es steht eine Undo-Funktion zur Verfügung, die alle Änderungen seit der letzten Bewegung des Textcursors rückgängig macht.

Ein Editor-Fenster weist (so wie jedes Fenster in der IDE) mehrere Bedienelemente auf. Neben dem Dateinamen erscheint im oberen Bereich des Rahmens ein Schließsymbol (links) und ein Zoomsysmbol (rechts). Am rechten und unteren Rahmenteil sind Rollbalken zum beschleunigten Blättern im Text vorhanden. Links unten erscheint ein Stern-Symbol, und zwar genau dann, wenn der Editorinhalt seit dem letzten Speichern geändert wurde. Unmittelbar daneben zeigt ein Zeilen/Spalten-Indikator an, an welcher Stelle Sie sich im Text befinden.

Das Editorfenster läßt sich am oberen Rand "anfassen" und innerhalb des Bildschirms verschieben und an der rechten unteren Ecke können Sie das Fenster anfassen, um dessen Größe zu ändern. Alles in allem dürften Ihnen diese Bedienungshinweise bereits genügen, um sich in dem integrierten Editor heimisch zu fühlen. Eine tabellarische Übersicht über die Tastaturkommandos finden Sie im Anhang D.

Die Datei in einem Editorfenster wird durch ein "Save" Kommando des Benutzers gespeichert. Zusätzlich erfolgt ein automatisches Speichern vor jedem Übersetzungslauf des Assemblers.

6. Der integrierte Assembler

Der integrierte Assembler ist ein 2-Pass Assembler, welcher Quelltexte beliebiger Länge in zwei (internen) Durchläufen übersetzt. Im ersten Durchlauf werden Symbole (Labels) erfaßt, im zweiten Durchlauf werden diese Symbole dann zur Bildung von Ausdrücken verwendet und der Code sowie ggf. ein Programmlisting erstellt. Diese Funktionsweise schließt sog. Vorwärtsreferenzen aus, d.h. Symbole müssen vor Ihrer Verwendung im Quelltext eindeutig definiert sein. Die Symboltabelle wird dynamisch verwaltet, die maximale Anzahl von Symbolen (Labels) wird somit nur durch den vorhandenen Hauptspeicher beschränkt (1000 Symbole benötigen ca. 40 KB Speicher).

Bei erfolgreicher Übersetzung erzeugt der Assembler eine Objektdatei im Motorola S-Record Format (Informationen zu diesem Dateiformat befinden sich im Anhang).

6.1. Syntax

Jede Quelltextzeile hat folgenden syntaktischen Aufbau (gänzlich leere Zeilen im Quelltext sind ebenfalls zulässig):

```
[LABEL] [[INSTRUCTION] OPERANDS] [COMMENT]
```

LABEL ist ein Bezeichner mit maximal 31 signifikanten Stellen. Labels müssen - wie üblich - in der ersten Spalte des Textes beginnen, sonst werden sie als Instruction (Befehl) gewertet. Ein Doppelpunkt am Ende des Labels ist nicht notwendig, kann aber bei Bedarf angefügt werden

(wird vom Assembler ignoriert). Zulässige Zeichen in Labels sind Buchstaben, Ziffern, Unterstrich, Punkt und das Zeichen "@", mit Ausnahme des ersten Zeichen eines Labels, welches *keine* Ziffer und *nicht* "@" sein darf. Es wird zwischen Groß- und Kleinschreibung unterschieden, es sei denn, die Option "Ignore case" ist aktiv ("Options/Assembler"). Der Assembler weist einem Label den Programcounter *der* Quelltextzeile zu, in der das Label definiert ist. Eine Ausnahme bildet der Steuerbefehl EQU, welcher dem vor EQU stehenden Label den Wert des nach EQU folgenden Ausdrucks zuweist (siehe Erläuterung zu EQU).

INSTRUCTION ist eine Assembleranweisung oder ein Steuerbefehl. Die Schreibweise der Assembleranweisungen entspricht den Angaben im HC11 Reference Manual von Motorola. Anhang H listet alle Assembleranweisungen des HC11 tabellarisch auf. Die Steuerbefehle des Assemblers sind weiter unten ausführlich beschrieben. Groß- bzw. Kleinschreibung ist bei Assembleranweisungen bzw. Befehlen *ohne* Bedeutung.

OPERANDS steht für den/die Operanden einer Assembleranweisung (bzw. eines Steuerbefehls). Operanden bestehen aus einem (numerischen) Ausdruck und ggf. einem Kennzeichen zur Bestimmung der erforderlichen Adressierungsart. Sind mehrere Operanden erforderlich, wird mit Leerzeichen getrennt. Alternativ akzeptiert der Assembler zur Trennung auch ein Komma.

COMMENT ist ein (inhaltlich beliebiger) Kommentar. Es gibt mehrere Varianten zur Deklaration eines Kommentars, zulässig sind terminierte und unterminierte Kommentare.

Unterminierte Kommentare erstrecken sich vom Kommentarsymbol (Semi-kolon oder doppelter Schrägstrich) bis zum Zeilenende. Ein Stern gilt ebenfalls als Beginn eines unterminierten Kommentars, wenn vor dem Stern keine anderen Zeichen (abgesehen von Leerzeichen und anderen Kommentaren) stehen. Einige Beispiele zulässiger Formulierungen:

```
;--------------------------------
;Kommentarzeile bis zum Zeilenende
;--------------------------------
label0:   LDAA #$40  ; Kommentar
MainProg  ; Kommentar
// ------ Bemerkung ---------
label1:   LDAA #$40  // Erläuterung
*
* dies ist eine gültige Kommentarzeile
                      * dies ebenfalls
label2:   LDAA #$40  * das ist unzulässig!
```

Alternativ sind terminierte Kommentare verwendbar, welche durch die Zeichenkombination Schrägstrich-Stern eingeleitet und durch Stern-Schrägstrich beendet werden:

```
/* Kommentar */
```

Diese Art Kommentare kann an jeder Position der Quelltextzeile stehen, der Assembler behandelt einen solchen Kommentar wie Leerzeichen. Der Kommentar kann sich jedoch nicht über mehrere Zeilen erstrecken, er muß in derselben Zeile enden, in der er begann. Verschachtelte Kommentare sind nicht zulässig.

6.2. Ausdrücke

Ausdrücke (berechnete numerische Werte) können aus folgenden Bausteinen bestehen:

- Dezimalzahl 1234
- Hexadezimalzahl $ABCD
- Binärzahl %10101010
- Zeichenkonstante "A" oder 'A'
- Bezeichner MAX_VALUE
- Programcounter *
- Klammern (bzw.)
- Additionszeichen +
- Subtraktion -
- Multiplikation *
- Division /
- Vorzeichen + bzw. -

Mit maximal 20 Klammerebenen stehen praktisch unbegrenzte Möglichkeiten zur Bildung komplexer Ausdrücke zur Verfügung. Intern berechnet der Assembler Ausdrücke mit 32-Bit Genauigkeit (long integer). Die Regeln zur Bildung von Ausdrücken (EXPR) zeigt das folgende Syntaxdiagramm:

```
EXPR   = [ "+" | "-" ] TERM { ( "+" | "-" ) TERM } .
TERM   = FACTOR { ( "*" | "/" ) FACTOR } .
FACTOR = BEZEICHNER | ZAHL | STRING | "*" | "(" EXPR ")" .
```

Das Sternsymbol "*" in einem Ausdruck liefert den Wert des PC in dieser Zeile. ASCII Konstanten werden (wie Strings) in doppelte *oder* einfache Hochkommas eingeschlossen. Hexadezimalzahlen wird das Dollarzeichen "$", Binärzahlen das Prozentzeichen "%" vorangestellt. Klein- bzw. Großschreibung ist bei Hexadezimalziffern unrelevant, sowohl "a...f" als auch "A...F" sind gleichberechtigt verwendbar.

Im Folgenden einige Beispiele für (gültige) Ausdrücke:

```
COUNT * (ITEMSIZE + 1)
COUNT * $ff + 1
$A000 * 2
"A" + 1
((Value1 + %100) / (Value2 - *)) / 16
-MIN + (2000000 / (BASE + 100))
```

6.3. Adressierungsarten

Die folgende Tabelle faßt die Adressierungsarten des HC11 zusammen und erläutert die Schreibweisen:

Adressierungsart	Notation	Beispiel
Inherent		NOP
Immediate	#ii	LDAA #1
Immediate 16	#jjkk	LDD #$FFFF
Direct	<dd	LDAA <0
Extended	hhll	LDAA $1234
Indexed,X	ff,X	LDAA 0,X
Indexed,Y	ff,Y	LDAA 0,Y
Relative	rr	BSR dest

Erläuterung:

ii	8-Bit vorzeichenloser Wert (Wertebereich 0 ... 255 bzw. $00 ... $FF)
jjkk	16-Bit vorzeichenloser Wert (Wertebereich 0 ... 65535 bzw. $0000 ... $FFFF)
dd	8-Bit Adresse (H-Byte = $00) (Wertebereich 0 ... 255 bzw. $00 ... $FF)
hhll	16-Bit Adresse (Wertebereich 0 ... 65535 bzw. $0000 ... $FFFF)
ff	8-Bit vorzeichenloser Offset zum X- bzw. Y-Register (Wertebereich 0 ... 255 bzw. $00 ... $FF)

rr	vorzeichenbehafteter 8-Bit Offset (Wertebereich -128 ... +127 bzw. $80 ... $FF, $00 ... $7F) berechnet aus der Differenz der 16-Bit Adresse "dest" und dem Stand des Programcounter *nach* dem aktuellen Befehl

Die Bezeichnung Inherent bedeutet, daß keine Operanden erforderlich sind. Die Assembleranweisung enthält bereits alle relevanten Informationen; z.B. trifft dies auf die Anweisungen NOP, CLC oder PULA zu.

Die Adressierungsart Immediate wird mit einem Doppelkreuz "#" vor dem Operanden gekennzeichnet. Der Operand stellt einen unmittelbaren Wert dar, welcher als Byte (8 Bit) bzw. Word (16 Bit) in den Code eingefügt wird.

Indexed Adressierung bezieht sich immer auf das X- bzw Y-Register der CPU. Der Inhalt dieses Registers, zzgl. des im Operanden anzugebenden Offsets, bezeichnet die Speicheradresse, auf welche die CPU zugreift. Kennzeichen für diese Adressierungsart ist ein dem eigentlichen Operanden folgende Kombination ",X" bzw. ",Y".

Die Adressierungsart Extended verlangt als Operand einen 16-Bit Wert, der als Speicheradresse interpretiert wird.

Läßt ein Befehl optional die verkürzte Adressierungsart "Direct" zu, muß explizit das Kleiner-Als-Zeichen "<" vor dem Operanden angegeben werden (der Assembler setzt Direct Adressing *nicht* automatisch ein). Das MSB der gebildete Speicheradresse wird auf Null gesetzt, nur das LSB wird durch den Operanden geliefert. Mittels Direct Adressierung kann man somit stets nur auf den Adreßbereich $0000 bis $00FF zugreifen.

Die Adressierungsart Relative berechnet aus der im Operanden angegebenen Zieladresse und dem aktuellen Stand des Programcounters einen vorzeichenbehafteten 8-Bit Sprungoffset. Ist dieser Offset außerhalb des zulässigen Wertebereiches (-128 ... +127), wird ein Fehler gemeldet.

Befehle, bei denen *nur* Immediate-Operanden zulässig sind, werden *ohne* "#" vor dem Operanden notiert. Befehle, bei denen *nur* Direct-Adressierung in Frage kommt, machen das "<" vor dem Operanden überflüssig. Diese beiden Ausnahmeregelungen betreffen die folgenden Befehle:

```
BSET  operand mask
BCLR  operand mask
BRSET operand mask rel
BRCLR operand mask rel
```

Beispiele für die Schreibweise dieser Befehle:

```
        BSET  0,X %00010000   ; Indexed,X
        BSET  0,Y $40         ; Indexed,Y
        BSET  $20 %10000000   ; Direct
here    BRSET 0,X $80 here    ; Indexed,X
wait    BRSET 0,Y 1 wait      ; Indexed,Y
loop    BRSET flag $FF loop   ; Direct
```

6.4. Steuerbefehle

Steuerbefehle sind Befehle an den Assembler, die nicht als Mnemonic interpretiert und übersetzt werden, jedoch dennoch z.T. Code generieren können.

```
    ORG expression
```

Setzt den Programcounter auf den Wert "expression". Der neue Programcounter darf nicht kleiner sein als der aktuelle Programcounter, da sich sonst u.U. überlappende Codebereiche ergeben könnten.

```
label   EQU expression
```

Dem Bezeichner "label" wird der Wert "expression" zugewiesen. "label" muß in jedem Fall angegeben werden und darf nicht mehrfach definiert werden. Die Wertzuweisung an das Label erfolgt mit einer Breite/Genauigkeit von 16 Bit.

```
[label] RMB expression
[label] RMW expression
[label] RML expression
```

Reserviert einen Speicherbereich der Länge (SIZE * expression). Dieser Speicherbereich wird nicht initialisiert. SIZE ist 1, 2 bzw. 4 für RMB, RMW bzw. RML. Durch Angabe eines Labels ist die Bezugnahme auf diesen Speicherbereich möglich.

```
[label] DB expression {, expression }
[label] DW expression {, expression }
[label] DL expression {, expression }
```

Definiert eine oder mehrere Byte-, Word- bzw. Long-Konstanten, die in den Programmcode eingetragen werden. DB belegt für jedes Vorkommen von "expression" ein Byte, DW zwei Byte und DL vier Byte. Wie bei Motorola Prozessoren üblich, wird im Speicher das höchstwertige Byte (MSB) zuerst und das niedrigstwertige Byte (LSB) zuletzt eingetragen.

Neben numerischen Ausdrücken sind hier für "expression" auch String-Konstanten verwendbar. Die in doppelte (oder einfache) Hochkomma eingeschlossene Zeichenkette wird als ASCII-String im Programmcode eingefügt (das Hochkomma selbst *nicht*). Durch den Backslash "\" als sogenanntes Escapezeichen sind innerhalb der Zeichenkette auch nicht-druckbare Zeichen bzw. Sonderzeichen verwendbar. Die folgende Tabelle zeigt die Kombinationsmöglichkeiten:

Sonderz.	Wert	Bedeutung
\a	$07	alarm
\b	$08	back

\f	$0C	formfeed
\n	$0A	newline
\r	$0D	carriage return
\t	$09	tab
\v	$0B	vertical tab
\x??	$??	?? = 00 ... FF
\??	$??	?? = 00 ... FF

Folgt auf einen Backslash ein anderes als die oben angegebenen Zeichen, ergibt dies das Zeichen selbst. Aus "\z" wird also "z" und aus "\\" schließlich "\". Der Backslash als Escapezeichen ermöglicht darüberhinaus die Definition des (zur Kennzeichnung der String-Begrenzung) reservierten einfachen bzw. doppelten Hochkommas. Zur Illustration einige Anwendungsbeispiele:

```
msg_str DB "This message\r\nis for you!\r\n\x00"
def1    DB "\x01SIO"
def2    DB $02,"PIO"
def3    DB "\03","CTC"
hello   DB "\aRing!",0
msg2    DB "Er sagte \"Guten Abend!\"."
```

Strings sind normalerweise nur im Zusammenhang mit dem DB Befehl sinnvoll. Werden Strings innerhalb eines DW Befehls angegeben, erzeugt der Assembler für jedes Zeichen der Zeichenkette zwei Byte Code, wobei das höherwertige Byte auf Null gesetzt wird. Das Verfahren für DL ist analog, hier sind die drei höherwertigen Bytes Null.

```
INCLUDE "file"
```

Die angegebene Quelltextdatei "file" wird in die Hauptdatei eingefügt und assembliert. Die Includedatei "file" wird im aktuellen Verzeichnis gesucht. Es ist auch möglich, einen absoluten oder relativen Pfad mit anzugeben:

```
INCLUDE "MYPRG.A"
INCLUDE "C:\IDE11\LIB\MYPRG.A"
INCLUDE "..\LIB\MYPRG.A"
```

```
INCLUDE <file.ext>
```

Die Datei "file.ext" wird als Includedatei in die Hauptdatei eingefügt und assembliert. Für "file.ext" kommt nur eine einfache Dateiangabe, bestehend aus Dateiname und ggf. Erweiterung, in Frage. Diese Datei wird im Standard-Include-Verzeichnis gesucht (einstellbar in der IDE11 im Menü Options/Assembler). Lautet dieses Verzeichnis z.B. auf "\IDE11\LIB", so wird mit dem Steuerkommando

```
INCLUDE <HC11.H>
```

auf die Datei "\IDE11\LIB\HC11.H" zugegriffen.

```
NOLIST
```

Unterdrückt (zeitweilig) die Erzeugung eines Programmlistings ab der folgenden Zeile bis zum nächsten LIST Befehl.

```
LIST
```

Erlaubt wieder die Erzeugung eines Programmlistings (nach einem vorher plazierten NOLIST Befehl).

6.5. Kompatibilität

Der Assembler unterstützt aus Gründen der Kompatibilität eine Reihe alternativer Kommandos, welche funktionell identisch mit den oben diskutierten Standardbefehlen sind. Die folgende Tabelle stellt die möglichen Alternativen gegenüber:

Standard	Alternativ
ORG	.ORG
EQU	.EQU
RMB	.DS / DEFS / DS / DS.B
RMW	DS.W

Standard	Alternativ
RML	DS.L
DB	.DB / DEFB / DC.B / BYTE / .BYTE / FCB STRING / DCC/ FCC
DW	.DW / DEFW / DC.W / WORD / FDB
DL	DC.L
INCLUDE	.INCLUDE / INCL / LIB
LIST	.LIST
NOLIST	.NOLIST

Folgende Steuerbefehle werden aus Gründen der Kompatibilität akzeptiert, bleiben jedoch ohne Wirkung:

```
BSS
TEXT
DATA
END / .END
```

Ein Doppelkreuz in der ersten Spalte einer Zeile wird wie ein Leerzeichen interpretiert. Daher kann der Steuerbefehl INCLUDE auch wie folgt notiert werden:

```
#include "file.ext"
```

7. Kommunikation zwischen Host-PC und Target

Zur Kommunikation zwischen Host-PC und Target verwendet IDE11 eine serielle Verbindung (siehe Hardwareinstallation) und verschiedene Softwarekomponenten. Die Interaktion der einzelnen Komponenten soll im folgenden erläutert werden, um dem Anwender die Arbeitsweise von IDE11 verständlich zu machen und bestimmte Verhaltensmuster von IDE11 zu illustrieren.

In jedem HC11 ist, neben evtl. vorhandenen EPROM-, OTP- oder MaskROM-Ressourcen, ein kleiner Firmware-ROM implementiert. Dieser ROM Bereich trägt HC11-Befehle, die einen Bootlader realisieren. Der genaue Inhalt des Bootladers ist im Anhang des HC11 Reference Manuals für etliche HC11 Typen abgedruckt.

Der Bootlader des HC11 wird aktiviert, indem der Chip in den Special Bootstrap Mode versetzt wird (Mode-Pins MODA und MODB auf L-Potential legen) und anschließend ein Reset ausgelöst wird. In Folge dessen arbeitet der HC11 das Bootladerprogramm ab. Dieses Programm initialisiert die benötigten Systemkomponenten und legt die Sendeleitung des SCI (TxD) permanent auf L-Pegel. Dann wird gewartet, ob etwas auf der Empfangsleitung (RxD) passiert. Empfängt der Bootlader ein $FF-Byte, so erkennt er die gesendete Baudrate und lädt nachfolgend eintreffende Bytes in den internen RAM. Als Baudrate stehen 7812 Baud und 1200 Baud zur Verfügung (bei einem Quarztakt von 8 MHz). Wieviele Bytes geladen werden, hängt vom eingesetzten HC11-Typ ab. Der 68HC11A8 erwartet z.B. genau 256 Bytes. Jedes dieser 256 Bytes wird vom Bootlader als Echo zurückgeschickt, was eine Kontrolle der Übertragung ermöglicht. Ist der Ladevorgang beendet, springt der Bootlader zu Adresse $0000, der eben geladene Code wird also nun abgearbeitet.

Diesen Ablauf macht sich IDE11 zunutze. Die Entwicklungsumgebung löst zuerst einen Resetimpuls aus. Das vom HC11 gesendete Dauer-Low erzeugt in der UART des PC einen sog.

BREAK-Interrupt. Wurde diese Rückmeldung erkannt, wird ein $FF-Byte mit 7812 Baud gesendet (die tatsächliche Baudrate beträgt 7680 Baud, die Toleranz von 1,7% ist aber akzeptabel). Anschließend transferiert IDE11 einen Talker in den RAM des HC11. Der Bootlader startet schließlich diesen Talker.

IDE11 verwendet verschiedene Talker, je nachdem, welche Aufgabe zu erfüllen ist. Es gibt Talker für das Laden und Auslesen von Speicherbereichen, für den Einzelschrittbetrieb und für die Manipulation des CONFIG Registers. IDE11 realisiert gewissermaßen ein "häppchenweise" geladenes Monitorprogramm und kommt dadurch mit dem recht knapp bemessenen Speicherplatz des internen RAM aus (maximal werden 256 Byte belegt).

Die Talker verfügen über verschiedene Kommandosätze, die IDE11 selbständig verwaltet und bedarfsweise zum Einsatz bringt. Die Übertragung von Bytes geschieht i.d.R. gesichert durch ein Echo des Talkers (ähnlich dem Verhalten des Bootladers). Übertragungsfehler werden dadurch sicher aufgedeckt.

Die o.g. Voraussetzung, daß der HC11 im Special Bootstrap Mode arbeitet, legt die Vermutung nahe, Systeme mit externen Speicherbausteinen könnten mit IDE11 nicht verwendet werden. Durch die Möglichkeit die Betriebsart des HC11 per Software vom Special Bootstrap in den Expanded Mode umzuschalten, sind jedoch selbst Systeme mit vollem externem Speicherausbau (64 KB, abzüglich der intern belegten Ressourcen) leicht handhabbar. Hierzu muß lediglich das HPRIO Register modifiziert werden. IDE11 realisiert dies durch Einsatz entsprechend gestalteter Talker.

IDE11 muß entscheiden, welche Betriebsart für das jeweilige Target zutreffend ist. Hierzu werden die Einstellungen im Menü "Options/Target/Detail/External mem" ausgewertet. An dieser Stelle kann eingestellt werden, welche externen Speicherbereiche im Target belegt sind. Lauten

beide Einträge "None", behält IDE11 den Special Bootstrap Mode bei, andernfalls wird in den Expanded Mode umgeschaltet.

Eine Besonderheit liegt vor, wenn die Mode-Pins des Target (vergleichbar mit der Fernsteuerung des Reset) durch den PC umgeschaltet werden können. Diese Aufgabe wird durch den /RTS Ausgang der seriellen Schnittstelle des PC wahrgenommen, wenn die entsprechende Option ("Options/Target/Detail/Mode ctrl") aktiviert wurde. Die Umschaltung in den Expanded Mode erfolgt dann nicht mehr durch Schreiben auf das HPRIO-Register, sondern durch Umschalten der Mode-Pins von L- auf H-Pegel mit nachfolgendem Reset.

8. Der integrierte Singlestep-Debugger

Ein Keyfeature der IDE ist der Singlestep-Debugger. Dieser Systembestandteil ermöglicht die schrittweise Abarbeitung von Target-Software direkt auf dem 68HC11. Bis auf wenige Ausnahmen sind sämtliche Befehle des Mikrocontrollers verwendbar. Die einzige Verbindung zwischen PC und Target ist die bereits weiter vorn beschriebene 4-Draht-Kopplung über die serielle Schnittstelle des PC.

Das Target-Programm muß zur schrittweisen Abarbeitung im EEPROM des Mikrocontrollers stehen. Assemblieren Sie dazu Ihren zu testenden Quelltext und programmieren Sie den EEPROM durch Aufruf der Funktion Run/Execute (<F8>).

Für Ihr Mikrocontroller-Programm steht Ihnen neben dem gesamten EEPROM-Bereich auch der On-Chip-RAM von $0000 bis $006F sowie $00D6 bis $00FF (letzteres bevorzugt als Stack) zur freien Verfügung. Den Bereich $0070 bis $00CF dürfen Sie auf keinen Fall in Ihrem Programm modifizieren, hier werden die Target-seitigen Routinen für den Einzelschrittbetrieb bereitgehalten. Weiterhin ist der OC5-Pseudovektor $00D3/4/5 für diese Zwecke reserviert.

Starten Sie den Einzelschrittbetrieb mit dem Menübefehl Debug/Trace (bzw. <F7>). Der erste Befehl (Start bei der EEPROM-Anfangsadresse) wird abgearbeitet und unmittelbar anschließend erscheint ein Dialogfenster mit den Informationen zum Target-Status. Neben den jeweils zwei Accus und Index-Registern informiert Sie das Feld CCR über die Prozessorflags. Die Bedeutung der Bits ist konform zur Bezeichnung im HC11 Reference Manual:

- S Stop Disable
- X XIRQ Mask

- H Half Carry
- I IRQ Mask
- N Negative
- Z Zero
- V Overflow
- C Carry

Die Belegung der Ports A bis E wird angezeigt. Dabei handelt es sich um den Zustand, den der Prozessor durch einen Lesezugriff auf die entsprechenden (nicht gelatchten) Portadressen feststellt. Unter Umständen können (insbesondere bei Ausgängen) Differenzen zur tatsächlichen Belegung auftreten; Einzelheiten dazu finden Sie bei den detaillierten Portbeschreibungen im HC11 Reference Manual.

Zur Orientierung werden der Stand von Programcounter (PC) und Stackpointer (SP) angezeigt. Der PC zeigt auf das erste Byte des nächsten abzuarbeitenden Befehls. Dieser Befehl wird reassembliert und in Mnemonic-Schreibweise im Feld Instruction angezeigt. Handelt es sich um einen illegalen Opcode erscheint die Ausschrift "???". In diesem Fall sollte die Programmabarbeitung nicht fortgesetzt werden.

Der nächste Befehl wird abgearbeitet durch klicken auf den Schalter "Step" bzw. die Taste <ENTER>. <ESC> bzw. "Cancel" bricht den Einzelschrittbetrieb ab. Eine Wiederaufnahme der Abarbeitung ist nur ausgehend von der Anfangsadresse möglich.

Folgende Einschränkungen sind während des Singlestep-Debugging zu beachten:

- Das SCI ist initialisiert auf 7812 Bd und darf nicht mit abweichenden Werten reinitialisiert werden.
- Die Vorteiler dürfen nicht verändert werden.
- Die Instruktionen SEI, SWI, WAI, RTI und alle illegalen Opcodes sind verboten.
- Es sollten keine lokalen Interrupt-Masken freigegeben werden.
- Die Interrupt-Pseudo-Vektoren im RAM sollten nicht beschrieben werden.
- Schreibzugriffe auf den Bereich $0070 bis $00D5 sind verboten.
- Bei dem Befehl TAP sollte nicht versehentlich das I-Bit gesetzt werden.
- Die Befehle STOP und TEST sind nicht verwendbar.
- Der Stackpointer ist auf $00FF initialisiert, die maximale Stackbelastung ist damit 28 Byte. Der Stackpointer kann allerdings reinitialisiert werden (z.B. auf $006F).

Die meisten dieser Einschränkungen betreffen den Interrupt-Betrieb, welcher im Einzelschritt ohnehin kaum hinreichend austestbar ist. Somit haben Sie dennoch ein Maximum an Freiheit beim Debuggen, jedoch nur sehr geringe Anforderungen bezüglich der Hardware.

Anhang A: Systemvoraussetzungen

Host:

Um mit IDE11 arbeiten zu können, ist ein "IBM-kompatibler" PC mit folgenden Merkmalen erforderlich:

- Prozessor ab 80386 (empfohlen)
- MS-DOS 3.2 oder höher (kein DOS-Fenster unter Windows 3.x!)
- ca. 400 KB freier konventioneller RAM
- benötigter Platz auf der Festplatte: ca. 250 KB
- VGA-Grafikkarte (nur Textmode mit 25 bzw. 50 Zeilen verwendet)
- Maus (empfohlen)
- Bei Anschluß eines Target: eine freie serielle Schnittstelle, eingestellt auf COM1...4 mit IRQ 3,4,5 oder 7

Target:

In Betracht kommende Zielsysteme müssen mit einer der folgenden Motorola-MCU´s (bzw. einem der funktionsgleichen Second-Source Bauteile von Toshiba) ausgestattet sein:

- MC68HC11A8
- MC68HC11A1

- MC68HC11A0
- MC68HC11E9
- MC68HC11E1
- MC68HC11E0
- MC68HC811E2
- MC68HC11F1

Zudem muß das Zielsystem folgende Anforderungen erfüllen:

- 8 MHz Quarztakt
- freier SCI-Anschluß (RxD, TxD)
- RS232 Pegelwandler

Neue HC11 Typen:

Sollten Sie ein Zielsystem mit einem HC11-Typen einsetzen, welcher (noch) nicht in der obigen Liste zu finden ist, so gibt es die Möglichkeit, IDE11 daran anzupassen. Wenn Sie uns ein Exemplar Ihrer Hardware zur Verfügung stellen, bauen wir den neuen Baustein für Sie in IDE11 ein. Dieser Service ist kostenlos, vorausgesetzt wir können es technisch realisieren! (Dieses Angebot ist unverbindlich, Widerruf vorbehalten).

Anhang B: Liste der Dateien

Die folgenden Dateien befinden sich im Lieferumfang von IDE11. Sie liegen gepackt in der Archivdatei IDE11V23.ZIP vor. Die Pfadangaben beziehen sich auf eine Standardinstallation.

C:\IDE11\

file_ide.diz	Stichwortartige Kurzbeschreibung der Programm-Eigenschaften (wird von z.B. von BBS Systemen ausgewertet)
instruct.doc	Liste der HC11 Befehle
license.doc	Lizenzbedingungen
manual.doc	Online-Handbuch (Kurzform)
order.doc	Information über Bezugsmöglichkeiten
readme.doc	Aktuelle Informationen

C:\IDE11\BIN\

config.bin	Targetroutine zur Bearbeitung des CONFIG Registers
ide11.exe	Hauptprogramm der Integrierten Entwicklungsumgebung IDE11

ide11_e!.bat	Hilfsdatei zum Aufruf eines EPROM Emulators (ist ggf. an das verwendete Modell anzupassen)
ide11_s!.bat	Hilfsdatei zum Aufruf des Software Simulators TESTE68 (dieses ist ein separat zu erwerbendes Produkt, TESTE68 stammt nicht von MCT und ist nicht Bestandteil von IDE11!)
talkere.bin	Targetroutine zur seriellen Kommunikation (Expanded Multiplexed Mode Version)
talkers.bin	Targetroutine zur seriellen Kommunikation (Special Bootstrap Mode Version)
tracere.bin	Targetroutine für den Einzelschrittbetrieb (Expanded Multiplexed Mode Version)
tracers.bin	Targetroutine für den Einzelschrittbetrieb (Special Bootstrap Mode Version)

C:\IDE11\LIB\

hc11ae.h	Includedatei mit Registerdefinitionen für 68HC11Ax, 68HC11Ex und 68HC812E2
hc11f1.h	Includedatei mit Registerdefinitionen für 68HC11F1

memdump.a	Routinen zur (formatierten) Ausgabe von Speicherbereichen und Registern (Beispiel)
output.a	Routinen für formatierte Ausgaben über das SCI
serial.a	Low-Level Routinen für das SCI

C:\IDE11\SRC\

blink.a	Demoprogramm für 68HC11Ax/Ex
blinke2.a	Demoprogramm für 68HC811E2
blinkf1.a	Demoprogramm für 68HC11F1
oc5vec.a	Hilfsprogramm für den Einzelschrittbetrieb (muß geladen werden, wenn Target im Expanded Mode laufen soll)
protect.a	Programm zur Aktivierung des Software-Schreibschutzes eines externen EEPROM-Bausteins (ab Adresse $8000) für ZWERG11plus u.a.
unprot.a	Programm zum Aufheben des Software-Schreibschutzes eines externen EEPROM-Bausteins (ab Adresse $8000) für ZWERG11plus u.a.

Dateien, die erst zur Laufzeit erzeugt werden:

Die Datei IDE11.CFG sichert die im Menü "Options" vorgenommenen Einstellungen. Sie wird bei Verlassen der IDE im aktuellen Arbeitsverzeichnis angelegt. Beim nächsten Start von IDE11 (im gleichen Verzeichnis) wird sie automatisch eingelesen.

Die Datei IDE11.LOG (im aktuellen Arbeitsverzeichnis) enthält den Mitschnitt des Terminalfensters. Der Mitschnitt wird im Menü "Options/Terminal" aktiviert bzw. deaktiviert. Erneutes Aktivieren führt zum Überschreiben einer vorhandenen Log-Datei.

Für Quelltextdateien des Assemblers wird die Endung *.A empfohlen, Backupdateien legt der Editor mit der Extension *.BAK an. Listingdateien des Assemblers erhalten die Endung *.LST, Symboldateien die Endung *.MAP und Executables (S-Record Dateien) enden auf *.S19.

Dateien mit der Endung *.ERR sind temporär, sie enthalten Fehlerinformationen des letzten Assemblerlaufs. EPSIM.ROM ist ebenfalls eine temporäre Datei zur Übergabe einer Binärdatei an den EPROM Emulator.

Anhang C: Fehlermeldungen der IDE

Die Fehlermeldungen sind in mehrere Kategorien eingeteilt, Fehlermeldungen des Assemblers befinden sich getrennt hiervon in Anhang D.

Kommunikation/Target:

No response from target (missing BREAK)

Nach einem Reset des Target schickt der HC11 ein permanentes Low über seine TxD-Leitung. Bleibt dieses Break aus, kommt es zu einer Fehlermeldung. Normalerweise schickt der HC11 (wenn er im Special Bootstrap Mode läuft) als Antwort auf einen Resetimpuls über den TxD Pin ein permanentes Low-Signal an den Host. Dieses Signal erzeugt an der Empfangsleitung im PC einen sog. BREAK-Zustand. Die obige Fehlermeldung sagt aus, daß dieses BREAK Signal durch den PC nicht festgestellt werden konnte. Als Fehlerursache kommen in Frage: Der COM-Port des PC, das Verbindungskabel, der V.24-Pegelwandler auf dem Target, der HC11 bzw. die Stromversorgung des Target. Abhilfe durch systematische Verfolgung des Signalspiels zwischen PC und Target (Pegelprüfstift etc.).

Boot loader error! Bad handshake at $aaaa: TxD=$xx RxD=$yy

Beim Laden einer Systemroutine bzw. eines Anwenderprogramms in den internen RAM des HC11 mittels des HC11 Bootloaders ist es zu einem Fehler gekommen. Das Echobyte vom Bootloader ($yy) ist nicht identisch mit dem ursprünglich gesendeten Byte $xx. Der Fehler trat auf bei Adresse $aaaa.

Unable to connect to target (Function: xxx)

Der Talker konnte keine Kommunikation mit der Zielhardware aufbauen. Überprüfen Sie Stromversorgung und serielle Kommunikation (siehe auch Anmerkungen zu "No response..."). Der Fehler trat bei Ausführung der angegebenen Funktion (read, load oder execute) auf.

Verify failed! TX=$xx RX=$yy at Addr=$aaaa

Der Talker schickt beim Laden von Speicherbereichen für jedes Byte ein Echo-Byte zur Kontrolle zurück. Dies geschieht ähnlich dem Verfahren des HC11-Bootladers, jedoch wird das Echo-Byte durch einen echten Lesezugriff auf die Zieladresse ermittelt. Die Fehlermeldung zeigt, daß der gewünschte Wert $xx nicht erfolgreich geschrieben werden konnte, stattdessen hat die Speicherzelle auf Adresse $aaaa den Wert $yy. Ursache können u.a. Timingprobleme oder ein defekter Speicherbaustein (oder falsche Target-Settings) sein.

Communication malfunction!

Kommunikationsfehler zwischen PC (Talker) und Target bei Ausführung der Kommandos "Reassemble" bzw. "Memory Dump".

Could not start trace!

Die Trace-Systemroutine konnte zwar geladen werden, reagiert aber nicht wie erwartet.

Can´t trace!
TraceInfo not received!

Fehler beim Einzelschrittbetrieb. Nach jedem Programmschritt schickt die Target-MCU einen Satz Statusinformationen (sog. TraceInfo). Ist diese Information unvollständig oder bleibt vollständig aus, so wird diese Fehlermeldung erzeugt. Mögliche Ursachen liegen in einer Unterbrechung der Verbindung zum Target, Stromversorgungsprobleme und Timing-Probleme bei der Bedienung der seriellen Schnittstelle des PC. Kommt ein illegaler Opcode zur Ausführung, oder wird eine der im Abschnitt 7 genannten Einschränkungen außer acht gelassen, so bleibt die Rückmeldung der MCU ebenfalls aus.

Unable to read CONFIG register!
Unable to verify CONFIG register!

Der Talker war nicht in der Lage, das CONFIG Register auszulesen (Talker Kommunikation gestört).

Unable to change CONFIG register!

Beim Schreiben des CONFIG Registers ist es zu einer Fehlfunktion gekommen.

CONFIG value is $XX, but should be $YY!

Fehler beim Programmieren des CONFIG Registers. Geschrieben werden sollte YY, das Kontroll-Lesen ergibt aber einen Wert von XX. Neben Problemen bei der Kommunikation zum Target bestehen mögliche Fehlerursachen in einer falschen Target-Einstellung (Menü Options).

Weiterhin ist nicht für jede MCU jeder CONFIG-Wert zulässig, die MCU weist dann unzulässige Bits ab.

Syntax/Semantik/Logik:

EPSIM code must start at >= $8000!

Bei Verwendung eines EPROM-Simulators (EPSIM oder andere) wird angenommen, dieser deckt einen 32 KB Bereich von $8000 bis $FFFF ab. Bereiche unterhalb $8000 dürfen somit keinen Code enthalten.

Warning: Code exceeds internal RAM size!

Wenn ein Anwenderprogramm bei Adresse $0000 beginnt, geht IDE11 davon aus, dieses Programm soll (mittels des HC11 Bootladers) in den internen RAM geladen und gestartet werden. Die Warnung tritt auf, wenn das Ende des Codebereichs außerhalb der RAM-Größe liegt - die überzähligen Bytes werden verworfen.

Assembler error(s)

Es sind ein oder mehrere Fehler beim Übersetzen aufgetreten. Nach Bestätigung dieser Meldung öffnet sich ein Fehler-Fenster mit den registrierten Fehlern.

Executable is up to date

Keine Fehlermeldung im eigtl. Sinne; der Assembler weist darauf hin, daß die Übersetzung abgebrochen wurde, da bereits ein aktueller Assembler-Output (S-Record-File *.S19) vorliegt. Möchten Sie eine erneute Übersetzung erzwingen, so verwenden Sie den Menübefehl Run/Rebuild.

No loadable memory at Addr=aaaa!

Das ausgewählte Target hat an der genannten Adresse keinen (ladbaren) Speicher, gleichwohl wurde im Anwenderprogramm diese Adresse belegt.

Dateisystem/Dateiformat:

Missing S-Record ID!

Formatfehler beim Einlesen einer Zeile einer S-Record-Datei: der Kennbuchstabe "S" in der ersten Spalte fehlte.

Invalid S-Record type!

Formatfehler beim Einlesen einer Zeile einer S-Record-Datei: die Typkennung in der zweiten Spalte war weder "1", "0" noch "9" - alle anderen Typen sind nicht zulässig

Overlapping code areas!

Eine Adresse wurde in der S-Record-Datei mehr als einmal belegt.

No code created!

Die S-Record-Datei enthielt (im interessierenden Adressbereich) keine Daten.

Can´t find <file>

Die angegebene Datei sollte in den internen RAM des HC11 geladen werden, konnte aber nicht geöffnet werden.

File <file> is empty!

Die angegebene Datei sollte in den internen RAM des HC11 geladen werden, enthält aber keine Daten. Die Datei ist entweder leer oder evtl. beschädigt. Überprüfen Sie ggf. daie Dateistruktur des Datenträgers (z.B. mit CHKDSK o.ä.).

Unable to create <file>

Die angegebene Datei konnte nicht erzeugt werden (Fehler im Dateisystem bzw. Datei bereits vorhanden und schreibgeschützt).

Save file before use

Die geänderte Quelltextdatei wird automatisch gespeichert, wenn ein Assemble-Befehl ausgeführt werden soll. Handelt es sich um eine neue Datei, und brechen Sie den Save-As-Befehl mit Cancel ab, so wird die Übersetzung mit dieser Meldung vorzeitig beendet.

Hardware/System:

Fatal error: COMn not detected, Press any key to abort...

Verhindert den Start der IDE11, da der angegebene COM-Port (bzw. COM1 als Default) nicht gefunden/initialisiert werden konnte.

Not enough memory for this operation

Es steht zu wenig Hauptspeicher zur Verfügung. Benutzen Sie keine Shells, von denen aus IDE11 gestartet wird. Versuchen Sie ggf. das System zu optimieren (CONFIG.SYS / AUTOEXEC.BAT), um mehr Hauptspeicher zur Verfügung zu stellen.

Can´t shell to DOS (errorcode $cccc)!

Es war nicht möglich, den Kommandointerpreter zu laden. Wahrscheinlichste Ursache ist zu wenig Hauptspeicher oder ein Fehler beim Finden von COMMAND.COM.

Anhang D: Fehlermeldungen des Assemblers

Source file contains non-ASCII's

Die Quelltextdatei enthält unzulässige Steuerzeichen. Das Betrifft Zeichen im Bereich unter $20, außer TAB, CR und LF.

Source line is too long

Eine Quelltextzeile darf maximal 255 Zeichen lang sein.

Can't open source file

Der Assembler konnte die Quelldatei nicht öffnen. Entweder ist die Datei gar nicht vorhanden, oder sie ist gesperrt (in Netzwerkumgebungen möglich). Eine denkbare Ursache wäre auch ein zu geringer Wert für den Parameter FILES in der Systemdatei CONFIG.SYS.

Can't create object file

Die Objektdatei (S-Record-Datei *.S19) konnte nicht angelegt werden. Ursachen können sein: Der Dateiname enthält unzulässige Zeichen, der Pfad existiert nicht, die Datei ist vorhanden und schreibgeschützt oder die Datei ist gesperrt (Netzwerke).

Can't create error file

Die Fehlerdatei (*.ERR) konnte nicht angelegt werden. Mögliche Ursachen siehe oben.

Can't create list file

Die Listingdatei (*.LST) konnte nicht angelegt werden. Mögliche Ursachen siehe oben.

Can't create map file

Die Symboldatei (*.MAP) konnte nicht angelegt werden. Mögliche Ursachen siehe oben.

Can´t read source file

Der Assembler konnte die Quelltextdatei zwar öffnen, jedoch war es nicht möglich, aus der Datei zu lesen.

Label redefined

Ein Symbol wurde mehr als einmal definiert. Da Symbole bis zu 31 signifikante Stellen haben können, ist die Verwechslungsgefahr selbst bei langen Bezeichnern sehr gering. Ist jedoch die Option "Ignore case" eingeschaltet (im Menü Options/Assembler) kann es zu Überschneidungen von solchen Labels kommen, die sich nur in der Groß-/Kleinschreibung unterscheiden.

Illegal character in label

Bezeichner für Labels dürfen nur aus "a-z", "A-Z", "0-9" und den Zeichen ".", "@" und "_" gebildet werden. Das erste Zeichen darf keine Ziffer und kein Punkt sein.

Syntax error (scanner)

Der Assembler hat einen Syntaxfehler entdeckt. Es handelt sich um einen Schreibfehler, nicht um einen inhaltlichen oder kontextabhängigen Fehler. Der Scanner prüft, ob die Quelltextzeile

zulässige Zeichen bzw. Zeichenfolgen enthält. Trifft er auf ein unzulässiges Zeichen, oder ist eine Zeichenfolge nicht korrekt, so meldet er einen Fehler. Tritt z.B. das Zeichen "!" auf, so meldet der Scanner einen Fehler (außer in Strings und Kommentaren), da dieses Zeichen nicht Bestandteil der Assemblersprache ist.

Unknown command

Ein als Assembleranweisung bzw. Steuerbefehl interpretierter Bezeichner ist nicht in der Befehlsliste des Assemblers enthalten. Wahrscheinlich liegt ein Schreibfehler vor, oder ein Label beginnt regelwidrig *nicht* in der ersten Spalte.

Illegal addressing mode

Die gewählte Adressierungsart steht bei diesem Befehl nicht zur Verfügung. Siehe dazu Liste der HC11 Assembleranweisungen.

Syntax error (parser)

Der Assembler hat einen Syntaxfehler entdeckt, der in einer unzulässigen Abfolge von Symbolen besteht. Trifft der Assembler z.B. in einer Quelltextzeile als erstes auf einen String, so ist dies ein Fehler, obwohl der String an sich u.U. korrekt gebildet ist.

Branch distance too long (-128..+127)

Die Sprungweite für einen (Relativ-) Sprung ist zu groß. Oftmals ist es möglich, durch Umordnen des Quelltextes einige solcher Fehler zu umgehen. Alternativ muß mit Absolutsprüngen gearbeitet werden, oder man teilt einen Relativsprung in zwei (kürzere) Teilsprünge auf.

8-bit operand out of range (0..255)

Ein 8-Bit Operand ist außerhalb des Wertebereichs. Der Assembler benutzt in diesem Fall nur die unteren 8 Bit des Ausdrucks.

16-bit operand out of range (0..65535)

Ein 16-Bit Operand ist außerhalb des Wertebereichs. Der Assembler benutzt in diesem Fall nur die unteren 16 Bit des Ausdrucks.

Numeric expression expected

Der Assembler erwartete an dieser Stelle einen (numerischen) Ausdruck.

Symbol (identifier) not found

Ein in einem Ausdruck benutzter Bezeichner ist nirgendwo definiert. Der Bezeichner ist zuvor als Label oder mittels EQU zu definieren.

Missing closing bracket

Einem Klammerausdruck fehlt die schließende Klammer. Die Anzahl öffnender und schließender Klammern muß identisch sein.

PC overflow (PC > $FFFF)

Der Programcounter hat den maximal zulässigen Wert $FFFF überschritten. Es ist Zeit, über einen 16/32-Bit Controller mit mehr Speicher nachzudenken.

PC rewinded

Es wurde versucht, den Programcounter "zurückzudrehen". Das birgt die Gefahr des Entstehens sich überlappender Codebereiche und ist deshalb nicht erlaubt. Ordnen Sie die Routinen im Quelltext in aufsteigender Reihenfolge an.

Forward reference

Ein Ausdruck konnte im Pass 2 des Assemblers nicht berechnet werden, weil er eine Bezugnahme auf einen weiter unten im Quelltext definierten Ausdruck enthält. Bezeichner müssen im Quelltext *vor* Ihrer Verwendung definiert werden.

Label required (EQU)

Die EQU Steueranweisung erfordert ein in der Zeile befindliches Label.

String expected

Der Assembler erwartete an dieser Stelle eine Zeichenkette. Zeichenketten können in Hochkaommas oder Anführungszeichen eingeschlossen werden, wobei das öffnende mit dem schließenden Zeichen identisch sein muß.

INCLUDE nesting too deep

Eine Include-Datei darf wiederum Include-Dateien enthalten. Die Verschachtelungstiefe ist jedoch auf max. 10 begrenzt.

INCLUDE syntax

Die Schreibweise des Operanden einer Include-Anweisung ist nicht korrekt. Der als Operand anzugebende Dateiname muß in Anführungszeichen oder in spitze Klammern eingeschlossen sein. Im ersten Fall kann ein Dateiname incl. Pfad (auch relativ) angegeben werden, im zweiten Fall ist nur ein Dateiname der Form name.ext (ohne Pfadangabe) erlaubt.

Line code buffer overflow

Eine Assemblerzeile darf nicht mehr als 255 Byte Code erzeugen. Dieser Fehler tritt beim HC11-Assembler nicht auf.

Internal error

Ein interner Fehler des Assemblers hat zum Abbruch der Übersetzung geführt. Derartige Fehler sollten dem Autor mitgeteilt werden (wenn möglich mit Angaben zur Hardware und dem konkreten Arbeitsumfeld).

Division by zero

Der Operand dieser Zeile enthält eine unzulässige Divisionsoperation durch Null.

Too many errors

Abbruch der Übersetzung durch zu viele (jedoch nicht-fatale) Fehler (mehr als 20). Dies ist keine Fehlermeldung an sich, sondern nur der Hinweis, daß im Quelltext u.U. weitere Fehler enthalten sind.

Too many warnings

Abbruch der Übersetzung durch zu viele Warnungen (mehr als 100). Dies ist keine Fehlermeldung an sich, sondern nur der Hinweis, daß im Quelltext u.U. weitere Warnungen auftreten.

Unknown error

Die Art von Fehlern, die Murphy stets beschreibt.

Anhang E: Tastaturkommandos des integrierten Editors

In der Tabelle wurden, soweit vorhanden, sowohl die primären als auch die sekundären (alternativen) Tastaturkommandos aufgeführt.

Cursorbewegungen:

^S		ein Zeichen nach links
^D		ein Zeichen nach rechts
^A		ein Wort nach links
^F		ein Wort nach rechts
^E		eine Zeile nach oben
^X		eine Zeile nach unten
^R	PgUp	eine Seite nach oben
^C	PgDn	eine Seite nach unten
^QR	^PgUp	zum Textanfang
^QC	^PgDn	zum Textende
^QD	Home	zum Zeilenanfang
^QS	End	zum Zeilenende

Löschbefehle:

^G	Del	Zeichen an Cursorposition löschen
^H	<--	Zeichen links des Cursors löschen
^T		Wort an Cursorposition löschen
^Y		Zeile löschen
^QH		ab Cursorposition bis Zeilenanfang löschen
^QY		ab Cursorposition bis Zeilenende löschen
^U		Änderungen seit letzter Cursorbewegung rückgängig machen (Undo)

Steuerbefehle

^V		Umschaltung Modus: Einfügen / Überschreiben
^O		Umschaltung Modus: einrücken / nicht einrücken

Blockbefehle:

^KB		Blockanfang markieren

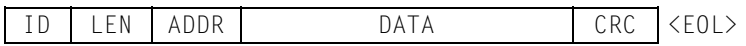
^KK	^Ins	Blockende markieren und Block in die Zwischenablage kopieren
^KC	Shift-Ins	Block aus der Zwischenablage einfügen
^KY	Shift-Del	Block ausschneiden und in die Zwischenablage übertragen
^KH		Blockmarkierung verbergen
^Del		Block löschen

Suchen und Ersetzen (jeweils ab aktueller Cursorposition):

^QF	Suchen
^QA	Suchen und Ersetzen
^L	Wiederhole Suchen (und Ersetzen)

Anhang F: Motorola S-Record Format

Das von Motorola publizierte S-Record Format ist ein Dateiformat zur Definition von Objektdateien (Maschinencode, Executables) unter Verwendung einer textuellen (ASCII-) Notation, die es erlaubt, diese Objektdateien mit jedem beliebigen Texteditor zu betrachten oder zu ändern. Eine S-Record Datei besteht aus einer beliebigen Anzahl S-Records bzw. Zeilen. Eine jede Zeile hat die folgende logische Struktur:



Das Feld ID gibt den S-Record Typ an. IDE11 verwendet die Zeichenpaare "S1" und "S9". Alle weiteren Felder werden gebildet aus Paaren von Hexziffern, beispielsweise "A9", "55" oder "0F".

Das Feld LEN besteht aus einem derartigen Paar und bestimmt die Anzahl der folgenden Ziffernpaare (enthält die Ziffernpaare der Felder ADDR, DATA und CRC).

ADDR ist die Anfangsadresse der Datenbytes dieser Zeile. Das Feld besteht aus zwei Byte (erst H-, dann L-Byte), d.h. aus zwei Ziffernpaaren.

DATA enthält die eigentlichen Codebytes, die das Maschinenprogramm bilden. DATA umfaßt (LEN - 3) Bytes bzw. Zeichenpaare.

Im Feld CRC befindet eine Prüfsumme. Sie wird gebildet aus den Werten der Zeichenpaare der Felder LEN, ADDR und DATA. CRC ist das (LSB des) Einerkomplement der Summe der vorgenannten Werte.

EOL schließlich steht symbolisch für den durch CR, LF ($0D, $0A) gebildeten Zeilenvorschub.

Ein Beispiel soll die Handhabung verdeutlichen:

S1	13	2000	13A400262741010167CC10FF05C7A501	D1

Dieser S1-Record definiert $13-3 = $10 Bytes ab Adresse $2000 des Zielsystems. Die Ziffernpaare des DATA Feldes ergeben eine Summe von $04FB. Addiert man die $13 aus dem LEN Feld sowie $20 und $00 aus dem ADDR Feld hinzu, ergibt sich ein Wert von $052E. Das Einerkomplement des LSB ($2E) ergibt $D1. Dies ist der korrekte Wert für das CRC Feld.

Neben den S1-Records, welche die eigentlichen Daten enthalten, wird auch der S9-Typ verwendet. Dieser Typ beendet eine Serie von S1-Records. Abgesehen von dieser Terminierungs-Funktion kann in einem S9-Record die Startadresse des Programms vermerkt werden. Der Aufbau des S9-Records entspricht dem S1 Typ, wobei jedoch das Feld DAT leer bleibt. Das Feld ADDR spezifiziert die Startadresse des Programms. Ist hier $0000 eingetragen, wird angenommen, das die Adresse des ersten geladenen Codebyte gleichzeitig die Startadresse des Programms ist. Ein typischer S9-Record sieht wie folgt aus:

S9	03	B600	46

Neben den beschriebenen S1- und S9-Records gibt es weiterhin S0-Records, die Beschreibungs- bzw. Kommentarzwecken vorbehalten sind. Die Typen S2 bis S8 spielen im

Zusammenhang mit anderen CPU´s bzw. Mikrocontrollern eine Rolle, werden von IDE11 jedoch nicht benötigt.

Zur Illustration sei eine komplette S-Record Datei gezeigt, dies ist die S-Record Datei des Beispielprogramms "BLINK.A":

```
S113B6008E00FFB6100084BFB71000BDB61BB61085
S113B610008A40B71000BDB61B20E8CE80000901A7
S106B62026FC39C8
S903B60046
```

Anhang G: Format der Symboldatei

IDE11 erzeugt (optional) bei jedem Assemblerlauf eine Symboldatei, welche u.a. zur Weiterverarbeitung durch den HC11 Software-Simulator TESTE68 (Fremdprodukt, nicht im Lieferumfang von IDE11 enthalten) geeignet ist. Um die Symboldatei bei einem Assemblerlauf zu erzeugen, muß die Option "Write symbols" im Menü "Options/Assembler" angekreuzt werden.

Die Symboldatei erhält den Namen der übersetzten Quelle und die Endung *.MAP. Sie enthält alle im übersetzten Quelltext definierten symbolischen Bezeichner sowie deren numerische Werte. Die Symboldatei ist eine Textdatei; sie enthält in jeder Zeile folgende Informationen:

VAL	SPACE	SYMBOL	<EOL>

Die Bedeutung der Felder:

VAL

ist eine maximal vierstellige Hexadezimalzahl (*ohne* führendes Dollarzeichen).

SPACE

ist (mindestens) ein Leerzeichen.

SYMBOL

ist der Bezeichner des Symbols mit max. 31 Stellen.

EOL
steht schließlich für das Zeilenende, auf PC´s wird dies durch die Zeichenfolge CR, LF ($0D, $0A) realisiert.

Die Ausgabe der Symbole erfolgt *nicht* geordnet. Leere Zeilen sowie Zeilen, die nicht mit einer Hexziffer beginnen (insbesondere /* ... */ als Kommentar), werden ignoriert.

Anhang H: Liste der HC11 Assembleranweisungen

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
ABA	Add Accumulators A + B -> A	--	1B	--	1	2	-	-	|	-	|	|	|	|
ABX	Add B to X X + 00:B -> X	--	3A	--	1	3	-	-	-	-	-	-	-	-
ABY	Add B to Y Y + 00:B -> Y	--	183A	--	2	4	-	-	-	-	-	-	-	-
ADCA	Add with carry to A A + M + C -> A	IMM	89	ii	2	2	-	-	|	-	|	|	|	|
		DIR	99	dd	2	3								
		EXT	B9	hhll	3	4								
		IX	A9	ff	2	4								
		IY	18A9	ff	3	5								
ADCB	Add with carry to B B + M + C -> B	IMM	C9	ii	2	2	-	-	|	-	|	|	|	|
		DIR	D9	dd	2	3								
		EXT	F9	hhll	3	4								
		IX	E9	ff	2	4								
		IY	18E9	ff	3	5								
ADDA	Add Memory to A A + M -> A	IMM	8B	ii	2	2	-	-	|	-	|	|	|	|
		DIR	9B	dd	2	3								
		EXT	BB	hhll	3	4								
		IX	AB	ff	2	4								
		IY	18AB	ff	3	5								

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
ADDB	Add Memory to B	IMM	CB	ii	2	2	-	-	|	-	|	|	|	|
	B + M -> B	DIR	DB	dd	2	3								
		EXT	FB	hhll	3	4								
		IX	EB	ff	2	4								
		IY	18EB	ff	3	5								
ADDD	Add 16-Bit to D	IMM	C3	jjkk	3	4	-	-	-	-	|	|	|	|
	D + M:M+1 -> D	DIR	D3	dd	2	5								
		EXT	F3	hhll	3	6								
		IX	E3	ff	2	6								
		IY	18E3	ff	3	7								
ANDA	AND A with memory	IMM	84	ii	2	2	-	-	-	-	|	|	0	-
	A * M -> A	DIR	94	dd	2	3								
		EXT	B4	hhll	3	4								
		IX	A4	ff	2	4								
		IY	18A4	ff	3	5								
ANDB	AND B with memory	IMM	C4	ii	2	2	-	-	-	-	|	|	0	-
	B * M -> B	DIR	D4	dd	2	3								
		EXT	F4	hhll	3	4								
		IX	E4	ff	2	4								
		IY	18E4	ff	3	5								
ASL	Arithmetic Shift Left	EXT	78	hhll	3	6	-	-	-	-	|	|	|	|
	C <- b7b6b5b4b3b2b1b0 <- 0	IX	68	ff	2	6								
		IY	1868	ff	3	7								
ASLA		--	48	--	1	2								
ASLB		--	58	--	1	2								
ASLD	Arithmetic Shift Left Double	--	05	--	1	3	-	-	-	-	|	|	|	|
	C <- b15b14b13..3b2b1b0 <- 0													

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
ASR	Arithmetic Shift Right	EXT	77	hhll	3	6	-	-	-	-	|	|	|	|
	b7 -> b7b6b5b4b3b2b1b0 -> C	IX	67	ff	2	6								
		IY	1867	ff	3	7								
ASRA		--	47	--	1	2								
ASRB		--	57	--	1	2								
BCC	Branch if Carry Clear	REL	24	rr	2	3	-	-	-	-	-	-	-	-
	?C=0													
BCLR	Clear Bit(s)	DIR	15	ddmm	3	6	-	-	-	-	|	|	0	-
	M*(/mm) -> M	IX	1D	ffmm	3	7								
		IY	181D	ffmm	4	8								
BCS	Branch if Carry Set	REL	25	rr	2	3	-	-	-	-	-	-	-	-
	?C=1													
BEQ	Branch if = Zero	REL	27	rr	2	3	-	-	-	-	-	-	-	-
	?Z=1													
BGE	Branch if >= Zero	REL	2C	rr	2	3	-	-	-	-	-	-	-	-
	?N^V=0													
BGT	Branch if > Zero	REL	2E	rr	2	3	-	-	-	-	-	-	-	-
	?Z+(N^V)=0													
BHI	Branch if Higher	REL	22	rr	2	3	-	-	-	-	-	-	-	-
	?C+Z=0													
BHS	Branch if Higher or Same	REL	24	rr	2	3	-	-	-	-	-	-	-	-
	?C=0													

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
BITA	Bit(s) Test A with Memory	IMM	85	ii	2	2	-	-	-	-	|	|	0	-
	A*M	DIR	95	dd	2	3								
		EXT	B5	hhll	3	4								
		IX	A5	ff	2	4								
		IY	18A5	ff	3	5								
BITB	Bit(s) Test B with Memory	IMM	C5	ii	2	2	-	-	-	-	|	|	0	-
	B*M	DIR	D5	dd	2	3								
		EXT	F5	hhll	3	4								
		IX	E5	ff	2	4								
		IY	18E5	ff	3	5								
BLE	Branch if <= Zero	REL	2F	rr	2	3	-	-	-	-	-	-	-	-
	?Z+(N^V)=1													
BLO	Branch if Lower	REL	25	rr	2	3	-	-	-	-	-	-	-	-
	?C=1													
BLS	Branch if Lower or Same	REL	23	rr	2	3	-	-	-	-	-	-	-	-
	?C+Z=1													
BLT	Branch < Zero	REL	2D	rr	2	3	-	-	-	-	-	-	-	-
	?N^V=1													
BMI	Branch if Minus	REL	2B	rr	2	3	-	-	-	-	-	-	-	-
	?N=1													
BNE	Branch if Not = Zero	REL	26	rr	2	3	-	-	-	-	-	-	-	-
	?Z=0													
BPL	Branch if Plus	REL	2A	rr	2	3	-	-	-	-	-	-	-	-
	?N=0													

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
BRA	Branch Always ?1=1	REL	20	rr	2	3	-	-	-	-	-	-	-	-
BRCLR	Branch if Bit(s) Clear ?M*mm=0	DIR IX IY	13 1F 181F	ddmmrr ffmmrr ffmmrr	4 4 5	6 7 8	-	-	-	-	-	-	-	-
BRN	Branch Never ?1=0	REL	21	rr	2	3	-	-	-	-	-	-	-	-
BRSET	Branch if Bit(s) Set ?(/M)*mm=0	DIR IX IY	12 1E 181E	ddmmrr ffmmrr ffmmrr	4 4 5	6 7 8	-	-	-	-	-	-	-	-
BSET	Set Bit(s) M+mm -> M	DIR IX IY	14 1C 181C	ddmm ffmm ffmm	3 3 4	6 7 8	-	-	-	-	|	|	0	-
BSR	Branch to Subroutine PC + 2 -> PC PCL -> (SP); SP-1 -> SP PCH -> (SP); SP-1 -> SP PC + rr -> PC	REL	8D	rr	2	6	-	-	-	-	-	-	-	-
BVC	Branch if Overflow Clear ?V=0	REL	28	rr	2	3	-	-	-	-	-	-	-	-
BVS	Branch if Overflow Set ?V=1	REL	29	rr	2	3	-	-	-	-	-	-	-	-
CBA	Compare A to B A-B	--	11	--	1	2	-	-	-	-	|	|	|	|

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
CLC	Clear Carry Bit 0 -> C	--	0C	--	1	2	-	-	-	-	-	-	-	0
CLI	Clear Interrupt Mask 0 -> I	--	0E	--	1	2	-	-	-	0	-	-	-	-
CLR	Clear Memory Byte 0 -> M	EXT	7F	hhll	3	6	-	-	-	-	0	1	0	0
		IX	6F	ff	2	6								
		IY	186f	ff	3	7								
CLRA	Clear Accumulator A 0 -> A	--	4F	--	1	2	-	-	-	-	0	1	0	0
CLRB	Clear Accumulator B 0 -> B	--	5F	--	1	2	-	-	-	-	0	1	0	0
CLV	Clear Overflow Flag 0 -> V	--	0A	--	1	2	-	-	-	-	-	-	0	-
CMPA	Compare A to Memory A-M	IMM	81	ii	2	2	-	-	-	-	|	|	|	|
		DIR	91	dd	2	3								
		EXT	B1	hhll	3	4								
		IX	A1	ff	2	4								
		IY	18A1	ff	3	5								
CMPB	Compare B to Memory B-M	IMM	C1	ii	2	2	-	-	-	-	|	|	|	|
		DIR	D1	dd	2	3								
		EXT	F1	hhll	3	4								
		IX	E1	ff	2	4								
		IY	18E1	ff	3	5								

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
COM	1's Complement Memory Byte	EXT	73	hhll	3	6	-	-	-	-	|	|	0	1
	$FF-M -> M	IX	63	ff	2	6								
		IY	1863	ff	3	7								
COMA	1's Complement A	--	43		1	2	-	-	-	-	|	|	0	1
	$FF-A -> A													
COMB	1's Complement B	--	53		1	2	-	-	-	-	|	|	0	1
	$FF-B -> B													
CPD	Compare D to Memory 16-Bit	IMM	1A83	jjkk	4	5	-	-	-	-	|	|	|	|
	D - M:M+1	DIR	1A93	dd	3	6								
		EXT	1AB3	hhll	4	7								
		IX	1AA3	ff	3	7								
		IY	CDA3	ff	3	7								
CPX	Compare X to Memory 16-Bit	IMM	8C	jjkk	3	4	-	-	-	-	|	|	|	|
	X - M:M+1	DIR	9C	dd	2	5								
		EXT	BC	hhll	3	6								
		IX	AC	ff	2	6								
		IY	CDAC	ff	3	7								
CPY	Compare Y to Memory 16-Bit	IMM	188C	jjkk	4	5	-	-	-	-	|	|	|	|
	Y - M:M+1	DIR	189C	dd	3	6								
		EXT	18BC	hhll	4	7								
		IX	1AAC	ff	3	7								
		IY	18AC	ff	3	7								
DAA	Decimal Adjust A	--	19	--	1	2	-	-	-	-	|	|	|	|
	Adjust Sum to BCD													

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
DEC	Decrement Memory Byte	EXT	7A	hhll	3	6	-	-	-	-	|	|	|	-
	M-1 -> M	IX	6A	ff	2	6								
		IY	186A	ff	3	7								
DECA	Decrement Accumulator A	--	4A	--	1	2	-	-	-	-	|	|	|	-
	A-1 -> A													
DECB	Decrement Accumulator B	--	5A	--	1	2	-	-	-	-	|	|	|	-
	B-1 -> B													
DES	Decrement Stack Pointer	--	34	--	1	3	-	-	-	-	-	-	-	-
	SP-1 -> SP													
DEX	Decrement Index Register X	--	09	--	1	3	-	-	-	-	-	|	-	-
	X-1 -> X													
DEY	Decrement Index Register Y	--	1809	--	2	4	-	-	-	-	-	|	-	-
	Y-1 -> Y													
EORA	Exclusive OR A with Memory	IMM	88	ii	2	2	-	-	-	-	|	|	0	-
	A ^ M -> A	DIR	98	dd	2	3								
		EXT	B8	hhll	3	4								
		IX	A8	ff	2	4								
		IY	18A8	ff	3	5								
EORB	Exclusive OR B with Memory	IMM	C8	ii	2	2	-	-	-	-	|	|	0	-
	B ^ M -> B	DIR	D8	dd	2	3								
		EXT	F8	hhll	3	4								
		IX	E8	ff	2	4								
		IY	18E8	ff	3	5								

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
FDIV	Fractional Divide 16 by 16 D/X -> X; r -> D	--	03	--	1	41	-	-	-	-	-	|	|	|
IDIV	Integer Divide 16 by 16 D/IX -> IX; r -> D	--	02	--	1	41	-	-	-	-	-	|	0	|
INC	Increment Memory Byte M + 1 -> M	EXT IX IY	7C 6C 186C	hhll ff ff	3 2 3	6 6 7	-	-	-	-	-	|	|	-
INCA	Increment Accumulator A A + 1 -> A	--	4C	--	1	2	-	-	-	-	-	|	|	-
INCB	Increment Accumulator B B + 1 -> B	--	5C	--	1	2	-	-	-	-	-	|	|	-
INS	Increment Stack Pointer SP + 1 -> SP	--	31	--	1	3	-	-	-	-	-	-	-	-
INX	Increment Index Register X X + 1 -> X	--	08	--	1	3	-	-	-	-	-	|	-	-
INY	Increment Index Register Y Y + 1 -> Y	--	1808	--	2	4	-	-	-	-	-	|	-	-
JMP	Jump Effective Address -> PC	EXT IX IY	7E 6E 186E	hhll ff ff	3 2 3	3 3 4	-	-	-	-	-	-	-	-

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
JSR	Jump to Subroutine	DIR	9D	dd	2	5	-	-	-	-	-	-	-	-
	PC + 3 -> PC (for EXT/IY)	EXT	BD	hhll	3	6								
	PC + 2 -> PC (for DIR/IX)	IX	AD	ff	2	6								
	PCL -> (SP); SP-1 -> SP	IY	18AD	ff	3	7								
	PCH -> (SP); SP-1 -> SP													
	Effective Address -> PC													
LDAA	Load Accumulator A	IMM	86	ii	2	2	-	-	-	-	|	|	0	-
	M -> A	DIR	96	dd	2	3								
		EXT	B6	hhll	3	4								
		IX	A6	ff	2	4								
		IY	18A6	ff	3	5								
LDAB	Load Accumulator B	IMM	C6	ii	2	2	-	-	-	-	|	|	0	-
	M -> B	DIR	D6	dd	2	3								
		EXT	F6	hhll	3	4								
		IX	E6	ff	2	4								
		IY	18E6	ff	3	5								
LDD	Load Double Accumulator D	IMM	CC	jjkk	3	3	-	-	-	-	|	|	0	-
	M -> A; M+1 -> B	DIR	DC	dd	2	4								
		EXT	FC	hhll	3	5								
		IX	EC	ff	2	5								
		IY	18EC	ff	3	6								
LDS	Load Stack Pointer	IMM	8E	jjkk	3	3	-	-	-	-	|	|	0	-
	M:M+1 -> SP	DIR	9E	dd	2	4								
		EXT	BE	hhll	3	5								
		IX	AE	ff	2	5								
		IY	18AE	ff	3	6								

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
LDX	Load Index Register X	IMM	CE	jjkk	3	3	-	-	-	-	|	|	0	-
	M:M+1 -> X	DIR	DE	dd	2	4								
		EXT	FE	hhll	3	5								
		IX	EE	ff	2	5								
		IY	CDEE	ff	3	6								
LDY	Load Index Register Y	IMM	18CE	jjkk	4	4	-	-	-	-	|	|	0	-
	M:M+1 -> Y	DIR	18DE	dd	3	5								
		EXT	18FE	hhll	4	6								
		IX	1AEE	ff	3	6								
		IY	18EE	ff	3	6								
LSL	Logical Shift Left	EXT	78	hhll	3	6	-	-	-	-	|	|	|	|
	C <- b7b6....b1b0 <- 0	IX	68	ff	2	6								
		IY	1868	ff	3	7								
LSLA		--	48	--	1	2								
LSLB		--	58	--	1	2								
LSLD	Logical Shift Left Double	--	05	--	1	3	-	-	-	-	|	|	|	|
	C <- b15b16....b1b0 <- 0													
LSR	Logical Shift Right	EXT	74	hhll	3	6	-	-	-	-	0	|	|	|
	0 -> b7b6....b1b0 -> C	IX	64	ff	2	6								
		IY	1864	ff	3	7								
LSRA		--	44	--	1	2								
LSRB		--	54	--	1	2								
LSRD	Logical Shift Right Double	--	04	--	1	3	-	-	-	-	0	|	|	|
	0 -> b15b16....b1b0 -> C													
MUL	Multiply 8 by 8	--	3D	--	1	10	-	-	-	-	-	-	-	|
	A*B -> D													

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
NEG	2's Complement Memory Byte	EXT	70	hhll	3	6	-	-	-	-	|	|	|	|
	0 - M -> M	IX	60	ff	2	6								
		IY	1860	ff	3	7								
NEGA	2's Complement A	--	40	--	1	2	-	-	-	-	|	|	|	|
	0 - A -> A													
NEGB	2's Complement B	--	50	--	1	2	-	-	-	-	|	|	|	|
	0 - B -> B													
NOP	No Operation	--	01	--	1	2	-	-	-	-	-	-	-	-
ORAA	OR Accumulator A (inclusive)	IMM	8A	ii	2	2	-	-	-	-	|	|	0	-
	A + M -> M	DIR	9A	dd	2	3								
		EXT	BA	hhll	3	4								
		IX	AA	ff	2	4								
		IY	18AA	ff	3	5								
ORAB	OR Accumulator B (inclusive)	IMM	CA	ii	2	2	-	-	-	-	|	|	0	-
	B + M -> M	DIR	DA	dd	2	3								
		EXT	FA	hhll	3	4								
		IX	EA	ff	2	4								
		IY	18EA	ff	3	5								
PSHA	Push A onto Stack	--	36	--	1	3	-	-	-	-	-	-	-	-
	A -> (SP); SP-1 -> SP													
PSHB	Push B onto Stack	--	37	--	1	3	-	-	-	-	-	-	-	-
	B -> (SP); SP-1 -> SP													
PSHX	Push X onto Stack (Lo First)	--	3C	--	1	4	-	-	-	-	-	-	-	-
	X -> (SP); SP-2 -> SP													

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
PSHY	Push Y onto Stack (Lo First) Y -> (SP); SP-2 -> SP	--	183C	--	2	5	-	-	-	-	-	-	-	-
PULA	Pull A from Stack SP+1 -> SP; (SP) -> A	--	32	--	1	4	-	-	-	-	-	-	-	-
PULB	Pull B from Stack SP+1 -> SP; (SP) -> B	--	33	--	1	4	-	-	-	-	-	-	-	-
PULX	Pull X from Stack SP+2 -> SP; (SP) -> X	--	38	--	1	5	-	-	-	-	-	-	-	-
PULY	Pull Y from Stack SP+2 -> SP; (SP) -> Y	--	1838	--	2	6	-	-	-	-	-	-	-	-
ROL	Rotate Left C <- b7b6...b1b0 <- C	EXT	79	hhll	3	6	-	-	-	-	|	|	|	|
		IX	69	ff	2	6								
		IY	1869	ff	3	7								
ROLA		--	49	--	1	2								
ROLB		--	59	--	1	2								
ROR	Rotate Right C -> b7b6...b1b0 -> C	EXT	76	hhll	3	6	-	-	-	-	|	|	|	|
		IX	66	ff	2	6								
		IY	1866	ff	3	7								
RORA		--	46	--	1	2								
RORB		--	56	--	1	2								

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
RTI	Return from Interrupt SP+1 -> SP; (SP) -> CCR SP+1 -> SP; (SP) -> A SP+1 -> SP; (SP) -> B SP+2 -> SP; (SP) -> X SP+2 -> SP; (SP) -> Y SP+2 -> SP; (SP) -> PC	--	3B	--	1	12	|	|	|	|	|	|	|	|
RTS	Return from Subroutine SP+2 -> SP; (SP) -> PC	--	39	--	1	5	-	-	-	-	-	-	-	-
SBA	Subtract B from A A-B -> A	--	10	--	1	2	-	-	-	-	|	|	|	|
SBCA	Subtract with Carry from A A-M-C -> A	IMM DIR EXT IX IY	82 92 B2 A2 18A2	ii dd hhll ff ff	2 2 3 2 3	2 3 4 4 5	-	-	-	-	|	|	|	|
SBCB	Subtract with Carry from B B-M-C -> B	IMM DIR EXT IX IY	C2 D2 F2 E2 18E2	ii dd hhll ff ff	2 2 3 2 3	2 3 4 4 5	-	-	-	-	|	|	|	|
SEC	Set Carry 1 -> C	--	0D	--	1	2	-	-	-	-	-	-	-	1
SEI	Set Interrupt Mask 1 -> I	--	0F	--	1	2	-	-	-	1	-	-	-	-

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
SEV	Set Overflow Flag 1 -> V	--	0B	--	1	2	-	-	-	-	-	-	1	-
STAA	Store Accumulator A A -> M	DIR EXT IX IY	97 B7 A7 18A7	dd hhll ff ff	2 3 2 3	3 4 4 5	-	-	-	-	|	|	0	-
STAB	Store Accumulator B B -> M	DIR EXT IX IY	D7 F7 E7 18E7	dd hhll ff ff	2 3 2 3	3 4 4 5	-	-	-	-	|	|	0	-
STD	Store Accumulator D A -> M, B -> M+1	DIR EXT IX IY	DD FD ED 18ED	dd hhll ff ff	2 3 2 3	4 5 5 6	-	-	-	-	|	|	0	-
STOP	Stop Internal Clocks	--	CF	--	1	2	-	-	-	-	-	-	-	-
STS	Store Stack Pointer SP -> M:M+1	DIR EXT IX IY	9F BF AF 18AF	dd hhll ff ff	2 3 2 3	4 5 5 6	-	-	-	-	|	|	0	-
STX	Store Index Register X X -> M:M+1	DIR EXT IX IY	DF FF EF CDEF	dd hhll ff ff	2 3 2 3	4 5 5 6	-	-	-	-	|	|	0	-

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
STY	Store Index Register Y Y -> M:M+1	DIR	18DF	dd	3	5	-	-	-	-	|	|	0	-
		EXT	18FF	hhll	4	6								
		IX	1AEF	ff	3	6								
		IY	18EF	ff	3	6								
SUBA	Subtract Memory from A A-M -> A	IMM	80	ii	2	2	-	-	-	-	|	|	|	|
		DIR	90	dd	2	3								
		EXT	B0	hhll	3	4								
		IX	A0	ff	2	4								
		IY	18A0	ff	3	5								
SUBB	Subtract Memory from B B-M -> B	IMM	C0	ii	2	2	-	-	-	-	|	|	|	|
		DIR	D0	dd	2	3								
		EXT	F0	hhll	3	4								
		IX	E0	ff	2	4								
		IY	18E0	ff	3	5								
SUBD	Subtract Memory from D D - M:M+1 -> D	IMM	83	iikk	3	4	-	-	-	-	|	|	|	|
		DIR	93	dd	2	5								
		EXT	B3	hhll	3	6								
		IX	A3	ff	2	6								
		IY	18A3	ff	3	7								
SWI	Software Interrupt PC+1 -> PC PC -> (SP); SP-2 -> SP Y -> (SP); SP-2 -> SP X -> (SP); SP-2 -> SP A -> (SP); SP-1 -> SP B -> (SP); SP-1 -> SP CCR -> (SP); SP-1 -> SP 1 -> I; SWI_VEC -> PC	--	3F	--	1	14	-	-	-	1	-	-	-	-

MNEMO	OPERATION	MODE	CODE	OPRNDS	B	C	S	X	H	I	N	Z	V	C
TAB	Transfer A to B A -> B	--	16	--	1	2	-	-	-	-	|	|	0	-
TAP	Transfer A to CCR A -> CCR	--	06	--	1	2	|	v	|	|	|	|	|	|
TBA	Transfer B to A B -> A	--	17	--	1	2	-	-	-	-	|	|	0	-
TEST	TEST (Only in Test Modes) Address Bus Counts	--	00	--	1	-	-	-	-	-	-	-	-	-
TPA	Transfer CCR to A CCR -> A	--	07	--	1	2	-	-	-	-	-	-	-	-
TST	Test for Zero or Minus M-0	EXT	7D	hhll	3	6	-	-	-	-	|	|	0	0
		IX	6D	ff	2	6								
		IY	186D	ff	3	7								
TSTA	A-0	--	4D	--	1	2								
TSTB	B-0	--	5D	--	1	2								
TSX	Transfer Stack Pointer to X SP+1 -> X	--	30	--	1	3	-	-	-	-	-	-	-	-
TSY	Transfer Stack Pointer to Y SP+1 -> Y	--	1830	--	2	4	-	-	-	-	-	-	-	-
TXS	Transfer X to Stack Pointer IX-1 -> SP	--	35	--	1	3	-	-	-	-	-	-	-	-
TYS	Transfer Y to Stack Pointer IY-1 -> SP	--	1835	--	2	4	-	-	-	-	-	-	-	-

```
-------------------------------------------------------------------------------
MNEMO  OPERATION                       MODE CODE OPRNDS B  C  S X H I N Z V C
-------------------------------------------------------------------------------
WAI    Wait for Interrupt              --   3E   --     1 **  - - - - - - - -
       PC+1 -> PC
       PC -> (SP); SP-2 -> SP
       Y -> (SP); SP-2 -> SP
       X -> (SP); SP-2 -> SP
       A -> (SP); SP-1 -> SP
       B -> (SP); SP-1 -> SP
       CCR -> (SP); SP-1 -> SP
       wait...
-------------------------------------------------------------------------------
XGDX   Exchange D with X               --   8F   --     1  3  - - - - - - - -
       X <-> D
-------------------------------------------------------------------------------
XGDY   Exchange D with Y               --   188F --     2  4  - - - - - - - -
       Y <-> D
-------------------------------------------------------------------------------
```

Operands:

```
    dd   8-bit direct address $0000-$00ff. (High byte assumed to be 0)
    ff   8-bit positive offset $00 (0) to $FF (256) added to index
    hh   High order byte of 16-bit extended address.
    ii   One byte of immediate data.
    jj   High order byte of 16-bit immediate data.
    kk   Low order byte of 16-bit immediate data.
    ll   Low order byte of 16-bit extendet address.
    mm   8-bit mask (set bits to be affected).
    rr   Signed relative offset $80 (-128) to $7F (+127). Offset
         relative to the address following the machine code offset byte
```

Condition Codes:

```
-    Bit not changed
0    Always cleared (logic 0).
1    Always set (logic 1).
|    Bit cleared or set depending on operation.
v    Bit may be cleared, cannot become set.
```

Notes:

```
**   12 cycles are used beginning with the opcode fetch. A wait
     state is entered which remains in effect for an integer
     number of MPU E-clock cycle(n) until an interrupt is
     recognized. Finally, two additional cycles are used to fetch
     the appropriate interrupt vector (total=14+n).
```