TwinPEEKs

Monitor für das HC12 Welcome Kit
Softwareversion 1.2

Benutzerhandbuch

1. Februar 2001

MCT Elektronikladen GbR
Hohe Str. 9-13 D-04107 Leipzig
Telefon: +49-(0)341-2118354
Fax: +49-(0)341-2118355
Email: leipzig@elektronikladen.de
Web: http://www.elektronikladen.de/mct

Inhalt

TwinPEEKs Monitor für HC12 Welcome Kit

Software Version 1.2

1. Über TwinPEEKs

TwinPEEKs ist ein Monitorprogramm für die 68HC12 Mikrocontrollerfamilie. In diesem Dokument wird ausschließlich die TwinPEEKs Version für das **HC12 Welcome Kit** beschrieben (MCU: 68HC812A4). Es gibt weitere Programmvarianten mit z.T. abweichenden Eigenschaften.

Wie für ein Monitorprogramm üblich, dient TwinPEEKs - neben dem Betrachten und Modifizieren von Speicherbereichen - in erster Linie zum Laden und Ausführen von Anwenderprogrammen.

TwinPEEKs belegt 2 KB des internen EEPROM-Bereichs und 256 Byte RAM. Somit verbleiben 2 KB EEPROM und 768 Byte RAM für Programme des Anwenders. Einzelheiten zur Speicheraufteilung siehe Abschnitt "Memory Map".

Das HC12-Zielsystem wird mit TwinPEEKs im Normal Single Chip Mode betrieben. Ein Einsatz im Expanded Mode ist grundsätzlich möglich, erfordert aber u.U. eine Anpassung der Initialisierungssequenz des Monitors. Dazu ist der TwinPEEKs Quelltext erforderlich, welcher als Erweiterungsprodukt zum HC12 Welcome Kit angeboten wird.

TwinPEEKs setzt voraus, daß das Zielsystem muß mit einer Quarzfrequenz von 16 MHz betrieben wird. Die Kommunikation mit dem Host-PC erfolgt über die erste serielle Schnittstelle (SCI0) mit 9600 Baud, das Übertragungsprotokoll ist 8N1 (keine weiteren Protokolleinstellungen notwendig).

2. Handhabungshinweise

Schreibweise von Zahlen

Alle Zahlenangaben erfolgen hexadezimal ohne weitere Zusätze. Groß-/Kleinschreibung spielt keine Rolle.

Beispiele:

```
1234
a90
BFFE
```

Monitorkommandos

Ein Monitorkommando besteht aus einem einzelnen Buchstaben, ggf. gefolgt von einer Liste von Argumenten.

Die Argumenteliste umfaßt null bis drei Adreßargumente, die voneinander durch Leerzeichen oder Komma getrennt sind.

Beispiele:

```
M 1000 1200 F000
m 1000,10a0,8000
m1000,10a0 8000
```

Adressen

Der Adreßraum ist umfaßt 64 KB, die Adressen sind demzufolge maximal 4-stellig. Beim HC12 beinhaltet dies (wie schon beim HC11) auch die Adreßbereiche der Ports und Steuerregister.

Wichtig: Endadressen beziehen sich stets auf das folgende, nicht enthaltene Byte. Der Befehl:

```
d 1000 1200
```

zeigt z.B. den Adreßbereich von $1000 bis incl. $11FF an.

Eingaben

Eingaben des Benutzers werden über einen Eingabepuffer abgewikkelt. Gültige ASCII-Zeichencodes liegen im Bereich $20 bis $7E.

Mittels Backspace ($08) kann das Zeichen links des Cursors gelöscht werden. Die <ENTER> Taste ($0A) schließt die Eingabe ab.

Warmstart

Bei Eingabe des Zeichens <ESC> wird die aktuelle Operation ohne Verzögerung abgebrochen und ein Warmstart durchgeführt. Dabei wird der Stackpointer auf den Ausgangswert zurückgesetzt und das SCI reinitialisiert.

3. Befehlsübersicht

Dump Memory

Syntax: D [<AADR> [<EADR>]]

Zeigt den Speicherinhalt an. In jeder Bildschirmzeile werden 16 Bytes in hexadezimaler Form dargestellt. Hinzu kommt die Darstellung der ASCII-Interpretation dieser 16 Zeichen. Zeichen außerhalb des druckbaren Bereiches ($20 bis $7E) werden durch einen Punkt ersetzt.

Für den Befehl sind ein, zwei oder kein Argument zulässig, z.B.:

`d f000 f800`	Zeigt den Speicherbereich von $F000 bis $F7FF an
`d f000`	Zeigt den Speicherbereich ab $F000 an. Es werden $0100 Bytes angezeigt.
`d`	Zeigt *die nächsten* $0100 Bytes an (ausgehend von der zuletzt erreichten Endadresse)

Edit Memory

Syntax: E [<AADR>]

Ermöglicht das Ändern von Speicherstellen, beginnend bei der angegebenen Adresse.

Für den Befehl sind ein oder kein Argument zulässig, z.B.:

`e f000`	Edit Memory ab $F000
`e`	Edit Memory (ausgehend von der zuletzt erreichten Endadresse)

Der Monitor zeigt daraufhin den aktuellen Inhalt an der jeweiligen Adresse an und erwartet eine Eingabe. Die Eingabe kann entweder ein neuer Wert für diese Adresse oder einer der folgenden Befehle sein:

<ENTER>	Zeigt die nächste Speicherstelle an.
-	Zeigt die vorhergehende Speicherstelle an.
=	Zeigt die aktuelle Speicherstelle erneut an.
.	Beendet den Edit Mode
Q	Ende (wie zuvor)

Wurde durch den Benutzer ein neuer Wert eingegeben, so schreibt der Monitor diesen Wert in den Speicher und zeigt daraufhin die nächste Speicherstelle an.

Ist die Speicheradresse eine EEPROM-Zelle, so wird automatisch der passende Programmieralgorithmus angewendet. Die vom Monitor belegten Teile des EEPROM sind jedoch stets schreibgeschützt. Ein Schreibzugriff auf diese Bereiche verursacht eine Fehlermeldung.

Fill Memory

Syntax: F <AADR> <EADR> <BY>

Füllt einen Speicherbereich mit dem angegebenen Bytewert.

Für den Befehl sind drei Argumente erforderlich, z.B.:

f f000 f800 ff	Füllt den Speicherbereich von $F000 bis $F7FF mit dem Wert $FF

Wenn sich ein Schreibfehler ereignet, wird die Funktion an der augenblicklichen Adresse abgebrochen.

Goto Address

Syntax: G [<AADR>]

Führt einen Unterprogramm-Sprung (mittels JSR) zur angegebenen Adresse aus. Der Monitor gibt eine Meldung aus, an welcher Adresse die Programmausführung fortgesetzt wird. Kehrt das Anwenderprogramm mittels RTS zum Monitor zurück, wird ein Warmstart ausgeführt (Initialisierung von Stackpointer und SCI).

Für den Befehl sind ein oder kein Argument zulässig, z.B.:

`g f100`	Ruft das Unterprogramm bei Adresse $F100 auf
`g`	Führt das Unterprogramm an der aktuellen Adresse aus

Help

Syntax: H

Gibt einen kurzen Hilfe-Text aus.

Der Befehl erfordert keine Argumente:

`h`	Zeigt die verfügbaren Kommandos an

Load S-Records

Syntax: L [<OFFS>]

Lädt Daten im S-Record Format in den Speicher. Die erforderlichen Speicheradressen sind in der S-Record Datei vermerkt, können aber durch Angabe eines Offsets verschoben werden.

Für den Befehl sind ein oder kein Argument zulässig, z.B.:

`l`	Lädt eine S-Record Datei in den Speicher
`l e000`	Lädt eine S-Record-Datei und verschiebt die enthaltenen Adreßangaben um den Offset $E000 (z.B. werden Daten aus der S-Record Datei im Bereich $1000 bis $10FF in den Speicher von $F000 bis $F0FF geladen)

Das PC-seitig verwendete Terminalprogramm muß es dem Benutzer ermöglichen, nach Ausgabe der Meldung "Loading..." des Monitors, die gewünschte S-Record Datei zu senden. Diese Funktion wird in der Terminalsoftware oft als "Text-" bzw. "ASCII-Upload" bezeichnet.

Die gesendeten Zeichen werden vom Monitor nicht geechot. Bei Auftreten eines Schreibfehlers bricht die Funktion ab.

Der Monitor verarbeitet S-Record Zeilen vom Typ S0, S1 und S9. Der Aufbau dieser S-Records ist im Anhang ausführlich beschrieben.

Beim Laden von S-Records direkt in den EEPROM ist zu beachten, daß jeder Schreibzugriff bis zu 20 ms dauern kann (10 ms, wenn die Zelle zuvor bereits den "Gelöscht"-Wert $FF enthielt). Da der Monitor nur eine begrenzte Anzahl Zeichen puffern kann, darf die Geschwindigkeit der Datenübertragung im Mittel einen bestimmten Wert nicht überschreiten. Dieser Wert ergibt sich aus der oben angeführten Programmierzeit und der Tatsache, daß in der S-Record Datei jedes Byte durch zwei Zeichen repräsentiert wird (siehe auch Anhang: S-Record Format). Es ergibt sich eine maximale Übertragungsrate von ca. 100 Zeichen pro Sekunde (200 Zeichen pro Sekunde, wenn zuvor bereits alle Bytes gelöscht wurden).

Es gibt zwei sinnvolle Varianten, um diesen Anforderungen zu genügen - sie sollen im folgenden kurz geschildert werden:

I. Zeilenweises Laden

Dies ist die effektivste Variante. Der Ablauf im Detail: TwinPEEKs empfängt nach Eingabe des "L"-Kommandos die erste Zeile der vom PC gesendeten S-Record Datei und überträgt sie in einen Zeilenpuffer. Anschließend wertet TwinPEEKs die Zeile aus und programmiert die darin enthaltenen Bytes. Schließlich sendet TwinPEEKs als Quittungszeichen ein Sternchen ("*") an den PC. Das PC-seitige Terminalprogramm muß in der Lage sein, vor dem Senden der nächsten Zeile auf dieses Quittungszeichen zu warten.

Bei dem verbreiteten Terminalprogramm aus Windows 3.x (TERMINAL.EXE) kann man folgende Einstellungen anwenden:

- Menü "Einstellungen", Eintrag "Textübertragung" anklicken
- "Protokoll" = Zeilenweise
- "Auf Sendeaufforderung warten" ankreuzen und ein * eintragen

II. Zwischenpufferung

Paßt der zu ladende Programmcode komplett in den RAM Bereich (768 Bytes stehen zur Verfügung), so kann die S-Record Datei mit der vollen Geschwindigkeit von 9600 Baud zuerst dorthin geladen werden. Zur Umleitung der Zieladressen in den RAM Bereich nutzt man die Offset-Option des Ladebefehls. Anschließend wird der Inhalt mit dem Move Kommando in den EEPROM Bereich kopiert.

Beispiel: Eine S-Record Datei enthalte Code für die Adressen $F000 bis $F1FF. Der freie RAM Bereich beginnt bei $0800. Die Lade- und Transferbefehle lauten in diesem Fall:

```
L 1800
M 0800 0A00 F000
```

Move Memory

Syntax: M <AADR> <EADR> <ADR2>

Kopiert den Speicherbereich von <AADR> bis <EADR> nach <ADR2>. Daraus ergibt sich die Länge <LEN> des zu kopierenden Speicherbereiches mit:

<LEN> = <EADR> - <AADR>

Für den Befehl sind drei Argumente erforderlich, z.B.:

`m 1000 1800 f000`	Kopiert den Speicherbereich von $1000 bis $17FF nach $F000 (bis $F7FF)

Eventuelle Überschneidungen von Quell- und Zielbereich werden korrekt behandelt, indem das Kopieren entweder mit der niedrigsten oder der höchsten Adresse beginnt. Ist die Zieladresse <ADR2> kleiner als die Quelladresse <AADR> wird mit steigenden Adresse kopiert, andernfalls mit sinkenden Adressen.

Bei Auftreten eines Schreibfehlers bricht die Funktion ab.

4. Schreibzugriffe auf den internen EEPROM

Der Monitor ist in der Lage, Schreibzugriffe auf den internen EEPROM (Adreßbereich $F000 - $F7FF) korrekt auszuführen. Hierzu wird bei jedem Schreibzugriff eine Programmierroutine in einen RAM-Puffer übertragen und die Routine im RAM-Puffer angesprungen. Das ist notwendig, weil der EEPROM während des Programmiervorgangs nicht auslesbar ist, demzufolge zu diesem Zeitpunkt auch kein Programm darin ablaufen kann.

Jeder Schreibzugriff wird durch den Monitor verifiziert. Dadurch können Schreibfehler erkannt werden, es erfolgt die Ausgabe einer Fehlermeldung.

Um die Gefahr der Zerstörung des Monitorcodes zu vermeiden, werden Schreibzugriffe auf den vom Monitor belegten Bereich ($F800-$FFFF) abgewiesen. Hierzu wird der Block Protect Mechanismus des HC12 wie folgt eingesetzt (Auszug aus dem Initialisierungsteil des Monitors):

```
ldaa #%10111111          ; Protect all EEPROM blocks
staa EEPROT              ; except BPROT6 area [F000-F7FF]
ldaa #$fe                ; Disable changes of EEPROT
staa EEMCR ; by setting PROTLCK bit
```

Um den gesamten EEPROM zu programmieren, also auch den von TwinPEEKs geschützten Teil, ist der Einsatz eines BDM12-basierten Laders erforderlich. Eine preisgünstige Lösung stellt das Motorola B32EVB dar, welches zusätzlich zum Programmieren des internen EEPROMs auch das Debuggen des Anwenderprogramms unterstützt (siehe http://www.elektronikladen.de/b32evb.html).

5. Schreiben auf Ports und Steuerregister

Nicht alle Ports bzw. Steuerregister des HC12 sind gleichermaßen lesbar und beschreibbar, schließlich handelt es sich nicht um simple RAM-Zellen Ein korrekt ausgeführter Schreibzugriff auf ein Steuerregister kann daher durchaus eine Fehlermeldung hervorrufen. Das passiert insbesondere dann, wenn der vom Register zurückgelesene Wert sich von dem geschriebenen unterscheidet.

6. Autostart-Funktion

Nach Reset und einigen grundlegenden Initialisierungen stellt das Monitorprogramm fest, ob die MCU-Signale PAD0 und PH7 miteinander verbunden sind. Ist dies der Fall, wird nicht das Monitorprogramm weiter abgearbeitet, sondern ein Sprung zur Adresse $F000 ausgeführt. Durch diese Funktion ist der Autostart eines bei $F000 beginnenden Anwenderprogramms möglich.

Um die Autostart-Funktion zu aktivieren, müssen lediglich die benachbarten Pins 35 (PH7) und 37 (PAD0) des Steckverbinders ST6 mittels Jumper verbunden werden.

7. Interruptvektoren

Die Interruptvektoren des HC12 liegen am Ende des 64 KB umfassenden Adreßraumes, d.h. innerhalb des schreibgeschützten Monitorcodes. Um dennoch Interruptfunktionen in einem Anwenderprogramm zu ermöglichen, leitet der Monitor alle Interruptvektoren (außer den Resetvektor) auf Adressen im internen RAM um. Das Verfahren entspricht der Vorgehensweise des HC11 im Special Bootstrap Mode.

Das Anwenderprogramm setzt den benötigten Interruptvektor, indem es einen Sprungbefehl in den RAM-Pseudovektor einträgt. Um z.B. den SPI-Interrupt nutzen zu können, muß ein Anwenderprogramm folgende Schritte ausführen:

```
ldaa #$06               ; JMP Opcode
staa $0BC7              ; SPI Pseudo Vector
ldd  #isrFunc           ; Jump Address
std  $0BC8              ; SPI Pseudo Vector + 1
```

Der folgende Ausschnitt aus dem Assemblerlisting des Monitorprogramms dokumentiert die Adressen der umgeleiteten Interruptvektoren:

```
FFCE : 0B B8    dc.w  RAMTOP-72  ; KWUH
FFD0 : 0B BB    dc.w  RAMTOP-69  ; KWUJ
FFD2 : 0B BE    dc.w  RAMTOP-66  ; ATD
FFD4 : 0B C1    dc.w  RAMTOP-63  ; SCI1
FFD6 : 0B C4    dc.w  RAMTOP-60  ; SCI / SCI0
FFD8 : 0B C7    dc.w  RAMTOP-57  ; SPI
FFDA : 0B CA    dc.w  RAMTOP-54  ; Pulse Accu Input Edge
FFDC : 0B CD    dc.w  RAMTOP-51  ; Pulse Accu Overflow
FFDE : 0B D0    dc.w  RAMTOP-48  ; Timer Overflow
FFE0 : 0B D3    dc.w  RAMTOP-45  ; TOC5 / TC7
FFE2 : 0B D6    dc.w  RAMTOP-42  ; TOC4 / TC6
FFE4 : 0B D9    dc.w  RAMTOP-39  ; TOC3 / TC5
FFE6 : 0B DC    dc.w  RAMTOP-36  ; TOC2 / TC4
FFE8 : 0B DF    dc.w  RAMTOP-33  ; TOC1 / TC3
FFEA : 0B E2    dc.w  RAMTOP-30  ; TIC3 / TC2
FFEC : 0B E5    dc.w  RAMTOP-27  ; TIC2 / TC1
FFEE : 0B E8    dc.w  RAMTOP-24  ; TIC1 / TC0
FFF0 : 0B EB    dc.w  RAMTOP-21  ; RTI
FFF2 : 0B EE    dc.w  RAMTOP-18  ; IRQ / KWUD
FFF4 : 0B F1    dc.w  RAMTOP-15  ; XIRQ
FFF6 : 0B F4    dc.w  RAMTOP-12  ; SWI
FFF8 : 0B F7    dc.w  RAMTOP-9   ; Illegal Opcode
FFFA : 0B FA    dc.w  RAMTOP-6   ; COP Fail
FFFC : 0B FD    dc.w  RAMTOP-3   ; Clock Monitor Fail
FFFE : F8 00    dc.w  main       ; Reset
```

8. Memory Map

Die folgende Tabelle gibt Auskunft über die Speicherbelegung des HC12 bei Verwendung des TwinPEEKs Monitorprogramms:

Start	Ende	Speicher	Belegung
$0800	$0AFF	RAM	= Frei für Anwenderprogramm =
$0B00	$0BB7	RAM	Monitor Variablen und Stack
$0BB8	$0BFF	RAM	Umgeleitete Interruptvektoren
$F000	$F7FF	EEPROM	= Frei für Anwenderprogramm =
$F800	$FFFF	EEPROM	Monitor Code

Anhang

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:

ID	LEN	ADDR	DATA	CS	<EOL>

Das Feld ID gibt den S-Record Typ an. Relevant sind die Typen "S1", "S9" und gelegentlich "S0" (Kommentarrecord). Außer dem ID Feld bestehen alle weiteren Felder 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 CS).

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 CS ist eine Prüfsumme enthalten. Sie wird gebildet aus den Werten der Zeichenpaare der Felder LEN, ADDR und DATA. CS ist das (niederwertigste Byte des) Einerkomplement der Summe aller 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 Prüfsummenfeld.

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, daß die Adresse des ersten geladenen Codebyte gleichzeitig die Startadresse des Programms ist. Ein typischer S9-Record sieht wie folgt aus:

S9	03	B600	46