EPROM-Emulator V1.1

EMU

Stand: 2.4.92

Dies Handbuch soll die Anwendung des EPROM-Emulators unterstützen und die Funktionsweise der Schaltung erklären. Sollte Ihnen ein Fehler auffallen, so verständigen Sie bitte den Verfasser:

Dipl.-Ing. Michael Schmidt
analoge und digitale Elektronik

Aureliusstr.. 22
52072 Aachen

Tel.: 02 41/ 2 05 22
Fax.: 02 41/ 40 89 58

Inhaltsverzeichnis

Funktionsbeschreibung.....1 ff

Befehlserweiterung.....6 ff

Steckverbindungen.....11

Bauteileliste.....12 f

Selektionsanleitung für BS 170.....14

Literaturliste.....15

Anhang:

Schaltbild

Timingdiagramm der Schaltung

Timing am Druckerport

Bestückungsplan

Optionales Verbindungskabel

Funktionsbeschreibung EPROM-Emulator V1.1

Zielsetzung:
Assemblerprogramme funktionieren selten auf Anhieb. Das Beschreiben von EPROMs dauert jedoch sehr lange und das Löschen noch viel länger. Um die Erprobungszeiten in einem vernünftigen Rahmen zu halten liegt es nahe die EPROMs in der Testphase einer Software durch ein statisches RAM zu ersetzen. Der Download geht flotter und ein RAM muß vorher nicht gelöscht werden. Außerdem wird einem RAM vom häufigen Beschreiben nicht schlecht, im Gegensatz zu den begrenzten Brennzyklen bei EPROMs.

Einige Prozessorkarten bieten die Möglichkeit ein Assemblerprogramm per serieller Schnittstelle vom Entwicklungssystem zu empfangen, in ihrem Speicher abzulegen und danach abzuarbeiten. Das ist bei den 8051-Prozessorderivaten nicht ganz unkompliziert, da Programm und Datenspeicher nicht identisch sind. In den Programmspeicher können die 8051-Derivate nicht hineinschreiben, und Assemblerprogramme im Datenspeicher können sie nicht ausführen. Es gibt zwar doch eine Möglichkeit, aber nur mit mehr Aufwand an Hardware und unter Einschänkung der Speicherkapazität. Man kann z.B. einen Teil des Datenspeichers auch als Programmspeicher freigeben (die verschiedenen Freigabesignale verknüpfen) oder die CS-Leitung eines RAMs nach dem Download umschalten, so daß es dann ausschließlich im Codebereich liegt. Im Dauereinsatz des Rechners hat man dadurch unbenutzte Schaltungsteile, die bestenfalls zu Störungen führen. Dazu wäre auch immer ein Monitorprogramm im Zielsystem von nöten, welches den Transfer unterstützt und gegebenenfalls gepackte Übertragungsformate (Intel-Hex o.ä.) aufbereitet. Auch im Entwicklungssystem benötigt man ein Übertragungsprogramm zur Steuerung der Schnittstelle.

Hier soll nun ein anderer Weg beschritten werden: Die Daten werden über eine Centronics-compatible Druckerschnittstelle in den Emulator übertragen. Dadurch kann jedes Entwicklungssystem, egal ob IBM, Atari, Amiga oder wie sie alle heißen, gleichermaßen benutzt werden. Die Betriebssysteme unterstützen die Übertragung, so daß kein spezielles Programm, und damit keine Anpassung, mehr nötig ist. Drucken darf man auch alles, also auch binär codierte Maschinenspracheprogramme.

Das Prinzip dieses Emulators wurde in den Artikeln [1] und [2] vorgestellt. Der Schwerpunkt dieser Entwicklung liegt eher auf der Handhabung: Der Emulator steckt im Bussystem neben der Rechnerkarte und fliegt nicht auf dem Arbeitstisch umher. Nach dem Download eines Programms wird automatisch ein Reset im Bussystem ausgelöst. Zusätzlich ist der Emulator batteriegepuffert, so daß die zu testenden Programme nicht nach jedem Einschalten des Systems neu geladen werden müssen. Die Speicherkapazität wurde speziell für unsere Rechnerkarte 8052-ECB (BasiControl) ausgelegt. EPROMs mit 8, 16 oder 32 kByte Speicherkapazität und einer Datenbusbreite von acht Bit werden emuliert. Als Speicher ist ein statisches RAM vom Typ 62256 oder 43256 mit einer Organisation 32k * 8Bit eingesetzt. Es lassen sich EPROMs der Typen 2764/128 und 27256 simulieren. Für noch kleinere Typen (die hoffentlich keine Verwendung mehr finden) sind die entprechenden Adreßleitungen auf logisch Null zu legen. Auf die Simulation größerer Bausteine wurde hier verzichtet, da mit 32 kByte Programmspeicher wohl alle 8-Bit-Controlleranwendungen abgedeckt sind. Auch der Betrieb als Stand-alone-Gerät ist möglich. Die Stromversorgung erfolgt in diesem Fall mit einem Steckernetzteil; ein Stabi ist auf der Platine vorhanden.

Schaltungsbeschreibung:
Die Schaltung des Emulators läßt sich in folgende Funktionsblöcke aufteilen: Zunächst soll sie das Timing eines Druckers nachahmen. Dies erledigt ein Monoflop des IC1 und ein D-Flip-Flop von IC2. Die Daten gelangen über den 8-Bit-Zwischenspeicher IC6 in das RAM IC7, den zentralen Baustein der Schaltung. Die Adresse, unter der das jeweilige Bit im Speicher landet, kommt aus den Zählerbausteinen IC4 und IC5. Nach jedem /STROBE-Impuls (Schreib-Aufforderung aus dem Computer an den Drucker) inkrementiert der Zählerstand. Das nächste Bit landet dann unter der nachfolgenden Adresse. Dieser Vorgang wiederholt sich, bis alle Daten im RAM liegen. Das zweite Monoflop erkennt das die /STROBE-Impulse ausbleiben und schaltet die Zählerausgänge und das Zwischen-speicher-FF IC6 in den hochohmigen Zustand. Die Daten- und Adreßleitungen des RAMs werden nun vom Prozessor des Zielsystems gesteuert. Außerdem geht die RESET-Leitung am EC-Bus wieder auf logisch Eins. Einige Millisekunden später startet der Zielrechner sein neues Programm. Zur Entkopplung von Rechner und Steuerschaltung dienen die Daten- und Adreßtreiber IC8, IC9 und IC10.

Im Detail:
Der Schaltplan (im Anhang) soll nun etwas genauer untersucht werden. Legt man Betriebsspannung an die Schaltung, so erzeugt die Spannungsüberwachung IC11 ein Low-Signal am /Reset-Ausgang (Pin 5). Das Signal setzt die beiden D-FFs in den aktiven Zustand, und die Monoflops zurück. Leitung D aus IC2 trennt die Zählerbausteine IC4 und IC5 vom RAM und sperrt die Ausgangsleitungen des Datentreibers IC6. Leitung E setzt die Zählerstände auf Null, gibt die Ausgangsleitungen des RAM-Bausteins IC7 frei und ebenso die Adreßtreiber IC8 und IC9. Die Schaltung kann nun ein EPROM emulieren.

Will der Zielrechner das Programm lesen, so legt er die Adressen auf den EPROM-Sockel. Über die Puffer IC8 und IC9 gelangen diese an das RAM. Sind die Adressen gültig gehen /OE und /CS am Sockel auf Null. IC3 gibt daraufhin den Datentreiber IC10 frei. Die Daten unter der vorgegebenen Adresse kommen aus dem RAM zum Rechner. Die Nummerierung der Anschlüsse im Schaltplan entspricht der Reihenfolge am EPROM-Sockel. Je nachdem welche EPROM-Typen emuliert werden, sind die Jumper J3 und J4 zu setzen. Sie trennen die jeweils nicht benötigten Adreßleitungen ab und legen die Eingänge am RAM auf Low. Die Tabelle gibt die jeweilige Position an:

EPROM-Typ	Kapazität	Jumper (Funktion des Pins)	
27256	32kByte	J3-A14	J4-A13
27128	16kByte	J3-GND (/PGM)	J4-A13
2764	8kByte	J3-GND (/PGM)	J4-GND (NC)

Die Angaben in Klammern sind die Anschlußfunktionen der Pins bei kleineren EPROM-Typen.

Timing am Druckerport:
Zum Laden eines neuen Programms in den Emulator legt der Computer jeweils ein Datum auf den Druckerport und zieht die /STROBE-Leitung für kurze Zeit auf Null. Das /STROBE-Signal ist die Aufforderung an einen Drucker die Daten zu akzeptieren. Der Computer erwartet mit der fallenden Flanke von /STROBE eine Bestätigung über die Leitung BUSY. Ein Drucker signalisiert damit seine Tätigkeit. Manche Rechner erwarten nach der ansteigenden Flanke von /STROBE noch eine weitere Bestätigung auf der Leitung /ACKNOWLEDGE. Legt der Drucker, oder hier der Emulator, BUSY wieder auf logisch Null, sendet der Computer das nächste Byte. Der Vorgang wiederholt sich bis alle Daten in den Emulator übertragen sind.

Das obere Monoflop von IC1 wird durch ein Low-Signal auf der /STROBE-Leitung gestartet und mit jedem weiteren Impuls nachgetriggert. Erst wenn zwei Sekunden lang keine Impulse mehr eintreffen, das Programm also übertragen ist, fällt es zurück. Leider begeben sich die Ausgänge des Monoflops nicht schnell genug in den aktiven Zustand. Deshalb ist das D-Flipflop (IC2 oben) nachgeschaltet. Durch den ersten /STROBE-Impuls wird es sehr schnell zurückgesetzt. Erst wenn das Monoflop abfällt steuert die Leitung A das FF wieder auf Eins. Die Leitung D aktiviert die Zählerausgänge, gibt die Zähler frei und auch den Ausgang des Datentreibers IC6. Der Ausgang E des Flipflops lößt mit dem Transistor V3 einen Reset auf dem EC-Bus aus und trennt die Adreßleitungen des Rechners vom RAM (IC8 und IC9). Während dieser Zeit bleibt die grüne LED dunkel.

Das zweite Monoflop und das D-Flipflop (IC2 unten) erzeugen die Signalfolge am Druckerport. Die fallende Flanke von /STROBE setzt das D-FF zurück, so daß F auf Null und BUSY auf Eins gehen. Die Leitung G wird Eins. Damit speichert das Puffer IC6 die Daten und die ICs 4 und 5 übertragen den aktuellen Zählerstand (noch Null) an die integrierten Ausgangslatches. Die zu beschreibende Adresse liegt am RAM-Baustein. Solange G gleich Eins ist leuchtet die rote LED. Die ansteigende Flanke von /STROBE triggert das Monoflop. Der Ausgang B steuert die /ACKNOWLEDGE-Leitung für 1us auf Null-Pegel. Leitung C steuert /WE am RAM-Baustein und speichert das Datum ab. Fällt das Monoflop zurück, entsprechend einer positiven Flanke auf C, geht BUSY auf Null und der Adreßzähler inkrementiert. Der Ablauf wiederholt sich bis alle Daten gespeichert sind. Der Signal-Zeit-Plan (im Anhang) stellt die Vorgänge schematisch dar.

Tröstlich, daß die Schaltung trotz dieser komplizierten Beschreibung funktioniert.

Pufferschaltung:
Schaltet man das Netzteil aus, so übernimmt der Akku die Stromversorgung für den Speicher. Die Freigabeleitung /CS des RAM-Bausteins soll zum Datenerhalt auf logisch Eins liegen. Der "Resetgenerator" TL 7705 (IC11) überwacht die Betriebsspannung und gibt ein Low-Signal an das Gate des FETs V4 sobald die Spannung kleiner als etwa 4,7 V ist. Damit werden, wie oben beschrieben, die Monoflops zurückgesetzt und die Verbindung von /CS nach Masse aufgetrennt. Legt man die Spannung wieder an, so bleibt dieser Zustand noch etwa 130 ms bestehen, bis alle ICs einen stabilen Betriebszustand erreicht haben. Leider kann der Resetgenerator den Low-Pegel nur bis etwa 2V Betriebsspannung halten, darunter ist der Ausgang /RESET (Pin 5) undefiniert. Für eine einwandfreie Funktion der Batteriepufferung ist es notwendig, daß der Transistor BS 170 bei einer Gate-Source-Spannung von etwa 1,8 V noch sperrt (experimentell ermittelt). Laut Datenblatt des V-MOS-Transistors sollte bei verbundener Gate-Drain-Strecke die Spannung Gate-Source größer 2V sein, womit die Schaltung einwandfrei funktioniert. Dieser Parameter ist leider großen Fertigungsschwankungen unterworfen. Wenn Sie einen Bausatz oder eine Leerplatine erworben haben müssen Sie den Transistor selektieren. Dazu dient die Schaltung im Anhang; mit ein paar Prüfklemmen ist das schnell realisiert. Die Spannung U_M darf dabei nicht unter 1,8V betragen.

Der Akku puffert das RAM über den Widerstand R13 und dem Jumper J1. Im Betrieb des Rechners erfolgt die Ladung des Akkus über den gleichen Zweig. Verwendet man einen kleineren Akku, so ist der Widerstand entsprechend zu vergrößern. Der Ladestrom ist auf etwa Q/100 eingestellt. Für eine andere Kapazität berechnet sich der Vorwiderstand R13 zu:

$R3 = (4{,}3V - U_{AKKU}) * 100/Q$

Verwendet man eine Lithiumbatterie, muß der Jumper J1 unbedingt entfallen. Der Ladestrom würde anderenfalls die Batterie zerstören. Die Versorgung des RAM-Bausteins erfolgt bei Batteriepufferung über die Diode D4 und den Jumper J2.

Stand-alone:
Für den Betrieb außerhalb des Rahmens ist ein Spannungsregler IC12 vorgesehen. An die Lötstifte ST4 kann man ein Steckernetzteil anschließen. Es sollte eine Gleichspannung größer 9V liefern. Die beiden Dioden D1 und D2 entkoppeln den Reglerausgang solange die Spannung vom EC-Bus anliegt.

Die ganze Platine kann man in einem Kunststoffgehäuse montieren (Innenmaß > 100 mm * 172 mm!). Die beiden LEDs und die Stiftleiste ST3 befestigt man dazu auf der Lötseite, so daß sie durch geeignete Aussparungen im Gehäuse zugängig sind. Die 25polige D-Sub-Verbindung führt man seitlich aus dem Kästchen. Das Resetsignal steht auch auf der Stiftleiste ST3 (der vorletzte Pin, Außenseite, ohne Nummer) zur Verfügung, so daß man auf den Busanschluß verzichten kann.

Verbindung zum Rechner:
Die Controllerkarte BasiControl kann über den EC-Bus keinen Programmcode lesen. Dies ist bewußt so ausgelegt da der Speicherplatz auf der Karte eigentlich groß genug ist. Zwangsläufig muß der Emulator über ein Flachband-kabel mit dem Rechner verbunden werden. Das Kabel sollte so kurz wie möglich, auf keinen Fall länger als 30cm, sein. Hier gibt es jetzt zwei Möglichkeiten:

Die Stiftleiste auf der Emulatorkarte steckt auf der Bestückungsseite. Auf das Gegenstück, den Pfostenstecker, quetscht man das Flachbandkabel und an das andere Ende einen IC-Fassungsverbinder. (Die Dinger heißen wirklich so! Immer wenn ich im Lade stehe habe ich den Namen wieder vergessen: Ich hätte gern so'n Stecker für ICs, Sie wissen schon...) Da der IC-Verbinder, mit 28 Polen, schmaler ist als der Pfostenstecker, mit 34 Polen, muß man einige Leitungen des Flachbandkabels abschneiden und zwar auf der richtigen Seite. (28polige Pfostenverbindungen gibt es leider nicht.) Verwendet man Stiftleisten mit Wanne und Auswurfhebeln läßt sich das Kabel nicht falsch herum aufstecken. Leider schaut der Pfostenverbinder im Rahmen nicht auf die Bestückungsseite der Rechnerplatine. Das Flachbandkabel muß also unter den Führungsschienen des Rahmens durchgefädelt werden. Das geht, ist aber etwas unbequem.

Alternativ montiert man die Stiftleiste auf der Lötseite der Emulatorplatine. Das oben beschriebene Kabel hier bitte nicht verwenden, es vertauscht die linken und rechten Anschlußreihen am IC, also Pin 1 mit Pin 28 usw. Der Adapter, entsprechend der Zeichnung im Anhang, steckt anstelle des EPROMs im Rechner und nimmt oben einen Pfostenstecker auf. Auf der Lochstreifenplatine werden die Leiterbahnen in der Mitte, zwischen zwei Lötaugen, aufgetrennt. Auf die Bestückungsseite lötet man eine 26polige Stiftleiste, am Besten mit Wanne und Auswurfhebeln. Von der Lötseite aus steckt man dann zwei IC-Steckadapter-leisten in die Platine (das sind einreihige Präzisions-Fassungen, mit Stiften auf beiden Seiten). Die Höhe des Adapters kann man noch verändern indem man einige IC-Sockel aufsteckt. Hier wurde eine 26polige Wanne verwendet, da die Konstruktion sonst sehr breit über den Sockel übersteht und Pin 1 und 28 des EPROMs ohnehin nicht angeschlossen sind. Das Ganze ist recht aufwendig zu fertigen, bietet aber auch die kürzeste Verbindung zum Rechner und ist leicht wieder aufzutrennen. Das Verbindungskabel sollte so kurz wie möglich sein, die kapazitive Last an den Prozessorpins wird sonst zu groß.

Download:
Zum Testen des Emulators kann man z.B. das unten angegebene Programm assemblieren. Es wurde mit dem Shareware-Programm TASM erstellt. Verwendet man einen anderen Crossassembler sind evtl. einige Anpassungen im Quellcode vorzunehmen. Die Programme produzieren häufig Intel-Hex-Formate als Defaultwert. Es sind die entsprechenden Einstellungen vorzunehmen damit eine Binärdatei geschrieben wird. Die Datei muß nun in den Emulator. Wie schon beschrieben wird sie einfach gedruckt. Bei DOSen überträgt "COPY [Dateiname] LPT1: /B" die Datei in binärer Form zum Emulator. Auf einem Atari reicht es die Datei anzuklicken und "Drucken" auszuwählen. Ein Atari hängt an das Ende der Datei noch CR und LF an, so daß eine 32 kByte Binärdatei um zwei Bytes gekürzt werden muß. Wie das Drucken bei anderen Rechnern geht entnehmen Sie bitte dem jeweiligen Handbuch.

Bitte laden Sie den Emulator nur, wenn alles, der Rechner und der Emulator, eingeschaltet ist. Man weiß sonst nie so genau ob der Centronics-Port überlebt. Soll das Programm gepuffert werden, so ist vor dem Ausschalten des Entwicklungssystems entweder der Centronicsstecker zu ziehen oder es muß das Zielsystem zuerst abgeschaltet werden. Der sonst resultierende Low-Pegel auf der /STROBE-Leitung würde einen Ladezyklus auslösen und das Programm, oder zumindest das erste Byte, überschreiben. Verwendet man die Controllerkarte 8052-ECB, so lassen sich die einzelnen Bytes zur Kontrolle anzeigen. Der Befehl: PH0. XBY(Adresse) gibt hexadezimal den unter der Adresse gespeicherten Wert auf dem Terminal aus.

Beispiel für eine Befehlserweiterung des MCS BASIC-52

Das abgedruckte Maschinenprogramm soll verdeutlichen, wie man eigene Basicbefehle erstellt. Das Hauptprogramm PRJ.ASM bleibt immer gleich, es teilt dem Interpreter die Existenz der neuen Befehle mit. Als Beispiel ist hier ein Kopierbefehl abgedruckt. Die Befehlserweiterung COPY [xxxxh],[yyyyh],[zzzzh] kopiert Daten von der Adresse [xxxxh] zur Adresse [yyyyh] und alle folgenden Daten in aufsteigender Reihenfolge. Die Größe des Bereichs wird durch [zzzzh] angegeben ([zzzzh] = 1 kopiert ein Datum). Angegeben werden die Adressen des 8052 und nicht die EC-Busadressen. Es kann nur im Datenspeicher kopiert werden. Mit dem neuen Befehl ist es möglich Memory-cards oder andere externe Speicher, wie Diskettenlaufwerke zu verwenden und auch Basicprogramme dort abzulegen. In dem Programm werden einige Maschinenroutinen des 8052-Basic genutzt. Dazu ist jeweils die laufende Nummer (OPBYTE) der Routine in den Akku zu laden und mit LCALL 30h das Unterprogramm zu starten. Genauere Informationen zu Kopplung der Basicerweiterungen entnimmt man am Besten aus [3].

```
;    PRJ.ASM   Projektdatei zum Einbinden diverser
;    Basicerweiterungen.               9.5.91
;
;-------------------------------------------------------
;    Damit ein EPROM-Programmierer oder Emulator ein
;    vollständiges File von 16KByte Länge erhält
;    ist es nötig, daß die erste und letzte Adresse
;    auftaucht:
;
          .ORG 0000h
;
;-------------------------------------------------------
;    Um die Special Funktion Register des 8051 zu
;    nutzen müssen zusätzliche Mnemonics aus der
;    Datei 8051.H eingebunden werden:
;
#include  "8051.H"
;
;-------------------------------------------------------
;    Wie sag ich's meinem Rechner?
;
;    Steht im Programmspeicher an der Adresse 2002h
;    das Datum 5Ah so erwartet der Basicinterpreter
;    eine Befehlserweiterung. Das stellt er wohl
;    schon beim Einschalten fest...
;
BAS_INIT  .ORG 2002h
          .DB 5Ah
;
;-------------------------------------------------------
;    Der Interpreter führt das Programm an der
;    Adresse 2048h im Codespeicher aus. Die Routine
;    setzt das Bit 45, das Option-Bit. (Adresse
;    2Dh, das 5te Bit des internen bitadressierbaren
;    Speichers Adresse 37 dez.) Damit erwartet der
;    Basicinterpreter eine Befehlserweiterung.
```

```
;
S_OPT_BIT .ORG 2048h
          setb 45
;
;    und zurück:
;
          ret
;-------------------------------------------------------
;    Nachdem Bit 45 gesetzt ist, testet der Inter-
;    preter beim Tokenisieren einer Zeile jedesmal
;    den Lookup table. Dessen Adresse wird in einem
;    Programm an der Adresse 2078h in den Data-
;    Pointer (DPTR) geladen:
;
          .ORG 2078h
          mov  DPTR, #LOOKUP_T
          ret
;
;-------------------------------------------------------
;    Die Adresse für das Label LOOKUP_T wird im
;    folgenden festgelegt:
;
LOOKUP_T  .ORG 2100h
;
;    Die verfügbaren Tokens sind 10h bis incl.
;    1Fh. Die Tabelle beginnt mit dem Token
;    (Nummer), dem dazugehörigen Namen (Befehl,
;    ASCII) und einer Null (00h) als Zeichen, das
;    weitere Tokens folgen. Der letzte Name wird
;    mit 0FFh abgeschlossen. Die Reihenfolge ist
;    beliebig.
;
          .DB 10h        ;Token
          .DB "COPY"     ;Blockkopie
;
;         .DB 00h        ;Terminierung
;         .DB 11h        ;naechstes Token
;         .DB "Befehl"   ;usw.
;
          .DB 0FFh       ;Ende der Tabelle
;
;-------------------------------------------------------
;    Nun kennt 8052-Basic die neuen Tokens, nimmt
;    die Befehle an und gibt sie im Listing auch
;    wieder aus.
;    Nur die Ausführung hapert noch...
;
;    Möchte der Interpreter unseren neuen
;    Befehl ausführen (nach run oder im Command-
;    Modus), so testet er zuerst das Bit 45 (Option-
;    Bit, siehe oben), dann wird ein Programm an
;    der Adresse 2070h ausgeführt, welches die
;    Adresse der Vektor Tabelle übergibt. Die
;    Adresse wird in den DPTR geschrieben.
;
          .ORG 2070h
          mov DPTR,#VEKTOR_T
          ret
;
```

```
;-------------------------------------------------------
;    Die Vektor Tabelle wiederum besteht aus
;    den Einsprungadressen für die Basic-
;    erweiterungen (Assemblerroutinen). Die
;    Reihenfolge hier muß mit der in dem Lookup-
;    table übereinstimmen!
;
VEKTOR_T  .ORG 2200h
;
;    Diese Adresse wurde wieder frei gewählt,
;    und dort steht:
;
          .MSFIRST
;    in der richtigen Reihenfolge (MSB-LSB)
;
          .DW COPY
;
          .LSFIRST
;    ...sonst stimmt nichts mehr
;
;-------------------------------------------------------
;    Nun kommt der eigentliche Code für die
;    Basicerweiterungen:
;         (das wurde auch Zeit...)
;
;-------------------------------------------------------
;    Copieren eines Datenblocks:
;
COPY
#include "COPY.ASM"
;
;-------------------------------------------------------
;    Ende des Quälcodes und auffüllen auf
;    16KByte Länge:
;
          .ORG 3FFFh
;
          .END


;    COPY.ASM                      13.8.91
;    Basicerweiterung zum Copieren von Datenbereichen
;
;    COPY [xxxxh],[yyyyh],[zzzzh] kopiert Daten von
;    der Adresse [xxxxh] zur Adresse [yyyyh] und alle
;    folgenden Daten in aufsteigender Reihenfolge.
;    Die Größe des Bereichs wird durch [zzzzh] an-
;    gegeben. ([zzzzh] = 1 kopiert ein Datum)
;    Angegeben werden die Adressen des 8052 und nicht
;    die Busadressen. Es kann nur im Datenspeicher
;    kopiert werden.
;
;    Hole den Wert des nachfolgenden Ausdrucks auf den
;    Argumentstack
          mov A,#39h     ;evaluiert expr.> Arg.st.
          lcall 30h
;
```

```
;    Der nächste Ausdruck sollte ein Komma sein
;    ',' = 2Ch
          mov A,#64
          lcall 30h
          cjne A,#2Ch,ERROR_03
;
;    Hole den Wert des zweiten Ausdrucks auf den
;    Argumentstack
          mov A,#39h     ;evaluiert expr.> Arg.st.
          lcall 30h
;
;    Der nächste Ausdruck sollte ein Komma sein
;    ',' = 2Ch
          mov A,#64
          lcall 30h
          cjne A,#2Ch,ERROR_03
;
;    Hole den Wert des dritten Ausdrucks auf den
;    Argumentstack
          mov A,#39h     ;evaluiert expr.> Arg.st.
          lcall 30h
;
;    Es soll xxhigh in Reg. 18h
;            xxlow          19h
;            yyhigh         1Ah
;            yylow          1Bh
;            zzhigh         1Ch
;            zzlow          1Dh
;            DPTR high      83h
;            DPTR low       82h
;
;    Integer von letztem Wert [zzzzh] in R3:R1
;
          mov A,#01h
          lcall 30h
;
;    R3 nach zzhigh, R1 nach zzlow
          mov 1Ch,R3
          mov 1Dh,R1
;
;    Integer von vorletztem Wert [yyyyh] in R3:R1
;
          mov A,#01h
          lcall 30h
;
;    R3 nach yyhigh, R1 nach yylow
          mov 1Ah,R3
          mov 1Bh,R1
;
;    Integer von erstem Wert [xxxxh] in R3:R1
;
          mov A,#01h
          lcall 30h
;
;    R3 nach xxhigh, R1 nach xxlow
          mov 18h,R3
          mov 19h,R1
;
```

```
;    Schleifenzähler testen
C_LOOP    mov A,1Dh ; zzlow
          jnz NOTZERO
          mov A,1Ch ; zzhigh
          jz READY
;
;    Schleifenzähler decrementieren
NOTZERO   dec 1Dh ; zzlow
          mov A,1Dh
          cjne A,#0FFh,CARRY_LC
          dec 1Ch ; zzhigh
;
;    Kopieren:
;    Quelladr. in DPTR
CARRY_LC  mov 83h,18h
          mov 82h,19h
;
;    Wert in Akku
          movx A,@DPTR
;
;    Zieladr. in DPTR
          mov 83h,1Ah
          mov 82h,1Bh
;
;    Akku ins Ziel
          movx @DPTR,A
;
;    Incrementieren der Quell- und Zieladresse:
          inc 19h ; xxlow
          mov A,19h
          jnz CARRY_1
          inc 18h ; xxhigh
CARRY_1   inc 1Bh ; yylow
          mov A,1Bh
          jnz CARRY_2
          inc 1Ah ; yyhigh
CARRY_2   sjmp C_LOOP
;
READY     ret
;
ERROR_03
;    Fehlermeldung: gibt an Console aus
;    CR,MSG_03 und geht zum Command-Modus
          mov A,#07h          ;CR
          lcall 30h
          mov R3,# (MSG_03 & 0FF00h) / 0100h
          mov R1,#  MSG_03 & 00FFh
          setb 52             ;Msg. im ROM
          mov A,#06h          ;Msg. ausgeben
          lcall 30h
          clr A               ;Command-mode
          ljmp 30h
;
MSG_03    .DB "COPY [xxxxh],[yyyyh],[zzzzh]"
          .DB 22h
;
```

Steckverbindungen EMU V1.1

EC-Busanschluß 64pol. Messerleiste ST1

Pin 1a/c	+5V	
Pin 31c	/RESET	Ausgang
Pin 32a/c	GND	

Ext. Betriebsspannung ST4
Lötstifte ohne Nummer

+ 9V	
GND	

Druckerport (Centronics) 25pol. D-Sub Stecker ST2

Pin 1	/Strobe	Steuerleitung
Pin 2	D0	Datenleitungen
Pin 3	D1	Pin 4 D2
Pin 5	D3	
Pin 6	D4	
Pin 7	D5	
Pin 8	D6	
Pin 9	D7	
Pin 10	/Acknowledge	Steuerleitung
Pin 11	Busy	Steuerleitung
Pin 18		
... 25	GND	Masse

EPROM-Anschluß (Stiftleiste) ST3

Pin 2	A12	Adreßleitung
Pin 3	A7	
Pin 4	A6	
Pin 5	A5	
Pin 6	A4	
Pin 7	A3	
Pin 8	A2	
Pin 9	A1	
Pin 10	A0	
Pin 11	D0	Datenleitung
Pin 12	D1	
Pin 13	D2	
Pin 14	GND	Masse
Pin 15	D3	
Pin 16	D4	
Pin 17	D5	
Pin 18	D6	
Pin 19	D7	
Pin 20	/CS	Freigabeleitung
Pin 21	A10	
Pin 22	/OE	Freigabeleitung
Pin 23	A11	
Pin 24	A9	
Pin 25	A8	
Pin 26	A13/NC	nur 27128 und 27256
Pin 27	A14//PGM	nur 27256
ohne Nr.	/RESET	Ausgang

Bauteileliste für EPROM-Emulator V1.1

1*		Platine EPROM-Emulator V1.1
1*	ST1	Messerleiste DIN 41.612 Bauform C, 64polig, gewinkelt, a+c bestückt (DIN BEZ.: C64M-C1A)
1*	ST2	D-Sub.-Stiftleiste, 25 polig (DB25), gewinkelte Lötstifte
1*		Befestigungswinkel zu D-Sub., 25 pol. (Kunstofformteil)
2*		Adapter-Gewindebolzen UNC 4-40/M3-05/08
1*		Kunststoffwinkel für Frontplatte (Isert)
1*		Kühlkörper für Spannungsregler, liegend, TO 220, ca. (35 * 17)mm max. Grundfläche
5*		Zylinderkopfschraube M3*8
7*		Mutter sechskant M3
2*		Zahnscheibe M3
2*		Zylinderkopfschraube M2,5*10
2*		Zylinderkopfschraube M2,5*8
4*		Mutter sechskant M2,5
1*		Stiftleiste gerade 2,54mm zweireihig 6 polig
1*		Stiftleiste gerade 2,54mm zweireihig 4 polig
2*	ST4	Lötstift 1mm
4*	J1..4	Jumper
1*	ST3	Stiftleiste gerade 2,54mm zweireihig 34 polig mit Auswerfern, stehende Montage
1*		Pfostenstecker (Federleiste) 34 polig für Flachbandkabel
1*		Flachbandkabel 10...30 cm, 28 polig
1*		IC-Sockel Steckverbinder (IC-Fassungverbinder) 28 polig
1*	RN1	Widerstandsarray SIL 8* 10k
1*	R9	Widerstand Metallfilm 270R
1*	R6	Widerstand Metallfilm 560R
1*	R3	Widerstand Metallfilm 1k
1*	R13	Widerstand Metallfilm 1k3
7*	R1,R5,R8,R11, R12,R15,R16	Widerstand Metallfilm 10k
3*	R4,R7,R10	Widerstand Metallfilm 22k
1*	R14	Widerstand Metallfilm 100k
1*	R2	Widerstand Metallfilm 1M
1*	C2	Keramikkondensator 1n5 RM2,54/5,08
11*	C3,C5,C7,C8, C10,C11, C13...C17	Keramikkondensator u1 RM2,54
1*	C1	Elko 4u7 10V RM2,54
4*	C4,C6,C9,C12	Elko 10u 16V RM2,54

1*	D3	Diode 1N4148 o.ä.	
2*	D1,D2	Diode 1N4007 o.ä.	
1*	D4	Schottky-Diode SD 101 o.ä.	
1*	L2	LED rot 5mm	
1*	L1	LED grün 5mm	
3*	V1...V3	Transistor NPN BC 547B o.ä.	
1*	V4	V-MOS-FET BS 170, selektiert nach U_{GS}>1,8V für $U_{GS}=U_{DS}$	
1*	IC11	TL 7705	
1*	IC12	7805	
1*	IC3	74LS02	Es können
1*	IC2	74LS74	auch HCT-Typen
1*	IC1	74LS123	verwendet werden!
3*	IC8...IC10	74LS245	
1*	IC6	74LS374	
2*	IC4,IC5	74LS590	
1*	IC7	RAM 43256-150	
1*		Präz.sockel 8pol.	
2*		Präz.sockel 14pol.	
3*		Präz.sockel 16pol.	
4*		Präz.sockel 20pol.	
1*		Präz.sockel 28pol.	
1*		Akku Varta 3,6V 60mAh oder Lithiumbatterie	

Best.nr.:Varta 53306 603 059

optional für Stand-alone Betrieb:

1*		Gehäuse für Europakarte
1*		Steckernetzteil und geeignete Steckverbindung für Betriebsspannung

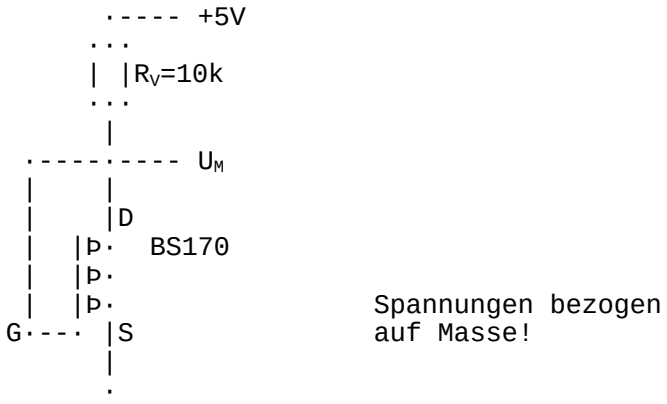
optional für EPROM-Anschluß auf der Platinenunterseite:

	ST3	Die Stiftleiste ST3 wird auf der Lötseite montiert. Der IC-Sockel-Steckverbinder darf dann nicht verwendet werden, Pin 1 und 2 des EPROMs wären vertauscht. Eine Lösung mittels Pfostenverbinder, Lötstreifenplatine und IC-Sockel ist möglich und auf Anfrage fertig lieferbar.

Selektionsanleitung für BS 170

Für eine einwandfreie Funktion der Batteriepufferung ist es notwendig, daß der Transistor BS 170 bei einer Gate-Source-Spannung von ca. 1,8 V noch einwandfrei sperrt. Beim Ein- und Ausschalten der Betriebsspannung erzeugt der Resetgenerator TL 7705 ein Low-Signal am Gate des Transistors. Damit wird die Schaltung zurückgesetzt und der RAM-Baustein deselektiert. Leider kann der Resetgenerator diesen Pegel nicht halten, bis U_B=0V ist. Laut Datenblatt des V-MOS-Transistors sollte bei verbundener Gate-Drain-Strecke die Spannung Gate-Source größer 2V sein, womit die Schaltung einwandfrei funktioniert. Dieser Parameter ist leider großen Fertigungsschwankungen unterworfen.

Sie sollten deshalb den Transistor vor dem Einlöten in der unten angegebenen Schaltung auf die Einhaltung dieses Wertes überprüfen. Die Spannung U_M darf dabei nicht unter 1,8V betragen.



Literaturliste

[1] B.C. Zschocke: EPROM-EMULATOR. ELEKTOR 11/89 Seite 14ff

[2] B.C. Zschocke: EPROM-EMULATOR. ELEKTOR 10/91 Seite 22ff

[3] INTEL: MCS BASIC-52 User's Manual. INTEL, 1989, ISBN: 1555-12-0857

Weiterführende Informationen finden Sie in:

[4] J.P.M. Steeman: 8052 AH-BASIC. Elektor-Verlag, 1989, ISBN: 3-921608-72-4

[5] INTEL: Embedded Controller Handbook, Vol. 1, 8-Bit. INTEL, 1987, ISBN: 1-55512-072-5

[6] VALVO: Die 8bit-Mikrocontroller-Familie 8051, Bd1 Eigenschaften. Verlag Boysen + Maasch, 1984, ISBN: 3-87095-260-1

[7] VALVO: Die 8bit-Mikrocontroller-Familie 8051, Bd2 Befehlsvorrat. Verlag Boysen + Maasch, 1984, ISBN: 3-87095-260-X