ROM-Schaltkreis U2365D45 BM200 ergänzt den UB8830

Gerhard Dugnus, Siegmar Müller
VEB Mikroelektronik „Karl Marx" Erfurt

Zur Unterstützung der Softwareentwicklung mit dem UB8830 wurde im VEB Mikroelektronik „Karl Marx" ein ROM-Baustein U2365D45 BM200 entwickelt.
Mit dessen Hilfe ist es möglich, in einer minimalen Konfiguration Programme für kleinere Steuerungsprojekte direkt am zu steuernden Gerät zu entwickeln und zu testen.

1. Bestandteile und Zweck

Der Einsatz von Mikrorechnern in industriellen Anlagen erfordert den Test von Programmen auch direkt an der zu steuernden Einrichtung. Weil dazu oft kein Entwicklungssystem eingesetzt werden kann, soll der Rechner selbst Werkzeuge zur Testung und eventuellen Korrektur der Software besitzen. Um dem Anwender des UB8830 die Entwicklung eines kleinen, leistungsfähigen Rechners mit diesen Hilfsmitteln zu erleichtern, wurde ein ROM-Bitmuster mit der dafür erforderlichen Software entwickelt. Diese beinhaltet folgende Komponenten:
– Systemmonitor mit Assembler und Disassembler
– Editor/Debugger für Tiny-MPBASIC.

Der Systemmonitor ermöglicht in Verbindung mit dem Assembler/Disassembler die Entwicklung und den Test von Programmen in Maschinensprache, während der Editor/Debugger Entwicklung und Test von Programmen in Tiny-MPBASIC /1/ unterstützt. Weiterhin ist die serielle Kopplung mit einem Wirtsrechner möglich, so daß auch dort Programme entwickelt und EPROMs programmiert werden können.
Das ROM wird u. a. in einem Entwicklungsmodul eingesetzt, welches beim VEB Mikroelektronik „Karl Marx" als unbestückte Leiterplatte nachgenutzt werden kann. Es handelt sich um einen universell einsetzbaren Einplatinenrechner, der einen für diese Geräteklasse sehr hohen Komfort für Programmentwicklung und -test bietet. Beim Bitmuster 200 handelt es sich um eine Arbeitsversion, die eigentlich nur zur ROM-Anprobe diente. Es erwies sich jedoch als so leistungsfähig, daß im Sinne einer breiteren und schnelleren Nutzung auf eine Überarbeitung verzichtet wurde. Deswegen erfolgen im Text Hinweise auf inzwischen bekannt gewordene Fehler und Unzulänglichkeiten.

2. Forderungen an die Hardware

Die Kommunikation mit dem Bediener erfolgt mit einem Datensichtgerät, das über eine (abgerüstete) V.24-Schnittstelle mit dem UB8830 verbunden ist. Die Übertragungsgeschwindigkeit wird beim Empfang des ersten Zeichens (Datenbit 0 dieses Zeichens muß 1 sein – z. B. %0D [Carriage Return]) durch Ausmessen des Startbits automatisch bestimmt. Die Taktfrequenz des UB8830 ist dabei unkritisch, für eine sichere Funktion bis 19200 Baud wird jedoch ein Quarz mit 7,3728 MHz empfohlen. Weiterhin kann über eine zweite serielle Schnittstelle (mit der gleichen Baudrate) ein Wirtsrechner angeschlossen werden (Bild 1a). Dieser ist als Entwicklungssystem und zum Abspeichern der entwickelten Programme nutzbar. Da der UB8830 nur einen seriellen Kanal besitzt, muß zwischen Wirtsrechner und Datensichtgerät umgeschaltet werden. Hierzu sind am Port 2 drei Steuersignale vorhanden:

– P20 = high: SIO mit Datensichtgerät verbunden (Terminal Mode)
– P21 = high: SIO mit Wirtsrechner verbunden (Host Mode)
– P22 = high: Datensichtgerät mit Wirtsrechner verbunden (Transparent Mode).

Wenn der Wirtsrechner mit einer eigenen Bildschirmsteuerung ausgestattet ist (z. B. BC A 5120, PC 1715), wird er gemäß Bild 1b mit dem UB8830-System verbunden. Die Signale an P20 bis P22 werden dann mittels Software ausgewertet.
Bild 2 zeigt die Schaltung eines Minimalsystems mit dem U2365D45 BM200. Benötigt werden neben dem UB8830 und der Anpassung an die serielle Schnittstelle noch 27 Byte RAM ab Adresse %800 und weitere ca. %100 Byte Arbeitsspeicher, die sich das ROM-Programm bei der Initialisierung von der Adresse %DFFF ab nach unten selbst sucht. Dazwischen liegt der Anwenderspeicher (auf dem Entwicklungsmodul extern erweiterbar). Die STOP-Taste (Taste nach Masse an P33) dient zum Anhalten von BASIC-Programmen und zum Verlassen des Transparent-Mode.

3. Beschreibung der ROM-Softwarekomponenten

3.1. Systemmonitor

Der Systemmonitor beinhaltet die zum Testen von Maschinenprogrammen erforderlichen Kommandos. Alle Kommandos können mit dem groß gedruckten Buchstaben abgekürzt werden.

Display ⟨adr⟩ [⟨num⟩]
Ausgabe von ⟨num⟩ Byte ab Adresse ⟨adr⟩. Wird ⟨num⟩ nicht angegeben, dann wird der Inhalt der Speicherstelle ⟨adr⟩ angezeigt und in den Eingabemodus gegangen. Danach kann entweder ein neuer Wert zum Abspeichern, RETURN zum Anzeigen des nächsten Bytes, '∧' zum Anzeigen des vorhergehenden Bytes oder 'Q' zum Verlassen des Eingabemodus eingegeben werden.

Register [⟨nr⟩[⟨wert⟩]]
Anzeigen bzw. Modifizieren von Registerinhalten. In das Register ⟨nr⟩ kann der Wert ⟨wert⟩ eingeschrieben werden. Ohne Angabe eines Wertes wird der Inhalt von Register ⟨nr⟩ angezeigt und kann wie bei Display modifiziert werden. Wurde keine Registernummer eingegeben, zeigt das Programm den aktuellen Arbeitsregistersatz.

Move ⟨adr1⟩ ⟨adr2⟩ ⟨len⟩
Kopieren von ⟨len⟩ Bytes von ⟨adr1⟩ nach ⟨adr2⟩. Move transportiert die Daten immer beginnend beim ersten Byte von ⟨adr1⟩ nach ⟨adr2⟩. Wenn ⟨adr1⟩ größer als ⟨adr2⟩ ist und beide Bereiche sich überlappen, funktioniert dieser Befehl nicht.

Bild 1 Kopplung mit einem Wirtsrechner
a) mit abgesetztem Terminal
b) mit eingebauter Bildschirmeinheit

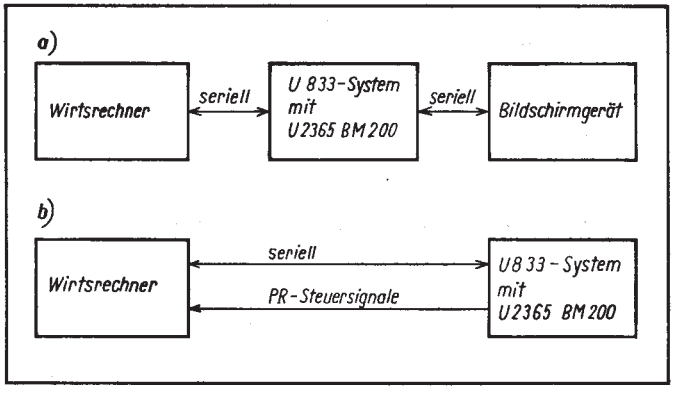
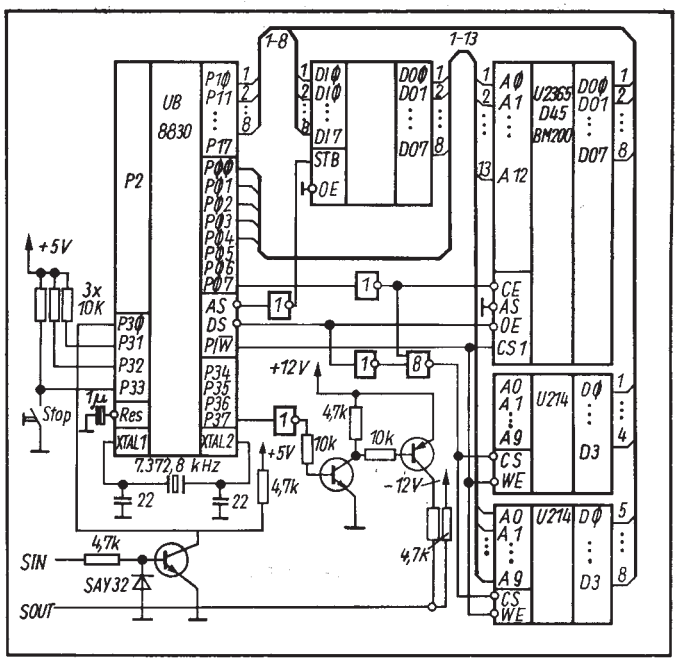


Bild 2 Minimalsystem mit U2365 D45 BM200



Compare ⟨adr1⟩ ⟨adr2⟩ ⟨len⟩
Vergleichen von ⟨len⟩ Bytes ab ⟨adr1⟩ mit ⟨len⟩ Bytes ab ⟨adr2⟩. Alle unterschiedlich belegten Speicherzellen werden angezeigt.

Jump [⟨adr⟩]
Setzen des Befehlszählers PC auf den Wert ⟨adr⟩ bzw. Anzeigen des aktuellen Wertes.

Break [⟨adr⟩]
Setzen des Unterbrechungspunktes auf Adresse ⟨adr⟩ bzw. Löschen. Der Unterbrechungspunkt darf nur auf das erste Byte eines Maschinenbefehls gesetzt werden.

Go [⟨adr⟩]
Starten des Anwenderprogramms von ⟨adr⟩ bzw. dem aktuellen Befehlszählerstand.
Das ROM-Programm enthält einige Hardwareinitialisierungen, die beim Start mit RESET ins Anwenderprogramm entfallen. Ein typisches Beispiel ist die Zeichenausgabe über die SIO. Das Bit 4 im IRQ-Register wird nach RESET in den Nicht-bereit-(0)-Zustand gesetzt. Man muß daher entweder mit OR IRQ, #%10 dieses Bit setzen oder die erste Ausgabe ohne Abfrage dieses Bits ausführen.

Execute [⟨adr⟩]
Ausführen eines Unterprogramms, dessen Eintrittspunkt der aktuelle Befehlszählerstand bzw. ⟨adr⟩ ist. Der Unterbrechungspunkt wird nicht gesetzt, und die Anwenderregisterspeicher werden weder gesetzt noch gerettet.

Next [⟨nr⟩]
Ausführen der nächsten ⟨nr⟩ Programmschritte. Ohne Argument wird der nächste Maschinenbefehl ausgeführt. Bei Extended Memory Timing muß das Register %F8 mit dem R-Befehl des Monitors auf %92 eingestellt werden. Damit wird auf Normaltiming umgestellt.
Dieser Befehl arbeitet mit dem Timer-O-Interrupt. Bei IO-Interrupts im Anwenderprogramm kann das zu Fehlfunktionen führen.

Transparent
Übergang in den Transparent Mode (s. oben). Der Transparent Mode wird mit dem Drücken der STOP-Taste verlassen.

Load ⟨file⟩
Laden eines Programms mit dem Namen ⟨file⟩ vom Wirtsrechner im Tektronix-Format mit Quittung (s. Punkt 4.).

Upload ⟨file⟩ ⟨adr⟩ ⟨len⟩ [⟨entry⟩]
Abladen eines ab ⟨adr⟩ gespeicherten Programms (bzw. Daten) mit der Länge ⟨len⟩ Bytes auf dem Wirtsrechner, ebenfalls im Tektronix-Format. Mit ⟨entry⟩ kann ein Eintrittspunkt angegeben werden, der ansonsten wie ⟨adr⟩ gesetzt wird. Beim Laden des Programms wird der Anwenderbefehlszähler mit diesem Wert geladen.

eXterne Speicheransprache
Setzt man vor eines der Kommandos Compare, Display oder Move ein 'X', so wirken diese Programme im Datenspeicher. Move transportiert dann Bytes vom Daten- in den Programmspeicher. Bei 'XD' (eXternal Display) können keine Daten in den Datenspeicher eingegeben werden.

Im ROM-Programm werden die Register 4 ... %2F benutzt. Die Anwenderregister dieses Bereichs sind deshalb als virtuelle CPU im RAM gespeichert und werden beim Programmstart mit 'Go' an die richtige Stelle gebracht. Dasselbe trifft auf die Steuerregister IMR, FLAGS, RP, SPH und SPL zu.
Der MPBASIC-Editorteil benutzt die Register %78 bis %7F.
MPBASIC-Programmzeilennummern, die in Hexschreibweise ein %0D im Lowbyte haben, werden nicht richtig einsortiert. Man vermeidet deshalb ungerade Zeilennummern oder prüft die Zahl mit dem PRINTHEX-Befehl.
Die Adressen %800 bis %81A werden vom ROM initialisiert. Nach RESET müssen eventuell geänderte Zellen in diesem Bereich neu eingestellt werden. Der Editor setzt die Eingabe (z. B.)

PROC SETR %50, 0, PTC %40
in
PROC SETR %50, 0; PROC PTC %40

um, um einen internen Fehler des UB8830 zu umgehen.

3.2. Assembler und Disassembler

Mit Assemble ⟨adr⟩ kann ein bei ⟨adr⟩ beginnendes Programmstück kontrolliert bzw. eingegeben werden. Zuerst wird der Disassembler aufgerufen und der bei ⟨adr⟩ stehende Befehl in disassemblierter Form angezeigt. Dahinter kann ein neuer Befehl in mnemonischer Form eingegeben werden. Mit CR (ASCII Carriage Return) wird der nächste Befehl rückübersetzt. Mit 'Q' kommt man in den Systemmonitor zurück. Der Assembler versteht neben den üblichen U8810-Mnemoniks die folgenden Pseudo-Befehle:

BVAL ⟨byte⟩
Byte Value, Definieren eines 8-Bit-Wertes

WVAL ⟨word⟩
Word Value, Definieren eines 16-Bit-Wertes

DEFS ⟨len⟩
Define Storage, Freihalten von Speicher der Länge ⟨len⟩

DEFM '⟨text⟩'
Define Message, Einsetzen von ASCII-Text in das Programm.

8-Bit-Operanden werden dezimal oder hexadezimal eingegeben, Hexzahlen werden mit einem vorangestellten '%' gekennzeichnet. 16-Bit-Operanden sind immer hexadezimal ('%' kann dabei entfallen). Die Adressen von Arbeitsregistern oder Arbeitsdoppelregistern sind immer dezimal einzugeben. 'a' (Commercial at) kennzeichnet direkte Operanden. Die Indexregister bei indizierter Adressierung werden in runde Klammern eingeschlossen.

3.3. Editor/Debugger für Tiny MPBASIC

Mit # als Monitorkommando wird Tiny-MPBASIC (im folgenden TMPB abgekürzt) aufgerufen. Es meldet sich der Editor/Debugger mit seinem Prompt '#'. Steht noch kein Programm im Speicher, muß zuerst das Kommando NEW gegeben werden. Beim Einschalten wird ein eventuell im gepufferten CMOS-RAM stehendes Programm nicht gelöscht. Daher ist der RAM beim erstmaligen Einschalten undefiniert belegt, und es fehlt die Endekennung. Diese wird mit 'NEW' initialisiert.

Die erste Adresse des BASIC-Textes steht im Doppelregister 6,7. Sie wird vom ROM-Programm mit %900 initialisiert, kann aber bei Bedarf mit PROC SETRR [6,% ⟨neue Adr⟩ I] im MPBASIC verändert werden. Letzteres trifft auch auf die Adresse der Prozedurtabelle zu, die im Register 8,9 steht und mit %0000 initialisiert wird (bedeutet: keine Prozedurtabelle vorhanden). Weitere Kommandos in dieser Programmkomponente sind:

LIST [⟨zeile⟩]
Auflisten der spezifizierten Programmzeile bzw. des gesamten Programms. Wenn nur eine Zeile aufgelistet wurde, kann durch anschließendes CR die nächste Zeile gelistet werden. Letztere Betriebsart ist insbesondere für den Betrieb mit einzeiligen Datensichtgeräten gedacht, wie sie bei den vorgesehenen Einsatzfällen typisch sind.

RUN
Starten des BASIC-Programms. RUN ruft den MPBASIC-Interpreter im UB8830 auf. Dieser benutzt für PRINT, PRINTHEX und INPUT die Programme PUTCHR und GETCHR /1/ mit den Eintrittspunkten %815 bzw. %818 (im externen Speicher). Das ROM-Programm initialisiert diese Adressen mit Sprungbefehlen in die ROM-eigenen Routinen. Bei der Verwendung anderer Ein- und Ausgabemittel können diese Sprungadressen mit dem Assembler auf andere Werte eingestellt werden.

Im Falle eines Fehlers bei der Abarbeitung des BASIC-Programms wird mit einer Fehlermeldung angehalten. Die folgenden Fehler werden dabei erkannt:

#1 Überlauf des GOSUB-Stack
#2 Auftreten von RETURN ohne GOSUB
#3 Auftreten von GOSUB ohne RETURN (Meldung erst am Programmende)
#4 Division durch Null
#8 Zahlenbereichsüberschreitung.

Bei Zahlenbereichsüberschreitung wird die Programmabarbeitung im Falle von Addition und Subtraktion nicht gestoppt. Die Fehlermeldung erfolgt dann erst am Programmende. Dadurch wird verhindert, daß eine gewollte Überschreitung, die z. B. bei hexadezimaler Adressenarithmetik auftreten kann, ein Anhalten des Programms bewirkt.

EXEC
Starten eines BASIC-Programms, jedoch kein STOP bei Fehlern

CONT[⟨zeile⟩]
Fortsetzen des Programms ab ⟨zeile⟩ bzw. ohne Argument nach STOP

STEP[⟨zeile⟩]
Abarbeiten der angegebenen bzw. der nächsten Programmzeile. Wenn einmal STEP ausgeführt wurde, kann mit CR die Folgezeile abgearbeitet werden.

SIZE
Ausgabe folgender Daten von Programm und Prozedurtabelle:
Anfangsadresse, Endadresse und Länge

GET ⟨programmname⟩
Laden eines BASIC-Programms vom Wirtsrechner. Das Programm wird auf die Adresse zurückgeladen, von der es auf den Wirtsrechner abgeladen wurde.

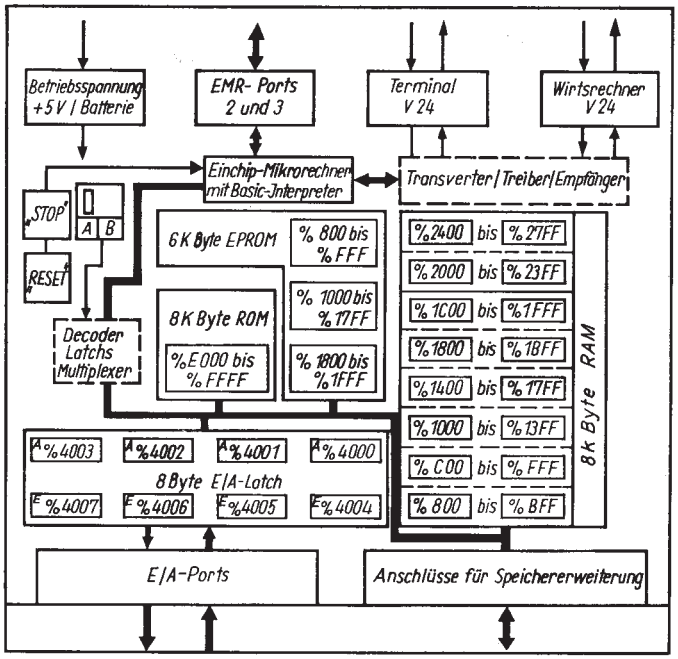


Bild 3 Blockschaltbild des UB8830-Entwicklungsmoduls

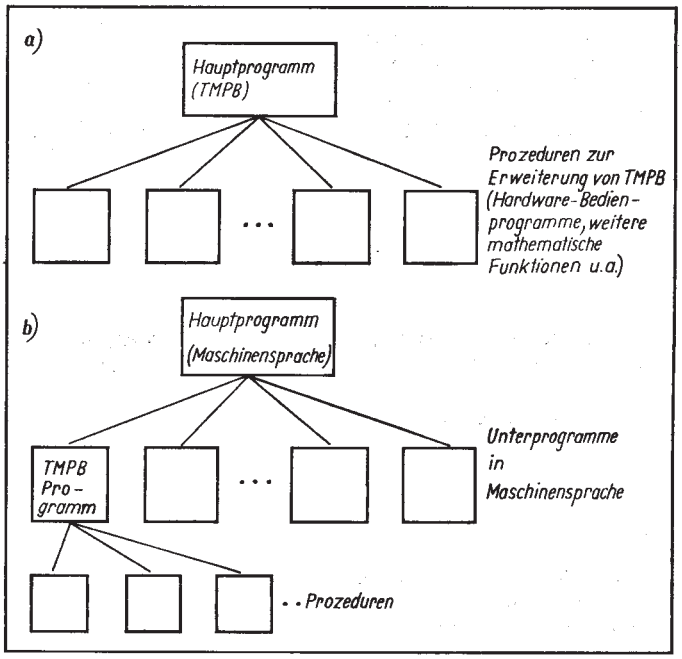


Bild 6 Mögliche Strukturen von Programmen

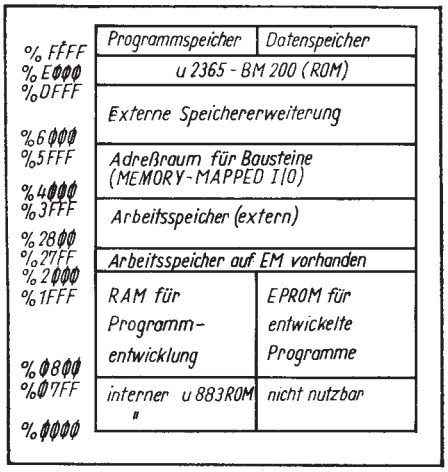


Bild 4 Speicheraufteilung im U8830-Entwicklungsmodul

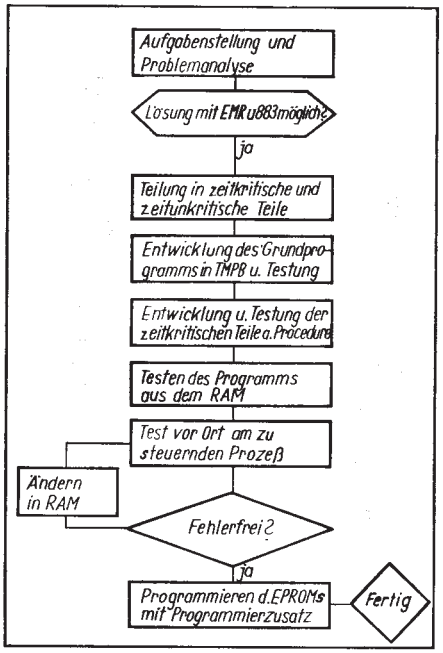


Bild 5 Programmentwicklung mit dem U8830-Entwicklungsmodul

SAVE ⟨programmname⟩
Abladen eines BASIC-Programms auf den Wirtsrechner

BYE
Rückkehr in den Systemmonitor.

4. Datenaustausch mit einem Wirtsrechner

Bevor der Datenaustausch beginnt, wird das Kommando "LOAD ⟨filename⟩" (Load oder GET) oder das Kommando "SEND ⟨filename⟩" (bei Upload und SAVE) an den Wirtsrechner gegeben, damit dieser das entsprechende, als Kommando aufrufbare Programm LOAD oder SEND in seinen Arbeitsspeicher laden und starten kann. Wenn SEND gestartet wurde, darf der Wirtsrechner auf die zu ihm gesendeten Zeichen kein Echo mehr ausgeben. Benutzt wird das Tektronix-Format zur seriellen Übertragung. Die Daten werden in Blöcke geteilt, von denen jeder eine Startadresse, die Bytezahl, zwei Testsummen sowie die eigentlichen Daten enthält.

Diese Blöcke sind folgendermaßen aufgebaut:
/⟨adresse(4)⟩ ⟨bytezahl (2)⟩ ⟨testsumme1 (2)⟩ ⟨Datenbyte(2)⟩ ...
... ⟨Datenbyte(2)⟩ ⟨testsumme2(2)⟩ ⟨CR⟩
Dabei markiert '/' den Anfang eines Datenblocks oder einer Fehlermeldung.
⟨adresse(4)⟩ ist die Adresse für das erste Datenbyte (in 4 ASCII Zeichen dargestellt).
⟨bytezahl(2)⟩ gibt die Anzahl der Datenbytes an.
⟨testsumme1(2)⟩ wird aus Anfangsadresse und Bytezahl gebildet.
⟨datenbyte(2)⟩ ist ein Datenbyte im Tektronix-Format.
⟨testsumme(2)⟩ wird für die Datenbytes berechnet.
⟨CR⟩ ist das ASCII-Zeichen CR (%0D).

Die Kodierung des Tektronix-Formats ist so festgelegt, daß ein Halbbyte (eine Hexziffer) als ASCII-Code übertragen wird (z. B. %8B wird als Folge %38 %42 codiert). Der letzte Block beinhaltet die Eintrittsadresse:

/⟨eintritt(4)⟩ ⟨bytezahl(2)⟩ ⟨testsumme(2)⟩ ⟨CR⟩
Anstelle der Blockanfangsadresse steht der Eintrittspunkt; die Bytezahl ist immer Null (%30 %30), und die Daten einschließlich testsumme2 fehlen.
Nach dem Senden eines Datenblocks wird die Quittung von der Gegenstelle erwartet. Bei fehlerfreier Übertragung wird mit '0' (%30) quittiert, '7' bedeutet Testsummenfehler, eine '9' bedeutet Systemfehler im Wirtsrechner. Bei Testsummenfehlern wird die Übertragung bis zu 10mal wiederholt, um Störungen zu unterdrücken. Auch eine Fehlermeldung der Form

//⟨fehlermeldung als ASCII-Text⟩ ⟨CR⟩

führt zum sofortigen Abbruch der Datenübertragung, wobei das zweite '/'-Zeichen den Block als Fehlermeldung kennzeichnet.

5. UB8830-Entwicklungsmodul

Bild 3 zeigt das Blockschaltbild des UB8830-Entwicklungsmoduls (im folgenden EM abgekürzt) und Bild 4 die Speicheraufteilung. Es kann ab Adresse %800 mit bis zu 6 KByte Daten- und 6 KByte Programmspeicher bestückt werden (RAMs 1K×4 und EPROM U2716). Darüber hinaus stehen maximal 2 KByte Arbeitsspeicher ab Adresse %2000 zur Verfügung. Mit S1 kann der 6-KByte-RAM-Bereich wahlweise als Daten- oder Programmspeicher geschaltet werden. Der EPROM ist der jeweils alternative Speicherbereich. Um mit dem ROM U2365D45 BM200 arbeiten zu können, muß der RAM als Programmspeicher geschaltet sein.
Mit dem Kommando "XM" des Systemmonitors ist es möglich, den EPROM-Inhalt (gewöhnlich Programme) in den RAM-Bereich auf die gleichen Adressen zu kopieren. Dies vereinfacht das Testen und Bearbeiten mit

dem Monitor. Damit ist es ebenfalls möglich, das Programm aus dem RAM direkt über einen Programmierzusatz in EPROMs zu programmieren. Mit dem Schalter S2 kann der RAM ein Schreibverbot erhalten, wodurch der EMR UB8830 bei der Startroutine einen ROM auf Adresse %800 ff. vermutet, was wiederum zum Start des Programms ab %812 führt. Damit kann das Programm so starten, als ob es bereits in einem EPROM stünde.
Von den maximal 2 KByte Arbeits-RAM werden vom ROM-Programm etwa %100 Byte benötigt. Der Rest dieses Bereichs steht dem Anwender zur Verfügung. Die Umschaltung zwischen Programm- und Datenspeicher betrifft nur die 6 KByte Entwicklungsspeicher, also nicht den ROM-Bereich ab %E000 und den Arbeits-RAM. Alle RAM-Bausteine sind in eine gepufferte Betriebsspannung angeschlossen, die es z. B. erlaubt, ein an der zu steuernden Anlage geändertes und getestetes Programm an einem anderen Ort (etwa im Labor) im EPROM zu programmieren oder auf einem Massenspeicher abzulegen. Acht Latches stehen mit je 8 Bit als Memory-mapped-I/O zur Verfügung, davon 4 für Eingabe- und 3 für Ausgabezwecke, wobei jeweils die Output-Enable- bzw. Strobeeingänge der Latches mit herausgeführt sind.
Der interne getriebene Bus des EM steht an einem 58poligen Steckverbinder für Erweiterungen zur Verfügung. Der Adreßraum des EM ist voll dekodiert, für jeweils 8 KByte liegen sogenannte Blockenablesignale am Erweiterungsstecker an.
Ein Transverter erzeugt die für die abgerüsteten V.24-Schnittstellen benötigten Spannungen von +12 V und −12 V. Das gesamte EM läßt sich daher mit einer Betriebsspannung von 5 V (ca. 1,6 A) betreiben. Mit dem zusätzlich verfügbaren EPROM-Programmierzusatz stellt das EM bereits ein für kleinere Anwendungsfälle komplettes Entwicklungssystem dar. Da die Anforderungen für Tastatur und Anzeige sehr unterschiedlich sind und oft bereits ein Rechner mit diesen Einheiten und einer seriellen Schnittstelle zur Verfügung steht, ist hier keine Standardlösung vorgesehen. Bekannt sind Lösungen mit Bildschirmsteuerung (komplettes Terminal), aber auch mit LED-Kombinationen wie VQC10 oder VQB201. Im allgemeinen wird dazu ein zweiter EMR zur Steuerung verwendet.

6. Softwareentwicklung

Im Bild 5 ist der Ablauf der Programmentwicklung mit dem UB8830-EM dargestellt. Vor Ort können Programmkorrekturen ohne Wirtsrechner durchgeführt werden. Einzige Voraussetzung ist eine seriell betriebene Tastatur- und Anzeigeeinheit. Ein typisches Programm für den UB8830 ist in Tiny-MPBASIC geschrieben. Nur zeitkritische Teile und in MPBASIC zu umständliche Teile schreibt man in Maschinensprache und bindet sie als Prozeduren in das MPBASIC-Programm mit ein (Bild 6a). Da es möglich ist, ein MPBASIC-Programm wie ein Maschinenprogramm aufzurufen (siehe /1/), kann ein Programmaufbau nach Bild 6b gewählt werden. Hinweise und Beispiele zu Tiny-MPBASIC sind in /3/ enthalten.

(Literatur und Kontakthinweis auf Seite 251)

Interpretativ arbeitende speicherprogrammierbare Steuerung SKS 02

Dr. Ronald Schoop, Holger Lisson,
Prof. Dr. Wolfgang Weller
Humboldt-Universität zu Berlin,
Sektion Elektronik

1. Einleitung

Eine zunehmende Ablösung von verbindungsprogrammierten Steuerungen, wie z. B. Relais oder ursalog 4000, auch für Steuerungsprobleme geringerer Komplexität wird wesentlich durch die Entwicklung neuer speicherprogrammierbarer Steuerungen (SPS), wie etwa EFE 700 /1/ oder S 2000 S /2/, forciert. Zur Lösung von Steuerungsaufgaben mit einer Anzahl von weniger als 40 Ein- und Ausgängen fehlen jedoch aufwandsminimale und damit wirtschaftliche SPS, insbesondere wenn auch der notwendige Aufwand für Programmier- und Inbetriebnahmegeräte mit berücksichtigt wird.
Um auch für derartige Steuerungsprobleme die bekannten Vorteile von SPS /3/ nutzbar zu machen, wurde eine speicherprogrammierbare Kleinsteuerung SKS 02 auf der Basis von Einchipmikrorechnern (EMR) mit folgender Zielstellung entwickelt /4/ bis /14/:

- minimaler Aufwand für Steuer- und Programmiergerät bei Aufgaben mit weniger als 30 Ein- und Ausgängen
- einfache Handhabung bei Programmierung und Inbetriebnahme
- hohe Störsicherheit
- Kompatibilität zum System ursalog 4000.

Die letztgenannte Forderung ermöglicht neben dem Aufbau eigenständiger SKS auch eine Aufwertung bisheriger Steuerungen auf der Basis von ursalog-4000-Baugruppen.

2. Gesamtkonzept

Zur Programmierung und Inbetriebnahme werden der Programmiergerätekomplex (PGK) und der Steuergerätekomplex (SGK) miteinander seriell (UART; 19,2 kbit/s) entsprechend Bild 1 verbunden. Im Steuerungsbetrieb kann diese Schnittstelle zur Kopplung mit einer zweiten Steuerung oder einem übergeordneten Rechner genutzt werden, wobei Merkerbelegungen oder Programme übertragen werden können.

3. Struktur des Steuergerätekomplexes

Die programmierte SKS kann mit dem Modell des endlichen Automaten beschrieben werden, der den Steueralgorithmus A verwirklicht. Notwendige Zähl- und Zeitfunktionen werden dabei gesonderten Funktionsgruppen Z bzw. T (Bild 2) zugeordnet. Demzufolge brauchen diese im zu programmierenden Steueralgorithmus A' nur insoweit berücksichtigt werden, als sie über Eingänge X' (Zählkonstante erreicht bzw. Zeit abgelaufen) und Ausgänge Y' (Zählerstand erhöhen bzw. Zeitablauf starten) verkoppelt sind. Zur Erhöhung der Flexibilität werden zusätzlich die Zähl- bzw. Zeitkonstanten vom Steueralgorithmus A', d. h. vom Anwenderprogramm festgelegt, so daß nichtüberlappende Mehrfachnutzungen prinzipiell möglich sind.
Sämtliche Funktionsgruppen T, Z und A' sind im Steuergerät der SKS 02 implementiert.
Die Grundkonfiguration wird dabei aus einer Zentraleinheit (ZE) und einer Expanderkarte (EX) gebildet. Zur Aufrüstung kann diese Anordnung um eine weitere Expanderkarte gemäß Bild 3 erweitert werden.
Den Kern der Zentraleinheit (Bild 4) bildet der Einchipmikrorechner UB 8820. Das Betriebssystem ist in einem EPROM U 2716 und das Anwenderprogramm wahlweise in einem 2-KByte-EPROM oder RAM abgelegt, was einer Kapazität von ca. 600 ... 900 Anwenderbefehlen entspricht. Über das serielle Interface kann eine TTL-kompatible Schnittstelle angeschlossen werden. Eine Funktionsgruppe zur Anzeige erlaubt das Darstellen der Belegung von 8 internen Merkervariablen und, in Verbindung mit der watch-dog-Schaltung, die Anzeige eines Fehlerkodes im Störfall. Letzterer wird durch eine Autodiagnose (EMR, RAM, EPROM) erkannt oder durch einen undefinierten Rechner-„absturz" verursacht und führt zum Stopp des Programmlaufes und zum Rücksetzen der Ausgänge. Weiterhin wird ein selbsttätiges Anlaufen nach Spannungsausfall und -wiederkehr verhindert.
In der Zentraleinheit implementiert sind 8 Zähler/Zeitgeber mit den Bereichen 1 . . 256 bzw. 0,1 ... 25,6 s, die mit einem Anwenderbefehl parametriert werden. Es können 128 Zustandsvariablen (Merker) belegt werden. Die Arbeitsweise der Zentraleinheit ist in

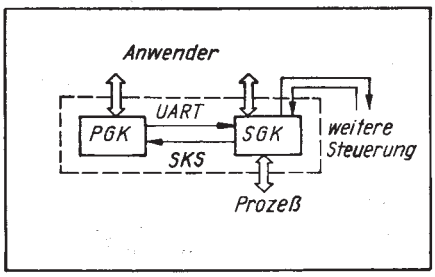


Bild 1 Gesamtstruktur der SKS 02

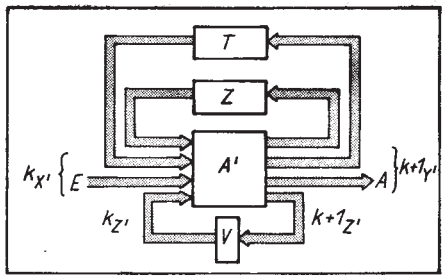


Bild 2 Partiell dekomponiertes Automatenmodell