Mikrorechner auf Z80-Basis

Der hier aufgebaute Rechner ist ein Demonstrationsobjekt zur Analyse einzelner Programmbefehle. Die Schaltungsidee entstammt der Zeitschrift „Funkamateur 3/85“. Dieser Rechner verfügt nur über die nötigsten Komponenten. Ein Beschreiben von EPROMs mit einem Betriebssystem ist nicht notwendig, da der Prozessor direkt das im Speicher stehende Programm abarbeitet.

Alle wichtigen Signale zur und von der CPU sind vom Bediener erfassbar. RESET, NMI und BUSRQ werden über Tasten am Bedienfeld gesteuert. Speicherzugriffe und Betriebszustände der CPU sind über die Anzeigetafel an den entsprechenden LEDs ablesbar. Für Tastatureingaben steht ein vollwertige Hexadezimaltastaur zur Verfügung. Auch der Speicherzugriff auf bestimmte Adressen ist über die vorhandenen Taster möglich.

Die CPU

Im Original kommt eine U880D als CPU zum Einsatz. Dieses Modell nutzt die kompatible Z80-CPU von Zilog. 16 Adress- und 8 Datenbits bieten eine Menge Freiraum für Versuche und halten außerdem die Möglichkeit für Erweiterungen offen. Über den Eingang WAIT kann die CPU im aktuellen Programmablauf angehalten werden. Durch M1, IORQ, WR, MREQ und RD werden Befehle angefordert bzw. Daten zu den entsprechenden Zielen gelenkt.

Der Taktgenerator

Die Frequenz des externen Oszillators ist mit 1 MHz spezifiziert, doch zur besseren Veranschaulichung wurde sie deutlich herabgesetzt. Dadurch lassen sich bestimmte Vorgänge besser erkennen.

Der Adresspuffer

Die Adressbits der CPU sind nicht direkt auf den Adressbus gelegt, sondern werden über Zählerstufen gepuffert. Das ermöglicht es, die CPU vom RAM zu trennen und manuell im Speicher gewisse Adressen zu begutachten oder sogar zu manipulieren. Im direkten Speicherzugriff (DMA) wird der Ladeeingang der Zähler freigegeben und die gewünschte Adresse kann dann über die Taster ZV und ZR eingestellt werden.

Der Datenmultiplexer

Der Datenbus muss in mehreen Richtungen funktionieren. Daten müssen aus der CPU in den RAM gelangen können. Von der Tastatur müssen Eingaben entweder direkt zur CPU oder in den RAM übertragen werden. Alle Datenbewegungen sind zudem immer auf der Anzeige verfolgbar. Die Weichen für alle diese Übertragungen stellt der Datenmultiplexer aus 6* D103D und den Dekodierern D110 und D100. Der Anschluss weiterer Peripheriegeräte ist möglich. Dazu müssen die Ein-/Ausgaberichtungen lediglich über freie/zusätzliche Pins an den Zustandsdekodierern angesprochen werden.

Die RAMs

Der Arbeitsspeicher (RAM) ist mit dem Adressbus verbunden. Die verwendeten Speicherbausteine besitzen eine Kapazität von 1024*1 Bit. Sie besitzen also 10 Adresseingänge und jeweils einen Datenein- bzw. -ausgang. Insgesamt verfügt der Rechner über 1 kByte (8*1024 Bit) Arbeitsspeicher. Desweiteren handelt es sich um SRAM, d.h. die Daten müssen nicht periodisch über ein Refresh-Signal am Leben gehalten werden. Der Chipselect-Eingang aller RAMs liegt auf A10 des Adressbuses. Daher wird der RAM durch A10=LOW angewählt. Eventuelle Erweiterungen, die ebenfalls an den Adressbus gekoppelt werden können, werden dann über die darüberliegenden Adressbits dekodiert und angesprochen. Die Eingabedaten bekommt der RAM vom Datenbus. Vom Speicher gelesene Daten werden über den Datenmultiplexer auf die richtigen Wege geleitet.Im Arbeitsspeicher steht das Programm, das die CPU im Betrieb abarbeitet. Um eben dieses Programm auf den Speicher schreiben zu können ist der direkte Speichzugriff mit Schreibbetrieb notwendig. Über die Hexadezimaltastatur werden die Daten somit in die Speicherzellen geschrieben.

Technische Kniffe

für den Elektroniker, oder den weit verbreiteten gemeinen Bastler sind aus diesem Projekt einige interessante Schaltungsideen zu entnehmen.

Die Impulserzeugung bei der Tastatureingabe ist ein schönes Beispiel wie aus mehreren Bedienelementen ein Signal gewonnen werden kann um weitere Ereignisse auszulösen. Auch wenn das nur bis zu einer gewissen Anzahl Sinn macht um die notwendigen Dioden in einem überschaubaren Rahmen zu halten.

Der Datenmultiplexer ist ein schönes Beispiel für Lenkung von Daten und die Auswahl spezieller Gatter für den jeweiligen Zweck. Wichtig zu erkennen ist, dass hier NAND-Gatter mit Opencollector-Ausgang bewusst eingesetzt wurden. Da Ausgänge von mehreren ICs aufeinander geschaltet werden, muss sichergestellt werden, dass keine unzulässigen Signalüberschneidungen zwischen LOW und HIGH erzeugt werden. Die abschließenden Gatter des Multiplexers werden nur bei Ein-/Ausgabeanforderungen aktiviert und bleiben anderenfalls hochohmig. Dadurch kann die CPU ihre Datensignale in allen anderen Situationen gefahrenlos durchsetzen.

Beim Nachbauen der Schaltung würde man auch auf einige Schwachstellen stoßen. So sind z.B. an einigen Stellen beim Umschalten des Speicherzugriffs und des Schreibzugriffs manche Eingänge unbeschaltet. Es gab und gibt Gatter die das verkraftet haben. Heutzutage müsste man allerdings eine äußerst exakte Auswahl an Schaltkreisen treffen um solche Kunststücke in der Gesamtschaltung zu meistern. Daher empfiehlt es sich an solchen kritischen Punkten noch Widerstände für definierte Pegel einzubinden. Der Dekoder des Multiplexers ist ein sehr gutes Beispiel dafür.

Die WAIT-Logik

Die CPU-Signale für Befehlsholezyklus (M1), Eingaben (IN), Ausgaben (OUT), Speicherlesen (MR) und Speicherschreiben (MW) können über die Schalter auf dem Bedienfeld an einen D130 gelegt werden. Wenn ein durchgereichtes Signal erkannt wird, setzt das Flip-Flop D174 das WAIT-Signal zurück und die CPU wartet mit dem weiteren bearbeiten von Befehlen. Durch eine Betätigung von ZV wird dies rückgängig gemacht und der Prozesor darf weitere Befehle abarbeiten, bis das nächste WAIT-Signal erkannt wird.

Ein kleines Testprogramm:

0000h -	**3E, 55**	**LD A, 55h**	**Schreibe 55h in das Register A**
0002h -	**32, 10, 00**	**LD 0010h, A**	**Schreibe das Register A auf die Speicherzelle 0010h**
0005h -	**76**	**HALT**	**Halt!**

Dieses Programm macht nichts weiter als eine Speicherzelle mit einem Bitmuster zu füllen.
Doch hier lassen sich schon ein paar schöne Dinge erkennen.

- Man kann im Programmablauf nie eine Speicherzelle direkt beschreiben.
  Der Wert muss immer erst in ein Register geladen werden, bevor er in die entsprechende Speicherzelle übertragen wird.
- Die Register können bei diesem Rechner nicht direkt auf die Anzeige ausgegeben werden.
  Der Trick ist, dass man das Register in eine Speicherzelle schreibt.
  Diese Daten können dann auf dem Datenbus abgelesen werden.
  Oder man schaut sich im nachhinein die Speicherposition im direkten Speicherzugriff an.
- Die Speicherzugriffe beim schrittweisen abarbeiten des Programms sind offensichtlich.
  Bei jedem neuen Befehl (M1) wird die Speicherzelle über den Datenbus ausgelesen (RD + MREQ).
  Beim Schreiben der Daten auf die Speicherzelle ist die programmierte Adresse aktiv,
  die programmierten Daten liegen auf dem Datenbus an und
  ein Schreibimpuls wird an die RAMs durchgereicht (WR + MREQ).

Die Hexadezimaltastatur

Über eine Diodenmatrix werden bei Betätigung einer Taste die vier Leitungen für das Halbbyte mit den entsprechenden Pegeln belegt. Über einen weiteren Diodenabgriff wird jeder Tastendruck in einen Impuls umgeformt, der als Ladesignal für die Register D195 dient. Dabei werden die alten Daten des Halbbytes in das zweite Register übernommen und sofort danach die neuen Daten von den Tastaturleitungen in das erste Register geschrieben. So setzt sich ein Befehlsbyte immer aus zwei Halbbytes je eines Tastendrucks zusammen. Die Ausgänge der Register werden dem Datenmultiplexer zugeführt und darüber in den Datenbus eingekoppelt.