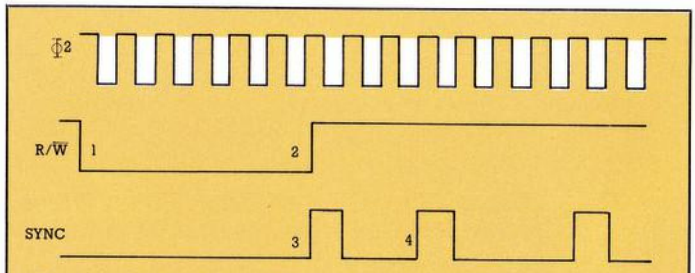
Kennen Sie Ihren C 64?

(Teil III)

Können Sie schon etwas durch Ihren C 64 hindurchsehen? In dieser Folge werden wir durch die Beschreibung eines Bus-Scanners unserem Ziel, den C 64 vollkommen zu verstehen, etwas näherkommen.

Mittlerweile gibt es für den C 64 unzählige Programme und Hardware-Erweiterungen. Doch im Bereich der Bus-Analyse gibt es für den C 64 kaum ein Produkt. Auch im professionellen Bereich wird der C 64 bekanntlich genutzt, aber auch hier fehlen Hilfsmittel zur Fehlerbeseitigung (Debugging). Kennt man die Arbeitsweise von »State-Analysern« und schaut man sich die Architektur des C 64 näher an, so weiß man auch, warum es diese Hilfsmittel für den C 64 nicht gibt. Die State-Analyser und/oder Datenlogger müßten speziell für den C 64 entwickelt werden, da ein kompliziertes und meßtechnisch schwieriges Timing erfaßt werden muß. So wird zum Beispiel die CPU zyklisch angehalten, weil der Master (GDP) in das System will. Auch die »normale« CPU 6502 hat einen Anschluß, der das Abarbeiten eines Befehles anzeigt, nämlich das Signal SYNC (Synchronisation). Immer, wenn die CPU einen Opcode (Befehl) holt, ist dieses Signal auf High (Bild 1). Dadurch wird das Aufzeichnen und spätere Analysieren sehr einfach. Doch bei der 6510 CPU (im C 64) existiert dieses Signal nicht. Dazu kommt, daß die C 64-Entwickler im Timing »gemogelt« haben. In normalen 65xx Systemen hat die CPU einen vollen Takt zur Verfügung. In der ersten Hälfte des Taktes stabilisieren sich die Adreßleitungen und die Schreib-/Leseleitung, während in der zweiten Takthälfte die Datenübertragung stattfindet. Im Prinzip ist das beim C 64 auch so. Nur das ganze Timing ist verschoben und die Zeit für die CPU 6510 wesentlich kürzer geworden. So kann man zum Beispiel keine VIA 6522 (wesentlich billiger als die CIA 6526) an das System anschließen, ohne der VIA einen anderen Mastertakt vorzugaukeln (Bild 2). Aber zurück zur Analyse. Was heißt Opcode bei einer CPU. Das Verfahren ist bei allen CPUs der Welt gleich: Wenn eine CPU keine Sprünge, Verzweigungen oder Schreibbefehle ausführen muß, so läuft Sie immer geradeaus. Sie hat dazu einen Adreßzähler, und der zählt die Adresse immer um eins hoch. Nehmen wir einmal an, daß die CPU einen Befehl beendet hat. Der Adreßzähler (Programcounter, PC) wird um eins erhöht, und die CPU liest das nächste Byte. Dieses Byte ist der Operationscode (Opcode), er bestimmt die Art der nächsten Operation (Bild 3). Zum Beispiel der Opcode #$AD (dezimal = 173, auf dem Datenbus, also binär = 10101101) wird von der 65xx-Serie als Ladebefehl, absolut, in den Akku der CPU interpretiert. Die CPU »weiß« nach dem Byte, daß sie noch zweimal den Adreßzähler erhöhen und die 2 Byte für die Absolutadresse holen muß — so einfach ist das. Hat die CPU die Adresse, legt sie diese Adresse auf den Bus, um das Byte aus der Adresse zu laden. Jedoch ist gerade für Anfänger in der Maschinensprache schwer erkennbar, was in ein paar millionstel Sekunden alles passiert ist. Hat man eine Flag nicht beachtet oder eine Verzweigung falsch gesetzt, läuft die CPU ganz woandershin — sie stürzt sogar wahrscheinlich ab. Hier müßte es ein Hilfsmittel geben, um zu erkennen, was die CPU macht, genauer gesagt, wo die CPU im Adreßraum läuft, und was auf dem Datenbus passiert. Aber das kann auch ein Datenanalyser oder Datenlogger nicht. Ein Vorgang wird zwar nach vereinbartem Triggerwort aufgezeichnet und ist hinterher auswertbar, was und wie oft in Echtzeit bei welcher Adresse geschrieben und gelesen wird, ob im IRQ oder nicht und welche Register welchen Inhalt haben, bleibt verborgen. Das alles ist in der Nachbereitung meistens sehr zeitraubend. Startet man zum Beispiel ein Maschinenprogramm und der Computer »kommt nicht wieder«, möchte man schon gerne wissen, ob die CPU auf »Wolke sieben« oder nur in einer Schleife läuft, aus der sie aufgrund eines Programmierfehlers nicht wieder herausfindet (Bild 4). Dazu kommen die vielfältigen Möglichkeiten der Portprogrammierungen. Ist bei einer Centronics-Schnittstelle am User-Port das Byte angekommen oder hat sich das Programm verlaufen? Wur-



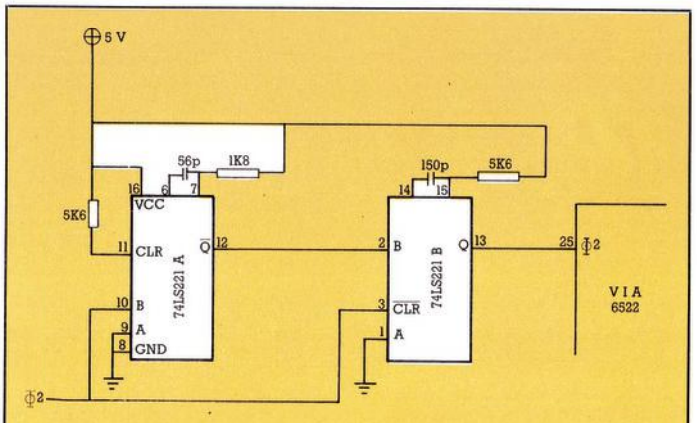
1. Bei einem Schreibzyklus liegt R/W auf Low und der SYNC-Impuls tritt nicht auf.

2. Wenn R/W auf High geht, wird ein Leservorgang der CPU angezeigt.

3. Anfang wird für einen Zyklus der SYNC-Impuls erzeugt, genauso lang, wie die CPU den opcode holt.

4. Der nächste SYNC-Impuls wird erzeugt, da die CPU den Befehl abgearbeitet hat und ein neuer Befehl beginnt. Der SYNC-Impuls ist also aperiodisch, sein Auftreten ist proportional zur Ausführungszeit eines Befehls.

Bild 1. Das SYNC Signal bei 65xx CPUs



Der Mastertakt der CPU 02 wird um 150 ns »nach vorne« verschoben. Dadurch kann die VIA ohne Probleme auf dem Bus arbeiten. Der Mastertakt des GDPs darf auf keinen Fall zur Synchronisation benutzt werden, da er (entgegen anderen Behauptungen) nicht ein Vielfaches vom Φ2 ist (Ein Grund, warum viele Z80 Karten nicht einwandfrei liefen).

Bild 2. Anschluß der VIA 6522 an den C 64

de ein Programm von der Floppy an die richtige Stelle geladen oder stimmen die Pointer nicht? Nach all diesen Kriterien wurde ein Scanner entwickelt, der sowohl in der Software- als auch in der Hardware-Entwicklung gute Dienste leisten kann. Er ist modular ausbaufähig, bis zum Analyser. So ein Scanner darf keinen Speicherplatz belegen und keine Software darf ihn »treiben«, sonst wäre das ein Kompromiß. Doch wie kann man den Lauf einer CPU sichtbar machen und das in einem vertretbaren finanziellen Aufwand? Bekanntlich liegt der Preis für Analyser ab 20 000 Mark aufwärts.

Flimmerstunde beim Entwickeln

Das menschliche Auge ist bekanntlich das am schlechtesten entwickelte Organ des Menschen. Bei Erwachsenen beträgt die Auflösung, das heißt das Erkennen von Hell/Dunkel-Wechsel, maximal 20 Hertz. Eine Glühbirne, gespeist mit Netzspannung, geht in einer Sekunde 50 mal an und aus. Die Netzfrequenz von 50 Hertz nimmt das Auge nicht mehr wahr. Diese Eigenschaft machen wir uns zunutze und lassen immer dann, wenn die CPU etwas tut, Leuchtdioden aufleuchten. Mit wenig Elektronik und geringem Aufwand ist die Grundform des Scanners fertig (Bild 5). Der Effekt ist verblüffend. Klar und deutlich erkennt man, wo die CPU sich befindet. Je öfter eine Adresse »angefaßt« wird, um so heller (häufiger !) leuchten die Leuchtdioden auf. Schaltet man die Adreß- oder Datenfalle ein, hält das Display bei der Triggerbedingung an, dann weiß man: Lesen oder Schreiben, IRQ oder nicht, etc. Man könnte diese Bytes auch in gewohnter Weise hexadezimal durch Digital-Anzeigen darstellen. Obwohl die Zeichen einzeln dargestellt werden, ist die hexadezimale Adresse und das Byte sofort ablesbar, weil das binäre und hexadezimale Zahlensystem in einem direkten Zusammenhang stehen. Für alte Programmierhasen ist es so und so unverständlich, warum das dezimale Zahlensystem in unserer Welt noch Vorrang hat. Da ein Byte 256 verschiedene Zustände haben kann, hat ein Halbbyte demnach 16 (0 bis F) Zustände. Demnach brauchen wir für den Adreßbereich (16 Leitungen = 4 Halbbyte) 64 Leuchtdioden. Dazu kommt der Datenbus (8 Leitungen = 2 Halbbyte) mit 32 Leuchtdioden. Positionieren wir dann 16 Dioden (0 bis F) in einer Reihe, können wir die Adresse beziehungsweise das Byte direkt ablesen. Dazu kommen noch die Leuchtdioden für »Trigger gefunden«, »Schreiben/Lesen« und »IRQ an/aus«. Das wären gut 100 Leuchtdioden zu einem Preis von zirka 12 bis 15 Mark. Ein paar TTL-Bausteine, Flachbandkabel, Expansionsstecker, die Hex — Schalter zum Einstellen der Trigger — und Scannbedingungen und die Schalter zum Modifizieren der Darstellung. Auch die Spannungsversorgung (5 Volt, 1 Ampere) darf nicht fehlen. Das Ganze paßt auf eine Europakarte. Soll das System ausgebaut werden (die Möglichkeiten sind nahezu unbegrenzt), sollte man Steckkarten nehmen. Und wer es ganz nobel möchte, kann die 19-Zoll-Einschubtechnik wählen. Macht man auf Hausbacken, ist eine Ausgabe von 120 Mark realistisch. Natürlich kann man später auch intelligente Displays verwenden, doch die Leuchtdiodentechnik ist als Scannerdisplay durch nichts zu ersetzen. Sollte große Nachfrage bestehen, wird eine Platine lieferbar sein, denn wer lötet schon gerne mehr als 100 Dioden ein (und die alle richtigherum). Es soll nicht verschwiegen werden, daß der Scanner leider auch für Soft- und Hardwareklaue geeignet ist. Doch das läßt sich leider nicht vermeiden und außerdem gibt es hier ja andere Methoden. Eine in diesen Kreisen immer beliebtere und wirksame (inzwischen auch für den C 64 erhältliche) Methode ist das »spanische Prinzip«, wie es vom »Freez Frame« angewendet wird. Man stoppt das ganze System und »zieht« eine komplette Kopie vom System mit allen Parametern. Später kann man dann in Ruhe das Ganze auswerten. Doch wer sucht schon gerne einen Programmierschutz, der 23mal an den unmöglichsten Stellen im System verteilt ist. Der Scanner arbeitet da anders. Unsichtbar für ein Programm meldet er Adressen, Daten und getriggerte Bedingungen, die in Echtzeit ablaufen. Zum Beispiel bereitet es Verdruß, wenn man als schlechter Wizard-Spieler (natürlich im Besitz einer Originalversion) immer gleich alle Schlüssel verliert.

Oder hätten Sie nicht auch gelegentlich schon gerne einmal die höheren Stufen mancher Spiele gesehen? Zu alledem muß an dieser Stelle noch vermerkt werden, daß das Verfolgen von Kopierschutzarten das Denkvermögen schult. Es ist schon interessant, wenn man dem, was Profis unter unendlichem Kopfzerbrechen ausgeheckt haben, auf die Spur kommt. Doch zurück zum Thema. Wie man mit dem Scanner arbeitet und was man noch alles damit machen kann, steht in der nächsten Folge.

(Logo/aw)

Instr.	Adr.	Data	Opcode
-0002	C434	20 3D C6	JSR $C63D
	0141	C4	Mem Write
	0140	36	Mem Write
	C436	C6	Mem Read
-0001	C63D	A5 68	LDA $68
	0068	00	Mem Read
+0000	C63F	D0 28	BNE $C669
+0001	C641	A6 7F	LDX $7F
	007F	00	Mem Read

Bild 3. Datenanalyser im »Disassembler«-Mode (65xx CPU)

Sehr gut ist hier bis in das kleinste Detail die Arbeitsweise der CPU zu erkennen. Bei -0002 steht ein Jump to Subroutine-Befehl (analog in Basic der GOSUB-Befehl). Die CPU packt die Rücksprungadresse ($C436) auf den Stack (Stapelspeicher) und macht bei $C636 mit LDA $68 weiter. Der Datenanalyser zeichnet also jedes Ereignis außerhalb der CPU auf.

Vor dem -0001 steht zwar schon die nächste Adresse, doch der Programmzähler der CPU »hat sich geirrt«. Diese Daten werden in der CPU »verworfen«.

Adresse	Daten	Befehl
$1000	A9 00	LDA #$00
$1002	85 D2	STA $D2
$1004	F0 FA	BEQ $1000
$1006	...	

Bild 4. Die Endlosschleife

Hier hilft nur noch der NMI (Stop/Restore-Taste). Der Wert #$00 wird unmittelbar in den Akkumulator der CPU geladen und beim zweiten Befehl in die Adresse $D2 gebracht. Der Verzweigungsbefehl fragt das Statusregister ab, und stellt fest, daß die Zero-Flag gesetzt ist. Also »brancht« (Verzweige, wenn Z=0) die CPU zur Adresse $1000 und das Spiel beginnt von vorn. Profis, die behaupten, das sei ihnen nie passiert, sind keine.

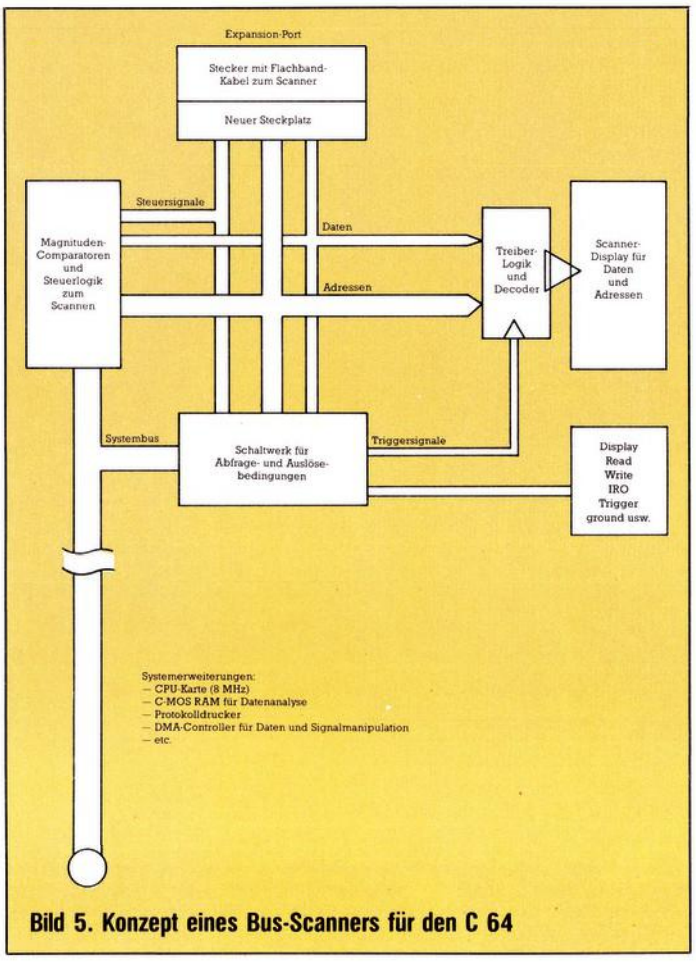


Bild 5. Konzept eines Bus-Scanners für den C 64