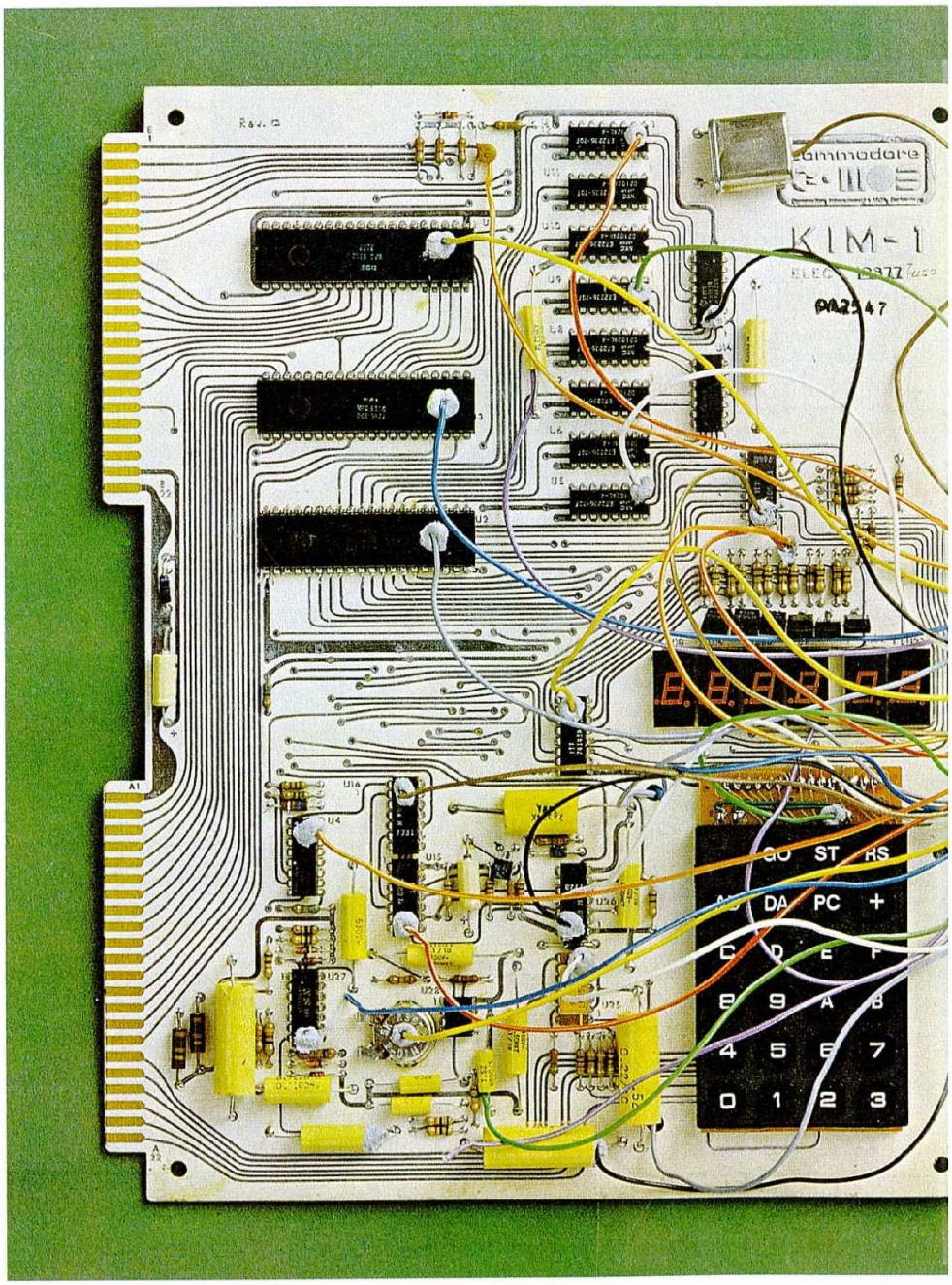
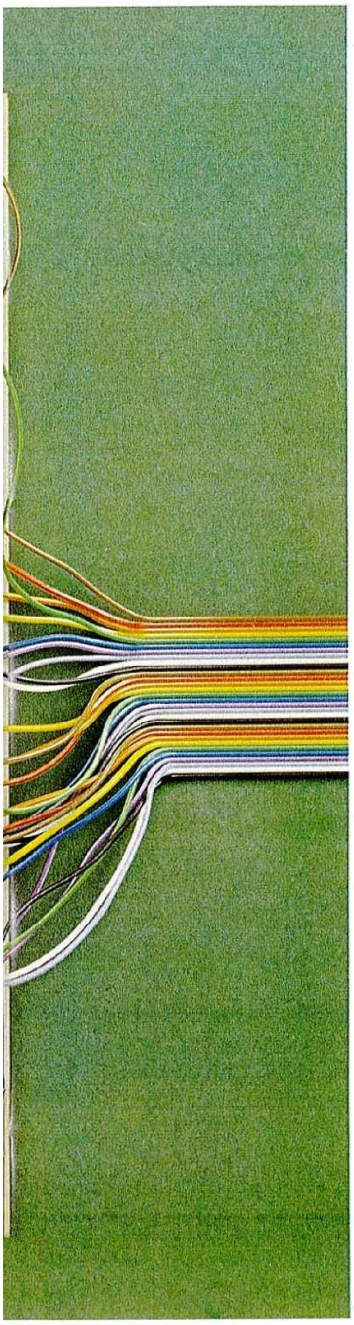

KIM-1
GO ST RS
DA PC +
C D E F
8 9 A B
4 5 6 7
0 1 2 3



KIM-1-Selbstdiagnose:

Stunde der Wahrheit

Jetzt können Sie ihn selbst überprüfen

C. GREVEN

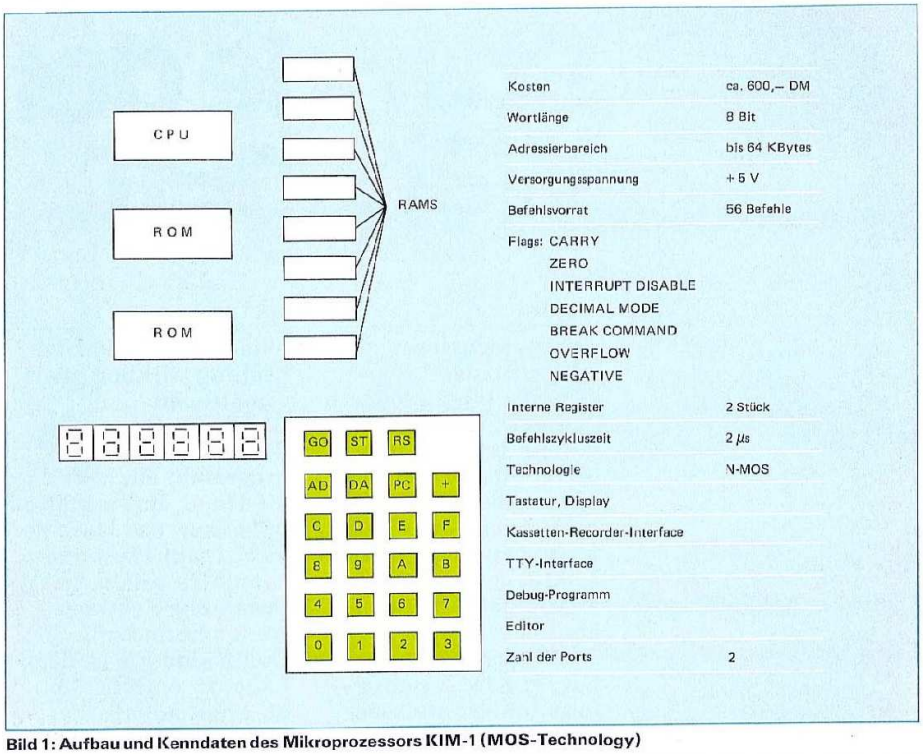
Mikroprozessoren sind in Deutschland schon seit vier Jahren bekannt und auf dem Markt vorgestellt worden. Inzwischen gibt es mehr als 50 verschiedene Modelle. Einer der populärsten ist der 6502 im Mikrocomputer KIM-1 der US-amerikanischen Firma MOS-Technology.
Ob Ihr KIM-1 richtig funktioniert, weiß der Hersteller in der Regel selbst nicht genau — ihm fehlt die Zeit für eine umfassende Überprüfung wirklich aller Funktionen.
CHIP gibt Ihnen mit dem KIM-Prüfprogramm ein Mittel in die Hand, Ihren Mikroprozessor, das Herz des KIM-1, auf Herz und Nieren zu testen und eventuelle Schwachstellen herauszufinden. Damit sind Sie in der Lage, zu entscheiden, ob eventuelle Fehler von der Hard- oder Software verursacht werden.

Der KIM-1 ist ein relativ schneller Mikrocomputer aus der Sippe der 8-bit-Typen. Er hat einen Befehlsvorrat von 56 Befehlen und der Mikroprozessor ist in NMOS-Technologie entwickelt worden, die mit 5 V= Versorgungsspannung auskommt. Weiter verfügt er über zwei P-Eingänge, sogenannte Ports, die per Befehl als Ein- oder Ausgang deklarierbar sind. Um überhaupt Befehle dem Prozessor mitteilen zu können, befindet sich auf der Platine eine Eingabetastatur. Und damit man erkennt, was sich bei Ein- oder Ausgabe so alles tut, sind noch Anzeigebauelemente vorhanden — vier Ziffern davon sind für die Adresse und zwei für den Speicherinhalt (Bild 1). Weiterhin ist noch ein externer Drucker (Teletype) anschließbar. Zur Abspeicherung von Programmen kann außerdem ein handelsüblicher Kassettenrekorder angeschlossen werden.

Welche Fehler gibt es bei einem Mikroprozessor?

Wenn ein Anwender einen Mikroprozessor (µP) in Betrieb nimmt, fragt er sich: 1. Welche Fehler können diese Platine bzw. die CPU (CPU heißt: Central Processor Unit — zentrale Recheneinheit) außer Funktion setzen?
Selbst wenn der Mikroprozessor bei Auslieferung einwandfrei arbeitet und über ein ausgetestetes Programm einen Prozeß steuert, kann er zwischenzeitlich durch Fehler ausfallen. Dies führt zu der zweiten Frage:

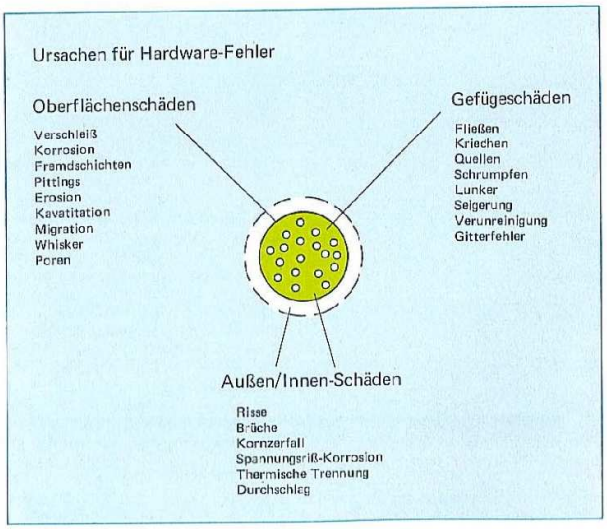
Kann der Anwender auf einfache Weise vor Steuerung eines Prozesses bzw. zwischendurch den Mikroprozessor auf



Kosten	ca. 600,– DM
Wortlänge	8 Bit
Adressierbereich	bis 64 KBytes
Versorgungsspannung	+5 V
Befehlsvorrat	56 Befehle
Flags: CARRY ZERO INTERRUPT DISABLE DECIMAL MODE BREAK COMMAND OVERFLOW NEGATIVE	
Interne Register	2 Stück
Befehlszykluszeit	2 µs
Technologie	N-MOS
Tastatur, Display	
Kassetten-Recorder-Interface	
TTY-Interface	
Debug-Programm	
Editor	
Zahl der Ports	2

Bild 1: Aufbau und Kenndaten des Mikroprozessors KIM-1 (MOS-Technology)

seine Funktionsfähigkeit hin überprüfen?
Bild 2 (links) gibt eine Übersicht über mögliche Hardwarefehler. Danach kann es durch Oberflächenschäden, Gefügeschäden oder Innen-/Außenschäden zur Leitungsunterbrechung bzw. zu einem Kurzschluß kommen. Im Prinzip sind auch intermittierende Signalpegel möglich. Die Hardwarefehler können als sporadische oder Dauerfehler auftreten, ferner als Einfach-, Zweifach- und Folgefehler. Möglich sind passive Fehler (drop-out) und aktive Fehler (drop-in) sowie Fehler bei den Bitmustern.
Bild 2 (rechts) zeigt eine Einteilung der Softwarefehler (nach Kopetz).



Welche Prüfmöglichkeiten gibt es bei einem Mikroprozessor?

Zur Prüfung einfacher wie komplexer digitaler Schaltnetze und Schaltwerke wurden außer den Fehleranalyse-Verfahren in den Werkstoffprüflaboratorien

in den Prüffeldern folgende Prüfverfahren erprobt:
Beim **Funktionstest** (Bild 3) werden unter Betriebsbedingungen die Sollfunktionen überprüft, so z.B. der Ablauf eines bestimmten Steuerprogramms. Dies kann immer nur eine Teilprüfung des Systems darstellen.

Beim **Vergleichstest** beaufschlagt ein Zufallsgenerator das Testobjekt und ein „gutes Muster" (Referenz) mit Signalfolgen. Eine EXOR-Schaltung (exclusive-**or**) liefert bei Abweichungen eine NO GO-Aussage (schlecht), jedoch keine Fehlerortung. Ferner erhebt sich die Frage, ob das „gute Muster" in Ordnung ist.

Testmuster zur Prüfung einer Schaltung können z.B. durch Simulation auf einem Großrechner erstellt werden. Dies erfordert jedoch neben dem Simulator einen Großspeicher. Durch lange Prüfzeiten kommt es zu hohen Prüfkosten.

Die **algorithmische Testerstellung** hat den Vorteil, daß der Mikroprozessor sich die erforderlichen Testmuster selbst erstellt. Jetzt wird der Speicherplatzbedarf geringer. Ferner sind Programmänderungen leicht durchführbar. Kosten verursachen bei dieser Prüfmethode vor allem die Programmierarbeiten.

Die **Selbstdiagnose-Prüfung** gab und gibt es bei den Großrechnern in Form von Probeläufen schon seit Anfang an. Diese Methode läßt sich auch beim Mikroprozessor anwenden, wobei die „intelligenten" Prüftests einfach und kurz sind, aber trotzdem eine hohe Prüftiefe aufweisen. Die nachstehend beschriebenen Tests beziehen sich nur auf die Überprüfung der CPU. Die Selbstdiagnose ist

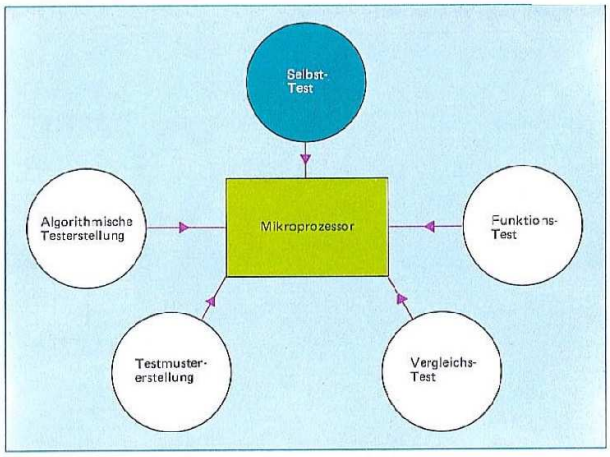


Bild 3: Prüfmöglichkeiten beim Mikroprozessor

jedoch mit geringem Mehraufwand auch auf die Speicherbausteinprüfung und die Kontrolle der Ein-/Ausgabeeinheiten erweiterbar.
Die Tafel (ganz rechts) stellt die Vor- und Nachteile der fünf Prüfmethoden einander gegenüber.

Überprüfung des Befehlssatzes beim KIM-1

Wie Bild 4 (links) zeigt, kann man den Befehlssatz beim KIM-1 in acht Kategorien einteilen. Die Abkürzungen der Befehle, wie z.B.

LDA = Load accumulator with memory
STA = Store accumulator in memory
ADC = Add memory to accumulator with carry
... = ...
... = ...
... = ...

sind dahinter aufgegliedert.
Außer der Befehlsanzahl (beim KIM-1: 56 Befehle) können viele Befehle mehrere Adressierungsarten aufweisen. In Bild 4 (rechts) sind von den 13 möglichen Adressierungsarten vier Adressierungsarten durch eine Skizze erläutert.

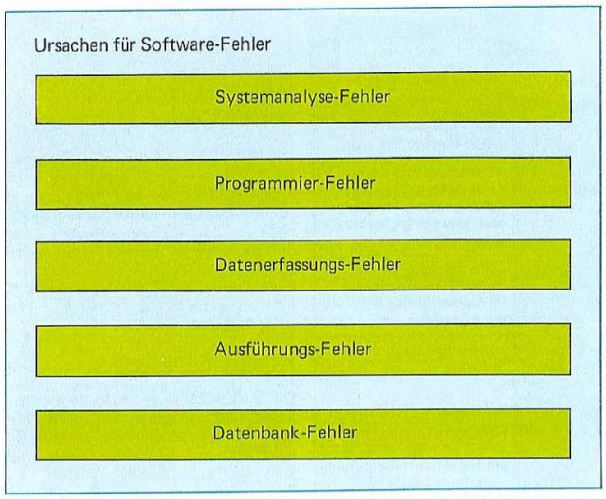


Bild 2: Mögliche Hardware/Software-Fehler bei einem Mikroprozessor

Indirekte Adressierung

In Zelle Ø25Ø steht der indirekte Sprung nach Ø2Ø1. In dieser Zelle ist der erste (niedere) Adreßteil ØØ, im Ø2Ø2 der zweite (höhere) Adreßteil Ø3 gespeichert, so daß das eigentliche Sprungziel die Zelle Ø3ØØ ist. Ändert man den Inhalt der Zellen Ø2Ø1 und Ø2Ø2, so erhält man jeweils andere Sprungziele.

Indizierte indirekte Adressierung

Im Programmablauf ist der Befehl LDA (ØØ, X) gespeichert. Dies bedeutet: Lade den Akkumulator mit dem Inhalt der Zelle, deren Adresse in den Zellen ØØ + X, Ø1 + X gespeichert ist. Ist X = Ø, so wird der Inhalt der Zelle ØØØØ, bei X = 2 der Inhalt der Zelle Ø21Ø und bei X = 4 der Inhalt von Ø2FF geholt. Hierbei wird also die Basisadresse durch den Inhalt des X-Registers verändert.

Stunde der Wahrheit

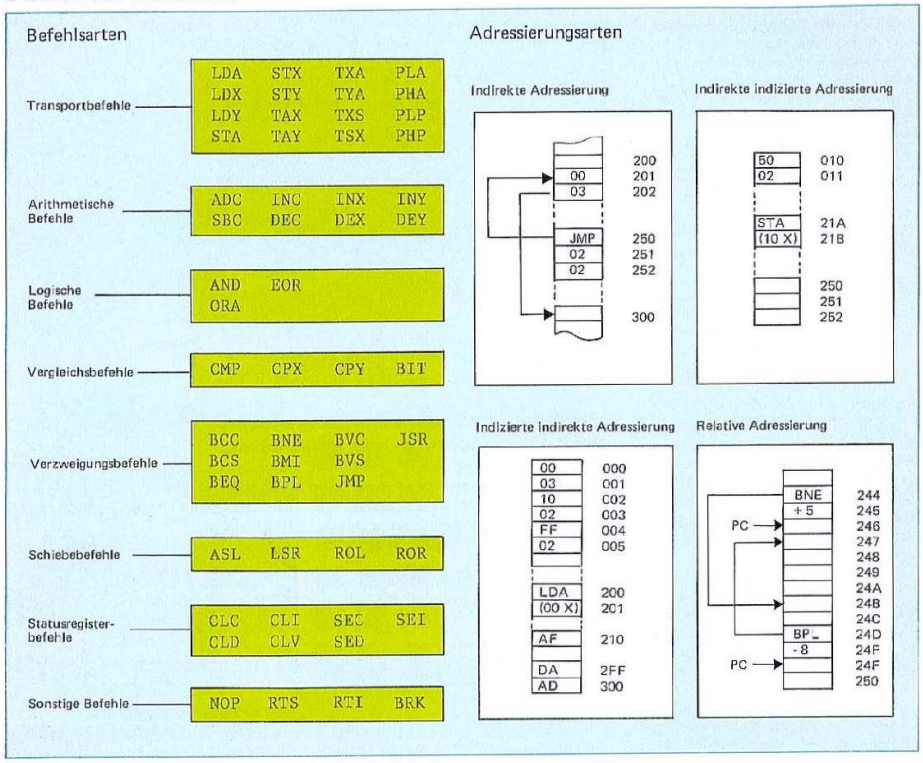


Bild 4: Befehlsarten/Adressierungsarten beim KIM-1

Vor- und Nachteile der fünf Prüfmethoden.

Prüfmethode	Vorteile	Nachteile
1. Funktionstest	Betriebsprüfung	unvollständige Prüfung
2. Vergleichstest	minimaler Hardwareaufwand, keine zusätzliche Software	Referenz-Muster o.k.? Keine Fehlerortung. Durch Zufallsgenerator ungültige oder sinnwidrige Instruktionen
3. Testmustererstellung	gezielte Fehlerdiagnose bei Kenntnis der Schaltung	hoher Speicherbedarf, lange Testzeiten, starres Programm; daher hohe Änderungskosten.
4. Algorithmische Testerstellung	Rechner erzeugt die Testmuster selbst, Diagnosehilfe durch Auswerteprogramm, leichte Programmänderung, mäßiger Speicherbedarf	aufwendige Hard- und Software
5. Selbstdiagnose	mäßiger Programmaufwand, Fehlerortung möglich	nur Befehlssatzüberprüfung

Indirekte indizierte Adressierung

Die Basisadressen beziehen sich immer auf die Seite Ø (Zero Page). Hier wird aber zur Indizierung das Indexregister Y benutzt. Der Inhalt dieses Registers wird nicht zur Basisadresse, sondern zu der dort hinterlegten Adresse hinzuaddiert. Der Befehl STA (1Ø,) Y speichert also den Inhalt des Akkumulators in der Zelle ØØ5Ø ab, wenn Y = Ø ist. Mit Y = 1 in Zelle ØØ51 usw.

Relative Adressierung

Die Zellen Ø244 und Ø245 sollen den Befehl BNE 5 (Springe, wenn nicht Null um 5 weiter) enthalten. Bei der Berechnung des effektiven Sprungzieles muß man vom augenblicklichen Stand des Befehlszählers ausgehen. Dieser steht bei der Entschlüsselung schon auf Zelle

Befehlsarten und ihre Bedeutung beim KIM-1

Transportbefehle

LDA Load accumulator with memory
LDX Load index X with memory
LDY Load index Y with memory
STA Store accumulator in memory
STX Store index X in memory
STY Store index Y in memory
TAX Transfer accumulator to index X
TAY Transfer accumulator to index Y
TXA Transfer index X to accumulator
TYA Transfer index Y to accumulator
TXS Transfer index X to stack pointer
TSX Transfer stack pointer to index X
PLA Pull accumulator from stack
PHA Push accumulator on stack
PLP Pull processor status from stack
PHP Push processor status on stack

Arithmetische Befehle

ADC Add memory to accumulator with carry
SBC Subtract memory from accumulator with borrow
INC Increment memory by one
DEC Decrement memory by one
INX Increment index X by one
DEX Decrement index X by one
INY Increment index Y by one
DEY Decrement index Y by one

Logische Befehle

AND „AND" memory with accumulator
ORA „OR" memory with accumulator
EOR „Exclusive-OR" memory with accumulator

Vergleichsbefehle

CMP Compare memory and accumulator
CPX Compare memory and index X
CPY Compare memory and index Y
BIT Test bits in memory with accumulator

Verzweigungsbefehle

BCC Branch on carry clear
BCS Branch on carry set
BEQ Branch on result zero
BNE Branch on result not zero
BMI Branch on result minus
BPL Branch on result plus
BVC Branch on overflow clear
BVS Branch on overflow set
JMP Jump to new location
JSR Jump to new location saving return address

Schiebebefehle

ASL Shift left one bit (memory or accumulator)
LSR Shift right one bit (memory or accumulator)
ROL Rotate one bit left (memory or accumulator)
ROR Rotate one bit right (memory or accumulator)

Statusregisterbefehle

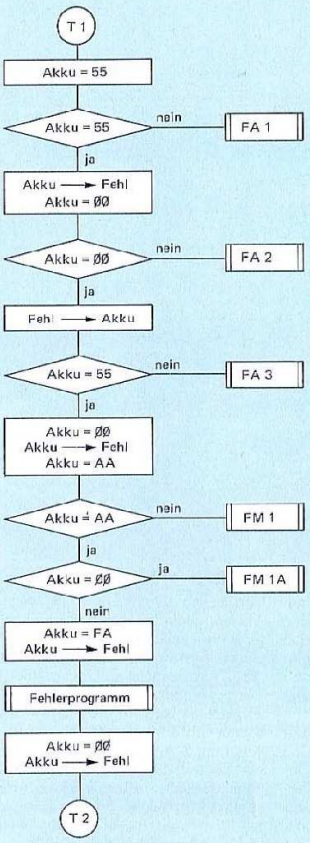
CLC Clear carry flag
CLD Clear decimal mode
CLI Clear interrupt disable bit
CLV Clear overflow flag
SEC Set carry flag
SED Set decimal mode
SEI Set interrupt disable status

Sonstige Befehle

NOP No operation
RTS Return from subroutine
RTI Return from interrupt
BRK Force break

Test 1

Prüfung von LDA, CMP, STA, BNE, BEQ, Flag Z, JSR, RTS




```
Label   Op. Code  Operand
Fehl
Start   LDA       # $ 55
        CMP       # $ 55
        BEQ       T 1A
FA 1    LDA       # $ Ø2
        STA       Fehl
        JSR
T 1A    STA       Fehl
        LDA       # $ ØØ
        CMP       # $ ØØ
        BEQ       T 1B
FA 2    LDA       # $ Ø3
        STA       Fehl
        JSR
T 1B    LDA       Fehl
        CMP       # $ 55
        BEQ       T 1C
FA 3    LDA       # $ Ø4
        STA       Fehl
        JSR
T 1C    LDA       # $ ØØ
        STA       Fehl
T 1D    LDA       # $ AA
        CMP       # $ AA
        BNE       FM 1
        BEQ       T 1E
FM 1    LDA       # $ A9
        STA       Fehl
        JSR
T 1E    CMP       # $ ØØ
        BEQ       FM 1A
        BNE       T 1F
FM 1A   LDA       # $ FØ
        STA       Fehl
        JSR
T 1F    LDA       # $ FA
        STA       Fehl
        JSR
        LDA       # $ ØØ
        STA       Fehl
```

Bild 5: Flußdiagramm/Assemblerprogramm zu Test 1

Ø246. Das Sprungziel ist somit Ø246 + 5 = Ø24B. Dasselbe trifft auch bei Rückwärtsverzweigungen zu. Der in Ø24D und Ø24E gespeicherte Sprung BPL — 8 (springe, wenn positiv um 8 zurück) führt auf die Adresse Ø247 (Ø24F — 8 = Ø247). Bei Rückverzweigung wird die Sprungweite im Zweierkomplement angegeben.

Überprüfung der Grundtestbefehle (T1)

LDA Load accumulator with memory
Lade Akkumulator vom Hauptspeicher

Wenn die MPU den Befehl LDA ausführt, werden Daten vom Hauptspeicher in den Akkumulator übertragen und dort gespeichert. LDA beeinflußt den Inhalt des Akkumulators, jedoch nicht das C-Flag (c-Kennzeichen) und das V-Flag. Wird der Inhalt des Akkumulators Null, wird das Z-Flag gesetzt, andernfalls zurückgesetzt. Das N-Flag wird gesetzt, wenn Bit 7 des Akkumulators = 1 ist, sonst wird es zurückgesetzt.

STA Store accumulator in memory
Speichere den Akkumulatorinhalt im Hauptspeicher

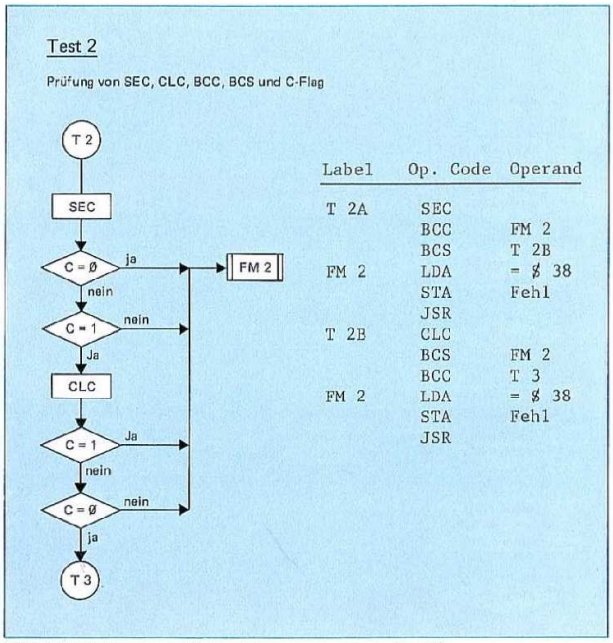


Bild 6: Flußdiagramm/Assemblerprogramm zu Test 2

Dieser Befehl überträgt den Inhalt des Akkumulators in den Speicher. Dieser Befehl hat keinen Einfluß auf den Akkumulator und beeinflußt keine der Flags.

CMP Compare memory and accumulator
Vergleiche Speicherinhalt mit dem Akkumulator
CMP subtrahiert den Inhalt der Speicherzelle vom Inhalt des Akkumulators. Der Akkumulator wird nicht beeinflußt. Bei Gleichheit wird das Z-Flag gesetzt. Das N-Flag wird gesetzt, wenn der Speicher kleiner/gleich dem Akkumulator ist. Ist der Speicherinhalt größer als der Akkumulator, wird das N-Flag zurückgesetzt. Bei Gleichheit wird das C-Flag ebenfalls gesetzt.

Im ersten Test wird geprüft, ob der Akku geladen werden kann. Dazu wird mit $55 ein sogenanntes kritisches Bitmuster geladen. Es entspricht einer Bitfolge von Ø1Ø1Ø1Ø1. Wird durch den Vergleichsbefehl festgestellt, daß der Akku mit $ 55 geladen war, wird der Inhalt des Akkus in der Adresse „Fehl" abgespeichert. Nun wird der Akku mit $ ØØ geladen und ebenfalls auf seinen Inhalt abgefragt, damit man sicher sein kann, daß der Inhalt des Akkus verändert wurde. Nachdem diese Abfrage erfolgt ist, wird der Akku mit dem Inhalt der Speicheradresse „Fehl" geladen. Durch diesen Vorgang wird sichergestellt, daß die Adresse „Fehl" ansprechbar ist. Um am Ende des Tests erkennen zu können, ob ein Fehler aufgetreten ist, wird die Adresse „Fehl" wieder mit $ ØØ geladen. Nun wird der Akku mit $ AA geladen. Dieses Muster wurde deshalb gewählt, weil es dem umgekehrten Muster $ 55 entspricht, nämlich 1Ø1Ø1Ø1Ø. Durch diese Wahl wird sichergestellt, daß kein Bit im Akku fest auf Ø oder 1 bleibt. Im nächsten Teil des Tests wird die Funktion des Bits Z im Prozessor-Status-Register auf seine einwandfreie Funktion überprüft. Dieses Flag Z wird während jeder Operation automatisch gesetzt, wenn das Ergebnis der Operation Ø ist.

Das Bit ist 1, wenn das Ergebnis im Akku Ø ist. Der Zustand von Z wird durch die Befehle BNE und BEQ abgefragt. Nach dem Laden des Akkumulators mit $ AA und einem anschließenden Vergleich mit $ AA wird bei Gleichheit das Z-Flag gesetzt, andernfalls nicht gesetzt oder gelöscht. Durch den BNE-Befehl wird bei Ungleichheit eine Fehlermeldung ausgeführt. Da durch den Compare-Befehl der Inhalt des Akkus nicht geändert wurde, wird nun erneut ein Vergleich des Akkuinhalts, der $ AA enthält, mit dem Bitmuster $ ØØ vorgenommen. Da diese beiden Bitmuster unterschiedlich sind, muß also das Z-Flag = Ø sein. Der BEQ-Befehl veranlaßt eine Verzweigung bei einem Ergebnis ungleich Ø. Es wird dann eine Fehlermeldung vorgenommen. Durch den BNE-Befehl erfolgt eine Verzweigung bei einem Ergebnis ungleich Ø.

Tritt innerhalb dieses Programms kein Fehler auf, kann man sicher sein, daß der Akku über den Datenbus ansprechbar ist und im Akku kein Bit fest auf Ø oder 1 bleibt, das Z-Flag betriebsbereit ist, also je nach Ergebnis von Operationen in der Lage ist, von Ø auf 1 oder umgekehrt von 1 auf Ø zu springen. Weiterhin kann man sagen, daß die beiden Sprungbefehle BNE und BEQ richtig ausgeführt wurden und die Speicheradresse „Fehl" ansprechbar ist. Die beiden Befehle BNE und BEQ brauchen nicht mehr mit einer anderen Adressierungsart überprüft zu werden, weil sie nur eine Adressierungsart, nämlich die Adressierungsart „relativ", haben.

Beim Test T 1 wurde als Voraussetzung angenommen, daß der Befehl JSR richtig ausgeführt wird, der bei einem eventuell aufgetretenen Fehler den Sprung zum Fehleranzeigeprogramm vorgenommen hätte. Nun soll aber auch dieser Befehl noch getestet werden, weil er in allen weiteren Tests verwendet wird. Dieser Test hat die Aufgabe, die Ausführung des richtigen Sprunges und die richtige Abarbeitung des Fehleranzeigeprogramms zu gewährleisten. Bei richtiger Abarbeitung des Fehleranzeigeprogramms wird ein Rücksprung ins Hauptprogramm vorgenommen und der Test bei T 2 fortgesetzt.

Überprüfung des Flag C und der bedingten Sprungbefehle BCC und BCS (T 2)

Carry (Übertrag) wird aufgrund spezieller arithmetischer Operationen oder durch „Setze Carry" oder „Lösche Carry" modifiziert. Mit den bedingten Sprungbefehlen BCC und BCS kann der Inhalt des Flag C abgefragt werden. Die Befehle haben folgende Bedeutung:

BCC Branch on carry clear
Verzweige bei gelöschtem C-Flag
BCC fragt den Zustand des Carry-Flag ab und führt einen bedingten Sprung aus, sofern das Carry-Flag zurückgesetzt bzw. gelöscht ist.

BCS Branch on carry set
Verzweige bei gesetztem Carry-Flag
BCS führt einen bedingten Sprung aus, wenn das Carry-Flag gesetzt ist.

Diese beiden Befehle beeinflussen keine Flags oder Register, sondern nur den Programmzähler, und diesen auch nur, wenn die entsprechende Bedingung er-

füllt ist. Zur Überprüfung auf Fehlermöglichkeiten werden zuerst die OP-Kodes der einzelnen Carry-Befehle betrachtet:

SEC : 38 ≙ 0011 1000
CLC : 18 ≙ 0001 1000
BCC : 90 ≙ 1001 0000
BCS : B0 ≙ 1010 0000

Bei diesen vier Befehlen sind folgende Fehler möglich:

1. Nach einem SEC-Befehl wird das Flag C nicht gesetzt oder zurückgesetzt, was der Wirkung eines richtig ausgeführten CLC-Befehls gleichkommt. Dies würde eine Fehlerinterpretation des Bits b_5 voraussetzen.
2. Nach einem CLC-Befehl wird das Flag C nicht gelöscht, wenn es gesetzt war bzw. gesetzt, wenn sie gelöscht war. Dies entspricht einem richtig ausgeführten SEC-Befehl. Es liegt wieder eine falsche Interpretation des Bits b_5 vor.
3. Beim Befehl BCC kann als Fehler ein Sprung bei gesetztem C-Flag erfolgen. Daraus ergibt sich eine richtige Ausführung des BCS-Befehls, was ebenfalls einer falschen Deutung des Bits b_5 gleichkommt.
4. Beim Befehl BCS kann ein Sprung gemacht werden, wenn das C-Flag gelöscht ist. Es ergibt sich wieder eine falsche Auslegung des Bits b_5.

Die angenommenen Fehler werden immer durch das Bit b_5 verursacht. Diese Fehler können mit einem Defekt in der Dekodierlogik oder einem Festlieger einer Datenleitung, und zwar in diesem Fall der Datenleitung DB 5 begründet werden. In diesem Test wird das C-Flag durch den Befehl SEC gesetzt. Durch den Befehl BCC (springe, wenn C = 0) wird das C-Flag abgefragt. Ist C = 0, wird eine Fehlermeldung vorgenommen. Ist C = 1, was durch den Befehl BCS geprüft wird, springt das Programm zum Testteil T 2 B. Andernfalls wird eine Fehlermeldung vorgenommen. Im Testteil T 2 B wird das C-Flag durch den Befehl CLC gelöscht. Durch den Befehl BCS (springe, wenn C = 1) wird die erste Abfrage vorgenommen. Bei C = 1 und bei C = 0, wird ein Fehler gemeldet. Das wird durch den Befehl BCC (springe, wenn C = 0) geprüft. Dann erst wird der Test T 2 B abgeschlossen und es folgt Test T 3. Tritt in dem Test T 2 kein Fehler auf, kann man mit großer Wahrscheinlichkeit annehmen, daß die Datenleitung DB 5 nicht fest auf 0 oder 1 liegt.

Ausdrucken einer Fehlermeldung

An den KIM-1 kann ein Teletype (Drucker + Eingabe) angeschlossen werden. Damit erhält man zusätzlich ein Prüfdokument. Bild 7 zeigt den Ausdruck von Test 1. Die ersten drei Ziffern sind Prüf-

```
TEST   T 1
**********

KIM
0001 A9 17F7
17F7 54 54.
17F8 00 00.
17F9 FF 0001
0001 A9 Q
;18000A955C955F007A902850020000285O0A900C900F007A90385089D
;18001900200002A500C955F007A90485002000O2A9008500A9AAC907AB
;18003AAD002FC07A9A9B500200002C900F002D007A9F085002000985
;18004902A9FA8500200002A900850D200002D95D5F199F1D1559FH07D0
;000004 0004

0000 00            ADRESSE 0000      FEHLER 00

FEHLERSIMULATION: FALSCH AUSGEFUEHRTER LADEBEFEHL

KIM
0001 A9 17F7
17F7 54 54.
17F8 00 00.
17F9 FF 0001
0001 A9 Q
;18000A955C955F007A902850020000285O0A900C900F007A90385089D
;18001900200002A500C955F007A90485002000O2A9008500A9AAC907AC
;18003AAD002FC07A9A9B500200002C900F002D007A9F085002000985
;18004902A9FA8500200002A900850D200002D95D5F199F1D1559FH07D0
;000004 0004

0000 A9            ADRESSE 0000      FEHLER A9

FEHLERSIMULATION: FALSCH AUSGEFUEHRTER COMPAREBEFEHL

KIM
0001 A9 17F7
17F7 54 54.
17F8 00 00.
17F9 FF 0001
0001 A9 Q
;18000A955C955F007A902850020000285O0A900C900F007A90385089D
;18001900200002A500C955F007A90485002000O2A9008500A9AAC907AB
;18003AA0002FC07A9A9B500200002C900F002D007A9F085002000986
;18004902A9FA8500200002A900850D200002D95D5F199F1D1559FH07D0
;000004 0004

0000 A9            ADRESSE 0000      FEHLER A9
```

Programmanfang

Programmende

Prüfbits + Adresse

Prüfbits

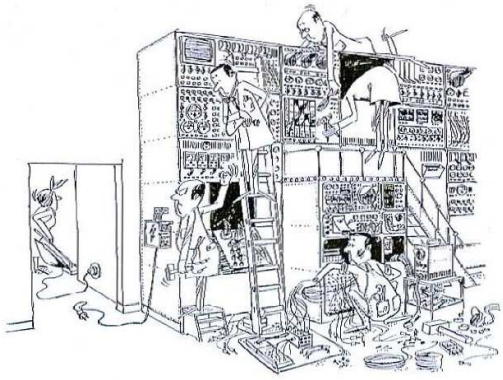
Bild 7: Beispiel einer Fehlermeldung beim Test 1

```
TEST   T 2
**********

KIM
0001 38 17F7
17F7 54 18.
17F8 00 00.
17F9 FF 0001
0001 38 Q
;18000389002BC07A9388500200002I8E0029007A9388500200002060B
;000001 0001

0000 00            ADRESSE 0000      FEHLER 00

FEHLERSIMULATION: C-FLAG WIRD NICHT GESETZT

KIM
0001 18 17F7
17F7 18 18.
17F8 0C 00.
17F9 FF 0001
0001 18 Q
;18000189002B007A9388500200002I8E0029007A9388500200C02O5EB
;000001 0001

0000 38            ADRESSE 0000      FEHLER 38
```

Bild 8: Beispiel einer Fehlermeldung beim Test 2

Nr.	Prüfziel	Fehlermeldung
1	LDA, CMP, STA, BNE, BEQ, Flag Z, JSR, RTS	FA 1, FA 2, FA 3, FM 1, FM 1A
2	SEC, CLC, BCC, BCS, Flag C	FM 2
3	BPL, BMI, Flag N	FM 3
4	JMP	FM 4
5	LDX, CPX, STX	FA 4, FA 5, FA 6, FM 5
6	LDY, CPY, STY	FA 7, FA 8, FA 9, FM 6
7	TAX, TXA	FM 7
8	TAY, TYA	FM 8
9	INX, DEX	FM 9
10	INY, DEY	FM 10
11	INC, DEC	FM 11
12	TXS, TSX	FM 12
13	PHA, PLA	FM 13
14	CLV, BVC, BVS, Flag V, ADC	FM 14A, FM 14
15	ADC	FM 15
16	SBC	FM 16
17	SBC	FM 17
18	AND	FM 18
19	ORA	FM 19
20	EOR	FM 20
21	LSR	FM 21, FM 21A
22	ASL	FM 22, FM 22A
23	ROL	FM 23, FM 23A
24	ROR	FM 24, FM 24A
25	BIT	FM 25

Bild 9: Prüfergebnis mit Fehlermeldungen der 25 Tests



„Sofort aufhören! Der Fehler liegt hier!"
Aus maul + computer-Späße

bits. Danach kommt die vierstellige Adresse, gefolgt vom eigentlichen Programm. Im Bild 7 (oben) wird kein Fehler simuliert. Ergebnis: kein Fehler. Bild 7 (Mitte) zeigt die Fehlererkennung bei einem falsch ausgeführten Ladebefehl. Bild 7 (unten) zeigt die Meldung eines falsch ausgeführten Comparebefehls (Vergleich): Anstelle von AA (Kreis) wurde AB eingetippt. Bild 8 zeigt nun einen fehlerfreien Test und eine (gekennzeichnete) Fehlermeldung, da das C-Flag nicht gesetzt worden war.

Wie sind die Tests zu bewerten?

Prüfziele/Fehlermeldungen

Bild 9 stellt in zwei Spalten die bei den 25 Testprogrammen überprüften Befehle und die dabei möglichen Fehlermeldungen zusammen. FA 1, 2 ... usw. sind die Fehleradreßmeldungen, also Hinweise auf den Fehlerort im Programm. FM 1, 2 ... usw. zeigen die Fehlerart: FM 1 ≙ LDA-Fehler.

Prüfrosetten

Um dem Leser einen anschaulichen Einblick in das Testgeschehen zu vermitteln, wurden bei jedem Test „Prüfrosetten" gekennzeichnet: Der Kreis ist in 56 Segmente unterteilt, gruppiert in acht Befehlskategorien. Die Striche bedeuten, daß bei dem Test dieser Befehl benutzt wurde, während die Zahl die Häufigkeit angibt. In Bild 10 (links) wurden in einer Prüfrosette alle bei den 25 Tests benutzten Befehle eingetragen. Wie man sieht, sind fast alle Befehlsbereiche angesprochen worden.

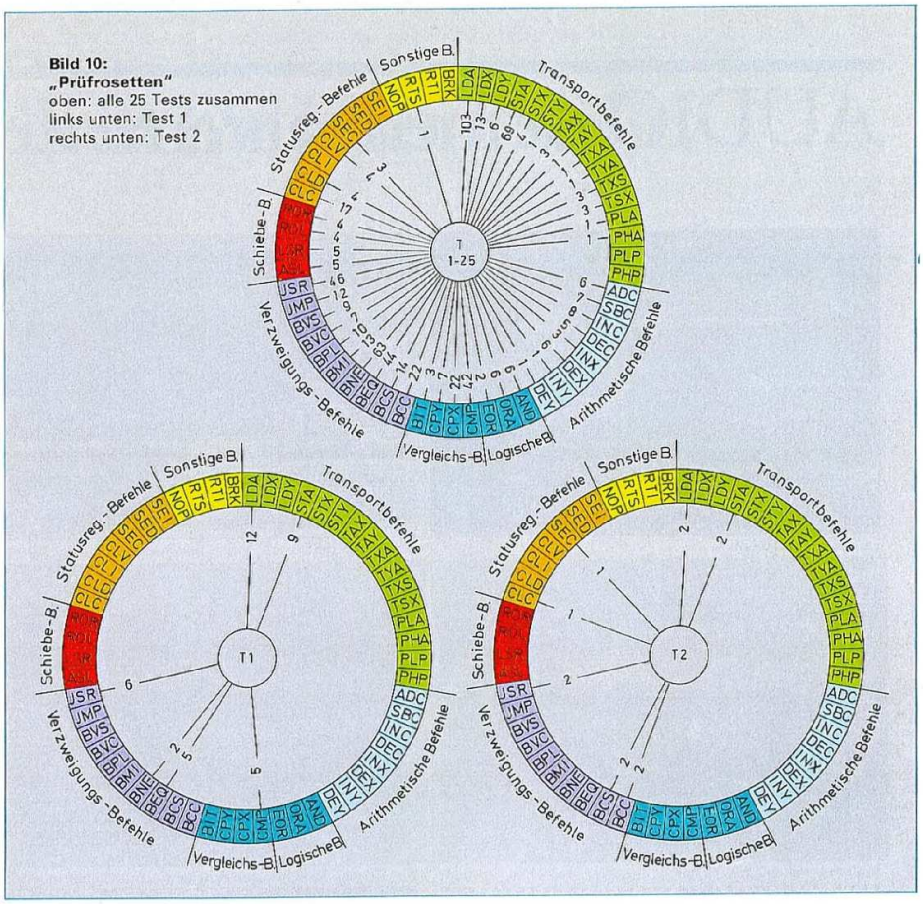
Verzeichnis der überprüften Befehle

In Bild 11 wurden alle Befehle nach Befehlskategorien aufgegliedert. Bei der Berücksichtigung der Adressierungsarten sind es insgesamt 151 Befehle. Davon wurden 62 Befehle in den 25 Tests geprüft, also rund 41% des Befehlssatzes.

Aussagekraft der µP-Selbstprüfung

Die Tafel in Bild 11 berücksichtigt weder die Prüfzeit noch die Häufigkeit der benutzten Befehle im Testprogramm. Würde man alle 56 Befehle und die sich (nach den Regeln der Kombinatorik) ergebenden Möglichkeiten abprüfen, so würde man bei einer minimalen Zykluszeit von 2 µs 4570 Jahre an Testzeit benötigen. Diese Angabe ist aber nicht sinnvoll, weil die Anzahl wirklich richtiger Kombinationen wesentlich geringer ist. Die Prüfzeit aller 25 Tests beträgt weniger als 100 ms (96,512 ms).

Bild 10:
„Prüfrosetten"
oben: alle 25 Tests zusammen
links unten: Test 1
rechts unten: Test 2



Befehlsart	Anzahl der Befehle	geprüft	Prozent	Adressierungs-arten	mögliche Befehle	geprüft	Prozent
Transferbefehle*	16	14	87,5	10	41	19	46,3
Arithmetische Befehle	8	8	100,0	9	28	10	35,7
Logische Befehle	3	3	100,0	8	24	5	20,8
Vergleichsbefehle	4	4	100,0	8	16	5	31,3
Verzweigungsbefehle	10	10	100,0	3	11	10	90,9
Schiebebefehle	4	4	100,0	5	20	8	40,0
Status-Register-Befehle	7	4	57,1	1	7	4	57,1
Sonstige Befehle	4	1	25,0	1	4	1	25,0
Gesamt	56	48	85,7	13	151	62	41,1

Bild 11: Verzeichnis der überprüften Befehle

* Ein Transferbefehl kann bis zu 8 verschiedene Adressierungsarten aufweisen.