THE LOVELY COUPLE OF QUICKDUMP AND VERSALOAD

Double speed loader, relocating and linking.

By John Oliver and Roland Löhr.

E: John Oliver, Ass. Professor of Astronomy at the University of Florida, doubled the speed of writing and reading magnetic tape as compared to Jim Butterfield's HYPERTAPE by coding each byte with only 8 bits instead of two ASCII characters. Prof. Oliver gave kind permission to reprint his SUPERDUMP and SUPERLOAD in this journal. The editor nevertheless decided to put the nucleus of these into another frame in order to win new features and to combine it with other useful utilities.

Thankful acknowledgement is made to Mr. Oliver for co-authoring his grand idea which formed the basis for all of this:

By now we have writing or reading 1 kbytes in about 12 seconds, input to old locations or relocated, open end input or protection of memory by buffer, DIRECTORY, DUPE, calling records by single-byte ID or by name (header of any 6 bytes). And together with the editor's RALOAD (first issue of 65xx MICRO MAG): Relocation of programs to new address space, a linking loader with same transformation. Last, not least the option to introduce 'external parameters' aside from header.

QUICKDUMP and VERSALOAD are a dependent pair, they are not compatible with other recording formats.

* * *

Jim Butterfield's HYPERTAPE beschleunigt das Bandschreiben und -laden um den Faktor 6. Sehr nützlich ist auch sein SUPER DUPE (Kopieren von Bandcassetten) und sein DIRECTORY (Lesen von Startadresse und ID vom Band ohne zu laden).

Den nächsten großen Schritt machte John Oliver, Astronomie-Professor an der University of Florida, Williamson Hall, Gainesville FL 32611, mit seinen Programmen SUPER DUMP und SUPERLOAD, zuerst veröffentlicht in den KIM User Notes 7/8. Im KIM-Monitor und auch noch in HYPERTAPE wird jedes Zeichen zunächst in 2 ASCII-codierte Sequenzen zerlegt und gesendet. John Oliver verdoppelt die Geschwindigkeit auf etwa 1 kBytes in 12 Sekunden, indem er je Byte nur einmal 8 Bits sendet; eine klare logische Konsequenz, auf die jemand erst kommen mußte.

Wie beim KIM, so wird auch hier das einzelne bit durch verschiedene und verschieden lange Frequenzen auf dem Magnetband abgebildet. (Zur Erläuterung: s. KILOBAUD, Heft 11/77, S. 66 ff.). Soweit als möglich verwenden beide Autoren Unterprogramme des KIM-Monitors und trixen sie z.T. aus.

John Oliver erteilte diesem Journal freundlichst Nachdruckrechte seiner Programme. Wir haben ihm dafür herzlich zu danken, denn sein Brief ermutigte in mehr als einwöchiger intensiver Arbeit eigene Weiterentwicklungen, die hier als QUICKDUMP und VERSALOAD präsentiert werden.

Um das Herzstück der utilities SUPERDUMP und SUPERLOAD baute der Herausgeber einen Rahmen, der sicher fortschrittlich ist:

VERSALOAD ist nicht nur ein reines Ladeprogramm, sondern zugleich auch Kopierprogramm (wie SUPER DUPE), Inhaltsverzeichnis (wie DIRECTORY), Etikettensucher (wie HEADHUNTER des Herausgebers) und es ist neben anderem auch ein

linking and/or relocating program loader.

Diese Eigenschaft dürfte die schönste und bequemste von allen sein. VERSALOAD nutzt die Dienste des in Heft 1 des 65xx MICRO MAG abgedruckte RALOAD (Verschiebung und Umrechnung): Programmsegmente werden jeweils ab nächstfolgender freier Speicherzelle geladen und für den neuen Adressenraum umgerechnet (zur notwendigen 'Syntax' s. Heft 1).

Nützlich ist auch die mögliche Festlegung eines Eingabepuffers. Ein Speicherbereich kann ab einer in Loc. 17F7/F8 festgelegten Anfangsadresse gegen unbeabsichtigtes Überschreiben durch das einzulesende Programm geschützt werden.

Programme oder Datensätze können wahlweise mit 1-Byte-ID oder mit einem Standard-Label (Name von 6 Bytes) versehen sein. VERSALOAD erkennt die gewählte Form der Identifizierung und sucht entsprechend nac Gleichheit. Es erkennt auch, ob ein Bandsatz darüber hinaus zusätzliche Bytes (bis 249) als Option mit sich führt, um z.B. Namen und Speicherplatz benötigter oder abzugebender 'externer Parameter' zu beschreiben.

QUICKDUMP und VERSALOAD sind als Unterprogramme geschrieben. Sie ermöglichen kontinuierliches, von der Maschine her gesteuertes Arbeiten (JSR ENTRY mit Parametern in A bzw. in X). Bei Aufruf mit einer der Kopfzeilen kann jedoch ebenso einmaliges Abarbeiten mit Rückkehr zum KIM-Monitor bewirkt werden. Das Datenformat beim Schreiben und das Blockdiagramm des Leseprogrammes sind auf den folgenden Seiten abgedruckt.

Dieses Programm-Paar ist mit anderen Aufzeichungsformen nicht kompatibel

In der Summe haben wir jetzt Dienstleistungen zur Hand, die schon denen einer größeren Datenverarbeitung ähneln. Insbesondere sei auch auf Möglichkeiten des Overlay hingewiesen: Bei begrenztem Speicher können lange Programme in Segmente zerlegt werden, die bei Bedarf in einen Puffer (Overlaybereich) nachgezogen werden.

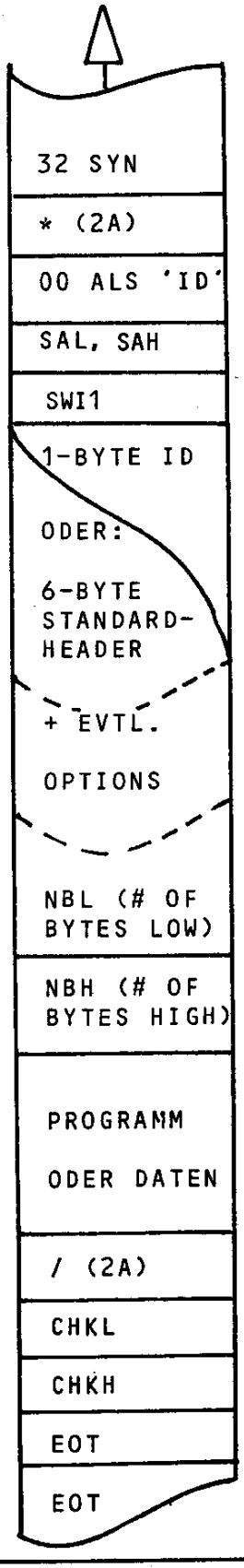
* * *

```
                        QUICKDUMP

0600   A9 00      STARTA  LDA #$ 00      CALL WITH NO HEADER
0602   AA                 TAX            00 TO X FOR SWITCHING
0603   F0 04              BEQ GOSUB      BRANCH ALWAYS
0605   A9 05      STARTB  LDA #$ 05      CALL WITH 6-BYTE HEADER
0607   A2 00              LDX #$ 00      FOR SWI2
0609   20 0F 06   GOSUB   JSR ENTRY1     CALL MAIN PGM AS A SUBROUTINE
060C   4C 4F 1C           JMP KIM        RETURN TO MONITOR

                        SUBROUTINE (MAIN)

060F   85 D4      ENTRY1  STA SWI1       SAVE PARAMETERS
0611   86 D5              STX SWI2       FROM START
0613   A2 04              LDX #$ 04      COUNTER FOR TRANSPORT
```

```
0615  BD 29 07  STOTAB  LDA TAB-1,X    PUT TRAILING TABLE TO
0618  95 CF             STA NPUL-1,X   ZEROPAGE
061A  CA                 DEX
061B  D0 F8              BNE STOTAB     DONE?

061D  A9 AD      SUPERD  LDA #$ AD      "STA"-OPCODE FOR VEB. MAIN BODY
061F  8D EC 17           STA VEB        OF JOHN OLIVERS PROGRAM
0622  20 32 19           JSR INTVEB     INITIALIZE VEB
0625  A9 27               LDA #$ 27
0627  85 CC              STA GANG       SBD OUTPUT WORD
0629  A9 BF              LDA #$ BF      OPEN CHANNELS
062B  8D 43 17           STA SBD
062E  A9 20              LDA #$ 20      SEND 32 SYNC CHARACTERS
0630  85 CD              STA TIC        SAVE CHAR COUNT
0632  A9 16              LDA #$ 16      ....SYNC....
0634  48         HIC1    PHA            SAVE THIS CHARACTER
0635  20 F2 06           JSR OUTCHT     SEND
0638  68                 PLA            RESTORE CHARACTER
0639  C6 CD              DEC TIC        REDUCE COUNTER
063B  D0 F7              BNE HIC1       FINISHED?

063D  A9 2A              LDA #$ 2A      TO SEND "*"
063F  20 F2 06           JSR OUTCHT

0642  A9 00              LDA #$ 00      DUMMY-ID
0644  20 D0 06           JSR OUTBT      SEND 2 ASCII
0647  A5 D5              LDA SWI2       DOES VERSALOAD REQUEST A DUPE?
0649  D0 0F              BNE DUP1       YES
064B  AD F5 17           LDA SAL        TAKE ORIGINAL VALUES
064E  20 D0 06           JSR OUTBT      AND SEND
0651  AD F6 17           LDA SAH
0654  20 D0 06           JSR OUTBT
0657  4C 64 06           JMP SENDID     SKIP

065A  A5 D6      DUP1    LDA OSAL       TAKE INSTEAD AND SEND,
065C  20 D0 06           JSR OUTBT      VALUES SUPPLIED BY VERSALOAD
065F  A5 D7              LAD OSAH
0661  20 D0 06           JSR OUTBT

0664  A5 D4      SENDID  LDA SWI1       SWI1 = 00 OR 05 OR OPTIONAL
0666  20 D0 06           JSR OUTBT      LENGTH OF TABLE [EXTERNAL PARMS]
0669  A5 D4              LDA SWI1       DECIDE TO SEND 1-BYTE ID OR
066B  D0 09              BNE SENDTB     A HEADER

066D  AD F9 17           LDA ID         SEND 1-BYTE ID
0670  20 F2 06           JSR OUTCHT
0673  4C 8A 06           JMP NUMBOB     SKIP ALWAYS

0676  A9 00      SENDTB  LDA #$ 00      SWI2 INSTALLED NOW AS COUNTER
0678  85 D5              STA SWI2
067A  A6 D5      LOSWI   LDX SWI2
067C  BD 80 17           LDA 1780,X     HEADER TABLE HERE INSTALLED
067F  20 F2 06           JSR OUTCHT     AND SENT
0682  E6 D5              INC SWI2       UPCOUNT...
0684  A5 D4              LDA SWI1       TO BE COMPARED
0686  C5 D5              CMP SWI2
0688  B0 F0              BCS LOSWI      IF SWI1 ≥ SWI2
```

```
068A  38        NUMBOB  SEC             PREPARE SBC TO CALCULATE
068B  AD F7 17          LDA EAL         NUMBER OF BYTES
068E  ED F5 17          SBC SAL         GIVING NBL
0691  08                PHP             SAVE CARRY STATUS
0692  20 F2 06          JSR OUTCHT      SEND NBL

0695  28                PLP             RESTORE CARRY STATUS
0696  AD F8 17          LDA EAH
0699  ED F6 17          SBC SAH         GIVING NBH
069C  20 F2 06          JSR OUTCHT      & SEND
069F  20 43 19          JSR INTVEB+17   RESET CHECKSUM TO 00

06A2  20 EC 17  SUPDP1  JSR VEB         GET BYTE ADDRESSED BY VEB...
06A5  20 EF 06          JSR OUTCHC      ...AND SEND IT
06A8  20 EA 19          JSR INCVEB      ADDRESSING + 1 FOR NEXT BYTE
06AB  AD ED 17          LDA VEB+1       ARE WE AT ...
06AE  CD F7 17          CMP EAL         ... END ADDRESS?
06B1  AD EE 17          LDA VEB+2
06B4  ED F8 17          SBC EAH
06B7  90 E9             BCC SUPDP1      NOT FINISHED, GET MORE
06B9  A9 2F             LDA #$ 2F       SEND "/"
06BB  20 F2 06          JSR OUTCHT

06BE  AD E7 17          LDA CHKL        SEND CHECKSUM
06C1  20 D0 06          JSR OUTBT
06C4  AD E8 17          LDA CHKH
06C7  20 D0 06          JSR OUTBT
06CA  A9 04             LDA #$ 04       EOT CHARACTER
06CC  20 F2 06          JSR OUTCHT
06CF  60                RTS             GOBACK

                        SUBROUTINES DIVISION
06D0  48        OUTBT   PHA             HEX OUTPUT ROUTINE: SAVE BYTE
06D1  4A 4A             LSR, LSR        ISOLATE MSD
06D3  4A 4A             LSR, LSR
06D5  20 E3 06          JSR HEXTA       & WRITE AS ASCII
06D8  20 F2 06          JSR OUTCHT
06DB  68                PLA             RESTORE BYTE
06DC  20 E3 06          JSR HEXTA       GET 4 LSB AND WRITE AS ASCII
06DF  20 F2 06          JSR OUTCHT
06E2  60                RTS

06E3  29 0F     HEXTA   AND #$ 0F       MASK OFF 4 LSB
06E5  C9 0A             CMP #$ 0A
06E7  18                CLC
06E8  30 02             BMI HEXTA1
06EA  69 07             ADC #$ 07       A TO F
06EC  69 30     HEXTA1  ADC #$ 30       Ø TO 9
06EE  60                RTS

06EF  20 4C 19  OUTCHC  JSR CHKT        CHECKSUM CALCULATION
06F2  A0 08     OUTCHT  LDY #$ Ø8        SET FOR 8 BITS
06F4  84 CE             STY COUNT       SAVE BIT COUNT
06F6  A0 02     TRY     LDY #$ 02       SET FOR 3 PHASES
06F8  84 CF             STY TRIB        SAVE PHASE COUNT
06FA  B6 D0     ZON     LDX NPUL,Y      # OF 1/2 CYCLES
06FC  48                PHA             SAVE CHARACTER
06FD  78        ZON1    SEI             DISABLE INTERRUPTS
```

QUICKDUMP
Aufzeichungs-
format der
Bandsätze



Weitere Hinweise zu QUICKDUMP

Schreiben eines Bandes mit 1-Byte-ID:

Startadresse	SAL/SAH	nach 17F5/F6
Endadresse+1	EAL/EAH	nach 17F7/F8
Identität	ID	nach 17F9
Start	STARTA	in 0600

Schreiben eines Bandes mit 6-Byte-Namen (Standard header):

Startadresse und Endadresse wie vor

Identität	Header nach 1780-1785 oder bei Veränderung der Adresse in LOSWI woanders.
Start	STARTB in 0605

Schreiben eines Bandes zusätzlich mit 'externen Parametern':

Startadresse, Endadresse und standard header wie vor.

```
LDA wirkliche Länge der in 1780 ... gespeicherten Tabelle.
LDX #$ 00
JSR ENTRY1
...
```

Die Aufzeichungsgeschwindigkeit kann mit den für 072A und 072C alternativ vorgesehenen Wertepaaren herabgesetzt werden. Die in TAB eingestellten Werte entsprechen x6 mit doppelter Schreibdichte. In Heft 10/11 der KIM-User Notes hatte John Oliver noch vorgeschlagen, das Wertepaar gegeneinander auszutauschen. Diese Empfehlung gilt heute nicht mehr. Sollten beim Lesen irgendwelche Schwierigkeiten auftreten, so wird stattdessen eine sorgfältige Einstellung des PLL-Potentiometers empfohlen. Dazu schreibt man eine größere Zahl kurzer Sätze auf ein Band und regelt beim Lesen solange ein, bis es eindeutig funktioniert.

ZERO PAGE USED BY QUICKDUMP & VERSALOAD

00C9 EALB	00D0 NPUL		00D6 OSAL
00CA EALH	00D1 TIMG		00D7 OSAH
00CB LFLG	00D2 "		00D8 CNTRL
00CD TIC	00D3 "		00D9 IDTEMP
00CE COUNT	00D4 SWI1		00E2 EOPL
00CF TRIB	00D5 SWI2		00E3 EOPH

```
06FE  2C 47 17  ZON2    BIT CLKRD1      TIMER DONE?
0701  10 FB             BPL ZON2        NO, WAIT
0703  B9 D1 00          LDA TIMG,Y      GET WAIT TIME IN ...
0706  8D 44 17          STA CLK1T       MICROSECONDS FOR TIMER
0709  A5 CC             LDA GANG        FLIP OUTPUT BIT ...
070B  49 80             EOR #$ 80       BETWEEN Ø AND 1
070D  8D 42 17          STA SBD         OUTPUT BIT
0710  58                CLI             ENABLE INTERRUPTS
0711  85 CC             STA GANG        SAVE OUTPUT BIT
0713  CA                DEX             ALL CYCLES SENT ?
0714  D0 E7             BNE ZON1        NO, SEND MORE

0716  68                PLA             RESTORE CHARACTER
0717  C6 CF             DEC TRIB        ONE LESS PHASE TO GO
0719  F0 05             BEQ SETZ        AND THIS IS PHASE 3
071B  30 07             BMI ROUT        ALL PHASES DONE

071D  4A                LSR             GET BIT ...
071E  90 DA             BCC ZON                 ... IF IT IS '1'...
0720  A0 00       SETZ   LDY #$ 00       ... CHANGE TO 2400 HZ
0722  F0 D6             BEQ ZON         FORCED BRANCH
0724  C6 CE       ROUT   DEC COUNT       ONE LESS BIT TO GO
0726  D0 CE             BNE TRY
0728  60                RTS             ALL DONE

0729  17                .BYTE           LOP, LENGTH OPERATOR
072A  02          TAB   .BYTE           VALUE FOR NPUL
072B  C3                .BYTE
072C  03                .BYTE           VALUE FOR TIMG+1
072D  7E                .BYTE
072E  EA                NOP             RALOAD-BYTE

            ALTERNATIVE VALUES FOR NPUL AND TIMG+1:

072A      ×3: $04   ×2: $06    ×1: $0C   SEE ADDITIONAL REMARKS
072C          $06       $09        $12
```

* * *

VERSALOAD - SUMMARY

Program renders 6 basic reading functions (see block-diagram) which are increased by the possibility to define a buffer+1 which may not be surpassed. Begin of buffer to SAL/SAH, end+1 to EAL/EAH.

Records to be read may have any of these formats:

a) single-byte ID to be compared with 17F9,
b) standard name of 6 bytes to be compared to table in 1780-85 or
c) name as in b) plus additional parameters which are read into 1786 ...

First cell to be filled on read is addressed by VEB+1,2 for LINK.., is old start address for LOADOE or LOADBU in all other cases it is the cell: begin of buffer.

VERSALOAD

Wie schon dargestellt, bietet VERSALOAD 6 grundsätzliche Dienstleistungen (Blockdiagramm auf der nächsten Seite). Welche im einzelnen ausgeführt wird, hängt vom gewählten Startpunkt ab oder von dem im Akkumulator mitgebrachten Parameter, wenn man JSR ENTRY aufruft.

Diese sechs Dienstleistungen werden durch die Buffer-Möglichkeit ergänzt: SAL/SAH in 17F5/F6 legen dann eine Anfangsadresse für das Speichern fest, EAL/EAH in 17F7/F8 das Buffer-Ende+1 (erste zu schützende Zelle). Wenn ein Ladevorgang wegen Erreichens des Schutzbereiches unvollständig abgebrochen werden mußte, so erfolgt Fehleranzeige durch den KIM-Monitor mit 'FFFF1C', zugleich wird die LFLAG auf 'FF' gesetzt. Lesefehler mit abweichender Checksum führen zur gleichen Monitor-Anzeige, die LFLAG wird aber auf 'FE' gesetzt.

Die Zahl der Dienstleistungen wird eigentlich fast verdreifacht, weil die Identitätsangabe für die zu lesenden Bandsätze verschieden sein darf:

a) Identität 1 Byte, Vergleich mit Zelle 17F9,
b) Standard-Name 6 Bytes, Vergleich mit Tabelle in 1780-85,
c) Standard-Name wie in b), zusätzlich externe Parameter, die beim Lesen in den Zellen 1786 ff. abgelegt werden.

Die Dienstleistungen im einzelnen:

LADOE, LOAD Open End. Ein Programm wird an seinem alten Speicherplatz geladen, es darf beliebig lang sein, weil ein Puffer nicht zu beachten ist.

LOADBU, LOAD only up to end of BUffer. Einspeicherung ab altem Speicherplatz aber nicht über Puffer-Ende hinaus.

DATALB, DATA Load to Buffer. Anfang und höchstmöglicher Speicherplatz sind festgelegt.

RELOE, RELocate Open End. Rechnet ein Programm sofort nach dem Laden ab festgelegter Speicheradresse auf den neuen Adressenbereich um, und zwar mit Hilfe des in Heft 1 von 65xx MICRO MAG beschriebenen RALOAD, das natürlich im Speicher resident sein muß. Ein Pufferende muß nicht berücksichtigt werden.

RELBUF, RELocate with BUffer. Wie vor, die Endadresse darf nicht überschritten werden.

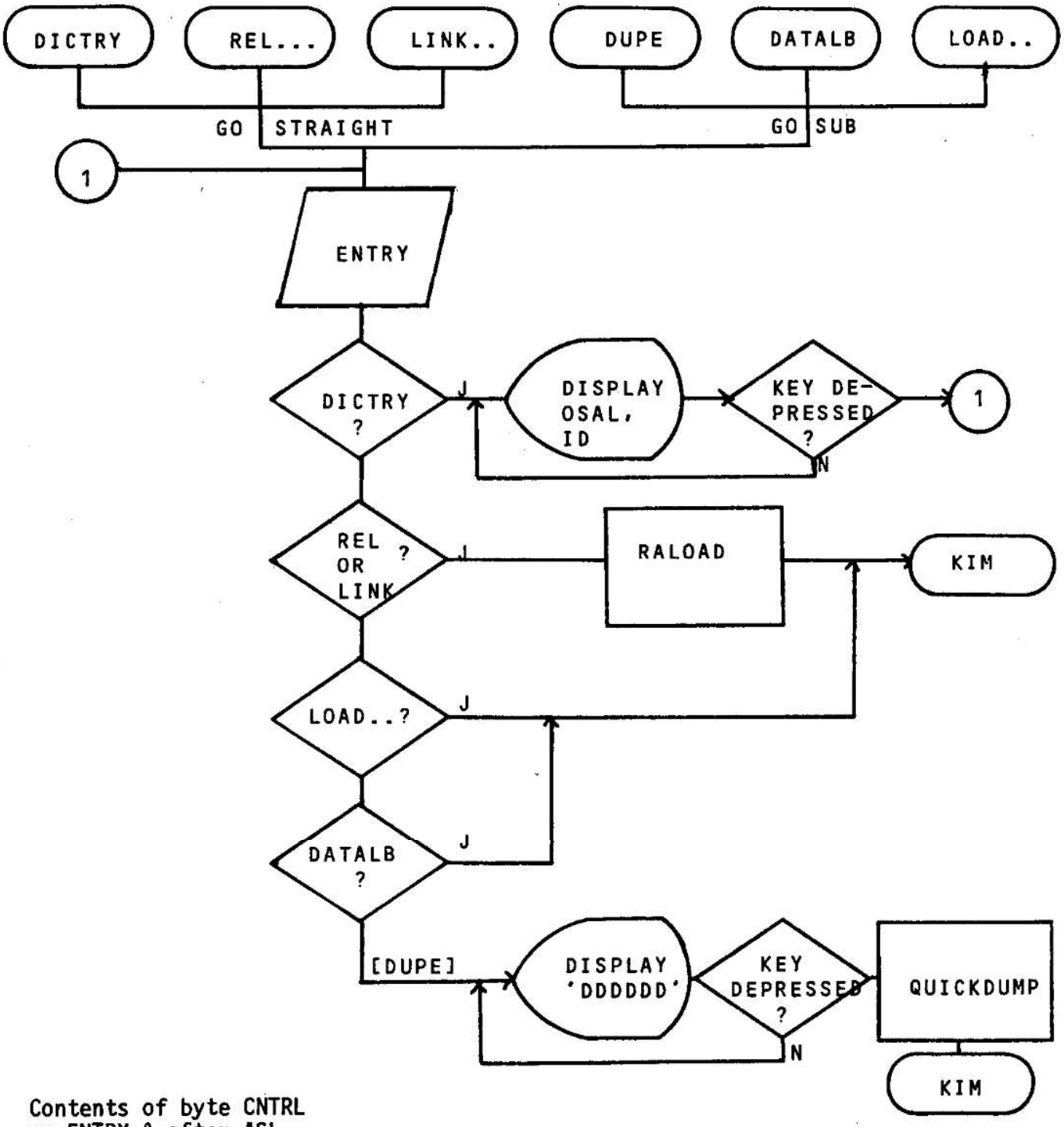
LINKOE, LINK Open End. Verwirklicht Laden und Verschweißen. Laden ab der in VEB+1,2 ausgewiesenen ersten freien Speicherstelle und Umrechnen mit RALOAD auf den neuen Adressenbereich.

LINKBU, LINK with BUffer. Entspr. LINKOE bzw. RELBUF.

DUPE entspricht den Funktionen von Jim Butterfields SUPER DUPE. Es gestattet, Programme von einem Band auf ein anderes zu kopieren. Zur Zwischenspeicherung dient ein Puffer, dessen Grenzen wie vor festzulegen sind. Ein zu kopierendes Programm muß mit seiner wirklichen Identität bzw. 00 oder mit seinem vollen Namen gesucht werden. Betätigung einer Taste löst Quickdump aus.

DICTRY lädt nichts, sondern bringt nur die alte Startadresse und ein weiteres Byte zur KIM-Anzeige. Dieses ist entweder die ID von einem BYTE oder das erste Zeichen des Namens (von 6 Zeichen). Nach Drücken einer Taste entspr. Anzeige für den nächsten Bandsatz.

PROGRAMMABLAUFPLAN VERSALOAD



Contents of byte CNTRL
on ENTRY & after ASL

	LINK	RELOCATE	BUFFER		LOADBU	DATALB	DUPE	DICTRY
LINK	RELOCATE	BUFFER		LOADBU	DATALB	DUPE	DICTRY	
BIT C	N	V	5	4	3	2	1	0

```
                        VERSALOAD
0800  A9 22     DUPE    LDA #$ 22       PARAMETER FOR CNTRL
0802  D0 1F             BNE GOSUB       BRANCH ALWAYS
0804  A9 01     DICTRY  LDA #$ 01       THESE ARE THE DIFFERENT
0806  D0 0E             BNE GOSTR       ENTRIES FOR FUNCTIONS
0808  A9 C0     LINKOE  LDA #$ C0       TO BE RENDERED ONCE
080A  D0 0A             BNE GOSTR
080C  A9 E0     LINKBU  LDA #$ E0
080E  D0 06             BNE GOSTR
0810  A9 40     RELOE   LDA #$ 40
0812  D0 02             BNE GOSTR
0814  A9 62     RELBUF  LDA #$ 62
0816  4C 29 08  GOSTR   JMP ENTRY       THIS ONE COULD HAVE BEEN
0819  A9 00     LOADOE  LDA #$ 00                DELETED.
081B  F0 06             BEQ GOSUB
081D  A9 28     LOADBU  LDA #$ 28
081F  D0 02             BNE GOSUB
0821  A9 24     DATALB  LDA #$ 24
0823  20 29 08  GOSUB   JSR ENTRY
0826  4C 4F 1C          JMP KIM         RETURN TO MONITOR

                        MAIN ROUTINE
0829  ØA       ENTRY   ASL             GIVES A BETTER CNTRL, CUTS OFF
082A  85 D8             STA CNTRL       LINK BIT TO CARRY
082C  90 0C             BCC NOLINK
082E  AD ED 17  LINK    LDA VEB+1       INSERT FIRST FREE PLACE TO
0831  8D F5 17          STA SAL         SAL/SAH
0834  AD EE 17          LDA VEB+2
0837  8D F6 17          STA SAH

083A  24 D8     NOLINK  BIT CNTRL       TEST
083C  50 0A             BVC XID         SKIP IF OPEN END LOADING
083E  AD F7 17          LDA EAL         INITIALIZE END OF BUFFER
0841  85 C9             STA EALB
0843  AD F8 17          LDA EAH
0846  85 CA             STA EAHB

0848  AD F9 17  XID     LDA ID          SAVE DESIRED ID
084B  85 D9             STA IDTEMP

084D  A9 00     SUPERL  LDA #$ 00
084F  8D F9 17          STA ID          KIM'S LOADT IS DUPED TO STAY IN
0852  85 CB             STA LFLG        FLAG = 00             SUBROUTINE
0854  85 D5             STA SWI2        COUNT = ØØ

0856  A9 60             LDA #$ 60       'RTS' OPCODE FOR VEB
0858  8D EC 17          STA VEB
085B  20 8C 18          JSR 188C        ENTER KIM'S LOADT
085E  85 D4             STA SWI1        LENGTH OF NAME & PARMS OR JUST ID
0860  20 24 1A  NEXCHT  JSR RDCHT       READ NEXT CHARACTER
0863  AD EA 17          LDA SAVX+1      GET FULL 8 BIT BYTE
0866  A6 D8             LDX CNTRL       TEST FOR DICTRY
0868  E0 02             CPX #$ 02
086A  D0 03             BNE             SKIP IF NOT
066C  4C 7E 09          JMP SHOWDI      DISPLAY AND WAIT FOR KEY

086F  A6 D4             LDX SWI1        TEST KIND OF ID/NAME & PARMS
0871  D0 0E             BNE COMTB       SKIP IF NO SINGLE BYTE ID
```

```
0873  C5 D9          CMP IDTEMP     IS IT THE SINGLE BYTE ID?
0875  D0 04          BNE SKPD       SKIP IF NOT
0877  85 D9          STA IDTEMP     A SERVICE TO DUPE IF 17F9 WAS ØØ
0879  F0 2B          BEQ NUMBYT     BRANCH ALWAYS

087B  E4 D9          CPX IDTEMP     X CONTAINS ØØ
087D  F0 27          BEQ NUMBYT     TREAT AS A MATCH
087F  D0 CC          BNE SUPERL     TEST NEXT RECORD

0881  A6 D5    COMPTB LDX SWI2      GET CURRENT OFFSET
0883  DD 80 17        CMP 1780,X     COMPARE TO NAME IN TABLE
0886  D0 CE          BNE SUPERL     GET NEXT RECORD ON NO MATCH
0888  E6 D5          INC SWI2       OFFSET+1
088A  A9 05          LDA #$ 05      TO COMPARE FOR END OF NAME
088C  C5 D5          CMP SWI2
088E  B0 D0          BCS NEXCHT     GET TOTAL OF 6 BYTES
0890  A5 D4    EOHEAD LDA SWI1      STORE ADDITIONAL PARMS FROM
0892  C5 D5          CMP SWI2       TAPE HEADER IF SWI1 STANDS FOR
0894  90 10          BCC NUMBYT     MORE THAN 6 BYTES  * ALL DONE
0896  20 24 1A       JSR RDCHT      GET NEXT PARM
0899  AD EA 17       LDA SAVX+1     GET 8-BIT-BYTE
089C  A6 D5          LDX SWI2       GET CURRENT OFFSET
089E  9D 80 17       STA 1780,X     STORE TO MEMORY
08A1  E6 D5          INC SWI2       FOR NEXT
08A3  4C 90 08       JMP EOHEAD     AND TEST FOR END OF PARMLIST

08A6  AD ED 17 NUMBYT LDA VEB+1     SAVE OLD START ADDRESS
08A9  85 D6          STA OSAL
08AB  AD EE 17       LDA VEB+2
08AE  85 D7          STA OSAH
08B0  A9 8D          LDA #$ 8D      'STA'-OPCODE FOR VEB
08B2  8D EC 17       STA VEB        REPLACES 'RTS'

08B7  20 24 1A       JSR RDCHT      NEXT CHAR
08B8  A5 D8          LDA CNTRL      TEST
08BA  F0 04          BEQ NADJ       IT'S LOAD TO OLD START ADDRESS
08BC  C9 10          CMP #$ 10
08BE  D0 06          BNE XCHNG      IT'S LOAD TO NEW START ADDRESS
08C0  20 3E 19       JSR INTVEB+12  INITIALIZE VEB AND CHKL CHKH
08C3  4C C9 08       JMP COMPU      SKIP

08C6  20 32 19 XCHNG  JSR INTVEB    INITIALIZE FULL WITH NEW START
08C9  18       COMPU  CLC                                  ADDRESS
08CA  AD EA 17       LDA SAVX+1     NBL WITH 8 BITS
08CD  6D ED 17       ADC VEB+1      TO COMPUTE ENDADDRESS OF PGM
08D0  85 E2          STA EOPL       AND SAVE
08D2  08             PHP            SAVE CARRY STATUS
08D3  20 24 1A       JSR RDCHT      NEXT CHAR IS NBH
08D6  28             PLP            GET BACK CARRY STATUS
08D7  AD EA 17       LDA SAVX+1     SAME FOR # HI
08DA  6D EE 17       ADC VEB+2
08DD  85 E3          STA EOPH

08DF  20 24 1A PATCH1 JSR RDCHT     JOHN OLIVER'S NUCLEUS
08E2  AD EA 17       LDA SAVX+1     NEXT 8 BITS FROM TAPE
08E5  20 4C 19       JSR CHKT       ADD TO CHECKSUM
08E8  20 EC 17       JSR VEB        STORE IT

08EB  20 EA 19       JSR INCVEB     INCREMENT VEB ADDRESS FOR STORE
```

```
08EE  AD ED 17        LDA VEB+1      END ADDRESS?
08F1  24 D8           BIT CNTRL
08F3  50 06           BVC PATCH3     DONT'T CARE FOR BUFFER
08F5  C5 C9           CMP EALB       BUFFER END?
08F7  D0 02           BNE PATCH3     NO
08F9  F0 04           BEQ PATCH4     MAYBE?
08FB  C5 E2   PATCH3  CMP EOPL       RECORD END?
08FD  D0 E0           BNE PATCH1     NO, GET MORE BYTES

08FF  AD EE 17 PATCH4 LDA VEB+2      SAME FOR HI
0902  24 D8           BIT CNTRL
0904  50 11           BVC PATCH5     DON'T CARE FOR BUFFER
0906  C5 CA           CMP EAHB       BUFFER END?
0908  D0 0D           BNE PATCH5     NO
090A  C5 E3           CMP EOPH       ALSO RECORD END?
090C  D0 64           BNE ERROR2     NO, ERROR EXIT
090E  AD ED 17        LDA VEB+1      LOW ORDER BYTE ALSO OK?
0911  C5 E2           CMP EOPL
0913  D0 5D           BNE ERROR2
0915  F0 04           BEQ PATCH6

0917  C5 E3   PATCH5  CMP EOPH       RECORD END?
0919  D0 C4           BNE PATCH1     NO, CONTINUE

091B  20 24 1A PATCH6 JSR RDCHT      GET ENDING CHARACTER
091E  C9 2F           CMP #$ 2F      '/' ?
0920  D0 4E           BNE ERROR

0922  20 F3 19        JSR RDBYT      GET CHECKSUM LO
0925  CD E7 17        CMP CHKL       CHECKSUM OK ?
0928  D0 46           BNE ERROR
092A  20 F3 19        JSR RDBYT      GET CHECKSUM HI
092D  CD E8 17        CMP CHKH
0930  D0 3E           BNE ERROR

0932  A5 D8           LDA CNTRL      TEST FOR LINK AND RELOCATE
0934  10 0D           BPL BUFA       BRANCH IF NOT
0936  A5 D6           LDA OSAL       SERVICE TO RALOAD PROGRAM
0938  8D F7 17        STA EAL
093B  A5 D7           LDA OSAH
093D  8D F8 17        STA EAH
0940  4C 27 02        JMP RALOAD     TAKE CARE THAT PROGRAM
                                     ACTUALLY RESIDES HERE.

0943  29 0C   BUFA    AND #$ 0C
0945  F0 36           BEQ EXIT       IF LOADOE OR LOADBU
0947  AD ED 17        LDA VEB+1      INSERT ENDADDRESS+1 FOR
094A  8D F7 17        STA EAL        DUPE AND DATALB
094D  AD EE 17        LDA VEB+2
0950  8D F8 17        STA EAH
0953  A5 D8           LDA CNTRL      TEST
0955  C9 48           CMP #$ 48      IS IT DATALB ?
0957  F0 24           BEQ EXIT       ALL DONE FOR THIS ONE

0959  A5 D9           LDA IDTEMP     GET PARAMETERS FOR DUPE
095B  8D F9 17        STA ID
095E  A9 DD           LDA #$ DD      TO SHOW 'DD DD DD'
0960  85 F9           STA INH
```

```
0962   85 FA          STA POINTL
0964   85 FB          STA POINTH
0966   20 1F 1F SHOW1 JSR SCANDS     DISPLAY 'DD DD DD' AND WAIT
0969   F0 FB          BEQ SHOW1      NO KEY DEPRESSED ?
096B   A2 01          LDX #$ 01      PARAMETER ≠ 00 FOR SWI2
096D   4C 11 06       JMP ENTRY1+2   TAKE CARE THAT QUICKDUMP'S
                                      ENTRY1 ACTUALLY RESIDES HERE !

0970   C6 CB    ERROR  DEC LFLG
0972   C6 CB    ERROR2 DEC LFLG
0974   24 D8          BIT CNTRL      TEST KIND OF ENTRY INTO PGM
0976   30 02          BMI SKPD       WAS STRAIGHT ENTRY, SKIP
0978   68 68    SKPD   PLA PLA       ADJUST SP
097A   4C 29 19       JMP LOADT9     KIM SHOWS 'FF FF 1C'

097D   60       EXIT   RTS

097E   85 F9    SHOWDI STA INH        DICTRY SHALL SHOW ID
0980   AD ED 17       LDA VEB+1      AND OLD START ADDRESS
0983   85 FA          STA POINTL
0985   AD EE 17       LDA VEB+2
0988   85 FB          STA POINTH
098A   20 1F 1F SHOW2 JSR SCANDS
098D   F0 FB          BEQ SHOW 2     NO KEY DEPRESSED ?
098F   4C 4D 08       JMP SUPERL     READ HEADER OF NEXT RECORD IF YE
0992   EA       END    NOP            RALOAD BYTE FOR RELOCATION
```

* * * * *

Wo soll man Arbeitsspeicher bereitstellen ?

Bei der Abfassung von Programmen tritt immer wieder ein Gewissenskonflikt auf: Soll man den Arbeitsspeicher

a) in die Zero Page legen, wie hier z.B. bei QUICKDUMP und VERSALOAD, oder aber
b) an das Ende des Programmes selbst, in die absolute Adressierung?

a) hat den Nachteil, daß die Zero Page sich sehr schnell füllt und daß ein Programm die Pointer eines anderen zerstören kann. Diese Gefahr besteht vor allem beim Einsatz von Programmen aus verschiedener Quelle.

b) ist eine eindeutige Methode, die Arbeitsspeicher zu entflechten. Es treten keine Schwierigkeiten bei der Programmverschiebung auf, wenn RALOAD und seine Formate verwendet werden. Nachteil: Solche Programme können nicht in Festwertspeicher übernommen werden.

Je nach Interessenlage wird man den Ausweg aus diesem Konflik in der Reservierung der Zero Page für Pointer und schnelle Verarbeitung und in der Bereitstellung einer anderen Page für Arbeitsspeicher suchen.