MTX User-Club Deutschland

Info 44
04.10.1991

Zweck: Zusammentragen und Austausch von Tips & Tricks u.s.w., Hilfestellung bei allen möglichen Problemen, Aufbau einer Programmbibliothek und Basteln von Hardware-Erweiterungen.

Programme (nur **Selbstgeschriebenes**): Tausch von kurzen und einfachen Routinen. Gute Programme (mit Dokumentation) können über den Club an alle Mitglieder verkauft werden. Wer solche Programme an uns schickt erhält ggf. Verbesserungshinweise und eine Besprechung im Info.

Mitglied kann jeder werden! Keine Beitragsgebühr! Anmeldung kostet DM 1.-.

Verpflichtungen: Einsendung unseres Anmeldeformulars.

Bitte: Einsendung von Tips & Tricks, Fragen, Antworten, kurzen Routinen, Programmen, Beiträgen zum Info, Hinweisen auf preiswerte Hard- und Software, und was noch so zusammenkommt und andere interessieren könnte.

Club-Info, unser Blatt, verschicken wir ca. 8-wöchentlich. Inhalt ist alles was uns über den MTX/FDX (ohne Copyright) in die Hände fällt. Es kostet nicht über DM 12.- je Exemplar. Jeder kann dazu Beiträge liefern und hier gratis Kleinanzeigen veröffentlichen.

Kosten: Wir berechnen ausschließlich Selbstkosten und verschicken **nichts**, wenn Ihr persönliches Guthaben nicht reicht! (s.u.)
Schüler, Studenten, Auszubildende, Grundwehrdienstleistende, Rentner und Arbeitslose erhalten einen Nachlaß von 40% auf die zukünftigen Infos nach Einsendung einer entsprechenden Bescheinigung für deren Gültigkeitzeitraum.

Geld/Konto: Für jedes Mitglied führt Herbert zur Nedden ein Konto, von dem die jeweils enstehenden Kosten abgehen. Der Kontostand wird bei **jeder** Sendung mitgeteilt (**er steht über der Anschrift**) und kann selbstverständlich jederzeit erfragt werden! Wir verschicken nur gegen Vorkasse!

Einzahlungen bitte auf's Club-Konto: (oder V-Scheck)
(**Absender!** incl Name **und** Anschrift bitte nicht vergessen!)
Postgiroamt Hamburg, BLZ 200 100 20,
Herbert zur Nedden, Sonderkonto C, Nr. 3480 00-200

Kontaktadressen:

Herbert zur Nedden
Alte Landstraße 21
2071 Siek
(04107) 99 00

Hans Gras
Statenhoek 49
NL 1506 VM Zaandam
(0031-75) 17 49 91

Telefon-Sprechzeiten
Herbert zur Nedden: Do 18 - 21 Uhr, Sa 9 - 14 Uhr
(Etwas klingeln lassen oder nochmal versuchen!)

I n h a l t s v e r z e i c h n i s

Preis für dieses Info: DM 8.-

Angebotsliste (Herbert zur Nedden, 2071)

Dieser Liste liegt diesem Info mal wieder bei - allerdings ohne die Aufstellung der nicht im CLUB zusammengestellten PDs (SIG/M, ...) , da sich der Teil nicht verändert hat.
Wenn Du etwas in dieser Liste anbietest, laß mich bitte wissen, ob Dein Angebot noch gültig ist! Sollte ich von Dir bis zum 1. Advent nichts gehört haben, nehme ich an, daß Dein Angebot gestrichen werden kann - und streiche es!
Sollte es sich bei Deinem Angebot um Software handeln, die Du nicht mehr vertreiben willst, überleg Dir doch bitte, ob Du bereit bist, das Teil zu Public-Domain zu machen ... und falls dem so ist, schick es mir bitte mit einem entsprechenden Hinweis.

Kontostand (Herbert zur Nedden, 2071)

Eine rote Markierung auf dem Umschlag bedeutet, daß er zu niedrig ist.

I n h a l t s v e r z e i c h n i s

Preis für dieses Info: DM 8.-

Angebotsliste (Herbert zur Nedden, 2071)
Dieser Liste liegt diesem Info mal wieder bei - allerdings ohne die Aufstellung der nicht im CLUB zusammengestellten PDs (SIG/M, ...) , da sich der Teil nicht verändert hat.
Wenn Du etwas in dieser Liste anbietest, laß mich bitte wissen, ob Dein Angebot noch gültig ist! Sollte ich von Dir bis zum 1. Advent nichts gehört haben, nehme ich an, daß Dein Angebot gestrichen werden kann - und streiche es!
Sollte es sich bei Deinem Angebot um Software handeln, die Du nicht mehr vertreiben willst, überleg Dir doch bitte, ob Du bereit bist, das Teil zu Public-Domain zu machen ... und falls dem so ist, schick es mir bitte mit einem entsprechenden Hinweis.

Kontostand (Herbert zur Nedden, 2071)
Eine rote Markierung auf dem Umschlag bedeutet, daß er zu niedrig ist.

Moin, moin!

Das ein Haus viel Zeit kostet, kann sich jeder denken, der selbst eines hat! Ich weiß es nun endlich auch! Daher mußte ich meine Aktivitäten für den Club immer wieder hinten an stellen - so leid es mir tut - zumal mich mein Beruf in den letzten Monaten sehr in Anspruch nahm. DiJey Version 3.00 (auf KLICK.018) und dieses Info verdankt Ihr eigentlich vor allem meinem derzeitigen Urlaub von zwei Wochen, in dem ich mich mal wieder die eine oder andere Stunde vor meine geliebte schwarze Kiste setzen konnte. Allerdings ist nun das gröbste vorbei, so daß ich wieder mehr Zeit habe.
Aus eben diesem Grund (Zeitmangel) liegt RAM 6.2 bei mir noch in der Schublade; es wird aber auf jeden Fall kommen! Was im Endeffekt dann wie realisiert sein wird, ist noch offen - auf jeden Fall werden die einen oder anderen Kleinigkeiten drin sein, die uns unser eh schon recht angenehmes Leben noch angenehmer machen können.

In diesem Info findest Du einen Artikel zum Anschluß einer IBM-Tastatur an die serielle Schnittstelle - ja, schon wieder! Ich habe mir vom Heise-Verlag (c't) ein paar Informationen zum Tastaturinterface von PCs schicken lassen, und so mein Interface DEUTLICH verbessern und damit die Anpassung an andere Tastaturen als meine Escom ERHEBLICH vereinfachen können! Insbesondere das Thema Baudrate hat sich erledigt. Bei mir liefen zwei andere Tastaturen mit dem neuen Interface auf Anhieb! Der Artikel beginnt übrigens mehr oder weniger bei Adam und Eva (sozusagen 'allgemeines Vorgeplänkel') und ist an einigen Stellen ausführlicher als der aus dem vorherigen Info (sprich auch für Nicht-MTX-er), da ich den Artikel auch an c't schicken werde - vielleicht wird er ja dort veröffentlicht. (Zwei Versionen dieses Atrikels - 1x für's Info und 1x für c't - das wollte ich mir nun doch verkneifen.)

Ich habe selbstverständlich vor, auch im kommenden Jahr ein Clubtreffen zu veranstalten. Bei diesem werde ich mir allerdings zusammen mit den Wirtsleuten eine Mimik ausdenken, die sicherstellt, daß jeder seine Zeche bezahlt! Die beiden letzten Male waren Zechpreller dabei, was ich für eine ausgemachte Gemeinheit halte! Es ist schon ärgerlich, daß dem so ist, aber anscheinend geht es nicht anders, zumal beide Male der Aufruf im Info ohne Erfolg blieb, sich zu überlegen, ob man selbst derjenige ist, der vergaß etwas zu löhnen und das Geld nachzureichen. Die beiden letzten Male haben nämlich Holger Göbel und ich das fehlende Geld ausgelegt ... äh berappt.

Anscheinend ist Urlaubszeit; ich habe schon seit langem kaum noch Beiträge zum Info oder zur PD-Sammlung erhalten! Schlummert nicht noch etwas bei Dir in der Schublade bzw. auf Diskette, was Du auch den anderen zur Verfügung stellen magst? Ich fände das schon toll, wenn!

Euer Herbert zur Nedden

Fragen

F: (Andreas Ficher, CH-4303; Alois Binder, 6728)
Hat jemand BTX-Software für unsere Kiste?
F: (Andreas Ficher, CH-4303; Alois Binder, 6728)
Wer hat Erfahrungen mit dem Betrieb eines Meteofax-Gerätes der Firma C-Data aus Pfaffenhofem am MTX. (**Anm.d.HzN:** Was ist das für ein Gerät?)
F: (Hans Gras, NL-1506)
Wer kann mir ein SDX-Floppy-Controller-Gehäuse verkaufen?

S o f t w a r e: DiJey I Version 3.00

DiJey I V 3.00 (Herbert zur Nedden, 2071)

Die Version 3.00 geht auf einige Wünsche ein, die von Euch (und von mir) gekommen sind. Was noch nicht implementiert wurde, sind die Unterstützung von Libraries, da das doch wesentlich aufwendiger als zunächst angenommen ist. Das Thema Uncrunchen usw. habe ich auf dem Zettel - dazu habe ich erst mal UNCRunch Version 2.8 disassembliert. Den LZH-Teil hatte Herbert Oppmann ja freundlicherweise schon fertig (siehe CLUB.057). Die wesentlichen Neuerungen im Überblick:

B Dieser Buchstabe bedeutete unter dem alten DiJey: Gehe eine Datei nach links! Was für eine Verschwendung, da eh jeder mit den Pfeiltasten arbeitet - das war auch nur drin, da NSWEEP das auch so machte. Nun ist B ein sogenanntes oder auch Vor-Zeichen, mit dem Du Dateien abhängig davon bearbeiten kannst, ob bestimmte Bits im Dateinamen (speich Attribute) gesetzt sind oder nicht. D.h. B funktioniert ähnlich wie das W, nur halt auf die Bits an Stelle der Zeichen im Dateinamen. Diese Option fehlte nun wirklich - Eure Anfragen dazu klingen Olaf und mir noch in den Ohren! Immerhin haben die Dateiattribute, insbesondere das Archiv-Bit, seit einiger Zeit eine doch recht wichtige Bedeutung erhalten.

K Dieses Kommando dient zum Kopieren von Dateien. Wie bitte?? Das war doch schon drin, oder? Klar doch: Kommando C. Aber das neue Kopier-Kommando K verrichtet seine Arbeit abhängig davon, ob die Ziel-Dateien (das sind die, die beim Kopieren überschrieben bzw. neu angelegt werden würden) schon vorhanden sind, oder nicht.
Wozu? Zum einen kommt es vor, daß ich von einer Datei eine Sicherungskopie unter einem neuen Namen anlegen will. Bislang mußte ich erst nachschauen, welche Dateien (Sicherungskopien) es auf der Ziel-Diskette schon gibt; nun kann ich das DiJey I überlassen. Andersherum kann ich jetzt bequem nur die Dateien kopieren, die es auf der Zieldiskette schon gibt (ich glaube, der Fachmann spricht hier von Backups).

SHIFT-HOME Mit dem Kommando X kann man bekanntlich (?) eine DiJey dazu veranlassen, eine bestimmte Kommandozeile in Abhängigkeit von der Datei-Extension aufzubauen, die man dann noch etwas editieren kann. Was Dir DiJey anbietet kannst Du natürlich installieren. Nach dem Verlassen von KLICK wird sie dann automatisch ausgeführt. Häufig will man die angebotene Kommandozeile aber gar nicht mehr verändern, sondern sofort unverändert ausführen lassen. SHIFT-HOME tut genau das: Baut die Zeile auf und verläßt umgehend DiJey und KLICK.

Cursorpositionierung bei Verwendung der Pfeile hoch und runter wurde etwas überarbeitet. Das wirkt sich am oberen bzw. unteren Ende einer Spalte aus.

CTRL-R bzw. **CTRL-C** kann man auf die erste bzw. letzte Datei positionieren.

CTRL-E, -X, -S und **-D** wirken jetzt wie die Pfeilstaten. Dies ist bei Verwendung von IBM-Tastaturen recht angenehm, da deren zusätzlicher Cursor-Block eben diese Codes liefert.

View akzeptiert nun statt RET auch die Leertaste am Dateiende.

Anzeige der aktuellen Kommando-Buchstaben oben rechts dem Bildschirm. Gelegentlich wußte ich nicht mehr, worauf sich die eine oder andere Frage von DiJey bezog ...

Anpassung an und Ausnutzung von RAM 6.1. Daher läuft DiJey I ab Version 3.00 nur noch unter RAM 6.1 oder besser.

(Übrigens: DiJey I V 3.00 ist von HzN, also bitte Fehlerinfos an mich!)

S o f t w a r e: MLOAD / R A M 6.1: NamedDirectories

MLOAD - oder wie patche ich mal eben (Herbert zur Nedden, 2071)

Gelegentlich möchte man ja 'mal eben' eine kleine Assemblerroutine in ein Programm einpatchen. Also kurz mit einem normalen Editor eingeben und assemblieren, um das Listing zu haben. Nun MONI (oder einen anderen (De)Bugger) nehmen, und die Routine Befehl für Befehl oder gar Byte für Byte eingeben. Dann Repeat
  Prüfen
  Ärgern
  Korrigieren, sprich fast alles nochmal eingeben
Until (2x ohne Fehler) or (Frust)

Nun lassen wir das Programm laufen und hoffen. Oft wiederholt sich das ganze nochmal - und ist so mühselig.

Warum hat eigentlich noch niemand ein Programm geschrieben, mit dem ich meinen Assembler-Source automatischer in ein Programm einpatchen kann - von mir auch, indem das Listing verwendet wird?

Weil es das schon gibt! Das Wunderding heißt MLOAD (CLUB.814). Was ist zu tun?

1. Man wandle seine Assemblerroutine um - natürlich für die gewünschte Ziel-Adresse (ORG-Befehl, ggf. auch mehrere) - und zwar in eine HEX-Datei. Mit dem SLR-Assembler geht das durch Angabe der Option /H; beim Linken (SLRLN oder L80) tut es auch diese Option.
2. Will ich OTTO.COM mit KARL.HEX patchen um KARLOTTO.COM zu bekommen, so gebe ich nur noch ein: MLOAD KARLOTTO.COM=OTTO.COM,KARL.HEX.
   Schon ist der obige Repeat-Until erledigt!

Eine Anwendung hierfür hat Hans Gras mir kürzlich geschickt:

RAM 6.1 und mehr Named Directories (Hans Gras, NL-1506)

```
; Intended to lower the Memotech Top of Available Memory (TOAM) to create more
; space fot the NDR Buffer. Be careful: adjust to adjust this data to availab-
; le space, so you can't try this in a 60k environment, because there is very
; little (< 100 Bytes) space left.

   MACLIB     Z3BASE.LIB

   ORG        07D8h
   DW         Z3NDIR
   ORG        0897h
   DW         Z3NDIR
   ORG        089ah
   DW         Z3NDIR+1
   ORG        0d3bh
   DW         Z3NDIR

; Use SLR180 NDR6/H and MLOAD RAM.COM=RAM61.COM,NDR6.HEX
```

Anm.d.HzN: Durch Änderung des Wertes für Z3NDIR in Z3BASE.LIB oder ersetzen der MACLIB-Zeile durch Z3NDIR equ ???? kannst Du den NDR-Puffer vergrößern - aber bitte mit Vorsicht! Zu niedrig darf der Wert bei großer Systemgröße nicht sein. Ist nämlich TOAM < FREE, dann darfst Du Dich über nichts mehr wundern ...

S o f t w a r e / H a r d w a r e: Bildschirmsequenzen

Bildschirmsequenzen (Herbert zur Nedden, 2071)

Gerade in Zusammenhang mit verschiedenen Monitor-Typen Monochrom (wie der übliche - in Grün oder Bernstein), Farbe oder Schwarz-Weiß mit Graustufen ist das Thema, welches Attribut hat welche Wirkung und insbesondere die Frage 'Was ist wann noch zu lesen?' nicht zu vernachlässigen.

RAM 6.x hilft uns bei der Problematik etwas: Einige Kombinationen der Vorder- mit der Hintergrundfarbe ist auf Monochrom-Monitoren problemlos lesbar, auf Farbschirmen sieht man aber nichts! Das liegt daran, daß für die Vorder- und Hintergrundfarbe je drei Bit vorgesehen wurden, von denen im Monochrom-Modus nur je zwei ausgenutzt werden. Auf meinem Schirm kann ich daher nicht sicher erkennen, was auf einem Farbschirm noch lesbar ist, oder nicht. Der Trick, den RAM 6.x daher anwendet ist der, daß bei identischer Vorder- und Hintergrundfarbe ein Bit der Vordergrundfarbe umgeknipst wird, welches bei Monochrommonitoren keine Auswirkung hat.

Die Farben setzen sich aus den drei Grundfarben Rot, Grün und Blau zusammen. Soweit ist das sicherlich noch nicht besonders überraschend - schließlich arbeiten auch Farbfernseher mit eben nur diesen drei Grundfarben. Der Farbmonitor-Ausgang der FDX heißt deshalb RGB-Ausgang. Hast Du es erraten? Klar RGB = RotGrünBlau.

Im Farbmodus haben die drei Grundfarben logischerweise eben die Bedeutung, die ihre Bezeichnung angibt. Durch Mischung dieser drei Farben kommen insgesamt acht mögliche Kombinationen heraus. Welche Farbe bzw. Farbkombination ausgegeben wird, legst Du mit den sogenannten Attributen fest, die Du über bestimmte Steuersequenzen aktivieren kannst. Das Attribut wird in einem Byte gespeichert, welches folgende Aufteilung hat:

```
Bit 0   Rot   \
Bit 1   Grün   > der Vordergrundfarbe
Bit 2   Blau  /
Bit 3   Rot   \
Bit 4   Grün   > der Hintergrundfarbe
Bit 5   Blau  /
Bit 6   Blinken
Bit 7   Wahl des Eproms: Zeichen(0) oder Grafik(1)
```

Durch setzen der entsprechenden Bits kannst Du damit problemlos und unabhängig voneinander die Vorder- und Hintergrundfarben aus den drei Grundfarben zusammenmischen.

Für Monochrom-Monitore machen Farben recht wenig Sinn! Statt Farben festzulegen, haben die o.g. Attribut-Bits im Monochrom-Modus eine andere Bedeutung:

```
Bit 0   Unterstreichen
Bit 1   keine Wirkung
Bit 2   Helle Zeichen
Bit 3   keine Wirkung
Bit 4   Inverse Zeichen
Bit 5   Hintergrund heller
Bit 6   Blinken
Bit 7   Wahl des Eproms: Zeichen(0) oder Grafik(1)
```

D.h. die Vordergrundfarbe entspricht den Attributen Unterstreichen und helle Zeichen, die Hintergrundfarbe den Attributen invers und heller Hintergrund. Und eben hierin liegt die Krux! Bildschirmausgaben, die auf einem Monochromschirm gut aussehen können in Farbe leicht unmöglich erscheinen!

S o f t w a r e / H a r d w a r e: Bildschirmsequenzen

Hier einmal die Aufstellung der verschiedenen Farben und deren Wirkung auf dem Monochrom-Schirm. Aus Platzgrnden verwende ich folgende Abkürzungen: US = Unterstrichen, HE = Hell, IN = Invers und HG = Hintergrund:

Farbnummer Dezimal	Binär	Farbe	Wirkung bei Monochrom Vordergrund	Hintergrund
0	000	Schwarz		
1	001	Rot	US	
2	010	Grün		IN
3	011	Gelb	US	IN
4	100	Blau	HE	HG
5	101	Magneta	US HE	HG
6	110	Cyan	HE	IN HG
7	111	Weiß	US HE	IN HG

Erinnern wir uns daran, dar RAM 6.x bei identischer Vorder- und Hintergrundfarbe einfach ein Bit der Vordergrundfarbe umknipst, welches sich im Monochrom-Betrieb nicht auswirkt: Wie unschwer zu erkennen ist es Bit 1 des Attribut-Bytes, also die Farbe Grün, genauer gesagt den Grünanteil der Vordergrundfarbe.

Bleibt noch die Frage: Wie sag' ich's meinem Memotech?

Die Vordergrundfarbe kann z.B. mit ^P bis ^W gesetzt werden:
^P setzt Farbe 0 (Schwarz), ^Q setzt Farbe 1 (Rot), ... ^W setzt Farbe 7 (Weiß)

Die Hintergrundfarbe kann z.B. mit ^D gefolt von 0 bis 7 (also ^§, ^A, ..., ^G) gesetzt werden. Naheliegenderweise:
^D^§ setzt Farbe 0, ^D^A setzt Farbe 1, ..., ^D^G setzt Farbe 7.

Sollen, was ja oft zumindest im Monochrom-Modus der Fall ist, beide Farben bzw. die Monochrom-Attribute auf einen Schlag gesetzt werden, so geht das mit ^F gefolgt von dem Attribut-Byte.

Mit Esc ^ bzw. Esc _ gefolgt von einem Byte kannst Du unter RAM 4.x oder 6.x das aktuell gültige Attribut-Byte mit dem angegebenen Byte mit ODER bzw. UND verknüpfen. Die ODERierung kannst Du verwenden, um bestimmte Bits des Attribut-Bytes zu setzen, ohne die anderen zu verändern. Entsprechend kannst Du mittels UNDierung einzelne löschen (der Freak sagt hierzu auch wegmaskieren).

Es gibt noch einige weitere Bildschirmsequenzen, mit denen Du das Attribut-Byte verändern kannst - schau doch einfach mal ins Handbuch oder in ASCII6.KLX.

Übrigens: Welche Wirkung die Attribute haben hängt davon ab, wo Du den Monitor an der FDX anschließt:
- Farbe, wenn Du ihn an dem 9-poligen Stecker unten anstöpselst
- Monochrom, wenn Du den Chinch-Stecker oben verwendest.

Hast Du einen Bildschirm, der weder RGB-Farbe noch Monochrom, sondern, wie z.B. Claudios mit seinem Schwarz-Grau-Weiß-Monitor, der mich zu diesem Artikel veranlaßte, schick mir doch mal Deine Anmerkungen dazu in Bezug darauf, was Du noch lesen kannst.

A s s e m b l e r: Flag-Macro

Flag-Macro (Claudio Romanazzi, 3070)

Ich muß ja doch gestehen, daß mein kostbares Macro von Herbert verbessert wurde. Daß mich das etwas fuchst, kann bestimmt jeder verstehen. Also macht ich mich daran und suchte nach Verbesserungen. Erst mal habe ich alles abgetippt und durch meinen Assembler gejagt. Erstes Fazit: ein Fehler lies sich nicht ausmerzen. Ein Anruf bei Herbert brachte einen Assemblerfehler zutage. Jetzt lief alles, doch mir erschien das alles zu lang und umständlich. Ich wälzte als meine Unterlagen, ob es denn nichts besseres gäbe und ich wurde fündig!

Anm.d.HzN: Das Listing des Macros folgt aus Platzgründen auf der nächsten Seite (Sonst wäre es über die Seitengrenze gegangen, was es schlechter lesbar macht.)

Tja, was fällt auf? Der Text ist leicht verkürzt und vereinfacht, soweit es den Text von Herbert betrifft. Dafür sind einige Sachen dazugekommen.

Herbert hatte entgegen seiner Ankündigung das A-Register vergessen, jetzt wird es mitbehandelt.

Eine weitere Fehlermeldung ist dazugekommen. Der Offset wird jetzt auch noch auf Gültigkeit abgefragt.

Wie man anständige Fehlermeldungen schreibt, hat Herbert hier nur mal nebenbei demonstriert. Ich mußte ihn erst mal anrufen, um zu kapieren, daß die Zeilen, die mit Fehler: (bei Herbert mit ERR:) anfangen, aufgrund ihrer Struktur so wie sie sind als Fehler von Assembler angezeigt werden und sozusagen kostenlos dazu die Werte, die falsch sind, einsetzen. Darauf muß man erst einmal kommen.

Resumee: Die Maschinensprache ist mit Hilfe eines Macros wieder etwas benutzerfreundlicher geworden und ich wünsche mir mehr solcher Art von Zusammenarbeit!

Anmerkung: Da ich Flags sehr intensiv nutze, die Lesbarkeit äußert wichtig, das Zeichen noch nicht vergeben ist und damit im Club jeder weiß was gemeint ist, steht am Anfang des Flag-Namens ein '%' als Kennzeichnung. Der Witz ist ja der: Lese ich nach längerer Pause meine Sources durch, finde ich zwar Namen, doch ich weiß nicht welcher Art Macro da gerade steht. Das Kennzeichen '%' zeigt mir an, daß hier ein Flag steht.

Beispiel von Herbert jetzt neu:

```
FlagDef %Otto,A,0
```

Alles klar??

Anm.d.HzN: Toll Claudio!

- Übrigens mache ich die Namen von Flags im Source meiner Programme i.a. dadurch kenntlich, daß der Name mit FLG oder FLAG beginnt oder endet. Bit-Informationen haben i.a. den Text BIT am Anfang oder Ende des Namens. Aber wie man die diversen Arten von Labels (Upro-Einstieg, Upro-Intern, Tabelle, Puffer, Bit-Info, Flag, ...) unterscheidbar benennt ist auf jeden Fall Geschmackssache. Erfreulicherweise sind die Zeiten, da Labels noch 6 oder 8 Stellen kurz sein mußten vorbei, so daß man sie sprechender machen kann.
- Die Offset-Prüfung habe ich noch etwas korrigiert. Der Offset darf sich nämlich im Bereich -128 bis +127 bewegen. +130 hat Claudio leider zugelassen, -10 aber nicht.

A s s e m b l e r: Flag-Macro

```
Flag    Macro FReg,RSB,FBit,FOffset
        local Hilf,RSB1,FBit1,FReg1,FOffset1,G

        Hilf    aset 01000000b                  ; Startwert
         irpc   XX,BRSbrs                       ; gr + kl Buchstaben testen
         if     '&RSB' = '&XX'                  ; wenn Übereinstimmung
          RSB1  aset Hilf                       ; weise Wert zu
         endif
         Hilf   aset Hilf + 01000000b           ; nächster Wert
         if     Hilf = 100000000b               ; wenn erste 3 Letter fertig
          Hilf  aset 01000000b                  ; Startwert wie oben
         endif
         endM                                   ; gilt nur für irpc

        if      Nul RSB1                        ; wenn RSB1 nicht gesetzt
Fehler: FlagDefMacro mit ungültiger der Option: RSB
        endif

        if      (FBit < 0) or (FBit >7)         ; ungültige Bits ausschließen
Fehler: FlagDefMacro mit ungültigem Bit: FBit
        endif

        FBit1   aset FBit shl 3                 ; Bitnummer in richtige Pos.

        if      Nul FOffset                     ; ist kein Offset da
         FOffset1 aset 0                        ; dann nimm 0
        else
         FOffset1 aset FOffset                  ; sonst nimm den angegebenen
        endif

        if      FOffset1 > 127 and FOffset1 < -128 ; ungültigen Offset ausschließen
Fehler: FlagDefMacro mit ungültigem Offset: FOffset
        endif

        Hilf    aset 0                          ; Startwert
         irpc   XX,BCDEHLMAbcdehlma             ; diese Buchstaben testen
         if     '&FReg' = '&XX'                 ; wenn Übereinstimmung
          FReg1 aset Hilf                       ; weise Wert zu
          db    0cbh                            ; erstes Befehlsbyte
         endif
         Hilf   aset (Hilf + 1) and 00000111b   ; nächster Wert höchstens 7
         endM                                   ; gilt nur für irpc

         if     '&FReg' = 'X' or '&FReg' = 'x'  ; jetzt IX abfragen
          FReg1 aset 6                          ; weise Wert zu
          db    0ddh,0cbh,FOffset1              ; konstr. Indexregister-Code
         endif

         if     '&FReg' = 'Y' or '&FReg' = 'y'  ; jetzt IY abfragen
          FReg1 aset 6                          ; weise Wert zu
          db    0fdh,0cbh,FOffset1              ; konstr. Indexregister-Code
         endif

        if      Nul FReg1                       ; wenn kein Wert zugewiesen
Fehler: FlagDefMacro mit ungültigem Register: FReg
        endif

        db      low (RSB1 or FBit1 or FReg1)    ; das interssante Bit erzeugen
        endM
```

Z C P R 3.3: Aliase

Aliase für Libraries (Herbert zur Nedden, 2000)

Es ist wieder mal so weit! Ich habe im Laufe der Zeit das eine oder andere Alias für mich gebastelt, und möchte es auf diesem Wege der Öffent- und damit Lächerlichkeit preis geben.

Wozu überhaupt Aliase? Hier einige Gründe dafür:

a) Wenn ich mich an bestimmte Namen von Programmen oder auch Programme gewöhnt habe, können mir Aliase vorgaukeln, das es sie noch gibt. Das ist insbesondere dann praktisch, wenn ich die alten Aufrufe in diversen ZEX-, SUB- oder Alias-Scripts verwendet habe, und die nicht alle ändern will.
b) Um bestehende ZEX- und SUB-Scripts abzulösen. Der Vorteil hierin liegt vor allem in der Platzersparnis, da eine Datei mindestens 2kB Speicherplatz auf der knappen RAM-Floppy oder Systemdiskette frißt; das Alias jedoch nur einige Bytes. Obendrein sind Aliase flexibler, so daß ich Erweiterungen machen kann.
c) Um z.T: komplexe Abläufe durch einen Befehl aufrufen zu können; dabei können häufig wiederkehrende Teilaufgaben unterprogrammartig in ein eigenes Alias ausgelagert werden. Das wäre mit ZEX- und SUB-Scripts schwer machbar, da sie nicht ohne weiteres geschachtelt werden dürfen, und die Chose schnell unübersichtlich wird, wenn ich anfange zig solcher Script-Dateien zu haben.
d) Die Parameterisierung von Aliasen ist recht flexibel. Ich kann sogar von einem Parameter nur das Laufwerk oder den Dateinamen abgreifen - mit SUB-Scripts muß ich mich auf den ganzen Parameter beschränken. Zu allem Überfluß werden NamedDirectory-Angaben durch den Alias-Prozessor in Laufwerk/User umgesetzt, so daß ich damit Programme, die NamedDirectories nicht unterstützen über Aliase trotzdem mit NamedDirectories arbeiten lassen kann.

Hier nun einge Aliase:

1. Ich habe QCC, hatte mich aber an MFT (Kopierprogramm) und MTC (Vergleicher) so gewöhnt. Hier helfen zwei einfache Aliase:

```
MFT  qcc $* /ßv          Nutze QCC, aber ohne Vergleichen, also mit
                         der Option ßv (= Nicht-Verify).
MFC  qcc $* /ßc          Nutze QCC, aber ohne Kopieren, also mit
                         der Option ßc (= Nicht-Copy).
```

   Aufruf: So wie MFT und MFC

2. Diskette formatieren und, falls o.k. auch prüfen:

```
FR   form6  $TD1?;       Formatiere die Disk im angegebenen Lauf-
                         werk (ohne zu fragen!).
     if ßer;rcheck6 $TD1?;fi   Falls O.K. (if ßer), dann Prüfen.
```

   Aufruf: FR <Laufwerk>: z.B. FR B:

3. Erstellen von KLIX-Overlays - und zwar ohne mit GETREL.SUB zu arbeiten, da ein Alias i.a. schneller ist. Und dabei natürlich im Fehlerfalle abbrechen, denn wozu gibt es IFs. Die erste Variante ist für den SLR-Assembler (bei mir heitß er M90) und SLR-Linker (L90), die zweite ist für den M80 und L80. Da letzterer die ZCPR-Fehlerflags nicht setzt, sind die diversen IFs sinnlos und entfallen.

Z C P R 3.3: Aliase

```
G    m90 $1/ru;                          Assembliere erst mal den Source als REL.
     if ßer;l90 /a:100,quelle1/n,$1,/e;  Wenn alles O.K. (if ßer), dann Linke
                                         das Programm auf Adresse 100h.
     if ßer;l90 /a:200,quelle2/n,$1,/e;  Dto. für Adresse 200h.
     getrel50 $1 /o;era $1.rel;          Wenn soweit alles O.K., dann mit GETREL50
                                         das KLX erzeugen und das REL löschen.
     if $2<>;load61 $1;                  Wurde beim Aufruf ein 2. Parameter angege-
                                         ben, dann lade das KLX!
     zif                                 Beende alle IFs.

G    m80 $1,=$1;                         Assembliere erst mal den Source als REL.
     l80 /a:100,quelle1/n,$1,/e;         Linke das Programm an Adresse 100h.
     l80 /a:200,quelle2/n,$1,/e;         Dto. für Adresse 200h.
     getrel50 $1 /o;era $1.rel;          Dann mit GETREL50 das KLX erzeugen.
     if $2<>;load61 $1;fi                Wurde beim Aufruf ein 2. Parameter angege-
                                         ben, dann lade das KLX.

Aufruf: G <Programm>                     z.B. G DIJEY    KLX erzeugen
        G <Programm> X                   z.B. G DIJEY X  KLX erzeugen und laden
```

4. Ein Anzeige-KLX erstellen. Dazu mußt Du erst mal mit MTX-Edit (oder was auch immer) den anzuzeigenden Text mit den entsprechenden Steuerzeilen für MsgToZ80 erstellen. Den Rest der Verarbeitung überlasse ich einem Alias:

```
MTK  msgtoz80 $1;                        MsgToZ80 aufrufen, um aus dem MSG den As-
                                         sembler-Source zu erzeugen.
     g $1 $2;                            Aus dem Source das KLX machen und ggf.
                                         gleich laden. Dazu dient natürlich das
                                         o.g. Alias G!
     era $1.bak;era $1.z80;era $1.rel    Temporären Kram löschen.

Aufruf: MTK <Programm>                   z.B. G DJHELP    KLX erzeugen
        MTK <Programm> X                 z.B. G DJHELP X  KLX erzeugen und laden
```

5. Kommen wir zum Arbeiten mit Libraries (*.LBR). Ich verwende Libraries zunehmend für Sicherungskopien von Programmen oder anderen Dingen, die mehrere Dateien umfassen. Sogar für Sicherungen von der RAM-Floppy während ich noch am Programm bastle. So erreiche ich es, daß meine Disketten übersichtlicher bleiben und ich zusammengehörige Dinge auch zueinander zuordnen kann. Beachte, daß ich immerhin 1.76 MB formatierte Kapazität auf meinen Scheiben habe. Das Verteilen über unterschiedliche User-Bereiche ist m.E. unpraktischer. Mit LPUT und LBREXT ist das Arbeiten mit Libraries ja genauso einfach wie mit einem Kopierprogramm geworden:
   - Statt dem Sichern mit QCC mache ich es halt mit LPUT
   - Das holen von Diskette erfolgt statt mit QCC halt mit LBREXT.

   Außerdem verwende ich Libraries für echte Backups, sprich um eine bestimmte Version eines Programmes dauerhaft zu sichern, oder um Programme auf PD-Disketten weiterzugeben. In diesen Fällen habe ich die Dateien in der Library gerne geCRUNCHt, da das enorm viel Platz sparen kann - obendrein kann LBREXT die Dateien beim Herausholen sogar dekomprimieren, so daß das Herausholen keinen zusätzlichen Aufwand macht.

   Leider kann LPUT nicht CRUNCHen; außerdem will ich die FOR-Dateien grundsätzlich nicht CRUNCHen (sie sind eh sehr klein und einige ältere Library-Programme wie NULU können nicht UNCRunchen - so kannst Du damit wenigstens in der FOR-Datei lesen, was Sache ist). Außerdem: gibt es ein .COM und ein .CIM

der gleichen Datei, darf ich nur eines CRUNCHen, da das zweite sonst das erste überschreiben würde - CRUNCHen macht aus .COM und .CIM einfach .CZM!

Im Laufe der Arbeit habe ich mir vier Aliase erstellt, die folgende Aufgaben erledigen (alle verwenden User 1 als Work-Bereich):

CRUN Komprimieren von Dateien, die o.g. Spielregeln für .FOR, .COM, .CIM beachtend. Dies geschieht vom akt. User auf User 1.
(CRUN rufe ich nie explizit auf; LMAKE und LCRUN brauchen es.)

LVER Prüfen einer Library: Alles aus der Libaray herausholen, auf User 1 packen und mit dem, was im akt. User steht vergleichen.

LMAKE Library erzeugen: Dateien des akt. Users komprimieren, in eine Library stecken und diese dann prüfen.

LCRUN Library Crunchen: Alles aus einer bestehenden Library herausholen, komprimieren, wieder in die Library stecken und diese dann prüfen.

Bevor ich zu den Aliasen komme, ein paar Anmerkungen zu QCC. Das ist ein Kopier- und Vergleichsprogramm, welches mehrere Source-Angaben akzeptiert und zusätzlich ggf. die Sourcedateien nach dem Kopieren löscht und/oder die Ziel-Dateien mit dem Archiv-Bit versieht. Wenn Du QCC nicht hast, kannst Du statt dessen mit ACOPY, einem Dateivergleichsprogramm und ERA die gleichen Effekte erreichen. Ich habe in den Alias-Beschreibungen jeweils angegeben, was QCC tut. Zu den Aliasen:

CRUN era 1:*.*;	*Lösch erst mal User 1*
if ßex *.cim;qcc 1:=*.for /m;	*Wenn es keine CIM-Dateien gibt (if ßey *.cim), dann kopiere nur die FOR-Dateien nach User 1 und lösche sie im akt. User (Option M=Move). Das Löschen muß erfolgen, damit die Dateien nicht anschließend aus Versehen geCRUNCHed werden. Nach dem CRUNCHen hole ich sie wieder zurück.*
else;qcc 1:=*.for *.com /m;fi;	*Anderenfalls, d.h. wenn es CIM-Dateien gibt, kopiere die FOR- und die COM-Dateien nach User 1 und lösche sie im akt. User.*
crunch *.* 1:;	*Komprimiere (CRUNCH) die restlichen Dateien nach User 1.*
qcc 1:*.for 1:*.com	*Kopiere die FOR und COM-Dateien von User 1 wieder zurück, da sie ja vorhin ggf. gelöscht wurden. Hier brauche ich nicht zu unterscheiden, ob ich vorher nur die FOR-Dateien oder auch die COM-Dateien kopiert habe. Ist ein .COM auf User 1 so ist es mit dem im akt. User identisch.*

CRUNCH Version 2.8 kann zwei Dinge leisten, mit denen Du dieses Alias etwas vereinfachen kannst:

- Bestimmte Dateien nicht CRUNCHen sondern immer kopieren - es sei denn der Name der Datei wurde komplett angegeben. Das wird z.B. für LIBRARIES (*.LBR) gemacht. Welche nicht geCRUNCHT werden kannst Du installieren.
- Das Archiv-Bit beachten.

Damit kannst Du CRUN wie folgt realisieren, wenn Du CRUNCH28 so installiert hast, daß es FOR-Dateien nicht CRUNCHt:

Z C P R 3.3: Aliase

```
CRUN  era 1:*.*;                    Lösch erst mal User 1.
      da *.* -a                     Lösch bei allen Dateien im akt. User
                                    das Archiv-Bit.
      if ex *.cim;qcc 1:=*.com /s;fi Wenn es CIM-Dateien gibt (if ey *.cim),
                                    dann kopiere die COM-Dateien nach User
                                    1 und setz bei ihnen im akt. User das
                                    Archiv-Bit (Option S=Source-Archivbit).
      crunch28 *.* 1: /a;           Komprimiere (CRUNCH) die Dateien mit
                                    gelöschtem Archiv-Bit nach User 1.
      da *.com -a                   Lösche die Archiv-Bits der COM-Dateien.

LVER  era 1:*.*;                    Lösch erst mal User 1
      da *.* -a                     Lösch bei allen Dateien im akt. User
                                    das Archiv-Bit.
      lbrext $1 1:*.*;              Hole die Dateien aus der angegebenen
                                    Library (Parameter $1) nach User 1; da-
                                    bei dekomprimiert LBREXT automatisch.
                                    Falls LBREXT bei Dir nicht dekompri-
                                    miert, installier's halt um.
      qcc 1:*.* /ßcmd;              Vergleiche die Datien von User 1 mit
                                    denen im akt. User, lösche die gleichen
                                    aus User 1 und setze bei deren Pendants
                                    im akt. User das Archiv-Bit. Optionen
                                    von QCC: ßC=Nicht Copy, M=Move, D=Dest-
                                    Archivbit
      dir 1:                        Sie nach, was in User 1 noch übrig ist;
                                    wenn alles stimmt, ist User 1 leer und
                                    im akt. User haben alle Dateien das
                                    Archiv-Bit gesetzt.
      da *.* ßa;                    Zeige all die Dateien des akt. Users
                                    an, die das Archiv-Bit nicht gesetzt
                                    haben (ßa). Es sollten keine sein!

LMAKE crun;                         Kompirimiere mit Alias CRUN (s.o.)
      lput $1 1:*.*;                Lege die Library an, indem die auf User
                                    1 stehenden komptimierten Dateien in
                                    die Library gesteckt werden.
      lver $1                       Prüfe die Library mit Alias LVER (s.o.)

LCRUN era *.*;era 1:*.*;            Lösche alles, was im Weg ist.
      lbrext $1 *.*;                Hole alles aus der angegebenen Library
                                    in den akt. User; LBREXT dekomprimiert.
      ren $1.old=$1.lbr;            Benenne die Original-Library in *.OLD
                                    um - nur so zur Sicherheit.
      lmake $1;                     Lege die Library neu mit Alias LMAKE
                                    (s.o.) an; dabei werden die Dateien be-
                                    kanntlich geCRUNCHt.
      if in Weg $1.old;era $1.old;fi Lösche die Sicherungskopie der Library
                                    (das *.OLD-Teil), wenn alles O.K. ist,
                                    sprich der Anwender auf Y oder die
                                    Leertaste drückt.

Aufruf: CRUN                        Wird eigentlich nur intern aufgerufen
        LVER <Libaray>              z.B. LVER d10:dijey300
        LMAKE <Libaray>             z.B. LMAKE d10:dijey300
        LCRUN <Libaray>             z.B. LCRUN d10:dijey300
```

H a r d w a r e: IBM-Tastatur

IBM-Tastatur von ESCOM am MTX (Herbert zur Nedden, 2071)

In c't 10/88 und 11/88: 'PC-Tastaturen am Atari-ST' wurde beschrieben, wie man eine IBM-Tastatur an einen Atari ST anschließen kann. Unlängst fiel mir eine IBM-Tastatur der Firma ESCOM in die Hände, die mir aufgrund der schwarzen Farbe sehr zusagte - schließlich ist auch der MTX (so heißt mein Computer) schwarz. Das diese Tastatur obendrein noch einen Taschenrechner mit eigenem LCD-Display drin hat, machte sie nur um so interessanter. Bei mir läuft diese Tastatur schon seit einiger Zeit sehr zufriedenstellend!

Die in der c't aufgezeigte Lösung zum Anschluß einer IBM-Tastatur an den Atari-ST wurde durch ein Interface gelöst, welches die seriellen Daten von der Tastatur in ein Schieberegister schaufelte. Aus diesem konnten sie dann entweder parallel (d.h. alle 8 Bit auf einen Schlag) oder über ein zweites Schieberegister seriell (also Bit für Bit), jedoch durchaus mit einer anderen Geschwindigkeit (Kenner sprechen von der Baudrate) abgeholt werden. Das Atari-ST-Interface ging letzteren Weg, da der Computer in Bezug auf die Geschwindigkeit und das Format, mit der die Daten von der Tastatur am Rechner ankommen sehr eigen ist, die Daten jedoch gerne seriell haben möchte.

Andererseits konnte ich c't 6/88 und 7/88: 'Knöpfchen, Knöpfchen' entnehmen, daß PCs auch Daten an die Tastatur senden können und es wohl auch tun, was mit den in den Interfaces für den Atari-ST nicht möglich ist.

Mein Ziel war es, eine IBM-Tastatur mit minimalem Aufwand so an meinen Rechner anzuschließen, daß ich Daten von der Tastatur empfangen (das ist ja der Hautpzweck), aber auch an diese senden kann, z.B. um die schönen Lämpchen für Caps-Lock oder NumLock ein- und auszuschalten.

Betrachten wir dazu die Datenformate, mit denen IBM-Tastaturen arbeiten:

PC-Modus: 1 Startbit gefolgt von 8 Datenbits
AT-Modus: 1 Startbit, 8 Datenbits, 1 Paritybit ungerade, 1 Stopbit

In den c't-Artikeln, in denen das Atari-ST-Interface für AT-Tastaturen beschrieben wird, wurde über das Datenformat, insbesondere über das Parirybit gemosert - damit hätte der ACIA (oder wie das Teil heißt) des Atari-ST Probleme.

Vergleiche ich jedoch die Datenformate mit denen einer normalen asynchronen seriellen Schnittstelle (wie z.B. einer RS232), so gefällt mir der AT-Modus auf Anhieb - es ist das serielle Format 8O1: 8 Bit pro Byte, odd (= ungerade) Parity und 1 Stop-Bit! Der PC-Modus hingegen gefällt mir überhaupt nicht!

Schnittstellen, die mit diesem asyncronen Format Daten übertragen haben als Schnittstellen-Baustein i.a. einen UART: Universal Asynchronous Reciever/Transmitter, also einen universellen asynchronen Empfänger/Sender. Im folgenden werden ich den Schnittstellen-Baustein daher als UART bezeichnen.

Üblicherweise müssen sich zwei zusammenarbeitende serielle Schnittstellen über die Baudrate einigen - meist muß sich dabei der Computer dem externen Gerät anpassen. Technisch sieht das dann so aus, das im Computer ein entsprechender Takt erzeugt und dem UART zur Verfügung gestellt wird. IBM-Tastaturen sind praktischerweise so freundlich, diesen Takt für die serielle Übertragung selbst bereitzustellen - und zwar für beide Übertragungsrichtungen. Daher brauch ich mich nicht einmal um die Baudrate zu kümmern.

H a r d w a r e: IBM-Tastatur

Normalerweise arbeiten die seriellen Schnittstellen mit ±12 Volt (daher auch die Bezeichnung V24, da zwischen +12 und -12 Volt ein Spannungsunterschied von 24 Volt ist). Computerintern wird jedoch mit TTL-Pegel, also 0 und 5 Volt gearbeitet. Daher sitzen i.a. Wandler-ICs zwischen dem UART und dem Stecker am Computer. Diese ICs machen aus den für das Computerinnere i.a. ungesunden ±12 Volt die 0 und 5 Volt - und natürlich auch umgekehrt.

Da IBM-Tastaturen mit TTL-Pegel, sprich 0 und 5 Volt arbeiten, müssen ggf. vorhandene Wandler-ICs beiseitegelassen, überbrückt, umgangen o.ä. werden. Um allen ewatigen Problemen aus dem Weg zu gehen, habe ich die Anschlüsse des UART, die ich benötige hochgebogen, damit sie keinen Kontakt woanders hin haben - sicher ist sicher -, und mein Interface direkt an den UART gelötet.

Erster Erfolg!

> Ich habe einfach die IBM-Tastatur direkt an den UART meiner seriellen Schnittstelle gehängt: Die Tastatur-Daten an die Daten-Empfangsleitung (Recieve Data, RxD oder wie der Pin auch immer bezeichnet sein mag) und den Tastatur-Takt an den Empfangs-Takteingang (Recieve-Clock, RxC, ...). Dazu noch 5 Volt und Masse für die Tastatur.
>
> Damit konnte ich einwandfrei die Daten meiner IBM-Tastatur empfangen! Ich finde, daß mein Interface dafür doch recht einfach geraten ist.

Anmerken möchte ich an dieser Stelle, daß sich meine IBM-Tastatur per DIP-Schalter in den AT-Modus versetzen läßt.

Kommen wir zum Thema 'Senden an die Tastatur'. In den o.g. Atrikeln von c't stand der Hinweis, daß der Rechner, so er senden möchte, die Taktleitung der Tastatur für eine Weile auf Low zieht und dann sendet. Allerdings wurde dann noch erwähnt, daß das Protokoll etwas aufwendiger sei, aber es eh nur in PCs mit einem 8042-Interface möglich sei, an die Tastatur zu senden, und eben dieses Interface sich um das Protokoll kümmere.

Nun habe ich keinen PC an dem ich nachmessen kann, was auf den Leitungen passiert, und folglich versucht, mit den Informationen aus c't der Tastatur was zu senden. Es klappte nicht so recht. Ich fragte bei der c't-Redaktion an, ob man mir nicht weitere Informationen zukommen lassen könne. Die Informationen kamen prompt (vielen Dank!). Übrigens konnten mir keine anderen Stellen (Hardwarelieferanten/Softwarehäuser) helfen.

So kam ich an das Protokoll, wie der Rechner der Tastatur mitteilt, daß er senden will und wie es dann mit dem Senden an und für Sich vor sich geht:

1. Der Rechner legt die Taktleitung für mindestens 60 ms auf Low.
   Hiermit unterbindet der Rechner jegliche Sendeaktivitäten der Tastatur.
2. Der Rechner bereitet sich auf das Senden vor, und zwar so weit, daß das Startbit (Low) schon auf der Datenleitung anliegt und wartet einen kurzen Augenblock in diesem Zustand.
   Sprich: Der Rechner legt auch die Datenleitung auf Low.
3. Der Rechner gibt die Taktleitung frei, d.h. läßt sie auf High-Pegel gehen.
4. Nun holt sich die Tastatur die Daten, indem sie selbst den Takt für die Übertragung liefert.

Es klingt eigentlich recht einfach, aber ....

H a r d w a r e: IBM-Tastatur

Wie sendet ein serieller Baustein? Ganz einfach: wenn der Takt kommt, geht es los! Aber genau da ist der Haken: Die Tastatur liefert erst dann den Takt, wenn das Senden schon angefangen hat, sprich das Startbit raus ist.

Daher macht mein Interface zusammen mit meinem Treiber folgendes, wenn gesendet werden soll:

1. Es legt die Taktleitung auf Low und wartet ca. 60 ms.
2. Es fängt mit dem Senden an, indem es sich selbst einen Takt liefert, bis die Sendeleitung auf Low liegt, d.h. das Startbit auf der Leitung ist.
3. Dann gibt es die Taktleitung frei.
4. Nun holt sich die Tastatur die Daten, indem sie den Takt liefert.

Und das funktioniert!

Da die Tastatur nur eine Daten und eine Taktleitung hat, kommen sogenannte Open-Collector-Treiber zum Einsatz. Normale TTL-Treiber leifern am Ausgang Low, High oder sind ggf. hochohmig. Bei Low oder High ist dieser Pegel durch den Ausgang erzwungen; hochohmig bedeutet, daß der Ausgang quasi abgeklemmt ist.
Open-Collector-Ausgänge hingegen sind der Kollektor eines NPN-Transistors, dessen Emitter an Masse liegt. Ist der Transistor geöffnet, liegt der Ausgang an Masse; ist er geschlossen, so hängt der Ausgang in der Luft. Erst, wenn er über einen PullUp-Widerstand an +5 Volt hängt, liefert der Ausgang bei gesperrtem Transistor High. Dank dieser Konstruktion können zwei Open-Collector-Ausgänge aneinander geschaltet werden. Liefert einer der beiden Low, so ist das gemeinsame Signal Low.

Und just diese Open-Collector-Treiber verwenden die IBM-Tastaturen. Daher kann der Rechner jederzeit die Takt- oder Datenleitung der Tastatur auf Low ziehen, ohne Schaden anzurichten. Naturgemäß verwende ich in meinem Interface ebenfalls Open-Collector-Treiber.

Was genau muß das Interface leisten:

1. Die Datenleitung der Tastatur muß an den Empfangs-Eingang des UART.
2. Der Sende-Ausgang des UART wird über einen Open-Collector-Treiber an die Datenleitung der Tasatur gelegt, damit auch gesendet werden kann.
3. Ein Eingang muß vorhanden sein, über den der Pegel der Datenleitung des UART abgefragt werden kann, um beim Senden das Anliegen des Stopbits (Low) erkennen zu können.
4. Ein Ausgang ist erforderlich, um im Interface einen Takt zu erzeugen, der benötigt wird, um das Senden soweit voranzutreiben, bis das Startbit anliegt.
5. Die Taktleitung der Tastatur ODER der selbsterzeugte Takt des Interfaces (siehe 4.) muß an den Sendetakt-Eingang des UART.
6. Die Taktleitung der Tastatur oder wahlweise das 'Gemisch' von 5. muß an den Empfgangstakt-Eingang des UART.
7. Die Taktleitung der Tastatur muß (per Open-Collector-Treiber) vorübergehend auf Low gezogen werden können.

H a r d w a r e: IBM-Tastatur

Der UART, den ich verwende ist ein Z80DART - genauer gesagt ein Kanal des DART. DART bedeutet Dual Asynchonous Reciever/Transmitter; das Dual weist darauf hin, daß der Baustein zwei UARTs enthält. Den Z80DART will ich hier nicht im Detail beschreiben, sondern mich auf die erforderlichen Signale und natürlich weiter hinten die Programmierung beschränken: (ein / vor einem Signal bedeutet aktiv Low)

RxD	serieller Dateneingang
TxD	serieller Datenausgang, High im Ruhezustand
RxC	Takteingang für's Empfangen (Mit diesen Signalen wird die Empfangs
TxC	Takteingang für's Senden bzw. Sende-Baudrate festgelegt.)

/DCD (Data Carrier Detect). Diese Eingangsleitung kann per Software abgefragt werden. Ich verwende sie, um das Anliegen des Startbits (Low) auf der Sendeleitung zu erkennen.

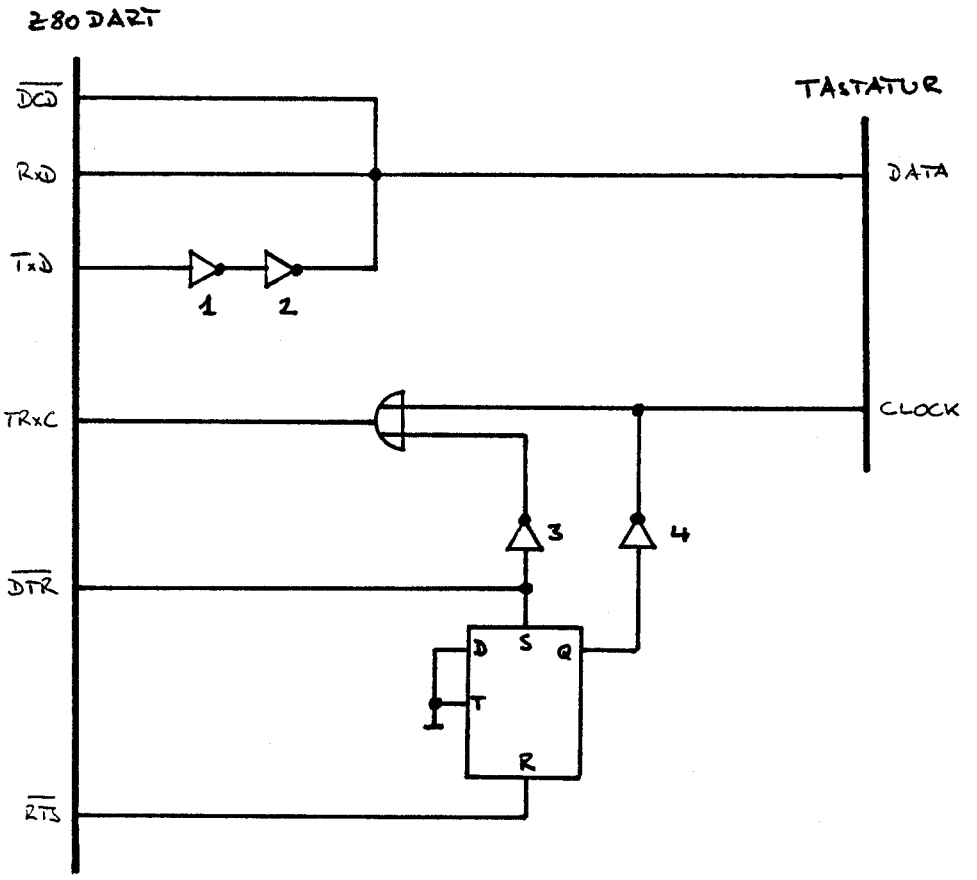
/DTR (Data Terminal Ready). Diese frei programmierbare Ausgangsleitung verwende ich, um die Taktleitung der Tastatur auf Low zu ziehen sowie, um den interface-internen Takt für das Beginnen des Sendens zu erzeugen.

/RTS (Request to Send). Diese Leitung kann per Software aktiviert werden. Deaktiviert wird sie, wenn dies per Software gefordert und der Sendepuffer leer ist bzw. leer geworden ist. Ich verwende sie, um die Taktleitung der Tastatur freizugeben.

Die Signale der IBM-Tasatur bezeichne ich im folgenden als:

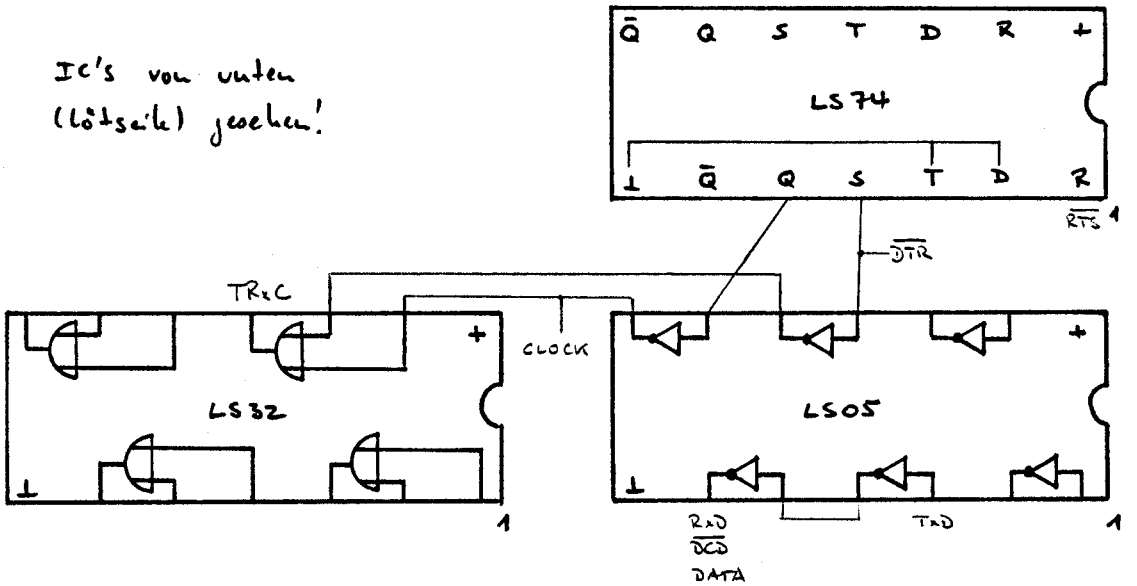
DATA	Datenleitung
CLOCK	Taktleitung

Hier der Schaltplan:



H a r d w a r e: IBM-Tastatur

Für diejenigen, die die Schaltung auf einer Lochraster-Platine löten wollen, hier die ICs und deren Verdrahtung - von UNTEN gesehen:



Kommen wir zur Funktionsweise des Interfaces. Das Interface kennt drei Betriebsmodi, die ich hier getrennt erläutern will. (In der TxD-Leitung liegen zwei Inverter, da ich den Inverter I3 für /DTR auf jeden Fall benötigte, und der 74LS05 noch genügend weitere Inverter frei hatte.)

Initialisierung:

Der DART wird zurückgesetzt; damit ist insbesondere der Sendepuffer leer, d.h. TxD ist High und zieht damit nicht DATA auf Low. Anschließend wird der DART initialisiert, wobei /DTR auf High und /RTS auf Low gelegt werden.

Da /RTS Low ist, wird das FlipFlop zurückgesetzt, damit dessen Ausgang Q auf Low und folglich der Ausgang des Inverters 4 auf High liegt. Damit ist CLOCK freigegeben. Weiterhin ist der Ausgang des Inverters 3 auf Low und somit arbeitet das ODER-Gatter als Treiber, da sein einer Eingang Low ist.

Empfangs-Betrieb:

/RTS ist Low, /DTR ist High und da keine Daten an die Tastatur gesendet werden, ist TxD ebenfalls High.

Folglich liegt das CLOCK an TxC und RxC (im Schaltplan gemeinsam als TRxC bezeichnet) und DATA an RxD an.

Da CLOCK und DATA seitens des Interfaces frei sind, kann die Tastatur senden, was sie auch tut, wenn Sie in die Tasten hauen. Die gesendeten Daten können problemlos vom Z80DART empfangen werden, da die Tastatur die Baudrate in Form des Taktes auf der CLOCK-Leitung bereitstellt.

H a r d w a r e: IBM-Tastatur

Senden von Daten and die Tastatur:

Das Senden erstreckt sich (s.o.) über mehrere Schritte, die nacheinander ablaufen müssen. Sollte ein Schritt fehlschlagen, muß ggf. wieder weiter oben aufgesetzt werden - doch dazu mehr weiter unten.

1. /DTR wird auf Low und /RTS auf High gelegt. Damit wird das FlipFlop gesetzt; sein Ausgang Q geht auf High und damit der Ausgang des Inverters 4 auf Low: CLOCK ist Low.
2. Warten ca. 60 Millisekunden (kürzer reicht vielleicht auch).
3. Das zu sendende Byte wird an den DART gegeben und /DTR solange zwischen Low und High hin- und hergeschaltet, bis /DCD auf Low liegt, d.h. das Startbit auf der Leitung ist. Das funktioniert, da CLOCK auf Low liegt und somit das ODER-Gatter als Treiber, jedoch diesmal für den Ausgang des Inverters 3, also das invertierte /DTR dient. Daher gelangen die Pegelwechsel von /DTR als Takt an den Sendetakt-Eingang - der DART legt mit dem Senden los.
4. Kurze Pause von 50 Mirosekunden, damit die Tastatur merkt, daß DATA = Low, was ihr signalisiert, daß der Rechner senden will.
5. /DTR wird wieder auf High und /RTS auf Low gelegt. Damit wird das FlipFlop zurückgesetzt und so CLOCK freigegeben.
6. Die Tastatur taktet nun munter drauflos und holt sich so die Daten ab. Da DATA mit RxD und sowie TxC mit RxC einander verbunden ist, empfängt auch der DART das eben gesendete Byte - daran kann der Treiber erkennen, ob sich die Tastatur die Daten geholt hat.
7. Die Tastatur bestätigt den Eingang des gesendeten Bytes i.a. mit einem ACK (0FAh). Auf dieses Byte sollte nun gewartet werden.
   Im Falle eines Reset der Tastatur sollte auf die Bestätiung eines erfolgreichen B.A.T. (Basic Assurance Test, 0AAh) gewartet werden.

Soll mehr als ein Byte gesendet werden, was z.B. beim Umschalten der Lämpchen der Tastatur erforderlich ist, geht das Ganze mit dem nächsten Byte von Vorne los.

Probleme beim Senden:

Gelegentlich funktioniert 3. nicht auf Anhieb. Das hängt u.a. vom Timing ab. Daher macht mein Treiber folgendes: höchstens acht Mal wird mit /DTR getaktet. Ist /DCD dann immer noch nicht auf Low, wird erneut mit 1. gestartet.

Den Erhalt eines Befehls quittiert die Tastatur meist sehr prompt - zum Teil so schnell, daß das Stop-Bit noch nicht ganz gesendet wurde, da sie es eh nicht zum Verständnis der Daten benötigt. Das bedeutet, daß sich das erste von der Tastatur gesendete Byte und das Stop-Bit überlagern - der DART interpretiert das Stop-Bit als Startbit und liest die gesendeten Daten beginnend mit dem 2. gesendeten Bit ein: An Stelle von 0FAh wird 0F4h empfangen, d.h. die um ein Bit nach links verschobene Information. Entsprechend kommt ggf. statt des 0FEh (Resend) ein 0FCh an. Der Treiber muß beides akzeptieren.
Diese Problematik tritt nur beim ersten von der Tastatur an den Rechner gesendeten Byte direkt nach dem senden eines Bytes an die Tastatur auf - die späteren Informationen von der Tastatur, also insbesondere die Tastatureingaben und die B.A.T.-Erfolgsmeldung kommen jedoch wieder einwandfrei!

Ein nicht erhaltenes ACK sollte die Software dazu veranlassen, den gesamten Sendevorgang erneut zu versuchen. Damit habe ich jedoch keine Probleme außer vereinzelt beim Zurücksetzen (Reset) der Tastatur nach dem Einschalten des Rechners. Dann genügt es jedoch, die Tastatur einmal kurz aus- und wieder einzustöpseln. Da die Tastatur jedoch beim Einschalten sich selbst zurücksetzt, kann der Treiber eigentlich auch daruauf verzichten.

H a r d w a r e: IBM-Tastatur

```
            (* Programm, um das Tastatur-Interface zu testen *)
           (* Turbo-Pascal 3.0, CP/M 2.2 oder ZCPR 3.3-System *)

            (* Copyright 1991, Herbert zur Nedden          *)

const DartCtl  : byte = $0f;      { Port-Adressen des DART: Control          }
      DartData : byte = $0d;      {                         Daten            }

var parity,stop,txlen,rxlen,dtr,rts: integer;

PROCEDURE RSInit;                 (* Z80DART initialisieren *)
begin
  { Konstanten fuer die Initialisierung des Z80DART }
  parity:=1;                      { ungerade Parity                          }
  stop:=1;                        { 1 StopBit                                }
  txlen:=3;                       { 8 Bit TxLaenge (Tx = Transmit = Sende)   }
  rxlen:=3;                       { 8 Bit RxLaenge (Rx = Receive = Empfangs) }
  dtr:=0;                         { DTR = 0, d.h. /DTR = High                }
  rts:=1;                         { RTS = 1, d.h. /RTS = Low                 }

  port[dartctl]:=24;   { Reset des Z80DART                                   }

  port[dartctl]:=1;    { kein Interrupt, kein Wait/Ready                     }
      port[dartctl]:=0;
  port[dartctl]:=3;    { RxLaenge, kein AutoEnables, RxEnable                }
      port[dartctl]:=64*rxlen+1;
  port[dartctl]:=4;    { Clock *1 (RxC & TxC nicht teilen), Stop Bits, Parity}
      port[dartctl]:=4*stop+parity;
  port[dartctl]:=5;    { DTR, TxLaenge, TxEnable, RTS                        }
      port[dartctl]:=128*dtr+32*txlen+8+2*rts;
end;

PROCEDURE SetDTR(b: byte);        (* DTR setzen *)
begin
     port[dartctl]:=5; port[dartctl]:=128*b+32*txlen+8;
end;

FUNCTION RSIn: byte;              (* Zeichen einlesen und ausgeben *)
const hexer: array[0..15] of char
      = ('0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F');
var b: byte;
begin
  if (port[dartctl] mod 2) > 0 then  { Ist da ein Zeichen ?                  }
     begin
     b := port[dartdata];           { Ja, dann holen und Hex ausgeben      }
     write(hexer[b shr 4],hexer[b and $f],' ');
     RSIn := b;
     end
  else
     RSIn := 0;                     { Nein, dann 0 zurueckgeben            }
end;
```

H a r d w a r e: IBM-Tastatur

```
PROCEDURE RSOut(b: byte);          (* Ein Byte senden *)
var i,j: integer;
begin
  rts := 0;                        { RTS = 0, d.h. /RTS = High                 }
  SetDTR(1);                       { DTR = 1, d.h. /DTR = Low  (setzt RTS mit) }
  delay(20);                       { etwas warten - mind. 60 ms                }

  port[dartdata] := b;             { Zeichen an den DART                       }
  port[dartctl] := 8;              { setze Ext/Status zurueck wegen DCD        }

  j := 0;                          { Zaehler, da nur begrenzte Zahl Versuche   }
  repeat
     j := succ(j);
     SetDTR(1); SetDTR(0);         { DTR =1, DTR = 0: einmal takten            }
     i := port[dartctl] and $8;    { /DCD ist Low, falls > 0                   }
     write(i:2);                   { DCD-Info ausgeben                         }
  until (i > 0) or (j > 7);

  if j > 7 then writeln('Retry')   { zu viele Versuche ?                       }
           else writeln;
  rts := 1;                        { RTS = 1, d.h. /RTS = Low                  }
  SetDTR(0);                       { DTR = 0, d.h. /DTR = High  (setzt RTS mit) }
end;

PROCEDURE RSSend(b:byte);          (* Senden und ACK abwarten *)
var k:integer;
begin
  k := 0;                          { Geduldsfaden                              }
  repeat
    k := succ(k);
    rsout(b);                      { Sendeversuch                              }
    { Senden bis ich das, was ich sende auch selbst wieder Empfange            }
  until (rsin = b) or (k > 10);

  if k > 10 then                   { zu viele erfolglose Versuche ?            }
     begin
     write('Hat nicht sollen sein!');
     exit;
     end;

  k := 0;
  repeat                           { warten auf ACK                            }
    k := succ(k);
  until (rsin in [$fa,$f4]) or (k>50); { 0f4h = SHL 0fah, siehe Text oben     }

  if k <= 50 then writeln('ACK')   { war's zu oft ?                            }
             else writeln('NACK');

  if b = 255 then                  { habe ich einen Reset (0ffh) gesendet ?    }
     begin
     repeat                        { dann warten auf B.A.T.-ACK (0aah)         }
        k := succ(k);
     until (rsin = $aa) or (k > 200);
     if k <= 200 then writeln('Reset') { hat der Reset geklappt ?             }
                 else writeln('Kein Reset?');
     end;
end;
```

H a r d w a r e: IBM-Tastatur

```
var b: byte;
    c: char;

BEGIN                                (* MAIN *)
  clrscr;
  rsinit;                            { Initialisiere die Schnittstelle          }
  repeat
  if keypressed then                 { Hat der Anwender eine Taste auf der nor- }
     begin                           { malen Tastatur gedr}cke? Wenn ja, dann:  }
     writeln; write('Befehl: ');     { Um Eingabe des Befehls bitten            }
     readln(c);                      { und ihn einlesen (nur 1. Zeichen)        }
     case upcase(c) of
     'H': halt;                      { H = Halt                                 }
     'R': begin;                     { R = Reset, d.h. sende OFFh               }
          RSSend(255);
          end;
     'L': begin;                     { L = Lampen, d.h. Sende OEDh gefolgt von  }
          write('  Lampen: ');       {     einem Byte, dessen Bit 0-2 je eine   }
          readln(b); b := b and 7;{     der Lampen CapsLock, NumLock, Scroll-  }
          RSSend($ed); RSSend(b); {     Lock bedeuten                          }
          end;
     'S': begin;                     { S = Sende das, was eingegeben wird       }
          write('  Senden: ');
          readln(b);
          RSSend(b);
          end;
     end; { case }
     end; { then begin }

  if RSIn = $f0 then                 { Lies naechstes Zeichen von der IBM-Tasta- }
                                     { tur und gib es aus. War es ein 0F0h, d.h. }
     begin                           { ein Break-Code-Prefix, gib das folgende  }
     b:=RSIn;                        { Zeichen auch aus und dann einen Zeilen-  }
     writeln;                        { Vorschub - zur Uebersichtlichkeit        }
     end;
  until false;                       { Loop for ever                            }
END.
```

Bei mir läuft die Tastatur mittels eines Interrupt-Treibers, d.h. daß der Z80DART nach Empfang eines Zeichens von der Tastatur die CPU per Hardware-Interrupt unterbricht. Die dadurch automatisch aufgerufene Interrupt-Routine verarbeitet das eingelesene Zeichen, stellt entsprechendes in den Tastaturpuffer und gibt anschließend die Kontrolle an das unterbrochene Programm zurück. Diese Lösung war für mich naheliegend, da die Z80CPU Hardware-Interrupts durch Z80-Peripherie-Bausteine vorbildlich - bis hin zur Proiritäten-Kette - unterstützt und ich so die CPU nicht mit unnötigen Polling-Aufgaben belaste: Die Interrupt-Routine (d.h. der Tastaturtreiber) wird nur dann aktiviert, wenn die Tastatur etwas gesendet hat.

Der neue KLX-Treiber für dieses Interface ist zusammen mit dem Maustreiber in KbdMou Version 2.0 (beachte die Versionsnummer 2.x!!!!) auf der PD KLICK.018.

Claudio Romanazzi hat den Treiber schon etwas erweitert: Beim Drücken einer bestimmten Taste ruft der Treiber ein Programm auf, mit dem Du eine Datei irgendwie (er hat's mir geschickt, ist noch in der Post) auf dem Bildschirm anklicken kannst - der Dateiname wird dann in den Tastaturpuffer gestellt. Damit verliert DiJey wieder etwas an Bedeutung, da das auch damit (Kommando X) geht ...

Sorry, aber diese Seite musste ich entfernen, weil ich für eine Veröffentlichung außer in der ehemaligen, gedruckten Form keine Freigabe habe.