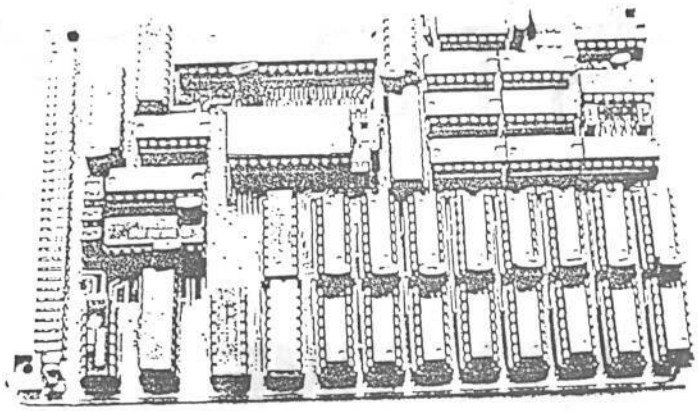
RTS-80

REAL-TIME OPERATING-SYSTEM

DAS Z-80 SYSTEM MIT UNIX FÄHIGKEITEN UND 16 MBYTE ADRESSRAUM



DIE 128K CPU-3 VON BITSCH COMPUTERSYSTEME

Sie haben sich für ein Z-80 ECB - System entschieden. Wir wollen Ihnen auf den folgenden Seiten eine Entscheidungshilfe bieten, die argumentativ ist und sich nicht in Werbesprüchen erschöpft. Die Vorteile des ECB - Bussystems sind Ihnen bekannt, wir brauchen daher darauf nicht näher einzugehen.

Warum Z-80 ?
Das immense Programmangebot auf Z-80 und CP-M 2.x macht diese Frage wohl überflüssig, Ausgereifte und professionelle Software ist immer noch das beste Entscheidungskriterium für ein Gesamtsystem. Zudem: Wer die Geschwindigkeitsvergleiche in den einschlägigen Fachzeitschriften verfolgt, wird feststellen, daß Z-80 Systeme mit 4 Mhz mit Leichtigkeit mit den heute angebotenen 16 Bit Systemen mithalten können, oft sogar schneller sind. Wir bieten unsere gebankten Systeme dagegen standardmässig mit 6 Mhz an, optional sogar mit 8 Mhz.

Warum gebankte Systeme ?
Der erste Vorteil liegt klar auf der Hand. In einem gebankten System gewinnt das Anwenderprogramm auf Anhieb einen sehr grossen Arbeitsbereich. Programme, die selbsttätig feststellen, wie gross der ihnen zur Verfügung stehende Datenbereich ist, können schon hierdurch einiges an Geschwindigkeit gewinnen. Hiermit ist jedoch

Banking-Konzeptes schon erschöpft. Nicht so unter RTS-80 : RTS-80 arbeitet mit internen Diskpuffern, die schon in der Normalversion zu einer erheblichen Geschwindigkeitssteigerung führen. In der gebankten Version kommt dies jedoch voll zum Tragen: fast die gesammte Systembank wird zusätzlich als schneller Diskpuffer benutzt. Dies bedeutet eine enorme Geschwindigkeitssteigerung.
Dieser Diskpuffer wird vom Betriebssystem verwaltet und nicht wie Notloesungen unter anderen Betriebsystemen vom Bios ! Denn nur das Betriebssystem kann wissen, was an Diskdaten auch wirklich gebraucht wird. Ins Bios ausgelagerte Diskpuffer wie unter anderen Betriebssystemen können im Gegenteil sogar eine Verlangsamung des Gesamtsystems bedeuten.

Die Vorteile einer solchen Diskpufferung sind enorm: Diskdaten kommen durch diese virtuelle Speichertechnik mit einer hohen Trefferrate nicht mehr von der Disk, sondern aus dem schnellen Arbeitsspeicher. Jedes normale CP/M -2 Programm wird dadurch ohne irgendwelche Eingriffe in das Programm selbst um ein vielfaches schneiler !
Hin- und her-PIPen wie unter anderen Betriebsszstemen mit ihren RAM- Disk Notloesungen und den vielen möglichen Fehlerquellen dabei entfallen ganz !

Wer dennoch nicht auf eine Ramdisk verzichten möchte, kann dies selbstverständlich auch unter RTS-80 haben. Wir empfehlen jedoch dann besser eine Harddisk zu implementieren, da sie aus dem Kosten- und Datensicherheitsaspekt die günstigere Alternative darstellt.

Die einzelnen Faktoren der Geschwindigkeitssteigerung von RTS-80 gegenüber anderen Betriebssystemen :

I.Statt wie andere Betriebssysteme waehrend einer Diskettenumdrehung ein oder zwei Sektoren einzulesen oder zu schreiben, liest und schreibt RTS-80 immer gleich mehrere zusammenhaengende Sektoren wahrend einer Umdrehung und puffert sie intern. Dies muss nicht etwa manuell wie unter anderen Betriebssystemen eingestellt werden, sondern das System macht dies selbstverständlich automatisch l

2.Diese Puffer bleiben für spätere Zugriffe erhalten. (Virtuelle Cache-Speichertechnik) Diskdaten kommen dann mit einer hohen Trefferrate nicht mehr von der Disk, sondern aus dem schnellen Arbeitsspeicher ! Ein intelligenter Least Recently Used (LRU) Algorithmus verwaltet die Puffer. Eine Geschwindigkeitssteigerung um den Faktor 20 kann daraus je nach Diskettenformat resultieren. Bei Programmen, die viele Random-acces Operationen machen (Datenbanken u. aehnl.), ist dies besonders gut zu beobachten. Notloesungen wie Ramdisks unter anderen langsamen Betriebssystemen mit ihren vielen Fehlerquellen werden dadurch überflüssig.

3.Noch größer ist die Geschwindigkeitssteigerung bei ausfuehrungsfaehigen Dateien (.COM Dateien). Hierfuer besitzt RTS-80 e'inen eigenen Programmladeoptimierer, der diese Dateien mit der maximal moeglichen Geschwindigkeit einliest !(Wordstar z.B. unter l Sekunde)

4.Da Directory-Operationen einen wesentlichen Teil der (langsamen) Diskaktivitäten ausmachen, wird das Directory immer

im schnellen Arbeitsspeicher gehalten,

Diese Geschwindigkeitssteigerungen kommen jedem normalen CP/M-2 Programm ohne irgendwelche Eingriffe und Änderungen zugute l Es muss also nicht wie unter anderen Betriebssystemen ein Eingriff in den (Source) Code des Programms vorgenommen werden, um Vorteile aus dem Banking-Konzept zu gewinnen !

Daneben gibt es unter RTS-80 noch eine Reihe· weiterer Geschwindigkeitsoptimierungen, die sie bitte der in unserer Info enthaltenen RTS-BO Kurzbeschreibung entnehmen.

Diese Geschwindigkeitssteigerungen machen es sinnvoll, den Adressraums fuer Programme auf Z-BO wesentlich zu erweitern. Dafuer wurde ein einfacher Mechanismus implementiert, der es erlaubt, bis zu 16 Mbyte grosse Programme unter RTS-BO zu fahren. (Siehe naechste Seite)

Ein weiterer Vorteil von RTS-80 in der gebankten Version ist die absolute CP-M 2 Kompatibilität. Dies bezieht sich nicht nur auf die BDOS-Aufrufe, selbst Programme, die Bios-Aufrufe machen, haben Zugriff auf eine CP-M 2 kompatible Biossprungleiste, die in anderen gebankten Systemen nicht mehr existiert. Das heißt, daß selbst CP-M Usergroup Programme unter dem gebankten RTS-80 System ohne Änderung laufen !

In der gebankten Version erhält der Benutzer die Möglichkeit Input-Output Redirection zu machen. Dies bedeutet insbesondere bei Programmdokumentationen und dergleichen eine immense Arbeitserleichterung, da Bildschirmmasken, Menüs oder sonstige Bildschirmausgaben zugleich in eine Datei geleitet werden können, die dann von einem Editor weiter bearbeitet werden können. Programmeingaben koennen aus einer Datei kommen, dies erleichtert weiter die Benutzung.

RTS-BO in der gebankten Version wird ausgeliefert inklusive der Spooler-Despooler Option. Diese sind integraler Bestandteil des Betriebssystems und können somit mit maximaler Geschwindigkeit operieren. Sie machen vollen Gebrauch von der RTS-80 Multitasking-Fähigkeit, d. h., sie werden von dem Betriebssystem-internen Dispatcher bedient, der eine Optimierung der konkurrierenden Tasks vornimmt. Dies wiederum im Gegensatz zu anderen Betriebssystemen, wo meist nur ein Despooler verfügbar ist und dieser dem Betriebssystem 'aufgepfropft' wird, d.h. keine Betriebssystem-Resourcen nützen kann und daher das Gesamtsystem stark verlangsamt.

Desweiteren werden in der gebankten RTS-80 Version UNIX-ähnliche PIPES unterstützt, die es dem versierten Programmierer erlauben, Messages zwischen Programmen, komplizierte Batch-Jobs oder auch ganz einfach Mailboxes aufzubauen. Diese Pipes werden nach dem FIFO (first in,first out) Prinzip bedient und vom Betriebssystem verwaltet. Ein spezielles Dateiflag wird vom Betriebssystem bedient und kann von Programmen abgefragt werden.

Weitere Unix-Features sind konditional Submits(Batch), selektives Kopieren, erweiterte Datei-Flags u.s.w.

16 Mbyte Programmcodelaenge auf ZSO im der BANKED—Version von RTS—80

Mit der BANKED-Version von RTS-BO wird eine neue Methode eingefuehrt, die es auf einfachste Weise erlaubt, den Programmcode-Adressraum des Z-80 bis auf 16 MByte zu erweitern. Dies stellt eine absolute Neuheit dar, da bisherige Betriebssysteme zwar den Datenbereich erweitert haben, die Programmlaenge selbst jedoch beschraenken mussten auf etwa 60 KByte, also wenig mehr als in einem nichtgebankten CP/M-2 System zur Verfuegung stand. Unter RTS-BO steht nun ein Mechanismus zur Verfuegung, der es erlaubt, sowohl von Hochsprachen wie Pascal,C, PL-1 als auch auf Assemblerebene den Adressraum des Z-80 Prozessors gewaltig zu erweitern. Dabei wurde, wie bei allen RTS-BO Funktionen, auf absolute Kompatabilitaet zu existierenden Programmen und den CP/M 2 Normen geachtet.

Anwendung des INTERBANK-Aufrufs

Da der Interbank-Aufruf eine vollkommen neue Maechtigkeit fuer 8-Bit Prozessoren schafft, hier einige Applikationen:

Es ist durch den Interbank-Aufruf moeglich sowohl von Hochsprachen aus, als auch auf Assemblerebene, als auch mit fertigen Programmen die Z-80 Adressierfaehigkeit auf jede (sinnvolle) Groesse zu erweitern. Bei der ueblichen Art des Banking (Z80, 64180, Z800) werden bis zu 16 Mbyte mit unseren Karten, respektive 512 KByte mit 64180 und Z800, an Programm-Code unterstuetzt. Wem 16 MByte viel erscheinen mag, der bedenke, dass mit den neuen l MByte Speicherchips 4 MByte Ram auf einer Europakarte Platz haben.
Zu betonen ist , dass der Interbank-Aufruf es ermoeglicht mit den gewohnten Entwicklungswerkzeugen (Compiler, Assembler, Linker, Debugger) weiterzuarbeiten. In absoluter Kompatabilitaet zu bestehenden Programmen ,koennen auf Z-80 Programme geschrieben werden , die weit ueber 64k gross sein koennen. Dies in absolutem Gegensatz zu anderen Betriebssystemen, die zwar auch einen groesser als 64k grossen Datenbereich kennen, die Programmlaenge selbst aber in jedem Fall auf kleiner 60 K beschraenken muessen. Zudem ist der Datenbereich nur theoretisch vergroessert, denn es gibt keine Systemfunktionen, die es Anwenderprogrammen erst ermoeglichen wuerden, auf einen groeseren Datenbereich zuzugreifen.

1. Applikationsorientierte Anwendung

Es soll ein Menuegesteuertes "Instant-On" Computersystem erstellt werden, das eine Anwendungsorientierte Sammlung der am haeufigst benoetigten Programme dem Anwender auf Knopfdruck zur Verfuegung stellt.
Loesung: Es wird ein triviales Menue-Programm erstellt, das nur aus dem Text der Menue-Auswahl besteht und einer entsprechenden Anzahl von Interbank-Aufrufen. Auf den jeweiligen Baenken sind Eproms installiert, die z.B. Wordstar, Dbase, oder sonstige fertige Programme ohne jede Aenderung enthalten. Nach Verlassen des jeweiligen Programmes kehrt die Kontrolle automatisch zum Menue-Programm zurueck.

2. Hochsprachenorientierte Anwendung

Selbst aus unmodifizierten Hochsprachen heraus laesst sich der Programmcode-Bereich praktisch unbeschraenkt erweitern. Da die meisten der heute professionell genuetzten Hochsprachen-Compiler Overlays unterstuetzen, stellen auch Programme von IMBYTE Codelaenge und mehr kein Problem fuer einen Z-BO Prozessor dar. Das zu schreibende Programm wird, wie in der professionellen Programmierpraxis sowieso ueblich, in eine Main-Datei und mehrere Overlays zerlegt. Die Main-Datei kann dabei wiederum aus einem trivialen Menue . bestehen, das die Funktionen als Subprogramme aufruft. Da mit vielen der professionell eingesetzten Compiler auch ein Disassembler mitgeliefert wird, muessen nur die Subprogramm-Calls in INTERBANK-Calls geaendert werden, ein triviales Ladeprogramm hinzugefuegt werden und der Z-80 bedient Programme bis zu 16 MBYTE Laenge !

Beachten Sie, dass dies mit jedem beliebigen Compiler zu machen ist, der Overlays kann !!

Kann der Compiler dagegen keine Overlays, wird wie in 1. gearbeitet: Das Problem wird in Teilprobleme zerlegt und die einzelnen Programme getrennt kompiliert. Zur Ausfuehrungszeit residieren sie dann in verschiedenen Baenken und werden von einem Main-Modul gecallt.

3. Assemblersprachenorientierte Anwendung

Auf der Assemblerebene ist die Anwendung von INTERBANK-Aufrufen selbstverstaendlich trivial. Normale Calls brauchen nur durch Interbank-Calls ersetzt zu werden, und der Programmierer hat einen praktisch unbegrenzten Speicherplatz zur Verfuegung.

Die Assemblierung, das Linken und Austesten kann mit den gewohnten Entwicklungswerkzeugen erfolgen l!

Eine Anmerkung am Rande. Wer von den bekannten Betriebssystemen herkommt wird vielleicht bei l Mbyte Code leichte Bedenken in Punkto Wartezeiten haben. Hier kommen jedoch gerade die vielen diskorientierten Geschwindigkeitsoptimierungen unter RTS-80 zum Tragen. Sie werden feststellen, dass der "gute alte Z-BO" selbst mit 6 MHZ die meisten der heute verfuegbaren 16 Bit-Prozessoren "in den Schatten" stellt. Unsere Systeme werden dagegen optional auch mit 8 MHZ geliefert.

Die Erklaerung hierfuer ist leicht zu finden. Eine nur oberflaechliche Analyse, was an Computern hauptsaechlich gemacht wird, zeigt , dass 90 ‘/. der Aktivitaeten Textverarbeitung ist. Selbst Compilierung, Assemblierung, Datenbanken sind letzten Endes nur Textverarbeitung. Und Textverarbeitung ist nun mal die Domaene der B-Bit Prozessoren, worin sie nicht zu schlagen sind. Die wenigen technisch-wissenschaftlichen Applikationen wo 16, 32 und hoehere Bitzahlen gefordert sind, koennen an einer Hand abgezaehlt werden. Hinzu kommt, dass der Z-BO Prozessor von Hause aus natuerlich nicht den Code-Overhaed einer linearen Adressierung von 16 Mbyte mit sich traegt. Die Code-Laenge hat dadurch im Maximalfall auch nur ein Viertel der Laenge seiner "groesseren Brueder" und dadurch wiederum naturgegeben eine sehr viel geringere Run-Time.

Fuer den interessierten hier die genaue Implementation des Interbank-Aufrufs:

In der BANKED-Version von RTS-80 gibt es neben den von CP/M her bekannten Systemeinspruengen auf Adresse 0 (Warmstart) und Adresse 5 (Bdos-Aufruf) einen weiteren Systemeinsprung auf Adresse ODH = 13 dezimal.

Der Systemeinsprung auf Adresse ODH wird bezeichnet als INTERBANK-Aufruf . Er dient dazu einen vom Betriebssystem koordinierten Aufruf in anderen, als der derzeit eingeschalteten, BANKs zu ermoeglichen. Es lassen sich damit Programme und Unterprogramme in anderen BANKs starten. Wurden sie mit einem Call aufgerufen, so kehrt die Kontrolle nach Abarbeitung zu dem Aufrufenden Programm zurueck. Wurden sie mit einem Jump angesprungen, bleibt die dabei gewaehlte BANK eingeschaltet. Es kann dann selbstverstaendlich mit einem weiteren INTERBANKsprung in eine andere BANK gesprungen werden, oder ein Programm in einer anderen BANK 'gecalled' werden. Dieser Mechanismus ermoeglicht die Verwendung von Programmen (!) die groesser 64k sind, ja bis zur maximalen Ausbaustufe des Systems, 512k resp. 16Mbyte reichen koennen. Dies steht im Gegensatz zu anderen Z-80 Betriebssystemen, die die Programmlaenge auf 60K im Maximalfall begrenzen. Der Datenbereich ist selbstverstaendlich ebenso nur vom Speicherausbau begrenzt.

Nutzung des INTERBANK-Aufrufs:

Der Interbankaufruf ist fuer einen Assemblerprogrammierer ebenso einfach zu handhaben, wie ein normaler Call oder Jump:

1.Call

```
      ld   hl,STARTADRESSE  ;Startadresse in neuer Bank vorgeben
      ld   a,ZIELBANK       ;Zielbank vorgeben
      call INTERBANK        ; (Sub-)Programm callen
                            ;Programm kehrt aus Zielbank hierher
                            ;zurueck
```

Hierbei ist als einziges zu beachten, was eigentlich selbstverstaendlich sein sollte, dass das Subprogramm entweder den vorgegeben Stack beibehaelt, oder aber bei intensiver Stacknutzung in guter alter CP/M Konvention den Stack bei Eintritt konserviert und bei Austritt restauriert.
Dies ist uebrigens bei fast allen professionellen CP/M Programmen der Fall, so dass sie ohne Aenderung zu nutzen sind.

2.Jump

```
      ld   hl,STARTADRESSE  ;Startadresse in neuer Bank vorgeben
      ld   a,ZIELBANK       ;Zielbank vorgeben
      jp   INTERBANK        ; (Sub-) Programm aufrufen

                            ;Programm kehrt aus Zielbank nicht
                            ;zurueck,     dort     kann     weiterer
                            ;Interbank-Sprung stehen
```

Es stehen selbstverstaendlich auch alle Mechanismen zur Verfuegung Baenke zu validitieren, vorzuwaehlen fuer Disk-I/O und Programm-Load.

INSTALLATION VON RTS-BO:

RTS-BO in der Banked-Version wird für unsere Karten-Systeme fertig implementiert geliefert. Dabei wird ein Adressraum von maximal 16 Mbyte unterstuetzt.

Für Fremdsysteme ist ein einfaches CP/M-2 ähnliches Bios erforderlich, das je nach gewünschtem Komfort ausgebaut werden kann. Als Grundfunktionen sind nur die normalen CP/M-2 Bios-Aufrufe erforderlich, die um eine zusätzliche kurze Bank-Umschaltungsroutine und Multisektor-Operationen ergänzt werden sollten. Hierzu werden ganz einfach die bestehenden READ- und WRITE-Routinen benutzt, wobei ein Zaehler heruntergezaehlt wird. Als Beispiel die einfache Implementation:

```
READRTSBO:
          ld   a, (SECTORCOUNT)      ;soviele sind zu lesen
          ld   b,a
readloop:
          push bc
          call READ1SEC              ;CP/M READ SECTOR AUFRUFEN
          pop  bc
          djnz readloop              ;NOCH WAS ZU TUN ?
          ret                        ;IN A ERROR-CODE
```

Alle weiteren Bios-Aufrufe sind optional (Uhrenfunktion, Multi-serielle Kanäle mit Modem-Control usw)

Es wird ein Beispiel-Source-Listing mitgeliefert, das oft ohne Änderungen übernommen werden kann.