Historical Look At

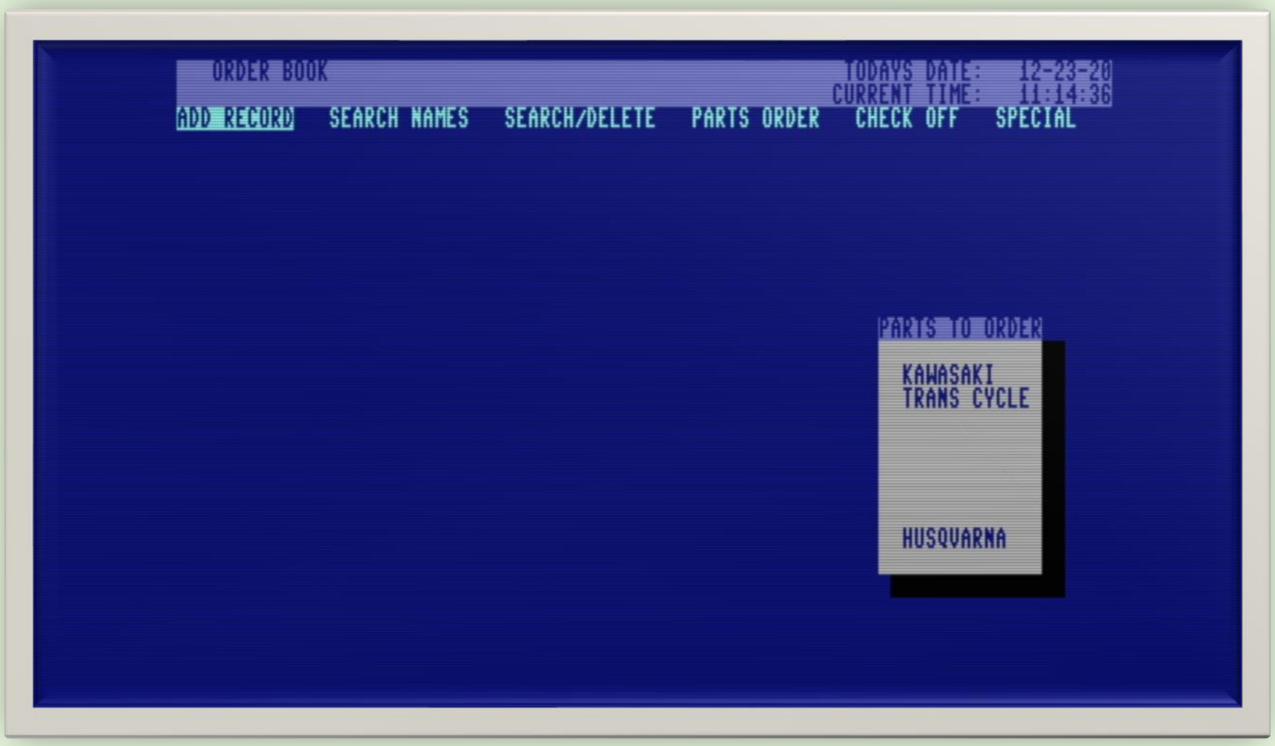
ORDERBOOK 128

by Gord L. Clink

written in

commodore BASIC 7.0

1987, 1988


```
ORDER BOOK                                   TODAYS DATE:    12-23-20
                                           CURRENT TIME:    11:14:36
ADD RECORD   SEARCH NAMES   SEARCH/DELETE   PARTS ORDER   CHECK OFF   SPECIAL

                                            PARTS TO ORDER

                                             KAWASAKI
                                             TRANS CYCLE


                                             HUSQVARNA
```

"This is the story of a piece of custom business software that was written for the Commodore 128 computer back in 1987,88".

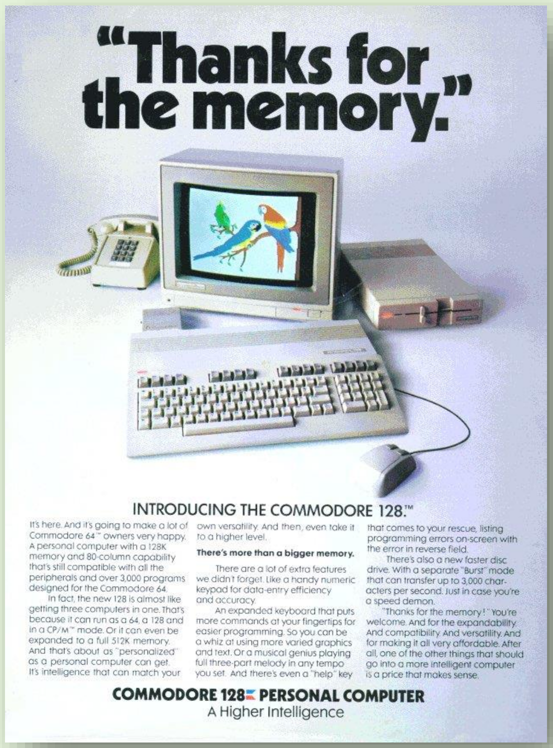
Way back in 1985 I was a 16-year-old computer geek who was also into cars, motorcycles, and snowmobiles. Most of my youth, I aspired to be an auto-mechanic. So, it was only natural that as soon as I could legally drive, I gained employment at a local automotive shop (which was also a Kawasaki and Arctic Cat dealer). Here I started pumping gas and slowly worked my way into doing maintenance on small engines, oil changes on automobiles etc. Eventually, I signed up for an auto-mechanic apprenticeship, as that was my future…so I thought!

My interest in computers started in 1980 at age 11, when I saw my Uncles Commodore PET 4016. I was absolutely blown away with the things you could do with it. Being able to tell a machine to do your bidding was intoxicating. Around late 1982, at the age of 13, I bought my first computer which was a VIC-20 that I bought with my own money from our local hardware retail chain 'Canadian Tire'. My dad was not impressed that I wanted a computer. He told me 'I should be focusing my attention on something that I can make a career out of'. Many years later, when my career was in the 'computer networking' field, he had to admit he was wrong.

It took 13 attempts before I got a working VIC-20. Number 12 even had smoke pouring out of the keyboard. Obviously, Commodores quality control wasn't the best at that time. Needless to say, my dad was not impressed and told me *"if the next one didn't work, I wouldn't be getting a computer."*

Thankfully lucky number 13 worked, and I went on to have great fun with the VIC-20, playing cartridge games like Gorf and Clowns, and typing in all kinds of programs from various magazines. Typing in programs from magazines, as it turns out, is a great way to learn how to program. Your learning, and don't even realize it. I also discovered 'Sargon II Chess' on the VIC, which started my interest in artificial intelligence. But that is a story for another time.

I never owned a Commodore 64 (The best-selling computer in history), although my cousin owned one and I thought it was amazing. I thought my cousin was some sort of wizard, because he knew how to code in Machine Language on his 64.



INTRODUCING THE COMMODORE 128.™

It's here. And it's going to make a lot of Commodore 64™ owners very happy. A personal computer with a 128K memory and 80-column capability that's still compatible with all the peripherals and over 3,000 programs designed for the Commodore 64.

In fact, the new 128 is almost like getting three computers in one. That's because it can run as a 64, a 128 and in a CP/M™ mode. Or it can even be expanded to a full 512K memory. And that's about as "personalized" as a personal computer can get. It's intelligence that can match your own versatility. And then, even take it to a higher level.

There's more than a bigger memory.

There are a lot of extra features we didn't forget. Like a handy numeric keypad for data-entry efficiency and accuracy.

An expanded keyboard that puts more commands at your fingertips for easier programming. So you can be a whiz at using more varied graphics and text. Or a musical genius playing full three-part melody in any tempo you set. And there's even a "help" key that comes to your rescue, listing programming errors on-screen with the error in reverse field.

There's also a new faster disc drive. With a separate "Burst" mode that can transfer up to 3,000 characters per second. Just in case you're a speed demon.

"Thanks for the memory!" You're welcome. And for the expandability. And compatibility. And versatility. And for making it all very affordable. After all, one of the other things that should go into a more intelligent computer is a price that makes sense.

COMMODORE 128™ PERSONAL COMPUTER
A Higher Intelligence

I continued using my VIC-20, and then in late 1984 I started hearing and reading about this computer from Commodore called the 128. I remember staring at the advertisement showing the 128 with parrots on the screen, and wished I could afford one. It's funny how this advertisement has a double meaning now. Shortly after the Commodore 128 was made available (June 1985), Commodore started putting effort into getting more computers into the education system in Canada, and had a promotion in place for teachers. The promotion consisted of a Commodore 128 computer, Commodore 1571 Disk Drive, and a Commodore 1902 Monitor, at a significantly discounted price. Lucky for me, another one of my cousins (much older) who was a teacher, had already purchased a new computer, so he didn't require a new one and offered to purchase the 128 at the discounted price for me, of which I readily accepted. If I remember correctly teachers could get the Commodore 128 computer, 1902 monitor and 1571 disk drive for about $1100.00 as apposed to the retail price of over $1500.00 Canadian. Although that was a lot of money for me at the time, my grandfather had just recently passed away, and I was given a small amount of money from his estate, and that is what I used to buy my 128. (I always think of my Grandfather when pondering about my life in the computer field and how he had a hand in it). I remember patiently waiting for the computer to be shipped. If my recollection is correct, it was late 1985 that I finally received it. When my mother found out what my cousin did for me, she was livid. She felt the two of us were being dis-honest and ripped off Commodore because I wasn't a teacher. I tried to explain that "my cousin WAS a teacher, and what he did with the computer after he buys it was none of their business. As well, I'm sure they would welcome any sale". Apparently, my mom marched over to my cousins and tore a strip off him. I'm not sure what was said to my mom, but she eventually calmed down about it.

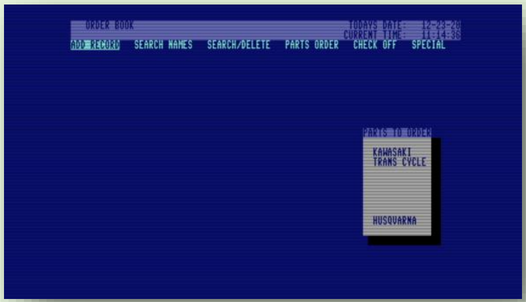
Wow, did I love that computer. It was so sleek and modern looking. I started really getting hooked on computers at this point, and started doing a bit of my own programming in BASIC. Living in North Western Ontario at the time, there was virtually no resources to speak of when it came to learning computers. Everything I learned was from magazines and manuals. I would read my monthly magazine subscriptions to **RUN Magazine** and **Compute's Gazette** cover to cover, multiple times. I just couldn't get enough information.

My new job was going well, as I was doing what I loved, working with small engines, and being around cars, motorcycles and snowmobiles. But at night I was getting more and more into programming. Finally, I realized that I would rather be a programmer than a mechanic. It was just as gratifying in my opinion, and a bonus that you didn't get your hands greasy. Over the next two years I did a lot on my 128, and never ever had an issue with it. I guess the quality control was improved from the VIC-20 days.

The garage I worked at was just beginning to jump onto the computer bandwagon. They had purchased a Tandy 1000 to do their accounting on, but other than accounting, they didn't use computers for anything else. I remember my boss asking me if he should get the 10MB hard drive or the 20MB. I told him 10MB was plenty. I think the accounting system was called Bedford, if I'm not mistaken. Around 1987 I started working full time and had been promoted to parts manager, and was in charge of organizing the parts room, and ordering stock and customers parts. We used a note book for writing down the parts that needed ordering, along with the name of the customer or job. At the end of each week, we would put an order in with the various suppliers. There were definite problems with this system. Sometimes you had trouble reading peoples writing. Sometimes things would get forgotten, and it was hard to figure out if something had been ordered or not, when employees would forget to stroke it off the sheet after it was ordered. You can imagine the problems.

Order Book 128

Shortly after becoming the parts manager (sometime in 1987), I decided to approach my boss about the idea of writing a **"parts ordering management system"** on my **Commodore 128** Computer. By this time, I had already upgraded to an Amiga 1000, so I wouldn't miss my 128. I would work on the code at times when I was working evenings, in-between customers. Surprisingly my boss agreed! Of course I was elated, but now when I look back, I think "Man, he must have had a lot of faith in me, letting an 18-year-old play on a computer at work and get paid at the same time". My boss ended up buying the computer from me in order to use the software I wrote, and I guess in a way, since I was getting paid while writing the software on my shifts, he bought the software too!



Although my programming knowledge was completely self-taught at that time, I was very fluent in Commodore BASIC 7.0, as I had been playing and fiddling with my 128 for a couple years by that time, and my VIC-20 before that. I really connected well with BASIC and found that it seemed to be the way my brain worked, so I could really make it do what ever I wanted. Structured programming wasn't something that I knew a lot about at the time. As a result, my code was not very well documented (well to be honest, it wasn't documented at all). Although I did uses subroutines as much as possible, there are still a few GOTO's that probably could have been eliminated by better programming, but I was able to accomplish coding a very intuitive, menu driven, parts ordering database application, that was used by the garage for all their parts ordering for almost 5 years. I do recall making the decision not to document the code with too many REM statements because I didn't want to use up too much presious memory that I may need for actual code. Of course, if I was to do it over again, I would definitely document it better.

The program went through 4 versions, from 1.0 to 1.3. There was a version 1.4, but it was never finished. I still have one early version (I don't know if it was 1.0 or 1.1 as it was not documented). I also have Version 1.2 and 1.3, which 1.3 was the last version in use until approximately 1992.

The main program code for version 1.3 is 175 blocks on disk (44,259 Bytes in memory) and has 1027 BASIC lines of code. At the time, this was the largest program I had ever written. It was made up of at least 19 subroutines.

One interesting story to note is that I almost landed a programming job because of 'Order Book 128'. On one of my days off, an individual came by the shop for fuel. and when he came in to pay for his purchase, he noticed my program running on the computer behind the counter. He asked the attendant "who created the software that was running on that computer?". He was told "Gord Clink… he works here". So, the man left his card and asked the attendant to have me call him. The next day I came into work, and was given the card. I called and it was a fellow from a computer company in a nearby city. When I called the number, he introduced himself and told me how impressed he was with the system that I had developed, and wanted me to come for an interview, as they were looking for programmers. A few days later, I drove the 3 ½ hours and arrived for my interview. The interview went well, and they told me that they wanted to hire me and that they would be sending me out information with all the details and a start date. About two weeks later I was informed that the company had gone bankrupt and closed its doors. The guy that originally talked to me told me he didn't see it coming, and had no idea. I'm assuming it was a parent company that most likely shut them down, or maybe the employees simply were not aware of the trouble the company was in. Regardless, it makes for a good story, and I often wonder what would have happened had I actually got the job. Would my career be in computer programming now instead of computer networking? Who knows!

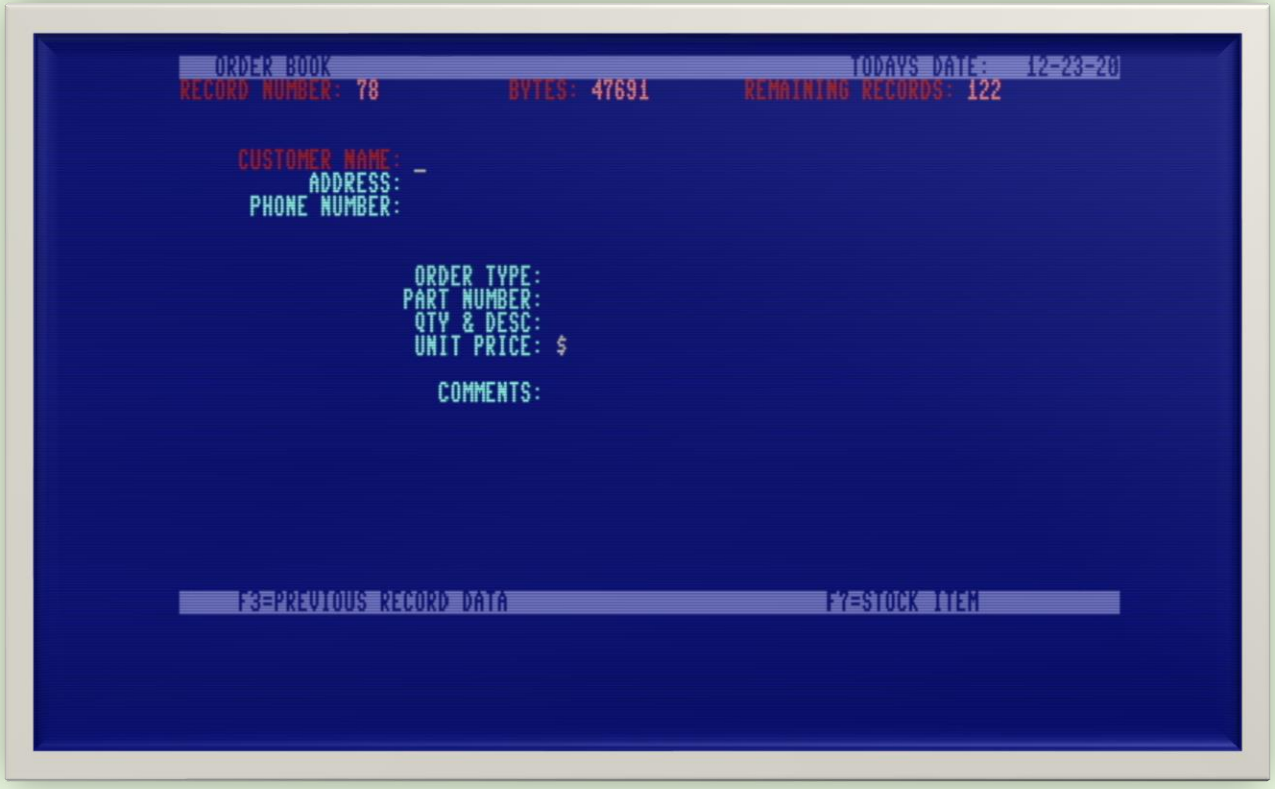
Now in 2020, while I sit in my house on lockdown for COVID, I decided it would be interesting for me to go through and document this system for myself, just so I have a better understanding of what I actually did. When you read code that you wrote 33 years ago, it really seems like you are looking at someone else's code. I see things I did that I'm not sure what they do, and of course with no documentation, it's quite interesting.

It then occurred to me that this might be of interest to others who love their old Commodore computers as much as I do. The Commodore 128 received a lot of negativity as being 'too little too late' (since the Amiga was released the same year) and many think the 128 was only used for its 64 mode to play games, and was never seriously used much in 80-column mode. I am here to tell you that, in my case, that is not entirely true, as my 128 was used for real business for almost 5 years. It was used in 128 Mode in 80 columns, and ran in FAST 2Mhz mode.

“Order Book 128” Features:

Record Limit

Number of records available was hard coded to a limit of 200, but that could be increased easily by changing a few variables, and was only really limited by the amount of memory remaining in the 64KB of dedicated ‘data RAM’. When adding a new record, the screen would show you the current record number, data memory remaining, and number of remaining records available based on what the limit was set to. A realistic upper limit would have been about 400 records if you consider each record was about 150 Bytes in length. This was way more than was required, as very rarely did the number of records get over 100, because once the customer picked up the part they ordered, the record was deleted.



```
  ORDER BOOK                                   TODAYS DATE:   12-23-20
RECORD NUMBER: 78          BYTES: 47691       REMAINING RECORDS: 122

     CUSTOMER NAME: _
           ADDRESS:
      PHONE NUMBER:

                       ORDER TYPE:
                      PART NUMBER:
                       QTY & DESC:
                       UNIT PRICE: $

                         COMMENTS:

     F3=PREVIOUS RECORD DATA                    F7=STOCK ITEM
```

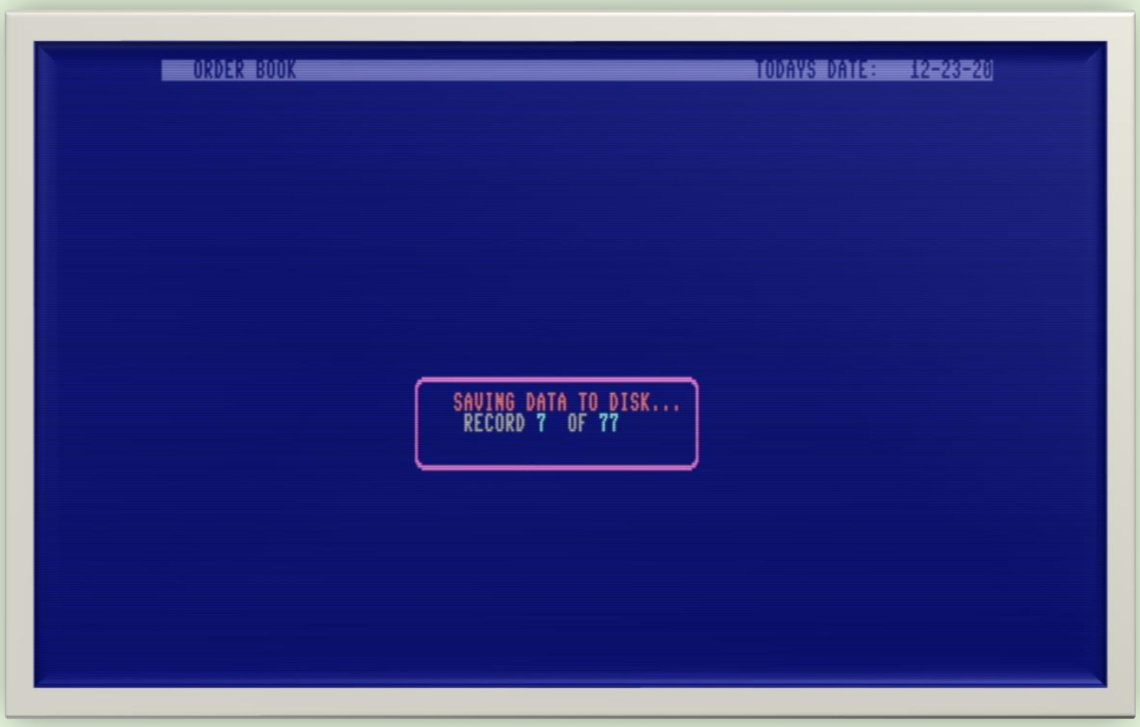
Sequential Data Files

Data was stored in 3 sequential files which were loaded into memory when the program was first run. The first one was called "RECORD NUMBER" and only contained the number of records that are contained in a second file that was named in the following format: "DTmmddyyhh:mm:ss". The file name is DT for data and the date and time that it was saved. So, a typical filename for data that was saved on **March 21, 1989 at 3:32:45** in the afternoon would be "DT03218915:32:45". The Commodore 1571 drive doesn’t have the ability to save the date and time of file writes, so I had to created my own method by embedding it in the filename. This was important information to know in the case of a power failure or disaster of some type, you would at least be able to tell how current your data is by looking at the filename. The records were loaded in one at a time, which took about 30 seconds.

A third file "PHONE NUMBERS" contained a phone directory that pops up when typing 'CMDR P'. This was of course editable from with the OrderBook128 application, and was automatically resaved to this file if you edited an entry in this list.

Why not relative files?

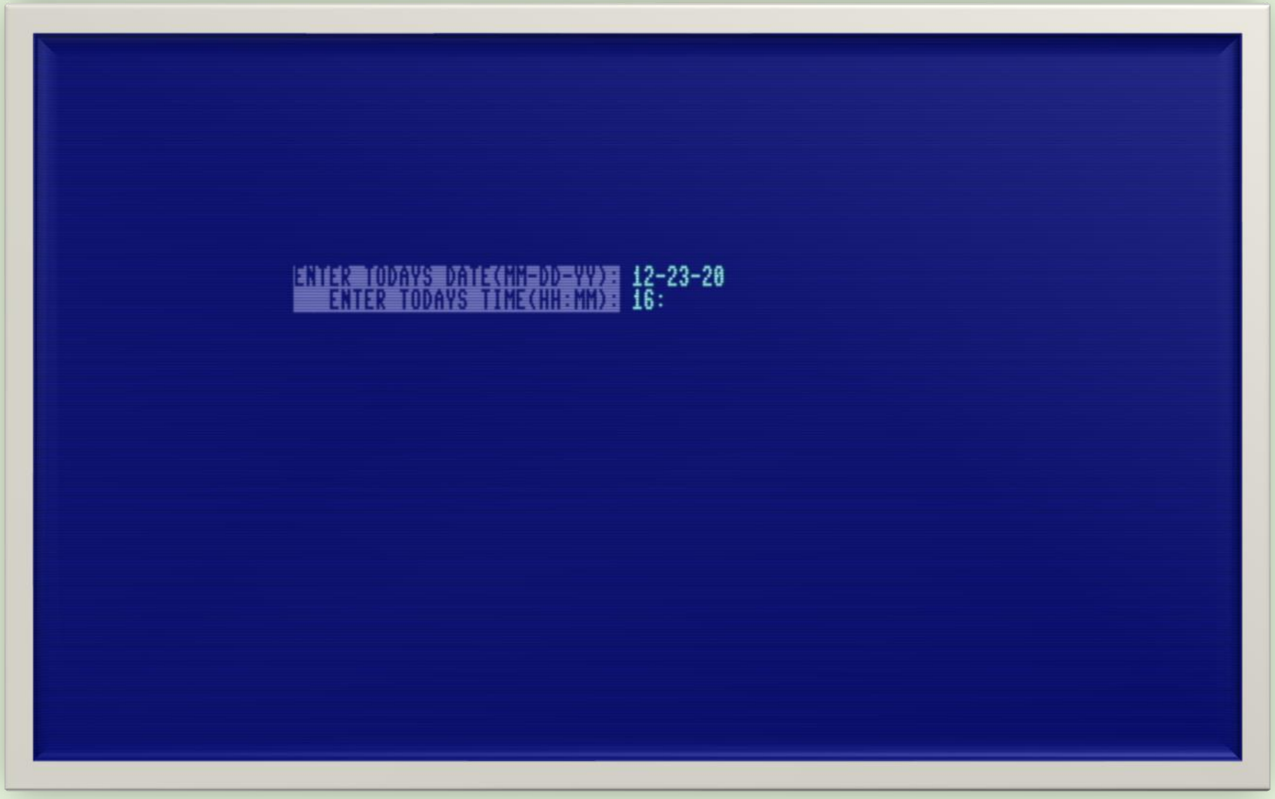
I remember initially wanting to use Commodores REL files, which would have made more sense, but I couldn't seem to get them working. This was probably because I didn't understand them completely, and for the sake of efficiency in getting the project done, I used SEQ files. The only downfall to sequential files was the fact that if data was entered, and not saved immediately, there was the potential to lose the data that was entered if there was a power failure. Because of this I programmed in the ability to save the data at any time by typing 'CMDR S' at the main menu. As well, I programmed an *autosave* feature that would save the data automatically to the working disk at noon and midnight each day. It would then put a prompt on the screen to insert a Backup Disk. The first person to use the system would see the prompt, pop in a Backup Disk, at which point another save would be performed, and then it would prompt you to put the working disk back into the drive. This worked quite well, and I never had a disaster in the almost 5 years it was in use. In fact, as far as I can remember, the original disk never ever failed. That's pretty incredible for being in a dirty garage environment.



```
ORDER BOOK                                    TODAYS DATE:   12-23-20

                    SAVING DATA TO DISK...
                     RECORD 7  OF 77
```

Date & Time

The system kept track of the date and time for each record (date entered and date ordered). Because the Commodore 128 didn't have a battery backed TOD clock, you had to enter the current date and time manually upon program startup. For this reason, the computer was generally just left on 24/7 to reduce the number of times you had to enter the date into the system.



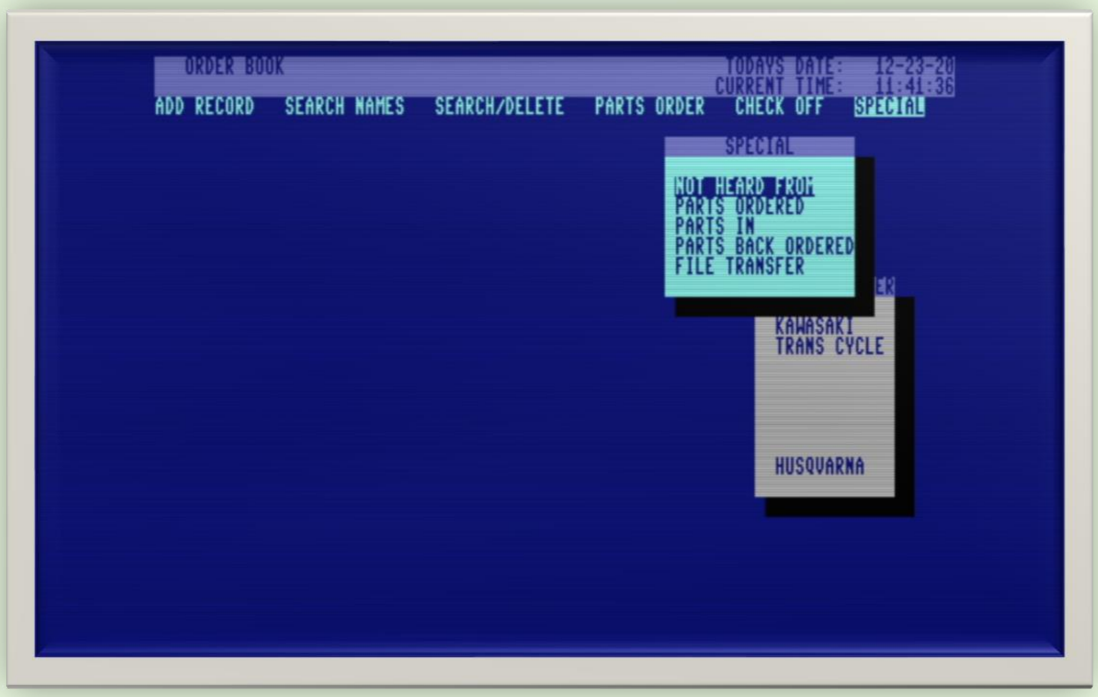
```
ENTER TODAYS DATE(MM-DD-YY): 12-23-20
   ENTER TODAYS TIME(HH:MM): 16:
```

As well, the 128 didn't have any built-in function for keeping track of the date (only time), so the date function was completely coded from scratch and stored in the variable TD$. It worked well with the exception that it didn't' account for leap years. To deal with this, I simply programmed in the ability to advance or retard the date one day at a time by pressing 'CMDR +' and 'CMDR –'. There were leap years in both 1988 and 1992, so I assume this feature was used twice.

Over time, I noticed that the 128's clock was not the most accurate, and time would drift by a few minutes over the period of a few days or weeks of operation. So, I put in the option to add or subtract minutes from the time by pressing 'shift +' or 'shift -' from the main menu. This made it really easy to adjust the time when needed.

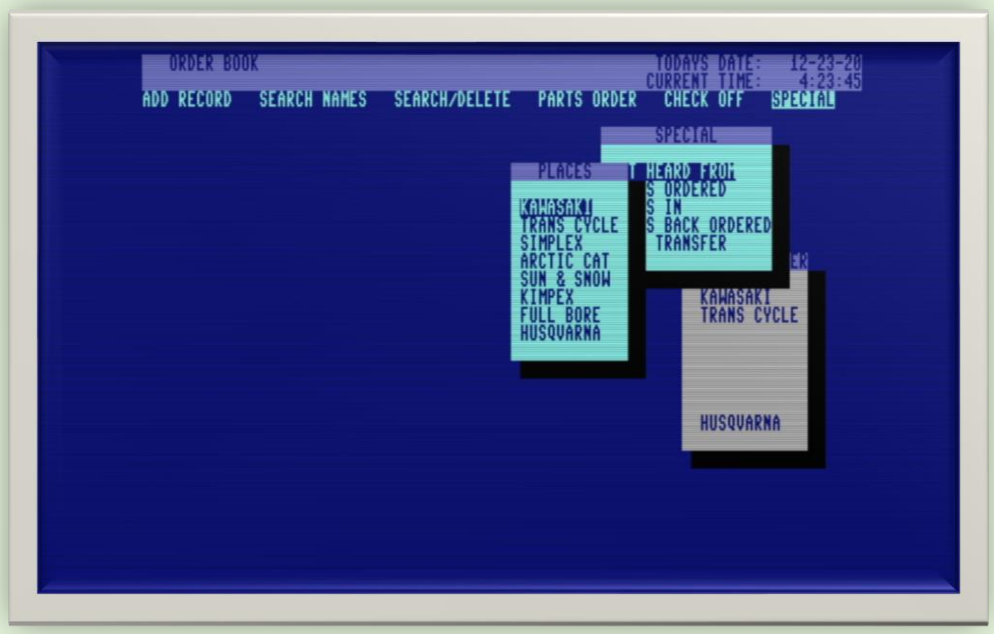
Drop Menus and Shadows

I had to create the menu feature from scratch, as BASIC 7.0 didn't have any built-in menu functions. I also had special areas on the screen for special information which was updated in real time. For example: on the main screen there is a window that is always updated with the name of any suppliers that have parts waiting to be ordered. This was very convenient as an instant reminder so orders were not forgotten. Both drop down menus and any other information areas that were open on the main screen had a nice shadow effect as well, which made it a little more pleasing to the eyes.

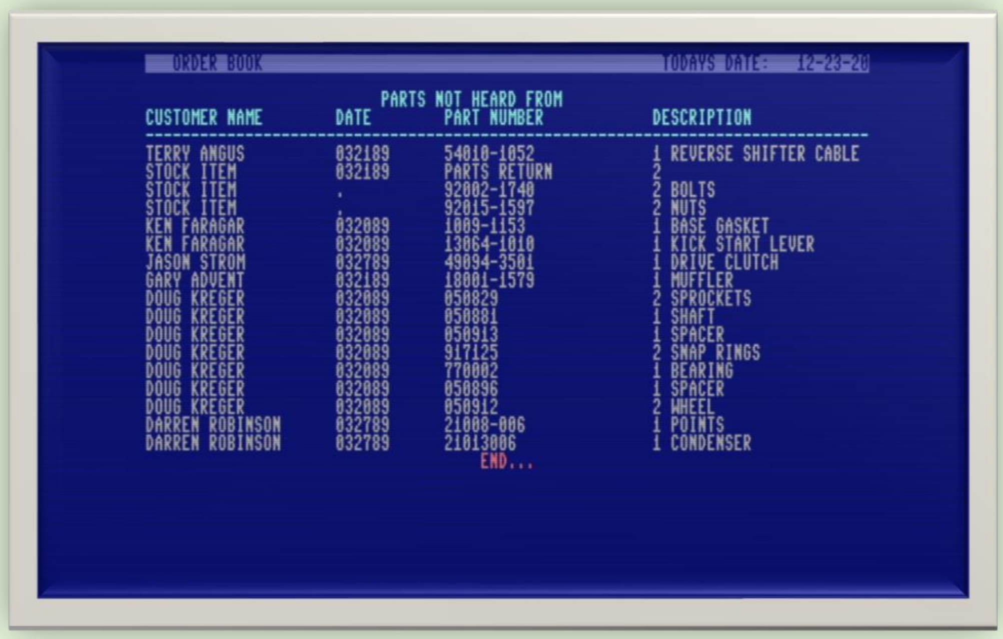


Reports

There were several reports available located under the *'SPECIAL'* menu. I'm not sure why I didn't call it *'REPORTS'* but there four basic reports that you could direct to the screen, or the printer. *'NOT HEARD FROM'* printed out a listing of all parts that were ordered, but haven't arrived yet. *'PARTS ORDERED'* was a listing of all parts that were ordered on the current day. This was handy, because sometimes you might want to know if anyone ordered any parts yet today, and you could see a quick listing. *'PARTS IN'* generated a list of all parts that were in, and not picked up by the customer yet. And the final report was a listing of all parts that were backordered from the vendor.



```
   ORDER BOOK                                          TODAYS DATE:   12-23-20
                                                      CURRENT TIME:    4:23:45
ADD RECORD   SEARCH NAMES   SEARCH/DELETE   PARTS ORDER   CHECK OFF   SPECIAL
```

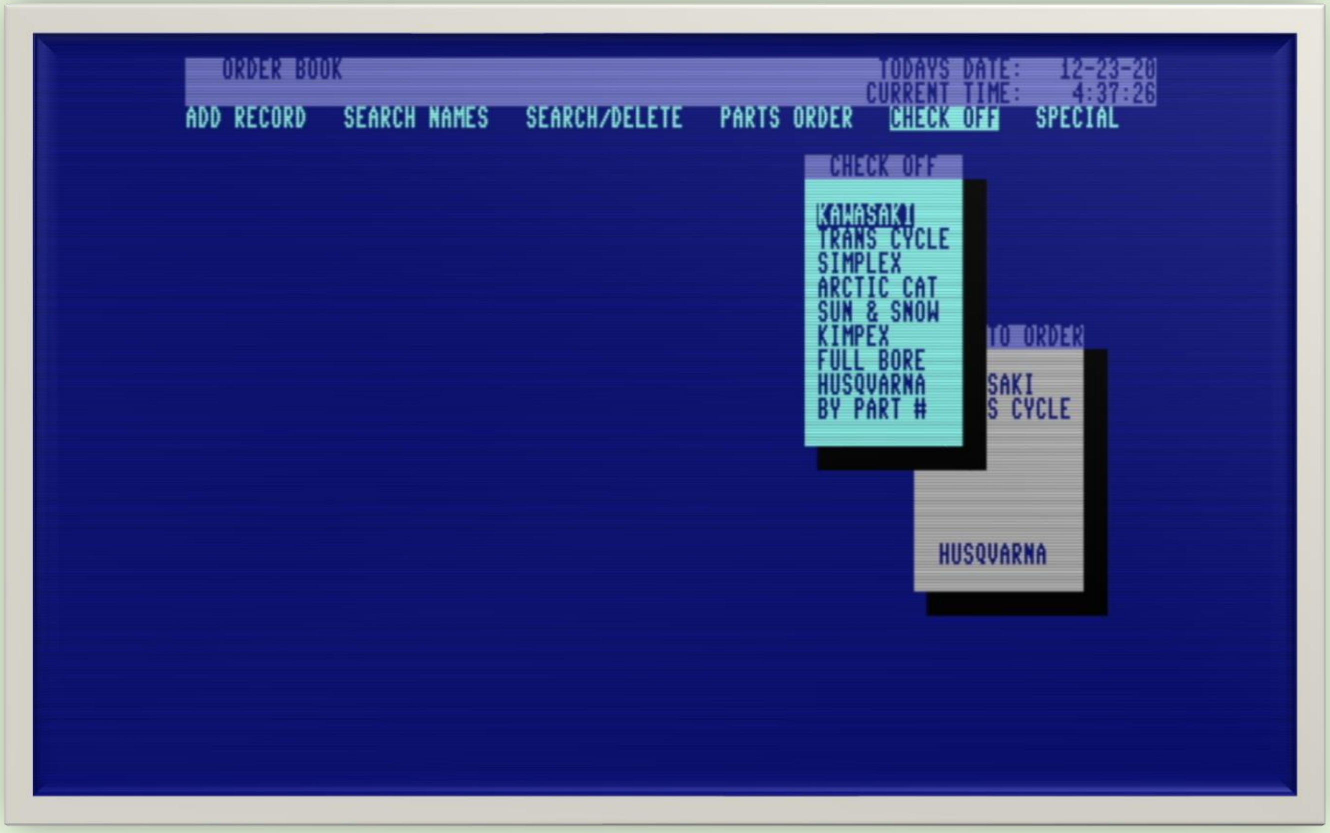


```
   ORDER BOOK                                               TODAYS DATE:   12-23-20

                               PARTS NOT HEARD FROM
CUSTOMER NAME          DATE        PART NUMBER            DESCRIPTION
-------------------------------------------------------------------------------
TERRY ANGUS            032189      54010-1052             1 REVERSE SHIFTER CABLE
STOCK ITEM             032189      PARTS RETURN           2
STOCK ITEM             .           92002-1740             2 BOLTS
STOCK ITEM             .           92015-1597             2 NUTS
KEN FARAGAR            032089      1009-1153              1 BASE GASKET
KEN FARAGAR            032089      13064-1010             1 KICK START LEVER
JASON STROM            032789      49094-3501             1 DRIVE CLUTCH
GARY ADVENT            032189      18001-1579             1 MUFFLER
DOUG KREGER            032089      050829                 2 SPROCKETS
DOUG KREGER            032089      050881                 1 SHAFT
DOUG KREGER            032089      050913                 1 SPACER
DOUG KREGER            032089      917125                 2 SNAP RINGS
DOUG KREGER            032089      770002                 1 BEARING
DOUG KREGER            032089      050896                 1 SPACER
DOUG KREGER            032089      050912                 2 WHEEL
DARREN ROBINSON        032789      21008-006              1 POINTS
DARREN ROBINSON        032789      21013006               1 CONDENSER
                                      END...
```

CHECKing OFF Orders

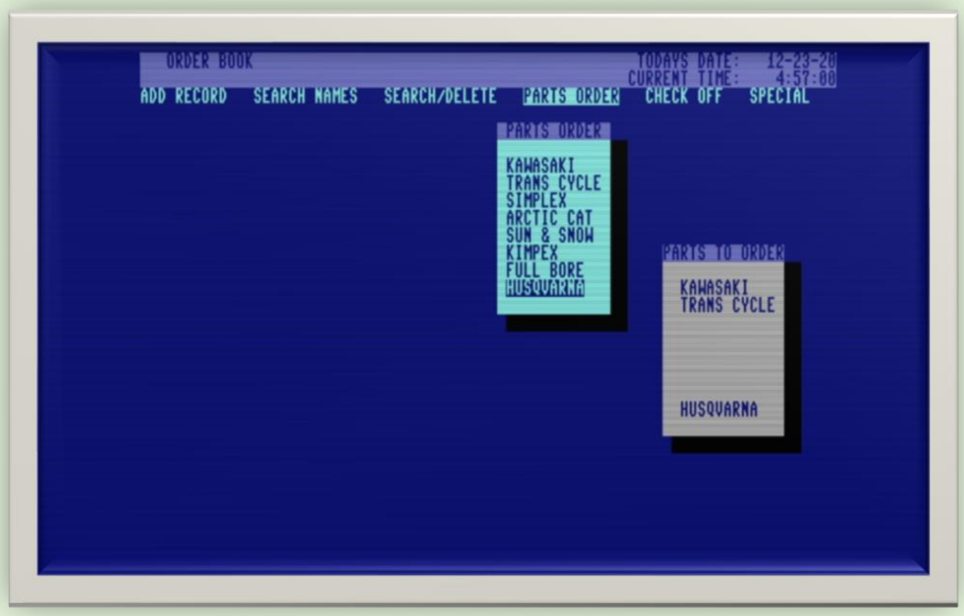
The '*CHECK OFF'* menu allowed you to check off order against a shipment that was received. You would take the packing list, and "check it off" against the Order Book. You would start by picking the vendor, and then you would enter a packing slip number, and the order date. The system would look up that order from that date, and then allow you to tell the database one record at a time if that item has been received in (I) or back ordered (B) or not heard from (N). Once this was done, the system would give you the option of printing labels for all of the parts that arrived in the order. These labels would have the customer name, part number, description, and amount owing.

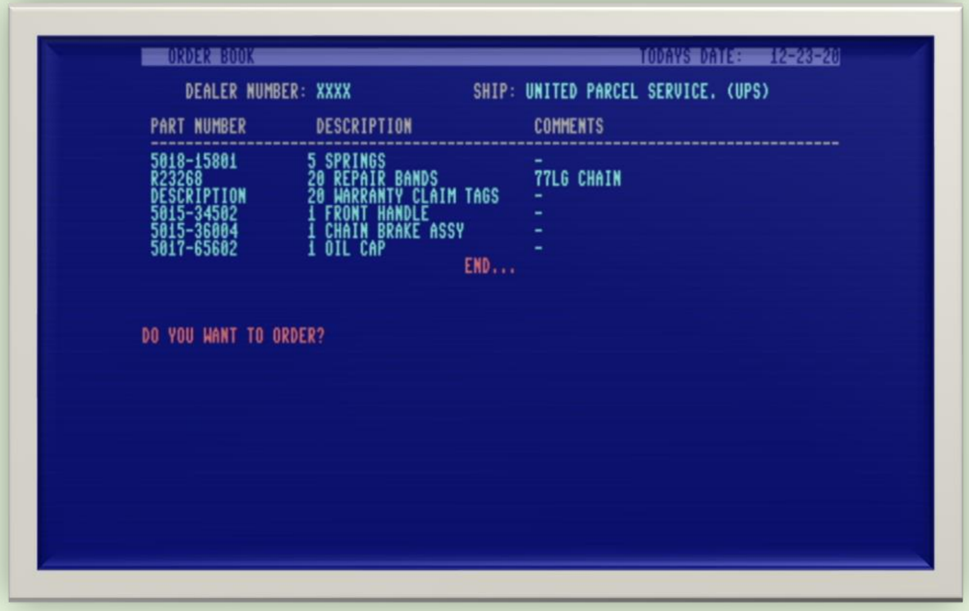
There was also an option to check off by part number. This was required in the case that a part was back ordered, and then arrived later by itself.



Parts Ordering

When parts were ordered from a vendor, you would use the *'PARTS ORDER'* menu. It allowed you to make an order to a specific vendor. You would make the order by phone, reading off the part numbers off the screen, then you would type 'Y' to the question 'Do you want to order?' and the system would mark the records from that order as ordered, and put an order date stamp on each record.





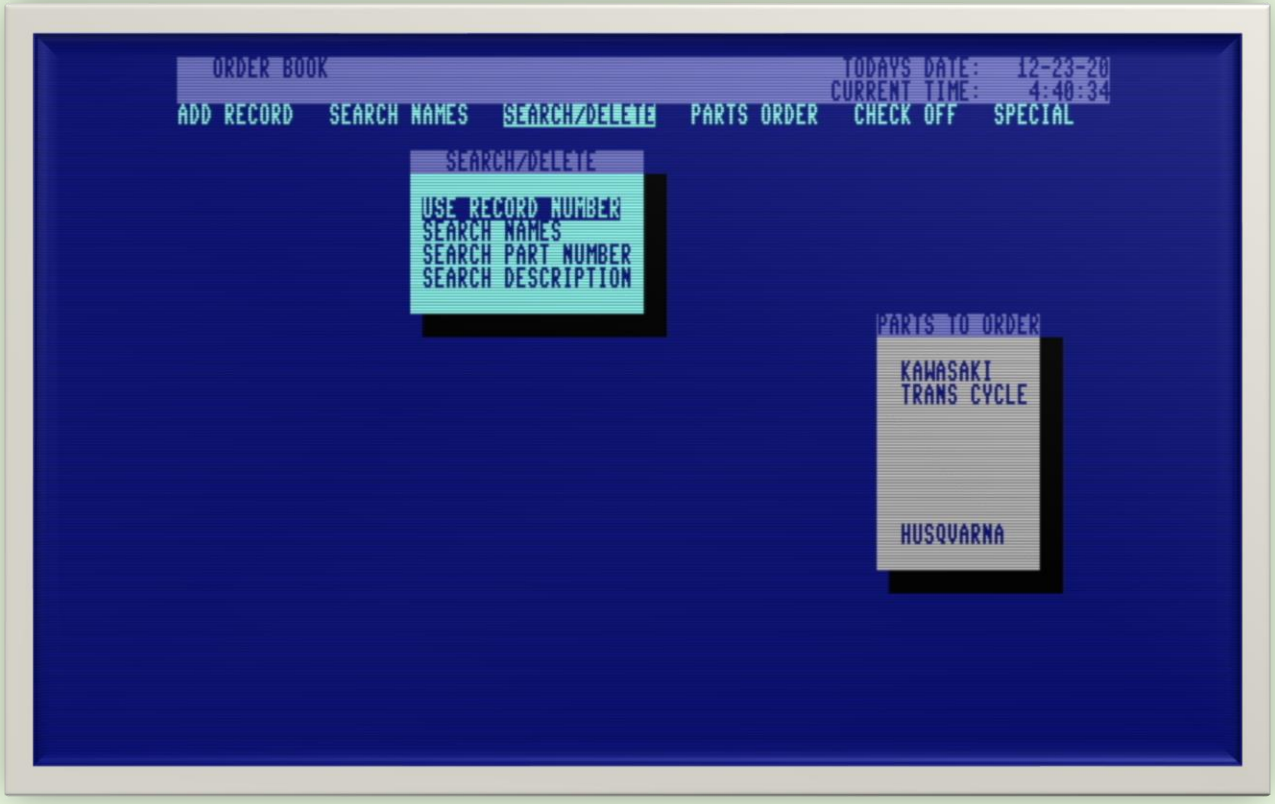
```
  ORDER BOOK                                        TODAYS DATE:   12-23-20
     DEALER NUMBER: XXXX              SHIP: UNITED PARCEL SERVICE. (UPS)

 PART NUMBER        DESCRIPTION              COMMENTS
 ---------------------------------------------------------------------------
 5018-15801        5 SPRINGS                 -
 R23268            20 REPAIR BANDS           77LG CHAIN
 DESCRIPTION       20 WARRANTY CLAIM TAGS    -
 5015-34502        1 FRONT HANDLE            -
 5015-36004        1 CHAIN BRAKE ASSY        -
 5017-65602        1 OIL CAP                 -
                                       END...

DO YOU WANT TO ORDER?
```

Searching

Orderbook 128 had some pretty good searching abilities. You could search by **Record Number**, which was handy if you were looking at a printout, and wanted to view or delete a specific record.

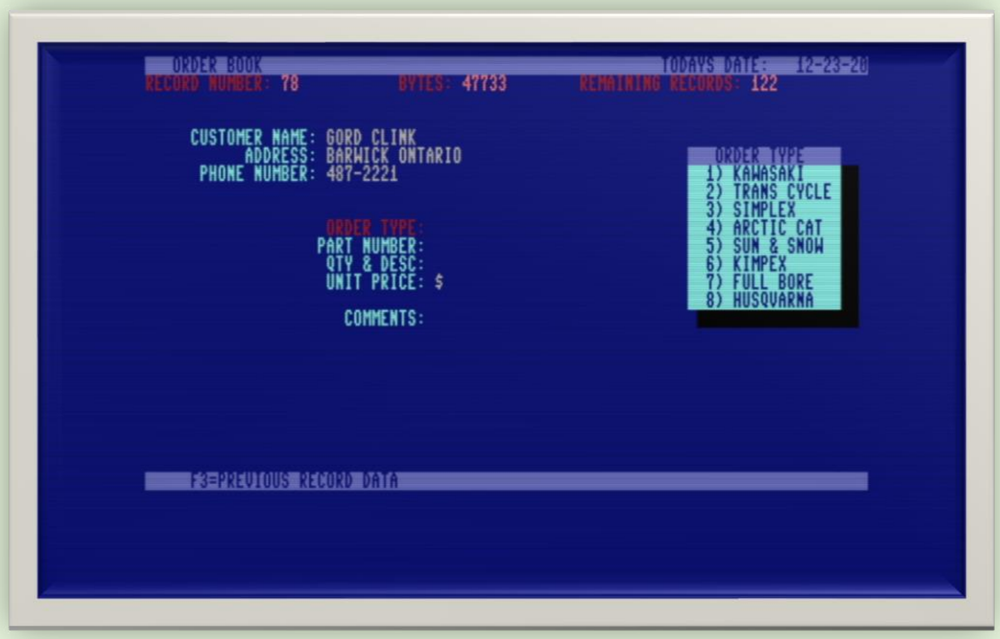
You could also search by **Names**, which was under the *'SEARCH/DELETE'* menu as well as its own *'SEARCH NAMES'* menu item. This was done because this is the type of search that was used most often. *(When a customer would come in looking for their part or parts, you would simply search for their name, and it would bring up all of their parts one at a time telling you if the part was in or not. You could hit any key to advance to the next record under their name.)*

As well as Record Number, and search by Names, you could also search by **Part Number** or **Description.**

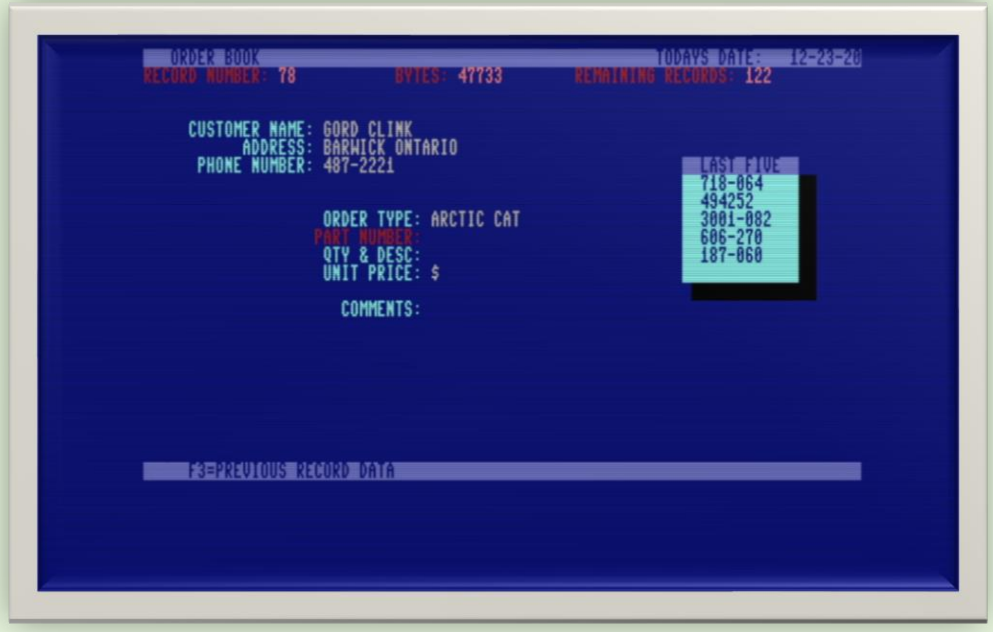


Adding Records

When a part needs to be put in the system, you use the *'ADD RECORD'* menu option. The system was designed to reduce the amount of typing required. Important for cranky mechanics that can't type well. When entering a part number, the last five part numbers entered were brought up on the screen, that way if you got distracted in the middle of entering a series of parts, you could come back and immediately be remined what you had entered already.



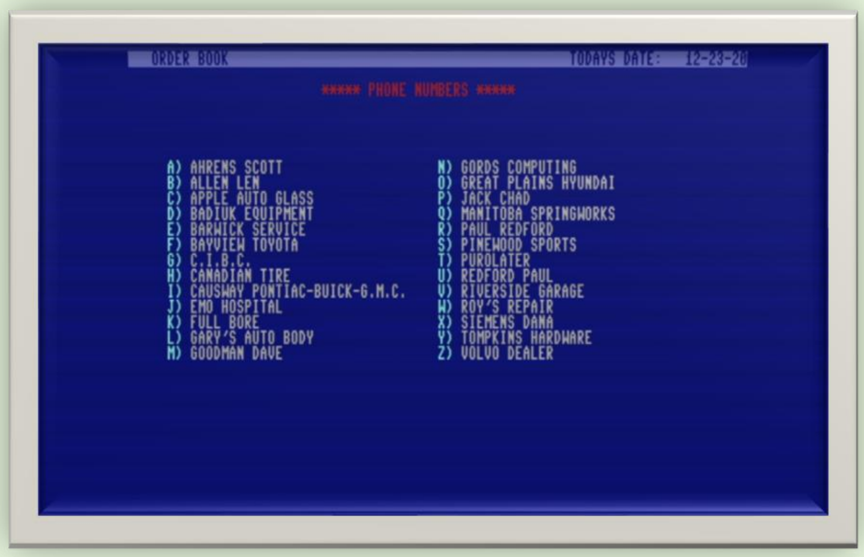
```
   ORDER BOOK                                      TODAYS DATE:   12-23-20
RECORD NUMBER: 78         BYTES: 47733      REMAINING RECORDS: 122

     CUSTOMER NAME: GORD CLINK
           ADDRESS: BARWICK ONTARIO                    ORDER TYPE
      PHONE NUMBER: 487-2221                          1) KAWASAKI
                                                      2) TRANS CYCLE
                                                      3) SIMPLEX
                    ORDER TYPE:                       4) ARCTIC CAT
                   PART NUMBER:                       5) SUN & SNOW
                    QTY & DESC:                       6) KIMPEX
                    UNIT PRICE: $                     7) FULL BORE
                                                      8) HUSQUARNA
                      COMMENTS:

     F3=PREVIOUS RECORD DATA
```



```
   ORDER BOOK                                      TODAYS DATE:   12-23-20
RECORD NUMBER: 78         BYTES: 47733      REMAINING RECORDS: 122

     CUSTOMER NAME: GORD CLINK
           ADDRESS: BARWICK ONTARIO
      PHONE NUMBER: 487-2221                          LAST FIVE
                                                      718-064
                                                      494252
                    ORDER TYPE: ARCTIC CAT            3001-082
                   PART NUMBER:                       606-270
                    QTY & DESC:                       187-060
                    UNIT PRICE: $

                      COMMENTS:

     F3=PREVIOUS RECORD DATA
```

Auto Dial of Phone Numbers

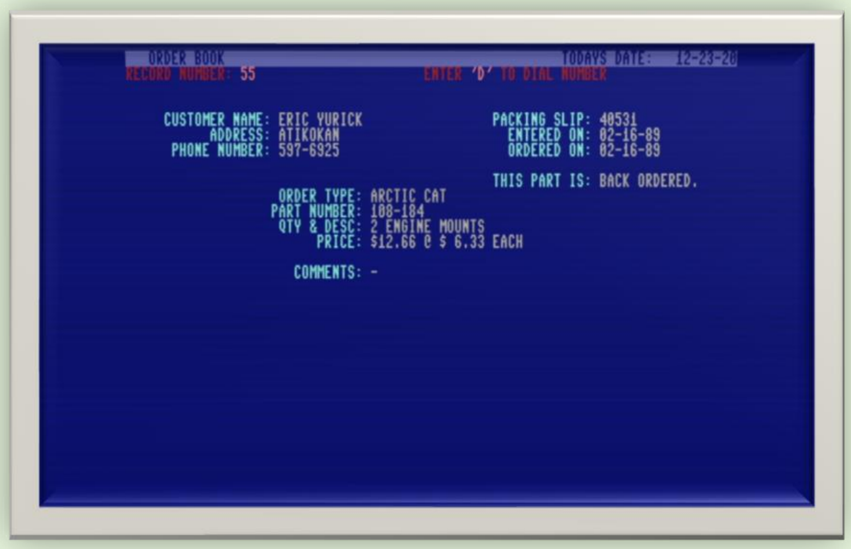
This was a really convenient feature that I built into the system that allowed for phone numbers to be automatically dialed. There was a phone book directory that allowed for 26 phone numbers. We used it for staff and suppliers. When you brought up the phone directory, by hitting 'CMDR P', You were given the option to pick one of the 26 entries, and dial the number. You just picked the phone up, and hit 'D' on the keyboard, and it would dial it for you.



```
   ORDER BOOK                                  TODAYS DATE:   12-23-20

                      ***** PHONE NUMBERS *****


        A) AHRENS SCOTT                 N) GORDS COMPUTING
        B) ALLEN LEN                    O) GREAT PLAINS HYUNDAI
        C) APPLE AUTO GLASS             P) JACK CHAD
        D) BADIUK EQUIPMENT             Q) MANITOBA SPRINGWORKS
        E) BARWICK SERVICE              R) PAUL REDFORD
        F) BAYVIEW TOYOTA               S) PINEWOOD SPORTS
        G) C.I.B.C.                     T) PUROLATER
        H) CANADIAN TIRE                U) REDFORD PAUL
        I) CAUSWAY PONTIAC-BUICK-G.M.C. V) RIVERSIDE GARAGE
        J) EMO HOSPITAL                 W) ROY'S REPAIR
        K) FULL BORE                    X) SIEMENS DANA
        L) GARY'S AUTO BODY             Y) TOMPKINS HARDWARE
        M) GOODMAN DAVE                 Z) VOLVO DEALER
```

As well, when bringing up a client's record using the '*Search Names'* menu option, you had the ability to dial the number by hitting the 'D' key.

These functions were accomplished by using a modem connected to the 128. I believe it was the 1200 baud (Commodore 1670) version that I had at the time, but any autodial modem would have worked.



```
   ORDER BOOK                                  TODAYS DATE:   12-23-20
  RECORD NUMBER: 55               ENTER 'D' TO DIAL NUMBER

     CUSTOMER NAME: ERIC YURICK             PACKING SLIP: 40531
           ADDRESS: ATIKOKAN                  ENTERED ON: 02-16-89
      PHONE NUMBER: 597-6925                  ORDERED ON: 02-16-89

                                            THIS PART IS: BACK ORDERED.
                   ORDER TYPE: ARCTIC CAT
                  PART NUMBER: 108-184
                   QTY & DESC: 2 ENGINE MOUNTS
                        PRICE: $12.66 @ $ 6.33 EACH

                     COMMENTS: -
```

Screen Dump

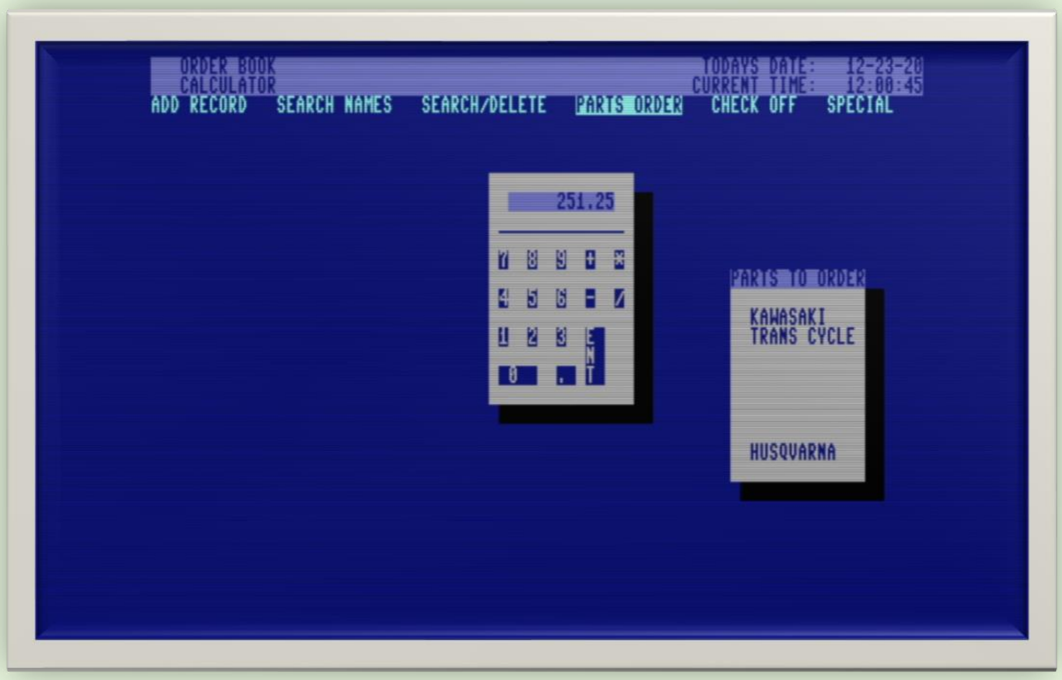
I integrated a screen dump utility written in machine language that I most likely got from a magazine. By hitting ‘shift/restore’ you could dump anything on the screen to the printer. When the system autobooted from the 1571 drive, it would first load a small program called ‘Order Book 128 V1.3’ that would install the machine language program ‘Screen Dump’ by reading data statements and poking them into the appropriate locations. Once done, it would then load the Order Book 128 main code, intuitively called ‘main code’, which was written entirely in BASIC 7.0.

File Transfer with MS-DOS

Under the ‘SPECIAL’ menu item, there is an option called ‘FILE TRANSFER’ which invoked a slightly modified version of ‘Super Sweep 128’ utility written by M. Garamszeghy. This utility allowed transferring files between MS-DOS disks and the 128. If I recall, I think we used this to transfer Orderbook data to the Tandy 1000 where we had a spreadsheet for total inventory. This program is written completely in BASIC 7.0, so if you selected this, the system would immediately start saving the current data to disk, and then load SuperSweep 128. Once you were done using SuperSweep, and selected ‘Q’ to quit, the system would automatically boot back into Orderbook 128 and reload the data.

On Screen Calculator

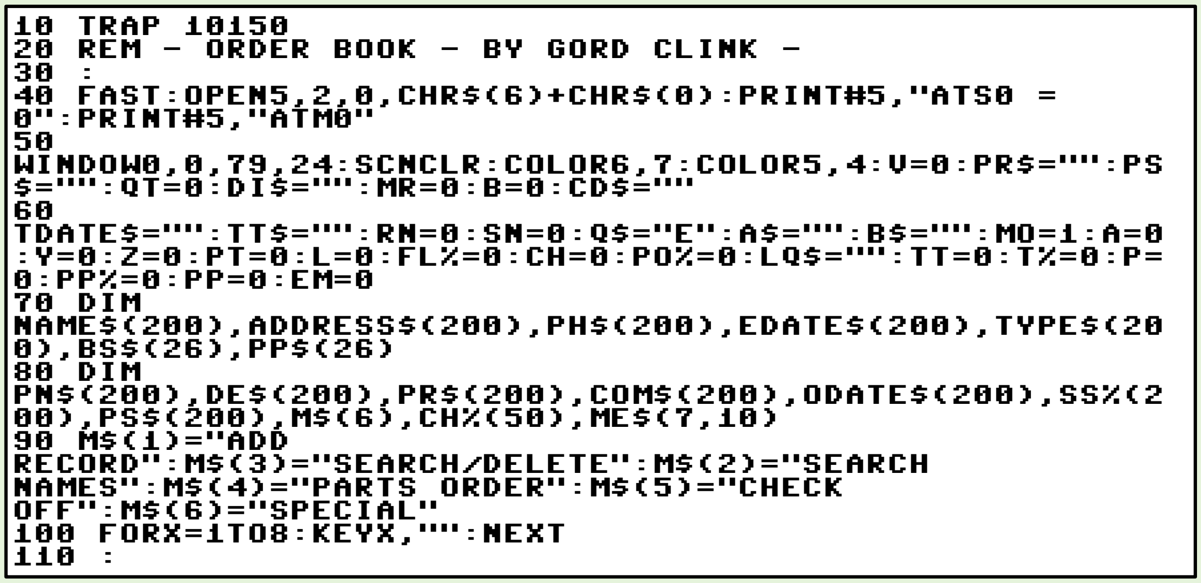
One handy feature of the system was by typing ‘CMDR C’ a calculator would pop up allowing you to do simple addition, subtraction, multiplication and division. Since the 128 had a nice number pad, it was very convenient for doing calculations for markup pricing when putting parts in the system for ordering.



Subroutines:

Variable Declaration

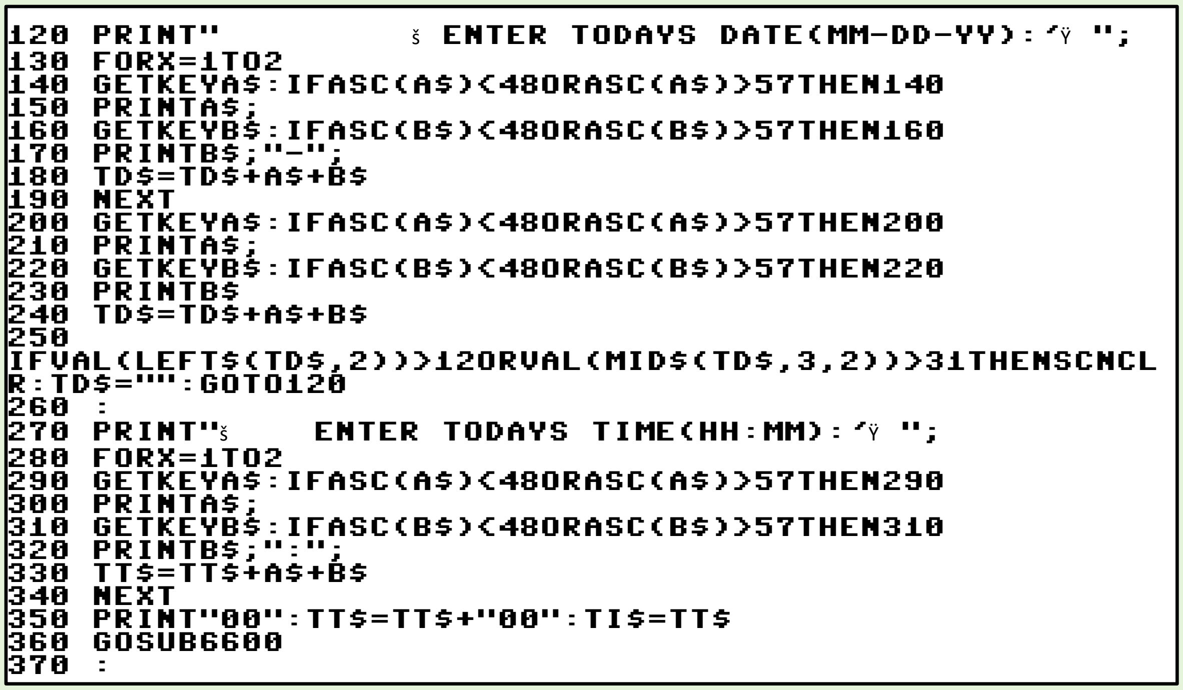
There are approximately 55 variables used of which 18 were subscripted. I noticed while going through my code that sometimes I would use the long form of the variable name, and sometimes the short form. i.e., tdate$ and td$. Of course, these are both the same as Commodore basic only recognizes the first two characters to identify the variable. I'm really not sure why I did this, other than maybe just being absent minded about it and not being consistent.



```
10 TRAP 10150
20 REM - ORDER BOOK - BY GORD CLINK -
30 :
40 FAST:OPEN5,2,0,CHR$(6)+CHR$(0):PRINT#5,"ATS0 =
0":PRINT#5,"ATM0"
50
WINDOW0,0,79,24:SCNCLR:COLOR6,7:COLOR5,4:V=0:PR$="":PS
$="":QT=0:DI$="":MR=0:B=0:CD$=""
60
TDATE$="":TT$="":RN=0:SN=0:Q$="E":A$="":B$="":MO=1:A=0
:Y=0:Z=0:PT=0:L=0:FL%=0:CH=0:PO%=0:LQ$="":TT=0:T%=0:P=
0:PP%=0:PP=0:EM=0
70 DIM
NAME$(200),ADDRESS$(200),PH$(200),EDATE$(200),TYPE$(20
0),BS$(26),PP$(26)
80 DIM
PN$(200),DE$(200),PR$(200),COM$(200),ODATE$(200),SS%(2
00),PS$(200),M$(6),CH%(50),ME$(7,10)
90 M$(1)="ADD
RECORD":M$(3)="SEARCH/DELETE":M$(2)="SEARCH
NAMES":M$(4)="PARTS ORDER":M$(5)="CHECK
OFF":M$(6)="SPECIAL"
100 FORX=1TO8:KEYX,"":NEXT
110 :
```

Set Date and Time

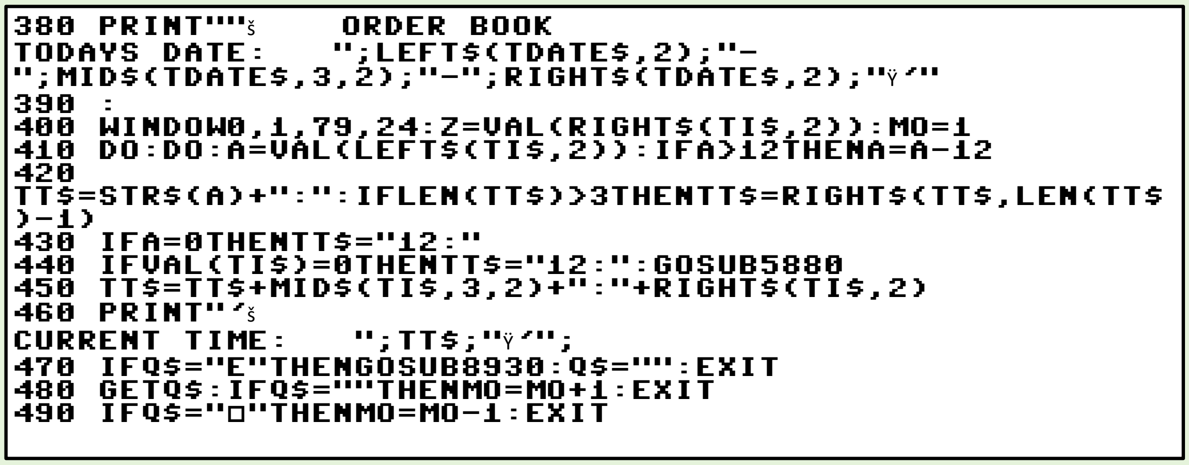
This routine is only used at program start for setting the current date and time. At the end of this routine, it gosubs to the Data Load routine



```
120 PRINT"          š ENTER TODAYS DATE(MM-DD-YY):´ÿ ";
130 FORX=1TO2
140 GETKEYA$:IFASC(A$)<48ORASC(A$)>57THEN140
150 PRINTA$;
160 GETKEYB$:IFASC(B$)<48ORASC(B$)>57THEN160
170 PRINTB$;"-";
180 TD$=TD$+A$+B$
190 NEXT
200 GETKEYA$:IFASC(A$)<48ORASC(A$)>57THEN200
210 PRINTA$;
220 GETKEYB$:IFASC(B$)<48ORASC(B$)>57THEN220
230 PRINTB$
240 TD$=TD$+A$+B$
250
IFVAL(LEFT$(TD$,2))>12ORVAL(MID$(TD$,3,2))>31THENSCNCL
R:TD$="":GOTO120
260 :
270 PRINT"š     ENTER TODAYS TIME(HH:MM):´ÿ ";
280 FORX=1TO2
290 GETKEYA$:IFASC(A$)<48ORASC(A$)>57THEN290
300 PRINTA$;
310 GETKEYB$:IFASC(B$)<48ORASC(B$)>57THEN310
320 PRINTB$;":";
330 TT$=TT$+A$+B$
340 NEXT
350 PRINT"00":TT$=TT$+"00":TI$=TT$
360 GOSUB6600
370 :
```

Main Menu

This routine is where the program sits most of the time waiting for user input. It displays the menus and also the current date and time, gosubs the ‘Parts To Order’ subroutine which displays current vendors with outstanding orders, and then, upon return, uses ‘get q$’ looking for cursor input which is used to select which menu item it highlights. The enter key will enter that menu item. While polling with q$, it is continuously updating the day and time on the screen. Besides cursor and enter-key input, the main menu routine will also scan for ‘Help’ key, CMDR A, CMDR C, CMDR S, CMDR O, CMDR M, CMDR +, CMDR -, SHFT +, SHFT -, and SHFT/RESTORE. Keep in mind that Commodore BASIC 7.0 didn’t have any built-in functions for menus, so I had to create the menu feature from scratch.



```
380 PRINT""š     ORDER BOOK
TODAYS DATE:    ";LEFT$(TDATE$,2);"-
";MID$(TDATE$,3,2);"-";RIGHT$(TDATE$,2);"ÿ´"
390 :
400 WINDOW0,1,79,24:Z=VAL(RIGHT$(TI$,2)):MO=1
410 DO:DO:A=VAL(LEFT$(TI$,2)):IFA>12THENA=A-12
420
TT$=STR$(A)+":":IFLEN(TT$)>3THENTT$=RIGHT$(TT$,LEN(TT$
)-1)
430 IFA=0THENTT$="12:"
440 IFVAL(TI$)=0THENTT$="12:":GOSUB5880
450 TT$=TT$+MID$(TI$,3,2)+":"+RIGHT$(TI$,2)
460 PRINT"´š
CURRENT TIME:    ";TT$;"ÿ´";
470 IFQ$="E"THENGOSUB8930:Q$="":EXIT
480 GETQ$:IFQ$=""THENMO=MO+1:EXIT
490 IFQ$="□"THENMO=MO-1:EXIT
```

```
500 IFPEEK(211)=16THENPRINT" š     PLEASE RELEASE CAPS-
LOCK KEY!´ ";
510 IFPEEK(211)=4ORPEEK(211)=8THENPRINT" š     DON´T
PRESS ME !!´ ";
520 IFPEEK(211)=1THENPRINT" š     PRESS ´+´ OR ´-´ TO
CHANGE TIME´ ";
530 IFQ$=CHR$(13)THENEXIT
540
IFQ$="█"THENBEGIN:B=VAL(TI$):B=B+100:IFB<100000THENTI$
="0"+RIGHT$(STR$(B),5)
550 IFB=>100000THENTI$=RIGHT$(STR$(B),6):BEND
560 IFQ$="▌"THENBEGIN:B=VAL(TI$):B=B-
100:IFB<100000THENTI$="0"+RIGHT$(STR$(B),5)
570 IFB=>100000THENTI$=RIGHT$(STR$(B),6):BEND
580 IFQ$="H"THENGOSUB7380
590 :
600 IFPEEK(211)=2THENBEGIN:PRINT"  š    FUNCTION
A,C,O,P,S,+,-   ´";
610
IFQ$="¦"THENBEGIN:B=VAL(TD$):B=B+100:IFB<100000THENTD$
="0"+RIGHT$(STR$(B),5)
620
IFB=>100000THENTD$=RIGHT$(STR$(B),6):WINDOW0,0,79,24
630 PRINT"  š     ORDER BOOK
TODAYS DATE:    ";LEFT$(TDATE$,2);"-
";MID$(TDATE$,3,2);"-";RIGHT$(TDATE$,2);"ÿ´"
640 WINDOW0,1,79,24:SCNCLR:Q$="E":BEND
650 IFQ$="▄"THENBEGIN:B=VAL(TD$):B=B-
100:IFB<100000THENTD$="0"+RIGHT$(STR$(B),5)
660
IFB=>100000THENTD$=RIGHT$(STR$(B),6):WINDOW0,0,79,24
670 PRINT"  š     ORDER BOOK
TODAYS DATE:    ";LEFT$(TDATE$,2);"-
";MID$(TDATE$,3,2);"-";RIGHT$(TDATE$,2);"ÿ´"
680 WINDOW0,1,79,24:SCNCLR:Q$="E":BEND
690 IFQ$="®"THENGOSUB3010:GOSUB6070
700 IFQ$="¼"THENGOSUB6820
710 IFQ$="¹"THENGOSUB8930:Q$="E"
720 IFQ$="-"THENGOSUB9100
730 IFQ$="§"THENGOSUB9630
740 BEND
750 LOOP
760 :
770 IFQ$=CHR$(27)THENEXIT
780 IFMO>6THENMO=1
790 IFMO<1THENMO=6
800 IFMO=1THENPRINT" ";M$(1);"´";"    ";M$(2);"
";M$(3);"    ";M$(4);"    ";M$(5);"    ";M$(6)
810 IFMO=2THENPRINTM$(1);"    ";" ";M$(2);"´";"
";M$(3);"    ";M$(4);"    ";M$(5);"    ";M$(6)
820 IFMO=3THENPRINTM$(1);"    ";M$(2);"
";" ";M$(3);"´";"    ";M$(4);"    ";M$(5);"    ";M$(6)
830 IFMO=4THENPRINTM$(1);"    ";M$(2);"    ";M$(3);"
";" ";M$(4);"´";"    ";M$(5);"    ";M$(6)
840 IFMO=5THENPRINTM$(1);"    ";M$(2);"    ";M$(3);"
";M$(4);"    ";" ";M$(5);"´";"    ";M$(6)
850 IFMO=6THENPRINTM$(1);"    ";M$(2);"    ";M$(3);"
";M$(4);"    ";M$(5);"    ";" ";M$(6);"´"
860 PRINT"´";
870 IFQ$=CHR$(13)THEN ONMO
GOSUB900,3250,2070,3540,4540,7550:PRINT" ";:Q$="E":DR$
="":DR%=0:L=0
880 LOOP
890 :
```

Add Records

All input within the program uses **'get'** and never **'input'**. I found the 'input' statement to be very sloppy and unprofessional. 'get' allows for precision control and makes for a more professional feel. I was able to do some neat things with data input using 'getkey'. For instance, when entering on the **QTY & Description** line, the computer expects you to enter a number, and when you do it records the number, and as soon as you hit the space-bar, it then switches to accepting the description. On the next line after entering the **Unit Price**, the computer will automatically calculate the extended price and put it at the end of the line. *(see screen image above for example).*

```
900 REM -- ADD RECORDS PROCEDURE --
910 :
920
DO:SCNCLR:Z=VAL(MID$(TI$,3,2)):RN=RN+1:MR=FRE(1):IFRN=
1THENMS=FRE(1):MR=MS-100:QT=0
930 IFRN>200THENSCNCLR:CHAR,25,10,"-PLEASE DELETE
UNWANTED RECORDS!ÿ":RN=RN-1:EXIT
940 PRINT" RECORD NUMBER:-";RN;"                BYTES:-
";FRE(1);"          REMAINING RECORDS:-";200-RN;"ÿ"
950 CHAR,5,3,"ÿCUSTOMER NAME: ›":CHAR,11,4,"ÿADDRESS:
›":CHAR,6,5,"ÿPHONE NUMBER: ›"
960 CHAR,20,8,"ÿORDER TYPE: ›":CHAR,19,9,"ÿPART NUMBER:
›":CHAR,20,10,"ÿQTY & DESC: ›":CHAR,20,11,"ÿUNIT PRICE:
›$"
970 CHAR,22,13,"ÿCOMMENTS: ›":CHAR,0,22," š
F3=PREVIOUS RECORD DATA
F7=STOCK ITEM                   ´":COLOR5,16
980 :
990 CHAR,5,3," CUSTOMER NAME:
›":CHAR,20,3:Q$="":KEY3,NAME$(RN-
1)+CHR$(13):KEY7,"STOCK
ITEM"+CHR$(13):DOUNTILQ$=CHR$(13)
1000 GETQ$:Y=VAL(MID$(TI$,3,2)):IFY-Z>5THENEXIT
1010
IFQ$=CHR$(20)ANDLEN(NA$(RN))>0THENNAME$(RN)=LEFT$(NAME
$(RN),LEN(NAME$(RN))-1):PRINTQ$;:Q$=""
1020 IFQ$=CHR$(20)ANDLEN(NA$(RN))=0THENQ$=""
1030
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)ORQ$=CHR$(34)THENQ$=""
1040
IFQ$<>CHR$(13)THENBEGIN:PRINTQ$;:PRINT"¤□";:NAME$(RN)=
NAME$(RN)+Q$:BEND
1050 LOOPUNTILQ$=CHR$(27):IFQ$=CHR$(27)THENEXIT
1060 IFY-Z>5THENRN=RN-1:GOSUB3010:GOSUB6070:EXIT
1070 PRINT" □";:CHAR,5,3,"ÿCUSTOMER
NAME:›":IFNA$(RN)=NA$(RN-1)THENAD$(RN)=AD$(RN-
1):PH$(RN)=PH$(RN-
1):CHAR,20,4,AD$(RN):CHAR,20,5,PH$(RN):GOTO1320
1080 IFNA$(RN)="STOCK
ITEM"THENAD$(RN)="¤":PH$(RN)="¤":CHAR,20,4,AD$(RN):CHAR
,20,5,PH$(RN):GOTO1320
1090 IFNA$(RN)=""THENNA$(RN)="-"
1100 :
1110 CHAR,11,4," ADDRESS:
›":CHAR,0,22," š      F3=PREVIOUS RECORD DATA
´":COLOR5,16
1120 KEY7,"":CHAR,20,4:Q$="":KEY3,AD$(RN-
1):DOUNTILQ$=CHR$(13):GETKEYQ$:IFQ$=CHR$(20)ANDLEN(AD$
(RN))>0THENAD$(RN)=LEFT$(AD$(RN),LEN(AD$(RN))-
1):PRINTQ$;:Q$=""
1130 IFQ$=CHR$(20)ANDLEN(AD$(RN))=0THENQ$=""
1140
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
```

```
R$(9)ORQ$=CHR$(10)ORQ$=CHR$(34)THENQ$=""
1150
IFQ$<>CHR$(13)ANDQ$<>CHR$(141)THENBEGIN:PRINTQ$;:PRINT
"¤□";:AD$(RN)=AD$(RN)+Q$:BEND
1160
LOOPUNTILQ$=CHR$(27)ORQ$=CHR$(141):IFQ$=CHR$(27)THENEX
IT
1170
IFQ$=CHR$(141)THENNA$(RN)="":AD$(RN)="":CHAR,11,4,"ÿAD
DRESS:                          ":GOTO990
1180 PRINT"
□";:CHAR,11,4,"ÿADDRESS:":IFAD$(RN)=""THENAD$(RN)="-"
1190 :
1200 CHAR,6,5," PHONE NUMBER:           ›"
1210 CHAR,20,5:Q$="":KEY3,PH$(RN-
1):DOUNTILQ$=CHR$(13):GETKEYQ$:IFQ$=CHR$(20)ANDLEN(PH$
(RN))>0THENPH$(RN)=LEFT$(PH$(RN),LEN(PH$(RN))-
1):PRINTQ$;
1220
IFLEN(PH$(RN))=3THENPH$(RN)=LEFT$(PH$(RN),LEN(PH$(RN))
-1):PRINTQ$;
1230 IFQ$=CHR$(20)THENQ$=""
1240 IFQ$=CHR$(20)ANDLEN(PH$(RN))=0THENQ$=""
1250
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)THENQ$=""
1260
IFQ$<>CHR$(13)ANDASC(Q$)>47ANDASC(Q$)<58THENBEGIN:PRIN
TQ$;:PRINT"¤□";:PH$(RN)=PH$(RN)+Q$:BEND
1270 IFLEN(PH$(RN))=3THENPRINT"-";:PH$(RN)=PH$(RN)+"-"
1280
LOOPUNTILQ$=CHR$(27)ORQ$=CHR$(141):IFQ$=CHR$(27)THENEX
IT
1290
IFQ$=CHR$(141)THENAD$(RN)="":PH$(RN)="":CHAR,6,5,"ÿPHO
NE NUMBER:›             ":GOTO1110
1300 PRINT" □";:CHAR,6,5,"ÿPHONE
NUMBER:":IFPH$(RN)=""THENPH$(RN)="-"
1310 :
1320 CHAR,20,8," ORDER TYPE:
":CHAR,0,22," š      F3=PREVIOUS RECORD DATA
´":COLOR5,16:KEY7,""
1330 FORX=5TO13:CHAR,77,X," □   ´ÿ":NEXT:CHAR,61,13," □
´ÿ"
1340 KEY3,"":CHAR,60,4,"š     ORDER TYPE
":CHAR,60,5,"ÿ   1) KAWASAKI     ´":CHAR,60,6,"   2)
TRANS CYCLE ´":CHAR,60,7,"   3) SIMPLEX      ´"
1350 CHAR,60,8,"   4) ARCTIC CAT   ´":CHAR,60,9,"   5)
SUN & SNOW   ´":CHAR,60,10,"   6) KIMPEX
´":CHAR,60,11,"   7) FULL BORE    ´"
1360 CHAR,60,12,"   8) HUSQVARNA    ´›"
1370
DOUNTILQ$=CHR$(141):CHAR,32,8:Q$="":KEY3,LQ$:GETKEYQ$:
LQ$=Q$
1380
IFQ$="1"THENTYPE$(RN)="KAWASAKI":PRINTTYPE$(RN):EXIT
1390 IFQ$="2"THENTYPE$(RN)="TRANS
CYCLE":PRINTTYPE$(RN):EXIT
1400
IFQ$="3"THENTYPE$(RN)="SIMPLEX":PRINTTYPE$(RN):EXIT
1410 IFQ$="4"THENTYPE$(RN)="ARCTIC
CAT":PRINTTYPE$(RN):EXIT
1420 IFQ$="5"THENTYPE$(RN)="SUN &
SNOW":PRINTTYPE$(RN):EXIT
1430
IFQ$="6"THENTYPE$(RN)="KIMPEX":PRINTTYPE$(RN):EXIT
1440 IFQ$="7"THENTYPE$(RN)="FULL
BORE":PRINTTYPE$(RN):EXIT
1450
IFQ$="8"THENTYPE$(RN)="HUSQVARNA":PRINTTYPE$(RN):EXIT
1460 IFQ$=CHR$(141)THENPH$(RN)="":EXIT
1470 LOOPUNTILQ$=CHR$(27):IFQ$=CHR$(27)THENEXIT
```

```
1480 FORX=4TO14:CHAR,60,X,"                              ":NEXT
1490 CHAR,20,8,"ÿORDER
TYPE:":IFTY$(RN)=""THENTY$(RN)="-"
1500 IFQ$=CHR$(141)THENGOTO1200
1510 :
1520 CHAR,19,9," PART NUMBER:›                     "
1530 IFRN>5THENBEGIN:CHAR,60,5,"š   LAST FIVE
´ÿ":FORX=6TO11:CHAR,60,X,"ÿ
´":NEXT:FORX=5TO1STEP-1:CHAR,60,5+6-X,"
"+LEFT$(PN$(RN-X),11)+"´":NEXT
1540 FORX=6TO12:CHAR,73,X," □   ´ÿ":NEXT:CHAR,61,12," □
´ÿ"
1550 BEND
1560 COLOR5,16:CHAR,32,9:WINDOW32,10,52,10
1570 Q$="":KEY3,PN$(RN-
1):DOUNTILQ$=CHR$(13):GETKEYQ$:IFQ$=CHR$(20)ANDLEN(PN$
(RN))>0THENPN$(RN)=LEFT$(PN$(RN),LEN(PN$(RN))-
1):PRINTQ$;:Q$=""
1580 IFQ$=CHR$(20)ANDLEN(PN$(RN))=0THENQ$=""
1590
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)THENQ$=""
1600
IFQ$<>CHR$(13)THENBEGIN:PRINTQ$;:PRINT"¤□";:PN$(RN)=PN
$(RN)+Q$:BEND
1610 LOOPUNTILQ$=CHR$(27)ORQ$=CHR$(141):PRINT"
□";:IFQ$=CHR$(27)THENEXIT
1620
IFQ$=CHR$(141)THENPN$(RN)="":TY$(RN)="":WINDOW0,1,79,2
4:CHAR,19,9,"ÿPART NUMBER:›
":Q$="":GOTO1320
1630 WINDOW0,1,79,24:FORX=12TO5STEP-1:CHAR,60,X,"
":NEXT
1640 CHAR,19,9,"ÿPART
NUMBER:":IFPN$(RN)=""THENPN$(RN)="-"
1650 :
1660 CHAR,19,10,"    QUANTITY:
›":CHAR,32,10
1670 Q$="":KEY3,DE$(RN-
1):DOUNTILQ$=CHR$(13):GETKEYQ$:IFQ$=CHR$(20)ANDLEN(DE$
(RN))>0THENDE$(RN)=LEFT$(DE$(RN),LEN(DE$(RN))-
1):PRINTQ$;:Q$=""
1680 IFQ$=CHR$(20)ANDLEN(DE$(RN))=0THENQ$=""
1690
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)THENQ$=""
1700 IFQ$=" "ANDLEN(DE$(RN))=0THENQ$=""
1710 IFINSTR(DE$(RN)," ")=0THENIFQ$<>"
"THENIFQ$<>CHR$(27)THENIFASC(Q$)<48ORASC(Q$)>57THENIFQ
$<>CHR$(141)THENQ$=""
1720 IFQ$=" "ANDINSTR(DE$(RN),"
")=0THENQT=VAL(DE$(RN)):CHAR,19,10," DESCRIPTION:›":CH
AR,(32+LEN(DE$(RN))),10
1730
IFQ$<>CHR$(13)ANDQ$<>CHR$(141)THENBEGIN:PRINTQ$;:PRINT
"¤□";:DE$(RN)=DE$(RN)+Q$:BEND
1740
LOOPUNTILQ$=CHR$(27)ORQ$=CHR$(141):IFQ$=CHR$(27)THENEX
IT
1750
IFQ$=CHR$(141)THENDE$(RN)="":PN$(RN)="":CHAR,20,10,"ÿQ
TY & DESC:›                  ":Q$="":GOTO1520
1760 PRINT" □";:CHAR,19,10,"ÿ QTY &
DESC:":IFDE$(RN)=""THENDE$(RN)="-"
1770 :
1780 CHAR,20,11," UNIT PRICE:› $
":CHAR,34,11
1790 Q$="":KEY3,PR$(RN-
1):DOUNTILQ$=CHR$(13):GETKEYQ$:IFQ$=CHR$(20)ANDLEN(PR$
(RN))>0THENPR$(RN)=LEFT$(PR$(RN),LEN(PR$(RN))-
1):PRINTQ$;:Q$=""
```

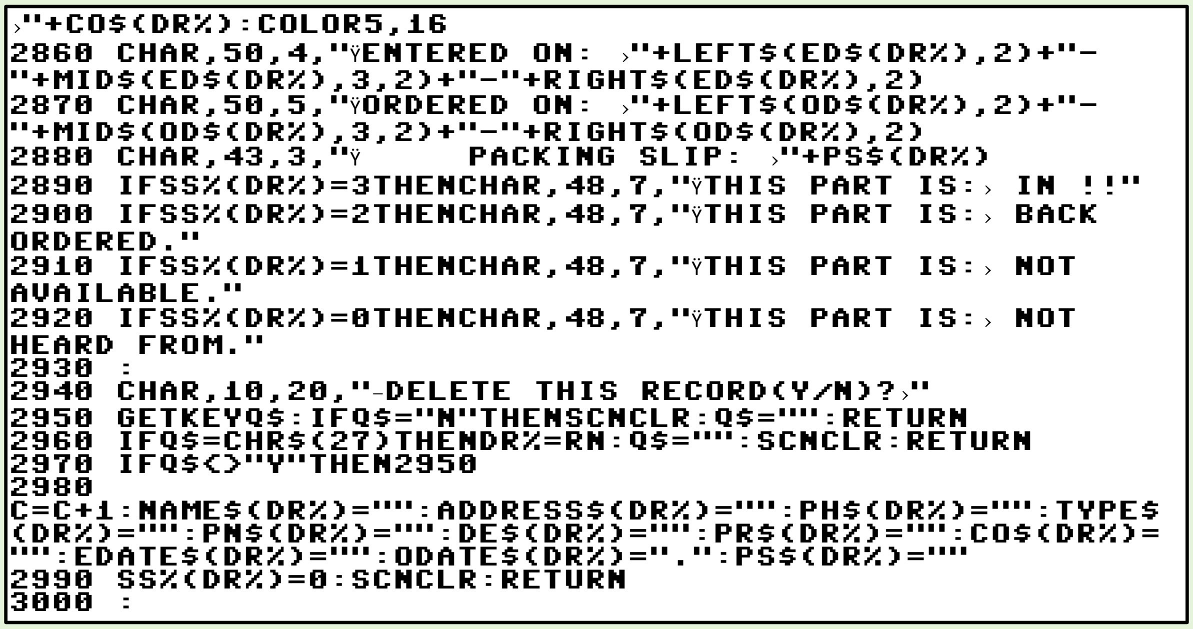
```
1800 IFQ$=CHR$(20)ANDLEN(PR$(RN))=0THENQ$=""
1810
IFQ$<>CHR$(13)THENIFQ$<>CHR$(27)THENIFASC(Q$)<48ORASC(
Q$)>57THENIFQ$<>CHR$(141)THENQ$=""
1820
IFQ$<>CHR$(13)ANDQ$<>CHR$(141)THENPRINTQ$;:PRINT"¤□";:
PR$(RN)=PR$(RN)+Q$
1830
LOOPUNTILQ$=CHR$(27)ORQ$=CHR$(141):IFQ$=CHR$(27)THENEX
IT
1840
IFQ$=CHR$(141)THENPR$(RN)="":DE$(RN)="":CHAR,20,11,"ÿU
NIT PRICE:› $                         ":Q$="":GOTO1660
1850 PRINT"
□";:IFPR$(RN)=""THENPR$(RN)="N/C":CHAR,34,11,PR$(RN):C
HAR,20,11,"ÿ      PRICE:›"
1860
IFPR$(RN)<>"N/C"THENBEGIN:PR$(RN)=STR$(VAL(PR$(RN))*QT
)
1870 PR$(RN)=RIGHT$(PR$(RN),LEN(PR$(RN))-
1):PR$(RN)=LEFT$(PR$(RN),LEN(PR$(RN))-
2)+"."+RIGHT$(PR$(RN),2):PR$(RN)=PR$(RN)+" @
$"+STR$(VAL(PR$(RN))/QT)
1880 CHAR,20,11,"ÿ
PRICE:›":CHAR,34,11,PR$(RN):BEND
1890 :
1900 CHAR,0,22," š      F3=PREVIOUS RECORD DATA
F7=TELL DAVE WHEN IN      ´":COLOR5,16
1910 CHAR,22,13," COMMENTS:›":CHAR,32,13:KEY7,"TELL
DAVE WHEN IN"
1920 Q$="":KEY3,CO$(RN-
1):DOUNTILQ$=CHR$(13):GETKEYQ$:IFQ$=CHR$(20)ANDLEN(CO$
(RN))>0THENCO$(RN)=LEFT$(CO$(RN),LEN(CO$(RN))-
1):PRINTQ$;:Q$=""
1930 IFQ$=CHR$(20)ANDLEN(CO$(RN))=0THENQ$=""
1940
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)THENQ$=""
1950
IFQ$<>CHR$(13)ANDQ$<>CHR$(141)THENBEGIN:PRINTQ$;:PRINT
"¤□";:CO$(RN)=CO$(RN)+Q$:BEND
1960
LOOPUNTILQ$=CHR$(27)ORQ$=CHR$(141):IFQ$=CHR$(27)THENEX
IT
1970
IFQ$=CHR$(141)THENPR$(RN)="":CO$(RN)="":CHAR,22,13,"ÿC
OMMENTS:                          ":GOTO1780
1980 PRINT"
□";:CHAR,22,13,"ÿCOMMENTS:":IFCO$(RN)=""THENCO$(RN)="-
"
1990 :
2000 Q$="": KEY3,"":KEY7,"":CHAR,8,20,"-IS THIS RECORD
CORRECT?:ÿ":DOUNTILQ$="Y"ORQ$="N":GETKEYQ$:LOOP
2010 CHAR,32,20,Q$
2020 IFQ$="N"THENRN=RN-1
2030 IFQ$="Y"THENED$(RN)=TD$:PS$(RN)="-":OD$(RN)="."
2040
LOOP:IFQ$=CHR$(27)THENED$(RN)="":NA$(RN)="":AD$(RN)=""
:PH$(RN)="":TYPE$(RN)="":PN$(RN)="":DE$(RN)="":PR$(RN)
="":COM$(RN)="":RN=RN-1:WINDOW0,1,79,24:SCNCLR
2050 KEY3,"":KEY7,"":RETURN
2060 :
```

Delete Records

The system always kept the records sorted in memory in alphabetical order. So if a record was deleted, it would then call the sort routine to re-sort the records before bringing you back to the main menu. This would effectively get rid of the hole that deleting a record would cause.

```
2070 REM -- DELETE RECORDS PROCEDURE --
2080 :
2090 C=0:IFRN=0THENCHAR,20,8,"-NO RECORDS IN
MEMORY.ÿ":SLEEP1:SCNCLR:Q$="E":RETURN
2100
MO=1:Q$="E":Z=VAL(MID$(TI$,3,2)):DO:DOUNTILQ$=CHR$(13)
ORQ$=CHR$(27)ORQ$="E":GETKEYQ$:IFQ$=" "THENMO=MO+1:EXI
T
2110 IFQ$="´"THENMO=MO-1:EXIT
2120 IFQ$=CHR$(27)THENEXIT
2130 Y=VAL(MID$(TI$,3,2)):IFY-Z>1THENQ$=CHR$(27):EXIT
2140 LOOP
2150 IFMO>4THENMO=1
2160 IFMO<1THENMO=4
2170 CHAR,20,3,"š     SEARCH/DELETE     ´ÿ":CHAR,20,4,"
´":CHAR,20,9,"                          ´"
2180 FORX=4TO10:CHAR,40,X," □   ´ÿ":NEXT:CHAR,21,10,"□
´ÿ"
2190 IFMO=1THENCHAR,20,5,"  ´USE RECORD NUMBER
´":CHAR,20,6,"  SEARCH NAMES       ´":CHAR,20,7,"
SEARCH PART NUMBER ´":CHAR,20,8,"  SEARCH DESCRIPTION
´"
2200 IFMO=2THENCHAR,20,5,"  USE RECORD NUMBER
´":CHAR,20,6,"  ´SEARCH NAMES      ´":CHAR,20,7,"
SEARCH PART NUMBER ´"
2210 IFMO=3THENCHAR,20,6,"  SEARCH NAMES
´":CHAR,20,7,"  ´SEARCH PART NUMBER  ´":CHAR,20,8,"
SEARCH DESCRIPTION ´"
2220 IFMO=4THENCHAR,20,7,"  SEARCH PART NUMBER
´":CHAR,20,8,"  ´SEARCH DESCRIPTION  ´":CHAR,20,5,"
USE RECORD NUMBER  ´"
2230 IFQ$=CHR$(27)ORQ$=CHR$(13)THENEXIT
2240 Q$="":LOOP
2250 IFQ$=CHR$(27)THENFORX=10TO3STEP-1:CHAR,20,X,"
":NEXT:MO=3:RETURN
2260 :
2270 SCNCLR:L=1:Q$="":DR$="":DR%=0
2280 IFMO=1THENBEGIN:YY=RN
2290 DO
2300 PRINT"ÿ   ENTER RECORD NUMBER(1 -
";RN;"):›";:FORX=1TO3:GETKEYA$:IFA$=CHR$(13)THENX=3:GO
TO2420
2310 IFA$=" "THENBEGIN:S=15:IFYY-S<2THENS=YY
2320 WINDOW40,5,79,21,1:FORY=YYTOYY-SSTEP-
1:PRINT"š";NA$(Y);"›";SPC(25-
LEN(NA$(Y)));Y:NEXT:WINDOW0,1,79,24:YY=YY-
15:IFYY<1THENYY=RN:A$=""
2330 A$=""
2340 BEND
2350 IFA$="´"THENBEGIN:S=15:IFYY+S>RNTHENS=RN-YY
2360 WINDOW40,5,79,21,1:FORY=YY+STOYYSTEP-
1:PRINT"š";NA$(Y);"›";SPC(25-
LEN(NA$(Y)));Y:NEXT:WINDOW0,1,79,24:YY=YY+15:IFYY>RNTH
ENYY=1:A$=""
2370 A$=""
2380 BEND
2390 IFA$=CHR$(27)THENBEGIN:IFC>0THENGOSUB3010
2400 X=3:MO=3:RN=RN-C:C=0:SCNCLR:RETURN:BEND
2410
IFA$<>CHR$(13)THENIFASC(A$)<48ORASC(A$)>57THENA$="":X=
X-1
2420 IFA$<>""THENQ$=Q$+A$:CHAR,32+X,3:PRINTA$"¤□";
2430 NEXT:DR%=VAL(Q$)
2440 IFDR%<1ORDR%>RNTHENGOTO2270
```

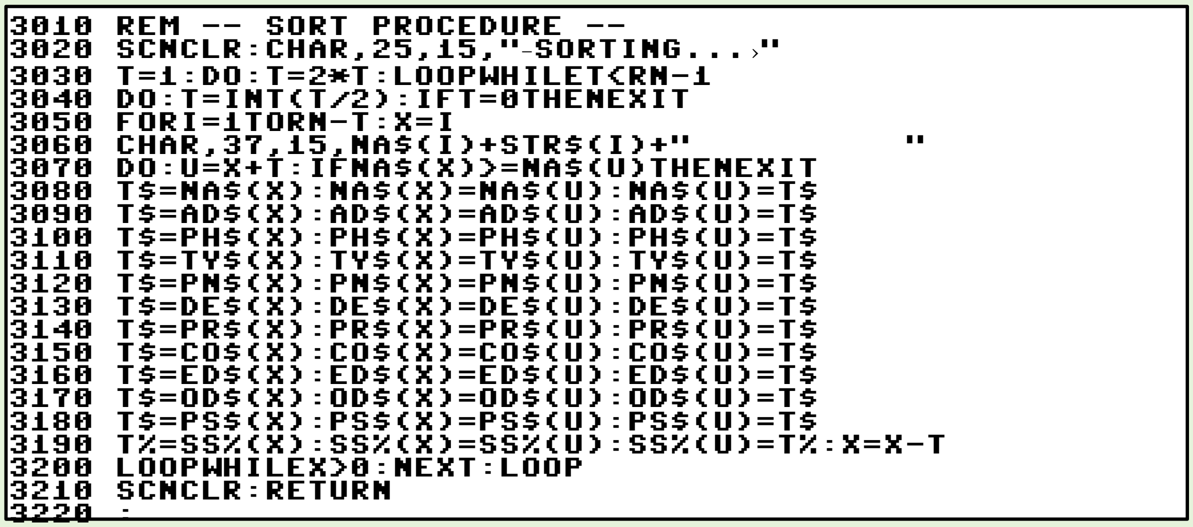
```
2450 GOSUB2810
2460 SCNCLR:Q$="":LOOP:BEND
2470 :
2480 IFMO=2THENBEGIN:PRINT"ÿ   ENTER NAME TO SEARCH
FOR:› ";
2490
DOUNTILQ$=CHR$(13)ORQ$=CHR$(27):PRINT"¤□";:GETKEYQ$:IF
Q$=CHR$(20)THENDR$=LEFT$(DR$,LEN(DR$)-
1):PRINTQ$;:Q$=""
2500
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)THENQ$=""
2510
IFQ$<>CHR$(13)THENBEGIN:PRINTQ$;:DR$=DR$+Q$:BEND:LOOP
2520 IFQ$=CHR$(27)THENSCNCLR:SCNCLR:MO=2:RETURN
2530
FORX=LTORN:IFINSTR(NA$(X),DR$)>0THENDR%=X:X=RN:FL%=1
2540 NEXT
2550 IFFL%=1THENGOSUB2810:L=DR%+1:GOTO2530
2560 SCNCLR:IFC>0THENGOSUB3010
2570 RN=RN-C:C=0:MO=3:RETURN:BEND
2580 :
2590 IFMO=3THENBEGIN:PRINT"ÿ   ENTER PART NUMBER TO
SEARCH FOR:› ";
2600
DOUNTILQ$=CHR$(13)ORQ$=CHR$(27):PRINT"¤□";:GETKEYQ$:IF
Q$=CHR$(20)THENDR$=LEFT$(DR$,LEN(DR$)-
1):PRINTQ$;:Q$=""
2610
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)THENQ$=""
2620
IFQ$<>CHR$(13)THENBEGIN:PRINTQ$;:DR$=DR$+Q$:BEND:LOOP
2630 IFQ$=CHR$(27)THENSCNCLR:SCNCLR:MO=2:RETURN
2640
FORX=LTORN:IFINSTR(PN$(X),DR$)>0THENDR%=X:X=RN:FL%=1
2650 NEXT
2660 IFFL%=1THENGOSUB2810:L=DR%+1:GOTO2640
2670 SCNCLR:IFC>0THENGOSUB3010
2680 RN=RN-C:C=0:MO=3:RETURN:BEND
2690 :
2700 IFMO=4THENBEGIN:PRINT"ÿ   ENTER PART DESCRIPION
TO SEARCH FOR:› ";
2710
DOUNTILQ$=CHR$(13)ORQ$=CHR$(27):PRINT"¤□";:GETKEYQ$:IF
Q$=CHR$(20)THENDR$=LEFT$(DR$,LEN(DR$)-
1):PRINTQ$;:Q$=""
2720
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)THENQ$=""
2730
IFQ$<>CHR$(13)THENBEGIN:PRINTQ$;:DR$=DR$+Q$:BEND:LOOP
2740 IFQ$=CHR$(27)THENSCNCLR:SCNCLR:MO=2:RETURN
2750
FORX=LTORN:IFINSTR(DE$(X),DR$)>0THENDR%=X:X=RN:FL%=1
2760 NEXT
2770 IFFL%=1THENGOSUB2810:L=DR%+1:GOTO2750
2780 SCNCLR:IFC>0THENGOSUB3010
2790 RN=RN-C:C=0:MO=3:RETURN:BEND
2800 :
2810 DO:SCNCLR:FL%=0:PRINT" RECORD NUMBER:-";DR%
2820 CHAR,22,13,"ÿCOMMENTS: ›"+CO$(DR%):COLOR5,16
2830 CHAR,5,3,"ÿCUSTOMER NAME:
›"+NA$(DR%):CHAR,11,4,"ÿADDRESS:
›"+AD$(DR%):CHAR,6,5,"ÿPHONE NUMBER: ›"+PH$(DR%)
2840 CHAR,20,8,"ÿORDER TYPE:
›"+TY$(DR%):CHAR,19,9,"ÿPART NUMBER:
›"+PN$(DR%):CHAR,20,10,"ÿQTY & DESC: ›"+DE$(DR%)
2850 CHAR,22,11,"ÿ   PRICE:
›$"+PR$(DR%):CHAR,22,13,"ÿCOMMENTS:
```



```
›"+CO$(DR%):COLOR5,16
2860 CHAR,50,4,"ÿENTERED ON:  ›"+LEFT$(ED$(DR%),2)+"-
"+MID$(ED$(DR%),3,2)+"-"+RIGHT$(ED$(DR%),2)
2870 CHAR,50,5,"ÿORDERED ON:  ›"+LEFT$(OD$(DR%),2)+"-
"+MID$(OD$(DR%),3,2)+"-"+RIGHT$(OD$(DR%),2)
2880 CHAR,43,3,"ÿ      PACKING SLIP:  ›"+PS$(DR%)
2890 IFSS%(DR%)=3THENCHAR,48,7,"ÿTHIS PART IS:› IN !!"
2900 IFSS%(DR%)=2THENCHAR,48,7,"ÿTHIS PART IS:› BACK
ORDERED."
2910 IFSS%(DR%)=1THENCHAR,48,7,"ÿTHIS PART IS:› NOT
AVAILABLE."
2920 IFSS%(DR%)=0THENCHAR,48,7,"ÿTHIS PART IS:› NOT
HEARD FROM."
2930 :
2940 CHAR,10,20,"-DELETE THIS RECORD(Y/N)?›"
2950 GETKEYQ$:IFQ$="N"THENSCNCLR:Q$="":RETURN
2960 IFQ$=CHR$(27)THENDR%=RN:Q$="":SCNCLR:RETURN
2970 IFQ$<>"Y"THEN2950
2980
C=C+1:NAME$(DR%)="":ADDRESS$(DR%)="":PH$(DR%)="":TYPE$
(DR%)="":PN$(DR%)="":DE$(DR%)="":PR$(DR%)="":CO$(DR%)=
"":EDATE$(DR%)="":ODATE$(DR%)=".":PS$(DR%)=""
2990 SS%(DR%)=0:SCNCLR:RETURN
3000 :
```

Sort Routine

I had learned several sort routines at the time. Bubble Sort, Insertion Sort, and Shell Sort. After coding examples of each and then testing, I felt that 'Shell Sort' was the most efficient and used that in the program. It seems to work well. Later I learned how to do 'Quick Sort' but didn't bother to implement it as the 'Shell Sort' was adequate. The system would automatically sort before saving the data to disk.



```
3010 REM -- SORT PROCEDURE --
3020 SCNCLR:CHAR,25,15,"-SORTING...›"
3030 T=1:DO:T=2*T:LOOPWHILET<RN-1
3040 DO:T=INT(T/2):IFT=0THENEXIT
3050 FORI=1TORN-T:X=I
3060 CHAR,37,15,NA$(I)+STR$(I)+"        "
3070 DO:U=X+T:IFNA$(X)>=NA$(U)THENEXIT
3080 T$=NA$(X):NA$(X)=NA$(U):NA$(U)=T$
3090 T$=AD$(X):AD$(X)=AD$(U):AD$(U)=T$
3100 T$=PH$(X):PH$(X)=PH$(U):PH$(U)=T$
3110 T$=TY$(X):TY$(X)=TY$(U):TY$(U)=T$
3120 T$=PN$(X):PN$(X)=PN$(U):PN$(U)=T$
3130 T$=DE$(X):DE$(X)=DE$(U):DE$(U)=T$
3140 T$=PR$(X):PR$(X)=PR$(U):PR$(U)=T$
3150 T$=CO$(X):CO$(X)=CO$(U):CO$(U)=T$
3160 T$=ED$(X):ED$(X)=ED$(U):ED$(U)=T$
3170 T$=OD$(X):OD$(X)=OD$(U):OD$(U)=T$
3180 T$=PS$(X):PS$(X)=PS$(U):PS$(U)=T$
3190 T%=SS%(X):SS%(X)=SS%(U):SS%(U)=T%:X=X-T
3200 LOOPWHILEX>0:NEXT:LOOP
3210 SCNCLR:RETURN
3220 :
```

Search Names

The search routine to bring up records based on names is really straight forward. It just uses a ‘for/next’ loop and the ‘instr()’ function. Once a record match is found, it displays the entire record on the screen. It also allows for auto-dialing the phone number associated with the record.

```
3230 REM --SEARCH NAME PROC --
3240 :
3250 Q$="":PRINT"ÿ   ENTER NAME TO SEARCH FOR:› ";
3260
DOUNTILQ$=CHR$(13)ORQ$=CHR$(27):PRINT"¤□";:GETKEYQ$:IF
Q$=CHR$(20)THENDR$=LEFT$(DR$,LEN(DR$)-
1):PRINTQ$;:Q$=""
3270
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)THENQ$=""
3280
IFQ$<>CHR$(13)THENBEGIN:PRINTQ$;:DR$=DR$+Q$:BEND:LOOP
3290 IFQ$=CHR$(27)THENSCNCLR:SCNCLR:MO=2:RETURN
3300
FORX=LTORN:IFINSTR(NA$(X),DR$)>0THENDR%=X:X=RN:FL%=1
3310 NEXT
3320 IFFL%=1THENGOSUB3360:Q$="":L=DR%+1:GOTO3300
3330 SCNCLR:MO=2:RETURN
3340 BEND
3350 :
3360 DO:SCNCLR:FL%=0:PRINT" RECORD NUMBER:-";DR%;"
ENTER -´D´  TO DIAL NUMBERÿ"
3370 CHAR,5,3,"ÿCUSTOMER NAME:
›"+NA$(DR%):CHAR,11,4,"ÿADDRESS:
›"+AD$(DR%):CHAR,6,5,"ÿPHONE NUMBER: ›"+PH$(DR%)
3380 CHAR,20,8,"ÿORDER TYPE:
›"+TY$(DR%):CHAR,19,9,"ÿPART NUMBER:
›"+PN$(DR%):CHAR,20,10,"ÿQTY & DESC:
›"+DE$(DR%):CHAR,23,11,"ÿ  PRICE: ›$"+PR$(DR%)
3390 CHAR,22,13,"ÿCOMMENTS: ›"+CO$(DR%)+"ÿ"
3400 CHAR,48,3,"ÿPACKING SLIP: ›"+PS$(DR%)
3410 CHAR,50,4,"ÿENTERED ON: ›"+LEFT$(ED$(DR%),2)+"-
"+MID$(ED$(DR%),3,2)+"-"+RIGHT$(ED$(DR%),2)
3420 CHAR,50,5,"ÿORDERED ON: ›"+LEFT$(OD$(DR%),2)+"-
"+MID$(OD$(DR%),3,2)+"-"+RIGHT$(OD$(DR%),2)
3430 IFSS%(DR%)=3THENCHAR,48,7,"ÿTHIS PART IS:› IN !!"
3440 IFSS%(DR%)=2THENCHAR,48,7,"ÿTHIS PART IS:› BACK
ORDERED."
3450 IFSS%(DR%)=1THENCHAR,48,7,"ÿTHIS PART IS:› NOT
AVAILABLE."
3460 IFSS%(DR%)=0THENCHAR,48,7,"ÿTHIS PART IS:› NOT
HEARD FROM."
3470 GETKEYQ$
3480 IFQ$="D"THENBEGIN:A$=MID$(PH$(DR%),1,3):B$=""
3490
IFA$<>"482"THENIFA$<>"486"THENIFA$<>"487"THENB$="1"
3500 PRINT#5,"ATDT"+B$+PH$(DR%):CHAR,25,22,"-
DIALING...ÿ":SLEEP1:PRINT#5,"ATM0":GETKEYQ$:BEND
3510 IFQ$=CHR$(27)THENDR%=RN:RETURN
3520 Q$="":RETURN
3530 :
```

Parts Order Menu

The parts ordering menu allows you to order any parts that are in the queue for the various vendors. The vendors are hard coded into the software unfortunately. In hind sight, I should have had this as an option, in a separate config menu, to allow adding or deleting vendors. But to be honest, the vendors never changed. After ordering, each record was marked as ordered and an order date of todays date was recorded for the corresponding record.

```
3540 REM -- PARTS ORDER PROC --
3550 :
3560
MO=1:Q$="E":DO:DOUNTILQ$=CHR$(13)ORQ$=CHR$(27)ORQ$="E"
:GETKEYQ$:IFQ$=" "THENMO=MO+1:EXIT
3570 IFQ$="´"THENMO=MO-1:EXIT
3580 IFQ$=CHR$(27)THENEXIT
3590 LOOP
3600 IFMO>8THENMO=1
3610 IFMO<1THENMO=8
3620 CHAR,41,3,"š  PARTS ORDER ´ÿ":CHAR,41,4,"
´":CHAR,41,13,"                  ´"
3630 FORX=4TO14:CHAR,54,X," □   ´ÿ":NEXT:CHAR,42,14," □
ÿ´"
3640 IFMO=1THENBEGIN:CHAR,41,5,"   ´KAWASAKI
´":CHAR,41,6,"  TRANS CYCLE ´":CHAR,41,7,"  SIMPLEX
´":CHAR,41,8,"  ARCTIC CAT   ´"
3650 CHAR,41,9,"  SUN & SNOW  ´":CHAR,41,10,"  KIMPEX
´":CHAR,41,11,"  FULL BORE   ´":CHAR,41,12,"
HUSQVARNA   ´":BEND
3660 IFMO=2THENCHAR,41,5,"  KAWASAKI
´":CHAR,41,6,"  ´TRANS CYCLE  ´":CHAR,41,7,"  SIMPLEX
´"
3670 IFMO=3THENCHAR,41,6,"  TRANS CYCLE
´":CHAR,41,7,"  ´SIMPLEX     ´":CHAR,41,8,"  ARCTIC
CAT  ´"
3680 IFMO=4THENCHAR,41,7,"  SIMPLEX
´":CHAR,41,8,"  ´ARCTIC CAT   ´":CHAR,41,9,"  SUN &
SNOW  ´"
3690 IFMO=5THENCHAR,41,8,"  ARCTIC CAT
´":CHAR,41,9,"  ´SUN & SNOW   ´":CHAR,41,10,"  KIMPEX
´"
3700 IFMO=6THENCHAR,41,9,"  SUN & SNOW
´":CHAR,41,10,"  ´KIMPEX       ´":CHAR,41,11,"  FULL
BORE   ´"
3710 IFMO=7THENCHAR,41,10,"  KIMPEX
´":CHAR,41,11,"  ´FULL BORE    ´":CHAR,41,12,"
HUSQVARNA    ´"
3720 IFMO=8THENCHAR,41,11,"  FULL BORE
´":CHAR,41,12,"  ´HUSQVARNA    ´":CHAR,41,5,"
KAWASAKI     ´"
3730 IFQ$=CHR$(27)ORQ$=CHR$(13)THENEXIT
3740 Q$="":LOOP
3750 IFQ$=CHR$(27)THENFORX=14TO3STEP-1:CHAR,41,X,"
":NEXT:MO=4:RETURN
3760 SCNCLR:L=0:Q$="":CHAR,1,3,"›PART
NUMBER":CHAR,20,3,"DESCRIPTION":CHAR,45,3,"COMMENTS"
3770 CHAR,1,4,"---------------------------------------
----------------------------------------ÿ"
3780 IFMO=1THENBEGIN:CHAR,5,1,"›DEALER NUMBER:ÿ 7299
›SHIP: ÿPARCEL POST SPECIAL DELIVERYÿ"
3790
FORX=1TORN:IFTYPE$(X)="KAWASAKI"ANDODATE$(X)="."THENBE
GIN:L=L+1:CHAR,1,L+4,LEFT$(PN$(X),18):CHAR,20,L+4,LEFT
$(DE$(X),24)
3800 CHAR,45,L+4,LEFT$(CO$(X),34):BEND
3810
IFL=15THENGETKEYQ$:WINDOW0,6,79,24:SCNCLR:WINDOW0,1,79
,24:L=1
3820 NEXT:BEND
3830 :
```

```
3840 IFMO=2THENBEGIN:CHAR,5,1,"›DEALER NUMBER:ÿ 71148
›SHIP: ÿPARCEL POST"
3850 FORX=1TORN:IFTYPE$(X)="TRANS
CYCLE"ANDOD$(X)="."THENBEGIN:L=L+1:CHAR,1,L+4,LEFT$(PN
$(X),18):CHAR,20,L+4,LEFT$(DE$(X),24)
3860 CHAR,45,L+4,LEFT$(CO$(X),35):BEND
3870
IFL=15THENGETKEYQ$:WINDOW0,6,79,24:SCNCLR:WINDOW0,1,79
,24:L=1
3880 NEXT:BEND
3890 :
3900 IFMO=3THENBEGIN:CHAR,5,1,"›DEALER NUMBER:ÿ XXXX
›SHIP: ÿPARCEL POST"
3910
FORX=1TORN:IFTYPE$(X)="SIMPLEX"ANDODATE$(X)="."THENBEG
IN:L=L+1:CHAR,1,L+4,LEFT$(PN$(X),18):CHAR,19,L+4,LEFT$
(DE$(X),24)
3920 CHAR,45,L+4,LEFT$(CO$(X),35):BEND
3930
IFL=15THENGETKEYQ$:WINDOW0,6,79,24:SCNCLR:WINDOW0,1,79
,24:L=1
3940 NEXT:BEND
3950 :
3960 IFMO=4THENBEGIN:CHAR,5,1,"›DEALER NUMBER:ÿ
5201905A            ›SHIP: ÿPURALATOR, NO INSURANCE."
3970 FORX=1TORN:IFTYPE$(X)="ARCTIC
CAT"ANDODATE$(X)="."THENBEGIN:L=L+1:CHAR,1,L+4,LEFT$(P
N$(X),18):CHAR,19,L+4,LEFT$(DE$(X),24)
3980 CHAR,45,L+4,LEFT$(CO$(X),35):BEND
3990
IFL=15THENGETKEYQ$:WINDOW0,6,79,24:SCNCLR:WINDOW0,1,79
,24:L=1
4000 NEXT:BEND
4010 :
4020 IFMO=5THENBEGIN:CHAR,5,1,"›DEALER NUMBER:ÿ XXXX
›SHIP: ÿBUS."
4030 FORX=1TORN:IFTYPE$(X)="SUN &
SNOW"ANDODATE$(X)="."THENBEGIN:L=L+1:CHAR,1,L+4,LEFT$(
PN$(X),18):CHAR,19,L+4,LEFT$(DE$(X),24)
4040 CHAR,45,L+4,LEFT$(CO$(X),35):BEND
4050
IFL=15THENGETKEYQ$:WINDOW0,6,79,24:SCNCLR:WINDOW0,1,79
,24:L=1
4060 NEXT:BEND
4070 :
4080 IFMO=6THENBEGIN:CHAR,5,1,"›DEALER NUMBER:ÿ 1514
›SHIP: ÿPARCEL POST."
4090
FORX=1TORN:IFTYPE$(X)="KIMPEX"ANDODATE$(X)="."THENBEGI
N:L=L+1:CHAR,1,L+4,LEFT$(PN$(X),18):CHAR,19,L+4,LEFT$(
DE$(X),24)
4100 CHAR,45,L+4,LEFT$(CO$(X),35):BEND
4110
IFL=15THENGETKEYQ$:WINDOW0,6,79,24:SCNCLR:WINDOW0,1,79
,24:L=1
4120 NEXT:BEND
4130 :
4140 IFMO=7THENBEGIN:CHAR,5,1,"›DEALER NUMBER:ÿ XXXXXX
›SHIP: ÿPARCEL POST."
4150 FORX=1TORN:IFTYPE$(X)="FULL
BORE"ANDODATE$(X)="."THENBEGIN:L=L+1:CHAR,1,L+4,LEFT$(
PN$(X),18):CHAR,19,L+4,LEFT$(DE$(X),24)
4160 CHAR,45,L+4,LEFT$(CO$(X),35):BEND
4170
IFL=15THENGETKEYQ$:WINDOW0,6,79,24:SCNCLR:WINDOW0,1,79
,24:L=1
4180 NEXT:BEND
4190 :
4200 IFMO=8THENBEGIN:CHAR,5,1,"›DEALER NUMBER:ÿ XXXX
›SHIP: ÿUNITED PARCEL SERVICE. (UPS)"
4210
```

```
FORX=1TORN:IFTYPE$(X)="HUSQVARNA"ANDODATE$(X)="."THENB
EGIN:L=L+1:CHAR,1,L+4,LEFT$(PN$(X),18):CHAR,19,L+4,LEF
T$(DE$(X),24)
4220 CHAR,45,L+4,LEFT$(CO$(X),35):BEND
4230
IFL=15THENGETKEYQ$:WINDOW0,6,79,24:SCNCLR:WINDOW0,1,79
,24:L=1
4240 NEXT:BEND
4250 :
4260 PRINT:PRINTTAB(37)" -END..."+CHR$(143):PRINT"    -
DO YOU WANT TO ORDER?":GETKEYQ$:IFQ$="Y"THENBEGIN
4270
IFMO=1THENBEGIN:FORX=1TORN:IFTYPE$(X)="KAWASAKI"ANDODA
TE$(X)="."THENODATE$(X)=TDATE$
4280 NEXT:BEND
4290 :
4300 IFMO=2THENBEGIN:FORX=1TORN:IFTYPE$(X)="TRANS
CYCLE"ANDODATE$(X)="."THENODATE$(X)=TDATE$
4310 NEXT:BEND
4320 :
4330
IFMO=3THENBEGIN:FORX=1TORN:IFTYPE$(X)="SIMPLEX"ANDODAT
E$(X)="."THENODATE$(X)=TDATE$
4340 NEXT:BEND
4350 :
4360 IFMO=4THENBEGINFORX=1TORN:IFTYPE$(X)="ARCTIC
CAT"ANDODATE$(X)="."THENODATE$(X)=TDATE$
4370 NEXT:BEND
4380 :
4390 IFMO=5THENBEGIN:FORX=1TORN:IFTYPE$(X)="SUN &
SNOW"ANDODATE$(X)="."THENODATE$(X)=TDATE$
4400 NEXT:BEND
4410 :
4420
IFMO=6THENBEGIN:FORX=1TORN:IFTYPE$(X)="KIMPEX"ANDODATE
$(X)="."THENODATE$(X)=TDATE$
4430 NEXT:BEND
4440 :
4450 IFMO=7THENBEGIN:FORX=1TORN:IFTYPE$(X)="FULL
BORE"ANDODATE$(X)="."THENODATE$(X)=TDATE$
4460 NEXT:BEND
4470 :
4480
IFMO=8THENBEGIN:FORX=1TORN:IFTYPE$(X)="HUSQVARNA"ANDOD
ATE$(X)="."THENODATE$(X)=TDATE$
4490 NEXT:BEND
4500 BEND
4510 :
4520 SCNCLR:MO=4:Q$="E":RETURN
4530 :
```

Parts Check Off

The CHECK OFF menu allows you to checkoff parts against the packing list of an order that has been received. Nothing too exciting here, just lots of for/next loops, and if statements. Its 131 lines of code.

```
4540 REM -- PARTS CHECK OFF --
4550 :
4560
MO=1:Q$="E":DO:DOUNTILQ$=CHR$(13)ORQ$=CHR$(27)ORQ$="E"
:GETKEYQ$:IFQ$=" "THENMO=MO+1:EXIT
4570 IFQ$="´"THENMO=MO-1:EXIT
4580 IFQ$=CHR$(27)THENEXIT
4590 LOOP
4600 IFMO>9THENMO=1
4610 IFMO<1THENMO=9
4620 CHAR,51,3,"š    CHECK OFF   ´ÿ":CHAR,51,4,"
´":CHAR,51,14,"                ´"
4630 FORX=4TO15:CHAR,64,X," □   ´ÿ":NEXT:CHAR,52,15," □
ÿ´"
4640 IFMO=1THENBEGIN:CHAR,51,5,"   ´KAWASAKI
´":CHAR,51,6,"  TRANS CYCLE ´":CHAR,51,7,"  SIMPLEX
´":CHAR,51,8,"  ARCTIC CAT   ´"
4650 CHAR,51,9,"  SUN & SNOW  ´":CHAR,51,10,"  KIMPEX
´":CHAR,51,11,"  FULL BORE   ´":CHAR,51,12,"
HUSQVARNA    ´":CHAR,51,13,"  BY PART #   ´":BEND
4660 IFMO=2THENCHAR,51,5,"  KAWASAKI
´":CHAR,51,6,"  ´TRANS CYCLE  ´":CHAR,51,7,"  SIMPLEX
´"
4670 IFMO=3THENCHAR,51,6,"  TRANS CYCLE
´":CHAR,51,7,"  ´SIMPLEX     ´":CHAR,51,8,"  ARCTIC
CAT  ´"
4680 IFMO=4THENCHAR,51,7,"  SIMPLEX
´":CHAR,51,8,"  ´ARCTIC CAT   ´":CHAR,51,9,"  SUN &
SNOW  ´"
4690 IFMO=5THENCHAR,51,8,"  ARCTIC CAT
´":CHAR,51,9,"  ´SUN & SNOW   ´":CHAR,51,10,"  KIMPEX
´"
4700 IFMO=6THENCHAR,51,9,"  SUN & SNOW
´":CHAR,51,10,"  ´KIMPEX       ´":CHAR,51,11,"  FULL
BORE   ´"
4710 IFMO=7THENCHAR,51,10,"  KIMPEX
´":CHAR,51,11,"  ´FULL BORE    ´":CHAR,51,12,"
HUSQVARNA   ´"
4720 IFMO=8THENCHAR,51,11,"  FULL BORE
´":CHAR,51,12,"  ´HUSQVARNA    ´":CHAR,51,13,"  BY
PART #   ´"
4730 IFMO=9THENCHAR,51,12,"  HUSQVARNA
´":CHAR,51,13,"  ´BY PART #    ´":CHAR,51,5,"
KAWASAKI    ´"
4740 IFQ$=CHR$(27)ORQ$=CHR$(13)THENEXIT
4750 Q$="":LOOP
4760 IFQ$=CHR$(27)THENFORX=15TO3STEP-1:CHAR,51,X,"
":NEXT:MO=5:RETURN
4770 :
4780 PS$=""
4790 IFMO<9THENBEGIN:SCNCLR
4800 CHAR,10,7,"ÿPACKING SLIP
NUM?:›":Q$="":DOUNTILQ$=CHR$(13):PRINT"¤□";:GETKEYQ$:I
FQ$=CHR$(20)ANDLEN(PS$)>0THENPS$=LEFT$(PS$,LEN(PS$)-
1):PRINTQ$;
4810 IFQ$=CHR$(20)THENQ$=""
4820 IFQ$=CHR$(20)ANDLEN(PS$)=0THENQ$=""
4830
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)THENQ$=""
4840
IFQ$<>CHR$(13)ANDASC(Q$)>47ANDASC(Q$)<58THENBEGIN:PRIN
TQ$;:PS$=PS$+Q$:BEND
4850
LOOPUNTILQ$=CHR$(27):IFQ$=CHR$(27)THENPS$="":MO=5:Q$="
```

```
E":SCNCLR:RETURN
4860 IFPS$=""THENPS$="-"
4870 :
4880 CD$="":CHAR,10,8,"ÿWHICH ORDER DATE?:›":CHAR,29,8
4890 FORX=1TO2
4900 GETKEYA$:IFASC(A$)<48ORASC(A$)>57THEN4900
4910 PRINTA$;
4920 GETKEYB$:IFASC(B$)<48ORASC(B$)>57THEN4920
4930 PRINTB$;"-";
4940 CD$=CD$+A$+B$
4950 NEXT
4960 GETKEYA$:IFASC(A$)<48ORASC(A$)>57THEN4960
4970 PRINTA$;
4980 GETKEYB$:IFASC(B$)<48ORASC(B$)>57THEN4980
4990 PRINTB$;
5000 CD$=CD$+A$+B$
5010 BEND
5020 :
5030 SCNCLR:CH=1
5040
IFMO=1THENBEGIN:FORX=1TORN:IFTY$(X)="KAWASAKI"ANDOD$(X
)=CD$THENCH%(CH)=X:CH=CH+1:PS$(X)=PS$
5050 NEXT:BEND
5060 :
5070 IFMO=2THENBEGIN:FORX=1TORN:IFTY$(X)="TRANS
CYCLE"ANDOD$(X)=CD$THENCH%(CH)=X:CH=CH+1:PS$(X)=PS$
5080 NEXT:BEND
5090 :
5100
IFMO=3THENBEGIN:FORX=1TORN:IFTY$(X)="SIMPLEX"ANDOD$(X)
=CD$THENCH%(CH)=X:CH=CH+1:PS$(X)=PS$
5110 NEXT:BEND
5120 :
5130 IFMO=4THENBEGIN:FORX=1TORN:IFTY$(X)="ARCTIC
CAT"ANDOD$(X)=CD$THENCH%(CH)=X:CH=CH+1:PS$(X)=PS$
5140 NEXT:BEND
5150 :
5160 IFMO=5THENBEGIN:FORX=1TORN:IFTY$(X)="SUN &
SNOW"ANDOD$(X)=CD$THENCH%(CH)=X:CH=CH+1:PS$(X)=PS$
5170 NEXT:BEND
5180 :
5190
IFMO=6THENBEGIN:FORX=1TORN:IFTY$(X)="KIMPEX"ANDOD$(X)=
CD$THENCH%(CH)=X:CH=CH+1:PS$(X)=PS$
5200 NEXT:BEND
5210 :
5220 IFMO=7THENBEGIN:FORX=1TORN:IFTY$(X)="FULL
BORE"ANDOD$(X)=CD$THENCH%(CH)=X:CH=CH+1:PS$(X)=PS$
5230 NEXT:BEND
5240 :
5250
IFMO=8THENBEGIN:FORX=1TORN:IFTY$(X)="HUSQVARNA"ANDOD$(
X)=CD$THENCH%(CH)=X:CH=CH+1:PS$(X)=PS$
5260 NEXT:BEND
5270 IFMO=9THENBEGIN:Q$="":PRINT"ÿ    ENTER PART NUMBER
TO SEARCH FOR:› ";
5280
DOUNTILQ$=CHR$(13)ORQ$=CHR$(27):PRINT"¤□";:GETKEYQ$:IF
Q$=CHR$(20)THENDR$=LEFT$(DR$,LEN(DR$)-
1):PRINTQ$;:Q$=""
5290
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)THENQ$=""
5300
IFQ$<>CHR$(13)THENBEGIN:PRINTQ$;:DR$=DR$+Q$:BEND:LOOP
5310 IFQ$=CHR$(27)THENSCNCLR:SCNCLR:MO=5:RETURN
5320
FORX=1TORN:IFINSTR(PN$(X),DR$)>0ANDOD$(X)<>"."THENCH%(
CH)=X:CH=CH+1
5330 NEXT
5340 BEND
5350 :
5360 IFCH=1ANDMO<9THENSCNCLR:CHAR,20,10,"-NOTHING WAS
```

```
ORDERED ON THIS DATE.ÿ":SLEEP1:SCNCLR:MO=5:RETURN
5370 IFCH=1ANDMO=9THENSCNCLR:CHAR,20,10,"-NO SUCH PART
NUMBER ON RECORD.ÿ":SLEEP1:SCNCLR:MO=5:RETURN
5380 FORX=1TOCH-1:SCNCLR:CHAR,20,5,"ÿ        NAME:
›"+NA$(CH%(X)):CHAR,20,6,"ÿ     ADDRESS:
›"+AD$(CH%(X)):CHAR,19,7,"ÿPHONE NUMBER: ›"+PH$(CH%(X))
5390 CHAR,20,8,"ÿ    COMMENTS: ›"+CO$(CH%(X))
5400 CHAR,20,11,"ÿPART NUMBER:
›"+PN$(CH%(X)):CHAR,20,12,"ÿDESCRIPTION:
›"+DE$(CH%(X)):CHAR,20,14,"-ENTER (I/B/N)"
5410
GETKEYQ$:IFQ$<>"I"THENIFQ$<>"B"THENIFQ$<>"N"THENIFQ$<>
CHR$(27)THENIFQ$<>"D"THEN5410
5420 IFQ$="I"THENSS%(CH%(X))=3
5430 IFQ$="B"THENSS%(CH%(X))=2
5440 IFQ$="N"THENSS%(CH%(X))=1
5450
IFQ$="D"THENBEGIN:IFLEFT$(PH$(CH%(X)),3)<>"482"ORLEFT$
(PH$(CH%(X)),3)<>"487"ORLEFT$(PH$(CH%(X)),3)<>"486"THE
NB$="1":ELSEB$=""
5460
PRINT#5,"ATDT"+B$+PH$(CH%(X)):SLEEP5:PRINT#5,"ATM0":GO
TO5410:BEND
5470 IFQ$=CHR$(27)THENX=CH
5480 NEXT
5490 SCNCLR:CHAR,10,5,"-DO YOU WANT TO PRINT
LABELS?ÿ":GETKEYQ$:IFQ$<>"Y"THENSCNCLR:MO=5:RETURN
5500 CHAR,10,5,"-PROCESSING                       ":OPEN4,4
5510
IFMO=1THENBEGIN:FORX=1TORN:IFTY$(X)="KAWASAKI"ANDSS%(X
)=3ANDOD$(X)=CD$THENBEGIN:PRINT#4,NA$(X):PRINT#4,PN$(X
):PRINT#4,DE$(X):PRINT#4,"$";PR$(X)
5520 PRINT#4,CO$(X):PRINT#4:BEND
5530 NEXT:BEND
5540 :
5550 IFMO=2THENBEGIN:FORX=1TORN:IFTY$(X)="TRANS
CYCLE"ANDSS%(X)=3ANDOD$(X)=CD$THENBEGIN:PRINT#4,NA$(X)
:PRINT#4,PN$(X):PRINT#4,DE$(X):PRINT#4,"$";PR$(X)
5560 PRINT#4,CO$(X):PRINT#4:BEND
5570 NEXT:BEND
5580 :
5590
IFMO=3THENBEGIN:FORX=1TORN:IFTY$(X)="SIMPLEX"ANDSS%(X)
=3ANDOD$(X)=CD$THENBEGIN:PRINT#4,NA$(X):PRINT#4,PN$(X)
:PRINT#4,DE$(X):PRINT#4,"$";PR$(X)
5600 PRINT#4,CO$(X):PRINT#4:BEND
5610 BEND
5620 :
5630 IFMO=4THENBEGIN:FORX=1TORN:IFTY$(X)="ARCTIC
CAT"ANDSS%(X)=3ANDOD$(X)=CD$THENBEGIN:PRINT#4,NA$(X):P
RINT#4,PN$(X):PRINT#4,DE$(X):PRINT#4,"$";PR$(X)
5640 PRINT#4,CO$(X):PRINT#4:BEND
5650 NEXT:BEND
5660 :
5670 IFMO=5THENBEGIN:FORX=1TORN:IFTY$(X)="SUN &
SNOW"ANDSS%(X)=3ANDOD$(X)=CD$THENBEGIN:PRINT#4,NA$(X):
PRINT#4,PN$(X):PRINT#4,DE$(X):PRINT#4,"$";PR$(X)
5680 PRINT#4,CO$(X):PRINT#4:BEND
5690 NEXT:BEND
5700 :
5710
IFMO=6THENBEGIN:FORX=1TORN:IFTY$(X)="KIMPEX"ANDSS%(X)=
3ANDOD$(X)=CD$THENBEGIN:PRINT#4,NA$(X):PRINT#4,PN$(X):
PRINT#4,DE$(X):PRINT#4,"$";PR$(X)
5720 PRINT#4,CO$(X):PRINT#4:BEND
5730 NEXT:BEND
5740 :
5750 IFMO=7THENBEGIN:FORX=1TORN:IFTY$(X)="LE
MANS"ANDSS%(X)=3ANDOD$(X)=CD$THENBEGIN:PRINT#4,NA$(X):
PRINT#4,PN$(X):PRINT#4,DE$(X):PRINT#4,"$";PR$(X)
5760 PRINT#4,CO$(X):PRINT#4:BEND
5770 NEXT:BEND
```

```
5780 :
5790
IFMO=8THENBEGIN:FORX=1TORN:IFTY$(X)="HUSQVARNA"ANDSS%(
X)=3ANDOD$(X)=CD$THENBEGIN:PRINT#4,NA$(X):PRINT#4,PN$(
X):PRINT#4,DE$(X):PRINT#4,"$";PR$(X)
5800 PRINT#4,CO$(X):PRINT#4:BEND
5810 NEXT:BEND
5820 :
5830
IFMO=9THENBEGIN:FORX=1TOCH:IFSS%(CH%(X))=3THENBEGIN:PR
INT#4,NA$(CH%(X)):PRINT#4,PN$(CH%(X)):PRINT#4,DE$(CH%(
X)):PRINT#4,"$";PR$(CH%(X))
5840 PRINT#4,CO$(CH%(X)):PRINT#4:BEND
5850 NEXT:BEND
5860 CLOSE4:SCNCLR:MO=5:RETURN
5870 :
```

Date Change

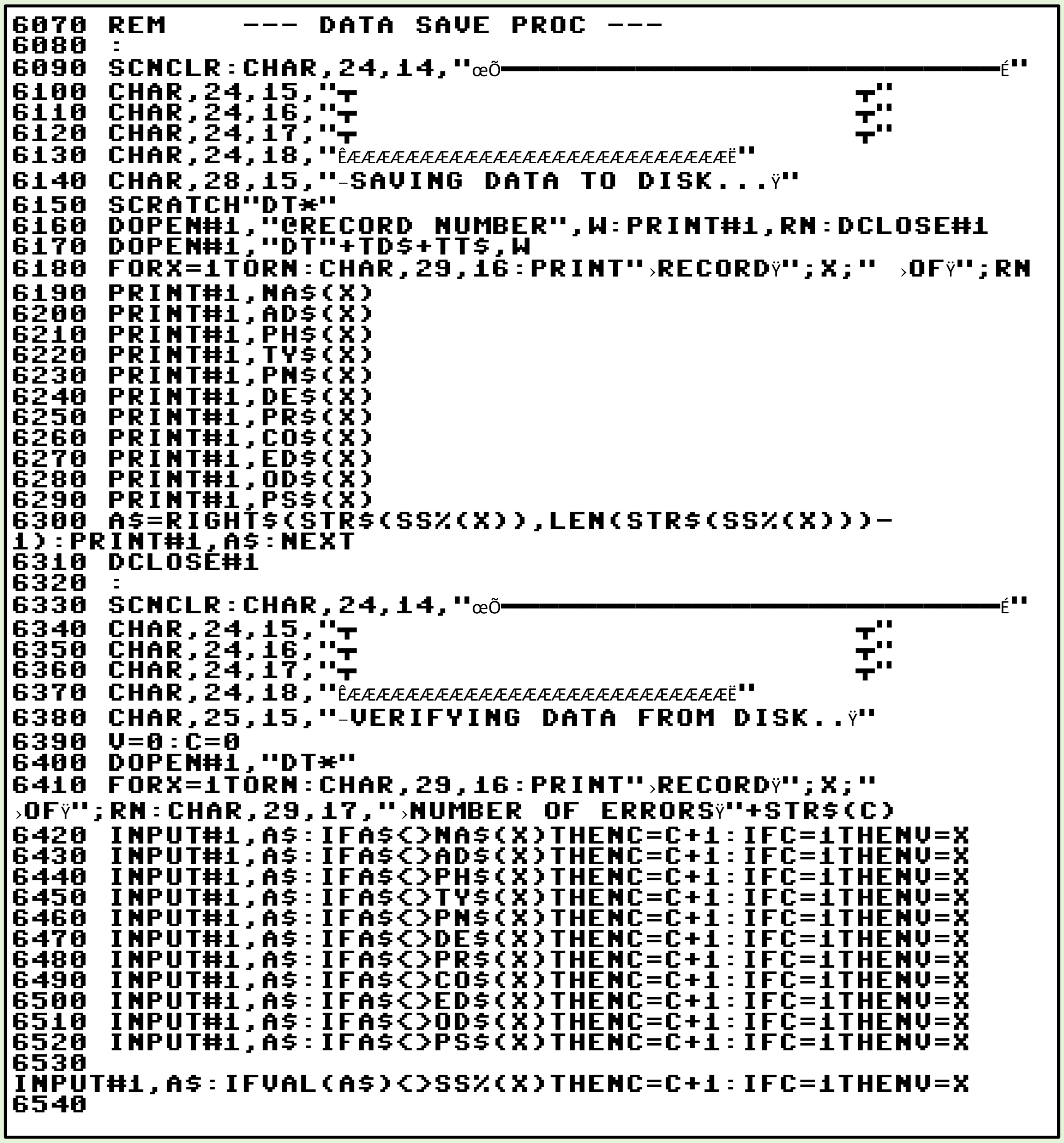
This routine is run when the main menu senses that the clock switches from 11:59pm to 12:00am and the date needs to be moved forward. It uses a lot of string manipulation using left$, right$, mid$ on the tdate$ variable.

```
5880 REM -- DATE CHANGE --
5890 :
5900
MM=VAL(LEFT$(TD$,2)):DD=VAL(MID$(TD$,3,2)):YY=VAL(RIGH
T$(TD$,2))
5910 DD=DD+1
5920 IFDD=29ANDMM=2THENMM=MM+1:DD=1
5930
IFDD=31THENIFMM=4ORMM=6ORMM=9ORMM=11THENMM=MM+1:DD=1
5940
IFDD=32THENIFMM=1ORMM=3ORMM=5ORMM=7ORMM=8ORMM=10ORMM=1
2THENMM=MM+1:DD=1
5950 IFMM=13THENYY=YY+1:MM=1:DD=1
5960 IFMM<10THENA$="0"+RIGHT$(STR$(MM),1):MM$=A$
5970 IFDD<10THENA$="0"+RIGHT$(STR$(DD),1):DD$=A$
5980 IFMM>9THENMM$=RIGHT$(STR$(MM),2)
5990 IFDD>9THENDD$=RIGHT$(STR$(DD),2)
6000 YY$=RIGHT$(STR$(YY),2)
6010 TD$=MM$+DD$+YY$:WINDOW0,0,79,24
6020 PRINT"  š    ORDER BOOK - BY GORD CLINK
TODAYS DATE:    ";LEFT$(TDATE$,2);"-
";MID$(TDATE$,3,2);"-";RIGHT$(TDATE$,2);"ÿ´"
6030
WINDOW0,1,79,24:SCNCLR:TI$="000600":GOSUB6070:IFC=0THE
NBEGIN:CHAR,30,18,"- INSERT BACK-UP
DISK!"+CHR$(143):GETKEYQ$:GOSUB6070
6040 CHAR,30,18,"- INSERT ORIGIONAL
DISK!"+CHR$(143):GETKEYQ$:BEND
6050 SCNCLR:Q$="E":RETURN
6060 :
```

Save Data Proc (includes Verify routine)

The data saving routine starts by displaying on the screen for the user that the system is saving data. It then deletes the main data file, and then over writes the 'record number' file with the current number of records in the system (using the @ feature). At this point the new data file is created with the current date and time embedded in the file name, and proceeds to write each record to the new data file, while at the same time updating the screen with the current progress.

After saving the data, it immediately does a verify of the data, to ensure that the data saved properly, and there are no disk issues. If the verify routine finds a discrepancy with a record in memory versus a record on disk, it will indicate that there is an error and display the record number effected.



```
6070 REM     --- DATA SAVE PROC ---
6080 :
6090 SCNCLR:CHAR,24,14,"œÕ────────────────────────────────────É"
6100 CHAR,24,15,"┬                            ┬"
6110 CHAR,24,16,"┬                            ┬"
6120 CHAR,24,17,"┬                            ┬"
6130 CHAR,24,18,"ÊÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆË"
6140 CHAR,28,15,"-SAVING DATA TO DISK...ÿ"
6150 SCRATCH"DT*"
6160 DOPEN#1,"@RECORD NUMBER",W:PRINT#1,RN:DCLOSE#1
6170 DOPEN#1,"DT"+TD$+TT$,W
6180 FORX=1TORN:CHAR,29,16:PRINT"›RECORDÿ";X;"  ›OFÿ";RN
6190 PRINT#1,NA$(X)
6200 PRINT#1,AD$(X)
6210 PRINT#1,PH$(X)
6220 PRINT#1,TY$(X)
6230 PRINT#1,PN$(X)
6240 PRINT#1,DE$(X)
6250 PRINT#1,PR$(X)
6260 PRINT#1,CO$(X)
6270 PRINT#1,ED$(X)
6280 PRINT#1,OD$(X)
6290 PRINT#1,PS$(X)
6300 A$=RIGHT$(STR$(SS%(X)),LEN(STR$(SS%(X)))-
1):PRINT#1,A$:NEXT
6310 DCLOSE#1
6320 :
6330 SCNCLR:CHAR,24,14,"œÕ────────────────────────────────────É"
6340 CHAR,24,15,"┬                            ┬"
6350 CHAR,24,16,"┬                            ┬"
6360 CHAR,24,17,"┬                            ┬"
6370 CHAR,24,18,"ÊÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆÆË"
6380 CHAR,25,15,"-VERIFYING DATA FROM DISK..ÿ"
6390 V=0:C=0
6400 DOPEN#1,"DT*"
6410 FORX=1TORN:CHAR,29,16:PRINT"›RECORDÿ";X;"
›OFÿ";RN:CHAR,29,17,"›NUMBER OF ERRORSÿ"+STR$(C)
6420 INPUT#1,A$:IFA$<>NA$(X)THENC=C+1:IFC=1THENV=X
6430 INPUT#1,A$:IFA$<>AD$(X)THENC=C+1:IFC=1THENV=X
6440 INPUT#1,A$:IFA$<>PH$(X)THENC=C+1:IFC=1THENV=X
6450 INPUT#1,A$:IFA$<>TY$(X)THENC=C+1:IFC=1THENV=X
6460 INPUT#1,A$:IFA$<>PN$(X)THENC=C+1:IFC=1THENV=X
6470 INPUT#1,A$:IFA$<>DE$(X)THENC=C+1:IFC=1THENV=X
6480 INPUT#1,A$:IFA$<>PR$(X)THENC=C+1:IFC=1THENV=X
6490 INPUT#1,A$:IFA$<>CO$(X)THENC=C+1:IFC=1THENV=X
6500 INPUT#1,A$:IFA$<>ED$(X)THENC=C+1:IFC=1THENV=X
6510 INPUT#1,A$:IFA$<>OD$(X)THENC=C+1:IFC=1THENV=X
6520 INPUT#1,A$:IFA$<>PS$(X)THENC=C+1:IFC=1THENV=X
6530
INPUT#1,A$:IFVAL(A$)<>SS%(X)THENC=C+1:IFC=1THENV=X
6540
```

```
NEXT:DCLOSE#1:IFC>0THENBEGIN:PRINT"" ":CHAR,25,15,"-"+
STR$(C)+" DATA VERIFY ERRORS":CHAR,26,16,"ÿCHECK
RECORD"+STR$(V):GETKEYQ$:BEND
6550 IFV=0THENSCNCLR:CHAR,25,15,"-VERIFYING
COMPLETE":SLEEP2
6560 SCNCLR:Q$="E"
6570 CLOSE5:OPEN5,2,0,CHR$(6)+CHR$(0):PRINT#5,"ATS0 =
0":PRINT#5,"ATM0"
6580 PRINT#5,"ATM0":RETURN
6590 :
```

Load Data Proc

Loading data works just like saving, but in reverse. It opens the ‘record number’ file and reads how many records to read. It uses this number in a for/next loop and then reads each record into memory one at a time. Once done, it simply goes back to the main menu.

```
6600 REM -- LOAD DATA PROC --
6610 :
6620 MS=FRE(1):SCNCLR:CHAR,25,15,"-LOADING DATA FROM
DISK.....ÿ"
6630 DOPEN#1,"RECORD
NUMBER":INPUT#1,RN:DCLOSE#1:IFRN=0THENSCNCLR:Q$="E":RE
TURN
6640 DOPEN#1,"DT*"
6650 FORX=1TORN:CHAR,25,16:PRINT"›RECORDÿ";X;" ›OFÿ";RN
6660 INPUT#1,NA$(X)
6670 INPUT#1,AD$(X)
6680 INPUT#1,PH$(X)
6690 INPUT#1,TY$(X)
6700 INPUT#1,PN$(X)
6710 INPUT#1,DE$(X)
6720 INPUT#1,PR$(X)
6730 INPUT#1,CO$(X)
6740 INPUT#1,ED$(X)
6750 INPUT#1,OD$(X)
6760 INPUT#1,PS$(X)
6770 INPUT#1,A$:SS%(X)=VAL(A$):NEXT
6780 DCLOSE#1
6790 DOPEN#1,"PHONE NUMBERS":SCNCLR:CHAR,25,15,"-
LOADING PHONE
NUMBERSÿ":FORX=1TO26:INPUT#1,BS$(X):INPUT#1,PP$(X):NEX
T
6800 DCLOSE#1:SCNCLR:Q$="E":RETURN
6810 :
```

Calculator Proc

The calculator routine pops up when you press ‘CMDR C’ at the main maneu. It is just for doing simple, addition, subtraction, multiplication and division. It will do 6 digits of precision after the decimal point, which is far more than needed.

```
6820 REM -- CALCULATOR PROC --
6830 :
6840 CHAR,35,5,"›                    ´"
6850 CHAR,35,6,"›    š            0›  □   ´"
6860 CHAR,35,7,"›                   □   ´"
6870 CHAR,35,8,"›  ££££££££££££££ □   ´"
6880 CHAR,35,9,"›  ´7    ´8    ´9    ´+    ´*   □   ´"
6890 CHAR,35,10,"›                   □   ´"
6900 CHAR,35,11,"›  ´4    ´5    ´6    ´-    ´/   □   ´"
6910 CHAR,35,12,"›                   □   ´"
6920 CHAR,35,13,"›  ´1    ´2    ´3    ´E       □   ´"
```

```
6930 CHAR,35,14,"›              ´N      □   ´"
6940 CHAR,35,15,"›   ´ 0      ´.   ´T      □   ´"
6950 CHAR,35,16,"›                       □   ´"
6960 CHAR,35,17,"□                          ´"
6970 :
6980 A$="":X=1
6990
B$="+":DO:TT$=LEFT$(TI$,2)+":":A=VAL(LEFT$(TT$,2)):IFA
>12THENA=A-
12:TT$=STR$(A)+":":IFLEN(TT$)>3THENTT$=RIGHT$(TT$,LEN(
TT$)-1)
7000 IFA=0THENTT$="12:"
7010 IFLEFT$(TI$,4)="2359"THENEXIT
7020 IFVAL(TI$)=0THENTT$="12:":GOSUB5880
7030 TT$=TT$+MID$(TI$,3,2)+":"+RIGHT$(TI$,2)
7040 CHAR,0,0:PRINT"´š     CALCULATOR
CURRENT TIME:    ";TT$;"ÿ´";
7050 GETQ$:IFQ$=CHR$(27)THENEXIT
7060 IFQ$=CHR$(13)THENBEGIN
7070 IFB$="+"THENTT=TT+VAL(A$):A$="":CHAR,37,6," š
´":CHAR,48-LEN(STR$(TT)),6," š"+STR$(TT):B$="+"
7080 IFB$="-"THENTT=TT-VAL(A$):A$="":CHAR,37,6," š
´":CHAR,48-LEN(STR$(TT)),6," š"+STR$(TT):B$="+"
7090 IFB$="*"THENTT=TT*VAL(A$):A$="":CHAR,37,6," š
´":CHAR,48-LEN(STR$(TT)),6," š"+STR$(TT):B$="+"
7100 IFB$="/"THENTT=TT/VAL(A$):A$="":CHAR,37,6," š
´":CHAR,48-LEN(STR$(TT)),6," š"+STR$(TT):B$="+"
7110 BEND
7120 IFQ$="C"THENA$="":TT=0:CHAR,37,6," š           0´"
7130 IFQ$="+"THENBEGIN:IFB$="+"THENTT=TT+VAL(A$):A$=""
7140 IFB$="-"THENTT=TT-VAL(A$):A$=""
7150 IFB$="*"THENTT=TT*VAL(A$):A$=""
7160 IFB$="/"THENTT=TT/VAL(A$):A$=""
7170 CHAR,37,6," š            ´":CHAR,48-
LEN(STR$(TT)),6," š"+STR$(TT):B$="+":BEND
7180 IFQ$="-"THENBEGIN:IFB$="-"THENTT=TT-VAL(A$):A$=""
7190 IFB$="+"THENTT=TT+VAL(A$):A$=""
7200 IFB$="*"THENTT=TT*VAL(A$):A$=""
7210 IFB$="/"THENTT=TT/VAL(A$):A$=""
7220 CHAR,37,6," š            ´":CHAR,48-
LEN(STR$(TT)),6," š"+STR$(TT):B$="-":BEND
7230 IFQ$="*"THENBEGIN:IFB$="+"THENTT=TT+VAL(A$):A$=""
7240 IFB$="-"THENTT=TT-VAL(A$):A$=""
7250 IFB$="*"THENTT=TT*VAL(A$):A$=""
7260 IFB$="/"THENTT=TT/VAL(A$):A$=""
7270 CHAR,37,6," š            ´":CHAR,48-
LEN(STR$(TT)),6," š"+STR$(TT):B$="*":BEND
7280 IFQ$="/"THENBEGIN:IFB$="-"THENTT=TT-VAL(A$):A$=""
7290 IFB$="+"THENTT=TT+VAL(A$):A$=""
7300 IFB$="*"THENTT=TT*VAL(A$):A$=""
7310 IFB$="/"THENTT=TT/VAL(A$):A$=""
7320 CHAR,37,6," š            ´":CHAR,48-
LEN(STR$(TT)),6," š"+STR$(TT):B$="/":BEND
7330
IFASC(Q$)>47ANDASC(Q$)<58ANDLEN(A$)<8ORQ$="."THENA$=A$
+Q$:CHAR,37,6," š            ´":CHAR,48-
LEN(A$),6," š"+A$
7340 LOOP
7350 FORX=17TO5STEP-1:CHAR,35,X,"
":NEXT
7360 A$="":TT=0:PRINT" ";:Q$="E":RETURN
7370 :
```

Help

The help screen would come up when you typed ‘CMDR H’. It was just a list of functions available with keyboard shortcuts that didn’t’ have a menu option.

```
7380 REM -- HELP SCREEN --
7390 :
7400
SCNCLR:FORX=1TO4:GETKEYQ$:NEXT:FORX=2TO17:CHAR,20,X,"›
                                                    ":NEXT
7410 CHAR,26,4," *** H E L P  S C R E E N **´"
7420 CHAR,23,6," CMDR & A ..............ALARM ON/OFF´"
7430 CHAR,23,7," CMDR & C .............CALCULATOR ON´"
7440 CHAR,23,8," CMDR & O ............PARTS TO ORDER´"
7450 CHAR,23,9," CMDR & P ............PHONE NUMBERS´"
7460 CHAR,23,10," CMDR & S ...............FORCED
SAVE´"
7470 CHAR,23,11," CMDR & ´ + ´...........ADVANCE
DATE´"
7480 CHAR,23,12," CMDR & ´ - ´..............BACK
DATE´"
7490 CHAR,23,13," SHFT & ´+´..........INCREASE
CLOCK´"
7500 CHAR,23,14," SHFT & ´+´..........DECREASE
CLOCK´"
7510 CHAR,23,15," SHFT & RESTORE..........SCREEN
DUMP´"
7520
Q$="":DOUNTILQ$=CHR$(27):GETQ$:IFLEFT$(TI$,4)="2359"TH
ENEXIT
7530 LOOP
7540 FORX=17TO2STEP-1:CHAR,20,X,"
":NEXT:SCNCLR:Q$="E":RETURN
```

Printer Routines *(should have been called Special)*

‘Printer Routines’ is simply the code for the ‘SPECIAL’ menu item. I’m guessing that I was originally going to call this menu ‘PRINT’, and then realized I need some things in there that didn’t have to do with printing, so I called it ‘SPECIAL’ instead.

```
7550 REM -- PRINTER ROUTINES --
7560 :
7570
MO=1:Q$="E":DO:DOUNTILQ$=CHR$(13)ORQ$=CHR$(27)ORQ$="E"
:GETKEYQ$:IFQ$=" "THENMO=MO+1:EXIT
7580 IFQ$="´"THENMO=MO-1:EXIT
7590 IFQ$=CHR$(27)THENEXIT
7600 LOOP
7610 IFMO>5THENMO=1
7620 IFMO<1THENMO=5
7630 CHAR,51,3,"š       SPECIAL       ´ÿ":CHAR,51,4,"
´":CHAR,51,10,"                      ´"
7640 FORX=4TO11:CHAR,70,X,"□    ´ÿ":NEXT:CHAR,52,11," □
´ÿ"
7650 IFMO=1THENBEGIN:CHAR,51,5,"   ´NOT HEARD FROM
´":CHAR,51,6,"  PARTS ORDERED     ´":CHAR,51,7,"
PARTS IN          ´":CHAR,51,8,"  PARTS BACK ORDERED´"
7660 CHAR,51,9,"  FILE TRANSFER      ´":BEND
7670 IFMO=2THENCHAR,51,5,"  NOT HEARD FROM
´":CHAR,51,6,"  ´PARTS ORDERED       ´":CHAR,51,7,"
PARTS IN          ´"
7680 IFMO=3THENCHAR,51,6,"  PARTS ORDERED
´":CHAR,51,7,"  ´PARTS IN           ´":CHAR,51,8,"
PARTS BACK ORDERED´"
7690 IFMO=4THENCHAR,51,7,"  PARTS IN
```

```
´":CHAR,51,8,"  ´PARTS BACK ORDERED´":CHAR,51,5,"  NOT
HEARD FROM    ´":CHAR,51,9,"  FILE TRANSFER      ´"
7700 IFMO=5THENCHAR,51,9,"  ´FILE TRANSFER
´":CHAR,51,8,"  PARTS BACK ORDERED´":CHAR,51,5,"  NOT
HEARD FROM  ´"
7710 IFQ$=CHR$(27)ORQ$=CHR$(13)THENEXIT
7720 Q$="":LOOP
7730 IFQ$=CHR$(27)THENFORX=11TO3STEP-1:CHAR,51,X,"
":NEXT:MO=6:RETURN
7740 :
```

All Data

This routine contains all of the code for each of the menu options in the 'SPECIAL' menu. It uses many BEGIN/BEND statements. The last menu option under 'SPECIAL' is the 'FILE TRANSFER' option which uses SuperSweep 128. Its interesting to see that I have it saving the data before it loads SuperSweep (as loading of SuperSweep would wipe the memory), but if the verify of the data is not successful, it will abort the loading of SuperSweep 128 and bring you back to the menu.

```
7750 REM -- ALL DATA --
7760 IFMO=1THENBEGIN
7770
MO=1:Q$="E":DO:DOUNTILQ$=CHR$(13)ORQ$=CHR$(27)ORQ$="E"
:GETKEYQ$:IFQ$=" "THENMO=MO+1:EXIT
7780 IFQ$="´"THENMO=MO-1:EXIT
7790 IFQ$=CHR$(27)THENEXIT
7800 LOOP
7810 IFMO>8THENMO=1
7820 IFMO<1THENMO=8
7830 CHAR,41,5,"š    PLACES     ´ÿ":CHAR,41,6,"
´":CHAR,41,15,"                 ´"
7840 FORX=6TO16:CHAR,54,X," □   ´ÿ":NEXT:CHAR,42,16," □
ÿ´"
7850 IFMO=1THENBEGIN:CHAR,41,7,"  ´KAWASAKI
´":CHAR,41,8,"  TRANS CYCLE ´":CHAR,41,9,"  SIMPLEX
´":CHAR,41,10,"  ARCTIC CAT   ´"
7860 CHAR,41,11,"  SUN & SNOW  ´":CHAR,41,12,"  KIMPEX
´":CHAR,41,13,"  FULL BORE    ´":CHAR,41,14,"
HUSQVARNA   ´":BEND
7870 IFMO=2THENCHAR,41,7,"  KAWASAKI
´":CHAR,41,8,"  ´TRANS CYCLE  ´":CHAR,41,9,"  SIMPLEX
´"
7880 IFMO=3THENCHAR,41,8,"  TRANS CYCLE
´":CHAR,41,9,"  ´SIMPLEX     ´":CHAR,41,10,"  ARCTIC
CAT  ´"
7890 IFMO=4THENCHAR,41,9,"  SIMPLEX
´":CHAR,41,10,"  ´ARCTIC CAT   ´":CHAR,41,11,"  SUN &
SNOW  ´"
7900 IFMO=5THENCHAR,41,10,"  ARCTIC CAT
´":CHAR,41,11,"  ´SUN & SNOW   ´":CHAR,41,12,"  KIMPEX
´"
7910 IFMO=6THENCHAR,41,11,"  SUN & SNOW
´":CHAR,41,12,"  ´KIMPEX       ´":CHAR,41,13,"  FULL
BORE    ´"
7920 IFMO=7THENCHAR,41,12,"  KIMPEX
´":CHAR,41,13,"  ´FULL BORE    ´":CHAR,41,14,"
HUSQVARNA   ´"
7930 IFMO=8THENCHAR,41,13,"  FULL BORE
´":CHAR,41,14,"  ´HUSQVARNA    ´":CHAR,41,7,"
KAWASAKI    ´"
7940 IFQ$=CHR$(27)ORQ$=CHR$(13)THENEXIT
```

```
7950 Q$="":LOOP
7960 IFQ$=CHR$(27)THENFORX=16TO3STEP-1:CHAR,41,X,"
":NEXT:MO=4:RETURN
7970 SCNCLR:PRINT" ÿ                               PARTS
NOT HEARD FROM"
7980 PRINT"CUSTOMER NAME        DATE        PART
NUMBER            DESCRIPTION"
7990 PRINT"-------------------------------------------------------
-------------------------------------
›":WINDOW0,5,79,24
8000 IFMO=1THENBEGIN
8010
FORX=1TORN:IFSS%(X)=0ANDTY$(X)="KAWASAKI"THENPRINTNA$(
X);:PRINTTAB(21);OD$(X);:PRINTTAB(33);PN$(X);:PRINTTAB
(56);LEFT$(DE$(X),23)
8020 NEXT:PRINTTAB(37)" -
END..."+CHR$(143):GETKEYA$:WINDOW0,1,79,24:SCNCLR:RETU
RN
8025 BEND
8030 :
8040 IFMO=2THENBEGIN
8050 FORX=1TORN:IFSS%(X)=0ANDTY$(X)="TRANS
CYCLE"THENPRINTNA$(X);:PRINTTAB(21);OD$(X);:PRINTTAB(3
3);PN$(X);:PRINTTAB(56);LEFT$(DE$(X),23)
8060 NEXT:PRINTTAB(37)" -
END..."+CHR$(143):GETKEYA$:WINDOW0,1,79,24:SCNCLR:RETU
RN
8070 BEND
8080 IFMO=3THENBEGIN
8090
FORX=1TORN:IFSS%(X)=0ANDTY$(X)="SIMPLEX"THENPRINTNA$(X
);:PRINTTAB(21);OD$(X);:PRINTTAB(33);PN$(X);:PRINTTAB(
56);LEFT$(DE$(X),23)
8100 NEXT:PRINTTAB(37)" -
END..."+CHR$(143):GETKEYA$:WINDOW0,1,79,24:SCNCLR:RETU
RN
8110 BEND
8120 IFMO=4THENBEGIN
8130 FORX=1TORN:IFSS%(X)=0ANDTY$(X)="ARCTIC
CAT"THENPRINTNA$(X);:PRINTTAB(21);OD$(X);:PRINTTAB(33)
;PN$(X);:PRINTTAB(56);LEFT$(DE$(X),23)
8140 NEXT:PRINTTAB(37)" -
END..."+CHR$(143):GETKEYA$:WINDOW0,1,79,24:SCNCLR:RETU
RN
8150 BEND
8160 IFMO=5THENBEGIN
8170 FORX=1TORN:IFSS%(X)=0ANDTY$(X)="SUN &
SNOW"THENPRINTNA$(X);:PRINTTAB(21);OD$(X);:PRINTTAB(33
);PN$(X);:PRINTTAB(56);LEFT$(DE$(X),23)
8180 NEXT:PRINTTAB(37)" -
END..."+CHR$(143):GETKEYA$:WINDOW0,1,79,24:SCNCLR:RETU
RN
8190 BEND
8200 IFMO=6THENBEGIN
8210
FORX=1TORN:IFSS%(X)=0ANDTY$(X)="KIMPEX"THENPRINTNA$(X)
;:PRINTTAB(21);OD$(X);:PRINTTAB(33);PN$(X);:PRINTTAB(5
6);LEFT$(DE$(X),23)
8220 NEXT:PRINTTAB(37)" -
END..."+CHR$(143):GETKEYA$:WINDOW0,1,79,24:SCNCLR:RETU
RN
8230 BEND
8231 IFMO=7THENBEGIN
8232 FORX=1TORN:IFSS%(X)=0ANDTY$(X)="FULL
BORE"THENPRINTNA$(X);:PRINTTAB(21);OD$(X);:PRINTTAB(33
);PN$(X);:PRINTTAB(56);LEFT$(DE$(X),23)
8233 NEXT:PRINTTAB(37)" -
END..."+CHR$(143):GETKEYA$:WINDOW0,1,79,24:SCNCLR:RETU
RN
8234 BEND
8235 IFMO=8THENBEGIN
8236
FORX=1TORN:IFSS%(X)=0ANDTY$(X)="HUSQVARNA"THENPRINTNA$
```

```
(X);:PRINTTAB(21);OD$(X);:PRINTTAB(33);PN$(X);:PRINTTA
B(56);LEFT$(DE$(X),23)
8237 NEXT:PRINTTAB(37)" -
END..."+CHR$(143):GETKEYA$:WINDOW0,1,79,24:SCNCLR:RETU
RN
8238 BEND
8239 MO=6
8240 BEND
8250 :
8260 REM -- PARTS ORDERED --
8270 IFMO=2THENBEGIN
8280 SCNCLR:CHAR,10,5,"-SCREEN OR PRINTER
(S/P)ÿ":GETKEYQ$:IFQ$="P"THENSCNCLR:CHAR,10,5,"-
PROCESSINGÿ":GOTO8380
8290 IFQ$=CHR$(27)THENSCNCLR:MO=6:RETURN
8300 SCNCLR:PRINT" ›                              PARTS
ORDERED ON ";LEFT$(TD$,2);"-";MID$(TD$,3,2);"-
";RIGHT$(TD$,2)
8310 PRINT"ORDER TYPE    PART NUMBER     DESCRIPTION
CUSTOMER NAME"
8320 PRINT"----------------------------------------------
----------------------------------
ÿ":WINDOW0,5,79,24
8330 FORX=RNTO1STEP-1:IFTD$=OD$(X)THENBEGIN
8340
PRINTTY$(X);:PRINTTAB(14)LEFT$(PN$(X),15);:PRINTTAB(30
);LEFT$(DE$(X),28);:PRINTTAB(60);LEFT$(NA$(X),20):BEND
8350 NEXT:PRINTTAB(37)" -END..."+CHR$(143)
8360 GETKEYQ$:WINDOW0,1,79,24:SCNCLR:MO=6:RETURN
8370 :
8380 OPEN4,4:CMD4:PRINT"                          PARTS
ORDERED ON ";LEFT$(TD$,2);"-";MID$(TD$,3,2);"-
";RIGHT$(TD$,2)
8390 PRINT:PRINT"ORDER TYPE    PART NUMBER
DESCRIPTION                CUSTOMER NAME"
8400 PRINT"----------------------------------------------
----------------------------------"
8410 FORX=RNTO1STEP-1:IFTD$=OD$(X)THENBEGIN
8420 PRINTTY$(X);:PRINTSPC(14-
LEN(TY$(X)));LEFT$(PN$(X),15);:PRINTSPC(1+(15-
LEN(LEFT$(PN$(X),15))));LEFT$(DE$(X),28);
8430 PRINTSPC(4+(25-
LEN(LEFT$(DE$(X),28))));LEFT$(NA$(X),20):BEND
8440 NEXT
8450 PRINT#4:CLOSE4:SCNCLR:Q$="":MO=6:RETURN
8460 BEND
8470 :
8480 REM -- PARTS IN --
8490 :
8500 IFMO=3THENBEGIN
8510 SCNCLR:CHAR,10,5,"-SCREEN OR PRINTER
(S/P)ÿ":GETKEYQ$:IFQ$="P"THENSCNCLR:CHAR,10,5,"-
PROCESSINGÿ"
8520 IFQ$="S"THENBEGIN:SCNCLR:PRINT" ›
PARTS THAT ARE IN"
8530 PRINT"CUSTOMER NAME        PART NUMBER
DESCRIPTION         PRICE"
8540 PRINT"----------------------------------------------
----------------------------------
ÿ":WINDOW0,5,79,24
8550 FORX=RNTO1STEP-
1:IFSS%(X)=3THENBEGIN:PRINTLEFT$(NA$(X),20);:PRINTTAB(
21)LEFT$(PN$(X),15);:PRINTTAB(37)LEFT$(DE$(X),20);:PRI
NTTAB(58)"$";PR$(X):BEND
8560 NEXT:PRINTTAB(37)" -END..."+CHR$(143)
8570 GETKEYQ$:BEND
8580 :
8590 IFQ$="P"THENBEGIN:OPEN4,4:CMD4
8600 SCNCLR:PRINT"                              PARTS
THAT ARE IN"
8610 PRINT:PRINT"CUSTOMER NAME          PART NUMBER
DESCRIPTION         PRICE"
```

```
8620 PRINT"-------------------------------------------
-------------------------------------"
8630 FORX=RNTO1STEP-
1:IFSS%(X)=3THENBEGIN:PRINTLEFT$(NA$(X),20);:PRINTSPC(
21-LEN(NA$(X)));LEFT$(PN$(X),15);:PRINTSPC(16-
LEN(LEFT$(PN$(X),15)));LEFT$(DE$(X),20);
8640 PRINTSPC(21-
LEN(LEFT$(DE$(X),20)));"$";PR$(X):BEND
8650 NEXT
8660 PRINT#4:CLOSE4:BEND
8670 WINDOW0,1,79,24:SCNCLR:MO=6:RETURN
8680 BEND
8690 :
8700 REM -- PARTS BACK ORDERED --
8710 :
8720 IFMO=4THENBEGIN
8730 SCNCLR:CHAR,10,5,"-SCREEN OR PRINTER
(S/P)ÿ":GETKEYQ$:IFQ$="P"THENSCNCLR:CHAR,10,5,"-
PROCESSINGÿ"
8740 IFQ$="S"THENBEGIN:SCNCLR:PRINT" ›
PARTS THAT ARE BACK ORDERED"
8750 PRINT"CUSTOMER NAME        PART NUMBER
DESCRIPTION          P.SLIP/DATE"
8760 PRINT"-------------------------------------------
-------------------------------------
ÿ":WINDOW0,5,79,24
8770 FORX=RNTO1STEP-
1:IFSS%(X)=2THENBEGIN:PRINTLEFT$(NA$(X),20);:PRINTTAB(
21)LEFT$(PN$(X),15);:PRINTTAB(37)LEFT$(DE$(X),20);:PRI
NTTAB(58)PS$(X);"/";OD$(X):BEND
8780 NEXT:PRINTTAB(37)" -END..."+CHR$(143)
8790 GETKEYQ$:BEND
8800 :
8810 IFQ$="P"THENBEGIN:OPEN4,4:CMD4
8820 SCNCLR:PRINT"                        PARTS THAT
ARE BACK ORDERED"
8830 PRINT:PRINT"CUSTOMER NAME        PART NUMBER
DESCRIPTION          COMMENTS"
8840 PRINT"-------------------------------------------
-------------------------------------"
8850 FORX=RNTO1STEP-
1:IFSS%(X)=2THENBEGIN:PRINTLEFT$(NA$(X),20);:PRINTSPC(
21-LEN(NA$(X)));LEFT$(PN$(X),15);:PRINTSPC(16-
LEN(LEFT$(PN$(X),15)));LEFT$(DE$(X),20);
8860 PRINTSPC(21-
LEN(LEFT$(DE$(X),20)));LEFT$(CO$(X),19):BEND
8870 NEXT
8880 PRINT#4:CLOSE4:BEND
8890 WINDOW0,1,79,24:SCNCLR:MO=6:RETURN:BEND
8900 :
8910
IFMO=5THENBEGIN:SCNCLR:GOSUB6070:IFV>0THENQ$="E":SCNCL
R:RETURN:ELSERUN"SUPERSWEEP 128":BEND
8920 :
```

Parts to Order

'Parts to Order' is the routine that updates the window that is on the main screen at all times that shows what vendors have parts waiting to be ordered. This window is refreshed when ever you return to the main menu, or if you press the 'E' key at the main menu. I'm unsure why I chose 'E' for this function.

```
8930 REM -- PARTS TO ORDER --
8940 :
8950 PRINT" š    PLEASE WAIT...´";
8960 CHAR,60,10,"š PARTS TO
ORDER›":FORX=11TO20:CHAR,60,X,"                      □
›":NEXT:CHAR,61,21,"□                        ›"
8970
FORX=1TORN:GETQ$:IFQ$="□"ORQ$=""ORQ$=CHR$(13)THENX=RN:
PRINT"  ÿ";:RETURN
8980
IFOD$(X)="."THENBEGIN:IFTY$(X)="KAWASAKI"THENCHAR,62,1
2,"KAWASAKI"
8990 IFTY$(X)="TRANS CYCLE"THENCHAR,62,13,"TRANS
CYCLE"
9000 IFTY$(X)="SIMPLEX"THENCHAR,62,14,"SIMPLEX"
9010 IFTY$(X)="ARCTIC CAT"THENCHAR,62,15,"ARCTIC CAT"
9020 IFTY$(X)="SUN & SNOW"THENCHAR,62,16,"SUN & SNOW"
9030 IFTY$(X)="KIMPEX"THENCHAR,62,17,"KIMPEX"
9040 IFTY$(X)="FULL BORE"THENCHAR,62,18,"FULL BORE"
9050 IFTY$(X)="HUSQVARNA"THENCHAR,62,19,"HUSQVARNA"
9060 BEND
9070 NEXT
9080 PRINT"´  ÿ";:RETURN
9090 :
```

Phone Numbers

Phone number routine displays the 26 quick dial phone directory on the screen, and allows you to dial the number automatically. It utilizes a modem connected to the user port for this purpose.

```
9100 REM --- PHONE NUMBERS ---
9110 :
9120 SCNCLR:CHAR,25,1," ***** PHONE NUMBERS *****ÿ"
9130 FORX=1TO13:CHAR,5,X+5,CHR$(64+X)+")
›"+BS$(X)+"ÿ":NEXT
9140 FORX=1TO13:CHAR,40,X+5,CHR$(64+X+13)+")
›"+BS$(X+13)+"ÿ":NEXT
9150 GETKEYQ$:PP%=ASC(Q$)-
64:IFQ$=CHR$(27)ORPP%<1ORPP%>26THENSCNCLR:Q$="E":RETUR
N
9160 IFBS$(PP%)="-"THENQ$="C":GOTO9230
9170 CHAR,15,20,"›"+PP$(PP%)+"     -(ÿD-)IAL OR (ÿC-
)HANGE?"
9180
GETKEYQ$::CHAR,35,20,"ÿ"+Q$+"ÿ":IFQ$=CHR$(27)THEN9210
9190 IFQ$<>"D"ANDQ$<>"C"THEN9180
9200
IFQ$="D"THENBEGIN:SCNCLR:PRINT#5,"ATDT"+PP$(PP%):SLEEP
5:PRINT#5,"ATM0"
9210 SCNCLR:Q$="E":RETURN:BEND
9220 :
9230
IFQ$="C"THENBEGIN:A??□ A(PP%):B$=PP$(PP%):BS$(PP%)="":
PP$(PP%)=""
```

```
9240 SCNCLR:CHAR,10,10,"ÿBUSINESS NAME:
›":WINDOW25,11,79,11
9250
Q$="":DOUNTILQ$=CHR$(13):PRINT"¤□";:GETKEYQ$:IFQ$=CHR$
(20)ANDLEN(BS$(PP%))>0THENBS$(PP%)=LEFT$(BS$(PP%),LEN(
BS$(PP%))-1):PRINTQ$;:Q$=""
9260 IFQ$=CHR$(20)ANDLEN(BS$(PP%))=0THENQ$=""
9270
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)THENQ$=""
9280
IFQ$<>CHR$(13)THENBEGIN:PRINTQ$;:BS$(PP%)=BS$(PP%)+Q$:
BEND
9290
LOOPUNTILQ$=CHR$(27):IFQ$=CHR$(27)THENBS$(PP%)=A$:PRIN
T" ";A$;
9300 WINDOW0,1,79,24:FORX=11TO5STEP-1:CHAR,60,X,"
":NEXT
9310 IFBS$(PP%)=""THENBS$(PP%)="-"
9320 :
9330 CHAR,10,11,"ÿ PHONE NUMBER:  ›":WINDOW25,12,79,12
9340
Q$="":DOUNTILQ$=CHR$(13):PRINT"¤□";:GETKEYQ$:IFQ$=CHR$
(20)ANDLEN(PP$(PP%))>0THENPP$(PP%)=LEFT$(PP$(PP%),LEN(
PP$(PP%))-1):PRINTQ$;:Q$=""
9350 IFQ$=CHR$(20)ANDLEN(PP$(PP%))=0THENQ$=""
9360
IFQ$=" "ORQ$="´"ORQ$=""ORQ$="□"ORQ$=" "ORQ$=""""ORQ$=CH
R$(9)ORQ$=CHR$(10)THENQ$=""
9370
IFQ$<>CHR$(13)THENBEGIN:PRINTQ$;:PP$(PP%)=PP$(PP%)+Q$:
BEND
9380
LOOPUNTILQ$=CHR$(27):IFQ$=CHR$(27)THENPP$(PP%)=B$:PRIN
T" ";B$;
```

Sort Procedure

SORT PROCEDURE is actually the sorting and saving of the phone directory. I'm not sure why I called it SORT PROCEDURE, and not DIRECTORY SORT or DIRECTORY SAVE. It does look like its uses the identical Shell Sort used to sort the regular data records.

```
9400 REM -- SORT PROCEDURE --
9410 WINDOW0,1,79,24:SCNCLR:CHAR,25,15,"-
SORTING...›":P=0
9420 T=1:DO:T=2*T:LOOPWHILET<25
9430 DO:T=INT(T/2):IFT=0THENEXIT
9440 FORI=1TO26-T:X=I
9450 DO:U=X+T:IFBS$(X)<=BS$(U)THENEXIT
9460 T$=BS$(X):BS$(X)=BS$(U):BS$(U)=T$
9470 T$=PP$(X):PP$(X)=PP$(U):PP$(U)=T$:X=X-T
9480 LOOPWHILEX>0:NEXT:LOOP
9490 :
9500
IFQ$<>CHR$(27)THENBEGIN:WINDOW0,1,79,24:SCNCLR:CHAR,25
,10,"-SAVING NUMBERS":DOPEN#1,"@PHONE
NUMBERS",W:FORX=1TO26:IFBS$(X)=""THENBS$(X)="-"
9510 IFPP$(X)=""THENPP$(X)="-"
9520 PRINT#1,BS$(X):PRINT#1,PP$(X):NEXT:DCLOSE#1
9530 WINDOW0,1,79,24:FORX=11TO5STEP-1:CHAR,60,X,"
":NEXT
9540 IFPP$(PP%)=""THENPP$(PP%)="-":BEND
9550 Q$="E":SCNCLR:RETURN
```

Error Trapping

I had a very minimal error trapping procedure to trap any programming errors I didn’t notice, and make it a little more graceful for the user. I seem to have it try to automatically resume it encounters a ‘break’ (error 30). If it encounters any other type of error, it displays the error message, the drive error condition, and then notifies them to contact Gord L. Clink. It then tries to resume after any key is pressed.

```
9560 REM -- ERROR TRAPPING --
9570 :
9580 IFER=30THENRESUME
9590 SCNCLR:PRINT"YOU HAVE A ";ERR$(ER);"IN
LINE";EL;"!!"
9600 PRINT"DISK DRIVE STATUS IS ";DS$;"!!"
9610 PRINT"IF THIS IS A PROBLEM, PLEASE CONTACT GORD
L. CLINK"
9620 GETQ$:RESUMENEXT

9630 :
```

Messages

Messages was a work in progress that was never finished. It was suppose to be a messaging system between staff members. This would have been useful for day staff leaving message for night staff, or vise versa. I’m not sure why I didn’t finish, it but its only ½ done by the looks of it.

```
9640 REM --- MESSAGES ---
9650
X=MO:MO=1:Q$="E":Z=VAL(MID$(TI$,3,2)):DO:DOUNTILQ$=CHR
$(13)ORQ$=CHR$(27)ORQ$="E":GETKEYQ$:IFQ$=" "THENMO=MO+
1:EXIT
9660 IFQ$="´"THENMO=MO-1:EXIT
9670 IFQ$=CHR$(27)THENEXIT
9680 Y=VAL(MID$(TI$,3,2)):IFY-Z>1THENQ$=CHR$(27):EXIT
9690 LOOP
9700 IFMO>4THENMO=1
9710 IFMO<1THENMO=4
9720 CHAR,27,7,"š       MESSAGES       ´ÿ":CHAR,27,8,"
´":CHAR,27,13,"                            ´"
9730 IFMO=1THENCHAR,27,9,"  ´READ MESSAGES
´":CHAR,27,10,"  WRITE MESSAGES      ´":CHAR,27,11,"
CHANGE EMPLOYEES   ´":CHAR,27,12,"  EXIT
´"
9740 IFMO=2THENCHAR,27,9,"  READ MESSAGES
´":CHAR,27,10,"  ´WRITE MESSAGES     ´":CHAR,27,11,"
CHANGE EMPLOYEES   ´"
9750 IFMO=3THENCHAR,27,10,"  WRITE MESSAGES
´":CHAR,27,11,"  ´CHANGE EMPLOYEES     ´":CHAR,27,12,"
EXIT               ´"
9760 IFMO=4THENCHAR,27,11,"  WRITE ??□ AGES
´":CHAR,27,12,"  ´EXIT                 ´":CHAR,27,9,"
READ MESSAGES      ´"
9770 IFQ$=CHR$(27)ORQ$=CHR$(13)THENEXIT
9780 Q$="":LOOP
9790 IFQ$=CHR$(27)ORMO=4THENFORX=13TO7STEP-
1:CHAR,27,X,"
":NEXT:MO=X:PRINT" ";:RETURN
9800 :
```

```
9810 FORX=13TO7STEP-1:CHAR,27,X,"
":NEXT
9820 IFMO<>4THENBEGIN
9830
EM=1:Q$="E":DO:DOUNTILQ$=CHR$(13)ORQ$=CHR$(27)ORQ$="E"
:GETKEYQ$:IFQ$=" "THENEM=EM+1:EXIT
9840 IFQ$="´"THENEM=EM-1:EXIT
9850 IFQ$=CHR$(27)THENEXIT
9860 LOOP
9870 IFEM>8THENEM=1
9880 IFEM<1THENEM=8
9890 CHAR,30,7,"š   EMPLOYEES   ´ÿ":CHAR,30,8,"
´":CHAR,30,17,"              ´"
9900 IFEM=1THENBEGIN:CHAR,30,9,"   ´DAVE
´":CHAR,30,10,"  PAUL          ´":CHAR,30,11,"  GORD
´":CHAR,30,12,"  LEONARD       ´"
9910 CHAR,30,13,"  CHADWICK     ´":CHAR,30,14,"  DANA
´":CHAR,30,15,"  SCOTT         ´":CHAR,30,16,"  EMPTY
´":BEND
9920 IFEM=2THENCHAR,30,9,"  DAVE
´":CHAR,30,10,"  ´PAUL          ´":CHAR,30,11,"  GORD
´"
9930 IFEM=3THENCHAR,30,10,"  PAUL
´":CHAR,30,11,"  ´GORD          ´":CHAR,30,12,"  LEONARD
´"
9940 IFEM=4THENCHAR,30,11,"  GORD
´":CHAR,30,12,"  ´LEONARD       ´":CHAR,30,13,"
CHADWICK     ´"
9950 IFEM=5THENCHAR,30,12,"  LEONARD
´":CHAR,30,13,"  ´CHADWICK      ´":CHAR,30,14,"  DANA
´"
9960 IFEM=6THENCHAR,30,13,"  CHAD
´":CHAR,30,14,"  ´DANA          ´":CHAR,30,15,"  SCOTT
´"
9970 IFEM=7THENCHAR,30,14,"  DANA
´":CHAR,30,15,"  ´SCOTT         ´":CHAR,30,16,"  EMPTY
´"
9980 IFEM=8THENCHAR,30,15,"  SCOTT
´":CHAR,30,16,"  ´EMPTY         ´":CHAR,30,9,"  DAVE
´"
9990 IFQ$=CHR$(27)ORQ$=CHR$(13)THENEXIT
10000 Q$="":LOOP
10010 IFQ$=CHR$(27)THENFORX=17TO7STEP-1:CHAR,30,X,"
":NEXT:MO=X:PRINT" ";:RETURN
10020 FORX=17TO7STEP-1:CHAR,30,X,"             ":NEXT
10030 :
10040 REM --- MESSAGE READ ROUTINE ---
10050 :
10060 IFMO=1THENBEGIN
10070 CHAR,23,17,"š        HIT ANY KEY FOR MORE
´ÿ"
10080 CHAR,23,5,"š           READ MESSAGE
´ÿ":WINDOW22,7,57,17,1:FORX=7TO16:PRINT"
 ":NEXT:WINDOW23,8,56,16
10090 CHAR,0,0:FORX=1TO10
10100 IFME$(EM,X)<>""THENPRINT" ";ME$(EM,X):GETKEYQ$
10110 NEXT
10120 WINDOW0,1,79,24,1:Q$="E":RETURN
10130 BEND
10140 :
10150 IFER=30THENRESUME
10160 IFER=5THENSCNCLR:Q$="E":RETURN
10170 IFER=20THENTT=0:RESUMENEXT
```

Conclusion

Many of these routines are quite large and after looking 30+ years later, they could have been written much more efficiently, and made into smaller routines. But of course, I know much more now than I did then, and to be fair, it worked and did exactly what it was suppose to. I suppose there is really no wrong way of doing something if the end result is what you wanted.

Gord L. Clink
Fort Frances, Ontario