RHB

the datac 1000 users' group

VOL.1 NO. 3
SEPT.-OCT. 1977

Editor: John Prenis

(Also layout and graphics design)

Contributors this issue:
Carmen DiCamillo, Robert Ferree, Rolland James, Gerry O'Halloran, Edwin R. Morris, Jack Wellin, John Prenis

Subscription rates are $5 for six issues. Letters, articles and news should be sent to John Prenis, 161 W. Penn St., Phila., Pa. 19144. You can make us very happy by typing them single spaced in columns 13.5 cm. (5.3 in.) wide.

The reason you've been waiting so long for the September issue is that we wanted to have the cassette interface in it. Despite a lot of hard work on the part of Carmen and Rolland, it's been taking a lot longer than anyone expected. Rather than delay any longer, we're putting the basic idea and the hardware part in this issue and saving the detailed software for next issue.

Like last year, the air conditioning didn't always work properly, but that didn't stop people from having a great time at Atlantic City. This was the fourth computerfest I'd attended, but the first I've seen from the other side of the exhibitor's booth. A lot of work goes into planning an exhibit. In the press of time many elaborate plans have to be abandoned. It's a severe test of equipment, too. I won't forget the TTY that wouldn't work, standing silent in a corner, or trying to write a demo program on the spot. You can expect that anything that can go wrong will, and at precisely the most embarrassing moment. I suffered along with the speaker whose cassette wouldn't load, and with another whose tape got wiped partway through his talk. I was very glad that our music program was on ROM! While I missed most of the talks, I got a lot out of the ones I did attend. Would I do it again? Sure!

Now, does anybody out there want to do a report on Boston?

As a gesture of appreciation, we have decided to present contributors to the newsletter with credit certificates good toward future purchases of Datac products. The actual amount will depend on length and content, at the discretion of the editor, but a typical article will probably net its writer a $5 certificate.

This will be the last issue of the Users Group Newsletter to be distributed free. If you want to get full use out of your Datac equipment, don't lose any time in sending in your five bucks.

John Prenis

The Cassette Interface

by Carmen DiCamillo and Rolland James

This article will begin a series on using the cassette interface on the DATAC 1000 card.

In this first part, we will discuss the hardware aspects of the interface and also consider the software needed for read and write operation. Future articles in this series will provide program listings and also the information needed for proper adjustment of cassette recorder controls.

We will discuss the basic operation of the cassette interface, as we indicate the needed parts for operation.

The 6850 ACIA (U-23) receives parallel data from the data bus under uP control, and transforms this into serial data. The speed at which this serialization takes place is controlled by the transmit clock. The transmit clock is produced by counting down the systems Ø2 clock. This is done by CD4059 divider (U-26), which must have its inputs set, to divide by the appropriate number.

The serial data output from the ACIA is then fed to a resistive divider, which then is applied to the MIC of AUX input of a cassette recorder.

Data originally applied to the tape recorder input (fig #1A) will, upon playback, appear as a series of positive and negative going spikes (fig #1B). To restore this data to its original

REMEMBER, THIS WILL BE YOUR LAST ISSUE UNLESS YOU HAVE PAID YOUR $5.00 FOR A SUBSCRIPTION.

form we apply it to a pair of voltage comparators at LM339 (U-22). One of the comparators is set up to respond to only positive going spikes (fig #1C). The other voltage comparator is set up to respond only to negative going spikes (fig #1D). Applying the output of first comparator to the set input of an R-S flip flop, and applying the output of the second comparator to the reset input of the flip-flop - we get our reconstructed data which is then applied to the ACIA for conversion back to parallel form.

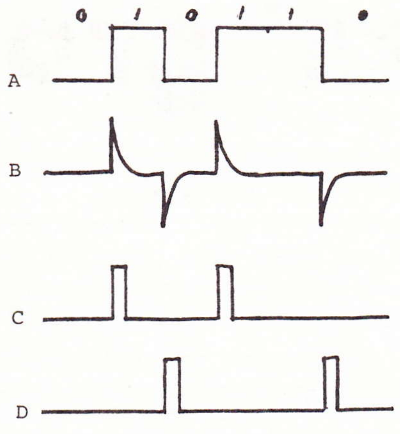


Fig #1

- Tape Format

We will now outline the format for writing data on tape. (refer to fig #2)

To begin with, there is a header. This indicates to the read program that there is data coming.

Next there is an identification number (ID) between ØØ - FF hex, which is user selected. The ID is very useful for storing several programs on one cassette. Upon playback, the ID is checked to see if that is the one you want. If it is,the data is read; if it is not correct, the data is ignored.

Next on the tape is the starting address of where the data is to be written.

After the starting address comes the block length. The block length indicates the total number of bytes in the record. We set a limit of 256 bytes per record. If the data you wish to record has more than 256 bytes in it, the cassette write program will automatically write the number of blocks needed.

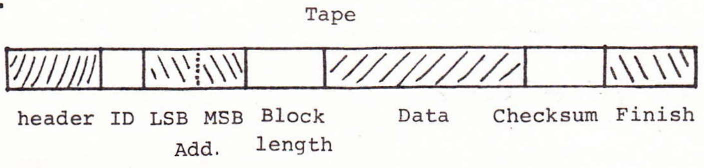


Fig #2

After the block length we finally have our data. At the end of our data there is a checksum. The checksum is the sum of all the data in the block. Its purpose is for error detection. Upon playback, the sum of all the data is taken; this is compared with the checksum written on the tape. If the two are not equal there is obviously an error and the tape must be replayed.

After the checksum there is a finish block indicator.

If there is another block of data we go through the whole format again.

- Population

In order to populate the cassette interface the following components will be needed:

U-23	6850	ACIA
U-26	4059	Divider
U-22	LM339	Comparator
U-13	7402	NOR Gate

R44	10K	R40	5K
R45	10K	R46	150
R41	10K	R47	10K
R42	10K	R37	1K
R43	10K	R38	1K
R39	5K	C4	5 mf

- The Clock

In our discussion of basic operation of the interface it was pointed out that we counted down the system clock by a divider chip (U-26) and applied this to the ACIA (U-23). Since the serialization is based upon this clock, it is recommended that it be crystal controlled. To do this, one needs only to place a 1MHZ crystal in the space provided below the microprocessor and place a 330K ohm resistor in place of the present R8. Now that we have a stable system clock input to our divider chip, we can jumper select the proper divisor.

We have selected the number "20" and suggest for the sake of standard analysis that everyone use this number.

To program the divider chip to divide by this number, simply connect the following pins of U-26 to either VCC of GND.

Pin #							
3	GND	22	GND	18	GND	10	GND
4	GND	21	GND	17	GND	9	GND
5	GND	20	+5V	16	GND	8	GND
6	GND	19	GND	15	GND	7	GND
11	+5V	13	GND	14	+5V		

The speed at which our cassette interface will operate will be the system clock frequency divided by <u>20</u> and finally divided by 16(which is done under program control by the ACIA) rendering us the rate of <u>2500</u> bits per second.

$1.000 \text{ MHZ} \div \underline{20} \div 16 = 2500$ bits/second

- The ACIA

Enough said about clocks - let's move on to the ACIA.

The ACIA is decoded by means of pin #9 which is a $\overline{\text{CS}}$. This pin is brought over to the decoder chips. We recommend decoding the ACIA at the address 14ØØ hex. This can easily be accomplished by placing a jumper from U10 pin 10 to just to the right of that chip, to a pad labeled ACIA. We now have the ACIA decoded. (Detailed instructions on installing the decoder chips U9, U10, U11 can be found in the newsletter vol1 #1 & 2.)

- The Comparators

The placing of the components around the comparator should be self-explanatory. When placing the LM339 (U-22) you must also install the parts for the power on reset (refer to newsletter vol 1 #1). Since the power on reset makes use of 1/4 of the LM339, if none of the parts for the reset are installed the ready line will be constantly held low, thus preventing the microprocessor from running. If the parts are installed, upon power-up, the reset line will go low and then return to its normal high state. This will not effect operation of the board if the diode (D1) between reset and run is disconnected.

Please note R39 and R40 should be placed to +5V; the silk screening on the PC board may lead you to believe that these resistors should go to GND - this is not correct.

Important EPROM Tip

By Rolland T. James

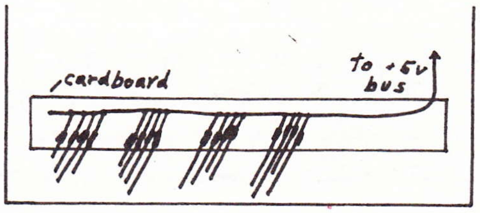
It has come to our attention in a most graphic and expensive manner that the popular 2708 EPROM used on the DATAC 1000 has a permanent failure mode that can be very easily stimulated. The means of producing this dastardly situation is to allow the VDD or VCC pins to drop lower than the VBB pin. Intel puts it this way: Apply VBB first and remove VBB last. How can this happen when VDD is +12, VCC is +5, and VBB is -5? Ask yourself, "Self, do they always sit there steady for all time?" The answer of course is no, you turn them on and you turn them off.

But what can be done? One thing you can do is put a couple of diodes on the VBB line. One to VCC and one to VDD. This should clamp any baddies before they cause the 2708 to commit hibachi. You have a brighter solution? Send it in.

ELIMINATING STAND UP RESISTORS

by Jack Wellin

The pull up resistors on my board were causing me grief, so I replaced them. The old resistors were cut off the board, and the now accessible feed through holes were soldered on both sides. Using the holes on the switch pads, a new set of resistors was installed on the back side of the board. A strip of cardboard prevents shorts. The end of the bus wire is covered with sleeving and soldered to the +5 volt bus.



Good Programming Practices

by John Prenis
from the Datac 1000 Tutorial Manual

Last time we discussed top-down design, modular programming, and structured programming. This time we will give examples of the structures and show how to implement them in machine language.

The use of subroutine calls or procedures is strongly encouraged by the top-down approach. The subroutine call then, is an important structure.

(The main difference between a subroutine and a procedure is that a subroutine is intended to be executed many times, a procedure only once.) TLC does not have an explicit subroutine call. Instead, any name which appears on a line by itself is automatically a call to the subroutine of that name. Like this:

```
    DO:THIS
```

The computer goes to the routine called DO:THIS, executes it, and returns to the next statement following the original call. Of course, somewhere there has to be a routine DO:THIS and it will look something like this:

```
  $ DO:THIS
    ....
    ....
    RTN
```

Matters are almost as simple in 6502 machine language:

```
1100    20   56   30    go to subroutine at 3056
```

The routine itself would look something like this:

```
3056    ( )( )
3058    ( )
3059    ( )( )( )       do routine
305C    60              return
```

Probably the most important structure is the conditional or "if-then-else" structure, used for making decisions. Here's one written in TLC.

```
    IF R=5
        DO:THIS
    ELSE
        DO:THIS:OTHER:THING
    FI
```

DO:THIS and DO:THIS:OTHER:THING are the names of subroutines. When the program has finished one or the other, it goes to the statement following the FI ("IF" spelled backwards). Notice the indentations, used to make the logical structure stand out clearly. The use of FI together with indentation helps to make things clearer when nested IF's are used.

Here is how a similar decision might be performed in 6502 machine language:

```
2010    A5   03          load accumulator with R
2012    C9   05          compare with 5
2014    F0   06          if R=5, then go to A
2016    20   ( )( )      else call DO:THIS:OTHER:THING
2019    4C   1F   20     go to B
201C    20   ( )( )  A   call DO:THIS
201F    ....         B   program continues
```

We were forced to use one go-to, but the principle of "one entrance,one exit" is maintained.

Next time, the "do-while"

ANNOUNCEMENTS

Jack Wellin (DA4-4751) would like to know if anyone can suggest ways to hook up a calculator to the board so as to use its keyboard and display.

Karl Amatneek, chairman of the Philadelphia section of the IEEE Update Committee, has just returned from a trip to Yugoslavia, where he taught a course on bench programming using the Datac board.

ERRATA

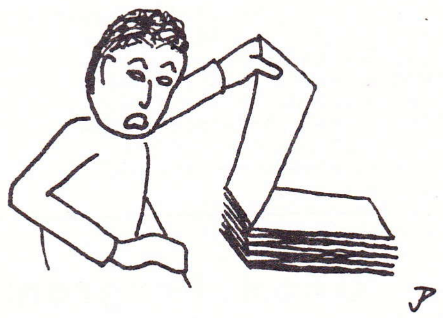
Page 7, August: Address 0000 should have an asterisk beside it. 0000 contains the low byte of the address, 0001 the high byte. 000B should be C0, not CD and 0010 should be C7, not E7. The addresses following, 0019 should be 001C and 001D. It really does work folks, honest! Board changes, p. 7, August: R37 and R38 are pull downs. There is no need to touch your board. This bit of confusion was caused by the fact that R37 and R38 are incorrectly shown as pull ups on the schematic.

IEEE UPDATE

And don't forget the IEEE µP I/O DEMO -

Thursday, November 17th, 1977
6-9 PM
Bell of PA Bldg -18th Floor
Arch St bet 15th and 16th

Ever stop to think that it would take 200 human mathematicians working for 40 years to produce a mistake like this?



DATAC ENGINEERING
P.O. BOX 406
SOUTHAMPTON, PA. 18966

FIRST CLASS

DID YOU REMEMBER TO SUBSCRIBE?