CLOAD
MAGAZINE, inc
BOX 1267
GOLETA, CA
93017

April in April,

Or, possibly April Fool? Here we are again - this makes six issues in three months, Guinness Book of World records, take note! I'd like to start out by sharing with you some of the feedback that our renewal subscribers are providing. Most renewal notes come with a check enclosed. This is quiet confirmation that the work we are doing is worthwhile, and it pleases our creditors, too. Some of the renewals come in with a letter written to us directly from the subscriber's TRS-80. This also pleases us, as the letters indicate that the TRS-80's themselves are happy. We get a few notes from the human element, too. These break down into three categories: Category one is the note telling us to continue in the groove we're in. Category two is the note telling us to get out of the rut we've fallen into. Category three is the note with about five cassettes enclosed, asking that they be re-recorded in our new format.

I'd like to speak further on category one, but I can't think of any properly inane statements on a point so obviously well thought out. Therefore, let's cover categories two and three instead. Category two notes are usually levelled at our preponderance of games, raising the age-old debate of (a) what is a game, and (b) what is a game good for? A side point is usually made that the TRS-80 was purchased for business, not games.

Well, the cop-out answer that would be so easy to say is that we publish what is submitted, and what is submitted to us is games. There are deeper reasons, however. Business software has several points that make it unsuitable for publication even if it was submitted. It is software that has to be excruciatingly well thought out and exceptionally well written. If it is not, it can put a business in deep trouble in short order. If the software passes this requirement, it must then also be of general interest. Examples might be payroll calculation, where the problem is relatively computation oriented instead of data oriented. The breakdown here is that there is no uniform problem setup. There are different taxes for different categories of different people in different states, counties and cities, belonging to different unions, trade associations and pension plans. A program covering one set of conditions would be useless to anyone else. A program covering all of them would be too unwieldy to write for publication, or would require the traditional "light editing" that we abhor. If we leave this example, we run into processes like bookkeeping and general accounting, which are even more complex and unwieldy. Let's say, though, that a program comes along that meets all the above requirements. How do we maintain it? Bugs must be taken care of on a continuing basis, and documentation of the order of a small book must be written and furnished to the user. A service of question answering must be provided, and we would have to troubleshoot problems that are encountered. We had a program called "loan" in one of our first issues. It was one of the most practical programs that we've published, and one of the most troublesome - we fielded complaints that $1.00 at 3 per cent interest for 300 years gave a wrong answer. We were told that it computed mortgage payments that were 40 cents off out of a mortgage payment in the $600.00 range. We were told that the algorithm was not the proper one for a certain type of loan. How do we reply to these, except to say that all of these statements are true, and that the program was intended for financial planning purposes, not checking up on the loan officer at the local bank.

Let's look at another aspect of the business software problem. Say that someone has come up with a system that addresses all these problems. That system is worth thousands of dollars, and many, many copies must be sold to recoup the programmer's costs. The only people that can deal in those numbers are the cats at Tandy Center, and that is one reason that their software is the only non-custom stuff around (and those who think that $1ØØ is too expensive for "just a cassette" are not looking at the product, they're looking at the media - a hundred dollar bill is printed on a scrap of paper, and can also be copied).

Then there's the old piracy problem. If you don't have a raft of lawyers on retainer, you can't protect your software except by making it too cheap to copy, and for significant level stuff, the numbers just don't work out. We are being widely pirated ourselves, and we're $3.ØØ a copy!

Then there's the old inverse piracy problem. On three occasions, with varying degrees of culpability, we have published material that didn't belong to us. We do not treat this situation lightly. If someone submits a significant package to us that we can afford to buy rights to, we need to do more than a cursory amount of "checking around" to assure that we aren't becoming party to theft (the height of absurdity occurs at regular intervals, when we are offered our own material, some of which we wrote "in - house"!).

All this, plus more that I'll spare you, points away from business software, though if the situation somehow eases we will be more than willing to give it a try. Some non-games coming up include a general system test program that I'm writing in my "spare" time. It is written in assembly code, and it is as rough as I can make it - so rough that I only work on the code before my first cup of coffee in the morning. Also slated for inclusion is a program which tells you how far and in which direction any point on the planet is from any other point. This is useful to people who plan looong trips, civil engineers who wish to set up informative signposts, and anarchists who wish to aim their Army surplus ICBM's accurately.

Let's back up to page one and cover the third category of renewal statement - that of re-recording our old issues in our new format. As in the case of category two, there's more trouble than meets the eye. Let me repeat our policy on returns to our new subscribers: any tape which is unloadable will be immediately replaced upon return (please, within about three months). Since they are sent First Class, any tapes received in a damaged condition can be returned without additional postage - just write "refused - damaged" on the envelope. Unfortunately, all we have to replace our earlier issues with are more of the same (they really do load, just not as nicely). Most of our early issues would need to be reworked before re-recording to remove bugs, to remove the three unowned programs mentioned above, and to remove some programs which we have subsequently released rights to. They would also need to be reshuffled to allow the level I and level II versions of the same program to exist on opposite sides of the same tape.

To do all this would require that we come out with a "best of CLOAD" issue, which we have done. The "Best of CLOAD, Volume I", or "six months of trying" has been mastered for about five months now, and we will get it published as soon as our regular issues are caught up (you've heard that song before, haven't you?). Fact is, we're about due for Volume II (as soon as Volume I is out...).

Announcements:

We have had a request to recommend a book for beginners that covers the subject of learning assembly language. A common observation is that most books covering the subject stress the architecture of the central processor (in this case, the Z-80 chip), and the instruction set. Nobody writing at the entry level seems to address the structure of an assembly language program as such. Well, the best book in this area that I have run across is the Z-80 microcomputer handbook, by William Bardon, Jr. It is published by Howard W. Sams & Co., and should be available at a local bookstore or electronics store, though it is apparently not standard Radio Shack stock (watch me get called on this). The book is devoted to both hardware/architecture, software, and systems - spending about 100 pages on each. It is written in a form that is intelligible to mere mortals, being a very straightforward dialect of technish written with a Southern California accent.

There is a list on one of the back pages which is a breakdown of seconds of playing time, the turns count of a CTR-41, and the turns count of a CTR-80. With it, any one value can be converted to any other. As one of our distinguished authors (C. W. Evans of Sun City, AZ) has pointed out, the turns count is proportional, the CTR-80 being about .59 times the CTR-41. He has submitted a program to compute this which (space permitting) we will publish. The values in the list were obtained in a way more appropriate to our mental abilities, however. We measured them with a stopwatch.

The author of the April Fool program recommends the following sequence of commands while showing it to friends: RUN, LIST 40, LIST, BREAK, RUN, NEW, BREAK, RUN, EDIT 40, BREAK, LIST 40, <*>, RUN, TRON, RUN, TROFF, RUN, CLOAD, HELP. The <*> is an "ENTER" if you are running level I, ignore it if you are running level II. Note that the level I version uses level II commands, and that they both include "HELP", a command on larger timeshare systems. This is an example of a large group of practical jokes which are played on larger computer systems, which include fake login procedures, mystery messages, bombs (programs to "crash" the host computer, and the like.

Hardware:

In our continuing series on controlling the world with a TRS-80, Let's look at the architecture of the 8255 I/O chip. It is set up internally as four ports, which in our particular lashup are 128, 129, 130 and 131 decimal (80,81,82 & 83 hex). We will henceforth refer to them as A, B, C and Control, respectively. We will also use mode 0 exclusively at first. This is the mode where ports A, B and C act as simple input or output ports, depending on the byte that has been output to the control port. The chart (figure 1, back page) has a list of bytes that can be sent to the control port, along with the resulting configuration in each case. Note that the form of all the bytes is 100xx0xx, where the x's are the only bits that are "legal" to change. Also note that port C is set up as two nibble ports instead of one byte port.

Example: we wish to set up all three ports as output ports.

```
In BASIC;          OUT 131,128

In assembly;       SETUP:  LD    80H,A   ;SETUP DATA
                           OUT   83H,A   ;PORT ADDRESS

In Machine code;           3E 80 D3 83
```

By the chart (figure 1 again) we have sent the pattern 1ØØØØØØ binary (128 decimal, 8Ø hex) to the command port, which tells the little pixies inside the 8255 to connect up the internal register ports in the output direction. The next thing one might want to do is send a pattern out there to the output pins. Let's send a Ø1Ø1Ø1Ø1 pattern to port A.

```
In BASIC;          OUT 128,85

In assembly;       WIGGLE:  LD    A,55H  ;DATA PATTERN
                            OUT   A,8ØH  ;PORT ADDRESS

In machine code;           3E 55 D3 8Ø
```

The voltage pattern Ø, 1, Ø, 1, Ø, 1, Ø, 1 (85 decimal, 55 hex) will now appear on pins 37, 38, 39, 4Ø, 1, 2, 3 and 4 in that order. The pattern will stay there until it is changed by another output to port 8Ø hex. Meanwhile, we can output to ports 81 hex and 82 hex without interfering with the pattern we put in 8Ø.

What we have at this point is a set of 24 pins (3 groups of 8) which can be switched high or low by the TRS-8Ø - "under software control", as we say in the argot of the trade. We could control an array of lights, say 4 columns of 6 lights each, using them to form letters which spell out a message, one letter at a time. The message might say something like "YOU ARE NOW WATCHING THE WORLD'S MOST EXPENSIVE AND DIFFICULT TO USE BULLETIN BOARD". With a bit more circuitry (I'll start into that next month), you could also turn your coffeepot on in the morning, but that would make you the servant of your coffeepot, not the other way around. Imagine a quiet Sunday morning, with the sun still just below the sharp horizon of a clear, blue sky. You have just awaken to the soft strains of music coming from your clock radio turned down low, and remember that today you don't have to get up right away. So you roll over and observe the first rays of the morning sun rising through the rosebush outside your window when the coffeepot announces: "ALRIGHT, GUY, COME AND GET IT OR I'LL BOIL IT DOWN TO POWDER"!

What we need is an input to tell the computer to turn the thing off. Obviously, in this example, the appropriate thing is to use a keyboard input but a machine might also need to send data to the computer - not too many machines have fingers, and those that do usually can't type very well. Let's suppose the circuitry has been designed to generate a voltage level (1 or Ø) that we want the computer to see. We would first set up the 8255 chip so that at least one port is an input port, and INPut from that port.

Example:

```
In BASIC;          OUT 131,13Ø : REM B INPUT ,A AND C OUTPUT
                   Q=INP(129)  : REM VOLTAGE PATTERN ON PINS 18-25 PUT INTO "Q"

In assembly;       SETUP:   LD    A,82H   ;B IN, A&C OUT (DATA)
                            OUT   83H,A   ;TO CTRL PORT (ADDRESS)

                   GET:     IN    A,82H   ;B PORT ADDRESS

In machine code;           3E 82 D3 83 DB 82
```

In actual practice, one would normally "set up" the 8255 once in a given program, and then work with the other ports only. In our next issue, we'll have some examples of circuitry that can take the output of the 8255's pins and turn a switch on and off, as well as circuitry that will determine whether a different switch is on or off.

Ralph McElroy
Publisher

Mode 0 (Basic Input/Output)

In this mode, simple input and output operations for each of the three ports are provided. No "handshaking" is required; data is simply written to or read from a specified port.

Mode 0 Port Definition Chart

No.	Control Word Bits								Group A		Group B	
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Port A	Port C (Upper)	Port B	Port C (Lower)
0	1	0	0	0	0	0	0	0	OUTPUT	OUTPUT	OUTPUT	OUTPUT
1	1	0	0	0	0	0	0	1	OUTPUT	OUTPUT	OUTPUT	INPUT
2	1	0	0	0	0	0	1	0	OUTPUT	OUTPUT	INPUT	OUTPUT
3	1	0	0	0	0	0	1	1	OUTPUT	OUTPUT	INPUT	INPUT
4	1	0	0	0	1	0	0	0	OUTPUT	INPUT	OUTPUT	OUTPUT
5	1	0	0	0	1	0	0	1	OUTPUT	INPUT	OUTPUT	INPUT
6	1	0	0	0	1	0	1	0	OUTPUT	INPUT	INPUT	OUTPUT
7	1	0	0	0	1	0	1	1	OUTPUT	INPUT	INPUT	INPUT
8	1	0	0	1	0	0	0	0	INPUT	OUTPUT	OUTPUT	OUTPUT
9	1	0	0	1	0	0	0	1	INPUT	OUTPUT	OUTPUT	INPUT
10	1	0	0	1	0	0	1	0	INPUT	OUTPUT	INPUT	OUTPUT
11	1	0	0	1	0	0	1	1	INPUT	OUTPUT	INPUT	INPUT
12	1	0	0	1	1	0	0	0	INPUT	INPUT	OUTPUT	OUTPUT
13	1	0	0	1	1	0	0	1	INPUT	INPUT	OUTPUT	INPUT
14	1	0	0	1	1	0	1	0	INPUT	INPUT	INPUT	OUTPUT
15	1	0	0	1	1	0	1	1	INPUT	INPUT	INPUT	INPUT

Figure 1

INS8255 Block Diagram

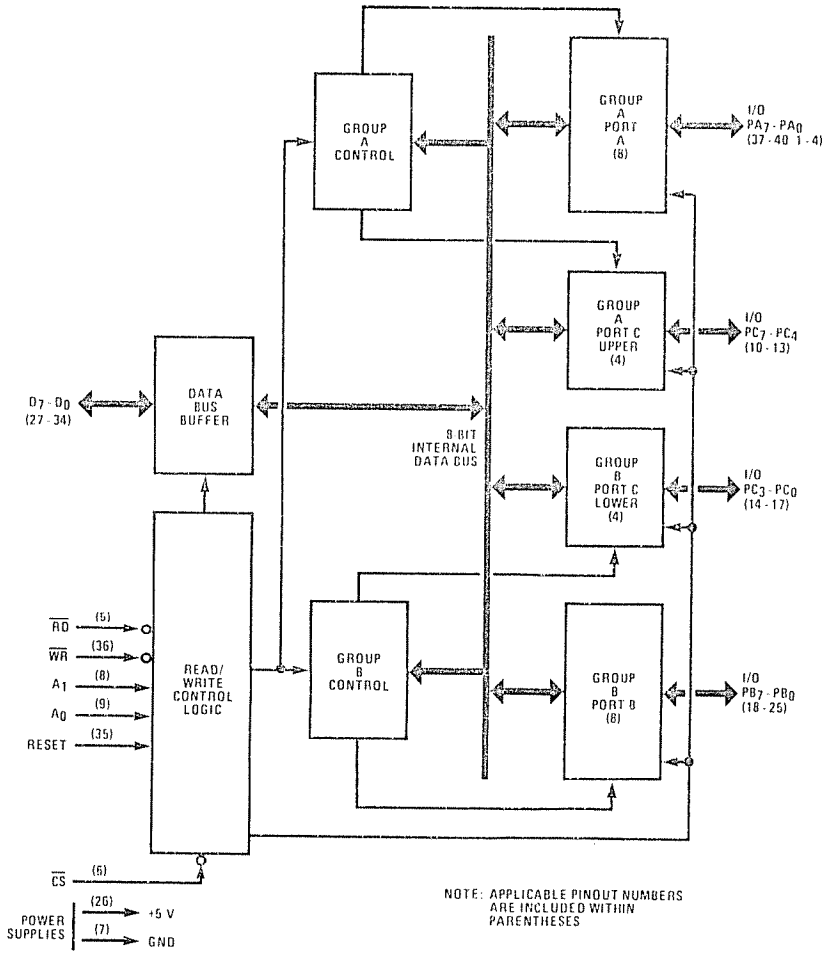

Figure 2

****
** CLOAD Magazine's Handy Dandy Time/Turns chart **
** for Radio Shack's CTR-41 and CTR-80 recorders **
****

Min:Sec	CTR-41 turns	CTR-80 turns
0:10	7	4
0:20	14	8
0:30	21	12
0:40	27	16
0:50	34	20
1:00	40	23
1:10	47	28
1:20	53	31
1:30	59	35
1:40	65	38
1:50	71	42
2:00	78	46
2:10	84	49
2:20	90	53
2:30	95	56
2:40	101	59
2:50	107	63
3:00	113	66
3:10	119	70
3:20	125	74
3:30	130	76
3:40	135	79
3:50	141	83
4:00	147	86
4:10	152	89
4:20	158	93
4:30	163	96
4:40	169	99
4:50	174	102
5:00	179	105
5:10	185	109
5:20	190	112
5:30	195	115
5:40	200	117
5:50	205	121
6:00	210	124
6:10	215	126
6:20	220	129
6:30	225	132
6:40	230	135
6:50	235	138
7:00	240	141
7:10	245	144
7:20	250	147
7:30	254	149
7:40	259	152
7:50	264	155
8:00	269	158
8:10	274	161
8:20	278	164
8:30	283	167
8:40	288	169
8:50	292	172
9:00	297	175
9:10	302	178
9:20	306	180
9:30	311	183
9:40	315	185
9:50	320	188
10:00	324	191
10:10	329	194
10:20	333	196
10:30	337	198
10:40	342	201
10:50	346	204
11:00	350	206
11:10	355	209
11:20	359	211
11:30	363	214
11:40	367	216
11:50	372	219
12:00	376	221
12:10	380	224
12:20	385	227
12:30	389	229
12:40	393	231
12:50	397	234
13:00	401	236
13:10	405	238
13:20	409	241
13:30	413	243
13:40	417	245
13:50	421	248
14:00	426	251
14:10	430	253
14:20	434	255
14:30	437	257
14:40	441	260
14:50	445	262
15:00	449	264