Getting the Most out of 18ØØ

by Don Gladden

"Why do you like the Astro Arcade so much? It only has tiny basic, and only 18ØØ bytes of R.A.M." Did you ever hear something like that in a computer store, or when talking with someone who is not very familiar with the arcade? Well, most of us have, but we realize the potential this little machine has.

Just look at some of the programs that have appeared in the last couple of years. For example, George Moses' 3-part harmony music with titles on the screen, and songs lasting up to ten minutes long or longer. Mike Peaces' "Pack Rat" and "Look out for the bull". Also, "Artillery Duel" which was published here in the Arcadian.

Now the question is; "How do they do it?" While its true that the arcade only has 18ØØ bytes of R.A.M. available without an add-on, the 'tiny basic' is a very powerful tool, when you know its capabilities. I am giving you just a few of the ways I have found to cut down memory, and make the program just a little better. Remember, the more memory you save, the more you can add to your program.

1 - Put all of your subroutines in the beginning of your program, using the 1-digit numbers. (i.e. 2-9). This will only cost you two bytes each time you call the subroutine. For example:

```
          GOSUB 2                         GOSUB 81Ø
   1 byte + 1 byte = 2 bytes       1 byte + 3 bytes = 4 bytes
```

Save line #1 for initialization, and telling the program where to go (???) (sometimes I'd like to!) to begin, jumping over the subroutines. Example:

```
1  A=Ø;B=Ø;C=6Ø;NT=Ø;GOTO 1Ø
2  (subroutine); return
3  (subroutine); return
4  ...5...etc.
1Ø (program beginning)
```

2 - If you have more than eight subroutines, (2-9), sometimes you can save a few bytes by using a variable for the subroutine No. Notice line 1 of the example above. (C=6Ø). You may now give your subroutine line #60, and call it with "GOSUB C". (2 bytes) If you call it more than five times in your program, you can pay for the bytes used for initialization in line #1 with what you've saved using the variable "C".

With this method, you also get a bonus by speeding up the program! This is because the bally executes subroutines, by going to the beginning of the program, and searching for them. It <u>does not</u> go directly to the subroutine.

3 - A good general rule in programming, to cut it down, is to CRAM IT!!! The more statements you can get on one line, the fewer line returns ("GO"=2 bytes) you have to use in your program.

However, do not add anything after an "if" statement, that you want to be executed when the "if" statement is false. The bally will not even see it!! And, of course, you don't add anything after "return" on a subroutine.

4 - The last hint I have, is to study your program carefully, and look for things that ~~it~~ it seems to do more than once, even if it is changing what it does slightly.

Many times I have found formulas or subroutines to do something in one line or so, that took up to 7-8 lines originally.

For example, lets say you want something printed on the screen at CY=Ø but at different CX values to keep your messages at the middle of the screen (a nice professional touch). Instead of using "CX=X;CY=Ø" (CY=Ø takes four bytes) many times, use a subroutine. "(line #) 2 CY=Ø;return". Then you can call it with "CX=X;GOSUB 2" (GOSUB 2 only takes 2 bytes) and save 2 bytes each time you do. If you use it 5 times or more, you are saving memory.

By using these and other methods, we can write programs for our 18ØØ bytes, that I have not seen equalled, even on some 16K systems. Hope they can help you!!