YACalP

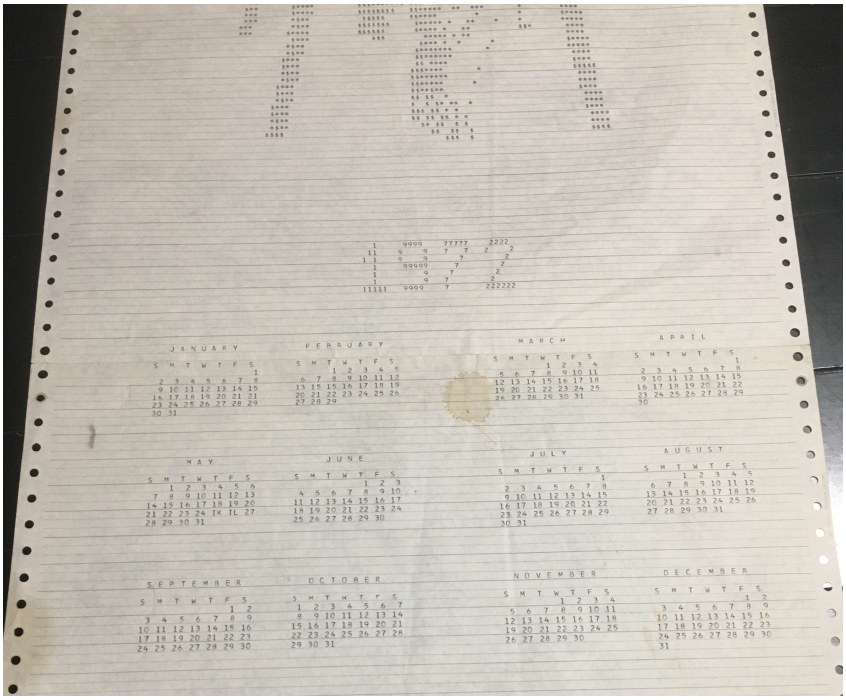
Yet Another Calendar Program
for
NEC PC-8201A/TRS-80 Model 100
by
Lloyd Johnson

1. Introduction

After dusting off the NEC PC-8201A that I had acquired in the early 80s, I have been rehosting/rewriting old BASIC games and other programs to run on it and the TRS-80 Model 100.

Back in the 80s, I, like many other computer people, had a dot matrix printer. My first dot matrix printer was an Okidata Microline 80. It was primitive and did not have descending lower-case letters. I later acquired a NEC 8023 dot matrix printer. This was a better printer and if I ever found one in good shape at a reasonable price, I might consider buying it. Or maybe not since I recently acquired an Okidata Microline 320 dot matrix printer from eBay.

The idea was to print banners, labels and calendars with this printer. Back in the early 70s when I was in college, computer students loved to print calendars with ASCII art. Although the ASCII art images were low resolution, some of these would very likely get you in trouble if you displayed them in the modern workplace. Here is a picture of a calendar that was printed approximately 49 years ago. (I recently found this while going through boxes from storage. I forgot I kept this.)



Although my Okidata Microline 320 printer does not print the wide line printer paper (136 columns), I decided to still put this printer to work and have it print an ASCII art image and a calendar on fan fold paper. I have it hanging by my workbench. I included a picture of this at the end of this documentation.

I found some web pages where you can upload a photograph and it will convert it to an ASCII art image. I've used such a program and updated my profile picture at www.github.com/LEJ-Projects with an ASCII art selfie. This would be the picture I use with my calendar.

What I didn't have was a calendar program. I did a brief search on the internet for a calendar program and the one I found had errors when executed on my NEC PC-8201A. I was about to debug it when I noticed that all the output statements were LPRINTs instead PRINT#1. I really wanted the flexibility of sending the output to either a text document or the RS-232 port in addition to the printer.

Rather than modify the program that was downloaded, I instead decided it would be fun to write my own. I call this program, YACalP.BA for Yet Another Calendar Program. I figured there are already lots of calendar programs out there. This is yet another one.

2. Program Description

After some brief calculations, I decided to make my calendar 3 months across and 4 months down. I was allowing 3 spaces per day. With 7 days per week and 3 months across, I needed 63 spaces for just the days alone. Since I was targeting for a paper width of 80, this gave me 17 spaces I could use for left and right margins and for spacing between the columns of months.

Building a calendar required knowing what day of the week that January 1 fell on. There are many different calendar algorithms found on the internet for determining the day of the week. They were far more complicated than what I needed since they were designed for determining the day of the week for any calendar date regardless of the year.

I created my own algorithm for determining the day of the week for January 1. I did it by using online algorithms for determining the day of the week for January 1, 1900 and then advancing the day of the week by one for each year beyond 1900. I also advanced the day of the week for each leap year that occurred between 1900 and the desired year. If the year is in the 2000s, then I do the same thing but instead use January 1, 2000 as my initial day of the week. This is when I learned that although 2000 is a leap year, 1900 was not. This has something to do with the Gregorian calendar adjustment. I adjusted my algorithm accordingly.

So here you have the major limitation of my calendar program in that it will only work for years between 1900 and 2099 inclusive. It probably wouldn't take much additional effort to make it work for any year, but then things get more complicated when you take into consideration the year that Gregorian calendar was adopted. This varies from country to country. The British Empire (including the colonies in the US) did not adopt the Gregorian calendar until 1752.

I reasoned that if this program is used to print the year someone (that is still alive) was born as well as printing the current year or any future years then only generating accurate calendars for 1900-2099 should be sufficient. Most people that know history will know that December 7, 1941 was on Sunday. You can confirm this yourself by printing a calendar for 1941.

After the year is entered, the output name is requested. If nothing is entered, the default is "COM:5I71XS" for the NEC and "COM:57I1E" for the TRS-80. This will direct the output to the RS-232 port at 1200 baud. If a different baud rate is desired, the user can either update the program or enter “COM:” followed by the desired parameters. On a TRS-80 model 100, the output will go to the printer if “LPT:” is entered. Unfortunately, this does not work with the NEC. For the NEC, I simply type a file name. For a 2021 calendar, I type CAL21 and CAL21.DO is created. To print CAL21.DO, I leave BASIC, select the file on the main menu and then use the f4 key to list the file. To print a .DO file on the TRS-80, the file is opened with the TEXT editor and will be printed if the print key is pressed while the shift key is held down.

Once the desired output is entered/selected, the calendar is built. There will be a slight delay during this process. I am storing the days for each month is a 12x42 array. The value of 42 represents six weeks (6x7=42). In any given year, some of the months will have days in six different weeks. For 2021, this occurs in January and October. Since I’m using the 0 index, the array M is defined as DIM M(11,41). For the second index, indices of 0-6 correspond to Sunday through Saturday for week 1, indices of 7-13 correspond to Sunday through Saturday for week 2, etc. Once the array for the first index = 0 is completed (January), the FOR loop increments the first index and the array is filled for the next month starting at the next day of the week. A month is completed when the day inserted into the array reaches the maximum number of days for that month. The day of the week is incremented and if it exceeds 6, it is reset back to 0.

After the calendar is built, it is printed. First the year is printed in large numeric characters and then the calendar itself is printed with the months and days of the week labeled.

The large size numbers used for the year was accomplished by using a 5x7 array. I decided to use an asterisk for the dot making up the 5x7 numeric character. My example from 1972 used the numbers for the digit as the dot in the 5x7 array. That’s cool, but I didn’t think it was worth the programming effort. I instead went with an asterisk for the dot. I was already using horizontal rows of asterisks and vertical columns of asterisks to separate the months on the calendar.

The rows of the 5x7 matrix are packed into a single number by assigning the bit position (or power of 2) for a dot as a 1 or a 0. I used a simple Excel spreadsheet to calculate the value of the row based on the asterisks inserted in the number cell. The number 2 is shown as an example.

	*	*	*		0	8	4	2	0	14
*				*	16	0	0	0	1	17
			*		0	0	0	2	0	2
		*			0	0	4	0	0	4
	*				0	8	0	0	0	8
*					16	0	0	0	0	16
*	*	*	*	*	16	8	4	2	1	31

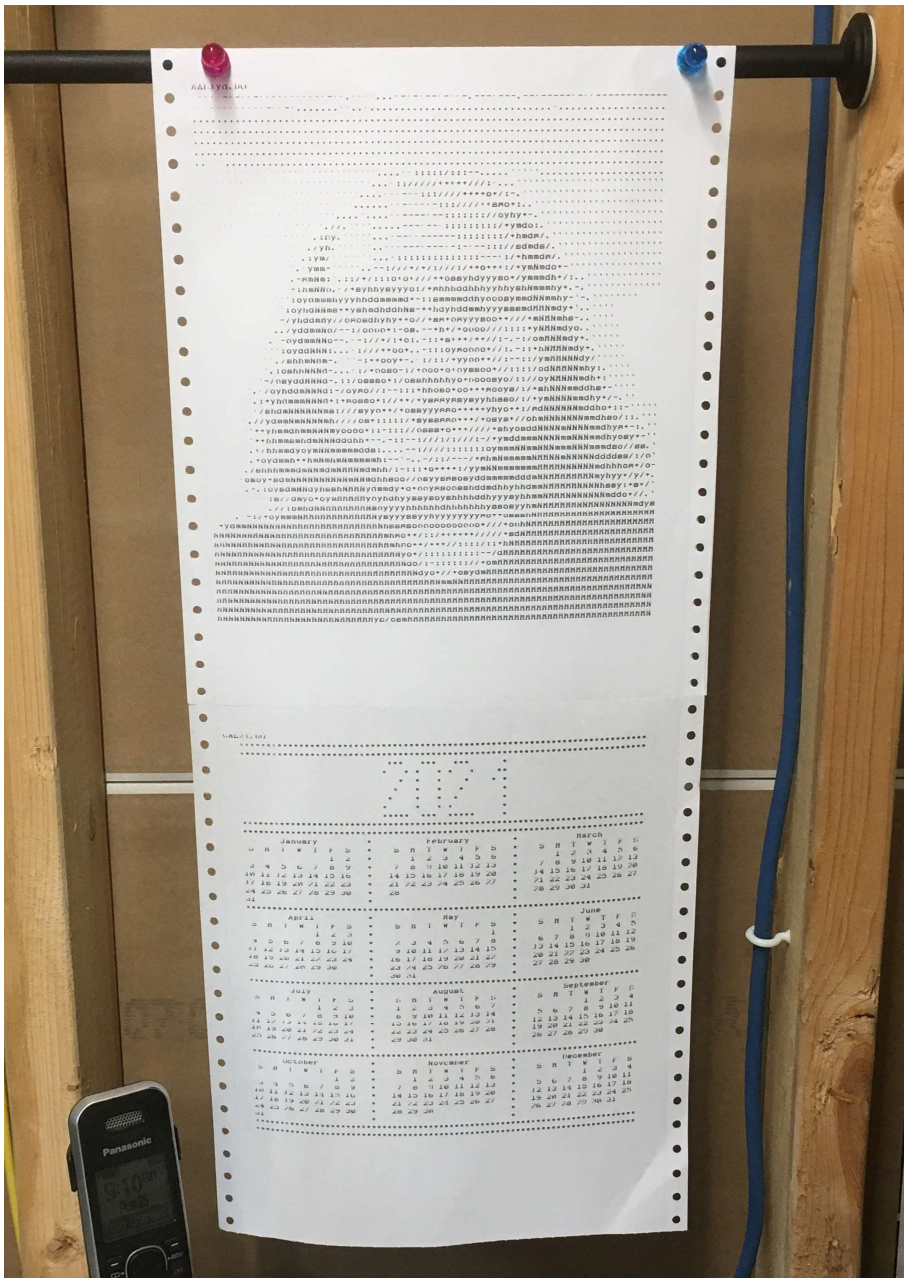
In the program there are 10 data statements each containing 7 numbers. The data statement for the number 2 is:

```
1100 DATA 14,17,2,4,8,16,31:'2
```

Once the year is printed, the calendar itself is printed after first printing a couple rows of asterisks as a divider. The routine for printing the calendar is designed for printing three months at a time. If I should ever acquire a dot matrix printer that does 136 columns, I might want to revise this routine to print four months in a row. That should be the only code that needs to be modified except for perhaps modifying the number of spaces printed to center the year. Based on what I have seen on eBay, it is doubtful I will ever be buying a dot matrix printer that can print 136 column wide paper.

Printing the calendar required many nested loops. The first loop is to print the 4 rows of months. The next loop is a short loop to print the first letter of each day of the week. This loop does this for each column of months and will be executed each time a new row of months is started. A new loop is then started which runs from 0 through 5. Each index corresponds to the week number within the month since a calendar month can span six weeks. Another nested loop is started that runs from 0 to 2 for each month in the row of months. Finally, the most inner loop is executed. This loop runs from 0 to 6 for each day of the week. If the value of the month array is 0, then spaces are printed. If it is nonzero than the value represents the day of the month and this value is printed allowing for 3 digits. Since the day of the month never exceeds two digits, this provides for a leading space or two leading spaces if the day of the month is single digit.

The following is a picture of a 2021 calendar with an ASCII image of me that I have displayed near my work bench.



I hope you will find this program with this documentation interesting or useful.