

P*ROM Application Note

Modifying PCS EPROM Programs to run on CF Series Flash Cards

This application note provides details on how to modify a program written for the PCS EPROM cards (8K, 32K, or 64K) to run on the P*ROM CF Series (8K, 32K, 64K, or 128K) Flash card. The modified program will continue to run on both the PCS card as well as the PCS Emulator. The modification is simple and only affects the bank switching techniques used. No modification or alteration of the program code itself is required.

Converting an 8K single-bank program

If your program fits on a PCS 8K EPROM card or within a single bank of a PCS 32K EPROM card, you need only declare one string array. The array is not used, and there is no bank switching (because this is a single-bank program).

Add a new array at the beginning of the list of dimensioned arrays. This new array must be first on the list and immediately after the CLEAR statement.

For example, if your program contains the following statements:

```
100 CLEAR : DIM A(27), B$(9), K(20) ...
```

change it to read

```
100 CLEAR : DIM Z$(3)*3, A(27), B$(9), K(20) ...
```

We've used Z\$() for this array, however, if you have already used the Z\$() array, you can use any other unused letter from B\$ to Y\$ for this array (do not use the letter A).

Do not use any of the elements in the Z\$() array, except as specifically shown below for multi-bank programs. The Z\$() array consumes an additional 18 bytes of RAM space. (Generally this presents no problem because there is 8K of RAM space for variables and most programs use only a small portion of it.)

The modified program can be installed on both the PCS 8K EPROM card and on any P*ROM CF Series Flash card. It will also run in the PCS Emulator.

Converting multi-bank programs

If your program has more than one bank of code, you need to dimension the new array as shown above and also modify the bank-switching method, as shown below. The initialization routine where arrays are dimensioned must appear in bank 0.

The A() array can continue to be used for variables, however, A(27) is no longer used to switch banks. The common headers are to be modified as shown below. The references to A(27) can be left if you wish, but will no longer have any effect on the switching of banks. (We have removed the A(27) references in the following examples.)

Suggested Common Header for Bank 0

The following header is suggested for bank 0 (the first bank of a multi-bank program) but does not have to be used exactly as shown. You may wish to modify it or use a somewhat different version. The only requirement is that the header in all banks be identical except for values assigned to Z\$(1) to switch banks, e.g., "000", "111", "222", "333", etc.).

```

1  IF Z = 0 \
2      GOSUB 20
2  RETURN
3  "A" :
4      GOSUB 1 :
5      Z$(1) = "000" :
6      GOTO "AA"
7
8  "B" :
9      GOSUB 1 :
10     Z$(1) = "000" :
11     GOTO "BB"
12
13 "C" :
14     GOSUB 1 :
15     Z$(1) = "111" :
16     GOTO "CC"
17
18 "D" :
19     GOSUB 1 :
20     Z$(1) = "111" :
21     GOTO "DD"
22
23 "J" :
24     GOSUB 1 :
25     Z$(1) = "222" :
26     GOTO "JJ"
27
28 "K" :
29     GOSUB 1 :
30     Z$(1) = "222" :
31     GOTO "KK"
32
33 "L" :
34     GOSUB 1 :
35     Z$(1) = "333" :
36     GOTO "LL"
37
38 "M" :
39     GOSUB 1 :
40     Z$(1) = "333" :
41     GOTO "MM"
42
43 ----- End of common header -----
44 CLEAR :
45     DIM Z$(3)*3, \
46     ... :
47     Z = 1 :
48     RETURN
49
50 "AA" : ...
51 "BB" : ...

```

' "A" key goes to "AA" in bank 0
' Note: This is 3 zeros within quotes
' "B" key goes to "BB" in bank 0
' "C" key goes to "CC" in bank 1
' Note: This is 3 1's in quotes
' "D" key goes to "DD" in bank 1
' "J" key goes to "JJ" in bank 2
' Note: This is 3 2's in quotes
' "K" key goes to "KK" in bank 2
' "L" key goes to "LL" in bank 3
' "M" key goes to "MM" in bank 3
' clears any dimensioned arrays
' dimension this array first
' followed by other arrays in any order
' set initialized flag to non-zero value
' return
' Code for "A" key
' Code for "B" key

Header for Other Banks

The header for all other banks should be the same as for bank 0 up to the comment line "----End of Common Header----". If you use the PCS Tokenizer, an identical copy of the code in bank 0 at line 20 must also appear in all other banks. If you use the SBC BASIC Compiler, the code at line 20 should only appear in bank 0 and does not have to appear in any other banks.

By changing the Z\$(1) = "xxx" line in the header, you can send any function key to any bank (xxx represents the bank number and is the characters "000", "111", "222", "333", "444", "555", "666", or "777" representing bank 0, 1, 2, 3, 4, 5, 6, & 7). The initialization code is always in bank 0 (see line 20 in bank 0 above) and, if you are using the PCS Tokenizer, must also appear identically in all other banks.

The modified program will run in both the PCS EPROM cards and the P*ROM CF Series Flash cards. It will also run in the PCS Emulator. Once the modifications are made, you can use the PCS Tokenizer to retokenize the program. The resulting files are then linked into a single file with the BANKS.EXE linker to program CF Series Flash Cards. (See Using Multiple Banks in a Flash Card in the P*ROM Application Note Programming CF Series Flash Cards.) The linked file can be used with P*ROM Flash Card Programmers as well as the PCS Programmer.

Calling Subroutines in Another Bank

Calling a subroutine is very similar to the bank switching shown above. The bank switching occurs with the assignment to Z\$(1) of the 3-character string containing the bank number.

The above sample headers assume all the executable code for a certain function key resides within a single bank. This is the simplest way to write a multi-bank program and is also the easiest to debug. However, if you want to call a subroutine in another bank, you can add the following line to the header in bank 0:

```
11 W$ = "000" : Z$(1) = Y$ : GOSUB X$ : Z$(1) = W$ : RETURN
```

The same line with the following change must be added to each of the other banks. The first segment W\$ = "000" should be replaced with W\$ = "111" in bank 1, W\$ = "222" in bank 2, W\$ = "333" in bank 3, and so forth.

To use a subroutine identified with a label, we store the label we want to execute in the variable X\$ and the bank in which the subroutine resides in Y\$. For example, to call the subroutine "TT" in bank 3, we would issue the following code:

```
200 X$ = "TT" :           ' Execute subroutine with label "TT"
     Y$ = "333" :       ' that is in bank 3
     GOSUB 11 :
     ...                ' Return here after subroutine is executed
```

Jumping to Labels/Lines in Another Bank

Using a technique even simpler than the one above, you can also jump to a label or line number in another bank. We add line 12 to the common header as follows:

```
12 Z$(1) = Y$ : GOTO X$
```

We store the bank we want to jump to in Y\$, and the label we want to jump to in X\$:

```
220 X$ = "YT" :           ' Label we're jumping to
     Y$ = "222" :       ' Bank the label is in
     GOTO 12             ' Execute the jump
```

If we want to jump to a line number instead of a label, modify the code as follows:

```
12 Z$(1) = Y$ : GOTO X
220 X = 448 :           ' Line number we're jumping to
     Y$ = "222" :       ' Line is in bank 2
     GOTO 12             ' Execute the jump
```

Note on Variable Usage

We have used the variables W\$, X\$, Y\$, and Z\$ for these GOSUB and GOTO operations from one bank to another. You can use any other single-letter variable or even array variables if you wish. If you use array variables, make sure to declare the array(s) in bank 0 and substitute the array variable in place of the W\$, X\$ (or X), Y\$, and Z\$ in the above lines of code.

Converting "Flex" Card Programs

Programs written for the PCS Flex Card that are longer than 8K cannot be made compatible so they will run on both a Flex Card and a Flash card. If the program is longer than 8K, you can convert it to run on a 16K Flash Card. See the P*ROM Application Note Programming CF Series Flash Cards.

Flex Card programs will run on a 16K RAM card without modification. (The Flex Card is based on a now-discontinued IC and will probably not continue to be available.)

Using the PCS Emulator

Your program will continue to work in the PCS Emulator even after you have modified the common headers as discussed above (most emulators can only handle up to 64K of program). This means you can continue to use your PCS Emulator to test programs regardless of whether they're installed on a PCS EPROM card or a P*ROM CF Series Flash card.

For more information, contact

**P*ROM Software, Inc. 1820 Shelburne Road, Burlington, Vermont 05406-4027
800 843-7766 802 862-7500 Fax 802 862-8357**