MC68705P5 Bootstrap ROM

The sole purpose of the Bootstrap ROM residing in Motorola's MC68705P5 single chip micro is to program its own EPROM by copying the data from an external EPROM (2716) which has been programmed with an exact duplicate of the required information (refer to the MC68705P5 data sheets for more info).

With the help of Matthieu Benoit I was able to confirm that the Bootstrap ROM in the MC68705P5 is identical to the ROM in the MC68705P3 except for the last 2 bytes. I'll only describe the difference here but for my complete disassembled listing of the MC68705P3 Bootstrap ROM please refer to the relevant pdf on this website.

Detail:

In this discussion I'll refer to the MC68705P3 as just P3 and the MC68705P5 as P5.

The Bootstrap ROM in both the P3 and P5 is 115 bytes long and is located between memory locations 0785H and 07F7H inclusive. As described in my P3 ROM listing the last 2 bytes (found in 07F6H and 07F7H) are used as a vector – when the P3 or P5 goes into Programming mode it forms an address using those 2 bytes and begins executing code from that memory location. In the P3 the 2 bytes are 07H and 85H which, as expected, is the starting address of the Bootstrap ROM (0785H).

However in the P5 this vector is not 0785H but 0782H instead. Note that 0782H, 0783H and 0784H are the addresses of the last 3 bytes in the Main User EPROM and can be programmed by the user (0784H is the MOR). Note too that a blank P5 has all EPROM locations set to 0.

So when a blank P5 is put into Programming mode (refer to P5 datasheet) execution starts at memory location 0782H. The 00H found there is a brset instruction (bit test and branch). Since this is a 3 byte instruction the actual instruction executed is

```
brset   0,0,0
```

which basically does nothing. Note that the 3rd byte is the MOR but as far as the P5 is concerned it's just a memory location which it reads as 00H.
The next instruction executed is the first one in the Bootstrap ROM so execution continues exactly the same as if the micro was a P3.

The reason the P5 does this extra step is so the user can place their own instruction in the last 2 bytes of the Main User EPROM thereby stopping the Bootstrap ROM from executing. To put the P5 into this "secure" mode the datasheet says:

- set bit3 (SNM) in the MOR register and
- program 20H & FEH into memory locations 0782H and 0783H respectively.

It seems that setting bit3 of the MOR is how you can stop the P5 from being able to enter NUM mode which would allow the EPROM to be read/programmed externally. Unfortunately there's essentially no information available on NUM mode however the P5 datasheet does say:

“when this bit is set ie programmed to a "1" it is not possible to access the EPROM contents of the MC68705P5 externally".

The second step is required too. Putting 20H & FEH in locations 0782H & 0783H creates an infinite loop (branch to itself) which prevents the Bootstrap ROM from running. This means that you can’t re-program the P5 or use some of the techniques discussed elsewhere to determine the contents of the User EPROM unless you erase it first.

How to determine if a P5 is secured:

It seems to me that you can’t easily test to see if bit3 of the MOR has been set. Most Programmers don’t use this mode anyway and with Matthieu Benoit’s help I was able to confirm that setting this bit has no affect on running the Bootstrap ROM. Since there’s no information available on how to program/read the contents of the P5 EPROM externally I think we can ignore this aspect of “secure” mode.

However if 20H, FEH (or something else) has been programmed into memory locations 07F6H, 07F7H to prevent the Bootstrap ROM from running we are able to test for that.

The example programming circuit in the P5 datasheet uses an external EPROM holding a duplicate copy of the information which needs to be programmed. When the Bootstrap ROM executes it burns a copy of the external EPROM into its own internal EPROM. Other Programmers do a similar thing but either load the data to be copied into external RAM or they simply feed the data one byte at a time to the P5 as the Bootstrap ROM requests it.

So to check if the Bootstrap ROM has been by-passed all that’s required is to monitor pin 15 on the P5 with an oscilloscope while performing a “dummy” program cycle. If you refer to the example programming circuit in the P5 datasheet pin 15 is labelled as /count and connects to pin 10 on the 4040 counter. This signal increments the counter so successive bytes can be read from the external EPROM.
If the Bootstrap ROM is running then negative going pulses will appear on that pin which means the P5 isn’t secured. This means the User EPROM contents can be read using various devices and techniques described elsewhere on this website.
Note that if you don’t have an oscilloscope you can just do the “dummy” program cycle and see if the Programmer freezes. If it does it means the P5 is secure.

How can we do a “dummy” program cycle?
A “dummy” program is possible due to two observations I made when looking at the Bootstrap ROM listing:

- the Bootstrap doesn’t check if the programming voltage (+21V) is applied
- the Bootstrap doesn’t program an EPROM location if the data is 00H.

So the idea is to disconnect the P5’s pin 6 (Vpp) from the Programmer and connect it to +5V instead (eg connect it to pin 3, Vcc). You can probably make up an adapter using a 28 pin IC socket and some header pins. Piggyback the socket on the header pins and connect every pin of the socket to a pin except for pin 6. Connect pin 6 of the IC socket to pin 3. Pin 6 of your Programmer’s socket will not be connected to anything.

Then depending on your Programmer program a blank 2716 EPROM with all zeros. NOTE this is very important because unlike the P5 normal EPROMs such as the 2716 have all locations set to FFH when erased. We need the 2716 to contain all 00Hs.
Alternatively your Programmer might require you to load a file. In that case make a blank file ie a file with 2048 zeros.

Now when you program the P5 its internal EPROM won't be modified because the programming voltage is too low and the data values to be programmed are all zeros. However with the oscilloscope look at pin 15 on the P5. If you see pulses then the P5 isn't secure & the Bootstrap is running. If you see nothing or the Programmer freezes it means the P5 is secure.

NOTE Use any information here at your own risk. It may or may not work as described and you could overwrite the contents of your P5.

Peter Ihnat, Jun 2013. pihnat@uow.edu.au