GPM for the M6800

Frits van der Wateren
Physical Laboratory
Wilhemina Hospital
Amsterdam 1056 - E.G
The Netherlands

Received: 77 Nov 28

This GPM version is written after an article in *The Computer Journal*, Vol. 8, page 225, by C. Strachey. I will give a short explanation about GPM here. For further details see the article mentioned above.

GPM is a general purpose macro generator, i.e. every character from the input device is directly echoed on the output device until the start of a macro call. This call is then replaced by its body, with all evaluation of parameters etc. And the final result appears then on the output device.

A macro call is initiated by the $ sign and looks like:

```
$MACRONAME, PARAMETER 1, PARAMETER 2;
```

We see three special characters: $, ; . The action of $ is already mentioned above. The , is used to separate the different items. And the ; closes the macro call. These characters are called "warning characters" in GPM. MACRONAME is now evaluated with the parameters: PARAMETER 1 and PARAMETER 2. Beyond the standard macros which are implemented in the system, you must define your own macros to be used. This is done by the standard macro DEF. We want, for instance, to define a macro SEQUENCE which gives as a result 0123456789. In GPM this looks like:

```
$DEF,SEQUENCE,0123456789;
```

If we now call the macro SEQUENCE by:

```
$SEQUENCE;
```

we get:

```
0123456789
```

as a result on the output device.

There are some other warning characters, namely: < > and #. The characters < and > are used as string quotes and have the effect of preventing evaluation of any macro calls inside.

In place of an evaluation, however, one "layer" of string quotes is removed. The warning character #, which is always followed by one single character, has, during the evaluation, as action that it is replaced by the parameter supplied in the call. The character which follows # may be any number from 0 to 9 and any letter from A to Z. A stands for 10, B for 11, etc. Now #1 stands for parameter 1 and so on, while #0 stands for the macro name itself.

Now a real example of a macro, which adds 2 to the contents of a memory location. We define ADD2 as:

```
$DEF,ADD2,<     LDAB        #1
          ADDB  <#>2
          STAB  #1
>;
```

We see that # in line 2 must be quoted to prevent evaluation, otherwise it would be replaced by parameter 2, which gives nothing as a result in this case.

A call to ADD2 with DATA as a parameter, thus:

```
$ADD2,DATA;
```

Will give as a result:

```
LDAB    DATA
ADDB    #2
STAB    DATA
```

Macro calls may be nested to any level. See, however, for a more detailed description of these techniques the article by Strachey.

Standard (or machine) macros in the M6800 implementation:

$DEF, macroname, macrobody;
Define the macro "macroname" with body: "macrobody". This macro body, in turn, may contain both macro calls and definitions.

$VAL, macroname;
This call has as a result the body of macro "macroname" without any evaluation. Applying this macro to a machine macro or a non existing macro, has nothing as a result.

$UPDATE, macroname, newbody;
This macro has the same sort of effect as DEF except that "macroname" is not newly defined but already exists, and is supplied with a new body, which is given as second argument in UPDATE. The new body may be of equal length or shorter. A longer body will be truncated, but will give no error message.

$CLEAR;
Reinitializes GPM, i.e all user-defined macros are deleted and all current evaluations are aborted. The warning characters are set to their default values.

$DUPLEX,n;
The only effect of DUPLEX is setting the input mode of the TTY to half or full duplex. When n=1 the input characters are echoed; when n=0 the input characters are not echoed. This feature is very useful when using GPM interactively.

$USER,xxxx,string;
Goto user routine at address xxxx (hex) with "string" as parameter. "string", however, may be omitted, thus: $USER,xxxx; when the address is shorter than 4 digits, the MSB digits are supplied with zeros. Non-hex characters are ignored. One must be familiar with the internal structure of GPM, when using this macro call.

$CW,<abcdef>;
Change warning characters in the following sequence: $,;#><. So "$" becomes "a" and "," becomes "b", etc. This is of course an absurd example. $CW,<[.] \ / >; would be a more sensible definition. Now all warning characters are lower case characters on most terminals. When using this feature it is best to use this macro immediately after starting GPM.

As we see this GPM version has no error messages at all so everything is allowed. This is the greatest difference between this implementation and that of Strachey; and in my opinion an improvement. When you use in a definition, for instance, #4 as a parameter call. And it seems that in the actual macro call parameter 4 is not supplied, the #4 is replaced by the null string i.e. nothing, rather than an error message. This may be very handy. The only problem we can meet sometimes is a lack of memory space. GPM then types: ** MEMORY FULL ** and the current evaluation is aborted. All macros defined so far are retained.