Format   CALL EXECUTE(cpu-address[,...])

                CALL EXECUTE(numeric-variable[,...])

Description

The EXECUTE subprogram directly goes to the cpu-address and
expects to find 4 bytes to be present. The bytes are 1 and 2
define the workspace register address. Bytes 3 and 4 define
the address to start execution at in cpu memory. Programmers
can see this is a BLWP at a cpu-address. The programmer is
responsible for keeping track of the workspace and program
space he is using. Also for any registers while doing a BL or
another context switch. A RTWP will end either a BL or a BLWP
as long as registers set are not changed. By using CALL LOAD
or CALL MOVES the programmer can set up a BLWP routine in the
lower 8K by filling the registers with values first, then
using CALL EXECUTE to directly complete these commands. This
is faster then CALL LINK as no interpretation of the access
or values are checked.
 EXECUTE runs a XML link from GPL by moving 12 bytes from the
Fast RAM at HEX 8300 to VDP at HEX 03C0 then moving the value
in FAC passed from XB to HEX 8304 and does a GPL XML >F0
After a RTWP by the Assembly program, it returns VDP HEX 03C0
to Fast RAM HEX 8300 so the 12 bytes are restored. Thus this
allows programmers use of FAC and ARG areas in Fast RAM.
 Here is the program loaded into Fast RAM by EXECUTE:

```
                AORG    >8300
 CPUPGM         DATA    >8302       First address.
                BLWP    @>834A      Switch context
                                    with FAC as dummy.
                CLR     @>837C      Clear for GPL return.
                RT                  Return to GPL.
                END
```

 If a programmer absolutely must use Fast RAM for his program
I suggest he set up a buffer for saving HEX 8300 to HEX 83FF
if only so it will not mess up any GPL pointers and don't go
and mess up the 12 bytes at VDP HEX >03C0. Then the only
thing to worry about is messing up something else.