

HowToCode7

| |
|----------------------|
| COLLABORATORS |
|----------------------|

| | | | |
|---------------|------------------------------|-------------------|------------------|
| | <i>TITLE :</i> HowToCode7 | | |
| <i>ACTION</i> | <i>NAME</i> | <i>DATE</i> | <i>SIGNATURE</i> |
| WRITTEN BY | | November 29, 2024 | |

| |
|-------------------------|
| REVISION HISTORY |
|-------------------------|

| NUMBER | DATE | DESCRIPTION | NAME |
|--------|------|-------------|------|
| | | | |

Contents

| | | |
|----------|-----------------------------------|----------|
| 1 | HowToCode7 | 1 |
| 1.1 | HowToCode: Debugging | 1 |
| 1.2 | Debugging with Enforcer | 1 |
| 1.3 | Breaking out of loops | 2 |
| 1.4 | Monitors | 3 |

Chapter 1

HowToCode7

1.1 HowToCode: Debugging

Debugging your Code

A decent debugger is **essential** for anyone planning to program in 68000. If you haven't got a decent debugger, I suggest you get MonAm (free with Hisoft Devpac and Hisoft Hi-Speed Pascal) which is very very good. If you're using AsmOne there is a debugger implemented. Use ad instead of a when assembling. Step down with arrow right (allows you to jump to subroutines). There are some other tips you can use:

- 1 Debugging with Enforcer
- 2 Breaking out of loops
- 3 AA Monitor problems

1.2 Debugging with Enforcer

Debugging with Enforcer

Commodore have written a number of utilities that are **excellent** for debugging. They are great for trapping errors in code, such as illegal memory access and using memory not previously allocated.

The down side is they need to things:

a) A Memory Management Unit (at least for Enforcer). This rules out any 68000 machine, and (unfortunately) the Amiga 1200 and the Amiga 4000/EC030. If you are seriously into programming insist on a FULL 68030/40 chip, accept no substitute. Amiga 2000 owners on a tight budget may want to look at the Commodore A2620 card (14Mhz 68020 with 68851 MMU fitted) which will work and is now very cheap.

b) A serial terminal. This is really essential anyway, any

serious programmer will have a terminal (I have an old Amiga 500 running NCOMM for this task) to output debug information with `dprintf()` from their code. This is the only sensible way to display debug info while messing with copperlists and hardware displays. If you can't afford a terminal, or you are very short of desk space, then another utility, called Sushi, will redirect this output to a file or window on your Amiga, although you will have to keep the system alive for this to work properly...

Enforcer, Mungwall and other utilities are available on Fred Fish Disks, `amiga.physik` and `wuarchive`, and probably on an issue of the excellent "The Source" coders magazine from Epsilon.

1.3 Breaking out of loops

How to break out of never-ending loops

Another great tip for Boerge here:

This is a simple tip I have. I needed to be able to break out of my code if I had neverending loops. I also needed to call my exit code when I did this. Therefore I could not just exit from the keyboard interrupt which I have taken over (along with the rest of the machine). My solution was to enter supervisor mode before I start my program, and if I set the stack back then I can do an RTE in the interrupt and just return from the Supervisor() call. This is snap'ed from my code:

```

lea      .SupervisorCode,a5
move.l   sp,a4          ;
move.l   (sp),a3        ;
EXEC     Supervisor
bra      ReturnWithOS

```

```

.SupervisorCode
move.l   sp,crashstack   ; remember SSP
move.l   USP,a7          ; swap USP and SSP
move.l   a3,-(sp)        ; push return address on stack

```

that last was needed because it was a subroutine that RTSes (boy did I have problems working out my crashes before I fixed that)

Then I have my exit code:

```

ReturnWithOS
tst.l    crashstack
beq      .nocrash
move.l   crashstack,sp
clr.l    crashstack
RTE                                ; return from supervisor mode
.nocrash

```

my exit code goes on after this.

This made it possible to escape from an interrupt without having to care for what the exception frames look like.

(CJ) I haven't tried this because my code never crashes. ;-)

1.4 Monitors

Monitors

If you are using AA-chipmodes, or want to make your code compatible with it, you must also make sure you code works with every MONITOR on the market. Not only the computer (Thanks to Alien/A poor group in Ankara but not Bronx), for spotting this in my JoyRide2 Intro.

This is **extremely** difficult. See Monitor Problems in the AGA chapter.