

**HowToCode7**

COLLABORATORS

|            |                       |                   |           |
|------------|-----------------------|-------------------|-----------|
|            | TITLE :<br>HowToCode7 |                   |           |
| ACTION     | NAME                  | DATE              | SIGNATURE |
| WRITTEN BY |                       | November 28, 2024 |           |

REVISION HISTORY

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

# Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>HowToCode7</b>                          | <b>1</b> |
| 1.1      | HowToCode: Startup and Exit . . . . .      | 1        |
| 1.2      | Your code won't run from an icon . . . . . | 1        |
| 1.3      | Error Numbers . . . . .                    | 1        |
| 1.4      | Avoiding Forbid() . . . . .                | 2        |
| 1.5      | Sprite Initialisation . . . . .            | 3        |

## Chapter 1

# HowToCode7

### 1.1 HowToCode: Startup and Exit

Startup and Exit Problems with Amiga Coding

=====

- 1 Your code won't run from an icon
- 2 Error codes on exit
- 3 Avoiding Forbid() and Permit()
- 4 Sprite problems

### 1.2 Your code won't run from an icon

Your code won't run from an icon

-----

You stick an icon for your new demo (not everyone uses the CLI!) and it either crashes or doesn't give back all the RAM it uses. Why?

Icon startup needs specific code to reply to the workbench message. With the excellent Hisoft Devpac assembler, all you need to do is add the line

```
include "misc/easystart.i"
```

and it magically works!

For those without Devpac, the relevant code is included in this archive as `iconstartup.i`

### 1.3 Error Numbers

Error numbers when run from CLI script files

-----

You may get an error like this from your code when run from

a batch file:

Program failed return code 184641234.

The return code is the value in D0 at the end of your program, so for a clean exit, always clear d0 immediately before your final RTS.

Of course you can use the return code in your code to allow conditional branching after your code in a script file. For example:

```
* Simple example - assemble to checkbutton
    opt      l-

    btst     #6,$bfe001      ; check left mouse button (hardware)
    bne.s    .notpressed
    moveq    #0,d0
    rts
.notpressed
    moveq    #5,d0
    rts
```

Assemble this, and you have a program that can tell if the mouse button is pressed during bootup. Ideal for switching between startup sequences, for example with this amigados script file.

```
checkbutton
if WARN
execute s:startup-nomousepressed
else
execute s:startup-mousepressed
endif
```

## 1.4 Avoiding Forbid()

Avoiding Forbid() and Permit()

I've tried it, this works, it's wonderful.

Instead of using Forbid() and Permit() to prevent the OS stealing time from your code, you could put your demo or game at a high task priority.

The following code at the beginning will do this:

```
move.l     4.w,a6
sub.l      a1,a1          ; Zero - Find current task
jsr        _LVOFindTask(a6)

move.l     d0,a1
moveq      #127,d0        ; task priority to very high...
jsr        _LVOSetTaskPri(a6)
```

Now, only essential system activity will dare to steal time from your code. This means you can now carry on using `dos.library` to load files from hard drives, CD-ROM, etc, while your code is running.

Try using this instead of `Forbid()` and `Permit()`, and insert a new floppy disk while your code is running. Wow... The system recognises the disk change.... But remember to add your input handler!!!

Of course this is purely up to you. You may prefer to `Forbid()` when your code is running (it makes it easier to write).

## 1.5 Sprite Initialisation

### Sprite Initialisation

-----

Some people doesn't initialize the sprites they don't want to use correctly. (This reminds me of Soundtracker.)  
A common error is unwanted sprites pointing at address \$0.  
If the longword at address \$0 isn't zero you'll get some funny looking sprites at unpredictable places.

The right way of getting rid of sprites is to point them to an address you for sure know is #00000000 (0.l), and with AGA you may need to point to FOUR long words of 0 on a 64-bit boundary

```
        CNOP    0,8
pointhere: dc.l    0,0,0,0
```

The second problem is people turning off the sprite DMA at the wrong time. Vertical stripes on the screen are not always beautiful. Wrong time means that you turn off the DMA when it is "drawing" a sprite.  
It is very easy to avoid this.  
Just turn off the DMA when the raster is in the vertical blank area.

Currently V39 Kickstart has a bug where sprite resolution and width are not always reset when you run your own code.  
See [Fixing Sprites in AGA](#)