

**HowToCode7**

COLLABORATORS
---------------

	TITLE : HowToCode7		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		March 28, 2025	

REVISION HISTORY
------------------

NUMBER	DATE	DESCRIPTION	NAME

# Contents

<b>1</b>	<b>HowToCode7</b>	<b>1</b>
1.1	HowToCode: Programming the Copper	1
1.2	Proper Copper Startup	1
1.3	graphics.library/LoadView()	2
1.4	graphics.library/WaitTOF()	2
1.5	Copper Wait Commands	3

# Chapter 1

## HowToCode7

### 1.1 HowToCode: Programming the Copper

Programming the Copper coprocessor  
-----

The copper is probably the most wonderful part of the Amiga architecture. It's so simple to program (it has only three different instructions), yet it is powerful enough to create the most complex displays.

Currently the Amiga OS is very limited in allowing programmers access to the copper, and as most demos rely on the power of the copper, most demo coders ignore the OS and build their own displays directly with their own copperlists.

- 1 Proper Copper Startup
- 2 Copper Wait Bugs

### 1.2 Proper Copper Startup

Proper Copper startup  
-----

(Please look at the startup example code in startup.asm ).

If you are going to use the copper then this is how you should set it up. The current workbench view and copper address are stored, and then the copper enabled. On exit the workbench view is restored.

This guarantees(\*) your demo will run on an AGA (Amiga 1200/4000) machine, even if set into some weird screen mode before running your code.

Otherwise under AGA, the hardware registers can be in some strange states before your code runs, beware!

The LoadView(NULL) forces the display to a standard, empty position, flushing the rubbish out of the hardware registers: Note. There is

---

a bug in the V39 OS on Amiga 1200/4000 and the sprite resolution is \*not\* reset, you will have to do this manually if you use sprites, see Resetting AGA sprite information .

Two WaitTOF() calls are needed after the LoadView to wait for both the long and short frame copperlists of interlaced displays to finish.

It has been suggested to me that instead of using the GfxBase gb\_ActiView I should instead use the Intuition ib\_ViewLord view. This will work just as well, but there has been debate as to whether in the future with retargetable graphics (RTG) this will work in the same way. As the GfxBase is at a lower level than Intuition, I prefer to access it this way (but thank's for the suggestion Boerge anyway!). Using gb\_ActiView code should run from non-Workbench environments (for example, being called from within Amos) too...

\* - Nothing is ever guaranteed where Commodore are involved. They may move the hardware registers into chipram next week :-)

### 1.3 graphics.library/LoadView()

LoadView -- Use a (possibly freshly created) copper list to create the current display.

```
LoadView( View )
-222      A1
```

```
void LoadView( struct View * );
```

IN:

View - a pointer to the View structure which contains the pointer to the constructed coprocessor instructions list, or NULL.

If the View pointer is non-NULL, the new View is displayed, according to your instructions. The vertical blank routine will pick this pointer up and direct the copper to start displaying this View.

If the View pointer is NULL, no View is displayed, and the hardware defaults back to standard chipset defaults (mostly).

Even though a LoadView(NULL) is performed, display DMA will still be active. Sprites will continue to be displayed after a LoadView(NULL) unless an OFF\_SPRITE is subsequently performed.

### 1.4 graphics.library/WaitTOF()

WaitTOF -- Wait for the top of the next video frame.

```
WaitTOF()
-270
```

```
void WaitTOF( void );
```

Wait for vertical blank to occur and all vertical blank interrupt routines to complete before returning to caller.

## 1.5 Copper Wait Commands

Copper Wait Commands

-----

The Hardware Reference manual states a copper wait for the start of line xx is done with:

```
$xx01,$fffe
```

However (as many of you have found out), this actually triggers just before the end of the previous line (around 4 or 5 low-res pixels in from the maximum overscan border).

For most operations this is not a problem (and indeed gives a little extra time to initialise stuff for the next line), but if you are changing the background colour (\$dff180), then there is a noticeable 'step' at the end of the scanline.

The correct way to do a copper wait to avoid this problem is

```
$xx07,$fffe.
```

This just misses the previous scanline, so the background colour is changed exactly at the start of the scanline, not before.