Part 2 : WhooWhoo, Coding.

Well, now we're off to actual Saturn coding. The first thing to do is to create a proper directory for our project. As this evolves making changes to the MAKEFILE file, take good note where you put it. A good choice will be "C:\dev\SATURN\Sgl\MyProj\proj1\". Why did I add an extra directory (and not simply C:\dev\SATURN\Sgl\proj1)? Well, because we need to copy 2 things:

The first is a directory that resides in the Sample directory, and it's called Common. It has workspace information for the SGL. Copy it (and the contents in it) to the "MyProj" directory, so that you have a "MyProj \Common" as well.

Next is the file called MAKEFILE. You have it in the Sgl/README directory, only renamed SGLMKFL. Or you can pull one out of the Samples dir. The first option is recommended, tough, as it is supposed to "work with almost all gcc releases". Whatever solution you choose, copy it (and rename it if you have to) so you have a file called MAKEFILE in the proj1 directory.

Now you have almost everything done. Let's write our basic C file, and we'll get back to setting up the remaining file after.

Open your favourite C editor (I assume it's not Notepad or Word :p). Write this program in:

```
// Simple SGL test

#include "sgl.h"

void     ss_main(void)
{
slInitSystem(TV320x224, NULL, 1);
slPrint("Hello, Saturn.",slLocate(1,1));
while(1)
        {
        slSynch();
        }
}
```

And save it as main.c in the proj1 dir. Open a new file and type this text:

```
# source_program
SRCS = main.c

SYSOBJS = $(CMNDIR)/cinit.o $(SGLLDR)/sglarea.o
OBJS = $(SRCS:.c=.o)
LIBS = -lsgl
```

Now, save it as Objects, in the proj1 directory. The objective of this file, is to avoid that you mess with the main makefile. When you want to compile programs bigger than 1 .c file (most likely), you just add the extra files in the SCR line with spaces between them... A program with 2 files, for example, would look like this:

```
SRCS = main.c func.c
```
(don't add header files, of course)

Don't forget, use always lowercase for the names. This is important because the compiler figures out the type of file by the extension, and a .C file is different from a .c file.

You are now ready to compile. Go to the MS-DOS Command Prompt you created in the previous lesson, don't forget to run SETENV.BAT. Go to the D:\dev\SATURN\Sgl\MyProj\proj1 and type MAKE. It should compile successfully. The result are 3 files. Sl.map (a pre-processing file) Sl.cof (a pseudo-binary) and Sl.bin (the really completed binary file)

In case of an Ultraedit setup, you could configure a command to compile. You need to go into the advanced menu and select tool configuration. There you can input a bat file with the make command, and possibly a pause so you get to see the output. Then just press insert.

Sure, but how to you see your handiwork? Well, there are 3 options :

Use a PC Saturn Emulator:

Use GiriGiri. You need a average computer, DirectX 8 and some kind of 3D card, I believe. You also need a ROM file, but I expect you to figure out those details by yourself.

Open GiriGiri, you'll get a small window with 3 big buttons. Now, maximise the window (it's the middle button on the top-right :p) and voilá, your tiny little emulator becomes a huge and complex Debugger (I figure that's why it's called GiriGiri Debugger ^o^).

What is that number on the Load Binary part?

The number 06000400 is the address where the file will be placed. The "Reset PC" checkbox means that the emulation will proceed at that address, or that your code will boot immediately after you click Go. We will change those values later to pretend that files were loaded in various sections (we will also learn to load files from CD too). All SGL executables run at that address.

Press the “Power On” button. Wait a few seconds (or minutes, depending on your PC speed) until it gives the expression “Time Passed...” on the left debugger window. Press “Pause” on the floating window on the right (it’s somewhat hidden by the size, resize it)
Then go to the f ile menu, select “Load Binary File”, pick the Sl.bin you created, and click OK.
Press “Go” on the left window. The program runs in the emulator window.

Upload to Saturn with an Action Replay:

Another option, in case you have an Action replay, a parallel cable and a PCCommslink card, is to download the executable directly to the Saturn. You need to have all of this working and then just use one of the programs like ssfexe.exe (these can be found on the saturndev page by Chris Barker) and use it to downloa d the executable to the actual hardware. To do this, boot the saturn with no CD, closed and the AR inserted and connected to the PC via the Card. Then execute the program and set it up. Select the sl.bin file and voila! Execute it . The standard address for a C program is 06004000, just some info.

Make a Saturn CD image.

Another option is to use buildcd and strip iso (from the cdtools package usually on PSX devel sites) to create an actual CD image to boot directly on the Saturn after burning it. In this case I highly recommend that you use CD-RW… although no responsibility is assumed for this process. You can find info on how to do that at the Sega Xtreme site. You also need to be able to do the swap trick (be really good at this and use a dummy file is recommended as in the following example), or have a cough moded cough saturn.

In this case you need to create a file like this:

```
Disc    CDROM TEST.DSK

LeadIn MODE1
PostGap 150
EndTrack

Track MODE1
        Pause 150
        Volume ISO9660 KDMP.DSK

        SystemArea "ip.bin"

        PrimaryVolume

        SystemIdentifier        "SEGA SEGASATURN"
        VolumeIdentifier        "CONTROL"
        VolumeSetIdentifier     "Kordamp"
        LogicalBlockSize        2048
        PublisherIdentifier     "SEGA ENTERPRISES, LTD."
        DataPreparerIdentifier  "SEGA ENTERPRISES, LTD."
        ApplicationIdentifier   "CONTROL"
        CopyrightFileIdentifier "CPY.TXT"
        AbstractFileIdentifier  "ABS.TXT"
        BibliographicFileIdentifier  "BIB.TXT"

        Lpath              ; Path tables as specified for Saturn
        Mpath

        Hierarchy
                File    0SL.BIN;1
                 Source SL.BIN
                EndFile

                File    ABS.TXT;1
                 Source ABS.TXT
                EndFile

                File    BIB.TXT;1
                 Source BIB.TXT
                EndFile

                File    CPY.TXT;1
                 Source CPY.TXT
                EndFile

                File    Doc.pdf;1
                 Source H:\Doc.pdf
                EndFile

        EndHierarchy
```

```
EndPrimaryVolume
EndVolume
PostGap 150
EndTrack

Leadout CDDA
 Empty  500
EndTrack
EndDisc
```

With the extension cti (ex image.cti). This is the normal input file for buildcd. This process is a bit more complicated.

First you need to get an IP.BIN file. This can be extracted from an actual Saturn disc, reading the first bytes of the CD. Its length is 4860 bytes. A suitable tool to achieve this is CDRWIN, although that process is beyond the scope of this document. You can also use the one Chris provides on his atclay demo, found on his page. That zip file contains a practical example along with a bat file suitable to execute the buildcd and stripiso (done later to get a really working image).

You can basically take this as skeleton and modify it. It is self explanatory. The only thing here that needs explanation is the pdf file added at the end. It is there because of a common belief that having a bigger iso is better to make the process of swapping easier. I tried it myself and could do it with an image with and without the larger image (it was a 50 MB PDF). It also makes the burn process slower =).
(thanks to Artemio Urbina for the texts)

For now, this will suffice. Digest all this and I hope you understand how to make a workable binary file. Notice that, for the project to work, it needs to be in the SGL tree as specified. Otherwise, a change in the MAKEFILE is needed.

Gambatte Kudasai, minna-san! ^o^