Compiling Train Software with MaRTE

1. Introduction

The following are instructions on how to take your railroad control software and convert it for use on the 486 (or Pentium) PC in ATC620 that runs the train hardware. There are modern PCs in ATC620 that run Windows 7. You can use the RTP one (beside the train rack) to write boot floppies and any to act as a terminal to mercury (using Putty). Compiling and linking must be done on mercury. (If you have a home PC running Linux you could set up MaRTE there but patches are required to make Swindows, Slogger, mgnatmake work.)

The MaRTE cross-compiler uses GNAT GPL 2009 for 686 Linux and you can use GNAT to develop normal Ada95/Ada05 programs on mercury, but MaRTE provides libraries and scripts to target a bare PC. The final executable, named **mprogram**, is a boot image that contains the run-time kernel, MaRTE_OS. as well as what you wrote.

The MaRTE commands are similar to the normal GNAT commands except that they have 'm' in front. For example, **mgcc –c my_package.adb** (instead of gcc –c etc). Note that the command without the 'm' will generally start off ok but then produce errors about missing library files. **In fact mgcc is not very useful…**

You will mostly use **make** – a suitable Makefile is provided to orchestrate the mgcc, mgnatmake and objcopy commands. This ends up with an executable file always named **mprogram**. It stores this in the current directory. To run it you boot the train 486 in ATC620 using a special boot disk. See "MaRTE Boot Disk".

You can download the "MaRTE User's Guide", **marte_ug.htm**, from Blackboard. See also http://marte.unican.es

2. Details

2.1 Setting up

- Ensure your railroad code is compatible with the current Simrail2 (version 2.2.1 or later) – the ZIP is on Blackboard.
- Make a copy of the source files and assuming your program uses Swindows, comment out all Text_Io and Int_Io Puts, Put_Lines and New_Lines except those in exception handlers. This output would get mixed up with Swindows as it would be on the same screen. Retest: the only normal MSDOS output should be from Simrail2/Swindows.
- (Optional) rename your source filenames to lower case now – actually it is easier to do this using WinSCP later.
- Log on to mercury (`mercury.it.swin.edu.au`) using **putty** (from a Windows PC) or **ssh** (from a Linux workstation) into your personal account, e.g. s1234567.

- (First time) Create a subdirectory on mercury for this subject. Then set up your environment by typing
  `hit3047`
  which should display a message about being set up for MaRTE. To confirm this
  `echo $GIVEN`
  should display the directory name
  `/home/students/subjects/hit3047/common/given`
  If the **`hit3047`** command doesn't work then edit your **.bash_profile** file. Add the line
  `alias hit3047='. /home/students/subjects/hit3047/common/marte.sh'`
  (Note the single-quotes and dot above.) Then log out and log back in again.

- Normally type
  `hit3047`
  each time you log in and want to use Ada (rather than GNU C/C++).
- Copy up your source files from your Windows PC using **WinSCP** (ensure that you transfer in text mode, preserve timestamp and that filenames are forced to lower-case).
  or (later)
  retrieve copies from your team's CVS or SVN or GIT repository on mercury.
- Remove all irrelevant files: **`rm adag*; rm sim*; rm io_*; rm swin*`**
- Copy the required files from the directory referred to by $GIVEN, ie
  `cp –p $GIVEN/* .`
  (Don't forget the dot. The "-p" is to preserve timestamp.)
- Note that the given files include `dio192defs.ads` and other `*defs.ads` that are the same as those in simrail2_src_2*.zip. However `unsigned_types, swindows, slogger, logger_ada, halls2` and `io_ports` are different.
  Beware: Don't accidentally have two versions, eg Swindows.adb from your windows files and swindows.adb from $GIVEN.
- If needed use **`mv`** commands to rename all your source files so their names are entirely lower-case. (You should use mixed-case names inside the sources – that's good style.)

2.2 Editing

- You can use the Editor provided with WinSCP – just double click files on mercury and an Edit window pops up on your Windows PC.
- You can also use **`gedit`** (or similar) from the Gnome or KDE environment on a Linux PC.
- You can use **`vi`** (i.e. vim) on mercury via a **putty** window. If so I strongly suggest you place in your mercury home directory a file named **`.exrc`** that contains these lines:
  ```
  set autoindent
  set background=dark
  set number
  ```
- Using any editor/IDE under Windows, you can edit a copy of a file on mercury and when finished transfer it to mercury using WinSCP.

2.3 What to Edit

- Edit your main program to remove references to Simrail2. (I recommend you simply comment them out so you can move your code back to Simrail2 later.)
- Similarly remove references to Simrail2 from the other source files.
- Important: Edit the file **Makefile** (provided in $GIVEN) to name your main program as the value of **`YOUR_PROG`**. (Lower-case without file extension.)

2.4 Compiling

- Note: to compile you **must be** on mercury via a putty or ssh window and have issued the hit3047 command.
- Start with Hello_World (compile, bind and link):
  `mgnatmake hello_world.adb`
  and run it on the 486 as outlined below. [mgnatmake is a script in marte/utils that calls gnatmake which calls mgcc and gnatlink etc.]
- Then compile a defs package:
  `mgnatmake dio192defs.ads`

This may produce a message "Compilation Error" even though there are no errors but there is a warning "No code generated for ..." This is ok.

- As described in the Introduction, instead of direct use mgnatmake, mgcc etc we use Linux's **make** system.
  - Edit Makefile so YOUR_PROG has the name of your program (2.3 above).
  - There are some targets that aren't programs. For example, io_ports is there so you can issue the command 'make io_ports' instead of 'mgcc –I..(long list of directories) -c io_ports.adb'.
  - Edit Makefile as appropriate for packages you are working on.
  - Then ...
- Compile some bodies (that are explicitly in Makefile), eg

  `make io_ports`

- Compile, bind & link your main (compiling anything needed) and reducing the execuatble's size (so it fits on a floppy):

  `make my_main`

  or, if you have set up the default target correctly, just

  `make`

  If it finishes with an error like:

  ```
  ln my_main mprogram
  ln: `mprogram': File exists
  make: *** [my_main] Error 1
  ```

  then it simply means you should have removed (rm) mprogram first. Do so and type the ln command manually. (2014 extra rm added to Makefile to fix this.)

- It is possible that your code tries to use GNAT or MaRTE library packages that have never been compiled. These include attempts to find logger_ada.adb in marte/misc. Usually this is not a problem as their .ali and .o files, just a few, should appear in your current directory. If the make fails and there are many .ali and .o files that look as if they belong to marte kernel then there may be a problem.
  - The most likely cause is that you have used a package name that is the same as one in marte. Look first in /opt/marteos/marte/misc – generally packages elsewhere have names that start with ada or marte or posix. Likely problem names are **keyboard, signal, types** and **debug**. (If the problem persists please tell me! Rob)
- A common error is to upload logger_ada.ads and logger_ada.adb. The latter is only needed with simrail2 -- the marte version of this package does not need a body and the correct .ads is in $GIVEN.
- Another common error is that you are editing one file and compiling another. Check that **the files on mercury are actually the latest versions.** Have you accidentally uploaded files relevant to Windows (adagraph and simrail) maybe overwriting yesterday's mercury versions?
- Use a (subversion) repository on mercury to back up your work!
- If the compiler warns you about delay statements (or "potentially blocking") within protected objects then restructure your code – do not ignore the warnings! This is a logic/design problem with your code.
- Past issues:
  - (2004) Unsigned_Types was introduced to provide portability and overcome a blunder in the MaRTE code. The version on mercury is different from the version for Windows.

- (2007) There was a spurious compiler error that reported a problem with Unsigned_8 &/or Unsigned_16 in interrupt_tables.ads, for example. This was related to the placement of "use Unsigned_Types;" in student code. It was cured by moving occurrences of this statement into package bodies (rather than ahead of the word "package") or deeper into subprogram bodies.
- (2008) The defs packages and io_ports were revised to minimize the need for such edits.

2.5 Testing in ATC620

- You need to boot the 486 PC from a “fake netboot” floppy disk. See “MaRTE Boot Disk 2013” which discusses copying your mprogram to a floppy.
- While the program is loading one can leave the electronics powered or unpowered. Turn on the power supplies using the big square orange switch on the Power Supply box (black). When your program shows its Swindows screen, if required, issue **your** Reset (R) command to initialise all the cards and electronics (you might do this automatically).
- Assuming there are bugs you will want to edit and recompile. Let someone else use the 486 while you do so. Power off the system in reverse order to the above: big orange switch off, 486 off. Of course power off when you leave.

2.6 Testing with VMware

- HIT3047>MARTE is/was one of the virtual PCs available via VMware. It is possible to configure this to boot from a floppy image file, grubfloppy.img, and to write to a pipe instead of a serial port. The default configuration has Tetris in its mprogram.
- This is not very useful for students as it doesn’t have simulated electronics nor trains, but has been useful for me in developing swindows and logging.
- See help/Doc_vm_student_hit3047 available via VmLauncher> HIT3047

Please ask me about anything in this document that is unclear.

Dr Rob Allen (2015-04-08)
Faculty of Science Engineering & Technology,
Swinburne University of Technology