Programming with TI-Nspire

Julian M.

Table of contents

1 Getting started

1.1 Differents: TI-Basic vs. C

These are some examples I picked to show some differents between TI-Basic and C.

Feature	TI-Basic	C
Functions	~	✓
On-Calc programming	✓	✗
Get pressed keys	✗	✓
Pixelwise drawing on screen	✗	✓
Timers	✗	✓
RS232 Serial port	✗	✓

Another point is that C is a lot faster than TI-Basic.

1.2 What is required?

Basic C knowledge

- No need for explanation; There are much tutorials out in the internet, just google it

Ndless sourcecode

- The sourcecode is required to build ndless

Nspire emulator

- The Nspire emulator is recommended to test all your programs directly on your PC

Text editor

- I will use Notepad++ in this Tutorial, but you can use every text editor you want

Some extra programs

- MSYS
- YAGARTO
- 7-Zip
- Tortoise SVN

2 Installation

2.1 MSYS

Just follow all instructions and install it to `C:\msys\1.0`

2.1.1 Adding MSYS to the PATH variable

Open the Start menu and choose "Run…", type `cmd` and hit enter.

Type the following into the console:

```
set %PATH%=%PATH%;C:\msys\1.0\bin
```

2.2 YAGARTO

Follow the instructions and install it to a destination of your choose.

Make sure "Add YAGARTO to the PATH variable" is checked in the "Choose Components"-Window!

2.3 7-Zip

Follow the instructions and install it to a destination of your choose.

2.3.1 Adding 7-Zip to the PATH variable

Open the Start menu and choose "Run…", type `cmd` and hit enter.

Type the following into the console:

```
set %PATH%=%PATH%;<7-Zip Installation Folder>
```

Replace *<7-Zip Installation Folder>* with your installation folder, for example `C:\Program Files\7-Zip`.

2.4 Tortoise SVN

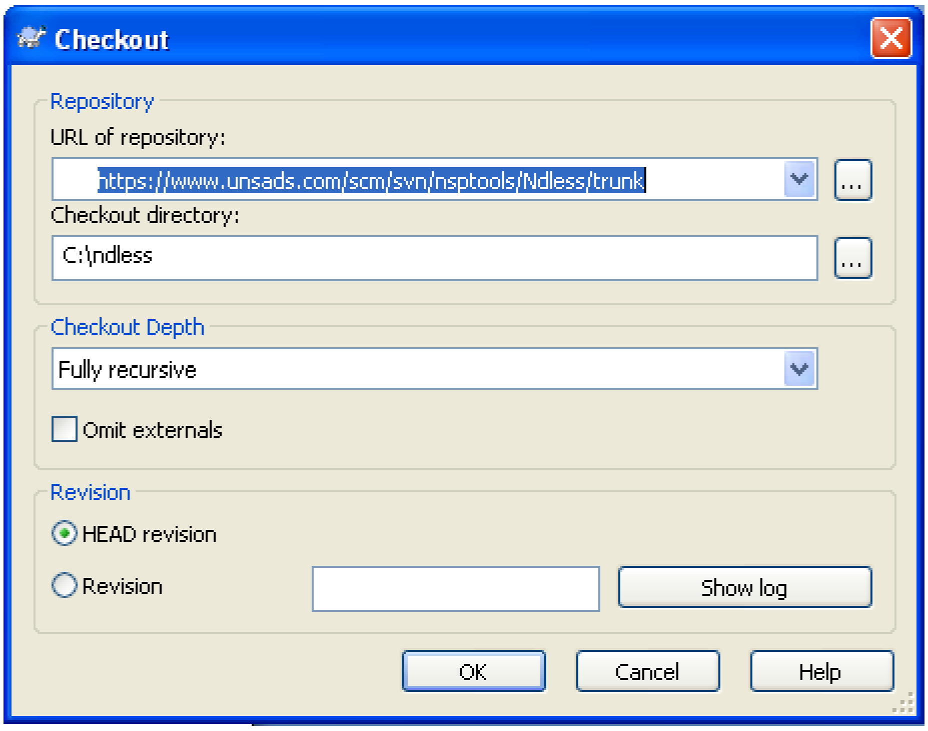
Tortoise SVN is not especially required; you can use any SVN client you want, but I will use Tortoise in this Tutorial.

Just follow all instructions and install it to a destination of your choose.

3 Building ndless

3.1 Check out the source code

Create a folder called “ndless” on your system drive and right-click it. Choose “SVN Checkout…” and enter `https://www.unsads.com/scm/svn/nsptools/Ndless/trunk` in “URL of repository”. You should end up like this:



URL of repository: https://www.unsads.com/scm/svn/nsptools/Ndless/trunk
Checkout directory: C:\ndless
Fully recursive
Omit externals
HEAD revision
Revision

Image 1: SVN Checkout

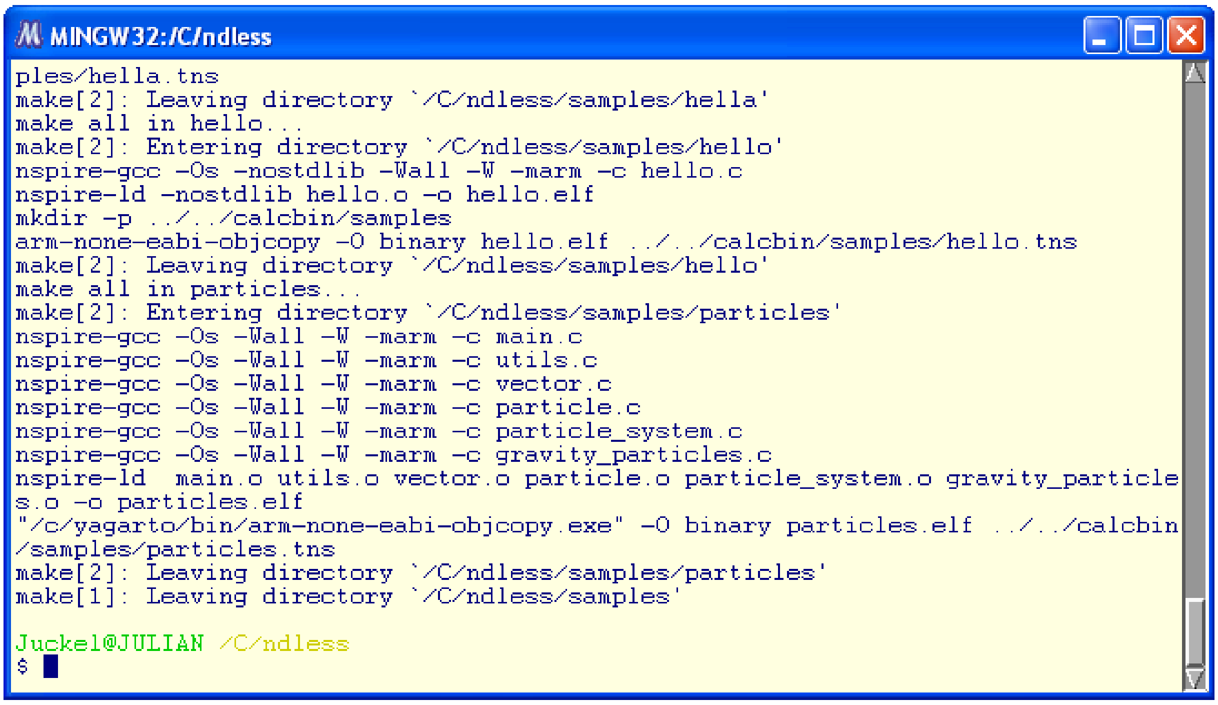
Now just click “OK”. You will be asked for username and password; Use `guest` for both.

After downloading the source code, you need to add the path where you installed ndless (e.g. `C:\ndless\bin`) to your PATH variable. Do it like you did before in 2.1.1 and 2.3.1.

3.2 Make!

Open Start→Program Files/All Programs→MinGW→MSYS→msys (rxvt). Type `cd /C/ndless` and press enter. Now type `make`. It may take a while to finish.

If your output looks like this:



```
ples/hella.tns
make[2]: Leaving directory `/C/ndless/samples/hella'
make all in hello...
make[2]: Entering directory `/C/ndless/samples/hello'
nspire-gcc -Os -nostdlib -Wall -W -marm -c hello.c
nspire-ld -nostdlib hello.o -o hello.elf
mkdir -p ../../calcbin/samples
arm-none-eabi-objcopy -O binary hello.elf ../../calcbin/samples/hello.tns
make[2]: Leaving directory `/C/ndless/samples/hello'
make all in particles...
make[2]: Entering directory `/C/ndless/samples/particles'
nspire-gcc -Os -Wall -W -marm -c main.c
nspire-gcc -Os -Wall -W -marm -c utils.c
nspire-gcc -Os -Wall -W -marm -c vector.c
nspire-gcc -Os -Wall -W -marm -c particle.c
nspire-gcc -Os -Wall -W -marm -c particle_system.c
nspire-gcc -Os -Wall -W -marm -c gravity_particles.c
nspire-ld  main.o utils.o vector.o particle.o particle_system.o gravity_particle
s.o -o particles.elf
"/c/yagarto/bin/arm-none-eabi-objcopy.exe" -O binary particles.elf ../../calcbin
/samples/particles.tns
make[2]: Leaving directory `/C/ndless/samples/particles'
make[1]: Leaving directory `/C/ndless/samples'

Juckel@JULIAN /C/ndless
$
```

Image 2: MSYS

You have done everything correctly.

4 Emulating the Nspire

Now that you have built ndless, you should download and set up nspire_emu.

4.1 Things you need

nspire_emu

- The emulator itself; Used to test your C programs.

imgdump

- Used to create raw boot2 images needed by nspire_emu.

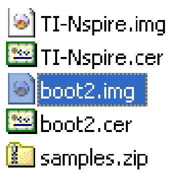
OS 2.1 Image file

- The basic OS needed to run on the emulator.

4.2 Setup

4.2.1 Create a raw image file

At first open the downloaded OS file with 7-Zip (Right-click→7-Zip→Open). These files should be included in the image:



```
TI-Nspire.img
TI-Nspire.cer
boot2.img
boot2.cer
samples.zip
```

Image 3: OS files

Now extract the selected file (`boot2.img`) to the `imgdump` folder, open a console (Start→Run…→cmd) and type:

```
cd <imgdump_path>
imgdump.exe boot2.img
```

Replace *<imgdump_path>* with the path where your `imgdump` is located. This will produce a file called `boot2.img.raw`. Copy it to your `nspire_emu` folder.

4.2.2 Install the OS

To create a flash image with preinstalled OS, type into the console:

```
cd <nspire_emu_path>
nspire_emu.exe /N /F=flash.bin /PO=<OS_Image_File_name>
```

Replace *<nspire_emu_path>* with the path where `nspire_emu` is located and *<OS_Image_File_name.tno>* with the name of your downloaded OS 2.1 Image file name.

Make sure to add /C to the console command if you are using a CAS OS.

Now you are ready to boot your emulated Nspire and install the OS. To make starting the emulator a lot easier, you should create a batch (`*.bat`) file in your emulator's directory that contains:

```
nspire_emu.exe /B=boot2.img.raw /F=flash.bin
```

Now you can just double-click this file and the emulator will boot your nSpire. When you start it for the first time, you have to press "I" when you are asked to. After the OS is installed (and you have chosen your language and font size) you need to save your flash to make sure you do not have to install the OS again every time the Nspire boots. To do so, go to File→Save Flash.

4.2.3 Preventing crashes

Every time the calculator tries to go in standby (normally after three minutes of inactivity), the emulator will freeze. To prevent those crashes, you can set the standby time to 30 minutes. Go to Settings & Status→Handheld Setup… and change the Power Standby time. After doing this, save the flash again.

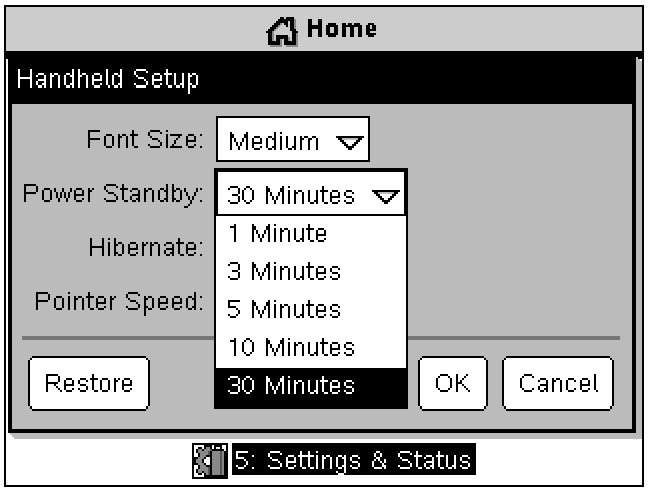
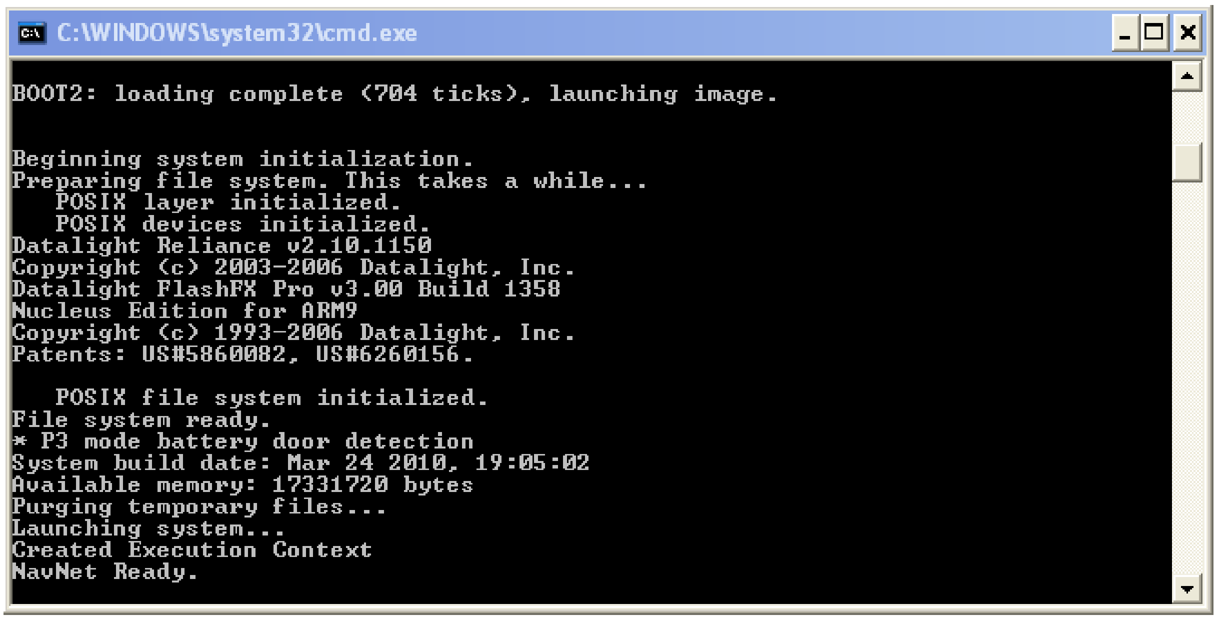

Home
Handheld Setup
Font Size: Medium
Power Standby: 30 Minutes
Hibernate:
Pointer Speed:

Image 4: Setting the standby time

4.3 Overview

Now you should have two windows. The first, the RS232 console, shows all the stuff the calc sends trough the RS232 interface[1]; The second is the calculator itself, including keypad and screen.

[1] I will explain the RS232 interface later in this tutorial.



```
C:\WINDOWS\system32\cmd.exe
BOOT2: loading complete (704 ticks), launching image.

Beginning system initialization.
Preparing file system. This takes a while...
   POSIX layer initialized.
   POSIX devices initialized.
Datalight Reliance v2.10.1150
Copyright (c) 2003-2006 Datalight, Inc.
Datalight FlashFX Pro v3.00 Build 1358
Nucleus Edition for ARM9
Copyright (c) 1993-2006 Datalight, Inc.
Patents: US#5860082, US#6260156.

   POSIX file system initialized.
File system ready.
* P3 mode battery door detection
System build date: Mar 24 2010, 19:05:02
Available memory: 17331720 bytes
Purging temporary files...
Launching system...
Created Execution Context
NavNet Ready.
```

Image 5: RS232 Console

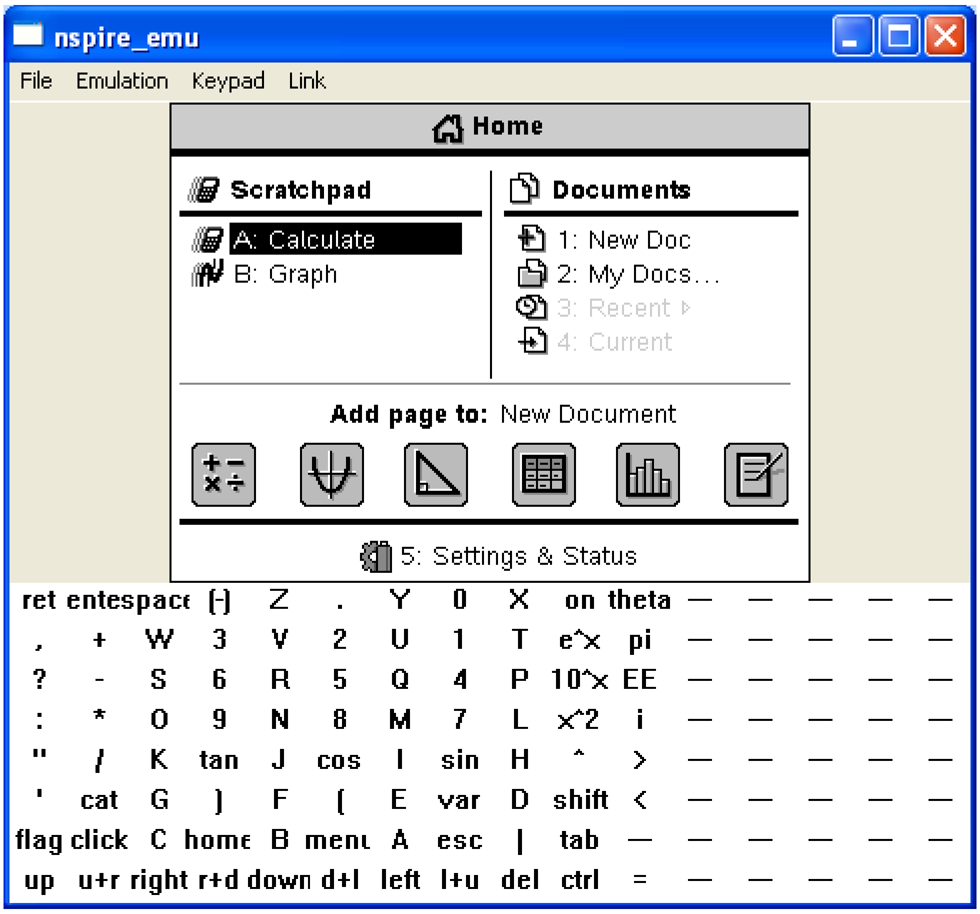
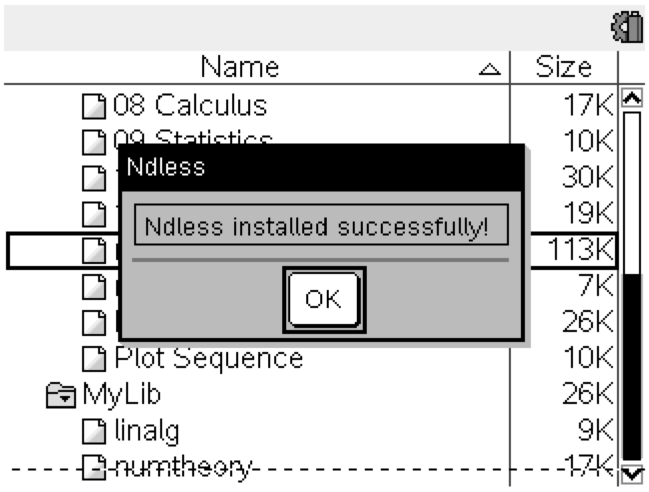

nspire_emu
File Emulation Keypad Link
Home
Scratchpad
A: Calculate
B: Graph
Documents
1: New Doc
2: My Docs...
3: Recent
4: Current
Add page to: New Document
5: Settings & Status
ret entespace (-) Z . Y 0 X on theta
, + W 3 V 2 U 1 T e^x pi
? - S 6 R 5 Q 4 P 10^x EE
: * O 9 N 8 M 7 L x^2 i
" / K tan J cos I sin H ^ >
' cat G) F (E var D shift <
flag click C home B menu A esc | tab
up u+r right r+d down d+l left l+u del ctrl =

Image 6: Calculator screen

4.4 Installing ndless

To install ndless, you need to send it to your emulated calculator first. Open Link→Connect to “plug the USB cable in”. Your RS232 console should output `usblink connected`. Now you can send the ndless files by choosing Link→Send Document...; the required files are named `ndless_installer_os-2.1.0.tns` and `ndless_resources.tns`. They are located in `<ndless_path>/calcbin`. Replace *<ndless_path>* with the path of your compiled ndless. After both files are sent, save the flash again. Now go to My Documents, select `ndless_installer_os-2.1.0` (should be located in Examples) and press Enter. After a successful installation a message will appear:


Ndless

Ndless installed successfully!

Image 7: ndless installation

If random pixel lines show up and the calculator freezes, the installation failed and you need to try again. If the calculator does not restart itself, you need to reset the CPU (Emulation→Reset CPU). The Nspire will restart then.

5 Your first program

Now that you know how to send files to the emulator, we will code our first C program. Create a new folder for this project.

5.1 The Makefile

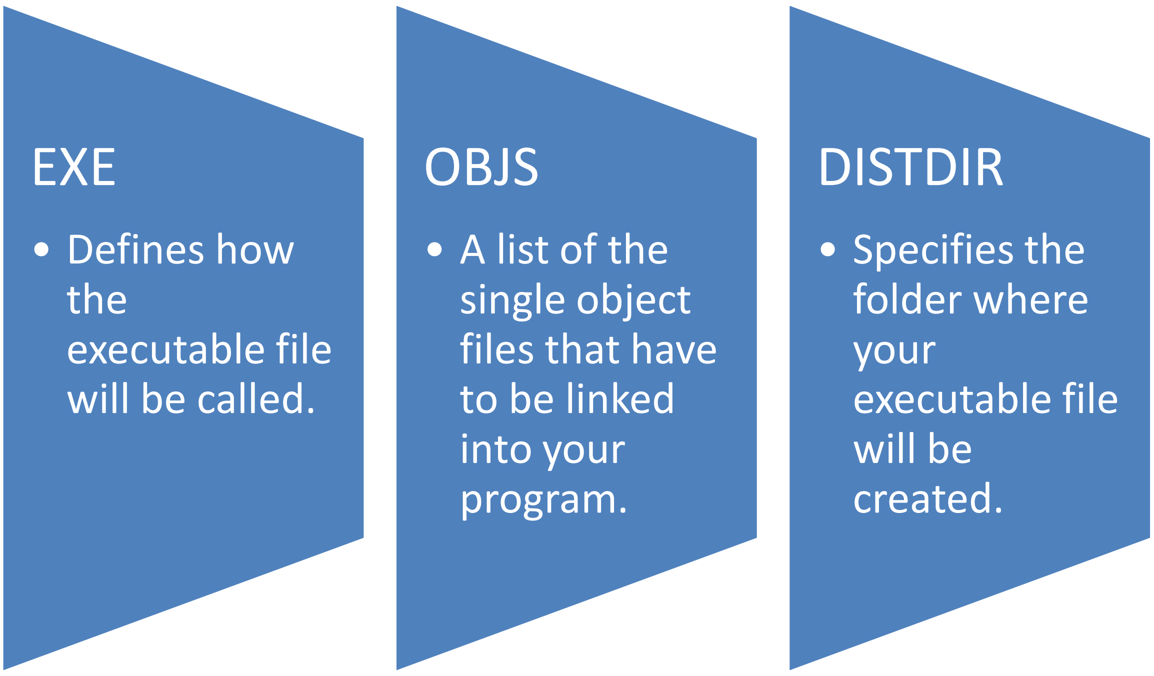
Every program you might code will need this file. Some important settings are located in it. An example Makefile can be found in ndless's `samples/hello` directory. Copy it in your folder.

A Makefile is not necessarily needed, but it will make compiling programs a lot easier.

```
GCC = nspire-gcc
LD = nspire-ld
GCCFLAGS = -Os -nostdlib -Wall -W -marm
LDFLAGS = -nostdlib
OBJCOPY := "$(shell which arm-elf-objcopy 2>/dev/null)"
ifeq (${OBJCOPY},"")
	OBJCOPY := arm-none-eabi-objcopy
endif
EXE = hello.tns
OBJS = hello.o
DISTDIR = ../../calcbin/samples
vpath %.tns $(DISTDIR)

all: $(EXE)

%.o: %.c
	$(GCC) $(GCCFLAGS) -c $<

$(EXE): $(OBJS)
	$(LD) $(LDFLAGS) $^ -o $(@:.tns=.elf)
	mkdir -p $(DISTDIR)
	$(OBJCOPY) -O binary $(@:.tns=.elf) $(DISTDIR)/$@

clean:
	rm -f *.o *.elf
	rm -f $(DISTDIR)/$(EXE)
```

You do not have to know what all these lines do, but some of them are important:

```
EXE = hello.tns
OBJS = hello.o
DISTDIR = ../../calcbin/samples
```



EXE

- Defines how the executable file will be called.

OBJS

- A list of the single object files that have to be linked into your program.

DISTDIR

- Specifies the folder where your executable file will be created.

So the these three lines say that the executable file will be called `hello.tns` and consists of `hello.o`. It will be located in the folder `../../calcbin/samples`. Because we want to have the executable in our directory, change the `DISTDIR` to `.`.

5.2 The source code

Create a new file called `hello.c`. Write these lines:

```
#include <os.h>

int main(void)
{
	puts("hello world!");
	return 0;
}
```

5.2.1 Explanation

This is a VERY basic program, so I think I do not have to say much.

```
#include <os.h>
```

Includes the file `os.h`. It is required by every ndless program.

```
int main(void)
```

Declares the `main`-function of our program. You should now that.

```
puts("hello world!");
```

Sends `hello world!` Through the RS232-Interface.

```
return 0;
```

Returns zero. Ends the program.

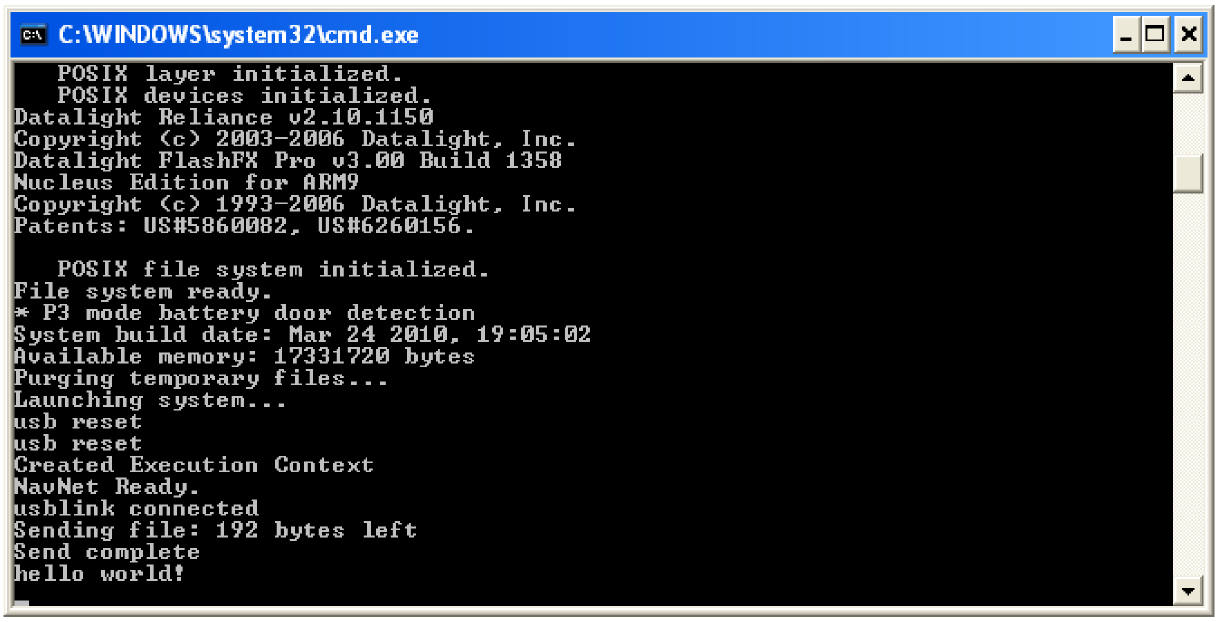
5.3 Compilation

Open up `msys (rxvt)`, navigate to your folder (using `cd`) and type `make`.
On a successful compilation your output will look like this:

```
nspire-gcc -Os -nostdlib -Wall -W -marm -c hello.c
nspire-ld -nostdlib hello.o -o hello.elf
mkdir -p .
arm-none-eabi-objcopy -O binary hello.elf ./hello.tns
```

5.4 Hello World!

Start the emulator, install ndless and send the file `hello.tns`. Now just click on it in the `My Documents` screen. A message will appear in the RS232 console:



```
   POSIX layer initialized.
   POSIX devices initialized.
Datalight Reliance v2.10.1150
Copyright (c) 2003-2006 Datalight, Inc.
Datalight FlashFX Pro v3.00 Build 1358
Nucleus Edition for ARM9
Copyright (c) 1993-2006 Datalight, Inc.
Patents: US#5860082, US#6260156.

   POSIX file system initialized.
File system ready.
* P3 mode battery door detection
System build date: Mar 24 2010, 19:05:02
Available memory: 17331720 bytes
Purging temporary files...
Launching system...
usb reset
usb reset
Created Execution Context
NavNet Ready.
usblink connected
Sending file: 192 bytes left
Send complete
hello world!
```

Image 8: Hello World!