This manual, written on 7 June 2004 by Pierre Pomiers <pierre@robosoft.fr> is version 1.0. It contains Robosoft Development Toolchain features and howto’s.

Robosoft Development Toolchain

Pierre Pomiers
# Table of Contents

Robosoft Development Toolchain: An Introduction ........................................... 1

1 Hardware ........................................ 2

2 Requirements .................................... 3

3 Installation .................................... 4
  3.1 Installing CAN hardware ............................. 5
    3.1.1 General purpose remarks .................. 6
    3.1.2 Installing the PCI 7841 board ........... 6
    3.1.3 Installing the CAN Dongle ............... 7
      3.1.3.1 Using parport as modules ............ 8
      3.1.3.2 Using kmmod .......................... 8
    3.2 Preparing Linux kernel for RTAI installation ... 9
    3.3 Installing and testing RTAI kernel .......... 10
    3.4 Installing the MPC555 cross compiler .......... 12
    3.5 Installing the C math library support ...... 13
    3.6 Installing SynDEx CAD ...................... 14
    3.7 Development tree organisation ............. 14

4 Setting toolchain variables ................. 16
  4.1 Bash environment variables .................. 16
  4.2 Application makefile rules ................... 16
    4.2.1 Linux paths ................................ 16
    4.2.2 RTAI paths ................................ 17
    4.2.3 Toolchain macros, compiler and utils paths .... 17
    4.2.4 Application name variable ............... 17
    4.2.5 Application specific dependencies .......... 17
    4.2.6 Additional makefile rules ................ 18

5 Developping applications .................... 19
  5.1 Reminder: the SynDEx compilation sequence .... 19
  5.2 About Robosoft development toolchain utilities .... 19
    5.2.1 SynDEx download tools: the ‘elf2sdxbin’ and 
        ‘bin2rtfifo’ utilities .......................... 20
    5.2.2 Another download tool: the ‘dwmbin’ utility ... 20
    5.2.3 Bootloader tools: the ‘bootloader’ utility .......... 22
    5.2.4 Flash tools: the ‘app2flash’, ‘bootapp’ and ‘sdx2img’ 
        utilities ...................................... 23
      5.2.4.1 To be done from SynDEx CAD .......... 23
      5.2.4.2 The standalone application basic principle ... 23
      5.2.4.3 Generating the application image .......... 23
      5.2.4.4 Flashing the application image ........... 25
  5.3 About FDLIBM the C math library support ...... 25
    5.3.1 double acosh (double): ...................... 25
    5.3.2 double acos (double): ...................... 26
    5.3.3 double asinh (double): ..................... 26
    5.3.4 double asin (double): ...................... 27
    5.3.5 double atan2 (double, double): .......... 27
    5.3.6 double atanh (double): .................... 28
    5.3.7 double atan (double): ...................... 29
5.3.8  double cbrt (double); ................................... 29
5.3.9  double ceil (double); ................................... 30
5.3.10 double cosh (double); ................................... 30
5.3.11 double cos (double); .................................... 31
5.3.12 double erf (double); .................................... 31
5.3.13 double erfc (double); ................................... 32
5.3.14 double exp (double); .................................... 32
5.3.15 double fabs (double); ................................... 33
5.3.16 int finite (double); ..................................... 34
5.3.17 double floor (double); ................................... 34
5.3.18 double fmod (double, double); .......................... 35
5.3.19 double frexp (double, int *); .......................... 35
5.3.20 double gamma (double); ................................ 36
5.3.21 double hypot (double, double); ........................ 36
5.3.22 int isnan (double); ..................................... 37
5.3.23 double j0 (double); ..................................... 38
5.3.24 double j1 (double); ..................................... 39
5.3.25 double jn (int, double); ................................ 39
5.3.26 double ldexp (double, int); ............................ 39
5.3.27 double lgamma (double); ................................ 39
5.3.28 double log (double); .................................... 41
5.3.29 double log10 (double); ................................ 41
5.3.30 double modf (double, double *); ..................... 41
5.3.31 double nextafter (double, double); .................... 41
5.3.32 double pow (double, double); .......................... 42
5.3.33 double sinh (double); ................................... 42
5.3.34 double sin (double); .................................... 43
5.3.35 double sqrt (double); ................................... 43
5.3.36 double tanh (double); .................................. 44
5.3.37 double tan (double); .................................... 44
5.3.38 double y0 (double); .................................... 45
5.3.39 double y1 (double); .................................... 45
5.3.40 double yn (int, double); ............................... 45
5.3.41 Sample application ....................................... 45

6  Launching applications ................................. 49

Document Index .............................................. 52
Robosoft Development Toolchain: An Introduction

Robosoft distributes its products with both MPC555 and Intel x86 based architectures. This document aims to present all features of the Robosoft Development Toolchain and give users enough practical keys for using it efficiently. You will also find here several related links to follow for obtaining additional information about tools our distribution is relying on.
Chapter 1: Hardware

1 Hardware

Both Robosoft control system and development platform are based on a fully distributed architectures. Such architectures are composed with computing units, linked together over a communication media network. Supported computing units are:

- x86 family processors (normally Intel processors from 80386 to recent P4 and others AMD or Cyrix clones are supported)
- Motorola MPC555 [See ‘doc/MPC555/MPC555UM/START.pdf’] Robosoft control boards

While, supported communication media are:

- Ethernet for communication between PC computers
- CAN bus for communication between both MPC555 boards and also between MPC555 and PC.

IMPORTANT!!!... Only Philips SJA1000 [See ‘doc/CAN/SJA1000_3.pdf’] -based CAN bus devices are supported. Please use devices of the following references:

- PCI 7841 [See ‘doc/CAN/PCI7841 Manual.PDF’] ADLINK Dual CAN ports
- CAN Dongle [See ‘doc/CAN/MANUAL_CANdolgleV160a-1.pdf’] for ECP/EPP

PCI 7841 [See ‘doc/CAN/PCI7841 Manual.PDF’] reference can only be used on PC with, at least one half PCI port on the backplane. While, CAN Dongle [See ‘doc/CAN/MANUAL_CANdolgleV160a-1.pdf’] interface could be used on any PC equipped with a parallel port (with EPP/ECP [See ‘doc/ECP/ecp_reg.pdf’] capabilities).
2 Requirements

Here is the list of software that needs to be installed on your Robosoft product system. You will also find links to related web site and download pages. Required for realtime support under PC:

- Linux kernel version 2.4.22 (download here)
- RTAI distribution version 24.1.12 (download here)

Required for developing applications for MPC555 (needed to build the cross compiler):

- GNU binutils version 2.14: providing linker ld, assembler as and many other utils (download sources here and read documentation here)
- GNU gcc core version gcc-3.3.2 (download sources here and read documentation here)
- newlib version 1.12.0 (download sources and read documentation here)
- fdlibm version 5.2 (download sources and read documentation here)

Required for designing heterogeneous and distributed applications:

- SynDEx CAD version 6.7.0 (download sources here and read documentation here)
- SynDEx macros for:
  - Robosoft control board
  - Motorola MPC555 microcontroller
  - x86 PC running under Linux/RTAI
  - CAN bus media
3 Installation

All installation procedure shown here assume your development tools directory is organized as follow (e.g. here user home directory is `/home/pierre` and developments done under `last` directory):

```
/home/pierre/syndex/current/last
-- appsv5
  |-- robcuarAna18-0.9
  |-- test-0.0-flash
  |-- test-0.1
  |-- test-0.2
  '-- test-0.4
-- appsv6
  |-- Base128v6-0.1
  |-- floatv6-0.1
  |-- floatv6-0.2
  |-- linuxIOv6-0.1
  |-- linuxIOv6-0.2
  |-- linuxIOv6-0.3
  |-- linuxIOv6-0.4
  |-- linuxIOv6-0.5
  |-- robcuarAna18-1.0-v6.6.8
  |-- robcuarAna18-1.0-v6.6.8_2RD_lo
  |-- robcuarAna18-1.0-v6.7.0
  |-- robcuarAna18-1.0-v6.7.0-flash
  |-- robcuarAna30-0.9-v6.6.8
  |-- robcuarAna30-0.9-v6.6.8-flash
  |-- robcuarPWM30-0.9
  |-- robcuarPWM30-1.0-v6.7.0
  |-- robuter-0.10
  |-- robuter-0.11
  |-- robuter-0.12
  |-- robuter-0.12f
  |-- robuter-1.0f
  |-- robuter-1.1f
  |-- robuter-1.2f
  |-- robuter-1.3f
  |-- swissknife-0.1
  |-- swissknife-0.2
  |-- swissknife-0.3
  |-- testv6-0.1
  |-- testv6-0.2
  |-- testv6-0.3
  |-- testv6-0.4
  |-- testv6-0.5
  '-- testv6-0.6
-- crossgcc
  |-- bin
  |-- fdlibm
  |-- include
  |-- info
  |-- lib
  |-- man
  |-- ppc-elf32
  '-- share
-- doc
  |-- CAN
```
Chapter 3: Installation

```
|-- ECP
|-- FDLIBM
|-- MPC555
|-- ROBO
|-- SDX
|-- pic
|-- ldscripts
|-- macrosv5
  |-- draft
|-- macrosv6
  |-- draft
|-- mini-crossgcc
  |-- bin
  |-- fdlibm
  |-- include
  |-- lib
  |-- ppc-elf32
|-- rtai
  |-- utils
    |-- app2flash
    |-- bin2rtfifo
    |-- bootapp
    |-- bootloader
    |-- dwnbin
    |-- elf2sdxbin
    |-- sdx2img
```

79 directories

The tree given above is organized the same way that the one you will find on the product cdrom. This will help you to recover a new clean development directory in case of problem. Let us note that all tarball required for installation procedures (described in the sections below) are located under the `required` directory.

```
/home/pierre/syndex/current/required
|-- crossgcc
  |-- binutils-2.14.tar.bz2
  |-- fdlibm-5.2.tar.bz2
  |-- gcc-core-3.3.2.tar.bz2
  |-- newlib-1.12.0.tar.gz
|-- linux
  |-- linux-2.4.22.tar.bz2
|-- rtai
  |-- rtai-24.1.13.tar.gz
|-- syndex
  |-- syndex-6.7.0-linux.tgz
  |-- syndex52linux.tgz
```

4 directories, 8 files

### 3.1 Installing CAN hardware

Robosoft development toolchain support two types of CAN device for handling communications between MPC555 boards and PC:
- PCI 7841 ADLINK Dual CAN ports (based on two Philips SJA1000)
- CAN Dongle (based on one Philips SJA1000)
Developing and running applications for Robosoft platforms requires to install only one of these two devices: this means that user receives his platform with the convinient device (either a PCI 7841 board or a CAN Dongle).

### 3.1.1 General purpose remarks

Support for the mentionned CAN devices is not done by classical drivers. The required pieces of software are generated automatically and included into the application binaries. This means that user do not have to install any additional software on his PC. Thus, most user are not concerned by the two subsections below.

Instead, advanced users wanting to build their own system, has to respect the following recommendations, depending on the choice of device they made.

### 3.1.2 Installing the PCI 7841 board

In order to ensure the support of the PCI 7841 board under realtime constraints, you have to be sure the PCI board does not share its interrupt with any other device (ie. there is no IRQ conflict). To check this point, you should display information about the PCI resources. Use the `lspci` command from the shell command line as shown below:

[root@localhost root]# lspci -v

00:00.0 Host bridge: Intel Corp. 82845 845 (Brookdale)
  Chipset Host Bridge (rev 11)
  Subsystem: Intel Corp. 82845 845 (Brookdale) Chipset Host Bridge
  Flags: bus master, fast devsel, latency 0
  Memory at e0000000 (32-bit, prefetchable) [size=64M]
  Capabilities: [e4] #09 [a104]
  Capabilities: [a0] AGP version 2.0

00:01.0 PCI bridge: Intel Corp. 82845 845 (Brookdale)
  Chipset AGP Bridge (rev 11) (prog-if 00 [Normal decode])
  Flags: bus master, 66Mhz, fast devsel, latency 64
  Bus: primary=00, secondary=01, subordinate=01, sec-latency=32
  I/O behind bridge: 00009000-000090ff
  Memory behind bridge: e4000000-e6ffffff

00:1e.0 PCI bridge: Intel Corp. 82801BA/CA/DB
  PCI Bridge (rev 05) (prog-if 00 [Normal decode])
  Flags: bus master, fast devsel, latency 0
  Bus: primary=00, secondary=02, subordinate=02, sec-latency=32
  I/O behind bridge: 0000a000-0000afff
  Memory behind bridge: e7000000-e8fffffff

00:1f.0 ISA bridge: Intel Corp. 82801BA ISA Bridge (LPC) (rev 05)
  Flags: bus master, medium devsel, latency 0

00:1f.1 IDE interface: Intel Corp. 82801BA
  IDE U100 (rev 05) (prog-if 80 [Master])
  Subsystem: Intel Corp.: Unknown device 2442
  Flags: bus master, medium devsel, latency 0
  I/O ports at f000 [size=16]

01:00.0 VGA compatible controller: ATI Technologies Inc
  Rage Mobility P/M AGP 2x (rev 64) (prog-if 00 [VGA])
  Subsystem: ATI Technologies Inc Rage Mobility P/M AGP 2x
  Flags: bus master, stepping, medium devsel, latency 32, IRQ 9
Memory at e4000000 (32-bit, non-prefetchable) [size=16M]
I/O ports at 9000 [size=256]
Memory at e6000000 (32-bit, non-prefetchable) [size=4K]
Expansion ROM at <unassigned> [disabled] [size=128K]
Capabilities: [50] AGP version 1.0
Capabilities: [5c] Power Management version 1

02:09.0 Network controller: Adlink Technology PCI-7841 (rev 02)
  Subsystem: Adlink Technology PCI-7841
  Flags: medium devsSel, IRQ 5
  I/O ports at a000 [size=128]
  I/O ports at a400 [size=256]
  I/O ports at a800 [size=256]

02:0a.0 Ethernet controller: 3Com Corporation 3c905C-TX/TX-M [Tornado] (rev 74)
  Subsystem: 3Com Corporation
  3C905C-TX Fast Etherlink for PC Management NIC
  Flags: bus master, medium devsSel, latency 32, IRQ 11
  I/O ports at ac00 [size=128]
  Memory at e8000000 (32-bit, non-prefetchable) [size=128]
  Expansion ROM at <unassigned> [disabled] [size=128K]
  Capabilities: [dc] Power Management version 2

[root@localhost root]#

This list reports that the PCI-7841 subsystem uses IRQ 5. This IRQ number appears nowhere else in the list, indicating this IRQ is not shared with any other devices. Thus, the reported installation is correct: there is no IRQ conflict.

In case the IRQ number appears twice or more in the lspci report, you should find and affect a free IRQ to the CAN board. To do this, you have two solutions:

- your PC is equipped with a BIOS allowing to configure PCI slots, then you can affect the IRQ number you desire (in our case an unused one).
- your PC does not have such a BIOS, only remains the possibility to move the CAN board to another PCI slot, until lspci reports no IRQ conflict.

NOTE FOR DEVELOPERS: Do not forget to specify the CAN speed you want to use for your application. For instance, if you intend to use a 800 Kbps CAN baudrate add the following line to your application dependent ‘.m4x’ file:

```c
define(‘CAN_speed’,800Kbps)
```

### 3.1.3 Installing the CAN Dongle

Configuring your PC for the CAN Dongle support is easier. This device uses the parallel port for driver the SJA1000 chipset. Hence, just be sure your parallel port is configure in ECP or EPP mode (to be done from the PC BIOS) and verify that, at boot time, the Linux kernel returns you the following display:

```bash
... parport0: PC-style at 0x378 (0x778) [PCSPP,TRISTATE] parport0: irq 7 detected ...
```

If your display is too fast, you can also check it after logging in by using dmesg from the command line:
[root@localhost root]# dmesg
...
EXT3 FS 2.4-0.9.19, 19 August 2002 on ide0(3,6), internal journal
EXT3-fs: mounted filesystem with ordered data mode.
SCSI subsystem driver Revision: 1.00
hdc: attached ide-scsi driver.
scsi0 : SCSI host adapter emulation for IDE ATAPI devices
   Vendor: MATSHITA Model: UJDA730 DVD/CDRW Rev: 1.02
   Type: CD-ROM ANSI SCSI revision: 02
parport0: PC-style at 0x378 (0x778) [PCSPP,TRISTATE]
parport0: irq 7 detected
Attached scsi CD-ROM sr0 at scsi0, channel 0, id 0, lun 0
sr0: scsi3-mmc drive: 24x/24x writer cd/rw xa/form2 cdda tray
Uniform CD-ROM driver Revision: 3.12
...

If you do not get the right information in this display and you are sure your
PC is equipped with a parallel port (remember that a DB25 connector could also
be attached to a serial line), you will have to configure it manually. You have
two solutions: doing it using directly kernel modules or using kmod. Please for
more advanced use refer to ‘/usr/src/linux/Documentation/parport.txt’ and
’/usr/src/linux/Documentation/parport-lowlevel.txt’

3.1.3.1 Using parport as modules

If you load the parport code as a module, say:

    # insmod parport

To load the generic parport code. You then must load the architecture-dependent code with
(for example):

    # insmod parport_pc io=0x3bc,0x378,0x278 irq=none,7,auto

To tell the parport code that you want three PC-style ports, one at 0x3bc with no IRQ,
one at 0x378 using IRQ 7, and one at 0x278 with an auto-detected IRQ. Currently, PC-style
(parport_pc), Sun ‘bpp’, Amiga, Atari, and MFC3 hardware is supported.

PCI parallel I/O card support comes from parport_pc. Base I/O addresses should not be
specified for supported PCI cards since they are automatically detected.

3.1.3.2 Using kmod

If you use kmod, you will find it useful to edit /etc/modules.conf. Here is an example of the
lines that need to be added:

    alias parport_lowlevel parport_pc
    options parport_pc io=0x378,0x278 irq=7,auto

KMod will then automatically load parport_pc (with the options "io=0x378,0x278
irq=7,auto") whenever a parallel port device driver (such as lp) is loaded.

Note that these are example lines only! Generally, you shouldn’t need to specify any options
to parport_pc in order to be able to use a parallel port.

NOTE FOR DEVELOPERS: by default, when using the CAN dongle, SynDEx kernel macros
assume that the dongle IRQ is 0x7 (which is the case on most computers). If it is not your case,
do not forget to add the convenient line to your application dependent ‘.m4’ file. For instance if your parallel port IRQ is 0x5 add this:

\[ \text{define('ECP_IRQ', '0x5')} \]

Moreover, as mentioned previously (See Section 3.1.2 [Installing the PCI 7841 board], page 6.) do not forget to specify the CAN speed you want to use for your application. For instance, if you intend to use a 800 Kbps CAN baudrate add the following line to your application dependent ‘.m4’ file:

\[ \text{define('CAN_speed', '800Kbps')} \]

### 3.2 Preparing Linux kernel for RTAI installation

First of all, for these operations, log as root. Get Linux kernel sources from the ‘required’ directory and move it under the ‘/usr/src’ directory. Get also RTAI sources from the ‘required’ directory and move it under your development tree (e.g. ‘syndex/[...]/last’).

\[
\begin{align*}
\text{cd &} & \text{required} \\
\text{cp &} & \text{linux/linux-2.4.xx.tar.bz2 /usr/src} \\
\text{cp &} & \text{rtai/rtai-24.1.xx.tgz syndex/[...]/last/rtai/}
\end{align*}
\]

Unpack RTAI sources:

\[
\begin{align*}
\text{cd &} & \text{syndex/[...]/last/rtai/} \\
\text{tar &} & \text{zxfv rtai-24.1.xx.tgz}
\end{align*}
\]

Unpack the Linux kernel sources:

\[
\begin{align*}
\text{cd &} & \text{/usr/src} \\
\text{tar &} & \text{--bzip2 -xvf linux-2.4.xx.tar.bz2} \\
\text{or} \\
\text{tar &} & \text{-xzvf linux-2.4.xx.tar.gz}
\end{align*}
\]

Apply the ADEOS patch for this kernel. The names of the patches are in the form ‘patch-2.4.xx-adeos-rx’, where ‘2.4.xx’ is the kernel version the patch applies to, and the final IDs are the interface and version number for the patch. Interface numbers are changed when there is an incompatible change made to the patch, and the version letter is changed whenever a bugfix is made.

Apply the patch. Taken that ‘$RTAIDIR’ is the directory where you have extracted your RTAI sources:

\[
\begin{align*}
\text{patch &} & \text{-p1 < $RTAIDIR/rtai-24.1.9/patches/patch-2.4.xx-adeos-rx}
\end{align*}
\]

Configure the kernel. For an ADEOS patched one enable ADEOS in the ”General setup” menu. You can have ADEOS either as a module or native within the Linux kernel. ADEOS mutex support can be enabled also but it does not affect RTAI. Under X11, command to this is:
Chapter 3: Installation

make xconfig

See the README in the kernel source, in the section "Configuring the kernel" for additional information.

One easy way to configure a kernel is to start from the configuration of a previous kernel. Copy the configuration file (typically found in `/boot`) to `.config`, and then run `make oldconfig`.

Be careful to:
- disable "Set version information on all module symbols" under the "Loadable module support" kernel configuration paragraph, to avoid missing links when you’ll use RTAI
- disable The kernel APM option (that is known to be suspicious)

Then, compile modules, the kernel and install:

make dep; make modules; make modules_install; make bzImage; make install

Let us note that bootloader (grub) is updated automatically. Now reboot the system with this new kernel.

3.3 Installing and testing RTAI kernel

Change directories to the RTAI source tree (e.g. `syndex/.../last/rtai/rtai-24.1.xx`), and configure RTAI using:

make config

or:

make menuconfig

This will ask you the location of the ADEOS/RTHAL-enabled kernel source that you just compiled, and then ask a bunch of configuration questions. The default answers are almost always the best selection for beginning users.

If you want to trust the configuration already found in the distribution and thus avoid answering any configuration question, simply type:

make oldconfig

Clearly the above command can be used any time you want to configure RTAI using any configuration already available, i.e. when you already have an RTAI `.config` file you like.

Compile RTAI using:

make dep; make

Install RTAI using (as root):

make install

make dev

This will install the compiled RTAI modules into the `/lib/modules` directory, so that they can be loaded using `modprobe`.

For administrators intending to make the robotics systems available for non-root users, it is possible to tune `sudo`. `sudo` is used for executing a command as another user. Typically, we
will use ‘sudo’ to allow a permitted user to execute a command as the superuser, as specified in the sudoers file. For allowing RTAI use you just need to set access to ‘sh’, ‘insmod’, ‘rmmod’ and ‘lsmod’. This configuration is done by editing ‘/etc/sudoers’ file. For this, use ‘visudo’ as root from the command line. Here is a copy of a sample ‘/etc/sudoers’ configuration that tunes ‘sh’, ‘insmod’, ‘rmmod’ and ‘lsmod’ access for user ‘guest’:

# sudoers file.
#
# This file MUST be edited with the ‘visudo’ command as root.
#
# See the sudoers man page for the details on how to write a sudoers file.
#
# Host alias specification
# User alias specification
# Cmd alias specification
# Defaults specification

# User privilege specification
root ALL=(ALL) ALL
guest ALL=/bin/sh,/sbin/insmod,/sbin/rmmod,/sbin/lsmod

# Uncomment to allow people in group wheel to run all commands
# %wheel ALL=(ALL) ALL

# Same thing without a password
# %wheel ALL=(ALL) NOPASSWD: ALL

# Samples
# %users ALL=/sbin/mount /cdrom,/sbin/umount /cdrom
# %users localhost=/sbin/shutdown -h now

Once configured, you can test your ‘sudo’ config. First, logon as user ‘guest’. For testing ‘lsmod’ command execution right, type:

[guest@localhost guest]$ sudo /sbin/lsmod

Now, ‘sudo’ will ask for a password. So, if user ‘guest’ password is GuEsT1234, give ‘sudo’ GuEsT1234 as a password. Then, you will get the displays returned by /sbin/lsmod command:

[guest@localhost guest]$ sudo /sbin/lsmod
Password:

Module      Size Used by Not tainted
trident   34228  0  (autoclean)
ac97_codec 16776  0  (autoclean)  [trident]
pcigame    2824  0  (autoclean)  [trident]
gameport  2980  0  (autoclean)  [pcigame]
soundcore  5860  2  (autoclean)  [trident]
radeon    88824 1
agpgart  47808 3
appletalk 27044 12  (autoclean)
binfmt_misc 7464 1
parport_pc 19012 1 (autoclean)
lp 8996 0 (autoclean)
parport 34528 1 (autoclean) [parport_pc lp]
8139too 18472 1
mii 3700 0 [8139too]
crc32 3592 0 [8139too]
sg 36844 0 (autoclean)
sr_mod 18168 0 (autoclean)
ide-scsi 12240 0
scsi_mod 106644 3 [sg sr_mod ide-scsi]
ide-cd 35776 0
cdrom 32960 0 [sr_mod ide-cd]
nls_iso8859-1 3516 1 (autoclean)
nls_cp437 5148 1 (autoclean)
vfat 12812 1 (autoclean)
fat 37752 0 (autoclean) [vfat]
keybdev 2944 0 (unused)
mousedev 5556 1
hid 24676 0 (unused)
input 5312 0 [keybdev mousedev hid]
usb-ohci 21768 0 (unused)
usbcore 76288 1 [hid usb-ohci]
ext3 71012 3
jbd 49856 3 [ext3]

[guest@localhost guest]$ 

If you prefer user ‘guest’ to access these commands without confirming his password, you can just add a NOPASSWD flag in the related ‘sudoers’ line:

guest ALL=NOPASSWD:/bin/sh,/sbin/insmod,/sbin/rmmod,/sbin/lsmod

For further information about ‘sudo’ configuration, please refer to ‘sudoers’, ‘sudo’ and ‘visudo’ man pages.

3.4 Installing the MPC555 cross compiler

If you need to re-install the cross compiler we gave you, please copy back the content of the ‘crossgcc’ directory into your development tree. If for any reason, you need to recompile the cross compiler, please follow directives given below.

First, delete the content of the ‘crossgcc’ directory located under your development tree:

rm -rf crossgcc/*

Now, copy back all compiler related files and extract them into this ‘crossgcc’ directory:

... 
cd crosgcc
cp ..//../required/crossgcc/* . /
tar jxvf binutils-x.x.tar.bz2
tar jxvf gcc-core-x.x.tar.bz2
tar zxvf newlib-x.x.x.tar.gz

Then, create build directories under the ‘crossgcc’ directory:
cd crossgcc
mkdir build-binutils build-gcc build-newlib

Now compile binutils:

cd build-binutils
CC=gcc
../binutils-x.xx/configuration --target=ppc-elf32
    --prefix=/home/pierre/[...]last/crossgcc/
    --with-newlib
make all install

Add binutils binary location to your PATH:

PATH=$PATH:/home/pierre/synx/devel/.../last/crossgcc/bin/

... more details on ‘binutils2.14/binutils/README’ file.

Now compile GCC:

cd ../build-gcc
../gcc-x.x.x/configuration --target=ppc-elf32
    --prefix=/home/pierre/[...]last/crossgcc/
    --with-newlib
make all install

... more details under ‘gcc-gcc-3.3.2/INSTALL’ directory.

Now compile newlib:

cd ../build-newlib
../newlib-x.xx.x/configuration --target=ppc-elf32
    --prefix=/home/pierre/[...]last/crossgcc/
make all install

Precious information about building cross compiler could be found at the following addresses:
• http://www.stoward.com/CrossGCC/
• http://www.kegel.com/crosstool/
• http://billgatif.com/twiki/bin/view/Crossgcc/WebHome.
• http://ecos.sourceforge.org/build-toolchain.html

3.5 Installing the C math library support

The math library support we use rely on fdlibm version 5.2, developed at SunSoft (a Sun Microsystems, Inc. business.).

FDLIBM is intended to provide a reasonably portable (see assumptions below), reference quality (below one ulp for major functions like sin,cos,exp,log) math library (libm.a). For a copy of FDLIBM, please see: http://www.netlib.org/fdlibm/ or http://www.validlab.com/software/

If you need to re-install the C math library support we gave you, please copy back the content of the ‘crossgcc’ directory into your development tree. If for any reason, you need to recompile the C math library support, please follow directives given below.

First, copy back the C math library sources archive under the ‘crossgcc’ directory, located under your development tree, and unpack it:
... 
cd crossgcc

cp .././/required/crossgcc/fdlibm-x.x.tar.bz2 .
tar jxvf fdlibm-x.x.tar.bz2

Change to ‘fdlibm’ directory and compile sources:

...

cd fdlibm
make "CFLAGS = -D_IEEE_LIBM"

Now, clean your C math library directory:

rm -f *.c *.o makefile index*

3.6 Installing SynDEx CAD

SynDEx is a system level CAD software based on the "algorithm-architecture adequation" (AAA) methodology, for rapid prototyping and optimizing the implementation of distributed real-time embedded applications onto multicomponent architectures. It has been designed and developed at INRIA in the Rocquencourt Research Unit France, by the OSTRE team. The SynDEx distribution on which is based our development toolchain is version 5.2. If you need to re-install the SynDEx CAD software, please copy the archive located under ‘required/syndex’ directory. Place the copy where are your other home applications (e.g. under ‘~/bin’) and unpack it:

...

cd required/syndex

cp syndex-x.x.x-linux.tgz ~/bin
cd ~/bin
tar zxvf syndex-x.x.x-linux.tgz

This will create in the current directory the SynDEx subdirectory containing all the distributed files. Read the ‘README’ file to complete the installation, and to start SynDEx.

If you experience any installation problem (and the answer is not in the ‘README’ file or in the FAQ), please send by e-mail, to SynDEx staff, a description of the problem, with enough informations to let them a chance to resolve it.

3.7 Development tree organisation

Once all software installations, mentioned above, are completed, only remains to build (or recover) a clean development tree. It is very IMPORTANT to note that the various toolchains utils, Makefiles and scripts rely on a tree dependent organisation: thus we highly encourage to respect the tree shape we describe here.

/home/pierre/syndex/current/last
-- appsv5
-- appsv6
-- crossgcc
-- doc
-- ldscripts
-- macrosv5
-- macrosv6
-- macrosv6-20041220.tgz
-- mini-crossgcc
-- rtai
'-- utils

10 directories, 1 file

First, let us define all directories:

- `apps5v5`: contains old applications developed with SynDEx V5 distributions. This directory should only be used for maintaining old applications, but not to create new ones, as SynDEx V5 is no more supported by OSTRE team.
- `apps6v6`: contains applications developed with the newly supported SynDEx V6 distribution.
- `crossgcc`: contains the cross MPC555 compiler tools (refer to previous sections for details).
- `doc`: contains all documents related to the development environment (datasheets, specifications, user guides, etc...). It also contains the documentation you are reading now. Please, leave the `doc` directory arranged this way if you want to keep benefits of dynamical links that exist between them.
- `ldscripts`: contains the linker scripts used by the cross compiler when producing MPC555 binaries. This files includes comments that may give you deep information on the MPC555 boards memory mapping and binary sections.
- `macros5v5`: contains the last SynDEx macros developed for V5 distributions
- `macros6v6`: contains the last SynDEx macros developed for V6 distributions
- `rtai`: contains the RTAI distribution (or at least a symbolic link pointing toward the directory where it is installed). Please, refer to previous sections for details.
- `utils`: contains a set of tools, developed by Robosoft, required for generating binaries, generating SynDEx bootloaders for MPC555, downloading applications and also preparing applications for being transfered into MPC555 internal flash memory.

As mentioned above, `macros5v5` and `macros6v6` directories contain the last macros developed for respective versions of SynDEx distributions. This mean that SynDEx macros, either for V5 or V6, provide programmers with exactly the same functionalities. Main SynDEx macros features, common to all supported processors, are documented over the dedicated SynDEx documentation web site.

Apart from this, a huge set of advanced macros has been developed for handling all MPC555 features. These macros are fully documented into The 555.m4 and RSB.m4x SynDEx Macro-Executives [See `doc/ROBO/555macros/555macros.pdf`] document.

Another macro has been developed for PCs running RTAI/Linux. It aims to provide Linux users with a C/C++ programming interface, for accessing hardware resources handled by SynDEx. This macro, named `linuxIO`, is documented into The SynDEx “linuxIO_” Macro: An Easy C/C++ Linux User’s Application Interface [See `doc/ROBO/linuxIO_/linuxIO_.pdf`].
4 Setting toolchain variables

Right after installing all the Robosoft development toolchain, remains few setting to be done. After this step, the toolchain will be fully functional and ready to be used for developing applications.

4.1 Bash environment variables

First, we recommend user to add the few lines into your ‘~/.bashrc’ file. Let assume you installed SynDEx CAD under your ‘~/bin’ directory. If you are using SynDEx CAD version V5 insert this:

\[
\text{alias \; syindex52='$HOME/bin/syindex52/syindex5'}
\]
\[
\text{SYINDEX5=$HOME/bin/syindex52}
\]
\[
\text{export SYINDEX5}
\]

If you are using SynDEx CAD version V6 just insert this:

\[
\text{alias \; syindex-6.6.8='$HOME/bin/syindex-6.6.8/syindex-6.6.8'}
\]

If you intend to both version at the same time, insert all these lines.

4.2 Application makefile rules

Each developed application, should include under its directory and application dependent makefile named ‘GNUmakefile’. Let us focus on the ‘robucar’ application.

\[
[guest@localhost robucarAna30-0.9-v6.6.7]$ \\
[guest@localhost robucarAna30-0.9-v6.6.7]$ pwd \\
/home/guest/syindex/last/appsV6/robucarAna30-0.9-v6.6.7 \\
[guest@localhost robucarAna30-0.9-v6.6.7]$ \\
[guest@localhost robucarAna30-0.9-v6.6.7]$ 11 \\
\text{total 56} \\
-rw-rw-r-- 1 guest guest 3860 Jan 26 17:30 f555.m4 \\
-rw-------- 1 guest guest 2523 Feb 20 14:28 GNUmakefile \\
-rw-rw-r-- 1 guest guest 7739 Jan 26 17:30 r555.m4 \\
-rw-rw-r-- 1 guest guest 290 Jan 26 17:30 robucar.m4 \\
-rw-------- 1 guest guest 11333 Apr 6 18:00 robucar.m4x \\
-rw-------- 1 guest guest 14362 Jan 26 17:03 robucar.sdx \\
-rw-rw-r-- 1 guest guest 2008 Jan 26 17:30 root.m4 \\
-rw-r--r-- 1 guest guest 78 Feb 5 18:00 User.mk \\
[guest@localhost robucarAna30-0.9-v6.6.7]$ \\
\]

‘GNUmakefile’ contains a set of rules, required for compiling application correctly, and also, a set of variables related to your toolchain installation. Let us comment the ‘GNUmakefile’ content step by step.

4.2.1 Linux paths

\[
\text{# $(LINUX_SRC) is the path to the Linux home directory} \\
\text{# $(LINUX_LIB) is the path to the Linux library files} \\
\text{LINUX_HOME = /usr/src/linux-2.4.22} \\
\text{LINUX_LIB = /usr/lib}
\]

Here you should indicate where to find the Linux sources and libraries (corresponding to the currently running kernel).
4.2.2 RTAI paths

# $(RTAI_HOME) is the path to the RTAI home directory
# If you change this path, don’t forget to change the one in the "run" script!
RTAI_HOME = ../../rtai/rtai-24.1.12

Here you should indicate the path of RTAI installation. Note that you can either locate this directory with an absolute path ('RTAI_HOME = /home/guest/syn dex/last/rtai/rtai-24.1.12') or, as set above, a relative path ('RTAI_HOME = ../../rtai/rtai-24.1.12').

4.2.3 Toolchain macros, compiler and utils paths

export Macros_Path = ../../macrosv6
export M4PATH = $(Macros_Path)
export LDCRIPTPATH = ../../ldscripts
export CROSSGCCBIN = ../../crossgcc/bin
export CROSSGCCLIB = ../../crossgcc/ppc-elf32/lib
export CROSSGCCINC = ../../crossgcc/ppc-elf32/include
export FDLIBM = ../../crossgcc/fdlibm
export UTILSPATH = ../../utils
M4=m4
STARTPRG = $(UTILSPATH)/bin2rtfifo/bin2rtfifo

Here are comments about these variables:

- **Macros_Path**: indicate the path where SynDEx macros may be found.
- **M4PATH**: here nothing to be done. This variable exists for backward compatibility with SynDEx V5 distributions.
- **LDCRIPTPATH**: indicate the path where linker scripts are located
- **CROSSGCCBIN**: indicate the path where to find cross-compiler tools
- **CROSSGCCLIB**: indicate the path where to find cross-compiler libraries
- **CROSSGCCINC**: indicate the path where to find cross-compiler header files
- **FDLIBM**: indicate the path where to find the mathematical library
- **UTILSPATH**: indicate the path where to find the Robosoft development toolchain utilities
- **M4**: indicate which macroprocessor is installed on your development workstation: here it is m4, but it could be gn4 depending on your Linux installation.
- **STARTPRG**: indicate which utility is used for passing application executables to the download sequence (at boot time, i.e. when starting application)

4.2.4 Application name variable

A = robucar

Here user just need to mention the application name. In this particular case, the application graphs have been stored under ‘robucar.sdx’, thus the corresponding application name is robucar.

4.2.5 Application specific dependencies

# application specific dependents:

P0.libs = myLib.555
This type of rule aim to give operators access to separately compiled sets of functions (also called libraries). For each operator (that belong to the main architecture description), exists a `GNUmakefile` variable which name is the operator name followed by `.libs` extension.

In this example, we indicate that object file `myLib.555` contains functions that may be called by operator P0. Such function calls are usually done using SynDEx macro `Ccall_`. For advanced information about `Ccall_` SynDEx macro, please refer to SynDEx online documentations and to the Robosoft document titled "The 555.m4x and RSB.m4x SynDEx Macro-Executives [See 'doc/ROBO/555macros/555macros.pdf']".

Note that an example of use is given farther to illustrate mathematical function calls. See Section 5.3 [About FDLIBM the C math library support], page 25.

### 4.2.6 Additional makefile rules

If for any reason (external function calls, separate compilation, etc...) user need to add his own extra makefile rules, he should include them into the local `User.mk` file.

By default, this file contains a rule needed for preparing standalone flash images (named `flash_image`): this rule should NEVER be modified nor deleted! ! !... Thus additional rules should be added at the end of the file.

```
[guest@localhost robucarAna30-0.9~v6.6.7]$ cat User.mk
flash_image: $(($(A).root)
    make $(A).mk
    $(UTILSPATH)/sdx2img/sdx2img $? > @

myLib.555 : myLib.c

[guest@localhost host robucarAna30-0.9~v6.6.7]$
```

For instance, here the added rule allow to compile `myLib.c` into an object file. This object file `myLib.555` contains functions used as processor P0 library in the subsection above (ie. `P0.libs = myLib.555`).
5 Developping applications

5.1 Reminder: the SynDEx compilation sequence

The figure above depicts the full compilation sequence following the SynDEx executives generation. You can see here the way GNU development utilities (macroprocessor, compiler, ...) are run for producing binaries. In our particular case, this sequence produce PowerPC ELF 32-bits binaries for MPC555 operators and kernel modules for PC operators (running RTI/Linux).

The Robosoft development toolchain utilities, we describe below, make use of the generated PowerPC ELF 32-bits binaries, for achieving various type of work.

5.2 About Robosoft development toolchain utilities

The Robosoft development toolchain contains a set of utilities, required for downloading applications, manipulating PowerPC ELF 32-bits binaries and preparing images for flashed applications. Here is the directory tree you can find under ‘.../last/utils’:

```
/home/pierre/syndex/current/last/utils
-- app2flash
-- bin2rtfifo
```
-- bootapp
-- bootloader
-- dwnbin
-- elf2sdxbin
'-- sdx2img

7 directories, 0 files
Let us browse and describe utilities features.

5.2.1 SynDEx download tools: the ‘elf2sdxbin’ and ‘bin2rtfifo’ utilities

IMPORTANT!... These tools are fully linked to the SynDEx download procedure: thus, they are not intended to be run directly by users. The following explanations just aim to familiarize users with SynDEx environment.

We recommend user to read about SynDEx download procedure. The following documents cover this topic: "SynDEx v5 Downloader Specification" [See 'doc/SDX/Dwnspec.pdf'] and "CAN Boot-Loader and Downloader Specification" [See 'doc/SDX/CANboot.pdf'] that is more specifically focused on the way a PC (running RTAI and Linux) should boot a set of MPC555 operators.

The ‘elf2sdxbin’ and ‘bin2rtfifo’ utilities are involved in download process right after the MPC555 executives compilation. At that step, one PowerPC ELF (32-bits) binary exists for each MPC555 operator of the main architecture. Refering the SynDEx download specifications (mentionned above), these ELF binaries should be converted into a SynDEx-download compatible format; this first step is done by ‘elf2sdxbin’.

Once converted, the SynDEx-download compatible binaries are forwarded to the main PC operator in order to be passed to the main download sequence. To this aim, the binary files (locally stored on the PC hard drive) are read and written sequentially to an RTAI realtime fifo. The second end of this fifo is handled by the main operator communication thread, in order to pass the received binaries to the loadDnto_sequence. This forwarding operation is done by ‘bin2rtfifo’.

5.2.2 Another download tool: the ‘dwnbin’ utility

For users intending to download executives into MPC555 boards without using the SynDEx automatic download sequence, we developed the ‘dwnbin’ utility.

This utility should be run by user logged as root (or with root privileges). It is executed from the shell command line with the following syntax:

```
./dwnbinecp BAUDRATE FILE0 SID0 FILE1 SID1 ... FILEn SIDn
```

where:
- **BAUDRATE** is used for passing CAN baudrate to be used for download; possible values are 800, 500 or 250 respectively for 800 Kbps, 500 Kbps or 250 Kbps.
- **FILE** is the name of a SynDEx binary file user want to download: WARNING! DO NOT FORGET that such a file should result from an elf2sdxbin conversion.
- **SID** is the identifier of the station where code is supposed to be downloaded.

Here is an example of use: here, we download the binary file ‘P0’ into the MPC555 board of Id. 0x4000 and the binary file ‘P1’ into the MPC555 board of Id. 0x4001. CAN baudrate during download operation is set to 250 Kbps.

```
[root@localhost dwnbin]# ./dwnbinecp 250 P0 0x4000 P1 0x4001
```

SynDEx binary file name: P0
Target station ID: 0x4000... loaded.
SynDEx binary file name: P1
Target station ID: 0x4001... loaded.
[root@localhost dwnbin]#

If the user forget an argument, he will get the following display:

[root@localhost dwnbin]# ./dwnbinecp P0 0x4000 P1 0x4001
ERROR: bad arguments number

USAGE: ./dwnbinecp BAUDRATE FILE0 SID0 FILE1 SID1 ... FILEn SIDn

BAUDRATE possible values are 800, 500 or 250 resp. for 800 Kbps, 500 Kbps or 250 Kbps CAN baudrate
FILE is expected to be a SynDEx downloader format file
SID is the identifier of the station where code is supposed to be loaded.

[root@localhost dwnbin]#

If the user specify an invalid baudrate, he will get the following display:

[root@localhost dwnbin]# ./dwnbinecp 150 P0 0x4000 P1 0x4001
ERROR: bad baudrate

USAGE: pass 800, 500 or 250 resp. for 800 Kbps, 500 Kbps or 250 Kbps settings! ! !... 

[root@localhost dwnbin]#

If ‘dwnbinecp’ does not manage to download the boards (because of a broken CAN cable or a missing power supply), it will return the following display:

[root@localhost dwnbin]# ./dwnbinecp 250 P0 0x4000 P1 0x4001
SynDEx binary file name: P0
Target station ID: 0x4000
WARNING! ! ! Time out has expired...

SynDEx binary file name: P1
Target station ID: 0x4001
WARNING! ! ! Time out has expired...
[root@localhost dwnbin]#

IMPORTANT REMARK: each time the download sequence fails, do not forget to reset the MPC555 boards by turning the off, waiting few seconds and turning them on back again. This way, you will be sure bootloaders are in a correct execution state.

‘dwnbinecp’ implement the download sequence in relation with the SynDEx download procedure. We recommend advanced user to read about SynDEx download procedure. The following documents cover this topic: "SynDEx v5 Downloader Specification" [See ‘doc/SDX/Dwnspec.pdf’] and "CAN Boot-Loader and Downloader Specification" [See ‘doc/SDX/CANboot.pdf’]. Implementation comments could also be found in the ‘dwnbinecp’ source code (located under the ‘dwnbin’ directory). We advise users that Robosoft is not responsible for the consequences of any change into the source code.
5.2.3 Bootloader tools: the ‘bootloader’ utility

Bootloader tool aim to prepare binary images to be flashed into MPC555, in order to make the MPC555 boards bootable through the CAN bus interface. These binaries consist of very small codes implementing the minimal board initialisations to make the SynDEx download sequence be possible.

What characterise a bootloader are:
- its identifier (a 32-bits integer value chosen by the application developer),
- and its CAN baudrate (indicating the CAN communications speed during the SynDEx download sequence)

Preparing a bootloader is rather simple. First, move to the ‘bootloader’ utility directory.

[guest@Gromit bootloader]$ pwd
/home/guest/syndex/last/utils/bo tloader
[guest@localhost bootloader]$ ll
total 40
-rw-r--r-- 1 guest guest 4242 Feb 20 16:56 bootloader-1.0.ld
-rw-r--r-- 1 guest guest 24241 Feb 5 09:47 bootloader-1.0.s
-rw-r--r-- 1 guest guest 1218 Feb 20 17:01 Makefile
-rw-r--r-- 1 guest guest 169 Feb 5 17:32 README
[guest@localhost bootloader]$ 

This directory contains the bootloader assembly sources (with .s extension), the related linker script (with .ld extension), a ‘README’ file (showing basic usage) and the ‘Makefile’. This ‘Makefile’ is used for passing bootloader appropriate parameters (identifier and CAN baudrate). Let us show how to use it over an example. Here is the command line a user should execute for generating a bootloader which identifier is 0x4000 and CAN baudrate is 800 Kbps:

make PROCID=0x4000 KBPS=800

Where:
- PROCID corresponds to the identifier 32-bits HEXADECIMAL integer value
- KBPS corresponds to bootloader CAN baudrate

IMPORTANT REMARK: Available CAN baudrates are 800 Kbps, 500 Kbps and 250Kbps. Passing other values will not returns error messages but will generate erroneous flash images.

On success, the following message will be displayed:

# -----------------------------------------------
# # Output file is: bootloader-1.0-0x4000-800.s19
# # -----------------------------------------------

It indicates user that the bootloader flash image has been written into an S19 file, named ‘bootloader-1.0-0x4000-800.s19’.

[guest@localhost bootloader]$ ll
total 68
-rwxrwxr-x 1 guest guest 25894 Apr 8 10:05 bootloader-1.0-0x4000-800.s19
-rw-r--r-- 1 guest guest 4242 Feb 20 16:56 bootloader-1.0.1d
-rw-r--r-- 1 guest guest 24241 Feb 5 09:47 bootloader-1.0.s
-rw-r--r-- 1 guest guest 1218 Feb 20 17:01 Makefile
-rw-r--r-- 1 guest guest 169 Feb 5 17:32 README
[guest@localhost bootloader]$
Now, only remains to program the MPC555 target board.

The S19 file, obtained above, is ready to be transferred to the flash memory of the main MPC555 microcontroller. Commercial tool allow to do this: for instance Raven or Blackbird BDM devices from Macraigor Systems. More information about these products could be found at the Macraigor Systems website.

At that step, programming MPC555 microcontroller flash memory is very hardware dependent. So please refer to the documentation of the BDM device you chose and follow instructions.

### 5.2.4 Flash tools: the ‘app2flash’, ‘bootapp’ and ‘sdx2img’ utilities

Flash tools aim to prepare binary images to be flashed into MPC555, for standalone applications (ie. running without PC). Generated images respect the convenient file format, in order to be compatible with commercial flash programmer.

Preparing such a standalone application (for MPC555 only) falls into the following steps:

#### 5.2.4.1 To be done from SynDEx CAD

When designing your application with SynDEx CAD, you will have to specify the hardware architecture. For standalone application, the hardware architecture will be composed of one MPC555, or a set of MPC555 boards communicating over a CAN bus. The board (if alone), or one of the boards has to be declared as the main operator (the "root" operator in version V5 of SynDEx CAD). This means that, at boot time, this main operator will be used for downloading the other boards of the architecture. At that step, nothing else is required.

#### 5.2.4.2 The standalone application basic principle

In a standalone application, the download sequence is handled by the main MPC555 board. To this aim, the main operator flash memory contains both executives to be downloaded and operators identifiers. At boot time, main operator reads executives from a dedicated section of its flash memory and sequentially download all operators of the architecture. Once all operator are loaded, main operator sends a null synchronization frame on the CAN bus for starting application execution.

IMPORTANT REMARK: in such standalone architecture, only the main MPC555 board embeds an application dependent flash memory image. All other operators contain classical SynDEx bootloaders. This property is quite useful, as for changing an application, you only need to change the main operator flash memory image (instead of reprogramming all operators flash memories).

#### 5.2.4.3 Generating the application image

Let us show how to generate the main operator flash image over an example. Let us assume that a user guest is developing a standalone application named "robucar" with two MPC555 boards.

```
[guest@localhost robucarAna30-0.9-v6.6.7-flash]$ pwd
/home/guest/syndex/last/appsv6/robucarAna30-0.9-v6.6.7-flash
[guest@localhost robucarAna30-0.9-v6.6.7-flash]$ ll
total 52
-rw-rw-r--  1 guest  guest  3747 Feb  5 17:42 f555.m4
-rw--------  1 guest  guest  2506 Feb 20 14:29 GNUmakefile
-rw-rw-r--  1 guest  guest  6888 Feb  5 17:42 r555.m4
-rw-rw-r--  1 guest  guest  236 Feb  5 17:42 robucar.m4
-rw--------  1 guest  guest 11623 Apr  6 18:02 robucar.m4x
-rw--------  1 guest  guest 13522 Jan 30 10:05 robucar.sdx
-rw-r--r--  1 guest  guest    78 Feb  5 18:00 User.mk
[guest@localhost robucarAna30-0.9-v6.6.7-flash]$
```
This application is designed with two operators named respectively "r555" and "f555"; with "r555" as the main operator. Moreover, let us assume that user affects 0x4000 as an identifier for "r555" and 0x4001 for "f555". Let us also consider that the chosen CAN baudrate for this application is 250 Kbps. Before generating the "r555" flash image, user has to check if the 'User.mk' exists in the local application directory: this file contains a part of makefile required for the generation.

```
[guest@localhost robucarAna30-0.9-v6.6.7-flash]$ cat User.mk
flash_image: $(A).root
    make $(A).mk
    $(UTILSPATH)/sdx2img/sdx2img $? > $0
[guest@localhost robucarAna30-0.9-v6.6.7-flash]$
```

Now, everything is ready. Let us move to the 'app2flash' utility directory.

```
[guest@localhost robucarAna30-0.9-v6.6.7-flash]$ cd ../../utils/app2flash/
[guest@localhost app2flash]$ ll
-rw-r--r--  1 guest guest  936 Feb 13 11:04 Makefile
-rw-r--r--  1 guest guest  286 Feb  5 17:53 README
[guest@localhost app2flash]$
```

All the flash image generation procedure is launched by the local 'Makefile'. The 'README' file contains instructions on how to use it. For compiling a bootable version of the user application located under 'robucarAna30-0.9-v6.6.7-flash' directory, user just have to type the following command line:

```
make APPSPATH=../../appsv6 APP=robucarAna30-0.9-v6.6.7-flash ROOTID=0x4000 KBPS=250
```

where:

- **APPSPATH** argument corresponds to the directory where applications are located: in our case, the 'appsv6' directory
- **APP** argument corresponds to the name of the application directory: in our case, the 'robucarAna30-0.9-v6.6.7-flash' directory
- **ROOTID** argument corresponds to the main operator identifier: in our case, 0x4000 (as previously mentionned). The main operator identifier should be passed as an HEXADECIMAL 32-bits integer.
- and **KBPS** argument corresponds to the chosen CAN baudrate: in our case, 250 Kbps (as previously mentionned).

**IMPORTANT REMARK**: Available CAN baudrates are 800 Kbps, 500 Kbps and 250Kbps. Passing other values will not returns error messages but will generate erroneous flash images.

Once run, the Makefile will call other utilities: 'bootapp' and 'sdx2img'. These utilities should not be used alone, they are only expected to be called by the 'app2flash' Makefile. On success, the following message will be displayed:

```
# ________________________________________________________________
#
# Output file is: robucarAna30-0.9-v6.6.7-flash.s19
#
# ________________________________________________________________
```

It indicates user that the main operator flash image has been written into an S19 file, named 'robucarAna30-0.9-v6.6.7-flash.s19'.

5.2.4.4 Flashing the application image

The S19 file, obtained above, is ready to be transferred to the flash memory of the main MPC555 microcontroller. Commercial tool allow to do this: for instance Raven or Blackbird BDM devices from Macraigor Systems. More information about these products could be found at the Macraigor Systems website.

At that step, programming MPC555 microcontroller flash memory is very hardware dependent. So please refer to the documentation of the BDM device you chose and follow instructions.

5.3 About FDLIBM the C math library support

The math library support we use rely on fdlibm version 5.2, developed at SunSoft (a Sun Microsystems, Inc. business.).

FDLIBM (Freely Distributable LIBM) is a C math library for machines that support IEEE 754 floating-point arithmetic. In this release, only double precision is supported. It provides user with ANSI POSIX functions, which documentations you can find below.

At the end of this section, an example of use is given to illustrate how to call mathematical functions from a user defined SynDex macro.

5.3.1 double acosh (double);

NAME
acosh - inverse hyperbolic cosine function

SYNOPSIS
#include <math.h>

double acosh(double x);

DESCRIPTION
The acosh() function calculates the inverse hyperbolic cosine of x; that is the value whose hyperbolic cosine is x. If x is less than 1.0, acosh() returns not-a-number (NaN) and errno is set.

ERRORS
EDOM x is out of range.

CONFORMING TO
SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO
asinh(3), atanh(3), cosh(3), sinh(3), tanh(3)
5.3.2 double acos (double);

**NAME**
acos - arc cosine function

**SYNOPSIS**
```
#include <math.h>

double acos(double x);
```

**DESCRIPTION**
The `acos()` function calculates the arc cosine of x; that is the value whose cosine is x. If x falls outside the range -1 to 1, `acos()` fails and `errno` is set.

**RETURN VALUE**
The `acos()` function returns the arc cosine in radians and the value is mathematically defined to be between 0 and PI (inclusive).

**ERRORS**
`EDOM` x is out of range.

**CONFORMING TO**
SVID 3, POSIX, BSD 4.3, ISO 9899

**SEE ALSO**
`asin(3), atan(3), atan2(3), cos(3), sin(3), tan(3)`

5.3.3 double asinh (double);

**NAME**
asinh - inverse hyperbolic sine function

**SYNOPSIS**
```
#include <math.h>

double asinh(double x);
```

**DESCRIPTION**
The `asinh()` function calculates the inverse hyperbolic sine of x; that is the value whose hyperbolic sine is x.
5.3.4 double asin (double);

NAME
asin - arc sine function

SYNOPSIS
#include <math.h>

double asin(double x);

DESCRIPTION
The asin() function calculates the arc sine of x; that is the value
whose sine is x. If x falls outside the range -1 to 1, asin() fails
and errno is set.

RETURN VALUE
The asin() function returns the arc sine in radians and the value is
mathematically defined to be between -PI/2 and PI/2 (inclusive).

ERRORS
EDOM  x is out of range.

CONFORMING TO
SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO
acosh(3), atanh(3), cosh(3), sinh(3), tanh(3)

5.3.5 double atan2 (double, double);

NAME
atan2 - arc tangent function of two variables

SYNOPSIS
#include <math.h>
double atan2(double y, double x);

**DESCRIPTION**
The `atan2()` function calculates the arc tangent of the two variables `x` and `y`. It is similar to calculating the arc tangent of `y / x`, except that the signs of both arguments are used to determine the quadrant of the result.

**RETURN VALUE**
The `atan2()` function returns the result in radians, which is between -PI and PI (inclusive).

**CONFORMING TO**
SVID 3, POSIX, BSD 4.3, ISO 9899

**SEE ALSO**
`acos(3)`, `asin(3)`, `atan(3)`, `cos(3)`, `sin(3)`, `tan(3)`

1993-06-08 ATAN2(3)

### 5.3.6 double atanh (double);

**ATANH(3)** Linux Programmer's Manual ATANH(3)

**NAME**
`atanh` - inverse hyperbolic tangent function

**SYNOPSIS**
```c
#include <math.h>

double atanh(double x);
```

**DESCRIPTION**
The `atanh()` function calculates the inverse hyperbolic tangent of `x`; that is the value whose hyperbolic tangent is `x`. If the absolute value of `x` is greater than 1.0, `acosh()` returns not-a-number (NaN) and `errno` is set.

**ERRORS**
`EDOM` `x` is out of range.

**CONFORMING TO**
SVID 3, POSIX, BSD 4.3, ISO 9899

**SEE ALSO**
`asinh(3)`, `acosh(3)`, `cosh(3)`, `sinh(3)`, `tanh(3)`

1993-06-13 ATANH(3)
5.3.7 double atan (double);

NAME
atan - arc tangent function

SYNOPSIS
#include <math.h>

double atan(double x);

DESCRIPTION
The atan() function calculates the arc tangent of x; that is the value
whose tangent is x.

RETURN VALUE
The atan() function returns the arc tangent in radians and the value is
mathematically defined to be between -PI/2 and PI/2 (inclusive).

CONFORMING TO
SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO
acos(3), asin(3), atan2(3), cos(3), sin(3), tan(3)

1993-06-08

5.3.8 double cbrt (double);

NAME
cbrt - cube root function

SYNOPSIS
#include <math.h>

double cbrt(double x);

DESCRIPTION
The cbrt() function returns the cube root of x. This function cannot
fail; every representable real value has a representable real cube
root.

CONFORMING TO
cbrt is a GNU extension.

SEE ALSO
sqrt(3), pow(3)
5.3.9 double ceil (double);

NAME
ceil, ceilf, ceill - ceiling function: smallest integral value not less than argument

SYNOPSIS
#include <math.h>

double ceil(double x);
float ceilf(float x);
long double ceill(long double x);

DESCRIPTION
These functions round x up to the nearest integer.

RETURN VALUE
The rounded integer value. If x is integral or infinite, x itself is returned.

ERRORS
No errors other than EDOM and ERANGE can occur. If x is NaN, then NaN is returned and errno may be set to EDOM.

NOTES
SUSv2 and POSIX 1003.1-2001 contain text about overflow (which might set errno to ERANGE, or raise an exception). In practice, the result cannot overflow on any current machine, so this error-handling stuff is just nonsense. (More precisely, overflow can happen only when the maximum value of the exponent is smaller than the number of mantissa bits. For the IEEE-754 standard 32-bit and 64-bit floating point numbers the maximum value of the exponent is 128 (resp. 1024), and the number of mantissa bits is 24 (resp. 53).)

CONFORMING TO
The ceil() function conforms to SVID 3, POSIX, BSD 4.3, ISO 9899. The other functions are from C99.

SEE ALSO
floor(3), lrint(3), nearbyint(3), rint(3), round(3), trunc(3)
NAME
cosh - hyperbolic cosine function

SYNOPSIS
#include <math.h>

double cosh(double x);

DESCRIPTION
The cosh() function returns the hyperbolic cosine of x, which is
defined mathematically as (exp(x) + exp(-x)) / 2.

CONFORMING TO
SVID 3, POSIX, BSD 4.3, ISO 9899 (C99)

SEE ALSO
acosh(3), asinh(3), atanh(3), sinh(3), tanh(3)

1993-06-13       COSH(3)

5.3.11 double cos (double);

COS(3)                Linux Programmer’s Manual              COS(3)

NAME
cos - cosine function

SYNOPSIS
#include <math.h>

double cos(double x);

DESCRIPTION
The cos() function returns the cosine of x, where x is given in radi-
ans.

RETURN VALUE
The cos() function returns a value between -1 and 1.

CONFORMING TO
SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO
acos(3), asin(3), atan(3), atan2(3), sin(3), tan(3)

1993-06-08       COS(3)

5.3.12 double erf (double);

ERF(3)                Linux Programmer’s Manual              ERF(3)
NAME
erf, erfc - error function and complementary error function

SYNOPSIS
#include <math.h>

double erf(double x);

double erfc(double x);

DESCRIPTION
The erf() function returns the error function of x; defined as
erf(x) = 2/sqrt(pi)* integral from 0 to x of exp(-t*t) dt

The erfc() function returns the complementary error function of x, that is 1.0 - erf(x).

CONFORMING TO
SVID 3, BSD 4.3

SEE ALSO
exp(3)

BSD 1993-06-25 ERF(3)

5.3.13 double erfc (double);

See Section 5.3.12 [erf], page 31.

5.3.14 double exp (double);

NAME
exp, log, log10, pow - exponential, logarithmic and power functions

SYNOPSIS
#include <math.h>

double exp(double x);

double log(double x);

double log10(double x);

double pow(double x, double y);

DESCRIPTION
The exp() function returns the value of e (the base of natural loga-
rithms) raised to the power of x.
The log() function returns the natural logarithm of x.
The log10() function returns the base-10 logarithm of x.
The pow() function returns the value of x raised to the power of y.

ERRORS
The log() and log10() functions can return the following errors:
EDOM The argument x is negative.
ERANGE The argument x is zero. The log of zero is not defined.

The pow() function can return the following error:
EDOM The argument x is negative and y is not an integral value. This would result in a complex number.

CONFORMING TO
SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO
sqrt(3), cbrt(3)

GNU 1993-06-16 EXP(3)

5.3.15 double fabs (double);

NAME
fabs, fabsf, fabsl - absolute value of floating-point number

SYNOPSIS
#include <math.h>

double fabs(double x);
float fabsf(float x);
long double fabsl(long double x);

DESCRIPTION
The fabs functions return the absolute value of the floating-point number x.

ERRORS
No errors can occur.

CONFORMING TO
The fabs() function conforms to SVID 3, POSIX, BSD 4.3, ISO 9899. The other functions are from C99.
5.3.16 int finite (double);

See Section 5.3.22 [isnan], page 37.

5.3.17 double floor (double);

NAME

floor, floorf, floorl - largest integral value not greater than argument

SYNOPSIS

#include <math.h>

double floor(double x);
float floorf(float x);
long double floorl(long double x);

DESCRIPTION

These functions round \( x \) down to the nearest integer.

RETURN VALUE

The rounded integer value. If \( x \) is integral or infinite, \( x \) itself is returned.

ERRORS

No errors other than EDOM and ERANGE can occur. If \( x \) is NaN, then NaN is returned and errno may be set to EDOM.

NOTES

SUSv2 and POSIX 1003.1-2001 contain text about overflow (which might set errno to ERANGE, or raise an exception). In practice, the result cannot overflow on any current machine, so this error-handling stuff is just nonsense. (More precisely, overflow can happen only when the maximum value of the exponent is smaller than the number of mantissa bits. For the IEEE-754 standard 32-bit and 64-bit floating point numbers the maximum value of the exponent is 128 (resp. 1024), and the number of mantissa bits is 24 (resp. 53).)

CONFORMING TO

The floor() function conforms to SVID 3, POSIX, BSD 4.3, ISO 9899. The other functions are from C99.

SEE ALSO

ceil(3), lrint(3), nearbyint(3), rint(3), round(3), trunc(3)
5.3.18 double fmod (double, double);

NAME
fmod - floating-point remainder function

SYNOPSIS
#include <math.h>

double fmod(double x, double y);

DESCRIPTION
The fmod() function computes the remainder of dividing x by y. The return value is x - n * y, where n is the quotient of x / y, rounded towards zero to an integer.

RETURN VALUE
The fmod() function returns the remainder, unless y is zero, when the function fails and errno is set.

ERRORS
EDOM The denominator y is zero.

CONFORMING TO
SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO
drem(3)

5.3.19 double frexp (double, int *);

NAME
frexp - convert floating-point number to fractional and integral components

SYNOPSIS
#include <math.h>

double frexp(double x, int *exp);

DESCRIPTION
The frexp() function is used to split the number x into a normalized fraction and an exponent which is stored in exp.
RETURN VALUE
The `frexp()` function returns the normalized fraction. If the argument
x is not zero, the normalized fraction is x times a power of two, and
is always in the range 1/2 (inclusive) to 1 (exclusive). If x is zero,
then the normalized fraction is zero and zero is stored in exp.

CONFORMING TO
SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO
ldexp(3), modf(3)

GNU 1993-06-06 FREXP(3)

5.3.20 double gamma (double);

NAME
gamma - logarithm of the gamma function

SYNOPSIS
#include <math.h>

double gamma (double x);
float gammaf (float x);
long double gammal (long double x);

DESCRIPTION
The gamma() functions exist for compatibility reasons. They are equiv-
alent to lgamma() etc. Use these. Use of the name gamma() is confusing,
since these functions do not compute the Gamma function, but the natu-
ral logarithm of the Gamma function.

CONFORMING TO
Nonstandard. Compatible with previous mistakes.

SEE ALSO
lgamma(3), signgam(3), tgamma(3)

GNU 2002-08-10 GAMMA(3)

5.3.21 double hypot (double, double);

NAME
hypot - Euclidean distance

SYNOPSIS
#include <math.h>

double hypot (double x, double y);
float hypotf (float x, float y);
long double hypotl (long double x, long double y);

DESCRIPTION
The hypot() function returns the Euclidean distance from the origin
to a point in the plane with coordinates (x, y).

CONFORMING TO
SVID 3, BSD 4.3, ISO 9899

SEE ALSO
math(3)

GNU 2002-08-10 HYPOT(3)
NAME

hypot - Euclidean distance function

SYNOPSIS

#include <math.h>

double hypot(double x, double y);

DESCRIPTION

The hypot() function returns the \( \sqrt{x^2 + y^2} \). This is the length
of the hypotenuse of a right-angle triangle with sides of length \( x \) and
\( y \), or the distance of the point \( (x, y) \) from the origin.

CONFORMING TO

SVID 3, BSD 4.3

SEE ALSO

sqrt(3)

1993-06-25

5.3.22 int isnan (double);

ISINF(3) Linux Programmer's Manual ISINF(3)

NAME

isinf, isnan, finite - test for infinity or not-a-number (NaN)

SYNOPSIS

#include <math.h>

int isinf(double value);

int isnan(double value);

int finite(double value);

DESCRIPTION

The isinf() function returns \(-1\) if value represents negative infinity,
\(1\) if value represents positive infinity, and \(0\) otherwise.

The isnan() function returns a non-zero value if value is "not-a-number" (NaN), and \(0\) otherwise.

The finite() function returns a non-zero value if value is neither
infinite nor a "not-a-number" (NaN) value, and \(0\) otherwise.

CONFORMING TO

BSD 4.3
5.3.23 double j0 (double);

NAME
j0, j0f, j0l, j1, j1f, j1l, jn, jnf, jnl, y0, y0f, y0l, y1, y1f, y1l,
yn, ynf, ynl - Bessel functions

SYNOPSIS
#include <math.h>

double j0(double x);
double j1(double x);
double jn(int n, double x);
double y0(double x);
double y1(double x);
double yn(int n, double x);

float j0f(float x);
float j1f(float x);
float jnf(int n, float x);
float y0f(float x);
float y1f(float x);
float ynf(int n, float x);

long double j0l(long double x);
long double j1l(long double x);
long double jnl(int n, long double x);
long double y0l(long double x);
long double y1l(long double x);
long double ynl(int n, long double x);

DESCRIPTION
The j0() and j1() functions return Bessel functions of x of the first
type of orders 0 and 1, respectively. The jn() function returns the
Bessel function of x of the first kind of order n.

The y0() and y1() functions return Bessel functions of x of the second
kind of orders 0 and 1, respectively. The yn() function returns the
Bessel function of x of the second kind of order n.

For the functions y0(), y1() and yn(), the value of x must be positive.
For negative values of x, these functions return -HUGE_VAL.

The j0f() etc. and j0l() etc. functions are versions that take and
return float and long double values, respectively.

CONFORMING TO
The functions returning double conform to SVID 3, BSD 4.3, XPG4, POSIX
1003.1-2001. The other functions exist by analogy, and exist on several
platforms.

BUGS
There are errors of up to \(2\times10^{-16}\) in the values returned by \(j_0()\), \(j_1()\) and \(jn()\) for values of \(x\) between \(-8\) and \(8\).

5.3.24 double \(j_1\) (double);

See Section 5.3.23 \[j0\], page 38.

5.3.25 double \(jn\) (int, double);

See Section 5.3.23 \[j0\], page 38.

5.3.26 double \(ldexp\) (double, int);

\texttt{LDEXP(3)} Linux Programmer's Manual \texttt{LDEXP(3)}

\textbf{NAME}
ldexp - multiply floating-point number by integral power of 2

\textbf{SYNOPSIS}

\begin{verbatim}
#include <math.h>

double ldexp(double x, int exp);
\end{verbatim}

\textbf{DESCRIPTION}
The \texttt{ldexp()} function returns the result of multiplying the floating-point number \(x\) by \(2\) raised to the power \(exp\).

\textbf{CONFORMING TO}
SVID 3, POSIX, BSD 4.3, ISO 9899

\textbf{SEE ALSO}
frexp(3), modf(3)

\texttt{BSD} \texttt{1993-06-06} \texttt{LDEXP(3)}

5.3.27 double \(lgamma\) (double);

\texttt{LGAMMA(3)} Linux Programmer's Manual \texttt{LGAMMA(3)}

\textbf{NAME}
lgamma - log gamma function

\textbf{SYNOPSIS}

\begin{verbatim}
#include <math.h>

double lgamma(double x);

float lgammaf(float x);

long double lgammal(long double x);
\end{verbatim}
double lgamma_r(double x, int *signp);
float lgammaf_r(float x, int *signp);
long double lgamma_l(long double x, int *signp);

DESCRIPTION
The Gamma function is defined by

\[ \Gamma(x) = \int_0^\infty t^{x-1} e^{-t} \, dt \]

It is defined for every real number except for nonpositive integers. For nonnegative integral \( m \) one has

\[ \Gamma(m+1) = m! \]

and, more generally, for all \( x \):

\[ \Gamma(x+1) = x \times \Gamma(x) \]

For \( x < 0.5 \) one can use

\[ \Gamma(x) \times \Gamma(1-x) = \pi / \sin(\pi x) \]

The \text{lgamma()} function returns the natural logarithm of the absolute value of the Gamma function. The sign of the Gamma function is returned in the external integer \text{signgam} declared in <math.h>. It is 1 when the Gamma function is positive or zero, -1 when it is negative.

Since using a constant location \text{signgam} is not thread-safe, the functions \text{lgamma_r()} etc. have been introduced; they return this sign via the parameter \text{signp}.

For nonpositive integer values of \( x \), \text{lgamma()} returns \text{HUGE_VAL}, sets \text{errno} to \text{ERANGE} and raises the zero divide exception. (Similarly, \text{lgammaf()} returns \text{HUGE_VALF} and \text{lgamma_l()} returns \text{HUGE_VALL}.)

ERRORS
An application wishing to check for error situations should set \text{errno} to zero and call \text{feclearexcept(FE_ALL_EXCEPT)} before calling these functions. On return, if \text{errno} is non-zero or \text{fetestexcept(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW)} is non-zero, an error has occurred.

ERANGE Invalid argument = nonpositive integer value of \( x \).

CONFORMING TO
C99, SVVID 3, BSD 4.3

SEE ALSO
tgamma(3)
5.3.28 double log (double);

See Section 5.3.14 [exp], page 32.

5.3.29 double log10 (double);

See Section 5.3.14 [exp], page 32.

5.3.30 double modf (double, double *);

NAME

modf — extract signed integral and fractional values from floating-point number

SYNOPSIS

#include <math.h>

double modf(double x, double *iptr);

DESCRIPTION

The modf() function breaks the argument x into an integral part and a fractional part, each of which has the same sign as x. The integral part is stored in iptr.

RETURN VALUE

The modf() function returns the fractional part of x.

CONFORMING TO

SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO

frexp(3), ldexp(3)

1993-06-06

5.3.31 double nextafter (double, double);

NAME
	nextafter, nexttoward — floating point number manipulation

SYNOPSIS

#include <math.h>

double nextafter(double x, double y);
float nextafterf(float x, float y);
long double nextafterl(long double x, long double y);

double nexttoward(double x, long double y);
float nexttoward(float x, long double y);
long double nexttowardl(long double x, long double y);

DESCRIPTION
The `nextafter()` functions return the next representable neighbor of `x`
in the direction towards `y`. The size of the step between `x` and the
result depends on the type of the result. If `x = y` the function simply
returns `y`. If either value is NaN, then NaN is returned. Otherwise a
value corresponding to the value of the least significant bit in the
mantissa is added or subtracted, depending on the direction.

The `nexttoward()` functions do the same as the `nextafter()` functions,
except that they have a long double second argument.

These functions will signal overflow or underflow if the result goes
outside of the range of normalized numbers.

CONFORMING TO
C99. This function is defined in IEC 559 (and the appendix with recommended functions in IEEE 754/IEEE 854).

SEE ALSO
`nearbyint(3)`

GNU 2002-08-10 NEXTAFTER(3)

5.3.32 double pow (double, double);

See Section 5.3.14 [exp], page 32.

5.3.33 double sinh (double);

NAME
`sinh` - hyperbolic sine function

SYNOPSIS
```
#include <math.h>

double sinh(double x);
```

DESCRIPTION
The `sinh()` function returns the hyperbolic sine of `x`, which is defined
mathematically as \((\exp(x) - \exp(-x)) / 2\).

CONFORMING TO
SVID 3, POSIX, BSD 4.3, ISO 9899 (C99)

SEE ALSO
`acosh(3), asinh(3), atanh(3), cosh(3), tanh(3)`
5.3.34 double sin (double);

NAME
sin - sine function

SYNOPSIS
#include <math.h>

double sin(double x);

DESCRIPTION
The sin() function returns the sine of x, where x is given in radians.

RETURN VALUE
The sin() function returns a value between -1 and 1.

CONFORMING TO
SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO
acos(3), asin(3), atan(3), atan2(3), cos(3), tan(3)

5.3.35 double sqrt (double);

NAME
sqrt - square root function

SYNOPSIS
#include <math.h>

double sqrt(double x);

DESCRIPTION
The sqrt() function returns the non-negative square root of x. It fails and sets errno to EDOM, if x is negative.

ERRORS
EDOM  x is negative.

CONFORMING TO
SVID 3, POSIX, BSD 4.3, ISO 9899
SEE ALSO
   hypot(3)

5.3.36 double tanh (double);

TANH(3) Linux Programmer's Manual TANH(3)

NAME
tanh - hyperbolic tangent function

SYNOPSIS
   #include <math.h>
   double tanh(double x);

DESCRIPTION
   The tanh() function returns the hyperbolic tangent of x, which is
defined mathematically as sinh(x) / cosh(x).

CONFORMING TO
   SVID 3, POSIX, BSD 4.3, ISO 9899 (C99)

SEE ALSO
   acosh(3), asinh(3), atanh(3), cosh(3), sinh(3)

5.3.37 double tan (double);

TAN(3) Linux Programmer's Manual TAN(3)

NAME
tan - tangent function

SYNOPSIS
   #include <math.h>
   double tan(double x);

DESCRIPTION
   The tan() function returns the tangent of x, where x is given in radi-
   ans.

CONFORMING TO
   SVID 3, POSIX, BSD 4.3, ISO 9899

SEE ALSO
acos(3), asin(3), atan(3), atan2(3), cos(3), sin(3)

5.3.38 double y0 (double);

See Section 5.3.23 [j0], page 38.

5.3.39 double y1 (double);

See Section 5.3.23 [j0], page 38.

5.3.40 double yn (int, double);

See Section 5.3.23 [j0], page 38.

5.3.41 Sample application

Let us illustrate how to use FDLIBM features over a very simple application example. You will find this example on the Robosoft development toolchain under the ‘apps6/floatv6-0.1/’ directory:

[guest@localhost apps6]$ [guest@localhost apps6]$ pwd /home/guest/syndex/last/apps6 [guest@localhost apps6]$
 [guest@localhost apps6]$ [guest@localhost apps6]$ [guest@localhost apps6]$ cd floatv6-0.1/
 [guest@localhost floatv6-0.1]$ [guest@localhost floatv6-0.1]$ 11
 total 32
 -rw-r--r-- 1 guest guest 2542 Apr 8 12:34 GNUmakefile
 -rw-r--r-- 1 guest guest 1018 Feb 19 18:10 HP.m4
 -rw-r--r-- 1 guest guest 156 Feb 20 14:34 myLib.c
 -rw-r--r-- 1 guest guest 1063 Feb 19 18:10 P0.m4
 -rw-r--r-- 1 guest guest 209 Feb 19 18:10 test.m4
 -rw-r--r-- 1 guest guest 671 Apr 6 18:03 test.m4x
 -rw-r--r-- 1 guest guest 1445 Feb 19 18:10 test.sdx
 -rw-r--r-- 1 guest guest 101 Apr 8 12:34 User.mk
 [guest@localhost floatv6-0.1]$
Here is a screen capture, from the SynDEx CAD session, showing the sample application.

The hardware architecture is composed of one MPC555 operator (named P0) and a PC operator (running RTAI/Linux, named HP). The designed algorithm implements a short sequence:

- read an analog port on P0 (that performs an analog to digital conversion)
- use the result of the conversion (a 32-bits integer) as input for function `func` (running on P0). `func` calls a separately compiled C function (we will define farther) and produces two values: a 32-bits integer (passed through output o0) and a floating point value (passed through output o1).
- send these two values (through the CAN bus) to operator HP, for being displayed.

We can see in the SynDEx CAD main window that `func` is a function of type `myFunc`. `myFunc` macro is defined by user into the file `test.m4x`.

```plaintext
[guest@localhost floatv6-0.1]$ [guest@localhost floatv6-0.1]$ cat test.m4x
divert(-1)
define('NOTRACEDEF')
define('ECP_IRQ','0x5')
define('CAN_speed_','800Kbps')
define('loop_period_','10000000')

# Robosoft MPC555 boards Serial Number (used by the download process)
define('PO_CANID_','0x4000')

# PP: Fix IT timer period to 156/15625=10ms (refer to 555.m4x comments).
dnl define('PIT_COUNTER', 1562)

# --------------
# myFunc()
#
define('myFunc', 'ifelse(dnl
MGC,'LOOP', 'Ccall_(void, 'myCall', int $1, int *$2, float *$3'))')

# --------------
```
# display()
#
#define('display', 'ifelse(MGC,'LOOP', 'dnl
{
int f = 0;
    f = (int)(*$1 + *$2);
    rt_printk("Sum of input values is %5d.\n", f);
})')

divert' 'dnl
[guest@localhost floatv6-0.1]$ 

The user macro myFunc appears to be defined as a call to the myCall function (separately compile). This is done using the Ccall_ macro. For advanced information about Ccall_ SynDEx macro, please refer to SynDEx online documentations and to the Robosoft document titled The 555.m4x and RSB.m4x SynDEx Macro-Executives [See 'doc/ROBO/555macros/555macros.pdf']
.

The myCall function is declared in the local file 'myLib.c'. Remark, that the function prototype must fit the prototype declaration passed to Ccall_ during the myFunc macro definition:

[guest@localhost floatv6-0.1]$ 
[guest@localhost floatv6-0.1]$ cat myLib.c 
#include "fdlibm.h"

void myCall (int in, int *iout, float *fout) 
{
    double f = 0.75;
    *iout = (int)(100 * sin(f));
    *fout = (float)(sin(in / 100.));
}
[pierre@Gromit floatv6-0.1]$ 

Now, just remains to add the appropriate makefile rules to the local 'GNUmakefile' and 'User.mk', in order to launch 'myLib.c' compilation automatically. 'myLib.c' should be compile to produce an object file named 'myLib.555'. Hence, called functions will be copied from 'myLib.555' and included into P0 operator object file. At final compilation step, the composed P0 operator object file will be linked to produce the PowerPC ELF 32-bits executable binary.

To do so, we have to declare the object file 'myLib.555' as a valid library for P0 operator. This is done by setting the following variable into the local 'GNUmakefile':

P0.libs = myLib.555

Then, we have to give the makefile rule needed for producing 'myLib.555'. This is done by adding the following line to the local 'User.mk':

myLib.555 : myLib.c

In fact, this rule only sets a dependency between 'myLib.555' and 'myLib.c'. This means, that for compiling 'myLib.c' an implicit rule will be applied. This implicit rule corresponds to the one already defined by the 'mpc555.m4m' SynDEx kernel macros. If user has any specific needs for compilation, he can write the convenient rule instead. For instance:

myLib.555 : myLib.c
   ./.crossgcc/bin/ppc-elf32-gcc -o myLib.555 -mcpu=powerpc
-mhard-float -I../crossgcc/fdlibm -O2 -c myLib.c

Now, user can simply start application by executing `make` from the shell command line. `myLib.c` compilation will be automatically done before producing `Po`.
6 Launching applications

In this section we give a brief summary of files you will require to generate and compile your executive. Code generation principles will be detailed in next sections. Files required are:

- One `processorName.m4` file for each processor specified in your main architecture, including `root.m4` (generally the file generated for the main operator). These files are generated by SynDEx during the executive generation step;
- `applicationName.m4m`;
- `applicationName.m4x`, which may be empty, but which generally contains the application dependent macros (written by the application programmer);
- `GNUmakefile` application dependent top makefile rules;
- `User.mk` another application dependent set of makefile rules;

For the files which are not generated by SynDEx, most of the time you can simply copy existing ones (for instance from the example directory) and make modifications explained in the comments of these files.

Once you gathered all these files, type `make` in your shell. This will compile, download and execute your application over the hardware architecture. You can also clean your directory by typing `make clean`.

For instance, if user guest want to execute the `robuca`, he will have to execute the following commands from the shell command line:

```
[guest@localhost robucarAna30-0.9-v6.6.7]$ make
[guest@localhost robucarAna30-0.9-v6.6.7]$ total 56
-rw-r--r--  1 guest  guest  3860 Jan 26 17:30 f555.m4
-rw-------  1 guest  guest  2523 Feb 20 14:28 GNUmakefile
-rw-r--r--  1 guest  guest  7739 Jan 26 17:30 robucar.m4
-rw-r--r--  1 guest  guest  290 Jan 26 17:30 robucar.m4
-rw-------  1 guest  guest  11333 Apr  6 18:00 robucar.m4x
-rw-------  1 guest  guest  14362 Jan 26 17:03 robucar.sdx
-rw-r--r--  1 guest  guest  2008 Jan 26 17:30 root.m4
-rw-r--r--  1 guest  guest    78 Feb  5 18:00 User.mk
```

```
[guest@localhost robucarAna30-0.9-v6.6.7]$ make clean
GNUmakefile:64: robucar.mk: No such file or directory
m4 robucar.m4 >robucar.mk
rm -f robucar.mk *~ *.o *.a
rm -f root root.map root.o root.c run *
rm -f f555 f555.elf f555.555 f555.s f555.elf.map
rm -f r555 r555.elf r555.555 r555.s r555.elf.map
[guest@localhost robucarAna30-0.9-v6.6.7]$ make
```

On success, the following message will be displayed:

```
==== RT memory manager v1.3 Loaded. ====  
```

```
***** STARTING THE UP REAL TIME SCHEDULER WITH LINUX *****  
***** FP SUPPORT AND READY FOR A PERIODIC TIMER *****  
***** LINUX TICK AT 100 (HZ) <****  
***** CALIBRATED CPU FREQUENCY 233871000 (HZ) <****  
***** CALIBRATED TIMER-INTERRUPT-TO-SCHEDULER LATENCY 2689 (ns) <****  
```
**PRELIMINARY INFORMATION**

**CALIBRATED ONE SHOT SETUP TIME 2009 (ns)**

Probed CAN base address: 0xE000
Probed CAN IRQ: 0xB

Downloading processor ID 0x4000 station!... loaded.
Downloading processor ID 0x4001 station!... loaded.

Download complete... application is running.

If a problem occurs during the download sequence, the following message will be displayed:

**Download failed. Please try again:**

1) make stop  
2) reset MPC555 stations  
3) make

[guest@localhost robucarAna30-0.9-v6.6.7]$ make stop

It indicates the procedure to follow for trying to download again, i.e.

- executing `make stop`
- resetting MPC555 boards
- executing `make` one more time

Last, for stopping a running application, user just has to execute:

[guest@localhost robucarAna30-0.9-v6.6.7]$ make stop

What will displays:

[guest@localhost robucarAna30-0.9-v6.6.7]$ make stop sync  
`./../rtai/rtai-24.1.12/scripts/rt_rmmod root` using sudo

```
sudo sh -c "/sbin/rmmod -r root"
rmm: module root not loaded
```

```
./../rtai/rtai-24.1.12/scripts/rt_rmmod rtai_shm
using sudo

sudo sh -c "/sbin/rmmod -r rtai_shm"
rmm: module rtai_shm not loaded
```

```
./../rtai/rtai-24.1.12/scripts/rt_rmmod rtai_fifos
using sudo

sudo sh -c "/sbin/rmmod -r rtai_fifos"
rmm: module rtai_fifos not loaded
```

```
./../rtai/rtai-24.1.12/scripts/rt_rmmod rtai_sched_up
using sudo

sudo sh -c "/sbin/rmmod -r rtai_sched_up"
rmm: module rtai_sched_up not loaded
```

```
./../rtai/rtai-24.1.12/scripts/rt_rmmod rtai
```

sudo sh -c "/sbin/rmmod -r rtai"
Chapter 6: Launching applications

rmmod: module rtai not loaded
===== RT memory manager max allocated: 4016 bytes.
===== RT memory manager kernel kmalloc()'s: 4
===== RT memory manager kernel kfree()'s: 4
===== RT memory manager overhead: 0 bytes.
===== RT memory manager leaks: 0 bytes.

===== RT memory manager v1.3 Unloaded. =====

***** THE UP REAL TIME SCHEDULER HAS BEEN REMOVED *****

ADEOS: Domain RTAI unregistered.
RTAI/Adeos unmounted.

[pierre@Gromit robucarAna30-0.9-v6.6.7]$ 

IMPORTANT REMARK: depending on the shell environment you are working on, the mentioned messages may not appear on the screen... In this case, use the commands dmesg or tail -f /var/log/messages for displaying them.
**Document Index**

This is an alphabetical listing of all the commands, variables, and topics discussed in this document.

A

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>App2flash</td>
<td>23</td>
</tr>
<tr>
<td>App2flash, CAN baudrate</td>
<td>22, 24</td>
</tr>
<tr>
<td>Application, bootloader</td>
<td>22</td>
</tr>
<tr>
<td>Application, CAN baudrate</td>
<td>7, 9</td>
</tr>
<tr>
<td>Application, CAN IRQ</td>
<td>8</td>
</tr>
<tr>
<td>Application, CAN_speed</td>
<td>7, 9</td>
</tr>
<tr>
<td>Application, dependent rules</td>
<td>16</td>
</tr>
<tr>
<td>Application, display</td>
<td>51</td>
</tr>
<tr>
<td>Application, ECP_IRQ</td>
<td>8</td>
</tr>
<tr>
<td>Application, flash</td>
<td>23</td>
</tr>
<tr>
<td>Application, GNUmakefile</td>
<td>16</td>
</tr>
<tr>
<td>Application, makefile rules</td>
<td>16</td>
</tr>
<tr>
<td>Application, name</td>
<td>17</td>
</tr>
<tr>
<td>Application, specific dependencies</td>
<td>17</td>
</tr>
<tr>
<td>Application, standalone</td>
<td>49</td>
</tr>
<tr>
<td>Application, start</td>
<td>51</td>
</tr>
<tr>
<td>Application, stop</td>
<td>51</td>
</tr>
<tr>
<td>Application, variable</td>
<td>17</td>
</tr>
<tr>
<td>Architecture components</td>
<td>2</td>
</tr>
<tr>
<td>as</td>
<td>3</td>
</tr>
</tbody>
</table>

B

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bash, bashrc</td>
<td>16</td>
</tr>
<tr>
<td>Bash, environment variables</td>
<td>16</td>
</tr>
<tr>
<td>BDM</td>
<td>23, 25</td>
</tr>
<tr>
<td>Binary, download</td>
<td>20</td>
</tr>
<tr>
<td>Binutils</td>
<td>3</td>
</tr>
<tr>
<td>Binutils, as</td>
<td>3</td>
</tr>
<tr>
<td>Binutils, assembler</td>
<td>3</td>
</tr>
<tr>
<td>Binutils, documentation</td>
<td>3</td>
</tr>
<tr>
<td>Binutils, download</td>
<td>3</td>
</tr>
<tr>
<td>Binutils, installation</td>
<td>13</td>
</tr>
<tr>
<td>Binutils, ld</td>
<td>3</td>
</tr>
<tr>
<td>Binutils, linker</td>
<td>3</td>
</tr>
<tr>
<td>BIOS, configure</td>
<td>7</td>
</tr>
<tr>
<td>BIOS, IRQ</td>
<td>7</td>
</tr>
<tr>
<td>Bootapp</td>
<td>24</td>
</tr>
<tr>
<td>Bootloader, CAN baudrate</td>
<td>22</td>
</tr>
<tr>
<td>Bootloader, identifier</td>
<td>22</td>
</tr>
<tr>
<td>Bootloader, image</td>
<td>22</td>
</tr>
<tr>
<td>Bootloader, S19 binary format</td>
<td>22</td>
</tr>
<tr>
<td>Bootloader, tools</td>
<td>22</td>
</tr>
</tbody>
</table>

C

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C programming</td>
<td>25, 45</td>
</tr>
<tr>
<td>C programming, Ccall, macro example</td>
<td>23</td>
</tr>
<tr>
<td>C programming, mathematical functions</td>
<td>25, 45</td>
</tr>
<tr>
<td>C/C++ API interface</td>
<td>15</td>
</tr>
<tr>
<td>CAN bus</td>
<td>2</td>
</tr>
<tr>
<td>CAN bus, CAN dongle</td>
<td>2</td>
</tr>
<tr>
<td>CAN bus, ECP/EPP board</td>
<td>2</td>
</tr>
<tr>
<td>CAN bus, parallel port</td>
<td>2</td>
</tr>
<tr>
<td>CAN bus, PCI board</td>
<td>2</td>
</tr>
<tr>
<td>CAN bus, Philips SJA1000.</td>
<td>2</td>
</tr>
<tr>
<td>CAN bus, specifications</td>
<td>2</td>
</tr>
<tr>
<td>CAN Dongle, drivers</td>
<td>7</td>
</tr>
<tr>
<td>CAN Dongle, IRQ</td>
<td>7</td>
</tr>
<tr>
<td>CAN Dongle, kmod</td>
<td>8</td>
</tr>
<tr>
<td>CAN Dongle, parport</td>
<td>8</td>
</tr>
<tr>
<td>CAN Dongle, realtime support</td>
<td>7</td>
</tr>
<tr>
<td>CAN, ADLink</td>
<td>6</td>
</tr>
<tr>
<td>CAN, BIOS</td>
<td>6</td>
</tr>
<tr>
<td>CAN, CAN Dongle</td>
<td>7</td>
</tr>
<tr>
<td>CAN, CAN_speed</td>
<td>7, 9</td>
</tr>
<tr>
<td>CAN, Dongle, ECP_IRQ</td>
<td>8</td>
</tr>
<tr>
<td>CAN, drivers</td>
<td>6, 7</td>
</tr>
<tr>
<td>CAN, IRQ</td>
<td>6, 7, 8</td>
</tr>
<tr>
<td>CAN, PCI</td>
<td>6, 7</td>
</tr>
<tr>
<td>CAN, PCI 7841</td>
<td>6</td>
</tr>
<tr>
<td>CAN, PCI board</td>
<td>6</td>
</tr>
<tr>
<td>CAN_speed</td>
<td>7, 9</td>
</tr>
<tr>
<td>Communication media</td>
<td>2</td>
</tr>
<tr>
<td>Communication media, CAN bus</td>
<td>2</td>
</tr>
<tr>
<td>Communication media, Ethernet</td>
<td>2</td>
</tr>
<tr>
<td>Cross compiler</td>
<td>12</td>
</tr>
<tr>
<td>Cross compiler, build</td>
<td>12</td>
</tr>
<tr>
<td>Cross compiler, documentation</td>
<td>13</td>
</tr>
<tr>
<td>Cross compiler, installation</td>
<td>12</td>
</tr>
<tr>
<td>Cross compiler, path</td>
<td>17</td>
</tr>
<tr>
<td>Cross compiler, required files</td>
<td>3</td>
</tr>
</tbody>
</table>

D

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development tools</td>
<td>3</td>
</tr>
<tr>
<td>Development tools, binutils</td>
<td>3</td>
</tr>
<tr>
<td>Development tools, documentation</td>
<td>3</td>
</tr>
<tr>
<td>Development tools, download</td>
<td>3</td>
</tr>
<tr>
<td>Development tools, ftilibm</td>
<td>3</td>
</tr>
<tr>
<td>Development tools, gcc core</td>
<td>3</td>
</tr>
<tr>
<td>Development tools, newlib</td>
<td>3</td>
</tr>
<tr>
<td>Displays</td>
<td>51</td>
</tr>
<tr>
<td>Displays, dmesg</td>
<td>51</td>
</tr>
<tr>
<td>Displays, kernel messages</td>
<td>51</td>
</tr>
<tr>
<td>Displays, tail</td>
<td>51</td>
</tr>
<tr>
<td>Distributed applications</td>
<td>3</td>
</tr>
<tr>
<td>Distributed applications, design</td>
<td>3</td>
</tr>
<tr>
<td>Dmesg</td>
<td>51</td>
</tr>
<tr>
<td>Download, binaries</td>
<td>20</td>
</tr>
<tr>
<td>Download, ELF files</td>
<td>20</td>
</tr>
<tr>
<td>Download, failure</td>
<td>21</td>
</tr>
<tr>
<td>Download, MPC555</td>
<td>20</td>
</tr>
<tr>
<td>Download, realtime fifo</td>
<td>20</td>
</tr>
<tr>
<td>Download, rebooting</td>
<td>21</td>
</tr>
<tr>
<td>Download, SynDEx download tools</td>
<td>20</td>
</tr>
<tr>
<td>Dwbin, download</td>
<td>20</td>
</tr>
<tr>
<td>Dwbin, error messages</td>
<td>20</td>
</tr>
<tr>
<td>Dwbin, example</td>
<td>20</td>
</tr>
<tr>
<td>Dwbin, failure</td>
<td>21</td>
</tr>
<tr>
<td>Dwbin, syntax</td>
<td>20</td>
</tr>
<tr>
<td>Dwbin, utility</td>
<td>20</td>
</tr>
</tbody>
</table>

E

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECP</td>
<td>2</td>
</tr>
<tr>
<td>ECP, specifications</td>
<td>2</td>
</tr>
<tr>
<td>ECP_IRQ</td>
<td>8</td>
</tr>
<tr>
<td>ELF files, conversion</td>
<td>20</td>
</tr>
<tr>
<td>ELF files, download</td>
<td>20</td>
</tr>
</tbody>
</table>
ELF files, preparing for download ........................................ 20
Environment variables .................................................. 16
Ethernet ........................................................................... 2

F
FDLIBM ................................................................. 3, 25, 45
FDLIBM, Ccall, macro .................................................... 45
FDLIBM, documentation .................................................. 3
FDLIBM, download .......................................................... 3
FDLIBM, example ........................................................... 45
FDLIBM, functions .......................................................... 25
FDLIBM, functions, acos .................................................. 26
FDLIBM, functions, acosh ............................................... 25
FDLIBM, functions, asin ................................................. 27
FDLIBM, functions, asinh ................................................ 26
FDLIBM, functions, atan .................................................. 29
FDLIBM, functions, atan2 .............................................. 27
FDLIBM, functions, atanh .............................................. 28
FDLIBM, functions, cbrt .................................................. 29
FDLIBM, functions, ceil .................................................. 30
FDLIBM, functions, cos .................................................. 31
FDLIBM, functions, cosh ................................................. 30
FDLIBM, functions, erf .................................................... 31
FDLIBM, functions, erfc .................................................. 32
FDLIBM, functions, exp ................................................... 32
FDLIBM, functions, fabs ................................................ 33
FDLIBM, functions, finite .............................................. 34
FDLIBM, functions, floor ............................................... 34
FDLIBM, functions, fmod .............................................. 35
FDLIBM, functions, frexp .............................................. 35
FDLIBM, functions, gamma .......................................... 36
FDLIBM, functions, hypot .............................................. 36
FDLIBM, functions, isnan .............................................. 37
FDLIBM, functions, j0 ................................................... 38
FDLIBM, functions, jl .................................................... 39
FDLIBM, functions, jn .................................................... 39
FDLIBM, functions, ldexp .............................................. 39
FDLIBM, functions, lgamma ......................................... 39
FDLIBM, functions, log ............................................... 41
FDLIBM, functions, log10 ............................................. 41
FDLIBM, functions, modf ............................................. 41
FDLIBM, functions, nextafter ....................................... 41
FDLIBM, functions, pow .............................................. 42
FDLIBM, functions, sin ................................................ 43
FDLIBM, functions, sinh .............................................. 42
FDLIBM, functions, sqrt ............................................. 43
FDLIBM, functions, tan ................................................ 44
FDLIBM, functions, tanh ............................................. 44
FDLIBM, functions, y0 ............................................... 45
FDLIBM, functions, y1 ............................................... 45
FDLIBM, functions, yn ............................................... 45
FDLIBM, installation ................................................... 13
FDLIBM, mathematical functions .................................... 25
FDLIBM, sample code .................................................. 45
Flash, application ...................................................... 23
Flash, BDM ............................................................. 23, 25
Flash, bootloader ......................................................... 22
Flash, image .............................................................. 23
Flash, programmer ......................................................... 23, 25
Flash, S19 binary format .............................................. 22, 24
Flash, tools ................................................................. 23
Floating point operations .................................................. 25, 45
Floating point operations, example .................................. 45
Floating point operations, sample code ............................. 45

G
gcc ................................................................. 3
Gcc core ............................................................... 3
Gcc core, documentation .............................................. 3
Gcc core, download ..................................................... 3
Gcc core, installation ................................................... 13
GNUmakefile .............................................................. 16
GNUmakefile, application dependent rules .................. 16

H
Heterogeneous applications ........................................... 3
Heterogeneous applications, designing ......................... 3

I
Identifier, bootloader .................................................. 22
IEEE 754 floating-point arithmetic ................................. 25, 45
IRQ, affect ............................................................... 7
IRQ, BIOS ............................................................... 7
IRQ, CAN device ......................................................... 7
IRQ, conflict ............................................................. 6, 7

K
Kernel, displays .......................................................... 51
Kernel, dbms ............................................................ 51
 Kernel, messages ....................................................... 51
Kernel, tail ............................................................... 51
kmod ................................................................. 8
kmod, options ........................................................... 8

L
1d ................................................................. 3
Linker, binary sections .................................................. 15
Linker, scripts .......................................................... 15
Linux ................................................................. 3
Linux, download ......................................................... 3
Linux, kernel ........................................................... 3
Linux, libraries path ................................................... 16
Linux, path ............................................................. 16
Linux, sources path ................................................... 16
Linux, version .......................................................... 3
LinuxIO ................................................................. 15

M
M4, gm4 ................................................................. 17
M4, m4 ................................................................. 17
M4, variable ............................................................. 17
macroprocessor .......................................................... 17, 19
macroprocessor, name variable .................................... 17
macroprocessor, use ................................................... 19
Makefile ............................................................... 18
Makefile, user rules .................................................... 18
Mathematical functions ................................................. 25
Mathematical functions, example .................................... 45
Mathematical functions, sample code ............................. 45
Motorola MPC555 ......................................................... 2
MPC555 ................................................................. 2
MPC555, bootloader ..................................................... 22
MPC555, CAN bus ........................................................ 2
MPC555, Cross compiler ............................................... 3
MPC555, flash .......................................................... 23
MPC555, linker scripts ................................................. 15
| MPC555, main operator | 23 |
| MPC555, memory mapping | 15 |
| MPC555, rebooting | 21 |
| MPC555, root operator | 23 |
| MPC555, standalone | 23 |
| MPC555, Cross compiler, installation | 12 |

| N |
| Newlib | 3 |
| Newlib, documentation | 3 |
| Newlib, download | 3 |
| Newlib, installation | 13 |

| P |
| Parallel port | 2 |
| Parallel port, CAN dongle | 2 |
| Parallel port, ECP | 2 |
| parport | 8 |
| parport, base address | 8 |
| parport, IRQ | 8 |
| parport, module | 8 |
| parport, options | 8 |
| PC, CAN bus | 2 |
| PC, Ethernet | 2 |
| PCI 7841, BIOS | 6 |
| PCI 7841, CAN board | 6 |
| PCI 7841, drivers | 6 |
| PCI 7841, IRQ | 6 |
| PCI 7841, realtime support | 6 |

| R |
| Realtime | 3 |
| Realtime, distribution | 3 |
| Realtime, software | 3 |
| RTAI | 3 |
| RTAI, distribution | 3 |
| RTAI, download | 3 |
| RTAI, installation | 10 |
| RTAI, installation path | 17 |
| RTAI, kernel | 3 |
| RTAI, loading modules | 10 |
| RTAI, modules | 10 |
| RTAI, modutils access | 10 |
| RTAI, path | 17 |
| RTAI, user access | 10 |
| RTAI, users rights | 10 |
| RTAI, version | 3 |

| S |
| S19 binary format | 22, 24 |
| Sdx2img | 24 |
| Start, application | 49 |
| Start, display | 49 |

| Stop, application | 51 |
| Stop, display | 51 |
| Sudo, `/etc/sudoers` | 10 |
| Sudo, configuration | 10 |
| Sudo, editing | 10 |
| Sudo, users rights | 10 |
| SynDEx | 3 |
| SynDEx, distribution | 3 |
| SynDEx, documentation | 3 |
| SynDEx, download | 3 |
| SynDEx, flashing | 23 |
| SynDEx, installation | 14 |
| SynDEx, macros | 3 |
| SynDEx, macros, CAN bus | 3 |
| SynDEx, macros, communication media | 3 |
| SynDEx, macros, Linux | 3 |
| SynDEx, macros, linuxIO | 15 |
| SynDEx, macros, Motorola MPC555 | 3, 15 |
| SynDEx, macros, processor | 3 |
| SynDEx, macros, Robosoft control board | 3 |
| SynDEx, macros, RTAI | 3, 15 |
| SynDEx, macros, x86 PC | 3 |
| SynDEx, standalone applications | 23 |
| SynDEx, version | 3 |

| T |
| Toolchain | 3 |
| Toolchain, `.bashrc` | 16 |
| Toolchain, as | 3 |
| Toolchain, bash | 16 |
| Toolchain, binutils | 3 |
| Toolchain, documentation | 3 |
| Toolchain, download | 3 |
| Toolchain, environment variables | 16 |
| Toolchain, find | 3 |
| Toolchain, gcc | 3 |
| Toolchain, ld | 3 |
| Toolchain, newlib | 3 |
| Toolchain, path | 17 |
| Trigonometric functions | 25, 45 |
| Trigonometric functions, example | 45 |
| Trigonometric functions, sample code | 45 |

| U |
| User interface, C/C++ | 15 |
| User interface, linuxIO | 15 |
| User.mk | 18 |
| Utils, app2flash | 23 |
| Utils, bootapp | 24 |
| Utils, path | 17 |
| Utils, sdx2img | 24 |