PacMAN Programming User Manual

Pack Management System

Last Updated 3/2/2017, Emilie Grybos

Originally Written 12/19/15, John Gehrig

Overview

This document describes the methods and procures to build and upload code to the Rev 0.3 and later PACMAN boards.

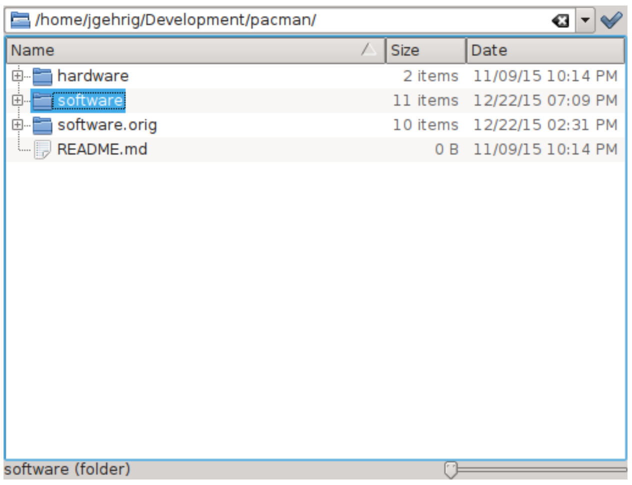
Prerequisites

You will need the following programs installed on the development machine. The procedure is designed for a Linux environment; the Oracle VM VirtualBox on the computer in the safety zone room 400 is useful for installation as all the necessary programs are installed.

- Oracle VM VirtualBox Information:
    - Username: “osboxes”
    - Password: “password”
    - Root Password: “password”
- AVR-GCC (http://www.nongnu.org/avr-libc/)
  (For details see http://avr-eclipse.sourceforge.net/wiki/index.php/The_AVR_GCC_Toolchain)
- GNU Make (https://www.gnu.org/software/make/)
- AVRDUDE (http://www.nongnu.org/avrdude/)
- Atmel ICE Basic Programmer

Software Build Procedure

All source code is located in the 'software' folder of the PacMAN repository; this has been uploaded to the 2017 website in a PacMAN software zip folder (https://sites.lafayette.edu/ece492-sp17/subsystems/tsv/pacman/).

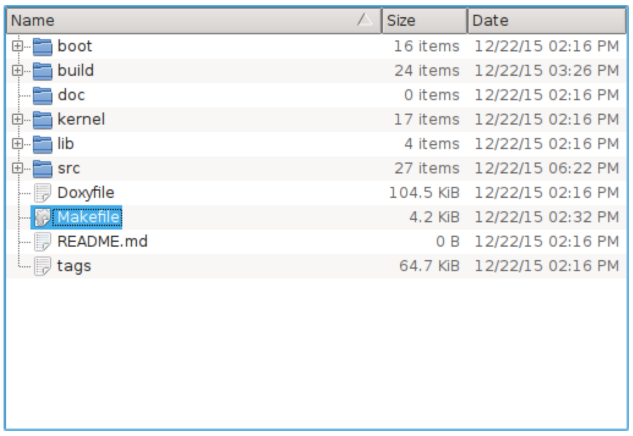


/home/jgehrig/Development/pacman/

Name	Size	Date
hardware	2 items	11/09/15 10:14 PM
software	11 items	12/22/15 07:09 PM
software.orig	10 items	12/22/15 02:31 PM
README.md	0 B	11/09/15 10:14 PM

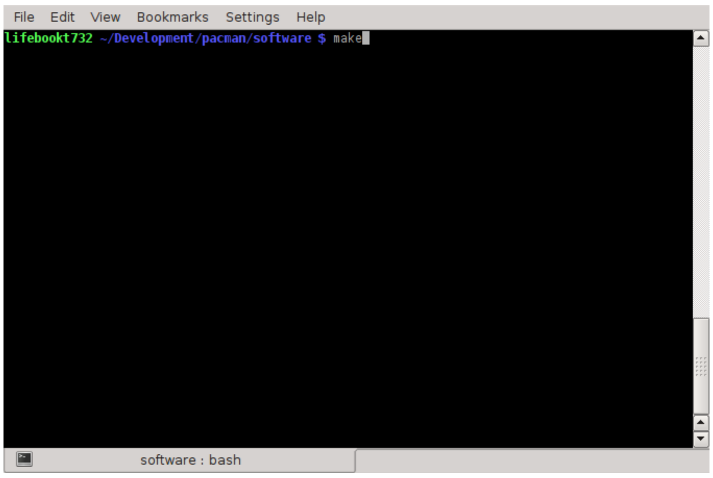
software (folder)

Navigate to the folder containing 'Makefile'



Name	Size	Date
boot	16 items	12/22/15 02:16 PM
build	24 items	12/22/15 03:26 PM
doc	0 items	12/22/15 02:16 PM
kernel	17 items	12/22/15 02:16 PM
lib	4 items	12/22/15 02:16 PM
src	27 items	12/22/15 06:22 PM
Doxyfile	104.5 KiB	12/22/15 02:16 PM
Makefile	4.2 KiB	12/22/15 02:32 PM
README.md	0 B	12/22/15 02:16 PM
tags	64.7 KiB	12/22/15 02:16 PM

Open a terminal and type `make`



```
lifebookt732 ~/Development/pacman/software $ make
```

Here is an example of “good” make output

```
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/main.c -o build/main.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/i2c.c -o build/i2c.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/task_gui.c -o build/task_gui.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/task_watchdog.c -o build/task_watchdog.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/task_control.c -o build/task_control.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/task_safety.c -o build/task_safety.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/task_config.c -o build/task_config.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/task_canbus.c -o build/task_canbus.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib src/tasklist.c -o build/tasklist.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -x assembler-with-cpp -Isrc -Ikernel kernel/atomport-asm.s -o
build/atomport-asm.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -I. -Isrc kernel/atomkernel.c -o build/atomkernel.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -I. -Isrc kernel/atomsem.c -o build/atomsem.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -I. -Isrc kernel/atommutex.c -o build/atommutex.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -I. -Isrc kernel/atomtimer.c -o build/atomtimer.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -I. -Isrc kernel/atomqueue.c -o build/atomqueue.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -I. -Isrc kernel/atomport.c -o build/atomport.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib lib/uart/uart.c -o build/uart.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib lib/atmel/TWI_Master.c -o build/TWI_Master.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib lib/atmel/can_drv.c -o build/can_drv.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib lib/atmel/can_lib.c -o build/can_lib.o
avr-gcc -c -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 -Isrc -Ikernel -Ilib lib/lcd/lcd.c -o build/lcd.o
#avr-gcc -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 build/main.o build/main.o build/i2c.o build/task_gui.o build/task_watchdog.o
build/task_control.o build/task_safety.o build/task_config.o build/task_canbus.o build/tasklist.o build/atomport-asm.o
build/atomkernel.o build/atomsem.o build/atommutex.o build/atomtimer.o build/atomqueue.o build/atomport.o build/uart.o
build/TWI_Master.o build/can_drv.o build/can_lib.o build/lcd.o -o build/pacman.elf -Wl,-Map,build/pacman.map
echo build/main.o build/i2c.o build/task_gui.o build/task_watchdog.o build/task_control.o build/task_safety.o build/task_config.o
build/task_canbus.o build/tasklist.o build/atomport-asm.o build/atomkernel.o build/atomsem.o build/atommutex.o build/atomtimer.o
build/atomqueue.o build/atomport.o build/uart.o build/TWI_Master.o build/can_drv.o build/can_lib.o build/lcd.o
build/main.o build/i2c.o build/task_gui.o build/task_watchdog.o build/task_control.o build/task_safety.o build/task_config.o
build/task_canbus.o build/tasklist.o build/atomport-asm.o build/atomkernel.o build/atomsem.o build/atommutex.o build/atomtimer.o
build/atomqueue.o build/atomport.o build/uart.o build/TWI_Master.o build/can_drv.o build/can_lib.o build/lcd.o
avr-gcc -g -Os -mmcu=at90can32 -Wall -Werror -std=c99 build/main.o build/i2c.o build/task_gui.o build/task_watchdog.o
build/task_control.o build/task_safety.o build/task_config.o build/task_canbus.o build/tasklist.o build/atomport-asm.o
build/atomkernel.o build/atomsem.o build/atommutex.o build/atomtimer.o build/atomqueue.o build/atomport.o build/uart.o
build/TWI_Master.o build/can_drv.o build/can_lib.o build/lcd.o -o build/pacman.elf -Wl,-Map,build/pacman.map
Building pacman.hex
avr-objcopy -j .text -j .data -O ihex build/pacman.elf build/pacman.hex
avr-size -C --mcu=at90can32 build/pacman.elf
AVR Memory Usage
----------------
Device: at90can32

Program:   11636 bytes (35.5% Full)
(.text + .data + .bootloader)

Data:        994 bytes (48.5% Full)
(.data + .bss + .noinit)
```

Make will output a file in the build directory based on the variable 'TARGET_NAME'. The default output is to build/pacman.hex. This file can be manually uploaded through avrdude, or automatically from make.

NOTE: Device resource usage is listed at the bottom of the make file.

After this completes, run `sudo make fuse` command; this will set the lower byte of the fusing outputs to set the clock speed. If this command is not run, the PacMAN will run off the internal clock instead of the external oscillator. This is an example of a good output for this command:

```
osboxes@osboxes:~/Desktop/Pacman_Software_0_15_Emilie_Marty/Software$ sudo make fuse
#see http://www.engbedded.com/fusecalc/ for details on what these settings do
avrdude -F -V -c atmelice -p c128 -U lfuse:w:0xcf:m -U hfuse:w:0x99:m -U efuse:w:0xff:m

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.09s

avrdude: Device signature = 0x1e9781 (probably c128)
avrdude: reading input file "0xcf"
avrdude: writing lfuse (1 bytes):

Writing | ################################################## | 100% 0.01s

avrdude: 1 bytes of lfuse written
avrdude: reading input file "0x99"
avrdude: writing hfuse (1 bytes):

Writing | ################################################## | 100% 0.01s

avrdude: 1 bytes of hfuse written
avrdude: reading input file "0xff"
avrdude: writing efuse (1 bytes):

Writing | ################################################## | 100% 0.01s

avrdude: 1 bytes of efuse written

avrdude: safemode: Fuses OK (E:FF, H:99, L:CF)

avrdude done.  Thank you.
```

Adding Files to the Project

SRC_DIR – Directory containing all source files.

LIB_DIR – Directory containing all library files.

BUILD_DIR – The directory to which program output should be directed.

APP_OBJECTS – A list of individual source files, must have a .o extension

LIB_OBJECTS – A list of library source files, must have a .o extension

To add a file to the project, you must first dermine if it should be treated as a library, or as a source file. Library files should be added to the LIB_OBJECTS variable with the .c extension replaced with .o, folder path should be included. Source files should be added to the APP_OBJECTS variable, no folder path is required, but the extension should also be converted from .c to .o.

For example, to add the source file src/foo.c to the project, the following code would be added to the make file.

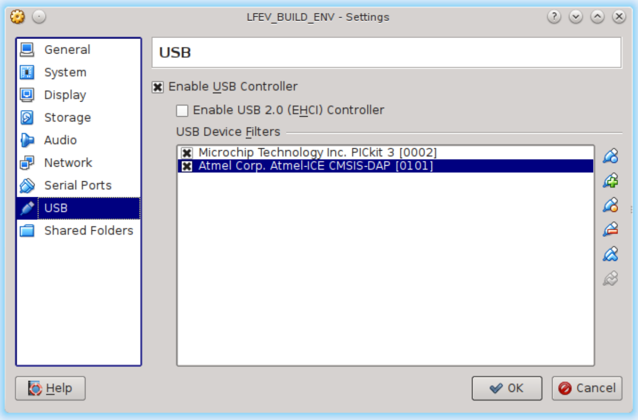
APP_OBJECTS = {all other source files} foo.o

A library file named baz/bar.c would be added as follows

LIB_OBJECTS = {all other source files} baz/bar.o

Programming the Target Device

Ensure the Atmel ICE device is attached to your computer. If you are using a virtual machine, ensure a filter rule is present so the device can be recognized by the guest machine.


LFEV_BUILD_ENV - Settings

General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders

USB

[✓] Enable USB Controller

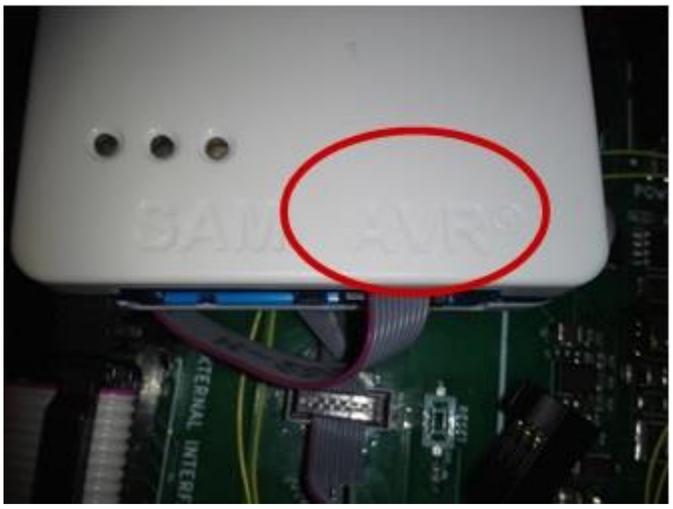
[ ] Enable USB 2.0 (EHCI) Controller

USB Device Filters

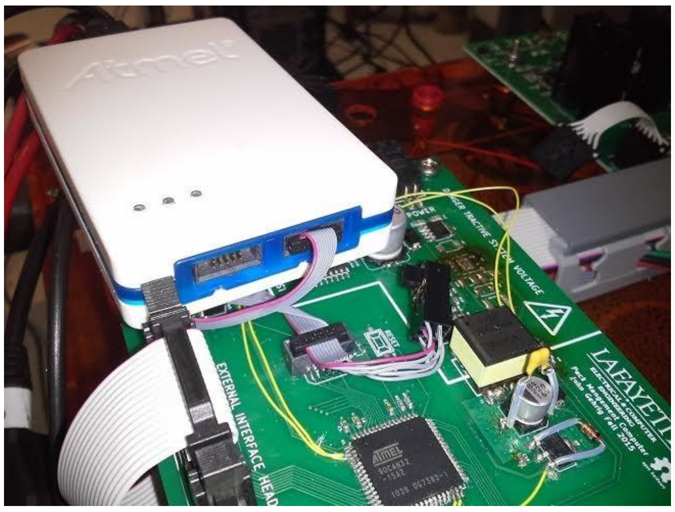
[✓] Microchip Technology Inc. PICkit 3 [0002]
[✓] Atmel Corp. Atmel-ICE CMSIS-DAP [0101]

Help
OK
Cancel

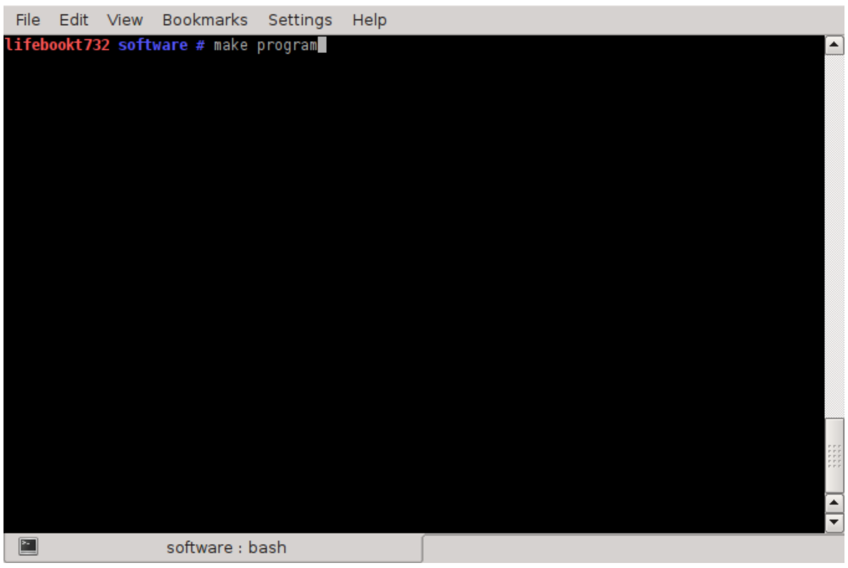
Place the JTAG connector in the AVR receptacle on the Atmel ICE



Attach the provided cable to the JTAG port of the PACMAN target board.



Open a terminal and type the following command (`sudo make program`):



```
lifebookt732 software # make program
```

NOTE: You will have to run this command as root!

Here is successful programming output:

```
lifebookt732 software # make program

avr-size -C build/pacman.hex

AVR Memory Usage

----------------

Device: Unknown

Program:       0 bytes

(.text + .data + .bootloader)


Data:          0 bytes

(.data + .bss + .noinit)


avrdude -F -V -c atmelice -p c32 -U flash:w:build/pacman.hex

avrdude: usbdev_open(): WARNING: failed to set configuration 1: Device or resource busy

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.07s

avrdude: Device signature = 0x1e9581

avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed

         To disable this feature, specify the -D option.

avrdude: erasing chip

avrdude: reading input file "build/pacman.hex"

avrdude: input file build/pacman.hex auto detected as Intel Hex

avrdude: writing flash (11636 bytes):

Writing | ################################################## | 100% 1.13s

avrdude: 11636 bytes of flash written

avrdude: safemode: Fuses OK (E:FF, H:99, L:CF)

avrdude done.  Thank you.
```

Note: Verify that the second to last line describing the Fuses OK indicates that `L:CF`. If the `make fuses` command was run successfully, then this should be the case. If not, run the `make fuses` command.