Программное обеспечение M50.
Версия с файловой системой в RAM.
Инструкция по установке и обновлению ПО.

0) Описание.

M50_release – дистрибутив **Linux Ffom Scratch**, созданный в компании «Метротек» со скриптами компиляции, установки и обновления основанный на **Buildroot**, работающий полностью из RAM. Основаная задача дистрибутива — ntp сервер (GLONASS/GPS)

Buildroot – набор automake/autoconf скриптов скачивающий, компилирующий и создающий образ файловой системы. Классика Linux Embedded.

Минимальные системные требования

- RAM 128Мбайт
- MMC 128Мбайт
- NAND 64Мбайт

1) Содержание mmc.

Раздел C: - fat32, 150mb, не обновляется, для загрузки из mmc.
mlo
u-boot.img
uImage
u-boot-nand.img
initrd.bin
rdallinst.scr
rdinst.scr
uimageinst.scr
unandinst.scr

Раздел D: -ext3, 1gb, конфиги, логи
Конфиги. Логи.

Раздел E: -fat, 150mb, для загрузки обновлений по ftp
Изначально пуст.

Раздел E: -fat, 40mb, резервный, для контрольной суммы раздела E:
Пуст.

2) Содержание nand.

/dev/mtd	-	весь nand
/dev/mtd1	-	раздел mlo, u-boot
/dev/mtd2	-	раздел с переменными u-bot
/dev/mtd3	-	раздел с ядром uImage
/dev/mtd4	-	раздел с образом рам диска initrd.bin

Физические адреса каждого раздела можно посмотреть по комманде
#: dmesg | grep nand
В результате мы увидим собственно адреса, и так же строку
Creating 5 MTD partitions on omap2-nand.0
Что может указывать на то, что разделы nand создаются динамически — ядром и u-boot, в соответствии с конфигурацией omap2.

3) Содержание коренвого каталога сборки.
M50_Build - содержит ядро, убут и buildroot. Создавать локальную копию этого каталога нужно крайне редко.

M50_Src - В этом каталоге находится все необходимое для сборки флешки, и загрузки обновлений на устройство.

В этом каталоге находится подкаталог files_patch — срез файловой системы изменяемый или создаваемый нами. Тут можно создавать, изменять файлы, папки. И это отобразится в образе рам диска, который копируется на mmc в момент ее разметки (см. раздел 4).

4.0) Создание локальной копии - создаем локальную svn копию каталога M50_Src.
В данном каталоге находится скрипт auto_inst.sh.

4.1) auto_inst.sh — Основной скрипт. Описание работы и опций командной строки.

Вставляем флешку в кард-ридер.
ВАЖНО. Если на флешке есть разделы - то монтируем их.
После монтирования убеждаемся, что все смонтировнные каталоги закрыты, включая консоль.
Если разделов нет, то оставляем как есть.

В простейшем случае, что бы создать рам диск и флешку, запускаем скрипт без аргументов
sh# auto_inst.sh
На вопрос
sh# ??? Create RamDisk (y/n): - отвечаем y. Можно ответить n, тогда скрипт попытается взять готовый рамдиск из папки ./out
Вводим пароль sudo
На дальнейшие подсказки жмем Enter
На вопрос
??? Create/Update MMC (y/n): отвечаем y. Можно ответить n, тогда после создания в папке ./out рамдиска флеш карта создаваться не будет.

Далле, в целях определения имени устройства карты mmc, в папке /dev/ скприм выдаст последние 20 строк команды dmesg. Это позволяет сделать предположение имени устройства. Можно так же воспользоваться утилитами gnome-disks и gparted. Пусть например это /dev/sdc
На вопрос
!!! ------- Enter DRIVE ---------- :
отвечаем- /dev/sdc
На вопрос
!!! ------- Enter DRIVE_base ---------- :
так же отвечаем- /dev/sdc
После этого скрпит выдаст разметку указанного диска. По разметке мы видим, наш ли это диск?
На вопрос
!!! Do you wand to RECREATE this disk (y/n):
отвечаем y если мы узнаем нашу флешку :) (Что бы не повредить другой диск).

Ждем.
На все дальнейшие вопросы жмем Enter.

Пусть теперь мы хотим пересобрать ядро, и положить его на флешку.

Запускаем ./auto_inst.sh kernel
На вопрос
??? Create RamDisk (y/n): отвечаем n
На вопрос
??? Create/Update MMC (y/n): отвечаем y
Вводим имена устройств.
На вопрос
!!! Do you wand to RECREATE this disk (y/n): отвечаем y
На дальнейшие вопросы жмем Enter.

Вставляем карту в устройство. Загружаемся с карты - работает.

Итак, загрузились из mmc.

Предположим, что мы создали svn ветку обновлений, и теперь хотим их установить.
На M50 установлен ftp сервер (IP - 192.168.2.106)

Предположим, так же, что на nand уже установлена нужная весия mlo и u-boot.
Установим теперь систему на nand.

Загрузим рамдиск и ядро на M50.

sh# auto_inst.sh updates kernel fs
На вопрос
??? Create RamDisk (y/n): отвечаем что хотим
На вопрос
??? Create/Update MMC (y/n): отвечаем y

Далее, у нас на M50 root без пароля, поэтому жмем enter.
Загружается рамдиск. (При этом иногда ftp сервер на M50 отбивает
соединение - нужно перезапустить команду).
Еще раз жмем enter - загружается ядро.

Теперь заходим в консоль M50.
Запускаем
fs_install
Запускаем
kernel_install

Перезагружаемся из nand - работает.

В консоли M50 смотрим результат вычисления md5 сумм.

cat /etc/Md5check
В данный момент мы получили установленные md5 суммы для разделов nand - mtd1 mtd2 mtd3 mtd4

4.2) Примеры

Пример1. (mmc должна быть вставлена)
Пусть надо: применить *изменения* в files_patch, создать загрузочную mmc.

#: ./auto_inst.sh

Пример2. (mmc должна быть вставлена)
Пусть надо: сконфигурировать-собрать buildroot, сконфигурировать-собрать ядро, применить обновления из filesPatch, создать загрузочную флешку.

#: ./auto_inst.sh buildroot kernel

Пример3.
(M50 должен быть способным принять файлы по ftp, на раздел mmc E. Ip: 192.168.2.106)
Пусть надо: применить *обновления* в files_patchU, сконфигурировать-собрать ядро, собрать u-boot_nand.igm, собрать скрипты u-boot для установки обновлений из u-boot, загрузить это все на M50 в раздел mmc E.

#: ./auto_inst.sh updates kernel ubootn uscr
теперь только ядро
#: ./auto_inst.sh kernel updates
теперь только uboot
#: ./auto_inst.sh updates ubootn

Параметры можно указывать в произвольном порядке и комбинациях.
Параметр updates означает, что работаем с веткой Updates, и вместо создания загрузочной флешки, загружаем файлы на M50.

4.3) Сводка опций.
Все опции:

buildroot — см пример 1.

ubootm — см пример 3 (но это uboot для mmc)

ubootn — см пример 3.

uscr — см пример 3.

kernel — см пример 2.

updates — см пример 3.

Примечание. При наличии в M50 mmc карты дистрибутив все логи и конфиги сохраняет на mmc. В противном случае все сохраняется в RAM и будет уничтожено при перезагрузке.

5) Установка

5.1) Первоначальная установка всего, из u-boot_mmc. **Прежде, достаточно выполнить**

./auto_inst.sh
В коммандной строке u-boot набираем:
mmc rescan
fatls mmc 0
fatload mmc 0 0x81000000 rdallinst.scr
source 0x81000000

(При этом на mmc должно находится установленных md5 сумм.)

5.2) Установка обновлений всего из u-boot_mmc. Сработает только после выполнения

./auto_inst.sh updates kernel ubootn uscr fs
т.е. после заливки всего на M50 mmc E:
В коммандной строке u-boot набираем:
mmc rescan
fatls mmc 0:3
fatload mmc 0:3 0x81000000 rdallinst.scr
source 0x81000000

5.3) Установка обновлений файловой системы средствами Linux. Прежде, достаточно выполнить

./auto_inst.sh updates fs
В коммандной строке M50 набираем:
#: fs_install

6) Конфиги, логи.

5.1) Конфиги

/mnt/D/etc/resolv.conf
/mnt/D/etc/ntp.conf
/mnt/D/etc/vsftpd.conf
/mnt/D/etc/network/interfaces
/mnt/D/usr/bin/Demetro/boot.cfg
/mnt/D/usr/bin/Demetro/enable.cfg

5.2) Логи. Demetro был пересобран и измененными путями к логам в скобках указано новый путь, при отсутствии mmc путь остается тем же но пишется в все в ram.

/ftpdir/stat.txt Файл статистики
mmc
(/mnt/D/usr/bin/Demetro/stat.txt)

/ramcache/demetro/errorlog.log файл логов критических ошибок
mmc
(/mnt/D/usr/bin/Demetro/errorlog.log)

/usr/bin/Demetro/nmea_emul_flag.txt флаг включение/выключенияэмулятора
mmc
(/mnt/D/usr/bin/Demetro/nmea_emul_flag.txt)

/usr/bin/Demetro/boot.cfg
mmc
(/usr/bin/Demetro/boot.cfg)

/ramcache/demetro/voltage.log файл значений напряжений.
ram
(изначально /usr/bin/Demetro/voltage.log)

/ramcache/demetro/voltage_temp.log временный файл для правильного формирования
(изначально /usr/bin/Demetro/voltage.log)

/ramcache/demetro/voltage.log
ram
(изначально /usr/bin/Demetro/*voltage_temp.log*)

/ramcache/demetro/voltage2.log то же, что и выше
ram
(изначально /usr/bin/Demetro/*voltage2.log*)

/ramcache/demetro/place.txt файл координат
ram
(так и остался)

/ramcache/demetro/nmeabuf буффер для разбора потока NMEA
ram
(так и остался)

/ramcache/demetro/nmea файл с количеством спутников и высотой (нету)
ram
(так и не появился, даже после afterupdate.sh)

Создание и администрирование svn репозитория.

SSH доступ на сервер.
ssh -X sedov1@192.168.2.172
123456

Создание репозитория
svnadmin create /usr/local/M50_REPO

Дать права на запись всем.
Дать доступ для записи пользователю Anonymous
В файле /usr/local/M50_REPO/conf/svnserve.conf
добавить под строкой
anon-access = read
строку
anon-access = write

Перезапустить сервер.
killall svnserve
svnserve -d -r /usr/local/M50_REPO

На локалке юзаем Eclipse и subclipse плагин, скачанный и установленный просто как плагин. Далее все операции делаются из Eclipse.

Добавление проэкта в вообще пустой репоизторий.

Из эклипса создаем локальную копию репоизтория (SVN checkout). Копируем туда каталог M50_Proj и комитим. Все.

Создание backup-ов по cron.

В каталоге sftp://192.168.2.172/usr/local/ есть подкаталог svn_backup

В нем подкаталоги
d - ежедневные бэкапы.
w - еженедельные бэкапы.
скрипт svn_backup.sh

Задания cron запускаются из /sedov1/crontab.txt (переназначен)
Тут же есть script.sh который в файл cron_out.txt печатает дату и время создания backup-ов.

Структура репозитория M50_REPO.

На данный момент в M50_REPO находится два проэкта:

M50_Proj и M50_PTP

Первым создавался M50_Proj и логичнее было бы его назвать M50_NTP. А проэкт M50_PTP создавался как ветка от M50_Proj с замененным ядром. (Ветка создавалась из Eclipse).

Поэтому рассмотрим только проэкт M50_PTP.

Все изменения вносятся в каталог turnk. При желании можно создавать ветки в каталоге branches.

Содоержимое trunk.

M50_Base - содержит все что касается сборки базовой ОС. Т.е. ядро, u-boot_mmc, u-boot_nand, и конфиг для buildroot. Сам buildroot находится на sftp://192.168.2.172/usr/local/archive/M50distr/M50_Proj/buildroot_clean.tar.gz. О его сборке напишем отдельно. При этом его сборка требуется крайне редко, потому что он уже собран, и результат сборки помещается сборочным скриптом в каталог M50_Src/out/rootfs.tar (см. ниже), скрипт сборки будет его использовать.

M50_Src - основной рабочий каталог.

- Eclipse - исходные коды для проэктов Eclipse. См. создание локальной копии.

- QT - исходные коды для прэктов QT.
- SCRIPTS - вспомогательные скрпиты для основного сборочного скрпита M50_Src/auto_inst.sh
- doc - документация.
- files_patch - срез файловой системы в которую вносятся все изменения.
- out - все результирующе файлы необходимые для работы операционной системы.
    u-bootы
    u-boot скрипты
    rootfs.tar - результат сборки buildroot
    initrd.bin - образ файловой системы. В этот образ сборочный скрпит делает
        rsync - из files_patch (если его не существует то он создается)
    uImage - образ ядра.

auto_inst.sh - основной сборочный скрипт.
chmod.sh - скрипт расстановки прав доступа. Дело в том, что svn снимает права на выполнение по умолчанию. Его можно настроить на сохранение прав, но проще было написать скрипт расстановки прав.

Создание локальной копии.

На виртуальной машине M50_Mint предустановленный Eclipse с плагином Subclipse.

Предлагаемая структура локальных каталогов.

В удобном месте создаем каталог Workspaces.
Для каждого проэкта верхнего уровня, например M50_ Proj и M50_PTP создаем отдельный workspace. Например
~/Workspaces/M50_Proj
~/Workspaces/M50_PTP

Создание локальной копии.

Подключить репозиторий в Subclipse. Для этого переключить перспективу SVN Rpository Exploring, создать новый Location - адрес:
    svn://192.168.2.172/M50_REPO/M50_PTP

Переключится в перспективу C/C++
Меню/SVN/Checkout Projects from SVN
Выбрать нужный репозиторий
Выбрать из репозитория нужный каталог например /turnk/M50_Src
Выбрать Check out as project in the workspace, имя оставляем как есть, пусть тооже как есть. В конктерном случае Eclipse подскажет путь ~/Workspaces/M50_PTP, его оставляем. Т.е мы не учитываем наличие turnk и branches. Жмем Finish. Локальная копия создана.

Создание проэкта Eclipse.

В каталоге ~/Workspaces/M50_PTP/ осздаем подкаталог Build. В нем создаем обычный C Hello World проэкт. Например Demetro. Т.е получаем катлог ~/Workspaces/M50_PTP/Demetro.
Удаляем в нем весь каталог ./src, и в место него создаем тут ссылку на ~/Workspaces/M50_PTP/M50_Src/Eclipse/Demetro/src/ т.е на локальную копию svn. Такой поход был выбран как более удобный, по сравнению с созданием SVN/C проэкта. И позволяет не смешивать исходный код (который хранитсья в svn), с созданием и управлением C проэктами Eclipse. Что бы еще больше разделить исходные коды и C проэкты Eclipse, можно кталог Build создать в другом workpsce. Это дело вкуса.

Создание ветки.

Для примера создадим ветку проэкта Demetro.
Для этого откроем перспективу SVN Rpository Exploring, Найдем Demetro, нажмем на нем правой кнопкой мыши, выберем Branch/Tag, укажем путь - куда сделать ветку (например в branches/Demetro_v2) и нажмем OK. При этом каталог в репозитории создастся автоматически если будет установлена галка «Create any intermediate folders thar are missing», она находится сразу под полем Copy to URL.

Далее все как обычно. Создаем локальную копию. Отдельный C Hello World проэкт. Удаляем в нем каталог src, и создаем на его месте ссылку на каталог src из вновь созданной рабочей копии ветки Demetro.

Объединение веток.

Делать автоматическое объединение веток не безопастно. Можно потерять данные.
Поэтому, в случае надобности изменения из branch и trunk луше вносить в ручную после тестирования в branch.

C программа генерации PTP пакетов.

Программа находится по адресу.
svn://192.168.2.172/M50_REPO/M50_PTP/M50_Src/Eclipse/PTP_Packgen/PTP_Client
Т.з. находится там же.
В программе практически каждая строчка имеет комметраий.

Программа загрузки и установки софта на сервер M50.

Программа расположена по адресу:
svn://192.168.2.172/M50_REPO/M50_PTP/M50_Src/QT
Программа заливает на M50 ftp сервер образ файловой системы. И далее в автоматическом режиме по telnet выполняет установку.
Программа не комментирована. Я не знаю как ее комментировать. Используется модификация функционала виждетов и многопоточность - что бы во время установки файловой системы, программа отвечала на действия пользователя.