www.kb-agava.ru
АГАВА
Настоящая газовая автоматика

ООО КБ «АГАВА»
620026, г. Екатеринбург, ул. Бажова, 174, 3 эт.
т/ф 343/262-92-76 (78,87)
agava@kb-agava.ru; http://www.kb-agava.ru

ПРОМЫШЛЕННЫЙ КОНТРОЛЛЕР

АГАВА ПК-30

РУКОВОДСТВО ПРОГРАММИСТА

АГСФ.421445.004 РП

Редакция 1.3

Екатеринбург

2018

Оглавление

1 Введение....4
1.1 Используемые термины и сокращения:....4
2 Назначение....5
3 Настройка прибора....6
3.1 Установка времени и даты....6
3.2 Доступ к файлам контроллера....7
3.2.1 Настройка сетевого доступа....7
4 Аппаратные ресурсы контроллера....8
4.1 Звуковой извещатель....8
4.2 Клавиатура....8
4.3 Программный переключатель SA1.2....8
4.4 Светодиоды....8
4.5 EEPROM....9
4.6 RTC....9
4.7 NAND flash....9
4.8 Последовательные порты....9
4.9 Ethernet....10
4.10 USB HOST....10
4.11 USB OTG....10
4.12 Фреймбуфер и виртуальная консоль....11
4.13 Датчик сети....11
5 Разработка пользовательского прикладного программного обеспечения с использованием компиляторов C/C++....12
5.1 Разработка простого приложения с использованием системы сборки make в среде Linux....12
5.1.1 Создание Makefile....12
5.1.2 Создание текста программы....12
5.1.3 Сборка приложения....12
5.2 Разработка простого приложения с использованием средств Visual Studio и VisualGDB в среде Windows....13
5.2.1 Создание проекта VisualGDB....13
5.2.2 Наполнение проекта кодом....22
5.2.3 Сборка приложения....24
5.2.4 Отладка приложения в среде Visual Studio....24
6 Обновление базовых программных компонентов Контроллера....27
6.1 Обновление компонентов загрузчика....27
6.2 Обновление компонентов ОС Linux....28
6.3 Обновление корневой файловой системы....28
6.4 Обновление прикладного (пользовательского) программного обеспечения....29

1 Введение

Руководство по эксплуатации содержит сведения, необходимые для обеспечения правильной эксплуатации и полного использования технических возможностей промышленного контроллера АГАВА ПК-30, далее по тексту ПРИБОР или КОНТРОЛЛЕР.

1.1 Используемые термины и сокращения:

SSH – Secure Shell

ПЛК – программируемый логический контроллер (промышленный контроллер);

ОС – операционная система;

ПО – программное обеспечение;

ОЗУ – оперативное запоминающее устройство;

ФС – файловая система.

2 Назначение

Промышленный контроллер АГАВА ПК-30 предназначен для создания систем автоматизированного управления технологическим оборудованием в различных областях промышленности, жилищно-коммунального и сельского хозяйства.

Логика работы контроллера определяется потребителем в процессе программирования контроллера. Программирование осуществляется с помощью различных средств разработки с использованием компиляторов C/C++.

Загрузка проекта в прибор и его отладка производятся через интерфейс Ethernet.

3 Настройка прибора

На уровне операционной системы прибор имеет файловые ресурсы и системную консоль. В файлах содержится необходимая информация для работы ОС и пользовательского прикладного программного обеспечения. Консоль служит для интерактивного взаимодействия с ОС (выполнения команд ОС и т.п.).

Файловая система состоит из системной ФС и монтируемой ФС, которая доступна как на чтение, так и для записи[1], имеющая следующие точки монтажа:

- /run/media/mmcblk* для SD-карты;
- /run/media/sda* для и USB-флеш;

Системная консоль находится на последовательном порту RS-232. Параметры терминала для консоли следующие:

- Скорость (бит/с): 115200
- Биты данных: 8
- Четность: Нет
- Стоповые биты: 1
- Управление потоком: Нет

Соединение контроллера с персональным компьютером по интерфейсу RS-232 производится нуль-модемным кабелем.

При загруженной ОС, подключенной и настроенной сети доступ к системной консоли можно получить по SSH.

Права администратора для входа по SSH:

- Логин: root
- Пароль отсутствует

3.1 Установка времени и даты

Для установки времени и даты следует воспользоваться командой:

```
date MMDDhhmmYYYY
```

где

- MM – месяц (1-12);
- DD – число (1-31);
- hh – часы (0-23);

[1] SD-карта может быть заблокирована на запись при установке переключателя защиты записи в соответствующее положение.

- mm – минуты (0-59);
- YYYY – год.

Для сохранения установленного времени и даты в часы реального времени воспользуйтесь командой:

```
hwclock –w
```

При подключении контроллера к сети Ethernet и наличии выхода в Интернет, происходит синхронизация времени с серверами точного времени.

Часовой пояс устанавливается в файле /etc/profile путем задания переменной окружения TZ. Например, export TZ="STD-5" (для Екатеринбурга).

3.2 Доступ к файлам контроллера

Доступ к файлам и ресурсам контроллера при загруженной ОС (в т.ч. запущенной системы исполнения CODESYS) можно получить следующими способами:

- через системную консоль на порте RS-232;
- через системную консоль SSH-сервиса;
- через sftp-сервер;

Для доступа к файлам контроллера через sftp-сервер следует пользоваться Unix-совместимым sftp-клиентом. Под ОС Windows это может быть, например, WinSCP, Total Commander и т.п.

Для того чтобы иметь доступ к SD-карте через USB-интерфейс, необходимо загрузить модуль *g_mass_storage.ko.* При подключении компьютера к разъему USB-OTG прибора, на компьютере появится съемный диск с содержимым SD-карты. Также загрузка данного модуля необходима для подключения к разъему USB-OTG других устройств (USB-флеш и т.п.).

```
modprobe g_mass_storage file=/dev/mmcblk0
```

3.2.1 Настройка сетевого доступа

Для использования сетевых ресурсов необходимо настроить подключение к сети Ethernet. По-умолчанию прибор настроен на получение сетевых настроек по DHCP. Просмотреть IP-адрес и другую сетевую конфигурацию можно из консоли, набрав команду:

```
ifconfig
```

Задать статический IP-адрес можно в файле /etc/systemd/network/10-eth.network, например:

```
[Network]
DHCP=no
Address=192.168.10.32/24
Gateway=192.168.10.10
```

4 Аппаратные ресурсы контроллера

4.1 Звуковой извещатель

Драйвер регистрируется как /sys/devices/beeper. Для подачи звукового сигнала введите в терминале команду:

```
echo -en "\07" > /dev/tty5
```

4.2 Клавиатура.

Драйвер регистрируется как /sys/devices/matrix_keypad@0. Для считывания нажимаемых на клавиатуре клавиш введите в терминале команду:

```
evtest /dev/input/by-path/platform-matrix_keypad@0-event
```

Альтернативный способ. Драйвер клавиатуры подхватывается терминалом /dev/tty0 и /dev/tty1. Выполнив команду

```
cat /dev/tty0
```

нажатые клавиши на клавиатуре должны немедленно отображаться на LCD-дисплее, а также, после нажатия клавиши "ВВОД" и на терминале.

4.3 Программный переключатель SA1.2.

Драйвер регистрируется как /sys/devices/gpio_buttons@0. В терминале подать команду:

```
evtest /dev/input/by-path/platform-gpio_buttons@0-event.
```

Включая и выключая переключатель SA1.2 убедиться, что регистрируются соответствующие события.

4.4 Светодиоды.

Драйвер регистрируется как /sys/*class*/leds. В каталоге присутствуют устройства для управления соответствующими светодиодами:

```
agava:programm:green - "Программа";
agava:sd:green - "Диск" (по-умолчанию настроен на событие mmc0 - управление от драйвера SD-карты);
agava:usb:green - "USB" (по-умолчанию настроен на событие usb-host - управление от драйвера хоста USB);
agava:work:green - "Работа";
agava:fault:red - "Авария";
agava:backlight:red - подсветка дисплея красный светодиод;
agava:backlight:green - подсветка дисплея зеленый светодиод;
agava:backlight:blue - подсветка дисплея голубой светодиод;
```

Для включения, например, светодиода "Программа", из терминала нужно подать команду

```
 echo 1 > /sys/class/leds/agava:programm:green/brightness.
```

Для выключения:

```
echo 0 > /sys/class/leds/agava:programm:green/brightness.
```

4.5 EEPROM.

Регистрируется как устройство /sys/bus/i2c/devices/0-0050/eeprom.

Для проверки записи необходимо выполнить команды:

```
cd /sys/bus/i2c/devices/0-0050
echo "TEST" > eeprom
head -c 300  eeprom | hexdump -C
```

Убедиться, что записалась строка "TEST".

После проведения операций с EEPROM необходимо установить защиту от записи.

4.6 RTC.

Установить системное время командой:

```
date -s "2016-06-28 17:47:00"
```

Записать время в RTC командой:

```
hwclock -w
```

Выключить контроллер. Через 1 минуту включить контроллер, командой date убедиться, что установлено правильное системное время.

4.7 NAND flash.

Для проверки функционирования NAND flash нужно выполнить команду:

```
nandtest -p 1 /dev/mtd9
```

Убедиться, что тест прошел без ошибок (Finished pass 1 successfully).

4.8 Последовательные порты.

Регистрируются как устройства /dev/ttyS0 - /dev/ttyS4.

Устройство /dev/ttyS0 – порт RS-232

Устройство /dev/ttyS1 - порт RS-485-1

Устройство /dev/ttyS2 - порт RS-485-2

Устройство /dev/ttyS3 - порт RS-485-3

Устройство /dev/ttyS4 - порт RS-485-4

Из терминала системной консоли можно выполнить команду установки скорости порта 1200 бит/сек:

```
stty -F /dev/ttyS1 1200
```

или 230400 бит/сек:

```
stty -F /dev/ttyS1 230400
```

Для включения и проверки аппаратного контроля управления потоком RTS/CTS подайте команду:

```
stty crtscts -F /dev/ttyS0
```

Выключение аппаратного контроля:

```
stty -crtscts -F /dev/ttyS0
```

Порт RS232 не использует сигналов DTR/DSR, поэтому в нуль-модемном кабеле они должны быть соединены между собой.

4.9 Ethernet.

Работа порта Ethernet поддерживается операционной системой, пользователю достаточно использовать интерфейс eth0 или другой, если была произведена смена имени интерфейса.

4.10 USB HOST

Порт USB HOST проверяется подключением внешней USB-флеш. Убедиться, что флеш определяется контроллером.

Перед отсоединением флеш размонтировать:

umount /run/media/sda1

4.11 USB OTG

Порт USB OTG функционирует при загруженном модуле гаджета g_mass_storage.ko. Для загрузки модуля необходимо выполнить команду:

```
modprobe g_mass_storage file=/dev/mmcblk0
```

После чего подключить контроллер кабелем miniUSB/USB к компьютеру, убедиться, что на компьютере появится новый диск - SD-карта, установленная в контроллере.

Для проверки режима USB host необходимо подключить к контроллеру кабель OTG, в который установить USB-флеш. Убедиться, что флеш определяется контроллером.

4.12 Фреймбуфер и виртуальная консоль.

Фреймбуфер регистрируется как устройство /dev/fb0. Текст на экран выводится через виртуальную консоль /dev/tty0.

Для проверки работы фреймбуфера и виртуальной консоли выполнить команду:

```
echo '12345' > /dev/tty0
```

При этом на ЖКИ-экране контроллера должен появиться текст 12345.

4.13 Датчик сети.

Драйвер датчика сети включен как модуль ядра agava_pwr_sensor.ko. В случае обнаружения отсутствия напряжения сети генерирует событие типа EV_PWR - KEY_SLEEP (142), при появлении напряжения сети - событие KEY_WAKEUP (143), если напряжение отсутствовало в течение программируемого интервала времени, генерируется событие KEY_POWER (116). Событие KEY_POWER не генерируется в случае задания значения интервала времени -1. При значении 0, событие генерируется сразу при пропадании сети, вместе с событием KEY_SLEEP.

Драйвер регистрируется как устройство /sys/devices/pwr_sensor.

События генерируются в /dev/input/by-path/platform-pwr_sensor-event.

Программируемый интервал времени задается, в порядке возрастания приоритета:

- параметром командной строки delay= при загрузке драйвера как модуля;
- параметром delay-sec в devicetree-файле am335x-agava.dts (при компиляции DTB файла) в случае компиляции встроенного в ядро драйвера, или в случае компиляции драйвера как модуля, при загрузке которого не указан параметр delay=;
- записью значения в /dev/pwrsens, которое вступает в силу с момента записи.

Для проверки генерации событий выполнить в консоли:

```
evtest /dev/input/by-path/platform-pwr_sensor-event
```

Отключая и подключая подачу напряжения на разъем X4 контроллера, можно убедиться что на консоль выводятся соответствующие сообщения и генерируются соответствующие события.

5 Разработка пользовательского прикладного программного обеспечения с использованием компиляторов C/C++

Данный раздел предназначен для специалистов, обладающих знаниями языка C/C++, а так же опытом сборки приложений с использованием системы make.

Детальное описание порядка разработки приложений на языках C/C++ приведено в соответствующей литературе. Ниже описываются действия, специфичные для контроллеров АГАВА на базе ОС Linux RT 4.4.

Для начала работы необходимо установить виртуальную машину VMWare Player с ОС UBUNTU 14.04, в которой уже установлено все необходимое программное обеспечение:

- Кросскомпилятор C/C++
- SFTP клиент
- SSH Server

5.1 Разработка простого приложения с использованием системы сборки make в среде Linux

Разработаем небольшое приложение, использующее аппаратное обеспечение контроллера АГАВА 6432.30.

Приложение будет поочередно включать и выключать все имеющиеся светодиоды, а так же светодиоды подсветки дисплея.

Исходные тексты приложения доступны в виртуальной машине по адресу: File://home/user/applications/LEDTest

5.1.1 Создание Makefile

Воспользуемся готовым основным makefile, содержащимся в файле “Makefile” в каталоге LEDTest. Описание принципов создания makefile приведено в соответствующей литературе. Основной makefile кроме прочих инструкций содержит установку имени файла выходного приложения, а так же имена файлов, содержащих исходные тексты.

Основной makefile дополняют два дополнительных: debug.mak и release.mak для отладочной и релизной версий программы соответственно. В дополнительных makefile указываются пути до кросскомпилятора, ключи компиляции и линковки.

5.1.2 Создание текста программы.

Напишем основной код приложения на языке C++, и поместим его в файл «LEDTest.cpp»

5.1.3 Сборка приложения

Для сборки приложения запустим команду make в директории с исходными текстами:

```
user@ubuntu:~/applications/LEDTest$ make
```

По умолчанию сборка будет выполнена в варианте Debug, то есть с дополнительной отладочной информацией.

При сборке приложения в консоль выводится информация о процессе сборки, ошибках и т.д.:

```
user@ubuntu:~/applications/LEDTest$ make
/home/user/ti-processor-sdk-linux-rt-am335x-evm-03.03.00.04/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnueabihf-g++ -std=c++11 -fexceptions -ggdb -ffunction-sections -O0 -DDEBUG=1 -c LEDTest.cpp -o Debug/LEDTest.o -MD -MF Debug/LEDTest.dep
/home/user/ti-processor-sdk-linux-rt-am335x-evm-03.03.00.04/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-linux-gnueabihf-g++ -o Debug/LEDTest -Wl,-gc-sections -Wl,--start-group Debug/LEDTest.o -Wl,--rpath='$ORIGIN' -Wl,--end-group
```

Для сборки релизного варианта приложения запустим make с указанием варианта сборки:

```
CONFIG=RELEASE make
```

Сборка приложения в этом варианте будет произведена аналогично предыдущему.

5.2 Разработка простого приложения с использованием средств Visual Studio и VisualGDB в среде Windows

Использование VisualGDB позволит значительно повысить удобство и скорость разработки ПО для контроллеров с ОС Linux, так как редактирование текстов и отладка ведется в среде Visual Studio.

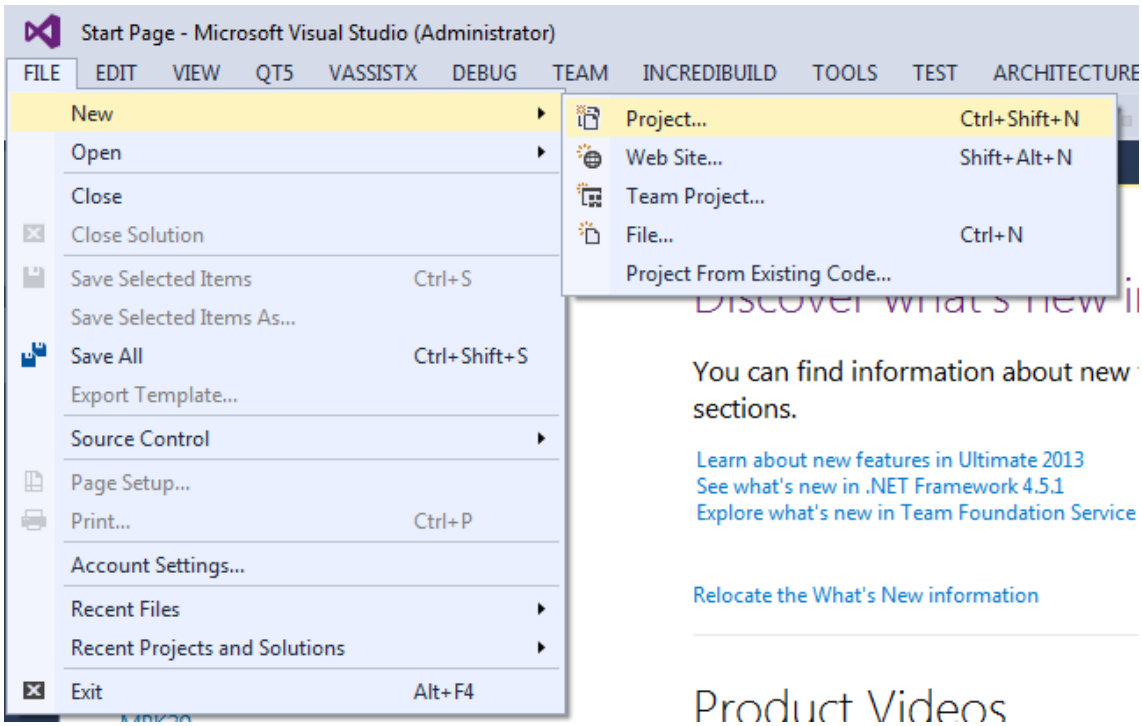
Для начала работы необходимо установить на ПК с Windows программное обеспечение дополнительно к тому, что указано в п. 0:

- VisualGDB
- Visual Studio 2013

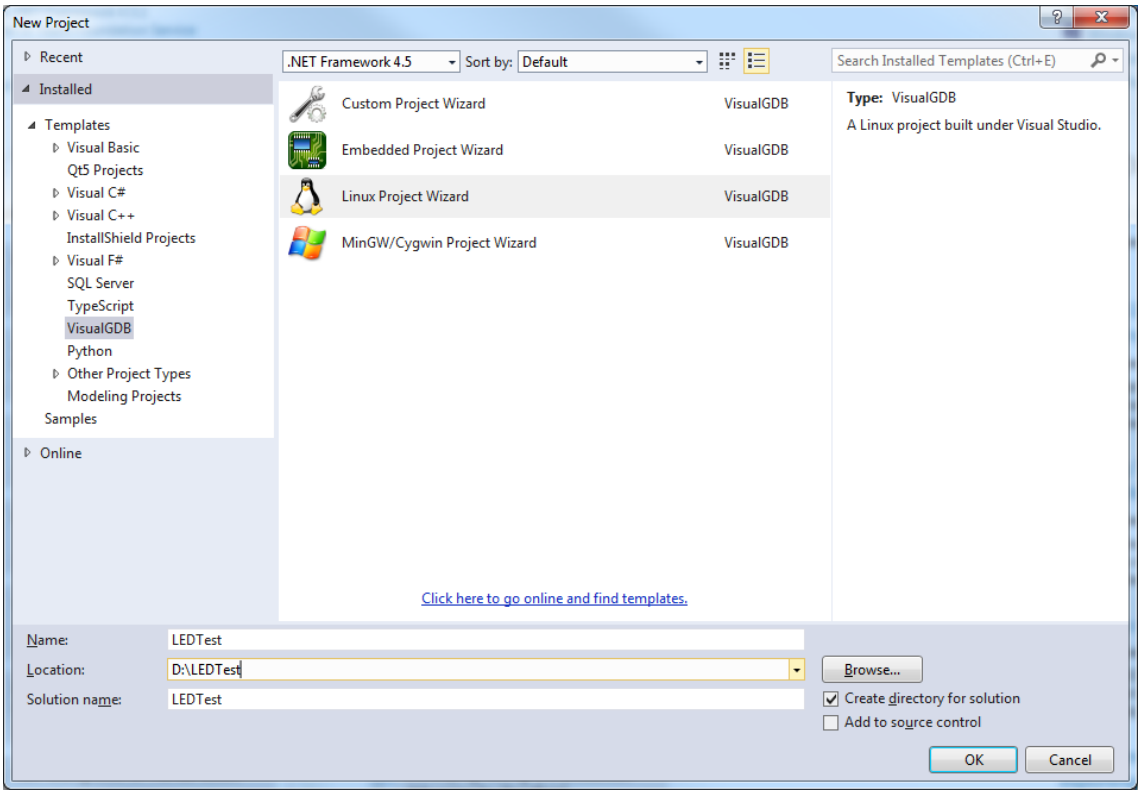
5.2.1 Создание проекта VisualGDB

После установки необходимого ПО создадим проект для VisualGDB, который будет хранить создаваемое нами приложение.

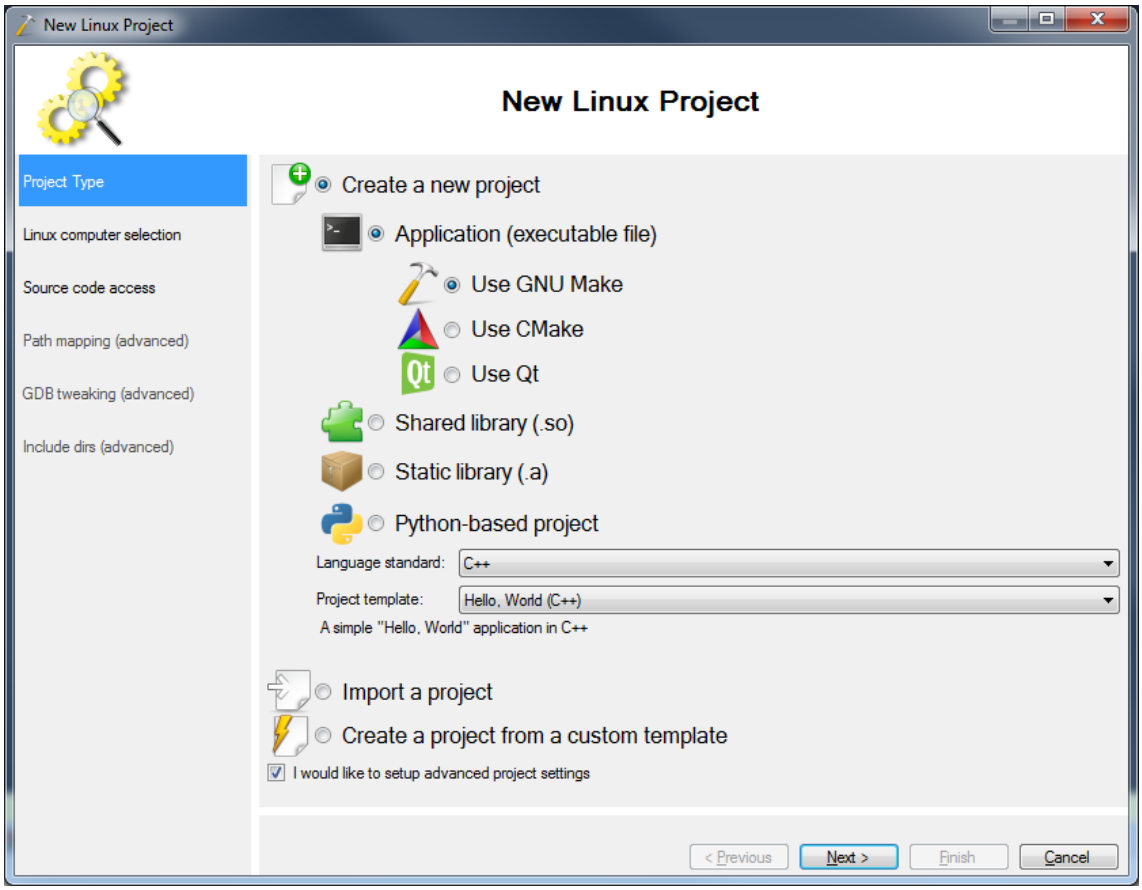
Открываем Visual Studio. И идем в меню File->New->Project



Выбираем VisualGDB -> Linux Project Wizard. Указываем папку для проекта и имя проекта.

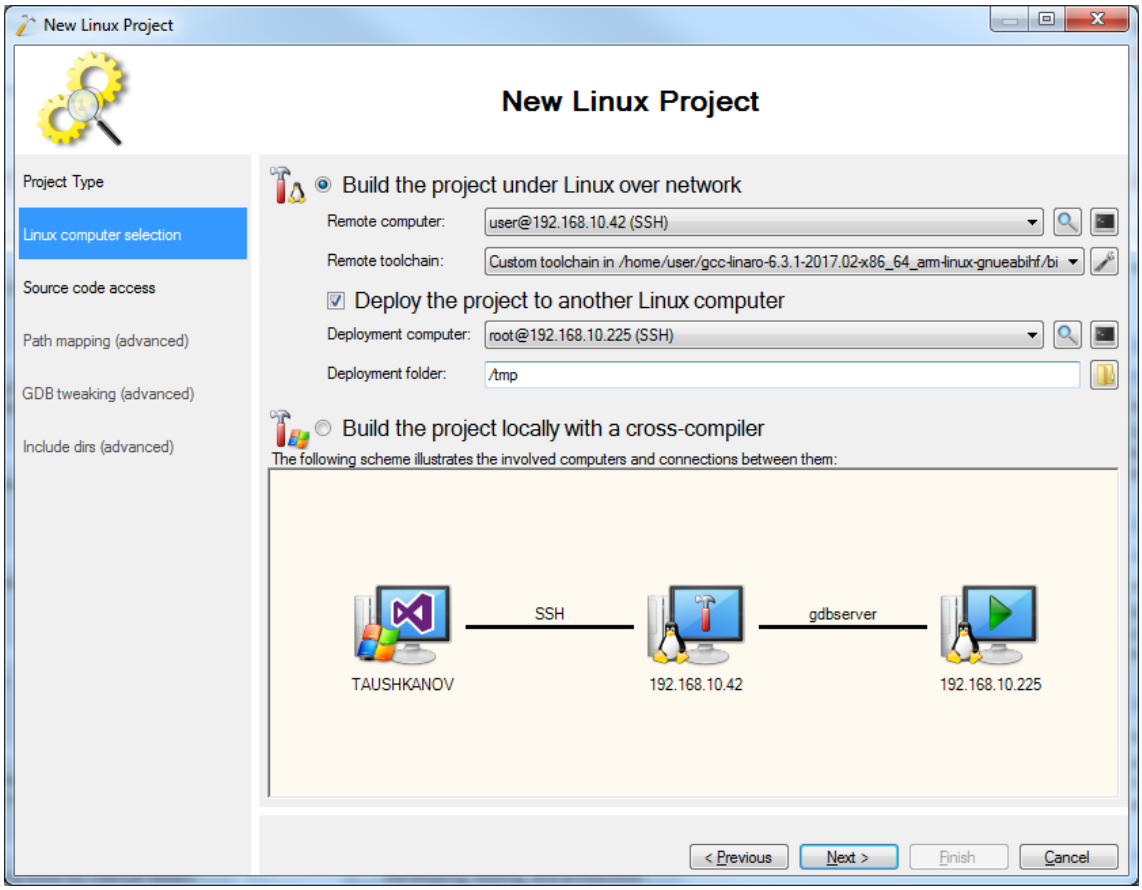


На данном этапе выбранные параметры оставляем без изменения:

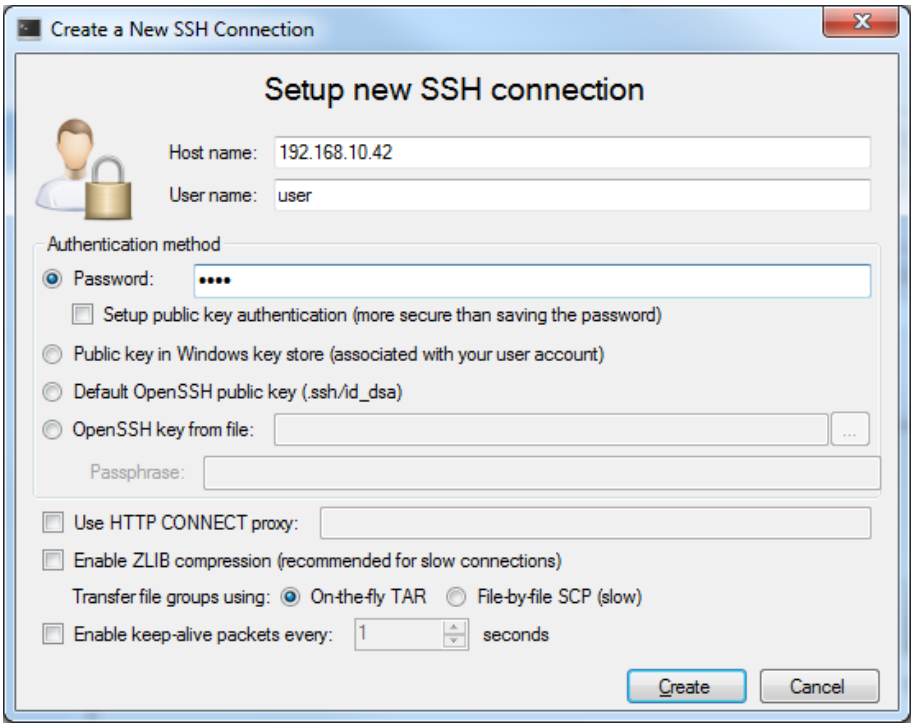


Нажимаем «Next»

Настраиваем параметры, как указано на рисунке ниже:



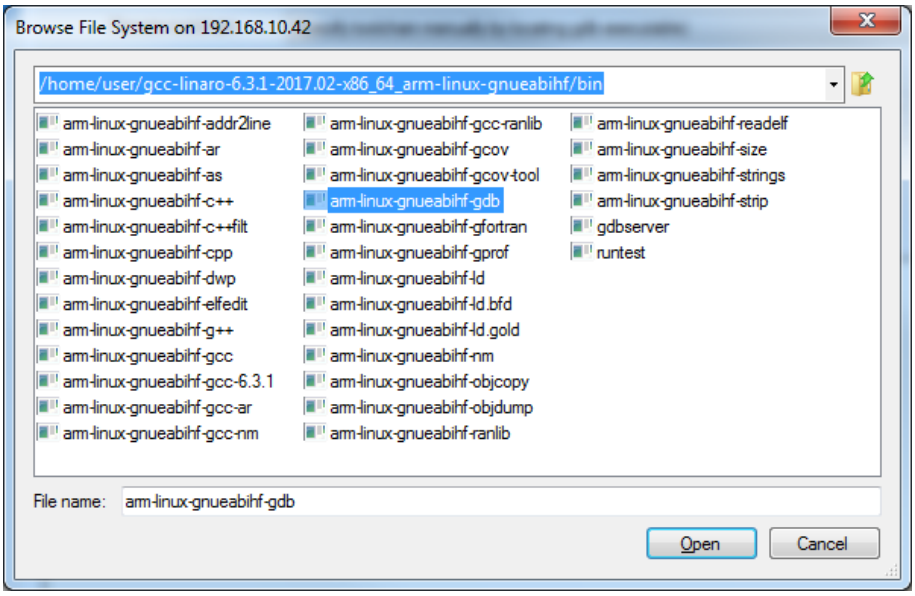
При настройке поля «Remote computer» выбираем пункт «New connection». Появится окно с параметрами соединения. Заполняем поля, как указано на рисунке ниже:



Нажимаем «Create». При первом подключении появится окно:

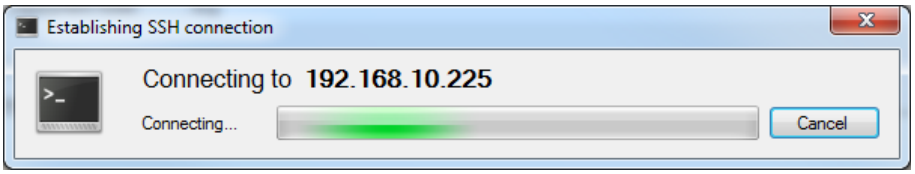
Нажимаем «Remember …» и еще раз подтверждаем свои намерения.

При заполнении поля «Remote toolchain» выбираем пункт «Specify toolchain manually by locating gdb» и указываем местонахождение gdb в файловой системе виртуальной машины:

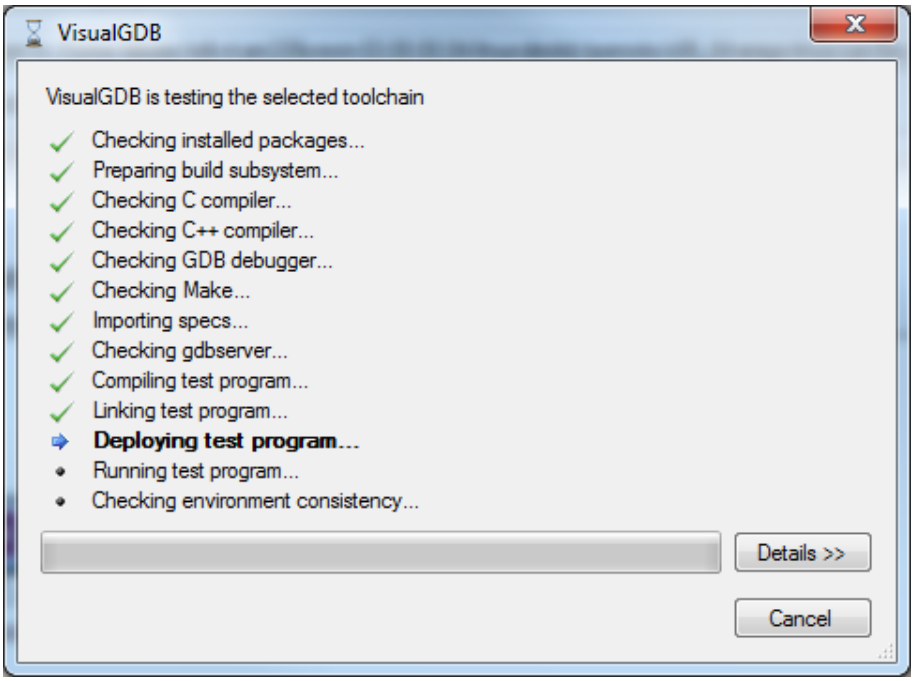


Нажимаем «Open» и возвращаемся к окну настройки соединений. Нажимаем «Next»

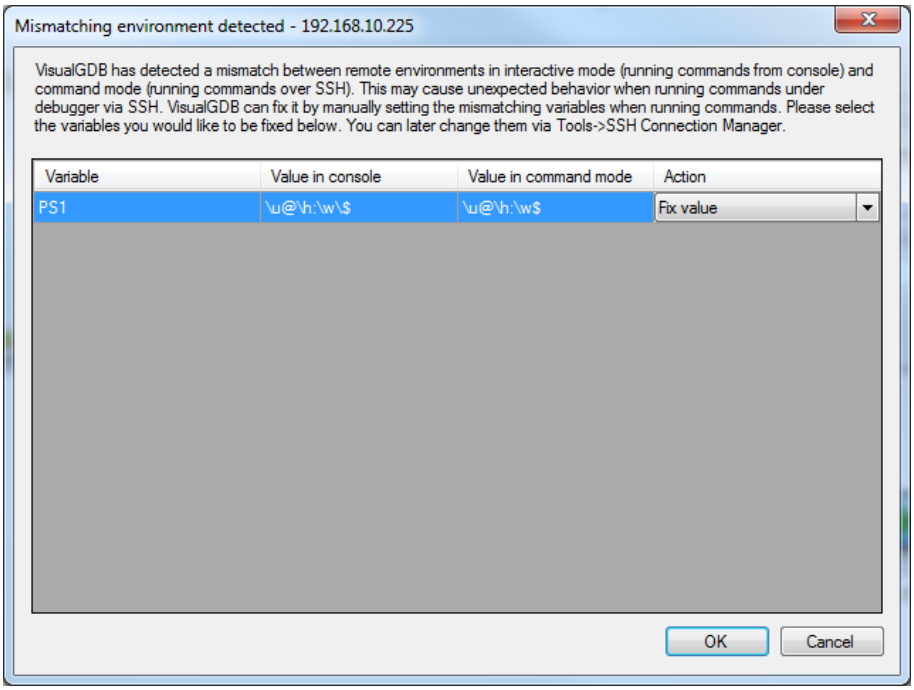
Происходит соединение с виртуальной машиной и целевым устройством. В случае неполадок будет отображено сообщение об отсутствии связи.



Далее идет проверка целевого устройства на наличие всех необходимых компонентов:

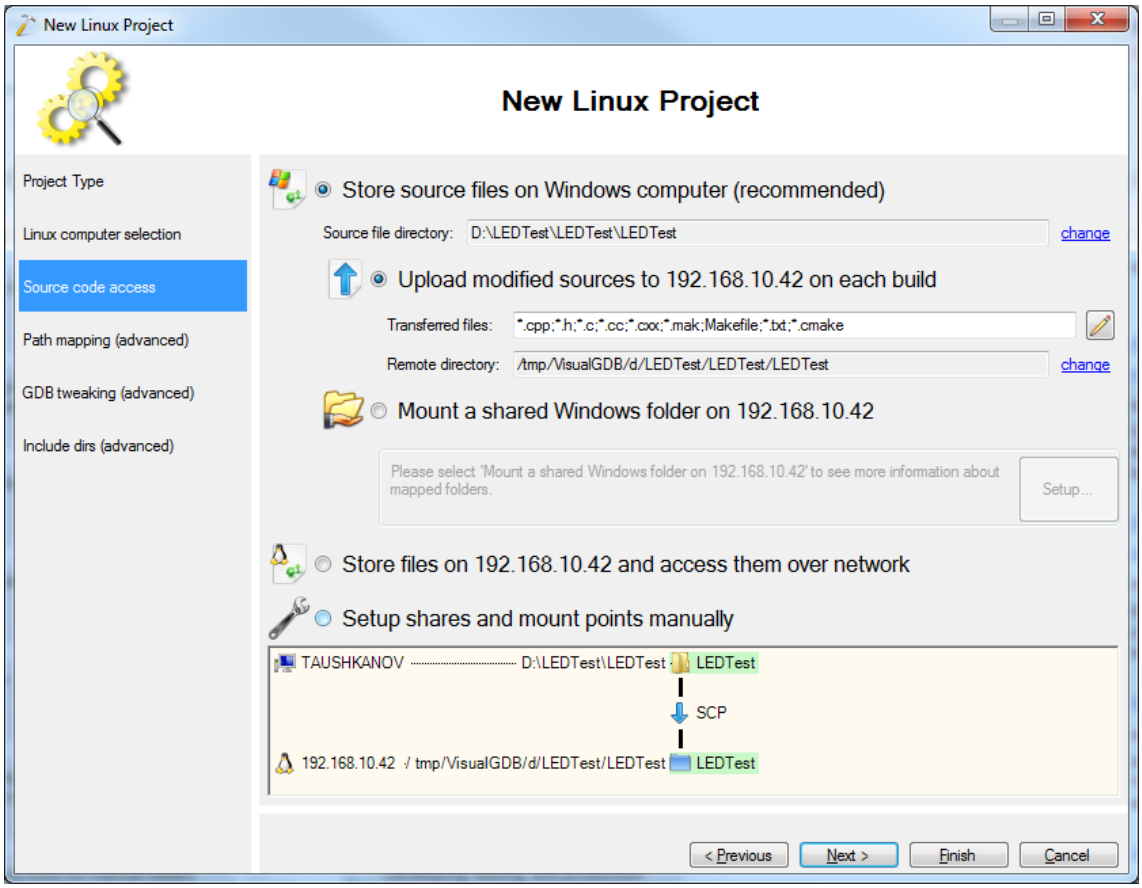


При проверке может обнаружиться несовпадение переменных окружения. В этом случае появиться окно:

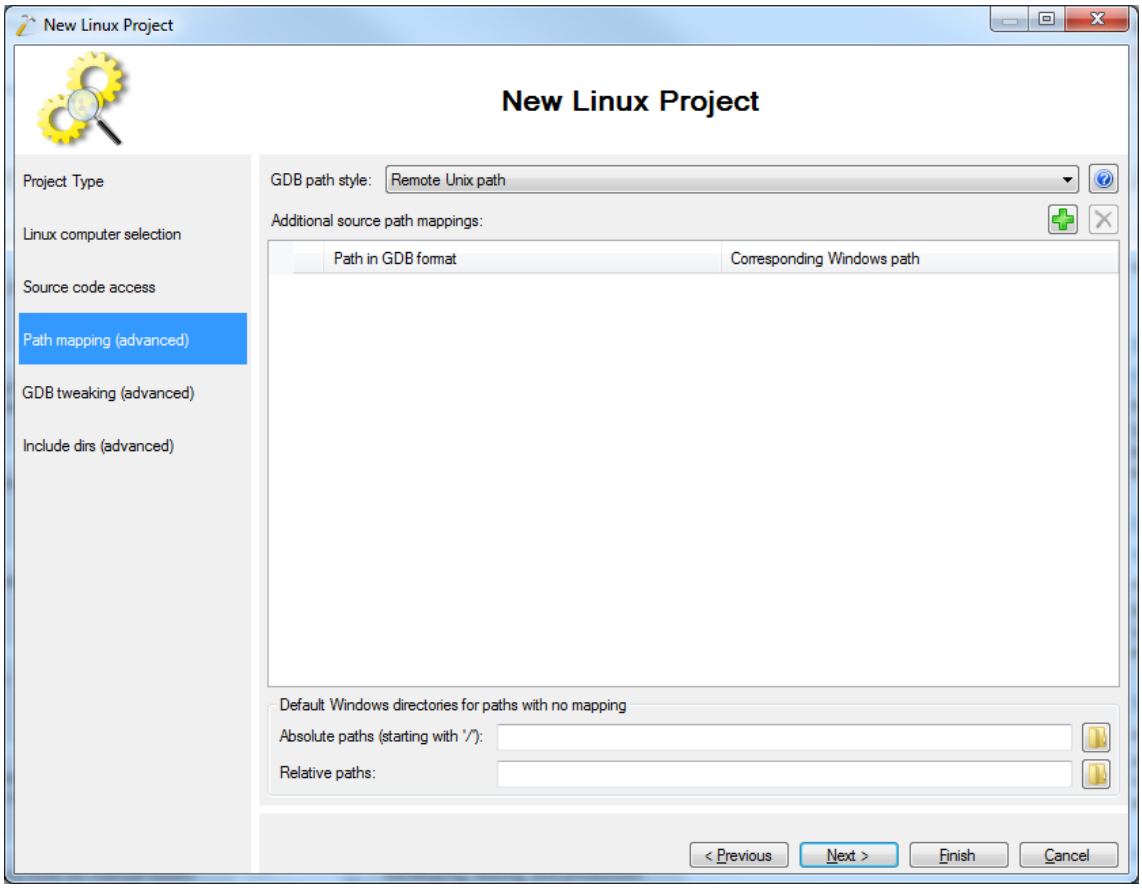


Согласитесь на предложенные варианты исправления и нажмите «OK»

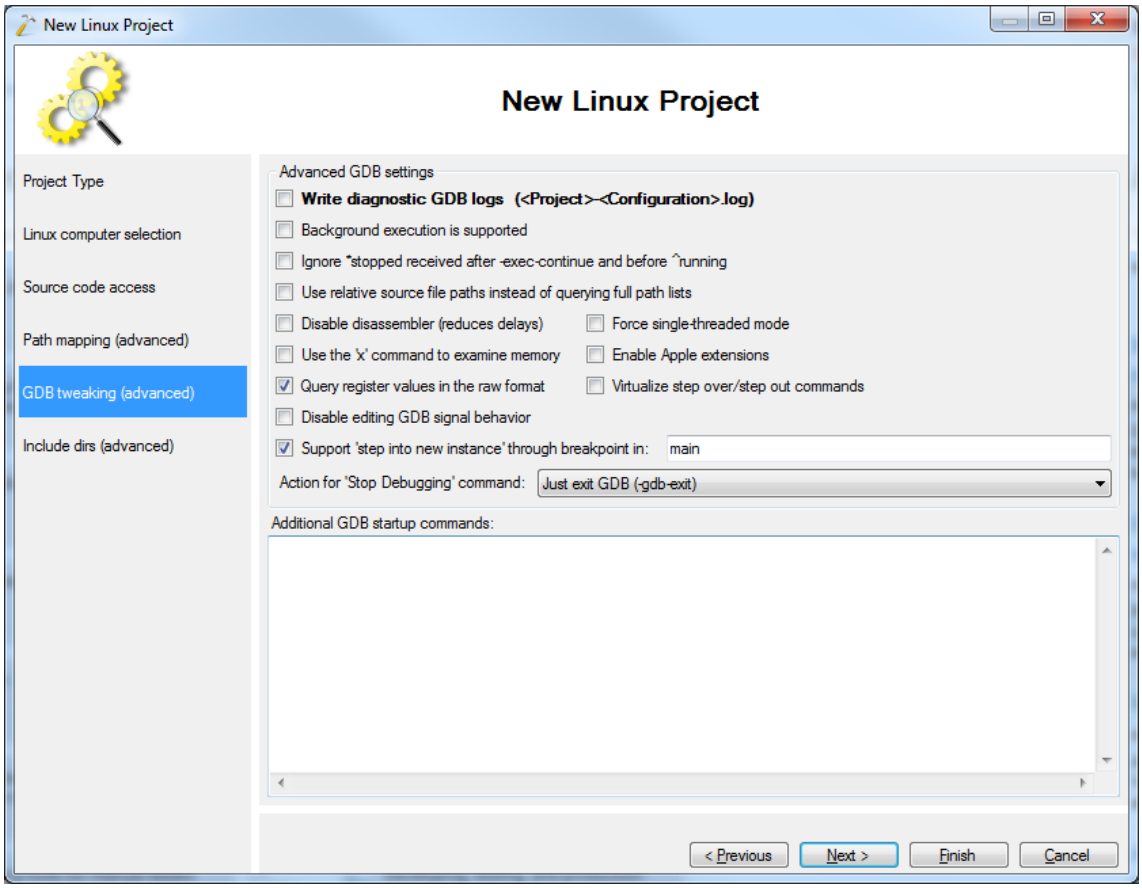
Далее выбираем пункт «Store source files on Windows computer»:



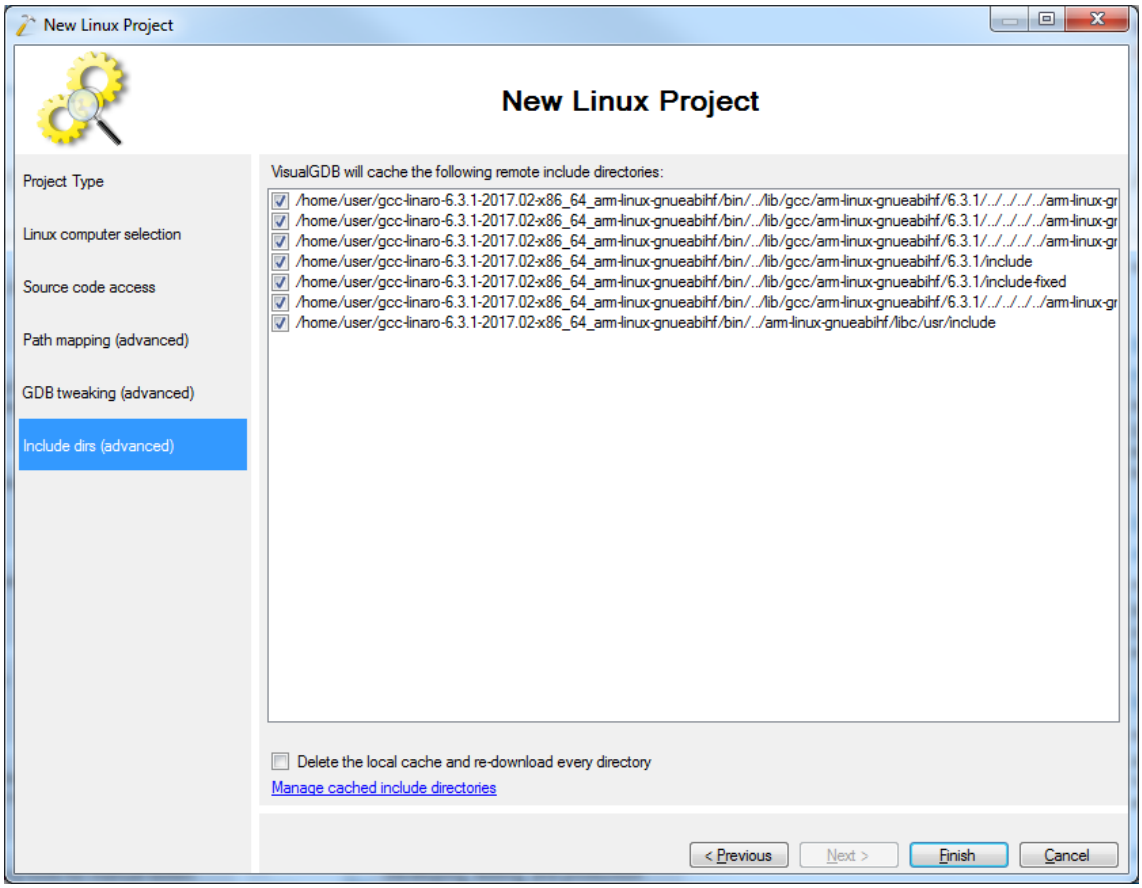
На этапе сопоставления путей можно ничего не менять.



Нажимаем «Next»



Нажимаем «Next»



Нажимаем «Finish». Создание проекта на этом закончено. Далее Visual Studio создаст решение, содержащее один проект, параметры которого мы только что закончили настраивать.

Позднее все настройки проекта можно изменить, выбрав в меню команду «Project-> VisualGDB Project Properties».

Так же при разработке проекта иногда требуется изменить параметры сборки, оптимизации, указать требуемые библиотеки и т.д. Эти параметры необходимо указать в makefile для соответствующего варианта сборки: Debug – debug.mak, Release – release.mak. Файлы makefile можно открыть через окно «Solution explorer».

5.2.2 Наполнение проекта кодом

Перейдем к наполнению проекта программным кодом.

Откроем окно «Solution explorer», содержащее все файлы нашего проекта, откроем файл «LEDTest.cpp». Файл содержит текст примера:

```
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
        char sz[] = "Hello, World!";    //Hover mouse over "sz" while debugging to see its contents
        cout << sz << endl;     //<================= Put a breakpoint here
        return 0;
}
```

Заменим весь имеющийся текст на текст нашей программы:

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <linux/fb.h>
#include <sys/ioctl.h>
#include <errno.h>
#include <string>
#include <string.h>
#include <vector>

using namespace std;
using std::vector;
using std::string;

int  g_iMode = 0;

int TestLEDs()
{
        vector<string>  arrDevices;

        arrDevices.push_back("/sys/class/leds/agava:backlight:red/brightness");
        arrDevices.push_back("/sys/class/leds/agava:backlight:green/brightness");
        arrDevices.push_back("/sys/class/leds/agava:backlight:blue/brightness");
        arrDevices.push_back("/sys/class/leds/agava:work:green/brightness");
        arrDevices.push_back("/sys/class/leds/agava:fault:red/brightness");
        arrDevices.push_back("/sys/class/leds/agava:programm:green/brightness");

        char val;
        FILE* f = NULL;

        int fd = -1;

        for (int t = 0; t < arrDevices.size(); t++)
        {
                f = fopen(arrDevices[t].c_str(), "r+");

                if (f != NULL)
                {
                        val = 0x31;
                        fwrite(&val, sizeof(char), 1, f);
                        fflush(f);

                        usleep(1000000);

                        val = 0x30;
                        fwrite(&val, sizeof(char), 1, f);

                        fclose(f);
                }
                else
                {
                        printf("Error opening led device %s.\n", arrDevices[t].c_str());
                }
        }

        return 0;
}

int main(int argc, char **argv)
{
        printf("LEDTest: AGAVA6432.30 leds testing program.\n");

        TestLEDs();

        return 0;
```

```
}
```

5.2.3 Сборка приложения

Скомпилируем проект: выбираем команду «Build->Build Solution».

В окне вывода появится текст:

```
1>------ Build started: Project: LEDTest, Configuration: Debug Win32 ------
1>  VisualGDB: Sending 6 updated source files to build machine...
1>  VisualGDB: Run "make CONFIG=Debug" in directory "/tmp/VisualGDB/d/LEDTest/LEDTest/LEDTest" on user@192.168.10.42
(SSH)
1>  mkdir Debug
1>  /home/user/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-g++ -ggdb -ffunction-
sections -O0   -DDEBUG=1  -c LEDTest.cpp -o Debug/LEDTest.o -MD -MF Debug/LEDTest.dep
1>  /home/user/gcc-linaro-6.3.1-2017.02-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-g++ -o Debug/LEDTest -
Wl,-gc-sections    -Wl,--start-group Debug/LEDTest.o   -Wl,--rpath='$ORIGIN'   -Wl,--end-group
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Приведенный текст содержит порядок действий системы при сборке приложения:

- отправка шести файлов с исходными текстами на машину сборки по адресу 192.168.10.42
- сборка приложения в варианте Debug на машине сборки 192.168.10.42

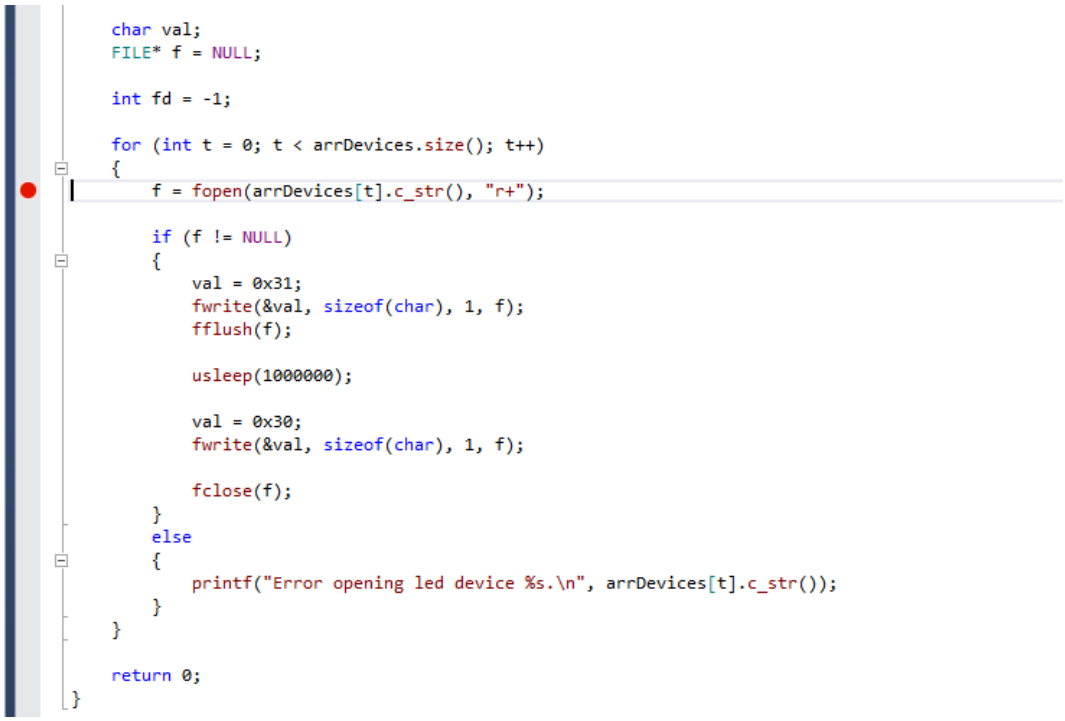
Сборка завершена, одно приложение собрано успешно.

5.2.4 Отладка приложения в среде Visual Studio

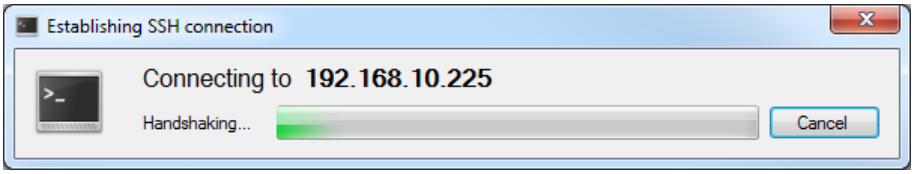
После успешной компиляции приложения можно перейти к отладке.

Поставим точку останова (клавиша F9 по умолчанию) на строке:

```
f = fopen(arrDevices[t].c_str(), "r+");
```



Вызовем команду «Debug->Start debugging with GDB». Появится окно подключения к целевому устройству:

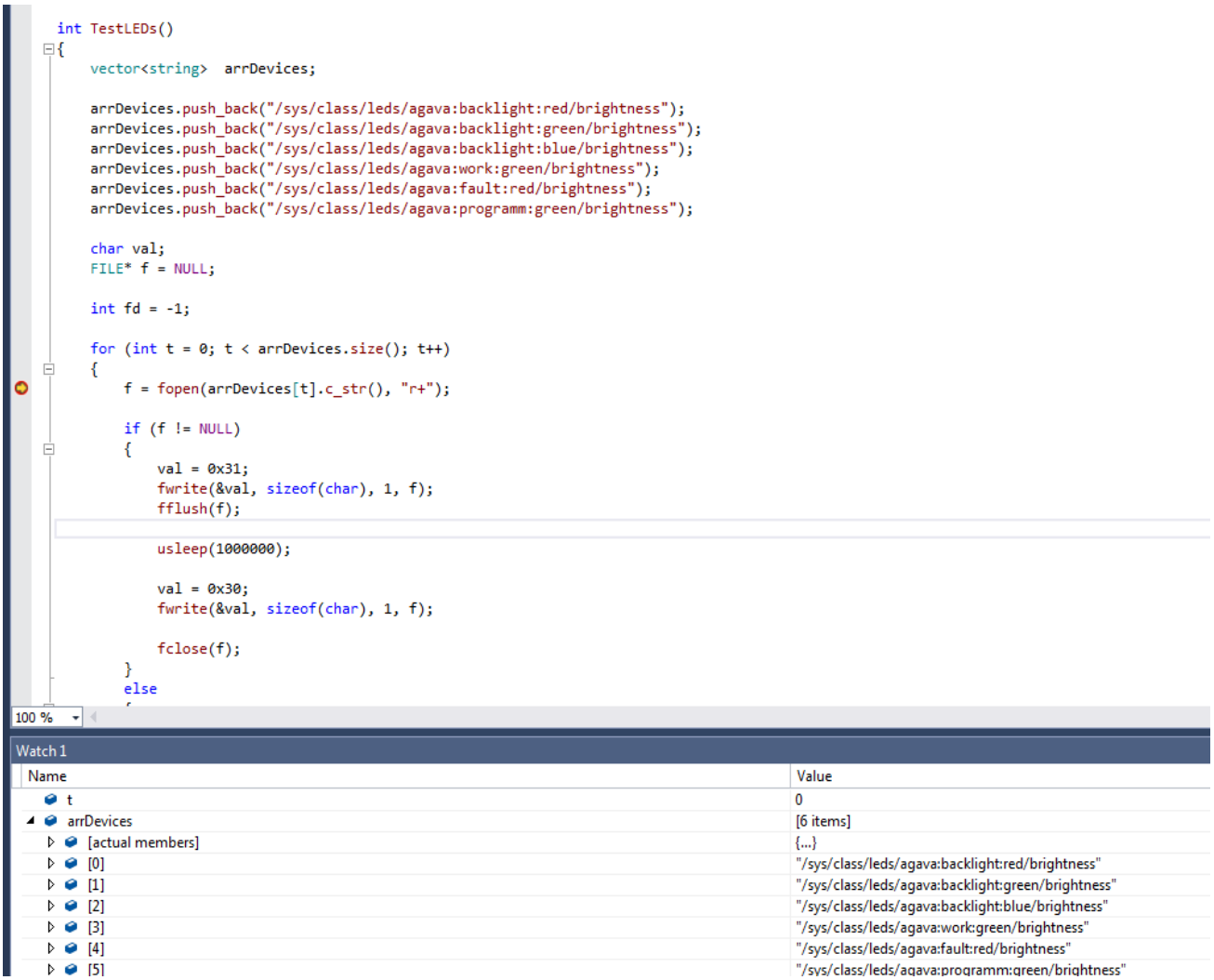


После подключения GDB клиента к GDB серверу на контролере начнется отладка программы и при достижении места с точкой останова выполнение будет приостановлено.

Вывод консоли:

```
Process /tmp/LEDTest created; pid = 640
Listening on port 2000
Remote debugging from host 192.168.10.42
LEDTest: AGAVA6432.30 leds testing program.
```

Откроем окно Watch 1, внесем в список отслеживаемых переменных переменные «t», «arrDevices»:



```
int TestLEDs()
{
    vector<string>  arrDevices;

    arrDevices.push_back("/sys/class/leds/agava:backlight:red/brightness");
    arrDevices.push_back("/sys/class/leds/agava:backlight:green/brightness");
    arrDevices.push_back("/sys/class/leds/agava:backlight:blue/brightness");
    arrDevices.push_back("/sys/class/leds/agava:work:green/brightness");
    arrDevices.push_back("/sys/class/leds/agava:fault:red/brightness");
    arrDevices.push_back("/sys/class/leds/agava:programm:green/brightness");

    char val;
    FILE* f = NULL;

    int fd = -1;

    for (int t = 0; t < arrDevices.size(); t++)
    {
        f = fopen(arrDevices[t].c_str(), "r+");

        if (f != NULL)
        {
            val = 0x31;
            fwrite(&val, sizeof(char), 1, f);
            fflush(f);

            usleep(1000000);

            val = 0x30;
            fwrite(&val, sizeof(char), 1, f);

            fclose(f);
        }
        else
```

100 %

Watch 1

Name	Value
t	0
arrDevices	[6 items]
[actual members]	{...}
[0]	"/sys/class/leds/agava:backlight:red/brightness"
[1]	"/sys/class/leds/agava:backlight:green/brightness"
[2]	"/sys/class/leds/agava:backlight:blue/brightness"
[3]	"/sys/class/leds/agava:work:green/brightness"
[4]	"/sys/class/leds/agava:fault:red/brightness"
[5]	"/sys/class/leds/agava:programm:green/brightness"

Далее отладку можно продолжить, запустив приложение на выполнение до следующей точки останова, либо продолжить выполнение программы по шагам.

6 Обновление базовых программных компонентов Контроллера

Прибор поставляется с установленными базовыми программными компонентами. В процессе эксплуатации прибора может возникнуть необходимость их обновления. Файлы программных компонентов могут быть получены через сайт Изготовителя – www.kb-agava.ru, либо предоставлены по запросу.

Базовое программное обеспечение Контроллера состоит из следующих модулей:

- Загрузчик;
- ОС Linux;
- Корневая файловая система;

Загрузчик служит для загрузки ОС, а также для обновления программных компонентов контроллера и по-умолчанию хранится во NAND-памяти Контроллера. Файлы компонентов загрузчика: u-boot.img (образ U-Boot) и MLO (первичный загрузчик). Данные файлы взаимосвязаны и должны применяться только совместно, одной и той же версии. При включении контроллера сначала происходит загрузка первичного загрузчика MLO во внутреннюю память процессора, который выполняет инициализацию необходимого оборудования и загружает основной загрузчик U-Boot, который впоследствии загружает компоненты ОС Linux и передает им управление. Контроллер позволяет выбирать источник загрузки при помощи микропереключателя 1, расположенного на боковой стенке лицевой панели – либо из NAND-памяти - это основной режим загрузки, положение микропереключателя «OFF» - вверх. Либо с SD-карты – это дополнительный режим загрузки для обновления или аварийного восстановления контроллера.

Программные компоненты ОС Linux хранятся в NAND-памяти контроллера и состоят из образа ядра Linux – файл zImage и файла описания устройств am335x-agava_30.dtb. Данные файлы взаимосвязаны и должны применяться только совместно, одной и той же версии.

Корневая файловая система содержит набор каталогов и утилит для нормальной работы ОС, хранится в NAND-памяти и монтируется при загрузке ядра ОС Linux. Имя файла образа для прошивки корневой файловой системы – agava.ubi.

Система исполнения CODESYS состоит из файлов codesyscontrol, codesyscontrol.a и CODESYSControl.cfg, размещена в корневой файловой системе в каталоге /usr/bin/codesys и запускается как сервис при загрузке ОС Linux. Данные файлы взаимосвязаны и должны применяться только совместно, одной и той же версии.

6.1 Обновление компонентов загрузчика

- Подготовить SD-карту с файловой системой FAT(12,16,32). Разместить в ее корневой каталог файлы для обновления MLO и u-boot.img. Установить SD-карту в прибор.
- Подключить Контроллер нуль-модемным кабелем к интерфейсу RS-232 компьютера. На компьютере настроить терминал, в соответствии с параметрами, указанными в п.0, выбрать соответствующий порт.
- Чтобы зайти в загрузчик, необходимо включить Контроллер и сразу нажимать любую клавишу в терминале компьютера до появления в нем строки приглашения:

```
AGAVA6432.30#.
```

- В терминале последовательно выполнить команды

```
run upd_mlo
run upd_u-boot
```

- Убедиться, что команды выполнились без ошибок.
- Выключить Контроллер, затем включить и повторно зайти в загрузчик как было указано выше. Убедиться, что произошла загрузка обновленной версии загрузчика.
- Обновить переменные окружения нового загрузчика, выполнив команды в терминале

```
env default –f –a
saveenv
reset
```

- Убедиться что, произошла полная загрузка контроллера.

6.2 Обновление компонентов ОС Linux

Порядок действия по обновлению компонентов ОС Linux:

- Подготовить SD-карту с файловой системой FAT(12,16,32). В корневом каталоге SD-карты создать папку boot. Разместить в папке файлы для обновления - zImage и am335x-agava_30.dtb. Установить SD-карту в прибор.
- Подключить Контроллер нуль-модемным кабелем к интерфейсу RS-232 компьютера. На компьютере настроить терминал, в соответствии с параметрами, указанными в п.0, выбрать соответствующий порт.
- Чтобы зайти в загрузчик, необходимо включить Контроллер и сразу нажимать любую клавишу в терминале компьютера до появления в нем строки **AGAVA6432.30#**.
- В терминале последовательно выполнить команды

```
run upd_fdt
run upd_linux
reset
```

- Убедиться, что команды выполнилась без ошибок и контроллер перезагрузился с обновленной версией Linux.

6.3 Обновление корневой файловой системы

Внимание! При обновлении корневой файловой системы все пользовательские настройки, проекты и иные файлы пользователя не сохраняются. Перед обновлением их необходимо сохранить самостоятельно.

Порядок действий по обновлению файловой системы:

- Подготовить SD-карту с файловой системой FAT(12,16,32). Разместить в корневой каталог файл образа корневой файловой системы agava.ubi. Установить SD-карту в прибор.

- Подключить Контроллер нуль-модемным кабелем к интерфейсу RS-232 компьютера. На компьютере настроить терминал, в соответствии с параметрами, указанными в п.0, выбрать соответствующий порт.
- Для входа в загрузчик необходимо включить Контроллер и сразу начать нажимать любую клавишу в терминале компьютера до появления в нем строки **AGAVA6432.30#**.
- В терминале последовательно выполнить команды

```
run upd_rootfs
reset
```

- Убедиться, что прошивка выполнилась без ошибок и контроллер перезагрузился с обновленной корневой файловой системой.

6.4 Обновление прикладного (пользовательского) программного обеспечения

В операционную систему встроен сервис обновления agava-uob, выполняющийся при загрузке контроллера.

При старте сервиса он проверяет на подключенных USB flash наличие файла update.sh, и если таковой обнаружен, выполняет его.

В скрипте update.sh описываются все действия, необходимые для выполнения обновления ПО.

Ниже приведен пример скрипта, производящего подачу звукового сигнала и перезагрузку контроллера:

```
#!/bin/sh
#Update script

USB_DIR=`dirname $0`
#Example usage:
#$USB_DIR/ledtest

#Make sound:
echo -en "\07" > /dev/tty5

#Echo message to console and reboot:
echo "Done!Rebooting" >> /dev/tty0
reboot
```

АГАВА ПК-30