Руководство программиста АГАВА ПК-70

□

Содержание

Введение
- Используемые термины и сокращения

Назначение

Настройка прибора
- Установка времени и даты
- Доступ к файлам контроллера
  - Настройка сетевого доступа

Аппаратные ресурсы контроллера
- Звуковой извещатель
- Светодиоды
- EEPROM
- RTC
- Подсветка TFT
- Touchscreen
- Последовательные порты
- Ethernet
- USB OTG
- Субмодули

Настройка виртуальной машины
- Вход в ВМ
- Настройка сетевого взаимодействия

Разработка пользовательского прикладного программного обеспечения с использованием компиляторов C/C++
- Разработка простого приложения с использованием системы сборки make в среде Linux
  - Создание Makefile
  - Создание текста программы.
  - Сборка приложения
- Разработка простого приложения с использованием средств Visual Studio и VisualGDB в среде Windows
  - Создание проекта VisualGDB
  - Наполнение проекта кодом

Сборка приложения
Отладка приложения в среде Visual Studio
Разработка графического приложения с использованием Qt
Создание проекта в Qt Creator
Дополнительная настройка проекта
Размещение приложения на целевом устройстве
Создание целевого устройства
Смена целевого устройства в комплекте сборки
Сборка и отладка приложения

1 Введение

Руководство по эксплуатации содержит сведения, необходимые для обеспечения правильной эксплуатации и полного использования технических возможностей промышленного контроллера АГАВА ПК-40, далее по тексту ПРИБОР, ПЛК или КОНТРОЛЛЕР.

Разработка приложений для контроллеров серии АГАВА ПК предполагает использование SDK в виде виртуальной машины с установленным программным обеспечением.

1.1 Используемые термины и сокращения

SDK – Software development kit – комплект средств разработки приложений

SSH – Secure Shell – протокол защищенного подключения

ВМ – виртуальная машина

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-40 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 обеспечение. 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 монтирования:

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

Системная консоль находится на последовательном порте 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);
- 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

По умолчанию Прибор настроен на получение сетевого адреса от DHCP-сервера.

Просмотреть IP-адрес и другую сетевую конфигурацию для всех интерфейсов Ethernet можно из консоли, набрав команду:

```
ifconfig
```

```
root@agava6432_40:~# ifconfig
eth0      Link encap:Ethernet  HWaddr A8:1B:6A:48:71:93
          inet addr:192.168.10.215  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::aa1b:6aff:fe48:7193%132688/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500 Metric:1
          RX packets:865 errors:0 dropped:16 overruns:0 frame:0
          TX packets:98 errors:0 dropped:0 overruns:0 carrier:0
```

```
          collisions:0 txqueuelen:1000
          RX bytes:75565 (73.7 KiB)  TX bytes:13702 (13.3 KiB)
          Interrupt:173

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1%132688/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536 Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:540 (540.0 B)  TX bytes:540 (540.0 B)
usb0      Link encap:Ethernet  HWaddr 46:10:3A:B3:AF:D9
          inet addr:192.168.7.1  Bcast:192.168.7.3  Mask:255.255.255.252
          UP BROADCAST MULTICAST  MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
root@agava6432_40:~#
```

Задать статический IP-адрес интерфеса eth0 можно в файле /etc/systemd/network/10-eth.network, например:

```
[Network]
DHCP=no
Address=192.168.10.100/24
Gateway=192.168.10.10
```

Для редактирования файла можно воспользоваться встроенным фаловым менеджером mc, либо установить подключение к Прибору через USB интерфейс (адрес ПЛК - 192.168.7.1) по протоколу SFTP и изменить файл имеющимся в ОС текстовым редактором.

4 Аппаратные ресурсы контроллера

4.1 Звуковой извещатель

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

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

4.2 Светодиоды

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

agava:work:green - зеленый светодиод "Работа";

agava:fault:red - красный светодиод "Авария";

agava:panel:fault:red - красный светодиод "Авария" на выносной панели;

agava:panel:fault_usr:green - зелёный светодиод "Авария" на выносной панели;

agava:panel:prog_usr:red - красный светодиод "Программа" на выносной панели;

agava:panel:programm:green - зелёный светодиод "Программа" на выносной панели;

agava:panel:work:green - красный светодиод "Работа" на выносной панели;

agava:panel:work_usr:red - красный светодиод "Работа" на выносной панели.

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.
```

4.3 EEPROM

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

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".

4.4 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.

4.5 Подсветка TFT

Для уменьшения уровня подсветки экрана выполните команду:

```
echo 1 > /sys/devices/platform/backlight/backlight/backlight/brightness
```

Для уменьшения уровня подсветки экрана выполните команду:

```
echo 8 > /sys/devices/platform/backlight/backlight/backlight/brightness
```

4.6 Touchscreen

Для проверки тачскрина, подключенному к встроенному в процессор контроллеру, выполнить команду:

```
evtest /dev/input/touchscreen0
```

или

```
evtest /dev/input/by-path/platform-TI-am335x-tsc-event
```

Нажимая на экран, можно убедиться, что регистрируются соответствующие события.

Для калибровки тачскрина при использовании графической системы Weston, необходимо удалить файлы, выполнив следующие команды:

```
rm /etc/udev/rules.d/ws-calibrate.rules
rm /run/media/mmcblk0p1/ws-calibrate.rules
reboot
```

4.7 Последовательные порты

Последовательные порты регистрируются в системе как устройства /dev/ttyS0 - /dev/ttyS5.

Устройство /dev/ttyS0 - локальная шина для связи с модулями.

Устройство /dev/ttyS1 - RS-232 (системная консоль).

Устройство /dev/ttyS2 - порт RS-485 №1.

Устройство /dev/ttyS3 - порт RS-485 №2.

Устройство /dev/ttyS4 - порт RS-485 №3.

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

```
stty -F /dev/ttyS0 1200
```

или 230400 бит/сек:

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

4.8 Ethernet

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

4.9 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.

4.10 Субмодули

Доступ к сигналам, обрабатываемым субмодулями, осуществляется по протоколу Modbus-RTU. Описание реализации протокола обмена приведено в отдельном документе.

5 Настройка виртуальной машины

5.1 Вход в ВМ

Для входа в ВМ используйте следующие реквизиты:

Пользователь: user

Пароль: user

5.2 Настройка сетевого взаимодействия

Для подключения ВМ к локальной вычислительной сети как полноценного ПК может потребоваться изменение IP-адреса.

Для разового задания IP-адреса машины, действующего до перезагрузки, воспользуйтесь командой (потребуется ввод пароля пользователя):

```
sudo ifconfig eth0 <address> netmask 255.255.255.0
```

Для задания постоянного адреса отредактируйте файл /etc/network/interfaces:

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback
```

```
# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.10.42
netmask 255.255.255.0
gateway 192.168.10.10
dns-nameservers 192.168.10.10
```

6 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 (см. раздел 9). 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
- Qt Creator

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

Разработаем небольшое приложение, использующее аппаратное обеспечение контроллера АГАВА 6432.40.

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

6.1.1 Создание Makefile

Воспользуемся готовым основным makefile, содержащимся в файле “Makefile” в каталоге LEDTest. Описание принципов создания makefile приведено в соответствующей литературе (см. раздел 9). 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.

6.1.2 Создание текста программы.

Напишем основной код приложения на языке C++, и поместим его в файл «LEDTest40.cpp»

6.1.3 Сборка приложения

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

user@ubuntu:~/applications/LEDTest40$ 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/LEDTest40$ 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 LEDTest40.cpp -o Debug/LEDTest40.o -MD -MF Debug/LEDTest40.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/LEDTest40 -Wl,-gc-sections   -Wl,--start-group
Debug/LEDTest40.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.

6.2 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

6.2.1 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».

6.2.2 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:panel:work:green/brightness");
        arrDevices.push_back("/sys/class/leds/agava:panel:work_usr:red/brightness");
        arrDevices.push_back("/sys/class/leds/agava:panel:fault_usr:green/brightness");
        arrDevices.push_back("/sys/class/leds/agava:panel:fault:red/brightness");
        arrDevices.push_back("/sys/class/leds/agava:panel:programm:green/brightness");
        arrDevices.push_back("/sys/class/leds/agava:panel:prog_usr:red/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 (fi!= 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: AGAVA6432.40 leds testing program.\n");
        TestLEDs();

        return 0;
}
```

6.2.3 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.

6.2.4 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 целеому 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/LEDTest40 created; pid = 640
Listening on port 2000
Remote debugging from host 192.168.10.42
LEDTest: AGAVA6432.40 leds testing program.
```

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

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

7 Разработка графического приложения с использованием Qt

7.1 Создание проекта в Qt Creator

Создадим проект с именем «QtTest1» в среде Qt Creator, который будет хранить создаваемое нами приложение.

Открываем среду разработки. Вызываем команду «New Project».

Открывается окно создания проекта, в котором на первом этапе необходимо выбрать тип разрабатываемого приложения. Выбираем вариант «Qt widgets application» и нажимаем кнопку «Choose…»:

Далее вводим имя нового приложения, а так же путь, по которому он будет храниться и нажимаем кнопку «Next >»:

Далее переходим к выбору комплектов сборки приложений. В SDK AGAVA в среде Qt Creator уже создан комплект «AGAVA40-ARM7», необходимый для сборки приложений под контроллер АГАВА ПК40. Оставляем выбор, как указано на рисунке ниже и нажимаем кнопку «Next >»:

Далее нам предоставляется возможность изменить имена создаваемых файлов и классов. Если требуется внести изменения, вносим их и нажимаем кнопку «Next >»:

На последнем этапе создания проекта можно подключить к проекту систему контроля версий. В виртуальной машине не установлены системы контроля версий, поэтому просто нажимаем кнопку «Finish»:

На этом создание проекта можно считать завершенным. Созданный проект можно скомпилировать, но без дополнительной настройки его невозможно разместить и запустить на целевом устройстве.

7.2 Дополнительная настройка проекта

Для добавления возможности компиляции и размещения проекта на целевом устройстве необходимо:

· Задать путь размещения проекта на целевом устройстве

· Создать нужное целевое устройство в Qt Creator для размещения и запуска проектов

7.2.1 Размещение приложения на целевом устройстве

Для определения пути размещения приложения на целевом устройстве необходимо в файле проекта разместить примерно следующие строки, определяющие тип целевого устройства и путь размещения:

```
linux-* {
target.path = /tmp
INSTALLS += target
}
```

7.2.2 Создание целевого устройства

Для размещения приложения на целевом устройстве и возможности его запуска/отладки необходимо создать целевое устройство в Qt Creator. Для этого вызовем команду «Tools | Options», и перейдем на вкладку «Devices»:

Рис. 1 Окно "Options"

Для создания нового устройства нажмем кнопку «Add».

В появившемся окне выбираем вариант «Generic Linux Device» и нажимаем кнопку «Start Wizard»:

Далее в появившемся окне «New Generic Linux Device Configuration setup» вводим параметры нового устройства, примерно как показано на рисунке ниже:

Нажимаем кнопку «Next >», и на следующем экране подтверждаем создание устройства. После завершения создания нового целевого устройства его можно указать в комплекте сборки как используемое.

7.2.3 Смена целевого устройства в комплекте сборки

При смене адреса целевого устройства или для размещения/отладки приложения на другом целевом устройстве необходимо в используемом комплекте сборки «AGAVA40-ARM7» указать другое целевое устройство. Процедура создания нового целевого устройства описана в п. 7.2.2. После создания устройства в окне «Options» перейдем на вкладку «Build & run», раздел «Kits», и в поле «Device» будет возможность выбрать новое целевое устройство.

7.3 Сборка и отладка приложения

---

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

Источник — https://docs.kb-agava.ru/index.php?title=Руководство_программиста_АГАВА_ПК-70&oldid=3375

Эта страница в последний раз была отредактирована 25 декабря 2025 в 12:18.