**Banana Pi M2 Zero**

**Porting and Compile**

****

***Ubuntu 16.04 LTS Version***

|  |
| --- |
| SECURITY WARNING  이 문서에 포함되어 있는 정보는 전자부품연구원(KETI)의 자산이며, 전자부품연구원(KETI)의 서면 허락 없이 타인에 배포되거나, 재가공 되거나, 제공된 기업사 이외로 반출 및 공개될 수 없습니다. |

문서정보 / 수정 내역

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 수정자 | 버전 | 추가/수정 항목 | 내 용 |
| 2020-06-18 | 오규태 | 1.0 |  | NanoPi NEO Air Porting & Compile |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

문 서 규 칙

* 작성 및 확인은 Microsoft Word 2010으로 작성 되었으며, Acrobat Reader로 읽는다.
* Category는 설계서, 기술문서, 분석서, 사양서로 구분하며, 기재된 정보가 표기되며, 머리글의 Category에 해당 구분 정보를 표기된다.
* 본 문서의 사용 대상이 되는 기술 명은 오른쪽 큰 여백에 기재되며(Embedded IT 1.x), 버전 정보만 틀린 경우에는 제품의 버전까지 표기된다.
* 첨부 파일 버전은 첨부 파일이 존재하는 경우에 기재되며, 첨부 파일의 버전이 표기된다
* 문서 최종 수정 일에는 문서의 최종 수정 일이 표기된다.

주 의 사 항

* 기술 군의 특성상 동일 제품명에도 여러 개의 버전이 존재할 수 있으므로, 이 문서에 표기된 버전을 확인하지 않거나, 본문에 나타난 주의 사항 등을 준수하지 않아 발생된 문제에 대해서는 전자부품연구원(KETI)에서는 책임을 지지 아니한다.
* 본 문서에 대한 정확한 용도를 확인할 수 없을 경우 기술지원부서(031-789-7000) 또는 기술문서 담당자에게 당 문서에 대한 용도를 확인 후 작업을 진행해야 한다.

목차

[1. introduction 4](#_Toc340482766)

[2. hardware specification 4](#_Toc340482769)

[3. software features 6](#_Toc340482772)

[3.1 write linux sd card based on windows platform 6](#_Toc340482773)

[3.2 boot your orangepi 6](#_Toc340482774)

[3.3 login via vnc and ssh 9](#_Toc340482775)

[4. install package and test program 10](#_Toc340482772)

[4.1 install package 1](#_Toc340482773)0

[4.2 test program 11](#_Toc340482773)

[4.2.1 ethernet test 11](#_Toc340482773)

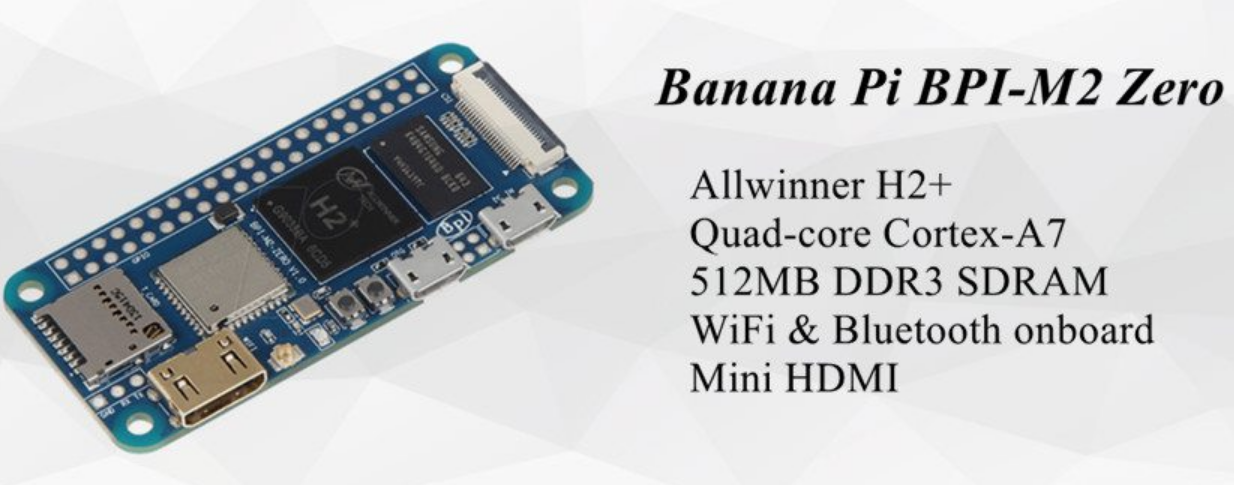
[4.2.2 uart test 14](#_Toc340482773)

[4.2.3 i2c test 15](#_Toc340482773)

[4.2.4 spi test 17](#_Toc340482773)

# Introduction

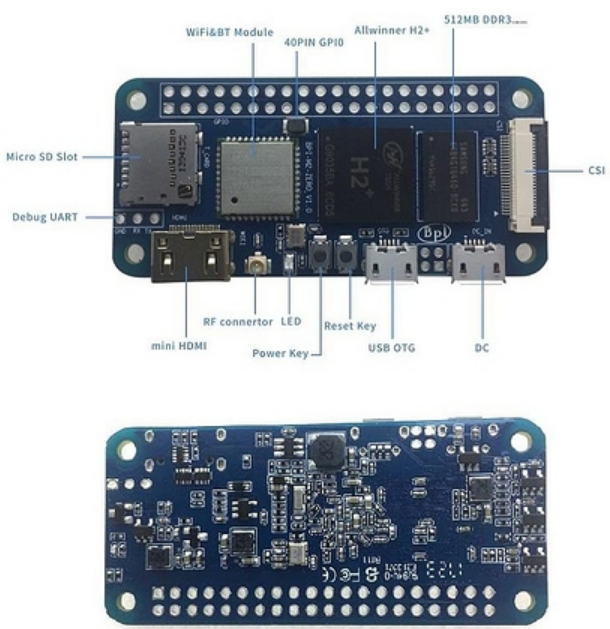
Banana Pi M2 Zero는 60mm \* 30mm 크기의 초소형 단일 보드 컴퓨터이다. 외부 인터페이스는 Raspberry Pi Zero W와 동일하며 거의 모든 케이스 및 Zero W 용 액세서리에 적합함. 512MB RAM의 쿼드 코어 Cortex A7 Allwinner H2 + 프로세서를 사용. 경량 시스템 및 공간 제한적인 애플리케이션에 이상적. Banana Pi의 다른 제품군과 마찬가지로 Linux 및 Android 운영 체제를 모두 지원 가능.



<Banana Pi BPI-M2 Zero>

# Hardware Spec

|  |  |
| --- | --- |
| **Hardware Specification of Banana Pi BPI-M2 Zero** | |
| CPU | H2+ Quad-core Cortex-A7 H265/HEVC 1080P |
| GPU | Mali400MP2 GPU @600Mhz, Supports OpenGL ES 2.0 |
| Memory (SDRAM) | 512MB DDR3(Shared with GPU) |
| Onboard Storage | TF card (Max. 64GB) |
| WiFi Antenna | SDIO AP6212（option AP6181、AP6335） |
| Onboard Network | NOPE (but can extension with PIN define) |
| Onboard WiFi | XR819, IEEE 80.211b/g/n |
| Audio Input | Mini HDMI |
| Video Outputs | Mini HDMI |
| Power Source | USB OTG can supply power(Integrated POE power supply) |
| USB 2.0 Ports | one USB 2.0 OTG |
| Buttons | Power button, Reset button |
| Low-level peripherals | 40 Pins Header,compatible with Raspberry Pi 3 |
| GPIO(1x3) pin | UART, ground |
| LED | Power led & Status led |
| Supported OS | Android, Ubuntu, Armbian |
| **Interface definition** | |
| Product size | 65mm x 30mm |
| Weight | 15g |
| Banana Pi™ SINOVOIP CO.,LIMITED | |





<Banana Pi M2 Zero Pin out>

|  |  |  |  |
| --- | --- | --- | --- |
| **40 PIN GPIO of Banana pi BPI-M2 zero** | | | |
| GPIO Pin Name | Default Function | Function2：GPIO | Function3 |
| CON2-P01 | VCC-3V3 |  |  |
| CON2-P02 | VCC-5V |  |  |
| CON2-P03 | TWI0-SDA | PA12-EINT12 |  |
| CON2-P04 | VCC-5V |  |  |
| CON2-P05 | TWI0-SCK | PA11-EINT11 |  |
| CON2-P06 | GND |  |  |
| CON2-P07 | PWM1 | PA6-EINT6 |  |
| CON2-P08 | UART3-TX | PA13-EINT13 | SPI1-CS |
| CON2-P09 | GND |  |  |
| CON2-P10 | UART3-RX | PA14-EINT14 | SPI1-CLK |
| CON2-P11 | UART2-RX | PA1-EINT1 |  |
| CON2-P12 | UART3-CTS | PA16-EINT16 | SPI1-MISO |
| CON2-P13 | UART2-TX | PA0-EINT0 |  |
| CON2-P14 | GND |  |  |
| CON2-P15 | UART2-CTS | PA3-EINT3 |  |
| CON2-P16 | UART3-RTS | PA15-EINT15 | SPI1-MOSI |
| CON2-P17 | VCC-3V3 |  |  |
| CON2-P18 | PC4 | PC4 |  |
| CON2-P19 | SPI0-MOSI | PC0 |  |
| CON2-P20 | GND |  |  |
| CON2-P21 | SPI0-MISO | PC1 |  |
| CON2-P22 | UART2-RTS | PA2-EINT2 |  |
| CON2-P23 | SPI0-CLK | PC2 |  |
| CON2-P24 | SPI0-CS | PC3 |  |
| CON2-P25 | GND |  |  |
| CON2-P26 | PC7 | PC7 |  |
| CON2-P27 | TWI1-SDA | PA19-EINT19 |  |
| CON2-P28 | TWI1-SCK | PA18-EINT18 |  |
| CON2-P29 | PA7-EINT7 | PA7-EINT7 |  |
| CON2-P30 | GND |  |  |
| CON2-P31 | PA8-EINT8 | PA8-EINT8 |  |
| CON2-P32 | PL2-S-EINT2 | PL2-S-EINT2 |  |
| CON2-P33 | PA9-EINT9 | PA9-EINT9 |  |
| CON2-P34 | GND |  |  |
| CON2-P35 | PA10-EINT10 | PA10-EINT10 |  |
| CON2-P36 | PL4-S-EINT4 | PL4-S-EINT4 |  |
| CON2-P37 | PA17-EINT17 | PA17-EINT17 | SPDIF-OUT |
| CON2-P38 | PA21-EINT21 | PA21-EINT21 |  |
| CON2-P39 | GND |  |  |
| CON2-P40 | PA20-EINT20 | PA20-EINT20 |  |

# Software Features

지원되는 소프트웨어는 Android, Ubuntu, Debian 이 있다. 본 문서에서는 Ububtu 16.04 LTS 버전을 설명함

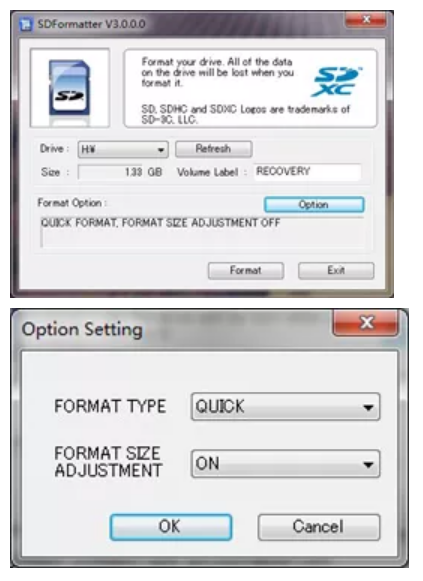
## Base your need below

* Banana pi SBC(single board computer)
* microSD Card/TFCard: Class 10 or Above, minimum 8GB SDHC
* power adapter via micro USB or DC port. A 5V/2A power is a must
* A Host computer running Ubuntu 16.04 64 bit system
* a serial communication board



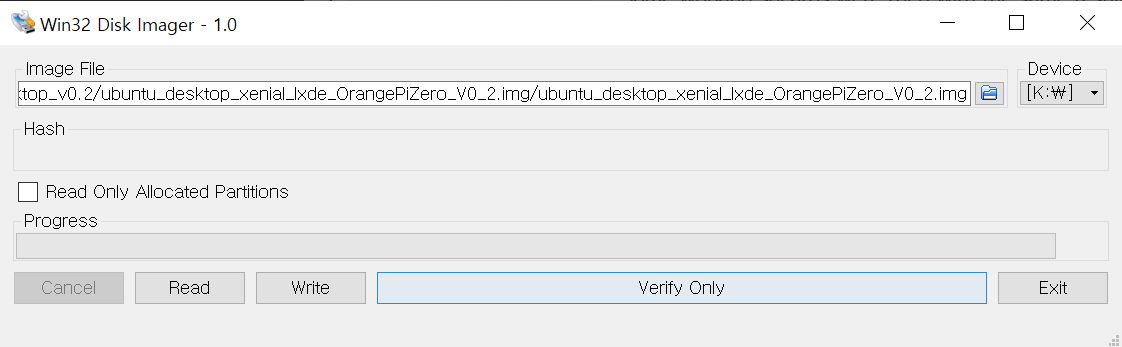
## Write Linux SD Card Based on Windows Platform

1. SD 카드를 컴퓨터에 삽입하면 SD 카드 용량이 운영 체제보다 커야 하며, 일반적으로 8GB 이상이 필요함
2. SD 카드 포맷
   1. SD 카드 포맷 전용 툴을 이용하여 SD 카드 포맷 진행
   2. <https://www.sdcard.org/downloads/formatter_4/eula_windows/>
   3. setup.exe 실행
   4. SD 카드 선택 후 빠른 부팅 진행



<SDFormatter.exe>

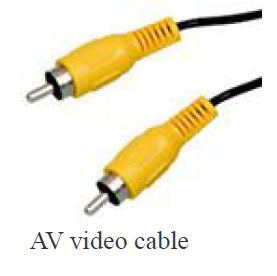
1. 우분투 운영체제 이미지를 다운
   1. <http://www.orangepi.org/downloadresources>
2. 압축 해제 후 이미지 쓰기 프로그램을 이용
   1. <http://sourceforge.net/projects/win32diskimager/files/Archive/>



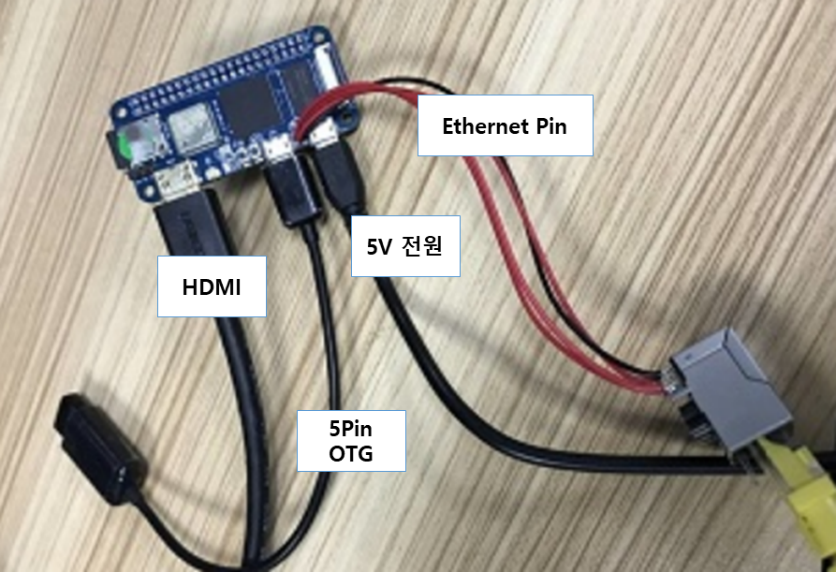
* 1. 압축 해제 후 우분투 이미지 파일 Write 실행

## Boot your Orange Pi

* SD 카드를 오렌지 파이에 삽입 후, 운영체제 설정을 위해 모니터 AV 케이블 셋팅 필요

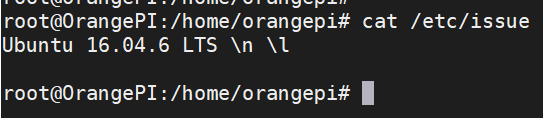


* 오펜지 파이 운영체제 설정 및 화면 설정을 위한 구성품
  1. AV 케이블
  2. AV type 지원 모니터
  3. 키보드&마우스 일체형 디바이스 (오렌지 파이는 기본적으로 1개의 USB 2.0을 지원)
  4. 5V A-type 전원 케이블
  5. Ethernet 케이블

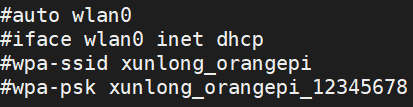


<Banana Pi M2 Zero development kit>

* Ununtu version



* 초기 패스워드
  1. 기본 username과 패스워드 모두 ‘bananapi’
  2. Ubuntu command를 이용하여 username&password 변경 가능
* 네트워크 설정
  1. 명령어 ifconfig를 통하여 현재 네트워크 상태 확인
     1. 자동 네트워크 설정
        + 현재 연결 된 공유기 및 LAN port를 통하여 자동으로 네트워크 인터페이스가 설정 됨
     2. 수동 네트워크 설정
        + 먼저 ssid와 psk (계정 및 비밀번호)를 알아야하며 해당 WLAN \*, ssid, psk를 입력함
        + sudo nano /etc/network/interfaces

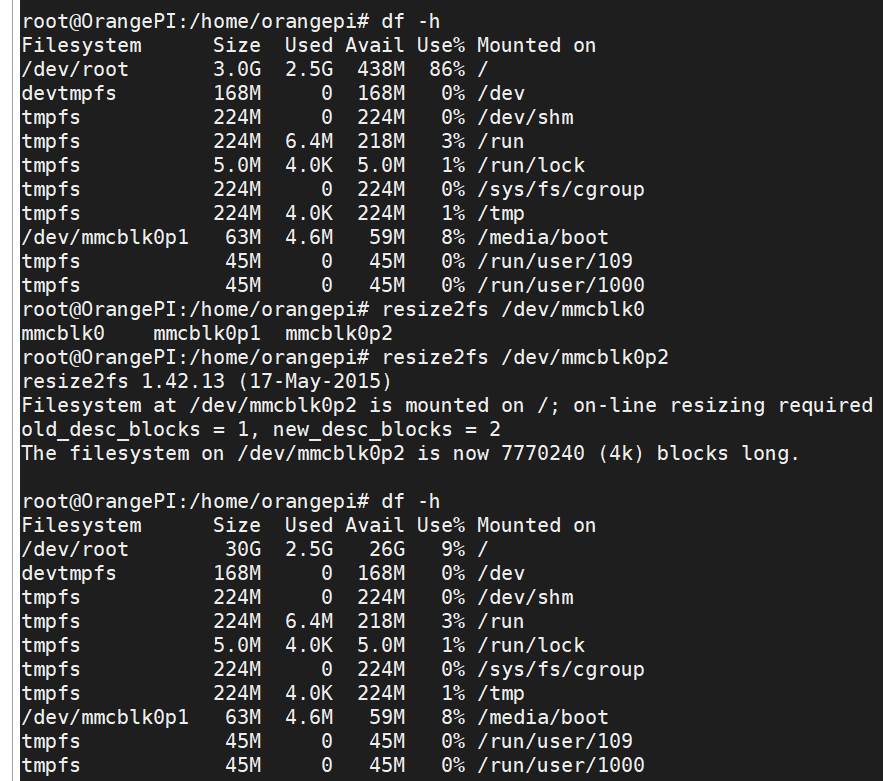


<Network interface>

* + - * 저장 후, reboot
* 언어 설정
  1. 환경 설정 이후 desktop preferences 메뉴에서 원하는 언어 설정
  2. 명령어를 통한 언어 설정
     1. sudo dpkg-reconfigure locales
  3. 설정 이후 reboot

## VNC 및 SSH를 통한 로그인

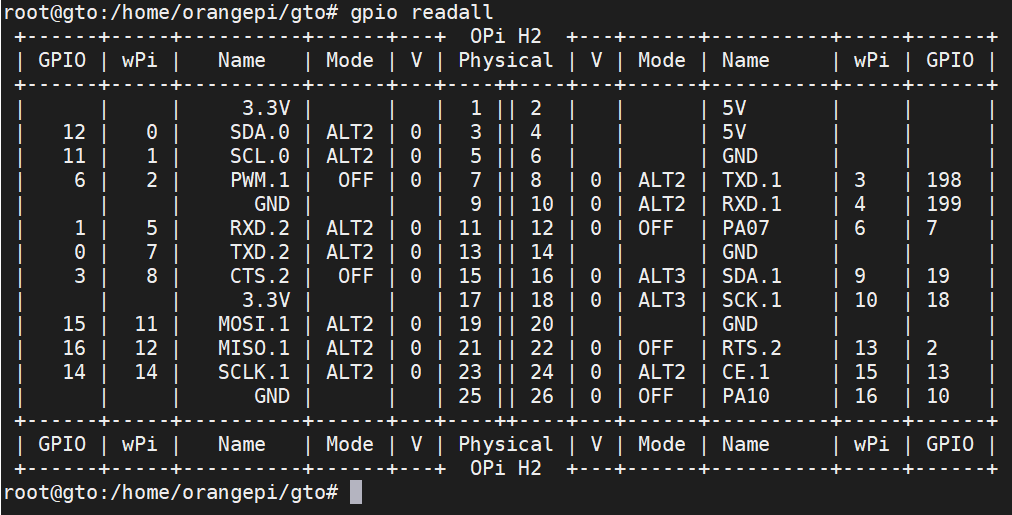
* HDMI 연결 조건이 없이 vnc 또는 ssh 원격 로그인을 통해 시스템에 들어가기 위한 설정이 필요
* ssh 설치
  1. sudo apt-get install ssh
* ssh 설정 경로
  1. /etc/ssh/sshd\_config
* 이후 ssh 응용 애플리케이션을 이용하여 ip와 port 입력 후 사용자가 정의한 user 정보에 의해 ssh 접속 가능
* 파티션 재정의



# Software Features

## Install package

* vim install
  1. vi editor를 이용하기 위해 설치 (gedit, nano… 등 다른 에디터를 이용해도 무관)
  2. sudo apt-get install wirinpi
* git install
  1. git 저장소 이용을 위해 설치 (오픈 소스 이용 시 용이)
  2. sudo apt-get install git
* wiringPi
  1. gpio 설정을 위해 필요한 라이브러리 설치
  2. <http://www.orangepi.org/Docs/WiringPi.html>
  3. git clone <https://github.com/orangepi-xunlong/WiringOP>
  4. chmod +x ./build
  5. sudo ./build
  6. gpio readall



<GPIO information>

* pip
  1. pip를 통한 파이썬 관련 라이브러리 및 지원 소프트웨어 설치를 위해 설치
  2. C언어로 프로그램 작성을 해도 무관하지만 라이브러리 접근이 용이한 파이썬이 디벨롭먼트 환경에 편리함
  3. sudo apt-get install python-pip
  4. pip명령어를 이용하여 다양한 라이브러리 이용 가능
  5. ex) pip list => 현재 설치되어 있는 패키지 목록 확인
  6. pip install –upgrage pip (version upgrade)
* i2c utils
  1. apt-get install i2c-utils
* exc utils
  1. pip install pillow==2.2.1 (i2c 디스플레이를 위한 라이브러리, 상위 버전은 이용 X)
  2. pip install smbus2 --no-cache-dir (i2c bus 라이브러리)

## Exam source

※ Example program PATH : */home/orangepi/gto/gto\_Exam*

1. Ethernet test

|  |
| --- |
| c\_udp\_server.c |
|  |
| * udp Target device : android device, up to API 4.3 * UDP 통신 순서   1. UDP 서버 프로그램 실행   2. 타겟 디바이스에서 서버 IP/Port 설정 이후 연결   3. 데이터 송수신 진행 |
| ***UDP Application test*** |

1. UART test

|  |
| --- |
|  |
| uart\_.py.py |
|  |
|  |

1. I2C test

※ Example program PATH : /home/orangepi/gto/gto\_Exam/i2c/ssd1306/examples/

|  |
| --- |
| sys\_info.py |
|  |
|  |

1. SPI

※ Example program PATH : /home/orangepi/gto/gto\_Exam/spi/spidev-test

|  |
| --- |
| vi spidev\_test.c |
| #include <stdint.h>  #include <unistd.h>  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <getopt.h>  #include <fcntl.h>  #include <sys/ioctl.h>  #include <linux/ioctl.h>  #include <sys/stat.h>  #include <linux/types.h>  #include <linux/spi/spidev.h>  #define ARRAY\_SIZE(a) (sizeof(a) / sizeof((a)[0]))  static void pabort(const char \*s)  {  perror(s);  abort();  }  static const char \*device = "/dev/spidev0.0";  static uint32\_t mode;  static uint8\_t bits = 8;  static char \*input\_file;  static char \*output\_file;  static uint32\_t speed = 500000;  static uint16\_t delay;  static int verbose;  uint8\_t default\_tx[] = {  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,  0x40, 0x00, 0x00, 0x00, 0x00, 0x95,  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,  0xF0, 0x0D,  };  uint8\_t default\_rx[ARRAY\_SIZE(default\_tx)] = {0, };  char \*input\_tx;  static void hex\_dump(const void \*src, size\_t length, size\_t line\_size,  char \*prefix)  {  int i = 0;  const unsigned char \*address = src;  const unsigned char \*line = address;  unsigned char c;  printf("%s | ", prefix);  while (length-- > 0) {  printf("%02X ", \*address++);  if (!(++i % line\_size) || (length == 0 && i % line\_size)) {  if (length == 0) {  while (i++ % line\_size)  printf("\_\_ ");  }  printf(" | "); /\* right close \*/  while (line < address) {  c = \*line++;  printf("%c", (c < 33 || c == 255) ? 0x2E : c);  }  printf("\n");  if (length > 0)  printf("%s | ", prefix);  }  }  }  /\*  \* Unescape - process hexadecimal escape character  \* converts shell input "\x23" -> 0x23  \*/  static int unescape(char \*\_dst, char \*\_src, size\_t len)  {  int ret = 0;  int match;  char \*src = \_src;  char \*dst = \_dst;  unsigned int ch;  while (\*src) {  if (\*src == '\\' && \*(src+1) == 'x') {  match = sscanf(src + 2, "%2x", &ch);  if (!match)  pabort("malformed input string");  src += 4;  \*dst++ = (unsigned char)ch;  } else {  \*dst++ = \*src++;  }  ret++;  }  return ret;  }  static void transfer(int fd, uint8\_t const \*tx, uint8\_t const \*rx, size\_t len)  {  int ret;  int out\_fd;  struct spi\_ioc\_transfer tr = {  .tx\_buf = (unsigned long)tx,  .rx\_buf = (unsigned long)rx,  .len = len,  .delay\_usecs = delay,  .speed\_hz = speed,  .bits\_per\_word = bits,  };  if (mode & SPI\_TX\_QUAD)  tr.tx\_nbits = 4;  else if (mode & SPI\_TX\_DUAL)  tr.tx\_nbits = 2;  if (mode & SPI\_RX\_QUAD)  tr.rx\_nbits = 4;  else if (mode & SPI\_RX\_DUAL)  tr.rx\_nbits = 2;  if (!(mode & SPI\_LOOP)) {  if (mode & (SPI\_TX\_QUAD | SPI\_TX\_DUAL))  tr.rx\_buf = 0;  else if (mode & (SPI\_RX\_QUAD | SPI\_RX\_DUAL))  tr.tx\_buf = 0;  }  ret = ioctl(fd, SPI\_IOC\_MESSAGE(1), &tr);  if (ret < 1)  pabort("can't send spi message");  if (verbose)  hex\_dump(tx, len, 32, "TX");  if (output\_file) {  out\_fd = open(output\_file, O\_WRONLY | O\_CREAT | O\_TRUNC, 0666);  if (out\_fd < 0)  pabort("could not open output file");  ret = write(out\_fd, rx, len);  if (ret != len)  pabort("not all bytes written to output file");  close(out\_fd);  }  if (verbose || !output\_file)  hex\_dump(rx, len, 32, "RX");  }  static void print\_usage(const char \*prog)  {  printf("Usage: %s [-DsbdlHOLC3]\n", prog);  puts(" -D --device device to use (default /dev/spidev0.0)\n"  " -s --speed max speed (Hz)\n"  " -d --delay delay (usec)\n"  " -b --bpw bits per word\n"  " -i --input input data from a file (e.g. \"test.bin\")\n"  " -o --output output data to a file (e.g. \"results.bin\")\n"  " -l --loop loopback\n"  " -H --cpha clock phase\n"  " -O --cpol clock polarity\n"  " -L --lsb least significant bit first\n"  " -C --cs-high chip select active high\n"  " -3 --3wire SI/SO signals shared\n"  " -v --verbose Verbose (show tx buffer)\n"  " -p Send data (e.g. \"1234\\xde\\xad\")\n"  " -N --no-cs no chip select\n"  " -R --ready slave pulls low to pause\n"  " -2 --dual dual transfer\n"  " -4 --quad quad transfer\n");  exit(1);  }  static void parse\_opts(int argc, char \*argv[])  {  while (1) {  static const struct option lopts[] = {  { "device", 1, 0, 'D' },  { "speed", 1, 0, 's' },  { "delay", 1, 0, 'd' },  { "bpw", 1, 0, 'b' },  { "input", 1, 0, 'i' },  { "output", 1, 0, 'o' },  { "loop", 0, 0, 'l' },  { "cpha", 0, 0, 'H' },  { "cpol", 0, 0, 'O' },  { "lsb", 0, 0, 'L' },  { "cs-high", 0, 0, 'C' },  { "3wire", 0, 0, '3' },  { "no-cs", 0, 0, 'N' },  { "ready", 0, 0, 'R' },  { "dual", 0, 0, '2' },  { "verbose", 0, 0, 'v' },  { "quad", 0, 0, '4' },  { NULL, 0, 0, 0 },  };  int c;  c = getopt\_long(argc, argv, "D:s:d:b:i:o:lHOLC3NR24p:v",  lopts, NULL);  if (c == -1)  break;  switch (c) {  case 'D':  device = optarg;  break;  case 's':  speed = atoi(optarg);  break;  case 'd':  delay = atoi(optarg);  break;  case 'b':  bits = atoi(optarg);  break;  case 'i':  input\_file = optarg;  break;  case 'o':  output\_file = optarg;  break;  case 'l':  mode |= SPI\_LOOP;  break;  case 'H':  mode |= SPI\_CPHA;  break;  case 'O':  mode |= SPI\_CPOL;  break;  case 'L':  mode |= SPI\_LSB\_FIRST;  break;  case 'C':  mode |= SPI\_CS\_HIGH;  break;  case '3':  mode |= SPI\_3WIRE;  break;  case 'N':  mode |= SPI\_NO\_CS;  break;  case 'v':  verbose = 1;  break;  case 'R':  mode |= SPI\_READY;  break;  case 'p':  input\_tx = optarg;  break;  case '2':  mode |= SPI\_TX\_DUAL;  break;  case '4':  mode |= SPI\_TX\_QUAD;  break;  default:  print\_usage(argv[0]);  break;  }  }  if (mode & SPI\_LOOP) {  if (mode & SPI\_TX\_DUAL)  mode |= SPI\_RX\_DUAL;  if (mode & SPI\_TX\_QUAD)  mode |= SPI\_RX\_QUAD;  }  }  static void transfer\_escaped\_string(int fd, char \*str)  {  size\_t size = strlen(str);  uint8\_t \*tx;  uint8\_t \*rx;  tx = malloc(size);  if (!tx)  pabort("can't allocate tx buffer");  rx = malloc(size);  if (!rx)  pabort("can't allocate rx buffer");  size = unescape((char \*)tx, str, size);  transfer(fd, tx, rx, size);  free(rx);  free(tx);  }  static void transfer\_file(int fd, char \*filename)  {  ssize\_t bytes;  struct stat sb;  int tx\_fd;  uint8\_t \*tx;  uint8\_t \*rx;  if (stat(filename, &sb) == -1)  pabort("can't stat input file");  tx\_fd = open(filename, O\_RDONLY);  if (tx\_fd < 0)  pabort("can't open input file");  tx = malloc(sb.st\_size);  if (!tx)  pabort("can't allocate tx buffer");  rx = malloc(sb.st\_size);  if (!rx)  pabort("can't allocate rx buffer");  bytes = read(tx\_fd, tx, sb.st\_size);  if (bytes != sb.st\_size)  pabort("failed to read input file");  transfer(fd, tx, rx, sb.st\_size);  free(rx);  free(tx);  close(tx\_fd);  }  int main(int argc, char \*argv[])  {  int ret = 0;  int fd;  parse\_opts(argc, argv);  fd = open(device, O\_RDWR);  if (fd < 0)  pabort("can't open device");  /\*  \* spi mode  \*/  ret = ioctl(fd, SPI\_IOC\_WR\_MODE, &mode);  if (ret == -1)  pabort("can't set spi mode");  ret = ioctl(fd, SPI\_IOC\_RD\_MODE, &mode);  if (ret == -1)  pabort("can't get spi mode");  /\*  \* bits per word  \*/  ret = ioctl(fd, SPI\_IOC\_WR\_BITS\_PER\_WORD, &bits);  if (ret == -1)  pabort("can't set bits per word");  ret = ioctl(fd, SPI\_IOC\_RD\_BITS\_PER\_WORD, &bits);  if (ret == -1)  pabort("can't get bits per word");  /\*  \* max speed hz  \*/  ret = ioctl(fd, SPI\_IOC\_WR\_MAX\_SPEED\_HZ, &speed);  if (ret == -1)  pabort("can't set max speed hz");  ret = ioctl(fd, SPI\_IOC\_RD\_MAX\_SPEED\_HZ, &speed);  if (ret == -1)  pabort("can't get max speed hz");  printf("spi mode: 0x%x\n", mode);  printf("bits per word: %d\n", bits);  printf("max speed: %d Hz (%d KHz)\n", speed, speed/1000);  if (input\_tx && input\_file)  pabort("only one of -p and --input may be selected");  if (input\_tx)  transfer\_escaped\_string(fd, input\_tx);  else if (input\_file)  transfer\_file(fd, input\_file);  else  transfer(fd, default\_tx, default\_rx, sizeof(default\_tx));  close(fd);  return ret;  } |
| * Loopback speed test ()MOSI, MISO0      * Program option |