

1. Appendix 1 – Code Examples – Push Buttons

1.1. Example for communicating to Push Buttons, for C:

```
// test program to read state of buttons on 4D Systems 4DPi displays

#include <stdio.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>

#define LCD4DPI_GET_KEYS _IOR('K', 1, unsigned char *)

void print_keys(int fd)
{
    unsigned char keys;

    if (ioctl(fd, LCD4DPI_GET_KEYS, &keys) == -1)
    {
        perror("_apps ioctl get");
    }
    else
    {
        printf("Keys : %2x\n", keys);
    }
}

int main(int argc, char *argv[])
{
    char *file_name = "/dev/fb1";
    int fd;

    fd = open(file_name, O_RDWR);
    if (fd == -1)
    {
        perror("_apps open");
        return 2;
    }

    print_keys(fd);
    printf("Ioctl Number: (dec)%d (hex)%x\n", LCD4DPI_GET_KEYS, LCD4DPI_GET_KEYS);

    close (fd);
    return 0;
}
```

1.2. Example for communicating to Push Buttons, for Python:

```
#!/usr/bin/python
import array, fcntl
from time import sleep
# test program to read state of buttons on 4D Systems 4DPi displays

#LCD4DPI_GET_KEYS = -2147202303

_IOC_NRBITS = 8
_IOC_TYPEBITS = 8
_IOC_SIZEBITS = 14
_IOC_DIRBITS = 2
_IOC_DIRMASK = (1 << _IOC_DIRBITS) - 1
_IOC_NRMASK = (1 << _IOC_NRBITS) - 1
_IOC_TYPMASK = (1 << _IOC_TYPEBITS) - 1
_IOC_NRSHIFT = 0
_IOC_TYPSHIFT = _IOC_NRSHIFT+_IOC_NRBITS
_IOC_SIZESHIFT = _IOC_TYPSHIFT+_IOC_TYPEBITS
_IOC_DIRSHIFT = _IOC_SIZESHIFT+_IOC_SIZEBITS
_IOC_NONE = 0
_IOC_WRITE = 1
_IOC_READ = 2

def _IOC(dir, type, nr, size):
# print 'dirshift {}, typeshift {}, nrshift {}, sizeshift
{}'.format(_IOC_DIRSHIFT, _IOC_TYPSHIFT, _IOC_NRSHIFT, _IOC_SIZESHIFT)
ioc = (dir << _IOC_DIRSHIFT) | (type << _IOC_TYPSHIFT) | (nr << _IOC_NRSHIFT)
| (size << _IOC_SIZESHIFT)
if ioc > 2147483647: ioc -= 4294967296
return ioc

#def _IO(type, nr):
# return _IOC(_IOC_NONE, type, nr, 0)
def _IOR(type,nr,size):
return _IOC(_IOC_READ, type, nr, size)
#def _IOW(type,nr,size):
# return _IOC(_IOC_WRITE, type, nr, sizeof(size))

LCD4DPI_GET_KEYS = _IOR(ord('K'), 1, 4)
buf = array.array('h', [0])

print 'Press Top & Bottom buttons simultaneously to exit'

with open('/dev/fb1', 'rw') as fd:

while True:
fcntl.ioctl(fd, LCD4DPI_GET_KEYS, buf, 1) # execute ioctl call to read the keys
keys = buf[0]

if not keys & 0b00001:
print "KEY1" ,
if not keys & 0b00010:
print "KEY2" ,
if not keys & 0b00100:
print "KEY3" ,
if not keys & 0b01000:
print "KEY4" ,
if not keys & 0b10000:
print "KEY5" ,

if keys != 0b11111:
print
if keys == 0b01110: # exit if top and bottom pressed
break

sleep(0.1)
```

1.3. Example for Shutdown and Reset buttons, for C

```
// test program to Shutdown or Restart Pi using buttons on 4D Systems 4DPi displays

#include <stdio.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>

#define LCD4DPI_GET_KEYS_IOR('K', 1, unsigned char *)

int get_keys(int fd, unsigned char *keys)
{
    if (ioctl(fd, LCD4DPI_GET_KEYS, keys) == -1)
    {
        perror("_apps ioctl get");
        return 1;
    }
    *keys &= 0b11111;
    return 0;
}

int main(int argc, char *argv[])
{
    char *file_name = "/dev/fb1";
    int fd;
    unsigned char key_status;

    fd = open(file_name, O_RDWR);
    if (fd == -1)
    {
        perror("_apps open");
        return 2;
    }

    key_status = 0b11111;
    while(key_status & 0b00001) // press key 1 to exit
    {
        if(get_keys(fd, &key_status) != 0)
            break;

        // printf("key_status: %x\n", key_status);

        if(!(key_status & 0b10000))
        {
            system("sudo shutdown -h now");
            break;
        }

        if(!(key_status & 0b01000))
        {
            system("sudo reboot");
            break;
        }

        sleep(0.1);
    }

    close(fd);
    return 0;
}
```

1.4. Example for Shutdown and Reset buttons, for Python

```
#!/usr/bin/python
import array, fcntl, os
from time import sleep
# test program to Shutdown or Restart Pi using buttons on 4D Systems 4DPi displays

LCD4DPI_GET_KEYS = -2147202303

_IOC_NRBITS = 8
_IOC_TYPEBITS = 8
_IOC_SIZEBITS = 14
_IOC_DIRBITS = 2

_IOC_DIRMASK = (1 << _IOC_DIRBITS) - 1
_IOC_NRMASK = (1 << _IOC_NRBITS) - 1
_IOC_TYPMASK = (1 << _IOC_TYPEBITS) - 1

_IOC_NRSHIFT = 0
_IOC_TYPSHIFT = _IOC_NRSHIFT + _IOC_NRBITS
_IOC_SIZESHIFT = _IOC_TYPSHIFT + _IOC_TYPEBITS
_IOC_DIRSHIFT = _IOC_SIZESHIFT + _IOC_SIZEBITS

_IOC_NONE = 0
_IOC_WRITE = 1
_IOC_READ = 2

def _IOC(dir, type, nr, size):
# print 'dirshift {}, typeshift {}, nrshift {}, sizeshift
{}'.format(_IOC_DIRSHIFT, _IOC_TYPSHIFT, _IOC_NRSHIFT, _IOC_SIZESHIFT)
ioc = (dir << _IOC_DIRSHIFT) | (type << _IOC_TYPSHIFT) | (nr << _IOC_NRSHIFT)
| (size << _IOC_SIZESHIFT)
if ioc > 2147483647: ioc -= 4294967296
return ioc
#def _IO(type, nr):
# return _IOC(_IOC_NONE, type, nr, 0)
def _IOR(type, nr, size):
return _IOC(_IOC_READ, type, nr, size)
#def _IOW(type, nr, size):
# return _IOC(_IOC_WRITE, type, nr, sizeof(size))

LCD4DPI_GET_KEYS = _IOR(ord('K'), 1, 4)
#print 'ssid {} {:12} {:0>8x} {:0>32b}'.format(ssd1289, hex(ssd1289), ssd1289,
ssid1289)
buf = array.array('h', [0])

with open('/dev/fb1', 'rw') as fd:

while True:
fcntl.ioctl(fd, LCD4DPI_GET_KEYS, buf, 1) # execute ioctl call to read the keys
keys = buf[0]

if not keys & 0b00001:
break

if not keys & 0b10000:
os.system("sudo shutdown -h now")
break

if not keys & 0b01000:
os.system("sudo reboot")
break;

sleep(0.1)
```