PTZ Driver API Manual

UDP Technology

Table of Contents

1. Introduction

1.1. Overview

PTZ Drive API manual allows you to make your own driver of PTZ device.

2. PTZ Driver Structure

2.1. Range definition

VERSION
It defines the version of ptzs.
If the version of ptzs is higher than ptzlib, the file can't be loaded.
e.g. version=1.0.0.

PROTOCOL.NAME
Name of the protocol

PROTOCOL.VERSION
Version of protocol

PROTOCOL.DESCRIPTION
Description of protocol

PROTOCOL.PRESET_RANGE
Preset range setting
If this range is not defined, relevant preset commands fail to work.

PROTOCOL.TOUR_RANGE
Tour range setting
If this range is not defined, relevant tour commands fail to work.

PROTOCOL.SCAN_RANGE
Scan range setting
If this range is not defined, relevant scan commands fail to work.

PROTOCOL.PATTERN_RANGE
Pattern range setting
If this range is not defined, relevant pattern commands fail to work.

PROTOCOL.PAN_SPEED
Pan speed range setting
The value (0~100) is normalized into this pan speed range. If you have not designate the speed range of pan, tilt and zoom, the default is -180 ~ 180.

PROTOCO.TILT_SPPED
Tilt speed range setting
The value (0~100) is normalized into this tilt speed range. If you have not designate the speed range of pan, tilt and zoom, the default is -180 ~ 180.

PROTOCOL.ZOOM_SPEED
Zoom speed range setting
The value (0~100) is normalized into this zoom speed range. If you have not designate the speed range of pan, tilt and zoom, the default is -180 ~ 180.

PROTOCOL.PAN_RANGE
When you command set_position, this value defines the range of pan angle. If this range is not defined, set_pos command fails to work because it compares the values on the basis of this range.

PROTOCOL.TILT_RANGE
When you command set_position, this value defines the range of tilt angle. If this range is not defined, set_pos command fails to work because it compares the values on the basis of this range.

PROTOCOL.PAN_REAL
When set_position and get_position is commanded, ptz lib match and convert the pan values from CGI into the real value for device. PAN_REAL command defines the range of pan movement in real.

PROTOCOL.TILT_REAL
When set_position and get_position is command, ptz lib match and convert the tilt values from CGI into the real value for device. TILT_REAL command defines the range of tilt movement in real.

PROTOCOL.ZOOM_REAL
When set_position and get_position is command, ptz lib match and convert the zoom values from CGI into the real value for device. ZOOM_REAL command defines the range of zoom movement in real.

PROTOCOL.PAN_AREA
It defines the movement amount of pan when CENTER is commanded. If you don’t define this range then, CENTER command will get error. Driver gets the center value

For your information, CENTER command is:
If users appoint a specific spot on Active X screen randomly, it enables the spot to be located at the center of the monitor screen.

PROTOCOL.TILT_AREA
It defines the movement amount of tilt when CENTER is commanded. If you don’t define this range then, CENTER command will get error.

STX
This is the value for PTZ driver to use internally not for library.

ETX
This is the value for PTZ driver to use internally not for library.

2.2. Supported commands by Lua

This is the basic library which Lua provides and refer to the homepage for more information. (http://www.lua.org/docs.html)

Below items are helper function PTZ Lib supports.

integer **bit_and**(v1, v2)
Get the value by Bitwise AND operation of *v1* and *v2*.

integer **bit_or**(v1, [v2…vn])
Get the value by Bitwise OR operation of *v1* ~ *vn*.

integer **bit_xor**(v1, v2)
Get the value by Bitwise XOR operation of *v1* and *v2*.

integer **lsb**(bits, val)
Get only the *bits* of *val* of low positioned bits.

integer **msb**(bits, val)
Get only the *bits* of *val* of top positioned bits.

integer **lb**(val)
Get only the low positioned 1 byte of *val*. Max *val* is restricted to 2 byte.

integer **hb**(val)
Get only the top positioned 1 byte of *val*. Max *val* is restricted to 2 byte.

integer **shift**(bits, val)
Shift the *val* with bits. If the *bits* is positive integer, it moves as *rshift* or *lshift*.

integer **abs**(val)
Get the absolute value of *val*.

integer **isplus**(val)
If *val* is positive integer, it returns 1. If negative integer or 0, it returns 0.

integer **isminus**(val)
If val is negative integer, it returns 1. If positive integer or 0, it returns 0.

integer **if3**(predict, value_if_true, value_if_false)
If the *prodict* is true, it returns *value_if_true* or *value_if_false*

void **write_serial**(packet)
Send the *packet* through serial connection. The serial comm. Port is designated according to the value of set comm. port. (Used in ptz lib)

packet **read_serial**(read_size, timeout_ms)
Get the *read_size* according to the *timeout_ms*. If the timeout occurs, it gets only that red size of packet.

void **trace**(message)
Output the *message* for the debugging message of *ptzscgid*. This is used for the purpose of debugging when developers make PTZ driver. (using fprintf (stderr, message)

void **tr**(msg, [format…])
Output the debugging message with trace function. It supports format transformation such as *printf*.

void **sleep**(timeout_ms)
Wait for *timeout_ms*

2.3. How to debug

1. Access via telnet and log in for an administrator account.

2. Kill the Daemon with *killall ptzscgid; ptzscgid &* and enable Daemon again.

3. You can see the messages such as function call status or packet information are printed from Daemon. You can also use the *tr()* or *trace()* and let the debugging message are printed.

3. Example code

The full example Pelco PTZ driver is provided for your convenience in this section. But this details only the commands for Pelco PTZ driver and if you deal with another maker's PTZ, you need to refer to the full manual and you are asked to refer to this example code just for reference purpose.

```
version = "1.0.0"
```

It defines the version of PTZ driver you are establishing now.

```
PROTOCOL.NAME = 'Pelco-D'
PROTOCOL.VERSION = '1.0.2.0'
PROTOCOL.DESCRIPTION = "Pelco-D protocol"
```

Protocol name, version and description are defined.

```
PROTOCOL.PRESET_RANGE={0,127}
PROTOCOL.TOUR_RANGE={0,16}
PROTOCOL.SCAN_RANGE={0,8}
PROTOCOL.PATTERN_RANGE={0,8}
```

The range of preset, tour, scan and pattern are defined.

```
PROTOCOL.PAN_SPEED = {0, 0x3e}
PROTOCOL.TILT_SPEED = {0, 0x3f}
PROTOCOL.ZOOM_SPEED = {0, 0x3f}
```

The range of pan, tilt and zoom speed

```
PROTOCOL.PAN_RANGE={0, 359}
PROTOCOL.TILT_RANGE={0, 359}
```

The range of angle users can define when moves on absolute point. If the range is not defined, the default is -180 ~ +180.

```
PROTOCOL.PAN_REAL = {0, 35999}
PROTOCOL.TILT_REAL = {0, 35999}
PROTOCOL.ZOOM_REAL = {0, 65535}
```

The range of angle users can define when moves on current point of pan, tilt and zoom.

```
Pelco-d protocol

0xFF ID CMD1 CMD2 DATA1 DATA2 CRC

CRC = sum (ID ~ DATA2) & 0xff
```

* Below commands are not supported for Pelco-d example.

presetmenu
tourmenu
patternmenu
scanmenu

menuon
menuoff
menuclear
menuenter

tour
scan

* setpos, getpos commands support pan, tilt and zoom separately.

```
SOH = 0xff

function get_crc()
  local val = 0
  for i = 2, #PK do
    val = val + PK[i]
end

  return lb(val)
end

function CMD_PRE_ACTION()
end
```

UTIL FUNCTION
#PK means the length of PK

```
function CMD_MOVE_PANTILTZOOM(pan, tilt, zoom)
    PK[1] = SOH
    PK[2] = lb(DEVICE.ADDR)
    PK[3] = 0
```

PK[ ] table is the reserved buffer required to bring the packet via device.
PK[2] value is the sub byte of address. For example, Lb (V) means getting the sub byte of V. for more information, refer to the Supported Commands by Lua on page 6 of this manual.

```
    PK[4] = bit_or(shift(1, isplus(pan)), shift(2, isminus(pan)) )
    PK[4] = bit_or(PK[4], shift(3, isplus(tilt)), shift(4, isminus(tilt)) )
    PK[4] = bit_or(PK[4], shift(6, isminus(zoom)), shift(5, isplus(zoom)) )
```

Create the control byte with pan value
In case of pelco-d, the bit of control-byte controls the direction.

```
    PK[5] = abs(pan)
    PK[6] = abs(tilt)
```

```
    PK[7] = get_crc()
end

function CMD_CALL_PRESET(preset)
    PK[1] = SOH
    PK[2] = lb(DEVICE.ADDR)
    PK[3] = 0
    PK[4] = 7
    PK[5] = 0
    PK[6] = preset
    PK[7] = get_crc()
end
```

```
function CMD_REMOVE_PRESET(preset)
    PK[1] = SOH
    PK[2] = lb(DEVICE.ADDR)
    PK[3] = 0
    PK[4] = 5
    PK[5] = 0
    PK[6] = preset
    PK[7] = get_crc()
end
```

```
function CMD_STORE_PRESET(preset)
    PK[1] = SOH
    PK[2] = lb(DEVICE.ADDR)
    PK[3] = 0
    PK[4] = 3
    PK[5] = 0
    PK[6] = preset
    PK[7] = get_crc()
end
```

```
function CMD_CALL_PATTERN(pattern)
    PK[1] = SOH
    PK[2] = lb(DEVICE.ADDR)
    PK[3] = 0
    PK[4] = 0x23
    PK[5] = 0
    PK[6] = pattern
    PK[7] = get_crc()
end
```

```
function CMD_MOVE_FOCUS(focus)
    PK[1] = SOH
    PK[2] = lb(DEVICE.ADDR)
    PK[3] = shift(0, isminus(focus))
    PK[4] = shift(7, isplus(focus))
    PK[5] = 0
```

```
    PK[6] = 0
    PK[7] = get_crc()
end
```

PK[3] = shift(0, isminus(focus)) –0x01 : focus near
PK[4] = shift(7,isplus(focus)) – 0x80 : focus far

```
function act_iris(iris)
    PK[1] = SOH
    PK[2] = lb(DEVICE.ADDR)
    PK[3] = iris
    PK[4] = 0
    PK[5] = 0
    PK[6] = 0
    PK[7] = get_crc()
end
```

```
function CMD_STEP_IRIS(iris)
    if iris > 0 then
        act_iris(2)
    else
        act_iris(4)
    end
end
```

act_iris(2) means open
act_iris(4) means close

```
function CMD_MENU_TOGGLE()
    PK[1] = SOH
    PK[2] = lb(DEVICE.ADDR)
    PK[3] = 0
    PK[4] = 0x7
    PK[5] = 0
    PK[6] = 95
    PK[7] = get_crc()
end
```

Define the speed of pan and tilt. Zoom is not defined with speed.

```
function CMD_MENU_ON()
    CMD_MENU_TOGGLE()
end
```

menu_on/off command allows the emulation by toggle.

```
function CMD_MENU_OFF()
    CMD_MENU_TOGGLE()
end
```

```
function CMD_MENU_LEFT()
```

```
   CMD_MOVE_PANTILTZOOM(-100, 0, 0)
   write_serial(PK)
   sleep(100)
   PK = {}
   CMD_MOVE_PANTILTZOOM(0, 0, 0)
end
```

For some PTZ cameras including Pelco_D, *pan_left* command is used in replace of the *menu_leftt* which is normally used for other cameras. And after this command, you need to wait for a sec with sleep( ) and command Stop as above example. And it's same for *menu_right, menu_up and menu_down* as well.

```
function CMD_MENU_RIGHT()
   CMD_MOVE_PANTILTZOOM(100, 0, 0)
   write_serial(PK)
   sleep(100)
   PK = {}
   CMD_MOVE_PANTILTZOOM(0, 0, 0)
end
```

```
function CMD_MENU_UP()
   CMD_MOVE_PANTILTZOOM(0, 100, 0)
   write_serial(PK)
   sleep(100)
   PK = {}
   CMD_MOVE_PANTILTZOOM(0, 0, 0)
end
```

```
function CMD_MENU_DOWN()
   CMD_MOVE_PANTILTZOOM(0, -100, 0)
   write_serial(PK)
   sleep(100)
   PK = {}
   CMD_MOVE_PANTILTZOOM(0, 0, 0)
end
```

```
function act_get_pos(cmd)
   PK[1] = SOH
   PK[2] = lb(DEVICE.ADDR)
   PK[3] = 0
   PK[4] = cmd
   PK[5] = 0
   PK[6] = 0
   PK[7] = get_crc()
   write_serial(PK)
end
```

Transfer the packet directly to the device .

```
function CMD_GET_POSITION()
```

```
    local buf
```

```
    act_get_pos(0x51)
    buf = read_serial(7, 100)
    if #buf < 7 then
        RESULT3(0,0,0)
        return
    end
    pan = bit_or(shift(8, buf[5]), buf[6])
```

This enables to get the pan position. Real_serial(7,100) means to wait for 7 bytes for 100ms.
#buf = the length of buf
RESULT3(0,0,0) is timeout.

```
    act_get_pos(0x53)
    buf = read_serial(7, 100)
    if #buf < 7 then
        RESULT3(0,0,0)
        return
    end
    tilt = bit_or(shift(8, buf[5]), buf[6])
```

This enables to get tilt position.

```
    act_get_pos(0x55)
    buf = read_serial(7, 100)
    if #buf < 7 then
        RESULT3(0,0,0)
        return
    end
    zoom = bit_or(shift(8, buf[5]), buf[6])
```

This enables to get zoom value.

```
    RESULT3(pan, tilt, zoom)
end
```

The function of RESULT3 is a utility function which let the PK table get the number.

```
function set_pos(cmd, val)
    PK[1] = SOH
    PK[2] = lb(DEVICE.ADDR)
    PK[3] = 0
    PK[4] = cmd
    PK[5] = hb(val)
    PK[6] = lb(val)
    PK[7] = get_crc()
end
```

```
function CMD_SET_POSITION(pan, tilt, zoom)
```

In case of Pelco-d protocol, SET_POSITION command can be used for pan, tilt and zoom position setting. Each command works separately.

```
    set_pos(0x4b, pan)
    write_serial(PK)

    set_pos(0x4d, tilt)
    write_serial(PK)

    set_pos(0x4f, zoom)
end
```

Revision History

Date	Revision	Description
2008-08-01	A	Created.