Writing Scripts for the Gentled CHDK2

Dave Mitchell – dave@zenoshrdlu.com

The Gentled CHDK2 lets you connect two servo channels to the USB port of a CHDK-enabled Canon camera. Scripts running in the camera can detect joystick movements on the two channels and carry out a variety of actions in response.

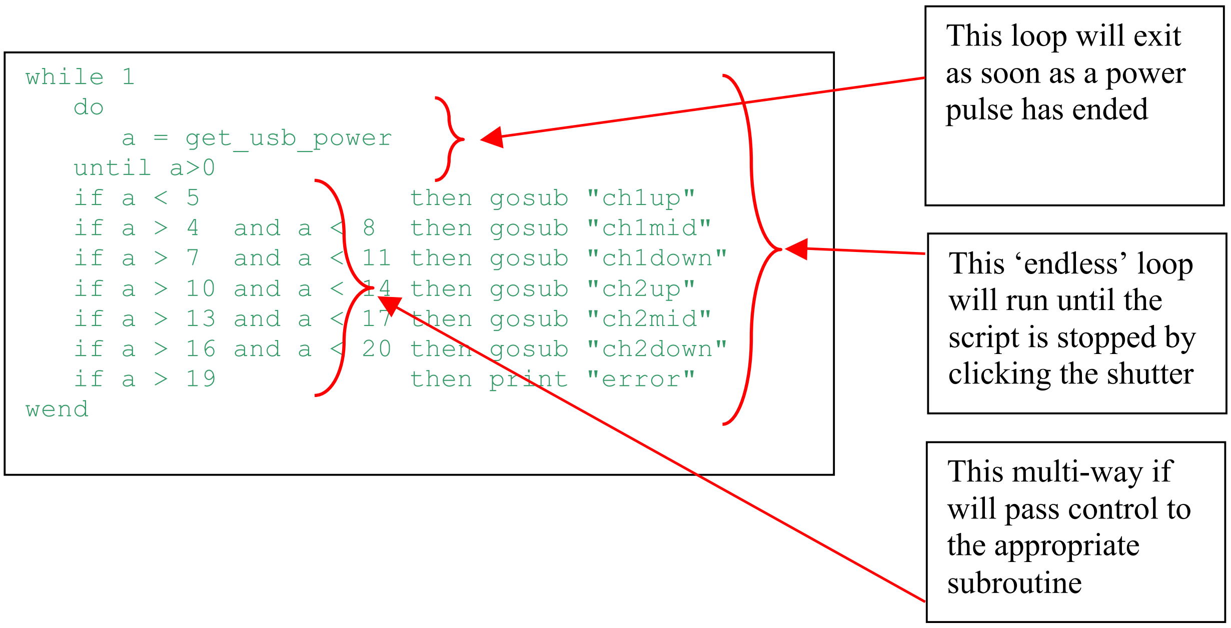
A CHDK script can detect when power is applied to the USB port and for how long that power is applied. The CHDK2 uses this to send power-on signals of six different durations[1]:

- a 30 ms signal when joystick 1 is moved up
- a 60ms signal when joystick 1 is centred
- a 90ms signal when joystick 1 is moved down
- a 120 ms signal when joystick 2 is moved up
- a 150ms signal when joystick 2 is centred
- a 180ms signal when joystick 2 is moved down

The **get_usb_power** CHDK script function returns the time power was applied in 10ms units, so an "up" movement of joystick 2 will return a value of 12 (or possibly 13 since it's not possible for the gentled to be exactly precise). By using this function a script can determine which joystick was moved and in which direction and use this to carry out an appropriate action.

Anatomy of a Script

Most of the sample scripts have the same basic structure. The main routine looks like this:



```
while 1
   do
      a = get_usb_power
   until a>0
   if a < 5                  then gosub "ch1up"
   if a > 4   and a < 8      then gosub "ch1mid"
   if a > 7   and a < 11     then gosub "ch1down"
   if a > 10  and a < 14     then gosub "ch2up"
   if a > 13  and a < 17     then gosub "ch2mid"
   if a > 16  and a < 20     then gosub "ch2down"
   if a > 19                 then print "error"
wend
```

This loop will exit as soon as a power pulse has ended

This 'endless' loop will run until the script is stopped by clicking the shutter

This multi-way if will pass control to the appropriate subroutine

This main routine is followed by six subroutines each of which will be called when a power pulse of the appropriate length is applied.

By writing appropriate subroutines you can attach any scriptable action to any of the 6 joystick positions. However, when assigning an action to the joystick **middle** position, bear in mind that it's not possible to move a joystick from the **up** position to the **down** position without going through the **middle** position. For this reason, most of the sample scripts do nothing when a joystick returns to the middle position.

The sample scripts deliberately have ***print*** statements in them so that when you can see what's supposed to happen when testing them at home.

[1] Actually there's a seventh signal, of duration 210ms, sent when the receiver loses touch with the transmitter.

Writing Scripts for the Gentled CHDK2

Dave Mitchell – dave@zenoshrdlu.com

The **CHDK2Tester.bas** Script

This simple script doesn't do anything useful – it's just designed to let you test that your Gentled CHDK2 is working correctly. When you run the script by clicking the shutter button it will run until you click the shutter button again, and while it's running you should see the camera's LCD display appropriate messages as the transmitter joysticks are moved.

This script should work on any CHDK-capable camera in more or less any mode.

The **SimpleZoomNShoot.bas** Script

This script is a simple elaboration of the test script, making joystick 1 *click the shutter* and joystick 2 *control the camera's zoom*. This script will work fine on A-series or SD/IXUS-series cameras which have 3x or 4x zooms with 8 or 14 zoom steps. Each joystick movement will move the zoom to the next step.

However, S-series cameras such as the S3 IS have more powerful 12x zooms with 128 steps, so although this script will work on such cameras, it will take **a lot** of joystick movements to do any useful zooming.

The **ZoomNShoot.bas** Script

This script should work on *all* CHDK-cameras that have zooms. It provides a run-time parameter, **c**, which if set to 0 (the default) will provide zooming for A- and SD/IXUS- cameras, while if it's set to 1 will work on S-series cameras. In either case moving the joystick will move the zoom a reasonable number of steps. The script uses the **set_zoom_rel** function to move the zoom by a number of steps, 2 steps for A and SD/IXUS cameras and 25 for the S-series.

Scripting Possibilities

There are many ways in which a CHDK script can exploit the six different signals the camera can be sent. For example, you can have joystick movements make exposure or aperture adjustments, perform a half or full shutter press, or switch to continuous mode and do some bracketing.

As an example, here's a pair of routines that make joystick 2 turn RAW capability ***on*** and ***off*** :

```
 :ch2up
   Print "set raw ON"
   Set_raw 1
   return

:ch2down
   Print "set raw OFF"
   Set_raw 0
   return
```

You don't have to stick to the basic logic of these simple scripts either. They all work in the same way, with the main routine waiting until power is applied to the USB port. If no power is currently being applied, the **get_usb_power** function returns a value of 0 immediately. That's why the sample scripts have this loop in them:

```
do
  a = get_usb_power
until a>0
```

The loop will execute many times until a joystick is moved and the CHDK2 sends a power pulse.

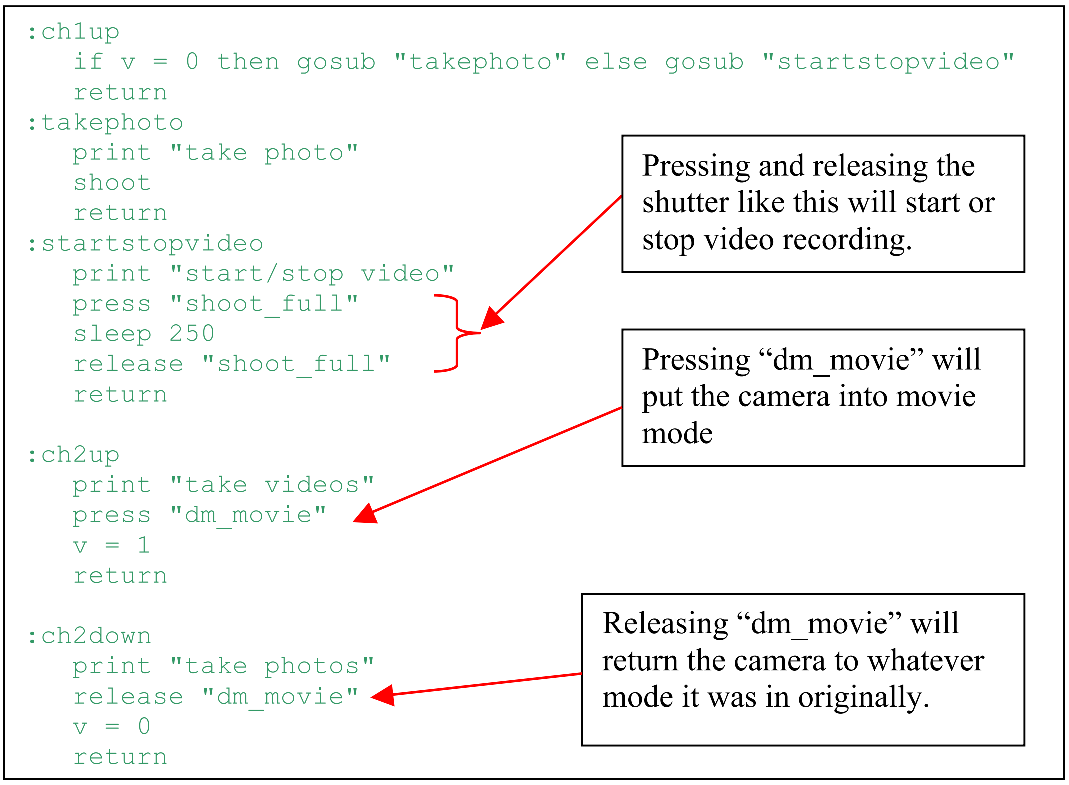
However, it’s possible to use the **get_usb_power** function is a slightly different way. The sample scripts don’t assign an action to the ch1down subroutine, but suppose we added one, like this:

```
:ch1down
do
   sleep 9000
   shoot
   a = get_usb_power
until a > 0
return
```

When joystick1 is moved down the camera will start doing “AutoKAP”, taking a photo every 9 seconds. And as long as no joystick is moved it will keep doing this (since **get_usb_power** will keep returning 0). But as soon as one of the joysticks is moved **get_usb**_power will return a non-zero value, the subroutine will return, and we’ll go back to the main script.

Utilising the Mode Dial Support Version of CHDK

There is a special version[2] of Allbest’s CHDK build for the A570 camera which lets scripts ‘turn’ the mode dial. This can be used to dynamically switch between taking photos and taking videos. Here’s the heart of the **MoviePhoto.bas** script which does this:



```
:ch1up
   if v = 0 then gosub "takephoto" else gosub "startstopvideo"
   return
:takephoto
   print "take photo"
   shoot
   return
:startstopvideo
   print "start/stop video"
   press "shoot_full"
   sleep 250
   release "shoot_full"
   return

:ch2up
   print "take videos"
   press "dm_movie"
   v = 1
   return

:ch2down
   print "take photos"
   release "dm_movie"
   v = 0
   return
```

Pressing and releasing the shutter like this will start or stop video recording.

Pressing “dm_movie” will put the camera into movie mode

Releasing “dm_movie” will return the camera to whatever mode it was in originally.

Note that it’s necessary to move joystick 1 up to **start** video recording and then again when you want to **stop**. If you switch back to taking photos without stopping video recording the recording will be stopped anyway.

Revised March 2010 to amend multi-way if logic.

[2] Available at http://www.zenoshrdlu.com/kapstuff/zchdk.html