#! /usr/bin/env python2

# Derived from camera1394 cfg

from dynamic\_reconfigure.parameter\_generator\_catkin import \*

RECONFIGURE\_CLOSE = 3 # Parameters that need a sensor to be stopped completely when changed

RECONFIGURE\_STOP = 1 # Parameters that need a sensor to stop streaming when changed

RECONFIGURE\_RUNNING = 0 # Parameters that can be changed while a sensor is streaming

gen = ParameterGenerator()

# Name, Type, Reconfiguration level, Description, Default, Min, Max

gen.add("vendor", str\_t, RECONFIGURE\_CLOSE,

"Vendor ID, hex digits (use camera of any vendor if null).",

"")

gen.add("product", str\_t, RECONFIGURE\_CLOSE,

"Product ID, hex digits (use camera of any model if null).",

"")

gen.add("serial", str\_t, RECONFIGURE\_CLOSE,

"Serial number, arbitrary string (use camera of any serial number if null).",

"")

gen.add("index", int\_t, RECONFIGURE\_CLOSE,

"Index into the list of cameras that match the above parameters.",

0, 0)

gen.add("width", int\_t, RECONFIGURE\_CLOSE,

"Image width.", 640, 0)

gen.add("height", int\_t, RECONFIGURE\_CLOSE,

"Image height.", 480, 0)

video\_modes = gen.enum([gen.const("uncompressed", str\_t, "uncompressed", "Use any uncompressed format"),

gen.const("compressed", str\_t, "compressed", "User any compressed format"),

gen.const("yuyv", str\_t, "yuyv", "YUYV"),

gen.const("uyvy", str\_t, "uyvy", "UYVY"),

gen.const("rgb", str\_t, "rgb", "RGB"),

gen.const("bgr", str\_t, "bgr", "BGR"),

gen.const("mjpeg", str\_t, "mjpeg", "MJPEG"),

gen.const("gray8", str\_t, "gray8", "gray8")],

"Video stream format")

gen.add("video\_mode", str\_t, RECONFIGURE\_CLOSE,

"Format of video stream from camera.", "uncompressed",

edit\_method = video\_modes)

gen.add("frame\_rate", double\_t, RECONFIGURE\_CLOSE,

"Camera speed, frames per second.", 15.0, 0.1, 1000.0)

timestamp\_methods = gen.enum([gen.const("PubTime", str\_t, "pub", "Time of publication"),

gen.const("FrameStartTime", str\_t, "start", "Time when raw frame capture began"),

gen.const("FrameStopTime", str\_t, "stop", "Time when raw frame capture ended"),

gen.const("HostReceiptTime", str\_t, "hostrcpt", "Time when camera-to-host transfer completed")],

"Methods for determining the timestamp")

gen.add("timestamp\_method", str\_t, RECONFIGURE\_CLOSE,

"Method for determining the timestamp.", "start",

edit\_method = timestamp\_methods)

gen.add("frame\_id", str\_t, RECONFIGURE\_RUNNING,

"ROS tf frame of reference, resolved with tf\_prefix unless absolute.",

"camera")

gen.add("camera\_info\_url", str\_t, RECONFIGURE\_RUNNING,

"Path to camera calibration file.", "")

# Camera Terminal controls

scanning\_modes = gen.enum([gen.const("Interlaced", int\_t, 0, ""),

gen.const("Progressive", int\_t, 1, "")],

"Scanning modes")

gen.add("scanning\_mode", int\_t, RECONFIGURE\_RUNNING,

"Scanning mode.", 0, 0, 1,

edit\_method = scanning\_modes)

auto\_exposure\_modes = gen.enum([gen.const("Manual", int\_t, 0, "Manual exposure, manual iris"),

gen.const("Auto", int\_t, 1, "Auto exposure, auto iris"),

gen.const("Shutter\_Priority", int\_t, 2, "manual exposure, auto iris"),

gen.const("Aperture\_Priority", int\_t, 3, "auto exposure, manual iris")],

"Auto-exposure modes")

gen.add("auto\_exposure", int\_t, RECONFIGURE\_RUNNING,

"Auto exposure mode.",

3, 0, 3, edit\_method = auto\_exposure\_modes)

gen.add("auto\_exposure\_priority", int\_t, RECONFIGURE\_RUNNING,

"In auto mode or shutter priority mode, allow the device to vary frame rate.",

0, 0, 1)

gen.add("exposure\_absolute", double\_t, RECONFIGURE\_RUNNING,

"Length of exposure, seconds.", 0., 0.0001, 10.0)

# TODO: relative exposure time

gen.add("iris\_absolute", double\_t, RECONFIGURE\_RUNNING,

"Aperture, f.", 0., 0., 655.35)

# TODO: relative iris

gen.add("auto\_focus", bool\_t, RECONFIGURE\_RUNNING,

"Maintain focus automatically.", True)

gen.add("focus\_absolute", int\_t, RECONFIGURE\_RUNNING,

"Absolute focal distance, millimeters.", 0, 0, 65536)

# TODO: relative focus

# TODO: zoom

gen.add("pan\_absolute", int\_t, RECONFIGURE\_RUNNING,

"Pan (clockwise), arc seconds.", 0, -180\*3600, 180\*3600)

gen.add("tilt\_absolute", int\_t, RECONFIGURE\_RUNNING,

"Tilt (up), arc seconds.", 0, -180\*3600, 180\*3600)

# TODO: relative pan/tilt

gen.add("roll\_absolute", int\_t, RECONFIGURE\_RUNNING,

"Roll (clockwise), degrees.", 0, -180, 180)

# TODO: relative roll

gen.add("privacy", bool\_t, RECONFIGURE\_RUNNING,

"Image capture disabled.", False)

# Processing Unit controls

gen.add("backlight\_compensation", int\_t, RECONFIGURE\_RUNNING,

"Backlight compensation, device-dependent (zero for none, increasing compensation above zero).",

0, 0, 65536)

gen.add("brightness", int\_t, RECONFIGURE\_RUNNING,

"Brightness, device dependent.", 0, -32768, 32767)

gen.add("contrast", int\_t, RECONFIGURE\_RUNNING,

"Contrast, device dependent.", 0, -32768, 32767)

gen.add("gain", int\_t, RECONFIGURE\_RUNNING,

"Gain, device dependent.", 0, 0, 65536)

power\_line\_frequency\_modes = gen.enum([gen.const("Disabled", int\_t, 0, "Disabled"),

gen.const("Freq\_50", int\_t, 1, "50 Hz"),

gen.const("Freq\_60", int\_t, 1, "60 Hz")],

"Power line frequency modes")

gen.add("power\_line\_frequency", int\_t, RECONFIGURE\_RUNNING,

"Power line frequency anti-flicker processing.",

0, 0, 2,

edit\_method = power\_line\_frequency\_modes)

gen.add("auto\_hue", bool\_t, RECONFIGURE\_RUNNING,

"Automatic hue control.", False)

gen.add("hue", double\_t, RECONFIGURE\_RUNNING,

"Hue, degrees.", 0., -180., 180.)

gen.add("saturation", int\_t, RECONFIGURE\_RUNNING,

"Saturation, device dependent (zero for grayscale).", 0, 0, 65536)

gen.add("sharpness", int\_t, RECONFIGURE\_RUNNING,

"Image sharpness, device dependent.",

0, 0, 65536)

# TODO: check range definition

gen.add("gamma", double\_t, RECONFIGURE\_RUNNING,

"Gamma.", 1.0, 0.01, 5.0)

gen.add("auto\_white\_balance", bool\_t, RECONFIGURE\_RUNNING,

"Automatic white balance.", False)

gen.add("white\_balance\_temperature", int\_t, RECONFIGURE\_RUNNING,

"White balance temperature, degrees.", 0, 0, 65536)

gen.add("white\_balance\_BU", double\_t, RECONFIGURE\_RUNNING,

"Blue or U component of white balance, device-dependent.",

0, 0, 65536)

gen.add("white\_balance\_RV", double\_t, RECONFIGURE\_RUNNING,

"Red or V component of white balance, device-dependent.",

0, 0, 65536)

# TODO: digital multiplier {,limit}

# TODO: analog video standard, analog video lock

exit(gen.generate('libuvc\_camera', "libuvc\_camera", "UVCCamera"))