Inhaltsverzeichnis

Multi-Analog Output with a Raspberry Pico

Out of the need for an unexpensive Multi- Analog- Output Board this idea came up:

A Raspi Pico receives a serial string via serial line and transfers the numeric values to Microchip MCP4802, which are connected in parallel to an SPI bus with multiple CS (Chip Select) signals.

This as a proof of concept to mayby later cascade these Picos like neopixels with a serial chain connection protocol.

Setup

The pico is set up as descripted in USB Keyboard and Gamepad Emulation for an old analog Joystick

The Software

```
import sys
import time
import board
import busio
import digitalio
from adafruit_bus_device.spi_device import SPIDevice

# list of available chip select signals
chip_select_ports=[board.GP13]

# define the conversion values to write values to outputs
# value max , output mask multiplier, output mask flags, byte size, endian
coding
output_channels =[
    {'max':0xFF , 'mult':16 , 'flags' : 0x1000 , 'size' : 2 , 'endian' :
'big' },
    {'max':0xFF , 'mult':16 , 'flags' : 0x9000 , 'size' : 2 , 'endian' :
'big' }
]

chip_selects=list(map(lambda port: digitalio.DigitalInOut(port),
chip_select_ports)) # assessing the cs ports

with busio.SPI(board.GP10, MOSI=board.GP11) as spi_bus:
    devices = list(map(lambda cs: SPIDevice(spi_bus, cs), chip_selects)) #
assessing the cs ports
    print('Start- please remember that the first input line can contain
garbage...')
    for line in sys.stdin:
        line=line.strip()
```

```
        values=line.split()
        value_counter=-1
        for device in devices:
            for channel in output_channels:
                value_counter += 1
                if value_counter>= len(values):
                    break
                this_val=values[ value_counter ].strip()
                try:
                    val_out = int( this_val )
                    val_out  &= channel['max'] # mask out everything > max
                except:
                    val_out=0

                bytes_out = (channel['flags'] | (val_out * channel['mult'] )
).to_bytes(channel['size'],channel['endian'])
                # The object assigned to spi in the with statements below
                # is the original spi_bus object. We are using the busio.SPI
                # operations busio.SPI.readinto() and busio.SPI.write().
                #bytes_read = bytearray(4)
                #with device as spi:
                #    spi.readinto(bytes_read)
                with device as spi:
                    print('send', ''.join('{:02x}'.format(x) for x in
bytes_out), val_out)
                    spi.write(bytes_out)
```

The Hardware

Fix Me!: Where is the schematics?