KMK FW - Split Keyboards

Split keyboards are mostly the same as unsplit. Wired UART is fully supported, and testing of Bluetooth splits, though we don't currently offer support for this.

Notice that this Split module must be added after the HoldTap module to the keyboard.modules.

Drive names

As you will have two CircuitPython drives to update regularly, it is advised to rename them to make your life easier. Follow the instructions on how to rename CIRCUITPY drives while making sure that: - The left side ends in "L", - the right side ends in "R", - the entire drive name is 11 characters or less! This is a limitation of the filesystem and you will receive an error if you choose a name longer than that.

For example: `NYQUISTL` for the left and `NYQUISTR` for the right half.

Wired UART

Wired connections can use UART over 1 or 2 wires. With 2 wires, you will be able to synchronize the halves allowing additional features in some extensions.

```
from kb import data_pin
from kmk.modules.split import Split, SplitSide

split = Split(split_side=SplitSide.LEFT)
keyboard.modules.append(split)
```

Bluetooth split (aka no TRRS) [Currently in testing]

Wireless splits are fully featured with 2 way communication allowing all extensions to work 100%.

```
from kb import data_pin
from kmk.modules.split import Split, SplitType, SplitSide


split = Split(split_type=SplitType.BLE, split_side=SplitSide.LEFT)
OR
```

```
split = Split(split_type=SplitType.BLE, split_side=SplitSide.RIGHT)
keyboard.modules.append(split)
```

Config

Useful config options:

```
split = Split(
    split_flip=True,  # If both halves are the same, but flipped, set this
True
    split_side=None,  # Sets if this is to SplitSide.LEFT or SplitSide.RIGHT,
or use EE hands
    split_type=SplitType.UART,  # Defaults to UART
    split_target_left=True,  # Assumes that left will be the one on USB. Set
to False if it will be the right
    uart_interval=20,  # Sets the uarts delay. Lower numbers draw more power
    data_pin=None,  # The primary data pin to talk to the secondary device
with
    data_pin2=None,  # Second uart pin to allow 2 way communication
    uart_flip=True,  # Reverses the RX and TX pins if both are provided
    use_pio=False,  # Use RP2040 PIO implementation of UART. Required if you
want to use other pins than RX/TX
)
```

`split_side`

This tells your microcontroller which side it handles. It's usually not necessary -- defaulting to `split_side = None` it results in: - Auto detection of the side from the drive name (ending with 'R'/'L'). - The `split_target` will be overridden. Each side will act as a `split_target` if connected to a USB host.

The default will cover most cases, but you can still choose to set all that manually, if for example: - You want to debug and/or upload to both sides at the same time over USB. Explicitly setting `split_side` and `split_target` prevents that both halves consider themselves as `split_target` when a USB connection is detected. - There are different peripherals on both sides, others than just mirrored the columns (see `split_flip` section). That means that the most boards with "flippable" PCBs do **not** need this. The following code is **not** a guideline, but an extraordinary example showing the flexibility of KMK (and would realistically be applicable only in messy handwired keyboards):

```
from storage import getmount

side = SplitSide.RIGHT if str(getmount('/').label)[-1] == 'R' else
```

```
SplitSide.LEFT

if side == SplitSide.RIGHT:
    keyboard.col_pins = ...
    keyboard.row_pins = ...
    keyboard.diode_orientation = ...
else:
    keyboard.col_pins = ...
    keyboard.row_pins = ...
    keyboard.diode_orientation = ...

split = Split(
    split_side=side,
    target_left=True,
    ...
)
```

Note: It is best to stay as consistent as possible, but thanks to the `coord_mapping` feature, none of the `col_pins`, `row_pins` and `diode_orientation` **need** to be the same for both side. It is however necessary for `len(col_pins) * len(row_pins)` to be the same to calculate the offset of the key number on the left side correctly.

split_flip

If your split keyboard uses the **same PCB for both sides**, but vertically flipped, set this to `True`, and `False` otherwise. `True` means the wiring is the same for both side except that the `col_pins` are reversed.

split_target_left

The "split_target" refers to the side that acts as the USB HID.

Setting `split_side = None` (similar to EE HANDS in QMK) this parameter will be overridden.

uart_flip

If your boards are connected through the **same** pins (like GPIO**4** of board A to GPIO**4** of board B): use `uart_flip = True`.

If your boards are connected through **different** pins (like GPIO**4** of board A to GPIO**10** of board B): use `uart_flip = False`.

use_pio

If you're using an RP2040 based board and want the split communication to use other pins than the ones with hardware UART support, you can use the PIO implementation. Typical use cases for this are premade boards, made with QMK's bitbanging protocols in mind.

In order to enable it, you must:

- Install CircuitPython version > 7.2,
- pass `use_pio=True` into the `Split()` constructor.

`data_pin/data_pin2`

For UART `SplitType`: on the `split_target` side, `data_pin` is the one use for RX, `data_pin2` the one for TX.

EE HANDS / AUTO HANDEDNESS

If you want to plug USB in on either side, or are using Bluetooth, this is for you. For this feature to work your CircuitPython drive must be renamed following the guidelines at the beginning of this doc.

For wired connections you need to pass the UART pins. For Bluetooth, remove the `split_side` like this

```
# Wired, adjust the pins to fit your hardware
split = Split(data_pin=board.D0,gddata_pin2=board.D1,)
# Wireless
split = Split(split_type=SplitType.BLE)
```