

Expert GUI Development

新たなリモート機能を追加する

- すでに現行バージョンに色々なリモート機能が定義されているが、gRPCに新たな機能を追加する。
- 現行定義済みの機能に支障をきたすことなく、gRPCファイルを生成する。
- GIT-Hubの使い方に慣れ、いつでもプログラム修正前のプログラムに戻れるように。ソフトウェア開発の基礎。

1. verify_latch(d)

```
intt.py
...
Input is dam object.
Output is -1 if error occurred, else the number of fibers latched, including zero.
...
def verify_latch(d):
    d.reg.latch_sync_rst = 1
    d.reg.latch_sync_rst = 0
    if d.reg.latch_sync == 0:
        latch_fpga(d)
        time.sleep(1)

        latch_arr = bin(d.reg.latch_sync)[2:].zfill(28)
        blatch_ok = [True if i=='1' else False for i in latch_arr][::-1]

        count_latch = latch_arr.count('1')

        R = '\033[31m'
        W = '\033[0m'
        G = '\033[32m'

        rtn_string = ""
        for c,i in enumerate(blatch_ok):
            color = G if i else R
            print("{}Ladder Channel #{0:3}: {}".format(c//2,i,color,W))
            rtn_string += "{}Ladder Channel #{0:3}: {}".format(c//2,i,color,W) + "\n"
        return count_latch, rtn_string
    else:
        return -1

.....
[UNDER DEVELOPMENT]
Brings FLX board to a state that can interact with the ROC.

BCO and ST CLK generated, internal logic reset, transceivers properly initialized.

Inputs: dam object
Outputs: None
.....
def cold_start(d):
    dma_dataset(d, test_mode=False)

    # Placeholder
    gtm_locked = True

    # Makes sure commands are sent to both ROCs
    d.reg.sc_target = 0x3
    # Initial clock distribution <- Felix internal clock si5345 clock chip slow control and high speed data
    # transceivers.
    if d.reg.si5345_pll.nLOL == 0:
        d.reg.si5345_pll.program = 1
        time.sleep(5)
    if d.reg.lmk_locked == 1: # <- 6x9.4MHz clock generates BCO and start is in stable mode
        init_clocks(d) # <- initialize BCO and START clocks
    else:
        ...
-- intt.py 52% L224 (Python EIDoc)
```

- Data fiberをlatchさせるファンクション、latch_fpga()はすでに現行バージョンで定義済みのようだ。
- 次にステップとして、Data FiberのLatchがうまく行ったかどうかを確認できるファンクション verify_latch(d);をgRPCで定義する。
- このファンクションはFEMのLEDに相当し、それぞれのファイバーチャンネルに対し、うまく行けばTRUEを返し、失敗ならFALSEを返す。
- gRPCでは、以下を定義する
 - 行き：verify_latch(d)の実行
 - 帰り：ファンクションが返す結果

2. New Function to be implemented

```
intt.py
0XCF559370,
0XCF561460,
0XCF569040,
0XCF571400,
0XCF579420,
0XCF581050,
0XCF5893F0,
]

for item in cheat:
    send_fphx_cmd(d, item)
    time.sleep(.001)

#INTT Default parameters for FPHX Chips
def ld_fphxparam(d, cheat=None):
    cheat = [
        0XCF50A800,
        0XCF511050,
        0XCF519010,
        0XCF5210F0, # DAC0 15
        0XCF529130, # DAC1 19
        0XCF531170, # DAC2 23
        0XCF5391B0, # DAC3 27
        0XCF5411F0, # DAC4 31
        0XCF549230, # DAC5 35
        0XCF551270, # DAC6 39
        0XCF5592B0, # DAC7 43
        0XCF561460,
        0XCF569040,
        0XCF571400,
        0XCF579420,
        0XCF581050,
        0XCF5893F0,
    ]

    for item in cheat:
        send_fphx_cmd(d, item)
        time.sleep(.001)

def write_sparam(d, command, wedge, dataword, sizeword):
    d.reg.roc_command = command
    d.reg.roc_wedge = wedge
    d.reg.roc_dataword = dataword
    d.reg.roc_sizeword = sizeword

def read_fphx(d, chip, register, wedge, roc):
    send_fphx_cmd(d, make_fphx_cmd(chip, register, 0, 0), wedge)
    time.sleep(0.1)
    output = [None, None]
    if roc == 1:
        readback = d.reg.fphx_readback1
    elif roc == 0:
        readback = d.reg.fphx_readback0
    else:
        raise Exception("Which ROC should I read? Accepted values are [0,1]")
    return output

--: intt.py 33% L142 (Python ELDoc)
```

- intt.pyに定義されている ld_fphxparam(d, cheat=None)をExpert GUIからgRPCを使って呼ぶ。
- このfunctionは現行バージョンには取り込まれていない。

intt.py

ld_fphxparam()のパラメータ