Pcal guardian kickoff

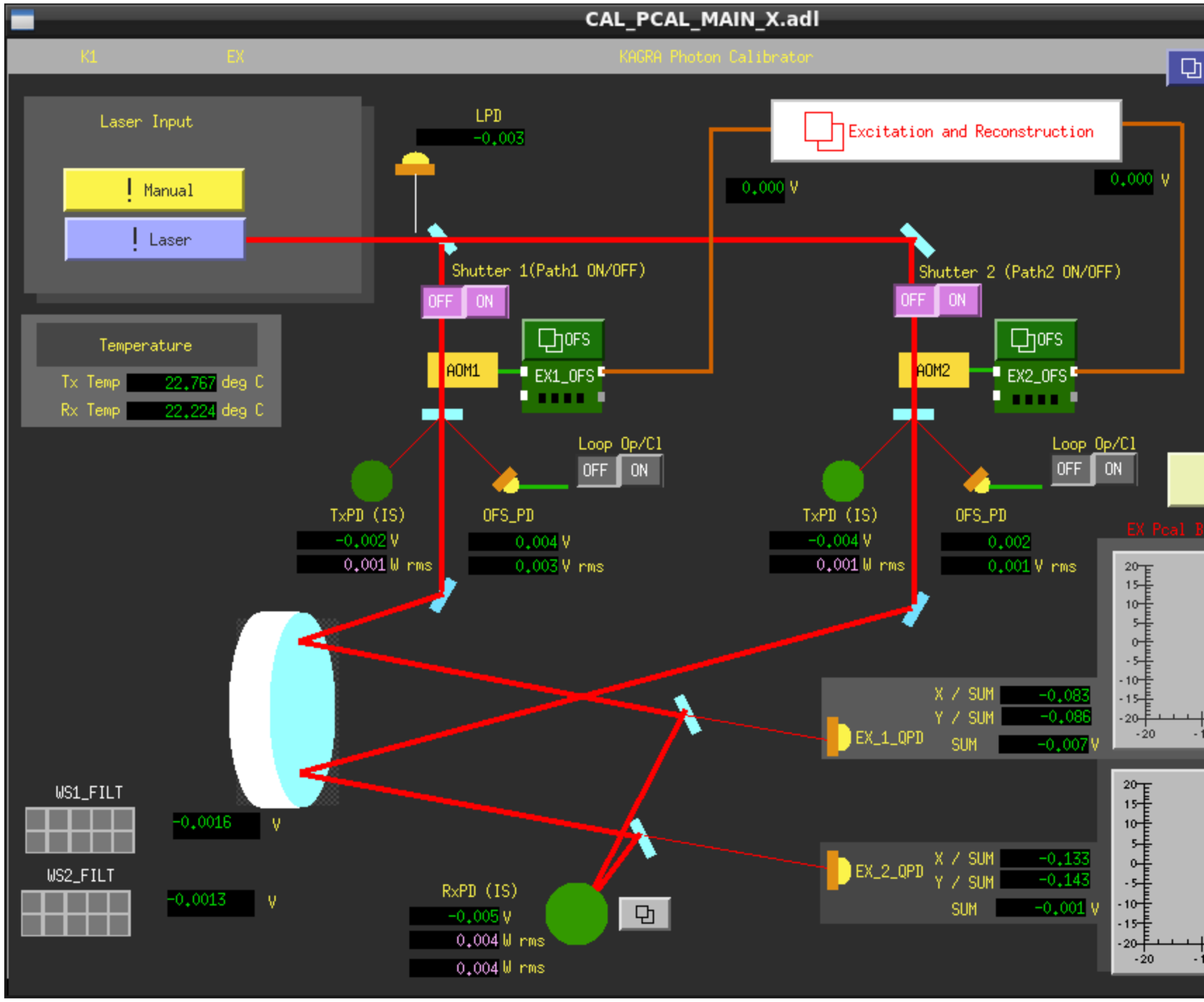
2021/07/29　YamaT

Guardian

- Pythonで書かれたState machine
- 基本的に全ての遅いハードウェア制御はGuardianで行われるべき
    - pcalで言うとOFS制御のON/OFF、較正ラインのON/OFFなど
- 今のPcal GuardianはO3直前に山本Tが書いた燃えないゴミ
    - 制御がサチってないかのチェックがザル等
- Guardian状態は最終的にKAGRAの観測モードフラグに関わってくる
- LIGO-T1500292 (https://dcc.ligo.org/LIGO-T1500292/public)

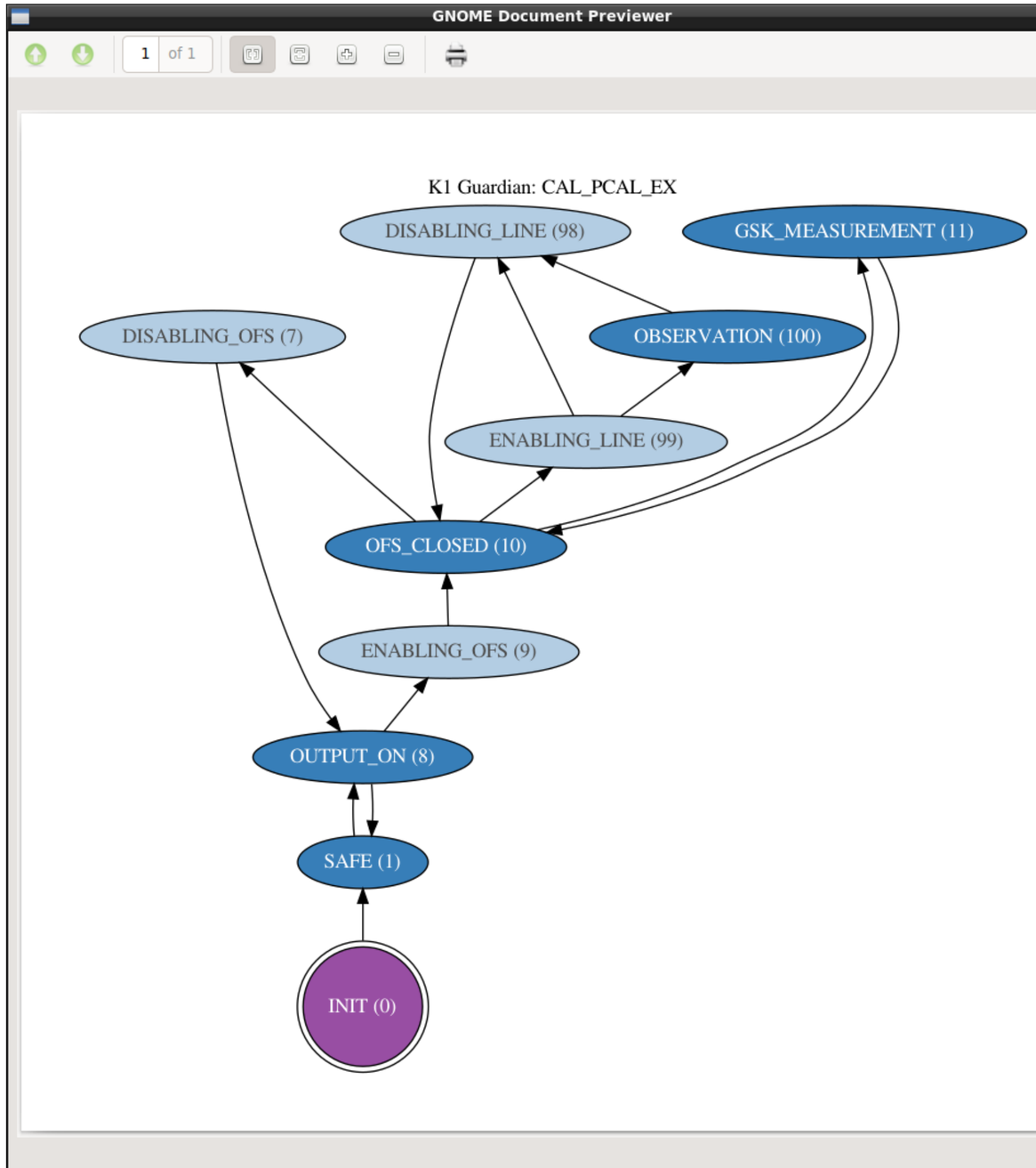
Pcal

- 干渉計状態を知るために既知の参照信号を注入する装置
- レーザーの輻射圧で鏡を動かす
- レーザーパワーの安定化制御が行われる
- 鏡に当たるレーザーパワーを正確に推定する事が肝



Pcal guardian

- **SAFE:** レーザーハザード的に安全な状態
- **OUTPUT_ON:** ハザード状態
- **OFS_CLOSED:** 制御がかかった状態
- **OBSERVATION:** 観測モード
- **GSK_MESASURMENT:** テキトー



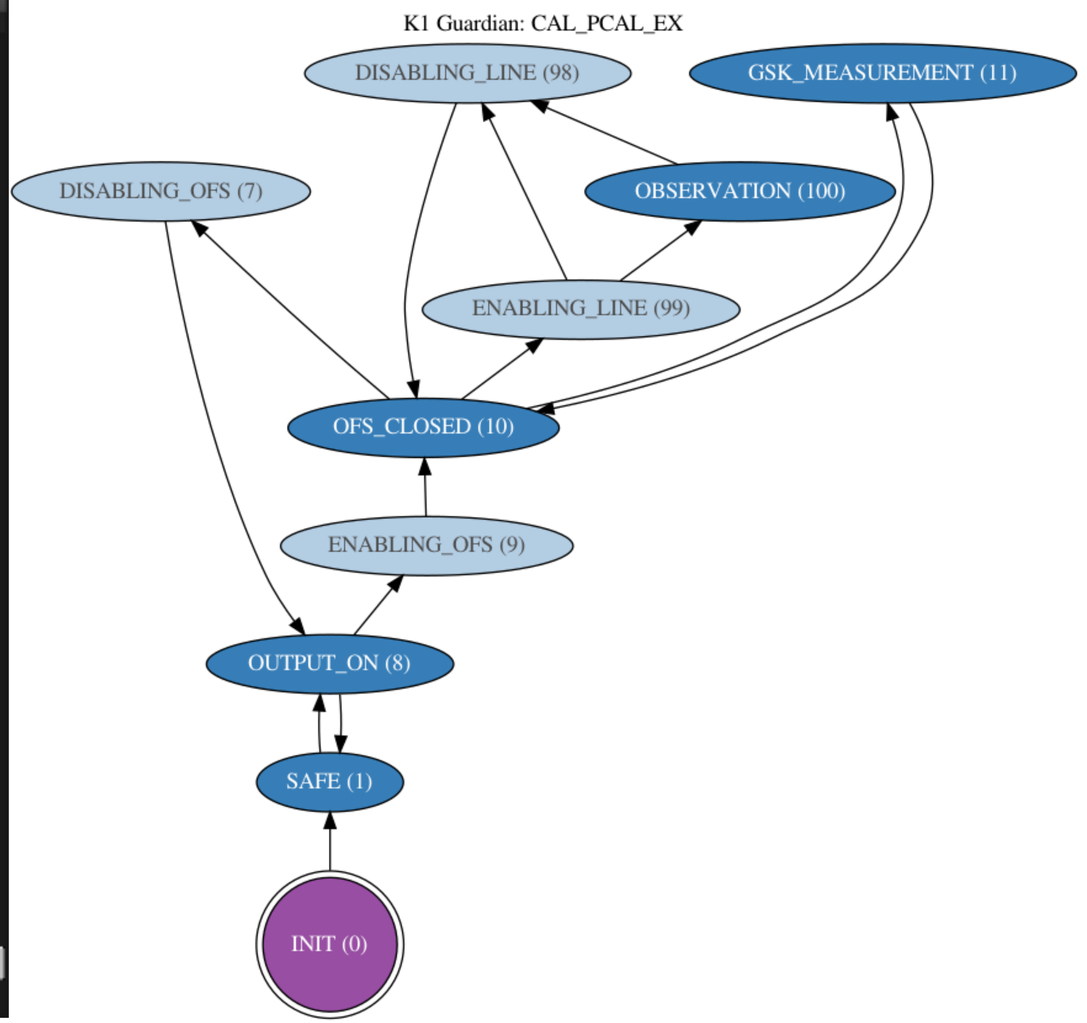
Pcal guardian

emacs@k1ctr0.kagra.icrr.u-tokyo.ac.jp

```
156 class ENABLING_OFS(GuardState):
157     index = 9
158     request = False
159
160     def main(self):
161         ofsctrl_sw('ON')
162
163     def run(self):
164         return True
165
166
167 class OFS_CLOSED(GuardState):
168     index = 10
169
170     @ofs_lock_check
171     def main(self):
172         sdf.restore(FEC, 'READY')
173
174     @ofs_lock_check
175     def run(self):
176         if ezca['GRD-CAL_PCAL_'+ARM+'_REQUEST'] == 'OFS_CLOSED' or is_ifo_locked():
177             return True
178         else:
179             notify('waiting IFO')
180
181
182 class GSK_MEASUREMENT(GuardState):
-:%%-  PCAL.py        61% (159,0)   Git-kamioka  (Python AC View ElDoc) [*class definition*]
```

GNOME Document Previewer

1 of 1



とりあえずやり始めたい事

- とりあえずSAFE状態〜干渉計に明け渡せるREADYな状態まで作る
- OBSERVATIONモードはFPMIとかPRFPMIになってから

- ステートの過不足は？
- 各ステートでは何(どの信号)をチェックすべき？
- 諸々の測定モードの取り扱いは？