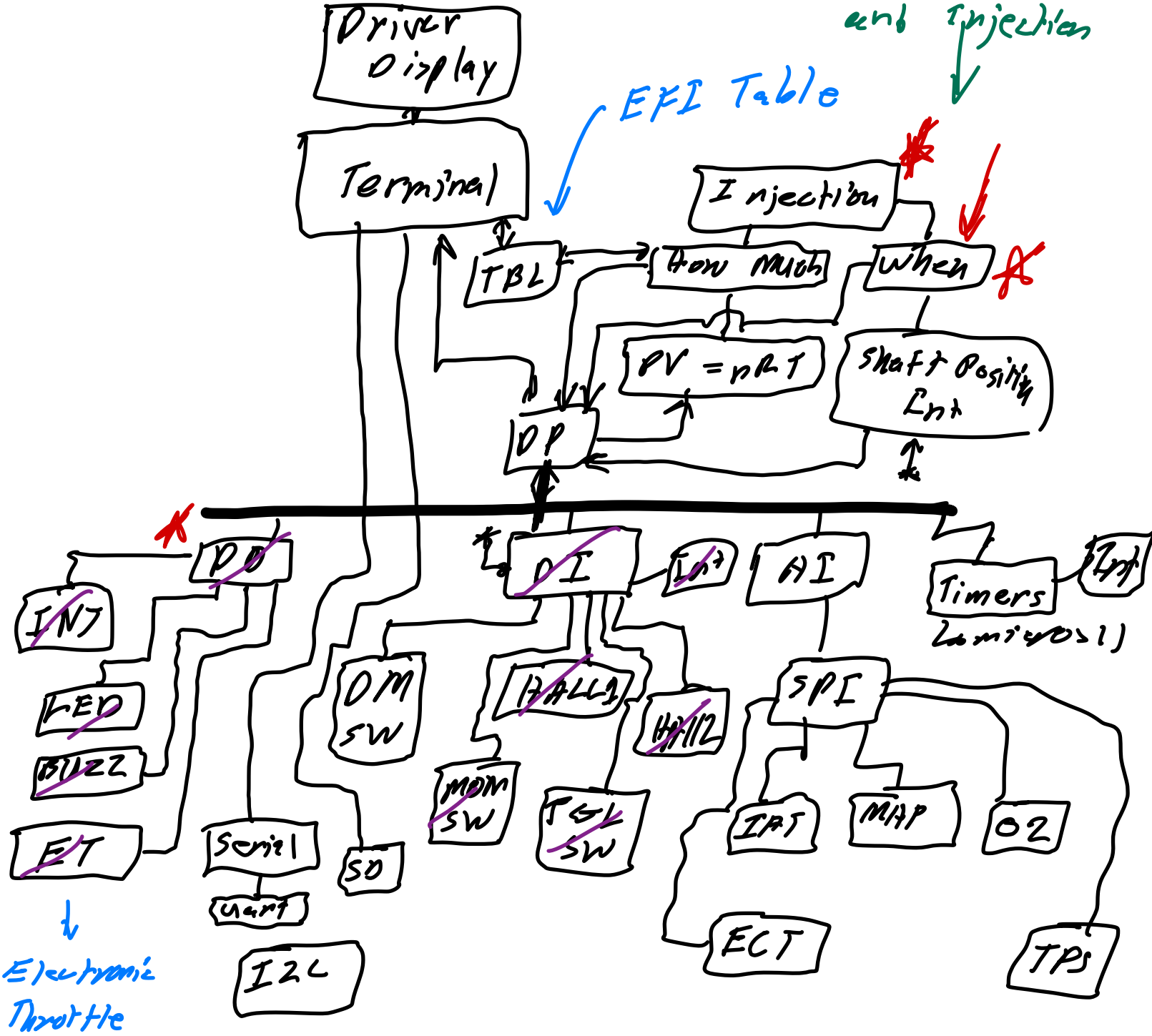
Main

- EFI Controller

Ideal Gas Law Calculation and Injection

EFI Table



Electronic Throttle

Loose End

- interrupts → PI should maintain pointers to interrupt functions, pass-b to it from PP.

- timers → Make an object to give PP timing functionality
  → PO need timing for injects

To include

- Serial Monitor
  - Statuses
  - Sensor values and settings
  - Data logging (serial or SD card)
- Misc
  - Driver Display
  - Battery Level ← Buzzer
  - Electronic throttle

- TM

  - maintain 2 tables
    - have $PV=nRT$ use I table, update other table, then update $PV=nRT$'s pointer.

→ Useful for mass updates of EFI table

Opaque Structs

main.c

```
#include "ob.h"

struct Obj* inst;
```

ob.h

```
struct Obj;

void initObj(struct Obj**);
```

ob.c

```
struct {
  int hi;
  float pi;
} Obj;

void initObj(... *ptr) {
    ptr = malloc(sizeof
                 (Obj));
    .
    .
    .
}
```

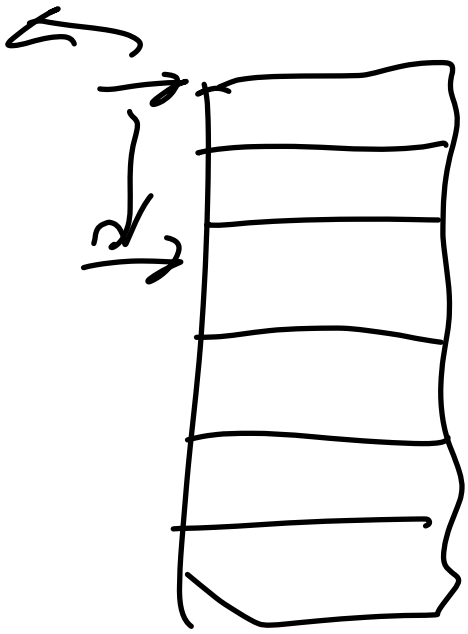
memory pool

→ blocks of constant size that can be allocated

→ in our application, we don't need to free memory.

mempool

```
uint32_t blocks[10,000];
```



Flow

```
setup()
   :
   :
loop() {
   getData();
   processData();
}
```

```
loop() {
  if (flag) {
    getData();
    processData();
    flag = 0;
  }
}
```

```
timer0
  ↓
int() {
   flag = 1;
}
```

→ 2 Time periods (Synchronous)

→ Calculation/Processing, F_c

→ HID, F_h

→ Asynchronous

→ inj

→ serial monitor inputs

Calculation / Processing Cycle

CALC_FLAG

↓

sensor values

↓

processing

↓

update variables atomically

↓

archive current PP

↓

exit

PP_update()

PV=nRT gets values from PP. Calculates new values.

→ PV=nRT tells PP new values and other information.

→ New inj length

→ Table values used

→ Time of last inj

→ PV=nRT updates injection pulse length atomically.

Terminal

→ Asynchronous input buffer updated

read operations

→ reacting to inputs in the buffer

e.g.: updating TBL, taking note of requests, toggles, etc.

Printing to serial monitor should be done in a non-blocking way

→ serial write out

write operations

→ buffer generation to a buffer.

→ Program can log events. These are not sent until terminal flag is set high

→ look at toggles, requests from user, etc. and put it into buffer

→ kick off outputting of buffer.

Terminal Capabilities

Requests

→ Changing TBL values

→ get current OP ✓ extra

→ current version value (readable format)

→ Toggles

→ send back OP

→ send back debug info

→ allow engine on