# To Build

g++ -pthread [main.cpp] -o [main] -lpigpiod\_if2 -lrt -std=c++20

\*\*make sure pigpio daemon is running before running program: sudo pigpiod

# Register Addresses

* **static constexpr char MAP = 1**;

IN4 mapping register; MAP[relay-1] mapped to IN4 relay=1-8; default = 1<<(4-1)

* **static constexpr char BOL = 2**;

Boolean mode. BOL[relay-1] 0=OR 1=AND SPI w/ parallel inputs; default=0

* **static constexpr char OVL = 3**;

current overload, 0=current limit only, 1=latching shutdown; default=0

* **static constexpr char OVT = 4**;

overtemp, 0=restart after cooldown, 1=latching shutdown; default=0

* **static constexpr char SLE = 5**;

slew rate, 0=10us 1=50us; default=0

* **static constexpr char STA = 6**;

DMOS state, read only. Use CTL, not STA to read on/off state. Not sure what this is but it's not the same; default=0

* **static constexpr char CTL = 7**;

Control of outputs 1-8 by setting CTL[OUTx] default = 0.

# Relay Diagnosis Statuses

* **static constexpr int RELAY\_CH\_OK = 0b11;** //normal function
* **static constexpr int RELAY\_CH\_OVERLOAD = 0b10;** //overcurrent or SCB
* **static constexpr int RELAY\_CH\_OPEN = 0b01**; //open load/open circuit
* **static constexpr int RELAY\_CH\_SHORT2GND = 0b00;** //short to ground

# Constructor

* **TLE7230(bool daisyChain = false, int baud = 4194304, int PI = 0x42)**

**: daisyChain(daisyChain), PI(PI)**

* **daisyChain** is whether the two TLE7230Rs’ SPI is in a daisy-chained circuit.
* **baud** is the SPI clock frequency in Hz
* **PI** is an (optional) external PiGPIO handle. Default value of 0x42 tells constructor to handle PiGPIO internally. Normally, pigpio\_start returns a handle = 0

# Destructor

* **~TLE7230()**
  + Closes SPI channels and then closes PIGPIO handle (if not external). Garbage collection calls this automatically.

# GPIO

* **int getGpioHandle() // accessor method, copies private PI member variable**
* **int getFLTN1() // returns 1 (set) or 0 (cleared)**
* **int getFLTN2() // returns 1 (set) or 0 (cleared)**
* **int writeRSTn(bool level) //level = 1 or 0. returns gpio\_write() return**

# Diagnostics

* **constexpr bitset<16> getDiagStatus(int device) // accessor (single device)**
* **pair<bitset<16>, bitset<16>> getDiagStatus() //accessor (both devices)**
* **int relayStatus(int device, int** **relay) // accessor (single relay of single device)**
* **void printDiagStatus() // prints to console in easy to read format**
* **int updateDiagStatus() // sends DIAGNOSIS\_ONLY command over SPI**

# Register Reads

* **pair<char, char> readRegisters(char addr1, char addr2 = addr1)**
  + **daisy-chain only, reads 2 registers at once**
  + **C++ does not allow default initialization from previous argument as order of eval is not guaranteed. This is \*actually\* implemented w/overloading, vs. what is syntactic sugar for this document.**
* **char readRegister(int device, char addr)**
  + **returns contents of a single register on a single device**

# Register Writes

* **int writeRegisters(char addr1, char addr2, char data1, char data2)**
* **int writeRegisters(char addr, char data1, char data2)**
* **int writeRegisters(char addr, char data)**
* **int writeRegister(int device, char addr, char data)**

# Register Resets

* **int resetRegisters(int device)**
* **int resetRegisters()**

# Relay Control

* **int turnRelayOn(int device, int relay)**
  + **device:** 1 = CSn1-selected (!daisyChain) or MOSI-receiving device (daisyChain), 2=CSN2/MISO-transmitting device
  + **relay:** 1-8
* **template <typename T> int turnRelaysOn(int device, T relays)**
  + **device:** 1 = CSn1-selected (!daisyChain) or MOSI-receiving device (daisyChain), 2=CSN2/MISO-transmitting device
  + **relays:** iterable collection<int> 1 through 8 (e.g., vector<int>, array<int>, list<int>, set<int>, unordered\_set<int>,

To use a custom [myClass]<int> put this into its definition:  
 template <typename T>  
 using is\_iterable = decltype(detail::is\_iterable\_impl<T>(0));

And implement begin()/end() etc.

OR, **T relays** can be a bitset<8>, string(“01001101”), unsigned int (e.g., 0b01010011), vector<bool>

* **int turnRelayOff(int device, int relay)**
  + **device:** 1 = CSn1-selected (!daisyChain) or MOSI-receiving device (daisyChain), 2=CSN2/MISO-transmitting device
  + **relay:** 1-8
* **template <typename T> int turnRelaysOff(int device, T relays)**
  + **device:** 1 = CSn1-selected (!daisyChain) or MOSI-receiving device (daisyChain), 2=CSN2/MISO-transmitting device
  + **relays:** iterable collection<int> 1 through 8 (e.g., vector<int>, array<int>, list<int>, set<int>, unordered\_set<int>,
  + To use a custom [myClass]<int> put this into its definition:  
     template <typename T>  
     using is\_iterable = decltype(detail::is\_iterable\_impl<T>(0));
  + And implement begin()/end() etc.
  + OR, **T relays** can be a bitset<8>, string(“01001101”), unsigned int (e.g., 0b01010011), vector<bool>

# Test

* **int test()**
  + Turns on each channel for 1 second and off for 1s, one at a time, device 1 and 2. Also tests GPIO pins.