Example RTM Bring-up Firmware (MicroPython)

This project contains an example firmware for the RTM Bring-up Adapter. The firmware runs on a Raspberry Pi Pico 2 or Raspberry Pi Pico 2W installed on this board. It is written in Python and implements a finite state machine (FSM) to control a connected MicroTCA RTM (Rear Transition Module). The firmware manages the I²C I/O expander, board presence detection, power enable, and LED status.

License

Author: Sakshi Pahalwan

Features

- Detects RTM presence via the PS# signal
- Monitors:
  - RTM handle (open/closed)
  - RTM Power Good (PG)
- Controls:
  - Blue, red, and green LEDs
  - RTM 12V power
  - RTM enable signal
- Supports safe hot-plug (insertion/removal)
- Prevents I²C crashes using retry logic

Hardware Setup

A MicroTCA-compliant RTM is expected to be connected to the RTM Bring-up Adapter.
For details, see: https://mtca.desy.de/Tools_and_Templates/RTM%20Template/

GPIO

Raspberry Pi Pico 2/2W Pin	Function
GP13	PS# (Board presence, LOW = present)
GP1	RTM_12V (power control)

RTM I²C Expander

Pin	Function
P0	RTM_HANDLE (input)
P1	Blue LED (output, inverted)
P2	Red LED (output, inverted)
P3	Green LED (output, non-inverted)
P4	RTM_PG (input)
P5	RTM_RESET# (output)
P6	RTM_ENABLE (output)
P7	RTM_EXP_WP (output)

Installation & Execution

The Raspberry Pi Pico 2/2W needs to be set up once. After installing the Python code in non-volatile memory using a Windows PC, the adapter operates in standalone mode and no longer requires a PC connection.

Code download and installation are expected to be performed using Thonny: https://thonny.org/

1. Connect the Raspberry Pi Pico 2/2W to your PC. Power it up normally (not in BOOTSEL mode).
2. Open the Thonny IDE.
3. Click the Python button in the bottom-right corner and select “Configure interpreter”:
   - Interpreter: *MicroPython (Raspberry Pi Pico)*
   - Port: *Board CDC @ COMx* (COMx corresponds to your device)
4. Click OK.
5. It is strongly recommended to test the code before permanent installation. Run `main.py` directly from Thonny. If there are issues (logic or syntax), error messages in the console will help with debugging.
6. If the code runs without issues (the RTM powers up and down using the handle), install it permanently:
   - Open `main.py`
   - Select “Save as” from the File menu
   - Choose *Raspberry Pi Pico* as the destination
7. Disconnect the USB cable, power-cycle the board, and verify standalone operation by operating the RTM handle.

FSM Behaviour

1. Board Detection

- PS# HIGH
  - Board not present
  - All LEDs OFF

    - Power OFF
- PS# LOW
    - Board detected
    - FSM starts

2. WAIT_HANDLE State

- Blue LED indicates waiting
- If the handle is:
    - OPEN (1) → remain in WAIT state
    - CLOSED (0):
    - Blue LED OFF
    - RTM power ON
    - Transition to BOARD_UP state

3. BOARD_UP State

- Monitors:
    - Handle state
    - Power Good signal (RTM_PG)
- If the handle opens:
    - Return to WAIT state
    - Power OFF
- If PG:
    - PG = 1 → Green LED ON (power good), red LED OFF
    - PG = 0 → Red LED ON (power fault), green LED OFF

Hot-Plug Support

- Safe handling of board insertion/removal
- I²C read/write operations protected with retry logic
- On removal:
    - LEDs OFF
    - Power OFF
    - FSM resets to WAIT_HANDLE state

Error Handling

- I²C failures handled with up to 3 retries
- Prevents crashes when:
    - The board is removed

Debugging Tips

- Use Thonny:
    - Run → Stop/Restart if the system becomes unresponsive
- Add a startup delay if needed:
    - time.sleep(2)