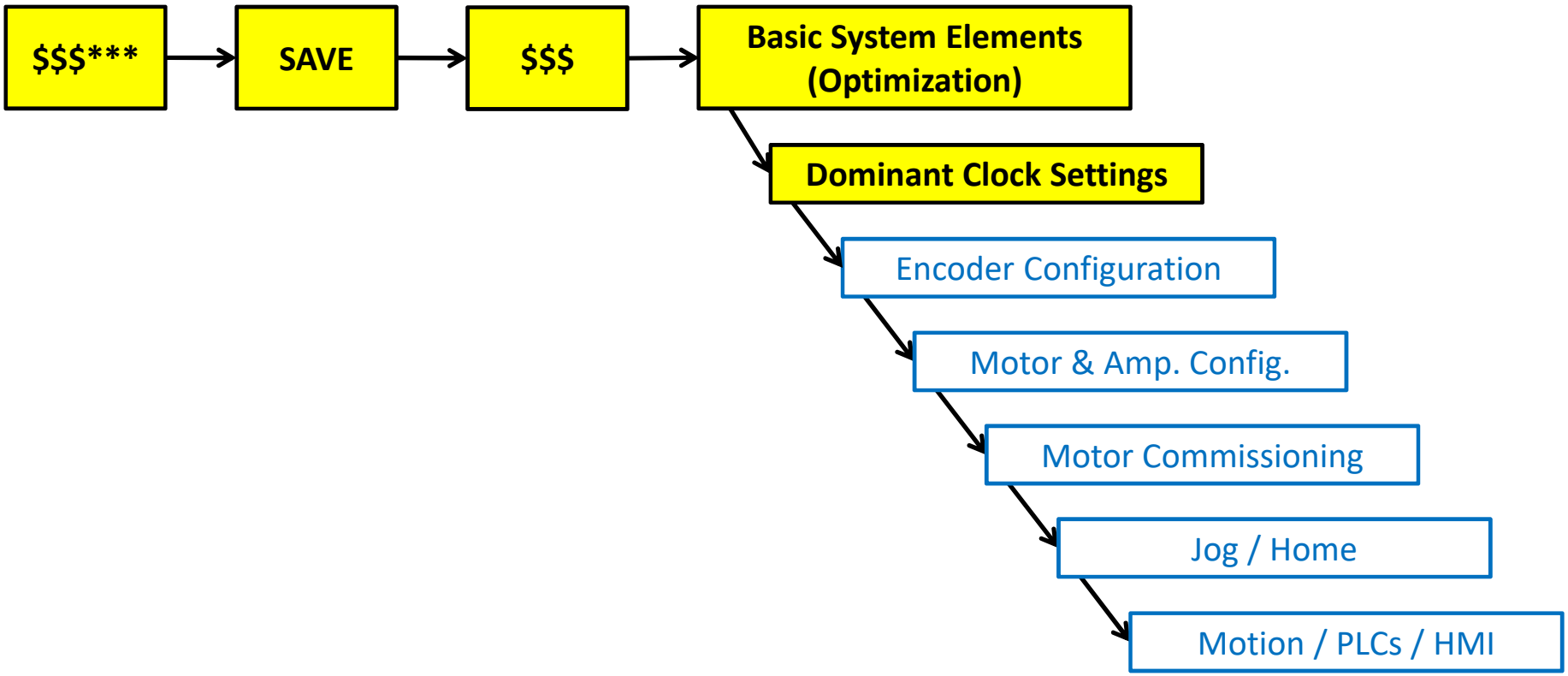
OMRON | DELTA TAU

# Power Brick LV
# System Configuration

# System Configuration



# Optimization

## ➢ Factory Default Reset Procedure

- $$$*** performs a factory default reset
- SAVE retains current configuration in flash memory
- $$$ standard reset (~equivalent to cycling power)

*Note*

Starting a new setup from factory default settings is high recommended. This ensures a clean starting point.

```
Sys.WpKey = $AAAAAAAA          // DISABLE WRITE PROTECTION

Sys.MaxCoords = 2              // CS USED (+1, CS0)
Sys.MaxMotors = 5              // MOTORS USED (+1, MOTOR #0)
Sys.MaxRtPlc = 0               // 0 ONLY RT PLC
Sys.NoShortCmds = 0            // ALLOW
Sys.MotorsPerRtInt = 0         // ALL MOTORS EVERY RTI
Sys.BgSleepTime = 2000         // FASTER COMM.
Sys.pAbortAll = 0              // POWER BRICK SPECIFIC
```

*Note*

Optimizing the number of active motors and coordinate systems reduces the CPU load

# Dominant Clock Settings

```
PowerBrick[0].PhaseFreq = 10000          // PHASE FREQUENCY [Hz]
PowerBrick[0].ServoClockDiv = 1          // SERVO = PhaseFreq / (ServoClockDiv + 1)

PowerBrick[1].PhaseFreq = 10000          // PHASE FREQUENCY [Hz]
PowerBrick[1].ServoClockDiv = 1          // SERVO = PhaseFreq / (ServoClockDiv + 1)

Sys.RtIntPeriod = 0                      // RTI EVERY (Sys.RtIntPeriod + 1) SERVO

Sys.ServoPeriod = 1000 * (PowerBrick[0].ServoClockDiv + 1) / PowerBrick[0].PhaseFreq
Sys.PhaseOverServoPeriod = 1 / (PowerBrick[0].ServoClockDiv + 1)

PowerBrick[0].Chan[0].PwmFreqMult = 3    // PWM = PhaseFreq * (PwmFreqMult + 1) / 2
PowerBrick[0].Chan[1].PwmFreqMult = 3
PowerBrick[0].Chan[2].PwmFreqMult = 3
PowerBrick[0].Chan[3].PwmFreqMult = 3
```

*Note* If an 8-axis Brick LV, then PowerBrick[1] clocks must be set up as well. Usually equal to the same frequencies as PowerBrick[0].

*Note* It is always recommended to do a **SAVE**, followed by a **$$$** when the clock settings are changed

# Dominant Clock Settings

- **Phase Frequency, PowerBrick[].PhaseFreq**
  - The phase cycle, among other tasks, primarily executes motor commutation (including user-written phase), current loop closure, and current sensor readings
    - 5 – 10 KHz lower performance systems
    - 20 – 40 KHz higher performance systems
  - If no phase tasks are needed, usually the phase is set the same as the servo frequency
- **Servo Frequency, PowerBrick[].ServoClockDiv**
  - Divided from the phase frequency by (ServoClockDiv + 1)
  - The servo cycle, among other tasks, primarily executes motor servo control (including user), fine trajectory interpolation, encoder pre-processing, and compensation tables
    - 2 – 4 KHz lower performance systems
    - 10 – 20 KHz higher performance (demanding) systems
      - High resolution encoders
      - High speed / precision / accuracy applications

# Dominant Clock Settings

## Real-Time Interrupt, Sys.RtIntPeriod

- The real-time interrupt frequency is executed every (Sys.RtIntPeriod + 1) servo cycles

- The real-time cycle, among other tasks, primarily executes foreground script and C PLCs, plans coarse trajectory, computes kinematics transformations and motor safety checks
  - Typical rate is same as servo (Sys.RtIntPeriod = 0)
  - In high CPU load applications (servo > 10KHz), it may be reduced to execute every other or more servo cycles
    - Exception: if a programmed task in a foreground PLC is required to execute at servo rate

## PWM Frequency, PowerBrick[].Chan[].PwmFreqMult

- With the ACC-24E3, the PWM frequency primarily governs the command output to direct digital PWM amplifiers
  - In motor applications, it is directly related to the inductance and resistance of the motor
  - Typically set to equal or half of phase