ASTA EPSILON ENGINEERING

Single and Three Phase Motor Starting Equipment from AEG

Voltage Rating: 120-270V AC 1~
400-13800V AC 3~

Power Rating: 0.75-10 kW 1~
200-25000 kW 3~

INDUSTRY STANDARDS

To cater to a global market, the following standards are recommended:

- **IEC Standards**:
  - IEC 60947-4-2: For soft starters.
  - IEC 60034: For motor-related requirements.
  - IEC 60529: For ingress protection (IP) ratings.
- **UL/CSA Standards** (for North American markets):
  - UL 508C: Power conversion equipment.
  - CSA C22.2 No. 14: Industrial control equipment.
- **RoHS/REACH Compliance**: For environmental sustainability.
- **Indian Standard IS 60947 (Part 4/Section 2):** Low-voltage switchgear and control gear – AC semiconductor motor controllers and starters (equivalent to IEC 60947-4-2).
- **Safety and General Standards:**
  - IS 9000: Basic environmental testing procedures for electronic and electrical items.
  - IS 302: General safety requirements for electrical appliances.
  - IS 3043: Code of practice for earthing.

1. Accessing Standards

- **Purchase Standards Documentation**:
  Standards like IEC, UL, and others are proprietary, and you need to purchase them from the respective organizations. For example:
  - **IEC Standards**: Available at IEC's official website.
  - **UL Standards**: Available through UL Solutions.
  - **CSA Standards**: Available at CSA Group.

Many standards organizations offer individual document purchases or subscriptions for regular updates.

- **Membership/Library Access**:
  Joining national or regional standard organizations (like BIS in India or ANSI in the USA) often provides access to these standards at reduced costs. University or public libraries sometimes provide access too.

2. Certification vs. Compliance

- **Building a Product to Comply with Standards**:
  Compliance means designing and manufacturing your product according to the guidelines in the standards. This doesn't automatically give you certification.
- **Getting Certified**:
  Certification is a separate process conducted by accredited certification bodies (e.g., UL, TUV, Intertek). Here's how it works:
  1. **Submit Your Product for Testing**: Send your product to an approved testing laboratory.
  2. **Laboratory Testing**: The lab tests your product against the applicable standard (e.g., IEC 60947-4-2 for soft starters).
  3. **Certification Issuance**: If the product meets all requirements, a certificate of conformity (e.g., UL mark, CE mark) is issued.
  4. **Factory Audits**: Many certification bodies conduct regular audits of your manufacturing process to ensure ongoing compliance.

3. Why Certification is Important

- **Market Access**: Certification like UL, CE, or CSA is mandatory in many regions for selling your products.
- **Consumer Confidence**: Certified products are perceived as safer and more reliable.
- **Legal Requirements**: In some jurisdictions, selling uncertified electrical equipment can lead to penalties or bans.

4. Costs Involved

- **Documentation Costs**: Standards documents can range from $50 to $500 or more per standard.
- **Testing and Certification**: Costs vary based on the complexity of the product, typically ranging from $5,000 to $20,000 per product.
- **Ongoing Compliance**: Annual audits or re-certifications may incur additional costs.

5. Steps to Follow

1. **Study the Standards**: Purchase the required documents or consult with a standards expert.

2. **Design for Compliance**: Use the standards as design guidelines during development.
3. **Prototype Testing**: Pre-test your prototypes using internal testing setups to ensure compliance before submitting them to a certification body.
4. **Apply for Certification**: Choose an accredited body, submit your product, and follow their process.

Why Indian Standards Matter

- **Local Compliance**: Products sold in India must meet BIS standards to comply with Indian regulations.
- **Certification**: The BIS certification (ISI mark) is mandatory for some equipment categories under the **Compulsory Registration Scheme (CRS)**.
- **Market Access**: Ensures acceptance in the Indian market and among public sector undertakings (PSUs).

How to Access Indian Standards

1. **Purchase Standards**:
   Visit the **BIS e-Catalogue** or their official website (https://www.bis.gov.in) to buy the relevant standards.
2. **Testing and Certification**:
   BIS has a network of testing labs across India. You can submit your product to these labs for testing and certification.
3. **Self-Certification**:
   Some products may qualify for self-certification under BIS, but periodic audits by BIS officials are required.

Steps to Follow

1. Identify the Indian Standards applicable to your equipment.
2. Design and test the product to meet these standards.
3. Apply for BIS certification if required.
4. Display the ISI mark or certification details once certified.

Related Costs

S.No	Standard No.	year	Title	Price
1.	IS/IEC 60947 : Part 4 : Sec 2	2020	Low-Voltage switchgear and Controlgear: Part 4-2 Contactors and Motorstarters semiconductor motor controllers starters and Soft starters	1,400.00

S.No	Standard No.	year	Title	Price
1.	IS 9000 : Part 5 : Sec 1 AND 2	1981	Basic environmental testing procedures for electronic and electrical items: Part 5 Damp heat (cyclic) test	0.00

S.No	Standard No.	year	Title	Price
1.	IS 9000 : Part 7 : Sec 3	2019	Environmental Testing Part 7 Tests Section 3 Test Ec: Rough handling shocks, primarily for equipment-types specimens (First Revision)	300.00

S.No	Standard No.	year	Title	Price
1.	IS 9000 : Part 19 : Sec 1 to 5	1986	Basic environmental testing procedures for electronic and electrical items: Part 19 Test U: Robustness of terminations and integral mounting devices	0.00

S.No	Standard No.	year	Title	Price
1.	IS 302 : Part 1	2024	Household and Similar Electrical Appliances - Safety - Part 1 General Requirements	2,500.00

S.No	Standard No.	year	Title	Price
1.	IS 3043	2018	Code of Practice for Earthing (Second Revision)	0.00

BASIC WORKING PRINCIPLE OF A SINGLE PHASE MOTOR SOFT STARTER AND CONTROL UNIT

Let's dive into the **working and design of a Single-Phase Single Motor Soft Starter and Control Unit**. We'll break this process into key components and operational stages to ensure clarity and completeness.

A soft starter reduces the inrush current and starting torque of an AC motor during startup. It does this by gradually increasing the voltage supplied to the motor. Once the motor reaches its nominal speed, the soft starter bypasses the control elements to allow the motor to run at full voltage.

Components of the Soft Starter

1. **Power Circuit**:

- **Thyristors (SCRs)**: Used for phase-controlled voltage regulation during startup.
- **Bypass Contactor**: Engages after the motor reaches full speed to minimize heat losses in the SCRs.
- **EMI Filters**: Reduce electromagnetic interference caused by switching devices.

2. **Control Circuit**:
   - **Microcontroller/PLC**: Manages voltage ramp-up, monitors motor parameters, and handles fault conditions.
   - **Voltage/Current Sensors**: Monitor input and motor conditions.
   - **Control Buttons and Indicators**: Start/Stop buttons, status LEDs, fault indicators.
3. **Protection Components**:
   - **Overcurrent Protection**: Prevents damage due to sustained high current.
   - **Thermal Protection**: Shuts down the system if the motor overheats.
   - **Surge Protection**: Protects against voltage spikes.
4. **Cooling System**:
   - **Heatsinks**: For thyristors and other power electronics.
   - **Fans**: For forced air cooling in high-power designs.

Step-by-Step Working

1. **Startup Phase**:
   - When the motor is started, the SCRs begin in a partially conductive state, providing reduced voltage to the motor.
   - The microcontroller gradually increases the firing angle of the SCRs, thereby increasing the output voltage and ensuring a smooth acceleration of the motor.
2. **Run Phase**:
   - Once the motor reaches full speed, the bypass contactor closes, bypassing the SCRs and allowing full voltage to flow directly to the motor.
   - This reduces heat generation in the soft starter components.
3. **Monitoring Phase**:
   - The control unit continuously monitors voltage, current, and temperature.
   - If any parameter exceeds safe limits, the soft starter initiates a protective shutdown.

4. **Stop Phase (Optional)**:
   - For controlled stopping, the SCRs gradually decrease the voltage supplied to the motor, providing a smooth deceleration.

Key Features to Implement

1. **Voltage Ramp-Up Time**:
   - Adjustable (e.g., 0–10 seconds) to cater to different motor types and loads.
2. **Current Limiting**:
   - Set a maximum allowable current during startup to prevent damage.
3. **Built-In Protection**:
   - Overload, short-circuit, phase-loss, and thermal protection.
4. **User Interface**:
   - Simple interface with start/stop buttons and adjustable settings for ramp time, current limit, etc.
5. **Compact Design**:
   - Space-saving design to ensure suitability for a wide range of installations.

Control Algorithm

1. **Initialization**:
   - Check system status (e.g., temperature, input voltage).
   - Set initial firing angle for SCRs.
2. **Startup**:
   - Gradually reduce the firing angle to increase voltage supplied to the motor.
   - Monitor motor current to ensure it stays within safe limits.
3. **Steady-State Operation**:
   - Activate bypass contactor.
   - Continuously monitor system parameters.
4. **Shutdown**:
   - Gradually increase the firing angle for soft stopping (if enabled).
   - Open bypass contactor and SCRs.

Design Considerations

1. **Power Rating**:
   - Design for a range of motor ratings (e.g., 0.75–10 kW).

2. **Input Voltage**:
    - Compatible with 230V single-phase supply.
3. **Ambient Conditions**:
    - Operate reliably across wide temperature and humidity ranges.
4. **Scalability**:
    - Allow modular upgrades or customization.

COMPONENT SELECTION AND PRICE

1. Power Components

SCRs (Thyristors)

- **Function**: Control voltage by phase angle modulation during motor startup.
- **Selection Criteria**:
    - Voltage Rating: 2x the supply voltage (e.g., for 230V, select 600V SCRs for safety margin).
    - Current Rating: 1.5x the motor's full-load current.
    - Thermal Resistance: Ensure efficient heat dissipation.

Example SCR Models:

- **BT152-600R** (600V, 20A) for lower power motors. (77.73 INR)
- **T1600N20TOF** (1600V, 200A) for higher power motors. (Inquiry required)

Bypass Contactor

- **Function**: Bypasses the SCRs once the motor reaches full speed to minimize heat loss.
- **Selection Criteria**:
    - Voltage Rating: Same as supply voltage (230V).
    - Current Rating: Match the motor's full-load current.
    - Coil Voltage: Compatible with your control circuit (e.g., 230V AC or 24V DC).

Example Contactor Models:

- **Siemens 3RT1016** for up to 10A. (4809 INR)
- **Schneider LC1D32** for higher currents. (3264 INR)

EMI Filters

- **Function**: Reduces electromagnetic interference caused by SCR switching.
- **Selection Criteria**:
  - Voltage Rating: Match the supply voltage.
  - Attenuation: Based on regulatory requirements (e.g., CISPR 22/24).

Example EMI Filters:

- **Schaffner FN2090** series (for single-phase applications). (3463 INR)

2. Control Components

Microcontroller (MCU)

- **Function**: Controls SCR firing angles, monitors voltage/current, and handles fault conditions.
- **Selection Criteria**:
  - ADC Channels: For voltage and current sensing.
  - PWM Outputs: To control SCR firing pulses.
  - Processing Power: For real-time control algorithms.

Example Microcontrollers:

- **STM32F103C8**: Affordable, with sufficient GPIOs and ADC. (155 INR)
- **Texas Instruments TMS320F28027**: Optimized for motor control applications. (700 INR)

Voltage and Current Sensors

- **Function**: Monitor input voltage, motor voltage, and current.
- **Selection Criteria**:
  - Voltage Sensor: High accuracy for low voltage ranges.
  - Current Sensor: Handle peak startup currents (3–6x full load).

Example Sensors:

- Voltage: **ZMPT101B** (up to 250V AC). (77 INR)
- Current: **ACS712** (20A or 30A variants). (60 INR)

User Interface Components

- **Start/Stop Buttons**:
  - Example: **Schneider XB4** series. (1953 INR)
- **Rotary Potentiometer**:
  - For adjustable ramp-up time (e.g., 0–10s).
- **LED Indicators**:
  - For status and fault indication (e.g., Red for Fault, Green for Run).

3. Protection Components

Thermal Protection

- **Function**: Shuts down the system in case of overheating.
- **Selection Criteria**:
  - Thermistor: Place near SCRs and heatsinks.
  - Temperature Range: Operating range of -40°C to 125°C.

Example Components:

- **NTC Thermistor MF72** (inrush current limiting and thermal protection). (1 INR)

Surge Protection

- **Function**: Protects SCRs and other components from voltage spikes.
- **Selection Criteria**:
  - Voltage Rating: Match or exceed supply voltage.

Example Components:

- **MOV (Metal Oxide Varistor) S10K250** for 230V applications. (39 INR)

4. Cooling System

Heatsinks

- **Function**: Dissipates heat from SCRs.
- **Selection Criteria**:
  - Thermal Resistance: Match the power dissipation of SCRs.
  - Size: Compact yet efficient.

Example:

- **Fischer SK 104** series. (256 INR)

Cooling Fans

- **Function**: Provides forced air cooling for high-power designs.
- **Selection Criteria**:
  - Voltage: Match control circuit (12V/24V DC or 230V AC).
  - Airflow: Sufficient for heat dissipation (CFM rating).

Example Fans:

- **Delta Electronics AFB1212** (for DC cooling). (1870 INR)

5. Power Supply

DC Power Supply for Control Circuit

- **Function**: Powers the microcontroller and auxiliary circuits.
- **Selection Criteria**:
  - Input Voltage: 230V AC.
  - Output Voltage: 5V/12V DC for MCU and control circuits.

Example Power Supply:

- **Meanwell LRS-35-12** (12V, 3A) (936 INR)

6. Enclosure

- **Material**: Polycarbonate or sheet metal for durability.
- **IP Rating**: IP54 or higher for dust and water resistance.
- **Size**: Compact but spacious enough for easy assembly and heat dissipation.

7. Optional Features

- **Soft Stop Circuit**: Use an additional control mechanism to ramp down voltage smoothly.
- **Communication Interface**:
  - Add Modbus or RS485 for remote monitoring and control.

Power Electronics Section

Component	Part Number/ Spec	Quantity	Description
SCR	BT151-600R	2	25A, 600V SCR for bidirectional voltage control
Snubber Resistor	220Ω, 2W	2	For snubber network
Snubber Capacitor	0.1μF, 630V	2	For snubber network
MOV	S20K250	1	Surge protector for 220V AC input

Sensors

Component	Part Number/ Spec	Quantity	Description
Current Transformer (CT)	SCT-013-030	1	30A, 1V output CT for current sensing
NTC Thermistor	10kΩ (B=3950K)	1	Temperature sensing
Voltage Divider Resistors	100kΩ, 1/4W + 10kΩ, 1/4W	2 each	For AC voltage sensing

Control Unit

Component	Part Number/ Spec	Quantity	Description
Microcontroller	TMS320F28027	1	TI C2000 series microcontroller
Ceramic Capacitor	0.1μF, 50V	2	Decoupling capacitors
Push Buttons	Generic	2	Start and Stop buttons

Gate Driver Circuit

Component	Part Number/ Spec	Quantity	Description
Optoisolator	TLP250	2	Gate driver with isolation
Gate Resistor	220Ω, 1/4W	2	Gate current limiting resistor
Gate RC Network	R=10Ω, C=0.01μF	2 sets	For gate snubber protection

User Interface

Component	Part Number/ Spec	Quantity	Description
LEDs	Generic, 3mm (Red, Green)	4	Status and error indicators
Current-Limiting Resistors	470Ω, 1/4W	4	For LEDs

Power Supply Section

Component	Part Number/ Spec	Quantity	Description
Transformer	230V to 12V AC	1	Step-down transformer
Bridge Rectifier	DB107	1	1A, 1000V bridge rectifier

Smoothing Capacitor	1000µF, 25V	1	For DC filtering
Voltage Regulator	7805	1	5V linear regulator
Diodes	1N4007	2	Rectification and reverse polarity protection

Miscellaneous

Component	Part Number/ Spec	Quantity	Description
Heat Sink	Generic (sized for BT151-600R)	2	For SCRs to dissipate heat
PCB	Custom Design	1	Custom PCB for the entire circuit
Screw Terminals	Generic	4	For input/output connections
Wires	16 AWG	As Required	For interconnections

Based on the above list, the estimated cost per unit will vary depending on sourcing but typically ranges from **₹2,500 to ₹4,000** (Indian market prices) for small-scale production.

WORKING PRINCIPLE OF A THREE PHASE MOTOR SOFT STARTER AND CONTROL UNIT (200+ kW Motors)

A **soft starter** is an electrical device used to reduce the inrush current and torque during the startup of three-phase motors. It achieves this by gradually ramping up the motor voltage during the start-up phase and allowing the motor to reach its full speed smoothly. This approach reduces mechanical stress on the motor and connected equipment and minimizes electrical stress on the power grid.

Key Components of a Three-Phase Soft Starter

1. **Power Electronics**:
   - **Thyristors (SCRs)**: Main components for voltage control, arranged in anti-parallel pairs for each phase.
2. **Control Unit**:
   - Microcontroller or DSP (e.g., TMS320F28027) to manage the soft start process.
3. **Sensors**:
   - Voltage, current, and temperature sensors for feedback and protection.
4. **User Interface**:
   - Push buttons, displays, and communication ports (e.g., MODBUS) for configuration and monitoring.
5. **Protection Circuitry**:

- Overcurrent, overvoltage, and phase loss detection.

6. **Cooling System**:
    - Fans or liquid cooling to dissipate heat generated by the SCRs.

Operating Phases of a Soft Starter

1. Startup Phase

- **Initial Voltage Reduction**:
    - The SCRs initially supply a reduced voltage to the motor by controlling the phase angle of the AC waveform.
    - Voltage is ramped up gradually over time (configurable, e.g., 5–30 seconds).
- **Current Control**:
    - The current is monitored to ensure it does not exceed the set limits during the ramp-up phase.
    - Limits inrush current to protect the motor and reduce mechanical wear.
- **Acceleration Control**:
    - By controlling the voltage, the motor's speed gradually increases without sudden torque surges.
    - This prevents mechanical shocks to connected loads.

2. Running Phase

- **Bypass Contactors**:
    - Once the motor reaches full speed, the SCRs are bypassed by contactors to reduce power loss and heat generation.
    - The motor then operates directly on the mains voltage.
- **Monitoring**:
    - The system continuously monitors current, voltage, and temperature during motor operation.

3. Stopping Phase

- **Soft Stop**:
    - Voltage is gradually reduced during the stopping process (optional).
    - Prevents abrupt stops, reducing mechanical wear on connected equipment.

Control Unit Operation

1. **Startup Sequence**:
    - Upon pressing the start button:

        - **Parameter Check**: Verify supply voltage, phase availability, and ambient temperature.
        - **SCR Triggering**: Gradually increase the SCR firing angle to supply increasing voltage.
2. **Running Sequence**:
    - Once full speed is achieved:
        - **Bypass Activation**: Activate bypass contactors.
        - **Continuous Monitoring**: Ensure no overloads, phase imbalances, or overheating occur.
3. **Stop Sequence**:
    - On receiving the stop command:
        - **SCR Ramp-Down**: Gradually reduce voltage by adjusting the SCR firing angle.

Control Algorithm

1. **Input Parameters**:
    - Motor specifications (e.g., rated voltage, current, and speed).
    - User-configurable settings for start/stop ramp time, current limit, etc.
2. **Feedback and Monitoring**:
    - Continuously monitor:
        - Line voltage and current.
        - Motor temperature (via sensors).
        - Phase loss or imbalance.
3. **Error Handling**:
    - Trigger alarms or shutdown in case of:
        - Overcurrent/overvoltage.
        - High temperature or thermal overload.
        - Phase faults.
4. **Communication**:
    - MODBUS integration for remote configuration and monitoring.

Advantages of the Soft Starter

1. **Electrical Benefits**:
    - Reduces inrush current.

- Minimizes voltage dips in the power supply network.

2. **Mechanical Benefits**:
   - Lowers stress on couplings, gears, and belts.
   - Prevents torque surges and associated wear.
3. **Operational Benefits**:
   - Enhances motor lifespan.
   - Provides smoother and quieter motor operation.

Example Sequence

1. **Startup**:
   - Initial voltage = 30% of rated voltage.
   - Gradually ramp up to 100% over 10 seconds.
   - Monitor current to ensure it stays within safe limits.
2. **Running**:
   - Motor runs at full speed with bypass contactors engaged.
   - Continuous monitoring of parameters (e.g., load current and temperature).
3. **Stopping**:
   - Voltage reduced from 100% to 0% over 5 seconds (soft stop).
   - Motor comes to a smooth halt.

CONTROL ALGORITHM FOR TEXAS INSTRUMENTS TMS320F28027

```
1. Start
2. Initialize peripherals and system checks
3. Wait for Start button press
4. Check ambient conditions:
   a. Temperature
   b. Input voltage
   c. System diagnostics
5. If all conditions are OK, proceed to startup:
   a. Gradually increase SCR firing angle
   b. Monitor motor current to limit inrush
```

6. When full speed is reached, engage bypass contactor
7. Enter monitoring mode:
   a. Continuously check motor parameters
   b. Display system status
8. If Stop button is pressed, proceed to stopping:
   a. Gradually decrease SCR firing angle
   b. Disengage bypass contactor
9. Handle any faults or warnings:
   a. Overcurrent
   b. Undervoltage/Overvoltage
   c. Overtemperature
10. Stop motor and return to idle
11. End

PSEUDOCODE FOR MOTOR SWITCHING MCU TMS320F28027

Initialisation Phase

```
void init_system() {
    // Initialize GPIO for buttons and indicators
    gpio_init();

    // Initialize ADC for voltage, current, and temperature
sensing
    adc_init();

    // Initialize PWM for SCR control
    pwm_init();

    // Check for hardware faults
    if (check_hardware_faults()) {
        display_error("Hardware Fault");
        return;
```

```
    }

    // Initialize variables
    motor_status = IDLE;
}
```

Startup Phase

```
void startup_procedure() {
    if (start_button_pressed()) {
        // Check ambient conditions
        if (!check_conditions()) {
            display_error("Startup Conditions Not Met");
            return;
        }

        // Gradual voltage ramp-up
        for (int angle = START_ANGLE; angle <=
FULL_VOLTAGE_ANGLE; angle += STEP_SIZE) {
            set_pwm_angle(angle);
            delay_ms(RAMP_DELAY); // Time between steps
            if (check_faults()) {
                handle_fault();
                return;
            }
        }

        // Engage bypass contactor
        engage_bypass_contactor();
        motor_status = RUNNING;
    }
}
```

Monitoring Phase

```
void monitor_motor() {
    while (motor_status == RUNNING) {
        // Monitor voltage
        float voltage = read_adc_voltage();
        if (voltage < MIN_VOLTAGE || voltage > MAX_VOLTAGE) {
            display_error("Voltage Fault");
            handle_fault();
            return;
        }

        // Monitor current
        float current = read_adc_current();
        if (current > MAX_CURRENT) {
            display_error("Overcurrent Fault");
            handle_fault();
            return;
        }

        // Monitor temperature
        float temperature = read_adc_temperature();
        if (temperature > MAX_TEMPERATURE) {
            display_error("Overtemperature Fault");
            handle_fault();
            return;
        }

        // Update system status
        display_status(voltage, current, temperature);
    }
}
```

Stopping Phase

```
void stopping_procedure() {
    if (stop_button_pressed()) {
        // Gradual voltage ramp-down
        for (int angle = FULL_VOLTAGE_ANGLE; angle >=
START_ANGLE; angle -= STEP_SIZE) {
            set_pwm_angle(angle);
            delay_ms(RAMP_DELAY); // Time between steps
            if (check_faults()) {
                handle_fault();
                return;
            }
        }

        // Disengage bypass contactor
        disengage_bypass_contactor();
        motor_status = IDLE;
    }
}
```

Fault Handling

```
void handle_fault() {
    // Turn off PWM
    stop_pwm();

    // Disengage bypass contactor
    disengage_bypass_contactor();

    // Display fault and log if needed
    log_fault();
    motor_status = FAULT;
}
```

Key Parameters

- **START_ANGLE**: Initial firing angle (e.g., 90° for low voltage).
- **FULL_VOLTAGE_ANGLE**: Firing angle for full voltage (e.g., 0°).
- **STEP_SIZE**: Incremental step for angle adjustment.
- **RAMP_DELAY**: Time delay (e.g., 10–50 ms) between angle increments

STANDARD CODE FOR MCU TMS320F28027 IN C PROGRAMMING LANGUAGE (CODE GENERATED USING AI POWERED GPT; DEBUGGING NOT DONE)

```
#include "F2802x_Device.h" // Include device-specific header file

#include "F2802x_Examples.h" // Include example functions and macros


// Constants

#define START_ANGLE 90       // Starting SCR firing angle in degrees

#define FULL_VOLTAGE_ANGLE 0 // Full voltage SCR firing angle in degrees

#define STEP_SIZE 5          // Step size for ramping up/down

#define RAMP_DELAY 50        // Delay between steps in milliseconds


// Thresholds

#define MIN_VOLTAGE 180      // Minimum allowable voltage (V)

#define MAX_VOLTAGE 240      // Maximum allowable voltage (V)

#define MAX_CURRENT 10       // Maximum allowable current (A)

#define MAX_TEMPERATURE 75   // Maximum allowable temperature (°C)


// Function Prototypes

void init_system(void);

void init_peripherals(void);

void startup_procedure(void);
```

```
void monitor_motor(void);
void stopping_procedure(void);
void handle_fault(void);

float read_adc_voltage(void);
float read_adc_current(void);
float read_adc_temperature(void);
void set_pwm_angle(int angle);
void display_status(float voltage, float current, float
temperature);
void display_error(const char *error);
void engage_bypass_contactor(void);
void disengage_bypass_contactor(void);
int check_conditions(void);
int check_faults(void);
int start_button_pressed(void);
int stop_button_pressed(void);
void delay_ms(int milliseconds);

// Global Variables
volatile int motor_status = 0; // 0: IDLE, 1: RUNNING, 2:
FAULT

void main(void) {
    // Step 1: Initialize the system
    init_system();

    // Main loop
    while (1) {
        if (motor_status == 0) { // IDLE
            if (start_button_pressed()) {
```

```
                startup_procedure();
            }
        } else if (motor_status == 1) { // RUNNING
            monitor_motor();
        } else if (motor_status == 2) { // FAULT
            handle_fault();
        }

        if (stop_button_pressed()) {
            stopping_procedure();
        }
    }
}

// System Initialization
void init_system(void) {
    InitSysCtrl(); // Initialize system control (PLL,
watchdog, etc.)
    DINT;          // Disable CPU interrupts
    InitPieCtrl(); // Initialize PIE control registers
    InitPieVectTable(); // Initialize the PIE vector table

    init_peripherals(); // Initialize GPIO, ADC, PWM, etc.

    motor_status = 0; // Set initial motor status to IDLE
}

// Peripherals Initialization
void init_peripherals(void) {
    // GPIO Initialization
    EALLOW;
```

```
    GpioCtrlRegs.GPADIR.all |= 0x0003; // Set GPIO0 (Start)
and GPIO1 (Stop) as inputs
    EDIS;

    // ADC Initialization
    AdcRegs.ADCTRL1.bit.ACQ_PS = 6; // Set acquisition window
    AdcRegs.ADCTRL1.bit.CPS = 1;    // Set ADC clock prescaler
    AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1 = 1; // Enable interrupt
for sequence 1

    // PWM Initialization for SCR control
    EPwm1Regs.TBPRD = 2000;           // Set PWM period
    EPwm1Regs.CMPA.half.CMPA = 0;     // Set initial duty
cycle
    EPwm1Regs.TBCTL.bit.CTRMODE = 2;  // Up-down count mode
    EPwm1Regs.AQCTLA.bit.CAU = 2;     // Clear on compare up
    EPwm1Regs.AQCTLA.bit.CAD = 1;     // Set on compare down
}

// Startup Procedure
void startup_procedure(void) {
    if (check_conditions()) {
        for (int angle = START_ANGLE; angle >=
FULL_VOLTAGE_ANGLE; angle -= STEP_SIZE) {
            set_pwm_angle(angle);
            delay_ms(RAMP_DELAY);

            if (check_faults()) {
                handle_fault();
                return;
            }
        }
```

```
        engage_bypass_contactor();
        motor_status = 1; // Running
    } else {
        display_error("Startup Conditions Not Met");
    }
}

// Monitor Motor Parameters
void monitor_motor(void) {
    float voltage, current, temperature;

    while (motor_status == 1) {
        voltage = read_adc_voltage();
        current = read_adc_current();
        temperature = read_adc_temperature();

        if (voltage < MIN_VOLTAGE || voltage > MAX_VOLTAGE ||
current > MAX_CURRENT || temperature > MAX_TEMPERATURE) {
            display_error("Fault Detected");
            handle_fault();
            return;
        }

        display_status(voltage, current, temperature);
    }
}

// Stopping Procedure
void stopping_procedure(void) {
    for (int angle = FULL_VOLTAGE_ANGLE; angle <= START_ANGLE;
angle += STEP_SIZE) {
```

```
        set_pwm_angle(angle);
        delay_ms(RAMP_DELAY);
    }
    disengage_bypass_contactor();
    motor_status = 0; // IDLE
}

// Fault Handling
void handle_fault(void) {
    disengage_bypass_contactor();
    set_pwm_angle(START_ANGLE); // Turn off SCR
    motor_status = 2;           // Fault state
}

// Helper Functions (Implement based on hardware specifics)
float read_adc_voltage(void) { return 230.0; } // Placeholder
float read_adc_current(void) { return 5.0; }   // Placeholder
float read_adc_temperature(void) { return 45.0; } //
Placeholder

void set_pwm_angle(int angle) {
    int duty = (2000 * angle) / 180; // Convert angle to duty
cycle
    EPwm1Regs.CMPA.half.CMPA = duty;
}

void display_status(float voltage, float current, float
temperature) {
    // Placeholder for displaying status on an LCD or serial
monitor
}
```

```
void display_error(const char *error) {
    // Placeholder for displaying error messages
}

void engage_bypass_contactor(void) {
    // Placeholder for activating the contactor
}

void disengage_bypass_contactor(void) {
    // Placeholder for deactivating the contactor
}

int check_conditions(void) {
    return 1; // Placeholder for condition checks
}

int check_faults(void) {
    return 0; // Placeholder for fault checks
}

int start_button_pressed(void) {
    return GpioDataRegs.GPADAT.bit.GPIO0; // Read GPIO0
}

int stop_button_pressed(void) {
    return GpioDataRegs.GPADAT.bit.GPIO1; // Read GPIO1
}

void delay_ms(int milliseconds) {
    DELAY_US(milliseconds * 1000);
}
```

Notes:

1. **ADC Readings**: Replace placeholder `read_adc_*` functions with actual ADC channel mappings for voltage, current, and temperature sensing.
2. **PWM Configuration**: Adjust `EPwm1Regs` configuration as per your SCR module's requirements.
3. **Error Handling**: Extend error messages and logs to a UART, CAN bus, or an LCD for detailed diagnostics.
4. **Testing**: Simulate the algorithm using TI's hardware debugging tools or a motor test bench.

MODBUS INTEGRATION FOR THREE-PHASE SOFT STARTER AND CONTROL UNIT IN INDUSTRIAL APPLICATIONS

1. Understanding MODBUS Integration

- **Protocol**: MODBUS is a serial communication protocol widely used in industrial applications.
- **Modes**: Implement either **RTU (Remote Terminal Unit)** or **ASCII** mode based on your application.
- **Communication Medium**: Use RS485 or RS232 for physical layer communication.

2. Hardware Requirements

1. **RS485 Transceiver**: For robust communication over industrial distances (e.g., MAX485 or SN65HVD08).
2. **UART Pins**:
    - Use SCIA (Serial Communication Interface A) on the F28027 for MODBUS communication.
3. **Termination Resistors**:
    - Use 120-ohm resistors at the ends of the RS485 bus for signal integrity.

3. Software Design for MODBUS

Features to Implement:

1. **MODBUS RTU Frame**:
    - Slave ID
    - Function Code
    - Data
    - CRC16

2. **Slave Implementation**:
   - Respond to MODBUS read/write commands.
3. **CRC Calculation**:
   - Use standard CRC-16 for MODBUS RTU.

MODBUS Integration Codes:

Libraries

```
#include "F2802x_Device.h"

#include "F2802x_Examples.h"
```

Global Variables

```
#define SLAVE_ID 1 // Set your MODBUS slave ID


// UART Buffer

#define UART_BUFFER_SIZE 256

volatile uint8_t uart_rx_buffer[UART_BUFFER_SIZE];

volatile uint8_t uart_rx_index = 0;


// MODBUS CRC Table (Optional: Pre-computed for performance)

const uint16_t modbus_crc_table[256] = { /* CRC table values
*/ };


// Flags

volatile uint8_t modbus_request_received = 0;


// MODBUS Registers

uint16_t holding_registers[10]; // Example: Holding Registers
for soft starter
```

Function Prototypes

```
void init_modbus_uart(void);

void send_modbus_response(uint8_t *response, uint8_t length);

uint16_t calculate_modbus_crc(uint8_t *data, uint8_t length);
```

```
void modbus_request_handler(void);
void parse_modbus_request(uint8_t *request, uint8_t length);
```

UART Initialisation

```
void init_modbus_uart(void) {
    // GPIO Configuration for UART
    EALLOW;
    GpioCtrlRegs.GPAPUD.bit.GPIO28 = 0; // Enable pull-up for RX
    GpioCtrlRegs.GPAPUD.bit.GPIO29 = 0; // Enable pull-up for TX
    GpioCtrlRegs.GPAMUX2.bit.GPIO28 = 1; // Configure GPIO28 as SCIRXDA
    GpioCtrlRegs.GPAMUX2.bit.GPIO29 = 1; // Configure GPIO29 as SCITXDA
    EDIS;

    // UART Configuration
    SciaRegs.SCICCR.all = 0x07; // 1 stop bit, 8 data bits, no parity
    SciaRegs.SCICTL1.all = 0x03; // Enable TX and RX
    SciaRegs.SCICTL2.bit.TXINTENA = 0; // Disable TX interrupt
    SciaRegs.SCICTL2.bit.RXBKINTENA = 1; // Enable RX interrupt
    SciaRegs.SCIHBAUD = 0x00; // Set baud rate (9600 as example)
    SciaRegs.SCILBAUD = 0xA0; // Set baud rate (9600 as example)
    SciaRegs.SCICTL1.bit.SWRESET = 1; // Release SCI from reset
}
```

MODBUS CRC Calculation

```
uint16_t calculate_modbus_crc(uint8_t *data, uint8_t length) {
    uint16_t crc = 0xFFFF;
```

```
    for (int i = 0; i < length; i++) {
        crc ^= data[i];
        for (int j = 0; j < 8; j++) {
            if (crc & 0x0001)
                crc = (crc >> 1) ^ 0xA001;
            else
                crc >>= 1;
        }
    }
    return crc;
}
```

MODBUS Request Handler

```
void modbus_request_handler(void) {
    if (!modbus_request_received) return;

    uint16_t received_crc, calculated_crc;
    uint8_t request_length = uart_rx_index - 2; // Exclude CRC
bytes
    received_crc = (uart_rx_buffer[uart_rx_index - 1] << 8) |
uart_rx_buffer[uart_rx_index - 2];
    calculated_crc = calculate_modbus_crc(uart_rx_buffer,
request_length);

    if (received_crc == calculated_crc) {
        parse_modbus_request(uart_rx_buffer, request_length);
    } else {
        // Send Error: CRC Mismatch
    }

    uart_rx_index = 0; // Reset buffer
    modbus_request_received = 0;
}
```

MODBUS Request Parsing

```
void parse_modbus_request(uint8_t *request, uint8_t length) {
    if (request[0] != SLAVE_ID) return; // Ignore non-matching slave IDs

    uint8_t function_code = request[1];
    switch (function_code) {
        case 0x03: // Read Holding Registers
            // Process read request and send response
            break;

        case 0x06: // Write Single Register
            // Process write request and update register
            break;

        default:
            // Send Error: Unsupported Function Code
            break;
    }
}
```

Send MODBUS Response

```
void send_modbus_response(uint8_t *response, uint8_t length) {
    uint16_t crc = calculate_modbus_crc(response, length);
    response[length] = crc & 0xFF;       // Append CRC Low Byte
    response[length + 1] = (crc >> 8);   // Append CRC High Byte

    for (int i = 0; i < length + 2; i++) {
        while (SciaRegs.SCICTL2.bit.TXRDY == 0); // Wait for TX buffer to be ready
        SciaRegs.SCITXBUF = response[i];
```

```
    }
}
```

Testing and Debugging

1. **MODBUS Tester**: Use a MODBUS Master software tool like:
    - **QModMaster**
    - **ModScan**
2. **RS485 Adapter**: Connect your microcontroller to a PC using an RS485-to-USB converter.
3. **Check CRC and Frames**: Ensure correct CRC calculation and response handling.

HARDWARE DEBUGGING FOR THREE-PHASE HIGH POWER MOTOR CONTROL AND SWITCHING SYSTEMS

1. Key Aspects to Consider in Hardware Debugging

a) Power Circuitry Debugging

- **Three-Phase Power Supply**:
    - Ensure proper alignment and balance in the three-phase power supply.
    - Use an oscilloscope to check voltage levels, phase alignment, and any unwanted harmonics.
    - Monitor power factor and efficiency of the machine.
- **Inverter & Switching Circuit**:
    - Debug the inverter output using gate-drive waveforms, checking for proper switching (on/off timing, dead time, etc.).
    - Inspect voltage spikes and overshoots with a high-speed oscilloscope.
    - Check current flowing into and out of the motor with shunt resistors or current transducers.

b) Control Loop Debugging

- **Microcontroller or Controller**:
    - Debugging embedded control algorithms like speed, torque, and power factor regulation.
    - Check for communication between the microcontroller and peripherals (such as Hall Effect sensors or encoders) using a logic analyser.

- **Feedback System**:
    - Verify encoder signals or other feedback components.
    - Ensure proper feedback loop stability using tools like Bode plot analysis.

c) Protection Circuitry Debugging

- **Overcurrent & Overvoltage Protections**:
    - Verify correct functioning of overcurrent and overvoltage detection circuits.
    - Check tripping thresholds and response time with a high-speed oscilloscope.
- **Ground Fault Protection**:
    - Test the ground fault detection to ensure proper shutdown and isolation.
    - Use isolation transformers and differential current sensors for grounding checks.

d) Cooling & Thermal Management Debugging

- **Thermal Sensors**:
    - Inspect the temperature sensors used for monitoring stator and rotor temperatures.
    - Ensure that cooling fans, liquid cooling systems, or heat sinks are functioning correctly.
    - Use thermal cameras to detect hotspots.

e) Gate Driver & IGBT Debugging

- **Gate Drive Waveforms**:
    - Use an oscilloscope to check gate-drive voltages for IGBTs and verify proper switching timings.
- **Cross-Conduction**:
    - Identify potential cross-conduction or shoot-through in the IGBT switches with detailed current analysis.

2. Tools for Hardware Debugging

- **Oscilloscope**:
    - High-speed oscilloscope with advanced triggering capabilities to capture switching transients, voltage spikes, and current waveforms.
- **Logic Analyzer**:
    - To debug communication between controllers, sensors, and other components.
- **Current Probes**:

    - Use AC/DC current probes to monitor high current loops.
- **Power Analyzer**:
    - For power measurement and monitoring efficiency and harmonic distortions.
- **Thermal Camera**:
    - For quick identification of overheating components.
- **Multimeter**:
    - For basic voltage, current, and resistance measurements.
- **In-Circuit Emulator**:
    - To debug the microcontroller or DSP running the control algorithms.

3. Recommended Debugging Steps

a) Initial Power Supply and Phase Checks

- **Measure Line Voltages**:
    - Ensure that all three phases of the power supply have equal voltage levels and minimal deviations.
- **Phase Rotation**:
    - Confirm that the rotation of each phase matches the expected direction.
- **Power Quality**:
    - Check voltage stability and the absence of harmonics using a power analyser.

b) Inverter Switching Waveforms

- **Check Switching Signals**:
    - Capture gate drive signals using an oscilloscope and ensure proper pulse widths, dead times, and switching behaviour.
- **Detect Spikes**:
    - Inspect for high-frequency noise, voltage spikes, and cross-conduction during switching.

c) Current and Load Testing

- **Monitor Motor Current**:
    - Use current probes to measure the current flowing into the motor. Look for any irregularities or current imbalances.
- **Torque and Power Analysis**:
    - Calculate the torque and power delivered to the motor to ensure they match expected values.

- **Efficiency Monitoring**:
  - Compare input power to output power and calculate efficiency, and ensure minimal losses.

d) Feedback Loop Verification

- **Encoder Signals**:
  - Ensure encoder signals (if used) have no missed pulses and operate within acceptable range.
- **Sensor Data**:
  - Check the integrity of analog and digital sensor signals feeding into the microcontroller.

e) Thermal Management System

- **Temperature Sensor Checks**:
  - Verify readings from thermal sensors attached to key components like the stator, rotor, and IGBT junctions.
- **Cooling System Debugging**:
  - Inspect fans, liquid cooling loops, and heat sinks for efficient heat dissipation.

4. Debugging Using Software Tools

- **Code Composer Studio (CCS)**:
  - Use the **Code Composer Studio** IDE for microcontroller debugging, where you can step through code, watch variable states, and monitor real-time system performance.
- **JTAG/SWD Debugger**:
  - Employ a JTAG/SWD debugger to interface directly with the microcontroller and read/write memory.

5. Key Considerations During Hardware Debugging

- **Protective Measures**:
  - Ensure you use adequate protection circuitry like fuses, surge arrestors, and circuit breakers.
- **Document Findings**:
  - Keep detailed records of any findings, signal behaviour, error logs, and improvements made during the debugging process.

OTHER IMPORTANT AND RELEVENT DATA

All the codes shown in this document are generated using an Artificial Intelligence powered Generative Pre-training Transformer (GPT). No human debugging has been implemented yet and hence the codes might be way off and full of syntax, logical or randomised errors.

All costs shown in this document regarding Indian Standard certifications are taken from the Bureau of Indian Standards (BIS) website. Legitimacy of such costs can be seen in the website of the organisation concerned.

All costs shown in this document apart from BIS certification costs are taken from various sources over the internet including legitimate electronic traders Mouser Electronics India Ltd. And others. Prices of such equipment is subject to change without prior notice.