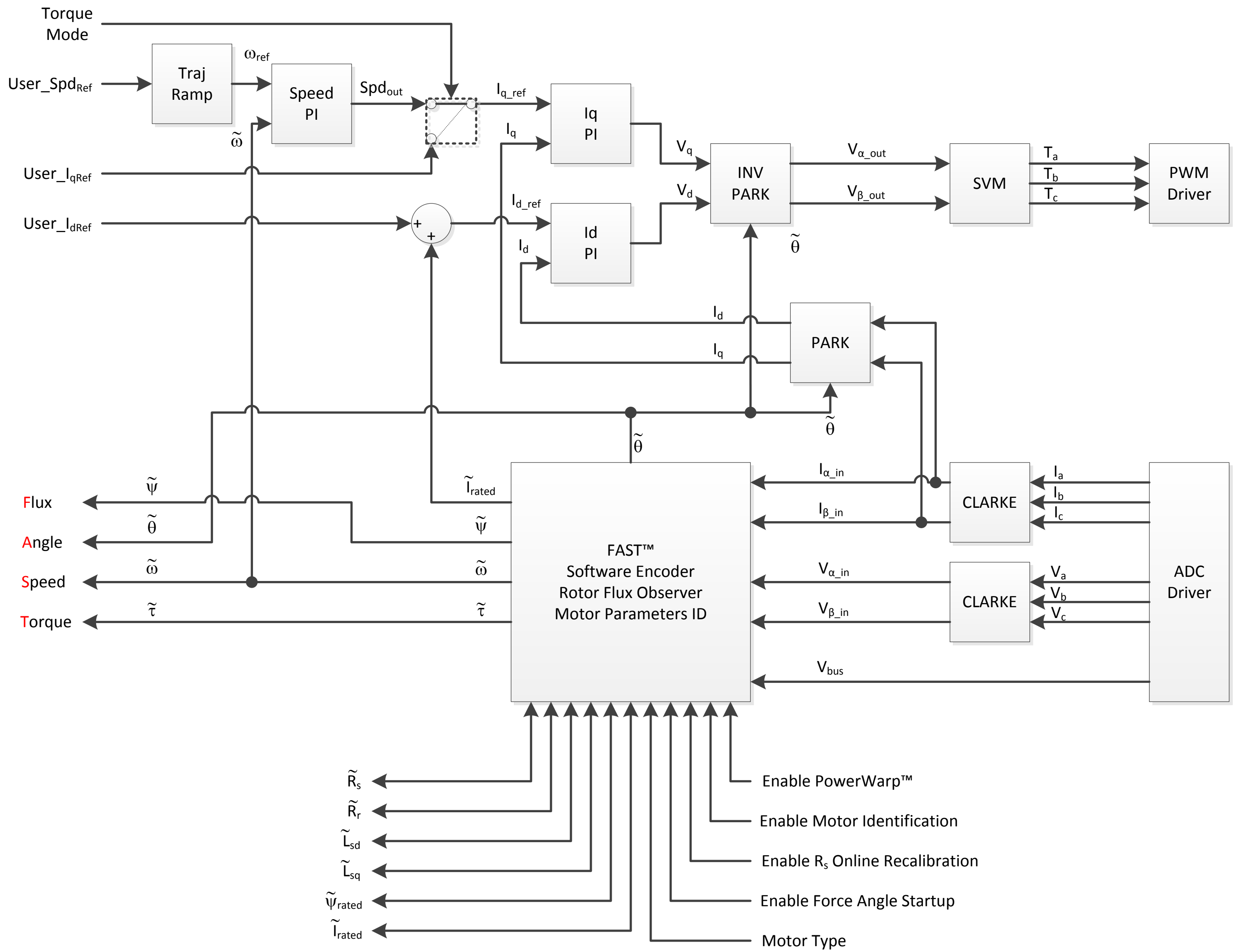
How to modify Motorware to run your own InstaSPIN-FOC board

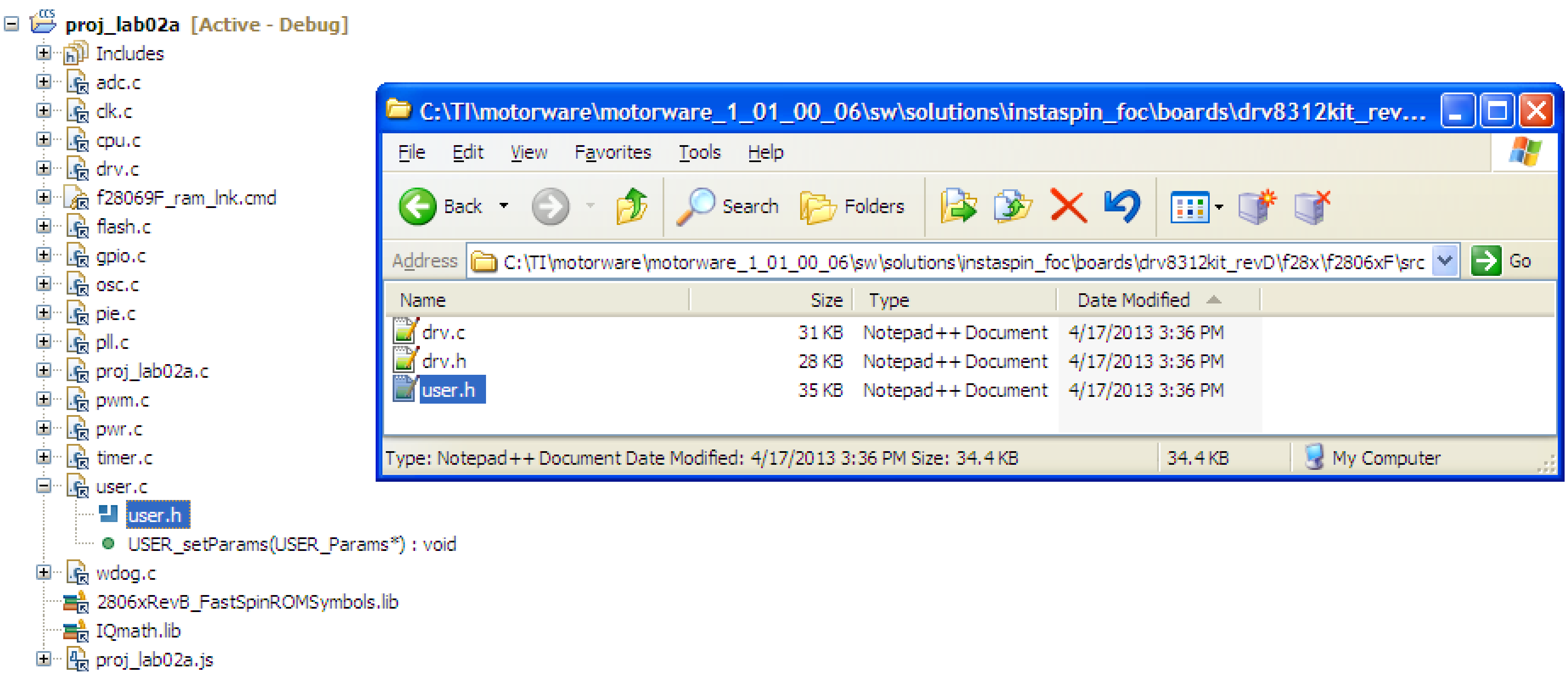
Texas Instruments

1. “user.h” parameters overview

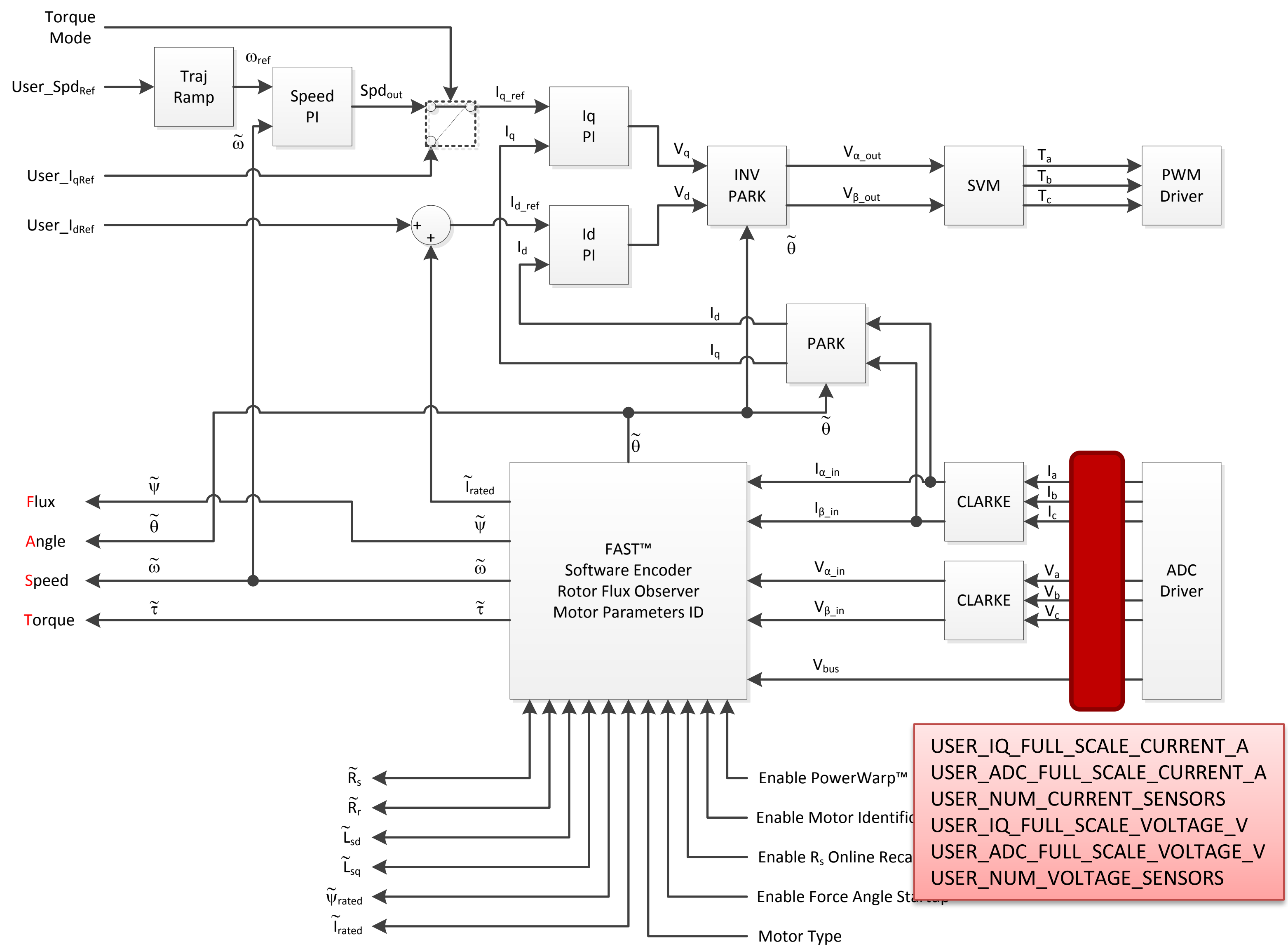
InstaSPIN-FOC Block Diagram



InstaSPIN-FOC User Parameters

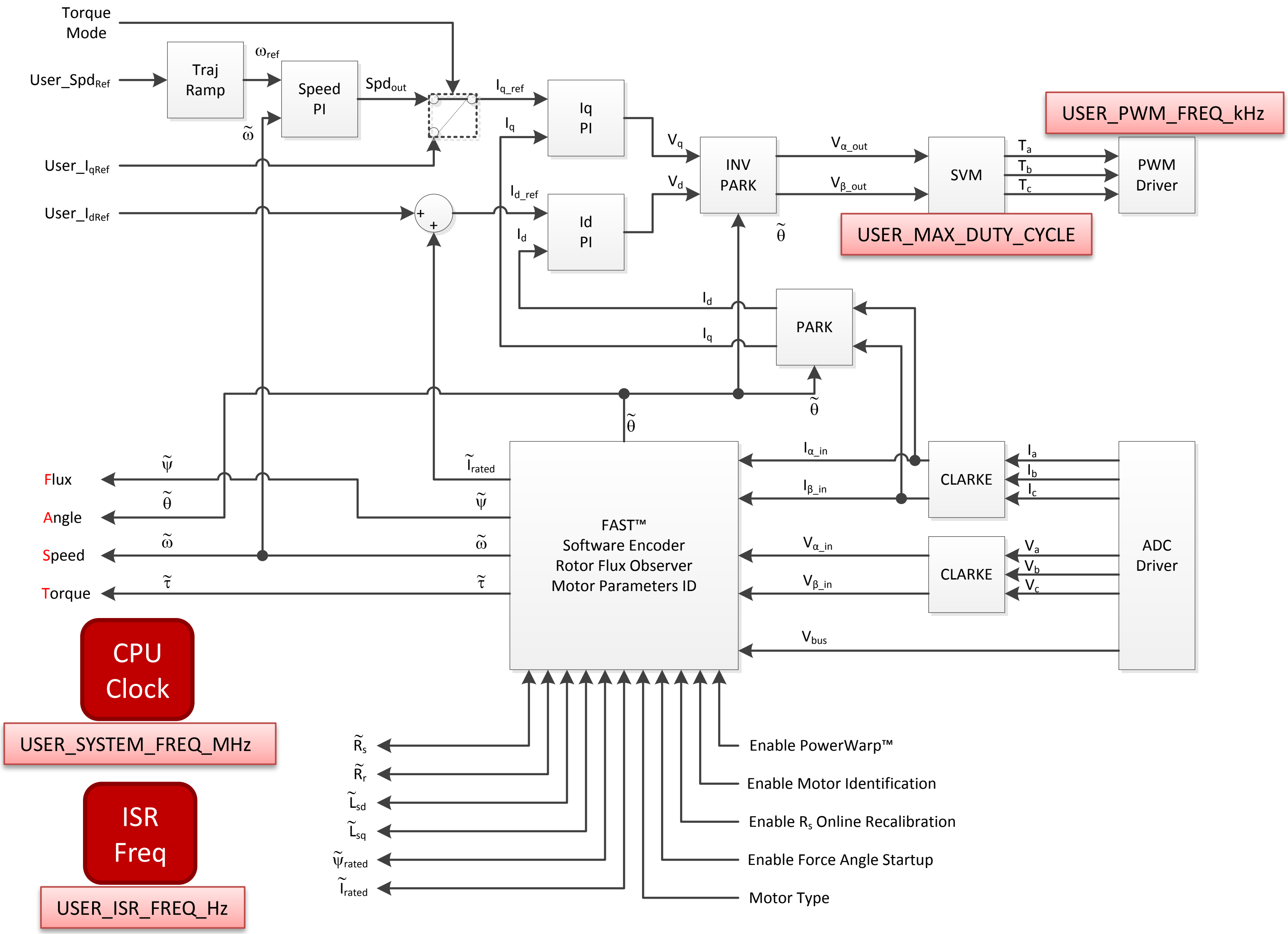


Currents and Voltage Parameters

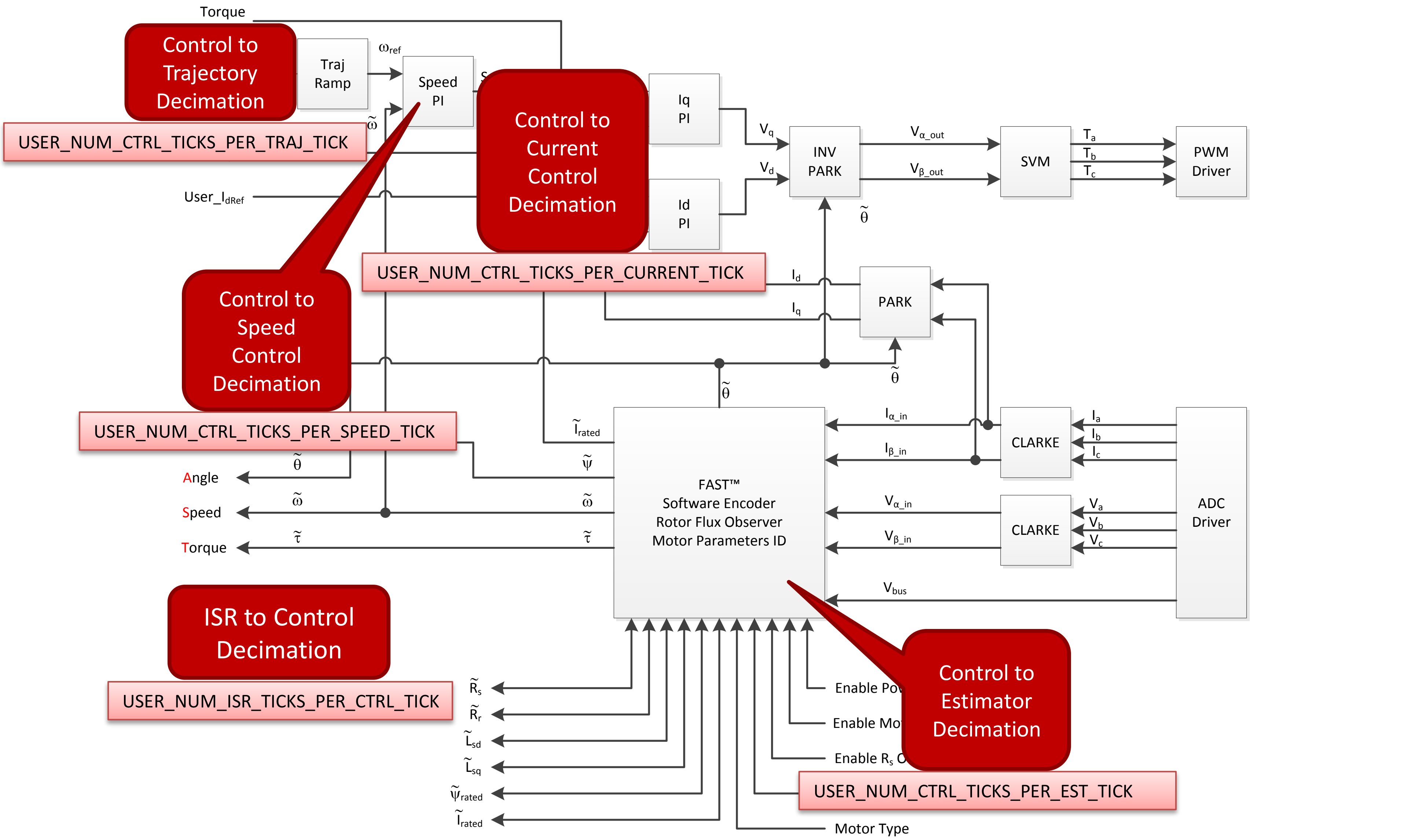


USER_IQ_FULL_SCALE_CURRENT_A
USER_ADC_FULL_SCALE_CURRENT_A
USER_NUM_CURRENT_SENSORS
USER_IQ_FULL_SCALE_VOLTAGE_V
USER_ADC_FULL_SCALE_VOLTAGE_V
USER_NUM_VOLTAGE_SENSORS

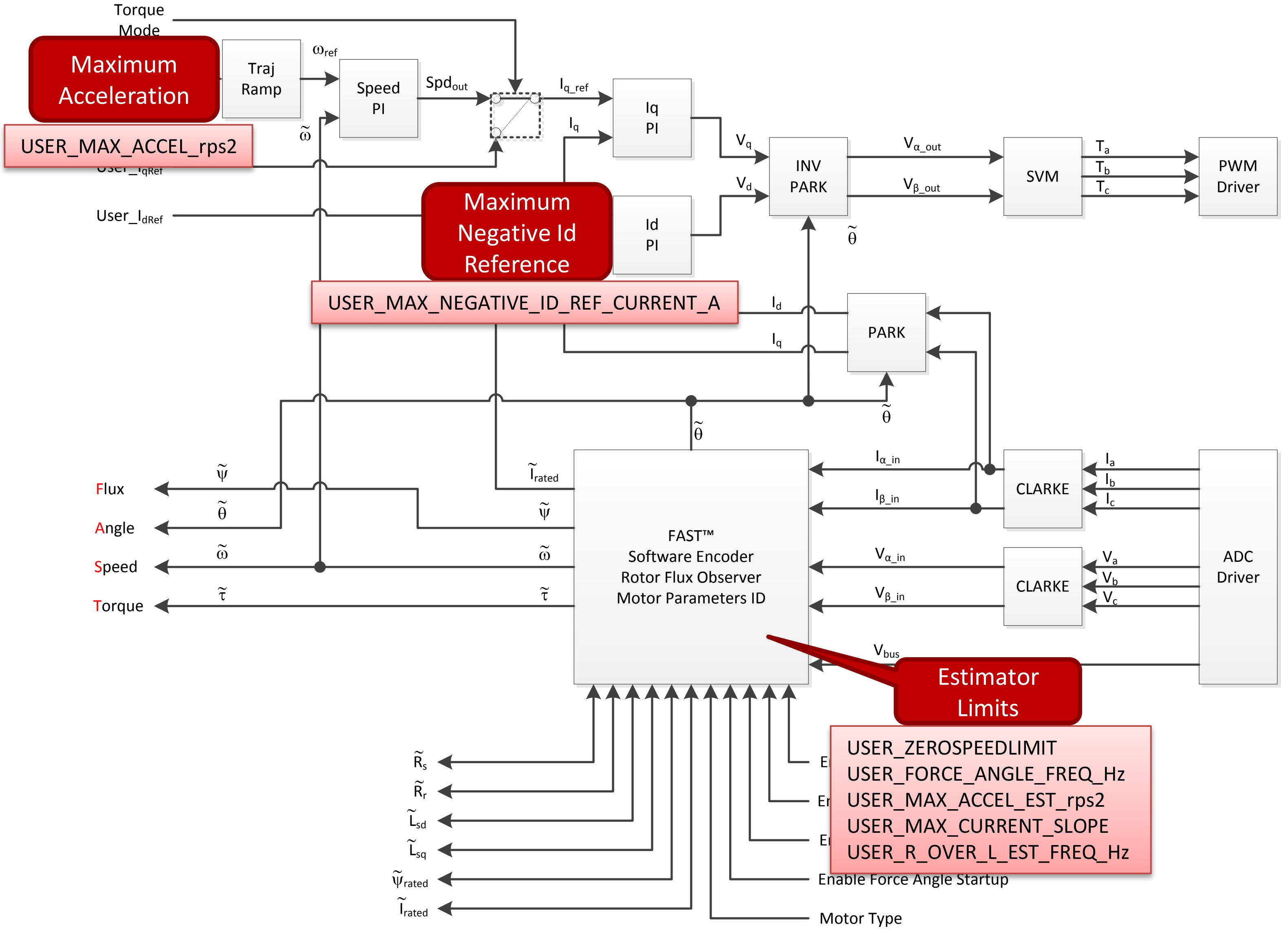
Clocks Parameters



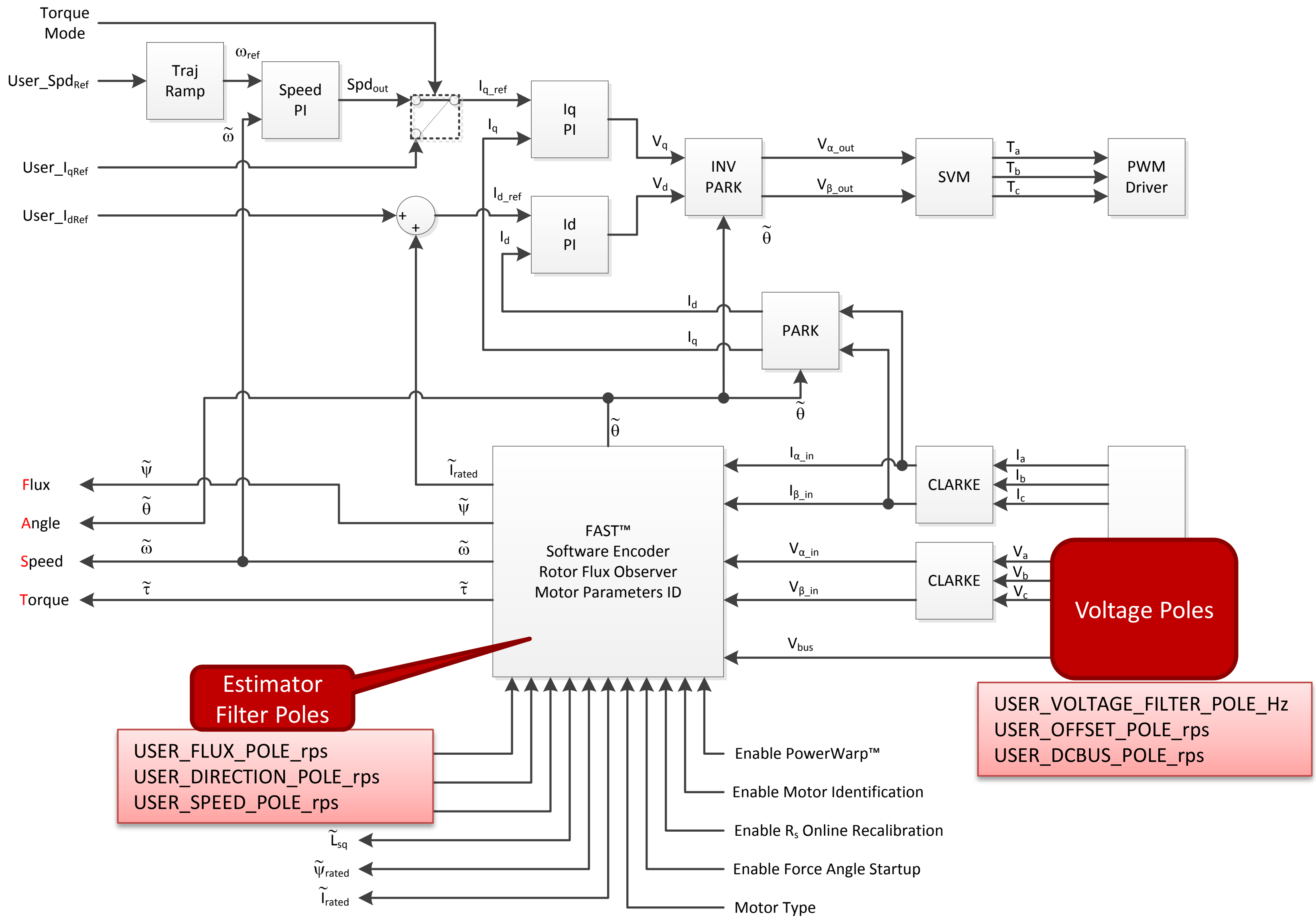
Decimation Parameters



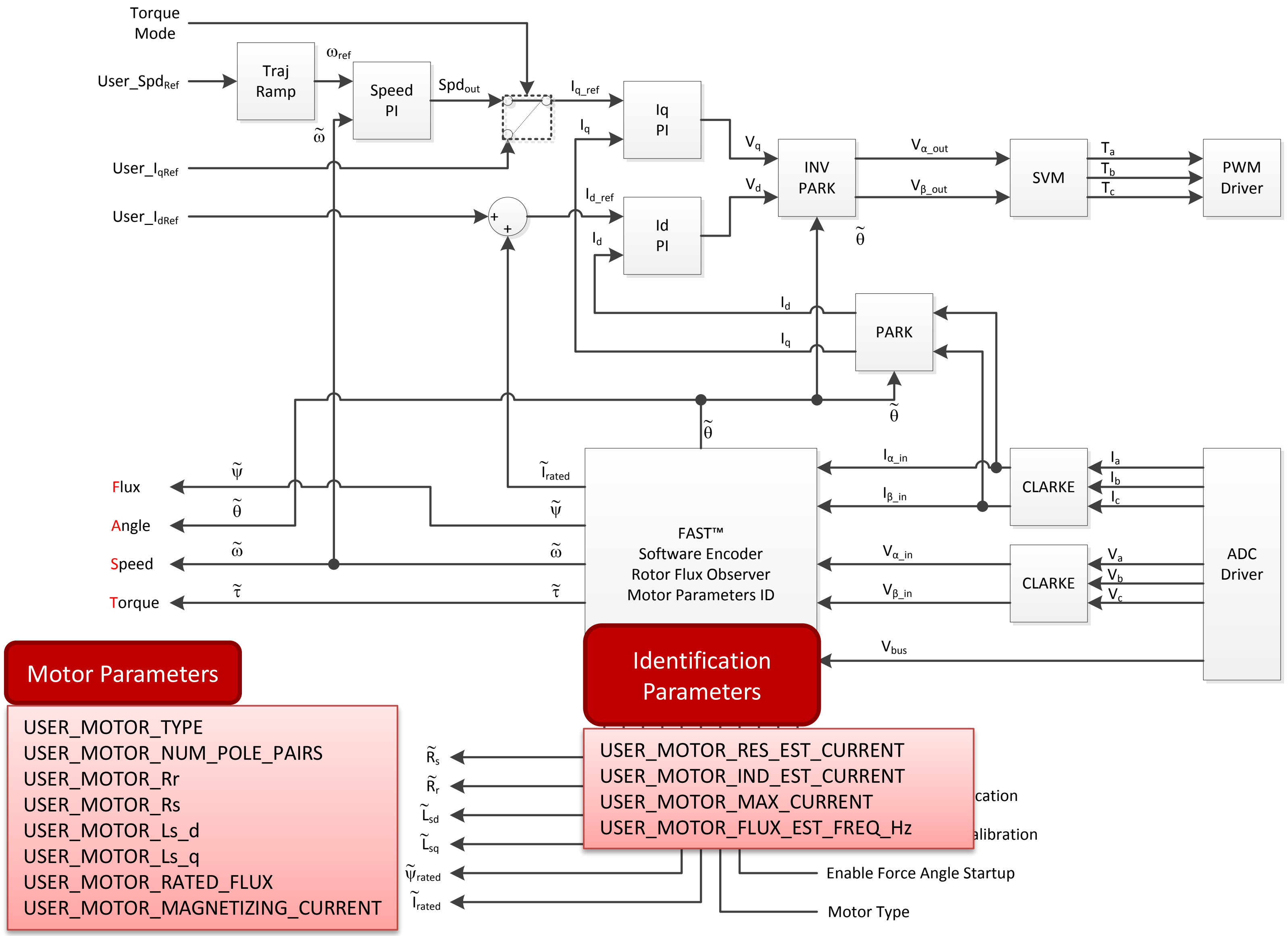
Limit Parameters



Filter Pole Parameters



Motor and Identification Parameters



2. HW/SW Preparation

- Analog inputs configuration
- PWM outputs configuration
- Voltage feedback configuration
- Current feedback gain and polarity configuration
- Number of shunt resistors configuration
- PWM dead-time configuration
- Voltage filter pole configuration
- 3-phase ADC current/voltage offsets configuration
- Software with hardware polarity configuration

2-1. Analog inputs configuration

- Modify motor current/voltage and DC-bus voltage sampling ADC channels according to your own schematics design -- Open "**HAL_setupAdc**" function in "**hal.c**":

```
//configure the SOCs for hvkit_rev1p1
// EXT IA-FB
ADC_setSocChanNumber(obj->adcHandle,ADC_SocNumber_0,ADC_SocChanNumber_A1); //U相电流采样ADC通道设置
ADC_setSocTrigSrc(obj->adcHandle,ADC_SocNumber_0,ADC_SocTrigSrc_EPWM1_ADCSOCA);
ADC_setSocSampleDelay(obj->adcHandle,ADC_SocNumber_0,ADC_SocSampleDelay_9_cycles);

// EXT IA-FB
// Duplicate conversion due to ADC Initial Conversion bug (SPRZ342)
ADC_setSocChanNumber(obj->adcHandle,ADC_SocNumber_1,ADC_SocChanNumber_A1); //U相电流采样ADC通道设置
ADC_setSocTrigSrc(obj->adcHandle,ADC_SocNumber_1,ADC_SocTrigSrc_EPWM1_ADCSOCA);
ADC_setSocSampleDelay(obj->adcHandle,ADC_SocNumber_1,ADC_SocSampleDelay_9_cycles);

// EXT IB-FB
ADC_setSocChanNumber(obj->adcHandle,ADC_SocNumber_2,ADC_SocChanNumber_B1); //V相电流采样ADC通道设置
ADC_setSocTrigSrc(obj->adcHandle,ADC_SocNumber_2,ADC_SocTrigSrc_EPWM1_ADCSOCA);
ADC_setSocSampleDelay(obj->adcHandle,ADC_SocNumber_2,ADC_SocSampleDelay_9_cycles);

// EXT IC-FB
ADC_setSocChanNumber(obj->adcHandle,ADC_SocNumber_3,ADC_SocChanNumber_A3); //W相电流采样ADC通道设置
ADC_setSocTrigSrc(obj->adcHandle,ADC_SocNumber_3,ADC_SocTrigSrc_EPWM1_ADCSOCA);
ADC_setSocSampleDelay(obj->adcHandle,ADC_SocNumber_3,ADC_SocSampleDelay_9_cycles);

// ADC-Vhb1
ADC_setSocChanNumber(obj->adcHandle,ADC_SocNumber_4,ADC_SocChanNumber_B7); //U相电压采样ADC通道设置
ADC_setSocTrigSrc(obj->adcHandle,ADC_SocNumber_4,ADC_SocTrigSrc_EPWM1_ADCSOCA);
ADC_setSocSampleDelay(obj->adcHandle,ADC_SocNumber_4,ADC_SocSampleDelay_9_cycles);

// ADC-Vhb2
ADC_setSocChanNumber(obj->adcHandle,ADC_SocNumber_5,ADC_SocChanNumber_B6); //V相电流采样ADC通道设置
ADC_setSocTrigSrc(obj->adcHandle,ADC_SocNumber_5,ADC_SocTrigSrc_EPWM1_ADCSOCA);
ADC_setSocSampleDelay(obj->adcHandle,ADC_SocNumber_5,ADC_SocSampleDelay_9_cycles);

// ADC-Vhb3
ADC_setSocChanNumber(obj->adcHandle,ADC_SocNumber_6,ADC_SocChanNumber_B4); //W相电流采样ADC通道设置
ADC_setSocTrigSrc(obj->adcHandle,ADC_SocNumber_6,ADC_SocTrigSrc_EPWM1_ADCSOCA);
ADC_setSocSampleDelay(obj->adcHandle,ADC_SocNumber_6,ADC_SocSampleDelay_9_cycles);

// VDCBUS
ADC_setSocChanNumber(obj->adcHandle,ADC_SocNumber_7,ADC_SocChanNumber_A7); //直流母线采样ADC通道设置
ADC_setSocTrigSrc(obj->adcHandle,ADC_SocNumber_7,ADC_SocTrigSrc_EPWM1_ADCSOCA);
ADC_setSocSampleDelay(obj->adcHandle,ADC_SocNumber_7,ADC_SocSampleDelay_9_cycles);
```

2-2. PWM outputs configuration

- Modify PWM channels according to your own schematics design -- Open "**HAL_setupGpios**" function in "**hal.c**":

```
// PWM1
GPIO_setMode(obj->gpioHandle,GPIO_Number_0,GPIO_0_Mode_EPWM1A);

// PWM2
GPIO_setMode(obj->gpioHandle,GPIO_Number_1,GPIO_1_Mode_EPWM1B);

// PWM3
GPIO_setMode(obj->gpioHandle,GPIO_Number_2,GPIO_2_Mode_EPWM2A);

// PWM4
GPIO_setMode(obj->gpioHandle,GPIO_Number_3,GPIO_3_Mode_EPWM2B);

// PWM5
GPIO_setMode(obj->gpioHandle,GPIO_Number_4,GPIO_4_Mode_EPWM3A);

// PWM6
GPIO_setMode(obj->gpioHandle,GPIO_Number_5,GPIO_5_Mode_EPWM3B);
```

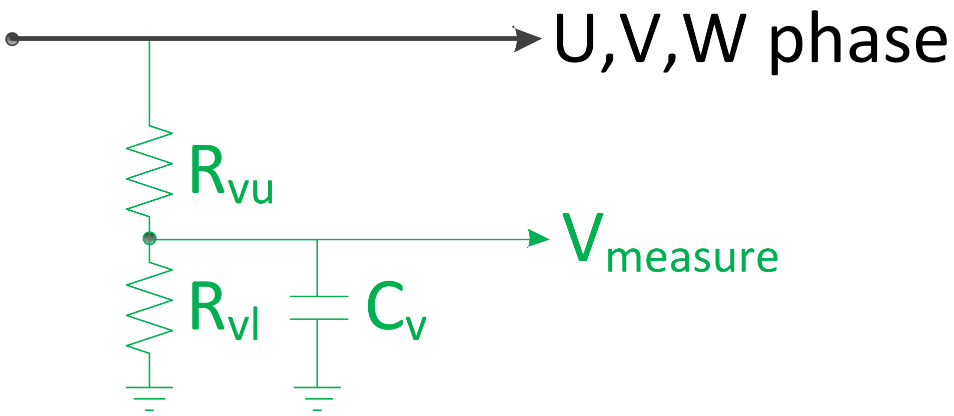
2-3. Voltage feedback configuration

USER_ADC_FULL_SCALE_VOLTAGE_V

- **Hardware dependent full scale voltage with respect to the maximum voltage of the A/D being used.**

$$USER\_ADC\_FULL\_SCALE\_VOLTAGE\_V = \frac{R_{vl}+R_{vu}}{R_{vl}} \cdot V_{adc\_max}$$

- **V_{adc_max} – A/D's maximum voltage (i.e. 3.3V)**



USER_IQ_FULL_SCALE_VOLTAGE_V

- **User selectable voltage scaling.**
- **Defines full scale value for the IQ30 variable of Voltage inside the system. All voltages are converted into (pu) based on the ratio to this value.**
  - **This value MUST be larger than the maximum value of any voltage calculated inside the estimator system otherwise the value can saturate and roll-over, causing an inaccurate value.**
  - **This value is OFTEN greater than the maximum measured ADC value, especially with high Bemf motors operating at higher than rated speeds.**
  - **If the motor is operated at multiples of its base voltage, in field weakening, then USER_IQ_FULL_SCALE_VOLTAGE_V must be increased.**

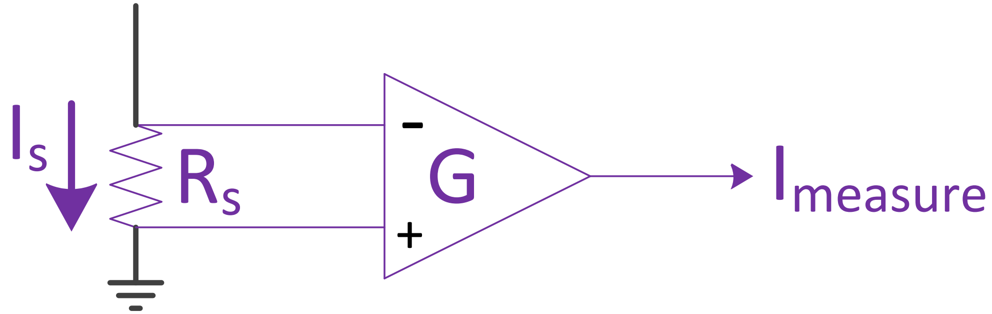
2-4. Current feedback gain and polarity configuration

USER_ADC_FULL_SCALE_CURRENT_A

- **Hardware dependent full scale current with respect to the maximum A/D voltage.**

$$\text{USER\_ADC\_FULL\_SCALE\_CURRENT\_A} = \frac{V_{adc\_max}}{R_S \cdot G}$$

- **V_{adc_max} – A/D's maximum voltage (i.e. 3.3V)**



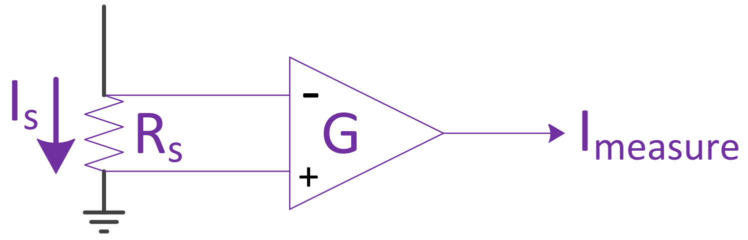
USER_IQ_FULL_SCALE_CURRENT_A

- **User selectable current scaling.**
- **All currents are converted into (pu) based on the ratio to this value.**
  - Prevent roll-over by keeping this value greater than half of USER_ADC_FULL_SCALE_CURRENT_A.

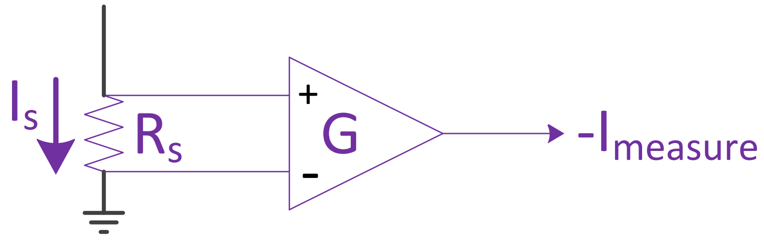
ADC Polarity

- **static inline void HAL_readAdcData()** in "**hal.h**"

  – Positive polarity for op-amp feedback



  – Negative polarity for op-amp feedback



ADC Polarity change in "hal.h"

In functions "**HAL_readAdcData**" and "**HAL_updateAdcBias**" :

- Positive polarity for x = 0, 1, and 2
  - "value" must be **POSITIVE**
  - "bias" must be **+**=

pAdcData->I.value[x] = value; (**HAL_readAdcData**)

bias **+**= OFFSET_getOffset(obj->offsetHandle_I[cnt]); (**HAL_updateAdcBias**)

- Negative polarity for x = 0, 1, and 2
  - "value" must be **NEGATIVE**
  - "bias" must be **-**=

pAdcData->I.value[x] = **-**value; (**HAL_readAdcData**)

bias **-**= OFFSET_getOffset(obj->offsetHandle_I[cnt]); (**HAL_updateAdcBias**)

2-5. Number of shunt resistors configuration

USER_NUM_CURRENT_SENSORS

- **Chooses the number of shunts available for current measurement.**
  - **Options are either 2 or 3.**

USER_NUM_VOLTAGE_SENSORS

- **Defines the number of voltage sensors**
  - **Must be 3**

2-6. PWM dead-time configuration

- Modify PWM dead-time value according to your own hardware requirement – Find “**HAL_PWM_DBFED_CNT** “and “**HAL_PWM_DBRED_CNT** in” in “**hal.h**”:

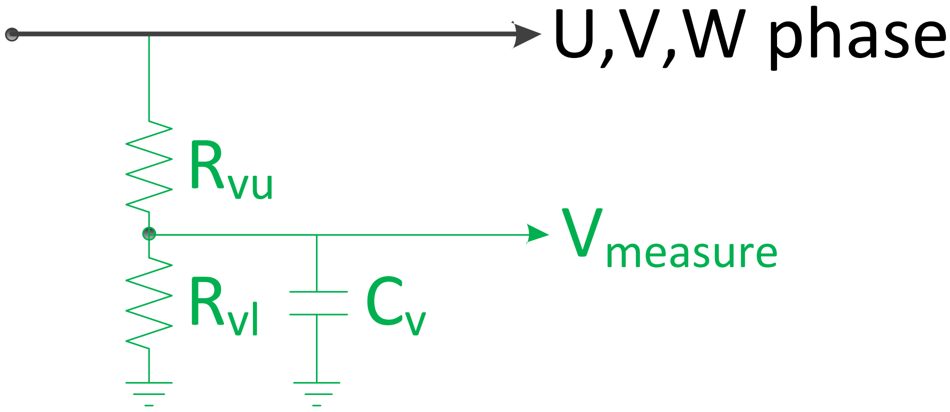
```
//! \brief Defines the PWM deadband falling edge delay count (system clocks)
//!
#define HAL_PWM_DBFED_CNT         (uint16_t)(2.0 * (float_t)USER_SYSTEM_FREQ_MHz)          // 2 usec


//! \brief Defines the PWM deadband rising edge delay count (system clocks)
//!
#define HAL_PWM_DBRED_CNT         (uint16_t)(2.0 * (float_t)USER_SYSTEM_FREQ_MHz)          // 2 usec
```

2-7. Voltage filter pole configuration

USER_VOLTAGE_FILTER_POLE_Hz

- Defines the analog voltage filter pole location in Hz.

$$\text{USER\_VOLTAGE\_FILTER\_POLE\_Hz} = \frac{C_v}{{R_{vl}}^{-1} + {R_{vu}}^{-1}} \cdot \frac{1}{2\pi}$$



2-8: 3-phase ADC current/voltage offsets configuration

- Running Lab3a on your own InstaSPIN board to read out "**gMotorVars.I_bias**" and "**gMotorVars.U_bias**" values and fill them into I_A_offset, I_B_offset, I_C_offset and V_A_offset, V_B_offset, V_C_offset in "**use.h**":

```
#define    I_A_offset    (0.9931434393)
#define    I_B_offset    (0.9894407988)
#define    I_C_offset    (1.00006938)

#define    V_A_offset    (0.3335669041)
#define    V_B_offset    (0.3322209716)
#define    V_C_offset    (0.3348609805)
```

← gMotorVars.I_bias.value[0]
← gMotorVars.I_bias.value[1]
← gMotorVars.I_bias.value[2]

← gMotorVars.V_bias.value[0]
← gMotorVars.V_bias.value[1]
← gMotorVars.V_bias.value[2]

2-9. Software with hardware configuration

USER_IQ_FULL_SCALE_FREQ_HZ

- **The maximum electrical frequency of the currents and voltages that the motor will operate at.**
  - **Set 20%-30% higher than the highest motor electrical frequency for headroom.**

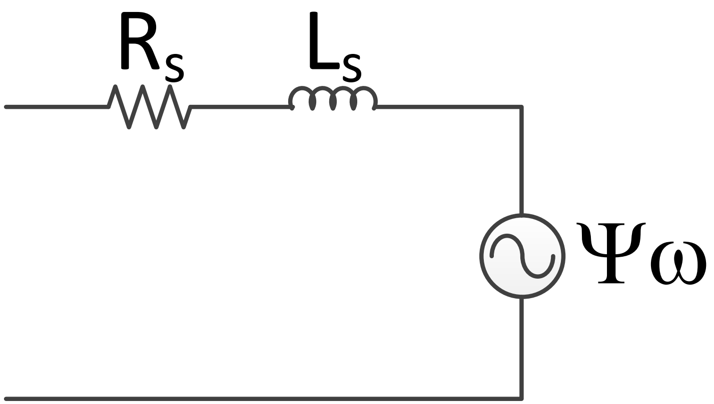
USER_PWM_FREQ_kHz

- **Defines the Pulse Width Modulation (PWM) frequency in kHz.**
  - **The PWM frequency can be very large but care must be taken to allow enough time for the main ISR to be finished.**

3. Motor ID Debugging

3-1.User Motor Parameters PMSM

```
#define USER_MOTOR_TYPE
#define USER_MOTOR_NUM_POLE_PAIRS

#define USER_MOTOR_Rs

#define USER_MOTOR_Ls_d
#define USER_MOTOR_Ls_q
#define USER_MOTOR_RATED_FLUX

#define USER_MOTOR_RES_EST_CURRENT
#define USER_MOTOR_IND_EST_CURRENT
#define USER_MOTOR_MAX_CURRENT

#define USER_MOTOR_FLUX_EST_FREQ_Hz
```



3-2.Motor Identification Parameters

- USER_MOTOR_TYPE
  - MOTOR_Type_Pm → PMSM
  - MOTOR_Type_Induction → AC Induction
- USER_MOTOR_NUM_POLE_PAIRS
  - Number of magnetic poles/2
- USER_MOTOR_RES_EST_CURRENT
  - Current used to measure the resistance (Rs) of the motor
    - Rs measurement requires a park start
  - Current used to start the motor during identification
    - Use higher values if the motor is loaded or has high cogging torque
- USER_MOTOR_IND_EST_CURRENT
  - Negative current setting used to measure the total inductance (Ls) of the motor
  - The magnitude will change the result of the Ls measurement. Determines where on the hysteresis curve Ls is measured.

3-3. Motor Identification Parameters

- USER_MOTOR_MAX_CURRENT
  - Maximum peak motor current
  - Equals maximum total current $\sqrt{I_q + I_d}$
- USER_MOTOR_FLUX_EST_FREQ_Hz
  - Electrical frequency that the motor is spun at during identification
  - Make higher if identification fails

3-4: Updating User.h from Motor Datasheet

INDIVIDUAL SPECIFICATIONS

Model	2310
Electrical Interface Option	P/C/Y
Resistance, phase to phase, [Ω]	0.72
Inductance, phase to phase, [mH]	0.40
Electrical Time Constant, [mS]	0.56
Back EMF (Ke), [Vpeak/kRPM]	4.64
Continuous Torque [oz-in][1,2]	39
Motor Poles	8 (4 Pairs)

$$R_s^{user.h} = R_s^{phase-phase} \cdot \frac{1}{2} = 0.72\Omega \cdot \frac{1}{2}$$

$$R_s^{user.h} = 0.36\Omega$$

$$L_{s\_d}^{user.h} = L_{s\_q}^{user.h} = L_s^{phase-phase} \cdot \frac{1}{2} = 0.4\text{mH} \cdot \frac{1}{2}$$

$$L_{s\_d}^{user.h} = L_{s\_q}^{user.h} = 0.0002\text{H}$$

$$\psi^{user.h} = K_e^{\frac{Vpeak}{kRPM}} \cdot \frac{60\text{sec}}{1\text{min}} \cdot \frac{1\text{kREV}}{1000\text{REV}} \cdot \frac{1\text{REV}}{\text{PolePairs}} \cdot \frac{1\text{V}_{\text{line-neutral}}}{\sqrt{3}\text{V}_{\text{line-line}}}$$

$$\psi^{user.h} = 4.64 \cdot 60 \cdot \frac{1}{1000} \cdot \frac{1}{4} \cdot \frac{1}{\sqrt{3}}$$

$$\psi^{user.h} = 0.0402 \frac{V}{Hz}$$

```
#define USER_MOTOR_NUM_POLE_PAIRS (4)
#define USER_MOTOR_Rs             (0.36)
#define USER_MOTOR_Ls_d           (0.0002)
#define USER_MOTOR_Ls_q           (0.0002)
#define USER_MOTOR_RATED_FLUX     (0.0402)
```

3-4. Motor ID labs testing

- For F2802xF, using lab2b to do motor ID
- For F2806xF/M and F2805xF/M, using lab2a to do motor ID
- If any errors occur during motor ID, stop motor ID, change related parameter settings in “use.h”, compile and download again, then start a new motor ID process.
- After motor ID has been successfully finished, manually input Rs, Rr(for induction motor only), Ld, Lq, Flux_VpHz(for PMSM and BLDC only) and MagnCurr_A (for induction motor only) into below settings in “use.h”:

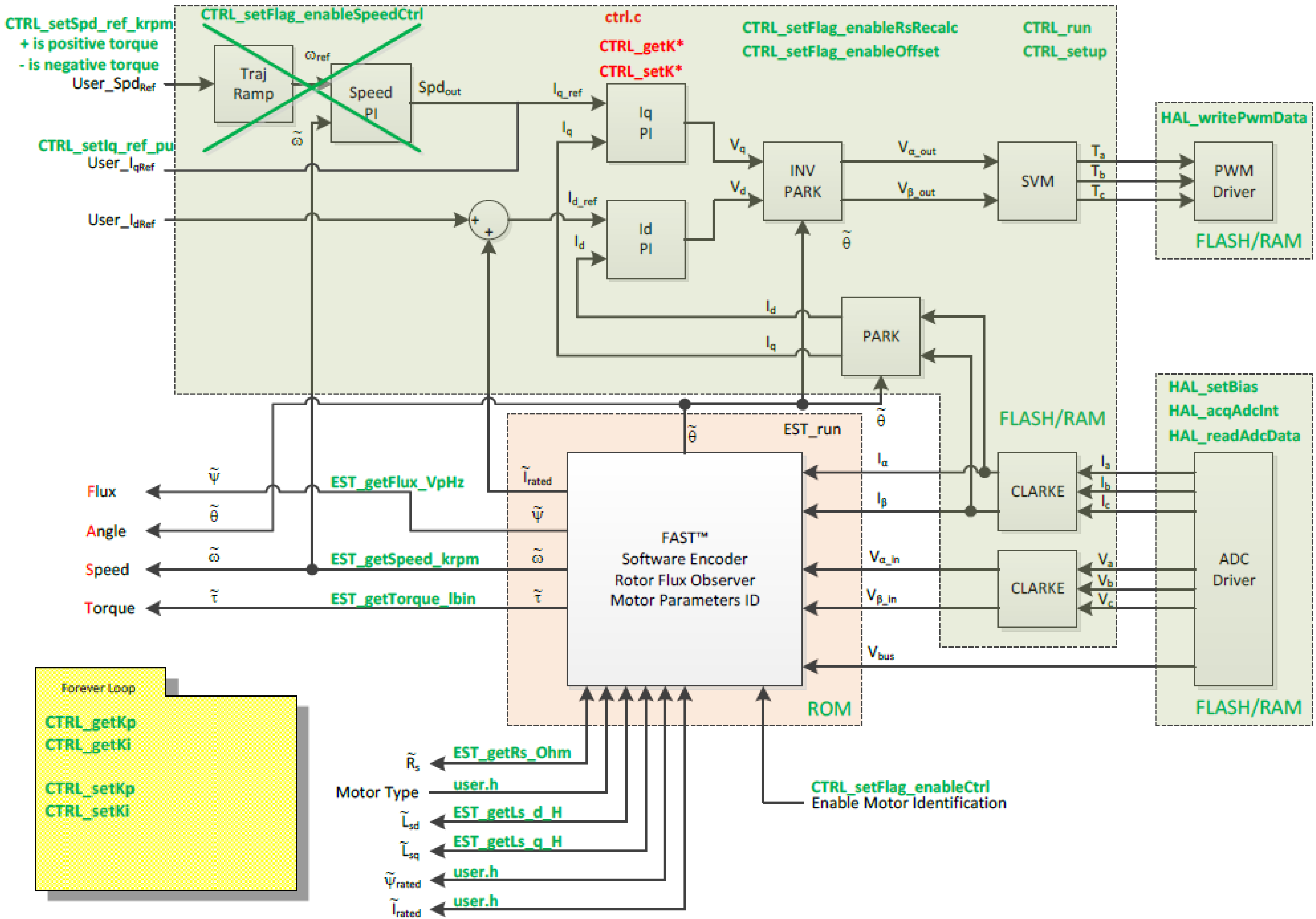
```
#define USER_MOTOR_TYPE                      MOTOR_Type_Pm
#define USER_MOTOR_NUM_POLE_PAIRS            (4)
#define USER_MOTOR_Rr                        (NULL)
#define USER_MOTOR_Rs                        (6.425438)
#define USER_MOTOR_Ls_d                      (0.05326611)
#define USER_MOTOR_Ls_q                      (0.05326611)
#define USER_MOTOR_RATED_FLUX                (0.5507123)
#define USER_MOTOR_MAGNETIZING_CURRENT       (NULL)
#define USER_MOTOR_RES_EST_CURRENT           (0.2)
#define USER_MOTOR_IND_EST_CURRENT           (-0.2)
#define USER_MOTOR_MAX_CURRENT               (4.0)
#define USER_MOTOR_FLUX_EST_FREQ_Hz          (30.0)
```

4. Motor Running Debugging

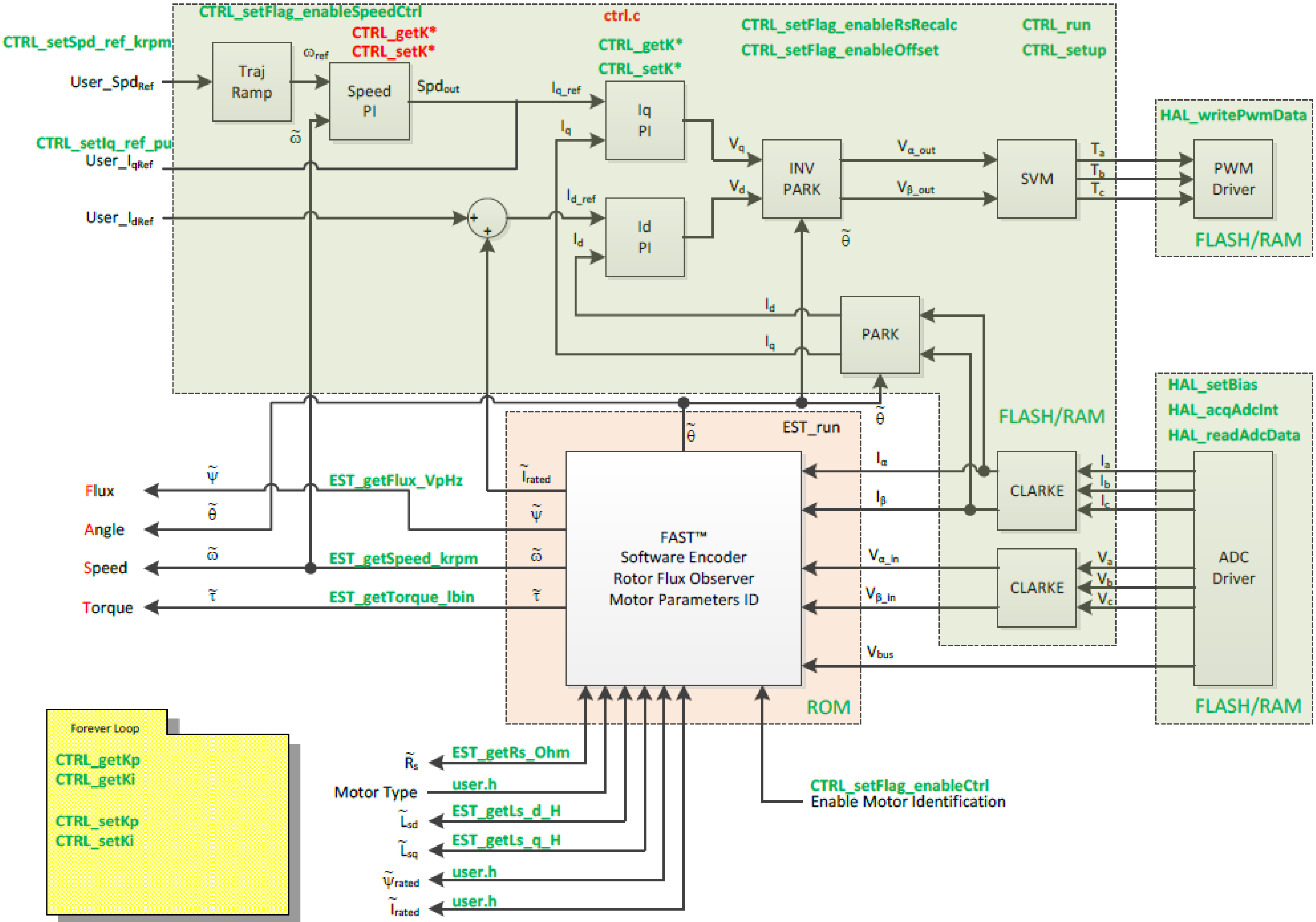
4-1. Motor Running labs testing

Motorware Labs	Functions	Control Modes	Debugging Notes
lab05a	Torque mode and tuning Id/Iq PI	Speed open-loop, manually set Iq_Ref value.	Id/Iq PI parameters fine-tuning.
lab05b	Speed mode and tuning speed PI	Speed closed-loop, manually set Speed_Ref value.	Speed PI and Id/Iq PI parameters fine-tuning.
Lab07(7a)	Rs Online (7a is for FPU)	Rs online calibration for temperature increasing condition	Using Rs_Online instead of Rs for estimator.
Lab09(9a)	Field Weakening (9a is for FPU)	Automatic field-weakening: calculated Id_Ref value; Manual field-weakening: Input Id_Ref value.	High speed testing requirement.

4-2: Block diagram of Torque Mode – lab 05a

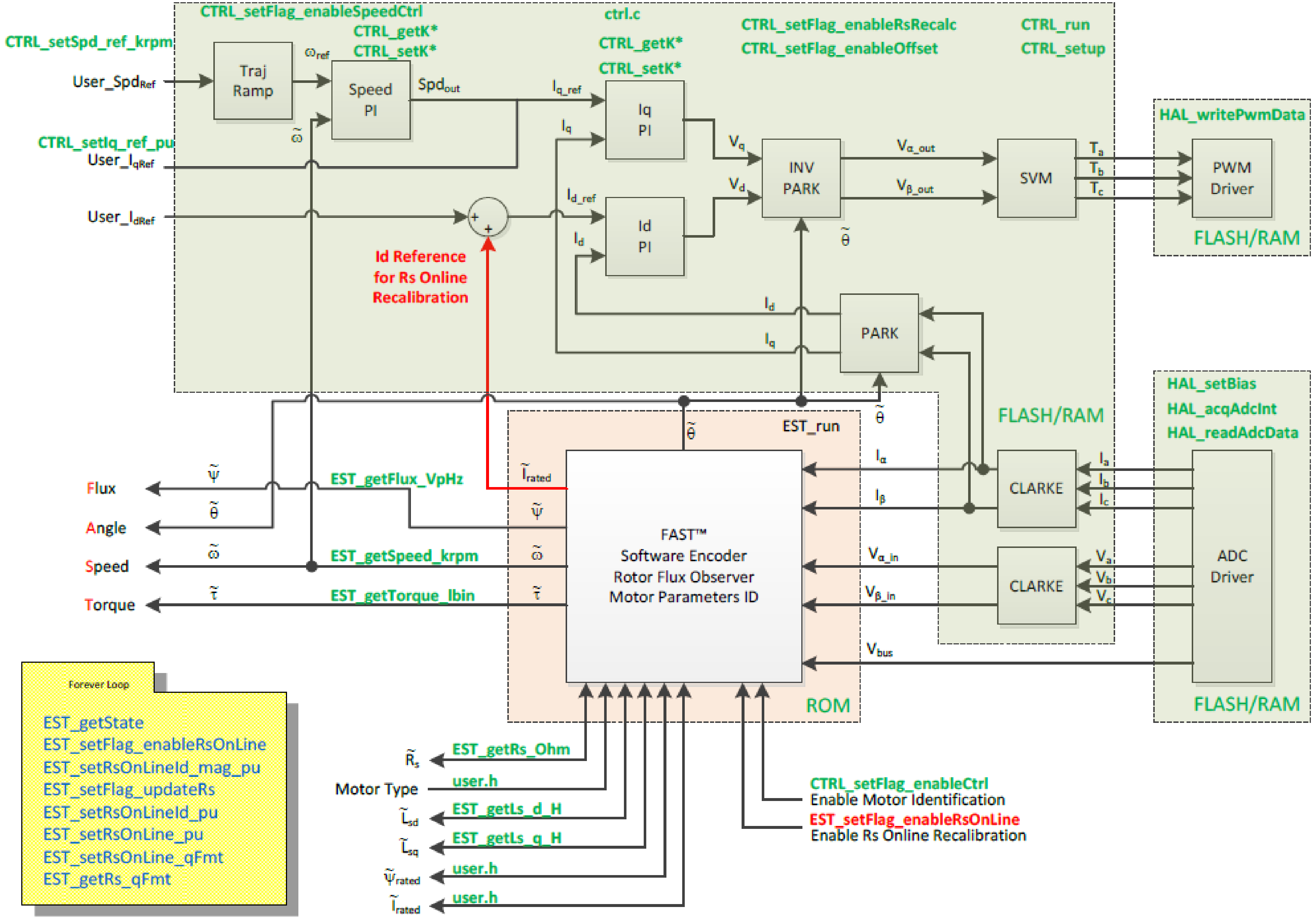


4-3: Block diagram of Speed Mode – lab 05b



Texas Instruments

4-4: Block diagram of Using Rs Online Calibration – lab 07(07a)



TEXAS INSTRUMENTS

Thanks!