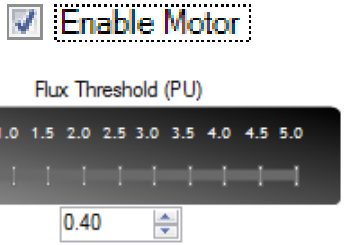
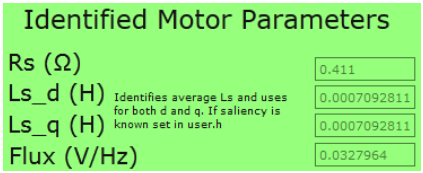
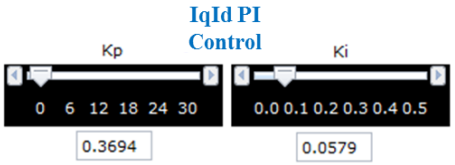
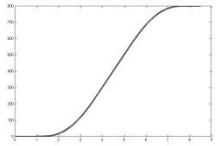
InstaSPIN™

Instantly Enabling Superior 3-phase Motor & Motion Solutions









Entry BLDC & sensorless
Simplified commutation
Bemf software sensor

Motor Parameter ID
Automatic FOC torque tuning
Robust software encoder

Simplified speed tuning
Premium performance
Motion & Planning

InstaSPIN™-BLDC

InstaSPIN™-FOC
with FAST™

InstaSPIN™-MOTION
with FAST™ and SpinTAC™

+ [sensorless]
+ 6-step [commutate]

+ better [sensorless]
+ sinewave [commutate]
+ ideal [torque]

+ ideal [speed]
+ ideal [position]
+ on-chip [motion]
+ integrated [plan]

Expertise Provided

InstaSPIN™ Resources

http://www.ti.com/ww/en/mcu/instaspin/index.shtml

Overview | InstaSPIN-MOTION | InstaSPIN-FOC | InstaSPIN-BLDC | Zero & Slow Speed | Tools & Software | Support & Community | InstaSPIN in Action

InstaSPIN™-FOC, –MOTION and BLDC

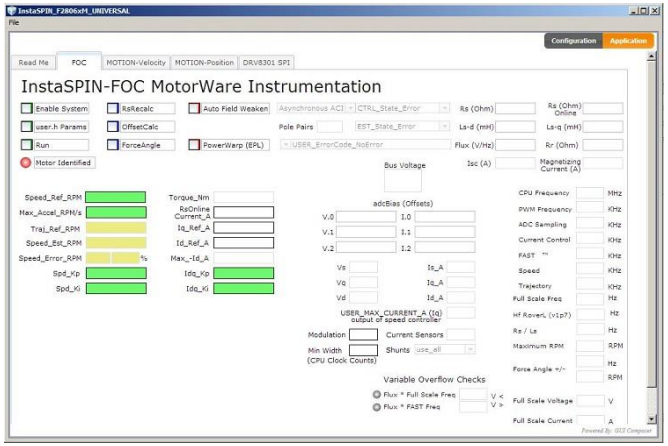
- Thorough Reference Manuals and User's Guide
- MotorWare projects, detailed lab documentation, and code examples provided
- Includes API information
- GUI and CCS

InstaSPIN Motor Control Kits

- Low Voltage kit
- Medium Current kit
- High Current kit
- High Voltage kit
- TMDSCNCD28027F controlCARD
- TMDSCNCD28069MISO controlCARD
- TMDSCNCD28054MISO controlCARD
- F28027F LaunchPad
- F28069M LaunchPad

InstaSPIN Resources

- TRM for F2806xF FOC
- TRM for F2806xM MOTION
- TRM for F2802xF FOC
- TRM for F2805xF FOC
- TRM for F2805xM MOTION
- Download User's Guide
- Download MotorWare



TI E2E™ Community

Training

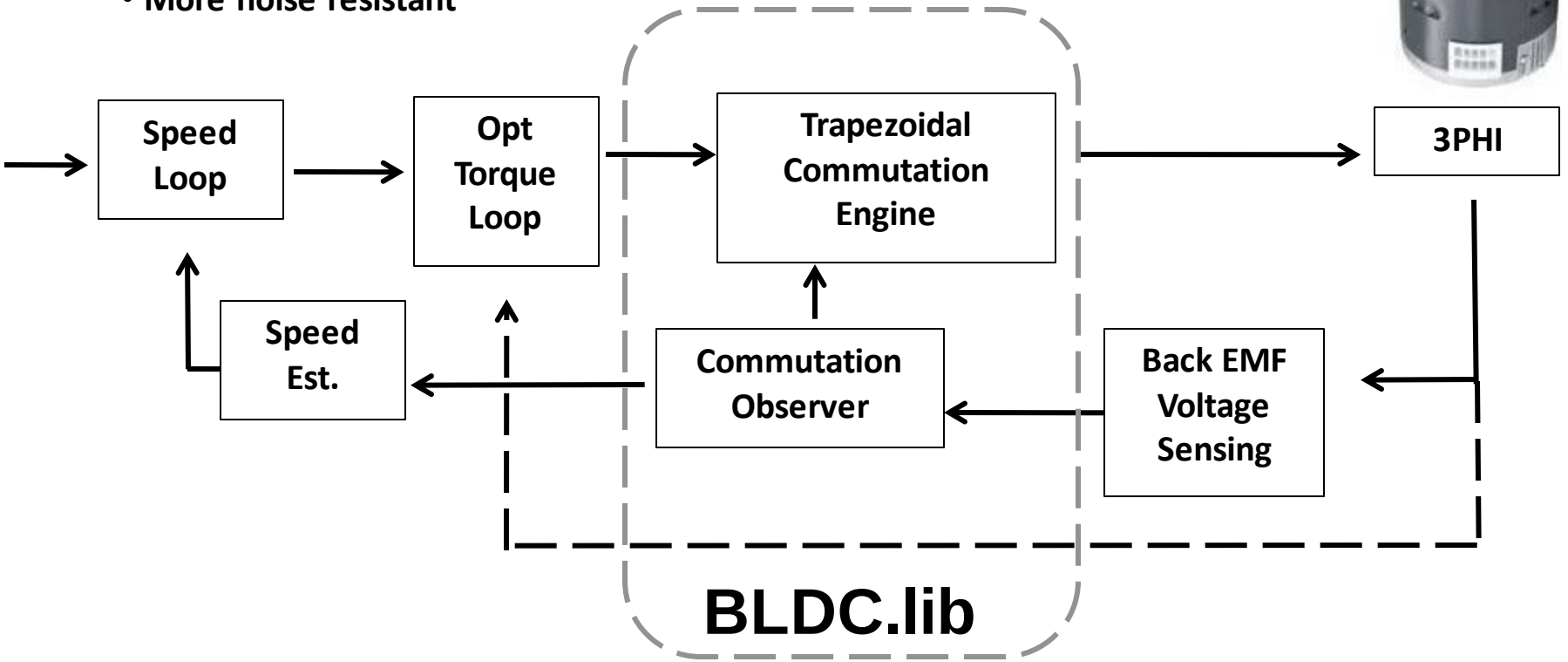
- •MotorWare projects offer self paced workshop style sessions
- •In person events coming soon

InstaSPIN –BLDC Details

www.ti.com/instaspin-bldc

InstaSPIN™-BLDC

- Low cost, Sensorless Control of BLDC/PMSM motors
- Bemf Voltage technique without issues of Bemf Zero Cross Detection!
- Robust start-up
- Robust during acceleration / deceleration
- Forward and Reverse control
- Capable of much lower and higher speeds
- More noise resistant



The InstaSPIN-BLDC Advantage

Robust!
InstaSPIN has currently been tested on about 80 different motors. In each case, the motor was running in under 30 seconds, and suitably tuned in under two minutes.

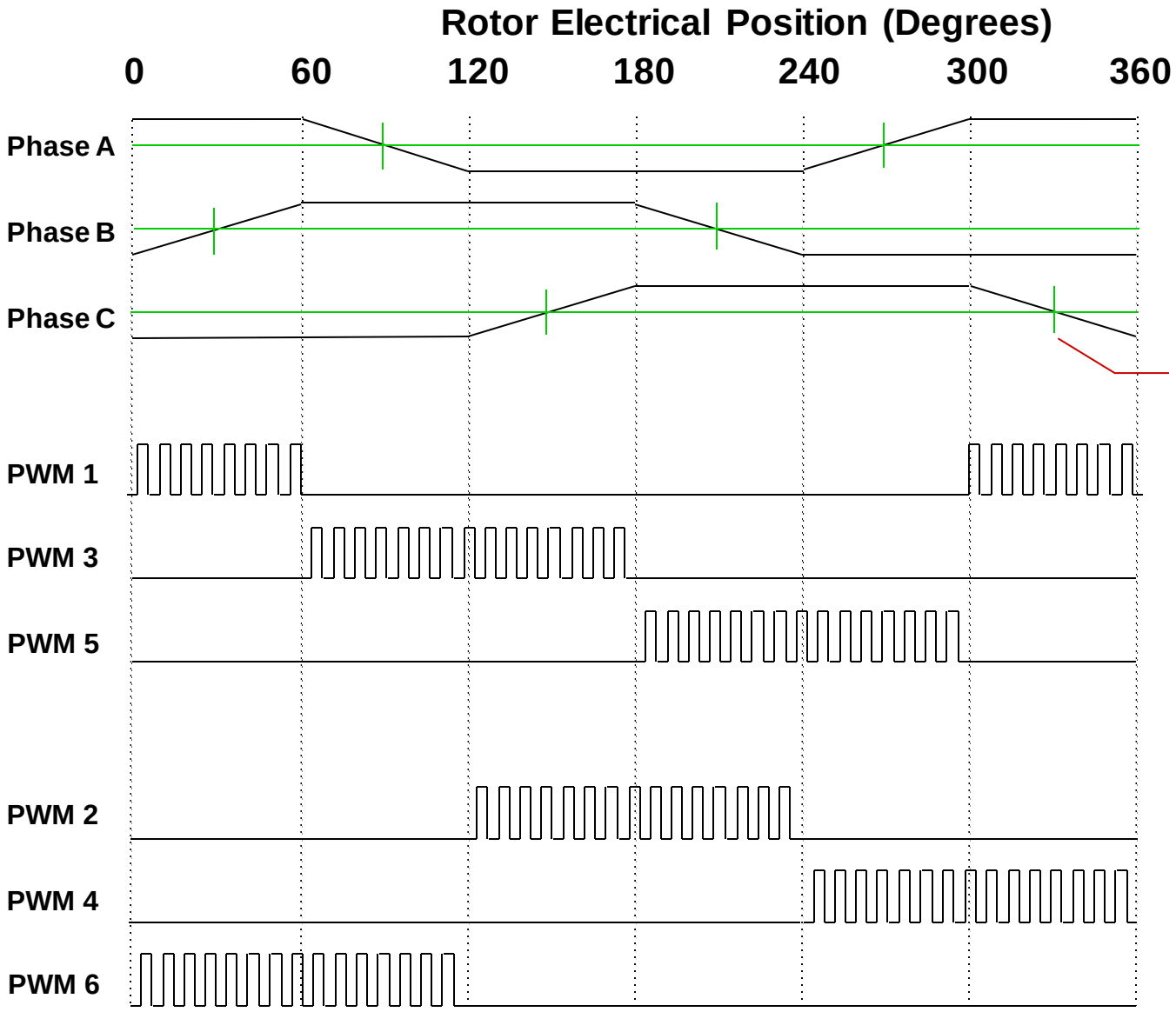
Simple!
Only one parameter needs to be adjusted to tune the commutation process.

Better High Speed!
Commutation by dynamically lowering the threshold level.

More Reliable Low Speed!
Back-EMF integration creates a less noisy signal for commutation control.

Adapts to Acceleration Changes!
unlike the zero-cross timing technique

Most Common Sensorless Technique: Zero Cross



Requires Timing

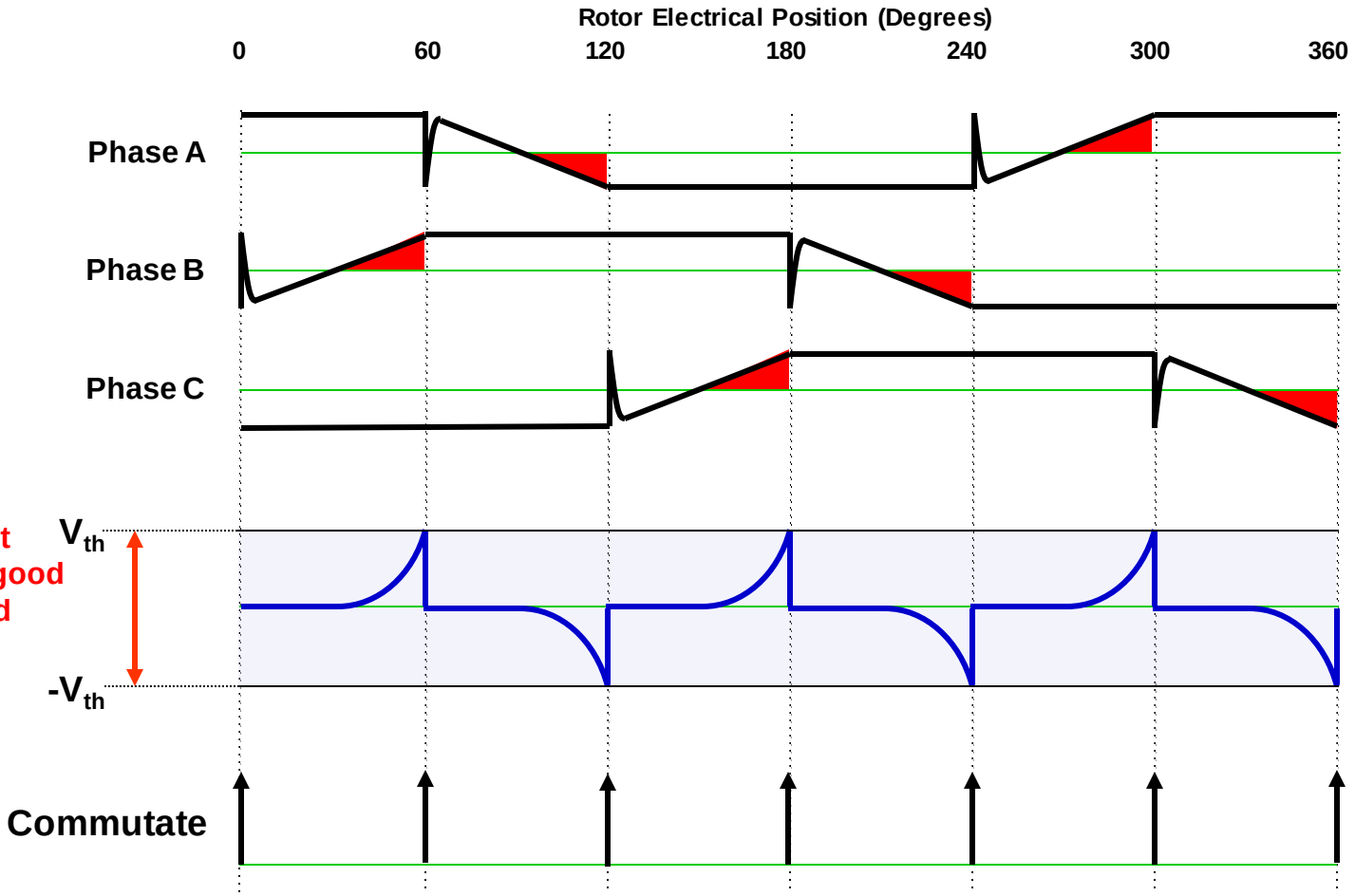
Uses past info as future, hence poor during speed and load changes

InstaSPIN - BLDC…A Peek Under the Hood

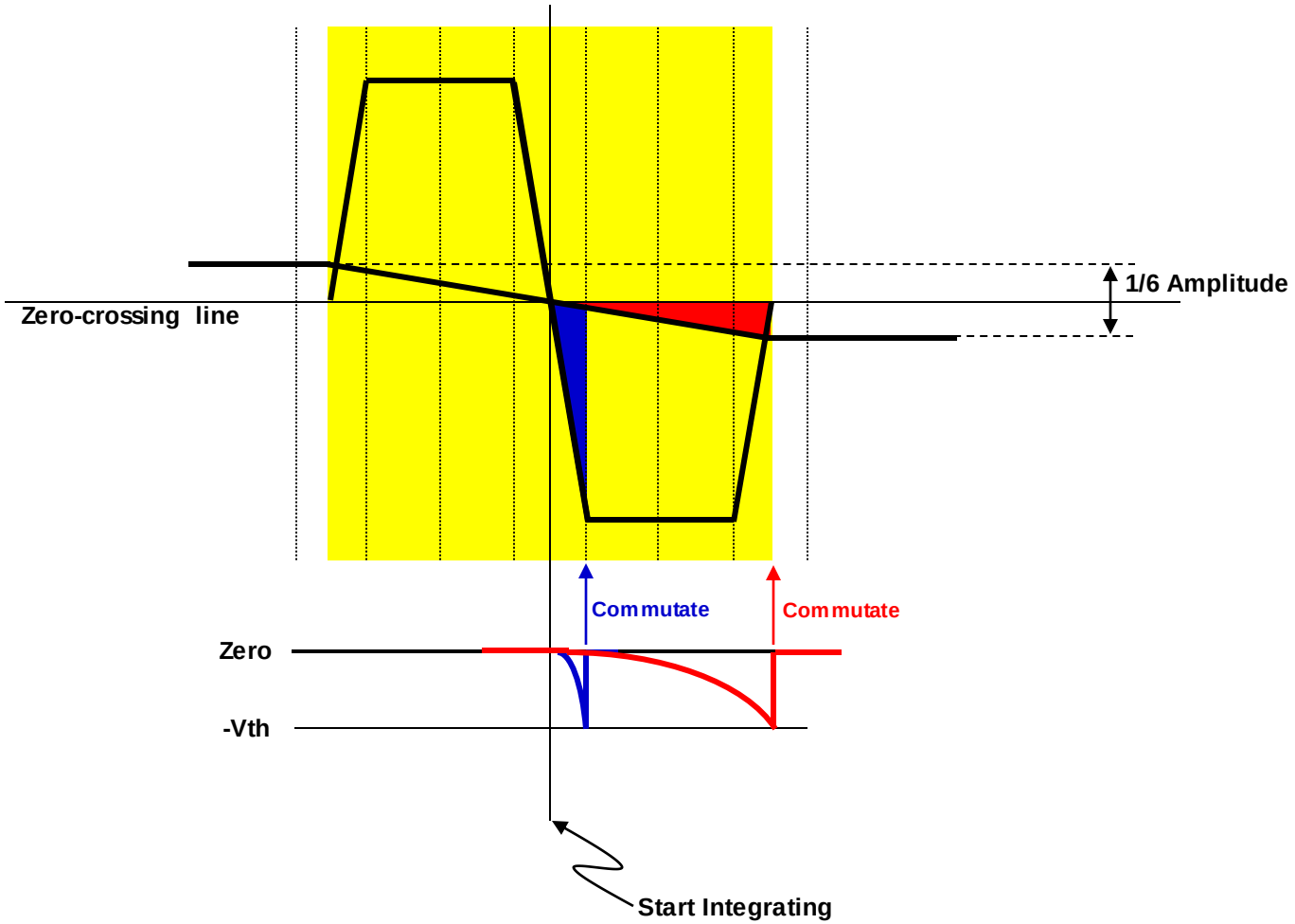
No Timing

Adjustable threshold voltage

Uses present info, hence good during speed and load changes

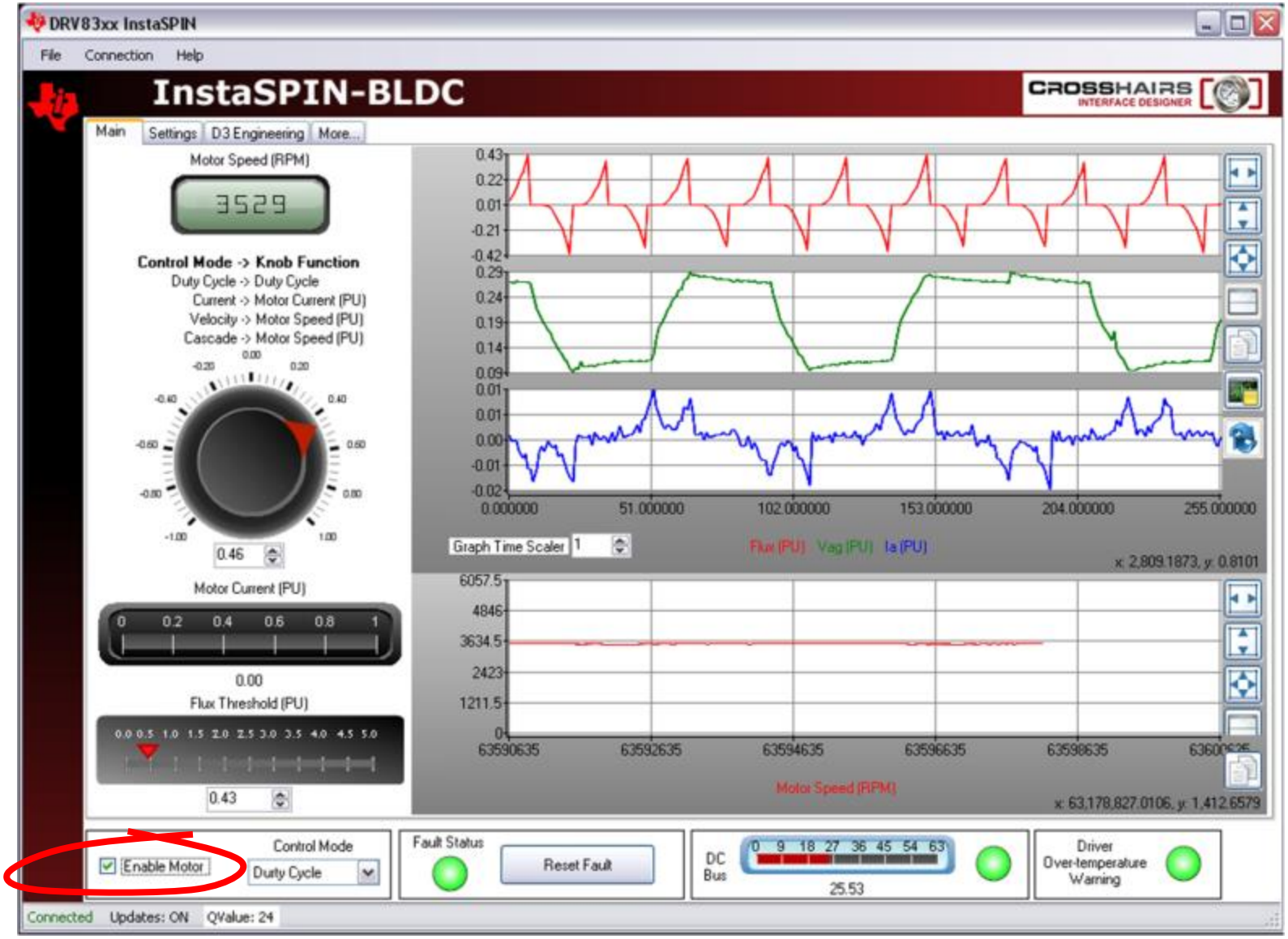


Speed Invariant Performance

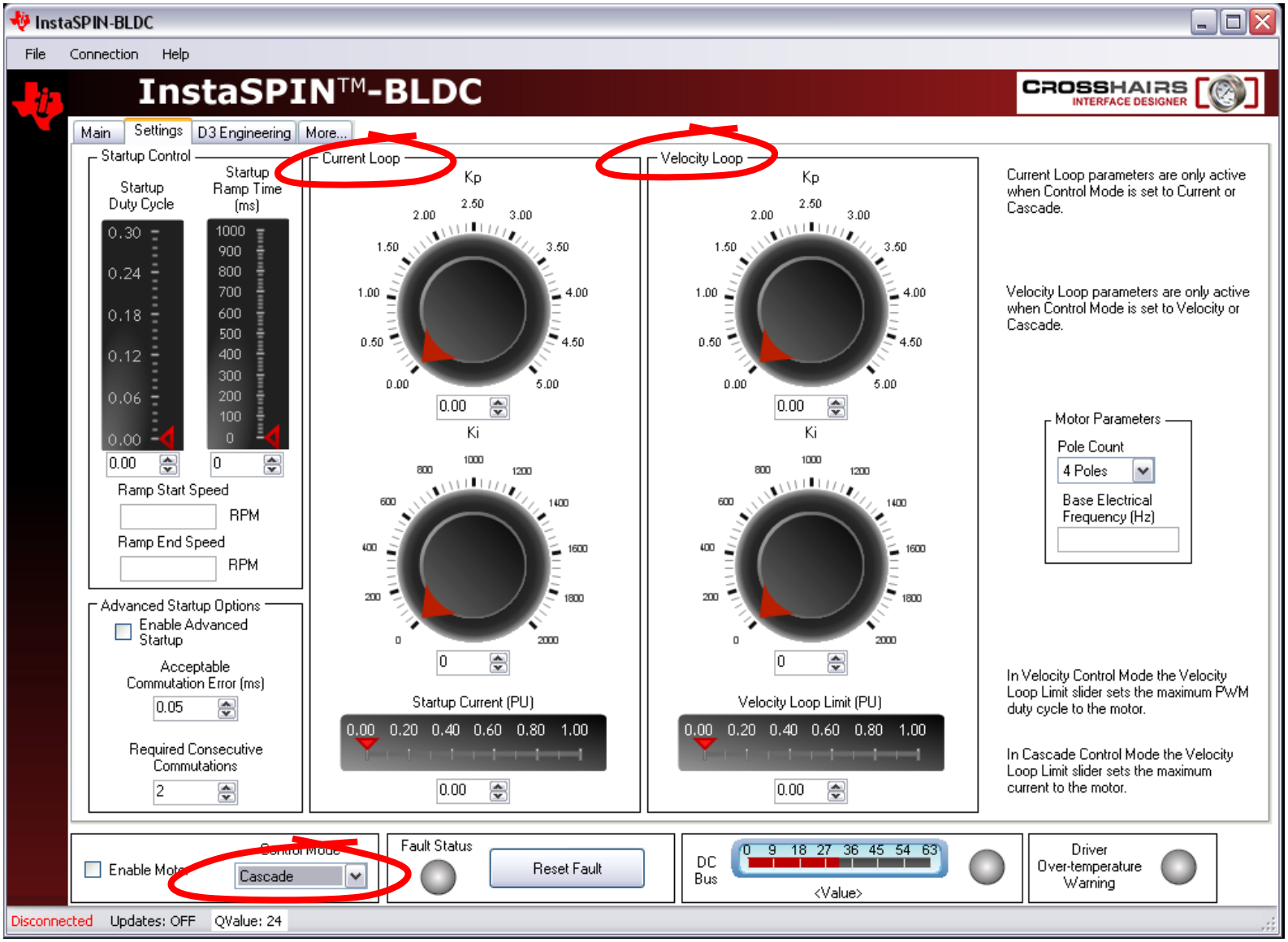


$$voltage = \frac{d\lambda}{dt}$$

Spin & Tune with GUI



Closed Loop with GUI



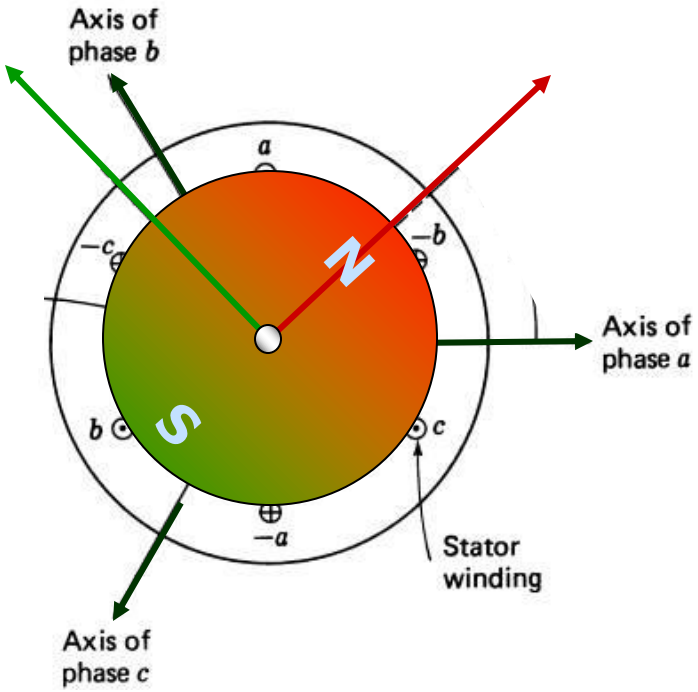
InstaSPIN –FOC Details

www.ti.com/instaspin-foc

FOC & Sensorless Challenges

FOC requires precise position knowledge of rotor magnetic field to create appropriate stator magnetic field, oriented to produce maximum torque.

1. **Costly sensor (encoder/resolver)**
   - Mechanical alignment
   - Not necessarily magnetic unless "absolute"
2. **Complex software algorithms (model observers)**
   + Lower cost and no repair or replacement
   + Can be used where sensors can't be
   - Not appropriate for FOC with position control



TI's new FAST software sensor is Superior

+ Works with synchronous & asynchronous motors
+ Model relies on fewer motor parameters
  + Optional start-up parameter ID tool
  + Optional run-time parameter tracking
+ Observer requires no tuning
+ More accurate, more dynamically robust
+ Stable at and through zero speed
+ Stall recovery
+ Better at start-up under load
+ Parameters used to set FOC current controllers
+ Highest fidelity feedback signals
- / + Proprietary technique, no source given

InstaSPIN™-FOC with FAST

simplify design of a sensorless torque controller

"simple" Motion | *"starting" Speed Control* | *Torque Control* | *Inverter Switching*

User Code or ROM

Speed Control

Iq Control

Id Control

3-phase Space Vector Generation

Field Weaken
Field Boost

Speed | Flux | Torque | Angle

Off-Line Motor ID

FAST™ SW Encoder

On-line Motor ID & Compensation

ACI PowerWarp™

Piccolo ROM

PWM

ADC

3PHI

F lux changes monitored in real time
A ngle accuracy best available, over widest range!
S peed signal independent of Angle - no phase lag!
T orque signal accurate to multiple decimals!

Feedback

Full load start-up features, 100% duty cycle, stable at and through 0 speed, four quadrant

InstaSPIN™-FOC: FAST™ replaces rotor sensor or software observer techniques of the past

Dramatically reduce challenges of sensorless FOC system development

- Motor parameters identified
- No tuning of FAST required (vs. other algos)
- Current loop automatically tuned
- Speed loop set for evaluation
- "Instant" stable system to start development
- Run-time parameter compensation
- Modes & features for common system challenges: start-up, at & through zero speed, field weakening, high modulation, PowerWarp™ for induction motors

Easy to use flexible software architecture

- Novice can call full system from ROM adjusting control gains
- Expert can fully customize control system calling only FAST from ROM

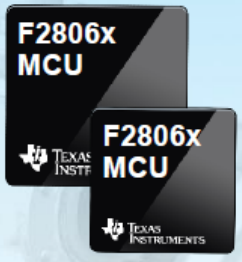
Benefit from high fidelity, low latency feedback signals

- **F**lux signal for field weakening / boosting
- **A**ngle accuracy over widest range
- **S**peed of rotor with near zero phase lag
- **T**orque signal is high bandwidth and high accuracy, enabling monitoring and control of loads and flows

TI's InstaSPIN™-enabled real-time controllers

TI's InstaSPIN three-phase motor control solutions are enabled by special libraries in the read-only memory (ROM) of Piccolo microcontrollers (MCUs) that allow you to create products with improved efficiency, performance, and reliability, while reducing development time from months to minutes. TI's InstaSPIN-enabled MCUs provide expertise to designers of sensorless (velocity and torque) or sensored (position, velocity and torque) motor control applications.

16x16 mm



14x14 mm

14x14 mm

9x9 mm

Click a part number to learn more

	InstaSPIN Solution	MHz	FPU	CLA Co-Processor	Motors	Flash (KB)	12b ADC Chs	PGA	CAN	QEP	USB	SPI	UART	I2C	Pins	Temp
F28069M	-MOTION	90	Y	Y	1 or 2	256	16 or 12	--	1	1	1	2	2	1	100/80	-40 to 105° C
F28068M	-MOTION			--		256										
F28069F	-FOC			Y		256										
F28068F	-FOC			--		256										
F28062F	-FOC			--		128										
F28054M	-MOTION	60	--	--	1 or 2	128	16	4	1	1	--	1	3	1	80	-40 to 125° C Q100
F28054F	-FOC					128										
F28052M	-MOTION					64										
F28052F	-FOC					64										
F28027F	-FOC	60	--	--	1	64	13	--	--	--	--	1	1	1	48	
F28026F	-FOC					32										

Texas Instruments

FAST Algorithm Performance

- Field orientation converges within first electrical cycle
- Works on all three phase motors
- Zero speed start-up operation at full torque
- Fast torque response, negligible torque ripple
- Stable across all corners in all quadrants
- Angle accuracy within +/-1 count of a 1024 encoder steady state
- Stall recovery without losing field orientation
- Includes automatic current loop tuning and Motor Commissioning
- ACI PowerWarp™: 80% energy savings vs. Triac, 45% vs. FOC
  - Real-world data from 200W 1400RPM Industrial Fan in-field installation

Identify, tune, and run best sensorless FOC available in minutes

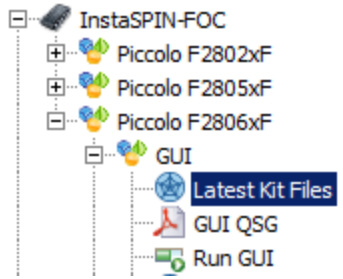
InstaSPIN GUI

- Two types of GUIs
  - InstaSPIN-FOC & InstaSPIN-MOTION GUI

    DEMONSTRATION

    only work with the TMDSCNCD28069MISO controlCARD

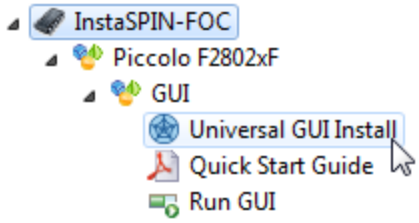
    binary project is “frozen”



  - MotorWare Universal GUI

    InstaSPIN enabled Piccolo device (2xF, 5xF, 5xM, 6xF, 6xM)

    Instruments any MotorWare proj_lab##

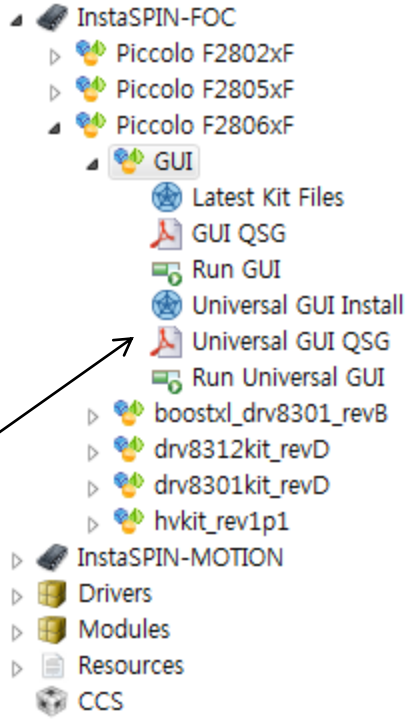


InstaSPIN UNIVERSAL GUI

- Latest files always at

  www.ti.com/tool/instaspinuniversalgui
- Interact with instaSPIN projects over a JTAG connection
- Create a binary from any MotorWare project using CCStudio
- Launch GUI Composer UNIVERSAL GUI using either method:
  - Standalone
  - Inside of CCStudio
- InstaSPIN UNIVERSAL GUI User Guide

Quick Start Guide



InstaSPIN™ –MOTION Details

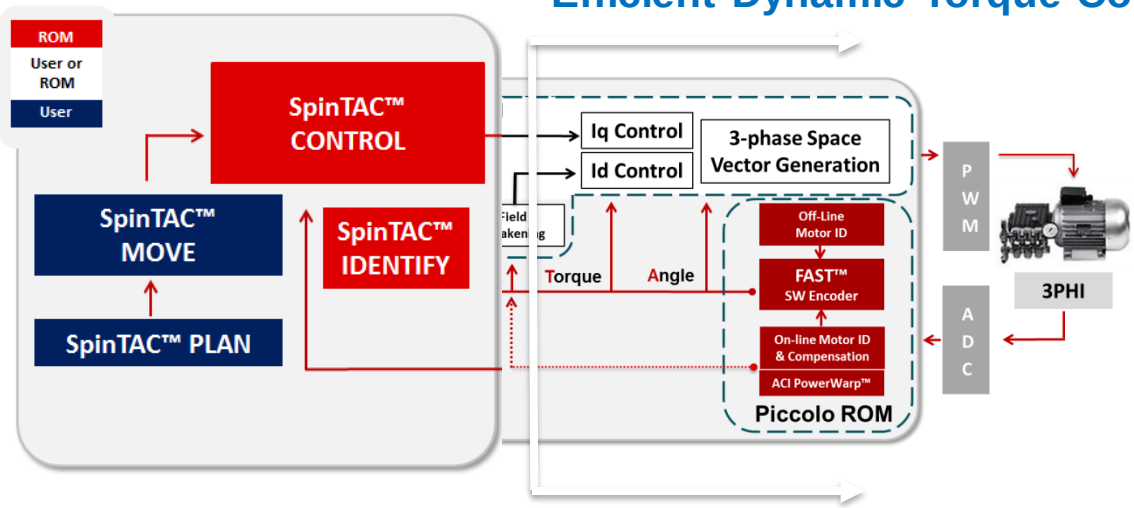
www.ti.com/instaspin-motion

InstaSPIN-FOC to InstaSPIN-MOTION for velocity control

InstaSPIN-FOC provides Efficient Dynamic Torque Control

Speed PI tuning
Complex
Inconsistent across use

Building Motions
Even more complex
Only as good as control



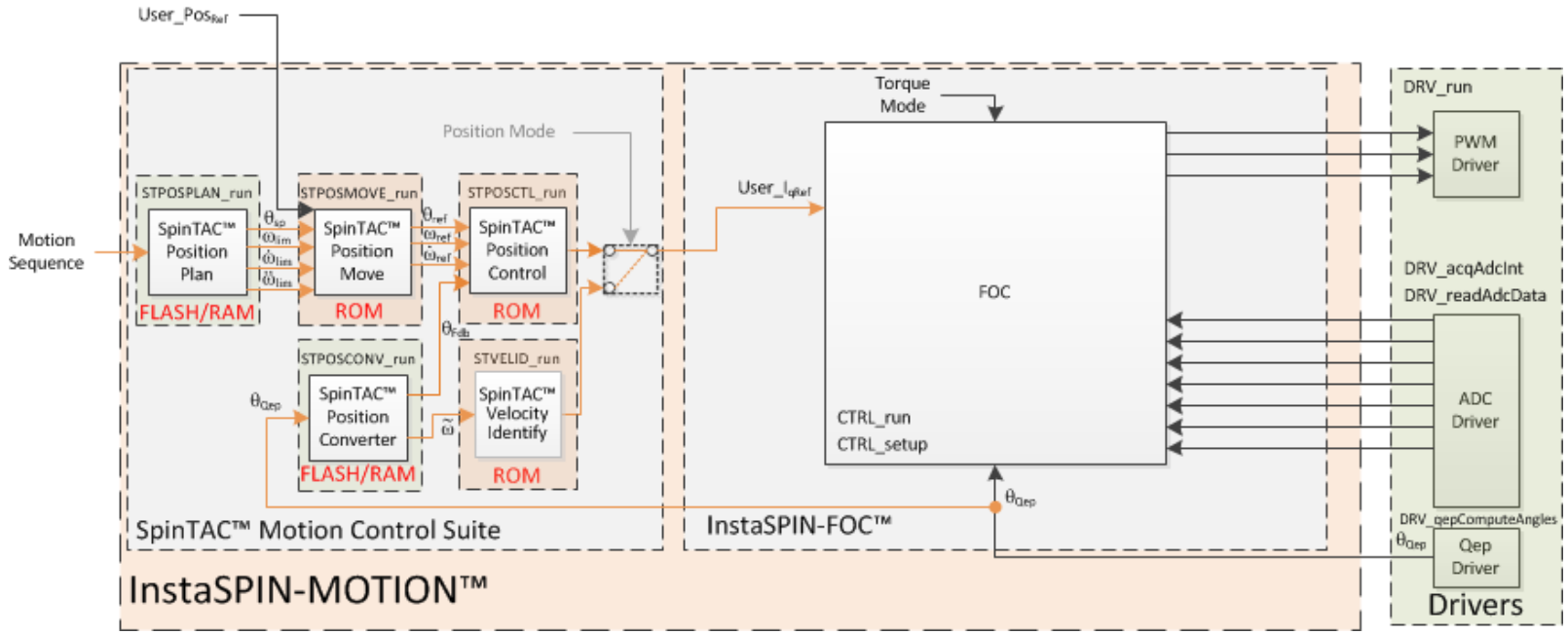
InstaSPIN-MOTION SpinTAC™ suite

- Builds upon InstaSPIN-FOC (or use with sensors)
- IDENTIFY: system inertia identification for enhanced feedback into controller
- CONTROL: single variable controller replaces PI and typically works across system conditions
- MOVE: generation of Speed A to Speed B with various trajectories (trap, S-curve, ST-curve)
- PLAN: logic-based execution of different MOVEs

InstaSPIN-FOC Speed Control

- Initial PI gains are just a first starting point
- Does not incorporate real inertia of system
- Control requires
  - Tuning of 2-variable PI controller
  - "gain staging", different sets of tuning at various operating points
- Movements / Trajectories
  - Only offers constant fixed acceleration

InstaSPIN-MOTION for Position Control



InstaSPIN-MOTION Position Solution

- Requires the use of a sensor
- CONVERTER: puts encoder feedback into mechanical units
- IDENTIFY: system inertia identification for enhanced feedback into controller
- CONTROL: single variable controller replaces PI position & velocity loops and typically works across system conditions
- MOVE: generation of Position A to Position B with various trajectories (trap, S-curve, ST-curve)
- PLAN: logic-based execution of different MOVEs

Key Benefits

- Single Parameter Tuning of BOTH Position & Velocity loops
- Disturbance rejecting controller
- Quiet zero position hold
- Great performance at ultra low speeds
- Jerk controlled ideal position changes

SpinTAC™ Components

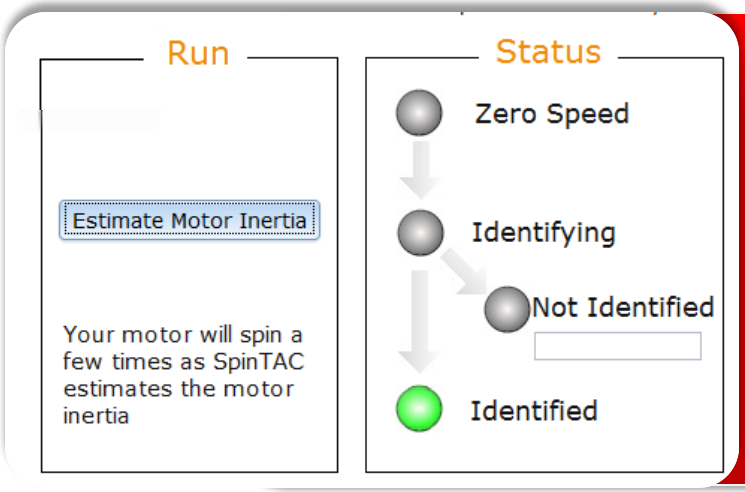
Account for mechanical inertia - Robust speed control - Simplified tuning

Identify:
Measure Inertia

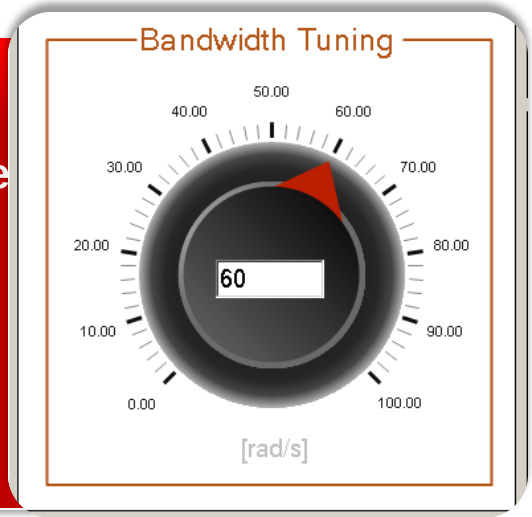
- Inertia is important for accurate control
- Short acceleration test to identify system inertia

Control:
Maximum control, minimum effort

- Disturbance-rejecting controller
- Single variable to tune response
- Typically effective across full variable speed and load range



1. Press button to measure inertia
2. Adjust knob to tune

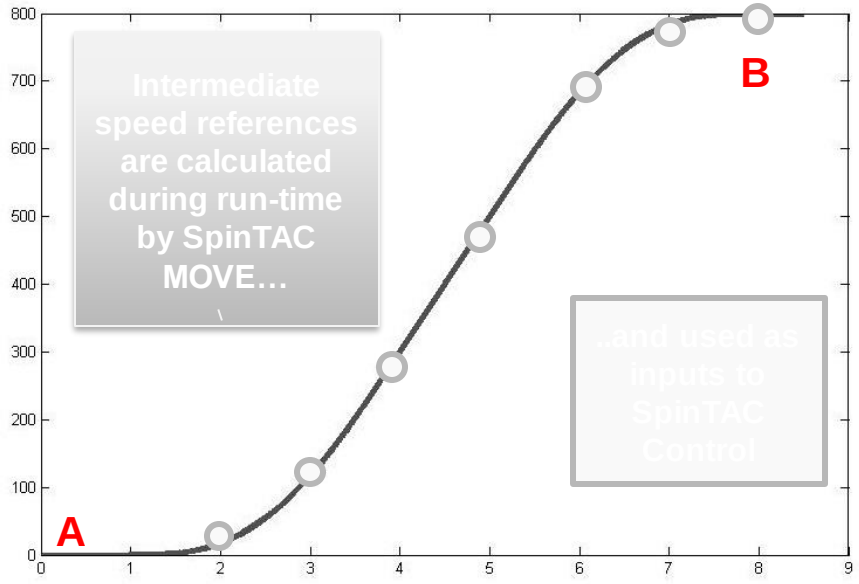


SpinTAC™ Components

Integrated Movement and Motion Design

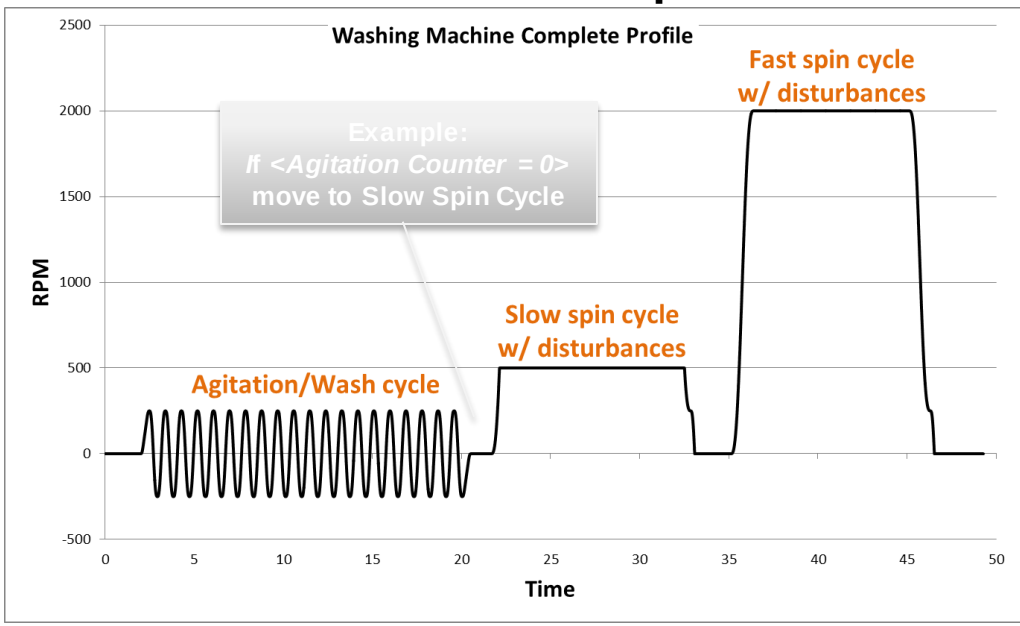
Move:
Build Trajectories

- Select Motion Type for Speed A to B
- Define constraints (accel, jerk)
- Move generates the ideal curve



Plan:
Design Motion Sequence

- Define operating states and transitions
- Connect logic-based Moves
- Execute the motion sequence



The Solution: InstaSPIN™-MOTION

Simplifies design, Improves performance

Robust speed control with compensation for the real mechanics of the system

- **IDENTIFY**: System inertia identification used as input for the most accurate control
- **CONTROL**: SpinTAC™ replaces standard PI and provides more accurate performance across the full operating range through real-time disturbance estimation and cancellation

Simple Tuning

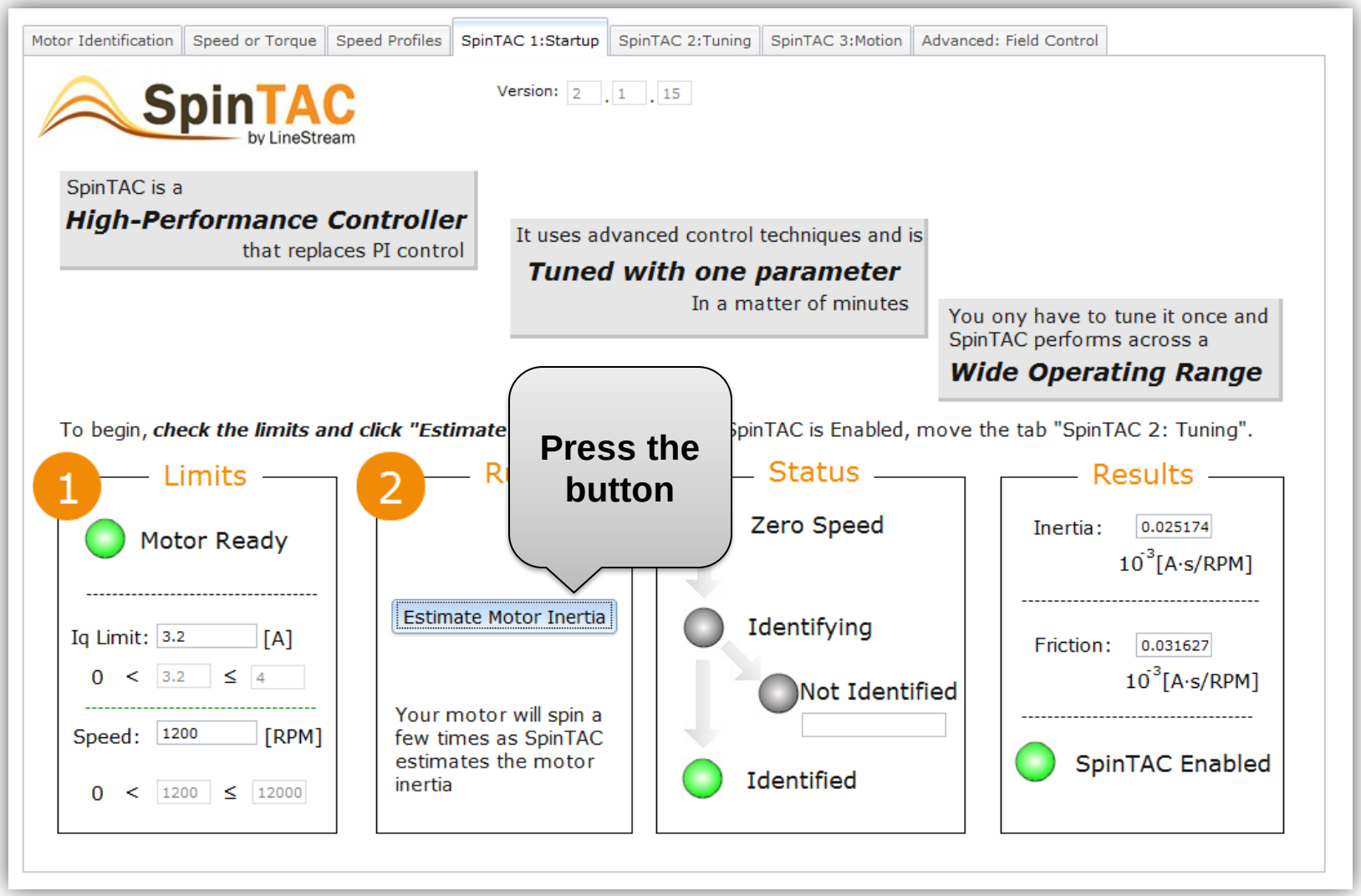
- 1-variable "gain" allows for simple, instant tuning
- Eliminates gain staging – single tuning is effective over the ENTIRE operating range

Easy motion design

- **MOVE:** Motion engine calculates the ideal reference signal (with feed forward) based on user-defined parameters. Supports standard industry curves, and LineStream's proprietary "smooth trajectory" curve
- **PLAN**: Motion Sequence Planner operates user-defined state transition maps

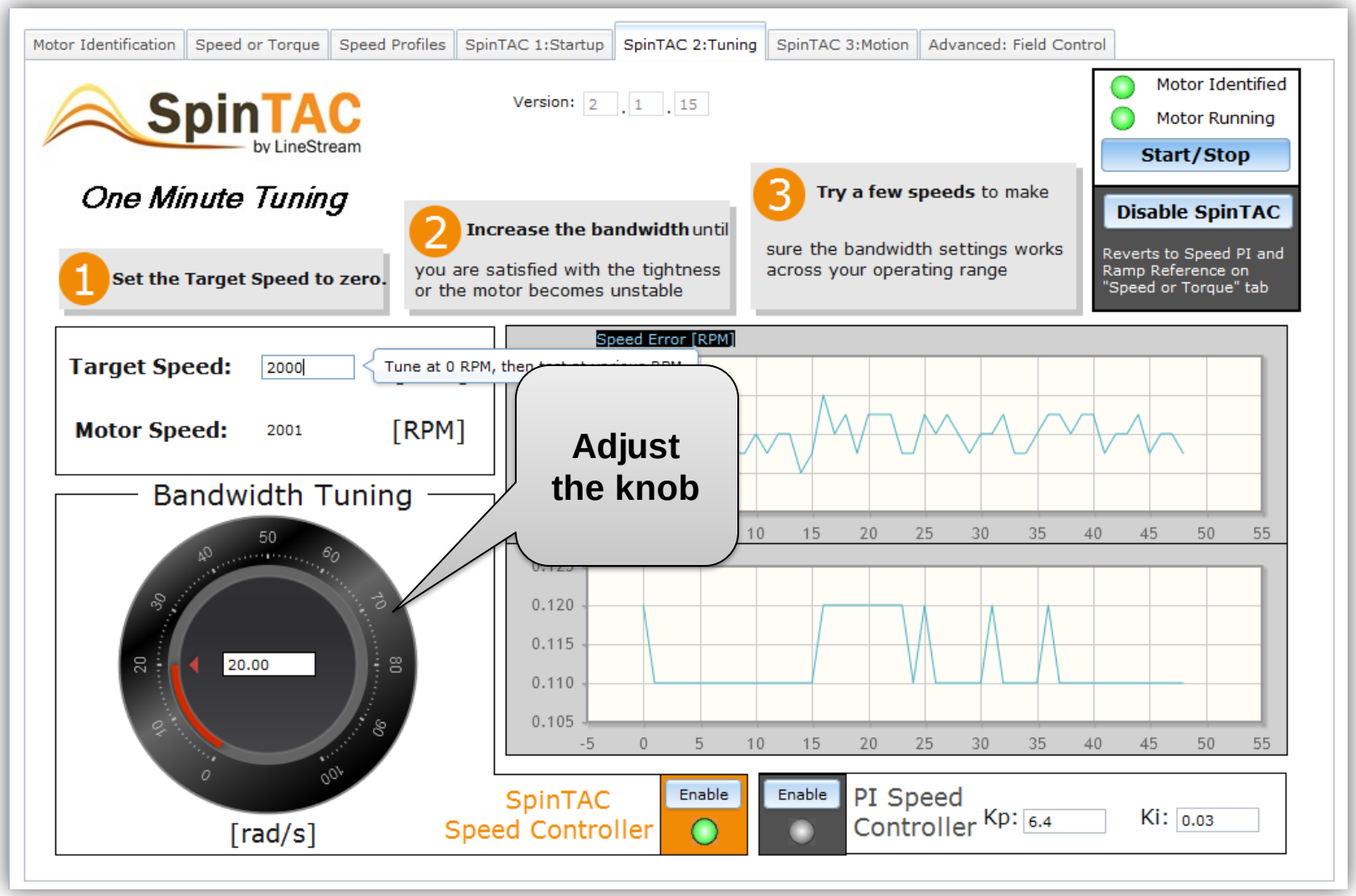
InstaSPIN-MOTION GUI

Step 1 – Estimate Inertia



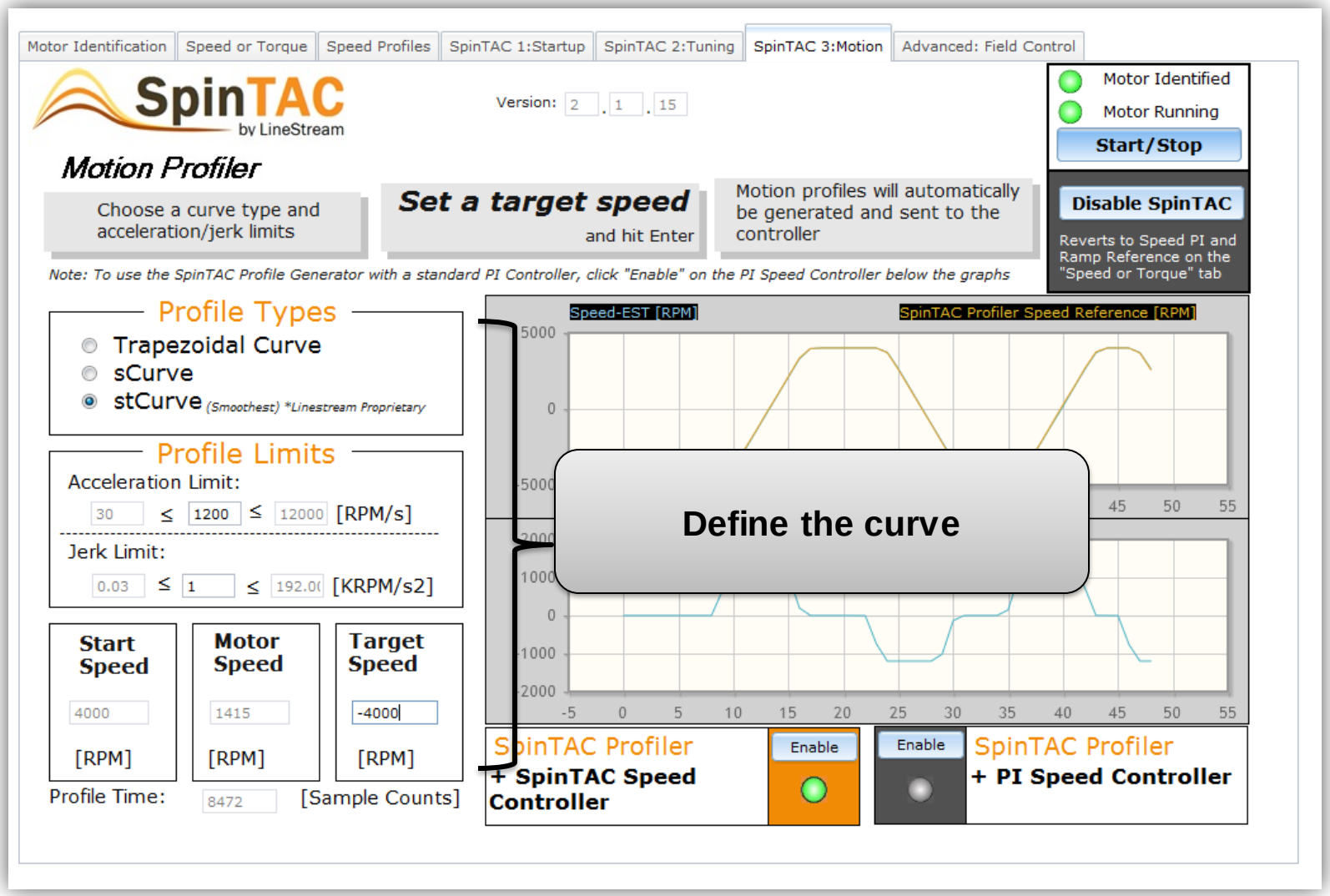
InstaSPIN-MOTION GUI

Step 2 – Tune the Controller

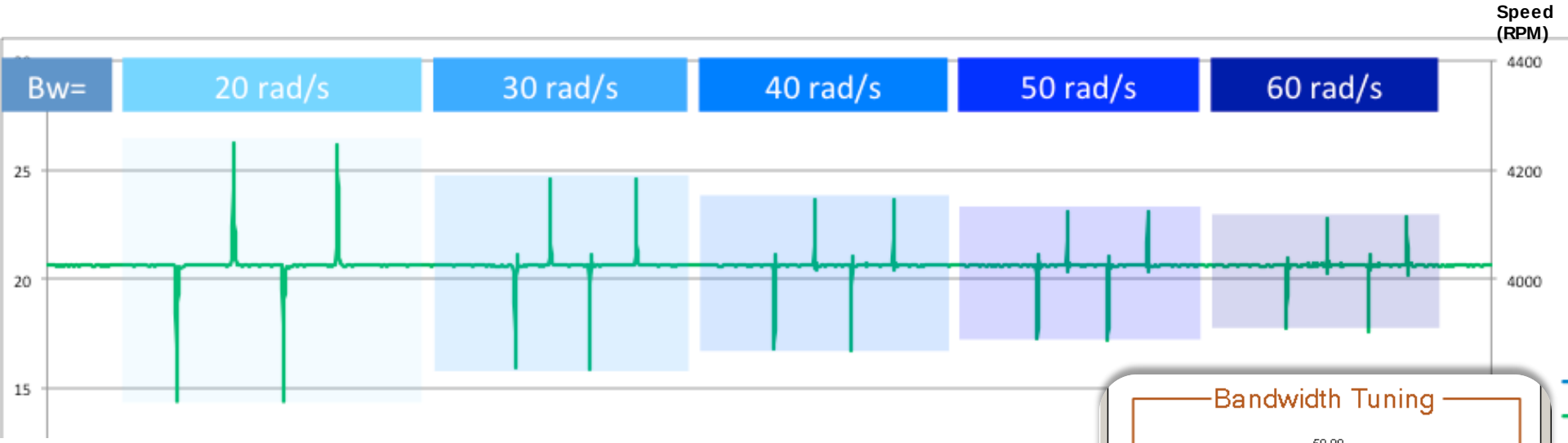


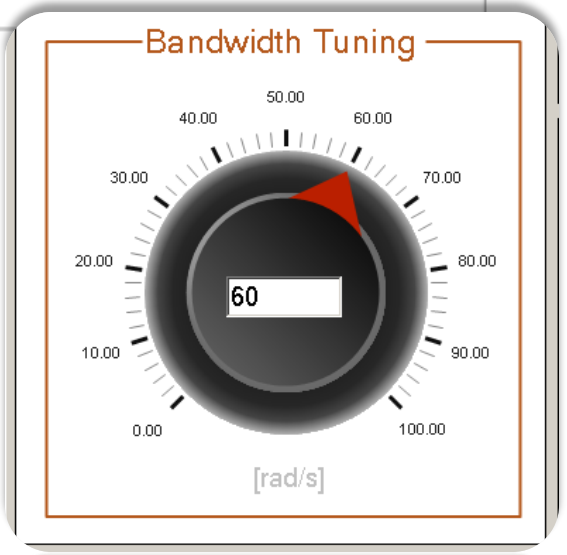
InstaSPIN-MOTION GUI

Step 3 – Profile your Moves



Higher Bandwidth = Tighter Control

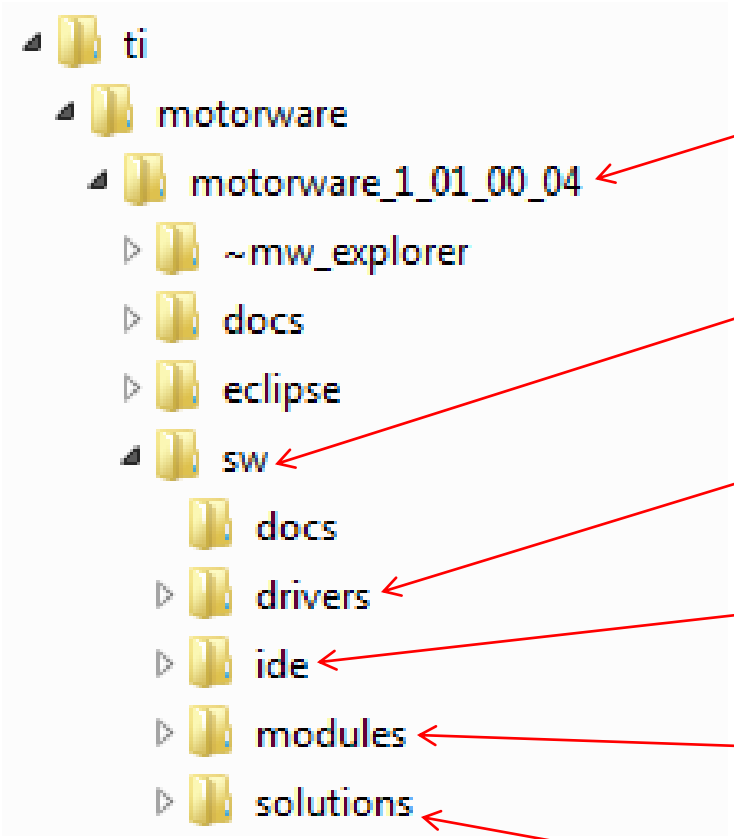




The SpinTAC™ Advantage

InstaSPIN-FOC (Demo)

MotorWare Directory Structure



Revision Number Folder

sw contains all MW code

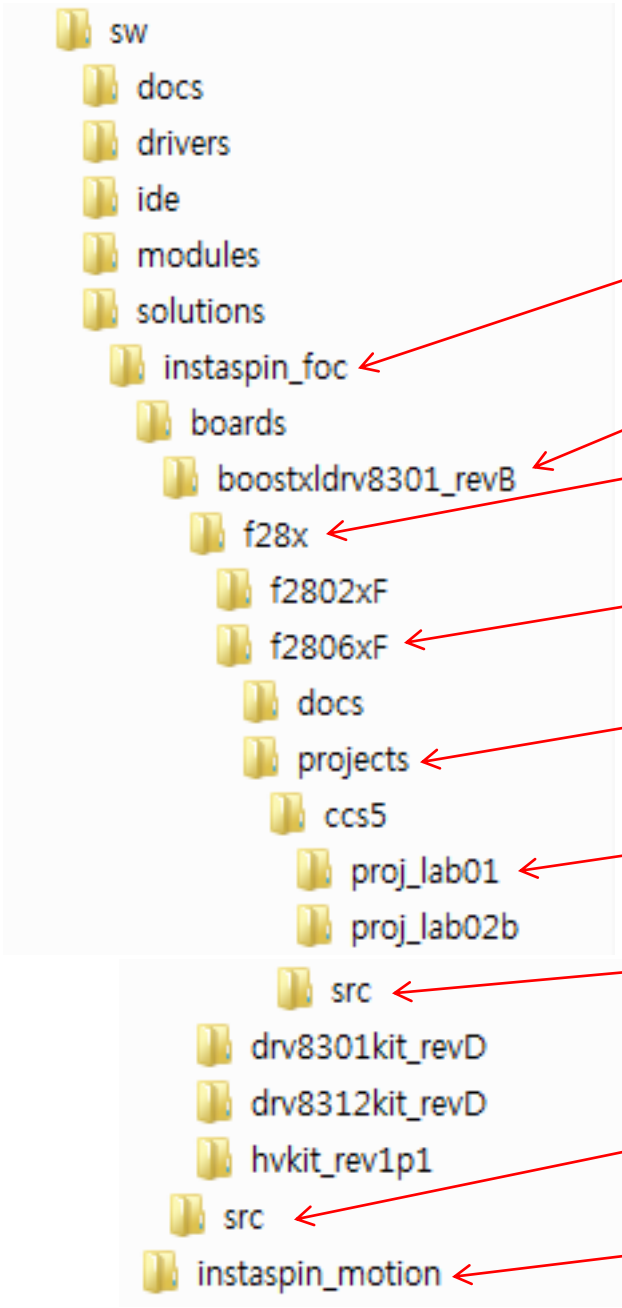
Peripheral driver code

Generic linker files

Function specific software

Kit example code

Solutions Directory Structure



FOC Software Solution

Hardware platform

Processor family

Processor target

A list of projects

Project

Common source code

Project source code

MOTION Software Solution

Lab 2b (Motor ID from RAM/FLASH)

- Call the API functions to set up the sensorless FOC system.
- Setup the user.h file for the motor and inverter.
- Start the automatic motor parameter estimation.
- Update user.h for your motor.
- Lab Procedure

  -. View Lab 2a procedure in Instaspin_labs.pdf

  (C:\ti\MotorWare\motorware_1_01_00_16\docs\labs\Instaspin_labs.pdf)

Motor Identification Parameters

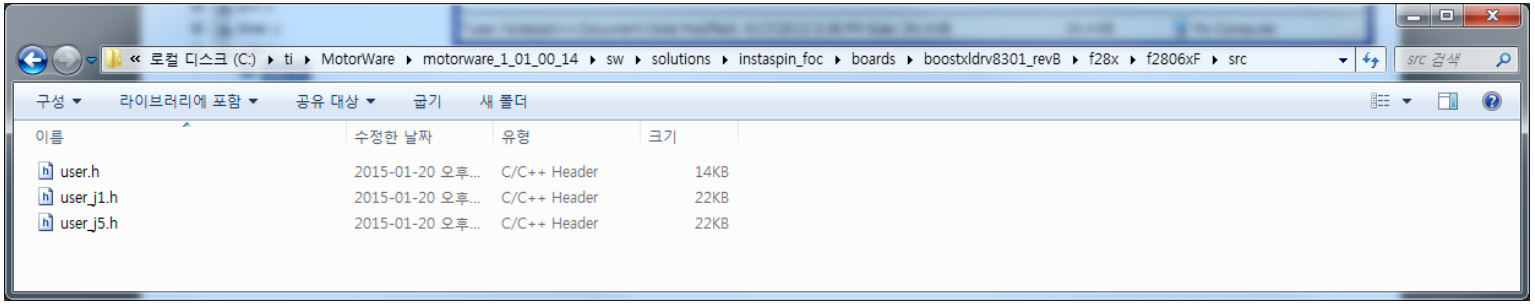
- USER_MOTOR_TYPE
  - –MOTOR_Type_Pm -> PMSM
  - –MOTOR_Type_Induction -> AC Induction
- USER_MOTOR_NUM_POLE_PAIRS
  - –Number of magnetic poles/2
- USER_MOTOR_RES_EST_CURRENT
  - –Current used to measure the resistance (Rs) of the motor
    - •Rs measurement requires a park start
  - –Current used to 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.

Motor Identification Parameters

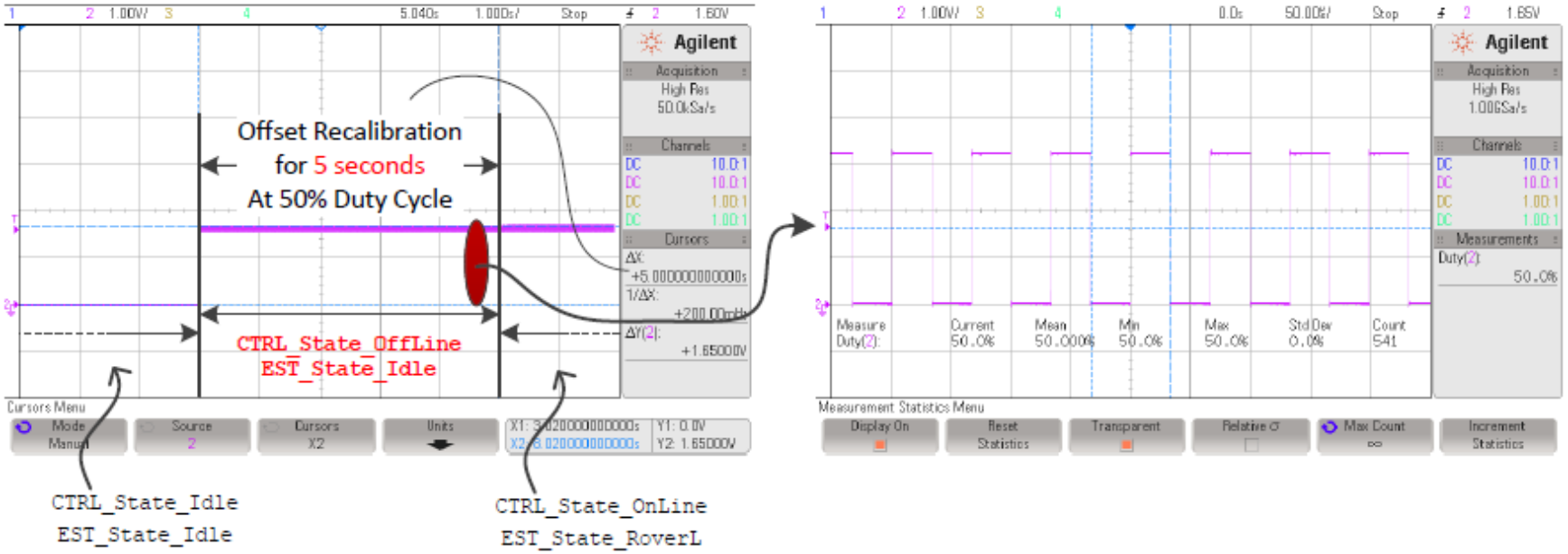
- USER_MOTOR_MAX_CURRENT
  - Maximum peak motor current
  - Equals maximum total current $\sqrt{I_q + I_d}$
- USER_MOTOR_FLUX_EST_FREQ_Hz
  - Electrical frequency that the motor is spun at during identification
  - Make higher if identification fails

InstaSPIN-FOC User Parameters

- proj_lab02b [Active - Release]
  - Includes
  - adc.c
  - clarke.c
  - clk.c
  - CodeStartBranch.asm
  - cpu.c
  - ctrl.c
  - drv8301.c
  - F28069F_ram_lnk.cmd
  - filter_fo.c
  - flash.c
  - gpio.c
  - hal.c
  - ipark.c
  - offset.c
  - osc.c
  - park.c
  - pid.c
  - pie.c
  - pll.c
  - proj_lab02b.c
  - pwm.c
  - pwr.c
  - spi.c
  - svgen.c
  - timer.c
  - traj.c
  - usDelay.asm
  - user.c
    - math.h
    - sw/modules/ctrl/src/32b/ctrl.h
    - user.h
      - USER_calcPIgains(CTRL_Handle) : void
      - USER_checkForErrors(USER_Params*) : void
      - USER_computeFlux(CTRL_Handle, const _iq) : _iq
      - USER_computeFlux_pu_to_VpHz_sf(void) : _iq
      - USER_computeFlux_pu_to_Wb_sf(void) : _iq
      - USER_computeTorque_Flux_Iq_pu_to_Nm_sf(void) : _iq
      - USER_computeTorque_Ibin(CTRL_Handle, const _iq, const _iq) : _iq
      - USER_computeTorque_Ls_Id_Iq_pu_to_Nm_sf(void) : _iq
      - USER_computeTorque_Nm(CTRL_Handle, const _iq, const _iq) : _iq
      - USER_getErrorCode(USER_Params*) : USER_ErrorCode_e
      - USER_setErrorCode(USER_Params*, const USER_ErrorCode_e) : void
      - USER_setParams(USER_Params*) : void

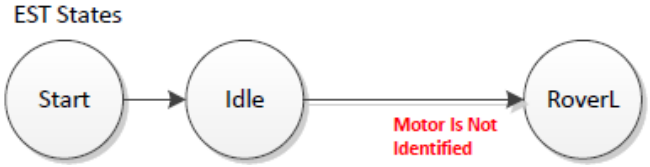


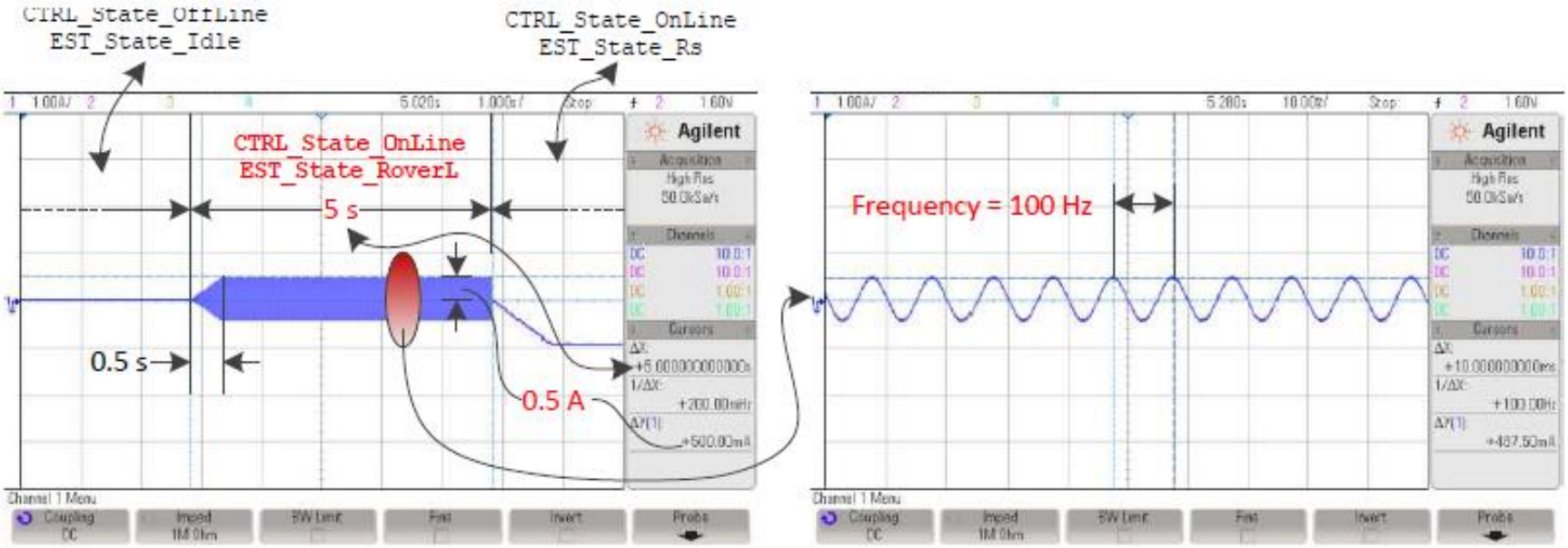
CTRL_State_OffLine and EST_State_Idle



drv.adcBias	struct _DRV_AdcDat...	{...}
I	struct _MATH_vec3_	{...}
value	long[3]	0x0000BB46@Data
(x)= [0]	long	0.8795804977 (Q-Value(24))
(x)= [1]	long	0.8918017149 (Q-Value(24))
(x)= [2]	long	0.8876305819 (Q-Value(24))
V	struct _MATH_vec3_	{...}
value	long[3]	0x0000BB4C@Data
(x)= [0]	long	0.2504473329 (Q-Value(24))
(x)= [1]	long	0.2495527864 (Q-Value(24))
(x)= [2]	long	0.2473165989 (Q-Value(24))

CTRL_State_OnLine and EST_State_RoverL



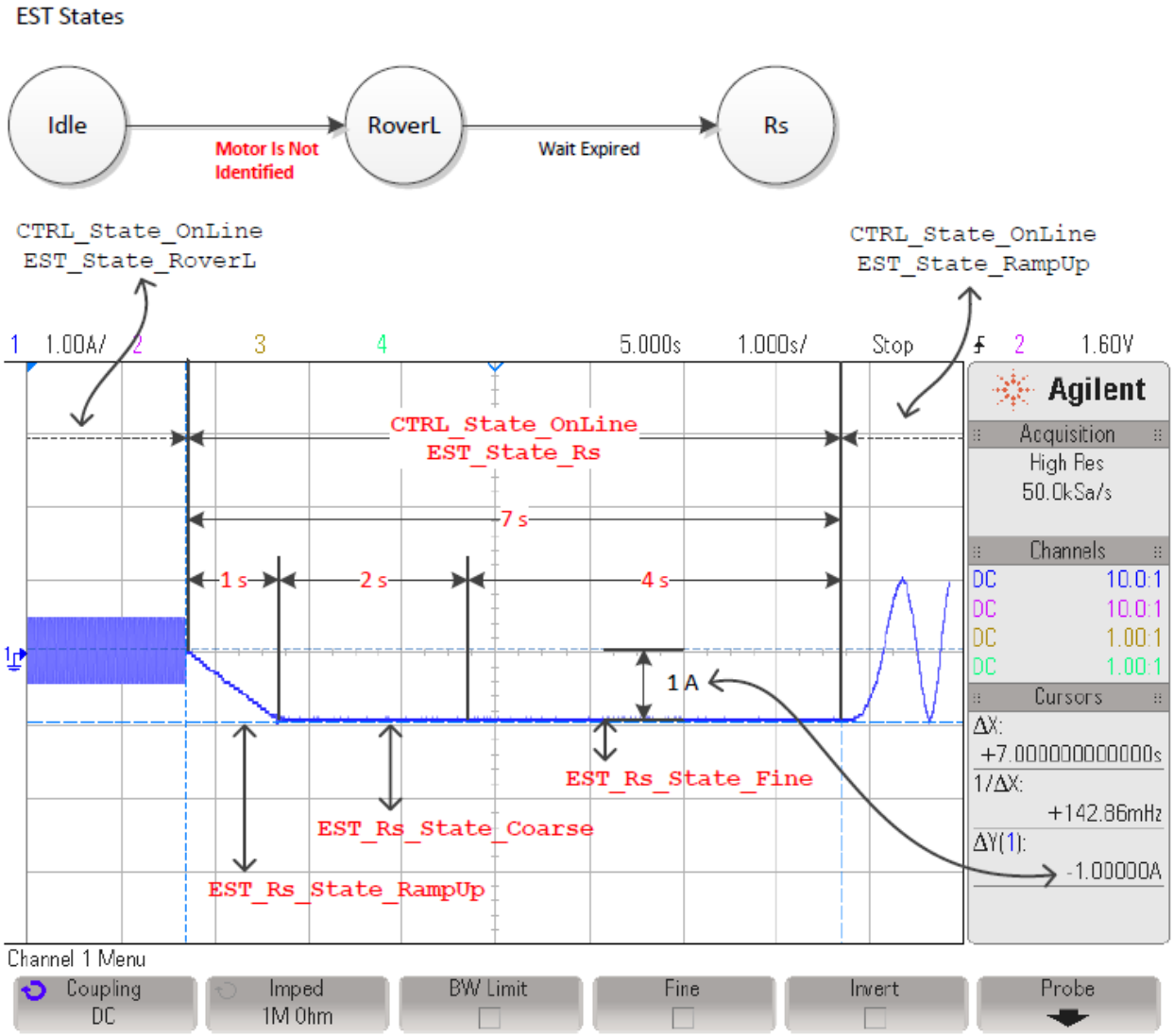


void CTRL_setKi(CTRL_Handle handle,**const** CTRL_Type_e ctrlType,**const** _iq Ki);

void CTRL_setKp(CTRL_Handle handle,**const** CTRL_Type_e ctrlType,**const** _iq Kp);

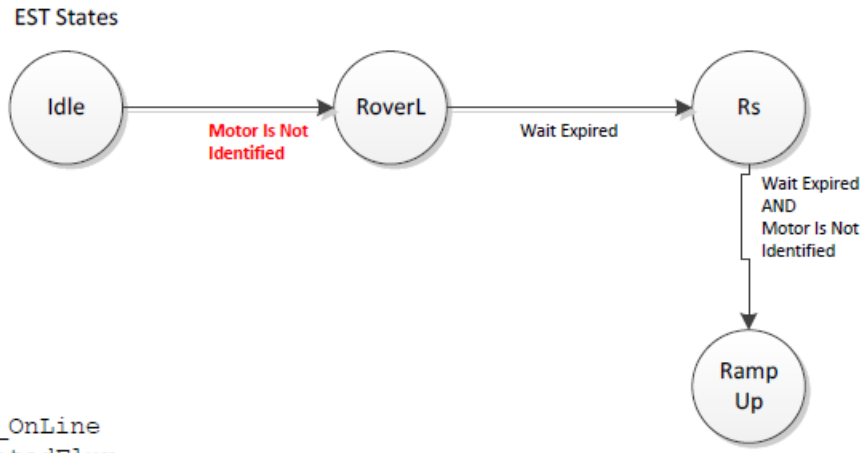
void CTRL_setGains(CTRL_Handle handle,**const** CTRL_Type_e ctrlType,
const _iq Kp,**const** _iq Ki,**const** _iq Kd);

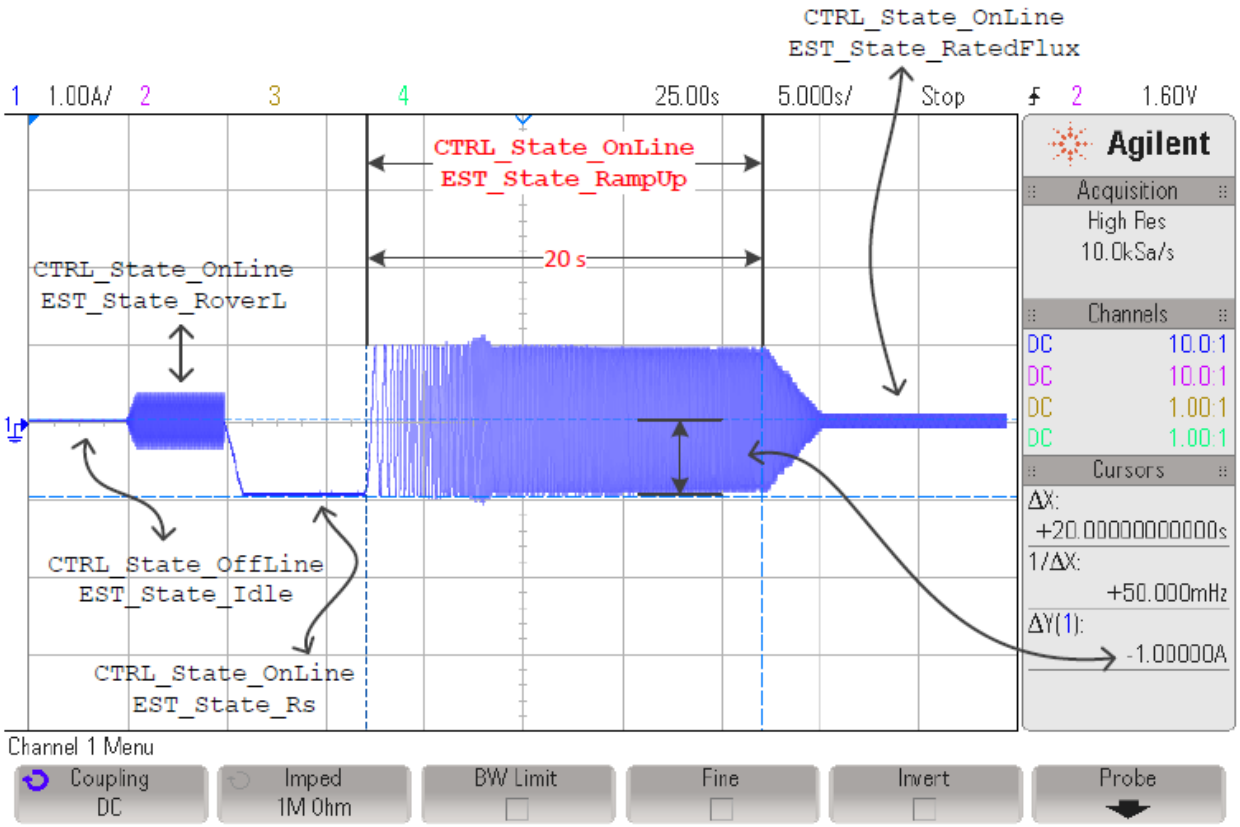
CTRL_State_OnLine and EST_State_Rs



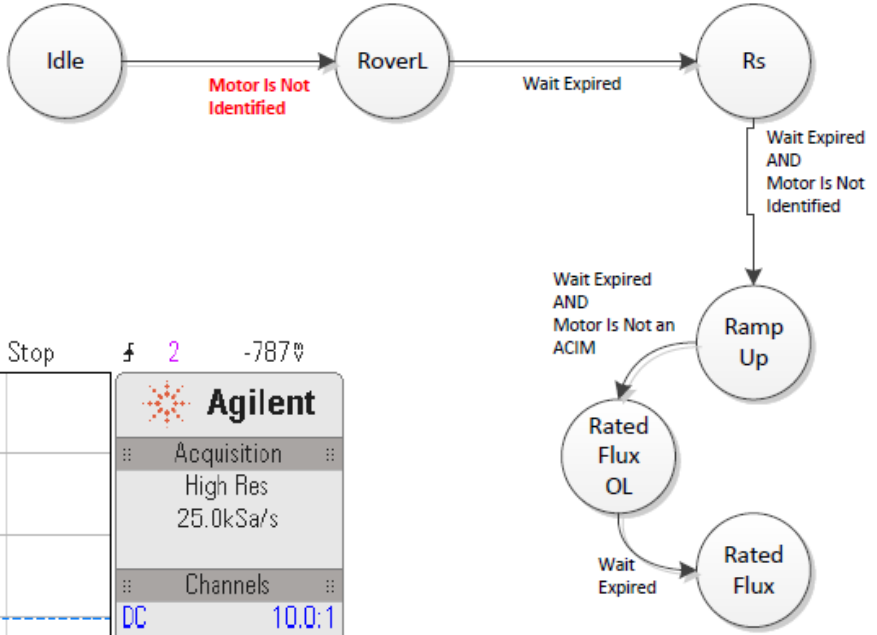

```
// get the stator resistance
gMotorVars.Rs_Ohm = EST_getRs_Ohm(obj->estHandle);
```

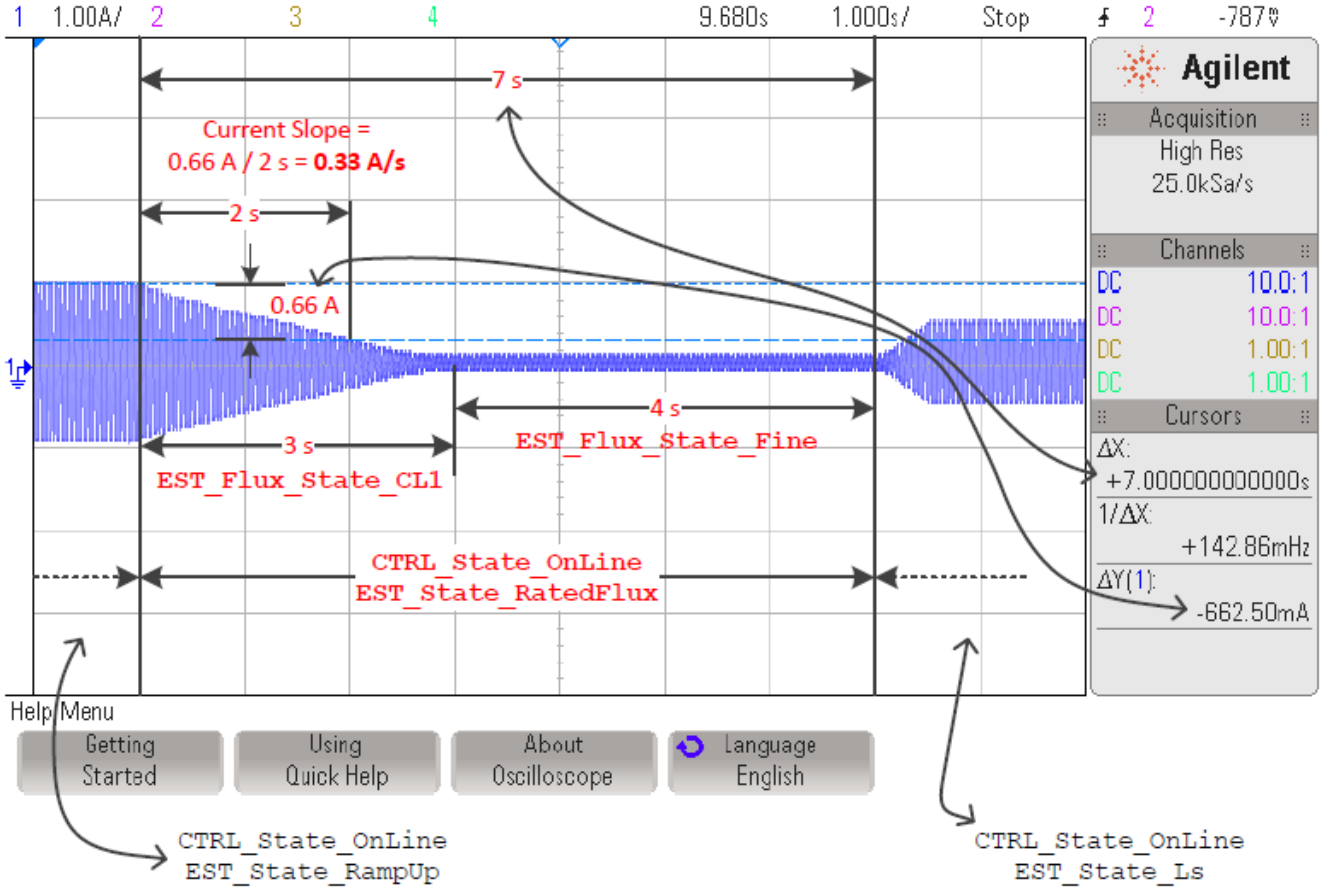
CTRL_State_OnLine and EST_State_RampUp



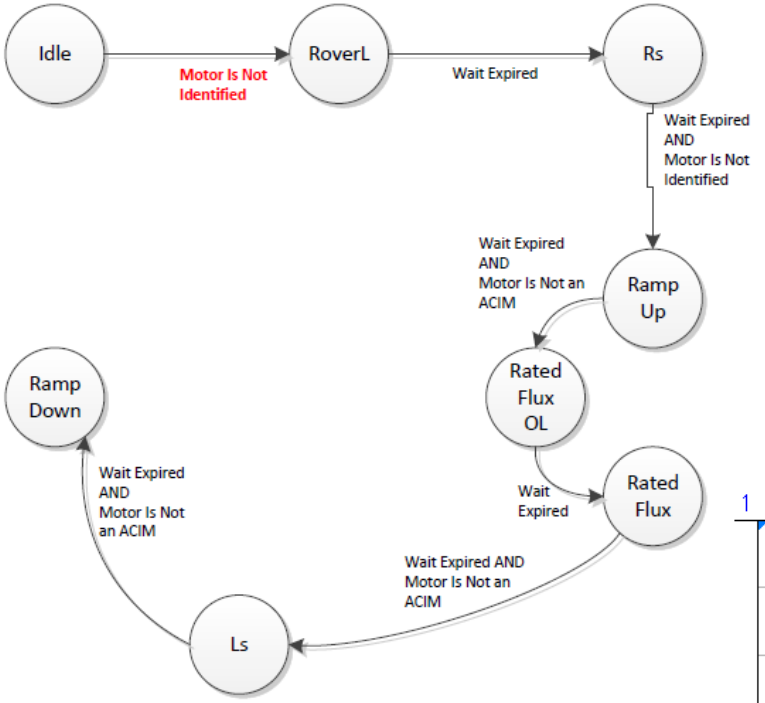


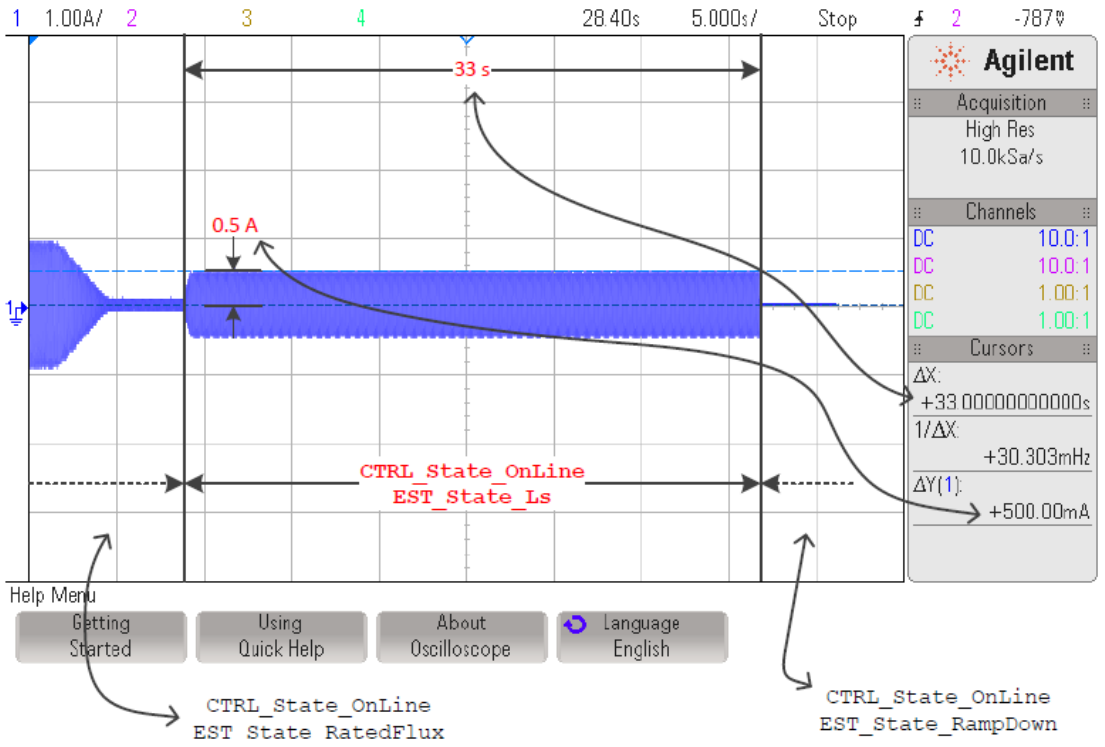
CTRL_State_OnLine and EST_State_RatedFlux



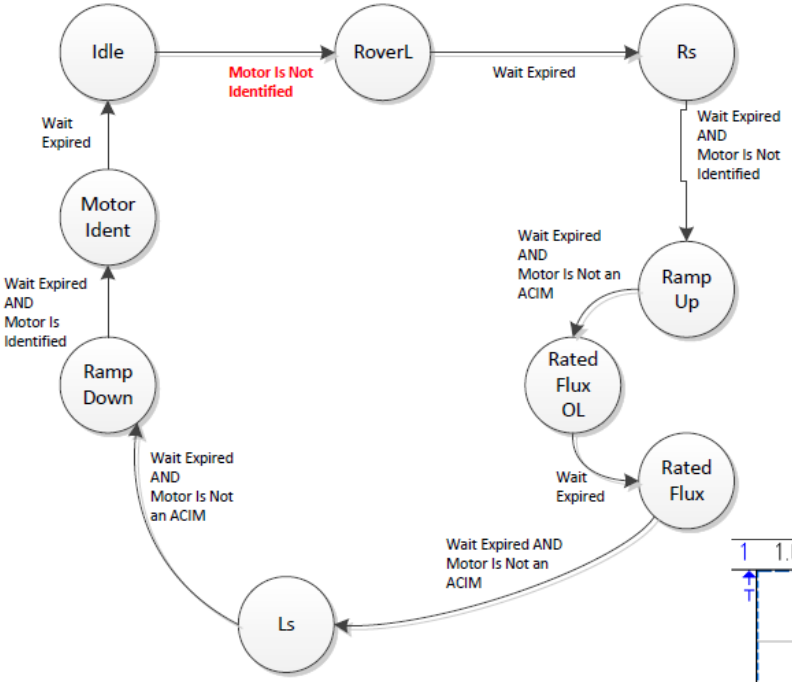


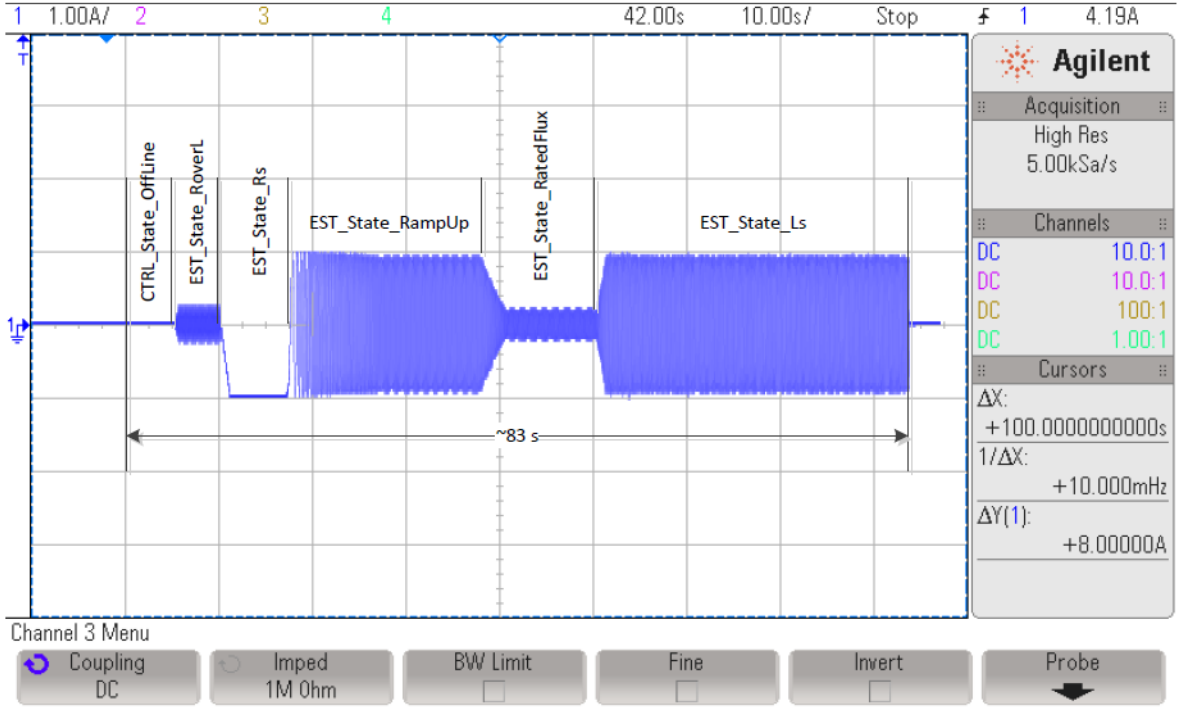
CTRL_State_OnLine and EST_State_Ls



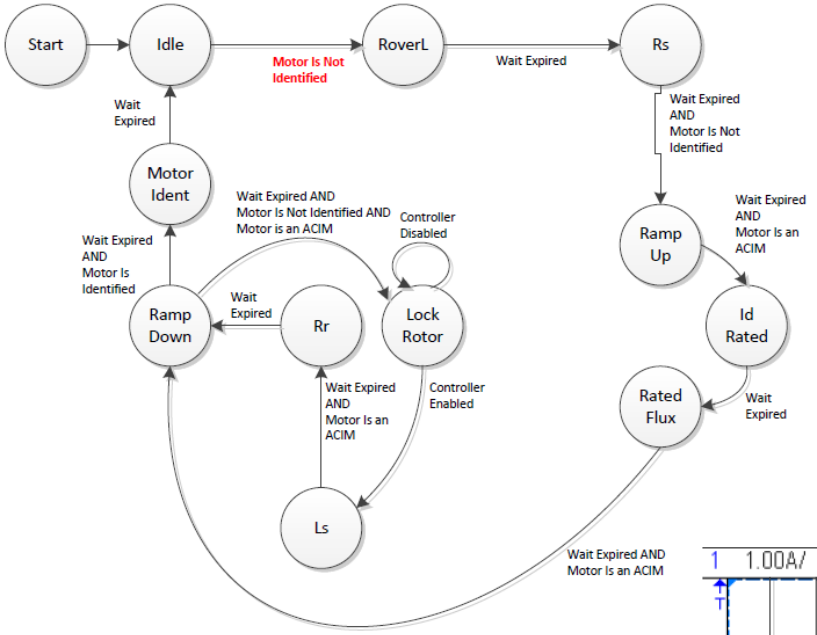


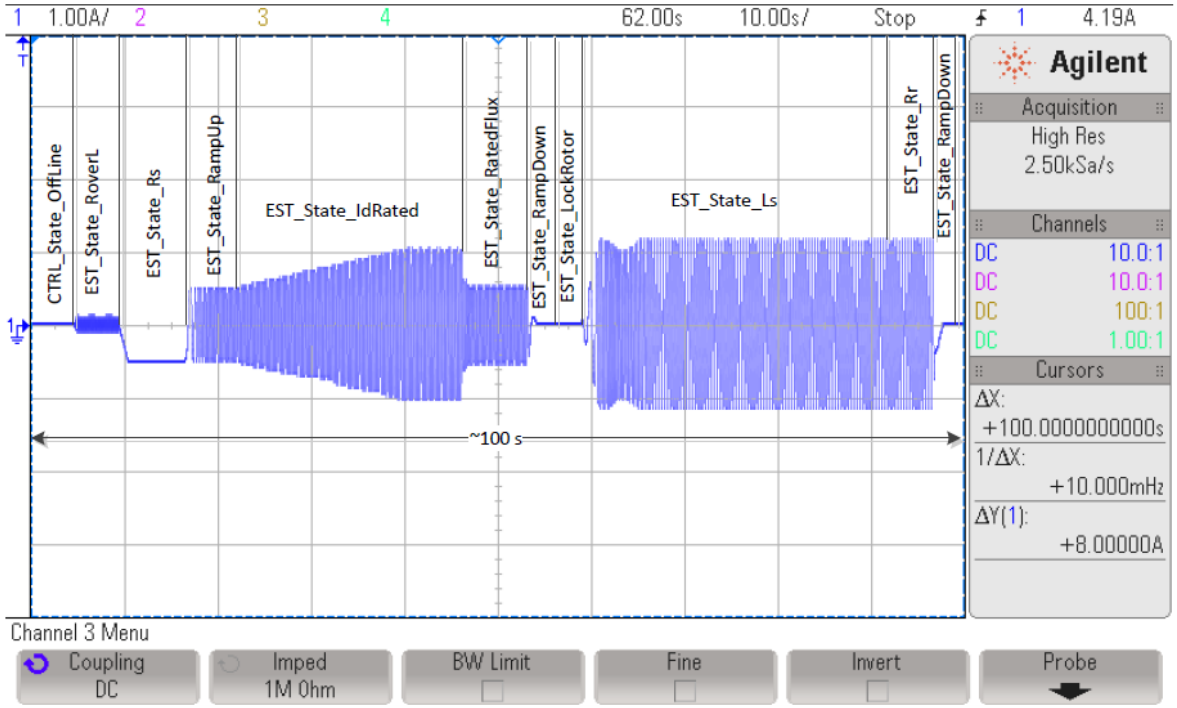
Full Identification of PMSM Motors - Summary



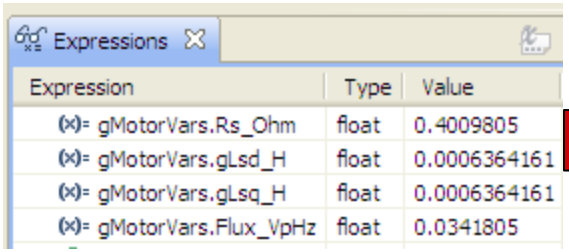


Full Identification of ACIM Motors - Summary





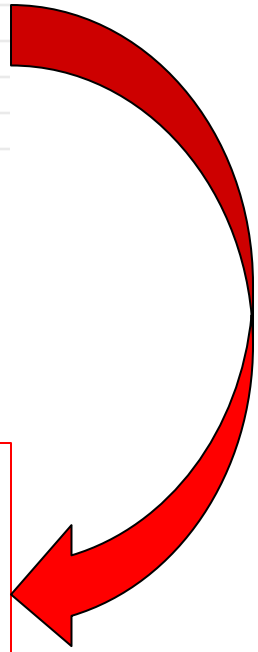
Updating User.h from Motor ID



#define USER_MOTOR_Rs (0.4009805)
#define USER_MOTOR_Ls_d (0.0006364161)
#define USER_MOTOR_Ls_q (0.0006364161)
#define USER_MOTOR_RATED_FLUX (0.0341805)

Updating User.h from Bias values

gMotorVars.I_bias	struct _MATH_vec3_	{...}
value	long[3]	0x00B3DC@Program (Q-Value(24))
[0]	long	0.8322011828 (Q-Value(24))
[1]	long	0.8340322971 (Q-Value(24))
[2]	long	0.8305362463 (Q-Value(24))
gMotorVars.V_bias	struct _MATH_vec3_	{...}
value	long[3]	0x00B3E2@Program (Q-Value(24))
[0]	long	0.4993093014 (Q-Value(24))
[1]	long	0.499632597 (Q-Value(24))
[2]	long	0.4965568185 (Q-Value(24))



```
#define I_A_offset (0.8322011828)
#define I_B_offset (0.8340322971)
#define I_C_offset (0.8305362463)
#define V_A_offset (0.4993093014)
#define V_B_offset (0.499632597)
#define V_C_offset (0.4965568185)
```

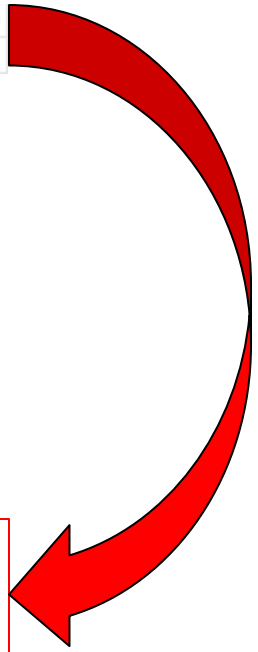
InstaSPIN-Motion (Demo)

Lab 5c (Inertia ID)

- Call the API functions to set up SpinTAC™ Velocity Identify
- Start the inertia identification process
- Update the inertia value for your motor in user.h

Updating User.h from Inertia and Friction

(x)= gMotorVars.SpinTAC.InertiaEstimate_Aperkrpm	long	0.02479678392 (Q-Value(24))
(x)= gMotorVars.SpinTAC.FrictionEstimate_Aperkrpm	long	0.1705551744 (Q-Value(24))



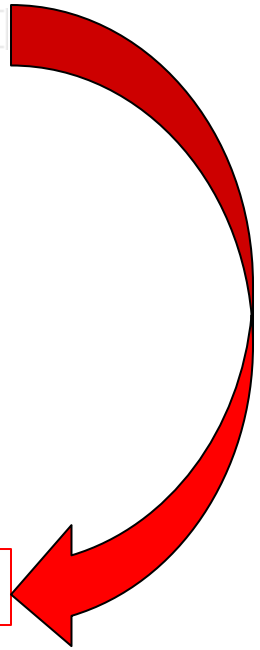
```
#define USER_SYSTEM_INERTIA (0.02479678392)
#define USER_SYSTEM_FRICTION (0.1705551744)
```

Lab 5e (Tuning SpinTAC Speed Controller)

- Quickly tune the SpinTAC™ controller for your motor
- Note the differences between tuning a PI controller and tuning the SpinTAC™ controller
- Realize the advanced control capabilities and enhanced performance characteristics of the SpinTAC™ speed controller

Updating User.h from Bandwidth

(x)= gMotorVars.SpinTAC.VelCtlBw_radps	long	80.0 (Q-Value(20))



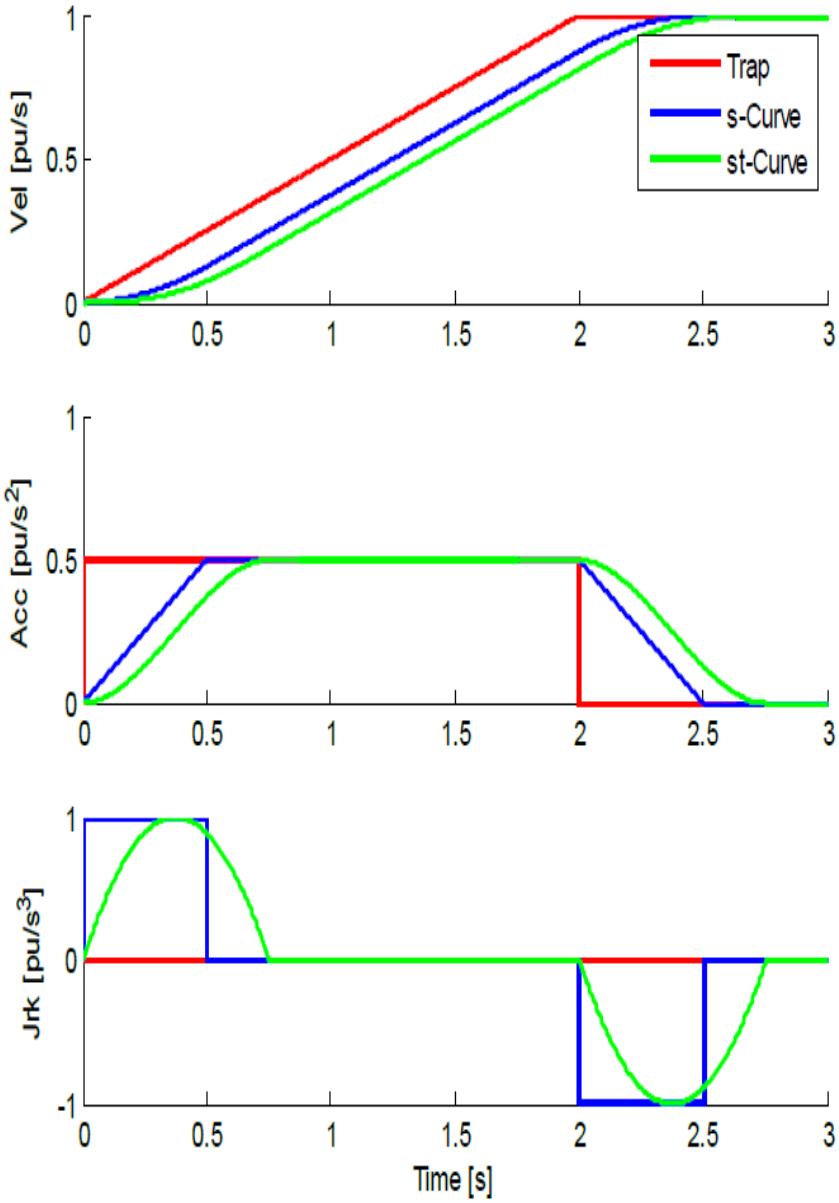
#define USER_SYSTEM_BANDWIDTH (80.0)

Lab 6a (Running SpinTAC Profile Generator)

- Use SpinTAC™ Move to transition between speeds
- Become familiar with the bounds that can be adjusted as part of SpinTAC™ Move
- Continue exploring how the SpinTAC Velocity Control takes advantage of the advanced features of SpinTAC™ Move

Available Curves

- Trapezoid
  - Bounded Acceleration
  - Impulse Jerk
- s-Curve
  - Continuous Acceleration
  - Bounded Jerk
- st-Curve (LineStream Propriety)
  - Smooth Acceleration
  - Continuous Jerk



InstaSPIN-FOC & Motion (Settings)

User Inverter Parameters

#define USER_IQ_FULL_SCALE_FREQ_HZ

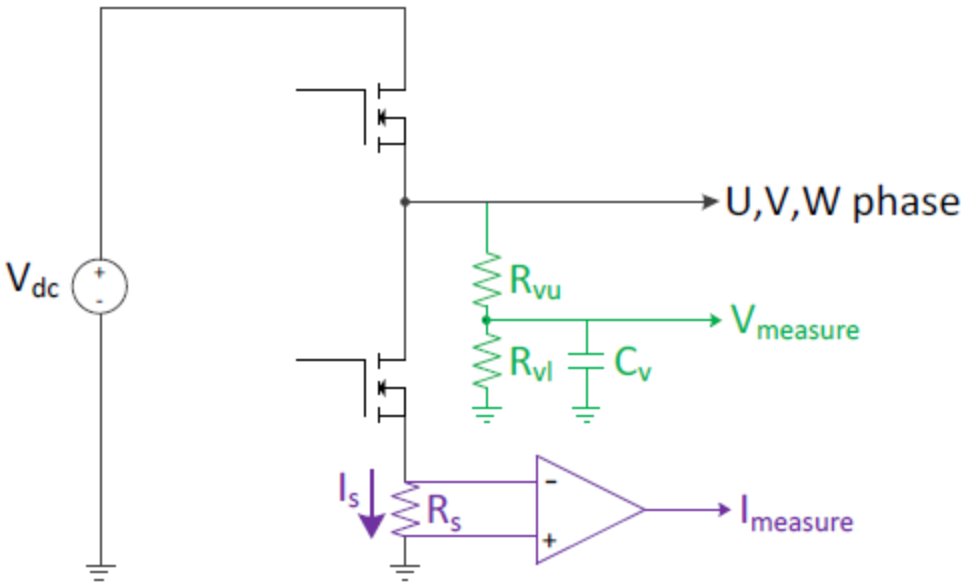
#define USER_IQ_FULL_SCALE_VOLTAGE_V
#define USER_ADC_FULL_SCALE_VOLTAGE_V

#define USER_IQ_FULL_SCALE_CURRENT_A
#define USER_ADC_FULL_SCALE_CURRENT_A

#define USER_NUM_CURRENT_SENSORS
#define USER_NUM_VOLTAGE_SENSORS

#define USER_PWM_FREQ_kHz

#define USER_VOLTAGE_FILTER_POLE_Hz



User Motor Parameters PMSM

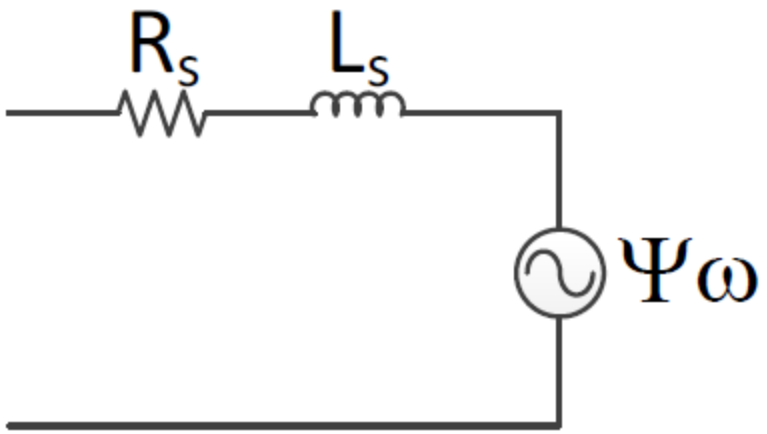
#define USER_MOTOR_TYPE
#define USER_MOTOR_NUM_POLE_PAIRS

#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



USER_IQ_FULL_SCALE_FREQ_HZ

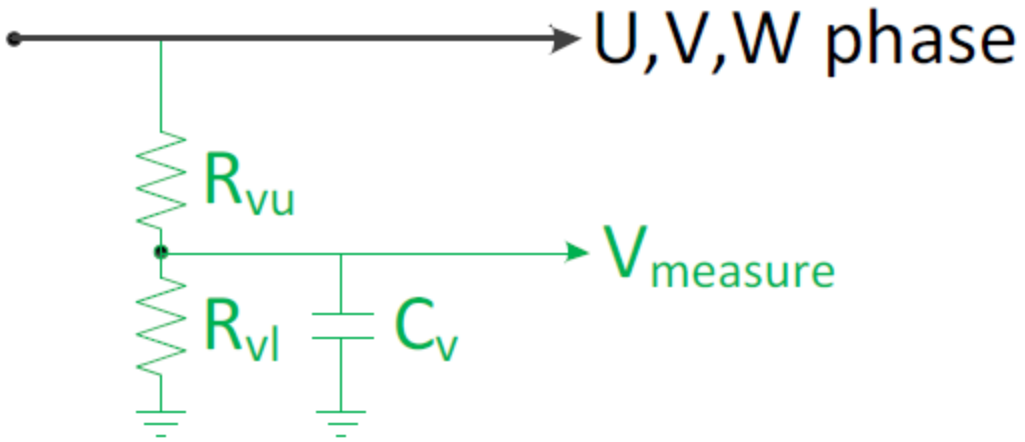
- The maximum electrical frequency of the currents and voltages that the motor will operate at.
  - –Set 20%-30% higher than the highest motor electrical frequency for headroom.

USER_ADC_FULL_SCALE_VOLTAGE_V

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

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

  - **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 ration 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.

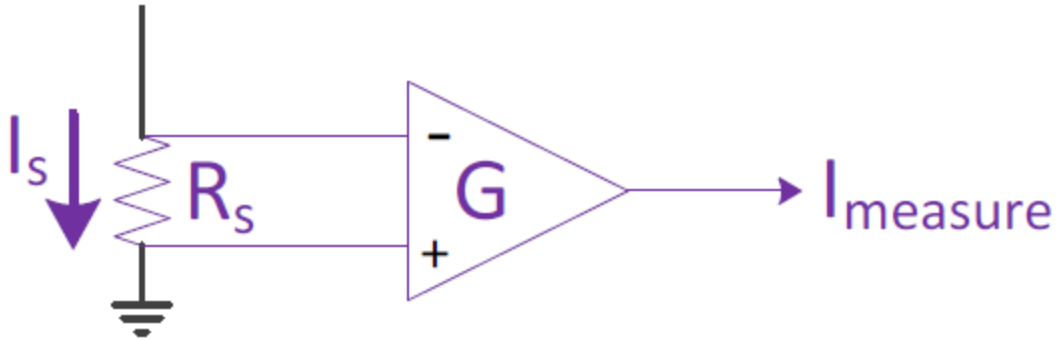
  –If the motor is operated at multiples of its base voltage, in field weakening, then USER_IQ_FULL_SCALE_VOLTAGE_V must be increased.

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.

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

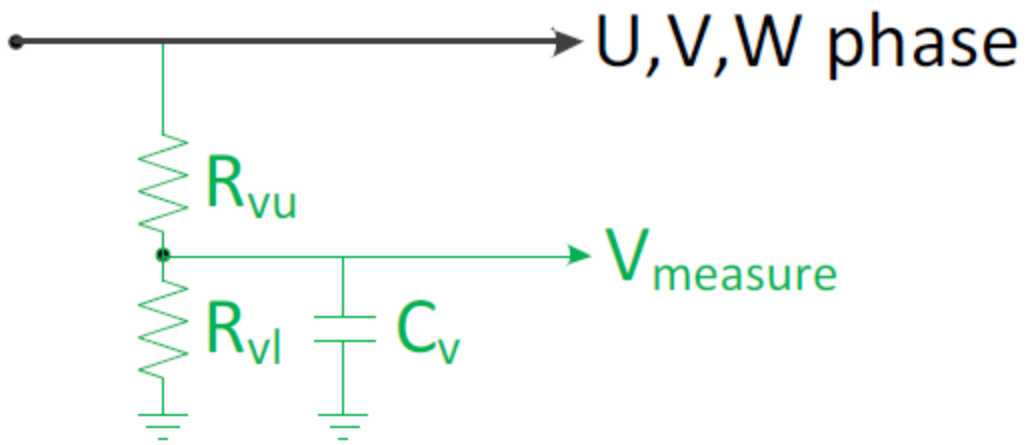
USER_PWM_FREQ_kHz

- 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.

USER_VOLTAGE_FILTER_POLE_Hz

- Defines the analog voltage filter pole location in Hz.

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



ADC Routing

- **void HAL_setupAdcs(HAL_Handle handle)** in hal.c

  –Set the channel number to sample and convert

  ADC_setSocChanNumber(obj->adcHandle, *ADC_SocNumber_1, ADC_SocChanNumber_A3*);

  –Set the start of conversion trigger

  ADC_setSocTrigSrc(obj->adcHandle, *ADC_SocNumber_1, ADC_SocTrigSrc_EPWM4_ADCSOCA*);

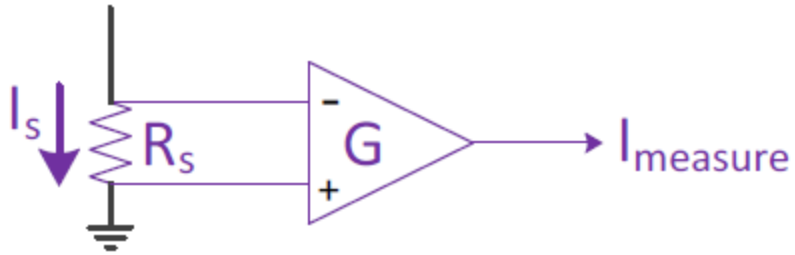
  –Set the number of cycles to sample

  ADC_setSocSampleDelay(obj->adcHandle, *ADC_SocNumber_1, ADC_SocSampleDelay_9_cycles*);

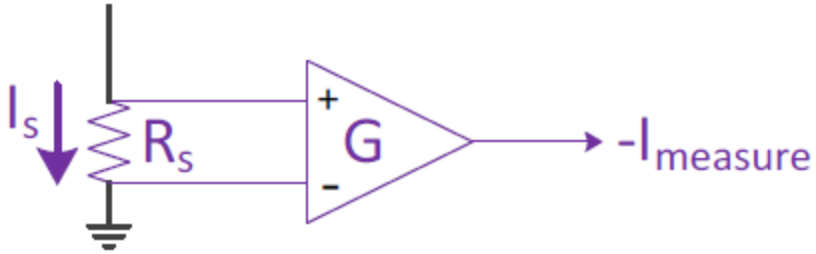
ADC Polarity

- **static inline void HAL_readAdcData(HAL_Handle handle,HAL_AdcData_t *pAdcData)** in hal.h

  – Positive polarity for op-amp feedback



  – Negative polarity for op-amp feedback



ADC Polarity change in hal.h

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

```
pAdcData->I.value[x] = value;

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

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

```
pAdcData->I.value[x] = -value;

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

Thank you