AGNIKUL

DrivePOCController
Version no - 0.17.4

Generated on Wed Jun 29 2022 12:10:57 for DrivePOCController by Doxygen 1.9.2

Wed Jun 29 2022 12:10:57

Chapter 1

DrivePOCController

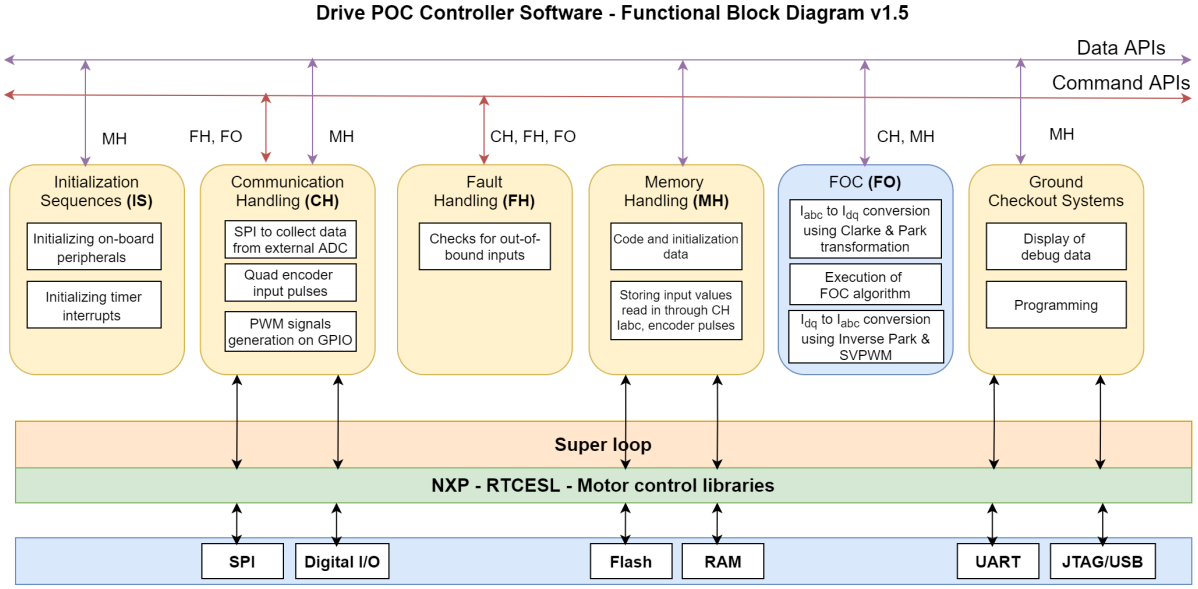
DrivePOCController_NXP + DrivePOCController Software

This a Embedded-C based software for AC Induction Motor Controller used in the POC version of the Drive project Implementation is done here by developing the V/F Control in Embedded-C on the NXP microcontroller using the evaluation board: FRDM-KV31F. NXP Controller FRDM-KV31F512VLL12 provides us with few of the Motor Controller libraries and they are used here for realization

---

Software used :- MCUXpresso IDE
Version of the software used :- V-3.0
SDK Version used :- 2.10.0
Version of the git commit :- V-0.17.4
Compiler Details :- GCC
Debugger Details :- OpenSDA with PEMicro Debugger
Date in which the documentation was made :- 28th July 2022
Documentation prepared by :- Sangeerth
People Involved in the project :- Sreedhar Mahadevan, Sangeerth P
MCUXpresso-IDE Download Link

Software architecture V-1.5- Block diagram aiding for better understanding:



The software implementation is split across the following files:
The DrivePOC_CommHandler has the functions that are associated with Communication of the NXP Microcontroller through its peripherals to the external world.

`DrivePOC_CommHandler.c && DrivePOC_CommHandler.h`

The DrivePOC_MemHandler has the functions that are associated with storing the values of data collected from the external world Here the MCU collects data from ADS Board to measure stator current, DC Bus current, DC Bus Voltage and stator voltages

`DrivePOC_MemHandler.c && DrivePOC_MemHandler.h`

The DrivePOC_Control_Loop has the functions associated with the V/f Algorithm Implementation for Acceleration, Steady State and Deceleration phase

`DrivePOC_Control_Loop.c && DrivePOC_Control_Loop.h`

The DrivePOC_Controller_NXP has the main function and the PIT Interrupt functions to schedule the actions in a timed fashion

`DrivePOC_Controller_NXP.c && DrivePOC_Controller_NXP.h`

The DrivePOC_FaultHandler has the functions associated with the fault handling. For now, the fault is handled by disabling the PWM. Later on, based on ECU demands this file have to be modified.

`DrivePOC_FaultHandler.c && DrivePOC_FaultHandler.h`

The Drive_Parameters has the Parameters of the LOX Motor-V1.5.1, Sensor data(assumed-V1.0), Mosfet Data(taken from SiC Mosfet Datasheet)

`Drive_Parameters.h`

The DrivePOC_Common_Header contains the structures and other datatypes that will be shared accross files.

`DrivePOC_Common_Header.h`

Chapter 2

Todo List

Global ltc_get_result (uint8_t channel_number, uint8_t channel_output)

Update the function based on Interrupt pin of LTC2986

Global ltc_measure_channel (uint8_t channel_number, uint8_t channel_output)

Update this function according to state machine

Global Measure_from_PTC_150 (void)

Update the threshold according to VIN & R1

Global Open_Loop_MotorStateMC (void)

: To check if this Ready state in Motor State Machine is required in future for any of the testings

Module SENSOR PARAMETERS

:-Ts value has to be updated after identifying latencies.

:-The current sensor scale factor values have to be updated.

Global System_FaultDiag (void)

: Add other functional checks to detect faults.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 6

Module Documentation

6.1 Communication Handler

Communication Handler group.

Functions

- void ReadFromEncoder (void)

  Function definition to read out of the FTM1 registers for Quadrature decoding.
- void PrintDebugInfo (void)

  Function definition to print debug data on the Serial terminal over UART @ 115200 bps baudrate.
- void DrivePOC_Comm_Handler_Init (void)

  Function definition to initialize the Communication Handler of the Drive POC controller for ADC and PWM.
- void PWM_Init (void)

  Initialize PWM Signals.
- void Comparator_Init (void)

  Initialize Comparator.
- void DrivePOC_Comm_Handler_PWMDis (void)

  Function definition to disable PWM.
- void DrivePOC_UpdateDutyCyc (GMCLIB_3COOR_T_FLT dutyCycle)

  Update duty cycle for PWM.
- void LTC_DSPI_MasterUserCallback (SPI_Type ∗base, dspi_master_edma_handle_t ∗handle, status_t status, void ∗userData)

  Function definition used a callback to check if the EDMA Transfer is successful or not for LTC Board.
- uint16_t ltc_get_start_address (uint16_t base_address, uint8_t channel_number)

  Function definition to get the start address corresponding to the channel number across which sensor is connected.
- uint32_t ltc_transfer_four_bytes (uint8_t ram_read_or_write, uint16_t start_address, uint32_t input_data)

  Function definiton to transfer 4 bytes.
- uint8_t ∗ ltc_spi_transfer_block (uint8_t TRANSFER_SIZEE, uint8_t ∗ttxx, uint8_t ∗rrxx)

  Function definition for SPI Transfer @detail The communication is Half Duplex Mode.
- void **ltc_configure_channels** (uint8_t channel_number, uint32_t channel_assignment_data)
- void delay (int delay_in_ms)

  Function which creates a delay in milli second @detail uses NOP - No Operation inside a for loop to create delays.
- uint8_t ltc_transfer_byte (uint8_t ram_read_or_write, uint16_t start_address, uint8_t input_data)

  Function definition for transfer of a single byte data.
- float_t ltc_measure_channel (uint8_t channel_number, uint8_t channel_output)

  Function definition to measure the sensor output.
- void ltc_convert_channel (uint8_t channel_number)

  Function definition which performs the Initiate conversion action.

- void ltc_wait_for_process_to_finish (void)

  Function definition to check if the value 0x40 is being returned by the LTC Board in MOSI line of the LTC Board(MISO Line of the MCU) which in turn indicates that the transfer of Initiate Conversion Command is a success.
- uint32_t ltc_get_result (uint8_t channel_number, uint8_t channel_output)

  Function which receives the Temperature or the Voltage value measured in the MISO line of the MCU.
- void ltc_print_conversion_result (uint32_t raw_conversion_result, uint8_t channel_output)

  Function definition to print the output in the console window.
- void ltc_read_voltage_or_resistance_results (uint8_t channel_number)

  Function definition to calculate if the value needed is in terms of Voltage or Resistance instead of Temperature.
- void ltc_print_fault_data (uint8_t fault_byte)

  Function to print the Fault byte in the console window indicating the kind of fault as per the Datasheet.
- void ltc_spi_edma_init (void)

  Initializes the DSPI & EDMA Peripheral.
- void comparator_init (void)

  Initializes the Comparator Peripherals for PTC-150.
- void InbuiltADC_Init (void)

  Function definition to initialize the Inbuilt ADC.
- float_t Collect_Data_from_PT_1000 (void)

  Function definition to collect the Temperature value.
- bool_t Measure_from_PTC_150 (void)

  Function definition to measure from PTC-150.
- void ads_spi_edma_init (void)

  Initializes SPI0 instance for Communication with the ADS8588 board.
- uint8_t * ads_spi_transfer_block (uint8_t *rrxx)

  Used for receiving the current and voltage value from the ADS8588 board using SPI Protocol.
- void ADS_DSPI_MasterUserCallback (SPI_Type *base, dspi_master_edma_handle_t *handle, status_t status, void *userData)

  Function definition used a callback to check if the EDMA Transfer is successful or not for ADS Board.
- void **DAC_init** (void)

  Function definition for DAC.
- void **DAC_Update** (void)

Variables

- mcs_acim_open_loop_str s_open_loop

RTD

- #define **SENSOR_TYPE_LSB** 27
- #define **SENSOR_TYPE__RTD_PT_10** (uint32_t) 0xA << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_PT_50** (uint32_t) 0xB << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_PT_100** (uint32_t) 0xC << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_PT_200** (uint32_t) 0xD << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_PT_500** (uint32_t) 0xE << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_PT_1000** (uint32_t) 0xF << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_PT_1000_375** (uint32_t) 0x10 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_NI_120** (uint32_t) 0x11 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_CUSTOM** (uint32_t) 0x12 << SENSOR_TYPE_LSB

Sense Resistor

- #define **SENSOR_TYPE__SENSE_RESISTOR** (uint32_t) 0x1D << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__NONE** (uint32_t) 0x0 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__ACTIVE_ANALOG** (uint32_t) 0x1F << SENSOR_TYPE_LSB
- #define **SENSE_RESISTOR_VALUE_LSB** 0

Direct ADC

- #define **SENSOR_TYPE__DIRECT_ADC** (uint32_t) 0x1E << SENSOR_TYPE_LSB

Thermistor

- #define **SENSOR_TYPE__THERMISTOR_44004_2P252K_25C** (uint32_t) 0x13 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_44005_3K_25C** (uint32_t) 0x14 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_44007_5K_25C** (uint32_t) 0x15 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_44006_10K_25C** (uint32_t) 0x16 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_44008_30K_25C** (uint32_t) 0x17 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_YSI_400_2P252K_25C** (uint32_t) 0x18 << SENSOR_TYPE↩ _LSB
- #define **SENSOR_TYPE__THERMISTOR_1003K_1K_25C** (uint32_t) 0x19 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_CUSTOM_STEINHART_HART** (uint32_t) 0x1A << SENSOR↩ _TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_CUSTOM_TABLE** (uint32_t) 0x1B << SENSOR_TYPE_LSB

Thermocouple

- #define **SENSOR_TYPE__TYPE_J_THERMOCOUPLE** (uint32_t) 0x1 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_K_THERMOCOUPLE** (uint32_t) 0x2 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_E_THERMOCOUPLE** (uint32_t) 0x3 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_N_THERMOCOUPLE** (uint32_t) 0x4 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_R_THERMOCOUPLE** (uint32_t) 0x5 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_S_THERMOCOUPLE** (uint32_t) 0x6 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_T_THERMOCOUPLE** (uint32_t) 0x7 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_B_THERMOCOUPLE** (uint32_t) 0x8 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__CUSTOM_THERMOCOUPLE** (uint32_t) 0x9 << SENSOR_TYPE_LSB

Off-Chip Diode

- #define **SENSOR_TYPE__OFF_CHIP_DIODE** (uint32_t) 0x1C << SENSOR_TYPE_LSB

rtd - rsense channel

- #define **RTD_RSENSE_CHANNEL_LSB** 22
- #define **RTD_RSENSE_CHANNEL__NONE** (uint32_t) 0x0 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__1** (uint32_t) 0x1 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__2** (uint32_t) 0x2 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__3** (uint32_t) 0x3 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__4** (uint32_t) 0x4 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__5** (uint32_t) 0x5 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__6** (uint32_t) 0x6 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__7** (uint32_t) 0x7 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__8** (uint32_t) 0x8 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__9** (uint32_t) 0x9 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__10** (uint32_t) 0xA << RTD_RSENSE_CHANNEL_LSB

rtd - num wires

- #define **RTD_NUM_WIRES_LSB** 20
- #define **RTD_NUM_WIRES__2_WIRE** (uint32_t) 0x0 << RTD_NUM_WIRES_LSB
- #define **RTD_NUM_WIRES__3_WIRE** (uint32_t) 0x1 << RTD_NUM_WIRES_LSB
- #define **RTD_NUM_WIRES__4_WIRE** (uint32_t) 0x2 << RTD_NUM_WIRES_LSB
- #define **RTD_NUM_WIRES__4_WIRE_KELVIN_RSENSE** (uint32_t) 0x3 << RTD_NUM_WIRES_LSB

rtd - excitation mode

- #define **RTD_EXCITATION_MODE_LSB** 18
- #define **RTD_EXCITATION_MODE__NO_ROTATION_NO_SHARING** (uint32_t) 0x0 << RTD_↩ EXCITATION_MODE_LSB
- #define **RTD_EXCITATION_MODE__NO_ROTATION_SHARING** (uint32_t) 0x1 << RTD_EXCITATION_↩ MODE_LSB
- #define **RTD_EXCITATION_MODE__ROTATION_SHARING** (uint32_t) 0x2 << RTD_EXCITATION_↩ MODE_LSB

rtd - excitation current

- #define **RTD_EXCITATION_CURRENT_LSB** 14
- #define **RTD_EXCITATION_CURRENT__EXTERNAL** (uint32_t) 0x0 << RTD_EXCITATION_CURRENT↩ _LSB
- #define **RTD_EXCITATION_CURRENT__5UA** (uint32_t) 0x1 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__10UA** (uint32_t) 0x2 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__25UA** (uint32_t) 0x3 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__50UA** (uint32_t) 0x4 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__100UA** (uint32_t) 0x5 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__250UA** (uint32_t) 0x6 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__500UA** (uint32_t) 0x7 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__1MA** (uint32_t) 0x8 << RTD_EXCITATION_CURRENT_LSB

rtd - standard

- #define **RTD_STANDARD_LSB** 12
- #define **RTD_STANDARD__EUROPEAN** (uint32_t) 0x0 << RTD_STANDARD_LSB
- #define **RTD_STANDARD__AMERICAN** (uint32_t) 0x1 << RTD_STANDARD_LSB
- #define **RTD_STANDARD__JAPANESE** (uint32_t) 0x2 << RTD_STANDARD_LSB
- #define **RTD_STANDARD__ITS_90** (uint32_t) 0x3 << RTD_STANDARD_LSB

rtd-custom

- #define **RTD_CUSTOM_ADDRESS_LSB** 6/∗rtd - custom address∗/
- #define **RTD_CUSTOM_LENGTH_1_LSB** 0/∗rtd - custom length-1∗/
- #define **RTD_CUSTOM_VALUES_LSB** 31/∗rtd - custom values∗/

active analog - differential

- #define **ACTIVE_ANALOG_DIFFERENTIAL_LSB** 26
- #define **ACTIVE_ANALOG_DIFFERENTIAL** (uint32_t) 0x0 << ACTIVE_ANALOG_DIFFERENTIAL_LSB
- #define **ACTIVE_ANALOG_SINGLE_ENDED** (uint32_t) 0x1 << ACTIVE_ANALOG_DIFFERENTIAL_LSB
- #define **ACTIVE_ANALOG_CUSTOM_ADDRESS_LSB** 6 /∗active analog - custom address∗/
- #define **ACTIVE_ANALOG_CUSTOM_LENGTH_1_LSB** 0 /∗active analog - custom length-1∗/
- #define **ACTIVE_ANALOG_CUSTOM_VALUES_LSB** 31 /∗active analog - custom values∗/

Direct ADC - differential

- #define **DIRECT_ADC_DIFFERENTIAL_LSB** 26
- #define **DIRECT_ADC_DIFFERENTIAL** (uint32_t) 0x0 << DIRECT_ADC_DIFFERENTIAL_LSB
- #define **DIRECT_ADC_SINGLE_ENDED** (uint32_t) 0x1 << DIRECT_ADC_DIFFERENTIAL_LSB

Direct ADC - custom

- #define **DIRECT_ADC_CUSTOM_LSB** 25
- #define **DIRECT_ADC_CUSTOM__NO** (uint32_t) 0x0 << DIRECT_ADC_CUSTOM_LSB
- #define **DIRECT_ADC_CUSTOM__YES** (uint32_t) 0x1 << DIRECT_ADC_CUSTOM_LSB
- #define **DIRECT_ADC_CUSTOM_ADDRESS_LSB** 6 /*Direct ADC - custom address*/
- #define **DIRECT_ADC_CUSTOM_LENGTH_1_LSB** 0 /*Direct ADC - custom length-1*/
- #define **DIRECT_ADC_CUSTOM_VALUES_LSB** 31 /*Direct ADC - custom values*/

thermistor - rsense channel

- #define **THERMISTOR_RSENSE_CHANNEL_LSB** 22
- #define **THERMISTOR_RSENSE_CHANNEL__NONE** (uint32_t) 0x0 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__1** (uint32_t) 0x1 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__2** (uint32_t) 0x2 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__3** (uint32_t) 0x3 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__4** (uint32_t) 0x4 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__5** (uint32_t) 0x5 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__6** (uint32_t) 0x6 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__7** (uint32_t) 0x7 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__8** (uint32_t) 0x8 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__9** (uint32_t) 0x9 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__10** (uint32_t) 0xA << THERMISTOR_RSENSE_↩ CHANNEL_LSB

thermistor - differential

- #define **THERMISTOR_DIFFERENTIAL_LSB** 21
- #define **THERMISTOR_DIFFERENTIAL** (uint32_t) 0x0 << THERMISTOR_DIFFERENTIAL_LSB
- #define **THERMISTOR_SINGLE_ENDED** (uint32_t) 0x1 << THERMISTOR_DIFFERENTIAL_LSB

thermistor - excitation mode

- #define **THERMISTOR_EXCITATION_MODE_LSB** 19
- #define **THERMISTOR_EXCITATION_MODE__NO_SHARING_NO_ROTATION** (uint32_t) 0x0 << THERMISTOR_EXCITATION_MODE_LSB
- #define **THERMISTOR_EXCITATION_MODE__SHARING_ROTATION** (uint32_t) 0x1 << THERMISTOR↩ _EXCITATION_MODE_LSB
- #define **THERMISTOR_EXCITATION_MODE__SHARING_NO_ROTATION** (uint32_t) 0x2 << THERMISTOR↩ _EXCITATION_MODE_LSB

thermistor - excitation current

- #define **THERMISTOR_EXCITATION_CURRENT_LSB** 15
- #define **THERMISTOR_EXCITATION_CURRENT__INVALID** (uint32_t) 0x0 << THERMISTOR_↩ EXCITATION_CURRENT_LSB

- #define **THERMISTOR_EXCITATION_CURRENT__250NA** (uint32_t) 0x1 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__500NA** (uint32_t) 0x2 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__1UA** (uint32_t) 0x3 << THERMISTOR_EXCITATION↩ _CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__5UA** (uint32_t) 0x4 << THERMISTOR_EXCITATION↩ _CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__10UA** (uint32_t) 0x5 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__25UA** (uint32_t) 0x6 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__50UA** (uint32_t) 0x7 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__100UA** (uint32_t) 0x8 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__250UA** (uint32_t) 0x9 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__500UA** (uint32_t) 0xA << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__1MA** (uint32_t) 0xB << THERMISTOR_EXCITATION↩ _CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__AUTORANGE** (uint32_t) 0xC << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__INVALID_** (uint32_t) 0xD << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__INVALID__** (uint32_t) 0xE << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__EXTERNAL** (uint32_t) 0xF << THERMISTOR_↩ EXCITATION_CURRENT_LSB

thermistor-address

- #define **THERMISTOR_CUSTOM_ADDRESS_LSB** 6 /∗ thermistor - custom address∗/
- #define **THERMISTOR_CUSTOM_LENGTH_1_LSB** 0 /∗thermistor - custom length-1∗/
- #define **THERMISTOR_CUSTOM_VALUES_LSB** 31 /∗thermistor - custom values∗/

Thermocouple - cold junction ch

- #define **TC_COLD_JUNCTION_CH_LSB** 22
- #define **TC_COLD_JUNCTION_CH__NONE** (uint32_t) 0x0 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__1** (uint32_t) 0x1 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__2** (uint32_t) 0x2 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__3** (uint32_t) 0x3 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__4** (uint32_t) 0x4 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__5** (uint32_t) 0x5 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__6** (uint32_t) 0x6 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__7** (uint32_t) 0x7 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__8** (uint32_t) 0x8 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__9** (uint32_t) 0x9 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__10** (uint32_t) 0xA << TC_COLD_JUNCTION_CH_LSB

thermocouple - differential

- #define **TC_DIFFERENTIAL_LSB** 21
- #define **TC_DIFFERENTIAL** (uint32_t) 0x0 << TC_DIFFERENTIAL_LSB
- #define **TC_SINGLE_ENDED** (uint32_t) 0x1 << TC_DIFFERENTIAL_LSB

thermocouple - open ckt detect

- #define **TC_OPEN_CKT_DETECT_LSB** 20
- #define **TC_OPEN_CKT_DETECT__NO** (uint32_t) 0x0 << TC_OPEN_CKT_DETECT_LSB
- #define **TC_OPEN_CKT_DETECT__YES** (uint32_t) 0x1 << TC_OPEN_CKT_DETECT_LSB

thermocouple - open ckt detect current

- #define **TC_OPEN_CKT_DETECT_CURRENT_LSB** 18
- #define **TC_OPEN_CKT_DETECT_CURRENT__10UA** (uint32_t) 0x0 << TC_OPEN_CKT_DETECT_↩ CURRENT_LSB
- #define **TC_OPEN_CKT_DETECT_CURRENT__100UA** (uint32_t) 0x1 << TC_OPEN_CKT_DETECT_↩ CURRENT_LSB
- #define **TC_OPEN_CKT_DETECT_CURRENT__500UA** (uint32_t) 0x2 << TC_OPEN_CKT_DETECT_↩ CURRENT_LSB
- #define **TC_OPEN_CKT_DETECT_CURRENT__1MA** (uint32_t) 0x3 << TC_OPEN_CKT_DETECT_↩ CURRENT_LSB
- #define **TC_CUSTOM_ADDRESS_LSB** 6 /∗ tc - custom address∗/
- #define **TC_CUSTOM_LENGTH_1_LSB** 0/∗ tc - custom length-1∗/
- #define **TC_CUSTOM_VALUES_LSB** 31 /∗tc - custom values∗/

off-chip diode - differential

- #define **DIODE_DIFFERENTIAL_LSB** 26
- #define **DIODE_DIFFERENTIAL** (uint32_t) 0x0 << DIODE_DIFFERENTIAL_LSB
- #define **DIODE_SINGLE_ENDED** (uint32_t) 0x1 << DIODE_DIFFERENTIAL_LSB

diode - num readings

- #define **DIODE_NUM_READINGS_LSB** 25
- #define **DIODE_NUM_READINGS__2** (uint32_t) 0x0 << DIODE_NUM_READINGS_LSB
- #define **DIODE_NUM_READINGS__3** (uint32_t) 0x1 << DIODE_NUM_READINGS_LSB

diode - averaging on

- #define **DIODE_AVERAGING_ON_LSB** 24
- #define **DIODE_AVERAGING_OFF** (uint32_t) 0x0 << DIODE_AVERAGING_ON_LSB
- #define **DIODE_AVERAGING_ON** (uint32_t) 0x1 << DIODE_AVERAGING_ON_LSB

diode - current

- #define **DIODE_CURRENT_LSB** 22
- #define **DIODE_CURRENT__10UA_40UA_80UA** (uint32_t) 0x0 << DIODE_CURRENT_LSB
- #define **DIODE_CURRENT__20UA_80UA_160UA** (uint32_t) 0x1 << DIODE_CURRENT_LSB
- #define **DIODE_CURRENT__40UA_160UA_320UA** (uint32_t) 0x2 << DIODE_CURRENT_LSB
- #define **DIODE_CURRENT__80UA_320UA_640UA** (uint32_t) 0x3 << DIODE_CURRENT_LSB
- #define **DIODE_IDEALITY_FACTOR_LSB** 0/∗∗diode - ideality factor(eta)∗/

GLOBAL CONFIGURATION CONSTANTS

- #define **REJECTION__50_60_HZ** (uint8_t) 0x0
- #define **REJECTION__60_HZ** (uint8_t) 0x1
- #define **REJECTION__50_HZ** (uint8_t) 0x2
- #define **TEMP_UNIT__C** (uint8_t) 0x0
- #define **TEMP_UNIT__F** (uint8_t) 0x4
- #define **ENABLE_KELVIN_3_WIRE_RTD_MODE** (uint8_t) 0x10
- #define **ENABLE_KELVIN_2_WIRE_RTD_MODE** (uint8_t) 0x20
- #define **ENABLE_KELVIN_DIFFERENTIAL_THERMISTOR_MODE** (uint8_t) 0x40
- #define **DISABLE_MINUS_999** (uint8_t) 0x80

STATUS BYTE CONSTANTS

- #define **SENSOR_HARD_FAILURE** (uint8_t) 0x80
- #define **ADC_HARD_FAILURE** (uint8_t) 0x40
- #define **CJ_HARD_FAILURE** (uint8_t) 0x20
- #define **CJ_SOFT_FAILURE** (uint8_t) 0x10
- #define **SENSOR_ABOVE** (uint8_t) 0x8
- #define **SENSOR_BELOW** (uint8_t) 0x4
- #define **ADC_RANGE_ERROR** (uint8_t) 0x2
- #define **VALID** (uint8_t) 0x1

ADDRESS BASE

- #define **COMMAND_STATUS_REGISTER** (uint16_t) 0x0000
- #define **CH_ADDRESS_BASE** (uint16_t) 0x0200
- #define **VOUT_CH_BASE** (uint16_t) 0x0060
- #define **READ_CH_BASE** (uint16_t) 0x0010
- #define **CONVERSION_RESULT_MEMORY_BASE** (uint16_t) 0x0010

DATA to be sent in the MOSI line of MCU

- #define **WRITE_TO_RAM** (uint8_t) 0x02
- #define **READ_FROM_RAM** (uint8_t) 0x03
- #define **CONVERSION_CONTROL_BYTE** (uint8_t) 0x80

OUTPUT TYPE

- #define **VOLTAGE** (uint8_t) 0x01
- #define **TEMPERATURE** (uint8_t) 0x02
- #define **CODE** (uint8_t) 0x03

LTC DSPI EDMA Peripheral

- #define **LTC_DSPI_MASTER_BASEADDR** SPI1
- #define **LTC_DSPI_MASTER_DMA_MUX_BASE** DMAMUX_BASE
- #define **LTC_DSPI_MASTER_DMA_BASE** DMA_BASE
- #define **LTC_DSPI_MASTER_DMA_RX_REQUEST_SOURCE** kDmaRequestMux0SPI1
- #define **LTC_DSPI_MASTER_CLK_SRC** DSPI1_CLK_SRC
- #define **LTC_DSPI_MASTER_CLK_FREQ** CLOCK_GetFreq(DSPI1_CLK_SRC)
- #define **LTC_DSPI_MASTER_PCS_FOR_INIT** kDSPI_Pcs0
- #define **LTC_DSPI_MASTER_PCS_FOR_TRANSFER** kDSPI_MasterPcs0
- #define **LTC_DSPI_MASTER_DMA_MUX_BASEADDR** ((DMAMUX_Type ∗)(LTC_DSPI_MASTER_DMA↩ _MUX_BASE))
- #define **LTC_DSPI_MASTER_DMA_BASEADDR** ((DMA_Type ∗)(LTC_DSPI_MASTER_DMA_BASE))
- #define **LTC_TRANSFER_SIZE** 8U /∗ Transfer dataSize ∗/
- #define **LTC_TRANSFER_BAUDRATE** 2000000U /∗ Transfer baudrate - 2M ∗/

ADS DSPI EDMA Peripheral

- #define **NUM_CHANNEL** 8
- #define **ADS_ADC_RANGE** 10.0F
- #define **ADS_DSPI_MASTER_BASEADDR** SPI0
- #define **ADS_DSPI_MASTER_DMA_MUX_BASE** DMAMUX_BASE
- #define **ADS_DSPI_MASTER_DMA_BASE** DMA_BASE
- #define **ADS_DSPI_MASTER_DMA_RX_REQUEST_SOURCE** kDmaRequestMux0SPI0Rx
- #define **ADS_DSPI_MASTER_DMA_TX_REQUEST_SOURCE** kDmaRequestMux0SPI0Tx

- #define **ADS_MASTER_CLK_SRC** DSPI0_CLK_SRC
- #define **ADS_MASTER_CLK_FREQ** 20000000
- #define **ADS_DSPI_MASTER_PCS_FOR_INIT** kDSPI_Pcs0
- #define **ADS_DSPI_MASTER_PCS_FOR_TRANSFER** kDSPI_MasterPcs0
- #define **ADS_TRANSFER_SIZE** 2U∗NUM_CHANNEL /∗ Transfer dataSize ∗/
- #define **ADS_TRANSFER_BAUDRATE** 20000000U /∗ Transfer baudrate - 20M ∗/
- #define **ADS_DSPI_MASTER_DMA_MUX_BASEADDR** ((DMAMUX_Type ∗)(ADS_DSPI_MASTER_↩ DMA_MUX_BASE))
- #define **ADS_DSPI_MASTER_DMA_BASEADDR** ((DMA_Type ∗)(ADS_DSPI_MASTER_DMA_BASE))

On-board Comparator for PTC-150

- #define **CMP_BASE** CMP0
- #define **CMP_USER_CHANNEL** 1U
- #define **CMP_DAC_CHANNEL** 7U
- #define **CMP_THRESHOLD** 31U

Inbuilt ADC & PT-1000 Sensor parameters

- #define **PT_1000_ADC16_BASE** ADC0
- #define **PT_1000_ADC16_CHANNEL_GROUP** 0U
- #define **PT_1000_ADC16_USER_CHANNEL** 8U /∗ PTB0, ADC0_SE8 ∗/
- #define **FSL_FEATURE_ADC16_MAX_RESOLUTION** (16)
- #define **ALPHA** 0.00385F
- #define **PT_1000_A** 3.908300E-3F
- #define **PT_1000_B** -5.775E-7
- #define **PT_1000_C** –4.183000E-12
- #define **POTENTIAL_DIVIDER_INPUT** 3.0F
- #define **RESISTANCE_0_DEG** 1000.0F
- #define **POTENTIAL_DIVIDER_R1** 6800.0F

6.1.1 Detailed Description

Communication Handler group.
Group pertaining to all communication in/out of the NXP controller

6.1.2 Function Documentation

6.1.2.1 ADS_DSPI_MasterUserCallback()

```
void ADS_DSPI_MasterUserCallback (
            SPI_Type * base,
            dspi_master_edma_handle_t * handle,
            status_t status,
            void * userData )
```

Function definition used a callback to check if the EDMA Transfer is successful or not for ADS Board.

Parameters

base	DSPI peripheral base address
handle	DSPI handle pointer to dspi_master_edma_handle_t
status	using the status flag verifies if the EDMA Transfer is successful or not
userData	A callback function parameter

Returns

None

6.1.2.2 ads_spi_edma_init()

```
void ads_spi_edma_init (
            void  )
```

Initializes SPI0 instance for Communication with the ADS8588 board.

Parameters

None	

Returns

None

6.1.2.3 ads_spi_transfer_block()

```
uint8_t * ads_spi_transfer_block (
            uint8_t * rrxx )
```

Used for receiving the current and voltage value from the ADS8588 board using SPI Protocol.

Parameters

starting	address value for the array which will store the current voltage values

Returns

starting address

Attention

The syntax is with a pointer. This was made in this format keeping in mind the problem of dangling pointer.

6.1.2.4 Collect_Data_from_PT_1000()

```
float_t Collect_Data_from_PT_1000 (
            void  )
```

Function definition to collect the Temperature value.

Parameters

None	

Returns

None

6.1.2.5 Comparator_Init()

```
void Comparator_Init (
```

```
            void  )
```

Initialize Comparator.

Parameters

None	

Returns

None

6.1.2.6 comparator_init()

```
void comparator_init (
            void  )
```

Initializes the Comparator Peripherals for PTC-150.

Returns

None

6.1.2.7 delay()

```
void delay (
            int delay_in_ms )
```

Function which creates a delay in milli second @detail uses NOP - No Operation inside a for loop to create delays.

Note

Can standardize this function using Counter to provide delays

Parameters

delay_in_ms	Delay value in milliseconds

6.1.2.8 DrivePOC_Comm_Handler_Init()

```
void DrivePOC_Comm_Handler_Init (
            void  )
```

Function definition to initialize the Communication Handler of the Drive POC controller for ADC and PWM.

Parameters

None	

Returns

None

6.1.2.9 DrivePOC_Comm_Handler_PWMDis()

```
void DrivePOC_Comm_Handler_PWMDis (
            void  )
```

Function definition to disable PWM.

Parameters

None	

Returns

None

6.1.2.10 DrivePOC_UpdateDutyCyc()

```
void DrivePOC_UpdateDutyCyc (
            GMCLIB_3COOR_T_FLT dutyCycle )
```

Update duty cycle for PWM.

Parameters

dutyCycle	- Duty cycle value in GMCLIB_3COOR_T_FLT

Returns

None

PWM duty cycles calculation and update
Enable PWM output

6.1.2.11 InbuiltADC_Init()

```
void InbuiltADC_Init (
            void  )
```

Function definition to initialize the Inbuilt ADC.

Parameters

None	

Returns

None

6.1.2.12 ltc_convert_channel()

```
void ltc_convert_channel (
            uint8_t channel_number )
```

Function definition which performs the Initiate conversion action.
A conversion is initiated by writing a measurement command into RAM memory location 0x000 and this function block takes care of this

Parameters

channel_number	Channel number at which the sensor is connected

Attention

If the sensor is connected between n and n-1 th channel the channel number to be chosen is n and not n-1

Returns

None

6.1.2.13 LTC_DSPI_MasterUserCallback()

```
void LTC_DSPI_MasterUserCallback (
            SPI_Type * base,
            dspi_master_edma_handle_t * handle,
            status_t status,
            void * userData )
```

Function definition used a callback to check if the EDMA Transfer is successful or not for LTC Board.

Parameters

base	DSPI peripheral base address
handle	DSPI handle pointer to dspi_master_edma_handle_t
status	using the status flag verifies if the EDMA Transfer is successful or not
userData	A callback function parameter

Returns

None

6.1.2.14 ltc_get_result()

```
uint32_t ltc_get_result (
            uint8_t channel_number,
            uint8_t channel_output )
```

Function which receives the Temperature or the Voltage value measured in the MISO line of the MCU.

Todo Update the function based on Interrupt pin of LTC2986

Parameters

channel_number	Channel number at which the sensor is connected

Attention

If the sensor is connected between n and n-1 th channel the channel number to be chosen is n and not n-1

Parameters

channel_output	Specify the kind of output, if the output we need is Voltage or Temperature or Resistance

6.1.2.15 ltc_get_start_address()

```
uint16_t ltc_get_start_address (
```

```
            uint16_t base_address,
            uint8_t channel_number )
```

Function definition to get the start address corresponding to the channel number across which sensor is connected.

Parameters

base_address	base address of the memory location
channel_number	represents the nth channel where the sensor gets connected to that particular channel and (n-1)th channel where n>1

Returns

start address corresponding to that channel

6.1.2.16 ltc_measure_channel()

```
float_t ltc_measure_channel (
            uint8_t channel_number,
            uint8_t channel_output )
```

Function definition to measure the sensor output.
We can measure the output in terms of Voltage, Resistance or Temperature and this function will have print options to display the result in the console window and hence its output is of void kind

Parameters

channel_number	Channel number at which the sensor is connected

Attention

If the sensor is connected between n and n-1 th channel the channel number to be chosen is n and not n-1

Parameters

channel_output	Specify the kind of output, if the output we need is Voltage or Temperature or Resistance

Returns

None

Attention

We have to update this part of code specific to the state machine because the value need not be printed and it has to be given as an input to the state machine to check if the Temperature values are well within the limits

Todo Update this function according to state machine

6.1.2.17 ltc_print_conversion_result()

```
void ltc_print_conversion_result (
            uint32_t raw_conversion_result,
            uint8_t channel_output )
```

Function definition to print the output in the console window.

Parameters

raw_conversion_result	32 bit output obtained in the MISO line of the MCU

Note

Of this 32 bit first 8 bit represent the kind of Fault - Refer to Page-num-36 in the LTC Board. This shows the fault bits for the Temperature sensor if it is of RTD kind

Parameters

channel_output	Specify the kind of output, if the output we need is Voltage or Temperature or Resistance

6.1.2.18 ltc_print_fault_data()

```
void ltc_print_fault_data (
            uint8_t fault_byte )
```

Function to print the Fault byte in the console window indicating the kind of fault as per the Datasheet.

Parameters

fault_byte	The first 8-bits obtained in the MISO Line of the MCU after the Initiate Conversion command indicates the Fault byte

Returns

None

6.1.2.19 ltc_read_voltage_or_resistance_results()

```
void ltc_read_voltage_or_resistance_results (
            uint8_t channel_number )
```

Function definition to calculate if the value needed is in terms of Voltage or Resistance instead of Temperature.

Parameters

channel_number	Channel number at which the sensor is connected

Attention

If the sensor is connected between n and n-1 th channel the channel number to be chosen is n and not n-1

Returns

None

6.1.2.20 ltc_spi_edma_init()

```
void ltc_spi_edma_init (
            void  )
```

Initializes the DSPI & EDMA Peripheral.

Returns

None

DMA Mux setting and EDMA init
DMA MUX init
Set up dspi master

6.1.2.21 ltc_spi_transfer_block()

```
uint8_t * ltc_spi_transfer_block (
            uint8_t TRANSFER_SIZEE,
            uint8_t * ttxx,
            uint8_t * rrxx )
```

Function definition for SPI Transfer @detail The communication is Half Duplex Mode.

Parameters

TRANSFER_SIZEE	number of bytes getting transferred
ttxx	array pointer to send TRANSFER_SIZEE of bytes from MCU to LTC Board
rrxx	array point to receive TRANSFER_SIZEE of bytes from LTC Board to MCU

Note

Had given the receive byte address also as an input because, without this the received value will disappear once the code exits this function block and this will lead to dangling pointer issue

6.1.2.22 ltc_transfer_byte()

```
uint8_t ltc_transfer_byte (
            uint8_t ram_read_or_write,
            uint16_t start_address,
            uint8_t input_data )
```

Function definition for transfer of a single byte data.

Parameters

ram_read_or_write	Read or Write Signal
start_address	Address Specific to Channel
input_data	The data that has to be sent via the MOSI line of the MCU to the LTC Board (i.e.) input to LTC Board

Note

This function is used while transferring Global Configuration Parameters

6.1.2.23 ltc_transfer_four_bytes()

```
uint32_t ltc_transfer_four_bytes (
            uint8_t ram_read_or_write,
            uint16_t start_address,
            uint32_t input_data )
```

Function definiton to transfer 4 bytes.
This function is used only while the Channel Assignment Data(Memory address Data specific to the channel) is being sent from the MCU to the LTC Board and also while receiving the measured temperature/voltage data from the LTC Board to the MCU

Parameters

ram_read_or_write	Read or Write signal is sent first from the MCU to the LTC Board
start_address	Channel specific address

See also

ltc_get_start_address()

Parameters

input_data	Represents the data which is sent as an input to the LTC Board from the MCU through MOSI Line

See also

ltc_spi_transfer_block()

6.1.2.24 ltc_wait_for_process_to_finish()

```
void ltc_wait_for_process_to_finish (
            void  )
```

Function definition to check if the value 0x40 is being returned by the LTC Board in MOSI line of the LTC Board(MISO Line of the MCU) which in turn indicates that the transfer of Initiate Conversion Command is a success.

Returns

None

See also

ltc_convert_channel()

6.1.2.25 Measure_from_PTC_150()

```
bool_t Measure_from_PTC_150 (
            void  )
```

Function definition to measure from PTC-150.

Parameters

None	

Returns

Boolean value '1' denotes that the output is higher than the set threshold value and '0' denotes that the output is lower than the threshold

Todo Update the threshold according to VIN & R1

6.1.2.26 PrintDebugInfo()

```
void PrintDebugInfo (
            void  )
```

Function definition to print debug data on the Serial terminal over UART @ 115200 bps baudrate.

Parameters

None	

Returns

None

6.1.2.27 PWM_Init()

```
void PWM_Init (
              void  )
```

Initialize PWM Signals.

Parameters

None	

Returns

None

6.1.2.28 ReadFromEncoder()

```
void ReadFromEncoder (
              void  )
```

Function definition to read out of the FTM1 registers for Quadrature decoding.

Parameters

None	

Returns

None

6.1.3 Variable Documentation

6.1.3.1 s_open_loop

```
mcs_acim_open_loop_str s_open_loop  [extern]
```

V/F Control Algorithm parameters

6.2 V/F Control Algorithm

V/F Control Algorithm group.

Functions

- void PTC_FaultDiag (void)

  Function definition corresponding to fault indicated by PTC 150 Thermocouple.

- void System_FaultDiag (void)

  Function definition corresponding to the Fault diagnosis.
- GMCLIB_3COOR_T_FLT Open_Loop_MainStateMC (void)

  Function definition corresponding to the main state machine of V/F control algorithm.
- void Open_Loop_MotorStateMC (void)

  Function definition corresponding to the Motor state machine of V/F Control algorithm.
- GMCLIB_3COOR_T_FLT Open_Loop_Control (void)

  Function definition corresponding to the Fast loop of the V/F control algorithm.
- void Get_Duty_Cycle (void)

  Function definition that gets the value of duty cycle from the duty cycle array.
- void Motor_SM_Calibration (void)

  ADC Calibration phase.
- void Motor_SM_Ready (void)

  Motor Ready State - Checks the Ready state of Gate Driver(both Ready-1 and Ready-2)

6.2.1 Detailed Description

V/F Control Algorithm group.
Group pertaining to all functionalities of the V/F Control Algorithm in the Drive POC Controller implementation

6.2.2 Function Documentation

6.2.2.1 Get_Duty_Cycle()

```
void Get_Duty_Cycle (
            void  )
```

Function definition that gets the value of duty cycle from the duty cycle array.

Parameters

None	

Returns

None

6.2.2.2 Motor_SM_Calibration()

```
void Motor_SM_Calibration (
            void  )
```

ADC Calibration phase.

Parameters

none	

Returns

None

Configure Channels at which the sensors are connected in LTC Board
Set the Global Configuration Register

6.2.2.3 Motor_SM_Ready()

```
void Motor_SM_Ready (
            void  )
```

Motor Ready State - Checks the Ready state of Gate Driver(both Ready-1 and Ready-2)

Parameters

none	

Returns

none

6.2.2.4 Open_Loop_Control()

```
GMCLIB_3COOR_T_FLT Open_Loop_Control (
            void  )
```

Function definition corresponding to the Fast loop of the V/F control algorithm.

Parameters

None	

Returns

GMCLIB_3COOR_T_FLT dutyCycles

Function call for the Main state machine of Open Loop algorithm

6.2.2.5 Open_Loop_MainStateMC()

```
GMCLIB_3COOR_T_FLT Open_Loop_MainStateMC (
            void  )
```

Function definition corresponding to the main state machine of V/F control algorithm.

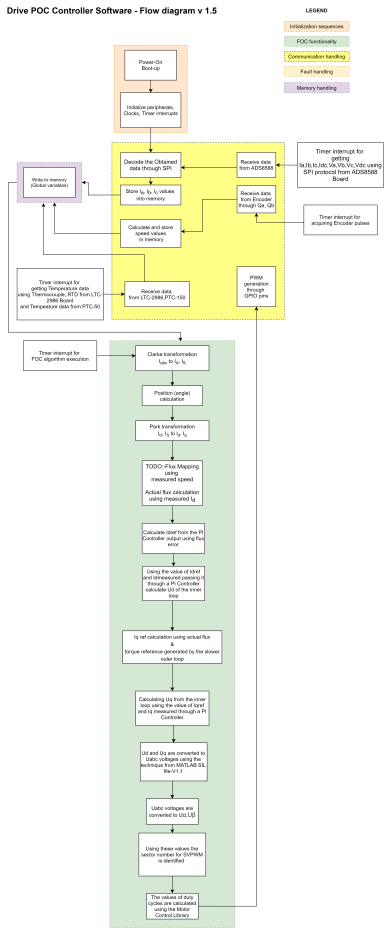
Parameters

None	

Returns

GMCLIB_3COOR_T_FLT dutyCycles

• Fault—the system detected a fault condition and waits until it is cleared. • Init—initialization of variables. • Stop—the system is initialized and waiting for the Run command. • Run—the system is running; it can be stopped by the Stop command. The following are transition functions between these state functions:
• Init → Stop —the initialization is done, the system is entering the Stop state. • Stop → Run —the Run command is applied, the system is entering the Run state (if the Run command is acknowledged). • Run → Stop —the Stop command is applied, the system is entering the Stop state (if the Stop command is acknowledged). • Fault → Stop —the fault flag is cleared, the system is entering the Stop state. • [Init, Stop, Run] → Fault—a fault condition occurred, the system is entering the Fault state.
The following image shows the software flow



Checking for faults / run command

6.2.2.6 Open_Loop_MotorStateMC()

```
void Open_Loop_MotorStateMC (
            void  )
```

Function definition corresponding to the Motor state machine of V/F Control algorithm.

Parameters

None	

Returns

None

The motor state machines are based on the main state machine structure. The Run state sub-states are added on top of the main structure to control the motors properly

Todo : To check if this Ready state in Motor State Machine is required in future for any of the testings

6.2.2.7 PTC_FaultDiag()

```
void PTC_FaultDiag (
            void  )
```

Function definition corresponding to fault indicated by PTC 150 Thermocouple.

Parameters

None	

Returns

None

6.2.2.8 System_FaultDiag()

```
void System_FaultDiag (
            void  )
```

Function definition corresponding to the Fault diagnosis.

Parameters

None	

Returns

None

Todo : Add other functional checks to detect faults.

6.3 Fault Handler

Fault Handler group.

Functions

- void Get_Fault_Duty_Cycle (void)

  Function definition that will make the Duty Cycle of all the switches to zero.

6.3.1 Detailed Description

Fault Handler group.
Group pertaining to all Memory and storage operations of the Drive POC Controller implementation

6.3.2 Function Documentation

6.3.2.1 Get_Fault_Duty_Cycle()

```
void Get_Fault_Duty_Cycle (
            void  )
```

Function definition that will make the Duty Cycle of all the switches to zero.

Parameters

None	

Returns

None

6.4 Memory Handler

Memory Handler group.

Functions

- bool DrivePOC_MH_UpdateEncoderSpeed (float rpm_recd)

  Function definition to update Encoder speed into Memory handler.
- float DrivePOC_MH_GetEncoderSpeed (void)

  Function definition to return Encoder speed from Memory handler.
- void Store_Temperature_from_PT_1000 (void)

  stores the value of temperature measured using PT-1000 RTD in the memory
- void Get_Duty_Cycle (void)

  Function definition that gets the value of duty cycle from the duty cycle array.
- void DrivePOC_MH_UpdateVIvalues (void)

  Function scales the value of Current using the Sensor scaling factors.
- void DrivePOC_MH_GetVIvalues (void)

  Function definition to allocate the variables to corresponding structures.
- void **Get_V_F_Duty_Cycle** (void)

  Function Defintion to give duty cycle during ramp up.
- void **Get_Deceleration_Duty_Cycle** (void)

  Function Declaration for Deceleration case.
- bool **Get_Start_Up_status** (void)

  Function Declaration to get Start Up state.
- bool **Get_Stop_status** (void)

  Function Declaration to get Stop done signal.

Sine Frequency

- #define **FSINE_800** 800
- #define **FSINE_669** 669
- #define **FSINE_600** 600
- #define **FSINE_500** 500
- #define **FSINE_400** 400
- #define **FSINE_300** 300
- #define **FSINE_200** 200
- #define **FSINE_100** 100

Switching Frequency 50kHz Data

- #define **FSWITCHING_800HZ_50KHZ** 50400
- #define **FSWITCHING_669HZ_50KHZ** 50175
- #define **FSWITCHING_600HZ_50KHZ** 50400
- #define **FSWITCHING_500HZ_50KHZ** 49500
- #define **FSWITCHING_400HZ_50KHZ** 50400
- #define **FSWITCHING_300HZ_50KHZ** 50400
- #define **FSWITCHING_200HZ_50KHZ** 49800
- #define **FSWITCHING_100HZ_50KHZ** 50100
- #define **SINE_LUT_800HZ_50KHZ** 21
- #define **SINE_LUT_669HZ_50KHZ** 25
- #define **SINE_LUT_600HZ_50KHZ** 28
- #define **SINE_LUT_500HZ_50KHZ** 33
- #define **SINE_LUT_400HZ_50KHZ** 42
- #define **SINE_LUT_300HZ_50KHZ** 56
- #define **SINE_LUT_200HZ_50KHZ** 83
- #define **SINE_LUT_100HZ_50KHZ** 167

Switching Frequency 40kHz Data

- #define **FSWITCHING_800HZ_40KHZ** 40800
- #define **FSWITCHING_669HZ_40KHZ** 40140
- #define **FSWITCHING_600HZ_40KHZ** 39600
- #define **FSWITCHING_500HZ_40KHZ** 40500
- #define **FSWITCHING_400HZ_40KHZ** 39600
- #define **FSWITCHING_300HZ_40KHZ** 39600
- #define **FSWITCHING_200HZ_40KHZ** 40200
- #define **FSWITCHING_100HZ_40KHZ** 40200
- #define **SINE_LUT_800HZ_40KHZ** 17
- #define **SINE_LUT_669HZ_40KHZ** 20
- #define **SINE_LUT_600HZ_40KHZ** 22
- #define **SINE_LUT_500HZ_40KHZ** 27
- #define **SINE_LUT_400HZ_40KHZ** 33
- #define **SINE_LUT_300HZ_40KHZ** 44
- #define **SINE_LUT_200HZ_40KHZ** 67
- #define **SINE_LUT_100HZ_40KHZ** 134

Switching Frequency 30kHz Data

- #define **FSWITCHING_800HZ_30KHZ** 31200
- #define **FSWITCHING_669HZ_30KHZ** 30105
- #define **FSWITCHING_600HZ_30KHZ** 30600
- #define **FSWITCHING_500HZ_30KHZ** 30000
- #define **FSWITCHING_400HZ_30KHZ** 30000
- #define **FSWITCHING_300HZ_30KHZ** 29700
- #define **FSWITCHING_200HZ_30KHZ** 30000
- #define **FSWITCHING_100HZ_30KHZ** 30000
- #define **SINE_LUT_800HZ_30KHZ** 13
- #define **SINE_LUT_669HZ_30KHZ** 15
- #define **SINE_LUT_600HZ_30KHZ** 17
- #define **SINE_LUT_500HZ_30KHZ** 20
- #define **SINE_LUT_400HZ_30KHZ** 25
- #define **SINE_LUT_300HZ_30KHZ** 33
- #define **SINE_LUT_200HZ_30KHZ** 50
- #define **SINE_LUT_100HZ_30KHZ** 100

Switching Frequency 20kHz Data

- #define **FSWITCHING_1000HZ_20KHZ** 18000
- #define **FSWITCHING_800HZ_20KHZ** 19200
- #define **FSWITCHING_669HZ_20KHZ** 20070
- #define **FSWITCHING_600HZ_20KHZ** 19800
- #define **FSWITCHING_500HZ_20KHZ** 19500
- #define **FSWITCHING_400HZ_20KHZ** 20400
- #define **FSWITCHING_300HZ_20KHZ** 19800
- #define **FSWITCHING_200HZ_20KHZ** 20400
- #define **FSWITCHING_100HZ_20KHZ** 20100
- #define **SINE_LUT_1000HZ_20KHZ** 6
- #define **SINE_LUT_800HZ_20KHZ** 8
- #define **SINE_LUT_669HZ_20KHZ** 10
- #define **SINE_LUT_600HZ_20KHZ** 11
- #define **SINE_LUT_500HZ_20KHZ** 13
- #define **SINE_LUT_400HZ_20KHZ** 17
- #define **SINE_LUT_300HZ_20KHZ** 22
- #define **SINE_LUT_200HZ_20KHZ** 37
- #define **SINE_LUT_100HZ_20KHZ** 67

Switching Frequency 10kHz Data

- #define **FSWITCHING_800HZ_10KHZ** 9600
- #define **FSWITCHING_669HZ_10KHZ** 10035
- #define **FSWITCHING_600HZ_10KHZ** 10800
- #define **FSWITCHING_500HZ_10KHZ** 10500
- #define **FSWITCHING_400HZ_10KHZ** 9600
- #define **FSWITCHING_300HZ_10KHZ** 9900
- #define **FSWITCHING_200HZ_10KHZ** 10200
- #define **FSWITCHING_100HZ_10KHZ** 10200
- #define **SINE_LUT_800HZ_10KHZ** 4
- #define **SINE_LUT_669HZ_10KHZ** 5
- #define **SINE_LUT_600HZ_10KHZ** 6
- #define **SINE_LUT_500HZ_10KHZ** 7
- #define **SINE_LUT_400HZ_10KHZ** 8
- #define **SINE_LUT_300HZ_10KHZ** 11
- #define **SINE_LUT_200HZ_10KHZ** 17
- #define **SINE_LUT_100HZ_10KHZ** 34

Interrupt Timing values

- #define **T_SWITCHING_50KHZ** 1000000/50000
- #define **T_SWITCHING_40KHZ** 1000000/40000
- #define **T_SWITCHING_30KHZ** 1000000/30000
- #define **T_SWITCHING_20KHZ** 1000000/20000
- #define **T_SWITCHING_10KHZ** 1000000/10000

LTC Interrupt Frequency

- #define **LTC_SWITCHING_FREQ_10KHZ** 10000
- #define **LTC_SWITCHING_FREQ_20KHZ** 20000
- #define **LTC_SWITCHING_FREQ_30KHZ** 30000
- #define **LTC_SWITCHING_FREQ_40KHZ** 40000
- #define **LTC_SWITCHING_FREQ_50KHZ** 50000

ADS Board Parameters

- #define **ADS_BOARD_POSITIVE_VOLTAGE_LIMIT** 0x07FFF
- #define **ADS_BOARD_ZERO_VAL** 0.0F

6.4.1 Detailed Description

Memory Handler group.
Group pertaining to all Memory and storage operations of the Drive POC Controller implementation

6.4.2 Function Documentation

6.4.2.1 DrivePOC_MH_GetEncoderSpeed()

```
float DrivePOC_MH_GetEncoderSpeed (
            void  )
```

Function definition to return Encoder speed from Memory handler.

Parameters

None	

Returns

float - Speed of encoder in RPM stored in Memory Handler

6.4.2.2 DrivePOC_MH_GetVIvalues()

```
void DrivePOC_MH_GetVIvalues (
            void  )
```

Function definition to allocate the variables to corresponding structures.

Parameters

A	global pointer variable

Returns

None

6.4.2.3 DrivePOC_MH_UpdateEncoderSpeed()

```
bool DrivePOC_MH_UpdateEncoderSpeed (
            float rpm_recd )
```

Function definition to update Encoder speed into Memory handler.

Parameters

float	rpm_recd - Speed of encoder calculated using input pulses in RPM

Returns

True, if successful

False, if failed

6.4.2.4 DrivePOC_MH_UpdateVIvalues()

```
void DrivePOC_MH_UpdateVIvalues (
            void  )
```

Function scales the value of Current using the Sensor scaling factors.

Parameters

None	

Returns

None

6.4.2.5 Get_Duty_Cycle()

```
void Get_Duty_Cycle (
            void  )
```

Function definition that gets the value of duty cycle from the duty cycle array.

Parameters

None	

Returns

None

6.4.2.6 Store_Temperature_from_PT_1000()

```
void Store_Temperature_from_PT_1000 (
            void  )
```

stores the value of temperature measured using PT-1000 RTD in the memory

Parameters

Temperature	value from ADC

Returns

None

Chapter 7

Data Structure Documentation

7.1 Main_SM_ControlSig Struct Reference

`#include <DrivePOC_Common_Header.h>`

Data Fields

- bool main_sm_ctrl_init_done
- bool main_sm_ctrl_fault
- bool main_sm_ctrl_stop
- bool main_sm_ctrl_run

7.1.1 Detailed Description

Main State machine's control signals for state transition

7.1.2 Field Documentation

7.1.2.1 main_sm_ctrl_fault

`bool Main_SM_ControlSig::main_sm_ctrl_fault`
Control signal indicating Fault state of Main State machine

7.1.2.2 main_sm_ctrl_init_done

`bool Main_SM_ControlSig::main_sm_ctrl_init_done`
Control signal indicating Initialization of Main State machine

7.1.2.3 main_sm_ctrl_run

`bool Main_SM_ControlSig::main_sm_ctrl_run`
Control signal indicating Run state of Main State machine

7.1.2.4 main_sm_ctrl_stop

`bool Main_SM_ControlSig::main_sm_ctrl_stop`
Control signal indicating Stop state of Main State machine
The documentation for this struct was generated from the following file:

- DrivePOC_Common_Header.h

7.2 mcs_acim_open_loop_str Struct Reference

`#include <DrivePOC_Common_Header.h>`

Data Fields

- GMCLIB_3COOR_T_F16 **s_dutyabc_f16**
- GMCLIB_3COOR_T_FLT **s_dutyabc_flt**
- GMCLIB_3COOR_T_FLT **s_iabc**
- GMCLIB_3COOR_T_FLT **s_vabc**
- float_t **fltudcbus**
- float_t **fltidcbus**

7.2.1 Detailed Description

Structure for V/F Control Open Loop
The documentation for this struct was generated from the following file:

- DrivePOC_Common_Header.h

7.3 Motor_SM_ControlSig Struct Reference

`#include <DrivePOC_Common_Header.h>`

Data Fields

- bool calib
- bool ready
- bool start_ok
- bool spin

7.3.1 Detailed Description

Motor State machine's control signals for state transition

7.3.2 Field Documentation

7.3.2.1 calib

`bool Motor_SM_ControlSig::calib`
Control signal indicating ADC calibration of Motor State machine

7.3.2.2 ready

`bool Motor_SM_ControlSig::ready`
Control signal indicating Speed command of Motor State machine

7.3.2.3 spin

`bool Motor_SM_ControlSig::spin`
Control signal indicating Startup OK status of Motor State machine

7.3.2.4 start_ok

```
bool Motor_SM_ControlSig::start_ok
```

Control signal indicating Start up is done

The documentation for this struct was generated from the following file:

- DrivePOC_Common_Header.h

Chapter 8

File Documentation

8.1 Drive_Parameters.h File Reference

: File containing the motor and drive constant parameters

```
#include "math.h"
#include "stdlib.h"
```

Macros

MOTOR PARAMETERS

- #define **PI** 3.14f
- #define LM 0.001296f
- #define RS 0.0054f
- #define RR 0.0043f
- #define LLS 0.000021f
- #define LLR 0.000011f
- #define J 0.009370048f
- #define B 0.0007f
- #define POLE 2
- #define POLE_PAIR 1
- #define F_NOM 669
- #define V_NOM_LL_RMS 345
- #define I_NOM_LL_RMS 153
- #define V_NOM_PHASE_RMS 199
- #define I_NOM_PHASE_RMS 153
- #define RATED_PF 0.87f
- #define MOTOR_EFFICIENCY 0.934f
- #define MIN_FREQ 1
- #define MAX_FREQ 669
- #define LR 0.001307f
- #define FC 16
- #define WC 628.32f
- #define VOUT_INVERTER_MAX 9000
- #define VIN_INVERTER_MAX 9000
- #define VOUT_INVERTER_MIN 20
- #define VIN_INVERTER_MIN 20
- #define IOUT_INVERTER_MAX 10000
- #define IIN_INVERTER_MAX 10000
- #define IOUT_INVERTER_MIN 2
- #define IIN_INVERTER_MIN 2
- #define MOTOR_MAX_TEMPERATURE 120

BATTERY SIDE PARAMETERS

- #define VDC_NOM 560
- #define VDC_MIN 500

- #define VDC_MAX 1000

SAMPLING RATES

- #define TVECT 0.00001f
- #define TSC 0.0004f
- #define TS 0.00001f

SENSOR PARAMETERS

Todo *:-Ts value has to be updated after identifying latencies.*

Todo *:-The current sensor scale factor values have to be updated.*

- #define **CURRENT_SENSOR_SCALE_FACTOR_PHASE_A** 1.0F
- #define **CURRENT_SENSOR_SCALE_FACTOR_PHASE_B** 1.0F
- #define **CURRENT_SENSOR_SCALE_FACTOR_PHASE_C** 1.0F
- #define **CURRENT_SENSOR_SCALE_FACTOR_DC** 1.0F
- #define **VOLTAGE_SENSOR_SCALE_FACTOR_PHASE_A** 140.0F
- #define **VOLTAGE_SENSOR_SCALE_FACTOR_PHASE_B** 140.0F
- #define **VOLTAGE_SENSOR_SCALE_FACTOR_PHASE_C** 140.0F
- #define **VOLTAGE_SENSOR_SCALE_FACTOR_DC** 140.0F

User Inputs

- #define **LUT_TEMP_VAL** SINE_LUT_669HZ_50KHZ
- #define **SWITCHING_TIME** T_SWITCHING_50KHZ
- #define **SWITCHING_FREQUENCY** 50175
- #define **LTC_FACTOR** LTC_SWITCHING_FREQ_50KHZ
- #define **MODULATION_INDEX** 1.0F
- #define **FREQ_NEEDED** (int)669
- #define **RAMP_TIME** 2U
- #define **BIT_MAX** 4096
- #define **SWITCHINGTIME** (float_t) 1/SWITCHING_FREQUENCY
- #define **FREQ_MAP** (float_t)((float_t)BIT_MAX/(float_t)MAX_FREQ)
- #define **MAX_NUM_OF_LUT_SIZE** SWITCHING_FREQUENCY/MAX_FREQ
- #define **START_UP_OK** (int)(RAMP_TIME∗SWITCHING_FREQUENCY)
- #define **DECELERATION_TIME** 3
- #define **STOP_OK** (int)(DECELERATION_TIME∗SWITCHING_FREQUENCY)

8.1.1 Detailed Description

: File containing the motor and drive constant parameters

Author

: Sreedhar, Sangeerth @company : Agnikul Cosmos Private Limited

Attention

8.1.2 Macro Definition Documentation

8.1.2.1 B

```
#define B 0.0007f
```

Friction in Nms

8.1.2.2 F_NOM

```
#define F_NOM 669
```
Nominal frequency in Hz

8.1.2.3 FC

```
#define FC 16
```
Flux filter cut off frequency

8.1.2.4 I_NOM_LL_RMS

```
#define I_NOM_LL_RMS 153
```
Rated line to line current in ampere

8.1.2.5 I_NOM_PHASE_RMS

```
#define I_NOM_PHASE_RMS 153
```
Rated phase current in ampere

8.1.2.6 IIN_INVERTER_MAX

```
#define IIN_INVERTER_MAX 10000
```
Inverter Input Current Maximum value

8.1.2.7 IIN_INVERTER_MIN

```
#define IIN_INVERTER_MIN 2
```
Inverter Input Current Minimum value

8.1.2.8 IOUT_INVERTER_MAX

```
#define IOUT_INVERTER_MAX 10000
```
Inverter Output Current Maximum value

8.1.2.9 IOUT_INVERTER_MIN

```
#define IOUT_INVERTER_MIN 2
```
Inverter Output Current Minimum value

8.1.2.10 J

```
#define J 0.009370048f
```
Moment of Inertia in kg∗m∗m

8.1.2.11 LLR

```
#define LLR 0.000011f
```
Rotor Leakage Inductance in henry

8.1.2.12 LLS

```
#define LLS 0.000021f
```
Stator Leakage Inductance in henry

8.1.2.13 LM

```
#define LM 0.001296f
```
Magnetizing Inductance in henry

8.1.2.14 LR

```
#define LR 0.001307f
```
Rotor inductance

8.1.2.15 MAX_FREQ

```
#define MAX_FREQ 669
```
Maximum frequency in Hz

8.1.2.16 MIN_FREQ

```
#define MIN_FREQ 1
```
Minimum frequency in Hz

8.1.2.17 MOTOR_EFFICIENCY

```
#define MOTOR_EFFICIENCY 0.934f
```
efficiency

8.1.2.18 MOTOR_MAX_TEMPERATURE

```
#define MOTOR_MAX_TEMPERATURE 120
```
Allowable maximum temperature for the Motor Windings in degree Celsius

8.1.2.19 POLE

```
#define POLE 2
```
Number of poles

8.1.2.20 POLE_PAIR

```
#define POLE_PAIR 1
```
Number of pole pairs

8.1.2.21 RATED_PF

```
#define RATED_PF 0.87f
```
Rated power factor

8.1.2.22 RR

```
#define RR 0.0043f
```
Rotor resistance in ohm

8.1.2.23 RS

```
#define RS 0.0054f
```
Stator resistance in ohm

8.1.2.24 TS

```
#define TS 0.00001f
```
Sampling time for V&I

8.1.2.25 TSC

```
#define TSC 0.0004f
```
Slow loop running rate

8.1.2.26 TVECT

```
#define TVECT 0.00001f
```
Fast loop running rate

8.1.2.27 V_NOM_LL_RMS

```
#define V_NOM_LL_RMS 345
```
Rated line to line voltage in volt

8.1.2.28 V_NOM_PHASE_RMS

```
#define V_NOM_PHASE_RMS 199
```
Rated phase voltage in volt

8.1.2.29 VDC_MAX

```
#define VDC_MAX 1000
```
DC Maximum Voltage in volt

8.1.2.30 VDC_MIN

```
#define VDC_MIN 500
```
DC Minimum Voltage in volt

8.1.2.31 VDC_NOM

```
#define VDC_NOM 560
```
DC Nominal Voltage in volt

8.1.2.32 VIN_INVERTER_MAX

```
#define VIN_INVERTER_MAX 9000
```
Inverter Input Voltage Maximum value

8.1.2.33 VIN_INVERTER_MIN

```
#define VIN_INVERTER_MIN 20
```
Inverter Input Voltage Minimum value

8.1.2.34 VOUT_INVERTER_MAX

```
#define VOUT_INVERTER_MAX 9000
```
Inverter Output Voltage Maximum value

8.1.2.35 VOUT_INVERTER_MIN

```
#define VOUT_INVERTER_MIN 20
```
Inverter Output Voltage Minimum value

8.1.2.36 WC

```
#define WC 628.32f
```
Speed filter cut off frequency

8.2 Drive_Parameters.h

Go to the documentation of this file.

```
1
15 #include "math.h"
16 #include "stdlib.h"
17 /*This file will contain the motor and drive parameters*/
18
19
22 #define PI 3.14f
23 #define LM 0.001296f
24 #define RS 0.0054f
25 #define RR 0.0043f
26 #define LLS 0.000021f
27 #define LLR 0.000011f
28 #define J 0.009370048f
29 #define B 0.0007f
30 #define POLE 2
31 #define POLE_PAIR 1
32 #define F_NOM   669
33 #define V_NOM_LL_RMS    345
34 #define I_NOM_LL_RMS    153
35 #define V_NOM_PHASE_RMS 199
36 #define I_NOM_PHASE_RMS 153
37 #define RATED_PF    0.87f
38 #define MOTOR_EFFICIENCY 0.934f
39 #define MIN_FREQ 1
40 #define MAX_FREQ 669
41 #define LR 0.001307f
42 #define FC 16
43 #define WC 628.32f
44 #define VOUT_INVERTER_MAX 9000
45 #define VIN_INVERTER_MAX 9000
46 #define VOUT_INVERTER_MIN 20
47 #define VIN_INVERTER_MIN 20
48 #define IOUT_INVERTER_MAX 10000
49 #define IIN_INVERTER_MAX  10000
50 #define IOUT_INVERTER_MIN 2
51 #define IIN_INVERTER_MIN  2
52 #define MOTOR_MAX_TEMPERATURE 120
54
55
58 #define VDC_NOM 560
59 #define VDC_MIN 500
60 #define VDC_MAX 1000
62
65 #define TVECT 0.00001f
66 #define TSC 0.0004f
67 #define TS 0.00001f
72
73
77 #define CURRENT_SENSOR_SCALE_FACTOR_PHASE_A 1.0F
78 #define CURRENT_SENSOR_SCALE_FACTOR_PHASE_B 1.0F
79 #define CURRENT_SENSOR_SCALE_FACTOR_PHASE_C 1.0F
80 #define CURRENT_SENSOR_SCALE_FACTOR_DC 1.0F
81 #define VOLTAGE_SENSOR_SCALE_FACTOR_PHASE_A 140.0F
82 #define VOLTAGE_SENSOR_SCALE_FACTOR_PHASE_B 140.0F
83 #define VOLTAGE_SENSOR_SCALE_FACTOR_PHASE_C 140.0F
84 #define VOLTAGE_SENSOR_SCALE_FACTOR_DC 140.0F
86
87
88
91 #define LUT_TEMP_VAL SINE_LUT_669HZ_50KHZ
92 #define SWITCHING_TIME T_SWITCHING_50KHZ
93 #define SWITCHING_FREQUENCY 50175
94 #define LTC_FACTOR LTC_SWITCHING_FREQ_50KHZ
95 #define MODULATION_INDEX 1.0F
96 #define FREQ_NEEDED (int)669
97 #define RAMP_TIME 2U
98 #define BIT_MAX 4096
99 #define SWITCHINGTIME (float_t) 1/SWITCHING_FREQUENCY
100 #define FREQ_MAP (float_t)((float_t)BIT_MAX/(float_t)MAX_FREQ)
101 #define MAX_NUM_OF_LUT_SIZE SWITCHING_FREQUENCY/MAX_FREQ
102 #define START_UP_OK (int)(RAMP_TIME*SWITCHING_FREQUENCY)
103 #define DECELERATION_TIME 3
104 #define STOP_OK (int)(DECELERATION_TIME*SWITCHING_FREQUENCY)
```

8.3 DrivePOC_CommHandler.c File Reference

: Communication Handler functions providing for all interfacing between NXP to other sensors/systems as well PWM Generation

```
#include "peripherals.h"
#include "fsl_debug_console.h"
#include "fsl_ftm.h"
#include "math.h"
#include "gmclib_FP.h"
#include "mcdrv_frdmkv31f.h"
#include "DrivePOC_CommHandler.h"
```

Macros

- #define **M1_MCDRV_PWM_CLK_INIT**() (InitClock())

Functions

- void ReadFromEncoder (void)

  Function definition to read out of the FTM1 registers for Quadrature decoding.
- void PrintDebugInfo (void)

  Function definition to print debug data on the Serial terminal over UART @ 115200 bps baudrate.
- void DrivePOC_Comm_Handler_Init (void)

  Function definition to initialize the Communication Handler of the Drive POC controller for ADC and PWM.
- void PWM_Init (void)

  Initialize PWM Signals.
- void Comparator_Init (void)

  Initialize Comparator.
- void DrivePOC_Comm_Handler_PWMDis (void)

  Function definition to disable PWM.
- void DrivePOC_UpdateDutyCyc (GMCLIB_3COOR_T_FLT dutyCycle)

  Update duty cycle for PWM.
- uint16_t ltc_get_start_address (uint16_t base_address, uint8_t channel_number)

  Function definition to get the start address corresponding to the channel number across which sensor is connected.
- uint32_t ltc_transfer_four_bytes (uint8_t ram_read_or_write, uint16_t start_address, uint32_t input_data)

  Function definiton to transfer 4 bytes.
- uint8_t * ltc_spi_transfer_block (uint8_t TRANSFER_SIZEE, uint8_t *ttxx, uint8_t *rrxx)

  Function definition for SPI Transfer @detail The communication is Half Duplex Mode.
- void **ltc_configure_channels** (uint8_t channel_number, uint32_t channel_assignment_data)
- void LTC_DSPI_MasterUserCallback (SPI_Type *base, dspi_master_edma_handle_t *handle, status_t status, void *userData)

  Function definition used a callback to check if the EDMA Transfer is successful or not for LTC Board.
- void delay (int delay_in_ms)

  Function which creates a delay in milli second @detail uses NOP - No Operation inside a for loop to create delays.
- uint8_t ltc_transfer_byte (uint8_t ram_read_or_write, uint16_t start_address, uint8_t input_data)

  Function definition for transfer of a single byte data.
- float_t ltc_measure_channel (uint8_t channel_number, uint8_t channel_output)

  Function definition to measure the sensor output.
- void ltc_convert_channel (uint8_t channel_number)

  Function definition which performs the Initiate conversion action.
- void ltc_wait_for_process_to_finish ()

  Function definition to check if the value 0x40 is being returned by the LTC Board in MOSI line of the LTC Board(MISO Line of the MCU) which in turn indicates that the transfer of Initiate Conversion Command is a success.
- uint32_t ltc_get_result (uint8_t channel_number, uint8_t channel_output)

  Function which receives the Temperature or the Voltage value measured in the MISO line of the MCU.
- void ltc_print_conversion_result (uint32_t raw_conversion_result, uint8_t channel_output)

Function definition to print the output in the console window.

- void ltc_read_voltage_or_resistance_results (uint8_t channel_number)

  Function definition to calculate if the value needed is in terms of Voltage or Resistance instead of Temperature.
- void ltc_print_fault_data (uint8_t fault_byte)

  Function to print the Fault byte in the console window indicating the kind of fault as per the Datasheet.
- void ltc_spi_edma_init (void)

  Initializes the DSPI & EDMA Peripheral.
- float_t Collect_Data_from_PT_1000 (void)

  Function definition to collect the Temperature value.
- bool_t Measure_from_PTC_150 (void)

  Function definition to measure from PTC-150.
- void InbuiltADC_Init (void)

  Function definition to initialize the Inbuilt ADC.
- void ads_spi_edma_init (void)

  Initializes SPI0 instance for Communication with the ADS8588 board.
- uint8_t * ads_spi_transfer_block (uint8_t *rrxx)

  Used for receiving the current and voltage value from the ADS8588 board using SPI Protocol.
- void ADS_DSPI_MasterUserCallback (SPI_Type *base, dspi_master_edma_handle_t *handle, status_t status, void *userData)

  Function definition used a callback to check if the EDMA Transfer is successful or not for ADS Board.
- void **DAC_init** (void)

  Function definition for DAC.
- void **DAC_Update** (void)

Variables

- dac_config_t **dacConfigStruct**

Encoder Variables

- volatile float **g_angular_vel** = 0.0f
- volatile float **g_delta_time_ms** = 0.0f
- volatile float **g_rpm** = 0.0f
- volatile uint32_t **g_prev_encoder_count** = 0U
- volatile uint32_t **g_cur_encoder_count** = 0U
- volatile uint32_t **g_delta_encoder_count** = 0U
- volatile uint8_t **g_dir_when_overflow** = 0U
- volatile bool **b_encoder_direction** = false

PT-1000 Variables

- adc16_config_t **adc16ConfigStruct**
- adc16_channel_config_t **adc16ChannelConfigStruct**
- float_t **g_v_across_pt_1000**
- float_t **g_resistance_pt_1000**

ADS SPI Variables

- dspi_master_config_t **ads_masterConfig**
- dspi_master_edma_handle_t **ads_g_dspi_edma_m_handle**
- edma_handle_t **ads_dspiEdmaMasterRxRegToRxDataHandle**
- edma_handle_t **ads_dspiEdmaMasterTxDataToIntermediaryHandle**
- edma_handle_t **ads_dspiEdmaMasterIntermediaryToTxRegHandle**
- dspi_transfer_t **ads_masterXfer**
- uint8_t **ads_masterRxData** [ADS_TRANSFER_SIZE] = {0}
- uint8_t **ads_masterTxData** [ADS_TRANSFER_SIZE] = {0}
- edma_config_t **ads_userConfig**
- uint32_t **ads_masterRxChannel** = 3U

- uint32_t **ads_masterIntermediaryChannel** = 4U
- uint32_t **ads_masterTxChannel** = 5U

LTC SPI Variables

- uint8_t **ltc_masterRxData** [LTC_TRANSFER_SIZE] = {0}
- uint8_t **ltc_masterTxData** [LTC_TRANSFER_SIZE] = {0}
- dspi_master_config_t **ltc_masterConfig**
- dspi_master_edma_handle_t **ltc_g_dspi_edma_m_handle**
- edma_handle_t **ltc_dspiEdmaMasterRxRegToRxDataHandle**
- edma_handle_t **ltc_dspiEdmaMasterTxDataToIntermediaryHandle**
- edma_handle_t **ltc_dspiEdmaMasterIntermediaryToTxRegHandle**
- volatile bool **ltc_isTransferCompleted** = false
- dspi_transfer_t **ltc_masterXfer**
- edma_config_t **ltc_userConfig**
- uint32_t **ltc_masterRxChannel** = 0U
- uint32_t **ltc_masterIntermediaryChannel** = 1U
- uint32_t **ltc_masterTxChannel** = 2U

8.3.1 Detailed Description

: Communication Handler functions providing for all interfacing between NXP to other sensors/systems as well PWM Generation

Author

: Sreedhar, Sangeerth @company : Agnikul Cosmos Private Limited

Attention

8.4 DrivePOC_CommHandler.h File Reference

: Communication Handler functions providing for all interfacing between NXP to other sensors/systems as well PWM Generation

```
#include "fsl_dspi_edma.h"
#include "fsl_dspi.h"
#include "fsl_edma.h"
#include "fsl_dmamux.h"
#include "fsl_cmp.h"
#include "DrivePOC_Common_Header.h"
```

Macros

RTD

- #define **SENSOR_TYPE_LSB** 27
- #define **SENSOR_TYPE__RTD_PT_10** (uint32_t) 0xA << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_PT_50** (uint32_t) 0xB << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_PT_100** (uint32_t) 0xC << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_PT_200** (uint32_t) 0xD << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_PT_500** (uint32_t) 0xE << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_PT_1000** (uint32_t) 0xF << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_PT_1000_375** (uint32_t) 0x10 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_NI_120** (uint32_t) 0x11 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__RTD_CUSTOM** (uint32_t) 0x12 << SENSOR_TYPE_LSB

Sense Resistor

- #define **SENSOR_TYPE__SENSE_RESISTOR** (uint32_t) 0x1D << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__NONE** (uint32_t) 0x0 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__ACTIVE_ANALOG** (uint32_t) 0x1F << SENSOR_TYPE_LSB
- #define **SENSE_RESISTOR_VALUE_LSB** 0

Direct ADC

- #define **SENSOR_TYPE__DIRECT_ADC** (uint32_t) 0x1E << SENSOR_TYPE_LSB

Thermistor

- #define **SENSOR_TYPE__THERMISTOR_44004_2P252K_25C** (uint32_t) 0x13 << SENSOR_TYPE_↩ LSB
- #define **SENSOR_TYPE__THERMISTOR_44005_3K_25C** (uint32_t) 0x14 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_44007_5K_25C** (uint32_t) 0x15 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_44006_10K_25C** (uint32_t) 0x16 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_44008_30K_25C** (uint32_t) 0x17 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_YSI_400_2P252K_25C** (uint32_t) 0x18 << SENSOR_↩ TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_1003K_1K_25C** (uint32_t) 0x19 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_CUSTOM_STEINHART_HART** (uint32_t) 0x1A << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__THERMISTOR_CUSTOM_TABLE** (uint32_t) 0x1B << SENSOR_TYPE_LSB

Thermocouple

- #define **SENSOR_TYPE__TYPE_J_THERMOCOUPLE** (uint32_t) 0x1 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_K_THERMOCOUPLE** (uint32_t) 0x2 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_E_THERMOCOUPLE** (uint32_t) 0x3 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_N_THERMOCOUPLE** (uint32_t) 0x4 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_R_THERMOCOUPLE** (uint32_t) 0x5 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_S_THERMOCOUPLE** (uint32_t) 0x6 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_T_THERMOCOUPLE** (uint32_t) 0x7 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__TYPE_B_THERMOCOUPLE** (uint32_t) 0x8 << SENSOR_TYPE_LSB
- #define **SENSOR_TYPE__CUSTOM_THERMOCOUPLE** (uint32_t) 0x9 << SENSOR_TYPE_LSB

Off-Chip Diode

- #define **SENSOR_TYPE__OFF_CHIP_DIODE** (uint32_t) 0x1C << SENSOR_TYPE_LSB

rtd - rsense channel

- #define **RTD_RSENSE_CHANNEL_LSB** 22
- #define **RTD_RSENSE_CHANNEL__NONE** (uint32_t) 0x0 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__1** (uint32_t) 0x1 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__2** (uint32_t) 0x2 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__3** (uint32_t) 0x3 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__4** (uint32_t) 0x4 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__5** (uint32_t) 0x5 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__6** (uint32_t) 0x6 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__7** (uint32_t) 0x7 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__8** (uint32_t) 0x8 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__9** (uint32_t) 0x9 << RTD_RSENSE_CHANNEL_LSB
- #define **RTD_RSENSE_CHANNEL__10** (uint32_t) 0xA << RTD_RSENSE_CHANNEL_LSB

rtd - num wires

- #define **RTD_NUM_WIRES_LSB** 20
- #define **RTD_NUM_WIRES__2_WIRE** (uint32_t) 0x0 << RTD_NUM_WIRES_LSB
- #define **RTD_NUM_WIRES__3_WIRE** (uint32_t) 0x1 << RTD_NUM_WIRES_LSB
- #define **RTD_NUM_WIRES__4_WIRE** (uint32_t) 0x2 << RTD_NUM_WIRES_LSB
- #define **RTD_NUM_WIRES__4_WIRE_KELVIN_RSENSE** (uint32_t) 0x3 << RTD_NUM_WIRES_LSB

rtd - excitation mode

- #define **RTD_EXCITATION_MODE_LSB** 18
- #define **RTD_EXCITATION_MODE__NO_ROTATION_NO_SHARING** (uint32_t) 0x0 << RTD_↩ EXCITATION_MODE_LSB
- #define **RTD_EXCITATION_MODE__NO_ROTATION_SHARING** (uint32_t) 0x1 << RTD_EXCITATION↩ _MODE_LSB
- #define **RTD_EXCITATION_MODE__ROTATION_SHARING** (uint32_t) 0x2 << RTD_EXCITATION_↩ MODE_LSB

rtd - excitation current

- #define **RTD_EXCITATION_CURRENT_LSB** 14
- #define **RTD_EXCITATION_CURRENT__EXTERNAL** (uint32_t) 0x0 << RTD_EXCITATION_↩ CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__5UA** (uint32_t) 0x1 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__10UA** (uint32_t) 0x2 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__25UA** (uint32_t) 0x3 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__50UA** (uint32_t) 0x4 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__100UA** (uint32_t) 0x5 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__250UA** (uint32_t) 0x6 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__500UA** (uint32_t) 0x7 << RTD_EXCITATION_CURRENT_LSB
- #define **RTD_EXCITATION_CURRENT__1MA** (uint32_t) 0x8 << RTD_EXCITATION_CURRENT_LSB

rtd - standard

- #define **RTD_STANDARD_LSB** 12
- #define **RTD_STANDARD__EUROPEAN** (uint32_t) 0x0 << RTD_STANDARD_LSB
- #define **RTD_STANDARD__AMERICAN** (uint32_t) 0x1 << RTD_STANDARD_LSB
- #define **RTD_STANDARD__JAPANESE** (uint32_t) 0x2 << RTD_STANDARD_LSB
- #define **RTD_STANDARD__ITS_90** (uint32_t) 0x3 << RTD_STANDARD_LSB

rtd-custom

- #define **RTD_CUSTOM_ADDRESS_LSB** 6/∗rtd - custom address∗/
- #define **RTD_CUSTOM_LENGTH_1_LSB** 0/∗rtd - custom length-1∗/
- #define **RTD_CUSTOM_VALUES_LSB** 31/∗rtd - custom values∗/

active analog - differential

- #define **ACTIVE_ANALOG_DIFFERENTIAL_LSB** 26
- #define **ACTIVE_ANALOG_DIFFERENTIAL** (uint32_t) 0x0 << ACTIVE_ANALOG_DIFFERENTIAL_↩ LSB
- #define **ACTIVE_ANALOG_SINGLE_ENDED** (uint32_t) 0x1 << ACTIVE_ANALOG_DIFFERENTIAL_↩ LSB
- #define **ACTIVE_ANALOG_CUSTOM_ADDRESS_LSB** 6 /∗active analog - custom address∗/
- #define **ACTIVE_ANALOG_CUSTOM_LENGTH_1_LSB** 0 /∗active analog - custom length-1∗/
- #define **ACTIVE_ANALOG_CUSTOM_VALUES_LSB** 31 /∗active analog - custom values∗/

Direct ADC - differential

- #define **DIRECT_ADC_DIFFERENTIAL_LSB** 26
- #define **DIRECT_ADC_DIFFERENTIAL** (uint32_t) 0x0 << DIRECT_ADC_DIFFERENTIAL_LSB
- #define **DIRECT_ADC_SINGLE_ENDED** (uint32_t) 0x1 << DIRECT_ADC_DIFFERENTIAL_LSB

Direct ADC - custom

- #define **DIRECT_ADC_CUSTOM_LSB** 25
- #define **DIRECT_ADC_CUSTOM__NO** (uint32_t) 0x0 << DIRECT_ADC_CUSTOM_LSB
- #define **DIRECT_ADC_CUSTOM__YES** (uint32_t) 0x1 << DIRECT_ADC_CUSTOM_LSB
- #define **DIRECT_ADC_CUSTOM_ADDRESS_LSB** 6 /∗Direct ADC - custom address∗/
- #define **DIRECT_ADC_CUSTOM_LENGTH_1_LSB** 0 /∗Direct ADC - custom length-1∗/
- #define **DIRECT_ADC_CUSTOM_VALUES_LSB** 31 /∗Direct ADC - custom values∗/

thermistor - rsense channel

- #define **THERMISTOR_RSENSE_CHANNEL_LSB** 22
- #define **THERMISTOR_RSENSE_CHANNEL__NONE** (uint32_t) 0x0 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__1** (uint32_t) 0x1 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__2** (uint32_t) 0x2 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__3** (uint32_t) 0x3 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__4** (uint32_t) 0x4 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__5** (uint32_t) 0x5 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__6** (uint32_t) 0x6 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__7** (uint32_t) 0x7 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__8** (uint32_t) 0x8 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__9** (uint32_t) 0x9 << THERMISTOR_RSENSE_↩ CHANNEL_LSB
- #define **THERMISTOR_RSENSE_CHANNEL__10** (uint32_t) 0xA << THERMISTOR_RSENSE_↩ CHANNEL_LSB

thermistor - differential

- #define **THERMISTOR_DIFFERENTIAL_LSB** 21
- #define **THERMISTOR_DIFFERENTIAL** (uint32_t) 0x0 << THERMISTOR_DIFFERENTIAL_LSB
- #define **THERMISTOR_SINGLE_ENDED** (uint32_t) 0x1 << THERMISTOR_DIFFERENTIAL_LSB

thermistor - excitation mode

- #define **THERMISTOR_EXCITATION_MODE_LSB** 19
- #define **THERMISTOR_EXCITATION_MODE__NO_SHARING_NO_ROTATION** (uint32_t) 0x0 << THERMISTOR_EXCITATION_MODE_LSB
- #define **THERMISTOR_EXCITATION_MODE__SHARING_ROTATION** (uint32_t) 0x1 << THERMISTOR↩ _EXCITATION_MODE_LSB
- #define **THERMISTOR_EXCITATION_MODE__SHARING_NO_ROTATION** (uint32_t) 0x2 << THERMISTOR↩ _EXCITATION_MODE_LSB

thermistor - excitation current

- #define **THERMISTOR_EXCITATION_CURRENT_LSB** 15
- #define **THERMISTOR_EXCITATION_CURRENT__INVALID** (uint32_t) 0x0 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__250NA** (uint32_t) 0x1 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__500NA** (uint32_t) 0x2 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__1UA** (uint32_t) 0x3 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__5UA** (uint32_t) 0x4 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__10UA** (uint32_t) 0x5 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__25UA** (uint32_t) 0x6 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__50UA** (uint32_t) 0x7 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__100UA** (uint32_t) 0x8 << THERMISTOR_↩ EXCITATION_CURRENT_LSB

- #define **THERMISTOR_EXCITATION_CURRENT__250UA** (uint32_t) 0x9 << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__500UA** (uint32_t) 0xA << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__1MA** (uint32_t) 0xB << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__AUTORANGE** (uint32_t) 0xC << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__INVALID_** (uint32_t) 0xD << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__INVALID__** (uint32_t) 0xE << THERMISTOR_↩ EXCITATION_CURRENT_LSB
- #define **THERMISTOR_EXCITATION_CURRENT__EXTERNAL** (uint32_t) 0xF << THERMISTOR_↩ EXCITATION_CURRENT_LSB

thermistor-address

- #define **THERMISTOR_CUSTOM_ADDRESS_LSB** 6 /∗ thermistor - custom address∗/
- #define **THERMISTOR_CUSTOM_LENGTH_1_LSB** 0 /∗thermistor - custom length-1∗/
- #define **THERMISTOR_CUSTOM_VALUES_LSB** 31 /∗thermistor - custom values∗/

Thermocouple - cold junction ch

- #define **TC_COLD_JUNCTION_CH_LSB** 22
- #define **TC_COLD_JUNCTION_CH__NONE** (uint32_t) 0x0 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__1** (uint32_t) 0x1 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__2** (uint32_t) 0x2 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__3** (uint32_t) 0x3 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__4** (uint32_t) 0x4 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__5** (uint32_t) 0x5 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__6** (uint32_t) 0x6 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__7** (uint32_t) 0x7 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__8** (uint32_t) 0x8 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__9** (uint32_t) 0x9 << TC_COLD_JUNCTION_CH_LSB
- #define **TC_COLD_JUNCTION_CH__10** (uint32_t) 0xA << TC_COLD_JUNCTION_CH_LSB

thermocouple - differential

- #define **TC_DIFFERENTIAL_LSB** 21
- #define **TC_DIFFERENTIAL** (uint32_t) 0x0 << TC_DIFFERENTIAL_LSB
- #define **TC_SINGLE_ENDED** (uint32_t) 0x1 << TC_DIFFERENTIAL_LSB

thermocouple - open ckt detect

- #define **TC_OPEN_CKT_DETECT_LSB** 20
- #define **TC_OPEN_CKT_DETECT__NO** (uint32_t) 0x0 << TC_OPEN_CKT_DETECT_LSB
- #define **TC_OPEN_CKT_DETECT__YES** (uint32_t) 0x1 << TC_OPEN_CKT_DETECT_LSB

thermocouple - open ckt detect current

- #define **TC_OPEN_CKT_DETECT_CURRENT_LSB** 18
- #define **TC_OPEN_CKT_DETECT_CURRENT__10UA** (uint32_t) 0x0 << TC_OPEN_CKT_DETECT_↩ CURRENT_LSB
- #define **TC_OPEN_CKT_DETECT_CURRENT__100UA** (uint32_t) 0x1 << TC_OPEN_CKT_DETECT↩ _CURRENT_LSB
- #define **TC_OPEN_CKT_DETECT_CURRENT__500UA** (uint32_t) 0x2 << TC_OPEN_CKT_DETECT↩ _CURRENT_LSB
- #define **TC_OPEN_CKT_DETECT_CURRENT__1MA** (uint32_t) 0x3 << TC_OPEN_CKT_DETECT_↩ CURRENT_LSB
- #define **TC_CUSTOM_ADDRESS_LSB** 6 /∗ tc - custom address∗/
- #define **TC_CUSTOM_LENGTH_1_LSB** 0/∗ tc - custom length-1∗/
- #define **TC_CUSTOM_VALUES_LSB** 31 /∗tc - custom values∗/

off-chip diode - differential

- #define **DIODE_DIFFERENTIAL_LSB** 26
- #define **DIODE_DIFFERENTIAL** (uint32_t) 0x0 << DIODE_DIFFERENTIAL_LSB
- #define **DIODE_SINGLE_ENDED** (uint32_t) 0x1 << DIODE_DIFFERENTIAL_LSB

diode - num readings

- #define **DIODE_NUM_READINGS_LSB** 25
- #define **DIODE_NUM_READINGS__2** (uint32_t) 0x0 << DIODE_NUM_READINGS_LSB
- #define **DIODE_NUM_READINGS__3** (uint32_t) 0x1 << DIODE_NUM_READINGS_LSB

diode - averaging on

- #define **DIODE_AVERAGING_ON_LSB** 24
- #define **DIODE_AVERAGING_OFF** (uint32_t) 0x0 << DIODE_AVERAGING_ON_LSB
- #define **DIODE_AVERAGING_ON** (uint32_t) 0x1 << DIODE_AVERAGING_ON_LSB

diode - current

- #define **DIODE_CURRENT_LSB** 22
- #define **DIODE_CURRENT__10UA_40UA_80UA** (uint32_t) 0x0 << DIODE_CURRENT_LSB
- #define **DIODE_CURRENT__20UA_80UA_160UA** (uint32_t) 0x1 << DIODE_CURRENT_LSB
- #define **DIODE_CURRENT__40UA_160UA_320UA** (uint32_t) 0x2 << DIODE_CURRENT_LSB
- #define **DIODE_CURRENT__80UA_320UA_640UA** (uint32_t) 0x3 << DIODE_CURRENT_LSB
- #define **DIODE_IDEALITY_FACTOR_LSB** 0/∗∗diode - ideality factor(eta)∗/

GLOBAL CONFIGURATION CONSTANTS

- #define **REJECTION__50_60_HZ** (uint8_t) 0x0
- #define **REJECTION__60_HZ** (uint8_t) 0x1
- #define **REJECTION__50_HZ** (uint8_t) 0x2
- #define **TEMP_UNIT__C** (uint8_t) 0x0
- #define **TEMP_UNIT__F** (uint8_t) 0x4
- #define **ENABLE_KELVIN_3_WIRE_RTD_MODE** (uint8_t) 0x10
- #define **ENABLE_KELVIN_2_WIRE_RTD_MODE** (uint8_t) 0x20
- #define **ENABLE_KELVIN_DIFFERENTIAL_THERMISTOR_MODE** (uint8_t) 0x40
- #define **DISABLE_MINUS_999** (uint8_t) 0x80

STATUS BYTE CONSTANTS

- #define **SENSOR_HARD_FAILURE** (uint8_t) 0x80
- #define **ADC_HARD_FAILURE** (uint8_t) 0x40
- #define **CJ_HARD_FAILURE** (uint8_t) 0x20
- #define **CJ_SOFT_FAILURE** (uint8_t) 0x10
- #define **SENSOR_ABOVE** (uint8_t) 0x8
- #define **SENSOR_BELOW** (uint8_t) 0x4
- #define **ADC_RANGE_ERROR** (uint8_t) 0x2
- #define **VALID** (uint8_t) 0x1

ADDRESS BASE

- #define **COMMAND_STATUS_REGISTER** (uint16_t) 0x0000
- #define **CH_ADDRESS_BASE** (uint16_t) 0x0200
- #define **VOUT_CH_BASE** (uint16_t) 0x0060
- #define **READ_CH_BASE** (uint16_t) 0x0010
- #define **CONVERSION_RESULT_MEMORY_BASE** (uint16_t) 0x0010

DATA to be sent in the MOSI line of MCU

- #define **WRITE_TO_RAM** (uint8_t) 0x02
- #define **READ_FROM_RAM** (uint8_t) 0x03
- #define **CONVERSION_CONTROL_BYTE** (uint8_t) 0x80

OUTPUT TYPE

- #define **VOLTAGE** (uint8_t) 0x01

- #define **TEMPERATURE** (uint8_t) 0x02
- #define **CODE** (uint8_t) 0x03

LTC DSPI EDMA Peripheral

- #define **LTC_DSPI_MASTER_BASEADDR** SPI1
- #define **LTC_DSPI_MASTER_DMA_MUX_BASE** DMAMUX_BASE
- #define **LTC_DSPI_MASTER_DMA_BASE** DMA_BASE
- #define **LTC_DSPI_MASTER_DMA_RX_REQUEST_SOURCE** kDmaRequestMux0SPI1
- #define **LTC_DSPI_MASTER_CLK_SRC** DSPI1_CLK_SRC
- #define **LTC_DSPI_MASTER_CLK_FREQ** CLOCK_GetFreq(DSPI1_CLK_SRC)
- #define **LTC_DSPI_MASTER_PCS_FOR_INIT** kDSPI_Pcs0
- #define **LTC_DSPI_MASTER_PCS_FOR_TRANSFER** kDSPI_MasterPcs0
- #define **LTC_DSPI_MASTER_DMA_MUX_BASEADDR** ((DMAMUX_Type ∗)(LTC_DSPI_MASTER_↩ DMA_MUX_BASE))
- #define **LTC_DSPI_MASTER_DMA_BASEADDR** ((DMA_Type ∗)(LTC_DSPI_MASTER_DMA_BASE))
- #define **LTC_TRANSFER_SIZE** 8U /∗ Transfer dataSize ∗/
- #define **LTC_TRANSFER_BAUDRATE** 2000000U /∗ Transfer baudrate - 2M ∗/

ADS DSPI EDMA Peripheral

- #define **NUM_CHANNEL** 8
- #define **ADS_ADC_RANGE** 10.0F
- #define **ADS_DSPI_MASTER_BASEADDR** SPI0
- #define **ADS_DSPI_MASTER_DMA_MUX_BASE** DMAMUX_BASE
- #define **ADS_DSPI_MASTER_DMA_BASE** DMA_BASE
- #define **ADS_DSPI_MASTER_DMA_RX_REQUEST_SOURCE** kDmaRequestMux0SPI0Rx
- #define **ADS_DSPI_MASTER_DMA_TX_REQUEST_SOURCE** kDmaRequestMux0SPI0Tx
- #define **ADS_MASTER_CLK_SRC** DSPI0_CLK_SRC
- #define **ADS_MASTER_CLK_FREQ** 20000000
- #define **ADS_DSPI_MASTER_PCS_FOR_INIT** kDSPI_Pcs0
- #define **ADS_DSPI_MASTER_PCS_FOR_TRANSFER** kDSPI_MasterPcs0
- #define **ADS_TRANSFER_SIZE** 2U∗NUM_CHANNEL /∗ Transfer dataSize ∗/
- #define **ADS_TRANSFER_BAUDRATE** 20000000U /∗ Transfer baudrate - 20M ∗/
- #define **ADS_DSPI_MASTER_DMA_MUX_BASEADDR** ((DMAMUX_Type ∗)(ADS_DSPI_MASTER_↩ DMA_MUX_BASE))
- #define **ADS_DSPI_MASTER_DMA_BASEADDR** ((DMA_Type ∗)(ADS_DSPI_MASTER_DMA_BASE))

On-board Comparator for PTC-150

- #define **CMP_BASE** CMP0
- #define **CMP_USER_CHANNEL** 1U
- #define **CMP_DAC_CHANNEL** 7U
- #define **CMP_THRESHOLD** 31U

Inbuilt ADC & PT-1000 Sensor parameters

- #define **PT_1000_ADC16_BASE** ADC0
- #define **PT_1000_ADC16_CHANNEL_GROUP** 0U
- #define **PT_1000_ADC16_USER_CHANNEL** 8U /∗ PTB0, ADC0_SE8 ∗/
- #define **FSL_FEATURE_ADC16_MAX_RESOLUTION** (16)
- #define **ALPHA** 0.00385F
- #define **PT_1000_A** 3.908300E-3F
- #define **PT_1000_B** -5.775E-7
- #define **PT_1000_C** –4.183000E-12
- #define **POTENTIAL_DIVIDER_INPUT** 3.0F
- #define **RESISTANCE_0_DEG** 1000.0F
- #define **POTENTIAL_DIVIDER_R1** 6800.0F

Functions

- void ReadFromEncoder (void)

  Function definition to read out of the FTM1 registers for Quadrature decoding.
- void PrintDebugInfo (void)

  Function definition to print debug data on the Serial terminal over UART @ 115200 bps baudrate.
- void DrivePOC_Comm_Handler_Init (void)

  Function definition to initialize the Communication Handler of the Drive POC controller for ADC and PWM.
- void PWM_Init (void)

  Initialize PWM Signals.
- void Comparator_Init (void)

  Initialize Comparator.
- void DrivePOC_Comm_Handler_PWMDis (void)

  Function definition to disable PWM.
- void DrivePOC_UpdateDutyCyc (GMCLIB_3COOR_T_FLT dutyCycle)

  Update duty cycle for PWM.
- void LTC_DSPI_MasterUserCallback (SPI_Type ∗base, dspi_master_edma_handle_t ∗handle, status_t status, void ∗userData)

  Function definition used a callback to check if the EDMA Transfer is successful or not for LTC Board.
- uint16_t ltc_get_start_address (uint16_t base_address, uint8_t channel_number)

  Function definition to get the start address corresponding to the channel number across which sensor is connected.
- uint32_t ltc_transfer_four_bytes (uint8_t ram_read_or_write, uint16_t start_address, uint32_t input_data)

  Function definiton to transfer 4 bytes.
- uint8_t ∗ ltc_spi_transfer_block (uint8_t TRANSFER_SIZEE, uint8_t ∗ttxx, uint8_t ∗rrxx)

  Function definition for SPI Transfer @detail The communication is Half Duplex Mode.
- void **ltc_configure_channels** (uint8_t channel_number, uint32_t channel_assignment_data)
- void delay (int delay_in_ms)

  Function which creates a delay in milli second @detail uses NOP - No Operation inside a for loop to create delays.
- uint8_t ltc_transfer_byte (uint8_t ram_read_or_write, uint16_t start_address, uint8_t input_data)

  Function definition for transfer of a single byte data.
- float_t ltc_measure_channel (uint8_t channel_number, uint8_t channel_output)

  Function definition to measure the sensor output.
- void ltc_convert_channel (uint8_t channel_number)

  Function definition which performs the Initiate conversion action.
- void ltc_wait_for_process_to_finish (void)

  Function definition to check if the value 0x40 is being returned by the LTC Board in MOSI line of the LTC Board(MISO Line of the MCU) which in turn indicates that the transfer of Initiate Conversion Command is a success.
- uint32_t ltc_get_result (uint8_t channel_number, uint8_t channel_output)

  Function which receives the Temperature or the Voltage value measured in the MISO line of the MCU.
- void ltc_print_conversion_result (uint32_t raw_conversion_result, uint8_t channel_output)

  Function definition to print the output in the console window.
- void ltc_read_voltage_or_resistance_results (uint8_t channel_number)

  Function definition to calculate if the value needed is in terms of Voltage or Resistance instead of Temperature.
- void ltc_print_fault_data (uint8_t fault_byte)

  Function to print the Fault byte in the console window indicating the kind of fault as per the Datasheet.
- void ltc_spi_edma_init (void)

  Initializes the DSPI & EDMA Peripheral.
- void comparator_init (void)

  Initializes the Comparator Peripherals for PTC-150.
- void InbuiltADC_Init (void)

  Function definition to initialize the Inbuilt ADC.
- float_t Collect_Data_from_PT_1000 (void)

Function definition to collect the Temperature value.

- bool_t Measure_from_PTC_150 (void)

  Function definition to measure from PTC-150.

- void ads_spi_edma_init (void)

  Initializes SPI0 instance for Communication with the ADS8588 board.

- uint8_t * ads_spi_transfer_block (uint8_t *rrxx)

  Used for receiving the current and voltage value from the ADS8588 board using SPI Protocol.

- void ADS_DSPI_MasterUserCallback (SPI_Type *base, dspi_master_edma_handle_t *handle, status_t status, void *userData)

  Function definition used a callback to check if the EDMA Transfer is successful or not for ADS Board.

- void **DAC_init** (void)

  Function definition for DAC.

- void **DAC_Update** (void)

Variables

- mcs_acim_open_loop_str s_open_loop

8.4.1 Detailed Description

: Communication Handler functions providing for all interfacing between NXP to other sensors/systems as well PWM Generation

Author

: Sreedhar, Sangeerth @company : Agnikul Cosmos Private Limited

Attention

8.5 DrivePOC_CommHandler.h

Go to the documentation of this file.

```
1
24 /*******************************************************************************
25  * Header inclusions
26  ******************************************************************************/
27 #include "fsl_dspi_edma.h"
28 #include "fsl_dspi.h"
29 #include "fsl_edma.h"
30 #include "fsl_dmamux.h"
31 #include "fsl_cmp.h"
32 #include "DrivePOC_Common_Header.h"
33
34 extern mcs_acim_open_loop_str s_open_loop;
35
36
37
       //*****************************************************************************************************************
38 // -- SENSOR TYPES --
39
       //*****************************************************************************************************************
42 #define SENSOR_TYPE_LSB 27
43 #define SENSOR_TYPE__RTD_PT_10 (uint32_t) 0xA « SENSOR_TYPE_LSB
44 #define SENSOR_TYPE__RTD_PT_50 (uint32_t) 0xB « SENSOR_TYPE_LSB
45 #define SENSOR_TYPE__RTD_PT_100 (uint32_t) 0xC « SENSOR_TYPE_LSB
46 #define SENSOR_TYPE__RTD_PT_200 (uint32_t) 0xD « SENSOR_TYPE_LSB
47 #define SENSOR_TYPE__RTD_PT_500 (uint32_t) 0xE « SENSOR_TYPE_LSB
48 #define SENSOR_TYPE__RTD_PT_1000 (uint32_t) 0xF « SENSOR_TYPE_LSB
49 #define SENSOR_TYPE__RTD_PT_1000_375 (uint32_t) 0x10 « SENSOR_TYPE_LSB
50 #define SENSOR_TYPE__RTD_NI_120 (uint32_t) 0x11 « SENSOR_TYPE_LSB
51 #define SENSOR_TYPE__RTD_CUSTOM (uint32_t) 0x12 « SENSOR_TYPE_LSB
```

```
53
54
57 #define SENSOR_TYPE__SENSE_RESISTOR (uint32_t) 0x1D « SENSOR_TYPE_LSB
58 #define SENSOR_TYPE__NONE (uint32_t) 0x0 « SENSOR_TYPE_LSB
59 #define SENSOR_TYPE__ACTIVE_ANALOG (uint32_t) 0x1F « SENSOR_TYPE_LSB
61
64 #define SENSOR_TYPE__DIRECT_ADC (uint32_t) 0x1E « SENSOR_TYPE_LSB
66
69 #define SENSOR_TYPE__THERMISTOR_44004_2P252K_25C (uint32_t) 0x13 « SENSOR_TYPE_LSB
70 #define SENSOR_TYPE__THERMISTOR_44005_3K_25C (uint32_t) 0x14 « SENSOR_TYPE_LSB
71 #define SENSOR_TYPE__THERMISTOR_44007_5K_25C (uint32_t) 0x15 « SENSOR_TYPE_LSB
72 #define SENSOR_TYPE__THERMISTOR_44006_10K_25C (uint32_t) 0x16 « SENSOR_TYPE_LSB
73 #define SENSOR_TYPE__THERMISTOR_44008_30K_25C (uint32_t) 0x17 « SENSOR_TYPE_LSB
74 #define SENSOR_TYPE__THERMISTOR_YSI_400_2P252K_25C (uint32_t) 0x18 « SENSOR_TYPE_LSB
75 #define SENSOR_TYPE__THERMISTOR_1003K_1K_25C (uint32_t) 0x19 « SENSOR_TYPE_LSB
76 #define SENSOR_TYPE__THERMISTOR_CUSTOM_STEINHART_HART (uint32_t) 0x1A « SENSOR_TYPE_LSB
77 #define SENSOR_TYPE__THERMISTOR_CUSTOM_TABLE (uint32_t) 0x1B « SENSOR_TYPE_LSB
79
82 #define SENSOR_TYPE__TYPE_J_THERMOCOUPLE (uint32_t) 0x1 « SENSOR_TYPE_LSB
83 #define SENSOR_TYPE__TYPE_K_THERMOCOUPLE (uint32_t) 0x2 « SENSOR_TYPE_LSB
84 #define SENSOR_TYPE__TYPE_E_THERMOCOUPLE (uint32_t) 0x3 « SENSOR_TYPE_LSB
85 #define SENSOR_TYPE__TYPE_N_THERMOCOUPLE (uint32_t) 0x4 « SENSOR_TYPE_LSB
86 #define SENSOR_TYPE__TYPE_R_THERMOCOUPLE (uint32_t) 0x5 « SENSOR_TYPE_LSB
87 #define SENSOR_TYPE__TYPE_S_THERMOCOUPLE (uint32_t) 0x6 « SENSOR_TYPE_LSB
88 #define SENSOR_TYPE__TYPE_T_THERMOCOUPLE (uint32_t) 0x7 « SENSOR_TYPE_LSB
89 #define SENSOR_TYPE__TYPE_B_THERMOCOUPLE (uint32_t) 0x8 « SENSOR_TYPE_LSB
90 #define SENSOR_TYPE__CUSTOM_THERMOCOUPLE (uint32_t) 0x9 « SENSOR_TYPE_LSB
92
95 #define SENSOR_TYPE__OFF_CHIP_DIODE (uint32_t) 0x1C « SENSOR_TYPE_LSB
97
100 #define RTD_RSENSE_CHANNEL_LSB 22
101 #define RTD_RSENSE_CHANNEL__NONE (uint32_t) 0x0 « RTD_RSENSE_CHANNEL_LSB
102 #define RTD_RSENSE_CHANNEL__1 (uint32_t) 0x1 « RTD_RSENSE_CHANNEL_LSB
103 #define RTD_RSENSE_CHANNEL__2 (uint32_t) 0x2 « RTD_RSENSE_CHANNEL_LSB
104 #define RTD_RSENSE_CHANNEL__3 (uint32_t) 0x3 « RTD_RSENSE_CHANNEL_LSB
105 #define RTD_RSENSE_CHANNEL__4 (uint32_t) 0x4 « RTD_RSENSE_CHANNEL_LSB
106 #define RTD_RSENSE_CHANNEL__5 (uint32_t) 0x5 « RTD_RSENSE_CHANNEL_LSB
107 #define RTD_RSENSE_CHANNEL__6 (uint32_t) 0x6 « RTD_RSENSE_CHANNEL_LSB
108 #define RTD_RSENSE_CHANNEL__7 (uint32_t) 0x7 « RTD_RSENSE_CHANNEL_LSB
109 #define RTD_RSENSE_CHANNEL__8 (uint32_t) 0x8 « RTD_RSENSE_CHANNEL_LSB
110 #define RTD_RSENSE_CHANNEL__9 (uint32_t) 0x9 « RTD_RSENSE_CHANNEL_LSB
111 #define RTD_RSENSE_CHANNEL__10 (uint32_t) 0xA « RTD_RSENSE_CHANNEL_LSB
113
116 #define RTD_NUM_WIRES_LSB 20
117 #define RTD_NUM_WIRES__2_WIRE (uint32_t) 0x0 « RTD_NUM_WIRES_LSB
118 #define RTD_NUM_WIRES__3_WIRE (uint32_t) 0x1 « RTD_NUM_WIRES_LSB
119 #define RTD_NUM_WIRES__4_WIRE (uint32_t) 0x2 « RTD_NUM_WIRES_LSB
120 #define RTD_NUM_WIRES__4_WIRE_KELVIN_RSENSE (uint32_t) 0x3 « RTD_NUM_WIRES_LSB
122
125 #define RTD_EXCITATION_MODE_LSB 18
126 #define RTD_EXCITATION_MODE__NO_ROTATION_NO_SHARING (uint32_t) 0x0 « RTD_EXCITATION_MODE_LSB
127 #define RTD_EXCITATION_MODE__NO_ROTATION_SHARING (uint32_t) 0x1 « RTD_EXCITATION_MODE_LSB
128 #define RTD_EXCITATION_MODE__ROTATION_SHARING (uint32_t) 0x2 « RTD_EXCITATION_MODE_LSB
130
133 #define RTD_EXCITATION_CURRENT_LSB 14
134 #define RTD_EXCITATION_CURRENT__EXTERNAL (uint32_t) 0x0 « RTD_EXCITATION_CURRENT_LSB
135 #define RTD_EXCITATION_CURRENT__5UA (uint32_t) 0x1 « RTD_EXCITATION_CURRENT_LSB
136 #define RTD_EXCITATION_CURRENT__10UA (uint32_t) 0x2 « RTD_EXCITATION_CURRENT_LSB
137 #define RTD_EXCITATION_CURRENT__25UA (uint32_t) 0x3 « RTD_EXCITATION_CURRENT_LSB
138 #define RTD_EXCITATION_CURRENT__50UA (uint32_t) 0x4 « RTD_EXCITATION_CURRENT_LSB
139 #define RTD_EXCITATION_CURRENT__100UA (uint32_t) 0x5 « RTD_EXCITATION_CURRENT_LSB
140 #define RTD_EXCITATION_CURRENT__250UA (uint32_t) 0x6 « RTD_EXCITATION_CURRENT_LSB
141 #define RTD_EXCITATION_CURRENT__500UA (uint32_t) 0x7 « RTD_EXCITATION_CURRENT_LSB
142 #define RTD_EXCITATION_CURRENT__1MA (uint32_t) 0x8 « RTD_EXCITATION_CURRENT_LSB
144
147 #define RTD_STANDARD_LSB 12
148 #define RTD_STANDARD__EUROPEAN (uint32_t) 0x0 « RTD_STANDARD_LSB
149 #define RTD_STANDARD__AMERICAN (uint32_t) 0x1 « RTD_STANDARD_LSB
150 #define RTD_STANDARD__JAPANESE (uint32_t) 0x2 « RTD_STANDARD_LSB
151 #define RTD_STANDARD__ITS_90 (uint32_t) 0x3 « RTD_STANDARD_LSB
153
156 #define RTD_CUSTOM_ADDRESS_LSB 6/*rtd - custom address*/
157 #define RTD_CUSTOM_LENGTH_1_LSB 0/*rtd - custom length-1*/
158 #define RTD_CUSTOM_VALUES_LSB 31/*rtd - custom values*/
160
163 #define SENSE_RESISTOR_VALUE_LSB 0
165
168 #define ACTIVE_ANALOG_DIFFERENTIAL_LSB 26
169 #define ACTIVE_ANALOG_DIFFERENTIAL (uint32_t) 0x0 « ACTIVE_ANALOG_DIFFERENTIAL_LSB
170 #define ACTIVE_ANALOG_SINGLE_ENDED (uint32_t) 0x1 « ACTIVE_ANALOG_DIFFERENTIAL_LSB
171 #define ACTIVE_ANALOG_CUSTOM_ADDRESS_LSB 6  /*active analog - custom address*/
172 #define ACTIVE_ANALOG_CUSTOM_LENGTH_1_LSB 0 /*active analog - custom length-1*/
173 #define ACTIVE_ANALOG_CUSTOM_VALUES_LSB 31  /*active analog - custom values*/
175
176
179 #define DIRECT_ADC_DIFFERENTIAL_LSB 26
180 #define DIRECT_ADC_DIFFERENTIAL (uint32_t) 0x0 « DIRECT_ADC_DIFFERENTIAL_LSB
```

```
181 #define DIRECT_ADC_SINGLE_ENDED (uint32_t) 0x1 « DIRECT_ADC_DIFFERENTIAL_LSB
183
186 #define DIRECT_ADC_CUSTOM_LSB 25
187 #define DIRECT_ADC_CUSTOM__NO (uint32_t) 0x0 « DIRECT_ADC_CUSTOM_LSB
188 #define DIRECT_ADC_CUSTOM__YES (uint32_t) 0x1 « DIRECT_ADC_CUSTOM_LSB
189 #define DIRECT_ADC_CUSTOM_ADDRESS_LSB 6 /*Direct ADC - custom address*/
190 #define DIRECT_ADC_CUSTOM_LENGTH_1_LSB 0 /*Direct ADC - custom length-1*/
191 #define DIRECT_ADC_CUSTOM_VALUES_LSB 31 /*Direct ADC - custom values*/
193
194
197 #define THERMISTOR_RSENSE_CHANNEL_LSB 22
198 #define THERMISTOR_RSENSE_CHANNEL__NONE (uint32_t) 0x0 « THERMISTOR_RSENSE_CHANNEL_LSB
199 #define THERMISTOR_RSENSE_CHANNEL__1 (uint32_t) 0x1 « THERMISTOR_RSENSE_CHANNEL_LSB
200 #define THERMISTOR_RSENSE_CHANNEL__2 (uint32_t) 0x2 « THERMISTOR_RSENSE_CHANNEL_LSB
201 #define THERMISTOR_RSENSE_CHANNEL__3 (uint32_t) 0x3 « THERMISTOR_RSENSE_CHANNEL_LSB
202 #define THERMISTOR_RSENSE_CHANNEL__4 (uint32_t) 0x4 « THERMISTOR_RSENSE_CHANNEL_LSB
203 #define THERMISTOR_RSENSE_CHANNEL__5 (uint32_t) 0x5 « THERMISTOR_RSENSE_CHANNEL_LSB
204 #define THERMISTOR_RSENSE_CHANNEL__6 (uint32_t) 0x6 « THERMISTOR_RSENSE_CHANNEL_LSB
205 #define THERMISTOR_RSENSE_CHANNEL__7 (uint32_t) 0x7 « THERMISTOR_RSENSE_CHANNEL_LSB
206 #define THERMISTOR_RSENSE_CHANNEL__8 (uint32_t) 0x8 « THERMISTOR_RSENSE_CHANNEL_LSB
207 #define THERMISTOR_RSENSE_CHANNEL__9 (uint32_t) 0x9 « THERMISTOR_RSENSE_CHANNEL_LSB
208 #define THERMISTOR_RSENSE_CHANNEL__10 (uint32_t) 0xA « THERMISTOR_RSENSE_CHANNEL_LSB
210
213 #define THERMISTOR_DIFFERENTIAL_LSB 21
214 #define THERMISTOR_DIFFERENTIAL (uint32_t) 0x0 « THERMISTOR_DIFFERENTIAL_LSB
215 #define THERMISTOR_SINGLE_ENDED (uint32_t) 0x1 « THERMISTOR_DIFFERENTIAL_LSB
217
220 #define THERMISTOR_EXCITATION_MODE_LSB 19
221 #define THERMISTOR_EXCITATION_MODE__NO_SHARING_NO_ROTATION (uint32_t) 0x0 «
       THERMISTOR_EXCITATION_MODE_LSB
222 #define THERMISTOR_EXCITATION_MODE__SHARING_ROTATION (uint32_t) 0x1 « THERMISTOR_EXCITATION_MODE_LSB
223 #define THERMISTOR_EXCITATION_MODE__SHARING_NO_ROTATION (uint32_t) 0x2 « THERMISTOR_EXCITATION_MODE_LSB
225
228 #define THERMISTOR_EXCITATION_CURRENT_LSB 15
229 #define THERMISTOR_EXCITATION_CURRENT__INVALID (uint32_t) 0x0 « THERMISTOR_EXCITATION_CURRENT_LSB
230 #define THERMISTOR_EXCITATION_CURRENT__250NA (uint32_t) 0x1 « THERMISTOR_EXCITATION_CURRENT_LSB
231 #define THERMISTOR_EXCITATION_CURRENT__500NA (uint32_t) 0x2 « THERMISTOR_EXCITATION_CURRENT_LSB
232 #define THERMISTOR_EXCITATION_CURRENT__1UA (uint32_t) 0x3 « THERMISTOR_EXCITATION_CURRENT_LSB
233 #define THERMISTOR_EXCITATION_CURRENT__5UA (uint32_t) 0x4 « THERMISTOR_EXCITATION_CURRENT_LSB
234 #define THERMISTOR_EXCITATION_CURRENT__10UA (uint32_t) 0x5 « THERMISTOR_EXCITATION_CURRENT_LSB
235 #define THERMISTOR_EXCITATION_CURRENT__25UA (uint32_t) 0x6 « THERMISTOR_EXCITATION_CURRENT_LSB
236 #define THERMISTOR_EXCITATION_CURRENT__50UA (uint32_t) 0x7 « THERMISTOR_EXCITATION_CURRENT_LSB
237 #define THERMISTOR_EXCITATION_CURRENT__100UA (uint32_t) 0x8 « THERMISTOR_EXCITATION_CURRENT_LSB
238 #define THERMISTOR_EXCITATION_CURRENT__250UA (uint32_t) 0x9 « THERMISTOR_EXCITATION_CURRENT_LSB
239 #define THERMISTOR_EXCITATION_CURRENT__500UA (uint32_t) 0xA « THERMISTOR_EXCITATION_CURRENT_LSB
240 #define THERMISTOR_EXCITATION_CURRENT__1MA (uint32_t) 0xB « THERMISTOR_EXCITATION_CURRENT_LSB
241 #define THERMISTOR_EXCITATION_CURRENT__AUTORANGE (uint32_t) 0xC « THERMISTOR_EXCITATION_CURRENT_LSB
242 #define THERMISTOR_EXCITATION_CURRENT__INVALID_ (uint32_t) 0xD « THERMISTOR_EXCITATION_CURRENT_LSB
243 #define THERMISTOR_EXCITATION_CURRENT__INVALID__ (uint32_t) 0xE « THERMISTOR_EXCITATION_CURRENT_LSB
244 #define THERMISTOR_EXCITATION_CURRENT__EXTERNAL (uint32_t) 0xF « THERMISTOR_EXCITATION_CURRENT_LSB
246
249 #define THERMISTOR_CUSTOM_ADDRESS_LSB 6 /* thermistor - custom address*/
250 #define THERMISTOR_CUSTOM_LENGTH_1_LSB 0 /*thermistor - custom length-1*/
251 #define THERMISTOR_CUSTOM_VALUES_LSB 31  /*thermistor - custom values*/
253
254
255
256
257
259 #define TC_COLD_JUNCTION_CH_LSB 22
260 #define TC_COLD_JUNCTION_CH__NONE (uint32_t) 0x0 « TC_COLD_JUNCTION_CH_LSB
261 #define TC_COLD_JUNCTION_CH__1 (uint32_t) 0x1 « TC_COLD_JUNCTION_CH_LSB
262 #define TC_COLD_JUNCTION_CH__2 (uint32_t) 0x2 « TC_COLD_JUNCTION_CH_LSB
263 #define TC_COLD_JUNCTION_CH__3 (uint32_t) 0x3 « TC_COLD_JUNCTION_CH_LSB
264 #define TC_COLD_JUNCTION_CH__4 (uint32_t) 0x4 « TC_COLD_JUNCTION_CH_LSB
265 #define TC_COLD_JUNCTION_CH__5 (uint32_t) 0x5 « TC_COLD_JUNCTION_CH_LSB
266 #define TC_COLD_JUNCTION_CH__6 (uint32_t) 0x6 « TC_COLD_JUNCTION_CH_LSB
267 #define TC_COLD_JUNCTION_CH__7 (uint32_t) 0x7 « TC_COLD_JUNCTION_CH_LSB
268 #define TC_COLD_JUNCTION_CH__8 (uint32_t) 0x8 « TC_COLD_JUNCTION_CH_LSB
269 #define TC_COLD_JUNCTION_CH__9 (uint32_t) 0x9 « TC_COLD_JUNCTION_CH_LSB
270 #define TC_COLD_JUNCTION_CH__10 (uint32_t) 0xA « TC_COLD_JUNCTION_CH_LSB
272
275 #define TC_DIFFERENTIAL_LSB 21
276 #define TC_DIFFERENTIAL (uint32_t) 0x0 « TC_DIFFERENTIAL_LSB
277 #define TC_SINGLE_ENDED (uint32_t) 0x1 « TC_DIFFERENTIAL_LSB
279
282 #define TC_OPEN_CKT_DETECT_LSB 20
283 #define TC_OPEN_CKT_DETECT__NO (uint32_t) 0x0 « TC_OPEN_CKT_DETECT_LSB
284 #define TC_OPEN_CKT_DETECT__YES (uint32_t) 0x1 « TC_OPEN_CKT_DETECT_LSB
286
289 #define TC_OPEN_CKT_DETECT_CURRENT_LSB 18
290 #define TC_OPEN_CKT_DETECT_CURRENT__10UA (uint32_t) 0x0 « TC_OPEN_CKT_DETECT_CURRENT_LSB
291 #define TC_OPEN_CKT_DETECT_CURRENT__100UA (uint32_t) 0x1 « TC_OPEN_CKT_DETECT_CURRENT_LSB
292 #define TC_OPEN_CKT_DETECT_CURRENT__500UA (uint32_t) 0x2 « TC_OPEN_CKT_DETECT_CURRENT_LSB
293 #define TC_OPEN_CKT_DETECT_CURRENT__1MA (uint32_t) 0x3 « TC_OPEN_CKT_DETECT_CURRENT_LSB
294 #define TC_CUSTOM_ADDRESS_LSB 6 /* tc - custom address*/
295 #define TC_CUSTOM_LENGTH_1_LSB 0/* tc - custom length-1*/
```

```
296 #define TC_CUSTOM_VALUES_LSB 31 /*tc - custom values*/
298
301 #define DIODE_DIFFERENTIAL_LSB 26
302 #define DIODE_DIFFERENTIAL (uint32_t) 0x0 « DIODE_DIFFERENTIAL_LSB
303 #define DIODE_SINGLE_ENDED (uint32_t) 0x1 « DIODE_DIFFERENTIAL_LSB
307 #define DIODE_NUM_READINGS_LSB 25
308 #define DIODE_NUM_READINGS__2 (uint32_t) 0x0 « DIODE_NUM_READINGS_LSB
309 #define DIODE_NUM_READINGS__3 (uint32_t) 0x1 « DIODE_NUM_READINGS_LSB
311
314 #define DIODE_AVERAGING_ON_LSB 24
315 #define DIODE_AVERAGING_OFF (uint32_t) 0x0 « DIODE_AVERAGING_ON_LSB
316 #define DIODE_AVERAGING_ON (uint32_t) 0x1 « DIODE_AVERAGING_ON_LSB
318
321 #define DIODE_CURRENT_LSB 22
322 #define DIODE_CURRENT__10UA_40UA_80UA (uint32_t) 0x0 « DIODE_CURRENT_LSB
323 #define DIODE_CURRENT__20UA_80UA_160UA (uint32_t) 0x1 « DIODE_CURRENT_LSB
324 #define DIODE_CURRENT__40UA_160UA_320UA (uint32_t) 0x2 « DIODE_CURRENT_LSB
325 #define DIODE_CURRENT__80UA_320UA_640UA (uint32_t) 0x3 « DIODE_CURRENT_LSB
326 #define DIODE_IDEALITY_FACTOR_LSB 0
328
331 #define REJECTION__50_60_HZ (uint8_t) 0x0
332 #define REJECTION__60_HZ    (uint8_t) 0x1
333 #define REJECTION__50_HZ    (uint8_t) 0x2
334 #define TEMP_UNIT__C        (uint8_t) 0x0
335 #define TEMP_UNIT__F        (uint8_t) 0x4
336 #define ENABLE_KELVIN_3_WIRE_RTD_MODE               (uint8_t) 0x10
337 #define ENABLE_KELVIN_2_WIRE_RTD_MODE               (uint8_t) 0x20
338 #define ENABLE_KELVIN_DIFFERENTIAL_THERMISTOR_MODE  (uint8_t) 0x40
339 #define DISABLE_MINUS_999                           (uint8_t) 0x80
341
344 #define SENSOR_HARD_FAILURE (uint8_t) 0x80
345 #define ADC_HARD_FAILURE    (uint8_t) 0x40
346 #define CJ_HARD_FAILURE     (uint8_t) 0x20
347 #define CJ_SOFT_FAILURE     (uint8_t) 0x10
348 #define SENSOR_ABOVE        (uint8_t) 0x8
349 #define SENSOR_BELOW        (uint8_t) 0x4
350 #define ADC_RANGE_ERROR     (uint8_t) 0x2
351 #define VALID               (uint8_t) 0x1
353
356 #define COMMAND_STATUS_REGISTER          (uint16_t) 0x0000
357 #define CH_ADDRESS_BASE                  (uint16_t) 0x0200
358 #define VOUT_CH_BASE                     (uint16_t) 0x0060
359 #define READ_CH_BASE                     (uint16_t) 0x0010
360 #define CONVERSION_RESULT_MEMORY_BASE    (uint16_t) 0x0010
362
365 #define WRITE_TO_RAM            (uint8_t) 0x02
366 #define READ_FROM_RAM           (uint8_t) 0x03
367 #define CONVERSION_CONTROL_BYTE (uint8_t) 0x80
369
372 #define VOLTAGE                 (uint8_t) 0x01
373 #define TEMPERATURE             (uint8_t) 0x02
374 #define CODE                    (uint8_t) 0x03
376
377
378
379
382 #define LTC_DSPI_MASTER_BASEADDR SPI1
383 #define LTC_DSPI_MASTER_DMA_MUX_BASE DMAMUX_BASE
384 #define LTC_DSPI_MASTER_DMA_BASE DMA_BASE
385 #define LTC_DSPI_MASTER_DMA_RX_REQUEST_SOURCE kDmaRequestMux0SPI1
386 #define LTC_DSPI_MASTER_CLK_SRC DSPI1_CLK_SRC
387 #define LTC_DSPI_MASTER_CLK_FREQ CLOCK_GetFreq(DSPI1_CLK_SRC)
388 #define LTC_DSPI_MASTER_PCS_FOR_INIT kDSPI_Pcs0
389 #define LTC_DSPI_MASTER_PCS_FOR_TRANSFER kDSPI_MasterPcs0
390 #define LTC_DSPI_MASTER_DMA_MUX_BASEADDR ((DMAMUX_Type *)(LTC_DSPI_MASTER_DMA_MUX_BASE))
391 #define LTC_DSPI_MASTER_DMA_BASEADDR ((DMA_Type *)(LTC_DSPI_MASTER_DMA_BASE))
392 #define LTC_TRANSFER_SIZE 8U        /* Transfer dataSize */
393 #define LTC_TRANSFER_BAUDRATE 2000000U /* Transfer baudrate - 2M */
395
396
399 #define NUM_CHANNEL 8
400 #define ADS_ADC_RANGE 10.0F
401 #define ADS_DSPI_MASTER_BASEADDR SPI0
402 #define ADS_DSPI_MASTER_DMA_MUX_BASE DMAMUX_BASE
403 #define ADS_DSPI_MASTER_DMA_BASE DMA_BASE
404 #define ADS_DSPI_MASTER_DMA_RX_REQUEST_SOURCE kDmaRequestMux0SPI0Rx
405 #define ADS_DSPI_MASTER_DMA_TX_REQUEST_SOURCE kDmaRequestMux0SPI0Tx
406 #define ADS_MASTER_CLK_SRC DSPI0_CLK_SRC
407 #define ADS_MASTER_CLK_FREQ 20000000
408 #define ADS_DSPI_MASTER_PCS_FOR_INIT kDSPI_Pcs0
409 #define ADS_DSPI_MASTER_PCS_FOR_TRANSFER kDSPI_MasterPcs0
410 #define ADS_TRANSFER_SIZE 2U*NUM_CHANNEL        /* Transfer dataSize */
411 #define ADS_TRANSFER_BAUDRATE 20000000U /* Transfer baudrate - 20M */
412 #define ADS_DSPI_MASTER_DMA_MUX_BASEADDR ((DMAMUX_Type *)(ADS_DSPI_MASTER_DMA_MUX_BASE))
413 #define ADS_DSPI_MASTER_DMA_BASEADDR ((DMA_Type *)(ADS_DSPI_MASTER_DMA_BASE))
414
417 #define CMP_BASE CMP0
```

```
418 #define CMP_USER_CHANNEL 1U
419 #define CMP_DAC_CHANNEL 7U
420 #define CMP_THRESHOLD 31U
422
425 #define PT_1000_ADC16_BASE          ADC0
426 #define PT_1000_ADC16_CHANNEL_GROUP 0U
427 #define PT_1000_ADC16_USER_CHANNEL  8U /* PTB0, ADC0_SE8 */
428 #define FSL_FEATURE_ADC16_MAX_RESOLUTION (16)
429 #define ALPHA 0.00385F
430 #define PT_1000_A    3.908300E-3F
431 #define PT_1000_B    -5.775E-7
432 #define PT_1000_C    -4.183000E-12
433 #define POTENTIAL_DIVIDER_INPUT 3.0F
434 #define RESISTANCE_0_DEG 1000.0F
435 #define POTENTIAL_DIVIDER_R1 6800.0F
437
438 /*******************************************************************************
439  * Function prototypes
440  ******************************************************************************/
441
442
449 void ReadFromEncoder(void);
450
459 void PrintDebugInfo(void);
460
469 void DrivePOC_Comm_Handler_Init(void);
470
476 void PWM_Init(void);
477
483 void Comparator_Init(void);
484
485
493 void DrivePOC_Comm_Handler_PWMDis(void);
494
502 void DrivePOC_UpdateDutyCyc(GMCLIB_3COOR_T_FLT dutyCycle);
503
514 void LTC_DSPI_MasterUserCallback(SPI_Type *base, dspi_master_edma_handle_t *handle, status_t status,
       void *userData);
515
523 uint16_t ltc_get_start_address(uint16_t base_address, uint8_t channel_number);
524
535 uint32_t ltc_transfer_four_bytes(uint8_t ram_read_or_write, uint16_t start_address, uint32_t
       input_data);
536
547 uint8_t *ltc_spi_transfer_block(uint8_t TRANSFER_SIZEE, uint8_t *ttxx, uint8_t *rrxx);
548
549
550 /*
551  * @brief configures channel
552  * @detail It is used in the Calibration phase of Motor state machine
553  * @param channel_number Channel number at which the sensor is connected
554  * @param channel_assignment_data Channel specific data
555  * @return None
556  */
557 void ltc_configure_channels(uint8_t channel_number,uint32_t channel_assignment_data);
558
565 void delay(int delay_in_ms);
566
574 uint8_t ltc_transfer_byte(uint8_t ram_read_or_write, uint16_t start_address, uint8_t input_data);
575
587 float_t ltc_measure_channel(uint8_t channel_number, uint8_t channel_output);
588
597 void ltc_convert_channel(uint8_t channel_number);
598
605 void ltc_wait_for_process_to_finish(void);
606
614 uint32_t ltc_get_result( uint8_t channel_number, uint8_t channel_output);
615
623 void ltc_print_conversion_result(uint32_t raw_conversion_result, uint8_t channel_output);
624
631 void ltc_read_voltage_or_resistance_results(uint8_t channel_number);
632
638 void ltc_print_fault_data(uint8_t fault_byte);
639
644 void ltc_spi_edma_init(void);
645
650 void comparator_init(void);
651
652
658 void InbuiltADC_Init(void);
659
665 float_t Collect_Data_from_PT_1000(void);
666
673 bool_t Measure_from_PTC_150(void);
674
675
681 void ads_spi_edma_init(void);
```

```
682
690 uint8_t *ads_spi_transfer_block(uint8_t *rrxx);
691
701 void ADS_DSPI_MasterUserCallback(SPI_Type *base, dspi_master_edma_handle_t *handle, status_t status,
       void *userData);
702
706 void DAC_init(void);
707
708 /*
709  *@brief Function defintion for updating DAC Value
710  * */
711 void DAC_Update(void);
712
713 //Closes the @defgroup block. Always kept last.
714
```

8.6 DrivePOC_Common_Header.h File Reference

: Common header containing structures and enumerations used across the entire project

```
#include "DrivePOC_Controller_NXP.h"
#include "mlib_FP.h"
#include "gflib_FP.h"
#include "gdflib_FP.h"
#include "gmclib_FP.h"
#include "amclib_FP.h"
```

Data Structures

- struct Main_SM_ControlSig
- struct Motor_SM_ControlSig
- struct mcs_acim_open_loop_str

Typedefs

- typedef enum Main_SM_States Main_SM_States
- typedef enum Motor_SM_States Motor_SM_States
- typedef struct Main_SM_ControlSig Main_SM_ControlSig
- typedef struct Motor_SM_ControlSig Motor_SM_ControlSig
- typedef struct mcs_acim_open_loop_str mcs_acim_open_loop_str

Enumerations

- enum Main_SM_States { MAIN_SM_INIT = 0 , MAIN_SM_FAULT = 1 , MAIN_SM_STOP = 2 , MAIN_SM_RUN = 3 }
- enum Motor_SM_States {
  MOTOR_SM_CALIB = 0 , MOTOR_SM_READY = 1 , MOTOR_SM_START = 2 , MOTOR_SM_SPIN = 3 , MOTOR_SM_DECELERATE =4 }

8.6.1 Detailed Description

: Common header containing structures and enumerations used across the entire project

Author

: Sreedhar, Sangeerth @company : Agnikul Cosmos Private Limited

Attention

8.6.2 Typedef Documentation

8.6.2.1 Main_SM_ControlSig

`typedef struct Main_SM_ControlSig Main_SM_ControlSig`
Main State machine's control signals for state transition

8.6.2.2 Main_SM_States

`typedef enum Main_SM_States Main_SM_States`
Motor control libraries headers Main state machine - States

8.6.2.3 mcs_acim_open_loop_str

`typedef struct mcs_acim_open_loop_str mcs_acim_open_loop_str`
Structure for V/F Control Open Loop

8.6.2.4 Motor_SM_ControlSig

`typedef struct Motor_SM_ControlSig Motor_SM_ControlSig`
Motor State machine's control signals for state transition

8.6.2.5 Motor_SM_States

`typedef enum Motor_SM_States Motor_SM_States`
Motor state machine - Run sub-states

8.6.3 Enumeration Type Documentation

8.6.3.1 Main_SM_States

`enum Main_SM_States`
Motor control libraries headers Main state machine - States

Enumerator

MAIN_SM_INIT	Initialization state of Main State machine
MAIN_SM_FAULT	Fault state of Main State machine
MAIN_SM_STOP	Stop state of Main State machine
MAIN_SM_RUN	Run state of Main State machine

8.6.3.2 Motor_SM_States

`enum Motor_SM_States`
Motor state machine - Run sub-states

Enumerator

MOTOR_SM_CALIB	Calibration state of Motor State machine

Enumerator

MOTOR_SM_READY	Ready state of Motor State machine
MOTOR_SM_START	Start up state of Motor State machine
MOTOR_SM_SPIN	Spin state of Motor State machine
MOTOR_SM_DECELERATE	Decelerate state of Motor State machine

8.7 DrivePOC_Common_Header.h

Go to the documentation of this file.

```
1
17 /*******************************************************************************
18  * Header inclusions
19  ******************************************************************************/
20 #include "DrivePOC_Controller_NXP.h"
21
23 #include "mlib_FP.h"
24 #include "gflib_FP.h"
25 #include "gdflib_FP.h"
26 #include "gmclib_FP.h"
27 #include "amclib_FP.h"
28
29
30 /*******************************************************************************
31  * Structures and Enumerations
32  ******************************************************************************/
36 typedef enum Main_SM_States
37 {
39     MAIN_SM_INIT = 0,
41     MAIN_SM_FAULT  = 1,
43     MAIN_SM_STOP  = 2,
45     MAIN_SM_RUN   = 3
46 } Main_SM_States;
47
51 typedef enum Motor_SM_States
52 {
54     MOTOR_SM_CALIB     = 0,
56     MOTOR_SM_READY     = 1,
58     MOTOR_SM_START       = 2,
60     MOTOR_SM_SPIN      = 3,
62     MOTOR_SM_DECELERATE =4
63 } Motor_SM_States;
64
68 typedef struct Main_SM_ControlSig
69 {
70     bool main_sm_ctrl_init_done;
71     bool main_sm_ctrl_fault;
72     bool main_sm_ctrl_stop;
73     bool main_sm_ctrl_run;
74 } Main_SM_ControlSig;
75
79 typedef struct Motor_SM_ControlSig
80 {
81     bool calib;
82     bool ready;
83     bool start_ok;
84     bool spin;
85 } Motor_SM_ControlSig;
86
90 typedef struct mcs_acim_open_loop_str{
91     GMCLIB_3COOR_T_F16 s_dutyabc_f16;
92     GMCLIB_3COOR_T_FLT s_dutyabc_flt;
93     GMCLIB_3COOR_T_FLT s_iabc;
94     GMCLIB_3COOR_T_FLT s_vabc;
95     float_t fltudcbus;
96     float_t fltidcbus;
97 }mcs_acim_open_loop_str;
```

8.8 DrivePOC_Control_Loop.c File Reference

: Functionalities pertaining to the V/F controls-Algorithm's state machines (Main & Motor)

```
#include "Drive_Parameters.h"
#include "stdbool.h"
#include "board.h"
```

```
#include "fsl_debug_console.h"
#include "fsl_cmp.h"
#include "fsl_gpio.h"
#include "DrivePOC_Controller_NXP.h"
#include "DrivePOC_CommHandler.h"
#include "DrivePOC_MemHandler.h"
#include "DrivePOC_Control_Loop.h"
#include "DrivePOC_FaultHandler.h"
```

Functions

- GMCLIB_3COOR_T_FLT Open_Loop_Control (void)

  Function definition corresponding to the Fast loop of the V/F control algorithm.
- void **Open_Loop_Control_Init** (void)
- GMCLIB_3COOR_T_FLT Open_Loop_MainStateMC (void)

  Function definition corresponding to the main state machine of V/F control algorithm.
- void System_FaultDiag (void)

  Function definition corresponding to the Fault diagnosis.
- void PTC_FaultDiag ()

  Function definition corresponding to fault indicated by PTC 150 Thermocouple.
- void Open_Loop_MotorStateMC (void)

  Function definition corresponding to the Motor state machine of V/F Control algorithm.
- void Motor_SM_Calibration (void)

  ADC Calibration phase.

Variables

- Main_SM_States main_sm_state
- Motor_SM_States motor_sm_state
- Main_SM_ControlSig main_ctrl_sig
- Motor_SM_ControlSig motor_ctrl_sig
- mcs_acim_open_loop_str s_open_loop

8.8.1 Detailed Description

: Functionalities pertaining to the V/F controls-Algorithm's state machines (Main & Motor)

Author

: Sreedhar, Sangeerth @company : Agnikul Cosmos Private Limited

Attention

8.8.2 Variable Documentation

8.8.2.1 main_ctrl_sig

```
Main_SM_ControlSig main_ctrl_sig
```

Main state machine - control signals for state transition

8.8.2.2 main_sm_state

`Main_SM_States main_sm_state`
Main state machine - States

8.8.2.3 motor_ctrl_sig

`Motor_SM_ControlSig motor_ctrl_sig`
Motor state machine - control signals for state transition

8.8.2.4 motor_sm_state

`Motor_SM_States motor_sm_state`
Motor state machine - Run sub-states

8.9 DrivePOC_Control_Loop.h File Reference

: Functionalities pertaining to the V/F controls-Algorithm's state machines (Main & Motor)

Functions

- void PTC_FaultDiag (void)

  Function definition corresponding to fault indicated by PTC 150 Thermocouple.
- void System_FaultDiag (void)

  Function definition corresponding to the Fault diagnosis.
- GMCLIB_3COOR_T_FLT Open_Loop_MainStateMC (void)

  Function definition corresponding to the main state machine of V/F control algorithm.
- void Open_Loop_MotorStateMC (void)

  Function definition corresponding to the Motor state machine of V/F Control algorithm.
- GMCLIB_3COOR_T_FLT Open_Loop_Control (void)

  Function definition corresponding to the Fast loop of the V/F control algorithm.
- void Get_Duty_Cycle (void)

  Function definition that gets the value of duty cycle from the duty cycle array.
- void Motor_SM_Calibration (void)

  ADC Calibration phase.
- void Motor_SM_Ready (void)

  Motor Ready State - Checks the Ready state of Gate Driver(both Ready-1 and Ready-2)

8.9.1 Detailed Description

: Functionalities pertaining to the V/F controls-Algorithm's state machines (Main & Motor)

Author

: Sreedhar, Sangeerth @company : Agnikul Cosmos Private Limited

Attention

8.10 DrivePOC_Control_Loop.h

Go to the documentation of this file.

```
1
25 /*******************************************************************************
26  * Header inclusions
27  ******************************************************************************/
28
29
30
31 /*******************************************************************************
32  * Function prototypes
33  ******************************************************************************/
34
35
42 void PTC_FaultDiag(void);
43
44
51 void System_FaultDiag(void);
52
53
65 GMCLIB_3COOR_T_FLT Open_Loop_MainStateMC(void);
66
77 void Open_Loop_MotorStateMC(void);
78
79
86 GMCLIB_3COOR_T_FLT Open_Loop_Control(void);
87
88
94 void Get_Duty_Cycle(void);
95
96
102 void Motor_SM_Calibration(void);
103
104
110 void Motor_SM_Ready(void);
111
112
113
114 //Closes the @defgroup block. Always kept last.
115
```

8.11 DrivePOC_Controller_NXP.c File Reference

: Main header file for implementing the Drive POC Controller using NXP microcontroller on the FRDM-KV31F eval board

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MKV31F51212.h"
#include "fsl_debug_console.h"
#include "fsl_ftm.h"
#include "fsl_pit.h"
#include "fsl_common.h"
#include "gmclib_FP.h"
#include "time.h"
#include "Drive_Parameters.h"
#include "DrivePOC_Controller_NXP.h"
#include "DrivePOC_CommHandler.h"
#include "DrivePOC_Control_Loop.h"
#include "DrivePOC_MemHandler.h"
#include "DrivePOC_FaultHandler.h"
#include "mcdrv_frdmkv31f.h"
```

Global Variables

- volatile uint32_t **g_loop_counter** = 0U
- bool_t **g_temperature_check**
- GMCLIB_3COOR_T_FLT **s_pwm_duty_cycle**
- bool **decrement** = false
- void PIT_Configuration (void)

  Function definition to configure the Periodic Interrupt Timer (PIT) @detail The function calls the PIT Handler which runs every 100us or 50us or 33us or 25us or 20 us based on the Switching Frequency For 10kHz - 100 us For 20kHz - 50 us For 30kHz - 33 us For 40kHz - 25 us For 50kHz - 20 us.
- int main (void)
- void PIT_IRQ_HANDLER (void)

8.11.1 Detailed Description

: Main header file for implementing the Drive POC Controller using NXP microcontroller on the FRDM-KV31F eval board

Author

: Sreedhar, Sangeerth @company : Agnikul Cosmos Private Limited

Attention

8.11.2 Function Documentation

8.11.2.1 main()

```
int main (
            void  )
```

Init board hardware.
Init FSL debug console.
Initialize peripherals used
Additional initializations (ADC in the future)
PIT configuration

8.11.2.2 PIT_Configuration()

```
void PIT_Configuration (
            void  )
```

Function definition to configure the Periodic Interrupt Timer (PIT) @detail The function calls the PIT Handler which runs every 100us or 50us or 33us or 25us or 20 us based on the Switching Frequency For 10kHz - 100 us For 20kHz - 50 us For 30kHz - 33 us For 40kHz - 25 us For 50kHz - 20 us.

Parameters

None	

Returns

None

Enable timer interrupts for channel 0

8.11.2.3 PIT_IRQ_HANDLER()

```
void PIT_IRQ_HANDLER (
            void  )
```

Printing debug data over UART on serial terminal

8.12 DrivePOC_Controller_NXP.h File Reference

: Main header file for implementing the Drive POC Controller using NXP microcontroller on the FRDM-KV31F eval board

```
#include "DrivePOC_MemHandler.h"
```

Macros

- #define PIT_IRQ_HANDLER PIT0_IRQHandler
- #define PIT_IRQ_ID PIT0_IRQn
- #define PIT_SOURCE_CLOCK CLOCK_GetFreq(kCLOCK_BusClk)
- #define FTM_SOURCE_CLOCK CLOCK_GetFreq(kCLOCK_BusClk)

DEFINES FOR DIFFERENT TIMES OF EXPECTED INTERRUPTS

- #define LTC_FREQ 20000
- #define PRINTING_FREQ 100000U
- #define **SLOW_LOOP_TIME** 400U

Functions

- void PIT_Configuration (void)

  Function definition to configure the Periodic Interrupt Timer (PIT) @detail The function calls the PIT Handler which runs every 100us or 50us or 33us or 25us or 20 us based on the Switching Frequency For 10kHz - 100 us For 20kHz - 50 us For 30kHz - 33 us For 40kHz - 25 us For 50kHz - 20 us.

8.12.1 Detailed Description

: Main header file for implementing the Drive POC Controller using NXP microcontroller on the FRDM-KV31F eval board

Author

: Sreedhar, Sangeerth @company : Agnikul Cosmos Private Limited

Attention

8.12.2 Macro Definition Documentation

8.12.2.1 FTM_SOURCE_CLOCK

```
#define FTM_SOURCE_CLOCK CLOCK_GetFreq(kCLOCK_BusClk)
```

Source clock for FTM

8.12.2.2 LTC_FREQ

```
#define LTC_FREQ 20000
```
Every 1 sec the LTC Interrupt will be working irrespective of the Switching Frequency

8.12.2.3 PIT_IRQ_HANDLER

```
#define PIT_IRQ_HANDLER(
              void ) PIT0_IRQHandler
```
Defining the peripheral ID of PIT

8.12.2.4 PIT_IRQ_ID

```
#define PIT_IRQ_ID PIT0_IRQn
```
Defining the IRQ ID of PIT

8.12.2.5 PIT_SOURCE_CLOCK

```
#define PIT_SOURCE_CLOCK CLOCK_GetFreq(kCLOCK_BusClk)
```
Source clock for PIT driver

8.12.2.6 PRINTING_FREQ

```
#define PRINTING_FREQ 100000U
```
Factor for 1000 ms

8.12.3 Function Documentation

8.12.3.1 PIT_Configuration()

```
void PIT_Configuration (
             void  )
```
Function definition to configure the Periodic Interrupt Timer (PIT) @detail The function calls the PIT Handler which runs every 100us or 50us or 33us or 25us or 20 us based on the Switching Frequency For 10kHz - 100 us For 20kHz - 50 us For 30kHz - 33 us For 40kHz - 25 us For 50kHz - 20 us.

Parameters

None	

Returns

None

Enable timer interrupts for channel 0

8.13 DrivePOC_Controller_NXP.h

Go to the documentation of this file.

```
1
18 /*******************************************************************************
19  * Header inclusions
20  ******************************************************************************/
21 #include "DrivePOC_MemHandler.h"
22
23 /*******************************************************************************
24  * Definitions
25  ******************************************************************************/
26 #define PIT_IRQ_HANDLER PIT0_IRQHandler
27 #define PIT_IRQ_ID      PIT0_IRQn
28 #define PIT_SOURCE_CLOCK CLOCK_GetFreq(kCLOCK_BusClk)
```

```
29 #define FTM_SOURCE_CLOCK CLOCK_GetFreq(kCLOCK_BusClk)
33 #define LTC_FREQ        20000
34 #define PRINTING_FREQ   100000U
35 #define SLOW_LOOP_TIME 400U
37
38 /*******************************************************************************
39  * Function prototypes
40  ******************************************************************************/
53 void PIT_Configuration(void);
```

8.14 DrivePOC_FaultHandler.c File Reference

: Functionalities pertaining to Handling faults in the complete drive system

```
#include "DrivePOC_FaultHandler.h"
#include "DrivePOC_CommHandler.h"
```

Functions

- void Get_Fault_Duty_Cycle (void)

  Function definition that will make the Duty Cycle of all the switches to zero.

8.14.1 Detailed Description

: Functionalities pertaining to Handling faults in the complete drive system

Author

: Sreedhar, Sangeerth @company : Agnikul Cosmos Private Limited

Attention

8.15 DrivePOC_FaultHandler.h File Reference

: Functionalities pertaining to Handling faults in the complete drive system

```
#include "fsl_debug_console.h"
#include "mcdrv_frdmkv31f.h"
```

Functions

- void Get_Fault_Duty_Cycle (void)

  Function definition that will make the Duty Cycle of all the switches to zero.

8.15.1 Detailed Description

: Functionalities pertaining to Handling faults in the complete drive system

Author

: Sreedhar, Sangeerth @company : Agnikul Cosmos Private Limited

Attention

8.16 DrivePOC_FaultHandler.h

Go to the documentation of this file.

```
1
25 /*******************************************************************************
26  * Header inclusions
27  ******************************************************************************/
28 #include "fsl_debug_console.h"
29 #include "mcdrv_frdmkv31f.h"
30 /*******************************************************************************
31  * Function definitions
32  ******************************************************************************/
33
39 void Get_Fault_Duty_Cycle(void);
40
41
42
```

8.17 DrivePOC_MemHandler.c File Reference

: Functionalities pertaining to all Memory and storage operations of the Drive POC Controller implementation

```
#include "Drive_Parameters.h"
#include "DrivePOC_Controller_NXP.h"
#include "DrivePOC_MemHandler.h"
#include "DrivePOC_CommHandler.h"
#include "DrivePOC_FaultHandler.h"
#include "fsl_gpio.h"
```

Functions

- bool DrivePOC_MH_UpdateEncoderSpeed (float rpm_recd)

  Function definition to update Encoder speed into Memory handler.
- float DrivePOC_MH_GetEncoderSpeed (void)

  Function definition to return Encoder speed from Memory handler.
- void Store_Temperature_from_PT_1000 (void)

  stores the value of temperature measured using PT-1000 RTD in the memory
- void **Get_V_F_Duty_Cycle** (void)

  Function Defintion to give duty cycle during ramp up.
- void Get_Duty_Cycle (void)

  Function definition that gets the value of duty cycle from the duty cycle array.
- void **Get_Deceleration_Duty_Cycle** (void)

  Function Declaration for Deceleration case.
- void DrivePOC_MH_GetVIvalues (void)

  Function definition to allocate the variables to corresponding structures.
- void DrivePOC_MH_UpdateVIvalues (void)

  Function scales the value of Current using the Sensor scaling factors.
- bool **Get_Start_Up_status** (void)

  Function Declaration to get Start Up state.
- bool **Get_Stop_status** (void)

  Function Declaration to get Stop done signal.

Variables

- float_t **g_temperature**
- int **g_phase_shift_A** =0
- int **g_phase_shift_B** =LUT_TEMP_VAL
- int **g_phase_shift_C** =LUT_TEMP_VAL∗2
- int **accu** =0
- int **count** =0
- float_t **freq_prev** =(float_t)MIN_FREQ
- float_t **freq**
- float_t **accel_del_freq** =(FREQ_NEEDED-MIN_FREQ)∗SWITCHINGTIME/(RAMP_TIME)
- float_t **decel_del_freq** =(FREQ_NEEDED-MIN_FREQ)∗SWITCHINGTIME/(DECELERATION_TIME)
- int **rem** =0
- int **start_check** =0
- int **end_check** =0
- bool **start_ok** =false
- bool **stop_ok** =false
- int **bit_max** =BIT_MAX
- float_t **f_freq_map** =((float_t)BIT_MAX/(float_t)FREQ_NEEDED)
- float_t ∗ duty_cycle =duty_cycle_50000hz_669hz

Duty Cycle LUTs for 50kHz Switching Frequency

- float_t duty_cycle_50000hz_800hz [SINE_LUT_800HZ_50KHZ ∗3]
- float_t duty_cycle_50000hz_669hz [SINE_LUT_669HZ_50KHZ ∗3]

Duty Cycle LUTs for 40kHz Switching Frequency

- float_t duty_cycle_40000hz_800hz [SINE_LUT_800HZ_40KHZ ∗3]
- float_t duty_cycle_40000hz_669hz [SINE_LUT_669HZ_40KHZ ∗3]

Duty Cycle LUTs for 30kHz Switching Frequency

- float_t duty_cycle_30000hz_800hz [SINE_LUT_800HZ_30KHZ ∗3]
- float_t duty_cycle_30000hz_669hz [SINE_LUT_669HZ_30KHZ ∗3]

Duty Cycle LUTs for 20kHz Switching Frequency

- float_t **duty_cycle_20000hz_1000hz** [SINE_LUT_1000HZ_20KHZ ∗3] ={0.5F,0.6710101F,0.↩
8213938F,0.9330127F,0.9924039F,0.9924039F,0.9330127F,0.8213938F,0.6710101F,0.5000000F,0.↩
3289899F,0.1786062F,0.6698730F,0.7596123F,0.007596123F,0.06698730F,0.1786062F,0.3289899F}
- float_t **duty_cycle_20000hz_800hz** [SINE_LUT_800HZ_20KHZ ∗3] ={0.500000F,0.629410F,0.↩
750000F,0.853553F,0.933013F,0.982963F,1.000000F,0.982963F,0.933013F,0.853553F,0.750000F,0.↩
629410F,0.500000F,0.370590F,0.250000F,0.146447F,0.066987F,0.017037F,0.000000F,0.017037F,0.↩
066987F,0.146447F,0.250000F,0.370590F}
- float_t **duty_cycle_20000hz_669hz** [SINE_LUT_669HZ_20KHZ ∗3] ={0.500000F,0.603956F,0.↩
703368F,0.793893F,0.871572F,0.933013F,0.975528F,0.997261F,0.997261F,0.975528F,0.933013F,0.↩
871572F,0.793893F,0.703368F,0.603956F,0.500000F,0.396044F,0.296632F,0.206107F,0.128428F,0.↩
066987F,0.024472F,0.002739F,0.002739F,0.024472F,0.066987F,0.128428F,0.206107F,0.296632F,0.↩
396044F}

Duty Cycle LUTs for 10kHz Switching Frequency

- float_t **duty_cycle_10000hz_800hz** [SINE_LUT_800HZ_10KHZ ∗3] ={0.500000F,0.750000F,0.↩
933013F,1.000000F,0.933013F,0.750000F,0.500000F,0.250000F,0.066987F,0.000000F,0.066987F,0.↩
250000F}
- float_t **duty_cycle_10000hz_669hz** [SINE_LUT_669HZ_10KHZ ∗3] ={0.500000F,0.703368F,0.↩
871572F,0.975528F,0.997261F,0.933013F,0.793893F,0.603956F,0.396044F,0.206107F,0.066987F,0.↩
002739F,0.024472F,0.128428F,0.296632F}

- uint8_t * **p_ads_rx**
- int16_t **g_Ia** =0
- int16_t **g_Ib** =0
- int16_t **g_Ic** =0
- int16_t **g_Idc** =0
- int16_t **g_Va** =0
- int16_t **g_Vb** =0
- int16_t **g_Vc** =0
- int16_t **g_Vdc** =0

8.17.1 Detailed Description

: Functionalities pertaining to all Memory and storage operations of the Drive POC Controller implementation

Author

: Sreedhar, Sangeerth @company : Agnikul Cosmos Private Limited

Attention

8.17.2 Variable Documentation

8.17.2.1 duty_cycle

```
float_t* duty_cycle =duty_cycle_50000hz_669hz
```

< User Input - User has to change this value declared to this pointer corresponding to the Switching frequency and AC Sine Frequency he/she wish to operate

8.17.2.2 duty_cycle_30000hz_669hz

```
float_t duty_cycle_30000hz_669hz[SINE_LUT_669HZ_30KHZ *3]
```

Initial value:

```
={0.500000F,0.569587F,0.637819F,0.703368F,0.764960F,0.821394F,0.871572F,0.914519F,0.949397F,0.975528F,0.992404F,0.999695F,0.9
        0.007596F,0.024472F,0.050603F,0.085481F,0.128428F,0.178606F,0.235040F,0.296632F,0.362181F,0.430413F}
```

8.17.2.3 duty_cycle_30000hz_800hz

```
float_t duty_cycle_30000hz_800hz[SINE_LUT_800HZ_30KHZ *3]
```

Initial value:

```
={0.500000F,0.580206F,0.658334F,0.732362F,0.800371F,0.860601F,0.911492F,0.951725F,0.980259F,0.996354F,0.999594F,0.989895F,0.9
        0.199629F,0.267638F,0.341666F,0.419794F}
```

8.17.2.4 duty_cycle_40000hz_669hz

```
float_t duty_cycle_40000hz_669hz[SINE_LUT_669HZ_40KHZ *3]
```

Initial value:

```
={0.500000F,0.552264F,0.603956F,0.654508F,0.703368F,0.750000F,0.793893F,0.834565F,0.871572F,0.904508F,0.933013F,0.956773F,0.9

       0.250000F,0.206107F,0.165435F,0.128428F,0.095492F,0.066987F,0.043227F,0.024472F,0.010926F,0.002739F,0.000000F,0.002739
```

8.17.2.5 duty_cycle_40000hz_800hz

```
float_t duty_cycle_40000hz_800hz[SINE_LUT_800HZ_40KHZ *3]
```

Initial value:

```
={0.500000F,0.561444F,0.621957F,0.680621F,0.736547F,0.788887F,0.836848F,0.879702F,0.916801F,0.947582F,0.971577F,0.988424F,0.9

       0.039547F,0.019087F,0.005917F,0.000237F,0.002133F,0.011576F,0.028423F,0.052418F,0.083199F,0.120298F,0.163152F,0.211113
```

8.17.2.6 duty_cycle_50000hz_669hz

```
float_t duty_cycle_50000hz_669hz[SINE_LUT_669HZ_50KHZ *3]
```

Initial value:

```
={0.500000F,0.541839F,0.583384F,0.624345F,0.664433F,0.703368F,0.740877F,0.776696F,0.810574F,0.842274F,0.871572F,0.898265F,0.92

       0.603956F,0.562667F,0.520938F,0.479062F,0.437333F,0.396044F,0.355484F,0.315938F,0.277682F,0.240986F,0.206107F,0.173290
```

8.17.2.7 duty_cycle_50000hz_800hz

```
float_t duty_cycle_50000hz_800hz[SINE_LUT_800HZ_50KHZ *3]
```

Initial value:

```
={0.500000F,0.549784F,0.599073F,0.647378F,0.694217F,0.739127F,0.781660F,0.821394F,0.857933F,0.890916F,0.920013F,0.944936F,0.96

       0.328990F,0.283058F,0.239282F,0.198098F,0.159914F,0.125109F,0.094031F,0.066987F,0.044247F,0.026036F,0.012536F,0.003880
```

8.18 DrivePOC_MemHandler.h File Reference

: Functionalities pertaining to all Memory and storage operations of the Drive POC Controller implementation

```
#include "fsl_debug_console.h"
#include "gmclib_types.h"
```

Macros

Sine Frequency

- #define **FSINE_800** 800
- #define **FSINE_669** 669
- #define **FSINE_600** 600
- #define **FSINE_500** 500
- #define **FSINE_400** 400
- #define **FSINE_300** 300
- #define **FSINE_200** 200
- #define **FSINE_100** 100

Switching Frequency 50kHz Data

- #define **FSWITCHING_800HZ_50KHZ** 50400
- #define **FSWITCHING_669HZ_50KHZ** 50175
- #define **FSWITCHING_600HZ_50KHZ** 50400
- #define **FSWITCHING_500HZ_50KHZ** 49500
- #define **FSWITCHING_400HZ_50KHZ** 50400
- #define **FSWITCHING_300HZ_50KHZ** 50400
- #define **FSWITCHING_200HZ_50KHZ** 49800
- #define **FSWITCHING_100HZ_50KHZ** 50100
- #define **SINE_LUT_800HZ_50KHZ** 21
- #define **SINE_LUT_669HZ_50KHZ** 25
- #define **SINE_LUT_600HZ_50KHZ** 28
- #define **SINE_LUT_500HZ_50KHZ** 33
- #define **SINE_LUT_400HZ_50KHZ** 42
- #define **SINE_LUT_300HZ_50KHZ** 56
- #define **SINE_LUT_200HZ_50KHZ** 83
- #define **SINE_LUT_100HZ_50KHZ** 167

Switching Frequency 40kHz Data

- #define **FSWITCHING_800HZ_40KHZ** 40800
- #define **FSWITCHING_669HZ_40KHZ** 40140
- #define **FSWITCHING_600HZ_40KHZ** 39600
- #define **FSWITCHING_500HZ_40KHZ** 40500
- #define **FSWITCHING_400HZ_40KHZ** 39600
- #define **FSWITCHING_300HZ_40KHZ** 39600
- #define **FSWITCHING_200HZ_40KHZ** 40200
- #define **FSWITCHING_100HZ_40KHZ** 40200
- #define **SINE_LUT_800HZ_40KHZ** 17
- #define **SINE_LUT_669HZ_40KHZ** 20
- #define **SINE_LUT_600HZ_40KHZ** 22
- #define **SINE_LUT_500HZ_40KHZ** 27
- #define **SINE_LUT_400HZ_40KHZ** 33
- #define **SINE_LUT_300HZ_40KHZ** 44
- #define **SINE_LUT_200HZ_40KHZ** 67
- #define **SINE_LUT_100HZ_40KHZ** 134

Switching Frequency 30kHz Data

- #define **FSWITCHING_800HZ_30KHZ** 31200
- #define **FSWITCHING_669HZ_30KHZ** 30105
- #define **FSWITCHING_600HZ_30KHZ** 30600
- #define **FSWITCHING_500HZ_30KHZ** 30000
- #define **FSWITCHING_400HZ_30KHZ** 30000
- #define **FSWITCHING_300HZ_30KHZ** 29700
- #define **FSWITCHING_200HZ_30KHZ** 30000
- #define **FSWITCHING_100HZ_30KHZ** 30000
- #define **SINE_LUT_800HZ_30KHZ** 13
- #define **SINE_LUT_669HZ_30KHZ** 15
- #define **SINE_LUT_600HZ_30KHZ** 17
- #define **SINE_LUT_500HZ_30KHZ** 20
- #define **SINE_LUT_400HZ_30KHZ** 25
- #define **SINE_LUT_300HZ_30KHZ** 33
- #define **SINE_LUT_200HZ_30KHZ** 50
- #define **SINE_LUT_100HZ_30KHZ** 100

Switching Frequency 20kHz Data

- #define **FSWITCHING_1000HZ_20KHZ** 18000
- #define **FSWITCHING_800HZ_20KHZ** 19200
- #define **FSWITCHING_669HZ_20KHZ** 20070
- #define **FSWITCHING_600HZ_20KHZ** 19800
- #define **FSWITCHING_500HZ_20KHZ** 19500
- #define **FSWITCHING_400HZ_20KHZ** 20400
- #define **FSWITCHING_300HZ_20KHZ** 19800
- #define **FSWITCHING_200HZ_20KHZ** 20400
- #define **FSWITCHING_100HZ_20KHZ** 20100
- #define **SINE_LUT_1000HZ_20KHZ** 6
- #define **SINE_LUT_800HZ_20KHZ** 8
- #define **SINE_LUT_669HZ_20KHZ** 10
- #define **SINE_LUT_600HZ_20KHZ** 11
- #define **SINE_LUT_500HZ_20KHZ** 13
- #define **SINE_LUT_400HZ_20KHZ** 17
- #define **SINE_LUT_300HZ_20KHZ** 22
- #define **SINE_LUT_200HZ_20KHZ** 37
- #define **SINE_LUT_100HZ_20KHZ** 67

Switching Frequency 10kHz Data

- #define **FSWITCHING_800HZ_10KHZ** 9600
- #define **FSWITCHING_669HZ_10KHZ** 10035
- #define **FSWITCHING_600HZ_10KHZ** 10800
- #define **FSWITCHING_500HZ_10KHZ** 10500

- #define **FSWITCHING_400HZ_10KHZ** 9600
- #define **FSWITCHING_300HZ_10KHZ** 9900
- #define **FSWITCHING_200HZ_10KHZ** 10200
- #define **FSWITCHING_100HZ_10KHZ** 10200
- #define **SINE_LUT_800HZ_10KHZ** 4
- #define **SINE_LUT_669HZ_10KHZ** 5
- #define **SINE_LUT_600HZ_10KHZ** 6
- #define **SINE_LUT_500HZ_10KHZ** 7
- #define **SINE_LUT_400HZ_10KHZ** 8
- #define **SINE_LUT_300HZ_10KHZ** 11
- #define **SINE_LUT_200HZ_10KHZ** 17
- #define **SINE_LUT_100HZ_10KHZ** 34

Interrupt Timing values

- #define **T_SWITCHING_50KHZ** 1000000/50000
- #define **T_SWITCHING_40KHZ** 1000000/40000
- #define **T_SWITCHING_30KHZ** 1000000/30000
- #define **T_SWITCHING_20KHZ** 1000000/20000
- #define **T_SWITCHING_10KHZ** 1000000/10000

LTC Interrupt Frequency

- #define **LTC_SWITCHING_FREQ_10KHZ** 10000
- #define **LTC_SWITCHING_FREQ_20KHZ** 20000
- #define **LTC_SWITCHING_FREQ_30KHZ** 30000
- #define **LTC_SWITCHING_FREQ_40KHZ** 40000
- #define **LTC_SWITCHING_FREQ_50KHZ** 50000

ADS Board Parameters

- #define **ADS_BOARD_POSITIVE_VOLTAGE_LIMIT** 0x07FFF
- #define **ADS_BOARD_ZERO_VAL** 0.0F

Functions

- bool DrivePOC_MH_UpdateEncoderSpeed (float rpm_recd)

  Function definition to update Encoder speed into Memory handler.
- float DrivePOC_MH_GetEncoderSpeed (void)

  Function definition to return Encoder speed from Memory handler.
- void Store_Temperature_from_PT_1000 (void)

  stores the value of temperature measured using PT-1000 RTD in the memory
- void Get_Duty_Cycle (void)

  Function definition that gets the value of duty cycle from the duty cycle array.
- void DrivePOC_MH_UpdateVIvalues (void)

  Function scales the value of Current using the Sensor scaling factors.
- void DrivePOC_MH_GetVIvalues (void)

  Function definition to allocate the variables to corresponding structures.
- void **Get_V_F_Duty_Cycle** (void)

  Function Defintion to give duty cycle during ramp up.
- void **Get_Deceleration_Duty_Cycle** (void)

  Function Declaration for Deceleration case.
- bool **Get_Start_Up_status** (void)

  Function Declaration to get Start Up state.
- bool **Get_Stop_status** (void)

  Function Declaration to get Stop done signal.

8.18.1 Detailed Description

: Functionalities pertaining to all Memory and storage operations of the Drive POC Controller implementation

Author

: Sreedhar, Sangeerth @company : Agnikul Cosmos Private Limited

Attention

8.19 DrivePOC_MemHandler.h

Go to the documentation of this file.

```
1
26 /*****************************************************************************
27  * Header inclusions
28  *****************************************************************************/
29 #include "fsl_debug_console.h"
30 #include "gmclib_types.h"
31
34 #define FSINE_800 800
35 #define FSINE_669 669
36 #define FSINE_600 600
37 #define FSINE_500 500
38 #define FSINE_400 400
39 #define FSINE_300 300
40 #define FSINE_200 200
41 #define FSINE_100 100
43
44
47 #define FSWITCHING_800HZ_50KHZ 50400
48 #define FSWITCHING_669HZ_50KHZ 50175
49 #define FSWITCHING_600HZ_50KHZ 50400
50 #define FSWITCHING_500HZ_50KHZ 49500
51 #define FSWITCHING_400HZ_50KHZ 50400
52 #define FSWITCHING_300HZ_50KHZ 50400
53 #define FSWITCHING_200HZ_50KHZ 49800
54 #define FSWITCHING_100HZ_50KHZ 50100
55 #define SINE_LUT_800HZ_50KHZ    21
56 #define SINE_LUT_669HZ_50KHZ    25
57 #define SINE_LUT_600HZ_50KHZ    28
58 #define SINE_LUT_500HZ_50KHZ    33
59 #define SINE_LUT_400HZ_50KHZ    42
60 #define SINE_LUT_300HZ_50KHZ    56
61 #define SINE_LUT_200HZ_50KHZ    83
62 #define SINE_LUT_100HZ_50KHZ    167
64
67 #define FSWITCHING_800HZ_40KHZ 40800
68 #define FSWITCHING_669HZ_40KHZ 40140
69 #define FSWITCHING_600HZ_40KHZ 39600
70 #define FSWITCHING_500HZ_40KHZ 40500
71 #define FSWITCHING_400HZ_40KHZ 39600
72 #define FSWITCHING_300HZ_40KHZ 39600
73 #define FSWITCHING_200HZ_40KHZ 40200
74 #define FSWITCHING_100HZ_40KHZ 40200
75 #define SINE_LUT_800HZ_40KHZ    17
76 #define SINE_LUT_669HZ_40KHZ    20
77 #define SINE_LUT_600HZ_40KHZ    22
78 #define SINE_LUT_500HZ_40KHZ    27
79 #define SINE_LUT_400HZ_40KHZ    33
80 #define SINE_LUT_300HZ_40KHZ    44
81 #define SINE_LUT_200HZ_40KHZ    67
82 #define SINE_LUT_100HZ_40KHZ    134
84
85
88 #define FSWITCHING_800HZ_30KHZ 31200
89 #define FSWITCHING_669HZ_30KHZ 30105
90 #define FSWITCHING_600HZ_30KHZ 30600
91 #define FSWITCHING_500HZ_30KHZ 30000
92 #define FSWITCHING_400HZ_30KHZ 30000
93 #define FSWITCHING_300HZ_30KHZ 29700
94 #define FSWITCHING_200HZ_30KHZ 30000
95 #define FSWITCHING_100HZ_30KHZ 30000
96 #define SINE_LUT_800HZ_30KHZ    13
```

```
97 #define SINE_LUT_669HZ_30KHZ    15
98 #define SINE_LUT_600HZ_30KHZ    17
99 #define SINE_LUT_500HZ_30KHZ    20
100 #define SINE_LUT_400HZ_30KHZ    25
101 #define SINE_LUT_300HZ_30KHZ    33
102 #define SINE_LUT_200HZ_30KHZ    50
103 #define SINE_LUT_100HZ_30KHZ    100
105
108 #define FSWITCHING_1000HZ_20KHZ 18000
109 #define FSWITCHING_800HZ_20KHZ 19200
110 #define FSWITCHING_669HZ_20KHZ 20070
111 #define FSWITCHING_600HZ_20KHZ 19800
112 #define FSWITCHING_500HZ_20KHZ 19500
113 #define FSWITCHING_400HZ_20KHZ 20400
114 #define FSWITCHING_300HZ_20KHZ 19800
115 #define FSWITCHING_200HZ_20KHZ 20400
116 #define FSWITCHING_100HZ_20KHZ 20100
117 #define SINE_LUT_1000HZ_20KHZ   6
118 #define SINE_LUT_800HZ_20KHZ    8
119 #define SINE_LUT_669HZ_20KHZ    10
120 #define SINE_LUT_600HZ_20KHZ    11
121 #define SINE_LUT_500HZ_20KHZ    13
122 #define SINE_LUT_400HZ_20KHZ    17
123 #define SINE_LUT_300HZ_20KHZ    22
124 #define SINE_LUT_200HZ_20KHZ    37
125 #define SINE_LUT_100HZ_20KHZ    67
127
128
131 #define FSWITCHING_800HZ_10KHZ 9600
132 #define FSWITCHING_669HZ_10KHZ 10035
133 #define FSWITCHING_600HZ_10KHZ 10800
134 #define FSWITCHING_500HZ_10KHZ 10500
135 #define FSWITCHING_400HZ_10KHZ 9600
136 #define FSWITCHING_300HZ_10KHZ 9900
137 #define FSWITCHING_200HZ_10KHZ 10200
138 #define FSWITCHING_100HZ_10KHZ 10200
139 #define SINE_LUT_800HZ_10KHZ    4
140 #define SINE_LUT_669HZ_10KHZ    5
141 #define SINE_LUT_600HZ_10KHZ    6
142 #define SINE_LUT_500HZ_10KHZ    7
143 #define SINE_LUT_400HZ_10KHZ    8
144 #define SINE_LUT_300HZ_10KHZ    11
145 #define SINE_LUT_200HZ_10KHZ    17
146 #define SINE_LUT_100HZ_10KHZ    34
148
151 #define T_SWITCHING_50KHZ 1000000/50000
152 #define T_SWITCHING_40KHZ 1000000/40000
153 #define T_SWITCHING_30KHZ 1000000/30000
154 #define T_SWITCHING_20KHZ 1000000/20000
155 #define T_SWITCHING_10KHZ 1000000/10000
157
160 #define LTC_SWITCHING_FREQ_10KHZ 10000
161 #define LTC_SWITCHING_FREQ_20KHZ 20000
162 #define LTC_SWITCHING_FREQ_30KHZ 30000
163 #define LTC_SWITCHING_FREQ_40KHZ 40000
164 #define LTC_SWITCHING_FREQ_50KHZ 50000
166
167
170 #define ADS_BOARD_POSITIVE_VOLTAGE_LIMIT 0x07FFF
171 #define ADS_BOARD_ZERO_VAL 0.0F
173
174
175 /*****************************************************************************
176  * Function prototypes
177  *****************************************************************************/
185 bool DrivePOC_MH_UpdateEncoderSpeed(float rpm_recd);
186
187
194 float DrivePOC_MH_GetEncoderSpeed(void);
195
196
202 void Store_Temperature_from_PT_1000(void);
203
204
205
211 void Get_Duty_Cycle(void);
212
213
219 void DrivePOC_MH_UpdateVIvalues(void);
220
221
227 void DrivePOC_MH_GetVIvalues(void);
228
229
233 void Get_V_F_Duty_Cycle(void);
234
235
```

```
239 void Get_Deceleration_Duty_Cycle(void);
240
241
245 bool Get_Start_Up_status(void);
246
247
251 bool Get_Stop_status(void);
252
```