Project 6

2

Generated by Doxygen 1.8.16

1 File Index 1

1.1 File List . . . 1

2 File Documentation 3

2.1 CANFunctions.c File Reference . . . 3

2.1.1 Detailed Description . . . 4

2.1.2 Function Documentation . . . 4

2.1.2.1 __ISR() [1/2] . . . 4

2.1.2.2 __ISR() [2/2] . . . 4

2.1.2.3 CAN1_process_RX() . . . 5

2.1.2.4 CAN1_send_RTR() . . . 5

2.1.2.5 CAN1_send_TX() . . . 6

2.1.2.6 CAN2_process_RX() . . . 6

2.1.2.7 CAN2_refill_RTR_buffer() . . . 6

2.1.2.8 initialize_CAN1() . . . 7

2.1.2.9 initialize_CAN2() . . . 7

2.1.3 Variable Documentation . . . 7

2.1.3.1 CAN1_received_flag . . . 8

2.1.3.2 CAN1MessageFifoArea . . . 8

2.1.3.3 CAN2_received_flag . . . 8

2.1.3.4 CAN2MessageFifoArea . . . 8

2.2 CANFunctions.h File Reference . . . 8

2.2.1 Detailed Description . . . 9

2.2.2 Macro Definition Documentation . . . 9

2.2.2.1 BYTE_0_MASK . . . 9

2.2.2.2 BYTE_1_MASK . . . 9

2.2.2.3 BYTE_2_MASK . . . 9

2.2.2.4 BYTE_3_MASK . . . 10

2.2.2.5 CAN1_CHANNELS . . . 10

2.2.2.6 CAN1_FIFO_BUFFERS . . . 10

2.2.2.7 CAN1_MSG_BUFF_SIZE . . . 10

2.2.2.8 CAN1_MSG_MEMORY . . . 10

2.2.2.9 CAN1_PWM_MESSAGE_ID . . . 10

2.2.2.10 CAN2_CHANNELS . . . 10

2.2.2.11 CAN2_FIFO_BUFFERS . . . 10

2.2.2.12 CAN2_MSG_BUFF_SIZE . . . 11

2.2.2.13 CAN2_MSG_MEMORY . . . 11

2.2.2.14 CAN2_RTR_MESSAGE_ID . . . 11

2.2.2.15 CAN_BUS_SPEED . . . 11

2.2.2.16 CAN_MESSAGE_RECEIVED . . . 11

2.2.2.17 CAN_NO_MESSAGE_RECEIVED . . . 11

2.2.2.18 EID_BIT_MASK . . . 11

2.2.2.19 EID_FILTER_MASK . . . 11
2.2.2.20 SID_BIT_MASK . . . 12
2.2.2.21 SID_FILTER_MASK . . . 12
2.2.3 Function Documentation . . . 12
2.2.3.1 CAN1_process_RX() . . . 12
2.2.3.2 CAN1_send_RTR() . . . 12
2.2.3.3 CAN1_send_TX() . . . 13
2.2.3.4 CAN2_process_RX() . . . 13
2.2.3.5 CAN2_refill_RTR_buffer() . . . 13
2.2.3.6 initialize_CAN1() . . . 14
2.2.3.7 initialize_CAN2() . . . 14
2.3 chipKIT_PRO_MX7.c File Reference . . . 14
2.3.1 Function Documentation . . . 15
2.3.1.1 chipKIT_PRO_MX7_Setup() . . . 15
2.4 chipKIT_PRO_MX7.h File Reference . . . 15
2.4.1 Macro Definition Documentation . . . 16
2.4.1.1 BRD_LEDS . . . 16
2.4.1.2 BTN1 . . . 16
2.4.1.3 BTN2 . . . 16
2.4.1.4 BTN3 . . . 16
2.4.1.5 CORE_MS_TICK_RATE . . . 17
2.4.1.6 FPB . . . 17
2.4.1.7 GetCoreClock . . . 17
2.4.1.8 GetPeripheralClock . . . 17
2.4.1.9 GetSystemClock . . . 17
2.4.1.10 LED1 . . . 17
2.4.1.11 LED1_IO . . . 17
2.4.1.12 LED2 . . . 18
2.4.1.13 LED2_IO . . . 18
2.4.1.14 LED3 . . . 18
2.4.1.15 LED3_IO . . . 18
2.4.1.16 LED4 . . . 18
2.4.1.17 LED4_IO . . . 18
2.4.1.18 LEDA . . . 18
2.4.1.19 LEDA_IO . . . 19
2.4.1.20 LEDB . . . 19
2.4.1.21 LEDB_IO . . . 19
2.4.1.22 LEDC . . . 19
2.4.1.23 LEDC_IO . . . 19
2.4.1.24 LEDD . . . 19
2.4.1.25 LEDD_IO . . . 19
2.4.1.26 LEDE . . . 20

2.4.1.27 LEDE_IO . . . 20
2.4.1.28 LEDF . . . 20
2.4.1.29 LEDF_IO . . . 20
2.4.1.30 LEDG . . . 20
2.4.1.31 LEDG_IO . . . 20
2.4.1.32 LEDH . . . 20
2.4.1.33 LEDH_IO . . . 21
2.4.1.34 SM1 . . . 21
2.4.1.35 SM2 . . . 21
2.4.1.36 SM3 . . . 21
2.4.1.37 SM4 . . . 21
2.4.1.38 SM_COILS . . . 21
2.4.1.39 SM_LEDS . . . 21
2.4.1.40 SYSTEM_FREQ . . . 22
2.4.1.41 XTAL . . . 22
2.4.2 Function Documentation . . . 22
2.4.2.1 chipKIT_PRO_MX7_Setup() . . . 22
2.5 config_bits.h File Reference . . . 22
2.6 FreeRTOSConfig.h File Reference . . . 22
2.6.1 Macro Definition Documentation . . . 23
2.6.1.1 configCHECK_FOR_STACK_OVERFLOW . . . 23
2.6.1.2 configCPU_CLOCK_HZ . . . 23
2.6.1.3 configGENERATE_RUN_TIME_STATS . . . 23
2.6.1.4 configIDLE_SHOULD_YIELD . . . 24
2.6.1.5 configISR_STACK_SIZE . . . 24
2.6.1.6 configKERNEL_INTERRUPT_PRIORITY . . . 24
2.6.1.7 configMAX_CO_ROUTINE_PRIORITIES . . . 24
2.6.1.8 configMAX_PRIORITIES . . . 24
2.6.1.9 configMAX_SYSCALL_INTERRUPT_PRIORITY . . . 24
2.6.1.10 configMAX_TASK_NAME_LEN . . . 24
2.6.1.11 configMINIMAL_STACK_SIZE . . . 24
2.6.1.12 configPERIPHERAL_CLOCK_HZ . . . 25
2.6.1.13 configQUEUE_REGISTRY_SIZE . . . 25
2.6.1.14 configTICK_RATE_HZ . . . 25
2.6.1.15 configTIMER_QUEUE_LENGTH . . . 25
2.6.1.16 configTIMER_TASK_PRIORITY . . . 25
2.6.1.17 configTIMER_TASK_STACK_DEPTH . . . 25
2.6.1.18 configTOTAL_HEAP_SIZE . . . 25
2.6.1.19 configUSE_16_BIT_TICKS . . . 25
2.6.1.20 configUSE_APPLICATION_TASK_TAG . . . 26
2.6.1.21 configUSE_CO_ROUTINES . . . 26
2.6.1.22 configUSE_COUNTING_SEMAPHORES . . . 26

2.6.1.23 configUSE_IDLE_HOOK . . . 26
2.6.1.24 configUSE_MALLOC_FAILED_HOOK . . . 26
2.6.1.25 configUSE_MUTEXES . . . 26
2.6.1.26 configUSE_PORT_OPTIMISED_TASK_SELECTION . . . 26
2.6.1.27 configUSE_PREEMPTION . . . 26
2.6.1.28 configUSE_RECURSIVE_MUTEXES . . . 27
2.6.1.29 configUSE_TICK_HOOK . . . 27
2.6.1.30 configUSE_TIMERS . . . 27
2.6.1.31 configUSE_TRACE_FACILITY . . . 27
2.6.1.32 INCLUDE_eTaskGetState . . . 27
2.6.1.33 INCLUDE_uxTaskGetStackHighWaterMark . . . 27
2.6.1.34 INCLUDE_uxTaskPriorityGet . . . 27
2.6.1.35 INCLUDE_vTaskCleanUpResources . . . 27
2.6.1.36 INCLUDE_vTaskDelay . . . 28
2.6.1.37 INCLUDE_vTaskDelayUntil . . . 28
2.6.1.38 INCLUDE_vTaskDelete . . . 28
2.6.1.39 INCLUDE_vTaskPrioritySet . . . 28
2.6.1.40 INCLUDE_vTaskSuspend . . . 28
2.7 input_capture.c File Reference . . . 28
2.7.1 Detailed Description . . . 29
2.7.2 Function Documentation . . . 29
2.7.2.1 __ISR() [1/2] . . . 29
2.7.2.2 __ISR() [2/2] . . . 29
2.7.2.3 get_average_rps() . . . 30
2.7.2.4 initialize_input_capture() . . . 30
2.7.3 Variable Documentation . . . 30
2.7.3.1 rps_buffer . . . 30
2.8 input_capture.h File Reference . . . 31
2.8.1 Detailed Description . . . 31
2.8.2 Macro Definition Documentation . . . 31
2.8.2.1 SPEED_BUFFER_LEN . . . 31
2.8.3 Function Documentation . . . 31
2.8.3.1 get_average_rps() . . . 31
2.8.3.2 initialize_input_capture() . . . 32
2.9 LCDlib.c File Reference . . . 32
2.9.1 Detailed Description . . . 33
2.9.2 Function Documentation . . . 33
2.9.2.1 _write_LCD() . . . 33
2.9.2.2 initialize_LCD() . . . 33
2.9.2.3 put_char_LCD() . . . 34
2.9.2.4 put_string_LCD() . . . 34
2.9.2.5 read_LCD() . . . 34

2.9.2.6 reset_clear_LCD() . . . 35
2.9.2.7 set_cursor_LCD() . . . 35
2.9.2.8 sw_delay_ms() . . . 35
2.9.2.9 sw_delay_us() . . . 36
2.10 LCDlib.h File Reference . . . 36
2.10.1 Detailed Description . . . 37
2.10.2 Macro Definition Documentation . . . 37
2.10.2.1 COUNTS_PER_MS . . . 37
2.10.2.2 FIRST_LINE_END . . . 37
2.10.2.3 FIRST_LINE_START . . . 37
2.10.2.4 LCD_RS_CMD . . . 37
2.10.2.5 LCD_RS_DATA . . . 38
2.10.2.6 SECOND_LINE_END . . . 38
2.10.2.7 SECOND_LINE_START . . . 38
2.10.3 Function Documentation . . . 38
2.10.3.1 _write_LCD() . . . 38
2.10.3.2 initialize_LCD() . . . 38
2.10.3.3 put_char_LCD() . . . 39
2.10.3.4 put_string_LCD() . . . 39
2.10.3.5 read_LCD() . . . 39
2.10.3.6 reset_clear_LCD() . . . 40
2.10.3.7 set_cursor_LCD() . . . 40
2.10.3.8 sw_delay_ms() . . . 40
2.10.3.9 sw_delay_us() . . . 40
2.11 main.c File Reference . . . 41
2.11.1 Detailed Description . . . 42
2.11.2 Function Documentation . . . 42
2.11.2.1 __ISR() . . . 42
2.11.2.2 _general_exception_handler() . . . 42
2.11.2.3 clear_string_buffer() . . . 42
2.11.2.4 create_RTOS_objects() . . . 43
2.11.2.5 create_tasks() . . . 43
2.11.2.6 initialize_hardware() . . . 43
2.11.2.7 isr_change_notice_handler() . . . 44
2.11.2.8 main() . . . 44
2.11.2.9 task_change_notice_handler() . . . 44
2.11.2.10 task_control_FSM() . . . 44
2.11.2.11 task_read_IO() . . . 45
2.11.2.12 task_send_RTR() . . . 45
2.11.2.13 task_update_pwm() . . . 45
2.11.2.14 vApplicationIdleHook() . . . 46
2.11.2.15 vApplicationStackOverflowHook() . . . 46

2.11.2.16 vApplicationTickHook() . . . 46
2.11.3 Variable Documentation . . . 46
2.11.3.1 current_state . . . 46
2.11.3.2 latest_pwm_setting . . . 46
2.11.3.3 latest_rps . . . 47
2.11.3.4 latest_temp . . . 47
2.11.3.5 previous_BTN1_status . . . 47
2.11.3.6 trace_CN . . . 47
2.11.3.7 trace_control_fsm . . . 47
2.11.3.8 trace_IO . . . 47
2.11.3.9 trace_RTR . . . 47
2.12 main.h File Reference . . . 47
2.12.1 Detailed Description . . . 48
2.12.2 Macro Definition Documentation . . . 48
2.12.2.1 DEBOUNCE_MS . . . 48
2.12.2.2 FALSE . . . 49
2.12.2.3 IO_FREQ_MS . . . 49
2.12.2.4 MS_TO_TICKS . . . 49
2.12.2.5 PWM_FREQUENCY_HZ . . . 49
2.12.2.6 PWM_LINEAR_MAX . . . 49
2.12.2.7 PWM_LINEAR_MIN . . . 49
2.12.2.8 PWM_MAX_VAL . . . 49
2.12.2.9 PWM_MIN_VAL . . . 50
2.12.2.10 RTR_FREQ_MS . . . 50
2.12.2.11 TASK_CHANGE_NOTICE_PRIORITY . . . 50
2.12.2.12 TASK_CONTROL_FSM_PRIORITY . . . 50
2.12.2.13 TASK_READ_IO_PRIORITY . . . 50
2.12.2.14 TASK_SEND_RTR_PRIORITY . . . 50
2.12.2.15 TASK_UPDATE_PWM_PRIORITY . . . 50
2.12.2.16 TRUE . . . 50
2.12.3 Function Documentation . . . 51
2.12.3.1 average_rps_calculator() . . . 51
2.12.3.2 clear_string_buffer() . . . 51
2.12.3.3 create_RTOS_objects() . . . 51
2.12.3.4 create_tasks() . . . 51
2.12.3.5 initialize_hardware() . . . 51
2.12.3.6 isr_change_notice_handler() . . . 51
2.12.3.7 task_change_notice_handler() . . . 52
2.12.3.8 task_control_FSM() . . . 52
2.12.3.9 task_read_IO() . . . 52
2.12.3.10 task_send_RTR() . . . 52
2.12.3.11 task_update_pwm() . . . 52

2.13 PWM_library.c File Reference . . . 52
2.13.1 Function Documentation . . . 53
2.13.1.1 __ISR() . . . 53
2.13.1.2 initialize_pwm() . . . 53
2.13.1.3 set_pwm() . . . 54
2.13.2 Variable Documentation . . . 54
2.13.2.1 t2_tick . . . 54
2.14 PWM_library.h File Reference . . . 54
2.14.1 Macro Definition Documentation . . . 54
2.14.1.1 T2_CLOCK_RATE . . . 55
2.14.1.2 T2_PRESCALE . . . 55
2.14.2 Function Documentation . . . 55
2.14.2.1 initialize_pwm() . . . 55
2.14.2.2 set_pwm() . . . 55
2.15 SMBus_IR.c File Reference . . . 56
2.15.1 Detailed Description . . . 56
2.15.2 Function Documentation . . . 56
2.15.2.1 initialize_ir_sensor() . . . 56
2.15.2.2 read_ir_temp() . . . 57
2.15.2.3 wait_for_ack() . . . 57
2.16 SMBus_IR.h File Reference . . . 58
2.16.1 Detailed Description . . . 58
2.16.2 Macro Definition Documentation . . . 58
2.16.2.1 ATTEMPT_COUNT . . . 58
2.16.2.2 BAUD_RATE . . . 59
2.16.2.3 CELCIUS_TO_FARENHEIT . . . 59
2.16.2.4 ERROR_TEMP . . . 59
2.16.2.5 I2C_CLOCK_VAL . . . 59
2.16.2.6 IR_SENSOR_RES . . . 59
2.16.2.7 KELVIN_TO_CELSIUS . . . 59
2.16.2.8 READ . . . 59
2.16.2.9 READ_ERROR_FLAG . . . 60
2.16.2.10 SLAVE_ADDR . . . 60
2.16.2.11 T_OBJ_ADDR . . . 60
2.16.2.12 WRITE . . . 60
2.16.3 Function Documentation . . . 60
2.16.3.1 initialize_ir_sensor() . . . 60
2.16.3.2 read_ir_temp() . . . 60
2.16.3.3 wait_for_ack() . . . 61
2.17 trcConfig.h File Reference . . . 61
2.17.1 Macro Definition Documentation . . . 62
2.17.1.1 TRC_CFG_FREERTOS_VERSION . . . 62

2.17.1.2 TRC_CFG_HARDWARE_PORT . . . 62
2.17.1.3 TRC_CFG_INCLUDE_EVENT_GROUP_EVENTS . . . 62
2.17.1.4 TRC_CFG_INCLUDE_ISR_TRACING . . . 62
2.17.1.5 TRC_CFG_INCLUDE_MEMMANG_EVENTS . . . 62
2.17.1.6 TRC_CFG_INCLUDE_OSTICK_EVENTS . . . 62
2.17.1.7 TRC_CFG_INCLUDE_PEND_FUNC_CALL_EVENTS . . . 62
2.17.1.8 TRC_CFG_INCLUDE_READY_EVENTS . . . 63
2.17.1.9 TRC_CFG_INCLUDE_STREAM_BUFFER_EVENTS . . . 63
2.17.1.10 TRC_CFG_INCLUDE_TIMER_EVENTS . . . 63
2.17.1.11 TRC_CFG_INCLUDE_USER_EVENTS . . . 63
2.17.1.12 TRC_CFG_MAX_ISR_NESTING . . . 63
2.17.1.13 TRC_CFG_RECORDER_BUFFER_ALLOCATION . . . 63
2.17.1.14 TRC_CFG_RECORDER_MODE . . . 63
2.17.1.15 TRC_CFG_SCHEDULING_ONLY . . . 63
2.18 trcHardwarePort.h File Reference . . . 64
2.18.1 Macro Definition Documentation . . . 64
2.18.1.1 TRC_HWTC_COUNT . . . 64
2.18.1.2 TRC_HWTC_DIVISOR . . . 64
2.18.1.3 TRC_HWTC_FREQ_HZ . . . 64
2.18.1.4 TRC_HWTC_PERIOD . . . 64
2.18.1.5 TRC_HWTC_TYPE . . . 65
2.18.1.6 TRC_IRQ_PRIORITY_ORDER . . . 65
2.18.1.7 TRC_PORT_SPECIFIC_INIT . . . 65
2.19 trcKernelPort.c File Reference . . . 65
2.20 trcKernelPort.h File Reference . . . 65
2.20.1 Macro Definition Documentation . . . 66
2.20.1.1 FREERTOS_VERSION_NOT_SET . . . 66
2.20.1.2 prvGetStreamBufferType . . . 66
2.20.1.3 STRING_CAST . . . 66
2.20.1.4 TickType . . . 66
2.20.1.5 TRC_FREERTOS_VERSION_10_0_0 . . . 66
2.20.1.6 TRC_FREERTOS_VERSION_7_3 . . . 66
2.20.1.7 TRC_FREERTOS_VERSION_7_4 . . . 66
2.20.1.8 TRC_FREERTOS_VERSION_7_5_OR_7_6 . . . 67
2.20.1.9 TRC_FREERTOS_VERSION_8_X . . . 67
2.20.1.10 TRC_FREERTOS_VERSION_9_0_0 . . . 67
2.20.1.11 TRC_FREERTOS_VERSION_9_0_1 . . . 67
2.20.1.12 TRC_FREERTOS_VERSION_9_0_2 . . . 67
2.20.1.13 TRC_FREERTOS_VERSION_9_X . . . 67
2.20.1.14 TRC_USE_TRACEALYZER_RECORDER . . . 67
2.20.1.15 vTraceSetEventGroupName . . . 68
2.20.1.16 vTraceSetMessageBufferName . . . 68

2.20.1.17 vTraceSetMutexName . . . 68
2.20.1.18 vTraceSetQueueName . . . 68
2.20.1.19 vTraceSetSemaphoreName . . . 68
2.20.1.20 vTraceSetStreamBufferName . . . 68
2.21 trcPortDefines.h File Reference . . . 69
2.21.1 Macro Definition Documentation . . . 70
2.21.1.1 CMD_LAST_COMMAND . . . 70
2.21.1.2 CMD_SET_ACTIVE . . . 70
2.21.1.3 FilterGroup0 . . . 70
2.21.1.4 FilterGroup1 . . . 70
2.21.1.5 FilterGroup10 . . . 70
2.21.1.6 FilterGroup11 . . . 70
2.21.1.7 FilterGroup12 . . . 70
2.21.1.8 FilterGroup13 . . . 71
2.21.1.9 FilterGroup14 . . . 71
2.21.1.10 FilterGroup15 . . . 71
2.21.1.11 FilterGroup2 . . . 71
2.21.1.12 FilterGroup3 . . . 71
2.21.1.13 FilterGroup4 . . . 71
2.21.1.14 FilterGroup5 . . . 71
2.21.1.15 FilterGroup6 . . . 71
2.21.1.16 FilterGroup7 . . . 72
2.21.1.17 FilterGroup8 . . . 72
2.21.1.18 FilterGroup9 . . . 72
2.21.1.19 TRC_CUSTOM_TIMER_DECR . . . 72
2.21.1.20 TRC_CUSTOM_TIMER_INCR . . . 72
2.21.1.21 TRC_FREE_RUNNING_32BIT_DECR . . . 72
2.21.1.22 TRC_FREE_RUNNING_32BIT_INCR . . . 72
2.21.1.23 TRC_HARDWARE_PORT_Altera_NiosII . . . 72
2.21.1.24 TRC_HARDWARE_PORT_APPLICATION_DEFINED . . . 73
2.21.1.25 TRC_HARDWARE_PORT_ARM_CORTEX_A9 . . . 73
2.21.1.26 TRC_HARDWARE_PORT_ARM_Cortex_M . . . 73
2.21.1.27 TRC_HARDWARE_PORT_Atmel_AT91SAM7 . . . 73
2.21.1.28 TRC_HARDWARE_PORT_Atmel_UC3A0 . . . 73
2.21.1.29 TRC_HARDWARE_PORT_HWIndependent . . . 73
2.21.1.30 TRC_HARDWARE_PORT_MICROCHIP_PIC24_PIC32 . . . 73
2.21.1.31 TRC_HARDWARE_PORT_NOT_SET . . . 73
2.21.1.32 TRC_HARDWARE_PORT_NXP_LPC210X . . . 74
2.21.1.33 TRC_HARDWARE_PORT_POWERPC_Z4 . . . 74
2.21.1.34 TRC_HARDWARE_PORT_Renesas_RX600 . . . 74
2.21.1.35 TRC_HARDWARE_PORT_TEXAS_INSTRUMENTS_MSP430 . . . 74
2.21.1.36 TRC_HARDWARE_PORT_TEXAS_INSTRUMENTS_TMS570_RM48 . . . 74

2.21.1.37 TRC_HARDWARE_PORT_Win32 . . . 74
2.21.1.38 TRC_HARDWARE_PORT_XILINX_MICROBLAZE . . . 74
2.21.1.39 TRC_HARDWARE_PORT_XILINX_PPC405 . . . 74
2.21.1.40 TRC_HARDWARE_PORT_XILINX_PPC440 . . . 75
2.21.1.41 TRC_INIT . . . 75
2.21.1.42 TRC_OS_TIMER_DECR . . . 75
2.21.1.43 TRC_OS_TIMER_INCR . . . 75
2.21.1.44 TRC_RECORDER_BUFFER_ALLOCATION_CUSTOM . . . 75
2.21.1.45 TRC_RECORDER_BUFFER_ALLOCATION_DYNAMIC . . . 75
2.21.1.46 TRC_RECORDER_BUFFER_ALLOCATION_STATIC . . . 75
2.21.1.47 TRC_RECORDER_MODE_SNAPSHOT . . . 75
2.21.1.48 TRC_RECORDER_MODE_STREAMING . . . 76
2.21.1.49 TRC_START . . . 76
2.21.1.50 TRC_START_AWAIT_HOST . . . 76
2.22 trcRecorder.h File Reference . . . 76
2.22.1 Macro Definition Documentation . . . 77
2.22.1.1 prvTraceSetReadyEventsEnabled . . . 77
2.22.1.2 TRC_ALLOC_CUSTOM_BUFFER . . . 77
2.22.1.3 uiTraceStart . . . 77
2.22.1.4 vTraceChannelPrint . . . 77
2.22.1.5 vTraceConsoleChannelPrintF [1/2] . . . 77
2.22.1.6 vTraceConsoleChannelPrintF [2/2] . . . 78
2.22.1.7 vTraceEnable . . . 78
2.22.1.8 vTraceExcludeTask . . . 78
2.22.1.9 vTraceInstanceFinishedNext . . . 78
2.22.1.10 vTraceInstanceFinishedNow . . . 78
2.22.1.11 vTracePrint . . . 78
2.22.1.12 vTracePrintF . . . 78
2.22.1.13 vTraceSetFilterGroup . . . 79
2.22.1.14 vTraceSetFilterMask . . . 79
2.22.1.15 vTraceSetRecorderDataBuffer . . . 79
2.22.1.16 vTraceSetStopHook . . . 79
2.22.1.17 vTraceStart . . . 79
2.22.1.18 vTraceStop . . . 79
2.22.1.19 vTraceStoreISRBegin . . . 79
2.22.1.20 vTraceStoreISREnd . . . 80
2.22.1.21 vTraceStoreKernelObjectName . . . 80
2.22.1.22 vTraceUBData . . . 80
2.22.1.23 xTraceIsRecordingEnabled . . . 80
2.22.1.24 xTraceRegisterChannelFormat . . . 80
2.22.1.25 xTraceRegisterString . . . 80
2.22.1.26 xTraceSetISRProperties . . . 80

2.22.2 Typedef Documentation . . . 81
2.22.2.1 traceHandle . . . 81
2.22.2.2 traceObjectClass . . . 81
2.22.2.3 traceString . . . 81
2.22.2.4 traceUBChannel . . . 81
2.23 trcSnapshotConfig.h File Reference . . . 81
2.23.1 Macro Definition Documentation . . . 82
2.23.1.1 TRC_CFG_EVENT_BUFFER_SIZE . . . 82
2.23.1.2 TRC_CFG_HEAP_SIZE_BELOW_16M . . . 82
2.23.1.3 TRC_CFG_INCLUDE_FLOAT_SUPPORT . . . 82
2.23.1.4 TRC_CFG_ISR_TAILCHAINING_THRESHOLD . . . 82
2.23.1.5 TRC_CFG_NAME_LEN_EVENTGROUP . . . 83
2.23.1.6 TRC_CFG_NAME_LEN_ISR . . . 83
2.23.1.7 TRC_CFG_NAME_LEN_MESSAGEBUFFER . . . 83
2.23.1.8 TRC_CFG_NAME_LEN_MUTEX . . . 83
2.23.1.9 TRC_CFG_NAME_LEN_QUEUE . . . 83
2.23.1.10 TRC_CFG_NAME_LEN_SEMAPHORE . . . 83
2.23.1.11 TRC_CFG_NAME_LEN_STREAMBUFFER . . . 83
2.23.1.12 TRC_CFG_NAME_LEN_TASK . . . 83
2.23.1.13 TRC_CFG_NAME_LEN_TIMER . . . 84
2.23.1.14 TRC_CFG_NEVENTGROUP . . . 84
2.23.1.15 TRC_CFG_NISR . . . 84
2.23.1.16 TRC_CFG_NMESSAGEBUFFER . . . 84
2.23.1.17 TRC_CFG_NMUTEX . . . 84
2.23.1.18 TRC_CFG_NQUEUE . . . 84
2.23.1.19 TRC_CFG_NSEMAPHORE . . . 84
2.23.1.20 TRC_CFG_NSTREAMBUFFER . . . 84
2.23.1.21 TRC_CFG_NTASK . . . 85
2.23.1.22 TRC_CFG_NTIMER . . . 85
2.23.1.23 TRC_CFG_SEPARATE_USER_EVENT_BUFFER_SIZE . . . 85
2.23.1.24 TRC_CFG_SNAPSHOT_MODE . . . 85
2.23.1.25 TRC_CFG_SYMBOL_TABLE_SIZE . . . 85
2.23.1.26 TRC_CFG_UB_CHANNELS . . . 85
2.23.1.27 TRC_CFG_USE_16BIT_OBJECT_HANDLES . . . 85
2.23.1.28 TRC_CFG_USE_IMPLICIT_IFE_RULES . . . 85
2.23.1.29 TRC_CFG_USE_SEPARATE_USER_EVENT_BUFFER . . . 86
2.23.1.30 TRC_CFG_USE_TRACE_ASSERT . . . 86
2.23.1.31 TRC_SNAPSHOT_MODE_RING_BUFFER . . . 86
2.23.1.32 TRC_SNAPSHOT_MODE_STOP_WHEN_FULL . . . 86
2.24 trcSnapshotRecorder.c File Reference . . . 86
2.25 trcStreamingConfig.h File Reference . . . 86
2.25.1 Macro Definition Documentation . . . 86

2.25.1.1 TRC_CFG_CTRL_TASK_DELAY 87
2.25.1.2 TRC_CFG_CTRL_TASK_PRIORITY 87
2.25.1.3 TRC_CFG_CTRL_TASK_STACK_SIZE 87
2.25.1.4 TRC_CFG_ISR_TAILCHAINING_THRESHOLD 87
2.25.1.5 TRC_CFG_OBJECT_DATA_SLOTS 87
2.25.1.6 TRC_CFG_PAGED_EVENT_BUFFER_PAGE_COUNT 87
2.25.1.7 TRC_CFG_PAGED_EVENT_BUFFER_PAGE_SIZE 87
2.25.1.8 TRC_CFG_SYMBOL_MAX_LENGTH 87
2.25.1.9 TRC_CFG_SYMBOL_TABLE_SLOTS 87
2.26 trcStreamingRecorder.c File Reference 87

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

CANFunctions.c
CAN program file. Implements CAN1 and CAN2 according to project requirements 3
CANFunctions.h
CAN header file. Defines the bus speed, channel size, and ID masks 8
chipKIT_PRO_MX7.c 14
chipKIT_PRO_MX7.h 15
config_bits.h 22
FreeRTOSConfig.h 22
input_capture.c
Input Capture source file. Allows for use of the IC5 peripheral (along with Timer 3) to measure the average RPS of the motor 28
input_capture.h
Input Capture header file. Defines the buffer length to use for calculating the average RPS . . 31
LCDlib.c
LCD library source file. Gives convenient functions for reading / writing to the LCD over PMP . 32
LCDlib.h
LCD library header file. Provides macros for start / end of lines 36
main.c
Main program file, implements a temperature-based remote motor control program over the CAN network 41
main.h
Main header file. Defines buffer lengths, task priority levels, and event timings 47
PWM_library.c 52
PWM_library.h 54
SMBus_IR.c
IR sensor file. Implements communication with the IR sensor over SMBus 56
SMBus_IR.h
IR sensor file header. Gives defines for baud rates, and settings for interfacing with IR sensor . 58
trcConfig.h 61
trcHardwarePort.h 64
trcKernelPort.c 65
trcKernelPort.h 65
trcPortDefines.h 69
trcRecorder.h 76
trcSnapshotConfig.h 81
trcSnapshotRecorder.c 86
trcStreamingConfig.h 86
trcStreamingRecorder.c 87

Chapter 2

File Documentation

2.1 CANFunctions.c File Reference

CAN program file. Implements CAN1 and CAN2 according to project requirements.

```
#include <plib.h>
#include "GenericTypeDefs.h"
#include "chipKIT_Pro_MX7.h"
#include "CANFunctions.h"
```

Functions

- void initialize_CAN1 (void)

  Initialize the CAN1 module. Channel 0 is the TX buffer, Channel 1 is the RX buffer. Channel 1 is configured for a filter for the CAN2 RTR.
- void initialize_CAN2 (void)

  Initialize the CAN2 module. Channel 0 is the TX buffer, Channel 1 is the RX buffer. Channel 0 is configured for a filter for the CAN2 RTR. Channel 1 is configured for a filter for the CAN1 PWM ID.
- unsigned int CAN1_process_RX (float ∗temperature, float ∗motor_speed, float ∗pwm_setting)

  Processes the receive channel for CAN1. This channel receives the RTR messages from CAN2.
- unsigned int CAN2_process_RX (float ∗desired_pwm_setting)

  Process the receive channel for CAN2. This channel gets data messages from CAN1 that correspond to the desired PWM setting.
- void CAN1_send_TX (float desired_pwm_setting)

  Send the passed desired PWM setting over the TX channel of CAN1.
- void CAN1_send_RTR (void)

  Function to send an RTR request from CAN1 to CAN2.
- void CAN2_refill_RTR_buffer (float temperature, float motor_speed, float pwm_setting)

  Refill the RTR buffer for CAN2 with the passed temperature, motor speed, and current PWM setting.
- void __ISR (_CAN_1_VECTOR, IPL4)

  ISR for all CAN1 events. Triggered only by the RX channel not being empty. This indicates that the RTR from CAN2 has been received. The corresponding global flag is set, and the message can be parsed by CAN1_process_RX().
- void __ISR (_CAN_2_VECTOR, ipl4)

  ISR for all CAN2 events. Triggered only by the RX channel not being empty. This indicates that the desired PWM from CAN1 has been received. The corresponding global flag is set, and the message can be parsed by CAN2_process_RX().

Variables

- static volatile BOOL CAN1_received_flag = FALSE
- static volatile BOOL CAN2_received_flag = FALSE
- BYTE CAN1MessageFifoArea [CAN1_MSG_MEMORY]
- BYTE CAN2MessageFifoArea [CAN2_MSG_MEMORY]

2.1.1 Detailed Description

CAN program file. Implements CAN1 and CAN2 according to project requirements.

Author

Collin Heist

Note

For data-sizes, see: http://dubworks.blogspot.com/2013/08/pic32-variable-type-defs.html

2.1.2 Function Documentation

2.1.2.1 __ISR() [1/2]

```
void __ISR (
            _CAN_1_VECTOR ,
            IPL4  )
```

ISR for all CAN1 events. Triggered only by the RX channel not being empty. This indicates that the RTR from CAN2 has been received. The corresponding global flag is set, and the message can be parsed by CAN1_process_RX().

Parameters

None.	

Returns

None.

2.1.2.2 __ISR() [2/2]

```
void __ISR (
            _CAN_2_VECTOR ,
            ipl4  )
```

ISR for all CAN2 events. Triggered only by the RX channel not being empty. This indicates that the desired PWM from CAN1 has been received. The corresponding global flag is set, and the message can be parsed by CAN2_process_RX().

Parameters

None.	

Returns

None.

2.1.2.3 CAN1_process_RX()

```
unsigned int CAN1_process_RX (
            float * temperature,
            float * motor_speed,
            float * pwm_setting )
```

Processes the receive channel for CAN1. This channel receives the RTR messages from CAN2.

Parameters

out	*temperature*	Floating point temperature as parsed from the RTR message from CAN2.
out	*motor_speed*	Floating point motor speed as parsed from the RTR message from CAN2.
out	*pwm_setting*	Floatign point pwm setting as parsed from the RTR message from CAN2.

Returns

unsigned integer that is either CAN_NO_MESSAGE_RECEIVED or CAN_MESSGE_RECEIVED and indicates whether or not CAN1's RX channel had values in it.

2.1.2.4 CAN1_send_RTR()

```
void CAN1_send_RTR (
            void  )
```

Function to send an RTR request from CAN1 to CAN2.

Parameters

None.	

Returns

None.

2.1.2.5 CAN1_send_TX()

```
void CAN1_send_TX (
            float desired_pwm_setting )
```

Send the passed desired PWM setting over the TX channel of CAN1.

Parameters

in	*desired_out_setting*	The desired PWM setting to send to CAN2.

Returns

None.

2.1.2.6 CAN2_process_RX()

```
unsigned int CAN2_process_RX (
            float * desired_pwm_setting )
```

Process the receive channel for CAN2. This channel gets data messages from CAN1 that correspond to the desired PWM setting.

Parameters

out	*desired_out_setting*	The requested PWM setting as sent by CAN1 to CAN2.

Returns

unsigned integer that is either CAN_NO_MESSAGE_RECEIVED or CAN_MESSGE_RECEIVED and indicates whether or not CAN2's RX channel had values in it.

2.1.2.7 CAN2_refill_RTR_buffer()

```
void CAN2_refill_RTR_buffer (
            float temperature,
            float motor_speed,
            float pwm_setting )
```

Refill the RTR buffer for CAN2 with the passed temperature, motor speed, and current PWM setting.

Parameters

in	*temperature*	The current temperature as read by the IR sensor.
in	*motor_speed*	The current motor speed, as read by the input capture event.
in	*pwm_setting*	The current PWM setting for the motor - determined by the low / high set points.

Returns

None.

2.1.2.8 initialize_CAN1()

```
void initialize_CAN1 (
             void  )
```

Initialize the CAN1 module. Channel 0 is the TX buffer, Channel 1 is the RX buffer. Channel 1 is configured for a filter for the CAN2 RTR.

Parameters

None.	

Returns

None.

2.1.2.9 initialize_CAN2()

```
void initialize_CAN2 (
             void  )
```

Initialize the CAN2 module. Channel 0 is the TX buffer, Channel 1 is the RX buffer. Channel 0 is configured for a filter for the CAN2 RTR. Channel 1 is configured for a filter for the CAN1 PWM ID.

Parameters

None.	

Returns

None.

2.1.3 Variable Documentation

2.1.3.1 CAN1_received_flag

```
volatile BOOL CAN1_received_flag = FALSE  [static]
```

2.1.3.2 CAN1MessageFifoArea

```
BYTE CAN1MessageFifoArea[CAN1_MSG_MEMORY]
```

2.1.3.3 CAN2_received_flag

```
volatile BOOL CAN2_received_flag = FALSE  [static]
```

2.1.3.4 CAN2MessageFifoArea

```
BYTE CAN2MessageFifoArea[CAN2_MSG_MEMORY]
```

2.2 CANFunctions.h File Reference

CAN header file. Defines the bus speed, channel size, and ID masks.

Macros

- #define CAN_BUS_SPEED 250000
- #define CAN1_CHANNELS 2
- #define CAN1_FIFO_BUFFERS 8
- #define CAN1_MSG_BUFF_SIZE 16
- #define CAN1_MSG_MEMORY CAN1_CHANNELS∗CAN1_FIFO_BUFFERS∗CAN1_MSG_BUFF_SIZE
- #define CAN2_CHANNELS 2
- #define CAN2_FIFO_BUFFERS 8
- #define CAN2_MSG_BUFF_SIZE 16
- #define CAN2_MSG_MEMORY CAN2_CHANNELS∗CAN2_FIFO_BUFFERS∗CAN2_MSG_BUFF_SIZE
- #define BYTE_0_MASK 0x00000000FF
- #define BYTE_1_MASK 0x000000FF00
- #define BYTE_2_MASK 0x0000FF0000
- #define BYTE_3_MASK 0x00FF000000
- #define SID_BIT_MASK 0x07FF
- #define EID_BIT_MASK 0x03FFFF
- #define SID_FILTER_MASK 0x07FF
- #define EID_FILTER_MASK 0x01FFFFFF
- #define CAN1_PWM_MESSAGE_ID 0x0204
- #define CAN2_RTR_MESSAGE_ID 0x0201
- #define CAN_NO_MESSAGE_RECEIVED 0
- #define CAN_MESSAGE_RECEIVED 1

Functions

- void initialize_CAN1 (void)

  Initialize the CAN1 module. Channel 0 is the TX buffer, Channel 1 is the RX buffer. Channel 1 is configured for a filter for the CAN2 RTR.
- void initialize_CAN2 (void)

  Initialize the CAN2 module. Channel 0 is the TX buffer, Channel 1 is the RX buffer. Channel 0 is configured for a filter for the CAN2 RTR. Channel 1 is configured for a filter for the CAN1 PWM ID.
- unsigned int CAN1_process_RX (float ∗temperature, float ∗motor_speed, float ∗pwm_setting)

  Processes the receive channel for CAN1. This channel receives the RTR messages from CAN2.
- unsigned int CAN2_process_RX (float ∗desired_pwm_setting)

  Process the receive channel for CAN2. This channel gets data messages from CAN1 that correspond to the desired PWM setting.
- void CAN1_send_TX (float desired_pwm_setting)

  Send the passed desired PWM setting over the TX channel of CAN1.
- void CAN1_send_RTR (void)

  Function to send an RTR request from CAN1 to CAN2.
- void CAN2_refill_RTR_buffer (float temperature, float motor_speed, float pwm_setting)

  Refill the RTR buffer for CAN2 with the passed temperature, motor speed, and current PWM setting.

2.2.1 Detailed Description

CAN header file. Defines the bus speed, channel size, and ID masks.

Author

Collin Heist

2.2.2 Macro Definition Documentation

2.2.2.1 BYTE_0_MASK

```
#define BYTE_0_MASK 0x00000000FF
```

2.2.2.2 BYTE_1_MASK

```
#define BYTE_1_MASK 0x000000FF00
```

2.2.2.3 BYTE_2_MASK

```
#define BYTE_2_MASK 0x0000FF0000
```

2.2.2.4 BYTE_3_MASK

```
#define BYTE_3_MASK 0x00FF000000
```

2.2.2.5 CAN1_CHANNELS

```
#define CAN1_CHANNELS 2
```

2.2.2.6 CAN1_FIFO_BUFFERS

```
#define CAN1_FIFO_BUFFERS 8
```

2.2.2.7 CAN1_MSG_BUFF_SIZE

```
#define CAN1_MSG_BUFF_SIZE 16
```

2.2.2.8 CAN1_MSG_MEMORY

```
#define CAN1_MSG_MEMORY CAN1_CHANNELS*CAN1_FIFO_BUFFERS*CAN1_MSG_BUFF_SIZE
```

2.2.2.9 CAN1_PWM_MESSAGE_ID

```
#define CAN1_PWM_MESSAGE_ID 0x0204
```

2.2.2.10 CAN2_CHANNELS

```
#define CAN2_CHANNELS 2
```

2.2.2.11 CAN2_FIFO_BUFFERS

```
#define CAN2_FIFO_BUFFERS 8
```

2.2.2.12 CAN2_MSG_BUFF_SIZE

```
#define CAN2_MSG_BUFF_SIZE 16
```

2.2.2.13 CAN2_MSG_MEMORY

```
#define CAN2_MSG_MEMORY CAN2_CHANNELS*CAN2_FIFO_BUFFERS*CAN2_MSG_BUFF_SIZE
```

2.2.2.14 CAN2_RTR_MESSAGE_ID

```
#define CAN2_RTR_MESSAGE_ID 0x0201
```

2.2.2.15 CAN_BUS_SPEED

```
#define CAN_BUS_SPEED 250000
```

2.2.2.16 CAN_MESSAGE_RECEIVED

```
#define CAN_MESSAGE_RECEIVED 1
```

2.2.2.17 CAN_NO_MESSAGE_RECEIVED

```
#define CAN_NO_MESSAGE_RECEIVED 0
```

2.2.2.18 EID_BIT_MASK

```
#define EID_BIT_MASK 0x03FFFF
```

2.2.2.19 EID_FILTER_MASK

```
#define EID_FILTER_MASK 0x01FFFFFF
```

2.2.2.20 SID_BIT_MASK

```
#define SID_BIT_MASK 0x07FF
```

2.2.2.21 SID_FILTER_MASK

```
#define SID_FILTER_MASK 0x07FF
```

2.2.3 Function Documentation

2.2.3.1 CAN1_process_RX()

```
unsigned int CAN1_process_RX (
             float * temperature,
             float * motor_speed,
             float * pwm_setting )
```

Processes the receive channel for CAN1. This channel receives the RTR messages from CAN2.

Parameters

out	*temperature*	Floating point temperature as parsed from the RTR message from CAN2.
out	*motor_speed*	Floating point motor speed as parsed from the RTR message from CAN2.
out	*pwm_setting*	Floatign point pwm setting as parsed from the RTR message from CAN2.

Returns

unsigned integer that is either CAN_NO_MESSAGE_RECEIVED or CAN_MESSGE_RECEIVED and indicates whether or not CAN1's RX channel had values in it.

2.2.3.2 CAN1_send_RTR()

```
void CAN1_send_RTR (
             void  )
```

Function to send an RTR request from CAN1 to CAN2.

Parameters

None.	

Returns

None.

2.2.3.3 CAN1_send_TX()

```
void CAN1_send_TX (
            float desired_pwm_setting )
```

Send the passed desired PWM setting over the TX channel of CAN1.

Parameters

in	*desired_out_setting*	The desired PWM setting to send to CAN2.

Returns

None.

2.2.3.4 CAN2_process_RX()

```
unsigned int CAN2_process_RX (
            float * desired_pwm_setting )
```

Process the receive channel for CAN2. This channel gets data messages from CAN1 that correspond to the desired PWM setting.

Parameters

out	*desired_out_setting*	The requested PWM setting as sent by CAN1 to CAN2.

Returns

unsigned integer that is either CAN_NO_MESSAGE_RECEIVED or CAN_MESSGE_RECEIVED and indicates whether or not CAN2's RX channel had values in it.

2.2.3.5 CAN2_refill_RTR_buffer()

```
void CAN2_refill_RTR_buffer (
            float temperature,
            float motor_speed,
            float pwm_setting )
```

Refill the RTR buffer for CAN2 with the passed temperature, motor speed, and current PWM setting.

Parameters

in	*temperature*	The current temperature as read by the IR sensor.
in	*motor_speed*	The current motor speed, as read by the input capture event.
in	*pwm_setting*	The current PWM setting for the motor - determined by the low / high set points.

Returns

None.

2.2.3.6 initialize_CAN1()

```
void initialize_CAN1 (
             void  )
```

Initialize the CAN1 module. Channel 0 is the TX buffer, Channel 1 is the RX buffer. Channel 1 is configured for a filter for the CAN2 RTR.

Parameters

None.	

Returns

None.

2.2.3.7 initialize_CAN2()

```
void initialize_CAN2 (
             void  )
```

Initialize the CAN2 module. Channel 0 is the TX buffer, Channel 1 is the RX buffer. Channel 0 is configured for a filter for the CAN2 RTR. Channel 1 is configured for a filter for the CAN1 PWM ID.

Parameters

None.	

Returns

None.

2.3 chipKIT_PRO_MX7.c File Reference

```
#include <plib.h>
```

```
#include "config_bits.h"
#include "chipKIT_PRO_MX7.h"
```

Functions

- void chipKIT_PRO_MX7_Setup (void)

2.3.1 Function Documentation

2.3.1.1 chipKIT_PRO_MX7_Setup()

```
void chipKIT_PRO_MX7_Setup (
            void  )
```

2.4 chipKIT_PRO_MX7.h File Reference

Macros

- #define BTN1 BIT_6
- #define BTN2 BIT_7
- #define BTN3 BIT_0
- #define LED1 BIT_12
- #define LED2 BIT_13
- #define LED3 BIT_14
- #define LED4 BIT_15
- #define BRD_LEDS (LED1 | LED2 | LED3 | LED4)
- #define LED1_IO(a) LATG.LATG12 = a
- #define LED2_IO(a) LATG.LATG13 = a
- #define LED3_IO(a) LATG.LATG14 = a
- #define LED4_IO(a) LATG.LATG15 = a
- #define LEDA_IO(a) LATB.LATB2 = a
- #define LEDB_IO(a) LATB.LATB3 = a
- #define LEDC_IO(a) LATB.LATB4 = a
- #define LEDD_IO(a) LATB.LATB6 = a
- #define LEDE_IO(a) LATB.LATB7 = a
- #define LEDF_IO(a) LATB.LATB8 = a
- #define LEDG_IO(a) LATB.LATB9 = a
- #define LEDH_IO(a) LATB.LATB10 = a
- #define LEDA BIT_2
- #define LEDB BIT_3
- #define LEDC BIT_4
- #define LEDD BIT_6
- #define LEDE BIT_7
- #define LEDF BIT_8
- #define LEDG BIT_9
- #define LEDH BIT_10

- #define SM1 LEDE
- #define SM2 LEDF
- #define SM3 LEDG
- #define SM4 LEDH
- #define SM_LEDS (LEDA | LEDB | LEDC | LEDD | LEDE | LEDF | LEDG | LEDH)
- #define SM_COILS (LEDE | LEDF | LEDG | LEDH)
- #define XTAL (8000000ul)
- #define GetSystemClock() (80000000ul)
- #define GetCoreClock() (GetSystemClock()/2)
- #define GetPeripheralClock() (GetSystemClock()/8)
- #define SYSTEM_FREQ GetSystemClock()
- #define FPB GetPeripheralClock()
- #define CORE_MS_TICK_RATE GetCoreClock()/1000

Functions

- void chipKIT_PRO_MX7_Setup (void)

2.4.1 Macro Definition Documentation

2.4.1.1 BRD_LEDS

```
#define BRD_LEDS (LED1 | LED2 | LED3 | LED4)
```

2.4.1.2 BTN1

```
#define BTN1 BIT_6
```

2.4.1.3 BTN2

```
#define BTN2 BIT_7
```

2.4.1.4 BTN3

```
#define BTN3 BIT_0
```

2.4.1.5 CORE_MS_TICK_RATE

```
#define CORE_MS_TICK_RATE GetCoreClock()/1000
```

2.4.1.6 FPB

```
#define FPB GetPeripheralClock()
```

2.4.1.7 GetCoreClock

```
#define GetCoreClock( ) (GetSystemClock()/2)
```

2.4.1.8 GetPeripheralClock

```
#define GetPeripheralClock( ) (GetSystemClock()/8)
```

2.4.1.9 GetSystemClock

```
#define GetSystemClock( ) (80000000ul)
```

2.4.1.10 LED1

```
#define LED1 BIT_12
```

2.4.1.11 LED1_IO

```
#define LED1_IO(
              a ) LATG.LATG12 = a
```

2.4.1.12 LED2

```
#define LED2 BIT_13
```

2.4.1.13 LED2_IO

```
#define LED2_IO(
              a ) LATG.LATG13 = a
```

2.4.1.14 LED3

```
#define LED3 BIT_14
```

2.4.1.15 LED3_IO

```
#define LED3_IO(
              a ) LATG.LATG14 = a
```

2.4.1.16 LED4

```
#define LED4 BIT_15
```

2.4.1.17 LED4_IO

```
#define LED4_IO(
              a ) LATG.LATG15 = a
```

2.4.1.18 LEDA

```
#define LEDA BIT_2
```

2.4.1.19 LEDA_IO

```
#define LEDA_IO(
              a ) LATB.LATB2 = a
```

2.4.1.20 LEDB

```
#define LEDB BIT_3
```

2.4.1.21 LEDB_IO

```
#define LEDB_IO(
              a ) LATB.LATB3 = a
```

2.4.1.22 LEDC

```
#define LEDC BIT_4
```

2.4.1.23 LEDC_IO

```
#define LEDC_IO(
              a ) LATB.LATB4 = a
```

2.4.1.24 LEDD

```
#define LEDD BIT_6
```

2.4.1.25 LEDD_IO

```
#define LEDD_IO(
              a ) LATB.LATB6 = a
```

2.4.1.26 LEDE

```
#define LEDE BIT_7
```

2.4.1.27 LEDE_IO

```
#define LEDE_IO(
              a ) LATB.LATB7 = a
```

2.4.1.28 LEDF

```
#define LEDF BIT_8
```

2.4.1.29 LEDF_IO

```
#define LEDF_IO(
              a ) LATB.LATB8 = a
```

2.4.1.30 LEDG

```
#define LEDG BIT_9
```

2.4.1.31 LEDG_IO

```
#define LEDG_IO(
              a ) LATB.LATB9 = a
```

2.4.1.32 LEDH

```
#define LEDH BIT_10
```

2.4.1.33 LEDH_IO

```
#define LEDH_IO(
              a ) LATB.LATB10 = a
```

2.4.1.34 SM1

```
#define SM1 LEDE
```

2.4.1.35 SM2

```
#define SM2 LEDF
```

2.4.1.36 SM3

```
#define SM3 LEDG
```

2.4.1.37 SM4

```
#define SM4 LEDH
```

2.4.1.38 SM_COILS

```
#define SM_COILS ( LEDE | LEDF | LEDG | LEDH)
```

2.4.1.39 SM_LEDS

```
#define SM_LEDS (LEDA | LEDB | LEDC | LEDD | LEDE | LEDF | LEDG | LEDH)
```

2.4.1.40 SYSTEM_FREQ

```
#define SYSTEM_FREQ GetSystemClock()
```

2.4.1.41 XTAL

```
#define XTAL (8000000ul)
```

2.4.2 Function Documentation

2.4.2.1 chipKIT_PRO_MX7_Setup()

```
void chipKIT_PRO_MX7_Setup (
            void  )
```

2.5 config_bits.h File Reference

2.6 FreeRTOSConfig.h File Reference

```
#include <p32xxxx.h>
#include "trcRecorder.h"
```

Macros

- #define configUSE_PREEMPTION 1
- #define configUSE_PORT_OPTIMISED_TASK_SELECTION 0
- #define configUSE_IDLE_HOOK 1
- #define configUSE_TICK_HOOK 1
- #define configTICK_RATE_HZ ((TickType_t) 1000)
- #define configCPU_CLOCK_HZ (80000000UL)
- #define configPERIPHERAL_CLOCK_HZ (10000000UL)
- #define configMAX_PRIORITIES (5UL)
- #define configMINIMAL_STACK_SIZE (256)
- #define configISR_STACK_SIZE (512)
- #define configTOTAL_HEAP_SIZE ((size_t) 28000)
- #define configMAX_TASK_NAME_LEN (8)
- #define configUSE_TRACE_FACILITY 1
- #define configUSE_16_BIT_TICKS 0
- #define configIDLE_SHOULD_YIELD 1
- #define configUSE_MUTEXES 1
- #define configCHECK_FOR_STACK_OVERFLOW 2

- #define configQUEUE_REGISTRY_SIZE 0
- #define configUSE_RECURSIVE_MUTEXES 0
- #define configUSE_MALLOC_FAILED_HOOK 0
- #define configUSE_APPLICATION_TASK_TAG 0
- #define configUSE_COUNTING_SEMAPHORES 0
- #define configGENERATE_RUN_TIME_STATS 0
- #define configUSE_CO_ROUTINES 0
- #define configMAX_CO_ROUTINE_PRIORITIES (2)
- #define configUSE_TIMERS 0
- #define configTIMER_TASK_PRIORITY (2)
- #define configTIMER_QUEUE_LENGTH 5
- #define configTIMER_TASK_STACK_DEPTH (configMINIMAL_STACK_SIZE * 2)
- #define INCLUDE_vTaskPrioritySet 1
- #define INCLUDE_uxTaskPriorityGet 1
- #define INCLUDE_vTaskDelete 1
- #define INCLUDE_vTaskCleanUpResources 0
- #define INCLUDE_vTaskSuspend 0
- #define INCLUDE_vTaskDelayUntil 1
- #define INCLUDE_vTaskDelay 1
- #define INCLUDE_uxTaskGetStackHighWaterMark 0
- #define INCLUDE_eTaskGetState 0
- #define configKERNEL_INTERRUPT_PRIORITY 0x01
- #define configMAX_SYSCALL_INTERRUPT_PRIORITY 0x03

2.6.1 Macro Definition Documentation

2.6.1.1 configCHECK_FOR_STACK_OVERFLOW

```
#define configCHECK_FOR_STACK_OVERFLOW 2
```

2.6.1.2 configCPU_CLOCK_HZ

```
#define configCPU_CLOCK_HZ ( 80000000UL )
```

2.6.1.3 configGENERATE_RUN_TIME_STATS

```
#define configGENERATE_RUN_TIME_STATS 0
```

2.6.1.4 configIDLE_SHOULD_YIELD

```
#define configIDLE_SHOULD_YIELD 1
```

2.6.1.5 configISR_STACK_SIZE

```
#define configISR_STACK_SIZE ( 512 )
```

2.6.1.6 configKERNEL_INTERRUPT_PRIORITY

```
#define configKERNEL_INTERRUPT_PRIORITY 0x01
```

2.6.1.7 configMAX_CO_ROUTINE_PRIORITIES

```
#define configMAX_CO_ROUTINE_PRIORITIES ( 2 )
```

2.6.1.8 configMAX_PRIORITIES

```
#define configMAX_PRIORITIES ( 5UL )
```

2.6.1.9 configMAX_SYSCALL_INTERRUPT_PRIORITY

```
#define configMAX_SYSCALL_INTERRUPT_PRIORITY 0x03
```

2.6.1.10 configMAX_TASK_NAME_LEN

```
#define configMAX_TASK_NAME_LEN ( 8 )
```

2.6.1.11 configMINIMAL_STACK_SIZE

```
#define configMINIMAL_STACK_SIZE ( 256 )
```

2.6.1.12 configPERIPHERAL_CLOCK_HZ

```
#define configPERIPHERAL_CLOCK_HZ ( 10000000UL )
```

2.6.1.13 configQUEUE_REGISTRY_SIZE

```
#define configQUEUE_REGISTRY_SIZE 0
```

2.6.1.14 configTICK_RATE_HZ

```
#define configTICK_RATE_HZ ( ( TickType_t ) 1000 )
```

2.6.1.15 configTIMER_QUEUE_LENGTH

```
#define configTIMER_QUEUE_LENGTH 5
```

2.6.1.16 configTIMER_TASK_PRIORITY

```
#define configTIMER_TASK_PRIORITY ( 2 )
```

2.6.1.17 configTIMER_TASK_STACK_DEPTH

```
#define configTIMER_TASK_STACK_DEPTH ( configMINIMAL_STACK_SIZE * 2 )
```

2.6.1.18 configTOTAL_HEAP_SIZE

```
#define configTOTAL_HEAP_SIZE ( ( size_t ) 28000 )
```

2.6.1.19 configUSE_16_BIT_TICKS

```
#define configUSE_16_BIT_TICKS 0
```

2.6.1.20 configUSE_APPLICATION_TASK_TAG

```
#define configUSE_APPLICATION_TASK_TAG 0
```

2.6.1.21 configUSE_CO_ROUTINES

```
#define configUSE_CO_ROUTINES 0
```

2.6.1.22 configUSE_COUNTING_SEMAPHORES

```
#define configUSE_COUNTING_SEMAPHORES 0
```

2.6.1.23 configUSE_IDLE_HOOK

```
#define configUSE_IDLE_HOOK 1
```

2.6.1.24 configUSE_MALLOC_FAILED_HOOK

```
#define configUSE_MALLOC_FAILED_HOOK 0
```

2.6.1.25 configUSE_MUTEXES

```
#define configUSE_MUTEXES 1
```

2.6.1.26 configUSE_PORT_OPTIMISED_TASK_SELECTION

```
#define configUSE_PORT_OPTIMISED_TASK_SELECTION 0
```

2.6.1.27 configUSE_PREEMPTION

```
#define configUSE_PREEMPTION 1
```

2.6.1.28 configUSE_RECURSIVE_MUTEXES

```
#define configUSE_RECURSIVE_MUTEXES 0
```

2.6.1.29 configUSE_TICK_HOOK

```
#define configUSE_TICK_HOOK 1
```

2.6.1.30 configUSE_TIMERS

```
#define configUSE_TIMERS 0
```

2.6.1.31 configUSE_TRACE_FACILITY

```
#define configUSE_TRACE_FACILITY 1
```

2.6.1.32 INCLUDE_eTaskGetState

```
#define INCLUDE_eTaskGetState 0
```

2.6.1.33 INCLUDE_uxTaskGetStackHighWaterMark

```
#define INCLUDE_uxTaskGetStackHighWaterMark 0
```

2.6.1.34 INCLUDE_uxTaskPriorityGet

```
#define INCLUDE_uxTaskPriorityGet 1
```

2.6.1.35 INCLUDE_vTaskCleanUpResources

```
#define INCLUDE_vTaskCleanUpResources 0
```

2.6.1.36 INCLUDE_vTaskDelay

```
#define INCLUDE_vTaskDelay 1
```

2.6.1.37 INCLUDE_vTaskDelayUntil

```
#define INCLUDE_vTaskDelayUntil 1
```

2.6.1.38 INCLUDE_vTaskDelete

```
#define INCLUDE_vTaskDelete 1
```

2.6.1.39 INCLUDE_vTaskPrioritySet

```
#define INCLUDE_vTaskPrioritySet 1
```

2.6.1.40 INCLUDE_vTaskSuspend

```
#define INCLUDE_vTaskSuspend 0
```

2.7 input_capture.c File Reference

Input Capture source file. Allows for use of the IC5 peripheral (along with Timer 3) to measure the average RPS of the motor.

```
#include <plib.h>
#include "input_capture.h"
```

Functions

- void initialize_input_capture (void)

  Function to initialize the input capture peripheral, as well as Timer 3.
- float get_average_rps (void)

  Function to calculate the average RPS from the global rps_buffer.
- void __ISR (_INPUT_CAPTURE_5_VECTOR, IPL3)

  ISR for the input capture module. Adds the latest reading to the rps_buffer.
- void __ISR (_TIMER_3_VECTOR, IPL2)

  ISR for Timer 3.

Variables

- static float rps_buffer [SPEED_BUFFER_LEN]

2.7.1 Detailed Description

Input Capture source file. Allows for use of the IC5 peripheral (along with Timer 3) to measure the average RPS of the motor.

Author

Collin Heist

2.7.2 Function Documentation

2.7.2.1 __ISR() [1/2]

```
void __ISR (
             _INPUT_CAPTURE_5_VECTOR ,
             IPL3  )
```

ISR for the input capture module. Adds the latest reading to the rps_buffer.

Parameters

None.	

Returns

None.

2.7.2.2 __ISR() [2/2]

```
void __ISR (
             _TIMER_3_VECTOR ,
             IPL2  )
```

ISR for Timer 3.

Parameters

None.	

Returns

None.

2.7.2.3 get_average_rps()

```
float get_average_rps (
            void  )
```

Function to calculate the average RPS from the global rps_buffer.

Parameters

None.	

Returns

Float that is the average of all values in the global rps_buffer.

2.7.2.4 initialize_input_capture()

```
void initialize_input_capture (
            void  )
```

Function to initialize the input capture peripheral, as well as Timer 3.

Parameters

None.	

Returns

None.

2.7.3 Variable Documentation

2.7.3.1 rps_buffer

```
float rps_buffer[SPEED_BUFFER_LEN]  [static]
```

2.8 input_capture.h File Reference

Input Capture header file. Defines the buffer length to use for calculating the average RPS.

Macros

- #define SPEED_BUFFER_LEN 16

Functions

- void initialize_input_capture (void)

  Function to initialize the input capture peripheral, as well as Timer 3.
- float get_average_rps (void)

  Function to calculate the average RPS from the global rps_buffer.

2.8.1 Detailed Description

Input Capture header file. Defines the buffer length to use for calculating the average RPS.

Author

Collin Heist

2.8.2 Macro Definition Documentation

2.8.2.1 SPEED_BUFFER_LEN

```
#define SPEED_BUFFER_LEN 16
```

2.8.3 Function Documentation

2.8.3.1 get_average_rps()

```
float get_average_rps (
            void  )
```

Function to calculate the average RPS from the global rps_buffer.

Parameters

None.	

Returns

Float that is the average of all values in the global rps_buffer.

2.8.3.2 initialize_input_capture()

```
void initialize_input_capture (
            void  )
```

Function to initialize the input capture peripheral, as well as Timer 3.

Parameters

None.	

Returns

None.

2.9 LCDlib.c File Reference

LCD library source file. Gives convenient functions for reading / writing to the LCD over PMP.

```
#include <plib.h>
#include "LCDlib.h"
```

Functions

- void initialize_LCD (void)

  Initialize the LCD module and PMP peripheral.
- void put_string_LCD (char *char_string)

  Function to write a provded string to the LCD.
- void put_char_LCD (char c)

  Function to place a single character on the LCD. Outlines behavior for 'special' characters (like , and \r).
- static void _write_LCD (int reg, char c)

  Function to write a given byte (char) to the selected register of the LCD.
- void set_cursor_LCD (unsigned int address)

  Function to move the cursor on the LCD.
- void reset_clear_LCD (void)

  Clear the LCD and reset the cursor to the start of line 1.

- unsigned int read_LCD (int address)

  Read the value on the LCD at the specified address.
- static void sw_delay_ms (unsigned int ms)

  Implements a software delay with resolution in milliseconds.
- static void sw_delay_us (unsigned int us)

  Implements a software delay with resolution in microseconds.

2.9.1 Detailed Description

LCD library source file. Gives convenient functions for reading / writing to the LCD over PMP.

Author

Collin Heist

2.9.2 Function Documentation

2.9.2.1 _write_LCD()

```
static void _write_LCD (
            int reg,
            char c )  [static]
```

Function to write a given byte (char) to the selected register of the LCD.

Parameters

in	*reg*	What register to select with the PMP peripheral.
in	*c*	What byte (character) to write to the LCD.

Returns

None. @notes The existing software delays are (for some reason) necessary.

2.9.2.2 initialize_LCD()

```
void initialize_LCD (
            void  )
```

Initialize the LCD module and PMP peripheral.

Parameters

None.	

Returns

None.

2.9.2.3 put_char_LCD()

```
void put_char_LCD (
            char c )
```

Function to place a single character on the LCD. Outlines behavior for 'special' characters (like , and \r).

Parameters

in	*c*	Character to be written to the LCD.

Returns

None.

2.9.2.4 put_string_LCD()

```
void put_string_LCD (
            char * char_string )
```

Function to write a provded string to the LCD.

Parameters

in	*char_string*	Character string that will be written to the LCD (must be null-terminated).

Returns

None.

2.9.2.5 read_LCD()

```
unsigned int read_LCD (
            int address )
```

Read the value on the LCD at the specified address.

Parameters

in	*What*	address to read from the LCD at.

Returns

unsigned int that represents the result of the PMP read.

2.9.2.6 reset_clear_LCD()

```
void reset_clear_LCD (
            void  )
```

Clear the LCD and reset the cursor to the start of line 1.

Parameters

None.	

Returns

None.

2.9.2.7 set_cursor_LCD()

```
void set_cursor_LCD (
            unsigned int address )
```

Function to move the cursor on the LCD.

Parameters

in	*address*	What address (location) to move the cursor to on the LCD.

Returns

None.

2.9.2.8 sw_delay_ms()

```
static void sw_delay_ms (
            unsigned int ms )  [static]
```

Implements a software delay with resolution in milliseconds.

Parameters

in	*ms*	How many milliseconds to wait

Returns

None. @notes This function uses the value of COUNTS_PER_MS as defined in the header file to implement the delay.

2.9.2.9 sw_delay_us()

```
static void sw_delay_us (
            unsigned int us )  [static]
```

Implements a software delay with resolution in microseconds.

Parameters

in	*us*	How many microseconds to wait

Returns

None.

2.10 LCDlib.h File Reference

LCD library header file. Provides macros for start / end of lines.

Macros

- #define COUNTS_PER_MS 8890
- #define FIRST_LINE_START 0x0000
- #define FIRST_LINE_END 0x000F
- #define SECOND_LINE_START 0x0040
- #define SECOND_LINE_END 0x004F
- #define LCD_RS_CMD 0
- #define LCD_RS_DATA 1

Functions

- void initialize_LCD ()

  Initialize the LCD module and PMP peripheral.
- void put_string_LCD (char ∗char_string)

  Function to write a provded string to the LCD.
- void put_char_LCD (char c)

Function to place a single character on the LCD. Outlines behavior for 'special' characters (like , and \r).

- static void _write_LCD (int reg, char c)
- void set_cursor_LCD (unsigned int address)

  Function to move the cursor on the LCD.
- void reset_clear_LCD ()

  Clear the LCD and reset the cursor to the start of line 1.
- unsigned int read_LCD (int address)

  Read the value on the LCD at the specified address.
- static void sw_delay_ms (unsigned int ms)
- static void sw_delay_us (unsigned int us)

2.10.1 Detailed Description

LCD library header file. Provides macros for start / end of lines.

Author

Collin Heist

2.10.2 Macro Definition Documentation

2.10.2.1 COUNTS_PER_MS

```
#define COUNTS_PER_MS 8890
```

2.10.2.2 FIRST_LINE_END

```
#define FIRST_LINE_END 0x000F
```

2.10.2.3 FIRST_LINE_START

```
#define FIRST_LINE_START 0x0000
```

2.10.2.4 LCD_RS_CMD

```
#define LCD_RS_CMD 0
```

2.10.2.5 LCD_RS_DATA

```
#define LCD_RS_DATA 1
```

2.10.2.6 SECOND_LINE_END

```
#define SECOND_LINE_END 0x004F
```

2.10.2.7 SECOND_LINE_START

```
#define SECOND_LINE_START 0x0040
```

2.10.3 Function Documentation

2.10.3.1 _write_LCD()

```
static void _write_LCD (
            int reg,
            char c )  [static]
```

2.10.3.2 initialize_LCD()

```
void initialize_LCD (
            void  )
```

Initialize the LCD module and PMP peripheral.

Parameters

None.	

Returns

None.

2.10.3.3 put_char_LCD()

```
void put_char_LCD (
            char c )
```

Function to place a single character on the LCD. Outlines behavior for 'special' characters (like
, and \r).

Parameters

in	*c*	Character to be written to the LCD.

Returns

None.

2.10.3.4 put_string_LCD()

```
void put_string_LCD (
            char * char_string )
```

Function to write a provded string to the LCD.

Parameters

in	*char_string*	Character string that will be written to the LCD (must be null-terminated).

Returns

None.

2.10.3.5 read_LCD()

```
unsigned int read_LCD (
            int address )
```

Read the value on the LCD at the specified address.

Parameters

in	*What*	address to read from the LCD at.

Returns

unsigned int that represents the result of the PMP read.

2.10.3.6 reset_clear_LCD()

```
void reset_clear_LCD (
            void  )
```

Clear the LCD and reset the cursor to the start of line 1.

Parameters

None.	

Returns

None.

2.10.3.7 set_cursor_LCD()

```
void set_cursor_LCD (
            unsigned int address )
```

Function to move the cursor on the LCD.

Parameters

in	*address*	What address (location) to move the cursor to on the LCD.

Returns

None.

2.10.3.8 sw_delay_ms()

```
static void sw_delay_ms (
            unsigned int ms )  [static]
```

2.10.3.9 sw_delay_us()

```
static void sw_delay_us (
            unsigned int us )  [static]
```

2.11 main.c File Reference

Main program file, implements a temperature-based remote motor control program over the CAN network.

```
#include <plib.h>
#include "chipKIT_Pro_MX7.h"
#include "FreeRTOS.h"
#include "FreeRTOSConfig.h"
#include "task.h"
#include "main.h"
#include "LCDlib.h"
#include "SMBus_IR.h"
#include "CANFunctions.h"
#include "PWM_library.h"
#include "input_capture.h"
```

Functions

- void __ISR (_CHANGE_NOTICE_VECTOR, IPL2)
- int main ()
- static void initialize_hardware (void)

  Initialize the hardware resources required for this project.
- static unsigned int create_RTOS_objects (void)

  Create all the FreeRTOS objects for this project.
- static unsigned int create_tasks ()

  Create all the FreeRTOS tasks for this project.
- void isr_change_notice_handler ()

  Change Notice ISR wrapper. Unblocks the CN handler task.
- static void task_read_IO (void *task_params)

  FreeRTOS task that reads the temperature and motor speed buffer.
- static void task_send_RTR (void *task_params)

  FreeRTOS task that sends an RTR from CAN1 to CAN2 every RTR_FREQ_MS milliseconds.
- static void task_change_notice_handler (void *task_params)

  Change Notice Handler task. Changes the current state on presses of BTN1.
- static void task_control_FSM (void *task_params)

  FreeRTOS task that implements the two control unit modes.
- static void task_update_pwm (void *task_params)

  FreeRTOS task that processes the CAN2 RX channel for new PWM settings.
- static void clear_string_buffer (char *buffer, unsigned int buffer_length)

  Function to clear the contents of a given string.
- void vApplicationTickHook (void)
- void vApplicationIdleHook (void)
- void vApplicationStackOverflowHook ()
- void _general_exception_handler (unsigned long ulCause, unsigned long ulStatus)

Variables

- current_state = CONFIGURATION_MODE
- float latest_temp
- float latest_rps
- float latest_pwm_setting
- unsigned int previous_BTN1_status
- traceString trace_IO
- traceString trace_RTR
- traceString trace_CN
- traceString trace_control_fsm

2.11.1 Detailed Description

Main program file, implements a temperature-based remote motor control program over the CAN network.

Author

Collin Heist

2.11.2 Function Documentation

2.11.2.1 __ISR()

```
void __ISR (
            _CHANGE_NOTICE_VECTOR ,
            IPL2  )
```

2.11.2.2 _general_exception_handler()

```
void _general_exception_handler (
            unsigned long ulCause,
            unsigned long ulStatus )
```

2.11.2.3 clear_string_buffer()

```
static void clear_string_buffer (
            char * buffer,
            unsigned int buffer_length )  [static]
```

Function to clear the contents of a given string.

Parameters

in	*buffer*	Pointer to the character array being cleared.
in	*buffer_length*	Length of the buffer (how many characters to clear).

Returns

None.

2.11.2.4 create_RTOS_objects()

```
static unsigned int create_RTOS_objects (
            void  )  [static]
```

Create all the FreeRTOS objects for this project.

Parameters

None.	

Returns

Boolean flag (TRUE or FALSE) if there was an error or not.

2.11.2.5 create_tasks()

```
static unsigned int create_tasks ( )  [static]
```

Create all the FreeRTOS tasks for this project.

Parameters

None.	

Returns

Boolean flag (TRUE or FALSE) if there was an error or not.

2.11.2.6 initialize_hardware()

```
static void initialize_hardware (
            void  )  [static]
```

Initialize the hardware resources required for this project.

Parameters

None.	

Returns

None.

2.11.2.7 isr_change_notice_handler()

```
void isr_change_notice_handler (
            void  )
```

Change Notice ISR wrapper. Unblocks the CN handler task.

Parameters

None.	

Returns

None.

2.11.2.8 main()

```
int main ( )
```

2.11.2.9 task_change_notice_handler()

```
static void task_change_notice_handler (
            void * task_params )  [static]
```

Change Notice Handler task. Changes the current state on presses of BTN1.

Parameters

in	*task_params*	Void Pointer that contains the parameters for this task - not used.

Returns

None.

2.11.2.10 task_control_FSM()

```
static void task_control_FSM (
            void * task_params )  [static]
```

FreeRTOS task that implements the two control unit modes.

Parameters

in	*task_params*	Void Pointer that contains the parameters for this task - not used.

Returns

None.

2.11.2.11 task_read_IO()

```
static void task_read_IO (
            void * task_params )  [static]
```

FreeRTOS task that reads the temperature and motor speed buffer.

Parameters

in	*task_params*	Void Pointer that contains the parameters for this task - not used.

Returns

None.

2.11.2.12 task_send_RTR()

```
static void task_send_RTR (
            void * task_params )  [static]
```

FreeRTOS task that sends an RTR from CAN1 to CAN2 every RTR_FREQ_MS milliseconds.

Parameters

in	*task_params*	Void Pointer that contains the parameters for this task - not used.

Returns

None.

2.11.2.13 task_update_pwm()

```
static void task_update_pwm (
            void * task_params )  [static]
```

FreeRTOS task that processes the CAN2 RX channel for new PWM settings.

Parameters

in	*task_params*	Void Pointer that contains the parameters for this task - not used.

Returns

None.

2.11.2.14 vApplicationIdleHook()

```
void vApplicationIdleHook (
             void  )
```

2.11.2.15 vApplicationStackOverflowHook()

```
void vApplicationStackOverflowHook ( )
```

2.11.2.16 vApplicationTickHook()

```
void vApplicationTickHook (
             void  )
```

2.11.3 Variable Documentation

2.11.3.1 current_state

```
current_state = CONFIGURATION_MODE
```

2.11.3.2 latest_pwm_setting

```
float latest_pwm_setting
```

2.11.3.3 latest_rps

```
float latest_rps
```

2.11.3.4 latest_temp

```
float latest_temp
```

2.11.3.5 previous_BTN1_status

```
unsigned int previous_BTN1_status
```

2.11.3.6 trace_CN

```
traceString trace_CN
```

2.11.3.7 trace_control_fsm

```
traceString trace_control_fsm
```

2.11.3.8 trace_IO

```
traceString trace_IO
```

2.11.3.9 trace_RTR

```
traceString trace_RTR
```

2.12 main.h File Reference

Main header file. Defines buffer lengths, task priority levels, and event timings.

Macros

- #define MS_TO_TICKS(ms) (ms / portTICK_RATE_MS)
- #define TASK_READ_IO_PRIORITY 3
- #define TASK_SEND_RTR_PRIORITY 2
- #define TASK_CHANGE_NOTICE_PRIORITY 3
- #define TASK_CONTROL_FSM_PRIORITY 2
- #define TASK_UPDATE_PWM_PRIORITY 2
- #define IO_FREQ_MS 500
- #define RTR_FREQ_MS 2000
- #define DEBOUNCE_MS 25
- #define PWM_FREQUENCY_HZ 1000
- #define PWM_MIN_VAL 20.0
- #define PWM_MAX_VAL 95.0
- #define PWM_LINEAR_MIN 30.0
- #define PWM_LINEAR_MAX 85.0
- #define FALSE 0
- #define TRUE 1

Functions

- static void initialize_hardware ()
- static unsigned int create_RTOS_objects ()
- static unsigned int create_tasks ()
- static void task_read_IO (void *task_params)
- static void task_send_RTR (void *task_params)
- void isr_change_notice_handler (void)

  Change Notice ISR wrapper. Unblocks the CN handler task.
- static void task_change_notice_handler (void *task_params)
- static void task_control_FSM (void *task_params)
- static void task_update_pwm (void *task_params)
- static float average_rps_calculator (void)
- static void clear_string_buffer (char *buffer, unsigned int buffer_length)

2.12.1 Detailed Description

Main header file. Defines buffer lengths, task priority levels, and event timings.

Author

Collin Heist

2.12.2 Macro Definition Documentation

2.12.2.1 DEBOUNCE_MS

```
#define DEBOUNCE_MS 25
```

2.12.2.2 FALSE

```
#define FALSE 0
```

2.12.2.3 IO_FREQ_MS

```
#define IO_FREQ_MS 500
```

2.12.2.4 MS_TO_TICKS

```
#define MS_TO_TICKS(
             ms ) (ms / portTICK_RATE_MS)
```

2.12.2.5 PWM_FREQUENCY_HZ

```
#define PWM_FREQUENCY_HZ 1000
```

2.12.2.6 PWM_LINEAR_MAX

```
#define PWM_LINEAR_MAX 85.0
```

2.12.2.7 PWM_LINEAR_MIN

```
#define PWM_LINEAR_MIN 30.0
```

2.12.2.8 PWM_MAX_VAL

```
#define PWM_MAX_VAL 95.0
```

2.12.2.9 PWM_MIN_VAL

```
#define PWM_MIN_VAL 20.0
```

2.12.2.10 RTR_FREQ_MS

```
#define RTR_FREQ_MS 2000
```

2.12.2.11 TASK_CHANGE_NOTICE_PRIORITY

```
#define TASK_CHANGE_NOTICE_PRIORITY 3
```

2.12.2.12 TASK_CONTROL_FSM_PRIORITY

```
#define TASK_CONTROL_FSM_PRIORITY 2
```

2.12.2.13 TASK_READ_IO_PRIORITY

```
#define TASK_READ_IO_PRIORITY 3
```

2.12.2.14 TASK_SEND_RTR_PRIORITY

```
#define TASK_SEND_RTR_PRIORITY 2
```

2.12.2.15 TASK_UPDATE_PWM_PRIORITY

```
#define TASK_UPDATE_PWM_PRIORITY 2
```

2.12.2.16 TRUE

```
#define TRUE 1
```

2.12.3 Function Documentation

2.12.3.1 average_rps_calculator()

```
static float average_rps_calculator (
            void  )  [static]
```

2.12.3.2 clear_string_buffer()

```
static void clear_string_buffer (
            char * buffer,
            unsigned int buffer_length )  [static]
```

2.12.3.3 create_RTOS_objects()

```
static unsigned int create_RTOS_objects ( )  [static]
```

2.12.3.4 create_tasks()

```
static unsigned int create_tasks ( )  [static]
```

2.12.3.5 initialize_hardware()

```
static void initialize_hardware ( )  [static]
```

2.12.3.6 isr_change_notice_handler()

```
void isr_change_notice_handler (
            void  )
```

Change Notice ISR wrapper. Unblocks the CN handler task.

Parameters

None.	

Returns

None.

2.12.3.7 task_change_notice_handler()

```
static void task_change_notice_handler (
            void * task_params )  [static]
```

2.12.3.8 task_control_FSM()

```
static void task_control_FSM (
            void * task_params )  [static]
```

2.12.3.9 task_read_IO()

```
static void task_read_IO (
            void * task_params )  [static]
```

2.12.3.10 task_send_RTR()

```
static void task_send_RTR (
            void * task_params )  [static]
```

2.12.3.11 task_update_pwm()

```
static void task_update_pwm (
            void * task_params )  [static]
```

2.13 PWM_library.c File Reference

```
#include <plib.h>
#include "chipKIT_Pro_MX7.h"
#include "PWM_library.h"
```

Functions

- unsigned int initialize_pwm (unsigned int duty_cycle, unsigned int pwm_freq)

  Initalize the PWM module for a given starting duty cycle and frequency.
- unsigned int set_pwm (unsigned int duty_cycle)

  Set the PWM output to a given duty cycle (between 0% and 100%).
- void __ISR (_TIMER_2_VECTOR, IPL2)

  ISR for Timer 2 - the PWM timer.

Variables

- static unsigned int t2_tick

2.13.1 Function Documentation

2.13.1.1 __ISR()

```
void __ISR (
             _TIMER_2_VECTOR ,
             IPL2  )
```

ISR for Timer 2 - the PWM timer.

Parameters

None.	

Returns

None.

2.13.1.2 initialize_pwm()

```
unsigned int initialize_pwm (
             unsigned int duty_cycle,
             unsigned int pwm_freq )
```

Initalize the PWM module for a given starting duty cycle and frequency.

Parameters

in	*duty_cycle*	What duty cycle (as a %) to initialize the PWM output to.
in	*pwm_freq*	What frequency to initialize Timer 2 to.

Returns

Boolean flag (TRUE or FALSE) if there was an error or not.

2.13.1.3 set_pwm()

```
unsigned int set_pwm (
            unsigned int duty_cycle )
```

Set the PWM output to a given duty cycle (between 0% and 100%).

Parameters

in	*duty_cycle*	What duty cycle (as a %) to set the PWM output to.

Returns

Boolean flag (TRUE or FALSE) if there was an error or not.

2.13.2 Variable Documentation

2.13.2.1 t2_tick

```
unsigned int t2_tick  [static]
```

2.14 PWM_library.h File Reference

Macros

- #define T2_PRESCALE 1
- #define T2_CLOCK_RATE (FPB / T2_PRESCALE)

Functions

- unsigned int initialize_pwm (unsigned int duty_cycle, unsigned int pwm_freq)

  Initalize the PWM module for a given starting duty cycle and frequency.
- unsigned int set_pwm (unsigned int duty_cycle)

  Set the PWM output to a given duty cycle (between 0% and 100%).

2.14.1 Macro Definition Documentation

2.14.1.1 T2_CLOCK_RATE

```
#define T2_CLOCK_RATE (FPB / T2_PRESCALE)
```

2.14.1.2 T2_PRESCALE

```
#define T2_PRESCALE 1
```

2.14.2 Function Documentation

2.14.2.1 initialize_pwm()

```
unsigned int initialize_pwm (
            unsigned int duty_cycle,
            unsigned int pwm_freq )
```

Initalize the PWM module for a given starting duty cycle and frequency.

Parameters

in	*duty_cycle*	What duty cycle (as a %) to initialize the PWM output to.
in	*pwm_freq*	What frequency to initialize Timer 2 to.

Returns

Boolean flag (TRUE or FALSE) if there was an error or not.

2.14.2.2 set_pwm()

```
unsigned int set_pwm (
            unsigned int duty_cycle )
```

Set the PWM output to a given duty cycle (between 0% and 100%).

Parameters

in	*duty_cycle*	What duty cycle (as a %) to set the PWM output to.

Returns

Boolean flag (TRUE or FALSE) if there was an error or not.

2.15 SMBus_IR.c File Reference

IR sensor file. Implements communication with the IR sensor over SMBus.

```
#include <plib.h>
#include "chipKIT_Pro_MX7.h"
#include "SMBus_IR.h"
```

Functions

- void initialize_ir_sensor (void)

  Initialize the hardware necessary for the IR Sensor (I2C1).
- float read_ir_temp (void)

  Read the temperature of the IR Sensor.
- static void wait_for_ack (unsigned int data_byte)

  Write to I2C1, resending the data until an ACK is received.

2.15.1 Detailed Description

IR sensor file. Implements communication with the IR sensor over SMBus.

Author

Collin Heist

2.15.2 Function Documentation

2.15.2.1 initialize_ir_sensor()

```
void initialize_ir_sensor (
            void  )
```

Initialize the hardware necessary for the IR Sensor (I2C1).

Parameters

None.	

Returns

None.

2.15.2.2 read_ir_temp()

```
float read_ir_temp (
            void  )
```

Read the temperature of the IR Sensor.

Parameters

None.	

Returns

(Float) Temperature listed in T_OBJ_ADDR of IR sensor, after conversion to Celsius.

Start a read by WRITING and selecting the temperature address, and then RESTARTING and sending a read bit. What follows will be the temp + pec.

Combine the just-read data in the correct order ([LSB, MSB, PEC]). Check if the most significant byte is high, indicating an error

The error flag is high, return accordingly

Return the temperature in degC

2.15.2.3 wait_for_ack()

```
static void wait_for_ack (
            unsigned int data_byte )  [static]
```

Write to I2C1, resending the data until an ACK is received.

Parameters

data_byte	is the I2C data byte being sent to I2C1.

Returns

None.

Resend the data byte until an acknowledge is received

ACKSTAT is 0 when the slave acknowledges

2.16 SMBus_IR.h File Reference

IR sensor file header. Gives defines for baud rates, and settings for interfacing with IR sensor.

Macros

- #define BAUD_RATE 100000
- #define I2C_CLOCK_VAL ((FPB / 2 / BAUD_RATE) - 2)
- #define ATTEMPT_COUNT 10
- #define SLAVE_ADDR 0x005A
- #define T_OBJ_ADDR 0x0007
- #define READ_ERROR_FLAG 0x8000
- #define ERROR_TEMP (-1000.0)
- #define IR_SENSOR_RES 0.02
- #define KELVIN_TO_CELSIUS(K) ((float) K - 273.15)
- #define CELCIUS_TO_FARENHEIT(C) ((float) C ∗ 9.0 / 5.0 + 32.0)
- #define WRITE 0
- #define READ 1

Functions

- void initialize_ir_sensor (void)

  Initialize the hardware necessary for the IR Sensor (I2C1).
- float read_ir_temp (void)

  Read the temperature of the IR Sensor.
- static void wait_for_ack (unsigned int data_byte)

2.16.1 Detailed Description

IR sensor file header. Gives defines for baud rates, and settings for interfacing with IR sensor.

Author

Collin Heist

2.16.2 Macro Definition Documentation

2.16.2.1 ATTEMPT_COUNT

```
#define ATTEMPT_COUNT 10
```

2.16.2.2 BAUD_RATE

```
#define BAUD_RATE 100000
```

2.16.2.3 CELCIUS_TO_FARENHEIT

```
#define CELCIUS_TO_FARENHEIT(
             C ) ((float) C * 9.0 / 5.0 + 32.0)
```

2.16.2.4 ERROR_TEMP

```
#define ERROR_TEMP (-1000.0)
```

2.16.2.5 I2C_CLOCK_VAL

```
#define I2C_CLOCK_VAL ((FPB / 2 / BAUD_RATE) - 2)
```

2.16.2.6 IR_SENSOR_RES

```
#define IR_SENSOR_RES 0.02
```

2.16.2.7 KELVIN_TO_CELSIUS

```
#define KELVIN_TO_CELSIUS(
             K ) ((float) K - 273.15)
```

2.16.2.8 READ

```
#define READ 1
```

2.16.2.9 READ_ERROR_FLAG

```
#define READ_ERROR_FLAG 0x8000
```

2.16.2.10 SLAVE_ADDR

```
#define SLAVE_ADDR 0x005A
```

2.16.2.11 T_OBJ_ADDR

```
#define T_OBJ_ADDR 0x0007
```

2.16.2.12 WRITE

```
#define WRITE 0
```

2.16.3 Function Documentation

2.16.3.1 initialize_ir_sensor()

```
void initialize_ir_sensor (
            void  )
```

Initialize the hardware necessary for the IR Sensor (I2C1).

Parameters

None.	

Returns

None.

2.16.3.2 read_ir_temp()

```
float read_ir_temp (
            void  )
```

Read the temperature of the IR Sensor.

Parameters

None.	

Returns

(Float) Temperature listed in T_OBJ_ADDR of IR sensor, after conversion to Celsius.

Start a read by WRITING and selecting the temperature address, and then RESTARTING and sending a read bit. What follows will be the temp + pec.

Combine the just-read data in the correct order ([LSB, MSB, PEC]). Check if the most significant byte is high, indicating an error

The error flag is high, return accordingly

Return the temperature in degC

2.16.3.3 wait_for_ack()

```
static void wait_for_ack (
            unsigned int data_byte )  [static]
```

2.17 trcConfig.h File Reference

```
#include "trcPortDefines.h"
#include "trcSnapshotConfig.h"
```

Macros

- #define TRC_CFG_HARDWARE_PORT TRC_HARDWARE_PORT_MICROCHIP_PIC24_PIC32
- #define TRC_CFG_RECORDER_MODE TRC_RECORDER_MODE_SNAPSHOT
- #define TRC_CFG_FREERTOS_VERSION TRC_FREERTOS_VERSION_8_X
- #define TRC_CFG_SCHEDULING_ONLY 0
- #define TRC_CFG_INCLUDE_MEMMANG_EVENTS 1
- #define TRC_CFG_INCLUDE_USER_EVENTS 1
- #define TRC_CFG_INCLUDE_ISR_TRACING 1
- #define TRC_CFG_INCLUDE_READY_EVENTS 1
- #define TRC_CFG_INCLUDE_OSTICK_EVENTS 1
- #define TRC_CFG_INCLUDE_EVENT_GROUP_EVENTS 0
- #define TRC_CFG_INCLUDE_TIMER_EVENTS 0
- #define TRC_CFG_INCLUDE_PEND_FUNC_CALL_EVENTS 0
- #define TRC_CFG_INCLUDE_STREAM_BUFFER_EVENTS 0
- #define TRC_CFG_RECORDER_BUFFER_ALLOCATION TRC_RECORDER_BUFFER_ALLOCATION_STATIC
- #define TRC_CFG_MAX_ISR_NESTING 8

2.17.1 Macro Definition Documentation

2.17.1.1 TRC_CFG_FREERTOS_VERSION

```
#define TRC_CFG_FREERTOS_VERSION TRC_FREERTOS_VERSION_8_X
```

2.17.1.2 TRC_CFG_HARDWARE_PORT

```
#define TRC_CFG_HARDWARE_PORT TRC_HARDWARE_PORT_MICROCHIP_PIC24_PIC32
```

2.17.1.3 TRC_CFG_INCLUDE_EVENT_GROUP_EVENTS

```
#define TRC_CFG_INCLUDE_EVENT_GROUP_EVENTS 0
```

2.17.1.4 TRC_CFG_INCLUDE_ISR_TRACING

```
#define TRC_CFG_INCLUDE_ISR_TRACING 1
```

2.17.1.5 TRC_CFG_INCLUDE_MEMMANG_EVENTS

```
#define TRC_CFG_INCLUDE_MEMMANG_EVENTS 1
```

2.17.1.6 TRC_CFG_INCLUDE_OSTICK_EVENTS

```
#define TRC_CFG_INCLUDE_OSTICK_EVENTS 1
```

2.17.1.7 TRC_CFG_INCLUDE_PEND_FUNC_CALL_EVENTS

```
#define TRC_CFG_INCLUDE_PEND_FUNC_CALL_EVENTS 0
```

2.17.1.8 TRC_CFG_INCLUDE_READY_EVENTS

```
#define TRC_CFG_INCLUDE_READY_EVENTS 1
```

2.17.1.9 TRC_CFG_INCLUDE_STREAM_BUFFER_EVENTS

```
#define TRC_CFG_INCLUDE_STREAM_BUFFER_EVENTS 0
```

2.17.1.10 TRC_CFG_INCLUDE_TIMER_EVENTS

```
#define TRC_CFG_INCLUDE_TIMER_EVENTS 0
```

2.17.1.11 TRC_CFG_INCLUDE_USER_EVENTS

```
#define TRC_CFG_INCLUDE_USER_EVENTS 1
```

2.17.1.12 TRC_CFG_MAX_ISR_NESTING

```
#define TRC_CFG_MAX_ISR_NESTING 8
```

2.17.1.13 TRC_CFG_RECORDER_BUFFER_ALLOCATION

```
#define TRC_CFG_RECORDER_BUFFER_ALLOCATION TRC_RECORDER_BUFFER_ALLOCATION_STATIC
```

2.17.1.14 TRC_CFG_RECORDER_MODE

```
#define TRC_CFG_RECORDER_MODE TRC_RECORDER_MODE_SNAPSHOT
```

2.17.1.15 TRC_CFG_SCHEDULING_ONLY

```
#define TRC_CFG_SCHEDULING_ONLY 0
```

2.18 trcHardwarePort.h File Reference

```
#include "trcPortDefines.h"
```

Macros

- #define TRC_HWTC_TYPE TRC_OS_TIMER_INCR
- #define TRC_HWTC_COUNT 0
- #define TRC_HWTC_PERIOD 1
- #define TRC_HWTC_DIVISOR 1
- #define TRC_HWTC_FREQ_HZ TRACE_TICK_RATE_HZ
- #define TRC_IRQ_PRIORITY_ORDER NOT_SET
- #define TRC_PORT_SPECIFIC_INIT()

2.18.1 Macro Definition Documentation

2.18.1.1 TRC_HWTC_COUNT

```
#define TRC_HWTC_COUNT 0
```

2.18.1.2 TRC_HWTC_DIVISOR

```
#define TRC_HWTC_DIVISOR 1
```

2.18.1.3 TRC_HWTC_FREQ_HZ

```
#define TRC_HWTC_FREQ_HZ TRACE_TICK_RATE_HZ
```

2.18.1.4 TRC_HWTC_PERIOD

```
#define TRC_HWTC_PERIOD 1
```

2.18.1.5 TRC_HWTC_TYPE

```
#define TRC_HWTC_TYPE TRC_OS_TIMER_INCR
```

2.18.1.6 TRC_IRQ_PRIORITY_ORDER

```
#define TRC_IRQ_PRIORITY_ORDER NOT_SET
```

2.18.1.7 TRC_PORT_SPECIFIC_INIT

```
#define TRC_PORT_SPECIFIC_INIT( )
```

2.19 trcKernelPort.c File Reference

```
#include "FreeRTOS.h"
```

2.20 trcKernelPort.h File Reference

```
#include "FreeRTOS.h"
#include "trcPortDefines.h"
```

Macros

- #define TRC_USE_TRACEALYZER_RECORDER configUSE_TRACE_FACILITY
- #define FREERTOS_VERSION_NOT_SET 0
- #define TRC_FREERTOS_VERSION_7_3 1 /* v7.3 is earliest supported.*/
- #define TRC_FREERTOS_VERSION_7_4 2
- #define TRC_FREERTOS_VERSION_7_5_OR_7_6 3
- #define TRC_FREERTOS_VERSION_8_X 4 /* Any v8.x.x*/
- #define TRC_FREERTOS_VERSION_9_0_0 5
- #define TRC_FREERTOS_VERSION_9_0_1 6
- #define TRC_FREERTOS_VERSION_9_0_2 7
- #define TRC_FREERTOS_VERSION_10_0_0 8 /* If using FreeRTOS v10.0.0 or later version */
- #define TRC_FREERTOS_VERSION_9_X 42 /* Not allowed anymore */
- #define prvGetStreamBufferType(x) 0
- #define STRING_CAST(x) ((signed char*) x)
- #define TickType portTickType
- #define vTraceSetQueueName(object, name)
- #define vTraceSetSemaphoreName(object, name)
- #define vTraceSetMutexName(object, name)
- #define vTraceSetEventGroupName(object, name)
- #define vTraceSetStreamBufferName(object, name)
- #define vTraceSetMessageBufferName(object, name)

2.20.1 Macro Definition Documentation

2.20.1.1 FREERTOS_VERSION_NOT_SET

```
#define FREERTOS_VERSION_NOT_SET 0
```

2.20.1.2 prvGetStreamBufferType

```
#define prvGetStreamBufferType(
              x ) 0
```

2.20.1.3 STRING_CAST

```
#define STRING_CAST(
              x ) ( (signed char*) x )
```

2.20.1.4 TickType

```
#define TickType portTickType
```

2.20.1.5 TRC_FREERTOS_VERSION_10_0_0

```
#define TRC_FREERTOS_VERSION_10_0_0 8 /* If using FreeRTOS v10.0.0 or later version */
```

2.20.1.6 TRC_FREERTOS_VERSION_7_3

```
#define TRC_FREERTOS_VERSION_7_3 1 /* v7.3 is earliest supported.*/
```

2.20.1.7 TRC_FREERTOS_VERSION_7_4

```
#define TRC_FREERTOS_VERSION_7_4 2
```

2.20.1.8 TRC_FREERTOS_VERSION_7_5_OR_7_6

```
#define TRC_FREERTOS_VERSION_7_5_OR_7_6 3
```

2.20.1.9 TRC_FREERTOS_VERSION_8_X

```
#define TRC_FREERTOS_VERSION_8_X 4 /* Any v8.x.x*/
```

2.20.1.10 TRC_FREERTOS_VERSION_9_0_0

```
#define TRC_FREERTOS_VERSION_9_0_0 5
```

2.20.1.11 TRC_FREERTOS_VERSION_9_0_1

```
#define TRC_FREERTOS_VERSION_9_0_1 6
```

2.20.1.12 TRC_FREERTOS_VERSION_9_0_2

```
#define TRC_FREERTOS_VERSION_9_0_2 7
```

2.20.1.13 TRC_FREERTOS_VERSION_9_X

```
#define TRC_FREERTOS_VERSION_9_X 42 /* Not allowed anymore */
```

2.20.1.14 TRC_USE_TRACEALYZER_RECORDER

```
#define TRC_USE_TRACEALYZER_RECORDER configUSE_TRACE_FACILITY
```

2.20.1.15 vTraceSetEventGroupName

```
#define vTraceSetEventGroupName(
              object,
              name )
```

2.20.1.16 vTraceSetMessageBufferName

```
#define vTraceSetMessageBufferName(
              object,
              name )
```

2.20.1.17 vTraceSetMutexName

```
#define vTraceSetMutexName(
              object,
              name )
```

2.20.1.18 vTraceSetQueueName

```
#define vTraceSetQueueName(
              object,
              name )
```

2.20.1.19 vTraceSetSemaphoreName

```
#define vTraceSetSemaphoreName(
              object,
              name )
```

2.20.1.20 vTraceSetStreamBufferName

```
#define vTraceSetStreamBufferName(
              object,
              name )
```

2.21 trcPortDefines.h File Reference

Macros

- #define TRC_FREE_RUNNING_32BIT_INCR 1
- #define TRC_FREE_RUNNING_32BIT_DECR 2
- #define TRC_OS_TIMER_INCR 3
- #define TRC_OS_TIMER_DECR 4
- #define TRC_CUSTOM_TIMER_INCR 5
- #define TRC_CUSTOM_TIMER_DECR 6
- #define TRC_INIT 0
- #define TRC_START 1
- #define TRC_START_AWAIT_HOST 2
- #define CMD_SET_ACTIVE 1 /* Start (param1 = 1) or Stop (param1 = 0) */
- #define CMD_LAST_COMMAND 1
- #define TRC_RECORDER_MODE_SNAPSHOT 0
- #define TRC_RECORDER_MODE_STREAMING 1
- #define TRC_RECORDER_BUFFER_ALLOCATION_STATIC (0x00)
- #define TRC_RECORDER_BUFFER_ALLOCATION_DYNAMIC (0x01)
- #define TRC_RECORDER_BUFFER_ALLOCATION_CUSTOM (0x02)
- #define FilterGroup0 (uint16_t)0x0001
- #define FilterGroup1 (uint16_t)0x0002
- #define FilterGroup2 (uint16_t)0x0004
- #define FilterGroup3 (uint16_t)0x0008
- #define FilterGroup4 (uint16_t)0x0010
- #define FilterGroup5 (uint16_t)0x0020
- #define FilterGroup6 (uint16_t)0x0040
- #define FilterGroup7 (uint16_t)0x0080
- #define FilterGroup8 (uint16_t)0x0100
- #define FilterGroup9 (uint16_t)0x0200
- #define FilterGroup10 (uint16_t)0x0400
- #define FilterGroup11 (uint16_t)0x0800
- #define FilterGroup12 (uint16_t)0x1000
- #define FilterGroup13 (uint16_t)0x2000
- #define FilterGroup14 (uint16_t)0x4000
- #define FilterGroup15 (uint16_t)0x8000
- #define TRC_HARDWARE_PORT_APPLICATION_DEFINED 98 /* - - */
- #define TRC_HARDWARE_PORT_NOT_SET 99 /* - - */
- #define TRC_HARDWARE_PORT_HWIndependent 0 /* Yes Any */
- #define TRC_HARDWARE_PORT_Win32 1 /* Yes FreeRTOS on Win32 */
- #define TRC_HARDWARE_PORT_Atmel_AT91SAM7 2 /* No Any */
- #define TRC_HARDWARE_PORT_Atmel_UC3A0 3 /* No Any */
- #define TRC_HARDWARE_PORT_ARM_Cortex_M 4 /* Yes Any */
- #define TRC_HARDWARE_PORT_Renesas_RX600 6 /* Yes Any */
- #define TRC_HARDWARE_PORT_MICROCHIP_PIC24_PIC32 7 /* Yes Any */
- #define TRC_HARDWARE_PORT_TEXAS_INSTRUMENTS_TMS570_RM48 8 /* Yes Any */
- #define TRC_HARDWARE_PORT_TEXAS_INSTRUMENTS_MSP430 9 /* No Any */
- #define TRC_HARDWARE_PORT_XILINX_PPC405 11 /* No FreeRTOS */
- #define TRC_HARDWARE_PORT_XILINX_PPC440 12 /* No FreeRTOS */
- #define TRC_HARDWARE_PORT_XILINX_MICROBLAZE 13 /* No Any */
- #define TRC_HARDWARE_PORT_NXP_LPC210X 14 /* No Any */
- #define TRC_HARDWARE_PORT_ARM_CORTEX_A9 15 /* Yes Any */
- #define TRC_HARDWARE_PORT_POWERPC_Z4 16 /* No FreeRTOS */
- #define TRC_HARDWARE_PORT_Altera_NiosII 17 /* No Any */

2.21.1 Macro Definition Documentation

2.21.1.1 CMD_LAST_COMMAND

```
#define CMD_LAST_COMMAND 1
```

2.21.1.2 CMD_SET_ACTIVE

```
#define CMD_SET_ACTIVE 1 /* Start (param1 = 1) or Stop (param1 = 0) */
```

2.21.1.3 FilterGroup0

```
#define FilterGroup0 (uint16_t)0x0001
```

2.21.1.4 FilterGroup1

```
#define FilterGroup1 (uint16_t)0x0002
```

2.21.1.5 FilterGroup10

```
#define FilterGroup10 (uint16_t)0x0400
```

2.21.1.6 FilterGroup11

```
#define FilterGroup11 (uint16_t)0x0800
```

2.21.1.7 FilterGroup12

```
#define FilterGroup12 (uint16_t)0x1000
```

2.21.1.8 FilterGroup13

```
#define FilterGroup13 (uint16_t)0x2000
```

2.21.1.9 FilterGroup14

```
#define FilterGroup14 (uint16_t)0x4000
```

2.21.1.10 FilterGroup15

```
#define FilterGroup15 (uint16_t)0x8000
```

2.21.1.11 FilterGroup2

```
#define FilterGroup2 (uint16_t)0x0004
```

2.21.1.12 FilterGroup3

```
#define FilterGroup3 (uint16_t)0x0008
```

2.21.1.13 FilterGroup4

```
#define FilterGroup4 (uint16_t)0x0010
```

2.21.1.14 FilterGroup5

```
#define FilterGroup5 (uint16_t)0x0020
```

2.21.1.15 FilterGroup6

```
#define FilterGroup6 (uint16_t)0x0040
```

2.21.1.16 FilterGroup7

```
#define FilterGroup7 (uint16_t)0x0080
```

2.21.1.17 FilterGroup8

```
#define FilterGroup8 (uint16_t)0x0100
```

2.21.1.18 FilterGroup9

```
#define FilterGroup9 (uint16_t)0x0200
```

2.21.1.19 TRC_CUSTOM_TIMER_DECR

```
#define TRC_CUSTOM_TIMER_DECR 6
```

2.21.1.20 TRC_CUSTOM_TIMER_INCR

```
#define TRC_CUSTOM_TIMER_INCR 5
```

2.21.1.21 TRC_FREE_RUNNING_32BIT_DECR

```
#define TRC_FREE_RUNNING_32BIT_DECR 2
```

2.21.1.22 TRC_FREE_RUNNING_32BIT_INCR

```
#define TRC_FREE_RUNNING_32BIT_INCR 1
```

2.21.1.23 TRC_HARDWARE_PORT_Altera_NiosII

```
#define TRC_HARDWARE_PORT_Altera_NiosII 17 /* No Any */
```

2.21.1.24 TRC_HARDWARE_PORT_APPLICATION_DEFINED

```
#define TRC_HARDWARE_PORT_APPLICATION_DEFINED 98 /* - - */
```

2.21.1.25 TRC_HARDWARE_PORT_ARM_CORTEX_A9

```
#define TRC_HARDWARE_PORT_ARM_CORTEX_A9 15 /* Yes Any */
```

2.21.1.26 TRC_HARDWARE_PORT_ARM_Cortex_M

```
#define TRC_HARDWARE_PORT_ARM_Cortex_M 4 /* Yes Any */
```

2.21.1.27 TRC_HARDWARE_PORT_Atmel_AT91SAM7

```
#define TRC_HARDWARE_PORT_Atmel_AT91SAM7 2 /* No Any */
```

2.21.1.28 TRC_HARDWARE_PORT_Atmel_UC3A0

```
#define TRC_HARDWARE_PORT_Atmel_UC3A0 3 /* No Any */
```

2.21.1.29 TRC_HARDWARE_PORT_HWIndependent

```
#define TRC_HARDWARE_PORT_HWIndependent 0 /* Yes Any */
```

2.21.1.30 TRC_HARDWARE_PORT_MICROCHIP_PIC24_PIC32

```
#define TRC_HARDWARE_PORT_MICROCHIP_PIC24_PIC32 7 /* Yes Any */
```

2.21.1.31 TRC_HARDWARE_PORT_NOT_SET

```
#define TRC_HARDWARE_PORT_NOT_SET 99 /* - - */
```

2.21.1.32 TRC_HARDWARE_PORT_NXP_LPC210X

```
#define TRC_HARDWARE_PORT_NXP_LPC210X 14 /* No Any */
```

2.21.1.33 TRC_HARDWARE_PORT_POWERPC_Z4

```
#define TRC_HARDWARE_PORT_POWERPC_Z4 16 /* No FreeRTOS */
```

2.21.1.34 TRC_HARDWARE_PORT_Renesas_RX600

```
#define TRC_HARDWARE_PORT_Renesas_RX600 6 /* Yes Any */
```

2.21.1.35 TRC_HARDWARE_PORT_TEXAS_INSTRUMENTS_MSP430

```
#define TRC_HARDWARE_PORT_TEXAS_INSTRUMENTS_MSP430 9 /* No Any */
```

2.21.1.36 TRC_HARDWARE_PORT_TEXAS_INSTRUMENTS_TMS570_RM48

```
#define TRC_HARDWARE_PORT_TEXAS_INSTRUMENTS_TMS570_RM48 8 /* Yes Any */
```

2.21.1.37 TRC_HARDWARE_PORT_Win32

```
#define TRC_HARDWARE_PORT_Win32 1 /* Yes FreeRTOS on Win32 */
```

2.21.1.38 TRC_HARDWARE_PORT_XILINX_MICROBLAZE

```
#define TRC_HARDWARE_PORT_XILINX_MICROBLAZE 13 /* No Any */
```

2.21.1.39 TRC_HARDWARE_PORT_XILINX_PPC405

```
#define TRC_HARDWARE_PORT_XILINX_PPC405 11 /* No FreeRTOS */
```

2.21.1.40 TRC_HARDWARE_PORT_XILINX_PPC440

```
#define TRC_HARDWARE_PORT_XILINX_PPC440 12 /* No FreeRTOS */
```

2.21.1.41 TRC_INIT

```
#define TRC_INIT 0
```

2.21.1.42 TRC_OS_TIMER_DECR

```
#define TRC_OS_TIMER_DECR 4
```

2.21.1.43 TRC_OS_TIMER_INCR

```
#define TRC_OS_TIMER_INCR 3
```

2.21.1.44 TRC_RECORDER_BUFFER_ALLOCATION_CUSTOM

```
#define TRC_RECORDER_BUFFER_ALLOCATION_CUSTOM (0x02)
```

2.21.1.45 TRC_RECORDER_BUFFER_ALLOCATION_DYNAMIC

```
#define TRC_RECORDER_BUFFER_ALLOCATION_DYNAMIC (0x01)
```

2.21.1.46 TRC_RECORDER_BUFFER_ALLOCATION_STATIC

```
#define TRC_RECORDER_BUFFER_ALLOCATION_STATIC (0x00)
```

2.21.1.47 TRC_RECORDER_MODE_SNAPSHOT

```
#define TRC_RECORDER_MODE_SNAPSHOT 0
```

2.21.1.48 TRC_RECORDER_MODE_STREAMING

```
#define TRC_RECORDER_MODE_STREAMING 1
```

2.21.1.49 TRC_START

```
#define TRC_START 1
```

2.21.1.50 TRC_START_AWAIT_HOST

```
#define TRC_START_AWAIT_HOST 2
```

2.22 trcRecorder.h File Reference

```
#include <stdint.h>
#include <stddef.h>
#include "trcConfig.h"
#include "trcPortDefines.h"
#include "trcHardwarePort.h"
#include "trcKernelPort.h"
```

Macros

- #define vTraceConsoleChannelPrintF(fmt, ...)
- #define vTraceEnable(x)
- #define xTraceRegisterString(x) 0; (void)x;
- #define vTracePrint(chn, ...) (void)chn
- #define vTracePrintF(chn, ...) (void)chn
- #define vTraceInstanceFinishedNow()
- #define vTraceInstanceFinishedNext()
- #define vTraceStoreISRBegin(x) (void)x
- #define vTraceStoreISREnd(x) (void)x
- #define xTraceSetISRProperties(a, b) 0
- #define vTraceStoreKernelObjectName(a, b)
- #define xTraceRegisterChannelFormat(eventLabel, formatStr) 0
- #define vTraceChannelPrint(label)
- #define vTraceUBData(label, ...)
- #define vTraceSetFilterGroup(x)
- #define vTraceSetFilterMask(x)
- #define prvTraceSetReadyEventsEnabled(status)
- #define vTraceExcludeTask(handle)
- #define uiTraceStart() (1)
- #define vTraceStart()
- #define vTraceStop()
- #define vTraceSetRecorderDataBuffer(pRecorderData)
- #define vTraceConsoleChannelPrintF(fmt, ...)
- #define TRC_ALLOC_CUSTOM_BUFFER(bufname)
- #define xTraceIsRecordingEnabled() (0)
- #define vTraceSetStopHook(x)

Typedefs

- typedef uint16_t traceString
- typedef uint8_t traceUBChannel
- typedef uint8_t traceObjectClass
- typedef uint8_t traceHandle

2.22.1 Macro Definition Documentation

2.22.1.1 prvTraceSetReadyEventsEnabled

```
#define prvTraceSetReadyEventsEnabled(
              status )
```

2.22.1.2 TRC_ALLOC_CUSTOM_BUFFER

```
#define TRC_ALLOC_CUSTOM_BUFFER(
              bufname )
```

2.22.1.3 uiTraceStart

```
#define uiTraceStart( ) (1)
```

2.22.1.4 vTraceChannelPrint

```
#define vTraceChannelPrint(
              label )
```

2.22.1.5 vTraceConsoleChannelPrintF [1/2]

```
#define vTraceConsoleChannelPrintF(
              fmt,
              ...  )
```

2.22.1.6 vTraceConsoleChannelPrintF [2/2]

```
#define vTraceConsoleChannelPrintF(
              fmt,
              ...  )
```

2.22.1.7 vTraceEnable

```
#define vTraceEnable(
              x )
```

2.22.1.8 vTraceExcludeTask

```
#define vTraceExcludeTask(
              handle )
```

2.22.1.9 vTraceInstanceFinishedNext

```
#define vTraceInstanceFinishedNext( )
```

2.22.1.10 vTraceInstanceFinishedNow

```
#define vTraceInstanceFinishedNow( )
```

2.22.1.11 vTracePrint

```
#define vTracePrint(
              chn,
              ...  ) (void)chn
```

2.22.1.12 vTracePrintF

```
#define vTracePrintF(
              chn,
              ...  ) (void)chn
```

2.22.1.13 vTraceSetFilterGroup

```
#define vTraceSetFilterGroup(
              x )
```

2.22.1.14 vTraceSetFilterMask

```
#define vTraceSetFilterMask(
              x )
```

2.22.1.15 vTraceSetRecorderDataBuffer

```
#define vTraceSetRecorderDataBuffer(
              pRecorderData )
```

2.22.1.16 vTraceSetStopHook

```
#define vTraceSetStopHook(
              x )
```

2.22.1.17 vTraceStart

```
#define vTraceStart( )
```

2.22.1.18 vTraceStop

```
#define vTraceStop( )
```

2.22.1.19 vTraceStoreISRBegin

```
#define vTraceStoreISRBegin(
              x ) (void)x
```

2.22.1.20 vTraceStoreISREnd

```
#define vTraceStoreISREnd(
              x ) (void)x
```

2.22.1.21 vTraceStoreKernelObjectName

```
#define vTraceStoreKernelObjectName(
              a,
              b )
```

2.22.1.22 vTraceUBData

```
#define vTraceUBData(
              label,
              ...  )
```

2.22.1.23 xTraceIsRecordingEnabled

```
#define xTraceIsRecordingEnabled( ) (0)
```

2.22.1.24 xTraceRegisterChannelFormat

```
#define xTraceRegisterChannelFormat(
              eventLabel,
              formatStr ) 0
```

2.22.1.25 xTraceRegisterString

```
#define xTraceRegisterString(
              x ) 0; (void)x;
```

2.22.1.26 xTraceSetISRProperties

```
#define xTraceSetISRProperties(
              a,
              b ) 0
```

2.22.2 Typedef Documentation

2.22.2.1 traceHandle

```
typedef uint8_t traceHandle
```

2.22.2.2 traceObjectClass

```
typedef uint8_t traceObjectClass
```

2.22.2.3 traceString

```
typedef uint16_t traceString
```

2.22.2.4 traceUBChannel

```
typedef uint8_t traceUBChannel
```

2.23 trcSnapshotConfig.h File Reference

Macros

- #define TRC_SNAPSHOT_MODE_RING_BUFFER (0x01)
- #define TRC_SNAPSHOT_MODE_STOP_WHEN_FULL (0x02)
- #define TRC_CFG_SNAPSHOT_MODE TRC_SNAPSHOT_MODE_RING_BUFFER
- #define TRC_CFG_EVENT_BUFFER_SIZE 1000
- #define TRC_CFG_NTASK 15
- #define TRC_CFG_NISR 5
- #define TRC_CFG_NQUEUE 10
- #define TRC_CFG_NSEMAPHORE 10
- #define TRC_CFG_NMUTEX 10
- #define TRC_CFG_NTIMER 5
- #define TRC_CFG_NEVENTGROUP 5
- #define TRC_CFG_NSTREAMBUFFER 5
- #define TRC_CFG_NMESSAGEBUFFER 5
- #define TRC_CFG_INCLUDE_FLOAT_SUPPORT 0
- #define TRC_CFG_SYMBOL_TABLE_SIZE 800
- #define TRC_CFG_NAME_LEN_TASK 15
- #define TRC_CFG_NAME_LEN_ISR 15

- #define TRC_CFG_NAME_LEN_QUEUE 15
- #define TRC_CFG_NAME_LEN_SEMAPHORE 15
- #define TRC_CFG_NAME_LEN_MUTEX 15
- #define TRC_CFG_NAME_LEN_TIMER 15
- #define TRC_CFG_NAME_LEN_EVENTGROUP 15
- #define TRC_CFG_NAME_LEN_STREAMBUFFER 15
- #define TRC_CFG_NAME_LEN_MESSAGEBUFFER 15
- #define TRC_CFG_HEAP_SIZE_BELOW_16M 0
- #define TRC_CFG_USE_IMPLICIT_IFE_RULES 1
- #define TRC_CFG_USE_16BIT_OBJECT_HANDLES 0
- #define TRC_CFG_USE_TRACE_ASSERT 1
- #define TRC_CFG_USE_SEPARATE_USER_EVENT_BUFFER 0
- #define TRC_CFG_SEPARATE_USER_EVENT_BUFFER_SIZE 200
- #define TRC_CFG_UB_CHANNELS 32
- #define TRC_CFG_ISR_TAILCHAINING_THRESHOLD 0

2.23.1 Macro Definition Documentation

2.23.1.1 TRC_CFG_EVENT_BUFFER_SIZE

```
#define TRC_CFG_EVENT_BUFFER_SIZE 1000
```

2.23.1.2 TRC_CFG_HEAP_SIZE_BELOW_16M

```
#define TRC_CFG_HEAP_SIZE_BELOW_16M 0
```

2.23.1.3 TRC_CFG_INCLUDE_FLOAT_SUPPORT

```
#define TRC_CFG_INCLUDE_FLOAT_SUPPORT 0
```

2.23.1.4 TRC_CFG_ISR_TAILCHAINING_THRESHOLD

```
#define TRC_CFG_ISR_TAILCHAINING_THRESHOLD 0
```

2.23.1.5 TRC_CFG_NAME_LEN_EVENTGROUP

```
#define TRC_CFG_NAME_LEN_EVENTGROUP 15
```

2.23.1.6 TRC_CFG_NAME_LEN_ISR

```
#define TRC_CFG_NAME_LEN_ISR 15
```

2.23.1.7 TRC_CFG_NAME_LEN_MESSAGEBUFFER

```
#define TRC_CFG_NAME_LEN_MESSAGEBUFFER 15
```

2.23.1.8 TRC_CFG_NAME_LEN_MUTEX

```
#define TRC_CFG_NAME_LEN_MUTEX 15
```

2.23.1.9 TRC_CFG_NAME_LEN_QUEUE

```
#define TRC_CFG_NAME_LEN_QUEUE 15
```

2.23.1.10 TRC_CFG_NAME_LEN_SEMAPHORE

```
#define TRC_CFG_NAME_LEN_SEMAPHORE 15
```

2.23.1.11 TRC_CFG_NAME_LEN_STREAMBUFFER

```
#define TRC_CFG_NAME_LEN_STREAMBUFFER 15
```

2.23.1.12 TRC_CFG_NAME_LEN_TASK

```
#define TRC_CFG_NAME_LEN_TASK 15
```

2.23.1.13 TRC_CFG_NAME_LEN_TIMER

```
#define TRC_CFG_NAME_LEN_TIMER 15
```

2.23.1.14 TRC_CFG_NEVENTGROUP

```
#define TRC_CFG_NEVENTGROUP 5
```

2.23.1.15 TRC_CFG_NISR

```
#define TRC_CFG_NISR 5
```

2.23.1.16 TRC_CFG_NMESSAGEBUFFER

```
#define TRC_CFG_NMESSAGEBUFFER 5
```

2.23.1.17 TRC_CFG_NMUTEX

```
#define TRC_CFG_NMUTEX 10
```

2.23.1.18 TRC_CFG_NQUEUE

```
#define TRC_CFG_NQUEUE 10
```

2.23.1.19 TRC_CFG_NSEMAPHORE

```
#define TRC_CFG_NSEMAPHORE 10
```

2.23.1.20 TRC_CFG_NSTREAMBUFFER

```
#define TRC_CFG_NSTREAMBUFFER 5
```

2.23.1.21 TRC_CFG_NTASK

```
#define TRC_CFG_NTASK 15
```

2.23.1.22 TRC_CFG_NTIMER

```
#define TRC_CFG_NTIMER 5
```

2.23.1.23 TRC_CFG_SEPARATE_USER_EVENT_BUFFER_SIZE

```
#define TRC_CFG_SEPARATE_USER_EVENT_BUFFER_SIZE 200
```

2.23.1.24 TRC_CFG_SNAPSHOT_MODE

```
#define TRC_CFG_SNAPSHOT_MODE TRC_SNAPSHOT_MODE_RING_BUFFER
```

2.23.1.25 TRC_CFG_SYMBOL_TABLE_SIZE

```
#define TRC_CFG_SYMBOL_TABLE_SIZE 800
```

2.23.1.26 TRC_CFG_UB_CHANNELS

```
#define TRC_CFG_UB_CHANNELS 32
```

2.23.1.27 TRC_CFG_USE_16BIT_OBJECT_HANDLES

```
#define TRC_CFG_USE_16BIT_OBJECT_HANDLES 0
```

2.23.1.28 TRC_CFG_USE_IMPLICIT_IFE_RULES

```
#define TRC_CFG_USE_IMPLICIT_IFE_RULES 1
```

2.23.1.29 TRC_CFG_USE_SEPARATE_USER_EVENT_BUFFER

```
#define TRC_CFG_USE_SEPARATE_USER_EVENT_BUFFER 0
```

2.23.1.30 TRC_CFG_USE_TRACE_ASSERT

```
#define TRC_CFG_USE_TRACE_ASSERT 1
```

2.23.1.31 TRC_SNAPSHOT_MODE_RING_BUFFER

```
#define TRC_SNAPSHOT_MODE_RING_BUFFER (0x01)
```

2.23.1.32 TRC_SNAPSHOT_MODE_STOP_WHEN_FULL

```
#define TRC_SNAPSHOT_MODE_STOP_WHEN_FULL (0x02)
```

2.24 trcSnapshotRecorder.c File Reference

```
#include "trcRecorder.h"
```

2.25 trcStreamingConfig.h File Reference

Macros

- #define TRC_CFG_SYMBOL_TABLE_SLOTS 40
- #define TRC_CFG_SYMBOL_MAX_LENGTH 25
- #define TRC_CFG_OBJECT_DATA_SLOTS 40
- #define TRC_CFG_CTRL_TASK_STACK_SIZE (configMINIMAL_STACK_SIZE * 2)
- #define TRC_CFG_CTRL_TASK_PRIORITY 1
- #define TRC_CFG_CTRL_TASK_DELAY ((10 * configTICK_RATE_HZ) / 1000)
- #define TRC_CFG_PAGED_EVENT_BUFFER_PAGE_COUNT 2
- #define TRC_CFG_PAGED_EVENT_BUFFER_PAGE_SIZE 2500
- #define TRC_CFG_ISR_TAILCHAINING_THRESHOLD 0

2.25.1 Macro Definition Documentation

2.25.1.1 TRC_CFG_CTRL_TASK_DELAY

```
#define TRC_CFG_CTRL_TASK_DELAY ((10 * configTICK_RATE_HZ) / 1000)
```

2.25.1.2 TRC_CFG_CTRL_TASK_PRIORITY

```
#define TRC_CFG_CTRL_TASK_PRIORITY 1
```

2.25.1.3 TRC_CFG_CTRL_TASK_STACK_SIZE

```
#define TRC_CFG_CTRL_TASK_STACK_SIZE (configMINIMAL_STACK_SIZE * 2)
```

2.25.1.4 TRC_CFG_ISR_TAILCHAINING_THRESHOLD

```
#define TRC_CFG_ISR_TAILCHAINING_THRESHOLD 0
```

2.25.1.5 TRC_CFG_OBJECT_DATA_SLOTS

```
#define TRC_CFG_OBJECT_DATA_SLOTS 40
```

2.25.1.6 TRC_CFG_PAGED_EVENT_BUFFER_PAGE_COUNT

```
#define TRC_CFG_PAGED_EVENT_BUFFER_PAGE_COUNT 2
```

2.25.1.7 TRC_CFG_PAGED_EVENT_BUFFER_PAGE_SIZE

```
#define TRC_CFG_PAGED_EVENT_BUFFER_PAGE_SIZE 2500
```

2.25.1.8 TRC_CFG_SYMBOL_MAX_LENGTH

```
#define TRC_CFG_SYMBOL_MAX_LENGTH 25
```

2.25.1.9 TRC_CFG_SYMBOL_TABLE_SLOTS

```
#define TRC_CFG_SYMBOL_TABLE_SLOTS 40
```

2.26 trcStreamingRecorder.c File Reference

```
#include "trcRecorder.h"
```