

Temperature Controller

Generated by Doxygen 1.8.11

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	CircularBuffer_t Struct Reference	5
3.1.1	Field Documentation	5
3.1.1.1	bufferEnd	5
3.1.1.2	bufferStart	5
3.1.1.3	head	5
3.1.1.4	itemSize	5
3.1.1.5	numItems	5
3.1.1.6	size	5
3.1.1.7	tail	5
3.2	CommandMessage Struct Reference	6
3.2.1	Field Documentation	6
3.2.1.1	checksum	6
3.2.1.2	cmd	6
3.2.1.3	data	6
3.3	DS8B20_ROMCode Union Reference	6
3.3.1	Member Function Documentation	6
3.3.1.1	__attribute__((__packed__))	6

3.3.2	Field Documentation	6
3.3.2.1	romBytes	6
3.4	DS8B20_Scratchpad Union Reference	7
3.4.1	Member Function Documentation	7
3.4.1.1	__attribute__((__packed__))	7
3.4.2	Field Documentation	7
3.4.2.1	scratchpadBytes	7
3.5	nRF24L01_CONFIG_t Union Reference	7
3.5.1	Field Documentation	8
3.5.1.1	b	8
3.5.1.2	B	8
3.5.1.3	crco	8
3.5.1.4	enCRC	8
3.5.1.5	maskMaxRt	8
3.5.1.6	maskRxDr	8
3.5.1.7	maskTxDs	8
3.5.1.8	primRx	8
3.5.1.9	pwrUp	8
3.5.1.10	reserved	8
3.6	nRF24L01_DYNPD_t Union Reference	8
3.6.1	Field Documentation	9
3.6.1.1	b	9
3.6.1.2	B	9
3.6.1.3	dplP0	9
3.6.1.4	dplP1	9
3.6.1.5	dplP2	9
3.6.1.6	dplP3	9
3.6.1.7	dplP4	9
3.6.1.8	dplP5	9
3.6.1.9	reserved	9

3.7	nRF24L01_EN_RXADDR_t Union Reference	9
3.7.1	Field Documentation	10
3.7.1.1	b	10
3.7.1.2	B	10
3.7.1.3	erxP0	10
3.7.1.4	erxP1	10
3.7.1.5	erxP2	10
3.7.1.6	erxP3	10
3.7.1.7	erxP4	10
3.7.1.8	erxP5	10
3.7.1.9	reserved	10
3.8	nRF24L01_ENAA_t Union Reference	10
3.8.1	Field Documentation	11
3.8.1.1	b	11
3.8.1.2	B	11
3.8.1.3	enaaP0	11
3.8.1.4	enaaP1	11
3.8.1.5	enaaP2	11
3.8.1.6	enaaP3	11
3.8.1.7	enaaP4	11
3.8.1.8	enaaP5	11
3.8.1.9	reserved	11
3.9	nRF24L01_FEATURE_t Union Reference	11
3.9.1	Field Documentation	12
3.9.1.1	b	12
3.9.1.2	B	12
3.9.1.3	enAckPay	12
3.9.1.4	enDpl	12
3.9.1.5	enDynAck	12
3.9.1.6	reserved	12

3.10 nRF24L01_FIFO_STATUS_t Union Reference	12
3.10.1 Field Documentation	13
3.10.1.1 b	13
3.10.1.2 B	13
3.10.1.3 reserved0	13
3.10.1.4 reserved1	13
3.10.1.5 rxEmpty	13
3.10.1.6 rxFull	13
3.10.1.7 txEmpty	13
3.10.1.8 txFull	13
3.10.1.9 txReuse	13
3.11 nRF24L01_OBSERVE_TX_t Union Reference	13
3.11.1 Field Documentation	13
3.11.1.1 arcCNT	13
3.11.1.2 b	13
3.11.1.3 B	13
3.11.1.4 plosCNT	13
3.12 nRF24L01_RF_CH_t Union Reference	14
3.12.1 Field Documentation	14
3.12.1.1 b	14
3.12.1.2 B	14
3.12.1.3 reserved	14
3.12.1.4 rfch	14
3.13 nRF24L01_RF_SETUP_t Union Reference	14
3.13.1 Field Documentation	15
3.13.1.1 b	15
3.13.1.2 B	15
3.13.1.3 lnaHCURR	15
3.13.1.4 pllLock	15
3.13.1.5 reserved	15

3.13.1.6 rfDR	15
3.13.1.7 rfPWR	15
3.14 nRF24L01_RX_PW_P0_t Union Reference	15
3.14.1 Field Documentation	15
3.14.1.1 b	15
3.14.1.2 B	15
3.14.1.3 reserved	15
3.14.1.4 rxPwP0	15
3.15 nRF24L01_RX_PW_P1_t Union Reference	16
3.15.1 Field Documentation	16
3.15.1.1 b	16
3.15.1.2 B	16
3.15.1.3 reserved	16
3.15.1.4 rxPwP1	16
3.16 nRF24L01_RX_PW_P2_t Union Reference	16
3.16.1 Field Documentation	17
3.16.1.1 b	17
3.16.1.2 B	17
3.16.1.3 reserved	17
3.16.1.4 rxPwP2	17
3.17 nRF24L01_RX_PW_P3_t Union Reference	17
3.17.1 Field Documentation	17
3.17.1.1 b	17
3.17.1.2 B	17
3.17.1.3 reserved	17
3.17.1.4 rxPwP3	17
3.18 nRF24L01_RX_PW_P4_t Union Reference	17
3.18.1 Field Documentation	18
3.18.1.1 b	18
3.18.1.2 B	18

3.18.1.3	reserved	18
3.18.1.4	rxPwP4	18
3.19	nRF24L01_RX_PW_P5_t Union Reference	18
3.19.1	Field Documentation	18
3.19.1.1	b	18
3.19.1.2	B	18
3.19.1.3	reserved	18
3.19.1.4	rxPwP5	18
3.20	nRF24L01_SETUP_AW_t Union Reference	19
3.20.1	Field Documentation	19
3.20.1.1	aw	19
3.20.1.2	b	19
3.20.1.3	B	19
3.20.1.4	reserved	19
3.21	nRF24L01_SETUP_RETR_t Union Reference	19
3.21.1	Field Documentation	20
3.21.1.1	arc	20
3.21.1.2	ard	20
3.21.1.3	b	20
3.21.1.4	B	20
3.22	nRF24L01_SPIMessage_t Struct Reference	20
3.22.1	Field Documentation	20
3.22.1.1	bytesToSend	20
3.22.1.2	command	20
3.22.1.3	data	20
3.22.1.4	spiCh	20
3.23	nRF24L01_STATUS_t Union Reference	20
3.23.1	Field Documentation	21
3.23.1.1	b	21
3.23.1.2	B	21

3.23.1.3	maxRT	21
3.23.1.4	reserved	21
3.23.1.5	rxDR	21
3.23.1.6	rxPno	21
3.23.1.7	txDS	21
3.23.1.8	txFull	21
3.24	TemperatureData Union Reference	21
3.24.1	Field Documentation	22
3.24.1.1	data	22
3.24.1.2	msg	22
3.25	TemperatureMessage Struct Reference	22
3.25.1	Field Documentation	22
3.25.1.1	checksum	22
3.25.1.2	cr	22
3.25.1.3	currentDesired	22
3.25.1.4	currentRange	22
3.25.1.5	currentTemp	22
3.25.1.6	lf	22
3.25.1.7	powerOn	22
4	File Documentation	23
4.1	CircularBuffer/circularbuffer.c File Reference	23
4.2	CircularBuffer/circularbuffer.h File Reference	23
4.2.1	Typedef Documentation	24
4.2.1.1	BufferState_e	24
4.2.1.2	CircularBuffer_t	24
4.2.2	Enumeration Type Documentation	25
4.2.2.1	BufferState	25
4.2.3	Function Documentation	25
4.2.3.1	CBufferAdd(CircularBuffer_t *cb, void *data, uint8_t DMAch)	25

4.2.3.2	CBufferAddItems(CircularBuffer_t *cb, void *data, uint32_t numToAdd, uint8_t DMAch)	25
4.2.3.3	CBufferDestruct(CircularBuffer_t **cb)	25
4.2.3.4	CBufferInit(uint32_t itemSize, uint32_t maxItems)	26
4.2.3.5	CBufferRemove(CircularBuffer_t *cb, void *data, uint8_t DMAch)	26
4.2.3.6	IsBufferEmpty(CircularBuffer_t *cb)	26
4.2.3.7	IsBufferFull(CircularBuffer_t *cb)	26
4.3	Data/data.c File Reference	26
4.3.1	Function Documentation	27
4.3.1.1	BigToLittle(int32_t data)	27
4.3.1.2	DumpMemory(uint8_t *start, uint32_t length)	28
4.3.1.3	LittleToBig(int32_t data)	28
4.3.1.4	MyAtoi(uint8_t *str)	28
4.3.1.5	MyFtoa(uint8_t *str, double data, int32_t decimalPlaces)	28
4.3.1.6	Myltoa(uint8_t *str, int32_t data, int32_t base)	28
4.4	Data/data.h File Reference	29
4.4.1	Macro Definition Documentation	30
4.4.1.1	ASCIILETTERBASE	30
4.4.1.2	ASCIINUMBASE	30
4.4.1.3	BYTE0_MASK	30
4.4.1.4	BYTE0_SHIFT	30
4.4.1.5	BYTE1_MASK	30
4.4.1.6	BYTE1_SHIFT	30
4.4.1.7	BYTE2_MASK	30
4.4.1.8	BYTE2_SHIFT	30
4.4.1.9	BYTE3_MASK	30
4.4.1.10	BYTE3_SHIFT	30
4.4.2	Function Documentation	30
4.4.2.1	BigToLittle(int32_t data)	30
4.4.2.2	DumpMemory(uint8_t *start, uint32_t length)	30
4.4.2.3	LittleToBig(int32_t data)	31

4.4.2.4	MyAtoi(uint8_t *str)	31
4.4.2.5	MyFtoa(uint8_t *str, double data, int32_t decimalPlaces)	31
4.4.2.6	Myltoa(uint8_t *str, int32_t data, int32_t base)	31
4.5	Display/sevensegment.c File Reference	31
4.5.1	Function Documentation	32
4.5.1.1	Display_ClearAll(Display_place displayNum)	32
4.5.1.2	InitDisplay(Display_place place)	32
4.5.1.3	UpdateDisplay(Display_place displayNum, uint8_t value)	32
4.6	Display/sevensegment.h File Reference	33
4.6.1	Macro Definition Documentation	34
4.6.1.1	DISPLAY_A_PORTB_PIN	34
4.6.1.2	DISPLAY_A_PORTE_PIN	34
4.6.1.3	DISPLAY_B_PORTB_PIN	34
4.6.1.4	DISPLAY_B_PORTE_PIN	34
4.6.1.5	DISPLAY_C_PORTB_PIN	34
4.6.1.6	DISPLAY_C_PORTE_PIN	34
4.6.1.7	DISPLAY_CLEAR	34
4.6.1.8	DISPLAY_D_PORTB_PIN	34
4.6.1.9	DISPLAY_D_PORTE_PIN	34
4.6.1.10	DISPLAY_E_PORTB_PIN	34
4.6.1.11	DISPLAY_E_PORTE_PIN	35
4.6.1.12	DISPLAY_F_PORTB_PIN	35
4.6.1.13	DISPLAY_F_PORTE_PIN	35
4.6.1.14	DISPLAY_G_PORTB_PIN	35
4.6.1.15	DISPLAY_G_PORTE_PIN	35
4.6.1.16	DISPLAY_H_PORTB_PIN	35
4.6.1.17	DISPLAY_H_PORTE_PIN	35
4.6.1.18	DISPLAY_PORTB_A	35
4.6.1.19	DISPLAY_PORTB_B	35
4.6.1.20	DISPLAY_PORTB_C	35

4.6.1.21	DISPLAY_PORTB_D	35
4.6.1.22	DISPLAY_PORTB_E	35
4.6.1.23	DISPLAY_PORTB_F	35
4.6.1.24	DISPLAY_PORTB_G	35
4.6.1.25	DISPLAY_PORTB_H	35
4.6.1.26	DISPLAY_PORTE_A	35
4.6.1.27	DISPLAY_PORTE_B	35
4.6.1.28	DISPLAY_PORTE_C	35
4.6.1.29	DISPLAY_PORTE_D	35
4.6.1.30	DISPLAY_PORTE_E	35
4.6.1.31	DISPLAY_PORTE_F	35
4.6.1.32	DISPLAY_PORTE_G	35
4.6.1.33	DISPLAY_PORTE_H	35
4.6.1.34	DISPLAY_SET	36
4.6.1.35	NUM_DIGITS	36
4.6.1.36	NUM_DISPLAYS	36
4.6.1.37	NUM_SEGMENTS	36
4.6.2	Enumeration Type Documentation	36
4.6.2.1	Display_Errors	36
4.6.2.2	Display_place	36
4.6.3	Function Documentation	36
4.6.3.1	Display_ClearAll(Display_place place)	36
4.6.3.2	InitDisplay(Display_place place)	36
4.6.3.3	UpdateDisplay(Display_place place, uint8_t value)	37
4.6.4	Variable Documentation	37
4.6.4.1	Display_Port	37
4.6.4.2	Display_Value	37
4.7	Logging/uartlogging.c File Reference	37
4.8	Logging/uartlogging.h File Reference	37
4.8.1	Typedef Documentation	38

4.8.1.1	DataType_t	38
4.8.2	Enumeration Type Documentation	38
4.8.2.1	DataType_t	38
4.8.3	Function Documentation	38
4.8.3.1	LOG0(uint8_t *data)	38
4.8.3.2	LOG1(uint8_t *data, uint32_t length, DataType_t dataType, uint32_t numParms,...)	38
4.9	Main/includeall.h File Reference	38
4.9.1	Macro Definition Documentation	39
4.9.1.1	NO_DMA	39
4.9.1.2	UART_LOGGING	39
4.10	Main/main.c File Reference	39
4.10.1	Function Documentation	39
4.10.1.1	main()	39
4.10.2	Variable Documentation	39
4.10.2.1	SPI_RXBuffer	39
4.10.2.2	SPI_TXBuffer	39
4.10.2.3	UART0_RXBuffer	39
4.10.2.4	UART0_TXBuffer	39
4.10.2.5	UART1_RXBuffer	39
4.10.2.6	UART1_TXBuffer	39
4.11	Memory/memory.c File Reference	39
4.11.1	Function Documentation	40
4.11.1.1	MyMemMove(uint8_t *src, uint8_t *dst, uint32_t numBytes, uint8_t DMAch)	40
4.11.1.2	MyMemSet(uint8_t *dst, uint32_t value, size_t numBytes, uint8_t DMAch)	40
4.11.1.3	MyReverse(uint8_t *src, uint32_t length)	41
4.11.1.4	MyStrLen(uint8_t *str)	41
4.11.2	Variable Documentation	41
4.11.2.1	dmaComplete	41
4.12	Memory/memory.h File Reference	41
4.12.1	Function Documentation	42

4.12.1.1	MyMemMove(uint8_t *src, uint8_t *dst, uint32_t length, uint8_t DMACh)	42
4.12.1.2	MyMemSet(uint8_t *dst, uint32_t value, size_t numBytes, uint8_t DMACh)	42
4.12.1.3	MyReverse(uint8_t *src, uint32_t length)	42
4.12.1.4	MyStrLen(uint8_t *str)	42
4.13	Messaging/messaging.c File Reference	42
4.13.1	Function Documentation	43
4.13.1.1	BuildCommandMessage(Commands_e cmd, uint8_t data)	43
4.13.1.2	CalculateCommandChecksum(CommandMessage_t *msg)	44
4.13.1.3	CalculateTemperatureChecksum(TemperatureMessage_t *msg)	44
4.13.1.4	DecodeCommandMessage(CommandMessage_t *msg)	44
4.13.2	Variable Documentation	44
4.13.2.1	UART1_TXBuffer	44
4.14	Messaging/messaging.h File Reference	44
4.14.1	Macro Definition Documentation	46
4.14.1.1	COMMAND_MSG_BYTES	46
4.14.1.2	ENABLE_MESSAGING	46
4.14.1.3	MAX_LENGTH	46
4.14.1.4	NUM_COMMANDS	46
4.14.1.5	TEMP_MSG_BYTES	46
4.14.2	Typedef Documentation	46
4.14.2.1	CommandMessage_t	46
4.14.2.2	TemperatureMessage_t	46
4.14.3	Enumeration Type Documentation	46
4.14.3.1	Commands_e	46
4.14.3.2	MessagingErrors_e	47
4.14.4	Function Documentation	47
4.14.4.1	__attribute__((packed))	47
4.14.4.2	BuildCommandMessage(Commands_e cmd, uint8_t data)	47
4.14.4.3	CalculateCommandChecksum(CommandMessage_t *msg)	47
4.14.4.4	CalculateTemperatureChecksum(TemperatureMessage_t *msg)	47

4.14.4.5	DecodeCommandMessage(CommandMessage_t *msg)	48
4.14.5	Variable Documentation	48
4.14.5.1	__attribute__	48
4.14.5.2	checksum	48
4.14.5.3	cmd	48
4.14.5.4	cr	48
4.14.5.5	currentDesired	48
4.14.5.6	currentRange	48
4.14.5.7	currentTemp	48
4.14.5.8	data	48
4.14.5.9	If	48
4.14.5.10	powerOn	48
4.15	Modules/adc.c File Reference	48
4.15.1	Function Documentation	49
4.15.1.1	ADC0_IRQHandler()	49
4.15.1.2	ADC_GetCurrentValue()	49
4.15.1.3	ADC_Init(ADC_InputChannel ADC_ch)	49
4.15.1.4	ADC_StartConversion(ADC_InputChannel ADC_ch)	50
4.15.2	Variable Documentation	50
4.15.2.1	ADC_value	50
4.16	Modules/adc.h File Reference	50
4.16.1	Macro Definition Documentation	51
4.16.1.1	ADC_CH_PIN	51
4.16.1.2	ADC_CHANNEL	51
4.16.1.3	ADC_PORT	51
4.16.1.4	MAX_VALUE	51
4.16.1.5	MUX_PIN_ANALOG	51
4.16.2	Enumeration Type Documentation	51
4.16.2.1	ADC_AvgSamples	51
4.16.2.2	ADC_InputChannel	52

4.16.3	Function Documentation	52
4.16.3.1	ADC_GetCurrentValue()	52
4.16.3.2	ADC_Init(ADC_InputChannel ADC_ch)	53
4.16.3.3	ADC_StartConversion(ADC_InputChannel ADC_ch)	53
4.17	Modules/diags.c File Reference	53
4.17.1	Function Documentation	53
4.17.1.1	ParseDiag(uint8_t *buffer)	53
4.18	Modules/diags.h File Reference	54
4.18.1	Function Documentation	54
4.18.1.1	ParseDiag(uint8_t *buffer)	54
4.19	Modules/dma.c File Reference	54
4.20	Modules/dma.h File Reference	54
4.20.1	Macro Definition Documentation	55
4.20.1.1	DMA_BCR_SIZE_MASK	55
4.20.1.2	MASK_32BIT	55
4.20.1.3	NO_DMA	55
4.20.2	Typedef Documentation	55
4.20.2.1	BufferSize_e	55
4.20.2.2	DMAErrors_e	55
4.20.2.3	TransferSize_e	55
4.20.3	Enumeration Type Documentation	55
4.20.3.1	BufferSize	55
4.20.3.2	DMAErrors	56
4.20.3.3	TransferSize	56
4.20.4	Function Documentation	56
4.20.4.1	InitDMA(uint8_t ch)	56
4.20.4.2	MemSet32bit(uint8_t ch, uint32_t data, uint8_t *dst, uint32_t numBytes)	56
4.20.4.3	MemSet8bit(uint8_t ch, uint8_t data, uint8_t *dst, uint32_t numBytes)	56
4.20.4.4	StartTransfer16bitMoves(uint8_t ch, uint8_t *src, uint8_t *dst, uint32_t numBytes)	56
4.20.4.5	StartTransfer32bitMoves(uint8_t ch, uint8_t *src, uint8_t *dst, uint32_t numBytes)	56

4.20.4.6	StartTransfer8bitMoves(uint8_t ch, uint8_t *src, uint8_t *dst, uint32_t numBytes)	56
4.21	Modules/ds18b20.c File Reference	56
4.21.1	Function Documentation	57
4.21.1.1	ConvertRawTemperatureData(uint16_t rawTemperatureData)	57
4.21.1.2	ReadTemp()	58
4.21.1.3	TransactionStepOne()	58
4.21.1.4	TransactionStepTwo()	58
4.22	Modules/ds18b20.h File Reference	58
4.22.1	Macro Definition Documentation	60
4.22.1.1	ALARM_SEARCH	60
4.22.1.2	CONVERT_T	60
4.22.1.3	COPY_SCRATCHPAD	60
4.22.1.4	MATCH_ROM	60
4.22.1.5	READ_POWER_SUPPLY	60
4.22.1.6	READ_ROM	60
4.22.1.7	READ_SCRATCHPAD	60
4.22.1.8	RECALL_E2	60
4.22.1.9	ROM_BYTES	60
4.22.1.10	SCRATCHPAD_BYTES	60
4.22.1.11	SEARCH_ROM	60
4.22.1.12	SERIAL_NUM_BYTES	60
4.22.1.13	SKIP_ROM	60
4.22.1.14	WAIT_TIME_LONG	60
4.22.1.15	WAIT_TIME_SHORT	60
4.22.1.16	WRITE_SCRATCHPAD	60
4.22.2	Function Documentation	60
4.22.2.1	ConvertRawTemperatureData(uint16_t rawTemperatureData)	60
4.22.2.2	ReadTemp()	61
4.22.2.3	TransactionStepOne()	61
4.22.2.4	TransactionStepTwo()	61

4.23 Modules/io.h File Reference	61
4.24 Modules/led.c File Reference	61
4.24.1 Function Documentation	62
4.24.1.1 CycleLEDs()	62
4.25 Modules/led.h File Reference	62
4.25.1 Macro Definition Documentation	63
4.25.1.1 BLUE_CHANNEL	63
4.25.1.2 BLUE_PIN	63
4.25.1.3 BLUE_TPM	63
4.25.1.4 GREEN_CHANNEL	63
4.25.1.5 GREEN_PIN	63
4.25.1.6 GREEN_TPM	63
4.25.1.7 RED_CHANNEL	63
4.25.1.8 RED_PIN	63
4.25.1.9 RED_TPM	63
4.25.2 Typedef Documentation	63
4.25.2.1 Color_t	63
4.25.3 Enumeration Type Documentation	63
4.25.3.1 Color_t	63
4.25.4 Function Documentation	64
4.25.4.1 CycleLEDs()	64
4.25.4.2 LEDSetup(void)	64
4.25.4.3 SwitchLEDs(uint8_t color)	64
4.26 Modules/nRF24L01.c File Reference	64
4.26.1 Function Documentation	65
4.26.1.1 nRF24L01_Activate(uint8_t SPI_ch)	65
4.26.1.2 nRF24L01_PowerDown(uint8_t SPI_ch)	65
4.26.1.3 nRF24L01_ReadReg(uint8_t SPI_ch, nRF24L01_Registers_e reg)	66
4.26.1.4 nRF24L01_SendData(nRF24L01_SPIMessage_t *msg)	66
4.26.1.5 nRF24L01_SendNOP(uint8_t SPI_ch)	66

4.26.1.6	nRF24L01_SetRXMode(uint8_t SPI_ch)	66
4.26.1.7	nRF24L01_SetTXMode(uint8_t SPI_ch)	66
4.26.1.8	nRF24L01_SetupChannel(uint8_t SPI_ch)	67
4.26.1.9	nRF24L01_StandbyMode(uint8_t SPI_ch)	67
4.26.1.10	nRF24L01_WriteReg(uint8_t SPI_ch, nRF24L01_Registers_e reg, uint8_t data↔ ToWrite)	67
4.26.2	Variable Documentation	67
4.26.2.1	readRegComplete	67
4.26.2.2	SPI_RXBuffer	67
4.26.2.3	SPI_TXBuffer	67
4.27	Modules/nRF24L01.h File Reference	67
4.27.1	Macro Definition Documentation	70
4.27.1.1	ACTIVATE	70
4.27.1.2	ACTIVATE_KEY	70
4.27.1.3	CRCO_MASK	70
4.27.1.4	EN_CRC_MASK	70
4.27.1.5	FLUSH_RX	70
4.27.1.6	FLUSH_TX	70
4.27.1.7	MASK_MAX_RT_MASK	70
4.27.1.8	MASK_RX_DR_MASK	70
4.27.1.9	MASK_TX_DS_MASK	70
4.27.1.10	NOP	70
4.27.1.11	nRF24L01_0_CE	70
4.27.1.12	nRF24L01_0_CE_PIN	70
4.27.1.13	nRF24L01_0_IRQ	71
4.27.1.14	nRF24L01_0_IRQ_PIN	71
4.27.1.15	nRF24L01_1_CE	71
4.27.1.16	nRF24L01_1_CE_PIN	71
4.27.1.17	nRF24L01_1_IRQ	71
4.27.1.18	nRF24L01_1_IRQ_PIN	71
4.27.1.19	nRF24L01_CHANNEL_FREQ	71

4.27.1.20	nRF24L01_DATA_RATE	71
4.27.1.21	nRF24L01_LNA_GAIN	71
4.27.1.22	nRF24L01_PA_CONTROL	71
4.27.1.23	PRIM_RX_MASK	71
4.27.1.24	PWR_UP_MASK	71
4.27.1.25	R_RX_PAYLOAD	71
4.27.1.26	R_RX_PL_WID	71
4.27.1.27	READ_REG	71
4.27.1.28	REUSE_TX_PL	71
4.27.1.29	W_ACK_PAYLOAD	71
4.27.1.30	W_TX_PAYLOAD	71
4.27.1.31	W_TX_PAYLOAD_NO_ACK	71
4.27.1.32	WRITE_REG	71
4.27.2	Enumeration Type Documentation	71
4.27.2.1	ADR_e	71
4.27.2.2	CHANNEL_e	72
4.27.2.3	nRF24L01_Registers_e	72
4.27.2.4	PWR_e	73
4.27.3	Function Documentation	73
4.27.3.1	nRF24L01_Activate(uint8_t SPI_ch)	73
4.27.3.2	nRF24L01_PowerDown(uint8_t SPI_ch)	73
4.27.3.3	nRF24L01_ReadReg(uint8_t SPI_ch, nRF24L01_Registers_e reg)	73
4.27.3.4	nRF24L01_SendData(nRF24L01_SPIMessage_t *msg)	74
4.27.3.5	nRF24L01_SendNOP(uint8_t SPI_ch)	74
4.27.3.6	nRF24L01_SetRXMode(uint8_t SPI_ch)	74
4.27.3.7	nRF24L01_SetTXMode(uint8_t SPI_ch)	74
4.27.3.8	nRF24L01_SetupChannel(uint8_t SPI_ch)	74
4.27.3.9	nRF24L01_StandbyMode(uint8_t SPI_ch)	75
4.27.3.10	nRF24L01_WriteReg(uint8_t SPI_ch, nRF24L01_Registers_e reg, uint8_t data↔ ToWrite)	75
4.28	Modules/pushbutton.c File Reference	75

4.28.1	Function Documentation	75
4.28.1.1	Button_Init(uint8_t buttonNum)	75
4.28.1.2	PORTA_IRQHandler()	76
4.29	Modules/pushbutton.h File Reference	76
4.29.1	Macro Definition Documentation	76
4.29.1.1	BUTTON0	76
4.29.1.2	BUTTON0_PIN	76
4.29.1.3	NUM_BUTTONS	76
4.29.2	Function Documentation	76
4.29.2.1	Button_Init(uint8_t buttonNum)	76
4.30	Modules/singlewirecomms.c File Reference	77
4.30.1	Function Documentation	77
4.30.1.1	SWC_Init()	77
4.30.1.2	SWC_ReadData(uint8_t bytesToRead, uint8_t *data)	78
4.30.1.3	SWC_ReadStatusAndWait()	78
4.30.1.4	SWC_ResetAndPresencePulses()	78
4.30.1.5	SWC_SendByte(uint8_t data)	78
4.31	Modules/singlewirecomms.h File Reference	79
4.31.1	Macro Definition Documentation	80
4.31.1.1	DATA_LINE	80
4.31.1.2	DATA_LINE_PIN	80
4.31.1.3	MAX_BYTES	80
4.31.1.4	PULL_LOW	80
4.31.1.5	READ_LINE	80
4.31.1.6	RELEASE_LINE	80
4.31.1.7	SWITCH_TO_RX	80
4.31.1.8	SWITCH_TO_TX	80
4.31.1.9	WRITE_0	80
4.31.1.10	WRITE_1	80
4.31.2	Function Documentation	80

4.31.2.1	SWC_Init()	80
4.31.2.2	SWC_ReadData(uint8_t bytesToRead, uint8_t *data)	80
4.31.2.3	SWC_ReadStatusAndWait()	81
4.31.2.4	SWC_ResetAndPresencePulses()	81
4.31.2.5	SWC_SendByte(uint8_t data)	81
4.32	Modules/spi.c File Reference	81
4.32.1	Function Documentation	82
4.32.1.1	SPI_Init(uint8_t SPI_ch, uint8_t master)	82
4.32.1.2	SPI_TransmitData(uint8_t SPI_ch, size_t numBytes)	82
4.32.2	Variable Documentation	82
4.32.2.1	SPI_RXBuffer	82
4.32.2.2	SPI_TXBuffer	82
4.32.2.3	spiDeviceLocation	82
4.33	Modules/spi.h File Reference	82
4.33.1	Macro Definition Documentation	84
4.33.1.1	ARRAY_SIZE	84
4.33.1.2	BBB_SPI_SPEED	84
4.33.1.3	BPW	84
4.33.1.4	DEVICE_LOC	84
4.33.1.5	MODE	84
4.33.1.6	SPI0_CS	84
4.33.1.7	SPI0_CS_PIN	84
4.33.1.8	SPI0_MISO	84
4.33.1.9	SPI0_MOSI	84
4.33.1.10	SPI0_SCK	84
4.33.1.11	SPI1_CS	84
4.33.1.12	SPI1_CS_PIN	84
4.33.1.13	SPI1_MISO	84
4.33.1.14	SPI1_MOSI	84
4.33.1.15	SPI1_SCK	84

4.33.1.16	SPI_0_5Mbps_BRD	84
4.33.1.17	SPI_0_5Mbps_PRESCALER	84
4.33.1.18	SPI_1Mbps_BRD	84
4.33.1.19	SPI_1Mbps_PRESCALER	84
4.33.1.20	SPI_2Mbps_BRD	84
4.33.1.21	SPI_2Mbps_PRESCALER	84
4.33.1.22	SPI_CHANNELS	84
4.33.1.23	SPI_RXBUFFER_SIZE	85
4.33.1.24	SPI_TXBUFFER_SIZE	85
4.33.2	Enumeration Type Documentation	85
4.33.2.1	SPI_BRDivisor	85
4.33.2.2	SPI_BRPrescaler	85
4.33.3	Function Documentation	85
4.33.3.1	SPI_Init(uint8_t SPI_ch, uint8_t master)	85
4.33.3.2	SPI_TransmitData(uint8_t SPI_ch, size_t numBytes)	86
4.34	Modules/timers.c File Reference	86
4.34.1	Function Documentation	86
4.34.1.1	GetElapsedTime(uint32_t start, uint32_t end)	86
4.34.1.2	GetTime()	86
4.35	Modules/timers.h File Reference	86
4.35.1	Macro Definition Documentation	87
4.35.1.1	_10US_PER_SEC	87
4.35.1.2	COUNTS_PER_US	87
4.35.1.3	MAX_MODULUS	87
4.35.1.4	MAX_PRESCALER	87
4.35.1.5	NS_PER_SEC	87
4.35.1.6	NS_PER_US	87
4.35.1.7	PROFILER_CH	87
4.35.1.8	PROFILER_PERIOD_IN_NS	87
4.35.1.9	PROFILER_TPM	87

4.35.1.10 UNITS_US	87
4.35.1.11 WAIT_CH	87
4.35.1.12 WAIT_TPM	87
4.35.2 Enumeration Type Documentation	87
4.35.2.1 TPM_TriggerOptions	87
4.35.3 Function Documentation	88
4.35.3.1 GetElapsedTime(uint32_t start, uint32_t end)	88
4.35.3.2 GetTime()	88
4.36 Modules/uart.c File Reference	88
4.36.1 Function Documentation	89
4.36.1.1 PutChar(uint8_t data)	89
4.36.1.2 UartRX(uint8_t *data)	89
4.36.1.3 UartSetup(uint8_t channel, uint32_t requestedBuadRate, uint8_t parity)	89
4.36.1.4 UartTX(uint8_t *buffer, uint32_t length)	89
4.36.2 Variable Documentation	90
4.36.2.1 parseDiag	90
4.36.2.2 UART0_RXBuffer	90
4.36.2.3 UART0_TXBuffer	90
4.36.2.4 UART1_RXBuffer	90
4.36.2.5 UART1_TXBuffer	90
4.37 Modules/uart.h File Reference	90
4.37.1 Macro Definition Documentation	91
4.37.1.1 BAUDRATE	91
4.37.1.2 BONEPATH	91
4.37.1.3 CR	91
4.37.1.4 DMA_RXBUFFER_SIZE	91
4.37.1.5 DMA_TXBUFFER_SIZE	91
4.37.1.6 DMACH_UART0RX	91
4.37.1.7 DMACH_UART0TX	91
4.37.1.8 LF	91

4.37.1.9	MODEMDEVICE	91
4.37.1.10	OSR	91
4.37.1.11	POSIX_SOURCE	91
4.37.1.12	RXBUFFER_SIZE	91
4.37.1.13	TXBUFFER_SIZE	91
4.37.2	Function Documentation	91
4.37.2.1	PutChar(uint8_t data)	91
4.37.2.2	UartRX(uint8_t *data)	92
4.37.2.3	UartSetup(uint8_t channel, uint32_t buadRate, uint8_t parity)	92
4.37.2.4	UartTX(uint8_t *buffer, uint32_t length)	92
4.37.3	Variable Documentation	92
4.37.3.1	parseDiag	92
4.38	README.txt File Reference	92
4.38.1	Variable Documentation	93
4.38.1.1	assemble	93
4.38.1.2	blue	93
4.38.1.3	commands	93
4.38.1.4	cyan	93
4.38.1.5	d	93
4.38.1.6	follows	93
4.38.1.7	green	93
4.38.1.8	i	94
4.38.1.9	lib	94
4.38.1.10	link	94
4.38.1.11	purple	94
4.38.1.12	S	94
4.38.1.13	upload	94
4.38.1.14	white	94
4.38.1.15	x	94
4.38.1.16	yellow	94

4.39 TemperatureController/controller.c File Reference	94
4.39.1 Function Documentation	95
4.39.1.1 Controller_ChangeDisplay(uint8_t value)	95
4.39.1.2 Controller_ChangeState()	96
4.39.1.3 Controller_Init()	96
4.39.1.4 Controller_SendTempData(uint8_t dontcare)	96
4.39.1.5 Controller_SetCurrentTemp(uint8_t newTemp)	96
4.39.1.6 Controller_SetDesiredTemp(uint8_t newTemp)	96
4.39.1.7 Controller_SetTempRange(uint8_t newRange)	97
4.39.1.8 Controller_StateMachine()	97
4.39.2 Variable Documentation	97
4.39.2.1 currentTemp	97
4.39.2.2 desiredTemp	97
4.39.2.3 power	97
4.39.2.4 state	97
4.39.2.5 tempRange	97
4.40 TemperatureController/controller.h File Reference	97
4.40.1 Macro Definition Documentation	99
4.40.1.1 CONVERT_C_TO_F	99
4.40.1.2 CONVERT_F_TO_C	99
4.40.1.3 MAX_DISPLAY_VAL	99
4.40.1.4 RELAY	99
4.40.1.5 RELAY_OFF	99
4.40.1.6 RELAY_ON	99
4.40.1.7 RELAY_PIN	99
4.40.2 Enumeration Type Documentation	99
4.40.2.1 ControllerState_e	99
4.40.3 Function Documentation	99
4.40.3.1 Controller_ChangeDisplay(uint8_t value)	99
4.40.3.2 Controller_ChangeState()	99

4.40.3.3	Controller_Init()	100
4.40.3.4	Controller_SendTempData(uint8_t dontcare)	100
4.40.3.5	Controller_SetCurrentTemp(uint8_t newTemp)	100
4.40.3.6	Controller_SetDesiredTemp(uint8_t newTemp)	100
4.40.3.7	Controller_SetTempRange(uint8_t newRange)	100
4.40.3.8	Controller_StateMachine()	101
4.41	Testing/circularbuffertesting.c File Reference	101
4.42	Testing/circularbuffertesting.h File Reference	101
4.43	Testing/datatesting.c File Reference	101
4.44	Testing/datatesting.h File Reference	101
4.45	Testing/memorytesting.c File Reference	101
4.46	Testing/memorytesting.h File Reference	101
4.46.1	Macro Definition Documentation	102
4.46.1.1	MAIN_HEADER	102
4.46.1.2	MEM_MOVE_SIZE	102
4.46.1.3	MY_MEM_ZERO_SIZE	102
4.46.1.4	MY_REVERSE_INPUTS	102
4.46.1.5	MY_STR_LEN_INPUTS	102
4.46.1.6	TESTING_DMA_CH	102
4.46.1.7	TESTSTRING	102
4.46.2	Function Documentation	102
4.46.2.1	MemoryTesting(void)	102
4.46.2.2	MyMemMoveUnitTest(void)	102
4.46.2.3	MyMemSetUnitTest(void)	102
4.46.2.4	MyReverseUnitTest(void)	102
4.46.2.5	MyStrLenUnitTest(void)	102
4.47	Testing/performancetesting.c File Reference	102
4.48	Testing/performancetesting.h File Reference	102
4.48.1	Macro Definition Documentation	103
4.48.1.1	MEMORY_TEST_CASES	103
4.48.1.2	PRINTING_BUFFER_SIZE	103
4.48.1.3	PRINTING_TEST_CASES	103
4.48.2	Function Documentation	103
4.48.2.1	DataPerformanceTesting(void)	103
4.48.2.2	MemoryPerformanceTesting(void)	103
4.48.2.3	PerformanceTesting(void)	103
4.48.2.4	PrintingPerformanceTesting(void)	103
4.49	Testing/testing.c File Reference	103
4.50	Testing/testing.h File Reference	103

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

CircularBuffer_t	5
CommandMessage	6
DS8B20_ROMCode	6
DS8B20_Scratchpad	7
nRF24L01_CONFIG_t	7
nRF24L01_DYNPD_t	8
nRF24L01_EN_RXADDR_t	9
nRF24L01_ENAA_t	10
nRF24L01_FEATURE_t	11
nRF24L01_FIFO_STATUS_t	12
nRF24L01_OBSERVE_TX_t	13
nRF24L01_RF_CH_t	14
nRF24L01_RF_SETUP_t	14
nRF24L01_RX_PW_P0_t	15
nRF24L01_RX_PW_P1_t	16
nRF24L01_RX_PW_P2_t	16
nRF24L01_RX_PW_P3_t	17
nRF24L01_RX_PW_P4_t	17
nRF24L01_RX_PW_P5_t	18
nRF24L01_SETUP_AW_t	19
nRF24L01_SETUP_RETR_t	19
nRF24L01_SPIMessage_t	20
nRF24L01_STATUS_t	20
TemperatureData	21
TemperatureMessage	22

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

CircularBuffer/circularbuffer.c	23
CircularBuffer/circularbuffer.h	23
Data/data.c	26
Data/data.h	29
Display/sevenssegment.c	31
Display/sevenssegment.h	33
Logging/uartlogging.c	37
Logging/uartlogging.h	37
Main/includeall.h	38
Main/main.c	39
Memory/memory.c	39
Memory/memory.h	41
Messaging/messaging.c	42
Messaging/messaging.h	44
Modules/adc.c	48
Modules/adc.h	50
Modules/diags.c	53
Modules/diags.h	54
Modules/dma.c	54
Modules/dma.h	54
Modules/ds18b20.c	56
Modules/ds18b20.h	58
Modules/io.h	61
Modules/led.c	61
Modules/led.h	62
Modules/nRF24L01.c	64
Modules/nRF24L01.h	67
Modules/pushbutton.c	75
Modules/pushbutton.h	76
Modules/singlewirecomms.c	77
Modules/singlewirecomms.h	79
Modules/spi.c	81
Modules/spi.h	82
Modules/timers.c	86
Modules/timers.h	86

Modules/ uart.c	88
Modules/ uart.h	90
TemperatureController/ controller.c	94
TemperatureController/ controller.h	97
Testing/ circularbuffertesting.c	101
Testing/ circularbuffertesting.h	101
Testing/ datatesting.c	101
Testing/ datatesting.h	101
Testing/ memorytesting.c	101
Testing/ memorytesting.h	101
Testing/ performancetesting.c	102
Testing/ performancetesting.h	102
Testing/ testing.c	103
Testing/ testing.h	103

Chapter 3

Data Structure Documentation

3.1 CircularBuffer_t Struct Reference

```
#include <circularbuffer.h>
```

Data Fields

- void * [bufferStart](#)
- void * [head](#)
- void * [tail](#)
- void * [bufferEnd](#)
- uint32_t [size](#)
- uint32_t [numItems](#)
- uint32_t [itemSize](#)

3.1.1 Field Documentation

3.1.1.1 void* CircularBuffer_t::bufferEnd

3.1.1.2 void* CircularBuffer_t::bufferStart

3.1.1.3 void* CircularBuffer_t::head

3.1.1.4 uint32_t CircularBuffer_t::itemSize

3.1.1.5 uint32_t CircularBuffer_t::numItems

3.1.1.6 uint32_t CircularBuffer_t::size

3.1.1.7 void* CircularBuffer_t::tail

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

- CircularBuffer/[circularbuffer.h](#)

3.2 CommandMessage Struct Reference

```
#include <messaging.h>
```

Data Fields

- uint8_t [cmd](#)
- uint8_t [data](#)
- uint8_t [checksum](#)

3.2.1 Field Documentation

3.2.1.1 uint8_t CommandMessage::checksum

3.2.1.2 uint8_t CommandMessage::cmd

3.2.1.3 uint8_t CommandMessage::data

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

- Messaging/[messaging.h](#)

3.3 DS8B20_ROMCode Union Reference

```
#include <ds18b20.h>
```

Public Member Functions

- struct [__attribute__](#) ((__packed__))

Data Fields

- uint8_t [romBytes](#) [[ROM_BYTES](#)]

3.3.1 Member Function Documentation

3.3.1.1 struct DS8B20_ROMCode::__attribute__ ((__packed__)) [\[inline\]](#)

3.3.2 Field Documentation

3.3.2.1 uint8_t DS8B20_ROMCode::romBytes[[ROM_BYTES](#)]

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

- Modules/[ds18b20.h](#)

3.4 DS8B20_Scratchpad Union Reference

```
#include <ds18b20.h>
```

Public Member Functions

- struct [__attribute__](#) ((__packed__))

Data Fields

- uint8_t [scratchpadBytes](#) [[SCRATCPAD_BYTES](#)]

3.4.1 Member Function Documentation

3.4.1.1 struct DS8B20_Scratchpad::__attribute__ ((__packed__)) [\[inline\]](#)

3.4.2 Field Documentation

3.4.2.1 uint8_t DS8B20_Scratchpad::scratchpadBytes[[SCRATCPAD_BYTES](#)]

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

- Modules/[ds18b20.h](#)

3.5 nRF24L01_CONFIG_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- struct {
 - uint8_t [primRx](#):1
 - uint8_t [pwrUp](#):1
 - uint8_t [crco](#):1
 - uint8_t [enCRC](#):1
 - uint8_t [maskMaxRt](#):1
 - uint8_t [maskTxDs](#):1
 - uint8_t [maskRxDr](#):1
 - uint8_t [reserved](#):1
 } [b](#)
- uint8_t [B](#)

3.5.1 Field Documentation

3.5.1.1 `struct { ... } nRF24L01_CONFIG_t::b`

3.5.1.2 `uint8_t nRF24L01_CONFIG_t::B`

3.5.1.3 `uint8_t nRF24L01_CONFIG_t::crco`

3.5.1.4 `uint8_t nRF24L01_CONFIG_t::enCRC`

3.5.1.5 `uint8_t nRF24L01_CONFIG_t::maskMaxRt`

3.5.1.6 `uint8_t nRF24L01_CONFIG_t::maskRxDr`

3.5.1.7 `uint8_t nRF24L01_CONFIG_t::maskTxDs`

3.5.1.8 `uint8_t nRF24L01_CONFIG_t::primRx`

3.5.1.9 `uint8_t nRF24L01_CONFIG_t::pwrUp`

3.5.1.10 `uint8_t nRF24L01_CONFIG_t::reserved`

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

- Modules/[nRF24L01.h](#)

3.6 nRF24L01_DYNPD_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- `struct {`
 - `uint8_t dplP0:1`
 - `uint8_t dplP1:1`
 - `uint8_t dplP2:1`
 - `uint8_t dplP3:1`
 - `uint8_t dplP4:1`
 - `uint8_t dplP5:1`
 - `uint8_t reserved:1``} b`
- `uint8_t B`

3.6.1 Field Documentation

3.6.1.1 struct { ... } nRF24L01_DYNPD_t::b

3.6.1.2 uint8_t nRF24L01_DYNPD_t::B

3.6.1.3 uint8_t nRF24L01_DYNPD_t::dplP0

3.6.1.4 uint8_t nRF24L01_DYNPD_t::dplP1

3.6.1.5 uint8_t nRF24L01_DYNPD_t::dplP2

3.6.1.6 uint8_t nRF24L01_DYNPD_t::dplP3

3.6.1.7 uint8_t nRF24L01_DYNPD_t::dplP4

3.6.1.8 uint8_t nRF24L01_DYNPD_t::dplP5

3.6.1.9 uint8_t nRF24L01_DYNPD_t::reserved

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

- Modules/[nRF24L01.h](#)

3.7 nRF24L01_EN_RXADDR_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- struct {
 - uint8_t [erxP0](#):1
 - uint8_t [erxP1](#):1
 - uint8_t [erxP2](#):1
 - uint8_t [erxP3](#):1
 - uint8_t [erxP4](#):1
 - uint8_t [erxP5](#):1
 - uint8_t [reserved](#):2
 } [b](#)
- uint8_t [B](#)

3.7.1 Field Documentation

3.7.1.1 `struct { ... } nRF24L01_EN_RXADDR_t::b`

3.7.1.2 `uint8_t nRF24L01_EN_RXADDR_t::B`

3.7.1.3 `uint8_t nRF24L01_EN_RXADDR_t::erxP0`

3.7.1.4 `uint8_t nRF24L01_EN_RXADDR_t::erxP1`

3.7.1.5 `uint8_t nRF24L01_EN_RXADDR_t::erxP2`

3.7.1.6 `uint8_t nRF24L01_EN_RXADDR_t::erxP3`

3.7.1.7 `uint8_t nRF24L01_EN_RXADDR_t::erxP4`

3.7.1.8 `uint8_t nRF24L01_EN_RXADDR_t::erxP5`

3.7.1.9 `uint8_t nRF24L01_EN_RXADDR_t::reserved`

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

- Modules/[nRF24L01.h](#)

3.8 nRF24L01_ENAA_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- `struct {`
 - `uint8_t enaaP0:1`
 - `uint8_t enaaP1:1`
 - `uint8_t enaaP2:1`
 - `uint8_t enaaP3:1`
 - `uint8_t enaaP4:1`
 - `uint8_t enaaP5:1`
 - `uint8_t reserved:2``} b`
- `uint8_t B`

3.8.1 Field Documentation

3.8.1.1 `struct { ... } nRF24L01_ENAA_t::b`

3.8.1.2 `uint8_t nRF24L01_ENAA_t::B`

3.8.1.3 `uint8_t nRF24L01_ENAA_t::enaaP0`

3.8.1.4 `uint8_t nRF24L01_ENAA_t::enaaP1`

3.8.1.5 `uint8_t nRF24L01_ENAA_t::enaaP2`

3.8.1.6 `uint8_t nRF24L01_ENAA_t::enaaP3`

3.8.1.7 `uint8_t nRF24L01_ENAA_t::enaaP4`

3.8.1.8 `uint8_t nRF24L01_ENAA_t::enaaP5`

3.8.1.9 `uint8_t nRF24L01_ENAA_t::reserved`

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

- Modules/[nRF24L01.h](#)

3.9 nRF24L01_FEATURE_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- `struct {
 uint8_t enDynAck:1
 uint8_t enAckPay:1
 uint8_t enDpl:1
 uint8_t reserved:5
} b`
- `uint8_t B`

3.9.1 Field Documentation

3.9.1.1 struct { ... } nRF24L01_FEATURE_t::b

3.9.1.2 uint8_t nRF24L01_FEATURE_t::B

3.9.1.3 uint8_t nRF24L01_FEATURE_t::enAckPay

3.9.1.4 uint8_t nRF24L01_FEATURE_t::enDpl

3.9.1.5 uint8_t nRF24L01_FEATURE_t::enDynAck

3.9.1.6 uint8_t nRF24L01_FEATURE_t::reserved

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

- Modules/[nRF24L01.h](#)

3.10 nRF24L01_FIFO_STATUS_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- struct {
 - uint8_t [rxEmpty](#):1
 - uint8_t [rxFull](#):1
 - uint8_t [reserved0](#):2
 - uint8_t [txEmpty](#):1
 - uint8_t [txFull](#):1
 - uint8_t [txReuse](#):1
 - uint8_t [reserved1](#):1
 } [b](#)
- uint8_t [B](#)

3.10.1 Field Documentation

- 3.10.1.1 `struct { ... } nRF24L01_FIFO_STATUS_t::b`
- 3.10.1.2 `uint8_t nRF24L01_FIFO_STATUS_t::B`
- 3.10.1.3 `uint8_t nRF24L01_FIFO_STATUS_t::reserved0`
- 3.10.1.4 `uint8_t nRF24L01_FIFO_STATUS_t::reserved1`
- 3.10.1.5 `uint8_t nRF24L01_FIFO_STATUS_t::rxEmpty`
- 3.10.1.6 `uint8_t nRF24L01_FIFO_STATUS_t::rxFull`
- 3.10.1.7 `uint8_t nRF24L01_FIFO_STATUS_t::txEmpty`
- 3.10.1.8 `uint8_t nRF24L01_FIFO_STATUS_t::txFull`
- 3.10.1.9 `uint8_t nRF24L01_FIFO_STATUS_t::txReuse`

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

- Modules/[nRF24L01.h](#)

3.11 nRF24L01_OBSERVE_TX_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- `struct {`
 - `uint8_t arcCNT:4`
 - `uint8_t plosCNT:4`- `} b`
- `uint8_t B`

3.11.1 Field Documentation

- 3.11.1.1 `uint8_t nRF24L01_OBSERVE_TX_t::arcCNT`
- 3.11.1.2 `struct { ... } nRF24L01_OBSERVE_TX_t::b`
- 3.11.1.3 `uint8_t nRF24L01_OBSERVE_TX_t::B`
- 3.11.1.4 `uint8_t nRF24L01_OBSERVE_TX_t::plosCNT`

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

- Modules/[nRF24L01.h](#)

3.12 nRF24L01_RF_CH_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- struct {
 uint8_t rfch:7
 uint8_t reserved:1
} b
- uint8_t B

3.12.1 Field Documentation

3.12.1.1 struct { ... } nRF24L01_RF_CH_t::b

3.12.1.2 uint8_t nRF24L01_RF_CH_t::B

3.12.1.3 uint8_t nRF24L01_RF_CH_t::reserved

3.12.1.4 uint8_t nRF24L01_RF_CH_t::rfch

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

- Modules/[nRF24L01.h](#)

3.13 nRF24L01_RF_SETUP_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- struct {
 uint8_t lnaHCURR:1
 uint8_t rfPWR:2
 uint8_t rfDR:1
 uint8_t pllLock:1
 uint8_t reserved:3
} b
- uint8_t B

3.13.1 Field Documentation

3.13.1.1 `struct { ... } nRF24L01_RF_SETUP_t::b`

3.13.1.2 `uint8_t nRF24L01_RF_SETUP_t::B`

3.13.1.3 `uint8_t nRF24L01_RF_SETUP_t::lnaHCURR`

3.13.1.4 `uint8_t nRF24L01_RF_SETUP_t::pllLock`

3.13.1.5 `uint8_t nRF24L01_RF_SETUP_t::reserved`

3.13.1.6 `uint8_t nRF24L01_RF_SETUP_t::rfDR`

3.13.1.7 `uint8_t nRF24L01_RF_SETUP_t::rfPWR`

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

- Modules/[nRF24L01.h](#)

3.14 nRF24L01_RX_PW_P0_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- `struct {
 uint8_t rxPwP0:6
 uint8_t reserved:2
} b`
- `uint8_t B`

3.14.1 Field Documentation

3.14.1.1 `struct { ... } nRF24L01_RX_PW_P0_t::b`

3.14.1.2 `uint8_t nRF24L01_RX_PW_P0_t::B`

3.14.1.3 `uint8_t nRF24L01_RX_PW_P0_t::reserved`

3.14.1.4 `uint8_t nRF24L01_RX_PW_P0_t::rxPwP0`

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

- Modules/[nRF24L01.h](#)

3.15 nRF24L01_RX_PW_P1_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- struct {
 uint8_t rxPwP1:6
 uint8_t reserved:2
} b
- uint8_t B

3.15.1 Field Documentation

3.15.1.1 struct { ... } nRF24L01_RX_PW_P1_t::b

3.15.1.2 uint8_t nRF24L01_RX_PW_P1_t::B

3.15.1.3 uint8_t nRF24L01_RX_PW_P1_t::reserved

3.15.1.4 uint8_t nRF24L01_RX_PW_P1_t::rxPwP1

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

- Modules/[nRF24L01.h](#)

3.16 nRF24L01_RX_PW_P2_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- struct {
 uint8_t rxPwP2:6
 uint8_t reserved:2
} b
- uint8_t B

3.16.1 Field Documentation

3.16.1.1 struct { ... } nRF24L01_RX_PW_P2_t::b

3.16.1.2 uint8_t nRF24L01_RX_PW_P2_t::B

3.16.1.3 uint8_t nRF24L01_RX_PW_P2_t::reserved

3.16.1.4 uint8_t nRF24L01_RX_PW_P2_t::rxPwP2

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

- Modules/[nRF24L01.h](#)

3.17 nRF24L01_RX_PW_P3_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- struct {
 uint8_t [rxPwP3](#):6
 uint8_t [reserved](#):2
} [b](#)
- uint8_t [B](#)

3.17.1 Field Documentation

3.17.1.1 struct { ... } nRF24L01_RX_PW_P3_t::b

3.17.1.2 uint8_t nRF24L01_RX_PW_P3_t::B

3.17.1.3 uint8_t nRF24L01_RX_PW_P3_t::reserved

3.17.1.4 uint8_t nRF24L01_RX_PW_P3_t::rxPwP3

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

- Modules/[nRF24L01.h](#)

3.18 nRF24L01_RX_PW_P4_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- struct {
 uint8_t rxPwP4:6
 uint8_t reserved:2
} b
- uint8_t B

3.18.1 Field Documentation

3.18.1.1 struct { ... } nRF24L01_RX_PW_P4_t::b

3.18.1.2 uint8_t nRF24L01_RX_PW_P4_t::B

3.18.1.3 uint8_t nRF24L01_RX_PW_P4_t::reserved

3.18.1.4 uint8_t nRF24L01_RX_PW_P4_t::rxPwP4

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

- Modules/[nRF24L01.h](#)

3.19 nRF24L01_RX_PW_P5_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- struct {
 uint8_t rxPwP5:6
 uint8_t reserved:2
} b
- uint8_t B

3.19.1 Field Documentation

3.19.1.1 struct { ... } nRF24L01_RX_PW_P5_t::b

3.19.1.2 uint8_t nRF24L01_RX_PW_P5_t::B

3.19.1.3 uint8_t nRF24L01_RX_PW_P5_t::reserved

3.19.1.4 uint8_t nRF24L01_RX_PW_P5_t::rxPwP5

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

- Modules/[nRF24L01.h](#)

3.20 nRF24L01_SETUP_AW_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- struct {
 uint8_t [aw](#):2
 uint8_t [reserved](#):6
} [b](#)
- uint8_t [B](#)

3.20.1 Field Documentation

3.20.1.1 uint8_t nRF24L01_SETUP_AW_t::aw

3.20.1.2 struct { ... } nRF24L01_SETUP_AW_t::b

3.20.1.3 uint8_t nRF24L01_SETUP_AW_t::B

3.20.1.4 uint8_t nRF24L01_SETUP_AW_t::reserved

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

- Modules/[nRF24L01.h](#)

3.21 nRF24L01_SETUP_RETR_t Union Reference

```
#include <nRF24L01.h>
```

Data Fields

- struct {
 uint8_t [arc](#):4
 uint8_t [ard](#):4
} [b](#)
- uint8_t [B](#)

3.21.1 Field Documentation

3.21.1.1 `uint8_t nRF24L01_SETUP_RETR_t::arc`

3.21.1.2 `uint8_t nRF24L01_SETUP_RETR_t::ard`

3.21.1.3 `struct { ... } nRF24L01_SETUP_RETR_t::b`

3.21.1.4 `uint8_t nRF24L01_SETUP_RETR_t::B`

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

- Modules/[nRF24L01.h](#)

3.22 nRF24L01_SPIMessage_t Struct Reference

```
#include <nRF24L01.h>
```

Data Fields

- `uint8_t spiCh`
- `uint8_t command`
- `uint8_t data` [6]
- `uint8_t bytesToSend`

3.22.1 Field Documentation

3.22.1.1 `uint8_t nRF24L01_SPIMessage_t::bytesToSend`

3.22.1.2 `uint8_t nRF24L01_SPIMessage_t::command`

3.22.1.3 `uint8_t nRF24L01_SPIMessage_t::data`[6]

3.22.1.4 `uint8_t nRF24L01_SPIMessage_t::spiCh`

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

- Modules/[nRF24L01.h](#)

3.23 nRF24L01_STATUS_t Union Reference

```
#include <nRF24L01.h>
```


Data Fields

- struct {
 - uint8_t txFull:1
 - uint8_t rxPno:3
 - uint8_t maxRT:1
 - uint8_t txDS:1
 - uint8_t rxDR:1
 - uint8_t reserved:1
 } b
- uint8_t B

3.23.1 Field Documentation

3.23.1.1 struct { ... } nRF24L01_STATUS_t::b

3.23.1.2 uint8_t nRF24L01_STATUS_t::B

3.23.1.3 uint8_t nRF24L01_STATUS_t::maxRT

3.23.1.4 uint8_t nRF24L01_STATUS_t::reserved

3.23.1.5 uint8_t nRF24L01_STATUS_t::rxDR

3.23.1.6 uint8_t nRF24L01_STATUS_t::rxPno

3.23.1.7 uint8_t nRF24L01_STATUS_t::txDS

3.23.1.8 uint8_t nRF24L01_STATUS_t::txFull

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

- Modules/[nRF24L01.h](#)

3.24 TemperatureData Union Reference

```
#include <messaging.h>
```

Collaboration diagram for TemperatureData:

Data Fields

- uint8_t data [TEMP_MSG_BYTES]
- [TemperatureMessage_t](#) msg

3.24.1 Field Documentation

3.24.1.1 `uint8_t TemperatureData::data[TEMP_MSG_BYTES]`

3.24.1.2 `TemperatureMessage_t TemperatureData::msg`

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

- Messaging/[messaging.h](#)

3.25 TemperatureMessage Struct Reference

```
#include <messaging.h>
```

Data Fields

- `uint8_t currentTemp`
- `uint8_t currentDesired`
- `uint8_t currentRange`
- `uint8_t powerOn`
- `uint8_t checksum`
- `uint8_t cr`
- `uint8_t lf`

3.25.1 Field Documentation

3.25.1.1 `uint8_t TemperatureMessage::checksum`

3.25.1.2 `uint8_t TemperatureMessage::cr`

3.25.1.3 `uint8_t TemperatureMessage::currentDesired`

3.25.1.4 `uint8_t TemperatureMessage::currentRange`

3.25.1.5 `uint8_t TemperatureMessage::currentTemp`

3.25.1.6 `uint8_t TemperatureMessage::lf`

3.25.1.7 `uint8_t TemperatureMessage::powerOn`

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

- Messaging/[messaging.h](#)

Chapter 4

File Documentation

4.1 CircularBuffer/circularbuffer.c File Reference

```
#include "circularbuffer.h"
```

Include dependency graph for circularbuffer.c:

4.2 CircularBuffer/circularbuffer.h File Reference

```
#include "includeall.h"
```

Include dependency graph for circularbuffer.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [CircularBuffer_t](#)

Typedefs

- typedef struct [CircularBuffer_t](#) [CircularBuffer_t](#)
- typedef enum [BufferState](#) [BufferState_e](#)

Enumerations

- enum [BufferState](#) { [BUFFER_NOT_FULL](#) = 0, [BUFFER_FULL](#) = 1, [BUFFER_NOT_EMPTY](#) = 0, [BUFFER_EMPTY](#) = 1 }

Functions

- [CircularBuffer_t](#) * [CBufferInit](#) (uint32_t itemSize, uint32_t maxItems)

Function: *CBufferInit* *

- Description: *Initializes a circular buffer. **
- Parameters: *uint32_t itemSize: Size of each item in the buffer in bytes* uint32_t maxItems: Maximum number of items that the buffer * can hold. **
- Return Value: [CircularBuffer_t](#) *: *pointer to the circular buffer data * structure. **

- [BufferState_e](#) [IsBufferFull](#) ([CircularBuffer_t](#) *cb)

Function: *IsBufferFull* *

- Description: *Returns a status if the buffer is full or not **
- Parameters: [CircularBuffer_t](#) * cb: *pointer to the buffer being checked.**
- Return Value: [BufferState_e](#): *enumeration of buffer status. **

- [BufferState_e](#) [IsBufferEmpty](#) ([CircularBuffer_t](#) *cb)

Function: *IsBufferEmpty* *

- Description: *Returns a status if the buffer is empty or not **
- Parameters: [CircularBuffer_t](#) * cb: *pointer to the buffer being checked.**
- Return Value: [BufferState_e](#): *enumeration of buffer status. **

- [BufferState_e](#) [CBufferAdd](#) ([CircularBuffer_t](#) *cb, void *data, uint8_t DMAch)

Function: *CBufferAdd* *

- Description: *Adds a value to the buffer **
- Parameters: [CircularBuffer_t](#) * cb: *pointer to the buffer that the item * is being added to. * void * data: Pointer to the data being added to the buffer * uint8_t DMAch: DMA channel being used to transfer the data.* If DMAch is 0xFF, then software is used for * the transfer. **
- Return Value: [CBufferState_e](#): *enumeration of buffer status. **

- [BufferState_e](#) [CBufferAddItems](#) ([CircularBuffer_t](#) *cb, void *data, uint32_t numToAdd, uint8_t DMAch)

Function: *CBufferAdd* *

- Description: *Adds multiple values to the buffer with the use of DMA. * The function currently does not work and is very * complicated. This would help improve performance of the * DMA transfers if needed. **
- Parameters: [CircularBuffer_t](#) * cb: *pointer to the buffer that the item * is being added to. * void * data: Pointer to the data being added to the buffer * uint32_t numToAdd: Number of items to be added. * uint8_t DMAch: DMA channel being used to transfer the data.* If DMAch is 0xFF, then software is used for * the transfer. **
- Return Value: [CBufferState_e](#): *enumeration of buffer status. **

- [BufferState_e](#) [CBufferRemove](#) ([CircularBuffer_t](#) *cb, void *data, uint8_t DMAch)

Function: *CBufferRemove* *

- Description: *Removes a value from the buffer **
- Parameters: [CircularBuffer_t](#) * cb: *pointer to the buffer that the item * is being removed from. * void * data: Pointer to memory that the item will be placed* uint8_t DMAch: DMA channel being used to transfer the data.* If DMAch is 0xFF, then software is used for * the transfer. **
- Return Value: [CBufferState_e](#): *enumeration of buffer status. **

- void [CBufferDestruct](#) ([CircularBuffer_t](#) **cb)

Function: *CBufferDestruct* *

- Description: *Destructs a circular buffer and frees the memory that was * allocated to the buffer. **
- Parameters: [CircularBuffer_t](#) ** pcb: *Double pointer to buffer to be * freed. **
- Return Value: *NONE **

4.2.1 Typedef Documentation

4.2.1.1 typedef enum BufferState BufferState_e

4.2.1.2 typedef struct CircularBuffer_t CircularBuffer_t

4.2.2 Enumeration Type Documentation

4.2.2.1 enum BufferState

Enumerator

BUFFER_NOT_FULL
BUFFER_FULL
BUFFER_NOT_EMPTY
BUFFER_EMPTY

4.2.3 Function Documentation

4.2.3.1 BufferState_e CBufferAdd (CircularBuffer_t * cb, void * data, uint8_t DMAch) [inline]

Function: CBufferAdd *

- Description: Adds a value to the buffer *
- Parameters: [CircularBuffer_t](#) * cb: pointer to the buffer that the item * is being added to. * void * data: Pointer to the data being added to the buffer * uint8_t DMAch: DMA channel being used to transfer the data.* If DMAch is 0xFF, then software is used for * the transfer. *
- Return Value: CBufferState_e: enumeration of buffer status. *

4.2.3.2 BufferState_e CBufferAddItems (CircularBuffer_t * cb, void * data, uint32_t numToAdd, uint8_t DMAch)

Function: CBufferAdd *

- Description: Adds multiple values to the buffer with the use of DMA. * The function currently does not work and is very * complicated. This would help improve performance of the * DMA transfers if needed. *
- Parameters: [CircularBuffer_t](#) * cb: pointer to the buffer that the item * is being added to. * void * data: Pointer to the data being added to the buffer * uint32_t numToAdd: Number of items to be added. * uint8_t DMAch: DMA channel being used to transfer the data.* If DMAch is 0xFF, then software is used for * the transfer. *
- Return Value: CBufferState_e: enumeration of buffer status. *

4.2.3.3 void CBufferDestruct (CircularBuffer_t ** cb)

Function: CBufferDestruct *

- Description: Destructs a circular buffer and frees the memory that was * allocated to the buffer. *
- Parameters: [CircularBuffer_t](#) ** pcb: Double pointer to buffer to be * freed. *
- Return Value: NONE *

4.2.3.4 `CircularBuffer_t* CBufferInit (uint32_t itemSize, uint32_t maxItems)`

Function: CBufferInit *

- Description: Initializes a circular buffer. *
- Parameters: `uint32_t itemSize`: Size of each item in the buffer in bytes* `uint32_t maxItems`: Maximum number of items that the buffer * can hold. *
- Return Value: `CircularBuffer_t` *: pointer to the circular buffer data * structure. *

4.2.3.5 `BufferState_e CBufferRemove (CircularBuffer_t * cb, void * data, uint8_t DMAch)` `[inline]`

Function: CBufferRemove *

- Description: Removes a value from the buffer *
- Parameters: `CircularBuffer_t` * *cb*: pointer to the buffer that the item * is being removed from. * `void` * *data*: Pointer to memory that the item will be placed* `uint8_t DMAch`: DMA channel being used to transfer the data.* If *DMAch* is 0xFF, then software is used for * the transfer. *
- Return Value: `BufferState_e`: enumeration of buffer status. *

4.2.3.6 `BufferState_e IsBufferEmpty (CircularBuffer_t * cb)` `[inline]`

Function: IsBufferEmpty *

- Description: Returns a status if the buffer is empty or not *
- Parameters: `CircularBuffer_t` * *cb*: pointer to the buffer being checked.*
- Return Value: `BufferState_e`: enumeration of buffer status. *

4.2.3.7 `BufferState_e IsBufferFull (CircularBuffer_t * cb)` `[inline]`

Function: IsBufferFull *

- Description: Returns a status if the buffer is full or not *
- Parameters: `CircularBuffer_t` * *cb*: pointer to the buffer being checked.*
- Return Value: `BufferState_e`: enumeration of buffer status. *

4.3 Data/data.c File Reference

```
#include "data.h"
```

Include dependency graph for data.c:

Functions

- `int8_t Myltoa (uint8_t *str, int32_t data, int32_t base)`

Function: Myltoa *

- Description: Inputs a integer value and coverts into a stirng. *
- Parameters: `uint8_t * str`: pointer to a string * `int32_t data`: interger to be converted into a string. * `int32_t base`: What base the integer is going to be * converted to * Return Value: `int8_t`: pass/fail value. A return of 0 is a successfull * conversion. Anything else is a failure. *

- `int32_t MyAtoi (uint8_t *str)`

Function: MyAtoi *

- Description: Inputs a string and coverts into a signed interger. *
- Parameters: `uint8_t * str`: pointer to a string to be converted. *
- Return Value: `int32_t`: integer value converted from the string input *

- `int32_t MyFtoa (uint8_t *str, double data, int32_t decimalPlaces)`

Function: MyFtoi *

- Description: Inputs floating point number and converts it to a string. *
- Parameters: `uint8_t * str`: pointer to a string buffer that is filled. * `float data`: floating point number to be converted * `int32_t decimalPlaces`: How many decimal places to be * to be converted to a string. *
- Return Value: `int32_t`: integer value converted from the string input *

- `void DumpMemory (uint8_t *start, uint32_t length)`

Function: DumpMemory *

- Description: Prints whats stored in memory starting at pointed * passed in for the length passed in. *
- Parameters: `uint8_t * start`: pointer to the start of memory desired to * be printed. * `uint32_t length`: Number of bytes to be printed * Return Value: void *

- `int32_t BigToLittle (int32_t data)`

Function: BigToLittle *

- Description: Converts a memory locaiton from big endian to * little endian. it is left to the user to know theat the * data being passed in is in big endian *
- Parameters: `uint8_t32`: Value that is in big endian to be converted * to little endian. *
- Return Value: `int32_t`: Return value of the resulting conversion *

- `int32_t LittleToBig (int32_t data)`

Function: littleToBig *

- Description: Converts a memory locaiton from little endian to * little endian. it is left to the user to know theat the * data being passed in is in little endian *
- Parameters: `uint8_t32`: Value that is in little endian to be converted * to big endian. *
- Return Value: `int32_t`: Return value of the resulting conversion *

4.3.1 Function Documentation

4.3.1.1 `int32_t BigToLittle (int32_t data)`

Function: BigToLittle *

- Description: Converts a memory locaiton from big endian to * little endian. it is left to the user to know theat the * data being passed in is in big endian *
- Parameters: `uint8_t32`: Value that is in big endian to be converted * to little endian. *
- Return Value: `int32_t`: Return value of the resulting conversion *

4.3.1.2 void DumpMemory (uint8_t * start, uint32_t length)

Function: DumpMemory *

- Description: Prints whats stored in memory starting at pointed * passed in for the length passed in. *
- Parameters: uint8_t * start: pointer to the start of memory desired to * be printed. * uint32_t length: Number of bytes to be printed * Return Value: void *

4.3.1.3 int32_t LittleToBig (int32_t data)

Function: littleToBig *

- Description: Converts a memory locaiton from little endian to * little endian. it is left to the user to know theat the * data being passed in is in little endian *
- Parameters: uint8_t32: Value that is in little endian to be converted * to big endian. *
- Return Value: int32_t: Return value of the resulting conversion *

4.3.1.4 int32_t MyAtoi (uint8_t * str)

Function: MyAtoi *

- Description: Inputs a string and coverts into a signed interger. *
- Parameters: uint8_t * str: pointer to a string to be converted. *
- Return Value: int32_t: integer value converted from the string input *

4.3.1.5 int32_t MyFtoa (uint8_t * str, double data, int32_t decimalPlaces)

Function: MyFtoi *

- Description: Inputs floating point number and converts it to a string. *
- Parameters: uint8_t * str: pointer to a string buffer that is filled. * float data: floating point number to be converted * int32_t decimalPlaces: How many decimal places to be * to be converted to a string. *
- Return Value: int32_t: integer value converted from the string input *

4.3.1.6 int8_t Myltoa (uint8_t * str, int32_t data, int32_t base)

Function: Myltoa *

- Description: Inputs a integer value and coverts into a stirng. *
- Parameters: uint8_t * str: pointer to a string * int32_t data: interger to be converted into a string. * int32_t base: What base the integer is going to be * converted to * Return Value: int8_t: pass/fail value. A return of 0 is a successfull * conversion. Anything else is a failure. *

4.4 Data/data.h File Reference

```
#include "includeall.h"
```

Include dependency graph for data.h: This graph shows which files directly or indirectly include this file:

Macros

- #define [BYTE0_MASK](#) 0x000000FF
- #define [BYTE1_MASK](#) 0x0000FF00
- #define [BYTE2_MASK](#) 0x00FF0000
- #define [BYTE3_MASK](#) 0xFF000000
- #define [BYTE3_SHIFT](#) 24
- #define [BYTE2_SHIFT](#) 16
- #define [BYTE1_SHIFT](#) 8
- #define [BYTE0_SHIFT](#) 0
- #define [ASCIINUMBASE](#) 0x30
- #define [ASCIILETTERBASE](#) 0x37

Functions

- [int8_t Myltoa](#) ([uint8_t](#) *str, [int32_t](#) data, [int32_t](#) base)
 - Function: *Myltoa* *
 - Description: Inputs a integer value and coverts into a stirng. *
 - Parameters: *uint8_t* * str: pointer to a string * *int32_t* data: interger to be converted into a string. * *int32_t* base: What base the integer is going to be * converted to * Return Value: *int8_t*: pass/fail value. A return of 0 is a successfull * conversion. Anything else is a failure. *
- [int32_t MyAtoi](#) ([uint8_t](#) *str)
 - Function: *MyAtoi* *
 - Description: Inputs a string and coverts into a signed interger. *
 - Parameters: *uint8_t* * str: pointer to a string to be converted. *
 - Return Value: *int32_t*: integer value converted from the string input *
- [int32_t MyFtoa](#) ([uint8_t](#) *str, double data, [int32_t](#) decimalPlaces)
 - Function: *MyFtoi* *
 - Description: Inputs floating point number and converts it to a string. *
 - Parameters: *uint8_t* * str: pointer to a string buffer that is filled. * float data: floating point number to be converted * *int32_t* decimalPlaces: How many decimal places to be * to be converted to a string. *
 - Return Value: *int32_t*: integer value converted from the string input *
- void [DumpMemory](#) ([uint8_t](#) *start, [uint32_t](#) length)
 - Function: *DumpMemory* *
 - Description: Prints whats stored in memory starting at pointed * passed in for the length passed in. *
 - Parameters: *uint8_t* * start: pointer to the start of memory desired to * be printed. * *uint32_t* length: Number of bytes to be printed * Return Value: void *
- [int32_t BigToLittle](#) ([int32_t](#) data)
 - Function: *BigToLittle* *
 - Description: Converts a memory locaiton from big endian to * little endian. it is left to the user to know theat the * data being passed in is in big endian *
 - Parameters: *uint8_t*32: Value that is in big endian to be converted * to little endian. *
 - Return Value: *int32_t*: Return value of the resulting conversion *
- [int32_t LittleToBig](#) ([int32_t](#) data)
 - Function: *littleToBig* *
 - Description: Converts a memory locaiton from little endian to * little endian. it is left to the user to know theat the * data being passed in is in little endian *
 - Parameters: *uint8_t*32: Value that is in little endian to be converted * to big endian. *
 - Return Value: *int32_t*: Return value of the resulting conversion *

4.4.1 Macro Definition Documentation

4.4.1.1 `#define ASCIILETTERBASE 0x37`

4.4.1.2 `#define ASCIINUMBASE 0x30`

4.4.1.3 `#define BYTE0_MASK 0x000000FF`

4.4.1.4 `#define BYTE0_SHIFT 0`

4.4.1.5 `#define BYTE1_MASK 0x0000FF00`

4.4.1.6 `#define BYTE1_SHIFT 8`

4.4.1.7 `#define BYTE2_MASK 0X00FF0000`

4.4.1.8 `#define BYTE2_SHIFT 16`

4.4.1.9 `#define BYTE3_MASK 0xFF000000`

4.4.1.10 `#define BYTE3_SHIFT 24`

4.4.2 Function Documentation

4.4.2.1 `int32_t BigToLittle (int32_t data)`

Function: BigToLittle *

- Description: Converts a memory locaiton from big endian to * little endian. it is left to the user to know theat the * data being passed in is in big endian *
- Parameters: `uint8_t32`: Value that is in big endian to be converted * to little endian. *
- Return Value: `int32_t`: Return value of the resulting conversion *

4.4.2.2 `void DumpMemory (uint8_t * start, uint32_t length)`

Function: DumpMemory *

- Description: Prints whats stored in memory starting at pointed * passed in for the length passed in. *
- Parameters: `uint8_t * start`: pointer to the start of memory desired to * be printed. * `uint32_t length`: Number of bytes to be printed * Return Value: `void` *

4.4.2.3 int32_t LittleToBig (int32_t data)

Function: littleToBig *

- Description: Converts a memory locaiton from little endian to * little endian. it is left to the user to know theat the * data being passed in is in little endian *
- Parameters: uint8_t32: Value that is in little endian to be converted * to big endian. *
- Return Value: int32_t: Return value of the resulting conversion *

4.4.2.4 int32_t MyAtoi (uint8_t * str)

Function: MyAtoi *

- Description: Inputs a string and coverts into a signed interger. *
- Parameters: uint8_t * str: pointer to a string to be converted. *
- Return Value: int32_t: integer value converted from the string input *

4.4.2.5 int32_t MyFtoa (uint8_t * str, double data, int32_t decimalPlaces)

Function: MyFtoi *

- Description: Inputs floating point number and converts it to a string. *
- Parameters: uint8_t * str: pointer to a string buffer that is filled. * float data: floating point number to be converted * int32_t decimalPlaces: How many decimal places to be * to be converted to a string. *
- Return Value: int32_t: integer value converted from the string input *

4.4.2.6 int8_t Myltoa (uint8_t * str, int32_t data, int32_t base)

Function: Myltoa *

- Description: Inputs a integer value and coverts into a stirng. *
- Parameters: uint8_t * str: pointer to a string * int32_t data: interger to be converted into a string. * int32_t base: What base the integer is going to be * converted to * Return Value: int8_t: pass/fail value. A return of 0 is a successfull * conversion. Anything else is a failure. *

4.5 Display/sevensegment.c File Reference

```
#include "sevensegment.h"
Include dependency graph for sevensegment.c:
```

Functions

- void `InitDisplay (Display_place place)`

Function: `InitDisplay` *

- Description: Initializes one of the seven segment displays. Sets up * necessities registers as GPIO creates a value that will * translate a number into lighting up the correct lights to * show that number. *
- Parameters: `Display_place displayNum`: Which display is being * initialized *
- Return Value: `NONE` *

- void `UpdateDisplay (Display_place displayNum, uint8_t value)`

Function: `UpdateDisplay` *

- Description: Takes an input of which display is being updated and what * the numbers that needs to be displayed. *
- Parameters: `Display_place displayNum`: Which display is being updated * `uint8_t value`: number to be shown on the display. *
- Return Value: `NONE` *

- void `Display_ClearAll (Display_place displayNum)`

Function: `ClearDisplay` *

- Description: Turns off all leds of the display indicated by the input *
- Parameters: `uint8_t displayNum`: Which display is being initialized *
- Return Value: `NONE` *

4.5.1 Function Documentation

4.5.1.1 void `Display_ClearAll (Display_place displayNum)`

Function: `ClearDisplay` *

- Description: Turns off all leds of the display indicated by the input *
- Parameters: `uint8_t displayNum`: Which display is being initialized *
- Return Value: `NONE` *

4.5.1.2 void `InitDisplay (Display_place place)`

Function: `InitDisplay` *

- Description: Initializes one of the seven segment displays. Sets up * necessities registers as GPIO creates a value that will * translate a number into lighting up the correct lights to * show that number. *
- Parameters: `Display_place displayNum`: Which display is being * initialized *
- Return Value: `NONE` *

4.5.1.3 void `UpdateDisplay (Display_place displayNum, uint8_t value)`

Function: `UpdateDisplay` *

- Description: Takes an input of which display is being updated and what * the numbers that needs to be displayed. *
- Parameters: `Display_place displayNum`: Which display is being updated * `uint8_t value`: number to be shown on the display. *
- Return Value: `NONE` *

4.6 Display/sevensegment.h File Reference

```
#include "includeall.h"
```

Include dependency graph for sevensegment.h: This graph shows which files directly or indirectly include this file:

Macros

- `#define NUM_DISPLAYS 3`
- `#define NUM_SEGMENTS 8`
- `#define NUM_DIGITS 10`
- `#define DISPLAY_CLEAR 0`
- `#define DISPLAY_SET 1`
- `#define DISPLAY_PORTE_A PORTE_PCR30`
- `#define DISPLAY_PORTE_B PORTE_PCR29`
- `#define DISPLAY_PORTE_C PORTE_PCR23`
- `#define DISPLAY_PORTE_D PORTE_PCR22`
- `#define DISPLAY_PORTE_E PORTE_PCR21`
- `#define DISPLAY_PORTE_F PORTE_PCR20`
- `#define DISPLAY_PORTE_G PORTE_PCR5`
- `#define DISPLAY_PORTE_H PORTE_PCR2`
- `#define DISPLAY_A_PORTE_PIN 0x40000000`
- `#define DISPLAY_B_PORTE_PIN 0x20000000`
- `#define DISPLAY_C_PORTE_PIN 0x00800000`
- `#define DISPLAY_D_PORTE_PIN 0x00400000`
- `#define DISPLAY_E_PORTE_PIN 0x00200000`
- `#define DISPLAY_F_PORTE_PIN 0x00100000`
- `#define DISPLAY_G_PORTE_PIN 0x00000020`
- `#define DISPLAY_H_PORTE_PIN 0x00000002`
- `#define DISPLAY_PORTB_A PORTB_PCR3`
- `#define DISPLAY_PORTB_B PORTB_PCR2`
- `#define DISPLAY_PORTB_C PORTB_PCR1`
- `#define DISPLAY_PORTB_D PORTB_PCR0`
- `#define DISPLAY_PORTB_E PORTB_PCR11`
- `#define DISPLAY_PORTB_F PORTB_PCR10`
- `#define DISPLAY_PORTB_G PORTB_PCR9`
- `#define DISPLAY_PORTB_H PORTB_PCR8`
- `#define DISPLAY_A_PORTB_PIN 0x00000008`
- `#define DISPLAY_B_PORTB_PIN 0x00000004`
- `#define DISPLAY_C_PORTB_PIN 0x00000002`
- `#define DISPLAY_D_PORTB_PIN 0x00000001`
- `#define DISPLAY_E_PORTB_PIN 0x00000800`
- `#define DISPLAY_F_PORTB_PIN 0x00000400`
- `#define DISPLAY_G_PORTB_PIN 0x00000200`
- `#define DISPLAY_H_PORTB_PIN 0x00000100`

Enumerations

- `enum Display_Errors { Display_NoError = 0, Display_NotSingleDigit }`
- `enum Display_place { Display_Tens = 0, Display_Ones }`

Functions

- void [InitDisplay](#) ([Display_place](#) place)

*Function: InitDisplay **

- *Description: Initializes one of the seven segment displays. Sets up * necessities registers as GPIO creates a value that will * translate a number into lighting up the correct lights to * show that number. **
- *Parameters: Display_place displayNum: Which display is being * initialized **
- *Return Value: NONE **

- void [UpdateDisplay](#) ([Display_place](#) place, uint8_t value)

*Function: UpdateDisplay **

- *Description: Takes an input of which display is being updated and what * the numbers that needs to be displayed. **
- *Parameters: Display_place displayNum: Which display is being updated * uint8_t value: number to be shown on the display. **
- *Return Value: NONE **

- void [Display_ClearAll](#) ([Display_place](#) place)

*Function: ClearDisplay **

- *Description: Turns off all leds of the display indicated by the input **
- *Parameters: uint8_t displayNum: Which display is being initialized **
- *Return Value: NONE **

Variables

- GPIO_Type * [Display_Port](#) [NUM_DISPLAYS]
- uint32_t [Display_Value](#) [NUM_DISPLAYS][NUM_DIGITS]

4.6.1 Macro Definition Documentation

4.6.1.1 `#define DISPLAY_A_PORTB_PIN 0x00000008`

4.6.1.2 `#define DISPLAY_A_PORTE_PIN 0x40000000`

4.6.1.3 `#define DISPLAY_B_PORTB_PIN 0x00000004`

4.6.1.4 `#define DISPLAY_B_PORTE_PIN 0x20000000`

4.6.1.5 `#define DISPLAY_C_PORTB_PIN 0x00000002`

4.6.1.6 `#define DISPLAY_C_PORTE_PIN 0x00800000`

4.6.1.7 `#define DISPLAY_CLEAR 0`

4.6.1.8 `#define DISPLAY_D_PORTB_PIN 0x00000001`

4.6.1.9 `#define DISPLAY_D_PORTE_PIN 0x00400000`

4.6.1.10 `#define DISPLAY_E_PORTB_PIN 0x00000800`

4.6.1.11 `#define DISPLAY_E_PORTE_PIN 0x00200000`

4.6.1.12 `#define DISPLAY_F_PORTB_PIN 0x00000400`

4.6.1.13 `#define DISPLAY_F_PORTE_PIN 0x00100000`

4.6.1.14 `#define DISPLAY_G_PORTB_PIN 0x00000200`

4.6.1.15 `#define DISPLAY_G_PORTE_PIN 0x00000020`

4.6.1.16 `#define DISPLAY_H_PORTB_PIN 0x00000100`

4.6.1.17 `#define DISPLAY_H_PORTE_PIN 0x00000002`

4.6.1.18 `#define DISPLAY_PORTB_A PORTB_PCR3`

4.6.1.19 `#define DISPLAY_PORTB_B PORTB_PCR2`

4.6.1.20 `#define DISPLAY_PORTB_C PORTB_PCR1`

4.6.1.21 `#define DISPLAY_PORTB_D PORTB_PCR0`

4.6.1.22 `#define DISPLAY_PORTB_E PORTB_PCR11`

4.6.1.23 `#define DISPLAY_PORTB_F PORTB_PCR10`

4.6.1.24 `#define DISPLAY_PORTB_G PORTB_PCR9`

4.6.1.25 `#define DISPLAY_PORTB_H PORTB_PCR8`

4.6.1.26 `#define DISPLAY_PORTE_A PORTE_PCR30`

4.6.1.27 `#define DISPLAY_PORTE_B PORTE_PCR29`

4.6.1.28 `#define DISPLAY_PORTE_C PORTE_PCR23`

4.6.1.29 `#define DISPLAY_PORTE_D PORTE_PCR22`

4.6.1.30 `#define DISPLAY_PORTE_E PORTE_PCR21`

4.6.1.31 `#define DISPLAY_PORTE_F PORTE_PCR20`

4.6.1.32 `#define DISPLAY_PORTE_G PORTE_PCR5`

4.6.1.33 `#define DISPLAY_PORTE_H PORTE_PCR2`

4.6.1.34 `#define DISPLAY_SET 1`

4.6.1.35 `#define NUM_DIGITS 10`

4.6.1.36 `#define NUM_DISPLAYS 3`

4.6.1.37 `#define NUM_SEGMENTS 8`

4.6.2 Enumeration Type Documentation

4.6.2.1 `enum Display_Errors`

Enumerator

Display_NoError

Display_NotSingleDigit

4.6.2.2 `enum Display_place`

Enumerator

Display_Tens

Display_Ones

4.6.3 Function Documentation

4.6.3.1 `void Display_ClearAll (Display_place place)`

Function: `ClearDisplay` *

- Description: Turns off all leds of the display indicated by the input *
- Parameters: `uint8_t displayNum`: Which display is being initialized *
- Return Value: `NONE` *

4.6.3.2 `void InitDisplay (Display_place place)`

Function: `InitDisplay` *

- Description: Initializes one of the seven segment displays. Sets up * necessities registers as GPIO creates a value that will * translate a number into lighting up the correct lights to * show that number. *
- Parameters: `Display_place displayNum`: Which display is being * initialized *
- Return Value: `NONE` *

4.6.3.3 void UpdateDisplay (Display_place place, uint8_t value)

Function: UpdateDisplay *

- Description: Takes an input of which display is being updated and what * the numbers that needs to be displayed. *
- Parameters: Display_place displayNum: Which display is being updated * uint8_t value: number to be shown on the display. *
- Return Value: NONE *

4.6.4 Variable Documentation

4.6.4.1 GPIO_Type* Display_Port[NUM_DISPLAYS]

4.6.4.2 uint32_t Display_Value[NUM_DISPLAYS][NUM_DIGITS]

4.7 Logging/uartlogging.c File Reference

```
#include "uartlogging.h"
```

Include dependency graph for uartlogging.c:

4.8 Logging/uartlogging.h File Reference

```
#include "includeall.h"
```

Include dependency graph for uartlogging.h: This graph shows which files directly or indirectly include this file:

Typedefs

- typedef enum [DataType_t](#) [DataType_t](#)

Enumerations

- enum [DataType_t](#) { [int_e](#) = 0, [double_e](#) }

Functions

- void [LOG0](#) (uint8_t *[data](#))
- void [LOG1](#) (uint8_t *[data](#), uint32_t length, [DataType_t](#) [dataType](#), uint32_t numParms,...)

4.8.1 Typedef Documentation

4.8.1.1 typedef enum DataType_t DataType_t

4.8.2 Enumeration Type Documentation

4.8.2.1 enum DataType_t

Enumerator

int_e

double_e

4.8.3 Function Documentation

4.8.3.1 void LOG0 (uint8_t * data)

4.8.3.2 void LOG1 (uint8_t * data, uint32_t length, DataType_t dataType, uint32_t numParms, ...)

4.9 Main/includeall.h File Reference

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <time.h>
#include <math.h>
#include "circularbuffer.h"
#include "uart.h"
#include "messaging.h"
#include "nRF24L01.h"
#include "uartlogging.h"
#include "led.h"
#include "memory.h"
#include "data.h"
#include "diags.h"
```

Include dependency graph for includeall.h: This graph shows which files directly or indirectly include this file:

Macros

- #define [NO_DMA](#) 0xFF
- #define [UART_LOGGING](#)

4.9.1 Macro Definition Documentation

4.9.1.1 `#define NO_DMA 0xFF`

4.9.1.2 `#define UART_LOGGING`

4.10 Main/main.c File Reference

```
#include "includeall.h"
Include dependency graph for main.c:
```

Functions

- int [main](#) ()

Variables

- [CircularBuffer_t](#) * [UART0_RXBuffer](#)
- [CircularBuffer_t](#) * [UART0_TXBuffer](#)
- [CircularBuffer_t](#) * [UART1_RXBuffer](#)
- [CircularBuffer_t](#) * [UART1_TXBuffer](#)
- [CircularBuffer_t](#) * [SPI_RXBuffer](#) [[SPI_CHANNELS](#)]
- [CircularBuffer_t](#) * [SPI_TXBuffer](#) [[SPI_CHANNELS](#)]

4.10.1 Function Documentation

4.10.1.1 `int main ()`

4.10.2 Variable Documentation

4.10.2.1 `CircularBuffer_t* SPI_RXBuffer[SPI_CHANNELS]`

4.10.2.2 `CircularBuffer_t* SPI_TXBuffer[SPI_CHANNELS]`

4.10.2.3 `CircularBuffer_t* UART0_RXBuffer`

4.10.2.4 `CircularBuffer_t* UART0_TXBuffer`

4.10.2.5 `CircularBuffer_t* UART1_RXBuffer`

4.10.2.6 `CircularBuffer_t* UART1_TXBuffer`

4.11 Memory/memory.c File Reference

```
#include "memory.h"
Include dependency graph for memory.c:
```

Functions

- `int8_t MyMemMove (uint8_t *src, uint8_t *dst, uint32_t numBytes, uint8_t DMACh)`
 Function: *MyMemMove* *
 - Description: Moves a portion of memory to another location in memory. *
 - Parameters: *uint8_t * src*: Start of memory to be moved. * *uint8_t * dst*: Start of memory to be copied to. * *int32_t length*: Number of bytes to be moved. *
 - Return Value: *int8_t*: pass/fail value. Success is a 0 value, all * values are a failure. *
- `int8_t MyMemSet (uint8_t *dst, uint32_t value, size_t numBytes, uint8_t DMACh)`
 Function: *MyMemSet* *
 - Description: Moves a value to a number of bytes in memory using DMA. *
 - Parameters: *uint8_t * src*: Start of memory to be set * *uint32_t value*: value to set in memory *int32_t length*: Number of bytes to be set *
 - Return Value: *int8_t*: pass/fail value. Success is a 0 value, all * values are a failure. *
- `int8_t MyReverse (uint8_t *src, uint32_t length)`
 Function: *MyReverse* *
 - Description: Reverses the bytes starting at a location given *
 - Parameters: *uint8_t * src*: Start of memory to be reversed * *int32_t length*: Number of bytes to be reversed *
 - Return Value: *int8_t*: pass/fail value. Success is a 0 value, all * values are a failure. *
- `int32_t MyStrLen (uint8_t *str)`
 Function: *MyStringLength* *
 - Description: Reverses the bytes starting at a location given *
 - Parameters: *uint8_t * src*: Start of memory to be reversed * *int32_t length*: Number of bytes to be reversed *
 - Return Value: *int8_t*: pass/fail value. Success is a 0 value, all * values are a failure. *

Variables

- `uint8_t dmaComplete` [4]

4.11.1 Function Documentation

4.11.1.1 `int8_t MyMemMove (uint8_t * src, uint8_t * dst, uint32_t numBytes, uint8_t DMACh)`

Function: *MyMemMove* *

- Description: Moves a portion of memory to another location in memory. *
- Parameters: *uint8_t * src*: Start of memory to be moved. * *uint8_t * dst*: Start of memory to be copied to. * *int32_t length*: Number of bytes to be moved. *
- Return Value: *int8_t*: pass/fail value. Success is a 0 value, all * values are a failure. *

4.11.1.2 `int8_t MyMemSet (uint8_t * dst, uint32_t value, size_t numBytes, uint8_t DMACh)`

Function: *MyMemSet* *

- Description: Moves a value to a number of bytes in memory using DMA. *
- Parameters: *uint8_t * src*: Start of memory to be set * *uint32_t value*: value to set in memory *int32_t length*: Number of bytes to be set *
- Return Value: *int8_t*: pass/fail value. Success is a 0 value, all * values are a failure. *

4.11.1.3 `int8_t MyReverse (uint8_t * src, uint32_t length)`Function: `MyReverse` *

- Description: Reverses the bytes starting at a location given *
- Parameters: `uint8_t * src`: Start of memory to be reversed * `int32_t length`: Number of bytes to be reversed *
- Return Value: `int8_t`: pass/fail value. Success is a 0 value, all * values are a failure. *

4.11.1.4 `int32_t MyStrLen (uint8_t * str)`Function: `MyStringLength` *

- Description: Reverses the bytes starting at a location given *
- Parameters: `uint8_t * src`: Start of memory to be reversed * `int32_t length`: Number of bytes to be reversed *
- Return Value: `int8_t`: pass/fail value. Success is a 0 value, all * values are a failure. *

4.11.2 Variable Documentation

4.11.2.1 `uint8_t dmaComplete[4]`

4.12 Memory/memory.h File Reference

#include "includeall.h"

Include dependency graph for memory.h: This graph shows which files directly or indirectly include this file:

Functions

- `int8_t MyMemMove (uint8_t *src, uint8_t *dst, uint32_t length, uint8_t DMAch)`
Function: `MyMemMove` *
 - Description: Moves a portion of memory to another location in memory. *
 - Parameters: `uint8_t * src`: Start of memory to be moved. * `uint8_t * dst`: Start of memory to be copied to. * `int32_t length`: Number of bytes to be moved. *
 - Return Value: `int8_t`: pass/fail value. Success is a 0 value, all * values are a failure. *
- `int8_t MyMemSet (uint8_t *dst, uint32_t value, size_t numBytes, uint8_t DMAch)`
Function: `MyMemSet` *
 - Description: Moves a value to a number of bytes in memory using DMA. *
 - Parameters: `uint8_t * src`: Start of memory to be set * `uint32_t value`: value to set in memory `int32_t length`: Number of bytes to be set *
 - Return Value: `int8_t`: pass/fail value. Success is a 0 value, all * values are a failure. *
- `int8_t MyReverse (uint8_t *src, uint32_t length)`
Function: `MyReverse` *
 - Description: Reverses the bytes starting at a location given *
 - Parameters: `uint8_t * src`: Start of memory to be reversed * `int32_t length`: Number of bytes to be reversed *
 - Return Value: `int8_t`: pass/fail value. Success is a 0 value, all * values are a failure. *
- `int32_t MyStrLen (uint8_t *str)`
Function: `MyStringLength` *
 - Description: Reverses the bytes starting at a location given *
 - Parameters: `uint8_t * src`: Start of memory to be reversed * `int32_t length`: Number of bytes to be reversed *
 - Return Value: `int8_t`: pass/fail value. Success is a 0 value, all * values are a failure. *

4.12.1 Function Documentation

4.12.1.1 `int8_t MyMemMove (uint8_t * src, uint8_t * dst, uint32_t length, uint8_t DMACh)`

Function: MyMemMove *

- Description: Moves a portion of memory to another location in memory. *
- Parameters: `uint8_t * src`: Start of memory to be moved. * `uint8_t * dst`: Start of memory to be copied to. * `uint32_t length`: Number of bytes to be moved. *
- Return Value: `int8_t`: pass/fail value. Success is a 0 value, all * values are a failure. *

4.12.1.2 `int8_t MyMemSet (uint8_t * dst, uint32_t value, size_t numBytes, uint8_t DMACh)`

Function: MyMemSet *

- Description: Moves a value to a number of bytes in memory using DMA. *
- Parameters: `uint8_t * src`: Start of memory to be set * `uint32_t value`: value to set in memory `uint32_t length`: Number of bytes to be set *
- Return Value: `int8_t`: pass/fail value. Success is a 0 value, all * values are a failure. *

4.12.1.3 `int8_t MyReverse (uint8_t * src, uint32_t length)`

Function: MyReverse *

- Description: Reverses the bytes starting at a location given *
- Parameters: `uint8_t * src`: Start of memory to be reversed * `uint32_t length`: Number of bytes to be reversed *
- Return Value: `int8_t`: pass/fail value. Success is a 0 value, all * values are a failure. *

4.12.1.4 `int32_t MyStrLen (uint8_t * str)`

Function: MyStringLength *

- Description: Reverses the bytes starting at a location given *
- Parameters: `uint8_t * src`: Start of memory to be reversed * `uint32_t length`: Number of bytes to be reversed *
- Return Value: `int8_t`: pass/fail value. Success is a 0 value, all * values are a failure. *

4.13 Messaging/messaging.c File Reference

```
#include "messaging.h"
```

Include dependency graph for messaging.c:

Functions

- [MessagingErrors_e BuildCommandMessage](#) ([Commands_e cmd](#), [uint8_t data](#))

Function: BuildCommandMesage *

- Description: Takes a command number and the input value and creates * the message to be sent to the FRDM board. *
- Parameters: [Commands_e cmd](#): Command to be sent in the message. * [uint8_t data](#): Input value of the command. *
- Return Value: [MessagingErrors_e](#): Enumeration of messaging errors. *
-

- [MessagingErrors_e CalculateCommandChecksum](#) ([CommandMessage_t *msg](#))

Function: CalculateCommandChecksum *

- Description: Creates the checksum for the command message. *
- Parameters: [CommandMessage_t * msg](#): Pointer to a command message struct*
- Return Value: [MessagingErrors_e](#): Enumeration of messaging errors. *
-

- [MessagingErrors_e CalculateTemperatureChecksum](#) ([TemperatureMessage_t *msg](#))

Function: CalculateTemperatureChecksum *

- Description: Creates the checksum for the temperature data message. *
- Parameters: [CommandMessage_t * msg](#): Pointer to a temperature data * message struct. *
- Return Value: [MessagingErrors_e](#): Enumeration of messaging errors. *
-

- [MessagingErrors_e DecodeCommandMessage](#) ([CommandMessage_t *msg](#))

Function: DecodeCommandMessage *

- Description: Parses the command message on the FRDM side and calls * the appropriate function from the commands function * pointer array. *
- Parameters: [CommandMessage_t * msg](#): Received command message. * [uint8_t data](#): Input value of the command. *
- Return Value: [MessagingErrors_e](#): Enumeration of messaging errors. *
-

Variables

- [CircularBuffer_t * UART1_TXBuffer](#)

4.13.1 Function Documentation

4.13.1.1 [MessagingErrors_e BuildCommandMessage](#) ([Commands_e cmd](#), [uint8_t data](#))

Function: BuildCommandMesage *

- Description: Takes a command number and the input value and creates * the message to be sent to the FRDM board. *
- Parameters: [Commands_e cmd](#): Command to be sent in the message. * [uint8_t data](#): Input value of the command. *
- Return Value: [MessagingErrors_e](#): Enumeration of messaging errors. *
-

4.13.1.2 **MessagingErrors_e** CalculateCommandChecksum (**CommandMessage_t** * *msg*)

Function: CalculateCommandChecksum *

- Description: Creates the checksum for the command message. *
- Parameters: **CommandMessage_t** * *msg*: Pointer to a command message struct*
- Return Value: **MessagingErrors_e**: Enumeration of messaging errors. *
-

4.13.1.3 **MessagingErrors_e** CalculateTemperatureChecksum (**TemperatureMessage_t** * *msg*)

Function: CalculateTemperatureChecksum *

- Description: Creates the checksum for the temperature data message. *
- Parameters: **CommandMessage_t** * *msg*: Pointer to a temperature data * message struct. *
- Return Value: **MessagingErrors_e**: Enumeration of messaging errors. *
-

4.13.1.4 **MessagingErrors_e** DecodeCommandMessage (**CommandMessage_t** * *msg*)

Function: DecodeCommandMessage *

- Description: Parses the command message on the FRDM side and calls * the appropriate function from the commands function * pointer array. *
- Parameters: **CommandMessage_t** * *msg*: Received command message. * **uint8_t** *data*: Input value of the command. *
- Return Value: **MessagingErrors_e**: Enumeration of messaging errors. *
-

4.13.2 Variable Documentation

4.13.2.1 **CircularBuffer_t*** UART1_TXBuffer

4.14 Messaging/messaging.h File Reference

```
#include "includeall.h"
```

Include dependency graph for messaging.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [CommandMessage](#)
- struct [TemperatureMessage](#)
- union [TemperatureData](#)

Macros

- #define [MAX_LENGTH](#) 128
- #define [ENABLE_MESSAGING](#) 1
- #define [NUM_COMMANDS](#) 7
- #define [COMMAND_MSG_BYTES](#) 3
- #define [TEMP_MSG_BYTES](#) 7

Typedefs

- typedef struct [CommandMessage](#) [CommandMessage_t](#)
- typedef struct [TemperatureMessage](#) [TemperatureMessage_t](#)

Enumerations

- enum [MessagingErrors_e](#) { [noError](#) = 0, [txBufferFull](#), [rxBufferFull](#) }
- enum [Commands_e](#) { [changeColor](#) = 0, [changePWM](#), [setTemp](#), [setDisplay](#), [setDesired](#), [setRange](#), [readTempData](#), [NOPcommand](#) = 0xFF }

Functions

- struct [CommandMessage](#) [__attribute__\(\(packed\)\)](#)
- [MessagingErrors_e](#) [BuildCommandMessage](#) ([Commands_e](#) cmd, uint8_t data)
 - Function: *BuildCommandMessage* *
 - Description: Takes a command number and the input value and creates * the message to be sent to the FRDM board. *
 - Parameters: *Commands_e cmd*: Command to be sent in the message. * *uint8_t data*: Input value of the command. *
 - Return Value: *MessagingErrors_e*: Enumeration of messaging errors. *
- [MessagingErrors_e](#) [CalculateCommandChecksum](#) ([CommandMessage_t](#) *msg)
 - Function: *CalculateCommandChecksum* *
 - Description: Creates the checksum for the command message. *
 - Parameters: *CommandMessage_t * msg*: Pointer to a command message struct*
 - Return Value: *MessagingErrors_e*: Enumeration of messaging errors. *
- [MessagingErrors_e](#) [CalculateTemperatureChecksum](#) ([TemperatureMessage_t](#) *msg)
 - Function: *CalculateTemperatureChecksum* *
 - Description: Creates the checksum for the temperature data message. *
 - Parameters: *CommandMessage_t * msg*: Pointer to a temperature data * message struct. *
 - Return Value: *MessagingErrors_e*: Enumeration of messaging errors. *
- [MessagingErrors_e](#) [DecodeCommandMessage](#) ([CommandMessage_t](#) *msg)
 - Function: *DecodeCommandMessage* *
 - Description: Parses the command message on the FRDM side and calls * the appropriate function from the commands function * pointer array. *
 - Parameters: *CommandMessage_t * msg*: Received command message. * *uint8_t data*: Input value of the command. *
 - Return Value: *MessagingErrors_e*: Enumeration of messaging errors. *

Variables

- typedef [__attribute__](#)
- uint8_t [cmd](#)
- uint8_t [data](#)
- uint8_t [checksum](#)
- uint8_t [currentTemp](#)
- uint8_t [currentDesired](#)
- uint8_t [currentRange](#)
- uint8_t [powerOn](#)
- uint8_t [cr](#)
- uint8_t [lf](#)

4.14.1 Macro Definition Documentation

4.14.1.1 `#define COMMAND_MSG_BYTES 3`

4.14.1.2 `#define ENABLE_MESSAGING 1`

4.14.1.3 `#define MAX_LENGTH 128`

4.14.1.4 `#define NUM_COMMANDS 7`

4.14.1.5 `#define TEMP_MSG_BYTES 7`

4.14.2 Typedef Documentation

4.14.2.1 `typedef struct CommandMessage CommandMessage_t`

4.14.2.2 `typedef struct TemperatureMessage TemperatureMessage_t`

4.14.3 Enumeration Type Documentation

4.14.3.1 `enum Commands_e`

Enumerator

changeColor

changePWM

setTemp

setDisplay

setDesired

setRange

readTempData

NOPcommand

4.14.3.2 enum MessagingErrors_e

Enumerator

noError
txBufferFull
rxBufferFull

4.14.4 Function Documentation

4.14.4.1 struct CommandMessage __attribute__((packed))

4.14.4.2 MessagingErrors_e BuildCommandMessage (Commands_e cmd, uint8_t data)

Function: BuildCommandMesage *

- Description: Takes a command number and the input value and creates * the message to be sent to the FRDM board. *
- Parameters: Commands_e cmd: Command to be sent in the message. * uint8_t data: Input value of the command. *
- Return Value: MessagingErrors_e: Enumeration of messaging errors. *
-

4.14.4.3 MessagingErrors_e CalculateCommandChecksum (CommandMessage_t * msg)

Function: CalculateCommandChecksum *

- Description: Creates the checksum for the command message. *
- Parameters: CommandMessage_t * msg: Pointer to a command message struct*
- Return Value: MessagingErrors_e: Enumeration of messaging errors. *
-

4.14.4.4 MessagingErrors_e CalculateTemperatureChecksum (TemperatureMessage_t * msg)

Function: CalculateTemperatureChecksum *

- Description: Creates the checksum for the temperature data message. *
- Parameters: CommandMessage_t * msg: Pointer to a temperature data * message struct. *
- Return Value: MessagingErrors_e: Enumeration of messaging errors. *
-

4.14.4.5 MessagingErrors_e DecodeCommandMessage (CommandMessage_t * msg)

Function: DecodeCommandMessage *

- Description: Parses the command message on the FRDM side and calls * the appropriate function from the commands function * pointer array. *
- Parameters: CommandMessage_t * msg: Received command message. * uint8_t data: Input value of the command. *
- Return Value: MessagingErrors_e: Enumeration of messaging errors. *
-

4.14.5 Variable Documentation

4.14.5.1 struct TemperatureMessage __attribute__

4.14.5.2 uint8_t checksum

4.14.5.3 uint8_t cmd

4.14.5.4 uint8_t cr

4.14.5.5 uint8_t currentDesired

4.14.5.6 uint8_t currentRange

4.14.5.7 uint8_t currentTemp

4.14.5.8 uint8_t data

4.14.5.9 uint8_t lf

4.14.5.10 uint8_t powerOn

4.15 Modules/adc.c File Reference

```
#include "adc.h"
```

Include dependency graph for adc.c:

Functions

- void [ADC_Init](#) ([ADC_InputChannel](#) ADC_ch)

Function: *ADC_Init* *

- Description: *Initializes the adc with the corresponding adc channel **
- Parameters: *ADC_InputChannel ADC_ch: The analog channel to being * initialized. **
- Return Value: *NONE **

- void [ADC_StartConversion](#) ([ADC_InputChannel](#) ADC_ch)

Function: *ADC_StartConversion* *

- Description: *Starts an analog to digital conversion and enables an * interrupt when the conversion is complete. **
- Parameters: *ADC_InputChannel ADC_ch: The analog channel starting * conversion **
- Return Value: *NONE **

- float [ADC_GetCurrentValue](#) ()

Function: *ADC_GetCurrentValue* *

- Description: *Returns a floating point percentage the current value * from the last conversion value divided by the max value **
- Parameters: *NONE **
- Return Value: *NONE **

- void [ADC0_IRQHandler](#) ()

Variables

- static uint8_t [ADC_value](#)

4.15.1 Function Documentation

4.15.1.1 void [ADC0_IRQHandler](#) ()

4.15.1.2 float [ADC_GetCurrentValue](#) ()

Function: *ADC_GetCurrentValue* *

- Description: *Returns a floating point percentage the current value * from the last conversion value divided by the max value **
- Parameters: *NONE **
- Return Value: *NONE **

4.15.1.3 void [ADC_Init](#) ([ADC_InputChannel](#) ADC_ch)

Function: *ADC_Init* *

- Description: *Initializes the adc with the corresponding adc channel **
- Parameters: *ADC_InputChannel ADC_ch: The analog channel to being * initialized. **
- Return Value: *NONE **

4.15.1.4 void ADC_StartConversion (ADC_InputChannel ADC_ch)

Function: ADC_StartConversion *

- Description: Starts an analog to digital conversion and enables an * interrupt when the conversion is complete. *
- Parameters: ADC_InputChannel ADC_ch: The analog channel starting * conversion *
- Return Value: NONE *

4.15.2 Variable Documentation

4.15.2.1 uint8_t ADC_value [static]

4.16 Modules/adc.h File Reference

```
#include "includeall.h"
```

Include dependency graph for adc.h: This graph shows which files directly or indirectly include this file:

Macros

- #define [ADC_PORT](#) SIM_SCGC5_PORTC_MASK
- #define [ADC_CH_PIN](#) PORTC_PCR0
- #define [ADC_CHANNEL](#) AD14
- #define [MUX_PIN_ANALOG](#) 0
- #define [MAX_VALUE](#) 255.0

Enumerations

- enum [ADC_InputChannel](#) {
[DADP0](#) = 0, [DADP1](#), [DADP2](#), [DADP3](#),
[AD4](#), [AD5](#), [AD6](#), [AD7](#),
[AD8](#), [AD9](#), [AD10](#), [AD11](#),
[AD12](#), [AD13](#), [AD14](#), [AD15](#),
[AD16](#), [AD17](#), [AD18](#), [AD19](#),
[AD20](#), [AD21](#), [AD22](#), [AD23](#),
[RESERVED0](#), [RESERVED1](#), [TEMP_SENSOR](#), [BANDGAP](#),
[RESERVED2](#), [VREFSH](#), [VREFSL](#), [DISABLED](#) }
- enum [ADC_AvgSamples](#) { [avg4Samples](#) = 0, [avg8Samples](#), [avg16Samples](#), [avg32Samples](#) }

Functions

- void [ADC_Init](#) ([ADC_InputChannel](#) ADC_ch)

Function: *ADC_Init* *

- Description: *Initializes the adc with the corresponding adc channel **
- Parameters: *ADC_InputChannel ADC_ch: The analog channel to be initialized. **
- Return Value: *NONE **

- void [ADC_StartConversion](#) ([ADC_InputChannel](#) ADC_ch)

Function: *ADC_StartConversion* *

- Description: *Starts an analog to digital conversion and enables an interrupt when the conversion is complete. **
- Parameters: *ADC_InputChannel ADC_ch: The analog channel starting conversion **
- Return Value: *NONE **

- float [ADC_GetCurrentValue](#) ()

Function: *ADC_GetCurrentValue* *

- Description: *Returns a floating point percentage the current value from the last conversion value divided by the max value **
- Parameters: *NONE **
- Return Value: *NONE **

4.16.1 Macro Definition Documentation

4.16.1.1 `#define ADC_CH_PIN PORTC_PCR0`

4.16.1.2 `#define ADC_CHANNEL AD14`

4.16.1.3 `#define ADC_PORT SIM_SCGC5_PORTC_MASK`

4.16.1.4 `#define MAX_VALUE 255.0`

4.16.1.5 `#define MUX_PIN_ANALOG 0`

4.16.2 Enumeration Type Documentation

4.16.2.1 `enum ADC_AvgSamples`

Enumerator

avg4Samples

avg8Samples

avg16Samples

avg32Samples

4.16.2.2 enum ADC_InputChannel

Enumerator

DADP0
DADP1
DADP2
DADP3
AD4
AD5
AD6
AD7
AD8
AD9
AD10
AD11
AD12
AD13
AD14
AD15
AD16
AD17
AD18
AD19
AD20
AD21
AD22
AD23
RESERVED0
RESERVED1
TEMP_SENSOR
BANDGAP
RESERVED2
VREFSH
VREFSL
DISABLED

4.16.3 Function Documentation

4.16.3.1 float ADC_GetCurrentValue ()

Function: ADC_GetCurrentValue *

- Description: Returns a floating point percentage the current value * from the last conversion value divided by the max value *
- Parameters: NONE *
- Return Value: NONE *

4.16.3.2 void ADC_Init (ADC_InputChannel ADC_ch)

Function: ADC_Init *

- Description: Initializes the adc with the corresponding adc channel *
- Parameters: ADC_InputChannel ADC_ch: The analog channel to be initialized. *
- Return Value: NONE *

4.16.3.3 void ADC_StartConversion (ADC_InputChannel ADC_ch)

Function: ADC_StartConversion *

- Description: Starts an analog to digital conversion and enables an interrupt when the conversion is complete. *
- Parameters: ADC_InputChannel ADC_ch: The analog channel starting conversion *
- Return Value: NONE *

4.17 Modules/diags.c File Reference

```
#include "diags.h"
```

Include dependency graph for diags.c:

Functions

- void [ParseDiag](#) (uint8_t *buffer)

*Function: ParseDiag **

- Description: Receives a string signifying a diag command from the serial port. *
- Parameters: uint8_t * buffer: pointer to a string buffer. *
- Return Value: NONE *

4.17.1 Function Documentation

4.17.1.1 void ParseDiag (uint8_t * buffer)

Function: ParseDiag *

- Description: Receives a string signifying a diag command from the serial port. *
- Parameters: uint8_t * buffer: pointer to a string buffer. *
- Return Value: NONE *

4.18 Modules/diags.h File Reference

```
#include "includeall.h"
```

Include dependency graph for diags.h: This graph shows which files directly or indirectly include this file:

Functions

- void [ParseDiag](#) (uint8_t *buffer)

*Function: ParseDiag **

- *Description: Receives a string signifying a diag command from * the serial port. **
- *Parameters: uint8_t * buffer: pointer to a string buffer. **
- *Return Value: NONE **

4.18.1 Function Documentation

4.18.1.1 void ParseDiag (uint8_t * buffer)

Function: ParseDiag *

- *Description: Receives a string signifying a diag command from * the serial port. **
- *Parameters: uint8_t * buffer: pointer to a string buffer. **
- *Return Value: NONE **

4.19 Modules/dma.c File Reference

4.20 Modules/dma.h File Reference

```
#include "includeall.h"
```

Include dependency graph for dma.h:

Macros

- #define [MASK_32BIT](#) 0xFFFFFFFF
- #define [DMA_BCR_SIZE_MASK](#) 0x0FFFF
- #define [NO_DMA](#) 0xFF

Typedefs

- typedef enum [DMAErrors](#) [DMAErrors_e](#)
- typedef enum [TransferSize](#) [TransferSize_e](#)
- typedef enum [BufferSize](#) [BufferSize_e](#)

Enumerations

- enum [DMAErrors](#) { [DMANoError](#) = 0, [DMANot16bitTransferSize](#), [DMANot32bitTransferSize](#) }
- enum [TransferSize](#) { [_32bit](#) = 0, [_8bit](#), [_16bit](#), [Reserved](#) }
- enum [BufferSize](#) {
[BufferDisabled](#) = 0, [_16Bytes](#), [_32Bytes](#), [_64Bytes](#),
[_128Bytes](#), [_256Bytes](#), [_512Bytes](#), [_1kBytes](#),
[_2kBytes](#), [_4kBytes](#), [_8kBytes](#), [_16kBytes](#),
[_32kBytes](#), [_64kBytes](#), [_128kBytes](#), [_256kBytes](#) }

Functions

- void [InitDMA](#) (uint8_t ch)
- [DMAErrors_e](#) [StartTransfer32bitMoves](#) (uint8_t ch, uint8_t *src, uint8_t *dst, uint32_t numBytes)
- [DMAErrors_e](#) [StartTransfer16bitMoves](#) (uint8_t ch, uint8_t *src, uint8_t *dst, uint32_t numBytes)
- [DMAErrors_e](#) [StartTransfer8bitMoves](#) (uint8_t ch, uint8_t *src, uint8_t *dst, uint32_t numBytes)
- [DMAErrors_e](#) [MemSet32bit](#) (uint8_t ch, uint32_t data, uint8_t *dst, uint32_t numBytes)
- [DMAErrors_e](#) [MemSet8bit](#) (uint8_t ch, uint8_t data, uint8_t *dst, uint32_t numBytes)

4.20.1 Macro Definition Documentation

4.20.1.1 `#define DMA_BCR_SIZE_MASK 0x0FFFFF`

4.20.1.2 `#define MASK_32BIT 0xFFFFFFFF`

4.20.1.3 `#define NO_DMA 0xFF`

4.20.2 Typedef Documentation

4.20.2.1 `typedef enum BufferSize BufferSize_e`

4.20.2.2 `typedef enum DMAErrors DMAErrors_e`

4.20.2.3 `typedef enum TransferSize TransferSize_e`

4.20.3 Enumeration Type Documentation

4.20.3.1 `enum BufferSize`

Enumerator

BufferDisabled

_16Bytes

_32Bytes

_64Bytes

_128Bytes

_256Bytes

_512Bytes

_1kBytes
_2kBytes
_4kBytes
_8kBytes
_16kBytes
_32kBytes
_64kBytes
_128kBytes
_256kBytes

4.20.3.2 enum DMAErrors

Enumerator

DMANoError
DMANot16bitTransferSize
DMANot32bitTransferSize

4.20.3.3 enum TransferSize

Enumerator

_32bit
_8bit
_16bit
Reserved

4.20.4 Function Documentation

4.20.4.1 void InitDMA (uint8_t *ch*)

4.20.4.2 DMAErrors_e MemSet32bit (uint8_t *ch*, uint32_t *data*, uint8_t * *dst*, uint32_t *numBytes*)

4.20.4.3 DMAErrors_e MemSet8bit (uint8_t *ch*, uint8_t *data*, uint8_t * *dst*, uint32_t *numBytes*)

4.20.4.4 DMAErrors_e StartTransfer16bitMoves (uint8_t *ch*, uint8_t * *src*, uint8_t * *dst*, uint32_t *numBytes*)

4.20.4.5 DMAErrors_e StartTransfer32bitMoves (uint8_t *ch*, uint8_t * *src*, uint8_t * *dst*, uint32_t *numBytes*)

4.20.4.6 DMAErrors_e StartTransfer8bitMoves (uint8_t *ch*, uint8_t * *src*, uint8_t * *dst*, uint32_t *numBytes*)

4.21 Modules/ds18b20.c File Reference

```
#include "ds18b20.h"
Include dependency graph for ds18b20.c:
```

Functions

- `uint8_t TransactionStepOne ()`

Function: *TransactionStepOne **

- Description: *Simply calls the first step of the ds18b20 tranaction * which is the Single Wire Comms Reset and precence pusle **
- Parameters: *NONE **
- Return Value: *uint8: Boolean to indicate if a presence pulse was * received. **
-

- `void TransactionStepTwo ()`

Function: *TransactionStepTwo **

- Description: *The second part of every ds18b20 transaction is a ROM * command and then reading the data coming from the device. **
- Parameters: *NONE **
- Return Value: *NONE **
-

- `float ReadTemp ()`

Function: *ReadTemp **

- Description: *Completes two different transactions to the ds18b20. * The first transaction is starting a temperature conversion* which then places the result in the scratchpad. The second* transaction is to read the scratchpad data to retrieve the* result of the temperate reading. **
- Parameters: *NONE **
- Return Value: *float: The floating point value of the temperature result**
-

- `float ConvertRawTemperatureData (uint16_t rawTemperatureData)`

Function: *ConvertRawTemperatureData **

- Description: *The raw temperature data needs to be converted to a value * that can be understood by the processor. The lowest nibble* is the fractional part of the temperature. The next byte * is the whole part but with the most significant bit is sign* extended to the MSB of the half word. **
- Parameters: *uint16_t rawTemperatureData: The raw temperature data that * needs to be converted to a floating point value. **
- Return Value: *float: The floating point value of the temperature result**
-

4.21.1 Function Documentation

4.21.1.1 `float ConvertRawTemperatureData (uint16_t rawTemperatureData)`

Function: *ConvertRawTemperatureData **

- Description: *The raw temperature data needs to be converted to a value * that can be understood by the processor. The lowest nibble* is the fractional part of the temperature. The next byte * is the whole part but with the most significant bit is sign* extended to the MSB of the half word. **
- Parameters: *uint16_t rawTemperatureData: The raw temperature data that * needs to be converted to a floating point value. **
- Return Value: *float: The floating point value of the temperature result**
-

4.21.1.2 float ReadTemp ()

Function: ReadTemp *

- Description: Completes two different transactions to the ds18b20. * The first transaction is starting a temperature conversion* which then places the result in the scratchpad. The second* transaction is to read the scratchpad data to retrieve the* result of the temperate reading. *
- Parameters: NONE *
- Return Value: float: The floating point value of the temperature result*
-

4.21.1.3 uint8_t TransactionStepOne ()

Function: TransactionStepOne *

- Description: Simply calls the first step of the ds18b20 tranaction * which is the Single Wire Comms Reset and precence pusle *
- Parameters: NONE *
- Return Value: uint8: Boolean to indicate if a presence pulse was * received. *
-

4.21.1.4 void TransactionStepTwo ()

Function: TransactionStepTwo *

- Description: The second part of every ds18b20 transaction is a ROM * command and then reading the data coming from the device. *
- Parameters: NONE *
- Return Value: NONE *
-

4.22 Modules/ds18b20.h File Reference

```
#include "includeall.h"
```

Include dependency graph for ds18b20.h: This graph shows which files directly or indirectly include this file:

Data Structures

- union [DS8B20_ROMCode](#)
- union [DS8B20_Scratchpad](#)

Macros

- #define [ROM_BYTES](#) 8
- #define [SERIAL_NUM_BYTES](#) 6
- #define [SCRATCHPAD_BYTES](#) 9
- #define [WAIT_TIME_LONG](#) 200
- #define [WAIT_TIME_SHORT](#) 20
- #define [SEARCH_ROM](#) 0xF0
- #define [READ_ROM](#) 0x33
- #define [MATCH_ROM](#) 0x55
- #define [SKIP_ROM](#) 0xCC
- #define [ALARM_SEARCH](#) 0xEC
- #define [CONVERT_T](#) 0x44
- #define [WRITE_SCRATCHPAD](#) 0x4E
- #define [READ_SCRATCHPAD](#) 0xBE
- #define [COPY_SCRATCHPAD](#) 0x48
- #define [RECALL_E2](#) 0xB8
- #define [READ_POWER_SUPPLY](#) 0xB4

Functions

- [uint8_t TransactionStepOne \(\)](#)
*Function: TransactionStepOne **
 - Description: Simply calls the first step of the ds18b20 tranaction * which is the Single Wire Comms Reset and precence pusle *
 - Parameters: NONE *
 - Return Value: uint8: Boolean to indicate if a presence pulse was * received. *
 -
- [void TransactionStepTwo \(\)](#)
*Function: TransactionStepTwo **
 - Description: The second part of every ds18b20 transaction is a ROM * command and then reading the data coming from the device. *
 - Parameters: NONE *
 - Return Value: NONE *
 -
- [float ReadTemp \(\)](#)
*Function: ReadTemp **
 - Description: Completes two different transactions to the ds18b20. * The first transaction is starting a temperature conversion* which then places the result in the scratchpad. The second* transaction is to read the scratchpad data to retrieve the* result of the temperate reading. *
 - Parameters: NONE *
 - Return Value: float: The floating point value of the temperature result*
 -
- [float ConvertRawTemperatureData \(uint16_t rawTemperatureData\)](#)
*Function: ConvertRawTemperatureData **
 - Description: The raw temperature data needs to be converted to a value * that can be understood by the processor. The lowest nibble* is the fractional part of the temperature. The next byte * is the whole part but with the most significant bit is sign* extended to the MSB of the half word. *
 - Parameters: uint16_t rawTemperatureData: The raw temperature data that * needs to be converted to a floating point value. *
 - Return Value: float: The floating point value of the temperature result*
 -

4.22.1 Macro Definition Documentation

4.22.1.1 `#define ALARM_SEARCH 0xEC`

4.22.1.2 `#define CONVERT_T 0x44`

4.22.1.3 `#define COPY_SCRATCHPAD 0x48`

4.22.1.4 `#define MATCH_ROM 0x55`

4.22.1.5 `#define READ_POWER_SUPPLY 0xB4`

4.22.1.6 `#define READ_ROM 0x33`

4.22.1.7 `#define READ_SCRATCHPAD 0xBE`

4.22.1.8 `#define RECALL_E2 0xB8`

4.22.1.9 `#define ROM_BYTES 8`

4.22.1.10 `#define SCRATCHPAD_BYTES 9`

4.22.1.11 `#define SEARCH_ROM 0xF0`

4.22.1.12 `#define SERIAL_NUM_BYTES 6`

4.22.1.13 `#define SKIP_ROM 0xCC`

4.22.1.14 `#define WAIT_TIME_LONG 200`

4.22.1.15 `#define WAIT_TIME_SHORT 20`

4.22.1.16 `#define WRITE_SCRATCHPAD 0x4E`

4.22.2 Function Documentation

4.22.2.1 `float ConvertRawTemperatureData (uint16_t rawTemperatureData)`

Function: `ConvertRawTemperatureData` *

- Description: The raw temperature data needs to be converted to a value * that can be understood by the processor. The lowest nibble* is the fractional part of the temperature. The next byte * is the whole part but with the most significant bit is sign* extended to the MSB of the half word. *
- Parameters: `uint16_t rawTemperatureData`: The raw temperature data that * needs to be converted to a floating point value. *
- Return Value: `float`: The floating point value of the temperature result*
-

4.22.2.2 float ReadTemp ()

Function: ReadTemp *

- Description: Completes two different transactions to the ds18b20. * The first transaction is starting a temperature conversion* which then places the result in the scratchpad. The second* transaction is to read the scratchpad data to retrieve the* result of the temperate reading. *
- Parameters: NONE *
- Return Value: float: The floating point value of the temperature result*
-

4.22.2.3 uint8_t TransactionStepOne ()

Function: TransactionStepOne *

- Description: Simply calls the first step of the ds18b20 tranaction * which is the Single Wire Comms Reset and precence pusle *
- Parameters: NONE *
- Return Value: uint8: Boolean to indicate if a presence pulse was * received. *
-

4.22.2.4 void TransactionStepTwo ()

Function: TransactionStepTwo *

- Description: The second part of every ds18b20 transaction is a ROM * command and then reading the data coming from the device. *
- Parameters: NONE *
- Return Value: NONE *
-

4.23 Modules/io.h File Reference

4.24 Modules/led.c File Reference

Functions

- void [CycleLEDs](#) ()

4.24.1 Function Documentation

4.24.1.1 void CycleLEDs ()

4.25 Modules/led.h File Reference

```
#include "includeall.h"
```

Include dependency graph for led.h: This graph shows which files directly or indirectly include this file:

Macros

- #define RED_PIN 1 << 18
- #define GREEN_PIN 1 << 19
- #define BLUE_PIN 1 << 1
- #define RED_TPM 2
- #define RED_CHANNEL 0
- #define GREEN_TPM 2
- #define GREEN_CHANNEL 1
- #define BLUE_TPM 0
- #define BLUE_CHANNEL 1

Typedefs

- typedef enum Color_t Color_t

Enumerations

- enum Color_t {
RED = 0, GREEN, BLUE, PURPLE,
YELLOW, CYAN, WHITE, OFF,
NONE }

Functions

- void LEDSetup (void)
- void SwitchLEDs (uint8_t color)
- void CycleLEDs ()

4.25.1 Macro Definition Documentation

4.25.1.1 `#define BLUE_CHANNEL 1`

4.25.1.2 `#define BLUE_PIN 1 << 1`

4.25.1.3 `#define BLUE_TPM 0`

4.25.1.4 `#define GREEN_CHANNEL 1`

4.25.1.5 `#define GREEN_PIN 1 << 19`

4.25.1.6 `#define GREEN_TPM 2`

4.25.1.7 `#define RED_CHANNEL 0`

4.25.1.8 `#define RED_PIN 1 << 18`

4.25.1.9 `#define RED_TPM 2`

4.25.2 Typedef Documentation

4.25.2.1 `typedef enum Color_t Color_t`

4.25.3 Enumeration Type Documentation

4.25.3.1 `enum Color_t`

Enumerator

RED

GREEN

BLUE

PURPLE

YELLOW

CYAN

WHITE

OFF

NONE

4.25.4 Function Documentation

4.25.4.1 void CycleLEDs ()

4.25.4.2 void LEDSetup (void)

4.25.4.3 void SwitchLEDs (uint8_t color)

4.26 Modules/nRF24L01.c File Reference

```
#include "nRF24L01.h"
```

Include dependency graph for nRF24L01.c:

Functions

- void [nRF24L01_Activate](#) (uint8_t SPI_ch)

Function: *nRF24L01_Activate* *

- Description: Sets up a msg that will be sent to the nRF24L01 module * that will activate the device. The command sets the * R_RX_PL_WID, W_ACK_PAYLOAD, and W_TX_PAYLOAD_NOACK * features. Also this will set up the CE and IRQ pins. *
- Parameters: uint8_t SPI_ch: The spi channel being used. *
- Return Value: NONE *

- void [nRF24L01_ReadReg](#) (uint8_t SPI_ch, [nRF24L01_Registers_e](#) reg)

Function: *nRF24L01_ReadReg* *

- Description: Sets up a msg that will be sent to the nRF24L01 module * that will read a register on the device. *
- Parameters: uint8_t SPI_ch: The spi channel being used. * [nRF24L01_Registers_e](#) reg: Register to be read. *
- Return Value: NONE *

- void [nRF24L01_WriteReg](#) (uint8_t SPI_ch, [nRF24L01_Registers_e](#) reg, uint8_t dataToWrite)

Function: *nRF24L01_WriteReg* *

- Description: Sets up a msg that will be sent to the nRF24L01 module * that will write a value to a register on the device. *
- Parameters: uint8_t SPI_ch: The spi channel being used. * [nRF24L01_Registers_e](#) reg: Register to be written to. * uint8_t dataToWrite: value to be written to register *
- Return Value: NONE *

- void [nRF24L01_SendData](#) ([nRF24L01_SPIMessage_t](#) *msg)

Function: *nRF24L01_SendData* *

- Description: Receives a msg structure and then adds that msg to the * circular buffer used for SPI transmissions. *
- Parameters: [nRF24L01_SPIMessage_t](#) * msg: pointer to the message struct *
- Return Value: NONE *

- void [nRF24L01_SetTXMode](#) (uint8_t SPI_ch)

Function: *nRF24L01_SetTXMode* *

- Description: Sets up the nRF24L01 in TX mode *
- Parameters: uint8_t SPI_ch: SPI channel being used. *
- Return Value: NONE *

- void [nRF24L01_SetRXMode](#) (uint8_t SPI_ch)

Function: *nRF24L01_SetRXMode* *

- Description: Sets up the nRF24L01 in RX mode *
- Parameters: uint8_t SPI_ch: SPI channel being used. *
- Return Value: NONE *

- void [nRF24L01_StandbyMode](#) (uint8_t SPI_ch)

- Function: `nRF24L01_StandbyMode` *
- Description: Sets up the nRF24L01 in RX mode *
 - Parameters: `uint8_t SPI_ch`: SPI channel being used. *
 - Return Value: NONE *
- void `nRF24L01_PowerDown` (`uint8_t SPI_ch`)
- Function: `nRF24L01_PowerDown` *
- Description: Sets up the nRF24L01 in RX mode *
 - Parameters: `uint8_t SPI_ch`: SPI channel being used. *
 - Return Value: NONE *
- void `nRF24L01_SetupChannel` (`uint8_t SPI_ch`)
- Function: `nRF24L01_SetupChannel` *
- Description: Sets up the nRF24L01 transmission channel. *
 - Parameters: `uint8_t SPI_ch`: SPI channel being used. *
 - Return Value: NONE *
- void `nRF24L01_SendNOP` (`uint8_t SPI_ch`)
- Function: `nRF24L01_SendNOP` *
- Description: Sends a NOP command to the nRF24L01 device *
 - Parameters: `uint8_t SPI_ch`: SPI channel being used. *
 - Return Value: NONE *

Variables

- `CircularBuffer_t` * `SPI_RXBuffer` [`SPI_CHANNELS`]
- `CircularBuffer_t` * `SPI_TXBuffer` [`SPI_CHANNELS`]
- `uint8_t` `readRegComplete`

4.26.1 Function Documentation

4.26.1.1 void nRF24L01_Activate (uint8_t SPI_ch)

Function: `nRF24L01_Activate` *

- Description: Sets up a msg that will be sent to the nRF24L01 module * that will activate the device. The command sets the * `R_RX_PL_WID`, `W_ACK_PAYLOAD`, and `W_TX_PAYLOAD_NOACK` * features. Also this will set up the CE and IRQ pins. *
- Parameters: `uint8_t SPI_ch`: The spi channel being used. *
- Return Value: NONE *

4.26.1.2 void nRF24L01_PowerDown (uint8_t SPI_ch)

Function: `nRF24L01_PowerDown` *

- Description: Sets up the nRF24L01 in RX mode *
- Parameters: `uint8_t SPI_ch`: SPI channel being used. *
- Return Value: NONE *

4.26.1.3 void nRF24L01_ReadReg (uint8_t *SPI_ch*, nRF24L01_Registers_e *reg*)

Function: nRF24L01_ReadReg *

- Description: Sets up a msg that will be sent to the nRF24L01 module * that will read a register on the device. *
- Parameters: uint8_t *SPI_ch*: The spi channel being used. * nRF24L01_Registers_e *reg*: Register to be read. *
- Return Value: NONE *

4.26.1.4 void nRF24L01_SendData (nRF24L01_SPIMessage_t * *msg*)

Function: nRF24L01_SendData *

- Description: Receives a msg structure and then adds that msg to the * circular buffer used for SPI transmissions. *
- Parameters: [nRF24L01_SPIMessage_t](#) * *msg*: pointer to the message struct *
- Return Value: NONE *

4.26.1.5 void nRF24L01_SendNOP (uint8_t *SPI_ch*)

Function: nRF24L01_SendNOP *

- Description: Sends a NOP command to the nRF24L01 device *
- Parameters: uint8_t *SPI_ch*: SPI channel being used. *
- Return Value: NONE *

4.26.1.6 void nRF24L01_SetRXMode (uint8_t *SPI_ch*)

Function: nRF24L01_SetRXMode *

- Description: Sets up the nRF24L01 in RX mode *
- Parameters: uint8_t *SPI_ch*: SPI channel being used. *
- Return Value: NONE *

4.26.1.7 void nRF24L01_SetTXMode (uint8_t *SPI_ch*)

Function: nRF24L01_SetTXMode *

- Description: Sets up the nRF24L01 in TX mode *
- Parameters: uint8_t *SPI_ch*: SPI channel being used. *
- Return Value: NONE *

4.26.1.8 void nRF24L01_SetupChannel (uint8_t SPI_ch)

Function: nRF24L01_SetupChannel *

- Description: Sets up the nRF24L01 transmission channel. *
- Parameters: uint8_t SPI_ch: SPI channel being used. *
- Return Value: NONE *

4.26.1.9 void nRF24L01_StandbyMode (uint8_t SPI_ch)

Function: nRF24L01_StandbyMode *

- Description: Sets up the nRF24L01 in RX mode *
- Parameters: uint8_t SPI_ch: SPI channel being used. *
- Return Value: NONE *

4.26.1.10 void nRF24L01_WriteReg (uint8_t SPI_ch, nRF24L01_Registers_e reg, uint8_t dataToWrite)

Function: nRF24L01_WriteReg *

- Description: Sets up a msg that will be sent to the nRF24L01 module * that will write a value to a register on the device. *
- Parameters: uint8_t SPI_ch: The spi channel being used. * nRF24L01_Registers_e reg: Register to be written to. * uint8_t dataToWrite: value to be written to register *
- Return Value: NONE *

4.26.2 Variable Documentation

4.26.2.1 uint8_t readRegComplete

4.26.2.2 CircularBuffer_t* SPI_RXBuffer[SPI_CHANNELS]

4.26.2.3 CircularBuffer_t* SPI_TXBuffer[SPI_CHANNELS]

4.27 Modules/nRF24L01.h File Reference

```
#include "includeall.h"
```

Include dependency graph for nRF24L01.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [nRF24L01_SPIMessage_t](#)
- union [nRF24L01_CONFIG_t](#)
- union [nRF24L01_ENAA_t](#)
- union [nRF24L01_EN_RXADDR_t](#)
- union [nRF24L01_SETUP_AW_t](#)
- union [nRF24L01_SETUP_RETR_t](#)
- union [nRF24L01_RF_CH_t](#)
- union [nRF24L01_RF_SETUP_t](#)
- union [nRF24L01_STATUS_t](#)
- union [nRF24L01_OBSERVE_TX_t](#)
- union [nRF24L01_RX_PW_P0_t](#)
- union [nRF24L01_RX_PW_P1_t](#)
- union [nRF24L01_RX_PW_P2_t](#)
- union [nRF24L01_RX_PW_P3_t](#)
- union [nRF24L01_RX_PW_P4_t](#)
- union [nRF24L01_RX_PW_P5_t](#)
- union [nRF24L01_FIFO_STATUS_t](#)
- union [nRF24L01_DYNPD_t](#)
- union [nRF24L01_FEATURE_t](#)

Macros

- #define [READ_REG](#)(reg) (uint8_t) ((reg) & 0x1F)
- #define [WRITE_REG](#)(reg) (uint8_t) (0x20 | (reg) & 0x1F)
- #define [R_RX_PAYLOAD](#) 0x61
- #define [W_TX_PAYLOAD](#) 0xA0
- #define [FLUSH_TX](#) 0xE1
- #define [FLUSH_RX](#) 0xE2
- #define [REUSE_TX_PL](#) 0xE3
- #define [ACTIVATE](#) 0x50
- #define [ACTIVATE_KEY](#) 0x73
- #define [R_RX_PL_WID](#) 0x60
- #define [W_ACK_PAYLOAD](#)(ppp) (0xA8 | ((ppp) & 0x7))
- #define [W_TX_PAYLOAD_NO_ACK](#) 0xB0
- #define [NOP](#) 0xFF
- #define [nRF24L01_0_CE](#) PORTC_PCR0
- #define [nRF24L01_0_CE_PIN](#) 0x0001
- #define [nRF24L01_0_IRQ](#) PORTC_PCR3
- #define [nRF24L01_0_IRQ_PIN](#) 0x0008
- #define [nRF24L01_1_CE](#) PORTE_PCR0
- #define [nRF24L01_1_CE_PIN](#) 0x0001
- #define [nRF24L01_1_IRQ](#) PORTE_PCR5
- #define [nRF24L01_1_IRQ_PIN](#) 0x0020
- #define [nRF24L01_DATA_RATE](#) ([ADR_1Mbps](#) << 3)
- #define [nRF24L01_PA_CONTROL](#) ([PWR_neg18dBm](#) << 1)
- #define [nRF24L01_CHANNEL_FREQ](#) CHANNEL_0
- #define [nRF24L01_LNA_GAIN](#) 0
- #define [PRIM_RX_MASK](#) 0x01
- #define [PWR_UP_MASK](#) 0x02
- #define [CRCO_MASK](#) 0x04
- #define [EN_CRC_MASK](#) 0x08
- #define [MASK_MAX_RT_MASK](#) 0x10
- #define [MASK_TX_DS_MASK](#) 0x20
- #define [MASK_RX_DR_MASK](#) 0x40

Enumerations

- enum [ADR_e](#) { [ADR_1Mbps](#) = 0, [ADR_2Mbps](#) }
- enum [PWR_e](#) { [PWR_neg18dBm](#) = 0, [PWR_neg12dBm](#), [PWR_neg6dBm](#), [PWR_0dBm](#) }
- enum [CHANNEL_e](#) {
[CHANNEL_0](#) = 0, [CHANNEL_1](#) = 10, [CHANNEL_2](#) = 20, [CHANNEL_3](#) = 30,
[CHANNEL_4](#) = 40, [CHANNEL_5](#) = 50, [CHANNEL_6](#) = 60, [CHANNEL_7](#) = 70,
[CHANNEL_8](#) = 80, [CHANNEL_9](#) = 90 }
- enum [nRF24L01_Registers_e](#) {
[CONFIG](#) = 0x0, [EN_AA](#), [EN_RXADDR](#), [SETUP_AW](#),
[SETUP_RETR](#), [RF_CH](#), [RF_SETUP](#), [STATUS](#),
[OBSERVE_TX](#), [CD](#), [RX_ADDR_P0](#), [RX_ADDR_P1](#),
[RX_ADDR_P2](#), [RX_ADDR_P3](#), [RX_ADDR_P4](#), [RX_ADDR_P5](#),
[TX_ADDR](#), [RX_PW_P0](#), [RX_PW_P1](#), [RX_PW_P2](#),
[RX_PW_P3](#), [RX_PW_P4](#), [RX_PW_P5](#), [FIFO_STATUS](#),
[DYNPD](#) = 0x1C, [FEATURE](#) }

Functions

- void [nRF24L01_Activate](#) (uint8_t SPI_ch)
Function: [nRF24L01_Activate](#) *
 - Description: Sets up a msg that will be sent to the nRF24L01 module * that will activate the device. The command sets the * [R_RX_PL_WID](#), [W_ACK_PAYLOAD](#), and [W_TX_PAYLOAD_NOACK](#) * features. Also this will set up the CE and IRQ pins. *
 - Parameters: uint8_t SPI_ch: The spi channel being used. *
 - Return Value: NONE *
- void [nRF24L01_ReadReg](#) (uint8_t SPI_ch, [nRF24L01_Registers_e](#) reg)
Function: [nRF24L01_ReadReg](#) *
 - Description: Sets up a msg that will be sent to the nRF24L01 module * that will read a register on the device. *
 - Parameters: uint8_t SPI_ch: The spi channel being used. * [nRF24L01_Registers_e](#) reg: Register to be read. *
 - Return Value: NONE *
- void [nRF24L01_WriteReg](#) (uint8_t SPI_ch, [nRF24L01_Registers_e](#) reg, uint8_t dataToWrite)
Function: [nRF24L01_WriteReg](#) *
 - Description: Sets up a msg that will be sent to the nRF24L01 module * that will write a value to a register on the device. *
 - Parameters: uint8_t SPI_ch: The spi channel being used. * [nRF24L01_Registers_e](#) reg: Register to be written to. * uint8_t dataToWrite: value to be written to register *
 - Return Value: NONE *
- void [nRF24L01_SendData](#) ([nRF24L01_SPIMessage_t](#) *msg)
Function: [nRF24L01_SendData](#) *
 - Description: Receives a msg structure and then adds that msg to the * circular buffer used for SPI transmissions. *
 - Parameters: [nRF24L01_SPIMessage_t](#) * msg: pointer to the message struct *
 - Return Value: NONE *
- void [nRF24L01_SetTXMode](#) (uint8_t SPI_ch)
Function: [nRF24L01_SetTXMode](#) *
 - Description: Sets up the nRF24L01 in TX mode *
 - Parameters: uint8_t SPI_ch: SPI channel being used. *
 - Return Value: NONE *
- void [nRF24L01_SetRXMode](#) (uint8_t SPI_ch)
Function: [nRF24L01_SetRXMode](#) *
 - Description: Sets up the nRF24L01 in RX mode *
 - Parameters: uint8_t SPI_ch: SPI channel being used. *
 - Return Value: NONE *

- void [nRF24L01_StandbyMode](#) (uint8_t SPI_ch)
 - Function: *nRF24L01_StandbyMode **
 - Description: Sets up the nRF24L01 in RX mode *
 - Parameters: *uint8_t SPI_ch: SPI channel being used. **
 - Return Value: *NONE **
- void [nRF24L01_PowerDown](#) (uint8_t SPI_ch)
 - Function: *nRF24L01_PowerDown **
 - Description: Sets up the nRF24L01 in RX mode *
 - Parameters: *uint8_t SPI_ch: SPI channel being used. **
 - Return Value: *NONE **
- void [nRF24L01_SetupChannel](#) (uint8_t SPI_ch)
 - Function: *nRF24L01_SetupChannel **
 - Description: Sets up the nRF24L01 transmission channel. *
 - Parameters: *uint8_t SPI_ch: SPI channel being used. **
 - Return Value: *NONE **
- void [nRF24L01_SendNOP](#) (uint8_t SPI_ch)
 - Function: *nRF24L01_SendNOP **
 - Description: Sends a NOP command to the nRF24L01 device *
 - Parameters: *uint8_t SPI_ch: SPI channel being used. **
 - Return Value: *NONE **

4.27.1 Macro Definition Documentation

4.27.1.1 `#define ACTIVATE 0x50`

4.27.1.2 `#define ACTIVATE_KEY 0x73`

4.27.1.3 `#define CRCO_MASK 0x04`

4.27.1.4 `#define EN_CRC_MASK 0x08`

4.27.1.5 `#define FLUSH_RX 0xE2`

4.27.1.6 `#define FLUSH_TX 0xE1`

4.27.1.7 `#define MASK_MAX_RT_MASK 0x10`

4.27.1.8 `#define MASK_RX_DR_MASK 0x40`

4.27.1.9 `#define MASK_TX_DS_MASK 0x20`

4.27.1.10 `#define NOP 0xFF`

4.27.1.11 `#define nRF24L01_0_CE PORTC_PCR0`

4.27.1.12 `#define nRF24L01_0_CE_PIN 0x0001`

4.27.1.13 `#define nRF24L01_0_IRQ PORTC_PCR3`

4.27.1.14 `#define nRF24L01_0_IRQ_PIN 0x0008`

4.27.1.15 `#define nRF24L01_1_CE PORTE_PCR0`

4.27.1.16 `#define nRF24L01_1_CE_PIN 0x0001`

4.27.1.17 `#define nRF24L01_1_IRQ PORTE_PCR5`

4.27.1.18 `#define nRF24L01_1_IRQ_PIN 0x0020`

4.27.1.19 `#define nRF24L01_CHANNEL_FREQ CHANNEL_0`

4.27.1.20 `#define nRF24L01_DATA_RATE (ADR_1Mbps << 3)`

4.27.1.21 `#define nRF24L01_LNA_GAIN 0`

4.27.1.22 `#define nRF24L01_PA_CONTROL (PWR_neg18dBm << 1)`

4.27.1.23 `#define PRIM_RX_MASK 0x01`

4.27.1.24 `#define PWR_UP_MASK 0x02`

4.27.1.25 `#define R_RX_PAYLOAD 0x61`

4.27.1.26 `#define R_RX_PL_WID 0x60`

4.27.1.27 `#define READ_REG(reg) (uint8_t) ((reg) & 0x1F)`

4.27.1.28 `#define REUSE_TX_PL 0xE3`

4.27.1.29 `#define W_ACK_PAYLOAD(ppp) (0xA8 | ((ppp) & 0x7))`

4.27.1.30 `#define W_TX_PAYLOAD 0xA0`

4.27.1.31 `#define W_TX_PAYLOAD_NO_ACK 0xB0`

4.27.1.32 `#define WRITE_REG(reg) (uint8_t) (0x20 | (reg) & 0x1F)`

4.27.2 Enumeration Type Documentation

4.27.2.1 `enum ADR_e`

Enumerator

ADR_1Mbps

ADR_2Mbps

4.27.2.2 enum CHANNEL_e

Enumerator

CHANNEL_0
CHANNEL_1
CHANNEL_2
CHANNEL_3
CHANNEL_4
CHANNEL_5
CHANNEL_6
CHANNEL_7
CHANNEL_8
CHANNEL_9

4.27.2.3 enum nRF24L01_Registers_e

Enumerator

CONFIG
EN_AA
EN_RXADDR
SETUP_AW
SETUP_RETR
RF_CH
RF_SETUP
STATUS
OBSERVE_TX
CD
RX_ADDR_P0
RX_ADDR_P1
RX_ADDR_P2
RX_ADDR_P3
RX_ADDR_P4
RX_ADDR_P5
TX_ADDR
RX_PW_P0
RX_PW_P1
RX_PW_P2
RX_PW_P3
RX_PW_P4
RX_PW_P5
FIFO_STATUS
DYNPD
FEATURE

4.27.2.4 enum PWR_e

Enumerator

*PWR_neg18dBm**PWR_neg12dBm**PWR_neg6dBm**PWR_0dBm*

4.27.3 Function Documentation

4.27.3.1 void nRF24L01_Activate (uint8_t SPI_ch)

Function: nRF24L01_Activate *

- Description: Sets up a msg that will be sent to the nRF24L01 module * that will activate the device. The command sets the * R_RX_PL_WID, W_ACK_PAYLOAD, and W_TX_PAYLOAD_NOACK * features. Also this will set up the CE and IRQ pins. *
- Parameters: uint8_t SPI_ch: The spi channel being used. *
- Return Value: NONE *

4.27.3.2 void nRF24L01_PowerDown (uint8_t SPI_ch)

Function: nRF24L01_PowerDown *

- Description: Sets up the nRF24L01 in RX mode *
- Parameters: uint8_t SPI_ch: SPI channel being used. *
- Return Value: NONE *

4.27.3.3 void nRF24L01_ReadReg (uint8_t SPI_ch, nRF24L01_Registers_e reg)

Function: nRF24L01_ReadReg *

- Description: Sets up a msg that will be sent to the nRF24L01 module * that will read a register on the device. *
- Parameters: uint8_t SPI_ch: The spi channel being used. * nRF24L01_Registers_e reg: Register to be read. *
- Return Value: NONE *

4.27.3.4 void nRF24L01_SendData (nRF24L01_SPIMessage_t * msg)

Function: nRF24L01_SendData *

- Description: Receives a msg structure and then adds that msg to the * circular buffer used for SPI transmissions. *
- Parameters: [nRF24L01_SPIMessage_t](#) * msg: pointer to the message struct *
- Return Value: NONE *

4.27.3.5 void nRF24L01_SendNOP (uint8_t SPI_ch)

Function: nRF24L01_SendNOP *

- Description: Sends a NOP command to the nRF24L01 device *
- Parameters: uint8_t SPI_ch: SPI channel being used. *
- Return Value: NONE *

4.27.3.6 void nRF24L01_SetRXMode (uint8_t SPI_ch)

Function: nRF24L01_SetRXMode *

- Description: Sets up the nRF24L01 in RX mode *
- Parameters: uint8_t SPI_ch: SPI channel being used. *
- Return Value: NONE *

4.27.3.7 void nRF24L01_SetTXMode (uint8_t SPI_ch)

Function: nRF24L01_SetTXMode *

- Description: Sets up the nRF24L01 in TX mode *
- Parameters: uint8_t SPI_ch: SPI channel being used. *
- Return Value: NONE *

4.27.3.8 void nRF24L01_SetupChannel (uint8_t SPI_ch)

Function: nRF24L01_SetupChannel *

- Description: Sets up the nRF24L01 transmission channel. *
- Parameters: uint8_t SPI_ch: SPI channel being used. *
- Return Value: NONE *

4.27.3.9 void nRF24L01_StandbyMode (uint8_t SPI_ch)

Function: nRF24L01_StandbyMode *

- Description: Sets up the nRF24L01 in RX mode *
- Parameters: uint8_t SPI_ch: SPI channel being used. *
- Return Value: NONE *

4.27.3.10 void nRF24L01_WriteReg (uint8_t SPI_ch, nRF24L01_Registers_e reg, uint8_t dataToWrite)

Function: nRF24L01_WriteReg *

- Description: Sets up a msg that will be sent to the nRF24L01 module * that will write a value to a register on the device. *
- Parameters: uint8_t SPI_ch: The spi channel being used. * nRF24L01_Registers_e reg: Register to be written to. * uint8_t dataToWrite: value to be written to register *
- Return Value: NONE *

4.28 Modules/pushbutton.c File Reference

#include "pushbutton.h"

Include dependency graph for pushbutton.c:

Functions

- void [Button_Init](#) (uint8_t buttonNum)
*Function: Button_Init **
 - Description: Initializes the GPIO settings for a pushbutton. *
 - Parameters: uint8_t buttonNum: Not currently being used but is * a placeholder if more than one pushbutton will be used in * the future. *
 - Return Value: NONE *
 -
- void [PORTA_IRQHandler](#) ()

4.28.1 Function Documentation

4.28.1.1 void Button_Init (uint8_t buttonNum)

Function: Button_Init *

- Description: Initializes the GPIO settings for a pushbutton. *
- Parameters: uint8_t buttonNum: Not currently being used but is * a placeholder if more than one pushbutton will be used in * the future. *
- Return Value: NONE *
-

4.28.1.2 void PORTA_IRQHandler ()

4.29 Modules/pushbutton.h File Reference

```
#include "includeall.h"
```

Include dependency graph for pushbutton.h: This graph shows which files directly or indirectly include this file:

Macros

- #define NUM_BUTTONS 1
- #define BUTTON0_PIN 0x00001000
- #define BUTTON0 PORTA_PCR12

Functions

- void Button_Init (uint8_t buttonNum)

*Function: Button_Init **

- *Description: Initializes the GPIO settings for a pushbutton. **
- *Parameters: uint8_t buttonNum: Not currently being used but is * a placeholder if more than one pushbutton will be used in * the future. **
- *Return Value: NONE **
-

4.29.1 Macro Definition Documentation

4.29.1.1 #define BUTTON0 PORTA_PCR12

4.29.1.2 #define BUTTON0_PIN 0x00001000

4.29.1.3 #define NUM_BUTTONS 1

4.29.2 Function Documentation

4.29.2.1 void Button_Init (uint8_t buttonNum)

Function: Button_Init *

- *Description: Initializes the GPIO settings for a pushbutton. **
- *Parameters: uint8_t buttonNum: Not currently being used but is * a placeholder if more than one pushbutton will be used in * the future. **
- *Return Value: NONE **
-

4.30 Modules/singlewirecomms.c File Reference

```
#include "singlewirecomms.h"
Include dependency graph for singlewirecomms.c:
```

Functions

- void [SWC_Init](#) ()
 - Function: *SWC_Init **
 - Description: *Initializes the GPIO for SWC **
 - Parameters: *NONE*
 - Return Value: *NONE **
- [uint8_t SWC_ResetAndPresencePulses](#) ()
 - Function: *SWC_ResetAndPresencePulses **
 - Description: *Pulls the comms line low for 480 us to reset the bus. Then* all slaves on the bus will pull line low after the master * releases the line. **
 - Parameters: *NONE **
 - Return Value: *uint8_t: Boolean indicating if a slave is present on the * bus. **
- void [SWC_SendByte](#) (uint8_t data)
 - Function: *SWC_SendByte **
 - Description: *Bit bangs out a byte of data on the SWC bus. **
 - Parameters: *uint8_t data: Byte to be sent on the bus. **
 - Return Value: *NONE **
- void [SWC_ReadData](#) (uint8_t bytesToRead, uint8_t *data)
 - Function: *SWC_ReadData **
 - Description: *Reads data on the bus. Each read window is initiated by * the master by pulling the line low for 1 us and then * released. Wait 15 us and then sample when the data from * the slave will be valid. If the bit is 1, then the slave * will release the line high, and keep it low if the bit is * a zero. **
 - Parameters: *uint8_t bytesToRead: Number of bytes that will be read * uint8_t * data: Pointer to a data buffer where the read * data is to be placed. **
 - Return Value: *NONE **
- void [SWC_ReadStatusAndWait](#) ()
 - Function: *SWC_ReadStatusAndWait **
 - Description: *Sometimes the slave will keep the line low while it * working a certain task. **
 - Parameters: *NONE **
 - Return Value: *NONE **

4.30.1 Function Documentation

4.30.1.1 void SWC_Init ()

Function: *SWC_Init **

- Description: *Initializes the GPIO for SWC **
- Parameters: *NONE*
- Return Value: *NONE **
-

4.30.1.2 void SWC_ReadData (uint8_t bytesToRead, uint8_t * data)

Function: SWC_ReadData *

- Description: Reads data on the bus. Each read window is initiated by * the master by pulling the line low for 1 us and then * released. Wait 15 us and then sample when the data from * the slave will be valid. If the bit is 1, then the slave * will release the line high, and keep it low if the bit is * a zero. *
- Parameters: uint8_t bytesToRead: Number of bytes that will be read * uint8_t * data: Pointer to a data buffer where the read * data is to be placed. *
- Return Value: NONE *
-

4.30.1.3 void SWC_ReadStatusAndWait ()

Function: SWC_ReadStatusAndWait *

- Description: Sometimes the slave will keep the line low while it * working a certain task. *
- Parameters: NONE *
- Return Value: NONE *
-

4.30.1.4 uint8_t SWC_ResetAndPresencePulses ()

Function: SWC_ResetAndPresencePulses *

- Description: Pulls the comms line low for 480 us to reset the bus. Then* all slaves on the bus will pull line low after the master * releases the line. *
- Parameters: NONE *
- Return Value: uint8_t: Boolean indicating if a slave is present on the * bus. *
-

4.30.1.5 void SWC_SendByte (uint8_t data)

Function: SWC_SendByte *

- Description: Bit bangs out a byte of data on the SWC bus. *
- Parameters: uint8_t data: Byte to be sent on the bus. *
- Return Value: NONE *
-

4.31 Modules/singlewirecomms.h File Reference

```
#include "includeall.h"
```

Include dependency graph for singlewirecomms.h: This graph shows which files directly or indirectly include this file:

Macros

- #define [DATA_LINE](#) PORTA_PCR5
- #define [DATA_LINE_PIN](#) 0x00000020
- #define [SWITCH_TO_RX](#) CLEAR_BITS_IN_REG(GPIOA_PDDR, [DATA_LINE_PIN](#))
- #define [SWITCH_TO_TX](#) SET_BIT_IN_REG(GPIOA_PDDR, [DATA_LINE_PIN](#))
- #define [WRITE_0](#) SET_BIT_IN_REG(GPIOA_PCOR, [DATA_LINE_PIN](#))
- #define [WRITE_1](#) SET_BIT_IN_REG(GPIOA_PSOR, [DATA_LINE_PIN](#))
- #define [PULL_LOW](#) [WRITE_0](#)
- #define [RELEASE_LINE](#) [WRITE_1](#)
- #define [READ_LINE](#) (GPIOA_PDIR & [DATA_LINE_PIN](#)) >> 5
- #define [MAX_BYTES](#) 8

Functions

- void [SWC_Init](#) ()
 - Function: *SWC_Init* *
 - Description: *Initializes the GPIO for SWC **
 - Parameters: *NONE*
 - Return Value: *NONE* *
 -
- uint8_t [SWC_ResetAndPresencePulses](#) ()
 - Function: *SWC_ResetAndPresencePulses* *
 - Description: *Pulls the comms line low for 480 us to reset the bus. Then* all slaves on the bus will pull line low after the master * releases the line. **
 - Parameters: *NONE* *
 - Return Value: *uint8_t: Boolean indicating if a slave is present on the * bus. **
 -
- void [SWC_SendByte](#) (uint8_t data)
 - Function: *SWC_SendByte* *
 - Description: *Bit bangs out a byte of data on the SWC bus. **
 - Parameters: *uint8_t data: Byte to be sent on the bus. **
 - Return Value: *NONE* *
 -
- void [SWC_ReadData](#) (uint8_t bytesToRead, uint8_t *data)
 - Function: *SWC_ReadData* *
 - Description: *Reads data on the bus. Each read window is initiated by * the master by pulling the line low for 1 us and then * released. Wait 15 us and then sample when the data from * the slave will be valid. If the bit is 1, then the slave * will release the line high, and keep it low if the bit is * a zero. **
 - Parameters: *uint8_t bytesToRead: Number of bytes that will be read * uint8_t * data: Pointer to a data buffer where the read * data is to be placed. **
 - Return Value: *NONE* *
 -
- void [SWC_ReadStatusAndWait](#) ()
 - Function: *SWC_ReadStatusAndWait* *
 - Description: *Sometimes the slave will keep the line low while it * working a certain task. **
 - Parameters: *NONE* *
 - Return Value: *NONE* *
 -

4.31.1 Macro Definition Documentation

4.31.1.1 `#define DATA_LINE PORTA_PCR5`

4.31.1.2 `#define DATA_LINE_PIN 0x00000020`

4.31.1.3 `#define MAX_BYTES 8`

4.31.1.4 `#define PULL_LOW WRITE_0`

4.31.1.5 `#define READ_LINE (GPIOA_PDIR & DATA_LINE_PIN) >> 5`

4.31.1.6 `#define RELEASE_LINE WRITE_1`

4.31.1.7 `#define SWITCH_TO_RX CLEAR_BITS_IN_REG(GPIOA_PDDR, DATA_LINE_PIN)`

4.31.1.8 `#define SWITCH_TO_TX SET_BIT_IN_REG(GPIOA_PDDR, DATA_LINE_PIN)`

4.31.1.9 `#define WRITE_0 SET_BIT_IN_REG(GPIOA_PCOR, DATA_LINE_PIN)`

4.31.1.10 `#define WRITE_1 SET_BIT_IN_REG(GPIOA_PSOR, DATA_LINE_PIN)`

4.31.2 Function Documentation

4.31.2.1 `void SWC_Init ()`

Function: `SWC_Init` *

- Description: Initializes the GPIO for SWC *
- Parameters: NONE
- Return Value: NONE *
-

4.31.2.2 `void SWC_ReadData (uint8_t bytesToRead, uint8_t * data)`

Function: `SWC_ReadData` *

- Description: Reads data on the bus. Each read window is initiated by * the master by pulling the line low for 1 us and then * released. Wait 15 us and then sample when the data from * the slave will be valid. If the bit is 1, then the slave * will release the line high, and keep it low if the bit is * a zero. *
- Parameters: `uint8_t bytesToRead`: Number of bytes that will be read * `uint8_t * data`: Pointer to a data buffer where the read * data is to be placed. *
- Return Value: NONE *
-

4.31.2.3 void SWC_ReadStatusAndWait ()

Function: SWC_ReadStatusAndWait *

- Description: Sometimes the slave will keep the line low while it * working a certain task. *
- Parameters: NONE *
- Return Value: NONE *
-

4.31.2.4 uint8_t SWC_ResetAndPresencePulses ()

Function: SWC_ResetAndPresencePulses *

- Description: Pulls the comms line low for 480 us to reset the bus. Then* all slaves on the bus will pull line low after the master * releases the line. *
- Parameters: NONE *
- Return Value: uint8_t: Boolean indicating if a slave is present on the * bus. *
-

4.31.2.5 void SWC_SendByte (uint8_t data)

Function: SWC_SendByte *

- Description: Bit bangs out a byte of data on the SWC bus. *
- Parameters: uint8_t data: Byte to be sent on the bus. *
- Return Value: NONE *
-

4.32 Modules/spi.c File Reference

#include "spi.h"

Include dependency graph for spi.c:

Functions

- void [SPI_Init](#) (uint8_t SPI_ch, uint8_t master)

*Function: SPI_Init **

- Description: Initializes a SPI channel along with the RX and TX * circular buffers. *
- Parameters: uint8_t SPI_ch: SPI channel being initialized. * uint8_t master: Is this channel going to be a master or a * slave. A value of 0 will set up the channel* as a slave and anything else will be master*
- Return Value: NONE *

- void [SPI_TransmitData](#) (uint8_t SPI_ch, size_t numBytes)

*Function: SPI_TransmitData **

- Description: Sets up the signals an necessary and starts sending data * out through the SPI channel. This function assumes that * the data to be sent is already in the TX buffer. *
- Parameters: uint8_t SPI_ch: SPI channel being initialized. * size_t numBytes: How many bytes are going to be sent. *
- Return Value: NONE *

Variables

- [CircularBuffer_t](#) * [SPI_RXBuffer](#) [[SPI_CHANNELS](#)]
- [CircularBuffer_t](#) * [SPI_TXBuffer](#) [[SPI_CHANNELS](#)]
- static int32_t [spiDeviceLocation](#) [[SPI_CHANNELS](#)]

4.32.1 Function Documentation

4.32.1.1 void SPI_Init (uint8_t SPI_ch, uint8_t master)

Function: SPI_Init *

- Description: Initializes a SPI channel along with the RX and TX * circular buffers. *
- Parameters: uint8_t SPI_ch: SPI channel being initialized. * uint8_t master: Is this channel going to be a master or a * slave. A value of 0 will set up the channel* as a slave and anything else will be master*
- Return Value: NONE *

4.32.1.2 void SPI_TransmitData (uint8_t SPI_ch, size_t numBytes)

Function: SPI_TransmitData *

- Description: Sets up the signals an necessary and starts sending data * out through the SPI channel. This function assumes that * the data to be sent is already in the TX buffer. *
- Parameters: uint8_t SPI_ch: SPI channel being initialized. * size_t numBytes: How many bytes are going to be sent. *
- Return Value: NONE *

4.32.2 Variable Documentation

4.32.2.1 CircularBuffer_t* SPI_RXBuffer[SPI_CHANNELS]

4.32.2.2 CircularBuffer_t* SPI_TXBuffer[SPI_CHANNELS]

4.32.2.3 int32_t spiDeviceLocation[SPI_CHANNELS] [static]

4.33 Modules/spi.h File Reference

```
#include "includeall.h"
```

Include dependency graph for spi.h: This graph shows which files directly or indirectly include this file:

Macros

- #define [SPI_RXBUFFER_SIZE](#) 128
- #define [SPI_TXBUFFER_SIZE](#) 128
- #define [SPI_CHANNELS](#) 2
- #define [SPI_1Mbps_PRESCALER](#) prescaler3
- #define [SPI_1Mbps_BRD](#) divisor8
- #define [SPI_2Mbps_PRESCALER](#) prescaler3
- #define [SPI_2Mbps_BRD](#) divisor4
- #define [SPI_0_5Mbps_PRESCALER](#) prescaler6
- #define [SPI_0_5Mbps_BRD](#) divisor8
- #define [SPI0_MOSI](#) PORTC_PCR6
- #define [SPI0_SCK](#) PORTC_PCR5
- #define [SPI0_MISO](#) PORTC_PCR7
- #define [SPI0_CS](#) PORTC_PCR4
- #define [SPI0_CS_PIN](#) 0x0010
- #define [SPI1_MOSI](#) PORTE_PCR1
- #define [SPI1_SCK](#) PORTE_PCR2
- #define [SPI1_MISO](#) PORTE_PCR3
- #define [SPI1_CS](#) PORTE_PCR4
- #define [SPI1_CS_PIN](#) 0x0010
- #define [DEVICE_LOC](#) "/dev/spidev1.0"
- #define [MODE](#) 1
- #define [BPW](#) 8
- #define [BBB_SPI_SPEED](#) 500000
- #define [ARRAY_SIZE](#)(a) (sizeof(a) / sizeof((a)[0]))

Enumerations

- enum [SPI_BRPrescaler](#) {
[prescaler1](#) = 0, [prescaler2](#), [prescaler3](#), [prescaler4](#),
[prescaler5](#), [prescaler6](#), [prescaler7](#), [prescaler8](#) }
- enum [SPI_BRDivisor](#) {
[divisor2](#) = 0, [divisor4](#), [divisor8](#), [divisor16](#),
[divisor32](#), [divisor64](#), [divisor128](#), [divisor256](#),
[divisor512](#) }

Functions

- void [SPI_Init](#) (uint8_t SPI_ch, uint8_t master)
*Function: SPI_Init **
 - *Description: Initializes a SPI channel along with the RX and TX * circular buffers. **
 - *Parameters: uint8_t SPI_ch: SPI channel being initialized. * uint8_t master: Is this channel going to be a master or a * slave. A value of 0 will set up the channel* as a slave and anything else will be master**
 - *Return Value: NONE **
- void [SPI_TransmitData](#) (uint8_t SPI_ch, size_t numBytes)
*Function: SPI_TransmitData **
 - *Description: Sets up the signals an necessary and starts sending data * out through the SPI channel. This function assumes that * the data to be sent is already in the TX buffer. **
 - *Parameters: uint8_t SPI_ch: SPI channel being initialized. * size_t numBytes: How many bytes are going to be sent. **
 - *Return Value: NONE **

4.33.1 Macro Definition Documentation

4.33.1.1 `#define ARRAY_SIZE(a) (sizeof(a) / sizeof((a)[0]))`

4.33.1.2 `#define BBB_SPI_SPEED 500000`

4.33.1.3 `#define BPW 8`

4.33.1.4 `#define DEVICE_LOC "/dev/spidev1.0"`

4.33.1.5 `#define MODE 1`

4.33.1.6 `#define SPI0_CS PORTC_PCR4`

4.33.1.7 `#define SPI0_CS_PIN 0x0010`

4.33.1.8 `#define SPI0_MISO PORTC_PCR7`

4.33.1.9 `#define SPI0_MOSI PORTC_PCR6`

4.33.1.10 `#define SPI0_SCK PORTC_PCR5`

4.33.1.11 `#define SPI1_CS PORTE_PCR4`

4.33.1.12 `#define SPI1_CS_PIN 0x0010`

4.33.1.13 `#define SPI1_MISO PORTE_PCR3`

4.33.1.14 `#define SPI1_MOSI PORTE_PCR1`

4.33.1.15 `#define SPI1_SCK PORTE_PCR2`

4.33.1.16 `#define SPI_0_5Mbps_BRD divisor8`

4.33.1.17 `#define SPI_0_5Mbps_PRESCALER prescaler6`

4.33.1.18 `#define SPI_1Mbps_BRD divisor8`

4.33.1.19 `#define SPI_1Mbps_PRESCALER prescaler3`

4.33.1.20 `#define SPI_2Mbps_BRD divisor4`

4.33.1.21 `#define SPI_2Mbps_PRESCALER prescaler3`

4.33.1.22 `#define SPI_CHANNELS 2`

4.33.1.23 `#define SPI_RXBUFFER_SIZE 128`

4.33.1.24 `#define SPI_TXBUFFER_SIZE 128`

4.33.2 Enumeration Type Documentation

4.33.2.1 enum `SPI_BRDivisor`

Enumerator

divisor2
divisor4
divisor8
divisor16
divisor32
divisor64
divisor128
divisor256
divisor512

4.33.2.2 enum `SPI_BRPrescaler`

Enumerator

prescaler1
prescaler2
prescaler3
prescaler4
prescaler5
prescaler6
prescaler7
prescaler8

4.33.3 Function Documentation

4.33.3.1 void `SPI_Init (uint8_t SPI_ch, uint8_t master)`

Function: `SPI_Init` *

- Description: Initializes a SPI channel along with the RX and TX * circular buffers. *
- Parameters: `uint8_t SPI_ch`: SPI channel being initialized. * `uint8_t master`: Is this channel going to be a master or a * slave. A value of 0 will set up the channel* as a slave and anything else will be master*
- Return Value: NONE *

4.33.3.2 void SPI_TransmitData (uint8_t SPI_ch, size_t numBytes)

Function: SPI_TransmitData *

- Description: Sets up the signals an necessary and starts sending data * out through the SPI channel. This function assumes that * the data to be sent is already in the TX buffer. *
- Parameters: uint8_t SPI_ch: SPI channel being initialized. * size_t numBytes: How many bytes are going to be sent. *
- Return Value: NONE *

4.34 Modules/timers.c File Reference

```
#include "timers.h"
```

Include dependency graph for timers.c:

Functions

- uint32_t [GetTime](#) ()
- uint32_t [GetElapsedTime](#) (uint32_t start, uint32_t end)

4.34.1 Function Documentation

4.34.1.1 uint32_t [GetElapsedTime](#) (uint32_t start, uint32_t end) `[inline]`

4.34.1.2 uint32_t [GetTime](#) () `[inline]`

4.35 Modules/timers.h File Reference

```
#include "includeall.h"
```

Include dependency graph for timers.h: This graph shows which files directly or indirectly include this file:

Macros

- #define [MAX_MODULUS](#) 0xFFFF
- #define [MAX_PRESCALER](#) 0x7
- #define [NS_PER_SEC](#) 1000000000
- #define [_10US_PER_SEC](#) 10000000
- #define [UNITS_US](#) 1000000
- #define [NS_PER_US](#) 1000
- #define [PROFILER_TPM](#) 1
- #define [PROFILER_CH](#) 1
- #define [PROFILER_PERIOD_IN_NS](#) 10000
- #define [COUNTS_PER_US](#) 48
- #define [WAIT_TPM](#) TPM1
- #define [WAIT_CH](#) 2

Enumerations

- enum [TPM_TriggerOptions](#) {
[externalTriggerPinInput](#) = 0, [CMP0Output](#), [reserved0](#), [reseved1](#),
[pitTrigger0](#), [pitTrigger1](#), [reserved2](#), [reserved3](#),
[tpm0Overflow](#), [tpm1Overflow](#), [tpm2Overflow](#), [reserved4](#),
[rtcAlarm](#), [rtcSeconds](#), [LPTMRtrigger](#), [reserved5](#) }

Functions

- uint32_t [GetTime](#) ()
- uint32_t [GetElapsedTime](#) (uint32_t start, uint32_t end)

4.35.1 Macro Definition Documentation

4.35.1.1 `#define _10US_PER_SEC 10000000`

4.35.1.2 `#define COUNTS_PER_US 48`

4.35.1.3 `#define MAX_MODULUS 0xFFFF`

4.35.1.4 `#define MAX_PRESCALER 0x7`

4.35.1.5 `#define NS_PER_SEC 1000000000`

4.35.1.6 `#define NS_PER_US 1000`

4.35.1.7 `#define PROFILER_CH 1`

4.35.1.8 `#define PROFILER_PERIOD_IN_NS 10000`

4.35.1.9 `#define PROFILER_TPM 1`

4.35.1.10 `#define UNITS_US 1000000`

4.35.1.11 `#define WAIT_CH 2`

4.35.1.12 `#define WAIT_TPM TPM1`

4.35.2 Enumeration Type Documentation

4.35.2.1 enum [TPM_TriggerOptions](#)

Enumerator

externalTriggerPinInput
CMP0Output

```

reserved0
reseved1
pitTrigger0
pitTrigger1
reserved2
reserved3
tpm0Overflow
tpm1Overflow
tpm2Overflow
reserved4
rtcAlarm
rtcSeconds
LPTMRtrigger
reserved5

```

4.35.3 Function Documentation

4.35.3.1 `uint32_t GetElapsedTime (uint32_t start, uint32_t end)` `[inline]`

4.35.3.2 `uint32_t GetTime ()` `[inline]`

4.36 Modules/uart.c File Reference

```
#include "uart.h"
```

Include dependency graph for uart.c:

Functions

- void [UartSetup](#) (uint8_t channel, uint32_t requestedBuadRate, uint8_t parity)
 - Function: *UartSetup* *
 - Description: Sets up UART. *
 - Parameters: *uint8_t channel*: UART channel to initialize. * *uint32_t requestedBaudRate*: Desired baud rate * *uint8_t parity*: Used as boolean if parity is desired. *
 - Return Value: NONE *
- void [UartTX](#) (uint8_t *buffer, uint32_t length)
 - Function: *UartTX* *
 - Description: Uses a blocking method to transmit on UART. *
 - Parameters: *uint8_t * buffer*: pointer to an array of characters that * are to be transmitted through the UART * *uint32_t length*: Num of bytes to be sent through *
 - Return Value: NONE *
- int8_t [UartRX](#) (uint8_t *data)
 - Function: *UartRX* *
 - Description: Uses the polling method to receive data through UART *
 - Parameters: NONE *
 - Return Value: *uint8_t*: byte received from UART *
- void [PutChar](#) (uint8_t data)
 - Function: *PutChar* *
 - Description: Transmits a single character through UART0 *
 - Parameters: *uint8_t data*: character to be sent. *
 - Return Value: NONE *

Variables

- [uint8_t parseDiag](#)
- [CircularBuffer_t * UART0_RXBuffer](#)
- [CircularBuffer_t * UART0_TXBuffer](#)
- [CircularBuffer_t * UART1_RXBuffer](#)
- [CircularBuffer_t * UART1_TXBuffer](#)

4.36.1 Function Documentation

4.36.1.1 void PutChar (uint8_t *data*)

Function: PutChar *

- Description: Transmits a single character through UART0 *
- Parameters: uint8_t data: character to be sent. *
- Return Value: NONE *

4.36.1.2 int8_t UartRX (uint8_t * *data*)

Function: UartRX *

- Description: Uses the polling method to receive data through UART *
- Parameters: NONE *
- Return Value: uint8_t: byte received from UART *

4.36.1.3 void UartSetup (uint8_t *channel*, uint32_t *requestedBaudRate*, uint8_t *parity*)

Function: UartSetup *

- Description: Sets up UART. *
- Parameters: uint8_t channel: UART channel to initialize. * uint32_t requestedBaudRate: Desired baud rate
* uint8_t parity: Used as boolean if parity is desired. *
- Return Value: NONE *

4.36.1.4 void UartTX (uint8_t * *buffer*, uint32_t *length*)

Function: UartTX *

- Description: Uses a blocking method to transmit on UART. *
- Parameters: uint8_t * buffer: pointer to an array of characters that * are to be transmitted through the UART
* uint32_t length: Num of bytes to be sent through *
- Return Value: NONE *

4.36.2 Variable Documentation

4.36.2.1 uint8_t parseDiag

4.36.2.2 CircularBuffer_t* UART0_RXBuffer

4.36.2.3 CircularBuffer_t* UART0_TXBuffer

4.36.2.4 CircularBuffer_t* UART1_RXBuffer

4.36.2.5 CircularBuffer_t* UART1_TXBuffer

4.37 Modules/uart.h File Reference

```
#include "includeall.h"
```

Include dependency graph for uart.h: This graph shows which files directly or indirectly include this file:

Macros

- #define [OSR](#) 16
- #define [CR](#) 13
- #define [LF](#) 10
- #define [RXBUFFER_SIZE](#) 128
- #define [TXBUFFER_SIZE](#) 128
- #define [DMA_RXBUFFER_SIZE_128Bytes](#)
- #define [DMA_TXBUFFER_SIZE_1kBytes](#)
- #define [DMACH_UART0RX_NO_DMA](#)
- #define [DMACH_UART0TX_NO_DMA](#)
- #define [MODEMDEVICE](#) "/dev/ttyO1"
- #define [BONEPATH](#) "/sys/devices/bone_capemgr.9/slots"
- #define [POSIX_SOURCE](#) 1
- #define [BAUDRATE](#) B115200

Functions

- void [UartSetup](#) (uint8_t channel, uint32_t buadRate, uint8_t parity)
 - Function: UartSetup **
 - *Description: Sets up UART. **
 - *Parameters: uint8_t channel: UART channel to initialize. * uint32_t requestedBaudRate: Desired baud rate * uint8_t parity: Used as boolean if parity is desired. **
 - *Return Value: NONE **
- void [UartTX](#) (uint8_t *buffer, uint32_t length)
 - Function: UartTX **
 - *Description: Uses a blocking method to transmit on UART. **
 - *Parameters: uint8_t * buffer: pointer to an array of characters that * are to be transmitted through the UART * uint32_t length: Num of bytes to be sent through **
 - *Return Value: NONE **
- int8_t [UartRX](#) (uint8_t *data)
 - Function: UartRX **
 - *Description: Uses the polling method to receive data through UART **
 - *Parameters: NONE **
 - *Return Value: uint8_t: byte received from UART **
- void [PutChar](#) (uint8_t data)
 - Function: PutChar **
 - *Description: Transmits a single character through UART0 **
 - *Parameters: uint8_t data: character to be sent. **
 - *Return Value: NONE **

Variables

- `uint8_t` [parseDiag](#)

4.37.1 Macro Definition Documentation

4.37.1.1 `#define BAUDRATE B115200`

4.37.1.2 `#define BONEPATH "/sys/devices/bone_capemgr.9/slots"`

4.37.1.3 `#define CR 13`

4.37.1.4 `#define DMA_RXBUFFER_SIZE _128Bytes`

4.37.1.5 `#define DMA_TXBUFFER_SIZE _1kBytes`

4.37.1.6 `#define DMACH_UART0RX NO_DMA`

4.37.1.7 `#define DMACH_UART0TX NO_DMA`

4.37.1.8 `#define LF 10`

4.37.1.9 `#define MODEMDEVICE "/dev/ttyO1"`

4.37.1.10 `#define OSR 16`

4.37.1.11 `#define POSIX_SOURCE 1`

4.37.1.12 `#define RXBUFFER_SIZE 128`

4.37.1.13 `#define TXBUFFER_SIZE 128`

4.37.2 Function Documentation

4.37.2.1 `void PutChar (uint8_t data)`

Function: PutChar *

- Description: Transmits a single character through UART0 *
- Parameters: `uint8_t data`: character to be sent. *
- Return Value: NONE *

4.37.2.2 `int8_t UartRX (uint8_t * data)`

Function: UartRX *

- Description: Uses the polling method to receive data through UART *
- Parameters: NONE *
- Return Value: `uint8_t`: byte received from UART *

4.37.2.3 `void UartSetup (uint8_t channel, uint32_t baudRate, uint8_t parity)`

Function: UartSetup *

- Description: Sets up UART. *
- Parameters: `uint8_t channel`: UART channel to initialize. * `uint32_t requestedBaudRate`: Desired baud rate
* `uint8_t parity`: Used as boolean if parity is desired. *
- Return Value: NONE *

4.37.2.4 `void UartTX (uint8_t * buffer, uint32_t length)`

Function: UartTX *

- Description: Uses a blocking method to transmit on UART. *
- Parameters: `uint8_t * buffer`: pointer to an array of characters that * are to be transmitted through the UART
* `uint32_t length`: Num of bytes to be sent through *
- Return Value: NONE *

4.37.3 Variable Documentation

4.37.3.1 `uint8_t parseDiag`

4.38 README.txt File Reference

Variables

- Makefile [commands](#)
- Makefile [assemble](#)
- Makefile and [link](#) for the three [main](#) targets make [upload](#)
- Makefile and [link](#) for the three [main](#) targets make [link](#)
- Makefile and [link](#) for the three [main](#) targets make and [upload](#) the executable to the Beagle Bone Black make build [lib](#)
- Makefile and [link](#) for the three [main](#) targets make and [upload](#) the executable to the Beagle Bone Black make build * [d](#)

- Makefile and [link](#) for the three [main](#) targets make and [upload](#) the executable to the Beagle Bone Black make build * [i](#)
- Makefile and [link](#) for the three [main](#) targets make and [upload](#) the executable to the Beagle Bone Black make build * [S](#)
- Makefile and [link](#) for the three [main](#) targets make and [upload](#) the executable to the Beagle Bone Black make build *map or *a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as [follows](#)
- Makefile and [link](#) for the three [main](#) targets make and [upload](#) the executable to the Beagle Bone Black make build *map or *a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as [green](#)
- Makefile and [link](#) for the three [main](#) targets make and [upload](#) the executable to the Beagle Bone Black make build *map or *a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as [blue](#)
- Makefile and [link](#) for the three [main](#) targets make and [upload](#) the executable to the Beagle Bone Black make build *map or *a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as [yellow](#)
- Makefile and [link](#) for the three [main](#) targets make and [upload](#) the executable to the Beagle Bone Black make build *map or *a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as [purple](#)
- Makefile and [link](#) for the three [main](#) targets make and [upload](#) the executable to the Beagle Bone Black make build *map or *a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as [cyan](#)
- Makefile and [link](#) for the three [main](#) targets make and [upload](#) the executable to the Beagle Bone Black make build *map or *a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as [white](#)
- Makefile and [link](#) for the three [main](#) targets make and [upload](#) the executable to the Beagle Bone Black make build *map or *a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as and off set [power x](#)

4.38.1 Variable Documentation

4.38.1.1 Makefile and link for the three main targets make assemble

4.38.1.2 Makefile and link for the three main targets make and upload the executable to the Beagle Bone Black make build * map or* a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as blue

4.38.1.3 Makefile commands

4.38.1.4 Makefile and link for the three main targets make and upload the executable to the Beagle Bone Black make build * map or* a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as cyan

4.38.1.5 Makefile and link for the three main targets make and upload the executable to the Beagle Bone Black make build * d

4.38.1.6 Makefile and link for the three main targets make and upload the executable to the Beagle Bone Black make build * map or* a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as follows

4.38.1.7 Makefile and link for the three main targets make and upload the executable to the Beagle Bone Black make build * map or* a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as green

- 4.38.1.8 Makefile and link for the three main targets make and upload the executable to the Beagle Bone Black make build * i
- 4.38.1.9 Makefile and link for the three main targets make and upload the executable to the Beagle Bone Black make build lib
- 4.38.1.10 Makefile and link for the three main targets make link
- 4.38.1.11 Makefile and link for the three main targets make and upload the executable to the Beagle Bone Black make build * map or* a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as purple
- 4.38.1.12 Makefile and link for the three main targets make and upload the executable to the Beagle Bone Black make build * S
- 4.38.1.13 Makefile and link for the three main targets make upload
- 4.38.1.14 Makefile and link for the three main targets make and upload the executable to the Beagle Bone Black make build * map or* a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as white
- 4.38.1.15 Makefile and link for the three main targets make and upload the executable to the Beagle Bone Black make build * map or* a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as and off set power x
- 4.38.1.16 Makefile and link for the three main targets make and upload the executable to the Beagle Bone Black make build * map or* a file located in the directory The size of the executable will be outputed after the specified build target is finished building The make file will output an executable as yellow

4.39 TemperatureController/controller.c File Reference

```
#include "controller.h"
```

Include dependency graph for controller.c:

Functions

- void [Controller_Init](#) ()
 - Function: *Controller_Init **
 - Description: *Initializes the the static variables **
 - Parameters: *NONE **
 - Return Value: *NONE **
- void [Controller_StateMachine](#) ()
 - Function: *Controller_StateMachine **
 - Description: *Initializes the the static variables **
 - Parameters: *NONE **
 - Return Value: *NONE **
- void [Controller_ChangeState](#) ()
 - Function: *Controller_ChangeState **
 - Description: *Updates the state the next state or to the start **
 - Parameters: *NONE **
 - Return Value: *NONE **

- void [Controller_SetCurrentTemp](#) (uint8_t newTemp)
 - Function: *Controller_SetCurrentTemp **
 - Description: *Updates the currenrt temperature value to a new value **
 - Parameters: *uint8_t newTemp: New value of the current temperature. **
 - Return Value: *NONE **
- void [Controller_SetDesiredTemp](#) (uint8_t newTemp)
 - Function: *Controller_SetDesiredTemp **
 - Description: *Updates the desired temperature value to a new value **
 - Parameters: *uint8_t newTemp: New value of the desired temperature. **
 - Return Value: *NONE **
- void [Controller_SetTempRange](#) (uint8_t newRange)
 - Function: *Controller_SetTempRange **
 - Description: *Updates the temperature range value to a new value **
 - Parameters: *uint8_t newRange: New value of the new temperature range. **
 - Return Value: *NONE **
- void [Controller_ChangeDisplay](#) (uint8_t value)
 - Function: *Controller_ChangeDisplay **
 - Description: *Updates the displays with the input value **
 - Parameters: *uint8_t value: New value sent to the diplays **
 - Return Value: *NONE **
- void [Controller_SendTempData](#) (uint8_t dontcare)
 - Function: *Controller_SendTempData **
 - Description: *Packeges the current values of the controller into a * message and the sends it back through UART **
 - Parameters: *NONE **
 - Return Value: *NONE **

Variables

- static uint8_t [currentTemp](#)
- static uint8_t [desiredTemp](#)
- static uint8_t [tempRange](#)
- static uint8_t [power](#)
- static [ControllerState_e](#) [state](#)

4.39.1 Function Documentation

4.39.1.1 void Controller_ChangeDisplay (uint8_t value)

Function: *Controller_ChangeDisplay **

- Description: *Updates the displays with the input value **
- Parameters: *uint8_t value: New value sent to the diplays **
- Return Value: *NONE **

4.39.1.2 void Controller_ChangeState ()

Function: Controller_ChangeState *

- Description: Updates the state the next state or to the start *
- Parameters: NONE *
- Return Value: NONE *

4.39.1.3 void Controller_Init ()

Function: Controller_Init *

- Description: Initializes the the static variables *
- Parameters: NONE *
- Return Value: NONE *

4.39.1.4 void Controller_SendTempData (uint8_t *dontcare*)

Function: Controller_SendTempData *

- Description: Packages the current values of the controller into a * message and the sends it back through UART *
- Parameters: NONE *
- Return Value: NONE *

4.39.1.5 void Controller_SetCurrentTemp (uint8_t *newTemp*)

Function: Controller_SetCurrentTemp *

- Description: Updates the curernt temperature value to a new value *
- Parameters: uint8_t newTemp: New value of the current temperature. *
- Return Value: NONE *

4.39.1.6 void Controller_SetDesiredTemp (uint8_t *newTemp*)

Function: Controller_SetDesiredTemp *

- Description: Updates the desired temperature value to a new value *
- Parameters: uint8_t newTemp: New value of the desired temperature. *
- Return Value: NONE *

4.39.1.7 void Controller_SetTempRange (uint8_t newRange)

Function: Controller_SetTempRange *

- Description: Updates the temperature range value to a new value *
- Parameters: uint8_t newRange: New value of the new temperature range. *
- Return Value: NONE *

4.39.1.8 void Controller_StateMachine ()

Function: Controller_StateMachine *

- Description: Initializes the the static variables *
- Parameters: NONE *
- Return Value: NONE *

4.39.2 Variable Documentation

4.39.2.1 uint8_t currentTemp [static]

4.39.2.2 uint8_t desiredTemp [static]

4.39.2.3 uint8_t power [static]

4.39.2.4 ControllerState_e state [static]

4.39.2.5 uint8_t tempRange [static]

4.40 TemperatureController/controller.h File Reference

```
#include "includeall.h"
```

Include dependency graph for controller.h: This graph shows which files directly or indirectly include this file:

Macros

- #define MAX_DISPLAY_VAL 99
- #define CONVERT_C_TO_F(tc) (tc) = ((tc) * 9 / 5) + 32
- #define CONVERT_F_TO_C(tf) (tf) = ((tf) - 32) * 5 / 9
- #define RELAY PORTC_PCR9
- #define RELAY_PIN 0x00000200
- #define RELAY_ON SET_BIT_IN_REG(GPIOC_PSOR, RELAY_PIN)
- #define RELAY_OFF SET_BIT_IN_REG(GPIOC_PCOR, RELAY_PIN)

Enumerations

- enum `ControllerState_e` { `noChange` = 0, `changeDesiredTemp`, `changeTempRange`, `printSettings` }

Functions

- void `Controller_Init` ()

Function: `Controller_Init` *

- Description: Initializes the the static variables *
- Parameters: `NONE` *
- Return Value: `NONE` *

- void `Controller_StateMachine` ()

Function: `Controller_StateMachine` *

- Description: Initializes the the static variables *
- Parameters: `NONE` *
- Return Value: `NONE` *

- void `Controller_ChangeState` ()

Function: `Controller_ChangeState` *

- Description: Updates the state the next state or to the start *
- Parameters: `NONE` *
- Return Value: `NONE` *

- void `Controller_SetCurrentTemp` (uint8_t newTemp)

Function: `Controller_SetCurrentTemp` *

- Description: Updates the curernt temperature value to a new value *
- Parameters: `uint8_t newTemp`: New value of the current temperature. *
- Return Value: `NONE` *

- void `Controller_SetDesiredTemp` (uint8_t newTemp)

Function: `Controller_SetDesiredTemp` *

- Description: Updates the desired temperature value to a new value *
- Parameters: `uint8_t newTemp`: New value of the desired temperature. *
- Return Value: `NONE` *

- void `Controller_SetTempRange` (uint8_t newRange)

Function: `Controller_SetTempRange` *

- Description: Updates the temperature range value to a new value *
- Parameters: `uint8_t newRange`: New value of the new temperature range. *
- Return Value: `NONE` *

- void `Controller_ChangeDisplay` (uint8_t value)

Function: `Controller_ChangeDisplay` *

- Description: Updates the displays with the input value *
- Parameters: `uint8_t value`: New value sent to the diplays *
- Return Value: `NONE` *

- void `Controller_SendTempData` (uint8_t dontcare)

Function: `Controller_SendTempData` *

- Description: Packages the current values of the controller into a * message and the sends it back through UART *
- Parameters: `NONE` *
- Return Value: `NONE` *

4.40.1 Macro Definition Documentation

4.40.1.1 `#define CONVERT_C_TO_F(tc) (tc) = ((tc) * 9 / 5) + 32`

4.40.1.2 `#define CONVERT_F_TO_C(tf) (tf) = ((tf) - 32) * 5 / 9`

4.40.1.3 `#define MAX_DISPLAY_VAL 99`

4.40.1.4 `#define RELAY PORTC_PCR9`

4.40.1.5 `#define RELAY_OFF SET_BIT_IN_REG(GPIOC_PCOR, RELAY_PIN)`

4.40.1.6 `#define RELAY_ON SET_BIT_IN_REG(GPIOC_PSOR, RELAY_PIN)`

4.40.1.7 `#define RELAY_PIN 0x00000200`

4.40.2 Enumeration Type Documentation

4.40.2.1 `enum ControllerState_e`

Enumerator

noChange
changeDesiredTemp
changeTempRange
printSettings

4.40.3 Function Documentation

4.40.3.1 `void Controller_ChangeDisplay (uint8_t value)`

Function: Controller_ChangeDisplay *

- Description: Updates the displays with the input value *
- Parameters: uint8_t value: New value sent to the diplays *
- Return Value: NONE *

4.40.3.2 `void Controller_ChangeState ()`

Function: Controller_ChangeState *

- Description: Updates the state the next state or to the start *
- Parameters: NONE *
- Return Value: NONE *

4.40.3.3 void Controller_Init ()

Function: Controller_Init *

- Description: Initializes the the static variables *
- Parameters: NONE *
- Return Value: NONE *

4.40.3.4 void Controller_SendTempData (uint8_t *dontcare*)

Function: Controller_SendTempData *

- Description: Packages the current values of the controller into a * message and the sends it back through UART *
- Parameters: NONE *
- Return Value: NONE *

4.40.3.5 void Controller_SetCurrentTemp (uint8_t *newTemp*)

Function: Controller_SetCurrentTemp *

- Description: Updates the currenrt temperature value to a new value *
- Parameters: uint8_t newTemp: New value of the current temperature. *
- Return Value: NONE *

4.40.3.6 void Controller_SetDesiredTemp (uint8_t *newTemp*)

Function: Controller_SetDesiredTemp *

- Description: Updates the desired temperature value to a new value *
- Parameters: uint8_t newTemp: New value of the desired temperature. *
- Return Value: NONE *

4.40.3.7 void Controller_SetTempRange (uint8_t *newRange*)

Function: Controller_SetTempRange *

- Description: Updates the temperature range value to a new value *
- Parameters: uint8_t newRange: New value of the new temperature range. *
- Return Value: NONE *

4.40.3.8 void Controller_StateMachine ()

Function: Controller_StateMachine *

- Description: Initializes the the static variables *
- Parameters: NONE *
- Return Value: NONE *

4.41 Testing/circularbuffertesting.c File Reference

4.42 Testing/circularbuffertesting.h File Reference

4.43 Testing/datatesting.c File Reference

4.44 Testing/datatesting.h File Reference

4.45 Testing/memorytesting.c File Reference

4.46 Testing/memorytesting.h File Reference

```
#include "includeall.h"
```

Include dependency graph for memorytesting.h:

Macros

- #define [MEM_MOVE_SIZE](#) 23
- #define [TESTSTRING](#) "This is my test string"
- #define [MY_MEM_ZERO_SIZE](#) 50
- #define [MY_REVERSE_INPUTS](#) 3
- #define [MY_STR_LEN_INPUTS](#) 2
- #define [MAIN_HEADER](#) 2
- #define [TESTING_DMA_CH](#) NO_DMA

Functions

- void [MemoryTesting](#) (void)
- void [MyMemMoveUnitTest](#) (void)
- void [MyMemSetUnitTest](#) (void)
- void [MyReverseUnitTest](#) (void)
- void [MyStrLenUnitTest](#) (void)

4.46.1 Macro Definition Documentation

4.46.1.1 `#define MAIN_HEADER 2`

4.46.1.2 `#define MEM_MOVE_SIZE 23`

4.46.1.3 `#define MY_MEM_ZERO_SIZE 50`

4.46.1.4 `#define MY_REVERSE_INPUTS 3`

4.46.1.5 `#define MY_STR_LEN_INPUTS 2`

4.46.1.6 `#define TESTING_DMA_CH NO_DMA`

4.46.1.7 `#define TESTSTRING "This is my test string"`

4.46.2 Function Documentation

4.46.2.1 `void MemoryTesting (void)`

4.46.2.2 `void MyMemMoveUnitTest (void)`

4.46.2.3 `void MyMemSetUnitTest (void)`

4.46.2.4 `void MyReverseUnitTest (void)`

4.46.2.5 `void MyStrLenUnitTest (void)`

4.47 Testing/performance/testing.c File Reference

4.48 Testing/performance/testing.h File Reference

```
#include "includeall.h"
```

Include dependency graph for performance/testing.h:

Macros

- `#define MEMORY_TEST_CASES 5`
- `#define PRINTING_TEST_CASES 4`
- `#define PRINTING_BUFFER_SIZE 50`

Functions

- `void PerformanceTesting (void)`
- `void MemoryPerformanceTesting (void)`
- `void DataPerformanceTesting (void)`
- `void PrintingPerformanceTesting (void)`

4.48.1 Macro Definition Documentation

4.48.1.1 `#define MEMORY_TEST_CASES 5`

4.48.1.2 `#define PRINTING_BUFFER_SIZE 50`

4.48.1.3 `#define PRINTING_TEST_CASES 4`

4.48.2 Function Documentation

4.48.2.1 `void DataPerformanceTesting (void)`

4.48.2.2 `void MemoryPerformanceTesting (void)`

4.48.2.3 `void PerformanceTesting (void)`

4.48.2.4 `void PrintingPerformanceTesting (void)`

4.49 Testing/testing.c File Reference

4.50 Testing/testing.h File Reference

Index

`_10US_PER_SEC`
 [timers.h, 87](#)

`_128Bytes`
 [dma.h, 55](#)

`_128kBytes`
 [dma.h, 56](#)

`_16Bytes`
 [dma.h, 55](#)

`_16bit`
 [dma.h, 56](#)

`_16kBytes`
 [dma.h, 56](#)

`_1kBytes`
 [dma.h, 55](#)

`_256Bytes`
 [dma.h, 55](#)

`_256kBytes`
 [dma.h, 56](#)

`_2kBytes`
 [dma.h, 56](#)

`_32Bytes`
 [dma.h, 55](#)

`_32bit`
 [dma.h, 56](#)

`_32kBytes`
 [dma.h, 56](#)

`_4kBytes`
 [dma.h, 56](#)

`_512Bytes`
 [dma.h, 55](#)

`_64Bytes`
 [dma.h, 55](#)

`_64kBytes`
 [dma.h, 56](#)

`_8bit`
 [dma.h, 56](#)

`_8kBytes`
 [dma.h, 56](#)

`__attribute__`
 [DS8B20_ROMCode, 6](#)
 [DS8B20_Scratchpad, 7](#)
 [messaging.h, 47, 48](#)

`ACTIVATE_KEY`
 [nRF24L01.h, 70](#)

`ACTIVATE`
 [nRF24L01.h, 70](#)

`AD10`
 [adc.h, 52](#)

`AD11`
 [adc.h, 52](#)

`AD12`
 [adc.h, 52](#)

`AD13`
 [adc.h, 52](#)

`AD14`
 [adc.h, 52](#)

`AD15`
 [adc.h, 52](#)

`AD16`
 [adc.h, 52](#)

`AD17`
 [adc.h, 52](#)

`AD18`
 [adc.h, 52](#)

`AD19`
 [adc.h, 52](#)

`AD20`
 [adc.h, 52](#)

`AD21`
 [adc.h, 52](#)

`AD22`
 [adc.h, 52](#)

`AD23`
 [adc.h, 52](#)

`AD4`
 [adc.h, 52](#)

`AD5`
 [adc.h, 52](#)

`AD6`
 [adc.h, 52](#)

`AD7`
 [adc.h, 52](#)

`AD8`
 [adc.h, 52](#)

`AD9`
 [adc.h, 52](#)

`ADC0_IRQHandler`
 [adc.c, 49](#)

`ADC_AvgSamples`
 [adc.h, 51](#)

`ADC_CH_PIN`
 [adc.h, 51](#)

`ADC_CHANNEL`
 [adc.h, 51](#)

`ADC_GetCurrentValue`
 [adc.c, 49](#)
 [adc.h, 52](#)

`ADC_Init`

- adc.c, [49](#)
- adc.h, [52](#)
- ADC_InputChannel
 - adc.h, [51](#)
- ADC_PORT
 - adc.h, [51](#)
- ADC_StartConversion
 - adc.c, [49](#)
 - adc.h, [53](#)
- ADC_value
 - adc.c, [50](#)
- ADR_1Mbps
 - nRF24L01.h, [71](#)
- ADR_2Mbps
 - nRF24L01.h, [71](#)
- ADR_e
 - nRF24L01.h, [71](#)
- ALARM_SEARCH
 - ds18b20.h, [60](#)
- ARRAY_SIZE
 - spi.h, [84](#)
- ASCIILETTERBASE
 - data.h, [30](#)
- ASCIINUMBASE
 - data.h, [30](#)
- adc.c
 - ADC0_IRQHandler, [49](#)
 - ADC_GetCurrentValue, [49](#)
 - ADC_Init, [49](#)
 - ADC_StartConversion, [49](#)
 - ADC_value, [50](#)
- adc.h
 - AD10, [52](#)
 - AD11, [52](#)
 - AD12, [52](#)
 - AD13, [52](#)
 - AD14, [52](#)
 - AD15, [52](#)
 - AD16, [52](#)
 - AD17, [52](#)
 - AD18, [52](#)
 - AD19, [52](#)
 - AD20, [52](#)
 - AD21, [52](#)
 - AD22, [52](#)
 - AD23, [52](#)
 - AD4, [52](#)
 - AD5, [52](#)
 - AD6, [52](#)
 - AD7, [52](#)
 - AD8, [52](#)
 - AD9, [52](#)
 - ADC_AvgSamples, [51](#)
 - ADC_CH_PIN, [51](#)
 - ADC_CHANNEL, [51](#)
 - ADC_GetCurrentValue, [52](#)
 - ADC_Init, [52](#)
 - ADC_InputChannel, [51](#)
 - ADC_PORT, [51](#)
 - ADC_StartConversion, [53](#)
 - avg16Samples, [51](#)
 - avg32Samples, [51](#)
 - avg4Samples, [51](#)
 - avg8Samples, [51](#)
 - BANDGAP, [52](#)
 - DADP0, [52](#)
 - DADP1, [52](#)
 - DADP2, [52](#)
 - DADP3, [52](#)
 - DISABLED, [52](#)
 - MAX_VALUE, [51](#)
 - MUX_PIN_ANALOG, [51](#)
 - RESERVED0, [52](#)
 - RESERVED1, [52](#)
 - RESERVED2, [52](#)
 - TEMP_SENSOR, [52](#)
 - VREFSH, [52](#)
 - VREFSL, [52](#)
- arc
 - nRF24L01_SETUP_RETR_t, [20](#)
- arcCNT
 - nRF24L01_OBSERVE_TX_t, [13](#)
- ard
 - nRF24L01_SETUP_RETR_t, [20](#)
- assemble
 - README.txt, [93](#)
- avg16Samples
 - adc.h, [51](#)
- avg32Samples
 - adc.h, [51](#)
- avg4Samples
 - adc.h, [51](#)
- avg8Samples
 - adc.h, [51](#)
- aw
 - nRF24L01_SETUP_AW_t, [19](#)
- B
 - nRF24L01_CONFIG_t, [8](#)
 - nRF24L01_DYNPD_t, [9](#)
 - nRF24L01_EN_RXADDR_t, [10](#)
 - nRF24L01_ENAA_t, [11](#)
 - nRF24L01_FEATURE_t, [12](#)
 - nRF24L01_FIFO_STATUS_t, [13](#)
 - nRF24L01_OBSERVE_TX_t, [13](#)
 - nRF24L01_RF_CH_t, [14](#)
 - nRF24L01_RF_SETUP_t, [15](#)
 - nRF24L01_RX_PW_P0_t, [15](#)
 - nRF24L01_RX_PW_P1_t, [16](#)
 - nRF24L01_RX_PW_P2_t, [17](#)
 - nRF24L01_RX_PW_P3_t, [17](#)
 - nRF24L01_RX_PW_P4_t, [18](#)
 - nRF24L01_RX_PW_P5_t, [18](#)
 - nRF24L01_SETUP_AW_t, [19](#)
 - nRF24L01_SETUP_RETR_t, [20](#)
 - nRF24L01_STATUS_t, [21](#)
- b

- nRF24L01_CONFIG_t, [8](#)
- nRF24L01_DYNPD_t, [9](#)
- nRF24L01_EN_RXADDR_t, [10](#)
- nRF24L01_ENAA_t, [11](#)
- nRF24L01_FEATURE_t, [12](#)
- nRF24L01_FIFO_STATUS_t, [13](#)
- nRF24L01_OBSERVE_TX_t, [13](#)
- nRF24L01_RF_CH_t, [14](#)
- nRF24L01_RF_SETUP_t, [15](#)
- nRF24L01_RX_PW_P0_t, [15](#)
- nRF24L01_RX_PW_P1_t, [16](#)
- nRF24L01_RX_PW_P2_t, [17](#)
- nRF24L01_RX_PW_P3_t, [17](#)
- nRF24L01_RX_PW_P4_t, [18](#)
- nRF24L01_RX_PW_P5_t, [18](#)
- nRF24L01_SETUP_AW_t, [19](#)
- nRF24L01_SETUP_RETR_t, [20](#)
- nRF24L01_STATUS_t, [21](#)
- BANDGAP
 - adc.h, [52](#)
- BAUDRATE
 - uart.h, [91](#)
- BBB_SPI_SPEED
 - spi.h, [84](#)
- BLUE_CHANNEL
 - led.h, [63](#)
- BLUE_PIN
 - led.h, [63](#)
- BLUE_TPM
 - led.h, [63](#)
- BLUE
 - led.h, [63](#)
- BONEPATH
 - uart.h, [91](#)
- BPW
 - spi.h, [84](#)
- BUFFER_EMPTY
 - circularbuffer.h, [25](#)
- BUFFER_FULL
 - circularbuffer.h, [25](#)
- BUFFER_NOT_EMPTY
 - circularbuffer.h, [25](#)
- BUFFER_NOT_FULL
 - circularbuffer.h, [25](#)
- BUTTON0
 - pushbutton.h, [76](#)
- BUTTON0_PIN
 - pushbutton.h, [76](#)
- BYTE0_MASK
 - data.h, [30](#)
- BYTE0_SHIFT
 - data.h, [30](#)
- BYTE1_MASK
 - data.h, [30](#)
- BYTE1_SHIFT
 - data.h, [30](#)
- BYTE2_MASK
 - data.h, [30](#)
- BYTE2_SHIFT
 - data.h, [30](#)
- BYTE3_MASK
 - data.h, [30](#)
- BYTE3_SHIFT
 - data.h, [30](#)
- BigToLittle
 - data.c, [27](#)
 - data.h, [30](#)
- blue
 - README.txt, [93](#)
- BufferDisabled
 - dma.h, [55](#)
- bufferEnd
 - CircularBuffer_t, [5](#)
- BufferSize
 - dma.h, [55](#)
- BufferSize_e
 - dma.h, [55](#)
- bufferStart
 - CircularBuffer_t, [5](#)
- BufferState
 - circularbuffer.h, [25](#)
- BufferState_e
 - circularbuffer.h, [24](#)
- BuildCommandMessage
 - messaging.c, [43](#)
 - messaging.h, [47](#)
- Button_Init
 - pushbutton.c, [75](#)
 - pushbutton.h, [76](#)
- bytesToSend
 - nRF24L01_SPIMessage_t, [20](#)
- CBufferAdd
 - circularbuffer.h, [25](#)
- CBufferAddItems
 - circularbuffer.h, [25](#)
- CBufferDestruct
 - circularbuffer.h, [25](#)
- CBufferInit
 - circularbuffer.h, [25](#)
- CBufferRemove
 - circularbuffer.h, [26](#)
- CHANNEL_0
 - nRF24L01.h, [72](#)
- CHANNEL_1
 - nRF24L01.h, [72](#)
- CHANNEL_2
 - nRF24L01.h, [72](#)
- CHANNEL_3
 - nRF24L01.h, [72](#)
- CHANNEL_4
 - nRF24L01.h, [72](#)
- CHANNEL_5
 - nRF24L01.h, [72](#)
- CHANNEL_6
 - nRF24L01.h, [72](#)
- CHANNEL_7

- nRF24L01.h, [72](#)
- CHANNEL_8
 - nRF24L01.h, [72](#)
- CHANNEL_9
 - nRF24L01.h, [72](#)
- CHANNEL_e
 - nRF24L01.h, [71](#)
- CMP0Output
 - timers.h, [87](#)
- COMMAND_MSG_BYTES
 - messaging.h, [46](#)
- CONFIG
 - nRF24L01.h, [72](#)
- CONVERT_C_TO_F
 - controller.h, [99](#)
- CONVERT_F_TO_C
 - controller.h, [99](#)
- CONVERT_T
 - ds18b20.h, [60](#)
- COPY_SCRATCHPAD
 - ds18b20.h, [60](#)
- COUNTS_PER_US
 - timers.h, [87](#)
- CRCO_MASK
 - nRF24L01.h, [70](#)
- CYAN
 - led.h, [63](#)
- CalculateCommandChecksum
 - messaging.c, [43](#)
 - messaging.h, [47](#)
- CalculateTemperatureChecksum
 - messaging.c, [44](#)
 - messaging.h, [47](#)
- CD
 - nRF24L01.h, [72](#)
- changeColor
 - messaging.h, [46](#)
- changeDesiredTemp
 - controller.h, [99](#)
- changePWM
 - messaging.h, [46](#)
- changeTempRange
 - controller.h, [99](#)
- checksum
 - CommandMessage, [6](#)
 - messaging.h, [48](#)
 - TemperatureMessage, [22](#)
- CircularBuffer/circularbuffer.c, [23](#)
- CircularBuffer/circularbuffer.h, [23](#)
- CircularBuffer_t, [5](#)
 - bufferEnd, [5](#)
 - bufferStart, [5](#)
 - circularbuffer.h, [24](#)
 - head, [5](#)
 - itemSize, [5](#)
 - numItems, [5](#)
 - size, [5](#)
 - tail, [5](#)
- circularbuffer.h
 - BUFFER_EMPTY, [25](#)
 - BUFFER_FULL, [25](#)
 - BUFFER_NOT_EMPTY, [25](#)
 - BUFFER_NOT_FULL, [25](#)
 - BufferState, [25](#)
 - BufferState_e, [24](#)
 - CBufferAdd, [25](#)
 - CBufferAddItems, [25](#)
 - CBufferDestruct, [25](#)
 - CBufferInit, [25](#)
 - CBufferRemove, [26](#)
 - CircularBuffer_t, [24](#)
 - IsBufferEmpty, [26](#)
 - IsBufferFull, [26](#)
- cmd
 - CommandMessage, [6](#)
 - messaging.h, [48](#)
- Color_t
 - led.h, [63](#)
- command
 - nRF24L01_SPIMessage_t, [20](#)
- CommandMessage, [6](#)
 - checksum, [6](#)
 - cmd, [6](#)
 - data, [6](#)
- CommandMessage_t
 - messaging.h, [46](#)
- commands
 - README.txt, [93](#)
- Commands_e
 - messaging.h, [46](#)
- controller.c
 - Controller_ChangeDisplay, [95](#)
 - Controller_ChangeState, [95](#)
 - Controller_Init, [96](#)
 - Controller_SendTempData, [96](#)
 - Controller_SetCurrentTemp, [96](#)
 - Controller_SetDesiredTemp, [96](#)
 - Controller_SetTempRange, [96](#)
 - Controller_StateMachine, [97](#)
 - currentTemp, [97](#)
 - desiredTemp, [97](#)
 - power, [97](#)
 - state, [97](#)
 - tempRange, [97](#)
- controller.h
 - CONVERT_C_TO_F, [99](#)
 - CONVERT_F_TO_C, [99](#)
 - changeDesiredTemp, [99](#)
 - changeTempRange, [99](#)
 - Controller_ChangeDisplay, [99](#)
 - Controller_ChangeState, [99](#)
 - Controller_Init, [99](#)
 - Controller_SendTempData, [100](#)
 - Controller_SetCurrentTemp, [100](#)
 - Controller_SetDesiredTemp, [100](#)
 - Controller_SetTempRange, [100](#)

- Controller_StateMachine, [100](#)
- ControllerState_e, [99](#)
- MAX_DISPLAY_VAL, [99](#)
- noChange, [99](#)
- printSettings, [99](#)
- RELAY_OFF, [99](#)
- RELAY_ON, [99](#)
- RELAY_PIN, [99](#)
- RELAY, [99](#)
- Controller_ChangeDisplay
 - controller.c, [95](#)
 - controller.h, [99](#)
- Controller_ChangeState
 - controller.c, [95](#)
 - controller.h, [99](#)
- Controller_Init
 - controller.c, [96](#)
 - controller.h, [99](#)
- Controller_SendTempData
 - controller.c, [96](#)
 - controller.h, [100](#)
- Controller_SetCurrentTemp
 - controller.c, [96](#)
 - controller.h, [100](#)
- Controller_SetDesiredTemp
 - controller.c, [96](#)
 - controller.h, [100](#)
- Controller_SetTempRange
 - controller.c, [96](#)
 - controller.h, [100](#)
- Controller_StateMachine
 - controller.c, [97](#)
 - controller.h, [100](#)
- ControllerState_e
 - controller.h, [99](#)
- ConvertRawTemperatureData
 - ds18b20.c, [57](#)
 - ds18b20.h, [60](#)
- CR
 - uart.h, [91](#)
- cr
 - messaging.h, [48](#)
 - TemperatureMessage, [22](#)
- crco
 - nRF24L01_CONFIG_t, [8](#)
- currentDesired
 - messaging.h, [48](#)
 - TemperatureMessage, [22](#)
- currentRange
 - messaging.h, [48](#)
 - TemperatureMessage, [22](#)
- currentTemp
 - controller.c, [97](#)
 - messaging.h, [48](#)
 - TemperatureMessage, [22](#)
- cyan
 - README.txt, [93](#)
- CycleLEDs
 - led.c, [62](#)
 - led.h, [64](#)
- d
 - README.txt, [93](#)
- DADP0
 - adc.h, [52](#)
- DADP1
 - adc.h, [52](#)
- DADP2
 - adc.h, [52](#)
- DADP3
 - adc.h, [52](#)
- DATA_LINE_PIN
 - singlewirecomms.h, [80](#)
- DATA_LINE
 - singlewirecomms.h, [80](#)
- DEVICE_LOC
 - spi.h, [84](#)
- DISABLED
 - adc.h, [52](#)
- DISPLAY_A_PORTB_PIN
 - sevenssegment.h, [34](#)
- DISPLAY_A_PORTE_PIN
 - sevenssegment.h, [34](#)
- DISPLAY_B_PORTB_PIN
 - sevenssegment.h, [34](#)
- DISPLAY_B_PORTE_PIN
 - sevenssegment.h, [34](#)
- DISPLAY_C_PORTB_PIN
 - sevenssegment.h, [34](#)
- DISPLAY_C_PORTE_PIN
 - sevenssegment.h, [34](#)
- DISPLAY_CLEAR
 - sevenssegment.h, [34](#)
- DISPLAY_D_PORTB_PIN
 - sevenssegment.h, [34](#)
- DISPLAY_D_PORTE_PIN
 - sevenssegment.h, [34](#)
- DISPLAY_E_PORTB_PIN
 - sevenssegment.h, [34](#)
- DISPLAY_E_PORTE_PIN
 - sevenssegment.h, [34](#)
- DISPLAY_F_PORTB_PIN
 - sevenssegment.h, [35](#)
- DISPLAY_F_PORTE_PIN
 - sevenssegment.h, [35](#)
- DISPLAY_G_PORTB_PIN
 - sevenssegment.h, [35](#)
- DISPLAY_G_PORTE_PIN
 - sevenssegment.h, [35](#)
- DISPLAY_H_PORTB_PIN
 - sevenssegment.h, [35](#)
- DISPLAY_H_PORTE_PIN
 - sevenssegment.h, [35](#)
- DISPLAY_PORTB_A
 - sevenssegment.h, [35](#)
- DISPLAY_PORTB_B
 - sevenssegment.h, [35](#)

DISPLAY_PORTB_C
 sevenssegment.h, 35
 DISPLAY_PORTB_D
 sevenssegment.h, 35
 DISPLAY_PORTB_E
 sevenssegment.h, 35
 DISPLAY_PORTB_F
 sevenssegment.h, 35
 DISPLAY_PORTB_G
 sevenssegment.h, 35
 DISPLAY_PORTB_H
 sevenssegment.h, 35
 DISPLAY_PORTE_A
 sevenssegment.h, 35
 DISPLAY_PORTE_B
 sevenssegment.h, 35
 DISPLAY_PORTE_C
 sevenssegment.h, 35
 DISPLAY_PORTE_D
 sevenssegment.h, 35
 DISPLAY_PORTE_E
 sevenssegment.h, 35
 DISPLAY_PORTE_F
 sevenssegment.h, 35
 DISPLAY_PORTE_G
 sevenssegment.h, 35
 DISPLAY_PORTE_H
 sevenssegment.h, 35
 DISPLAY_SET
 sevenssegment.h, 35
 DMA_BCR_SIZE_MASK
 dma.h, 55
 DMA_RXBUFFER_SIZE
 uart.h, 91
 DMA_TXBUFFER_SIZE
 uart.h, 91
 DMACH_UART0RX
 uart.h, 91
 DMACH_UART0TX
 uart.h, 91
 DMAErrors
 dma.h, 56
 DMAErrors_e
 dma.h, 55
 DMANoError
 dma.h, 56
 DMANot16bitTransferSize
 dma.h, 56
 DMANot32bitTransferSize
 dma.h, 56
 DS8B20_ROMCode, 6
 __attribute__, 6
 romBytes, 6
 DS8B20_Scratchpad, 7
 __attribute__, 7
 scratchpadBytes, 7
 DYNPD
 nRF24L01.h, 72
 data
 CommandMessage, 6
 messaging.h, 48
 nRF24L01_SPIMessage_t, 20
 TemperatureData, 22
 data.c
 BigToLittle, 27
 DumpMemory, 27
 LittleToBig, 28
 MyAtoi, 28
 MyFtoa, 28
 Myltoa, 28
 data.h
 ASCIILETTERBASE, 30
 ASCIINUMBASE, 30
 BYTE0_MASK, 30
 BYTE0_SHIFT, 30
 BYTE1_MASK, 30
 BYTE1_SHIFT, 30
 BYTE2_MASK, 30
 BYTE2_SHIFT, 30
 BYTE3_MASK, 30
 BYTE3_SHIFT, 30
 BigToLittle, 30
 DumpMemory, 30
 LittleToBig, 30
 MyAtoi, 31
 MyFtoa, 31
 Myltoa, 31
 Data/data.c, 26
 Data/data.h, 29
 DataPerformanceTesting
 performancetesting.h, 103
 DataType_t
 uartlogging.h, 38
 DecodeCommandMessage
 messaging.c, 44
 messaging.h, 47
 desiredTemp
 controller.c, 97
 diags.c
 ParseDiag, 53
 diags.h
 ParseDiag, 54
 Display/sevenssegment.c, 31
 Display/sevenssegment.h, 33
 Display_ClearAll
 sevenssegment.c, 32
 sevenssegment.h, 36
 Display_Errors
 sevenssegment.h, 36
 Display_NoError
 sevenssegment.h, 36
 Display_NotSingleDigit
 sevenssegment.h, 36
 Display_Ones
 sevenssegment.h, 36
 Display_Port

- sevenssegment.h, 37
- Display_Tens
 - sevenssegment.h, 36
- Display_Value
 - sevenssegment.h, 37
- Display_place
 - sevenssegment.h, 36
- divisor128
 - spl.h, 85
- divisor16
 - spl.h, 85
- divisor2
 - spl.h, 85
- divisor256
 - spl.h, 85
- divisor32
 - spl.h, 85
- divisor4
 - spl.h, 85
- divisor512
 - spl.h, 85
- divisor64
 - spl.h, 85
- divisor8
 - spl.h, 85
- dma.h
 - _128Bytes, 55
 - _128kBytes, 56
 - _16Bytes, 55
 - _16bit, 56
 - _16kBytes, 56
 - _1kBytes, 55
 - _256Bytes, 55
 - _256kBytes, 56
 - _2kBytes, 56
 - _32Bytes, 55
 - _32bit, 56
 - _32kBytes, 56
 - _4kBytes, 56
 - _512Bytes, 55
 - _64Bytes, 55
 - _64kBytes, 56
 - _8bit, 56
 - _8kBytes, 56
 - BufferDisabled, 55
 - BufferSize, 55
 - BufferSize_e, 55
 - DMA_BCR_SIZE_MASK, 55
 - DMAErrors, 56
 - DMAErrors_e, 55
 - DMANoError, 56
 - DMANot16bitTransferSize, 56
 - DMANot32bitTransferSize, 56
 - InitDMA, 56
 - MASK_32BIT, 55
 - MemSet32bit, 56
 - MemSet8bit, 56
 - NO_DMA, 55
 - Reserved, 56
 - StartTransfer16bitMoves, 56
 - StartTransfer32bitMoves, 56
 - StartTransfer8bitMoves, 56
 - TransferSize, 56
 - TransferSize_e, 55
- dmaComplete
 - memory.c, 41
- double_e
 - uartlogging.h, 38
- dpIP0
 - nRF24L01_DYNPD_t, 9
- dpIP1
 - nRF24L01_DYNPD_t, 9
- dpIP2
 - nRF24L01_DYNPD_t, 9
- dpIP3
 - nRF24L01_DYNPD_t, 9
- dpIP4
 - nRF24L01_DYNPD_t, 9
- dpIP5
 - nRF24L01_DYNPD_t, 9
- ds18b20.c
 - ConvertRawTemperatureData, 57
 - ReadTemp, 57
 - TransactionStepOne, 58
 - TransactionStepTwo, 58
- ds18b20.h
 - ALARM_SEARCH, 60
 - CONVERT_T, 60
 - COPY_SCRATCHPAD, 60
 - ConvertRawTemperatureData, 60
 - MATCH_ROM, 60
 - READ_POWER_SUPPLY, 60
 - READ_ROM, 60
 - READ_SCRATCHPAD, 60
 - RECALL_E2, 60
 - ROM_BYTES, 60
 - ReadTemp, 60
 - SCRATCHPAD_BYTES, 60
 - SEARCH_ROM, 60
 - SERIAL_NUM_BYTES, 60
 - SKIP_ROM, 60
 - TransactionStepOne, 61
 - TransactionStepTwo, 61
 - WAIT_TIME_LONG, 60
 - WAIT_TIME_SHORT, 60
 - WRITE_SCRATCHPAD, 60
- DumpMemory
 - data.c, 27
 - data.h, 30
- EN_AA
 - nRF24L01.h, 72
- EN_CRC_MASK
 - nRF24L01.h, 70
- EN_RXADDR
 - nRF24L01.h, 72
- ENABLE_MESSAGING

- messaging.h, 46
- enAckPay
 - nRF24L01_FEATURE_t, 12
- enCRC
 - nRF24L01_CONFIG_t, 8
- enDpl
 - nRF24L01_FEATURE_t, 12
- enDynAck
 - nRF24L01_FEATURE_t, 12
- enaaP0
 - nRF24L01_ENAA_t, 11
- enaaP1
 - nRF24L01_ENAA_t, 11
- enaaP2
 - nRF24L01_ENAA_t, 11
- enaaP3
 - nRF24L01_ENAA_t, 11
- enaaP4
 - nRF24L01_ENAA_t, 11
- enaaP5
 - nRF24L01_ENAA_t, 11
- erxP0
 - nRF24L01_EN_RXADDR_t, 10
- erxP1
 - nRF24L01_EN_RXADDR_t, 10
- erxP2
 - nRF24L01_EN_RXADDR_t, 10
- erxP3
 - nRF24L01_EN_RXADDR_t, 10
- erxP4
 - nRF24L01_EN_RXADDR_t, 10
- erxP5
 - nRF24L01_EN_RXADDR_t, 10
- externalTriggerPinInput
 - timers.h, 87
- FEATURE
 - nRF24L01.h, 72
- FIFO_STATUS
 - nRF24L01.h, 72
- FLUSH_RX
 - nRF24L01.h, 70
- FLUSH_TX
 - nRF24L01.h, 70
- follows
 - README.txt, 93
- GREEN_CHANNEL
 - led.h, 63
- GREEN_PIN
 - led.h, 63
- GREEN_TPM
 - led.h, 63
- GREEN
 - led.h, 63
- GetElapsedTime
 - timers.c, 86
 - timers.h, 88
- GetTime
 - timers.c, 86
 - timers.h, 88
- green
 - README.txt, 93
- head
 - CircularBuffer_t, 5
- i
 - README.txt, 93
- includeall.h
 - NO_DMA, 39
 - UART_LOGGING, 39
- InitDMA
 - dma.h, 56
- InitDisplay
 - sevenssegment.c, 32
 - sevenssegment.h, 36
- int_e
 - uartlogging.h, 38
- IsBufferEmpty
 - circularbuffer.h, 26
- IsBufferFull
 - circularbuffer.h, 26
- itemSize
 - CircularBuffer_t, 5
- LEDSetup
 - led.h, 64
- LOG0
 - uartlogging.h, 38
- LOG1
 - uartlogging.h, 38
- LPTMRtrigger
 - timers.h, 88
- led.c
 - CycleLEDs, 62
- led.h
 - BLUE_CHANNEL, 63
 - BLUE_PIN, 63
 - BLUE_TPM, 63
 - BLUE, 63
 - CYAN, 63
 - Color_t, 63
 - CycleLEDs, 64
 - GREEN_CHANNEL, 63
 - GREEN_PIN, 63
 - GREEN_TPM, 63
 - GREEN, 63
 - LEDSetup, 64
 - NONE, 63
 - OFF, 63
 - PURPLE, 63
 - RED_CHANNEL, 63
 - RED_PIN, 63
 - RED_TPM, 63
 - RED, 63
 - SwitchLEDs, 64
 - WHITE, 63

- YELLOW, 63
- LF
 - uart.h, 91
- If
 - messaging.h, 48
 - TemperatureMessage, 22
- lib
 - README.txt, 94
- link
 - README.txt, 94
- LittleToBig
 - data.c, 28
 - data.h, 30
- InaHCURR
 - nRF24L01_RF_SETUP_t, 15
- Logging/uartlogging.c, 37
- Logging/uartlogging.h, 37
- MAIN_HEADER
 - memorytesting.h, 102
- MASK_32BIT
 - dma.h, 55
- MASK_MAX_RT_MASK
 - nRF24L01.h, 70
- MASK_RX_DR_MASK
 - nRF24L01.h, 70
- MASK_TX_DS_MASK
 - nRF24L01.h, 70
- MATCH_ROM
 - ds18b20.h, 60
- MAX_BYTES
 - singlewirecomms.h, 80
- MAX_DISPLAY_VAL
 - controller.h, 99
- MAX_LENGTH
 - messaging.h, 46
- MAX_MODULUS
 - timers.h, 87
- MAX_PRESCALER
 - timers.h, 87
- MAX_VALUE
 - adc.h, 51
- MEM_MOVE_SIZE
 - memorytesting.h, 102
- MEMORY_TEST_CASES
 - performancetesting.h, 103
- MODEMDEVICE
 - uart.h, 91
- MODE
 - spi.h, 84
- MUX_PIN_ANALOG
 - adc.h, 51
- MY_MEM_ZERO_SIZE
 - memorytesting.h, 102
- MY_REVERSE_INPUTS
 - memorytesting.h, 102
- MY_STR_LEN_INPUTS
 - memorytesting.h, 102
- main
 - main.c, 39
- main.c
 - main, 39
 - SPI_RXBuffer, 39
 - SPI_TXBuffer, 39
 - UART0_RXBuffer, 39
 - UART0_TXBuffer, 39
 - UART1_RXBuffer, 39
 - UART1_TXBuffer, 39
- Main/includeall.h, 38
- Main/main.c, 39
- maskMaxRt
 - nRF24L01_CONFIG_t, 8
- maskRxDr
 - nRF24L01_CONFIG_t, 8
- maskTxDs
 - nRF24L01_CONFIG_t, 8
- maxRT
 - nRF24L01_STATUS_t, 21
- MemSet32bit
 - dma.h, 56
- MemSet8bit
 - dma.h, 56
- memory.c
 - dmaComplete, 41
 - MyMemMove, 40
 - MyMemSet, 40
 - MyReverse, 40
 - MyStrLen, 41
- memory.h
 - MyMemMove, 42
 - MyMemSet, 42
 - MyReverse, 42
 - MyStrLen, 42
- Memory/memory.c, 39
- Memory/memory.h, 41
- MemoryPerformanceTesting
 - performancetesting.h, 103
- MemoryTesting
 - memorytesting.h, 102
- memorytesting.h
 - MAIN_HEADER, 102
 - MEM_MOVE_SIZE, 102
 - MY_MEM_ZERO_SIZE, 102
 - MY_REVERSE_INPUTS, 102
 - MY_STR_LEN_INPUTS, 102
 - MemoryTesting, 102
 - MyMemMoveUnitTest, 102
 - MyMemSetUnitTest, 102
 - MyReverseUnitTest, 102
 - MyStrLenUnitTest, 102
 - TESTING_DMA_CH, 102
 - TESTSTRING, 102
- messaging.c
 - BuildCommandMessage, 43
 - CalculateCommandChecksum, 43
 - CalculateTemperatureChecksum, 44
 - DecodeCommandMessage, 44

- UART1_TXBuffer, 44
- messaging.h
 - __attribute__, 47, 48
 - BuildCommandMessage, 47
 - COMMAND_MSG_BYTES, 46
 - CalculateCommandChecksum, 47
 - CalculateTemperatureChecksum, 47
 - changeColor, 46
 - changePWM, 46
 - checksum, 48
 - cmd, 48
 - CommandMessage_t, 46
 - Commands_e, 46
 - cr, 48
 - currentDesired, 48
 - currentRange, 48
 - currentTemp, 48
 - data, 48
 - DecodeCommandMessage, 47
 - ENABLE_MESSAGING, 46
 - If, 48
 - MAX_LENGTH, 46
 - MessagingErrors_e, 46
 - NOPcommand, 46
 - NUM_COMMANDS, 46
 - noError, 47
 - powerOn, 48
 - readTempData, 46
 - rxBufferFull, 47
 - setDesired, 46
 - setDisplay, 46
 - setRange, 46
 - setTemp, 46
 - TEMP_MSG_BYTES, 46
 - TemperatureMessage_t, 46
 - txBufferFull, 47
- Messaging/messaging.c, 42
- Messaging/messaging.h, 44
- MessagingErrors_e
 - messaging.h, 46
- Modules/adc.c, 48
- Modules/adc.h, 50
- Modules/diags.c, 53
- Modules/diags.h, 54
- Modules/dma.c, 54
- Modules/dma.h, 54
- Modules/ds18b20.c, 56
- Modules/ds18b20.h, 58
- Modules/io.h, 61
- Modules/led.c, 61
- Modules/led.h, 62
- Modules/nRF24L01.c, 64
- Modules/nRF24L01.h, 67
- Modules/pushbutton.c, 75
- Modules/pushbutton.h, 76
- Modules/singlewirecomms.c, 77
- Modules/singlewirecomms.h, 79
- Modules/spi.c, 81
- Modules/spi.h, 82
- Modules/timers.c, 86
- Modules/timers.h, 86
- Modules/uart.c, 88
- Modules/uart.h, 90
- msg
 - TemperatureData, 22
- MyAtoi
 - data.c, 28
 - data.h, 31
- MyFtoa
 - data.c, 28
 - data.h, 31
- MyItoa
 - data.c, 28
 - data.h, 31
- MyMemMove
 - memory.c, 40
 - memory.h, 42
- MyMemMoveUnitTest
 - memorytesting.h, 102
- MyMemSet
 - memory.c, 40
 - memory.h, 42
- MyMemSetUnitTest
 - memorytesting.h, 102
- MyReverse
 - memory.c, 40
 - memory.h, 42
- MyReverseUnitTest
 - memorytesting.h, 102
- MyStrLen
 - memory.c, 41
 - memory.h, 42
- MyStrLenUnitTest
 - memorytesting.h, 102
- NO_DMA
 - dma.h, 55
 - includeall.h, 39
- NONE
 - led.h, 63
- NOPcommand
 - messaging.h, 46
- NOP
 - nRF24L01.h, 70
- nRF24L01.c
 - nRF24L01_Activate, 65
 - nRF24L01_PowerDown, 65
 - nRF24L01_ReadReg, 65
 - nRF24L01_SendData, 66
 - nRF24L01_SendNOP, 66
 - nRF24L01_SetRXMode, 66
 - nRF24L01_SetTXMode, 66
 - nRF24L01_SetupChannel, 66
 - nRF24L01_StandbyMode, 67
 - nRF24L01_WriteReg, 67
 - readRegComplete, 67
 - SPI_RXBuffer, 67

- SPI_TXBuffer, [67](#)
- nRF24L01.h
 - ACTIVATE_KEY, [70](#)
 - ACTIVATE, [70](#)
 - ADR_1Mbps, [71](#)
 - ADR_2Mbps, [71](#)
 - ADR_e, [71](#)
 - CHANNEL_0, [72](#)
 - CHANNEL_1, [72](#)
 - CHANNEL_2, [72](#)
 - CHANNEL_3, [72](#)
 - CHANNEL_4, [72](#)
 - CHANNEL_5, [72](#)
 - CHANNEL_6, [72](#)
 - CHANNEL_7, [72](#)
 - CHANNEL_8, [72](#)
 - CHANNEL_9, [72](#)
 - CHANNEL_e, [71](#)
 - CONFIG, [72](#)
 - CRCO_MASK, [70](#)
 - CD, [72](#)
 - DYNPD, [72](#)
 - EN_AA, [72](#)
 - EN_CRC_MASK, [70](#)
 - EN_RXADDR, [72](#)
 - FEATURE, [72](#)
 - FIFO_STATUS, [72](#)
 - FLUSH_RX, [70](#)
 - FLUSH_TX, [70](#)
 - MASK_MAX_RT_MASK, [70](#)
 - MASK_RX_DR_MASK, [70](#)
 - MASK_TX_DS_MASK, [70](#)
 - NOP, [70](#)
 - nRF24L01_0_CE_PIN, [70](#)
 - nRF24L01_0_CE, [70](#)
 - nRF24L01_0_IRQ_PIN, [71](#)
 - nRF24L01_0_IRQ, [70](#)
 - nRF24L01_1_CE_PIN, [71](#)
 - nRF24L01_1_CE, [71](#)
 - nRF24L01_1_IRQ_PIN, [71](#)
 - nRF24L01_1_IRQ, [71](#)
 - nRF24L01_Activate, [73](#)
 - nRF24L01_CHANNEL_FREQ, [71](#)
 - nRF24L01_DATA_RATE, [71](#)
 - nRF24L01_LNA_GAIN, [71](#)
 - nRF24L01_PA_CONTROL, [71](#)
 - nRF24L01_PowerDown, [73](#)
 - nRF24L01_ReadReg, [73](#)
 - nRF24L01_Registers_e, [72](#)
 - nRF24L01_SendData, [73](#)
 - nRF24L01_SendNOP, [74](#)
 - nRF24L01_SetRXMode, [74](#)
 - nRF24L01_SetTXMode, [74](#)
 - nRF24L01_SetupChannel, [74](#)
 - nRF24L01_StandbyMode, [74](#)
 - nRF24L01_WriteReg, [75](#)
 - OBSERVE_TX, [72](#)
 - PRIM_RX_MASK, [71](#)
 - PWR_0dBm, [73](#)
 - PWR_UP_MASK, [71](#)
 - PWR_e, [72](#)
 - PWR_neg12dBm, [73](#)
 - PWR_neg18dBm, [73](#)
 - PWR_neg6dBm, [73](#)
 - R_RX_PAYLOAD, [71](#)
 - R_RX_PL_WID, [71](#)
 - READ_REG, [71](#)
 - REUSE_TX_PL, [71](#)
 - RF_CH, [72](#)
 - RF_SETUP, [72](#)
 - RX_ADDR_P0, [72](#)
 - RX_ADDR_P1, [72](#)
 - RX_ADDR_P2, [72](#)
 - RX_ADDR_P3, [72](#)
 - RX_ADDR_P4, [72](#)
 - RX_ADDR_P5, [72](#)
 - RX_PW_P0, [72](#)
 - RX_PW_P1, [72](#)
 - RX_PW_P2, [72](#)
 - RX_PW_P3, [72](#)
 - RX_PW_P4, [72](#)
 - RX_PW_P5, [72](#)
 - SETUP_AW, [72](#)
 - SETUP_RETR, [72](#)
 - STATUS, [72](#)
 - TX_ADDR, [72](#)
 - W_ACK_PAYLOAD, [71](#)
 - W_TX_PAYLOAD_NO_ACK, [71](#)
 - W_TX_PAYLOAD, [71](#)
 - WRITE_REG, [71](#)
- nRF24L01_0_CE_PIN
 - nRF24L01.h, [70](#)
- nRF24L01_0_CE
 - nRF24L01.h, [70](#)
- nRF24L01_0_IRQ_PIN
 - nRF24L01.h, [71](#)
- nRF24L01_0_IRQ
 - nRF24L01.h, [70](#)
- nRF24L01_1_CE_PIN
 - nRF24L01.h, [71](#)
- nRF24L01_1_CE
 - nRF24L01.h, [71](#)
- nRF24L01_1_IRQ_PIN
 - nRF24L01.h, [71](#)
- nRF24L01_1_IRQ
 - nRF24L01.h, [71](#)
- nRF24L01_Activate
 - nRF24L01.c, [65](#)
 - nRF24L01.h, [73](#)
- nRF24L01_CHANNEL_FREQ
 - nRF24L01.h, [71](#)
- nRF24L01_CONFIG_t, [7](#)
 - B, [8](#)
 - b, [8](#)
 - crco, [8](#)
 - enCRC, [8](#)

- maskMaxRt, [8](#)
- maskRxDr, [8](#)
- maskTxDs, [8](#)
- primRx, [8](#)
- pwrUp, [8](#)
- reserved, [8](#)
- nRF24L01_DATA_RATE
 - nRF24L01.h, [71](#)
- nRF24L01_DYNPD_t, [8](#)
 - B, [9](#)
 - b, [9](#)
 - dplP0, [9](#)
 - dplP1, [9](#)
 - dplP2, [9](#)
 - dplP3, [9](#)
 - dplP4, [9](#)
 - dplP5, [9](#)
 - reserved, [9](#)
- nRF24L01_EN_RXADDR_t, [9](#)
 - B, [10](#)
 - b, [10](#)
 - erxP0, [10](#)
 - erxP1, [10](#)
 - erxP2, [10](#)
 - erxP3, [10](#)
 - erxP4, [10](#)
 - erxP5, [10](#)
 - reserved, [10](#)
- nRF24L01_ENAA_t, [10](#)
 - B, [11](#)
 - b, [11](#)
 - enaaP0, [11](#)
 - enaaP1, [11](#)
 - enaaP2, [11](#)
 - enaaP3, [11](#)
 - enaaP4, [11](#)
 - enaaP5, [11](#)
 - reserved, [11](#)
- nRF24L01_FEATURE_t, [11](#)
 - B, [12](#)
 - b, [12](#)
 - enAckPay, [12](#)
 - enDpl, [12](#)
 - enDynAck, [12](#)
 - reserved, [12](#)
- nRF24L01_FIFO_STATUS_t, [12](#)
 - B, [13](#)
 - b, [13](#)
 - reserved0, [13](#)
 - reserved1, [13](#)
 - rxEmpty, [13](#)
 - rxFull, [13](#)
 - txEmpty, [13](#)
 - txFull, [13](#)
 - txReuse, [13](#)
- nRF24L01_LNA_GAIN
 - nRF24L01.h, [71](#)
- nRF24L01_OBSERVE_TX_t, [13](#)
 - arcCNT, [13](#)
 - B, [13](#)
 - b, [13](#)
 - plosCNT, [13](#)
- nRF24L01_PA_CONTROL
 - nRF24L01.h, [71](#)
- nRF24L01_PowerDown
 - nRF24L01.c, [65](#)
 - nRF24L01.h, [73](#)
- nRF24L01_RF_CH_t, [14](#)
 - B, [14](#)
 - b, [14](#)
 - reserved, [14](#)
 - rfch, [14](#)
- nRF24L01_RF_SETUP_t, [14](#)
 - B, [15](#)
 - b, [15](#)
 - lnaHCURR, [15](#)
 - pllLock, [15](#)
 - reserved, [15](#)
 - rfDR, [15](#)
 - rfPWR, [15](#)
- nRF24L01_RX_PW_P0_t, [15](#)
 - B, [15](#)
 - b, [15](#)
 - reserved, [15](#)
 - rxPwP0, [15](#)
- nRF24L01_RX_PW_P1_t, [16](#)
 - B, [16](#)
 - b, [16](#)
 - reserved, [16](#)
 - rxPwP1, [16](#)
- nRF24L01_RX_PW_P2_t, [16](#)
 - B, [17](#)
 - b, [17](#)
 - reserved, [17](#)
 - rxPwP2, [17](#)
- nRF24L01_RX_PW_P3_t, [17](#)
 - B, [17](#)
 - b, [17](#)
 - reserved, [17](#)
 - rxPwP3, [17](#)
- nRF24L01_RX_PW_P4_t, [17](#)
 - B, [18](#)
 - b, [18](#)
 - reserved, [18](#)
 - rxPwP4, [18](#)
- nRF24L01_RX_PW_P5_t, [18](#)
 - B, [18](#)
 - b, [18](#)
 - reserved, [18](#)
 - rxPwP5, [18](#)
- nRF24L01_ReadReg
 - nRF24L01.c, [65](#)
 - nRF24L01.h, [73](#)
- nRF24L01_Registers_e
 - nRF24L01.h, [72](#)
- nRF24L01_SETUP_AW_t, [19](#)

- aw, [19](#)
- B, [19](#)
- b, [19](#)
- reserved, [19](#)
- nRF24L01_SETUP_RETR_t, [19](#)
 - arc, [20](#)
 - ard, [20](#)
 - B, [20](#)
 - b, [20](#)
- nRF24L01_SPIMessage_t, [20](#)
 - bytesToSend, [20](#)
 - command, [20](#)
 - data, [20](#)
 - spiCh, [20](#)
- nRF24L01_STATUS_t, [20](#)
 - B, [21](#)
 - b, [21](#)
 - maxRT, [21](#)
 - reserved, [21](#)
 - rxDR, [21](#)
 - rxPno, [21](#)
 - txDS, [21](#)
 - txFull, [21](#)
- nRF24L01_SendData
 - nRF24L01.c, [66](#)
 - nRF24L01.h, [73](#)
- nRF24L01_SendNOP
 - nRF24L01.c, [66](#)
 - nRF24L01.h, [74](#)
- nRF24L01_SetRXMode
 - nRF24L01.c, [66](#)
 - nRF24L01.h, [74](#)
- nRF24L01_SetTXMode
 - nRF24L01.c, [66](#)
 - nRF24L01.h, [74](#)
- nRF24L01_SetupChannel
 - nRF24L01.c, [66](#)
 - nRF24L01.h, [74](#)
- nRF24L01_StandbyMode
 - nRF24L01.c, [67](#)
 - nRF24L01.h, [74](#)
- nRF24L01_WriteReg
 - nRF24L01.c, [67](#)
 - nRF24L01.h, [75](#)
- NS_PER_SEC
 - timers.h, [87](#)
- NS_PER_US
 - timers.h, [87](#)
- NUM_BUTTONS
 - pushbutton.h, [76](#)
- NUM_COMMANDS
 - messaging.h, [46](#)
- NUM_DIGITS
 - sevensegment.h, [36](#)
- NUM_DISPLAYS
 - sevensegment.h, [36](#)
- NUM_SEGMENTS
 - sevensegment.h, [36](#)
- noChange
 - controller.h, [99](#)
- noError
 - messaging.h, [47](#)
- numItems
 - CircularBuffer_t, [5](#)
- OBSERVE_TX
 - nRF24L01.h, [72](#)
- OFF
 - led.h, [63](#)
- OSR
 - uart.h, [91](#)
- PORTA_IRQHandler
 - pushbutton.c, [75](#)
- POSIX_SOURCE
 - uart.h, [91](#)
- PRIM_RX_MASK
 - nRF24L01.h, [71](#)
- PRINTING_BUFFER_SIZE
 - performancetesting.h, [103](#)
- PRINTING_TEST_CASES
 - performancetesting.h, [103](#)
- PROFILER_CH
 - timers.h, [87](#)
- PROFILER_PERIOD_IN_NS
 - timers.h, [87](#)
- PROFILER_TPM
 - timers.h, [87](#)
- PULL_LOW
 - singlewirecomms.h, [80](#)
- PURPLE
 - led.h, [63](#)
- PWR_0dBm
 - nRF24L01.h, [73](#)
- PWR_UP_MASK
 - nRF24L01.h, [71](#)
- PWR_e
 - nRF24L01.h, [72](#)
- PWR_neg12dBm
 - nRF24L01.h, [73](#)
- PWR_neg18dBm
 - nRF24L01.h, [73](#)
- PWR_neg6dBm
 - nRF24L01.h, [73](#)
- ParseDiag
 - diags.c, [53](#)
 - diags.h, [54](#)
- parseDiag
 - uart.c, [90](#)
 - uart.h, [92](#)
- PerformanceTesting
 - performancetesting.h, [103](#)
- performancetesting.h
 - DataPerformanceTesting, [103](#)
 - MEMORY_TEST_CASES, [103](#)
 - MemoryPerformanceTesting, [103](#)
 - PRINTING_BUFFER_SIZE, [103](#)

PRINTING_TEST_CASES, 103
 PerformanceTesting, 103
 PrintingPerformanceTesting, 103
 pitTrigger0
 timers.h, 88
 pitTrigger1
 timers.h, 88
 pllLock
 nRF24L01_RF_SETUP_t, 15
 plosCNT
 nRF24L01_OBSERVE_TX_t, 13
 power
 controller.c, 97
 powerOn
 messaging.h, 48
 TemperatureMessage, 22
 prescaler1
 spi.h, 85
 prescaler2
 spi.h, 85
 prescaler3
 spi.h, 85
 prescaler4
 spi.h, 85
 prescaler5
 spi.h, 85
 prescaler6
 spi.h, 85
 prescaler7
 spi.h, 85
 prescaler8
 spi.h, 85
 primRx
 nRF24L01_CONFIG_t, 8
 printSettings
 controller.h, 99
 PrintingPerformanceTesting
 performancetesting.h, 103
 purple
 README.txt, 94
 pushbutton.c
 Button_Init, 75
 PORTA_IRQHandler, 75
 pushbutton.h
 BUTTON0, 76
 BUTTON0_PIN, 76
 Button_Init, 76
 NUM_BUTTONS, 76
 PutChar
 uart.c, 89
 uart.h, 91
 pwrUp
 nRF24L01_CONFIG_t, 8

 R_RX_PAYLOAD
 nRF24L01.h, 71
 R_RX_PL_WID
 nRF24L01.h, 71
 READ_LINE
 singlewirecomms.h, 80
 READ_POWER_SUPPLY
 ds18b20.h, 60
 READ_REG
 nRF24L01.h, 71
 READ_ROM
 ds18b20.h, 60
 READ_SCRATCHPAD
 ds18b20.h, 60
 README.txt, 92
 assemble, 93
 blue, 93
 commands, 93
 cyan, 93
 d, 93
 follows, 93
 green, 93
 i, 93
 lib, 94
 link, 94
 purple, 94
 S, 94
 upload, 94
 white, 94
 x, 94
 yellow, 94
 RECALL_E2
 ds18b20.h, 60
 RED_CHANNEL
 led.h, 63
 RED_PIN
 led.h, 63
 RED_TPM
 led.h, 63
 RELAY_OFF
 controller.h, 99
 RELAY_ON
 controller.h, 99
 RELAY_PIN
 controller.h, 99
 RELAY
 controller.h, 99
 RELEASE_LINE
 singlewirecomms.h, 80
 RESERVED0
 adc.h, 52
 RESERVED1
 adc.h, 52
 RESERVED2
 adc.h, 52
 REUSE_TX_PL
 nRF24L01.h, 71
 RED
 led.h, 63
 RF_CH
 nRF24L01.h, 72
 RF_SETUP
 nRF24L01.h, 72

ROM_BYTES
 ds18b20.h, 60
 RX_ADDR_P0
 nRF24L01.h, 72
 RX_ADDR_P1
 nRF24L01.h, 72
 RX_ADDR_P2
 nRF24L01.h, 72
 RX_ADDR_P3
 nRF24L01.h, 72
 RX_ADDR_P4
 nRF24L01.h, 72
 RX_ADDR_P5
 nRF24L01.h, 72
 RX_PW_P0
 nRF24L01.h, 72
 RX_PW_P1
 nRF24L01.h, 72
 RX_PW_P2
 nRF24L01.h, 72
 RX_PW_P3
 nRF24L01.h, 72
 RX_PW_P4
 nRF24L01.h, 72
 RX_PW_P5
 nRF24L01.h, 72
 RXBUFFER_SIZE
 uart.h, 91
 readRegComplete
 nRF24L01.c, 67
 ReadTemp
 ds18b20.c, 57
 ds18b20.h, 60
 readTempData
 messaging.h, 46
 Reserved
 dma.h, 56
 reserved
 nRF24L01_CONFIG_t, 8
 nRF24L01_DYNPD_t, 9
 nRF24L01_EN_RXADDR_t, 10
 nRF24L01_ENAA_t, 11
 nRF24L01_FEATURE_t, 12
 nRF24L01_RF_CH_t, 14
 nRF24L01_RF_SETUP_t, 15
 nRF24L01_RX_PW_P0_t, 15
 nRF24L01_RX_PW_P1_t, 16
 nRF24L01_RX_PW_P2_t, 17
 nRF24L01_RX_PW_P3_t, 17
 nRF24L01_RX_PW_P4_t, 18
 nRF24L01_RX_PW_P5_t, 18
 nRF24L01_SETUP_AW_t, 19
 nRF24L01_STATUS_t, 21
 reserved0
 nRF24L01_FIFO_STATUS_t, 13
 timers.h, 87
 reserved1
 nRF24L01_FIFO_STATUS_t, 13
 reserved2
 timers.h, 88
 reserved3
 timers.h, 88
 reserved4
 timers.h, 88
 reserved5
 timers.h, 88
 reseved1
 timers.h, 88
 rfDR
 nRF24L01_RF_SETUP_t, 15
 rfPWR
 nRF24L01_RF_SETUP_t, 15
 rfch
 nRF24L01_RF_CH_t, 14
 romBytes
 DS8B20_ROMCode, 6
 rtcAlarm
 timers.h, 88
 rtcSeconds
 timers.h, 88
 rxBufferFull
 messaging.h, 47
 rxDR
 nRF24L01_STATUS_t, 21
 rxEmpty
 nRF24L01_FIFO_STATUS_t, 13
 rxFull
 nRF24L01_FIFO_STATUS_t, 13
 rxPno
 nRF24L01_STATUS_t, 21
 rxPwP0
 nRF24L01_RX_PW_P0_t, 15
 rxPwP1
 nRF24L01_RX_PW_P1_t, 16
 rxPwP2
 nRF24L01_RX_PW_P2_t, 17
 rxPwP3
 nRF24L01_RX_PW_P3_t, 17
 rxPwP4
 nRF24L01_RX_PW_P4_t, 18
 rxPwP5
 nRF24L01_RX_PW_P5_t, 18
 S
 README.txt, 94
 SCRATCPAD_BYTES
 ds18b20.h, 60
 SEARCH_ROM
 ds18b20.h, 60
 SERIAL_NUM_BYTES
 ds18b20.h, 60
 SETUP_AW
 nRF24L01.h, 72
 SETUP_RETR
 nRF24L01.h, 72
 SKIP_ROM
 ds18b20.h, 60

SPI0_CS_PIN
 spi.h, 84
 SPI0_CS
 spi.h, 84
 SPI0_MISO
 spi.h, 84
 SPI0_MOSI
 spi.h, 84
 SPI0_SCK
 spi.h, 84
 SPI1_CS_PIN
 spi.h, 84
 SPI1_CS
 spi.h, 84
 SPI1_MISO
 spi.h, 84
 SPI1_MOSI
 spi.h, 84
 SPI1_SCK
 spi.h, 84
 SPI_0_5Mbps_BRD
 spi.h, 84
 SPI_0_5Mbps_PRESCALER
 spi.h, 84
 SPI_1Mbps_BRD
 spi.h, 84
 SPI_1Mbps_PRESCALER
 spi.h, 84
 SPI_2Mbps_BRD
 spi.h, 84
 SPI_2Mbps_PRESCALER
 spi.h, 84
 SPI_BRDvisor
 spi.h, 85
 SPI_BRPrescaler
 spi.h, 85
 SPI_CHANNELS
 spi.h, 84
 SPI_Init
 spi.c, 82
 spi.h, 85
 SPI_RXBUFFER_SIZE
 spi.h, 84
 SPI_RXBuffer
 main.c, 39
 nRF24L01.c, 67
 spi.c, 82
 SPI_TXBUFFER_SIZE
 spi.h, 85
 SPI_TXBuffer
 main.c, 39
 nRF24L01.c, 67
 spi.c, 82
 SPI_TransmitData
 spi.c, 82
 spi.h, 85
 STATUS
 nRF24L01.h, 72
 SWC_Init
 singlewirecomms.c, 77
 singlewirecomms.h, 80
 SWC_ReadData
 singlewirecomms.c, 77
 singlewirecomms.h, 80
 SWC_ReadStatusAndWait
 singlewirecomms.c, 78
 singlewirecomms.h, 80
 SWC_ResetAndPresencePulses
 singlewirecomms.c, 78
 singlewirecomms.h, 81
 SWC_SendByte
 singlewirecomms.c, 78
 singlewirecomms.h, 81
 SWITCH_TO_RX
 singlewirecomms.h, 80
 SWITCH_TO_TX
 singlewirecomms.h, 80
 scratchpadBytes
 DS8B20_Scratchpad, 7
 setDesired
 messaging.h, 46
 setDisplay
 messaging.h, 46
 setRange
 messaging.h, 46
 setTemp
 messaging.h, 46
 sevensegment.c
 Display_ClearAll, 32
 InitDisplay, 32
 UpdateDisplay, 32
 sevensegment.h
 DISPLAY_A_PORTB_PIN, 34
 DISPLAY_A_PORTE_PIN, 34
 DISPLAY_B_PORTB_PIN, 34
 DISPLAY_B_PORTE_PIN, 34
 DISPLAY_C_PORTB_PIN, 34
 DISPLAY_C_PORTE_PIN, 34
 DISPLAY_CLEAR, 34
 DISPLAY_D_PORTB_PIN, 34
 DISPLAY_D_PORTE_PIN, 34
 DISPLAY_E_PORTB_PIN, 34
 DISPLAY_E_PORTE_PIN, 34
 DISPLAY_F_PORTB_PIN, 35
 DISPLAY_F_PORTE_PIN, 35
 DISPLAY_G_PORTB_PIN, 35
 DISPLAY_G_PORTE_PIN, 35
 DISPLAY_H_PORTB_PIN, 35
 DISPLAY_H_PORTE_PIN, 35
 DISPLAY_PORTB_A, 35
 DISPLAY_PORTB_B, 35
 DISPLAY_PORTB_C, 35
 DISPLAY_PORTB_D, 35
 DISPLAY_PORTB_E, 35
 DISPLAY_PORTB_F, 35
 DISPLAY_PORTB_G, 35

- DISPLAY_PORTB_H, [35](#)
- DISPLAY_PORTE_A, [35](#)
- DISPLAY_PORTE_B, [35](#)
- DISPLAY_PORTE_C, [35](#)
- DISPLAY_PORTE_D, [35](#)
- DISPLAY_PORTE_E, [35](#)
- DISPLAY_PORTE_F, [35](#)
- DISPLAY_PORTE_G, [35](#)
- DISPLAY_PORTE_H, [35](#)
- DISPLAY_SET, [35](#)
- Display_ClearAll, [36](#)
- Display_Errors, [36](#)
- Display_NoError, [36](#)
- Display_NotSingleDigit, [36](#)
- Display_Ones, [36](#)
- Display_Port, [37](#)
- Display_Tens, [36](#)
- Display_Value, [37](#)
- Display_place, [36](#)
- InitDisplay, [36](#)
- NUM_DIGITS, [36](#)
- NUM_DISPLAYS, [36](#)
- NUM_SEGMENTS, [36](#)
- UpdateDisplay, [36](#)
- singlewirecomms.c
 - SWC_Init, [77](#)
 - SWC_ReadData, [77](#)
 - SWC_ReadStatusAndWait, [78](#)
 - SWC_ResetAndPresencePulses, [78](#)
 - SWC_SendByte, [78](#)
- singlewirecomms.h
 - DATA_LINE_PIN, [80](#)
 - DATA_LINE, [80](#)
 - MAX_BYTES, [80](#)
 - PULL_LOW, [80](#)
 - READ_LINE, [80](#)
 - RELEASE_LINE, [80](#)
 - SWC_Init, [80](#)
 - SWC_ReadData, [80](#)
 - SWC_ReadStatusAndWait, [80](#)
 - SWC_ResetAndPresencePulses, [81](#)
 - SWC_SendByte, [81](#)
 - SWITCH_TO_RX, [80](#)
 - SWITCH_TO_TX, [80](#)
 - WRITE_0, [80](#)
 - WRITE_1, [80](#)
- size
 - CircularBuffer_t, [5](#)
- spi.c
 - SPI_Init, [82](#)
 - SPI_RXBuffer, [82](#)
 - SPI_TXBuffer, [82](#)
 - SPI_TransmitData, [82](#)
 - spiDeviceLocation, [82](#)
- spi.h
 - ARRAY_SIZE, [84](#)
 - BBB_SPI_SPEED, [84](#)
 - BPW, [84](#)
 - DEVICE_LOC, [84](#)
 - divisor128, [85](#)
 - divisor16, [85](#)
 - divisor2, [85](#)
 - divisor256, [85](#)
 - divisor32, [85](#)
 - divisor4, [85](#)
 - divisor512, [85](#)
 - divisor64, [85](#)
 - divisor8, [85](#)
 - MODE, [84](#)
 - prescaler1, [85](#)
 - prescaler2, [85](#)
 - prescaler3, [85](#)
 - prescaler4, [85](#)
 - prescaler5, [85](#)
 - prescaler6, [85](#)
 - prescaler7, [85](#)
 - prescaler8, [85](#)
 - SPI0_CS_PIN, [84](#)
 - SPI0_CS, [84](#)
 - SPI0_MISO, [84](#)
 - SPI0_MOSI, [84](#)
 - SPI0_SCK, [84](#)
 - SPI1_CS_PIN, [84](#)
 - SPI1_CS, [84](#)
 - SPI1_MISO, [84](#)
 - SPI1_MOSI, [84](#)
 - SPI1_SCK, [84](#)
 - SPI_0_5Mbps_BRD, [84](#)
 - SPI_0_5Mbps_PRESCALER, [84](#)
 - SPI_1Mbps_BRD, [84](#)
 - SPI_1Mbps_PRESCALER, [84](#)
 - SPI_2Mbps_BRD, [84](#)
 - SPI_2Mbps_PRESCALER, [84](#)
 - SPI_BRDivisor, [85](#)
 - SPI_BRPrescaler, [85](#)
 - SPI_CHANNELS, [84](#)
 - SPI_Init, [85](#)
 - SPI_RXBUFFER_SIZE, [84](#)
 - SPI_TXBUFFER_SIZE, [85](#)
 - SPI_TransmitData, [85](#)
- spiCh
 - nRF24L01_SPIMessage_t, [20](#)
- spiDeviceLocation
 - spi.c, [82](#)
- StartTransfer16bitMoves
 - dma.h, [56](#)
- StartTransfer32bitMoves
 - dma.h, [56](#)
- StartTransfer8bitMoves
 - dma.h, [56](#)
- state
 - controller.c, [97](#)
- SwitchLEDs
 - led.h, [64](#)
- TEMP_MSG_BYTES
 - messaging.h, [46](#)

TEMP_SENSOR
 adc.h, 52
 TESTING_DMA_CH
 memorytesting.h, 102
 TESTSTRING
 memorytesting.h, 102
 TPM_TriggerOptions
 timers.h, 87
 TX_ADDR
 nRF24L01.h, 72
 TXBUFFER_SIZE
 uart.h, 91
 tail
 CircularBuffer_t, 5
 tempRange
 controller.c, 97
 TemperatureController/controller.c, 94
 TemperatureController/controller.h, 97
 TemperatureData, 21
 data, 22
 msg, 22
 TemperatureMessage, 22
 checksum, 22
 cr, 22
 currentDesired, 22
 currentRange, 22
 currentTemp, 22
 lf, 22
 powerOn, 22
 TemperatureMessage_t
 messaging.h, 46
 Testing/circularbuffertesting.c, 101
 Testing/circularbuffertesting.h, 101
 Testing/datatesting.c, 101
 Testing/datatesting.h, 101
 Testing/memorytesting.c, 101
 Testing/memorytesting.h, 101
 Testing/performancetesting.c, 102
 Testing/performancetesting.h, 102
 Testing/testing.c, 103
 Testing/testing.h, 103
 timers.c
 GetElapsedTime, 86
 GetTime, 86
 timers.h
 _10US_PER_SEC, 87
 CMP0Output, 87
 COUNTS_PER_US, 87
 externalTriggerPinInput, 87
 GetElapsedTime, 88
 GetTime, 88
 LPTMRtrigger, 88
 MAX_MODULUS, 87
 MAX_PRESCALER, 87
 NS_PER_SEC, 87
 NS_PER_US, 87
 PROFILER_CH, 87
 PROFILER_PERIOD_IN_NS, 87
 PROFILER_TPM, 87
 pitTrigger0, 88
 pitTrigger1, 88
 reserved0, 87
 reserved2, 88
 reserved3, 88
 reserved4, 88
 reserved5, 88
 reseved1, 88
 rtcAlarm, 88
 rtcSeconds, 88
 TPM_TriggerOptions, 87
 tpm0Overflow, 88
 tpm1Overflow, 88
 tpm2Overflow, 88
 UNITS_US, 87
 WAIT_CH, 87
 WAIT_TPM, 87
 tpm0Overflow
 timers.h, 88
 tpm1Overflow
 timers.h, 88
 tpm2Overflow
 timers.h, 88
 TransactionStepOne
 ds18b20.c, 58
 ds18b20.h, 61
 TransactionStepTwo
 ds18b20.c, 58
 ds18b20.h, 61
 TransferSize
 dma.h, 56
 TransferSize_e
 dma.h, 55
 txBufferFull
 messaging.h, 47
 txDS
 nRF24L01_STATUS_t, 21
 txEmpty
 nRF24L01_FIFO_STATUS_t, 13
 txFull
 nRF24L01_FIFO_STATUS_t, 13
 nRF24L01_STATUS_t, 21
 txReuse
 nRF24L01_FIFO_STATUS_t, 13
 UART0_RXBuffer
 main.c, 39
 uart.c, 90
 UART0_TXBuffer
 main.c, 39
 uart.c, 90
 UART1_RXBuffer
 main.c, 39
 uart.c, 90
 UART1_TXBuffer
 main.c, 39
 messaging.c, 44
 uart.c, 90

UART_LOGGING
 includeall.h, 39

UNITS_US
 timers.h, 87

uart.c
 parseDiag, 90
 PutChar, 89
 UART0_RXBuffer, 90
 UART0_TXBuffer, 90
 UART1_RXBuffer, 90
 UART1_TXBuffer, 90
 UartRX, 89
 UartSetup, 89
 UartTX, 89

uart.h
 BAUDRATE, 91
 BONEPATH, 91
 CR, 91
 DMA_RXBUFFER_SIZE, 91
 DMA_TXBUFFER_SIZE, 91
 DMACH_UART0RX, 91
 DMACH_UART0TX, 91
 LF, 91
 MODEMDEVICE, 91
 OSR, 91
 POSIX_SOURCE, 91
 parseDiag, 92
 PutChar, 91
 RXBUFFER_SIZE, 91
 TXBUFFER_SIZE, 91
 UartRX, 91
 UartSetup, 92
 UartTX, 92

UartRX
 uart.c, 89
 uart.h, 91

UartSetup
 uart.c, 89
 uart.h, 92

UartTX
 uart.c, 89
 uart.h, 92

uartlogging.h
 DataType_t, 38
 double_e, 38
 int_e, 38
 LOG0, 38
 LOG1, 38

UpdateDisplay
 sevenssegment.c, 32
 sevenssegment.h, 36

upload
 README.txt, 94

VREFSH
 adc.h, 52

VREFSL
 adc.h, 52

W_ACK_PAYLOAD
 nRF24L01.h, 71

W_TX_PAYLOAD_NO_ACK
 nRF24L01.h, 71

W_TX_PAYLOAD
 nRF24L01.h, 71

WAIT_CH
 timers.h, 87

WAIT_TIME_LONG
 ds18b20.h, 60

WAIT_TIME_SHORT
 ds18b20.h, 60

WAIT_TPM
 timers.h, 87

WHITE
 led.h, 63

WRITE_0
 singlewirecomms.h, 80

WRITE_1
 singlewirecomms.h, 80

WRITE_REG
 nRF24L01.h, 71

WRITE_SCRATCHPAD
 ds18b20.h, 60

white
 README.txt, 94

x
 README.txt, 94

YELLOW
 led.h, 63

yellow
 README.txt, 94