

CM-Bot_API

Erzeugt von Doxygen 1.7.1

Thu Dec 30 2010 16:08:55

Inhaltsverzeichnis

1	Datenstruktur-Verzeichnis	1
1.1	Datenstrukturen	1
2	Datei-Verzeichnis	2
2.1	Auflistung der Dateien	2
3	Datenstruktur-Dokumentation	4
3.1	DT_half_circle Strukturreferenz	4
3.1.1	Ausführliche Beschreibung	4
3.1.2	Dokumentation der Datenelemente	4
3.1.2.1	sqr_r	4
3.2	DT_individuum Strukturreferenz	4
3.2.1	Ausführliche Beschreibung	5
3.2.2	Dokumentation der Datenelemente	5
3.2.2.1	F	5
3.2.2.2	G	5
3.2.2.3	S	5
3.3	DT_leg Strukturreferenz	5
3.3.1	Ausführliche Beschreibung	5
3.3.2	Dokumentation der Datenelemente	5
3.3.2.1	foot	5
3.3.2.2	hip	6
3.3.2.3	knee	6
3.3.2.4	trans	6
3.4	DT_lin_func Strukturreferenz	6
3.4.1	Ausführliche Beschreibung	6

3.4.2	Dokumentation der Datenelemente	6
3.4.2.1	m	6
3.4.2.2	n	6
3.5	DT_point Strukturreferenz	7
3.5.1	Ausführliche Beschreibung	7
3.5.2	Dokumentation der Datenelemente	7
3.5.2.1	x	7
3.5.2.2	y	7
3.5.2.3	z	7
3.6	DT_servo Strukturreferenz	7
3.6.1	Ausführliche Beschreibung	7
3.6.2	Dokumentation der Datenelemente	8
3.6.2.1	act_value	8
3.6.2.2	id	8
3.6.2.3	set_value	8
3.7	DT_transformation Strukturreferenz	8
3.7.1	Ausführliche Beschreibung	8
3.7.2	Dokumentation der Datenelemente	8
3.7.2.1	x	8
3.7.2.2	y	8
3.7.2.3	zRotation	8
3.8	DT_vector Strukturreferenz	9
3.8.1	Ausführliche Beschreibung	9
3.8.2	Dokumentation der Datenelemente	9
3.8.2.1	x	9
3.8.2.2	y	9
3.9	Usart_and_buffer Strukturreferenz	9
3.9.1	Ausführliche Beschreibung	9
3.9.2	Dokumentation der Datenelemente	10
3.9.2.1	buffer	10
3.9.2.2	dreIntLevel	10
3.9.2.3	lastPacketLength	10
3.9.2.4	port	10
3.9.2.5	usart	10

3.10	USART_Buffer Strukturreferenz	10
3.10.1	Dokumentation der Datenelemente	10
3.10.1.1	RX	10
3.10.1.2	RX_Head	10
3.10.1.3	RX_Tail	10
3.10.1.4	TX	10
3.10.1.5	TX_Head	10
3.10.1.6	TX_Tail	10
4	Datei-Dokumentation	11
4.1	clksys_driver.c-Dateireferenz	11
4.1.1	Ausführliche Beschreibung	12
4.1.2	Dokumentation der Funktionen	13
4.1.2.1	CCPWrite	13
4.1.2.2	CLKSYS_AutoCalibration_Enable	13
4.1.2.3	CLKSYS_Configuration_Lock	14
4.1.2.4	CLKSYS_Disable	14
4.1.2.5	CLKSYS_Main_ClockSource_Select	14
4.1.2.6	CLKSYS_PLL_Config	15
4.1.2.7	CLKSYS_Prescalers_Config	15
4.1.2.8	CLKSYS_RTC_ClockSource_Enable	15
4.1.2.9	CLKSYS_XOSC_Config	16
4.1.2.10	CLKSYS_XOSC_FailureDetection_Enable	16
4.2	communication.c-Dateireferenz	16
4.2.1	Ausführliche Beschreibung	18
4.2.2	Makro-Dokumentation	18
4.2.2.1	COM_START_BYTE	18
4.2.3	Dokumentation der Funktionen	18
4.2.3.1	COM_byteArrayToDouble	18
4.2.3.2	COM_doubleToByteArray	19
4.2.3.3	COM_getAngleFromPacket	19
4.2.3.4	COM_getChecksum	19
4.2.3.5	COM_getCpuID	20
4.2.3.6	COM_getPointFromPacket	20

4.2.3.7	COM_getSpeedFromPacket	20
4.2.3.8	COM_isAlive	20
4.2.3.9	COM_isFoot	20
4.2.3.10	COM_isGlobal	21
4.2.3.11	COM_isHip	21
4.2.3.12	COM_isKnee	21
4.2.3.13	COM_isLeftLeg	22
4.2.3.14	COM_isRightLeg	22
4.2.3.15	COM_receive	22
4.2.3.16	COM_requestStatus	23
4.2.3.17	COM_send	23
4.2.3.18	COM_sendACK	23
4.2.3.19	COM_sendAction	24
4.2.3.20	COM_sendAngle	24
4.2.3.21	COM_sendNAK	24
4.2.3.22	COM_sendPoint	24
4.2.3.23	COM_sendPointAndSpeed	25
4.3	dynamixel.c-Dateireferenz	25
4.3.1	Ausführliche Beschreibung	27
4.3.2	Makro-Dokumentation	28
4.3.2.1	ACT	28
4.3.2.2	ALR_SHUTDWN	28
4.3.2.3	BD	28
4.3.2.4	GL_POS	28
4.3.2.5	ID	28
4.3.2.6	LED	28
4.3.2.7	MAX_TMP	28
4.3.2.8	MV_SPEED	28
4.3.2.9	PING	28
4.3.2.10	PRT_POS	28
4.3.2.11	PRT_SPEED	28
4.3.2.12	PRT_TMP	28
4.3.2.13	RD_DATA	28
4.3.2.14	REG_WR	28

4.3.2.15	RESET	28
4.3.2.16	START_BYTE	28
4.3.2.17	STS_RT_LVL	28
4.3.2.18	SYC_WR	28
4.3.2.19	WR_DATA	28
4.3.3	Dokumentation der Funktionen	28
4.3.3.1	DNX_convertAngle	28
4.3.3.2	DNX_correctAngles	29
4.3.3.3	DNX_getAngle	29
4.3.3.4	DNX_getChecksum	29
4.3.3.5	DNX_getConnectedIDs	30
4.3.3.6	DNX_getLed	30
4.3.3.7	DNX_getSpeed	30
4.3.3.8	DNX_receive	30
4.3.3.9	DNX_send	31
4.3.3.10	DNX_sendAction	31
4.3.3.11	DNX_setAngle	31
4.3.3.12	DNX_setAngleAndSpeed	31
4.3.3.13	DNX_setId	32
4.3.3.14	DNX_setLed	32
4.3.3.15	DNX_setSpeed	32
4.4	evolutionaryAlgorithm.c-Dateireferenz	32
4.4.1	Ausführliche Beschreibung	33
4.4.2	Makro-Dokumentation	34
4.4.2.1	X_MAX	34
4.4.2.2	X_MIN	34
4.4.2.3	Y_MAX	34
4.4.2.4	Y_MIN	34
4.4.3	Dokumentation der Funktionen	34
4.4.3.1	bestSelection	34
4.4.3.2	evolutionaryAlgorithm	34
4.4.3.3	fitnessproportionalSelection	34
4.4.3.4	generatePoint	34
4.4.3.5	generatePopulation	34

4.4.3.6	getIsectFromIndividuum	34
4.4.3.7	getPointFromIndividuum	34
4.4.3.8	getRandomNumber	34
4.4.3.9	getScores	34
4.4.3.10	gleichverteilte_reellwertige_mutation	34
4.4.3.11	mutation	34
4.4.3.12	printPopulation	34
4.4.3.13	recombination	34
4.4.3.14	uniformCrossover	34
4.5	evolutionaryHelper.c-Dateireferenz	34
4.5.1	Ausführliche Beschreibung	36
4.5.2	Makro-Dokumentation	36
4.5.2.1	FIRST_VALUE	36
4.5.2.2	NO_VALUE	36
4.5.2.3	SECOND_VALUE	36
4.5.3	Dokumentation der Funktionen	37
4.5.3.1	bubblesort	37
4.5.3.2	f_circ	37
4.5.3.3	f_lin	37
4.5.3.4	getDistance	37
4.5.3.5	getFunctionOfPoints	37
4.5.3.6	getNearerPoint	37
4.5.3.7	initEvoAlg	37
4.5.3.8	initFunctions	37
4.5.3.9	initPoints	38
4.5.3.10	isBetweenPoints	38
4.5.3.11	isectLinCirc	38
4.5.3.12	isectLinFuncs	38
4.5.3.13	isInArea	38
4.5.3.14	isVectorialPoint	38
4.5.3.15	max	38
4.5.3.16	min	38
4.5.3.17	scorePoint	38
4.5.4	Variablen-Dokumentation	38

4.5.4.1	A	38
4.5.4.2	AB	38
4.5.4.3	B	38
4.5.4.4	C	38
4.5.4.5	CD	38
4.5.4.6	CEA	38
4.5.4.7	D	38
4.5.4.8	DFB	38
4.5.4.9	E	38
4.5.4.10	F	38
4.5.4.11	G	38
4.6	evolutionaryWalking.c-Dateireferenz	38
4.6.1	Ausführliche Beschreibung	40
4.6.2	Makro-Dokumentation	41
4.6.2.1	NO_OFFSET	41
4.6.2.2	OFFSET	41
4.6.2.3	TEST_ON	41
4.6.3	Dokumentation der Funktionen	41
4.6.3.1	calculateMovementPoints	41
4.6.3.2	copyPoint	41
4.6.3.3	doStep	41
4.6.3.4	doStepMove	41
4.6.3.5	evolutionaryCalculation	41
4.6.3.6	init_pMpSpMiddle	41
4.6.3.7	initConf	41
4.6.3.8	invertVector	41
4.6.3.9	main	41
4.6.3.10	master	41
4.6.3.11	prepareStepMove	41
4.6.3.12	switchLegs	41
4.6.3.13	TripodGaitMove	41
4.6.3.14	waitForButton3	41
4.6.4	Variablen-Dokumentation	41
4.6.4.1	cpuID	41

4.6.4.2	isectM	41
4.6.4.3	isectS	41
4.6.4.4	leg_l	41
4.6.4.5	leg_r	41
4.6.4.6	MasterActive	41
4.6.4.7	MasterInactive	41
4.6.4.8	midM	41
4.6.4.9	midS	41
4.6.4.10	pM	41
4.6.4.11	pMiddle	41
4.6.4.12	pS	41
4.6.4.13	SlavesActive	41
4.6.4.14	SlavesInactive	41
4.7	include/avr_compiler.h-Dateireferenz	41
4.7.1	Ausführliche Beschreibung	42
4.7.2	Makro-Dokumentation	43
4.7.2.1	AVR_ENTER_CRITICAL_REGION	43
4.7.2.2	AVR_LEAVE_CRITICAL_REGION	43
4.7.2.3	F_CPU	43
4.8	include/clksys_driver.h-Dateireferenz	43
4.8.1	Ausführliche Beschreibung	45
4.8.2	Makro-Dokumentation	46
4.8.2.1	CLKSYS_AutoCalibration_Disable	46
4.8.2.2	CLKSYS_Enable	46
4.8.2.3	CLKSYS_IsReady	46
4.8.2.4	CLKSYS_RTC_ClockSource_Disable	47
4.8.3	Dokumentation der Funktionen	47
4.8.3.1	CCPWrite	47
4.8.3.2	CLKSYS_AutoCalibration_Enable	47
4.8.3.3	CLKSYS_Configuration_Lock	48
4.8.3.4	CLKSYS_Disable	48
4.8.3.5	CLKSYS_Main_ClockSource_Select	48
4.8.3.6	CLKSYS_PLL_Config	48
4.8.3.7	CLKSYS_Prescalers_Config	49

4.8.3.8	CLKSYS_RTC_ClockSource_Enable	49
4.8.3.9	CLKSYS_XOSC_Config	49
4.8.3.10	CLKSYS_XOSC_FailureDetection_Enable	50
4.9	include/communication.h-Dateireferenz	50
4.9.1	Ausführliche Beschreibung	52
4.9.2	Makro-Dokumentation	53
4.9.2.1	COM_ACK	53
4.9.2.2	COM_ACTION	53
4.9.2.3	COM_ANGLE	53
4.9.2.4	COM_BRDCAST_ID	53
4.9.2.5	COM_CONF_FOOT	53
4.9.2.6	COM_CONF_GLOB	53
4.9.2.7	COM_CONF_HIP	53
4.9.2.8	COM_CONF_KNEE	53
4.9.2.9	COM_CONF_LEFT	53
4.9.2.10	COM_CONF_RIGHT	53
4.9.2.11	COM_ERR_ANGLE_LIMIT	53
4.9.2.12	COM_ERR_DEFAULT_ERROR	53
4.9.2.13	COM_ERR_POINT_OUT_OF_BOUNDS	53
4.9.2.14	COM_IS_ALIVE	53
4.9.2.15	COM_MASTER	53
4.9.2.16	COM_NAK	53
4.9.2.17	COM_NOCPUID	53
4.9.2.18	COM_POINT	53
4.9.2.19	COM_SLAVE1B	53
4.9.2.20	COM_SLAVE3F	53
4.9.2.21	COM_SPEED	53
4.9.2.22	COM_STATUS	53
4.9.3	Dokumentation der Funktionen	53
4.9.3.1	COM_byteArrayToDouble	53
4.9.3.2	COM_doubleToByteArray	54
4.9.3.3	COM_getAngleFromPacket	54
4.9.3.4	COM_getCpuID	54
4.9.3.5	COM_getPointFromPacket	54

4.9.3.6	COM_getSpeedFromPacket	55
4.9.3.7	COM_isAlive	55
4.9.3.8	COM_isFoot	55
4.9.3.9	COM_isGlobal	55
4.9.3.10	COM_isHip	56
4.9.3.11	COM_isKnee	56
4.9.3.12	COM_isLeftLeg	56
4.9.3.13	COM_isRightLeg	57
4.9.3.14	COM_receive	57
4.9.3.15	COM_requestStatus	57
4.9.3.16	COM_send	58
4.9.3.17	COM_sendACK	58
4.9.3.18	COM_sendAction	58
4.9.3.19	COM_sendAngle	58
4.9.3.20	COM_sendNAK	59
4.9.3.21	COM_sendPoint	59
4.9.3.22	COM_sendPointAndSpeed	59
4.10	include/datatypes.h-Dateireferenz	60
4.10.1	Ausführliche Beschreibung	61
4.10.2	Makro-Dokumentation	61
4.10.2.1	DT_RESULT_BUFFER_SIZE	61
4.10.3	Dokumentation der benutzerdefinierten Typen	62
4.10.3.1	DT_bool	62
4.10.3.2	DT_byte	62
4.10.3.3	DT_char	62
4.10.3.4	DT_cmd	62
4.10.3.5	DT_double	62
4.10.3.6	DT_int	62
4.10.3.7	DT_size	62
4.10.3.8	DT_type	62
4.11	include/dynamixel.h-Dateireferenz	62
4.11.1	Ausführliche Beschreibung	63
4.11.2	Makro-Dokumentation	63
4.11.2.1	DNX_BRDCAST_ID	63

4.11.3	Dokumentation der Funktionen	63
4.11.3.1	DNX_getAngle	63
4.11.3.2	DNX_getChecksum	64
4.11.3.3	DNX_getConnectedIDs	64
4.11.3.4	DNX_getLed	64
4.11.3.5	DNX_getSpeed	64
4.11.3.6	DNX_receive	65
4.11.3.7	DNX_send	65
4.11.3.8	DNX_sendAction	65
4.11.3.9	DNX_setAngle	66
4.11.3.10	DNX_setAngleAndSpeed	66
4.11.3.11	DNX_setId	66
4.11.3.12	DNX_setLed	66
4.11.3.13	DNX_setSpeed	67
4.12	include/evolutionaryAlgorithm.h-Dateireferenz	67
4.12.1	Ausführliche Beschreibung	67
4.12.2	Dokumentation der Funktionen	67
4.12.2.1	evolutionaryAlgorithm	67
4.12.2.2	getIsectFromIndividuum	67
4.12.2.3	getPointFromIndividuum	67
4.13	include/evolutionaryHelper.h-Dateireferenz	67
4.13.1	Ausführliche Beschreibung	68
4.13.2	Makro-Dokumentation	68
4.13.2.1	Z	68
4.13.3	Dokumentation der Funktionen	68
4.13.3.1	bubblesort	68
4.13.3.2	getDistance	68
4.13.3.3	getFunctionOfPoints	68
4.13.3.4	initEvoAlg	68
4.13.3.5	isInArea	68
4.13.3.6	max	68
4.13.3.7	min	68
4.13.3.8	scorePoint	68
4.14	include/kinematics.h-Dateireferenz	68

4.14.1	Ausführliche Beschreibung	69
4.14.2	Makro-Dokumentation	69
4.14.2.1	KIN_COLUMNS	69
4.14.2.2	KIN_ROWS	69
4.14.3	Dokumentation der Funktionen	69
4.14.3.1	KIN_calcDH	69
4.14.3.2	KIN_calcLocalPoint	70
4.14.3.3	KIN_calcServos	70
4.14.3.4	KIN_makeMovement	70
4.14.3.5	KIN_setTransMat	70
4.15	include/movement.h-Dateireferenz	70
4.15.1	Ausführliche Beschreibung	72
4.15.2	Makro-Dokumentation	72
4.15.2.1	MV_DST_X	72
4.15.2.2	MV_DST_Y	72
4.15.3	Dokumentation der Funktionen	72
4.15.3.1	MV_action	72
4.15.3.2	MV_doInitPosition	72
4.15.3.3	MV_getPntForCpuSide	72
4.15.3.4	MV_masterCheckAlive	73
4.15.3.5	MV_point	73
4.15.3.6	MV_pointAndSpeed	73
4.15.3.7	MV_slave	74
4.15.3.8	MV_slaveAngle	74
4.15.3.9	MV_slavePoint	74
4.15.3.10	MV_slavePointAndSpeed	75
4.15.3.11	MV_slaveStatus	75
4.15.3.12	MV_switchLegs	75
4.16	include/remote.h-Dateireferenz	76
4.16.1	Ausführliche Beschreibung	77
4.16.2	Dokumentation der Funktionen	77
4.16.2.1	RMT_getCommand	77
4.16.2.2	RMT_isButton1Pressed	77
4.16.2.3	RMT_isButton2Pressed	77

4.16.2.4	RMT_isButton3Pressed	77
4.16.2.5	RMT_isButton4Pressed	78
4.16.2.6	RMT_isButton5Pressed	78
4.16.2.7	RMT_isButton6Pressed	78
4.16.2.8	RMT_isDownPressed	79
4.16.2.9	RMT_isLeftPressed	79
4.16.2.10	RMT_isRightPressed	79
4.16.2.11	RMT_isUpPressed	79
4.16.2.12	RMT_NonPressed	80
4.16.2.13	RMT_receive	80
4.17	include/usart_driver.h-Dateireferenz	80
4.17.1	Ausführliche Beschreibung	83
4.17.2	Makro-Dokumentation	84
4.17.2.1	USART_Baudrate_Set	84
4.17.2.2	USART_DreInterruptLevel_Set	85
4.17.2.3	USART_Format_Set	85
4.17.2.4	USART_GetChar	85
4.17.2.5	USART_IsRXComplete	86
4.17.2.6	USART_IsTXDataRegisterEmpty	86
4.17.2.7	USART_PutChar	86
4.17.2.8	USART_RX_BUFFER_MASK	86
4.17.2.9	USART_RX_BUFFER_SIZE	86
4.17.2.10	USART_Rx_Disable	86
4.17.2.11	USART_Rx_Enable	87
4.17.2.12	USART_RxdInterruptLevel_Set	87
4.17.2.13	USART_SetMode	87
4.17.2.14	USART_TX_BUFFER_MASK	88
4.17.2.15	USART_TX_BUFFER_SIZE	88
4.17.2.16	USART_Tx_Disable	88
4.17.2.17	USART_Tx_Enable	88
4.17.2.18	USART_TxdInterruptLevel_Set	88
4.17.3	Dokumentation der benutzerdefinierten Typen	88
4.17.3.1	USART_Buffer_t	88
4.17.3.2	USART_data_t	88

4.17.4	Dokumentation der Funktionen	89
4.17.4.1	USART_DataRegEmpty	89
4.17.4.2	USART_InterruptDriver_DreInterruptLevel_Set . .	89
4.17.4.3	USART_InterruptDriver_Initialize	89
4.17.4.4	USART_NineBits_GetChar	90
4.17.4.5	USART_NineBits_PutChar	90
4.17.4.6	USART_RXBuffer_checkPointerDiff	90
4.17.4.7	USART_RXBuffer_GetByte	90
4.17.4.8	USART_RXBufferData_Available	91
4.17.4.9	USART_RXComplete	91
4.17.4.10	USART_TXBuffer_FreeSpace	91
4.17.4.11	USART_TXBuffer_PutByte	92
4.18	include/utils.h-Dateireferenz	92
4.18.1	Ausführliche Beschreibung	93
4.18.2	Makro-Dokumentation	93
4.18.2.1	DEBUG	93
4.18.2.2	DEBUG_BYTE	93
4.18.2.3	DEBUG_ON	93
4.18.2.4	UTL_DEG	93
4.18.2.5	UTL_RAD	93
4.18.3	Dokumentation der Funktionen	93
4.18.3.1	UTL_byteToHexChar	93
4.18.3.2	UTL_getDegree	94
4.18.3.3	UTL_getPointOfDH	94
4.18.3.4	UTL_getRadiant	94
4.18.3.5	UTL_printDebug	94
4.18.3.6	UTL_printDebugByte	95
4.18.3.7	UTL_printLeg	95
4.18.3.8	UTL_printMatrix	95
4.18.3.9	UTL_printPoint	95
4.18.3.10	UTL_wait	95
4.19	include/xmega.h-Dateireferenz	96
4.19.1	Ausführliche Beschreibung	97
4.19.2	Makro-Dokumentation	98

4.19.2.1	SWITCH_PRESSED	98
4.19.2.2	SWITCH_RELEASED	98
4.19.2.3	SWITCHMASK	98
4.19.2.4	SWITCHPORT	98
4.19.2.5	XM_LED_MASK	98
4.19.2.6	XM_LED_OFF	98
4.19.2.7	XM_LED_ON	98
4.19.2.8	XM_LED_TGL	98
4.19.2.9	XM_OE_MASK	98
4.19.2.10	XM_PORT_COM1	98
4.19.2.11	XM_PORT_COM3	98
4.19.2.12	XM_PORT_DEBUG	98
4.19.2.13	XM_PORT_LED	98
4.19.2.14	XM_PORT_REMOTE	98
4.19.2.15	XM_PORT_SERVO_L	98
4.19.2.16	XM_PORT_SERVO_R	98
4.19.2.17	XM_USART_COM1	98
4.19.2.18	XM_USART_COM3	98
4.19.2.19	XM_USART_DEBUG	98
4.19.2.20	XM_USART_FAILURE	98
4.19.2.21	XM_USART_REMOTE	99
4.19.2.22	XM_USART_SERVO_L	99
4.19.2.23	XM_USART_SERVO_R	99
4.19.3	Dokumentation der Funktionen	99
4.19.3.1	XM_init_com	99
4.19.3.2	XM_init_cpu	99
4.19.3.3	XM_init_dnx	99
4.19.3.4	XM_init_remote	99
4.19.3.5	XM_USART_send	99
4.19.4	Variablen-Dokumentation	100
4.19.4.1	XM_com_data1	100
4.19.4.2	XM_com_data3	100
4.19.4.3	XM_debug_data	100
4.19.4.4	XM_remote_data	100

4.19.4.5	XM_servo_data_L	100
4.19.4.6	XM_servo_data_R	100
4.20	kinematics.c-Dateireferenz	100
4.20.1	Ausführliche Beschreibung	101
4.20.2	Makro-Dokumentation	101
4.20.2.1	DIST_DZ	101
4.20.2.2	DIST_FE	101
4.20.2.3	DIST_HK	101
4.20.2.4	DIST_KF	102
4.20.3	Dokumentation der Funktionen	102
4.20.3.1	KIN_calcDH	102
4.20.3.2	KIN_calcLocalPoint	102
4.20.3.3	KIN_calcServos	102
4.20.3.4	KIN_setTransMat	103
4.21	main.c-Dateireferenz	103
4.21.1	Makro-Dokumentation	103
4.21.1.1	F_CPU	103
4.21.1.2	TEST_OFF	103
4.22	movement.c-Dateireferenz	103
4.22.1	Ausführliche Beschreibung	105
4.22.2	Makro-Dokumentation	105
4.22.2.1	MV_DST_X	105
4.22.3	Dokumentation der Funktionen	105
4.22.3.1	MV_action	105
4.22.3.2	MV_doInitPosition	105
4.22.3.3	MV_getPntForCpuSide	105
4.22.3.4	MV_masterCheckAlive	106
4.22.3.5	MV_point	106
4.22.3.6	MV_pointAndSpeed	106
4.22.3.7	MV_slave	106
4.22.3.8	MV_slaveAngle	107
4.22.3.9	MV_slavePoint	107
4.22.3.10	MV_slavePointAndSpeed	107
4.22.3.11	MV_slaveStatus	108

4.22.3.12	MV_switchLegs	108
4.23	movement4Points.c-Dateireferenz	108
4.23.1	Ausführliche Beschreibung	108
4.23.2	Makro-Dokumentation	109
4.23.2.1	TEST_OFF	109
4.24	movementMultiPoints.c-Dateireferenz	109
4.24.1	Ausführliche Beschreibung	109
4.24.2	Makro-Dokumentation	109
4.24.2.1	TEST_OFF	109
4.25	remote.c-Dateireferenz	109
4.25.1	Ausführliche Beschreibung	110
4.25.2	Makro-Dokumentation	111
4.25.2.1	B_1	111
4.25.2.2	B_2	111
4.25.2.3	B_3	111
4.25.2.4	B_4	111
4.25.2.5	B_5	111
4.25.2.6	B_6	111
4.25.2.7	B_D	111
4.25.2.8	B_L	111
4.25.2.9	B_NON_PRESSED	111
4.25.2.10	B_R	111
4.25.2.11	B_U	111
4.25.3	Dokumentation der Funktionen	111
4.25.3.1	RMT_getCommand	111
4.25.3.2	RMT_isButton1Pressed	111
4.25.3.3	RMT_isButton2Pressed	112
4.25.3.4	RMT_isButton3Pressed	112
4.25.3.5	RMT_isButton4Pressed	112
4.25.3.6	RMT_isButton5Pressed	112
4.25.3.7	RMT_isButton6Pressed	113
4.25.3.8	RMT_isDownPressed	113
4.25.3.9	RMT_isLeftPressed	113
4.25.3.10	RMT_isRightPressed	114

4.25.3.11 RMT_isUpPressed	114
4.25.3.12 RMT_NonPressed	114
4.25.3.13 RMT_receive	114
4.26 testArithmetic.c-Dateireferenz	115
4.26.1 Makro-Dokumentation	115
4.26.1.1 TEST_OFF	115
4.27 testCom.c-Dateireferenz	115
4.27.1 Ausführliche Beschreibung	115
4.27.2 Makro-Dokumentation	115
4.27.2.1 TEST_OFF	115
4.28 testCom2.c-Dateireferenz	115
4.28.1 Makro-Dokumentation	116
4.28.1.1 TEST_OFF	116
4.29 testCom3.c-Dateireferenz	116
4.29.1 Makro-Dokumentation	116
4.29.1.1 TEST_OFF	116
4.30 testDnx.c-Dateireferenz	116
4.30.1 Ausführliche Beschreibung	116
4.30.2 Makro-Dokumentation	116
4.30.2.1 TEST_OFF	116
4.31 testEvolutionaryDistanceWalking.c-Dateireferenz	116
4.31.1 Makro-Dokumentation	117
4.31.1.1 TEST_OFF	117
4.32 testKin.c-Dateireferenz	117
4.32.1 Ausführliche Beschreibung	117
4.32.2 Makro-Dokumentation	117
4.32.2.1 TEST_OFF	117
4.33 testKin2.c-Dateireferenz	117
4.33.1 Makro-Dokumentation	117
4.33.1.1 TEST_OFF	117
4.34 testMovement.c-Dateireferenz	117
4.34.1 Makro-Dokumentation	118
4.34.1.1 TEST_OFF	118
4.35 testRemote.c-Dateireferenz	118

4.35.1	Makro-Dokumentation	118
4.35.1.1	TEST_OFF	118
4.36	testRingBuffer.c-Dateireferenz	118
4.36.1	Ausführliche Beschreibung	118
4.36.2	Makro-Dokumentation	118
4.36.2.1	TEST_OFF	118
4.37	testSpeed.c-Dateireferenz	118
4.37.1	Ausführliche Beschreibung	118
4.37.2	Makro-Dokumentation	119
4.37.2.1	TEST_OFF	119
4.38	usart_driver.c-Dateireferenz	119
4.38.1	Ausführliche Beschreibung	120
4.38.2	Dokumentation der Funktionen	121
4.38.2.1	USART_DataRegEmpty	121
4.38.2.2	USART_InterruptDriver_DreInterruptLevel_Set . .	121
4.38.2.3	USART_InterruptDriver_Initialize	122
4.38.2.4	USART_NineBits_GetChar	122
4.38.2.5	USART_NineBits_PutChar	122
4.38.2.6	USART_RXBuffer_checkPointerDiff	122
4.38.2.7	USART_RXBuffer_GetByte	123
4.38.2.8	USART_RXBufferData_Available	123
4.38.2.9	USART_RXComplete	123
4.38.2.10	USART_TXBuffer_FreeSpace	124
4.38.2.11	USART_TXBuffer_PutByte	124
4.39	utils.c-Dateireferenz	124
4.39.1	Ausführliche Beschreibung	125
4.39.2	Makro-Dokumentation	125
4.39.2.1	USART_ON	125
4.39.3	Dokumentation der Funktionen	126
4.39.3.1	UTL_byteToHexChar	126
4.39.3.2	UTL_getDegree	126
4.39.3.3	UTL_getPointOfDH	126
4.39.3.4	UTL_getRadiant	126
4.39.3.5	UTL_printDebug	127

4.39.3.6	UTL_printDebugByte	127
4.39.3.7	UTL_printLeg	127
4.39.3.8	UTL_printMatrix	127
4.39.3.9	UTL_printPoint	128
4.39.3.10	UTL_wait	128
4.40	xmega.c-Dateireferenz	128
4.40.1	Ausführliche Beschreibung	130
4.40.2	Dokumentation der Funktionen	130
4.40.2.1	ISR	130
4.40.2.2	ISR	130
4.40.2.3	ISR	130
4.40.2.4	ISR	130
4.40.2.5	ISR	130
4.40.2.6	ISR	130
4.40.2.7	ISR	130
4.40.2.8	ISR	130
4.40.2.9	ISR	130
4.40.2.10	ISR	131
4.40.2.11	ISR	131
4.40.2.12	ISR	131
4.40.2.13	ISR	131
4.40.2.14	XM_init_com	131
4.40.2.15	XM_init_cpu	131
4.40.2.16	XM_init_dnx	131
4.40.2.17	XM_init_remote	131
4.40.2.18	XM_USART_send	131

Kapitel 1

Datenstruktur-Verzeichnis

1.1 Datenstrukturen

Hier folgt die Aufzählung aller Datenstrukturen mit einer Kurzbeschreibung:

DT_half_circle (Datenstruktur zur Speicherung einer Kreisfunktion: $y = \sqrt{r^2 - x^2}$)	4
DT_individuum (Datenstruktur für ein Individuum des Evolutionären Algorithmus zur Startpunktfindung)	4
DT_leg (Datenstruktur zur Speicherung von Servodaten bezüglich eines kompletten Beines)	5
DT_lin_func (Datenstruktur zur Speicherung einer linearen Funktion: $y = mx + n$)	6
DT_point (Datenstruktur zur Speicherung kartesischer Koordinaten)	7
DT_servo (Datenstruktur zur Speicherung von ID, Soll- und Ist-Wert eines Servos)	7
DT_transformation (Struktur zur vereinfachten Koordinatentransformation) .	8
DT_vector (Datenstruktur zur Speicherung eines Vektors)	9
Usart_and_buffer (Struct used when interrupt driven driver is used)	9
USART_Buffer	10

Kapitel 2

Datei-Verzeichnis

2.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

clksys_driver.c (XMEGA Clock System driver source file)	11
communication.c (Methoden zur Kommunikation der CPUs)	16
dynamixel.c (Methoden zur Steuerung der Dynamixel AX-12)	25
evolutionaryAlgorithm.c (Evolutionärer Algorithmus zur Startpunktfindung)	32
evolutionaryHelper.c (Hilfsfunktion für Evolutionären Algorithmus)	34
evolutionaryWalking.c (Testprogramm für Evolutionären Algorithmus zur Startpunktfindung)	38
kinematics.c (Lösungsmethoden der Kinematik)	100
main.c	103
movement.c (Stellt Grundfunktionen für einen Laufalgorithmus bereit) . . .	103
movement4Points.c (Algorithmus fuer das Vorwaertslaufen ueber 4 Punkte)	108
movementMultiPoints.c (Algorithmus fuer das Vorwaertslaufen ueber 4 Punkte)	109
remote.c (Methoden zur Steuerung durch den RC-100 Remote Controller) .	109
testArithmetic.c	115
testCom.c (Testprogramm für Kommunikation der CPUs)	115
testCom2.c	115
testCom3.c	116
testDnx.c (Testprogramm für Ansteuerung der Servos)	116
testEvolutionaryDistanceWalking.c	116
testKin.c (Testprogramm für die Kinematik)	117
testKin2.c	117
testMovement.c	117
testRemote.c	118
testRingBuffer.c (Testprogramm für einen Ringbuffer)	118
testSpeed.c (Testprogramm für Speed-Änderung der Servos)	118
usart_driver.c (XMEGA USART driver source file)	119
utils.c (Verschiedene Hilfsmethoden)	124

xmega.c (Spezifische Funktionen für den Mikrocontroller ATXmega128A1)	128
include/ avr_compiler.h (This file implements some macros that makes the IAR C-compiler and avr-gcc work with the same code base for the AVR architecture)	41
include/ clksys_driver.h (XMEGA Clock System driver header file)	43
include/ communication.h (Methoden zur Kommunikation der CPUs)	50
include/ datatypes.h (Abstrahiert Datentypen)	60
include/ dynamixel.h (Methoden zur Steuerung der Dynamixel AX-12)	62
include/ evolutionaryAlgorithm.h (Evolutionärer Algorithmus zur Startpunktfindung)	67
include/ evolutionaryHelper.h (Hilfsfunktion für Evolutionären Algorithmus)	67
include/ kinematics.h (Lösungsmethoden der Kinematik)	68
include/ movement.h (Stellt Grundfunktionen für einen Laufalgorithmus bereit)	70
include/ remote.h (Methoden zur Steuerung durch den RC-100 Remote Controller)	76
include/ usart_driver.h (XMEGA USART driver header file)	80
include/ utils.h (Verschiedene Hilfsmethoden)	92
include/ xmega.h (Spezifische Funktionen für den Mikrocontroller ATXmega128A1)	96

Kapitel 3

Datenstruktur-Dokumentation

3.1 DT_half_circle Strukturreferenz

Datenstruktur zur Speicherung einer Kreisfunktion: $y = \sqrt{r^2 - x^2}$.

```
#include <datatypes.h>
```

Datenfelder

- **DT_double** `sqr_r`

3.1.1 Ausführliche Beschreibung

Datenstruktur zur Speicherung einer Kreisfunktion: $y = \sqrt{r^2 - x^2}$.

3.1.2 Dokumentation der Datenelemente

3.1.2.1 DT_double DT_half_circle::sqr_r

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- `include/datatypes.h`

3.2 DT_individuum Strukturreferenz

Datenstruktur für ein Individuum des Evolutionären Algorithmus zur Startpunktfindung.

```
#include <datatypes.h>
```

Datenfelder

- DT_point G
- DT_double F
- DT_point S

3.2.1 Ausführliche Beschreibung

Datenstruktur für ein Individuum des Evolutionären Algorithmus zur Startpunktfindung.

3.2.2 Dokumentation der Datenelemente

3.2.2.1 DT_double DT_individuum::F

3.2.2.2 DT_point DT_individuum::G

3.2.2.3 DT_point DT_individuum::S

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- include/datatypes.h

3.3 DT_leg Strukturreferenz

Datenstruktur zur Speicherung von Servodaten bezüglich eines kompletten Beines.

```
#include <datatypes.h>
```

Datenfelder

- DT_servo hip
- DT_servo knee
- DT_servo foot
- DT_transformation trans

3.3.1 Ausführliche Beschreibung

Datenstruktur zur Speicherung von Servodaten bezüglich eines kompletten Beines.

3.3.2 Dokumentation der Datenelemente

3.3.2.1 DT_servo DT_leg::foot

Fußgelenk.

3.3.2.2 DT_servo DT_leg::hip

HÄftgelenk.

3.3.2.3 DT_servo DT_leg::knee

Kniegelenk.

3.3.2.4 DT_transformation DT_leg::trans

Infos fuer Koordinatentransformation.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- include/datatypes.h

3.4 DT_lin_func Strukturreferenz

Datenstruktur zur Speicherung einer linearen Funktion: $y = mx + n$.

```
#include <datatypes.h>
```

Datenfelder

- DT_double m
- DT_double n

3.4.1 Ausführliche Beschreibung

Datenstruktur zur Speicherung einer linearen Funktion: $y = mx + n$.

3.4.2 Dokumentation der Datenelemente

3.4.2.1 DT_double DT_lin_func::m

3.4.2.2 DT_double DT_lin_func::n

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- include/datatypes.h

3.5 DT_point Strukturreferenz

Datenstruktur zur Speicherung karthesischer Koordinaten.

```
#include <datatypes.h>
```

Datenfelder

- DT_double x
- DT_double y
- DT_double z

3.5.1 Ausführliche Beschreibung

Datenstruktur zur Speicherung karthesischer Koordinaten.

3.5.2 Dokumentation der Datenelemente

3.5.2.1 DT_double DT_point::x

3.5.2.2 DT_double DT_point::y

3.5.2.3 DT_double DT_point::z

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- include/datatypes.h

3.6 DT_servo Strukturreferenz

Datenstruktur zur Speicherung von ID, Soll- und Ist-Wert eines Servos.

```
#include <datatypes.h>
```

Datenfelder

- DT_byte id
- DT_double set_value
- DT_double act_value

3.6.1 Ausführliche Beschreibung

Datenstruktur zur Speicherung von ID, Soll- und Ist-Wert eines Servos.

3.6.2 Dokumentation der Datenelemente

3.6.2.1 DT_double DT_servo::act_value

Ist-Wert.

3.6.2.2 DT_byte DT_servo::id

Servo-ID.

3.6.2.3 DT_double DT_servo::set_value

Soll-Wert.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- include/datatypes.h

3.7 DT_transformation Strukturreferenz

Struktur zur vereinfachten Koordinatentransformation.

```
#include <datatypes.h>
```

Datenfelder

- DT_double x
- DT_double y
- DT_bool zRotation

3.7.1 Ausführliche Beschreibung

Struktur zur vereinfachten Koordinatentransformation.

3.7.2 Dokumentation der Datenelemente

3.7.2.1 DT_double DT_transformation::x

3.7.2.2 DT_double DT_transformation::y

3.7.2.3 DT_bool DT_transformation::zRotation

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- include/datatypes.h

3.8 DT_vector Strukturreferenz

Datenstruktur zur Speicherung eines Vektors.

```
#include <datatypes.h>
```

Datenfelder

- **DT_double x**
- **DT_double y**

3.8.1 Ausführliche Beschreibung

Datenstruktur zur Speicherung eines Vektors.

3.8.2 Dokumentation der Datenelemente

3.8.2.1 DT_double DT_vector::x

3.8.2.2 DT_double DT_vector::y

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- include/datatypes.h

3.9 Usart_and_buffer Strukturreferenz

Struct used when interrupt driven driver is used.

```
#include <usart_driver.h>
```

Datenfelder

- **USART_t * usart**
- **USART_DREINTLVL_t dreIntLevel**
- **USART_Buffer_t buffer**
- **DT_byte lastPacketLength**
- **PORT_t * port**

3.9.1 Ausführliche Beschreibung

Struct used when interrupt driven driver is used. Struct containing pointer to a usart, a buffer and a location to store Data register interrupt level temporary.

3.9.2 Dokumentation der Datenelemente

3.9.2.1 USART_Buffer_t Usart_and_buffer::buffer

3.9.2.2 USART_DREINTLVL_t Usart_and_buffer::dreIntLevel

3.9.2.3 DT_byte Usart_and_buffer::lastPacketLength

3.9.2.4 PORT_t* Usart_and_buffer::port

3.9.2.5 USART_t* Usart_and_buffer::usart

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- include/usart_driver.h

3.10 USART_Buffer Strukturreferenz

```
#include <usart_driver.h>
```

Datenfelder

- volatile uint8_t **RX** [USART_RX_BUFFER_SIZE]
- volatile uint8_t **TX** [USART_TX_BUFFER_SIZE]
- volatile uint8_t **RX_Head**
- volatile uint8_t **RX_Tail**
- volatile uint8_t **TX_Head**
- volatile uint8_t **TX_Tail**

3.10.1 Dokumentation der Datenelemente

3.10.1.1 volatile uint8_t USART_Buffer::RX[USART_RX_BUFFER_SIZE]

3.10.1.2 volatile uint8_t USART_Buffer::RX_Head

3.10.1.3 volatile uint8_t USART_Buffer::RX_Tail

3.10.1.4 volatile uint8_t USART_Buffer::TX[USART_TX_BUFFER_SIZE]

3.10.1.5 volatile uint8_t USART_Buffer::TX_Head

3.10.1.6 volatile uint8_t USART_Buffer::TX_Tail

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- include/usart_driver.h

Kapitel 4

Datei-Dokumentation

4.1 clksys_driver.c-Dateireferenz

XMEGA Clock System driver source file.

```
#include "include/clksys_driver.h"
```

Funktionen

- void **CCPWrite** (volatile uint8_t *address, uint8_t value)
CCP write helper function written in assembly.
- void **CLKSYS_XOSC_Config** (OSC_FRQRANGE_t freqRange, bool low-Power32kHz, OSC_XOSCSEL_t xoscModeSelection)
This function configures the external oscillator.
- void **CLKSYS_PLL_Config** (OSC_PLLSRC_t clockSource, uint8_t factor)
This function configures the internal high-frequency PLL.
- uint8_t **CLKSYS_Disable** (uint8_t oscSel)
This function disables the selected oscillator.
- void **CLKSYS_Prescalers_Config** (CLK_PSADIV_t PSAfactor, CLK_PSBCDIV_t PSBCfactor)
This function changes the prescaler configuration.
- uint8_t **CLKSYS_Main_ClockSource_Select** (CLK_SCLKSEL_t clockSource)
This function selects the main system clock source.
- void **CLKSYS_RTC_ClockSource_Enable** (CLK_RTCSRC_t clockSource)
This function selects a Real-Time Counter clock source.

- void **CLKSYS_AutoCalibration_Enable** (uint8_t clkSource, bool extReference)

This function enables automatic calibration of the selected internal oscillator.

- void **CLKSYS_XOSC_FailureDetection_Enable** (void)

This function enables the External Oscillator Failure Detection (XOSCFD) feature.

- void **CLKSYS_Configuration_Lock** (void)

This function lock the entire clock system configuration.

4.1.1 Ausführliche Beschreibung

XMEGA Clock System driver source file. This file contains the function implementations for the XMEGA Clock System driver.

The driver is not intended for size and/or speed critical code, since most functions are just a few lines of code, and the function call overhead would decrease code performance. The driver is intended for rapid prototyping and documentation purposes for getting started with the XMEGA Clock System.

For size and/or speed critical code, it is recommended to copy the function contents directly into your application instead of making a function call.

Several functions use the following construct: "some_register = ... | (some_parameter ? SOME_BIT_bm : 0) | ..." Although the use of the ternary operator (if ? then : else) is discouraged, in some occasions the operator makes it possible to write pretty clean and neat code. In this driver, the construct is used to set or not set a configuration bit based on a boolean input parameter, such as the "some_parameter" in the example above.

Application note:

AVR1003: Using the XMEGA Clock System

Documentation

For comprehensive code documentation, supported compilers, compiler settings and supported devices see readme.html

Autor

Atmel Corporation: <http://www.atmel.com>
Support email: avr@atmel.com

Revision:

2771

Date:

2009-09-11 11:54:26 +0200 (fr, 11 sep 2009)

Copyright (c) 2008, Atmel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of ATMEL may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.1.2 Dokumentation der Funktionen

4.1.2.1 void CCPWrite (volatile uint8_t * *address*, uint8_t *value*)

CCP write helper function written in assembly.

This function is written in assembly because of the timecritical operation of writing to the registers.

Parameter

address A pointer to the address to write to.

value The value to put in to the register.

4.1.2.2 void CLKSYS_AutoCalibration_Enable (uint8_t *clkSource*, bool *extReference*)

This function enables automatic calibration of the selected internal oscillator.

Either the internal 32kHz RC oscillator or an external 32kHz crystal can be used as a calibration reference. The user must make sure that the selected reference is ready and running.

Parameter

clkSource Clock source to calibrate, either OSC_RC2MCREF_bm or OSC_RC32MCREF_bm.

extReference True if external crystal should be used as reference.

4.1.2.3 void CLKSYS_Configuration_Lock (void)

This function lock the entire clock system configuration.

This will lock the configuration until the next reset, or until the External Oscillator Failure Detections (XOSCFD) feature detects a failure and switches to internal 2MHz RC oscillator.

4.1.2.4 uint8_t CLKSYS_Disable (uint8_t oscSel)

This function disables the selected oscillator.

This function will disable the selected oscillator if possible. If it is currently used as a main system clock source, hardware will disregard the disable attempt, and this function will return zero. If it fails, change to another main system clock source and try again.

Parameter

oscSel Bitmask of selected clock. Can be one of the following OSC_RC2MEN_bm, OSC_RC32MEN_bm, OSC_RC32KEN_bm, OSC_XOSCEN_bm, OSC_PLEN_bm.

Rückgabe

Non-zero if oscillator was disabled successfully.

4.1.2.5 uint8_t CLKSYS_Main_ClockSource_Select (CLK_SCLKSEL_t clockSource)

This function selects the main system clock source.

Hardware will disregard any attempts to select a clock source that is not enabled or not stable. If the change fails, make sure the source is ready and running and try again.

Parameter

clockSource Clock source to use as input for the system clock prescaler block.

Rückgabe

Non-zero if change was successful.

4.1.2.6 void CLKSYS_PLL_Config (OSC_PLLSRC_t *clockSource*, uint8_t *factor*)

This function configures the internal high-frequency PLL.

Configuration of the internal high-frequency PLL to the correct values. It is used to define the input of the PLL and the factor of multiplication of the input clock source.

Zu beachten

Note that the oscillator cannot be used as a main system clock source without being enabled and stable first. Check the ready flag before using the clock. The macro **CLKSYS_IsReady(_oscSel)** (S. 46) can be used to check this.

Parameter

clockSource Reference clock source for the PLL, must be above 0.4MHz.

factor PLL multiplication factor, must be from 1 to 31, inclusive.

4.1.2.7 void CLKSYS_Prescalers_Config (CLK_PSADIV_t *PSAfactor*, CLK_PSBCDIV_t *PSBCfactor*)

This function changes the prescaler configuration.

Change the configuration of the three system clock prescaler is one single operation. The user must make sure that the main CPU clock does not exceed recommended limits.

Parameter

PSAfactor Prescaler A division factor, OFF or 2 to 512 in powers of two.

PSBCfactor Prescaler B and C division factor, in the combination of (1,1), (1,2), (4,1) or (2,2).

4.1.2.8 void CLKSYS_RTC_ClockSource_Enable (CLK_RTCSRC_t *clockSource*)

This function selects a Real-Time Counter clock source.

Selects the clock source for use by the Real-Time Counter (RTC) and enables clock signal routing to the RTC module.

Parameter

clockSource Clock source to use for the RTC.

4.1.2.9 void CLKSYS_XOSC_Config (OSC_FRQRANGE_t *freqRange*, bool *lowPower32kHz*, OSC_XOSCSEL_t *xoscModeSelection*)

This function configures the external oscillator.

Zu beachten

Note that the oscillator cannot be used as a main system clock source without being enabled and stable first. Check the ready flag before using the clock. The macro **CLKSYS_IsReady(_oscSel)** (S. 46) can be used to check this.

Parameter

- freqRange*** Frequency range for high-frequency crystal, does not apply for external clock or 32kHz crystals.
- lowPower32kHz*** True if high-quality watch crystals are used and low-power oscillator is desired.
- xoscModeSelection*** Combined selection of oscillator type (or external clock) and startup times.

4.1.2.10 void CLKSYS_XOSC_FailureDetection_Enable (void)

This function enables the External Oscillator Failure Detection (XOSCFD) feature.

The feature will stay enabled until next reset. Note that the XOSCFD _will_ issue the XOSCF Non-maskable Interrupt (NMI) regardless of any interrupt priorities and settings. Therefore, make sure that a handler is implemented for the XOSCF NMI when you enable it.

4.2 communication.c-Dateireferenz

Methoden zur Kommunikation der CPUs.

```
#include "include/communication.h"
#include "include/utils.h"
#include "include/xmega.h"
```

Makrodefinitionen

- **#define COM_START_BYTE** 0xFF

Funktionen

- **DT_byte COM_getChecksum** (const DT_byte *const packet, DT_size l)
Berechnet die Checksum.

- **DT_byte COM_getCpuID** (const **DT_leg** *const leg_l)
- **DT_byte COM_receive** (**USART_data_t** *const usart_data, **DT_byte** *const dest)
USART-Empfangsmethode für Prozessorkommunikation.
- **DT_byte COM_send** (**DT_byte** *const packet, **DT_size** l, **DT_byte** *const result, **DT_bool** hasResponse)
Versenden von Daten an anderen Controller.
- **DT_size COM_requestStatus** (**DT_byte** cpuID, **DT_byte** param, **DT_byte** *const result)
Ruft den Status eines Controllers ab.
- **void COM_doubleToByteArray** (const **DT_double** value, **DT_byte** *const array)
Konvertiert einen Double-Wert in ein Byte-Array.
- **DT_double COM_byteArrayToDouble** (const **DT_byte** *const array)
Konvertiert ein Byte-Array zu einem Double-Wert.
- **DT_bool COM_isRightLeg** (const **DT_byte** *const result)
Prüft ob Flag für rechtes Bein gesetzt ist.
- **DT_bool COM_isLeftLeg** (const **DT_byte** *const result)
Prüft ob Flag für linkes Bein gesetzt ist.
- **DT_bool COM_isGlobal** (const **DT_byte** *const result)
Prüft ob Flag für globalen Punkt gesetzt ist.
- **DT_bool COM_isHip** (const **DT_byte** *const result)
Prüft ob Flag für Hüftgelenk gesetzt ist.
- **DT_bool COM_isKnee** (const **DT_byte** *const result)
Prüft ob Flag für Kniegelenk gesetzt ist.
- **DT_bool COM_isFoot** (const **DT_byte** *const result)
Prüft ob Flag für Fußgelenk gesetzt ist.
- **DT_bool COM_sendPoint** (**DT_byte** cpuID, const **DT_point** *const point, const **DT_byte** config)
Sendet einen Punkt an einen Controller.
- **DT_bool COM_sendPointAndSpeed** (**DT_byte** cpuID, const **DT_point** *const point, const **DT_double** speed, const **DT_byte** config)
Sendet einen Punkt und Anfahrsgeschwindigkeit an einen Controller.

- **DT_point COM_getPointFromPacket** (const **DT_byte** *const result)
Ermittelt aus einem Packet den Punkt.
- **DT_double COM_getSpeedFromPacket** (const **DT_byte** *const result)
Ermittelt aus einem Packet die Anfahrsgeschwindigkeit.
- **DT_bool COM_sendAngle** (**DT_byte** cpuID, const **DT_double** angle, const **DT_byte** config)
Sendet einen Winkel an einen Controller.
- **DT_double COM_getAngleFromPacket** (const **DT_byte** *const result)
Ermittelt aus einem Packet den Winkel.
- **void COM_sendAction** (**DT_byte** cpuID)
Sendet das ACTION-Kommando an einen Controller.
- **DT_bool COM_isAlive** (**DT_byte** cpuID)
Sendet eine isAlive-Anfrage an einen Controller.
- **void COM_sendACK** (**DT_byte** cpuID)
Sendet ein ACK an einen Controller.
- **void COM_sendNAK** (**DT_byte** cpuID, **DT_byte** errCode)
Sendet ein NAK an einen Controller.

4.2.1 Ausführliche Beschreibung

Methoden zur Kommunikation der CPUs.

4.2.2 Makro-Dokumentation

4.2.2.1 #define COM_START_BYTE 0xFF

4.2.3 Dokumentation der Funktionen

4.2.3.1 DT_double COM_byteArrayToDouble (const DT_byte *const array)

Konvertiert ein Byte-Array zu einem Double-Wert.

Konvertiert ein Byte-Array zu einem Double-Wert.

Parameter

array Byte-Array mit Double-Wert

Rückgabe

Double-Wert

4.2.3.2 `void COM_doubleToByteArray (const DT_double value, DT_byte
*const array)`

Konvertiert einen Double-Wert in ein Byte-Array.

Konvertiert einen Double-Wert in ein Byte-Array.

Parameter

value Double-Wert

array Zielfeld

4.2.3.3 `DT_double COM_getAngleFromPacket (const DT_byte *const result
)`

Ermittelt aus einem Packet den Winkel.

Ermittelt aus einem Packet den Winkel.

Parameter

result Zu prüfendes Packet

Rückgabe

Winkel

4.2.3.4 `DT_byte COM_getChecksum (const DT_byte *const packet, DT_size
l)`

Berechnet die Checksum.

Parameter

packet Paket

l Größe des pakets

Rückgabe

Checksum

4.2.3.5 DT_byte COM_getCpuID (const DT_leg *const *leg_l*)**4.2.3.6 DT_point COM_getPointFromPacket (const DT_byte *const *result*)**

Ermittelt aus einem Packet den Punkt.

Ermittelt aus einem Packet den Punkt.

Parameter

result Zu prüfendes Packet

Rückgabe

Point

4.2.3.7 DT_double COM_getSpeedFromPacket (const DT_byte *const *result*)

Ermittelt aus einem Packet die Anfahrgeschwindigkeit.

Ermittelt aus einem Packet die Anfahrgeschwindigkeit.

Parameter

result Zu prüfendes Packet

Rückgabe

Anfahrgeschwindigkeit

4.2.3.8 DT_bool COM_isAlive (DT_byte *cpuID*)

Sendet eine isAlive-Anfrage an einen Controller.

Sendet eine isAlive-Anfrage an einen Controller.

Parameter

cpuID ID des Controllers

Rückgabe

true, wenn Alive

4.2.3.9 DT_bool COM_isFoot (const DT_byte *const *result*)

Prüft ob Flag für Fußgelenk gesetzt ist.

Prüft ob Flag für Fußgelenk gesetzt ist.

Parameter

result Zu prüfendes Packet

Rückgabe

true, wenn Flag gesetzt

4.2.3.10 DT_bool COM_isGlobal (const DT_byte *const *result*)

Prüft ob Flag für globalen Punkt gesetzt ist.

Prüft ob Flag für globalen Punkt gesetzt ist.

Parameter

result Zu prüfendes Packet

Rückgabe

true, wenn Flag gesetzt

4.2.3.11 DT_bool COM_isHip (const DT_byte *const *result*)

Prüft ob Flag für Hüftgelenk gesetzt ist.

Prüft ob Flag für Hüftgelenk gesetzt ist.

Parameter

result Zu prüfendes Packet

Rückgabe

true, wenn Flag gesetzt

4.2.3.12 DT_bool COM_isKnee (const DT_byte *const *result*)

Prüft ob Flag für Kniegelenk gesetzt ist.

Prüft ob Flag für Kniegelenk gesetzt ist.

Parameter

result Zu prüfendes Packet

Rückgabe

true, wenn Flag gesetzt

4.2.3.13 DT_bool COM_isLeftLeg (const DT_byte *const result)

Prüft ob Flag für linkes Bein gesetzt ist.

Prüft ob Flag für linkes Bein gesetzt ist.

Parameter

result Zu prüfendes Packet

Rückgabe

true, wenn Flag gesetzt

4.2.3.14 DT_bool COM_isRightLeg (const DT_byte *const result)

Prüft ob Flag für rechtes Bein gesetzt ist.

Prüft ob Flag für rechtes Bein gesetzt ist.

Parameter

result Zu prüfendes Packet

Rückgabe

true, wenn Flag gesetzt

4.2.3.15 DT_byte COM_receive (USART_data_t *const usart_data, DT_byte *const dest)

USART-Empfangsmethode für Prozessorkommunikation.

Diese Methode liest den jeweiligen USART-Buffer aus und prüft, ob ein vollständiges Paket gemäß des Communication-Protokoll empfangen wurde.

Parameter

usart_data USART-Datenstruktur

dest Byte-Array für Antwort-Paket

Rückgabe

Länge des Antwortpakets

4.2.3.16 DT_size COM_requestStatus (DT_byte *cpuID*, DT_byte *param*, DT_byte *const *result*)

Ruft den Status eines Controllers ab.

Ruft den Status eines Controllers ab. Broadcast nicht möglich!

Parameter

cpuID ID des Controllers

param Parameter

result Zielfeld für Antwort

Rückgabe

Größe der empfangenen Antwort

4.2.3.17 DT_byte COM_send (DT_byte *const *packet*, DT_size *l*, DT_byte *const *result*, DT_bool *hasResponse*)

Versenden von Daten an anderen Controller.

Blockierendes Senden mit gleichzeitigem Empfangen der Antwort.

Parameter

packet Zuversendendes Paket

l Größe des Pakets

result Zielfeld für Antwort

hasResponse Wartet auf eine Antwort, wenn true

Rückgabe

Größe der empfangenen Antwort

4.2.3.18 void COM_sendACK (DT_byte *cpuID*)

Sendet ein ACK an einen Controller.

Sendet ein ACK an einen Controller.

Parameter

cpuID ID des Controllers

4.2.3.19 void COM_sendAction (DT_byte *cpuID*)

Sendet das ACTION-Kommando an einen Controller.

Sendet das ACTION-Kommando an einen Controller.

Parameter

cpuID ID des Controllers

4.2.3.20 DT_bool COM_sendAngle (DT_byte *cpuID*, const DT_double *angle*, const DT_byte *config*)

Sendet einen Winkel an einen Controller.

Sendet einen Winkel an einen Controller.

Parameter

cpuID ID des Controllers

angle Zuversendender Winkel

config Parameter, z.B. global, left, right ...

Rückgabe

false, wenn Fehler

4.2.3.21 void COM_sendNAK (DT_byte *cpuID*, DT_byte *errCode*)

Sendet ein NAK an einen Controller.

Sendet ein NAK an einen Controller.

Parameter

cpuID ID des Controllers

errCode Fehlercode

4.2.3.22 DT_bool COM_sendPoint (DT_byte *cpuID*, const DT_point *const *point*, const DT_byte *config*)

Sendet einen Punkt an einen Controller.

Sendet einen Punkt an einen Controller.

Parameter

cpuID ID des Controllers

point Zuversendender Punkt

config Parameter, z.B. global, left, right ...

Rückgabe

false, wenn Fehler

4.2.3.23 `DT_bool COM_sendPointAndSpeed (DT_byte cpuID, const DT_point *const point, const DT_double speed, const DT_byte config)`

Sendet einen Punkt und Anfahrgeschwindigkeit an einen Controller.

Sendet einen Punkt und Anfahrgeschwindigkeit zusammen in einem Packet an einen Controller.

Parameter

cpuID ID des Controllers

point Zuversendender Punkt

speed Anfahrgeschwindigkeit

config Parameter, z.B. global, left, right ...

Rückgabe

false, wenn Fehler

4.3 dynamixel.c-Dateireferenz

Methoden zur Steuerung der Dynamixel AX-12.

```
#include "include/dynamixel.h"
#include "include/utils.h"
#include "include/xmega.h"
#include <math.h>
```

Makrodefinitionen

- `#define START_BYTE 0xFF`
- `#define PING 0x01`
- `#define RD_DATA 0x02`
- `#define WR_DATA 0x03`
- `#define REG_WR 0x04`
- `#define ACT 0x05`
- `#define RESET 0x06`
- `#define SYC_WR 0x83`

- `#define ID 0x03`
- `#define BD 0x04`
- `#define MAX_TMP 0x0B`
- `#define STS_RT_LVL 0x10`
- `#define ALR_SHUTDOWN 0x12`
- `#define GL_POS 0x1E`
- `#define LED 0x19`
- `#define MV_SPEED 0x20`
- `#define PRT_POS 0x24`
- `#define PRT_SPEED 0x26`
- `#define PRT_TMP 0x2B`

Funktionen

- **DT_byte DNX_getChecksum** (const **DT_byte** *const packet, **DT_size** l)
Berechnet die Checksum.
- **DT_byte DNX_receive** (**USART_data_t** *const usart_data, **DT_byte** *const dest)
USART-Empfangsmethode.
- **DT_byte DNX_send** (**DT_byte** *const packet, **DT_size** l, **DT_byte** *const result, **DT_bool** hasResponse)
Versenden von Daten an Dynamixel.
- **DT_double DNX_convertAngle** (**DT_double** value)
Konvertiert Winkel in Bezug auf einen neuen Nullpunkt.
- **DT_double DNX_correctAngles** (**DT_byte** id, **DT_double** value)
Korrigiert Winkel für Dynamixel.
- **DT_bool DNX_setAngle** (**DT_byte** id, **DT_double** value, **DT_bool** regWrite)
Sendet einen Winkel an Servo.
- **DT_bool DNX_setAngleAndSpeed** (**DT_byte** id, **DT_double** angle, **DT_double** speed, **DT_bool** regWrite)
Sendet Winkel und Speed an Servo.
- **void DNX_setId** (**DT_byte** idOld, **DT_byte** idNew)
Vergibt einem Servos eine neue ID (ungetestet).
- **void DNX_setSpeed** (**DT_byte** id, **DT_byte** speed)
Setzt die Anfahrsgeschwindigkeit eines Servos (unvollendet).
- **DT_bool DNX_setLed** (**DT_byte** id, **DT_byte** value)
Schaltet die LED eines Servos an/aus.

- **void DNX_sendAction (DT_byte id)**
Sendet das ACTION-Kommando an einen Servo.
- **DT_double DNX_getAngle (DT_byte id)**
Liest den aktuellen Winkel eines Servos aus (unfertig).
- **DT_byte DNX_getSpeed (DT_byte id)**
Liest die Anfahrsgeschwindigkeit eines Servos aus (unfertig).
- **DT_byte DNX_getLed (DT_byte id)**
Liest den Status der LED aus (unfertig).
- **void DNX_getConnectedIDs (DT_leg *const leg_r, DT_leg *const leg_l)**
Ermittlung der angeschlossenen Dynamixel.

4.3.1 Ausführliche Beschreibung

Methoden zur Steuerung der Dynamixel AX-12.

4.3.2 Makro-Dokumentation

4.3.2.1 **#define ACT 0x05**

4.3.2.2 **#define ALR_SHUTDOWN 0x12**

4.3.2.3 **#define BD 0x04**

4.3.2.4 **#define GL_POS 0x1E**

4.3.2.5 **#define ID 0x03**

4.3.2.6 **#define LED 0x19**

4.3.2.7 **#define MAX_TMP 0x0B**

4.3.2.8 **#define MV_SPEED 0x20**

4.3.2.9 **#define PING 0x01**

4.3.2.10 **#define PRT_POS 0x24**

4.3.2.11 **#define PRT_SPEED 0x26**

4.3.2.12 **#define PRT_TMP 0x2B**

4.3.2.13 **#define RD_DATA 0x02**

4.3.2.14 **#define REG_WR 0x04**

4.3.2.15 **#define RESET 0x06**

4.3.2.16 **#define START_BYTE 0xFF**

4.3.2.17 **#define STS_RT_LVL 0x10**

4.3.2.18 **#define SYNC_WR 0x83**

4.3.2.19 **#define WR_DATA 0x03**

4.3.3 Dokumentation der Funktionen

4.3.3.1 **DT_double DNX_convertAngle (DT_double *value*)**

Konvertiert Winkel in Bezug auf einen neuen Nullpunkt.

Parameter

value Winkel in Grad

Rückgabe

Konvertierter Winkel in Grad

4.3.3.2 DT_double DNX_correctAngles (DT_byte *id*, DT_double *value*)

Korrigiert Winkel für Dynamixel.

Korrigiert Winkel für Dynamixel um hardwareseitige Veränderungen auszuschließen.

Parameter

id ID des Servos

value Winkel in Grad

Rückgabe

Korrigierter Winkel in Grad

4.3.3.3 DT_double DNX_getAngle (DT_byte *id*)

Liest den aktuellen Winkel eines Servos aus (unfertig).

Parameter

id ID des Servos

Rückgabe

Winkel in Grad

4.3.3.4 DT_byte DNX_getChecksum (const DT_byte *const *packet*, DT_size *l*)

Berechnet die Checksum.

Parameter

packet Paket

l Größe des pakets

Rückgabe

Checksum

4.3.3.5 void DNX_getConnectedIDs (DT_leg *const *leg_r*, DT_leg *const *leg_l*)

Ermittlung der angeschlossenen Dynamixel.

Parameter

leg_r Bein rechts

leg_l Bein links

4.3.3.6 DT_byte DNX_getLed (DT_byte *id*)

Liest den Status der LED aus (unfertig).

Parameter

id ID des Servos

Rückgabe

Wert der LED

4.3.3.7 DT_byte DNX_getSpeed (DT_byte *id*)

Liest die Anfahrsgeschwindigkeit eines Servos aus (unfertig).

Parameter

id ID des Servos

Rückgabe

Geschwindigkeit

4.3.3.8 DT_byte DNX_receive (USART_data_t *const *usart_data*, DT_byte *const *dest*)

USART-Empfangsmethode.

Diese Methode liest den jeweiligen USART-Buffer aus und prüft, ob ein vollständiges Paket gemäß des Dynamixel-Protokoll empfangen wurde.

Parameter

usart_data USART

dest Byte-Array für Antwort-Paket

Rückgabe

Länge des Antwortpakets

4.3.3.9 DT_byte DNX_send (DT_byte *const *packet*, DT_size *l*, DT_byte *const *result*, DT_bool *hasResponse*)

Versenden von Daten an Dynamixel.

Blockierendes Senden mit gleichzeitigem Empfangen der Antwort.

Parameter

packet Zuversendendes Paket

l Größe des Pakets

result Zielfeld für Antwort

hasResponse Wartet auf eine Antwort, wenn true

Rückgabe

Größe der empfangenen Antwort

4.3.3.10 void DNX_sendAction (DT_byte *id*)

Sendet das ACTION-Kommando an einen Servo.

Sendet das ACTION-Kommando an einen Servo.

Parameter

id ID des Servo

4.3.3.11 DT_bool DNX_setAngle (DT_byte *id*, DT_double *value*, DT_bool *regWrite*)

Sendet einen Winkel an Servo.

Parameter

id ID des Servos

value Winkel in Grad

regWrite Winkel wird in Puffer des Servos gespeichert und erst bei ACTION angefahren, wenn true

4.3.3.12 DT_bool DNX_setAngleAndSpeed (DT_byte *id*, DT_double *angle*, DT_double *speed*, DT_bool *regWrite*)

Sendet Winkel und Speed an Servo.

Parameter

id ID des Servos

angle Winkel in Grad

speed Anfahrgeschwindigkeit

regWrite Werte werden in Puffer des Servos gespeichert und erst bei ACTION ausgeführt, wenn true

4.3.3.13 void DNX_setId (DT_byte *idOld*, DT_byte *idNew*)

Vergibt einem Servos eine neue ID (ungetestet).

Parameter

idOld ID des zu verändernden Servos

idNew Zusetzende ID

4.3.3.14 DT_bool DNX_setLed (DT_byte *id*, DT_byte *value*)

Schaltet die LED eines Servos an/aus.

Parameter

id ID des Servos

value Wert für LED (0x00 / 0x01)

4.3.3.15 void DNX_setSpeed (DT_byte *id*, DT_byte *speed*)

Setzt die Anfahrgeschwindigkeit eines Servos (unvollendet).

Parameter

id ID des Servos

speed Geschwindigkeit

4.4 evolutionaryAlgorithm.c-Dateireferenz

Evolutionärer Algorithmus zur Startpunktfindung.

```
#include "include/datatypes.h"
#include "include/evolutionaryHelper.h"
#include "include/evolutionaryAlgorithm.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

Makrodefinitionen

- `#define X_MIN -83`
- `#define X_MAX 83`
- `#define Y_MIN 42`
- `#define Y_MAX 129`

Funktionen

- `void generatePopulation (DT_individuum *const , const DT_int)`
- `void printPopulation (DT_individuum *, DT_int)`
- `DT_point generatePoint (DT_int)`
- `DT_double getRandomNumber (DT_int, DT_int)`
- `void fitnessproportionalSelection (const DT_individuum *const , DT_int *, const DT_int, const DT_int)`
- `void recombination (const DT_individuum const *, DT_individuum *const , const DT_int const *, const DT_int, const DT_int, const DT_double)`
- `void uniformCrossover (const DT_individuum const *, const DT_individuum const *, DT_individuum *, DT_individuum *)`
- `void mutation (DT_individuum *, const DT_int, const DT_double)`
- `void gleichverteilte_reellwertige_mutation (DT_point *, const DT_double)`
- `void bestSelection (DT_individuum *const , DT_individuum *, const DT_int, const DT_int)`
- `void getScores (DT_vector *const , DT_individuum *const , const DT_int)`
- `DT_individuum evolutionaryAlgorithm (const DT_int popsize, const DT_int generations, DT_vector *const v)`
- `DT_point getPointFromIndividuum (DT_individuum *A)`
- `DT_point getIsectFromIndividuum (DT_individuum *A)`

4.4.1 Ausführliche Beschreibung

Evolutionärer Algorithmus zur Startpunktfindung. Startpunktfindung auf Basis eines Evolutionären Algorithmus.

4.4.2 Makro-Dokumentation

4.4.2.1 `#define X_MAX 83`

4.4.2.2 `#define X_MIN -83`

4.4.2.3 `#define Y_MAX 129`

4.4.2.4 `#define Y_MIN 42`

4.4.3 Dokumentation der Funktionen

4.4.3.1 `void bestSelection (DT_individuum * const P_nextGen,
DT_individuum * P, const DT_int popsizeNextGen, const DT_int
popsize)`

4.4.3.2 `DT_individuum evolutionaryAlgorithm (const DT_int popsize, const
DT_int generations, DT_vector *const v)`

4.4.3.3 `void fitnessproportionalSelection (const DT_individuum * const P,
DT_int * I, const DT_int popsize, const DT_int parentCnt)`

4.4.3.4 `DT_point generatePoint (DT_int prob)`

4.4.3.5 `void generatePopulation (DT_individuum * const P, const DT_int
popsize)`

4.4.3.6 `DT_point getIsectFromIndividuum (DT_individuum * A)`

4.4.3.7 `DT_point getPointFromIndividuum (DT_individuum * A)`

4.4.3.8 `DT_double getRandomNumber (DT_int min, DT_int max)`

4.4.3.9 `void getScores (DT_vector * const v, DT_individuum * const P, const
DT_int popsize)`

4.4.3.10 `void gleichverteilte_reellwertige_mutation (DT_point * A, const
DT_double pm)`

4.4.3.11 `void mutation (DT_individuum * P, const DT_int popsize, const
DT_double pm)`

4.4.3.12 `void printPopulation (DT_individuum * P, DT_int size)`

4.4.3.13 `void recombination (const DT_individuum const * P, DT_individuum
* const P_nextGen, const DT_int const * I, const DT_int popsize,
const DT_int parentCnt, const DT_double px)`

4.4.3.14 `void uniformCrossover (const DT_individuum const * A, const
DT_individuum const * B, DT_individuum * C, DT_individuum * D
)`

Erzeugt am Thu Dec 30 2010 16:08:55 für CM-Bot_API von Doxygen

4.5 evolutionaryHelper.c-Dateireferenz

Hilfsfunktion für Evolutionären Algorithmus.

```
#include "include/evolutionaryHelper.h"
#include <math.h>
#include <stdio.h>
```

Makrodefinitionen

- **#define NO_VALUE 0x00**
Kein Rückgabewert der Funktion.
- **#define FIRST_VALUE 0x01**
Der erste Rückgabewert enthält einen gültigen Schnittpunkt.
- **#define SECOND_VALUE 0x02**
Der zweite Rückgabewert enthält einen gültigen Schnittpunkt.

Funktionen

- **DT_bool isectLinFuncs** (const **DT_lin_func** *, const **DT_lin_func** *, **DT_point** *)
- **void initFunctions** ()
Initialisiert die globalen Funktionen.
- **void initPoints** ()
- **DT_double f_lin** (const **DT_lin_func** *, const **DT_double**)
- **DT_double f_circ** (const **DT_half_circle** *, const **DT_double**)
- **DT_byte isectLinCirc** (const **DT_lin_func** *, const **DT_half_circle** *, **DT_point** *, **DT_point** *)
- **DT_bool isBetweenPoints** (const **DT_point** *, const **DT_point** *, const **DT_point** *)
- **DT_double getDistance** (const **DT_point** *const, const **DT_point** *const)
- **DT_point getNearerPoint** (const **DT_point** *, const **DT_point** *, const **DT_point** *)
- **DT_bool isVectorialPoint** (const **DT_point** *const, const **DT_vector** *, const **DT_point** *const)
- **void initEvoAlg** ()
- **DT_bool isInArea** (const **DT_point** *p)
- **DT_double scorePoint** (**DT_vector** *const v, const **DT_point** *const p, **DT_point** *s)
- **void bubblesort** (**DT_individuum** *P, const **DT_int** n)
- **DT_double max** (**DT_double** x, **DT_double** y)
- **DT_double min** (**DT_double** x, **DT_double** y)
- **void getFunctionOfPoints** (**DT_lin_func** *f, const **DT_point** *const p1, const **DT_point** *const p2)

Variablen

- DT_point A
- DT_point B
- DT_point C
- DT_point D
- DT_point E
- DT_point F
- DT_point G
- DT_lin_func AB
- DT_lin_func CD
- DT_half_circle CEA
- DT_half_circle DFB

4.5.1 Ausführliche Beschreibung

Hilfsfunktion für Evolutionären Algorithmus. Mathematische Funktionen.

4.5.2 Makro-Dokumentation

4.5.2.1 #define FIRST_VALUE 0x01

Der erste Rückgabewert enthält einen gültigen Schnittpunkt.

4.5.2.2 #define NO_VALUE 0x00

Kein Rückgabewert der Funktion.

4.5.2.3 #define SECOND_VALUE 0x02

Der zweite Rückgabewert enthält einen gültigen Schnittpunkt.

4.5.3 Dokumentation der Funktionen

4.5.3.1 void bubblesort (DT_individuum * *P*, const DT_int *n*)

4.5.3.2 DT_double f_circ (const DT_half_circle * *fI*, const DT_double *x*)

4.5.3.3 DT_double f_lin (const DT_lin_func * *f*, const DT_double *x*)

4.5.3.4 DT_double getDistance (const DT_point * const *p1*, const DT_point * const *p2*)

4.5.3.5 void getFunctionOfPoints (DT_lin_func * *f*, const DT_point * const *p1*, const DT_point * const *p2*)

4.5.3.6 DT_point getNearerPoint (const DT_point * *p_ref*, const DT_point * *p1*, const DT_point * *p2*)

4.5.3.7 void initEvoAlg ()

4.5.3.8 void initFunctions ()

Initialisiert die globalen Funktionen.

Initialisiert die globalen Funktionen zur Beschreibung der Schnittfläche des Arbeitsraums.

4.5.3.9 void initPoints ()

4.5.3.10 DT_bool isBetweenPoints (const DT_point * *p1*, const DT_point * *p2*, const DT_point * *p_between*)

4.5.3.11 DT_byte isectLinCirc (const DT_lin_func * *f1*, const DT_half_circle * *f2*, DT_point * *isect1*, DT_point * *isect2*)

4.5.3.12 DT_bool isectLinFuncs (const DT_lin_func * *f1*, const DT_lin_func * *f2*, DT_point * *isect*)

4.5.3.13 DT_bool isInArea (const DT_point * *p*)

4.5.3.14 DT_bool isVectorialPoint (const DT_point * const *p_ref*, const DT_vector * *v*, const DT_point * const *p_chk*)

4.5.3.15 DT_double max (DT_double *x*, DT_double *y*)

4.5.3.16 DT_double min (DT_double *x*, DT_double *y*)

4.5.3.17 DT_double scorePoint (DT_vector *const *v*, const DT_point *const *p*, DT_point * *s*)

4.5.4 Variablen-Dokumentation

4.5.4.1 DT_point A

4.5.4.2 DT_lin_func AB

4.5.4.3 DT_point B

4.5.4.4 DT_point C

4.5.4.5 DT_lin_func CD

4.5.4.6 DT_half_circle CEA

4.5.4.7 DT_point D

4.5.4.8 DT_half_circle DFB

4.5.4.9 DT_point E

4.5.4.10 DT_point F

4.5.4.11 DT_point G

4.6 evolutionaryWalking.c-Dateireferenz

```
#include <stdio.h>
#include <math.h>
#include "include/kinematics.h"
#include "include/utils.h"
#include "include/evolutionaryHelper.h"
#include "include/evolutionaryAlgorithm.h"
#include "include/dynamixel.h"
#include "include/xmega.h"
#include "include/communication.h"
#include "include/movement.h"
#include "include/remote.h"
```

Makrodefinitionen

- `#define TEST_ON`
- `#define OFFSET 50`
- `#define NO_OFFSET 0`

Funktionen

- `void master ()`
- `int main (void)`
- `void invertVector (DT_vector *v)`
- `DT_point copyPoint (const DT_point *const p)`
- `void switchLegs ()`
- `void initConf ()`
- `void TripodGaitMove (DT_point *pM, DT_point *pS, const DT_double speed, const DT_double offset)`
- `void init_pMpSpMiddle ()`
- `void calculateMovementPoints ()`
- `void doStepMove (DT_point *pM, DT_point *pS, const DT_double speed)`
- `void prepareStepMove (DT_point *pM, DT_point *pS, const DT_double speed, const DT_double offset)`
- `void doStep (const DT_double speed)`
- `void evolutionaryCalculation (DT_vector *v, const DT_double speed)`
- `void waitForButton3 ()`

Variablen

- `DT_leg leg_r`
- `DT_leg leg_l`

- DT_byte MasterActive
- DT_byte SlavesActive
- DT_byte MasterInactive
- DT_byte SlavesInactive
- DT_byte cpuID
- DT_point pM
- DT_point pS
- DT_point midM
- DT_point midS
- DT_point isectM
- DT_point isectS
- DT_point pMiddle

4.6.1 Ausführliche Beschreibung

Testprogramm für Evolutionären Algorithmus zur Startpunktfindung.

4.6.2 Makro-Dokumentation

4.6.2.1 `#define NO_OFFSET 0`

4.6.2.2 `#define OFFSET 50`

4.6.2.3 `#define TEST_ON`

4.6.3 Dokumentation der Funktionen

4.6.3.1 `void calculateMovementPoints ()`

4.6.3.2 `DT_point copyPoint (const DT_point *const p)`

4.6.3.3 `void doStep (const DT_double speed)`

4.6.3.4 `void doStepMove (DT_point * pM, DT_point * pS, const DT_double speed)`

4.6.3.5 `void evolutionaryCalculation (DT_vector * v, const DT_double speed)`

4.6.3.6 `void init_pMpSpMiddle ()`

4.6.3.7 `void initConf ()`

4.6.3.8 `void invertVector (DT_vector * v)`

4.6.3.9 `int main (void)`

4.6.3.10 `void master ()`

4.6.3.11 `void prepareStepMove (DT_point * pM, DT_point * pS, const DT_double speed, const DT_double offset)`

4.6.3.12 `void switchLegs ()`

4.6.3.13 `void TripodGaitMove (DT_point * pM, DT_point * pS, const DT_double speed, const DT_double offset)`

4.6.3.14 `void waitForButton3 ()`

4.6.4 Variablen-Dokumentation

4.6.4.1 `DT_byte cpuID`

4.6.4.2 `DT_point isectM`

4.6.4.3 `DT_point isectS`

4.6.4.4 `DT_leg leg_l`

Erzeugt am Thu Dec 30 2010 16:08:55 für CM-Bot_API von Doxygen

4.6.4.5 `DT_leg leg_r`

4.6.4.6 `DT_byte MasterActive`

4.6.4.7 `DT_byte MasterInactive`

4.6.4.8 `DT_point midM`

```
#include <stdint.h>
#include <stdbool.h>
#include <stdlib.h>
```

Makrodefinitionen

- **#define F_CPU 2000000UL**
Define default CPU frequency, if this is not already defined.
- **#define AVR_ENTER_CRITICAL_REGION()**
This macro will protect the following code from interrupts.
- **#define AVR_LEAVE_CRITICAL_REGION() SREG = saved_sreg;**
This macro must always be used in conjunction with AVR_ENTER_CRITICAL_REGION so the interrupts are enabled again.

4.7.1 Ausführliche Beschreibung

This file implements some macros that makes the IAR C-compiler and avr-gcc work with the same code base for the AVR architecture.

Documentation

For comprehensive code documentation, supported compilers, compiler settings and supported devices see readme.html

Autor

Atmel Corporation: <http://www.atmel.com>
Support email: avr@atmel.com

Revision:

613

Date:

2006-04-07 14:40:07 +0200 (fr, 07 apr 2006)

Copyright (c) 2008, Atmel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name of ATMEL may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.7.2 Makro-Dokumentation

4.7.2.1 #define AVR_ENTER_CRITICAL_REGION()

Wert:

```
uint8_t volatile saved_sreg = SREG; \
    cli();
```

This macro will protect the following code from interrupts.

4.7.2.2 #define AVR_LEAVE_CRITICAL_REGION() SREG = saved_sreg;

This macro must always be used in conjunction with AVR_ENTER_CRITICAL_REGION so the interrupts are enabled again.

4.7.2.3 #define F_CPU 2000000UL

Define default CPU frequency, if this is not already defined.

4.8 include/clksys_driver.h-Dateireferenz

XMEGA Clock System driver header file.

```
#include "avr_compiler.h"
```

Makrodefinitionen

- #define **CLKSYS_Enable**(_oscSel) (OSC.CTRL |= (_oscSel))

This macro enables the selected oscillator.

- **#define CLKSYS_IsReady(_oscSel)** (OSC.STATUS & (_oscSel))
This macro check if selected oscillator is ready.
- **#define CLKSYS_RTC_ClockSource_Disable()** (CLK.RTCCTRL &= ~CLK_RTCEN_bm)
This macro disables routing of clock signals to the Real-Time Counter (RTC).
- **#define CLKSYS_AutoCalibration_Disable(_clk)** ((_clk).CTRL &= ~DFLL_ENABLE_bm)
This macro disables the automatic calibration of the selected internal oscillator.

Funktionen

- void **CCPWrite** (volatile uint8_t *address, uint8_t value)
CCP write helper function written in assembly.
- void **CLKSYS_XOSC_Config** (OSC_FRQRANGE_t freqRange, bool lowPower32kHz, OSC_XOSCSEL_t xoscModeSelection)
This function configures the external oscillator.
- void **CLKSYS_PLL_Config** (OSC_PLLSRC_t clockSource, uint8_t factor)
This function configures the internal high-frequency PLL.
- uint8_t **CLKSYS_Disable** (uint8_t oscSel)
This function disables the selected oscillator.
- void **CLKSYS_Prescalers_Config** (CLK_PSADIV_t PSAfactor, CLK_PSBCDIV_t PSBCfactor)
This function changes the prescaler configuration.
- uint8_t **CLKSYS_Main_ClockSource_Select** (CLK_SCLKSEL_t clockSource)
This function selects the main system clock source.
- void **CLKSYS_RTC_ClockSource_Enable** (CLK_RTCSRC_t clockSource)
This function selects a Real-Time Counter clock source.
- void **CLKSYS_AutoCalibration_Enable** (uint8_t clkSource, bool extReference)
This function enables automatic calibration of the selected internal oscillator.
- void **CLKSYS_XOSC_FailureDetection_Enable** (void)
This function enables the External Oscillator Failure Detection (XOSCFD) feature.

- void **CLKSYS_Configuration_Lock** (void)

This function lock the entire clock system configuration.

4.8.1 Ausführliche Beschreibung

XMEGA Clock System driver header file. This file contains the function prototypes and enumerator definitions for various configuration parameters for the XMEGA Clock System driver.

The driver is not intended for size and/or speed critical code, since most functions are just a few lines of code, and the function call overhead would decrease code performance. The driver is intended for rapid prototyping and documentation purposes for getting started with the XMEGA Clock System.

For size and/or speed critical code, it is recommended to copy the function contents directly into your application instead of making a function call.

Application note:

AVR1003: Using the XMEGA Clock System

Documentation

For comprehensive code documentation, supported compilers, compiler settings and supported devices see readme.html

Autor

Atmel Corporation: <http://www.atmel.com>
Support email: avr@atmel.com

Revision:

1665

Date:

2008-06-05 09:21:50 +0200 (to, 05 jun 2008)

Copyright (c) 2008, Atmel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of ATMEL may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.8.2 Makro-Dokumentation

4.8.2.1 `#define CLKSYS_AutoCalibration_Disable(_clk) ((_clk).CTRL &= ~DFLL_ENABLE_bm)`

This macro disables the automatic calibration of the selected internal oscillator.

Parameter

`_clk` Clock source calibration to disable, either DFLLRC2M or DFLLRC32M.

4.8.2.2 `#define CLKSYS_Enable(_oscSel) (OSC.CTRL |= (_oscSel))`

This macro enables the selected oscillator.

Zu beachten

Note that the oscillator cannot be used as a main system clock source without being enabled and stable first. Check the ready flag before using the clock. The function `CLKSYS_IsReady(_oscSel)` (S. 46) can be used to check this.

Parameter

`_oscSel` Bitmask of selected clock. Can be one of the following `OSC_RC2MEN_bm`, `OSC_RC32MEN_bm`, `OSC_RC32KEN_bm`, `OSC_XOSCEN_bm`, `OSC_PLEN_bm`.

4.8.2.3 `#define CLKSYS_IsReady(_oscSel) (OSC.STATUS & (_oscSel))`

This macro check if selected oscillator is ready.

This macro will return non-zero if is is running, regardless if it is used as a main clock source or not.

Parameter

_oscSel Bitmask of selected clock. Can be one of the following OSC_RC2MEN_bm, OSC_RC32MEN_bm, OSC_RC32KEN_bm, OSC_XOSCEN_bm, OSC_PPLEN_bm.

Rückgabe

Non-zero if oscillator is ready and running.

4.8.2.4 #define CLKSYS_RTC_ClockSource_Disable() (CLK.RTCCTRL &= ~CLK_RTCEN_bm)

This macro disables routing of clock signals to the Real-Time Counter (RTC).

Disabling the RTC saves power if the RTC is not in use.

4.8.3 Dokumentation der Funktionen

4.8.3.1 void CCPWrite (volatile uint8_t * address, uint8_t value)

CCP write helper function written in assembly.

This function is written in assembly because of the timecritical operation of writing to the registers.

Parameter

address A pointer to the address to write to.

value The value to put in to the register.

4.8.3.2 void CLKSYS_AutoCalibration_Enable (uint8_t clkSource, bool extReference)

This function enables automatic calibration of the selected internal oscillator.

Either the internal 32kHz RC oscillator or an external 32kHz crystal can be used as a calibration reference. The user must make sure that the selected reference is ready and running.

Parameter

clkSource Clock source to calibrate, either OSC_RC2MCREF_bm or OSC_RC32MCREF_bm.

extReference True if external crystal should be used as reference.

4.8.3.3 void CLKSYS_Configuration_Lock (void)

This function lock the entire clock system configuration.

This will lock the configuration until the next reset, or until the External Oscillator Failure Detections (XOSCFD) feature detects a failure and switches to internal 2MHz RC oscillator.

4.8.3.4 uint8_t CLKSYS_Disable (uint8_t *oscSel*)

This function disables the selected oscillator.

This function will disable the selected oscillator if possible. If it is currently used as a main system clock source, hardware will disregard the disable attempt, and this function will return zero. If it fails, change to another main system clock source and try again.

Parameter

oscSel Bitmask of selected clock. Can be one of the following OSC_RC2MEN_bm, OSC_RC32MEN_bm, OSC_RC32KEN_bm, OSC_XOSCEN_bm, OSC_PLEN_bm.

Rückgabe

Non-zero if oscillator was disabled successfully.

4.8.3.5 uint8_t CLKSYS_Main_ClockSource_Select (CLK_SCLKSEL_t *clockSource*)

This function selects the main system clock source.

Hardware will disregard any attempts to select a clock source that is not enabled or not stable. If the change fails, make sure the source is ready and running and try again.

Parameter

clockSource Clock source to use as input for the system clock prescaler block.

Rückgabe

Non-zero if change was successful.

4.8.3.6 void CLKSYS_PLL_Config (OSC_PLLSRC_t *clockSource*, uint8_t *factor*)

This function configures the internal high-frequency PLL.

Configuration of the internal high-frequency PLL to the correct values. It is used to define the input of the PLL and the factor of multiplication of the input clock source.

Zu beachten

Note that the oscillator cannot be used as a main system clock source without being enabled and stable first. Check the ready flag before using the clock. The macro **CLKSYS_IsReady(_oscSel)** (S. 46) can be used to check this.

Parameter

clockSource Reference clock source for the PLL, must be above 0.4MHz.

factor PLL multiplication factor, must be from 1 to 31, inclusive.

4.8.3.7 void CLKSYS_Prescalers_Config (CLK_PSADIV_t *PSAfactor*, CLK_PSBCDIV_t *PSBCfactor*)

This function changes the prescaler configuration.

Change the configuration of the three system clock prescaler is one single operation. The user must make sure that the main CPU clock does not exceed recommended limits.

Parameter

PSAfactor Prescaler A division factor, OFF or 2 to 512 in powers of two.

PSBCfactor Prescaler B and C division factor, in the combination of (1,1), (1,2), (4,1) or (2,2).

4.8.3.8 void CLKSYS_RTC_ClockSource_Enable (CLK_RTCSRC_t *clockSource*)

This function selects a Real-Time Counter clock source.

Selects the clock source for use by the Real-Time Counter (RTC) and enables clock signal routing to the RTC module.

Parameter

clockSource Clock source to use for the RTC.

4.8.3.9 void CLKSYS_XOSC_Config (OSC_FRQRANGE_t *freqRange*, bool *lowPower32kHz*, OSC_XOSCSEL_t *xoscModeSelection*)

This function configures the external oscillator.

Zu beachten

Note that the oscillator cannot be used as a main system clock source without being enabled and stable first. Check the ready flag before using the clock. The macro **CLKSYS_IsReady(_oscSel)** (S. 46) can be used to check this.

Parameter

freqRange Frequency range for high-frequency crystal, does not apply for external clock or 32kHz crystals.

lowPower32kHz True of high-quality watch crystals are used and low-power oscillator is desired.

xoscModeSelection Combined selection of oscillator type (or external clock) and startup times.

4.8.3.10 void CLKSYS_XOSC_FailureDetection_Enable (void)

This function enables the External Oscillator Failure Detection (XOSCFD) feature.

The feature will stay enabled until next reset. Note that the XOSCFD _will_ issue the XOSCF Non-maskable Interrupt (NMI) regardless of any interrupt priorities and settings. Therefore, make sure that a handler is implemented for the XOSCF NMI when you enable it.

4.9 include/communication.h-Dateireferenz

Methoden zur Kommunikation der CPUs.

```
#include "datatypes.h"
#include "usart_driver.h"
```

Makrodefinitionen

- #define **COM_MASTER** 0x02
- #define **COM_SLAVE1B** 0x01
- #define **COM_SLAVE3F** 0x03
- #define **COM_BRDCAST_ID** 0xFE
- #define **COM_NOCPUID** 0x00
- #define **COM_STATUS** 0x01
- #define **COM_ACTION** 0x02
- #define **COM_POINT** 0x03
- #define **COM_ANGLE** 0x04
- #define **COM_SPEED** 0x05
- #define **COM_IS_ALIVE** 0x01
- #define **COM_ACK** 0x06
- #define **COM_NAK** 0x15
- #define **COM_ERR_ANGLE_LIMIT** 0x01
- #define **COM_ERR_POINT_OUT_OF_BOUNDS** 0x02
- #define **COM_ERR_DEFAULT_ERROR** 0x03
- #define **COM_CONF_RIGHT** 0x01
- #define **COM_CONF_LEFT** 0x02

- `#define COM_CONF_GLOB 0x04`
- `#define COM_CONF_HIP 0x08`
- `#define COM_CONF_KNEE 0x10`
- `#define COM_CONF_FOOT 0x20`

Funktionen

- **DT_byte COM_send (DT_byte *const, DT_size, DT_byte *const, DT_bool)**
Versenden von Daten an anderen Controller.
- **DT_byte COM_receive (USART_data_t *const, DT_byte *const)**
USART-Empfangsmethode für Prozessorkommunikation.
- **DT_size COM_requestStatus (DT_byte, DT_byte, DT_byte *const)**
Ruft den Status eines Controllers ab.
- **DT_bool COM_sendPoint (DT_byte, const DT_point *const, const DT_byte)**
Sendet einen Punkt an einen Controller.
- **DT_bool COM_sendPointAndSpeed (DT_byte, const DT_point *const, const DT_double, const DT_byte)**
Sendet einen Punkt und Anfahrsgeschwindigkeit an einen Controller.
- **DT_bool COM_sendAngle (DT_byte, const DT_double, const DT_byte)**
Sendet einen Winkel an einen Controller.
- **void COM_sendAction (DT_byte)**
Sendet das ACTION-Kommando an einen Controller.
- **DT_bool COM_isAlive (DT_byte)**
Sendet eine isAlive-Anfrage an einen Controller.
- **void COM_sendACK (DT_byte)**
Sendet ein ACK an einen Controller.
- **void COM_sendNAK (DT_byte, DT_byte)**
Sendet ein NAK an einen Controller.
- **DT_byte COM_getCpuID (const DT_leg *const)**
- **DT_double COM_byteArrayToDouble (const DT_byte *const)**
Konvertiert ein Byte-Array zu einem Double-Wert.
- **void COM_doubleToByteArray (const DT_double, DT_byte *const)**
Konvertiert einen Double-Wert in ein Byte-Array.

- **DT_point COM_getPointFromPacket** (const **DT_byte** *const)
Ermittelt aus einem Packet den Punkt.
- **DT_double COM_getAngleFromPacket** (const **DT_byte** *const)
Ermittelt aus einem Packet den Winkel.
- **DT_double COM_getSpeedFromPacket** (const **DT_byte** *const)
Ermittelt aus einem Packet die Anfahrsgeschwindigkeit.
- **DT_bool COM_isLeftLeg** (const **DT_byte** *const result)
Prüft ob Flag für linkes Bein gesetzt ist.
- **DT_bool COM_isRightLeg** (const **DT_byte** *const result)
Prüft ob Flag für rechtes Bein gesetzt ist.
- **DT_bool COM_isGlobal** (const **DT_byte** *const result)
Prüft ob Flag für globalen Punkt gesetzt ist.
- **DT_bool COM_isHip** (const **DT_byte** *const result)
Prüft ob Flag für Hüftgelenk gesetzt ist.
- **DT_bool COM_isKnee** (const **DT_byte** *const result)
Prüft ob Flag für Kniegelenk gesetzt ist.
- **DT_bool COM_isFoot** (const **DT_byte** *const result)
Prüft ob Flag für Fußgelenk gesetzt ist.

4.9.1 Ausführliche Beschreibung

Methoden zur Kommunikation der CPUs.

4.9.2 Makro-Dokumentation

4.9.2.1 `#define COM_ACK 0x06`

4.9.2.2 `#define COM_ACTION 0x02`

4.9.2.3 `#define COM_ANGLE 0x04`

4.9.2.4 `#define COM_BRDCAST_ID 0xFE`

4.9.2.5 `#define COM_CONF_FOOT 0x20`

4.9.2.6 `#define COM_CONF_GLOB 0x04`

4.9.2.7 `#define COM_CONF_HIP 0x08`

4.9.2.8 `#define COM_CONF_KNEE 0x10`

4.9.2.9 `#define COM_CONF_LEFT 0x02`

4.9.2.10 `#define COM_CONF_RIGHT 0x01`

4.9.2.11 `#define COM_ERR_ANGLE_LIMIT 0x01`

4.9.2.12 `#define COM_ERR_DEFAULT_ERROR 0x03`

4.9.2.13 `#define COM_ERR_POINT_OUT_OF_BOUNDS 0x02`

4.9.2.14 `#define COM_IS_ALIVE 0x01`

4.9.2.15 `#define COM_MASTER 0x02`

4.9.2.16 `#define COM_NAK 0x15`

4.9.2.17 `#define COM_NOCPUID 0x00`

4.9.2.18 `#define COM_POINT 0x03`

4.9.2.19 `#define COM_SLAVE1B 0x01`

4.9.2.20 `#define COM_SLAVE3F 0x03`

4.9.2.21 `#define COM_SPEED 0x05`

4.9.2.22 `#define COM_STATUS 0x01`

4.9.3 Dokumentation der Funktionen

4.9.3.1 `DT_double COM_byteArrayToDouble (const DT_byte *const array)`

Konvertiert ein Byte-Array zu einem Double-Wert.
Erzeugt am Thu Dec 30 2010 16:08:55 für CM-Bot_API von Doxygen

Konvertiert ein Byte-Array zu einem Double-Wert.

Parameter

array Byte-Array mit Double-Wert

Rückgabe

Double-Wert

4.9.3.2 void COM_doubleToByteArray (const DT_double *value*, DT_byte *const *array*)

Konvertiert einen Double-Wert in ein Byte-Array.

Konvertiert einen Double-Wert in ein Byte-Array.

Parameter

value Double-Wert

array Zielfeld

4.9.3.3 DT_double COM_getAngleFromPacket (const DT_byte *const *result*)

Ermittelt aus einem Packet den Winkel.

Ermittelt aus einem Packet den Winkel.

Parameter

result Zu prüfendes Packet

Rückgabe

Winkel

4.9.3.4 DT_byte COM_getCpuID (const DT_leg * *const*)**4.9.3.5 DT_point COM_getPointFromPacket (const DT_byte *const *result*)**

Ermittelt aus einem Packet den Punkt.

Ermittelt aus einem Packet den Punkt.

Parameter

result Zu prüfendes Packet

Rückgabe

Point

4.9.3.6 DT_double COM_getSpeedFromPacket (const DT_byte *const *result*)

Ermittelt aus einem Packet die Anfahrgeschwindigkeit.

Ermittelt aus einem Packet die Anfahrgeschwindigkeit.

Parameter

result Zu prüfendes Packet

Rückgabe

Anfahrgeschwindigkeit

4.9.3.7 DT_bool COM_isAlive (DT_byte *cpuID*)

Sendet eine isAlive-Anfrage an einen Controller.

Sendet eine isAlive-Anfrage an einen Controller.

Parameter

cpuID ID des Controllers

Rückgabe

true, wenn Alive

4.9.3.8 DT_bool COM_isFoot (const DT_byte *const *result*)

Prüft ob Flag für Fußgelenk gesetzt ist.

Prüft ob Flag für Fußgelenk gesetzt ist.

Parameter

result Zu prüfendes Packet

Rückgabe

true, wenn Flag gesetzt

4.9.3.9 DT_bool COM_isGlobal (const DT_byte *const *result*)

Prüft ob Flag für globalen Punkt gesetzt ist.

Prüft ob Flag für globalen Punkt gesetzt ist.

Parameter

result Zu prüfendes Packet

Rückgabe

true, wenn Flag gesetzt

4.9.3.10 DT_bool COM_isHip (const DT_byte *const *result*)

Prüft ob Flag für Hüftgelenk gesetzt ist.

Prüft ob Flag für Hüftgelenk gesetzt ist.

Parameter

result Zu prüfendes Packet

Rückgabe

true, wenn Flag gesetzt

4.9.3.11 DT_bool COM_isKnee (const DT_byte *const *result*)

Prüft ob Flag für Kniegelenk gesetzt ist.

Prüft ob Flag für Kniegelenk gesetzt ist.

Parameter

result Zu prüfendes Packet

Rückgabe

true, wenn Flag gesetzt

4.9.3.12 DT_bool COM_isLeftLeg (const DT_byte *const *result*)

Prüft ob Flag für linkes Bein gesetzt ist.

Prüft ob Flag für linkes Bein gesetzt ist.

Parameter

result Zu prüfendes Packet

Rückgabe

true, wenn Flag gesetzt

4.9.3.13 DT_bool COM_isRightLeg (const DT_byte *const *result*)

Prüft ob Flag für rechtes Bein gesetzt ist.

Prüft ob Flag für rechtes Bein gesetzt ist.

Parameter

result Zu prüfendes Packet

Rückgabe

true, wenn Flag gesetzt

4.9.3.14 DT_byte COM_receive (USART_data_t *const *usart_data*, DT_byte *const *dest*)

USART-Empfangsmethode für Prozessorkommunikation.

Diese Methode liest den jeweiligen USART-Buffer aus und prüft, ob ein vollständiges Paket gemäß des Communication-Protokoll empfangen wurde.

Parameter

usart_data USART-Datenstruktur

dest Byte-Array für Antwort-Paket

Rückgabe

Länge des Antwortpakets

4.9.3.15 DT_size COM_requestStatus (DT_byte *cpuID*, DT_byte *param*, DT_byte *const *result*)

Ruft den Status eines Controllers ab.

Ruft den Status eines Controllers ab. Broadcast nicht möglich!

Parameter

cpuID ID des Controllers

param Parameter

result Zielfeld für Antwort

Rückgabe

Größe der empfangenen Antwort

4.9.3.16 DT_byte COM_send (DT_byte *const *packet*, DT_size *l*, DT_byte *const *result*, DT_bool *hasResponse*)

Versenden von Daten an anderen Controller.

Blockierendes Senden mit gleichzeitigem Empfangen der Antwort.

Parameter

packet Zuversendendes Paket

l Größe des Pakets

result Zielfeld für Antwort

hasResponse Wartet auf eine Antwort, wenn true

Rückgabe

Größe der empfangenen Antwort

4.9.3.17 void COM_sendACK (DT_byte *cpuID*)

Sendet ein ACK an einen Controller.

Sendet ein ACK an einen Controller.

Parameter

cpuID ID des Controllers

4.9.3.18 void COM_sendAction (DT_byte *cpuID*)

Sendet das ACTION-Kommando an einen Controller.

Sendet das ACTION-Kommando an einen Controller.

Parameter

cpuID ID des Controllers

4.9.3.19 DT_bool COM_sendAngle (DT_byte *cpuID*, const DT_double *angle*, const DT_byte *config*)

Sendet einen Winkel an einen Controller.

Sendet einen Winkel an einen Controller.

Parameter

cpuID ID des Controllers

angle Zuversendender Winkel

config Parameter, z.B. global, left, right ...

Rückgabe

false, wenn Fehler

4.9.3.20 void COM_sendNAK (DT_byte *cpuID*, DT_byte *errCode*)

Sendet ein NAK an einen Controller.

Sendet ein NAK an einen Controller.

Parameter

cpuID ID des Controllers

errCode Fehlercode

4.9.3.21 DT_bool COM_sendPoint (DT_byte *cpuID*, const DT_point *const *point*, const DT_byte *config*)

Sendet einen Punkt an einen Controller.

Sendet einen Punkt an einen Controller.

Parameter

cpuID ID des Controllers

point Zuversendender Punkt

config Parameter, z.B. global, left, right ...

Rückgabe

false, wenn Fehler

4.9.3.22 DT_bool COM_sendPointAndSpeed (DT_byte *cpuID*, const DT_point *const *point*, const DT_double *speed*, const DT_byte *config*)

Sendet einen Punkt und Anfahrgeschwindigkeit an einen Controller.

Sendet einen Punkt und Anfahrgeschwindigkeit zusammen in einem Packet an einen Controller.

Parameter

cpuID ID des Controllers

point Zuversendender Punkt
speed Anfahrgeschwindigkeit
config Parameter, z.B. global, left, right ...

Rückgabe

false, wenn Fehler

4.10 include/datatypes.h-Dateireferenz

Abstrahiert Datentypen.

```
#include <stdint.h>
#include <stdlib.h>
#include <stdbool.h>
```

Datenstrukturen

- struct **DT_servo**
Datenstruktur zur Speicherung von ID, Soll- und Ist-Wert eines Servos.
- struct **DT_transformation**
Struktur zur vereinfachten Koordinatentransformation.
- struct **DT_leg**
Datenstruktur zur Speicherung von Servodaten bezüglich eines kompletten Beines.
- struct **DT_point**
Datenstruktur zur Speicherung karthesischer Koordinaten.
- struct **DT_vector**
Datenstruktur zur Speicherung eines Vektors.
- struct **DT_lin_func**
Datenstruktur zur Speicherung einer linearen Funktion: $y = mx + n$.
- struct **DT_half_circle**
Datenstruktur zur Speicherung einer Kreisfunktion: $y = \sqrt{r^2 - x^2}$.
- struct **DT_individuum**
Datenstruktur für ein Individuum des Evolutionären Algorithmus zur Startpunktfindung.

Makrodefinitionen

- `#define DT_RESULT_BUFFER_SIZE 128`

Typdefinitionen

- `typedef bool DT_bool`
- `typedef int DT_int`
- `typedef double DT_double`
- `typedef uint8_t DT_byte`
- `typedef uint16_t DT_size`
- `typedef uint8_t DT_type`
- `typedef char DT_char`
- `typedef uint16_t DT_cmd`

4.10.1 Ausführliche Beschreibung

Abstrahiert Datentypen.

4.10.2 Makro-Dokumentation

4.10.2.1 `#define DT_RESULT_BUFFER_SIZE 128`

Größe des Buffers für ein empfangenes Paket.

4.10.3 Dokumentation der benutzerdefinierten Typen

4.10.3.1 `typedef bool DT_bool`

4.10.3.2 `typedef uint8_t DT_byte`

4.10.3.3 `typedef char DT_char`

4.10.3.4 `typedef uint16_t DT_cmd`

4.10.3.5 `typedef double DT_double`

4.10.3.6 `typedef int DT_int`

4.10.3.7 `typedef uint16_t DT_size`

4.10.3.8 `typedef uint8_t DT_type`

4.11 include/dynamixel.h-Dateireferenz

Methoden zur Steuerung der Dynamixel AX-12.

```
#include "datatypes.h"
```

```
#include "usart_driver.h"
```

Makrodefinitionen

- `#define DNX_BRDCAST_ID 0xFE`

Funktionen

- **`DT_byte DNX_send (DT_byte *const, DT_size, DT_byte *const, DT_bool)`**
Versenden von Daten an Dynamixel.
- **`DT_byte DNX_receive (USART_data_t *const, DT_byte *const)`**
USART-Empfangsmethode.
- **`DT_byte DNX_getChecksum (const DT_byte *const, DT_size)`**
Berechnet die Checksum.
- **`DT_bool DNX_setAngle (DT_byte, DT_double, DT_bool)`**
Sendet einen Winkel an Servo.
- **`DT_bool DNX_setAngleAndSpeed (DT_byte id, DT_double angle, DT_double speed, DT_bool regWrite)`**
Sendet Winkel und Speed an Servo.

- **void DNX_setId (DT_byte, DT_byte)**
Vergibt einem Servos eine neue ID (ungetestet).
- **void DNX_setSpeed (DT_byte, DT_byte)**
Setzt die Anfahrgeschwindigkeit eines Servos (unvollendet).
- **DT_bool DNX_setLed (DT_byte, DT_byte)**
Schaltet die LED eines Servos an/aus.
- **DT_double DNX_getAngle (DT_byte)**
Liest den aktuellen Winkel eines Servos aus (unfertig).
- **DT_byte DNX_getSpeed (DT_byte)**
Liest die Anfahrgeschwindigkeit eines Servos aus (unfertig).
- **DT_byte DNX_getLed (DT_byte)**
Liest den Status der LED aus (unfertig).
- **void DNX_getConnectedIDs (DT_leg *const, DT_leg *const)**
Ermittlung der angeschlossenen Dynamixel.
- **void DNX_sendAction (DT_byte)**
Sendet das ACTION-Kommando an einen Servo.

4.11.1 Ausführliche Beschreibung

Methoden zur Steuerung der Dynamixel AX-12.

4.11.2 Makro-Dokumentation

4.11.2.1 #define DNX_BRDCAST_ID 0xFE

4.11.3 Dokumentation der Funktionen

4.11.3.1 DT_double DNX_getAngle (DT_byte *id*)

Liest den aktuellen Winkel eines Servos aus (unfertig).

Parameter

id ID des Servos

Rückgabe

Winkel in Grad

4.11.3.2 DT_byte DNX_getChecksum (const DT_byte *const *packet*, DT_size *l*)

Berechnet die Checksum.

Parameter

packet Paket

l Größe des pakets

Rückgabe

Checksum

4.11.3.3 void DNX_getConnectedIDs (DT_leg *const *leg_r*, DT_leg *const *leg_l*)

Ermittlung der angeschlossenen Dynamixel.

Parameter

leg_r Bein rechts

leg_l Bein links

4.11.3.4 DT_byte DNX_getLed (DT_byte *id*)

Liest den Status der LED aus (unfertig).

Parameter

id ID des Servos

Rückgabe

Wert der LED

4.11.3.5 DT_byte DNX_getSpeed (DT_byte *id*)

Liest die Anfahrgeschwindigkeit eines Servos aus (unfertig).

Parameter

id ID des Servos

Rückgabe

Geschwindigkeit

4.11.3.6 DT_byte DNX_receive (USART_data_t *const *usart_data*, DT_byte *const *dest*)

USART-Empfangsmethode.

Diese Methode liest den jeweiligen USART-Buffer aus und prüft, ob ein vollständiges Paket gemäß des Dynamixel-Protokoll empfangen wurde.

Parameter

usart_data USART

dest Byte-Array für Antwort-Paket

Rückgabe

Länge des Antwortpakets

4.11.3.7 DT_byte DNX_send (DT_byte *const *packet*, DT_size *l*, DT_byte *const *result*, DT_bool *hasResponse*)

Versenden von Daten an Dynamixel.

Blockierendes Senden mit gleichzeitigem Empfangen der Antwort.

Parameter

packet Zuversendendes Paket

l Größe des Pakets

result Zielfeld für Antwort

hasResponse Wartet auf eine Antwort, wenn true

Rückgabe

Größe der empfangenen Antwort

4.11.3.8 void DNX_sendAction (DT_byte *id*)

Sendet das ACTION-Kommando an einen Servo.

Sendet das ACTION-Kommando an einen Servo.

Parameter

id ID des Servo

4.11.3.9 DT_bool DNX_setAngle (DT_byte *id*, DT_double *value*, DT_bool *regWrite*)

Sendet einen Winkel an Servo.

Parameter

id ID des Servos

value Winkel in Grad

regWrite Winkel wird in Puffer des Servos gespeichert und erst bei ACTION angefahren, wenn true

4.11.3.10 DT_bool DNX_setAngleAndSpeed (DT_byte *id*, DT_double *angle*, DT_double *speed*, DT_bool *regWrite*)

Sendet Winkel und Speed an Servo.

Parameter

id ID des Servos

angle Winkel in Grad

speed Anfahrgeschwindigkeit

regWrite Werte werden in Puffer des Servos gespeichert und erst bei ACTION ausgeführt, wenn true

4.11.3.11 void DNX_setId (DT_byte *idOld*, DT_byte *idNew*)

Vergibt einem Servo eine neue ID (ungetestet).

Parameter

idOld ID des zu verändernden Servos

idNew Zusetzende ID

4.11.3.12 DT_bool DNX_setLed (DT_byte *id*, DT_byte *value*)

Schaltet die LED eines Servos an/aus.

Parameter

id ID des Servos

value Wert für LED (0x00 / 0x01)

4.11.3.13 void DNX_setSpeed (DT_byte *id*, DT_byte *speed*)

Setzt die Anfahrgeschwindigkeit eines Servos (unvollendet).

Parameter

id ID des Servos

speed Geschwindigkeit

4.12 include/evolutionaryAlgorithm.h-Dateireferenz

Evolutionärer Algorithmus zur Startpunktfindung.

```
#include "datatypes.h"
```

Funktionen

- **DT_individuum evolutionaryAlgorithm** (const DT_int popsize, const DT_int generations, DT_vector *const)
- **DT_point getPointFromIndividuum** (DT_individuum *)
- **DT_point getIsectFromIndividuum** (DT_individuum *)

4.12.1 Ausführliche Beschreibung

Evolutionärer Algorithmus zur Startpunktfindung. Startpunktfindung auf Basis eines Evolutionären Algorithmus.

4.12.2 Dokumentation der Funktionen

4.12.2.1 DT_individuum evolutionaryAlgorithm (const DT_int *popsize*, const DT_int *generations*, DT_vector * *const*)

4.12.2.2 DT_point getIsectFromIndividuum (DT_individuum *)

4.12.2.3 DT_point getPointFromIndividuum (DT_individuum *)

4.13 include/evolutionaryHelper.h-Dateireferenz

Hilfsfunktion für Evolutionären Algorithmus.

```
#include "datatypes.h"
```

Makrodefinitionen

- **#define Z** -129.1041

Funktionen

- **DT_bool** `isInArea` (`const DT_point *`)
- **DT_double** `scorePoint` (`DT_vector *const`, `const DT_point *const`, `DT_point *`)
- `void` `bubblesort` (`DT_individuum *`, `const DT_int`)
- `void` `initEvoAlg` ()
- **DT_double** `max` (`DT_double`, `DT_double`)
- **DT_double** `min` (`DT_double`, `DT_double`)
- **DT_double** `getDistance` (`const DT_point *const`, `const DT_point *const`)
- `void` `getFunctionOfPoints` (`DT_lin_func *`, `const DT_point *const`, `const DT_point *const`)

4.13.1 Ausführliche Beschreibung

Hilfsfunktion für Evolutionären Algorithmus. Mathematische Funktionen.

4.13.2 Makro-Dokumentation

4.13.2.1 #define Z -129.1041

4.13.3 Dokumentation der Funktionen

4.13.3.1 void bubblesort (DT_individuum *, const DT_int)

4.13.3.2 DT_double getDistance (const DT_point * *const*, const DT_point * *const*)

4.13.3.3 void getFunctionOfPoints (DT_lin_func *, const DT_point * *const*, const DT_point * *const*)

4.13.3.4 void initEvoAlg ()

4.13.3.5 DT_bool isInArea (const DT_point *)

4.13.3.6 DT_double max (DT_double , DT_double)

4.13.3.7 DT_double min (DT_double , DT_double)

4.13.3.8 DT_double scorePoint (DT_vector * *const*, const DT_point * *const*, DT_point *)

4.14 include/kinematics.h-Dateireferenz

Lösungsmethoden der Kinematik.

```
#include "datatypes.h"
```

Makrodefinitionen

- `#define KIN_ROWS 4`
- `#define KIN_COLUMNS 4`

Funktionen

- `void KIN_setTransMat (DT_leg *const)`
Liefert fuer ein Bein die Struktur zur Koordinatentransformation.
- `void KIN_calcDH (const DT_leg *const, DT_double **)`
Lösung des kinematischen Problems.
- `DT_bool KIN_calcServos (const DT_point *const, DT_leg *const)`
Lösung des inversen kinematischen Problems.
- `DT_point KIN_calcLocalPoint (const DT_point *const, const DT_transformation *const)`
Transformiert einen Punkt in das Roboterkoordinatensystem.
- `DT_bool KIN_makeMovement (DT_leg *leg_l, DT_leg *leg_r)`

4.14.1 Ausführliche Beschreibung

Lösungsmethoden der Kinematik. Lösungsmethoden der Kinematik speziell für den CM-Bot.

4.14.2 Makro-Dokumentation

4.14.2.1 `#define KIN_COLUMNS 4`

4.14.2.2 `#define KIN_ROWS 4`

4.14.3 Dokumentation der Funktionen

4.14.3.1 `void KIN_calcDH (const DT_leg *const leg, DT_double ** dh03)`

Lösung des kinematischen Problems.

Lösung der Denavit-Hartenberg-Transformation.

Parameter

leg Bein mit den Soll-Winkel der Gelenke

dh03 Zielmatrix für die Lösung

4.14.3.2 DT_point KIN_calcLocalPoint (const DT_point *const *p*, const DT_transformation *const *trans*)

Transformiert einen Punkt in das Roboterkoordinatensystem.

Transformiert einen Punkt des Weltkoordinatensystems in das Roboterkoordinatensystem eines Beines. Berechnet nur eine x-y-Verschiebung und eine Rotation von 0/180 Grad um die z-Achse.

Parameter

p Punkt im Weltkoordinatensystem

trans Struktur mit Informationen fuer die Koordinatentransformation

Rückgabe

Punkt im Roboterkoordinatensystem eines Beines

4.14.3.3 DT_bool KIN_calcServos (const DT_point *const *p*, DT_leg *const *leg*)

Lösung des inversen kinematischen Problems.

Lösung des inversen kinematischen Problems mit Hilfe eines geometrischen Verfahrens mit leichten Einschränkungen.

Parameter

p Punkt (Roboterkoordinate)

leg Bein für die zusetzenden Winkel

Rückgabe

true, wenn Berechnung erfolgreich

4.14.3.4 DT_bool KIN_makeMovement (DT_leg * *leg_l*, DT_leg * *leg_r*)

4.14.3.5 void KIN_setTransMat (DT_leg *const *leg*)

Liefert fuer ein Bein die Struktur zur Koordinatentransformation.

Liefert fuer ein Bein die Struktur zur Koordinatentransformation.

Parameter

leg Bein

4.15 include/movement.h-Dateireferenz

Stellt Grundfunktionen für einen Laufalgorithmus bereit.

```
#include "datatypes.h"
```

Makrodefinitionen

- `#define MV_DST_Y 208.5`
- `#define MV_DST_X 168.5`

Funktionen

- **void MV_action (DT_leg *const, DT_leg *const)**
Sendet das ACTION-Kommando an ein Bein.
- **void MV_slave (DT_byte, DT_leg *const, DT_leg *const)**
Standard-Methode für einen Slave-Controller.
- **void MV_slaveStatus (const DT_byte *const, const DT_size)**
Antwortet auf eine Status-Anfrage eines Masters. (Slave).
- **void MV_slavePoint (DT_leg *const, DT_leg *const, const DT_byte *const, DT_size)**
Führt die benötigten Aktionen für einen empfangenen Punkt aus. (Slave).
- **void MV_slavePointAndSpeed (DT_leg *const, DT_leg *const, const DT_byte *const, DT_size)**
Führt die benötigten Aktionen für einen empfangenen Punkt und Anfahrgeschwindigkeit aus. (Slave).
- **void MV_slaveAngle (DT_leg *const, DT_leg *const, const DT_byte *const, DT_size)**
Führt die benötigten Aktionen für einen empfangenen Winkel aus. (Slave).
- **DT_bool MV_point (DT_leg *const, const DT_point *const, DT_bool)**
Berechnet Winkel anhand des Punktes für die Servos und versendet diese.
- **DT_bool MV_pointAndSpeed (DT_leg *const, const DT_point *const, const DT_double, DT_bool)**
Berechnet Winkel anhand des Punktes für die Servos und versendet diese zusammen mit der Anfahrgeschwindigkeit.
- **void MV_masterCheckAlive ()**
Prüft ob die Slaves alive sind. (Master).
- **void MV_doInitPosition (DT_leg *const, DT_leg *const)**
Fährt das rechte und linke Bein in eine Startposition.
- **void MV_switchLegs (DT_byte *side, DT_byte *master_dwn, DT_byte *master_up, DT_byte *slave_dwn, DT_byte *slave_up)**
Wechselt die aktiven und passiven Beine.

- **DT_point** **MV_getPntForCpuSide** (const **DT_point** *const, const **DT_byte**, const **DT_byte**)

Rechnet einen lokalen Punkt mit Bezug auf das mittlere Bein in einen globalen Punkt für einen Controller um.

4.15.1 Ausführliche Beschreibung

Stellt Grundfunktionen für einen Laufalgorithmus bereit.

4.15.2 Makro-Dokumentation

4.15.2.1 **#define** **MV_DST_X** 168.5

4.15.2.2 **#define** **MV_DST_Y** 208.5

4.15.3 Dokumentation der Funktionen

4.15.3.1 **void** **MV_action** (**DT_leg** *const *leg_r*, **DT_leg** *const *leg_l*)

Sendet das ACTION-Kommando an ein Bein.

Sendet das ACTION-Kommando an alle Servos eines Beines.

Parameter

leg_r rechtes Bein

leg_l linkes Bein

4.15.3.2 **void** **MV_doInitPosition** (**DT_leg** *const *leg_r*, **DT_leg** *const *leg_l*)

Führt das rechte und linke Bein in eine Startposition.

Führt das rechte und linke Bein in eine Startposition.

Parameter

leg_r rechtes Bein

leg_l linkes Bein

4.15.3.3 **DT_point** **MV_getPntForCpuSide** (const **DT_point** *const *p*, const **DT_byte** *cpuld*, const **DT_byte** *side*)

Rechnet einen lokalen Punkt mit Bezug auf das mittlere Bein in einen globalen Punkt für einen Controller um.

Rechnet einen lokalen Punkt mit Bezug auf das mittlere Bein in einen globalen Punkt für einen Controller um.

Parameter

p lokaler Punkt
cpuId ID des Controllers
side Seite (Links/Rechts)

Rückgabe

Globaler Punkt

4.15.3.4 void MV_masterCheckAlive ()

Prüft ob die Slaves alive sind. (Master).

Prüft blockierend ob die Slaves alive sind.

4.15.3.5 DT_bool MV_point (DT_leg *const leg, const DT_point *const point, DT_bool isGlobal)

Berechnet Winkel anhand des Punktes für die Servos und versendet diese.

Berechnet Winkel anhand des Punktes für die Servos und versendet diese.

Parameter

leg Bein
point Punkt
isGlobal Weltkoordinate, wenn true

4.15.3.6 DT_bool MV_pointAndSpeed (DT_leg *const leg, const DT_point *const point, const DT_double speed, DT_bool isGlobal)

Berechnet Winkel anhand des Punktes für die Servos und versendet diese zusammen mit der Anfahrgeschwindigkeit.

Berechnet Winkel anhand des Punktes für die Servos und versendet diese zusammen mit der Anfahrgeschwindigkeit.

Parameter

leg Bein
point Punkt
speed Anfahrgeschwindigkeit
isGlobal Weltkoordinate, wenn true

4.15.3.7 void MV_slave (DT_byte *cpuID*, DT_leg *const *leg_r*, DT_leg *const *leg_l*)

Standard-Methode für einen Slave-Controller.

Standard-Methode für einen Slave-Controller. Nimmt Befehle eines Masters entgegen und führt die entsprechenden Aktionen aus.

Parameter

cpuID ID des Controllers auf dem die Methode ausgeführt wird

leg_r rechtes Bein

leg_l linkes Bein

4.15.3.8 void MV_slaveAngle (DT_leg *const *leg_r*, DT_leg *const *leg_l*, const DT_byte *const *result*, DT_size *len*)

Führt die benötigten Aktionen für einen empfangenen Winkel aus. (Slave).

Führt die benötigten Aktionen für einen empfangenen Winkel aus. (Slave)

Parameter

leg_r rechtes Bein

leg_l linkes Bein

result Anfrage-Paket zum Auswerten

len Länge des Pakets

4.15.3.9 void MV_slavePoint (DT_leg *const *leg_r*, DT_leg *const *leg_l*, const DT_byte *const *result*, DT_size *len*)

Führt die benötigten Aktionen für einen empfangenen Punkt aus. (Slave).

Führt die benötigten Aktionen für einen empfangenen Punkt aus. (Slave)

Parameter

leg_r rechtes Bein

leg_l linkes Bein

result Anfrage-Paket zum Auswerten

len Länge des Pakets

4.15.3.10 void MV_slavePointAndSpeed (DT_leg *const *leg_r*, DT_leg *const *leg_l*, const DT_byte *const *result*, DT_size *len*)

Führt die benötigten Aktionen für einen empfangenen Punkt und Anfahrsgeschwindigkeit aus. (Slave).

Führt die benötigten Aktionen für einen empfangenen Punkt und Anfahrsgeschwindigkeit aus. (Slave)

Parameter

leg_r rechtes Bein

leg_l linkes Bein

result Anfrage-Paket zum Auswerten

len Länge des Pakets

4.15.3.11 void MV_slaveStatus (const DT_byte *const *result*, const DT_size *len*)

Antwortet auf eine Status-Anfrage eines Masters. (Slave).

Antwortet auf eine Status-Anfrage eines Masters. (Slave)

Parameter

result Anfrage-Paket zum Auswerten

len Länge des Pakets

4.15.3.12 void MV_switchLegs (DT_byte * *side*, DT_byte * *master_dwn*, DT_byte * *master_up*, DT_byte * *slave_dwn*, DT_byte * *slave_up*)

Wechselt die aktiven und passiven Beine.

Wechselt die aktiven und passiven Beine (Oben/Unten). Übergebene Variablen können in den jeweiligen Laufalgorithmus ausgewertet werden.

Parameter

side Aktive Seite (Seite mit Bodenkontakt), mittleres Beinpaar als Referenz

master_dwn Seite (Links/Rechts) für den Master mit Bodenkontakt

master_up Seite (Links/Rechts) für den Master ohne Bodenkontakt

slave_dwn Seite (Links/Rechts) für den Slave mit Bodenkontakt

slave_up Seite (Links/Rechts) für den Master ohne Bodenkontakt

4.16 include/remote.h-Dateireferenz

Methoden zur Steuerung durch den RC-100 Remote Controller.

```
#include "datatypes.h"
#include "usart_driver.h"
```

Funktionen

- **DT_cmd RMT_getCommand ()**
Liest empfangene Befehle vom Remote-Controller aus.
- **DT_byte RMT_receive (USART_data_t *const, DT_byte *const)**
USART-Empfangsmethode für Remote-Controller.
- **DT_bool RMT_NonPressed (DT_cmd)**
Kein Taster gedrückt.
- **DT_bool RMT_isUpPressed (DT_cmd)**
Ist Taster Up gedrückt.
- **DT_bool RMT_isDownPressed (DT_cmd)**
Ist Taster Down gedrückt.
- **DT_bool RMT_isLeftPressed (DT_cmd)**
Ist Taster Left gedrückt.
- **DT_bool RMT_isRightPressed (DT_cmd)**
Ist Taster Right gedrückt.
- **DT_bool RMT_isButton1Pressed (DT_cmd)**
Ist Taster 1 gedrückt.
- **DT_bool RMT_isButton2Pressed (DT_cmd)**
Ist Taster 2 gedrückt.
- **DT_bool RMT_isButton3Pressed (DT_cmd)**
Ist Taster 3 gedrückt.
- **DT_bool RMT_isButton4Pressed (DT_cmd)**
Ist Taster 4 gedrückt.
- **DT_bool RMT_isButton5Pressed (DT_cmd)**
Ist Taster 5 gedrückt.
- **DT_bool RMT_isButton6Pressed (DT_cmd)**
Ist Taster 6 gedrückt.

4.16.1 Ausführliche Beschreibung

Methoden zur Steuerung durch den RC-100 Remote Controller.

4.16.2 Dokumentation der Funktionen

4.16.2.1 DT_cmd RMT_getCommand ()

Liest empfangene Befehle vom Remote-Controller aus.

Instruction aus dem High- und Low-Teil zusammensetzen Bsp: Paket für B_2: 0 1 2 3 4
5 ... FF;55;20;DF;00;FF;FF;55;00;FF;00;FF; LL HH Eigentliche Information: 0x0020
d.h. H = Paket[4] L = Paket[2]

4.16.2.2 DT_bool RMT_isButton1Pressed (DT_cmd cmd)

Ist Taster 1 gedrückt.

Achtung: cmd muss zunächst durch getCommand abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.16.2.3 DT_bool RMT_isButton2Pressed (DT_cmd cmd)

Ist Taster 2 gedrückt.

Achtung: cmd muss zunächst durch getCommand abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.16.2.4 DT_bool RMT_isButton3Pressed (DT_cmd cmd)

Ist Taster 3 gedrückt.

Achtung: cmd muss zunächst durch getCommand abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.16.2.5 DT_bool RMT_isButton4Pressed (DT_cmd *cmd*)

Ist Taster 4 gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.16.2.6 DT_bool RMT_isButton5Pressed (DT_cmd *cmd*)

Ist Taster 5 gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.16.2.7 DT_bool RMT_isButton6Pressed (DT_cmd *cmd*)

Ist Taster 6 gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.16.2.8 DT_bool RMT_isDownPressed (DT_cmd *cmd*)

Ist Taster Down gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.16.2.9 DT_bool RMT_isLeftPressed (DT_cmd *cmd*)

Ist Taster Left gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.16.2.10 DT_bool RMT_isRightPressed (DT_cmd *cmd*)

Ist Taster Right gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.16.2.11 DT_bool RMT_isUpPressed (DT_cmd *cmd*)

Ist Taster Up gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.16.2.12 DT_bool RMT_NonPressed (DT_cmd *cmd*)

Kein Taster gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

**4.16.2.13 DT_byte RMT_receive (USART_data_t *const *usart_data*,
DT_byte *const *dest*)**

USART-Empfangsmethode für Remote-Controller.

Diese Methode liest den Remote-USART-Buffer aus und prüft, ob ein vollständiges Paket empfangen wurde.

Parameter

usart_data USART

dest Byte-Array für Antwort-Paket

Rückgabe

Länge des Antwortpakets

4.17 include/usart_driver.h-Dateireferenz

XMEGA USART driver header file.

```
#include "avr_compiler.h"
```

```
#include "datatypes.h"
```

Datenstrukturen

- struct **USART_Buffer**
- struct **Usart_and_buffer**

Struct used when interrupt driven driver is used.

Makrodefinitionen

- **#define USART_RX_BUFFER_SIZE** 128
- **#define USART_TX_BUFFER_SIZE** 128
- **#define USART_RX_BUFFER_MASK** (USART_RX_BUFFER_SIZE - 1)
- **#define USART_TX_BUFFER_MASK** (USART_TX_BUFFER_SIZE - 1)
- **#define USART_Format_Set**(_usart, _charSize, _parityMode, _twoStopBits)
Macro that sets the USART frame format.
- **#define USART_Baudrate_Set**(_usart, _bseIValue, _bScaleFactor)
Set USART baud rate.
- **#define USART_Rx_Enable**(_usart) ((_usart)->CTRLB |= USART_RXEN_bm)
Enable USART receiver.
- **#define USART_Rx_Disable**(_usart) ((_usart)->CTRLB &= ~USART_RXEN_bm)
Disable USART receiver.
- **#define USART_Tx_Enable**(_usart) ((_usart)->CTRLB |= USART_TXEN_bm)
Enable USART transmitter.
- **#define USART_Tx_Disable**(_usart) ((_usart)->CTRLB &= ~USART_TXEN_bm)
Disable USART transmitter.
- **#define USART_RxdInterruptLevel_Set**(_usart, _rxIntLevel) ((_usart)->CTRLA = ((_usart)->CTRLA & ~USART_RXCINTLVL_gm) | _rxIntLevel)
Set USART RXD interrupt level.
- **#define USART_TxdInterruptLevel_Set**(_usart, _txIntLevel) ((_usart)->CTRLA = ((_usart)->CTRLA & ~USART_TXCINTLVL_gm) | _txIntLevel)
Set USART TXD interrupt level.
- **#define USART_DreInterruptLevel_Set**(_usart, _dreIntLevel) ((_usart)->CTRLA = ((_usart)->CTRLA & ~USART_DREINTLVL_gm) | _dreIntLevel)
Set USART DRE interrupt level.
- **#define USART_SetMode**(_usart, _usartMode) ((_usart)->CTRLC = ((_usart)->CTRLC & (~USART_CMODOE_gm)) | _usartMode)
Set the mode the USART run in.

- **#define USART_IsTXDataRegisterEmpty(_usart)** (((_usart)->STATUS & USART_DREIF_bm) != 0)
Check if data register empty flag is set.
- **#define USART_PutChar(_usart, _data)** ((_usart)->DATA = _data)
Put data (5-8 bit character).
- **#define USART_IsRXComplete(_usart)** (((_usart)->STATUS & USART_RXCIF_bm) != 0)
Checks if the RX complete interrupt flag is set.
- **#define USART_GetChar(_usart)** ((_usart)->DATA)
Get received data (5-8 bit character).

Typdefinitionen

- **typedef struct USART_Buffer USART_Buffer_t**
- **typedef struct Usart_and_buffer USART_data_t**
Struct used when interrupt driven driver is used.

Funktionen

- **void USART_InterruptDriver_Initialize (USART_data_t *usart_data, USART_t *usart, USART_DREINTLVL_t dreIntLevel)**
Initializes buffer and selects what USART module to use.
- **void USART_InterruptDriver_DreInterruptLevel_Set (USART_data_t *usart_data, USART_DREINTLVL_t dreIntLevel)**
Set USART DRE interrupt level.
- **bool USART_TXBuffer_FreeSpace (USART_data_t *usart_data)**
Test if there is data in the transmitter software buffer.
- **bool USART_TXBuffer_PutByte (USART_data_t *usart_data, uint8_t data)**
Put data (5-8 bit character).
- **bool USART_RXBufferData_Available (USART_data_t *usart_data)**
Test if there is data in the receive software buffer.
- **uint8_t USART_RXBuffer_GetByte (USART_data_t *usart_data)**
Get received data (5-8 bit character).
- **bool USART_RXComplete (USART_data_t *usart_data)**

RX Complete Interrupt Service Routine.

- void **USART_DataRegEmpty** (USART_data_t *usart_data)
Data Register Empty Interrupt Service Routine.
- DT_bool **USART_RXBuffer_checkPointerDiff** (DT_byte, DT_byte, DT_byte)
Überprüft Differenzen der Empfangs-Pointer.
- void **USART_NineBits_PutChar** (USART_t *usart, uint16_t data)
Put data (9 bit character).
- uint16_t **USART_NineBits_GetChar** (USART_t *usart)
Get received data (9 bit character).

4.17.1 Ausführliche Beschreibung

XMEGA USART driver header file. This file contains the function prototypes and enumerator definitions for various configuration parameters for the XMEGA USART driver.

The driver is not intended for size and/or speed critical code, since most functions are just a few lines of code, and the function call overhead would decrease code performance. The driver is intended for rapid prototyping and documentation purposes for getting started with the XMEGA ADC module.

For size and/or speed critical code, it is recommended to copy the function contents directly into your application instead of making a function call.

Application note:

AVR1307: Using the XMEGA USART

Documentation

For comprehensive code documentation, supported compilers, compiler settings and supported devices see readme.html

Autor

Atmel Corporation: <http://www.atmel.com>
Support email: avr@atmel.com

Revision:

481

Date:

2007-03-06 10:12:53 +0100 (ty, 06 mar 2007)

Copyright (c) 2008, Atmel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of ATMEL may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.17.2 Makro-Dokumentation

4.17.2.1 #define USART_Baudrate_Set(_usart, _bselValue, _bScaleFactor)

Wert:

```
(_usart)->BAUDCTRLA =(uint8_t)_bselValue;
    \
    (_usart)->BAUDCTRLB =(_bScaleFactor << USART_BSCALE0_bp) | (_bselValue >> 8
    )
```

Set USART baud rate.

Sets the USART's baud rate register.

UBRR_Value : Value written to UBRR ScaleFactor : Time Base Generator Scale Factor

Equation for calculation of BSEL value in asynchronous normal speed mode: If ScaleFactor ≥ 0 BSEL = $((I/O \text{ clock frequency})/(2^{(ScaleFactor*16*Baudrate)}))-1$ If ScaleFactor < 0 BSEL = $(1/(2^{(ScaleFactor*16)}))*((I/O \text{ clock frequency})/Baudrate)-1$

Zu beachten

See XMEGA manual for equations for calculation of BSEL value in other modes.

Parameter

- _usart*** Pointer to the USART module.
- _bseIValue*** Value to write to BSEL part of Baud control register. Use uint16_t type.
- _bScaleFactor*** USART baud rate scale factor. Use uint8_t type

4.17.2.2 `#define USART_DreInterruptLevel_Set(_usart, _dreIntLevel) (_usart)->CTRLA = ((_usart)->CTRLA & ~USART_DREINTLVL_gm) | _dreIntLevel`

Set USART DRE interrupt level.

Sets the interrupt level on Data Register interrupt.

Parameter

- _usart*** Pointer to the USART module.
- _dreIntLevel*** Interrupt level of the DRE interrupt. Use USART_DREINTLVL_t type.

4.17.2.3 `#define USART_Format_Set(_usart, _charSize, _parityMode, _twoStopBits)`

Wert:

```
(_usart)->CTRLC = (uint8_t) _charSize | _parityMode | \
                (_twoStopBits ? USART_SBMODE_bm : 0)
```

Macro that sets the USART frame format.

Sets the frame format, Frame Size, parity mode and number of stop bits.

Parameter

- _usart*** Pointer to the USART module
- _charSize*** The character size. Use USART_CHSIZE_t type.
- _parityMode*** The parity Mode. Use USART_PMODE_t type.
- _twoStopBits*** Enable two stop bit mode. Use bool type.

4.17.2.4 `#define USART_GetChar(_usart) ((_usart)->DATA)`

Get received data (5-8 bit character).

This macro reads out the RX register. Use the macro USART_RX_Complete to check if anything is received.

Parameter

_usart The USART module.

Rückgabewerte

Received data.

4.17.2.5 `#define USART_IsRXComplete(_usart) (((_usart)->STATUS & USART_RXCIF_bm) != 0)`

Checks if the RX complete interrupt flag is set.

Checks if the RX complete interrupt flag is set.

Parameter

_usart The USART module.

4.17.2.6 `#define USART_IsTXDataRegisterEmpty(_usart) (((_usart)->STATUS & USART_DREIF_bm) != 0)`

Check if data register empty flag is set.

Parameter

_usart The USART module.

4.17.2.7 `#define USART_PutChar(_usart, _data) ((_usart)->DATA = _data)`

Put data (5-8 bit character).

Use the macro USART_IsTXDataRegisterEmpty before using this function to put data to the TX register.

Parameter

_usart The USART module.

_data The data to send.

4.17.2.8 `#define USART_RX_BUFFER_MASK (USART_RX_BUFFER_SIZE - 1)`

4.17.2.9 `#define USART_RX_BUFFER_SIZE 128`

4.17.2.10 `#define USART_Rx_Disable(_usart) ((_usart)->CTRLB &= ~USART_RXEN_bm)`

Disable USART receiver.

Parameter

_usart Pointer to the USART module.

4.17.2.11 `#define USART_Rx_Enable(_usart) ((_usart)->CTRLB |= USART_RXEN_bm)`

Enable USART receiver.

Parameter

_usart Pointer to the USART module

4.17.2.12 `#define USART_RxdInterruptLevel_Set(_usart, _rxdIntLevel) ((_usart)->CTRLA = ((_usart)->CTRLA & ~USART_RXCINTLVL_gm) | _rxdIntLevel)`

Set USART RXD interrupt level.

Sets the interrupt level on RX Complete interrupt.

Parameter

_usart Pointer to the USART module.

_rxdIntLevel Interrupt level of the RXD interrupt. Use USART_RXCINTLVL_t type.

4.17.2.13 `#define USART_SetMode(_usart, _usartMode) ((_usart)->CTRLC = ((_usart)->CTRLC & (~USART_CMODE_gm)) | _usartMode)`

Set the mode the USART run in.

Set the mode the USART run in. The default mode is asynchronous mode.

Parameter

_usart Pointer to the USART module register section.

_usartMode Selects the USART mode. Use USART_CMODE_t type.

USART modes:

- 0x0 : Asynchronous mode.
- 0x1 : Synchronous mode.
- 0x2 : IrDA mode.
- 0x3 : Master SPI mode.

4.17.2.14 `#define USART_TX_BUFFER_MASK (USART_TX_BUFFER_SIZE - 1)`

4.17.2.15 `#define USART_TX_BUFFER_SIZE 128`

4.17.2.16 `#define USART_Tx_Disable(_usart) ((_usart)->CTRLB &= ~USART_TXEN_bm)`

Disable USART transmitter.

Parameter

`_usart` Pointer to the USART module.

4.17.2.17 `#define USART_Tx_Enable(_usart) ((_usart)->CTRLB |= USART_TXEN_bm)`

Enable USART transmitter.

Parameter

`_usart` Pointer to the USART module.

4.17.2.18 `#define USART_TxdInterruptLevel_Set(_usart, _txdIntLevel) (_usart)->CTRLA = ((_usart)->CTRLA & ~USART_TXCINTLVL_gm) | _txdIntLevel`

Set USART TXD interrupt level.

Sets the interrupt level on TX Complete interrupt.

Parameter

`_usart` Pointer to the USART module.

`_txdIntLevel` Interrupt level of the TXD interrupt. Use USART_TXCINTLVL_t type.

4.17.3 Dokumentation der benutzerdefinierten Typen

4.17.3.1 `typedef struct USART_Buffer USART_Buffer_t`

4.17.3.2 `typedef struct Usart_and_buffer USART_data_t`

Struct used when interrupt driven driver is used.

Struct containing pointer to a usart, a buffer and a location to store Data register interrupt level temporary.

4.17.4 Dokumentation der Funktionen

4.17.4.1 void USART_DataRegEmpty (USART_data_t * *usart_data*)

Data Register Empty Interrupt Service Routine.

Data Register Empty Interrupt Service Routine. Transmits one byte from TX software buffer. Disables DRE interrupt if buffer is empty. Argument is pointer to USART (USART_data_t).

Parameter

usart_data The USART_data_t struct instance.

4.17.4.2 void USART_InterruptDriver_DreInterruptLevel_Set (USART_data_t * *usart_data*, USART_DREINTLVL_t *dreIntLevel*)

Set USART DRE interrupt level.

Set the interrupt level on Data Register interrupt.

Zu beachten

Changing the DRE interrupt level in the interrupt driver while it is running will not change the DRE interrupt level in the USART before the DRE interrupt have been disabled and enabled again.

Parameter

usart_data The USART_data_t struct instance

dreIntLevel Interrupt level of the DRE interrupt.

4.17.4.3 void USART_InterruptDriver_Initialize (USART_data_t * *usart_data*, USART_t * *usart*, USART_DREINTLVL_t *dreIntLevel*)

Initializes buffer and selects what USART module to use.

Initializes receive and transmit buffer and selects what USART module to use, and stores the data register empty interrupt level.

Parameter

usart_data The USART_data_t struct instance.

usart The USART module.

dreIntLevel Data register empty interrupt level.

4.17.4.4 uint16_t USART_NineBits_GetChar (USART_t * *usart*)

Get received data (9 bit character).

This function reads out the received 9 bit character (uint16_t). Use the function USART_IsRXComplete to check if anything is received.

Parameter

usart The USART module.

Rückgabewerte

Received data.

4.17.4.5 void USART_NineBits_PutChar (USART_t * *usart*, uint16_t *data*)

Put data (9 bit character).

Use the function USART_IsTXDataRegisterEmpty before using this function to put 9 bit character to the TX register.

Parameter

usart The USART module.

data The data to send.

4.17.4.6 DT_bool USART_RXBuffer_checkPointerDiff (DT_byte *tail*, DT_byte *head*, DT_byte *diff*)

Überprüft Differenzen der Empfangs-Pointer.

Parameter

tail Tail-Pointer

head Head-Pointer

diff Differenz

Rückgabe

Bool

4.17.4.7 uint8_t USART_RXBuffer_GetByte (USART_data_t * *usart_data*)

Get received data (5-8 bit character).

The function USART_RXBufferData_Available should be used before this function is used to ensure that data is available.

Returns data from RX software buffer.

Parameter

usart_data The USART_data_t struct instance.

Rückgabe

Received data.

4.17.4.8 bool USART_RXBufferData_Available (USART_data_t * usart_data)

Test if there is data in the receive software buffer.

This function can be used to test if there is data in the receive software buffer.

Parameter

usart_data The USART_data_t struct instance

Rückgabewerte

true There is data in the receive buffer.

false The receive buffer is empty.

4.17.4.9 bool USART_RXComplete (USART_data_t * usart_data)

RX Complete Interrupt Service Routine.

RX Complete Interrupt Service Routine. Stores received data in RX software buffer.

Parameter

usart_data The USART_data_t struct instance.

4.17.4.10 bool USART_TXBuffer_FreeSpace (USART_data_t * usart_data)

Test if there is data in the transmitter software buffer.

This function can be used to test if there is free space in the transmitter software buffer.

Parameter

usart_data The USART_data_t struct instance.

Rückgabewerte

true There is data in the receive buffer.

false The receive buffer is empty.

4.17.4.11 `bool USART_TXBuffer_PutByte (USART_data_t * usart_data, uint8_t data)`

Put data (5-8 bit character).

Stores data byte in TX software buffer and enables DRE interrupt if there is free space in the TX software buffer.

Parameter

usart_data The USART_data_t struct instance.

data The data to send.

4.18 include/utils.h-Dateireferenz

Verschiedene Hilfsmethoden.

```
#include "datatypes.h"
```

Makrodefinitionen

- `#define UTL_DEG 1`
- `#define UTL_RAD 0`
- `#define DEBUG_ON debug`
- `#define DEBUG(output) UTL_printDebug output;`
- `#define DEBUG_BYTE(output) UTL_printDebugByte output;`

Funktionen

- `void UTL_printMatrix (const DT_double **const, DT_size, DT_size)`
Gibt eine Matrix auf stdo aus.
- `void UTL_printLeg (const DT_leg *const, DT_type)`
Gibt die Soll-Winkel eines Beines auf stdo aus.
- `void UTL_printPoint (const DT_point *const)`
Gibt einen Punkt auf stdo aus.
- `DT_double UTL_getRadiant (DT_double)`
Umrechnung in das Bogenmaß.
- `DT_double UTL_getDegree (DT_double)`
Umrechnung in das Gradmaß.
- `DT_point UTL_getPointOfDH (const DT_double **const)`
Extrahiert den Punkt aus der DH-Matrix.

- void **UTL_printDebug** (const **DT_char** *const, **DT_size**)
Debug-Ausgabe.
- void **UTL_printDebugByte** (const **DT_byte** *const, **DT_size**)
Debug-Ausgabe von Bytes.
- **DT_byte** **UTL_byteToHexChar** (**DT_char** *const, const **DT_byte** *const, **DT_size**)
Konvertierung eines Bytes in das Hexadezimal-Format.
- void **UTL_wait** (**DT_size**)
Abstraktion von einer Pause/Delay.

4.18.1 Ausführliche Beschreibung

Verschiedene Hilfsmethoden. Stellt verschiedene Hilfsmethoden für allgemeinen Gebrauch zur Verfügung.

4.18.2 Makro-Dokumentation

4.18.2.1 **#define** **DEBUG**(*output*) **UTL_printDebug** *output*;

4.18.2.2 **#define** **DEBUG_BYTE**(*output*) **UTL_printDebugByte** *output*;

4.18.2.3 **#define** **DEBUG_ON** *debug*

4.18.2.4 **#define** **UTL_DEG** 1

4.18.2.5 **#define** **UTL_RAD** 0

4.18.3 Dokumentation der Funktionen

4.18.3.1 **DT_byte** **UTL_byteToHexChar** (**DT_char** *const *dest*, const **DT_byte** *const *src*, **DT_size** *size*)

Konvertierung eines Bytes in das Hexadezimal-Format.

Parameter

dest Pointer auf das Zielfeld

src Pointer auf das Quellfeld

size Größe des Quellfelds

4.18.3.2 DT_double UTL_getDegree (DT_double *radiant*)

Umrechnung in das Gradmaß.

Rechnet einen Winkel in das Gradmaß um.

Parameter

radiant Winkel im Bogenmaß

Rückgabe

In das Gradmaß umgerechneter Winkel

4.18.3.3 DT_point UTL_getPointOfDH (const DT_double **const *dh*)

Extrahiert den Punkt aus der DH-Matrix.

Extrahiert den Punkt ohne Orientierung aus der DH-Matrix.

Parameter

dh DH-Matrix

Rückgabe

Extrahierter Punkt

4.18.3.4 DT_double UTL_getRadiant (DT_double *angle*)

Umrechnung in das Bogenmaß.

Rechnet einen Winkel in das Bogenmaß um.

Parameter

angle Winkel in Grad

Rückgabe

In das Bogenmaß umgerechneter Winkel

4.18.3.5 void UTL_printDebug (const DT_char *const *msg*, DT_size *size*)

Debug-Ausgabe.

Gibt einen Text auf der stdo oder der definierten Debug-USART aus.

Parameter

msg Text für die Ausgabe

size Länge des Textes

4.18.3.6 void UTL_printDebugByte (const DT_byte *const *packet*, DT_size *size*)

Debug-Ausgabe von Bytes.

Gibt Bytes in Hexadezimal auf der stdo oder der definierten Debug-USART aus.

Parameter

packet Paket für die Ausgabe

size Größe des Pakets

4.18.3.7 void UTL_printLeg (const DT_leg *const *leg*, DT_type *type*)

Gibt die Soll-Winkel eines Beines auf stdo aus.

Parameter

leg Bein für das die Daten ausgegeben werden sollen

type Typ der Ausgabe in Bogenmaß oder Grad

4.18.3.8 void UTL_printMatrix (const DT_double **const *mat*, DT_size *rows*, DT_size *columns*)

Gibt eine Matrix auf stdo aus.

Parameter

mat Point auf Matrix

rows Zeilenanzahl

columns Spaltenanzahl

4.18.3.9 void UTL_printPoint (const DT_point *const *p*)

Gibt einen Punkt auf stdo aus.

Parameter

p Punkt zur Ausgabe

4.18.3.10 void UTL_wait (DT_size *rounds*)

Abstraktion von einer Pause/Delay.

Implementierung durch Schleifen ($\text{rounds} * 64k$).

Parameter

rounds Länge der Pause

4.19 include/xmega.h-Dateireferenz

Spezifische Funktionen für den Mikrocontroller ATXmega128A1.

```
#include "usart_driver.h"
```

```
#include "datatypes.h"
```

Makrodefinitionen

- #define **XM_PORT_SERVO_L** PORTC
- #define **XM_PORT_SERVO_R** PORTD
- #define **XM_PORT_COM1** PORTD
- #define **XM_PORT_COM3** PORTE
- #define **XM_PORT_REMOTE** PORTE
- #define **XM_PORT_DEBUG** PORTF
- #define **XM_USART_SERVO_L** USARTC0
- #define **XM_USART_SERVO_R** USARTD0
- #define **XM_USART_COM1** USARTD1
- #define **XM_USART_COM3** USARTE0
- #define **XM_USART_REMOTE** USARTE1
- #define **XM_USART_DEBUG** USARTF0
- #define **XM_PORT_LED** PORTQ
- #define **XM_LED_MASK** (1<<PIN3)
- #define **XM_LED_ON** XM_PORT_LED.OUTCLR = XM_LED_MASK;
- #define **XM_LED_OFF** XM_PORT_LED.OUTSET = XM_LED_MASK;
- #define **XM_LED_TGL** XM_PORT_LED.OUTTGL = XM_LED_MASK;
- #define **SWITCHPORT** PORTQ
- #define **SWITCHMASK** (1<<PIN2)
- #define **SWITCH_PRESSED** (SWITCHPORT.IN&SWITCHMASK)!=SWITCHMASK
- #define **SWITCH_RELEASED** (SWITCHPORT.IN&SWITCHMASK)==SWITCHMASK
- #define **XM_OE_MASK** (1<<PIN0)
- #define **XM_USART_FAILURE** 0xFF

Funktionen

- void **XM_init_cpu** ()
Initialisierung der CPU.
- void **XM_init_remote** ()
Initialisiert den Zigbee-Fernsteuerung.
- void **XM_init_dnx** ()
Initialisiert die Servo-USARTs.
- void **XM_init_com** (DT_byte)

Initialisiert USARTs für die CPU-Kommunikation.

- void **XM_USART_send** (USART_data_t *const, const DT_byte *const, DT_size)

USART-Sendemethode.

Variablen

- USART_data_t XM_servo_data_L
- USART_data_t XM_servo_data_R
- USART_data_t XM_debug_data
- USART_data_t XM_remote_data
- USART_data_t XM_com_data1
- USART_data_t XM_com_data3

4.19.1 Ausführliche Beschreibung

Spezifische Funktionen für den Mikrocontroller ATXmega128A1.

4.19.2 Makro-Dokumentation

4.19.2.1 **#define SWITCH_PRESSED (SWITCH-PORT.IN&SWITCHMASK)!=SWITCHMASK**

4.19.2.2 **#define SWITCH_RELEASED (SWITCH-PORT.IN&SWITCHMASK)==SWITCHMASK**

4.19.2.3 **#define SWITCHMASK (1<<PIN2)**

4.19.2.4 **#define SWITCHPORT PORTQ**

4.19.2.5 **#define XM_LED_MASK (1<<PIN3)**

4.19.2.6 **#define XM_LED_OFF XM_PORT_LED.OUTSET = XM_LED_MASK;**

4.19.2.7 **#define XM_LED_ON XM_PORT_LED.OUTCLR = XM_LED_MASK;**

4.19.2.8 **#define XM_LED_TGL XM_PORT_LED.OUTTGL = XM_LED_MASK;**

4.19.2.9 **#define XM_OE_MASK (1<<PIN0)**

4.19.2.10 **#define XM_PORT_COM1 PORTD**

4.19.2.11 **#define XM_PORT_COM3 PORTE**

4.19.2.12 **#define XM_PORT_DEBUG PORTF**

4.19.2.13 **#define XM_PORT_LED PORTQ**

4.19.2.14 **#define XM_PORT_REMOTE PORTE**

4.19.2.15 **#define XM_PORT_SERVO_L PORTC**

4.19.2.16 **#define XM_PORT_SERVO_R PORTD**

4.19.2.17 **#define XM_USART_COM1 USARTD1**

4.19.2.18 **#define XM_USART_COM3 USARTE0**

4.19.2.19 **#define XM_USART_DEBUG USARTF0**

4.19.2.20 **#define XM_USART_FAILURE 0xFF**

signalisiert Fehler beim Empfangen

4.19.2.21 `#define XM_USART_REMOTE USARTE1`

4.19.2.22 `#define XM_USART_SERVO_L USARTC0`

4.19.2.23 `#define XM_USART_SERVO_R USARTD0`

4.19.3 Dokumentation der Funktionen

4.19.3.1 `void XM_init_com (DT_byte cpuID)`

Initialisiert USARTs für die CPU-Kommunikation.

ID des Controllers wird aufgrund des Hardwaredefekts für die Master-Slave-Kommunikation benötigt.

Parameter

cpuID ID des Controllers

4.19.3.2 `void XM_init_cpu ()`

Initialisierung der CPU.

4.19.3.3 `void XM_init_dnx ()`

Initialisiert die Servo-USARTs.

4.19.3.4 `void XM_init_remote ()`

Initialisiert den Zigbee-Fernsteuerung.

4.19.3.5 `void XM_USART_send (USART_data_t *const usart_data, const DT_byte *const txData, DT_size bytes)`

USART-Sendemethode.

Diese Methode setzt zunächst das jeweilige Output-Enable (!OE) auf Senden und schreibt das zu sendende Paket in den USART-Buffer. Anschließend wird der TX-Interrupt aktiviert, der ausgelöst wird, wenn das letzte Paket gesendet wurde.

Parameter

usart_data USART-Datenstruktur der zu benutzenden USART

txData Byte-Array mit zu sendendem Paket

bytes Länge des zu sendenden Pakets

4.19.4 Variablen-Dokumentation

4.19.4.1 USART_data_t XM_com_data1

USART-Struktur für Communication (Master -> Slave 1).

4.19.4.2 USART_data_t XM_com_data3

USART-Struktur für Communication (Master -> Slave 3).

4.19.4.3 USART_data_t XM_debug_data

USART-Struktur für Debug-Ausgaben.

4.19.4.4 USART_data_t XM_remote_data

USART-Struktur für Remote-Controller.

4.19.4.5 USART_data_t XM_servo_data_L

USART-Struktur für linke Dynamixel.

4.19.4.6 USART_data_t XM_servo_data_R

USART-Struktur für rechte Dynamixel.

4.20 kinematics.c-Dateireferenz

Lösungsmethoden der Kinematik.

```
#include <math.h>
#include <stdlib.h>
#include "include/kinematics.h"
#include "include/utils.h"
#include "include/dynamixel.h"
```

Makrodefinitionen

- #define **DIST_HK** 50
Abstand von Hüfte zu Knie.
- #define **DIST_KF** 85

Abstand von Knie zu Fuß.

- **#define DIST_FE 55**
Abstand von Fuß zu Fußende.
- **#define DIST_DZ -14**
Versatz in z-Richtung von Knie in Bezug auf Hüfte.

Funktionen

- void **KIN_setTransMat** (**DT_leg** *const leg)
Liefert fuer ein Bein die Struktur zur Koordinatentransformation.
- void **KIN_calcDH** (const **DT_leg** *const leg, **DT_double** **dh03)
Lösung des kinematischen Problems.
- **DT_bool** **KIN_calcServos** (const **DT_point** *const p, **DT_leg** *const leg)
Lösung des inversen kinematischen Problems.
- **DT_point** **KIN_calcLocalPoint** (const **DT_point** *const p, const **DT_transformation** *const trans)
Transformiert einen Punkt in das Roboterkoordinatensystem.

4.20.1 Ausführliche Beschreibung

Lösungsmethoden der Kinematik. Lösungsmethoden der Kinematik speziell für den CM-Bot.

4.20.2 Makro-Dokumentation

4.20.2.1 **#define DIST_DZ -14**

Versatz in z-Richtung von Knie in Bezug auf Hüfte.

4.20.2.2 **#define DIST_FE 55**

Abstand von Fuß zu Fußende.

4.20.2.3 **#define DIST_HK 50**

Abstand von Hüfte zu Knie.

4.20.2.4 #define DIST_KF 85

Abstand von Knie zu Fuß.

4.20.3 Dokumentation der Funktionen

4.20.3.1 void KIN_calcDH (const DT_leg *const leg, DT_double ** dh03)

Lösung des kinematischen Problems.

Lösung der Denavit-Hartenberg-Transformation.

Parameter

leg Bein mit den Soll-Winkel der Gelenke

dh03 Zielmatrix für die Lösung

4.20.3.2 DT_point KIN_calcLocalPoint (const DT_point *const p, const DT_transformation *const trans)

Transformiert einen Punkt in das Roboterkoordinatensystem.

Transformiert einen Punkt des Weltkoordinatensystems in das Roboterkoordinatensystem eines Beines. Berechnet nur eine x-y-Verschiebung und eine Rotation von 0/180 Grad um die z-Achse.

Parameter

p Punkt im Weltkoordinatensystem

trans Struktur mit Informationen fuer die Koordinatentransformation

Rückgabe

Punkt im Roboterkoordinatensystem eines Beines

4.20.3.3 DT_bool KIN_calcServos (const DT_point *const p, DT_leg *const leg)

Lösung des inversen kinematischen Problems.

Lösung des inversen kinematischen Problems mit Hilfe eines geometrischen Verfahrens mit leichten Einschränkungen.

Parameter

p Punkt (Roboterkoordinate)

leg Bein für die zusetzenden Winkel

Rückgabe

true, wenn Berechnung erfolgreich

4.20.3.4 void KIN_setTransMat (DT_leg *const leg)

Liefert fuer ein Bein die Struktur zur Koordinatentransformation.

Liefert fuer ein Bein die Struktur zur Koordinatentransformation.

Parameter

leg Bein

4.21 main.c-Dateireferenz

```
#include "avr/io.h"
#include "include/avr_compiler.h"
#include "include/usart_driver.h"
#include "math.h"
```

Makrodefinitionen

- #define F_CPU 32000000UL
- #define TEST_OFF

4.21.1 Makro-Dokumentation

4.21.1.1 #define F_CPU 32000000UL

4.21.1.2 #define TEST_OFF

4.22 movement.c-Dateireferenz

Stellt Grundfunktionen für einen Laufalgorithmus bereit.

```
#include "include/movement.h"
#include "include/xmega.h"
#include "include/utils.h"
#include "include/communication.h"
#include "include/dynamixel.h"
#include "include/kinematics.h"
```

Makrodefinitionen

- #define MV_DST_X 168.5

Funktionen

- void **MV_action** (DT_leg *const leg_r, DT_leg *const leg_l)
Sendet das ACTION-Kommando an ein Bein.
- void **MV_slave** (DT_byte cpuID, DT_leg *const leg_r, DT_leg *const leg_l)
Standard-Methode für einen Slave-Controller.
- void **MV_masterCheckAlive** ()
Prüft ob die Slaves alive sind. (Master).
- void **MV_slaveStatus** (const DT_byte *const result, const DT_size len)
Antwortet auf eine Status-Anfrage eines Masters. (Slave).
- void **MV_slavePoint** (DT_leg *const leg_r, DT_leg *const leg_l, const DT_byte *const result, DT_size len)
Führt die benötigten Aktionen für einen empfangenen Punkt aus. (Slave).
- void **MV_slavePointAndSpeed** (DT_leg *const leg_r, DT_leg *const leg_l, const DT_byte *const result, DT_size len)
Führt die benötigten Aktionen für einen empfangenen Punkt und Anfahrgeschwindigkeit aus. (Slave).
- void **MV_slaveAngle** (DT_leg *const leg_r, DT_leg *const leg_l, const DT_byte *const result, DT_size len)
Führt die benötigten Aktionen für einen empfangenen Winkel aus. (Slave).
- DT_bool **MV_point** (DT_leg *const leg, const DT_point *const point, DT_bool isGlobal)
Berechnet Winkel anhand des Punktes für die Servos und versendet diese.
- DT_bool **MV_pointAndSpeed** (DT_leg *const leg, const DT_point *const point, const DT_double speed, DT_bool isGlobal)
Berechnet Winkel anhand des Punktes für die Servos und versendet diese zusammen mit der Anfahrgeschwindigkeit.
- void **MV_doInitPosition** (DT_leg *const leg_r, DT_leg *const leg_l)
Führt das rechte und linke Bein in eine Startposition.
- void **MV_switchLegs** (DT_byte *side, DT_byte *master_dwn, DT_byte *master_up, DT_byte *slave_dwn, DT_byte *slave_up)
Wechselt die aktiven und passiven Beine.
- DT_point **MV_getPntForCpuSide** (const DT_point *const p, const DT_byte cpuId, const DT_byte side)
Rechnet einen lokalen Punkt mit Bezug auf das mittlere Bein in einen globalen Punkt für einen Controller um.

4.22.1 Ausführliche Beschreibung

Stellt Grundfunktionen für einen Laufalgorithmus bereit.

4.22.2 Makro-Dokumentation

4.22.2.1 #define MV_DST_X 168.5

4.22.3 Dokumentation der Funktionen

4.22.3.1 void MV_action (DT_leg *const *leg_r*, DT_leg *const *leg_l*)

Sendet das ACTION-Kommando an ein Bein.

Sendet das ACTION-Kommando an alle Servos eines Beines.

Parameter

leg_r rechtes Bein

leg_l linkes Bein

4.22.3.2 void MV_doInitPosition (DT_leg *const *leg_r*, DT_leg *const *leg_l*)

Führt das rechte und linke Bein in eine Startposition.

Führt das rechte und linke Bein in eine Startposition.

Parameter

leg_r rechtes Bein

leg_l linkes Bein

4.22.3.3 DT_point MV_getPntForCpuSide (const DT_point *const *p*, const DT_byte *cpuId*, const DT_byte *side*)

Rechnet einen lokalen Punkt mit Bezug auf das mittlere Bein in einen globalen Punkt für einen Controller um.

Rechnet einen lokalen Punkt mit Bezug auf das mittlere Bein in einen globalen Punkt für einen Controller um.

Parameter

p lokaler Punkt

cpuId ID des Controllers

side Seite (Links/Rechts)

Rückgabe

Globaler Punkt

4.22.3.4 void MV_masterCheckAlive ()

Prüft ob die Slaves alive sind. (Master).

Prüft blockierend ob die Slaves alive sind.

4.22.3.5 DT_bool MV_point (DT_leg *const *leg*, const DT_point *const *point*, DT_bool *isGlobal*)

Berechnet Winkel anhand des Punktes für die Servos und versendet diese.

Berechnet Winkel anhand des Punktes für die Servos und versendet diese.

Parameter

leg Bein

point Punkt

isGlobal Weltkoordinate, wenn true

4.22.3.6 DT_bool MV_pointAndSpeed (DT_leg *const *leg*, const DT_point *const *point*, const DT_double *speed*, DT_bool *isGlobal*)

Berechnet Winkel anhand des Punktes für die Servos und versendet diese zusammen mit der Anfahrgeschwindigkeit.

Berechnet Winkel anhand des Punktes für die Servos und versendet diese zusammen mit der Anfahrgeschwindigkeit.

Parameter

leg Bein

point Punkt

speed Anfahrgeschwindigkeit

isGlobal Weltkoordinate, wenn true

4.22.3.7 void MV_slave (DT_byte *cpuID*, DT_leg *const *leg_r*, DT_leg *const *leg_l*)

Standard-Methode für einen Slave-Controller.

Standard-Methode für einen Slave-Controller. Nimmt Befehle eines Masters entgegen und führt die entsprechenden Aktionen aus.

Parameter

cpuID ID des Controllers auf dem die Methode ausgeführt wird

leg_r rechtes Bein

leg_l linkes Bein

4.22.3.8 void MV_slaveAngle (DT_leg *const *leg_r*, DT_leg *const *leg_l*,
const DT_byte *const *result*, DT_size *len*)

Führt die benötigten Aktionen für einen empfangenen Winkel aus. (Slave).

Führt die benötigten Aktionen für einen empfangenen Winkel aus. (Slave)

Parameter

leg_r rechtes Bein

leg_l linkes Bein

result Anfrage-Paket zum Auswerten

len Länge des Pakets

4.22.3.9 void MV_slavePoint (DT_leg *const *leg_r*, DT_leg *const *leg_l*,
const DT_byte *const *result*, DT_size *len*)

Führt die benötigten Aktionen für einen empfangenen Punkt aus. (Slave).

Führt die benötigten Aktionen für einen empfangenen Punkt aus. (Slave)

Parameter

leg_r rechtes Bein

leg_l linkes Bein

result Anfrage-Paket zum Auswerten

len Länge des Pakets

4.22.3.10 void MV_slavePointAndSpeed (DT_leg *const *leg_r*, DT_leg *const
leg_l, const DT_byte *const *result*, DT_size *len*)

Führt die benötigten Aktionen für einen empfangenen Punkt und An-
fahrsgeschwindigkeit aus. (Slave).

Führt die benötigten Aktionen für einen empfangenen Punkt und An-
fahrsgeschwindigkeit aus. (Slave)

Parameter

leg_r rechtes Bein

leg_l linkes Bein

result Anfrage-Paket zum Auswerten

len Länge des Pakets

4.22.3.11 void MV_slaveStatus (const DT_byte *const *result*, const DT_size *len*)

Antwortet auf eine Status-Anfrage eines Masters. (Slave).

Antwortet auf eine Status-Anfrage eines Masters. (Slave)

Parameter

result Anfrage-Paket zum Auswerten

len Länge des Pakets

4.22.3.12 void MV_switchLegs (DT_byte * *side*, DT_byte * *master_dwn*, DT_byte * *master_up*, DT_byte * *slave_dwn*, DT_byte * *slave_up*)

Wechselt die aktiven und passiven Beine.

Wechselt die aktiven und passiven Beine (Oben/Unten). Übergebene Variablen können in den jeweiligen Laufalgorithmus ausgewertet werden.

Parameter

side Aktive Seite (Seite mit Bodenkontakt), mittleres Beinpaar als Referenz

master_dwn Seite (Links/Rechts) für den Master mit Bodenkontakt

master_up Seite (Links/Rechts) für den Master ohne Bodenkontakt

slave_dwn Seite (Links/Rechts) für den Slave mit Bodenkontakt

slave_up Seite (Links/Rechts) für den Master ohne Bodenkontakt

4.23 movement4Points.c-Dateireferenz

Algorithmus fuer das Vorwaertslaufen ueber 4 Punkte.

Makrodefinitionen

- #define TEST_OFF TEST

4.23.1 Ausführliche Beschreibung

Algorithmus fuer das Vorwaertslaufen ueber 4 Punkte.

4.23.2 Makro-Dokumentation

4.23.2.1 #define TEST_OFF TEST

4.24 movementMultiPoints.c-Dateireferenz

Algorithmus fuer das Vorwaertslaufen ueber 4 Punkte.

Makrodefinitionen

- #define TEST_OFF TEST

4.24.1 Ausführliche Beschreibung

Algorithmus fuer das Vorwaertslaufen ueber 4 Punkte.

4.24.2 Makro-Dokumentation

4.24.2.1 #define TEST_OFF TEST

4.25 remote.c-Dateireferenz

Methoden zur Steuerung durch den RC-100 Remote Controller.

```
#include "include/remote.h"
```

```
#include "include/xmega.h"
```

```
#include "include/utils.h"
```

Makrodefinitionen

- #define B_NON_PRESSED 0x0000
- #define B_U 0x0001
- #define B_D 0x0002
- #define B_L 0x0004
- #define B_R 0x0008
- #define B_1 0x0010
- #define B_2 0x0020
- #define B_3 0x0040
- #define B_4 0x0080
- #define B_5 0x0100
- #define B_6 0x0200

Funktionen

- **DT_cmd RMT_getCommand ()**
Liest empfangene Befehle vom Remote-Controller aus.
- **DT_byte RMT_receive (USART_data_t *const usart_data, DT_byte *const dest)**
USART-Empfangsmethode für Remote-Controller.
- **DT_bool RMT_NonPressed (DT_cmd cmd)**
Kein Taster gedrückt.
- **DT_bool RMT_isUpPressed (DT_cmd cmd)**
Ist Taster Up gedrückt.
- **DT_bool RMT_isDownPressed (DT_cmd cmd)**
Ist Taster Down gedrückt.
- **DT_bool RMT_isLeftPressed (DT_cmd cmd)**
Ist Taster Left gedrückt.
- **DT_bool RMT_isRightPressed (DT_cmd cmd)**
Ist Taster Right gedrückt.
- **DT_bool RMT_isButton1Pressed (DT_cmd cmd)**
Ist Taster 1 gedrückt.
- **DT_bool RMT_isButton2Pressed (DT_cmd cmd)**
Ist Taster 2 gedrückt.
- **DT_bool RMT_isButton3Pressed (DT_cmd cmd)**
Ist Taster 3 gedrückt.
- **DT_bool RMT_isButton4Pressed (DT_cmd cmd)**
Ist Taster 4 gedrückt.
- **DT_bool RMT_isButton5Pressed (DT_cmd cmd)**
Ist Taster 5 gedrückt.
- **DT_bool RMT_isButton6Pressed (DT_cmd cmd)**
Ist Taster 6 gedrückt.

4.25.1 Ausführliche Beschreibung

Methoden zur Steuerung durch den RC-100 Remote Controller.

4.25.2 Makro-Dokumentation

4.25.2.1 `#define B_1 0x0010`

4.25.2.2 `#define B_2 0x0020`

4.25.2.3 `#define B_3 0x0040`

4.25.2.4 `#define B_4 0x0080`

4.25.2.5 `#define B_5 0x0100`

4.25.2.6 `#define B_6 0x0200`

4.25.2.7 `#define B_D 0x0002`

4.25.2.8 `#define B_L 0x0004`

4.25.2.9 `#define B_NON_PRESSED 0x0000`

4.25.2.10 `#define B_R 0x0008`

4.25.2.11 `#define B_U 0x0001`

4.25.3 Dokumentation der Funktionen

4.25.3.1 `DT_cmd RMT_getCommand ()`

Liest empfangene Befehle vom Remote-Controller aus.

Instruction aus dem High- und Low-Teil zusammensetzen Bsp: Paket für B_2: 0 1 2 3 4
5 ... FF;55;20;DF;00;FF;FF;55;00;FF;00;FF; LL HH Eigentliche Information: 0x0020
d.h. H = Paket[4] L = Paket[2]

4.25.3.2 `DT_bool RMT_isButton1Pressed (DT_cmd cmd)`

Ist Taster 1 gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.25.3.3 DT_bool RMT_isButton2Pressed (DT_cmd *cmd*)

Ist Taster 2 gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.25.3.4 DT_bool RMT_isButton3Pressed (DT_cmd *cmd*)

Ist Taster 3 gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.25.3.5 DT_bool RMT_isButton4Pressed (DT_cmd *cmd*)

Ist Taster 4 gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.25.3.6 DT_bool RMT_isButton5Pressed (DT_cmd *cmd*)

Ist Taster 5 gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.25.3.7 DT_bool RMT_isButton6Pressed (DT_cmd *cmd*)

Ist Taster 6 gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.25.3.8 DT_bool RMT_isDownPressed (DT_cmd *cmd*)

Ist Taster Down gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.25.3.9 DT_bool RMT_isLeftPressed (DT_cmd *cmd*)

Ist Taster Left gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.25.3.10 DT_bool RMT_isRightPressed (DT_cmd *cmd*)

Ist Taster Right gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.25.3.11 DT_bool RMT_isUpPressed (DT_cmd *cmd*)

Ist Taster Up gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

4.25.3.12 DT_bool RMT_NonPressed (DT_cmd *cmd*)

Kein Taster gedrückt.

Achtung: *cmd* muss zunächst durch `getCommand` abgerufen werden.

Parameter

cmd 16-Bit Command-Wert

Rückgabe

Bool

**4.25.3.13 DT_byte RMT_receive (USART_data_t *const *usart_data*,
DT_byte *const *dest*)**

USART-Empfangsmethode für Remote-Controller.

Diese Methode liest den Remote-USART-Buffer aus und prüft, ob ein vollständiges Paket empfangen wurde.

Parameter

usart_data USART

dest Byte-Array für Antwort-Paket

Rückgabe

Länge des Antwortpakets

4.26 testArithmetic.c-Dateireferenz

Makrodefinitionen

- #define TEST_OFF TEST

4.26.1 Makro-Dokumentation

4.26.1.1 #define TEST_OFF TEST

4.27 testCom.c-Dateireferenz

Testprogramm für Kommunikation der CPUs.

Makrodefinitionen

- #define TEST_OFF

4.27.1 Ausführliche Beschreibung

Testprogramm für Kommunikation der CPUs. Testprogramm für Evolutionären Algorithmus zur Startpunktfindung.

4.27.2 Makro-Dokumentation

4.27.2.1 #define TEST_OFF

4.28 testCom2.c-Dateireferenz

Makrodefinitionen

- #define TEST_OFF

4.28.1 Makro-Dokumentation

4.28.1.1 #define TEST_OFF

4.29 testCom3.c-Dateireferenz

Makrodefinitionen

- #define TEST_OFF TEST

4.29.1 Makro-Dokumentation

4.29.1.1 #define TEST_OFF TEST

4.30 testDnx.c-Dateireferenz

Testprogramm für Ansteuerung der Servos.

```
#include "include/xmega.h"
#include "include/utils.h"
#include "include/dynamixel.h"
```

Makrodefinitionen

- #define TEST_OFF

4.30.1 Ausführliche Beschreibung

Testprogramm für Ansteuerung der Servos.

4.30.2 Makro-Dokumentation

4.30.2.1 #define TEST_OFF

4.31 testEvolutionaryDistanceWalking.c-Dateireferenz

Makrodefinitionen

- #define TEST_OFF

4.31.1 Makro-Dokumentation

4.31.1.1 #define TEST_OFF

4.32 testKin.c-Dateireferenz

Testprogramm für die Kinematik.

Makrodefinitionen

- #define TEST_OFF

4.32.1 Ausführliche Beschreibung

Testprogramm für die Kinematik.

4.32.2 Makro-Dokumentation

4.32.2.1 #define TEST_OFF

4.33 testKin2.c-Dateireferenz

Makrodefinitionen

- #define TEST_OFF TEST

4.33.1 Makro-Dokumentation

4.33.1.1 #define TEST_OFF TEST

4.34 testMovement.c-Dateireferenz

Makrodefinitionen

- #define TEST_OFF TEST

4.34.1 Makro-Dokumentation

4.34.1.1 #define TEST_OFF TEST

4.35 testRemote.c-Dateireferenz

Makrodefinitionen

- #define TEST_OFF

4.35.1 Makro-Dokumentation

4.35.1.1 #define TEST_OFF

4.36 testRingBuffer.c-Dateireferenz

Testprogramm für einen Ringbuffer.

Makrodefinitionen

- #define TEST_OFF

4.36.1 Ausführliche Beschreibung

Testprogramm für einen Ringbuffer.

4.36.2 Makro-Dokumentation

4.36.2.1 #define TEST_OFF

4.37 testSpeed.c-Dateireferenz

Testprogramm für Speed-Änderung der Servos.

Makrodefinitionen

- #define TEST_OFF

4.37.1 Ausführliche Beschreibung

Testprogramm für Speed-Änderung der Servos.

4.37.2 Makro-Dokumentation

4.37.2.1 #define TEST_OFF

4.38 usart_driver.c-Dateireferenz

XMEGA USART driver source file.

```
#include "include/usart_driver.h"
```

Funktionen

- void **USART_InterruptDriver_Initialize** (USART_data_t *usart_data, USART_t *usart, USART_DREINTLVL_t dreIntLevel)
Initializes buffer and selects what USART module to use.
- void **USART_InterruptDriver_DreInterruptLevel_Set** (USART_data_t *usart_data, USART_DREINTLVL_t dreIntLevel)
Set USART DRE interrupt level.
- bool **USART_TXBuffer_FreeSpace** (USART_data_t *usart_data)
Test if there is data in the transmitter software buffer.
- bool **USART_TXBuffer_PutByte** (USART_data_t *usart_data, uint8_t data)
Put data (5-8 bit character).
- bool **USART_RXBufferData_Available** (USART_data_t *usart_data)
Test if there is data in the receive software buffer.
- uint8_t **USART_RXBuffer_GetByte** (USART_data_t *usart_data)
Get received data (5-8 bit character).
- **DT_bool** **USART_RXBuffer_checkPointerDiff** (DT_byte tail, DT_byte head, DT_byte diff)
Überprüft Differenzen der Empfangs-Pointer.
- bool **USART_RXComplete** (USART_data_t *usart_data)
RX Complete Interrupt Service Routine.
- void **USART_DataRegEmpty** (USART_data_t *usart_data)
Data Register Empty Interrupt Service Routine.
- void **USART_NineBits_PutChar** (USART_t *usart, uint16_t data)
Put data (9 bit character).

- `uint16_t USART_NineBits_GetChar (USART_t *usart)`

Get received data (9 bit character).

4.38.1 Ausführliche Beschreibung

XMEGA USART driver source file. This file contains the function implementations the XMEGA interrupt and polled USART driver.

The driver is not intended for size and/or speed critical code, since most functions are just a few lines of code, and the function call overhead would decrease code performance. The driver is intended for rapid prototyping and documentation purposes for getting started with the XMEGA ADC module.

For size and/or speed critical code, it is recommended to copy the function contents directly into your application instead of making a function call.

Some functions use the following construct: "some_register = ... | (some_parameter ? SOME_BIT_bm : 0) | ..." Although the use of the ternary operator (if ? then : else) is discouraged, in some occasions the operator makes it possible to write pretty clean and neat code. In this driver, the construct is used to set or not set a configuration bit based on a boolean input parameter, such as the "some_parameter" in the example above.

Application note:

AVR1307: Using the XMEGA USART

Documentation

For comprehensive code documentation, supported compilers, compiler settings and supported devices see [readme.html](#)

Autor

Atmel Corporation: <http://www.atmel.com>
Support email: avr@atmel.com

Revision:

481

Date:

2007-03-06 10:12:53 +0100 (ty, 06 mar 2007)

Copyright (c) 2008, Atmel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name of ATMEL may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY ATMEL "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.38.2 Dokumentation der Funktionen

4.38.2.1 void USART_DataRegEmpty (USART_data_t * usart_data)

Data Register Empty Interrupt Service Routine.

Data Register Empty Interrupt Service Routine. Transmits one byte from TX software buffer. Disables DRE interrupt if buffer is empty. Argument is pointer to USART (USART_data_t).

Parameter

usart_data The USART_data_t struct instance.

4.38.2.2 void USART_InterruptDriver_DreInterruptLevel_Set (USART_data_t * usart_data, USART_DREINTLVL_t dreIntLevel)

Set USART DRE interrupt level.

Set the interrupt level on Data Register interrupt.

Zu beachten

Changing the DRE interrupt level in the interrupt driver while it is running will not change the DRE interrupt level in the USART before the DRE interrupt have been disabled and enabled again.

Parameter

usart_data The USART_data_t struct instance

dreIntLevel Interrupt level of the DRE interrupt.

4.38.2.3 void USART_InterruptDriver_Initialize (USART_data_t *
usart_data, USART_t * usart, USART_DREINTLVL_t dreIntLevel)

Initializes buffer and selects what USART module to use.

Initializes receive and transmit buffer and selects what USART module to use, and stores the data register empty interrupt level.

Parameter

usart_data The USART_data_t struct instance.

usart The USART module.

dreIntLevel Data register empty interrupt level.

4.38.2.4 uint16_t USART_NineBits_GetChar (USART_t * usart)

Get received data (9 bit character).

This function reads out the received 9 bit character (uint16_t). Use the function USART_IsRXComplete to check if anything is received.

Parameter

usart The USART module.

Rückgabewerte

Received data.

4.38.2.5 void USART_NineBits_PutChar (USART_t * usart, uint16_t data)

Put data (9 bit character).

Use the function USART_IsTXDataRegisterEmpty before using this function to put 9 bit character to the TX register.

Parameter

usart The USART module.

data The data to send.

4.38.2.6 DT_bool USART_RXBuffer_checkPointerDiff (DT_byte tail,
DT_byte head, DT_byte diff)

Überprüft Differenzen der Empfangs-Pointer.

Parameter

tail Tail-Pointer

head Head-Pointer

diff Differenz

Rückgabe

Bool

4.38.2.7 uint8_t USART_RXBuffer_GetByte (USART_data_t * usart_data)

Get received data (5-8 bit character).

The function USART_RXBufferData_Available should be used before this function is used to ensure that data is available.

Returns data from RX software buffer.

Parameter

usart_data The USART_data_t struct instance.

Rückgabe

Received data.

4.38.2.8 bool USART_RXBufferData_Available (USART_data_t * usart_data)

Test if there is data in the receive software buffer.

This function can be used to test if there is data in the receive software buffer.

Parameter

usart_data The USART_data_t struct instance

Rückgabewerte

true There is data in the receive buffer.

false The receive buffer is empty.

4.38.2.9 bool USART_RXComplete (USART_data_t * usart_data)

RX Complete Interrupt Service Routine.

RX Complete Interrupt Service Routine. Stores received data in RX software buffer.

Parameter

usart_data The USART_data_t struct instance.

4.38.2.10 bool USART_TXBuffer_FreeSpace (USART_data_t * *usart_data*)

Test if there is data in the transmitter software buffer.

This function can be used to test if there is free space in the transmitter software buffer.

Parameter

usart_data The USART_data_t struct instance.

Rückgabewerte

true There is data in the receive buffer.

false The receive buffer is empty.

**4.38.2.11 bool USART_TXBuffer_PutByte (USART_data_t * *usart_data*,
uint8_t *data*)**

Put data (5-8 bit character).

Stores data byte in TX software buffer and enables DRE interrupt if there is free space in the TX software buffer.

Parameter

usart_data The USART_data_t struct instance.

data The data to send.

4.39 utils.c-Dateireferenz

Verschiedene Hilfsmethoden.

```
#include <stdio.h>
#include <math.h>
#include "include/utils.h"
#include "include/xmega.h"
```

Makrodefinitionen

- **#define USART_ON**
Debug-Ausgabe auf stdo oder USART.

Funktionen

- void **UTL_printMatrix** (const DT_double **const mat, DT_size rows, DT_size columns)

Gibt eine Matrix auf stdo aus.

- void **UTL_printLeg** (const **DT_leg** *const leg, **DT_type** type)
Gibt die Soll-Winkel eines Beines auf stdo aus.
- void **UTL_printPoint** (const **DT_point** *const p)
Gibt einen Punkt auf stdo aus.
- **DT_double** **UTL_getRadiant** (**DT_double** angle)
Umrechnung in das Bogenmaß.
- **DT_double** **UTL_getDegree** (**DT_double** radiant)
Umrechnung in das Gradmaß.
- **DT_point** **UTL_getPointOfDH** (const **DT_double** **const dh)
Extrahiert den Punkt aus der DH-Matrix.
- void **UTL_printDebug** (const **DT_char** *const msg, **DT_size** size)
Debug-Ausgabe.
- void **UTL_printDebugByte** (const **DT_byte** *const packet, **DT_size** size)
Debug-Ausgabe von Bytes.
- **DT_byte** **UTL_byteToHexChar** (**DT_char** *const dest, const **DT_byte** *const src, **DT_size** size)
Konvertierung eines Bytes in das Hexadezimal-Format.
- void **UTL_wait** (**DT_size** rounds)
Abstraktion von einer Pause/Delay.

4.39.1 Ausführliche Beschreibung

Verschiedene Hilfsmethoden. Stellt verschiedene Hilfsmethoden für allgemeinen Gebrauch zur Verfügung.

4.39.2 Makro-Dokumentation

4.39.2.1 #define USART_ON

Debug-Ausgabe auf stdo oder USART.

4.39.3 Dokumentation der Funktionen

4.39.3.1 DT_byte UTL_byteToHexChar (DT_char *const *dest*, const DT_byte *const *src*, DT_size *size*)

Konvertierung eines Bytes in das Hexadezimal-Format.

Parameter

dest Pointer auf das Zielfeld

src Pointer auf das Quellfeld

size Größe des Quellfelds

4.39.3.2 DT_double UTL_getDegree (DT_double *radiant*)

Umrechnung in das Gradmaß.

Rechnet einen Winkel in das Gradmaß um.

Parameter

radiant Winkel im Bogenmaß

Rückgabe

In das Gradmaß umgerechneter Winkel

4.39.3.3 DT_point UTL_getPointOfDH (const DT_double **const *dh*)

Extrahiert den Punkt aus der DH-Matrix.

Extrahiert den Punkt ohne Orientierung aus der DH-Matrix.

Parameter

dh DH-Matrix

Rückgabe

Extrahierter Punkt

4.39.3.4 DT_double UTL_getRadiant (DT_double *angle*)

Umrechnung in das Bogenmaß.

Rechnet einen Winkel in das Bogenmaß um.

Parameter

angle Winkel in Grad

Rückgabe

In das Bogenmaß umgerechneter Winkel

4.39.3.5 void UTL_printDebug (const DT_char *const *msg*, DT_size *size*)

Debug-Ausgabe.

Gibt einen Text auf der stdo oder der definierten Debug-USART aus.

Parameter

msg Text für die Ausgabe

size Länge des Textes

4.39.3.6 void UTL_printDebugByte (const DT_byte *const *packet*, DT_size *size*)

Debug-Ausgabe von Bytes.

Gibt Bytes in Hexadezimal auf der stdo oder der definierten Debug-USART aus.

Parameter

packet Paket für die Ausgabe

size Größe des Pakets

4.39.3.7 void UTL_printLeg (const DT_leg *const *leg*, DT_type *type*)

Gibt die Soll-Winkel eines Beines auf stdo aus.

Parameter

leg Bein für das die Daten ausgegeben werden sollen

type Typ der Ausgabe in Bogenmaß oder Grad

4.39.3.8 void UTL_printMatrix (const DT_double **const *mat*, DT_size *rows*, DT_size *columns*)

Gibt eine Matrix auf stdo aus.

Parameter

mat Point auf Matrix

rows Zeilenanzahl

columns Spaltenanzahl

4.39.3.9 void UTL_printPoint (const DT_point *const p)

Gibt einen Punkt auf stdo aus.

Parameter

p Punkt zur Ausgabe

4.39.3.10 void UTL_wait (DT_size rounds)

Abstraktion von einer Pause/Delay.

Implementierung durch Schleifen (rounds * 64k).

Parameter

rounds Länge der Pause

4.40 xmega.c-Dateireferenz

Spezifische Funktionen für den Mikrocontroller ATXmega128A1.

```
#include "include/xmega.h"
#include "include/dynamixel.h"
#include "include/utils.h"
#include "include/clksys_driver.h"
#include "include/avr_compiler.h"
#include "include/communication.h"
#include <avr/io.h>
#include <stdlib.h>
```

Funktionen

- void **XM_init_cpu** ()
Initialisierung der CPU.
- void **XM_init_remote** ()
Initialisiert den Zigbee-Fernsteuerung.
- void **XM_init_dnx** ()
Initialisiert die Servo-USARTs.
- void **XM_init_com** (DT_byte cpuID)
Initialisiert USARTs für die CPU-Kommunikation.

- void **XM_USART_send** (**USART_data_t** *const usart_data, const **DT_byte** *const txData, **DT_size** bytes)
USART-Sendemethode.
- **ISR** (USARTC0_TXC_vect)
ISR für abgeschlossenen Sendevorgang der USARTC0 (SERVO L).
- **ISR** (USARTC0_DRE_vect)
ISR für Sendebereitschaft der USARTC0 (SERVO L).
- **ISR** (USARTC0_RXC_vect)
ISR für Empfangsvorgang der USARTC0 (SERVO L).
- **ISR** (USARTD0_TXC_vect)
ISR für abgeschlossenen Sendevorgang der USARTD0 (SERVO R).
- **ISR** (USARTD0_DRE_vect)
ISR für Sendebereitschaft der USARTD0 (SERVO R).
- **ISR** (USARTD0_RXC_vect)
ISR für Empfangsvorgang der USARTD0 (SERVO R).
- **ISR** (USARTD1_TXC_vect)
ISR für abgeschlossenen Sendevorgang der USARTD0 (SERVO R).
- **ISR** (USARTD1_DRE_vect)
ISR für Sendebereitschaft der USARTD0 (SERVO R).
- **ISR** (USARTD1_RXC_vect)
ISR für Empfangsvorgang der USARTD0 (SERVO R).
- **ISR** (USARTE0_TXC_vect)
ISR für abgeschlossenen Sendevorgang der USARTD0 (SERVO R).
- **ISR** (USARTE0_DRE_vect)
ISR für Sendebereitschaft der USARTD0 (SERVO R).
- **ISR** (USARTE0_RXC_vect)
ISR für Empfangsvorgang der USARTD0 (SERVO R).
- **ISR** (USARTE1_RXC_vect)
ISR für Empfangsvorgang der USARTE1 (REMOTE).

4.40.1 Ausführliche Beschreibung

Spezifische Funktionen für den Mikrocontroller ATXmega128A1.

4.40.2 Dokumentation der Funktionen

4.40.2.1 ISR (USARTC0_TXC_vect)

ISR für abgeschlossenen Sendevorgang der USARTC0 (SERVO L).

4.40.2.2 ISR (USARTC0_RXC_vect)

ISR für Empfangsvorgang der USARTC0 (SERVO L).

4.40.2.3 ISR (USARTE0_TXC_vect)

ISR für abgeschlossenen Sendevorgang der USARTE0 (SERVO R).

4.40.2.4 ISR (USARTE0_DRE_vect)

ISR für Sendebereitschaft der USARTE0 (SERVO R).

4.40.2.5 ISR (USARTE0_TXC_vect)

ISR für abgeschlossenen Sendevorgang der USARTE0 (SERVO R).

4.40.2.6 ISR (USARTE0_RXC_vect)

ISR für Empfangsvorgang der USARTE0 (SERVO R).

4.40.2.7 ISR (USARTE1_DRE_vect)

ISR für Sendebereitschaft der USARTE1 (SERVO R).

4.40.2.8 ISR (USARTE1_RXC_vect)

ISR für Empfangsvorgang der USARTE1 (REMOTE).

4.40.2.9 ISR (USARTE1_DRE_vect)

ISR für Sendebereitschaft der USARTE1 (SERVO R).

4.40.2.10 ISR (USARTD0_RXC_vect)

ISR für Empfangsvorgang der USARTD0 (SERVO R).

4.40.2.11 ISR (USARTC0_DRE_vect)

ISR für Sendebereitschaft der USARTC0 (SERVO L).

4.40.2.12 ISR (USARTD1_TXC_vect)

ISR für abgeschlossenen Sendevorgang der USARTD0 (SERVO R).

4.40.2.13 ISR (USARTD1_RXC_vect)

ISR für Empfangsvorgang der USARTD0 (SERVO R).

4.40.2.14 void XM_init_com (DT_byte *cpuID*)

Initialisiert USARTs für die CPU-Kommunikation.

ID des Controllers wird aufgrund des Hardwaredefekts für die Master-Slave-Kommunikation benötigt.

Parameter

cpuID ID des Controllers

4.40.2.15 void XM_init_cpu ()

Initialisierung der CPU.

4.40.2.16 void XM_init_dnx ()

Initialisiert die Servo-USARTs.

4.40.2.17 void XM_init_remote ()

Initialisiert den Zigbee-Fernsteuerung.

4.40.2.18 void XM_USART_send (USART_data_t *const *usart_data*, const DT_byte *const *txData*, DT_size *bytes*)

USART-Sendemethode.

Diese Methode setzt zunächst das jeweilige Output-Enable (!OE) auf Senden und schreibt das zu sendende Paket in den USART-Buffer. Anschließend wird der TX-Interrupt aktiviert, der ausgelöst wird, wenn das letzte Paket gesendet wurde.

Parameter

usart_data USART-Datenstruktur der zu benutzenden USART

txData Byte-Array mit zu sendendem Paket

bytes Länge des zu sendenden Pakets

Index

- A
 - evolutionaryHelper.c, 38
- AB
 - evolutionaryHelper.c, 38
- ACT
 - dynamixel.c, 28
- act_value
 - DT_servo, 8
- ALR_SHUTDOWN
 - dynamixel.c, 28
- avr_compiler.h
 - AVR_ENTER_CRITICAL_REGION, 43
 - AVR_LEAVE_CRITICAL_REGION, 43
 - F_CPU, 43
- AVR_ENTER_CRITICAL_REGION
 - avr_compiler.h, 43
- AVR_LEAVE_CRITICAL_REGION
 - avr_compiler.h, 43
- B
 - evolutionaryHelper.c, 38
- B_1
 - remote.c, 111
- B_2
 - remote.c, 111
- B_3
 - remote.c, 111
- B_4
 - remote.c, 111
- B_5
 - remote.c, 111
- B_6
 - remote.c, 111
- B_D
 - remote.c, 111
- B_L
 - remote.c, 111
- B_NON_PRESSED
 - remote.c, 111
- B_R
 - remote.c, 111
- B_U
 - remote.c, 111
- BD
 - dynamixel.c, 28
- bestSelection
 - evolutionaryAlgorithm.c, 34
- bubblesort
 - evolutionaryHelper.c, 37
 - evolutionaryHelper.h, 68
- buffer
 - Usart_and_buffer, 10
- C
 - evolutionaryHelper.c, 38
- calculateMovementPoints
 - evolutionaryWalking.c, 41
- CCPWrite
 - clksys_driver.c, 13
 - clksys_driver.h, 47
- CD
 - evolutionaryHelper.c, 38
- CEA
 - evolutionaryHelper.c, 38
- CLKSYS_AutoCalibration_Disable
 - clksys_driver.h, 46
- CLKSYS_AutoCalibration_Enable
 - clksys_driver.c, 13
 - clksys_driver.h, 47
- CLKSYS_Configuration_Lock
 - clksys_driver.c, 14
 - clksys_driver.h, 47
- CLKSYS_Disable
 - clksys_driver.c, 14
 - clksys_driver.h, 48
- clksys_driver.c, 11
 - CCPWrite, 13
 - CLKSYS_AutoCalibration_Enable, 13
 - CLKSYS_Configuration_Lock, 14

- CLKSYS_Disable, 14
- CLKSYS_Main_ClockSource_-
 - Select, 14
- CLKSYS_PLL_Config, 14
- CLKSYS_Prescalers_Config, 15
- CLKSYS_RTC_ClockSource_-
 - Enable, 15
- CLKSYS_XOSC_Config, 15
- CLKSYS_XOSC_-
 - FailureDetection_Enable, 16
- clksys_driver.h
 - CCPWrite, 47
 - CLKSYS_AutoCalibration_Disable, 46
 - CLKSYS_AutoCalibration_Enable, 47
 - CLKSYS_Configuration_Lock, 47
 - CLKSYS_Disable, 48
 - CLKSYS_Enable, 46
 - CLKSYS_IsReady, 46
 - CLKSYS_Main_ClockSource_-
 - Select, 48
 - CLKSYS_PLL_Config, 48
 - CLKSYS_Prescalers_Config, 49
 - CLKSYS_RTC_ClockSource_-
 - Disable, 47
 - CLKSYS_RTC_ClockSource_-
 - Enable, 49
 - CLKSYS_XOSC_Config, 49
 - CLKSYS_XOSC_-
 - FailureDetection_Enable, 50
- CLKSYS_Enable
 - clksys_driver.h, 46
- CLKSYS_IsReady
 - clksys_driver.h, 46
- CLKSYS_Main_ClockSource_Select
 - clksys_driver.c, 14
 - clksys_driver.h, 48
- CLKSYS_PLL_Config
 - clksys_driver.c, 14
 - clksys_driver.h, 48
- CLKSYS_Prescalers_Config
 - clksys_driver.c, 15
 - clksys_driver.h, 49
- CLKSYS_RTC_ClockSource_Disable
 - clksys_driver.h, 47
- CLKSYS_RTC_ClockSource_Enable
 - clksys_driver.c, 15
- clksys_driver.h, 49
- CLKSYS_XOSC_Config
 - clksys_driver.c, 15
 - clksys_driver.h, 49
- CLKSYS_XOSC_FailureDetection_-
 - Enable
 - clksys_driver.c, 16
 - clksys_driver.h, 50
- COM_ACK
 - communication.h, 53
- COM_ACTION
 - communication.h, 53
- COM_ANGLE
 - communication.h, 53
- COM_BRDCAST_ID
 - communication.h, 53
- COM_byteArrayToDouble
 - communication.c, 18
 - communication.h, 53
- COM_CONF_FOOT
 - communication.h, 53
- COM_CONF_GLOB
 - communication.h, 53
- COM_CONF_HIP
 - communication.h, 53
- COM_CONF_KNEE
 - communication.h, 53
- COM_CONF_LEFT
 - communication.h, 53
- COM_CONF_RIGHT
 - communication.h, 53
- COM_doubleToByteArray
 - communication.c, 19
 - communication.h, 54
- COM_ERR_ANGLE_LIMIT
 - communication.h, 53
- COM_ERR_DEFAULT_ERROR
 - communication.h, 53
- COM_ERR_POINT_OUT_OF_-
 - BOUNDS
 - communication.h, 53
- COM_getAngleFromPacket
 - communication.c, 19
 - communication.h, 54
- COM_getChecksum
 - communication.c, 19
- COM_getCpuID
 - communication.c, 19
 - communication.h, 54
- COM_getPointFromPacket

- communication.c, 20
- communication.h, 54
- COM_getSpeedFromPacket
 - communication.c, 20
 - communication.h, 54
- COM_IS_ALIVE
 - communication.h, 53
- COM_isAlive
 - communication.c, 20
 - communication.h, 55
- COM_isFoot
 - communication.c, 20
 - communication.h, 55
- COM_isGlobal
 - communication.c, 21
 - communication.h, 55
- COM_isHip
 - communication.c, 21
 - communication.h, 56
- COM_isKnee
 - communication.c, 21
 - communication.h, 56
- COM_isLeftLeg
 - communication.c, 21
 - communication.h, 56
- COM_isRightLeg
 - communication.c, 22
 - communication.h, 56
- COM_MASTER
 - communication.h, 53
- COM_NAK
 - communication.h, 53
- COM_NOCPUID
 - communication.h, 53
- COM_POINT
 - communication.h, 53
- COM_receive
 - communication.c, 22
 - communication.h, 57
- COM_requestStatus
 - communication.c, 22
 - communication.h, 57
- COM_send
 - communication.c, 23
 - communication.h, 57
- COM_sendACK
 - communication.c, 23
 - communication.h, 58
- COM_sendAction
 - communication.c, 23
 - communication.h, 58
- COM_sendAngle
 - communication.c, 24
 - communication.h, 58
- COM_sendNAK
 - communication.c, 24
 - communication.h, 59
- COM_sendPoint
 - communication.c, 24
 - communication.h, 59
- COM_sendPointAndSpeed
 - communication.c, 25
 - communication.h, 59
- COM_SLAVE1B
 - communication.h, 53
- COM_SLAVE3F
 - communication.h, 53
- COM_SPEED
 - communication.h, 53
- COM_START_BYTE
 - communication.c, 18
- COM_STATUS
 - communication.h, 53
- communication.c, 16
 - COM_byteArrayToDouble, 18
 - COM_doubleToByteArray, 19
 - COM_getAngleFromPacket, 19
 - COM_getChecksum, 19
 - COM_getCpuID, 19
 - COM_getPointFromPacket, 20
 - COM_getSpeedFromPacket, 20
 - COM_isAlive, 20
 - COM_isFoot, 20
 - COM_isGlobal, 21
 - COM_isHip, 21
 - COM_isKnee, 21
 - COM_isLeftLeg, 21
 - COM_isRightLeg, 22
 - COM_receive, 22
 - COM_requestStatus, 22
 - COM_send, 23
 - COM_sendACK, 23
 - COM_sendAction, 23
 - COM_sendAngle, 24
 - COM_sendNAK, 24
 - COM_sendPoint, 24
 - COM_sendPointAndSpeed, 25
 - COM_START_BYTE, 18
- communication.h
 - COM_ACK, 53

- COM_ACTION, 53
- COM_ANGLE, 53
- COM_BRDCAST_ID, 53
- COM_byteArrayToDouble, 53
- COM_CONF_FOOT, 53
- COM_CONF_GLOB, 53
- COM_CONF_HIP, 53
- COM_CONF_KNEE, 53
- COM_CONF_LEFT, 53
- COM_CONF_RIGHT, 53
- COM_doubleToByteArray, 54
- COM_ERR_ANGLE_LIMIT, 53
- COM_ERR_DEFAULT_ERROR, 53
- COM_ERR_POINT_OUT_OF_BOUNDS, 53
- COM_getAngleFromPacket, 54
- COM_getCpuID, 54
- COM_getPointFromPacket, 54
- COM_getSpeedFromPacket, 54
- COM_IS_ALIVE, 53
- COM_isAlive, 55
- COM_isFoot, 55
- COM_isGlobal, 55
- COM_isHip, 56
- COM_isKnee, 56
- COM_isLeftLeg, 56
- COM_isRightLeg, 56
- COM_MASTER, 53
- COM_NAK, 53
- COM_NOCPUID, 53
- COM_POINT, 53
- COM_receive, 57
- COM_requestStatus, 57
- COM_send, 57
- COM_sendACK, 58
- COM_sendAction, 58
- COM_sendAngle, 58
- COM_sendNAK, 59
- COM_sendPoint, 59
- COM_sendPointAndSpeed, 59
- COM_SLAVE1B, 53
- COM_SLAVE3F, 53
- COM_SPEED, 53
- COM_STATUS, 53
- copyPoint
 - evolutionaryWalking.c, 41
- cpuID
 - evolutionaryWalking.c, 41
- D
 - evolutionaryHelper.c, 38
- datatypes.h
 - DT_bool, 62
 - DT_byte, 62
 - DT_char, 62
 - DT_cmd, 62
 - DT_double, 62
 - DT_int, 62
 - DT_RESULT_BUFFER_SIZE, 61
 - DT_size, 62
 - DT_type, 62
- DEBUG
 - utils.h, 93
- DEBUG_BYTE
 - utils.h, 93
- DEBUG_ON
 - utils.h, 93
- DFB
 - evolutionaryHelper.c, 38
- DIST_DZ
 - kinematics.c, 101
- DIST_FE
 - kinematics.c, 101
- DIST_HK
 - kinematics.c, 101
- DIST_KF
 - kinematics.c, 101
- DNX_BRDCAST_ID
 - dynamixel.h, 63
- DNX_convertAngle
 - dynamixel.c, 28
- DNX_correctAngles
 - dynamixel.c, 29
- DNX_getAngle
 - dynamixel.c, 29
 - dynamixel.h, 63
- DNX_getChecksum
 - dynamixel.c, 29
 - dynamixel.h, 63
- DNX_getConnectedIDs
 - dynamixel.c, 29
 - dynamixel.h, 64
- DNX_getLed
 - dynamixel.c, 30
 - dynamixel.h, 64
- DNX_getSpeed
 - dynamixel.c, 30
 - dynamixel.h, 64
- DNX_receive

- dynamixel.c, 30
- dynamixel.h, 64
- DNX_send
 - dynamixel.c, 30
 - dynamixel.h, 65
- DNX_sendAction
 - dynamixel.c, 31
 - dynamixel.h, 65
- DNX_setAngle
 - dynamixel.c, 31
 - dynamixel.h, 65
- DNX_setAngleAndSpeed
 - dynamixel.c, 31
 - dynamixel.h, 66
- DNX_setId
 - dynamixel.c, 32
 - dynamixel.h, 66
- DNX_setLed
 - dynamixel.c, 32
 - dynamixel.h, 66
- DNX_setSpeed
 - dynamixel.c, 32
 - dynamixel.h, 66
- doStep
 - evolutionaryWalking.c, 41
- doStepMove
 - evolutionaryWalking.c, 41
- dreIntLevel
 - Usart_and_buffer, 10
- DT_bool
 - datatypes.h, 62
- DT_byte
 - datatypes.h, 62
- DT_char
 - datatypes.h, 62
- DT_cmd
 - datatypes.h, 62
- DT_double
 - datatypes.h, 62
- DT_half_circle, 4
 - sqr_r, 4
- DT_individuum, 4
 - F, 5
 - G, 5
 - S, 5
- DT_int
 - datatypes.h, 62
- DT_leg, 5
 - foot, 5
 - hip, 5
 - knee, 6
 - trans, 6
- DT_lin_func, 6
 - m, 6
 - n, 6
- DT_point, 7
 - x, 7
 - y, 7
 - z, 7
- DT_RESULT_BUFFER_SIZE
 - datatypes.h, 61
- DT_servo, 7
 - act_value, 8
 - id, 8
 - set_value, 8
- DT_size
 - datatypes.h, 62
- DT_transformation, 8
 - x, 8
 - y, 8
 - zRotation, 8
- DT_type
 - datatypes.h, 62
- DT_vector, 9
 - x, 9
 - y, 9
- dynamixel.c, 25
 - ACT, 28
 - ALR_SHUTDOWN, 28
 - BD, 28
 - DNX_convertAngle, 28
 - DNX_correctAngles, 29
 - DNX_getAngle, 29
 - DNX_getChecksum, 29
 - DNX_getConnectedIDs, 29
 - DNX_getLed, 30
 - DNX_getSpeed, 30
 - DNX_receive, 30
 - DNX_send, 30
 - DNX_sendAction, 31
 - DNX_setAngle, 31
 - DNX_setAngleAndSpeed, 31
 - DNX_setId, 32
 - DNX_setLed, 32
 - DNX_setSpeed, 32
 - GL_POS, 28
 - ID, 28
 - LED, 28
 - MAX_TMP, 28
 - MV_SPEED, 28

- PING, 28
- PRT_POS, 28
- PRT_SPEED, 28
- PRT_TMP, 28
- RD_DATA, 28
- REG_WR, 28
- RESET, 28
- START_BYTE, 28
- STS_RT_LVL, 28
- SYC_WR, 28
- WR_DATA, 28
- dynamixel.h
 - DNX_BRDCAST_ID, 63
 - DNX_getAngle, 63
 - DNX_getChecksum, 63
 - DNX_getConnectedIDs, 64
 - DNX_getLed, 64
 - DNX_getSpeed, 64
 - DNX_receive, 64
 - DNX_send, 65
 - DNX_sendAction, 65
 - DNX_setAngle, 65
 - DNX_setAngleAndSpeed, 66
 - DNX_setId, 66
 - DNX_setLed, 66
 - DNX_setSpeed, 66
- E
 - evolutionaryHelper.c, 38
- evolutionaryAlgorithm
 - evolutionaryAlgorithm.c, 34
 - evolutionaryAlgorithm.h, 67
- evolutionaryAlgorithm.c, 32
 - bestSelection, 34
 - evolutionaryAlgorithm, 34
 - fitnessproportionalSelection, 34
 - generatePoint, 34
 - generatePopulation, 34
 - getIsectFromIndividuum, 34
 - getPointFromIndividuum, 34
 - getRandomNumber, 34
 - getScores, 34
 - gleichverteilte_reellwertige_mutation, 34
 - mutation, 34
 - printPopulation, 34
 - recombination, 34
 - uniformCrossover, 34
 - X_MAX, 34
 - X_MIN, 34
 - Y_MAX, 34
 - Y_MIN, 34
- evolutionaryAlgorithm.h
 - evolutionaryAlgorithm, 67
 - getIsectFromIndividuum, 67
 - getPointFromIndividuum, 67
- evolutionaryCalculation
 - evolutionaryWalking.c, 41
- evolutionaryHelper.c, 34
 - A, 38
 - AB, 38
 - B, 38
 - bubblesort, 37
 - C, 38
 - CD, 38
 - CEA, 38
 - D, 38
 - DFB, 38
 - E, 38
 - F, 38
 - f_circ, 37
 - f_lin, 37
 - FIRST_VALUE, 36
 - G, 38
 - getDistance, 37
 - getFunctionOfPoints, 37
 - getNearerPoint, 37
 - initEvoAlg, 37
 - initFunctions, 37
 - initPoints, 37
 - isBetweenPoints, 38
 - isectLinCirc, 38
 - isectLinFuncs, 38
 - isInArea, 38
 - isVectorialPoint, 38
 - max, 38
 - min, 38
 - NO_VALUE, 36
 - scorePoint, 38
 - SECOND_VALUE, 36
- evolutionaryHelper.h
 - bubblesort, 68
 - getDistance, 68
 - getFunctionOfPoints, 68
 - initEvoAlg, 68
 - isInArea, 68
 - max, 68
 - min, 68
 - scorePoint, 68
 - Z, 68

- evolutionaryWalking.c, 38
 - calculateMovementPoints, 41
 - copyPoint, 41
 - cpuID, 41
 - doStep, 41
 - doStepMove, 41
 - evolutionaryCalculation, 41
 - init_pMpSpMiddle, 41
 - initConf, 41
 - invertVector, 41
 - isectM, 41
 - isectS, 41
 - leg_l, 41
 - leg_r, 41
 - main, 41
 - master, 41
 - MasterActive, 41
 - MasterInactive, 41
 - midM, 41
 - midS, 41
 - NO_OFFSET, 41
 - OFFSET, 41
 - pM, 41
 - pMiddle, 41
 - prepareStepMove, 41
 - pS, 41
 - SlavesActive, 41
 - SlavesInactive, 41
 - switchLegs, 41
 - TEST_ON, 41
 - TripodGaitMove, 41
 - waitForButton3, 41
- F
 - DT_individuum, 5
 - evolutionaryHelper.c, 38
- f_circ
 - evolutionaryHelper.c, 37
- F_CPU
 - avr_compiler.h, 43
 - main.c, 103
- f_lin
 - evolutionaryHelper.c, 37
- FIRST_VALUE
 - evolutionaryHelper.c, 36
- fitnessproportionalSelection
 - evolutionaryAlgorithm.c, 34
- foot
 - DT_leg, 5
- G
 - DT_individuum, 5
 - evolutionaryHelper.c, 38
- generatePoint
 - evolutionaryAlgorithm.c, 34
- generatePopulation
 - evolutionaryAlgorithm.c, 34
- getDistance
 - evolutionaryHelper.c, 37
 - evolutionaryHelper.h, 68
- getFunctionOfPoints
 - evolutionaryHelper.c, 37
 - evolutionaryHelper.h, 68
- getIsectFromIndividuum
 - evolutionaryAlgorithm.c, 34
 - evolutionaryAlgorithm.h, 67
- getNearerPoint
 - evolutionaryHelper.c, 37
- getPointFromIndividuum
 - evolutionaryAlgorithm.c, 34
 - evolutionaryAlgorithm.h, 67
- getRandomNumber
 - evolutionaryAlgorithm.c, 34
- getScores
 - evolutionaryAlgorithm.c, 34
- GL_POS
 - dynamixel.c, 28
- gleichverteilte_reellwertige_mutation
 - evolutionaryAlgorithm.c, 34
- hip
 - DT_leg, 5
- ID
 - dynamixel.c, 28
- id
 - DT_servo, 8
- include/avr_compiler.h, 41
- include/clksys_driver.h, 43
- include/communication.h, 50
- include/datatypes.h, 60
- include/dynamixel.h, 62
- include/evolutionaryAlgorithm.h, 67
- include/evolutionaryHelper.h, 67
- include/kinematics.h, 68
- include/movement.h, 70
- include/remote.h, 76
- include/usart_driver.h, 80
- include/utils.h, 92
- include/xmega.h, 96

- init_pMpSpMiddle
 - evolutionaryWalking.c, 41
- initConf
 - evolutionaryWalking.c, 41
- initEvoAlg
 - evolutionaryHelper.c, 37
 - evolutionaryHelper.h, 68
- initFunctions
 - evolutionaryHelper.c, 37
- initPoints
 - evolutionaryHelper.c, 37
- invertVector
 - evolutionaryWalking.c, 41
- isBetweenPoints
 - evolutionaryHelper.c, 38
- isectLinCirc
 - evolutionaryHelper.c, 38
- isectLinFuncs
 - evolutionaryHelper.c, 38
- isectM
 - evolutionaryWalking.c, 41
- isectS
 - evolutionaryWalking.c, 41
- isInArea
 - evolutionaryHelper.c, 38
 - evolutionaryHelper.h, 68
- ISR
 - xmega.c, 130, 131
- isVectorialPoint
 - evolutionaryHelper.c, 38
- KIN_calcDH
 - kinematics.c, 102
 - kinematics.h, 69
- KIN_calcLocalPoint
 - kinematics.c, 102
 - kinematics.h, 69
- KIN_calcServos
 - kinematics.c, 102
 - kinematics.h, 70
- KIN_COLUMNS
 - kinematics.h, 69
- KIN_makeMovement
 - kinematics.h, 70
- KIN_ROWS
 - kinematics.h, 69
- KIN_setTransMat
 - kinematics.c, 102
 - kinematics.h, 70
- kinematics.c, 100
 - DIST_DZ, 101
 - DIST_FE, 101
 - DIST_HK, 101
 - DIST_KF, 101
 - KIN_calcDH, 102
 - KIN_calcLocalPoint, 102
 - KIN_calcServos, 102
 - KIN_setTransMat, 102
- kinematics.h
 - KIN_calcDH, 69
 - KIN_calcLocalPoint, 69
 - KIN_calcServos, 70
 - KIN_COLUMNS, 69
 - KIN_makeMovement, 70
 - KIN_ROWS, 69
 - KIN_setTransMat, 70
- knee
 - DT_leg, 6
- lastPacketLength
 - Usart_and_buffer, 10
- LED
 - dynamixel.c, 28
- leg_l
 - evolutionaryWalking.c, 41
- leg_r
 - evolutionaryWalking.c, 41
- m
 - DT_lin_func, 6
- main
 - evolutionaryWalking.c, 41
- main.c, 103
 - F_CPU, 103
 - TEST_OFF, 103
- master
 - evolutionaryWalking.c, 41
- MasterActive
 - evolutionaryWalking.c, 41
- MasterInactive
 - evolutionaryWalking.c, 41
- max
 - evolutionaryHelper.c, 38
 - evolutionaryHelper.h, 68
- MAX_TMP
 - dynamixel.c, 28
- midM
 - evolutionaryWalking.c, 41
- midS
 - evolutionaryWalking.c, 41

- min
 - evolutionaryHelper.c, 38
 - evolutionaryHelper.h, 68
- movement.c, 103
 - MV_action, 105
 - MV_doInitPosition, 105
 - MV_DST_X, 105
 - MV_getPntForCpuSide, 105
 - MV_masterCheckAlive, 106
 - MV_point, 106
 - MV_pointAndSpeed, 106
 - MV_slave, 106
 - MV_slaveAngle, 107
 - MV_slavePoint, 107
 - MV_slavePointAndSpeed, 107
 - MV_slaveStatus, 108
 - MV_switchLegs, 108
- movement.h
 - MV_action, 72
 - MV_doInitPosition, 72
 - MV_DST_X, 72
 - MV_DST_Y, 72
 - MV_getPntForCpuSide, 72
 - MV_masterCheckAlive, 73
 - MV_point, 73
 - MV_pointAndSpeed, 73
 - MV_slave, 73
 - MV_slaveAngle, 74
 - MV_slavePoint, 74
 - MV_slavePointAndSpeed, 74
 - MV_slaveStatus, 75
 - MV_switchLegs, 75
- movement4Points.c, 108
 - TEST_OFF, 109
- movementMultiPoints.c, 109
 - TEST_OFF, 109
- mutation
 - evolutionaryAlgorithm.c, 34
- MV_action
 - movement.c, 105
 - movement.h, 72
- MV_doInitPosition
 - movement.c, 105
 - movement.h, 72
- MV_DST_X
 - movement.c, 105
 - movement.h, 72
- MV_DST_Y
 - movement.h, 72
- MV_getPntForCpuSide
 - movement.c, 105
 - movement.h, 72
- MV_masterCheckAlive
 - movement.c, 106
 - movement.h, 73
- MV_point
 - movement.c, 106
 - movement.h, 73
- MV_pointAndSpeed
 - movement.c, 106
 - movement.h, 73
- MV_slave
 - movement.c, 106
 - movement.h, 73
- MV_slaveAngle
 - movement.c, 107
 - movement.h, 74
- MV_slavePoint
 - movement.c, 107
 - movement.h, 74
- MV_slavePointAndSpeed
 - movement.c, 107
 - movement.h, 74
- MV_slaveStatus
 - movement.c, 108
 - movement.h, 75
- MV_SPEED
 - dynamixel.c, 28
- MV_switchLegs
 - movement.c, 108
 - movement.h, 75
- n
 - DT_lin_func, 6
- NO_OFFSET
 - evolutionaryWalking.c, 41
- NO_VALUE
 - evolutionaryHelper.c, 36
- OFFSET
 - evolutionaryWalking.c, 41
- PING
 - dynamixel.c, 28
- pM
 - evolutionaryWalking.c, 41
- pMiddle
 - evolutionaryWalking.c, 41
- port
 - Usart_and_buffer, 10

- prepareStepMove
 - evolutionaryWalking.c, 41
- printPopulation
 - evolutionaryAlgorithm.c, 34
- PRT_POS
 - dynamixel.c, 28
- PRT_SPEED
 - dynamixel.c, 28
- PRT_TMP
 - dynamixel.c, 28
- pS
 - evolutionaryWalking.c, 41
- RD_DATA
 - dynamixel.c, 28
- recombination
 - evolutionaryAlgorithm.c, 34
- REG_WR
 - dynamixel.c, 28
- remote.c, 109
 - B_1, 111
 - B_2, 111
 - B_3, 111
 - B_4, 111
 - B_5, 111
 - B_6, 111
 - B_D, 111
 - B_L, 111
 - B_NON_PRESSED, 111
 - B_R, 111
 - B_U, 111
 - RMT_getCommand, 111
 - RMT_isButton1Pressed, 111
 - RMT_isButton2Pressed, 111
 - RMT_isButton3Pressed, 112
 - RMT_isButton4Pressed, 112
 - RMT_isButton5Pressed, 112
 - RMT_isButton6Pressed, 113
 - RMT_isDownPressed, 113
 - RMT_isLeftPressed, 113
 - RMT_isRightPressed, 113
 - RMT_isUpPressed, 114
 - RMT_NonPressed, 114
 - RMT_receive, 114
- remote.h
 - RMT_getCommand, 77
 - RMT_isButton1Pressed, 77
 - RMT_isButton2Pressed, 77
 - RMT_isButton3Pressed, 77
 - RMT_isButton4Pressed, 78
 - RMT_isButton5Pressed, 78
 - RMT_isButton6Pressed, 78
 - RMT_isDownPressed, 78
 - RMT_isLeftPressed, 79
 - RMT_isRightPressed, 79
 - RMT_isUpPressed, 79
 - RMT_NonPressed, 80
 - RMT_receive, 80
- RESET
 - dynamixel.c, 28
- RMT_getCommand
 - remote.c, 111
 - remote.h, 77
- RMT_isButton1Pressed
 - remote.c, 111
 - remote.h, 77
- RMT_isButton2Pressed
 - remote.c, 111
 - remote.h, 77
- RMT_isButton3Pressed
 - remote.c, 112
 - remote.h, 77
- RMT_isButton4Pressed
 - remote.c, 112
 - remote.h, 78
- RMT_isButton5Pressed
 - remote.c, 112
 - remote.h, 78
- RMT_isButton6Pressed
 - remote.c, 113
 - remote.h, 78
- RMT_isDownPressed
 - remote.c, 113
 - remote.h, 78
- RMT_isLeftPressed
 - remote.c, 113
 - remote.h, 79
- RMT_isRightPressed
 - remote.c, 113
 - remote.h, 79
- RMT_isUpPressed
 - remote.c, 114
 - remote.h, 79
- RMT_NonPressed
 - remote.c, 114
 - remote.h, 80
- RMT_receive
 - remote.c, 114
 - remote.h, 80
- RX

- USART_Buffer, 10
- RX_Head
 - USART_Buffer, 10
- RX_Tail
 - USART_Buffer, 10
- S
 - DT_individuum, 5
- scorePoint
 - evolutionaryHelper.c, 38
 - evolutionaryHelper.h, 68
- SECOND_VALUE
 - evolutionaryHelper.c, 36
- set_value
 - DT_servo, 8
- SlavesActive
 - evolutionaryWalking.c, 41
- SlavesInactive
 - evolutionaryWalking.c, 41
- sqr_r
 - DT_half_circle, 4
- START_BYTE
 - dynamixel.c, 28
- STS_RT_LVL
 - dynamixel.c, 28
- SWITCH_PRESSED
 - xmega.h, 98
- SWITCH_RELEASED
 - xmega.h, 98
- switchLegs
 - evolutionaryWalking.c, 41
- SWITCHMASK
 - xmega.h, 98
- SWITCHPORT
 - xmega.h, 98
- SYC_WR
 - dynamixel.c, 28
- TEST_OFF
 - main.c, 103
 - movement4Points.c, 109
 - movementMultiPoints.c, 109
 - testArithmetic.c, 115
 - testCom.c, 115
 - testCom2.c, 116
 - testCom3.c, 116
 - testDnx.c, 116
 - testEvolutionaryDistanceWalking.c, 117
 - testKin.c, 117
 - testKin2.c, 117
 - testMovement.c, 118
 - testRemote.c, 118
 - testRingBuffer.c, 118
 - testSpeed.c, 119
- TEST_ON
 - evolutionaryWalking.c, 41
- testArithmetic.c, 115
 - TEST_OFF, 115
- testCom.c, 115
 - TEST_OFF, 115
- testCom2.c, 115
 - TEST_OFF, 116
- testCom3.c, 116
 - TEST_OFF, 116
- testDnx.c, 116
 - TEST_OFF, 116
- testEvolutionaryDistanceWalking.c, 116
 - TEST_OFF, 117
- testKin.c, 117
 - TEST_OFF, 117
- testKin2.c, 117
 - TEST_OFF, 117
- testMovement.c, 117
 - TEST_OFF, 118
- testRemote.c, 118
 - TEST_OFF, 118
- testRingBuffer.c, 118
 - TEST_OFF, 118
- testSpeed.c, 118
 - TEST_OFF, 119
- trans
 - DT_leg, 6
- TripodGaitMove
 - evolutionaryWalking.c, 41
- TX
 - USART_Buffer, 10
- TX_Head
 - USART_Buffer, 10
- TX_Tail
 - USART_Buffer, 10
- uniformCrossover
 - evolutionaryAlgorithm.c, 34
- usart
 - Usart_and_buffer, 10
- Usart_and_buffer, 9
 - buffer, 10
 - dreIntLevel, 10
 - lastPacketLength, 10

- port, 10
 - usart, 10
 - USART_Baudrate_Set
 - usart_driver.h, 84
 - USART_Buffer, 10
 - RX, 10
 - RX_Head, 10
 - RX_Tail, 10
 - TX, 10
 - TX_Head, 10
 - TX_Tail, 10
 - USART_Buffer_t
 - usart_driver.h, 88
 - USART_data_t
 - usart_driver.h, 88
 - USART_DataRegEmpty
 - usart_driver.c, 121
 - usart_driver.h, 89
 - USART_DreInterruptLevel_Set
 - usart_driver.h, 85
 - usart_driver.c, 119
 - USART_DataRegEmpty, 121
 - USART_InterruptDriver_-
 - DreInterruptLevel_Set, 121
 - USART_InterruptDriver_Initialize, 121
 - USART_NineBits_GetChar, 122
 - USART_NineBits_PutChar, 122
 - USART_RXBuffer_-
 - checkPointerDiff, 122
 - USART_RXBuffer_GetByte, 123
 - USART_RXBufferData_Available, 123
 - USART_RXComplete, 123
 - USART_TXBuffer_FreeSpace, 123
 - USART_TXBuffer_PutByte, 124
 - usart_driver.h
 - USART_Baudrate_Set, 84
 - USART_Buffer_t, 88
 - USART_data_t, 88
 - USART_DataRegEmpty, 89
 - USART_DreInterruptLevel_Set, 85
 - USART_Format_Set, 85
 - USART_GetChar, 85
 - USART_InterruptDriver_-
 - DreInterruptLevel_Set, 89
 - USART_InterruptDriver_Initialize, 89
 - USART_IsRXComplete, 86
 - USART_IsTXDataRegisterEmpty, 86
 - USART_NineBits_GetChar, 89
 - USART_NineBits_PutChar, 90
 - USART_PutChar, 86
 - USART_RX_BUFFER_MASK, 86
 - USART_RX_BUFFER_SIZE, 86
 - USART_Rx_Disable, 86
 - USART_Rx_Enable, 87
 - USART_RXBuffer_-
 - checkPointerDiff, 90
 - USART_RXBuffer_GetByte, 90
 - USART_RXBufferData_Available, 91
 - USART_RXComplete, 91
 - USART_RxdInterruptLevel_Set, 87
 - USART_SetMode, 87
 - USART_TX_BUFFER_MASK, 87
 - USART_TX_BUFFER_SIZE, 88
 - USART_Tx_Disable, 88
 - USART_Tx_Enable, 88
 - USART_TXBuffer_FreeSpace, 91
 - USART_TXBuffer_PutByte, 91
 - USART_TxdInterruptLevel_Set, 88
- USART_Format_Set
 - usart_driver.h, 85
 - USART_GetChar
 - usart_driver.h, 85
 - USART_InterruptDriver_-
 - DreInterruptLevel_Set
 - usart_driver.c, 121
 - usart_driver.h, 89
 - USART_InterruptDriver_Initialize
 - usart_driver.c, 121
 - usart_driver.h, 89
 - USART_IsRXComplete
 - usart_driver.h, 86
 - USART_IsTXDataRegisterEmpty
 - usart_driver.h, 86
 - USART_NineBits_GetChar
 - usart_driver.c, 122
 - usart_driver.h, 89
 - USART_NineBits_PutChar
 - usart_driver.c, 122
 - usart_driver.h, 90
 - USART_ON
 - utils.c, 125
 - USART_PutChar
 - usart_driver.h, 86
 - USART_RX_BUFFER_MASK

- usart_driver.h, 86
- USART_RX_BUFFER_SIZE
 - usart_driver.h, 86
- USART_Rx_Disable
 - usart_driver.h, 86
- USART_Rx_Enable
 - usart_driver.h, 87
- USART_RXBuffer_checkPointerDiff
 - usart_driver.c, 122
 - usart_driver.h, 90
- USART_RXBuffer_GetByte
 - usart_driver.c, 123
 - usart_driver.h, 90
- USART_RXBufferData_Available
 - usart_driver.c, 123
 - usart_driver.h, 91
- USART_RXComplete
 - usart_driver.c, 123
 - usart_driver.h, 91
- USART_RxdInterruptLevel_Set
 - usart_driver.h, 87
- USART_SetMode
 - usart_driver.h, 87
- USART_TX_BUFFER_MASK
 - usart_driver.h, 87
- USART_TX_BUFFER_SIZE
 - usart_driver.h, 88
- USART_Tx_Disable
 - usart_driver.h, 88
- USART_Tx_Enable
 - usart_driver.h, 88
- USART_TXBuffer_FreeSpace
 - usart_driver.c, 123
 - usart_driver.h, 91
- USART_TXBuffer_PutByte
 - usart_driver.c, 124
 - usart_driver.h, 91
- USART_TxdInterruptLevel_Set
 - usart_driver.h, 88
- utils.c, 124
 - USART_ON, 125
 - UTL_byteToHexChar, 126
 - UTL_getDegree, 126
 - UTL_getPointOfDH, 126
 - UTL_getRadiant, 126
 - UTL_printDebug, 127
 - UTL_printDebugByte, 127
 - UTL_printLeg, 127
 - UTL_printMatrix, 127
 - UTL_printPoint, 127
- UTL_wait, 128
- utils.h
 - DEBUG, 93
 - DEBUG_BYTE, 93
 - DEBUG_ON, 93
 - UTL_byteToHexChar, 93
 - UTL_DEG, 93
 - UTL_getDegree, 93
 - UTL_getPointOfDH, 94
 - UTL_getRadiant, 94
 - UTL_printDebug, 94
 - UTL_printDebugByte, 94
 - UTL_printLeg, 95
 - UTL_printMatrix, 95
 - UTL_printPoint, 95
 - UTL_RAD, 93
 - UTL_wait, 95
- UTL_byteToHexChar
 - utils.c, 126
 - utils.h, 93
- UTL_DEG
 - utils.h, 93
- UTL_getDegree
 - utils.c, 126
 - utils.h, 93
- UTL_getPointOfDH
 - utils.c, 126
 - utils.h, 94
- UTL_getRadiant
 - utils.c, 126
 - utils.h, 94
- UTL_printDebug
 - utils.c, 127
 - utils.h, 94
- UTL_printDebugByte
 - utils.c, 127
 - utils.h, 94
- UTL_printLeg
 - utils.c, 127
 - utils.h, 95
- UTL_printMatrix
 - utils.c, 127
 - utils.h, 95
- UTL_printPoint
 - utils.c, 127
 - utils.h, 95
- UTL_RAD
 - utils.h, 93
- UTL_wait
 - utils.c, 128

- utils.h, 95
- waitForButton3
 - evolutionaryWalking.c, 41
- WR_DATA
 - dynamixel.c, 28
- x
 - DT_point, 7
 - DT_transformation, 8
 - DT_vector, 9
- X_MAX
 - evolutionaryAlgorithm.c, 34
- X_MIN
 - evolutionaryAlgorithm.c, 34
- XM_com_data1
 - xmega.h, 100
- XM_com_data3
 - xmega.h, 100
- XM_debug_data
 - xmega.h, 100
- XM_init_com
 - xmega.c, 131
 - xmega.h, 99
- XM_init_cpu
 - xmega.c, 131
 - xmega.h, 99
- XM_init_dnx
 - xmega.c, 131
 - xmega.h, 99
- XM_init_remote
 - xmega.c, 131
 - xmega.h, 99
- XM_LED_MASK
 - xmega.h, 98
- XM_LED_OFF
 - xmega.h, 98
- XM_LED_ON
 - xmega.h, 98
- XM_LED_TGL
 - xmega.h, 98
- XM_OE_MASK
 - xmega.h, 98
- XM_PORT_COM1
 - xmega.h, 98
- XM_PORT_COM3
 - xmega.h, 98
- XM_PORT_DEBUG
 - xmega.h, 98
- XM_PORT_LED
 - xmega.h, 98
- XM_PORT_REMOTE
 - xmega.h, 98
- XM_PORT_SERVO_L
 - xmega.h, 98
- XM_PORT_SERVO_R
 - xmega.h, 98
- XM_remote_data
 - xmega.h, 100
- XM_servo_data_L
 - xmega.h, 100
- XM_servo_data_R
 - xmega.h, 100
- XM_USART_COM1
 - xmega.h, 98
- XM_USART_COM3
 - xmega.h, 98
- XM_USART_DEBUG
 - xmega.h, 98
- XM_USART_FAILURE
 - xmega.h, 98
- XM_USART_REMOTE
 - xmega.h, 98
- XM_USART_send
 - xmega.c, 131
 - xmega.h, 99
- XM_USART_SERVO_L
 - xmega.h, 99
- XM_USART_SERVO_R
 - xmega.h, 99
- xmega.c, 128
 - ISR, 130, 131
 - XM_init_com, 131
 - XM_init_cpu, 131
 - XM_init_dnx, 131
 - XM_init_remote, 131
 - XM_USART_send, 131
- xmega.h
 - SWITCH_PRESSED, 98
 - SWITCH_RELEASED, 98
 - SWITCHMASK, 98
 - SWITCHPORT, 98
 - XM_com_data1, 100
 - XM_com_data3, 100
 - XM_debug_data, 100
 - XM_init_com, 99
 - XM_init_cpu, 99
 - XM_init_dnx, 99
 - XM_init_remote, 99
 - XM_LED_MASK, 98

- XM_LED_OFF, 98
- XM_LED_ON, 98
- XM_LED_TGL, 98
- XM_OE_MASK, 98
- XM_PORT_COM1, 98
- XM_PORT_COM3, 98
- XM_PORT_DEBUG, 98
- XM_PORT_LED, 98
- XM_PORT_REMOTE, 98
- XM_PORT_SERVO_L, 98
- XM_PORT_SERVO_R, 98
- XM_remote_data, 100
- XM_servo_data_L, 100
- XM_servo_data_R, 100
- XM_USART_COM1, 98
- XM_USART_COM3, 98
- XM_USART_DEBUG, 98
- XM_USART_FAILURE, 98
- XM_USART_REMOTE, 98
- XM_USART_send, 99
- XM_USART_SERVO_L, 99
- XM_USART_SERVO_R, 99

y

- DT_point, 7
- DT_transformation, 8
- DT_vector, 9

Y_MAX

- evolutionaryAlgorithm.c, 34

Y_MIN

- evolutionaryAlgorithm.c, 34

Z

- evolutionaryHelper.h, 68

z

- DT_point, 7

zRotation

- DT_transformation, 8