

libcaer

2.5.0

Generated by Doxygen 1.8.13

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	caer_bias_coarsefine Struct Reference	5
3.1.1	Detailed Description	5
3.2	caer_bias_dynapse Struct Reference	5
3.2.1	Detailed Description	6
3.3	caer_bias_shiftedsources Struct Reference	6
3.3.1	Detailed Description	6
3.4	caer_bias_vdac Struct Reference	7
3.4.1	Detailed Description	7
3.5	caer_davis_info Struct Reference	7
3.5.1	Detailed Description	8
3.5.2	Field Documentation	8
3.5.2.1	deviceString	8
3.6	caer_device_discovery_result Struct Reference	9
3.6.1	Detailed Description	9
3.7	caer_dvs128_info Struct Reference	9
3.7.1	Detailed Description	10
3.7.2	Field Documentation	10
3.7.2.1	deviceString	10
3.8	caer_dynapse_info Struct Reference	10
3.8.1	Detailed Description	11
3.8.2	Field Documentation	11
3.8.2.1	deviceString	11
3.9	caer_edvs_info Struct Reference	11
3.9.1	Detailed Description	11
3.9.2	Field Documentation	12
3.9.2.1	deviceString	12
3.10	caer_filter_dvs_pixel Struct Reference	12
3.10.1	Detailed Description	12

4 File Documentation	13
4.1 devices/davis.h File Reference	13
4.1.1 Detailed Description	23
4.1.2 Macro Definition Documentation	23
4.1.2.1 CAER_DEVICE_DAVIS	23
4.1.2.2 CAER_DEVICE_DAVIS_FX2	24
4.1.2.3 CAER_DEVICE_DAVIS_FX3	24
4.1.2.4 CAER_DEVICE_DAVIS_RPI	24
4.1.2.5 DAVIS128_CONFIG_BIAS_ADCCOMPBP	24
4.1.2.6 DAVIS128_CONFIG_BIAS_ADCREFHIGH	24
4.1.2.7 DAVIS128_CONFIG_BIAS_ADCREFLOW	25
4.1.2.8 DAVIS128_CONFIG_BIAS_AEPDBN	25
4.1.2.9 DAVIS128_CONFIG_BIAS_AEPUXBP	25
4.1.2.10 DAVIS128_CONFIG_BIAS_AEPUYBP	26
4.1.2.11 DAVIS128_CONFIG_BIAS_APSCAS	26
4.1.2.12 DAVIS128_CONFIG_BIAS_APSOEVERFLOWLEVEL	26
4.1.2.13 DAVIS128_CONFIG_BIAS_APSROSFBN	27
4.1.2.14 DAVIS128_CONFIG_BIAS_BIASBUFFER	27
4.1.2.15 DAVIS128_CONFIG_BIAS_COLSELLOWBN	27
4.1.2.16 DAVIS128_CONFIG_BIAS_DACBUFBP	28
4.1.2.17 DAVIS128_CONFIG_BIAS_DIFFBN	28
4.1.2.18 DAVIS128_CONFIG_BIAS_IFREFRBN	28
4.1.2.19 DAVIS128_CONFIG_BIAS_IFTHRBN	29
4.1.2.20 DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN	29
4.1.2.21 DAVIS128_CONFIG_BIAS_LOCALBUFBN	29
4.1.2.22 DAVIS128_CONFIG_BIAS_OFFBN	30
4.1.2.23 DAVIS128_CONFIG_BIAS_ONBN	30
4.1.2.24 DAVIS128_CONFIG_BIAS_PADFOLLBN	30
4.1.2.25 DAVIS128_CONFIG_BIAS_PIXINVBN	31
4.1.2.26 DAVIS128_CONFIG_BIAS_PRBP	31

4.1.2.27	DAVIS128_CONFIG_BIAS_PRSFBP	31
4.1.2.28	DAVIS128_CONFIG_BIAS_READOUTBUFBP	32
4.1.2.29	DAVIS128_CONFIG_BIAS_REFRBP	32
4.1.2.30	DAVIS128_CONFIG_BIAS_SSN	32
4.1.2.31	DAVIS128_CONFIG_BIAS_SSP	33
4.1.2.32	DAVIS128_CONFIG_CHIP_AERNAROW	33
4.1.2.33	DAVIS128_CONFIG_CHIP_ANALOGMUX0	33
4.1.2.34	DAVIS128_CONFIG_CHIP_ANALOGMUX1	33
4.1.2.35	DAVIS128_CONFIG_CHIP_ANALOGMUX2	33
4.1.2.36	DAVIS128_CONFIG_CHIP_BIASMUX0	34
4.1.2.37	DAVIS128_CONFIG_CHIP_DIGITALMUX0	34
4.1.2.38	DAVIS128_CONFIG_CHIP_DIGITALMUX1	34
4.1.2.39	DAVIS128_CONFIG_CHIP_DIGITALMUX2	34
4.1.2.40	DAVIS128_CONFIG_CHIP_DIGITALMUX3	34
4.1.2.41	DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER	34
4.1.2.42	DAVIS128_CONFIG_CHIP_RESETCALIBNEURON	35
4.1.2.43	DAVIS128_CONFIG_CHIP_RESETTESTPIXEL	35
4.1.2.44	DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER	35
4.1.2.45	DAVIS128_CONFIG_CHIP_TYPCALIBNEURON	35
4.1.2.46	DAVIS128_CONFIG_CHIP_USEAOUT	35
4.1.2.47	DAVIS208_CONFIG_BIAS_ADCCOMPBP	35
4.1.2.48	DAVIS208_CONFIG_BIAS_ADCREFHIGH	36
4.1.2.49	DAVIS208_CONFIG_BIAS_ADCREFLOW	36
4.1.2.50	DAVIS208_CONFIG_BIAS_AEPDBN	36
4.1.2.51	DAVIS208_CONFIG_BIAS_AEPUXBP	37
4.1.2.52	DAVIS208_CONFIG_BIAS_AEPUYBP	37
4.1.2.53	DAVIS208_CONFIG_BIAS_APSCAS	37
4.1.2.54	DAVIS208_CONFIG_BIAS_APSOEVERFLOWLEVEL	38
4.1.2.55	DAVIS208_CONFIG_BIAS_APSROSFBN	38
4.1.2.56	DAVIS208_CONFIG_BIAS_BIASBUFFER	38

4.1.2.57	DAVIS208_CONFIG_BIAS_COLSELLOWBN	39
4.1.2.58	DAVIS208_CONFIG_BIAS_DACBUFBP	39
4.1.2.59	DAVIS208_CONFIG_BIAS_DIFFBN	39
4.1.2.60	DAVIS208_CONFIG_BIAS_IFREFRBN	40
4.1.2.61	DAVIS208_CONFIG_BIAS_IFTHRBN	40
4.1.2.62	DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN	40
4.1.2.63	DAVIS208_CONFIG_BIAS_LOCALBUFBN	41
4.1.2.64	DAVIS208_CONFIG_BIAS_OFFBN	41
4.1.2.65	DAVIS208_CONFIG_BIAS_ONBN	41
4.1.2.66	DAVIS208_CONFIG_BIAS_PADFOLLBN	42
4.1.2.67	DAVIS208_CONFIG_BIAS_PIXINBN	42
4.1.2.68	DAVIS208_CONFIG_BIAS_PRBP	42
4.1.2.69	DAVIS208_CONFIG_BIAS_PRSFBP	43
4.1.2.70	DAVIS208_CONFIG_BIAS_READOUTBUFBP	43
4.1.2.71	DAVIS208_CONFIG_BIAS_REFRBP	43
4.1.2.72	DAVIS208_CONFIG_BIAS_REFSS	44
4.1.2.73	DAVIS208_CONFIG_BIAS_REFSSBN	44
4.1.2.74	DAVIS208_CONFIG_BIAS_REGBIASBP	44
4.1.2.75	DAVIS208_CONFIG_BIAS_RESETHIGHPASS	45
4.1.2.76	DAVIS208_CONFIG_BIAS_SSN	45
4.1.2.77	DAVIS208_CONFIG_BIAS_SSP	45
4.1.2.78	DAVIS208_CONFIG_CHIP_AERNAROW	46
4.1.2.79	DAVIS208_CONFIG_CHIP_ANALOGMUX0	46
4.1.2.80	DAVIS208_CONFIG_CHIP_ANALOGMUX1	46
4.1.2.81	DAVIS208_CONFIG_CHIP_ANALOGMUX2	46
4.1.2.82	DAVIS208_CONFIG_CHIP_BIASMUX0	46
4.1.2.83	DAVIS208_CONFIG_CHIP_DIGITALMUX0	46
4.1.2.84	DAVIS208_CONFIG_CHIP_DIGITALMUX1	47
4.1.2.85	DAVIS208_CONFIG_CHIP_DIGITALMUX2	47
4.1.2.86	DAVIS208_CONFIG_CHIP_DIGITALMUX3	47

4.1.2.87	DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER	47
4.1.2.88	DAVIS208_CONFIG_CHIP_RESETCALIBNEURON	47
4.1.2.89	DAVIS208_CONFIG_CHIP_RESETTESTPIXEL	47
4.1.2.90	DAVIS208_CONFIG_CHIP_SELECTBIASREFSS	48
4.1.2.91	DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER	48
4.1.2.92	DAVIS208_CONFIG_CHIP_SELECTHIGHPASS	48
4.1.2.93	DAVIS208_CONFIG_CHIP_SELECTPOSFB	48
4.1.2.94	DAVIS208_CONFIG_CHIP_SELECTPREMPAVG	48
4.1.2.95	DAVIS208_CONFIG_CHIP_SELECTSENSE	48
4.1.2.96	DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON	49
4.1.2.97	DAVIS208_CONFIG_CHIP_USEAOUT	49
4.1.2.98	DAVIS240_CONFIG_BIAS_AEPDBN	49
4.1.2.99	DAVIS240_CONFIG_BIAS_AEPUXBP	49
4.1.2.100	DAVIS240_CONFIG_BIAS_AEPUYBP	50
4.1.2.101	DAVIS240_CONFIG_BIAS_APSCASEPC	50
4.1.2.102	DAVIS240_CONFIG_BIAS_APSEVERFLOWLEVELBN	50
4.1.2.103	DAVIS240_CONFIG_BIAS_APSROSFBN	50
4.1.2.104	DAVIS240_CONFIG_BIAS_BIASBUFFER	51
4.1.2.105	DAVIS240_CONFIG_BIAS_DIFFBN	51
4.1.2.106	DAVIS240_CONFIG_BIAS_DIFFCASBNC	51
4.1.2.107	DAVIS240_CONFIG_BIAS_IFREFRBN	51
4.1.2.108	DAVIS240_CONFIG_BIAS_IFTHRBN	52
4.1.2.109	DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN	52
4.1.2.110	DAVIS240_CONFIG_BIAS_LOCALBUFBN	52
4.1.2.111	DAVIS240_CONFIG_BIAS_OFFBN	52
4.1.2.112	DAVIS240_CONFIG_BIAS_ONBN	53
4.1.2.113	DAVIS240_CONFIG_BIAS_PADFOLLBN	53
4.1.2.114	DAVIS240_CONFIG_BIAS_PIXINVBN	53
4.1.2.115	DAVIS240_CONFIG_BIAS_PRBP	53
4.1.2.116	DAVIS240_CONFIG_BIAS_PRSFBN	54

4.1.2.117 DAVIS240_CONFIG_BIAS_REFRBP	54
4.1.2.118 DAVIS240_CONFIG_BIAS_SSN	54
4.1.2.119 DAVIS240_CONFIG_BIAS_SSP	54
4.1.2.120 DAVIS240_CONFIG_CHIP_AERNAROW	55
4.1.2.121 DAVIS240_CONFIG_CHIP_ANALOGMUX0	55
4.1.2.122 DAVIS240_CONFIG_CHIP_ANALOGMUX1	55
4.1.2.123 DAVIS240_CONFIG_CHIP_ANALOGMUX2	55
4.1.2.124 DAVIS240_CONFIG_CHIP_BIASMUX0	55
4.1.2.125 DAVIS240_CONFIG_CHIP_DIGITALMUX0	55
4.1.2.126 DAVIS240_CONFIG_CHIP_DIGITALMUX1	56
4.1.2.127 DAVIS240_CONFIG_CHIP_DIGITALMUX2	56
4.1.2.128 DAVIS240_CONFIG_CHIP_DIGITALMUX3	56
4.1.2.129 DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER	56
4.1.2.130 DAVIS240_CONFIG_CHIP_RESETCALIBNEURON	56
4.1.2.131 DAVIS240_CONFIG_CHIP_RESETTESTPIXEL	56
4.1.2.132 DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL	57
4.1.2.133 DAVIS240_CONFIG_CHIP_TYPCALIBNEURON	57
4.1.2.134 DAVIS240_CONFIG_CHIP_USEAOUT	57
4.1.2.135 DAVIS346_CONFIG_BIAS_ADCCOMPBP	57
4.1.2.136 DAVIS346_CONFIG_BIAS_ADCREFHIGH	57
4.1.2.137 DAVIS346_CONFIG_BIAS_ADCREFLOW	58
4.1.2.138 DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE	58
4.1.2.139 DAVIS346_CONFIG_BIAS_AEPDBN	58
4.1.2.140 DAVIS346_CONFIG_BIAS_AEPUXBP	59
4.1.2.141 DAVIS346_CONFIG_BIAS_AEPUYBP	59
4.1.2.142 DAVIS346_CONFIG_BIAS_APSCAS	59
4.1.2.143 DAVIS346_CONFIG_BIAS_APSEVERFLOWLEVEL	60
4.1.2.144 DAVIS346_CONFIG_BIAS_APSROSFBN	60
4.1.2.145 DAVIS346_CONFIG_BIAS_BIASBUFFER	60
4.1.2.146 DAVIS346_CONFIG_BIAS_COLSELLOWBN	61

4.1.2.147 DAVIS346_CONFIG_BIAS_DACBUFBP	61
4.1.2.148 DAVIS346_CONFIG_BIAS_DIFFBN	61
4.1.2.149 DAVIS346_CONFIG_BIAS_IFREFRBN	62
4.1.2.150 DAVIS346_CONFIG_BIAS_IFTHRBN	62
4.1.2.151 DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN	62
4.1.2.152 DAVIS346_CONFIG_BIAS_LOCALBUFBN	63
4.1.2.153 DAVIS346_CONFIG_BIAS_OFFBN	63
4.1.2.154 DAVIS346_CONFIG_BIAS_ONBN	63
4.1.2.155 DAVIS346_CONFIG_BIAS_PADFOLLBN	64
4.1.2.156 DAVIS346_CONFIG_BIAS_PIXINVBN	64
4.1.2.157 DAVIS346_CONFIG_BIAS_PRBP	64
4.1.2.158 DAVIS346_CONFIG_BIAS_PRSFBP	65
4.1.2.159 DAVIS346_CONFIG_BIAS_READOUTBUFBP	65
4.1.2.160 DAVIS346_CONFIG_BIAS_REFRBP	65
4.1.2.161 DAVIS346_CONFIG_BIAS_SSN	66
4.1.2.162 DAVIS346_CONFIG_BIAS_SSP	66
4.1.2.163 DAVIS346_CONFIG_CHIP_AERNAROW	66
4.1.2.164 DAVIS346_CONFIG_CHIP_ANALOGMUX0	66
4.1.2.165 DAVIS346_CONFIG_CHIP_ANALOGMUX1	67
4.1.2.166 DAVIS346_CONFIG_CHIP_ANALOGMUX2	67
4.1.2.167 DAVIS346_CONFIG_CHIP_BIASMUX0	67
4.1.2.168 DAVIS346_CONFIG_CHIP_DIGITALMUX0	67
4.1.2.169 DAVIS346_CONFIG_CHIP_DIGITALMUX1	67
4.1.2.170 DAVIS346_CONFIG_CHIP_DIGITALMUX2	67
4.1.2.171 DAVIS346_CONFIG_CHIP_DIGITALMUX3	68
4.1.2.172 DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER	68
4.1.2.173 DAVIS346_CONFIG_CHIP_RESETCALIBNEURON	68
4.1.2.174 DAVIS346_CONFIG_CHIP_RESETTESTPIXEL	68
4.1.2.175 DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER	68
4.1.2.176 DAVIS346_CONFIG_CHIP_TESTADC	68

4.1.2.177 DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON	69
4.1.2.178 DAVIS346_CONFIG_CHIP_USEAOUT	69
4.1.2.179 DAVIS640_CONFIG_BIAS_ADCCOMPBP	69
4.1.2.180 DAVIS640_CONFIG_BIAS_ADCREFHIGH	69
4.1.2.181 DAVIS640_CONFIG_BIAS_ADCREFLOW	70
4.1.2.182 DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE	70
4.1.2.183 DAVIS640_CONFIG_BIAS_AEPDBN	70
4.1.2.184 DAVIS640_CONFIG_BIAS_AEPUXBP	71
4.1.2.185 DAVIS640_CONFIG_BIAS_AEPUYBP	71
4.1.2.186 DAVIS640_CONFIG_BIAS_APSCAS	71
4.1.2.187 DAVIS640_CONFIG_BIAS_APSOEVERFLOWLEVEL	72
4.1.2.188 DAVIS640_CONFIG_BIAS_APSROSFBN	72
4.1.2.189 DAVIS640_CONFIG_BIAS_BIASBUFFER	72
4.1.2.190 DAVIS640_CONFIG_BIAS_COLSELLOWBN	73
4.1.2.191 DAVIS640_CONFIG_BIAS_DACBUFBP	73
4.1.2.192 DAVIS640_CONFIG_BIAS_DIFFBN	73
4.1.2.193 DAVIS640_CONFIG_BIAS_IFREFRBN	74
4.1.2.194 DAVIS640_CONFIG_BIAS_IFTHRBN	74
4.1.2.195 DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN	74
4.1.2.196 DAVIS640_CONFIG_BIAS_LOCALBUFBN	75
4.1.2.197 DAVIS640_CONFIG_BIAS_OFFBN	75
4.1.2.198 DAVIS640_CONFIG_BIAS_ONBN	75
4.1.2.199 DAVIS640_CONFIG_BIAS_PADFOLLBN	76
4.1.2.200 DAVIS640_CONFIG_BIAS_PIXINBN	76
4.1.2.201 DAVIS640_CONFIG_BIAS_PRBP	76
4.1.2.202 DAVIS640_CONFIG_BIAS_PRSFBN	77
4.1.2.203 DAVIS640_CONFIG_BIAS_READOUTBUFBP	77
4.1.2.204 DAVIS640_CONFIG_BIAS_REFRBN	77
4.1.2.205 DAVIS640_CONFIG_BIAS_SSN	78
4.1.2.206 DAVIS640_CONFIG_BIAS_SSP	78

4.1.2.207 DAVIS640_CONFIG_CHIP_AERNAROW	78
4.1.2.208 DAVIS640_CONFIG_CHIP_ANALOGMUX0	78
4.1.2.209 DAVIS640_CONFIG_CHIP_ANALOGMUX1	79
4.1.2.210 DAVIS640_CONFIG_CHIP_ANALOGMUX2	79
4.1.2.211 DAVIS640_CONFIG_CHIP_BIASMUX0	79
4.1.2.212 DAVIS640_CONFIG_CHIP_DIGITALMUX0	79
4.1.2.213 DAVIS640_CONFIG_CHIP_DIGITALMUX1	79
4.1.2.214 DAVIS640_CONFIG_CHIP_DIGITALMUX2	79
4.1.2.215 DAVIS640_CONFIG_CHIP_DIGITALMUX3	80
4.1.2.216 DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER	80
4.1.2.217 DAVIS640_CONFIG_CHIP_RESETCALIBNEURON	80
4.1.2.218 DAVIS640_CONFIG_CHIP_RESETTESTPIXEL	80
4.1.2.219 DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER	80
4.1.2.220 DAVIS640_CONFIG_CHIP_TESTADC	80
4.1.2.221 DAVIS640_CONFIG_CHIP_TYPCALIBNEURON	81
4.1.2.222 DAVIS640_CONFIG_CHIP_USEAOUT	81
4.1.2.223 DAVIS_APS_ROI_REGIONS_MAX	81
4.1.2.224 DAVIS_CHIP_DAVIS128	81
4.1.2.225 DAVIS_CHIP_DAVIS208	81
4.1.2.226 DAVIS_CHIP_DAVIS240A	81
4.1.2.227 DAVIS_CHIP_DAVIS240B	81
4.1.2.228 DAVIS_CHIP_DAVIS240C	82
4.1.2.229 DAVIS_CHIP_DAVIS346A	82
4.1.2.230 DAVIS_CHIP_DAVIS346B	82
4.1.2.231 DAVIS_CHIP_DAVIS346C	82
4.1.2.232 DAVIS_CHIP_DAVIS640	82
4.1.2.233 DAVIS_CHIP_DAVISRGB	82
4.1.2.234 DAVIS_CONFIG_APS	82
4.1.2.235 DAVIS_CONFIG_APS_ADC_TEST_MODE	83
4.1.2.236 DAVIS_CONFIG_APS_AUTOEXPOSURE	83

4.1.2.237 DAVIS_CONFIG_APS_COLOR_FILTER	83
4.1.2.238 DAVIS_CONFIG_APS_COLUMN_SETTLE	83
4.1.2.239 DAVIS_CONFIG_APS_END_COLUMN_0	83
4.1.2.240 DAVIS_CONFIG_APS_END_COLUMN_1	83
4.1.2.241 DAVIS_CONFIG_APS_END_COLUMN_2	83
4.1.2.242 DAVIS_CONFIG_APS_END_COLUMN_3	84
4.1.2.243 DAVIS_CONFIG_APS_END_ROW_0	84
4.1.2.244 DAVIS_CONFIG_APS_END_ROW_1	84
4.1.2.245 DAVIS_CONFIG_APS_END_ROW_2	84
4.1.2.246 DAVIS_CONFIG_APS_END_ROW_3	84
4.1.2.247 DAVIS_CONFIG_APS_EXPOSURE	84
4.1.2.248 DAVIS_CONFIG_APS_FRAME_DELAY	84
4.1.2.249 DAVIS_CONFIG_APS_GLOBAL_SHUTTER	85
4.1.2.250 DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER	85
4.1.2.251 DAVIS_CONFIG_APS_HAS_INTERNAL_ADC	85
4.1.2.252 DAVIS_CONFIG_APS_HAS_QUAD_ROI	85
4.1.2.253 DAVIS_CONFIG_APS_NULL_SETTLE	85
4.1.2.254 DAVIS_CONFIG_APS_ORIENTATION_INFO	85
4.1.2.255 DAVIS_CONFIG_APS_RAMP_RESET	86
4.1.2.256 DAVIS_CONFIG_APS_RAMP_SHORT_RESET	86
4.1.2.257 DAVIS_CONFIG_APS_RESET_READ	86
4.1.2.258 DAVIS_CONFIG_APS_RESET_SETTLE	86
4.1.2.259 DAVIS_CONFIG_APS_ROI0_ENABLED	86
4.1.2.260 DAVIS_CONFIG_APS_ROI1_ENABLED	86
4.1.2.261 DAVIS_CONFIG_APS_ROI2_ENABLED	86
4.1.2.262 DAVIS_CONFIG_APS_ROI3_ENABLED	87
4.1.2.263 DAVIS_CONFIG_APS_ROW_SETTLE	87
4.1.2.264 DAVIS_CONFIG_APS_RUN	87
4.1.2.265 DAVIS_CONFIG_APS_SAMPLE_ENABLE	87
4.1.2.266 DAVIS_CONFIG_APS_SAMPLE_SETTLE	87

4.1.2.267 DAVIS_CONFIG_APS_SIZE_COLUMNS	87
4.1.2.268 DAVIS_CONFIG_APS_SIZE_ROWS	87
4.1.2.269 DAVIS_CONFIG_APS_SNAPSHOT	88
4.1.2.270 DAVIS_CONFIG_APS_START_COLUMN_0	88
4.1.2.271 DAVIS_CONFIG_APS_START_COLUMN_1	88
4.1.2.272 DAVIS_CONFIG_APS_START_COLUMN_2	88
4.1.2.273 DAVIS_CONFIG_APS_START_COLUMN_3	88
4.1.2.274 DAVIS_CONFIG_APS_START_ROW_0	88
4.1.2.275 DAVIS_CONFIG_APS_START_ROW_1	89
4.1.2.276 DAVIS_CONFIG_APS_START_ROW_2	89
4.1.2.277 DAVIS_CONFIG_APS_START_ROW_3	89
4.1.2.278 DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL	89
4.1.2.279 DAVIS_CONFIG_BIAS	89
4.1.2.280 DAVIS_CONFIG_CHIP	89
4.1.2.281 DAVIS_CONFIG_DDRAER	90
4.1.2.282 DAVIS_CONFIG_DDRAER_ACK_DELAY	90
4.1.2.283 DAVIS_CONFIG_DDRAER_REQ_DELAY	90
4.1.2.284 DAVIS_CONFIG_DDRAER_RUN	90
4.1.2.285 DAVIS_CONFIG_DVS	90
4.1.2.286 DAVIS_CONFIG_DVS_ACK_DELAY_COLUMN	90
4.1.2.287 DAVIS_CONFIG_DVS_ACK_DELAY_ROW	90
4.1.2.288 DAVIS_CONFIG_DVS_ACK_EXTENSION_COLUMN	91
4.1.2.289 DAVIS_CONFIG_DVS_ACK_EXTENSION_ROW	91
4.1.2.290 DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL	91
4.1.2.291 DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY	91
4.1.2.292 DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_TIME	91
4.1.2.293 DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN	91
4.1.2.294 DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW	92
4.1.2.295 DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN	92
4.1.2.296 DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW	92

4.1.2.297 DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN	92
4.1.2.298 DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW	92
4.1.2.299 DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN	92
4.1.2.300 DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW	92
4.1.2.301 DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN	93
4.1.2.302 DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW	93
4.1.2.303 DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN	93
4.1.2.304 DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW	93
4.1.2.305 DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN	93
4.1.2.306 DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW	93
4.1.2.307 DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN	93
4.1.2.308 DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW	94
4.1.2.309 DAVIS_CONFIG_DVS_FILTER_PIXEL_AUTO_TRAIN	94
4.1.2.310 DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD	94
4.1.2.311 DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD_TIME	94
4.1.2.312 DAVIS_CONFIG_DVS_FILTER_ROI_END_COLUMN	94
4.1.2.313 DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW	94
4.1.2.314 DAVIS_CONFIG_DVS_FILTER_ROI_START_COLUMN	94
4.1.2.315 DAVIS_CONFIG_DVS_FILTER_ROI_START_ROW	95
4.1.2.316 DAVIS_CONFIG_DVS_FILTER_ROW_ONLY_EVENTS	95
4.1.2.317 DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER	95
4.1.2.318 DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER	95
4.1.2.319 DAVIS_CONFIG_DVS_HAS_ROI_FILTER	95
4.1.2.320 DAVIS_CONFIG_DVS_HAS_STATISTICS	95
4.1.2.321 DAVIS_CONFIG_DVS_HAS_TEST_EVENT_GENERATOR	96
4.1.2.322 DAVIS_CONFIG_DVS_ORIENTATION_INFO	96
4.1.2.323 DAVIS_CONFIG_DVS_RUN	96
4.1.2.324 DAVIS_CONFIG_DVS_SIZE_COLUMNS	96
4.1.2.325 DAVIS_CONFIG_DVS_SIZE_ROWS	96
4.1.2.326 DAVIS_CONFIG_DVS_STATISTICS_EVENTS_COLUMN	96

4.1.2.327 DAVIS_CONFIG_DVS_STATISTICS_EVENTS_DROPPED	97
4.1.2.328 DAVIS_CONFIG_DVS_STATISTICS_EVENTS_ROW	97
4.1.2.329 DAVIS_CONFIG_DVS_STATISTICS_FILTERED_BACKGROUND_ACTIVITY	97
4.1.2.330 DAVIS_CONFIG_DVS_STATISTICS_FILTERED_PIXELS	97
4.1.2.331 DAVIS_CONFIG_DVS_STATISTICS_FILTERED_REFRACTORY_PERIOD	97
4.1.2.332 DAVIS_CONFIG_DVS_TEST_EVENT_GENERATOR_ENABLE	97
4.1.2.333 DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL	98
4.1.2.334 DAVIS_CONFIG_EXTINPUT	98
4.1.2.335 DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES	98
4.1.2.336 DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES1	98
4.1.2.337 DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES2	98
4.1.2.338 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH	98
4.1.2.339 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH1	99
4.1.2.340 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH2	99
4.1.2.341 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY	99
4.1.2.342 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY1	99
4.1.2.343 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY2	99
4.1.2.344 DAVIS_CONFIG_EXTINPUT_DETECT_PULSES	99
4.1.2.345 DAVIS_CONFIG_EXTINPUT_DETECT_PULSES1	100
4.1.2.346 DAVIS_CONFIG_EXTINPUT_DETECT_PULSES2	100
4.1.2.347 DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES	100
4.1.2.348 DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES1	100
4.1.2.349 DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES2	100
4.1.2.350 DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_FALLING_EDGE	100
4.1.2.351 DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_RISING_EDGE	101
4.1.2.352 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL	101
4.1.2.353 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH	101
4.1.2.354 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY	101
4.1.2.355 DAVIS_CONFIG_EXTINPUT_GENERATE_USE_CUSTOM_SIGNAL	101
4.1.2.356 DAVIS_CONFIG_EXTINPUT_HAS_EXTRA_DETECTORS	101

4.1.2.357 DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR	102
4.1.2.358 DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR	102
4.1.2.359 DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR1	102
4.1.2.360 DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR2	102
4.1.2.361 DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR	102
4.1.2.362 DAVIS_CONFIG_IMU	102
4.1.2.363 DAVIS_CONFIG_IMU_ACCEL_DLPF	103
4.1.2.364 DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE	103
4.1.2.365 DAVIS_CONFIG_IMU_ACCEL_STANDBY	103
4.1.2.366 DAVIS_CONFIG_IMU_GYRO_DLPF	103
4.1.2.367 DAVIS_CONFIG_IMU_GYRO_FULL_SCALE	103
4.1.2.368 DAVIS_CONFIG_IMU_GYRO_STANDBY	104
4.1.2.369 DAVIS_CONFIG_IMU_LP_CYCLE	104
4.1.2.370 DAVIS_CONFIG_IMU_LP_WAKEUP	104
4.1.2.371 DAVIS_CONFIG_IMU_ORIENTATION_INFO	104
4.1.2.372 DAVIS_CONFIG_IMU_RUN	104
4.1.2.373 DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER	104
4.1.2.374 DAVIS_CONFIG_IMU_TEMP_STANDBY	105
4.1.2.375 DAVIS_CONFIG_MUX	105
4.1.2.376 DAVIS_CONFIG_MUX_DROP_APS_ON_TRANSFER_STALL	105
4.1.2.377 DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL	105
4.1.2.378 DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL	105
4.1.2.379 DAVIS_CONFIG_MUX_DROP_IMU_ON_TRANSFER_STALL	105
4.1.2.380 DAVIS_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE	106
4.1.2.381 DAVIS_CONFIG_MUX_HAS_STATISTICS	106
4.1.2.382 DAVIS_CONFIG_MUX_RUN	106
4.1.2.383 DAVIS_CONFIG_MUX_STATISTICS_APS_DROPPED	106
4.1.2.384 DAVIS_CONFIG_MUX_STATISTICS_DVS_DROPPED	106
4.1.2.385 DAVIS_CONFIG_MUX_STATISTICS_EXTINPUT_DROPPED	106
4.1.2.386 DAVIS_CONFIG_MUX_STATISTICS_IMU_DROPPED	107

4.1.2.387 DAVIS_CONFIG_MUX_TIMESTAMP_RESET	107
4.1.2.388 DAVIS_CONFIG_MUX_TIMESTAMP_RUN	107
4.1.2.389 DAVIS_CONFIG_SYSINFO	107
4.1.2.390 DAVIS_CONFIG_SYSINFO_ADC_CLOCK	107
4.1.2.391 DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER	107
4.1.2.392 DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER	108
4.1.2.393 DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK	108
4.1.2.394 DAVIS_CONFIG_SYSINFO_LOGIC_VERSION	108
4.1.2.395 DAVIS_CONFIG_USB	108
4.1.2.396 DAVIS_CONFIG_USB_EARLY_PACKET_DELAY	108
4.1.2.397 DAVIS_CONFIG_USB_RUN	108
4.1.2.398 DAVISRGB_CONFIG_APS_GSFDRESET	109
4.1.2.399 DAVISRGB_CONFIG_APS_GSPDRESET	109
4.1.2.400 DAVISRGB_CONFIG_APS_GSRESETFALL	109
4.1.2.401 DAVISRGB_CONFIG_APS_GSTXFALL	109
4.1.2.402 DAVISRGB_CONFIG_APS_RSFDSETTLE	109
4.1.2.403 DAVISRGB_CONFIG_APS_TRANSFER	109
4.1.2.404 DAVISRGB_CONFIG_BIAS_ADCCOMPBP	110
4.1.2.405 DAVISRGB_CONFIG_BIAS_ADCREFHIGH	110
4.1.2.406 DAVISRGB_CONFIG_BIAS_ADCREFLOW	110
4.1.2.407 DAVISRGB_CONFIG_BIAS_ADCTESTVOLTAGE	111
4.1.2.408 DAVISRGB_CONFIG_BIAS_AEPDBN	111
4.1.2.409 DAVISRGB_CONFIG_BIAS_AEPUXBP	111
4.1.2.410 DAVISRGB_CONFIG_BIAS_AEPUYBP	112
4.1.2.411 DAVISRGB_CONFIG_BIAS_APSCAS	112
4.1.2.412 DAVISRGB_CONFIG_BIAS_APSROSFBN	112
4.1.2.413 DAVISRGB_CONFIG_BIAS_ARRAYBIASBUFFERBN	113
4.1.2.414 DAVISRGB_CONFIG_BIAS_ARRAYLOGICBUFFERBN	113
4.1.2.415 DAVISRGB_CONFIG_BIAS_BIASBUFFER	113
4.1.2.416 DAVISRGB_CONFIG_BIAS_DACBUFBP	114

4.1.2.417 DAVISRGB_CONFIG_BIAS_DIFFBN	114
4.1.2.418 DAVISRGB_CONFIG_BIAS_FALLTIMEBN	114
4.1.2.419 DAVISRGB_CONFIG_BIAS_GND07	115
4.1.2.420 DAVISRGB_CONFIG_BIAS_IFREFRBN	115
4.1.2.421 DAVISRGB_CONFIG_BIAS_IFTHRBN	115
4.1.2.422 DAVISRGB_CONFIG_BIAS_LCOLTIMEOUTBN	116
4.1.2.423 DAVISRGB_CONFIG_BIAS_LOCALBUFBN	116
4.1.2.424 DAVISRGB_CONFIG_BIAS_OFFBN	116
4.1.2.425 DAVISRGB_CONFIG_BIAS_ONBN	117
4.1.2.426 DAVISRGB_CONFIG_BIAS_OVG1LO	117
4.1.2.427 DAVISRGB_CONFIG_BIAS_OVG2LO	117
4.1.2.428 DAVISRGB_CONFIG_BIAS_PADFOLLBN	118
4.1.2.429 DAVISRGB_CONFIG_BIAS_PIXINVBN	118
4.1.2.430 DAVISRGB_CONFIG_BIAS_PRBP	118
4.1.2.431 DAVISRGB_CONFIG_BIAS_PRSFBP	119
4.1.2.432 DAVISRGB_CONFIG_BIAS_READOUTBUFBP	119
4.1.2.433 DAVISRGB_CONFIG_BIAS_REFRBP	119
4.1.2.434 DAVISRGB_CONFIG_BIAS_RISETIMEBP	120
4.1.2.435 DAVISRGB_CONFIG_BIAS_SSN	120
4.1.2.436 DAVISRGB_CONFIG_BIAS_SSP	120
4.1.2.437 DAVISRGB_CONFIG_BIAS_TX2OVG2HI	121
4.1.2.438 DAVISRGB_CONFIG_CHIP_ADJUSTOVG1LO	121
4.1.2.439 DAVISRGB_CONFIG_CHIP_ADJUSTOVG2LO	121
4.1.2.440 DAVISRGB_CONFIG_CHIP_ADJUSTTX2OVG2HI	121
4.1.2.441 DAVISRGB_CONFIG_CHIP_AERNAROW	121
4.1.2.442 DAVISRGB_CONFIG_CHIP_ANALOGMUX0	122
4.1.2.443 DAVISRGB_CONFIG_CHIP_ANALOGMUX1	122
4.1.2.444 DAVISRGB_CONFIG_CHIP_ANALOGMUX2	122
4.1.2.445 DAVISRGB_CONFIG_CHIP_BIASMUX0	122
4.1.2.446 DAVISRGB_CONFIG_CHIP_DIGITALMUX0	122

4.1.2.447	DAVISRGB_CONFIG_CHIP_DIGITALMUX1	122
4.1.2.448	DAVISRGB_CONFIG_CHIP_DIGITALMUX2	123
4.1.2.449	DAVISRGB_CONFIG_CHIP_DIGITALMUX3	123
4.1.2.450	DAVISRGB_CONFIG_CHIP_RESETCALIBNEURON	123
4.1.2.451	DAVISRGB_CONFIG_CHIP_RESETTESTPIXEL	123
4.1.2.452	DAVISRGB_CONFIG_CHIP_SELECTGRAYCOUNTER	123
4.1.2.453	DAVISRGB_CONFIG_CHIP_TESTADC	123
4.1.2.454	DAVISRGB_CONFIG_CHIP_TYPENCALIBNEURON	124
4.1.2.455	DAVISRGB_CONFIG_CHIP_USEAOUT	124
4.1.2.456	IS_DAVIS128	124
4.1.2.457	IS_DAVIS208	124
4.1.2.458	IS_DAVIS240	124
4.1.2.459	IS_DAVIS240A	124
4.1.2.460	IS_DAVIS240B	125
4.1.2.461	IS_DAVIS240C	125
4.1.2.462	IS_DAVIS346	125
4.1.2.463	IS_DAVIS346A	125
4.1.2.464	IS_DAVIS346B	125
4.1.2.465	IS_DAVIS346C	125
4.1.2.466	IS_DAVIS640	126
4.1.2.467	IS_DAVISRGB	126
4.1.3	Enumeration Type Documentation	126
4.1.3.1	caer_bias_shiftedsource_operating_mode	126
4.1.3.2	caer_bias_shiftedsource_voltage_level	126
4.1.4	Function Documentation	127
4.1.4.1	caerBiasCoarseFineGenerate()	127
4.1.4.2	caerBiasCoarseFineParse()	127
4.1.4.3	caerBiasShiftedSourceGenerate()	127
4.1.4.4	caerBiasShiftedSourceParse()	128
4.1.4.5	caerBiasVDACGenerate()	128

4.1.4.6	caerBiasVDACParse()	129
4.1.4.7	caerDavisInfoGet()	129
4.1.4.8	caerDavisROIConfigure()	129
4.2	devices/device.h File Reference	130
4.2.1	Detailed Description	131
4.2.2	Macro Definition Documentation	131
4.2.2.1	CAER_HOST_CONFIG_DATAEXCHANGE	131
4.2.2.2	CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING	131
4.2.2.3	CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE	131
4.2.2.4	CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS	131
4.2.2.5	CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS	131
4.2.2.6	CAER_HOST_CONFIG_LOG	132
4.2.2.7	CAER_HOST_CONFIG_LOG_LEVEL	132
4.2.2.8	CAER_HOST_CONFIG_PACKETS	132
4.2.2.9	CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL	132
4.2.2.10	CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_PACKET_SIZE	132
4.2.2.11	CAER_SUPPORTED_DEVICES_NUMBER	132
4.2.3	Typedef Documentation	132
4.2.3.1	caerDeviceHandle	133
4.2.4	Function Documentation	133
4.2.4.1	caerDeviceClose()	133
4.2.4.2	caerDeviceConfigGet()	133
4.2.4.3	caerDeviceConfigGet64()	134
4.2.4.4	caerDeviceConfigSet()	134
4.2.4.5	caerDeviceDataGet()	135
4.2.4.6	caerDeviceDataStart()	135
4.2.4.7	caerDeviceDataStop()	136
4.2.4.8	caerDeviceSendDefaultConfig()	136
4.3	devices/device_discover.h File Reference	136
4.3.1	Detailed Description	137

4.3.2	Macro Definition Documentation	137
4.3.2.1	CAER_DEVICE_DISCOVER_ALL	137
4.3.3	Typedef Documentation	137
4.3.3.1	caerDeviceDiscoveryResult	137
4.3.4	Function Documentation	137
4.3.4.1	caerDeviceDiscover()	137
4.3.4.2	caerDeviceDiscoverOpen()	138
4.4	devices/dvs128.h File Reference	138
4.4.1	Detailed Description	139
4.4.2	Macro Definition Documentation	139
4.4.2.1	CAER_DEVICE_DVS128	139
4.4.2.2	DVS128_CONFIG_BIAS	139
4.4.2.3	DVS128_CONFIG_BIAS_CAS	140
4.4.2.4	DVS128_CONFIG_BIAS_DIFF	140
4.4.2.5	DVS128_CONFIG_BIAS_DIFFOFF	140
4.4.2.6	DVS128_CONFIG_BIAS_DIFFON	140
4.4.2.7	DVS128_CONFIG_BIAS_FOLL	140
4.4.2.8	DVS128_CONFIG_BIAS_INJGND	140
4.4.2.9	DVS128_CONFIG_BIAS_PR	140
4.4.2.10	DVS128_CONFIG_BIAS_PUX	141
4.4.2.11	DVS128_CONFIG_BIAS_PUY	141
4.4.2.12	DVS128_CONFIG_BIAS_REFR	141
4.4.2.13	DVS128_CONFIG_BIAS_REQ	141
4.4.2.14	DVS128_CONFIG_BIAS_REQPD	141
4.4.2.15	DVS128_CONFIG_DVS	141
4.4.2.16	DVS128_CONFIG_DVS_ARRAY_RESET	141
4.4.2.17	DVS128_CONFIG_DVS_RUN	142
4.4.2.18	DVS128_CONFIG_DVS_TIMESTAMP_RESET	142
4.4.2.19	DVS128_CONFIG_DVS_TS_MASTER	142
4.4.3	Function Documentation	142

4.4.3.1	caerDVS128InfoGet()	142
4.5	devices/dynapse.h File Reference	142
4.5.1	Detailed Description	147
4.5.2	Macro Definition Documentation	147
4.5.2.1	CAER_DEVICE_DYNAPSE	147
4.5.2.2	DYNAPSE_CHIP_DYNAPSE	148
4.5.2.3	DYNAPSE_CONFIG_AER	148
4.5.2.4	DYNAPSE_CONFIG_AER_ACK_DELAY	148
4.5.2.5	DYNAPSE_CONFIG_AER_ACK_EXTENSION	148
4.5.2.6	DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONTROL	148
4.5.2.7	DYNAPSE_CONFIG_AER_HAS_STATISTICS	148
4.5.2.8	DYNAPSE_CONFIG_AER_RUN	148
4.5.2.9	DYNAPSE_CONFIG_AER_STATISTICS_EVENTS	149
4.5.2.10	DYNAPSE_CONFIG_AER_STATISTICS_EVENTS_DROPPED	149
4.5.2.11	DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_STALL	149
4.5.2.12	DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P	149
4.5.2.13	DYNAPSE_CONFIG_CHIP	149
4.5.2.14	DYNAPSE_CONFIG_CHIP_CONTENT	149
4.5.2.15	DYNAPSE_CONFIG_CHIP_ID	150
4.5.2.16	DYNAPSE_CONFIG_CHIP_REQ_DELAY	150
4.5.2.17	DYNAPSE_CONFIG_CHIP_REQ_EXTENSION	150
4.5.2.18	DYNAPSE_CONFIG_CHIP_RUN	150
4.5.2.19	DYNAPSE_CONFIG_CLEAR_CAM	150
4.5.2.20	DYNAPSE_CONFIG_DEFAULT_SRAM	150
4.5.2.21	DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY	150
4.5.2.22	DYNAPSE_CONFIG_MONITOR_NEU	151
4.5.2.23	DYNAPSE_CONFIG_MUX	151
4.5.2.24	DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRANSFER_STALL	151
4.5.2.25	DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE	151
4.5.2.26	DYNAPSE_CONFIG_MUX_HAS_STATISTICS	151

4.5.2.27	DYNAPSE_CONFIG_MUX_RUN	151
4.5.2.28	DYNAPSE_CONFIG_MUX_STATISTICS_AER_DROPPED	152
4.5.2.29	DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET	152
4.5.2.30	DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN	152
4.5.2.31	DYNAPSE_CONFIG_POISSONSPIKEGEN	152
4.5.2.32	DYNAPSE_CONFIG_POISSONSPIKEGEN_CHIPID	152
4.5.2.33	DYNAPSE_CONFIG_POISSONSPIKEGEN_RUN	152
4.5.2.34	DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEADDRESS	152
4.5.2.35	DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEDATA	153
4.5.2.36	DYNAPSE_CONFIG_SPIKEGEN	153
4.5.2.37	DYNAPSE_CONFIG_SPIKEGEN_BASEADDR	153
4.5.2.38	DYNAPSE_CONFIG_SPIKEGEN_ISI	153
4.5.2.39	DYNAPSE_CONFIG_SPIKEGEN_ISIBASE	153
4.5.2.40	DYNAPSE_CONFIG_SPIKEGEN_REPEAT	153
4.5.2.41	DYNAPSE_CONFIG_SPIKEGEN_RUN	153
4.5.2.42	DYNAPSE_CONFIG_SPIKEGEN_STIMCOUNT	154
4.5.2.43	DYNAPSE_CONFIG_SPIKEGEN_VARMODE	154
4.5.2.44	DYNAPSE_CONFIG_SRAM	154
4.5.2.45	DYNAPSE_CONFIG_SRAM_ADDRESS	154
4.5.2.46	DYNAPSE_CONFIG_SRAM_BURSTMODE	154
4.5.2.47	DYNAPSE_CONFIG_SRAM_DIRECTION_POS	154
4.5.2.48	DYNAPSE_CONFIG_SRAM_READ	155
4.5.2.49	DYNAPSE_CONFIG_SRAM_READDATA	155
4.5.2.50	DYNAPSE_CONFIG_SRAM_RWCOMMAND	155
4.5.2.51	DYNAPSE_CONFIG_SRAM_WRITE	155
4.5.2.52	DYNAPSE_CONFIG_SRAM_WRITEDATA	155
4.5.2.53	DYNAPSE_CONFIG_SYNAPSERECONFIG	155
4.5.2.54	DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPSELECT	156
4.5.2.55	DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOBAKERNEL	156
4.5.2.56	DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN	156

4.5.2.57	DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAMBASEADDR	156
4.5.2.58	DYNAPSE_CONFIG_SYNAPSERECONFIG_USESRAMKERNELS	156
4.5.2.59	DYNAPSE_CONFIG_SYSINFO	156
4.5.2.60	DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER	157
4.5.2.61	DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER	157
4.5.2.62	DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK	157
4.5.2.63	DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION	157
4.5.2.64	DYNAPSE_CONFIG_TAU1_RESET	157
4.5.2.65	DYNAPSE_CONFIG_TAU2_RESET	157
4.5.2.66	DYNAPSE_CONFIG_TAU2_SET	158
4.5.2.67	DYNAPSE_CONFIG_USB	158
4.5.2.68	DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY	158
4.5.2.69	DYNAPSE_CONFIG_USB_RUN	158
4.5.2.70	DYNAPSE_X4BOARD_COREX	158
4.5.2.71	DYNAPSE_X4BOARD_COREY	158
4.5.2.72	DYNAPSE_X4BOARD_NEUX	159
4.5.2.73	DYNAPSE_X4BOARD_NEUY	159
4.5.2.74	DYNAPSE_X4BOARD_NUMCHIPS	159
4.5.3	Function Documentation	159
4.5.3.1	caerBiasDynapseGenerate()	159
4.5.3.2	caerBiasDynapseParse()	159
4.5.3.3	caerDynapseCoreAddrToNeuronId()	160
4.5.3.4	caerDynapseCoreXYToNeuronId()	160
4.5.3.5	caerDynapseGenerateCamBits()	161
4.5.3.6	caerDynapseGenerateSramBits()	161
4.5.3.7	caerDynapseInfoGet()	162
4.5.3.8	caerDynapseSendDataToUSB()	162
4.5.3.9	caerDynapseSpikeEventFromXY()	163
4.5.3.10	caerDynapseSpikeEventGetX()	163
4.5.3.11	caerDynapseSpikeEventGetY()	163

4.5.3.12	caerDynapseWriteCam()	164
4.5.3.13	caerDynapseWritePoissonSpikeRate()	164
4.5.3.14	caerDynapseWriteSram()	165
4.5.3.15	caerDynapseWriteSramN()	166
4.5.3.16	caerDynapseWriteSramWords()	166
4.6	devices/edvs.h File Reference	167
4.6.1	Detailed Description	168
4.6.2	Macro Definition Documentation	168
4.6.2.1	CAER_DEVICE_EDVS	168
4.6.2.2	EDVS_CONFIG_BIAS	168
4.6.2.3	EDVS_CONFIG_BIAS_CAS	168
4.6.2.4	EDVS_CONFIG_BIAS_DIFF	168
4.6.2.5	EDVS_CONFIG_BIAS_DIFFOFF	168
4.6.2.6	EDVS_CONFIG_BIAS_DIFFON	168
4.6.2.7	EDVS_CONFIG_BIAS_FOLL	169
4.6.2.8	EDVS_CONFIG_BIAS_INJGND	169
4.6.2.9	EDVS_CONFIG_BIAS_PR	169
4.6.2.10	EDVS_CONFIG_BIAS_PUX	169
4.6.2.11	EDVS_CONFIG_BIAS_PUY	169
4.6.2.12	EDVS_CONFIG_BIAS_REFR	169
4.6.2.13	EDVS_CONFIG_BIAS_REQ	169
4.6.2.14	EDVS_CONFIG_BIAS_REQPD	170
4.6.2.15	EDVS_CONFIG_DVS	170
4.6.2.16	EDVS_CONFIG_DVS_RUN	170
4.6.2.17	EDVS_CONFIG_DVS_TIMESTAMP_RESET	170
4.6.3	Function Documentation	170
4.6.3.1	caerEDVSInfoGet()	170
4.7	devices/serial.h File Reference	171
4.7.1	Detailed Description	171
4.7.2	Macro Definition Documentation	171

4.7.2.1	CAER_HOST_CONFIG_SERIAL	171
4.7.2.2	CAER_HOST_CONFIG_SERIAL_BAUD_RATE_12M	171
4.7.2.3	CAER_HOST_CONFIG_SERIAL_BAUD_RATE_2M	172
4.7.2.4	CAER_HOST_CONFIG_SERIAL_BAUD_RATE_4M	172
4.7.2.5	CAER_HOST_CONFIG_SERIAL_BAUD_RATE_8M	172
4.7.2.6	CAER_HOST_CONFIG_SERIAL_READ_SIZE	172
4.7.3	Function Documentation	172
4.7.3.1	caerDeviceOpenSerial()	172
4.8	devices/usb.h File Reference	173
4.8.1	Detailed Description	173
4.8.2	Macro Definition Documentation	173
4.8.2.1	CAER_HOST_CONFIG_USB	173
4.8.2.2	CAER_HOST_CONFIG_USB_BUFFER_NUMBER	173
4.8.2.3	CAER_HOST_CONFIG_USB_BUFFER_SIZE	174
4.8.3	Function Documentation	174
4.8.3.1	caerDeviceOpen()	174
4.9	events/common.h File Reference	174
4.9.1	Detailed Description	176
4.9.2	Macro Definition Documentation	176
4.9.2.1	CAER_DEFAULT_EVENT_TYPES_COUNT	176
4.9.2.2	CAER_EVENT_PACKET_HEADER_SIZE	177
4.9.2.3	CAER_ITERATOR_ALL_END	177
4.9.2.4	CAER_ITERATOR_ALL_START	177
4.9.2.5	CAER_ITERATOR_VALID_END	177
4.9.2.6	CAER_ITERATOR_VALID_START	177
4.9.2.7	TS_OVERFLOW_SHIFT	178
4.9.2.8	VALID_MARK_MASK	178
4.9.2.9	VALID_MARK_SHIFT	178
4.9.3	Typedef Documentation	178
4.9.3.1	caerEventPacketHeader	178

4.9.4	Enumeration Type Documentation	178
4.9.4.1	caer_default_event_types	178
4.9.5	Function Documentation	179
4.9.5.1	caerEventPacketAllocate()	179
4.9.5.2	caerEventPacketAppend()	179
4.9.5.3	caerEventPacketClean()	180
4.9.5.4	caerEventPacketClear()	180
4.9.5.5	caerEventPacketCopy()	180
4.9.5.6	caerEventPacketCopyOnlyEvents()	181
4.9.5.7	caerEventPacketCopyOnlyValidEvents()	181
4.9.5.8	caerEventPacketEquals()	182
4.9.5.9	caerEventPacketGetDataSize()	182
4.9.5.10	caerEventPacketGetDataSizeEvents()	182
4.9.5.11	caerEventPacketGetSize()	183
4.9.5.12	caerEventPacketGetSizeEvents()	183
4.9.5.13	caerEventPacketGrow()	183
4.9.5.14	caerEventPacketHeaderGetEventCapacity()	184
4.9.5.15	caerEventPacketHeaderGetEventNumber()	184
4.9.5.16	caerEventPacketHeaderGetEventSize()	185
4.9.5.17	caerEventPacketHeaderGetEventSource()	185
4.9.5.18	caerEventPacketHeaderGetEventTSOffset()	185
4.9.5.19	caerEventPacketHeaderGetEventTSOverflow()	186
4.9.5.20	caerEventPacketHeaderGetEventType()	186
4.9.5.21	caerEventPacketHeaderGetEventValid()	186
4.9.5.22	caerEventPacketHeaderSetEventCapacity()	187
4.9.5.23	caerEventPacketHeaderSetEventNumber()	187
4.9.5.24	caerEventPacketHeaderSetEventSize()	187
4.9.5.25	caerEventPacketHeaderSetEventSource()	188
4.9.5.26	caerEventPacketHeaderSetEventTSOffset()	188
4.9.5.27	caerEventPacketHeaderSetEventTSOverflow()	188

4.9.5.28	<code>caerEventPacketHeaderSetEventType()</code>	189
4.9.5.29	<code>caerEventPacketHeaderSetEventValid()</code>	189
4.9.5.30	<code>caerEventPacketResize()</code>	189
4.9.5.31	<code>caerGenericEventCopy()</code>	190
4.9.5.32	<code>caerGenericEventGetEvent()</code>	190
4.9.5.33	<code>caerGenericEventGetTimestamp()</code>	191
4.9.5.34	<code>caerGenericEventGetTimestamp64()</code>	191
4.9.5.35	<code>caerGenericEventIsValid()</code>	192
4.9.5.36	<code>PACKED_STRUCT()</code>	192
4.10	<code>events/config.h</code> File Reference	192
4.10.1	Detailed Description	194
4.10.2	Macro Definition Documentation	194
4.10.2.1	<code>CAER_CONFIGURATION_CONST_ITERATOR_ALL_START</code>	194
4.10.2.2	<code>CAER_CONFIGURATION_CONST_ITERATOR_VALID_START</code>	194
4.10.2.3	<code>CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_ALL_START</code>	195
4.10.2.4	<code>CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_VALID_START</code>	195
4.10.2.5	<code>CAER_CONFIGURATION_ITERATOR_ALL_END</code>	195
4.10.2.6	<code>CAER_CONFIGURATION_ITERATOR_ALL_START</code>	196
4.10.2.7	<code>CAER_CONFIGURATION_ITERATOR_VALID_END</code>	196
4.10.2.8	<code>CAER_CONFIGURATION_ITERATOR_VALID_START</code>	196
4.10.2.9	<code>CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_END</code>	197
4.10.2.10	<code>CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_START</code>	197
4.10.2.11	<code>CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_END</code>	197
4.10.2.12	<code>CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_START</code>	197
4.10.2.13	<code>CONFIG_MODULE_ADDR_MASK</code>	198
4.10.2.14	<code>CONFIG_MODULE_ADDR_SHIFT</code>	198
4.10.3	Typedef Documentation	198
4.10.3.1	<code>caerConfigurationEvent</code>	198
4.10.3.2	<code>caerConfigurationEventPacket</code>	198
4.10.4	Function Documentation	198

4.10.4.1	caerConfigurationEventGetModuleAddress()	198
4.10.4.2	caerConfigurationEventGetParameter()	199
4.10.4.3	caerConfigurationEventGetParameterAddress()	199
4.10.4.4	caerConfigurationEventGetTimestamp()	199
4.10.4.5	caerConfigurationEventGetTimestamp64()	200
4.10.4.6	caerConfigurationEventInvalidate()	200
4.10.4.7	caerConfigurationEventsIsValid()	201
4.10.4.8	caerConfigurationEventPacketAllocate()	201
4.10.4.9	caerConfigurationEventPacketFromPacketHeader()	201
4.10.4.10	caerConfigurationEventPacketFromPacketHeaderConst()	202
4.10.4.11	caerConfigurationEventPacketGetEvent()	202
4.10.4.12	caerConfigurationEventPacketGetEventConst()	202
4.10.4.13	caerConfigurationEventSetModuleAddress()	203
4.10.4.14	caerConfigurationEventSetParameter()	203
4.10.4.15	caerConfigurationEventSetParameterAddress()	203
4.10.4.16	caerConfigurationEventSetTimestamp()	204
4.10.4.17	caerConfigurationEventValidate()	204
4.10.4.18	PACKED_STRUCT() [1/2]	204
4.10.4.19	PACKED_STRUCT() [2/2]	205
4.11	events/ear.h File Reference	205
4.11.1	Detailed Description	206
4.11.2	Macro Definition Documentation	206
4.11.2.1	CAER_EAR_CONST_ITERATOR_ALL_START	206
4.11.2.2	CAER_EAR_CONST_ITERATOR_VALID_START	207
4.11.2.3	CAER_EAR_CONST_REVERSE_ITERATOR_ALL_START	207
4.11.2.4	CAER_EAR_CONST_REVERSE_ITERATOR_VALID_START	207
4.11.2.5	CAER_EAR_ITERATOR_ALL_END	208
4.11.2.6	CAER_EAR_ITERATOR_ALL_START	208
4.11.2.7	CAER_EAR_ITERATOR_VALID_END	208
4.11.2.8	CAER_EAR_ITERATOR_VALID_START	208

4.11.2.9	CAER_EAR_REVERSE_ITERATOR_ALL_END	209
4.11.2.10	CAER_EAR_REVERSE_ITERATOR_ALL_START	209
4.11.2.11	CAER_EAR_REVERSE_ITERATOR_VALID_END	209
4.11.2.12	CAER_EAR_REVERSE_ITERATOR_VALID_START	209
4.11.2.13	EAR_CHANNEL_MASK	210
4.11.2.14	EAR_CHANNEL_SHIFT	210
4.11.2.15	EAR_FILTER_MASK	210
4.11.2.16	EAR_FILTER_SHIFT	210
4.11.2.17	EAR_MASK	210
4.11.2.18	EAR_NEURON_MASK	210
4.11.2.19	EAR_NEURON_SHIFT	211
4.11.2.20	EAR_SHIFT	211
4.11.3	Typedef Documentation	211
4.11.3.1	caerEarEvent	211
4.11.3.2	caerEarEventPacket	211
4.11.4	Function Documentation	211
4.11.4.1	caerEarEventGetChannel()	211
4.11.4.2	caerEarEventGetEar()	212
4.11.4.3	caerEarEventGetTimestamp()	212
4.11.4.4	caerEarEventGetTimestamp64()	213
4.11.4.5	caerEarEventInvalidate()	213
4.11.4.6	caerEarEventIsValid()	213
4.11.4.7	caerEarEventPacketAllocate()	214
4.11.4.8	caerEarEventPacketFromPacketHeader()	214
4.11.4.9	caerEarEventPacketFromPacketHeaderConst()	214
4.11.4.10	caerEarEventPacketGetEvent()	216
4.11.4.11	caerEarEventPacketGetEventConst()	216
4.11.4.12	caerEarEventSetChannel()	217
4.11.4.13	caerEarEventSetEar()	217
4.11.4.14	caerEarEventSetTimestamp()	217

4.11.4.15 caerEarEventValidate()	218
4.11.4.16 PACKED_STRUCT() [1/2]	218
4.11.4.17 PACKED_STRUCT() [2/2]	218
4.12 events/frame.h File Reference	218
4.12.1 Detailed Description	221
4.12.2 Macro Definition Documentation	221
4.12.2.1 CAER_FRAME_CONST_ITERATOR_ALL_START	221
4.12.2.2 CAER_FRAME_CONST_ITERATOR_VALID_START	221
4.12.2.3 CAER_FRAME_CONST_REVERSE_ITERATOR_ALL_START	222
4.12.2.4 CAER_FRAME_CONST_REVERSE_ITERATOR_VALID_START	222
4.12.2.5 CAER_FRAME_ITERATOR_ALL_END	222
4.12.2.6 CAER_FRAME_ITERATOR_ALL_START	223
4.12.2.7 CAER_FRAME_ITERATOR_VALID_END	223
4.12.2.8 CAER_FRAME_ITERATOR_VALID_START	223
4.12.2.9 CAER_FRAME_REVERSE_ITERATOR_ALL_END	223
4.12.2.10 CAER_FRAME_REVERSE_ITERATOR_ALL_START	224
4.12.2.11 CAER_FRAME_REVERSE_ITERATOR_VALID_END	224
4.12.2.12 CAER_FRAME_REVERSE_ITERATOR_VALID_START	224
4.12.2.13 FRAME_COLOR_CHANNELS_MASK	225
4.12.2.14 FRAME_COLOR_CHANNELS_SHIFT	225
4.12.2.15 FRAME_COLOR_FILTER_MASK	225
4.12.2.16 FRAME_COLOR_FILTER_SHIFT	225
4.12.2.17 FRAME_ROI_IDENTIFIER_MASK	225
4.12.2.18 FRAME_ROI_IDENTIFIER_SHIFT	226
4.12.3 Typedef Documentation	226
4.12.3.1 caerFrameEvent	226
4.12.3.2 caerFrameEventPacket	226
4.12.4 Enumeration Type Documentation	226
4.12.4.1 caer_frame_event_color_channels	226
4.12.4.2 caer_frame_event_color_filter	227

4.12.5	Function Documentation	227
4.12.5.1	caerFrameEventGetChannelNumber()	227
4.12.5.2	caerFrameEventGetColorFilter()	227
4.12.5.3	caerFrameEventGetExposureLength()	228
4.12.5.4	caerFrameEventGetLengthX()	228
4.12.5.5	caerFrameEventGetLengthY()	228
4.12.5.6	caerFrameEventGetPixel()	230
4.12.5.7	caerFrameEventGetPixelArrayUnsafe()	230
4.12.5.8	caerFrameEventGetPixelArrayUnsafeConst()	231
4.12.5.9	caerFrameEventGetPixelForChannel()	231
4.12.5.10	caerFrameEventGetPixelForChannelUnsafe()	232
4.12.5.11	caerFrameEventGetPixelsMaxIndex()	232
4.12.5.12	caerFrameEventGetPixelsSize()	232
4.12.5.13	caerFrameEventGetPixelUnsafe()	233
4.12.5.14	caerFrameEventGetPositionX()	233
4.12.5.15	caerFrameEventGetPositionY()	234
4.12.5.16	caerFrameEventGetROIIdentifier()	234
4.12.5.17	caerFrameEventGetTimestamp()	234
4.12.5.18	caerFrameEventGetTimestamp64()	235
4.12.5.19	caerFrameEventGetTSEndOfExposure()	235
4.12.5.20	caerFrameEventGetTSEndOfExposure64()	235
4.12.5.21	caerFrameEventGetTSEndOfFrame()	236
4.12.5.22	caerFrameEventGetTSEndOfFrame64()	236
4.12.5.23	caerFrameEventGetTSStartOfExposure()	237
4.12.5.24	caerFrameEventGetTSStartOfExposure64()	237
4.12.5.25	caerFrameEventGetTSStartOfFrame()	237
4.12.5.26	caerFrameEventGetTSStartOfFrame64()	238
4.12.5.27	caerFrameEventInvalidate()	238
4.12.5.28	caerFrameEventIsValid()	239
4.12.5.29	caerFrameEventPacketAllocate()	239

4.12.5.30 caerFrameEventPacketFromPacketHeader()	240
4.12.5.31 caerFrameEventPacketFromPacketHeaderConst()	240
4.12.5.32 caerFrameEventPacketGetEvent()	240
4.12.5.33 caerFrameEventPacketGetEventConst()	241
4.12.5.34 caerFrameEventPacketGetPixelsMaxIndex()	241
4.12.5.35 caerFrameEventPacketGetPixelsSize()	241
4.12.5.36 caerFrameEventSetColorFilter()	243
4.12.5.37 caerFrameEventSetLengthXLengthYChannelNumber()	243
4.12.5.38 caerFrameEventSetPixel()	244
4.12.5.39 caerFrameEventSetPixelForChannel()	244
4.12.5.40 caerFrameEventSetPixelForChannelUnsafe()	244
4.12.5.41 caerFrameEventSetPixelUnsafe()	245
4.12.5.42 caerFrameEventSetPositionX()	245
4.12.5.43 caerFrameEventSetPositionY()	246
4.12.5.44 caerFrameEventSetROIIdentifier()	246
4.12.5.45 caerFrameEventSetTSEndOfExposure()	246
4.12.5.46 caerFrameEventSetTSEndOfFrame()	247
4.12.5.47 caerFrameEventSetTSSStartOfExposure()	247
4.12.5.48 caerFrameEventSetTSSStartOfFrame()	247
4.12.5.49 caerFrameEventValidate()	247
4.12.5.50 PACKED_STRUCT() [1/2]	248
4.12.5.51 PACKED_STRUCT() [2/2]	248
4.13 events/imu6.h File Reference	248
4.13.1 Detailed Description	250
4.13.2 Macro Definition Documentation	250
4.13.2.1 CAER_IMU6_CONST_ITERATOR_ALL_START	250
4.13.2.2 CAER_IMU6_CONST_ITERATOR_VALID_START	250
4.13.2.3 CAER_IMU6_CONST_REVERSE_ITERATOR_ALL_START	251
4.13.2.4 CAER_IMU6_CONST_REVERSE_ITERATOR_VALID_START	251
4.13.2.5 CAER_IMU6_ITERATOR_ALL_END	251

4.13.2.6	CAER_IMU6_ITERATOR_ALL_START	252
4.13.2.7	CAER_IMU6_ITERATOR_VALID_END	252
4.13.2.8	CAER_IMU6_ITERATOR_VALID_START	252
4.13.2.9	CAER_IMU6_REVERSE_ITERATOR_ALL_END	252
4.13.2.10	CAER_IMU6_REVERSE_ITERATOR_ALL_START	253
4.13.2.11	CAER_IMU6_REVERSE_ITERATOR_VALID_END	253
4.13.2.12	CAER_IMU6_REVERSE_ITERATOR_VALID_START	253
4.13.3	Typedef Documentation	253
4.13.3.1	caerIMU6Event	254
4.13.3.2	caerIMU6EventPacket	254
4.13.4	Function Documentation	254
4.13.4.1	caerIMU6EventGetAccelX()	254
4.13.4.2	caerIMU6EventGetAccelY()	254
4.13.4.3	caerIMU6EventGetAccelZ()	255
4.13.4.4	caerIMU6EventGetGyroX()	255
4.13.4.5	caerIMU6EventGetGyroY()	255
4.13.4.6	caerIMU6EventGetGyroZ()	256
4.13.4.7	caerIMU6EventGetTemp()	256
4.13.4.8	caerIMU6EventGetTimestamp()	256
4.13.4.9	caerIMU6EventGetTimestamp64()	257
4.13.4.10	caerIMU6EventInvalidate()	257
4.13.4.11	caerIMU6EventIsValid()	258
4.13.4.12	caerIMU6EventPacketAllocate()	258
4.13.4.13	caerIMU6EventPacketFromPacketHeader()	258
4.13.4.14	caerIMU6EventPacketFromPacketHeaderConst()	259
4.13.4.15	caerIMU6EventPacketGetEvent()	259
4.13.4.16	caerIMU6EventPacketGetEventConst()	260
4.13.4.17	caerIMU6EventSetAccelX()	260
4.13.4.18	caerIMU6EventSetAccelY()	260
4.13.4.19	caerIMU6EventSetAccelZ()	261

4.13.4.20 caerIMU6EventSetGyroX()	261
4.13.4.21 caerIMU6EventSetGyroY()	261
4.13.4.22 caerIMU6EventSetGyroZ()	261
4.13.4.23 caerIMU6EventSetTemp()	262
4.13.4.24 caerIMU6EventSetTimestamp()	262
4.13.4.25 caerIMU6EventValidate()	262
4.13.4.26 PACKED_STRUCT() [1/2]	263
4.13.4.27 PACKED_STRUCT() [2/2]	263
4.14 events/imu9.h File Reference	263
4.14.1 Detailed Description	264
4.14.2 Macro Definition Documentation	265
4.14.2.1 CAER_IMU9_CONST_ITERATOR_ALL_START	265
4.14.2.2 CAER_IMU9_CONST_ITERATOR_VALID_START	265
4.14.2.3 CAER_IMU9_CONST_REVERSE_ITERATOR_ALL_START	266
4.14.2.4 CAER_IMU9_CONST_REVERSE_ITERATOR_VALID_START	266
4.14.2.5 CAER_IMU9_ITERATOR_ALL_END	266
4.14.2.6 CAER_IMU9_ITERATOR_ALL_START	267
4.14.2.7 CAER_IMU9_ITERATOR_VALID_END	267
4.14.2.8 CAER_IMU9_ITERATOR_VALID_START	267
4.14.2.9 CAER_IMU9_REVERSE_ITERATOR_ALL_END	267
4.14.2.10 CAER_IMU9_REVERSE_ITERATOR_ALL_START	268
4.14.2.11 CAER_IMU9_REVERSE_ITERATOR_VALID_END	268
4.14.2.12 CAER_IMU9_REVERSE_ITERATOR_VALID_START	268
4.14.3 Typedef Documentation	268
4.14.3.1 caerIMU9Event	269
4.14.3.2 caerIMU9EventPacket	269
4.14.4 Function Documentation	269
4.14.4.1 caerIMU9EventGetAccelX()	269
4.14.4.2 caerIMU9EventGetAccelY()	269
4.14.4.3 caerIMU9EventGetAccelZ()	270

4.14.4.4	caerIMU9EventGetCompX()	270
4.14.4.5	caerIMU9EventGetCompY()	270
4.14.4.6	caerIMU9EventGetCompZ()	271
4.14.4.7	caerIMU9EventGetGyroX()	271
4.14.4.8	caerIMU9EventGetGyroY()	271
4.14.4.9	caerIMU9EventGetGyroZ()	273
4.14.4.10	caerIMU9EventGetTemp()	273
4.14.4.11	caerIMU9EventGetTimestamp()	273
4.14.4.12	caerIMU9EventGetTimestamp64()	274
4.14.4.13	caerIMU9EventInvalidate()	274
4.14.4.14	caerIMU9EventIsValid()	275
4.14.4.15	caerIMU9EventPacketAllocate()	275
4.14.4.16	caerIMU9EventPacketFromPacketHeader()	275
4.14.4.17	caerIMU9EventPacketFromPacketHeaderConst()	276
4.14.4.18	caerIMU9EventPacketGetEvent()	276
4.14.4.19	caerIMU9EventPacketGetEventConst()	277
4.14.4.20	caerIMU9EventSetAccelX()	277
4.14.4.21	caerIMU9EventSetAccelY()	277
4.14.4.22	caerIMU9EventSetAccelZ()	278
4.14.4.23	caerIMU9EventSetCompX()	278
4.14.4.24	caerIMU9EventSetCompY()	278
4.14.4.25	caerIMU9EventSetCompZ()	278
4.14.4.26	caerIMU9EventSetGyroX()	279
4.14.4.27	caerIMU9EventSetGyroY()	279
4.14.4.28	caerIMU9EventSetGyroZ()	279
4.14.4.29	caerIMU9EventSetTemp()	280
4.14.4.30	caerIMU9EventSetTimestamp()	280
4.14.4.31	caerIMU9EventValidate()	280
4.14.4.32	PACKED_STRUCT() [1/2]	281
4.14.4.33	PACKED_STRUCT() [2/2]	281

4.15 events/matrix4x4.h File Reference	281
4.15.1 Detailed Description	283
4.15.2 Macro Definition Documentation	283
4.15.2.1 CAER_MATRIX4x4_CONST_ITERATOR_ALL_START	283
4.15.2.2 CAER_MATRIX4x4_CONST_ITERATOR_VALID_START	284
4.15.2.3 CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_ALL_START	284
4.15.2.4 CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_VALID_START	285
4.15.2.5 CAER_MATRIX4x4_ITERATOR_ALL_END	285
4.15.2.6 CAER_MATRIX4x4_ITERATOR_ALL_START	285
4.15.2.7 CAER_MATRIX4x4_ITERATOR_VALID_END	285
4.15.2.8 CAER_MATRIX4x4_ITERATOR_VALID_START	286
4.15.2.9 CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_END	286
4.15.2.10 CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_START	286
4.15.2.11 CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_END	287
4.15.2.12 CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_START	287
4.15.2.13 MATRIX4x4_SCALE_MASK	287
4.15.2.14 MATRIX4x4_SCALE_SHIFT	287
4.15.2.15 MATRIX4x4_TYPE_MASK	287
4.15.2.16 MATRIX4x4_TYPE_SHIFT	288
4.15.3 Typedef Documentation	288
4.15.3.1 caerMatrix4x4Event	288
4.15.3.2 caerMatrix4x4EventPacket	288
4.15.4 Function Documentation	288
4.15.4.1 caerMatrix4x4EventGetM00()	288
4.15.4.2 caerMatrix4x4EventGetM01()	289
4.15.4.3 caerMatrix4x4EventGetM02()	289
4.15.4.4 caerMatrix4x4EventGetM03()	289
4.15.4.5 caerMatrix4x4EventGetM10()	290
4.15.4.6 caerMatrix4x4EventGetM11()	290
4.15.4.7 caerMatrix4x4EventGetM12()	290

4.15.4.8 caerMatrix4x4EventGetM13()	291
4.15.4.9 caerMatrix4x4EventGetM20()	291
4.15.4.10 caerMatrix4x4EventGetM21()	291
4.15.4.11 caerMatrix4x4EventGetM22()	292
4.15.4.12 caerMatrix4x4EventGetM23()	292
4.15.4.13 caerMatrix4x4EventGetM30()	292
4.15.4.14 caerMatrix4x4EventGetM31()	294
4.15.4.15 caerMatrix4x4EventGetM32()	294
4.15.4.16 caerMatrix4x4EventGetM33()	294
4.15.4.17 caerMatrix4x4EventGetScale()	296
4.15.4.18 caerMatrix4x4EventGetTimestamp()	296
4.15.4.19 caerMatrix4x4EventGetTimestamp64()	297
4.15.4.20 caerMatrix4x4EventGetType()	297
4.15.4.21 caerMatrix4x4EventInvalidate()	297
4.15.4.22 caerMatrix4x4EventIsValid()	298
4.15.4.23 caerMatrix4x4EventPacketAllocate()	298
4.15.4.24 caerMatrix4x4EventPacketFromPacketHeader()	298
4.15.4.25 caerMatrix4x4EventPacketFromPacketHeaderConst()	299
4.15.4.26 caerMatrix4x4EventPacketGetEvent()	299
4.15.4.27 caerMatrix4x4EventPacketGetEventConst()	300
4.15.4.28 caerMatrix4x4EventSetM00()	300
4.15.4.29 caerMatrix4x4EventSetM01()	300
4.15.4.30 caerMatrix4x4EventSetM02()	301
4.15.4.31 caerMatrix4x4EventSetM03()	301
4.15.4.32 caerMatrix4x4EventSetM10()	301
4.15.4.33 caerMatrix4x4EventSetM11()	301
4.15.4.34 caerMatrix4x4EventSetM12()	302
4.15.4.35 caerMatrix4x4EventSetM13()	302
4.15.4.36 caerMatrix4x4EventSetM20()	302
4.15.4.37 caerMatrix4x4EventSetM21()	303

4.15.4.38 caerMatrix4x4EventSetM22()	303
4.15.4.39 caerMatrix4x4EventSetM23()	303
4.15.4.40 caerMatrix4x4EventSetM30()	304
4.15.4.41 caerMatrix4x4EventSetM31()	304
4.15.4.42 caerMatrix4x4EventSetM32()	304
4.15.4.43 caerMatrix4x4EventSetM33()	305
4.15.4.44 caerMatrix4x4EventSetScale()	305
4.15.4.45 caerMatrix4x4EventSetTimestamp()	305
4.15.4.46 caerMatrix4x4EventSetType()	306
4.15.4.47 caerMatrix4x4EventValidate()	306
4.15.4.48 PACKED_STRUCT() [1/2]	306
4.15.4.49 PACKED_STRUCT() [2/2]	306
4.16 events/packetContainer.h File Reference	307
4.16.1 Detailed Description	308
4.16.2 Macro Definition Documentation	308
4.16.2.1 CAER_EVENT_PACKET_CONTAINER_CONST_ITERATOR_START	308
4.16.2.2 CAER_EVENT_PACKET_CONTAINER_ITERATOR_END	309
4.16.2.3 CAER_EVENT_PACKET_CONTAINER_ITERATOR_START	309
4.16.3 Typedef Documentation	309
4.16.3.1 caerEventPacketContainer	309
4.16.4 Function Documentation	309
4.16.4.1 caerEventPacketContainerAllocate()	309
4.16.4.2 caerEventPacketContainerCopyAllEvents()	310
4.16.4.3 caerEventPacketContainerCopyValidEvents()	310
4.16.4.4 caerEventPacketContainerFindEventPacketByType()	311
4.16.4.5 caerEventPacketContainerFindEventPacketByTypeConst()	311
4.16.4.6 caerEventPacketContainerFree()	311
4.16.4.7 caerEventPacketContainerGetEventPacket()	312
4.16.4.8 caerEventPacketContainerGetEventPacketConst()	312
4.16.4.9 caerEventPacketContainerGetEventPacketsNumber()	312

4.16.4.10 caerEventPacketContainerGetEventsNumber()	313
4.16.4.11 caerEventPacketContainerGetEventsValidNumber()	313
4.16.4.12 caerEventPacketContainerGetHighestEventTimestamp()	314
4.16.4.13 caerEventPacketContainerGetLowestEventTimestamp()	314
4.16.4.14 caerEventPacketContainerSetEventPacket()	314
4.16.4.15 caerEventPacketContainerSetEventPacketsNumber()	315
4.16.4.16 caerEventPacketContainerUpdateStatistics()	315
4.16.4.17 PACKED_STRUCT()	315
4.17 events/point1d.h File Reference	315
4.17.1 Detailed Description	317
4.17.2 Macro Definition Documentation	317
4.17.2.1 CAER_POINT1D_CONST_ITERATOR_ALL_START	317
4.17.2.2 CAER_POINT1D_CONST_ITERATOR_VALID_START	317
4.17.2.3 CAER_POINT1D_CONST_REVERSE_ITERATOR_ALL_START	318
4.17.2.4 CAER_POINT1D_CONST_REVERSE_ITERATOR_VALID_START	318
4.17.2.5 CAER_POINT1D_ITERATOR_ALL_END	318
4.17.2.6 CAER_POINT1D_ITERATOR_ALL_START	319
4.17.2.7 CAER_POINT1D_ITERATOR_VALID_END	319
4.17.2.8 CAER_POINT1D_ITERATOR_VALID_START	319
4.17.2.9 CAER_POINT1D_REVERSE_ITERATOR_ALL_END	319
4.17.2.10 CAER_POINT1D_REVERSE_ITERATOR_ALL_START	320
4.17.2.11 CAER_POINT1D_REVERSE_ITERATOR_VALID_END	320
4.17.2.12 CAER_POINT1D_REVERSE_ITERATOR_VALID_START	320
4.17.2.13 POINT1D_SCALE_MASK	320
4.17.2.14 POINT1D_SCALE_SHIFT	321
4.17.2.15 POINT1D_TYPE_MASK	321
4.17.2.16 POINT1D_TYPE_SHIFT	321
4.17.3 Typedef Documentation	321
4.17.3.1 caerPoint1DEvent	321
4.17.3.2 caerPoint1DEventPacket	321

4.17.4	Function Documentation	321
4.17.4.1	caerPoint1DEventGetScale()	321
4.17.4.2	caerPoint1DEventGetTimestamp()	322
4.17.4.3	caerPoint1DEventGetTimestamp64()	322
4.17.4.4	caerPoint1DEventGetType()	323
4.17.4.5	caerPoint1DEventGetX()	323
4.17.4.6	caerPoint1DEventInvalidate()	323
4.17.4.7	caerPoint1DEventIsValid()	324
4.17.4.8	caerPoint1DEventPacketAllocate()	324
4.17.4.9	caerPoint1DEventPacketFromPacketHeader()	324
4.17.4.10	caerPoint1DEventPacketFromPacketHeaderConst()	325
4.17.4.11	caerPoint1DEventPacketGetEvent()	325
4.17.4.12	caerPoint1DEventPacketGetEventConst()	326
4.17.4.13	caerPoint1DEventSetScale()	326
4.17.4.14	caerPoint1DEventSetTimestamp()	326
4.17.4.15	caerPoint1DEventSetType()	327
4.17.4.16	caerPoint1DEventSetX()	327
4.17.4.17	caerPoint1DEventValidate()	327
4.17.4.18	PACKED_STRUCT() [1/2]	328
4.17.4.19	PACKED_STRUCT() [2/2]	328
4.18	events/point2d.h File Reference	328
4.18.1	Detailed Description	329
4.18.2	Macro Definition Documentation	329
4.18.2.1	CAER_POINT2D_CONST_ITERATOR_ALL_START	330
4.18.2.2	CAER_POINT2D_CONST_ITERATOR_VALID_START	330
4.18.2.3	CAER_POINT2D_CONST_REVERSE_ITERATOR_ALL_START	330
4.18.2.4	CAER_POINT2D_CONST_REVERSE_ITERATOR_VALID_START	331
4.18.2.5	CAER_POINT2D_ITERATOR_ALL_END	331
4.18.2.6	CAER_POINT2D_ITERATOR_ALL_START	331
4.18.2.7	CAER_POINT2D_ITERATOR_VALID_END	331

4.18.2.8	CAER_POINT2D_ITERATOR_VALID_START	332
4.18.2.9	CAER_POINT2D_REVERSE_ITERATOR_ALL_END	332
4.18.2.10	CAER_POINT2D_REVERSE_ITERATOR_ALL_START	332
4.18.2.11	CAER_POINT2D_REVERSE_ITERATOR_VALID_END	332
4.18.2.12	CAER_POINT2D_REVERSE_ITERATOR_VALID_START	333
4.18.2.13	POINT2D_SCALE_MASK	333
4.18.2.14	POINT2D_SCALE_SHIFT	333
4.18.2.15	POINT2D_TYPE_MASK	333
4.18.2.16	POINT2D_TYPE_SHIFT	333
4.18.3	Typedef Documentation	334
4.18.3.1	caerPoint2DEvent	334
4.18.3.2	caerPoint2DEventPacket	334
4.18.4	Function Documentation	334
4.18.4.1	caerPoint2DEventGetScale()	334
4.18.4.2	caerPoint2DEventGetTimestamp()	334
4.18.4.3	caerPoint2DEventGetTimestamp64()	335
4.18.4.4	caerPoint2DEventGetType()	335
4.18.4.5	caerPoint2DEventGetX()	336
4.18.4.6	caerPoint2DEventGetY()	336
4.18.4.7	caerPoint2DEventInvalidate()	336
4.18.4.8	caerPoint2DEventIsValid()	337
4.18.4.9	caerPoint2DEventPacketAllocate()	337
4.18.4.10	caerPoint2DEventPacketFromPacketHeader()	337
4.18.4.11	caerPoint2DEventPacketFromPacketHeaderConst()	338
4.18.4.12	caerPoint2DEventPacketGetEvent()	338
4.18.4.13	caerPoint2DEventPacketGetEventConst()	339
4.18.4.14	caerPoint2DEventSetScale()	339
4.18.4.15	caerPoint2DEventSetTimestamp()	339
4.18.4.16	caerPoint2DEventSetType()	340
4.18.4.17	caerPoint2DEventSetX()	340

4.18.4.18 caerPoint2DEventSetY()	340
4.18.4.19 caerPoint2DEventValidate()	341
4.18.4.20 PACKED_STRUCT() [1/2]	341
4.18.4.21 PACKED_STRUCT() [2/2]	341
4.19 events/point3d.h File Reference	341
4.19.1 Detailed Description	343
4.19.2 Macro Definition Documentation	343
4.19.2.1 CAER_POINT3D_CONST_ITERATOR_ALL_START	343
4.19.2.2 CAER_POINT3D_CONST_ITERATOR_VALID_START	344
4.19.2.3 CAER_POINT3D_CONST_REVERSE_ITERATOR_ALL_START	344
4.19.2.4 CAER_POINT3D_CONST_REVERSE_ITERATOR_VALID_START	344
4.19.2.5 CAER_POINT3D_ITERATOR_ALL_END	345
4.19.2.6 CAER_POINT3D_ITERATOR_ALL_START	345
4.19.2.7 CAER_POINT3D_ITERATOR_VALID_END	345
4.19.2.8 CAER_POINT3D_ITERATOR_VALID_START	345
4.19.2.9 CAER_POINT3D_REVERSE_ITERATOR_ALL_END	346
4.19.2.10 CAER_POINT3D_REVERSE_ITERATOR_ALL_START	346
4.19.2.11 CAER_POINT3D_REVERSE_ITERATOR_VALID_END	346
4.19.2.12 CAER_POINT3D_REVERSE_ITERATOR_VALID_START	346
4.19.2.13 POINT3D_SCALE_MASK	347
4.19.2.14 POINT3D_SCALE_SHIFT	347
4.19.2.15 POINT3D_TYPE_MASK	347
4.19.2.16 POINT3D_TYPE_SHIFT	347
4.19.3 Typedef Documentation	347
4.19.3.1 caerPoint3DEvent	347
4.19.3.2 caerPoint3DEventPacket	347
4.19.4 Function Documentation	348
4.19.4.1 caerPoint3DEventGetScale()	348
4.19.4.2 caerPoint3DEventGetTimestamp()	348
4.19.4.3 caerPoint3DEventGetTimestamp64()	348

4.19.4.4	caerPoint3DEventGetType()	349
4.19.4.5	caerPoint3DEventGetX()	349
4.19.4.6	caerPoint3DEventGetY()	350
4.19.4.7	caerPoint3DEventGetZ()	350
4.19.4.8	caerPoint3DEventInvalidate()	350
4.19.4.9	caerPoint3DEventIsValid()	351
4.19.4.10	caerPoint3DEventPacketAllocate()	351
4.19.4.11	caerPoint3DEventPacketFromPacketHeader()	351
4.19.4.12	caerPoint3DEventPacketFromPacketHeaderConst()	352
4.19.4.13	caerPoint3DEventPacketGetEvent()	352
4.19.4.14	caerPoint3DEventPacketGetEventConst()	353
4.19.4.15	caerPoint3DEventSetScale()	353
4.19.4.16	caerPoint3DEventSetTimestamp()	353
4.19.4.17	caerPoint3DEventSetType()	354
4.19.4.18	caerPoint3DEventSetX()	354
4.19.4.19	caerPoint3DEventSetY()	354
4.19.4.20	caerPoint3DEventSetZ()	355
4.19.4.21	caerPoint3DEventValidate()	355
4.19.4.22	PACKED_STRUCT() [1/2]	355
4.19.4.23	PACKED_STRUCT() [2/2]	355
4.20	events/point4d.h File Reference	356
4.20.1	Detailed Description	357
4.20.2	Macro Definition Documentation	357
4.20.2.1	CAER_POINT4D_CONST_ITERATOR_ALL_START	357
4.20.2.2	CAER_POINT4D_CONST_ITERATOR_VALID_START	358
4.20.2.3	CAER_POINT4D_CONST_REVERSE_ITERATOR_ALL_START	358
4.20.2.4	CAER_POINT4D_CONST_REVERSE_ITERATOR_VALID_START	358
4.20.2.5	CAER_POINT4D_ITERATOR_ALL_END	359
4.20.2.6	CAER_POINT4D_ITERATOR_ALL_START	359
4.20.2.7	CAER_POINT4D_ITERATOR_VALID_END	359

4.20.2.8	CAER_POINT4D_ITERATOR_VALID_START	359
4.20.2.9	CAER_POINT4D_REVERSE_ITERATOR_ALL_END	360
4.20.2.10	CAER_POINT4D_REVERSE_ITERATOR_ALL_START	360
4.20.2.11	CAER_POINT4D_REVERSE_ITERATOR_VALID_END	360
4.20.2.12	CAER_POINT4D_REVERSE_ITERATOR_VALID_START	360
4.20.2.13	POINT4D_SCALE_MASK	361
4.20.2.14	POINT4D_SCALE_SHIFT	361
4.20.2.15	POINT4D_TYPE_MASK	361
4.20.2.16	POINT4D_TYPE_SHIFT	361
4.20.3	Typedef Documentation	361
4.20.3.1	caerPoint4DEvent	361
4.20.3.2	caerPoint4DEventPacket	361
4.20.4	Function Documentation	362
4.20.4.1	caerPoint4DEventGetScale()	362
4.20.4.2	caerPoint4DEventGetTimestamp()	362
4.20.4.3	caerPoint4DEventGetTimestamp64()	362
4.20.4.4	caerPoint4DEventGetType()	363
4.20.4.5	caerPoint4DEventGetW()	363
4.20.4.6	caerPoint4DEventGetX()	364
4.20.4.7	caerPoint4DEventGetY()	364
4.20.4.8	caerPoint4DEventGetZ()	364
4.20.4.9	caerPoint4DEventInvalidate()	365
4.20.4.10	caerPoint4DEventIsValid()	365
4.20.4.11	caerPoint4DEventPacketAllocate()	365
4.20.4.12	caerPoint4DEventPacketFromPacketHeader()	366
4.20.4.13	caerPoint4DEventPacketFromPacketHeaderConst()	366
4.20.4.14	caerPoint4DEventPacketGetEvent()	366
4.20.4.15	caerPoint4DEventPacketGetEventConst()	367
4.20.4.16	caerPoint4DEventSetScale()	367
4.20.4.17	caerPoint4DEventSetTimestamp()	368

4.20.4.18 caerPoint4DEventSetType()	368
4.20.4.19 caerPoint4DEventSetW()	368
4.20.4.20 caerPoint4DEventSetX()	368
4.20.4.21 caerPoint4DEventSetY()	369
4.20.4.22 caerPoint4DEventSetZ()	369
4.20.4.23 caerPoint4DEventValidate()	369
4.20.4.24 PACKED_STRUCT() [1/2]	371
4.20.4.25 PACKED_STRUCT() [2/2]	371
4.21 events/polarity.h File Reference	371
4.21.1 Detailed Description	372
4.21.2 Macro Definition Documentation	372
4.21.2.1 CAER_POLARITY_CONST_ITERATOR_ALL_START	373
4.21.2.2 CAER_POLARITY_CONST_ITERATOR_VALID_START	373
4.21.2.3 CAER_POLARITY_CONST_REVERSE_ITERATOR_ALL_START	373
4.21.2.4 CAER_POLARITY_CONST_REVERSE_ITERATOR_VALID_START	374
4.21.2.5 CAER_POLARITY_ITERATOR_ALL_END	374
4.21.2.6 CAER_POLARITY_ITERATOR_ALL_START	374
4.21.2.7 CAER_POLARITY_ITERATOR_VALID_END	374
4.21.2.8 CAER_POLARITY_ITERATOR_VALID_START	375
4.21.2.9 CAER_POLARITY_REVERSE_ITERATOR_ALL_END	375
4.21.2.10 CAER_POLARITY_REVERSE_ITERATOR_ALL_START	375
4.21.2.11 CAER_POLARITY_REVERSE_ITERATOR_VALID_END	375
4.21.2.12 CAER_POLARITY_REVERSE_ITERATOR_VALID_START	376
4.21.2.13 POLARITY_MASK	376
4.21.2.14 POLARITY_SHIFT	376
4.21.2.15 POLARITY_X_ADDR_MASK	376
4.21.2.16 POLARITY_X_ADDR_SHIFT	376
4.21.2.17 POLARITY_Y_ADDR_MASK	377
4.21.2.18 POLARITY_Y_ADDR_SHIFT	377
4.21.3 Typedef Documentation	377

4.21.3.1	caerPolarityEvent	377
4.21.3.2	caerPolarityEventPacket	377
4.21.4	Function Documentation	377
4.21.4.1	caerPolarityEventGetPolarity()	377
4.21.4.2	caerPolarityEventGetTimestamp()	378
4.21.4.3	caerPolarityEventGetTimestamp64()	378
4.21.4.4	caerPolarityEventGetX()	378
4.21.4.5	caerPolarityEventGetY()	379
4.21.4.6	caerPolarityEventInvalidate()	379
4.21.4.7	caerPolarityEventIsValid()	380
4.21.4.8	caerPolarityEventPacketAllocate()	380
4.21.4.9	caerPolarityEventPacketFromPacketHeader()	380
4.21.4.10	caerPolarityEventPacketFromPacketHeaderConst()	381
4.21.4.11	caerPolarityEventPacketGetEvent()	381
4.21.4.12	caerPolarityEventPacketGetEventConst()	381
4.21.4.13	caerPolarityEventSetPolarity()	382
4.21.4.14	caerPolarityEventSetTimestamp()	382
4.21.4.15	caerPolarityEventSetX()	382
4.21.4.16	caerPolarityEventSetY()	383
4.21.4.17	caerPolarityEventValidate()	383
4.21.4.18	PACKED_STRUCT() [1/2]	383
4.21.4.19	PACKED_STRUCT() [2/2]	384
4.22	events/sample.h File Reference	384
4.22.1	Detailed Description	385
4.22.2	Macro Definition Documentation	385
4.22.2.1	CAER_SAMPLE_CONST_ITERATOR_ALL_START	385
4.22.2.2	CAER_SAMPLE_CONST_ITERATOR_VALID_START	386
4.22.2.3	CAER_SAMPLE_CONST_REVERSE_ITERATOR_ALL_START	386
4.22.2.4	CAER_SAMPLE_CONST_REVERSE_ITERATOR_VALID_START	386
4.22.2.5	CAER_SAMPLE_ITERATOR_ALL_END	387

4.22.2.6	CAER_SAMPLE_ITERATOR_ALL_START	387
4.22.2.7	CAER_SAMPLE_ITERATOR_VALID_END	387
4.22.2.8	CAER_SAMPLE_ITERATOR_VALID_START	387
4.22.2.9	CAER_SAMPLE_REVERSE_ITERATOR_ALL_END	388
4.22.2.10	CAER_SAMPLE_REVERSE_ITERATOR_ALL_START	388
4.22.2.11	CAER_SAMPLE_REVERSE_ITERATOR_VALID_END	388
4.22.2.12	CAER_SAMPLE_REVERSE_ITERATOR_VALID_START	388
4.22.2.13	SAMPLE_MASK	389
4.22.2.14	SAMPLE_SHIFT	389
4.22.2.15	SAMPLE_TYPE_MASK	389
4.22.2.16	SAMPLE_TYPE_SHIFT	389
4.22.3	Typedef Documentation	389
4.22.3.1	caerSampleEvent	389
4.22.3.2	caerSampleEventPacket	389
4.22.4	Function Documentation	390
4.22.4.1	caerSampleEventGetSample()	390
4.22.4.2	caerSampleEventGetTimestamp()	390
4.22.4.3	caerSampleEventGetTimestamp64()	390
4.22.4.4	caerSampleEventGetType()	391
4.22.4.5	caerSampleEventInvalidate()	391
4.22.4.6	caerSampleEventIsValid()	391
4.22.4.7	caerSampleEventPacketAllocate()	392
4.22.4.8	caerSampleEventPacketFromPacketHeader()	392
4.22.4.9	caerSampleEventPacketFromPacketHeaderConst()	393
4.22.4.10	caerSampleEventPacketGetEvent()	393
4.22.4.11	caerSampleEventPacketGetEventConst()	393
4.22.4.12	caerSampleEventSetSample()	394
4.22.4.13	caerSampleEventSetTimestamp()	394
4.22.4.14	caerSampleEventSetType()	394
4.22.4.15	caerSampleEventValidate()	395

4.22.4.16	PACKED_STRUCT() [1/2]	395
4.22.4.17	PACKED_STRUCT() [2/2]	395
4.23	events/special.h File Reference	395
4.23.1	Detailed Description	397
4.23.2	Macro Definition Documentation	397
4.23.2.1	CAER_SPECIAL_CONST_ITERATOR_ALL_START	397
4.23.2.2	CAER_SPECIAL_CONST_ITERATOR_VALID_START	398
4.23.2.3	CAER_SPECIAL_CONST_REVERSE_ITERATOR_ALL_START	398
4.23.2.4	CAER_SPECIAL_CONST_REVERSE_ITERATOR_VALID_START	399
4.23.2.5	CAER_SPECIAL_ITERATOR_ALL_END	399
4.23.2.6	CAER_SPECIAL_ITERATOR_ALL_START	399
4.23.2.7	CAER_SPECIAL_ITERATOR_VALID_END	399
4.23.2.8	CAER_SPECIAL_ITERATOR_VALID_START	400
4.23.2.9	CAER_SPECIAL_REVERSE_ITERATOR_ALL_END	400
4.23.2.10	CAER_SPECIAL_REVERSE_ITERATOR_ALL_START	400
4.23.2.11	CAER_SPECIAL_REVERSE_ITERATOR_VALID_END	400
4.23.2.12	CAER_SPECIAL_REVERSE_ITERATOR_VALID_START	401
4.23.2.13	SPECIAL_DATA_MASK	401
4.23.2.14	SPECIAL_DATA_SHIFT	401
4.23.2.15	SPECIAL_TYPE_MASK	401
4.23.2.16	SPECIAL_TYPE_SHIFT	401
4.23.3	Typedef Documentation	402
4.23.3.1	caerSpecialEvent	402
4.23.3.2	caerSpecialEventPacket	402
4.23.4	Enumeration Type Documentation	402
4.23.4.1	caer_special_event_types	402
4.23.5	Function Documentation	403
4.23.5.1	caerSpecialEventGetData()	403
4.23.5.2	caerSpecialEventGetTimestamp()	403
4.23.5.3	caerSpecialEventGetTimestamp64()	404

4.23.5.4	caerSpecialEventGetType()	404
4.23.5.5	caerSpecialEventInvalidate()	405
4.23.5.6	caerSpecialEventIsValid()	405
4.23.5.7	caerSpecialEventPacketAllocate()	405
4.23.5.8	caerSpecialEventPacketFindEventByType()	406
4.23.5.9	caerSpecialEventPacketFindEventByTypeConst()	406
4.23.5.10	caerSpecialEventPacketFindValidEventByType()	406
4.23.5.11	caerSpecialEventPacketFindValidEventByTypeConst()	407
4.23.5.12	caerSpecialEventPacketFromPacketHeader()	407
4.23.5.13	caerSpecialEventPacketFromPacketHeaderConst()	408
4.23.5.14	caerSpecialEventPacketGetEvent()	408
4.23.5.15	caerSpecialEventPacketGetEventConst()	408
4.23.5.16	caerSpecialEventSetData()	409
4.23.5.17	caerSpecialEventSetTimestamp()	409
4.23.5.18	caerSpecialEventSetType()	409
4.23.5.19	caerSpecialEventValidate()	410
4.23.5.20	PACKED_STRUCT() [1/2]	410
4.23.5.21	PACKED_STRUCT() [2/2]	410
4.24	events/spike.h File Reference	411
4.24.1	Detailed Description	412
4.24.2	Macro Definition Documentation	412
4.24.2.1	CAER_SPIKE_CONST_ITERATOR_ALL_START	412
4.24.2.2	CAER_SPIKE_CONST_ITERATOR_VALID_START	413
4.24.2.3	CAER_SPIKE_CONST_REVERSE_ITERATOR_ALL_START	413
4.24.2.4	CAER_SPIKE_CONST_REVERSE_ITERATOR_VALID_START	413
4.24.2.5	CAER_SPIKE_ITERATOR_ALL_END	414
4.24.2.6	CAER_SPIKE_ITERATOR_ALL_START	414
4.24.2.7	CAER_SPIKE_ITERATOR_VALID_END	414
4.24.2.8	CAER_SPIKE_ITERATOR_VALID_START	414
4.24.2.9	CAER_SPIKE_REVERSE_ITERATOR_ALL_END	415

4.24.2.10	CAER_SPIKE_REVERSE_ITERATOR_ALL_START	415
4.24.2.11	CAER_SPIKE_REVERSE_ITERATOR_VALID_END	415
4.24.2.12	CAER_SPIKE_REVERSE_ITERATOR_VALID_START	415
4.24.2.13	SPIKE_CHIP_ID_MASK	416
4.24.2.14	SPIKE_CHIP_ID_SHIFT	416
4.24.2.15	SPIKE_NEURON_ID_MASK	416
4.24.2.16	SPIKE_NEURON_ID_SHIFT	416
4.24.2.17	SPIKE_SOURCE_CORE_ID_MASK	416
4.24.2.18	SPIKE_SOURCE_CORE_ID_SHIFT	416
4.24.3	Typedef Documentation	416
4.24.3.1	caerSpikeEvent	417
4.24.3.2	caerSpikeEventPacket	417
4.24.4	Function Documentation	417
4.24.4.1	caerSpikeEventGetChipID()	417
4.24.4.2	caerSpikeEventGetNeuronID()	417
4.24.4.3	caerSpikeEventGetSourceCoreID()	418
4.24.4.4	caerSpikeEventGetTimestamp()	418
4.24.4.5	caerSpikeEventGetTimestamp64()	418
4.24.4.6	caerSpikeEventInvalidate()	419
4.24.4.7	caerSpikeEventIsValid()	419
4.24.4.8	caerSpikeEventPacketAllocate()	419
4.24.4.9	caerSpikeEventPacketFromPacketHeader()	420
4.24.4.10	caerSpikeEventPacketFromPacketHeaderConst()	420
4.24.4.11	caerSpikeEventPacketGetEvent()	421
4.24.4.12	caerSpikeEventPacketGetEventConst()	421
4.24.4.13	caerSpikeEventSetChipID()	421
4.24.4.14	caerSpikeEventSetNeuronID()	423
4.24.4.15	caerSpikeEventSetSourceCoreID()	423
4.24.4.16	caerSpikeEventSetTimestamp()	423
4.24.4.17	caerSpikeEventValidate()	424

4.24.4.18	PACKED_STRUCT() [1/2]	424
4.24.4.19	PACKED_STRUCT() [2/2]	424
4.25	filters/dvs_noise.h File Reference	424
4.25.1	Detailed Description	425
4.25.2	Macro Definition Documentation	425
4.25.2.1	CAER_FILTER_DVS_BACKGROUND_ACTIVITY_CHECK_POLARITY	426
4.25.2.2	CAER_FILTER_DVS_BACKGROUND_ACTIVITY_ENABLE	426
4.25.2.3	CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS	426
4.25.2.4	CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MAX	426
4.25.2.5	CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MIN	426
4.25.2.6	CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TIME	426
4.25.2.7	CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TWO_LEVELS	426
4.25.2.8	CAER_FILTER_DVS_HOTPIXEL_COUNT	427
4.25.2.9	CAER_FILTER_DVS_HOTPIXEL_ENABLE	427
4.25.2.10	CAER_FILTER_DVS_HOTPIXEL_LEARN	427
4.25.2.11	CAER_FILTER_DVS_HOTPIXEL_STATISTICS	427
4.25.2.12	CAER_FILTER_DVS_HOTPIXEL_TIME	427
4.25.2.13	CAER_FILTER_DVS_LOG_LEVEL	427
4.25.2.14	CAER_FILTER_DVS_REFRACTORY_PERIOD_ENABLE	427
4.25.2.15	CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS	428
4.25.2.16	CAER_FILTER_DVS_REFRACTORY_PERIOD_TIME	428
4.25.2.17	CAER_FILTER_DVS_RESET	428
4.25.3	Typedef Documentation	428
4.25.3.1	caerFilterDVSNoise	428
4.25.3.2	caerFilterDVSPixel	428
4.25.4	Function Documentation	428
4.25.4.1	caerFilterDVSNoiseApply()	428
4.25.4.2	caerFilterDVSNoiseConfigGet()	429
4.25.4.3	caerFilterDVSNoiseConfigSet()	429
4.25.4.4	caerFilterDVSNoiseDestroy()	430

4.25.4.5	caerFilterDVSNoiseGetHotPixels()	430
4.25.4.6	caerFilterDVSNoiseInitialize()	430
4.26	frame_utils.h File Reference	431
4.26.1	Detailed Description	431
4.27	libcaer.h File Reference	431
4.27.1	Detailed Description	433
4.27.2	Macro Definition Documentation	433
4.27.2.1	CLEAR_NUMBITS16	433
4.27.2.2	CLEAR_NUMBITS32	433
4.27.2.3	CLEAR_NUMBITS8	433
4.27.2.4	GET_NUMBITS16	434
4.27.2.5	GET_NUMBITS32	434
4.27.2.6	GET_NUMBITS8	434
4.27.2.7	I16T	434
4.27.2.8	I32T	434
4.27.2.9	I64T	434
4.27.2.10	I8T	435
4.27.2.11	LIBCAER_HAVE_OPENCV	435
4.27.2.12	LIBCAER_HAVE_SERIALDEV	435
4.27.2.13	LIBCAER_NAME_STRING	435
4.27.2.14	LIBCAER_VERSION	435
4.27.2.15	LIBCAER_VERSION_STRING	435
4.27.2.16	MASK_NUMBITS32	435
4.27.2.17	MASK_NUMBITS64	436
4.27.2.18	SET_NUMBITS16	436
4.27.2.19	SET_NUMBITS32	436
4.27.2.20	SET_NUMBITS8	436
4.27.2.21	SWAP_VAR	436
4.27.2.22	U16T	436
4.27.2.23	U32T	437

4.27.2.24	U64T	437
4.27.2.25	U8T	437
4.27.3	Enumeration Type Documentation	437
4.27.3.1	caer_error_codes	437
4.27.4	Function Documentation	437
4.27.4.1	caerByteArrayToInteger()	437
4.27.4.2	caerIntegerToByteArray()	438
4.27.4.3	caerStrEquals()	438
4.27.4.4	caerStrEqualsUpTo()	439
4.28	log.h File Reference	439
4.28.1	Detailed Description	440
4.28.2	Enumeration Type Documentation	440
4.28.2.1	caer_log_level	440
4.28.3	Function Documentation	440
4.28.3.1	caerLog()	440
4.28.3.2	caerLogDisable()	440
4.28.3.3	caerLogDisabled()	441
4.28.3.4	caerLogFileDescriptorsGetFirst()	441
4.28.3.5	caerLogFileDescriptorsGetSecond()	441
4.28.3.6	caerLogFileDescriptorsSet()	441
4.28.3.7	caerLogLevelGet()	442
4.28.3.8	caerLogLevelSet()	442
4.28.3.9	caerLogVA()	442
4.28.3.10	caerLogVAFull()	443
4.29	network.h File Reference	443
4.29.1	Detailed Description	444
4.30	portable_endian.h File Reference	444
4.30.1	Detailed Description	444

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

caer_bias_coarsefine	5
caer_bias_dynapse	5
caer_bias_shiftedsources	6
caer_bias_vdac	7
caer_davis_info	7
caer_device_discovery_result	9
caer_dvs128_info	9
caer_dynapse_info	10
caer_edvs_info	11
caer_filter_dvs_pixel	12

Chapter 2

File Index

2.1 File List

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

frame_utils.h	431
libcaer.h	431
log.h	439
network.h	443
portable_endian.h	444
ringbuffer.h	??
devices/ davis.h	13
devices/ device.h	130
devices/ device_discover.h	136
devices/ dvs128.h	138
devices/ dynapse.h	142
devices/ edvs.h	167
devices/ serial.h	171
devices/ usb.h	173
events/ common.h	174
events/ config.h	192
events/ ear.h	205
events/ frame.h	218
events/ imu6.h	248
events/ imu9.h	263
events/ matrix4x4.h	281
events/ packetContainer.h	307
events/ point1d.h	315
events/ point2d.h	328
events/ point3d.h	341
events/ point4d.h	356
events/ polarity.h	371
events/ sample.h	384
events/ special.h	395
events/ spike.h	411
filters/ dvs_noise.h	424

Chapter 3

Data Structure Documentation

3.1 caer_bias_coarsefine Struct Reference

```
#include <davis.h>
```

Data Fields

- uint8_t [coarseValue](#)
Coarse current, from 0 to 7, creates big variations in output current.
- uint8_t [fineValue](#)
Fine current, from 0 to 255, creates small variations in output current.
- bool [enabled](#)
Whether this bias is enabled or not.
- bool [sexN](#)
Bias sex: true for 'N' type, false for 'P' type.
- bool [typeNormal](#)
Bias type: true for 'Normal', false for 'Cascode'.
- bool [currentLevelNormal](#)
Bias current level: true for 'Normal', false for 'Low'.

3.1.1 Detailed Description

On-chip coarse-fine bias current configuration. See '<https://inivation.com/support/hardware/biasing/>' for more details.

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

- devices/[davis.h](#)

3.2 caer_bias_dynapse Struct Reference

```
#include <dynapse.h>
```

Data Fields

- uint8_t [biasAddress](#)
Address of bias to configure, see DYNAPSE_CONFIG_BIAS_ defines.*
- uint8_t [coarseValue](#)
Coarse current, from 0 to 7, creates big variations in output current.
- uint8_t [fineValue](#)
Fine current, from 0 to 255, creates small variations in output current.
- bool [enabled](#)
Whether this bias is enabled or not.
- bool [sexN](#)
Bias sex: true for 'N' type, false for 'P' type.
- bool [typeNormal](#)
Bias type: true for 'Normal', false for 'Cascode'.
- bool [biasHigh](#)
Bias current level: true for 'HighBias', false for 'LowBias'.

3.2.1 Detailed Description

On-chip coarse-fine bias current configuration for Dynap-se. See '<https://ai-ctx.com/support/>' section 'Neuron's behaviors and parameters tuning'.

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

- [devices/dynapse.h](#)

3.3 caer_bias_shiftedsources Struct Reference

```
#include <davis.h>
```

Data Fields

- uint8_t [refValue](#)
Shifted-source bias level, from 0 to 63.
- uint8_t [regValue](#)
Shifted-source bias current for buffer amplifier, from 0 to 63.
- enum [caer_bias_shiftedsources_operating_mode](#) [operatingMode](#)
Shifted-source operating mode (see 'enum caer_bias_shiftedsources_operating_mode').
- enum [caer_bias_shiftedsources_voltage_level](#) [voltageLevel](#)
Shifted-source voltage level (see 'enum caer_bias_shiftedsources_voltage_level').

3.3.1 Detailed Description

On-chip shifted-source bias current configuration. See '<https://inivation.com/support/hardware/biasing/>' for more details.

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

- [devices/davis.h](#)

3.4 caer_bias_vdac Struct Reference

```
#include <davis.h>
```

Data Fields

- uint8_t [voltageValue](#)
Voltage, between 0 and 63, as a fraction of 1/64th of VDD=3.3V.
- uint8_t [currentValue](#)
Current, between 0 and 7, that drives the voltage.

3.4.1 Detailed Description

On-chip voltage digital-to-analog converter configuration. See <https://inivation.com/support/hardware/biasing/> for more details.

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

- devices/[davis.h](#)

3.5 caer_davis_info Struct Reference

```
#include <davis.h>
```

Data Fields

- int16_t [deviceId](#)
Unique device identifier. Also 'source' for events.
- char [deviceSerialNumber](#) [8+1]
Device serial number.
- uint8_t [deviceUSBBusNumber](#)
Device USB bus number.
- uint8_t [deviceUSBDeviceAddress](#)
Device USB device address.
- char * [deviceString](#)
- int16_t [logicVersion](#)
Logic (FPGA/CPLD) version.
- bool [deviceIsMaster](#)
Whether the device is a time-stamp master or slave.
- int16_t [logicClock](#)
Clock in MHz for main logic (FPGA/CPLD).
- int16_t [adcClock](#)
Clock in MHz for ADC/APS logic (FPGA/CPLD).
- int16_t [chipID](#)
Chip identifier/type.
- int16_t [dvsSizeX](#)

- DVS X axis resolution.*
- `int16_t` [dvsSizeY](#)
 - DVS Y axis resolution.*
- `bool` [dvsHasPixelFilter](#)
 - Feature test: DVS pixel-level filtering.*
- `bool` [dvsHasBackgroundActivityFilter](#)
 - Feature test: DVS Background Activity filter (and Refractory Period filter).*
- `bool` [dvsHasTestEventGenerator](#)
 - Feature test: fake event generator (testing/debug).*
- `int16_t` [apsSizeX](#)
 - APS X axis resolution.*
- `int16_t` [apsSizeY](#)
 - APS Y axis resolution.*
- `enum` [caer_frame_event_color_filter](#) [apsColorFilter](#)
 - APS color filter type.*
- `bool` [apsHasGlobalShutter](#)
 - Feature test: APS supports Global Shutter.*
- `bool` [apsHasQuadROI](#)
 - Feature test: APS supports Quadruple Region-of-Interest readout.*
- `bool` [apsHasExternalADC](#)
 - Feature test: APS supports External ADC for getting the image.*
- `bool` [apsHasInternalADC](#)
 - Feature test: APS supports Internal (on-chip) ADC for getting the image.*
- `bool` [extInputHasGenerator](#)
 - Feature test: External Input module supports Signal-Generation.*
- `bool` [extInputHasExtraDetectors](#)
 - Feature test: External Input module supports extra detectors (1 & 2).*
- `bool` [dvsHasROIFilter](#)
 - Feature test: DVS ROI filter.*
- `bool` [dvsHasStatistics](#)
 - Feature test: DVS statistics support.*
- `bool` [muxHasStatistics](#)
 - Feature test: Multiplexer statistics support (event drops).*

3.5.1 Detailed Description

DAVIS device-related information.

3.5.2 Field Documentation

3.5.2.1 deviceString

```
char* caer_davis_info::deviceString
```

Device information string, for logging purposes. If not NULL, pointed-to memory is *only* valid while the corresponding device is open! After calling `deviceClose()` this is invalid memory!

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

- `devices/davis.h`

3.6 caer_device_discovery_result Struct Reference

```
#include <device_discover.h>
```

Data Fields

- uint16_t **deviceType**
- bool **deviceErrorOpen**
- bool **deviceErrorVersion**
-
- union {
 struct caer_dvs128_info **dvs128Info**
 struct caer_edvs_info **edvsInfo**
 struct caer_davis_info **davisInfo**
 struct caer_dynapse_info **dynapseInfo**
 } **deviceInfo**

3.6.1 Detailed Description

Result of a device discovery operation. Contains the type of the device and its informational structure; use the device type to properly select the right info structure! In the info structures, 'deviceId' will always be set to -1 and 'deviceString' will always be NULL, as those are not present during the generic discovery phase.

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

- devices/[device_discover.h](#)

3.7 caer_dvs128_info Struct Reference

```
#include <dvs128.h>
```

Data Fields

- int16_t **deviceId**
Unique device identifier. Also 'source' for events.
- char **deviceSerialNumber** [8+1]
Device serial number.
- uint8_t **deviceUSBBusNumber**
Device USB bus number.
- uint8_t **deviceUSBDeviceAddress**
Device USB device address.
- char * **deviceString**
- int16_t **logicVersion**
Logic (FPGA/CPLD) version.
- bool **deviceIsMaster**
Whether the device is a time-stamp master or slave.
- int16_t **dvsSizeX**
DVS X axis resolution.
- int16_t **dvsSizeY**
DVS Y axis resolution.

3.7.1 Detailed Description

DVS128 device-related information.

3.7.2 Field Documentation

3.7.2.1 deviceString

```
char* caer_dvs128_info::deviceString
```

Device information string, for logging purposes. If not NULL, pointed-to memory is *only* valid while the corresponding device is open! After calling deviceClose() this is invalid memory!

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

- [devices/dvs128.h](#)

3.8 caer_dynapse_info Struct Reference

```
#include <dynapse.h>
```

Data Fields

- `int16_t deviceID`
Unique device identifier. Also 'source' for events.
- `char deviceSerialNumber [8+1]`
Device serial number.
- `uint8_t deviceUSBBusNumber`
Device USB bus number.
- `uint8_t deviceUSBDeviceAddress`
Device USB device address.
- `char * deviceString`
- `int16_t logicVersion`
Logic (FPGA/CPLD) version.
- `bool devicesMaster`
Whether the device is a time-stamp master or slave.
- `int16_t logicClock`
Clock in MHz for main logic (FPGA/CPLD).
- `int16_t chipID`
Chip identifier/type.
- `bool aerHasStatistics`
Feature test: AER (spikes) statistics support.
- `bool muxHasStatistics`
Feature test: Multiplexer statistics support (event drops).

3.8.1 Detailed Description

Dynap-se device-related information.

3.8.2 Field Documentation

3.8.2.1 deviceString

```
char* caer_dynapse_info::deviceString
```

Device information string, for logging purposes. If not NULL, pointed-to memory is *only* valid while the corresponding device is open! After calling deviceClose() this is invalid memory!

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

- [devices/dynapse.h](#)

3.9 caer_edvs_info Struct Reference

```
#include <edvs.h>
```

Data Fields

- `int16_t` [deviceId](#)
Unique device identifier. Also 'source' for events.
- `char *` [deviceString](#)
- `bool` [devicesMaster](#)
Whether the device is a time-stamp master or slave.
- `int16_t` [dvsSizeX](#)
DVS X axis resolution.
- `int16_t` [dvsSizeY](#)
DVS Y axis resolution.
- `char` [serialPortName](#) [64]
Connected serial port name (OS-specific).
- `uint32_t` [serialBaudRate](#)
Serial connection baud-rate.

3.9.1 Detailed Description

EDVS device-related information.

3.9.2 Field Documentation

3.9.2.1 deviceString

```
char* caer_edvs_info::deviceString
```

Device information string, for logging purposes. If not NULL, pointed-to memory is *only* valid while the corresponding device is open! After calling deviceClose() this is invalid memory!

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

- [devices/edvs.h](#)

3.10 caer_filter_dvs_pixel Struct Reference

```
#include <dvs_noise.h>
```

Data Fields

- uint16_t **x**
- uint16_t **y**

3.10.1 Detailed Description

Structure representing a single DVS pixel address, with X and Y components. Used in DVS filtering support.

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

- [filters/dvs_noise.h](#)

Chapter 4

File Documentation

4.1 devices/davis.h File Reference

```
#include "../events/frame.h"
#include "../events/imu6.h"
#include "../events/polarity.h"
#include "../events/special.h"
#include "usb.h"
```

Data Structures

- struct [caer_davis_info](#)
- struct [caer_bias_vdac](#)
- struct [caer_bias_coarsefine](#)
- struct [caer_bias_shiftedsources](#)

Macros

- #define [CAER_DEVICE_DAVIS_FX2](#) 1
- #define [CAER_DEVICE_DAVIS_FX3](#) 2
- #define [CAER_DEVICE_DAVIS](#) 4
- #define [CAER_DEVICE_DAVIS_RPI](#) 6
- #define [DAVIS_CHIP_DAVIS240A](#) 0
- #define [DAVIS_CHIP_DAVIS240B](#) 1
- #define [DAVIS_CHIP_DAVIS240C](#) 2
- #define [DAVIS_CHIP_DAVIS128](#) 3
- #define [DAVIS_CHIP_DAVIS346A](#) 4
- #define [DAVIS_CHIP_DAVIS346B](#) 5
- #define [DAVIS_CHIP_DAVIS640](#) 6
- #define [DAVIS_CHIP_DAVISRGB](#) 7
- #define [DAVIS_CHIP_DAVIS208](#) 8
- #define [DAVIS_CHIP_DAVIS346C](#) 9
- #define [DAVIS_CONFIG_MUX](#) 0
- #define [DAVIS_CONFIG_DVS](#) 1
- #define [DAVIS_CONFIG_APS](#) 2

- #define [DAVIS_CONFIG_IMU](#) 3
- #define [DAVIS_CONFIG_EXTINPUT](#) 4
- #define [DAVIS_CONFIG_BIAS](#) 5
- #define [DAVIS_CONFIG_CHIP](#) 5
- #define [DAVIS_CONFIG_SYSINFO](#) 6
- #define [DAVIS_CONFIG_USB](#) 9
- #define [DAVIS_CONFIG_DDRAER](#) 9
- #define [DAVIS_CONFIG_MUX_RUN](#) 0
- #define [DAVIS_CONFIG_MUX_TIMESTAMP_RUN](#) 1
- #define [DAVIS_CONFIG_MUX_TIMESTAMP_RESET](#) 2
- #define [DAVIS_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE](#) 3
- #define [DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL](#) 4
- #define [DAVIS_CONFIG_MUX_DROP_APS_ON_TRANSFER_STALL](#) 5
- #define [DAVIS_CONFIG_MUX_DROP_IMU_ON_TRANSFER_STALL](#) 6
- #define [DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL](#) 7
- #define [DAVIS_CONFIG_MUX_HAS_STATISTICS](#) 10
- #define [DAVIS_CONFIG_MUX_STATISTICS_DVS_DROPPED](#) 11
- #define [DAVIS_CONFIG_MUX_STATISTICS_APS_DROPPED](#) 13
- #define [DAVIS_CONFIG_MUX_STATISTICS_IMU_DROPPED](#) 15
- #define [DAVIS_CONFIG_MUX_STATISTICS_EXTINPUT_DROPPED](#) 17
- #define [DAVIS_CONFIG_DVS_SIZE_COLUMNS](#) 0
- #define [DAVIS_CONFIG_DVS_SIZE_ROWS](#) 1
- #define [DAVIS_CONFIG_DVS_ORIENTATION_INFO](#) 2
- #define [DAVIS_CONFIG_DVS_RUN](#) 3
- #define [DAVIS_CONFIG_DVS_ACK_DELAY_ROW](#) 4
- #define [DAVIS_CONFIG_DVS_ACK_DELAY_COLUMN](#) 5
- #define [DAVIS_CONFIG_DVS_ACK_EXTENSION_ROW](#) 6
- #define [DAVIS_CONFIG_DVS_ACK_EXTENSION_COLUMN](#) 7
- #define [DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL](#) 8
- #define [DAVIS_CONFIG_DVS_FILTER_ROW_ONLY_EVENTS](#) 9
- #define [DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL](#) 10
- #define [DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER](#) 11
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW](#) 12
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN](#) 13
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW](#) 14
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN](#) 15
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW](#) 16
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN](#) 17
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW](#) 18
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN](#) 19
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW](#) 20
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN](#) 21
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW](#) 22
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN](#) 23
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW](#) 24
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN](#) 25
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW](#) 26
- #define [DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN](#) 27
- #define [DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER](#) 28
- #define [DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY](#) 29
- #define [DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_TIME](#) 30
- #define [DAVIS_CONFIG_DVS_HAS_TEST_EVENT_GENERATOR](#) 31
- #define [DAVIS_CONFIG_DVS_TEST_EVENT_GENERATOR_ENABLE](#) 32
- #define [DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD](#) 33
- #define [DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD_TIME](#) 34

- `#define DAVIS_CONFIG_DVS_HAS_ROI_FILTER` 35
- `#define DAVIS_CONFIG_DVS_FILTER_ROI_START_COLUMN` 36
- `#define DAVIS_CONFIG_DVS_FILTER_ROI_START_ROW` 37
- `#define DAVIS_CONFIG_DVS_FILTER_ROI_END_COLUMN` 38
- `#define DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW` 39
- `#define DAVIS_CONFIG_DVS_HAS_STATISTICS` 40
- `#define DAVIS_CONFIG_DVS_STATISTICS_EVENTS_ROW` 41
- `#define DAVIS_CONFIG_DVS_STATISTICS_EVENTS_COLUMN` 43
- `#define DAVIS_CONFIG_DVS_STATISTICS_EVENTS_DROPPED` 45
- `#define DAVIS_CONFIG_DVS_STATISTICS_FILTERED_PIXELS` 47
- `#define DAVIS_CONFIG_DVS_STATISTICS_FILTERED_BACKGROUND_ACTIVITY` 49
- `#define DAVIS_CONFIG_DVS_STATISTICS_FILTERED_REFRACTORY_PERIOD` 51
- `#define DAVIS_CONFIG_DVS_FILTER_PIXEL_AUTO_TRAIN` 80
- `#define DAVIS_CONFIG_APS_SIZE_COLUMNS` 0
- `#define DAVIS_CONFIG_APS_SIZE_ROWS` 1
- `#define DAVIS_CONFIG_APS_ORIENTATION_INFO` 2
- `#define DAVIS_CONFIG_APS_COLOR_FILTER` 3
- `#define DAVIS_CONFIG_APS_RUN` 4
- `#define DAVIS_CONFIG_APS_RESET_READ` 5
- `#define DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL` 6
- `#define DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER` 7
- `#define DAVIS_CONFIG_APS_GLOBAL_SHUTTER` 8
- `#define DAVIS_CONFIG_APS_START_COLUMN_0` 9
- `#define DAVIS_CONFIG_APS_START_ROW_0` 10
- `#define DAVIS_CONFIG_APS_END_COLUMN_0` 11
- `#define DAVIS_CONFIG_APS_END_ROW_0` 12
- `#define DAVIS_CONFIG_APS_EXPOSURE` 13
- `#define DAVIS_CONFIG_APS_FRAME_DELAY` 14
- `#define DAVIS_CONFIG_APS_RESET_SETTLE` 15
- `#define DAVIS_CONFIG_APS_COLUMN_SETTLE` 16
- `#define DAVIS_CONFIG_APS_ROW_SETTLE` 17
- `#define DAVIS_CONFIG_APS_NULL_SETTLE` 18
- `#define DAVIS_CONFIG_APS_HAS_QUAD_ROI` 19
- `#define DAVIS_CONFIG_APS_START_COLUMN_1` 20
- `#define DAVIS_CONFIG_APS_START_ROW_1` 21
- `#define DAVIS_CONFIG_APS_END_COLUMN_1` 22
- `#define DAVIS_CONFIG_APS_END_ROW_1` 23
- `#define DAVIS_CONFIG_APS_START_COLUMN_2` 24
- `#define DAVIS_CONFIG_APS_START_ROW_2` 25
- `#define DAVIS_CONFIG_APS_END_COLUMN_2` 26
- `#define DAVIS_CONFIG_APS_END_ROW_2` 27
- `#define DAVIS_CONFIG_APS_START_COLUMN_3` 28
- `#define DAVIS_CONFIG_APS_START_ROW_3` 29
- `#define DAVIS_CONFIG_APS_END_COLUMN_3` 30
- `#define DAVIS_CONFIG_APS_END_ROW_3` 31
- `#define DAVIS_CONFIG_APS_HAS_INTERNAL_ADC` 33
- `#define DAVIS_CONFIG_APS_SAMPLE_ENABLE` 35
- `#define DAVIS_CONFIG_APS_SAMPLE_SETTLE` 36
- `#define DAVIS_CONFIG_APS_RAMP_RESET` 37
- `#define DAVIS_CONFIG_APS_RAMP_SHORT_RESET` 38
- `#define DAVIS_CONFIG_APS_ADC_TEST_MODE` 39
- `#define DAVIS_CONFIG_APS_ROI0_ENABLED` 40
- `#define DAVIS_CONFIG_APS_ROI1_ENABLED` 41
- `#define DAVIS_CONFIG_APS_ROI2_ENABLED` 42
- `#define DAVIS_CONFIG_APS_ROI3_ENABLED` 43

- #define [DAVISRGB_CONFIG_APS_TRANSFER](#) 50
- #define [DAVISRGB_CONFIG_APS_RSFDSETTLE](#) 51
- #define [DAVISRGB_CONFIG_APS_GSPDRESET](#) 52
- #define [DAVISRGB_CONFIG_APS_GSRESETFALL](#) 53
- #define [DAVISRGB_CONFIG_APS_GSTXFALL](#) 54
- #define [DAVISRGB_CONFIG_APS_GSFDRESET](#) 55
- #define [DAVIS_CONFIG_APS_SNAPSHOT](#) 80
- #define [DAVIS_CONFIG_APS_AUTOEXPOSURE](#) 81
- #define [DAVIS_CONFIG_IMU_RUN](#) 0
- #define [DAVIS_CONFIG_IMU_TEMP_STANDBY](#) 1
- #define [DAVIS_CONFIG_IMU_ACCEL_STANDBY](#) 2
- #define [DAVIS_CONFIG_IMU_GYRO_STANDBY](#) 3
- #define [DAVIS_CONFIG_IMU_LP_CYCLE](#) 4
- #define [DAVIS_CONFIG_IMU_LP_WAKEUP](#) 5
- #define [DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER](#) 6
- #define [DAVIS_CONFIG_IMU_GYRO_DLPF](#) 7
- #define **DAVIS_CONFIG_IMU_DIGITAL_LOW_PASS_FILTER** [DAVIS_CONFIG_IMU_GYRO_DLPF](#)
- #define [DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE](#) 8
- #define [DAVIS_CONFIG_IMU_GYRO_FULL_SCALE](#) 9
- #define [DAVIS_CONFIG_IMU_ORIENTATION_INFO](#) 10
- #define [DAVIS_CONFIG_IMU_ACCEL_DLPF](#) 11
- #define [DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR](#) 0
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES](#) 1
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES](#) 2
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_PULSES](#) 3
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY](#) 4
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH](#) 5
- #define [DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR](#) 6
- #define [DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR](#) 7
- #define [DAVIS_CONFIG_EXTINPUT_GENERATE_USE_CUSTOM_SIGNAL](#) 8
- #define [DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY](#) 9
- #define [DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL](#) 10
- #define [DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH](#) 11
- #define [DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_RISING_EDGE](#) 12
- #define [DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_FALLING_EDGE](#) 13
- #define [DAVIS_CONFIG_EXTINPUT_HAS_EXTRA_DETECTORS](#) 14
- #define [DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR1](#) 15
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES1](#) 16
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES1](#) 17
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_PULSES1](#) 18
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY1](#) 19
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH1](#) 20
- #define [DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR2](#) 21
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES2](#) 22
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES2](#) 23
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_PULSES2](#) 24
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY2](#) 25
- #define [DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH2](#) 26
- #define [DAVIS_CONFIG_SYSINFO_LOGIC_VERSION](#) 0
- #define [DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER](#) 1
- #define [DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER](#) 2
- #define [DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK](#) 3
- #define [DAVIS_CONFIG_SYSINFO_ADC_CLOCK](#) 4
- #define [DAVIS_CONFIG_USB_RUN](#) 0
- #define [DAVIS_CONFIG_USB_EARLY_PACKET_DELAY](#) 1

- #define [DAVIS_CONFIG_DDRAER_RUN](#) 0
 - #define [DAVIS_CONFIG_DDRAER_REQ_DELAY](#) 1
 - #define [DAVIS_CONFIG_DDRAER_ACK_DELAY](#) 2
 - #define [DAVIS_APS_ROI_REGIONS_MAX](#) 4
-
- #define [IS_DAVIS128\(chipID\)](#) ((chipID) == [DAVIS_CHIP_DAVIS128](#))
 - #define [IS_DAVIS208\(chipID\)](#) ((chipID) == [DAVIS_CHIP_DAVIS208](#))
 - #define [IS_DAVIS240A\(chipID\)](#) ((chipID) == [DAVIS_CHIP_DAVIS240A](#))
 - #define [IS_DAVIS240B\(chipID\)](#) ((chipID) == [DAVIS_CHIP_DAVIS240B](#))
 - #define [IS_DAVIS240C\(chipID\)](#) ((chipID) == [DAVIS_CHIP_DAVIS240C](#))
 - #define [IS_DAVIS240\(chipID\)](#) ([IS_DAVIS240A\(chipID\)](#) || [IS_DAVIS240B\(chipID\)](#) || [IS_DAVIS240C\(chipID\)](#))
 - #define [IS_DAVIS346A\(chipID\)](#) ((chipID) == [DAVIS_CHIP_DAVIS346A](#))
 - #define [IS_DAVIS346B\(chipID\)](#) ((chipID) == [DAVIS_CHIP_DAVIS346B](#))
 - #define [IS_DAVIS346C\(chipID\)](#) ((chipID) == [DAVIS_CHIP_DAVIS346C](#))
 - #define [IS_DAVIS346\(chipID\)](#) ([IS_DAVIS346A\(chipID\)](#) || [IS_DAVIS346B\(chipID\)](#) || [IS_DAVIS346C\(chipID\)](#))
 - #define [IS_DAVIS640\(chipID\)](#) ((chipID) == [DAVIS_CHIP_DAVIS640](#))
 - #define [IS_DAVISRGB\(chipID\)](#) ((chipID) == [DAVIS_CHIP_DAVISRGB](#))
-
- #define [DAVIS128_CONFIG_BIAS_APSOVERFLOWLEVEL](#) 0
 - #define [DAVIS128_CONFIG_BIAS_APSCAS](#) 1
 - #define [DAVIS128_CONFIG_BIAS_ADCREFHIGH](#) 2
 - #define [DAVIS128_CONFIG_BIAS_ADCREFLOW](#) 3
 - #define [DAVIS128_CONFIG_BIAS_LOCALBUFBN](#) 8
 - #define [DAVIS128_CONFIG_BIAS_PADFOLLBN](#) 9
 - #define [DAVIS128_CONFIG_BIAS_DIFFBN](#) 10
 - #define [DAVIS128_CONFIG_BIAS_ONBN](#) 11
 - #define [DAVIS128_CONFIG_BIAS_OFFBN](#) 12
 - #define [DAVIS128_CONFIG_BIAS_PIXINBN](#) 13
 - #define [DAVIS128_CONFIG_BIAS_PRBP](#) 14
 - #define [DAVIS128_CONFIG_BIAS_PRSBP](#) 15
 - #define [DAVIS128_CONFIG_BIAS_REFRBP](#) 16
 - #define [DAVIS128_CONFIG_BIAS_READOUTBUFBP](#) 17
 - #define [DAVIS128_CONFIG_BIAS_APSROSBN](#) 18
 - #define [DAVIS128_CONFIG_BIAS_ADCCOMPBP](#) 19
 - #define [DAVIS128_CONFIG_BIAS_COLSELLOWBN](#) 20
 - #define [DAVIS128_CONFIG_BIAS_DACBUFBP](#) 21
 - #define [DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN](#) 22
 - #define [DAVIS128_CONFIG_BIAS_AEPDBN](#) 23
 - #define [DAVIS128_CONFIG_BIAS_AEPUXBP](#) 24
 - #define [DAVIS128_CONFIG_BIAS_AEPUYBP](#) 25
 - #define [DAVIS128_CONFIG_BIAS_IFREFRBN](#) 26
 - #define [DAVIS128_CONFIG_BIAS_IFTHRBN](#) 27
 - #define [DAVIS128_CONFIG_BIAS_BIASBUFFER](#) 34
 - #define [DAVIS128_CONFIG_BIAS_SSP](#) 35
 - #define [DAVIS128_CONFIG_BIAS_SSN](#) 36

- #define [DAVIS128_CONFIG_CHIP_DIGITALMUX0](#) 128
- #define [DAVIS128_CONFIG_CHIP_DIGITALMUX1](#) 129
- #define [DAVIS128_CONFIG_CHIP_DIGITALMUX2](#) 130
- #define [DAVIS128_CONFIG_CHIP_DIGITALMUX3](#) 131
- #define [DAVIS128_CONFIG_CHIP_ANALOGMUX0](#) 132
- #define [DAVIS128_CONFIG_CHIP_ANALOGMUX1](#) 133
- #define [DAVIS128_CONFIG_CHIP_ANALOGMUX2](#) 134
- #define [DAVIS128_CONFIG_CHIP_BIASMUX0](#) 135
- #define [DAVIS128_CONFIG_CHIP_RESETCALIBNEURON](#) 136
- #define [DAVIS128_CONFIG_CHIP_TYPENCALIBNEURON](#) 137
- #define [DAVIS128_CONFIG_CHIP_RESETTESTPIXEL](#) 138
- #define [DAVIS128_CONFIG_CHIP_AERNAROW](#) 140
- #define [DAVIS128_CONFIG_CHIP_USEAOUT](#) 141
- #define [DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER](#) 142
- #define [DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER](#) 143

- #define [DAVIS208_CONFIG_BIAS_APSOEVERFLOWLEVEL](#) 0
- #define [DAVIS208_CONFIG_BIAS_APSCAS](#) 1
- #define [DAVIS208_CONFIG_BIAS_ADCREFHIGH](#) 2
- #define [DAVIS208_CONFIG_BIAS_ADCREFLOW](#) 3
- #define [DAVIS208_CONFIG_BIAS_RESETHIGHPASS](#) 6
- #define [DAVIS208_CONFIG_BIAS_REFSS](#) 7
- #define [DAVIS208_CONFIG_BIAS_LOCALBUFBN](#) 8
- #define [DAVIS208_CONFIG_BIAS_PADFOLLBN](#) 9
- #define [DAVIS208_CONFIG_BIAS_DIFFBN](#) 10
- #define [DAVIS208_CONFIG_BIAS_ONBN](#) 11
- #define [DAVIS208_CONFIG_BIAS_OFFBN](#) 12
- #define [DAVIS208_CONFIG_BIAS_PIXINVBN](#) 13
- #define [DAVIS208_CONFIG_BIAS_PRBP](#) 14
- #define [DAVIS208_CONFIG_BIAS_PRSFBP](#) 15
- #define [DAVIS208_CONFIG_BIAS_REFRBP](#) 16
- #define [DAVIS208_CONFIG_BIAS_READOUTBUFBP](#) 17
- #define [DAVIS208_CONFIG_BIAS_APSROSFBN](#) 18
- #define [DAVIS208_CONFIG_BIAS_ADCCOMPBP](#) 19
- #define [DAVIS208_CONFIG_BIAS_COLSELLOWBN](#) 20
- #define [DAVIS208_CONFIG_BIAS_DACBUFBP](#) 21
- #define [DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN](#) 22
- #define [DAVIS208_CONFIG_BIAS_AEPDBN](#) 23
- #define [DAVIS208_CONFIG_BIAS_AEPUXBP](#) 24
- #define [DAVIS208_CONFIG_BIAS_AEPUYBP](#) 25
- #define [DAVIS208_CONFIG_BIAS_IFREFRBN](#) 26
- #define [DAVIS208_CONFIG_BIAS_IFTHRBN](#) 27
- #define [DAVIS208_CONFIG_BIAS_REGBIASBP](#) 28
- #define [DAVIS208_CONFIG_BIAS_REFSSBN](#) 30
- #define [DAVIS208_CONFIG_BIAS_BIASBUFFER](#) 34
- #define [DAVIS208_CONFIG_BIAS_SSP](#) 35
- #define [DAVIS208_CONFIG_BIAS_SSN](#) 36

- #define [DAVIS208_CONFIG_CHIP_DIGITALMUX0](#) 128
- #define [DAVIS208_CONFIG_CHIP_DIGITALMUX1](#) 129
- #define [DAVIS208_CONFIG_CHIP_DIGITALMUX2](#) 130
- #define [DAVIS208_CONFIG_CHIP_DIGITALMUX3](#) 131
- #define [DAVIS208_CONFIG_CHIP_ANALOGMUX0](#) 132
- #define [DAVIS208_CONFIG_CHIP_ANALOGMUX1](#) 133
- #define [DAVIS208_CONFIG_CHIP_ANALOGMUX2](#) 134
- #define [DAVIS208_CONFIG_CHIP_BIASMUX0](#) 135
- #define [DAVIS208_CONFIG_CHIP_RESETCALIBNEURON](#) 136
- #define [DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON](#) 137
- #define [DAVIS208_CONFIG_CHIP_RESETTESTPIXEL](#) 138
- #define [DAVIS208_CONFIG_CHIP_AERNAROW](#) 140
- #define [DAVIS208_CONFIG_CHIP_USEAOUT](#) 141
- #define [DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER](#) 142
- #define [DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER](#) 143
- #define [DAVIS208_CONFIG_CHIP_SELECTPREAMPVG](#) 145
- #define [DAVIS208_CONFIG_CHIP_SELECTBIASREFSS](#) 146
- #define [DAVIS208_CONFIG_CHIP_SELECTSENSE](#) 147
- #define [DAVIS208_CONFIG_CHIP_SELECTPOSFB](#) 148
- #define [DAVIS208_CONFIG_CHIP_SELECTHIGHPASS](#) 149

- #define [DAVIS240_CONFIG_BIAS_DIFFBN](#) 0
- #define [DAVIS240_CONFIG_BIAS_ONBN](#) 1
- #define [DAVIS240_CONFIG_BIAS_OFFBN](#) 2
- #define [DAVIS240_CONFIG_BIAS_APSCASEPC](#) 3
- #define [DAVIS240_CONFIG_BIAS_DIFFCASBNC](#) 4
- #define [DAVIS240_CONFIG_BIAS_APSROSFBN](#) 5
- #define [DAVIS240_CONFIG_BIAS_LOCALBUFBN](#) 6
- #define [DAVIS240_CONFIG_BIAS_PIXINVBN](#) 7
- #define [DAVIS240_CONFIG_BIAS_PRBP](#) 8
- #define [DAVIS240_CONFIG_BIAS_PRSFBP](#) 9
- #define [DAVIS240_CONFIG_BIAS_REFRBP](#) 10
- #define [DAVIS240_CONFIG_BIAS_AEPDBN](#) 11
- #define [DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN](#) 12
- #define [DAVIS240_CONFIG_BIAS_AEPUXBP](#) 13
- #define [DAVIS240_CONFIG_BIAS_AEPUYBP](#) 14
- #define [DAVIS240_CONFIG_BIAS_IFTHRBN](#) 15
- #define [DAVIS240_CONFIG_BIAS_IFREFRBN](#) 16
- #define [DAVIS240_CONFIG_BIAS_PADFOLLBN](#) 17
- #define [DAVIS240_CONFIG_BIAS_APSOEVERFLOWLEVELBN](#) 18
- #define [DAVIS240_CONFIG_BIAS_BIASBUFFER](#) 19
- #define [DAVIS240_CONFIG_BIAS_SSP](#) 20
- #define [DAVIS240_CONFIG_BIAS_SSN](#) 21

- #define [DAVIS240_CONFIG_CHIP_DIGITALMUX0](#) 128
- #define [DAVIS240_CONFIG_CHIP_DIGITALMUX1](#) 129
- #define [DAVIS240_CONFIG_CHIP_DIGITALMUX2](#) 130
- #define [DAVIS240_CONFIG_CHIP_DIGITALMUX3](#) 131

- `#define DAVIS240_CONFIG_CHIP_ANALOGMUX0` 132
- `#define DAVIS240_CONFIG_CHIP_ANALOGMUX1` 133
- `#define DAVIS240_CONFIG_CHIP_ANALOGMUX2` 134
- `#define DAVIS240_CONFIG_CHIP_BIASMUX0` 135
- `#define DAVIS240_CONFIG_CHIP_RESETCALIBNEURON` 136
- `#define DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON` 137
- `#define DAVIS240_CONFIG_CHIP_RESETTESTPIXEL` 138
- `#define DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL` 139
- `#define DAVIS240_CONFIG_CHIP_AERNAROW` 140
- `#define DAVIS240_CONFIG_CHIP_USEAOUT` 141
- `#define DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER` 142

- `#define DAVIS346_CONFIG_BIAS_APSOEVERFLOWLEVEL` 0
- `#define DAVIS346_CONFIG_BIAS_APSCAS` 1
- `#define DAVIS346_CONFIG_BIAS_ADCREFHIGH` 2
- `#define DAVIS346_CONFIG_BIAS_ADCREFLOW` 3
- `#define DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE` 4
- `#define DAVIS346_CONFIG_BIAS_LOCALBUFBN` 8
- `#define DAVIS346_CONFIG_BIAS_PADFOLLBN` 9
- `#define DAVIS346_CONFIG_BIAS_DIFFBN` 10
- `#define DAVIS346_CONFIG_BIAS_ONBN` 11
- `#define DAVIS346_CONFIG_BIAS_OFFBN` 12
- `#define DAVIS346_CONFIG_BIAS_PIXINVBN` 13
- `#define DAVIS346_CONFIG_BIAS_PRBP` 14
- `#define DAVIS346_CONFIG_BIAS_PRSFBP` 15
- `#define DAVIS346_CONFIG_BIAS_REFRBP` 16
- `#define DAVIS346_CONFIG_BIAS_READOUTBUFBP` 17
- `#define DAVIS346_CONFIG_BIAS_APSROSFBP` 18
- `#define DAVIS346_CONFIG_BIAS_ADCCOMPBP` 19
- `#define DAVIS346_CONFIG_BIAS_COLSELOWBN` 20
- `#define DAVIS346_CONFIG_BIAS_DACBUFBP` 21
- `#define DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN` 22
- `#define DAVIS346_CONFIG_BIAS_AEPDBN` 23
- `#define DAVIS346_CONFIG_BIAS_AEPUXBP` 24
- `#define DAVIS346_CONFIG_BIAS_AEPUYBP` 25
- `#define DAVIS346_CONFIG_BIAS_IFREFRBN` 26
- `#define DAVIS346_CONFIG_BIAS_IFTHRBN` 27
- `#define DAVIS346_CONFIG_BIAS_BIASBUFFER` 34
- `#define DAVIS346_CONFIG_BIAS_SSP` 35
- `#define DAVIS346_CONFIG_BIAS_SSN` 36

- `#define DAVIS346_CONFIG_CHIP_DIGITALMUX0` 128
- `#define DAVIS346_CONFIG_CHIP_DIGITALMUX1` 129
- `#define DAVIS346_CONFIG_CHIP_DIGITALMUX2` 130
- `#define DAVIS346_CONFIG_CHIP_DIGITALMUX3` 131
- `#define DAVIS346_CONFIG_CHIP_ANALOGMUX0` 132
- `#define DAVIS346_CONFIG_CHIP_ANALOGMUX1` 133
- `#define DAVIS346_CONFIG_CHIP_ANALOGMUX2` 134

- #define [DAVIS346_CONFIG_CHIP_BIASMUX0](#) 135
- #define [DAVIS346_CONFIG_CHIP_RESETCALIBNEURON](#) 136
- #define [DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON](#) 137
- #define [DAVIS346_CONFIG_CHIP_RESETTESTPIXEL](#) 138
- #define [DAVIS346_CONFIG_CHIP_AERNAROW](#) 140
- #define [DAVIS346_CONFIG_CHIP_USEAOUT](#) 141
- #define [DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER](#) 142
- #define [DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER](#) 143
- #define [DAVIS346_CONFIG_CHIP_TESTADC](#) 144

- #define [DAVIS640_CONFIG_BIAS_APSOEVERFLOWLEVEL](#) 0
- #define [DAVIS640_CONFIG_BIAS_APSCAS](#) 1
- #define [DAVIS640_CONFIG_BIAS_ADCREFHIGH](#) 2
- #define [DAVIS640_CONFIG_BIAS_ADCREFLOW](#) 3
- #define [DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE](#) 4
- #define [DAVIS640_CONFIG_BIAS_LOCALBUFBN](#) 8
- #define [DAVIS640_CONFIG_BIAS_PADFOLLBN](#) 9
- #define [DAVIS640_CONFIG_BIAS_DIFFBN](#) 10
- #define [DAVIS640_CONFIG_BIAS_ONBN](#) 11
- #define [DAVIS640_CONFIG_BIAS_OFFBN](#) 12
- #define [DAVIS640_CONFIG_BIAS_PIXINVBN](#) 13
- #define [DAVIS640_CONFIG_BIAS_PRBP](#) 14
- #define [DAVIS640_CONFIG_BIAS_PRSFBP](#) 15
- #define [DAVIS640_CONFIG_BIAS_REFRBP](#) 16
- #define [DAVIS640_CONFIG_BIAS_READOUTBUFBP](#) 17
- #define [DAVIS640_CONFIG_BIAS_APSROSFBN](#) 18
- #define [DAVIS640_CONFIG_BIAS_ADCCOMPBP](#) 19
- #define [DAVIS640_CONFIG_BIAS_COLSELLOWBN](#) 20
- #define [DAVIS640_CONFIG_BIAS_DACBUFBP](#) 21
- #define [DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN](#) 22
- #define [DAVIS640_CONFIG_BIAS_AEPDBN](#) 23
- #define [DAVIS640_CONFIG_BIAS_AEPUXBP](#) 24
- #define [DAVIS640_CONFIG_BIAS_AEPUYBP](#) 25
- #define [DAVIS640_CONFIG_BIAS_IFREFRBN](#) 26
- #define [DAVIS640_CONFIG_BIAS_IFTHRBN](#) 27
- #define [DAVIS640_CONFIG_BIAS_BIASBUFFER](#) 34
- #define [DAVIS640_CONFIG_BIAS_SSP](#) 35
- #define [DAVIS640_CONFIG_BIAS_SSN](#) 36

- #define [DAVIS640_CONFIG_CHIP_DIGITALMUX0](#) 128
- #define [DAVIS640_CONFIG_CHIP_DIGITALMUX1](#) 129
- #define [DAVIS640_CONFIG_CHIP_DIGITALMUX2](#) 130
- #define [DAVIS640_CONFIG_CHIP_DIGITALMUX3](#) 131
- #define [DAVIS640_CONFIG_CHIP_ANALOGMUX0](#) 132
- #define [DAVIS640_CONFIG_CHIP_ANALOGMUX1](#) 133
- #define [DAVIS640_CONFIG_CHIP_ANALOGMUX2](#) 134
- #define [DAVIS640_CONFIG_CHIP_BIASMUX0](#) 135
- #define [DAVIS640_CONFIG_CHIP_RESETCALIBNEURON](#) 136

- #define [DAVIS640_CONFIG_CHIP_TYPENCALIBNEURON](#) 137
- #define [DAVIS640_CONFIG_CHIP_RESETTESTPIXEL](#) 138
- #define [DAVIS640_CONFIG_CHIP_AERNAROW](#) 140
- #define [DAVIS640_CONFIG_CHIP_USEAOUT](#) 141
- #define [DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER](#) 142
- #define [DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER](#) 143
- #define [DAVIS640_CONFIG_CHIP_TESTADC](#) 144

- #define [DAVISRGB_CONFIG_BIAS_APSCAS](#) 0
- #define [DAVISRGB_CONFIG_BIAS_OVG1LO](#) 1
- #define [DAVISRGB_CONFIG_BIAS_OVG2LO](#) 2
- #define [DAVISRGB_CONFIG_BIAS_TX2OVG2HI](#) 3
- #define [DAVISRGB_CONFIG_BIAS_GND07](#) 4
- #define [DAVISRGB_CONFIG_BIAS_ADCTESTVOLTAGE](#) 5
- #define [DAVISRGB_CONFIG_BIAS_ADCREFHIGH](#) 6
- #define [DAVISRGB_CONFIG_BIAS_ADCREFLOW](#) 7
- #define [DAVISRGB_CONFIG_BIAS_IFREFRBN](#) 8
- #define [DAVISRGB_CONFIG_BIAS_IFTHRBN](#) 9
- #define [DAVISRGB_CONFIG_BIAS_LOCALBUFBN](#) 10
- #define [DAVISRGB_CONFIG_BIAS_PADFOLLBN](#) 11
- #define [DAVISRGB_CONFIG_BIAS_PIXINBN](#) 13
- #define [DAVISRGB_CONFIG_BIAS_DIFFBN](#) 14
- #define [DAVISRGB_CONFIG_BIAS_ONBN](#) 15
- #define [DAVISRGB_CONFIG_BIAS_OFFBN](#) 16
- #define [DAVISRGB_CONFIG_BIAS_PRBP](#) 17
- #define [DAVISRGB_CONFIG_BIAS_PRSFBP](#) 18
- #define [DAVISRGB_CONFIG_BIAS_REFRBP](#) 19
- #define [DAVISRGB_CONFIG_BIAS_ARRAYBIASBUFFERBN](#) 20
- #define [DAVISRGB_CONFIG_BIAS_ARRAYLOGICBUFFERBN](#) 22
- #define [DAVISRGB_CONFIG_BIAS_FALLTIMEBN](#) 23
- #define [DAVISRGB_CONFIG_BIAS_RISETIMEBP](#) 24
- #define [DAVISRGB_CONFIG_BIAS_READOUTBUFBP](#) 25
- #define [DAVISRGB_CONFIG_BIAS_APSROSFBN](#) 26
- #define [DAVISRGB_CONFIG_BIAS_ADCCOMPBP](#) 27
- #define [DAVISRGB_CONFIG_BIAS_DACBUFBP](#) 28
- #define [DAVISRGB_CONFIG_BIAS_LCOLTIMEOUTBN](#) 30
- #define [DAVISRGB_CONFIG_BIAS_AEPDBN](#) 31
- #define [DAVISRGB_CONFIG_BIAS_AEPUXBP](#) 32
- #define [DAVISRGB_CONFIG_BIAS_AEPUYBP](#) 33
- #define [DAVISRGB_CONFIG_BIAS_BIASBUFFER](#) 34
- #define [DAVISRGB_CONFIG_BIAS_SSP](#) 35
- #define [DAVISRGB_CONFIG_BIAS_SSN](#) 36

- #define [DAVISRGB_CONFIG_CHIP_DIGITALMUX0](#) 128
- #define [DAVISRGB_CONFIG_CHIP_DIGITALMUX1](#) 129
- #define [DAVISRGB_CONFIG_CHIP_DIGITALMUX2](#) 130
- #define [DAVISRGB_CONFIG_CHIP_DIGITALMUX3](#) 131
- #define [DAVISRGB_CONFIG_CHIP_ANALOGMUX0](#) 132

- `#define DAVISRGB_CONFIG_CHIP_ANALOGMUX1` 133
- `#define DAVISRGB_CONFIG_CHIP_ANALOGMUX2` 134
- `#define DAVISRGB_CONFIG_CHIP_BIASMUX0` 135
- `#define DAVISRGB_CONFIG_CHIP_RESETCALIBNEURON` 136
- `#define DAVISRGB_CONFIG_CHIP_TYPCALIBNEURON` 137
- `#define DAVISRGB_CONFIG_CHIP_RESETTESTPIXEL` 138
- `#define DAVISRGB_CONFIG_CHIP_AERNAROW` 140
- `#define DAVISRGB_CONFIG_CHIP_USEAOUT` 141
- `#define DAVISRGB_CONFIG_CHIP_SELECTGRAYCOUNTER` 143
- `#define DAVISRGB_CONFIG_CHIP_TESTADC` 144
- `#define DAVISRGB_CONFIG_CHIP_ADJUSTOVG1LO` 145
- `#define DAVISRGB_CONFIG_CHIP_ADJUSTOVG2LO` 146
- `#define DAVISRGB_CONFIG_CHIP_ADJUSTTX2OVG2HI` 147

Enumerations

- enum `caer_bias_shiftedsource_operating_mode` { `SHIFTED_SOURCE` = 0, `HI_Z` = 1, `TIED_TO_RAIL` = 2 }
- enum `caer_bias_shiftedsource_voltage_level` { `SPLIT_GATE` = 0, `SINGLE_DIODE` = 1, `DOUBLE_DIODE` = 2 }

Functions

- struct `caer_davis_info` `caerDavisInfoGet` (`caerDeviceHandle` handle)
- `uint16_t` `caerBiasVDACGenerate` (const struct `caer_bias_vdac` `vdacBias`)
- struct `caer_bias_vdac` `caerBiasVDACParse` (const `uint16_t` `vdacBias`)
- `uint16_t` `caerBiasCoarseFineGenerate` (const struct `caer_bias_coarsefine` `coarseFineBias`)
- struct `caer_bias_coarsefine` `caerBiasCoarseFineParse` (const `uint16_t` `coarseFineBias`)
- `uint16_t` `caerBiasShiftedSourceGenerate` (const struct `caer_bias_shiftedsource` `shiftedSourceBias`)
- struct `caer_bias_shiftedsource` `caerBiasShiftedSourceParse` (const `uint16_t` `shiftedSourceBias`)
- bool `caerDavisROIConfigure` (`caerDeviceHandle` handle, `uint8_t` `roiRegion`, bool `enable`, `uint16_t` `startX`, `uint16_t` `startY`, `uint16_t` `endX`, `uint16_t` `endY`)

4.1.1 Detailed Description

DAVIS specific configuration defines and information structures.

4.1.2 Macro Definition Documentation

4.1.2.1 CAER_DEVICE_DAVIS

```
#define CAER_DEVICE_DAVIS 4
```

Device type definition for iniVation DAVIS boards, supporting both FX2 and FX3 generation devices. This is the preferred way to access cameras now.

4.1.2.2 CAER_DEVICE_DAVIS_FX2

```
#define CAER_DEVICE_DAVIS_FX2 1
```

Device type definition for iniVation DAVIS FX2-based boards, like DAVIS240a/b/c. Deprecated in favor of CAER_DEVICE_DAVIS.

4.1.2.3 CAER_DEVICE_DAVIS_FX3

```
#define CAER_DEVICE_DAVIS_FX3 2
```

Device type definition for iniVation DAVIS FX3-based boards, like DAVIS346. Deprecated in favor of CAER_DEVICE_DAVIS.

4.1.2.4 CAER_DEVICE_DAVIS_RPI

```
#define CAER_DEVICE_DAVIS_RPI 6
```

Device type definition for iniVation Raspberry Pi-based DAVIS boards.

4.1.2.5 DAVIS128_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS128_CONFIG_BIAS_ADCCOMPBP 19
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.6 DAVIS128_CONFIG_BIAS_ADCCREFHIGH

```
#define DAVIS128_CONFIG_BIAS_ADCCREFHIGH 2
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.7 DAVIS128_CONFIG_BIAS_ADCREFLOW

```
#define DAVIS128_CONFIG_BIAS_ADCREFLOW 3
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.8 DAVIS128_CONFIG_BIAS_AEPDBN

```
#define DAVIS128_CONFIG_BIAS_AEPDBN 23
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.9 DAVIS128_CONFIG_BIAS_AEPUXBP

```
#define DAVIS128_CONFIG_BIAS_AEPUXBP 24
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.10 DAVIS128_CONFIG_BIAS_AEPUYBP

```
#define DAVIS128_CONFIG_BIAS_AEPUYBP 25
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.11 DAVIS128_CONFIG_BIAS_APSCAS

```
#define DAVIS128_CONFIG_BIAS_APSCAS 1
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.12 DAVIS128_CONFIG_BIAS_APSEVERFLOWLEVEL

```
#define DAVIS128_CONFIG_BIAS_APSEVERFLOWLEVEL 0
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.13 DAVIS128_CONFIG_BIAS_APSROSFBN

```
#define DAVIS128_CONFIG_BIAS_APSROSFBN 18
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.14 DAVIS128_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS128_CONFIG_BIAS_BIASBUFFER 34
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.15 DAVIS128_CONFIG_BIAS_COLSELLOWBN

```
#define DAVIS128_CONFIG_BIAS_COLSELLOWBN 20
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.16 DAVIS128_CONFIG_BIAS_DACBUFBP

```
#define DAVIS128_CONFIG_BIAS_DACBUFBP 21
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.17 DAVIS128_CONFIG_BIAS_DIFFBN

```
#define DAVIS128_CONFIG_BIAS_DIFFBN 10
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.18 DAVIS128_CONFIG_BIAS_IFREFRBN

```
#define DAVIS128_CONFIG_BIAS_IFREFRBN 26
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.19 DAVIS128_CONFIG_BIAS_IFTHRBN

```
#define DAVIS128_CONFIG_BIAS_IFTHRBN 27
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.20 DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN 22
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.21 DAVIS128_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS128_CONFIG_BIAS_LOCALBUFBN 8
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.22 DAVIS128_CONFIG_BIAS_OFFBN

```
#define DAVIS128_CONFIG_BIAS_OFFBN 12
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.23 DAVIS128_CONFIG_BIAS_ONBN

```
#define DAVIS128_CONFIG_BIAS_ONBN 11
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.24 DAVIS128_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS128_CONFIG_BIAS_PADFOLLBN 9
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.25 DAVIS128_CONFIG_BIAS_PIXINBN

```
#define DAVIS128_CONFIG_BIAS_PIXINBN 13
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.26 DAVIS128_CONFIG_BIAS_PRBP

```
#define DAVIS128_CONFIG_BIAS_PRBP 14
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.27 DAVIS128_CONFIG_BIAS_PRSFBP

```
#define DAVIS128_CONFIG_BIAS_PRSFBP 15
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.28 DAVIS128_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS128_CONFIG_BIAS_READOUTBUFBP 17
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.29 DAVIS128_CONFIG_BIAS_REFRBP

```
#define DAVIS128_CONFIG_BIAS_REFRBP 16
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.30 DAVIS128_CONFIG_BIAS_SSN

```
#define DAVIS128_CONFIG_BIAS_SSN 36
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.31 DAVIS128_CONFIG_BIAS_SSP

```
#define DAVIS128_CONFIG_BIAS_SSP 35
```

Parameter address for module DAVIS128_CONFIG_BIAS: DAVIS128 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.32 DAVIS128_CONFIG_CHIP_AERNAROW

```
#define DAVIS128_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.33 DAVIS128_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS128_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.34 DAVIS128_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS128_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.35 DAVIS128_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS128_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.36 DAVIS128_CONFIG_CHIP_BIASMUX0

```
#define DAVIS128_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.37 DAVIS128_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS128_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.38 DAVIS128_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS128_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.39 DAVIS128_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS128_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.40 DAVIS128_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS128_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.41 DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.42 DAVIS128_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS128_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.43 DAVIS128_CONFIG_CHIP_RESETESTPIXEL

```
#define DAVIS128_CONFIG_CHIP_RESETESTPIXEL 138
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.44 DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.45 DAVIS128_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS128_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.46 DAVIS128_CONFIG_CHIP_USEAOUT

```
#define DAVIS128_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVIS128_CONFIG_CHIP: DAVIS128 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.47 DAVIS208_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS208_CONFIG_BIAS_ADCCOMPBP 19
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.48 DAVIS208_CONFIG_BIAS_ADCCREFHIGH

```
#define DAVIS208_CONFIG_BIAS_ADCCREFHIGH 2
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.49 DAVIS208_CONFIG_BIAS_ADCCREFLOW

```
#define DAVIS208_CONFIG_BIAS_ADCCREFLOW 3
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.50 DAVIS208_CONFIG_BIAS_AEPDBN

```
#define DAVIS208_CONFIG_BIAS_AEPDBN 23
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.51 DAVIS208_CONFIG_BIAS_AEPUXBP

```
#define DAVIS208_CONFIG_BIAS_AEPUXBP 24
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.52 DAVIS208_CONFIG_BIAS_AEPUYBP

```
#define DAVIS208_CONFIG_BIAS_AEPUYBP 25
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.53 DAVIS208_CONFIG_BIAS_APSCAS

```
#define DAVIS208_CONFIG_BIAS_APSCAS 1
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.54 DAVIS208_CONFIG_BIAS_APSOVERFLOWLEVEL

```
#define DAVIS208_CONFIG_BIAS_APSOVERFLOWLEVEL 0
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.55 DAVIS208_CONFIG_BIAS_APSROSFBN

```
#define DAVIS208_CONFIG_BIAS_APSROSFBN 18
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.56 DAVIS208_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS208_CONFIG_BIAS_BIASBUFFER 34
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.57 DAVIS208_CONFIG_BIAS_COLSELLOWBN

```
#define DAVIS208_CONFIG_BIAS_COLSELLOWBN 20
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.58 DAVIS208_CONFIG_BIAS_DACBUFBP

```
#define DAVIS208_CONFIG_BIAS_DACBUFBP 21
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.59 DAVIS208_CONFIG_BIAS_DIFFBN

```
#define DAVIS208_CONFIG_BIAS_DIFFBN 10
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.60 DAVIS208_CONFIG_BIAS_IFREFRBN

```
#define DAVIS208_CONFIG_BIAS_IFREFRBN 26
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.61 DAVIS208_CONFIG_BIAS_IFTHRBN

```
#define DAVIS208_CONFIG_BIAS_IFTHRBN 27
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.62 DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN 22
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.63 DAVIS208_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS208_CONFIG_BIAS_LOCALBUFBN 8
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.64 DAVIS208_CONFIG_BIAS_OFFBN

```
#define DAVIS208_CONFIG_BIAS_OFFBN 12
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.65 DAVIS208_CONFIG_BIAS_ONBN

```
#define DAVIS208_CONFIG_BIAS_ONBN 11
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.66 DAVIS208_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS208_CONFIG_BIAS_PADFOLLBN 9
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.67 DAVIS208_CONFIG_BIAS_PIXINBN

```
#define DAVIS208_CONFIG_BIAS_PIXINBN 13
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.68 DAVIS208_CONFIG_BIAS_PRBP

```
#define DAVIS208_CONFIG_BIAS_PRBP 14
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.69 DAVIS208_CONFIG_BIAS_PRSFBP

```
#define DAVIS208_CONFIG_BIAS_PRSFBP 15
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.70 DAVIS208_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS208_CONFIG_BIAS_READOUTBUFBP 17
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.71 DAVIS208_CONFIG_BIAS_REFRBP

```
#define DAVIS208_CONFIG_BIAS_REFRBP 16
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.72 DAVIS208_CONFIG_BIAS_REFSS

```
#define DAVIS208_CONFIG_BIAS_REFSS 7
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.73 DAVIS208_CONFIG_BIAS_REFSSBN

```
#define DAVIS208_CONFIG_BIAS_REFSSBN 30
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.74 DAVIS208_CONFIG_BIAS_REGBIASBP

```
#define DAVIS208_CONFIG_BIAS_REGBIASBP 28
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.75 DAVIS208_CONFIG_BIAS_RESETHIGHPASS

```
#define DAVIS208_CONFIG_BIAS_RESETHIGHPASS 6
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.76 DAVIS208_CONFIG_BIAS_SSN

```
#define DAVIS208_CONFIG_BIAS_SSN 36
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.77 DAVIS208_CONFIG_BIAS_SSP

```
#define DAVIS208_CONFIG_BIAS_SSP 35
```

Parameter address for module DAVIS208_CONFIG_BIAS: DAVIS208 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.78 DAVIS208_CONFIG_CHIP_AERNAROW

```
#define DAVIS208_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.79 DAVIS208_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS208_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.80 DAVIS208_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS208_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.81 DAVIS208_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS208_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.82 DAVIS208_CONFIG_CHIP_BIASMUX0

```
#define DAVIS208_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.83 DAVIS208_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS208_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.84 DAVIS208_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS208_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.85 DAVIS208_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS208_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.86 DAVIS208_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS208_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.87 DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.88 DAVIS208_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS208_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.89 DAVIS208_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVIS208_CONFIG_CHIP_RESETTESTPIXEL 138
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.90 DAVIS208_CONFIG_CHIP_SELECTBIASREFSS

```
#define DAVIS208_CONFIG_CHIP_SELECTBIASREFSS 146
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.91 DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.92 DAVIS208_CONFIG_CHIP_SELECTHIGHPASS

```
#define DAVIS208_CONFIG_CHIP_SELECTHIGHPASS 149
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.93 DAVIS208_CONFIG_CHIP_SELECTPOSB

```
#define DAVIS208_CONFIG_CHIP_SELECTPOSB 148
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.94 DAVIS208_CONFIG_CHIP_SELECTPREMPAVG

```
#define DAVIS208_CONFIG_CHIP_SELECTPREMPAVG 145
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.95 DAVIS208_CONFIG_CHIP_SELECTSENSE

```
#define DAVIS208_CONFIG_CHIP_SELECTSENSE 147
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.96 DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.97 DAVIS208_CONFIG_CHIP_USEAOUT

```
#define DAVIS208_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVIS208_CONFIG_CHIP: DAVIS208 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.98 DAVIS240_CONFIG_BIAS_AEPDBN

```
#define DAVIS240_CONFIG_BIAS_AEPDBN 11
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.99 DAVIS240_CONFIG_BIAS_AEPUXBP

```
#define DAVIS240_CONFIG_BIAS_AEPUXBP 13
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.100 DAVIS240_CONFIG_BIAS_AEPUYBP

```
#define DAVIS240_CONFIG_BIAS_AEPUYBP 14
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.101 DAVIS240_CONFIG_BIAS_APSCASEPC

```
#define DAVIS240_CONFIG_BIAS_APSCASEPC 3
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.102 DAVIS240_CONFIG_BIAS_APSOEVERFLOWLEVELBN

```
#define DAVIS240_CONFIG_BIAS_APSOEVERFLOWLEVELBN 18
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.103 DAVIS240_CONFIG_BIAS_APSROSFBN

```
#define DAVIS240_CONFIG_BIAS_APSROSFBN 5
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.104 DAVIS240_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS240_CONFIG_BIAS_BIASBUFFER 19
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.105 DAVIS240_CONFIG_BIAS_DIFFBN

```
#define DAVIS240_CONFIG_BIAS_DIFFBN 0
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.106 DAVIS240_CONFIG_BIAS_DIFFCASBNC

```
#define DAVIS240_CONFIG_BIAS_DIFFCASBNC 4
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.107 DAVIS240_CONFIG_BIAS_IFREFRBN

```
#define DAVIS240_CONFIG_BIAS_IFREFRBN 16
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.108 DAVIS240_CONFIG_BIAS_IFTHRBN

```
#define DAVIS240_CONFIG_BIAS_IFTHRBN 15
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.109 DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN 12
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.110 DAVIS240_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS240_CONFIG_BIAS_LOCALBUFBN 6
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.111 DAVIS240_CONFIG_BIAS_OFFBN

```
#define DAVIS240_CONFIG_BIAS_OFFBN 2
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.112 DAVIS240_CONFIG_BIAS_ONBN

```
#define DAVIS240_CONFIG_BIAS_ONBN 1
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.113 DAVIS240_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS240_CONFIG_BIAS_PADFOLLBN 17
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.114 DAVIS240_CONFIG_BIAS_PIXINBN

```
#define DAVIS240_CONFIG_BIAS_PIXINBN 7
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.115 DAVIS240_CONFIG_BIAS_PRBP

```
#define DAVIS240_CONFIG_BIAS_PRBP 8
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.116 DAVIS240_CONFIG_BIAS_PRSFBP

```
#define DAVIS240_CONFIG_BIAS_PRSFBP 9
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.117 DAVIS240_CONFIG_BIAS_REFRBP

```
#define DAVIS240_CONFIG_BIAS_REFRBP 10
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.118 DAVIS240_CONFIG_BIAS_SSN

```
#define DAVIS240_CONFIG_BIAS_SSN 21
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.119 DAVIS240_CONFIG_BIAS_SSP

```
#define DAVIS240_CONFIG_BIAS_SSP 20
```

Parameter address for module DAVIS240_CONFIG_BIAS: DAVIS240chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.120 DAVIS240_CONFIG_CHIP_AERNAROW

```
#define DAVIS240_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.121 DAVIS240_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS240_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.122 DAVIS240_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS240_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.123 DAVIS240_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS240_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.124 DAVIS240_CONFIG_CHIP_BIASMUX0

```
#define DAVIS240_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.125 DAVIS240_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS240_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.126 DAVIS240_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS240_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.127 DAVIS240_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS240_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.128 DAVIS240_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS240_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.129 DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.130 DAVIS240_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS240_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.131 DAVIS240_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVIS240_CONFIG_CHIP_RESETTESTPIXEL 138
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.132 DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL

```
#define DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL 139
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.133 DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.134 DAVIS240_CONFIG_CHIP_USEAOUT

```
#define DAVIS240_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVIS240_CONFIG_CHIP: DAVIS240 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead. On DAVIS240B cameras, DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL can be used to enable the test pixel array.

4.1.2.135 DAVIS346_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS346_CONFIG_BIAS_ADCCOMPBP 19
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.136 DAVIS346_CONFIG_BIAS_ADCCREFHIGH

```
#define DAVIS346_CONFIG_BIAS_ADCCREFHIGH 2
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.137 DAVIS346_CONFIG_BIAS_ADCREFLOW

```
#define DAVIS346_CONFIG_BIAS_ADCREFLOW 3
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.138 DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE

```
#define DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE 4
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.139 DAVIS346_CONFIG_BIAS_AEPDBN

```
#define DAVIS346_CONFIG_BIAS_AEPDBN 23
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.140 DAVIS346_CONFIG_BIAS_AEPUXBP

```
#define DAVIS346_CONFIG_BIAS_AEPUXBP 24
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.141 DAVIS346_CONFIG_BIAS_AEPUYBP

```
#define DAVIS346_CONFIG_BIAS_AEPUYBP 25
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.142 DAVIS346_CONFIG_BIAS_APSCAS

```
#define DAVIS346_CONFIG_BIAS_APSCAS 1
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.143 DAVIS346_CONFIG_BIAS_APSOVERFLOWLEVEL

```
#define DAVIS346_CONFIG_BIAS_APSOVERFLOWLEVEL 0
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.144 DAVIS346_CONFIG_BIAS_APSROSFBN

```
#define DAVIS346_CONFIG_BIAS_APSROSFBN 18
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.145 DAVIS346_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS346_CONFIG_BIAS_BIASBUFFER 34
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.146 DAVIS346_CONFIG_BIAS_COLSELLOWBN

```
#define DAVIS346_CONFIG_BIAS_COLSELLOWBN 20
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.147 DAVIS346_CONFIG_BIAS_DACBUFBP

```
#define DAVIS346_CONFIG_BIAS_DACBUFBP 21
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.148 DAVIS346_CONFIG_BIAS_DIFFBN

```
#define DAVIS346_CONFIG_BIAS_DIFFBN 10
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.149 DAVIS346_CONFIG_BIAS_IFREFRBN

```
#define DAVIS346_CONFIG_BIAS_IFREFRBN 26
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.150 DAVIS346_CONFIG_BIAS_IFTHRBN

```
#define DAVIS346_CONFIG_BIAS_IFTHRBN 27
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.151 DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN 22
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.152 DAVIS346_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS346_CONFIG_BIAS_LOCALBUFBN 8
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.153 DAVIS346_CONFIG_BIAS_OFFBN

```
#define DAVIS346_CONFIG_BIAS_OFFBN 12
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.154 DAVIS346_CONFIG_BIAS_ONBN

```
#define DAVIS346_CONFIG_BIAS_ONBN 11
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.155 DAVIS346_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS346_CONFIG_BIAS_PADFOLLBN 9
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.156 DAVIS346_CONFIG_BIAS_PIXINBN

```
#define DAVIS346_CONFIG_BIAS_PIXINBN 13
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.157 DAVIS346_CONFIG_BIAS_PRBP

```
#define DAVIS346_CONFIG_BIAS_PRBP 14
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.158 DAVIS346_CONFIG_BIAS_PRSFBP

```
#define DAVIS346_CONFIG_BIAS_PRSFBP 15
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.159 DAVIS346_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS346_CONFIG_BIAS_READOUTBUFBP 17
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.160 DAVIS346_CONFIG_BIAS_REFRBP

```
#define DAVIS346_CONFIG_BIAS_REFRBP 16
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.161 DAVIS346_CONFIG_BIAS_SSN

```
#define DAVIS346_CONFIG_BIAS_SSN 36
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.162 DAVIS346_CONFIG_BIAS_SSP

```
#define DAVIS346_CONFIG_BIAS_SSP 35
```

Parameter address for module DAVIS346_CONFIG_BIAS: DAVIS346 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.163 DAVIS346_CONFIG_CHIP_AERNAROW

```
#define DAVIS346_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.164 DAVIS346_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS346_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.165 DAVIS346_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS346_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.166 DAVIS346_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS346_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.167 DAVIS346_CONFIG_CHIP_BIASMUX0

```
#define DAVIS346_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.168 DAVIS346_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS346_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.169 DAVIS346_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS346_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.170 DAVIS346_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS346_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.171 DAVIS346_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS346_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.172 DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.173 DAVIS346_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS346_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.174 DAVIS346_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVIS346_CONFIG_CHIP_RESETTESTPIXEL 138
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.175 DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.176 DAVIS346_CONFIG_CHIP_TESTADC

```
#define DAVIS346_CONFIG_CHIP_TESTADC 144
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.177 DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.178 DAVIS346_CONFIG_CHIP_USEAOUT

```
#define DAVIS346_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVIS346_CONFIG_CHIP: DAVIS346 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.179 DAVIS640_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS640_CONFIG_BIAS_ADCCOMPBP 19
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.180 DAVIS640_CONFIG_BIAS_ADCREFHIGH

```
#define DAVIS640_CONFIG_BIAS_ADCREFHIGH 2
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.181 DAVIS640_CONFIG_BIAS_ADCREFLOW

```
#define DAVIS640_CONFIG_BIAS_ADCREFLOW 3
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.182 DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE

```
#define DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE 4
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.183 DAVIS640_CONFIG_BIAS_AEPDBN

```
#define DAVIS640_CONFIG_BIAS_AEPDBN 23
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.184 DAVIS640_CONFIG_BIAS_AEPUXBP

```
#define DAVIS640_CONFIG_BIAS_AEPUXBP 24
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.185 DAVIS640_CONFIG_BIAS_AEPUYBP

```
#define DAVIS640_CONFIG_BIAS_AEPUYBP 25
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.186 DAVIS640_CONFIG_BIAS_APSCAS

```
#define DAVIS640_CONFIG_BIAS_APSCAS 1
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.187 DAVIS640_CONFIG_BIAS_APSOEVERFLOWLEVEL

```
#define DAVIS640_CONFIG_BIAS_APSOEVERFLOWLEVEL 0
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.188 DAVIS640_CONFIG_BIAS_APSROSFBN

```
#define DAVIS640_CONFIG_BIAS_APSROSFBN 18
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.189 DAVIS640_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS640_CONFIG_BIAS_BIASBUFFER 34
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.190 DAVIS640_CONFIG_BIAS_COLSELLOWBN

```
#define DAVIS640_CONFIG_BIAS_COLSELLOWBN 20
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.191 DAVIS640_CONFIG_BIAS_DACBUFBP

```
#define DAVIS640_CONFIG_BIAS_DACBUFBP 21
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.192 DAVIS640_CONFIG_BIAS_DIFFBN

```
#define DAVIS640_CONFIG_BIAS_DIFFBN 10
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.193 DAVIS640_CONFIG_BIAS_IFREFRBN

```
#define DAVIS640_CONFIG_BIAS_IFREFRBN 26
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.194 DAVIS640_CONFIG_BIAS_IFTHRBN

```
#define DAVIS640_CONFIG_BIAS_IFTHRBN 27
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.195 DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN 22
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.196 DAVIS640_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS640_CONFIG_BIAS_LOCALBUFBN 8
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.197 DAVIS640_CONFIG_BIAS_OFFBN

```
#define DAVIS640_CONFIG_BIAS_OFFBN 12
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.198 DAVIS640_CONFIG_BIAS_ONBN

```
#define DAVIS640_CONFIG_BIAS_ONBN 11
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.199 DAVIS640_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS640_CONFIG_BIAS_PADFOLLBN 9
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.200 DAVIS640_CONFIG_BIAS_PIXINBN

```
#define DAVIS640_CONFIG_BIAS_PIXINBN 13
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.201 DAVIS640_CONFIG_BIAS_PRBP

```
#define DAVIS640_CONFIG_BIAS_PRBP 14
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.202 DAVIS640_CONFIG_BIAS_PRSFBP

```
#define DAVIS640_CONFIG_BIAS_PRSFBP 15
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.203 DAVIS640_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS640_CONFIG_BIAS_READOUTBUFBP 17
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.204 DAVIS640_CONFIG_BIAS_REFRBP

```
#define DAVIS640_CONFIG_BIAS_REFRBP 16
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.205 DAVIS640_CONFIG_BIAS_SSN

```
#define DAVIS640_CONFIG_BIAS_SSN 36
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.206 DAVIS640_CONFIG_BIAS_SSP

```
#define DAVIS640_CONFIG_BIAS_SSP 35
```

Parameter address for module DAVIS640_CONFIG_BIAS: DAVIS640 chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.207 DAVIS640_CONFIG_CHIP_AERNAROW

```
#define DAVIS640_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.208 DAVIS640_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS640_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.209 DAVIS640_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS640_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.210 DAVIS640_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS640_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.211 DAVIS640_CONFIG_CHIP_BIASMUX0

```
#define DAVIS640_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.212 DAVIS640_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS640_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.213 DAVIS640_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS640_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.214 DAVIS640_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS640_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.215 DAVIS640_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS640_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.216 DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.217 DAVIS640_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS640_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.218 DAVIS640_CONFIG_CHIP_RESETESTPIXEL

```
#define DAVIS640_CONFIG_CHIP_RESETESTPIXEL 138
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.219 DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.220 DAVIS640_CONFIG_CHIP_TESTADC

```
#define DAVIS640_CONFIG_CHIP_TESTADC 144
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.221 DAVIS640_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS640_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.222 DAVIS640_CONFIG_CHIP_USEAOUT

```
#define DAVIS640_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVIS640_CONFIG_CHIP: DAVIS640 chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.223 DAVIS_APS_ROI_REGIONS_MAX

```
#define DAVIS_APS_ROI_REGIONS_MAX 4
```

DAVIS: maximum number of APS Regions-of-Interest that can be produced by any DAVIS camera. See 'struct [caer_davis_info](#)' for actual number: 4 if apsHasQuadROI=true, else 1.

4.1.2.224 DAVIS_CHIP_DAVIS128

```
#define DAVIS_CHIP_DAVIS128 3
```

DAVIS128 chip identifier. 128x128, color possible, internal ADC.

4.1.2.225 DAVIS_CHIP_DAVIS208

```
#define DAVIS_CHIP_DAVIS208 8
```

DAVIS208 chip identifier. 208x192, special sensitive test pixels, color possible, internal ADC.

4.1.2.226 DAVIS_CHIP_DAVIS240A

```
#define DAVIS_CHIP_DAVIS240A 0
```

DAVIS240A chip identifier. 240x180, no color, no global shutter.

4.1.2.227 DAVIS_CHIP_DAVIS240B

```
#define DAVIS_CHIP_DAVIS240B 1
```

DAVIS240B chip identifier. 240x180, no color, 50 test columns left-side.

4.1.2.228 DAVIS_CHIP_DAVIS240C

```
#define DAVIS_CHIP_DAVIS240C 2
```

DAVIS240C chip identifier. 240x180, no color.

4.1.2.229 DAVIS_CHIP_DAVIS346A

```
#define DAVIS_CHIP_DAVIS346A 4
```

DAVIS346A chip identifier. 346x260, color possible, internal ADC.

4.1.2.230 DAVIS_CHIP_DAVIS346B

```
#define DAVIS_CHIP_DAVIS346B 5
```

DAVIS346B chip identifier. 346x260, color possible, internal ADC.

4.1.2.231 DAVIS_CHIP_DAVIS346C

```
#define DAVIS_CHIP_DAVIS346C 9
```

DAVIS346C chip identifier. 346x260, BSI, color possible, internal ADC.

4.1.2.232 DAVIS_CHIP_DAVIS640

```
#define DAVIS_CHIP_DAVIS640 6
```

DAVIS640 chip identifier. 640x480, color possible, internal ADC.

4.1.2.233 DAVIS_CHIP_DAVISRGB

```
#define DAVIS_CHIP_DAVISRGB 7
```

DAVISRGB chip identifier. 640x480 APS, 320x240 DVS, color possible, internal ADC.

4.1.2.234 DAVIS_CONFIG_APS

```
#define DAVIS_CONFIG_APS 2
```

Module address: device-side APS (Frame) configuration. The APS (Active-Pixel-Sensor) is responsible for getting the normal, synchronous frame from the camera chip. It supports various options for very precise timing control, as well as Region of Interest imaging.

4.1.2.235 DAVIS_CONFIG_APS_ADC_TEST_MODE

```
#define DAVIS_CONFIG_APS_ADC_TEST_MODE 39
```

Parameter address for module DAVIS_CONFIG_APS: put all APS pixels into reset, while keeping everything else running. This is only useful for testing and characterizing the internal ADC, to minimize noise.

4.1.2.236 DAVIS_CONFIG_APS_AUTOEXPOSURE

```
#define DAVIS_CONFIG_APS_AUTOEXPOSURE 81
```

Parameter address for module DAVIS_CONFIG_APS: automatic exposure control, tries to set the exposure value automatically to an appropriate value to maximize information in the scene and minimize under- and over-exposure.

4.1.2.237 DAVIS_CONFIG_APS_COLOR_FILTER

```
#define DAVIS_CONFIG_APS_COLOR_FILTER 3
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, contains information on the type of color filter present on the device. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get proper color filter information.

4.1.2.238 DAVIS_CONFIG_APS_COLUMN_SETTLE

```
#define DAVIS_CONFIG_APS_COLUMN_SETTLE 16
```

Parameter address for module DAVIS_CONFIG_APS: column settle time in ADCClock cycles.

4.1.2.239 DAVIS_CONFIG_APS_END_COLUMN_0

```
#define DAVIS_CONFIG_APS_END_COLUMN_0 11
```

Parameter address for module DAVIS_CONFIG_APS: end position on the X axis for Region of Interest 0. Must be between 0 and APS_SIZE_X-1, and be greater or equal to DAVIS_CONFIG_APS_START_COLUMN_0.

4.1.2.240 DAVIS_CONFIG_APS_END_COLUMN_1

```
#define DAVIS_CONFIG_APS_END_COLUMN_1 22
```

Parameter address for module DAVIS_CONFIG_APS: end position on the X axis for Region of Interest 1. Must be between 0 and APS_SIZE_X-1, and be greater or equal to DAVIS_CONFIG_APS_START_COLUMN_1.

4.1.2.241 DAVIS_CONFIG_APS_END_COLUMN_2

```
#define DAVIS_CONFIG_APS_END_COLUMN_2 26
```

Parameter address for module DAVIS_CONFIG_APS: end position on the X axis for Region of Interest 2. Must be between 0 and APS_SIZE_X-1, and be greater or equal to DAVIS_CONFIG_APS_START_COLUMN_2.

4.1.2.242 DAVIS_CONFIG_APS_END_COLUMN_3

```
#define DAVIS_CONFIG_APS_END_COLUMN_3 30
```

Parameter address for module DAVIS_CONFIG_APS: end position on the X axis for Region of Interest 3. Must be between 0 and APS_SIZE_X-1, and be greater or equal to DAVIS_CONFIG_APS_START_COLUMN_3.

4.1.2.243 DAVIS_CONFIG_APS_END_ROW_0

```
#define DAVIS_CONFIG_APS_END_ROW_0 12
```

Parameter address for module DAVIS_CONFIG_APS: end position on the Y axis for Region of Interest 0. Must be between 0 and APS_SIZE_Y-1, and be greater or equal to DAVIS_CONFIG_APS_START_ROW_0.

4.1.2.244 DAVIS_CONFIG_APS_END_ROW_1

```
#define DAVIS_CONFIG_APS_END_ROW_1 23
```

Parameter address for module DAVIS_CONFIG_APS: end position on the Y axis for Region of Interest 1. Must be between 0 and APS_SIZE_Y-1, and be greater or equal to DAVIS_CONFIG_APS_START_ROW_1.

4.1.2.245 DAVIS_CONFIG_APS_END_ROW_2

```
#define DAVIS_CONFIG_APS_END_ROW_2 27
```

Parameter address for module DAVIS_CONFIG_APS: end position on the Y axis for Region of Interest 2. Must be between 0 and APS_SIZE_Y-1, and be greater or equal to DAVIS_CONFIG_APS_START_ROW_2.

4.1.2.246 DAVIS_CONFIG_APS_END_ROW_3

```
#define DAVIS_CONFIG_APS_END_ROW_3 31
```

Parameter address for module DAVIS_CONFIG_APS: end position on the Y axis for Region of Interest 3. Must be between 0 and APS_SIZE_Y-1, and be greater or equal to DAVIS_CONFIG_APS_START_ROW_3.

4.1.2.247 DAVIS_CONFIG_APS_EXPOSURE

```
#define DAVIS_CONFIG_APS_EXPOSURE 13
```

Parameter address for module DAVIS_CONFIG_APS: frame exposure time in microseconds, up to about one second maximum. Very precise for Global Shutter, slightly less exact for Rolling Shutter due to column-based timing constraints.

4.1.2.248 DAVIS_CONFIG_APS_FRAME_DELAY

```
#define DAVIS_CONFIG_APS_FRAME_DELAY 14
```

Parameter address for module DAVIS_CONFIG_APS: delay between consecutive frames in microseconds, up to about one second maximum. This can be used to achieve slower frame-rates, down to about 1 Hertz.

4.1.2.249 DAVIS_CONFIG_APS_GLOBAL_SHUTTER

```
#define DAVIS_CONFIG_APS_GLOBAL_SHUTTER 8
```

Parameter address for module DAVIS_CONFIG_APS: enable Global Shutter mode instead of Rolling Shutter. The Global Shutter eliminates motion artifacts, but is noisier than the Rolling Shutter (worse quality).

4.1.2.250 DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER

```
#define DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER 7
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, information about the presence of the global shutter feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.251 DAVIS_CONFIG_APS_HAS_INTERNAL_ADC

```
#define DAVIS_CONFIG_APS_HAS_INTERNAL_ADC 33
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, information about the presence of an internal, on-chip ADC to read the pixel values. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.252 DAVIS_CONFIG_APS_HAS_QUAD_ROI

```
#define DAVIS_CONFIG_APS_HAS_QUAD_ROI 19
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, information about the presence of the Quadruple Region of Interest feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.253 DAVIS_CONFIG_APS_NULL_SETTLE

```
#define DAVIS_CONFIG_APS_NULL_SETTLE 18
```

Parameter address for module DAVIS_CONFIG_APS: null (between states) settle time in ADCClock cycles.

4.1.2.254 DAVIS_CONFIG_APS_ORIENTATION_INFO

```
#define DAVIS_CONFIG_APS_ORIENTATION_INFO 2
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, contains information on the orientation of the X/Y axes, whether they should be inverted or not on the host when parsing incoming pixels, as well as if the X or Y axes need to be flipped when reading the pixels. Bit 2: apsInvertXY Bit 1: apsFlipX Bit 0: apsFlipY This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get proper size information that already considers the rotation and orientation settings.

4.1.2.255 DAVIS_CONFIG_APS_RAMP_RESET

```
#define DAVIS_CONFIG_APS_RAMP_RESET 37
```

Parameter address for module DAVIS_CONFIG_APS: ramp reset time in ADCClock cycles.

4.1.2.256 DAVIS_CONFIG_APS_RAMP_SHORT_RESET

```
#define DAVIS_CONFIG_APS_RAMP_SHORT_RESET 38
```

Parameter address for module DAVIS_CONFIG_APS: only perform a short ramp (half length) during reset reads, given that the voltage should always be close to the top of the range. This increases the frame-rate, but may have impacts on image quality, especially in very bright regions.

4.1.2.257 DAVIS_CONFIG_APS_RESET_READ

```
#define DAVIS_CONFIG_APS_RESET_READ 5
```

Parameter address for module DAVIS_CONFIG_APS: enable the reset read phase in addition to the signal read, to allow for correlated double sampling schemes. This heavily improves image quality and should always be turned on. In special cases, especially when the camera is perfectly stationary, this can be turned off for longer periods of time to achieve a higher frame-rate and significantly faster frame capture.

4.1.2.258 DAVIS_CONFIG_APS_RESET_SETTLE

```
#define DAVIS_CONFIG_APS_RESET_SETTLE 15
```

Parameter address for module DAVIS_CONFIG_APS: column reset settle time in ADCClock cycles.

4.1.2.259 DAVIS_CONFIG_APS_ROI0_ENABLED

```
#define DAVIS_CONFIG_APS_ROI0_ENABLED 40
```

Parameter address for module DAVIS_CONFIG_APS: Enable/disable ROI region 0. ROI region 0 is always present.

4.1.2.260 DAVIS_CONFIG_APS_ROI1_ENABLED

```
#define DAVIS_CONFIG_APS_ROI1_ENABLED 41
```

Parameter address for module DAVIS_CONFIG_APS: Enable/disable ROI region 1. ROI region 1 is only available when `apsHasQuadROI=true`, see 'struct [caer_davis_info](#)' for more information.

4.1.2.261 DAVIS_CONFIG_APS_ROI2_ENABLED

```
#define DAVIS_CONFIG_APS_ROI2_ENABLED 42
```

Parameter address for module DAVIS_CONFIG_APS: Enable/disable ROI region 2. ROI region 2 is only available when `apsHasQuadROI=true`, see 'struct [caer_davis_info](#)' for more information.

4.1.2.262 DAVIS_CONFIG_APS_ROI3_ENABLED

```
#define DAVIS_CONFIG_APS_ROI3_ENABLED 43
```

Parameter address for module DAVIS_CONFIG_APS: Enable/disable ROI region 3. ROI region 3 is only available when `apsHasQuadROI=true`, see 'struct [caer_davis_info](#)' for more information.

4.1.2.263 DAVIS_CONFIG_APS_ROW_SETTLE

```
#define DAVIS_CONFIG_APS_ROW_SETTLE 17
```

Parameter address for module DAVIS_CONFIG_APS: row settle time in ADCClock cycles.

4.1.2.264 DAVIS_CONFIG_APS_RUN

```
#define DAVIS_CONFIG_APS_RUN 4
```

Parameter address for module DAVIS_CONFIG_APS: enable the APS module and take intensity images of the scene. While this parameter is enabled, frames will be taken continuously. To slow down the frame-rate, see DAVIS_CONFIG_APS_FRAME_DELAY. To only take snapshots, see DAVIS_CONFIG_APS_SNAPSHOT.

4.1.2.265 DAVIS_CONFIG_APS_SAMPLE_ENABLE

```
#define DAVIS_CONFIG_APS_SAMPLE_ENABLE 35
```

Parameter address for module DAVIS_CONFIG_APS: enable sampling of pixel voltage by the internal ADC circuitry. Must always be enabled to get proper frame values.

4.1.2.266 DAVIS_CONFIG_APS_SAMPLE_SETTLE

```
#define DAVIS_CONFIG_APS_SAMPLE_SETTLE 36
```

Parameter address for module DAVIS_CONFIG_APS: sample settle time in ADCClock cycles.

4.1.2.267 DAVIS_CONFIG_APS_SIZE_COLUMNS

```
#define DAVIS_CONFIG_APS_SIZE_COLUMNS 0
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, contains the X axis resolution of the APS frames returned by the camera. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get proper size information that already considers the rotation and orientation settings.

4.1.2.268 DAVIS_CONFIG_APS_SIZE_ROWS

```
#define DAVIS_CONFIG_APS_SIZE_ROWS 1
```

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, contains the Y axis resolution of the APS frames returned by the camera. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get proper size information that already considers the rotation and orientation settings.

4.1.2.269 DAVIS_CONFIG_APS_SNAPSHOT

```
#define DAVIS_CONFIG_APS_SNAPSHOT 80
```

Parameter address for module DAVIS_CONFIG_APS: takes a snapshot (one frame), like a photo-camera. More efficient implementation that just toggling the DAVIS_CONFIG_APS_RUN parameter. The APS module should not be running prior to calling this, as it only makes sense if frames are not being generated at the time. Also, DAVIS_CONFIG_APS_FRAME_DELAY should be set to zero if only doing snapshots, to ensure a quicker readiness for the next one, since the delay is always observed after taking a frame.

4.1.2.270 DAVIS_CONFIG_APS_START_COLUMN_0

```
#define DAVIS_CONFIG_APS_START_COLUMN_0 9
```

Parameter address for module DAVIS_CONFIG_APS: start position on the X axis for Region of Interest 0. Must be between 0 and APS_SIZE_X-1, and be smaller or equal to DAVIS_CONFIG_APS_END_COLUMN_0.

4.1.2.271 DAVIS_CONFIG_APS_START_COLUMN_1

```
#define DAVIS_CONFIG_APS_START_COLUMN_1 20
```

Parameter address for module DAVIS_CONFIG_APS: start position on the X axis for Region of Interest 1. Must be between 0 and APS_SIZE_X-1, and be smaller or equal to DAVIS_CONFIG_APS_END_COLUMN_1.

4.1.2.272 DAVIS_CONFIG_APS_START_COLUMN_2

```
#define DAVIS_CONFIG_APS_START_COLUMN_2 24
```

Parameter address for module DAVIS_CONFIG_APS: start position on the X axis for Region of Interest 2. Must be between 0 and APS_SIZE_X-1, and be smaller or equal to DAVIS_CONFIG_APS_END_COLUMN_2.

4.1.2.273 DAVIS_CONFIG_APS_START_COLUMN_3

```
#define DAVIS_CONFIG_APS_START_COLUMN_3 28
```

Parameter address for module DAVIS_CONFIG_APS: start position on the X axis for Region of Interest 3. Must be between 0 and APS_SIZE_X-1, and be smaller or equal to DAVIS_CONFIG_APS_END_COLUMN_3.

4.1.2.274 DAVIS_CONFIG_APS_START_ROW_0

```
#define DAVIS_CONFIG_APS_START_ROW_0 10
```

Parameter address for module DAVIS_CONFIG_APS: start position on the Y axis for Region of Interest 0. Must be between 0 and APS_SIZE_Y-1, and be smaller or equal to DAVIS_CONFIG_APS_END_ROW_0.

4.1.2.275 DAVIS_CONFIG_APS_START_ROW_1

```
#define DAVIS_CONFIG_APS_START_ROW_1 21
```

Parameter address for module DAVIS_CONFIG_APS: start position on the Y axis for Region of Interest 1. Must be between 0 and APS_SIZE_Y-1, and be smaller or equal to DAVIS_CONFIG_APS_END_ROW_1.

4.1.2.276 DAVIS_CONFIG_APS_START_ROW_2

```
#define DAVIS_CONFIG_APS_START_ROW_2 25
```

Parameter address for module DAVIS_CONFIG_APS: start position on the Y axis for Region of Interest 2. Must be between 0 and APS_SIZE_Y-1, and be smaller or equal to DAVIS_CONFIG_APS_END_ROW_2.

4.1.2.277 DAVIS_CONFIG_APS_START_ROW_3

```
#define DAVIS_CONFIG_APS_START_ROW_3 29
```

Parameter address for module DAVIS_CONFIG_APS: start position on the Y axis for Region of Interest 3. Must be between 0 and APS_SIZE_Y-1, and be smaller or equal to DAVIS_CONFIG_APS_END_ROW_3.

4.1.2.278 DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL 6
```

Parameter address for module DAVIS_CONFIG_APS: if the output FIFO for this module is full, stall the APS state machine and wait until it's free again, instead of just dropping the pixels as they are being read out. This guarantees a complete frame readout, at the possible cost of slight timing differences between pixels. If disabled, incomplete frames may be transmitted and will then be dropped on the host, resulting in lower frame-rates, especially during high DVS traffic.

4.1.2.279 DAVIS_CONFIG_BIAS

```
#define DAVIS_CONFIG_BIAS 5
```

Module address: device-side chip bias configuration. Shared with DAVIS_CONFIG_CHIP. This state machine is responsible for configuring the chip's bias generator.

4.1.2.280 DAVIS_CONFIG_CHIP

```
#define DAVIS_CONFIG_CHIP 5
```

Module address: device-side chip control configuration. Shared with DAVIS_CONFIG_BIAS. This state machine is responsible for configuring the chip's internal control shift registers, to set special options.

4.1.2.281 DAVIS_CONFIG_DDRAER

```
#define DAVIS_CONFIG_DDRAER 9
```

Module address: device-side DDR-AER output configuration. The DDR-AER output module forwards the data from the device and the FPGA/CPLD to some external device using a 4-phase handshake with data on both flanks.

4.1.2.282 DAVIS_CONFIG_DDRAER_ACK_DELAY

```
#define DAVIS_CONFIG_DDRAER_ACK_DELAY 2
```

Parameter address for module DAVIS_CONFIG_DDRAER: wait this many cycles after having received the acknowledge.

4.1.2.283 DAVIS_CONFIG_DDRAER_REQ_DELAY

```
#define DAVIS_CONFIG_DDRAER_REQ_DELAY 1
```

Parameter address for module DAVIS_CONFIG_DDRAER: delay the request by this many cycles after having output the data.

4.1.2.284 DAVIS_CONFIG_DDRAER_RUN

```
#define DAVIS_CONFIG_DDRAER_RUN 0
```

Parameter address for module DAVIS_CONFIG_DDRAER: enable the DDR-AER output module, which transfers the data from the FPGA/CPLD to some external device like a Raspberry Pi.

4.1.2.285 DAVIS_CONFIG_DVS

```
#define DAVIS_CONFIG_DVS 1
```

Module address: device-side DVS configuration. The DVS state machine handshakes with the chip's AER bus and gets the polarity events from it. It supports various configurable delays, as well as advanced filtering capabilities on the polarity events.

4.1.2.286 DAVIS_CONFIG_DVS_ACK_DELAY_COLUMN

```
#define DAVIS_CONFIG_DVS_ACK_DELAY_COLUMN 5
```

Parameter address for module DAVIS_CONFIG_DVS: delay capturing the data and acknowledging it on the AER bus for the column events (serial AER protocol) by this many LogicClock cycles.

4.1.2.287 DAVIS_CONFIG_DVS_ACK_DELAY_ROW

```
#define DAVIS_CONFIG_DVS_ACK_DELAY_ROW 4
```

Parameter address for module DAVIS_CONFIG_DVS: delay capturing the data and acknowledging it on the AER bus for the row events (serial AER protocol) by this many LogicClock cycles.

4.1.2.288 DAVIS_CONFIG_DVS_ACK_EXTENSION_COLUMN

```
#define DAVIS_CONFIG_DVS_ACK_EXTENSION_COLUMN 7
```

Parameter address for module DAVIS_CONFIG_DVS: extend the length of the acknowledge on the AER bus for the column events (serial AER protocol) by this many LogicClock cycles.

4.1.2.289 DAVIS_CONFIG_DVS_ACK_EXTENSION_ROW

```
#define DAVIS_CONFIG_DVS_ACK_EXTENSION_ROW 6
```

Parameter address for module DAVIS_CONFIG_DVS: extend the length of the acknowledge on the AER bus for the row events (serial AER protocol) by this many LogicClock cycles.

4.1.2.290 DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL

```
#define DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL 10
```

Parameter address for module DAVIS_CONFIG_DVS: enable external AER control. This ensures the chip and the DVS pixel array are running, but doesn't do the handshake and leaves the ACK pin in high-impedance, to allow for an external system to take over the AER communication with the chip. DAVIS_CONFIG_DVS_RUN has to be turned off for this to work.

4.1.2.291 DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY

```
#define DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY 29
```

Parameter address for module DAVIS_CONFIG_DVS: enable the background-activity filter, which tries to remove events caused by transistor leakage, by rejecting uncorrelated events.

4.1.2.292 DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_TIME

```
#define DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_TIME 30
```

Parameter address for module DAVIS_CONFIG_DVS: specify the time difference constant for the background-activity filter in 250µs units. Events that are correlated within this time-frame are let through, while others are filtered out.

4.1.2.293 DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN 13
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 0, X axis setting.

4.1.2.294 DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW 12
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 0, Y axis setting.

4.1.2.295 DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN 15
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 1, X axis setting.

4.1.2.296 DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW 14
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 1, Y axis setting.

4.1.2.297 DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN 17
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 2, X axis setting.

4.1.2.298 DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW 16
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 2, Y axis setting.

4.1.2.299 DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN 19
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 3, X axis setting.

4.1.2.300 DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW 18
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 3, Y axis setting.

4.1.2.301 DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN 21
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 4, X axis setting.

4.1.2.302 DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW 20
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 4, Y axis setting.

4.1.2.303 DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN 23
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 5, X axis setting.

4.1.2.304 DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW 22
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 5, Y axis setting.

4.1.2.305 DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN 25
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 6, X axis setting.

4.1.2.306 DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW 24
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 6, Y axis setting.

4.1.2.307 DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN 27
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 7, X axis setting.

4.1.2.308 DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW 26
```

Parameter address for module DAVIS_CONFIG_DVS: the pixel filter completely suppresses up to eight pixels in the DVS array, filtering out all events produced by them. This is the pixel 7, Y axis setting.

4.1.2.309 DAVIS_CONFIG_DVS_FILTER_PIXEL_AUTO_TRAIN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_AUTO_TRAIN 80
```

Parameter address for module DAVIS_CONFIG_DVS: automatically discover the eight most active pixels (above ~5KHz) and set up the hardware pixel filter to remove them from the output.

4.1.2.310 DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD

```
#define DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD 33
```

Parameter address for module DAVIS_CONFIG_DVS: enable the refractory period filter, which limits the firing rate of pixels. This is supported together with the background-activity filter.

4.1.2.311 DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD_TIME

```
#define DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD_TIME 34
```

Parameter address for module DAVIS_CONFIG_DVS: specify the time constant for the refractory period filter, in 250µs units. Pixels will be inhibited from generating new events during this time after the last even has fired.

4.1.2.312 DAVIS_CONFIG_DVS_FILTER_ROI_END_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_ROI_END_COLUMN 38
```

Parameter address for module DAVIS_CONFIG_DVS: end position on the X axis for Region of Interest. Must be between 0 and DVS_SIZE_X-1, and be greater or equal to DAVIS_CONFIG_DVS_FILTER_ROI_START_COLUMN.

4.1.2.313 DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW 39
```

Parameter address for module DAVIS_CONFIG_DVS: end position on the Y axis for Region of Interest. Must be between 0 and DVS_SIZE_Y-1, and be greater or equal to DAVIS_CONFIG_DVS_FILTER_ROI_START_ROW.

4.1.2.314 DAVIS_CONFIG_DVS_FILTER_ROI_START_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_ROI_START_COLUMN 36
```

Parameter address for module DAVIS_CONFIG_DVS: start position on the X axis for Region of Interest. Must be between 0 and DVS_SIZE_X-1, and be smaller or equal to DAVIS_CONFIG_DVS_FILTER_ROI_END_COLUMN.

4.1.2.315 DAVIS_CONFIG_DVS_FILTER_ROI_START_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_ROI_START_ROW 37
```

Parameter address for module DAVIS_CONFIG_DVS: start position on the Y axis for Region of Interest. Must be between 0 and DVS_SIZE_Y-1, and be smaller or equal to DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW.

4.1.2.316 DAVIS_CONFIG_DVS_FILTER_ROW_ONLY_EVENTS

```
#define DAVIS_CONFIG_DVS_FILTER_ROW_ONLY_EVENTS 9
```

Parameter address for module DAVIS_CONFIG_DVS: enable row-only event filter, to eliminate spurious row events with no following columns events. This can happen on DAVIS240 chips, or following the various pixel and background-activity filtering stages, which drop column events to achieve their effect. This should always be enabled!

4.1.2.317 DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER

```
#define DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER 28
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, information about the presence of the background-activity filter feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.318 DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER

```
#define DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER 11
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, information about the presence of the pixel filter feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.319 DAVIS_CONFIG_DVS_HAS_ROI_FILTER

```
#define DAVIS_CONFIG_DVS_HAS_ROI_FILTER 35
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, information about the presence of the ROI filter feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.320 DAVIS_CONFIG_DVS_HAS_STATISTICS

```
#define DAVIS_CONFIG_DVS_HAS_STATISTICS 40
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, information about the presence of the statistics feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.321 DAVIS_CONFIG_DVS_HAS_TEST_EVENT_GENERATOR

```
#define DAVIS_CONFIG_DVS_HAS_TEST_EVENT_GENERATOR 31
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, information about the presence of the test event generator feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.322 DAVIS_CONFIG_DVS_ORIENTATION_INFO

```
#define DAVIS_CONFIG_DVS_ORIENTATION_INFO 2
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, contains information on the orientation of the X/Y axes, whether they should be inverted or not on the host when parsing incoming events. Bit 2: dvsInvert↔XY Bit 1: reserved Bit 0: reserved This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get proper size information that already considers the rotation and orientation settings.

4.1.2.323 DAVIS_CONFIG_DVS_RUN

```
#define DAVIS_CONFIG_DVS_RUN 3
```

Parameter address for module DAVIS_CONFIG_DVS: run the DVS state machine and get polarity events from the chip by handshaking with its AER bus.

4.1.2.324 DAVIS_CONFIG_DVS_SIZE_COLUMNS

```
#define DAVIS_CONFIG_DVS_SIZE_COLUMNS 0
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, contains the X axis resolution of the DVS events returned by the camera. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get proper size information that already considers the rotation and orientation settings.

4.1.2.325 DAVIS_CONFIG_DVS_SIZE_ROWS

```
#define DAVIS_CONFIG_DVS_SIZE_ROWS 1
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, contains the Y axis resolution of the DVS events returned by the camera. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get proper size information that already considers the rotation and orientation settings.

4.1.2.326 DAVIS_CONFIG_DVS_STATISTICS_EVENTS_COLUMN

```
#define DAVIS_CONFIG_DVS_STATISTICS_EVENTS_COLUMN 43
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, representing the number of column event transactions completed on the device. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.1.2.327 DAVIS_CONFIG_DVS_STATISTICS_EVENTS_DROPPED

```
#define DAVIS_CONFIG_DVS_STATISTICS_EVENTS_DROPPED 45
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, representing the number of dropped transaction sequences on the device due to full buffers. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.1.2.328 DAVIS_CONFIG_DVS_STATISTICS_EVENTS_ROW

```
#define DAVIS_CONFIG_DVS_STATISTICS_EVENTS_ROW 41
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, representing the number of row event transactions completed on the device. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.1.2.329 DAVIS_CONFIG_DVS_STATISTICS_FILTERED_BACKGROUND_ACTIVITY

```
#define DAVIS_CONFIG_DVS_STATISTICS_FILTERED_BACKGROUND_ACTIVITY 49
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, representing the number of dropped events due to the background-activity filter. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.1.2.330 DAVIS_CONFIG_DVS_STATISTICS_FILTERED_PIXELS

```
#define DAVIS_CONFIG_DVS_STATISTICS_FILTERED_PIXELS 47
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, representing the number of dropped events due to the pixel filter. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.1.2.331 DAVIS_CONFIG_DVS_STATISTICS_FILTERED_REFRACTORY_PERIOD

```
#define DAVIS_CONFIG_DVS_STATISTICS_FILTERED_REFRACTORY_PERIOD 51
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, representing the number of dropped events due to the refractory period filter. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.1.2.332 DAVIS_CONFIG_DVS_TEST_EVENT_GENERATOR_ENABLE

```
#define DAVIS_CONFIG_DVS_TEST_EVENT_GENERATOR_ENABLE 32
```

Parameter address for module DAVIS_CONFIG_DVS: enable the test event generator for debugging purposes. This generates fake events that appear to originate from all rows sequentially, and for each row going through all its columns, first with an ON polarity and then with an OFF polarity. Both DAVIS_CONFIG_DVS_RUN and DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL have to be turned off for this to work.

4.1.2.333 DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL 8
```

Parameter address for module DAVIS_CONFIG_DVS: if the output FIFO for this module is full, stall the AER handshake with the chip and wait until it's free again, instead of just continuing the handshake and dropping the resulting events.

4.1.2.334 DAVIS_CONFIG_EXTINPUT

```
#define DAVIS_CONFIG_EXTINPUT 4
```

Module address: device-side External Input (signal detector/generator) configuration. The External Input module is used to detect external signals on the external input jack and inject an event into the event stream when this happens. It can detect pulses of a specific length or rising and falling edges. On some systems, a signal generator module is also present, which can generate PWM-like pulsed signals with configurable timing.

4.1.2.335 DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES 2
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT_FALLING_EDGE event when a falling edge is detected (transition from high voltage to low).

4.1.2.336 DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES1

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES1 17
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT1_FALLING_EDGE event when a falling edge is detected (transition from high voltage to low).

4.1.2.337 DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES2

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES2 23
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT2_FALLING_EDGE event when a falling edge is detected (transition from high voltage to low).

4.1.2.338 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH 5
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the minimal length that a pulse must have to trigger the sending of a special event. This is measured in cycles at LogicClock frequency (see 'struct [caer_davis_info](#)' for details on how to get the frequency).

4.1.2.339 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH1

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH1 20
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the minimal length that a pulse must have to trigger the sending of a special event. This is measured in cycles at LogicClock frequency (see 'struct [caer_davis_info](#)' for details on how to get the frequency).

4.1.2.340 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH2

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH2 26
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the minimal length that a pulse must have to trigger the sending of a special event. This is measured in cycles at LogicClock frequency (see 'struct [caer_davis_info](#)' for details on how to get the frequency).

4.1.2.341 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY 4
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the polarity the pulse must exhibit to be detected as such. '1' means active high; a pulse will start when the signal goes from low to high and will continue to be seen as the same pulse as long as it stays high. '0' means active low; a pulse will start when the signal goes from high to low and will continue to be seen as the same pulse as long as it stays low.

4.1.2.342 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY1

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY1 19
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the polarity the pulse must exhibit to be detected as such. '1' means active high; a pulse will start when the signal goes from low to high and will continue to be seen as the same pulse as long as it stays high. '0' means active low; a pulse will start when the signal goes from high to low and will continue to be seen as the same pulse as long as it stays low.

4.1.2.343 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY2

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY2 25
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the polarity the pulse must exhibit to be detected as such. '1' means active high; a pulse will start when the signal goes from low to high and will continue to be seen as the same pulse as long as it stays high. '0' means active low; a pulse will start when the signal goes from high to low and will continue to be seen as the same pulse as long as it stays low.

4.1.2.344 DAVIS_CONFIG_EXTINPUT_DETECT_PULSES

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSES 3
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT_PULSE event when a pulse, of a specified, configurable polarity and length, is detected. See DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY and DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH for more details.

4.1.2.345 DAVIS_CONFIG_EXTINPUT_DETECT_PULSES1

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSES1 18
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT1_PULSE event when a pulse, of a specified, configurable polarity and length, is detected. See DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY1 and DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH1 for more details.

4.1.2.346 DAVIS_CONFIG_EXTINPUT_DETECT_PULSES2

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSES2 24
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT2_PULSE event when a pulse, of a specified, configurable polarity and length, is detected. See DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY2 and DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH2 for more details.

4.1.2.347 DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES 1
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT_RISING_EDGE event when a rising edge is detected (transition from low voltage to high).

4.1.2.348 DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES1

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES1 16
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT1_RISING_EDGE event when a rising edge is detected (transition from low voltage to high).

4.1.2.349 DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES2

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES2 22
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: send a special EXTERNAL_INPUT2_RISING_EDGE event when a rising edge is detected (transition from low voltage to high).

4.1.2.350 DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_FALLING_EDGE

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_FALLING_EDGE 13
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: enables event injection when a falling edge occurs in the generated signal; a special event EXTERNAL_GENERATOR_FALLING_EDGE is emitted into the event stream.

4.1.2.351 DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_RISING_EDGE

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_RISING_EDGE 12
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: enables event injection when a rising edge occurs in the generated signal; a special event EXTERNAL_GENERATOR_RISING_EDGE is emitted into the event stream.

4.1.2.352 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL 10
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the interval between the start of two consecutive pulses, expressed in cycles at LogicClock frequency (see 'struct [caer_davis_info](#)' for details on how to get the frequency). This must be bigger or equal to DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH. To generate a signal with 50% duty cycle, this would have to be exactly double of DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH.

4.1.2.353 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH 11
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the length a pulse stays active, expressed in cycles at LogicClock frequency (see 'struct [caer_davis_info](#)' for details on how to get the frequency). This must be smaller or equal to DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL. To generate a signal with 50% duty cycle, this would have to be exactly half of DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL.

4.1.2.354 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY 9
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: polarity of the PWM-like signal to be generated. '1' means active high, '0' means active low.

4.1.2.355 DAVIS_CONFIG_EXTINPUT_GENERATE_USE_CUSTOM_SIGNAL

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_USE_CUSTOM_SIGNAL 8
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: instead of generating a PWM-like signal by using the configured parameters, use a signal on the FPGA/CPLD that's passed as an input to the External Input module. By default this is disabled and tied to ground, but it can be useful for customized logic designs.

4.1.2.356 DAVIS_CONFIG_EXTINPUT_HAS_EXTRA_DETECTORS

```
#define DAVIS_CONFIG_EXTINPUT_HAS_EXTRA_DETECTORS 14
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: read-only parameter, information about the presence of the extra detectors feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.357 DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR

```
#define DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR 6
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: read-only parameter, information about the presence of the signal generator feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.358 DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR

```
#define DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR 0
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: enable the signal detector module. It generates events when it sees certain types of signals, such as edges or pulses of a defined length, on the IN JACK signal. This can be useful to inject events into the event stream in response to external stimuli or controls, such as turning on a LED lamp.

4.1.2.359 DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR1

```
#define DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR1 15
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: enable the signal detector module. It generates events when it sees certain types of signals, such as edges or pulses of a defined length, on the B1P20 input pin. This can be useful to inject events into the event stream in response to external stimuli or controls, such as turning on a LED lamp.

4.1.2.360 DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR2

```
#define DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR2 21
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: enable the signal detector module. It generates events when it sees certain types of signals, such as edges or pulses of a defined length, on the B1P21 input pin. This can be useful to inject events into the event stream in response to external stimuli or controls, such as turning on a LED lamp.

4.1.2.361 DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR

```
#define DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR 7
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: enable the signal generator module. It generates a PWM-like signal based on configurable parameters and outputs it on the OUT JACK signal.

4.1.2.362 DAVIS_CONFIG_IMU

```
#define DAVIS_CONFIG_IMU 3
```

Module address: device-side IMU (Inertial Measurement Unit) configuration. The IMU module connects to the external IMU chip and sends data on the device's movement in space. It can configure various options on the external chip, such as accelerometer range or gyroscope refresh rate.

4.1.2.363 DAVIS_CONFIG_IMU_ACCEL_DLPF

```
#define DAVIS_CONFIG_IMU_ACCEL_DLPF 11
```

Parameter address for module DAVIS_CONFIG_IMU: this configures the digital low-pass filter for the accelerometer on devices using the InvenSense MPU 9250. Valid values are from 0 to 7 and have the following meaning:

0 - Accel: BW=218.1Hz, Delay=1.88ms, FS=1kHz 1 - Accel: BW=218.1Hz, Delay=1.88ms, FS=1kHz 2 - Accel: BW=99Hz, Delay=2.88ms, FS=1kHz 3 - Accel: BW=44.8Hz, Delay=4.88ms, FS=1kHz 4 - Accel: BW=21.2Hz, Delay=8.87ms, FS=1kHz 5 - Accel: BW=10.2Hz, Delay=16.83ms, FS=1kHz 6 - Accel: BW=5.05Hz, Delay=32.48ms, FS=1kHz 7 - Accel: BW=420Hz, Delay=1.38ms, FS=1kHz

4.1.2.364 DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE

```
#define DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE 8
```

Parameter address for module DAVIS_CONFIG_IMU: select the full scale range of the accelerometer outputs. Valid values are: 0 - +- 2 g 1 - +- 4 g 2 - +- 8 g 3 - +- 16 g

4.1.2.365 DAVIS_CONFIG_IMU_ACCEL_STANDBY

```
#define DAVIS_CONFIG_IMU_ACCEL_STANDBY 2
```

Parameter address for module DAVIS_CONFIG_IMU: put the accelerometer sensor in standby, disabling it.

4.1.2.366 DAVIS_CONFIG_IMU_GYRO_DLPF

```
#define DAVIS_CONFIG_IMU_GYRO_DLPF 7
```

Parameter address for module DAVIS_CONFIG_IMU: this configures the digital low-pass filter for both the accelerometer and the gyroscope on InvenSense MPU 6050/6150 IMU devices, or for the gyroscope only on InvenSense MPU 9250. Valid values are from 0 to 7 and have the following meaning:

On InvenSense MPU 6050/6150: 0 - Accel: BW=260Hz, Delay=0ms, FS=1kHz - Gyro: BW=256Hz, Delay=0.98ms, FS=8kHz 1 - Accel: BW=184Hz, Delay=2.0ms, FS=1kHz - Gyro: BW=188Hz, Delay=1.9ms, FS=1kHz 2 - Accel: BW=94Hz, Delay=3.0ms, FS=1kHz - Gyro: BW=98Hz, Delay=2.8ms, FS=1kHz 3 - Accel: BW=44Hz, Delay=4.9ms, FS=1kHz - Gyro: BW=42Hz, Delay=4.8ms, FS=1kHz 4 - Accel: BW=21Hz, Delay=8.5ms, FS=1kHz - Gyro: BW=20Hz, Delay=8.3ms, FS=1kHz 5 - Accel: BW=10Hz, Delay=13.8ms, FS=1kHz - Gyro: BW=10Hz, Delay=13.4ms, FS=1kHz 6 - Accel: BW=5Hz, Delay=19.0ms, FS=1kHz - Gyro: BW=5Hz, Delay=18.6ms, FS=1kHz 7 - Accel: RESERVED, FS=1kHz - Gyro: RESERVED, FS=8kHz

On InvenSense MPU 9250: 0 - Gyro: BW=250Hz, Delay=0.97ms, FS=8kHz 1 - Gyro: BW=184Hz, Delay=2.9ms, FS=1kHz 2 - Gyro: BW=92Hz, Delay=3.9ms, FS=1kHz 3 - Gyro: BW=41Hz, Delay=5.9ms, FS=1kHz 4 - Gyro: BW=20Hz, Delay=9.9ms, FS=1kHz 5 - Gyro: BW=10Hz, Delay=17.85ms, FS=1kHz 6 - Gyro: BW=5Hz, Delay=33.48ms, FS=1kHz 7 - Gyro: BW=3600Hz, Delay=0.17ms, FS=8kHz

4.1.2.367 DAVIS_CONFIG_IMU_GYRO_FULL_SCALE

```
#define DAVIS_CONFIG_IMU_GYRO_FULL_SCALE 9
```

Parameter address for module DAVIS_CONFIG_IMU: select the full scale range of the gyroscope outputs. Valid values are: 0 - +- 250 °/s 1 - +- 500 °/s 2 - +- 1000 °/s 3 - +- 2000 °/s

4.1.2.368 DAVIS_CONFIG_IMU_GYRO_STANDBY

```
#define DAVIS_CONFIG_IMU_GYRO_STANDBY 3
```

Parameter address for module DAVIS_CONFIG_IMU: put the gyroscope sensor in standby, disabling it.

4.1.2.369 DAVIS_CONFIG_IMU_LP_CYCLE

```
#define DAVIS_CONFIG_IMU_LP_CYCLE 4
```

Parameter address for module DAVIS_CONFIG_IMU: put the IMU into Cycle Mode. In Cycle Mode, the device cycles between sleep mode and waking up to take a single sample of data from the accelerometer at a rate determined by DAVIS_CONFIG_IMU_LP_WAKEUP.

4.1.2.370 DAVIS_CONFIG_IMU_LP_WAKEUP

```
#define DAVIS_CONFIG_IMU_LP_WAKEUP 5
```

Parameter address for module DAVIS_CONFIG_IMU: rate at which the IMU takes an accelerometer sample while in Cycle Mode (see DAVIS_CONFIG_IMU_LP_CYCLE). Valid values are: 0 - 1.25 Hz wake-up frequency 1 - 5 Hz wake-up frequency 2 - 20 Hz wake-up frequency 3 - 40 Hz wake-up frequency

4.1.2.371 DAVIS_CONFIG_IMU_ORIENTATION_INFO

```
#define DAVIS_CONFIG_IMU_ORIENTATION_INFO 10
```

Parameter address for module DAVIS_CONFIG_IMU: read-only parameter, contains information on the orientation of the X/Y/Z axes, whether they should be flipped or not on the host when parsing incoming IMU data samples. Bit 2: imuFlipX Bit 1: imuFlipY Bit 0: imuFlipZ This is reserved for internal use and should not be used by anything other than libcaer. Generated IMU events are already properly flipped when returned to the user.

4.1.2.372 DAVIS_CONFIG_IMU_RUN

```
#define DAVIS_CONFIG_IMU_RUN 0
```

Parameter address for module DAVIS_CONFIG_IMU: run the IMU state machine to get information about the movement and position of the device. This takes the IMU chip out of sleep.

4.1.2.373 DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER

```
#define DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER 6
```

Parameter address for module DAVIS_CONFIG_IMU: this specifies the divider from the Gyroscope Output Rate used to generate the Sample Rate for the IMU. Valid values are from 0 to 255. The Sample Rate is generated like this: $\text{Sample Rate} = \text{Gyroscope Output Rate} / (1 + \text{DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER})$ where Gyroscope Output Rate = 8 kHz when DAVIS_CONFIG_IMU_DIGITAL_LOW_PASS_FILTER is disabled (set to 0 or 7), and 1 kHz when enabled. Note: the accelerometer output rate is 1 kHz. This means that for a Sample Rate greater than 1 kHz, the same accelerometer sample may be output multiple times.

4.1.2.374 DAVIS_CONFIG_IMU_TEMP_STANDBY

```
#define DAVIS_CONFIG_IMU_TEMP_STANDBY 1
```

Parameter address for module DAVIS_CONFIG_IMU: put the temperature sensor in standby, disabling it.

4.1.2.375 DAVIS_CONFIG_MUX

```
#define DAVIS_CONFIG_MUX 0
```

Module address: device-side Multiplexer configuration. The Multiplexer is responsible for mixing, timestamping and outputting (via USB) the various event types generated by the device. It is also responsible for timestamp generation and synchronization.

4.1.2.376 DAVIS_CONFIG_MUX_DROP_APS_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_MUX_DROP_APS_ON_TRANSFER_STALL 5
```

Parameter address for module DAVIS_CONFIG_MUX: drop APS events if the USB output FIFO is full, instead of having them pile up at the input FIFOs. This normally should not be enabled to guarantee complete, coherent frame events, though small timing differences may cause a reduction in observed image quality.

4.1.2.377 DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL 4
```

Parameter address for module DAVIS_CONFIG_MUX: drop DVS events if the USB output FIFO is full, instead of having them pile up at the input FIFOs.

4.1.2.378 DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL 7
```

Parameter address for module DAVIS_CONFIG_MUX: drop External Input events if the USB output FIFO is full, instead of having them pile up at the input FIFOs.

4.1.2.379 DAVIS_CONFIG_MUX_DROP_IMU_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_MUX_DROP_IMU_ON_TRANSFER_STALL 6
```

Parameter address for module DAVIS_CONFIG_MUX: drop IMU events if the USB output FIFO is full, instead of having them pile up at the input FIFOs. This normally should not be enabled to guarantee complete, coherent IMU events, and not get incomplete or wrong IMU information.

4.1.2.380 DAVIS_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE

```
#define DAVIS_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE 3
```

Parameter address for module DAVIS_CONFIG_MUX: under normal circumstances, the chip's bias generator is only powered up when either the DVS or the APS state machines are running, to save power. This flag forces the bias generator to be powered up all the time, which may be useful when one wants to shut-down both APS and DVS temporarily, but still have a quick and well-defined resume behavior.

4.1.2.381 DAVIS_CONFIG_MUX_HAS_STATISTICS

```
#define DAVIS_CONFIG_MUX_HAS_STATISTICS 10
```

Parameter address for module DAVIS_CONFIG_MUX: read-only parameter, information about the presence of the statistics feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.382 DAVIS_CONFIG_MUX_RUN

```
#define DAVIS_CONFIG_MUX_RUN 0
```

Parameter address for module DAVIS_CONFIG_MUX: run the Multiplexer state machine, which is responsible for mixing the various event types at the device level, timestamping them and outputting them via USB or other connectors.

4.1.2.383 DAVIS_CONFIG_MUX_STATISTICS_APS_DROPPED

```
#define DAVIS_CONFIG_MUX_STATISTICS_APS_DROPPED 13
```

Parameter address for module DAVIS_CONFIG_MUX: read-only parameter, representing the number of dropped APS events on the device due to full USB buffers. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.1.2.384 DAVIS_CONFIG_MUX_STATISTICS_DVS_DROPPED

```
#define DAVIS_CONFIG_MUX_STATISTICS_DVS_DROPPED 11
```

Parameter address for module DAVIS_CONFIG_MUX: read-only parameter, representing the number of dropped DVS events on the device due to full USB buffers. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.1.2.385 DAVIS_CONFIG_MUX_STATISTICS_EXTINPUT_DROPPED

```
#define DAVIS_CONFIG_MUX_STATISTICS_EXTINPUT_DROPPED 17
```

Parameter address for module DAVIS_CONFIG_MUX: read-only parameter, representing the number of dropped External Input events on the device due to full USB buffers. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.1.2.386 DAVIS_CONFIG_MUX_STATISTICS_IMU_DROPPED

```
#define DAVIS_CONFIG_MUX_STATISTICS_IMU_DROPPED 15
```

Parameter address for module DAVIS_CONFIG_MUX: read-only parameter, representing the number of dropped IMU events on the device due to full USB buffers. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.1.2.387 DAVIS_CONFIG_MUX_TIMESTAMP_RESET

```
#define DAVIS_CONFIG_MUX_TIMESTAMP_RESET 2
```

Parameter address for module DAVIS_CONFIG_MUX: reset the Timestamp Generator to zero. This also sends a reset pulse to all connected slave devices, resetting their timestamp too.

4.1.2.388 DAVIS_CONFIG_MUX_TIMESTAMP_RUN

```
#define DAVIS_CONFIG_MUX_TIMESTAMP_RUN 1
```

Parameter address for module DAVIS_CONFIG_MUX: run the Timestamp Generator inside the Multiplexer state machine, which will provide microsecond accurate timestamps to the events passing through.

4.1.2.389 DAVIS_CONFIG_SYSINFO

```
#define DAVIS_CONFIG_SYSINFO 6
```

Module address: device-side system information. The system information module provides various details on the device, such as currently installed logic revision or clock speeds. All its parameters are read-only. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation for more details on what information is available.

4.1.2.390 DAVIS_CONFIG_SYSINFO_ADC_CLOCK

```
#define DAVIS_CONFIG_SYSINFO_ADC_CLOCK 4
```

Parameter address for module DAVIS_CONFIG_SYSINFO: read-only parameter, the frequency in MHz at which the FPGA/CPLD logic related to APS frame grabbing is running. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.391 DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER

```
#define DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER 1
```

Parameter address for module DAVIS_CONFIG_SYSINFO: read-only parameter, an integer used to identify the different types of sensor chips used on the device. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.392 DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER

```
#define DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER 2
```

Parameter address for module DAVIS_CONFIG_SYSINFO: read-only parameter, whether the device is currently a timestamp master or slave when synchronizing multiple devices together. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.393 DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK

```
#define DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK 3
```

Parameter address for module DAVIS_CONFIG_SYSINFO: read-only parameter, the frequency in MHz at which the main FPGA/CPLD logic is running. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.394 DAVIS_CONFIG_SYSINFO_LOGIC_VERSION

```
#define DAVIS_CONFIG_SYSINFO_LOGIC_VERSION 0
```

Parameter address for module DAVIS_CONFIG_SYSINFO: read-only parameter, the version of the logic currently running on the device's FPGA/CPLD. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_davis_info](#)' documentation to get this information.

4.1.2.395 DAVIS_CONFIG_USB

```
#define DAVIS_CONFIG_USB 9
```

Module address: device-side USB output configuration. The USB output module forwards the data from the device and the FPGA/CPLD to the USB chip, usually a Cypress FX2 or FX3.

4.1.2.396 DAVIS_CONFIG_USB_EARLY_PACKET_DELAY

```
#define DAVIS_CONFIG_USB_EARLY_PACKET_DELAY 1
```

Parameter address for module DAVIS_CONFIG_USB: the time delay after which a packet of data is committed to USB, even if it is not full yet (short USB packet). The value is in 125µs time-slices, corresponding to how USB schedules its operations (a value of 4 for example would mean waiting at most 0.5ms until sending a short USB packet to the host).

4.1.2.397 DAVIS_CONFIG_USB_RUN

```
#define DAVIS_CONFIG_USB_RUN 0
```

Parameter address for module DAVIS_CONFIG_USB: enable the USB FIFO module, which transfers the data from the FPGA/CPLD to the USB chip, to be then sent to the host. Turning this off will suppress any USB data communication!

4.1.2.398 DAVISRGB_CONFIG_APS_GSFDRESET

```
#define DAVISRGB_CONFIG_APS_GSFDRESET 55
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS RGB chip): Global Shutter FD reset time in ADCClock cycles.

4.1.2.399 DAVISRGB_CONFIG_APS_GSPDRESET

```
#define DAVISRGB_CONFIG_APS_GSPDRESET 52
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS RGB chip): Global Shutter PD reset time in ADCClock cycles.

4.1.2.400 DAVISRGB_CONFIG_APS_GSRESETFALL

```
#define DAVISRGB_CONFIG_APS_GSRESETFALL 53
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS RGB chip): Global Shutter Reset Fall time in ADCClock cycles.

4.1.2.401 DAVISRGB_CONFIG_APS_GSTXFALL

```
#define DAVISRGB_CONFIG_APS_GSTXFALL 54
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS RGB chip): Global Shutter Transfer Fall time in ADCClock cycles.

4.1.2.402 DAVISRGB_CONFIG_APS_RSFDSETTLE

```
#define DAVISRGB_CONFIG_APS_RSFDSETTLE 51
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS RGB chip): Rolling Shutter FD settle time in ADCClock cycles.

4.1.2.403 DAVISRGB_CONFIG_APS_TRANSFER

```
#define DAVISRGB_CONFIG_APS_TRANSFER 50
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS RGB chip): charge transfer time in ADCClock cycles.

4.1.2.404 DAVISRGB_CONFIG_BIAS_ADCCOMPBP

```
#define DAVISRGB_CONFIG_BIAS_ADCCOMPBP 27
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.405 DAVISRGB_CONFIG_BIAS_ADCREFHIGH

```
#define DAVISRGB_CONFIG_BIAS_ADCREFHIGH 6
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.406 DAVISRGB_CONFIG_BIAS_ADCREFLOW

```
#define DAVISRGB_CONFIG_BIAS_ADCREFLOW 7
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.407 DAVISRGB_CONFIG_BIAS_ADCTESTVOLTAGE

```
#define DAVISRGB_CONFIG_BIAS_ADCTESTVOLTAGE 5
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.408 DAVISRGB_CONFIG_BIAS_AEPDBN

```
#define DAVISRGB_CONFIG_BIAS_AEPDBN 31
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.409 DAVISRGB_CONFIG_BIAS_AEPUXBP

```
#define DAVISRGB_CONFIG_BIAS_AEPUXBP 32
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.410 DAVISRGB_CONFIG_BIAS_AEPUYBP

```
#define DAVISRGB_CONFIG_BIAS_AEPUYBP 33
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.411 DAVISRGB_CONFIG_BIAS_APSCAS

```
#define DAVISRGB_CONFIG_BIAS_APSCAS 0
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.412 DAVISRGB_CONFIG_BIAS_APSROSFBN

```
#define DAVISRGB_CONFIG_BIAS_APSROSFBN 26
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.413 DAVISRGB_CONFIG_BIAS_ARRAYBIASBUFFERBN

```
#define DAVISRGB_CONFIG_BIAS_ARRAYBIASBUFFERBN 20
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.414 DAVISRGB_CONFIG_BIAS_ARRAYLOGICBUFFERBN

```
#define DAVISRGB_CONFIG_BIAS_ARRAYLOGICBUFFERBN 22
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.415 DAVISRGB_CONFIG_BIAS_BIASBUFFER

```
#define DAVISRGB_CONFIG_BIAS_BIASBUFFER 34
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.416 DAVISRGB_CONFIG_BIAS_DACBUFBP

```
#define DAVISRGB_CONFIG_BIAS_DACBUFBP 28
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.417 DAVISRGB_CONFIG_BIAS_DIFFBN

```
#define DAVISRGB_CONFIG_BIAS_DIFFBN 14
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.418 DAVISRGB_CONFIG_BIAS_FALLTIMEBN

```
#define DAVISRGB_CONFIG_BIAS_FALLTIMEBN 23
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.419 DAVISRGB_CONFIG_BIAS_GND07

```
#define DAVISRGB_CONFIG_BIAS_GND07 4
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.420 DAVISRGB_CONFIG_BIAS_IFREFRBN

```
#define DAVISRGB_CONFIG_BIAS_IFREFRBN 8
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.421 DAVISRGB_CONFIG_BIAS_IFTHRBN

```
#define DAVISRGB_CONFIG_BIAS_IFTHRBN 9
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.422 DAVISRGB_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVISRGB_CONFIG_BIAS_LCOLTIMEOUTBN 30
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.423 DAVISRGB_CONFIG_BIAS_LOCALBUFBN

```
#define DAVISRGB_CONFIG_BIAS_LOCALBUFBN 10
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.424 DAVISRGB_CONFIG_BIAS_OFFBN

```
#define DAVISRGB_CONFIG_BIAS_OFFBN 16
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.425 DAVISRGB_CONFIG_BIAS_ONBN

```
#define DAVISRGB_CONFIG_BIAS_ONBN 15
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.426 DAVISRGB_CONFIG_BIAS_OVG1LO

```
#define DAVISRGB_CONFIG_BIAS_OVG1LO 1
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.427 DAVISRGB_CONFIG_BIAS_OVG2LO

```
#define DAVISRGB_CONFIG_BIAS_OVG2LO 2
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.428 DAVISRGB_CONFIG_BIAS_PADFOLLBN

```
#define DAVISRGB_CONFIG_BIAS_PADFOLLBN 11
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.429 DAVISRGB_CONFIG_BIAS_PIXINBN

```
#define DAVISRGB_CONFIG_BIAS_PIXINBN 13
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.430 DAVISRGB_CONFIG_BIAS_PRBP

```
#define DAVISRGB_CONFIG_BIAS_PRBP 17
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.431 DAVISRGB_CONFIG_BIAS_PRSFBP

```
#define DAVISRGB_CONFIG_BIAS_PRSFBP 18
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.432 DAVISRGB_CONFIG_BIAS_READOUTBUFBP

```
#define DAVISRGB_CONFIG_BIAS_READOUTBUFBP 25
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.433 DAVISRGB_CONFIG_BIAS_REFRBP

```
#define DAVISRGB_CONFIG_BIAS_REFRBP 19
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.434 DAVISRGB_CONFIG_BIAS_RISETIMEBP

```
#define DAVISRGB_CONFIG_BIAS_RISETIMEBP 24
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.435 DAVISRGB_CONFIG_BIAS_SSN

```
#define DAVISRGB_CONFIG_BIAS_SSN 36
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.436 DAVISRGB_CONFIG_BIAS_SSP

```
#define DAVISRGB_CONFIG_BIAS_SSP 35
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/biases>' for more details.

4.1.2.437 DAVISRGB_CONFIG_BIAS_TX2OVG2HI

```
#define DAVISRGB_CONFIG_BIAS_TX2OVG2HI 3
```

Parameter address for module DAVISRGB_CONFIG_BIAS: DAVISRGB chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasVDACGenerate\(\)](#) for VDAC (voltage) biases.
- [caerBiasCoarseFineGenerate\(\)](#) for coarse-fine (current) biases.
- [caerBiasShiftedSourceGenerate\(\)](#) for shifted-source biases. See '<https://inivation.com/support/hardware/bi>' for more details.

4.1.2.438 DAVISRGB_CONFIG_CHIP_ADJUSTOVG1LO

```
#define DAVISRGB_CONFIG_CHIP_ADJUSTOVG1LO 145
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.439 DAVISRGB_CONFIG_CHIP_ADJUSTOVG2LO

```
#define DAVISRGB_CONFIG_CHIP_ADJUSTOVG2LO 146
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.440 DAVISRGB_CONFIG_CHIP_ADJUSTTX2OVG2HI

```
#define DAVISRGB_CONFIG_CHIP_ADJUSTTX2OVG2HI 147
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.441 DAVISRGB_CONFIG_CHIP_AERNAROW

```
#define DAVISRGB_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.442 DAVISRGB_CONFIG_CHIP_ANALOGMUX0

```
#define DAVISRGB_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.443 DAVISRGB_CONFIG_CHIP_ANALOGMUX1

```
#define DAVISRGB_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.444 DAVISRGB_CONFIG_CHIP_ANALOGMUX2

```
#define DAVISRGB_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.445 DAVISRGB_CONFIG_CHIP_BIASMUX0

```
#define DAVISRGB_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.446 DAVISRGB_CONFIG_CHIP_DIGITALMUX0

```
#define DAVISRGB_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.447 DAVISRGB_CONFIG_CHIP_DIGITALMUX1

```
#define DAVISRGB_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.448 DAVISRGB_CONFIG_CHIP_DIGITALMUX2

```
#define DAVISRGB_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.449 DAVISRGB_CONFIG_CHIP_DIGITALMUX3

```
#define DAVISRGB_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.450 DAVISRGB_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVISRGB_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.451 DAVISRGB_CONFIG_CHIP_RESETESTPIXEL

```
#define DAVISRGB_CONFIG_CHIP_RESETESTPIXEL 138
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.452 DAVISRGB_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVISRGB_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.453 DAVISRGB_CONFIG_CHIP_TESTADC

```
#define DAVISRGB_CONFIG_CHIP_TESTADC 144
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.454 DAVISRGB_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVISRGB_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.455 DAVISRGB_CONFIG_CHIP_USEAOUT

```
#define DAVISRGB_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVISRGB_CONFIG_CHIP: DAVISRGB chip configuration. These are for expert control and should never be used or changed unless for advanced debugging purposes. To change the Global Shutter configuration, please use DAVIS_CONFIG_APS_GLOBAL_SHUTTER instead.

4.1.2.456 IS_DAVIS128

```
#define IS_DAVIS128(  
    chipID ) ((chipID) == DAVIS_CHIP_DAVIS128)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.457 IS_DAVIS208

```
#define IS_DAVIS208(  
    chipID ) ((chipID) == DAVIS_CHIP_DAVIS208)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.458 IS_DAVIS240

```
#define IS_DAVIS240(  
    chipID ) (IS_DAVIS240A(chipID) || IS_DAVIS240B(chipID) || IS_DAVIS240C(chipID))
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.459 IS_DAVIS240A

```
#define IS_DAVIS240A(  
    chipID ) ((chipID) == DAVIS_CHIP_DAVIS240A)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.460 IS_DAVIS240B

```
#define IS_DAVIS240B(  
    chipID ) ((chipID) == DAVIS_CHIP_DAVIS240B)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.461 IS_DAVIS240C

```
#define IS_DAVIS240C(  
    chipID ) ((chipID) == DAVIS_CHIP_DAVIS240C)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.462 IS_DAVIS346

```
#define IS_DAVIS346(  
    chipID ) (IS_DAVIS346A(chipID) || IS_DAVIS346B(chipID) || IS_DAVIS346C(chipID))
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.463 IS_DAVIS346A

```
#define IS_DAVIS346A(  
    chipID ) ((chipID) == DAVIS_CHIP_DAVIS346A)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.464 IS_DAVIS346B

```
#define IS_DAVIS346B(  
    chipID ) ((chipID) == DAVIS_CHIP_DAVIS346B)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.465 IS_DAVIS346C

```
#define IS_DAVIS346C(  
    chipID ) ((chipID) == DAVIS_CHIP_DAVIS346C)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.466 IS_DAVIS640

```
#define IS_DAVIS640(  
    chipID ) ((chipID) == DAVIS_CHIP_DAVIS640)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.2.467 IS_DAVISRGB

```
#define IS_DAVISRGB(  
    chipID ) ((chipID) == DAVIS_CHIP_DAVISRGB)
```

Macros to check a chip identifier integer against the known chip types. Returns true if a chip identifier matches, false otherwise.

4.1.3 Enumeration Type Documentation

4.1.3.1 caer_bias_shiftedsource_operating_mode

```
enum caer_bias_shiftedsource_operating_mode
```

Shifted-source bias operating mode.

Enumerator

SHIFTED_SOURCE	Standard mode.
HI_Z	High impedance (driven from outside).
TIED_TO_RAIL	Tied to ground (SSN) or VDD (SSP).

4.1.3.2 caer_bias_shiftedsource_voltage_level

```
enum caer_bias_shiftedsource_voltage_level
```

Shifted-source bias voltage level.

Enumerator

SPLIT_GATE	Standard mode (200-400mV).
SINGLE_DIODE	Higher shifted-source voltage (one cascode).
DOUBLE_DIODE	Even higher shifted-source voltage (two cascodes).

4.1.4 Function Documentation

4.1.4.1 caerBiasCoarseFineGenerate()

```
uint16_t caerBiasCoarseFineGenerate (
    const struct caer_bias_coarsefine coarseFineBias )
```

Transform coarse-fine bias structure into internal integer representation, suited for sending directly to the device via [caerDeviceConfigSet\(\)](#).

Parameters

<i>coarseFineBias</i>	coarse-fine bias structure.
-----------------------	-----------------------------

Returns

internal integer representation for device configuration.

4.1.4.2 caerBiasCoarseFineParse()

```
struct caer_bias_coarsefine caerBiasCoarseFineParse (
    const uint16_t coarseFineBias )
```

Transform internal integer representation, as received by calls to [caerDeviceConfigGet\(\)](#), into a coarse-fine bias structure, for easier handling and understanding of the various parameters.

Parameters

<i>coarseFineBias</i>	internal integer representation from device.
-----------------------	--

Returns

coarse-fine bias structure.

4.1.4.3 caerBiasShiftedSourceGenerate()

```
uint16_t caerBiasShiftedSourceGenerate (
    const struct caer_bias_shiftedsource shiftedSourceBias )
```

Transform shifted-source bias structure into internal integer representation, suited for sending directly to the device via [caerDeviceConfigSet\(\)](#).

Parameters

<i>shiftedSourceBias</i>	shifted-source bias structure.
--------------------------	--------------------------------

Returns

internal integer representation for device configuration.

4.1.4.4 caerBiasShiftedSourceParse()

```
struct caer_bias_shiftedsource caerBiasShiftedSourceParse (  
    const uint16_t shiftedSourceBias )
```

Transform internal integer representation, as received by calls to [caerDeviceConfigGet\(\)](#), into a shifted-source bias structure, for easier handling and understanding of the various parameters.

Parameters

<i>shiftedSourceBias</i>	internal integer representation from device.
--------------------------	--

Returns

shifted-source bias structure.

4.1.4.5 caerBiasVDACGenerate()

```
uint16_t caerBiasVDACGenerate (  
    const struct caer_bias_vdac vdacBias )
```

Transform VDAC bias structure into internal integer representation, suited for sending directly to the device via [caerDeviceConfigSet\(\)](#).

Parameters

<i>vdacBias</i>	VDAC bias structure.
-----------------	----------------------

Returns

internal integer representation for device configuration.

4.1.4.6 caerBiasVDACParse()

```
struct caer_bias_vdac caerBiasVDACParse (
    const uint16_t vdacBias )
```

Transform internal integer representation, as received by calls to [caerDeviceConfigGet\(\)](#), into a VDAC bias structure, for easier handling and understanding of the various parameters.

Parameters

<i>vdacBias</i>	internal integer representation from device.
-----------------	--

Returns

VDAC bias structure.

4.1.4.7 caerDavisInfoGet()

```
struct caer_davis_info caerDavisInfoGet (
    caerDeviceHandle handle )
```

Return basic information on the device, such as its ID, its resolution, the logic version, and so on. See the 'struct [caer_davis_info](#)' documentation for more details.

Parameters

<i>handle</i>	a valid device handle.
---------------	------------------------

Returns

a copy of the device information structure if successful, an empty structure (all zeros) on failure.

4.1.4.8 caerDavisROIConfigure()

```
bool caerDavisROIConfigure (
    caerDeviceHandle handle,
    uint8_t roiRegion,
    bool enable,
    uint16_t startX,
    uint16_t startY,
    uint16_t endX,
    uint16_t endY )
```

Configure an APS ROI region in one step. This function guarantees efficiency and atomicity (no intermediate-sized results possible).

Parameters

<i>handle</i>	a valid device handle.
<i>roiRegion</i>	which ROI region to configure, 0 or [0,3] if 'apsHasQuadROI' is defined.
<i>enable</i>	whether to enable or disable this ROI region.
<i>startX</i>	start corner X coordinate (0, 0 is upper left of frame).
<i>startY</i>	start corner Y coordinate (0, 0 is upper left of frame).
<i>endX</i>	end corner X coordinate (0, 0 is upper left of frame). Must be bigger than start.
<i>endY</i>	end corner Y coordinate (0, 0 is upper left of frame). Must be bigger than start.

Returns

true on success, false otherwise.

4.2 devices/device.h File Reference

```
#include "../libcaer.h"
#include "../events/packetContainer.h"
```

Macros

- `#define CAER_SUPPORTED_DEVICES_NUMBER 7`
- `#define CAER_HOST_CONFIG_DATAEXCHANGE -2`
- `#define CAER_HOST_CONFIG_PACKETS -3`
- `#define CAER_HOST_CONFIG_LOG -4`
- `#define CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE 0`
- `#define CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING 1`
- `#define CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS 2`
- `#define CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS 3`
- `#define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_PACKET_SIZE 0`
- `#define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL 1`
- `#define CAER_HOST_CONFIG_LOG_LEVEL 0`

Typedefs

- `typedef struct caer_device_handle * caerDeviceHandle`

Functions

- `bool caerDeviceClose (caerDeviceHandle *handle)`
- `bool caerDeviceSendDefaultConfig (caerDeviceHandle handle)`
- `bool caerDeviceConfigSet (caerDeviceHandle handle, int8_t modAddr, uint8_t paramAddr, uint32_t param)`
- `bool caerDeviceConfigGet (caerDeviceHandle handle, int8_t modAddr, uint8_t paramAddr, uint32_t *param)`
- `bool caerDeviceConfigGet64 (caerDeviceHandle handle, int8_t modAddr, uint8_t paramAddr, uint64_t *param)`
- `bool caerDeviceDataStart (caerDeviceHandle handle, void(*dataNotifyIncrease)(void *ptr), void(*dataNotifyDecrease)(void *ptr), void *dataNotifyUserPtr, void(*dataShutdownNotify)(void *ptr), void *dataShutdownUserPtr)`
- `bool caerDeviceDataStop (caerDeviceHandle handle)`
- `caerEventPacketContainer caerDeviceDataGet (caerDeviceHandle handle)`

4.2.1 Detailed Description

Common functions to access, configure and exchange data with supported devices. Also contains defines for host related configuration options.

4.2.2 Macro Definition Documentation

4.2.2.1 CAER_HOST_CONFIG_DATAEXCHANGE

```
#define CAER_HOST_CONFIG_DATAEXCHANGE -2
```

Module address: host-side data exchange (ring-buffer) configuration.

4.2.2.2 CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING

```
#define CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING 1
```

Parameter address for module CAER_HOST_CONFIG_DATAEXCHANGE: when calling [caerDeviceDataGet\(\)](#), the function can either be blocking, meaning it waits until it has a valid EventPacketContainer to return, or not, meaning it returns right away. This behavior can be set with this flag. Please see the [caerDeviceDataGet\(\)](#) documentation for more information on its return values.

4.2.2.3 CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE

```
#define CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE 0
```

Parameter address for module CAER_HOST_CONFIG_DATAEXCHANGE: set size of elements that can be held by the thread-safe FIFO buffer between the data transfer thread and the main thread. The default values are usually fine, only change them if you're running into lots of dropped/missing packets; you can turn on the INFO log level to see when this is the case.

4.2.2.4 CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS

```
#define CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS 2
```

Parameter address for module CAER_HOST_CONFIG_DATAEXCHANGE: whether to start all the data producer modules on the device (DVS, APS, Mux, ...) automatically when starting the data transfer thread with [caerDeviceDataStart\(\)](#) or not. If disabled, be aware you will have to start the right modules manually, which can be useful if you need precise control over which ones are running at any time.

4.2.2.5 CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS

```
#define CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS 3
```

Parameter address for module CAER_HOST_CONFIG_DATAEXCHANGE: whether to stop all the data producer modules on the device (DVS, APS, Mux, ...) automatically when stopping the data transfer thread with [caerDeviceDataStop\(\)](#) or not. If disabled, be aware you will have to stop the right modules manually, to halt the data flow, which can be useful if you need precise control over which ones are running at any time.

4.2.2.6 CAER_HOST_CONFIG_LOG

```
#define CAER_HOST_CONFIG_LOG -4
```

Module address: host-side logging configuration.

4.2.2.7 CAER_HOST_CONFIG_LOG_LEVEL

```
#define CAER_HOST_CONFIG_LOG_LEVEL 0
```

Parameter address for module CAER_HOST_CONFIG_LOG: set the log-level for this device, to be used when logging messages. Defaults to the value of the global log-level when the device was first opened.

4.2.2.8 CAER_HOST_CONFIG_PACKETS

```
#define CAER_HOST_CONFIG_PACKETS -3
```

Module address: host-side event packets generation configuration.

4.2.2.9 CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL

```
#define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL 1
```

Parameter address for module CAER_HOST_CONFIG_PACKETS: set the time interval between subsequent packet containers. Must be at least 1 microsecond. The value is in microseconds, and is checked across all types of events contained in the EventPacketContainer.

4.2.2.10 CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_PACKET_SIZE

```
#define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_PACKET_SIZE 0
```

Parameter address for module CAER_HOST_CONFIG_PACKETS: set the maximum number of events any of a packet container's packets may hold before it's made available to the user. Set to zero to disable. This is checked for each number of events held in each typed EventPacket that is a part of the EventPacketContainer.

4.2.2.11 CAER_SUPPORTED_DEVICES_NUMBER

```
#define CAER_SUPPORTED_DEVICES_NUMBER 7
```

Number of devices supported by this library. 0 - CAER_DEVICE_DVS128 1 - CAER_DEVICE_DAVIS_FX2 2 - CAER_DEVICE_DAVIS_FX3 3 - CAER_DEVICE_DYNAPSE 4 - CAER_DEVICE_DAVIS 5 - CAER_DEVICE_EDVS 6 - CAER_DEVICE_DAVIS_RPI

4.2.3 Typedef Documentation

4.2.3.1 caerDeviceHandle

```
typedef struct caer_device_handle* caerDeviceHandle
```

Pointer to an open device on which to operate.

4.2.4 Function Documentation

4.2.4.1 caerDeviceClose()

```
bool caerDeviceClose (
    caerDeviceHandle * handle )
```

Close a previously opened device and invalidate its handle.

Parameters

<i>handle</i>	pointer to a valid device handle. Will set handle to NULL if closing is successful, to prevent further usage of this handle for other operations.
---------------	---

Returns

true if closing was successful, false on errors.

4.2.4.2 caerDeviceConfigGet()

```
bool caerDeviceConfigGet (
    caerDeviceHandle handle,
    int8_t modAddr,
    uint8_t paramAddr,
    uint32_t * param )
```

Get the value of a configuration parameter.

Parameters

<i>handle</i>	a valid device handle.
<i>modAddr</i>	a module address, used to specify which configuration module one wants to query. Negative addresses are used for host-side configuration, while positive addresses (including zero) are used for device-side configuration.
<i>paramAddr</i>	a parameter address, to select a specific parameter to query from this particular configuration module. Only positive numbers (including zero) are allowed.
<i>param</i>	a pointer to an integer, in which to store the configuration parameter's current value. The integer will always be either set to zero (on failure), or to the current value (on success).

Returns

true if getting the configuration was successful, false on errors.

4.2.4.3 caerDeviceConfigGet64()

```
bool caerDeviceConfigGet64 (
    caerDeviceHandle handle,
    int8_t modAddr,
    uint8_t paramAddr,
    uint64_t * param )
```

Get the value of a 64bit configuration parameter. This is for special read-only configuration parameters only! Use only when required by the parameter's documentation!

Parameters

<i>handle</i>	a valid device handle.
<i>modAddr</i>	a module address, used to specify which configuration module one wants to query. Negative addresses are used for host-side configuration, while positive addresses (including zero) are used for device-side configuration.
<i>paramAddr</i>	a parameter address, to select a specific parameter to query from this particular configuration module. Only positive numbers (including zero) are allowed.
<i>param</i>	a pointer to a 64bit integer, in which to store the configuration parameter's current value. The integer will always be either set to zero (on failure), or to the current value (on success).

Returns

true if getting the configuration was successful, false on errors.

4.2.4.4 caerDeviceConfigSet()

```
bool caerDeviceConfigSet (
    caerDeviceHandle handle,
    int8_t modAddr,
    uint8_t paramAddr,
    uint32_t param )
```

Set a configuration parameter to a given value.

Parameters

<i>handle</i>	a valid device handle.
<i>modAddr</i>	a module address, used to specify which configuration module one wants to update. Negative addresses are used for host-side configuration, while positive addresses (including zero) are used for device-side configuration.
<i>paramAddr</i>	a parameter address, to select a specific parameter to update from this particular configuration module. Only positive numbers (including zero) are allowed.
<i>param</i>	a configuration parameter's new value.

Returns

true if sending the configuration was successful, false on errors.

4.2.4.5 caerDeviceDataGet()

```
caerEventPacketContainer caerDeviceDataGet (
    caerDeviceHandle handle )
```

Get an event packet container, which contains events of various types generated by the device, for further processing. The returned data structures are allocated in memory and will need to be freed. The [caerEventPacketContainerFree\(\)](#) function can be used to correctly free the full container memory. For single caerEventPackets, just use free(). This function can be made blocking with the CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING configuration parameter. By default it is non-blocking.

Parameters

<i>handle</i>	a valid device handle.
---------------	------------------------

Returns

a valid event packet container. NULL will be returned on errors, such as exceptional device shutdown, or when there is no container available in non-blocking mode. Always check this return value!

4.2.4.6 caerDeviceDataStart()

```
bool caerDeviceDataStart (
    caerDeviceHandle handle,
    void(*) (void *ptr) dataNotifyIncrease,
    void(*) (void *ptr) dataNotifyDecrease,
    void * dataNotifyUserPtr,
    void(*) (void *ptr) dataShutdownNotify,
    void * dataShutdownUserPtr )
```

Start getting data from the device, setting up the data transfers and starting the data producers (see CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS). Supports notification of new data and exceptional shutdown events via user-defined call-backs.

Parameters

<i>handle</i>	a valid device handle.
<i>dataNotifyIncrease</i>	function pointer, called every time a new piece of data available and has been put in the FIFO buffer for consumption. dataNotifyUserPtr will be passed as parameter to the function.
<i>dataNotifyDecrease</i>	function pointer, called every time a new piece of data has been consumed from the FIFO buffer inside caerDeviceDataGet() . dataNotifyUserPtr will be passed as parameter to the function.
<i>dataNotifyUserPtr</i>	pointer that will be passed to the dataNotifyIncrease and dataNotifyDecrease functions. Can be NULL.
<i>dataShutdownNotify</i> Generated by Doxygen	function pointer, called on exceptional shut-down of the data transfers. This is used to detect exceptional shut-downs that do not come from calling caerDeviceDataStop() , such as when the device is disconnected or all data transfers fail.
<i>dataShutdownUserPtr</i>	pointer that will be passed to the dataShutdownNotify function. Can be NULL.

Returns

true if starting the data transfer was successful, false on errors.

4.2.4.7 caerDeviceDataStop()

```
bool caerDeviceDataStop (
    caerDeviceHandle handle )
```

Stop getting data from the device, shutting down the data transfers and stopping the data producers (see CAER_↔HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS). This normal shut-down will not generate a notification (see caerDeviceDataStart()).

Parameters

<i>handle</i>	a valid device handle.
---------------	------------------------

Returns

true if stopping the data transfer was successful, false on errors.

4.2.4.8 caerDeviceSendDefaultConfig()

```
bool caerDeviceSendDefaultConfig (
    caerDeviceHandle handle )
```

Send a set of good default configuration settings to the device. This avoids users having to set every configuration option each time, especially when wanting to get going quickly or just needing to change a few settings to get to the desired operating mode.

Parameters

<i>handle</i>	a valid device handle.
---------------	------------------------

Returns

true if sending the configuration was successful, false on errors.

4.3 devices/device_discover.h File Reference

```
#include "davis.h"
#include "dvs128.h"
#include "dynapse.h"
#include "edvs.h"
```

Data Structures

- struct [caer_device_discovery_result](#)

Macros

- `#define CAER_DEVICE_DISCOVER_ALL -1`

Typedefs

- typedef struct [caer_device_discovery_result](#) * [caerDeviceDiscoveryResult](#)

Functions

- `ssize_t caerDeviceDiscover (int16_t deviceType, caerDeviceDiscoveryResult *discoveredDevices)`
- `caerDeviceHandle caerDeviceDiscoverOpen (uint16_t deviceId, caerDeviceDiscoveryResult discoveredDevice)`

4.3.1 Detailed Description

Functions to discover supported devices attached to the current host system, and then open them.

4.3.2 Macro Definition Documentation

4.3.2.1 CAER_DEVICE_DISCOVER_ALL

```
#define CAER_DEVICE_DISCOVER_ALL -1
```

Define for special value to discover all device types.

4.3.3 Typedef Documentation

4.3.3.1 caerDeviceDiscoveryResult

```
typedef struct caer\_device\_discovery\_result* caerDeviceDiscoveryResult
```

Pointer to result of a device discovery operation.

4.3.4 Function Documentation

4.3.4.1 caerDeviceDiscover()

```
ssize_t caerDeviceDiscover (
    int16_t deviceType,
    caerDeviceDiscoveryResult * discoveredDevices )
```

Discover all supported devices that are accessible on this system. Use -1 as 'deviceType' to search for any device, or an actual device type ID to only search for matches of that specific type.

Parameters

<i>deviceType</i>	type of device to search for, use -1 for any.
<i>discoveredDevices</i>	pointer to array of results, memory will be allocated for it automatically. On error, the pointer is set to NULL. Remember to free() the memory once done!

Returns

number of discovered devices, 0 if no device could be found; or -1 if an error occurred.

4.3.4.2 caerDeviceDiscoverOpen()

```
caerDeviceHandle caerDeviceDiscoverOpen (
    uint16_t deviceID,
    caerDeviceDiscoveryResult discoveredDevice )
```

Open a specific device based on information returned by [caerDeviceDiscover\(\)](#), then assign an ID to it and return a handle for further usage.

Parameters

<i>deviceID</i>	a unique ID to identify the device from others. Will be used as the source for EventPackets being generated from its data.
<i>discoveredDevice</i>	pointer to the result of a device discovery operation. Uniquely identifies a particular device.

Returns

a valid device handle that can be used with the other libcaer functions, or NULL on error. Always check for this!

4.4 devices/dvs128.h File Reference

```
#include "../events/polarity.h"
#include "../events/special.h"
#include "usb.h"
```

Data Structures

- struct [caer_dvs128_info](#)

Macros

- #define [CAER_DEVICE_DVS128](#) 0
- #define [DVS128_CONFIG_DVS](#) 0
- #define [DVS128_CONFIG_BIAS](#) 1
- #define [DVS128_CONFIG_DVS_RUN](#) 0
- #define [DVS128_CONFIG_DVS_TIMESTAMP_RESET](#) 1
- #define [DVS128_CONFIG_DVS_ARRAY_RESET](#) 2
- #define [DVS128_CONFIG_DVS_TS_MASTER](#) 3
- #define [DVS128_CONFIG_BIAS_CAS](#) 0
- #define [DVS128_CONFIG_BIAS_INJGND](#) 1
- #define [DVS128_CONFIG_BIAS_REQPD](#) 2
- #define [DVS128_CONFIG_BIAS_PUX](#) 3
- #define [DVS128_CONFIG_BIAS_DIFFOFF](#) 4
- #define [DVS128_CONFIG_BIAS_REQ](#) 5
- #define [DVS128_CONFIG_BIAS_REFR](#) 6
- #define [DVS128_CONFIG_BIAS_PUY](#) 7
- #define [DVS128_CONFIG_BIAS_DIFFON](#) 8
- #define [DVS128_CONFIG_BIAS_DIFF](#) 9
- #define [DVS128_CONFIG_BIAS_FOLL](#) 10
- #define [DVS128_CONFIG_BIAS_PR](#) 11

Functions

- struct [caer_dvs128_info](#) [caerDVS128InfoGet](#) ([caerDeviceHandle](#) handle)

4.4.1 Detailed Description

DVS128 specific configuration defines and information structures.

4.4.2 Macro Definition Documentation

4.4.2.1 CAER_DEVICE_DVS128

```
#define CAER_DEVICE_DVS128 0
```

Device type definition for iniVation DVS128.

4.4.2.2 DVS128_CONFIG_BIAS

```
#define DVS128_CONFIG_BIAS 1
```

Module address: device-side chip bias generator configuration.

4.4.2.3 DVS128_CONFIG_BIAS_CAS

```
#define DVS128_CONFIG_BIAS_CAS 0
```

Parameter address for module DVS128_CONFIG_BIAS: First stage amplifier cascode bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.4.2.4 DVS128_CONFIG_BIAS_DIFF

```
#define DVS128_CONFIG_BIAS_DIFF 9
```

Parameter address for module DVS128_CONFIG_BIAS: Differential (second stage amplifier) bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.4.2.5 DVS128_CONFIG_BIAS_DIFFOFF

```
#define DVS128_CONFIG_BIAS_DIFFOFF 4
```

Parameter address for module DVS128_CONFIG_BIAS: Off events threshold bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.4.2.6 DVS128_CONFIG_BIAS_DIFFON

```
#define DVS128_CONFIG_BIAS_DIFFON 8
```

Parameter address for module DVS128_CONFIG_BIAS: On events threshold bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.4.2.7 DVS128_CONFIG_BIAS_FOLL

```
#define DVS128_CONFIG_BIAS_FOLL 10
```

Parameter address for module DVS128_CONFIG_BIAS: Source follower bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.4.2.8 DVS128_CONFIG_BIAS_INJGND

```
#define DVS128_CONFIG_BIAS_INJGND 1
```

Parameter address for module DVS128_CONFIG_BIAS: Injected ground bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.4.2.9 DVS128_CONFIG_BIAS_PR

```
#define DVS128_CONFIG_BIAS_PR 11
```

Parameter address for module DVS128_CONFIG_BIAS: Photoreceptor bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.4.2.10 DVS128_CONFIG_BIAS_PUX

```
#define DVS128_CONFIG_BIAS_PUX 3
```

Parameter address for module DVS128_CONFIG_BIAS: Pull up on request from X arbiter (AER). See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.4.2.11 DVS128_CONFIG_BIAS_PUY

```
#define DVS128_CONFIG_BIAS_PUY 7
```

Parameter address for module DVS128_CONFIG_BIAS: Pull up on request from Y arbiter (AER). See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.4.2.12 DVS128_CONFIG_BIAS_REFR

```
#define DVS128_CONFIG_BIAS_REFR 6
```

Parameter address for module DVS128_CONFIG_BIAS: Refractory period bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.4.2.13 DVS128_CONFIG_BIAS_REQ

```
#define DVS128_CONFIG_BIAS_REQ 5
```

Parameter address for module DVS128_CONFIG_BIAS: Pull down for passive load inverters in digital AER pixel circuitry. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.4.2.14 DVS128_CONFIG_BIAS_REQPD

```
#define DVS128_CONFIG_BIAS_REQPD 2
```

Parameter address for module DVS128_CONFIG_BIAS: Pull down on chip request (AER). See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.4.2.15 DVS128_CONFIG_DVS

```
#define DVS128_CONFIG_DVS 0
```

Module address: device-side DVS configuration.

4.4.2.16 DVS128_CONFIG_DVS_ARRAY_RESET

```
#define DVS128_CONFIG_DVS_ARRAY_RESET 2
```

Parameter address for module DVS128_CONFIG_DVS: reset the whole DVS pixel array. This is a temporary configuration switch and will reset itself right away.

4.4.2.17 DVS128_CONFIG_DVS_RUN

```
#define DVS128_CONFIG_DVS_RUN 0
```

Parameter address for module DVS128_CONFIG_DVS: run the DVS chip and generate polarity event data.

4.4.2.18 DVS128_CONFIG_DVS_TIMESTAMP_RESET

```
#define DVS128_CONFIG_DVS_TIMESTAMP_RESET 1
```

Parameter address for module DVS128_CONFIG_DVS: reset the time-stamp counter of the device. This is a temporary configuration switch and will reset itself right away.

4.4.2.19 DVS128_CONFIG_DVS_TS_MASTER

```
#define DVS128_CONFIG_DVS_TS_MASTER 3
```

Parameter address for module DVS128_CONFIG_DVS: control if this DVS is a timestamp master device. Default is enabled.

4.4.3 Function Documentation

4.4.3.1 caerDVS128InfoGet()

```
struct caer_dvs128_info caerDVS128InfoGet (
    caerDeviceHandle handle )
```

Return basic information on the device, such as its ID, its resolution, the logic version, and so on. See the 'struct caer_dvs128_info' documentation for more details.

Parameters

<i>handle</i>	a valid device handle.
---------------	------------------------

Returns

a copy of the device information structure if successful, an empty structure (all zeros) on failure.

4.5 devices/dynapse.h File Reference

```
#include "../events/special.h"
#include "../events/spike.h"
#include "usb.h"
```

Data Structures

- struct [caer_dynapse_info](#)
- struct [caer_bias_dynapse](#)

Macros

- `#define CAER_DEVICE_DYNAPSE` 3
- `#define DYNAPSE_CHIP_DYNAPSE` 64
- `#define DYNAPSE_CONFIG_MUX` 0
- `#define DYNAPSE_CONFIG_AER` 1
- `#define DYNAPSE_CONFIG_CHIP` 5
- `#define DYNAPSE_CONFIG_SYSINFO` 6
- `#define DYNAPSE_CONFIG_USB` 9
- `#define DYNAPSE_CONFIG_CLEAR_CAM` 10
- `#define DYNAPSE_CONFIG_DEFAULT_SRAM` 11
- `#define DYNAPSE_CONFIG_MONITOR_NEU` 12
- `#define DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY` 13
- `#define DYNAPSE_CONFIG_SRAM` 14
- `#define DYNAPSE_CONFIG_SYNAPSERECONFIG` 15
- `#define DYNAPSE_CONFIG_SPIKEGEN` 16
- `#define DYNAPSE_CONFIG_TAU2_SET` 17
- `#define DYNAPSE_CONFIG_POISSONSPIKEGEN` 18
- `#define DYNAPSE_CONFIG_TAU1_RESET` 19
- `#define DYNAPSE_CONFIG_TAU2_RESET` 20
- `#define DYNAPSE_CONFIG_POISSONSPIKEGEN_RUN` 0
- `#define DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEADDRESS` 1
- `#define DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEDATA` 2
- `#define DYNAPSE_CONFIG_POISSONSPIKEGEN_CHIPID` 3
- `#define DYNAPSE_CONFIG_SPIKEGEN_RUN` 0
- `#define DYNAPSE_CONFIG_SPIKEGEN_VARMODE` 1
- `#define DYNAPSE_CONFIG_SPIKEGEN_BASEADDR` 2
- `#define DYNAPSE_CONFIG_SPIKEGEN_STIMCOUNT` 3
- `#define DYNAPSE_CONFIG_SPIKEGEN_ISI` 4
- `#define DYNAPSE_CONFIG_SPIKEGEN_ISIBASE` 5
- `#define DYNAPSE_CONFIG_SPIKEGEN_REPEAT` 6
- `#define DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN` 0
- `#define DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOBALKERNEL` 1
- `#define DYNAPSE_CONFIG_SYNAPSERECONFIG_USESRAMKERNELS` 2
- `#define DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPSELECT` 3
- `#define DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAMBASEADDR` 4
- `#define DYNAPSE_CONFIG_SRAM_ADDRESS` 1
- `#define DYNAPSE_CONFIG_SRAM_READDATA` 2
- `#define DYNAPSE_CONFIG_SRAM_WRITEDATA` 3
- `#define DYNAPSE_CONFIG_SRAM_RWCOMMAND` 4
- `#define DYNAPSE_CONFIG_SRAM_READ` 0
- `#define DYNAPSE_CONFIG_SRAM_WRITE` 1
- `#define DYNAPSE_CONFIG_SRAM_BURSTMODE` 5
- `#define DYNAPSE_CONFIG_MUX_RUN` 0
- `#define DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN` 1
- `#define DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET` 2
- `#define DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE` 3
- `#define DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRANSFER_STALL` 4
- `#define DYNAPSE_CONFIG_MUX_HAS_STATISTICS` 10

- #define DYNAPSE_CONFIG_MUX_STATISTICS_AER_DROPPED 11
- #define DYNAPSE_CONFIG_AER_RUN 3
- #define DYNAPSE_CONFIG_AER_ACK_DELAY 4
- #define DYNAPSE_CONFIG_AER_ACK_EXTENSION 6
- #define DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_STALL 8
- #define DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONTROL 10
- #define DYNAPSE_CONFIG_AER_HAS_STATISTICS 40
- #define DYNAPSE_CONFIG_AER_STATISTICS_EVENTS 41
- #define DYNAPSE_CONFIG_AER_STATISTICS_EVENTS_DROPPED 45
- #define DYNAPSE_CONFIG_CHIP_RUN 0
- #define DYNAPSE_CONFIG_CHIP_ID 1
- #define DYNAPSE_CONFIG_CHIP_CONTENT 2
- #define DYNAPSE_CONFIG_CHIP_REQ_DELAY 3
- #define DYNAPSE_CONFIG_CHIP_REQ_EXTENSION 4
- #define DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION 0
- #define DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER 1
- #define DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER 2
- #define DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK 3
- #define DYNAPSE_CONFIG_USB_RUN 0
- #define DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY 1
- #define DYNAPSE_CONFIG_SRAM_DIRECTION_POS 0
- #define DYNAPSE_CONFIG_SRAM_DIRECTION_NEG 1
- #define DYNAPSE_CONFIG_SRAM_DIRECTION_Y_NORTH 0
- #define DYNAPSE_CONFIG_SRAM_DIRECTION_Y_SOUTH 1
- #define DYNAPSE_CONFIG_SRAM_DIRECTION_X_EAST 0
- #define DYNAPSE_CONFIG_SRAM_DIRECTION_X_WEST 1
- #define DYNAPSE_X4BOARD_NUMCHIPS 4
- #define DYNAPSE_X4BOARD_NEUX 64
- #define DYNAPSE_X4BOARD_NEUY 64
- #define DYNAPSE_X4BOARD_COREX 4
- #define DYNAPSE_X4BOARD_COREY 4
- #define DYNAPSE_CONFIG_DYNAPSE_U0 0
- *Chip 0 ID.*
- #define DYNAPSE_CONFIG_DYNAPSE_U1 1
- *Chip 1 ID.*
- #define DYNAPSE_CONFIG_DYNAPSE_U2 2
- *Chip 2 ID.*
- #define DYNAPSE_CONFIG_DYNAPSE_U3 3
- *Chip 3 ID.*
- #define DYNAPSE_CONFIG_NUMCORES 4
- *Number of cores per chip.*
- #define DYNAPSE_CONFIG_NUMNEURONS 1024
- *Number of neurons in single chip.*
- #define DYNAPSE_CONFIG_NUMNEURONS_CORE 256
- *Number of neurons per core.*
- #define DYNAPSE_CONFIG_XCHIPSIZE 32
- *Number of columns of neurons in a chip.*
- #define DYNAPSE_CONFIG_YCHIPSIZE 32
- *Number of rows of neurons in a core.*
- #define DYNAPSE_CONFIG_NEUCOL 16
- *Number of columns of neurons in a core.*
- #define DYNAPSE_CONFIG_NEUROW 16
- *Number of rows of neurons in a core.*

- #define [DYNAPSE_CONFIG_CAMCOL](#) 16
Number of columns of CAMs in a core.
- #define [DYNAPSE_CONFIG_NUMCAM_NEU](#) 64
Number of CAMs per neuron.
- #define [DYNAPSE_CONFIG_NUMSRAM_NEU](#) 4
Number of SRAM cells per neuron.
- #define [DYNAPSE_CONFIG_CAMTYPE_F_EXC](#) 3
Fast excitatory synapse.
- #define [DYNAPSE_CONFIG_CAMTYPE_S_EXC](#) 2
Slow excitatory synapse.
- #define [DYNAPSE_CONFIG_CAMTYPE_F_INH](#) 1
Fast inhibitory synapse.
- #define [DYNAPSE_CONFIG_CAMTYPE_S_INH](#) 0
Slow inhibitory synapse.
- #define [DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P](#) 0
- #define [DYNAPSE_CONFIG_BIAS_C0_PS_WEIGHT_INH_S_N](#) 2
- #define [DYNAPSE_CONFIG_BIAS_C0_PS_WEIGHT_INH_F_N](#) 4
- #define [DYNAPSE_CONFIG_BIAS_C0_PS_WEIGHT_EXC_S_N](#) 6
- #define [DYNAPSE_CONFIG_BIAS_C0_PS_WEIGHT_EXC_F_N](#) 8
- #define [DYNAPSE_CONFIG_BIAS_C0_IF_RFR_N](#) 10
- #define [DYNAPSE_CONFIG_BIAS_C0_IF_TAU1_N](#) 12
- #define [DYNAPSE_CONFIG_BIAS_C0_IF_AHTAU_N](#) 14
- #define [DYNAPSE_CONFIG_BIAS_C0_IF_CASC_N](#) 16
- #define [DYNAPSE_CONFIG_BIAS_C0_IF_TAU2_N](#) 18
- #define [DYNAPSE_CONFIG_BIAS_C0_IF_BUF_P](#) 20
- #define [DYNAPSE_CONFIG_BIAS_C0_IF_AHTHR_N](#) 22
- #define [DYNAPSE_CONFIG_BIAS_C0_IF_THR_N](#) 24
- #define [DYNAPSE_CONFIG_BIAS_C0_NPDPIE_THR_S_P](#) 26
- #define [DYNAPSE_CONFIG_BIAS_C0_NPDPIE_THR_F_P](#) 28
- #define [DYNAPSE_CONFIG_BIAS_C0_NPDPII_THR_F_P](#) 30
- #define [DYNAPSE_CONFIG_BIAS_C0_NPDPII_THR_S_P](#) 32
- #define [DYNAPSE_CONFIG_BIAS_C0_IF_NMDA_N](#) 34
- #define [DYNAPSE_CONFIG_BIAS_C0_IF_DC_P](#) 36
- #define [DYNAPSE_CONFIG_BIAS_C0_IF_AHW_P](#) 38
- #define [DYNAPSE_CONFIG_BIAS_C0_NPDPII_TAU_S_P](#) 40
- #define [DYNAPSE_CONFIG_BIAS_C0_NPDPII_TAU_F_P](#) 42
- #define [DYNAPSE_CONFIG_BIAS_C0_NPDPIE_TAU_F_P](#) 44
- #define [DYNAPSE_CONFIG_BIAS_C0_NPDPIE_TAU_S_P](#) 46
- #define [DYNAPSE_CONFIG_BIAS_C0_R2R_P](#) 48
- #define [DYNAPSE_CONFIG_BIAS_C1_PULSE_PWLK_P](#) 1
- #define [DYNAPSE_CONFIG_BIAS_C1_PS_WEIGHT_INH_S_N](#) 3
- #define [DYNAPSE_CONFIG_BIAS_C1_PS_WEIGHT_INH_F_N](#) 5
- #define [DYNAPSE_CONFIG_BIAS_C1_PS_WEIGHT_EXC_S_N](#) 7
- #define [DYNAPSE_CONFIG_BIAS_C1_PS_WEIGHT_EXC_F_N](#) 9
- #define [DYNAPSE_CONFIG_BIAS_C1_IF_RFR_N](#) 11
- #define [DYNAPSE_CONFIG_BIAS_C1_IF_TAU1_N](#) 13
- #define [DYNAPSE_CONFIG_BIAS_C1_IF_AHTAU_N](#) 15
- #define [DYNAPSE_CONFIG_BIAS_C1_IF_CASC_N](#) 17
- #define [DYNAPSE_CONFIG_BIAS_C1_IF_TAU2_N](#) 19
- #define [DYNAPSE_CONFIG_BIAS_C1_IF_BUF_P](#) 21
- #define [DYNAPSE_CONFIG_BIAS_C1_IF_AHTHR_N](#) 23
- #define [DYNAPSE_CONFIG_BIAS_C1_IF_THR_N](#) 25
- #define [DYNAPSE_CONFIG_BIAS_C1_NPDPIE_THR_S_P](#) 27

- #define DYNAPSE_CONFIG_BIAS_C1_NPDPIE_THR_F_P 29
- #define DYNAPSE_CONFIG_BIAS_C1_NPDPII_THR_F_P 31
- #define DYNAPSE_CONFIG_BIAS_C1_NPDPII_THR_S_P 33
- #define DYNAPSE_CONFIG_BIAS_C1_IF_NMDA_N 35
- #define DYNAPSE_CONFIG_BIAS_C1_IF_DC_P 37
- #define DYNAPSE_CONFIG_BIAS_C1_IF_AHW_P 39
- #define DYNAPSE_CONFIG_BIAS_C1_NPDPII_TAU_S_P 41
- #define DYNAPSE_CONFIG_BIAS_C1_NPDPII_TAU_F_P 43
- #define DYNAPSE_CONFIG_BIAS_C1_NPDPIE_TAU_F_P 45
- #define DYNAPSE_CONFIG_BIAS_C1_NPDPIE_TAU_S_P 47
- #define DYNAPSE_CONFIG_BIAS_C1_R2R_P 49
- #define DYNAPSE_CONFIG_BIAS_U_BUFFER 50
- #define DYNAPSE_CONFIG_BIAS_U_SSP 51
- #define DYNAPSE_CONFIG_BIAS_U_SSN 52
- #define DYNAPSE_CONFIG_BIAS_C2_PULSE_PWLK_P 64
- #define DYNAPSE_CONFIG_BIAS_C2_PS_WEIGHT_INH_S_N 66
- #define DYNAPSE_CONFIG_BIAS_C2_PS_WEIGHT_INH_F_N 68
- #define DYNAPSE_CONFIG_BIAS_C2_PS_WEIGHT_EXC_S_N 70
- #define DYNAPSE_CONFIG_BIAS_C2_PS_WEIGHT_EXC_F_N 72
- #define DYNAPSE_CONFIG_BIAS_C2_IF_RFR_N 74
- #define DYNAPSE_CONFIG_BIAS_C2_IF_TAU1_N 76
- #define DYNAPSE_CONFIG_BIAS_C2_IF_AHTAU_N 78
- #define DYNAPSE_CONFIG_BIAS_C2_IF_CASC_N 80
- #define DYNAPSE_CONFIG_BIAS_C2_IF_TAU2_N 82
- #define DYNAPSE_CONFIG_BIAS_C2_IF_BUF_P 84
- #define DYNAPSE_CONFIG_BIAS_C2_IF_AHTHR_N 86
- #define DYNAPSE_CONFIG_BIAS_C2_IF_THR_N 88
- #define DYNAPSE_CONFIG_BIAS_C2_NPDPIE_THR_S_P 90
- #define DYNAPSE_CONFIG_BIAS_C2_NPDPIE_THR_F_P 92
- #define DYNAPSE_CONFIG_BIAS_C2_NPDPII_THR_F_P 94
- #define DYNAPSE_CONFIG_BIAS_C2_NPDPII_THR_S_P 96
- #define DYNAPSE_CONFIG_BIAS_C2_IF_NMDA_N 98
- #define DYNAPSE_CONFIG_BIAS_C2_IF_DC_P 100
- #define DYNAPSE_CONFIG_BIAS_C2_IF_AHW_P 102
- #define DYNAPSE_CONFIG_BIAS_C2_NPDPII_TAU_S_P 104
- #define DYNAPSE_CONFIG_BIAS_C2_NPDPII_TAU_F_P 106
- #define DYNAPSE_CONFIG_BIAS_C2_NPDPIE_TAU_F_P 108
- #define DYNAPSE_CONFIG_BIAS_C2_NPDPIE_TAU_S_P 110
- #define DYNAPSE_CONFIG_BIAS_C2_R2R_P 112
- #define DYNAPSE_CONFIG_BIAS_C3_PULSE_PWLK_P 65
- #define DYNAPSE_CONFIG_BIAS_C3_PS_WEIGHT_INH_S_N 67
- #define DYNAPSE_CONFIG_BIAS_C3_PS_WEIGHT_INH_F_N 69
- #define DYNAPSE_CONFIG_BIAS_C3_PS_WEIGHT_EXC_S_N 71
- #define DYNAPSE_CONFIG_BIAS_C3_PS_WEIGHT_EXC_F_N 73
- #define DYNAPSE_CONFIG_BIAS_C3_IF_RFR_N 75
- #define DYNAPSE_CONFIG_BIAS_C3_IF_TAU1_N 77
- #define DYNAPSE_CONFIG_BIAS_C3_IF_AHTAU_N 79
- #define DYNAPSE_CONFIG_BIAS_C3_IF_CASC_N 81
- #define DYNAPSE_CONFIG_BIAS_C3_IF_TAU2_N 83
- #define DYNAPSE_CONFIG_BIAS_C3_IF_BUF_P 85
- #define DYNAPSE_CONFIG_BIAS_C3_IF_AHTHR_N 87
- #define DYNAPSE_CONFIG_BIAS_C3_IF_THR_N 89
- #define DYNAPSE_CONFIG_BIAS_C3_NPDPIE_THR_S_P 91
- #define DYNAPSE_CONFIG_BIAS_C3_NPDPIE_THR_F_P 93
- #define DYNAPSE_CONFIG_BIAS_C3_NPDPII_THR_F_P 95

- `#define DYNAPSE_CONFIG_BIAS_C3_NPDPII_THR_S_P` 97
- `#define DYNAPSE_CONFIG_BIAS_C3_IF_NMDA_N` 99
- `#define DYNAPSE_CONFIG_BIAS_C3_IF_DC_P` 101
- `#define DYNAPSE_CONFIG_BIAS_C3_IF_AHW_P` 103
- `#define DYNAPSE_CONFIG_BIAS_C3_NPDPII_TAU_S_P` 105
- `#define DYNAPSE_CONFIG_BIAS_C3_NPDPII_TAU_F_P` 107
- `#define DYNAPSE_CONFIG_BIAS_C3_NPDPIE_TAU_F_P` 109
- `#define DYNAPSE_CONFIG_BIAS_C3_NPDPIE_TAU_S_P` 111
- `#define DYNAPSE_CONFIG_BIAS_C3_R2R_P` 113
- `#define DYNAPSE_CONFIG_BIAS_D_BUFFER` 114
- `#define DYNAPSE_CONFIG_BIAS_D_SSP` 115
- `#define DYNAPSE_CONFIG_BIAS_D_SSN` 116

Functions

- `struct caer_dynapse_info caerDynapseInfoGet` (`caerDeviceHandle` handle)
- `uint32_t caerBiasDynapseGenerate` (`const struct caer_bias_dynapse` dynapseBias)
- `struct caer_bias_dynapse caerBiasDynapseParse` (`const uint32_t` dynapseBias)
- `bool caerDynapseWriteSramWords` (`caerDeviceHandle` handle, `const uint16_t *data`, `uint32_t` baseAddr, `size_t` numWords)
- `bool caerDynapseWritePoissonSpikeRate` (`caerDeviceHandle` handle, `uint16_t` neuronAddr, `float` rateHz)
- `bool caerDynapseWriteSram` (`caerDeviceHandle` handle, `uint8_t` coreId, `uint8_t` neuronAddrCore, `uint8_t` virtualCoreId, `bool` sx, `uint8_t` dx, `bool` sy, `uint8_t` dy, `uint8_t` sramId, `uint8_t` destinationCore)
- `bool caerDynapseWriteSramN` (`caerDeviceHandle` handle, `uint16_t` neuronAddr, `uint8_t` sramId, `uint8_t` virtualCoreId, `bool` sx, `uint8_t` dx, `bool` sy, `uint8_t` dy, `uint8_t` destinationCore)
- `bool caerDynapseWriteCam` (`caerDeviceHandle` handle, `uint16_t` inputNeuronAddr, `uint16_t` neuronAddr, `uint8_t` camId, `uint8_t` synapseType)
- `bool caerDynapseSendDataToUSB` (`caerDeviceHandle` handle, `const uint32_t *data`, `size_t` numConfig)
- `uint32_t caerDynapseGenerateCamBits` (`uint16_t` inputNeuronAddr, `uint16_t` neuronAddr, `uint8_t` camId, `uint8_t` synapseType)
- `uint32_t caerDynapseGenerateSramBits` (`uint16_t` neuronAddr, `uint8_t` sramId, `uint8_t` virtualCoreId, `bool` sx, `uint8_t` dx, `bool` sy, `uint8_t` dy, `uint8_t` destinationCore)
- `uint16_t caerDynapseCoreXYToNeuronId` (`uint8_t` coreId, `uint8_t` columnX, `uint8_t` rowY)
- `uint16_t caerDynapseCoreAddrToNeuronId` (`uint8_t` coreId, `uint8_t` neuronAddrCore)
- `uint16_t caerDynapseSpikeEventGetX` (`caerSpikeEventConst` event)
- `uint16_t caerDynapseSpikeEventGetY` (`caerSpikeEventConst` event)
- `struct caer_spike_event caerDynapseSpikeEventFromXY` (`uint16_t` x, `uint16_t` y)

4.5.1 Detailed Description

Dynap-se specific configuration defines and information structures.

4.5.2 Macro Definition Documentation

4.5.2.1 CAER_DEVICE_DYNAPSE

```
#define CAER_DEVICE_DYNAPSE 3
```

Device type definition for aiCTX Dynap-se FX2-based boards.

4.5.2.2 DYNAPSE_CHIP_DYNAPSE

```
#define DYNAPSE_CHIP_DYNAPSE 64
```

Dynap-se chip identifier.

4.5.2.3 DYNAPSE_CONFIG_AER

```
#define DYNAPSE_CONFIG_AER 1
```

Module address: device-side AER configuration (from chip). The AER state machine handshakes with the chip's AER bus and gets the spike events from it. It supports various configurable delays.

4.5.2.4 DYNAPSE_CONFIG_AER_ACK_DELAY

```
#define DYNAPSE_CONFIG_AER_ACK_DELAY 4
```

Parameter address for module DYNAPSE_CONFIG_AER: delay capturing the data and acknowledging it on the AER bus for the events by this many LogicClock cycles.

4.5.2.5 DYNAPSE_CONFIG_AER_ACK_EXTENSION

```
#define DYNAPSE_CONFIG_AER_ACK_EXTENSION 6
```

Parameter address for module DYNAPSE_CONFIG_AER: extend the length of the acknowledge on the AER bus for the events by this many LogicClock cycles.

4.5.2.6 DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONTROL

```
#define DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONTROL 10
```

Parameter address for module DYNAPSE_CONFIG_AER: enable external AER control. This ensures the chip and the neuron array are running, but doesn't do the handshake and leaves the ACK pin in high-impedance, to allow for an external system to take over the AER communication with the chip. DYNAPSE_CONFIG_AER_RUN has to be turned off for this to work.

4.5.2.7 DYNAPSE_CONFIG_AER_HAS_STATISTICS

```
#define DYNAPSE_CONFIG_AER_HAS_STATISTICS 40
```

Parameter address for module DYNAPSE_CONFIG_AER: read-only parameter, information about the presence of the statistics feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_dynapse_info](#)' documentation to get this information.

4.5.2.8 DYNAPSE_CONFIG_AER_RUN

```
#define DYNAPSE_CONFIG_AER_RUN 3
```

Parameter address for module DYNAPSE_CONFIG_AER: run the AER state machine and get spike events from the chip by handshaking with its AER bus.

4.5.2.9 DYNAPSE_CONFIG_AER_STATISTICS_EVENTS

```
#define DYNAPSE_CONFIG_AER_STATISTICS_EVENTS 41
```

Parameter address for module DYNAPSE_CONFIG_AER: read-only parameter, representing the number of event transactions completed on the device. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.5.2.10 DYNAPSE_CONFIG_AER_STATISTICS_EVENTS_DROPPED

```
#define DYNAPSE_CONFIG_AER_STATISTICS_EVENTS_DROPPED 45
```

Parameter address for module DYNAPSE_CONFIG_AER: read-only parameter, representing the number of dropped transaction sequences on the device due to full buffers. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.5.2.11 DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_STALL

```
#define DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_STALL 8
```

Parameter address for module DYNAPSE_CONFIG_AER: if the output FIFO for this module is full, stall the AER handshake with the chip and wait until it's free again, instead of just continuing the handshake and dropping the resulting events.

4.5.2.12 DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P

```
#define DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P 0
```

Parameter address for module DYNAPSE_CONFIG_BIAS: DYNAPSE chip biases. Bias configuration values must be generated using the proper functions, which are:

- [caerBiasDynapseGenerate\(\)](#) for Dynap-se coarse-fine (current) biases. See '<https://ai-ctx.com/support/>' section 'Neuron's behaviors and parameters tuning'.

4.5.2.13 DYNAPSE_CONFIG_CHIP

```
#define DYNAPSE_CONFIG_CHIP 5
```

Module address: device-side chip control configuration. This state machine is responsible for configuring the chip's internal control registers, to set special options and biases.

4.5.2.14 DYNAPSE_CONFIG_CHIP_CONTENT

```
#define DYNAPSE_CONFIG_CHIP_CONTENT 2
```

Parameter address for module DYNAPSE_CONFIG_CHIP: set the configuration content to send to the chip. Every time this changes, the chip ID is appended and the configuration is sent out to the chip.

4.5.2.15 DYNAPSE_CONFIG_CHIP_ID

```
#define DYNAPSE_CONFIG_CHIP_ID 1
```

Parameter address for module DYNAPSE_CONFIG_CHIP: set the chip ID to which configuration content is being sent.

4.5.2.16 DYNAPSE_CONFIG_CHIP_REQ_DELAY

```
#define DYNAPSE_CONFIG_CHIP_REQ_DELAY 3
```

Parameter address for module DYNAPSE_CONFIG_CHIP: delay doing the request after putting out the data by this many LogicClock cycles.

4.5.2.17 DYNAPSE_CONFIG_CHIP_REQ_EXTENSION

```
#define DYNAPSE_CONFIG_CHIP_REQ_EXTENSION 4
```

Parameter address for module DYNAPSE_CONFIG_CHIP: extend the request after receiving the ACK by this many LogicClock cycles.

4.5.2.18 DYNAPSE_CONFIG_CHIP_RUN

```
#define DYNAPSE_CONFIG_CHIP_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_CHIP: enable the configuration AER state machine to send bias and control configuration to the chip.

4.5.2.19 DYNAPSE_CONFIG_CLEAR_CAM

```
#define DYNAPSE_CONFIG_CLEAR_CAM 10
```

Clear CAM content, on all cores of a chip. No arguments are used. Remember to select the chip you want to configure before this!

4.5.2.20 DYNAPSE_CONFIG_DEFAULT_SRAM

```
#define DYNAPSE_CONFIG_DEFAULT_SRAM 11
```

Clear SRAM content, use one SRAM cell (cell 0, out of the four available) to monitor neurons via USB. 'paramAddr' is the chip ID on which to operate, other arguments are unused. Remember to also select the chip you want to configure before this!

4.5.2.21 DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY

```
#define DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY 13
```

Clear SRAM content, route nothing outside (all four SRAM cells zero). No arguments are used. Remember to select the chip you want to configure before this!

4.5.2.22 DYNAPSE_CONFIG_MONITOR_NEU

```
#define DYNAPSE_CONFIG_MONITOR_NEU 12
```

Setup analog neuron monitoring via SMA connectors. 'paramAddr' takes the core ID to be monitored, 'param' the neuron ID. Remember to select the chip you want to configure before this!

4.5.2.23 DYNAPSE_CONFIG_MUX

```
#define DYNAPSE_CONFIG_MUX 0
```

Module address: device-side Multiplexer configuration. The Multiplexer is responsible for mixing, timestamping and outputting (via USB) the various event types generated by the device. It is also responsible for timestamp generation.

4.5.2.24 DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRANSFER_STALL

```
#define DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRANSFER_STALL 4
```

Parameter address for module DYNAPSE_CONFIG_MUX: drop AER events if the USB output FIFO is full, instead of having them pile up at the input FIFOs.

4.5.2.25 DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE

```
#define DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE 3
```

Parameter address for module DYNAPSE_CONFIG_MUX: under normal circumstances, the chip's bias generator is only powered up when either the AER or the configuration state machines are running, to save power. This flag forces the bias generator to be powered up all the time.

4.5.2.26 DYNAPSE_CONFIG_MUX_HAS_STATISTICS

```
#define DYNAPSE_CONFIG_MUX_HAS_STATISTICS 10
```

Parameter address for module DYNAPSE_CONFIG_MUX: read-only parameter, information about the presence of the statistics feature. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_dynapse_info](#)' documentation to get this information.

4.5.2.27 DYNAPSE_CONFIG_MUX_RUN

```
#define DYNAPSE_CONFIG_MUX_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_MUX: run the Multiplexer state machine, which is responsible for mixing the various event types at the device level, timestamping them and outputting them via USB or other connectors.

4.5.2.28 DYNAPSE_CONFIG_MUX_STATISTICS_AER_DROPPED

```
#define DYNAPSE_CONFIG_MUX_STATISTICS_AER_DROPPED 11
```

Parameter address for module DYNAPSE_CONFIG_MUX: read-only parameter, representing the number of dropped AER (spike) events on the device due to full USB buffers. This is a 64bit value, and should always be read using the function: [caerDeviceConfigGet64\(\)](#).

4.5.2.29 DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET

```
#define DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET 2
```

Parameter address for module DYNAPSE_CONFIG_MUX: reset the Timestamp Generator to zero. This also sends a reset pulse to all connected slave devices, resetting their timestamp too.

4.5.2.30 DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN

```
#define DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN 1
```

Parameter address for module DYNAPSE_CONFIG_MUX: run the Timestamp Generator inside the Multiplexer state machine, which will provide microsecond accurate timestamps to the events passing through.

4.5.2.31 DYNAPSE_CONFIG_POISSONSPIKEGEN

```
#define DYNAPSE_CONFIG_POISSONSPIKEGEN 18
```

Module address: Device side poisson generator configuration Provides run/stop control of poisson spike generation and rate setting for 1024 sources.

4.5.2.32 DYNAPSE_CONFIG_POISSONSPIKEGEN_CHIPID

```
#define DYNAPSE_CONFIG_POISSONSPIKEGEN_CHIPID 3
```

Parameter address for module DYNAPSE_CONFIG_POISSONSPIKEGEN: Chip ID of the chip that will receive events generated by the poisson spike generator.

4.5.2.33 DYNAPSE_CONFIG_POISSONSPIKEGEN_RUN

```
#define DYNAPSE_CONFIG_POISSONSPIKEGEN_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_POISSONSPIKEGEN: Enables or disables generation of poisson spike trains.

4.5.2.34 DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEADDRESS

```
#define DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEADDRESS 1
```

Parameter address for module DYNAPSE_CONFIG_POISSONSPIKEGEN: Selects the address of a poisson spike train source. Writing to this parameter will apply the rate previously written to the WRITEDATA field.

4.5.2.35 DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEDATA

```
#define DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEDATA 2
```

Parameter address for module DYNAPSE_CONFIG_POISSONSPIKEGEN: Holds data that will be written to the address specified by WRITEADDRESS.

4.5.2.36 DYNAPSE_CONFIG_SPIKEGEN

```
#define DYNAPSE_CONFIG_SPIKEGEN 16
```

Module address: Device side spike generator module configuration. Provides start/stop control of spike train application and selection of fixed/variable inter-spike intervals and their location in memory.

4.5.2.37 DYNAPSE_CONFIG_SPIKEGEN_BASEADDR

```
#define DYNAPSE_CONFIG_SPIKEGEN_BASEADDR 2
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Sets the start address of a spike train in memory.

4.5.2.38 DYNAPSE_CONFIG_SPIKEGEN_ISI

```
#define DYNAPSE_CONFIG_SPIKEGEN_ISI 4
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Sets the inter-spike interval that will be used in fixed ISI mode (VARMODE false).

4.5.2.39 DYNAPSE_CONFIG_SPIKEGEN_ISIBASE

```
#define DYNAPSE_CONFIG_SPIKEGEN_ISIBASE 5
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Sets the time base resolution for inter-spike intervals as the number of FPGA clock cycles.

4.5.2.40 DYNAPSE_CONFIG_SPIKEGEN_REPEAT

```
#define DYNAPSE_CONFIG_SPIKEGEN_REPEAT 6
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Sets repeat mode to true or false.

4.5.2.41 DYNAPSE_CONFIG_SPIKEGEN_RUN

```
#define DYNAPSE_CONFIG_SPIKEGEN_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Instructs the spike generator to start applying the configured spike train when the parameter changes from false to true.

4.5.2.42 DYNAPSE_CONFIG_SPIKEGEN_STIMCOUNT

```
#define DYNAPSE_CONFIG_SPIKEGEN_STIMCOUNT 3
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Sets the number of events to read from memory for a single application of a spike train.

4.5.2.43 DYNAPSE_CONFIG_SPIKEGEN_VARMODE

```
#define DYNAPSE_CONFIG_SPIKEGEN_VARMODE 1
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Selects variable inter-spike interval mode (true) or fixed inter-spike interval mode (false).

4.5.2.44 DYNAPSE_CONFIG_SRAM

```
#define DYNAPSE_CONFIG_SRAM 14
```

Module address: Device side SRAM controller configuration. The module holds an address, a word to be written to SRAM, the most recent word read using a read command, and a read/write command. Reads/writes are triggered when the address field is changed. Example: `caerDynapseWriteSramWords(devHandle, SRAMData, baseAddr, numWords);` Writes `numWords` words from array `SRAMData` to the SRAM, starting at `baseAddr`. This define is for internal use of [caerDynapseWriteSramWords\(\)](#); it can be used on its own, but we recommend using the above function that hides all the internal details of writing to the FPGA SRAM.

4.5.2.45 DYNAPSE_CONFIG_SRAM_ADDRESS

```
#define DYNAPSE_CONFIG_SRAM_ADDRESS 1
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Holds the address that will be used for the next read/write. Writing or reading this field will trigger the command contained in the command register to be executed on the FPGA.

4.5.2.46 DYNAPSE_CONFIG_SRAM_BURSTMODE

```
#define DYNAPSE_CONFIG_SRAM_BURSTMODE 5
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Burst mode enable for fast writing. Disables updates on address change and instead updates on data change, while automatically incrementing the writing address. Two 16-bit words are written per 32-bit word sent to the SPI controller starting with the least significant half word.

4.5.2.47 DYNAPSE_CONFIG_SRAM_DIRECTION_POS

```
#define DYNAPSE_CONFIG_SRAM_DIRECTION_POS 0
```

On-chip SRAM for spike routing.

4.5.2.48 DYNAPSE_CONFIG_SRAM_READ

```
#define DYNAPSE_CONFIG_SRAM_READ 0
```

Command for module DYNAPSE_CONFIG_SRAM: Read command for the RWCOMMAND field. Example: caer↵ DeviceConfigSet(devHandle, DYNAPSE_CONFIG_SRAM, DYNAPSE_CONFIG_SRAM_RWCOMMAND, DYNA↵ PSE_CONFIG_SRAM_READ); Sets the SRAM controller up for doing reads.

4.5.2.49 DYNAPSE_CONFIG_SRAM_READDATA

```
#define DYNAPSE_CONFIG_SRAM_READDATA 2
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Holds the most recently read data from the SRAM. Read-only parameter.

4.5.2.50 DYNAPSE_CONFIG_SRAM_RWCOMMAND

```
#define DYNAPSE_CONFIG_SRAM_RWCOMMAND 4
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Holds the command that will be executed when the address field is written to. Example: caerDeviceConfigSet(devHandle, DYNAPSE_CONFIG_SRAM, DYNAPSE↵ _CONFIG_SRAM_RWCOMMAND, DYNAPSE_CONFIG_SRAM_WRITE); Sets the SRAM controller up for doing writes. DYNAPSE_CONFIG_SRAM_READ and DYNAPSE_CONFIG_SRAM_WRITE are supported.

4.5.2.51 DYNAPSE_CONFIG_SRAM_WRITE

```
#define DYNAPSE_CONFIG_SRAM_WRITE 1
```

Command for module DYNAPSE_CONFIG_SRAM: Write command for the RWCOMMAND field. Example: caer↵ DeviceConfigSet(devHandle, DYNAPSE_CONFIG_SRAM, DYNAPSE_CONFIG_SRAM_RWCOMMAND, DYNA↵ PSE_CONFIG_SRAM_WRITE); Sets the SRAM controller up for doing writes.

4.5.2.52 DYNAPSE_CONFIG_SRAM_WRITEDATA

```
#define DYNAPSE_CONFIG_SRAM_WRITEDATA 3
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Holds the data that will be written on the next write. Example: caerDeviceConfigSet(devHandle, DYNAPSE_CONFIG_SRAM, DYNAPSE_CONFIG_SRAM_WRITE↵ DATA, wData); caerDeviceConfigSet(devHandle, DYNAPSE_CONFIG_SRAM, DYNAPSE_CONFIG_SRAM_R↵ WCOMMAND, DYNAPSE_CONFIG_SRAM_WRITE); caerDeviceConfigSet(devHandle, DYNAPSE_CONFIG_S↵ RAM, DYNAPSE_CONFIG_SRAM_ADDRESS, wAddr); Writes wData to the address specified by wAddr.

4.5.2.53 DYNAPSE_CONFIG_SYNAPSERECONFIG

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG 15
```

Module address: Device side Synapse Reconfiguration module configuration. Provides run control, selection between using a single kernel for all neurons and reading per-neuron kernels from SRAM, programming of the global kernel, as well as target output chip ID selection and SRAM kernel table base address.

4.5.2.54 DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPSELECT

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPSELECT 3
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG: Select which chip outputs should go to.

4.5.2.55 DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOBAKERNEL

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOBAKERNEL 1
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG: Bits 16 down to 12 select the address in the global kernel table and bits 11 down to 0 specify the data. The 12 data bits are split into 4*3 synaptic weight bits which map onto positive/negative polarity events from 2 DVS pixels.

4.5.2.56 DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG: Run control. Starts and stops handshaking with DVS.

4.5.2.57 DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAMBASEADDR

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAMBASEADDR 4
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG: SRAM base address configuration in increments of 32 Kib. Setting this to N will place the SRAM kernel LUT in the range $[N \cdot 2^{15}, ((N+1) \cdot 2^{15}) - 1]$

4.5.2.58 DYNAPSE_CONFIG_SYNAPSERECONFIG_USESRAMKERNELS

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_USESRAMKERNELS 2
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG: Boolean parameter for selecting between using kernels stored in SRAM or the global kernel table. 1 for SRAM, 0 for global kernel table.

4.5.2.59 DYNAPSE_CONFIG_SYSINFO

```
#define DYNAPSE_CONFIG_SYSINFO 6
```

Module address: device-side system information. The system information module provides various details on the device, such as currently installed logic revision or clock speeds. All its parameters are read-only. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_dynapse_info](#)' documentation for more details on what information is available.

4.5.2.60 DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER

```
#define DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER 1
```

Parameter address for module DYNAPSE_CONFIG_SYSINFO: read-only parameter, an integer used to identify the different types of sensor chips used on the device. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_dynapse_info](#)' documentation to get this information.

4.5.2.61 DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER

```
#define DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER 2
```

Parameter address for module DYNAPSE_CONFIG_SYSINFO: read-only parameter, whether the device is currently a timestamp master or slave when synchronizing multiple devices together. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_dynapse_info](#)' documentation to get this information.

4.5.2.62 DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK

```
#define DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK 3
```

Parameter address for module DYNAPSE_CONFIG_SYSINFO: read-only parameter, the frequency in MHz at which the main FPGA/CPLD logic is running. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_dynapse_info](#)' documentation to get this information.

4.5.2.63 DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION

```
#define DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION 0
```

Parameter address for module DYNAPSE_CONFIG_SYSINFO: read-only parameter, the version of the logic currently running on the device's FPGA/CPLD. This is reserved for internal use and should not be used by anything other than libcaer. Please see the 'struct [caer_dynapse_info](#)' documentation to get this information.

4.5.2.64 DYNAPSE_CONFIG_TAU1_RESET

```
#define DYNAPSE_CONFIG_TAU1_RESET 19
```

Reset all neurons of a core to use the TAU1 neuron leak bias. 'paramAddr' takes the core ID to be reset, other arguments are unused. Remember to select the chip you want to configure before this!

4.5.2.65 DYNAPSE_CONFIG_TAU2_RESET

```
#define DYNAPSE_CONFIG_TAU2_RESET 20
```

Reset all neurons of a core to use the TAU2 neuron leak bias. 'paramAddr' takes the core ID to be reset, other arguments are unused. Remember to select the chip you want to configure before this!

4.5.2.66 DYNAPSE_CONFIG_TAU2_SET

```
#define DYNAPSE_CONFIG_TAU2_SET 17
```

Set certain neurons of a core to use the TAU2 neuron leak bias. By default neurons use the TAU1 neuron leak bias. You can also use DYNAPSE_CONFIG_TAU1_RESET and DYNAPSE_CONFIG_TAU2_RESET to reset all neurons in a core to the same bias. 'paramAddr' takes the core ID to be set, 'param' the neuron ID. Remember to select the chip you want to configure before this!

4.5.2.67 DYNAPSE_CONFIG_USB

```
#define DYNAPSE_CONFIG_USB 9
```

Module address: device-side USB output configuration. The USB output module forwards the data from the device and the FPGA/CPLD to the USB chip, usually a Cypress FX2 or FX3.

4.5.2.68 DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY

```
#define DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY 1
```

Parameter address for module DYNAPSE_CONFIG_USB: the time delay after which a packet of data is committed to USB, even if it is not full yet (short USB packet). The value is in 125 μ s time-slices, corresponding to how USB schedules its operations (a value of 4 for example would mean waiting at most 0.5ms until sending a short USB packet to the host).

4.5.2.69 DYNAPSE_CONFIG_USB_RUN

```
#define DYNAPSE_CONFIG_USB_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_USB: enable the USB FIFO module, which transfers the data from the FPGA/CPLD to the USB chip, to be then sent to the host. Turning this off will suppress any USB data communication!

4.5.2.70 DYNAPSE_X4BOARD_COREX

```
#define DYNAPSE_X4BOARD_COREX 4
```

Number of cores in the x direction of the board.

4.5.2.71 DYNAPSE_X4BOARD_COREY

```
#define DYNAPSE_X4BOARD_COREY 4
```

Number of cores in the y direction of the board.

4.5.2.72 DYNAPSE_X4BOARD_NEUX

```
#define DYNAPSE_X4BOARD_NEUX 64
```

Number of neurons in the x direction of the board.

4.5.2.73 DYNAPSE_X4BOARD_NEUY

```
#define DYNAPSE_X4BOARD_NEUY 64
```

Number of neurons in the y direction of the board.

4.5.2.74 DYNAPSE_X4BOARD_NUMCHIPS

```
#define DYNAPSE_X4BOARD_NUMCHIPS 4
```

Number of chips on the board.

4.5.3 Function Documentation

4.5.3.1 caerBiasDynapseGenerate()

```
uint32_t caerBiasDynapseGenerate (
    const struct caer_bias_dynapse dynapseBias )
```

Transform coarse-fine bias structure into internal integer representation, suited for sending directly to the device via [caerDeviceConfigSet\(\)](#).

Parameters

<i>dynapseBias</i>	coarse-fine bias structure.
--------------------	-----------------------------

Returns

internal integer representation for device configuration.

4.5.3.2 caerBiasDynapseParse()

```
struct caer_bias_dynapse caerBiasDynapseParse (
    const uint32_t dynapseBias )
```

Transform internal integer representation, as received by calls to [caerDeviceConfigGet\(\)](#), into a coarse-fine bias structure, for easier handling and understanding of the various parameters.

Parameters

<i>dynapseBias</i>	internal integer representation from device.
--------------------	--

Returns

coarse-fine bias structure.

4.5.3.3 caerDynapseCoreAddrToNeuronId()

```
uint16_t caerDynapseCoreAddrToNeuronId (
    uint8_t coreId,
    uint8_t neuronAddrCore )
```

Map core ID and per-core neuron address to the correct chip global neuron address.

Parameters

<i>coreId</i>	the chip's core ID, range [0,3].
<i>neuronAddrCore</i>	the neuron's address within this core, range [0,255].

Returns

chip global neuron address.

4.5.3.4 caerDynapseCoreXYToNeuronId()

```
uint16_t caerDynapseCoreXYToNeuronId (
    uint8_t coreId,
    uint8_t columnX,
    uint8_t rowY )
```

Map core ID and column/row address to the correct chip global neuron address.

Parameters

<i>coreId</i>	the chip's core ID, range [0,3].
<i>columnX</i>	the neuron's column address, range [0,15].
<i>rowY</i>	the neuron's row address, range [0,15].

Returns

chip global neuron address.

4.5.3.5 caerDynapseGenerateCamBits()

```
uint32_t caerDynapseGenerateCamBits (
    uint16_t inputNeuronAddr,
    uint16_t neuronAddr,
    uint8_t camId,
    uint8_t synapseType )
```

Generate bits to write a single CAM, to specify which spikes are allowed as input into a neuron.

Parameters

<i>inputNeuronAddr</i>	the neuron address that should be let in as input to this neuron, range [0,1023] (use caerDynapseCoreXYToNeuronId() for a 2D mapping).
<i>neuronAddr</i>	the neuron to program, range [0,1023] (use caerDynapseCoreXYToNeuronId() for a 2D mapping).
<i>camId</i>	CAM address (synapse), each neuron has 64, range [0,63].
<i>synapseType</i>	one of the four possible synaptic weights: [DYNAPSE_CONFIG_CAMTYPE_F_EXC,DYNAPSE_CONFIG_CAMTYPE_S_EXC,DYNAPSE_CONFIG_CAMTYPE_F_INH,DYNAPSE_CONFIG_CAMTYPE_S_INH].

Returns

bits to send to device.

4.5.3.6 caerDynapseGenerateSramBits()

```
uint32_t caerDynapseGenerateSramBits (
    uint16_t neuronAddr,
    uint8_t sramId,
    uint8_t virtualCoreId,
    bool sx,
    uint8_t dx,
    bool sy,
    uint8_t dy,
    uint8_t destinationCore )
```

Generate bits to write one of the 4 SRAMs of a single neuron. Writing the SRAM means writing the destination address of where the spikes will be routed to. This works on the on-chip SRAM!

Parameters

<i>neuronAddr</i>	the neuron to program, range [0,1023] (use caerDynapseCoreXYToNeuronId() for a 2D mapping).
<i>sramId</i>	SRAM address (one of four cells), range [0,3].
<i>virtualCoreId</i>	fake source core ID, set it to this value instead of the actual source core ID, range [0,3].
<i>sx</i>	X direction, can be one of: [DYNAPSE_CONFIG_SRAM_DIRECTION_X_EAST,DYNAPSE_CONFIG_SRAM_DIRECTION_X_WEST].
<i>dx</i>	X delta, number of chips to jumps before reaching destination, range is [0,3].
<i>sy</i>	Y direction, can be one of: [DYNAPSE_CONFIG_SRAM_DIRECTION_Y_NORTH,DYNAPSE_CONFIG_SRAM_DIRECTION_Y_SOUTH].
<i>dy</i>	Y delta, number of chips to jumps before reaching destination, range is [0,3].
<i>destinationCore</i>	spike destination core, uses one-hot coding for the 4 cores: [C3,C2,C1,C0] -> [0,0,0,0] (0 decimal) no core, [1,1,1,1] (15 decimal) all cores

Returns

bits to send to device.

4.5.3.7 caerDynapseInfoGet()

```
struct caer_dynapse_info caerDynapseInfoGet (
    caerDeviceHandle handle )
```

Return basic information on the device, such as its ID, the logic version, and so on. See the 'struct [caer_dynapse_info](#)' documentation for more details.

Parameters

<i>handle</i>	a valid device handle.
---------------	------------------------

Returns

a copy of the device information structure if successful, an empty structure (all zeros) on failure.

4.5.3.8 caerDynapseSendDataToUSB()

```
bool caerDynapseSendDataToUSB (
    caerDeviceHandle handle,
    const uint32_t * data,
    size_t numConfig )
```

Send array of configuration parameters to the device via USB.

Remember to select the chip you want to configure before calling this function!

Parameters

<i>handle</i>	a valid device handle.
<i>data</i>	an array of integers holding configuration data.
<i>numConfig</i>	number of configuration parameters to send.

Returns

true on success, false otherwise.

4.5.3.9 caerDynapseSpikeEventFromXY()

```
struct caer_spike_event caerDynapseSpikeEventFromXY (
    uint16_t x,
    uint16_t y )
```

Get the chip ID, core ID and neuron ID from the X and Y coordinates. This is the reverse transform to: [caerDynapseSpikeEventGetX\(\)](#) / [caerDynapseSpikeEventGetY\(\)](#). The return value is a 'struct caer_spike_event' because it already has functions to get/set all the needed values.

Parameters

<i>x</i>	a X coordinate as returned by caerDynapseSpikeEventGetX() .
<i>y</i>	a Y coordinate as returned by caerDynapseSpikeEventGetY() .

Returns

a SpikeEvent struct holding chip ID, core ID and neuron ID.

4.5.3.10 caerDynapseSpikeEventGetX()

```
uint16_t caerDynapseSpikeEventGetX (
    caerSpikeEventConst event )
```

Get the X (column) address for a spike event, in pixels. The (0, 0) address is in the upper left corner.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
--------------	---

Returns

the event X address in pixels.

4.5.3.11 caerDynapseSpikeEventGetY()

```
uint16_t caerDynapseSpikeEventGetY (
    caerSpikeEventConst event )
```

Get the Y (row) address for a spike event, in pixels. The (0, 0) address is in the upper left corner.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
--------------	---

Returns

the event Y address in pixels.

4.5.3.12 caerDynapseWriteCam()

```
bool caerDynapseWriteCam (
    caerDeviceHandle handle,
    uint16_t inputNeuronAddr,
    uint16_t neuronAddr,
    uint8_t camId,
    uint8_t synapseType )
```

Write a single CAM, to specify which spikes are allowed as input into a neuron.

Remember to select the chip you want to configure before calling this function!

Parameters

<i>handle</i>	a valid device handle.
<i>inputNeuronAddr</i>	the neuron address that should be let in as input to this neuron, range [0,1023].
<i>neuronAddr</i>	the neuron address whose CAM should be programmed, range [0,1023].
<i>camId</i>	CAM address (synapse), each neuron has 64, range [0,63].
<i>synapseType</i>	one of the four possible synaptic weights: [DYNAPSE_CONFIG_CAMTYPE_F_EXC,DYNAPSE_CONFIG_CAMTYPE_S_EXC,DY↔ NAPSE_CONFIG_CAMTYPE_F_INH,DYNAPSE_CONFIG_CAMTYPE_S_INH].

Returns

true on success, false otherwise.

4.5.3.13 caerDynapseWritePoissonSpikeRate()

```
bool caerDynapseWritePoissonSpikeRate (
    caerDeviceHandle handle,
    uint16_t neuronAddr,
    float rateHz )
```

Specifies the poisson spike generator's spike rate.

Parameters

<i>handle</i>	a valid device handle.
<i>neuronAddr</i>	The target neuron of the poisson spike train, range [0,1023].
<i>rateHz</i>	The rate in Hz of the spike train, this will be quantized to the nearest supported level, range [0,4300].

Returns

true on success, false otherwise.

4.5.3.14 caerDynapseWriteSram()

```
bool caerDynapseWriteSram (
    caerDeviceHandle handle,
    uint8_t coreId,
    uint8_t neuronAddrCore,
    uint8_t virtualCoreId,
    bool sx,
    uint8_t dx,
    bool sy,
    uint8_t dy,
    uint8_t sramId,
    uint8_t destinationCore )
```

THIS FUNCTION IS DEPRECATED. USE [caerDynapseWriteSramN\(\)](#) INSTEAD! The new function uses the global neuron ID (range [0,1023]) like all others, instead of the separate core ID/neuron ID syntax. Also the arguments are in the same order as [caerDynapseGenerateSramBits\(\)](#), in particular the 'sramId' comes right after 'neuronId'.

Write one of the 4 SRAMs of a single neuron. Writing the SRAM means writing the destination address of where the spikes will be routed to. This works on the on-chip SRAM!

Remember to select the chip you want to configure before calling this function!

Parameters

<i>handle</i>	a valid device handle.
<i>coreId</i>	the chip's core ID, range [0,3].
<i>neuronAddrCore</i>	the neuron's address within this core, range [0,255].
<i>virtualCoreId</i>	fake source core ID, set it to this value instead of the actual source core ID, range [0,3].
<i>sx</i>	X direction, can be one of: [DYNAPSE_CONFIG_SRAM_DIRECTION_X_EAST,DYNAPSE_CONFIG_SRAM_DIRECTION_X_WEST].
<i>dx</i>	X delta, number of chips to jumps before reaching destination, range is [0,3].
<i>sy</i>	Y direction, can be one of: [DYNAPSE_CONFIG_SRAM_DIRECTION_Y_NORTH,DYNAPSE_CONFIG_SRAM_DIRECTION_Y_SOUTH].
<i>dy</i>	Y delta, number of chips to jumps before reaching destination, range is [0,3].
<i>sramId</i>	SRAM address (one of four cells), range [0,3].
<i>destinationCore</i>	spike destination core, uses one-hot coding for the 4 cores: [C3,C2,C1,C0] -> [0,0,0,0] (0 decimal) no core, [1,1,1,1] (15 decimal) all cores

Returns

true on success, false otherwise.

4.5.3.15 caerDynapseWriteSramN()

```
bool caerDynapseWriteSramN (
    caerDeviceHandle handle,
    uint16_t neuronAddr,
    uint8_t sramId,
    uint8_t virtualCoreId,
    bool sx,
    uint8_t dx,
    bool sy,
    uint8_t dy,
    uint8_t destinationCore )
```

Write one of the 4 SRAMs of a single neuron. Writing the SRAM means writing the destination address of where the spikes will be routed to. This works on the on-chip SRAM!

Remember to select the chip you want to configure before calling this function!

Parameters

<i>handle</i>	a valid device handle.
<i>neuronAddr</i>	the neuron to program, range [0,1023] (use caerDynapseCoreXYToNeuronId() for a 2D mapping).
<i>sramId</i>	SRAM address (one of four cells), range [0,3].
<i>virtualCoreId</i>	fake source core ID, set it to this value instead of the actual source core ID, range [0,3].
<i>sx</i>	X direction, can be one of: [DYNAPSE_CONFIG_SRAM_DIRECTION_X_EAST,DYNAPSE_CONFIG_SRAM_DIRECTION_X_WEST].
<i>dx</i>	X delta, number of chips to jumps before reaching destination, range is [0,3].
<i>sy</i>	Y direction, can be one of: [DYNAPSE_CONFIG_SRAM_DIRECTION_Y_NORTH,DYNAPSE_CONFIG_SRAM_DIRECTION_Y_SOUTH].
<i>dy</i>	Y delta, number of chips to jumps before reaching destination, range is [0,3].
<i>destinationCore</i>	spike destination core, uses one-hot coding for the 4 cores: [C3,C2,C1,C0] -> [0,0,0,0] (0 decimal) no core, [1,1,1,1] (15 decimal) all cores

Returns

true on success, false otherwise.

4.5.3.16 caerDynapseWriteSramWords()

```
bool caerDynapseWriteSramWords (
    caerDeviceHandle handle,
    const uint16_t * data,
    uint32_t baseAddr,
    size_t numWords )
```

Transfer 16bit words from memory to device SRAM, with configurable starting address and number of words. This works on the FPGA SRAM!

Parameters

<i>handle</i>	a valid device handle.
<i>data</i>	array from which to read data to send to SRAM.
<i>baseAddr</i>	SRAM start address where to put the data.
<i>numWords</i>	number of 16bit words to transfer.

Returns

true on success, false otherwise.

4.6 devices/edvs.h File Reference

```
#include "../events/polarity.h"
#include "../events/special.h"
#include "serial.h"
```

Data Structures

- struct [caer_edvs_info](#)

Macros

- #define [CAER_DEVICE_EDVS](#) 5
- #define [EDVS_CONFIG_DVS](#) 0
- #define [EDVS_CONFIG_BIAS](#) 1
- #define [EDVS_CONFIG_DVS_RUN](#) 0
- #define [EDVS_CONFIG_DVS_TIMESTAMP_RESET](#) 1
- #define [EDVS_CONFIG_BIAS_CAS](#) 0
- #define [EDVS_CONFIG_BIAS_INJGND](#) 1
- #define [EDVS_CONFIG_BIAS_REQPD](#) 2
- #define [EDVS_CONFIG_BIAS_PUX](#) 3
- #define [EDVS_CONFIG_BIAS_DIFFOFF](#) 4
- #define [EDVS_CONFIG_BIAS_REQ](#) 5
- #define [EDVS_CONFIG_BIAS_REFR](#) 6
- #define [EDVS_CONFIG_BIAS_PUY](#) 7
- #define [EDVS_CONFIG_BIAS_DIFFON](#) 8
- #define [EDVS_CONFIG_BIAS_DIFF](#) 9
- #define [EDVS_CONFIG_BIAS_FOLL](#) 10
- #define [EDVS_CONFIG_BIAS_PR](#) 11

Functions

- struct [caer_edvs_info](#) [caerEDVSInfoGet](#) ([caerDeviceHandle](#) handle)

4.6.1 Detailed Description

EDVS-4337 specific configuration defines and information structures.

4.6.2 Macro Definition Documentation

4.6.2.1 CAER_DEVICE_EDVS

```
#define CAER_DEVICE_EDVS 5
```

Device type definition for iniVation EDVS-4337.

4.6.2.2 EDVS_CONFIG_BIAS

```
#define EDVS_CONFIG_BIAS 1
```

Module address: device-side chip bias generator configuration.

4.6.2.3 EDVS_CONFIG_BIAS_CAS

```
#define EDVS_CONFIG_BIAS_CAS 0
```

Parameter address for module EDVS_CONFIG_BIAS: First stage amplifier cascode bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.6.2.4 EDVS_CONFIG_BIAS_DIFF

```
#define EDVS_CONFIG_BIAS_DIFF 9
```

Parameter address for module EDVS_CONFIG_BIAS: Differential (second stage amplifier) bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.6.2.5 EDVS_CONFIG_BIAS_DIFFOFF

```
#define EDVS_CONFIG_BIAS_DIFFOFF 4
```

Parameter address for module EDVS_CONFIG_BIAS: Off events threshold bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.6.2.6 EDVS_CONFIG_BIAS_DIFFON

```
#define EDVS_CONFIG_BIAS_DIFFON 8
```

Parameter address for module EDVS_CONFIG_BIAS: On events threshold bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.6.2.7 EDVS_CONFIG_BIAS_FOLL

```
#define EDVS_CONFIG_BIAS_FOLL 10
```

Parameter address for module EDVS_CONFIG_BIAS: Source follower bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.6.2.8 EDVS_CONFIG_BIAS_INJGND

```
#define EDVS_CONFIG_BIAS_INJGND 1
```

Parameter address for module EDVS_CONFIG_BIAS: Injected ground bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.6.2.9 EDVS_CONFIG_BIAS_PR

```
#define EDVS_CONFIG_BIAS_PR 11
```

Parameter address for module EDVS_CONFIG_BIAS: Photoreceptor bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.6.2.10 EDVS_CONFIG_BIAS_PUX

```
#define EDVS_CONFIG_BIAS_PUX 3
```

Parameter address for module EDVS_CONFIG_BIAS: Pull up on request from X arbiter (AER). See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.6.2.11 EDVS_CONFIG_BIAS_PUY

```
#define EDVS_CONFIG_BIAS_PUY 7
```

Parameter address for module EDVS_CONFIG_BIAS: Pull up on request from Y arbiter (AER). See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.6.2.12 EDVS_CONFIG_BIAS_REFR

```
#define EDVS_CONFIG_BIAS_REFR 6
```

Parameter address for module EDVS_CONFIG_BIAS: Refractory period bias. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.6.2.13 EDVS_CONFIG_BIAS_REQ

```
#define EDVS_CONFIG_BIAS_REQ 5
```

Parameter address for module EDVS_CONFIG_BIAS: Pull down for passive load inverters in digital AER pixel circuitry. See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.6.2.14 EDVS_CONFIG_BIAS_REQPD

```
#define EDVS_CONFIG_BIAS_REQPD 2
```

Parameter address for module EDVS_CONFIG_BIAS: Pull down on chip request (AER). See '<https://inivation.com/support/hardware/biasing/>' for more details.

4.6.2.15 EDVS_CONFIG_DVS

```
#define EDVS_CONFIG_DVS 0
```

Module address: device-side DVS configuration.

4.6.2.16 EDVS_CONFIG_DVS_RUN

```
#define EDVS_CONFIG_DVS_RUN 0
```

Parameter address for module EDVS_CONFIG_DVS: run the DVS chip and generate polarity event data.

4.6.2.17 EDVS_CONFIG_DVS_TIMESTAMP_RESET

```
#define EDVS_CONFIG_DVS_TIMESTAMP_RESET 1
```

Parameter address for module EDVS_CONFIG_DVS: reset the time-stamp counter of the device. This is a temporary configuration switch and will reset itself right away.

4.6.3 Function Documentation

4.6.3.1 caerEDVSInfoGet()

```
struct caer_edvs_info caerEDVSInfoGet (
    caerDeviceHandle handle )
```

Return basic information on the device, such as its ID, its resolution, the logic version, and so on. See the 'struct [caer_edvs_info](#)' documentation for more details.

Parameters

<i>handle</i>	a valid device handle.
---------------	------------------------

Returns

a copy of the device information structure if successful, an empty structure (all zeros) on failure.

4.7 devices/serial.h File Reference

```
#include "device.h"
```

Macros

- `#define CAER_HOST_CONFIG_SERIAL -1`
- `#define CAER_HOST_CONFIG_SERIAL_READ_SIZE 0`
- `#define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_2M 2000000`
- `#define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_4M 4000000`
- `#define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_8M 8000000`
- `#define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_12M 12000000`

Functions

- `caerDeviceHandle caerDeviceOpenSerial` (uint16_t deviceId, uint16_t deviceType, const char *serialPort↔
Name, uint32_t serialBaudRate)

4.7.1 Detailed Description

Common functions to access, configure and exchange data with supported serial port devices. Also contains defines for serial port specific configuration options.

4.7.2 Macro Definition Documentation

4.7.2.1 CAER_HOST_CONFIG_SERIAL

```
#define CAER_HOST_CONFIG_SERIAL -1
```

Module address: host-side serial port configuration.

4.7.2.2 CAER_HOST_CONFIG_SERIAL_BAUD_RATE_12M

```
#define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_12M 12000000
```

Parameter values for module CAER_HOST_CONFIG_SERIAL: possible baud-rates for serial port communication.

4.7.2.3 CAER_HOST_CONFIG_SERIAL_BAUD_RATE_2M

```
#define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_2M 2000000
```

Parameter values for module CAER_HOST_CONFIG_SERIAL: possible baud-rates for serial port communication.

4.7.2.4 CAER_HOST_CONFIG_SERIAL_BAUD_RATE_4M

```
#define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_4M 4000000
```

Parameter values for module CAER_HOST_CONFIG_SERIAL: possible baud-rates for serial port communication.

4.7.2.5 CAER_HOST_CONFIG_SERIAL_BAUD_RATE_8M

```
#define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_8M 8000000
```

Parameter values for module CAER_HOST_CONFIG_SERIAL: possible baud-rates for serial port communication.

4.7.2.6 CAER_HOST_CONFIG_SERIAL_READ_SIZE

```
#define CAER_HOST_CONFIG_SERIAL_READ_SIZE 0
```

Parameter address for module CAER_HOST_CONFIG_SERIAL: read size for serial port communication.

4.7.3 Function Documentation

4.7.3.1 caerDeviceOpenSerial()

```
caerDeviceHandle caerDeviceOpenSerial (
    uint16_t deviceID,
    uint16_t deviceType,
    const char * serialPortName,
    uint32_t serialBaudRate )
```

Open a specified serial port device, assign an ID to it and return a handle for further usage. Various means can be employed to limit the selection of the device.

Parameters

<i>deviceID</i>	a unique ID to identify the device from others. Will be used as the source for EventPackets being generated from its data.
<i>deviceType</i>	type of the device to open. Currently supported are: CAER_DEVICE_EDVS
<i>serialPortName</i>	name of the serial port device to open.
<i>serialBaudRate</i>	baud-rate for serial port communication.

Returns

a valid device handle that can be used with the other libcaer functions, or NULL on error. Always check for this! On error, errno is also set to provide more precise information about the failure cause.

4.8 devices/usb.h File Reference

```
#include "device.h"
```

Macros

- `#define CAER_HOST_CONFIG_USB -1`
- `#define CAER_HOST_CONFIG_USB_BUFFER_NUMBER 0`
- `#define CAER_HOST_CONFIG_USB_BUFFER_SIZE 1`

Functions

- `caerDeviceHandle caerDeviceOpen` (uint16_t deviceID, uint16_t deviceType, uint8_t busNumberRestrict, uint8_t devAddressRestrict, const char *serialNumberRestrict)

4.8.1 Detailed Description

Common functions to access, configure and exchange data with supported USB devices. Also contains defines for USB specific configuration options.

4.8.2 Macro Definition Documentation

4.8.2.1 CAER_HOST_CONFIG_USB

```
#define CAER_HOST_CONFIG_USB -1
```

Module address: host-side USB configuration.

4.8.2.2 CAER_HOST_CONFIG_USB_BUFFER_NUMBER

```
#define CAER_HOST_CONFIG_USB_BUFFER_NUMBER 0
```

Parameter address for module CAER_HOST_CONFIG_USB: set number of buffers used by libusb for asynchronous data transfers with the USB device. The default values are usually fine, only change them if you're running into I/O limits.

4.8.2.3 CAER_HOST_CONFIG_USB_BUFFER_SIZE

```
#define CAER_HOST_CONFIG_USB_BUFFER_SIZE 1
```

Parameter address for module CAER_HOST_CONFIG_USB: set size of each buffer used by libusb for asynchronous data transfers with the USB device. The default values are usually fine, only change them if you're running into I/O limits.

4.8.3 Function Documentation

4.8.3.1 caerDeviceOpen()

```
caerDeviceHandle caerDeviceOpen (
    uint16_t deviceID,
    uint16_t deviceType,
    uint8_t busNumberRestrict,
    uint8_t devAddressRestrict,
    const char * serialNumberRestrict )
```

Open a specified USB device, assign an ID to it and return a handle for further usage. Various means can be employed to limit the selection of the device.

Parameters

<i>deviceID</i>	a unique ID to identify the device from others. Will be used as the source for EventPackets being generated from its data.
<i>deviceType</i>	type of the device to open. Currently supported are: CAER_DEVICE_DVS128, CAER_DEVICE_DAVIS, CAER_DEVICE_DYNAPSE
<i>busNumberRestrict</i>	restrict the search for viable devices to only this USB bus number.
<i>devAddressRestrict</i>	restrict the search for viable devices to only this USB device address.
<i>serialNumberRestrict</i>	restrict the search for viable devices to only devices which do possess the given Serial Number in their USB SerialNumber descriptor.

Returns

a valid device handle that can be used with the other libcaer functions, or NULL on error. Always check for this! On error, errno is also set to provide more precise information about the failure cause.

4.9 events/common.h File Reference

```
#include "../libcaer.h"
```

Macros

- `#define caerLogEHO caerLog`

- `#define TS_OVERFLOW_SHIFT 31`
 - `#define CAER_DEFAULT_EVENT_TYPES_COUNT 14`
 - `#define CAER_EVENT_PACKET_HEADER_SIZE 28`
 - `#define CAER_ITERATOR_ALL_START(PACKET_HEADER, EVENT_TYPE)`
 - `#define CAER_ITERATOR_ALL_END }`
 - `#define CAER_ITERATOR_VALID_START(PACKET_HEADER, EVENT_TYPE)`
 - `#define CAER_ITERATOR_VALID_END }`
-
- `#define VALID_MARK_SHIFT 0`
 - `#define VALID_MARK_MASK 0x00000001`

Typedefs

- `typedef struct caer_event_packet_header * caerEventPacketHeader`
- `typedef const struct caer_event_packet_header * caerEventPacketHeaderConst`

Enumerations

- `enum caer_default_event_types {`
`SPECIAL_EVENT = 0, POLARITY_EVENT = 1, FRAME_EVENT = 2, IMU6_EVENT = 3,`
`IMU9_EVENT = 4, SAMPLE_EVENT = 5, EAR_EVENT = 6, CONFIG_EVENT = 7,`
`POINT1D_EVENT = 8, POINT2D_EVENT = 9, POINT3D_EVENT = 10, POINT4D_EVENT = 11,`
`SPIKE_EVENT = 12, MATRIX4x4_EVENT = 13 }`

Functions

- `PACKED_STRUCT` (struct caer_event_packet_header { int16_t eventType;int16_t eventSource;int32_t eventSize;int32_t eventTSOffset;int32_t eventTSOverflow;int32_t eventCapacity;int32_t eventNumber;int32_t eventValid;})
- `static int16_t caerEventPacketHeaderGetEventType (caerEventPacketHeaderConst header)`
- `static void caerEventPacketHeaderSetEventType (caerEventPacketHeader header, int16_t eventType)`
- `static int16_t caerEventPacketHeaderGetEventSource (caerEventPacketHeaderConst header)`
- `static void caerEventPacketHeaderSetEventSource (caerEventPacketHeader header, int16_t eventSource)`
- `static int32_t caerEventPacketHeaderGetEventSize (caerEventPacketHeaderConst header)`
- `static void caerEventPacketHeaderSetEventSize (caerEventPacketHeader header, int32_t eventSize)`
- `static int32_t caerEventPacketHeaderGetEventTSOffset (caerEventPacketHeaderConst header)`
- `static void caerEventPacketHeaderSetEventTSOffset (caerEventPacketHeader header, int32_t eventTSOffset)`
- `static int32_t caerEventPacketHeaderGetEventTSOverflow (caerEventPacketHeaderConst header)`
- `static void caerEventPacketHeaderSetEventTSOverflow (caerEventPacketHeader header, int32_t eventTSOverflow)`
- `static int32_t caerEventPacketHeaderGetEventCapacity (caerEventPacketHeaderConst header)`
- `static void caerEventPacketHeaderSetEventCapacity (caerEventPacketHeader header, int32_t eventsCapacity)`
- `static int32_t caerEventPacketHeaderGetEventNumber (caerEventPacketHeaderConst header)`
- `static void caerEventPacketHeaderSetEventNumber (caerEventPacketHeader header, int32_t eventsNumber)`
- `static int32_t caerEventPacketHeaderGetEventValid (caerEventPacketHeaderConst header)`
- `static void caerEventPacketHeaderSetEventValid (caerEventPacketHeader header, int32_t eventsValid)`

- static const void * [caerGenericEventGetEvent](#) (caerEventPacketHeaderConst headerPtr, int32_t n)
- static int32_t [caerGenericEventGetTimestamp](#) (const void *eventPtr, caerEventPacketHeaderConst headerPtr)
- static int64_t [caerGenericEventGetTimestamp64](#) (const void *eventPtr, caerEventPacketHeaderConst headerPtr)
- static bool [caerGenericEventsValid](#) (const void *eventPtr)
- static bool [caerGenericEventCopy](#) (void *eventPtrDestination, const void *eventPtrSource, caerEventPacketHeaderConst headerPtrDestination, caerEventPacketHeaderConst headerPtrSource)
- static int64_t [caerEventPacketGetDataSize](#) (caerEventPacketHeaderConst header)
- static int64_t [caerEventPacketGetSize](#) (caerEventPacketHeaderConst header)
- static int64_t [caerEventPacketGetDataSizeEvents](#) (caerEventPacketHeaderConst header)
- static int64_t [caerEventPacketGetSizeEvents](#) (caerEventPacketHeaderConst header)
- static bool [caerEventPacketEquals](#) (caerEventPacketHeaderConst firstPacket, caerEventPacketHeaderConst secondPacket)
- static void [caerEventPacketClear](#) (caerEventPacketHeader packet)
- static void [caerEventPacketClean](#) (caerEventPacketHeader packet)
- **memset** (((uint8_t *) packet)+offset, 0, (size_t)((eventCapacity - eventValid) * eventSize))
- **caerEventPacketHeaderSetEventNumber** (packet, eventValid)
- static caerEventPacketHeader [caerEventPacketResize](#) (caerEventPacketHeader packet, int32_t newEventCapacity)
- static caerEventPacketHeader [caerEventPacketGrow](#) (caerEventPacketHeader packet, int32_t newEventCapacity)
- static caerEventPacketHeader [caerEventPacketAppend](#) (caerEventPacketHeader packet, caerEventPacketHeader appendPacket)
- static caerEventPacketHeader [caerEventPacketCopy](#) (caerEventPacketHeaderConst packet)
- static caerEventPacketHeader [caerEventPacketCopyOnlyEvents](#) (caerEventPacketHeaderConst packet)
- static caerEventPacketHeader [caerEventPacketCopyOnlyValidEvents](#) (caerEventPacketHeaderConst packet)
- **caerEventPacketHeaderSetEventCapacity** (packetCopy, eventValid)
- **caerEventPacketHeaderSetEventNumber** (packetCopy, eventValid)
- **return** (packetCopy)
- static caerEventPacketHeader [caerEventPacketAllocate](#) (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow, int16_t eventType, int32_t eventSize, int32_t eventTSOffset)

4.9.1 Detailed Description

Common EventPacket header format definition and handling functions. Every EventPacket, of any type, has as a first member a common header, which describes various properties of the contained events. This allows easy parsing of events. See the 'struct caer_event_packet_header' documentation for more details.

4.9.2 Macro Definition Documentation

4.9.2.1 CAER_DEFAULT_EVENT_TYPES_COUNT

```
#define CAER_DEFAULT_EVENT_TYPES_COUNT 14
```

Number of default event types that are part of libcaer. Corresponds to the count of definitions inside the 'enum caer_default_event_types' enumeration.

4.9.2.2 CAER_EVENT_PACKET_HEADER_SIZE

```
#define CAER_EVENT_PACKET_HEADER_SIZE 28
```

Size of the EventPacket header. This is constant across all supported systems.

4.9.2.3 CAER_ITERATOR_ALL_END

```
#define CAER_ITERATOR_ALL_END }
```

Generic iterator close statement.

4.9.2.4 CAER_ITERATOR_ALL_START

```
#define CAER_ITERATOR_ALL_START(  
    PACKET_HEADER,  
    EVENT_TYPE )
```

Value:

```
for (int32_t caerIteratorCounter = 0; caerIteratorCounter <  
    caerEventPacketHeaderGetEventNumber(PACKET_HEADER); \  
    caerIteratorCounter++) {  
    \  
    EVENT_TYPE caerIteratorElement = (EVENT_TYPE) caerGenericEventGetEvent(  
        PACKET_HEADER, caerIteratorCounter);
```

Generic iterator over all events in a packet. Returns the current index in the 'caerIteratorCounter' variable of type 'int32_t' and the current event in the 'caerIteratorElement' variable of type EVENT_TYPE.

PACKET_HEADER: a valid EventPacket header pointer. Cannot be NULL. EVENT_TYPE: the event pointer type for this EventPacket (ie. caerPolarityEvent or caerFrameEvent).

4.9.2.5 CAER_ITERATOR_VALID_END

```
#define CAER_ITERATOR_VALID_END }
```

Generic iterator close statement.

4.9.2.6 CAER_ITERATOR_VALID_START

```
#define CAER_ITERATOR_VALID_START(  
    PACKET_HEADER,  
    EVENT_TYPE )
```

Value:

```
for (int32_t caerIteratorCounter = 0; caerIteratorCounter <  
    caerEventPacketHeaderGetEventNumber(PACKET_HEADER); \  
    caerIteratorCounter++) {  
    \  
    EVENT_TYPE caerIteratorElement = (EVENT_TYPE) caerGenericEventGetEvent(  
        PACKET_HEADER, caerIteratorCounter); \  
    if (!caerGenericEventIsValid(caerIteratorElement)) {  
        \  
        continue;  
    }  
}
```

Generic iterator over only the valid events in a packet. Returns the current index in the 'caerIteratorCounter' variable of type 'int32_t' and the current event in the 'caerIteratorElement' variable of type EVENT_TYPE.

PACKET_HEADER: a valid EventPacket header pointer. Cannot be NULL. EVENT_TYPE: the event pointer type for this EventPacket (ie. caerPolarityEvent or caerFrameEvent).

4.9.2.7 TS_OVERFLOW_SHIFT

```
#define TS_OVERFLOW_SHIFT 31
```

64bit timestamp support: since timestamps wrap around after some time, being only 31 bit (32 bit signed int), another timestamp at the packet level provides another 31 bit (32 bit signed int), to enable the generation of a 62 bit (64 bit signed int) microsecond timestamp which is guaranteed to never wrap around (in the next 146'138 years at least). The TSOvflow needs to be shifted by 31 thus when constructing such a timestamp.

4.9.2.8 VALID_MARK_MASK

```
#define VALID_MARK_MASK 0x00000001
```

Generic validity mark: this bit is used to mark whether an event is still valid or not, and can be used to efficiently filter out events from a packet. The caerXXXEventValidate() and caerXXXEventInvalidate() functions should be used to toggle this! 0 in the 0th bit of the first byte means invalid, 1 means valid. This way zeroing-out an event packet sets all its events to invalid. Care must be taken to put the field containing the validity mark always as the first member of an event.

4.9.2.9 VALID_MARK_SHIFT

```
#define VALID_MARK_SHIFT 0
```

Generic validity mark: this bit is used to mark whether an event is still valid or not, and can be used to efficiently filter out events from a packet. The caerXXXEventValidate() and caerXXXEventInvalidate() functions should be used to toggle this! 0 in the 0th bit of the first byte means invalid, 1 means valid. This way zeroing-out an event packet sets all its events to invalid. Care must be taken to put the field containing the validity mark always as the first member of an event.

4.9.3 Typedef Documentation

4.9.3.1 caerEventPacketHeader

```
typedef struct caer_event_packet_header* caerEventPacketHeader
```

Type for pointer to EventPacket header data structure.

4.9.4 Enumeration Type Documentation

4.9.4.1 caer_default_event_types

```
enum caer_default_event_types
```

List of supported event types. Each event type has its own integer representation. All event types below 100 are reserved for use by libcaer and cAER. DO NOT USE THEM FOR YOUR OWN EVENT TYPES!

Enumerator

SPECIAL_EVENT	Special events.
POLARITY_EVENT	Polarity (change, DVS) events.
FRAME_EVENT	Frame (intensity, APS) events.
IMU6_EVENT	6 axes IMU events.
IMU9_EVENT	9 axes IMU events.
SAMPLE_EVENT	ADC sample events.
EAR_EVENT	Ear (cochlea) events.
CONFIG_EVENT	Device configuration events.
POINT1D_EVENT	1D measurement events.
POINT2D_EVENT	2D measurement events.
POINT3D_EVENT	3D measurement events.
POINT4D_EVENT	4D measurement events.
SPIKE_EVENT	Spike events.
MATRIX4x4_EVENT	4D matrix events.

4.9.5 Function Documentation

4.9.5.1 caerEventPacketAllocate()

```
static caerEventPacketHeader caerEventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow,
    int16_t eventType,
    int32_t eventSize,
    int32_t eventTSOffset ) [inline], [static]
```

Allocate memory for an event packet and fill its header with the proper information. THIS FUNCTION IS INTENDED FOR INTERNAL USE ONLY BY THE VARIOUS EVENT PACKET TYPES FOR MEMORY ALLOCATION.

Returns

memory for an event packet, NULL on error.

4.9.5.2 caerEventPacketAppend()

```
static caerEventPacketHeader caerEventPacketAppend (
    caerEventPacketHeader packet,
    caerEventPacketHeader appendPacket ) [inline], [static]
```

Appends an event packet to another. This is a simple append operation, no timestamp reordering is done. Please ensure time is monotonically increasing over the two packets! Use free() to reclaim this memory afterwards.

Parameters

<i>packet</i>	the main events packet.
<i>appendPacket</i>	the events packet to append on the main one.

Returns

a valid event packet handle or NULL on error. On success, the old packet handle is to be considered invalid and not to be used anymore. On failure, the old packet handle is not touched in any way. The appendPacket handle is never touched in any way.

4.9.5.3 caerEventPacketClean()

```
static void caerEventPacketClean (
    caerEventPacketHeader packet ) [inline], [static]
```

Clean a packet by removing all invalid events, so that the total number of events is the number of valid events. The packet's capacity doesn't change.

Parameters

<i>packet</i>	an event packet to clean.
---------------	---------------------------

4.9.5.4 caerEventPacketClear()

```
static void caerEventPacketClear (
    caerEventPacketHeader packet ) [inline], [static]
```

Clear a packet by zeroing out all events. Capacity doesn't change, event number is set to zero.

Parameters

<i>packet</i>	an event packet to clear out.
---------------	-------------------------------

4.9.5.5 caerEventPacketCopy()

```
static caerEventPacketHeader caerEventPacketCopy (
    caerEventPacketHeaderConst packet ) [inline], [static]
```

Make a full copy of an event packet (up to eventCapacity).

Parameters

<i>packet</i>	an event packet to copy.
---------------	--------------------------

Returns

a full copy of an event packet.

4.9.5.6 caerEventPacketCopyOnlyEvents()

```
static caerEventPacketHeader caerEventPacketCopyOnlyEvents (  
    caerEventPacketHeaderConst packet ) [inline], [static]
```

Make a copy of an event packet, sized down to only include the currently present events (eventNumber, valid+invalid), and not including the possible extra unused events (up to eventCapacity).

Parameters

<i>packet</i>	an event packet to copy.
---------------	--------------------------

Returns

a sized down copy of an event packet.

4.9.5.7 caerEventPacketCopyOnlyValidEvents()

```
static caerEventPacketHeader caerEventPacketCopyOnlyValidEvents (  
    caerEventPacketHeaderConst packet ) [inline], [static]
```

Make a copy of an event packet, sized down to only include the currently valid events (eventValid), and discarding everything else.

Parameters

<i>packet</i>	an event packet to copy.
---------------	--------------------------

Returns

a copy of an event packet, containing only valid events.

4.9.5.8 caerEventPacketEquals()

```
static bool caerEventPacketEquals (
    caerEventPacketHeaderConst firstPacket,
    caerEventPacketHeaderConst secondPacket ) [inline], [static]
```

Verify if two event packets are equal. This means that the header and all events are equal.

Parameters

<i>firstPacket</i>	an event packet to be compared.
<i>secondPacket</i>	the other event packet to compare against.

Returns

true if both are the same, false otherwise.

4.9.5.9 caerEventPacketGetDataSize()

```
static int64_t caerEventPacketGetDataSize (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Get the data size of an event packet, in bytes. This is only the size of the data portion, excluding the header.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
---------------	---

Returns

the event packet data size in bytes.

4.9.5.10 caerEventPacketGetDataSizeEvents()

```
static int64_t caerEventPacketGetDataSizeEvents (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Get the data size of an event packet, in bytes, up to its last actual event. This means only up to EventNumber, not up to EventCapacity. This is only the size of the data portion, excluding the header.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
---------------	---

Returns

the event packet data size in bytes (up to event number).

4.9.5.11 caerEventPacketGetSize()

```
static int64_t caerEventPacketGetSize (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Get the full size of an event packet, in bytes. This includes both the header and the data portion.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
---------------	---

Returns

the event packet size in bytes.

4.9.5.12 caerEventPacketGetSizeEvents()

```
static int64_t caerEventPacketGetSizeEvents (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Get the full size of an event packet, in bytes, up to its last actual event. This means only up to EventNumber, not up to EventCapacity. This includes both the header and the data portion.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
---------------	---

Returns

the event packet size in bytes (up to event number).

4.9.5.13 caerEventPacketGrow()

```
static caerEventPacketHeader caerEventPacketGrow (
    caerEventPacketHeader packet,
    int32_t newEventCapacity ) [inline], [static]
```

Grows an event packet. This only supports strictly increasing the size of a packet. For a more flexible resize operation, see [caerEventPacketResize\(\)](#). Use [free\(\)](#) to reclaim this memory afterwards.

Parameters

<i>packet</i>	the current event packet.
<i>newEventCapacity</i>	the new maximum number of events this packet can hold. Cannot be zero.

Returns

a valid event packet handle or NULL on error. On success, the old packet handle is to be considered invalid and not to be used anymore. On failure, the old packet handle is not touched in any way.

4.9.5.14 caerEventPacketHeaderGetEventCapacity()

```
static int32_t caerEventPacketHeaderGetEventCapacity (  
    caerEventPacketHeaderConst header ) [inline], [static]
```

Get the maximum number of events this packet can store.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
---------------	---

Returns

the number of events this packet can hold.

4.9.5.15 caerEventPacketHeaderGetEventNumber()

```
static int32_t caerEventPacketHeaderGetEventNumber (  
    caerEventPacketHeaderConst header ) [inline], [static]
```

Get the number of events currently stored in this packet, considering both valid and invalid events.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
---------------	---

Returns

the number of events in this packet.

4.9.5.16 caerEventPacketHeaderGetEventSize()

```
static int32_t caerEventPacketHeaderGetEventSize (  
    caerEventPacketHeaderConst header ) [inline], [static]
```

Get the size of a single event, in bytes. All events inside an event packet always have the same size.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
---------------	---

Returns

the event size in bytes.

4.9.5.17 caerEventPacketHeaderGetEventSource()

```
static int16_t caerEventPacketHeaderGetEventSource (  
    caerEventPacketHeaderConst header ) [inline], [static]
```

Get the numerical event source ID, representing the event source that generated all the events present in this packet.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
---------------	---

Returns

the numerical event source ID.

4.9.5.18 caerEventPacketHeaderGetEventTSOffset()

```
static int32_t caerEventPacketHeaderGetEventTSOffset (  
    caerEventPacketHeaderConst header ) [inline], [static]
```

Get the offset, in bytes, to where the field with the main 32 bit timestamp is stored. This is useful for generic access to the timestamp field, given that different event types might have it at different offsets or might even have multiple timestamps, in which case this offset references the 'main' timestamp, the most representative one.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
---------------	---

Returns

the event timestamp offset in bytes.

4.9.5.19 caerEventPacketHeaderGetEventTSOverflow()

```
static int32_t caerEventPacketHeaderGetEventTSOverflow (  
    caerEventPacketHeaderConst header ) [inline], [static]
```

Get the 32 bit timestamp overflow counter (in microseconds). This is per-packet and is used to generate a 64 bit timestamp that never wraps around. Since timestamps wrap around after some time, being only 31 bit (32 bit signed int), another timestamp at the packet level provides another 31 bit (32 bit signed int), to enable the generation of a 62 bit (64 bit signed int) microsecond timestamp which is guaranteed to never wrap around (in the next 146'138 years at least).

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
---------------	---

Returns

the packet-level timestamp overflow counter, in microseconds.

4.9.5.20 caerEventPacketHeaderGetEventType()

```
static int16_t caerEventPacketHeaderGetEventType (  
    caerEventPacketHeaderConst header ) [inline], [static]
```

Return the numerical event type ID, representing the event type this EventPacket is containing.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
---------------	---

Returns

the numerical event type (see 'enum caer_default_event_types').

4.9.5.21 caerEventPacketHeaderGetEventValid()

```
static int32_t caerEventPacketHeaderGetEventValid (  
    caerEventPacketHeaderConst header ) [inline], [static]
```

Get the number of valid events in this packet, disregarding invalid ones (where the invalid mark is set).

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
---------------	---

Returns

the number of valid events in this packet.

4.9.5.22 caerEventPacketHeaderSetEventCapacity()

```
static void caerEventPacketHeaderSetEventCapacity (
    caerEventPacketHeader header,
    int32_t eventsCapacity ) [inline], [static]
```

Set the maximum number of events this packet can store. This is determined at packet allocation time and should not be changed during the life-time of the packet.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
<i>eventsCapacity</i>	the number of events this packet can hold.

4.9.5.23 caerEventPacketHeaderSetEventNumber()

```
static void caerEventPacketHeaderSetEventNumber (
    caerEventPacketHeader header,
    int32_t eventsNumber ) [inline], [static]
```

Set the number of events currently stored in this packet, considering both valid and invalid events.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
<i>eventsNumber</i>	the number of events in this packet.

4.9.5.24 caerEventPacketHeaderSetEventSize()

```
static void caerEventPacketHeaderSetEventSize (
    caerEventPacketHeader header,
    int32_t eventSize ) [inline], [static]
```

Set the size of a single event, in bytes. All events inside an event packet always have the same size.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
<i>eventSize</i>	the event size in bytes.

4.9.5.25 caerEventPacketHeaderSetEventSource()

```
static void caerEventPacketHeaderSetEventSource (
    caerEventPacketHeader header,
    int16_t eventSource ) [inline], [static]
```

Set the numerical event source ID, representing the event source that generated all the events present in this packet. This ID should be unique at least within a process, if not within the whole system, to guarantee correct identification of who generated an event later on.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
<i>eventSource</i>	the numerical event source ID.

4.9.5.26 caerEventPacketHeaderSetEventTSOffset()

```
static void caerEventPacketHeaderSetEventTSOffset (
    caerEventPacketHeader header,
    int32_t eventTSOffset ) [inline], [static]
```

Set the offset, in bytes, to where the field with the main 32 bit timestamp is stored. This is useful for generic access to the timestamp field, given that different event types might have it at different offsets or might even have multiple timestamps, in which case this offset references the 'main' timestamp, the most representative one.

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
<i>eventTSOffset</i>	the event timestamp offset in bytes.

4.9.5.27 caerEventPacketHeaderSetEventTSOverflow()

```
static void caerEventPacketHeaderSetEventTSOverflow (
    caerEventPacketHeader header,
    int32_t eventTSOverflow ) [inline], [static]
```

Set the 32 bit timestamp overflow counter (in microseconds). This is per-packet and is used to generate a 64 bit timestamp that never wraps around. Since timestamps wrap around after some time, being only 31 bit (32 bit signed

int), another timestamp at the packet level provides another 31 bit (32 bit signed int), to enable the generation of a 62 bit (64 bit signed int) microsecond timestamp which is guaranteed to never wrap around (in the next 146'138 years at least).

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
<i>eventTSOverflow</i>	the packet-level timestamp overflow counter, in microseconds.

4.9.5.28 caerEventPacketHeaderSetEventType()

```
static void caerEventPacketHeaderSetEventType (
    caerEventPacketHeader header,
    int16_t eventType ) [inline], [static]
```

Set the numerical event type ID, representing the event type this EventPacket will contain. All event types below 100 are reserved for use by libcaer and cAER. DO NOT USE THEM FOR YOUR OWN EVENT TYPES!

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
<i>eventType</i>	the numerical event type (see 'enum caer_default_event_types').

4.9.5.29 caerEventPacketHeaderSetEventValid()

```
static void caerEventPacketHeaderSetEventValid (
    caerEventPacketHeader header,
    int32_t eventsValid ) [inline], [static]
```

Set the number of valid events in this packet, disregarding invalid ones (where the invalid mark is set).

Parameters

<i>header</i>	a valid EventPacket header pointer. Cannot be NULL.
<i>eventsValid</i>	the number of valid events in this packet.

4.9.5.30 caerEventPacketResize()

```
static caerEventPacketHeader caerEventPacketResize (
    caerEventPacketHeader packet,
    int32_t newEventCapacity ) [inline], [static]
```

Resize an event packet. First, the packet is cleaned (all invalid events removed), then:

- If the old and new event capacity are equal, nothing else changes.
- If the new capacity is bigger, the packet is enlarged and the new events are initialized to all zeros (invalid).
- If the new capacity is smaller, the packet is truncated at the given point. Use `free()` to reclaim this memory afterwards.

Parameters

<i>packet</i>	the current event packet.
<i>newEventCapacity</i>	the new maximum number of events this packet can hold. Cannot be zero.

Returns

a valid event packet handle or NULL on error. On success, the old packet handle is to be considered invalid and not to be used anymore. On failure, the old packet handle is still valid, but will have been cleaned of all invalid events!

4.9.5.31 `caerGenericEventCopy()`

```
static bool caerGenericEventCopy (
    void * eventPtrDestination,
    const void * eventPtrSource,
    caerEventPacketHeaderConst headerPtrDestination,
    caerEventPacketHeaderConst headerPtrSource ) [inline], [static]
```

Copy a given event's content to another location in memory.

Parameters

<i>eventPtrDestination</i>	a generic pointer to an event to copy to. Cannot be NULL.
<i>eventPtrSource</i>	a generic pointer to an event to copy from. Cannot be NULL.
<i>headerPtrDestination</i>	a valid EventPacket header pointer from the destination packet. Cannot be NULL.
<i>headerPtrSource</i>	a valid EventPacket header pointer from the source packet. Cannot be NULL.

Returns

true on successful copy, false otherwise.

4.9.5.32 `caerGenericEventGetEvent()`

```
static const void* caerGenericEventGetEvent (
    caerEventPacketHeaderConst headerPtr,
    int32_t n ) [inline], [static]
```

Get a generic pointer to an event, without having to know what event type the packet is containing.

Parameters

<i>headerPtr</i>	a valid EventPacket header pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventNumber] bounds.

Returns

a generic pointer to the requested event. NULL on error. This points to unmodifiable memory, as it should never be used for anything other than read operations, such as [caerGenericEventGetTimestamp\(\)](#). Don't modify the memory, you have no idea what it is! If you do know, just use the proper typed packet functions.

4.9.5.33 caerGenericEventGetTimestamp()

```
static int32_t caerGenericEventGetTimestamp (  
    const void * eventPtr,  
    caerEventPacketHeaderConst headerPtr ) [inline], [static]
```

Get the main 32 bit timestamp for a generic event, without having to know what event type the packet is containing.

Parameters

<i>eventPtr</i>	a generic pointer to an event. Cannot be NULL.
<i>headerPtr</i>	a valid EventPacket header pointer. Cannot be NULL.

Returns

the main 32 bit timestamp of this event.

4.9.5.34 caerGenericEventGetTimestamp64()

```
static int64_t caerGenericEventGetTimestamp64 (  
    const void * eventPtr,  
    caerEventPacketHeaderConst headerPtr ) [inline], [static]
```

Get the main 64 bit timestamp for a generic event, without having to know what event type the packet is containing. This takes the per-packet timestamp into account too, generating a timestamp that doesn't suffer from overflow problems.

Parameters

<i>eventPtr</i>	a generic pointer to an event. Cannot be NULL.
<i>headerPtr</i>	a valid EventPacket header pointer. Cannot be NULL.

Returns

the main 64 bit timestamp of this event.

4.9.5.35 caerGenericEventIsValid()

```
static bool caerGenericEventIsValid (
    const void * eventPtr ) [inline], [static]
```

Check if the given generic event is valid or not.

Parameters

<i>eventPtr</i>	a generic pointer to an event. Cannot be NULL.
-----------------	--

Returns

true if the event is valid, false otherwise.

4.9.5.36 PACKED_STRUCT()

```
PACKED_STRUCT (
    struct caer_event_packet_header { int16_t eventType;int16_t eventSource;int32_t
eventSize;int32_t eventTSOffset;int32_t eventTSOverflow;int32_t eventCapacity;int32_t event↵
Number;int32_t eventValid;} )
```

EventPacket header data structure definition. The size, also defined in CAER_EVENT_PACKET_HEADER_SIZE, must always be constant. The header is common to all types of event packets and is always the very first member of an event packet data structure. Signed integers are used for compatibility with languages that do not have unsigned ones, such as Java.

4.10 events/config.h File Reference

```
#include "common.h"
```

Macros

- `#define CAER_CONFIGURATION_ITERATOR_ALL_START(CONFIGURATION_PACKET)`
- `#define CAER_CONFIGURATION_CONST_ITERATOR_ALL_START(CONFIGURATION_PACKET)`
- `#define CAER_CONFIGURATION_ITERATOR_ALL_END }`
- `#define CAER_CONFIGURATION_ITERATOR_VALID_START(CONFIGURATION_PACKET)`
- `#define CAER_CONFIGURATION_CONST_ITERATOR_VALID_START(CONFIGURATION_PACKET)`
- `#define CAER_CONFIGURATION_ITERATOR_VALID_END }`

- `#define CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_START`(CONFIGURATION_PACKET)
 - `#define CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_ALL_START`(CONFIGURATION_PACKET)
 - `#define CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_END` }
 - `#define CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_START`(CONFIGURATION_PACKET)
 - `#define CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_VALID_START`(CONFIGURATION_PACKET)
 - `#define CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_END` }
-
- `#define CONFIG_MODULE_ADDR_SHIFT` 1
 - `#define CONFIG_MODULE_ADDR_MASK` 0x0000007F

Typedefs

- `typedef struct caer_configuration_event * caerConfigurationEvent`
- `typedef const struct caer_configuration_event * caerConfigurationEventConst`
- `typedef struct caer_configuration_event_packet * caerConfigurationEventPacket`
- `typedef const struct caer_configuration_event_packet * caerConfigurationEventPacketConst`

Functions

- `PACKED_STRUCT` (struct caer_configuration_event { uint8_t moduleAddress;uint8_t parameter;uint32_t parameter;int32_t timestamp;})
- `PACKED_STRUCT` (struct caer_configuration_event_packet { struct caer_event_packet_header packetHeader;struct caer_configuration_event events[];})
- static `caerConfigurationEventPacket caerConfigurationEventPacketAllocate` (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static `caerConfigurationEventPacket caerConfigurationEventPacketFromPacketHeader` (caerEventPacketHeader header)
- static `caerConfigurationEventPacketConst caerConfigurationEventPacketFromPacketHeaderConst` (caerEventPacketHeaderConst header)
- static `caerConfigurationEvent caerConfigurationEventPacketGetEvent` (caerConfigurationEventPacket packet, int32_t n)
- static `caerConfigurationEventConst caerConfigurationEventPacketGetEventConst` (caerConfigurationEventPacketConst packet, int32_t n)
- static `int32_t caerConfigurationEventGetTimestamp` (caerConfigurationEventConst event)
- static `int64_t caerConfigurationEventGetTimestamp64` (caerConfigurationEventConst event, caerConfigurationEventPacketConst packet)
- static void `caerConfigurationEventSetTimestamp` (caerConfigurationEvent event, int32_t timestamp)
- static bool `caerConfigurationEventsIsValid` (caerConfigurationEventConst event)
- static void `caerConfigurationEventValidate` (caerConfigurationEvent event, caerConfigurationEventPacket packet)
- static void `caerConfigurationEventInvalidate` (caerConfigurationEvent event, caerConfigurationEventPacket packet)
- static `uint8_t caerConfigurationEventGetModuleAddress` (caerConfigurationEventConst event)
- static void `caerConfigurationEventSetModuleAddress` (caerConfigurationEvent event, uint8_t moduleAddress)
- static `uint8_t caerConfigurationEventGetParameterAddress` (caerConfigurationEventConst event)
- static void `caerConfigurationEventSetParameterAddress` (caerConfigurationEvent event, uint8_t parameterAddress)
- static `uint32_t caerConfigurationEventGetParameter` (caerConfigurationEventConst event)
- static void `caerConfigurationEventSetParameter` (caerConfigurationEvent event, uint32_t parameter)

4.10.1 Detailed Description

Configuration Events format definition and handling functions. This event contains information about the current configuration of the device. By having configuration as a standardized event format, it becomes host-software agnostic, and it also becomes part of the event stream, enabling easy tracking of changes through time, by putting them into the event stream at the moment they happen. While the resolution of the timestamps for these events is in microseconds for compatibility with all other event types, the precision is in the order of ~ 1 -20 milliseconds, given that these events are generated and injected on the host-side.

4.10.2 Macro Definition Documentation

4.10.2.1 CAER_CONFIGURATION_CONST_ITERATOR_ALL_START

```
#define CAER_CONFIGURATION_CONST_ITERATOR_ALL_START (
    CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter = 0;
     caerConfigurationIteratorCounter
     < caerEventPacketHeaderGetEventNumber (& (CONFIGURATION_PACKET) ->
     packetHeader); \
     caerConfigurationIteratorCounter++) {
    caerConfigurationEventConst caerConfigurationIteratorElement
    = caerConfigurationEventPacketGetEventConst (
    CONFIGURATION_PACKET, caerConfigurationIteratorCounter);
```

Const-Iterator over all configuration events in a packet. Returns the current index in the 'caerConfigurationIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEventConst.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.2 CAER_CONFIGURATION_CONST_ITERATOR_VALID_START

```
#define CAER_CONFIGURATION_CONST_ITERATOR_VALID_START (
    CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter = 0;
     caerConfigurationIteratorCounter
     < caerEventPacketHeaderGetEventNumber (& (CONFIGURATION_PACKET) ->
     packetHeader); \
     caerConfigurationIteratorCounter++) {
    caerConfigurationEventConst caerConfigurationIteratorElement
    = caerConfigurationEventPacketGetEventConst (
    CONFIGURATION_PACKET, caerConfigurationIteratorCounter); \
    if (!caerConfigurationEventIsValid(caerConfigurationIteratorElement))
    {
        continue;
    }
```

Const-Iterator over only the valid configuration events in a packet. Returns the current index in the 'caerConfigurationIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEventConst.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.3 CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_ALL_START(
    CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter
    = caerEventPacketHeaderGetEventNumber(&(CONFIGURATION_PACKET)->
    packetHeader) - 1; \
    caerConfigurationIteratorCounter >= 0; caerConfigurationIteratorCounter--) { \
    caerConfigurationEventConst caerConfigurationIteratorElement \
    = caerConfigurationEventPacketGetEventConst( \
    CONFIGURATION_PACKET, caerConfigurationIteratorCounter);
```

Const-Reverse iterator over all configuration events in a packet. Returns the current index in the 'caerConfigurationIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEventConst.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.4 CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_VALID_START(
    CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter
    = caerEventPacketHeaderGetEventNumber(&(CONFIGURATION_PACKET)->
    packetHeader) - 1; \
    caerConfigurationIteratorCounter >= 0; caerConfigurationIteratorCounter--) { \
    caerConfigurationEventConst caerConfigurationIteratorElement \
    = caerConfigurationEventPacketGetEventConst( \
    CONFIGURATION_PACKET, caerConfigurationIteratorCounter); \
    if (!caerConfigurationEventIsValid(caerConfigurationIteratorElement)) \
    { \
        continue; \
    }
```

Const-Reverse iterator over only the valid configuration events in a packet. Returns the current index in the 'caerConfigurationIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEventConst.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.5 CAER_CONFIGURATION_ITERATOR_ALL_END

```
#define CAER_CONFIGURATION_ITERATOR_ALL_END }
```

Iterator close statement.

4.10.2.6 CAER_CONFIGURATION_ITERATOR_ALL_START

```
#define CAER_CONFIGURATION_ITERATOR_ALL_START(
    CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter = 0;
     caerConfigurationIteratorCounter
     < caerEventPacketHeaderGetEventNumber (&(CONFIGURATION_PACKET) ->
     packetHeader); \
     caerConfigurationIteratorCounter++) {
    caerConfigurationEvent caerConfigurationIteratorElement
    = caerConfigurationEventPacketGetEvent (CONFIGURATION_PACKET
    , caerConfigurationIteratorCounter);
```

Iterator over all configuration events in a packet. Returns the current index in the 'caerConfigurationIteratorCounter' variable of type 'int32_t' and the current event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEvent.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.7 CAER_CONFIGURATION_ITERATOR_VALID_END

```
#define CAER_CONFIGURATION_ITERATOR_VALID_END }
```

Iterator close statement.

4.10.2.8 CAER_CONFIGURATION_ITERATOR_VALID_START

```
#define CAER_CONFIGURATION_ITERATOR_VALID_START(
    CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter = 0;
     caerConfigurationIteratorCounter
     < caerEventPacketHeaderGetEventNumber (&(CONFIGURATION_PACKET) ->
     packetHeader); \
     caerConfigurationIteratorCounter++) {
    caerConfigurationEvent caerConfigurationIteratorElement
    = caerConfigurationEventPacketGetEvent (CONFIGURATION_PACKET
    , caerConfigurationIteratorCounter); \
    if (!caerConfigurationEventIsValid(caerConfigurationIteratorElement))
    {
        continue;
    }
```

Iterator over only the valid configuration events in a packet. Returns the current index in the 'caerConfigurationIteratorCounter' variable of type 'int32_t' and the current event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEvent.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.9 CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_END

```
#define CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.10.2.10 CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_START

```
#define CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_START (
    CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter
    = caerEventPacketHeaderGetEventNumber (& (CONFIGURATION_PACKET) ->
    packetHeader) - 1; \
    caerConfigurationIteratorCounter >= 0; caerConfigurationIteratorCounter--) { \
    caerConfigurationEvent caerConfigurationIteratorElement \
    = caerConfigurationEventPacketGetEvent (CONFIGURATION_PACKET \
    , caerConfigurationIteratorCounter);
```

Reverse iterator over all configuration events in a packet. Returns the current index in the 'caerConfiguration↔IteratorCounter' variable of type 'int32_t' and the current event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEvent.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.11 CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_END

```
#define CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.10.2.12 CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_START

```
#define CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_START (
    CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter
    = caerEventPacketHeaderGetEventNumber (& (CONFIGURATION_PACKET) ->
    packetHeader) - 1; \
    caerConfigurationIteratorCounter >= 0; caerConfigurationIteratorCounter--) { \
    \
    caerConfigurationEvent caerConfigurationIteratorElement \
    = caerConfigurationEventPacketGetEvent (CONFIGURATION_PACKET \
    , caerConfigurationIteratorCounter); \
    if (!caerConfigurationEventIsValid(caerConfigurationIteratorElement)) \
    { \
        continue; \
    } \
}
```

Reverse iterator over only the valid configuration events in a packet. Returns the current index in the 'caer↔ConfigurationIteratorCounter' variable of type 'int32_t' and the current event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEvent.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.13 CONFIG_MODULE_ADDR_MASK

```
#define CONFIG_MODULE_ADDR_MASK 0x0000007F
```

Shift and mask values for the module address. Module address is only 7 bits, since the eighth bit is used device-side to differentiate reads from writes. Here we can just re-use it for the validity mark.

4.10.2.14 CONFIG_MODULE_ADDR_SHIFT

```
#define CONFIG_MODULE_ADDR_SHIFT 1
```

Shift and mask values for the module address. Module address is only 7 bits, since the eighth bit is used device-side to differentiate reads from writes. Here we can just re-use it for the validity mark.

4.10.3 Typedef Documentation

4.10.3.1 caerConfigurationEvent

```
typedef struct caer_configuration_event* caerConfigurationEvent
```

Type for pointer to configuration event data structure.

4.10.3.2 caerConfigurationEventPacket

```
typedef struct caer_configuration_event_packet* caerConfigurationEventPacket
```

Type for pointer to configuration event packet data structure.

4.10.4 Function Documentation

4.10.4.1 caerConfigurationEventGetModuleAddress()

```
static uint8_t caerConfigurationEventGetModuleAddress (
    caerConfigurationEventConst event ) [inline], [static]
```

Get the configuration event's module address.

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
--------------	---

Returns

configuration module address.

4.10.4.2 caerConfigurationEventGetParameter()

```
static uint32_t caerConfigurationEventGetParameter (  
    caerConfigurationEventConst event ) [inline], [static]
```

Get the configuration event's parameter.

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
--------------	---

Returns

configuration parameter.

4.10.4.3 caerConfigurationEventGetParameterAddress()

```
static uint8_t caerConfigurationEventGetParameterAddress (  
    caerConfigurationEventConst event ) [inline], [static]
```

Get the configuration event's parameter address.

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
--------------	---

Returns

configuration parameter address.

4.10.4.4 caerConfigurationEventGetTimestamp()

```
static int32_t caerConfigurationEventGetTimestamp (  
    caerConfigurationEventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See ['caerEventPacketHeaderGetEventTSOverflow\(\)'](#) documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond timestamp.

4.10.4.5 caerConfigurationEventGetTimestamp64()

```
static int64_t caerConfigurationEventGetTimestamp64 (  
    caerConfigurationEventConst event,  
    caerConfigurationEventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
<i>packet</i>	the ConfigurationEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.10.4.6 caerConfigurationEventInvalidate()

```
static void caerConfigurationEventInvalidate (  
    caerConfigurationEvent event,  
    caerConfigurationEventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
<i>packet</i>	the ConfigurationEventPacket pointer for the packet containing this event. Cannot be NULL.

4.10.4.7 caerConfigurationEventIsValid()

```
static bool caerConfigurationEventIsValid (
    caerConfigurationEventConst event ) [inline], [static]
```

Check if this configuration event is valid.

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
--------------	---

Returns

true if valid, false if not.

4.10.4.8 caerConfigurationEventPacketAllocate()

```
static caerConfigurationEventPacket caerConfigurationEventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]
```

Allocate a new configuration events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid ConfigurationEventPacket handle or NULL on error.

4.10.4.9 caerConfigurationEventPacketFromPacketHeader()

```
static caerConfigurationEventPacket caerConfigurationEventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```

Transform a generic event packet header into a Configuration event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.10.4.10 caerConfigurationEventPacketFromPacketHeaderConst()

```
static caerConfigurationEventPacketConst caerConfigurationEventPacketFromPacketHeaderConst (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only Configuration event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.10.4.11 caerConfigurationEventPacketGetEvent()

```
static caerConfigurationEvent caerConfigurationEventPacketGetEvent (
    caerConfigurationEventPacket packet,
    int32_t n ) [inline], [static]
```

Get the configuration event at the given index from the event packet.

Parameters

<i>packet</i>	a valid ConfigurationEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested configuration event. NULL on error.

4.10.4.12 caerConfigurationEventPacketGetEventConst()

```
static caerConfigurationEventConst caerConfigurationEventPacketGetEventConst (
    caerConfigurationEventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the configuration event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid ConfigurationEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only configuration event. NULL on error.

4.10.4.13 caerConfigurationEventSetModuleAddress()

```
static void caerConfigurationEventSetModuleAddress (  
    caerConfigurationEvent event,  
    uint8_t moduleAddress ) [inline], [static]
```

Set the configuration event's module address.

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
<i>moduleAddress</i>	configuration module address.

4.10.4.14 caerConfigurationEventSetParameter()

```
static void caerConfigurationEventSetParameter (  
    caerConfigurationEvent event,  
    uint32_t parameter ) [inline], [static]
```

Set the configuration event's parameter.

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
<i>parameter</i>	configuration parameter.

4.10.4.15 caerConfigurationEventSetParameterAddress()

```
static void caerConfigurationEventSetParameterAddress (  
    caerConfigurationEvent event,  
    uint8_t parameterAddress ) [inline], [static]
```

Set the configuration event's parameter address.

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
<i>parameterAddress</i>	configuration parameter address.

4.10.4.16 caerConfigurationEventSetTimestamp()

```
static void caerConfigurationEventSetTimestamp (
    caerConfigurationEvent event,
    int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.10.4.17 caerConfigurationEventValidate()

```
static void caerConfigurationEventValidate (
    caerConfigurationEvent event,
    caerConfigurationEventPacket packet ) [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
<i>packet</i>	the ConfigurationEventPacket pointer for the packet containing this event. Cannot be NULL.

4.10.4.18 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
    struct caer_configuration_event { uint8_t moduleAddress;uint8_t parameterAddress;uint32_t parameter;int32_t timestamp;} )
```

Configuration event data structure definition. This contains the actual configuration module address, the parameter address and the actual parameter content, as well as the 32 bit event timestamp. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.10.4.19 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
    struct caer_configuration_event_packet { struct caer_event_packet_header packet←
Header; struct caer_configuration_event events[]; } )
```

Configuration event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.11 events/ear.h File Reference

```
#include "common.h"
```

Macros

- `#define CAER_EAR_ITERATOR_ALL_START(EAR_PACKET)`
- `#define CAER_EAR_CONST_ITERATOR_ALL_START(EAR_PACKET)`
- `#define CAER_EAR_ITERATOR_ALL_END }`
- `#define CAER_EAR_ITERATOR_VALID_START(EAR_PACKET)`
- `#define CAER_EAR_CONST_ITERATOR_VALID_START(EAR_PACKET)`
- `#define CAER_EAR_ITERATOR_VALID_END }`
- `#define CAER_EAR_REVERSE_ITERATOR_ALL_START(EAR_PACKET)`
- `#define CAER_EAR_CONST_REVERSE_ITERATOR_ALL_START(EAR_PACKET)`
- `#define CAER_EAR_REVERSE_ITERATOR_ALL_END }`
- `#define CAER_EAR_REVERSE_ITERATOR_VALID_START(EAR_PACKET)`
- `#define CAER_EAR_CONST_REVERSE_ITERATOR_VALID_START(EAR_PACKET)`
- `#define CAER_EAR_REVERSE_ITERATOR_VALID_END }`

- `#define EAR_SHIFT 1`
- `#define EAR_MASK 0x0000000F`
- `#define EAR_CHANNEL_SHIFT 5`
- `#define EAR_CHANNEL_MASK 0x000007FF`
- `#define EAR_NEURON_SHIFT 16`
- `#define EAR_NEURON_MASK 0x000000FF`
- `#define EAR_FILTER_SHIFT 24`
- `#define EAR_FILTER_MASK 0x000000FF`

Typedefs

- `typedef struct caer_ear_event * caerEarEvent`
- `typedef const struct caer_ear_event * caerEarEventConst`
- `typedef struct caer_ear_event_packet * caerEarEventPacket`
- `typedef const struct caer_ear_event_packet * caerEarEventPacketConst`

Functions

- [PACKED_STRUCT](#) (struct caer_ear_event { uint32_t data;int32_t timestamp;})
- [PACKED_STRUCT](#) (struct caer_ear_event_packet { struct caer_event_packet_header packetHeader;struct caer_ear_event events[;];})
- static [caerEarEventPacket caerEarEventPacketAllocate](#) (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static [caerEarEventPacket caerEarEventPacketFromPacketHeader](#) ([caerEventPacketHeader](#) header)
- static caerEarEventPacketConst [caerEarEventPacketFromPacketHeaderConst](#) ([caerEventPacketHeaderConst](#) header)
- static [caerEarEvent caerEarEventPacketGetEvent](#) ([caerEarEventPacket](#) packet, int32_t n)
- static caerEarEventConst [caerEarEventPacketGetEventConst](#) ([caerEarEventPacketConst](#) packet, int32_t n)
- static int32_t [caerEarEventGetTimestamp](#) ([caerEarEventConst](#) event)
- static int64_t [caerEarEventGetTimestamp64](#) ([caerEarEventConst](#) event, [caerEarEventPacketConst](#) packet)
- static void [caerEarEventSetTimestamp](#) ([caerEarEvent](#) event, int32_t timestamp)
- static bool [caerEarEventsValid](#) ([caerEarEventConst](#) event)
- static void [caerEarEventValidate](#) ([caerEarEvent](#) event, [caerEarEventPacket](#) packet)
- static void [caerEarEventInvalidate](#) ([caerEarEvent](#) event, [caerEarEventPacket](#) packet)
- static uint8_t [caerEarEventGetEar](#) ([caerEarEventConst](#) event)
- static void [caerEarEventSetEar](#) ([caerEarEvent](#) event, uint8_t ear)
- static uint16_t [caerEarEventGetChannel](#) ([caerEarEventConst](#) event)
- static void [caerEarEventSetChannel](#) ([caerEarEvent](#) event, uint16_t channel)
- static uint8_t [caerEarEventGetNeuron](#) ([caerEarEventConst](#) event)
- static void [caerEarEventSetNeuron](#) ([caerEarEvent](#) event, uint8_t neuron)
- static uint8_t [caerEarEventGetFilter](#) ([caerEarEventConst](#) event)
- static void [caerEarEventSetFilter](#) ([caerEarEvent](#) event, uint8_t filter)

4.11.1 Detailed Description

Ear (Cochlea) Events format definition and handling functions. This encodes events from a silicon cochlea chip, containing information about which ear (microphone) generated the event, as well as which channel was involved and additional information on filters and neurons.

4.11.2 Macro Definition Documentation

4.11.2.1 CAER_EAR_CONST_ITERATOR_ALL_START

```
#define CAER_EAR_CONST_ITERATOR_ALL_START(  
    EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = 0; \
    caerEarIteratorCounter < caerEventPacketHeaderGetEventNumber(&( \
    EAR_PACKET)->packetHeader); \
    caerEarIteratorCounter++) { \
    caerEarEventConst caerEarIteratorElement = \
    caerEarEventPacketGetEventConst(EAR_PACKET, caerEarIteratorCounter);
```

Const-Iterator over all ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerEarIteratorElement' variable of type caerEarEventConst.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.2 CAER_EAR_CONST_ITERATOR_VALID_START

```
#define CAER_EAR_CONST_ITERATOR_VALID_START(
    EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = 0;
     caerEarIteratorCounter < caerEventPacketHeaderGetEventNumber(&(
     EAR_PACKET)->packetHeader); \
     caerEarIteratorCounter++) {
    caerEarEventConst caerEarIteratorElement
        = caerEarEventPacketGetEventConst(EAR_PACKET,
        caerEarIteratorCounter);
    if (!caerEarEventIsValid(caerEarIteratorElement)) {
        continue;
    }
```

Const-Iterator over only the valid ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerEarIteratorElement' variable of type caerEar↔EventConst.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.3 CAER_EAR_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_EAR_CONST_REVERSE_ITERATOR_ALL_START(
    EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = caerEventPacketHeaderGetEventNumber
     (&(EAR_PACKET)->packetHeader) - 1; \
     caerEarIteratorCounter >= 0; caerEarIteratorCounter--) {
    caerEarEventConst caerEarIteratorElement =
    caerEarEventPacketGetEventConst(EAR_PACKET, caerEarIteratorCounter);
```

Const-Reverse iterator over all ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerEarIteratorElement' variable of type caerEar↔EventConst.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.4 CAER_EAR_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_EAR_CONST_REVERSE_ITERATOR_VALID_START(
    EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = caerEventPacketHeaderGetEventNumber
     (&(EAR_PACKET)->packetHeader) - 1; \
     caerEarIteratorCounter >= 0; caerEarIteratorCounter--) {
    caerEarEventConst caerEarIteratorElement
        = caerEarEventPacketGetEventConst(EAR_PACKET,
        caerEarIteratorCounter);
    if (!caerEarEventIsValid(caerEarIteratorElement)) {
        continue;
    }
```

Const-Reverse iterator over only the valid ear events in a packet. Returns the current index in the 'caerEarIterator↔Counter' variable of type 'int32_t' and the current read-only event in the 'caerEarIteratorElement' variable of type caerEarEventConst.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.5 CAER_EAR_ITERATOR_ALL_END

```
#define CAER_EAR_ITERATOR_ALL_END }
```

Iterator close statement.

4.11.2.6 CAER_EAR_ITERATOR_ALL_START

```
#define CAER_EAR_ITERATOR_ALL_START(  
    EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = 0; \
     caerEarIteratorCounter < caerEventPacketHeaderGetEventNumber(&(
     EAR_PACKET)->packetHeader); \
     caerEarIteratorCounter++) { \
     caerEarEvent caerEarIteratorElement = caerEarEventPacketGetEvent(
     EAR_PACKET, caerEarIteratorCounter);
```

Iterator over all ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current event in the 'caerEarIteratorElement' variable of type caerEarEvent.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.7 CAER_EAR_ITERATOR_VALID_END

```
#define CAER_EAR_ITERATOR_VALID_END }
```

Iterator close statement.

4.11.2.8 CAER_EAR_ITERATOR_VALID_START

```
#define CAER_EAR_ITERATOR_VALID_START(  
    EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = 0; \
     caerEarIteratorCounter < caerEventPacketHeaderGetEventNumber(&(
     EAR_PACKET)->packetHeader); \
     caerEarIteratorCounter++) { \
     caerEarEvent caerEarIteratorElement = caerEarEventPacketGetEvent(
     EAR_PACKET, caerEarIteratorCounter); \
     if (!caerEarEventIsValid(caerEarIteratorElement)) { \
         continue; \
     }
```

Iterator over only the valid ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current event in the 'caerEarIteratorElement' variable of type caerEarEvent.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.9 CAER_EAR_REVERSE_ITERATOR_ALL_END

```
#define CAER_EAR_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.11.2.10 CAER_EAR_REVERSE_ITERATOR_ALL_START

```
#define CAER_EAR_REVERSE_ITERATOR_ALL_START(  
    EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(EAR_PACKET)->packetHeader) - 1; \  
    caerEarIteratorCounter >= 0; caerEarIteratorCounter--) {  
    \  
    caerEarEvent caerEarIteratorElement = caerEarEventPacketGetEvent(  
        EAR_PACKET, caerEarIteratorCounter);
```

Reverse iterator over all ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current event in the 'caerEarIteratorElement' variable of type caerEarEvent.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.11 CAER_EAR_REVERSE_ITERATOR_VALID_END

```
#define CAER_EAR_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.11.2.12 CAER_EAR_REVERSE_ITERATOR_VALID_START

```
#define CAER_EAR_REVERSE_ITERATOR_VALID_START(  
    EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(EAR_PACKET)->packetHeader) - 1; \  
    caerEarIteratorCounter >= 0; caerEarIteratorCounter--) {  
    \  
    caerEarEvent caerEarIteratorElement = caerEarEventPacketGetEvent(  
        EAR_PACKET, caerEarIteratorCounter); \  
    if (!caerEarEventIsValid(caerEarIteratorElement)) {  
        \  
        continue;  
    }
```

Reverse iterator over only the valid ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current event in the 'caerEarIteratorElement' variable of type caerEarEvent.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.13 EAR_CHANNEL_MASK

```
#define EAR_CHANNEL_MASK 0x000007FF
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.11.2.14 EAR_CHANNEL_SHIFT

```
#define EAR_CHANNEL_SHIFT 5
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.11.2.15 EAR_FILTER_MASK

```
#define EAR_FILTER_MASK 0x000000FF
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.11.2.16 EAR_FILTER_SHIFT

```
#define EAR_FILTER_SHIFT 24
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.11.2.17 EAR_MASK

```
#define EAR_MASK 0x0000000F
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.11.2.18 EAR_NEURON_MASK

```
#define EAR_NEURON_MASK 0x000000FF
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.11.2.19 EAR_NEURON_SHIFT

```
#define EAR_NEURON_SHIFT 16
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.11.2.20 EAR_SHIFT

```
#define EAR_SHIFT 1
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.11.3 Typedef Documentation

4.11.3.1 caerEarEvent

```
typedef struct caer_ear_event* caerEarEvent
```

Type for pointer to ear (cochlea) event data structure.

4.11.3.2 caerEarEventPacket

```
typedef struct caer_ear_event_packet* caerEarEventPacket
```

Type for pointer to ear (cochlea) event packet data structure.

4.11.4 Function Documentation

4.11.4.1 caerEarEventGetChannel()

```
static uint16_t caerEarEventGetChannel (
    caerEarEventConst event ) [inline], [static]
```

Get the channel (frequency band) ID. The channels count from 0 upward, where 0 is the highest frequency channel, while higher numbers are progressively lower frequency channels. This is derived from how the actual human ear works.

Parameters

<i>event</i>	a valid EarEvent pointer. Cannot be NULL.
--------------	---

Returns

the channel (frequency band) ID.

4.11.4.2 caerEarEventGetEar()

```
static uint8_t caerEarEventGetEar (  
    caerEarEventConst event ) [inline], [static]
```

Get the numerical ID of the ear (microphone). Usually, 0 is left, 1 is right for 2 ear cochleas. For 4 ear cochleas, 0 is front left, 1 is front right, 2 is back left and 3 is back right.

Parameters

<i>event</i>	a valid EarEvent pointer. Cannot be NULL.
--------------	---

Returns

the ear (microphone) ID.

4.11.4.3 caerEarEventGetTimestamp()

```
static int32_t caerEarEventGetTimestamp (  
    caerEarEventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid EarEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond timestamp.

4.11.4.4 caerEarEventGetTimestamp64()

```
static int64_t caerEarEventGetTimestamp64 (
    caerEarEventConst event,
    caerEarEventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid EarEvent pointer. Cannot be NULL.
<i>packet</i>	the EarEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.11.4.5 caerEarEventInvalidate()

```
static void caerEarEventInvalidate (
    caerEarEvent event,
    caerEarEventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid EarEvent pointer. Cannot be NULL.
<i>packet</i>	the EarEventPacket pointer for the packet containing this event. Cannot be NULL.

4.11.4.6 caerEarEventIsValid()

```
static bool caerEarEventIsValid (
    caerEarEventConst event ) [inline], [static]
```

Check if this ear (cochlea) event is valid.

Parameters

<i>event</i>	a valid EarEvent pointer. Cannot be NULL.
--------------	---

Returns

true if valid, false if not.

4.11.4.7 caerEarEventPacketAllocate()

```
static caerEarEventPacket caerEarEventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]
```

Allocate a new ear (cochlea) events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid EarEventPacket handle or NULL on error.

4.11.4.8 caerEarEventPacketFromPacketHeader()

```
static caerEarEventPacket caerEarEventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```

Transform a generic event packet header into an Ear event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.11.4.9 caerEarEventPacketFromPacketHeaderConst()

```
static caerEarEventPacketConst caerEarEventPacketFromPacketHeaderConst (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only Ear event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.11.4.10 caerEarEventPacketGetEvent()

```
static caerEarEvent caerEarEventPacketGetEvent (
    caerEarEventPacket packet,
    int32_t n )  [inline], [static]
```

Get the ear (cochlea) event at the given index from the event packet.

Parameters

<i>packet</i>	a valid EarEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested ear (cochlea) event. NULL on error.

4.11.4.11 caerEarEventPacketGetEventConst()

```
static caerEarEventConst caerEarEventPacketGetEventConst (
    caerEarEventPacketConst packet,
    int32_t n )  [inline], [static]
```

Get the ear (cochlea) event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid EarEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only ear (cochlea) event. NULL on error.

4.11.4.12 caerEarEventSetChannel()

```
static void caerEarEventSetChannel (
    caerEarEvent event,
    uint16_t channel ) [inline], [static]
```

Set the channel (frequency band) ID. The channels count from 0 upward, where 0 is the highest frequency channel, while higher numbers are progressively lower frequency channels. This is derived from how the actual human ear works.

Parameters

<i>event</i>	a valid EarEvent pointer. Cannot be NULL.
<i>channel</i>	the channel (frequency band) ID.

4.11.4.13 caerEarEventSetEar()

```
static void caerEarEventSetEar (
    caerEarEvent event,
    uint8_t ear ) [inline], [static]
```

Set the numerical ID of the ear (microphone). Usually, 0 is left, 1 is right for 2 ear cochleas. For 4 ear cochleas, 0 is front left, 1 is front right, 2 is back left and 3 is back right.

Parameters

<i>event</i>	a valid EarEvent pointer. Cannot be NULL.
<i>ear</i>	the ear (microphone) ID.

4.11.4.14 caerEarEventSetTimestamp()

```
static void caerEarEventSetTimestamp (
    caerEarEvent event,
    int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid EarEvent pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.11.4.15 caerEarEventValidate()

```
static void caerEarEventValidate (
    caerEarEvent event,
    caerEarEventPacket packet ) [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid EarEvent pointer. Cannot be NULL.
<i>packet</i>	the EarEventPacket pointer for the packet containing this event. Cannot be NULL.

4.11.4.16 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
    struct caer_ear_event { uint32_t data;int32_t timestamp;} )
```

Ear (cochlea) event data structure definition. Contains information on events gotten from a cochlea chip: ears, channels, neurons and filters are stored. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.11.4.17 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
    struct caer_ear_event_packet { struct caer_event_packet_header packetHeader;struct
    caer_ear_event events[];} )
```

Ear (cochlea) event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.12 events/frame.h File Reference

```
#include "common.h"
```


Macros

- `#define CAER_FRAME_ITERATOR_ALL_START(FRAME_PACKET)`
 - `#define CAER_FRAME_CONST_ITERATOR_ALL_START(FRAME_PACKET)`
 - `#define CAER_FRAME_ITERATOR_ALL_END }`
 - `#define CAER_FRAME_ITERATOR_VALID_START(FRAME_PACKET)`
 - `#define CAER_FRAME_CONST_ITERATOR_VALID_START(FRAME_PACKET)`
 - `#define CAER_FRAME_ITERATOR_VALID_END }`
 - `#define CAER_FRAME_REVERSE_ITERATOR_ALL_START(FRAME_PACKET)`
 - `#define CAER_FRAME_CONST_REVERSE_ITERATOR_ALL_START(FRAME_PACKET)`
 - `#define CAER_FRAME_REVERSE_ITERATOR_ALL_END }`
 - `#define CAER_FRAME_REVERSE_ITERATOR_VALID_START(FRAME_PACKET)`
 - `#define CAER_FRAME_CONST_REVERSE_ITERATOR_VALID_START(FRAME_PACKET)`
 - `#define CAER_FRAME_REVERSE_ITERATOR_VALID_END }`
-
- `#define FRAME_COLOR_CHANNELS_SHIFT 1`
 - `#define FRAME_COLOR_CHANNELS_MASK 0x00000007`
 - `#define FRAME_COLOR_FILTER_SHIFT 4`
 - `#define FRAME_COLOR_FILTER_MASK 0x0000000F`
 - `#define FRAME_ROI_IDENTIFIER_SHIFT 8`
 - `#define FRAME_ROI_IDENTIFIER_MASK 0x0000007F`

Typedefs

- `typedef struct caer_frame_event * caerFrameEvent`
- `typedef const struct caer_frame_event * caerFrameEventConst`
- `typedef struct caer_frame_event_packet * caerFrameEventPacket`
- `typedef const struct caer_frame_event_packet * caerFrameEventPacketConst`

Enumerations

- `enum caer_frame_event_color_channels { GRAYSCALE = 1, RGB = 3, RGBA = 4 }`
- `enum caer_frame_event_color_filter {
MONO = 0, RGBG = 1, GRGB = 2, GBGR = 3,
BGRG = 4, RGBW = 5, GRWB = 6, WBGR = 7,
BWRG = 8 }`

Functions

- `PACKED_STRUCT` (struct caer_frame_event { uint32_t info;int32_t ts_startframe;int32_t ts_endframe;int32_t ts_startexposure;int32_t ts_endexposure;int32_t lengthX;int32_t lengthY;int32_t positionX;int32_t positionY;uint16_t pixels[1];})
- `PACKED_STRUCT` (struct caer_frame_event_packet { struct caer_event_packet_header packetHeader;})
- `static caerFrameEventPacket caerFrameEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow, int32_t maxLengthX, int32_t maxLengthY, int16_t maxChannelNumber)`
- `static caerFrameEventPacket caerFrameEventPacketFromPacketHeader (caerEventPacketHeader header)`
- `static caerFrameEventPacketConst caerFrameEventPacketFromPacketHeaderConst (caerEventPacketHeaderConst header)`
- `static caerFrameEvent caerFrameEventPacketGetEvent (caerFrameEventPacket packet, int32_t n)`

- static caerFrameEventConst [caerFrameEventPacketGetEventConst](#) (caerFrameEventPacketConst packet, int32_t n)
- static int32_t [caerFrameEventGetTSSStartOfFrame](#) (caerFrameEventConst event)
- static int64_t [caerFrameEventGetTSSStartOfFrame64](#) (caerFrameEventConst event, caerFrameEventPacketConst packet)
- static void [caerFrameEventSetTSSStartOfFrame](#) (caerFrameEvent event, int32_t startFrame)
- static int32_t [caerFrameEventGetTSEndOfFrame](#) (caerFrameEventConst event)
- static int64_t [caerFrameEventGetTSEndOfFrame64](#) (caerFrameEventConst event, caerFrameEventPacketConst packet)
- static void [caerFrameEventSetTSEndOfFrame](#) (caerFrameEvent event, int32_t endFrame)
- static int32_t [caerFrameEventGetTSSStartOfExposure](#) (caerFrameEventConst event)
- static int64_t [caerFrameEventGetTSSStartOfExposure64](#) (caerFrameEventConst event, caerFrameEventPacketConst packet)
- static void [caerFrameEventSetTSSStartOfExposure](#) (caerFrameEvent event, int32_t startExposure)
- static int32_t [caerFrameEventGetTSEndOfExposure](#) (caerFrameEventConst event)
- static int64_t [caerFrameEventGetTSEndOfExposure64](#) (caerFrameEventConst event, caerFrameEventPacketConst packet)
- static void [caerFrameEventSetTSEndOfExposure](#) (caerFrameEvent event, int32_t endExposure)
- static int32_t [caerFrameEventGetExposureLength](#) (caerFrameEventConst event)
- static int32_t [caerFrameEventGetTimestamp](#) (caerFrameEventConst event)
- static int64_t [caerFrameEventGetTimestamp64](#) (caerFrameEventConst event, caerFrameEventPacketConst packet)
- static bool [caerFrameEventIsValid](#) (caerFrameEventConst event)
- static void [caerFrameEventValidate](#) (caerFrameEvent event, caerFrameEventPacket packet)
- static void [caerFrameEventInvalidate](#) (caerFrameEvent event, caerFrameEventPacket packet)
- static size_t [caerFrameEventPacketGetPixelsSize](#) (caerFrameEventPacketConst packet)
- static size_t [caerFrameEventPacketGetPixelsMaxIndex](#) (caerFrameEventPacketConst packet)
- static uint8_t [caerFrameEventGetROIIdentifier](#) (caerFrameEventConst event)
- static void [caerFrameEventSetROIIdentifier](#) (caerFrameEvent event, uint8_t roiIdentifier)
- static enum [caer_frame_event_color_filter](#) [caerFrameEventGetColorFilter](#) (caerFrameEventConst event)
- static void [caerFrameEventSetColorFilter](#) (caerFrameEvent event, enum [caer_frame_event_color_filter](#) colorFilter)
- static int32_t [caerFrameEventGetLengthX](#) (caerFrameEventConst event)
- static int32_t [caerFrameEventGetLengthY](#) (caerFrameEventConst event)
- static enum [caer_frame_event_color_channels](#) [caerFrameEventGetChannelNumber](#) (caerFrameEventConst event)
- static void [caerFrameEventSetLengthXLengthYChannelNumber](#) (caerFrameEvent event, int32_t lengthX, int32_t lengthY, enum [caer_frame_event_color_channels](#) channelNumber, caerFrameEventPacketConst packet)
- static size_t [caerFrameEventGetPixelsMaxIndex](#) (caerFrameEventConst event)
- static size_t [caerFrameEventGetPixelsSize](#) (caerFrameEventConst event)
- static int32_t [caerFrameEventGetPositionX](#) (caerFrameEventConst event)
- static void [caerFrameEventSetPositionX](#) (caerFrameEvent event, int32_t positionX)
- static int32_t [caerFrameEventGetPositionY](#) (caerFrameEventConst event)
- static void [caerFrameEventSetPositionY](#) (caerFrameEvent event, int32_t positionY)
- static uint16_t [caerFrameEventGetPixel](#) (caerFrameEventConst event, int32_t xAddress, int32_t yAddress)
- static void [caerFrameEventSetPixel](#) (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint16_t pixelValue)
- static uint16_t [caerFrameEventGetPixelForChannel](#) (caerFrameEventConst event, int32_t xAddress, int32_t yAddress, uint8_t channel)
- static void [caerFrameEventSetPixelForChannel](#) (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint8_t channel, uint16_t pixelValue)
- static uint16_t [caerFrameEventGetPixelUnsafe](#) (caerFrameEventConst event, int32_t xAddress, int32_t yAddress)
- static void [caerFrameEventSetPixelUnsafe](#) (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint16_t pixelValue)

- static uint16_t [caerFrameEventGetPixelForChannelUnsafe](#) (caerFrameEventConst event, int32_t xAddress, int32_t yAddress, uint8_t channel)
- static void [caerFrameEventSetPixelForChannelUnsafe](#) (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint8_t channel, uint16_t pixelValue)
- static uint16_t * [caerFrameEventGetPixelArrayUnsafe](#) (caerFrameEvent event)
- static const uint16_t * [caerFrameEventGetPixelArrayUnsafeConst](#) (caerFrameEventConst event)

4.12.1 Detailed Description

Frame Events format definition and handling functions. This event type encodes intensity frames, like you would get from a normal APS camera. It supports multiple channels for color, color filter information, as well as multiple Regions of Interest (ROI). The (0, 0) pixel is in the upper left corner of the screen, like in OpenCV/computer graphics. The pixel array is laid out row by row (increasing X axis), going from top to bottom (increasing Y axis). To copy a frame event, the usual assignment operator = cannot be used. Please use [caerGenericEventCopy\(\)](#) to copy frame events!

4.12.2 Macro Definition Documentation

4.12.2.1 CAER_FRAME_CONST_ITERATOR_ALL_START

```
#define CAER_FRAME_CONST_ITERATOR_ALL_START(  
    FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = 0; \
     caerFrameIteratorCounter < caerEventPacketHeaderGetEventNumber(  
     &(FRAME_PACKET)->packetHeader); \
     caerFrameIteratorCounter++) { \
     caerFrameEventConst caerFrameIteratorElement \
     = caerFrameEventPacketGetEventConst (FRAME_PACKET, \
     caerFrameIteratorCounter);
```

Const-Iterator over all frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerFrameIteratorElement' variable of type caerFrameEventConst.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.2 CAER_FRAME_CONST_ITERATOR_VALID_START

```
#define CAER_FRAME_CONST_ITERATOR_VALID_START(  
    FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = 0; \
     caerFrameIteratorCounter < caerEventPacketHeaderGetEventNumber(  
     &(FRAME_PACKET)->packetHeader); \
     caerFrameIteratorCounter++) { \
     caerFrameEventConst caerFrameIteratorElement \
     = caerFrameEventPacketGetEventConst (FRAME_PACKET, \
     caerFrameIteratorCounter); \
     if (!caerFrameEventIsValid(caerFrameIteratorElement)) { \
         continue; \
     }
```

Const-Iterator over only the valid frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerFrameIteratorElement' variable of type caerFrameEventConst.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.3 CAER_FRAME_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_FRAME_CONST_REVERSE_ITERATOR_ALL_START(  
    FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(FRAME_PACKET)->packetHeader) - 1; \  
    caerFrameIteratorCounter >= 0; caerFrameIteratorCounter--) {  
    \  
    caerFrameEventConst caerFrameIteratorElement  
    \  
    = caerFrameEventPacketGetEventConst (FRAME_PACKET,  
    caerFrameIteratorCounter);
```

Const-Reverse iterator over all frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerFrameIteratorElement' variable of type caerFrameEventConst.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.4 CAER_FRAME_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_FRAME_CONST_REVERSE_ITERATOR_VALID_START(  
    FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(FRAME_PACKET)->packetHeader) - 1; \  
    caerFrameIteratorCounter >= 0; caerFrameIteratorCounter--) {  
    \  
    caerFrameEventConst caerFrameIteratorElement  
    \  
    = caerFrameEventPacketGetEventConst (FRAME_PACKET,  
    caerFrameIteratorCounter);  
    if (!caerFrameEventIsValid(caerFrameIteratorElement)) {  
    \  
        continue;  
    }  
    }
```

Const-Reverse iterator over only the valid frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerFrameIteratorElement' variable of type caerFrameEventConst.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.5 CAER_FRAME_ITERATOR_ALL_END

```
#define CAER_FRAME_ITERATOR_ALL_END }
```

Iterator close statement.

4.12.2.6 CAER_FRAME_ITERATOR_ALL_START

```
#define CAER_FRAME_ITERATOR_ALL_START(
    FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = 0; \
     caerFrameIteratorCounter < caerEventPacketHeaderGetEventNumber( \
     &(FRAME_PACKET)->packetHeader); \
     caerFrameIteratorCounter++) { \
    caerFrameEvent caerFrameIteratorElement = caerFrameEventPacketGetEvent( \
    FRAME_PACKET, caerFrameIteratorCounter);
```

Iterator over all frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current event in the 'caerFrameIteratorElement' variable of type caerFrameEvent.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.7 CAER_FRAME_ITERATOR_VALID_END

```
#define CAER_FRAME_ITERATOR_VALID_END }
```

Iterator close statement.

4.12.2.8 CAER_FRAME_ITERATOR_VALID_START

```
#define CAER_FRAME_ITERATOR_VALID_START(
    FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = 0; \
     caerFrameIteratorCounter < caerEventPacketHeaderGetEventNumber( \
     &(FRAME_PACKET)->packetHeader); \
     caerFrameIteratorCounter++) { \
    caerFrameEvent caerFrameIteratorElement \
    = caerFrameEventPacketGetEvent(FRAME_PACKET, \
    caerFrameIteratorCounter); \
    if (!caerFrameEventIsValid(caerFrameIteratorElement)) { \
        continue; \
    }
```

Iterator over only the valid frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current event in the 'caerFrameIteratorElement' variable of type caerFrameEvent.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.9 CAER_FRAME_REVERSE_ITERATOR_ALL_END

```
#define CAER_FRAME_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.12.2.10 CAER_FRAME_REVERSE_ITERATOR_ALL_START

```
#define CAER_FRAME_REVERSE_ITERATOR_ALL_START(  
    FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(FRAME_PACKET)->packetHeader) - 1; \  
    caerFrameIteratorCounter >= 0; caerFrameIteratorCounter--) {  
    \  
    caerFrameEvent caerFrameIteratorElement = caerFrameEventPacketGetEvent(  
        FRAME_PACKET, caerFrameIteratorCounter);
```

Reverse iterator over all frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current event in the 'caerFrameIteratorElement' variable of type caerFrameEvent.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.11 CAER_FRAME_REVERSE_ITERATOR_VALID_END

```
#define CAER_FRAME_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.12.2.12 CAER_FRAME_REVERSE_ITERATOR_VALID_START

```
#define CAER_FRAME_REVERSE_ITERATOR_VALID_START(  
    FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(FRAME_PACKET)->packetHeader) - 1; \  
    caerFrameIteratorCounter >= 0; caerFrameIteratorCounter--) {  
    \  
    caerFrameEvent caerFrameIteratorElement  
    \  
    = caerFrameEventPacketGetEvent(FRAME_PACKET,  
        caerFrameIteratorCounter);  
    if (!caerFrameEventIsValid(caerFrameIteratorElement)) {  
        \  
        continue;  
    }  
}
```

Reverse iterator over only the valid frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current event in the 'caerFrameIteratorElement' variable of type caerFrameEvent.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.13 FRAME_COLOR_CHANNELS_MASK

```
#define FRAME_COLOR_CHANNELS_MASK 0x00000007
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame_event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see ['common.h'](#) for more details.

4.12.2.14 FRAME_COLOR_CHANNELS_SHIFT

```
#define FRAME_COLOR_CHANNELS_SHIFT 1
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame_event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see ['common.h'](#) for more details.

4.12.2.15 FRAME_COLOR_FILTER_MASK

```
#define FRAME_COLOR_FILTER_MASK 0x0000000F
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame_event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see ['common.h'](#) for more details.

4.12.2.16 FRAME_COLOR_FILTER_SHIFT

```
#define FRAME_COLOR_FILTER_SHIFT 4
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame_event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see ['common.h'](#) for more details.

4.12.2.17 FRAME_ROI_IDENTIFIER_MASK

```
#define FRAME_ROI_IDENTIFIER_MASK 0x0000007F
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame_event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see ['common.h'](#) for more details.

4.12.2.18 FRAME_ROI_IDENTIFIER_SHIFT

```
#define FRAME_ROI_IDENTIFIER_SHIFT 8
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame_event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see [common.h](#) for more details.

4.12.3 Typedef Documentation

4.12.3.1 caerFrameEvent

```
typedef struct caer_frame_event* caerFrameEvent
```

Type for pointer to frame event data structure.

4.12.3.2 caerFrameEventPacket

```
typedef struct caer_frame_event_packet* caerFrameEventPacket
```

Type for pointer to frame event packet data structure.

4.12.4 Enumeration Type Documentation

4.12.4.1 caer_frame_event_color_channels

```
enum caer_frame_event_color_channels
```

List of all frame event color channel identifiers. Used to interpret the frame event color channel field.

Enumerator

GRAYSCALE	Grayscale, one channel only.
RGB	Red Green Blue, 3 color channels.
RGBA	Red Green Blue Alpha, 3 color channels plus transparency.

4.12.4.2 caer_frame_event_color_filter

```
enum caer_frame_event_color_filter
```

List of all frame event color filter identifiers. Used to interpret the frame event color filter field.

Enumerator

MONO	No color filter present, all light passes.
RGBG	Standard Bayer color filter, 1 red 2 green 1 blue. Variation 1.
GRGB	Standard Bayer color filter, 1 red 2 green 1 blue. Variation 2.
GBGR	Standard Bayer color filter, 1 red 2 green 1 blue. Variation 3.
BGRG	Standard Bayer color filter, 1 red 2 green 1 blue. Variation 4.
RGBW	Modified Bayer color filter, with white (pass all light) instead of extra green. Variation 1.
GRWB	Modified Bayer color filter, with white (pass all light) instead of extra green. Variation 2.
WBGR	Modified Bayer color filter, with white (pass all light) instead of extra green. Variation 3.
BWRG	Modified Bayer color filter, with white (pass all light) instead of extra green. Variation 4.

4.12.5 Function Documentation

4.12.5.1 caerFrameEventGetChannelNumber()

```
static enum caer_frame_event_color_channels caerFrameEventGetChannelNumber (  
    caerFrameEventConst event ) [inline], [static]
```

Get the actual color channels number for the current frame. This can be used to store RGB frames for example.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

frame color channels number.

4.12.5.2 caerFrameEventGetColorFilter()

```
static enum caer_frame_event_color_filter caerFrameEventGetColorFilter (  
    caerFrameEventConst event ) [inline], [static]
```

Get the identifier for the color filter used by the sensor. Useful for interpolating color images.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

color filter identifier.

4.12.5.3 caerFrameEventGetExposureLength()

```
static int32_t caerFrameEventGetExposureLength (  
    caerFrameEventConst event ) [inline], [static]
```

The total length, in microseconds, of the frame exposure time.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

the exposure time in microseconds.

4.12.5.4 caerFrameEventGetLengthX()

```
static int32_t caerFrameEventGetLengthX (  
    caerFrameEventConst event ) [inline], [static]
```

Get the actual X axis length for the current frame.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

frame X axis length.

4.12.5.5 caerFrameEventGetLengthY()

```
static int32_t caerFrameEventGetLengthY (  
    caerFrameEventConst event ) [inline], [static]
```

Get the actual Y axis length for the current frame.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

frame Y axis length.

4.12.5.6 caerFrameEventGetPixel()

```
static uint16_t caerFrameEventGetPixel (  
    caerFrameEventConst event,  
    int32_t xAddress,  
    int32_t yAddress ) [inline], [static]
```

Get the pixel value at the specified (X, Y) address. (X, Y) are checked against the actual possible values for this frame. Different channels are not taken into account! The (0, 0) pixel is in the upper left corner, like in OpenC↔V/computer graphics.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>xAddress</i>	X address value (checked).
<i>yAddress</i>	Y address value (checked).

Returns

pixel value (normalized to 16 bit depth).

4.12.5.7 caerFrameEventGetPixelArrayUnsafe()

```
static uint16_t* caerFrameEventGetPixelArrayUnsafe (  
    caerFrameEvent event ) [inline], [static]
```

Get a direct pointer to the underlying pixels array. This can be used to both get and set values. No checks at all are performed at any point, nor any conversions, use this at your own risk! Remember that the 16 bit pixel values are in little-endian! The pixel array is laid out row by row (increasing X axis), going from top to bottom (increasing Y axis).

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

the pixels array (16 bit integers are little-endian).

4.12.5.8 caerFrameEventGetPixelArrayUnsafeConst()

```
static const uint16_t* caerFrameEventGetPixelArrayUnsafeConst (
    caerFrameEventConst event ) [inline], [static]
```

Get a direct read-only pointer to the underlying pixels array. This can be used to only get values. No checks at all are performed at any point, nor any conversions, use this at your own risk! Remember that the 16 bit pixel values are in little-endian! The pixel array is laid out row by row (increasing X axis), going from top to bottom (increasing Y axis).

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

the read-only pixels array (16 bit integers are little-endian).

4.12.5.9 caerFrameEventGetPixelForChannel()

```
static uint16_t caerFrameEventGetPixelForChannel (
    caerFrameEventConst event,
    int32_t xAddress,
    int32_t yAddress,
    uint8_t channel ) [inline], [static]
```

Get the pixel value at the specified (X, Y) address, taking into account the specified channel. (X, Y) and the channel number are checked against the actual possible values for this frame. The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>xAddress</i>	X address value (checked).
<i>yAddress</i>	Y address value (checked).
<i>channel</i>	the channel number (checked).

Returns

pixel value (normalized to 16 bit depth).

4.12.5.10 caerFrameEventGetPixelForChannelUnsafe()

```
static uint16_t caerFrameEventGetPixelForChannelUnsafe (
    caerFrameEventConst event,
    int32_t xAddress,
    int32_t yAddress,
    uint8_t channel ) [inline], [static]
```

Get the pixel value at the specified (X, Y) address, taking into account the specified channel. No checks on (X, Y) and the channel number are performed! The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>xAddress</i>	X address value (unchecked).
<i>yAddress</i>	Y address value (unchecked).
<i>channel</i>	the channel number (unchecked).

Returns

pixel value (normalized to 16 bit depth).

4.12.5.11 caerFrameEventGetPixelsMaxIndex()

```
static size_t caerFrameEventGetPixelsMaxIndex (
    caerFrameEventConst event ) [inline], [static]
```

Get the maximum valid index into the pixel array, at which you can still get valid pixels.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

maximum valid pixels array index.

4.12.5.12 caerFrameEventGetPixelsSize()

```
static size_t caerFrameEventGetPixelsSize (
    caerFrameEventConst event ) [inline], [static]
```

Get the maximum size of the pixels array in bytes, in which you can still get valid pixels.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

maximum valid pixels array size in bytes.

4.12.5.13 caerFrameEventGetPixelUnsafe()

```
static uint16_t caerFrameEventGetPixelUnsafe (
    caerFrameEventConst event,
    int32_t xAddress,
    int32_t yAddress ) [inline], [static]
```

Get the pixel value at the specified (X, Y) address. No checks on (X, Y) are performed! The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>xAddress</i>	X address value (unchecked).
<i>yAddress</i>	Y address value (unchecked).

Returns

pixel value (normalized to 16 bit depth).

4.12.5.14 caerFrameEventGetPositionX()

```
static int32_t caerFrameEventGetPositionX (
    caerFrameEventConst event ) [inline], [static]
```

Get the X axis position offset. This is used to place partial frames, like the ones gotten from ROI readouts, in the visual space.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

X axis position offset.

4.12.5.15 caerFrameEventGetPositionY()

```
static int32_t caerFrameEventGetPositionY (
    caerFrameEventConst event ) [inline], [static]
```

Get the Y axis position offset. This is used to place partial frames, like the ones gotten from ROI readouts, in the visual space.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

Y axis position offset.

4.12.5.16 caerFrameEventGetROIIdentifier()

```
static uint8_t caerFrameEventGetROIIdentifier (
    caerFrameEventConst event ) [inline], [static]
```

Get the numerical identifier for the Region of Interest (ROI) region, to distinguish between multiple of them.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

numerical ROI identifier.

4.12.5.17 caerFrameEventGetTimestamp()

```
static int32_t caerFrameEventGetTimestamp (
    caerFrameEventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. This is a median of the exposure timestamps. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond timestamp.

4.12.5.18 caerFrameEventGetTimestamp64()

```
static int64_t caerFrameEventGetTimestamp64 (
    caerFrameEventConst event,
    caerFrameEventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. This is a median of the exposure timestamps. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>packet</i>	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.12.5.19 caerFrameEventGetTSEndOfExposure()

```
static int32_t caerFrameEventGetTSEndOfExposure (
    caerFrameEventConst event ) [inline], [static]
```

Get the 32bit end of exposure timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond end of exposure timestamp.

4.12.5.20 caerFrameEventGetTSEndOfExposure64()

```
static int64_t caerFrameEventGetTSEndOfExposure64 (
    caerFrameEventConst event,
    caerFrameEventPacketConst packet ) [inline], [static]
```

Get the 64bit end of exposure timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>packet</i>	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond end of exposure timestamp.

4.12.5.21 caerFrameEventGetTSEndOfFrame()

```
static int32_t caerFrameEventGetTSEndOfFrame (
    caerFrameEventConst event ) [inline], [static]
```

Get the 32bit end of frame capture timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond end of frame timestamp.

4.12.5.22 caerFrameEventGetTSEndOfFrame64()

```
static int64_t caerFrameEventGetTSEndOfFrame64 (
    caerFrameEventConst event,
    caerFrameEventPacketConst packet ) [inline], [static]
```

Get the 64bit end of frame capture timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>packet</i>	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond end of frame timestamp.

4.12.5.23 caerFrameEventGetTSStartOfExposure()

```
static int32_t caerFrameEventGetTSStartOfExposure (  
    caerFrameEventConst event ) [inline], [static]
```

Get the 32bit start of exposure timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond start of exposure timestamp.

4.12.5.24 caerFrameEventGetTSStartOfExposure64()

```
static int64_t caerFrameEventGetTSStartOfExposure64 (  
    caerFrameEventConst event,  
    caerFrameEventPacketConst packet ) [inline], [static]
```

Get the 64bit start of exposure timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>packet</i>	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond start of exposure timestamp.

4.12.5.25 caerFrameEventGetTSStartOfFrame()

```
static int32_t caerFrameEventGetTSStartOfFrame (  
    caerFrameEventConst event ) [inline], [static]
```

Get the 32bit start of frame capture timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond start of frame timestamp.

4.12.5.26 caerFrameEventGetTSStartOfFrame64()

```
static int64_t caerFrameEventGetTSStartOfFrame64 (
    caerFrameEventConst event,
    caerFrameEventPacketConst packet ) [inline], [static]
```

Get the 64bit start of frame capture timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>packet</i>	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond start of frame timestamp.

4.12.5.27 caerFrameEventInvalidate()

```
static void caerFrameEventInvalidate (
    caerFrameEvent event,
    caerFrameEventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>packet</i>	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

4.12.5.28 caerFrameEventsValid()

```
static bool caerFrameEventIsValid (
    caerFrameEventConst event ) [inline], [static]
```

Check if this frame event is valid.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

true if valid, false if not.

4.12.5.29 caerFrameEventPacketAllocate()

```
static caerFrameEventPacket caerFrameEventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow,
    int32_t maxLengthX,
    int32_t maxLengthY,
    int16_t maxChannelNumber ) [inline], [static]
```

Allocate a new frame events packet. Use free() to reclaim this memory. The frame events allocate memory for a maximum sized pixels array, depending on the parameters passed to this function, so that every event occupies the same amount of memory (constant size). The actual frames inside of it might be smaller than that, for example when using ROI, and their actual size is stored inside the frame event and should always be queried from there. The unused part of a pixels array is guaranteed to be zeros.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.
<i>maxLengthX</i>	the maximum expected X axis size for frames in this packet.
<i>maxLengthY</i>	the maximum expected Y axis size for frames in this packet.
<i>maxChannelNumber</i>	the maximum expected number of channels for frames in this packet.

Returns

a valid FrameEventPacket handle or NULL on error.

4.12.5.30 caerFrameEventPacketFromPacketHeader()

```
static caerFrameEventPacket caerFrameEventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```

Transform a generic event packet header into a Frame event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.12.5.31 caerFrameEventPacketFromPacketHeaderConst()

```
static caerFrameEventPacketConst caerFrameEventPacketFromPacketHeaderConst (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only Frame event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.12.5.32 caerFrameEventPacketGetEvent()

```
static caerFrameEvent caerFrameEventPacketGetEvent (
    caerFrameEventPacket packet,
    int32_t n ) [inline], [static]
```

Get the frame event at the given index from the event packet. To copy a frame event, the usual assignment operator = cannot be used. Please use [caerGenericEventCopy\(\)](#) to copy frame events!

Parameters

<i>packet</i>	a valid FrameEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested frame event. NULL on error.

4.12.5.33 caerFrameEventPacketGetEventConst()

```
static caerFrameEventConst caerFrameEventPacketGetEventConst (
    caerFrameEventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the frame event at the given index from the event packet. This is a read-only event, do not change its contents in any way! To copy a frame event, the usual assignment operator = cannot be used. Please use [caerGeneric↔EventCopy\(\)](#) to copy frame events!

Parameters

<i>packet</i>	a valid FrameEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only frame event. NULL on error.

4.12.5.34 caerFrameEventPacketGetPixelsMaxIndex()

```
static size_t caerFrameEventPacketGetPixelsMaxIndex (
    caerFrameEventPacketConst packet ) [inline], [static]
```

Get the maximum index into the pixels array, based upon how much memory was allocated to it by '[caerFrame↔EventPacketAllocate\(\)](#)'.

Parameters

<i>packet</i>	a valid FrameEventPacket pointer. Cannot be NULL.
---------------	---

Returns

maximum pixels array index.

4.12.5.35 caerFrameEventPacketGetPixelsSize()

```
static size_t caerFrameEventPacketGetPixelsSize (
    caerFrameEventPacketConst packet ) [inline], [static]
```

Get the maximum size of the pixels array in bytes, based upon how much memory was allocated to it by '[caer↔FrameEventPacketAllocate\(\)](#)'.

Parameters

<i>packet</i>	a valid FrameEventPacket pointer. Cannot be NULL.
---------------	---

Returns

maximum pixels array size in bytes.

4.12.5.36 caerFrameEventSetColorFilter()

```
static void caerFrameEventSetColorFilter (
    caerFrameEvent event,
    enum caer_frame_event_color_filter colorFilter ) [inline], [static]
```

Set the identifier for the color filter used by the sensor. Useful for interpolating color images.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>colorFilter</i>	color filter identifier.

4.12.5.37 caerFrameEventSetLengthXLengthYChannelNumber()

```
static void caerFrameEventSetLengthXLengthYChannelNumber (
    caerFrameEvent event,
    int32_t lengthX,
    int32_t lengthY,
    enum caer_frame_event_color_channels channelNumber,
    caerFrameEventPacketConst packet ) [inline], [static]
```

Set the X and Y axes length and the color channels number for a frame, while taking into account the maximum amount of memory available for the pixel array, as allocated in '[caerFrameEventPacketAllocate\(\)](#)'.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>lengthX</i>	the frame's X axis length.
<i>lengthY</i>	the frame's Y axis length.
<i>channelNumber</i>	the number of color channels for this frame.
<i>packet</i>	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

4.12.5.38 caerFrameEventSetPixel()

```
static void caerFrameEventSetPixel (
    caerFrameEvent event,
    int32_t xAddress,
    int32_t yAddress,
    uint16_t pixelValue ) [inline], [static]
```

Set the pixel value at the specified (X, Y) address. (X, Y) are checked against the actual possible values for this frame. Different channels are not taken into account! The (0, 0) pixel is in the upper left corner, like in OpenC↵V/computer graphics.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>xAddress</i>	X address value (checked).
<i>yAddress</i>	Y address value (checked).
<i>pixelValue</i>	pixel value (normalized to 16 bit depth).

4.12.5.39 caerFrameEventSetPixelForChannel()

```
static void caerFrameEventSetPixelForChannel (
    caerFrameEvent event,
    int32_t xAddress,
    int32_t yAddress,
    uint8_t channel,
    uint16_t pixelValue ) [inline], [static]
```

Set the pixel value at the specified (X, Y) address, taking into account the specified channel. (X, Y) and the channel number are checked against the actual possible values for this frame. The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>xAddress</i>	X address value (checked).
<i>yAddress</i>	Y address value (checked).
<i>channel</i>	the channel number (checked).
<i>pixelValue</i>	pixel value (normalized to 16 bit depth).

4.12.5.40 caerFrameEventSetPixelForChannelUnsafe()

```
static void caerFrameEventSetPixelForChannelUnsafe (
    caerFrameEvent event,
    int32_t xAddress,
```

```
int32_t yAddress,
uint8_t channel,
uint16_t pixelValue ) [inline], [static]
```

Set the pixel value at the specified (X, Y) address, taking into account the specified channel. No checks on (X, Y) and the channel number are performed! The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>xAddress</i>	X address value (unchecked).
<i>yAddress</i>	Y address value (unchecked).
<i>channel</i>	the channel number (unchecked).
<i>pixelValue</i>	pixel value (normalized to 16 bit depth).

4.12.5.41 caerFrameEventSetPixelUnsafe()

```
static void caerFrameEventSetPixelUnsafe (
    caerFrameEvent event,
    int32_t xAddress,
    int32_t yAddress,
    uint16_t pixelValue ) [inline], [static]
```

Set the pixel value at the specified (X, Y) address. No checks on (X, Y) are performed! The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>xAddress</i>	X address value (unchecked).
<i>yAddress</i>	Y address value (unchecked).
<i>pixelValue</i>	pixel value (normalized to 16 bit depth).

4.12.5.42 caerFrameEventSetPositionX()

```
static void caerFrameEventSetPositionX (
    caerFrameEvent event,
    int32_t positionX ) [inline], [static]
```

Set the X axis position offset. This is used to place partial frames, like the ones gotten from ROI readouts, in the visual space.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>positionX</i>	X axis position offset.

4.12.5.43 caerFrameEventSetPositionY()

```
static void caerFrameEventSetPositionY (
    caerFrameEvent event,
    int32_t positionY ) [inline], [static]
```

Set the Y axis position offset. This is used to place partial frames, like the ones gotten from ROI readouts, in the visual space.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>positionY</i>	Y axis position offset.

4.12.5.44 caerFrameEventSetROIIdentifier()

```
static void caerFrameEventSetROIIdentifier (
    caerFrameEvent event,
    uint8_t roiIdentifier ) [inline], [static]
```

Set the numerical identifier for the Region of Interest (ROI) region, to distinguish between multiple of them.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>roiIdentifier</i>	numerical ROI identifier.

4.12.5.45 caerFrameEventSetTSEndOfExposure()

```
static void caerFrameEventSetTSEndOfExposure (
    caerFrameEvent event,
    int32_t endExposure ) [inline], [static]
```

Set the 32bit end of exposure timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>endExposure</i>	a positive 32bit microsecond timestamp.

4.12.5.46 caerFrameEventSetTSEndOfFrame()

```
static void caerFrameEventSetTSEndOfFrame (
    caerFrameEvent event,
    int32_t endFrame ) [inline], [static]
```

Set the 32bit end of frame capture timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>endFrame</i>	a positive 32bit microsecond timestamp.

4.12.5.47 caerFrameEventSetTSStartOfExposure()

```
static void caerFrameEventSetTSStartOfExposure (
    caerFrameEvent event,
    int32_t startExposure ) [inline], [static]
```

Set the 32bit start of exposure timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>startExposure</i>	a positive 32bit microsecond timestamp.

4.12.5.48 caerFrameEventSetTSStartOfFrame()

```
static void caerFrameEventSetTSStartOfFrame (
    caerFrameEvent event,
    int32_t startFrame ) [inline], [static]
```

Set the 32bit start of frame capture timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>startFrame</i>	a positive 32bit microsecond timestamp.

4.12.5.49 caerFrameEventValidate()

```
static void caerFrameEventValidate (
```

```

    caerFrameEvent event,
    caerFrameEventPacket packet ) [inline], [static]

```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
<i>packet</i>	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

4.12.5.50 PACKED_STRUCT() [1/2]

```

PACKED_STRUCT (
    struct caer_frame_event { uint32_t info;int32_t ts_startframe;int32_t ts_endframe;int32_t ts_startexposure;int32_t ts_endexposure;int32_t lengthX;int32_t lengthY;int32_t positionX;int32_t positionY;uint16_t pixels[1];} )

```

Frame event data structure definition. This contains the actual information on the frame (ROI, color channels, color filter), several timestamps to signal start and end of capture and of exposure, as well as the actual pixels, in a 16 bit normalized format. The (0, 0) address is in the upper left corner, like in OpenCV/computer graphics. The pixel array is laid out row by row (increasing X axis), going from top to bottom (increasing Y axis). Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java. To copy a frame event, the usual assignment operator = cannot be used. Please use [caerGenericEventCopy\(\)](#) to copy frame events!

4.12.5.51 PACKED_STRUCT() [2/2]

```

PACKED_STRUCT (
    struct caer_frame_event_packet { struct caer_event_packet_header packetHeader;}
)

```

Frame event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block. Direct access to the events array is not possible for Frame events. To calculate position offsets, use the 'eventSize' field in the packet header.

4.13 events/imu6.h File Reference

```
#include "common.h"
```

Macros

- `#define CAER_IMU6_ITERATOR_ALL_START(IMU6_PACKET)`
- `#define CAER_IMU6_CONST_ITERATOR_ALL_START(IMU6_PACKET)`
- `#define CAER_IMU6_ITERATOR_ALL_END }`
- `#define CAER_IMU6_ITERATOR_VALID_START(IMU6_PACKET)`
- `#define CAER_IMU6_CONST_ITERATOR_VALID_START(IMU6_PACKET)`
- `#define CAER_IMU6_ITERATOR_VALID_END }`
- `#define CAER_IMU6_REVERSE_ITERATOR_ALL_START(IMU6_PACKET)`
- `#define CAER_IMU6_CONST_REVERSE_ITERATOR_ALL_START(IMU6_PACKET)`
- `#define CAER_IMU6_REVERSE_ITERATOR_ALL_END }`
- `#define CAER_IMU6_REVERSE_ITERATOR_VALID_START(IMU6_PACKET)`
- `#define CAER_IMU6_CONST_REVERSE_ITERATOR_VALID_START(IMU6_PACKET)`
- `#define CAER_IMU6_REVERSE_ITERATOR_VALID_END }`

Typedefs

- `typedef struct caer_imu6_event * caerIMU6Event`
- `typedef const struct caer_imu6_event * caerIMU6EventConst`
- `typedef struct caer_imu6_event_packet * caerIMU6EventPacket`
- `typedef const struct caer_imu6_event_packet * caerIMU6EventPacketConst`

Functions

- `PACKED_STRUCT` (struct caer_imu6_event { uint32_t info;int32_t timestamp;float accel_x;float accel_y;float accel_z;float gyro_x;float gyro_y;float gyro_z;float temp;})
- `PACKED_STRUCT` (struct caer_imu6_event_packet { struct caer_event_packet_header packetHeader;struct caer_imu6_event events[];})
- static `caerIMU6EventPacket caerIMU6EventPacketAllocate` (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static `caerIMU6EventPacket caerIMU6EventPacketFromPacketHeader` (caerEventPacketHeader header)
- static `caerIMU6EventPacketConst caerIMU6EventPacketFromPacketHeaderConst` (caerEventPacketHeaderConst header)
- static `caerIMU6Event caerIMU6EventPacketGetEvent` (caerIMU6EventPacket packet, int32_t n)
- static `caerIMU6EventConst caerIMU6EventPacketGetEventConst` (caerIMU6EventPacketConst packet, int32_t n)
- static `int32_t caerIMU6EventGetTimestamp` (caerIMU6EventConst event)
- static `int64_t caerIMU6EventGetTimestamp64` (caerIMU6EventConst event, caerIMU6EventPacketConst packet)
- static void `caerIMU6EventSetTimestamp` (caerIMU6Event event, int32_t timestamp)
- static bool `caerIMU6EventIsValid` (caerIMU6EventConst event)
- static void `caerIMU6EventValidate` (caerIMU6Event event, caerIMU6EventPacket packet)
- static void `caerIMU6EventInvalidate` (caerIMU6Event event, caerIMU6EventPacket packet)
- static float `caerIMU6EventGetAccelX` (caerIMU6EventConst event)
- static void `caerIMU6EventSetAccelX` (caerIMU6Event event, float accelX)
- static float `caerIMU6EventGetAccelY` (caerIMU6EventConst event)
- static void `caerIMU6EventSetAccelY` (caerIMU6Event event, float accelY)
- static float `caerIMU6EventGetAccelZ` (caerIMU6EventConst event)
- static void `caerIMU6EventSetAccelZ` (caerIMU6Event event, float accelZ)
- static float `caerIMU6EventGetGyroX` (caerIMU6EventConst event)
- static void `caerIMU6EventSetGyroX` (caerIMU6Event event, float gyroX)
- static float `caerIMU6EventGetGyroY` (caerIMU6EventConst event)
- static void `caerIMU6EventSetGyroY` (caerIMU6Event event, float gyroY)
- static float `caerIMU6EventGetGyroZ` (caerIMU6EventConst event)
- static void `caerIMU6EventSetGyroZ` (caerIMU6Event event, float gyroZ)
- static float `caerIMU6EventGetTemp` (caerIMU6EventConst event)
- static void `caerIMU6EventSetTemp` (caerIMU6Event event, float temp)

4.13.1 Detailed Description

IMU6 (6 axes) Events format definition and handling functions. This contains data coming from the Inertial Measurement Unit chip, with the 3-axes accelerometer and 3-axes gyroscope. Temperature is also included.

4.13.2 Macro Definition Documentation

4.13.2.1 CAER_IMU6_CONST_ITERATOR_ALL_START

```
#define CAER_IMU6_CONST_ITERATOR_ALL_START (
    IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = 0; \
     caerIMU6IteratorCounter < caerEventPacketHeaderGetEventNumber (& \
     (IMU6_PACKET)->packetHeader); \
     caerIMU6IteratorCounter++) { \
    caerIMU6EventConst caerIMU6IteratorElement \
    = caerIMU6EventPacketGetEventConst (IMU6_PACKET, \
    caerIMU6IteratorCounter);
```

Const-Iterator over all IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU6IteratorElement' variable of type caerIMU6EventConst.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.2 CAER_IMU6_CONST_ITERATOR_VALID_START

```
#define CAER_IMU6_CONST_ITERATOR_VALID_START (
    IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = 0; \
     caerIMU6IteratorCounter < caerEventPacketHeaderGetEventNumber (& \
     (IMU6_PACKET)->packetHeader); \
     caerIMU6IteratorCounter++) { \
    caerIMU6EventConst caerIMU6IteratorElement \
    = caerIMU6EventPacketGetEventConst (IMU6_PACKET, \
    caerIMU6IteratorCounter); \
    if (!caerIMU6EventIsValid(caerIMU6IteratorElement)) { \
        continue; \
    }
```

Const-Iterator over only the valid IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU6IteratorElement' variable of type caerIMU6EventConst.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.3 CAER_IMU6_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_IMU6_CONST_REVERSE_ITERATOR_ALL_START(  
    IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(IMU6_PACKET)->packetHeader) - 1; \  
    caerIMU6IteratorCounter >= 0; caerIMU6IteratorCounter--) {  
    \  
    caerIMU6EventConst caerIMU6IteratorElement  
    \  
    = caerIMU6EventPacketGetEventConst (IMU6_PACKET,  
    caerIMU6IteratorCounter);
```

Const-Reverse iterator over all IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU6IteratorElement' variable of type caerIMU6EventConst.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.4 CAER_IMU6_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_IMU6_CONST_REVERSE_ITERATOR_VALID_START(  
    IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(IMU6_PACKET)->packetHeader) - 1; \  
    caerIMU6IteratorCounter >= 0; caerIMU6IteratorCounter--) {  
    \  
    caerIMU6EventConst caerIMU6IteratorElement  
    \  
    = caerIMU6EventPacketGetEventConst (IMU6_PACKET,  
    caerIMU6IteratorCounter);  
    if (!caerIMU6EventIsValid(caerIMU6IteratorElement)) {  
        \  
        continue;  
    }  
    \  
}
```

Const-Reverse iterator over only the valid IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU6IteratorElement' variable of type caerIMU6EventConst.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.5 CAER_IMU6_ITERATOR_ALL_END

```
#define CAER_IMU6_ITERATOR_ALL_END }
```

Iterator close statement.

4.13.2.6 CAER_IMU6_ITERATOR_ALL_START

```
#define CAER_IMU6_ITERATOR_ALL_START(  
    IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = 0; \
     caerIMU6IteratorCounter < caerEventPacketHeaderGetEventNumber(&  
    (IMU6_PACKET)->packetHeader); \
     caerIMU6IteratorCounter++) { \
    caerIMU6Event caerIMU6IteratorElement = caerIMU6EventPacketGetEvent(  
    IMU6_PACKET, caerIMU6IteratorCounter);
```

Iterator over all IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU6IteratorElement' variable of type caerIMU6Event.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.7 CAER_IMU6_ITERATOR_VALID_END

```
#define CAER_IMU6_ITERATOR_VALID_END }
```

Iterator close statement.

4.13.2.8 CAER_IMU6_ITERATOR_VALID_START

```
#define CAER_IMU6_ITERATOR_VALID_START(  
    IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = 0; \
     caerIMU6IteratorCounter < caerEventPacketHeaderGetEventNumber(&  
    (IMU6_PACKET)->packetHeader); \
     caerIMU6IteratorCounter++) { \
    caerIMU6Event caerIMU6IteratorElement = caerIMU6EventPacketGetEvent(  
    IMU6_PACKET, caerIMU6IteratorCounter); \
    if (!caerIMU6EventIsValid(caerIMU6IteratorElement)) { \
        continue; \
    }
```

Iterator over only the valid IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU6IteratorElement' variable of type caerIMU6Event.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.9 CAER_IMU6_REVERSE_ITERATOR_ALL_END

```
#define CAER_IMU6_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.13.2.10 CAER_IMU6_REVERSE_ITERATOR_ALL_START

```
#define CAER_IMU6_REVERSE_ITERATOR_ALL_START (
    IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = caerEventPacketHeaderGetEventNumber
    (&(IMU6_PACKET)->packetHeader) - 1; \
    caerIMU6IteratorCounter >= 0; caerIMU6IteratorCounter--) {
    \
    caerIMU6Event caerIMU6IteratorElement = caerIMU6EventPacketGetEvent (
    IMU6_PACKET, caerIMU6IteratorCounter);
```

Reverse iterator over all IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU6IteratorElement' variable of type caerIMU6Event.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.11 CAER_IMU6_REVERSE_ITERATOR_VALID_END

```
#define CAER_IMU6_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.13.2.12 CAER_IMU6_REVERSE_ITERATOR_VALID_START

```
#define CAER_IMU6_REVERSE_ITERATOR_VALID_START (
    IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = caerEventPacketHeaderGetEventNumber
    (&(IMU6_PACKET)->packetHeader) - 1; \
    caerIMU6IteratorCounter >= 0; caerIMU6IteratorCounter--) {
    \
    caerIMU6Event caerIMU6IteratorElement = caerIMU6EventPacketGetEvent (
    IMU6_PACKET, caerIMU6IteratorCounter); \
    if (!caerIMU6EventIsValid(caerIMU6IteratorElement)) {
        \
        continue;
    }
}
```

Reverse iterator over only the valid IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU6IteratorElement' variable of type caerIMU6Event.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.3 Typedef Documentation

4.13.3.1 caerIMU6Event

```
typedef struct caer_imu6_event* caerIMU6Event
```

Type for pointer to IMU 6-axes event data structure.

4.13.3.2 caerIMU6EventPacket

```
typedef struct caer_imu6_event_packet* caerIMU6EventPacket
```

Type for pointer to IMU 6-axes event packet data structure.

4.13.4 Function Documentation

4.13.4.1 caerIMU6EventGetAccelX()

```
static float caerIMU6EventGetAccelX (  
    caerIMU6EventConst event ) [inline], [static]
```

Get the X axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
--------------	--

Returns

acceleration on the X axis.

4.13.4.2 caerIMU6EventGetAccelY()

```
static float caerIMU6EventGetAccelY (  
    caerIMU6EventConst event ) [inline], [static]
```

Get the Y axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
--------------	--

Returns

acceleration on the Y axis.

4.13.4.3 caerIMU6EventGetAccelZ()

```
static float caerIMU6EventGetAccelZ (  
    caerIMU6EventConst event ) [inline], [static]
```

Get the Z axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
--------------	--

Returns

acceleration on the Z axis.

4.13.4.4 caerIMU6EventGetGyroX()

```
static float caerIMU6EventGetGyroX (  
    caerIMU6EventConst event ) [inline], [static]
```

Get the X axis (roll) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
--------------	--

Returns

angular velocity on the X axis (roll).

4.13.4.5 caerIMU6EventGetGyroY()

```
static float caerIMU6EventGetGyroY (  
    caerIMU6EventConst event ) [inline], [static]
```

Get the Y axis (pitch) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
--------------	--

Returns

angular velocity on the Y axis (pitch).

4.13.4.6 caerIMU6EventGetGyroZ()

```
static float caerIMU6EventGetGyroZ (  
    caerIMU6EventConst event ) [inline], [static]
```

Get the Z axis (yaw) angular velocity reading (from gyroscope). This is in %s (deg/sec).

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
--------------	--

Returns

angular velocity on the Z axis (yaw).

4.13.4.7 caerIMU6EventGetTemp()

```
static float caerIMU6EventGetTemp (  
    caerIMU6EventConst event ) [inline], [static]
```

Get the temperature reading. This is in °C.

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
--------------	--

Returns

temperature in °C.

4.13.4.8 caerIMU6EventGetTimestamp()

```
static int32_t caerIMU6EventGetTimestamp (  
    caerIMU6EventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
--------------	--

Returns

this event's 32bit microsecond timestamp.

4.13.4.9 caerIMU6EventGetTimestamp64()

```
static int64_t caerIMU6EventGetTimestamp64 (  
    caerIMU6EventConst event,  
    caerIMU6EventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
<i>packet</i>	the IMU6EventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.13.4.10 caerIMU6EventInvalidate()

```
static void caerIMU6EventInvalidate (  
    caerIMU6Event event,  
    caerIMU6EventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
<i>packet</i>	the IMU6EventPacket pointer for the packet containing this event. Cannot be NULL.

4.13.4.11 caerIMU6EventIsValid()

```
static bool caerIMU6EventIsValid (
    caerIMU6EventConst event ) [inline], [static]
```

Check if this IMU 6-axes event is valid.

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
--------------	--

Returns

true if valid, false if not.

4.13.4.12 caerIMU6EventPacketAllocate()

```
static caerIMU6EventPacket caerIMU6EventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]
```

Allocate a new IMU 6-axes events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid IMU6EventPacket handle or NULL on error.

4.13.4.13 caerIMU6EventPacketFromPacketHeader()

```
static caerIMU6EventPacket caerIMU6EventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```

Transform a generic event packet header into an IMU 6-axes event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.13.4.14 caerIMU6EventPacketFromPacketHeaderConst()

```
static caerIMU6EventPacketConst caerIMU6EventPacketFromPacketHeaderConst (  
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only IMU 6-axes event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.13.4.15 caerIMU6EventPacketGetEvent()

```
static caerIMU6Event caerIMU6EventPacketGetEvent (  
    caerIMU6EventPacket packet,  
    int32_t n ) [inline], [static]
```

Get the IMU 6-axes event at the given index from the event packet.

Parameters

<i>packet</i>	a valid IMU6EventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested IMU 6-axes event. NULL on error.

4.13.4.16 caerIMU6EventPacketGetEventConst()

```
static caerIMU6EventConst caerIMU6EventPacketGetEventConst (
    caerIMU6EventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the IMU 6-axes event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid IMU6EventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only IMU 6-axes event. NULL on error.

4.13.4.17 caerIMU6EventSetAccelX()

```
static void caerIMU6EventSetAccelX (
    caerIMU6Event event,
    float accelX ) [inline], [static]
```

Set the X axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
<i>accelX</i>	acceleration on the X axis.

4.13.4.18 caerIMU6EventSetAccelY()

```
static void caerIMU6EventSetAccelY (
    caerIMU6Event event,
    float accelY ) [inline], [static]
```

Set the Y axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
<i>accelY</i>	acceleration on the Y axis.

4.13.4.19 caerIMU6EventSetAccelZ()

```
static void caerIMU6EventSetAccelZ (
    caerIMU6Event event,
    float accelZ ) [inline], [static]
```

Set the Z axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
<i>accelZ</i>	acceleration on the Z axis.

4.13.4.20 caerIMU6EventSetGyroX()

```
static void caerIMU6EventSetGyroX (
    caerIMU6Event event,
    float gyroX ) [inline], [static]
```

Set the X axis (roll) angular velocity reading (from gyroscope). This is in %s (deg/sec).

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
<i>gyroX</i>	angular velocity on the X axis (roll).

4.13.4.21 caerIMU6EventSetGyroY()

```
static void caerIMU6EventSetGyroY (
    caerIMU6Event event,
    float gyroY ) [inline], [static]
```

Set the Y axis (pitch) angular velocity reading (from gyroscope). This is in %s (deg/sec).

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
<i>gyroY</i>	angular velocity on the Y axis (pitch).

4.13.4.22 caerIMU6EventSetGyroZ()

```
static void caerIMU6EventSetGyroZ (
```

```

    caerIMU6Event event,
    float gyroZ ) [inline], [static]

```

Set the Z axis (yaw) angular velocity reading (from gyroscope). This is in %s (deg/sec).

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
<i>gyroZ</i>	angular velocity on the Z axis (yaw).

4.13.4.23 caerIMU6EventSetTemp()

```

static void caerIMU6EventSetTemp (
    caerIMU6Event event,
    float temp ) [inline], [static]

```

Set the temperature reading. This is in °C.

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
<i>temp</i>	temperature in °C.

4.13.4.24 caerIMU6EventSetTimestamp()

```

static void caerIMU6EventSetTimestamp (
    caerIMU6Event event,
    int32_t timestamp ) [inline], [static]

```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.13.4.25 caerIMU6EventValidate()

```

static void caerIMU6EventValidate (
    caerIMU6Event event,
    caerIMU6EventPacket packet ) [inline], [static]

```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid IMU6Event pointer. Cannot be NULL.
<i>packet</i>	the IMU6EventPacket pointer for the packet containing this event. Cannot be NULL.

4.13.4.26 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
    struct caer_imu6_event { uint32_t info;int32_t timestamp;float accel_x;float
    accel_y;float accel_z;float gyro_x;float gyro_y;float gyro_z;float temp;} )
```

IMU 6-axes event data structure definition. This contains accelerometer and gyroscope headings, plus temperature. The X, Y and Z axes are referred to the camera plane. X increases to the right, Y going up and Z towards where the lens is pointing. Rotation for the gyroscope is counter-clockwise along the increasing axis, for all three axes. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.13.4.27 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
    struct caer_imu6_event_packet { struct caer_event_packet_header packetHeader;struct
    caer_imu6_event events[];} )
```

IMU 6-axes event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.14 events/imu9.h File Reference

```
#include "common.h"
```

Macros

- `#define CAER_IMU9_ITERATOR_ALL_START(IMU9_PACKET)`
- `#define CAER_IMU9_CONST_ITERATOR_ALL_START(IMU9_PACKET)`
- `#define CAER_IMU9_ITERATOR_ALL_END }`
- `#define CAER_IMU9_ITERATOR_VALID_START(IMU9_PACKET)`
- `#define CAER_IMU9_CONST_ITERATOR_VALID_START(IMU9_PACKET)`
- `#define CAER_IMU9_ITERATOR_VALID_END }`
- `#define CAER_IMU9_REVERSE_ITERATOR_ALL_START(IMU9_PACKET)`
- `#define CAER_IMU9_CONST_REVERSE_ITERATOR_ALL_START(IMU9_PACKET)`
- `#define CAER_IMU9_REVERSE_ITERATOR_ALL_END }`
- `#define CAER_IMU9_REVERSE_ITERATOR_VALID_START(IMU9_PACKET)`
- `#define CAER_IMU9_CONST_REVERSE_ITERATOR_VALID_START(IMU9_PACKET)`
- `#define CAER_IMU9_REVERSE_ITERATOR_VALID_END }`

Typedefs

- typedef struct caer_imu9_event * [caerIMU9Event](#)
- typedef const struct caer_imu9_event * **caerIMU9EventConst**
- typedef struct caer_imu9_event_packet * [caerIMU9EventPacket](#)
- typedef const struct caer_imu9_event_packet * **caerIMU9EventPacketConst**

Functions

- [PACKED_STRUCT](#) (struct caer_imu9_event { uint32_t info;int32_t timestamp;float accel_x;float accel_y;float accel_z;float gyro_x;float gyro_y;float gyro_z;float temp;float comp_x;float comp_y;float comp_z;})
- [PACKED_STRUCT](#) (struct caer_imu9_event_packet { struct caer_event_packet_header packetHeader;struct caer_imu9_event events[];})
- static [caerIMU9EventPacket](#) [caerIMU9EventPacketAllocate](#) (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static [caerIMU9EventPacket](#) [caerIMU9EventPacketFromPacketHeader](#) ([caerEventPacketHeader](#) header)
- static caerIMU9EventPacketConst [caerIMU9EventPacketFromPacketHeaderConst](#) ([caerEventPacketHeaderConst](#) header)
- static [caerIMU9Event](#) [caerIMU9EventPacketGetEvent](#) ([caerIMU9EventPacket](#) packet, int32_t n)
- static caerIMU9EventConst [caerIMU9EventPacketGetEventConst](#) ([caerIMU9EventPacketConst](#) packet, int32_t n)
- static int32_t [caerIMU9EventGetTimestamp](#) ([caerIMU9EventConst](#) event)
- static int64_t [caerIMU9EventGetTimestamp64](#) ([caerIMU9EventConst](#) event, [caerIMU9EventPacketConst](#) packet)
- static void [caerIMU9EventSetTimestamp](#) ([caerIMU9Event](#) event, int32_t timestamp)
- static bool [caerIMU9EventIsValid](#) ([caerIMU9EventConst](#) event)
- static void [caerIMU9EventValidate](#) ([caerIMU9Event](#) event, [caerIMU9EventPacket](#) packet)
- static void [caerIMU9EventInvalidate](#) ([caerIMU9Event](#) event, [caerIMU9EventPacket](#) packet)
- static float [caerIMU9EventGetAccelX](#) ([caerIMU9EventConst](#) event)
- static void [caerIMU9EventSetAccelX](#) ([caerIMU9Event](#) event, float accelX)
- static float [caerIMU9EventGetAccelY](#) ([caerIMU9EventConst](#) event)
- static void [caerIMU9EventSetAccelY](#) ([caerIMU9Event](#) event, float accelY)
- static float [caerIMU9EventGetAccelZ](#) ([caerIMU9EventConst](#) event)
- static void [caerIMU9EventSetAccelZ](#) ([caerIMU9Event](#) event, float accelZ)
- static float [caerIMU9EventGetGyroX](#) ([caerIMU9EventConst](#) event)
- static void [caerIMU9EventSetGyroX](#) ([caerIMU9Event](#) event, float gyroX)
- static float [caerIMU9EventGetGyroY](#) ([caerIMU9EventConst](#) event)
- static void [caerIMU9EventSetGyroY](#) ([caerIMU9Event](#) event, float gyroY)
- static float [caerIMU9EventGetGyroZ](#) ([caerIMU9EventConst](#) event)
- static void [caerIMU9EventSetGyroZ](#) ([caerIMU9Event](#) event, float gyroZ)
- static float [caerIMU9EventGetTemp](#) ([caerIMU9EventConst](#) event)
- static void [caerIMU9EventSetTemp](#) ([caerIMU9Event](#) event, float temp)
- static float [caerIMU9EventGetCompX](#) ([caerIMU9EventConst](#) event)
- static void [caerIMU9EventSetCompX](#) ([caerIMU9Event](#) event, float compX)
- static float [caerIMU9EventGetCompY](#) ([caerIMU9EventConst](#) event)
- static void [caerIMU9EventSetCompY](#) ([caerIMU9Event](#) event, float compY)
- static float [caerIMU9EventGetCompZ](#) ([caerIMU9EventConst](#) event)
- static void [caerIMU9EventSetCompZ](#) ([caerIMU9Event](#) event, float compZ)

4.14.1 Detailed Description

IMU9 (9 axes) Events format definition and handling functions. This contains data coming from the Inertial Measurement Unit chip, with the 3-axes accelerometer and 3-axes gyroscope. Temperature is also included. Further, 3-axes from the magnetometer are included, which can be used to get a compass-like heading.

4.14.2 Macro Definition Documentation

4.14.2.1 CAER_IMU9_CONST_ITERATOR_ALL_START

```
#define CAER_IMU9_CONST_ITERATOR_ALL_START(  
    IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = 0; \
     caerIMU9IteratorCounter < caerEventPacketHeaderGetEventNumber (&  
     (IMU9_PACKET)->packetHeader); \
     caerIMU9IteratorCounter++) { \
     caerIMU9EventConst caerIMU9IteratorElement \
     = caerIMU9EventPacketGetEventConst (IMU9_PACKET,  
     caerIMU9IteratorCounter);
```

Const-Iterator over all IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU9IteratorElement' variable of type caerIMU9EventConst.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.2 CAER_IMU9_CONST_ITERATOR_VALID_START

```
#define CAER_IMU9_CONST_ITERATOR_VALID_START(  
    IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = 0; \
     caerIMU9IteratorCounter < caerEventPacketHeaderGetEventNumber (&  
     (IMU9_PACKET)->packetHeader); \
     caerIMU9IteratorCounter++) { \
     caerIMU9EventConst caerIMU9IteratorElement \
     = caerIMU9EventPacketGetEventConst (IMU9_PACKET,  
     caerIMU9IteratorCounter); \
     if (!caerIMU9EventIsValid(caerIMU9IteratorElement)) { \
         continue; \
     }
```

Const-Iterator over only the valid IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU9IteratorElement' variable of type caerIMU9EventConst.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.3 CAER_IMU9_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_IMU9_CONST_REVERSE_ITERATOR_ALL_START(  
    IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(IMU9_PACKET)->packetHeader) - 1; \  
    caerIMU9IteratorCounter >= 0; caerIMU9IteratorCounter--) {  
    \  
    caerIMU9EventConst caerIMU9IteratorElement  
    \  
    = caerIMU9EventPacketGetEventConst (IMU9_PACKET,  
    caerIMU9IteratorCounter);
```

Const-Reverse iterator over all IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU9IteratorElement' variable of type caerIMU9EventConst.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.4 CAER_IMU9_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_IMU9_CONST_REVERSE_ITERATOR_VALID_START(  
    IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(IMU9_PACKET)->packetHeader) - 1; \  
    caerIMU9IteratorCounter >= 0; caerIMU9IteratorCounter--) {  
    \  
    caerIMU9EventConst caerIMU9IteratorElement  
    \  
    = caerIMU9EventPacketGetEventConst (IMU9_PACKET,  
    caerIMU9IteratorCounter);  
    if (!caerIMU9EventIsValid(caerIMU9IteratorElement)) {  
        \  
        continue;  
    }  
    \  
}
```

Const-Reverse iterator over only the valid IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU9IteratorElement' variable of type caerIMU9EventConst.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.5 CAER_IMU9_ITERATOR_ALL_END

```
#define CAER_IMU9_ITERATOR_ALL_END }
```

Iterator close statement.

4.14.2.6 CAER_IMU9_ITERATOR_ALL_START

```
#define CAER_IMU9_ITERATOR_ALL_START(
    IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = 0;
     caerIMU9IteratorCounter < caerEventPacketHeaderGetEventNumber(&
     (IMU9_PACKET)->packetHeader); \
     caerIMU9IteratorCounter++) {
    caerIMU9Event caerIMU9IteratorElement = caerIMU9EventPacketGetEvent(
    IMU9_PACKET, caerIMU9IteratorCounter);
```

Iterator over all IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU9IteratorElement' variable of type caerIMU9Event.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.7 CAER_IMU9_ITERATOR_VALID_END

```
#define CAER_IMU9_ITERATOR_VALID_END }
```

Iterator close statement.

4.14.2.8 CAER_IMU9_ITERATOR_VALID_START

```
#define CAER_IMU9_ITERATOR_VALID_START(
    IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = 0;
     caerIMU9IteratorCounter < caerEventPacketHeaderGetEventNumber(&
     (IMU9_PACKET)->packetHeader); \
     caerIMU9IteratorCounter++) {
    caerIMU9Event caerIMU9IteratorElement = caerIMU9EventPacketGetEvent(
    IMU9_PACKET, caerIMU9IteratorCounter); \
    if (!caerIMU9EventIsValid(caerIMU9IteratorElement)) {
        continue;
    }
```

Iterator over only the valid IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU9IteratorElement' variable of type caerIMU9Event.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.9 CAER_IMU9_REVERSE_ITERATOR_ALL_END

```
#define CAER_IMU9_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.14.2.10 CAER_IMU9_REVERSE_ITERATOR_ALL_START

```
#define CAER_IMU9_REVERSE_ITERATOR_ALL_START(
    IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = caerEventPacketHeaderGetEventNumber
    (&(IMU9_PACKET)->packetHeader) - 1; \
    caerIMU9IteratorCounter >= 0; caerIMU9IteratorCounter--) {
    \
    caerIMU9Event caerIMU9IteratorElement = caerIMU9EventPacketGetEvent(
    IMU9_PACKET, caerIMU9IteratorCounter);
```

Reverse iterator over all IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU9IteratorElement' variable of type caerIMU9Event.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.11 CAER_IMU9_REVERSE_ITERATOR_VALID_END

```
#define CAER_IMU9_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.14.2.12 CAER_IMU9_REVERSE_ITERATOR_VALID_START

```
#define CAER_IMU9_REVERSE_ITERATOR_VALID_START(
    IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = caerEventPacketHeaderGetEventNumber
    (&(IMU9_PACKET)->packetHeader) - 1; \
    caerIMU9IteratorCounter >= 0; caerIMU9IteratorCounter--) {
    \
    caerIMU9Event caerIMU9IteratorElement = caerIMU9EventPacketGetEvent(
    IMU9_PACKET, caerIMU9IteratorCounter); \
    if (!caerIMU9EventIsValid(caerIMU9IteratorElement)) {
        \
        continue;
    }
}
```

Reverse iterator over only the valid IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU9IteratorElement' variable of type caerIMU9Event.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.3 Typedef Documentation

4.14.3.1 caerIMU9Event

```
typedef struct caer_imu9_event* caerIMU9Event
```

Type for pointer to IMU 9-axes event data structure.

4.14.3.2 caerIMU9EventPacket

```
typedef struct caer_imu9_event_packet* caerIMU9EventPacket
```

Type for pointer to IMU 9-axes event packet data structure.

4.14.4 Function Documentation

4.14.4.1 caerIMU9EventGetAccelX()

```
static float caerIMU9EventGetAccelX (  
    caerIMU9EventConst event ) [inline], [static]
```

Get the X axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

acceleration on the X axis.

4.14.4.2 caerIMU9EventGetAccelY()

```
static float caerIMU9EventGetAccelY (  
    caerIMU9EventConst event ) [inline], [static]
```

Get the Y axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

acceleration on the Y axis.

4.14.4.3 caerIMU9EventGetAccelZ()

```
static float caerIMU9EventGetAccelZ (  
    caerIMU9EventConst event ) [inline], [static]
```

Get the Z axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

acceleration on the Z axis.

4.14.4.4 caerIMU9EventGetCompX()

```
static float caerIMU9EventGetCompX (  
    caerIMU9EventConst event ) [inline], [static]
```

Get the X axis compass heading (from magnetometer). This is in μ T.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

X axis compass heading.

4.14.4.5 caerIMU9EventGetCompY()

```
static float caerIMU9EventGetCompY (  
    caerIMU9EventConst event ) [inline], [static]
```

Get the Y axis compass heading (from magnetometer). This is in μ T.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

Y axis compass heading.

4.14.4.6 caerIMU9EventGetCompZ()

```
static float caerIMU9EventGetCompZ (  
    caerIMU9EventConst event ) [inline], [static]
```

Get the Z axis compass heading (from magnetometer). This is in μT .

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

Z axis compass heading.

4.14.4.7 caerIMU9EventGetGyroX()

```
static float caerIMU9EventGetGyroX (  
    caerIMU9EventConst event ) [inline], [static]
```

Get the X axis (roll) angular velocity reading (from gyroscope). This is in $^\circ/\text{s}$ (deg/sec).

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

angular velocity on the X axis (roll).

4.14.4.8 caerIMU9EventGetGyroY()

```
static float caerIMU9EventGetGyroY (  
    caerIMU9EventConst event ) [inline], [static]
```

Get the Y axis (pitch) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

angular velocity on the Y axis (pitch).

4.14.4.9 caerIMU9EventGetGyroZ()

```
static float caerIMU9EventGetGyroZ (  
    caerIMU9EventConst event ) [inline], [static]
```

Get the Z axis (yaw) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

angular velocity on the Z axis (yaw).

4.14.4.10 caerIMU9EventGetTemp()

```
static float caerIMU9EventGetTemp (  
    caerIMU9EventConst event ) [inline], [static]
```

Get the temperature reading. This is in °C.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

temperature in °C.

4.14.4.11 caerIMU9EventGetTimestamp()

```
static int32_t caerIMU9EventGetTimestamp (  
    caerIMU9EventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

this event's 32bit microsecond timestamp.

4.14.4.12 caerIMU9EventGetTimestamp64()

```
static int64_t caerIMU9EventGetTimestamp64 (
    caerIMU9EventConst event,
    caerIMU9EventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>packet</i>	the IMU9EventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.14.4.13 caerIMU9EventInvalidate()

```
static void caerIMU9EventInvalidate (
    caerIMU9Event event,
    caerIMU9EventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>packet</i>	the IMU9EventPacket pointer for the packet containing this event. Cannot be NULL.

4.14.4.14 caerIMU9EventsValid()

```
static bool caerIMU9EventIsValid (
    caerIMU9EventConst event ) [inline], [static]
```

Check if this IMU 9-axes event is valid.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

true if valid, false if not.

4.14.4.15 caerIMU9EventPacketAllocate()

```
static caerIMU9EventPacket caerIMU9EventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]
```

Allocate a new IMU 9-axes events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid IMU9EventPacket handle or NULL on error.

4.14.4.16 caerIMU9EventPacketFromPacketHeader()

```
static caerIMU9EventPacket caerIMU9EventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```

Transform a generic event packet header into an IMU 9-axes event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.14.4.17 caerIMU9EventPacketFromPacketHeaderConst()

```
static caerIMU9EventPacketConst caerIMU9EventPacketFromPacketHeaderConst (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only IMU 9-axes event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.14.4.18 caerIMU9EventPacketGetEvent()

```
static caerIMU9Event caerIMU9EventPacketGetEvent (
    caerIMU9EventPacket packet,
    int32_t n ) [inline], [static]
```

Get the IMU 9-axes event at the given index from the event packet.

Parameters

<i>packet</i>	a valid IMU9EventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested IMU 9-axes event. NULL on error.

4.14.4.19 caerIMU9EventPacketGetEventConst()

```
static caerIMU9EventConst caerIMU9EventPacketGetEventConst (
    caerIMU9EventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the IMU 9-axes event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid IMU9EventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only IMU 9-axes event. NULL on error.

4.14.4.20 caerIMU9EventSetAccelX()

```
static void caerIMU9EventSetAccelX (
    caerIMU9Event event,
    float accelX ) [inline], [static]
```

Set the X axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>accelX</i>	acceleration on the X axis.

4.14.4.21 caerIMU9EventSetAccelY()

```
static void caerIMU9EventSetAccelY (
    caerIMU9Event event,
    float accelY ) [inline], [static]
```

Set the Y axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>accelY</i>	acceleration on the Y axis.

4.14.4.22 caerIMU9EventSetAccelZ()

```
static void caerIMU9EventSetAccelZ (
    caerIMU9Event event,
    float accelZ ) [inline], [static]
```

Set the Z axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>accelZ</i>	acceleration on the Z axis.

4.14.4.23 caerIMU9EventSetCompX()

```
static void caerIMU9EventSetCompX (
    caerIMU9Event event,
    float compX ) [inline], [static]
```

Set the X axis compass heading (from magnetometer). This is in μ T.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>compX</i>	X axis compass heading.

4.14.4.24 caerIMU9EventSetCompY()

```
static void caerIMU9EventSetCompY (
    caerIMU9Event event,
    float compY ) [inline], [static]
```

Set the Y axis compass heading (from magnetometer). This is in μ T.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>compY</i>	Y axis compass heading.

4.14.4.25 caerIMU9EventSetCompZ()

```
static void caerIMU9EventSetCompZ (
```

```
caerIMU9Event event,  
float compZ ) [inline], [static]
```

Set the Z axis compass heading (from magnetometer). This is in μT .

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>compZ</i>	Z axis compass heading.

4.14.4.26 caerIMU9EventSetGyroX()

```
static void caerIMU9EventSetGyroX (  
    caerIMU9Event event,  
    float gyroX ) [inline], [static]
```

Set the X axis (roll) angular velocity reading (from gyroscope). This is in $^{\circ}/\text{s}$ (deg/sec).

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>gyroX</i>	angular velocity on the X axis (roll).

4.14.4.27 caerIMU9EventSetGyroY()

```
static void caerIMU9EventSetGyroY (  
    caerIMU9Event event,  
    float gyroY ) [inline], [static]
```

Set the Y axis (pitch) angular velocity reading (from gyroscope). This is in $^{\circ}/\text{s}$ (deg/sec).

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>gyroY</i>	angular velocity on the Y axis (pitch).

4.14.4.28 caerIMU9EventSetGyroZ()

```
static void caerIMU9EventSetGyroZ (  
    caerIMU9Event event,  
    float gyroZ ) [inline], [static]
```

Set the Z axis (yaw) angular velocity reading (from gyroscope). This is in $^{\circ}/\text{s}$ (deg/sec).

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>gyroZ</i>	angular velocity on the Z axis (yaw).

4.14.4.29 caerIMU9EventSetTemp()

```
static void caerIMU9EventSetTemp (
    caerIMU9Event event,
    float temp ) [inline], [static]
```

Set the temperature reading. This is in °C.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>temp</i>	temperature in °C.

4.14.4.30 caerIMU9EventSetTimestamp()

```
static void caerIMU9EventSetTimestamp (
    caerIMU9Event event,
    int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.14.4.31 caerIMU9EventValidate()

```
static void caerIMU9EventValidate (
    caerIMU9Event event,
    caerIMU9EventPacket packet ) [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>packet</i>	the IMU9EventPacket pointer for the packet containing this event. Cannot be NULL.

4.14.4.32 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
    struct caer_imu9_event { uint32_t info;int32_t timestamp;float accel_x;float
    accel_y;float accel_z;float gyro_x;float gyro_y;float gyro_z;float temp;float comp_x;float
    comp_y;float comp_z;} )
```

IMU 9-axes event data structure definition. This contains accelerometer and gyroscope headings, plus temperature, and magnetometer readings. The X, Y and Z axes are referred to the camera plane. X increases to the right, Y going up and Z towards where the lens is pointing. Rotation for the gyroscope is counter-clockwise along the increasing axis, for all three axes. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.14.4.33 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
    struct caer_imu9_event_packet { struct caer_event_packet_header packetHeader;struct
    caer_imu9_event events[];} )
```

IMU 9-axes event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.15 events/matrix4x4.h File Reference

```
#include "common.h"
```

Macros

- #define CAER_MATRIX4x4_ITERATOR_ALL_START(MATRIX4x4_PACKET)
 - #define CAER_MATRIX4x4_CONST_ITERATOR_ALL_START(MATRIX4x4_PACKET)
 - #define CAER_MATRIX4x4_ITERATOR_ALL_END }
 - #define CAER_MATRIX4x4_ITERATOR_VALID_START(MATRIX4x4_PACKET)
 - #define CAER_MATRIX4x4_CONST_ITERATOR_VALID_START(MATRIX4x4_PACKET)
 - #define CAER_MATRIX4x4_ITERATOR_VALID_END }
 - #define CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_START(MATRIX4x4_PACKET)
 - #define CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_ALL_START(MATRIX4x4_PACKET)
 - #define CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_END }
 - #define CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_START(MATRIX4x4_PACKET)
 - #define CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_VALID_START(MATRIX4x4_PACKET)
 - #define CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_END }
-
- #define MATRIX4x4_TYPE_SHIFT 1
 - #define MATRIX4x4_TYPE_MASK 0x0000007F
 - #define MATRIX4x4_SCALE_SHIFT 8
 - #define MATRIX4x4_SCALE_MASK 0x000000FF

Typedefs

- typedef struct caer_matrix4x4_event * [caerMatrix4x4Event](#)
- typedef const struct caer_matrix4x4_event * **caerMatrix4x4EventConst**
- typedef struct caer_matrix4x4_event_packet * [caerMatrix4x4EventPacket](#)
- typedef const struct caer_matrix4x4_event_packet * **caerMatrix4x4EventPacketConst**

Functions

- [PACKED_STRUCT](#) (struct caer_matrix4x4_event { uint32_t info; float m[4][4]; int32_t timestamp; })
- [PACKED_STRUCT](#) (struct caer_matrix4x4_event_packet { struct caer_event_packet_header packet; Header; struct caer_matrix4x4_event events[]; })
- static [caerMatrix4x4EventPacket](#) [caerMatrix4x4EventPacketAllocate](#) (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static [caerMatrix4x4EventPacket](#) [caerMatrix4x4EventPacketFromPacketHeader](#) ([caerEventPacketHeader](#) header)
- static [caerMatrix4x4EventPacketConst](#) [caerMatrix4x4EventPacketFromPacketHeaderConst](#) ([caerEventPacketHeaderConst](#) header)
- static [caerMatrix4x4Event](#) [caerMatrix4x4EventPacketGetEvent](#) ([caerMatrix4x4EventPacket](#) packet, int32_t n)
- static [caerMatrix4x4EventConst](#) [caerMatrix4x4EventPacketGetEventConst](#) ([caerMatrix4x4EventPacketConst](#) packet, int32_t n)
- static int32_t [caerMatrix4x4EventGetTimestamp](#) ([caerMatrix4x4EventConst](#) event)
- static int64_t [caerMatrix4x4EventGetTimestamp64](#) ([caerMatrix4x4EventConst](#) event, [caerMatrix4x4EventPacketConst](#) packet)
- static void [caerMatrix4x4EventSetTimestamp](#) ([caerMatrix4x4Event](#) event, int32_t timestamp)
- static bool [caerMatrix4x4EventIsValid](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventValidate](#) ([caerMatrix4x4Event](#) event, [caerMatrix4x4EventPacket](#) packet)
- static void [caerMatrix4x4EventInvalidate](#) ([caerMatrix4x4Event](#) event, [caerMatrix4x4EventPacket](#) packet)
- static uint8_t [caerMatrix4x4EventGetType](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetType](#) ([caerMatrix4x4Event](#) event, uint8_t type)
- static int8_t [caerMatrix4x4EventGetScale](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetScale](#) ([caerMatrix4x4Event](#) event, int8_t scale)
- static float [caerMatrix4x4EventGetM00](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetM00](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM01](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetM01](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM02](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetM02](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM03](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetM03](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM10](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetM10](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM11](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetM11](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM12](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetM12](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM13](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetM13](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM20](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetM20](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM21](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetM21](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM22](#) ([caerMatrix4x4EventConst](#) event)
- static void [caerMatrix4x4EventSetM22](#) ([caerMatrix4x4Event](#) event, float x)

- static float [caerMatrix4x4EventGetM23](#) (caerMatrix4x4EventConst event)
- static void [caerMatrix4x4EventSetM23](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM30](#) (caerMatrix4x4EventConst event)
- static void [caerMatrix4x4EventSetM30](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM31](#) (caerMatrix4x4EventConst event)
- static void [caerMatrix4x4EventSetM31](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM32](#) (caerMatrix4x4EventConst event)
- static void [caerMatrix4x4EventSetM32](#) ([caerMatrix4x4Event](#) event, float x)
- static float [caerMatrix4x4EventGetM33](#) (caerMatrix4x4EventConst event)
- static void [caerMatrix4x4EventSetM33](#) ([caerMatrix4x4Event](#) event, float x)

4.15.1 Detailed Description

THIS EVENT DEFINITION IS STILL TO BE CONSIDERED EXPERIMENTAL AND IS SUBJECT TO FUTURE CHANGES AND REVISIONS!

Matrix4x4 Events format definition and handling functions. This contains a matrix of dimensions 4x4 with floats entries, together with support for distinguishing type and scale. Useful for homogeneous coordinates for example.

m00 m01 m02 m03 m10 m11 m12 m13 m20 m21 m22 m23 m30 m31 m32 m33

4.15.2 Macro Definition Documentation

4.15.2.1 CAER_MATRIX4x4_CONST_ITERATOR_ALL_START

```
#define CAER_MATRIX4x4_CONST_ITERATOR_ALL_START(
    MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter = 0;
    caerMatrix4x4IteratorCounter < caerEventPacketHeaderGetEventNumber
    (&(MATRIX4x4_PACKET)->packetHeader); \
    caerMatrix4x4IteratorCounter++) {
    caerMatrix4x4EventConst caerMatrix4x4IteratorElement
    = caerMatrix4x4EventPacketGetEventConst (MATRIX4x4_PACKET,
    caerMatrix4x4IteratorCounter);
```

Const-Iterator over all Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerMatrix4x4IteratorElement' variable of type caerMatrix4x4EventConst.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.2 CAER_MATRIX4x4_CONST_ITERATOR_VALID_START

```
#define CAER_MATRIX4x4_CONST_ITERATOR_VALID_START(
    MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter = 0;
    caerMatrix4x4IteratorCounter < caerEventPacketHeaderGetEventNumber
    (&(MATRIX4x4_PACKET)->packetHeader); \
    caerMatrix4x4IteratorCounter++) {
    caerMatrix4x4EventConst caerMatrix4x4IteratorElement
    = caerMatrix4x4EventPacketGetEventConst (MATRIX4x4_PACKET,
    caerMatrix4x4IteratorCounter);
    if (!caerMatrix4x4EventIsValid(caerMatrix4x4IteratorElement)) {
        continue;
    }
```

Const-Iterator over only the valid Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerMatrix4x4IteratorElement' variable of type caerMatrix4x4EventConst.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.3 CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_ALL_START(
    MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter
    = caerEventPacketHeaderGetEventNumber (&(MATRIX4x4_PACKET)->
    packetHeader) - 1; \
    caerMatrix4x4IteratorCounter >= 0; caerMatrix4x4IteratorCounter--) {
    caerMatrix4x4EventConst caerMatrix4x4IteratorElement
    = caerMatrix4x4EventPacketGetEventConst (MATRIX4x4_PACKET,
    caerMatrix4x4IteratorCounter);
```

Const-Reverse iterator over all Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerMatrix4x4IteratorElement' variable of type caerMatrix4x4EventConst.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.4 CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_VALID_START(
    MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter
    = caerEventPacketHeaderGetEventNumber (&(MATRIX4x4_PACKET)->
    packetHeader) - 1;
    caerMatrix4x4IteratorCounter >= 0; caerMatrix4x4IteratorCounter--) {
    caerMatrix4x4EventConst caerMatrix4x4IteratorElement
        = caerMatrix4x4EventPacketGetEventConst (MATRIX4x4_PACKET,
    caerMatrix4x4IteratorCounter);
    if (!caerMatrix4x4EventIsValid(caerMatrix4x4IteratorElement)) {
        continue;
    }
```

Const-Reverse iterator over only the valid Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerMatrix4x4IteratorElement' variable of type caerMatrix4x4EventConst.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.5 CAER_MATRIX4x4_ITERATOR_ALL_END

```
#define CAER_MATRIX4x4_ITERATOR_ALL_END }
```

Iterator close statement.

4.15.2.6 CAER_MATRIX4x4_ITERATOR_ALL_START

```
#define CAER_MATRIX4x4_ITERATOR_ALL_START(
    MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter = 0;
    caerMatrix4x4IteratorCounter < caerEventPacketHeaderGetEventNumber
    (&(MATRIX4x4_PACKET)->packetHeader);
    caerMatrix4x4IteratorCounter++) {
    caerMatrix4x4Event caerMatrix4x4IteratorElement
        = caerMatrix4x4EventPacketGetEvent (MATRIX4x4_PACKET,
    caerMatrix4x4IteratorCounter);
```

Iterator over all Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4IteratorCounter' variable of type 'int32_t' and the current event in the 'caerMatrix4x4IteratorElement' variable of type caerMatrix4x4Event.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.7 CAER_MATRIX4x4_ITERATOR_VALID_END

```
#define CAER_MATRIX4x4_ITERATOR_VALID_END }
```

Iterator close statement.

4.15.2.8 CAER_MATRIX4x4_ITERATOR_VALID_START

```
#define CAER_MATRIX4x4_ITERATOR_VALID_START(
    MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter = 0;
    \ caerMatrix4x4IteratorCounter < caerEventPacketHeaderGetEventNumber
    (& (MATRIX4x4_PACKET)->packetHeader); \
    caerMatrix4x4IteratorCounter++) {
    \ caerMatrix4x4Event caerMatrix4x4IteratorElement
    \ = caerMatrix4x4EventPacketGetEvent (MATRIX4x4_PACKET,
    caerMatrix4x4IteratorCounter); \
    if (!caerMatrix4x4EventIsValid(caerMatrix4x4IteratorElement)) {
        \ continue;
    \ }
}
```

Iterator over only the valid Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4IteratorCounter' variable of type 'int32_t' and the current event in the 'caerMatrix4x4IteratorElement' variable of type caerMatrix4x4Event.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.9 CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_END

```
#define CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.15.2.10 CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_START

```
#define CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_START(
    MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter
    = caerEventPacketHeaderGetEventNumber (& (MATRIX4x4_PACKET)->
    packetHeader) - 1; \
    caerMatrix4x4IteratorCounter >= 0; caerMatrix4x4IteratorCounter--) {
    \ caerMatrix4x4Event caerMatrix4x4IteratorElement
    \ = caerMatrix4x4EventPacketGetEvent (MATRIX4x4_PACKET,
    caerMatrix4x4IteratorCounter);
}
```

Reverse iterator over all Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4IteratorCounter' variable of type 'int32_t' and the current event in the 'caerMatrix4x4IteratorElement' variable of type caerMatrix4x4Event.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.11 CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_END

```
#define CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.15.2.12 CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_START

```
#define CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_START (
    MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter                \
    = caerEventPacketHeaderGetEventNumber (& (MATRIX4x4_PACKET) -> \
    packetHeader) - 1; \
    caerMatrix4x4IteratorCounter >= 0; caerMatrix4x4IteratorCounter--) { \
    caerMatrix4x4Event caerMatrix4x4IteratorElement \
    = caerMatrix4x4EventPacketGetEvent (MATRIX4x4_PACKET, \
    caerMatrix4x4IteratorCounter); \
    if (!caerMatrix4x4EventIsValid (caerMatrix4x4IteratorElement)) { \
        continue; \
    } }
```

Reverse iterator over only the valid Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4IteratorCounter' variable of type 'int32_t' and the current event in the 'caerMatrix4x4IteratorElement' variable of type caerMatrix4x4Event.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.13 MATRIX4x4_SCALE_MASK

```
#define MATRIX4x4_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Matrix4x4 event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.15.2.14 MATRIX4x4_SCALE_SHIFT

```
#define MATRIX4x4_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Matrix4x4 event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.15.2.15 MATRIX4x4_TYPE_MASK

```
#define MATRIX4x4_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Matrix4x4 event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.15.2.16 MATRIX4x4_TYPE_SHIFT

```
#define MATRIX4x4_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Matrix4x4 event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.15.3 Typedef Documentation

4.15.3.1 caerMatrix4x4Event

```
typedef struct caer_matrix4x4_event* caerMatrix4x4Event
```

Type for pointer to Matrix4x4 event data structure.

4.15.3.2 caerMatrix4x4EventPacket

```
typedef struct caer_matrix4x4_event_packet* caerMatrix4x4EventPacket
```

Type for pointer to Matrix4x4 event packet data structure.

4.15.4 Function Documentation

4.15.4.1 caerMatrix4x4EventGetM00()

```
static float caerMatrix4x4EventGetM00 (
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M00 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M00 element.

4.15.4.2 caerMatrix4x4EventGetM01()

```
static float caerMatrix4x4EventGetM01 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M01 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M01 element.

4.15.4.3 caerMatrix4x4EventGetM02()

```
static float caerMatrix4x4EventGetM02 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M02 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M02 element.

4.15.4.4 caerMatrix4x4EventGetM03()

```
static float caerMatrix4x4EventGetM03 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M03 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M01 element.

4.15.4.5 caerMatrix4x4EventGetM10()

```
static float caerMatrix4x4EventGetM10 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M10 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M10 element.

4.15.4.6 caerMatrix4x4EventGetM11()

```
static float caerMatrix4x4EventGetM11 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M11 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M11 element.

4.15.4.7 caerMatrix4x4EventGetM12()

```
static float caerMatrix4x4EventGetM12 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M12 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M12 element.

4.15.4.8 caerMatrix4x4EventGetM13()

```
static float caerMatrix4x4EventGetM13 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M13 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M13 element.

4.15.4.9 caerMatrix4x4EventGetM20()

```
static float caerMatrix4x4EventGetM20 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M20 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M20 element.

4.15.4.10 caerMatrix4x4EventGetM21()

```
static float caerMatrix4x4EventGetM21 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M21 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M21 element.

4.15.4.11 caerMatrix4x4EventGetM22()

```
static float caerMatrix4x4EventGetM22 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M22 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M22 element.

4.15.4.12 caerMatrix4x4EventGetM23()

```
static float caerMatrix4x4EventGetM23 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M23 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M23 element.

4.15.4.13 caerMatrix4x4EventGetM30()

```
static float caerMatrix4x4EventGetM30 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M30 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M30 element.

4.15.4.14 caerMatrix4x4EventGetM31()

```
static float caerMatrix4x4EventGetM31 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M31 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M31 element.

4.15.4.15 caerMatrix4x4EventGetM32()

```
static float caerMatrix4x4EventGetM32 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M32 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M32 element.

4.15.4.16 caerMatrix4x4EventGetM33()

```
static float caerMatrix4x4EventGetM33 (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the M33 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

M33 element.

4.15.4.17 caerMatrix4x4EventGetScale()

```
static int8_t caerMatrix4x4EventGetScale (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^{-2}) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

the Matrix4x4 measurement scale.

4.15.4.18 caerMatrix4x4EventGetTimestamp()

```
static int32_t caerMatrix4x4EventGetTimestamp (  
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond timestamp.

4.15.4.19 caerMatrix4x4EventGetTimestamp64()

```
static int64_t caerMatrix4x4EventGetTimestamp64 (
    caerMatrix4x4EventConst event,
    caerMatrix4x4EventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>packet</i>	the Matrix4x4EventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.15.4.20 caerMatrix4x4EventGetType()

```
static uint8_t caerMatrix4x4EventGetType (
    caerMatrix4x4EventConst event ) [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

the Matrix4x4 measurement type.

4.15.4.21 caerMatrix4x4EventInvalidate()

```
static void caerMatrix4x4EventInvalidate (
    caerMatrix4x4Event event,
    caerMatrix4x4EventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>packet</i>	the Matrix4x4EventPacket pointer for the packet containing this event. Cannot be NULL.

4.15.4.22 caerMatrix4x4EventIsValid()

```
static bool caerMatrix4x4EventIsValid (
    caerMatrix4x4EventConst event ) [inline], [static]
```

Check if this Matrix4x4 event is valid.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
--------------	---

Returns

true if valid, false if not.

4.15.4.23 caerMatrix4x4EventPacketAllocate()

```
static caerMatrix4x4EventPacket caerMatrix4x4EventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]
```

Allocate a new Matrix4x4 events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid Matrix4x4EventPacket handle or NULL on error.

4.15.4.24 caerMatrix4x4EventPacketFromPacketHeader()

```
static caerMatrix4x4EventPacket caerMatrix4x4EventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```


Transform a generic event packet header into a Matrix4x4 event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.15.4.25 caerMatrix4x4EventPacketFromPacketHeaderConst()

```
static caerMatrix4x4EventPacketConst caerMatrix4x4EventPacketFromPacketHeaderConst (  
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only Matrix4x4 event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.15.4.26 caerMatrix4x4EventPacketGetEvent()

```
static caerMatrix4x4Event caerMatrix4x4EventPacketGetEvent (  
    caerMatrix4x4EventPacket packet,  
    int32_t n ) [inline], [static]
```

Get the Matrix4x4 event at the given index from the event packet.

Parameters

<i>packet</i>	a valid Matrix4x4EventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Matrix4x4 event. NULL on error.

4.15.4.27 caerMatrix4x4EventPacketGetEventConst()

```
static caerMatrix4x4EventConst caerMatrix4x4EventPacketGetEventConst (
    caerMatrix4x4EventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the Matrix4x4 event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid Matrix4x4EventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only Matrix4x4 event. NULL on error.

4.15.4.28 caerMatrix4x4EventSetM00()

```
static void caerMatrix4x4EventSetM00 (
    caerMatrix4x4Event event,
    float x ) [inline], [static]
```

Set the M00 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	m00 value.

4.15.4.29 caerMatrix4x4EventSetM01()

```
static void caerMatrix4x4EventSetM01 (
    caerMatrix4x4Event event,
    float x ) [inline], [static]
```

Set the M01 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	m01 value.

4.15.4.30 caerMatrix4x4EventSetM02()

```
static void caerMatrix4x4EventSetM02 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M02 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	m02 value.

4.15.4.31 caerMatrix4x4EventSetM03()

```
static void caerMatrix4x4EventSetM03 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M03 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	m03 value.

4.15.4.32 caerMatrix4x4EventSetM10()

```
static void caerMatrix4x4EventSetM10 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M10 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	m10 value.

4.15.4.33 caerMatrix4x4EventSetM11()

```
static void caerMatrix4x4EventSetM11 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M11 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	m11 value.

4.15.4.34 caerMatrix4x4EventSetM12()

```
static void caerMatrix4x4EventSetM12 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M12 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	m12 value.

4.15.4.35 caerMatrix4x4EventSetM13()

```
static void caerMatrix4x4EventSetM13 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M13 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	m13 value.

4.15.4.36 caerMatrix4x4EventSetM20()

```
static void caerMatrix4x4EventSetM20 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M20 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	M20 value.

4.15.4.37 caerMatrix4x4EventSetM21()

```
static void caerMatrix4x4EventSetM21 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M21 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	M21 value.

4.15.4.38 caerMatrix4x4EventSetM22()

```
static void caerMatrix4x4EventSetM22 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M22 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	M22 value.

4.15.4.39 caerMatrix4x4EventSetM23()

```
static void caerMatrix4x4EventSetM23 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M23 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	M23 value.

4.15.4.40 caerMatrix4x4EventSetM30()

```
static void caerMatrix4x4EventSetM30 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M30 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	M30 value.

4.15.4.41 caerMatrix4x4EventSetM31()

```
static void caerMatrix4x4EventSetM31 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M31 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	M31 value.

4.15.4.42 caerMatrix4x4EventSetM32()

```
static void caerMatrix4x4EventSetM32 (  
    caerMatrix4x4Event event,  
    float x ) [inline], [static]
```

Set the M32 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	M32 value.

4.15.4.43 caerMatrix4x4EventSetM33()

```
static void caerMatrix4x4EventSetM33 (
    caerMatrix4x4Event event,
    float x ) [inline], [static]
```

Set the M33 element.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>x</i>	M33 value.

4.15.4.44 caerMatrix4x4EventSetScale()

```
static void caerMatrix4x4EventSetScale (
    caerMatrix4x4Event event,
    int8_t scale ) [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^{-2}) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>scale</i>	the Matrix4x4 measurement scale.

4.15.4.45 caerMatrix4x4EventSetTimestamp()

```
static void caerMatrix4x4EventSetTimestamp (
    caerMatrix4x4Event event,
    int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.15.4.46 caerMatrix4x4EventSetType()

```
static void caerMatrix4x4EventSetType (
    caerMatrix4x4Event event,
    uint8_t type ) [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>type</i>	the Matrix4x4 measurement type.

4.15.4.47 caerMatrix4x4EventValidate()

```
static void caerMatrix4x4EventValidate (
    caerMatrix4x4Event event,
    caerMatrix4x4EventPacket packet ) [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid Matrix4x4Event pointer. Cannot be NULL.
<i>packet</i>	the Matrix4x4EventPacket pointer for the packet containing this event. Cannot be NULL.

4.15.4.48 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
    struct caer_matrix4x4_event { uint32_t info;float m[4][4];int32_t timestamp;} )
```

Matrix4x4 event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The measurements are stored as floats. m00 m01 m02 m03 m10 m11 m12 m13 m20 m21 m22 m23 m30 m31 m32 m33 Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.15.4.49 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
    struct caer_matrix4x4_event_packet { struct caer_event_packet_header packet↔
Header;struct caer_matrix4x4_event events[];} )
```

Matrix4x4 event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.16 events/packetContainer.h File Reference

```
#include "common.h"
```

Macros

- `#define CAER_EVENT_PACKET_CONTAINER_ITERATOR_START(PACKET_CONTAINER)`
- `#define CAER_EVENT_PACKET_CONTAINER_CONST_ITERATOR_START(PACKET_CONTAINER)`
- `#define CAER_EVENT_PACKET_CONTAINER_ITERATOR_END`

Typedefs

- `typedef struct caer_event_packet_container * caerEventPacketContainer`
- `typedef const struct caer_event_packet_container * caerEventPacketContainerConst`

Functions

- `PACKED_STRUCT` (struct caer_event_packet_container { int64_t lowestEventTimestamp;int64_t highestEventTimestamp;int32_t eventsNumber;int32_t eventsValidNumber;int32_t eventPacketsNumber;caerEventPacketHeader eventPackets[];})
- static `caerEventPacketContainer caerEventPacketContainerAllocate` (int32_t eventPacketsNumber)
- static void `caerEventPacketContainerUpdateStatistics` (caerEventPacketContainer container)
- static int32_t `caerEventPacketContainerGetEventPacketsNumber` (caerEventPacketContainerConst container)
- static void `caerEventPacketContainerSetEventPacketsNumber` (caerEventPacketContainer container, int32_t eventPacketsNumber)
- static `caerEventPacketHeader caerEventPacketContainerGetEventPacket` (caerEventPacketContainerConst container, int32_t n)
- static `caerEventPacketHeaderConst caerEventPacketContainerGetEventPacketConst` (caerEventPacketContainerConst container, int32_t n)
- static void `caerEventPacketContainerSetEventPacket` (caerEventPacketContainer container, int32_t n, caerEventPacketHeader packetHeader)
- static void `caerEventPacketContainerFree` (caerEventPacketContainer container)
- static int64_t `caerEventPacketContainerGetLowestEventTimestamp` (caerEventPacketContainerConst container)
- static int64_t `caerEventPacketContainerGetHighestEventTimestamp` (caerEventPacketContainerConst container)
- static int32_t `caerEventPacketContainerGetEventsNumber` (caerEventPacketContainerConst container)
- static int32_t `caerEventPacketContainerGetEventsValidNumber` (caerEventPacketContainerConst container)
- static `caerEventPacketHeader caerEventPacketContainerFindEventPacketByType` (caerEventPacketContainerConst container, int16_t typeId)
- static `caerEventPacketHeaderConst caerEventPacketContainerFindEventPacketByTypeConst` (caerEventPacketContainerConst container, int16_t typeId)
- static `caerEventPacketContainer caerEventPacketContainerCopyAllEvents` (caerEventPacketContainerConst container)
- static `caerEventPacketContainer caerEventPacketContainerCopyValidEvents` (caerEventPacketContainerConst container)

4.16.1 Detailed Description

EventPacketContainer format definition and handling functions. An EventPacketContainer is a logical construct that contains packets of events (EventPackets) of different event types, with the aim of keeping related events of differing types, such as DVS and IMU data, together. Such a relation is usually based on time intervals, trying to keep groups of event happening in a certain time-slice together. This time-order is based on the *main* time-stamp of an event, the one whose offset is referenced in the event packet header and that is used by the caerGenericEvent*() functions. It's guaranteed that all conforming input modules keep to this rule, generating containers that include all events from all types within the given time-slice. The smallest and largest timestamps are tracked at the packet container level as a convenience, to avoid having to examine all packets for this often useful piece of information. All integers are in their native host format, as this is a purely internal, in-memory data structure, never meant for exchange between different systems (and different endianness).

== Packet Containers and Input Modules == The "packeting system" works in this way: events are accumulated by type in a packet, and that packet is part of a packet container, by an input module. The packet container is then sent out for processing when either the configured time limit or the size limit are hit. The time limit is always active, in microseconds, and basically tells you the time-span an event packet covers. This enables regular, constant delivery of packets, that cover a period of time. The size limit is an add-on to prevent packets to grow to immense sizes (like if the time limit is high and there is lots of activity). As soon as a packet hits the number of events in the size limit, it is sent out. The regular time limit is not reset in this case. This size limit can be disabled by setting it to 0. The cAER DVS128/DAVIS/File/Network input modules call these two configuration variables "PacketContainerInterval" and "PacketContainerMaxPacketSize". Too small packet sizes or intervals simply mean more packets, which may negatively affect performance. It's usually a good idea to set the size to something around 4-8K, and the time to a good value based on the application you're building, so if you need ms-reaction-time, you probably want to set it to 1000µs, so that you do get new data every ms. If on the other hand you're looking at a static scene and just want to detect that something is passing by once every while, a higher number like 100ms might also be perfectly appropriate.

4.16.2 Macro Definition Documentation

4.16.2.1 CAER_EVENT_PACKET_CONTAINER_CONST_ITERATOR_START

```
#define CAER_EVENT_PACKET_CONTAINER_CONST_ITERATOR_START (
    PACKET_CONTAINER )
```

Value:

```
if ((PACKET_CONTAINER) != NULL) {
    for (int32_t caerEventPacketContainerIteratorCounter = 0;
        caerEventPacketContainerIteratorCounter
        < caerEventPacketContainerGetEventPacketsNumber (
            PACKET_CONTAINER); \
        caerEventPacketContainerIteratorCounter++) {
        caerEventPacketHeaderConst caerEventPacketContainerIteratorElement \
        = caerEventPacketContainerGetEventPacketConst (
            \
            PACKET_CONTAINER, caerEventPacketContainerIteratorCounter); \
        if (caerEventPacketContainerIteratorElement == NULL) {
            continue;
        }
    }
```

Const-Iterator over all event packets in an event packet container. Returns the current index in the 'caerEvent↵PacketContainerIteratorCounter' variable of type 'int32_t' and the current read-only event packet in the 'caerEvent↵PacketContainerIteratorElement' variable of type caerEventPacketHeaderConst. The current packet may be NULL, in which case it is skipped during iteration.

PACKET_CONTAINER: a valid EventPacketContainer handle. If NULL, no iteration is performed.

4.16.2.2 CAER_EVENT_PACKET_CONTAINER_ITERATOR_END

```
#define CAER_EVENT_PACKET_CONTAINER_ITERATOR_END
```

Value:

```
}
    }
```

Iterator close statement.

4.16.2.3 CAER_EVENT_PACKET_CONTAINER_ITERATOR_START

```
#define CAER_EVENT_PACKET_CONTAINER_ITERATOR_START(
    PACKET_CONTAINER )
```

Value:

```
if ((PACKET_CONTAINER) != NULL) {
    for (int32_t caerEventPacketContainerIteratorCounter = 0;
        caerEventPacketContainerIteratorCounter
        < caerEventPacketContainerGetEventPacketsNumber(
            PACKET_CONTAINER);
        caerEventPacketContainerIteratorCounter++) {
        caerEventPacketHeader caerEventPacketContainerIteratorElement
            = caerEventPacketContainerGetEventPacket(
                PACKET_CONTAINER, caerEventPacketContainerIteratorCounter);
        if (caerEventPacketContainerIteratorElement == NULL) {
            continue;
        }
    }
}
```

Iterator over all event packets in an event packet container. Returns the current index in the 'caerEventPacketContainerIteratorCounter' variable of type 'int32_t' and the current event packet in the 'caerEventPacketContainerIteratorElement' variable of type caerEventPacketHeader. The current packet may be NULL, in which case it is skipped during iteration.

PACKET_CONTAINER: a valid EventPacketContainer handle. If NULL, no iteration is performed.

4.16.3 Typedef Documentation

4.16.3.1 caerEventPacketContainer

```
typedef struct caer_event_packet_container* caerEventPacketContainer
```

Type for pointer to EventPacketContainer data structure.

4.16.4 Function Documentation

4.16.4.1 caerEventPacketContainerAllocate()

```
static caerEventPacketContainer caerEventPacketContainerAllocate (
    int32_t eventPacketsNumber ) [inline], [static]
```

Allocate a new EventPacketContainer with enough space to store up to the given number of EventPacket pointers. All packet pointers will be NULL initially.

Parameters

<i>eventPacketsNumber</i>	the maximum number of EventPacket pointers that can be stored in this container.
---------------------------	--

Returns

a valid EventPacketContainer handle or NULL on error.

4.16.4.2 caerEventPacketContainerCopyAllEvents()

```
static caerEventPacketContainer caerEventPacketContainerCopyAllEvents (  
    caerEventPacketContainerConst container ) [inline], [static]
```

Make a deep copy of an event packet container and all of its event packets and their current events.

Parameters

<i>container</i>	an event packet container to copy.
------------------	------------------------------------

Returns

a deep copy of an event packet container, containing all events.

4.16.4.3 caerEventPacketContainerCopyValidEvents()

```
static caerEventPacketContainer caerEventPacketContainerCopyValidEvents (  
    caerEventPacketContainerConst container ) [inline], [static]
```

Make a deep copy of an event packet container, with its event packets sized down to only include the currently valid events (eventValid), and discarding everything else.

Parameters

<i>container</i>	an event packet container to copy.
------------------	------------------------------------

Returns

a deep copy of an event packet container, containing only valid events.

4.16.4.4 caerEventPacketContainerFindEventPacketByType()

```
static caerEventPacketHeader caerEventPacketContainerFindEventPacketByType (
    caerEventPacketContainerConst container,
    int16_t typeID ) [inline], [static]
```

Get the pointer to an EventPacket stored in this container with the given event type. This returns the first found event packet with that type ID, or NULL if we get to the end without finding any such event packet.

Parameters

<i>container</i>	a valid EventPacketContainer handle. If NULL, returns NULL too.
<i>typeID</i>	the event type to search for.

Returns

a pointer to an EventPacket with a certain type or NULL if none found.

4.16.4.5 caerEventPacketContainerFindEventPacketByTypeConst()

```
static caerEventPacketHeaderConst caerEventPacketContainerFindEventPacketByTypeConst (
    caerEventPacketContainerConst container,
    int16_t typeID ) [inline], [static]
```

Get the pointer to a read-only EventPacket stored in this container with the given event type. This returns the first found event packet with that type ID, or NULL if we get to the end without finding any such event packet.

Parameters

<i>container</i>	a valid EventPacketContainer handle. If NULL, returns NULL too.
<i>typeID</i>	the event type to search for.

Returns

a pointer to a read-only EventPacket with a certain type or NULL if none found.

4.16.4.6 caerEventPacketContainerFree()

```
static void caerEventPacketContainerFree (
    caerEventPacketContainer container ) [inline], [static]
```

Free the memory occupied by an EventPacketContainer, as well as freeing all of its contained EventPackets and their memory. If you don't want the contained EventPackets to be freed, make sure that you set their pointers to NULL before calling this.

Parameters

<i>container</i>	the container to be freed.
------------------	----------------------------

4.16.4.7 caerEventPacketContainerGetEventPacket()

```
static caerEventPacketHeader caerEventPacketContainerGetEventPacket (
    caerEventPacketContainerConst container,
    int32_t n ) [inline], [static]
```

Get the pointer to the EventPacket stored in this container at the given index.

Parameters

<i>container</i>	a valid EventPacketContainer handle. If NULL, returns NULL too.
<i>n</i>	the index of the EventPacket to get.

Returns

a pointer to an EventPacket or NULL on error.

4.16.4.8 caerEventPacketContainerGetEventPacketConst()

```
static caerEventPacketHeaderConst caerEventPacketContainerGetEventPacketConst (
    caerEventPacketContainerConst container,
    int32_t n ) [inline], [static]
```

Get the pointer to the EventPacket stored in this container at the given index. This is a read-only EventPacket, do not change its contents in any way!

Parameters

<i>container</i>	a valid EventPacketContainer handle. If NULL, returns NULL too.
<i>n</i>	the index of the EventPacket to get.

Returns

a pointer to a read-only EventPacket or NULL on error.

4.16.4.9 caerEventPacketContainerGetEventPacketsNumber()

```
static int32_t caerEventPacketContainerGetEventPacketsNumber (
    caerEventPacketContainerConst container ) [inline], [static]
```

Get the maximum number of EventPacket pointers that can be stored in this particular EventPacketContainer.

Parameters

<i>container</i>	a valid EventPacketContainer handle. If NULL, zero is returned.
------------------	---

Returns

the number of EventPacket pointers that can be contained.

4.16.4.10 caerEventPacketContainerGetEventsNumber()

```
static int32_t caerEventPacketContainerGetEventsNumber (  
    caerEventPacketContainerConst container ) [inline], [static]
```

Get the number of events contained in this event packet container.

Parameters

<i>container</i>	a valid EventPacketContainer handle. If NULL, 0 is returned.
------------------	--

Returns

the number of events in this container.

4.16.4.11 caerEventPacketContainerGetEventsValidNumber()

```
static int32_t caerEventPacketContainerGetEventsValidNumber (  
    caerEventPacketContainerConst container ) [inline], [static]
```

Get the number of valid events contained in this event packet container.

Parameters

<i>container</i>	a valid EventPacketContainer handle. If NULL, 0 is returned.
------------------	--

Returns

the number of valid events in this container.

4.16.4.12 caerEventPacketContainerGetHighestEventTimestamp()

```
static int64_t caerEventPacketContainerGetHighestEventTimestamp (
    caerEventPacketContainerConst container ) [inline], [static]
```

Get the highest timestamp contained in this event packet container.

Parameters

<i>container</i>	a valid EventPacketContainer handle. If NULL, -1 is returned.
------------------	---

Returns

the highest timestamp (in μ s) or -1 if not initialized.

4.16.4.13 caerEventPacketContainerGetLowestEventTimestamp()

```
static int64_t caerEventPacketContainerGetLowestEventTimestamp (
    caerEventPacketContainerConst container ) [inline], [static]
```

Get the lowest timestamp contained in this event packet container.

Parameters

<i>container</i>	a valid EventPacketContainer handle. If NULL, -1 is returned.
------------------	---

Returns

the lowest timestamp (in μ s) or -1 if not initialized.

4.16.4.14 caerEventPacketContainerSetEventPacket()

```
static void caerEventPacketContainerSetEventPacket (
    caerEventPacketContainer container,
    int32_t n,
    caerEventPacketHeader packetHeader ) [inline], [static]
```

Set the pointer to the EventPacket stored in this container at the given index.

Parameters

<i>container</i>	a valid EventPacketContainer handle. If NULL, nothing happens.
<i>n</i>	the index of the EventPacket to set.
<i>packetHeader</i>	a pointer to an EventPacket's header. Can be NULL.

4.16.4.15 caerEventPacketContainerSetEventPacketsNumber()

```
static void caerEventPacketContainerSetEventPacketsNumber (
    caerEventPacketContainer container,
    int32_t eventPacketsNumber ) [inline], [static]
```

Set the maximum number of EventPacket pointers that can be stored in this particular EventPacketContainer. This should never be used directly, [caerEventPacketContainerAllocate\(\)](#) sets this for you.

Parameters

<i>container</i>	a valid EventPacketContainer handle. If NULL, nothing happens.
<i>eventPacketsNumber</i>	the number of EventPacket pointers that can be contained.

4.16.4.16 caerEventPacketContainerUpdateStatistics()

```
static void caerEventPacketContainerUpdateStatistics (
    caerEventPacketContainer container ) [inline], [static]
```

Recalculates and updates all the packet-container level statistics (event counts and timestamps).

Parameters

<i>container</i>	a valid EventPacketContainer handle. If NULL, nothing happens.
------------------	--

4.16.4.17 PACKED_STRUCT()

```
PACKED_STRUCT (
    struct caer_event_packet_container { int64_t lowestEventTimestamp;int64_t highest↔
EventTimestamp;int32_t eventsNumber;int32_t eventsValidNumber;int32_t eventPacketsNumber;caer↔
EventPacketHeader eventPackets[]; } )
```

EventPacketContainer data structure definition. Signed integers are used for compatibility with languages that do not have unsigned ones, such as Java.

4.17 events/point1d.h File Reference

```
#include "common.h"
```

Macros

- `#define CAER_POINT1D_ITERATOR_ALL_START`(POINT1D_PACKET)
 - `#define CAER_POINT1D_CONST_ITERATOR_ALL_START`(POINT1D_PACKET)
 - `#define CAER_POINT1D_ITERATOR_ALL_END` }
 - `#define CAER_POINT1D_ITERATOR_VALID_START`(POINT1D_PACKET)
 - `#define CAER_POINT1D_CONST_ITERATOR_VALID_START`(POINT1D_PACKET)
 - `#define CAER_POINT1D_ITERATOR_VALID_END` }
 - `#define CAER_POINT1D_REVERSE_ITERATOR_ALL_START`(POINT1D_PACKET)
 - `#define CAER_POINT1D_CONST_REVERSE_ITERATOR_ALL_START`(POINT1D_PACKET)
 - `#define CAER_POINT1D_REVERSE_ITERATOR_ALL_END` }
 - `#define CAER_POINT1D_REVERSE_ITERATOR_VALID_START`(POINT1D_PACKET)
 - `#define CAER_POINT1D_CONST_REVERSE_ITERATOR_VALID_START`(POINT1D_PACKET)
 - `#define CAER_POINT1D_REVERSE_ITERATOR_VALID_END` }
-
- `#define POINT1D_TYPE_SHIFT` 1
 - `#define POINT1D_TYPE_MASK` 0x0000007F
 - `#define POINT1D_SCALE_SHIFT` 8
 - `#define POINT1D_SCALE_MASK` 0x000000FF

Typedefs

- `typedef struct caer_point1d_event * caerPoint1DEvent`
- `typedef const struct caer_point1d_event * caerPoint1DEventConst`
- `typedef struct caer_point1d_event_packet * caerPoint1DEventPacket`
- `typedef const struct caer_point1d_event_packet * caerPoint1DEventPacketConst`

Functions

- `PACKED_STRUCT` (struct caer_point1d_event { uint32_t info; float x; int32_t timestamp; })
- `PACKED_STRUCT` (struct caer_point1d_event_packet { struct caer_event_packet_header packet; Header; struct caer_point1d_event events[]; })
- static `caerPoint1DEventPacket caerPoint1DEventPacketAllocate` (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static `caerPoint1DEventPacket caerPoint1DEventPacketFromPacketHeader` (caerEventPacketHeader header)
- static `caerPoint1DEventPacketConst caerPoint1DEventPacketFromPacketHeaderConst` (caerEventPacketHeaderConst header)
- static `caerPoint1DEvent caerPoint1DEventPacketGetEvent` (caerPoint1DEventPacket packet, int32_t n)
- static `caerPoint1DEventConst caerPoint1DEventPacketGetEventConst` (caerPoint1DEventPacketConst packet, int32_t n)
- static `int32_t caerPoint1DEventGetTimestamp` (caerPoint1DEventConst event)
- static `int64_t caerPoint1DEventGetTimestamp64` (caerPoint1DEventConst event, caerPoint1DEventPacketConst packet)
- static void `caerPoint1DEventSetTimestamp` (caerPoint1DEvent event, int32_t timestamp)
- static bool `caerPoint1DEventIsValid` (caerPoint1DEventConst event)
- static void `caerPoint1DEventValidate` (caerPoint1DEvent event, caerPoint1DEventPacket packet)
- static void `caerPoint1DEventInvalidate` (caerPoint1DEvent event, caerPoint1DEventPacket packet)
- static `uint8_t caerPoint1DEventGetType` (caerPoint1DEventConst event)
- static void `caerPoint1DEventSetType` (caerPoint1DEvent event, uint8_t type)
- static `int8_t caerPoint1DEventGetScale` (caerPoint1DEventConst event)
- static void `caerPoint1DEventSetScale` (caerPoint1DEvent event, int8_t scale)
- static float `caerPoint1DEventGetX` (caerPoint1DEventConst event)
- static void `caerPoint1DEventSetX` (caerPoint1DEvent event, float x)

4.17.1 Detailed Description

Point1D Events format definition and handling functions. This contains one dimensional data points as floats, together with support for distinguishing type and scale.

4.17.2 Macro Definition Documentation

4.17.2.1 CAER_POINT1D_CONST_ITERATOR_ALL_START

```
#define CAER_POINT1D_CONST_ITERATOR_ALL_START(  
    POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter = 0; \
     caerPoint1DIteratorCounter < caerEventPacketHeaderGetEventNumber  
    (&(POINT1D_PACKET)->packetHeader); \
     caerPoint1DIteratorCounter++) {  
    \
    caerPoint1DEventConst caerPoint1DIteratorElement  
    \
    = caerPoint1DEventPacketGetEventConst (POINT1D_PACKET,  
    caerPoint1DIteratorCounter);
```

Const-Iterator over all Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint1DIteratorElement' variable of type caer↵Point1DEventConst.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.2 CAER_POINT1D_CONST_ITERATOR_VALID_START

```
#define CAER_POINT1D_CONST_ITERATOR_VALID_START(  
    POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter = 0; \
     caerPoint1DIteratorCounter < caerEventPacketHeaderGetEventNumber  
    (&(POINT1D_PACKET)->packetHeader); \
     caerPoint1DIteratorCounter++) {  
    \
    caerPoint1DEventConst caerPoint1DIteratorElement  
    \
    = caerPoint1DEventPacketGetEventConst (POINT1D_PACKET,  
    caerPoint1DIteratorCounter);  
    if (!caerPoint1DEventIsValid(caerPoint1DIteratorElement)) {  
        \
        continue;  
    }  
    }
```

Const-Iterator over only the valid Point1D events in a packet. Returns the current index in the 'caerPoint1DIterator↵Counter' variable of type 'int32_t' and the current read-only event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEventConst.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.3 CAER_POINT1D_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT1D_CONST_REVERSE_ITERATOR_ALL_START(  
    POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter           \  
    = caerEventPacketHeaderGetEventNumber (& (POINT1D_PACKET) -> \  
    packetHeader) - 1; \  
    caerPoint1DIteratorCounter >= 0; caerPoint1DIteratorCounter--) {    \  
    caerPoint1DEventConst caerPoint1DIteratorElement    \  
        = caerPoint1DEventPacketGetEventConst (POINT1D_PACKET, \  
    caerPoint1DIteratorCounter);
```

Const-Reverse iterator over all Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEventConst.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.4 CAER_POINT1D_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT1D_CONST_REVERSE_ITERATOR_VALID_START(  
    POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter           \  
    = caerEventPacketHeaderGetEventNumber (& (POINT1D_PACKET) -> \  
    packetHeader) - 1; \  
    caerPoint1DIteratorCounter >= 0; caerPoint1DIteratorCounter--) {    \  
    caerPoint1DEventConst caerPoint1DIteratorElement    \  
        = caerPoint1DEventPacketGetEventConst (POINT1D_PACKET, \  
    caerPoint1DIteratorCounter); \  
    if (!caerPoint1DEventIsValid(caerPoint1DIteratorElement)) {    \  
        continue;    \  
    }
```

Const-Reverse iterator over only the valid Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEventConst.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.5 CAER_POINT1D_ITERATOR_ALL_END

```
#define CAER_POINT1D_ITERATOR_ALL_END }
```

Iterator close statement.

4.17.2.6 CAER_POINT1D_ITERATOR_ALL_START

```
#define CAER_POINT1D_ITERATOR_ALL_START(  
    POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter = 0; \
     caerPoint1DIteratorCounter < caerEventPacketHeaderGetEventNumber  
(&(POINT1D_PACKET)->packetHeader); \
     caerPoint1DIteratorCounter++) { \
    caerPoint1DEvent caerPoint1DIteratorElement \
    = caerPoint1DEventPacketGetEvent(POINT1D_PACKET,  
    caerPoint1DIteratorCounter);
```

Iterator over all Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEvent.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.7 CAER_POINT1D_ITERATOR_VALID_END

```
#define CAER_POINT1D_ITERATOR_VALID_END }
```

Iterator close statement.

4.17.2.8 CAER_POINT1D_ITERATOR_VALID_START

```
#define CAER_POINT1D_ITERATOR_VALID_START(  
    POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter = 0; \
     caerPoint1DIteratorCounter < caerEventPacketHeaderGetEventNumber  
(&(POINT1D_PACKET)->packetHeader); \
     caerPoint1DIteratorCounter++) { \
    caerPoint1DEvent caerPoint1DIteratorElement \
    = caerPoint1DEventPacketGetEvent(POINT1D_PACKET,  
    caerPoint1DIteratorCounter); \
    if (!caerPoint1DEventIsValid(caerPoint1DIteratorElement)) { \
        continue; \
    }
```

Iterator over only the valid Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEvent.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.9 CAER_POINT1D_REVERSE_ITERATOR_ALL_END

```
#define CAER_POINT1D_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.17.2.10 CAER_POINT1D_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT1D_REVERSE_ITERATOR_ALL_START(
    POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter
    = caerEventPacketHeaderGetEventNumber (& (POINT1D_PACKET) ->
    packetHeader) - 1; \
    caerPoint1DIteratorCounter >= 0; caerPoint1DIteratorCounter--) {
    caerPoint1DEvent caerPoint1DIteratorElement
    = caerPoint1DEventPacketGetEvent (POINT1D_PACKET,
    caerPoint1DIteratorCounter);
```

Reverse iterator over all Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEvent.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.11 CAER_POINT1D_REVERSE_ITERATOR_VALID_END

```
#define CAER_POINT1D_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.17.2.12 CAER_POINT1D_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT1D_REVERSE_ITERATOR_VALID_START(
    POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter
    = caerEventPacketHeaderGetEventNumber (& (POINT1D_PACKET) ->
    packetHeader) - 1; \
    caerPoint1DIteratorCounter >= 0; caerPoint1DIteratorCounter--) {
    caerPoint1DEvent caerPoint1DIteratorElement
    = caerPoint1DEventPacketGetEvent (POINT1D_PACKET,
    caerPoint1DIteratorCounter); \
    if (!caerPoint1DEventIsValid(caerPoint1DIteratorElement)) {
        continue;
    }
```

Reverse iterator over only the valid Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEvent.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.13 POINT1D_SCALE_MASK

```
#define POINT1D_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Point1D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see ['common.h'](#) for more details.

4.17.2.14 POINT1D_SCALE_SHIFT

```
#define POINT1D_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Point1D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see 'common.h' for more details.

4.17.2.15 POINT1D_TYPE_MASK

```
#define POINT1D_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Point1D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see 'common.h' for more details.

4.17.2.16 POINT1D_TYPE_SHIFT

```
#define POINT1D_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Point1D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see 'common.h' for more details.

4.17.3 Typedef Documentation

4.17.3.1 caerPoint1DEvent

```
typedef struct caer_point1d_event* caerPoint1DEvent
```

Type for pointer to Point1D event data structure.

4.17.3.2 caerPoint1DEventPacket

```
typedef struct caer_point1d_event_packet* caerPoint1DEventPacket
```

Type for pointer to Point1D event packet data structure.

4.17.4 Function Documentation

4.17.4.1 caerPoint1DEventGetScale()

```
static int8_t caerPoint1DEventGetScale (
    caerPoint1DEventConst event ) [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^{-2}) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

<i>event</i>	a valid Point1DEvent pointer. Cannot be NULL.
--------------	---

Returns

the Point1D measurement scale.

4.17.4.2 caerPoint1DEventGetTimestamp()

```
static int32_t caerPoint1DEventGetTimestamp (
    caerPoint1DEventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid Point1DEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond timestamp.

4.17.4.3 caerPoint1DEventGetTimestamp64()

```
static int64_t caerPoint1DEventGetTimestamp64 (
    caerPoint1DEventConst event,
    caerPoint1DEventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid Point1DEvent pointer. Cannot be NULL.
<i>packet</i>	the Point1DEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.17.4.4 caerPoint1DEventGetType()

```
static uint8_t caerPoint1DEventGetType (
    caerPoint1DEventConst event ) [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

<i>event</i>	a valid Point1DEvent pointer. Cannot be NULL.
--------------	---

Returns

the Point1D measurement type.

4.17.4.5 caerPoint1DEventGetX()

```
static float caerPoint1DEventGetX (
    caerPoint1DEventConst event ) [inline], [static]
```

Get the X axis measurement.

Parameters

<i>event</i>	a valid Point1DEvent pointer. Cannot be NULL.
--------------	---

Returns

X axis measurement.

4.17.4.6 caerPoint1DEventInvalidate()

```
static void caerPoint1DEventInvalidate (
    caerPoint1DEvent event,
    caerPoint1DEventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid Point1DEvent pointer. Cannot be NULL.
<i>packet</i>	the Point1DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.17.4.7 caerPoint1DEventIsValid()

```
static bool caerPoint1DEventIsValid (
    caerPoint1DEventConst event ) [inline], [static]
```

Check if this Point1D event is valid.

Parameters

<i>event</i>	a valid Point1DEvent pointer. Cannot be NULL.
--------------	---

Returns

true if valid, false if not.

4.17.4.8 caerPoint1DEventPacketAllocate()

```
static caerPoint1DEventPacket caerPoint1DEventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]
```

Allocate a new Point1D events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid Point1DEventPacket handle or NULL on error.

4.17.4.9 caerPoint1DEventPacketFromPacketHeader()

```
static caerPoint1DEventPacket caerPoint1DEventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```

Transform a generic event packet header into a Point1D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.17.4.10 caerPoint1DEventPacketFromPacketHeaderConst()

```
static caerPoint1DEventPacketConst caerPoint1DEventPacketFromPacketHeaderConst (  
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only Point1D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.17.4.11 caerPoint1DEventPacketGetEvent()

```
static caerPoint1DEvent caerPoint1DEventPacketGetEvent (  
    caerPoint1DEventPacket packet,  
    int32_t n ) [inline], [static]
```

Get the Point1D event at the given index from the event packet.

Parameters

<i>packet</i>	a valid Point1DEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Point1D event. NULL on error.

4.17.4.12 caerPoint1DEventPacketGetEventConst()

```
static caerPoint1DEventConst caerPoint1DEventPacketGetEventConst (
    caerPoint1DEventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the Point1D event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid Point1DEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only Point1D event. NULL on error.

4.17.4.13 caerPoint1DEventSetScale()

```
static void caerPoint1DEventSetScale (
    caerPoint1DEvent event,
    int8_t scale ) [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^{-2}) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

<i>event</i>	a valid Point1DEvent pointer. Cannot be NULL.
<i>scale</i>	the Point1D measurement scale.

4.17.4.14 caerPoint1DEventSetTimestamp()

```
static void caerPoint1DEventSetTimestamp (
    caerPoint1DEvent event,
    int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid Point1DEvent pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.17.4.15 caerPoint1DEventSetType()

```
static void caerPoint1DEventSetType (
    caerPoint1DEvent event,
    uint8_t type ) [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

<i>event</i>	a valid Point1DEvent pointer. Cannot be NULL.
<i>type</i>	the Point1D measurement type.

4.17.4.16 caerPoint1DEventSetX()

```
static void caerPoint1DEventSetX (
    caerPoint1DEvent event,
    float x ) [inline], [static]
```

Set the X axis measurement.

Parameters

<i>event</i>	a valid Point1DEvent pointer. Cannot be NULL.
<i>x</i>	X axis measurement.

4.17.4.17 caerPoint1DEventValidate()

```
static void caerPoint1DEventValidate (
    caerPoint1DEvent event,
    caerPoint1DEventPacket packet ) [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid Point1DEvent pointer. Cannot be NULL.
<i>packet</i>	the Point1DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.17.4.18 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
    struct caer_point1d_event { uint32_t info;float x;int32_t timestamp;} )
```

Point1D event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The one measurement (x) is stored as a float. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.17.4.19 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
    struct caer_point1d_event_packet { struct caer_event_packet_header packetHeader;struct
    caer_point1d_event events[];} )
```

Point1D event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.18 events/point2d.h File Reference

```
#include "common.h"
```

Macros

- `#define CAER_POINT2D_ITERATOR_ALL_START(POINT2D_PACKET)`
 - `#define CAER_POINT2D_CONST_ITERATOR_ALL_START(POINT2D_PACKET)`
 - `#define CAER_POINT2D_ITERATOR_ALL_END }`
 - `#define CAER_POINT2D_ITERATOR_VALID_START(POINT2D_PACKET)`
 - `#define CAER_POINT2D_CONST_ITERATOR_VALID_START(POINT2D_PACKET)`
 - `#define CAER_POINT2D_ITERATOR_VALID_END }`
 - `#define CAER_POINT2D_REVERSE_ITERATOR_ALL_START(POINT2D_PACKET)`
 - `#define CAER_POINT2D_CONST_REVERSE_ITERATOR_ALL_START(POINT2D_PACKET)`
 - `#define CAER_POINT2D_REVERSE_ITERATOR_ALL_END }`
 - `#define CAER_POINT2D_REVERSE_ITERATOR_VALID_START(POINT2D_PACKET)`
 - `#define CAER_POINT2D_CONST_REVERSE_ITERATOR_VALID_START(POINT2D_PACKET)`
 - `#define CAER_POINT2D_REVERSE_ITERATOR_VALID_END }`
-
- `#define POINT2D_TYPE_SHIFT 1`
 - `#define POINT2D_TYPE_MASK 0x0000007F`
 - `#define POINT2D_SCALE_SHIFT 8`
 - `#define POINT2D_SCALE_MASK 0x000000FF`

Typedefs

- typedef struct caer_point2d_event * [caerPoint2DEvent](#)
- typedef const struct caer_point2d_event * **caerPoint2DEventConst**
- typedef struct caer_point2d_event_packet * [caerPoint2DEventPacket](#)
- typedef const struct caer_point2d_event_packet * **caerPoint2DEventPacketConst**

Functions

- [PACKED_STRUCT](#) (struct caer_point2d_event { uint32_t info;float x;float y;int32_t timestamp;})
- [PACKED_STRUCT](#) (struct caer_point2d_event_packet { struct caer_event_packet_header packet↵ Header;struct caer_point2d_event events[];})
- static [caerPoint2DEventPacket](#) [caerPoint2DEventPacketAllocate](#) (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static [caerPoint2DEventPacket](#) [caerPoint2DEventPacketFromPacketHeader](#) ([caerEventPacketHeader](#) header)
- static [caerPoint2DEventPacketConst](#) [caerPoint2DEventPacketFromPacketHeaderConst](#) ([caerEventPacketHeaderConst](#) header)
- static [caerPoint2DEvent](#) [caerPoint2DEventPacketGetEvent](#) ([caerPoint2DEventPacket](#) packet, int32_t n)
- static [caerPoint2DEventConst](#) [caerPoint2DEventPacketGetEventConst](#) ([caerPoint2DEventPacketConst](#) packet, int32_t n)
- static int32_t [caerPoint2DEventGetTimestamp](#) ([caerPoint2DEventConst](#) event)
- static int64_t [caerPoint2DEventGetTimestamp64](#) ([caerPoint2DEventConst](#) event, [caerPoint2DEventPacketConst](#) packet)
- static void [caerPoint2DEventSetTimestamp](#) ([caerPoint2DEvent](#) event, int32_t timestamp)
- static bool [caerPoint2DEventsValid](#) ([caerPoint2DEventConst](#) event)
- static void [caerPoint2DEventValidate](#) ([caerPoint2DEvent](#) event, [caerPoint2DEventPacket](#) packet)
- static void [caerPoint2DEventInvalidate](#) ([caerPoint2DEvent](#) event, [caerPoint2DEventPacket](#) packet)
- static uint8_t [caerPoint2DEventGetType](#) ([caerPoint2DEventConst](#) event)
- static void [caerPoint2DEventSetType](#) ([caerPoint2DEvent](#) event, uint8_t type)
- static int8_t [caerPoint2DEventGetScale](#) ([caerPoint2DEventConst](#) event)
- static void [caerPoint2DEventSetScale](#) ([caerPoint2DEvent](#) event, int8_t scale)
- static float [caerPoint2DEventGetX](#) ([caerPoint2DEventConst](#) event)
- static void [caerPoint2DEventSetX](#) ([caerPoint2DEvent](#) event, float x)
- static float [caerPoint2DEventGetY](#) ([caerPoint2DEventConst](#) event)
- static void [caerPoint2DEventSetY](#) ([caerPoint2DEvent](#) event, float y)

4.18.1 Detailed Description

Point2D Events format definition and handling functions. This contains two dimensional data points as floats, together with support for distinguishing type and scale.

4.18.2 Macro Definition Documentation

4.18.2.1 CAER_POINT2D_CONST_ITERATOR_ALL_START

```
#define CAER_POINT2D_CONST_ITERATOR_ALL_START(  
    POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter = 0;  
     caerPoint2DIteratorCounter < caerEventPacketHeaderGetEventNumber  
     (&(POINT2D_PACKET)->packetHeader); \  
     caerPoint2DIteratorCounter++) {  
    \  
    caerPoint2DEventConst caerPoint2DIteratorElement  
    \  
    = caerPoint2DEventPacketGetEventConst (POINT2D_PACKET,  
    caerPoint2DIteratorCounter);
```

Const-Iterator over all Point2D events in a packet. Returns the current index in the 'caerPoint2DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint2DIteratorElement' variable of type caer↔Point2DEventConst.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.2 CAER_POINT2D_CONST_ITERATOR_VALID_START

```
#define CAER_POINT2D_CONST_ITERATOR_VALID_START(  
    POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter = 0;  
     caerPoint2DIteratorCounter < caerEventPacketHeaderGetEventNumber  
     (&(POINT2D_PACKET)->packetHeader); \  
     caerPoint2DIteratorCounter++) {  
    \  
    caerPoint2DEventConst caerPoint2DIteratorElement  
    \  
    = caerPoint2DEventPacketGetEventConst (POINT2D_PACKET,  
    caerPoint2DIteratorCounter);  
    if (!caerPoint2DEventIsValid(caerPoint2DIteratorElement)) {  
        \  
        continue;  
    }  
    \  
}
```

Const-Iterator over only the valid Point2D events in a packet. Returns the current index in the 'caerPoint2DIterator↔Counter' variable of type 'int32_t' and the current read-only event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEventConst.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.3 CAER_POINT2D_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT2D_CONST_REVERSE_ITERATOR_ALL_START(  
    POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter  
     = caerEventPacketHeaderGetEventNumber (&(POINT2D_PACKET)->  
     packetHeader) - 1; \  
     caerPoint2DIteratorCounter >= 0; caerPoint2DIteratorCounter--) {  
    caerPoint2DEventConst caerPoint2DIteratorElement  
    = caerPoint2DEventPacketGetEventConst (POINT2D_PACKET,  
    caerPoint2DIteratorCounter);
```

Const-Reverse iterator over all Point2D events in a packet. Returns the current index in the 'caerPoint2DIterator↔Counter' variable of type 'int32_t' and the current read-only event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEventConst.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.4 CAER_POINT2D_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT2D_CONST_REVERSE_ITERATOR_VALID_START(
    POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter
    = caerEventPacketHeaderGetEventNumber (&(POINT2D_PACKET)->
    packetHeader) - 1;
    caerPoint2DIteratorCounter >= 0; caerPoint2DIteratorCounter--) {
    caerPoint2DEventConst caerPoint2DIteratorElement
    = caerPoint2DEventPacketGetEventConst (POINT2D_PACKET,
    caerPoint2DIteratorCounter);
    if (!caerPoint2DIsValid(caerPoint2DIteratorElement)) {
        continue;
    }
```

Const-Reverse iterator over only the valid Point2D events in a packet. Returns the current index in the 'caerPoint2DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEventConst.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.5 CAER_POINT2D_ITERATOR_ALL_END

```
#define CAER_POINT2D_ITERATOR_ALL_END }
```

Iterator close statement.

4.18.2.6 CAER_POINT2D_ITERATOR_ALL_START

```
#define CAER_POINT2D_ITERATOR_ALL_START(
    POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter = 0;
    caerPoint2DIteratorCounter < caerEventPacketHeaderGetEventNumber
    (&(POINT2D_PACKET)->packetHeader);
    caerPoint2DIteratorCounter++) {
    caerPoint2DEvent caerPoint2DIteratorElement
    = caerPoint2DEventPacketGetEvent (POINT2D_PACKET,
    caerPoint2DIteratorCounter);
```

Iterator over all Point2D events in a packet. Returns the current index in the 'caerPoint2DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEvent.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.7 CAER_POINT2D_ITERATOR_VALID_END

```
#define CAER_POINT2D_ITERATOR_VALID_END }
```

Iterator close statement.

4.18.2.8 CAER_POINT2D_ITERATOR_VALID_START

```
#define CAER_POINT2D_ITERATOR_VALID_START (
    POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter = 0;
     caerPoint2DIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(POINT2D_PACKET)->packetHeader); \
     caerPoint2DIteratorCounter++) {
    caerPoint2DEvent caerPoint2DIteratorElement
    \
    = caerPoint2DEventPacketGetEvent (POINT2D_PACKET,
    caerPoint2DIteratorCounter);
    if (!caerPoint2DEventIsValid(caerPoint2DIteratorElement)) {
        continue;
    }
}
```

Iterator over only the valid Point2D events in a packet. Returns the current index in the 'caerPoint2DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEvent.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.9 CAER_POINT2D_REVERSE_ITERATOR_ALL_END

```
#define CAER_POINT2D_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.18.2.10 CAER_POINT2D_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT2D_REVERSE_ITERATOR_ALL_START (
    POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter
     = caerEventPacketHeaderGetEventNumber (&(POINT2D_PACKET)->
     packetHeader) - 1; \
     caerPoint2DIteratorCounter >= 0; caerPoint2DIteratorCounter--) {
    caerPoint2DEvent caerPoint2DIteratorElement
    \
    = caerPoint2DEventPacketGetEvent (POINT2D_PACKET,
    caerPoint2DIteratorCounter);
}
```

Reverse iterator over all Point2D events in a packet. Returns the current index in the 'caerPoint2DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEvent.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.11 CAER_POINT2D_REVERSE_ITERATOR_VALID_END

```
#define CAER_POINT2D_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.18.2.12 CAER_POINT2D_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT2D_REVERSE_ITERATOR_VALID_START(
    POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter = caerEventPacketHeaderGetEventNumber (& (POINT2D_PACKET) ->
packetHeader) - 1; caerPoint2DIteratorCounter >= 0; caerPoint2DIteratorCounter--) {
    caerPoint2DEvent caerPoint2DIteratorElement = caerPoint2DEventPacketGetEvent (POINT2D_PACKET,
caerPoint2DIteratorCounter);
    if (!caerPoint2DEventIsValid (caerPoint2DIteratorElement)) {
        continue;
    }
```

Reverse iterator over only the valid Point2D events in a packet. Returns the current index in the 'caerPoint2DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEvent.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.13 POINT2D_SCALE_MASK

```
#define POINT2D_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Point2D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see 'common.h' for more details.

4.18.2.14 POINT2D_SCALE_SHIFT

```
#define POINT2D_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Point2D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see 'common.h' for more details.

4.18.2.15 POINT2D_TYPE_MASK

```
#define POINT2D_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Point2D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see 'common.h' for more details.

4.18.2.16 POINT2D_TYPE_SHIFT

```
#define POINT2D_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Point2D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see 'common.h' for more details.

4.18.3 Typedef Documentation

4.18.3.1 caerPoint2DEvent

```
typedef struct caer_point2d_event* caerPoint2DEvent
```

Type for pointer to Point2D event data structure.

4.18.3.2 caerPoint2DEventPacket

```
typedef struct caer_point2d_event_packet* caerPoint2DEventPacket
```

Type for pointer to Point2D event packet data structure.

4.18.4 Function Documentation

4.18.4.1 caerPoint2DEventGetScale()

```
static int8_t caerPoint2DEventGetScale (
    caerPoint2DEventConst event ) [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^{-2}) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
--------------	---

Returns

the Point2D measurement scale.

4.18.4.2 caerPoint2DEventGetTimestamp()

```
static int32_t caerPoint2DEventGetTimestamp (
    caerPoint2DEventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond timestamp.

4.18.4.3 caerPoint2DEventGetTimestamp64()

```
static int64_t caerPoint2DEventGetTimestamp64 (  
    caerPoint2DEventConst event,  
    caerPoint2DEventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
<i>packet</i>	the Point2DEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.18.4.4 caerPoint2DEventGetType()

```
static uint8_t caerPoint2DEventGetType (  
    caerPoint2DEventConst event ) [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
--------------	---

Returns

the Point2D measurement type.

4.18.4.5 caerPoint2DEventGetX()

```
static float caerPoint2DEventGetX (
    caerPoint2DEventConst event ) [inline], [static]
```

Get the X axis measurement.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
--------------	---

Returns

X axis measurement.

4.18.4.6 caerPoint2DEventGetY()

```
static float caerPoint2DEventGetY (
    caerPoint2DEventConst event ) [inline], [static]
```

Get the Y axis measurement.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
--------------	---

Returns

Y axis measurement.

4.18.4.7 caerPoint2DEventInvalidate()

```
static void caerPoint2DEventInvalidate (
    caerPoint2DEvent event,
    caerPoint2DEventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
<i>packet</i>	the Point2DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.18.4.8 caerPoint2DEventsValid()

```
static bool caerPoint2DEventIsValid (
    caerPoint2DEventConst event ) [inline], [static]
```

Check if this Point2D event is valid.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
--------------	---

Returns

true if valid, false if not.

4.18.4.9 caerPoint2DEventPacketAllocate()

```
static caerPoint2DEventPacket caerPoint2DEventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]
```

Allocate a new Point2D events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid Point2DEventPacket handle or NULL on error.

4.18.4.10 caerPoint2DEventPacketFromPacketHeader()

```
static caerPoint2DEventPacket caerPoint2DEventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```

Transform a generic event packet header into a Point2D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.18.4.11 caerPoint2DEventPacketFromPacketHeaderConst()

```
static caerPoint2DEventPacketConst caerPoint2DEventPacketFromPacketHeaderConst (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only Point2D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.18.4.12 caerPoint2DEventPacketGetEvent()

```
static caerPoint2DEvent caerPoint2DEventPacketGetEvent (
    caerPoint2DEventPacket packet,
    int32_t n ) [inline], [static]
```

Get the Point2D event at the given index from the event packet.

Parameters

<i>packet</i>	a valid Point2DEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Point2D event. NULL on error.

4.18.4.13 caerPoint2DEventPacketGetEventConst()

```
static caerPoint2DEventConst caerPoint2DEventPacketGetEventConst (
    caerPoint2DEventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the Point2D event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid Point2DEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only Point2D event. NULL on error.

4.18.4.14 caerPoint2DEventSetScale()

```
static void caerPoint2DEventSetScale (
    caerPoint2DEvent event,
    int8_t scale ) [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^{-2}) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
<i>scale</i>	the Point2D measurement scale.

4.18.4.15 caerPoint2DEventSetTimestamp()

```
static void caerPoint2DEventSetTimestamp (
    caerPoint2DEvent event,
    int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.18.4.16 caerPoint2DEventSetType()

```
static void caerPoint2DEventSetType (
    caerPoint2DEvent event,
    uint8_t type ) [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
<i>type</i>	the Point2D measurement type.

4.18.4.17 caerPoint2DEventSetX()

```
static void caerPoint2DEventSetX (
    caerPoint2DEvent event,
    float x ) [inline], [static]
```

Set the X axis measurement.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
<i>x</i>	X axis measurement.

4.18.4.18 caerPoint2DEventSetY()

```
static void caerPoint2DEventSetY (
    caerPoint2DEvent event,
    float y ) [inline], [static]
```

Set the Y axis measurement.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
<i>y</i>	Y axis measurement.

4.18.4.19 caerPoint2DEventValidate()

```
static void caerPoint2DEventValidate (
    caerPoint2DEvent event,
    caerPoint2DEventPacket packet ) [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid Point2DEvent pointer. Cannot be NULL.
<i>packet</i>	the Point2DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.18.4.20 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
    struct caer_point2d_event { uint32_t info;float x;float y;int32_t timestamp;} )
```

Point2D event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The two measurements (x, y) are stored as floats. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.18.4.21 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
    struct caer_point2d_event_packet { struct caer_event_packet_header packetHeader;struct
    caer_point2d_event events[];} )
```

Point2D event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.19 events/point3d.h File Reference

```
#include "common.h"
```

Macros

- `#define CAER_POINT3D_ITERATOR_ALL_START`(POINT3D_PACKET)
 - `#define CAER_POINT3D_CONST_ITERATOR_ALL_START`(POINT3D_PACKET)
 - `#define CAER_POINT3D_ITERATOR_ALL_END` }
 - `#define CAER_POINT3D_ITERATOR_VALID_START`(POINT3D_PACKET)
 - `#define CAER_POINT3D_CONST_ITERATOR_VALID_START`(POINT3D_PACKET)
 - `#define CAER_POINT3D_ITERATOR_VALID_END` }
 - `#define CAER_POINT3D_REVERSE_ITERATOR_ALL_START`(POINT3D_PACKET)
 - `#define CAER_POINT3D_CONST_REVERSE_ITERATOR_ALL_START`(POINT3D_PACKET)
 - `#define CAER_POINT3D_REVERSE_ITERATOR_ALL_END` }
 - `#define CAER_POINT3D_REVERSE_ITERATOR_VALID_START`(POINT3D_PACKET)
 - `#define CAER_POINT3D_CONST_REVERSE_ITERATOR_VALID_START`(POINT3D_PACKET)
 - `#define CAER_POINT3D_REVERSE_ITERATOR_VALID_END` }
-
- `#define POINT3D_TYPE_SHIFT` 1
 - `#define POINT3D_TYPE_MASK` 0x0000007F
 - `#define POINT3D_SCALE_SHIFT` 8
 - `#define POINT3D_SCALE_MASK` 0x000000FF

Typedefs

- `typedef struct caer_point3d_event * caerPoint3DEvent`
- `typedef const struct caer_point3d_event * caerPoint3DEventConst`
- `typedef struct caer_point3d_event_packet * caerPoint3DEventPacket`
- `typedef const struct caer_point3d_event_packet * caerPoint3DEventPacketConst`

Functions

- `PACKED_STRUCT` (struct caer_point3d_event { uint32_t info;float x;float y;float z;int32_t timestamp;})
- `PACKED_STRUCT` (struct caer_point3d_event_packet { struct caer_event_packet_header packet↵ Header;struct caer_point3d_event events[];})
- static `caerPoint3DEventPacket caerPoint3DEventPacketAllocate` (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static `caerPoint3DEventPacket caerPoint3DEventPacketFromPacketHeader` (caerEventPacketHeader header)
- static `caerPoint3DEventPacketConst caerPoint3DEventPacketFromPacketHeaderConst` (caerEventPacket↵ HeaderConst header)
- static `caerPoint3DEvent caerPoint3DEventPacketGetEvent` (caerPoint3DEventPacket packet, int32_t n)
- static `caerPoint3DEventConst caerPoint3DEventPacketGetEventConst` (caerPoint3DEventPacketConst packet, int32_t n)
- static `int32_t caerPoint3DEventGetTimestamp` (caerPoint3DEventConst event)
- static `int64_t caerPoint3DEventGetTimestamp64` (caerPoint3DEventConst event, caerPoint3DEventPacket↵ Const packet)
- static void `caerPoint3DEventSetTimestamp` (caerPoint3DEvent event, int32_t timestamp)
- static bool `caerPoint3DEventIsValid` (caerPoint3DEventConst event)
- static void `caerPoint3DEventValidate` (caerPoint3DEvent event, caerPoint3DEventPacket packet)
- static void `caerPoint3DEventInvalidate` (caerPoint3DEvent event, caerPoint3DEventPacket packet)
- static `uint8_t caerPoint3DEventGetType` (caerPoint3DEventConst event)
- static void `caerPoint3DEventSetType` (caerPoint3DEvent event, uint8_t type)

- static int8_t [caerPoint3DEventGetScale](#) (caerPoint3DEventConst event)
- static void [caerPoint3DEventSetScale](#) (caerPoint3DEvent event, int8_t scale)
- static float [caerPoint3DEventGetX](#) (caerPoint3DEventConst event)
- static void [caerPoint3DEventSetX](#) (caerPoint3DEvent event, float x)
- static float [caerPoint3DEventGetY](#) (caerPoint3DEventConst event)
- static void [caerPoint3DEventSetY](#) (caerPoint3DEvent event, float y)
- static float [caerPoint3DEventGetZ](#) (caerPoint3DEventConst event)
- static void [caerPoint3DEventSetZ](#) (caerPoint3DEvent event, float z)

4.19.1 Detailed Description

Point3D Events format definition and handling functions. This contains three dimensional data points as floats, together with support for distinguishing type and scale.

4.19.2 Macro Definition Documentation

4.19.2.1 CAER_POINT3D_CONST_ITERATOR_ALL_START

```
#define CAER_POINT3D_CONST_ITERATOR_ALL_START(  
    POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter = 0; \
     caerPoint3DIteratorCounter < caerEventPacketHeaderGetEventNumber \
     (&(POINT3D_PACKET)->packetHeader); \
     caerPoint3DIteratorCounter++) { \
    \
    caerPoint3DEventConst caerPoint3DIteratorElement \
    \
    = caerPoint3DEventPacketGetEventConst (POINT3D_PACKET, \
    caerPoint3DIteratorCounter);
```

Const-Iterator over all Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEventConst.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.2 CAER_POINT3D_CONST_ITERATOR_VALID_START

```
#define CAER_POINT3D_CONST_ITERATOR_VALID_START (
    POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter = 0;
     caerPoint3DIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(POINT3D_PACKET)->packetHeader); \
     caerPoint3DIteratorCounter++) {
    caerPoint3DEventConst caerPoint3DIteratorElement
    = caerPoint3DEventPacketGetEventConst (POINT3D_PACKET,
    caerPoint3DIteratorCounter);
    if (!caerPoint3DEventIsValid(caerPoint3DIteratorElement)) {
        continue;
    }
```

Const-Iterator over only the valid Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEventConst.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.3 CAER_POINT3D_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT3D_CONST_REVERSE_ITERATOR_ALL_START (
    POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter
     = caerEventPacketHeaderGetEventNumber (&(POINT3D_PACKET)->
     packetHeader) - 1; \
     caerPoint3DIteratorCounter >= 0; caerPoint3DIteratorCounter--) {
    caerPoint3DEventConst caerPoint3DIteratorElement
    = caerPoint3DEventPacketGetEventConst (POINT3D_PACKET,
    caerPoint3DIteratorCounter);
```

Const-Reverse iterator over all Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEventConst.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.4 CAER_POINT3D_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT3D_CONST_REVERSE_ITERATOR_VALID_START (
    POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter
     = caerEventPacketHeaderGetEventNumber (&(POINT3D_PACKET)->
     packetHeader) - 1; \
     caerPoint3DIteratorCounter >= 0; caerPoint3DIteratorCounter--) {
    caerPoint3DEventConst caerPoint3DIteratorElement
    = caerPoint3DEventPacketGetEventConst (POINT3D_PACKET,
    caerPoint3DIteratorCounter);
    if (!caerPoint3DEventIsValid(caerPoint3DIteratorElement)) {
        continue;
    }
```

Const-Reverse iterator over only the valid Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEventConst.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.5 CAER_POINT3D_ITERATOR_ALL_END

```
#define CAER_POINT3D_ITERATOR_ALL_END }
```

Iterator close statement.

4.19.2.6 CAER_POINT3D_ITERATOR_ALL_START

```
#define CAER_POINT3D_ITERATOR_ALL_START(  
    POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter = 0; \
     caerPoint3DIteratorCounter < caerEventPacketHeaderGetEventNumber  
    (&(POINT3D_PACKET)->packetHeader); \
     caerPoint3DIteratorCounter++) {  
    \
    caerPoint3DEvent caerPoint3DIteratorElement  
    \
    = caerPoint3DEventPacketGetEvent (POINT3D_PACKET,  
    caerPoint3DIteratorCounter);
```

Iterator over all Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEvent.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.7 CAER_POINT3D_ITERATOR_VALID_END

```
#define CAER_POINT3D_ITERATOR_VALID_END }
```

Iterator close statement.

4.19.2.8 CAER_POINT3D_ITERATOR_VALID_START

```
#define CAER_POINT3D_ITERATOR_VALID_START(  
    POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter = 0; \
     caerPoint3DIteratorCounter < caerEventPacketHeaderGetEventNumber  
    (&(POINT3D_PACKET)->packetHeader); \
     caerPoint3DIteratorCounter++) {  
    \
    caerPoint3DEvent caerPoint3DIteratorElement  
    \
    = caerPoint3DEventPacketGetEvent (POINT3D_PACKET,  
    caerPoint3DIteratorCounter);  
    if (!caerPoint3DEventIsValid(caerPoint3DIteratorElement)) {  
        \
        continue;  
    }  
    \
}
```

Iterator over only the valid Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEvent.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.9 CAER_POINT3D_REVERSE_ITERATOR_ALL_END

```
#define CAER_POINT3D_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.19.2.10 CAER_POINT3D_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT3D_REVERSE_ITERATOR_ALL_START (
    POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter
    = caerEventPacketHeaderGetEventNumber (& (POINT3D_PACKET) ->
    packetHeader) - 1; \
    caerPoint3DIteratorCounter >= 0; caerPoint3DIteratorCounter--) {
    caerPoint3DEvent caerPoint3DIteratorElement
    = caerPoint3DEventPacketGetEvent (POINT3D_PACKET,
    caerPoint3DIteratorCounter);
```

Reverse iterator over all Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEvent.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.11 CAER_POINT3D_REVERSE_ITERATOR_VALID_END

```
#define CAER_POINT3D_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.19.2.12 CAER_POINT3D_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT3D_REVERSE_ITERATOR_VALID_START (
    POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter
    = caerEventPacketHeaderGetEventNumber (& (POINT3D_PACKET) ->
    packetHeader) - 1; \
    caerPoint3DIteratorCounter >= 0; caerPoint3DIteratorCounter--) {
    caerPoint3DEvent caerPoint3DIteratorElement
    = caerPoint3DEventPacketGetEvent (POINT3D_PACKET,
    caerPoint3DIteratorCounter); \
    if (!caerPoint3DEventIsValid(caerPoint3DIteratorElement)) {
        continue;
    }
```

Reverse iterator over only the valid Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEvent.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.13 POINT3D_SCALE_MASK

```
#define POINT3D_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Point3D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.19.2.14 POINT3D_SCALE_SHIFT

```
#define POINT3D_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Point3D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.19.2.15 POINT3D_TYPE_MASK

```
#define POINT3D_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Point3D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.19.2.16 POINT3D_TYPE_SHIFT

```
#define POINT3D_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Point3D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.19.3 Typedef Documentation

4.19.3.1 caerPoint3DEvent

```
typedef struct caer_point3d_event* caerPoint3DEvent
```

Type for pointer to Point3D event data structure.

4.19.3.2 caerPoint3DEventPacket

```
typedef struct caer_point3d_event_packet* caerPoint3DEventPacket
```

Type for pointer to Point3D event packet data structure.

4.19.4 Function Documentation

4.19.4.1 caerPoint3DEventGetScale()

```
static int8_t caerPoint3DEventGetScale (
    caerPoint3DEventConst event ) [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^{-2}) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
--------------	---

Returns

the Point3D measurement scale.

4.19.4.2 caerPoint3DEventGetTimestamp()

```
static int32_t caerPoint3DEventGetTimestamp (
    caerPoint3DEventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond timestamp.

4.19.4.3 caerPoint3DEventGetTimestamp64()

```
static int64_t caerPoint3DEventGetTimestamp64 (
    caerPoint3DEventConst event,
    caerPoint3DEventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
<i>packet</i>	the Point3DEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.19.4.4 caerPoint3DEventGetType()

```
static uint8_t caerPoint3DEventGetType (  
    caerPoint3DEventConst event ) [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
--------------	---

Returns

the Point3D measurement type.

4.19.4.5 caerPoint3DEventGetX()

```
static float caerPoint3DEventGetX (  
    caerPoint3DEventConst event ) [inline], [static]
```

Get the X axis measurement.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
--------------	---

Returns

X axis measurement.

4.19.4.6 caerPoint3DEventGetY()

```
static float caerPoint3DEventGetY (
    caerPoint3DEventConst event ) [inline], [static]
```

Get the Y axis measurement.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
--------------	---

Returns

Y axis measurement.

4.19.4.7 caerPoint3DEventGetZ()

```
static float caerPoint3DEventGetZ (
    caerPoint3DEventConst event ) [inline], [static]
```

Get the Z axis measurement.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
--------------	---

Returns

Z axis measurement.

4.19.4.8 caerPoint3DEventInvalidate()

```
static void caerPoint3DEventInvalidate (
    caerPoint3DEvent event,
    caerPoint3DEventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
<i>packet</i>	the Point3DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.19.4.9 caerPoint3DEventsValid()

```
static bool caerPoint3DEventIsValid (
    caerPoint3DEventConst event ) [inline], [static]
```

Check if this Point3D event is valid.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
--------------	---

Returns

true if valid, false if not.

4.19.4.10 caerPoint3DEventPacketAllocate()

```
static caerPoint3DEventPacket caerPoint3DEventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]
```

Allocate a new Point3D events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid Point3DEventPacket handle or NULL on error.

4.19.4.11 caerPoint3DEventPacketFromPacketHeader()

```
static caerPoint3DEventPacket caerPoint3DEventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```

Transform a generic event packet header into a Point3D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.19.4.12 caerPoint3DEventPacketFromPacketHeaderConst()

```
static caerPoint3DEventPacketConst caerPoint3DEventPacketFromPacketHeaderConst (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only Point3D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.19.4.13 caerPoint3DEventPacketGetEvent()

```
static caerPoint3DEvent caerPoint3DEventPacketGetEvent (
    caerPoint3DEventPacket packet,
    int32_t n ) [inline], [static]
```

Get the Point3D event at the given index from the event packet.

Parameters

<i>packet</i>	a valid Point3DEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Point3D event. NULL on error.

4.19.4.14 caerPoint3DEventPacketGetEventConst()

```
static caerPoint3DEventConst caerPoint3DEventPacketGetEventConst (
    caerPoint3DEventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the Point3D event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid Point3DEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only Point3D event. NULL on error.

4.19.4.15 caerPoint3DEventSetScale()

```
static void caerPoint3DEventSetScale (
    caerPoint3DEvent event,
    int8_t scale ) [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^{-2}) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
<i>scale</i>	the Point3D measurement scale.

4.19.4.16 caerPoint3DEventSetTimestamp()

```
static void caerPoint3DEventSetTimestamp (
    caerPoint3DEvent event,
    int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.19.4.17 caerPoint3DEventSetType()

```
static void caerPoint3DEventSetType (
    caerPoint3DEvent event,
    uint8_t type ) [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
<i>type</i>	the Point3D measurement type.

4.19.4.18 caerPoint3DEventSetX()

```
static void caerPoint3DEventSetX (
    caerPoint3DEvent event,
    float x ) [inline], [static]
```

Set the X axis measurement.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
<i>x</i>	X axis measurement.

4.19.4.19 caerPoint3DEventSetY()

```
static void caerPoint3DEventSetY (
    caerPoint3DEvent event,
    float y ) [inline], [static]
```

Set the Y axis measurement.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
<i>y</i>	Y axis measurement.

4.19.4.20 caerPoint3DEventSetZ()

```
static void caerPoint3DEventSetZ (
    caerPoint3DEvent event,
    float z ) [inline], [static]
```

Set the Z axis measurement.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
<i>z</i>	Z axis measurement.

4.19.4.21 caerPoint3DEventValidate()

```
static void caerPoint3DEventValidate (
    caerPoint3DEvent event,
    caerPoint3DEventPacket packet ) [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid Point3DEvent pointer. Cannot be NULL.
<i>packet</i>	the Point3DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.19.4.22 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
    struct caer_point3d_event { uint32_t info;float x;float y;float z;int32_t timestamp;}
)
```

Point3D event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The three measurements (x, y, z) are stored as floats. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.19.4.23 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
    struct caer_point3d_event_packet { struct caer_event_packet_header packetHeader;struct
    caer_point3d_event events[];} )
```

Point3D event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.20 events/point4d.h File Reference

```
#include "common.h"
```

Macros

- `#define CAER_POINT4D_ITERATOR_ALL_START`(POINT4D_PACKET)
 - `#define CAER_POINT4D_CONST_ITERATOR_ALL_START`(POINT4D_PACKET)
 - `#define CAER_POINT4D_ITERATOR_ALL_END` }
 - `#define CAER_POINT4D_ITERATOR_VALID_START`(POINT4D_PACKET)
 - `#define CAER_POINT4D_CONST_ITERATOR_VALID_START`(POINT4D_PACKET)
 - `#define CAER_POINT4D_ITERATOR_VALID_END` }
 - `#define CAER_POINT4D_REVERSE_ITERATOR_ALL_START`(POINT4D_PACKET)
 - `#define CAER_POINT4D_CONST_REVERSE_ITERATOR_ALL_START`(POINT4D_PACKET)
 - `#define CAER_POINT4D_REVERSE_ITERATOR_ALL_END` }
 - `#define CAER_POINT4D_REVERSE_ITERATOR_VALID_START`(POINT4D_PACKET)
 - `#define CAER_POINT4D_CONST_REVERSE_ITERATOR_VALID_START`(POINT4D_PACKET)
 - `#define CAER_POINT4D_REVERSE_ITERATOR_VALID_END` }
-
- `#define POINT4D_TYPE_SHIFT` 1
 - `#define POINT4D_TYPE_MASK` 0x0000007F
 - `#define POINT4D_SCALE_SHIFT` 8
 - `#define POINT4D_SCALE_MASK` 0x000000FF

Typedefs

- `typedef struct caer_point4d_event * caerPoint4DEvent`
- `typedef const struct caer_point4d_event * caerPoint4DEventConst`
- `typedef struct caer_point4d_event_packet * caerPoint4DEventPacket`
- `typedef const struct caer_point4d_event_packet * caerPoint4DEventPacketConst`

Functions

- `PACKED_STRUCT` (struct caer_point4d_event { uint32_t info; float x; float y; float z; float w; int32_t timestamp; })
- `PACKED_STRUCT` (struct caer_point4d_event_packet { struct caer_event_packet_header packet↵ Header; struct caer_point4d_event events[]; })
- static `caerPoint4DEventPacket caerPoint4DEventPacketAllocate` (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static `caerPoint4DEventPacket caerPoint4DEventPacketFromPacketHeader` (caerEventPacketHeader header)
- static `caerPoint4DEventPacketConst caerPoint4DEventPacketFromPacketHeaderConst` (caerEventPacket↵ HeaderConst header)
- static `caerPoint4DEvent caerPoint4DEventPacketGetEvent` (caerPoint4DEventPacket packet, int32_t n)
- static `caerPoint4DEventConst caerPoint4DEventPacketGetEventConst` (caerPoint4DEventPacketConst packet, int32_t n)
- static `int32_t caerPoint4DEventGetTimestamp` (caerPoint4DEventConst event)

- static int64_t [caerPoint4DEventGetTimestamp64](#) (caerPoint4DEventConst event, caerPoint4DEventPacket↔ Const packet)
- static void [caerPoint4DEventSetTimestamp](#) (caerPoint4DEvent event, int32_t timestamp)
- static bool [caerPoint4DEventsValid](#) (caerPoint4DEventConst event)
- static void [caerPoint4DEventValidate](#) (caerPoint4DEvent event, caerPoint4DEventPacket packet)
- static void [caerPoint4DEventInvalidate](#) (caerPoint4DEvent event, caerPoint4DEventPacket packet)
- static uint8_t [caerPoint4DEventGetType](#) (caerPoint4DEventConst event)
- static void [caerPoint4DEventSetType](#) (caerPoint4DEvent event, uint8_t type)
- static int8_t [caerPoint4DEventGetScale](#) (caerPoint4DEventConst event)
- static void [caerPoint4DEventSetScale](#) (caerPoint4DEvent event, int8_t scale)
- static float [caerPoint4DEventGetX](#) (caerPoint4DEventConst event)
- static void [caerPoint4DEventSetX](#) (caerPoint4DEvent event, float x)
- static float [caerPoint4DEventGetY](#) (caerPoint4DEventConst event)
- static void [caerPoint4DEventSetY](#) (caerPoint4DEvent event, float y)
- static float [caerPoint4DEventGetZ](#) (caerPoint4DEventConst event)
- static void [caerPoint4DEventSetZ](#) (caerPoint4DEvent event, float z)
- static float [caerPoint4DEventGetW](#) (caerPoint4DEventConst event)
- static void [caerPoint4DEventSetW](#) (caerPoint4DEvent event, float w)

4.20.1 Detailed Description

Point4D Events format definition and handling functions. This contains four dimensional data points as floats, together with support for distinguishing type and scale. Useful for homogeneous coordinates for example.

4.20.2 Macro Definition Documentation

4.20.2.1 CAER_POINT4D_CONST_ITERATOR_ALL_START

```
#define CAER_POINT4D_CONST_ITERATOR_ALL_START(  
    POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter = 0; \
     caerPoint4DIteratorCounter < caerEventPacketHeaderGetEventNumber \
     (&(POINT4D_PACKET)->packetHeader); \
     caerPoint4DIteratorCounter++) { \
    \
    caerPoint4DEventConst caerPoint4DIteratorElement \
    \
    = caerPoint4DEventPacketGetEventConst (POINT4D_PACKET, \
    caerPoint4DIteratorCounter);
```

Const-Iterator over all Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint4DIteratorElement' variable of type caer↔ Point4DEventConst.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.2 CAER_POINT4D_CONST_ITERATOR_VALID_START

```
#define CAER_POINT4D_CONST_ITERATOR_VALID_START (
    POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter = 0;
     caerPoint4DIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(POINT4D_PACKET)->packetHeader); \
     caerPoint4DIteratorCounter++) {
    caerPoint4DEventConst caerPoint4DIteratorElement
    = caerPoint4DEventPacketGetEventConst (POINT4D_PACKET,
    caerPoint4DIteratorCounter);
    if (!caerPoint4DEventIsValid(caerPoint4DIteratorElement)) {
        continue;
    }
```

Const-Iterator over only the valid Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEventConst.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.3 CAER_POINT4D_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT4D_CONST_REVERSE_ITERATOR_ALL_START (
    POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter
     = caerEventPacketHeaderGetEventNumber (&(POINT4D_PACKET)->
     packetHeader) - 1; \
     caerPoint4DIteratorCounter >= 0; caerPoint4DIteratorCounter--) {
    caerPoint4DEventConst caerPoint4DIteratorElement
    = caerPoint4DEventPacketGetEventConst (POINT4D_PACKET,
    caerPoint4DIteratorCounter);
```

Const-Reverse iterator over all Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEventConst.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.4 CAER_POINT4D_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT4D_CONST_REVERSE_ITERATOR_VALID_START (
    POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter
     = caerEventPacketHeaderGetEventNumber (&(POINT4D_PACKET)->
     packetHeader) - 1; \
     caerPoint4DIteratorCounter >= 0; caerPoint4DIteratorCounter--) {
    caerPoint4DEventConst caerPoint4DIteratorElement
    = caerPoint4DEventPacketGetEventConst (POINT4D_PACKET,
    caerPoint4DIteratorCounter);
    if (!caerPoint4DEventIsValid(caerPoint4DIteratorElement)) {
        continue;
    }
```

Const-Reverse iterator over only the valid Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEventConst.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.5 CAER_POINT4D_ITERATOR_ALL_END

```
#define CAER_POINT4D_ITERATOR_ALL_END }
```

Iterator close statement.

4.20.2.6 CAER_POINT4D_ITERATOR_ALL_START

```
#define CAER_POINT4D_ITERATOR_ALL_START(  
    POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter = 0; \
     caerPoint4DIteratorCounter < caerEventPacketHeaderGetEventNumber  
    (&(POINT4D_PACKET)->packetHeader); \
     caerPoint4DIteratorCounter++) { \
    caerPoint4DEvent caerPoint4DIteratorElement \
    = caerPoint4DEventPacketGetEvent (POINT4D_PACKET,  
    caerPoint4DIteratorCounter);
```

Iterator over all Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEvent.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.7 CAER_POINT4D_ITERATOR_VALID_END

```
#define CAER_POINT4D_ITERATOR_VALID_END }
```

Iterator close statement.

4.20.2.8 CAER_POINT4D_ITERATOR_VALID_START

```
#define CAER_POINT4D_ITERATOR_VALID_START(  
    POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter = 0; \
     caerPoint4DIteratorCounter < caerEventPacketHeaderGetEventNumber  
    (&(POINT4D_PACKET)->packetHeader); \
     caerPoint4DIteratorCounter++) { \
    caerPoint4DEvent caerPoint4DIteratorElement \
    = caerPoint4DEventPacketGetEvent (POINT4D_PACKET,  
    caerPoint4DIteratorCounter); \
    if (!caerPoint4DEventIsValid(caerPoint4DIteratorElement)) { \
        continue; \
    }
```

Iterator over only the valid Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEvent.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.9 CAER_POINT4D_REVERSE_ITERATOR_ALL_END

```
#define CAER_POINT4D_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.20.2.10 CAER_POINT4D_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT4D_REVERSE_ITERATOR_ALL_START (
    POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter
    = caerEventPacketHeaderGetEventNumber (& (POINT4D_PACKET) ->
    packetHeader) - 1; \
    caerPoint4DIteratorCounter >= 0; caerPoint4DIteratorCounter--) {
    caerPoint4DEvent caerPoint4DIteratorElement
    = caerPoint4DEventPacketGetEvent (POINT4D_PACKET,
    caerPoint4DIteratorCounter);
```

Reverse iterator over all Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEvent.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.11 CAER_POINT4D_REVERSE_ITERATOR_VALID_END

```
#define CAER_POINT4D_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.20.2.12 CAER_POINT4D_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT4D_REVERSE_ITERATOR_VALID_START (
    POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter
    = caerEventPacketHeaderGetEventNumber (& (POINT4D_PACKET) ->
    packetHeader) - 1; \
    caerPoint4DIteratorCounter >= 0; caerPoint4DIteratorCounter--) {
    caerPoint4DEvent caerPoint4DIteratorElement
    = caerPoint4DEventPacketGetEvent (POINT4D_PACKET,
    caerPoint4DIteratorCounter); \
    if (!caerPoint4DEventIsValid(caerPoint4DIteratorElement)) {
        continue;
    }
```

Reverse iterator over only the valid Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEvent.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.13 POINT4D_SCALE_MASK

```
#define POINT4D_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Point4D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.20.2.14 POINT4D_SCALE_SHIFT

```
#define POINT4D_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Point4D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.20.2.15 POINT4D_TYPE_MASK

```
#define POINT4D_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Point4D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.20.2.16 POINT4D_TYPE_SHIFT

```
#define POINT4D_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Point4D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^{-128} to 10^{127} . Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.20.3 Typedef Documentation

4.20.3.1 caerPoint4DEvent

```
typedef struct caer_point4d_event* caerPoint4DEvent
```

Type for pointer to Point4D event data structure.

4.20.3.2 caerPoint4DEventPacket

```
typedef struct caer_point4d_event_packet* caerPoint4DEventPacket
```

Type for pointer to Point4D event packet data structure.

4.20.4 Function Documentation

4.20.4.1 caerPoint4DEventGetScale()

```
static int8_t caerPoint4DEventGetScale (
    caerPoint4DEventConst event ) [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^{-2}) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
--------------	---

Returns

the Point4D measurement scale.

4.20.4.2 caerPoint4DEventGetTimestamp()

```
static int32_t caerPoint4DEventGetTimestamp (
    caerPoint4DEventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond timestamp.

4.20.4.3 caerPoint4DEventGetTimestamp64()

```
static int64_t caerPoint4DEventGetTimestamp64 (
    caerPoint4DEventConst event,
    caerPoint4DEventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
<i>packet</i>	the Point4DEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.20.4.4 caerPoint4DEventGetType()

```
static uint8_t caerPoint4DEventGetType (  
    caerPoint4DEventConst event ) [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
--------------	---

Returns

the Point4D measurement type.

4.20.4.5 caerPoint4DEventGetW()

```
static float caerPoint4DEventGetW (  
    caerPoint4DEventConst event ) [inline], [static]
```

Get the W axis measurement.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
--------------	---

Returns

W axis measurement.

4.20.4.6 caerPoint4DEventGetX()

```
static float caerPoint4DEventGetX (
    caerPoint4DEventConst event ) [inline], [static]
```

Get the X axis measurement.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
--------------	---

Returns

X axis measurement.

4.20.4.7 caerPoint4DEventGetY()

```
static float caerPoint4DEventGetY (
    caerPoint4DEventConst event ) [inline], [static]
```

Get the Y axis measurement.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
--------------	---

Returns

Y axis measurement.

4.20.4.8 caerPoint4DEventGetZ()

```
static float caerPoint4DEventGetZ (
    caerPoint4DEventConst event ) [inline], [static]
```

Get the Z axis measurement.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
--------------	---

Returns

Z axis measurement.

4.20.4.9 caerPoint4DEventInvalidate()

```
static void caerPoint4DEventInvalidate (
    caerPoint4DEvent event,
    caerPoint4DEventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
<i>packet</i>	the Point4DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.20.4.10 caerPoint4DEventIsValid()

```
static bool caerPoint4DEventIsValid (
    caerPoint4DEventConst event ) [inline], [static]
```

Check if this Point4D event is valid.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
--------------	---

Returns

true if valid, false if not.

4.20.4.11 caerPoint4DEventPacketAllocate()

```
static caerPoint4DEventPacket caerPoint4DEventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]
```

Allocate a new Point4D events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid `Point4DEventPacket` handle or `NULL` on error.

4.20.4.12 caerPoint4DEventPacketFromPacketHeader()

```
static caerPoint4DEventPacket caerPoint4DEventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```

Transform a generic event packet header into a `Point4D` event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be <code>NULL</code> .
---------------	--

Returns

a properly converted, typed event packet pointer.

4.20.4.13 caerPoint4DEventPacketFromPacketHeaderConst()

```
static caerPoint4DEventPacketConst caerPoint4DEventPacketFromPacketHeaderConst (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only `Point4D` event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be <code>NULL</code> .
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.20.4.14 caerPoint4DEventPacketGetEvent()

```
static caerPoint4DEvent caerPoint4DEventPacketGetEvent (
    caerPoint4DEventPacket packet,
    int32_t n ) [inline], [static]
```

Get the `Point4D` event at the given index from the event packet.

Parameters

<i>packet</i>	a valid Point4DEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Point4D event. NULL on error.

4.20.4.15 caerPoint4DEventPacketGetEventConst()

```
static caerPoint4DEventConst caerPoint4DEventPacketGetEventConst (
    caerPoint4DEventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the Point4D event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid Point4DEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only Point4D event. NULL on error.

4.20.4.16 caerPoint4DEventSetScale()

```
static void caerPoint4DEventSetScale (
    caerPoint4DEvent event,
    int8_t scale ) [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^{-2}) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
<i>scale</i>	the Point4D measurement scale.

4.20.4.17 caerPoint4DEventSetTimestamp()

```
static void caerPoint4DEventSetTimestamp (
    caerPoint4DEvent event,
    int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.20.4.18 caerPoint4DEventSetType()

```
static void caerPoint4DEventSetType (
    caerPoint4DEvent event,
    uint8_t type ) [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
<i>type</i>	the Point4D measurement type.

4.20.4.19 caerPoint4DEventSetW()

```
static void caerPoint4DEventSetW (
    caerPoint4DEvent event,
    float w ) [inline], [static]
```

Set the W axis measurement.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
<i>w</i>	W axis measurement.

4.20.4.20 caerPoint4DEventSetX()

```
static void caerPoint4DEventSetX (
```

```
    caerPoint4DEvent event,  
    float x )    [inline], [static]
```

Set the X axis measurement.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
<i>x</i>	X axis measurement.

4.20.4.21 caerPoint4DEventSetY()

```
static void caerPoint4DEventSetY (  
    caerPoint4DEvent event,  
    float y )    [inline], [static]
```

Set the Y axis measurement.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
<i>y</i>	Y axis measurement.

4.20.4.22 caerPoint4DEventSetZ()

```
static void caerPoint4DEventSetZ (  
    caerPoint4DEvent event,  
    float z )    [inline], [static]
```

Set the Z axis measurement.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
<i>z</i>	Z axis measurement.

4.20.4.23 caerPoint4DEventValidate()

```
static void caerPoint4DEventValidate (  
    caerPoint4DEvent event,  
    caerPoint4DEventPacket packet )    [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid Point4DEvent pointer. Cannot be NULL.
<i>packet</i>	the Point4DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.20.4.24 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
    struct caer_point4d_event { uint32_t info;float x;float y;float z;float w;int32_t
timestamp;} )
```

Point4D event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The four measurements (x, y, z, w) are stored as floats. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.20.4.25 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
    struct caer_point4d_event_packet { struct caer_event_packet_header packetHeader;struct
caer_point4d_event events[];} )
```

Point4D event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.21 events/polarity.h File Reference

```
#include "common.h"
```

Macros

- `#define CAER_POLARITY_ITERATOR_ALL_START(POLARITY_PACKET)`
 - `#define CAER_POLARITY_CONST_ITERATOR_ALL_START(POLARITY_PACKET)`
 - `#define CAER_POLARITY_ITERATOR_ALL_END }`
 - `#define CAER_POLARITY_ITERATOR_VALID_START(POLARITY_PACKET)`
 - `#define CAER_POLARITY_CONST_ITERATOR_VALID_START(POLARITY_PACKET)`
 - `#define CAER_POLARITY_ITERATOR_VALID_END }`
 - `#define CAER_POLARITY_REVERSE_ITERATOR_ALL_START(POLARITY_PACKET)`
 - `#define CAER_POLARITY_CONST_REVERSE_ITERATOR_ALL_START(POLARITY_PACKET)`
 - `#define CAER_POLARITY_REVERSE_ITERATOR_ALL_END }`
 - `#define CAER_POLARITY_REVERSE_ITERATOR_VALID_START(POLARITY_PACKET)`
 - `#define CAER_POLARITY_CONST_REVERSE_ITERATOR_VALID_START(POLARITY_PACKET)`
 - `#define CAER_POLARITY_REVERSE_ITERATOR_VALID_END }`
-
- `#define POLARITY_SHIFT 1`
 - `#define POLARITY_MASK 0x00000001`
 - `#define POLARITY_Y_ADDR_SHIFT 2`
 - `#define POLARITY_Y_ADDR_MASK 0x00007FFF`
 - `#define POLARITY_X_ADDR_SHIFT 17`
 - `#define POLARITY_X_ADDR_MASK 0x00007FFF`

Typedefs

- typedef struct caer_polarity_event * [caerPolarityEvent](#)
- typedef const struct caer_polarity_event * **caerPolarityEventConst**
- typedef struct caer_polarity_event_packet * [caerPolarityEventPacket](#)
- typedef const struct caer_polarity_event_packet * **caerPolarityEventPacketConst**

Functions

- [PACKED_STRUCT](#) (struct caer_polarity_event { uint32_t data;int32_t timestamp;})
- [PACKED_STRUCT](#) (struct caer_polarity_event_packet { struct caer_event_packet_header packet↔Header;struct caer_polarity_event events[];})
- static [caerPolarityEventPacket](#) [caerPolarityEventPacketAllocate](#) (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static [caerPolarityEventPacket](#) [caerPolarityEventPacketFromPacketHeader](#) ([caerEventPacketHeader](#) header)
- static [caerPolarityEventPacketConst](#) [caerPolarityEventPacketFromPacketHeaderConst](#) ([caerEventPacketHeaderConst](#) header)
- static [caerPolarityEvent](#) [caerPolarityEventPacketGetEvent](#) ([caerPolarityEventPacket](#) packet, int32_t n)
- static [caerPolarityEventConst](#) [caerPolarityEventPacketGetEventConst](#) ([caerPolarityEventPacketConst](#) packet, int32_t n)
- static int32_t [caerPolarityEventGetTimestamp](#) ([caerPolarityEventConst](#) event)
- static int64_t [caerPolarityEventGetTimestamp64](#) ([caerPolarityEventConst](#) event, [caerPolarityEventPacketConst](#) packet)
- static void [caerPolarityEventSetTimestamp](#) ([caerPolarityEvent](#) event, int32_t timestamp)
- static bool [caerPolarityEventsIsValid](#) ([caerPolarityEventConst](#) event)
- static void [caerPolarityEventValidate](#) ([caerPolarityEvent](#) event, [caerPolarityEventPacket](#) packet)
- static void [caerPolarityEventInvalidate](#) ([caerPolarityEvent](#) event, [caerPolarityEventPacket](#) packet)
- static bool [caerPolarityEventGetPolarity](#) ([caerPolarityEventConst](#) event)
- static void [caerPolarityEventSetPolarity](#) ([caerPolarityEvent](#) event, bool polarity)
- static uint16_t [caerPolarityEventGetY](#) ([caerPolarityEventConst](#) event)
- static void [caerPolarityEventSetY](#) ([caerPolarityEvent](#) event, uint16_t yAddress)
- static uint16_t [caerPolarityEventGetX](#) ([caerPolarityEventConst](#) event)
- static void [caerPolarityEventSetX](#) ([caerPolarityEvent](#) event, uint16_t xAddress)

4.21.1 Detailed Description

Polarity Events format definition and handling functions. This event contains change information, with an X/Y address and an ON/OFF polarity. The (0, 0) address is in the upper left corner of the screen, like in OpenCV/computer graphics.

4.21.2 Macro Definition Documentation

4.21.2.1 CAER_POLARITY_CONST_ITERATOR_ALL_START

```
#define CAER_POLARITY_CONST_ITERATOR_ALL_START(
    POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter = 0;
     caerPolarityIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(POLARITY_PACKET)->packetHeader); \
     caerPolarityIteratorCounter++) {
    \
    caerPolarityEventConst caerPolarityIteratorElement
    \
    = caerPolarityEventPacketGetEventConst (POLARITY_PACKET,
    caerPolarityIteratorCounter);
```

Const-Iterator over all polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPolarityIteratorElement' variable of type caerPolarityEventConst.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.2 CAER_POLARITY_CONST_ITERATOR_VALID_START

```
#define CAER_POLARITY_CONST_ITERATOR_VALID_START(
    POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter = 0;
     caerPolarityIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(POLARITY_PACKET)->packetHeader); \
     caerPolarityIteratorCounter++) {
    \
    caerPolarityEventConst caerPolarityIteratorElement
    \
    = caerPolarityEventPacketGetEventConst (POLARITY_PACKET,
    caerPolarityIteratorCounter);
    if (!caerPolarityEventIsValid(caerPolarityIteratorElement)) {
        \
        continue;
    }
    \
}
```

Const-Iterator over only the valid polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPolarityIteratorElement' variable of type caerPolarityEventConst.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.3 CAER_POLARITY_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_POLARITY_CONST_REVERSE_ITERATOR_ALL_START(
    POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter
     = caerEventPacketHeaderGetEventNumber (&(POLARITY_PACKET)->
     packetHeader) - 1; \
     caerPolarityIteratorCounter >= 0; caerPolarityIteratorCounter--) {
    caerPolarityEventConst caerPolarityIteratorElement
    \
    = caerPolarityEventPacketGetEventConst (POLARITY_PACKET,
    caerPolarityIteratorCounter);
```

Const-Reverse iterator over all polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPolarityIteratorElement' variable of type caerPolarityEventConst.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.4 CAER_POLARITY_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_POLARITY_CONST_REVERSE_ITERATOR_VALID_START (
    POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter
    = caerEventPacketHeaderGetEventNumber (& (POLARITY_PACKET)->
    packetHeader) - 1;
    caerPolarityIteratorCounter >= 0; caerPolarityIteratorCounter--) {
    caerPolarityEventConst caerPolarityIteratorElement
    = caerPolarityEventPacketGetEventConst (POLARITY_PACKET,
    caerPolarityIteratorCounter); \
    if (!caerPolarityEventIsValid(caerPolarityIteratorElement)) {
        continue; \
    }
```

Const-Reverse iterator over only the valid polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPolarityIteratorElement' variable of type caerPolarityEventConst.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.5 CAER_POLARITY_ITERATOR_ALL_END

```
#define CAER_POLARITY_ITERATOR_ALL_END }
```

Iterator close statement.

4.21.2.6 CAER_POLARITY_ITERATOR_ALL_START

```
#define CAER_POLARITY_ITERATOR_ALL_START (
    POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter = 0;
    caerPolarityIteratorCounter < caerEventPacketHeaderGetEventNumber
    (& (POLARITY_PACKET)->packetHeader); \
    caerPolarityIteratorCounter++) {
    caerPolarityEvent caerPolarityIteratorElement
    = caerPolarityEventPacketGetEvent (POLARITY_PACKET,
    caerPolarityIteratorCounter);
```

Iterator over all polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPolarityIteratorElement' variable of type caerPolarityEvent.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.7 CAER_POLARITY_ITERATOR_VALID_END

```
#define CAER_POLARITY_ITERATOR_VALID_END }
```

Iterator close statement.

4.21.2.8 CAER_POLARITY_ITERATOR_VALID_START

```
#define CAER_POLARITY_ITERATOR_VALID_START(
    POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter = 0;
     caerPolarityIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(POLARITY_PACKET)->packetHeader); \
     caerPolarityIteratorCounter++) {
    \
    caerPolarityEvent caerPolarityIteratorElement
    \
    = caerPolarityEventPacketGetEvent(POLARITY_PACKET,
    caerPolarityIteratorCounter);
    if (!caerPolarityEventIsValid(caerPolarityIteratorElement)) {
        \
        continue;
    }
    \
}
```

Iterator over only the valid polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPolarityIteratorElement' variable of type caerPolarityEvent.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.9 CAER_POLARITY_REVERSE_ITERATOR_ALL_END

```
#define CAER_POLARITY_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.21.2.10 CAER_POLARITY_REVERSE_ITERATOR_ALL_START

```
#define CAER_POLARITY_REVERSE_ITERATOR_ALL_START(
    POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter
     = caerEventPacketHeaderGetEventNumber (&(POLARITY_PACKET)->
     packetHeader) - 1; \
     caerPolarityIteratorCounter >= 0; caerPolarityIteratorCounter--) {
    caerPolarityEvent caerPolarityIteratorElement
    \
    = caerPolarityEventPacketGetEvent(POLARITY_PACKET,
    caerPolarityIteratorCounter);
    \
}
```

Reverse iterator over all polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPolarityIteratorElement' variable of type caerPolarityEvent.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.11 CAER_POLARITY_REVERSE_ITERATOR_VALID_END

```
#define CAER_POLARITY_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.21.2.12 CAER_POLARITY_REVERSE_ITERATOR_VALID_START

```
#define CAER_POLARITY_REVERSE_ITERATOR_VALID_START(
    POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter           \
     = caerEventPacketHeaderGetEventNumber (&(POLARITY_PACKET)-> \
     packetHeader) - 1; \
     caerPolarityIteratorCounter >= 0; caerPolarityIteratorCounter--) { \
    caerPolarityEvent caerPolarityIteratorElement \
    = caerPolarityEventPacketGetEvent (POLARITY_PACKET, \
    caerPolarityIteratorCounter); \
    if (!caerPolarityEventIsValid(caerPolarityIteratorElement)) { \
        \
        continue; \
    } \
}
```

Reverse iterator over only the valid polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPolarityIteratorElement' variable of type caerPolarityEvent.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.13 POLARITY_MASK

```
#define POLARITY_MASK 0x00000001
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 15 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see [common.h](#) for more details.

4.21.2.14 POLARITY_SHIFT

```
#define POLARITY_SHIFT 1
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 15 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see [common.h](#) for more details.

4.21.2.15 POLARITY_X_ADDR_MASK

```
#define POLARITY_X_ADDR_MASK 0x00007FFF
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 15 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see [common.h](#) for more details.

4.21.2.16 POLARITY_X_ADDR_SHIFT

```
#define POLARITY_X_ADDR_SHIFT 17
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 15 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see [common.h](#) for more details.

4.21.2.17 POLARITY_Y_ADDR_MASK

```
#define POLARITY_Y_ADDR_MASK 0x00007FFF
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 15 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.21.2.18 POLARITY_Y_ADDR_SHIFT

```
#define POLARITY_Y_ADDR_SHIFT 2
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 15 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.21.3 Typedef Documentation

4.21.3.1 caerPolarityEvent

```
typedef struct caer_polarity_event* caerPolarityEvent
```

Type for pointer to polarity event data structure.

4.21.3.2 caerPolarityEventPacket

```
typedef struct caer_polarity_event_packet* caerPolarityEventPacket
```

Type for pointer to polarity event packet data structure.

4.21.4 Function Documentation

4.21.4.1 caerPolarityEventGetPolarity()

```
static bool caerPolarityEventGetPolarity (  
    caerPolarityEventConst event ) [inline], [static]
```

Get the change event polarity. 1 is ON, 0 is OFF.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
--------------	--

Returns

event polarity value.

4.21.4.2 caerPolarityEventGetTimestamp()

```
static int32_t caerPolarityEventGetTimestamp (
    caerPolarityEventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
--------------	--

Returns

this event's 32bit microsecond timestamp.

4.21.4.3 caerPolarityEventGetTimestamp64()

```
static int64_t caerPolarityEventGetTimestamp64 (
    caerPolarityEventConst event,
    caerPolarityEventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
<i>packet</i>	the PolarityEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.21.4.4 caerPolarityEventGetX()

```
static uint16_t caerPolarityEventGetX (
    caerPolarityEventConst event ) [inline], [static]
```


Get the X (column) address for a change event, in pixels. The (0, 0) address is in the upper left corner, like in OpenCV/computer graphics.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
--------------	--

Returns

the event X address.

4.21.4.5 caerPolarityEventGetY()

```
static uint16_t caerPolarityEventGetY (  
    caerPolarityEventConst event ) [inline], [static]
```

Get the Y (row) address for a change event, in pixels. The (0, 0) address is in the upper left corner, like in OpenCV/computer graphics.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
--------------	--

Returns

the event Y address.

4.21.4.6 caerPolarityEventInvalidate()

```
static void caerPolarityEventInvalidate (  
    caerPolarityEvent event,  
    caerPolarityEventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
<i>packet</i>	the PolarityEventPacket pointer for the packet containing this event. Cannot be NULL.

4.21.4.7 caerPolarityEventIsValid()

```
static bool caerPolarityEventIsValid (
    caerPolarityEventConst event ) [inline], [static]
```

Check if this polarity event is valid.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
--------------	--

Returns

true if valid, false if not.

4.21.4.8 caerPolarityEventPacketAllocate()

```
static caerPolarityEventPacket caerPolarityEventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]
```

Allocate a new polarity events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid PolarityEventPacket handle or NULL on error.

4.21.4.9 caerPolarityEventPacketFromPacketHeader()

```
static caerPolarityEventPacket caerPolarityEventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```

Transform a generic event packet header into a Polarity event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.21.4.10 caerPolarityEventPacketFromPacketHeaderConst()

```
static caerPolarityEventPacketConst caerPolarityEventPacketFromPacketHeaderConst (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only Polarity event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.21.4.11 caerPolarityEventPacketGetEvent()

```
static caerPolarityEvent caerPolarityEventPacketGetEvent (
    caerPolarityEventPacket packet,
    int32_t n ) [inline], [static]
```

Get the polarity event at the given index from the event packet.

Parameters

<i>packet</i>	a valid PolarityEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested polarity event. NULL on error.

4.21.4.12 caerPolarityEventPacketGetEventConst()

```
static caerPolarityEventConst caerPolarityEventPacketGetEventConst (
    caerPolarityEventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the polarity event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid PolarityEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only polarity event. NULL on error.

4.21.4.13 caerPolarityEventSetPolarity()

```
static void caerPolarityEventSetPolarity (
    caerPolarityEvent event,
    bool polarity ) [inline], [static]
```

Set the change event polarity. 1 is ON, 0 is OFF.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
<i>polarity</i>	event polarity value.

4.21.4.14 caerPolarityEventSetTimestamp()

```
static void caerPolarityEventSetTimestamp (
    caerPolarityEvent event,
    int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.21.4.15 caerPolarityEventSetX()

```
static void caerPolarityEventSetX (
    caerPolarityEvent event,
    uint16_t xAddress ) [inline], [static]
```

Set the X (column) address for a change event, in pixels. The (0, 0) address is in the upper left corner, like in OpenCV/computer graphics.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
<i>xAddress</i>	the event X address.

4.21.4.16 caerPolarityEventSetY()

```
static void caerPolarityEventSetY (
    caerPolarityEvent event,
    uint16_t yAddress ) [inline], [static]
```

Set the Y (row) address for a change event, in pixels. The (0, 0) address is in the upper left corner, like in OpenCV/computer graphics.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
<i>yAddress</i>	the event Y address.

4.21.4.17 caerPolarityEventValidate()

```
static void caerPolarityEventValidate (
    caerPolarityEvent event,
    caerPolarityEventPacket packet ) [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
<i>packet</i>	the PolarityEventPacket pointer for the packet containing this event. Cannot be NULL.

4.21.4.18 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
    struct caer_polarity_event { uint32_t data;int32_t timestamp;} )
```

Polarity event data structure definition. This contains the actual X/Y addresses, the polarity, as well as the 32 bit event timestamp. The (0, 0) address is in the upper left corner of the screen, like in OpenCV/computer graphics. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.21.4.19 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
    struct caer_polarity_event_packet { struct caer_event_packet_header packet↔
Header; struct caer_polarity_event events[]; } )
```

Polarity event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.22 events/sample.h File Reference

```
#include "common.h"
```

Macros

- #define [CAER_SAMPLE_ITERATOR_ALL_START](#)(SAMPLE_PACKET)
 - #define [CAER_SAMPLE_CONST_ITERATOR_ALL_START](#)(SAMPLE_PACKET)
 - #define [CAER_SAMPLE_ITERATOR_ALL_END](#) }
 - #define [CAER_SAMPLE_ITERATOR_VALID_START](#)(SAMPLE_PACKET)
 - #define [CAER_SAMPLE_CONST_ITERATOR_VALID_START](#)(SAMPLE_PACKET)
 - #define [CAER_SAMPLE_ITERATOR_VALID_END](#) }
 - #define [CAER_SAMPLE_REVERSE_ITERATOR_ALL_START](#)(SAMPLE_PACKET)
 - #define [CAER_SAMPLE_CONST_REVERSE_ITERATOR_ALL_START](#)(SAMPLE_PACKET)
 - #define [CAER_SAMPLE_REVERSE_ITERATOR_ALL_END](#) }
 - #define [CAER_SAMPLE_REVERSE_ITERATOR_VALID_START](#)(SAMPLE_PACKET)
 - #define [CAER_SAMPLE_CONST_REVERSE_ITERATOR_VALID_START](#)(SAMPLE_PACKET)
 - #define [CAER_SAMPLE_REVERSE_ITERATOR_VALID_END](#) }
-
- #define [SAMPLE_TYPE_SHIFT](#) 1
 - #define [SAMPLE_TYPE_MASK](#) 0x0000007F
 - #define [SAMPLE_SHIFT](#) 8
 - #define [SAMPLE_MASK](#) 0x00FFFFFF

Typedefs

- typedef struct caer_sample_event * [caerSampleEvent](#)
- typedef const struct caer_sample_event * **caerSampleEventConst**
- typedef struct caer_sample_event_packet * [caerSampleEventPacket](#)
- typedef const struct caer_sample_event_packet * **caerSampleEventPacketConst**

Functions

- [PACKED_STRUCT](#) (struct caer_sample_event { uint32_t data;int32_t timestamp;})
- [PACKED_STRUCT](#) (struct caer_sample_event_packet { struct caer_event_packet_header packet↵ Header;struct caer_sample_event events[];})
- static [caerSampleEventPacket](#) [caerSampleEventPacketAllocate](#) (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static [caerSampleEventPacket](#) [caerSampleEventPacketFromPacketHeader](#) ([caerEventPacketHeader](#) header)
- static [caerSampleEventPacketConst](#) [caerSampleEventPacketFromPacketHeaderConst](#) ([caerEventPacketHeaderConst](#) header)
- static [caerSampleEvent](#) [caerSampleEventPacketGetEvent](#) ([caerSampleEventPacket](#) packet, int32_t n)
- static [caerSampleEventConst](#) [caerSampleEventPacketGetEventConst](#) ([caerSampleEventPacketConst](#) packet, int32_t n)
- static int32_t [caerSampleEventGetTimestamp](#) ([caerSampleEventConst](#) event)
- static int64_t [caerSampleEventGetTimestamp64](#) ([caerSampleEventConst](#) event, [caerSampleEventPacketConst](#)↵ packet)
- static void [caerSampleEventSetTimestamp](#) ([caerSampleEvent](#) event, int32_t timestamp)
- static bool [caerSampleEventsIsValid](#) ([caerSampleEventConst](#) event)
- static void [caerSampleEventValidate](#) ([caerSampleEvent](#) event, [caerSampleEventPacket](#) packet)
- static void [caerSampleEventInvalidate](#) ([caerSampleEvent](#) event, [caerSampleEventPacket](#) packet)
- static uint8_t [caerSampleEventGetType](#) ([caerSampleEventConst](#) event)
- static void [caerSampleEventSetType](#) ([caerSampleEvent](#) event, uint8_t type)
- static uint32_t [caerSampleEventGetSample](#) ([caerSampleEventConst](#) event)
- static void [caerSampleEventSetSample](#) ([caerSampleEvent](#) event, uint32_t sample)

4.22.1 Detailed Description

Sample (ADC) Events format definition and handling functions. Represents different types of ADC readings, up to 24 bits of resolution.

4.22.2 Macro Definition Documentation

4.22.2.1 CAER_SAMPLE_CONST_ITERATOR_ALL_START

```
#define CAER_SAMPLE_CONST_ITERATOR_ALL_START(
    SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = 0;
     caerSampleIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(SAMPLE_PACKET)->packetHeader); \
     caerSampleIteratorCounter++) {
    caerSampleEventConst caerSampleIteratorElement
    = caerSampleEventPacketGetEventConst (SAMPLE_PACKET,
    caerSampleIteratorCounter);
```

Const-iterator over all sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSampleIteratorElement' variable of type caer↵ SampleEventConst.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.2 CAER_SAMPLE_CONST_ITERATOR_VALID_START

```
#define CAER_SAMPLE_CONST_ITERATOR_VALID_START(  
    SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = 0; \
     caerSampleIteratorCounter < caerEventPacketHeaderGetEventNumber \
     (&(SAMPLE_PACKET)->packetHeader); \
     caerSampleIteratorCounter++) { \
    caerSampleEventConst caerSampleIteratorElement \
    = caerSampleEventPacketGetEventConst(SAMPLE_PACKET, \
    caerSampleIteratorCounter); \
    if (!caerSampleEventIsValid(caerSampleIteratorElement)) { \
        continue; \
    }
```

Const-Iterator over only the valid sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSampleIteratorElement' variable of type caerSampleEventConst.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.3 CAER_SAMPLE_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_SAMPLE_CONST_REVERSE_ITERATOR_ALL_START(  
    SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = caerEventPacketHeaderGetEventNumber \
     (&(SAMPLE_PACKET)->packetHeader) - 1; \
     caerSampleIteratorCounter >= 0; caerSampleIteratorCounter--) { \
    caerSampleEventConst caerSampleIteratorElement \
    = caerSampleEventPacketGetEventConst(SAMPLE_PACKET, \
    caerSampleIteratorCounter);
```

Const-Reverse iterator over all sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSampleIteratorElement' variable of type caerSampleEventConst.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.4 CAER_SAMPLE_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_SAMPLE_CONST_REVERSE_ITERATOR_VALID_START(  
    SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = caerEventPacketHeaderGetEventNumber \
     (&(SAMPLE_PACKET)->packetHeader) - 1; \
     caerSampleIteratorCounter >= 0; caerSampleIteratorCounter--) { \
    caerSampleEventConst caerSampleIteratorElement \
    = caerSampleEventPacketGetEventConst(SAMPLE_PACKET, \
    caerSampleIteratorCounter); \
    if (!caerSampleEventIsValid(caerSampleIteratorElement)) { \
        continue; \
    }
```

Const-Reverse iterator over only the valid sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSampleIteratorElement' variable of type caerSampleEventConst.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.5 CAER_SAMPLE_ITERATOR_ALL_END

```
#define CAER_SAMPLE_ITERATOR_ALL_END }
```

Iterator close statement.

4.22.2.6 CAER_SAMPLE_ITERATOR_ALL_START

```
#define CAER_SAMPLE_ITERATOR_ALL_START(  
    SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = 0;                               \  
    caerSampleIteratorCounter < caerEventPacketHeaderGetEventNumber    \  
    (&(SAMPLE_PACKET)->packetHeader); \                                \  
    caerSampleIteratorCounter++) {                                       \  
    caerSampleEvent caerSampleIteratorElement                            \  
        = caerSampleEventPacketGetEvent(SAMPLE_PACKET,                  \  
    caerSampleIteratorCounter);
```

Iterator over all sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSampleIteratorElement' variable of type caerSampleEvent.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.7 CAER_SAMPLE_ITERATOR_VALID_END

```
#define CAER_SAMPLE_ITERATOR_VALID_END }
```

Iterator close statement.

4.22.2.8 CAER_SAMPLE_ITERATOR_VALID_START

```
#define CAER_SAMPLE_ITERATOR_VALID_START(  
    SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = 0;                               \  
    caerSampleIteratorCounter < caerEventPacketHeaderGetEventNumber    \  
    (&(SAMPLE_PACKET)->packetHeader); \                                \  
    caerSampleIteratorCounter++) {                                       \  
    caerSampleEvent caerSampleIteratorElement                            \  
        = caerSampleEventPacketGetEvent(SAMPLE_PACKET,                  \  
    caerSampleIteratorCounter); \                                       \  
    if (!caerSampleEventIsValid(caerSampleIteratorElement)) {           \  
        continue; \                                                     \  
    }
```

Iterator over only the valid sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSampleIteratorElement' variable of type caerSampleEvent.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.9 CAER_SAMPLE_REVERSE_ITERATOR_ALL_END

```
#define CAER_SAMPLE_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.22.2.10 CAER_SAMPLE_REVERSE_ITERATOR_ALL_START

```
#define CAER_SAMPLE_REVERSE_ITERATOR_ALL_START(  
    SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(SAMPLE_PACKET)->packetHeader) - 1; \  
    caerSampleIteratorCounter >= 0; caerSampleIteratorCounter--) {  
    caerSampleEvent caerSampleIteratorElement  
        = caerSampleEventPacketGetEvent (SAMPLE_PACKET,  
        caerSampleIteratorCounter);
```

Reverse iterator over all sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSampleIteratorElement' variable of type caerSampleEvent.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.11 CAER_SAMPLE_REVERSE_ITERATOR_VALID_END

```
#define CAER_SAMPLE_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.22.2.12 CAER_SAMPLE_REVERSE_ITERATOR_VALID_START

```
#define CAER_SAMPLE_REVERSE_ITERATOR_VALID_START(  
    SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(SAMPLE_PACKET)->packetHeader) - 1; \  
    caerSampleIteratorCounter >= 0; caerSampleIteratorCounter--) {  
    caerSampleEvent caerSampleIteratorElement  
        = caerSampleEventPacketGetEvent (SAMPLE_PACKET,  
        caerSampleIteratorCounter);  
    if (!caerSampleEventIsValid(caerSampleIteratorElement)) {  
        continue;  
    }  
}
```

Reverse iterator over only the valid sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSampleIteratorElement' variable of type caerSampleEvent.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.13 SAMPLE_MASK

```
#define SAMPLE_MASK 0x00FFFFFF
```

Shift and mask values for the sample type and the actual sample value of an ADC sample. Up to 128 sample types are supported, with 24 bits of data per sample. Higher values mean a higher voltage, 0 is ground. Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.22.2.14 SAMPLE_SHIFT

```
#define SAMPLE_SHIFT 8
```

Shift and mask values for the sample type and the actual sample value of an ADC sample. Up to 128 sample types are supported, with 24 bits of data per sample. Higher values mean a higher voltage, 0 is ground. Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.22.2.15 SAMPLE_TYPE_MASK

```
#define SAMPLE_TYPE_MASK 0x0000007F
```

Shift and mask values for the sample type and the actual sample value of an ADC sample. Up to 128 sample types are supported, with 24 bits of data per sample. Higher values mean a higher voltage, 0 is ground. Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.22.2.16 SAMPLE_TYPE_SHIFT

```
#define SAMPLE_TYPE_SHIFT 1
```

Shift and mask values for the sample type and the actual sample value of an ADC sample. Up to 128 sample types are supported, with 24 bits of data per sample. Higher values mean a higher voltage, 0 is ground. Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.22.3 Typedef Documentation

4.22.3.1 caerSampleEvent

```
typedef struct caer_sample_event* caerSampleEvent
```

Type for pointer to ADC sample event data structure.

4.22.3.2 caerSampleEventPacket

```
typedef struct caer_sample_event_packet* caerSampleEventPacket
```

Type for pointer to ADC sample event packet data structure.

4.22.4 Function Documentation

4.22.4.1 caerSampleEventGetSample()

```
static uint32_t caerSampleEventGetSample (
    caerSampleEventConst event ) [inline], [static]
```

Get the ADC sample value. Up to 24 bits of resolution are possible. Higher values mean a higher voltage, 0 is ground.

Parameters

<i>event</i>	a valid SampleEvent pointer. Cannot be NULL.
--------------	--

Returns

the ADC sample value.

4.22.4.2 caerSampleEventGetTimestamp()

```
static int32_t caerSampleEventGetTimestamp (
    caerSampleEventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid SampleEvent pointer. Cannot be NULL.
--------------	--

Returns

this event's 32bit microsecond timestamp.

4.22.4.3 caerSampleEventGetTimestamp64()

```
static int64_t caerSampleEventGetTimestamp64 (
    caerSampleEventConst event,
    caerSampleEventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid SampleEvent pointer. Cannot be NULL.
<i>packet</i>	the SampleEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.22.4.4 caerSampleEventGetType()

```
static uint8_t caerSampleEventGetType (  
    caerSampleEventConst event ) [inline], [static]
```

Get the ADC sample event type. This is useful to distinguish between different measurements, for example from two separate microphones on a device.

Parameters

<i>event</i>	a valid SampleEvent pointer. Cannot be NULL.
--------------	--

Returns

the ADC sample type.

4.22.4.5 caerSampleEventInvalidate()

```
static void caerSampleEventInvalidate (  
    caerSampleEvent event,  
    caerSampleEventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid SampleEvent pointer. Cannot be NULL.
<i>packet</i>	the SampleEventPacket pointer for the packet containing this event. Cannot be NULL.

4.22.4.6 caerSampleEventIsValid()

```
static bool caerSampleEventIsValid (  

```

```
caerSampleEventConst event ) [inline], [static]
```

Check if this ADC sample event is valid.

Parameters

<i>event</i>	a valid SampleEvent pointer. Cannot be NULL.
--------------	--

Returns

true if valid, false if not.

4.22.4.7 caerSampleEventPacketAllocate()

```
static caerSampleEventPacket caerSampleEventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]
```

Allocate a new ADC sample events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid SampleEventPacket handle or NULL on error.

4.22.4.8 caerSampleEventPacketFromPacketHeader()

```
static caerSampleEventPacket caerSampleEventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```

Transform a generic event packet header into a Sample event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.22.4.9 caerSampleEventPacketFromPacketHeaderConst()

```
static caerSampleEventPacketConst caerSampleEventPacketFromPacketHeaderConst (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only Sample event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.22.4.10 caerSampleEventPacketGetEvent()

```
static caerSampleEvent caerSampleEventPacketGetEvent (
    caerSampleEventPacket packet,
    int32_t n ) [inline], [static]
```

Get the ADC sample event at the given index from the event packet.

Parameters

<i>packet</i>	a valid SampleEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested ADC sample event. NULL on error.

4.22.4.11 caerSampleEventPacketGetEventConst()

```
static caerSampleEventConst caerSampleEventPacketGetEventConst (
    caerSampleEventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the ADC sample event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid SampleEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only ADC sample event. NULL on error.

4.22.4.12 caerSampleEventSetSample()

```
static void caerSampleEventSetSample (
    caerSampleEvent event,
    uint32_t sample ) [inline], [static]
```

Set the ADC sample value. Up to 24 bits of resolution are possible. Higher values mean a higher voltage, 0 is ground.

Parameters

<i>event</i>	a valid SampleEvent pointer. Cannot be NULL.
<i>sample</i>	the ADC sample value.

4.22.4.13 caerSampleEventSetTimestamp()

```
static void caerSampleEventSetTimestamp (
    caerSampleEvent event,
    int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid SampleEvent pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.22.4.14 caerSampleEventSetType()

```
static void caerSampleEventSetType (
    caerSampleEvent event,
    uint8_t type ) [inline], [static]
```


Set the ADC sample event type. This is useful to distinguish between different measurements, for example from two separate microphones on a device.

Parameters

<i>event</i>	a valid SampleEvent pointer. Cannot be NULL.
<i>type</i>	the ADC sample type.

4.22.4.15 caerSampleEventValidate()

```
static void caerSampleEventValidate (
    caerSampleEvent event,
    caerSampleEventPacket packet ) [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid SampleEvent pointer. Cannot be NULL.
<i>packet</i>	the SampleEventPacket pointer for the packet containing this event. Cannot be NULL.

4.22.4.16 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
    struct caer_sample_event { uint32_t data;int32_t timestamp;} )
```

ADC sample event data structure definition. Contains a type indication to separate different ADC readouts, as well as a value for that readout, up to 24 bits resolution. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.22.4.17 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
    struct caer_sample_event_packet { struct caer_event_packet_header packetHeader;struct
    caer_sample_event events[];} )
```

ADC sample event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.23 events/special.h File Reference

```
#include "common.h"
```

Macros

- `#define CAER_SPECIAL_ITERATOR_ALL_START(SPECIAL_PACKET)`
 - `#define CAER_SPECIAL_CONST_ITERATOR_ALL_START(SPECIAL_PACKET)`
 - `#define CAER_SPECIAL_ITERATOR_ALL_END }`
 - `#define CAER_SPECIAL_ITERATOR_VALID_START(SPECIAL_PACKET)`
 - `#define CAER_SPECIAL_CONST_ITERATOR_VALID_START(SPECIAL_PACKET)`
 - `#define CAER_SPECIAL_ITERATOR_VALID_END }`
 - `#define CAER_SPECIAL_REVERSE_ITERATOR_ALL_START(SPECIAL_PACKET)`
 - `#define CAER_SPECIAL_CONST_REVERSE_ITERATOR_ALL_START(SPECIAL_PACKET)`
 - `#define CAER_SPECIAL_REVERSE_ITERATOR_ALL_END }`
 - `#define CAER_SPECIAL_REVERSE_ITERATOR_VALID_START(SPECIAL_PACKET)`
 - `#define CAER_SPECIAL_CONST_REVERSE_ITERATOR_VALID_START(SPECIAL_PACKET)`
 - `#define CAER_SPECIAL_REVERSE_ITERATOR_VALID_END }`
-
- `#define SPECIAL_TYPE_SHIFT 1`
 - `#define SPECIAL_TYPE_MASK 0x0000007F`
 - `#define SPECIAL_DATA_SHIFT 8`
 - `#define SPECIAL_DATA_MASK 0x00FFFFFF`

Typedefs

- `typedef struct caer_special_event * caerSpecialEvent`
- `typedef const struct caer_special_event * caerSpecialEventConst`
- `typedef struct caer_special_event_packet * caerSpecialEventPacket`
- `typedef const struct caer_special_event_packet * caerSpecialEventPacketConst`

Enumerations

- `enum caer_special_event_types {`
`TIMESTAMP_WRAP = 0, TIMESTAMP_RESET = 1, EXTERNAL_INPUT_RISING_EDGE = 2, EXTERNAL_INPUT_FALLING_EDGE = 3,`
`EXTERNAL_INPUT_PULSE = 4, DVS_ROW_ONLY = 5, EXTERNAL_INPUT1_RISING_EDGE = 6, EXTERNAL_INPUT1_FALLING_EDGE = 7,`
`EXTERNAL_INPUT1_PULSE = 8, EXTERNAL_INPUT2_RISING_EDGE = 9, EXTERNAL_INPUT2_FALLING_EDGE = 10, EXTERNAL_INPUT2_PULSE = 11,`
`EXTERNAL_GENERATOR_RISING_EDGE = 12, EXTERNAL_GENERATOR_FALLING_EDGE = 13, APS_FRAME_START = 14, APS_FRAME_END = 15,`
`APS_EXPOSURE_START = 16, APS_EXPOSURE_END = 17 }`

Functions

- [PACKED_STRUCT](#) (struct caer_special_event { uint32_t data;int32_t timestamp;})
- [PACKED_STRUCT](#) (struct caer_special_event_packet { struct caer_event_packet_header packet↵ Header;struct caer_special_event events[];})
- static [caerSpecialEventPacket caerSpecialEventPacketAllocate](#) (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static [caerSpecialEventPacket caerSpecialEventPacketFromPacketHeader](#) (caerEventPacketHeader header)
- static caerSpecialEventPacketConst [caerSpecialEventPacketFromPacketHeaderConst](#) (caerEventPacket↵ HeaderConst header)
- static [caerSpecialEvent caerSpecialEventPacketGetEvent](#) (caerSpecialEventPacket packet, int32_t n)
- static caerSpecialEventConst [caerSpecialEventPacketGetEventConst](#) (caerSpecialEventPacketConst packet, int32_t n)
- static int32_t [caerSpecialEventGetTimestamp](#) (caerSpecialEventConst event)
- static int64_t [caerSpecialEventGetTimestamp64](#) (caerSpecialEventConst event, caerSpecialEventPacket↵ Const packet)
- static void [caerSpecialEventSetTimestamp](#) (caerSpecialEvent event, int32_t timestamp)
- static bool [caerSpecialEventsIsValid](#) (caerSpecialEventConst event)
- static void [caerSpecialEventValidate](#) (caerSpecialEvent event, caerSpecialEventPacket packet)
- static void [caerSpecialEventInvalidate](#) (caerSpecialEvent event, caerSpecialEventPacket packet)
- static uint8_t [caerSpecialEventGetType](#) (caerSpecialEventConst event)
- static void [caerSpecialEventSetType](#) (caerSpecialEvent event, uint8_t type)
- static uint32_t [caerSpecialEventGetData](#) (caerSpecialEventConst event)
- static void [caerSpecialEventSetData](#) (caerSpecialEvent event, uint32_t data)
- static [caerSpecialEvent caerSpecialEventPacketFindEventByType](#) (caerSpecialEventPacket packet, uint8_t type)
- static caerSpecialEventConst [caerSpecialEventPacketFindEventByTypeConst](#) (caerSpecialEventPacket↵ Const packet, uint8_t type)
- static [caerSpecialEvent caerSpecialEventPacketFindValidEventByType](#) (caerSpecialEventPacket packet, uint8_t type)
- static caerSpecialEventConst [caerSpecialEventPacketFindValidEventByTypeConst](#) (caerSpecialEvent↵ PacketConst packet, uint8_t type)

4.23.1 Detailed Description

Special Events format definition and handling functions. This event type encodes special occurrences, such as timestamp related notifications or external input events.

4.23.2 Macro Definition Documentation

4.23.2.1 CAER_SPECIAL_CONST_ITERATOR_ALL_START

```
#define CAER_SPECIAL_CONST_ITERATOR_ALL_START(  
    SPECIAL_PACKET )
```

Value:

```

for (int32_t caerSpecialIteratorCounter = 0;
     caerSpecialIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(SPECIAL_PACKET)->packetHeader); \
     caerSpecialIteratorCounter++) {
    \
    caerSpecialEventConst caerSpecialIteratorElement
    \
    = caerSpecialEventPacketGetEventConst (SPECIAL_PACKET,
    caerSpecialIteratorCounter);

```

Const-Iterator over all special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpecialIteratorElement' variable of type caerSpecialEventConst.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.2 CAER_SPECIAL_CONST_ITERATOR_VALID_START

```

#define CAER_SPECIAL_CONST_ITERATOR_VALID_START(
    SPECIAL_PACKET )

```

Value:

```

for (int32_t caerSpecialIteratorCounter = 0;
     caerSpecialIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(SPECIAL_PACKET)->packetHeader); \
     caerSpecialIteratorCounter++) {
    \
    caerSpecialEventConst caerSpecialIteratorElement
    \
    = caerSpecialEventPacketGetEventConst (SPECIAL_PACKET,
    caerSpecialIteratorCounter);
    if (!caerSpecialEventIsValid(caerSpecialIteratorElement)) {
        \
        continue;
    }
}

```

Const-Iterator over only the valid special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpecialIteratorElement' variable of type caerSpecialEventConst.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.3 CAER_SPECIAL_CONST_REVERSE_ITERATOR_ALL_START

```

#define CAER_SPECIAL_CONST_REVERSE_ITERATOR_ALL_START(
    SPECIAL_PACKET )

```

Value:

```

for (int32_t caerSpecialIteratorCounter
     = caerEventPacketHeaderGetEventNumber (&(SPECIAL_PACKET)->
     packetHeader) - 1; \
     caerSpecialIteratorCounter >= 0; caerSpecialIteratorCounter--) {
    \
    caerSpecialEventConst caerSpecialIteratorElement
    \
    = caerSpecialEventPacketGetEventConst (SPECIAL_PACKET,
    caerSpecialIteratorCounter);
}

```

Const-Reverse iterator over all special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpecialIteratorElement' variable of type caerSpecialEventConst.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.4 CAER_SPECIAL_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_SPECIAL_CONST_REVERSE_ITERATOR_VALID_START(
    SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter
    = caerEventPacketHeaderGetEventNumber (& (SPECIAL_PACKET)->
    packetHeader) - 1;
    caerSpecialIteratorCounter >= 0; caerSpecialIteratorCounter--) {
    caerSpecialEventConst caerSpecialIteratorElement
    = caerSpecialEventPacketGetEventConst (SPECIAL_PACKET,
    caerSpecialIteratorCounter);
    if (!caerSpecialEventIsValid(caerSpecialIteratorElement)) {
        continue;
    }
```

Const-Reverse iterator over only the valid special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpecialIteratorElement' variable of type caerSpecialEventConst.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.5 CAER_SPECIAL_ITERATOR_ALL_END

```
#define CAER_SPECIAL_ITERATOR_ALL_END }
```

Iterator close statement.

4.23.2.6 CAER_SPECIAL_ITERATOR_ALL_START

```
#define CAER_SPECIAL_ITERATOR_ALL_START(
    SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter = 0;
    caerSpecialIteratorCounter < caerEventPacketHeaderGetEventNumber
    (& (SPECIAL_PACKET)->packetHeader);
    caerSpecialIteratorCounter++) {
    caerSpecialEvent caerSpecialIteratorElement
    = caerSpecialEventPacketGetEvent (SPECIAL_PACKET,
    caerSpecialIteratorCounter);
```

Iterator over all special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpecialIteratorElement' variable of type caerSpecialEvent.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.7 CAER_SPECIAL_ITERATOR_VALID_END

```
#define CAER_SPECIAL_ITERATOR_VALID_END }
```

Iterator close statement.

4.23.2.8 CAER_SPECIAL_ITERATOR_VALID_START

```
#define CAER_SPECIAL_ITERATOR_VALID_START (
    SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter = 0;
     caerSpecialIteratorCounter < caerEventPacketHeaderGetEventNumber
     (& (SPECIAL_PACKET)->packetHeader); \
     caerSpecialIteratorCounter++) {
    caerSpecialEvent caerSpecialIteratorElement
    \
    = caerSpecialEventPacketGetEvent (SPECIAL_PACKET,
    caerSpecialIteratorCounter);
    if (!caerSpecialEventIsValid(caerSpecialIteratorElement)) {
        continue;
    }
}
```

Iterator over only the valid special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpecialIteratorElement' variable of type caerSpecialEvent.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.9 CAER_SPECIAL_REVERSE_ITERATOR_ALL_END

```
#define CAER_SPECIAL_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.23.2.10 CAER_SPECIAL_REVERSE_ITERATOR_ALL_START

```
#define CAER_SPECIAL_REVERSE_ITERATOR_ALL_START (
    SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter
     = caerEventPacketHeaderGetEventNumber (& (SPECIAL_PACKET)->
     packetHeader) - 1; \
     caerSpecialIteratorCounter >= 0; caerSpecialIteratorCounter--) {
    caerSpecialEvent caerSpecialIteratorElement
    \
    = caerSpecialEventPacketGetEvent (SPECIAL_PACKET,
    caerSpecialIteratorCounter);
}
```

Reverse iterator over all special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpecialIteratorElement' variable of type caerSpecialEvent.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.11 CAER_SPECIAL_REVERSE_ITERATOR_VALID_END

```
#define CAER_SPECIAL_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.23.2.12 CAER_SPECIAL_REVERSE_ITERATOR_VALID_START

```
#define CAER_SPECIAL_REVERSE_ITERATOR_VALID_START(
    SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter           \
     = caerEventPacketHeaderGetEventNumber (& (SPECIAL_PACKET)-> \
     packetHeader) - 1; \
     caerSpecialIteratorCounter >= 0; caerSpecialIteratorCounter--) { \
    caerSpecialEvent caerSpecialIteratorElement \
    = caerSpecialEventPacketGetEvent (SPECIAL_PACKET, \
    caerSpecialIteratorCounter); \
    if (!caerSpecialEventIsValid(caerSpecialIteratorElement)) { \
        \
        continue; \
    } \
}
```

Reverse iterator over only the valid special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpecialIteratorElement' variable of type caerSpecialEvent.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.13 SPECIAL_DATA_MASK

```
#define SPECIAL_DATA_MASK 0x00FFFFFF
```

Shift and mask values for the type and data portions of a special event. Up to 128 types, with 24 bits of data each, are possible. Bit 0 is the valid mark, see ['common.h'](#) for more details.

4.23.2.14 SPECIAL_DATA_SHIFT

```
#define SPECIAL_DATA_SHIFT 8
```

Shift and mask values for the type and data portions of a special event. Up to 128 types, with 24 bits of data each, are possible. Bit 0 is the valid mark, see ['common.h'](#) for more details.

4.23.2.15 SPECIAL_TYPE_MASK

```
#define SPECIAL_TYPE_MASK 0x0000007F
```

Shift and mask values for the type and data portions of a special event. Up to 128 types, with 24 bits of data each, are possible. Bit 0 is the valid mark, see ['common.h'](#) for more details.

4.23.2.16 SPECIAL_TYPE_SHIFT

```
#define SPECIAL_TYPE_SHIFT 1
```

Shift and mask values for the type and data portions of a special event. Up to 128 types, with 24 bits of data each, are possible. Bit 0 is the valid mark, see ['common.h'](#) for more details.

4.23.3 Typedef Documentation

4.23.3.1 caerSpecialEvent

```
typedef struct caer_special_event* caerSpecialEvent
```

Type for pointer to special event data structure.

4.23.3.2 caerSpecialEventPacket

```
typedef struct caer_special_event_packet* caerSpecialEventPacket
```

Type for pointer to special event packet data structure.

4.23.4 Enumeration Type Documentation

4.23.4.1 caer_special_event_types

```
enum caer_special_event_types
```

List of all special event type identifiers. Used to interpret the special event type field.

Enumerator

TIMESTAMP_WRAP	A 32 bit timestamp wrap occurred.
TIMESTAMP_RESET	A timestamp reset occurred.
EXTERNAL_INPUT_RISING_EDGE	A rising edge was detected (External Input module on device).
EXTERNAL_INPUT_FALLING_EDGE	A falling edge was detected (External Input module on device).
EXTERNAL_INPUT_PULSE	A pulse was detected (External Input module on device).
DVS_ROW_ONLY	A DVS row-only event was detected (a row address without any following column addresses).
EXTERNAL_INPUT1_RISING_EDGE	A rising edge was detected (External Input 1 module on device).
EXTERNAL_INPUT1_FALLING_EDGE	A falling edge was detected (External Input 1 module on device).
EXTERNAL_INPUT1_PULSE	A pulse was detected (External Input 1 module on device).
EXTERNAL_INPUT2_RISING_EDGE	A rising edge was detected (External Input 2 module on device).
EXTERNAL_INPUT2_FALLING_EDGE	A falling edge was detected (External Input 2 module on device).
EXTERNAL_INPUT2_PULSE	A pulse was detected (External Input 2 module on device).

Enumerator

EXTERNAL_GENERATOR_RISING_EDGE	A rising edge was generated (External Input Generator module on device).
EXTERNAL_GENERATOR_FALLING_EDGE	A falling edge was generated (External Input Generator module on device).
APS_FRAME_START	An APS frame capture has started (Frame Event will follow).
APS_FRAME_END	An APS frame capture has completed (Frame Event is alongside).
APS_EXPOSURE_START	An APS frame exposure has started (Frame Event will follow).
APS_EXPOSURE_END	An APS frame exposure has completed (Frame Event will follow).

4.23.5 Function Documentation

4.23.5.1 caerSpecialEventGetData()

```
static uint32_t caerSpecialEventGetData (
    caerSpecialEventConst event ) [inline], [static]
```

Get the special event data. Its meaning depends on the type. Current types that make use of it are (see 'enum caer_special_event_types'):

- DVS_ROW_ONLY: encodes the address of the row from the row-only event.

Parameters

<i>event</i>	a valid SpecialEvent pointer. Cannot be NULL.
--------------	---

Returns

the special event data.

4.23.5.2 caerSpecialEventGetTimestamp()

```
static int32_t caerSpecialEventGetTimestamp (
    caerSpecialEventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid SpecialEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond timestamp.

4.23.5.3 caerSpecialEventGetTimestamp64()

```
static int64_t caerSpecialEventGetTimestamp64 (  
    caerSpecialEventConst event,  
    caerSpecialEventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid SpecialEvent pointer. Cannot be NULL.
<i>packet</i>	the SpecialEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.23.5.4 caerSpecialEventGetType()

```
static uint8_t caerSpecialEventGetType (  
    caerSpecialEventConst event ) [inline], [static]
```

Get the numerical special event type.

Parameters

<i>event</i>	a valid SpecialEvent pointer. Cannot be NULL.
--------------	---

Returns

the special event type (see 'enum caer_special_event_types').

4.23.5.5 caerSpecialEventInvalidate()

```
static void caerSpecialEventInvalidate (
    caerSpecialEvent event,
    caerSpecialEventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid SpecialEvent pointer. Cannot be NULL.
<i>packet</i>	the SpecialEventPacket pointer for the packet containing this event. Cannot be NULL.

4.23.5.6 caerSpecialEventIsValid()

```
static bool caerSpecialEventIsValid (
    caerSpecialEventConst event ) [inline], [static]
```

Check if this special event is valid.

Parameters

<i>event</i>	a valid SpecialEvent pointer. Cannot be NULL.
--------------	---

Returns

true if valid, false if not.

4.23.5.7 caerSpecialEventPacketAllocate()

```
static caerSpecialEventPacket caerSpecialEventPacketAllocate (
    int32_t eventCapacity,
    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]
```

Allocate a new special events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid `SpecialEventPacket` handle or NULL on error.

4.23.5.8 caerSpecialEventPacketFindEventByType()

```
static caerSpecialEvent caerSpecialEventPacketFindEventByType (
    caerSpecialEventPacket packet,
    uint8_t type ) [inline], [static]
```

Get the first special event with the given event type in this event packet. This returns the first found event with that type ID, or NULL if we get to the end without finding any such event.

Parameters

<i>packet</i>	a valid <code>SpecialEventPacket</code> pointer. Cannot be NULL.
<i>type</i>	the special event type to search for.

Returns

the requested special event or NULL on error/not found.

4.23.5.9 caerSpecialEventPacketFindEventByTypeConst()

```
static caerSpecialEventConst caerSpecialEventPacketFindEventByTypeConst (
    caerSpecialEventPacketConst packet,
    uint8_t type ) [inline], [static]
```

Get the first special event with the given event type in this event packet. This returns the first found event with that type ID, or NULL if we get to the end without finding any such event. The returned event is read-only!

Parameters

<i>packet</i>	a valid <code>SpecialEventPacket</code> pointer. Cannot be NULL.
<i>type</i>	the special event type to search for.

Returns

the requested read-only special event or NULL on error/not found.

4.23.5.10 caerSpecialEventPacketFindValidEventByType()

```
static caerSpecialEvent caerSpecialEventPacketFindValidEventByType (
    caerSpecialEventPacket packet,
    uint8_t type ) [inline], [static]
```

Get the first valid special event with the given event type in this event packet. This returns the first found valid event with that type ID, or NULL if we get to the end without finding any such event.

Parameters

<i>packet</i>	a valid SpecialEventPacket pointer. Cannot be NULL.
<i>type</i>	the special event type to search for.

Returns

the requested valid special event or NULL on error/not found.

4.23.5.11 caerSpecialEventPacketFindValidEventByTypeConst()

```
static caerSpecialEventConst caerSpecialEventPacketFindValidEventByTypeConst (
    caerSpecialEventPacketConst packet,
    uint8_t type ) [inline], [static]
```

Get the first valid special event with the given event type in this event packet. This returns the first found valid event with that type ID, or NULL if we get to the end without finding any such event. The returned event is read-only!

Parameters

<i>packet</i>	a valid SpecialEventPacket pointer. Cannot be NULL.
<i>type</i>	the special event type to search for.

Returns

the requested read-only valid special event or NULL on error/not found.

4.23.5.12 caerSpecialEventPacketFromPacketHeader()

```
static caerSpecialEventPacket caerSpecialEventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]
```

Transform a generic event packet header into a Special event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.23.5.13 caerSpecialEventPacketFromPacketHeaderConst()

```
static caerSpecialEventPacketConst caerSpecialEventPacketFromPacketHeaderConst (
    caerEventPacketHeaderConst header ) [inline], [static]
```

Transform a generic read-only event packet header into a read-only Special event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.23.5.14 caerSpecialEventPacketGetEvent()

```
static caerSpecialEvent caerSpecialEventPacketGetEvent (
    caerSpecialEventPacket packet,
    int32_t n ) [inline], [static]
```

Get the special event at the given index from the event packet.

Parameters

<i>packet</i>	a valid SpecialEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested special event. NULL on error.

4.23.5.15 caerSpecialEventPacketGetEventConst()

```
static caerSpecialEventConst caerSpecialEventPacketGetEventConst (
    caerSpecialEventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the special event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid SpecialEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only special event. NULL on error.

4.23.5.16 caerSpecialEventSetData()

```
static void caerSpecialEventSetData (
    caerSpecialEvent event,
    uint32_t data ) [inline], [static]
```

Set the special event data. Its meaning depends on the type. Current types that make use of it are (see 'enum caer_special_event_types'):

- DVS_ROW_ONLY: encodes the address of the row from the row-only event.

Parameters

<i>event</i>	a valid SpecialEvent pointer. Cannot be NULL.
<i>data</i>	the special event data.

4.23.5.17 caerSpecialEventSetTimestamp()

```
static void caerSpecialEventSetTimestamp (
    caerSpecialEvent event,
    int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid SpecialEvent pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.23.5.18 caerSpecialEventSetType()

```
static void caerSpecialEventSetType (
```

```

    caerSpecialEvent event,
    uint8_t type ) [inline], [static]

```

Set the numerical special event type.

Parameters

<i>event</i>	a valid SpecialEvent pointer. Cannot be NULL.
<i>type</i>	the special event type (see 'enum caer_special_event_types').

4.23.5.19 caerSpecialEventValidate()

```

static void caerSpecialEventValidate (
    caerSpecialEvent event,
    caerSpecialEventPacket packet ) [inline], [static]

```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid SpecialEvent pointer. Cannot be NULL.
<i>packet</i>	the SpecialEventPacket pointer for the packet containing this event. Cannot be NULL.

4.23.5.20 PACKED_STRUCT() [1/2]

```

PACKED_STRUCT (
    struct caer_special_event { uint32_t data;int32_t timestamp;} )

```

Special event data structure definition. This contains the actual data, as well as the 32 bit event timestamp. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.23.5.21 PACKED_STRUCT() [2/2]

```

PACKED_STRUCT (
    struct caer_special_event_packet { struct caer_event_packet_header packetHeader;struct
    caer_special_event events[];} )

```

Special event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.24 events/spike.h File Reference

```
#include "common.h"
```

Macros

- `#define CAER_SPIKE_ITERATOR_ALL_START(SPIKE_PACKET)`
 - `#define CAER_SPIKE_CONST_ITERATOR_ALL_START(SPIKE_PACKET)`
 - `#define CAER_SPIKE_ITERATOR_ALL_END }`
 - `#define CAER_SPIKE_ITERATOR_VALID_START(SPIKE_PACKET)`
 - `#define CAER_SPIKE_CONST_ITERATOR_VALID_START(SPIKE_PACKET)`
 - `#define CAER_SPIKE_ITERATOR_VALID_END }`
 - `#define CAER_SPIKE_REVERSE_ITERATOR_ALL_START(SPIKE_PACKET)`
 - `#define CAER_SPIKE_CONST_REVERSE_ITERATOR_ALL_START(SPIKE_PACKET)`
 - `#define CAER_SPIKE_REVERSE_ITERATOR_ALL_END }`
 - `#define CAER_SPIKE_REVERSE_ITERATOR_VALID_START(SPIKE_PACKET)`
 - `#define CAER_SPIKE_CONST_REVERSE_ITERATOR_VALID_START(SPIKE_PACKET)`
 - `#define CAER_SPIKE_REVERSE_ITERATOR_VALID_END }`
-
- `#define SPIKE_SOURCE_CORE_ID_SHIFT 1`
 - `#define SPIKE_SOURCE_CORE_ID_MASK 0x0000001F`
 - `#define SPIKE_CHIP_ID_SHIFT 6`
 - `#define SPIKE_CHIP_ID_MASK 0x0000003F`
 - `#define SPIKE_NEURON_ID_SHIFT 12`
 - `#define SPIKE_NEURON_ID_MASK 0x000FFFFF`

Typedefs

- `typedef struct caer_spike_event * caerSpikeEvent`
- `typedef const struct caer_spike_event * caerSpikeEventConst`
- `typedef struct caer_spike_event_packet * caerSpikeEventPacket`
- `typedef const struct caer_spike_event_packet * caerSpikeEventPacketConst`

Functions

- `PACKED_STRUCT (struct caer_spike_event { uint32_t data;int32_t timestamp;})`
- `PACKED_STRUCT (struct caer_spike_event_packet { struct caer_event_packet_header packetHeader;struct caer_spike_event events[];})`
- `static caerSpikeEventPacket caerSpikeEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)`
- `static caerSpikeEventPacket caerSpikeEventPacketFromPacketHeader (caerEventPacketHeader header)`
- `static caerSpikeEventPacketConst caerSpikeEventPacketFromPacketHeaderConst (caerEventPacketHeaderConst header)`
- `static caerSpikeEvent caerSpikeEventPacketGetEvent (caerSpikeEventPacket packet, int32_t n)`
- `static caerSpikeEventConst caerSpikeEventPacketGetEventConst (caerSpikeEventPacketConst packet, int32_t n)`
- `static int32_t caerSpikeEventGetTimestamp (caerSpikeEventConst event)`

- static int64_t [caerSpikeEventGetTimestamp64](#) (caerSpikeEventConst event, caerSpikeEventPacketConst packet)
- static void [caerSpikeEventSetTimestamp](#) (caerSpikeEvent event, int32_t timestamp)
- static bool [caerSpikeEventIsValid](#) (caerSpikeEventConst event)
- static void [caerSpikeEventValidate](#) (caerSpikeEvent event, caerSpikeEventPacket packet)
- static void [caerSpikeEventInvalidate](#) (caerSpikeEvent event, caerSpikeEventPacket packet)
- static uint8_t [caerSpikeEventGetSourceCoreID](#) (caerSpikeEventConst event)
- static void [caerSpikeEventSetSourceCoreID](#) (caerSpikeEvent event, uint8_t sourceCoreID)
- static uint8_t [caerSpikeEventGetChipID](#) (caerSpikeEventConst event)
- static void [caerSpikeEventSetChipID](#) (caerSpikeEvent event, uint8_t chipID)
- static uint32_t [caerSpikeEventGetNeuronID](#) (caerSpikeEventConst event)
- static void [caerSpikeEventSetNeuronID](#) (caerSpikeEvent event, uint32_t neuronID)

4.24.1 Detailed Description

Spike Events format definition and handling functions. This contains spikes generated by a neuron-array chip.

4.24.2 Macro Definition Documentation

4.24.2.1 CAER_SPIKE_CONST_ITERATOR_ALL_START

```
#define CAER_SPIKE_CONST_ITERATOR_ALL_START(  
    SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = 0;  
     caerSpikeIteratorCounter < caerEventPacketHeaderGetEventNumber(  
         &(SPIKE_PACKET)->packetHeader); \br/>     caerSpikeIteratorCounter++) {  
     caerSpikeEventConst caerSpikeIteratorElement  
         = caerSpikeEventPacketGetEventConst (SPIKE_PACKET,  
         caerSpikeIteratorCounter);
```

Const-Iterator over all Spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpikeIteratorElement' variable of type caerSpikeEvent↵ Const.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.2 CAER_SPIKE_CONST_ITERATOR_VALID_START

```
#define CAER_SPIKE_CONST_ITERATOR_VALID_START(  
    SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = 0; \
     caerSpikeIteratorCounter < caerEventPacketHeaderGetEventNumber(  
     &(SPIKE_PACKET)->packetHeader); \
     caerSpikeIteratorCounter++) { \
    caerSpikeEventConst caerSpikeIteratorElement \
    = caerSpikeEventPacketGetEventConst(SPIKE_PACKET, \
    caerSpikeIteratorCounter); \
    if (!caerSpikeEventIsValid(caerSpikeIteratorElement)) { \
        continue; \
    }
```

Const-Iterator over only the valid Spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpikeIteratorElement' variable of type caerSpikeEventConst.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.3 CAER_SPIKE_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_SPIKE_CONST_REVERSE_ITERATOR_ALL_START(  
    SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = caerEventPacketHeaderGetEventNumber  
     (&(SPIKE_PACKET)->packetHeader) - 1; \
     caerSpikeIteratorCounter >= 0; caerSpikeIteratorCounter--) { \
    caerSpikeEventConst caerSpikeIteratorElement \
    = caerSpikeEventPacketGetEventConst(SPIKE_PACKET, \
    caerSpikeIteratorCounter);
```

Const-Reverse iterator over all spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpikeIteratorElement' variable of type caerSpikeEventConst.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.4 CAER_SPIKE_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_SPIKE_CONST_REVERSE_ITERATOR_VALID_START(  
    SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = caerEventPacketHeaderGetEventNumber  
     (&(SPIKE_PACKET)->packetHeader) - 1; \
     caerSpikeIteratorCounter >= 0; caerSpikeIteratorCounter--) { \
    caerSpikeEventConst caerSpikeIteratorElement \
    = caerSpikeEventPacketGetEventConst(SPIKE_PACKET, \
    caerSpikeIteratorCounter); \
    if (!caerSpikeEventIsValid(caerSpikeIteratorElement)) { \
        continue; \
    }
```

Const-Reverse iterator over only the valid spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpikeIteratorElement' variable of type caerSpikeEventConst.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.5 CAER_SPIKE_ITERATOR_ALL_END

```
#define CAER_SPIKE_ITERATOR_ALL_END }
```

Iterator close statement.

4.24.2.6 CAER_SPIKE_ITERATOR_ALL_START

```
#define CAER_SPIKE_ITERATOR_ALL_START(  
    SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = 0; \
     caerSpikeIteratorCounter < caerEventPacketHeaderGetEventNumber(  
     &(SPIKE_PACKET)->packetHeader); \
     caerSpikeIteratorCounter++) { \
     caerSpikeEvent caerSpikeIteratorElement = caerSpikeEventPacketGetEvent(  
     SPIKE_PACKET, caerSpikeIteratorCounter);
```

Iterator over all Spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpikeIteratorElement' variable of type caerSpikeEvent.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.7 CAER_SPIKE_ITERATOR_VALID_END

```
#define CAER_SPIKE_ITERATOR_VALID_END }
```

Iterator close statement.

4.24.2.8 CAER_SPIKE_ITERATOR_VALID_START

```
#define CAER_SPIKE_ITERATOR_VALID_START(  
    SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = 0; \
     caerSpikeIteratorCounter < caerEventPacketHeaderGetEventNumber(  
     &(SPIKE_PACKET)->packetHeader); \
     caerSpikeIteratorCounter++) { \
     caerSpikeEvent caerSpikeIteratorElement \
     = caerSpikeEventPacketGetEvent(SPIKE_PACKET, \
     caerSpikeIteratorCounter); \
     if (!caerSpikeEventIsValid(caerSpikeIteratorElement)) { \
         continue; \
     }
```

Iterator over only the valid Spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpikeIteratorElement' variable of type caerSpikeEvent.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.9 CAER_SPIKE_REVERSE_ITERATOR_ALL_END

```
#define CAER_SPIKE_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.24.2.10 CAER_SPIKE_REVERSE_ITERATOR_ALL_START

```
#define CAER_SPIKE_REVERSE_ITERATOR_ALL_START(  
    SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(SPIKE_PACKET)->packetHeader) - 1; \  
    caerSpikeIteratorCounter >= 0; caerSpikeIteratorCounter--) {  
    \  
    caerSpikeEvent caerSpikeIteratorElement = caerSpikeEventPacketGetEvent(  
        SPIKE_PACKET, caerSpikeIteratorCounter);
```

Reverse iterator over all spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpikeIteratorElement' variable of type caerSpikeEvent.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.11 CAER_SPIKE_REVERSE_ITERATOR_VALID_END

```
#define CAER_SPIKE_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.24.2.12 CAER_SPIKE_REVERSE_ITERATOR_VALID_START

```
#define CAER_SPIKE_REVERSE_ITERATOR_VALID_START(  
    SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = caerEventPacketHeaderGetEventNumber  
    (&(SPIKE_PACKET)->packetHeader) - 1; \  
    caerSpikeIteratorCounter >= 0; caerSpikeIteratorCounter--) {  
    \  
    caerSpikeEvent caerSpikeIteratorElement  
    \  
    = caerSpikeEventPacketGetEvent(SPIKE_PACKET,  
    caerSpikeIteratorCounter);  
    if (!caerSpikeEventIsValid(caerSpikeIteratorElement)) {  
    \  
        continue;  
    }  
}
```

Reverse iterator over only the valid spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpikeIteratorElement' variable of type caerSpikeEvent.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.13 SPIKE_CHIP_ID_MASK

```
#define SPIKE_CHIP_ID_MASK 0x0000003F
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.24.2.14 SPIKE_CHIP_ID_SHIFT

```
#define SPIKE_CHIP_ID_SHIFT 6
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.24.2.15 SPIKE_NEURON_ID_MASK

```
#define SPIKE_NEURON_ID_MASK 0x000FFFFF
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.24.2.16 SPIKE_NEURON_ID_SHIFT

```
#define SPIKE_NEURON_ID_SHIFT 12
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.24.2.17 SPIKE_SOURCE_CORE_ID_MASK

```
#define SPIKE_SOURCE_CORE_ID_MASK 0x0000001F
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.24.2.18 SPIKE_SOURCE_CORE_ID_SHIFT

```
#define SPIKE_SOURCE_CORE_ID_SHIFT 1
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see '[common.h](#)' for more details.

4.24.3 Typedef Documentation

4.24.3.1 caerSpikeEvent

```
typedef struct caer_spike_event* caerSpikeEvent
```

Type for pointer to Spike event data structure.

4.24.3.2 caerSpikeEventPacket

```
typedef struct caer_spike_event_packet* caerSpikeEventPacket
```

Type for pointer to Spike event packet data structure.

4.24.4 Function Documentation

4.24.4.1 caerSpikeEventGetChipID()

```
static uint8_t caerSpikeEventGetChipID (  
    caerSpikeEventConst event ) [inline], [static]
```

Get the chip ID.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
--------------	---

Returns

the Spike's chip ID.

4.24.4.2 caerSpikeEventGetNeuronID()

```
static uint32_t caerSpikeEventGetNeuronID (  
    caerSpikeEventConst event ) [inline], [static]
```

Get the neuron ID.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
--------------	---

Returns

the Spike's neuron ID.

4.24.4.3 caerSpikeEventGetSourceCoreID()

```
static uint8_t caerSpikeEventGetSourceCoreID (
    caerSpikeEventConst event ) [inline], [static]
```

Get the source core ID.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
--------------	---

Returns

the Spike's source core ID.

4.24.4.4 caerSpikeEventGetTimestamp()

```
static int32_t caerSpikeEventGetTimestamp (
    caerSpikeEventConst event ) [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
--------------	---

Returns

this event's 32bit microsecond timestamp.

4.24.4.5 caerSpikeEventGetTimestamp64()

```
static int64_t caerSpikeEventGetTimestamp64 (
    caerSpikeEventConst event,
    caerSpikeEventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See '[caerEventPacketHeaderGetEventTSOverflow\(\)](#)' documentation for more details on the 64bit timestamp.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
<i>packet</i>	the SpikeEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.24.4.6 caerSpikeEventInvalidate()

```
static void caerSpikeEventInvalidate (
    caerSpikeEvent event,
    caerSpikeEventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
<i>packet</i>	the SpikeEventPacket pointer for the packet containing this event. Cannot be NULL.

4.24.4.7 caerSpikeEventsValid()

```
static bool caerSpikeEventIsValid (
    caerSpikeEventConst event ) [inline], [static]
```

Check if this Spike event is valid.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
--------------	---

Returns

true if valid, false if not.

4.24.4.8 caerSpikeEventPacketAllocate()

```
static caerSpikeEventPacket caerSpikeEventPacketAllocate (
    int32_t eventCapacity,
```

```

    int16_t eventSource,
    int32_t tsOverflow ) [inline], [static]

```

Allocate a new Spike events packet. Use free() to reclaim this memory.

Parameters

<i>eventCapacity</i>	the maximum number of events this packet will hold.
<i>eventSource</i>	the unique ID representing the source/generator of this packet.
<i>tsOverflow</i>	the current timestamp overflow counter value for this packet.

Returns

a valid SpikeEventPacket handle or NULL on error.

4.24.4.9 caerSpikeEventPacketFromPacketHeader()

```

static caerSpikeEventPacket caerSpikeEventPacketFromPacketHeader (
    caerEventPacketHeader header ) [inline], [static]

```

Transform a generic event packet header into a Spike event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, typed event packet pointer.

4.24.4.10 caerSpikeEventPacketFromPacketHeaderConst()

```

static caerSpikeEventPacketConst caerSpikeEventPacketFromPacketHeaderConst (
    caerEventPacketHeaderConst header ) [inline], [static]

```

Transform a generic read-only event packet header into a read-only Spike event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

<i>header</i>	a valid read-only event packet header pointer. Cannot be NULL.
---------------	--

Returns

a properly converted, read-only typed event packet pointer.

4.24.4.11 caerSpikeEventPacketGetEvent()

```
static caerSpikeEvent caerSpikeEventPacketGetEvent (
    caerSpikeEventPacket packet,
    int32_t n ) [inline], [static]
```

Get the Spike event at the given index from the event packet.

Parameters

<i>packet</i>	a valid SpikeEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Spike event. NULL on error.

4.24.4.12 caerSpikeEventPacketGetEventConst()

```
static caerSpikeEventConst caerSpikeEventPacketGetEventConst (
    caerSpikeEventPacketConst packet,
    int32_t n ) [inline], [static]
```

Get the Spike event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

<i>packet</i>	a valid SpikeEventPacket pointer. Cannot be NULL.
<i>n</i>	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only Spike event. NULL on error.

4.24.4.13 caerSpikeEventSetChipID()

```
static void caerSpikeEventSetChipID (
    caerSpikeEvent event,
    uint8_t chipID ) [inline], [static]
```

Set the chip ID.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
<i>chipID</i>	the Spike's chip ID.

4.24.4.14 caerSpikeEventSetNeuronID()

```
static void caerSpikeEventSetNeuronID (  
    caerSpikeEvent event,  
    uint32_t neuronID ) [inline], [static]
```

Set the neuron ID.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
<i>neuronID</i>	the Spike's neuron ID.

4.24.4.15 caerSpikeEventSetSourceCoreID()

```
static void caerSpikeEventSetSourceCoreID (  
    caerSpikeEvent event,  
    uint8_t sourceCoreID ) [inline], [static]
```

Set the source core ID.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
<i>sourceCoreID</i>	the Spike's source core ID.

4.24.4.16 caerSpikeEventSetTimestamp()

```
static void caerSpikeEventSetTimestamp (  
    caerSpikeEvent event,  
    int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
<i>timestamp</i>	a positive 32bit microsecond timestamp.

4.24.4.17 caerSpikeEventValidate()

```
static void caerSpikeEventValidate (
    caerSpikeEvent event,
    caerSpikeEventPacket packet ) [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

<i>event</i>	a valid SpikeEvent pointer. Cannot be NULL.
<i>packet</i>	the SpikeEventPacket pointer for the packet containing this event. Cannot be NULL.

4.24.4.18 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
    struct caer_spike_event { uint32_t data;int32_t timestamp;} )
```

Spike event data structure definition. This contains the core ID, the neuron ID and the timestamp of the received spike, together with the usual validity mark. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.24.4.19 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
    struct caer_spike_event_packet { struct caer_event_packet_header packetHeader;struct
    caer_spike_event events[];} )
```

Spike event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.25 filters/dvs_noise.h File Reference

```
#include "../events/polarity.h"
```

Data Structures

- struct [caer_filter_dvs_pixel](#)

Macros

- #define [CAER_FILTER_DVS_HOTPIXEL_LEARN](#) 0
- #define [CAER_FILTER_DVS_HOTPIXEL_TIME](#) 1
- #define [CAER_FILTER_DVS_HOTPIXEL_COUNT](#) 2
- #define [CAER_FILTER_DVS_HOTPIXEL_ENABLE](#) 3
- #define [CAER_FILTER_DVS_HOTPIXEL_STATISTICS](#) 4
- #define [CAER_FILTER_DVS_BACKGROUND_ACTIVITY_ENABLE](#) 5
- #define [CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TIME](#) 6
- #define [CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS](#) 7
- #define [CAER_FILTER_DVS_REFRACTORY_PERIOD_ENABLE](#) 8
- #define [CAER_FILTER_DVS_REFRACTORY_PERIOD_TIME](#) 9
- #define [CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS](#) 10
- #define [CAER_FILTER_DVS_LOG_LEVEL](#) 11
- #define [CAER_FILTER_DVS_RESET](#) 12
- #define [CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TWO_LEVELS](#) 13
- #define [CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MIN](#) 14
- #define [CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MAX](#) 15
- #define [CAER_FILTER_DVS_BACKGROUND_ACTIVITY_CHECK_POLARITY](#) 16

Typedefs

- typedef struct [caer_filter_dvs_pixel](#) * [caerFilterDVSPixel](#)
- typedef struct [caer_filter_dvs_noise](#) * [caerFilterDVSNoise](#)

Functions

- [caerFilterDVSNoise](#) [caerFilterDVSNoiseInitialize](#) (uint16_t sizeX, uint16_t sizeY)
- void [caerFilterDVSNoiseDestroy](#) ([caerFilterDVSNoise](#) noiseFilter)
- void [caerFilterDVSNoiseApply](#) ([caerFilterDVSNoise](#) noiseFilter, [caerPolarityEventPacket](#) polarity)
- bool [caerFilterDVSNoiseConfigSet](#) ([caerFilterDVSNoise](#) noiseFilter, uint8_t paramAddr, uint64_t param)
- bool [caerFilterDVSNoiseConfigGet](#) ([caerFilterDVSNoise](#) noiseFilter, uint8_t paramAddr, uint64_t *param)
- ssize_t [caerFilterDVSNoiseGetHotPixels](#) ([caerFilterDVSNoise](#) noiseFilter, [caerFilterDVSPixel](#) *hotPixels)

4.25.1 Detailed Description

The DVS noise filter combines a HotPixel filter (high activity pixels), a Background-Activity filter (uncorrelated events), and a Refractory Period filter (limit event rate of a pixel). The HotPixel and Background-Activity filters reduce noise due to transistor mismatch, the Refractory Period filter can reduce the event rate and is efficient to implement together with the Background-Activity filter, requiring only one pixel memory map for both. Please note that the filter is not thread-safe, all function calls should happen on the same thread, unless you take care that they never overlap.

4.25.2 Macro Definition Documentation

4.25.2.1 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_CHECK_POLARITY

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_CHECK_POLARITY 16
```

DVS Background-Activity Filter: whether polarity is considered when searching the neighbors for supporting activity.

4.25.2.2 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_ENABLE

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_ENABLE 5
```

DVS Background-Activity Filter: enable the background-activity filter, which tries to remove events caused by transistor leakage, by rejecting uncorrelated events.

4.25.2.3 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS 7
```

DVS Background-Activity Filter: number of events filtered out by the background-activity filter.

4.25.2.4 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MAX

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MAX 15
```

DVS Background-Activity Filter: maximum number of pixels in the immediate neighborhood that can support the current pixel for it to be considered valid.

4.25.2.5 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MIN

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MIN 14
```

DVS Background-Activity Filter: minimum number of pixels in the immediate neighborhood that must support the current pixel for it to be considered valid.

4.25.2.6 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TIME

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TIME 6
```

DVS Background-Activity Filter: specify the time difference constant for the background-activity filter in microseconds. Events that do correlated within this time-frame are let through, while others are filtered out.

4.25.2.7 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TWO_LEVELS

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TWO_LEVELS 13
```

DVS Background-Activity Filter: repeat the background-activity check, that at least one neighbor pixel supports this pixel, on each pixel that supported the current pixel in turn, basically repeating the check for a second level of pixels.

4.25.2.8 CAER_FILTER_DVS_HOTPIXEL_COUNT

```
#define CAER_FILTER_DVS_HOTPIXEL_COUNT 2
```

DVS HotPixel Filter: Minimum number of events, during the given learning time, for a pixel to be considered hot.

4.25.2.9 CAER_FILTER_DVS_HOTPIXEL_ENABLE

```
#define CAER_FILTER_DVS_HOTPIXEL_ENABLE 3
```

DVS HotPixel Filter: Enable the hot pixel filter, filtering out the last learned hot pixels.

4.25.2.10 CAER_FILTER_DVS_HOTPIXEL_LEARN

```
#define CAER_FILTER_DVS_HOTPIXEL_LEARN 0
```

DVS HotPixel Filter: Turn on learning to determine which pixels are hot, meaning abnormally active within a certain time period. In the absence of external stimuli, the only pixels behaving as such must be noise. Once learning is enabled, do not disable it until completed. To verify completion, query this parameter and wait for it to switch from 'true' back to 'false'.

4.25.2.11 CAER_FILTER_DVS_HOTPIXEL_STATISTICS

```
#define CAER_FILTER_DVS_HOTPIXEL_STATISTICS 4
```

DVS HotPixel Filter: Number of events filtered out by the hot pixel filter.

4.25.2.12 CAER_FILTER_DVS_HOTPIXEL_TIME

```
#define CAER_FILTER_DVS_HOTPIXEL_TIME 1
```

DVS HotPixel Filter: Minimum time (in μ s) to accumulate events for during learning.

4.25.2.13 CAER_FILTER_DVS_LOG_LEVEL

```
#define CAER_FILTER_DVS_LOG_LEVEL 11
```

DVS Noise Filter: set a custom log-level for an instance of the DVS Noise filter.

4.25.2.14 CAER_FILTER_DVS_REFRACTORY_PERIOD_ENABLE

```
#define CAER_FILTER_DVS_REFRACTORY_PERIOD_ENABLE 8
```

DVS Refractory Period Filter: enable the refractory period filter, which limits the firing rate of pixels.

4.25.2.15 CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS

```
#define CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS 10
```

DVS Refractory Period Filter: number of events filtered out by the refractory period filter.

4.25.2.16 CAER_FILTER_DVS_REFRACTORY_PERIOD_TIME

```
#define CAER_FILTER_DVS_REFRACTORY_PERIOD_TIME 9
```

DVS Refractory Period Filter: specify the time constant for the refractory period filter. Pixels will be inhibited from generating new events during this time after the last even has fired.

4.25.2.17 CAER_FILTER_DVS_RESET

```
#define CAER_FILTER_DVS_RESET 12
```

DVS Noise Filter: reset this instance of the filter to its initial state, forgetting any learned hot pixels and clearing the timestamp map and the statistics. This does not change or reset the configuration.

4.25.3 Typedef Documentation

4.25.3.1 caerFilterDVSNoise

```
typedef struct caer_filter_dvs_noise* caerFilterDVSNoise
```

Pointer to DVS noise filter structure (private).

4.25.3.2 caerFilterDVSPixel

```
typedef struct caer_filter_dvs_pixel* caerFilterDVSPixel
```

Pointer to DVS pixel address structure.

4.25.4 Function Documentation

4.25.4.1 caerFilterDVSNoiseApply()

```
void caerFilterDVSNoiseApply (
    caerFilterDVSNoise noiseFilter,
    caerPolarityEventPacket polarity )
```

Apply the DVS noise filter to the given polarity events packet. This will filter out events by marking them as invalid, depending on the given filter configuration.

Parameters

<i>noiseFilter</i>	a valid DVS noise filter instance.
<i>polarity</i>	a valid polarity event packet. If NULL, no operation is performed.

4.25.4.2 caerFilterDVSNoiseConfigGet()

```
bool caerFilterDVSNoiseConfigGet (
    caerFilterDVSNoise noiseFilter,
    uint8_t paramAddr,
    uint64_t * param )
```

Get DVS noise filter configuration parameters.

Parameters

<i>noiseFilter</i>	a valid DVS noise filter instance.
<i>paramAddr</i>	a configuration parameter address, see defines CAER_FILTER_DVS_*.
<i>param</i>	a pointer to a configuration parameter value integer, in which to store the current value.

Returns

true if operation successful, false otherwise.

4.25.4.3 caerFilterDVSNoiseConfigSet()

```
bool caerFilterDVSNoiseConfigSet (
    caerFilterDVSNoise noiseFilter,
    uint8_t paramAddr,
    uint64_t param )
```

Set DVS noise filter configuration parameters.

Parameters

<i>noiseFilter</i>	a valid DVS noise filter instance.
<i>paramAddr</i>	a configuration parameter address, see defines CAER_FILTER_DVS_*.
<i>param</i>	a configuration parameter value integer.

Returns

true if operation successful, false otherwise.

4.25.4.4 caerFilterDVSNoiseDestroy()

```
void caerFilterDVSNoiseDestroy (
    caerFilterDVSNoise noiseFilter )
```

Destroy a DVS noise filter instance and free its memory.

Parameters

<i>noiseFilter</i>	a valid DVS noise filter instance.
--------------------	------------------------------------

4.25.4.5 caerFilterDVSNoiseGetHotPixels()

```
ssize_t caerFilterDVSNoiseGetHotPixels (
    caerFilterDVSNoise noiseFilter,
    caerFilterDVSPixel * hotPixels )
```

Get an array of currently learned hot pixels, in order of activity (most active first, least active last). Useful for working with hardware-based pixel filtering (FPGA/CPLD).

Parameters

<i>noiseFilter</i>	a valid DVS noise filter instance.
<i>hotPixels</i>	array of DVS pixel addresses, sorted by activity (most active first). Memory will be allocated for it automatically. On error, the pointer is set to NULL. Remember to free() the memory once done!

Returns

number of hot pixels in array, 0 if no hot pixels were found; or -1 if an error occurred.

4.25.4.6 caerFilterDVSNoiseInitialize()

```
caerFilterDVSNoise caerFilterDVSNoiseInitialize (
    uint16_t sizeX,
    uint16_t sizeY )
```

Allocate memory and initialize the DVS noise filter. This filter combines a HotPixel filter (high activity pixels), a Background-Activity filter (uncorrelated events), and a Refractory Period filter (limit event rate of a pixel). The HotPixel and Background-Activity filters reduce noise due to transistor mismatch, the Refractory Period filter can reduce the event rate and is efficient to implement together with the Background-Activity filter, requiring only one pixel memory map for both. At initialization, all filters are disabled. You must configure and enable them using [caerFilterDVSNoiseConfigSet\(\)](#). You must specify the maximum resolution at initialization, as it is used to set up efficient lookup tables.

Parameters

<i>sizeX</i>	maximum X axis resolution.
<i>sizeY</i>	maximum Y axis resolution.

Returns

DVS noise filter instance, NULL on error.

4.26 frame_utils.h File Reference

```
#include "events/frame.h"
```

Enumerations

- enum **caer_frame_utils_demosaic_types** { DEMOSAIC_STANDARD = 0, DEMOSAIC_OPENCV_NORMAL = 1, DEMOSAIC_OPENCV_EDGE_AWARE = 2 }
- enum **caer_frame_utils_contrast_types** { CONTRAST_STANDARD = 0, CONTRAST_OPENCV_NORMALIZATION = 1, CONTRAST_OPENCV_HISTOGRAM_EQUALIZATION = 2, CONTRAST_OPENCV_CLAHE = 3 }

Functions

- [caerFrameEventPacket](#) **caerFrameUtilsDemosaic** (caerFrameEventPacketConst framePacket, enum caer_frame_utils_demosaic_types demosaicType)
- void **caerFrameUtilsContrast** ([caerFrameEventPacket](#) framePacket, enum caer_frame_utils_contrast_types contrastType)

4.26.1 Detailed Description

Functions for frame enhancement and demosaicing. Basic variants that don't require any external dependencies, such as OpenCV. Use of the OpenCV variants is recommended for quality and performance, and can optionally be enabled at build-time.

4.27 libcaer.h File Reference

```
#include <stddef.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdint.h>
#include <inttypes.h>
#include <string.h>
#include <errno.h>
#include "portable_endian.h"
#include "log.h"
```

Macros

- #define **PACKED_STRUCT**(STRUCT_DECLARATION) STRUCT_DECLARATION
 - #define **DEPRECATED_FUNCTION**(DEPR_MSG)
 - #define **LIBCAER_VERSION** ((2 * 10000) + (5 * 100) + 0)
 - #define **LIBCAER_NAME_STRING** "libcaer"
 - #define **LIBCAER_VERSION_STRING** "2.5.0"
 - #define **LIBCAER_HAVE_SERIALDEV** 1
 - #define **LIBCAER_HAVE_OPENCV** 1
 - #define **U8T**(X) ((uint8_t) (X))
 - #define **U16T**(X) ((uint16_t) (X))
 - #define **U32T**(X) ((uint32_t) (X))
 - #define **U64T**(X) ((uint64_t) (X))
 - #define **I8T**(X) ((int8_t) (X))
 - #define **I16T**(X) ((int16_t) (X))
 - #define **I32T**(X) ((int32_t) (X))
 - #define **I64T**(X) ((int64_t) (X))
 - #define **MASK_NUMBITS32**(X) **U32T**(**U32T**(**U32T**(1) << X) - 1)
 - #define **MASK_NUMBITS64**(X) **U64T**(**U64T**(**U64T**(1) << X) - 1)
 - #define **SWAP_VAR**(type, x, y) { type tmpv; tmpv = (x); (x) = (y); (y) = tmpv; }
-
- #define **CLEAR_NUMBITS32**(VAR, SHIFT, MASK) (VAR) &= htobe32(~(**U32T**(**U32T**(MASK) << (SHIFT))))
 - #define **CLEAR_NUMBITS16**(VAR, SHIFT, MASK) (VAR) &= htobe16(~(**U16T**(**U16T**(MASK) << (SHIFT))))
 - #define **CLEAR_NUMBITS8**(VAR, SHIFT, MASK) (VAR) &= **U8T**(~(**U8T**(**U8T**(MASK) << (SHIFT))))
-
- #define **SET_NUMBITS32**(VAR, SHIFT, MASK, VALUE) (VAR) |= htobe32(**U32T**((**U32T**(VALUE) & (MASK)) << (SHIFT)))
 - #define **SET_NUMBITS16**(VAR, SHIFT, MASK, VALUE) (VAR) |= htobe16(**U16T**((**U16T**(VALUE) & (MASK)) << (SHIFT)))
 - #define **SET_NUMBITS8**(VAR, SHIFT, MASK, VALUE) (VAR) |= **U8T**((**U8T**(VALUE) & (MASK)) << (SHIFT))
-
- #define **GET_NUMBITS32**(VAR, SHIFT, MASK) ((le32toh(VAR) >> (SHIFT)) & (MASK))
 - #define **GET_NUMBITS16**(VAR, SHIFT, MASK) ((le16toh(VAR) >> (SHIFT)) & (MASK))
 - #define **GET_NUMBITS8**(VAR, SHIFT, MASK) ((**U8T**(VAR) >> (SHIFT)) & (MASK))

Enumerations

- enum **caer_error_codes** {
CAER_ERROR_MEMORY_ALLOCATION = -1, **CAER_ERROR_RESOURCE_ALLOCATION** = -2, **CAER_ERROR_OPEN_ACCESS** = -3, **CAER_ERROR_COMMUNICATION** = -4,
CAER_ERROR_FW_VERSION = -5, **CAER_ERROR_LOGIC_VERSION** = -6 }

Functions

- static bool [caerStrEquals](#) (const char *s1, const char *s2)
- static bool [caerStrEqualsUpTo](#) (const char *s1, const char *s2, size_t len)
- static void [caerIntegerToByteArray](#) (const uint32_t integer, uint8_t *byteArray, const uint8_t byteArrayLength)
- static uint32_t [caerByteArrayToInteger](#) (const uint8_t *byteArray, const uint8_t byteArrayLength)

4.27.1 Detailed Description

Main libcaer header; provides inclusions for common system functions and definitions for useful macros used often in the code. Also includes the logging functions and definitions and several useful static inline functions for string comparison and byte array manipulation. When including libcaer, please make sure to always use the full path, ie. `#include <libcaer/libcaer.h>` and not just `#include <libcaer.h>`.

4.27.2 Macro Definition Documentation

4.27.2.1 CLEAR_NUMBITS16

```
#define CLEAR_NUMBITS16(  
    VAR,  
    SHIFT,  
    MASK ) (VAR) &= htobe16(~(U16T(U16T(MASK) << (SHIFT))))
```

Clear bits given by mask (amount) and shift (position).

4.27.2.2 CLEAR_NUMBITS32

```
#define CLEAR_NUMBITS32(  
    VAR,  
    SHIFT,  
    MASK ) (VAR) &= htobe32(~(U32T(U32T(MASK) << (SHIFT))))
```

Clear bits given by mask (amount) and shift (position).

4.27.2.3 CLEAR_NUMBITS8

```
#define CLEAR_NUMBITS8(  
    VAR,  
    SHIFT,  
    MASK ) (VAR) &= U8T(~(U8T(U8T(MASK) << (SHIFT))))
```

Clear bits given by mask (amount) and shift (position).

4.27.2.4 GET_NUMBITS16

```
#define GET_NUMBITS16(
    VAR,
    SHIFT,
    MASK ) ((le16toh(VAR) >> (SHIFT)) & (MASK))
```

Get value of bits given by mask (amount) and shift (position).

4.27.2.5 GET_NUMBITS32

```
#define GET_NUMBITS32(
    VAR,
    SHIFT,
    MASK ) ((le32toh(VAR) >> (SHIFT)) & (MASK))
```

Get value of bits given by mask (amount) and shift (position).

4.27.2.6 GET_NUMBITS8

```
#define GET_NUMBITS8(
    VAR,
    SHIFT,
    MASK ) ((U8T(VAR) >> (SHIFT)) & (MASK))
```

Get value of bits given by mask (amount) and shift (position).

4.27.2.7 I16T

```
#define I16T(
    X ) ((int16_t) (X))
```

Cast argument to int16_t (16bit signed integer).

4.27.2.8 I32T

```
#define I32T(
    X ) ((int32_t) (X))
```

Cast argument to int32_t (32bit signed integer).

4.27.2.9 I64T

```
#define I64T(
    X ) ((int64_t) (X))
```

Cast argument to int64_t (64bit signed integer).

4.27.2.10 I8T

```
#define I8T(  
    X ) ((int8_t) (X))
```

Cast argument to int8_t (8bit signed integer).

4.27.2.11 LIBCAER_HAVE_OPENCV

```
#define LIBCAER_HAVE_OPENCV 1
```

libcaer OpenCV support.

4.27.2.12 LIBCAER_HAVE_SERIALDEV

```
#define LIBCAER_HAVE_SERIALDEV 1
```

libcaer serial devices support.

4.27.2.13 LIBCAER_NAME_STRING

```
#define LIBCAER_NAME_STRING "libcaer"
```

libcaer name string.

4.27.2.14 LIBCAER_VERSION

```
#define LIBCAER_VERSION ((2 * 10000) + (5 * 100) + 0)
```

libcaer version (MAJOR * 10000 + MINOR * 100 + PATCH).

4.27.2.15 LIBCAER_VERSION_STRING

```
#define LIBCAER_VERSION_STRING "2.5.0"
```

libcaer version string.

4.27.2.16 MASK_NUMBITS32

```
#define MASK_NUMBITS32(  
    X ) U32T(U32T(U32T(1) << X) - 1)
```

Mask and keep only the lower X bits of a 32bit (unsigned) integer.

4.27.2.17 MASK_NUMBITS64

```
#define MASK_NUMBITS64(  
    X )  U64T(U64T(1) << X) - 1)
```

Mask and keep only the lower X bits of a 64bit (unsigned) integer.

4.27.2.18 SET_NUMBITS16

```
#define SET_NUMBITS16(  
    VAR,  
    SHIFT,  
    MASK,  
    VALUE ) (VAR) |= htole16(U16T((U16T(VALUE) & (MASK)) << (SHIFT)))
```

Set bits given by mask (amount) and shift (position) to a value.

4.27.2.19 SET_NUMBITS32

```
#define SET_NUMBITS32(  
    VAR,  
    SHIFT,  
    MASK,  
    VALUE ) (VAR) |= htole32(U32T((U32T(VALUE) & (MASK)) << (SHIFT)))
```

Set bits given by mask (amount) and shift (position) to a value.

4.27.2.20 SET_NUMBITS8

```
#define SET_NUMBITS8(  
    VAR,  
    SHIFT,  
    MASK,  
    VALUE ) (VAR) |= U8T((U8T(VALUE) & (MASK)) << (SHIFT))
```

Set bits given by mask (amount) and shift (position) to a value.

4.27.2.21 SWAP_VAR

```
#define SWAP_VAR(  
    type,  
    x,  
    y ) { type tmpv; tmpv = (x); (x) = (y); (y) = tmpv; }
```

Swap the two values of the two variables X and Y, of a common type TYPE.

4.27.2.22 U16T

```
#define U16T(  
    X ) ((uint16_t) (X))
```

Cast argument to uint16_t (16bit unsigned integer).

4.27.2.23 U32T

```
#define U32T(  
    X ) ((uint32_t) (X))
```

Cast argument to uint32_t (32bit unsigned integer).

4.27.2.24 U64T

```
#define U64T(  
    X ) ((uint64_t) (X))
```

Cast argument to uint64_t (64bit unsigned integer).

4.27.2.25 U8T

```
#define U8T(  
    X ) ((uint8_t) (X))
```

Cast argument to uint8_t (8bit unsigned integer).

4.27.3 Enumeration Type Documentation

4.27.3.1 caer_error_codes

```
enum caer_error_codes
```

Error codes, used for the `errno` variable to give more precise information on errors, in addition to the logging output. All functions setting `errno` do note so in their documentation.

4.27.4 Function Documentation

4.27.4.1 caerByteArrayToInteger()

```
static uint32_t caerByteArrayToInteger (  
    const uint8_t * byteArray,  
    const uint8_t byteArrayLength ) [inline], [static]
```

Convert an unsigned byte array of up to four bytes into a 32bit unsigned integer. The byte array length decides how many resulting bits in the integer are set, and the single bytes are placed in the integer following big-endian ordering.

Parameters

<i>byteArray</i>	pointer to the byte array with parts of the value stored.
<i>byteArrayLength</i>	length of the array from which to convert.

Returns

integer representing the value stored in the byte array.

4.27.4.2 caerIntegerToByteArray()

```
static void caerIntegerToByteArray (
    const uint32_t integer,
    uint8_t * byteArray,
    const uint8_t byteArrayLength ) [inline], [static]
```

Convert a 32bit unsigned integer into an unsigned byte array of up to four bytes. The integer will be stored in big-endian order, and the length will specify how many bits to convert, starting from the lowest bit.

Parameters

<i>integer</i>	the integer to convert.
<i>byteArray</i>	pointer to the byte array in which to store the converted values.
<i>byteArrayLength</i>	length of the byte array to convert to.

4.27.4.3 caerStrEquals()

```
static bool caerStrEquals (
    const char * s1,
    const char * s2 ) [inline], [static]
```

Compare two strings for equality.

Parameters

<i>s1</i>	the first string, cannot be NULL.
<i>s2</i>	the second string, cannot be NULL.

Returns

true if equal, false otherwise.

4.27.4.4 caerStrEqualsUpTo()

```
static bool caerStrEqualsUpTo (
    const char * s1,
    const char * s2,
    size_t len ) [inline], [static]
```

Compare two strings for equality, up to a specified maximum length.

Parameters

<i>s1</i>	the first string, cannot be NULL.
<i>s2</i>	the second string, cannot be NULL.
<i>len</i>	maximum comparison length, cannot be zero.

Returns

true if equal, false otherwise.

4.28 log.h File Reference

```
#include <stdarg.h>
#include <stdint.h>
```

Macros

- #define **ATTRIBUTE_FORMAT**(N)
- #define **ATTRIBUTE_FORMAT_VA**(N)

Enumerations

- enum [caer_log_level](#) {
CAER_LOG_EMERGENCY = 0, **CAER_LOG_ALERT** = 1, **CAER_LOG_CRITICAL** = 2, **CAER_LOG_ERROR** = 3,
CAER_LOG_WARNING = 4, **CAER_LOG_NOTICE** = 5, **CAER_LOG_INFO** = 6, **CAER_LOG_DEBUG** = 7
}

Functions

- void [caerLogLevelSet](#) (enum [caer_log_level](#) logLevel)
- enum [caer_log_level](#) [caerLogLevelGet](#) (void)
- void [caerLogFileDescriptorsSet](#) (int fd1, int fd2)
- int [caerLogFileDescriptorsGetFirst](#) (void)
- int [caerLogFileDescriptorsGetSecond](#) (void)
- void [caerLogDisable](#) (bool disableLogging)
- bool [caerLogDisabled](#) (void)
- void [caerLog](#) (enum [caer_log_level](#) logLevel, const char *subSystem, const char *format,...) **ATTRIBUTE_FORMAT**(3)
- void [caerLogVA](#) (enum [caer_log_level](#) logLevel, const char *subSystem, const char *format, va_list args) **ATTRIBUTE_FORMAT_VA**(3)
- void [caerLogVAFull](#) (int logFileDescriptor1, int logFileDescriptor2, uint8_t systemLogLevel, enum [caer_log_level](#) logLevel, const char *subSystem, const char *format, va_list args) **ATTRIBUTE_FORMAT_VA**(6)

4.28.1 Detailed Description

Logging functions to print useful messages for the user.

4.28.2 Enumeration Type Documentation

4.28.2.1 `caer_log_level`

```
enum caer_log_level
```

Log levels for `caerLog()` logging function. Log messages only get printed if their log level is equal or above the global system log level, which can be set with `caerLogLevelSet()`. The default log level is `CAER_LOG_ERROR`. `CAER_LOG_EMERGENCY` is the most urgent log level and will always be printed, while `CAER_LOG_DEBUG` is the least urgent log level and will only be delivered if configured by the user.

4.28.3 Function Documentation

4.28.3.1 `caerLog()`

```
void caerLog (
    enum caer_log_level logLevel,
    const char * subSystem,
    const char * format,
    ... )
```

Main logging function. This function takes messages, formats them and sends them out to a file descriptor, respecting the system-wide log level setting and prepending the current time, the log level and a user-specified common string to the actual formatted output. The format is specified exactly as with the `printf()` family of functions. Please see their manual-page for more information.

Parameters

<i>logLevel</i>	the message-specific log level.
<i>subSystem</i>	a common, user-specified string to prepend before the message.
<i>format</i>	the message format string (see <code>printf()</code>).
...	the parameters to be formatted according to the format string (see <code>printf()</code>).

4.28.3.2 `caerLogDisable()`

```
void caerLogDisable (
    bool disableLogging )
```

Disable all logging for this thread only. Call again with different argument to re-enable.

Parameters

<i>disableLogging</i>	true to disable logging for this thread, false to enable it again.
-----------------------	--

4.28.3.3 caerLogDisabled()

```
bool caerLogDisabled (
    void )
```

Status of logging for this thread.

Returns

true if logging is disabled for this thread, false if it is enabled.

4.28.3.4 caerLogFileDescriptorsGetFirst()

```
int caerLogFileDescriptorsGetFirst (
    void )
```

Get the current output file descriptor 1.

Returns

the current output file descriptor 1.

4.28.3.5 caerLogFileDescriptorsGetSecond()

```
int caerLogFileDescriptorsGetSecond (
    void )
```

Get the current output file descriptor 2.

Returns

the current output file descriptor 2.

4.28.3.6 caerLogFileDescriptorsSet()

```
void caerLogFileDescriptorsSet (
    int fd1,
    int fd2 )
```

Set to which file descriptors log messages are sent. Up to two different file descriptors can be configured here. By default logging to STDERR only is enabled. If both file descriptors are identical, logging to it will only happen once, as if the second one was disabled.

Parameters

<i>fd1</i>	first file descriptor to log to. A negative value will disable it.
<i>fd2</i>	second file descriptor to log to. A negative value will disable it.

4.28.3.7 caerLogLevelGet()

```
enum caer_log_level caerLogLevelGet (
    void )
```

Get the current system-wide log level. Log messages are only printed if their level is equal or above this level.

Returns

the current system-wide log level.

4.28.3.8 caerLogLevelSet()

```
void caerLogLevelSet (
    enum caer_log_level logLevel )
```

Set the system-wide log level. Log messages will only be printed if their level is equal or above this level.

Parameters

<i>logLevel</i>	the system-wide log level.
-----------------	----------------------------

4.28.3.9 caerLogVA()

```
void caerLogVA (
    enum caer_log_level logLevel,
    const char * subSystem,
    const char * format,
    va_list args )
```

Secondary logging function. This function takes messages, formats them and sends them out to a file descriptor, respecting the system-wide log level setting and prepending the current time, the log level and a user-specified common string to the actual formatted output. The format is specified exactly as with the printf() family of functions. The argument list is a va_list as returned by va_start(), following the vprintf() family of functions in its functionality. Please see their manual-page for more information.

Parameters

<i>logLevel</i>	the message-specific log level.
<i>subSystem</i>	a common, user-specified string to prepend before the message.
<i>format</i>	the message format string (see printf()).
<i>args</i>	the parameters to be formatted according to the format string (see printf()). This is an argument list as returned by va_start().

4.28.3.10 caerLogVAFull()

```
void caerLogVAFull (
    int logFileDescriptor1,
    int logFileDescriptor2,
    uint8_t systemLogLevel,
    enum caer_log_level logLevel,
    const char * subSystem,
    const char * format,
    va_list args )
```

Tertiary logging function. This function takes messages, formats them and sends them out to up to two file descriptors, fully specified by the user; allows a user-given system log level setting to also be specified, and then prepends the current time, the message log level and a user-specified common string to the actual formatted output. The format is specified exactly as with the printf() family of functions. The argument list is a va_list as returned by va_start(), following the vprintf() family of functions in its functionality. Please see their manual-page for more information.

Parameters

<i>logFileDescriptor1</i>	first output file descriptor.
<i>logFileDescriptor2</i>	second output file descriptor.
<i>systemLogLevel</i>	the system-wide log level.
<i>logLevel</i>	the message-specific log level.
<i>subSystem</i>	a common, user-specified string to prepend before the message.
<i>format</i>	the message format string (see printf()).
<i>args</i>	the parameters to be formatted according to the format string (see printf()). This is an argument list as returned by va_start().

4.29 network.h File Reference

```
#include "libcaer.h"
```

Macros

- `#define AEDAT3_NETWORK_HEADER_LENGTH 20`
- `#define AEDAT3_NETWORK_MAGIC_NUMBER 0x1D378BC90B9A6658`
- `#define AEDAT3_NETWORK_VERSION 0x01`
- `#define AEDAT3_FILE_VERSION "3.1"`
- `#define AEDAT3_MAX_UDP_SIZE (1472 - AEDAT3_NETWORK_HEADER_LENGTH)`

Functions

- **PACKED_STRUCT** (struct aedat3_network_header { int64_t magicNumber;int64_t sequenceNumber;int8_t↵_t versionNumber;int8_t formatNumber;int16_t sourceID;})
- static struct aedat3_network_header **caerParseNetworkHeader** (const uint8_t *dataBuffer)

4.29.1 Detailed Description

Useful functions for AEDAT 3.X network streams.

4.30 portable_endian.h File Reference

```
#include <stdint.h>
#include <string.h>
```

Functions

- static float **htobeft** (float val)
- static float **htoleft** (float val)
- static float **befltoh** (float val)
- static float **lefltoh** (float val)

4.30.1 Detailed Description

Endianness conversion functions for a wide variety of systems, including Linux, FreeBSD, MacOS X and Windows.

Index

CAER_CONFIGURATION_CONST_ITERATOR_ALL↔
_START
config.h, [194](#)

CAER_CONFIGURATION_CONST_ITERATOR_VA↔
LID_START
config.h, [194](#)

CAER_CONFIGURATION_CONST_REVERSE_ITE↔
RATOR_ALL_START
config.h, [194](#)

CAER_CONFIGURATION_CONST_REVERSE_ITE↔
RATOR_VALID_START
config.h, [195](#)

CAER_CONFIGURATION_ITERATOR_ALL_END
config.h, [195](#)

CAER_CONFIGURATION_ITERATOR_ALL_START
config.h, [195](#)

CAER_CONFIGURATION_ITERATOR_VALID_END
config.h, [196](#)

CAER_CONFIGURATION_ITERATOR_VALID_START
config.h, [196](#)

CAER_CONFIGURATION_REVERSE_ITERATOR↔
ALL_END
config.h, [196](#)

CAER_CONFIGURATION_REVERSE_ITERATOR↔
ALL_START
config.h, [197](#)

CAER_CONFIGURATION_REVERSE_ITERATOR↔
VALID_END
config.h, [197](#)

CAER_CONFIGURATION_REVERSE_ITERATOR↔
VALID_START
config.h, [197](#)

CAER_DEFAULT_EVENT_TYPES_COUNT
common.h, [176](#)

CAER_DEVICE_DAVIS_FX2
davis.h, [23](#)

CAER_DEVICE_DAVIS_FX3
davis.h, [24](#)

CAER_DEVICE_DAVIS_RPI
davis.h, [24](#)

CAER_DEVICE_DAVIS
davis.h, [23](#)

CAER_DEVICE_DISCOVER_ALL
device_discover.h, [137](#)

CAER_DEVICE_DVS128
dvs128.h, [139](#)

CAER_DEVICE_DYNAPSE
dynapse.h, [147](#)

CAER_DEVICE_EDVS
edvs.h, [168](#)

CAER_EAR_CONST_ITERATOR_ALL_START
ear.h, [206](#)

CAER_EAR_CONST_ITERATOR_VALID_START
ear.h, [206](#)

CAER_EAR_CONST_REVERSE_ITERATOR_ALL↔
START
ear.h, [207](#)

CAER_EAR_CONST_REVERSE_ITERATOR_VALI↔
D_START
ear.h, [207](#)

CAER_EAR_ITERATOR_ALL_END
ear.h, [207](#)

CAER_EAR_ITERATOR_ALL_START
ear.h, [208](#)

CAER_EAR_ITERATOR_VALID_END
ear.h, [208](#)

CAER_EAR_ITERATOR_VALID_START
ear.h, [208](#)

CAER_EAR_REVERSE_ITERATOR_ALL_END
ear.h, [208](#)

CAER_EAR_REVERSE_ITERATOR_ALL_START
ear.h, [209](#)

CAER_EAR_REVERSE_ITERATOR_VALID_END
ear.h, [209](#)

CAER_EAR_REVERSE_ITERATOR_VALID_START
ear.h, [209](#)

CAER_EVENT_PACKET_CONTAINER_CONST_IT↔
ERATOR_START
packetContainer.h, [308](#)

CAER_EVENT_PACKET_CONTAINER_ITERATOR↔
_END
packetContainer.h, [308](#)

CAER_EVENT_PACKET_CONTAINER_ITERATOR↔
_START
packetContainer.h, [309](#)

CAER_EVENT_PACKET_HEADER_SIZE
common.h, [176](#)

CAER_FILTER_DVS_BACKGROUND_ACTIVITY_C↔
HECK_POLARITY
dvs_noise.h, [425](#)

CAER_FILTER_DVS_BACKGROUND_ACTIVITY_E↔
NABLE
dvs_noise.h, [426](#)

CAER_FILTER_DVS_BACKGROUND_ACTIVITY_S↔
TATISTICS
dvs_noise.h, [426](#)

CAER_FILTER_DVS_BACKGROUND_ACTIVITY_S↔
UPPORT_MAX

- dvs_noise.h, [426](#)
- CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MIN
 - dvs_noise.h, [426](#)
- CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TIME
 - dvs_noise.h, [426](#)
- CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TIME_LEVELS
 - dvs_noise.h, [426](#)
- CAER_FILTER_DVS_HOTPIXEL_COUNT
 - dvs_noise.h, [426](#)
- CAER_FILTER_DVS_HOTPIXEL_ENABLE
 - dvs_noise.h, [427](#)
- CAER_FILTER_DVS_HOTPIXEL_LEARN
 - dvs_noise.h, [427](#)
- CAER_FILTER_DVS_HOTPIXEL_STATISTICS
 - dvs_noise.h, [427](#)
- CAER_FILTER_DVS_HOTPIXEL_TIME
 - dvs_noise.h, [427](#)
- CAER_FILTER_DVS_LOG_LEVEL
 - dvs_noise.h, [427](#)
- CAER_FILTER_DVS_REFRACTORY_PERIOD_ENABLED
 - dvs_noise.h, [427](#)
- CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS
 - dvs_noise.h, [427](#)
- CAER_FILTER_DVS_REFRACTORY_PERIOD_TIME
 - dvs_noise.h, [428](#)
- CAER_FILTER_DVS_RESET
 - dvs_noise.h, [428](#)
- CAER_FRAME_CONST_ITERATOR_ALL_START
 - frame.h, [221](#)
- CAER_FRAME_CONST_ITERATOR_VALID_START
 - frame.h, [221](#)
- CAER_FRAME_CONST_REVERSE_ITERATOR_ALL_START
 - frame.h, [221](#)
- CAER_FRAME_CONST_REVERSE_ITERATOR_VALID_START
 - frame.h, [222](#)
- CAER_FRAME_ITERATOR_ALL_END
 - frame.h, [222](#)
- CAER_FRAME_ITERATOR_ALL_START
 - frame.h, [222](#)
- CAER_FRAME_ITERATOR_VALID_END
 - frame.h, [223](#)
- CAER_FRAME_ITERATOR_VALID_START
 - frame.h, [223](#)
- CAER_FRAME_REVERSE_ITERATOR_ALL_END
 - frame.h, [223](#)
- CAER_FRAME_REVERSE_ITERATOR_ALL_START
 - frame.h, [223](#)
- CAER_FRAME_REVERSE_ITERATOR_VALID_END
 - frame.h, [224](#)
- CAER_FRAME_REVERSE_ITERATOR_VALID_START
 - frame.h, [224](#)
- frame.h, [224](#)
- CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING
 - device.h, [131](#)
- CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE
 - device.h, [131](#)
- CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS
 - device.h, [131](#)
- CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS
 - device.h, [131](#)
- CAER_HOST_CONFIG_DATAEXCHANGE
 - device.h, [131](#)
- CAER_HOST_CONFIG_LOG_LEVEL
 - device.h, [132](#)
- CAER_HOST_CONFIG_LOG
 - device.h, [131](#)
- CAER_HOST_CONFIG_PACKETS_MAX_CONTAINERS_INTERVAL
 - device.h, [132](#)
- CAER_HOST_CONFIG_PACKETS_MAX_CONTAINERS_PACKET_SIZE
 - device.h, [132](#)
- CAER_HOST_CONFIG_PACKETS
 - device.h, [132](#)
- CAER_HOST_CONFIG_SERIAL_BAUD_RATE_12M
 - serial.h, [171](#)
- CAER_HOST_CONFIG_SERIAL_BAUD_RATE_2M
 - serial.h, [171](#)
- CAER_HOST_CONFIG_SERIAL_BAUD_RATE_4M
 - serial.h, [172](#)
- CAER_HOST_CONFIG_SERIAL_BAUD_RATE_8M
 - serial.h, [172](#)
- CAER_HOST_CONFIG_SERIAL_READ_SIZE
 - serial.h, [172](#)
- CAER_HOST_CONFIG_SERIAL
 - serial.h, [171](#)
- CAER_HOST_CONFIG_USB_BUFFER_NUMBER
 - usb.h, [173](#)
- CAER_HOST_CONFIG_USB_BUFFER_SIZE
 - usb.h, [173](#)
- CAER_HOST_CONFIG_USB
 - usb.h, [173](#)
- CAER_IMU6_CONST_ITERATOR_ALL_START
 - imu6.h, [250](#)
- CAER_IMU6_CONST_ITERATOR_VALID_START
 - imu6.h, [250](#)
- CAER_IMU6_CONST_REVERSE_ITERATOR_ALL_START
 - imu6.h, [250](#)
- CAER_IMU6_CONST_REVERSE_ITERATOR_VALID_START
 - imu6.h, [251](#)
- CAER_IMU6_ITERATOR_ALL_END
 - imu6.h, [251](#)
- CAER_IMU6_ITERATOR_ALL_START

- imu6.h, [251](#)
- CAER_IMU6_ITERATOR_VALID_END
 - imu6.h, [252](#)
- CAER_IMU6_ITERATOR_VALID_START
 - imu6.h, [252](#)
- CAER_IMU6_REVERSE_ITERATOR_ALL_END
 - imu6.h, [252](#)
- CAER_IMU6_REVERSE_ITERATOR_ALL_START
 - imu6.h, [252](#)
- CAER_IMU6_REVERSE_ITERATOR_VALID_END
 - imu6.h, [253](#)
- CAER_IMU6_REVERSE_ITERATOR_VALID_START
 - imu6.h, [253](#)
- CAER_IMU9_CONST_ITERATOR_ALL_START
 - imu9.h, [265](#)
- CAER_IMU9_CONST_ITERATOR_VALID_START
 - imu9.h, [265](#)
- CAER_IMU9_CONST_REVERSE_ITERATOR_ALL_↔
 - START
 - imu9.h, [265](#)
- CAER_IMU9_CONST_REVERSE_ITERATOR_VALID_↔
 - D_START
 - imu9.h, [266](#)
- CAER_IMU9_ITERATOR_ALL_END
 - imu9.h, [266](#)
- CAER_IMU9_ITERATOR_ALL_START
 - imu9.h, [266](#)
- CAER_IMU9_ITERATOR_VALID_END
 - imu9.h, [267](#)
- CAER_IMU9_ITERATOR_VALID_START
 - imu9.h, [267](#)
- CAER_IMU9_REVERSE_ITERATOR_ALL_END
 - imu9.h, [267](#)
- CAER_IMU9_REVERSE_ITERATOR_ALL_START
 - imu9.h, [267](#)
- CAER_IMU9_REVERSE_ITERATOR_VALID_END
 - imu9.h, [268](#)
- CAER_IMU9_REVERSE_ITERATOR_VALID_START
 - imu9.h, [268](#)
- CAER_ITERATOR_ALL_END
 - common.h, [177](#)
- CAER_ITERATOR_ALL_START
 - common.h, [177](#)
- CAER_ITERATOR_VALID_END
 - common.h, [177](#)
- CAER_ITERATOR_VALID_START
 - common.h, [177](#)
- CAER_MATRIX4x4_CONST_ITERATOR_ALL_START
 - matrix4x4.h, [283](#)
- CAER_MATRIX4x4_CONST_ITERATOR_VALID_ST↔
 - ART
 - matrix4x4.h, [283](#)
- CAER_MATRIX4x4_CONST_REVERSE_ITERATOR↔
 - _ALL_START
 - matrix4x4.h, [284](#)
- CAER_MATRIX4x4_CONST_REVERSE_ITERATOR↔
 - _VALID_START
 - matrix4x4.h, [284](#)
- CAER_MATRIX4x4_ITERATOR_ALL_END
 - matrix4x4.h, [285](#)
- CAER_MATRIX4x4_ITERATOR_ALL_START
 - matrix4x4.h, [285](#)
- CAER_MATRIX4x4_ITERATOR_VALID_END
 - matrix4x4.h, [285](#)
- CAER_MATRIX4x4_ITERATOR_VALID_START
 - matrix4x4.h, [285](#)
- CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_END
 - matrix4x4.h, [286](#)
- CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_S↔
 - TART
 - matrix4x4.h, [286](#)
- CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_↔
 - END
 - matrix4x4.h, [286](#)
- CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_↔
 - START
 - matrix4x4.h, [287](#)
- CAER_POINT1D_CONST_ITERATOR_ALL_START
 - point1d.h, [317](#)
- CAER_POINT1D_CONST_ITERATOR_VALID_START
 - point1d.h, [317](#)
- CAER_POINT1D_CONST_REVERSE_ITERATOR↔
 - ALL_START
 - point1d.h, [317](#)
- CAER_POINT1D_CONST_REVERSE_ITERATOR↔
 - VALID_START
 - point1d.h, [318](#)
- CAER_POINT1D_ITERATOR_ALL_END
 - point1d.h, [318](#)
- CAER_POINT1D_ITERATOR_ALL_START
 - point1d.h, [318](#)
- CAER_POINT1D_ITERATOR_VALID_END
 - point1d.h, [319](#)
- CAER_POINT1D_ITERATOR_VALID_START
 - point1d.h, [319](#)
- CAER_POINT1D_REVERSE_ITERATOR_ALL_END
 - point1d.h, [319](#)
- CAER_POINT1D_REVERSE_ITERATOR_ALL_STA↔
 - RT
 - point1d.h, [319](#)
- CAER_POINT1D_REVERSE_ITERATOR_VALID_END
 - point1d.h, [320](#)
- CAER_POINT1D_REVERSE_ITERATOR_VALID_S↔
 - TART
 - point1d.h, [320](#)
- CAER_POINT2D_CONST_ITERATOR_ALL_START
 - point2d.h, [329](#)
- CAER_POINT2D_CONST_ITERATOR_VALID_START
 - point2d.h, [330](#)
- CAER_POINT2D_CONST_REVERSE_ITERATOR↔
 - ALL_START
 - point2d.h, [330](#)
- CAER_POINT2D_CONST_REVERSE_ITERATOR↔
 - VALID_START
 - point2d.h, [330](#)
- CAER_POINT2D_ITERATOR_ALL_END

- point2d.h, [331](#)
- CAER_POINT2D_ITERATOR_ALL_START
 - point2d.h, [331](#)
- CAER_POINT2D_ITERATOR_VALID_END
 - point2d.h, [331](#)
- CAER_POINT2D_ITERATOR_VALID_START
 - point2d.h, [331](#)
- CAER_POINT2D_REVERSE_ITERATOR_ALL_END
 - point2d.h, [332](#)
- CAER_POINT2D_REVERSE_ITERATOR_ALL_START
 - point2d.h, [332](#)
- CAER_POINT2D_REVERSE_ITERATOR_VALID_END
 - point2d.h, [332](#)
- CAER_POINT2D_REVERSE_ITERATOR_VALID_START
 - point2d.h, [332](#)
- CAER_POINT3D_CONST_ITERATOR_ALL_START
 - point3d.h, [343](#)
- CAER_POINT3D_CONST_ITERATOR_VALID_START
 - point3d.h, [343](#)
- CAER_POINT3D_CONST_REVERSE_ITERATOR_ALL_START
 - point3d.h, [344](#)
- CAER_POINT3D_CONST_REVERSE_ITERATOR_VALID_START
 - point3d.h, [344](#)
- CAER_POINT3D_ITERATOR_ALL_END
 - point3d.h, [344](#)
- CAER_POINT3D_ITERATOR_ALL_START
 - point3d.h, [345](#)
- CAER_POINT3D_ITERATOR_VALID_END
 - point3d.h, [345](#)
- CAER_POINT3D_ITERATOR_VALID_START
 - point3d.h, [345](#)
- CAER_POINT3D_REVERSE_ITERATOR_ALL_END
 - point3d.h, [345](#)
- CAER_POINT3D_REVERSE_ITERATOR_ALL_START
 - point3d.h, [346](#)
- CAER_POINT3D_REVERSE_ITERATOR_VALID_END
 - point3d.h, [346](#)
- CAER_POINT3D_REVERSE_ITERATOR_VALID_START
 - point3d.h, [346](#)
- CAER_POINT4D_CONST_ITERATOR_ALL_START
 - point4d.h, [357](#)
- CAER_POINT4D_CONST_ITERATOR_VALID_START
 - point4d.h, [357](#)
- CAER_POINT4D_CONST_REVERSE_ITERATOR_ALL_START
 - point4d.h, [358](#)
- CAER_POINT4D_CONST_REVERSE_ITERATOR_VALID_START
 - point4d.h, [358](#)
- CAER_POINT4D_ITERATOR_ALL_END
 - point4d.h, [358](#)
- CAER_POINT4D_ITERATOR_ALL_START
 - point4d.h, [359](#)
- CAER_POINT4D_ITERATOR_VALID_END
 - point4d.h, [359](#)
- CAER_POINT4D_ITERATOR_VALID_START
 - point4d.h, [359](#)
- CAER_POINT4D_REVERSE_ITERATOR_ALL_END
 - point4d.h, [359](#)
- CAER_POINT4D_REVERSE_ITERATOR_ALL_START
 - point4d.h, [360](#)
- CAER_POINT4D_REVERSE_ITERATOR_VALID_END
 - point4d.h, [360](#)
- CAER_POINT4D_REVERSE_ITERATOR_VALID_START
 - point4d.h, [360](#)
- CAER_POLARITY_CONST_ITERATOR_ALL_START
 - polarity.h, [372](#)
- CAER_POLARITY_CONST_ITERATOR_VALID_START
 - polarity.h, [373](#)
- CAER_POLARITY_CONST_REVERSE_ITERATOR_ALL_START
 - polarity.h, [373](#)
- CAER_POLARITY_CONST_REVERSE_ITERATOR_VALID_START
 - polarity.h, [373](#)
- CAER_POLARITY_ITERATOR_ALL_END
 - polarity.h, [374](#)
- CAER_POLARITY_ITERATOR_ALL_START
 - polarity.h, [374](#)
- CAER_POLARITY_ITERATOR_VALID_END
 - polarity.h, [374](#)
- CAER_POLARITY_ITERATOR_VALID_START
 - polarity.h, [374](#)
- CAER_POLARITY_REVERSE_ITERATOR_ALL_END
 - polarity.h, [375](#)
- CAER_POLARITY_REVERSE_ITERATOR_ALL_START
 - polarity.h, [375](#)
- CAER_POLARITY_REVERSE_ITERATOR_VALID_END
 - polarity.h, [375](#)
- CAER_POLARITY_REVERSE_ITERATOR_VALID_START
 - polarity.h, [375](#)
- CAER_SAMPLE_CONST_ITERATOR_ALL_START
 - sample.h, [385](#)
- CAER_SAMPLE_CONST_ITERATOR_VALID_START
 - sample.h, [385](#)
- CAER_SAMPLE_CONST_REVERSE_ITERATOR_ALL_START
 - sample.h, [386](#)
- CAER_SAMPLE_CONST_REVERSE_ITERATOR_VALID_START
 - sample.h, [386](#)
- CAER_SAMPLE_ITERATOR_ALL_END
 - sample.h, [386](#)
- CAER_SAMPLE_ITERATOR_ALL_START

- sample.h, [387](#)
- CAER_SAMPLE_ITERATOR_VALID_END
 - sample.h, [387](#)
- CAER_SAMPLE_ITERATOR_VALID_START
 - sample.h, [387](#)
- CAER_SAMPLE_REVERSE_ITERATOR_ALL_END
 - sample.h, [387](#)
- CAER_SAMPLE_REVERSE_ITERATOR_ALL_START
 - sample.h, [388](#)
- CAER_SAMPLE_REVERSE_ITERATOR_VALID_END
 - sample.h, [388](#)
- CAER_SAMPLE_REVERSE_ITERATOR_VALID_START
 - sample.h, [388](#)
- CAER_SPECIAL_CONST_ITERATOR_ALL_START
 - special.h, [397](#)
- CAER_SPECIAL_CONST_ITERATOR_VALID_START
 - special.h, [398](#)
- CAER_SPECIAL_CONST_REVERSE_ITERATOR_ALL_START
 - special.h, [398](#)
- CAER_SPECIAL_CONST_REVERSE_ITERATOR_VALID_START
 - special.h, [398](#)
- CAER_SPECIAL_ITERATOR_ALL_END
 - special.h, [399](#)
- CAER_SPECIAL_ITERATOR_ALL_START
 - special.h, [399](#)
- CAER_SPECIAL_ITERATOR_VALID_END
 - special.h, [399](#)
- CAER_SPECIAL_ITERATOR_VALID_START
 - special.h, [399](#)
- CAER_SPECIAL_REVERSE_ITERATOR_ALL_END
 - special.h, [400](#)
- CAER_SPECIAL_REVERSE_ITERATOR_ALL_START
 - special.h, [400](#)
- CAER_SPECIAL_REVERSE_ITERATOR_VALID_END
 - special.h, [400](#)
- CAER_SPECIAL_REVERSE_ITERATOR_VALID_START
 - special.h, [400](#)
- CAER_SPIKE_CONST_ITERATOR_ALL_START
 - spike.h, [412](#)
- CAER_SPIKE_CONST_ITERATOR_VALID_START
 - spike.h, [412](#)
- CAER_SPIKE_CONST_REVERSE_ITERATOR_ALL_START
 - spike.h, [413](#)
- CAER_SPIKE_CONST_REVERSE_ITERATOR_VALID_START
 - spike.h, [413](#)
- CAER_SPIKE_ITERATOR_ALL_END
 - spike.h, [413](#)
- CAER_SPIKE_ITERATOR_ALL_START
 - spike.h, [414](#)
- CAER_SPIKE_ITERATOR_VALID_END
 - spike.h, [414](#)
- CAER_SPIKE_ITERATOR_VALID_START
 - spike.h, [414](#)
- CAER_SPIKE_REVERSE_ITERATOR_ALL_END
 - spike.h, [414](#)
- CAER_SPIKE_REVERSE_ITERATOR_ALL_START
 - spike.h, [415](#)
- CAER_SPIKE_REVERSE_ITERATOR_VALID_END
 - spike.h, [415](#)
- CAER_SPIKE_REVERSE_ITERATOR_VALID_START
 - spike.h, [415](#)
- CAER_SUPPORTED_DEVICES_NUMBER
 - device.h, [132](#)
- CLEAR_NUMBITS16
 - libcaer.h, [433](#)
- CLEAR_NUMBITS32
 - libcaer.h, [433](#)
- CLEAR_NUMBITS8
 - libcaer.h, [433](#)
- CONFIG_MODULE_ADDR_MASK
 - config.h, [197](#)
- CONFIG_MODULE_ADDR_SHIFT
 - config.h, [198](#)
- caer_bias_coarsefine, [5](#)
- caer_bias_dynapse, [5](#)
- caer_bias_shiftedsources, [6](#)
- caer_bias_shiftedsources_operating_mode
 - davis.h, [126](#)
- caer_bias_shiftedsources_voltage_level
 - davis.h, [126](#)
- caer_bias_vdac, [7](#)
- caer_davis_info, [7](#)
 - deviceString, [8](#)
- caer_default_event_types
 - common.h, [178](#)
- caer_device_discovery_result, [9](#)
- caer_dvs128_info, [9](#)
 - deviceString, [10](#)
- caer_dynapse_info, [10](#)
 - deviceString, [11](#)
- caer_edvs_info, [11](#)
 - deviceString, [12](#)
- caer_error_codes
 - libcaer.h, [437](#)
- caer_filter_dvs_pixel, [12](#)
- caer_frame_event_color_channels
 - frame.h, [226](#)
- caer_frame_event_color_filter
 - frame.h, [226](#)
- caer_log_level
 - log.h, [440](#)
- caer_special_event_types
 - special.h, [402](#)
- caerBiasCoarseFineGenerate
 - davis.h, [127](#)
- caerBiasCoarseFineParse
 - davis.h, [127](#)
- caerBiasDynapseGenerate
 - dynapse.h, [159](#)
- caerBiasDynapseParse

- dynapse.h, 159
- caerBiasShiftedSourceGenerate
 - davis.h, 127
- caerBiasShiftedSourceParse
 - davis.h, 128
- caerBiasVDACGenerate
 - davis.h, 128
- caerBiasVDACParse
 - davis.h, 128
- caerByteArrayToInteger
 - libcaer.h, 437
- caerConfigurationEvent
 - config.h, 198
- caerConfigurationEventGetModuleAddress
 - config.h, 198
- caerConfigurationEventGetParameter
 - config.h, 199
- caerConfigurationEventGetParameterAddress
 - config.h, 199
- caerConfigurationEventGetTimestamp
 - config.h, 199
- caerConfigurationEventGetTimestamp64
 - config.h, 200
- caerConfigurationEventInvalidate
 - config.h, 200
- caerConfigurationEventIsValid
 - config.h, 200
- caerConfigurationEventPacket
 - config.h, 198
- caerConfigurationEventPacketAllocate
 - config.h, 201
- caerConfigurationEventPacketFromPacketHeader
 - config.h, 201
- caerConfigurationEventPacketFromPacketHeaderConst
 - config.h, 202
- caerConfigurationEventPacketGetEvent
 - config.h, 202
- caerConfigurationEventPacketGetEventConst
 - config.h, 202
- caerConfigurationEventSetModuleAddress
 - config.h, 203
- caerConfigurationEventSetParameter
 - config.h, 203
- caerConfigurationEventSetParameterAddress
 - config.h, 203
- caerConfigurationEventSetTimestamp
 - config.h, 204
- caerConfigurationEventValidate
 - config.h, 204
- caerDVS128InfoGet
 - dvs128.h, 142
- caerDavisInfoGet
 - davis.h, 129
- caerDavisROIConfigure
 - davis.h, 129
- caerDeviceClose
 - device.h, 133
- caerDeviceConfigGet
 - device.h, 133
- caerDeviceConfigGet64
 - device.h, 134
- caerDeviceConfigSet
 - device.h, 134
- caerDeviceDataGet
 - device.h, 135
- caerDeviceDataStart
 - device.h, 135
- caerDeviceDataStop
 - device.h, 136
- caerDeviceDiscover
 - device_discover.h, 137
- caerDeviceDiscoverOpen
 - device_discover.h, 138
- caerDeviceDiscoveryResult
 - device_discover.h, 137
- caerDeviceHandle
 - device.h, 132
- caerDeviceOpen
 - usb.h, 174
- caerDeviceOpenSerial
 - serial.h, 172
- caerDeviceSendDefaultConfig
 - device.h, 136
- caerDynapseCoreAddrToNeuronId
 - dynapse.h, 160
- caerDynapseCoreXYToNeuronId
 - dynapse.h, 160
- caerDynapseGenerateCamBits
 - dynapse.h, 160
- caerDynapseGenerateSramBits
 - dynapse.h, 161
- caerDynapseInfoGet
 - dynapse.h, 162
- caerDynapseSendDataToUSB
 - dynapse.h, 162
- caerDynapseSpikeEventFromXY
 - dynapse.h, 162
- caerDynapseSpikeEventGetX
 - dynapse.h, 163
- caerDynapseSpikeEventGetY
 - dynapse.h, 163
- caerDynapseWriteCam
 - dynapse.h, 164
- caerDynapseWritePoissonSpikeRate
 - dynapse.h, 164
- caerDynapseWriteSram
 - dynapse.h, 165
- caerDynapseWriteSramWords
 - dynapse.h, 166
- caerDynapseWriteSramN
 - dynapse.h, 165
- caerEDVSInfoGet
 - edvs.h, 170
- caerEarEvent
 - ear.h, 211
- caerEarEventGetChannel

- ear.h, [211](#)
- caerEarEventGetEar
 - ear.h, [212](#)
- caerEarEventGetTimestamp
 - ear.h, [212](#)
- caerEarEventGetTimestamp64
 - ear.h, [212](#)
- caerEarEventInvalidate
 - ear.h, [213](#)
- caerEarEventsIsValid
 - ear.h, [213](#)
- caerEarEventPacket
 - ear.h, [211](#)
- caerEarEventPacketAllocate
 - ear.h, [214](#)
- caerEarEventPacketFromPacketHeader
 - ear.h, [214](#)
- caerEarEventPacketFromPacketHeaderConst
 - ear.h, [214](#)
- caerEarEventPacketGetEvent
 - ear.h, [216](#)
- caerEarEventPacketGetEventConst
 - ear.h, [216](#)
- caerEarEventSetChannel
 - ear.h, [216](#)
- caerEarEventSetEar
 - ear.h, [217](#)
- caerEarEventSetTimestamp
 - ear.h, [217](#)
- caerEarEventValidate
 - ear.h, [217](#)
- caerEventPacketAllocate
 - common.h, [179](#)
- caerEventPacketAppend
 - common.h, [179](#)
- caerEventPacketClean
 - common.h, [180](#)
- caerEventPacketClear
 - common.h, [180](#)
- caerEventPacketContainer
 - packetContainer.h, [309](#)
- caerEventPacketContainerAllocate
 - packetContainer.h, [309](#)
- caerEventPacketContainerCopyAllEvents
 - packetContainer.h, [310](#)
- caerEventPacketContainerCopyValidEvents
 - packetContainer.h, [310](#)
- caerEventPacketContainerFindEventPacketByType
 - packetContainer.h, [310](#)
- caerEventPacketContainerFindEventPacketByType↔
Const
 - packetContainer.h, [311](#)
- caerEventPacketContainerFree
 - packetContainer.h, [311](#)
- caerEventPacketContainerGetEventPacket
 - packetContainer.h, [312](#)
- caerEventPacketContainerGetEventPacketConst
 - packetContainer.h, [312](#)
- caerEventPacketContainerGetEventPacketsNumber
 - packetContainer.h, [312](#)
- caerEventPacketContainerGetEventsNumber
 - packetContainer.h, [313](#)
- caerEventPacketContainerGetEventsValidNumber
 - packetContainer.h, [313](#)
- caerEventPacketContainerGetHighestEventTimestamp
 - packetContainer.h, [313](#)
- caerEventPacketContainerGetLowestEventTimestamp
 - packetContainer.h, [314](#)
- caerEventPacketContainerSetEventPacket
 - packetContainer.h, [314](#)
- caerEventPacketContainerSetEventPacketsNumber
 - packetContainer.h, [315](#)
- caerEventPacketContainerUpdateStatistics
 - packetContainer.h, [315](#)
- caerEventPacketCopy
 - common.h, [180](#)
- caerEventPacketCopyOnlyEvents
 - common.h, [181](#)
- caerEventPacketCopyOnlyValidEvents
 - common.h, [181](#)
- caerEventPacketEquals
 - common.h, [181](#)
- caerEventPacketGetDataSize
 - common.h, [182](#)
- caerEventPacketGetDataSizeEvents
 - common.h, [182](#)
- caerEventPacketGetSize
 - common.h, [183](#)
- caerEventPacketGetSizeEvents
 - common.h, [183](#)
- caerEventPacketGrow
 - common.h, [183](#)
- caerEventPacketHeader
 - common.h, [178](#)
- caerEventPacketHeaderGetEventCapacity
 - common.h, [184](#)
- caerEventPacketHeaderGetEventNumber
 - common.h, [184](#)
- caerEventPacketHeaderGetEventSize
 - common.h, [184](#)
- caerEventPacketHeaderGetEventSource
 - common.h, [185](#)
- caerEventPacketHeaderGetEventTSOffset
 - common.h, [185](#)
- caerEventPacketHeaderGetEventTSOverflow
 - common.h, [186](#)
- caerEventPacketHeaderGetEventType
 - common.h, [186](#)
- caerEventPacketHeaderGetEventValid
 - common.h, [186](#)
- caerEventPacketHeaderSetEventCapacity
 - common.h, [187](#)
- caerEventPacketHeaderSetEventNumber
 - common.h, [187](#)
- caerEventPacketHeaderSetEventSize
 - common.h, [187](#)

- caerEventPacketHeaderSetEventSource
common.h, [188](#)
- caerEventPacketHeaderSetEventTSOffset
common.h, [188](#)
- caerEventPacketHeaderSetEventTSOverflow
common.h, [188](#)
- caerEventPacketHeaderSetEventType
common.h, [189](#)
- caerEventPacketHeaderSetEventValid
common.h, [189](#)
- caerEventPacketResize
common.h, [189](#)
- caerFilterDVSNoise
dvs_noise.h, [428](#)
- caerFilterDVSNoiseApply
dvs_noise.h, [428](#)
- caerFilterDVSNoiseConfigGet
dvs_noise.h, [429](#)
- caerFilterDVSNoiseConfigSet
dvs_noise.h, [429](#)
- caerFilterDVSNoiseDestroy
dvs_noise.h, [429](#)
- caerFilterDVSNoiseGetHotPixels
dvs_noise.h, [430](#)
- caerFilterDVSNoiseInitialize
dvs_noise.h, [430](#)
- caerFilterDVSPixel
dvs_noise.h, [428](#)
- caerFrameEvent
frame.h, [226](#)
- caerFrameEventGetChannelNumber
frame.h, [227](#)
- caerFrameEventGetColorFilter
frame.h, [227](#)
- caerFrameEventGetExposureLength
frame.h, [228](#)
- caerFrameEventGetLengthX
frame.h, [228](#)
- caerFrameEventGetLengthY
frame.h, [228](#)
- caerFrameEventGetPixel
frame.h, [230](#)
- caerFrameEventGetPixelArrayUnsafe
frame.h, [230](#)
- caerFrameEventGetPixelArrayUnsafeConst
frame.h, [231](#)
- caerFrameEventGetPixelForChannel
frame.h, [231](#)
- caerFrameEventGetPixelForChannelUnsafe
frame.h, [231](#)
- caerFrameEventGetPixelUnsafe
frame.h, [233](#)
- caerFrameEventGetPixelsMaxIndex
frame.h, [232](#)
- caerFrameEventGetPixelsSize
frame.h, [232](#)
- caerFrameEventGetPositionX
frame.h, [233](#)
- caerFrameEventGetPositionY
frame.h, [233](#)
- caerFrameEventGetROIIdentifier
frame.h, [234](#)
- caerFrameEventGetTSEndOfExposure
frame.h, [235](#)
- caerFrameEventGetTSEndOfExposure64
frame.h, [235](#)
- caerFrameEventGetTSEndOfFrame
frame.h, [236](#)
- caerFrameEventGetTSEndOfFrame64
frame.h, [236](#)
- caerFrameEventGetTSSStartOfExposure
frame.h, [237](#)
- caerFrameEventGetTSSStartOfExposure64
frame.h, [237](#)
- caerFrameEventGetTSSStartOfFrame
frame.h, [237](#)
- caerFrameEventGetTSSStartOfFrame64
frame.h, [238](#)
- caerFrameEventGetTimestamp
frame.h, [234](#)
- caerFrameEventGetTimestamp64
frame.h, [235](#)
- caerFrameEventInvalidate
frame.h, [238](#)
- caerFrameEventIsValid
frame.h, [239](#)
- caerFrameEventPacket
frame.h, [226](#)
- caerFrameEventPacketAllocate
frame.h, [239](#)
- caerFrameEventPacketFromPacketHeader
frame.h, [239](#)
- caerFrameEventPacketFromPacketHeaderConst
frame.h, [240](#)
- caerFrameEventPacketGetEvent
frame.h, [240](#)
- caerFrameEventPacketGetEventConst
frame.h, [241](#)
- caerFrameEventPacketGetPixelsMaxIndex
frame.h, [241](#)
- caerFrameEventPacketGetPixelsSize
frame.h, [241](#)
- caerFrameEventSetColorFilter
frame.h, [243](#)
- caerFrameEventSetLengthXLengthYChannelNumber
frame.h, [243](#)
- caerFrameEventSetPixel
frame.h, [243](#)
- caerFrameEventSetPixelForChannel
frame.h, [244](#)
- caerFrameEventSetPixelForChannelUnsafe
frame.h, [244](#)
- caerFrameEventSetPixelUnsafe
frame.h, [245](#)
- caerFrameEventSetPositionX
frame.h, [245](#)

caerFrameEventSetPositionY
frame.h, [246](#)

caerFrameEventSetROIIdentifier
frame.h, [246](#)

caerFrameEventSetTSEndOfExposure
frame.h, [246](#)

caerFrameEventSetTSEndOfFrame
frame.h, [246](#)

caerFrameEventSetTSStartOfExposure
frame.h, [247](#)

caerFrameEventSetTSStartOfFrame
frame.h, [247](#)

caerFrameEventValidate
frame.h, [247](#)

caerGenericEventCopy
common.h, [190](#)

caerGenericEventGetEvent
common.h, [190](#)

caerGenericEventGetTimestamp
common.h, [191](#)

caerGenericEventGetTimestamp64
common.h, [191](#)

caerGenericEventsIsValid
common.h, [192](#)

caerIMU6Event
imu6.h, [253](#)

caerIMU6EventGetAccelX
imu6.h, [254](#)

caerIMU6EventGetAccelY
imu6.h, [254](#)

caerIMU6EventGetAccelZ
imu6.h, [255](#)

caerIMU6EventGetGyroX
imu6.h, [255](#)

caerIMU6EventGetGyroY
imu6.h, [255](#)

caerIMU6EventGetGyroZ
imu6.h, [256](#)

caerIMU6EventGetTemp
imu6.h, [256](#)

caerIMU6EventGetTimestamp
imu6.h, [256](#)

caerIMU6EventGetTimestamp64
imu6.h, [257](#)

caerIMU6EventInvalidate
imu6.h, [257](#)

caerIMU6EventsIsValid
imu6.h, [258](#)

caerIMU6EventPacket
imu6.h, [254](#)

caerIMU6EventPacketAllocate
imu6.h, [258](#)

caerIMU6EventPacketFromPacketHeader
imu6.h, [258](#)

caerIMU6EventPacketFromPacketHeaderConst
imu6.h, [259](#)

caerIMU6EventPacketGetEvent
imu6.h, [259](#)

caerIMU6EventPacketGetEventConst
imu6.h, [259](#)

caerIMU6EventSetAccelX
imu6.h, [260](#)

caerIMU6EventSetAccelY
imu6.h, [260](#)

caerIMU6EventSetAccelZ
imu6.h, [260](#)

caerIMU6EventSetGyroX
imu6.h, [261](#)

caerIMU6EventSetGyroY
imu6.h, [261](#)

caerIMU6EventSetGyroZ
imu6.h, [261](#)

caerIMU6EventSetTemp
imu6.h, [262](#)

caerIMU6EventSetTimestamp
imu6.h, [262](#)

caerIMU6EventValidate
imu6.h, [262](#)

caerIMU9Event
imu9.h, [268](#)

caerIMU9EventGetAccelX
imu9.h, [269](#)

caerIMU9EventGetAccelY
imu9.h, [269](#)

caerIMU9EventGetAccelZ
imu9.h, [270](#)

caerIMU9EventGetCompX
imu9.h, [270](#)

caerIMU9EventGetCompY
imu9.h, [270](#)

caerIMU9EventGetCompZ
imu9.h, [271](#)

caerIMU9EventGetGyroX
imu9.h, [271](#)

caerIMU9EventGetGyroY
imu9.h, [271](#)

caerIMU9EventGetGyroZ
imu9.h, [273](#)

caerIMU9EventGetTemp
imu9.h, [273](#)

caerIMU9EventGetTimestamp
imu9.h, [273](#)

caerIMU9EventGetTimestamp64
imu9.h, [274](#)

caerIMU9EventInvalidate
imu9.h, [274](#)

caerIMU9EventsIsValid
imu9.h, [275](#)

caerIMU9EventPacket
imu9.h, [269](#)

caerIMU9EventPacketAllocate
imu9.h, [275](#)

caerIMU9EventPacketFromPacketHeader
imu9.h, [275](#)

caerIMU9EventPacketFromPacketHeaderConst
imu9.h, [276](#)

- caerIMU9EventPacketGetEvent
imu9.h, [276](#)
- caerIMU9EventPacketGetEventConst
imu9.h, [276](#)
- caerIMU9EventSetAccelX
imu9.h, [277](#)
- caerIMU9EventSetAccelY
imu9.h, [277](#)
- caerIMU9EventSetAccelZ
imu9.h, [277](#)
- caerIMU9EventSetCompX
imu9.h, [278](#)
- caerIMU9EventSetCompY
imu9.h, [278](#)
- caerIMU9EventSetCompZ
imu9.h, [278](#)
- caerIMU9EventSetGyroX
imu9.h, [279](#)
- caerIMU9EventSetGyroY
imu9.h, [279](#)
- caerIMU9EventSetGyroZ
imu9.h, [279](#)
- caerIMU9EventSetTemp
imu9.h, [280](#)
- caerIMU9EventSetTimestamp
imu9.h, [280](#)
- caerIMU9EventValidate
imu9.h, [280](#)
- caerIntegerToByteArray
libcaer.h, [438](#)
- caerLog
log.h, [440](#)
- caerLogDisable
log.h, [440](#)
- caerLogDisabled
log.h, [441](#)
- caerLogFileDescriptorsGetFirst
log.h, [441](#)
- caerLogFileDescriptorsGetSecond
log.h, [441](#)
- caerLogFileDescriptorsSet
log.h, [441](#)
- caerLogLevelGet
log.h, [442](#)
- caerLogLevelSet
log.h, [442](#)
- caerLogVAFull
log.h, [443](#)
- caerLogVA
log.h, [442](#)
- caerMatrix4x4Event
matrix4x4.h, [288](#)
- caerMatrix4x4EventGetM00
matrix4x4.h, [288](#)
- caerMatrix4x4EventGetM01
matrix4x4.h, [288](#)
- caerMatrix4x4EventGetM02
matrix4x4.h, [289](#)
- caerMatrix4x4EventGetM03
matrix4x4.h, [289](#)
- caerMatrix4x4EventGetM10
matrix4x4.h, [290](#)
- caerMatrix4x4EventGetM11
matrix4x4.h, [290](#)
- caerMatrix4x4EventGetM12
matrix4x4.h, [290](#)
- caerMatrix4x4EventGetM13
matrix4x4.h, [291](#)
- caerMatrix4x4EventGetM20
matrix4x4.h, [291](#)
- caerMatrix4x4EventGetM21
matrix4x4.h, [291](#)
- caerMatrix4x4EventGetM22
matrix4x4.h, [292](#)
- caerMatrix4x4EventGetM23
matrix4x4.h, [292](#)
- caerMatrix4x4EventGetM30
matrix4x4.h, [292](#)
- caerMatrix4x4EventGetM31
matrix4x4.h, [294](#)
- caerMatrix4x4EventGetM32
matrix4x4.h, [294](#)
- caerMatrix4x4EventGetM33
matrix4x4.h, [294](#)
- caerMatrix4x4EventGetScale
matrix4x4.h, [296](#)
- caerMatrix4x4EventGetTimestamp
matrix4x4.h, [296](#)
- caerMatrix4x4EventGetTimestamp64
matrix4x4.h, [296](#)
- caerMatrix4x4EventGetType
matrix4x4.h, [297](#)
- caerMatrix4x4EventInvalidate
matrix4x4.h, [297](#)
- caerMatrix4x4EventIsValid
matrix4x4.h, [298](#)
- caerMatrix4x4EventPacket
matrix4x4.h, [288](#)
- caerMatrix4x4EventPacketAllocate
matrix4x4.h, [298](#)
- caerMatrix4x4EventPacketFromPacketHeader
matrix4x4.h, [298](#)
- caerMatrix4x4EventPacketFromPacketHeaderConst
matrix4x4.h, [299](#)
- caerMatrix4x4EventPacketGetEvent
matrix4x4.h, [299](#)
- caerMatrix4x4EventPacketGetEventConst
matrix4x4.h, [299](#)
- caerMatrix4x4EventSetM00
matrix4x4.h, [300](#)
- caerMatrix4x4EventSetM01
matrix4x4.h, [300](#)
- caerMatrix4x4EventSetM02
matrix4x4.h, [300](#)
- caerMatrix4x4EventSetM03
matrix4x4.h, [301](#)

caerMatrix4x4EventSetM10
matrix4x4.h, [301](#)

caerMatrix4x4EventSetM11
matrix4x4.h, [301](#)

caerMatrix4x4EventSetM12
matrix4x4.h, [302](#)

caerMatrix4x4EventSetM13
matrix4x4.h, [302](#)

caerMatrix4x4EventSetM20
matrix4x4.h, [302](#)

caerMatrix4x4EventSetM21
matrix4x4.h, [303](#)

caerMatrix4x4EventSetM22
matrix4x4.h, [303](#)

caerMatrix4x4EventSetM23
matrix4x4.h, [303](#)

caerMatrix4x4EventSetM30
matrix4x4.h, [304](#)

caerMatrix4x4EventSetM31
matrix4x4.h, [304](#)

caerMatrix4x4EventSetM32
matrix4x4.h, [304](#)

caerMatrix4x4EventSetM33
matrix4x4.h, [304](#)

caerMatrix4x4EventSetScale
matrix4x4.h, [305](#)

caerMatrix4x4EventSetTimestamp
matrix4x4.h, [305](#)

caerMatrix4x4EventSetType
matrix4x4.h, [305](#)

caerMatrix4x4EventValidate
matrix4x4.h, [306](#)

caerPoint1DEvent
point1d.h, [321](#)

caerPoint1DEventGetScale
point1d.h, [321](#)

caerPoint1DEventGetTimestamp
point1d.h, [322](#)

caerPoint1DEventGetTimestamp64
point1d.h, [322](#)

caerPoint1DEventGetType
point1d.h, [322](#)

caerPoint1DEventGetX
point1d.h, [323](#)

caerPoint1DEventInvalidate
point1d.h, [323](#)

caerPoint1DEventIsValid
point1d.h, [324](#)

caerPoint1DEventPacket
point1d.h, [321](#)

caerPoint1DEventPacketAllocate
point1d.h, [324](#)

caerPoint1DEventPacketFromPacketHeader
point1d.h, [324](#)

caerPoint1DEventPacketFromPacketHeaderConst
point1d.h, [325](#)

caerPoint1DEventPacketGetEvent
point1d.h, [325](#)

caerPoint1DEventPacketGetEventConst
point1d.h, [325](#)

caerPoint1DEventSetScale
point1d.h, [326](#)

caerPoint1DEventSetTimestamp
point1d.h, [326](#)

caerPoint1DEventSetType
point1d.h, [327](#)

caerPoint1DEventSetX
point1d.h, [327](#)

caerPoint1DEventValidate
point1d.h, [327](#)

caerPoint2DEvent
point2d.h, [334](#)

caerPoint2DEventGetScale
point2d.h, [334](#)

caerPoint2DEventGetTimestamp
point2d.h, [334](#)

caerPoint2DEventGetTimestamp64
point2d.h, [335](#)

caerPoint2DEventGetType
point2d.h, [335](#)

caerPoint2DEventGetX
point2d.h, [335](#)

caerPoint2DEventGetY
point2d.h, [336](#)

caerPoint2DEventInvalidate
point2d.h, [336](#)

caerPoint2DEventIsValid
point2d.h, [337](#)

caerPoint2DEventPacket
point2d.h, [334](#)

caerPoint2DEventPacketAllocate
point2d.h, [337](#)

caerPoint2DEventPacketFromPacketHeader
point2d.h, [337](#)

caerPoint2DEventPacketFromPacketHeaderConst
point2d.h, [338](#)

caerPoint2DEventPacketGetEvent
point2d.h, [338](#)

caerPoint2DEventPacketGetEventConst
point2d.h, [338](#)

caerPoint2DEventSetScale
point2d.h, [339](#)

caerPoint2DEventSetTimestamp
point2d.h, [339](#)

caerPoint2DEventSetType
point2d.h, [340](#)

caerPoint2DEventSetX
point2d.h, [340](#)

caerPoint2DEventSetY
point2d.h, [340](#)

caerPoint2DEventValidate
point2d.h, [340](#)

caerPoint3DEvent
point3d.h, [347](#)

caerPoint3DEventGetScale
point3d.h, [348](#)

caerPoint3DEventGetTimestamp
point3d.h, 348

caerPoint3DEventGetTimestamp64
point3d.h, 348

caerPoint3DEventGetType
point3d.h, 349

caerPoint3DEventGetX
point3d.h, 349

caerPoint3DEventGetY
point3d.h, 349

caerPoint3DEventGetZ
point3d.h, 350

caerPoint3DEventInvalidate
point3d.h, 350

caerPoint3DEventIsValid
point3d.h, 351

caerPoint3DEventPacket
point3d.h, 347

caerPoint3DEventPacketAllocate
point3d.h, 351

caerPoint3DEventPacketFromPacketHeader
point3d.h, 351

caerPoint3DEventPacketFromPacketHeaderConst
point3d.h, 352

caerPoint3DEventPacketGetEvent
point3d.h, 352

caerPoint3DEventPacketGetEventConst
point3d.h, 352

caerPoint3DEventSetScale
point3d.h, 353

caerPoint3DEventSetTimestamp
point3d.h, 353

caerPoint3DEventSetType
point3d.h, 354

caerPoint3DEventSetX
point3d.h, 354

caerPoint3DEventSetY
point3d.h, 354

caerPoint3DEventSetZ
point3d.h, 354

caerPoint3DEventValidate
point3d.h, 355

caerPoint4DEvent
point4d.h, 361

caerPoint4DEventGetScale
point4d.h, 362

caerPoint4DEventGetTimestamp
point4d.h, 362

caerPoint4DEventGetTimestamp64
point4d.h, 362

caerPoint4DEventGetType
point4d.h, 363

caerPoint4DEventGetW
point4d.h, 363

caerPoint4DEventGetX
point4d.h, 363

caerPoint4DEventGetY
point4d.h, 364

caerPoint4DEventGetZ
point4d.h, 364

caerPoint4DEventInvalidate
point4d.h, 365

caerPoint4DEventIsValid
point4d.h, 365

caerPoint4DEventPacket
point4d.h, 361

caerPoint4DEventPacketAllocate
point4d.h, 365

caerPoint4DEventPacketFromPacketHeader
point4d.h, 366

caerPoint4DEventPacketFromPacketHeaderConst
point4d.h, 366

caerPoint4DEventPacketGetEvent
point4d.h, 366

caerPoint4DEventPacketGetEventConst
point4d.h, 367

caerPoint4DEventSetScale
point4d.h, 367

caerPoint4DEventSetTimestamp
point4d.h, 367

caerPoint4DEventSetType
point4d.h, 368

caerPoint4DEventSetW
point4d.h, 368

caerPoint4DEventSetX
point4d.h, 368

caerPoint4DEventSetY
point4d.h, 369

caerPoint4DEventSetZ
point4d.h, 369

caerPoint4DEventValidate
point4d.h, 369

caerPolarityEvent
polarity.h, 377

caerPolarityEventGetPolarity
polarity.h, 377

caerPolarityEventGetTimestamp
polarity.h, 378

caerPolarityEventGetTimestamp64
polarity.h, 378

caerPolarityEventGetX
polarity.h, 378

caerPolarityEventGetY
polarity.h, 379

caerPolarityEventInvalidate
polarity.h, 379

caerPolarityEventIsValid
polarity.h, 379

caerPolarityEventPacket
polarity.h, 377

caerPolarityEventPacketAllocate
polarity.h, 380

caerPolarityEventPacketFromPacketHeader
polarity.h, 380

caerPolarityEventPacketFromPacketHeaderConst
polarity.h, 381

caerPolarityEventPacketGetEvent
polarity.h, [381](#)

caerPolarityEventPacketGetEventConst
polarity.h, [381](#)

caerPolarityEventSetPolarity
polarity.h, [382](#)

caerPolarityEventSetTimestamp
polarity.h, [382](#)

caerPolarityEventSetX
polarity.h, [382](#)

caerPolarityEventSetY
polarity.h, [383](#)

caerPolarityEventValidate
polarity.h, [383](#)

caerSampleEvent
sample.h, [389](#)

caerSampleEventGetSample
sample.h, [390](#)

caerSampleEventGetTimestamp
sample.h, [390](#)

caerSampleEventGetTimestamp64
sample.h, [390](#)

caerSampleEventGetType
sample.h, [391](#)

caerSampleEventInvalidate
sample.h, [391](#)

caerSampleEventsIsValid
sample.h, [391](#)

caerSampleEventPacket
sample.h, [389](#)

caerSampleEventPacketAllocate
sample.h, [392](#)

caerSampleEventPacketFromPacketHeader
sample.h, [392](#)

caerSampleEventPacketFromPacketHeaderConst
sample.h, [393](#)

caerSampleEventPacketGetEvent
sample.h, [393](#)

caerSampleEventPacketGetEventConst
sample.h, [393](#)

caerSampleEventSetSample
sample.h, [394](#)

caerSampleEventSetTimestamp
sample.h, [394](#)

caerSampleEventSetType
sample.h, [394](#)

caerSampleEventValidate
sample.h, [395](#)

caerSpecialEvent
special.h, [402](#)

caerSpecialEventGetData
special.h, [403](#)

caerSpecialEventGetTimestamp
special.h, [403](#)

caerSpecialEventGetTimestamp64
special.h, [404](#)

caerSpecialEventGetType
special.h, [404](#)

caerSpecialEventInvalidate
special.h, [404](#)

caerSpecialEventsIsValid
special.h, [405](#)

caerSpecialEventPacket
special.h, [402](#)

caerSpecialEventPacketAllocate
special.h, [405](#)

caerSpecialEventPacketFindEventByType
special.h, [406](#)

caerSpecialEventPacketFindEventByTypeConst
special.h, [406](#)

caerSpecialEventPacketFindValidEventByType
special.h, [406](#)

caerSpecialEventPacketFindValidEventByTypeConst
special.h, [407](#)

caerSpecialEventPacketFromPacketHeader
special.h, [407](#)

caerSpecialEventPacketFromPacketHeaderConst
special.h, [408](#)

caerSpecialEventPacketGetEvent
special.h, [408](#)

caerSpecialEventPacketGetEventConst
special.h, [408](#)

caerSpecialEventSetData
special.h, [409](#)

caerSpecialEventSetTimestamp
special.h, [409](#)

caerSpecialEventSetType
special.h, [409](#)

caerSpecialEventValidate
special.h, [410](#)

caerSpikeEvent
spike.h, [416](#)

caerSpikeEventGetChipID
spike.h, [417](#)

caerSpikeEventGetNeuronID
spike.h, [417](#)

caerSpikeEventGetSourceCoreID
spike.h, [418](#)

caerSpikeEventGetTimestamp
spike.h, [418](#)

caerSpikeEventGetTimestamp64
spike.h, [418](#)

caerSpikeEventInvalidate
spike.h, [419](#)

caerSpikeEventsIsValid
spike.h, [419](#)

caerSpikeEventPacket
spike.h, [417](#)

caerSpikeEventPacketAllocate
spike.h, [419](#)

caerSpikeEventPacketFromPacketHeader
spike.h, [420](#)

caerSpikeEventPacketFromPacketHeaderConst
spike.h, [420](#)

caerSpikeEventPacketGetEvent
spike.h, [421](#)

- caerSpikeEventPacketGetEventConst
 - spike.h, [421](#)
- caerSpikeEventSetChiplD
 - spike.h, [421](#)
- caerSpikeEventSetNeuronID
 - spike.h, [423](#)
- caerSpikeEventSetSourceCoreID
 - spike.h, [423](#)
- caerSpikeEventSetTimestamp
 - spike.h, [423](#)
- caerSpikeEventValidate
 - spike.h, [424](#)
- caerStrEquals
 - libcaer.h, [438](#)
- caerStrEqualsUpTo
 - libcaer.h, [438](#)
- common.h
 - CAER_DEFAULT_EVENT_TYPES_COUNT, [176](#)
 - CAER_EVENT_PACKET_HEADER_SIZE, [176](#)
 - CAER_ITERATOR_ALL_END, [177](#)
 - CAER_ITERATOR_ALL_START, [177](#)
 - CAER_ITERATOR_VALID_END, [177](#)
 - CAER_ITERATOR_VALID_START, [177](#)
 - caer_default_event_types, [178](#)
 - caerEventPacketAllocate, [179](#)
 - caerEventPacketAppend, [179](#)
 - caerEventPacketClean, [180](#)
 - caerEventPacketClear, [180](#)
 - caerEventPacketCopy, [180](#)
 - caerEventPacketCopyOnlyEvents, [181](#)
 - caerEventPacketCopyOnlyValidEvents, [181](#)
 - caerEventPacketEquals, [181](#)
 - caerEventPacketGetDataSize, [182](#)
 - caerEventPacketGetDataSizeEvents, [182](#)
 - caerEventPacketGetSize, [183](#)
 - caerEventPacketGetSizeEvents, [183](#)
 - caerEventPacketGrow, [183](#)
 - caerEventPacketHeader, [178](#)
 - caerEventPacketHeaderGetEventCapacity, [184](#)
 - caerEventPacketHeaderGetEventNumber, [184](#)
 - caerEventPacketHeaderGetEventSize, [184](#)
 - caerEventPacketHeaderGetEventSource, [185](#)
 - caerEventPacketHeaderGetEventTSOffset, [185](#)
 - caerEventPacketHeaderGetEventTSOverflow, [186](#)
 - caerEventPacketHeaderGetEventType, [186](#)
 - caerEventPacketHeaderGetEventValid, [186](#)
 - caerEventPacketHeaderSetEventCapacity, [187](#)
 - caerEventPacketHeaderSetEventNumber, [187](#)
 - caerEventPacketHeaderSetEventSize, [187](#)
 - caerEventPacketHeaderSetEventSource, [188](#)
 - caerEventPacketHeaderSetEventTSOffset, [188](#)
 - caerEventPacketHeaderSetEventTSOverflow, [188](#)
 - caerEventPacketHeaderSetEventType, [189](#)
 - caerEventPacketHeaderSetEventValid, [189](#)
 - caerEventPacketResize, [189](#)
 - caerGenericEventCopy, [190](#)
 - caerGenericEventGetEvent, [190](#)
 - caerGenericEventGetTimestamp, [191](#)
 - caerGenericEventGetTimestamp64, [191](#)
 - caerGenericEventIsValid, [192](#)
 - PACKED_STRUCT, [192](#)
 - TS_OVERFLOW_SHIFT, [177](#)
 - VALID_MARK_MASK, [178](#)
 - VALID_MARK_SHIFT, [178](#)
- config.h
 - CAER_CONFIGURATION_CONST_ITERATOR↔
_ALL_START, [194](#)
 - CAER_CONFIGURATION_CONST_ITERATOR↔
_VALID_START, [194](#)
 - CAER_CONFIGURATION_CONST_REVERSE↔
_ITERATOR_ALL_START, [194](#)
 - CAER_CONFIGURATION_CONST_REVERSE↔
_ITERATOR_VALID_START, [195](#)
 - CAER_CONFIGURATION_ITERATOR_ALL_E↔
ND, [195](#)
 - CAER_CONFIGURATION_ITERATOR_ALL_ST↔
ART, [195](#)
 - CAER_CONFIGURATION_ITERATOR_VALID↔
END, [196](#)
 - CAER_CONFIGURATION_ITERATOR_VALID↔
START, [196](#)
 - CAER_CONFIGURATION_REVERSE_ITERAT↔
OR_ALL_END, [196](#)
 - CAER_CONFIGURATION_REVERSE_ITERAT↔
OR_ALL_START, [197](#)
 - CAER_CONFIGURATION_REVERSE_ITERAT↔
OR_VALID_END, [197](#)
 - CAER_CONFIGURATION_REVERSE_ITERAT↔
OR_VALID_START, [197](#)
 - CONFIG_MODULE_ADDR_MASK, [197](#)
 - CONFIG_MODULE_ADDR_SHIFT, [198](#)
 - caerConfigurationEvent, [198](#)
 - caerConfigurationEventGetModuleAddress, [198](#)
 - caerConfigurationEventGetParameter, [199](#)
 - caerConfigurationEventGetParameterAddress, [199](#)
 - caerConfigurationEventGetTimestamp, [199](#)
 - caerConfigurationEventGetTimestamp64, [200](#)
 - caerConfigurationEventInvalidate, [200](#)
 - caerConfigurationEventsIsValid, [200](#)
 - caerConfigurationEventPacket, [198](#)
 - caerConfigurationEventPacketAllocate, [201](#)
 - caerConfigurationEventPacketFromPacketHeader,
[201](#)
 - caerConfigurationEventPacketFromPacket↔
HeaderConst, [202](#)
 - caerConfigurationEventPacketGetEvent, [202](#)
 - caerConfigurationEventPacketGetEventConst, [202](#)
 - caerConfigurationEventSetModuleAddress, [203](#)
 - caerConfigurationEventSetParameter, [203](#)
 - caerConfigurationEventSetParameterAddress, [203](#)
 - caerConfigurationEventSetTimestamp, [204](#)
 - caerConfigurationEventValidate, [204](#)
 - PACKED_STRUCT, [204](#)
- DAVIS128_CONFIG_BIAS_ADCCOMPBP
 - davis.h, [24](#)
- DAVIS128_CONFIG_BIAS_ADCREFHIGH

- davis.h, [24](#)
DAVIS128_CONFIG_BIAS_ADCREFLOW
davis.h, [24](#)
DAVIS128_CONFIG_BIAS_AEPDBN
davis.h, [25](#)
DAVIS128_CONFIG_BIAS_AEPUXBP
davis.h, [25](#)
DAVIS128_CONFIG_BIAS_AEPUYBP
davis.h, [25](#)
DAVIS128_CONFIG_BIAS_APSCAS
davis.h, [26](#)
DAVIS128_CONFIG_BIAS_APSOEVERFLOWLEVEL
davis.h, [26](#)
DAVIS128_CONFIG_BIAS_APSROSFBN
davis.h, [26](#)
DAVIS128_CONFIG_BIAS_BIASBUFFER
davis.h, [27](#)
DAVIS128_CONFIG_BIAS_COLSELLOWBN
davis.h, [27](#)
DAVIS128_CONFIG_BIAS_DACBUFBP
davis.h, [27](#)
DAVIS128_CONFIG_BIAS_DIFFBN
davis.h, [28](#)
DAVIS128_CONFIG_BIAS_IFREFRBN
davis.h, [28](#)
DAVIS128_CONFIG_BIAS_IFTHRBN
davis.h, [28](#)
DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN
davis.h, [29](#)
DAVIS128_CONFIG_BIAS_LOCALBUFBN
davis.h, [29](#)
DAVIS128_CONFIG_BIAS_OFFBN
davis.h, [29](#)
DAVIS128_CONFIG_BIAS_ONBN
davis.h, [30](#)
DAVIS128_CONFIG_BIAS_PADFOLLBN
davis.h, [30](#)
DAVIS128_CONFIG_BIAS_PIXINBN
davis.h, [30](#)
DAVIS128_CONFIG_BIAS_PRBP
davis.h, [31](#)
DAVIS128_CONFIG_BIAS_PRSFBP
davis.h, [31](#)
DAVIS128_CONFIG_BIAS_READOUTBUFBP
davis.h, [31](#)
DAVIS128_CONFIG_BIAS_REFRBP
davis.h, [32](#)
DAVIS128_CONFIG_BIAS_SSN
davis.h, [32](#)
DAVIS128_CONFIG_BIAS_SSP
davis.h, [32](#)
DAVIS128_CONFIG_CHIP_AERNAROW
davis.h, [33](#)
DAVIS128_CONFIG_CHIP_ANALOGMUX0
davis.h, [33](#)
DAVIS128_CONFIG_CHIP_ANALOGMUX1
davis.h, [33](#)
DAVIS128_CONFIG_CHIP_ANALOGMUX2
davis.h, [33](#)
DAVIS128_CONFIG_CHIP_BIASMUX0
davis.h, [33](#)
DAVIS128_CONFIG_CHIP_DIGITALMUX0
davis.h, [34](#)
DAVIS128_CONFIG_CHIP_DIGITALMUX1
davis.h, [34](#)
DAVIS128_CONFIG_CHIP_DIGITALMUX2
davis.h, [34](#)
DAVIS128_CONFIG_CHIP_DIGITALMUX3
davis.h, [34](#)
DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER
davis.h, [34](#)
DAVIS128_CONFIG_CHIP_RESETCALIBNEURON
davis.h, [34](#)
DAVIS128_CONFIG_CHIP_RESETTESTPIXEL
davis.h, [35](#)
DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER
davis.h, [35](#)
DAVIS128_CONFIG_CHIP_TYPENCALIBNEURON
davis.h, [35](#)
DAVIS128_CONFIG_CHIP_USEAOUT
davis.h, [35](#)
DAVIS208_CONFIG_BIAS_ADCCOMPBP
davis.h, [35](#)
DAVIS208_CONFIG_BIAS_ADCREFHIGH
davis.h, [35](#)
DAVIS208_CONFIG_BIAS_ADCREFLOW
davis.h, [36](#)
DAVIS208_CONFIG_BIAS_AEPDBN
davis.h, [36](#)
DAVIS208_CONFIG_BIAS_AEPUXBP
davis.h, [36](#)
DAVIS208_CONFIG_BIAS_AEPUYBP
davis.h, [37](#)
DAVIS208_CONFIG_BIAS_APSCAS
davis.h, [37](#)
DAVIS208_CONFIG_BIAS_APSOEVERFLOWLEVEL
davis.h, [37](#)
DAVIS208_CONFIG_BIAS_APSROSFBN
davis.h, [38](#)
DAVIS208_CONFIG_BIAS_BIASBUFFER
davis.h, [38](#)
DAVIS208_CONFIG_BIAS_COLSELLOWBN
davis.h, [38](#)
DAVIS208_CONFIG_BIAS_DACBUFBP
davis.h, [39](#)
DAVIS208_CONFIG_BIAS_DIFFBN
davis.h, [39](#)
DAVIS208_CONFIG_BIAS_IFREFRBN
davis.h, [39](#)
DAVIS208_CONFIG_BIAS_IFTHRBN
davis.h, [40](#)
DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN
davis.h, [40](#)
DAVIS208_CONFIG_BIAS_LOCALBUFBN
davis.h, [40](#)
DAVIS208_CONFIG_BIAS_OFFBN

- davis.h, [41](#)
- DAVIS208_CONFIG_BIAS_ONBN
 - davis.h, [41](#)
- DAVIS208_CONFIG_BIAS_PADFOLLBN
 - davis.h, [41](#)
- DAVIS208_CONFIG_BIAS_PIXINVBN
 - davis.h, [42](#)
- DAVIS208_CONFIG_BIAS_PRBP
 - davis.h, [42](#)
- DAVIS208_CONFIG_BIAS_PRSFBP
 - davis.h, [42](#)
- DAVIS208_CONFIG_BIAS_READOUTBUFBP
 - davis.h, [43](#)
- DAVIS208_CONFIG_BIAS_REFRBP
 - davis.h, [43](#)
- DAVIS208_CONFIG_BIAS_REFSSBN
 - davis.h, [44](#)
- DAVIS208_CONFIG_BIAS_REFSS
 - davis.h, [43](#)
- DAVIS208_CONFIG_BIAS_REGBIASBP
 - davis.h, [44](#)
- DAVIS208_CONFIG_BIAS_RESETHIGHPASS
 - davis.h, [44](#)
- DAVIS208_CONFIG_BIAS_SSN
 - davis.h, [45](#)
- DAVIS208_CONFIG_BIAS_SSP
 - davis.h, [45](#)
- DAVIS208_CONFIG_CHIP_AERNAROW
 - davis.h, [45](#)
- DAVIS208_CONFIG_CHIP_ANALOGMUX0
 - davis.h, [46](#)
- DAVIS208_CONFIG_CHIP_ANALOGMUX1
 - davis.h, [46](#)
- DAVIS208_CONFIG_CHIP_ANALOGMUX2
 - davis.h, [46](#)
- DAVIS208_CONFIG_CHIP_BIASMUX0
 - davis.h, [46](#)
- DAVIS208_CONFIG_CHIP_DIGITALMUX0
 - davis.h, [46](#)
- DAVIS208_CONFIG_CHIP_DIGITALMUX1
 - davis.h, [46](#)
- DAVIS208_CONFIG_CHIP_DIGITALMUX2
 - davis.h, [47](#)
- DAVIS208_CONFIG_CHIP_DIGITALMUX3
 - davis.h, [47](#)
- DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER
 - davis.h, [47](#)
- DAVIS208_CONFIG_CHIP_RESETCALIBNEURON
 - davis.h, [47](#)
- DAVIS208_CONFIG_CHIP_RESETTESTPIXEL
 - davis.h, [47](#)
- DAVIS208_CONFIG_CHIP_SELECTBIASREFSS
 - davis.h, [47](#)
- DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER
 - davis.h, [48](#)
- DAVIS208_CONFIG_CHIP_SELECTHIGHPASS
 - davis.h, [48](#)
- DAVIS208_CONFIG_CHIP_SELECTPOSFB
 - davis.h, [48](#)
- DAVIS208_CONFIG_CHIP_SELECTPREAMPavg
 - davis.h, [48](#)
- DAVIS208_CONFIG_CHIP_SELECTSENSE
 - davis.h, [48](#)
- DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON
 - davis.h, [48](#)
- DAVIS208_CONFIG_CHIP_USEAOUT
 - davis.h, [49](#)
- DAVIS240_CONFIG_BIAS_AEPDBN
 - davis.h, [49](#)
- DAVIS240_CONFIG_BIAS_AEPUXBP
 - davis.h, [49](#)
- DAVIS240_CONFIG_BIAS_AEPUYBP
 - davis.h, [49](#)
- DAVIS240_CONFIG_BIAS_APSCASEPC
 - davis.h, [50](#)
- DAVIS240_CONFIG_BIAS_APSEVERFLOWLEVELBN
 - davis.h, [50](#)
- DAVIS240_CONFIG_BIAS_APSROSFBN
 - davis.h, [50](#)
- DAVIS240_CONFIG_BIAS_BIASBUFFER
 - davis.h, [50](#)
- DAVIS240_CONFIG_BIAS_DIFFBN
 - davis.h, [51](#)
- DAVIS240_CONFIG_BIAS_DIFFCASBNC
 - davis.h, [51](#)
- DAVIS240_CONFIG_BIAS_IFREFRBN
 - davis.h, [51](#)
- DAVIS240_CONFIG_BIAS_IFTHRBN
 - davis.h, [51](#)
- DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN
 - davis.h, [52](#)
- DAVIS240_CONFIG_BIAS_LOCALBUFBN
 - davis.h, [52](#)
- DAVIS240_CONFIG_BIAS_OFFBN
 - davis.h, [52](#)
- DAVIS240_CONFIG_BIAS_ONBN
 - davis.h, [52](#)
- DAVIS240_CONFIG_BIAS_PADFOLLBN
 - davis.h, [53](#)
- DAVIS240_CONFIG_BIAS_PIXINVBN
 - davis.h, [53](#)
- DAVIS240_CONFIG_BIAS_PRBP
 - davis.h, [53](#)
- DAVIS240_CONFIG_BIAS_PRSFBP
 - davis.h, [53](#)
- DAVIS240_CONFIG_BIAS_REFRBP
 - davis.h, [54](#)
- DAVIS240_CONFIG_BIAS_SSN
 - davis.h, [54](#)
- DAVIS240_CONFIG_BIAS_SSP
 - davis.h, [54](#)
- DAVIS240_CONFIG_CHIP_AERNAROW
 - davis.h, [54](#)
- DAVIS240_CONFIG_CHIP_ANALOGMUX0
 - davis.h, [55](#)
- DAVIS240_CONFIG_CHIP_ANALOGMUX1

- davis.h, [55](#)
DAVIS240_CONFIG_CHIP_ANALOGMUX2
davis.h, [55](#)
DAVIS240_CONFIG_CHIP_BIASMUX0
davis.h, [55](#)
DAVIS240_CONFIG_CHIP_DIGITALMUX0
davis.h, [55](#)
DAVIS240_CONFIG_CHIP_DIGITALMUX1
davis.h, [55](#)
DAVIS240_CONFIG_CHIP_DIGITALMUX2
davis.h, [56](#)
DAVIS240_CONFIG_CHIP_DIGITALMUX3
davis.h, [56](#)
DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER
davis.h, [56](#)
DAVIS240_CONFIG_CHIP_RESETCALIBNEURON
davis.h, [56](#)
DAVIS240_CONFIG_CHIP_RESETTESTPIXEL
davis.h, [56](#)
DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL
davis.h, [56](#)
DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON
davis.h, [57](#)
DAVIS240_CONFIG_CHIP_USEAOUT
davis.h, [57](#)
DAVIS346_CONFIG_BIAS_ADCCOMPBP
davis.h, [57](#)
DAVIS346_CONFIG_BIAS_ADCREFHIGH
davis.h, [57](#)
DAVIS346_CONFIG_BIAS_ADCREFLOW
davis.h, [57](#)
DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE
davis.h, [58](#)
DAVIS346_CONFIG_BIAS_AEPDBN
davis.h, [58](#)
DAVIS346_CONFIG_BIAS_AEPUXBP
davis.h, [58](#)
DAVIS346_CONFIG_BIAS_AEPUYBP
davis.h, [59](#)
DAVIS346_CONFIG_BIAS_APSCAS
davis.h, [59](#)
DAVIS346_CONFIG_BIAS_APSOEVERFLOWLEVEL
davis.h, [59](#)
DAVIS346_CONFIG_BIAS_APSROSFBN
davis.h, [60](#)
DAVIS346_CONFIG_BIAS_BIASBUFFER
davis.h, [60](#)
DAVIS346_CONFIG_BIAS_COLSELLOWBN
davis.h, [60](#)
DAVIS346_CONFIG_BIAS_DACBUFBP
davis.h, [61](#)
DAVIS346_CONFIG_BIAS_DIFFBN
davis.h, [61](#)
DAVIS346_CONFIG_BIAS_IFREFRBN
davis.h, [61](#)
DAVIS346_CONFIG_BIAS_IFTHRBN
davis.h, [62](#)
DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN
davis.h, [62](#)
DAVIS346_CONFIG_BIAS_LOCALBUFBN
davis.h, [62](#)
DAVIS346_CONFIG_BIAS_OFFBN
davis.h, [63](#)
DAVIS346_CONFIG_BIAS_ONBN
davis.h, [63](#)
DAVIS346_CONFIG_BIAS_PADFOLLBN
davis.h, [63](#)
DAVIS346_CONFIG_BIAS_PIXINVCN
davis.h, [64](#)
DAVIS346_CONFIG_BIAS_PRBP
davis.h, [64](#)
DAVIS346_CONFIG_BIAS_PRSFBP
davis.h, [64](#)
DAVIS346_CONFIG_BIAS_READOUTBUFBP
davis.h, [65](#)
DAVIS346_CONFIG_BIAS_REFRBP
davis.h, [65](#)
DAVIS346_CONFIG_BIAS_SSN
davis.h, [65](#)
DAVIS346_CONFIG_BIAS_SSP
davis.h, [66](#)
DAVIS346_CONFIG_CHIP_AERNAROW
davis.h, [66](#)
DAVIS346_CONFIG_CHIP_ANALOGMUX0
davis.h, [66](#)
DAVIS346_CONFIG_CHIP_ANALOGMUX1
davis.h, [66](#)
DAVIS346_CONFIG_CHIP_ANALOGMUX2
davis.h, [67](#)
DAVIS346_CONFIG_CHIP_BIASMUX0
davis.h, [67](#)
DAVIS346_CONFIG_CHIP_DIGITALMUX0
davis.h, [67](#)
DAVIS346_CONFIG_CHIP_DIGITALMUX1
davis.h, [67](#)
DAVIS346_CONFIG_CHIP_DIGITALMUX2
davis.h, [67](#)
DAVIS346_CONFIG_CHIP_DIGITALMUX3
davis.h, [67](#)
DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER
davis.h, [68](#)
DAVIS346_CONFIG_CHIP_RESETCALIBNEURON
davis.h, [68](#)
DAVIS346_CONFIG_CHIP_RESETTESTPIXEL
davis.h, [68](#)
DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER
davis.h, [68](#)
DAVIS346_CONFIG_CHIP_TESTADC
davis.h, [68](#)
DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON
davis.h, [68](#)
DAVIS346_CONFIG_CHIP_USEAOUT
davis.h, [69](#)
DAVIS640_CONFIG_BIAS_ADCCOMPBP
davis.h, [69](#)
DAVIS640_CONFIG_BIAS_ADCREFHIGH

- davis.h, [69](#)
- DAVIS640_CONFIG_BIAS_ADCREFLOW
 - davis.h, [69](#)
- DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE
 - davis.h, [70](#)
- DAVIS640_CONFIG_BIAS_AEPDBN
 - davis.h, [70](#)
- DAVIS640_CONFIG_BIAS_AEPUXBP
 - davis.h, [70](#)
- DAVIS640_CONFIG_BIAS_AEPUYBP
 - davis.h, [71](#)
- DAVIS640_CONFIG_BIAS_APSCAS
 - davis.h, [71](#)
- DAVIS640_CONFIG_BIAS_APSOEVERFLOWLEVEL
 - davis.h, [71](#)
- DAVIS640_CONFIG_BIAS_APSROSFBN
 - davis.h, [72](#)
- DAVIS640_CONFIG_BIAS_BIASBUFFER
 - davis.h, [72](#)
- DAVIS640_CONFIG_BIAS_COLSELLOWBN
 - davis.h, [72](#)
- DAVIS640_CONFIG_BIAS_DACBUFBP
 - davis.h, [73](#)
- DAVIS640_CONFIG_BIAS_DIFFBN
 - davis.h, [73](#)
- DAVIS640_CONFIG_BIAS_IFREFRBN
 - davis.h, [73](#)
- DAVIS640_CONFIG_BIAS_IFTHRBN
 - davis.h, [74](#)
- DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN
 - davis.h, [74](#)
- DAVIS640_CONFIG_BIAS_LOCALBUFBN
 - davis.h, [74](#)
- DAVIS640_CONFIG_BIAS_OFFBN
 - davis.h, [75](#)
- DAVIS640_CONFIG_BIAS_ONBN
 - davis.h, [75](#)
- DAVIS640_CONFIG_BIAS_PADFOLLBN
 - davis.h, [75](#)
- DAVIS640_CONFIG_BIAS_PIXINBN
 - davis.h, [76](#)
- DAVIS640_CONFIG_BIAS_PRBP
 - davis.h, [76](#)
- DAVIS640_CONFIG_BIAS_PRSFBP
 - davis.h, [76](#)
- DAVIS640_CONFIG_BIAS_READOUTBUFBP
 - davis.h, [77](#)
- DAVIS640_CONFIG_BIAS_REFRBP
 - davis.h, [77](#)
- DAVIS640_CONFIG_BIAS_SSN
 - davis.h, [77](#)
- DAVIS640_CONFIG_BIAS_SSP
 - davis.h, [78](#)
- DAVIS640_CONFIG_CHIP_AERNAROW
 - davis.h, [78](#)
- DAVIS640_CONFIG_CHIP_ANALOGMUX0
 - davis.h, [78](#)
- DAVIS640_CONFIG_CHIP_ANALOGMUX1
 - davis.h, [78](#)
- DAVIS640_CONFIG_CHIP_ANALOGMUX2
 - davis.h, [79](#)
- DAVIS640_CONFIG_CHIP_BIASMUX0
 - davis.h, [79](#)
- DAVIS640_CONFIG_CHIP_DIGITALMUX0
 - davis.h, [79](#)
- DAVIS640_CONFIG_CHIP_DIGITALMUX1
 - davis.h, [79](#)
- DAVIS640_CONFIG_CHIP_DIGITALMUX2
 - davis.h, [79](#)
- DAVIS640_CONFIG_CHIP_DIGITALMUX3
 - davis.h, [79](#)
- DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER
 - davis.h, [80](#)
- DAVIS640_CONFIG_CHIP_RESETCALIBNEURON
 - davis.h, [80](#)
- DAVIS640_CONFIG_CHIP_RESETTESTPIXEL
 - davis.h, [80](#)
- DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER
 - davis.h, [80](#)
- DAVIS640_CONFIG_CHIP_TESTADC
 - davis.h, [80](#)
- DAVIS640_CONFIG_CHIP_TYPENCALIBNEURON
 - davis.h, [80](#)
- DAVIS640_CONFIG_CHIP_USEAOUT
 - davis.h, [81](#)
- DAVIS_APS_ROI_REGIONS_MAX
 - davis.h, [81](#)
- DAVIS_CHIP_DAVIS128
 - davis.h, [81](#)
- DAVIS_CHIP_DAVIS208
 - davis.h, [81](#)
- DAVIS_CHIP_DAVIS240A
 - davis.h, [81](#)
- DAVIS_CHIP_DAVIS240B
 - davis.h, [81](#)
- DAVIS_CHIP_DAVIS240C
 - davis.h, [81](#)
- DAVIS_CHIP_DAVIS346A
 - davis.h, [82](#)
- DAVIS_CHIP_DAVIS346B
 - davis.h, [82](#)
- DAVIS_CHIP_DAVIS346C
 - davis.h, [82](#)
- DAVIS_CHIP_DAVIS640
 - davis.h, [82](#)
- DAVIS_CHIP_DAVISRGB
 - davis.h, [82](#)
- DAVIS_CONFIG_APS_ADC_TEST_MODE
 - davis.h, [82](#)
- DAVIS_CONFIG_APS_AUTOEXPOSURE
 - davis.h, [83](#)
- DAVIS_CONFIG_APS_COLOR_FILTER
 - davis.h, [83](#)
- DAVIS_CONFIG_APS_COLUMN_SETTLE
 - davis.h, [83](#)
- DAVIS_CONFIG_APS_END_COLUMN_0

- davis.h, [83](#)
- DAVIS_CONFIG_APS_END_COLUMN_1
 - davis.h, [83](#)
- DAVIS_CONFIG_APS_END_COLUMN_2
 - davis.h, [83](#)
- DAVIS_CONFIG_APS_END_COLUMN_3
 - davis.h, [83](#)
- DAVIS_CONFIG_APS_END_ROW_0
 - davis.h, [84](#)
- DAVIS_CONFIG_APS_END_ROW_1
 - davis.h, [84](#)
- DAVIS_CONFIG_APS_END_ROW_2
 - davis.h, [84](#)
- DAVIS_CONFIG_APS_END_ROW_3
 - davis.h, [84](#)
- DAVIS_CONFIG_APS_EXPOSURE
 - davis.h, [84](#)
- DAVIS_CONFIG_APS_FRAME_DELAY
 - davis.h, [84](#)
- DAVIS_CONFIG_APS_GLOBAL_SHUTTER
 - davis.h, [84](#)
- DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER
 - davis.h, [85](#)
- DAVIS_CONFIG_APS_HAS_INTERNAL_ADC
 - davis.h, [85](#)
- DAVIS_CONFIG_APS_HAS_QUAD_ROI
 - davis.h, [85](#)
- DAVIS_CONFIG_APS_NULL_SETTLE
 - davis.h, [85](#)
- DAVIS_CONFIG_APS_ORIENTATION_INFO
 - davis.h, [85](#)
- DAVIS_CONFIG_APS_RAMP_RESET
 - davis.h, [85](#)
- DAVIS_CONFIG_APS_RAMP_SHORT_RESET
 - davis.h, [86](#)
- DAVIS_CONFIG_APS_RESET_READ
 - davis.h, [86](#)
- DAVIS_CONFIG_APS_RESET_SETTLE
 - davis.h, [86](#)
- DAVIS_CONFIG_APS_ROI0_ENABLED
 - davis.h, [86](#)
- DAVIS_CONFIG_APS_ROI1_ENABLED
 - davis.h, [86](#)
- DAVIS_CONFIG_APS_ROI2_ENABLED
 - davis.h, [86](#)
- DAVIS_CONFIG_APS_ROI3_ENABLED
 - davis.h, [86](#)
- DAVIS_CONFIG_APS_ROW_SETTLE
 - davis.h, [87](#)
- DAVIS_CONFIG_APS_RUN
 - davis.h, [87](#)
- DAVIS_CONFIG_APS_SAMPLE_ENABLE
 - davis.h, [87](#)
- DAVIS_CONFIG_APS_SAMPLE_SETTLE
 - davis.h, [87](#)
- DAVIS_CONFIG_APS_SIZE_COLUMNS
 - davis.h, [87](#)
- DAVIS_CONFIG_APS_SIZE_ROWS
 - davis.h, [87](#)
- DAVIS_CONFIG_APS_SNAPSHOT
 - davis.h, [87](#)
- DAVIS_CONFIG_APS_START_COLUMN_0
 - davis.h, [88](#)
- DAVIS_CONFIG_APS_START_COLUMN_1
 - davis.h, [88](#)
- DAVIS_CONFIG_APS_START_COLUMN_2
 - davis.h, [88](#)
- DAVIS_CONFIG_APS_START_COLUMN_3
 - davis.h, [88](#)
- DAVIS_CONFIG_APS_START_ROW_0
 - davis.h, [88](#)
- DAVIS_CONFIG_APS_START_ROW_1
 - davis.h, [88](#)
- DAVIS_CONFIG_APS_START_ROW_2
 - davis.h, [89](#)
- DAVIS_CONFIG_APS_START_ROW_3
 - davis.h, [89](#)
- DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL
 - davis.h, [89](#)
- DAVIS_CONFIG_APS
 - davis.h, [82](#)
- DAVIS_CONFIG_BIAS
 - davis.h, [89](#)
- DAVIS_CONFIG_CHIP
 - davis.h, [89](#)
- DAVIS_CONFIG_DDRAER_ACK_DELAY
 - davis.h, [90](#)
- DAVIS_CONFIG_DDRAER_REQ_DELAY
 - davis.h, [90](#)
- DAVIS_CONFIG_DDRAER_RUN
 - davis.h, [90](#)
- DAVIS_CONFIG_DDRAER
 - davis.h, [89](#)
- DAVIS_CONFIG_DVS_ACK_DELAY_COLUMN
 - davis.h, [90](#)
- DAVIS_CONFIG_DVS_ACK_DELAY_ROW
 - davis.h, [90](#)
- DAVIS_CONFIG_DVS_ACK_EXTENSION_COLUMN
 - davis.h, [90](#)
- DAVIS_CONFIG_DVS_ACK_EXTENSION_ROW
 - davis.h, [91](#)
- DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL
 - davis.h, [91](#)
- DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_TIME
 - davis.h, [91](#)
- DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY
 - davis.h, [91](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN
 - davis.h, [91](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW
 - davis.h, [91](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN
 - davis.h, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW
 - davis.h, [92](#)

- davis.h, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN
davis.h, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW
davis.h, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN
davis.h, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW
davis.h, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN
davis.h, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW
davis.h, [93](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN
davis.h, [93](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW
davis.h, [93](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN
davis.h, [93](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW
davis.h, [93](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN
davis.h, [93](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW
davis.h, [93](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_AUTO_TRAIN
davis.h, [94](#)
- DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD_TIME
davis.h, [94](#)
- DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD
davis.h, [94](#)
- DAVIS_CONFIG_DVS_FILTER_ROI_END_COLUMN
davis.h, [94](#)
- DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW
davis.h, [94](#)
- DAVIS_CONFIG_DVS_FILTER_ROI_START_COLUMN
davis.h, [94](#)
- DAVIS_CONFIG_DVS_FILTER_ROI_START_ROW
davis.h, [94](#)
- DAVIS_CONFIG_DVS_FILTER_ROW_ONLY_EVENTS
davis.h, [95](#)
- DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER
davis.h, [95](#)
- DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER
davis.h, [95](#)
- DAVIS_CONFIG_DVS_HAS_ROI_FILTER
davis.h, [95](#)
- DAVIS_CONFIG_DVS_HAS_STATISTICS
davis.h, [95](#)
- DAVIS_CONFIG_DVS_HAS_TEST_EVENT_GENERATOR
davis.h, [95](#)
- DAVIS_CONFIG_DVS_ORIENTATION_INFO
davis.h, [96](#)
- DAVIS_CONFIG_DVS_RUN
davis.h, [96](#)
- DAVIS_CONFIG_DVS_SIZE_COLUMNS
davis.h, [96](#)
- DAVIS_CONFIG_DVS_SIZE_ROWS
davis.h, [96](#)
- DAVIS_CONFIG_DVS_STATISTICS_EVENTS_COLUMNS
davis.h, [96](#)
- DAVIS_CONFIG_DVS_STATISTICS_EVENTS_DROPPED
davis.h, [96](#)
- DAVIS_CONFIG_DVS_STATISTICS_EVENTS_ROW
davis.h, [97](#)
- DAVIS_CONFIG_DVS_STATISTICS_FILTERED_BACKGROUND_ACTIVITY
davis.h, [97](#)
- DAVIS_CONFIG_DVS_STATISTICS_FILTERED_PIXELS
davis.h, [97](#)
- DAVIS_CONFIG_DVS_STATISTICS_FILTERED_REFRACTORY_PERIOD
davis.h, [97](#)
- DAVIS_CONFIG_DVS_TEST_EVENT_GENERATOR_ENABLE
davis.h, [97](#)
- DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL
davis.h, [97](#)
- DAVIS_CONFIG_DVS
davis.h, [90](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGE1
davis.h, [98](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGE2
davis.h, [98](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGE
davis.h, [98](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH1
davis.h, [98](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH2
davis.h, [99](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH
davis.h, [98](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY1
davis.h, [99](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY2
davis.h, [99](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY
davis.h, [99](#)

- DAVIS_CONFIG_EXTINPUT_DETECT_PULSES1
davis.h, [99](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSES2
davis.h, [100](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSES
davis.h, [99](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGE↔
GES1
davis.h, [100](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGE↔
GES2
davis.h, [100](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGE↔
GES
davis.h, [100](#)
- DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT↔
ON_FALLING_EDGE
davis.h, [100](#)
- DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT↔
ON_RISING_EDGE
davis.h, [100](#)
- DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL↔
davis.h, [101](#)
- DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH↔
davis.h, [101](#)
- DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY↔
davis.h, [101](#)
- DAVIS_CONFIG_EXTINPUT_GENERATE_USE_CUSTOM_SIGNAL↔
davis.h, [101](#)
- DAVIS_CONFIG_EXTINPUT_HAS_EXTRA_DETECTORS↔
davis.h, [101](#)
- DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR
davis.h, [101](#)
- DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR1
davis.h, [102](#)
- DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR2
davis.h, [102](#)
- DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR
davis.h, [102](#)
- DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR
davis.h, [102](#)
- DAVIS_CONFIG_EXTINPUT
davis.h, [98](#)
- DAVIS_CONFIG_IMU_ACCEL_DLPF
davis.h, [102](#)
- DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE
davis.h, [103](#)
- DAVIS_CONFIG_IMU_ACCEL_STANDBY
davis.h, [103](#)
- DAVIS_CONFIG_IMU_GYRO_DLPF
davis.h, [103](#)
- DAVIS_CONFIG_IMU_GYRO_FULL_SCALE
davis.h, [103](#)
- DAVIS_CONFIG_IMU_GYRO_STANDBY
davis.h, [103](#)
- DAVIS_CONFIG_IMU_LP_CYCLE
davis.h, [104](#)
- DAVIS_CONFIG_IMU_LP_WAKEUP
davis.h, [104](#)
- DAVIS_CONFIG_IMU_ORIENTATION_INFO
davis.h, [104](#)
- DAVIS_CONFIG_IMU_RUN
davis.h, [104](#)
- DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER
davis.h, [104](#)
- DAVIS_CONFIG_IMU_TEMP_STANDBY
davis.h, [104](#)
- DAVIS_CONFIG_IMU
davis.h, [102](#)
- DAVIS_CONFIG_MUX_DROP_APS_ON_TRANSFER_STALL↔
davis.h, [105](#)
- DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL↔
davis.h, [105](#)
- DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL↔
davis.h, [105](#)
- DAVIS_CONFIG_MUX_DROP_IMU_ON_TRANSFER_STALL↔
davis.h, [105](#)
- DAVIS_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE
davis.h, [105](#)
- DAVIS_CONFIG_MUX_HAS_STATISTICS
davis.h, [106](#)
- DAVIS_CONFIG_MUX_RUN
davis.h, [106](#)
- DAVIS_CONFIG_MUX_STATISTICS_APS_DROPPED
davis.h, [106](#)
- DAVIS_CONFIG_MUX_STATISTICS_DVS_DROPPED
davis.h, [106](#)
- DAVIS_CONFIG_MUX_STATISTICS_EXTINPUT_DROPPED↔
davis.h, [106](#)
- DAVIS_CONFIG_MUX_STATISTICS_IMU_DROPPED
davis.h, [106](#)
- DAVIS_CONFIG_MUX_TIMESTAMP_RESET
davis.h, [107](#)
- DAVIS_CONFIG_MUX_TIMESTAMP_RUN
davis.h, [107](#)
- DAVIS_CONFIG_MUX
davis.h, [105](#)
- DAVIS_CONFIG_SYSINFO_ADC_CLOCK
davis.h, [107](#)
- DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER
davis.h, [107](#)
- DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER
davis.h, [107](#)
- DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK
davis.h, [108](#)
- DAVIS_CONFIG_SYSINFO_LOGIC_VERSION

- davis.h, [108](#)
- DAVIS_CONFIG_SYSINFO
 - davis.h, [107](#)
- DAVIS_CONFIG_USB_EARLY_PACKET_DELAY
 - davis.h, [108](#)
- DAVIS_CONFIG_USB_RUN
 - davis.h, [108](#)
- DAVIS_CONFIG_USB
 - davis.h, [108](#)
- DAVISRGB_CONFIG_APS_GSFDRESET
 - davis.h, [108](#)
- DAVISRGB_CONFIG_APS_GSPDRESET
 - davis.h, [109](#)
- DAVISRGB_CONFIG_APS_GSRESETFALL
 - davis.h, [109](#)
- DAVISRGB_CONFIG_APS_GSTXFALL
 - davis.h, [109](#)
- DAVISRGB_CONFIG_APS_RSFDSETTLE
 - davis.h, [109](#)
- DAVISRGB_CONFIG_APS_TRANSFER
 - davis.h, [109](#)
- DAVISRGB_CONFIG_BIAS_ADCCOMPBP
 - davis.h, [109](#)
- DAVISRGB_CONFIG_BIAS_ADCREFHIGH
 - davis.h, [110](#)
- DAVISRGB_CONFIG_BIAS_ADCREFLOW
 - davis.h, [110](#)
- DAVISRGB_CONFIG_BIAS_ADCTESTVOLTAGE
 - davis.h, [110](#)
- DAVISRGB_CONFIG_BIAS_AEPDBN
 - davis.h, [111](#)
- DAVISRGB_CONFIG_BIAS_AEPUXBP
 - davis.h, [111](#)
- DAVISRGB_CONFIG_BIAS_AEPUYBP
 - davis.h, [111](#)
- DAVISRGB_CONFIG_BIAS_APSCAS
 - davis.h, [112](#)
- DAVISRGB_CONFIG_BIAS_APSROSFBN
 - davis.h, [112](#)
- DAVISRGB_CONFIG_BIAS_ARRAYBIASBUFFERBN
 - davis.h, [112](#)
- DAVISRGB_CONFIG_BIAS_ARRAYLOGICBUFFERBN
 - davis.h, [113](#)
- DAVISRGB_CONFIG_BIAS_BIASBUFFER
 - davis.h, [113](#)
- DAVISRGB_CONFIG_BIAS_DACBUFBP
 - davis.h, [113](#)
- DAVISRGB_CONFIG_BIAS_DIFFBN
 - davis.h, [114](#)
- DAVISRGB_CONFIG_BIAS_FALLTIMEBN
 - davis.h, [114](#)
- DAVISRGB_CONFIG_BIAS_GND07
 - davis.h, [114](#)
- DAVISRGB_CONFIG_BIAS_IFREFRBN
 - davis.h, [115](#)
- DAVISRGB_CONFIG_BIAS_IFTHRBN
 - davis.h, [115](#)
- DAVISRGB_CONFIG_BIAS_LCOLTIMEOUTBN
 - davis.h, [115](#)
- DAVISRGB_CONFIG_BIAS_LOCALBUFBN
 - davis.h, [116](#)
- DAVISRGB_CONFIG_BIAS_OFFBN
 - davis.h, [116](#)
- DAVISRGB_CONFIG_BIAS_ONBN
 - davis.h, [116](#)
- DAVISRGB_CONFIG_BIAS_OVG1LO
 - davis.h, [117](#)
- DAVISRGB_CONFIG_BIAS_OVG2LO
 - davis.h, [117](#)
- DAVISRGB_CONFIG_BIAS_PADFOLLBN
 - davis.h, [117](#)
- DAVISRGB_CONFIG_BIAS_PIXINVCN
 - davis.h, [118](#)
- DAVISRGB_CONFIG_BIAS_PRBP
 - davis.h, [118](#)
- DAVISRGB_CONFIG_BIAS_PRSFBP
 - davis.h, [118](#)
- DAVISRGB_CONFIG_BIAS_READOUTBUFBP
 - davis.h, [119](#)
- DAVISRGB_CONFIG_BIAS_REFRBP
 - davis.h, [119](#)
- DAVISRGB_CONFIG_BIAS_RISETIMEBP
 - davis.h, [119](#)
- DAVISRGB_CONFIG_BIAS_SSN
 - davis.h, [120](#)
- DAVISRGB_CONFIG_BIAS_SSP
 - davis.h, [120](#)
- DAVISRGB_CONFIG_BIAS_TX2OVG2HI
 - davis.h, [120](#)
- DAVISRGB_CONFIG_CHIP_ADJUSTOVG1LO
 - davis.h, [121](#)
- DAVISRGB_CONFIG_CHIP_ADJUSTOVG2LO
 - davis.h, [121](#)
- DAVISRGB_CONFIG_CHIP_ADJUSTTX2OVG2HI
 - davis.h, [121](#)
- DAVISRGB_CONFIG_CHIP_AERNAROW
 - davis.h, [121](#)
- DAVISRGB_CONFIG_CHIP_ANALOGMUX0
 - davis.h, [121](#)
- DAVISRGB_CONFIG_CHIP_ANALOGMUX1
 - davis.h, [122](#)
- DAVISRGB_CONFIG_CHIP_ANALOGMUX2
 - davis.h, [122](#)
- DAVISRGB_CONFIG_CHIP_BIASMUX0
 - davis.h, [122](#)
- DAVISRGB_CONFIG_CHIP_DIGITALMUX0
 - davis.h, [122](#)
- DAVISRGB_CONFIG_CHIP_DIGITALMUX1
 - davis.h, [122](#)
- DAVISRGB_CONFIG_CHIP_DIGITALMUX2
 - davis.h, [122](#)
- DAVISRGB_CONFIG_CHIP_DIGITALMUX3
 - davis.h, [123](#)
- DAVISRGB_CONFIG_CHIP_RESETCALIBNEURON
 - davis.h, [123](#)

- DAVISRGB_CONFIG_CHIP_RESETESTPIXEL
davis.h, 123
- DAVISRGB_CONFIG_CHIP_SELECTGRAYCOUNT↔
ER
davis.h, 123
- DAVISRGB_CONFIG_CHIP_TESTADC
davis.h, 123
- DAVISRGB_CONFIG_CHIP_TYPENCALIBNEURON
davis.h, 123
- DAVISRGB_CONFIG_CHIP_USEAOUT
davis.h, 124
- DVS128_CONFIG_BIAS_CAS
dvs128.h, 139
- DVS128_CONFIG_BIAS_DIFFOFF
dvs128.h, 140
- DVS128_CONFIG_BIAS_DIFFON
dvs128.h, 140
- DVS128_CONFIG_BIAS_DIFF
dvs128.h, 140
- DVS128_CONFIG_BIAS_FOLL
dvs128.h, 140
- DVS128_CONFIG_BIAS_INJGND
dvs128.h, 140
- DVS128_CONFIG_BIAS_PUX
dvs128.h, 140
- DVS128_CONFIG_BIAS_PUY
dvs128.h, 141
- DVS128_CONFIG_BIAS_PR
dvs128.h, 140
- DVS128_CONFIG_BIAS_REFR
dvs128.h, 141
- DVS128_CONFIG_BIAS_REQPD
dvs128.h, 141
- DVS128_CONFIG_BIAS_REQ
dvs128.h, 141
- DVS128_CONFIG_BIAS
dvs128.h, 139
- DVS128_CONFIG_DVS_ARRAY_RESET
dvs128.h, 141
- DVS128_CONFIG_DVS_RUN
dvs128.h, 141
- DVS128_CONFIG_DVS_TIMESTAMP_RESET
dvs128.h, 142
- DVS128_CONFIG_DVS_TS_MASTER
dvs128.h, 142
- DVS128_CONFIG_DVS
dvs128.h, 141
- DYNAPSE_CONFIG_DYNAPSE
dynapse.h, 147
- DYNAPSE_CONFIG_AER_ACK_DELAY
dynapse.h, 148
- DYNAPSE_CONFIG_AER_ACK_EXTENSION
dynapse.h, 148
- DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONT↔
ROL
dynapse.h, 148
- DYNAPSE_CONFIG_AER_HAS_STATISTICS
dynapse.h, 148
- DYNAPSE_CONFIG_AER_RUN
dynapse.h, 148
- DYNAPSE_CONFIG_AER_STATISTICS_EVENTS↔
DROPPED
dynapse.h, 149
- DYNAPSE_CONFIG_AER_STATISTICS_EVENTS
dynapse.h, 148
- DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER↔
STALL
dynapse.h, 149
- DYNAPSE_CONFIG_AER
dynapse.h, 148
- DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P
dynapse.h, 149
- DYNAPSE_CONFIG_CHIP_CONTENT
dynapse.h, 149
- DYNAPSE_CONFIG_CHIP_ID
dynapse.h, 149
- DYNAPSE_CONFIG_CHIP_REQ_DELAY
dynapse.h, 150
- DYNAPSE_CONFIG_CHIP_REQ_EXTENSION
dynapse.h, 150
- DYNAPSE_CONFIG_CHIP_RUN
dynapse.h, 150
- DYNAPSE_CONFIG_CHIP
dynapse.h, 149
- DYNAPSE_CONFIG_CLEAR_CAM
dynapse.h, 150
- DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY
dynapse.h, 150
- DYNAPSE_CONFIG_DEFAULT_SRAM
dynapse.h, 150
- DYNAPSE_CONFIG_MONITOR_NEU
dynapse.h, 150
- DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRAN↔
SFER_STALL
dynapse.h, 151
- DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_E↔
NABLE
dynapse.h, 151
- DYNAPSE_CONFIG_MUX_HAS_STATISTICS
dynapse.h, 151
- DYNAPSE_CONFIG_MUX_RUN
dynapse.h, 151
- DYNAPSE_CONFIG_MUX_STATISTICS_AER_DRO↔
PPED
dynapse.h, 151
- DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET
dynapse.h, 152
- DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN
dynapse.h, 152
- DYNAPSE_CONFIG_MUX
dynapse.h, 151
- DYNAPSE_CONFIG_POISSONSPIKEGEN_CHIPID
dynapse.h, 152
- DYNAPSE_CONFIG_POISSONSPIKEGEN_RUN
dynapse.h, 152

- DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITE↔
ADDRESS
dynapse.h, [152](#)
- DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITE↔
DATA
dynapse.h, [152](#)
- DYNAPSE_CONFIG_POISSONSPIKEGEN
dynapse.h, [152](#)
- DYNAPSE_CONFIG_SPIKEGEN_BASEADDR
dynapse.h, [153](#)
- DYNAPSE_CONFIG_SPIKEGEN_ISIBASE
dynapse.h, [153](#)
- DYNAPSE_CONFIG_SPIKEGEN_ISI
dynapse.h, [153](#)
- DYNAPSE_CONFIG_SPIKEGEN_REPEAT
dynapse.h, [153](#)
- DYNAPSE_CONFIG_SPIKEGEN_RUN
dynapse.h, [153](#)
- DYNAPSE_CONFIG_SPIKEGEN_STIMCOUNT
dynapse.h, [153](#)
- DYNAPSE_CONFIG_SPIKEGEN_VARMODE
dynapse.h, [154](#)
- DYNAPSE_CONFIG_SPIKEGEN
dynapse.h, [153](#)
- DYNAPSE_CONFIG_SRAM_ADDRESS
dynapse.h, [154](#)
- DYNAPSE_CONFIG_SRAM_BURSTMODE
dynapse.h, [154](#)
- DYNAPSE_CONFIG_SRAM_DIRECTION_POS
dynapse.h, [154](#)
- DYNAPSE_CONFIG_SRAM_READDATA
dynapse.h, [155](#)
- DYNAPSE_CONFIG_SRAM_READ
dynapse.h, [154](#)
- DYNAPSE_CONFIG_SRAM_RWCOMMAND
dynapse.h, [155](#)
- DYNAPSE_CONFIG_SRAM_WRITEDATA
dynapse.h, [155](#)
- DYNAPSE_CONFIG_SRAM_WRITE
dynapse.h, [155](#)
- DYNAPSE_CONFIG_SRAM
dynapse.h, [154](#)
- DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPS↔
ELECT
dynapse.h, [155](#)
- DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOB↔
ALKERNEL
dynapse.h, [156](#)
- DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN
dynapse.h, [156](#)
- DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAM↔
BASEADDR
dynapse.h, [156](#)
- DYNAPSE_CONFIG_SYNAPSERECONFIG_USES↔
RAMKERNELS
dynapse.h, [156](#)
- DYNAPSE_CONFIG_SYNAPSERECONFIG
dynapse.h, [155](#)
- DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER
dynapse.h, [156](#)
- DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER
dynapse.h, [157](#)
- DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK
dynapse.h, [157](#)
- DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION
dynapse.h, [157](#)
- DYNAPSE_CONFIG_SYSINFO
dynapse.h, [156](#)
- DYNAPSE_CONFIG_TAU1_RESET
dynapse.h, [157](#)
- DYNAPSE_CONFIG_TAU2_RESET
dynapse.h, [157](#)
- DYNAPSE_CONFIG_TAU2_SET
dynapse.h, [157](#)
- DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY
dynapse.h, [158](#)
- DYNAPSE_CONFIG_USB_RUN
dynapse.h, [158](#)
- DYNAPSE_CONFIG_USB
dynapse.h, [158](#)
- DYNAPSE_X4BOARD_COREX
dynapse.h, [158](#)
- DYNAPSE_X4BOARD_COREY
dynapse.h, [158](#)
- DYNAPSE_X4BOARD_NEUX
dynapse.h, [158](#)
- DYNAPSE_X4BOARD_NEUY
dynapse.h, [159](#)
- DYNAPSE_X4BOARD_NUMCHIPS
dynapse.h, [159](#)
- davis.h
 - CAER_DEVICE_DAVIS_FX2, [23](#)
 - CAER_DEVICE_DAVIS_FX3, [24](#)
 - CAER_DEVICE_DAVIS_RPI, [24](#)
 - CAER_DEVICE_DAVIS, [23](#)
 - caer_bias_shiftedsourcesource_operating_mode, [126](#)
 - caer_bias_shiftedsourcesource_voltage_level, [126](#)
 - caerBiasCoarseFineGenerate, [127](#)
 - caerBiasCoarseFineParse, [127](#)
 - caerBiasShiftedSourceGenerate, [127](#)
 - caerBiasShiftedSourceParse, [128](#)
 - caerBiasVDACGenerate, [128](#)
 - caerBiasVDACParse, [128](#)
 - caerDavisInfoGet, [129](#)
 - caerDavisROIConfigure, [129](#)
 - DAVIS128_CONFIG_BIAS_ADCCOMPBP, [24](#)
 - DAVIS128_CONFIG_BIAS_ADCCREFHIGH, [24](#)
 - DAVIS128_CONFIG_BIAS_ADCCREFLOW, [24](#)
 - DAVIS128_CONFIG_BIAS_AEPDBN, [25](#)
 - DAVIS128_CONFIG_BIAS_AEPUXBP, [25](#)
 - DAVIS128_CONFIG_BIAS_AEPUYBP, [25](#)
 - DAVIS128_CONFIG_BIAS_APSCAS, [26](#)
 - DAVIS128_CONFIG_BIAS_APSSOVERFLOWLE↔
VEL, [26](#)
 - DAVIS128_CONFIG_BIAS_APSROSFBN, [26](#)
 - DAVIS128_CONFIG_BIAS_BIASBUFFER, [27](#)

- DAVIS128_CONFIG_BIAS_COLSELLOWBN, 27
 DAVIS128_CONFIG_BIAS_DACBUFBP, 27
 DAVIS128_CONFIG_BIAS_DIFFBN, 28
 DAVIS128_CONFIG_BIAS_IFREFRBN, 28
 DAVIS128_CONFIG_BIAS_IFTHRBN, 28
 DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN, 29
 DAVIS128_CONFIG_BIAS_LOCALBUFBN, 29
 DAVIS128_CONFIG_BIAS_OFFBN, 29
 DAVIS128_CONFIG_BIAS_ONBN, 30
 DAVIS128_CONFIG_BIAS_PADFOLLBN, 30
 DAVIS128_CONFIG_BIAS_PIXINVBN, 30
 DAVIS128_CONFIG_BIAS_PRBP, 31
 DAVIS128_CONFIG_BIAS_PRSFBP, 31
 DAVIS128_CONFIG_BIAS_READOUTBUFBP, 31
 DAVIS128_CONFIG_BIAS_REFRBP, 32
 DAVIS128_CONFIG_BIAS_SSN, 32
 DAVIS128_CONFIG_BIAS_SSP, 32
 DAVIS128_CONFIG_CHIP_AERNAROW, 33
 DAVIS128_CONFIG_CHIP_ANALOGMUX0, 33
 DAVIS128_CONFIG_CHIP_ANALOGMUX1, 33
 DAVIS128_CONFIG_CHIP_ANALOGMUX2, 33
 DAVIS128_CONFIG_CHIP_BIASMUX0, 33
 DAVIS128_CONFIG_CHIP_DIGITALMUX0, 34
 DAVIS128_CONFIG_CHIP_DIGITALMUX1, 34
 DAVIS128_CONFIG_CHIP_DIGITALMUX2, 34
 DAVIS128_CONFIG_CHIP_DIGITALMUX3, 34
 DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER, 34
 DAVIS128_CONFIG_CHIP_RESETCALIBNEU←RON, 34
 DAVIS128_CONFIG_CHIP_RESETTESTPIXEL, 35
 DAVIS128_CONFIG_CHIP_SELECTGRAYCO←UNTER, 35
 DAVIS128_CONFIG_CHIP_TYPENCALIBNEU←RON, 35
 DAVIS128_CONFIG_CHIP_USEAOUT, 35
 DAVIS208_CONFIG_BIAS_ADCCOMPBP, 35
 DAVIS208_CONFIG_BIAS_ADCREFHIGH, 35
 DAVIS208_CONFIG_BIAS_ADCREFLOW, 36
 DAVIS208_CONFIG_BIAS_AEPDBN, 36
 DAVIS208_CONFIG_BIAS_AEPUXBP, 36
 DAVIS208_CONFIG_BIAS_AEPUYBP, 37
 DAVIS208_CONFIG_BIAS_APSCAS, 37
 DAVIS208_CONFIG_BIAS_APSOEVERFLOWLE←VEL, 37
 DAVIS208_CONFIG_BIAS_APSROSFBN, 38
 DAVIS208_CONFIG_BIAS_BIASBUFFER, 38
 DAVIS208_CONFIG_BIAS_COLSELLOWBN, 38
 DAVIS208_CONFIG_BIAS_DACBUFBP, 39
 DAVIS208_CONFIG_BIAS_DIFFBN, 39
 DAVIS208_CONFIG_BIAS_IFREFRBN, 39
 DAVIS208_CONFIG_BIAS_IFTHRBN, 40
 DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN, 40
 DAVIS208_CONFIG_BIAS_LOCALBUFBN, 40
 DAVIS208_CONFIG_BIAS_OFFBN, 41
 DAVIS208_CONFIG_BIAS_ONBN, 41
 DAVIS208_CONFIG_BIAS_PADFOLLBN, 41
 DAVIS208_CONFIG_BIAS_PIXINVBN, 42
 DAVIS208_CONFIG_BIAS_PRBP, 42
 DAVIS208_CONFIG_BIAS_PRSFBP, 42
 DAVIS208_CONFIG_BIAS_READOUTBUFBP, 43
 DAVIS208_CONFIG_BIAS_REFRBP, 43
 DAVIS208_CONFIG_BIAS_REFSSBN, 44
 DAVIS208_CONFIG_BIAS_REFSS, 43
 DAVIS208_CONFIG_BIAS_REGBIASBP, 44
 DAVIS208_CONFIG_BIAS_RESETHIGHPASS, 44
 DAVIS208_CONFIG_BIAS_SSN, 45
 DAVIS208_CONFIG_BIAS_SSP, 45
 DAVIS208_CONFIG_CHIP_AERNAROW, 45
 DAVIS208_CONFIG_CHIP_ANALOGMUX0, 46
 DAVIS208_CONFIG_CHIP_ANALOGMUX1, 46
 DAVIS208_CONFIG_CHIP_ANALOGMUX2, 46
 DAVIS208_CONFIG_CHIP_BIASMUX0, 46
 DAVIS208_CONFIG_CHIP_DIGITALMUX0, 46
 DAVIS208_CONFIG_CHIP_DIGITALMUX1, 46
 DAVIS208_CONFIG_CHIP_DIGITALMUX2, 47
 DAVIS208_CONFIG_CHIP_DIGITALMUX3, 47
 DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER, 47
 DAVIS208_CONFIG_CHIP_RESETCALIBNEU←RON, 47
 DAVIS208_CONFIG_CHIP_RESETTESTPIXEL, 47
 DAVIS208_CONFIG_CHIP_SELECTBIASREFSS, 47
 DAVIS208_CONFIG_CHIP_SELECTGRAYCO←UNTER, 48
 DAVIS208_CONFIG_CHIP_SELECTHIGHPASS, 48
 DAVIS208_CONFIG_CHIP_SELECTPOSFB, 48
 DAVIS208_CONFIG_CHIP_SELECTPREAMPA←VG, 48
 DAVIS208_CONFIG_CHIP_SELECTSENSE, 48
 DAVIS208_CONFIG_CHIP_TYPENCALIBNEU←RON, 48
 DAVIS208_CONFIG_CHIP_USEAOUT, 49
 DAVIS240_CONFIG_BIAS_AEPDBN, 49
 DAVIS240_CONFIG_BIAS_AEPUXBP, 49
 DAVIS240_CONFIG_BIAS_AEPUYBP, 49
 DAVIS240_CONFIG_BIAS_APSCASEPC, 50
 DAVIS240_CONFIG_BIAS_APSOEVERFLOWLE←VELBN, 50
 DAVIS240_CONFIG_BIAS_APSROSFBN, 50
 DAVIS240_CONFIG_BIAS_BIASBUFFER, 50
 DAVIS240_CONFIG_BIAS_DIFFBN, 51
 DAVIS240_CONFIG_BIAS_DIFFCASBNC, 51
 DAVIS240_CONFIG_BIAS_IFREFRBN, 51
 DAVIS240_CONFIG_BIAS_IFTHRBN, 51
 DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN, 52
 DAVIS240_CONFIG_BIAS_LOCALBUFBN, 52
 DAVIS240_CONFIG_BIAS_OFFBN, 52

- DAVIS240_CONFIG_BIAS_ONBN, [52](#)
- DAVIS240_CONFIG_BIAS_PADFOLLB, [53](#)
- DAVIS240_CONFIG_BIAS_PIXINVB, [53](#)
- DAVIS240_CONFIG_BIAS_PRBP, [53](#)
- DAVIS240_CONFIG_BIAS_PRFBP, [53](#)
- DAVIS240_CONFIG_BIAS_REFRBP, [54](#)
- DAVIS240_CONFIG_BIAS_SSN, [54](#)
- DAVIS240_CONFIG_BIAS_SSP, [54](#)
- DAVIS240_CONFIG_CHIP_AERNAROW, [54](#)
- DAVIS240_CONFIG_CHIP_ANALOGMUX0, [55](#)
- DAVIS240_CONFIG_CHIP_ANALOGMUX1, [55](#)
- DAVIS240_CONFIG_CHIP_ANALOGMUX2, [55](#)
- DAVIS240_CONFIG_CHIP_BIASMUX0, [55](#)
- DAVIS240_CONFIG_CHIP_DIGITALMUX0, [55](#)
- DAVIS240_CONFIG_CHIP_DIGITALMUX1, [55](#)
- DAVIS240_CONFIG_CHIP_DIGITALMUX2, [56](#)
- DAVIS240_CONFIG_CHIP_DIGITALMUX3, [56](#)
- DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER, [56](#)
- DAVIS240_CONFIG_CHIP_RESETCALIBNEU↵RON, [56](#)
- DAVIS240_CONFIG_CHIP_RESETESTPIXEL, [56](#)
- DAVIS240_CONFIG_CHIP_SPECIALPIXELCO↵NTROL, [56](#)
- DAVIS240_CONFIG_CHIP_TYPENCALIBNEU↵RON, [57](#)
- DAVIS240_CONFIG_CHIP_USEAOUT, [57](#)
- DAVIS346_CONFIG_BIAS_ADCCOMPBP, [57](#)
- DAVIS346_CONFIG_BIAS_ADCREFHIGH, [57](#)
- DAVIS346_CONFIG_BIAS_ADCREFLOW, [57](#)
- DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE, [58](#)
- DAVIS346_CONFIG_BIAS_AEPDBN, [58](#)
- DAVIS346_CONFIG_BIAS_AEPUXBP, [58](#)
- DAVIS346_CONFIG_BIAS_AEPUYBP, [59](#)
- DAVIS346_CONFIG_BIAS_APSCAS, [59](#)
- DAVIS346_CONFIG_BIAS_APSEVERFLOWLE↵VEL, [59](#)
- DAVIS346_CONFIG_BIAS_APSROSFBN, [60](#)
- DAVIS346_CONFIG_BIAS_BIASBUFFER, [60](#)
- DAVIS346_CONFIG_BIAS_COLSELOWBN, [60](#)
- DAVIS346_CONFIG_BIAS_DACBUFBP, [61](#)
- DAVIS346_CONFIG_BIAS_DIFFBN, [61](#)
- DAVIS346_CONFIG_BIAS_IFREFRBN, [61](#)
- DAVIS346_CONFIG_BIAS_IFTHRBN, [62](#)
- DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN, [62](#)
- DAVIS346_CONFIG_BIAS_LOCALBUFBN, [62](#)
- DAVIS346_CONFIG_BIAS_OFFBN, [63](#)
- DAVIS346_CONFIG_BIAS_ONBN, [63](#)
- DAVIS346_CONFIG_BIAS_PADFOLLB, [63](#)
- DAVIS346_CONFIG_BIAS_PIXINVB, [64](#)
- DAVIS346_CONFIG_BIAS_PRBP, [64](#)
- DAVIS346_CONFIG_BIAS_PRFBP, [64](#)
- DAVIS346_CONFIG_BIAS_READOUTBUFBP, [65](#)
- DAVIS346_CONFIG_BIAS_REFRBP, [65](#)
- DAVIS346_CONFIG_BIAS_SSN, [65](#)
- DAVIS346_CONFIG_BIAS_SSP, [66](#)
- DAVIS346_CONFIG_CHIP_AERNAROW, [66](#)
- DAVIS346_CONFIG_CHIP_ANALOGMUX0, [66](#)
- DAVIS346_CONFIG_CHIP_ANALOGMUX1, [66](#)
- DAVIS346_CONFIG_CHIP_ANALOGMUX2, [67](#)
- DAVIS346_CONFIG_CHIP_BIASMUX0, [67](#)
- DAVIS346_CONFIG_CHIP_DIGITALMUX0, [67](#)
- DAVIS346_CONFIG_CHIP_DIGITALMUX1, [67](#)
- DAVIS346_CONFIG_CHIP_DIGITALMUX2, [67](#)
- DAVIS346_CONFIG_CHIP_DIGITALMUX3, [67](#)
- DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER, [68](#)
- DAVIS346_CONFIG_CHIP_RESETCALIBNEU↵RON, [68](#)
- DAVIS346_CONFIG_CHIP_RESETESTPIXEL, [68](#)
- DAVIS346_CONFIG_CHIP_SELECTGRAYCO↵UNTER, [68](#)
- DAVIS346_CONFIG_CHIP_TESTADC, [68](#)
- DAVIS346_CONFIG_CHIP_TYPENCALIBNEU↵RON, [68](#)
- DAVIS346_CONFIG_CHIP_USEAOUT, [69](#)
- DAVIS640_CONFIG_BIAS_ADCCOMPBP, [69](#)
- DAVIS640_CONFIG_BIAS_ADCREFHIGH, [69](#)
- DAVIS640_CONFIG_BIAS_ADCREFLOW, [69](#)
- DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE, [70](#)
- DAVIS640_CONFIG_BIAS_AEPDBN, [70](#)
- DAVIS640_CONFIG_BIAS_AEPUXBP, [70](#)
- DAVIS640_CONFIG_BIAS_AEPUYBP, [71](#)
- DAVIS640_CONFIG_BIAS_APSCAS, [71](#)
- DAVIS640_CONFIG_BIAS_APSEVERFLOWLE↵VEL, [71](#)
- DAVIS640_CONFIG_BIAS_APSROSFBN, [72](#)
- DAVIS640_CONFIG_BIAS_BIASBUFFER, [72](#)
- DAVIS640_CONFIG_BIAS_COLSELOWBN, [72](#)
- DAVIS640_CONFIG_BIAS_DACBUFBP, [73](#)
- DAVIS640_CONFIG_BIAS_DIFFBN, [73](#)
- DAVIS640_CONFIG_BIAS_IFREFRBN, [73](#)
- DAVIS640_CONFIG_BIAS_IFTHRBN, [74](#)
- DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN, [74](#)
- DAVIS640_CONFIG_BIAS_LOCALBUFBN, [74](#)
- DAVIS640_CONFIG_BIAS_OFFBN, [75](#)
- DAVIS640_CONFIG_BIAS_ONBN, [75](#)
- DAVIS640_CONFIG_BIAS_PADFOLLB, [75](#)
- DAVIS640_CONFIG_BIAS_PIXINVB, [76](#)
- DAVIS640_CONFIG_BIAS_PRBP, [76](#)
- DAVIS640_CONFIG_BIAS_PRFBP, [76](#)
- DAVIS640_CONFIG_BIAS_READOUTBUFBP, [77](#)
- DAVIS640_CONFIG_BIAS_REFRBP, [77](#)
- DAVIS640_CONFIG_BIAS_SSN, [77](#)
- DAVIS640_CONFIG_BIAS_SSP, [78](#)
- DAVIS640_CONFIG_CHIP_AERNAROW, [78](#)
- DAVIS640_CONFIG_CHIP_ANALOGMUX0, [78](#)
- DAVIS640_CONFIG_CHIP_ANALOGMUX1, [78](#)
- DAVIS640_CONFIG_CHIP_ANALOGMUX2, [79](#)
- DAVIS640_CONFIG_CHIP_BIASMUX0, [79](#)

- DAVIS640_CONFIG_CHIP_DIGITALMUX0, [79](#)
- DAVIS640_CONFIG_CHIP_DIGITALMUX1, [79](#)
- DAVIS640_CONFIG_CHIP_DIGITALMUX2, [79](#)
- DAVIS640_CONFIG_CHIP_DIGITALMUX3, [79](#)
- DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER, [80](#)
- DAVIS640_CONFIG_CHIP_RESETCALIBNEU↔RON, [80](#)
- DAVIS640_CONFIG_CHIP_RESETTESTPIXEL, [80](#)
- DAVIS640_CONFIG_CHIP_SELECTGRAYCO↔UNTER, [80](#)
- DAVIS640_CONFIG_CHIP_TESTADC, [80](#)
- DAVIS640_CONFIG_CHIP_TYPENCALIBNEU↔RON, [80](#)
- DAVIS640_CONFIG_CHIP_USEAOUT, [81](#)
- DAVIS_APS_ROI_REGIONS_MAX, [81](#)
- DAVIS_CHIP_DAVIS128, [81](#)
- DAVIS_CHIP_DAVIS208, [81](#)
- DAVIS_CHIP_DAVIS240A, [81](#)
- DAVIS_CHIP_DAVIS240B, [81](#)
- DAVIS_CHIP_DAVIS240C, [81](#)
- DAVIS_CHIP_DAVIS346A, [82](#)
- DAVIS_CHIP_DAVIS346B, [82](#)
- DAVIS_CHIP_DAVIS346C, [82](#)
- DAVIS_CHIP_DAVIS640, [82](#)
- DAVIS_CHIP_DAVISRGB, [82](#)
- DAVIS_CONFIG_APS_ADC_TEST_MODE, [82](#)
- DAVIS_CONFIG_APS_AUTOEXPOSURE, [83](#)
- DAVIS_CONFIG_APS_COLOR_FILTER, [83](#)
- DAVIS_CONFIG_APS_COLUMN_SETTLE, [83](#)
- DAVIS_CONFIG_APS_END_COLUMN_0, [83](#)
- DAVIS_CONFIG_APS_END_COLUMN_1, [83](#)
- DAVIS_CONFIG_APS_END_COLUMN_2, [83](#)
- DAVIS_CONFIG_APS_END_COLUMN_3, [83](#)
- DAVIS_CONFIG_APS_END_ROW_0, [84](#)
- DAVIS_CONFIG_APS_END_ROW_1, [84](#)
- DAVIS_CONFIG_APS_END_ROW_2, [84](#)
- DAVIS_CONFIG_APS_END_ROW_3, [84](#)
- DAVIS_CONFIG_APS_EXPOSURE, [84](#)
- DAVIS_CONFIG_APS_FRAME_DELAY, [84](#)
- DAVIS_CONFIG_APS_GLOBAL_SHUTTER, [84](#)
- DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTT↔ER, [85](#)
- DAVIS_CONFIG_APS_HAS_INTERNAL_ADC, [85](#)
- DAVIS_CONFIG_APS_HAS_QUAD_ROI, [85](#)
- DAVIS_CONFIG_APS_NULL_SETTLE, [85](#)
- DAVIS_CONFIG_APS_ORIENTATION_INFO, [85](#)
- DAVIS_CONFIG_APS_RAMP_RESET, [85](#)
- DAVIS_CONFIG_APS_RAMP_SHORT_RESET, [86](#)
- DAVIS_CONFIG_APS_RESET_READ, [86](#)
- DAVIS_CONFIG_APS_RESET_SETTLE, [86](#)
- DAVIS_CONFIG_APS_ROI0_ENABLED, [86](#)
- DAVIS_CONFIG_APS_ROI1_ENABLED, [86](#)
- DAVIS_CONFIG_APS_ROI2_ENABLED, [86](#)
- DAVIS_CONFIG_APS_ROI3_ENABLED, [86](#)
- DAVIS_CONFIG_APS_ROW_SETTLE, [87](#)
- DAVIS_CONFIG_APS_RUN, [87](#)
- DAVIS_CONFIG_APS_SAMPLE_ENABLE, [87](#)
- DAVIS_CONFIG_APS_SAMPLE_SETTLE, [87](#)
- DAVIS_CONFIG_APS_SIZE_COLUMNS, [87](#)
- DAVIS_CONFIG_APS_SIZE_ROWS, [87](#)
- DAVIS_CONFIG_APS_SNAPSHOT, [87](#)
- DAVIS_CONFIG_APS_START_COLUMN_0, [88](#)
- DAVIS_CONFIG_APS_START_COLUMN_1, [88](#)
- DAVIS_CONFIG_APS_START_COLUMN_2, [88](#)
- DAVIS_CONFIG_APS_START_COLUMN_3, [88](#)
- DAVIS_CONFIG_APS_START_ROW_0, [88](#)
- DAVIS_CONFIG_APS_START_ROW_1, [88](#)
- DAVIS_CONFIG_APS_START_ROW_2, [89](#)
- DAVIS_CONFIG_APS_START_ROW_3, [89](#)
- DAVIS_CONFIG_APS_WAIT_ON_TRANSFER↔_STALL, [89](#)
- DAVIS_CONFIG_APS, [82](#)
- DAVIS_CONFIG_BIAS, [89](#)
- DAVIS_CONFIG_CHIP, [89](#)
- DAVIS_CONFIG_DDRAER_ACK_DELAY, [90](#)
- DAVIS_CONFIG_DDRAER_REQ_DELAY, [90](#)
- DAVIS_CONFIG_DDRAER_RUN, [90](#)
- DAVIS_CONFIG_DDRAER, [89](#)
- DAVIS_CONFIG_DVS_ACK_DELAY_COLUMN, [90](#)
- DAVIS_CONFIG_DVS_ACK_DELAY_ROW, [90](#)
- DAVIS_CONFIG_DVS_ACK_EXTENSION_CO↔LUMN, [90](#)
- DAVIS_CONFIG_DVS_ACK_EXTENSION_ROW, [91](#)
- DAVIS_CONFIG_DVS_EXTERNAL_AER_CON↔TROL, [91](#)
- DAVIS_CONFIG_DVS_FILTER_BACKGROUN↔D_ACTIVITY_TIME, [91](#)
- DAVIS_CONFIG_DVS_FILTER_BACKGROUN↔D_ACTIVITY, [91](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COL↔UMN, [91](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW, [91](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COL↔UMN, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COL↔UMN, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COL↔UMN, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COL↔UMN, [92](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW, [93](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COL↔UMN, [93](#)

- DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW, 93
- DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COL↔ UMN, 93
- DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW, 93
- DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COL↔ UMN, 93
- DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW, 93
- DAVIS_CONFIG_DVS_FILTER_PIXEL_AUTO↔ TRAIN, 94
- DAVIS_CONFIG_DVS_FILTER_REFRACTORY↔ _PERIOD_TIME, 94
- DAVIS_CONFIG_DVS_FILTER_REFRACTORY↔ _PERIOD, 94
- DAVIS_CONFIG_DVS_FILTER_ROI_END_CO↔ LUMN, 94
- DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW, 94
- DAVIS_CONFIG_DVS_FILTER_ROI_START_C↔ OLUMN, 94
- DAVIS_CONFIG_DVS_FILTER_ROI_START_R↔ OW, 94
- DAVIS_CONFIG_DVS_FILTER_ROW_ONLY↔ EVENTS, 95
- DAVIS_CONFIG_DVS_HAS_BACKGROUND↔ ACTIVITY_FILTER, 95
- DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER, 95
- DAVIS_CONFIG_DVS_HAS_ROI_FILTER, 95
- DAVIS_CONFIG_DVS_HAS_STATISTICS, 95
- DAVIS_CONFIG_DVS_HAS_TEST_EVENT_G↔ ENERATOR, 95
- DAVIS_CONFIG_DVS_ORIENTATION_INFO, 96
- DAVIS_CONFIG_DVS_RUN, 96
- DAVIS_CONFIG_DVS_SIZE_COLUMNS, 96
- DAVIS_CONFIG_DVS_SIZE_ROWS, 96
- DAVIS_CONFIG_DVS_STATISTICS_EVENTS↔ COLUMN, 96
- DAVIS_CONFIG_DVS_STATISTICS_EVENTS↔ DROPPED, 96
- DAVIS_CONFIG_DVS_STATISTICS_EVENTS↔ ROW, 97
- DAVIS_CONFIG_DVS_STATISTICS_FILTERE↔ D_BACKGROUND_ACTIVITY, 97
- DAVIS_CONFIG_DVS_STATISTICS_FILTERE↔ D_PIXELS, 97
- DAVIS_CONFIG_DVS_STATISTICS_FILTERE↔ D_REFRACTORY_PERIOD, 97
- DAVIS_CONFIG_DVS_TEST_EVENT_GENER↔ ATOR_ENABLE, 97
- DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER↔ _STALL, 97
- DAVIS_CONFIG_DVS, 90
- DAVIS_CONFIG_EXTINPUT_DETECT_FALLIN↔ G_EDGES1, 98
- DAVIS_CONFIG_EXTINPUT_DETECT_FALLIN↔ G_EDGES2, 98
- DAVIS_CONFIG_EXTINPUT_DETECT_FALLIN↔ G_EDGES, 98
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE↔ _LENGTH1, 98
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE↔ _LENGTH2, 99
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE↔ _LENGTH, 98
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE↔ _POLARITY1, 99
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE↔ _POLARITY2, 99
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE↔ _POLARITY, 99
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE↔ S1, 99
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE↔ S2, 100
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSES, 99
- DAVIS_CONFIG_EXTINPUT_DETECT_RISING↔ _EDGES1, 100
- DAVIS_CONFIG_EXTINPUT_DETECT_RISING↔ _EDGES2, 100
- DAVIS_CONFIG_EXTINPUT_DETECT_RISING↔ _EDGES, 100
- DAVIS_CONFIG_EXTINPUT_GENERATE_INJ↔ ECT_ON_FALLING_EDGE, 100
- DAVIS_CONFIG_EXTINPUT_GENERATE_INJ↔ ECT_ON_RISING_EDGE, 100
- DAVIS_CONFIG_EXTINPUT_GENERATE_PUL↔ SE_INTERVAL, 101
- DAVIS_CONFIG_EXTINPUT_GENERATE_PUL↔ SE_LENGTH, 101
- DAVIS_CONFIG_EXTINPUT_GENERATE_PUL↔ SE_POLARITY, 101
- DAVIS_CONFIG_EXTINPUT_GENERATE_US↔ E_CUSTOM_SIGNAL, 101
- DAVIS_CONFIG_EXTINPUT_HAS_EXTRA_DE↔ TECTORS, 101
- DAVIS_CONFIG_EXTINPUT_HAS_GENERAT↔ OR, 101
- DAVIS_CONFIG_EXTINPUT_RUN_DETECTO↔ R1, 102
- DAVIS_CONFIG_EXTINPUT_RUN_DETECTO↔ R2, 102
- DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR, 102
- DAVIS_CONFIG_EXTINPUT_RUN_GENERAT↔ OR, 102
- DAVIS_CONFIG_EXTINPUT, 98
- DAVIS_CONFIG_IMU_ACCEL_DLPF, 102
- DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE, 103
- DAVIS_CONFIG_IMU_ACCEL_STANDBY, 103
- DAVIS_CONFIG_IMU_GYRO_DLPF, 103
- DAVIS_CONFIG_IMU_GYRO_FULL_SCALE, 103
- DAVIS_CONFIG_IMU_GYRO_STANDBY, 103

- DAVIS_CONFIG_IMU_LP_CYCLE, [104](#)
- DAVIS_CONFIG_IMU_LP_WAKEUP, [104](#)
- DAVIS_CONFIG_IMU_ORIENTATION_INFO, [104](#)
- DAVIS_CONFIG_IMU_RUN, [104](#)
- DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER, [104](#)
- DAVIS_CONFIG_IMU_TEMP_STANDBY, [104](#)
- DAVIS_CONFIG_IMU, [102](#)
- DAVIS_CONFIG_MUX_DROP_APS_ON_TRANSFER_STALL, [105](#)
- DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL, [105](#)
- DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL, [105](#)
- DAVIS_CONFIG_MUX_DROP_IMU_ON_TRANSFER_STALL, [105](#)
- DAVIS_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE, [105](#)
- DAVIS_CONFIG_MUX_HAS_STATISTICS, [106](#)
- DAVIS_CONFIG_MUX_RUN, [106](#)
- DAVIS_CONFIG_MUX_STATISTICS_APS_DROPPED, [106](#)
- DAVIS_CONFIG_MUX_STATISTICS_DVS_DROPPED, [106](#)
- DAVIS_CONFIG_MUX_STATISTICS_EXTINPUT_DROPPED, [106](#)
- DAVIS_CONFIG_MUX_STATISTICS_IMU_DROPPED, [106](#)
- DAVIS_CONFIG_MUX_TIMESTAMP_RESET, [107](#)
- DAVIS_CONFIG_MUX_TIMESTAMP_RUN, [107](#)
- DAVIS_CONFIG_MUX, [105](#)
- DAVIS_CONFIG_SYSINFO_ADC_CLOCK, [107](#)
- DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER, [107](#)
- DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER, [107](#)
- DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK, [108](#)
- DAVIS_CONFIG_SYSINFO_LOGIC_VERSION, [108](#)
- DAVIS_CONFIG_SYSINFO, [107](#)
- DAVIS_CONFIG_USB_EARLY_PACKET_DELAY, [108](#)
- DAVIS_CONFIG_USB_RUN, [108](#)
- DAVIS_CONFIG_USB, [108](#)
- DAVISRGB_CONFIG_APS_GSFDRESET, [108](#)
- DAVISRGB_CONFIG_APS_GSPDRESET, [109](#)
- DAVISRGB_CONFIG_APS_GSRESETFALL, [109](#)
- DAVISRGB_CONFIG_APS_GSTXFALL, [109](#)
- DAVISRGB_CONFIG_APS_RSFDSETTLE, [109](#)
- DAVISRGB_CONFIG_APS_TRANSFER, [109](#)
- DAVISRGB_CONFIG_BIAS_ADCCOMPBP, [109](#)
- DAVISRGB_CONFIG_BIAS_ADCREFHIGH, [110](#)
- DAVISRGB_CONFIG_BIAS_ADCREFLOW, [110](#)
- DAVISRGB_CONFIG_BIAS_ADCTESTVOLTAGE, [110](#)
- DAVISRGB_CONFIG_BIAS_AEPDBN, [111](#)
- DAVISRGB_CONFIG_BIAS_AEPUXBP, [111](#)
- DAVISRGB_CONFIG_BIAS_AEPUYBP, [111](#)
- DAVISRGB_CONFIG_BIAS_APSCAS, [112](#)
- DAVISRGB_CONFIG_BIAS_APSROSFBN, [112](#)
- DAVISRGB_CONFIG_BIAS_ARRAYBIASBUFFER, [112](#)
- DAVISRGB_CONFIG_BIAS_ARRAYLOGICBUFFER, [113](#)
- DAVISRGB_CONFIG_BIAS_BIASBUFFER, [113](#)
- DAVISRGB_CONFIG_BIAS_DACBUFBP, [113](#)
- DAVISRGB_CONFIG_BIAS_DIFFBN, [114](#)
- DAVISRGB_CONFIG_BIAS_FALLTIMEBN, [114](#)
- DAVISRGB_CONFIG_BIAS_GND07, [114](#)
- DAVISRGB_CONFIG_BIAS_IFREFRBN, [115](#)
- DAVISRGB_CONFIG_BIAS_IFTHRBN, [115](#)
- DAVISRGB_CONFIG_BIAS_LCOLTIMEOUTBN, [115](#)
- DAVISRGB_CONFIG_BIAS_LOCALBUFBN, [116](#)
- DAVISRGB_CONFIG_BIAS_OFFBN, [116](#)
- DAVISRGB_CONFIG_BIAS_ONBN, [116](#)
- DAVISRGB_CONFIG_BIAS_OVG1LO, [117](#)
- DAVISRGB_CONFIG_BIAS_OVG2LO, [117](#)
- DAVISRGB_CONFIG_BIAS_PADFOLLBN, [117](#)
- DAVISRGB_CONFIG_BIAS_PIXINBN, [118](#)
- DAVISRGB_CONFIG_BIAS_PRBP, [118](#)
- DAVISRGB_CONFIG_BIAS_PRSFBP, [118](#)
- DAVISRGB_CONFIG_BIAS_READOUTBUFBP, [119](#)
- DAVISRGB_CONFIG_BIAS_REFRBP, [119](#)
- DAVISRGB_CONFIG_BIAS_RISETIMEBP, [119](#)
- DAVISRGB_CONFIG_BIAS_SSN, [120](#)
- DAVISRGB_CONFIG_BIAS_SSP, [120](#)
- DAVISRGB_CONFIG_BIAS_TX2OVG2HI, [120](#)
- DAVISRGB_CONFIG_CHIP_ADJUSTOVG1LO, [121](#)
- DAVISRGB_CONFIG_CHIP_ADJUSTOVG2LO, [121](#)
- DAVISRGB_CONFIG_CHIP_ADJUSTTX2OVG2HI, [121](#)
- DAVISRGB_CONFIG_CHIP_AERNAROW, [121](#)
- DAVISRGB_CONFIG_CHIP_ANALOGMUX0, [121](#)
- DAVISRGB_CONFIG_CHIP_ANALOGMUX1, [122](#)
- DAVISRGB_CONFIG_CHIP_ANALOGMUX2, [122](#)
- DAVISRGB_CONFIG_CHIP_BIASMUX0, [122](#)
- DAVISRGB_CONFIG_CHIP_DIGITALMUX0, [122](#)
- DAVISRGB_CONFIG_CHIP_DIGITALMUX1, [122](#)
- DAVISRGB_CONFIG_CHIP_DIGITALMUX2, [122](#)
- DAVISRGB_CONFIG_CHIP_DIGITALMUX3, [123](#)
- DAVISRGB_CONFIG_CHIP_RESETCALIBNEURON, [123](#)
- DAVISRGB_CONFIG_CHIP_RESETTESTPIXEL, [123](#)
- DAVISRGB_CONFIG_CHIP_SELECTGRAYCONINTER, [123](#)
- DAVISRGB_CONFIG_CHIP_TESTADC, [123](#)
- DAVISRGB_CONFIG_CHIP_TYPENCALIBNEURON, [123](#)
- DAVISRGB_CONFIG_CHIP_USEAOUT, [124](#)
- IS_DAVIS128, [124](#)

- IS_DAVIS208, [124](#)
- IS_DAVIS240, [124](#)
- IS_DAVIS240A, [124](#)
- IS_DAVIS240B, [124](#)
- IS_DAVIS240C, [125](#)
- IS_DAVIS346, [125](#)
- IS_DAVIS346A, [125](#)
- IS_DAVIS346B, [125](#)
- IS_DAVIS346C, [125](#)
- IS_DAVIS640, [125](#)
- IS_DAVISRGB, [126](#)
- device.h
 - CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING, [131](#)
 - CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE, [131](#)
 - CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS, [131](#)
 - CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS, [131](#)
 - CAER_HOST_CONFIG_DATAEXCHANGE, [131](#)
 - CAER_HOST_CONFIG_LOG_LEVEL, [132](#)
 - CAER_HOST_CONFIG_LOG, [131](#)
 - CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL, [132](#)
 - CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_PACKET_SIZE, [132](#)
 - CAER_HOST_CONFIG_PACKETS, [132](#)
 - CAER_SUPPORTED_DEVICES_NUMBER, [132](#)
 - caerDeviceClose, [133](#)
 - caerDeviceConfigGet, [133](#)
 - caerDeviceConfigGet64, [134](#)
 - caerDeviceConfigSet, [134](#)
 - caerDeviceDataGet, [135](#)
 - caerDeviceDataStart, [135](#)
 - caerDeviceDataStop, [136](#)
 - caerDeviceHandle, [132](#)
 - caerDeviceSendDefaultConfig, [136](#)
- device_discover.h
 - CAER_DEVICE_DISCOVER_ALL, [137](#)
 - caerDeviceDiscover, [137](#)
 - caerDeviceDiscoverOpen, [138](#)
 - caerDeviceDiscoveryResult, [137](#)
- deviceString
 - caer_davis_info, [8](#)
 - caer_dvs128_info, [10](#)
 - caer_dynapse_info, [11](#)
 - caer_edvs_info, [12](#)
- devices/davis.h, [13](#)
- devices/device.h, [130](#)
- devices/device_discover.h, [136](#)
- devices/dvs128.h, [138](#)
- devices/dynapse.h, [142](#)
- devices/edvs.h, [167](#)
- devices/serial.h, [171](#)
- devices/usb.h, [173](#)
- dvs128.h
 - CAER_DEVICE_DVS128, [139](#)
 - caerDVS128InfoGet, [142](#)
 - DVS128_CONFIG_BIAS_CAS, [139](#)
 - DVS128_CONFIG_BIAS_DIFFOFF, [140](#)
 - DVS128_CONFIG_BIAS_DIFFON, [140](#)
 - DVS128_CONFIG_BIAS_DIFF, [140](#)
 - DVS128_CONFIG_BIAS_FOLL, [140](#)
 - DVS128_CONFIG_BIAS_INJGND, [140](#)
 - DVS128_CONFIG_BIAS_PUX, [140](#)
 - DVS128_CONFIG_BIAS_PUY, [141](#)
 - DVS128_CONFIG_BIAS_PR, [140](#)
 - DVS128_CONFIG_BIAS_REFR, [141](#)
 - DVS128_CONFIG_BIAS_REQPD, [141](#)
 - DVS128_CONFIG_BIAS_REQ, [141](#)
 - DVS128_CONFIG_BIAS, [139](#)
 - DVS128_CONFIG_DVS_ARRAY_RESET, [141](#)
 - DVS128_CONFIG_DVS_RUN, [141](#)
 - DVS128_CONFIG_DVS_TIMESTAMP_RESET, [142](#)
 - DVS128_CONFIG_DVS_TS_MASTER, [142](#)
 - DVS128_CONFIG_DVS, [141](#)
- dvs_noise.h
 - CAER_FILTER_DVS_BACKGROUND_ACTIVITY_CHECK_POLARITY, [425](#)
 - CAER_FILTER_DVS_BACKGROUND_ACTIVITY_ENABLE, [426](#)
 - CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS, [426](#)
 - CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MAX, [426](#)
 - CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MIN, [426](#)
 - CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TIME, [426](#)
 - CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TWO_LEVELS, [426](#)
 - CAER_FILTER_DVS_HOTPIXEL_COUNT, [426](#)
 - CAER_FILTER_DVS_HOTPIXEL_ENABLE, [427](#)
 - CAER_FILTER_DVS_HOTPIXEL_LEARN, [427](#)
 - CAER_FILTER_DVS_HOTPIXEL_STATISTICS, [427](#)
 - CAER_FILTER_DVS_HOTPIXEL_TIME, [427](#)
 - CAER_FILTER_DVS_LOG_LEVEL, [427](#)
 - CAER_FILTER_DVS_REFRACTORY_PERIOD_ENABLE, [427](#)
 - CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS, [427](#)
 - CAER_FILTER_DVS_REFRACTORY_PERIOD_TIME, [428](#)
 - CAER_FILTER_DVS_RESET, [428](#)
 - caerFilterDVSNoise, [428](#)
 - caerFilterDVSNoiseApply, [428](#)
 - caerFilterDVSNoiseConfigGet, [429](#)
 - caerFilterDVSNoiseConfigSet, [429](#)
 - caerFilterDVSNoiseDestroy, [429](#)
 - caerFilterDVSNoiseGetHotPixels, [430](#)
 - caerFilterDVSNoiseInitialize, [430](#)
 - caerFilterDVSPixel, [428](#)
- dynapse.h

- CAER_DEVICE_DYNAPSE, 147
- caerBiasDynapseGenerate, 159
- caerBiasDynapseParse, 159
- caerDynapseCoreAddrToNeuronId, 160
- caerDynapseCoreXYToNeuronId, 160
- caerDynapseGenerateCamBits, 160
- caerDynapseGenerateSramBits, 161
- caerDynapseInfoGet, 162
- caerDynapseSendDataToUSB, 162
- caerDynapseSpikeEventFromXY, 162
- caerDynapseSpikeEventGetX, 163
- caerDynapseSpikeEventGetY, 163
- caerDynapseWriteCam, 164
- caerDynapseWritePoissonSpikeRate, 164
- caerDynapseWriteSram, 165
- caerDynapseWriteSramWords, 166
- caerDynapseWriteSramN, 165
- DYNAPSE_CHIP_DYNAPSE, 147
- DYNAPSE_CONFIG_AER_ACK_DELAY, 148
- DYNAPSE_CONFIG_AER_ACK_EXTENSION, 148
- DYNAPSE_CONFIG_AER_EXTERNAL_AER_↵
CONTROL, 148
- DYNAPSE_CONFIG_AER_HAS_STATISTICS, 148
- DYNAPSE_CONFIG_AER_RUN, 148
- DYNAPSE_CONFIG_AER_STATISTICS_EVEN_↵
TS_DROPPED, 149
- DYNAPSE_CONFIG_AER_STATISTICS_EVEN_↵
TS, 148
- DYNAPSE_CONFIG_AER_WAIT_ON_TRANSF_↵
ER_STALL, 149
- DYNAPSE_CONFIG_AER, 148
- DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_↵
_P, 149
- DYNAPSE_CONFIG_CHIP_CONTENT, 149
- DYNAPSE_CONFIG_CHIP_ID, 149
- DYNAPSE_CONFIG_CHIP_REQ_DELAY, 150
- DYNAPSE_CONFIG_CHIP_REQ_EXTENSION, 150
- DYNAPSE_CONFIG_CHIP_RUN, 150
- DYNAPSE_CONFIG_CHIP, 149
- DYNAPSE_CONFIG_CLEAR_CAM, 150
- DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY, 150
- DYNAPSE_CONFIG_DEFAULT_SRAM, 150
- DYNAPSE_CONFIG_MONITOR_NEU, 150
- DYNAPSE_CONFIG_MUX_DROP_AER_ON_T_↵
RANSFER_STALL, 151
- DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIA_↵
S_ENABLE, 151
- DYNAPSE_CONFIG_MUX_HAS_STATISTICS, 151
- DYNAPSE_CONFIG_MUX_RUN, 151
- DYNAPSE_CONFIG_MUX_STATISTICS_AER_↵
DROPPED, 151
- DYNAPSE_CONFIG_MUX_TIMESTAMP_RES_↵
ET, 152
- DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN, 152
- DYNAPSE_CONFIG_MUX, 151
- DYNAPSE_CONFIG_POISSONSPIKEGEN_CH_↵
IPID, 152
- DYNAPSE_CONFIG_POISSONSPIKEGEN_RUN, 152
- DYNAPSE_CONFIG_POISSONSPIKEGEN_W_↵
RITEADDRESS, 152
- DYNAPSE_CONFIG_POISSONSPIKEGEN_W_↵
RITEDATA, 152
- DYNAPSE_CONFIG_POISSONSPIKEGEN, 152
- DYNAPSE_CONFIG_SPIKEGEN_BASEADDR, 153
- DYNAPSE_CONFIG_SPIKEGEN_ISIBASE, 153
- DYNAPSE_CONFIG_SPIKEGEN_ISI, 153
- DYNAPSE_CONFIG_SPIKEGEN_REPEAT, 153
- DYNAPSE_CONFIG_SPIKEGEN_RUN, 153
- DYNAPSE_CONFIG_SPIKEGEN_STIMCOUNT, 153
- DYNAPSE_CONFIG_SPIKEGEN_VARMODE, 154
- DYNAPSE_CONFIG_SPIKEGEN, 153
- DYNAPSE_CONFIG_SRAM_ADDRESS, 154
- DYNAPSE_CONFIG_SRAM_BURSTMODE, 154
- DYNAPSE_CONFIG_SRAM_DIRECTION_POS, 154
- DYNAPSE_CONFIG_SRAM_READDATA, 155
- DYNAPSE_CONFIG_SRAM_READ, 154
- DYNAPSE_CONFIG_SRAM_RWCOMMAND, 155
- DYNAPSE_CONFIG_SRAM_WRITEDATA, 155
- DYNAPSE_CONFIG_SRAM_WRITE, 155
- DYNAPSE_CONFIG_SRAM, 154
- DYNAPSE_CONFIG_SYNAPSERECONFIG_C_↵
HIPSELECT, 155
- DYNAPSE_CONFIG_SYNAPSERECONFIG_G_↵
LOBALKERNEL, 156
- DYNAPSE_CONFIG_SYNAPSERECONFIG_R_↵
UN, 156
- DYNAPSE_CONFIG_SYNAPSERECONFIG_S_↵
RAMBASEADDR, 156
- DYNAPSE_CONFIG_SYNAPSERECONFIG_U_↵
SESRAMKERNELS, 156
- DYNAPSE_CONFIG_SYNAPSERECONFIG, 155
- DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTI_↵
FIER, 156
- DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_M_↵
ASTER, 157
- DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK, 157
- DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSI_↵
ON, 157
- DYNAPSE_CONFIG_SYSINFO, 156
- DYNAPSE_CONFIG_TAU1_RESET, 157
- DYNAPSE_CONFIG_TAU2_RESET, 157
- DYNAPSE_CONFIG_TAU2_SET, 157
- DYNAPSE_CONFIG_USB_EARLY_PACKET_D_↵
ELAY, 158

- DYNAPSE_CONFIG_USB_RUN, [158](#)
- DYNAPSE_CONFIG_USB, [158](#)
- DYNAPSE_X4BOARD_COREX, [158](#)
- DYNAPSE_X4BOARD_COREY, [158](#)
- DYNAPSE_X4BOARD_NEUX, [158](#)
- DYNAPSE_X4BOARD_NEUY, [159](#)
- DYNAPSE_X4BOARD_NUMCHIPS, [159](#)
- EAR_CHANNEL_MASK
 - ear.h, [209](#)
- EAR_CHANNEL_SHIFT
 - ear.h, [210](#)
- EAR_FILTER_MASK
 - ear.h, [210](#)
- EAR_FILTER_SHIFT
 - ear.h, [210](#)
- EAR_MASK
 - ear.h, [210](#)
- EAR_NEURON_MASK
 - ear.h, [210](#)
- EAR_NEURON_SHIFT
 - ear.h, [210](#)
- EAR_SHIFT
 - ear.h, [211](#)
- EDVS_CONFIG_BIAS_CAS
 - edvs.h, [168](#)
- EDVS_CONFIG_BIAS_DIFFOFF
 - edvs.h, [168](#)
- EDVS_CONFIG_BIAS_DIFFON
 - edvs.h, [168](#)
- EDVS_CONFIG_BIAS_DIFF
 - edvs.h, [168](#)
- EDVS_CONFIG_BIAS_FOLL
 - edvs.h, [168](#)
- EDVS_CONFIG_BIAS_INJGND
 - edvs.h, [169](#)
- EDVS_CONFIG_BIAS_PUX
 - edvs.h, [169](#)
- EDVS_CONFIG_BIAS_PUY
 - edvs.h, [169](#)
- EDVS_CONFIG_BIAS_PR
 - edvs.h, [169](#)
- EDVS_CONFIG_BIAS_REFR
 - edvs.h, [169](#)
- EDVS_CONFIG_BIAS_REQPD
 - edvs.h, [169](#)
- EDVS_CONFIG_BIAS_REQ
 - edvs.h, [169](#)
- EDVS_CONFIG_BIAS
 - edvs.h, [168](#)
- EDVS_CONFIG_DVS_RUN
 - edvs.h, [170](#)
- EDVS_CONFIG_DVS_TIMESTAMP_RESET
 - edvs.h, [170](#)
- EDVS_CONFIG_DVS
 - edvs.h, [170](#)
- ear.h
 - CAER_EAR_CONST_ITERATOR_ALL_START, [206](#)
- CAER_EAR_CONST_ITERATOR_VALID_STA↔RT, [206](#)
- CAER_EAR_CONST_REVERSE_ITERATOR↔ALL_START, [207](#)
- CAER_EAR_CONST_REVERSE_ITERATOR↔VALID_START, [207](#)
- CAER_EAR_ITERATOR_ALL_END, [207](#)
- CAER_EAR_ITERATOR_ALL_START, [208](#)
- CAER_EAR_ITERATOR_VALID_END, [208](#)
- CAER_EAR_ITERATOR_VALID_START, [208](#)
- CAER_EAR_REVERSE_ITERATOR_ALL_END, [208](#)
- CAER_EAR_REVERSE_ITERATOR_ALL_STA↔RT, [209](#)
- CAER_EAR_REVERSE_ITERATOR_VALID_E↔ND, [209](#)
- CAER_EAR_REVERSE_ITERATOR_VALID_S↔TART, [209](#)
- caerEarEvent, [211](#)
- caerEarEventGetChannel, [211](#)
- caerEarEventGetEar, [212](#)
- caerEarEventGetTimestamp, [212](#)
- caerEarEventGetTimestamp64, [212](#)
- caerEarEventInvalidate, [213](#)
- caerEarEventsValid, [213](#)
- caerEarEventPacket, [211](#)
- caerEarEventPacketAllocate, [214](#)
- caerEarEventPacketFromPacketHeader, [214](#)
- caerEarEventPacketFromPacketHeaderConst, [214](#)
- caerEarEventPacketGetEvent, [216](#)
- caerEarEventPacketGetEventConst, [216](#)
- caerEarEventSetChannel, [216](#)
- caerEarEventSetEar, [217](#)
- caerEarEventSetTimestamp, [217](#)
- caerEarEventValidate, [217](#)
- EAR_CHANNEL_MASK, [209](#)
- EAR_CHANNEL_SHIFT, [210](#)
- EAR_FILTER_MASK, [210](#)
- EAR_FILTER_SHIFT, [210](#)
- EAR_MASK, [210](#)
- EAR_NEURON_MASK, [210](#)
- EAR_NEURON_SHIFT, [210](#)
- EAR_SHIFT, [211](#)
- PACKED_STRUCT, [218](#)
- edvs.h
 - CAER_DEVICE_EDVS, [168](#)
 - caerEDVSInfoGet, [170](#)
 - EDVS_CONFIG_BIAS_CAS, [168](#)
 - EDVS_CONFIG_BIAS_DIFFOFF, [168](#)
 - EDVS_CONFIG_BIAS_DIFFON, [168](#)
 - EDVS_CONFIG_BIAS_DIFF, [168](#)
 - EDVS_CONFIG_BIAS_FOLL, [168](#)
 - EDVS_CONFIG_BIAS_INJGND, [169](#)
 - EDVS_CONFIG_BIAS_PUX, [169](#)
 - EDVS_CONFIG_BIAS_PUY, [169](#)
 - EDVS_CONFIG_BIAS_PR, [169](#)
 - EDVS_CONFIG_BIAS_REFR, [169](#)
 - EDVS_CONFIG_BIAS_REQPD, [169](#)

- EDVS_CONFIG_BIAS_REQ, 169
- EDVS_CONFIG_BIAS, 168
- EDVS_CONFIG_DVS_RUN, 170
- EDVS_CONFIG_DVS_TIMESTAMP_RESET, 170
- EDVS_CONFIG_DVS, 170
- events/common.h, 174
- events/config.h, 192
- events/ear.h, 205
- events/frame.h, 218
- events/imu6.h, 248
- events/imu9.h, 263
- events/matrix4x4.h, 281
- events/packetContainer.h, 307
- events/point1d.h, 315
- events/point2d.h, 328
- events/point3d.h, 341
- events/point4d.h, 356
- events/polarity.h, 371
- events/sample.h, 384
- events/special.h, 395
- events/spike.h, 411
- FRAME_COLOR_CHANNELS_MASK
 - frame.h, 224
- FRAME_COLOR_CHANNELS_SHIFT
 - frame.h, 225
- FRAME_COLOR_FILTER_MASK
 - frame.h, 225
- FRAME_COLOR_FILTER_SHIFT
 - frame.h, 225
- FRAME_ROI_IDENTIFIER_MASK
 - frame.h, 225
- FRAME_ROI_IDENTIFIER_SHIFT
 - frame.h, 225
- filters/dvs_noise.h, 424
- frame.h
 - CAER_FRAME_CONST_ITERATOR_ALL_START, 221
 - CAER_FRAME_CONST_ITERATOR_VALID_START, 221
 - CAER_FRAME_CONST_REVERSE_ITERATOR_ALL_START, 221
 - CAER_FRAME_CONST_REVERSE_ITERATOR_VALID_START, 222
 - CAER_FRAME_ITERATOR_ALL_END, 222
 - CAER_FRAME_ITERATOR_ALL_START, 222
 - CAER_FRAME_ITERATOR_VALID_END, 223
 - CAER_FRAME_ITERATOR_VALID_START, 223
 - CAER_FRAME_REVERSE_ITERATOR_ALL_END, 223
 - CAER_FRAME_REVERSE_ITERATOR_ALL_START, 223
 - CAER_FRAME_REVERSE_ITERATOR_VALID_END, 224
 - CAER_FRAME_REVERSE_ITERATOR_VALID_START, 224
 - caer_frame_event_color_channels, 226
 - caer_frame_event_color_filter, 226
 - caerFrameEvent, 226
 - caerFrameEventGetChannelNumber, 227
 - caerFrameEventGetColorFilter, 227
 - caerFrameEventGetExposureLength, 228
 - caerFrameEventGetLengthX, 228
 - caerFrameEventGetLengthY, 228
 - caerFrameEventGetPixel, 230
 - caerFrameEventGetPixelArrayUnsafe, 230
 - caerFrameEventGetPixelArrayUnsafeConst, 231
 - caerFrameEventGetPixelForChannel, 231
 - caerFrameEventGetPixelForChannelUnsafe, 231
 - caerFrameEventGetPixelUnsafe, 233
 - caerFrameEventGetPixelsMaxIndex, 232
 - caerFrameEventGetPixelsSize, 232
 - caerFrameEventGetPositionX, 233
 - caerFrameEventGetPositionY, 233
 - caerFrameEventGetROIIdentifier, 234
 - caerFrameEventGetTSEndOfExposure, 235
 - caerFrameEventGetTSEndOfExposure64, 235
 - caerFrameEventGetTSEndOfFrame, 236
 - caerFrameEventGetTSEndOfFrame64, 236
 - caerFrameEventGetTSSStartOfExposure, 237
 - caerFrameEventGetTSSStartOfExposure64, 237
 - caerFrameEventGetTSSStartOfFrame, 237
 - caerFrameEventGetTSSStartOfFrame64, 238
 - caerFrameEventGetTimestamp, 234
 - caerFrameEventGetTimestamp64, 235
 - caerFrameEventInvalidate, 238
 - caerFrameEventsIsValid, 239
 - caerFrameEventPacket, 226
 - caerFrameEventPacketAllocate, 239
 - caerFrameEventPacketFromPacketHeader, 239
 - caerFrameEventPacketFromPacketHeaderConst, 240
 - caerFrameEventPacketGetEvent, 240
 - caerFrameEventPacketGetEventConst, 241
 - caerFrameEventPacketGetPixelsMaxIndex, 241
 - caerFrameEventPacketGetPixelsSize, 241
 - caerFrameEventSetColorFilter, 243
 - caerFrameEventSetLengthXLengthYChannelNumber, 243
 - caerFrameEventSetPixel, 243
 - caerFrameEventSetPixelForChannel, 244
 - caerFrameEventSetPixelForChannelUnsafe, 244
 - caerFrameEventSetPixelUnsafe, 245
 - caerFrameEventSetPositionX, 245
 - caerFrameEventSetPositionY, 246
 - caerFrameEventSetROIIdentifier, 246
 - caerFrameEventSetTSEndOfExposure, 246
 - caerFrameEventSetTSEndOfFrame, 246
 - caerFrameEventSetTSSStartOfExposure, 247
 - caerFrameEventSetTSSStartOfFrame, 247
 - caerFrameEventValidate, 247
 - FRAME_COLOR_CHANNELS_MASK, 224
 - FRAME_COLOR_CHANNELS_SHIFT, 225
 - FRAME_COLOR_FILTER_MASK, 225
 - FRAME_COLOR_FILTER_SHIFT, 225
 - FRAME_ROI_IDENTIFIER_MASK, 225
 - FRAME_ROI_IDENTIFIER_SHIFT, 225

- PACKED_STRUCT, 248
- frame_utils.h, 431
- GET_NUMBITS16
 - libcaer.h, 433
- GET_NUMBITS32
 - libcaer.h, 434
- GET_NUMBITS8
 - libcaer.h, 434
- I16T
 - libcaer.h, 434
- I32T
 - libcaer.h, 434
- I64T
 - libcaer.h, 434
- I8T
 - libcaer.h, 434
- IS_DAVIS128
 - davis.h, 124
- IS_DAVIS208
 - davis.h, 124
- IS_DAVIS240
 - davis.h, 124
- IS_DAVIS240A
 - davis.h, 124
- IS_DAVIS240B
 - davis.h, 124
- IS_DAVIS240C
 - davis.h, 125
- IS_DAVIS346
 - davis.h, 125
- IS_DAVIS346A
 - davis.h, 125
- IS_DAVIS346B
 - davis.h, 125
- IS_DAVIS346C
 - davis.h, 125
- IS_DAVIS640
 - davis.h, 125
- IS_DAVISRGB
 - davis.h, 126
- imu6.h
 - CAER_IMU6_CONST_ITERATOR_ALL_START, 250
 - CAER_IMU6_CONST_ITERATOR_VALID_START, 250
 - CAER_IMU6_CONST_REVERSE_ITERATOR_ALL_START, 250
 - CAER_IMU6_CONST_REVERSE_ITERATOR_VALID_START, 251
 - CAER_IMU6_ITERATOR_ALL_END, 251
 - CAER_IMU6_ITERATOR_ALL_START, 251
 - CAER_IMU6_ITERATOR_VALID_END, 252
 - CAER_IMU6_ITERATOR_VALID_START, 252
 - CAER_IMU6_REVERSE_ITERATOR_ALL_END, 252
 - CAER_IMU6_REVERSE_ITERATOR_ALL_START, 252
- CAER_IMU6_REVERSE_ITERATOR_VALID_END, 253
- CAER_IMU6_REVERSE_ITERATOR_VALID_START, 253
- caerIMU6Event, 253
- caerIMU6EventGetAccelX, 254
- caerIMU6EventGetAccelY, 254
- caerIMU6EventGetAccelZ, 255
- caerIMU6EventGetGyroX, 255
- caerIMU6EventGetGyroY, 255
- caerIMU6EventGetGyroZ, 256
- caerIMU6EventGetTemp, 256
- caerIMU6EventGetTimestamp, 256
- caerIMU6EventGetTimestamp64, 257
- caerIMU6EventInvalidate, 257
- caerIMU6EventIsValid, 258
- caerIMU6EventPacket, 254
- caerIMU6EventPacketAllocate, 258
- caerIMU6EventPacketFromPacketHeader, 258
- caerIMU6EventPacketFromPacketHeaderConst, 259
- caerIMU6EventPacketGetEvent, 259
- caerIMU6EventPacketGetEventConst, 259
- caerIMU6EventSetAccelX, 260
- caerIMU6EventSetAccelY, 260
- caerIMU6EventSetAccelZ, 260
- caerIMU6EventSetGyroX, 261
- caerIMU6EventSetGyroY, 261
- caerIMU6EventSetGyroZ, 261
- caerIMU6EventSetTemp, 262
- caerIMU6EventSetTimestamp, 262
- caerIMU6EventValidate, 262
- PACKED_STRUCT, 263
- imu9.h
 - CAER_IMU9_CONST_ITERATOR_ALL_START, 265
 - CAER_IMU9_CONST_ITERATOR_VALID_START, 265
 - CAER_IMU9_CONST_REVERSE_ITERATOR_ALL_START, 265
 - CAER_IMU9_CONST_REVERSE_ITERATOR_VALID_START, 266
 - CAER_IMU9_ITERATOR_ALL_END, 266
 - CAER_IMU9_ITERATOR_ALL_START, 266
 - CAER_IMU9_ITERATOR_VALID_END, 267
 - CAER_IMU9_ITERATOR_VALID_START, 267
 - CAER_IMU9_REVERSE_ITERATOR_ALL_END, 267
 - CAER_IMU9_REVERSE_ITERATOR_ALL_START, 267
 - CAER_IMU9_REVERSE_ITERATOR_VALID_END, 268
 - CAER_IMU9_REVERSE_ITERATOR_VALID_START, 268
 - caerIMU9Event, 268
 - caerIMU9EventGetAccelX, 269
 - caerIMU9EventGetAccelY, 269
 - caerIMU9EventGetAccelZ, 270

- caerIMU9EventGetCompX, [270](#)
- caerIMU9EventGetCompY, [270](#)
- caerIMU9EventGetCompZ, [271](#)
- caerIMU9EventGetGyroX, [271](#)
- caerIMU9EventGetGyroY, [271](#)
- caerIMU9EventGetGyroZ, [273](#)
- caerIMU9EventGetTemp, [273](#)
- caerIMU9EventGetTimestamp, [273](#)
- caerIMU9EventGetTimestamp64, [274](#)
- caerIMU9EventInvalidate, [274](#)
- caerIMU9EventIsValid, [275](#)
- caerIMU9EventPacket, [269](#)
- caerIMU9EventPacketAllocate, [275](#)
- caerIMU9EventPacketFromPacketHeader, [275](#)
- caerIMU9EventPacketFromPacketHeaderConst, [276](#)
- caerIMU9EventPacketGetEvent, [276](#)
- caerIMU9EventPacketGetEventConst, [276](#)
- caerIMU9EventSetAccelX, [277](#)
- caerIMU9EventSetAccelY, [277](#)
- caerIMU9EventSetAccelZ, [277](#)
- caerIMU9EventSetCompX, [278](#)
- caerIMU9EventSetCompY, [278](#)
- caerIMU9EventSetCompZ, [278](#)
- caerIMU9EventSetGyroX, [279](#)
- caerIMU9EventSetGyroY, [279](#)
- caerIMU9EventSetGyroZ, [279](#)
- caerIMU9EventSetTemp, [280](#)
- caerIMU9EventSetTimestamp, [280](#)
- caerIMU9EventValidate, [280](#)
- PACKED_STRUCT, [281](#)
- LIBCAER_HAVE_OPENCV
 - libcaer.h, [435](#)
- LIBCAER_HAVE_SERIALDEV
 - libcaer.h, [435](#)
- LIBCAER_NAME_STRING
 - libcaer.h, [435](#)
- LIBCAER_VERSION_STRING
 - libcaer.h, [435](#)
- LIBCAER_VERSION
 - libcaer.h, [435](#)
- libcaer.h, [431](#)
 - CLEAR_NUMBITS16, [433](#)
 - CLEAR_NUMBITS32, [433](#)
 - CLEAR_NUMBITS8, [433](#)
 - caer_error_codes, [437](#)
 - caerByteArrayToInteger, [437](#)
 - caerIntegerToByteArray, [438](#)
 - caerStrEquals, [438](#)
 - caerStrEqualsUpTo, [438](#)
 - GET_NUMBITS16, [433](#)
 - GET_NUMBITS32, [434](#)
 - GET_NUMBITS8, [434](#)
 - I16T, [434](#)
 - I32T, [434](#)
 - I64T, [434](#)
 - I8T, [434](#)
 - LIBCAER_HAVE_OPENCV, [435](#)
 - LIBCAER_HAVE_SERIALDEV, [435](#)
 - LIBCAER_NAME_STRING, [435](#)
 - LIBCAER_VERSION_STRING, [435](#)
 - LIBCAER_VERSION, [435](#)
 - MASK_NUMBITS32, [435](#)
 - MASK_NUMBITS64, [435](#)
 - SET_NUMBITS16, [436](#)
 - SET_NUMBITS32, [436](#)
 - SET_NUMBITS8, [436](#)
 - SWAP_VAR, [436](#)
 - U16T, [436](#)
 - U32T, [436](#)
 - U64T, [437](#)
 - U8T, [437](#)
 - log.h, [439](#)
 - caer_log_level, [440](#)
 - caerLog, [440](#)
 - caerLogDisable, [440](#)
 - caerLogDisabled, [441](#)
 - caerLogFileDescriptorsGetFirst, [441](#)
 - caerLogFileDescriptorsGetSecond, [441](#)
 - caerLogFileDescriptorsSet, [441](#)
 - caerLogLevelGet, [442](#)
 - caerLogLevelSet, [442](#)
 - caerLogVAFull, [443](#)
 - caerLogVA, [442](#)
 - MASK_NUMBITS32
 - libcaer.h, [435](#)
 - MASK_NUMBITS64
 - libcaer.h, [435](#)
 - MATRIX4x4_SCALE_MASK
 - matrix4x4.h, [287](#)
 - MATRIX4x4_SCALE_SHIFT
 - matrix4x4.h, [287](#)
 - MATRIX4x4_TYPE_MASK
 - matrix4x4.h, [287](#)
 - MATRIX4x4_TYPE_SHIFT
 - matrix4x4.h, [287](#)
 - matrix4x4.h
 - CAER_MATRIX4x4_CONST_ITERATOR_ALL_↔START, [283](#)
 - CAER_MATRIX4x4_CONST_ITERATOR_VALID_↔D_START, [283](#)
 - CAER_MATRIX4x4_CONST_REVERSE_ITER_↔ATOR_ALL_START, [284](#)
 - CAER_MATRIX4x4_CONST_REVERSE_ITER_↔ATOR_VALID_START, [284](#)
 - CAER_MATRIX4x4_ITERATOR_ALL_END, [285](#)
 - CAER_MATRIX4x4_ITERATOR_ALL_START, [285](#)
 - CAER_MATRIX4x4_ITERATOR_VALID_END, [285](#)
 - CAER_MATRIX4x4_ITERATOR_VALID_START, [285](#)
 - CAER_MATRIX4x4_REVERSE_ITERATOR_AL_↔L_END, [286](#)
 - CAER_MATRIX4x4_REVERSE_ITERATOR_AL_↔L_START, [286](#)

- CAER_MATRIX4x4_REVERSE_ITERATOR_V↔
ALID_END, [286](#)
- CAER_MATRIX4x4_REVERSE_ITERATOR_V↔
ALID_START, [287](#)
- caerMatrix4x4Event, [288](#)
- caerMatrix4x4EventGetM00, [288](#)
- caerMatrix4x4EventGetM01, [288](#)
- caerMatrix4x4EventGetM02, [289](#)
- caerMatrix4x4EventGetM03, [289](#)
- caerMatrix4x4EventGetM10, [290](#)
- caerMatrix4x4EventGetM11, [290](#)
- caerMatrix4x4EventGetM12, [290](#)
- caerMatrix4x4EventGetM13, [291](#)
- caerMatrix4x4EventGetM20, [291](#)
- caerMatrix4x4EventGetM21, [291](#)
- caerMatrix4x4EventGetM22, [292](#)
- caerMatrix4x4EventGetM23, [292](#)
- caerMatrix4x4EventGetM30, [292](#)
- caerMatrix4x4EventGetM31, [294](#)
- caerMatrix4x4EventGetM32, [294](#)
- caerMatrix4x4EventGetM33, [294](#)
- caerMatrix4x4EventGetScale, [296](#)
- caerMatrix4x4EventGetTimestamp, [296](#)
- caerMatrix4x4EventGetTimestamp64, [296](#)
- caerMatrix4x4EventGetType, [297](#)
- caerMatrix4x4EventInvalidate, [297](#)
- caerMatrix4x4EventIsValid, [298](#)
- caerMatrix4x4EventPacket, [288](#)
- caerMatrix4x4EventPacketAllocate, [298](#)
- caerMatrix4x4EventPacketFromPacketHeader,
[298](#)
- caerMatrix4x4EventPacketFromPacketHeader↔
Const, [299](#)
- caerMatrix4x4EventPacketGetEvent, [299](#)
- caerMatrix4x4EventPacketGetEventConst, [299](#)
- caerMatrix4x4EventSetM00, [300](#)
- caerMatrix4x4EventSetM01, [300](#)
- caerMatrix4x4EventSetM02, [300](#)
- caerMatrix4x4EventSetM03, [301](#)
- caerMatrix4x4EventSetM10, [301](#)
- caerMatrix4x4EventSetM11, [301](#)
- caerMatrix4x4EventSetM12, [302](#)
- caerMatrix4x4EventSetM13, [302](#)
- caerMatrix4x4EventSetM20, [302](#)
- caerMatrix4x4EventSetM21, [303](#)
- caerMatrix4x4EventSetM22, [303](#)
- caerMatrix4x4EventSetM23, [303](#)
- caerMatrix4x4EventSetM30, [304](#)
- caerMatrix4x4EventSetM31, [304](#)
- caerMatrix4x4EventSetM32, [304](#)
- caerMatrix4x4EventSetM33, [304](#)
- caerMatrix4x4EventSetScale, [305](#)
- caerMatrix4x4EventSetTimestamp, [305](#)
- caerMatrix4x4EventSetType, [305](#)
- caerMatrix4x4EventValidate, [306](#)
- MATRIX4x4_SCALE_MASK, [287](#)
- MATRIX4x4_SCALE_SHIFT, [287](#)
- MATRIX4x4_TYPE_MASK, [287](#)
- MATRIX4x4_TYPE_SHIFT, [287](#)
- PACKED_STRUCT, [306](#)
- network.h, [443](#)
- PACKED_STRUCT
 - common.h, [192](#)
 - config.h, [204](#)
 - ear.h, [218](#)
 - frame.h, [248](#)
 - imu6.h, [263](#)
 - imu9.h, [281](#)
 - matrix4x4.h, [306](#)
 - packetContainer.h, [315](#)
 - point1d.h, [328](#)
 - point2d.h, [341](#)
 - point3d.h, [355](#)
 - point4d.h, [371](#)
 - polarity.h, [383](#)
 - sample.h, [395](#)
 - special.h, [410](#)
 - spike.h, [424](#)
- POINT1D_SCALE_MASK
 - point1d.h, [320](#)
- POINT1D_SCALE_SHIFT
 - point1d.h, [320](#)
- POINT1D_TYPE_MASK
 - point1d.h, [321](#)
- POINT1D_TYPE_SHIFT
 - point1d.h, [321](#)
- POINT2D_SCALE_MASK
 - point2d.h, [333](#)
- POINT2D_SCALE_SHIFT
 - point2d.h, [333](#)
- POINT2D_TYPE_MASK
 - point2d.h, [333](#)
- POINT2D_TYPE_SHIFT
 - point2d.h, [333](#)
- POINT3D_SCALE_MASK
 - point3d.h, [346](#)
- POINT3D_SCALE_SHIFT
 - point3d.h, [347](#)
- POINT3D_TYPE_MASK
 - point3d.h, [347](#)
- POINT3D_TYPE_SHIFT
 - point3d.h, [347](#)
- POINT4D_SCALE_MASK
 - point4d.h, [360](#)
- POINT4D_SCALE_SHIFT
 - point4d.h, [361](#)
- POINT4D_TYPE_MASK
 - point4d.h, [361](#)
- POINT4D_TYPE_SHIFT
 - point4d.h, [361](#)
- POLARITY_MASK
 - polarity.h, [376](#)
- POLARITY_SHIFT
 - polarity.h, [376](#)
- POLARITY_X_ADDR_MASK

- polarity.h, [376](#)
- POLARITY_X_ADDR_SHIFT
 - polarity.h, [376](#)
- POLARITY_Y_ADDR_MASK
 - polarity.h, [376](#)
- POLARITY_Y_ADDR_SHIFT
 - polarity.h, [377](#)
- packetContainer.h
 - CAER_EVENT_PACKET_CONTAINER_CONS↔
T_ITERATOR_START, [308](#)
 - CAER_EVENT_PACKET_CONTAINER_ITERA↔
TOR_END, [308](#)
 - CAER_EVENT_PACKET_CONTAINER_ITERA↔
TOR_START, [309](#)
 - caerEventPacketContainer, [309](#)
 - caerEventPacketContainerAllocate, [309](#)
 - caerEventPacketContainerCopyAllEvents, [310](#)
 - caerEventPacketContainerCopyValidEvents, [310](#)
 - caerEventPacketContainerFindEventPacketBy↔
Type, [310](#)
 - caerEventPacketContainerFindEventPacketBy↔
TypeConst, [311](#)
 - caerEventPacketContainerFree, [311](#)
 - caerEventPacketContainerGetEventPacket, [312](#)
 - caerEventPacketContainerGetEventPacketConst, [312](#)
 - caerEventPacketContainerGetEventPackets↔
Number, [312](#)
 - caerEventPacketContainerGetEventsNumber, [313](#)
 - caerEventPacketContainerGetEventsValidNumber, [313](#)
 - caerEventPacketContainerGetHighestEvent↔
Timestamp, [313](#)
 - caerEventPacketContainerGetLowestEvent↔
Timestamp, [314](#)
 - caerEventPacketContainerSetEventPacket, [314](#)
 - caerEventPacketContainerSetEventPackets↔
Number, [315](#)
 - caerEventPacketContainerUpdateStatistics, [315](#)
 - PACKED_STRUCT, [315](#)
- point1d.h
 - CAER_POINT1D_CONST_ITERATOR_ALL_ST↔
ART, [317](#)
 - CAER_POINT1D_CONST_ITERATOR_VALID↔
START, [317](#)
 - CAER_POINT1D_CONST_REVERSE_ITERAT↔
OR_ALL_START, [317](#)
 - CAER_POINT1D_CONST_REVERSE_ITERAT↔
OR_VALID_START, [318](#)
 - CAER_POINT1D_ITERATOR_ALL_END, [318](#)
 - CAER_POINT1D_ITERATOR_ALL_START, [318](#)
 - CAER_POINT1D_ITERATOR_VALID_END, [319](#)
 - CAER_POINT1D_ITERATOR_VALID_START, [319](#)
 - CAER_POINT1D_REVERSE_ITERATOR_ALL↔
END, [319](#)
 - CAER_POINT1D_REVERSE_ITERATOR_ALL↔
START, [319](#)
 - CAER_POINT1D_REVERSE_ITERATOR_VALI↔
D_END, [320](#)
 - CAER_POINT1D_REVERSE_ITERATOR_VALI↔
D_START, [320](#)
 - caerPoint1DEvent, [321](#)
 - caerPoint1DEventGetScale, [321](#)
 - caerPoint1DEventGetTimestamp, [322](#)
 - caerPoint1DEventGetTimestamp64, [322](#)
 - caerPoint1DEventGetType, [322](#)
 - caerPoint1DEventGetX, [323](#)
 - caerPoint1DEventInvalidate, [323](#)
 - caerPoint1DEventIsValid, [324](#)
 - caerPoint1DEventPacket, [321](#)
 - caerPoint1DEventPacketAllocate, [324](#)
 - caerPoint1DEventPacketFromPacketHeader, [324](#)
 - caerPoint1DEventPacketFromPacketHeaderConst, [325](#)
 - caerPoint1DEventPacketGetEvent, [325](#)
 - caerPoint1DEventPacketGetEventConst, [325](#)
 - caerPoint1DEventSetScale, [326](#)
 - caerPoint1DEventSetTimestamp, [326](#)
 - caerPoint1DEventSetType, [327](#)
 - caerPoint1DEventSetX, [327](#)
 - caerPoint1DEventValidate, [327](#)
 - PACKED_STRUCT, [328](#)
 - POINT1D_SCALE_MASK, [320](#)
 - POINT1D_SCALE_SHIFT, [320](#)
 - POINT1D_TYPE_MASK, [321](#)
 - POINT1D_TYPE_SHIFT, [321](#)
- point2d.h
 - CAER_POINT2D_CONST_ITERATOR_ALL_ST↔
ART, [329](#)
 - CAER_POINT2D_CONST_ITERATOR_VALID↔
START, [330](#)
 - CAER_POINT2D_CONST_REVERSE_ITERAT↔
OR_ALL_START, [330](#)
 - CAER_POINT2D_CONST_REVERSE_ITERAT↔
OR_VALID_START, [330](#)
 - CAER_POINT2D_ITERATOR_ALL_END, [331](#)
 - CAER_POINT2D_ITERATOR_ALL_START, [331](#)
 - CAER_POINT2D_ITERATOR_VALID_END, [331](#)
 - CAER_POINT2D_ITERATOR_VALID_START, [331](#)
 - CAER_POINT2D_REVERSE_ITERATOR_ALL↔
END, [332](#)
 - CAER_POINT2D_REVERSE_ITERATOR_ALL↔
START, [332](#)
 - CAER_POINT2D_REVERSE_ITERATOR_VALI↔
D_END, [332](#)
 - CAER_POINT2D_REVERSE_ITERATOR_VALI↔
D_START, [332](#)
 - caerPoint2DEvent, [334](#)
 - caerPoint2DEventGetScale, [334](#)
 - caerPoint2DEventGetTimestamp, [334](#)
 - caerPoint2DEventGetTimestamp64, [335](#)
 - caerPoint2DEventGetType, [335](#)
 - caerPoint2DEventGetX, [335](#)
 - caerPoint2DEventGetY, [336](#)

- caerPoint2DEventInvalidate, [336](#)
- caerPoint2DEventIsValid, [337](#)
- caerPoint2DEventPacket, [334](#)
- caerPoint2DEventPacketAllocate, [337](#)
- caerPoint2DEventPacketFromPacketHeader, [337](#)
- caerPoint2DEventPacketFromPacketHeaderConst, [338](#)
- caerPoint2DEventPacketGetEvent, [338](#)
- caerPoint2DEventPacketGetEventConst, [338](#)
- caerPoint2DEventSetScale, [339](#)
- caerPoint2DEventSetTimestamp, [339](#)
- caerPoint2DEventSetType, [340](#)
- caerPoint2DEventSetX, [340](#)
- caerPoint2DEventSetY, [340](#)
- caerPoint2DEventSetZ, [340](#)
- caerPoint2DEventValidate, [340](#)
- PACKED_STRUCT, [341](#)
- POINT2D_SCALE_MASK, [333](#)
- POINT2D_SCALE_SHIFT, [333](#)
- POINT2D_TYPE_MASK, [333](#)
- POINT2D_TYPE_SHIFT, [333](#)
- point3d.h
 - CAER_POINT3D_CONST_ITERATOR_ALL_START, [343](#)
 - CAER_POINT3D_CONST_ITERATOR_VALID_START, [343](#)
 - CAER_POINT3D_CONST_REVERSE_ITERATOR_OR_ALL_START, [344](#)
 - CAER_POINT3D_CONST_REVERSE_ITERATOR_OR_VALID_START, [344](#)
 - CAER_POINT3D_ITERATOR_ALL_END, [344](#)
 - CAER_POINT3D_ITERATOR_ALL_START, [345](#)
 - CAER_POINT3D_ITERATOR_VALID_END, [345](#)
 - CAER_POINT3D_ITERATOR_VALID_START, [345](#)
 - CAER_POINT3D_REVERSE_ITERATOR_ALL_END, [345](#)
 - CAER_POINT3D_REVERSE_ITERATOR_ALL_START, [346](#)
 - CAER_POINT3D_REVERSE_ITERATOR_VALID_END, [346](#)
 - CAER_POINT3D_REVERSE_ITERATOR_VALID_START, [346](#)
- caerPoint3DEvent, [347](#)
- caerPoint3DEventGetScale, [348](#)
- caerPoint3DEventGetTimestamp, [348](#)
- caerPoint3DEventGetTimestamp64, [348](#)
- caerPoint3DEventGetType, [349](#)
- caerPoint3DEventGetX, [349](#)
- caerPoint3DEventGetY, [349](#)
- caerPoint3DEventGetZ, [350](#)
- caerPoint3DEventInvalidate, [350](#)
- caerPoint3DEventIsValid, [351](#)
- caerPoint3DEventPacket, [347](#)
- caerPoint3DEventPacketAllocate, [351](#)
- caerPoint3DEventPacketFromPacketHeader, [351](#)
- caerPoint3DEventPacketFromPacketHeaderConst, [352](#)
- caerPoint3DEventPacketGetEvent, [352](#)
- caerPoint3DEventPacketGetEventConst, [352](#)
- caerPoint3DEventSetScale, [353](#)
- caerPoint3DEventSetTimestamp, [353](#)
- caerPoint3DEventSetType, [354](#)
- caerPoint3DEventSetX, [354](#)
- caerPoint3DEventSetY, [354](#)
- caerPoint3DEventSetZ, [354](#)
- caerPoint3DEventValidate, [355](#)
- PACKED_STRUCT, [355](#)
- POINT3D_SCALE_MASK, [346](#)
- POINT3D_SCALE_SHIFT, [347](#)
- POINT3D_TYPE_MASK, [347](#)
- POINT3D_TYPE_SHIFT, [347](#)
- point4d.h
 - CAER_POINT4D_CONST_ITERATOR_ALL_START, [357](#)
 - CAER_POINT4D_CONST_ITERATOR_VALID_START, [357](#)
 - CAER_POINT4D_CONST_REVERSE_ITERATOR_OR_ALL_START, [358](#)
 - CAER_POINT4D_CONST_REVERSE_ITERATOR_OR_VALID_START, [358](#)
 - CAER_POINT4D_ITERATOR_ALL_END, [358](#)
 - CAER_POINT4D_ITERATOR_ALL_START, [359](#)
 - CAER_POINT4D_ITERATOR_VALID_END, [359](#)
 - CAER_POINT4D_ITERATOR_VALID_START, [359](#)
 - CAER_POINT4D_REVERSE_ITERATOR_ALL_END, [359](#)
 - CAER_POINT4D_REVERSE_ITERATOR_ALL_START, [360](#)
 - CAER_POINT4D_REVERSE_ITERATOR_VALID_END, [360](#)
 - CAER_POINT4D_REVERSE_ITERATOR_VALID_START, [360](#)
- caerPoint4DEvent, [361](#)
- caerPoint4DEventGetScale, [362](#)
- caerPoint4DEventGetTimestamp, [362](#)
- caerPoint4DEventGetTimestamp64, [362](#)
- caerPoint4DEventGetType, [363](#)
- caerPoint4DEventGetW, [363](#)
- caerPoint4DEventGetX, [363](#)
- caerPoint4DEventGetY, [364](#)
- caerPoint4DEventGetZ, [364](#)
- caerPoint4DEventInvalidate, [365](#)
- caerPoint4DEventIsValid, [365](#)
- caerPoint4DEventPacket, [361](#)
- caerPoint4DEventPacketAllocate, [365](#)
- caerPoint4DEventPacketFromPacketHeader, [366](#)
- caerPoint4DEventPacketFromPacketHeaderConst, [366](#)
- caerPoint4DEventPacketGetEvent, [366](#)
- caerPoint4DEventPacketGetEventConst, [367](#)
- caerPoint4DEventSetScale, [367](#)
- caerPoint4DEventSetTimestamp, [367](#)
- caerPoint4DEventSetType, [368](#)
- caerPoint4DEventSetW, [368](#)
- caerPoint4DEventSetX, [368](#)

- caerPoint4DEventSetY, [369](#)
- caerPoint4DEventSetZ, [369](#)
- caerPoint4DEventValidate, [369](#)
- PACKED_STRUCT, [371](#)
- POINT4D_SCALE_MASK, [360](#)
- POINT4D_SCALE_SHIFT, [361](#)
- POINT4D_TYPE_MASK, [361](#)
- POINT4D_TYPE_SHIFT, [361](#)
- polarity.h
 - CAER_POLARITY_CONST_ITERATOR_ALL_START, [372](#)
 - CAER_POLARITY_CONST_ITERATOR_VALID_START, [373](#)
 - CAER_POLARITY_CONST_REVERSE_ITERATOR_ALL_START, [373](#)
 - CAER_POLARITY_CONST_REVERSE_ITERATOR_VALID_START, [373](#)
 - CAER_POLARITY_ITERATOR_ALL_END, [374](#)
 - CAER_POLARITY_ITERATOR_ALL_START, [374](#)
 - CAER_POLARITY_ITERATOR_VALID_END, [374](#)
 - CAER_POLARITY_ITERATOR_VALID_START, [374](#)
 - CAER_POLARITY_REVERSE_ITERATOR_ALL_END, [375](#)
 - CAER_POLARITY_REVERSE_ITERATOR_ALL_START, [375](#)
 - CAER_POLARITY_REVERSE_ITERATOR_VALID_END, [375](#)
 - CAER_POLARITY_REVERSE_ITERATOR_VALID_START, [375](#)
 - caerPolarityEvent, [377](#)
 - caerPolarityEventGetPolarity, [377](#)
 - caerPolarityEventGetTimestamp, [378](#)
 - caerPolarityEventGetTimestamp64, [378](#)
 - caerPolarityEventGetX, [378](#)
 - caerPolarityEventGetY, [379](#)
 - caerPolarityEventInvalidate, [379](#)
 - caerPolarityEventIsValid, [379](#)
 - caerPolarityEventPacket, [377](#)
 - caerPolarityEventPacketAllocate, [380](#)
 - caerPolarityEventPacketFromPacketHeader, [380](#)
 - caerPolarityEventPacketFromPacketHeaderConst, [381](#)
 - caerPolarityEventPacketGetEvent, [381](#)
 - caerPolarityEventPacketGetEventConst, [381](#)
 - caerPolarityEventSetPolarity, [382](#)
 - caerPolarityEventSetTimestamp, [382](#)
 - caerPolarityEventSetX, [382](#)
 - caerPolarityEventSetY, [383](#)
 - caerPolarityEventValidate, [383](#)
 - PACKED_STRUCT, [383](#)
 - POLARITY_MASK, [376](#)
 - POLARITY_SHIFT, [376](#)
 - POLARITY_X_ADDR_MASK, [376](#)
 - POLARITY_X_ADDR_SHIFT, [376](#)
 - POLARITY_Y_ADDR_MASK, [376](#)
 - POLARITY_Y_ADDR_SHIFT, [377](#)
- portable_endian.h, [444](#)
- SAMPLE_MASK
 - sample.h, [388](#)
- SAMPLE_SHIFT
 - sample.h, [389](#)
- SAMPLE_TYPE_MASK
 - sample.h, [389](#)
- SAMPLE_TYPE_SHIFT
 - sample.h, [389](#)
- SET_NUMBITS16
 - libcaer.h, [436](#)
- SET_NUMBITS32
 - libcaer.h, [436](#)
- SET_NUMBITS8
 - libcaer.h, [436](#)
- SPECIAL_DATA_MASK
 - special.h, [401](#)
- SPECIAL_DATA_SHIFT
 - special.h, [401](#)
- SPECIAL_TYPE_MASK
 - special.h, [401](#)
- SPECIAL_TYPE_SHIFT
 - special.h, [401](#)
- SPIKE_CHIP_ID_MASK
 - spike.h, [415](#)
- SPIKE_CHIP_ID_SHIFT
 - spike.h, [416](#)
- SPIKE_NEURON_ID_MASK
 - spike.h, [416](#)
- SPIKE_NEURON_ID_SHIFT
 - spike.h, [416](#)
- SPIKE_SOURCE_CORE_ID_MASK
 - spike.h, [416](#)
- SPIKE_SOURCE_CORE_ID_SHIFT
 - spike.h, [416](#)
- SWAP_VAR
 - libcaer.h, [436](#)
- sample.h
 - CAER_SAMPLE_CONST_ITERATOR_ALL_START, [385](#)
 - CAER_SAMPLE_CONST_ITERATOR_VALID_START, [385](#)
 - CAER_SAMPLE_CONST_REVERSE_ITERATOR_ALL_START, [386](#)
 - CAER_SAMPLE_CONST_REVERSE_ITERATOR_VALID_START, [386](#)
 - CAER_SAMPLE_ITERATOR_ALL_END, [386](#)
 - CAER_SAMPLE_ITERATOR_ALL_START, [387](#)
 - CAER_SAMPLE_ITERATOR_VALID_END, [387](#)
 - CAER_SAMPLE_ITERATOR_VALID_START, [387](#)
 - CAER_SAMPLE_REVERSE_ITERATOR_ALL_END, [387](#)
 - CAER_SAMPLE_REVERSE_ITERATOR_ALL_START, [388](#)
 - CAER_SAMPLE_REVERSE_ITERATOR_VALID_END, [388](#)
 - CAER_SAMPLE_REVERSE_ITERATOR_VALID_START, [388](#)
 - caerSampleEvent, [389](#)

- caerSampleEventGetSample, 390
- caerSampleEventGetTimestamp, 390
- caerSampleEventGetTimestamp64, 390
- caerSampleEventGetType, 391
- caerSampleEventInvalidate, 391
- caerSampleEventIsValid, 391
- caerSampleEventPacket, 389
- caerSampleEventPacketAllocate, 392
- caerSampleEventPacketFromPacketHeader, 392
- caerSampleEventPacketFromPacketHeaderConst, 393
- caerSampleEventPacketGetEvent, 393
- caerSampleEventPacketGetEventConst, 393
- caerSampleEventSetSample, 394
- caerSampleEventSetTimestamp, 394
- caerSampleEventSetType, 394
- caerSampleEventValidate, 395
- PACKED_STRUCT, 395
- SAMPLE_MASK, 388
- SAMPLE_SHIFT, 389
- SAMPLE_TYPE_MASK, 389
- SAMPLE_TYPE_SHIFT, 389
- serial.h
 - CAER_HOST_CONFIG_SERIAL_BAUD_RATE↔_12M, 171
 - CAER_HOST_CONFIG_SERIAL_BAUD_RATE↔_2M, 171
 - CAER_HOST_CONFIG_SERIAL_BAUD_RATE↔_4M, 172
 - CAER_HOST_CONFIG_SERIAL_BAUD_RATE↔_8M, 172
 - CAER_HOST_CONFIG_SERIAL_READ_SIZE, 172
 - CAER_HOST_CONFIG_SERIAL, 171
 - caerDeviceOpenSerial, 172
- special.h
 - CAER_SPECIAL_CONST_ITERATOR_ALL_START↔ART, 397
 - CAER_SPECIAL_CONST_ITERATOR_VALID↔START, 398
 - CAER_SPECIAL_CONST_REVERSE_ITERATOR↔OR_ALL_START, 398
 - CAER_SPECIAL_CONST_REVERSE_ITERATOR↔OR_VALID_START, 398
 - CAER_SPECIAL_ITERATOR_ALL_END, 399
 - CAER_SPECIAL_ITERATOR_ALL_START, 399
 - CAER_SPECIAL_ITERATOR_VALID_END, 399
 - CAER_SPECIAL_ITERATOR_VALID_START, 399
 - CAER_SPECIAL_REVERSE_ITERATOR_ALL↔END, 400
 - CAER_SPECIAL_REVERSE_ITERATOR_ALL↔START, 400
 - CAER_SPECIAL_REVERSE_ITERATOR_VALID↔END, 400
 - CAER_SPECIAL_REVERSE_ITERATOR_VALID↔START, 400
 - caer_special_event_types, 402
 - caerSpecialEvent, 402
 - caerSpecialEventGetData, 403
 - caerSpecialEventGetTimestamp, 403
 - caerSpecialEventGetTimestamp64, 404
 - caerSpecialEventGetType, 404
 - caerSpecialEventInvalidate, 404
 - caerSpecialEventIsValid, 405
 - caerSpecialEventPacket, 402
 - caerSpecialEventPacketAllocate, 405
 - caerSpecialEventPacketFindEventByType, 406
 - caerSpecialEventPacketFindEventByTypeConst, 406
 - caerSpecialEventPacketFindValidEventByType, 406
 - caerSpecialEventPacketFindValidEventByType↔Const, 407
 - caerSpecialEventPacketFromPacketHeader, 407
 - caerSpecialEventPacketFromPacketHeaderConst, 408
 - caerSpecialEventPacketGetEvent, 408
 - caerSpecialEventPacketGetEventConst, 408
 - caerSpecialEventSetData, 409
 - caerSpecialEventSetTimestamp, 409
 - caerSpecialEventSetType, 409
 - caerSpecialEventValidate, 410
 - PACKED_STRUCT, 410
 - SPECIAL_DATA_MASK, 401
 - SPECIAL_DATA_SHIFT, 401
 - SPECIAL_TYPE_MASK, 401
 - SPECIAL_TYPE_SHIFT, 401
 - spike.h
 - CAER_SPIKE_CONST_ITERATOR_ALL_START, 412
 - CAER_SPIKE_CONST_ITERATOR_VALID_START↔ART, 412
 - CAER_SPIKE_CONST_REVERSE_ITERATOR↔ALL_START, 413
 - CAER_SPIKE_CONST_REVERSE_ITERATOR↔VALID_START, 413
 - CAER_SPIKE_ITERATOR_ALL_END, 413
 - CAER_SPIKE_ITERATOR_ALL_START, 414
 - CAER_SPIKE_ITERATOR_VALID_END, 414
 - CAER_SPIKE_ITERATOR_VALID_START, 414
 - CAER_SPIKE_REVERSE_ITERATOR_ALL_END, 414
 - CAER_SPIKE_REVERSE_ITERATOR_ALL_START↔ART, 415
 - CAER_SPIKE_REVERSE_ITERATOR_VALID↔END, 415
 - CAER_SPIKE_REVERSE_ITERATOR_VALID↔START, 415
 - caerSpikeEvent, 416
 - caerSpikeEventGetChipID, 417
 - caerSpikeEventGetNeuronID, 417
 - caerSpikeEventGetSourceCoreID, 418
 - caerSpikeEventGetTimestamp, 418
 - caerSpikeEventGetTimestamp64, 418
 - caerSpikeEventInvalidate, 419

- caerSpikeEventIsValid, [419](#)
- caerSpikeEventPacket, [417](#)
- caerSpikeEventPacketAllocate, [419](#)
- caerSpikeEventPacketFromPacketHeader, [420](#)
- caerSpikeEventPacketFromPacketHeaderConst, [420](#)
- caerSpikeEventPacketGetEvent, [421](#)
- caerSpikeEventPacketGetEventConst, [421](#)
- caerSpikeEventSetChipID, [421](#)
- caerSpikeEventSetNeuronID, [423](#)
- caerSpikeEventSetSourceCoreID, [423](#)
- caerSpikeEventSetTimestamp, [423](#)
- caerSpikeEventValidate, [424](#)
- PACKED_STRUCT, [424](#)
- SPIKE_CHIP_ID_MASK, [415](#)
- SPIKE_CHIP_ID_SHIFT, [416](#)
- SPIKE_NEURON_ID_MASK, [416](#)
- SPIKE_NEURON_ID_SHIFT, [416](#)
- SPIKE_SOURCE_CORE_ID_MASK, [416](#)
- SPIKE_SOURCE_CORE_ID_SHIFT, [416](#)

TS_OVERFLOW_SHIFT
common.h, [177](#)

U16T
libcaer.h, [436](#)

U32T
libcaer.h, [436](#)

U64T
libcaer.h, [437](#)

U8T
libcaer.h, [437](#)

usb.h

- CAER_HOST_CONFIG_USB_BUFFER_NUMBERS, [173](#)
- CAER_HOST_CONFIG_USB_BUFFER_SIZE, [173](#)
- CAER_HOST_CONFIG_USB, [173](#)
- caerDeviceOpen, [174](#)

VALID_MARK_MASK
common.h, [178](#)

VALID_MARK_SHIFT
common.h, [178](#)