

libcaer
1.0.3-r8280

Generated by Doxygen 1.8.10

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_shiftedsources Struct Reference	5
3.2.1	Detailed Description	6
3.3	caer_bias_vdac Struct Reference	6
3.3.1	Detailed Description	6
3.4	caer_configuration_event Struct Reference	6
3.4.1	Detailed Description	7
3.5	caer_configuration_event_packet Struct Reference	7
3.5.1	Detailed Description	7
3.6	caer_davis_info Struct Reference	7
3.6.1	Detailed Description	8
3.7	caer_dvs128_info Struct Reference	8
3.7.1	Detailed Description	9
3.8	caer_ear_event Struct Reference	9
3.8.1	Detailed Description	9
3.9	caer_ear_event_packet Struct Reference	9
3.9.1	Detailed Description	10
3.10	caer_event_packet_container Struct Reference	10
3.10.1	Detailed Description	10
3.11	caer_event_packet_header Struct Reference	10
3.11.1	Detailed Description	11
3.12	caer_frame_event Struct Reference	11
3.12.1	Detailed Description	11
3.12.2	Field Documentation	11

3.12.2.1	pixels	11
3.13	caer_frame_event_packet Struct Reference	12
3.13.1	Detailed Description	12
3.14	caer_imu6_event Struct Reference	12
3.14.1	Detailed Description	13
3.15	caer_imu6_event_packet Struct Reference	13
3.15.1	Detailed Description	13
3.16	caer_imu9_event Struct Reference	13
3.16.1	Detailed Description	14
3.17	caer_imu9_event_packet Struct Reference	14
3.17.1	Detailed Description	14
3.18	caer_polarity_event Struct Reference	14
3.18.1	Detailed Description	15
3.19	caer_polarity_event_packet Struct Reference	15
3.19.1	Detailed Description	15
3.20	caer_sample_event Struct Reference	15
3.20.1	Detailed Description	16
3.21	caer_sample_event_packet Struct Reference	16
3.21.1	Detailed Description	16
3.22	caer_special_event Struct Reference	16
3.22.1	Detailed Description	16
3.23	caer_special_event_packet Struct Reference	17
3.23.1	Detailed Description	17
4	File Documentation	19
4.1	devices/davis.h File Reference	19
4.1.1	Detailed Description	27
4.1.2	Macro Definition Documentation	28
4.1.2.1	CAER_DEVICE_DAVIS_FX2	28
4.1.2.2	CAER_DEVICE_DAVIS_FX3	28
4.1.2.3	DAVIS128_CONFIG_BIAS_ADCCOMPBP	28
4.1.2.4	DAVIS128_CONFIG_BIAS_ADCREFHIGH	28
4.1.2.5	DAVIS128_CONFIG_BIAS_ADCREFLOW	28
4.1.2.6	DAVIS128_CONFIG_BIAS_AEPDBN	28
4.1.2.7	DAVIS128_CONFIG_BIAS_AEPUXBP	29
4.1.2.8	DAVIS128_CONFIG_BIAS_AEPUYBP	29
4.1.2.9	DAVIS128_CONFIG_BIAS_APSCAS	29
4.1.2.10	DAVIS128_CONFIG_BIAS_APSOEVERFLOWLEVEL	29
4.1.2.11	DAVIS128_CONFIG_BIAS_APSROSFBN	29
4.1.2.12	DAVIS128_CONFIG_BIAS_BIASBUFFER	30

4.1.2.13	DAVIS128_CONFIG_BIAS_COLSELLOWBN	30
4.1.2.14	DAVIS128_CONFIG_BIAS_DACBUFBP	30
4.1.2.15	DAVIS128_CONFIG_BIAS_DIFFBN	30
4.1.2.16	DAVIS128_CONFIG_BIAS_IFREFRBN	30
4.1.2.17	DAVIS128_CONFIG_BIAS_IFTHRBN	31
4.1.2.18	DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN	31
4.1.2.19	DAVIS128_CONFIG_BIAS_LOCALBUFBP	31
4.1.2.20	DAVIS128_CONFIG_BIAS_OFFBN	31
4.1.2.21	DAVIS128_CONFIG_BIAS_ONBN	31
4.1.2.22	DAVIS128_CONFIG_BIAS_PADFOLLBN	32
4.1.2.23	DAVIS128_CONFIG_BIAS_PIXINVB	32
4.1.2.24	DAVIS128_CONFIG_BIAS_PRBP	32
4.1.2.25	DAVIS128_CONFIG_BIAS_PRSFBP	32
4.1.2.26	DAVIS128_CONFIG_BIAS_READOUTBUFBP	32
4.1.2.27	DAVIS128_CONFIG_BIAS_REFRBP	33
4.1.2.28	DAVIS128_CONFIG_BIAS_SSN	33
4.1.2.29	DAVIS128_CONFIG_BIAS_SSP	33
4.1.2.30	DAVIS128_CONFIG_CHIP_AERNAROW	33
4.1.2.31	DAVIS128_CONFIG_CHIP_ANALOGMUX0	33
4.1.2.32	DAVIS128_CONFIG_CHIP_ANALOGMUX1	33
4.1.2.33	DAVIS128_CONFIG_CHIP_ANALOGMUX2	34
4.1.2.34	DAVIS128_CONFIG_CHIP_BIASMUX0	34
4.1.2.35	DAVIS128_CONFIG_CHIP_DIGITALMUX0	34
4.1.2.36	DAVIS128_CONFIG_CHIP_DIGITALMUX1	34
4.1.2.37	DAVIS128_CONFIG_CHIP_DIGITALMUX2	34
4.1.2.38	DAVIS128_CONFIG_CHIP_DIGITALMUX3	34
4.1.2.39	DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER	34
4.1.2.40	DAVIS128_CONFIG_CHIP_RESETCALIBNEURON	34
4.1.2.41	DAVIS128_CONFIG_CHIP_RESETTESTPIXEL	34
4.1.2.42	DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER	35
4.1.2.43	DAVIS128_CONFIG_CHIP_TYPENCALIBNEURON	35
4.1.2.44	DAVIS128_CONFIG_CHIP_USEAOUT	35
4.1.2.45	DAVIS208_CONFIG_BIAS_ADCCOMPBP	35
4.1.2.46	DAVIS208_CONFIG_BIAS_ADCREFHIGH	35
4.1.2.47	DAVIS208_CONFIG_BIAS_ADCREFLOW	35
4.1.2.48	DAVIS208_CONFIG_BIAS_AEPDBN	36
4.1.2.49	DAVIS208_CONFIG_BIAS_AEPUXBP	36
4.1.2.50	DAVIS208_CONFIG_BIAS_AEPUYBP	36
4.1.2.51	DAVIS208_CONFIG_BIAS_APSCAS	36
4.1.2.52	DAVIS208_CONFIG_BIAS_APSOEVERFLOWLEVEL	36

4.1.2.53	DAVIS208_CONFIG_BIAS_APSROSFBN	37
4.1.2.54	DAVIS208_CONFIG_BIAS_BIASBUFFER	37
4.1.2.55	DAVIS208_CONFIG_BIAS_COLSELLOWBN	37
4.1.2.56	DAVIS208_CONFIG_BIAS_DACBUFBP	37
4.1.2.57	DAVIS208_CONFIG_BIAS_DIFFBN	37
4.1.2.58	DAVIS208_CONFIG_BIAS_IFREFRBN	38
4.1.2.59	DAVIS208_CONFIG_BIAS_IFTHRBN	38
4.1.2.60	DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN	38
4.1.2.61	DAVIS208_CONFIG_BIAS_LOCALBUFBN	38
4.1.2.62	DAVIS208_CONFIG_BIAS_OFFBN	38
4.1.2.63	DAVIS208_CONFIG_BIAS_ONBN	39
4.1.2.64	DAVIS208_CONFIG_BIAS_PADFOLLBN	39
4.1.2.65	DAVIS208_CONFIG_BIAS_PIXINVBN	39
4.1.2.66	DAVIS208_CONFIG_BIAS_PRBP	39
4.1.2.67	DAVIS208_CONFIG_BIAS_PRSFBP	39
4.1.2.68	DAVIS208_CONFIG_BIAS_READOUTBUFBP	40
4.1.2.69	DAVIS208_CONFIG_BIAS_REFRBP	40
4.1.2.70	DAVIS208_CONFIG_BIAS_REFSS	40
4.1.2.71	DAVIS208_CONFIG_BIAS_REFSSBN	40
4.1.2.72	DAVIS208_CONFIG_BIAS_REGBIASBP	40
4.1.2.73	DAVIS208_CONFIG_BIAS_RESETHIGHPASS	41
4.1.2.74	DAVIS208_CONFIG_BIAS_SSN	41
4.1.2.75	DAVIS208_CONFIG_BIAS_SSP	41
4.1.2.76	DAVIS208_CONFIG_CHIP_AERNAROW	41
4.1.2.77	DAVIS208_CONFIG_CHIP_ANALOGMUX0	41
4.1.2.78	DAVIS208_CONFIG_CHIP_ANALOGMUX1	41
4.1.2.79	DAVIS208_CONFIG_CHIP_ANALOGMUX2	42
4.1.2.80	DAVIS208_CONFIG_CHIP_BIASMUX0	42
4.1.2.81	DAVIS208_CONFIG_CHIP_DIGITALMUX0	42
4.1.2.82	DAVIS208_CONFIG_CHIP_DIGITALMUX1	42
4.1.2.83	DAVIS208_CONFIG_CHIP_DIGITALMUX2	42
4.1.2.84	DAVIS208_CONFIG_CHIP_DIGITALMUX3	42
4.1.2.85	DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER	42
4.1.2.86	DAVIS208_CONFIG_CHIP_RESETCALIBNEURON	42
4.1.2.87	DAVIS208_CONFIG_CHIP_RESETTESTPIXEL	42
4.1.2.88	DAVIS208_CONFIG_CHIP_SELECTBIASREFSS	43
4.1.2.89	DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER	43
4.1.2.90	DAVIS208_CONFIG_CHIP_SELECTHIGHPASS	43
4.1.2.91	DAVIS208_CONFIG_CHIP_SELECTPOSBP	43
4.1.2.92	DAVIS208_CONFIG_CHIP_SELECTPREMPAVG	43

4.1.2.93	DAVIS208_CONFIG_CHIP_SELECTSENSE	43
4.1.2.94	DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON	43
4.1.2.95	DAVIS208_CONFIG_CHIP_USEAOUT	43
4.1.2.96	DAVIS240_CONFIG_BIAS_AEPDBN	44
4.1.2.97	DAVIS240_CONFIG_BIAS_AEPUXBP	44
4.1.2.98	DAVIS240_CONFIG_BIAS_AEPUYBP	44
4.1.2.99	DAVIS240_CONFIG_BIAS_APSCASEPC	44
4.1.2.100	DAVIS240_CONFIG_BIAS_APSOEVERFLOWLEVELBN	44
4.1.2.101	DAVIS240_CONFIG_BIAS_APSROSFBN	45
4.1.2.102	DAVIS240_CONFIG_BIAS_BIASBUFFER	45
4.1.2.103	DAVIS240_CONFIG_BIAS_DIFFBN	45
4.1.2.104	DAVIS240_CONFIG_BIAS_DIFFCASBNC	45
4.1.2.105	DAVIS240_CONFIG_BIAS_IFREFRBN	45
4.1.2.106	DAVIS240_CONFIG_BIAS_IFTHRBN	46
4.1.2.107	DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN	46
4.1.2.108	DAVIS240_CONFIG_BIAS_LOCALBUFBN	46
4.1.2.109	DAVIS240_CONFIG_BIAS_OFFBN	46
4.1.2.110	DAVIS240_CONFIG_BIAS_ONBN	46
4.1.2.111	DAVIS240_CONFIG_BIAS_PADFOLLBN	47
4.1.2.112	DAVIS240_CONFIG_BIAS_PIXINVBN	47
4.1.2.113	DAVIS240_CONFIG_BIAS_PRBP	47
4.1.2.114	DAVIS240_CONFIG_BIAS_PRSFBP	47
4.1.2.115	DAVIS240_CONFIG_BIAS_REFRBP	47
4.1.2.116	DAVIS240_CONFIG_BIAS_SSN	48
4.1.2.117	DAVIS240_CONFIG_BIAS_SSP	48
4.1.2.118	DAVIS240_CONFIG_CHIP_AERNAROW	48
4.1.2.119	DAVIS240_CONFIG_CHIP_ANALOGMUX0	48
4.1.2.120	DAVIS240_CONFIG_CHIP_ANALOGMUX1	48
4.1.2.121	DAVIS240_CONFIG_CHIP_ANALOGMUX2	48
4.1.2.122	DAVIS240_CONFIG_CHIP_BIASMUX0	48
4.1.2.123	DAVIS240_CONFIG_CHIP_DIGITALMUX0	49
4.1.2.124	DAVIS240_CONFIG_CHIP_DIGITALMUX1	49
4.1.2.125	DAVIS240_CONFIG_CHIP_DIGITALMUX2	49
4.1.2.126	DAVIS240_CONFIG_CHIP_DIGITALMUX3	49
4.1.2.127	DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER	49
4.1.2.128	DAVIS240_CONFIG_CHIP_RESETCALIBNEURON	49
4.1.2.129	DAVIS240_CONFIG_CHIP_RESETTESTPIXEL	49
4.1.2.130	DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL	49
4.1.2.131	DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON	50
4.1.2.132	DAVIS240_CONFIG_CHIP_USEAOUT	50

4.1.2.133 DAVIS346_CONFIG_BIAS_ADCCOMPBP	50
4.1.2.134 DAVIS346_CONFIG_BIAS_ADCREFHIGH	50
4.1.2.135 DAVIS346_CONFIG_BIAS_ADCREFLOW	50
4.1.2.136 DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE	51
4.1.2.137 DAVIS346_CONFIG_BIAS_AEPDBN	51
4.1.2.138 DAVIS346_CONFIG_BIAS_AEPUXBP	51
4.1.2.139 DAVIS346_CONFIG_BIAS_AEPUYBP	51
4.1.2.140 DAVIS346_CONFIG_BIAS_APSCAS	51
4.1.2.141 DAVIS346_CONFIG_BIAS_APSOEVERFLOWLEVEL	52
4.1.2.142 DAVIS346_CONFIG_BIAS_APSROSFBN	52
4.1.2.143 DAVIS346_CONFIG_BIAS_BIASBUFFER	52
4.1.2.144 DAVIS346_CONFIG_BIAS_COLSELLOWBN	52
4.1.2.145 DAVIS346_CONFIG_BIAS_DACBUFBP	52
4.1.2.146 DAVIS346_CONFIG_BIAS_DIFFBN	53
4.1.2.147 DAVIS346_CONFIG_BIAS_IFREFRBN	53
4.1.2.148 DAVIS346_CONFIG_BIAS_IFTHRBN	53
4.1.2.149 DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN	53
4.1.2.150 DAVIS346_CONFIG_BIAS_LOCALBUFBN	53
4.1.2.151 DAVIS346_CONFIG_BIAS_OFFBN	54
4.1.2.152 DAVIS346_CONFIG_BIAS_ONBN	54
4.1.2.153 DAVIS346_CONFIG_BIAS_PADFOLLBN	54
4.1.2.154 DAVIS346_CONFIG_BIAS_PIXINVBN	54
4.1.2.155 DAVIS346_CONFIG_BIAS_PRBP	54
4.1.2.156 DAVIS346_CONFIG_BIAS_PRSFBP	55
4.1.2.157 DAVIS346_CONFIG_BIAS_READOUTBUFBP	55
4.1.2.158 DAVIS346_CONFIG_BIAS_REFRBP	55
4.1.2.159 DAVIS346_CONFIG_BIAS_SSN	55
4.1.2.160 DAVIS346_CONFIG_BIAS_SSP	55
4.1.2.161 DAVIS346_CONFIG_CHIP_AERNAROW	56
4.1.2.162 DAVIS346_CONFIG_CHIP_ANALOGMUX0	56
4.1.2.163 DAVIS346_CONFIG_CHIP_ANALOGMUX1	56
4.1.2.164 DAVIS346_CONFIG_CHIP_ANALOGMUX2	56
4.1.2.165 DAVIS346_CONFIG_CHIP_BIASMUX0	56
4.1.2.166 DAVIS346_CONFIG_CHIP_DIGITALMUX0	56
4.1.2.167 DAVIS346_CONFIG_CHIP_DIGITALMUX1	56
4.1.2.168 DAVIS346_CONFIG_CHIP_DIGITALMUX2	56
4.1.2.169 DAVIS346_CONFIG_CHIP_DIGITALMUX3	56
4.1.2.170 DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER	57
4.1.2.171 DAVIS346_CONFIG_CHIP_RESETCALIBNEURON	57
4.1.2.172 DAVIS346_CONFIG_CHIP_RESETTESTPIXEL	57

4.1.2.173 DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER	57
4.1.2.174 DAVIS346_CONFIG_CHIP_TESTADC	57
4.1.2.175 DAVIS346_CONFIG_CHIP_TYPCALIBNEURON	57
4.1.2.176 DAVIS346_CONFIG_CHIP_USEAOUT	57
4.1.2.177 DAVIS640_CONFIG_BIAS_ADCCOMPBP	57
4.1.2.178 DAVIS640_CONFIG_BIAS_ADCCREFHIGH	58
4.1.2.179 DAVIS640_CONFIG_BIAS_ADCCREFLOW	58
4.1.2.180 DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE	58
4.1.2.181 DAVIS640_CONFIG_BIAS_AEPDBN	58
4.1.2.182 DAVIS640_CONFIG_BIAS_AEPUXBP	59
4.1.2.183 DAVIS640_CONFIG_BIAS_AEPUYBP	59
4.1.2.184 DAVIS640_CONFIG_BIAS_APSCAS	59
4.1.2.185 DAVIS640_CONFIG_BIAS_APSEVERFLOWLEVEL	59
4.1.2.186 DAVIS640_CONFIG_BIAS_APSROSFBN	59
4.1.2.187 DAVIS640_CONFIG_BIAS_BIASBUFFER	60
4.1.2.188 DAVIS640_CONFIG_BIAS_COLSELOWBN	60
4.1.2.189 DAVIS640_CONFIG_BIAS_DACBUFBP	60
4.1.2.190 DAVIS640_CONFIG_BIAS_DIFFBN	60
4.1.2.191 DAVIS640_CONFIG_BIAS_IFREFRBN	60
4.1.2.192 DAVIS640_CONFIG_BIAS_IFTHRBN	61
4.1.2.193 DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN	61
4.1.2.194 DAVIS640_CONFIG_BIAS_LOCALBUFBN	61
4.1.2.195 DAVIS640_CONFIG_BIAS_OFFBN	61
4.1.2.196 DAVIS640_CONFIG_BIAS_ONBN	61
4.1.2.197 DAVIS640_CONFIG_BIAS_PADFOLLBN	62
4.1.2.198 DAVIS640_CONFIG_BIAS_PIXINBN	62
4.1.2.199 DAVIS640_CONFIG_BIAS_PRBP	62
4.1.2.200 DAVIS640_CONFIG_BIAS_PRSEFBP	62
4.1.2.201 DAVIS640_CONFIG_BIAS_READOUTBUFBP	62
4.1.2.202 DAVIS640_CONFIG_BIAS_REFRBP	63
4.1.2.203 DAVIS640_CONFIG_BIAS_SSN	63
4.1.2.204 DAVIS640_CONFIG_BIAS_SSP	63
4.1.2.205 DAVIS640_CONFIG_CHIP_AERNAROW	63
4.1.2.206 DAVIS640_CONFIG_CHIP_ANALOGMUX0	63
4.1.2.207 DAVIS640_CONFIG_CHIP_ANALOGMUX1	63
4.1.2.208 DAVIS640_CONFIG_CHIP_ANALOGMUX2	64
4.1.2.209 DAVIS640_CONFIG_CHIP_BIASMUX0	64
4.1.2.210 DAVIS640_CONFIG_CHIP_DIGITALMUX0	64
4.1.2.211 DAVIS640_CONFIG_CHIP_DIGITALMUX1	64
4.1.2.212 DAVIS640_CONFIG_CHIP_DIGITALMUX2	64

4.1.2.213 DAVIS640_CONFIG_CHIP_DIGITALMUX3	64
4.1.2.214 DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER	64
4.1.2.215 DAVIS640_CONFIG_CHIP_RESETCALIBNEURON	64
4.1.2.216 DAVIS640_CONFIG_CHIP_RESETTESTPIXEL	64
4.1.2.217 DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER	65
4.1.2.218 DAVIS640_CONFIG_CHIP_TESTADC	65
4.1.2.219 DAVIS640_CONFIG_CHIP_TYPCENCALIBNEURON	65
4.1.2.220 DAVIS640_CONFIG_CHIP_USEAOUT	65
4.1.2.221 DAVIS_CHIP_DAVIS128	65
4.1.2.222 DAVIS_CHIP_DAVIS208	65
4.1.2.223 DAVIS_CHIP_DAVIS240A	65
4.1.2.224 DAVIS_CHIP_DAVIS240B	65
4.1.2.225 DAVIS_CHIP_DAVIS240C	65
4.1.2.226 DAVIS_CHIP_DAVIS346A	65
4.1.2.227 DAVIS_CHIP_DAVIS346B	66
4.1.2.228 DAVIS_CHIP_DAVIS346C	66
4.1.2.229 DAVIS_CHIP_DAVIS640	66
4.1.2.230 DAVIS_CHIP_DAVISRGB	66
4.1.2.231 DAVIS_CONFIG_APS	66
4.1.2.232 DAVIS_CONFIG_APS_ADC_TEST_MODE	66
4.1.2.233 DAVIS_CONFIG_APS_COLOR_FILTER	66
4.1.2.234 DAVIS_CONFIG_APS_COLUMN_SETTLE	66
4.1.2.235 DAVIS_CONFIG_APS_END_COLUMN_0	66
4.1.2.236 DAVIS_CONFIG_APS_END_COLUMN_1	66
4.1.2.237 DAVIS_CONFIG_APS_END_COLUMN_2	66
4.1.2.238 DAVIS_CONFIG_APS_END_COLUMN_3	67
4.1.2.239 DAVIS_CONFIG_APS_END_ROW_0	67
4.1.2.240 DAVIS_CONFIG_APS_END_ROW_1	67
4.1.2.241 DAVIS_CONFIG_APS_END_ROW_2	67
4.1.2.242 DAVIS_CONFIG_APS_END_ROW_3	67
4.1.2.243 DAVIS_CONFIG_APS_EXPOSURE	67
4.1.2.244 DAVIS_CONFIG_APS_FRAME_DELAY	67
4.1.2.245 DAVIS_CONFIG_APS_GLOBAL_SHUTTER	67
4.1.2.246 DAVIS_CONFIG_APS_HAS_EXTERNAL_ADC	67
4.1.2.247 DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER	67
4.1.2.248 DAVIS_CONFIG_APS_HAS_INTERNAL_ADC	68
4.1.2.249 DAVIS_CONFIG_APS_HAS_QUAD_ROI	68
4.1.2.250 DAVIS_CONFIG_APS_NULL_SETTLE	68
4.1.2.251 DAVIS_CONFIG_APS_ORIENTATION_INFO	68
4.1.2.252 DAVIS_CONFIG_APS_RAMP_RESET	68

4.1.2.253 DAVIS_CONFIG_APS_RAMP_SHORT_RESET	68
4.1.2.254 DAVIS_CONFIG_APS_RESET_READ	68
4.1.2.255 DAVIS_CONFIG_APS_RESET_SETTLE	68
4.1.2.256 DAVIS_CONFIG_APS_ROW_SETTLE	68
4.1.2.257 DAVIS_CONFIG_APS_RUN	69
4.1.2.258 DAVIS_CONFIG_APS_SAMPLE_ENABLE	69
4.1.2.259 DAVIS_CONFIG_APS_SAMPLE_SETTLE	69
4.1.2.260 DAVIS_CONFIG_APS_SIZE_COLUMNS	69
4.1.2.261 DAVIS_CONFIG_APS_SIZE_ROWS	69
4.1.2.262 DAVIS_CONFIG_APS_SNAPSHOT	69
4.1.2.263 DAVIS_CONFIG_APS_START_COLUMN_0	69
4.1.2.264 DAVIS_CONFIG_APS_START_COLUMN_1	69
4.1.2.265 DAVIS_CONFIG_APS_START_COLUMN_2	70
4.1.2.266 DAVIS_CONFIG_APS_START_COLUMN_3	70
4.1.2.267 DAVIS_CONFIG_APS_START_ROW_0	70
4.1.2.268 DAVIS_CONFIG_APS_START_ROW_1	70
4.1.2.269 DAVIS_CONFIG_APS_START_ROW_2	70
4.1.2.270 DAVIS_CONFIG_APS_START_ROW_3	70
4.1.2.271 DAVIS_CONFIG_APS_USE_INTERNAL_ADC	70
4.1.2.272 DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL	70
4.1.2.273 DAVIS_CONFIG_BIAS	70
4.1.2.274 DAVIS_CONFIG_CHIP	71
4.1.2.275 DAVIS_CONFIG_DVS	71
4.1.2.276 DAVIS_CONFIG_DVS_ACK_DELAY_COLUMN	71
4.1.2.277 DAVIS_CONFIG_DVS_ACK_DELAY_ROW	71
4.1.2.278 DAVIS_CONFIG_DVS_ACK_EXTENSION_COLUMN	71
4.1.2.279 DAVIS_CONFIG_DVS_ACK_EXTENSION_ROW	71
4.1.2.280 DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL	71
4.1.2.281 DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY	71
4.1.2.282 DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_DELTAT	71
4.1.2.283 DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN	72
4.1.2.284 DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW	72
4.1.2.285 DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN	72
4.1.2.286 DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW	72
4.1.2.287 DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN	72
4.1.2.288 DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW	72
4.1.2.289 DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN	72
4.1.2.290 DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW	72
4.1.2.291 DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN	72
4.1.2.292 DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW	72

4.1.2.293 DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN	73
4.1.2.294 DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW	73
4.1.2.295 DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN	73
4.1.2.296 DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW	73
4.1.2.297 DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN	73
4.1.2.298 DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW	73
4.1.2.299 DAVIS_CONFIG_DVS_FILTER_ROW_ONLY_EVENTS	73
4.1.2.300 DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER	73
4.1.2.301 DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER	73
4.1.2.302 DAVIS_CONFIG_DVS_HAS_TEST_EVENT_GENERATOR	74
4.1.2.303 DAVIS_CONFIG_DVS_ORIENTATION_INFO	74
4.1.2.304 DAVIS_CONFIG_DVS_RUN	74
4.1.2.305 DAVIS_CONFIG_DVS_SIZE_COLUMNS	74
4.1.2.306 DAVIS_CONFIG_DVS_SIZE_ROWS	74
4.1.2.307 DAVIS_CONFIG_DVS_TEST_EVENT_GENERATOR_ENABLE	74
4.1.2.308 DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL	74
4.1.2.309 DAVIS_CONFIG_EXTINPUT	74
4.1.2.310 DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES	75
4.1.2.311 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH	75
4.1.2.312 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY	75
4.1.2.313 DAVIS_CONFIG_EXTINPUT_DETECT_PULSES	75
4.1.2.314 DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES	75
4.1.2.315 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL	75
4.1.2.316 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH	75
4.1.2.317 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY	75
4.1.2.318 DAVIS_CONFIG_EXTINPUT_GENERATE_USE_CUSTOM_SIGNAL	76
4.1.2.319 DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR	76
4.1.2.320 DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR	76
4.1.2.321 DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR	76
4.1.2.322 DAVIS_CONFIG_IMU	76
4.1.2.323 DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE	76
4.1.2.324 DAVIS_CONFIG_IMU_ACCEL_STANDBY	76
4.1.2.325 DAVIS_CONFIG_IMU_DIGITAL_LOW_PASS_FILTER	76
4.1.2.326 DAVIS_CONFIG_IMU_GYRO_FULL_SCALE	77
4.1.2.327 DAVIS_CONFIG_IMU_GYRO_STANDBY	77
4.1.2.328 DAVIS_CONFIG_IMU_LP_CYCLE	77
4.1.2.329 DAVIS_CONFIG_IMU_LP_WAKEUP	77
4.1.2.330 DAVIS_CONFIG_IMU_RUN	77
4.1.2.331 DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER	77
4.1.2.332 DAVIS_CONFIG_IMU_TEMP_STANDBY	77

4.1.2.333 DAVIS_CONFIG_MUX	77
4.1.2.334 DAVIS_CONFIG_MUX_DROP_APS_ON_TRANSFER_STALL	77
4.1.2.335 DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL	78
4.1.2.336 DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL	78
4.1.2.337 DAVIS_CONFIG_MUX_DROP_IMU_ON_TRANSFER_STALL	78
4.1.2.338 DAVIS_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE	78
4.1.2.339 DAVIS_CONFIG_MUX_RUN	78
4.1.2.340 DAVIS_CONFIG_MUX_TIMESTAMP_RESET	78
4.1.2.341 DAVIS_CONFIG_MUX_TIMESTAMP_RUN	78
4.1.2.342 DAVIS_CONFIG_SYSINFO	78
4.1.2.343 DAVIS_CONFIG_SYSINFO_ADC_CLOCK	78
4.1.2.344 DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER	79
4.1.2.345 DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER	79
4.1.2.346 DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK	79
4.1.2.347 DAVIS_CONFIG_SYSINFO_LOGIC_VERSION	79
4.1.2.348 DAVIS_CONFIG_USB	79
4.1.2.349 DAVIS_CONFIG_USB_EARLY_PACKET_DELAY	79
4.1.2.350 DAVIS_CONFIG_USB_RUN	79
4.1.2.351 DAVISRGB_CONFIG_APS_GSFDRESET	79
4.1.2.352 DAVISRGB_CONFIG_APS_GSPDRESET	80
4.1.2.353 DAVISRGB_CONFIG_APS_GSRESETFALL	80
4.1.2.354 DAVISRGB_CONFIG_APS_GSTXFALL	80
4.1.2.355 DAVISRGB_CONFIG_APS_RSFDSETTLE	80
4.1.2.356 DAVISRGB_CONFIG_APS_TRANSFER	80
4.1.2.357 DAVISRGB_CONFIG_BIAS_ADCCOMBPB	80
4.1.2.358 DAVISRGB_CONFIG_BIAS_ADCREFHIGH	80
4.1.2.359 DAVISRGB_CONFIG_BIAS_ADCREFLOW	81
4.1.2.360 DAVISRGB_CONFIG_BIAS_ADCTESTVOLTAGE	81
4.1.2.361 DAVISRGB_CONFIG_BIAS_AEPDBN	81
4.1.2.362 DAVISRGB_CONFIG_BIAS_AEPUXBP	81
4.1.2.363 DAVISRGB_CONFIG_BIAS_AEPUYBP	81
4.1.2.364 DAVISRGB_CONFIG_BIAS_APSCAS	82
4.1.2.365 DAVISRGB_CONFIG_BIAS_APSROSFBN	82
4.1.2.366 DAVISRGB_CONFIG_BIAS_ARRAYBIASBUFFERBN	82
4.1.2.367 DAVISRGB_CONFIG_BIAS_ARRAYLOGICBUFFERBN	82
4.1.2.368 DAVISRGB_CONFIG_BIAS_BIASBUFFER	82
4.1.2.369 DAVISRGB_CONFIG_BIAS_DACBUFBP	83
4.1.2.370 DAVISRGB_CONFIG_BIAS_DIFFBN	83
4.1.2.371 DAVISRGB_CONFIG_BIAS_FALLTIMEBN	83
4.1.2.372 DAVISRGB_CONFIG_BIAS_GND07	83

4.1.2.373 DAVISRGB_CONFIG_BIAS_IFREFRBN	83
4.1.2.374 DAVISRGB_CONFIG_BIAS_IFTHRBN	84
4.1.2.375 DAVISRGB_CONFIG_BIAS_LCOLTIMEOUTBN	84
4.1.2.376 DAVISRGB_CONFIG_BIAS_LOCALBUFBN	84
4.1.2.377 DAVISRGB_CONFIG_BIAS_OFFBN	84
4.1.2.378 DAVISRGB_CONFIG_BIAS_ONBN	84
4.1.2.379 DAVISRGB_CONFIG_BIAS_OVG1LO	85
4.1.2.380 DAVISRGB_CONFIG_BIAS_OVG2LO	85
4.1.2.381 DAVISRGB_CONFIG_BIAS_PADFOLLBN	85
4.1.2.382 DAVISRGB_CONFIG_BIAS_PIXINBN	85
4.1.2.383 DAVISRGB_CONFIG_BIAS_PRBP	85
4.1.2.384 DAVISRGB_CONFIG_BIAS_PRSFBP	86
4.1.2.385 DAVISRGB_CONFIG_BIAS_READOUTBUFBP	86
4.1.2.386 DAVISRGB_CONFIG_BIAS_REFRBP	86
4.1.2.387 DAVISRGB_CONFIG_BIAS_RISETIMEBP	86
4.1.2.388 DAVISRGB_CONFIG_BIAS_SSN	86
4.1.2.389 DAVISRGB_CONFIG_BIAS_SSP	87
4.1.2.390 DAVISRGB_CONFIG_BIAS_TX2OVG2HI	87
4.1.2.391 DAVISRGB_CONFIG_CHIP_ADJUSTOVG1LO	87
4.1.2.392 DAVISRGB_CONFIG_CHIP_ADJUSTOVG2LO	87
4.1.2.393 DAVISRGB_CONFIG_CHIP_ADJUSTTX2OVG2HI	87
4.1.2.394 DAVISRGB_CONFIG_CHIP_AERNAROW	87
4.1.2.395 DAVISRGB_CONFIG_CHIP_ANALOGMUX0	87
4.1.2.396 DAVISRGB_CONFIG_CHIP_ANALOGMUX1	88
4.1.2.397 DAVISRGB_CONFIG_CHIP_ANALOGMUX2	88
4.1.2.398 DAVISRGB_CONFIG_CHIP_BIASMUX0	88
4.1.2.399 DAVISRGB_CONFIG_CHIP_DIGITALMUX0	88
4.1.2.400 DAVISRGB_CONFIG_CHIP_DIGITALMUX1	88
4.1.2.401 DAVISRGB_CONFIG_CHIP_DIGITALMUX2	88
4.1.2.402 DAVISRGB_CONFIG_CHIP_DIGITALMUX3	88
4.1.2.403 DAVISRGB_CONFIG_CHIP_RESETCALIBNEURON	88
4.1.2.404 DAVISRGB_CONFIG_CHIP_RESETTESTPIXEL	88
4.1.2.405 DAVISRGB_CONFIG_CHIP_SELECTGRAYCOUNTER	89
4.1.2.406 DAVISRGB_CONFIG_CHIP_TESTADC	89
4.1.2.407 DAVISRGB_CONFIG_CHIP_TYPENCALIBNEURON	89
4.1.2.408 DAVISRGB_CONFIG_CHIP_USEAOUT	89
4.1.2.409 IS_DAVIS128	89
4.1.2.410 IS_DAVIS208	89
4.1.2.411 IS_DAVIS240	89
4.1.2.412 IS_DAVIS240A	89

4.1.2.413	IS_DAVIS240B	89
4.1.2.414	IS_DAVIS240C	90
4.1.2.415	IS_DAVIS346	90
4.1.2.416	IS_DAVIS346A	90
4.1.2.417	IS_DAVIS346B	90
4.1.2.418	IS_DAVIS346C	90
4.1.2.419	IS_DAVIS640	90
4.1.2.420	IS_DAVISRGB	90
4.1.3	Enumeration Type Documentation	90
4.1.3.1	caer_bias_shiftedsource_operating_mode	90
4.1.3.2	caer_bias_shiftedsource_voltage_level	90
4.1.4	Function Documentation	91
4.1.4.1	caerBiasCoarseFineGenerate(struct caer_bias_coarsefine coarseFineBias)	91
4.1.4.2	caerBiasCoarseFineParse(uint16_t coarseFineBias)	91
4.1.4.3	caerBiasShiftedSourceGenerate(struct caer_bias_shiftedsource shiftedSource↵ Bias)	91
4.1.4.4	caerBiasShiftedSourceParse(uint16_t shiftedSourceBias)	91
4.1.4.5	caerBiasVDACGenerate(struct caer_bias_vdac vdacBias)	92
4.1.4.6	caerBiasVDACParse(uint16_t vdacBias)	92
4.1.4.7	caerDavisInfoGet(caerDeviceHandle handle)	92
4.2	devices/dvs128.h File Reference	92
4.2.1	Detailed Description	93
4.2.2	Macro Definition Documentation	93
4.2.2.1	CAER_DEVICE_DVS128	93
4.2.2.2	DVS128_CONFIG_BIAS	93
4.2.2.3	DVS128_CONFIG_BIAS_CAS	93
4.2.2.4	DVS128_CONFIG_BIAS_DIFF	94
4.2.2.5	DVS128_CONFIG_BIAS_DIFFOFF	94
4.2.2.6	DVS128_CONFIG_BIAS_DIFFON	94
4.2.2.7	DVS128_CONFIG_BIAS_FOLL	94
4.2.2.8	DVS128_CONFIG_BIAS_INJGND	94
4.2.2.9	DVS128_CONFIG_BIAS_PR	94
4.2.2.10	DVS128_CONFIG_BIAS_PUX	94
4.2.2.11	DVS128_CONFIG_BIAS_PUY	94
4.2.2.12	DVS128_CONFIG_BIAS_REFR	94
4.2.2.13	DVS128_CONFIG_BIAS_REQ	94
4.2.2.14	DVS128_CONFIG_BIAS_REQPD	95
4.2.2.15	DVS128_CONFIG_DVS	95
4.2.2.16	DVS128_CONFIG_DVS_ARRAY_RESET	95
4.2.2.17	DVS128_CONFIG_DVS_RUN	95

4.2.2.18	DVS128_CONFIG_DVS_TIMESTAMP_RESET	95
4.2.2.19	DVS128_CONFIG_DVS_TS_MASTER	95
4.2.3	Function Documentation	95
4.2.3.1	caerDVS128InfoGet(caerDeviceHandle handle)	95
4.3	devices/usb.h File Reference	95
4.3.1	Detailed Description	96
4.3.2	Macro Definition Documentation	96
4.3.2.1	CAER_HOST_CONFIG_DATAEXCHANGE	96
4.3.2.2	CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING	97
4.3.2.3	CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE	97
4.3.2.4	CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS	97
4.3.2.5	CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS	97
4.3.2.6	CAER_HOST_CONFIG_PACKETS	97
4.3.2.7	CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL	97
4.3.2.8	CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_SIZE	97
4.3.2.9	CAER_HOST_CONFIG_PACKETS_MAX_FRAME_INTERVAL	97
4.3.2.10	CAER_HOST_CONFIG_PACKETS_MAX_FRAME_SIZE	98
4.3.2.11	CAER_HOST_CONFIG_PACKETS_MAX_IMU6_INTERVAL	98
4.3.2.12	CAER_HOST_CONFIG_PACKETS_MAX_IMU6_SIZE	98
4.3.2.13	CAER_HOST_CONFIG_PACKETS_MAX_POLARITY_INTERVAL	98
4.3.2.14	CAER_HOST_CONFIG_PACKETS_MAX_POLARITY_SIZE	98
4.3.2.15	CAER_HOST_CONFIG_PACKETS_MAX_SPECIAL_INTERVAL	98
4.3.2.16	CAER_HOST_CONFIG_PACKETS_MAX_SPECIAL_SIZE	98
4.3.2.17	CAER_HOST_CONFIG_USB	98
4.3.2.18	CAER_HOST_CONFIG_USB_BUFFER_NUMBER	98
4.3.2.19	CAER_HOST_CONFIG_USB_BUFFER_SIZE	99
4.3.3	Typedef Documentation	99
4.3.3.1	caerDeviceHandle	99
4.3.4	Function Documentation	99
4.3.4.1	caerDeviceClose(caerDeviceHandle *handle)	99
4.3.4.2	caerDeviceConfigGet(caerDeviceHandle handle, int8_t modAddr, uint8_t paramAddr, uint32_t *param)	99
4.3.4.3	caerDeviceConfigSet(caerDeviceHandle handle, int8_t modAddr, uint8_t paramAddr, uint32_t param)	99
4.3.4.4	caerDeviceDataGet(caerDeviceHandle handle)	100
4.3.4.5	caerDeviceDataStart(caerDeviceHandle handle, void(*dataNotifyIncrease)(void *ptr), void(*dataNotifyDecrease)(void *ptr), void *dataNotifyUserPtr, void(*dataShutdownNotify)(void *ptr), void *dataShutdownUserPtr)	100
4.3.4.6	caerDeviceDataStop(caerDeviceHandle handle)	101
4.3.4.7	caerDeviceOpen(uint16_t deviceID, uint16_t deviceType, uint8_t busNumber, Restrict, uint8_t devAddressRestrict, const char *serialNumberRestrict)	101

4.3.4.8	<code>caerDeviceSendDefaultConfig(caerDeviceHandle handle)</code>	101
4.4	<code>events/common.h</code> File Reference	101
4.4.1	Detailed Description	103
4.4.2	Macro Definition Documentation	103
4.4.2.1	<code>CAER_EVENT_PACKET_HEADER_SIZE</code>	103
4.4.2.2	<code>CAER_ITERATOR_ALL_END</code>	103
4.4.2.3	<code>CAER_ITERATOR_ALL_START</code>	103
4.4.2.4	<code>CAER_ITERATOR_VALID_END</code>	103
4.4.2.5	<code>CAER_ITERATOR_VALID_START</code>	104
4.4.2.6	<code>TS_OVERFLOW_SHIFT</code>	104
4.4.2.7	<code>VALID_MARK_MASK</code>	104
4.4.2.8	<code>VALID_MARK_SHIFT</code>	104
4.4.3	Typedef Documentation	104
4.4.3.1	<code>caerEventPacketHeader</code>	104
4.4.4	Enumeration Type Documentation	104
4.4.4.1	<code>caer_default_event_types</code>	104
4.4.5	Function Documentation	105
4.4.5.1	<code>caerCopyEventPacket(void *eventPacket)</code>	105
4.4.5.2	<code>caerCopyEventPacketOnlyEvents(void *eventPacket)</code>	105
4.4.5.3	<code>caerCopyEventPacketOnlyValidEvents(void *eventPacket)</code>	105
4.4.5.4	<code>caerEventPacketHeaderGetEventCapacity(caerEventPacketHeader header)</code>	105
4.4.5.5	<code>caerEventPacketHeaderGetEventNumber(caerEventPacketHeader header)</code>	106
4.4.5.6	<code>caerEventPacketHeaderGetEventSize(caerEventPacketHeader header)</code>	106
4.4.5.7	<code>caerEventPacketHeaderGetEventSource(caerEventPacketHeader header)</code>	106
4.4.5.8	<code>caerEventPacketHeaderGetEventTSOffset(caerEventPacketHeader header)</code>	106
4.4.5.9	<code>caerEventPacketHeaderGetEventTSOverflow(caerEventPacketHeader header)</code>	107
4.4.5.10	<code>caerEventPacketHeaderGetEventType(caerEventPacketHeader header)</code>	107
4.4.5.11	<code>caerEventPacketHeaderGetEventValid(caerEventPacketHeader header)</code>	107
4.4.5.12	<code>caerEventPacketHeaderSetEventCapacity(caerEventPacketHeader header, int32_t eventsCapacity)</code>	107
4.4.5.13	<code>caerEventPacketHeaderSetEventNumber(caerEventPacketHeader header, int32_t eventsNumber)</code>	108
4.4.5.14	<code>caerEventPacketHeaderSetEventSize(caerEventPacketHeader header, int32_t eventSize)</code>	108
4.4.5.15	<code>caerEventPacketHeaderSetEventSource(caerEventPacketHeader header, int16_t eventSource)</code>	108
4.4.5.16	<code>caerEventPacketHeaderSetEventTSOffset(caerEventPacketHeader header, int32_t eventTSOffset)</code>	108
4.4.5.17	<code>caerEventPacketHeaderSetEventTSOverflow(caerEventPacketHeader header, int32_t eventTSOverflow)</code>	108
4.4.5.18	<code>caerEventPacketHeaderSetEventType(caerEventPacketHeader header, int16_t eventType)</code>	109

4.4.5.19	caerEventPacketHeaderSetEventValid(caerEventPacketHeader header, int32_t eventsValid)	109
4.4.5.20	caerGenericEventGetEvent(caerEventPacketHeader headerPtr, int32_t n)	109
4.4.5.21	caerGenericEventGetTimestamp(void *eventPtr, caerEventPacketHeader headerPtr)	109
4.4.5.22	caerGenericEventGetTimestamp64(void *eventPtr, caerEventPacketHeader headerPtr)	110
4.4.5.23	caerGenericEventsValid(void *eventPtr)	110
4.5	events/config.h File Reference	110
4.5.1	Detailed Description	111
4.5.2	Macro Definition Documentation	111
4.5.2.1	CAER_CONFIGURATION_ITERATOR_ALL_END	111
4.5.2.2	CAER_CONFIGURATION_ITERATOR_ALL_START	112
4.5.2.3	CAER_CONFIGURATION_ITERATOR_VALID_END	112
4.5.2.4	CAER_CONFIGURATION_ITERATOR_VALID_START	112
4.5.2.5	MODULE_ADDR_MASK	112
4.5.2.6	MODULE_ADDR_SHIFT	112
4.5.3	Typedef Documentation	112
4.5.3.1	caerConfigurationEvent	112
4.5.3.2	caerConfigurationEventPacket	113
4.5.4	Function Documentation	113
4.5.4.1	caerConfigurationEventGetModuleAddress(caerConfigurationEvent event)	113
4.5.4.2	caerConfigurationEventGetParameter(caerConfigurationEvent event)	113
4.5.4.3	caerConfigurationEventGetParameterAddress(caerConfigurationEvent event)	113
4.5.4.4	caerConfigurationEventGetTimestamp(caerConfigurationEvent event)	113
4.5.4.5	caerConfigurationEventGetTimestamp64(caerConfigurationEvent event, caerConfigurationEventPacket packet)	114
4.5.4.6	caerConfigurationEventInvalidate(caerConfigurationEvent event, caerConfigurationEventPacket packet)	114
4.5.4.7	caerConfigurationEventsValid(caerConfigurationEvent event)	114
4.5.4.8	caerConfigurationEventPacketAllocate(int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)	114
4.5.4.9	caerConfigurationEventPacketGetEvent(caerConfigurationEventPacket packet, int32_t n)	115
4.5.4.10	caerConfigurationEventSetModuleAddress(caerConfigurationEvent event, uint8_t moduleAddress)	115
4.5.4.11	caerConfigurationEventSetParameter(caerConfigurationEvent event, uint32_t parameter)	115
4.5.4.12	caerConfigurationEventSetParameterAddress(caerConfigurationEvent event, uint8_t parameterAddress)	115
4.5.4.13	caerConfigurationEventSetTimestamp(caerConfigurationEvent event, int32_t timestamp)	116
4.5.4.14	caerConfigurationEventValidate(caerConfigurationEvent event, caerConfigurationEventPacket packet)	116

4.6	events/ear.h File Reference	116
4.6.1	Detailed Description	117
4.6.2	Macro Definition Documentation	117
4.6.2.1	CAER_EAR_ITERATOR_ALL_END	117
4.6.2.2	CAER_EAR_ITERATOR_ALL_START	118
4.6.2.3	CAER_EAR_ITERATOR_VALID_END	118
4.6.2.4	CAER_EAR_ITERATOR_VALID_START	118
4.6.2.5	CHANNEL_MASK	118
4.6.2.6	CHANNEL_SHIFT	118
4.6.2.7	EAR_MASK	118
4.6.2.8	EAR_SHIFT	118
4.6.2.9	FILTER_MASK	119
4.6.2.10	FILTER_SHIFT	119
4.6.2.11	NEURON_MASK	119
4.6.2.12	NEURON_SHIFT	119
4.6.3	Typedef Documentation	119
4.6.3.1	caerEarEvent	119
4.6.3.2	caerEarEventPacket	119
4.6.4	Function Documentation	119
4.6.4.1	caerEarEventGetChannel(caerEarEvent event)	119
4.6.4.2	caerEarEventGetEar(caerEarEvent event)	120
4.6.4.3	caerEarEventGetTimestamp(caerEarEvent event)	121
4.6.4.4	caerEarEventGetTimestamp64(caerEarEvent event, caerEarEventPacket packet)	121
4.6.4.5	caerEarEventInvalidate(caerEarEvent event, caerEarEventPacket packet)	121
4.6.4.6	caerEarEventIsValid(caerEarEvent event)	121
4.6.4.7	caerEarEventPacketAllocate(int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)	122
4.6.4.8	caerEarEventPacketGetEvent(caerEarEventPacket packet, int32_t n)	122
4.6.4.9	caerEarEventSetChannel(caerEarEvent event, uint16_t channel)	122
4.6.4.10	caerEarEventSetEar(caerEarEvent event, uint8_t ear)	122
4.6.4.11	caerEarEventSetTimestamp(caerEarEvent event, int32_t timestamp)	123
4.6.4.12	caerEarEventValidate(caerEarEvent event, caerEarEventPacket packet)	123
4.7	events/frame.h File Reference	123
4.7.1	Detailed Description	125
4.7.2	Macro Definition Documentation	126
4.7.2.1	CAER_FRAME_ITERATOR_ALL_END	126
4.7.2.2	CAER_FRAME_ITERATOR_ALL_START	126
4.7.2.3	CAER_FRAME_ITERATOR_VALID_END	126
4.7.2.4	CAER_FRAME_ITERATOR_VALID_START	126
4.7.2.5	COLOR_CHANNELS_MASK	126

4.7.2.6	COLOR_CHANNELS_SHIFT	126
4.7.2.7	COLOR_FILTER_MASK	127
4.7.2.8	COLOR_FILTER_SHIFT	127
4.7.2.9	ROI_IDENTIFIER_MASK	127
4.7.2.10	ROI_IDENTIFIER_SHIFT	127
4.7.3	Typedef Documentation	127
4.7.3.1	caerFrameEvent	127
4.7.3.2	caerFrameEventPacket	127
4.7.4	Enumeration Type Documentation	127
4.7.4.1	caer_frame_event_color_channels	127
4.7.4.2	caer_frame_event_color_filter	128
4.7.5	Function Documentation	128
4.7.5.1	caerFrameEventGetChannelNumber(caerFrameEvent event)	128
4.7.5.2	caerFrameEventGetColorFilter(caerFrameEvent event)	128
4.7.5.3	caerFrameEventGetExposureLength(caerFrameEvent event)	128
4.7.5.4	caerFrameEventGetLengthX(caerFrameEvent event)	128
4.7.5.5	caerFrameEventGetLengthY(caerFrameEvent event)	129
4.7.5.6	caerFrameEventGetPixel(caerFrameEvent event, int32_t xAddress, int32_t y↔ Address)	129
4.7.5.7	caerFrameEventGetPixelArrayCGFormat(caerFrameEvent event)	129
4.7.5.8	caerFrameEventGetPixelArrayUnsafe(caerFrameEvent event)	129
4.7.5.9	caerFrameEventGetPixelForChannel(caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint8_t channel)	130
4.7.5.10	caerFrameEventGetPixelForChannelUnsafe(caerFrameEvent event, int32_t x↔ Address, int32_t yAddress, uint8_t channel)	130
4.7.5.11	caerFrameEventGetPixelsMaxIndex(caerFrameEvent event)	130
4.7.5.12	caerFrameEventGetPixelsSize(caerFrameEvent event)	130
4.7.5.13	caerFrameEventGetPixelUnsafe(caerFrameEvent event, int32_t xAddress, int32_t yAddress)	131
4.7.5.14	caerFrameEventGetPositionX(caerFrameEvent event)	131
4.7.5.15	caerFrameEventGetPositionY(caerFrameEvent event)	131
4.7.5.16	caerFrameEventGetROIIdentifier(caerFrameEvent event)	131
4.7.5.17	caerFrameEventGetTimestamp(caerFrameEvent event)	132
4.7.5.18	caerFrameEventGetTimestamp64(caerFrameEvent event, caerFrameEvent↔ Packet packet)	132
4.7.5.19	caerFrameEventGetTSEndOfExposure(caerFrameEvent event)	132
4.7.5.20	caerFrameEventGetTSEndOfExposure64(caerFrameEvent event, caerFrame↔ EventPacket packet)	132
4.7.5.21	caerFrameEventGetTSEndOfFrame(caerFrameEvent event)	133
4.7.5.22	caerFrameEventGetTSEndOfFrame64(caerFrameEvent event, caerFrame↔ EventPacket packet)	133
4.7.5.23	caerFrameEventGetTSStartOfExposure(caerFrameEvent event)	133

4.7.5.24	<code>caerFrameEventGetTSSStartOfExposure64(caerFrameEvent event, caerFrameEventPacket packet)</code>	134
4.7.5.25	<code>caerFrameEventGetTSSStartOfFrame(caerFrameEvent event)</code>	135
4.7.5.26	<code>caerFrameEventGetTSSStartOfFrame64(caerFrameEvent event, caerFrameEventPacket packet)</code>	135
4.7.5.27	<code>caerFrameEventInvalidate(caerFrameEvent event, caerFrameEventPacket packet)</code>	135
4.7.5.28	<code>caerFrameEventIsValid(caerFrameEvent event)</code>	135
4.7.5.29	<code>caerFrameEventPacketAllocate(int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow, int32_t maxLengthX, int32_t maxLengthY, int16_t maxChannelNumber)</code>	136
4.7.5.30	<code>caerFrameEventPacketGetEvent(caerFrameEventPacket packet, int32_t n)</code>	136
4.7.5.31	<code>caerFrameEventPacketGetPixelsMaxIndex(caerFrameEventPacket packet)</code>	136
4.7.5.32	<code>caerFrameEventPacketGetPixelsSize(caerFrameEventPacket packet)</code>	137
4.7.5.33	<code>caerFrameEventSetColorFilter(caerFrameEvent event, enum caer_frame_event_color_filter colorFilter)</code>	137
4.7.5.34	<code>caerFrameEventSetLengthXLengthYChannelNumber(caerFrameEvent event, int32_t lengthX, int32_t lengthY, enum caer_frame_event_color_channels channelNumber, caerFrameEventPacket packet)</code>	137
4.7.5.35	<code>caerFrameEventSetPixel(caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint16_t pixelValue)</code>	137
4.7.5.36	<code>caerFrameEventSetPixelForChannel(caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint8_t channel, uint16_t pixelValue)</code>	138
4.7.5.37	<code>caerFrameEventSetPixelForChannelUnsafe(caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint8_t channel, uint16_t pixelValue)</code>	138
4.7.5.38	<code>caerFrameEventSetPixelUnsafe(caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint16_t pixelValue)</code>	138
4.7.5.39	<code>caerFrameEventSetPositionX(caerFrameEvent event, int32_t positionX)</code>	138
4.7.5.40	<code>caerFrameEventSetPositionY(caerFrameEvent event, int32_t positionY)</code>	139
4.7.5.41	<code>caerFrameEventSetROIIdentifier(caerFrameEvent event, uint8_t roiIdentifier)</code>	139
4.7.5.42	<code>caerFrameEventSetTSEndOfExposure(caerFrameEvent event, int32_t endExposure)</code>	139
4.7.5.43	<code>caerFrameEventSetTSEndOfFrame(caerFrameEvent event, int32_t endFrame)</code>	139
4.7.5.44	<code>caerFrameEventSetTSSStartOfExposure(caerFrameEvent event, int32_t startExposure)</code>	139
4.7.5.45	<code>caerFrameEventSetTSSStartOfFrame(caerFrameEvent event, int32_t startFrame)</code>	140
4.7.5.46	<code>caerFrameEventValidate(caerFrameEvent event, caerFrameEventPacket packet)</code>	140
4.7.6	Variable Documentation	140
4.7.6.1	<code>pixels</code>	140
4.8	events/imu6.h File Reference	140
4.8.1	Detailed Description	142
4.8.2	Macro Definition Documentation	142
4.8.2.1	<code>CAER_IMU6_ITERATOR_ALL_END</code>	142
4.8.2.2	<code>CAER_IMU6_ITERATOR_ALL_START</code>	142
4.8.2.3	<code>CAER_IMU6_ITERATOR_VALID_END</code>	142

4.8.2.4	CAER_IMU6_ITERATOR_VALID_START	142
4.8.3	Typedef Documentation	142
4.8.3.1	caerIMU6Event	142
4.8.3.2	caerIMU6EventPacket	143
4.8.4	Function Documentation	143
4.8.4.1	caerIMU6EventGetAccelX(caerIMU6Event event)	143
4.8.4.2	caerIMU6EventGetAccelY(caerIMU6Event event)	143
4.8.4.3	caerIMU6EventGetAccelZ(caerIMU6Event event)	143
4.8.4.4	caerIMU6EventGetGyroX(caerIMU6Event event)	143
4.8.4.5	caerIMU6EventGetGyroY(caerIMU6Event event)	143
4.8.4.6	caerIMU6EventGetGyroZ(caerIMU6Event event)	144
4.8.4.7	caerIMU6EventGetTemp(caerIMU6Event event)	144
4.8.4.8	caerIMU6EventGetTimestamp(caerIMU6Event event)	144
4.8.4.9	caerIMU6EventGetTimestamp64(caerIMU6Event event, caerIMU6EventPacket packet)	144
4.8.4.10	caerIMU6EventInvalidate(caerIMU6Event event, caerIMU6EventPacket packet)	145
4.8.4.11	caerIMU6EventIsValid(caerIMU6Event event)	145
4.8.4.12	caerIMU6EventPacketAllocate(int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)	145
4.8.4.13	caerIMU6EventPacketGetEvent(caerIMU6EventPacket packet, int32_t n)	145
4.8.4.14	caerIMU6EventSetAccelX(caerIMU6Event event, float accelX)	146
4.8.4.15	caerIMU6EventSetAccelY(caerIMU6Event event, float accelY)	146
4.8.4.16	caerIMU6EventSetAccelZ(caerIMU6Event event, float accelZ)	146
4.8.4.17	caerIMU6EventSetGyroX(caerIMU6Event event, float gyroX)	146
4.8.4.18	caerIMU6EventSetGyroY(caerIMU6Event event, float gyroY)	146
4.8.4.19	caerIMU6EventSetGyroZ(caerIMU6Event event, float gyroZ)	147
4.8.4.20	caerIMU6EventSetTemp(caerIMU6Event event, float temp)	147
4.8.4.21	caerIMU6EventSetTimestamp(caerIMU6Event event, int32_t timestamp)	147
4.8.4.22	caerIMU6EventValidate(caerIMU6Event event, caerIMU6EventPacket packet)	147
4.9	events/imu9.h File Reference	147
4.9.1	Detailed Description	149
4.9.2	Macro Definition Documentation	149
4.9.2.1	CAER_IMU9_ITERATOR_ALL_END	149
4.9.2.2	CAER_IMU9_ITERATOR_ALL_START	149
4.9.2.3	CAER_IMU9_ITERATOR_VALID_END	149
4.9.2.4	CAER_IMU9_ITERATOR_VALID_START	150
4.9.3	Typedef Documentation	150
4.9.3.1	caerIMU9Event	150
4.9.3.2	caerIMU9EventPacket	150
4.9.4	Function Documentation	150

4.9.4.1	caerIMU9EventGetAccelX(caerIMU9Event event)	150
4.9.4.2	caerIMU9EventGetAccelY(caerIMU9Event event)	150
4.9.4.3	caerIMU9EventGetAccelZ(caerIMU9Event event)	150
4.9.4.4	caerIMU9EventGetCompX(caerIMU9Event event)	151
4.9.4.5	caerIMU9EventGetCompY(caerIMU9Event event)	151
4.9.4.6	caerIMU9EventGetCompZ(caerIMU9Event event)	151
4.9.4.7	caerIMU9EventGetGyroX(caerIMU9Event event)	151
4.9.4.8	caerIMU9EventGetGyroY(caerIMU9Event event)	152
4.9.4.9	caerIMU9EventGetGyroZ(caerIMU9Event event)	153
4.9.4.10	caerIMU9EventGetTemp(caerIMU9Event event)	153
4.9.4.11	caerIMU9EventGetTimestamp(caerIMU9Event event)	153
4.9.4.12	caerIMU9EventGetTimestamp64(caerIMU9Event event, caerIMU9EventPacket packet)	153
4.9.4.13	caerIMU9EventInvalidate(caerIMU9Event event, caerIMU9EventPacket packet)	154
4.9.4.14	caerIMU9EventIsValid(caerIMU9Event event)	154
4.9.4.15	caerIMU9EventPacketAllocate(int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)	154
4.9.4.16	caerIMU9EventPacketGetEvent(caerIMU9EventPacket packet, int32_t n)	154
4.9.4.17	caerIMU9EventSetAccelX(caerIMU9Event event, float accelX)	155
4.9.4.18	caerIMU9EventSetAccelY(caerIMU9Event event, float accelY)	155
4.9.4.19	caerIMU9EventSetAccelZ(caerIMU9Event event, float accelZ)	155
4.9.4.20	caerIMU9EventSetCompX(caerIMU9Event event, float compX)	155
4.9.4.21	caerIMU9EventSetCompY(caerIMU9Event event, float compY)	155
4.9.4.22	caerIMU9EventSetCompZ(caerIMU9Event event, float compZ)	156
4.9.4.23	caerIMU9EventSetGyroX(caerIMU9Event event, float gyroX)	156
4.9.4.24	caerIMU9EventSetGyroY(caerIMU9Event event, float gyroY)	156
4.9.4.25	caerIMU9EventSetGyroZ(caerIMU9Event event, float gyroZ)	156
4.9.4.26	caerIMU9EventSetTemp(caerIMU9Event event, float temp)	156
4.9.4.27	caerIMU9EventSetTimestamp(caerIMU9Event event, int32_t timestamp)	156
4.9.4.28	caerIMU9EventValidate(caerIMU9Event event, caerIMU9EventPacket packet)	157
4.10	events/packetContainer.h File Reference	157
4.10.1	Detailed Description	158
4.10.2	Typedef Documentation	158
4.10.2.1	caerEventPacketContainer	158
4.10.3	Function Documentation	158
4.10.3.1	caerEventPacketContainerAllocate(int32_t eventPacketsNumber)	158
4.10.3.2	caerEventPacketContainerFree(caerEventPacketContainer container)	158
4.10.3.3	caerEventPacketContainerGetEventPacket(caerEventPacketContainer container, int32_t n)	158
4.10.3.4	caerEventPacketContainerGetEventPacketsNumber(caerEventPacketContainer container)	158

4.10.3.5	<code>caerEventPacketContainerSetEventPacket(caerEventPacketContainer container, int32_t n, caerEventPacketHeader packetHeader)</code>	159
4.10.3.6	<code>caerEventPacketContainerSetEventPacketsNumber(caerEventPacketContainer container, int32_t eventPacketsNumber)</code>	159
4.11	events/polarity.h File Reference	159
4.11.1	Detailed Description	160
4.11.2	Macro Definition Documentation	160
4.11.2.1	<code>CAER_POLARITY_ITERATOR_ALL_END</code>	160
4.11.2.2	<code>CAER_POLARITY_ITERATOR_ALL_START</code>	161
4.11.2.3	<code>CAER_POLARITY_ITERATOR_VALID_END</code>	161
4.11.2.4	<code>CAER_POLARITY_ITERATOR_VALID_START</code>	161
4.11.2.5	<code>POLARITY_MASK</code>	161
4.11.2.6	<code>POLARITY_SHIFT</code>	161
4.11.2.7	<code>X_ADDR_MASK</code>	161
4.11.2.8	<code>X_ADDR_SHIFT</code>	161
4.11.2.9	<code>Y_ADDR_MASK</code>	162
4.11.2.10	<code>Y_ADDR_SHIFT</code>	162
4.11.3	Typedef Documentation	162
4.11.3.1	<code>caerPolarityEvent</code>	162
4.11.3.2	<code>caerPolarityEventPacket</code>	162
4.11.4	Function Documentation	162
4.11.4.1	<code>caerPolarityEventGetPolarity(caerPolarityEvent event)</code>	162
4.11.4.2	<code>caerPolarityEventGetTimestamp(caerPolarityEvent event)</code>	162
4.11.4.3	<code>caerPolarityEventGetTimestamp64(caerPolarityEvent event, caerPolarityEventPacket packet)</code>	162
4.11.4.4	<code>caerPolarityEventGetX(caerPolarityEvent event)</code>	163
4.11.4.5	<code>caerPolarityEventGetY(caerPolarityEvent event)</code>	163
4.11.4.6	<code>caerPolarityEventInvalidate(caerPolarityEvent event, caerPolarityEventPacket packet)</code>	163
4.11.4.7	<code>caerPolarityEventIsValid(caerPolarityEvent event)</code>	163
4.11.4.8	<code>caerPolarityEventPacketAllocate(int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)</code>	164
4.11.4.9	<code>caerPolarityEventPacketGetEvent(caerPolarityEventPacket packet, int32_t n)</code>	165
4.11.4.10	<code>caerPolarityEventSetPolarity(caerPolarityEvent event, bool polarity)</code>	165
4.11.4.11	<code>caerPolarityEventSetTimestamp(caerPolarityEvent event, int32_t timestamp)</code>	165
4.11.4.12	<code>caerPolarityEventSetX(caerPolarityEvent event, uint16_t xAddress)</code>	165
4.11.4.13	<code>caerPolarityEventSetY(caerPolarityEvent event, uint16_t yAddress)</code>	165
4.11.4.14	<code>caerPolarityEventValidate(caerPolarityEvent event, caerPolarityEventPacket packet)</code>	166
4.12	events/sample.h File Reference	166
4.12.1	Detailed Description	167
4.12.2	Macro Definition Documentation	167

4.12.2.1	CAER_SAMPLE_ITERATOR_ALL_END	167
4.12.2.2	CAER_SAMPLE_ITERATOR_ALL_START	167
4.12.2.3	CAER_SAMPLE_ITERATOR_VALID_END	167
4.12.2.4	CAER_SAMPLE_ITERATOR_VALID_START	168
4.12.2.5	SAMPLE_MASK	168
4.12.2.6	SAMPLE_SHIFT	168
4.12.2.7	SAMPLE_TYPE_MASK	168
4.12.2.8	SAMPLE_TYPE_SHIFT	168
4.12.3	Typedef Documentation	168
4.12.3.1	caerSampleEvent	168
4.12.3.2	caerSampleEventPacket	168
4.12.4	Function Documentation	168
4.12.4.1	caerSampleEventGetSample(caerSampleEvent event)	169
4.12.4.2	caerSampleEventGetTimestamp(caerSampleEvent event)	170
4.12.4.3	caerSampleEventGetTimestamp64(caerSampleEvent event, caerSampleEventPacket packet)	170
4.12.4.4	caerSampleEventGetType(caerSampleEvent event)	170
4.12.4.5	caerSampleEventInvalidate(caerSampleEvent event, caerSampleEventPacket packet)	170
4.12.4.6	caerSampleEventIsValid(caerSampleEvent event)	171
4.12.4.7	caerSampleEventPacketAllocate(int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)	171
4.12.4.8	caerSampleEventPacketGetEvent(caerSampleEventPacket packet, int32_t n)	171
4.12.4.9	caerSampleEventSetSample(caerSampleEvent event, uint32_t sample)	171
4.12.4.10	caerSampleEventSetTimestamp(caerSampleEvent event, int32_t timestamp)	172
4.12.4.11	caerSampleEventSetType(caerSampleEvent event, uint8_t type)	172
4.12.4.12	caerSampleEventValidate(caerSampleEvent event, caerSampleEventPacket packet)	172
4.13	events/special.h File Reference	172
4.13.1	Detailed Description	173
4.13.2	Macro Definition Documentation	174
4.13.2.1	CAER_SPECIAL_ITERATOR_ALL_END	174
4.13.2.2	CAER_SPECIAL_ITERATOR_ALL_START	174
4.13.2.3	CAER_SPECIAL_ITERATOR_VALID_END	174
4.13.2.4	CAER_SPECIAL_ITERATOR_VALID_START	174
4.13.2.5	DATA_MASK	174
4.13.2.6	DATA_SHIFT	174
4.13.2.7	TYPE_MASK	174
4.13.2.8	TYPE_SHIFT	175
4.13.3	Typedef Documentation	175
4.13.3.1	caerSpecialEvent	175

4.13.3.2	<code>caerSpecialEventPacket</code>	175
4.13.4	Enumeration Type Documentation	175
4.13.4.1	<code>caer_special_event_types</code>	175
4.13.5	Function Documentation	175
4.13.5.1	<code>caerSpecialEventGetData(caerSpecialEvent event)</code>	175
4.13.5.2	<code>caerSpecialEventGetTimestamp(caerSpecialEvent event)</code>	175
4.13.5.3	<code>caerSpecialEventGetTimestamp64(caerSpecialEvent event, caerSpecialEvent↔ Packet packet)</code>	176
4.13.5.4	<code>caerSpecialEventGetType(caerSpecialEvent event)</code>	176
4.13.5.5	<code>caerSpecialEventInvalidate(caerSpecialEvent event, caerSpecialEventPacket packet)</code>	176
4.13.5.6	<code>caerSpecialEventIsValid(caerSpecialEvent event)</code>	176
4.13.5.7	<code>caerSpecialEventPacketAllocate(int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)</code>	177
4.13.5.8	<code>caerSpecialEventPacketGetEvent(caerSpecialEventPacket packet, int32_t n)</code>	177
4.13.5.9	<code>caerSpecialEventSetData(caerSpecialEvent event, uint32_t data)</code>	177
4.13.5.10	<code>caerSpecialEventSetTimestamp(caerSpecialEvent event, int32_t timestamp)</code>	177
4.13.5.11	<code>caerSpecialEventSetType(caerSpecialEvent event, uint8_t type)</code>	178
4.13.5.12	<code>caerSpecialEventValidate(caerSpecialEvent event, caerSpecialEventPacket packet)</code>	178
4.14	<code>libcaer.h</code> File Reference	178
4.14.1	Detailed Description	179
4.14.2	Macro Definition Documentation	179
4.14.2.1	<code>CLEAR_NUMBITS16</code>	179
4.14.2.2	<code>CLEAR_NUMBITS32</code>	179
4.14.2.3	<code>CLEAR_NUMBITS8</code>	179
4.14.2.4	<code>GET_NUMBITS16</code>	179
4.14.2.5	<code>GET_NUMBITS32</code>	179
4.14.2.6	<code>GET_NUMBITS8</code>	180
4.14.2.7	<code>I16T</code>	180
4.14.2.8	<code>I32T</code>	180
4.14.2.9	<code>I64T</code>	180
4.14.2.10	<code>I8T</code>	180
4.14.2.11	<code>MASK_NUMBITS32</code>	180
4.14.2.12	<code>MASK_NUMBITS64</code>	180
4.14.2.13	<code>SET_NUMBITS16</code>	180
4.14.2.14	<code>SET_NUMBITS32</code>	180
4.14.2.15	<code>SET_NUMBITS8</code>	180
4.14.2.16	<code>SWAP_VAR</code>	180
4.14.2.17	<code>U16T</code>	180
4.14.2.18	<code>U32T</code>	181
4.14.2.19	<code>U64T</code>	181

4.14.2.20 U8T	181
4.14.3 Function Documentation	181
4.14.3.1 caerByteArrayToInteger(uint8_t *byteArray, uint8_t byteArrayLength)	181
4.14.3.2 caerIntegerToByteArray(uint32_t integer, uint8_t *byteArray, uint8_t byteArray↵ Length)	181
4.14.3.3 caerStrEquals(const char *s1, const char *s2)	181
4.14.3.4 caerStrEqualsUpTo(const char *s1, const char *s2, size_t len)	182
4.15 log.h File Reference	183
4.15.1 Detailed Description	183
4.15.2 Macro Definition Documentation	183
4.15.2.1 CAER_LOG_ALERT	183
4.15.2.2 CAER_LOG_CRITICAL	183
4.15.2.3 CAER_LOG_DEBUG	184
4.15.2.4 CAER_LOG_EMERGENCY	184
4.15.2.5 CAER_LOG_ERROR	184
4.15.2.6 CAER_LOG_INFO	184
4.15.2.7 CAER_LOG_NOTICE	184
4.15.2.8 CAER_LOG_WARNING	184
4.15.3 Function Documentation	184
4.15.3.1 caerLog(uint8_t logLevel, const char *subSystem, const char *format,...) __↵ attribute__((format(printf	184
4.15.3.2 caerLogFileDescriptorsSet(int fd1, int fd2)	185
4.15.3.3 caerLogLevelGet(void)	185
4.15.3.4 caerLogLevelSet(uint8_t logLevel)	185
4.16 portable_endian.h File Reference	185
4.16.1 Detailed Description	185
Index	187

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

caer_bias_coarsefine	5
caer_bias_shiftedsources	5
caer_bias_vdac	6
caer_configuration_event	6
caer_configuration_event_packet	7
caer_davis_info	7
caer_dvs128_info	8
caer_ear_event	9
caer_ear_event_packet	9
caer_event_packet_container	10
caer_event_packet_header	10
caer_frame_event	11
caer_frame_event_packet	12
caer_imu6_event	12
caer_imu6_event_packet	13
caer_imu9_event	13
caer_imu9_event_packet	14
caer_polarity_event	14
caer_polarity_event_packet	15
caer_sample_event	15
caer_sample_event_packet	16
caer_special_event	16
caer_special_event_packet	17

Chapter 2

File Index

2.1 File List

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

libcaer.h	178
log.h	183
portable_endian.h	185
devices/davis.h	19
devices/dvs128.h	92
devices/usb.h	95
events/common.h	101
events/config.h	110
events/ear.h	116
events/frame.h	123
events/imu6.h	140
events/imu9.h	147
events/packetContainer.h	157
events/polarity.h	159
events/sample.h	166
events/special.h	172

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 '<http://inilabs.com/support/biasing/>' for more details.

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

- devices/[davis.h](#)

3.2 caer_bias_shiftedsources Struct Reference

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

Data Fields

- uint8_t [refValue](#)

- `uint8_t regValue`
Shifted-source bias level, from 0 to 63.
- `enum caer_bias_shiftedsource_operating_mode operatingMode`
Shifted-source bias current for buffer amplifier, from 0 to 63.
- `enum caer_bias_shiftedsource_operating_mode operatingMode`
Shifted-source operating mode (see 'enum caer_bias_shiftedsource_operating_mode').
- `enum caer_bias_shiftedsource_voltage_level voltageLevel`
Shifted-source voltage level (see 'enum caer_bias_shiftedsource_voltage_level').

3.2.1 Detailed Description

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

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

- `devices/davis.h`

3.3 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.

3.3.1 Detailed Description

On-chip voltage digital-to-analog converter configuration. See '<http://inilabs.com/support/biasing/>' for more details.

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

- `devices/davis.h`

3.4 caer_configuration_event Struct Reference

```
#include <config.h>
```

Data Fields

- `uint8_t moduleAddress`
Configuration module address. First (also) because of valid mark.
- `uint8_t parameterAddress`
Configuration parameter address.
- `uint32_t parameter`
Configuration parameter content (4 bytes).
- `int32_t timestamp`
Event timestamp.

3.4.1 Detailed Description

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.

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

- [events/config.h](#)

3.5 caer_configuration_event_packet Struct Reference

```
#include <config.h>
```

Data Fields

- struct [caer_event_packet_header](#) packetHeader
The common event packet header.
- struct [caer_configuration_event](#) events []
The events array.

3.5.1 Detailed Description

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.

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

- [events/config.h](#)

3.6 caer_davis_info Struct Reference

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

Data Fields

- int16_t [deviceId](#)
Unique device identifier. Also 'source' for events.
- char * [deviceString](#)
Device information string, for logging purposes.
- 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 [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.
- 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.

3.6.1 Detailed Description

DAVIS device-related information.

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

- [devices/davis.h](#)

3.7 caer_dvs128_info Struct Reference

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

Data Fields

- int16_t [deviceId](#)
Unique device identifier. Also 'source' for events.
- char * [deviceString](#)
Device information string, for logging purposes.
- 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.

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

- `devices/dvs128.h`

3.8 caer_ear_event Struct Reference

```
#include <ear.h>
```

Data Fields

- `uint32_t data`
Event data. First because of valid mark.
- `int32_t timestamp`
Event timestamp.

3.8.1 Detailed Description

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.

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

- `events/ear.h`

3.9 caer_ear_event_packet Struct Reference

```
#include <ear.h>
```

Data Fields

- struct `caer_event_packet_header` `packetHeader`
The common event packet header.
- struct `caer_ear_event` `events []`
The events array.

3.9.1 Detailed Description

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.

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

- [events/ear.h](#)

3.10 caer_event_packet_container Struct Reference

```
#include <packetContainer.h>
```

Data Fields

- [int32_t eventPacketsNumber](#)
Number of different event packets contained.
- [caerEventPacketHeader eventPackets \[\]](#)
Array of pointers to the actual event packets.

3.10.1 Detailed Description

EventPacketContainer data structure definition. Signed integers are used for compatibility with languages that do not have unsigned ones, such as Java.

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

- [events/packetContainer.h](#)

3.11 caer_event_packet_header Struct Reference

```
#include <common.h>
```

Data Fields

- [int16_t eventType](#)
Numerical type ID, unique to each event type (see 'enum caer_default_event_types').
- [int16_t eventSource](#)
Numerical source ID, unique inside a process, identifies who generated the events.
- [int32_t eventSize](#)
Size of one event in bytes.
- [int32_t eventTSOffset](#)
Offset from the start of an event, in bytes, at which the main 32 bit time-stamp can be found.
- [int32_t eventTSOverflow](#)
Overflow counter for the standard 32bit event time-stamp. Used to generate the 64 bit time-stamp.
- [int32_t eventCapacity](#)
Maximum number of events this packet can store.
- [int32_t eventNumber](#)
Total number of events present in this packet (valid + invalid).
- [int32_t eventValid](#)
Total number of valid events present in this packet.

3.11.1 Detailed Description

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.

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

- [events/common.h](#)

3.12 caer_frame_event Struct Reference

```
#include <frame.h>
```

Data Fields

- [uint32_t info](#)
Event information (ROI region, color channels, color filter). First because of valid mark.
- [int32_t ts_startframe](#)
Start of Frame (SOF) timestamp.
- [int32_t ts_endframe](#)
End of Frame (EOF) timestamp.
- [int32_t ts_startexposure](#)
Start of Exposure (SOE) timestamp.
- [int32_t ts_endexposure](#)
End of Exposure (EOE) timestamp.
- [int32_t lengthX](#)
X axis length in pixels.
- [int32_t lengthY](#)
Y axis length in pixels.
- [int32_t positionX](#)
X axis position (lower left offset) in pixels.
- [int32_t positionY](#)
Y axis position (lower left offset) in pixels.
- [uint16_t pixels\[\]](#)

3.12.1 Detailed Description

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 lower left corner, like in OpenGL. The pixel array is laid out row by row (increasing X axis), going from bottom to top (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.

3.12.2 Field Documentation

3.12.2.1 [uint16_t caer_frame_event::pixels\[\]](#)

Pixel array, 16 bit unsigned integers, normalized to 16 bit depth. The pixel array is laid out row by row (increasing X axis), going from bottom to top (increasing Y axis).

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

- [events/frame.h](#)

3.13 caer_frame_event_packet Struct Reference

```
#include <frame.h>
```

Data Fields

- struct [caer_event_packet_header](#) packetHeader
The common event packet header.

3.13.1 Detailed Description

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.

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

- [events/frame.h](#)

3.14 caer_imu6_event Struct Reference

```
#include <imu6.h>
```

Data Fields

- uint32_t [info](#)
Event information. First because of valid mark.
- int32_t [timestamp](#)
Event timestamp.
- float [accel_x](#)
Acceleration in the X axis, measured in g (9.81m/s²).
- float [accel_y](#)
Acceleration in the Y axis, measured in g (9.81m/s²).
- float [accel_z](#)
Acceleration in the Z axis, measured in g (9.81m/s²).
- float [gyro_x](#)
Rotation in the X axis, measured in °s.
- float [gyro_y](#)
Rotation in the Y axis, measured in °s.
- float [gyro_z](#)
Rotation in the Z axis, measured in °s.
- float [temp](#)
Temperature, measured in °C.

3.14.1 Detailed Description

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.

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

- [events/imu6.h](#)

3.15 caer_imu6_event_packet Struct Reference

```
#include <imu6.h>
```

Data Fields

- struct [caer_event_packet_header](#) packetHeader
The common event packet header.
- struct [caer_imu6_event](#) events []
The events array.

3.15.1 Detailed Description

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.

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

- [events/imu6.h](#)

3.16 caer_imu9_event Struct Reference

```
#include <imu9.h>
```

Data Fields

- uint32_t [info](#)
Event information. First because of valid mark.
- int32_t [timestamp](#)
Event timestamp.
- float [accel_x](#)
Acceleration in the X axis, measured in g (9.81m/s²).
- float [accel_y](#)
Acceleration in the Y axis, measured in g (9.81m/s²).
- float [accel_z](#)
Acceleration in the Z axis, measured in g (9.81m/s²).
- float [gyro_x](#)
Rotation in the X axis, measured in °s.
- float [gyro_y](#)

Rotation in the Y axis, measured in %s.

- float [gyro_z](#)

Rotation in the Z axis, measured in %s.

- float [temp](#)

Temperature, measured in °C.

- float [comp_x](#)

Magnetometer X axis, measured in μT (magnetic flux density).

- float [comp_y](#)

Magnetometer Y axis, measured in μT (magnetic flux density).

- float [comp_z](#)

Magnetometer Z axis, measured in μT (magnetic flux density).

3.16.1 Detailed Description

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.

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

- [events/imu9.h](#)

3.17 caer_imu9_event_packet Struct Reference

```
#include <imu9.h>
```

Data Fields

- struct [caer_event_packet_header](#) packetHeader

The common event packet header.

- struct [caer_imu9_event](#) events []

The events array.

3.17.1 Detailed Description

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.

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

- [events/imu9.h](#)

3.18 caer_polarity_event Struct Reference

```
#include <polarity.h>
```

Data Fields

- uint32_t [data](#)
Event data. First because of valid mark.
- int32_t [timestamp](#)
Event timestamp.

3.18.1 Detailed Description

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 lower left corner of the screen, like in OpenGL. 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.

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

- [events/polarity.h](#)

3.19 caer_polarity_event_packet Struct Reference

```
#include <polarity.h>
```

Data Fields

- struct [caer_event_packet_header](#) [packetHeader](#)
The common event packet header.
- struct [caer_polarity_event](#) [events](#) []
The events array.

3.19.1 Detailed Description

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.

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

- [events/polarity.h](#)

3.20 caer_sample_event Struct Reference

```
#include <sample.h>
```

Data Fields

- uint32_t [data](#)
Event data. First because of valid mark.
- int32_t [timestamp](#)
Event timestamp.

3.20.1 Detailed Description

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.

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

- [events/sample.h](#)

3.21 caer_sample_event_packet Struct Reference

```
#include <sample.h>
```

Data Fields

- struct [caer_event_packet_header](#) packetHeader
The common event packet header.
- struct [caer_sample_event](#) events []
The events array.

3.21.1 Detailed Description

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.

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

- [events/sample.h](#)

3.22 caer_special_event Struct Reference

```
#include <special.h>
```

Data Fields

- uint32_t data
Event data. First because of valid mark.
- int32_t timestamp
Event timestamp.

3.22.1 Detailed Description

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.

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

- [events/special.h](#)

3.23 caer_special_event_packet Struct Reference

```
#include <special.h>
```

Data Fields

- struct [caer_event_packet_header](#) packetHeader
The common event packet header.
- struct [caer_special_event](#) events []
The events array.

3.23.1 Detailed Description

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.

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

- events/[special.h](#)

Chapter 4

File Documentation

4.1 devices/davis.h File Reference

```
#include "usb.h"
#include "../events/polarity.h"
#include "../events/special.h"
#include "../events/frame.h"
#include "../events/imu6.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 [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_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_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_DELTAT 30`
- `#define DAVIS_CONFIG_DVS_HAS_TEST_EVENT_GENERATOR 31`
- `#define DAVIS_CONFIG_DVS_TEST_EVENT_GENERATOR_ENABLE 32`
- `#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_EXTERNAL_ADC` 32
- `#define DAVIS_CONFIG_APS_HAS_INTERNAL_ADC` 33
- `#define DAVIS_CONFIG_APS_USE_INTERNAL_ADC` 34
- `#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 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_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_DIGITAL_LOW_PASS_FILTER` 7
- `#define DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE` 8
- `#define DAVIS_CONFIG_IMU_GYRO_FULL_SCALE` 9
- `#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_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 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_TYPCALIBNEURON](#) 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_APSEVERFLOWLEVEL](#) 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_PRSEFBP](#) 15
 - #define [DAVIS208_CONFIG_BIAS_REFRBP](#) 16
 - #define [DAVIS208_CONFIG_BIAS_READOUTBUFBP](#) 17
 - #define [DAVIS208_CONFIG_BIAS_APSROSEFBP](#) 18
 - #define [DAVIS208_CONFIG_BIAS_ADCCOMPBP](#) 19
 - #define [DAVIS208_CONFIG_BIAS_COLSELOWBN](#) 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_TYPCALIBNEURON](#) 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_SELECTPREMPAVG](#) 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_APSROSFBN](#) 18
 - #define [DAVIS346_CONFIG_BIAS_ADCCOMPBP](#) 19
 - #define [DAVIS346_CONFIG_BIAS_COLSELLOWBN](#) 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` (struct `caer_bias_vdac` `vdacBias`)
- struct `caer_bias_vdac` `caerBiasVDACParse` (`uint16_t` `vdacBias`)
- `uint16_t` `caerBiasCoarseFineGenerate` (struct `caer_bias_coarsefine` `coarseFineBias`)
- struct `caer_bias_coarsefine` `caerBiasCoarseFineParse` (`uint16_t` `coarseFineBias`)
- `uint16_t` `caerBiasShiftedSourceGenerate` (struct `caer_bias_shiftedsource` `shiftedSourceBias`)
- struct `caer_bias_shiftedsource` `caerBiasShiftedSourceParse` (`uint16_t` `shiftedSourceBias`)

4.1.1 Detailed Description

DAVIS specific configuration defines and information structures.

4.1.2 Macro Definition Documentation

4.1.2.1 `#define CAER_DEVICE_DAVIS_FX2 1`

Device type definition for iniLabs DAVIS FX2-based boards, like DAVIS240a/b/c.

4.1.2.2 `#define CAER_DEVICE_DAVIS_FX3 2`

Device type definition for iniLabs DAVIS FX3-based boards, like DAVIS640.

4.1.2.3 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.4 `#define DAVIS128_CONFIG_BIAS_ADCREFHIGH 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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.5 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.6 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.7 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.8 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.9 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.10 #define DAVIS128_CONFIG_BIAS_APSOEVERFLOWLEVEL 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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.11 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.12 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.13 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.14 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.15 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.16 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.17 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.18 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.19 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.20 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.21 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.22 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.23 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.24 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.25 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.26 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.27 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.28 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.29 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.30 #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.

4.1.2.31 #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.

4.1.2.32 #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.

4.1.2.33 `#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.

4.1.2.34 `#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.

4.1.2.35 `#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.

4.1.2.36 `#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.

4.1.2.37 `#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.

4.1.2.38 `#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.

4.1.2.39 `#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.

4.1.2.40 `#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.

4.1.2.41 `#define DAVIS128_CONFIG_CHIP_RESETTESTPIXEL` 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.

4.1.2.42 #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.

4.1.2.43 #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.

4.1.2.44 #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.

4.1.2.45 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.46 #define DAVIS208_CONFIG_BIAS_ADCREFHIGH 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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.47 #define DAVIS208_CONFIG_BIAS_ADCREFLOW 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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.48 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.49 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.50 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.51 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.52 `#define DAVIS208_CONFIG_BIAS_APSOEVERFLOWLEVEL 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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.53 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.54 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.55 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.56 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.57 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.58 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.59 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.60 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.61 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.62 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.63 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.64 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.65 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.66 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.67 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.68 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.69 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.70 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.71 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.72 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.73 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.74 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.75 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.76 `#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.

4.1.2.77 `#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.

4.1.2.78 `#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.

4.1.2.79 #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.

4.1.2.80 #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.

4.1.2.81 #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.

4.1.2.82 #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.

4.1.2.83 #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.

4.1.2.84 #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.

4.1.2.85 #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.

4.1.2.86 #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.

4.1.2.87 #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.

4.1.2.88 #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.

4.1.2.89 #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.

4.1.2.90 #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.

4.1.2.91 #define DAVIS208_CONFIG_CHIP_SELECTPOSFB 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.

4.1.2.92 #define DAVIS208_CONFIG_CHIP_SELECTPREAMPAVG 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.

4.1.2.93 #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.

4.1.2.94 #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.

4.1.2.95 #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.

4.1.2.96 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.97 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.98 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.99 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.100 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.101 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.102 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.103 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.104 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.105 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.106 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.107 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.108 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.109 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.110 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.111 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.112 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.113 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.114 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.115 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.116 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.117 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.118 `#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.

4.1.2.119 `#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.

4.1.2.120 `#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.

4.1.2.121 `#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.

4.1.2.122 `#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.

4.1.2.123 #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.

4.1.2.124 #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.

4.1.2.125 #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.

4.1.2.126 #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.

4.1.2.127 #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.

4.1.2.128 #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.

4.1.2.129 #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.

4.1.2.130 #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.

4.1.2.131 `#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.

4.1.2.132 `#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.

4.1.2.133 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.134 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.135 `#define DAVIS346_CONFIG_BIAS_ADCCREFLOW 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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.136 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.137 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.138 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.139 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.140 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.141 `#define DAVIS346_CONFIG_BIAS_APSOEVERFLOWLEVEL 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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.142 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.143 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.144 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.145 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.146 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.147 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.148 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.149 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.150 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.151 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.152 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.153 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.154 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.155 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.156 #define DAVIS346_CONFIG_BIAS_PRFBP 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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.157 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.158 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.159 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.160 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.161 #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.

4.1.2.162 #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.

4.1.2.163 #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.

4.1.2.164 #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.

4.1.2.165 #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.

4.1.2.166 #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.

4.1.2.167 #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.

4.1.2.168 #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.

4.1.2.169 #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.

4.1.2.170 `#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.

4.1.2.171 `#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.

4.1.2.172 `#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.

4.1.2.173 `#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.

4.1.2.174 `#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.

4.1.2.175 `#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.

4.1.2.176 `#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.

4.1.2.177 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.178 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.179 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.180 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.181 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.182 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.183 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.184 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.185 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.186 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.187 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.188 `#define DAVIS640_CONFIG_BIAS_COLSELOWBN 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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.189 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.190 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.191 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.192 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.193 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.194 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.195 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.196 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.197 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.198 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.199 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.200 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.201 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.202 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.203 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.204 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.205 #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.

4.1.2.206 #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.

4.1.2.207 #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.

4.1.2.208 #define DAVIS640_CONFIG_CHIP_ANALGMUX2 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.

4.1.2.209 #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.

4.1.2.210 #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.

4.1.2.211 #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.

4.1.2.212 #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.

4.1.2.213 #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.

4.1.2.214 #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.

4.1.2.215 #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.

4.1.2.216 #define DAVIS640_CONFIG_CHIP_RESETTESTPIXEL 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.

4.1.2.217 `#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.

4.1.2.218 `#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.

4.1.2.219 `#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.

4.1.2.220 `#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.

4.1.2.221 `#define DAVIS_CHIP_DAVIS128 3`

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

4.1.2.222 `#define DAVIS_CHIP_DAVIS208 8`

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

4.1.2.223 `#define DAVIS_CHIP_DAVIS240A 0`

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

4.1.2.224 `#define DAVIS_CHIP_DAVIS240B 1`

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

4.1.2.225 `#define DAVIS_CHIP_DAVIS240C 2`

DAVIS240C chip identifier. 240x180, no color.

4.1.2.226 `#define DAVIS_CHIP_DAVIS346A 4`

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

4.1.2.227 #define DAVIS_CHIP_DAVIS346B 5

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

4.1.2.228 #define DAVIS_CHIP_DAVIS346C 9

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

4.1.2.229 #define DAVIS_CHIP_DAVIS640 6

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

4.1.2.230 #define DAVIS_CHIP_DAVISRGB 7

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

4.1.2.231 #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.

4.1.2.232 #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.

4.1.2.233 #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.

4.1.2.234 #define DAVIS_CONFIG_APS_COLUMN_SETTLE 16

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

4.1.2.235 #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.

4.1.2.236 #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.

4.1.2.237 #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.

4.1.2.238 #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.

4.1.2.239 #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.

4.1.2.240 #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.

4.1.2.241 #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.

4.1.2.242 #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.

4.1.2.243 #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.

4.1.2.244 #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.

4.1.2.245 #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).

4.1.2.246 #define DAVIS_CONFIG_APS_HAS_EXTERNAL_ADC 32

Parameter address for module DAVIS_CONFIG_APS: read-only parameter, information about the presence of an external 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.

4.1.2.247 #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.

4.1.2.248 #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.

4.1.2.249 #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.

4.1.2.250 #define DAVIS_CONFIG_APS_NULL_SETTLE 18

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

4.1.2.251 #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.

4.1.2.252 #define DAVIS_CONFIG_APS_RAMP_RESET 37

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

4.1.2.253 #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.

4.1.2.254 #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.

4.1.2.255 #define DAVIS_CONFIG_APS_RESET_SETTLE 15

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

4.1.2.256 #define DAVIS_CONFIG_APS_ROW_SETTLE 17

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

4.1.2.257 #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.

4.1.2.258 #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.

4.1.2.259 #define DAVIS_CONFIG_APS_SAMPLE_SETTLE 36

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

4.1.2.260 #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.

4.1.2.261 #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.

4.1.2.262 #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 than 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.

4.1.2.263 #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 for the ROI region to be enabled. Setting it to APS_SIZE_X itself deactivates this ROI region completely.

4.1.2.264 #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 for the ROI region to be enabled. Setting it to APS_SIZE_X itself deactivates this ROI region completely.

4.1.2.265 #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 for the ROI region to be enabled. Setting it to APS_SIZE_X itself deactivates this ROI region completely.

4.1.2.266 #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 for the ROI region to be enabled. Setting it to APS_SIZE_X itself deactivates this ROI region completely.

4.1.2.267 #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.

4.1.2.268 #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.

4.1.2.269 #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.

4.1.2.270 #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.

4.1.2.271 #define DAVIS_CONFIG_APS_USE_INTERNAL_ADC 34

Parameter address for module DAVIS_CONFIG_APS: use the internal, on-chip ADC instead of the external one. This enables a much faster and more power-efficient readout for the frames, and should as such always be preferred.

4.1.2.272 #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.

4.1.2.273 #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.

4.1.2.274 #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.

4.1.2.275 #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.

4.1.2.276 #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.

4.1.2.277 #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.

4.1.2.278 #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.

4.1.2.279 #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.

4.1.2.280 #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.

4.1.2.281 #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.

4.1.2.282 #define DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_DELTAT 30

Parameter address for module DAVIS_CONFIG_DVS: 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.

4.1.2.283 #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.

4.1.2.284 #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.

4.1.2.285 #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.

4.1.2.286 #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.

4.1.2.287 #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.

4.1.2.288 #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.

4.1.2.289 #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.

4.1.2.290 #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.

4.1.2.291 #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.

4.1.2.292 #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.

4.1.2.293 #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.

4.1.2.294 #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.

4.1.2.295 #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.

4.1.2.296 #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.

4.1.2.297 #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.

4.1.2.298 #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.

4.1.2.299 #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!

4.1.2.300 #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.

4.1.2.301 #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.

4.1.2.302 `#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.

4.1.2.303 `#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 0: `dvsInvert` ↔ XY 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.

4.1.2.304 `#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.

4.1.2.305 `#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.

4.1.2.306 `#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.

4.1.2.307 `#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.

4.1.2.308 `#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 AE ↔ R handshake with the chip and wait until it's free again, instead of just continuing the handshake and dropping the resulting events.

4.1.2.309 `#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.

4.1.2.310 `#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).

4.1.2.311 `#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).

4.1.2.312 `#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.

4.1.2.313 `#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.

4.1.2.314 `#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).

4.1.2.315 `#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`.

4.1.2.316 `#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`.

4.1.2.317 `#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.

4.1.2.318 `#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.

4.1.2.319 `#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.

4.1.2.320 `#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.

4.1.2.321 `#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.

4.1.2.322 `#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.

4.1.2.323 `#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

4.1.2.324 `#define DAVIS_CONFIG_IMU_ACCEL_STANDBY 2`

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

4.1.2.325 `#define DAVIS_CONFIG_IMU_DIGITAL_LOW_PASS_FILTER 7`

Parameter address for module `DAVIS_CONFIG_IMU`: this configures the digital low-pass filter for both the accelerometer and the gyroscope. Valid values are from 0 to 7 and have the following meaning: 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

4.1.2.326 #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

4.1.2.327 #define DAVIS_CONFIG_IMU_GYRO_STANDBY 3

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

4.1.2.328 #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.

4.1.2.329 #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

4.1.2.330 #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.

4.1.2.331 #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.

4.1.2.332 #define DAVIS_CONFIG_IMU_TEMP_STANDBY 1

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

4.1.2.333 #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.

4.1.2.334 #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.

4.1.2.335 #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.

4.1.2.336 #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.

4.1.2.337 #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.

4.1.2.338 #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.

4.1.2.339 #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.

4.1.2.340 #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.

4.1.2.341 #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.

4.1.2.342 #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.

4.1.2.343 #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.

4.1.2.344 #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.

4.1.2.345 #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.

4.1.2.346 #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.

4.1.2.347 #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. It usually represents a specific SVN revision, at which the logic code was synthesized. 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.

4.1.2.348 #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.

4.1.2.349 #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).

4.1.2.350 #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!

4.1.2.351 #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.

4.1.2.352 `#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.

4.1.2.353 `#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.

4.1.2.354 `#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.

4.1.2.355 `#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.

4.1.2.356 `#define DAVISRGB_CONFIG_APS_TRANSFER 50`

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

4.1.2.357 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.358 `#define DAVISRGB_CONFIG_BIAS_ADCCREFHIGH 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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.359 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.360 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.361 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.362 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.363 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.364 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.365 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.366 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.367 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.368 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.369 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.370 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.371 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.372 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.373 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.374 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.375 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.376 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.377 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.378 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.379 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.380 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.381 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.382 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.383 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.384 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.385 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.386 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.387 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.388 `#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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.389 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.390 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.1.2.391 #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.

4.1.2.392 #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.

4.1.2.393 #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.

4.1.2.394 #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.

4.1.2.395 #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.

4.1.2.396 #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.

4.1.2.397 #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.

4.1.2.398 #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.

4.1.2.399 #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.

4.1.2.400 #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.

4.1.2.401 #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.

4.1.2.402 #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.

4.1.2.403 #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.

4.1.2.404 #define DAVISRGB_CONFIG_CHIP_RESETTESTPIXEL 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.

4.1.2.405 `#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.

4.1.2.406 `#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.

4.1.2.407 `#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.

4.1.2.408 `#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.

4.1.2.409 `#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.

4.1.2.410 `#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.

4.1.2.411 `#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.

4.1.2.412 `#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.

4.1.2.413 `#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.

4.1.2.414 `#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.

4.1.2.415 `#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.

4.1.2.416 `#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.

4.1.2.417 `#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.

4.1.2.418 `#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.

4.1.2.419 `#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.

4.1.2.420 `#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 `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 `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 uint16_t caerBiasCoarseFineGenerate (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 struct caer_bias_coarsefine caerBiasCoarseFineParse (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 uint16_t caerBiasShiftedSourceGenerate (struct caer_bias_shiftedsourceshiftedSourceBias)

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

Parameters

<i>shiftedSource↔ Bias</i>	shifted-source bias structure.
--------------------------------	--------------------------------

Returns

internal integer representation for device configuration.

4.1.4.4 struct caer_bias_shiftedsourceshiftedSourceParse (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>shiftedSource↔ Bias</i>	internal integer representation from device.
--------------------------------	--

Returns

shifted-source bias structure.

4.1.4.5 uint16_t caerBiasVDACGenerate (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 struct caer_bias_vdac caerBiasVDACParse (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 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.

4.2 devices/dvs128.h File Reference

```
#include "usb.h"
#include "../events/polarity.h"
#include "../events/special.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)

4.2.1 Detailed Description

DVS128 specific configuration defines and information structures.

4.2.2 Macro Definition Documentation

4.2.2.1 #define CAER_DEVICE_DVS128 0

Device type definition for iniLabs DVS128.

4.2.2.2 #define DVS128_CONFIG_BIAS 1

Module address: device-side chip bias generator configuration.

4.2.2.3 #define DVS128_CONFIG_BIAS_CAS 0

Parameter address for module DVS128_CONFIG_BIAS: First stage amplifier cascode bias. See '<http://inilabs.com/support/biasing/>' for more details.

4.2.2.4 #define DVS128_CONFIG_BIAS_DIFF 9

Parameter address for module DVS128_CONFIG_BIAS: Differential (second stage amplifier) bias. See '<http://inilabs.com/support/biasing/>' for more details.

4.2.2.5 #define DVS128_CONFIG_BIAS_DIFFOFF 4

Parameter address for module DVS128_CONFIG_BIAS: Off events threshold bias. See '<http://inilabs.com/support/biasing/>' for more details.

4.2.2.6 #define DVS128_CONFIG_BIAS_DIFFON 8

Parameter address for module DVS128_CONFIG_BIAS: On events threshold bias. See '<http://inilabs.com/support/biasing/>' for more details.

4.2.2.7 #define DVS128_CONFIG_BIAS_FOLL 10

Parameter address for module DVS128_CONFIG_BIAS: Source follower bias. See '<http://inilabs.com/support/biasing/>' for more details.

4.2.2.8 #define DVS128_CONFIG_BIAS_INJGND 1

Parameter address for module DVS128_CONFIG_BIAS: Injected ground bias. See '<http://inilabs.com/support/biasing/>' for more details.

4.2.2.9 #define DVS128_CONFIG_BIAS_PR 11

Parameter address for module DVS128_CONFIG_BIAS: Photoreceptor bias. See '<http://inilabs.com/support/biasing/>' for more details.

4.2.2.10 #define DVS128_CONFIG_BIAS_PUX 3

Parameter address for module DVS128_CONFIG_BIAS: Pull up on request from X arbiter (AER). See '<http://inilabs.com/support/biasing/>' for more details.

4.2.2.11 #define DVS128_CONFIG_BIAS_PUY 7

Parameter address for module DVS128_CONFIG_BIAS: Pull up on request from Y arbiter (AER). See '<http://inilabs.com/support/biasing/>' for more details.

4.2.2.12 #define DVS128_CONFIG_BIAS_REFR 6

Parameter address for module DVS128_CONFIG_BIAS: Refractory period bias. See '<http://inilabs.com/support/biasing/>' for more details.

4.2.2.13 #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 '<http://inilabs.com/support/biasing/>' for more details.

4.2.2.14 #define DVS128_CONFIG_BIAS_REQPD 2

Parameter address for module DVS128_CONFIG_BIAS: Pull down on chip request (AER). See '<http://inilabs.com/support/biasing/>' for more details.

4.2.2.15 #define DVS128_CONFIG_DVS 0

Module address: device-side DVS configuration.

4.2.2.16 #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.

4.2.2.17 #define DVS128_CONFIG_DVS_RUN 0

Parameter address for module DVS128_CONFIG_DVS: run the DVS chip and generate polarity event data.

4.2.2.18 #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.

4.2.2.19 #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.

4.2.3 Function Documentation**4.2.3.1 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.

4.3 devices/usb.h File Reference

```
#include "../libcaer.h"
#include "../events/packetContainer.h"
```

Macros

- `#define CAER_HOST_CONFIG_USB -1`
- `#define CAER_HOST_CONFIG_DATAEXCHANGE -2`
- `#define CAER_HOST_CONFIG_PACKETS -3`
- `#define CAER_HOST_CONFIG_USB_BUFFER_NUMBER 0`
- `#define CAER_HOST_CONFIG_USB_BUFFER_SIZE 1`
- `#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_SIZE 0`
- `#define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL 1`
- `#define CAER_HOST_CONFIG_PACKETS_MAX_POLARITY_SIZE 2`
- `#define CAER_HOST_CONFIG_PACKETS_MAX_POLARITY_INTERVAL 3`
- `#define CAER_HOST_CONFIG_PACKETS_MAX_SPECIAL_SIZE 4`
- `#define CAER_HOST_CONFIG_PACKETS_MAX_SPECIAL_INTERVAL 5`
- `#define CAER_HOST_CONFIG_PACKETS_MAX_FRAME_SIZE 6`
- `#define CAER_HOST_CONFIG_PACKETS_MAX_FRAME_INTERVAL 7`
- `#define CAER_HOST_CONFIG_PACKETS_MAX_IMU6_SIZE 8`
- `#define CAER_HOST_CONFIG_PACKETS_MAX_IMU6_INTERVAL 9`

Typedefs

- `typedef struct caer_device_handle * caerDeviceHandle`

Functions

- `caerDeviceHandle caerDeviceOpen` (uint16_t deviceId, uint16_t deviceType, uint8_t busNumberRestrict, uint8_t devAddressRestrict, const char *serialNumberRestrict)
- `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 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)

4.3.1 Detailed Description

Common functions to access, configure and exchange data with supported USB devices. Also contains defines for host/USB related configuration options.

4.3.2 Macro Definition Documentation

4.3.2.1 `#define CAER_HOST_CONFIG_DATAEXCHANGE -2`

Module address: host-side data exchange (ring-buffer) configuration.

4.3.2.2 `#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.

4.3.2.3 `#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 USB 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.

4.3.2.4 `#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 USB data transfer thread with `caer↵DeviceDataStart()` 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.

4.3.2.5 `#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 USB data transfer thread with `caer↵DeviceDataStop()` 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.

4.3.2.6 `#define CAER_HOST_CONFIG_PACKETS -3`

Module address: host-side event packets generation configuration.

4.3.2.7 `#define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL 1`

Parameter address for module `CAER_HOST_CONFIG_PACKETS`: set the maximum interval between the earliest and the latest event in a packet container before it's made available to the user via `caerDeviceDataGet()`. The value is in microseconds, and is checked across all types of events contained in the `EventPacketContainer`.

4.3.2.8 `#define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_SIZE 0`

Parameter address for module `CAER_HOST_CONFIG_PACKETS`: set the maximum number of events a packet container may hold before it's made available to the user via `caerDeviceDataGet()`. This is a sum of the number of events held in each typed `EventPacket` that is a part of the `EventPacketContainer`.

4.3.2.9 `#define CAER_HOST_CONFIG_PACKETS_MAX_FRAME_INTERVAL 7`

Parameter address for module `CAER_HOST_CONFIG_PACKETS`: set the maximum interval between the earliest and the latest event in a `FrameEventPacket` before it's made available to the user via `caerDeviceDataGet()`. The value is in microseconds. This will trigger the commit of the full `EventPacketContainer`.

4.3.2.10 `#define CAER_HOST_CONFIG_PACKETS_MAX_FRAME_SIZE 6`

Parameter address for module `CAER_HOST_CONFIG_PACKETS`: set the maximum number of events a `Frame↔EventPacket` may hold before it's made available to the user via `caerDeviceDataGet()`. This will trigger the commit of the full `EventPacketContainer`.

4.3.2.11 `#define CAER_HOST_CONFIG_PACKETS_MAX_IMU6_INTERVAL 9`

Parameter address for module `CAER_HOST_CONFIG_PACKETS`: set the maximum interval between the earliest and the latest event in an `IMU6EventPacket` before it's made available to the user via `caerDeviceDataGet()`. The value is in microseconds. This will trigger the commit of the full `EventPacketContainer`.

4.3.2.12 `#define CAER_HOST_CONFIG_PACKETS_MAX_IMU6_SIZE 8`

Parameter address for module `CAER_HOST_CONFIG_PACKETS`: set the maximum number of events an `IMU6↔EventPacket` may hold before it's made available to the user via `caerDeviceDataGet()`. This will trigger the commit of the full `EventPacketContainer`.

4.3.2.13 `#define CAER_HOST_CONFIG_PACKETS_MAX_POLARITY_INTERVAL 3`

Parameter address for module `CAER_HOST_CONFIG_PACKETS`: set the maximum interval between the earliest and the latest event in a `PolarityEventPacket` before it's made available to the user via `caerDeviceDataGet()`. The value is in microseconds. This will trigger the commit of the full `EventPacketContainer`.

4.3.2.14 `#define CAER_HOST_CONFIG_PACKETS_MAX_POLARITY_SIZE 2`

Parameter address for module `CAER_HOST_CONFIG_PACKETS`: set the maximum number of events a `Polarity↔EventPacket` may hold before it's made available to the user via `caerDeviceDataGet()`. This will trigger the commit of the full `EventPacketContainer`.

4.3.2.15 `#define CAER_HOST_CONFIG_PACKETS_MAX_SPECIAL_INTERVAL 5`

Parameter address for module `CAER_HOST_CONFIG_PACKETS`: set the maximum interval between the earliest and the latest event in a `SpecialEventPacket` before it's made available to the user via `caerDeviceDataGet()`. The value is in microseconds. This will trigger the commit of the full `EventPacketContainer`.

4.3.2.16 `#define CAER_HOST_CONFIG_PACKETS_MAX_SPECIAL_SIZE 4`

Parameter address for module `CAER_HOST_CONFIG_PACKETS`: set the maximum number of events a `Special↔EventPacket` may hold before it's made available to the user via `caerDeviceDataGet()`. This will trigger the commit of the full `EventPacketContainer`.

4.3.2.17 `#define CAER_HOST_CONFIG_USB -1`

Module address: host-side USB configuration.

4.3.2.18 `#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.

4.3.2.19 `#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.

4.3.3 Typedef Documentation

4.3.3.1 `typedef struct caer_device_handle* caerDeviceHandle`

Reference to an open device on which to operate.

4.3.4 Function Documentation

4.3.4.1 `bool caerDeviceClose (caerDeviceHandle * handle)`

Close a previously opened USB 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.

4.3.4.2 `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 sending the configuration was successful, false on errors.

4.3.4.3 `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.

4.3.4.4 `caerEventPacketContainer caerDeviceDataGet (caerDeviceHandle handle)`

Get an event packet container, which contains events of various types generated by the device, from the USB data transfer thread 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, or when there is no container available in non-blocking mode. Always check for this!

4.3.4.5 `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 USB data transfer thread and starting the data producers (see `CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS`). Supports notification of new data and shutdown events via user-defined call-backs.

Parameters

<i>handle</i>	a valid device handle.
<i>dataNotify↔ Increase</i>	function pointer, called every time a new piece of data available and has been put in the FIFO buffer for consumption. <code>dataNotifyUserPtr</code> will be passed as parameter to the function.
<i>dataNotify↔ Decrease</i>	function pointer, called every time a new piece of data has been consumed from the FIFO buffer inside <code>caerDeviceDataGet()</code> . <code>dataNotifyUserPtr</code> will be passed as parameter to the function.
<i>dataNotify↔ UserPtr</i>	pointer that will be passed to the <code>dataNotifyIncrease</code> and <code>dataNotifyDecrease</code> functions. Can be NULL.
<i>dataShutdown↔ Notify</i>	function pointer, called on shut-down of the USB data transfer thread. This can be used to detect exceptional shut-downs that do not come from calling <code>caerDeviceDataStop()</code> , such as when the device is disconnected or all USB transfers fail.
<i>dataShutdown↔ UserPtr</i>	pointer that will be passed to the <code>dataShutdownNotify</code> function. Can be NULL.

Returns

true if starting the data transfer was successful, false on errors.

4.3.4.6 `bool caerDeviceDataStop (caerDeviceHandle handle)`

Stop getting data from the device, shutting down the USB data transfer thread and stopping the data producers (see `CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS`). This normal shut-down will also 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.

4.3.4.7 `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 generate from its data.
<i>deviceType</i>	type of the device to open. Currently supported are: <code>CAER_DEVICE_DVS128</code> , <code>CAER_DEVICE_DAVIS_FX2</code> , <code>CAER_DEVICE_DAVIS_FX3</code>
<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!

4.3.4.8 `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.

4.4 events/common.h File Reference

```
#include "../libcaer.h"
```

Data Structures

- struct [caer_event_packet_header](#)

Macros

- `#define TS_OVERFLOW_SHIFT 31`
- `#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](#)

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 }

Functions

- struct [caer_event_packet_header](#) **__attribute__**((**__packed__**))
- static int16_t [caerEventPacketHeaderGetEventType](#) ([caerEventPacketHeader](#) header)
- static void [caerEventPacketHeaderSetEventType](#) ([caerEventPacketHeader](#) header, int16_t [eventType](#))
- static int16_t [caerEventPacketHeaderGetEventSource](#) ([caerEventPacketHeader](#) header)
- static void [caerEventPacketHeaderSetEventSource](#) ([caerEventPacketHeader](#) header, int16_t [eventSource](#))
- static int32_t [caerEventPacketHeaderGetEventSize](#) ([caerEventPacketHeader](#) header)
- static void [caerEventPacketHeaderSetEventSize](#) ([caerEventPacketHeader](#) header, int32_t [eventSize](#))
- static int32_t [caerEventPacketHeaderGetEventTSOffset](#) ([caerEventPacketHeader](#) header)
- static void [caerEventPacketHeaderSetEventTSOffset](#) ([caerEventPacketHeader](#) header, int32_t [eventTSOffset](#))
- static int32_t [caerEventPacketHeaderGetEventTSOverflow](#) ([caerEventPacketHeader](#) header)
- static void [caerEventPacketHeaderSetEventTSOverflow](#) ([caerEventPacketHeader](#) header, int32_t [eventTSOverflow](#))
- static int32_t [caerEventPacketHeaderGetEventCapacity](#) ([caerEventPacketHeader](#) header)
- static void [caerEventPacketHeaderSetEventCapacity](#) ([caerEventPacketHeader](#) header, int32_t [eventsCapacity](#))
- static int32_t [caerEventPacketHeaderGetEventNumber](#) ([caerEventPacketHeader](#) header)
- static void [caerEventPacketHeaderSetEventNumber](#) ([caerEventPacketHeader](#) header, int32_t [eventsNumber](#))
- static int32_t [caerEventPacketHeaderGetEventValid](#) ([caerEventPacketHeader](#) header)
- static void [caerEventPacketHeaderSetEventValid](#) ([caerEventPacketHeader](#) header, int32_t [eventsValid](#))
- static void * [caerGenericEventGetEvent](#) ([caerEventPacketHeader](#) headerPtr, int32_t n)
- static int32_t [caerGenericEventGetTimestamp](#) (void *eventPtr, [caerEventPacketHeader](#) headerPtr)
- static int64_t [caerGenericEventGetTimestamp64](#) (void *eventPtr, [caerEventPacketHeader](#) headerPtr)
- static bool [caerGenericEventsValid](#) (void *eventPtr)
- static void * [caerCopyEventPacket](#) (void *eventPacket)
- static void * [caerCopyEventPacketOnlyEvents](#) (void *eventPacket)
- static void * [caerCopyEventPacketOnlyValidEvents](#) (void *eventPacket)

Variables

- `int16_t eventType`
Numerical type ID, unique to each event type (see 'enum caer_default_event_types').
- `int16_t eventSource`
Numerical source ID, unique inside a process, identifies who generated the events.
- `int32_t eventSize`
Size of one event in bytes.
- `int32_t eventTSOffset`
Offset from the start of an event, in bytes, at which the main 32 bit time-stamp can be found.
- `int32_t eventTSOverflow`
Overflow counter for the standard 32bit event time-stamp. Used to generate the 64 bit time-stamp.
- `int32_t eventCapacity`
Maximum number of events this packet can store.
- `int32_t eventNumber`
Total number of events present in this packet (valid + invalid).
- `int32_t eventValid`
Total number of valid events present in this packet.

4.4.1 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.

4.4.2 Macro Definition Documentation

4.4.2.1 `#define CAER_EVENT_PACKET_HEADER_SIZE 28`

Size of the EventPacket header. This is constant across all supported systems.

4.4.2.2 `#define CAER_ITERATOR_ALL_END }`

Generic iterator close statement.

4.4.2.3 `#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).

4.4.2.4 `#define CAER_ITERATOR_VALID_END }`

Generic iterator close statement.

4.4.2.5 #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).

4.4.2.6 #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 TSOverflow needs to be shifted by 31 thus when constructing such a timestamp.

4.4.2.7 #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.

4.4.2.8 #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.

4.4.3 Typedef Documentation

4.4.3.1 typedef struct caer_event_packet_header* caerEventPacketHeader

Type for pointer to EventPacket header data structure.

4.4.4 Enumeration Type Documentation

4.4.4.1 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.

4.4.5 Function Documentation

4.4.5.1 `static void* caerCopyEventPacket (void * eventPacket)` `[inline],[static]`

Make a full copy of an event packet (up to eventCapacity).

Parameters

<i>eventPacket</i>	an event packet to copy.
--------------------	--------------------------

Returns

a full copy of an event packet.

4.4.5.2 `static void* caerCopyEventPacketOnlyEvents (void * eventPacket)` `[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>eventPacket</i>	an event packet to copy.
--------------------	--------------------------

Returns

a sized down copy of an event packet.

4.4.5.3 `static void* caerCopyEventPacketOnlyValidEvents (void * eventPacket)` `[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>eventPacket</i>	an event packet to copy.
--------------------	--------------------------

Returns

a copy of an event packet, containing only valid events.

4.4.5.4 `static int32_t caerEventPacketHeaderGetEventCapacity (caerEventPacketHeader 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.

4.4.5.5 `static int32_t caerEventPacketHeaderGetEventNumber (caerEventPacketHeader 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.

4.4.5.6 `static int32_t caerEventPacketHeaderGetEventSize (caerEventPacketHeader 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.

4.4.5.7 `static int16_t caerEventPacketHeaderGetEventSource (caerEventPacketHeader 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.

4.4.5.8 `static int32_t caerEventPacketHeaderGetEventTSOffset (caerEventPacketHeader 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.

4.4.5.9 `static int32_t caerEventPacketHeaderGetEventTSOverflow (caerEventPacketHeader 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.

4.4.5.10 `static int16_t caerEventPacketHeaderGetEventType (caerEventPacketHeader 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').

4.4.5.11 `static int32_t caerEventPacketHeaderGetEventValid (caerEventPacketHeader 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.

4.4.5.12 `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.

4.4.5.13 `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.

4.4.5.14 `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.

4.4.5.15 `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.

4.4.5.16 `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.

4.4.5.17 `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.

4.4.5.18 `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').

4.4.5.19 `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.

4.4.5.20 `static void* caerGenericEventGetEvent (caerEventPacketHeader 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,eventCapacity[bounds.

Returns

a generic pointer to the requested event. NULL on error.

4.4.5.21 `static int32_t caerGenericEventGetTimestamp (void * eventPtr, caerEventPacketHeader 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.

4.4.5.22 `static int64_t caerGenericEventGetTimestamp64 (void * eventPtr, caerEventPacketHeader 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.

4.4.5.23 `static bool caerGenericEventIsValid (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.

4.5 events/config.h File Reference

```
#include "common.h"
```

Data Structures

- struct [caer_configuration_event](#)
- struct [caer_configuration_event_packet](#)

Macros

- `#define` [CAER_CONFIGURATION_ITERATOR_ALL_START](#)(CONFIGURATION_PACKET)
- `#define` [CAER_CONFIGURATION_ITERATOR_ALL_END](#) }
- `#define` [CAER_CONFIGURATION_ITERATOR_VALID_START](#)(CONFIGURATION_PACKET)
- `#define` [CAER_CONFIGURATION_ITERATOR_VALID_END](#) }
- `#define` [MODULE_ADDR_SHIFT](#) 1
- `#define` [MODULE_ADDR_MASK](#) 0x0000007F

Typedefs

- typedef struct [caer_configuration_event](#) * [caerConfigurationEvent](#)
- typedef struct [caer_configuration_event_packet](#) * [caerConfigurationEventPacket](#)

Functions

- struct `caer_configuration_event` `__attribute__((__packed__))`
- `caerConfigurationEventPacket` `caerConfigurationEventPacketAllocate` (`int32_t eventCapacity`, `int16_t eventSource`, `int32_t tsOverflow`)
- static `caerConfigurationEvent` `caerConfigurationEventPacketGetEvent` (`caerConfigurationEventPacket` packet, `int32_t n`)
- static `int32_t` `caerConfigurationEventGetTimestamp` (`caerConfigurationEvent` event)
- static `int64_t` `caerConfigurationEventGetTimestamp64` (`caerConfigurationEvent` event, `caerConfigurationEventPacket` packet)
- static void `caerConfigurationEventSetTimestamp` (`caerConfigurationEvent` event, `int32_t timestamp`)
- static bool `caerConfigurationEventIsValid` (`caerConfigurationEvent` event)
- static void `caerConfigurationEventValidate` (`caerConfigurationEvent` event, `caerConfigurationEventPacket` packet)
- static void `caerConfigurationEventInvalidate` (`caerConfigurationEvent` event, `caerConfigurationEventPacket` packet)
- static `uint8_t` `caerConfigurationEventGetModuleAddress` (`caerConfigurationEvent` event)
- static void `caerConfigurationEventSetModuleAddress` (`caerConfigurationEvent` event, `uint8_t moduleAddress`)
- static `uint8_t` `caerConfigurationEventGetParameterAddress` (`caerConfigurationEvent` event)
- static void `caerConfigurationEventSetParameterAddress` (`caerConfigurationEvent` event, `uint8_t parameterAddress`)
- static `uint32_t` `caerConfigurationEventGetParameter` (`caerConfigurationEvent` event)
- static void `caerConfigurationEventSetParameter` (`caerConfigurationEvent` event, `uint32_t parameter`)

Variables

- `uint8_t moduleAddress`
Configuration module address. First (also) because of valid mark.
- `uint8_t parameterAddress`
Configuration parameter address.
- `uint32_t parameter`
Configuration parameter content (4 bytes).
- `int32_t timestamp`
Event timestamp.
- struct `caer_event_packet_header` `packetHeader`
The common event packet header.
- struct `caer_configuration_event` `events` [`]`
The events array.

4.5.1 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.

4.5.2 Macro Definition Documentation

4.5.2.1 `#define CAER_CONFIGURATION_ITERATOR_ALL_END`

Iterator close statement.

4.5.2.2 #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.

4.5.2.3 #define CAER_CONFIGURATION_ITERATOR_VALID_END }

Iterator close statement.

4.5.2.4 #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.

4.5.2.5 #define 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.

4.5.2.6 #define 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.

4.5.3 Typedef Documentation

4.5.3.1 typedef struct caer_configuration_event* caerConfigurationEvent

Type for pointer to configuration event data structure.

4.5.3.2 typedef struct caer_configuration_event_packet* caerConfigurationEventPacket

Type for pointer to configuration event packet data structure.

4.5.4 Function Documentation

4.5.4.1 static uint8_t caerConfigurationEventGetModuleAddress (caerConfigurationEvent 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.

4.5.4.2 static uint32_t caerConfigurationEventGetParameter (caerConfigurationEvent event) [inline], [static]

Get the configuration event's parameter.

Parameters

<i>event</i>	a valid ConfigurationEvent pointer. Cannot be NULL.
--------------	---

Returns

configuration parameter.

4.5.4.3 static uint8_t caerConfigurationEventGetParameterAddress (caerConfigurationEvent 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.

4.5.4.4 static int32_t caerConfigurationEventGetTimestamp (caerConfigurationEvent 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.

4.5.4.5 `static int64_t caerConfigurationEventGetTimestamp64 (caerConfigurationEvent event, caerConfigurationEventPacket 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.

4.5.4.6 `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.

4.5.4.7 `static bool caerConfigurationEventsValid (caerConfigurationEvent 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.

4.5.4.8 `caerConfigurationEventPacket caerConfigurationEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)`

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.

4.5.4.9 `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.

4.5.4.10 `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.

4.5.4.11 `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.

4.5.4.12 `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>parameter</i> ↔ <i>Address</i>	configuration parameter address.
--------------------------------------	----------------------------------

4.5.4.13 `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.

4.5.4.14 `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.

4.6 events/ear.h File Reference

```
#include "common.h"
```

Data Structures

- struct [caer_ear_event](#)
- struct [caer_ear_event_packet](#)

Macros

- `#define CAER_EAR_ITERATOR_ALL_START(EAR_PACKET)`
- `#define CAER_EAR_ITERATOR_ALL_END }`
- `#define CAER_EAR_ITERATOR_VALID_START(EAR_PACKET)`
- `#define CAER_EAR_ITERATOR_VALID_END }`
- `#define EAR_SHIFT 1`
- `#define EAR_MASK 0x0000000F`
- `#define CHANNEL_SHIFT 5`
- `#define CHANNEL_MASK 0x000007FF`
- `#define NEURON_SHIFT 16`
- `#define NEURON_MASK 0x000000FF`
- `#define FILTER_SHIFT 24`
- `#define FILTER_MASK 0x000000FF`

Typedefs

- typedef struct `caer_ear_event` * `caerEarEvent`
- typedef struct `caer_ear_event_packet` * `caerEarEventPacket`

Functions

- struct `caer_ear_event` **__attribute__** ((__packed__))
- `caerEarEventPacket caerEarEventPacketAllocate` (int32_t `eventCapacity`, int16_t `eventSource`, int32_t `timestamp`, int32_t `tsOverflow`)
- static `caerEarEvent caerEarEventPacketGetEvent` (`caerEarEventPacket` `packet`, int32_t `n`)
- static int32_t `caerEarEventGetTimestamp` (`caerEarEvent` `event`)
- static int64_t `caerEarEventGetTimestamp64` (`caerEarEvent` `event`, `caerEarEventPacket` `packet`)
- static void `caerEarEventSetTimestamp` (`caerEarEvent` `event`, int32_t `timestamp`)
- static bool `caerEarEventsValid` (`caerEarEvent` `event`)
- static void `caerEarEventValidate` (`caerEarEvent` `event`, `caerEarEventPacket` `packet`)
- static void `caerEarEventInvalidate` (`caerEarEvent` `event`, `caerEarEventPacket` `packet`)
- static uint8_t `caerEarEventGetEar` (`caerEarEvent` `event`)
- static void `caerEarEventSetEar` (`caerEarEvent` `event`, uint8_t `ear`)
- static uint16_t `caerEarEventGetChannel` (`caerEarEvent` `event`)
- static void `caerEarEventSetChannel` (`caerEarEvent` `event`, uint16_t `channel`)
- static uint8_t `caerEarEventGetNeuron` (`caerEarEvent` `event`)
- static void `caerEarEventSetNeuron` (`caerEarEvent` `event`, uint8_t `neuron`)
- static uint8_t `caerEarEventGetFilter` (`caerEarEvent` `event`)
- static void `caerEarEventSetFilter` (`caerEarEvent` `event`, uint8_t `filter`)

Variables

- uint32_t `data`
Event data. First because of valid mark.
- int32_t `timestamp`
Event timestamp.
- struct `caer_event_packet_header` `packetHeader`
The common event packet header.
- struct `caer_ear_event` `events` []
The events array.

4.6.1 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.

4.6.2 Macro Definition Documentation

4.6.2.1 #define CAER_EAR_ITERATOR_ALL_END }

Iterator close statement.

4.6.2.2 #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.

4.6.2.3 #define CAER_EAR_ITERATOR_VALID_END }

Iterator close statement.

4.6.2.4 #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.

4.6.2.5 #define 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.

4.6.2.6 #define 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.

4.6.2.7 #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.

4.6.2.8 #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.

4.6.2.9 #define 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.

4.6.2.10 #define 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.

4.6.2.11 #define 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.

4.6.2.12 #define 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.

4.6.3 Typedef Documentation

4.6.3.1 typedef struct caer_ear_event* caerEarEvent

Type for pointer to ear (cochlea) event data structure.

4.6.3.2 typedef struct caer_ear_event_packet* caerEarEventPacket

Type for pointer to ear (cochlea) event packet data structure.

4.6.4 Function Documentation

4.6.4.1 static uint16_t caerEarEventGetChannel (caerEarEvent 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.

4.6.4.2 `static uint8_t caerEarEventGetEar (caerEarEvent 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.

4.6.4.3 static int32_t caerEarEventGetTimestamp (caerEarEvent 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.

4.6.4.4 static int64_t caerEarEventGetTimestamp64 (caerEarEvent event, caerEarEventPacket 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.

4.6.4.5 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.

4.6.4.6 static bool caerEarEventsValid (caerEarEvent 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.

4.6.4.7 `caerEarEventPacket caerEarEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)`

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.

4.6.4.8 `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.

4.6.4.9 `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.

4.6.4.10 `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.

4.6.4.11 `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.

4.6.4.12 `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.

4.7 events/frame.h File Reference

```
#include "common.h"
```

Data Structures

- struct [caer_frame_event](#)
- struct [caer_frame_event_packet](#)

Macros

- `#define` [CAER_FRAME_ITERATOR_ALL_START](#)(FRAME_PACKET)
- `#define` [CAER_FRAME_ITERATOR_ALL_END](#) }
- `#define` [CAER_FRAME_ITERATOR_VALID_START](#)(FRAME_PACKET)
- `#define` [CAER_FRAME_ITERATOR_VALID_END](#) }
- `#define` [COLOR_CHANNELS_SHIFT](#) 1
- `#define` [COLOR_CHANNELS_MASK](#) 0x00000007
- `#define` [COLOR_FILTER_SHIFT](#) 4
- `#define` [COLOR_FILTER_MASK](#) 0x00000007
- `#define` [ROI_IDENTIFIER_SHIFT](#) 7
- `#define` [ROI_IDENTIFIER_MASK](#) 0x0000007F

Typedefs

- typedef struct `caer_frame_event` * `caerFrameEvent`
- typedef struct `caer_frame_event_packet` * `caerFrameEventPacket`

Enumerations

- enum `caer_frame_event_color_channels` { `GRAYSCALE` = 1, `RGB` = 3, `RGBA` = 4 }
- enum `caer_frame_event_color_filter` { `MONO` = 0, `RGBG` = 1, `RGBW` = 2 }

Functions

- struct `caer_frame_event` **__attribute__((packed))**
- `caerFrameEventPacket` `caerFrameEventPacketAllocate` (int32_t `eventCapacity`, int16_t `eventSource`, int32_t `tsOverflow`, int32_t `maxLengthX`, int32_t `maxLengthY`, int16_t `maxChannelNumber`)
- static `caerFrameEvent` `caerFrameEventPacketGetEvent` (`caerFrameEventPacket` `packet`, int32_t `n`)
- static int32_t `caerFrameEventGetTSStartOfFrame` (`caerFrameEvent` `event`)
- static int64_t `caerFrameEventGetTSStartOfFrame64` (`caerFrameEvent` `event`, `caerFrameEventPacket` `packet`)
- static void `caerFrameEventSetTSStartOfFrame` (`caerFrameEvent` `event`, int32_t `startFrame`)
- static int32_t `caerFrameEventGetTSEndOfFrame` (`caerFrameEvent` `event`)
- static int64_t `caerFrameEventGetTSEndOfFrame64` (`caerFrameEvent` `event`, `caerFrameEventPacket` `packet`)
- static void `caerFrameEventSetTSEndOfFrame` (`caerFrameEvent` `event`, int32_t `endFrame`)
- static int32_t `caerFrameEventGetTSStartOfExposure` (`caerFrameEvent` `event`)
- static int64_t `caerFrameEventGetTSStartOfExposure64` (`caerFrameEvent` `event`, `caerFrameEventPacket` `packet`)
- static void `caerFrameEventSetTSStartOfExposure` (`caerFrameEvent` `event`, int32_t `startExposure`)
- static int32_t `caerFrameEventGetTSEndOfExposure` (`caerFrameEvent` `event`)
- static int64_t `caerFrameEventGetTSEndOfExposure64` (`caerFrameEvent` `event`, `caerFrameEventPacket` `packet`)
- static void `caerFrameEventSetTSEndOfExposure` (`caerFrameEvent` `event`, int32_t `endExposure`)
- static int32_t `caerFrameEventGetExposureLength` (`caerFrameEvent` `event`)
- static int32_t `caerFrameEventGetTimestamp` (`caerFrameEvent` `event`)
- static int64_t `caerFrameEventGetTimestamp64` (`caerFrameEvent` `event`, `caerFrameEventPacket` `packet`)
- static bool `caerFrameEventsValid` (`caerFrameEvent` `event`)
- static void `caerFrameEventValidate` (`caerFrameEvent` `event`, `caerFrameEventPacket` `packet`)
- static void `caerFrameEventInvalidate` (`caerFrameEvent` `event`, `caerFrameEventPacket` `packet`)
- static size_t `caerFrameEventPacketGetPixelsSize` (`caerFrameEventPacket` `packet`)
- static size_t `caerFrameEventPacketGetPixelsMaxIndex` (`caerFrameEventPacket` `packet`)
- static uint8_t `caerFrameEventGetROIIdentifier` (`caerFrameEvent` `event`)
- static void `caerFrameEventSetROIIdentifier` (`caerFrameEvent` `event`, uint8_t `roiIdentifier`)
- static enum `caer_frame_event_color_filter` `caerFrameEventGetColorFilter` (`caerFrameEvent` `event`)
- static void `caerFrameEventSetColorFilter` (`caerFrameEvent` `event`, enum `caer_frame_event_color_filter` `colorFilter`)
- static int32_t `caerFrameEventGetLengthX` (`caerFrameEvent` `event`)
- static int32_t `caerFrameEventGetLengthY` (`caerFrameEvent` `event`)
- static enum `caer_frame_event_color_channels` `caerFrameEventGetChannelNumber` (`caerFrameEvent` `event`)
- static void `caerFrameEventSetLengthXLengthYChannelNumber` (`caerFrameEvent` `event`, int32_t `lengthX`, int32_t `lengthY`, enum `caer_frame_event_color_channels` `channelNumber`, `caerFrameEventPacket` `packet`)
- static size_t `caerFrameEventGetPixelsMaxIndex` (`caerFrameEvent` `event`)
- static size_t `caerFrameEventGetPixelsSize` (`caerFrameEvent` `event`)
- static int32_t `caerFrameEventGetPositionX` (`caerFrameEvent` `event`)
- static void `caerFrameEventSetPositionX` (`caerFrameEvent` `event`, int32_t `positionX`)
- static int32_t `caerFrameEventGetPositionY` (`caerFrameEvent` `event`)

- static void `caerFrameEventSetPositionY` (`caerFrameEvent` event, `int32_t` `positionY`)
- static `uint16_t` `caerFrameEventGetPixel` (`caerFrameEvent` 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` (`caerFrameEvent` 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` (`caerFrameEvent` 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` (`caerFrameEvent` 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 `uint16_t` * `caerFrameEventGetPixelArrayCGFormat` (`caerFrameEvent` event)

Variables

- enum `caer_frame_event_color_channels` `__attribute__((__packed__))`
- `uint32_t` `info`
Event information (ROI region, color channels, color filter). First because of valid mark.
- `int32_t` `ts_startframe`
Start of Frame (SOF) timestamp.
- `int32_t` `ts_endframe`
End of Frame (EOF) timestamp.
- `int32_t` `ts_startexposure`
Start of Exposure (SOE) timestamp.
- `int32_t` `ts_endexposure`
End of Exposure (EOE) timestamp.
- `int32_t` `lengthX`
X axis length in pixels.
- `int32_t` `lengthY`
Y axis length in pixels.
- `int32_t` `positionX`
X axis position (lower left offset) in pixels.
- `int32_t` `positionY`
Y axis position (lower left offset) in pixels.
- `uint16_t` `pixels` []
- struct `caer_event_packet_header` `packetHeader`
The common event packet header.

4.7.1 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 lower left corner of the screen, like in OpenGL. The pixel array is laid out row by row (increasing X axis), going from bottom to top (increasing Y axis).

4.7.2 Macro Definition Documentation

4.7.2.1 `#define CAER_FRAME_ITERATOR_ALL_END`

Iterator close statement.

4.7.2.2 `#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.

4.7.2.3 `#define CAER_FRAME_ITERATOR_VALID_END`

Iterator close statement.

4.7.2.4 `#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.

4.7.2.5 `#define 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.

4.7.2.6 `#define 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.

4.7.2.7 #define COLOR_FILTER_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.

4.7.2.8 #define 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.

4.7.2.9 #define 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.

4.7.2.10 #define ROI_IDENTIFIER_SHIFT 7

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.

4.7.3 Typedef Documentation

4.7.3.1 typedef struct caer_frame_event* caerFrameEvent

Type for pointer to frame event data structure.

4.7.3.2 typedef struct caer_frame_event_packet* caerFrameEventPacket

Type for pointer to frame event packet data structure.

4.7.4 Enumeration Type Documentation

4.7.4.1 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.

4.7.4.2 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.

RGBW Modified Bayer color filter, with white (pass all light) instead of extra green.

4.7.5 Function Documentation

4.7.5.1 static enum caer_frame_event_color_channels caerFrameEventGetChannelNumber (caerFrameEvent 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.

4.7.5.2 static enum caer_frame_event_color_filter caerFrameEventGetColorFilter (caerFrameEvent 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.

4.7.5.3 static int32_t caerFrameEventGetExposureLength (caerFrameEvent 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.

4.7.5.4 static int32_t caerFrameEventGetLengthX (caerFrameEvent 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.

4.7.5.5 `static int32_t caerFrameEventGetLengthY (caerFrameEvent 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.

4.7.5.6 `static uint16_t caerFrameEventGetPixel (caerFrameEvent 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 lower left corner, like in OpenGL.

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).

4.7.5.7 `static uint16_t* caerFrameEventGetPixelArrayCGFormat (caerFrameEvent event)` `[inline],[static]`

Get a direct reference to a copy of the pixels array, rotated in such a way that the first pixel is not at the bottom left as usual (OpenGL format), but at the top left (Computer Graphics format). 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 (decreasing Y axis). Please remember to free this new pixels array after usage!

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

the reformatted pixels array (16 bit integers are little-endian) or NULL on error. Remember to free the pixels array after usage.

4.7.5.8 `static uint16_t* caerFrameEventGetPixelArrayUnsafe (caerFrameEvent event)` `[inline],[static]`

Get a direct reference 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 bottom to top (increasing Y axis).

Parameters

<i>event</i>	a valid FrameEvent pointer. Cannot be NULL.
--------------	---

Returns

the pixels array (16 bit integers are little-endian).

4.7.5.9 `static uint16_t caerFrameEventGetPixelForChannel (caerFrameEvent 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 lower left corner, like in OpenGL.

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).

4.7.5.10 `static uint16_t caerFrameEventGetPixelForChannelUnsafe (caerFrameEvent 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 lower left corner, like in OpenGL.

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).

4.7.5.11 `static size_t caerFrameEventGetPixelsMaxIndex (caerFrameEvent 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.

4.7.5.12 `static size_t caerFrameEventGetPixelsSize (caerFrameEvent 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.

4.7.5.13 `static uint16_t caerFrameEventGetPixelUnsafe (caerFrameEvent 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 lower left corner, like in OpenGL.

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).

4.7.5.14 `static int32_t caerFrameEventGetPositionX (caerFrameEvent 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.

4.7.5.15 `static int32_t caerFrameEventGetPositionY (caerFrameEvent 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.

4.7.5.16 `static uint8_t caerFrameEventGetROIIdentifier (caerFrameEvent 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.

4.7.5.17 `static int32_t caerFrameEventGetTimestamp (caerFrameEvent 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.

4.7.5.18 `static int64_t caerFrameEventGetTimestamp64 (caerFrameEvent event, caerFrameEventPacket 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.

4.7.5.19 `static int32_t caerFrameEventGetTSEndOfExposure (caerFrameEvent 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.

4.7.5.20 `static int64_t caerFrameEventGetTSEndOfExposure64 (caerFrameEvent event, caerFrameEventPacket 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.

4.7.5.21 `static int32_t caerFrameEventGetTSEndOfFrame (caerFrameEvent 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.

4.7.5.22 `static int64_t caerFrameEventGetTSEndOfFrame64 (caerFrameEvent event, caerFrameEventPacket 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.

4.7.5.23 `static int32_t caerFrameEventGetTSStartOfExposure (caerFrameEvent 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.

4.7.5.24 `static int64_t caerFrameEventGetTSStartOfExposure64 (caerFrameEvent event, caerFrameEventPacket 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.

4.7.5.25 `static int32_t caerFrameEventGetTSStartOfFrame (caerFrameEvent 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.

4.7.5.26 `static int64_t caerFrameEventGetTSStartOfFrame64 (caerFrameEvent event, caerFrameEventPacket 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.

4.7.5.27 `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.

4.7.5.28 `static bool caerFrameEventsValid (caerFrameEvent 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.

4.7.5.29 `caerFrameEventPacket caerFrameEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow, int32_t maxLengthX, int32_t maxLengthY, int16_t maxChannelNumber)`

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.

4.7.5.30 `static caerFrameEvent caerFrameEventPacketGetEvent (caerFrameEventPacket packet, int32_t n)`
[inline], [static]

Get the frame event at the given index from the event packet.

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.

4.7.5.31 `static size_t caerFrameEventPacketGetPixelsMaxIndex (caerFrameEventPacket packet)` [inline], [static]

Get the maximum index into the pixels array, based upon how much memory was allocated to it by '[caerFrameEventPacketAllocate\(\)](#)'.

Parameters

<i>packet</i>	a valid FrameEventPacket pointer. Cannot be NULL.
---------------	---

Returns

maximum pixels array index.

4.7.5.32 `static size_t caerFrameEventPacketGetPixelsSize (caerFrameEventPacket packet)` `[inline],[static]`

Get the maximum size of the pixels array in bytes, based upon how much memory was allocated to it by '[caerFrameEventPacketAllocate\(\)](#)'.

Parameters

<i>packet</i>	a valid FrameEventPacket pointer. Cannot be NULL.
---------------	---

Returns

maximum pixels array size in bytes.

4.7.5.33 `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.

4.7.5.34 `static void caerFrameEventSetLengthXLengthYChannelNumber (caerFrameEvent event, int32_t lengthX, int32_t lengthY, enum caer_frame_event_color_channels channelNumber, caerFrameEventPacket 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.

4.7.5.35 `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 lower left corner, like in OpenGL.

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).

4.7.5.36 `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 lower left corner, like in OpenGL.

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).

4.7.5.37 `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 lower left corner, like in OpenGL.

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).

4.7.5.38 `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 lower left corner, like in OpenGL.

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).

4.7.5.39 `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.

4.7.5.40 `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.

4.7.5.41 `static void caerFrameEventSetROIIdentifier (caerFrameEvent event, uint8_t roIdentifier)` `[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>roIdentifier</i>	numerical ROI identifier.

4.7.5.42 `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.

4.7.5.43 `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.

4.7.5.44 `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.

4.7.5.45 `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.

4.7.5.46 `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.

4.7.6 Variable Documentation

4.7.6.1 `uint16_t pixels[]`

Pixel array, 16 bit unsigned integers, normalized to 16 bit depth. The pixel array is laid out row by row (increasing X axis), going from bottom to top (increasing Y axis).

4.8 events/imu6.h File Reference

```
#include "common.h"
```

Data Structures

- struct [caer_imu6_event](#)
- struct [caer_imu6_event_packet](#)

Macros

- `#define CAER_IMU6_ITERATOR_ALL_START(IMU6_PACKET)`
- `#define CAER_IMU6_ITERATOR_ALL_END }`
- `#define CAER_IMU6_ITERATOR_VALID_START(IMU6_PACKET)`
- `#define CAER_IMU6_ITERATOR_VALID_END }`

Typedefs

- typedef struct [caer_imu6_event](#) * [caerIMU6Event](#)
- typedef struct [caer_imu6_event_packet](#) * [caerIMU6EventPacket](#)

Functions

- struct [caer_imu6_event](#) **__attribute__((packed))**
- [caerIMU6EventPacket](#) [caerIMU6EventPacketAllocate](#) (int32_t [eventCapacity](#), int16_t [eventSource](#), int32_t [tsOverflow](#))
- static [caerIMU6Event](#) [caerIMU6EventPacketGetEvent](#) ([caerIMU6EventPacket](#) packet, int32_t n)
- static int32_t [caerIMU6EventGetTimestamp](#) ([caerIMU6Event](#) event)
- static int64_t [caerIMU6EventGetTimestamp64](#) ([caerIMU6Event](#) event, [caerIMU6EventPacket](#) packet)
- static void [caerIMU6EventSetTimestamp](#) ([caerIMU6Event](#) event, int32_t [timestamp](#))
- static bool [caerIMU6EventIsValid](#) ([caerIMU6Event](#) event)
- static void [caerIMU6EventValidate](#) ([caerIMU6Event](#) event, [caerIMU6EventPacket](#) packet)
- static void [caerIMU6EventInvalidate](#) ([caerIMU6Event](#) event, [caerIMU6EventPacket](#) packet)
- static float [caerIMU6EventGetAccelX](#) ([caerIMU6Event](#) event)
- static void [caerIMU6EventSetAccelX](#) ([caerIMU6Event](#) event, float [accelX](#))
- static float [caerIMU6EventGetAccelY](#) ([caerIMU6Event](#) event)
- static void [caerIMU6EventSetAccelY](#) ([caerIMU6Event](#) event, float [accelY](#))
- static float [caerIMU6EventGetAccelZ](#) ([caerIMU6Event](#) event)
- static void [caerIMU6EventSetAccelZ](#) ([caerIMU6Event](#) event, float [accelZ](#))
- static float [caerIMU6EventGetGyroX](#) ([caerIMU6Event](#) event)
- static void [caerIMU6EventSetGyroX](#) ([caerIMU6Event](#) event, float [gyroX](#))
- static float [caerIMU6EventGetGyroY](#) ([caerIMU6Event](#) event)
- static void [caerIMU6EventSetGyroY](#) ([caerIMU6Event](#) event, float [gyroY](#))
- static float [caerIMU6EventGetGyroZ](#) ([caerIMU6Event](#) event)
- static void [caerIMU6EventSetGyroZ](#) ([caerIMU6Event](#) event, float [gyroZ](#))
- static float [caerIMU6EventGetTemp](#) ([caerIMU6Event](#) event)
- static void [caerIMU6EventSetTemp](#) ([caerIMU6Event](#) event, float [temp](#))

Variables

- uint32_t [info](#)
Event information. First because of valid mark.
- int32_t [timestamp](#)
Event timestamp.
- float [accel_x](#)
Acceleration in the X axis, measured in g (9.81m/s²).
- float [accel_y](#)
Acceleration in the Y axis, measured in g (9.81m/s²).
- float [accel_z](#)
Acceleration in the Z axis, measured in g (9.81m/s²).
- float [gyro_x](#)
Rotation in the X axis, measured in °s.
- float [gyro_y](#)
Rotation in the Y axis, measured in °s.
- float [gyro_z](#)
Rotation in the Z axis, measured in °s.
- float [temp](#)
Temperature, measured in °C.

- struct `caer_event_packet_header` `packetHeader`
The common event packet header.
- struct `caer_imu6_event` `events` []
The events array.

4.8.1 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.

4.8.2 Macro Definition Documentation

4.8.2.1 `#define CAER_IMU6_ITERATOR_ALL_END`

Iterator close statement.

4.8.2.2 `#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.

4.8.2.3 `#define CAER_IMU6_ITERATOR_VALID_END`

Iterator close statement.

4.8.2.4 `#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.

4.8.3 Typedef Documentation

4.8.3.1 `typedef struct caer_imu6_event* caerIMU6Event`

Type for pointer to IMU 6-axes event data structure.

4.8.3.2 typedef struct caer_imu6_event_packet* caerIMU6EventPacket

Type for pointer to IMU 6-axes event packet data structure.

4.8.4 Function Documentation

4.8.4.1 static float caerIMU6EventGetAccelX (caerIMU6Event 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.

4.8.4.2 static float caerIMU6EventGetAccelY (caerIMU6Event 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.

4.8.4.3 static float caerIMU6EventGetAccelZ (caerIMU6Event 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.

4.8.4.4 static float caerIMU6EventGetGyroX (caerIMU6Event 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).

4.8.4.5 static float caerIMU6EventGetGyroY (caerIMU6Event 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).

4.8.4.6 `static float caerIMU6EventGetGyroZ (caerIMU6Event 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).

4.8.4.7 `static float caerIMU6EventGetTemp (caerIMU6Event 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.

4.8.4.8 `static int32_t caerIMU6EventGetTimestamp (caerIMU6Event 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.

4.8.4.9 `static int64_t caerIMU6EventGetTimestamp64 (caerIMU6Event event, caerIMU6EventPacket 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.

4.8.4.10 `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.

4.8.4.11 `static bool caerIMU6EventsValid (caerIMU6Event 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.

4.8.4.12 `caerIMU6EventPacket caerIMU6EventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)`

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.

4.8.4.13 `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.

4.8.4.14 `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.

4.8.4.15 `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.

4.8.4.16 `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.

4.8.4.17 `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).

4.8.4.18 `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).

4.8.4.19 `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).

4.8.4.20 `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.

4.8.4.21 `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.

4.8.4.22 `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.

4.9 events/imu9.h File Reference

```
#include "common.h"
```

Data Structures

- struct [caer_imu9_event](#)
- struct [caer_imu9_event_packet](#)

Macros

- #define `CAER_IMU9_ITERATOR_ALL_START`(IMU9_PACKET)
- #define `CAER_IMU9_ITERATOR_ALL_END` }
- #define `CAER_IMU9_ITERATOR_VALID_START`(IMU9_PACKET)
- #define `CAER_IMU9_ITERATOR_VALID_END` }

Typedefs

- typedef struct `caer_imu9_event` * `caerIMU9Event`
- typedef struct `caer_imu9_event_packet` * `caerIMU9EventPacket`

Functions

- struct `caer_imu9_event` **__attribute__** ((__packed__))
- `caerIMU9EventPacket` `caerIMU9EventPacketAllocate` (int32_t `eventCapacity`, int16_t `eventSource`, int32_t `tsOverflow`)
- static `caerIMU9Event` `caerIMU9EventPacketGetEvent` (`caerIMU9EventPacket` `packet`, int32_t `n`)
- static int32_t `caerIMU9EventGetTimestamp` (`caerIMU9Event` `event`)
- static int64_t `caerIMU9EventGetTimestamp64` (`caerIMU9Event` `event`, `caerIMU9EventPacket` `packet`)
- static void `caerIMU9EventSetTimestamp` (`caerIMU9Event` `event`, int32_t `timestamp`)
- static bool `caerIMU9EventIsValid` (`caerIMU9Event` `event`)
- static void `caerIMU9EventValidate` (`caerIMU9Event` `event`, `caerIMU9EventPacket` `packet`)
- static void `caerIMU9EventInvalidate` (`caerIMU9Event` `event`, `caerIMU9EventPacket` `packet`)
- static float `caerIMU9EventGetAccelX` (`caerIMU9Event` `event`)
- static void `caerIMU9EventSetAccelX` (`caerIMU9Event` `event`, float `accelX`)
- static float `caerIMU9EventGetAccelY` (`caerIMU9Event` `event`)
- static void `caerIMU9EventSetAccelY` (`caerIMU9Event` `event`, float `accelY`)
- static float `caerIMU9EventGetAccelZ` (`caerIMU9Event` `event`)
- static void `caerIMU9EventSetAccelZ` (`caerIMU9Event` `event`, float `accelZ`)
- static float `caerIMU9EventGetGyroX` (`caerIMU9Event` `event`)
- static void `caerIMU9EventSetGyroX` (`caerIMU9Event` `event`, float `gyroX`)
- static float `caerIMU9EventGetGyroY` (`caerIMU9Event` `event`)
- static void `caerIMU9EventSetGyroY` (`caerIMU9Event` `event`, float `gyroY`)
- static float `caerIMU9EventGetGyroZ` (`caerIMU9Event` `event`)
- static void `caerIMU9EventSetGyroZ` (`caerIMU9Event` `event`, float `gyroZ`)
- static float `caerIMU9EventGetTemp` (`caerIMU9Event` `event`)
- static void `caerIMU9EventSetTemp` (`caerIMU9Event` `event`, float `temp`)
- static float `caerIMU9EventGetCompX` (`caerIMU9Event` `event`)
- static void `caerIMU9EventSetCompX` (`caerIMU9Event` `event`, float `compX`)
- static float `caerIMU9EventGetCompY` (`caerIMU9Event` `event`)
- static void `caerIMU9EventSetCompY` (`caerIMU9Event` `event`, float `compY`)
- static float `caerIMU9EventGetCompZ` (`caerIMU9Event` `event`)
- static void `caerIMU9EventSetCompZ` (`caerIMU9Event` `event`, float `compZ`)

Variables

- uint32_t `info`
Event information. First because of valid mark.
- int32_t `timestamp`
Event timestamp.
- float `accel_x`
Acceleration in the X axis, measured in g (9.81m/s²).

- float [accel_y](#)
Acceleration in the Y axis, measured in g (9.81m/s²).
- float [accel_z](#)
Acceleration in the Z axis, measured in g (9.81m/s²).
- float [gyro_x](#)
Rotation in the X axis, measured in °/s.
- float [gyro_y](#)
Rotation in the Y axis, measured in °/s.
- float [gyro_z](#)
Rotation in the Z axis, measured in °/s.
- float [temp](#)
Temperature, measured in °C.
- float [comp_x](#)
Magnetometer X axis, measured in μT (magnetic flux density).
- float [comp_y](#)
Magnetometer Y axis, measured in μT (magnetic flux density).
- float [comp_z](#)
Magnetometer Z axis, measured in μT (magnetic flux density).
- struct [caer_event_packet_header](#) packetHeader
The common event packet header.
- struct [caer_imu9_event](#) events []
The events array.

4.9.1 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.

4.9.2 Macro Definition Documentation

4.9.2.1 #define CAER_IMU9_ITERATOR_ALL_END }

Iterator close statement.

4.9.2.2 #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.

4.9.2.3 #define CAER_IMU9_ITERATOR_VALID_END }

Iterator close statement.

4.9.2.4 #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.

4.9.3 Typedef Documentation

4.9.3.1 typedef struct caer_imu9_event* caerIMU9Event

Type for pointer to IMU 9-axes event data structure.

4.9.3.2 typedef struct caer_imu9_event_packet* caerIMU9EventPacket

Type for pointer to IMU 9-axes event packet data structure.

4.9.4 Function Documentation

4.9.4.1 static float caerIMU9EventGetAccelX (caerIMU9Event 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.

4.9.4.2 static float caerIMU9EventGetAccelY (caerIMU9Event 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.

4.9.4.3 static float caerIMU9EventGetAccelZ (caerIMU9Event 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.

4.9.4.4 `static float caerIMU9EventGetCompX (caerIMU9Event event) [inline],[static]`

Get the X axis compass heading (from magnetometer). This is in μT .

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

X axis compass heading.

4.9.4.5 `static float caerIMU9EventGetCompY (caerIMU9Event event) [inline],[static]`

Get the Y axis compass heading (from magnetometer). This is in μT .

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
--------------	--

Returns

Y axis compass heading.

4.9.4.6 `static float caerIMU9EventGetCompZ (caerIMU9Event 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.

4.9.4.7 `static float caerIMU9EventGetGyroX (caerIMU9Event 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).

4.9.4.8 `static float caerIMU9EventGetGyroY (caerIMU9Event 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).

4.9.4.9 `static float caerIMU9EventGetGyroZ (caerIMU9Event 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).

4.9.4.10 `static float caerIMU9EventGetTemp (caerIMU9Event 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.

4.9.4.11 `static int32_t caerIMU9EventGetTimestamp (caerIMU9Event 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.

4.9.4.12 `static int64_t caerIMU9EventGetTimestamp64 (caerIMU9Event event, caerIMU9EventPacket 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.

4.9.4.13 `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.

4.9.4.14 `static bool caerIMU9EventsValid (caerIMU9Event 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.

4.9.4.15 `caerIMU9EventPacket caerIMU9EventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)`

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.

4.9.4.16 `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.

4.9.4.17 `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.

4.9.4.18 `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.

4.9.4.19 `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.

4.9.4.20 `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.

4.9.4.21 `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.

4.9.4.22 `static void caerIMU9EventSetCompZ (caerIMU9Event event, float compZ)` `[inline],[static]`

Set the Z axis compass heading (from magnetometer). This is in μ T.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>compZ</i>	Z axis compass heading.

4.9.4.23 `static void caerIMU9EventSetGyroX (caerIMU9Event 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 IMU9Event pointer. Cannot be NULL.
<i>gyroX</i>	angular velocity on the X axis (roll).

4.9.4.24 `static void caerIMU9EventSetGyroY (caerIMU9Event 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 IMU9Event pointer. Cannot be NULL.
<i>gyroY</i>	angular velocity on the Y axis (pitch).

4.9.4.25 `static void caerIMU9EventSetGyroZ (caerIMU9Event 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 IMU9Event pointer. Cannot be NULL.
<i>gyroZ</i>	angular velocity on the Z axis (yaw).

4.9.4.26 `static void caerIMU9EventSetTemp (caerIMU9Event event, float temp)` `[inline],[static]`

Set the temperature reading. This is in $^{\circ}$ C.

Parameters

<i>event</i>	a valid IMU9Event pointer. Cannot be NULL.
<i>temp</i>	temperature in $^{\circ}$ C.

4.9.4.27 `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.

4.9.4.28 `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.

4.10 events/packetContainer.h File Reference

```
#include "common.h"
```

Data Structures

- struct [caer_event_packet_container](#)

Typedefs

- typedef struct [caer_event_packet_container](#) * [caerEventPacketContainer](#)

Functions

- struct [caer_event_packet_container](#) **__attribute__**((**__packed__**))
- [caerEventPacketContainer](#) [caerEventPacketContainerAllocate](#) (int32_t eventPacketsNumber)
- void [caerEventPacketContainerFree](#) ([caerEventPacketContainer](#) container)
- static int32_t [caerEventPacketContainerGetEventPacketsNumber](#) ([caerEventPacketContainer](#) container)
- static void [caerEventPacketContainerSetEventPacketsNumber](#) ([caerEventPacketContainer](#) container, int32_t eventPacketsNumber)
- static [caerEventPacketHeader](#) [caerEventPacketContainerGetEventPacket](#) ([caerEventPacketContainer](#) container, int32_t n)
- static void [caerEventPacketContainerSetEventPacket](#) ([caerEventPacketContainer](#) container, int32_t n, [caerEventPacketHeader](#) packetHeader)

Variables

- int32_t [eventPacketsNumber](#)
Number of different event packets contained.
- [caerEventPacketHeader](#) [eventPackets](#) []
Array of pointers to the actual event packets.

4.10.1 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.

4.10.2 Typedef Documentation

4.10.2.1 typedef struct caer_event_packet_container* caerEventPacketContainer

Type for pointer to EventPacketContainer data structure.

4.10.3 Function Documentation

4.10.3.1 caerEventPacketContainer caerEventPacketContainerAllocate (int32_t eventPacketsNumber)

Allocate a new EventPacketContainer with enough space to store up to the given number of EventPacket references. All packet references will be NULL initially.

Parameters

<i>eventPacketsNumber</i>	the maximum number of EventPacket references that can be stored in this container.
---------------------------	--

Returns

a valid EventPacketContainer handle or NULL on error.

4.10.3.2 void caerEventPacketContainerFree (caerEventPacketContainer container)

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 reference to NULL before calling this.

Parameters

<i>container</i>	the container to be freed.
------------------	----------------------------

4.10.3.3 static caerEventPacketHeader caerEventPacketContainerGetEventPacket (caerEventPacketContainer container, int32_t n) [inline],[static]

Get the reference for 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 reference to an EventPacket or NULL on error.

4.10.3.4 static int32_t caerEventPacketContainerGetEventPacketsNumber (caerEventPacketContainer container) [inline],[static]

Get the maximum number of EventPacket references 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 references that can be contained.

4.10.3.5 `static void caerEventPacketContainerSetEventPacket (caerEventPacketContainer container, int32_t n, caerEventPacketHeader packetHeader)` `[inline],[static]`

Set the reference for 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 reference to an EventPacket's header. Can be NULL.

4.10.3.6 `static void caerEventPacketContainerSetEventPacketsNumber (caerEventPacketContainer container, int32_t eventPacketsNumber)` `[inline],[static]`

Set the maximum number of EventPacket references 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 references that can be contained.

4.11 events/polarity.h File Reference

```
#include "common.h"
```

Data Structures

- struct `caer_polarity_event`
- struct `caer_polarity_event_packet`

Macros

- `#define CAER_POLARITY_ITERATOR_ALL_START(POLARITY_PACKET)`
- `#define CAER_POLARITY_ITERATOR_ALL_END }`
- `#define CAER_POLARITY_ITERATOR_VALID_START(POLARITY_PACKET)`
- `#define CAER_POLARITY_ITERATOR_VALID_END }`
- `#define POLARITY_SHIFT 1`
- `#define POLARITY_MASK 0x00000001`
- `#define Y_ADDR_SHIFT 2`
- `#define Y_ADDR_MASK 0x00007FFF`
- `#define X_ADDR_SHIFT 17`
- `#define X_ADDR_MASK 0x00007FFF`

Typedefs

- typedef struct `caer_polarity_event` * `caerPolarityEvent`
- typedef struct `caer_polarity_event_packet` * `caerPolarityEventPacket`

Functions

- struct `caer_polarity_event` `__attribute__((__packed__))`
- `caerPolarityEventPacket` `caerPolarityEventPacketAllocate` (int32_t `eventCapacity`, int16_t `eventSource`, int32_t `tsOverflow`)
- static `caerPolarityEvent` `caerPolarityEventPacketGetEvent` (`caerPolarityEventPacket` `packet`, int32_t `n`)
- static int32_t `caerPolarityEventGetTimestamp` (`caerPolarityEvent` `event`)
- static int64_t `caerPolarityEventGetTimestamp64` (`caerPolarityEvent` `event`, `caerPolarityEventPacket` `packet`)
- static void `caerPolarityEventSetTimestamp` (`caerPolarityEvent` `event`, int32_t `timestamp`)
- static bool `caerPolarityEventIsValid` (`caerPolarityEvent` `event`)
- static void `caerPolarityEventValidate` (`caerPolarityEvent` `event`, `caerPolarityEventPacket` `packet`)
- static void `caerPolarityEventInvalidate` (`caerPolarityEvent` `event`, `caerPolarityEventPacket` `packet`)
- static bool `caerPolarityEventGetPolarity` (`caerPolarityEvent` `event`)
- static void `caerPolarityEventSetPolarity` (`caerPolarityEvent` `event`, bool `polarity`)
- static uint16_t `caerPolarityEventGetY` (`caerPolarityEvent` `event`)
- static void `caerPolarityEventSetY` (`caerPolarityEvent` `event`, uint16_t `yAddress`)
- static uint16_t `caerPolarityEventGetX` (`caerPolarityEvent` `event`)
- static void `caerPolarityEventSetX` (`caerPolarityEvent` `event`, uint16_t `xAddress`)

Variables

- uint32_t `data`
Event data. First because of valid mark.
- int32_t `timestamp`
Event timestamp.
- struct `caer_event_packet_header` `packetHeader`
The common event packet header.
- struct `caer_polarity_event` `events` []
The events array.

4.11.1 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 lower left corner of the screen, like in OpenGL.

4.11.2 Macro Definition Documentation

4.11.2.1 `#define CAER_POLARITY_ITERATOR_ALL_END`

Iterator close statement.

4.11.2.2 `#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.

4.11.2.3 `#define CAER_POLARITY_ITERATOR_VALID_END }`

Iterator close statement.

4.11.2.4 `#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.

4.11.2.5 `#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.

4.11.2.6 `#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.

4.11.2.7 `#define 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.

4.11.2.8 `#define 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.

4.11.2.9 `#define 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.

4.11.2.10 `#define 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.

4.11.3 Typedef Documentation

4.11.3.1 `typedef struct caer_polarity_event* caerPolarityEvent`

Type for pointer to polarity event data structure.

4.11.3.2 `typedef struct caer_polarity_event_packet* caerPolarityEventPacket`

Type for pointer to polarity event packet data structure.

4.11.4 Function Documentation

4.11.4.1 `static bool caerPolarityEventGetPolarity (caerPolarityEvent 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.

4.11.4.2 `static int32_t caerPolarityEventGetTimestamp (caerPolarityEvent 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.

4.11.4.3 `static int64_t caerPolarityEventGetTimestamp64 (caerPolarityEvent event, caerPolarityEventPacket 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.

4.11.4.4 static uint16_t caerPolarityEventGetX (caerPolarityEvent event) [inline],[static]

Get the X (column) address for a change event, in pixels. The (0, 0) address is in the lower left corner, like in OpenGL.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
--------------	--

Returns

the event X address.

4.11.4.5 static uint16_t caerPolarityEventGetY (caerPolarityEvent event) [inline],[static]

Get the Y (row) address for a change event, in pixels. The (0, 0) address is in the lower left corner, like in OpenGL.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
--------------	--

Returns

the event Y address.

4.11.4.6 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.

4.11.4.7 static bool caerPolarityEventsValid (caerPolarityEvent 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.

4.11.4.8 **caerPolarityEventPacket** caerPolarityEventPacketAllocate (int32_t *eventCapacity*, int16_t *eventSource*, int32_t *tsOverflow*)

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.

4.11.4.9 `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.

4.11.4.10 `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.

4.11.4.11 `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.

4.11.4.12 `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 lower left corner, like in OpenGL.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
<i>xAddress</i>	the event X address.

4.11.4.13 `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 lower left corner, like in OpenGL.

Parameters

<i>event</i>	a valid PolarityEvent pointer. Cannot be NULL.
<i>yAddress</i>	the event Y address.

4.11.4.14 `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.

4.12 events/sample.h File Reference

```
#include "common.h"
```

Data Structures

- struct [caer_sample_event](#)
- struct [caer_sample_event_packet](#)

Macros

- `#define CAER_SAMPLE_ITERATOR_ALL_START(SAMPLE_PACKET)`
- `#define CAER_SAMPLE_ITERATOR_ALL_END }`
- `#define CAER_SAMPLE_ITERATOR_VALID_START(SAMPLE_PACKET)`
- `#define CAER_SAMPLE_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 struct [caer_sample_event_packet](#) * [caerSampleEventPacket](#)

Functions

- struct [caer_sample_event](#) `__attribute__((__packed__))`
- [caerSampleEventPacket](#) [caerSampleEventPacketAllocate](#) (int32_t [eventCapacity](#), int16_t [eventSource](#), int32_t [tsOverflow](#))
- static [caerSampleEvent](#) [caerSampleEventPacketGetEvent](#) ([caerSampleEventPacket](#) [packet](#), int32_t [n](#))
- static int32_t [caerSampleEventGetTimestamp](#) ([caerSampleEvent](#) [event](#))

- static int64_t caerSampleEventGetTimestamp64 (caerSampleEvent event, caerSampleEventPacket packet)
- static void caerSampleEventSetTimestamp (caerSampleEvent event, int32_t timestamp)
- static bool caerSampleEventsValid (caerSampleEvent event)
- static void caerSampleEventValidate (caerSampleEvent event, caerSampleEventPacket packet)
- static void caerSampleEventInvalidate (caerSampleEvent event, caerSampleEventPacket packet)
- static uint8_t caerSampleEventGetType (caerSampleEvent event)
- static void caerSampleEventSetType (caerSampleEvent event, uint8_t type)
- static uint32_t caerSampleEventGetSample (caerSampleEvent event)
- static void caerSampleEventSetSample (caerSampleEvent event, uint32_t sample)

Variables

- uint32_t data
Event data. First because of valid mark.
- int32_t timestamp
Event timestamp.
- struct caer_event_packet_header packetHeader
The common event packet header.
- struct caer_sample_event events []
The events array.

4.12.1 Detailed Description

Sample (ADC) Events format definition and handling functions. Represents different types of ADC readings, up to 24 bits of resolution.

4.12.2 Macro Definition Documentation

4.12.2.1 #define CAER_SAMPLE_ITERATOR_ALL_END }

Iterator close statement.

4.12.2.2 #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.

4.12.2.3 #define CAER_SAMPLE_ITERATOR_VALID_END }

Iterator close statement.

4.12.2.4 `#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.

4.12.2.5 `#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.

4.12.2.6 `#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.

4.12.2.7 `#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.

4.12.2.8 `#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.

4.12.3 Typedef Documentation

4.12.3.1 `typedef struct caer_sample_event* caerSampleEvent`

Type for pointer to ADC sample event data structure.

4.12.3.2 `typedef struct caer_sample_event_packet* caerSampleEventPacket`

Type for pointer to ADC sample event packet data structure.

4.12.4 Function Documentation

4.12.4.1 `static uint32_t caerSampleEventGetSample (caerSampleEvent 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.

4.12.4.2 static int32_t caerSampleEventGetTimestamp (caerSampleEvent 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.

4.12.4.3 static int64_t caerSampleEventGetTimestamp64 (caerSampleEvent event, caerSampleEventPacket 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.

4.12.4.4 static uint8_t caerSampleEventGetType (caerSampleEvent 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.

4.12.4.5 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.

4.12.4.6 static bool caerSampleEventsValid (caerSampleEvent 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.

4.12.4.7 caerSampleEventPacket caerSampleEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)

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.

4.12.4.8 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.

4.12.4.9 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.

4.12.4.10 `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.

4.12.4.11 `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.

4.12.4.12 `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.

4.13 events/special.h File Reference

```
#include "common.h"
```

Data Structures

- struct [caer_special_event](#)
- struct [caer_special_event_packet](#)

Macros

- `#define` [CAER_SPECIAL_ITERATOR_ALL_START](#)(SPECIAL_PACKET)
- `#define` [CAER_SPECIAL_ITERATOR_ALL_END](#) }
- `#define` [CAER_SPECIAL_ITERATOR_VALID_START](#)(SPECIAL_PACKET)
- `#define` [CAER_SPECIAL_ITERATOR_VALID_END](#) }

- #define `TYPE_SHIFT` 1
- #define `TYPE_MASK` 0x0000007F
- #define `DATA_SHIFT` 8
- #define `DATA_MASK` 0x00FFFFFF

Typedefs

- typedef struct `caer_special_event` * `caerSpecialEvent`
- typedef struct `caer_special_event_packet` * `caerSpecialEventPacket`

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 }

Functions

- struct `caer_special_event` `__attribute__((packed))`
- `caerSpecialEventPacket` `caerSpecialEventPacketAllocate` (int32_t `eventCapacity`, int16_t `eventSource`, int32_t `tsOverflow`)
- static `caerSpecialEvent` `caerSpecialEventPacketGetEvent` (`caerSpecialEventPacket` `packet`, int32_t `n`)
- static int32_t `caerSpecialEventGetTimestamp` (`caerSpecialEvent` `event`)
- static int64_t `caerSpecialEventGetTimestamp64` (`caerSpecialEvent` `event`, `caerSpecialEventPacket` `packet`)
- static void `caerSpecialEventSetTimestamp` (`caerSpecialEvent` `event`, int32_t `timestamp`)
- static bool `caerSpecialEventIsValid` (`caerSpecialEvent` `event`)
- static void `caerSpecialEventValidate` (`caerSpecialEvent` `event`, `caerSpecialEventPacket` `packet`)
- static void `caerSpecialEventInvalidate` (`caerSpecialEvent` `event`, `caerSpecialEventPacket` `packet`)
- static uint8_t `caerSpecialEventGetType` (`caerSpecialEvent` `event`)
- static void `caerSpecialEventSetType` (`caerSpecialEvent` `event`, uint8_t `type`)
- static uint32_t `caerSpecialEventGetData` (`caerSpecialEvent` `event`)
- static void `caerSpecialEventSetData` (`caerSpecialEvent` `event`, uint32_t `data`)

Variables

- enum `caer_special_event_types` `__attribute__((packed))`
- uint32_t `data`
Event data. First because of valid mark.
- int32_t `timestamp`
Event timestamp.
- struct `caer_event_packet_header` `packetHeader`
The common event packet header.
- struct `caer_special_event` `events` []
The events array.

4.13.1 Detailed Description

Special Events format definition and handling functions. This event type encodes special occurrences, such as timestamp related notifications or external input events.

4.13.2 Macro Definition Documentation

4.13.2.1 `#define CAER_SPECIAL_ITERATOR_ALL_END` }

Iterator close statement.

4.13.2.2 `#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.

4.13.2.3 `#define CAER_SPECIAL_ITERATOR_VALID_END` }

Iterator close statement.

4.13.2.4 `#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.

4.13.2.5 `#define 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.

4.13.2.6 `#define 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.

4.13.2.7 `#define 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.

4.13.2.8 `#define 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.

4.13.3 Typedef Documentation

4.13.3.1 `typedef struct caer_special_event* caerSpecialEvent`

Type for pointer to special event data structure.

4.13.3.2 `typedef struct caer_special_event_packet* caerSpecialEventPacket`

Type for pointer to special event packet data structure.

4.13.4 Enumeration Type Documentation

4.13.4.1 `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).

4.13.5 Function Documentation

4.13.5.1 `static uint32_t caerSpecialEventGetData (caerSpecialEvent 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.

4.13.5.2 `static int32_t caerSpecialEventGetTimestamp (caerSpecialEvent 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.

4.13.5.3 `static int64_t caerSpecialEventGetTimestamp64 (caerSpecialEvent event, caerSpecialEventPacket 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.

4.13.5.4 `static uint8_t caerSpecialEventGetType (caerSpecialEvent 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').

4.13.5.5 `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.

4.13.5.6 `static bool caerSpecialEventsValid (caerSpecialEvent 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.

4.13.5.7 `caerSpecialEventPacket caerSpecialEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)`

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.

4.13.5.8 `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.

4.13.5.9 `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.

4.13.5.10 `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.

4.13.5.11 `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').

4.13.5.12 `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.

4.14 libcaer.h File Reference

```
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <stdint.h>
#include <inttypes.h>
#include <string.h>
#include <errno.h>
#include "portable_endian.h"
#include "log.h"
```

Macros

- `#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) &= htole32(~(`U32T`(`U32T`(MASK) << (SHIFT))))
- #define `CLEAR_NUMBITS16`(VAR, SHIFT, MASK) (VAR) &= htole16(~(`U16T`(`U16T`(MASK) << (SHIFT))))
- #define `CLEAR_NUMBITS8`(VAR, SHIFT, MASK) (VAR) &= `U8T`(~(`U8T`(`U8T`(MASK) << (SHIFT))))
- #define `SET_NUMBITS32`(VAR, SHIFT, MASK, VALUE) (VAR) |= htole32(`U32T`((`U32T`(VALUE) & (MASK)) << (SHIFT)))
- #define `SET_NUMBITS16`(VAR, SHIFT, MASK, VALUE) (VAR) |= htole16(`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))

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` (uint32_t integer, uint8_t *byteArray, uint8_t byteArrayLength)
- static uint32_t `caerByteArrayToInteger` (uint8_t *byteArray, uint8_t byteArrayLength)

4.14.1 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>`.

4.14.2 Macro Definition Documentation

4.14.2.1 `#define CLEAR_NUMBITS16(VAR, SHIFT, MASK) (VAR) &= htole16(~(U16T(U16T(MASK) << (SHIFT))))`

Clear bits given by mask (amount) and shift (position).

4.14.2.2 `#define CLEAR_NUMBITS32(VAR, SHIFT, MASK) (VAR) &= htole32(~(U32T(U32T(MASK) << (SHIFT))))`

Clear bits given by mask (amount) and shift (position).

4.14.2.3 `#define CLEAR_NUMBITS8(VAR, SHIFT, MASK) (VAR) &= U8T(~(U8T(U8T(MASK) << (SHIFT))))`

Clear bits given by mask (amount) and shift (position).

4.14.2.4 `#define GET_NUMBITS16(VAR, SHIFT, MASK) ((le16toh(VAR) >> (SHIFT)) & (MASK))`

Get value of bits given by mask (amount) and shift (position).

4.14.2.5 `#define GET_NUMBITS32(VAR, SHIFT, MASK) ((le32toh(VAR) >> (SHIFT)) & (MASK))`

Get value of bits given by mask (amount) and shift (position).

4.14.2.6 `#define GET_NUMBITS8(VAR, SHIFT, MASK) ((U8T(VAR) >> (SHIFT)) & (MASK))`

Get value of bits given by mask (amount) and shift (position).

4.14.2.7 `#define I16T(X) ((int16_t) (X))`

Cast argument to `int16_t` (16bit signed integer).

4.14.2.8 `#define I32T(X) ((int32_t) (X))`

Cast argument to `int32_t` (32bit signed integer).

4.14.2.9 `#define I64T(X) ((int64_t) (X))`

Cast argument to `int64_t` (64bit signed integer).

4.14.2.10 `#define I8T(X) ((int8_t) (X))`

Cast argument to `int8_t` (8bit signed integer).

4.14.2.11 `#define MASK_NUMBITS32(X) U32T(U32T(U32T(1) << X) - 1)`

Mask and keep only the lower X bits of a 32bit (unsigned) integer.

4.14.2.12 `#define MASK_NUMBITS64(X) U64T(U64T(U64T(1) << X) - 1)`

Mask and keep only the lower X bits of a 64bit (unsigned) integer.

4.14.2.13 `#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.

4.14.2.14 `#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.

4.14.2.15 `#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.

4.14.2.16 `#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.

4.14.2.17 `#define U16T(X) ((uint16_t) (X))`

Cast argument to `uint16_t` (16bit unsigned integer).

4.14.2.18 `#define U32T(X) ((uint32_t)(X))`

Cast argument to `uint32_t` (32bit unsigned integer).

4.14.2.19 `#define U64T(X) ((uint64_t)(X))`

Cast argument to `uint64_t` (64bit unsigned integer).

4.14.2.20 `#define U8T(X) ((uint8_t)(X))`

Cast argument to `uint8_t` (8bit unsigned integer).

4.14.3 Function Documentation

4.14.3.1 `static uint32_t caerByteArrayToInteger (uint8_t * byteArray, 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.

4.14.3.2 `static void caerIntegerToByteArray (uint32_t integer, uint8_t * byteArray, 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.

4.14.3.3 `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.

4.14.3.4 `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.

4.15 log.h File Reference

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

Macros

- `#define CAER_LOG_EMERGENCY (0)`
- `#define CAER_LOG_ALERT (1)`
- `#define CAER_LOG_CRITICAL (2)`
- `#define CAER_LOG_ERROR (3)`
- `#define CAER_LOG_WARNING (4)`
- `#define CAER_LOG_NOTICE (5)`
- `#define CAER_LOG_INFO (6)`
- `#define CAER_LOG_DEBUG (7)`

Functions

- void `caerLogLevelSet` (uint8_t logLevel)
- uint8_t `caerLogLevelGet` (void)
- void `caerLogFileDescriptorsSet` (int fd1, int fd2)
- void `caerLog` (uint8_t logLevel, const char *subSystem, const char *format,...) `__attribute__((format(printf`

4.15.1 Detailed Description

Logging functions to print useful messages for the user.

4.15.2 Macro Definition Documentation

4.15.2.1 `#define CAER_LOG_ALERT (1)`

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.

4.15.2.2 `#define CAER_LOG_CRITICAL (2)`

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.

4.15.2.3 `#define CAER_LOG_DEBUG (7)`

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.

4.15.2.4 `#define CAER_LOG_EMERGENCY (0)`

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.

4.15.2.5 `#define CAER_LOG_ERROR (3)`

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.

4.15.2.6 `#define CAER_LOG_INFO (6)`

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.

4.15.2.7 `#define CAER_LOG_NOTICE (5)`

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.

4.15.2.8 `#define CAER_LOG_WARNING (4)`

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.

4.15.3 Function Documentation

4.15.3.1 `void caerLog (uint8_t 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 printf()).
...	the parameters to be formatted according to the format string (see printf()).

4.15.3.2 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.

4.15.3.3 uint8_t 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.

4.15.3.4 void caerLogLevelSet (uint8_t *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.
-----------------	----------------------------

4.16 portable_endian.h File Reference

4.16.1 Detailed Description

Endianness conversion functions for a wide variety of systems, including Linux, FreeBSD, MacOS X and Windows.

Index

CAER_CONFIGURATION_ITERATOR_ALL_END
config.h, 111

CAER_CONFIGURATION_ITERATOR_ALL_START
config.h, 111

CAER_CONFIGURATION_ITERATOR_VALID_END
config.h, 112

CAER_CONFIGURATION_ITERATOR_VALID_START
config.h, 112

CAER_DEVICE_DAVIS_FX2
davis.h, 28

CAER_DEVICE_DAVIS_FX3
davis.h, 28

CAER_DEVICE_DVS128
dvs128.h, 93

CAER_EAR_ITERATOR_ALL_END
ear.h, 117

CAER_EAR_ITERATOR_ALL_START
ear.h, 117

CAER_EAR_ITERATOR_VALID_END
ear.h, 118

CAER_EAR_ITERATOR_VALID_START
ear.h, 118

CAER_EVENT_PACKET_HEADER_SIZE
common.h, 103

CAER_FRAME_ITERATOR_ALL_END
frame.h, 126

CAER_FRAME_ITERATOR_ALL_START
frame.h, 126

CAER_FRAME_ITERATOR_VALID_END
frame.h, 126

CAER_FRAME_ITERATOR_VALID_START
frame.h, 126

CAER_HOST_CONFIG_DATAEXCHANGE
usb.h, 96

CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING
usb.h, 96

CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE
usb.h, 97

CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS
usb.h, 97

CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS
usb.h, 97

CAER_HOST_CONFIG_PACKETS
usb.h, 97

CAER_HOST_CONFIG_PACKETS_MAX_CONTAINERS_INTERVAL
usb.h, 97

CAER_HOST_CONFIG_PACKETS_MAX_CONTAINERS_SIZE
usb.h, 97

CAER_HOST_CONFIG_PACKETS_MAX_FRAME_INTERVAL
usb.h, 97

CAER_HOST_CONFIG_PACKETS_MAX_FRAME_SIZE
usb.h, 97

CAER_HOST_CONFIG_PACKETS_MAX_IMU6_INTERVAL
usb.h, 98

CAER_HOST_CONFIG_PACKETS_MAX_IMU6_SIZE
usb.h, 98

CAER_HOST_CONFIG_PACKETS_MAX_POLARITY_INTERVAL
usb.h, 98

CAER_HOST_CONFIG_PACKETS_MAX_POLARITY_SIZE
usb.h, 98

CAER_HOST_CONFIG_PACKETS_MAX_SPECIAL_INTERVAL
usb.h, 98

CAER_HOST_CONFIG_PACKETS_MAX_SPECIAL_SIZE
usb.h, 98

CAER_HOST_CONFIG_USB
usb.h, 98

CAER_HOST_CONFIG_USB_BUFFER_NUMBER
usb.h, 98

CAER_HOST_CONFIG_USB_BUFFER_SIZE
usb.h, 98

CAER_IMU6_ITERATOR_ALL_END
imu6.h, 142

CAER_IMU6_ITERATOR_ALL_START
imu6.h, 142

CAER_IMU6_ITERATOR_VALID_END
imu6.h, 142

CAER_IMU6_ITERATOR_VALID_START
imu6.h, 142

CAER_IMU9_ITERATOR_ALL_END
imu9.h, 149

CAER_IMU9_ITERATOR_ALL_START
imu9.h, 149

CAER_IMU9_ITERATOR_VALID_END
imu9.h, 149

- CAER_IMU9_ITERATOR_VALID_START
imu9.h, [149](#)
- CAER_ITERATOR_ALL_END
common.h, [103](#)
- CAER_ITERATOR_ALL_START
common.h, [103](#)
- CAER_ITERATOR_VALID_END
common.h, [103](#)
- CAER_ITERATOR_VALID_START
common.h, [103](#)
- CAER_LOG_ALERT
log.h, [183](#)
- CAER_LOG_CRITICAL
log.h, [183](#)
- CAER_LOG_DEBUG
log.h, [183](#)
- CAER_LOG_EMERGENCY
log.h, [184](#)
- CAER_LOG_ERROR
log.h, [184](#)
- CAER_LOG_INFO
log.h, [184](#)
- CAER_LOG_NOTICE
log.h, [184](#)
- CAER_LOG_WARNING
log.h, [184](#)
- CAER_POLARITY_ITERATOR_ALL_END
polarity.h, [160](#)
- CAER_POLARITY_ITERATOR_ALL_START
polarity.h, [160](#)
- CAER_POLARITY_ITERATOR_VALID_END
polarity.h, [161](#)
- CAER_POLARITY_ITERATOR_VALID_START
polarity.h, [161](#)
- CAER_SAMPLE_ITERATOR_ALL_END
sample.h, [167](#)
- CAER_SAMPLE_ITERATOR_ALL_START
sample.h, [167](#)
- CAER_SAMPLE_ITERATOR_VALID_END
sample.h, [167](#)
- CAER_SAMPLE_ITERATOR_VALID_START
sample.h, [167](#)
- CAER_SPECIAL_ITERATOR_ALL_END
special.h, [174](#)
- CAER_SPECIAL_ITERATOR_ALL_START
special.h, [174](#)
- CAER_SPECIAL_ITERATOR_VALID_END
special.h, [174](#)
- CAER_SPECIAL_ITERATOR_VALID_START
special.h, [174](#)
- CHANNEL_MASK
ear.h, [118](#)
- CHANNEL_SHIFT
ear.h, [118](#)
- CLEAR_NUMBITS16
libcaer.h, [179](#)
- CLEAR_NUMBITS32
libcaer.h, [179](#)
- CLEAR_NUMBITS8
libcaer.h, [179](#)
- COLOR_CHANNELS_MASK
frame.h, [126](#)
- COLOR_CHANNELS_SHIFT
frame.h, [126](#)
- COLOR_FILTER_MASK
frame.h, [126](#)
- COLOR_FILTER_SHIFT
frame.h, [127](#)
- CONFIG_EVENT
common.h, [105](#)
- caer_bias_coarsefine, [5](#)
- caer_bias_shiftedsources, [5](#)
- caer_bias_shiftedsources_operating_mode
davis.h, [90](#)
- caer_bias_shiftedsources_voltage_level
davis.h, [90](#)
- caer_bias_vdac, [6](#)
- caer_configuration_event, [6](#)
- caer_configuration_event_packet, [7](#)
- caer_davis_info, [7](#)
- caer_default_event_types
common.h, [104](#)
- caer_dvs128_info, [8](#)
- caer_ear_event, [9](#)
- caer_ear_event_packet, [9](#)
- caer_event_packet_container, [10](#)
- caer_event_packet_header, [10](#)
- caer_frame_event, [11](#)
pixels, [11](#)
- caer_frame_event_color_channels
frame.h, [127](#)
- caer_frame_event_color_filter
frame.h, [127](#)
- caer_frame_event_packet, [12](#)
- caer_imu6_event, [12](#)
- caer_imu6_event_packet, [13](#)
- caer_imu9_event, [13](#)
- caer_imu9_event_packet, [14](#)
- caer_polarity_event, [14](#)
- caer_polarity_event_packet, [15](#)
- caer_sample_event, [15](#)
- caer_sample_event_packet, [16](#)
- caer_special_event, [16](#)
- caer_special_event_packet, [17](#)
- caer_special_event_types
special.h, [175](#)
- caerBiasCoarseFineGenerate
davis.h, [91](#)
- caerBiasCoarseFineParse
davis.h, [91](#)
- caerBiasShiftedSourceGenerate
davis.h, [91](#)
- caerBiasShiftedSourceParse
davis.h, [91](#)
- caerBiasVDACGenerate
davis.h, [92](#)

- caerBiasVDACParse
 - davis.h, [92](#)
- caerByteArrayToInteger
 - libcaer.h, [181](#)
- caerConfigurationEvent
 - config.h, [112](#)
- caerConfigurationEventGetModuleAddress
 - config.h, [113](#)
- caerConfigurationEventGetParameter
 - config.h, [113](#)
- caerConfigurationEventGetParameterAddress
 - config.h, [113](#)
- caerConfigurationEventGetTimestamp
 - config.h, [113](#)
- caerConfigurationEventGetTimestamp64
 - config.h, [114](#)
- caerConfigurationEventInvalidate
 - config.h, [114](#)
- caerConfigurationEventIsValid
 - config.h, [114](#)
- caerConfigurationEventPacket
 - config.h, [112](#)
- caerConfigurationEventPacketAllocate
 - config.h, [114](#)
- caerConfigurationEventPacketGetEvent
 - config.h, [115](#)
- caerConfigurationEventSetModuleAddress
 - config.h, [115](#)
- caerConfigurationEventSetParameter
 - config.h, [115](#)
- caerConfigurationEventSetParameterAddress
 - config.h, [115](#)
- caerConfigurationEventSetTimestamp
 - config.h, [116](#)
- caerConfigurationEventValidate
 - config.h, [116](#)
- caerCopyEventPacket
 - common.h, [105](#)
- caerCopyEventPacketOnlyEvents
 - common.h, [105](#)
- caerCopyEventPacketOnlyValidEvents
 - common.h, [105](#)
- caerDVS128InfoGet
 - dvs128.h, [95](#)
- caerDavisInfoGet
 - davis.h, [92](#)
- caerDeviceClose
 - usb.h, [99](#)
- caerDeviceConfigGet
 - usb.h, [99](#)
- caerDeviceConfigSet
 - usb.h, [99](#)
- caerDeviceDataGet
 - usb.h, [100](#)
- caerDeviceDataStart
 - usb.h, [100](#)
- caerDeviceDataStop
 - usb.h, [100](#)
- caerDeviceHandle
 - usb.h, [99](#)
- caerDeviceOpen
 - usb.h, [101](#)
- caerDeviceSendDefaultConfig
 - usb.h, [101](#)
- caerEarEvent
 - ear.h, [119](#)
- caerEarEventGetChannel
 - ear.h, [119](#)
- caerEarEventGetEar
 - ear.h, [119](#)
- caerEarEventGetTimestamp
 - ear.h, [121](#)
- caerEarEventGetTimestamp64
 - ear.h, [121](#)
- caerEarEventInvalidate
 - ear.h, [121](#)
- caerEarEventIsValid
 - ear.h, [121](#)
- caerEarEventPacket
 - ear.h, [119](#)
- caerEarEventPacketAllocate
 - ear.h, [122](#)
- caerEarEventPacketGetEvent
 - ear.h, [122](#)
- caerEarEventSetChannel
 - ear.h, [122](#)
- caerEarEventSetEar
 - ear.h, [122](#)
- caerEarEventSetTimestamp
 - ear.h, [123](#)
- caerEarEventValidate
 - ear.h, [123](#)
- caerEventPacketContainer
 - packetContainer.h, [158](#)
- caerEventPacketContainerAllocate
 - packetContainer.h, [158](#)
- caerEventPacketContainerFree
 - packetContainer.h, [158](#)
- caerEventPacketContainerGetEventPacket
 - packetContainer.h, [158](#)
- caerEventPacketContainerGetEventPacketsNumber
 - packetContainer.h, [158](#)
- caerEventPacketContainerSetEventPacket
 - packetContainer.h, [159](#)
- caerEventPacketContainerSetEventPacketsNumber
 - packetContainer.h, [159](#)
- caerEventPacketHeader
 - common.h, [104](#)
- caerEventPacketHeaderGetEventCapacity
 - common.h, [105](#)
- caerEventPacketHeaderGetEventNumber
 - common.h, [106](#)
- caerEventPacketHeaderGetEventSize
 - common.h, [106](#)
- caerEventPacketHeaderGetEventSource
 - common.h, [106](#)

- caerEventPacketHeaderGetEventTSOffset
common.h, [106](#)
- caerEventPacketHeaderGetEventTSOverflow
common.h, [107](#)
- caerEventPacketHeaderGetEventType
common.h, [107](#)
- caerEventPacketHeaderGetEventValid
common.h, [107](#)
- caerEventPacketHeaderSetEventCapacity
common.h, [107](#)
- caerEventPacketHeaderSetEventNumber
common.h, [108](#)
- caerEventPacketHeaderSetEventSize
common.h, [108](#)
- caerEventPacketHeaderSetEventSource
common.h, [108](#)
- caerEventPacketHeaderSetEventTSOffset
common.h, [108](#)
- caerEventPacketHeaderSetEventTSOverflow
common.h, [108](#)
- caerEventPacketHeaderSetEventType
common.h, [109](#)
- caerEventPacketHeaderSetEventValid
common.h, [109](#)
- caerFrameEvent
frame.h, [127](#)
- caerFrameEventGetChannelNumber
frame.h, [128](#)
- caerFrameEventGetColorFilter
frame.h, [128](#)
- caerFrameEventGetExposureLength
frame.h, [128](#)
- caerFrameEventGetLengthX
frame.h, [128](#)
- caerFrameEventGetLengthY
frame.h, [129](#)
- caerFrameEventGetPixel
frame.h, [129](#)
- caerFrameEventGetPixelFormatCGFormat
frame.h, [129](#)
- caerFrameEventGetPixelFormatUnsafe
frame.h, [129](#)
- caerFrameEventGetPixelForChannel
frame.h, [130](#)
- caerFrameEventGetPixelForChannelUnsafe
frame.h, [130](#)
- caerFrameEventGetPixelUnsafe
frame.h, [131](#)
- caerFrameEventGetPixelsMaxIndex
frame.h, [130](#)
- caerFrameEventGetPixelsSize
frame.h, [130](#)
- caerFrameEventGetPositionX
frame.h, [131](#)
- caerFrameEventGetPositionY
frame.h, [131](#)
- caerFrameEventGetROIIdentifier
frame.h, [131](#)
- caerFrameEventGetTSEndOfExposure
frame.h, [132](#)
- caerFrameEventGetTSEndOfExposure64
frame.h, [132](#)
- caerFrameEventGetTSEndOfFrame
frame.h, [133](#)
- caerFrameEventGetTSEndOfFrame64
frame.h, [133](#)
- caerFrameEventGetTSStartOfExposure
frame.h, [133](#)
- caerFrameEventGetTSStartOfExposure64
frame.h, [133](#)
- caerFrameEventGetTSStartOfFrame
frame.h, [135](#)
- caerFrameEventGetTSStartOfFrame64
frame.h, [135](#)
- caerFrameEventGetTimestamp
frame.h, [132](#)
- caerFrameEventGetTimestamp64
frame.h, [132](#)
- caerFrameEventInvalidate
frame.h, [135](#)
- caerFrameEventIsValid
frame.h, [135](#)
- caerFrameEventPacket
frame.h, [127](#)
- caerFrameEventPacketAllocate
frame.h, [136](#)
- caerFrameEventPacketGetEvent
frame.h, [136](#)
- caerFrameEventPacketGetPixelsMaxIndex
frame.h, [136](#)
- caerFrameEventPacketGetPixelsSize
frame.h, [137](#)
- caerFrameEventSetColorFilter
frame.h, [137](#)
- caerFrameEventSetLengthXLengthYChannelNumber
frame.h, [137](#)
- caerFrameEventSetPixel
frame.h, [137](#)
- caerFrameEventSetPixelForChannel
frame.h, [138](#)
- caerFrameEventSetPixelForChannelUnsafe
frame.h, [138](#)
- caerFrameEventSetPixelUnsafe
frame.h, [138](#)
- caerFrameEventSetPositionX
frame.h, [138](#)
- caerFrameEventSetPositionY
frame.h, [139](#)
- caerFrameEventSetROIIdentifier
frame.h, [139](#)
- caerFrameEventSetTSEndOfExposure
frame.h, [139](#)
- caerFrameEventSetTSEndOfFrame
frame.h, [139](#)
- caerFrameEventSetTSStartOfExposure
frame.h, [139](#)

caerFrameEventSetTSStartOfFrame
frame.h, [140](#)

caerFrameEventValidate
frame.h, [140](#)

caerGenericEventGetEvent
common.h, [109](#)

caerGenericEventGetTimestamp
common.h, [109](#)

caerGenericEventGetTimestamp64
common.h, [109](#)

caerGenericEventsIsValid
common.h, [110](#)

caerIMU6Event
imu6.h, [142](#)

caerIMU6EventGetAccelX
imu6.h, [143](#)

caerIMU6EventGetAccelY
imu6.h, [143](#)

caerIMU6EventGetAccelZ
imu6.h, [143](#)

caerIMU6EventGetGyroX
imu6.h, [143](#)

caerIMU6EventGetGyroY
imu6.h, [143](#)

caerIMU6EventGetGyroZ
imu6.h, [144](#)

caerIMU6EventGetTemp
imu6.h, [144](#)

caerIMU6EventGetTimestamp
imu6.h, [144](#)

caerIMU6EventGetTimestamp64
imu6.h, [144](#)

caerIMU6EventInvalidate
imu6.h, [145](#)

caerIMU6EventsIsValid
imu6.h, [145](#)

caerIMU6EventPacket
imu6.h, [142](#)

caerIMU6EventPacketAllocate
imu6.h, [145](#)

caerIMU6EventPacketGetEvent
imu6.h, [145](#)

caerIMU6EventSetAccelX
imu6.h, [146](#)

caerIMU6EventSetAccelY
imu6.h, [146](#)

caerIMU6EventSetAccelZ
imu6.h, [146](#)

caerIMU6EventSetGyroX
imu6.h, [146](#)

caerIMU6EventSetGyroY
imu6.h, [146](#)

caerIMU6EventSetGyroZ
imu6.h, [147](#)

caerIMU6EventSetTemp
imu6.h, [147](#)

caerIMU6EventSetTimestamp
imu6.h, [147](#)

caerIMU6EventValidate
imu6.h, [147](#)

caerIMU9Event
imu9.h, [150](#)

caerIMU9EventGetAccelX
imu9.h, [150](#)

caerIMU9EventGetAccelY
imu9.h, [150](#)

caerIMU9EventGetAccelZ
imu9.h, [150](#)

caerIMU9EventGetCompX
imu9.h, [151](#)

caerIMU9EventGetCompY
imu9.h, [151](#)

caerIMU9EventGetCompZ
imu9.h, [151](#)

caerIMU9EventGetGyroX
imu9.h, [151](#)

caerIMU9EventGetGyroY
imu9.h, [151](#)

caerIMU9EventGetGyroZ
imu9.h, [153](#)

caerIMU9EventGetTemp
imu9.h, [153](#)

caerIMU9EventGetTimestamp
imu9.h, [153](#)

caerIMU9EventGetTimestamp64
imu9.h, [153](#)

caerIMU9EventInvalidate
imu9.h, [154](#)

caerIMU9EventsIsValid
imu9.h, [154](#)

caerIMU9EventPacket
imu9.h, [150](#)

caerIMU9EventPacketAllocate
imu9.h, [154](#)

caerIMU9EventPacketGetEvent
imu9.h, [154](#)

caerIMU9EventSetAccelX
imu9.h, [155](#)

caerIMU9EventSetAccelY
imu9.h, [155](#)

caerIMU9EventSetAccelZ
imu9.h, [155](#)

caerIMU9EventSetCompX
imu9.h, [155](#)

caerIMU9EventSetCompY
imu9.h, [155](#)

caerIMU9EventSetCompZ
imu9.h, [156](#)

caerIMU9EventSetGyroX
imu9.h, [156](#)

caerIMU9EventSetGyroY
imu9.h, [156](#)

caerIMU9EventSetGyroZ
imu9.h, [156](#)

caerIMU9EventSetTemp
imu9.h, [156](#)

- caerIMU9EventSetTimestamp
 - imu9.h, [156](#)
- caerIMU9EventValidate
 - imu9.h, [157](#)
- caerIntegerToByteArray
 - libcaer.h, [181](#)
- caerLog
 - log.h, [184](#)
- caerLogFileDescriptorsSet
 - log.h, [185](#)
- caerLogLevelGet
 - log.h, [185](#)
- caerLogLevelSet
 - log.h, [185](#)
- caerPolarityEvent
 - polarity.h, [162](#)
- caerPolarityEventGetPolarity
 - polarity.h, [162](#)
- caerPolarityEventGetTimestamp
 - polarity.h, [162](#)
- caerPolarityEventGetTimestamp64
 - polarity.h, [162](#)
- caerPolarityEventGetX
 - polarity.h, [163](#)
- caerPolarityEventGetY
 - polarity.h, [163](#)
- caerPolarityEventInvalidate
 - polarity.h, [163](#)
- caerPolarityEventIsValid
 - polarity.h, [163](#)
- caerPolarityEventPacket
 - polarity.h, [162](#)
- caerPolarityEventPacketAllocate
 - polarity.h, [163](#)
- caerPolarityEventPacketGetEvent
 - polarity.h, [165](#)
- caerPolarityEventSetPolarity
 - polarity.h, [165](#)
- caerPolarityEventSetTimestamp
 - polarity.h, [165](#)
- caerPolarityEventSetX
 - polarity.h, [165](#)
- caerPolarityEventSetY
 - polarity.h, [165](#)
- caerPolarityEventValidate
 - polarity.h, [166](#)
- caerSampleEvent
 - sample.h, [168](#)
- caerSampleEventGetSample
 - sample.h, [168](#)
- caerSampleEventGetTimestamp
 - sample.h, [170](#)
- caerSampleEventGetTimestamp64
 - sample.h, [170](#)
- caerSampleEventGetType
 - sample.h, [170](#)
- caerSampleEventInvalidate
 - sample.h, [170](#)
- caerSampleEventIsValid
 - sample.h, [171](#)
- caerSampleEventPacket
 - sample.h, [168](#)
- caerSampleEventPacketAllocate
 - sample.h, [171](#)
- caerSampleEventPacketGetEvent
 - sample.h, [171](#)
- caerSampleEventSetSample
 - sample.h, [171](#)
- caerSampleEventSetTimestamp
 - sample.h, [172](#)
- caerSampleEventSetType
 - sample.h, [172](#)
- caerSampleEventValidate
 - sample.h, [172](#)
- caerSpecialEvent
 - special.h, [175](#)
- caerSpecialEventGetData
 - special.h, [175](#)
- caerSpecialEventGetTimestamp
 - special.h, [175](#)
- caerSpecialEventGetTimestamp64
 - special.h, [176](#)
- caerSpecialEventGetType
 - special.h, [176](#)
- caerSpecialEventInvalidate
 - special.h, [176](#)
- caerSpecialEventIsValid
 - special.h, [176](#)
- caerSpecialEventPacket
 - special.h, [175](#)
- caerSpecialEventPacketAllocate
 - special.h, [177](#)
- caerSpecialEventPacketGetEvent
 - special.h, [177](#)
- caerSpecialEventSetData
 - special.h, [177](#)
- caerSpecialEventSetTimestamp
 - special.h, [177](#)
- caerSpecialEventSetType
 - special.h, [178](#)
- caerSpecialEventValidate
 - special.h, [178](#)
- caerStrEquals
 - libcaer.h, [181](#)
- caerStrEqualsUpTo
 - libcaer.h, [181](#)
- common.h
 - CAER_EVENT_PACKET_HEADER_SIZE, [103](#)
 - CAER_ITERATOR_ALL_END, [103](#)
 - CAER_ITERATOR_ALL_START, [103](#)
 - CAER_ITERATOR_VALID_END, [103](#)
 - CAER_ITERATOR_VALID_START, [103](#)
 - CONFIG_EVENT, [105](#)
 - caer_default_event_types, [104](#)
 - caerCopyEventPacket, [105](#)
 - caerCopyEventPacketOnlyEvents, [105](#)

- caerCopyEventPacketOnlyValidEvents, 105
- caerEventPacketHeader, 104
- caerEventPacketHeaderGetEventCapacity, 105
- caerEventPacketHeaderGetEventNumber, 106
- caerEventPacketHeaderGetEventSize, 106
- caerEventPacketHeaderGetEventSource, 106
- caerEventPacketHeaderGetEventTSOffset, 106
- caerEventPacketHeaderGetEventTSOverflow, 107
- caerEventPacketHeaderGetEventType, 107
- caerEventPacketHeaderGetEventValid, 107
- caerEventPacketHeaderSetEventCapacity, 107
- caerEventPacketHeaderSetEventNumber, 108
- caerEventPacketHeaderSetEventSize, 108
- caerEventPacketHeaderSetEventSource, 108
- caerEventPacketHeaderSetEventTSOffset, 108
- caerEventPacketHeaderSetEventTSOverflow, 108
- caerEventPacketHeaderSetEventType, 109
- caerEventPacketHeaderSetEventValid, 109
- caerGenericEventGetEvent, 109
- caerGenericEventGetTimestamp, 109
- caerGenericEventGetTimestamp64, 109
- caerGenericEventsIsValid, 110
- EAR_EVENT, 105
- FRAME_EVENT, 105
- IMU6_EVENT, 105
- IMU9_EVENT, 105
- POLARITY_EVENT, 104
- SAMPLE_EVENT, 105
- SPECIAL_EVENT, 104
- TS_OVERFLOW_SHIFT, 104
- VALID_MARK_MASK, 104
- VALID_MARK_SHIFT, 104
- config.h
 - CAER_CONFIGURATION_ITERATOR_ALL_EVENTS, 111
 - CAER_CONFIGURATION_ITERATOR_ALL_START, 111
 - CAER_CONFIGURATION_ITERATOR_VALID_END, 112
 - CAER_CONFIGURATION_ITERATOR_VALID_START, 112
 - caerConfigurationEvent, 112
 - caerConfigurationEventGetModuleAddress, 113
 - caerConfigurationEventGetParameter, 113
 - caerConfigurationEventGetParameterAddress, 113
 - caerConfigurationEventGetTimestamp, 113
 - caerConfigurationEventGetTimestamp64, 114
 - caerConfigurationEventInvalidate, 114
 - caerConfigurationEventsIsValid, 114
 - caerConfigurationEventPacket, 112
 - caerConfigurationEventPacketAllocate, 114
 - caerConfigurationEventPacketGetEvent, 115
 - caerConfigurationEventSetModuleAddress, 115
 - caerConfigurationEventSetParameter, 115
 - caerConfigurationEventSetParameterAddress, 115
 - caerConfigurationEventSetTimestamp, 116
 - caerConfigurationEventValidate, 116
 - MODULE_ADDR_MASK, 112
 - MODULE_ADDR_SHIFT, 112
 - DATA_MASK
 - special.h, 174
 - DATA_SHIFT
 - special.h, 174
 - DAVIS128_CONFIG_BIAS_ADCCOMPBP
 - davis.h, 28
 - DAVIS128_CONFIG_BIAS_ADCREFHIGH
 - davis.h, 28
 - DAVIS128_CONFIG_BIAS_ADCREFLOW
 - davis.h, 28
 - DAVIS128_CONFIG_BIAS_AEPDBN
 - davis.h, 28
 - DAVIS128_CONFIG_BIAS_AEPUXBP
 - davis.h, 28
 - DAVIS128_CONFIG_BIAS_AEPUYBP
 - davis.h, 29
 - DAVIS128_CONFIG_BIAS_APSCAS
 - davis.h, 29
 - DAVIS128_CONFIG_BIAS_APSEVERFLOWLEVEL
 - davis.h, 29
 - DAVIS128_CONFIG_BIAS_APSROSFBN
 - davis.h, 29
 - DAVIS128_CONFIG_BIAS_BIASBUFFER
 - davis.h, 29
 - DAVIS128_CONFIG_BIAS_COLSELLOWBN
 - davis.h, 30
 - DAVIS128_CONFIG_BIAS_DACBUFBP
 - davis.h, 30
 - DAVIS128_CONFIG_BIAS_DIFFBN
 - davis.h, 30
 - DAVIS128_CONFIG_BIAS_IFREFRBN
 - davis.h, 30
 - DAVIS128_CONFIG_BIAS_IFTHRBN
 - davis.h, 30
 - DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN
 - davis.h, 31
 - DAVIS128_CONFIG_BIAS_LOCALBUFBN
 - davis.h, 31
 - DAVIS128_CONFIG_BIAS_OFFBN
 - davis.h, 31
 - DAVIS128_CONFIG_BIAS_ONBN
 - davis.h, 31
 - DAVIS128_CONFIG_BIAS_PADFOLLBN
 - davis.h, 31
 - DAVIS128_CONFIG_BIAS_PIXINBN
 - davis.h, 32
 - DAVIS128_CONFIG_BIAS_PRBP
 - davis.h, 32
 - DAVIS128_CONFIG_BIAS_PRSFBP
 - davis.h, 32
 - DAVIS128_CONFIG_BIAS_READOUTBUFBP
 - davis.h, 32
 - DAVIS128_CONFIG_BIAS_REFRBP
 - davis.h, 32
 - DAVIS128_CONFIG_BIAS_SSN
 - davis.h, 33
 - DAVIS128_CONFIG_BIAS_SSP

- davis.h, [33](#)
- 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, [34](#)
- 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, [34](#)
- DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER
 - davis.h, [35](#)
- DAVIS128_CONFIG_CHIP_TYPCALIBNEURON
 - 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, [35](#)
- DAVIS208_CONFIG_BIAS_AEPDBN
 - davis.h, [35](#)
- DAVIS208_CONFIG_BIAS_AEPUXBP
 - davis.h, [36](#)
- DAVIS208_CONFIG_BIAS_AEPUYBP
 - davis.h, [36](#)
- DAVIS208_CONFIG_BIAS_APSCAS
 - davis.h, [36](#)
- DAVIS208_CONFIG_BIAS_APSEVERFLOWLEVEL
 - davis.h, [36](#)
- DAVIS208_CONFIG_BIAS_APSROSFBN
 - davis.h, [36](#)
- DAVIS208_CONFIG_BIAS_BIASBUFFER
 - davis.h, [37](#)
- DAVIS208_CONFIG_BIAS_COLSELLOWBN
 - davis.h, [37](#)
- DAVIS208_CONFIG_BIAS_DACBUFBP
 - davis.h, [37](#)
- DAVIS208_CONFIG_BIAS_DIFFBN
 - davis.h, [37](#)
- DAVIS208_CONFIG_BIAS_IFREFRBN
 - davis.h, [37](#)
- DAVIS208_CONFIG_BIAS_IFTHRBN
 - davis.h, [38](#)
- DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN
 - davis.h, [38](#)
- DAVIS208_CONFIG_BIAS_LOCALBUFBN
 - davis.h, [38](#)
- DAVIS208_CONFIG_BIAS_OFFBN
 - davis.h, [38](#)
- DAVIS208_CONFIG_BIAS_ONBN
 - davis.h, [38](#)
- DAVIS208_CONFIG_BIAS_PADFOLLBN
 - davis.h, [39](#)
- DAVIS208_CONFIG_BIAS_PIXINVBN
 - davis.h, [39](#)
- DAVIS208_CONFIG_BIAS_PRBP
 - davis.h, [39](#)
- DAVIS208_CONFIG_BIAS_PRFSBP
 - davis.h, [39](#)
- DAVIS208_CONFIG_BIAS_READOUTBUFBP
 - davis.h, [39](#)
- DAVIS208_CONFIG_BIAS_REFRBP
 - davis.h, [40](#)
- DAVIS208_CONFIG_BIAS_REFSS
 - davis.h, [40](#)
- DAVIS208_CONFIG_BIAS_REFSSBN
 - davis.h, [40](#)
- DAVIS208_CONFIG_BIAS_REGBIASBP
 - davis.h, [40](#)
- DAVIS208_CONFIG_BIAS_RESETHIGHPASS
 - davis.h, [40](#)
- DAVIS208_CONFIG_BIAS_SSN
 - davis.h, [41](#)
- DAVIS208_CONFIG_BIAS_SSP
 - davis.h, [41](#)
- DAVIS208_CONFIG_CHIP_AERNAROW
 - davis.h, [41](#)
- DAVIS208_CONFIG_CHIP_ANALOGMUX0
 - davis.h, [41](#)
- DAVIS208_CONFIG_CHIP_ANALOGMUX1
 - davis.h, [41](#)
- DAVIS208_CONFIG_CHIP_ANALOGMUX2
 - davis.h, [41](#)
- DAVIS208_CONFIG_CHIP_BIASMUX0
 - davis.h, [42](#)
- DAVIS208_CONFIG_CHIP_DIGITALMUX0
 - davis.h, [42](#)
- DAVIS208_CONFIG_CHIP_DIGITALMUX1
 - davis.h, [42](#)
- DAVIS208_CONFIG_CHIP_DIGITALMUX2
 - davis.h, [42](#)
- DAVIS208_CONFIG_CHIP_DIGITALMUX3
 - davis.h, [42](#)
- DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER
 - davis.h, [42](#)
- DAVIS208_CONFIG_CHIP_RESETCALIBNEURON
 - davis.h, [42](#)
- DAVIS208_CONFIG_CHIP_RESETTESTPIXEL

- davis.h, [42](#)
- DAVIS208_CONFIG_CHIP_SELECTBIASREFSS
 - davis.h, [43](#)
- DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER
 - davis.h, [43](#)
- DAVIS208_CONFIG_CHIP_SELECTHIGHPASS
 - davis.h, [43](#)
- DAVIS208_CONFIG_CHIP_SELECTPOSFB
 - davis.h, [43](#)
- DAVIS208_CONFIG_CHIP_SELECTPREAMPAVG
 - davis.h, [43](#)
- DAVIS208_CONFIG_CHIP_SELECTSENSE
 - davis.h, [43](#)
- DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON
 - davis.h, [43](#)
- DAVIS208_CONFIG_CHIP_USEAOUT
 - davis.h, [43](#)
- DAVIS240_CONFIG_BIAS_AEPDBN
 - davis.h, [43](#)
- DAVIS240_CONFIG_BIAS_AEPUXBP
 - davis.h, [44](#)
- DAVIS240_CONFIG_BIAS_AEPUYBP
 - davis.h, [44](#)
- DAVIS240_CONFIG_BIAS_APSCASEPC
 - davis.h, [44](#)
- DAVIS240_CONFIG_BIAS_APSOEVERFLOWLEVEL↵
BN
 - davis.h, [44](#)
- DAVIS240_CONFIG_BIAS_APSROSFBN
 - davis.h, [44](#)
- DAVIS240_CONFIG_BIAS_BIASBUFFER
 - davis.h, [45](#)
- DAVIS240_CONFIG_BIAS_DIFFBN
 - davis.h, [45](#)
- DAVIS240_CONFIG_BIAS_DIFFCASBNC
 - davis.h, [45](#)
- DAVIS240_CONFIG_BIAS_IFREFRBN
 - davis.h, [45](#)
- DAVIS240_CONFIG_BIAS_IFTHRBN
 - davis.h, [45](#)
- DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN
 - davis.h, [46](#)
- DAVIS240_CONFIG_BIAS_LOCALBUFBN
 - davis.h, [46](#)
- DAVIS240_CONFIG_BIAS_OFFBN
 - davis.h, [46](#)
- DAVIS240_CONFIG_BIAS_ONBN
 - davis.h, [46](#)
- DAVIS240_CONFIG_BIAS_PADFOLLBN
 - davis.h, [46](#)
- DAVIS240_CONFIG_BIAS_PIXINBN
 - davis.h, [47](#)
- DAVIS240_CONFIG_BIAS_PRBP
 - davis.h, [47](#)
- DAVIS240_CONFIG_BIAS_PRSFBN
 - davis.h, [47](#)
- DAVIS240_CONFIG_BIAS_REFRBP
 - davis.h, [47](#)
- DAVIS240_CONFIG_BIAS_SSN
 - davis.h, [47](#)
- DAVIS240_CONFIG_BIAS_SSP
 - davis.h, [48](#)
- DAVIS240_CONFIG_CHIP_AERNAROW
 - davis.h, [48](#)
- DAVIS240_CONFIG_CHIP_ANALOGMUX0
 - davis.h, [48](#)
- DAVIS240_CONFIG_CHIP_ANALOGMUX1
 - davis.h, [48](#)
- DAVIS240_CONFIG_CHIP_ANALOGMUX2
 - davis.h, [48](#)
- DAVIS240_CONFIG_CHIP_BIASMUX0
 - davis.h, [48](#)
- DAVIS240_CONFIG_CHIP_DIGITALMUX0
 - davis.h, [48](#)
- DAVIS240_CONFIG_CHIP_DIGITALMUX1
 - davis.h, [49](#)
- DAVIS240_CONFIG_CHIP_DIGITALMUX2
 - davis.h, [49](#)
- DAVIS240_CONFIG_CHIP_DIGITALMUX3
 - davis.h, [49](#)
- DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER
 - davis.h, [49](#)
- DAVIS240_CONFIG_CHIP_RESETCALIBNEURON
 - davis.h, [49](#)
- DAVIS240_CONFIG_CHIP_RESETTESTPIXEL
 - davis.h, [49](#)
- DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTR↵
OL
 - davis.h, [49](#)
- DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON
 - davis.h, [50](#)
- DAVIS240_CONFIG_CHIP_USEAOUT
 - davis.h, [50](#)
- DAVIS346_CONFIG_BIAS_ADCCOMPBP
 - davis.h, [50](#)
- DAVIS346_CONFIG_BIAS_ADCREFHIGH
 - davis.h, [50](#)
- DAVIS346_CONFIG_BIAS_ADCREFLOW
 - davis.h, [50](#)
- DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE
 - davis.h, [50](#)
- DAVIS346_CONFIG_BIAS_AEPDBN
 - davis.h, [51](#)
- DAVIS346_CONFIG_BIAS_AEPUXBP
 - davis.h, [51](#)
- DAVIS346_CONFIG_BIAS_AEPUYBP
 - davis.h, [51](#)
- DAVIS346_CONFIG_BIAS_APSCAS
 - davis.h, [51](#)
- DAVIS346_CONFIG_BIAS_APSOEVERFLOWLEVEL
 - davis.h, [51](#)
- DAVIS346_CONFIG_BIAS_APSROSFBN
 - davis.h, [52](#)
- DAVIS346_CONFIG_BIAS_BIASBUFFER
 - davis.h, [52](#)
- DAVIS346_CONFIG_BIAS_COLSELLOWBN

- davis.h, [52](#)
- DAVIS346_CONFIG_BIAS_DACBUFBP
 - davis.h, [52](#)
- DAVIS346_CONFIG_BIAS_DIFFBN
 - davis.h, [52](#)
- DAVIS346_CONFIG_BIAS_IFREFRBN
 - davis.h, [53](#)
- DAVIS346_CONFIG_BIAS_IFTHRBN
 - davis.h, [53](#)
- DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN
 - davis.h, [53](#)
- DAVIS346_CONFIG_BIAS_LOCALBUFBP
 - davis.h, [53](#)
- DAVIS346_CONFIG_BIAS_OFFBN
 - davis.h, [53](#)
- DAVIS346_CONFIG_BIAS_ONBN
 - davis.h, [54](#)
- DAVIS346_CONFIG_BIAS_PADFOLLBN
 - davis.h, [54](#)
- DAVIS346_CONFIG_BIAS_PIXINVBN
 - davis.h, [54](#)
- DAVIS346_CONFIG_BIAS_PRBP
 - davis.h, [54](#)
- DAVIS346_CONFIG_BIAS_PRSFBP
 - davis.h, [54](#)
- DAVIS346_CONFIG_BIAS_READOUTBUFBP
 - davis.h, [55](#)
- DAVIS346_CONFIG_BIAS_REFRBP
 - davis.h, [55](#)
- DAVIS346_CONFIG_BIAS_SSN
 - davis.h, [55](#)
- DAVIS346_CONFIG_BIAS_SSP
 - davis.h, [55](#)
- DAVIS346_CONFIG_CHIP_AERNAROW
 - davis.h, [55](#)
- DAVIS346_CONFIG_CHIP_ANALOGMUX0
 - davis.h, [56](#)
- DAVIS346_CONFIG_CHIP_ANALOGMUX1
 - davis.h, [56](#)
- DAVIS346_CONFIG_CHIP_ANALOGMUX2
 - davis.h, [56](#)
- DAVIS346_CONFIG_CHIP_BIASMUX0
 - davis.h, [56](#)
- DAVIS346_CONFIG_CHIP_DIGITALMUX0
 - davis.h, [56](#)
- DAVIS346_CONFIG_CHIP_DIGITALMUX1
 - davis.h, [56](#)
- DAVIS346_CONFIG_CHIP_DIGITALMUX2
 - davis.h, [56](#)
- DAVIS346_CONFIG_CHIP_DIGITALMUX3
 - davis.h, [56](#)
- DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER
 - davis.h, [57](#)
- DAVIS346_CONFIG_CHIP_RESETCALIBNEURON
 - davis.h, [57](#)
- DAVIS346_CONFIG_CHIP_RESETTESTPIXEL
 - davis.h, [57](#)
- DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER
 - davis.h, [57](#)
- DAVIS346_CONFIG_CHIP_TESTADC
 - davis.h, [57](#)
- DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON
 - davis.h, [57](#)
- DAVIS346_CONFIG_CHIP_USEAOUT
 - davis.h, [57](#)
- DAVIS640_CONFIG_BIAS_ADCCOMPBP
 - davis.h, [57](#)
- DAVIS640_CONFIG_BIAS_ADCREFHIGH
 - davis.h, [58](#)
- DAVIS640_CONFIG_BIAS_ADCREFLOW
 - davis.h, [58](#)
- DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE
 - davis.h, [58](#)
- DAVIS640_CONFIG_BIAS_AEPDBN
 - davis.h, [58](#)
- DAVIS640_CONFIG_BIAS_AEPUXBP
 - davis.h, [58](#)
- DAVIS640_CONFIG_BIAS_AEPUYBP
 - davis.h, [59](#)
- DAVIS640_CONFIG_BIAS_APSCAS
 - davis.h, [59](#)
- DAVIS640_CONFIG_BIAS_APSEVERFLOWLEVEL
 - davis.h, [59](#)
- DAVIS640_CONFIG_BIAS_APSROSFBP
 - davis.h, [59](#)
- DAVIS640_CONFIG_BIAS_BIASBUFFER
 - davis.h, [59](#)
- DAVIS640_CONFIG_BIAS_COLSELLOWBN
 - davis.h, [60](#)
- DAVIS640_CONFIG_BIAS_DACBUFBP
 - davis.h, [60](#)
- DAVIS640_CONFIG_BIAS_DIFFBN
 - davis.h, [60](#)
- DAVIS640_CONFIG_BIAS_IFREFRBN
 - davis.h, [60](#)
- DAVIS640_CONFIG_BIAS_IFTHRBN
 - davis.h, [60](#)
- DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN
 - davis.h, [61](#)
- DAVIS640_CONFIG_BIAS_LOCALBUFBP
 - davis.h, [61](#)
- DAVIS640_CONFIG_BIAS_OFFBN
 - davis.h, [61](#)
- DAVIS640_CONFIG_BIAS_ONBN
 - davis.h, [61](#)
- DAVIS640_CONFIG_BIAS_PADFOLLBN
 - davis.h, [61](#)
- DAVIS640_CONFIG_BIAS_PIXINVBN
 - davis.h, [62](#)
- DAVIS640_CONFIG_BIAS_PRBP
 - davis.h, [62](#)
- DAVIS640_CONFIG_BIAS_PRSFBP
 - davis.h, [62](#)
- DAVIS640_CONFIG_BIAS_READOUTBUFBP
 - davis.h, [62](#)
- DAVIS640_CONFIG_BIAS_REFRBP
 - davis.h, [62](#)

- davis.h, [62](#)
- DAVIS640_CONFIG_BIAS_SSN
 - davis.h, [63](#)
- DAVIS640_CONFIG_BIAS_SSP
 - davis.h, [63](#)
- DAVIS640_CONFIG_CHIP_AERNAROW
 - davis.h, [63](#)
- DAVIS640_CONFIG_CHIP_ANALOGMUX0
 - davis.h, [63](#)
- DAVIS640_CONFIG_CHIP_ANALOGMUX1
 - davis.h, [63](#)
- DAVIS640_CONFIG_CHIP_ANALOGMUX2
 - davis.h, [63](#)
- DAVIS640_CONFIG_CHIP_BIASMUX0
 - davis.h, [64](#)
- DAVIS640_CONFIG_CHIP_DIGITALMUX0
 - davis.h, [64](#)
- DAVIS640_CONFIG_CHIP_DIGITALMUX1
 - davis.h, [64](#)
- DAVIS640_CONFIG_CHIP_DIGITALMUX2
 - davis.h, [64](#)
- DAVIS640_CONFIG_CHIP_DIGITALMUX3
 - davis.h, [64](#)
- DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER
 - davis.h, [64](#)
- DAVIS640_CONFIG_CHIP_RESETCALIBNEURON
 - davis.h, [64](#)
- DAVIS640_CONFIG_CHIP_RESETTESTPIXEL
 - davis.h, [64](#)
- DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER
 - davis.h, [65](#)
- DAVIS640_CONFIG_CHIP_TESTADC
 - davis.h, [65](#)
- DAVIS640_CONFIG_CHIP_TYPCALIBNEURON
 - davis.h, [65](#)
- DAVIS640_CONFIG_CHIP_USEAOUT
 - davis.h, [65](#)
- DAVIS_CHIP_DAVIS128
 - davis.h, [65](#)
- DAVIS_CHIP_DAVIS208
 - davis.h, [65](#)
- DAVIS_CHIP_DAVIS240A
 - davis.h, [65](#)
- DAVIS_CHIP_DAVIS240B
 - davis.h, [65](#)
- DAVIS_CHIP_DAVIS240C
 - davis.h, [65](#)
- DAVIS_CHIP_DAVIS346A
 - davis.h, [65](#)
- DAVIS_CHIP_DAVIS346B
 - davis.h, [65](#)
- DAVIS_CHIP_DAVIS346C
 - davis.h, [66](#)
- DAVIS_CHIP_DAVIS640
 - davis.h, [66](#)
- DAVIS_CHIP_DAVISRGB
 - davis.h, [66](#)
- DAVIS_CONFIG_APS
 - davis.h, [66](#)
- DAVIS_CONFIG_APS_ADC_TEST_MODE
 - davis.h, [66](#)
- DAVIS_CONFIG_APS_COLOR_FILTER
 - davis.h, [66](#)
- DAVIS_CONFIG_APS_COLUMN_SETTLE
 - davis.h, [66](#)
- DAVIS_CONFIG_APS_END_COLUMN_0
 - davis.h, [66](#)
- DAVIS_CONFIG_APS_END_COLUMN_1
 - davis.h, [66](#)
- DAVIS_CONFIG_APS_END_COLUMN_2
 - davis.h, [66](#)
- DAVIS_CONFIG_APS_END_COLUMN_3
 - davis.h, [66](#)
- DAVIS_CONFIG_APS_END_ROW_0
 - davis.h, [67](#)
- DAVIS_CONFIG_APS_END_ROW_1
 - davis.h, [67](#)
- DAVIS_CONFIG_APS_END_ROW_2
 - davis.h, [67](#)
- DAVIS_CONFIG_APS_END_ROW_3
 - davis.h, [67](#)
- DAVIS_CONFIG_APS_EXPOSURE
 - davis.h, [67](#)
- DAVIS_CONFIG_APS_FRAME_DELAY
 - davis.h, [67](#)
- DAVIS_CONFIG_APS_GLOBAL_SHUTTER
 - davis.h, [67](#)
- DAVIS_CONFIG_APS_HAS_EXTERNAL_ADC
 - davis.h, [67](#)
- DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER
 - davis.h, [67](#)
- DAVIS_CONFIG_APS_HAS_INTERNAL_ADC
 - davis.h, [67](#)
- DAVIS_CONFIG_APS_HAS_QUAD_ROI
 - davis.h, [68](#)
- DAVIS_CONFIG_APS_NULL_SETTLE
 - davis.h, [68](#)
- DAVIS_CONFIG_APS_ORIENTATION_INFO
 - davis.h, [68](#)
- DAVIS_CONFIG_APS_RAMP_RESET
 - davis.h, [68](#)
- DAVIS_CONFIG_APS_RAMP_SHORT_RESET
 - davis.h, [68](#)
- DAVIS_CONFIG_APS_RESET_READ
 - davis.h, [68](#)
- DAVIS_CONFIG_APS_RESET_SETTLE
 - davis.h, [68](#)
- DAVIS_CONFIG_APS_ROW_SETTLE
 - davis.h, [68](#)
- DAVIS_CONFIG_APS_RUN
 - davis.h, [68](#)
- DAVIS_CONFIG_APS_SAMPLE_ENABLE
 - davis.h, [69](#)
- DAVIS_CONFIG_APS_SAMPLE_SETTLE
 - davis.h, [69](#)
- DAVIS_CONFIG_APS_SIZE_COLUMNS

- davis.h, [69](#)
- DAVIS_CONFIG_APS_SIZE_ROWS
 - davis.h, [69](#)
- DAVIS_CONFIG_APS_SNAPSHOT
 - davis.h, [69](#)
- DAVIS_CONFIG_APS_START_COLUMN_0
 - davis.h, [69](#)
- DAVIS_CONFIG_APS_START_COLUMN_1
 - davis.h, [69](#)
- DAVIS_CONFIG_APS_START_COLUMN_2
 - davis.h, [69](#)
- DAVIS_CONFIG_APS_START_COLUMN_3
 - davis.h, [70](#)
- DAVIS_CONFIG_APS_START_ROW_0
 - davis.h, [70](#)
- DAVIS_CONFIG_APS_START_ROW_1
 - davis.h, [70](#)
- DAVIS_CONFIG_APS_START_ROW_2
 - davis.h, [70](#)
- DAVIS_CONFIG_APS_START_ROW_3
 - davis.h, [70](#)
- DAVIS_CONFIG_APS_USE_INTERNAL_ADC
 - davis.h, [70](#)
- DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_START_LL
 - davis.h, [70](#)
- DAVIS_CONFIG_BIAS
 - davis.h, [70](#)
- DAVIS_CONFIG_CHIP
 - davis.h, [70](#)
- DAVIS_CONFIG_DVS
 - davis.h, [71](#)
- DAVIS_CONFIG_DVS_ACK_DELAY_COLUMN
 - davis.h, [71](#)
- DAVIS_CONFIG_DVS_ACK_DELAY_ROW
 - davis.h, [71](#)
- DAVIS_CONFIG_DVS_ACK_EXTENSION_COLUMN
 - davis.h, [71](#)
- DAVIS_CONFIG_DVS_ACK_EXTENSION_ROW
 - davis.h, [71](#)
- DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL
 - davis.h, [71](#)
- DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY
 - davis.h, [71](#)
- DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_DELTAT
 - davis.h, [71](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN
 - davis.h, [71](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW
 - davis.h, [72](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN
 - davis.h, [72](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW
 - davis.h, [72](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN
 - davis.h, [72](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW
 - davis.h, [72](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN
 - davis.h, [72](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW
 - davis.h, [72](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN
 - davis.h, [72](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW
 - davis.h, [72](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN
 - davis.h, [72](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW
 - davis.h, [73](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN
 - davis.h, [73](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW
 - davis.h, [73](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN
 - davis.h, [73](#)
- DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW
 - davis.h, [73](#)
- DAVIS_CONFIG_DVS_FILTER_ROW_ONLY_EVENT_START
 - davis.h, [73](#)
- DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER
 - davis.h, [73](#)
- DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER
 - davis.h, [73](#)
- DAVIS_CONFIG_DVS_HAS_TEST_EVENT_GENERATOR
 - davis.h, [73](#)
- DAVIS_CONFIG_DVS_ORIENTATION_INFO
 - davis.h, [74](#)
- DAVIS_CONFIG_DVS_RUN
 - davis.h, [74](#)
- DAVIS_CONFIG_DVS_SIZE_COLUMNS
 - davis.h, [74](#)
- DAVIS_CONFIG_DVS_SIZE_ROWS
 - davis.h, [74](#)
- DAVIS_CONFIG_DVS_TEST_EVENT_GENERATOR_ENABLE
 - davis.h, [74](#)
- DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_START_LL
 - davis.h, [74](#)
- DAVIS_CONFIG_EXTINPUT
 - davis.h, [74](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGE
 - davis.h, [74](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH
 - davis.h, [75](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY
 - davis.h, [75](#)

- DAVIS_CONFIG_EXTINPUT_DETECT_PULSES
davis.h, [75](#)
- DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGE
davis.h, [75](#)
- DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL
davis.h, [75](#)
- DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH
davis.h, [75](#)
- DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY
davis.h, [75](#)
- DAVIS_CONFIG_EXTINPUT_GENERATE_USE_CUSTOM_SIGNAL
davis.h, [75](#)
- DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR
davis.h, [76](#)
- DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR
davis.h, [76](#)
- DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR
davis.h, [76](#)
- DAVIS_CONFIG_IMU
davis.h, [76](#)
- DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE
davis.h, [76](#)
- DAVIS_CONFIG_IMU_ACCEL_STANDBY
davis.h, [76](#)
- DAVIS_CONFIG_IMU_DIGITAL_LOW_PASS_FILTER
davis.h, [76](#)
- DAVIS_CONFIG_IMU_GYRO_FULL_SCALE
davis.h, [76](#)
- DAVIS_CONFIG_IMU_GYRO_STANDBY
davis.h, [77](#)
- DAVIS_CONFIG_IMU_LP_CYCLE
davis.h, [77](#)
- DAVIS_CONFIG_IMU_LP_WAKEUP
davis.h, [77](#)
- DAVIS_CONFIG_IMU_RUN
davis.h, [77](#)
- DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER
davis.h, [77](#)
- DAVIS_CONFIG_IMU_TEMP_STANDBY
davis.h, [77](#)
- DAVIS_CONFIG_MUX
davis.h, [77](#)
- DAVIS_CONFIG_MUX_DROP_APS_ON_TRANSFER_STALL
davis.h, [77](#)
- DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL
davis.h, [77](#)
- DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL
davis.h, [78](#)
- DAVIS_CONFIG_MUX_DROP_IMU_ON_TRANSFER_STALL
davis.h, [78](#)
- DAVIS_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE
davis.h, [78](#)
- DAVIS_CONFIG_MUX_RUN
davis.h, [78](#)
- DAVIS_CONFIG_MUX_TIMESTAMP_RESET
davis.h, [78](#)
- DAVIS_CONFIG_MUX_TIMESTAMP_RUN
davis.h, [78](#)
- DAVIS_CONFIG_SYSINFO
davis.h, [78](#)
- DAVIS_CONFIG_SYSINFO_ADC_CLOCK
davis.h, [78](#)
- DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER
davis.h, [78](#)
- DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER
davis.h, [79](#)
- DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK
davis.h, [79](#)
- DAVIS_CONFIG_SYSINFO_LOGIC_VERSION
davis.h, [79](#)
- DAVIS_CONFIG_USB
davis.h, [79](#)
- DAVIS_CONFIG_USB_EARLY_PACKET_DELAY
davis.h, [79](#)
- DAVIS_CONFIG_USB_RUN
davis.h, [79](#)
- DAVISRGB_CONFIG_APS_GSFDRESET
davis.h, [79](#)
- DAVISRGB_CONFIG_APS_GSPDRESET
davis.h, [79](#)
- DAVISRGB_CONFIG_APS_GSRESETFALL
davis.h, [80](#)
- DAVISRGB_CONFIG_APS_GSTXFALL
davis.h, [80](#)
- DAVISRGB_CONFIG_APS_RSFDSETTLE
davis.h, [80](#)
- DAVISRGB_CONFIG_APS_TRANSFER
davis.h, [80](#)
- DAVISRGB_CONFIG_BIAS_ADCCOMPBP
davis.h, [80](#)
- DAVISRGB_CONFIG_BIAS_ADCREFHIGH
davis.h, [80](#)
- DAVISRGB_CONFIG_BIAS_ADCREFLOW
davis.h, [80](#)
- DAVISRGB_CONFIG_BIAS_ADCTESTVOLTAGE
davis.h, [81](#)
- DAVISRGB_CONFIG_BIAS_AEPDBN
davis.h, [81](#)
- DAVISRGB_CONFIG_BIAS_AEPUXBP
davis.h, [81](#)
- DAVISRGB_CONFIG_BIAS_AEPUYBP
davis.h, [81](#)
- DAVISRGB_CONFIG_BIAS_APSCAS
davis.h, [81](#)
- DAVISRGB_CONFIG_BIAS_APSROSFBN
davis.h, [82](#)

DAVISRGB_CONFIG_BIAS_ARRAYBIASBUFFERBN
davis.h, [82](#)

DAVISRGB_CONFIG_BIAS_ARRAYLOGICBUFFER↔
BN
davis.h, [82](#)

DAVISRGB_CONFIG_BIAS_BIASBUFFER
davis.h, [82](#)

DAVISRGB_CONFIG_BIAS_DACBUFBP
davis.h, [82](#)

DAVISRGB_CONFIG_BIAS_DIFFBN
davis.h, [83](#)

DAVISRGB_CONFIG_BIAS_FALLTIMEBN
davis.h, [83](#)

DAVISRGB_CONFIG_BIAS_GND07
davis.h, [83](#)

DAVISRGB_CONFIG_BIAS_IFREFRBN
davis.h, [83](#)

DAVISRGB_CONFIG_BIAS_IFTHRBN
davis.h, [83](#)

DAVISRGB_CONFIG_BIAS_LCOLTIMEOUTBN
davis.h, [84](#)

DAVISRGB_CONFIG_BIAS_LOCALBUFBN
davis.h, [84](#)

DAVISRGB_CONFIG_BIAS_OFFBN
davis.h, [84](#)

DAVISRGB_CONFIG_BIAS_ONBN
davis.h, [84](#)

DAVISRGB_CONFIG_BIAS_OVG1LO
davis.h, [84](#)

DAVISRGB_CONFIG_BIAS_OVG2LO
davis.h, [85](#)

DAVISRGB_CONFIG_BIAS_PADFOLLBN
davis.h, [85](#)

DAVISRGB_CONFIG_BIAS_PIXINBPN
davis.h, [85](#)

DAVISRGB_CONFIG_BIAS_PRBP
davis.h, [85](#)

DAVISRGB_CONFIG_BIAS_PRSFBP
davis.h, [85](#)

DAVISRGB_CONFIG_BIAS_READOUTBUFBP
davis.h, [86](#)

DAVISRGB_CONFIG_BIAS_REFRBP
davis.h, [86](#)

DAVISRGB_CONFIG_BIAS_RISETIMEBP
davis.h, [86](#)

DAVISRGB_CONFIG_BIAS_SSN
davis.h, [86](#)

DAVISRGB_CONFIG_BIAS_SSP
davis.h, [86](#)

DAVISRGB_CONFIG_BIAS_TX2OVG2HI
davis.h, [87](#)

DAVISRGB_CONFIG_CHIP_ADJUSTOVG1LO
davis.h, [87](#)

DAVISRGB_CONFIG_CHIP_ADJUSTOVG2LO
davis.h, [87](#)

DAVISRGB_CONFIG_CHIP_ADJUSTTX2OVG2HI
davis.h, [87](#)

DAVISRGB_CONFIG_CHIP_AERNAROW
davis.h, [87](#)

DAVISRGB_CONFIG_CHIP_ANALOGMUX0
davis.h, [87](#)

DAVISRGB_CONFIG_CHIP_ANALOGMUX1
davis.h, [87](#)

DAVISRGB_CONFIG_CHIP_ANALOGMUX2
davis.h, [88](#)

DAVISRGB_CONFIG_CHIP_BIASMUX0
davis.h, [88](#)

DAVISRGB_CONFIG_CHIP_DIGITALMUX0
davis.h, [88](#)

DAVISRGB_CONFIG_CHIP_DIGITALMUX1
davis.h, [88](#)

DAVISRGB_CONFIG_CHIP_DIGITALMUX2
davis.h, [88](#)

DAVISRGB_CONFIG_CHIP_DIGITALMUX3
davis.h, [88](#)

DAVISRGB_CONFIG_CHIP_RESETCALIBNEURON
davis.h, [88](#)

DAVISRGB_CONFIG_CHIP_RESETTESTPIXEL
davis.h, [88](#)

DAVISRGB_CONFIG_CHIP_SELECTGRAYCOUNT↔
ER
davis.h, [89](#)

DAVISRGB_CONFIG_CHIP_TESTADC
davis.h, [89](#)

DAVISRGB_CONFIG_CHIP_TYPCALIBNEURON
davis.h, [89](#)

DAVISRGB_CONFIG_CHIP_USEAOUT
davis.h, [89](#)

DOUBLE_DIODE
davis.h, [91](#)

DVS128_CONFIG_BIAS
dvs128.h, [93](#)

DVS128_CONFIG_BIAS_CAS
dvs128.h, [93](#)

DVS128_CONFIG_BIAS_DIFF
dvs128.h, [93](#)

DVS128_CONFIG_BIAS_DIFFOFF
dvs128.h, [94](#)

DVS128_CONFIG_BIAS_DIFFON
dvs128.h, [94](#)

DVS128_CONFIG_BIAS_FOLL
dvs128.h, [94](#)

DVS128_CONFIG_BIAS_INJGND
dvs128.h, [94](#)

DVS128_CONFIG_BIAS_PR
dvs128.h, [94](#)

DVS128_CONFIG_BIAS_PUX
dvs128.h, [94](#)

DVS128_CONFIG_BIAS_PUY
dvs128.h, [94](#)

DVS128_CONFIG_BIAS_REFR
dvs128.h, [94](#)

DVS128_CONFIG_BIAS_REQ
dvs128.h, [94](#)

DVS128_CONFIG_BIAS_REQPD
dvs128.h, [94](#)

- DVS128_CONFIG_DVS
 - dvs128.h, [95](#)
- DVS128_CONFIG_DVS_ARRAY_RESET
 - dvs128.h, [95](#)
- DVS128_CONFIG_DVS_RUN
 - dvs128.h, [95](#)
- DVS128_CONFIG_DVS_TIMESTAMP_RESET
 - dvs128.h, [95](#)
- DVS128_CONFIG_DVS_TS_MASTER
 - dvs128.h, [95](#)
- DVS_ROW_ONLY
 - special.h, [175](#)
- davis.h
 - CAER_DEVICE_DAVIS_FX2, [28](#)
 - CAER_DEVICE_DAVIS_FX3, [28](#)
 - caer_bias_shiftedsourcesource_operating_mode, [90](#)
 - caer_bias_shiftedsourcesource_voltage_level, [90](#)
 - caerBiasCoarseFineGenerate, [91](#)
 - caerBiasCoarseFineParse, [91](#)
 - caerBiasShiftedSourceGenerate, [91](#)
 - caerBiasShiftedSourceParse, [91](#)
 - caerBiasVDACGenerate, [92](#)
 - caerBiasVDACParse, [92](#)
 - caerDavisInfoGet, [92](#)
 - DAVIS128_CONFIG_BIAS_ADCCOMPBP, [28](#)
 - DAVIS128_CONFIG_BIAS_ADCREFHIGH, [28](#)
 - DAVIS128_CONFIG_BIAS_ADCREFLOW, [28](#)
 - DAVIS128_CONFIG_BIAS_AEPDBN, [28](#)
 - DAVIS128_CONFIG_BIAS_AEPUXBP, [28](#)
 - DAVIS128_CONFIG_BIAS_AEPUYBP, [29](#)
 - DAVIS128_CONFIG_BIAS_APSCAS, [29](#)
 - DAVIS128_CONFIG_BIAS_APSEVERFLOWLE←VEL, [29](#)
 - DAVIS128_CONFIG_BIAS_APSROSFBN, [29](#)
 - DAVIS128_CONFIG_BIAS_BIASBUFFER, [29](#)
 - DAVIS128_CONFIG_BIAS_COLSELOWBN, [30](#)
 - DAVIS128_CONFIG_BIAS_DACBUFBP, [30](#)
 - DAVIS128_CONFIG_BIAS_DIFFBN, [30](#)
 - DAVIS128_CONFIG_BIAS_IFREFRBN, [30](#)
 - DAVIS128_CONFIG_BIAS_IFTHRBN, [30](#)
 - DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN, [31](#)
 - DAVIS128_CONFIG_BIAS_LOCALBUFBN, [31](#)
 - DAVIS128_CONFIG_BIAS_OFFBN, [31](#)
 - DAVIS128_CONFIG_BIAS_ONBN, [31](#)
 - DAVIS128_CONFIG_BIAS_PADFOLLBN, [31](#)
 - DAVIS128_CONFIG_BIAS_PIXINBN, [32](#)
 - DAVIS128_CONFIG_BIAS_PRBP, [32](#)
 - DAVIS128_CONFIG_BIAS_PRSEFBP, [32](#)
 - DAVIS128_CONFIG_BIAS_READOUTBUFBP, [32](#)
 - DAVIS128_CONFIG_BIAS_REFRBP, [32](#)
 - DAVIS128_CONFIG_BIAS_SSN, [33](#)
 - DAVIS128_CONFIG_BIAS_SSP, [33](#)
 - DAVIS128_CONFIG_CHIP_AERNAROW, [33](#)
 - DAVIS128_CONFIG_CHIP_ANALOGMUX0, [33](#)
 - DAVIS128_CONFIG_CHIP_ANALOGMUX1, [33](#)
 - DAVIS128_CONFIG_CHIP_ANALOGMUX2, [33](#)
 - DAVIS128_CONFIG_CHIP_BIASMUX0, [34](#)
 - 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_RESETESTPIXEL, [34](#)
 - 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, [35](#)
 - DAVIS208_CONFIG_BIAS_AEPDBN, [35](#)
 - DAVIS208_CONFIG_BIAS_AEPUXBP, [36](#)
 - DAVIS208_CONFIG_BIAS_AEPUYBP, [36](#)
 - DAVIS208_CONFIG_BIAS_APSCAS, [36](#)
 - DAVIS208_CONFIG_BIAS_APSEVERFLOWLE←VEL, [36](#)
 - DAVIS208_CONFIG_BIAS_APSROSFBN, [36](#)
 - DAVIS208_CONFIG_BIAS_BIASBUFFER, [37](#)
 - DAVIS208_CONFIG_BIAS_COLSELOWBN, [37](#)
 - DAVIS208_CONFIG_BIAS_DACBUFBP, [37](#)
 - DAVIS208_CONFIG_BIAS_DIFFBN, [37](#)
 - DAVIS208_CONFIG_BIAS_IFREFRBN, [37](#)
 - DAVIS208_CONFIG_BIAS_IFTHRBN, [38](#)
 - DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN, [38](#)
 - DAVIS208_CONFIG_BIAS_LOCALBUFBN, [38](#)
 - DAVIS208_CONFIG_BIAS_OFFBN, [38](#)
 - DAVIS208_CONFIG_BIAS_ONBN, [38](#)
 - DAVIS208_CONFIG_BIAS_PADFOLLBN, [39](#)
 - DAVIS208_CONFIG_BIAS_PIXINBN, [39](#)
 - DAVIS208_CONFIG_BIAS_PRBP, [39](#)
 - DAVIS208_CONFIG_BIAS_PRSEFBP, [39](#)
 - DAVIS208_CONFIG_BIAS_READOUTBUFBP, [39](#)
 - DAVIS208_CONFIG_BIAS_REFRBP, [40](#)
 - DAVIS208_CONFIG_BIAS_REFSS, [40](#)
 - DAVIS208_CONFIG_BIAS_REFSSBN, [40](#)
 - DAVIS208_CONFIG_BIAS_REGBIASBP, [40](#)
 - DAVIS208_CONFIG_BIAS_RESETHIGHPASS, [40](#)
 - DAVIS208_CONFIG_BIAS_SSN, [41](#)
 - DAVIS208_CONFIG_BIAS_SSP, [41](#)
 - DAVIS208_CONFIG_CHIP_AERNAROW, [41](#)
 - DAVIS208_CONFIG_CHIP_ANALOGMUX0, [41](#)
 - DAVIS208_CONFIG_CHIP_ANALOGMUX1, [41](#)
 - DAVIS208_CONFIG_CHIP_ANALOGMUX2, [41](#)
 - DAVIS208_CONFIG_CHIP_BIASMUX0, [42](#)
 - DAVIS208_CONFIG_CHIP_DIGITALMUX0, [42](#)
 - DAVIS208_CONFIG_CHIP_DIGITALMUX1, [42](#)
 - DAVIS208_CONFIG_CHIP_DIGITALMUX2, [42](#)
 - DAVIS208_CONFIG_CHIP_DIGITALMUX3, [42](#)

- DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER, 42
- DAVIS208_CONFIG_CHIP_RESETCALIBNEU↵RON, 42
- DAVIS208_CONFIG_CHIP_RESETTESTPIXEL, 42
- DAVIS208_CONFIG_CHIP_SELECTBIASREFSS, 43
- DAVIS208_CONFIG_CHIP_SELECTGRAYCO↵UNTER, 43
- DAVIS208_CONFIG_CHIP_SELECTHIGHPASS, 43
- DAVIS208_CONFIG_CHIP_SELECTPOSFB, 43
- DAVIS208_CONFIG_CHIP_SELECTPREAMPA↵VG, 43
- DAVIS208_CONFIG_CHIP_SELECTSENSE, 43
- DAVIS208_CONFIG_CHIP_TYPENCALIBNEU↵RON, 43
- DAVIS208_CONFIG_CHIP_USEAOUT, 43
- DAVIS240_CONFIG_BIAS_AEPDBN, 43
- DAVIS240_CONFIG_BIAS_AEPUXBP, 44
- DAVIS240_CONFIG_BIAS_AEPUYBP, 44
- DAVIS240_CONFIG_BIAS_APSCASEPC, 44
- DAVIS240_CONFIG_BIAS_APSOEVERFLOWLE↵VELBN, 44
- DAVIS240_CONFIG_BIAS_APSROSFBN, 44
- DAVIS240_CONFIG_BIAS_BIASBUFFER, 45
- DAVIS240_CONFIG_BIAS_DIFFBN, 45
- DAVIS240_CONFIG_BIAS_DIFFCASBNC, 45
- DAVIS240_CONFIG_BIAS_IFREFRBN, 45
- DAVIS240_CONFIG_BIAS_IFTHRBN, 45
- DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN, 46
- DAVIS240_CONFIG_BIAS_LOCALBUFBN, 46
- DAVIS240_CONFIG_BIAS_OFFBN, 46
- DAVIS240_CONFIG_BIAS_ONBN, 46
- DAVIS240_CONFIG_BIAS_PADFOLLBN, 46
- DAVIS240_CONFIG_BIAS_PIXINBN, 47
- DAVIS240_CONFIG_BIAS_PRBP, 47
- DAVIS240_CONFIG_BIAS_PRSFBN, 47
- DAVIS240_CONFIG_BIAS_REFRBP, 47
- DAVIS240_CONFIG_BIAS_SSN, 47
- DAVIS240_CONFIG_BIAS_SSP, 48
- DAVIS240_CONFIG_CHIP_AERNAROW, 48
- DAVIS240_CONFIG_CHIP_ANALOGMUX0, 48
- DAVIS240_CONFIG_CHIP_ANALOGMUX1, 48
- DAVIS240_CONFIG_CHIP_ANALOGMUX2, 48
- DAVIS240_CONFIG_CHIP_BIASMUX0, 48
- DAVIS240_CONFIG_CHIP_DIGITALMUX0, 48
- DAVIS240_CONFIG_CHIP_DIGITALMUX1, 49
- DAVIS240_CONFIG_CHIP_DIGITALMUX2, 49
- DAVIS240_CONFIG_CHIP_DIGITALMUX3, 49
- DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER, 49
- DAVIS240_CONFIG_CHIP_RESETCALIBNEU↵RON, 49
- DAVIS240_CONFIG_CHIP_RESETTESTPIXEL, 49
- DAVIS240_CONFIG_CHIP_SPECIALPIXELCO↵NTROL, 49
- DAVIS240_CONFIG_CHIP_TYPENCALIBNEU↵RON, 50
- DAVIS240_CONFIG_CHIP_USEAOUT, 50
- DAVIS346_CONFIG_BIAS_ADCCOMPBP, 50
- DAVIS346_CONFIG_BIAS_ADCREFHIGH, 50
- DAVIS346_CONFIG_BIAS_ADCREFLOW, 50
- DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE, 50
- DAVIS346_CONFIG_BIAS_AEPDBN, 51
- DAVIS346_CONFIG_BIAS_AEPUXBP, 51
- DAVIS346_CONFIG_BIAS_AEPUYBP, 51
- DAVIS346_CONFIG_BIAS_APSCAS, 51
- DAVIS346_CONFIG_BIAS_APSOEVERFLOWLE↵VEL, 51
- DAVIS346_CONFIG_BIAS_APSROSFBN, 52
- DAVIS346_CONFIG_BIAS_BIASBUFFER, 52
- DAVIS346_CONFIG_BIAS_COLSELOWBN, 52
- DAVIS346_CONFIG_BIAS_DACBUFBP, 52
- DAVIS346_CONFIG_BIAS_DIFFBN, 52
- DAVIS346_CONFIG_BIAS_IFREFRBN, 53
- DAVIS346_CONFIG_BIAS_IFTHRBN, 53
- DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN, 53
- DAVIS346_CONFIG_BIAS_LOCALBUFBN, 53
- DAVIS346_CONFIG_BIAS_OFFBN, 53
- DAVIS346_CONFIG_BIAS_ONBN, 54
- DAVIS346_CONFIG_BIAS_PADFOLLBN, 54
- DAVIS346_CONFIG_BIAS_PIXINBN, 54
- DAVIS346_CONFIG_BIAS_PRBP, 54
- DAVIS346_CONFIG_BIAS_PRSFBN, 54
- DAVIS346_CONFIG_BIAS_READOUTBUFBP, 55
- DAVIS346_CONFIG_BIAS_REFRBP, 55
- DAVIS346_CONFIG_BIAS_SSN, 55
- DAVIS346_CONFIG_BIAS_SSP, 55
- DAVIS346_CONFIG_CHIP_AERNAROW, 55
- DAVIS346_CONFIG_CHIP_ANALOGMUX0, 56
- DAVIS346_CONFIG_CHIP_ANALOGMUX1, 56
- DAVIS346_CONFIG_CHIP_ANALOGMUX2, 56
- DAVIS346_CONFIG_CHIP_BIASMUX0, 56
- DAVIS346_CONFIG_CHIP_DIGITALMUX0, 56
- DAVIS346_CONFIG_CHIP_DIGITALMUX1, 56
- DAVIS346_CONFIG_CHIP_DIGITALMUX2, 56
- DAVIS346_CONFIG_CHIP_DIGITALMUX3, 56
- DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER, 57
- DAVIS346_CONFIG_CHIP_RESETCALIBNEU↵RON, 57
- DAVIS346_CONFIG_CHIP_RESETTESTPIXEL, 57
- DAVIS346_CONFIG_CHIP_SELECTGRAYCO↵UNTER, 57
- DAVIS346_CONFIG_CHIP_TESTADC, 57
- DAVIS346_CONFIG_CHIP_TYPENCALIBNEU↵RON, 57
- DAVIS346_CONFIG_CHIP_USEAOUT, 57
- DAVIS640_CONFIG_BIAS_ADCCOMPBP, 57

- DAVIS640_CONFIG_BIAS_ADCREFHIGH, 58
- DAVIS640_CONFIG_BIAS_ADCREFLOW, 58
- DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE, 58
- DAVIS640_CONFIG_BIAS_AEPDBN, 58
- DAVIS640_CONFIG_BIAS_AEPUXBP, 58
- DAVIS640_CONFIG_BIAS_AEPUYBP, 59
- DAVIS640_CONFIG_BIAS_APSCAS, 59
- DAVIS640_CONFIG_BIAS_APSOEVERFLOWLEVEL, 59
- DAVIS640_CONFIG_BIAS_APSROSFBN, 59
- DAVIS640_CONFIG_BIAS_BIASBUFFER, 59
- DAVIS640_CONFIG_BIAS_COLSELOWBN, 60
- DAVIS640_CONFIG_BIAS_DACBUFBP, 60
- DAVIS640_CONFIG_BIAS_DIFFBN, 60
- DAVIS640_CONFIG_BIAS_IFREFRBN, 60
- DAVIS640_CONFIG_BIAS_IFTHRBN, 60
- DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN, 61
- DAVIS640_CONFIG_BIAS_LOCALBUFBN, 61
- DAVIS640_CONFIG_BIAS_OFFBN, 61
- DAVIS640_CONFIG_BIAS_ONBN, 61
- DAVIS640_CONFIG_BIAS_PADFOLLBN, 61
- DAVIS640_CONFIG_BIAS_PIXINBN, 62
- DAVIS640_CONFIG_BIAS_PRBP, 62
- DAVIS640_CONFIG_BIAS_PRSFBP, 62
- DAVIS640_CONFIG_BIAS_READOUTBUFBP, 62
- DAVIS640_CONFIG_BIAS_REFRBP, 62
- DAVIS640_CONFIG_BIAS_SSN, 63
- DAVIS640_CONFIG_BIAS_SSP, 63
- DAVIS640_CONFIG_CHIP_AERNAROW, 63
- DAVIS640_CONFIG_CHIP_ANALOGMUX0, 63
- DAVIS640_CONFIG_CHIP_ANALOGMUX1, 63
- DAVIS640_CONFIG_CHIP_ANALOGMUX2, 63
- DAVIS640_CONFIG_CHIP_BIASMUX0, 64
- DAVIS640_CONFIG_CHIP_DIGITALMUX0, 64
- DAVIS640_CONFIG_CHIP_DIGITALMUX1, 64
- DAVIS640_CONFIG_CHIP_DIGITALMUX2, 64
- DAVIS640_CONFIG_CHIP_DIGITALMUX3, 64
- DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER, 64
- DAVIS640_CONFIG_CHIP_RESETCALIBNEURON, 64
- DAVIS640_CONFIG_CHIP_RESETTESTPIXEL, 64
- DAVIS640_CONFIG_CHIP_SELECTGRAYCONTROLLER, 65
- DAVIS640_CONFIG_CHIP_TESTADC, 65
- DAVIS640_CONFIG_CHIP_TYPENCALIBNEURON, 65
- DAVIS640_CONFIG_CHIP_USEAOUT, 65
- DAVIS_CHIP_DAVIS128, 65
- DAVIS_CHIP_DAVIS208, 65
- DAVIS_CHIP_DAVIS240A, 65
- DAVIS_CHIP_DAVIS240B, 65
- DAVIS_CHIP_DAVIS240C, 65
- DAVIS_CHIP_DAVIS346A, 65
- DAVIS_CHIP_DAVIS346B, 65
- DAVIS_CHIP_DAVIS346C, 66
- DAVIS_CHIP_DAVIS640, 66
- DAVIS_CHIP_DAVISRGB, 66
- DAVIS_CONFIG_APS, 66
- DAVIS_CONFIG_APS_ADC_TEST_MODE, 66
- DAVIS_CONFIG_APS_COLOR_FILTER, 66
- DAVIS_CONFIG_APS_COLUMN_SETTLE, 66
- DAVIS_CONFIG_APS_END_COLUMN_0, 66
- DAVIS_CONFIG_APS_END_COLUMN_1, 66
- DAVIS_CONFIG_APS_END_COLUMN_2, 66
- DAVIS_CONFIG_APS_END_COLUMN_3, 66
- DAVIS_CONFIG_APS_END_ROW_0, 67
- DAVIS_CONFIG_APS_END_ROW_1, 67
- DAVIS_CONFIG_APS_END_ROW_2, 67
- DAVIS_CONFIG_APS_END_ROW_3, 67
- DAVIS_CONFIG_APS_EXPOSURE, 67
- DAVIS_CONFIG_APS_FRAME_DELAY, 67
- DAVIS_CONFIG_APS_GLOBAL_SHUTTER, 67
- DAVIS_CONFIG_APS_HAS_EXTERNAL_ADC, 67
- DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER, 67
- DAVIS_CONFIG_APS_HAS_INTERNAL_ADC, 67
- DAVIS_CONFIG_APS_HAS_QUAD_ROI, 68
- DAVIS_CONFIG_APS_NULL_SETTLE, 68
- DAVIS_CONFIG_APS_ORIENTATION_INFO, 68
- DAVIS_CONFIG_APS_RAMP_RESET, 68
- DAVIS_CONFIG_APS_RAMP_SHORT_RESET, 68
- DAVIS_CONFIG_APS_RESET_READ, 68
- DAVIS_CONFIG_APS_RESET_SETTLE, 68
- DAVIS_CONFIG_APS_ROW_SETTLE, 68
- DAVIS_CONFIG_APS_RUN, 68
- DAVIS_CONFIG_APS_SAMPLE_ENABLE, 69
- DAVIS_CONFIG_APS_SAMPLE_SETTLE, 69
- DAVIS_CONFIG_APS_SIZE_COLUMNS, 69
- DAVIS_CONFIG_APS_SIZE_ROWS, 69
- DAVIS_CONFIG_APS_SNAPSHOT, 69
- DAVIS_CONFIG_APS_START_COLUMN_0, 69
- DAVIS_CONFIG_APS_START_COLUMN_1, 69
- DAVIS_CONFIG_APS_START_COLUMN_2, 69
- DAVIS_CONFIG_APS_START_COLUMN_3, 70
- DAVIS_CONFIG_APS_START_ROW_0, 70
- DAVIS_CONFIG_APS_START_ROW_1, 70
- DAVIS_CONFIG_APS_START_ROW_2, 70
- DAVIS_CONFIG_APS_START_ROW_3, 70
- DAVIS_CONFIG_APS_USE_INTERNAL_ADC, 70
- DAVIS_CONFIG_APS_WAIT_ON_TRANSFERSTALL, 70
- DAVIS_CONFIG_BIAS, 70
- DAVIS_CONFIG_CHIP, 70
- DAVIS_CONFIG_DVS, 71
- DAVIS_CONFIG_DVS_ACK_DELAY_COLUMN, 71
- DAVIS_CONFIG_DVS_ACK_DELAY_ROW, 71
- DAVIS_CONFIG_DVS_ACK_EXTENSION_COLUMN, 71

- DAVIS_CONFIG_DVS_ACK_EXTENSION_ROW, 71
- DAVIS_CONFIG_DVS_EXTERNAL_AER_CON←TROL, 71
- DAVIS_CONFIG_DVS_FILTER_BACKGROUN←D_ACTIVITY, 71
- DAVIS_CONFIG_DVS_FILTER_BACKGROUN←D_ACTIVITY_DELTAT, 71
- DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COL←UMN, 71
- DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW, 72
- DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COL←UMN, 72
- DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW, 72
- DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COL←UMN, 72
- DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW, 72
- DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COL←UMN, 72
- DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW, 72
- DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COL←UMN, 72
- DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW, 72
- DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COL←UMN, 72
- DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW, 73
- DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COL←UMN, 73
- DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW, 73
- DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COL←UMN, 73
- DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW, 73
- DAVIS_CONFIG_DVS_FILTER_ROW_ONLY←EVENTS, 73
- DAVIS_CONFIG_DVS_HAS_BACKGROUND←ACTIVITY_FILTER, 73
- DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER, 73
- DAVIS_CONFIG_DVS_HAS_TEST_EVENT_G←ENERATOR, 73
- DAVIS_CONFIG_DVS_ORIENTATION_INFO, 74
- DAVIS_CONFIG_DVS_RUN, 74
- DAVIS_CONFIG_DVS_SIZE_COLUMNS, 74
- DAVIS_CONFIG_DVS_SIZE_ROWS, 74
- DAVIS_CONFIG_DVS_TEST_EVENT_GENER←ATOR_ENABLE, 74
- DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER←_STALL, 74
- DAVIS_CONFIG_EXTINPUT, 74
- DAVIS_CONFIG_EXTINPUT_DETECT_FALLIN←G_EDGES, 74
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE←_LENGTH, 75
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSE←_POLARITY, 75
- DAVIS_CONFIG_EXTINPUT_DETECT_PULSES, 75
- DAVIS_CONFIG_EXTINPUT_DETECT_RISING←_EDGES, 75
- DAVIS_CONFIG_EXTINPUT_GENERATE_PUL←SE_INTERVAL, 75
- DAVIS_CONFIG_EXTINPUT_GENERATE_PUL←SE_LENGTH, 75
- DAVIS_CONFIG_EXTINPUT_GENERATE_PUL←SE_POLARITY, 75
- DAVIS_CONFIG_EXTINPUT_GENERATE_US←E_CUSTOM_SIGNAL, 75
- DAVIS_CONFIG_EXTINPUT_HAS_GENERAT←OR, 76
- DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR, 76
- DAVIS_CONFIG_EXTINPUT_RUN_GENERAT←OR, 76
- DAVIS_CONFIG_IMU, 76
- DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE, 76
- DAVIS_CONFIG_IMU_ACCEL_STANDBY, 76
- DAVIS_CONFIG_IMU_DIGITAL_LOW_PASS_F←ILTER, 76
- DAVIS_CONFIG_IMU_GYRO_FULL_SCALE, 76
- DAVIS_CONFIG_IMU_GYRO_STANDBY, 77
- DAVIS_CONFIG_IMU_LP_CYCLE, 77
- DAVIS_CONFIG_IMU_LP_WAKEUP, 77
- DAVIS_CONFIG_IMU_RUN, 77
- DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVID←ER, 77
- DAVIS_CONFIG_IMU_TEMP_STANDBY, 77
- DAVIS_CONFIG_MUX, 77
- DAVIS_CONFIG_MUX_DROP_APS_ON_TRA←NSFER_STALL, 77
- DAVIS_CONFIG_MUX_DROP_DVS_ON_TRA←NSFER_STALL, 77
- DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON←_TRANSFER_STALL, 78
- DAVIS_CONFIG_MUX_DROP_IMU_ON_TRAN←SFER_STALL, 78
- DAVIS_CONFIG_MUX_FORCE_CHIP_BIAS_E←NABLE, 78
- DAVIS_CONFIG_MUX_RUN, 78
- DAVIS_CONFIG_MUX_TIMESTAMP_RESET, 78
- DAVIS_CONFIG_MUX_TIMESTAMP_RUN, 78
- DAVIS_CONFIG_SYSINFO, 78
- DAVIS_CONFIG_SYSINFO_ADC_CLOCK, 78
- DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER, 78
- DAVIS_CONFIG_SYSINFO_DEVICE_IS_MAST←ER, 79
- DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK, 79
- DAVIS_CONFIG_SYSINFO_LOGIC_VERSION, 79

- DAVIS_CONFIG_USB, [79](#)
- DAVIS_CONFIG_USB_EARLY_PACKET_DELAY, [79](#)
- DAVIS_CONFIG_USB_RUN, [79](#)
- DAVISRGB_CONFIG_APS_GSFDRESET, [79](#)
- DAVISRGB_CONFIG_APS_GSPDRESET, [79](#)
- DAVISRGB_CONFIG_APS_GSRESETFALL, [80](#)
- DAVISRGB_CONFIG_APS_GSTXFALL, [80](#)
- DAVISRGB_CONFIG_APS_RSFDSETTLE, [80](#)
- DAVISRGB_CONFIG_APS_TRANSFER, [80](#)
- DAVISRGB_CONFIG_BIAS_ADCCOMPBP, [80](#)
- DAVISRGB_CONFIG_BIAS_ADCREFHIGH, [80](#)
- DAVISRGB_CONFIG_BIAS_ADCREFLOW, [80](#)
- DAVISRGB_CONFIG_BIAS_ADCTESTVOLTAGE, [81](#)
- DAVISRGB_CONFIG_BIAS_AEPDBN, [81](#)
- DAVISRGB_CONFIG_BIAS_AEPUXBP, [81](#)
- DAVISRGB_CONFIG_BIAS_AEPUYBP, [81](#)
- DAVISRGB_CONFIG_BIAS_APSCAS, [81](#)
- DAVISRGB_CONFIG_BIAS_APSROSFBN, [82](#)
- DAVISRGB_CONFIG_BIAS_ARRAYBIASBUFFER, [82](#)
- DAVISRGB_CONFIG_BIAS_ARRAYLOGICBUFFER, [82](#)
- DAVISRGB_CONFIG_BIAS_BIASBUFFER, [82](#)
- DAVISRGB_CONFIG_BIAS_DACBUFBP, [82](#)
- DAVISRGB_CONFIG_BIAS_DIFFBN, [83](#)
- DAVISRGB_CONFIG_BIAS_FALLTIMEBN, [83](#)
- DAVISRGB_CONFIG_BIAS_GND07, [83](#)
- DAVISRGB_CONFIG_BIAS_IFREFRBN, [83](#)
- DAVISRGB_CONFIG_BIAS_IFTHRBN, [83](#)
- DAVISRGB_CONFIG_BIAS_LCOLTIMEOUTBN, [84](#)
- DAVISRGB_CONFIG_BIAS_LOCALBUFBN, [84](#)
- DAVISRGB_CONFIG_BIAS_OFFBN, [84](#)
- DAVISRGB_CONFIG_BIAS_ONBN, [84](#)
- DAVISRGB_CONFIG_BIAS_OVG1LO, [84](#)
- DAVISRGB_CONFIG_BIAS_OVG2LO, [85](#)
- DAVISRGB_CONFIG_BIAS_PADFOLLBN, [85](#)
- DAVISRGB_CONFIG_BIAS_PIXINVBN, [85](#)
- DAVISRGB_CONFIG_BIAS_PRBP, [85](#)
- DAVISRGB_CONFIG_BIAS_PRSFBN, [85](#)
- DAVISRGB_CONFIG_BIAS_READOUTBUFBP, [86](#)
- DAVISRGB_CONFIG_BIAS_REFRBP, [86](#)
- DAVISRGB_CONFIG_BIAS_RISETIMEBP, [86](#)
- DAVISRGB_CONFIG_BIAS_SSN, [86](#)
- DAVISRGB_CONFIG_BIAS_SSP, [86](#)
- DAVISRGB_CONFIG_BIAS_TX2OVG2HI, [87](#)
- DAVISRGB_CONFIG_CHIP_ADJUSTOVG1LO, [87](#)
- DAVISRGB_CONFIG_CHIP_ADJUSTOVG2LO, [87](#)
- DAVISRGB_CONFIG_CHIP_ADJUSTTX2OVG2HI, [87](#)
- DAVISRGB_CONFIG_CHIP_AERNAROW, [87](#)
- DAVISRGB_CONFIG_CHIP_ANALOGMUX0, [87](#)
- DAVISRGB_CONFIG_CHIP_ANALOGMUX1, [87](#)
- DAVISRGB_CONFIG_CHIP_ANALOGMUX2, [88](#)
- DAVISRGB_CONFIG_CHIP_BIASMUX0, [88](#)
- DAVISRGB_CONFIG_CHIP_DIGITALMUX0, [88](#)
- DAVISRGB_CONFIG_CHIP_DIGITALMUX1, [88](#)
- DAVISRGB_CONFIG_CHIP_DIGITALMUX2, [88](#)
- DAVISRGB_CONFIG_CHIP_DIGITALMUX3, [88](#)
- DAVISRGB_CONFIG_CHIP_RESETCALIBNEURON, [88](#)
- DAVISRGB_CONFIG_CHIP_RESETTESTPIXEL, [88](#)
- DAVISRGB_CONFIG_CHIP_SELECTGRAYCONINTER, [89](#)
- DAVISRGB_CONFIG_CHIP_TESTADC, [89](#)
- DAVISRGB_CONFIG_CHIP_TYPENCALIBNEURON, [89](#)
- DAVISRGB_CONFIG_CHIP_USEAOUT, [89](#)
- DOUBLE_DIODE, [91](#)
- HI_Z, [90](#)
- IS_DAVIS128, [89](#)
- IS_DAVIS208, [89](#)
- IS_DAVIS240, [89](#)
- IS_DAVIS240A, [89](#)
- IS_DAVIS240B, [89](#)
- IS_DAVIS240C, [89](#)
- IS_DAVIS346, [90](#)
- IS_DAVIS346A, [90](#)
- IS_DAVIS346B, [90](#)
- IS_DAVIS346C, [90](#)
- IS_DAVIS640, [90](#)
- IS_DAVISRGB, [90](#)
- SHIFTED_SOURCE, [90](#)
- SINGLE_DIODE, [91](#)
- SPLIT_GATE, [91](#)
- TIED_TO_RAIL, [90](#)
- devices/davis.h, [19](#)
- devices/dvs128.h, [92](#)
- devices/usb.h, [95](#)
- dvs128.h
 - CAER_DEVICE_DVS128, [93](#)
 - caerDVS128InfoGet, [95](#)
 - DVS128_CONFIG_BIAS, [93](#)
 - DVS128_CONFIG_BIAS_CAS, [93](#)
 - DVS128_CONFIG_BIAS_DIFF, [93](#)
 - DVS128_CONFIG_BIAS_DIFFOFF, [94](#)
 - DVS128_CONFIG_BIAS_DIFFON, [94](#)
 - DVS128_CONFIG_BIAS_FOLL, [94](#)
 - DVS128_CONFIG_BIAS_INJGND, [94](#)
 - DVS128_CONFIG_BIAS_PR, [94](#)
 - DVS128_CONFIG_BIAS_PUX, [94](#)
 - DVS128_CONFIG_BIAS_PUY, [94](#)
 - DVS128_CONFIG_BIAS_REFR, [94](#)
 - DVS128_CONFIG_BIAS_REQ, [94](#)
 - DVS128_CONFIG_BIAS_REQPD, [94](#)
 - DVS128_CONFIG_DVS, [95](#)
 - DVS128_CONFIG_DVS_ARRAY_RESET, [95](#)
 - DVS128_CONFIG_DVS_RUN, [95](#)
 - DVS128_CONFIG_DVS_TIMESTAMP_RESET, [95](#)

- DVS128_CONFIG_DVS_TS_MASTER, 95
- EAR_EVENT
 - common.h, 105
- EAR_MASK
 - ear.h, 118
- EAR_SHIFT
 - ear.h, 118
- EXTERNAL_INPUT_FALLING_EDGE
 - special.h, 175
- EXTERNAL_INPUT_PULSE
 - special.h, 175
- EXTERNAL_INPUT_RISING_EDGE
 - special.h, 175
- ear.h
 - CAER_EAR_ITERATOR_ALL_END, 117
 - CAER_EAR_ITERATOR_ALL_START, 117
 - CAER_EAR_ITERATOR_VALID_END, 118
 - CAER_EAR_ITERATOR_VALID_START, 118
 - CHANNEL_MASK, 118
 - CHANNEL_SHIFT, 118
 - caerEarEvent, 119
 - caerEarEventGetChannel, 119
 - caerEarEventGetEar, 119
 - caerEarEventGetTimestamp, 121
 - caerEarEventGetTimestamp64, 121
 - caerEarEventInvalidate, 121
 - caerEarEventsValid, 121
 - caerEarEventPacket, 119
 - caerEarEventPacketAllocate, 122
 - caerEarEventPacketGetEvent, 122
 - caerEarEventSetChannel, 122
 - caerEarEventSetEar, 122
 - caerEarEventSetTimestamp, 123
 - caerEarEventValidate, 123
 - EAR_MASK, 118
 - EAR_SHIFT, 118
 - FILTER_MASK, 118
 - FILTER_SHIFT, 119
 - NEURON_MASK, 119
 - NEURON_SHIFT, 119
- events/common.h, 101
- events/config.h, 110
- events/ear.h, 116
- events/frame.h, 123
- events/imu6.h, 140
- events/imu9.h, 147
- events/packetContainer.h, 157
- events/polarity.h, 159
- events/sample.h, 166
- events/special.h, 172
- FILTER_MASK
 - ear.h, 118
- FILTER_SHIFT
 - ear.h, 119
- FRAME_EVENT
 - common.h, 105
- frame.h
 - CAER_FRAME_ITERATOR_ALL_END, 126
 - CAER_FRAME_ITERATOR_ALL_START, 126
 - CAER_FRAME_ITERATOR_VALID_END, 126
 - CAER_FRAME_ITERATOR_VALID_START, 126
 - COLOR_CHANNELS_MASK, 126
 - COLOR_CHANNELS_SHIFT, 126
 - COLOR_FILTER_MASK, 126
 - COLOR_FILTER_SHIFT, 127
 - caer_frame_event_color_channels, 127
 - caer_frame_event_color_filter, 127
 - caerFrameEvent, 127
 - caerFrameEventGetChannelNumber, 128
 - caerFrameEventGetColorFilter, 128
 - caerFrameEventGetExposureLength, 128
 - caerFrameEventGetLengthX, 128
 - caerFrameEventGetLengthY, 129
 - caerFrameEventGetPixel, 129
 - caerFrameEventGetPixelFormatCGFormat, 129
 - caerFrameEventGetPixelFormatUnsafe, 129
 - caerFrameEventGetPixelForChannel, 130
 - caerFrameEventGetPixelForChannelUnsafe, 130
 - caerFrameEventGetPixelUnsafe, 131
 - caerFrameEventGetPixelsMaxIndex, 130
 - caerFrameEventGetPixelsSize, 130
 - caerFrameEventGetPositionX, 131
 - caerFrameEventGetPositionY, 131
 - caerFrameEventGetROIIdentifier, 131
 - caerFrameEventGetTSEndOfExposure, 132
 - caerFrameEventGetTSEndOfExposure64, 132
 - caerFrameEventGetTSEndOfFrame, 133
 - caerFrameEventGetTSEndOfFrame64, 133
 - caerFrameEventGetTSSStartOfExposure, 133
 - caerFrameEventGetTSSStartOfExposure64, 133
 - caerFrameEventGetTSSStartOfFrame, 135
 - caerFrameEventGetTSSStartOfFrame64, 135
 - caerFrameEventGetTimestamp, 132
 - caerFrameEventGetTimestamp64, 132
 - caerFrameEventInvalidate, 135
 - caerFrameEventsValid, 135
 - caerFrameEventPacket, 127
 - caerFrameEventPacketAllocate, 136
 - caerFrameEventPacketGetEvent, 136
 - caerFrameEventPacketGetPixelsMaxIndex, 136
 - caerFrameEventPacketGetPixelsSize, 137
 - caerFrameEventSetColorFilter, 137
 - caerFrameEventSetLengthXLengthYChannel↵
Number, 137
 - caerFrameEventSetPixel, 137
 - caerFrameEventSetPixelForChannel, 138
 - caerFrameEventSetPixelForChannelUnsafe, 138
 - caerFrameEventSetPixelUnsafe, 138
 - caerFrameEventSetPositionX, 138
 - caerFrameEventSetPositionY, 139
 - caerFrameEventSetROIIdentifier, 139
 - caerFrameEventSetTSEndOfExposure, 139
 - caerFrameEventSetTSEndOfFrame, 139
 - caerFrameEventSetTSSStartOfExposure, 139
 - caerFrameEventSetTSSStartOfFrame, 140

- caerFrameEventValidate, [140](#)
- GRAYSCALE, [127](#)
- MONO, [128](#)
- pixels, [140](#)
- RGB, [127](#)
- RGBA, [127](#)
- RGBG, [128](#)
- RGBW, [128](#)
- ROI_IDENTIFIER_MASK, [127](#)
- ROI_IDENTIFIER_SHIFT, [127](#)
- GET_NUMBITS16
 - libcaer.h, [179](#)
- GET_NUMBITS32
 - libcaer.h, [179](#)
- GET_NUMBITS8
 - libcaer.h, [179](#)
- GRAYSCALE
 - frame.h, [127](#)
- HI_Z
 - davis.h, [90](#)
- I16T
 - libcaer.h, [180](#)
- I32T
 - libcaer.h, [180](#)
- I64T
 - libcaer.h, [180](#)
- I8T
 - libcaer.h, [180](#)
- IMU6_EVENT
 - common.h, [105](#)
- IMU9_EVENT
 - common.h, [105](#)
- IS_DAVIS128
 - davis.h, [89](#)
- IS_DAVIS208
 - davis.h, [89](#)
- IS_DAVIS240
 - davis.h, [89](#)
- IS_DAVIS240A
 - davis.h, [89](#)
- IS_DAVIS240B
 - davis.h, [89](#)
- IS_DAVIS240C
 - davis.h, [89](#)
- IS_DAVIS346
 - davis.h, [90](#)
- IS_DAVIS346A
 - davis.h, [90](#)
- IS_DAVIS346B
 - davis.h, [90](#)
- IS_DAVIS346C
 - davis.h, [90](#)
- IS_DAVIS640
 - davis.h, [90](#)
- IS_DAVISRGB
 - davis.h, [90](#)
- imu6.h
 - CAER_IMU6_ITERATOR_ALL_END, [142](#)
 - CAER_IMU6_ITERATOR_ALL_START, [142](#)
 - CAER_IMU6_ITERATOR_VALID_END, [142](#)
 - CAER_IMU6_ITERATOR_VALID_START, [142](#)
 - caerIMU6Event, [142](#)
 - caerIMU6EventGetAccelX, [143](#)
 - caerIMU6EventGetAccelY, [143](#)
 - caerIMU6EventGetAccelZ, [143](#)
 - caerIMU6EventGetGyroX, [143](#)
 - caerIMU6EventGetGyroY, [143](#)
 - caerIMU6EventGetGyroZ, [144](#)
 - caerIMU6EventGetTemp, [144](#)
 - caerIMU6EventGetTimestamp, [144](#)
 - caerIMU6EventGetTimestamp64, [144](#)
 - caerIMU6EventInvalidate, [145](#)
 - caerIMU6EventIsValid, [145](#)
 - caerIMU6EventPacket, [142](#)
 - caerIMU6EventPacketAllocate, [145](#)
 - caerIMU6EventPacketGetEvent, [145](#)
 - caerIMU6EventSetAccelX, [146](#)
 - caerIMU6EventSetAccelY, [146](#)
 - caerIMU6EventSetAccelZ, [146](#)
 - caerIMU6EventSetGyroX, [146](#)
 - caerIMU6EventSetGyroY, [146](#)
 - caerIMU6EventSetGyroZ, [147](#)
 - caerIMU6EventSetTemp, [147](#)
 - caerIMU6EventSetTimestamp, [147](#)
 - caerIMU6EventValidate, [147](#)
- imu9.h
 - CAER_IMU9_ITERATOR_ALL_END, [149](#)
 - CAER_IMU9_ITERATOR_ALL_START, [149](#)
 - CAER_IMU9_ITERATOR_VALID_END, [149](#)
 - CAER_IMU9_ITERATOR_VALID_START, [149](#)
 - caerIMU9Event, [150](#)
 - caerIMU9EventGetAccelX, [150](#)
 - caerIMU9EventGetAccelY, [150](#)
 - caerIMU9EventGetAccelZ, [150](#)
 - caerIMU9EventGetCompX, [151](#)
 - caerIMU9EventGetCompY, [151](#)
 - caerIMU9EventGetCompZ, [151](#)
 - caerIMU9EventGetGyroX, [151](#)
 - caerIMU9EventGetGyroY, [151](#)
 - caerIMU9EventGetGyroZ, [153](#)
 - caerIMU9EventGetTemp, [153](#)
 - caerIMU9EventGetTimestamp, [153](#)
 - caerIMU9EventGetTimestamp64, [153](#)
 - caerIMU9EventInvalidate, [154](#)
 - caerIMU9EventIsValid, [154](#)
 - caerIMU9EventPacket, [150](#)
 - caerIMU9EventPacketAllocate, [154](#)
 - caerIMU9EventPacketGetEvent, [154](#)
 - caerIMU9EventSetAccelX, [155](#)
 - caerIMU9EventSetAccelY, [155](#)
 - caerIMU9EventSetAccelZ, [155](#)
 - caerIMU9EventSetCompX, [155](#)
 - caerIMU9EventSetCompY, [155](#)
 - caerIMU9EventSetCompZ, [156](#)

- caerIMU9EventSetGyroX, 156
- caerIMU9EventSetGyroY, 156
- caerIMU9EventSetGyroZ, 156
- caerIMU9EventSetTemp, 156
- caerIMU9EventSetTimestamp, 156
- caerIMU9EventValidate, 157
- libcaer.h, 178
 - CLEAR_NUMBITS16, 179
 - CLEAR_NUMBITS32, 179
 - CLEAR_NUMBITS8, 179
 - caerByteArrayToInteger, 181
 - caerIntegerToByteArray, 181
 - caerStrEquals, 181
 - caerStrEqualsUpTo, 181
 - GET_NUMBITS16, 179
 - GET_NUMBITS32, 179
 - GET_NUMBITS8, 179
 - I16T, 180
 - I32T, 180
 - I64T, 180
 - I8T, 180
 - MASK_NUMBITS32, 180
 - MASK_NUMBITS64, 180
 - SET_NUMBITS16, 180
 - SET_NUMBITS32, 180
 - SET_NUMBITS8, 180
 - SWAP_VAR, 180
 - U16T, 180
 - U32T, 180
 - U64T, 181
 - U8T, 181
- log.h, 183
 - CAER_LOG_ALERT, 183
 - CAER_LOG_CRITICAL, 183
 - CAER_LOG_DEBUG, 183
 - CAER_LOG_EMERGENCY, 184
 - CAER_LOG_ERROR, 184
 - CAER_LOG_INFO, 184
 - CAER_LOG_NOTICE, 184
 - CAER_LOG_WARNING, 184
 - caerLog, 184
 - caerLogFileDescriptorsSet, 185
 - caerLogLevelGet, 185
 - caerLogLevelSet, 185
- MASK_NUMBITS32
 - libcaer.h, 180
- MASK_NUMBITS64
 - libcaer.h, 180
- MODULE_ADDR_MASK
 - config.h, 112
- MODULE_ADDR_SHIFT
 - config.h, 112
- MONO
 - frame.h, 128
- NEURON_MASK
 - ear.h, 119
- NEURON_SHIFT
 - ear.h, 119
- POLARITY_EVENT
 - common.h, 104
- POLARITY_MASK
 - polarity.h, 161
- POLARITY_SHIFT
 - polarity.h, 161
- packetContainer.h
 - caerEventPacketContainer, 158
 - caerEventPacketContainerAllocate, 158
 - caerEventPacketContainerFree, 158
 - caerEventPacketContainerGetEventPacket, 158
 - caerEventPacketContainerGetEventPackets↔
 - Number, 158
 - caerEventPacketContainerSetEventPacket, 159
 - caerEventPacketContainerSetEventPackets↔
 - Number, 159
- pixels
 - caer_frame_event, 11
 - frame.h, 140
- polarity.h
 - CAER_POLARITY_ITERATOR_ALL_END, 160
 - CAER_POLARITY_ITERATOR_ALL_START, 160
 - CAER_POLARITY_ITERATOR_VALID_END, 161
 - CAER_POLARITY_ITERATOR_VALID_START, 161
 - caerPolarityEvent, 162
 - caerPolarityEventGetPolarity, 162
 - caerPolarityEventGetTimestamp, 162
 - caerPolarityEventGetTimestamp64, 162
 - caerPolarityEventGetX, 163
 - caerPolarityEventGetY, 163
 - caerPolarityEventInvalidate, 163
 - caerPolarityEventIsValid, 163
 - caerPolarityEventPacket, 162
 - caerPolarityEventPacketAllocate, 163
 - caerPolarityEventPacketGetEvent, 165
 - caerPolarityEventSetPolarity, 165
 - caerPolarityEventSetTimestamp, 165
 - caerPolarityEventSetX, 165
 - caerPolarityEventSetY, 165
 - caerPolarityEventValidate, 166
 - POLARITY_MASK, 161
 - POLARITY_SHIFT, 161
 - X_ADDR_MASK, 161
 - X_ADDR_SHIFT, 161
 - Y_ADDR_MASK, 161
 - Y_ADDR_SHIFT, 162
- portable_endian.h, 185
- RGB
 - frame.h, 127
- RGBA
 - frame.h, 127
- RGBG
 - frame.h, 128
- RGBW

- frame.h, 128
- ROI_IDENTIFIER_MASK
 - frame.h, 127
- ROI_IDENTIFIER_SHIFT
 - frame.h, 127
- SAMPLE_EVENT
 - common.h, 105
- SAMPLE_MASK
 - sample.h, 168
- SAMPLE_SHIFT
 - sample.h, 168
- SAMPLE_TYPE_MASK
 - sample.h, 168
- SAMPLE_TYPE_SHIFT
 - sample.h, 168
- SET_NUMBITS16
 - libcaer.h, 180
- SET_NUMBITS32
 - libcaer.h, 180
- SET_NUMBITS8
 - libcaer.h, 180
- SHIFTED_SOURCE
 - davis.h, 90
- SINGLE_DIODE
 - davis.h, 91
- SPECIAL_EVENT
 - common.h, 104
- SPLIT_GATE
 - davis.h, 91
- SWAP_VAR
 - libcaer.h, 180
- sample.h
 - CAER_SAMPLE_ITERATOR_ALL_END, 167
 - CAER_SAMPLE_ITERATOR_ALL_START, 167
 - CAER_SAMPLE_ITERATOR_VALID_END, 167
 - CAER_SAMPLE_ITERATOR_VALID_START, 167
 - caerSampleEvent, 168
 - caerSampleEventGetSample, 168
 - caerSampleEventGetTimestamp, 170
 - caerSampleEventGetTimestamp64, 170
 - caerSampleEventGetType, 170
 - caerSampleEventInvalidate, 170
 - caerSampleEventIsValid, 171
 - caerSampleEventPacket, 168
 - caerSampleEventPacketAllocate, 171
 - caerSampleEventPacketGetEvent, 171
 - caerSampleEventSetSample, 171
 - caerSampleEventSetTimestamp, 172
 - caerSampleEventSetType, 172
 - caerSampleEventValidate, 172
 - SAMPLE_MASK, 168
 - SAMPLE_SHIFT, 168
 - SAMPLE_TYPE_MASK, 168
 - SAMPLE_TYPE_SHIFT, 168
- special.h
 - CAER_SPECIAL_ITERATOR_ALL_END, 174
 - CAER_SPECIAL_ITERATOR_ALL_START, 174
 - CAER_SPECIAL_ITERATOR_VALID_END, 174
 - CAER_SPECIAL_ITERATOR_VALID_START, 174
 - caer_special_event_types, 175
 - caerSpecialEvent, 175
 - caerSpecialEventGetData, 175
 - caerSpecialEventGetTimestamp, 175
 - caerSpecialEventGetTimestamp64, 176
 - caerSpecialEventGetType, 176
 - caerSpecialEventInvalidate, 176
 - caerSpecialEventIsValid, 176
 - caerSpecialEventPacket, 175
 - caerSpecialEventPacketAllocate, 177
 - caerSpecialEventPacketGetEvent, 177
 - caerSpecialEventSetData, 177
 - caerSpecialEventSetTimestamp, 177
 - caerSpecialEventSetType, 178
 - caerSpecialEventValidate, 178
 - DATA_MASK, 174
 - DATA_SHIFT, 174
 - DVS_ROW_ONLY, 175
 - EXTERNAL_INPUT_FALLING_EDGE, 175
 - EXTERNAL_INPUT_PULSE, 175
 - EXTERNAL_INPUT_RISING_EDGE, 175
 - TIMESTAMP_RESET, 175
 - TIMESTAMP_WRAP, 175
 - TYPE_MASK, 174
 - TYPE_SHIFT, 174
- TIED_TO_RAIL
 - davis.h, 90
- TIMESTAMP_RESET
 - special.h, 175
- TIMESTAMP_WRAP
 - special.h, 175
- TS_OVERFLOW_SHIFT
 - common.h, 104
- TYPE_MASK
 - special.h, 174
- TYPE_SHIFT
 - special.h, 174
- U16T
 - libcaer.h, 180
- U32T
 - libcaer.h, 180
- U64T
 - libcaer.h, 181
- U8T
 - libcaer.h, 181
- usb.h
 - CAER_HOST_CONFIG_DATAEXCHANGE, 96
 - CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING, 96
 - CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE, 97
 - CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS, 97
 - CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS, 97

CAER_HOST_CONFIG_PACKETS, [97](#)
 CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL, [97](#)
 CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_SIZE, [97](#)
 CAER_HOST_CONFIG_PACKETS_MAX_FRAME_INTERVAL, [97](#)
 CAER_HOST_CONFIG_PACKETS_MAX_FRAME_SIZE, [97](#)
 CAER_HOST_CONFIG_PACKETS_MAX_IMU6_INTERVAL, [98](#)
 CAER_HOST_CONFIG_PACKETS_MAX_IMU6_SIZE, [98](#)
 CAER_HOST_CONFIG_PACKETS_MAX_POLARITY_INTERVAL, [98](#)
 CAER_HOST_CONFIG_PACKETS_MAX_POLARITY_SIZE, [98](#)
 CAER_HOST_CONFIG_PACKETS_MAX_SPECTRAL_INTERVAL, [98](#)
 CAER_HOST_CONFIG_PACKETS_MAX_SPECTRAL_SIZE, [98](#)
 CAER_HOST_CONFIG_USB, [98](#)
 CAER_HOST_CONFIG_USB_BUFFER_NUMBER, [98](#)
 CAER_HOST_CONFIG_USB_BUFFER_SIZE, [98](#)
 caerDeviceClose, [99](#)
 caerDeviceConfigGet, [99](#)
 caerDeviceConfigSet, [99](#)
 caerDeviceDataGet, [100](#)
 caerDeviceDataStart, [100](#)
 caerDeviceDataStop, [100](#)
 caerDeviceHandle, [99](#)
 caerDeviceOpen, [101](#)
 caerDeviceSendDefaultConfig, [101](#)

 VALID_MARK_MASK
 common.h, [104](#)
 VALID_MARK_SHIFT
 common.h, [104](#)

 X_ADDR_MASK
 polarity.h, [161](#)
 X_ADDR_SHIFT
 polarity.h, [161](#)

 Y_ADDR_MASK
 polarity.h, [161](#)
 Y_ADDR_SHIFT
 polarity.h, [162](#)