libcaer

3.0.0

Generated by Doxygen 1.8.14

Contents

1 Data Structure Index 1

1.1 Data Structures . . . 1

2 File Index 3

2.1 File List . . . 3

3 Data Structure Documentation 5

3.1 caer_bias_coarsefine Struct Reference . . . 5

3.1.1 Detailed Description . . . 5

3.2 caer_bias_dynapse Struct Reference . . . 5

3.2.1 Detailed Description . . . 6

3.3 caer_bias_shiftedsource Struct Reference . . . 6

3.3.1 Detailed Description . . . 6

3.4 caer_bias_vdac Struct Reference . . . 7

3.4.1 Detailed Description . . . 7

3.5 caer_davis_info Struct Reference . . . 7

3.5.1 Detailed Description . . . 8

3.5.2 Field Documentation . . . 8

3.5.2.1 deviceString . . . 8

3.6 caer_device_discovery_result Struct Reference . . . 9

3.6.1 Detailed Description . . . 9

3.7 caer_dvs128_info Struct Reference . . . 9

3.7.1 Detailed Description . . . 10

3.7.2 Field Documentation . . . 10

3.7.2.1 deviceString . . . 10

3.8 caer_dynapse_info Struct Reference . . . 10

3.8.1 Detailed Description . . . 11

3.8.2 Field Documentation . . . 11

3.8.2.1 deviceString . . . 11

3.9 caer_edvs_info Struct Reference . . . 11

3.9.1 Detailed Description . . . 11

3.9.2 Field Documentation . . . 12

3.9.2.1 deviceString . . . 12

3.10 caer_filter_dvs_pixel Struct Reference . . . 12

3.10.1 Detailed Description . . . 12

4 File Documentation **13**

4.1 devices/davis.h File Reference 13

4.1.1 Detailed Description 22

4.1.2 Macro Definition Documentation 23

4.1.2.1 CAER_DEVICE_DAVIS 23

4.1.2.2 CAER_DEVICE_DAVIS_FX2 23

4.1.2.3 CAER_DEVICE_DAVIS_FX3 23

4.1.2.4 CAER_DEVICE_DAVIS_RPI 23

4.1.2.5 DAVIS128_CONFIG_BIAS_ADCCOMPBP 23

4.1.2.6 DAVIS128_CONFIG_BIAS_ADCREFHIGH 24

4.1.2.7 DAVIS128_CONFIG_BIAS_ADCREFLOW 24

4.1.2.8 DAVIS128_CONFIG_BIAS_AEPDBN 24

4.1.2.9 DAVIS128_CONFIG_BIAS_AEPUXBP 25

4.1.2.10 DAVIS128_CONFIG_BIAS_AEPUYBP 25

4.1.2.11 DAVIS128_CONFIG_BIAS_APSCAS 25

4.1.2.12 DAVIS128_CONFIG_BIAS_APSOVERFLOWLEVEL 26

4.1.2.13 DAVIS128_CONFIG_BIAS_APSROSFBN 26

4.1.2.14 DAVIS128_CONFIG_BIAS_BIASBUFFER 26

4.1.2.15 DAVIS128_CONFIG_BIAS_COLSELLOWBN 27

4.1.2.16 DAVIS128_CONFIG_BIAS_DACBUFBP 27

4.1.2.17 DAVIS128_CONFIG_BIAS_DIFFBN 27

4.1.2.18 DAVIS128_CONFIG_BIAS_IFREFRBN 28

4.1.2.19 DAVIS128_CONFIG_BIAS_IFTHRBN 28

4.1.2.20 DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN 28

4.1.2.21 DAVIS128_CONFIG_BIAS_LOCALBUFBN 29

4.1.2.22 DAVIS128_CONFIG_BIAS_OFFBN 29

4.1.2.23 DAVIS128_CONFIG_BIAS_ONBN 29

4.1.2.24 DAVIS128_CONFIG_BIAS_PADFOLLBN 30

4.1.2.25 DAVIS128_CONFIG_BIAS_PIXINVBN 30

4.1.2.26 DAVIS128_CONFIG_BIAS_PRBP 30

4.1.2.27 DAVIS128_CONFIG_BIAS_PRSFBP 31
4.1.2.28 DAVIS128_CONFIG_BIAS_READOUTBUFBP 31
4.1.2.29 DAVIS128_CONFIG_BIAS_REFRBP 31
4.1.2.30 DAVIS128_CONFIG_BIAS_SSN 32
4.1.2.31 DAVIS128_CONFIG_BIAS_SSP 32
4.1.2.32 DAVIS128_CONFIG_CHIP_AERNAROW 32
4.1.2.33 DAVIS128_CONFIG_CHIP_ANALOGMUX0 32
4.1.2.34 DAVIS128_CONFIG_CHIP_ANALOGMUX1 33
4.1.2.35 DAVIS128_CONFIG_CHIP_ANALOGMUX2 33
4.1.2.36 DAVIS128_CONFIG_CHIP_BIASMUX0 33
4.1.2.37 DAVIS128_CONFIG_CHIP_DIGITALMUX0 33
4.1.2.38 DAVIS128_CONFIG_CHIP_DIGITALMUX1 33
4.1.2.39 DAVIS128_CONFIG_CHIP_DIGITALMUX2 33
4.1.2.40 DAVIS128_CONFIG_CHIP_DIGITALMUX3 34
4.1.2.41 DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER 34
4.1.2.42 DAVIS128_CONFIG_CHIP_RESETCALIBNEURON 34
4.1.2.43 DAVIS128_CONFIG_CHIP_RESETTESTPIXEL 34
4.1.2.44 DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER 34
4.1.2.45 DAVIS128_CONFIG_CHIP_TYPENCALIBNEURON 34
4.1.2.46 DAVIS128_CONFIG_CHIP_USEAOUT 35
4.1.2.47 DAVIS208_CONFIG_BIAS_ADCCOMPBP 35
4.1.2.48 DAVIS208_CONFIG_BIAS_ADCREFHIGH 35
4.1.2.49 DAVIS208_CONFIG_BIAS_ADCREFLOW 35
4.1.2.50 DAVIS208_CONFIG_BIAS_AEPDBN 36
4.1.2.51 DAVIS208_CONFIG_BIAS_AEPUXBP 36
4.1.2.52 DAVIS208_CONFIG_BIAS_AEPUYBP 36
4.1.2.53 DAVIS208_CONFIG_BIAS_APSCAS 37
4.1.2.54 DAVIS208_CONFIG_BIAS_APSOVERFLOWLEVEL 37
4.1.2.55 DAVIS208_CONFIG_BIAS_APSROSFBN 37
4.1.2.56 DAVIS208_CONFIG_BIAS_BIASBUFFER 38

4.1.2.57 DAVIS208_CONFIG_BIAS_COLSELLOWBN 38
4.1.2.58 DAVIS208_CONFIG_BIAS_DACBUFBP 38
4.1.2.59 DAVIS208_CONFIG_BIAS_DIFFBN 39
4.1.2.60 DAVIS208_CONFIG_BIAS_IFREFRBN 39
4.1.2.61 DAVIS208_CONFIG_BIAS_IFTHRBN 39
4.1.2.62 DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN 40
4.1.2.63 DAVIS208_CONFIG_BIAS_LOCALBUFBN 40
4.1.2.64 DAVIS208_CONFIG_BIAS_OFFBN 40
4.1.2.65 DAVIS208_CONFIG_BIAS_ONBN 41
4.1.2.66 DAVIS208_CONFIG_BIAS_PADFOLLBN 41
4.1.2.67 DAVIS208_CONFIG_BIAS_PIXINVBN 41
4.1.2.68 DAVIS208_CONFIG_BIAS_PRBP 42
4.1.2.69 DAVIS208_CONFIG_BIAS_PRSFBP 42
4.1.2.70 DAVIS208_CONFIG_BIAS_READOUTBUFBP 42
4.1.2.71 DAVIS208_CONFIG_BIAS_REFRBP 43
4.1.2.72 DAVIS208_CONFIG_BIAS_REFSS 43
4.1.2.73 DAVIS208_CONFIG_BIAS_REFSSBN 43
4.1.2.74 DAVIS208_CONFIG_BIAS_REGBIASBP 44
4.1.2.75 DAVIS208_CONFIG_BIAS_RESETHIGHPASS 44
4.1.2.76 DAVIS208_CONFIG_BIAS_SSN 44
4.1.2.77 DAVIS208_CONFIG_BIAS_SSP 45
4.1.2.78 DAVIS208_CONFIG_CHIP_AERNAROW 45
4.1.2.79 DAVIS208_CONFIG_CHIP_ANALOGMUX0 45
4.1.2.80 DAVIS208_CONFIG_CHIP_ANALOGMUX1 45
4.1.2.81 DAVIS208_CONFIG_CHIP_ANALOGMUX2 45
4.1.2.82 DAVIS208_CONFIG_CHIP_BIASMUX0 46
4.1.2.83 DAVIS208_CONFIG_CHIP_DIGITALMUX0 46
4.1.2.84 DAVIS208_CONFIG_CHIP_DIGITALMUX1 46
4.1.2.85 DAVIS208_CONFIG_CHIP_DIGITALMUX2 46
4.1.2.86 DAVIS208_CONFIG_CHIP_DIGITALMUX3 46

4.1.2.87 DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER 46
4.1.2.88 DAVIS208_CONFIG_CHIP_RESETCALIBNEURON 47
4.1.2.89 DAVIS208_CONFIG_CHIP_RESETTESTPIXEL 47
4.1.2.90 DAVIS208_CONFIG_CHIP_SELECTBIASREFSS 47
4.1.2.91 DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER 47
4.1.2.92 DAVIS208_CONFIG_CHIP_SELECTHIGHPASS 47
4.1.2.93 DAVIS208_CONFIG_CHIP_SELECTPOSFB 47
4.1.2.94 DAVIS208_CONFIG_CHIP_SELECTPREAMPAVG 48
4.1.2.95 DAVIS208_CONFIG_CHIP_SELECTSENSE 48
4.1.2.96 DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON 48
4.1.2.97 DAVIS208_CONFIG_CHIP_USEAOUT 48
4.1.2.98 DAVIS240_CONFIG_BIAS_AEPDBN 48
4.1.2.99 DAVIS240_CONFIG_BIAS_AEPUXBP 49
4.1.2.100 DAVIS240_CONFIG_BIAS_AEPUYBP 49
4.1.2.101 DAVIS240_CONFIG_BIAS_APSCASEPC 49
4.1.2.102 DAVIS240_CONFIG_BIAS_APSOVERFLOWLEVELBN 49
4.1.2.103 DAVIS240_CONFIG_BIAS_APSROSFBN 50
4.1.2.104 DAVIS240_CONFIG_BIAS_BIASBUFFER 50
4.1.2.105 DAVIS240_CONFIG_BIAS_DIFFBN 50
4.1.2.106 DAVIS240_CONFIG_BIAS_DIFFCASBNC 50
4.1.2.107 DAVIS240_CONFIG_BIAS_IFREFRBN 51
4.1.2.108 DAVIS240_CONFIG_BIAS_IFTHRBN 51
4.1.2.109 DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN 51
4.1.2.110 DAVIS240_CONFIG_BIAS_LOCALBUFBN 51
4.1.2.111 DAVIS240_CONFIG_BIAS_OFFBN 52
4.1.2.112 DAVIS240_CONFIG_BIAS_ONBN 52
4.1.2.113 DAVIS240_CONFIG_BIAS_PADFOLLBN 52
4.1.2.114 DAVIS240_CONFIG_BIAS_PIXINVBN 52
4.1.2.115 DAVIS240_CONFIG_BIAS_PRBP 53
4.1.2.116 DAVIS240_CONFIG_BIAS_PRSFBP 53

4.1.2.117 DAVIS240_CONFIG_BIAS_REFRBP 53
4.1.2.118 DAVIS240_CONFIG_BIAS_SSN 53
4.1.2.119 DAVIS240_CONFIG_BIAS_SSP 54
4.1.2.120 DAVIS240_CONFIG_CHIP_AERNAROW 54
4.1.2.121 DAVIS240_CONFIG_CHIP_ANALOGMUX0 54
4.1.2.122 DAVIS240_CONFIG_CHIP_ANALOGMUX1 54
4.1.2.123 DAVIS240_CONFIG_CHIP_ANALOGMUX2 54
4.1.2.124 DAVIS240_CONFIG_CHIP_BIASMUX0 55
4.1.2.125 DAVIS240_CONFIG_CHIP_DIGITALMUX0 55
4.1.2.126 DAVIS240_CONFIG_CHIP_DIGITALMUX1 55
4.1.2.127 DAVIS240_CONFIG_CHIP_DIGITALMUX2 55
4.1.2.128 DAVIS240_CONFIG_CHIP_DIGITALMUX3 55
4.1.2.129 DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER 55
4.1.2.130 DAVIS240_CONFIG_CHIP_RESETCALIBNEURON 56
4.1.2.131 DAVIS240_CONFIG_CHIP_RESETTESTPIXEL 56
4.1.2.132 DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL 56
4.1.2.133 DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON 56
4.1.2.134 DAVIS240_CONFIG_CHIP_USEAOUT 56
4.1.2.135 DAVIS346_CONFIG_BIAS_ADCCOMPBP 57
4.1.2.136 DAVIS346_CONFIG_BIAS_ADCREFHIGH 57
4.1.2.137 DAVIS346_CONFIG_BIAS_ADCREFLOW 57
4.1.2.138 DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE 58
4.1.2.139 DAVIS346_CONFIG_BIAS_AEPDBN 58
4.1.2.140 DAVIS346_CONFIG_BIAS_AEPUXBP 58
4.1.2.141 DAVIS346_CONFIG_BIAS_AEPUYBP 59
4.1.2.142 DAVIS346_CONFIG_BIAS_APSCAS 59
4.1.2.143 DAVIS346_CONFIG_BIAS_APSOVERFLOWLEVEL 59
4.1.2.144 DAVIS346_CONFIG_BIAS_APSROSFBN 60
4.1.2.145 DAVIS346_CONFIG_BIAS_BIASBUFFER 60
4.1.2.146 DAVIS346_CONFIG_BIAS_COLSELLOWBN 60

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

4.1.2.177 DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON 68
4.1.2.178 DAVIS346_CONFIG_CHIP_USEAOUT 68
4.1.2.179 DAVIS640_CONFIG_BIAS_ADCCOMPBP 69
4.1.2.180 DAVIS640_CONFIG_BIAS_ADCREFHIGH 69
4.1.2.181 DAVIS640_CONFIG_BIAS_ADCREFLOW 69
4.1.2.182 DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE 70
4.1.2.183 DAVIS640_CONFIG_BIAS_AEPDBN 70
4.1.2.184 DAVIS640_CONFIG_BIAS_AEPUXBP 70
4.1.2.185 DAVIS640_CONFIG_BIAS_AEPUYBP 71
4.1.2.186 DAVIS640_CONFIG_BIAS_APSCAS 71
4.1.2.187 DAVIS640_CONFIG_BIAS_APSOVERFLOWLEVEL 71
4.1.2.188 DAVIS640_CONFIG_BIAS_APSROSFBN 72
4.1.2.189 DAVIS640_CONFIG_BIAS_BIASBUFFER 72
4.1.2.190 DAVIS640_CONFIG_BIAS_COLSELLOWBN 72
4.1.2.191 DAVIS640_CONFIG_BIAS_DACBUFBP 73
4.1.2.192 DAVIS640_CONFIG_BIAS_DIFFBN 73
4.1.2.193 DAVIS640_CONFIG_BIAS_IFREFRBN 73
4.1.2.194 DAVIS640_CONFIG_BIAS_IFTHRBN 74
4.1.2.195 DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN 74
4.1.2.196 DAVIS640_CONFIG_BIAS_LOCALBUFBN 74
4.1.2.197 DAVIS640_CONFIG_BIAS_OFFBN 75
4.1.2.198 DAVIS640_CONFIG_BIAS_ONBN 75
4.1.2.199 DAVIS640_CONFIG_BIAS_PADFOLLBN 75
4.1.2.200 DAVIS640_CONFIG_BIAS_PIXINVBN 76
4.1.2.201 DAVIS640_CONFIG_BIAS_PRBP 76
4.1.2.202 DAVIS640_CONFIG_BIAS_PRSFBP 76
4.1.2.203 DAVIS640_CONFIG_BIAS_READOUTBUFBP 77
4.1.2.204 DAVIS640_CONFIG_BIAS_REFRBP 77
4.1.2.205 DAVIS640_CONFIG_BIAS_SSN 77
4.1.2.206 DAVIS640_CONFIG_BIAS_SSP 78

4.1.2.207 DAVIS640_CONFIG_CHIP_AERNAROW 78
4.1.2.208 DAVIS640_CONFIG_CHIP_ANALOGMUX0 78
4.1.2.209 DAVIS640_CONFIG_CHIP_ANALOGMUX1 78
4.1.2.210 DAVIS640_CONFIG_CHIP_ANALOGMUX2 78
4.1.2.211 DAVIS640_CONFIG_CHIP_BIASMUX0 79
4.1.2.212 DAVIS640_CONFIG_CHIP_DIGITALMUX0 79
4.1.2.213 DAVIS640_CONFIG_CHIP_DIGITALMUX1 79
4.1.2.214 DAVIS640_CONFIG_CHIP_DIGITALMUX2 79
4.1.2.215 DAVIS640_CONFIG_CHIP_DIGITALMUX3 79
4.1.2.216 DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER 79
4.1.2.217 DAVIS640_CONFIG_CHIP_RESETCALIBNEURON 80
4.1.2.218 DAVIS640_CONFIG_CHIP_RESETTESTPIXEL 80
4.1.2.219 DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER 80
4.1.2.220 DAVIS640_CONFIG_CHIP_TESTADC 80
4.1.2.221 DAVIS640_CONFIG_CHIP_TYPENCALIBNEURON 80
4.1.2.222 DAVIS640_CONFIG_CHIP_USEAOUT 80
4.1.2.223 DAVIS640H_CONFIG_APS_GSFDRESET 81
4.1.2.224 DAVIS640H_CONFIG_APS_GSPDRESET 81
4.1.2.225 DAVIS640H_CONFIG_APS_GSRESETFALL 81
4.1.2.226 DAVIS640H_CONFIG_APS_GSTXFALL 81
4.1.2.227 DAVIS640H_CONFIG_APS_RSFDSETTLE 81
4.1.2.228 DAVIS640H_CONFIG_APS_TRANSFER 81
4.1.2.229 DAVIS640H_CONFIG_BIAS_ADCCOMPBP 82
4.1.2.230 DAVIS640H_CONFIG_BIAS_ADCREFHIGH 82
4.1.2.231 DAVIS640H_CONFIG_BIAS_ADCREFLOW 82
4.1.2.232 DAVIS640H_CONFIG_BIAS_ADCTESTVOLTAGE 83
4.1.2.233 DAVIS640H_CONFIG_BIAS_AEPDBN 83
4.1.2.234 DAVIS640H_CONFIG_BIAS_AEPUXBP 83
4.1.2.235 DAVIS640H_CONFIG_BIAS_AEPUYBP 84
4.1.2.236 DAVIS640H_CONFIG_BIAS_APSCAS 84

4.1.2.237 DAVIS640H_CONFIG_BIAS_APSROSFBN 84
4.1.2.238 DAVIS640H_CONFIG_BIAS_ARRAYBIASBUFFERBN 85
4.1.2.239 DAVIS640H_CONFIG_BIAS_ARRAYLOGICBUFFERBN 85
4.1.2.240 DAVIS640H_CONFIG_BIAS_BIASBUFFER 85
4.1.2.241 DAVIS640H_CONFIG_BIAS_DACBUFBP 86
4.1.2.242 DAVIS640H_CONFIG_BIAS_DIFFBN 86
4.1.2.243 DAVIS640H_CONFIG_BIAS_FALLTIMEBN 86
4.1.2.244 DAVIS640H_CONFIG_BIAS_GND07 87
4.1.2.245 DAVIS640H_CONFIG_BIAS_IFREFRBN 87
4.1.2.246 DAVIS640H_CONFIG_BIAS_IFTHRBN 87
4.1.2.247 DAVIS640H_CONFIG_BIAS_LCOLTIMEOUTBN 88
4.1.2.248 DAVIS640H_CONFIG_BIAS_LOCALBUFBN 88
4.1.2.249 DAVIS640H_CONFIG_BIAS_OFFBN 88
4.1.2.250 DAVIS640H_CONFIG_BIAS_ONBN 89
4.1.2.251 DAVIS640H_CONFIG_BIAS_OVG1LO 89
4.1.2.252 DAVIS640H_CONFIG_BIAS_OVG2LO 89
4.1.2.253 DAVIS640H_CONFIG_BIAS_PADFOLLBN 90
4.1.2.254 DAVIS640H_CONFIG_BIAS_PIXINVBN 90
4.1.2.255 DAVIS640H_CONFIG_BIAS_PRBP 90
4.1.2.256 DAVIS640H_CONFIG_BIAS_PRSFBP 91
4.1.2.257 DAVIS640H_CONFIG_BIAS_READOUTBUFBP 91
4.1.2.258 DAVIS640H_CONFIG_BIAS_REFRBP 91
4.1.2.259 DAVIS640H_CONFIG_BIAS_RISETIMEBP 92
4.1.2.260 DAVIS640H_CONFIG_BIAS_SSN 92
4.1.2.261 DAVIS640H_CONFIG_BIAS_SSP 92
4.1.2.262 DAVIS640H_CONFIG_BIAS_TX2OVG2HI 93
4.1.2.263 DAVIS640H_CONFIG_CHIP_ADJUSTOVG1LO 93
4.1.2.264 DAVIS640H_CONFIG_CHIP_ADJUSTOVG2LO 93
4.1.2.265 DAVIS640H_CONFIG_CHIP_ADJUSTTX2OVG2HI 93
4.1.2.266 DAVIS640H_CONFIG_CHIP_AERNAROW 93

4.1.2.267 DAVIS640H_CONFIG_CHIP_ANALOGMUX0 94
4.1.2.268 DAVIS640H_CONFIG_CHIP_ANALOGMUX1 94
4.1.2.269 DAVIS640H_CONFIG_CHIP_ANALOGMUX2 94
4.1.2.270 DAVIS640H_CONFIG_CHIP_BIASMUX0 94
4.1.2.271 DAVIS640H_CONFIG_CHIP_DIGITALMUX0 94
4.1.2.272 DAVIS640H_CONFIG_CHIP_DIGITALMUX1 94
4.1.2.273 DAVIS640H_CONFIG_CHIP_DIGITALMUX2 95
4.1.2.274 DAVIS640H_CONFIG_CHIP_DIGITALMUX3 95
4.1.2.275 DAVIS640H_CONFIG_CHIP_RESETCALIBNEURON 95
4.1.2.276 DAVIS640H_CONFIG_CHIP_RESETTESTPIXEL 95
4.1.2.277 DAVIS640H_CONFIG_CHIP_SELECTGRAYCOUNTER 95
4.1.2.278 DAVIS640H_CONFIG_CHIP_TESTADC 95
4.1.2.279 DAVIS640H_CONFIG_CHIP_TYPENCALIBNEURON 96
4.1.2.280 DAVIS640H_CONFIG_CHIP_USEAOUT 96
4.1.2.281 DAVIS_CHIP_DAVIS128 96
4.1.2.282 DAVIS_CHIP_DAVIS208 96
4.1.2.283 DAVIS_CHIP_DAVIS240A 96
4.1.2.284 DAVIS_CHIP_DAVIS240B 96
4.1.2.285 DAVIS_CHIP_DAVIS240C 96
4.1.2.286 DAVIS_CHIP_DAVIS346A 97
4.1.2.287 DAVIS_CHIP_DAVIS346B 97
4.1.2.288 DAVIS_CHIP_DAVIS346C 97
4.1.2.289 DAVIS_CHIP_DAVIS640 97
4.1.2.290 DAVIS_CHIP_DAVIS640H 97
4.1.2.291 DAVIS_CONFIG_APS 97
4.1.2.292 DAVIS_CONFIG_APS_AUTOEXPOSURE 97
4.1.2.293 DAVIS_CONFIG_APS_COLOR_FILTER 98
4.1.2.294 DAVIS_CONFIG_APS_END_COLUMN_0 98
4.1.2.295 DAVIS_CONFIG_APS_END_ROW_0 98
4.1.2.296 DAVIS_CONFIG_APS_EXPOSURE 98

4.1.2.297 DAVIS_CONFIG_APS_FRAME_INTERVAL 98
4.1.2.298 DAVIS_CONFIG_APS_FRAME_MODE 98
4.1.2.299 DAVIS_CONFIG_APS_GLOBAL_SHUTTER 99
4.1.2.300 DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER 99
4.1.2.301 DAVIS_CONFIG_APS_ORIENTATION_INFO 99
4.1.2.302 DAVIS_CONFIG_APS_RUN 99
4.1.2.303 DAVIS_CONFIG_APS_SIZE_COLUMNS 99
4.1.2.304 DAVIS_CONFIG_APS_SIZE_ROWS 99
4.1.2.305 DAVIS_CONFIG_APS_SNAPSHOT 100
4.1.2.306 DAVIS_CONFIG_APS_START_COLUMN_0 100
4.1.2.307 DAVIS_CONFIG_APS_START_ROW_0 100
4.1.2.308 DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL 100
4.1.2.309 DAVIS_CONFIG_BIAS 100
4.1.2.310 DAVIS_CONFIG_CHIP 100
4.1.2.311 DAVIS_CONFIG_DDRAER 101
4.1.2.312 DAVIS_CONFIG_DDRAER_RUN 101
4.1.2.313 DAVIS_CONFIG_DVS 101
4.1.2.314 DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL 101
4.1.2.315 DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY 101
4.1.2.316 DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_TIME 101
4.1.2.317 DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN 102
4.1.2.318 DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW 102
4.1.2.319 DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN 102
4.1.2.320 DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW 102
4.1.2.321 DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN 102
4.1.2.322 DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW 102
4.1.2.323 DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN 102
4.1.2.324 DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW 103
4.1.2.325 DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN 103
4.1.2.326 DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW 103

4.1.2.327 DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN 103
4.1.2.328 DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW 103
4.1.2.329 DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN 103
4.1.2.330 DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW 103
4.1.2.331 DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN 104
4.1.2.332 DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW 104
4.1.2.333 DAVIS_CONFIG_DVS_FILTER_PIXEL_AUTO_TRAIN 104
4.1.2.334 DAVIS_CONFIG_DVS_FILTER_POLARITY_FLATTEN 104
4.1.2.335 DAVIS_CONFIG_DVS_FILTER_POLARITY_SUPPRESS 104
4.1.2.336 DAVIS_CONFIG_DVS_FILTER_POLARITY_SUPPRESS_TYPE 104
4.1.2.337 DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD 104
4.1.2.338 DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD_TIME 105
4.1.2.339 DAVIS_CONFIG_DVS_FILTER_ROI_END_COLUMN 105
4.1.2.340 DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW 105
4.1.2.341 DAVIS_CONFIG_DVS_FILTER_ROI_START_COLUMN 105
4.1.2.342 DAVIS_CONFIG_DVS_FILTER_ROI_START_ROW 105
4.1.2.343 DAVIS_CONFIG_DVS_FILTER_SKIP_EVENTS 105
4.1.2.344 DAVIS_CONFIG_DVS_FILTER_SKIP_EVENTS_EVERY 106
4.1.2.345 DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER 106
4.1.2.346 DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER 106
4.1.2.347 DAVIS_CONFIG_DVS_HAS_POLARITY_FILTER 106
4.1.2.348 DAVIS_CONFIG_DVS_HAS_ROI_FILTER 106
4.1.2.349 DAVIS_CONFIG_DVS_HAS_SKIP_FILTER 106
4.1.2.350 DAVIS_CONFIG_DVS_HAS_STATISTICS 107
4.1.2.351 DAVIS_CONFIG_DVS_ORIENTATION_INFO 107
4.1.2.352 DAVIS_CONFIG_DVS_RUN 107
4.1.2.353 DAVIS_CONFIG_DVS_SIZE_COLUMNS 107
4.1.2.354 DAVIS_CONFIG_DVS_SIZE_ROWS 107
4.1.2.355 DAVIS_CONFIG_DVS_STATISTICS_EVENTS_COLUMN 107
4.1.2.356 DAVIS_CONFIG_DVS_STATISTICS_EVENTS_DROPPED 108

4.1.2.357 DAVIS_CONFIG_DVS_STATISTICS_EVENTS_ROW 108
4.1.2.358 DAVIS_CONFIG_DVS_STATISTICS_FILTERED_BACKGROUND_ACTIVITY . 108
4.1.2.359 DAVIS_CONFIG_DVS_STATISTICS_FILTERED_PIXELS 108
4.1.2.360 DAVIS_CONFIG_DVS_STATISTICS_FILTERED_REFRACTORY_PERIOD . . 108
4.1.2.361 DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL 108
4.1.2.362 DAVIS_CONFIG_EXTINPUT 109
4.1.2.363 DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES 109
4.1.2.364 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH 109
4.1.2.365 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY 109
4.1.2.366 DAVIS_CONFIG_EXTINPUT_DETECT_PULSES 109
4.1.2.367 DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES 109
4.1.2.368 DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_FALLING_EDGE . . . 110
4.1.2.369 DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_RISING_EDGE 110
4.1.2.370 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL 110
4.1.2.371 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH 110
4.1.2.372 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY 110
4.1.2.373 DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR 110
4.1.2.374 DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR 111
4.1.2.375 DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR 111
4.1.2.376 DAVIS_CONFIG_IMU 111
4.1.2.377 DAVIS_CONFIG_IMU_ACCEL_DLPF 111
4.1.2.378 DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE 111
4.1.2.379 DAVIS_CONFIG_IMU_GYRO_DLPF 112
4.1.2.380 DAVIS_CONFIG_IMU_GYRO_FULL_SCALE 112
4.1.2.381 DAVIS_CONFIG_IMU_ORIENTATION_INFO 112
4.1.2.382 DAVIS_CONFIG_IMU_RUN_ACCELEROMETER 112
4.1.2.383 DAVIS_CONFIG_IMU_RUN_GYROSCOPE 112
4.1.2.384 DAVIS_CONFIG_IMU_RUN_TEMPERATURE 112
4.1.2.385 DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER 113
4.1.2.386 DAVIS_CONFIG_IMU_TYPE 113

4.1.2.387 DAVIS_CONFIG_MUX 113
4.1.2.388 DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL 113
4.1.2.389 DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL 113
4.1.2.390 DAVIS_CONFIG_MUX_HAS_STATISTICS 113
4.1.2.391 DAVIS_CONFIG_MUX_RUN 114
4.1.2.392 DAVIS_CONFIG_MUX_RUN_CHIP 114
4.1.2.393 DAVIS_CONFIG_MUX_STATISTICS_DVS_DROPPED 114
4.1.2.394 DAVIS_CONFIG_MUX_STATISTICS_EXTINPUT_DROPPED 114
4.1.2.395 DAVIS_CONFIG_MUX_TIMESTAMP_RESET 114
4.1.2.396 DAVIS_CONFIG_MUX_TIMESTAMP_RUN 114
4.1.2.397 DAVIS_CONFIG_SYSINFO 115
4.1.2.398 DAVIS_CONFIG_SYSINFO_ADC_CLOCK 115
4.1.2.399 DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER 115
4.1.2.400 DAVIS_CONFIG_SYSINFO_CLOCK_DEVIATION 115
4.1.2.401 DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER 115
4.1.2.402 DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK 115
4.1.2.403 DAVIS_CONFIG_SYSINFO_LOGIC_VERSION 116
4.1.2.404 DAVIS_CONFIG_SYSINFO_USB_CLOCK 116
4.1.2.405 DAVIS_CONFIG_USB 116
4.1.2.406 DAVIS_CONFIG_USB_EARLY_PACKET_DELAY 116
4.1.2.407 DAVIS_CONFIG_USB_RUN 116
4.1.2.408 IS_DAVIS128 116
4.1.2.409 IS_DAVIS208 117
4.1.2.410 IS_DAVIS240 117
4.1.2.411 IS_DAVIS240A 117
4.1.2.412 IS_DAVIS240B 117
4.1.2.413 IS_DAVIS240C 117
4.1.2.414 IS_DAVIS346 117
4.1.2.415 IS_DAVIS346A 118
4.1.2.416 IS_DAVIS346B 118

4.1.2.417 IS_DAVIS346C 118
4.1.2.418 IS_DAVIS640 118
4.1.2.419 IS_DAVIS640H 118
4.1.3 Enumeration Type Documentation 118
4.1.3.1 caer_bias_shiftedsource_operating_mode 118
4.1.3.2 caer_bias_shiftedsource_voltage_level 119
4.1.3.3 caer_davis_aps_frame_modes 119
4.1.3.4 caer_davis_imu_invensense_accel_scale 119
4.1.3.5 caer_davis_imu_invensense_gyro_scale 119
4.1.3.6 caer_davis_imu_types 119
4.1.4 Function Documentation 120
4.1.4.1 caerBiasCoarseFineFromCurrent() 120
4.1.4.2 caerBiasCoarseFineGenerate() 120
4.1.4.3 caerBiasCoarseFineParse() 120
4.1.4.4 caerBiasCoarseFineToCurrent() 121
4.1.4.5 caerBiasShiftedSourceGenerate() 121
4.1.4.6 caerBiasShiftedSourceParse() 121
4.1.4.7 caerBiasVDACGenerate() 122
4.1.4.8 caerBiasVDACParse() 122
4.1.4.9 caerDavisInfoGet() 123
4.1.4.10 caerDavisROIConfigure() 123
4.2 devices/device.h File Reference 123
4.2.1 Detailed Description 124
4.2.2 Macro Definition Documentation 124
4.2.2.1 CAER_HOST_CONFIG_DATAEXCHANGE 124
4.2.2.2 CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING 125
4.2.2.3 CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE 125
4.2.2.4 CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS 125
4.2.2.5 CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS 125
4.2.2.6 CAER_HOST_CONFIG_LOG 125

4.2.2.7 CAER_HOST_CONFIG_LOG_LEVEL 125
4.2.2.8 CAER_HOST_CONFIG_PACKETS 126
4.2.2.9 CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL 126
4.2.2.10 CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_PACKET_SIZE 126
4.2.2.11 CAER_SUPPORTED_DEVICES_NUMBER 126
4.2.3 Typedef Documentation 126
4.2.3.1 caerDeviceHandle 126
4.2.4 Function Documentation 126
4.2.4.1 caerDeviceClose() 126
4.2.4.2 caerDeviceConfigGet() 127
4.2.4.3 caerDeviceConfigGet64() 127
4.2.4.4 caerDeviceConfigSet() 128
4.2.4.5 caerDeviceDataGet() 128
4.2.4.6 caerDeviceDataStart() 130
4.2.4.7 caerDeviceDataStop() 130
4.2.4.8 caerDeviceSendDefaultConfig() 131
4.3 devices/device_discover.h File Reference 131
4.3.1 Detailed Description 132
4.3.2 Macro Definition Documentation 132
4.3.2.1 CAER_DEVICE_DISCOVER_ALL 132
4.3.3 Typedef Documentation 132
4.3.3.1 caerDeviceDiscoveryResult 132
4.3.4 Function Documentation 132
4.3.4.1 caerDeviceDiscover() 132
4.3.4.2 caerDeviceDiscoverOpen() 133
4.4 devices/dvs128.h File Reference 133
4.4.1 Detailed Description 134
4.4.2 Macro Definition Documentation 134
4.4.2.1 CAER_DEVICE_DVS128 134
4.4.2.2 DVS128_CONFIG_BIAS 134

4.4.2.3 DVS128_CONFIG_BIAS_CAS 135
4.4.2.4 DVS128_CONFIG_BIAS_DIFF 135
4.4.2.5 DVS128_CONFIG_BIAS_DIFFOFF 135
4.4.2.6 DVS128_CONFIG_BIAS_DIFFON 135
4.4.2.7 DVS128_CONFIG_BIAS_FOLL 135
4.4.2.8 DVS128_CONFIG_BIAS_INJGND 135
4.4.2.9 DVS128_CONFIG_BIAS_PR 135
4.4.2.10 DVS128_CONFIG_BIAS_PUX 136
4.4.2.11 DVS128_CONFIG_BIAS_PUY 136
4.4.2.12 DVS128_CONFIG_BIAS_REFR 136
4.4.2.13 DVS128_CONFIG_BIAS_REQ 136
4.4.2.14 DVS128_CONFIG_BIAS_REQPD 136
4.4.2.15 DVS128_CONFIG_DVS 136
4.4.2.16 DVS128_CONFIG_DVS_ARRAY_RESET 136
4.4.2.17 DVS128_CONFIG_DVS_RUN 137
4.4.2.18 DVS128_CONFIG_DVS_TIMESTAMP_RESET 137
4.4.2.19 DVS128_CONFIG_DVS_TS_MASTER 137
4.4.3 Function Documentation 137
4.4.3.1 caerDVS128InfoGet() 137
4.5 devices/dynapse.h File Reference 137
4.5.1 Detailed Description 142
4.5.2 Macro Definition Documentation 142
4.5.2.1 CAER_DEVICE_DYNAPSE 142
4.5.2.2 DYNAPSE_CHIP_DYNAPSE 143
4.5.2.3 DYNAPSE_CONFIG_AER 143
4.5.2.4 DYNAPSE_CONFIG_AER_ACK_DELAY 143
4.5.2.5 DYNAPSE_CONFIG_AER_ACK_EXTENSION 143
4.5.2.6 DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONTROL 143
4.5.2.7 DYNAPSE_CONFIG_AER_HAS_STATISTICS 143
4.5.2.8 DYNAPSE_CONFIG_AER_RUN 143

4.5.2.9 DYNAPSE_CONFIG_AER_STATISTICS_EVENTS 144
4.5.2.10 DYNAPSE_CONFIG_AER_STATISTICS_EVENTS_DROPPED 144
4.5.2.11 DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_STALL 144
4.5.2.12 DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P 144
4.5.2.13 DYNAPSE_CONFIG_CHIP 144
4.5.2.14 DYNAPSE_CONFIG_CHIP_CONTENT 144
4.5.2.15 DYNAPSE_CONFIG_CHIP_ID 145
4.5.2.16 DYNAPSE_CONFIG_CHIP_REQ_DELAY 145
4.5.2.17 DYNAPSE_CONFIG_CHIP_REQ_EXTENSION 145
4.5.2.18 DYNAPSE_CONFIG_CHIP_RUN 145
4.5.2.19 DYNAPSE_CONFIG_CLEAR_CAM 145
4.5.2.20 DYNAPSE_CONFIG_DEFAULT_SRAM 145
4.5.2.21 DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY 145
4.5.2.22 DYNAPSE_CONFIG_MONITOR_NEU 146
4.5.2.23 DYNAPSE_CONFIG_MUX 146
4.5.2.24 DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRANSFER_STALL 146
4.5.2.25 DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE 146
4.5.2.26 DYNAPSE_CONFIG_MUX_HAS_STATISTICS 146
4.5.2.27 DYNAPSE_CONFIG_MUX_RUN 146
4.5.2.28 DYNAPSE_CONFIG_MUX_STATISTICS_AER_DROPPED 147
4.5.2.29 DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET 147
4.5.2.30 DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN 147
4.5.2.31 DYNAPSE_CONFIG_POISSONSPIKEGEN 147
4.5.2.32 DYNAPSE_CONFIG_POISSONSPIKEGEN_CHIPID 147
4.5.2.33 DYNAPSE_CONFIG_POISSONSPIKEGEN_RUN 147
4.5.2.34 DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEADDRESS 147
4.5.2.35 DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEDATA 148
4.5.2.36 DYNAPSE_CONFIG_SPIKEGEN 148
4.5.2.37 DYNAPSE_CONFIG_SPIKEGEN_BASEADDR 148
4.5.2.38 DYNAPSE_CONFIG_SPIKEGEN_ISI 148

4.5.2.39 DYNAPSE_CONFIG_SPIKEGEN_ISIBASE 148
4.5.2.40 DYNAPSE_CONFIG_SPIKEGEN_REPEAT 148
4.5.2.41 DYNAPSE_CONFIG_SPIKEGEN_RUN 148
4.5.2.42 DYNAPSE_CONFIG_SPIKEGEN_STIMCOUNT 149
4.5.2.43 DYNAPSE_CONFIG_SPIKEGEN_VARMODE 149
4.5.2.44 DYNAPSE_CONFIG_SRAM 149
4.5.2.45 DYNAPSE_CONFIG_SRAM_ADDRESS 149
4.5.2.46 DYNAPSE_CONFIG_SRAM_BURSTMODE 149
4.5.2.47 DYNAPSE_CONFIG_SRAM_DIRECTION_POS 149
4.5.2.48 DYNAPSE_CONFIG_SRAM_READ 150
4.5.2.49 DYNAPSE_CONFIG_SRAM_READDATA 150
4.5.2.50 DYNAPSE_CONFIG_SRAM_RWCOMMAND 150
4.5.2.51 DYNAPSE_CONFIG_SRAM_WRITE 150
4.5.2.52 DYNAPSE_CONFIG_SRAM_WRITEDATA 150
4.5.2.53 DYNAPSE_CONFIG_SYNAPSERECONFIG 150
4.5.2.54 DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPSELECT 151
4.5.2.55 DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOBALKERNEL 151
4.5.2.56 DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN 151
4.5.2.57 DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAMBASEADDR 151
4.5.2.58 DYNAPSE_CONFIG_SYNAPSERECONFIG_USESRAMKERNELS 151
4.5.2.59 DYNAPSE_CONFIG_SYSINFO 151
4.5.2.60 DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER 152
4.5.2.61 DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER 152
4.5.2.62 DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK 152
4.5.2.63 DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION 152
4.5.2.64 DYNAPSE_CONFIG_TAU1_RESET 152
4.5.2.65 DYNAPSE_CONFIG_TAU2_RESET 152
4.5.2.66 DYNAPSE_CONFIG_TAU2_SET 153
4.5.2.67 DYNAPSE_CONFIG_USB 153
4.5.2.68 DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY 153

4.5.2.69 DYNAPSE_CONFIG_USB_RUN 153
4.5.2.70 DYNAPSE_X4BOARD_COREX 153
4.5.2.71 DYNAPSE_X4BOARD_COREY 153
4.5.2.72 DYNAPSE_X4BOARD_NEUX 154
4.5.2.73 DYNAPSE_X4BOARD_NEUY 154
4.5.2.74 DYNAPSE_X4BOARD_NUMCHIPS 154
4.5.3 Function Documentation 154
4.5.3.1 caerBiasDynapseGenerate() 154
4.5.3.2 caerBiasDynapseParse() 154
4.5.3.3 caerDynapseCoreAddrToNeuronId() 155
4.5.3.4 caerDynapseCoreXYToNeuronId() 155
4.5.3.5 caerDynapseGenerateCamBits() 156
4.5.3.6 caerDynapseGenerateSramBits() 156
4.5.3.7 caerDynapseInfoGet() 157
4.5.3.8 caerDynapseSendDataToUSB() 157
4.5.3.9 caerDynapseSpikeEventFromXY() 158
4.5.3.10 caerDynapseSpikeEventGetX() 158
4.5.3.11 caerDynapseSpikeEventGetY() 158
4.5.3.12 caerDynapseWriteCam() 159
4.5.3.13 caerDynapseWritePoissonSpikeRate() 159
4.5.3.14 caerDynapseWriteSram() 160
4.5.3.15 caerDynapseWriteSramN() 161
4.5.3.16 caerDynapseWriteSramWords() 161
4.6 devices/edvs.h File Reference 162
4.6.1 Detailed Description 163
4.6.2 Macro Definition Documentation 163
4.6.2.1 CAER_DEVICE_EDVS 163
4.6.2.2 EDVS_CONFIG_BIAS 163
4.6.2.3 EDVS_CONFIG_BIAS_CAS 163
4.6.2.4 EDVS_CONFIG_BIAS_DIFF 163

4.6.2.5 EDVS_CONFIG_BIAS_DIFFOFF 163
4.6.2.6 EDVS_CONFIG_BIAS_DIFFON 163
4.6.2.7 EDVS_CONFIG_BIAS_FOLL 164
4.6.2.8 EDVS_CONFIG_BIAS_INJGND 164
4.6.2.9 EDVS_CONFIG_BIAS_PR 164
4.6.2.10 EDVS_CONFIG_BIAS_PUX 164
4.6.2.11 EDVS_CONFIG_BIAS_PUY 164
4.6.2.12 EDVS_CONFIG_BIAS_REFR 164
4.6.2.13 EDVS_CONFIG_BIAS_REQ 164
4.6.2.14 EDVS_CONFIG_BIAS_REQPD 165
4.6.2.15 EDVS_CONFIG_DVS 165
4.6.2.16 EDVS_CONFIG_DVS_RUN 165
4.6.2.17 EDVS_CONFIG_DVS_TIMESTAMP_RESET 165
4.6.3 Function Documentation 165
4.6.3.1 caerEDVSInfoGet() 165
4.7 devices/serial.h File Reference 166
4.7.1 Detailed Description 166
4.7.2 Macro Definition Documentation 166
4.7.2.1 CAER_HOST_CONFIG_SERIAL 166
4.7.2.2 CAER_HOST_CONFIG_SERIAL_BAUD_RATE_12M 166
4.7.2.3 CAER_HOST_CONFIG_SERIAL_BAUD_RATE_2M 167
4.7.2.4 CAER_HOST_CONFIG_SERIAL_BAUD_RATE_4M 167
4.7.2.5 CAER_HOST_CONFIG_SERIAL_BAUD_RATE_8M 167
4.7.2.6 CAER_HOST_CONFIG_SERIAL_READ_SIZE 167
4.7.3 Function Documentation 167
4.7.3.1 caerDeviceOpenSerial() 167
4.8 devices/usb.h File Reference 168
4.8.1 Detailed Description 168
4.8.2 Macro Definition Documentation 168
4.8.2.1 CAER_HOST_CONFIG_USB 168

4.8.2.2 CAER_HOST_CONFIG_USB_BUFFER_NUMBER 168
4.8.2.3 CAER_HOST_CONFIG_USB_BUFFER_SIZE 169
4.8.3 Function Documentation 169
4.8.3.1 caerDeviceOpen() 169
4.9 events/common.h File Reference 169
4.9.1 Detailed Description 171
4.9.2 Macro Definition Documentation 171
4.9.2.1 CAER_DEFAULT_EVENT_TYPES_COUNT 171
4.9.2.2 CAER_EVENT_PACKET_HEADER_SIZE 172
4.9.2.3 CAER_ITERATOR_ALL_END 172
4.9.2.4 CAER_ITERATOR_ALL_START 172
4.9.2.5 CAER_ITERATOR_VALID_END 172
4.9.2.6 CAER_ITERATOR_VALID_START 172
4.9.2.7 TS_OVERFLOW_SHIFT 173
4.9.2.8 VALID_MARK_MASK 173
4.9.2.9 VALID_MARK_SHIFT 173
4.9.3 Typedef Documentation 173
4.9.3.1 caerEventPacketHeader 173
4.9.4 Enumeration Type Documentation 173
4.9.4.1 caer_default_event_types 173
4.9.5 Function Documentation 174
4.9.5.1 caerEventPacketAllocate() 174
4.9.5.2 caerEventPacketAppend() 174
4.9.5.3 caerEventPacketClean() 175
4.9.5.4 caerEventPacketClear() 175
4.9.5.5 caerEventPacketCopy() 175
4.9.5.6 caerEventPacketCopyOnlyEvents() 176
4.9.5.7 caerEventPacketCopyOnlyValidEvents() 176
4.9.5.8 caerEventPacketEquals() 177
4.9.5.9 caerEventPacketGetDataSize() 177

4.9.5.10 caerEventPacketGetDataSizeEvents() 177
4.9.5.11 caerEventPacketGetSize() 178
4.9.5.12 caerEventPacketGetSizeEvents() 178
4.9.5.13 caerEventPacketGrow() 178
4.9.5.14 caerEventPacketHeaderGetEventCapacity() 179
4.9.5.15 caerEventPacketHeaderGetEventNumber() 179
4.9.5.16 caerEventPacketHeaderGetEventSize() 180
4.9.5.17 caerEventPacketHeaderGetEventSource() 180
4.9.5.18 caerEventPacketHeaderGetEventTSOffset() 180
4.9.5.19 caerEventPacketHeaderGetEventTSOverflow() 181
4.9.5.20 caerEventPacketHeaderGetEventType() 181
4.9.5.21 caerEventPacketHeaderGetEventValid() 181
4.9.5.22 caerEventPacketHeaderSetEventCapacity() 182
4.9.5.23 caerEventPacketHeaderSetEventNumber() 182
4.9.5.24 caerEventPacketHeaderSetEventSize() 182
4.9.5.25 caerEventPacketHeaderSetEventSource() 183
4.9.5.26 caerEventPacketHeaderSetEventTSOffset() 183
4.9.5.27 caerEventPacketHeaderSetEventTSOverflow() 183
4.9.5.28 caerEventPacketHeaderSetEventType() 184
4.9.5.29 caerEventPacketHeaderSetEventValid() 184
4.9.5.30 caerEventPacketResize() 184
4.9.5.31 caerGenericEventCopy() 185
4.9.5.32 caerGenericEventGetEvent() 185
4.9.5.33 caerGenericEventGetTimestamp() 186
4.9.5.34 caerGenericEventGetTimestamp64() 186
4.9.5.35 caerGenericEventIsValid() 187
4.9.5.36 PACKED_STRUCT() 187
4.10 events/config.h File Reference 187
4.10.1 Detailed Description 189
4.10.2 Macro Definition Documentation 189

4.10.2.1 CAER_CONFIGURATION_CONST_ITERATOR_ALL_START 189
4.10.2.2 CAER_CONFIGURATION_CONST_ITERATOR_VALID_START 189
4.10.2.3 CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_ALL_START . . . 190
4.10.2.4 CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_VALID_START . . 190
4.10.2.5 CAER_CONFIGURATION_ITERATOR_ALL_END 190
4.10.2.6 CAER_CONFIGURATION_ITERATOR_ALL_START 191
4.10.2.7 CAER_CONFIGURATION_ITERATOR_VALID_END 191
4.10.2.8 CAER_CONFIGURATION_ITERATOR_VALID_START 191
4.10.2.9 CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_END 192
4.10.2.10 CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_START 192
4.10.2.11 CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_END 192
4.10.2.12 CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_START 192
4.10.2.13 CONFIG_MODULE_ADDR_MASK 193
4.10.2.14 CONFIG_MODULE_ADDR_SHIFT 193
4.10.3 Typedef Documentation 193
4.10.3.1 caerConfigurationEvent 193
4.10.3.2 caerConfigurationEventPacket 193
4.10.4 Function Documentation 193
4.10.4.1 caerConfigurationEventGetModuleAddress() 193
4.10.4.2 caerConfigurationEventGetParameter() 194
4.10.4.3 caerConfigurationEventGetParameterAddress() 194
4.10.4.4 caerConfigurationEventGetTimestamp() 194
4.10.4.5 caerConfigurationEventGetTimestamp64() 195
4.10.4.6 caerConfigurationEventInvalidate() 195
4.10.4.7 caerConfigurationEventIsValid() 196
4.10.4.8 caerConfigurationEventPacketAllocate() 196
4.10.4.9 caerConfigurationEventPacketFromPacketHeader() 196
4.10.4.10 caerConfigurationEventPacketFromPacketHeaderConst() 197
4.10.4.11 caerConfigurationEventPacketGetEvent() 197
4.10.4.12 caerConfigurationEventPacketGetEventConst() 197

4.10.4.13 caerConfigurationEventSetModuleAddress() 198
4.10.4.14 caerConfigurationEventSetParameter() 198
4.10.4.15 caerConfigurationEventSetParameterAddress() 198
4.10.4.16 caerConfigurationEventSetTimestamp() 199
4.10.4.17 caerConfigurationEventValidate() 199
4.10.4.18 PACKED_STRUCT() [1/2] 199
4.10.4.19 PACKED_STRUCT() [2/2] 200
4.11 events/ear.h File Reference 200
4.11.1 Detailed Description 201
4.11.2 Macro Definition Documentation 201
4.11.2.1 CAER_EAR_CONST_ITERATOR_ALL_START 201
4.11.2.2 CAER_EAR_CONST_ITERATOR_VALID_START 202
4.11.2.3 CAER_EAR_CONST_REVERSE_ITERATOR_ALL_START 202
4.11.2.4 CAER_EAR_CONST_REVERSE_ITERATOR_VALID_START 202
4.11.2.5 CAER_EAR_ITERATOR_ALL_END 203
4.11.2.6 CAER_EAR_ITERATOR_ALL_START 203
4.11.2.7 CAER_EAR_ITERATOR_VALID_END 203
4.11.2.8 CAER_EAR_ITERATOR_VALID_START 203
4.11.2.9 CAER_EAR_REVERSE_ITERATOR_ALL_END 204
4.11.2.10 CAER_EAR_REVERSE_ITERATOR_ALL_START 204
4.11.2.11 CAER_EAR_REVERSE_ITERATOR_VALID_END 204
4.11.2.12 CAER_EAR_REVERSE_ITERATOR_VALID_START 204
4.11.2.13 EAR_CHANNEL_MASK 205
4.11.2.14 EAR_CHANNEL_SHIFT 205
4.11.2.15 EAR_FILTER_MASK 205
4.11.2.16 EAR_FILTER_SHIFT 205
4.11.2.17 EAR_MASK 205
4.11.2.18 EAR_NEURON_MASK 205
4.11.2.19 EAR_NEURON_SHIFT 206
4.11.2.20 EAR_SHIFT 206

4.11.3 Typedef Documentation 206
4.11.3.1 caerEarEvent 206
4.11.3.2 caerEarEventPacket 206
4.11.4 Function Documentation 206
4.11.4.1 caerEarEventGetChannel() 206
4.11.4.2 caerEarEventGetEar() 207
4.11.4.3 caerEarEventGetTimestamp() 207
4.11.4.4 caerEarEventGetTimestamp64() 208
4.11.4.5 caerEarEventInvalidate() 208
4.11.4.6 caerEarEventIsValid() 208
4.11.4.7 caerEarEventPacketAllocate() 209
4.11.4.8 caerEarEventPacketFromPacketHeader() 209
4.11.4.9 caerEarEventPacketFromPacketHeaderConst() 209
4.11.4.10 caerEarEventPacketGetEvent() 211
4.11.4.11 caerEarEventPacketGetEventConst() 211
4.11.4.12 caerEarEventSetChannel() 212
4.11.4.13 caerEarEventSetEar() 212
4.11.4.14 caerEarEventSetTimestamp() 212
4.11.4.15 caerEarEventValidate() 213
4.11.4.16 PACKED_STRUCT() [1/2] 213
4.11.4.17 PACKED_STRUCT() [2/2] 213
4.12 events/frame.h File Reference 213
4.12.1 Detailed Description 216
4.12.2 Macro Definition Documentation 216
4.12.2.1 CAER_FRAME_CONST_ITERATOR_ALL_START 216
4.12.2.2 CAER_FRAME_CONST_ITERATOR_VALID_START 217
4.12.2.3 CAER_FRAME_CONST_REVERSE_ITERATOR_ALL_START 217
4.12.2.4 CAER_FRAME_CONST_REVERSE_ITERATOR_VALID_START 217
4.12.2.5 CAER_FRAME_ITERATOR_ALL_END 218
4.12.2.6 CAER_FRAME_ITERATOR_ALL_START 218

4.12.2.7 CAER_FRAME_ITERATOR_VALID_END 218
4.12.2.8 CAER_FRAME_ITERATOR_VALID_START 218
4.12.2.9 CAER_FRAME_REVERSE_ITERATOR_ALL_END 219
4.12.2.10 CAER_FRAME_REVERSE_ITERATOR_ALL_START 219
4.12.2.11 CAER_FRAME_REVERSE_ITERATOR_VALID_END 219
4.12.2.12 CAER_FRAME_REVERSE_ITERATOR_VALID_START 219
4.12.2.13 FRAME_COLOR_CHANNELS_MASK 220
4.12.2.14 FRAME_COLOR_CHANNELS_SHIFT 220
4.12.2.15 FRAME_COLOR_FILTER_MASK 220
4.12.2.16 FRAME_COLOR_FILTER_SHIFT 220
4.12.2.17 FRAME_ROI_IDENTIFIER_MASK 220
4.12.2.18 FRAME_ROI_IDENTIFIER_SHIFT 221
4.12.3 Typedef Documentation 221
4.12.3.1 caerFrameEvent 221
4.12.3.2 caerFrameEventPacket 221
4.12.4 Enumeration Type Documentation 221
4.12.4.1 caer_frame_event_color_channels 221
4.12.4.2 caer_frame_event_color_filter 222
4.12.5 Function Documentation 222
4.12.5.1 caerFrameEventGetChannelNumber() 222
4.12.5.2 caerFrameEventGetColorFilter() 222
4.12.5.3 caerFrameEventGetExposureLength() 223
4.12.5.4 caerFrameEventGetLengthX() 223
4.12.5.5 caerFrameEventGetLengthY() 223
4.12.5.6 caerFrameEventGetPixel() 225
4.12.5.7 caerFrameEventGetPixelArrayUnsafe() 225
4.12.5.8 caerFrameEventGetPixelArrayUnsafeConst() 226
4.12.5.9 caerFrameEventGetPixelForChannel() 226
4.12.5.10 caerFrameEventGetPixelForChannelUnsafe() 227
4.12.5.11 caerFrameEventGetPixelsMaxIndex() 227

4.12.5.12 caerFrameEventGetPixelsSize() 227
4.12.5.13 caerFrameEventGetPixelUnsafe() 228
4.12.5.14 caerFrameEventGetPositionX() 228
4.12.5.15 caerFrameEventGetPositionY() 229
4.12.5.16 caerFrameEventGetROIIdentifier() 229
4.12.5.17 caerFrameEventGetTimestamp() 229
4.12.5.18 caerFrameEventGetTimestamp64() 230
4.12.5.19 caerFrameEventGetTSEndOfExposure() 230
4.12.5.20 caerFrameEventGetTSEndOfExposure64() 230
4.12.5.21 caerFrameEventGetTSEndOfFrame() 231
4.12.5.22 caerFrameEventGetTSEndOfFrame64() 231
4.12.5.23 caerFrameEventGetTSStartOfExposure() 232
4.12.5.24 caerFrameEventGetTSStartOfExposure64() 232
4.12.5.25 caerFrameEventGetTSStartOfFrame() 232
4.12.5.26 caerFrameEventGetTSStartOfFrame64() 233
4.12.5.27 caerFrameEventInvalidate() 233
4.12.5.28 caerFrameEventIsValid() 234
4.12.5.29 caerFrameEventPacketAllocate() 234
4.12.5.30 caerFrameEventPacketAllocateNumPixels() 235
4.12.5.31 caerFrameEventPacketFromPacketHeader() 235
4.12.5.32 caerFrameEventPacketFromPacketHeaderConst() 236
4.12.5.33 caerFrameEventPacketGetEvent() 236
4.12.5.34 caerFrameEventPacketGetEventConst() 236
4.12.5.35 caerFrameEventPacketGetPixelsMaxIndex() 237
4.12.5.36 caerFrameEventPacketGetPixelsSize() 237
4.12.5.37 caerFrameEventSetColorFilter() 238
4.12.5.38 caerFrameEventSetLengthXLengthYChannelNumber() 238
4.12.5.39 caerFrameEventSetPixel() 238
4.12.5.40 caerFrameEventSetPixelForChannel() 239
4.12.5.41 caerFrameEventSetPixelForChannelUnsafe() 239

4.12.5.42 caerFrameEventSetPixelUnsafe() 240
4.12.5.43 caerFrameEventSetPositionX() 240
4.12.5.44 caerFrameEventSetPositionY() 240
4.12.5.45 caerFrameEventSetROIIdentifier() 241
4.12.5.46 caerFrameEventSetTSEndOfExposure() 241
4.12.5.47 caerFrameEventSetTSEndOfFrame() 241
4.12.5.48 caerFrameEventSetTSStartOfExposure() 242
4.12.5.49 caerFrameEventSetTSStartOfFrame() 242
4.12.5.50 caerFrameEventValidate() 242
4.12.5.51 PACKED_STRUCT() [1/2] 243
4.12.5.52 PACKED_STRUCT() [2/2] 243
4.13 events/imu6.h File Reference 243
4.13.1 Detailed Description 244
4.13.2 Macro Definition Documentation 244
4.13.2.1 CAER_IMU6_CONST_ITERATOR_ALL_START 245
4.13.2.2 CAER_IMU6_CONST_ITERATOR_VALID_START 245
4.13.2.3 CAER_IMU6_CONST_REVERSE_ITERATOR_ALL_START 245
4.13.2.4 CAER_IMU6_CONST_REVERSE_ITERATOR_VALID_START 246
4.13.2.5 CAER_IMU6_ITERATOR_ALL_END 246
4.13.2.6 CAER_IMU6_ITERATOR_ALL_START 246
4.13.2.7 CAER_IMU6_ITERATOR_VALID_END 246
4.13.2.8 CAER_IMU6_ITERATOR_VALID_START 247
4.13.2.9 CAER_IMU6_REVERSE_ITERATOR_ALL_END 247
4.13.2.10 CAER_IMU6_REVERSE_ITERATOR_ALL_START 247
4.13.2.11 CAER_IMU6_REVERSE_ITERATOR_VALID_END 247
4.13.2.12 CAER_IMU6_REVERSE_ITERATOR_VALID_START 248
4.13.3 Typedef Documentation 248
4.13.3.1 caerIMU6Event 248
4.13.3.2 caerIMU6EventPacket 248
4.13.4 Function Documentation 248

4.13.4.1 caerIMU6EventGetAccelX() 248
4.13.4.2 caerIMU6EventGetAccelY() 249
4.13.4.3 caerIMU6EventGetAccelZ() 249
4.13.4.4 caerIMU6EventGetGyroX() 249
4.13.4.5 caerIMU6EventGetGyroY() 251
4.13.4.6 caerIMU6EventGetGyroZ() 251
4.13.4.7 caerIMU6EventGetTemp() 251
4.13.4.8 caerIMU6EventGetTimestamp() 253
4.13.4.9 caerIMU6EventGetTimestamp64() 253
4.13.4.10 caerIMU6EventInvalidate() 254
4.13.4.11 caerIMU6EventIsValid() 254
4.13.4.12 caerIMU6EventPacketAllocate() 254
4.13.4.13 caerIMU6EventPacketFromPacketHeader() 255
4.13.4.14 caerIMU6EventPacketFromPacketHeaderConst() 255
4.13.4.15 caerIMU6EventPacketGetEvent() 255
4.13.4.16 caerIMU6EventPacketGetEventConst() 256
4.13.4.17 caerIMU6EventSetAccelX() 256
4.13.4.18 caerIMU6EventSetAccelY() 257
4.13.4.19 caerIMU6EventSetAccelZ() 257
4.13.4.20 caerIMU6EventSetGyroX() 257
4.13.4.21 caerIMU6EventSetGyroY() 257
4.13.4.22 caerIMU6EventSetGyroZ() 258
4.13.4.23 caerIMU6EventSetTemp() 258
4.13.4.24 caerIMU6EventSetTimestamp() 258
4.13.4.25 caerIMU6EventValidate() 259
4.13.4.26 PACKED_STRUCT() [1/2] 259
4.13.4.27 PACKED_STRUCT() [2/2] 259
4.14 events/imu9.h File Reference 259
4.14.1 Detailed Description 261
4.14.2 Macro Definition Documentation 261

4.14.2.1 CAER_IMU9_CONST_ITERATOR_ALL_START 261
4.14.2.2 CAER_IMU9_CONST_ITERATOR_VALID_START 262
4.14.2.3 CAER_IMU9_CONST_REVERSE_ITERATOR_ALL_START 262
4.14.2.4 CAER_IMU9_CONST_REVERSE_ITERATOR_VALID_START 262
4.14.2.5 CAER_IMU9_ITERATOR_ALL_END 263
4.14.2.6 CAER_IMU9_ITERATOR_ALL_START 263
4.14.2.7 CAER_IMU9_ITERATOR_VALID_END 263
4.14.2.8 CAER_IMU9_ITERATOR_VALID_START 263
4.14.2.9 CAER_IMU9_REVERSE_ITERATOR_ALL_END 264
4.14.2.10 CAER_IMU9_REVERSE_ITERATOR_ALL_START 264
4.14.2.11 CAER_IMU9_REVERSE_ITERATOR_VALID_END 264
4.14.2.12 CAER_IMU9_REVERSE_ITERATOR_VALID_START 264
4.14.3 Typedef Documentation 265
4.14.3.1 caerIMU9Event 265
4.14.3.2 caerIMU9EventPacket 265
4.14.4 Function Documentation 265
4.14.4.1 caerIMU9EventGetAccelX() 265
4.14.4.2 caerIMU9EventGetAccelY() 265
4.14.4.3 caerIMU9EventGetAccelZ() 266
4.14.4.4 caerIMU9EventGetCompX() 266
4.14.4.5 caerIMU9EventGetCompY() 266
4.14.4.6 caerIMU9EventGetCompZ() 268
4.14.4.7 caerIMU9EventGetGyroX() 268
4.14.4.8 caerIMU9EventGetGyroY() 268
4.14.4.9 caerIMU9EventGetGyroZ() 270
4.14.4.10 caerIMU9EventGetTemp() 270
4.14.4.11 caerIMU9EventGetTimestamp() 270
4.14.4.12 caerIMU9EventGetTimestamp64() 271
4.14.4.13 caerIMU9EventInvalidate() 271
4.14.4.14 caerIMU9EventIsValid() 272

4.14.4.15 caerIMU9EventPacketAllocate() 272
4.14.4.16 caerIMU9EventPacketFromPacketHeader() 272
4.14.4.17 caerIMU9EventPacketFromPacketHeaderConst() 273
4.14.4.18 caerIMU9EventPacketGetEvent() 273
4.14.4.19 caerIMU9EventPacketGetEventConst() 274
4.14.4.20 caerIMU9EventSetAccelX() 274
4.14.4.21 caerIMU9EventSetAccelY() 274
4.14.4.22 caerIMU9EventSetAccelZ() 275
4.14.4.23 caerIMU9EventSetCompX() 275
4.14.4.24 caerIMU9EventSetCompY() 275
4.14.4.25 caerIMU9EventSetCompZ() 275
4.14.4.26 caerIMU9EventSetGyroX() 276
4.14.4.27 caerIMU9EventSetGyroY() 276
4.14.4.28 caerIMU9EventSetGyroZ() 276
4.14.4.29 caerIMU9EventSetTemp() 277
4.14.4.30 caerIMU9EventSetTimestamp() 277
4.14.4.31 caerIMU9EventValidate() 277
4.14.4.32 PACKED_STRUCT() [1/2] 278
4.14.4.33 PACKED_STRUCT() [2/2] 278
4.15 events/matrix4x4.h File Reference 278
4.15.1 Detailed Description 280
4.15.2 Macro Definition Documentation 280
4.15.2.1 CAER_MATRIX4x4_CONST_ITERATOR_ALL_START 280
4.15.2.2 CAER_MATRIX4x4_CONST_ITERATOR_VALID_START 281
4.15.2.3 CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_ALL_START 281
4.15.2.4 CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_VALID_START 282
4.15.2.5 CAER_MATRIX4x4_ITERATOR_ALL_END 282
4.15.2.6 CAER_MATRIX4x4_ITERATOR_ALL_START 282
4.15.2.7 CAER_MATRIX4x4_ITERATOR_VALID_END 282
4.15.2.8 CAER_MATRIX4x4_ITERATOR_VALID_START 283

4.15.2.9 CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_END 283
4.15.2.10 CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_START 283
4.15.2.11 CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_END 284
4.15.2.12 CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_START 284
4.15.2.13 MATRIX4x4_SCALE_MASK 284
4.15.2.14 MATRIX4x4_SCALE_SHIFT 284
4.15.2.15 MATRIX4x4_TYPE_MASK 284
4.15.2.16 MATRIX4x4_TYPE_SHIFT 285
4.15.3 Typedef Documentation 285
4.15.3.1 caerMatrix4x4Event 285
4.15.3.2 caerMatrix4x4EventPacket 285
4.15.4 Function Documentation 285
4.15.4.1 caerMatrix4x4EventGetM00() 285
4.15.4.2 caerMatrix4x4EventGetM01() 286
4.15.4.3 caerMatrix4x4EventGetM02() 286
4.15.4.4 caerMatrix4x4EventGetM03() 286
4.15.4.5 caerMatrix4x4EventGetM10() 287
4.15.4.6 caerMatrix4x4EventGetM11() 287
4.15.4.7 caerMatrix4x4EventGetM12() 287
4.15.4.8 caerMatrix4x4EventGetM13() 288
4.15.4.9 caerMatrix4x4EventGetM20() 288
4.15.4.10 caerMatrix4x4EventGetM21() 288
4.15.4.11 caerMatrix4x4EventGetM22() 289
4.15.4.12 caerMatrix4x4EventGetM23() 289
4.15.4.13 caerMatrix4x4EventGetM30() 289
4.15.4.14 caerMatrix4x4EventGetM31() 291
4.15.4.15 caerMatrix4x4EventGetM32() 291
4.15.4.16 caerMatrix4x4EventGetM33() 291
4.15.4.17 caerMatrix4x4EventGetScale() 293
4.15.4.18 caerMatrix4x4EventGetTimestamp() 293

4.15.4.19 caerMatrix4x4EventGetTimestamp64() 294
4.15.4.20 caerMatrix4x4EventGetType() 294
4.15.4.21 caerMatrix4x4EventInvalidate() 294
4.15.4.22 caerMatrix4x4EventIsValid() 295
4.15.4.23 caerMatrix4x4EventPacketAllocate() 295
4.15.4.24 caerMatrix4x4EventPacketFromPacketHeader() 295
4.15.4.25 caerMatrix4x4EventPacketFromPacketHeaderConst() 296
4.15.4.26 caerMatrix4x4EventPacketGetEvent() 296
4.15.4.27 caerMatrix4x4EventPacketGetEventConst() 297
4.15.4.28 caerMatrix4x4EventSetM00() 297
4.15.4.29 caerMatrix4x4EventSetM01() 297
4.15.4.30 caerMatrix4x4EventSetM02() 298
4.15.4.31 caerMatrix4x4EventSetM03() 298
4.15.4.32 caerMatrix4x4EventSetM10() 298
4.15.4.33 caerMatrix4x4EventSetM11() 298
4.15.4.34 caerMatrix4x4EventSetM12() 299
4.15.4.35 caerMatrix4x4EventSetM13() 299
4.15.4.36 caerMatrix4x4EventSetM20() 299
4.15.4.37 caerMatrix4x4EventSetM21() 300
4.15.4.38 caerMatrix4x4EventSetM22() 300
4.15.4.39 caerMatrix4x4EventSetM23() 300
4.15.4.40 caerMatrix4x4EventSetM30() 301
4.15.4.41 caerMatrix4x4EventSetM31() 301
4.15.4.42 caerMatrix4x4EventSetM32() 301
4.15.4.43 caerMatrix4x4EventSetM33() 302
4.15.4.44 caerMatrix4x4EventSetScale() 302
4.15.4.45 caerMatrix4x4EventSetTimestamp() 302
4.15.4.46 caerMatrix4x4EventSetType() 303
4.15.4.47 caerMatrix4x4EventValidate() 303
4.15.4.48 PACKED_STRUCT() [1/2] 303

4.15.4.49 PACKED_STRUCT() [2/2] 303
4.16 events/packetContainer.h File Reference 304
4.16.1 Detailed Description 305
4.16.2 Macro Definition Documentation 305
4.16.2.1 CAER_EVENT_PACKET_CONTAINER_CONST_ITERATOR_START 305
4.16.2.2 CAER_EVENT_PACKET_CONTAINER_ITERATOR_END 306
4.16.2.3 CAER_EVENT_PACKET_CONTAINER_ITERATOR_START 306
4.16.3 Typedef Documentation 306
4.16.3.1 caerEventPacketContainer 306
4.16.4 Function Documentation 306
4.16.4.1 caerEventPacketContainerAllocate() 306
4.16.4.2 caerEventPacketContainerCopyAllEvents() 307
4.16.4.3 caerEventPacketContainerCopyValidEvents() 307
4.16.4.4 caerEventPacketContainerFindEventPacketByType() 308
4.16.4.5 caerEventPacketContainerFindEventPacketByTypeConst() 308
4.16.4.6 caerEventPacketContainerFree() 308
4.16.4.7 caerEventPacketContainerGetEventPacket() 309
4.16.4.8 caerEventPacketContainerGetEventPacketConst() 309
4.16.4.9 caerEventPacketContainerGetEventPacketsNumber() 309
4.16.4.10 caerEventPacketContainerGetEventsNumber() 310
4.16.4.11 caerEventPacketContainerGetEventsValidNumber() 310
4.16.4.12 caerEventPacketContainerGetHighestEventTimestamp() 311
4.16.4.13 caerEventPacketContainerGetLowestEventTimestamp() 311
4.16.4.14 caerEventPacketContainerSetEventPacket() 311
4.16.4.15 caerEventPacketContainerSetEventPacketsNumber() 312
4.16.4.16 caerEventPacketContainerUpdateStatistics() 312
4.16.4.17 PACKED_STRUCT() 312
4.17 events/point1d.h File Reference 312
4.17.1 Detailed Description 314
4.17.2 Macro Definition Documentation 314

4.17.2.1 CAER_POINT1D_CONST_ITERATOR_ALL_START 314
4.17.2.2 CAER_POINT1D_CONST_ITERATOR_VALID_START 314
4.17.2.3 CAER_POINT1D_CONST_REVERSE_ITERATOR_ALL_START 315
4.17.2.4 CAER_POINT1D_CONST_REVERSE_ITERATOR_VALID_START 315
4.17.2.5 CAER_POINT1D_ITERATOR_ALL_END 315
4.17.2.6 CAER_POINT1D_ITERATOR_ALL_START 316
4.17.2.7 CAER_POINT1D_ITERATOR_VALID_END 316
4.17.2.8 CAER_POINT1D_ITERATOR_VALID_START 316
4.17.2.9 CAER_POINT1D_REVERSE_ITERATOR_ALL_END 316
4.17.2.10 CAER_POINT1D_REVERSE_ITERATOR_ALL_START 317
4.17.2.11 CAER_POINT1D_REVERSE_ITERATOR_VALID_END 317
4.17.2.12 CAER_POINT1D_REVERSE_ITERATOR_VALID_START 317
4.17.2.13 POINT1D_SCALE_MASK 317
4.17.2.14 POINT1D_SCALE_SHIFT 318
4.17.2.15 POINT1D_TYPE_MASK 318
4.17.2.16 POINT1D_TYPE_SHIFT 318
4.17.3 Typedef Documentation 318
4.17.3.1 caerPoint1DEvent 318
4.17.3.2 caerPoint1DEventPacket 318
4.17.4 Function Documentation 318
4.17.4.1 caerPoint1DEventGetScale() 318
4.17.4.2 caerPoint1DEventGetTimestamp() 319
4.17.4.3 caerPoint1DEventGetTimestamp64() 319
4.17.4.4 caerPoint1DEventGetType() 320
4.17.4.5 caerPoint1DEventGetX() 320
4.17.4.6 caerPoint1DEventInvalidate() 320
4.17.4.7 caerPoint1DEventIsValid() 321
4.17.4.8 caerPoint1DEventPacketAllocate() 321
4.17.4.9 caerPoint1DEventPacketFromPacketHeader() 321
4.17.4.10 caerPoint1DEventPacketFromPacketHeaderConst() 322

4.17.4.11 caerPoint1DEventPacketGetEvent() 322
4.17.4.12 caerPoint1DEventPacketGetEventConst() 323
4.17.4.13 caerPoint1DEventSetScale() 323
4.17.4.14 caerPoint1DEventSetTimestamp() 323
4.17.4.15 caerPoint1DEventSetType() 324
4.17.4.16 caerPoint1DEventSetX() 324
4.17.4.17 caerPoint1DEventValidate() 324
4.17.4.18 PACKED_STRUCT() [1/2] 325
4.17.4.19 PACKED_STRUCT() [2/2] 325
4.18 events/point2d.h File Reference 325
4.18.1 Detailed Description 326
4.18.2 Macro Definition Documentation 326
4.18.2.1 CAER_POINT2D_CONST_ITERATOR_ALL_START 327
4.18.2.2 CAER_POINT2D_CONST_ITERATOR_VALID_START 327
4.18.2.3 CAER_POINT2D_CONST_REVERSE_ITERATOR_ALL_START 327
4.18.2.4 CAER_POINT2D_CONST_REVERSE_ITERATOR_VALID_START 328
4.18.2.5 CAER_POINT2D_ITERATOR_ALL_END 328
4.18.2.6 CAER_POINT2D_ITERATOR_ALL_START 328
4.18.2.7 CAER_POINT2D_ITERATOR_VALID_END 328
4.18.2.8 CAER_POINT2D_ITERATOR_VALID_START 329
4.18.2.9 CAER_POINT2D_REVERSE_ITERATOR_ALL_END 329
4.18.2.10 CAER_POINT2D_REVERSE_ITERATOR_ALL_START 329
4.18.2.11 CAER_POINT2D_REVERSE_ITERATOR_VALID_END 329
4.18.2.12 CAER_POINT2D_REVERSE_ITERATOR_VALID_START 330
4.18.2.13 POINT2D_SCALE_MASK 330
4.18.2.14 POINT2D_SCALE_SHIFT 330
4.18.2.15 POINT2D_TYPE_MASK 330
4.18.2.16 POINT2D_TYPE_SHIFT 330
4.18.3 Typedef Documentation 331
4.18.3.1 caerPoint2DEvent 331

4.18.3.2 caerPoint2DEventPacket 331
4.18.4 Function Documentation 331
4.18.4.1 caerPoint2DEventGetScale() 331
4.18.4.2 caerPoint2DEventGetTimestamp() 331
4.18.4.3 caerPoint2DEventGetTimestamp64() 332
4.18.4.4 caerPoint2DEventGetType() 332
4.18.4.5 caerPoint2DEventGetX() 333
4.18.4.6 caerPoint2DEventGetY() 333
4.18.4.7 caerPoint2DEventInvalidate() 333
4.18.4.8 caerPoint2DEventIsValid() 334
4.18.4.9 caerPoint2DEventPacketAllocate() 334
4.18.4.10 caerPoint2DEventPacketFromPacketHeader() 334
4.18.4.11 caerPoint2DEventPacketFromPacketHeaderConst() 335
4.18.4.12 caerPoint2DEventPacketGetEvent() 335
4.18.4.13 caerPoint2DEventPacketGetEventConst() 336
4.18.4.14 caerPoint2DEventSetScale() 336
4.18.4.15 caerPoint2DEventSetTimestamp() 336
4.18.4.16 caerPoint2DEventSetType() 337
4.18.4.17 caerPoint2DEventSetX() 337
4.18.4.18 caerPoint2DEventSetY() 337
4.18.4.19 caerPoint2DEventValidate() 338
4.18.4.20 PACKED_STRUCT() [1/2] 338
4.18.4.21 PACKED_STRUCT() [2/2] 338
4.19 events/point3d.h File Reference 338
4.19.1 Detailed Description 340
4.19.2 Macro Definition Documentation 340
4.19.2.1 CAER_POINT3D_CONST_ITERATOR_ALL_START 340
4.19.2.2 CAER_POINT3D_CONST_ITERATOR_VALID_START 341
4.19.2.3 CAER_POINT3D_CONST_REVERSE_ITERATOR_ALL_START 341
4.19.2.4 CAER_POINT3D_CONST_REVERSE_ITERATOR_VALID_START 341

4.19.2.5 CAER_POINT3D_ITERATOR_ALL_END 342
4.19.2.6 CAER_POINT3D_ITERATOR_ALL_START 342
4.19.2.7 CAER_POINT3D_ITERATOR_VALID_END 342
4.19.2.8 CAER_POINT3D_ITERATOR_VALID_START 342
4.19.2.9 CAER_POINT3D_REVERSE_ITERATOR_ALL_END 343
4.19.2.10 CAER_POINT3D_REVERSE_ITERATOR_ALL_START 343
4.19.2.11 CAER_POINT3D_REVERSE_ITERATOR_VALID_END 343
4.19.2.12 CAER_POINT3D_REVERSE_ITERATOR_VALID_START 343
4.19.2.13 POINT3D_SCALE_MASK 344
4.19.2.14 POINT3D_SCALE_SHIFT 344
4.19.2.15 POINT3D_TYPE_MASK 344
4.19.2.16 POINT3D_TYPE_SHIFT 344
4.19.3 Typedef Documentation 344
4.19.3.1 caerPoint3DEvent 344
4.19.3.2 caerPoint3DEventPacket 344
4.19.4 Function Documentation 345
4.19.4.1 caerPoint3DEventGetScale() 345
4.19.4.2 caerPoint3DEventGetTimestamp() 345
4.19.4.3 caerPoint3DEventGetTimestamp64() 345
4.19.4.4 caerPoint3DEventGetType() 346
4.19.4.5 caerPoint3DEventGetX() 346
4.19.4.6 caerPoint3DEventGetY() 347
4.19.4.7 caerPoint3DEventGetZ() 347
4.19.4.8 caerPoint3DEventInvalidate() 347
4.19.4.9 caerPoint3DEventIsValid() 348
4.19.4.10 caerPoint3DEventPacketAllocate() 348
4.19.4.11 caerPoint3DEventPacketFromPacketHeader() 348
4.19.4.12 caerPoint3DEventPacketFromPacketHeaderConst() 349
4.19.4.13 caerPoint3DEventPacketGetEvent() 349
4.19.4.14 caerPoint3DEventPacketGetEventConst() 350

4.19.4.15 caerPoint3DEventSetScale() 350

4.19.4.16 caerPoint3DEventSetTimestamp() 350

4.19.4.17 caerPoint3DEventSetType() 351

4.19.4.18 caerPoint3DEventSetX() 351

4.19.4.19 caerPoint3DEventSetY() 351

4.19.4.20 caerPoint3DEventSetZ() 352

4.19.4.21 caerPoint3DEventValidate() 352

4.19.4.22 PACKED_STRUCT() [1/2] 352

4.19.4.23 PACKED_STRUCT() [2/2] 352

4.20 events/point4d.h File Reference 353

4.20.1 Detailed Description 354

4.20.2 Macro Definition Documentation 354

4.20.2.1 CAER_POINT4D_CONST_ITERATOR_ALL_START 354

4.20.2.2 CAER_POINT4D_CONST_ITERATOR_VALID_START 355

4.20.2.3 CAER_POINT4D_CONST_REVERSE_ITERATOR_ALL_START 355

4.20.2.4 CAER_POINT4D_CONST_REVERSE_ITERATOR_VALID_START 355

4.20.2.5 CAER_POINT4D_ITERATOR_ALL_END 356

4.20.2.6 CAER_POINT4D_ITERATOR_ALL_START 356

4.20.2.7 CAER_POINT4D_ITERATOR_VALID_END 356

4.20.2.8 CAER_POINT4D_ITERATOR_VALID_START 356

4.20.2.9 CAER_POINT4D_REVERSE_ITERATOR_ALL_END 357

4.20.2.10 CAER_POINT4D_REVERSE_ITERATOR_ALL_START 357

4.20.2.11 CAER_POINT4D_REVERSE_ITERATOR_VALID_END 357

4.20.2.12 CAER_POINT4D_REVERSE_ITERATOR_VALID_START 357

4.20.2.13 POINT4D_SCALE_MASK 358

4.20.2.14 POINT4D_SCALE_SHIFT 358

4.20.2.15 POINT4D_TYPE_MASK 358

4.20.2.16 POINT4D_TYPE_SHIFT 358

4.20.3 Typedef Documentation 358

4.20.3.1 caerPoint4DEvent 358

4.20.3.2 caerPoint4DEventPacket 358
4.20.4 Function Documentation 359
4.20.4.1 caerPoint4DEventGetScale() 359
4.20.4.2 caerPoint4DEventGetTimestamp() 359
4.20.4.3 caerPoint4DEventGetTimestamp64() 359
4.20.4.4 caerPoint4DEventGetType() 360
4.20.4.5 caerPoint4DEventGetW() 360
4.20.4.6 caerPoint4DEventGetX() 361
4.20.4.7 caerPoint4DEventGetY() 361
4.20.4.8 caerPoint4DEventGetZ() 361
4.20.4.9 caerPoint4DEventInvalidate() 362
4.20.4.10 caerPoint4DEventIsValid() 362
4.20.4.11 caerPoint4DEventPacketAllocate() 362
4.20.4.12 caerPoint4DEventPacketFromPacketHeader() 363
4.20.4.13 caerPoint4DEventPacketFromPacketHeaderConst() 363
4.20.4.14 caerPoint4DEventPacketGetEvent() 363
4.20.4.15 caerPoint4DEventPacketGetEventConst() 364
4.20.4.16 caerPoint4DEventSetScale() 364
4.20.4.17 caerPoint4DEventSetTimestamp() 365
4.20.4.18 caerPoint4DEventSetType() 365
4.20.4.19 caerPoint4DEventSetW() 365
4.20.4.20 caerPoint4DEventSetX() 365
4.20.4.21 caerPoint4DEventSetY() 366
4.20.4.22 caerPoint4DEventSetZ() 366
4.20.4.23 caerPoint4DEventValidate() 366
4.20.4.24 PACKED_STRUCT() [1/2] 368
4.20.4.25 PACKED_STRUCT() [2/2] 368
4.21 events/polarity.h File Reference 368
4.21.1 Detailed Description 369
4.21.2 Macro Definition Documentation 369

4.21.2.1 CAER_POLARITY_CONST_ITERATOR_ALL_START 370
4.21.2.2 CAER_POLARITY_CONST_ITERATOR_VALID_START 370
4.21.2.3 CAER_POLARITY_CONST_REVERSE_ITERATOR_ALL_START 370
4.21.2.4 CAER_POLARITY_CONST_REVERSE_ITERATOR_VALID_START 371
4.21.2.5 CAER_POLARITY_ITERATOR_ALL_END 371
4.21.2.6 CAER_POLARITY_ITERATOR_ALL_START 371
4.21.2.7 CAER_POLARITY_ITERATOR_VALID_END 371
4.21.2.8 CAER_POLARITY_ITERATOR_VALID_START 372
4.21.2.9 CAER_POLARITY_REVERSE_ITERATOR_ALL_END 372
4.21.2.10 CAER_POLARITY_REVERSE_ITERATOR_ALL_START 372
4.21.2.11 CAER_POLARITY_REVERSE_ITERATOR_VALID_END 372
4.21.2.12 CAER_POLARITY_REVERSE_ITERATOR_VALID_START 373
4.21.2.13 POLARITY_MASK 373
4.21.2.14 POLARITY_SHIFT 373
4.21.2.15 POLARITY_X_ADDR_MASK 373
4.21.2.16 POLARITY_X_ADDR_SHIFT 373
4.21.2.17 POLARITY_Y_ADDR_MASK 374
4.21.2.18 POLARITY_Y_ADDR_SHIFT 374
4.21.3 Typedef Documentation 374
4.21.3.1 caerPolarityEvent 374
4.21.3.2 caerPolarityEventPacket 374
4.21.4 Function Documentation 374
4.21.4.1 caerPolarityEventGetPolarity() 374
4.21.4.2 caerPolarityEventGetTimestamp() 375
4.21.4.3 caerPolarityEventGetTimestamp64() 375
4.21.4.4 caerPolarityEventGetX() 375
4.21.4.5 caerPolarityEventGetY() 376
4.21.4.6 caerPolarityEventInvalidate() 376
4.21.4.7 caerPolarityEventIsValid() 377
4.21.4.8 caerPolarityEventPacketAllocate() 377

4.21.4.9 caerPolarityEventPacketFromPacketHeader() 377
4.21.4.10 caerPolarityEventPacketFromPacketHeaderConst() 378
4.21.4.11 caerPolarityEventPacketGetEvent() 378
4.21.4.12 caerPolarityEventPacketGetEventConst() 378
4.21.4.13 caerPolarityEventSetPolarity() 379
4.21.4.14 caerPolarityEventSetTimestamp() 379
4.21.4.15 caerPolarityEventSetX() 379
4.21.4.16 caerPolarityEventSetY() 380
4.21.4.17 caerPolarityEventValidate() 380
4.21.4.18 PACKED_STRUCT() [1/2] 380
4.21.4.19 PACKED_STRUCT() [2/2] 381
4.22 events/sample.h File Reference 381
4.22.1 Detailed Description 382
4.22.2 Macro Definition Documentation 382
4.22.2.1 CAER_SAMPLE_CONST_ITERATOR_ALL_START 382
4.22.2.2 CAER_SAMPLE_CONST_ITERATOR_VALID_START 383
4.22.2.3 CAER_SAMPLE_CONST_REVERSE_ITERATOR_ALL_START 383
4.22.2.4 CAER_SAMPLE_CONST_REVERSE_ITERATOR_VALID_START 383
4.22.2.5 CAER_SAMPLE_ITERATOR_ALL_END 384
4.22.2.6 CAER_SAMPLE_ITERATOR_ALL_START 384
4.22.2.7 CAER_SAMPLE_ITERATOR_VALID_END 384
4.22.2.8 CAER_SAMPLE_ITERATOR_VALID_START 384
4.22.2.9 CAER_SAMPLE_REVERSE_ITERATOR_ALL_END 385
4.22.2.10 CAER_SAMPLE_REVERSE_ITERATOR_ALL_START 385
4.22.2.11 CAER_SAMPLE_REVERSE_ITERATOR_VALID_END 385
4.22.2.12 CAER_SAMPLE_REVERSE_ITERATOR_VALID_START 385
4.22.2.13 SAMPLE_MASK 386
4.22.2.14 SAMPLE_SHIFT 386
4.22.2.15 SAMPLE_TYPE_MASK 386
4.22.2.16 SAMPLE_TYPE_SHIFT 386

4.22.3 Typedef Documentation 386
4.22.3.1 caerSampleEvent 386
4.22.3.2 caerSampleEventPacket 386
4.22.4 Function Documentation 387
4.22.4.1 caerSampleEventGetSample() 387
4.22.4.2 caerSampleEventGetTimestamp() 387
4.22.4.3 caerSampleEventGetTimestamp64() 387
4.22.4.4 caerSampleEventGetType() 388
4.22.4.5 caerSampleEventInvalidate() 388
4.22.4.6 caerSampleEventIsValid() 388
4.22.4.7 caerSampleEventPacketAllocate() 389
4.22.4.8 caerSampleEventPacketFromPacketHeader() 389
4.22.4.9 caerSampleEventPacketFromPacketHeaderConst() 390
4.22.4.10 caerSampleEventPacketGetEvent() 390
4.22.4.11 caerSampleEventPacketGetEventConst() 390
4.22.4.12 caerSampleEventSetSample() 391
4.22.4.13 caerSampleEventSetTimestamp() 391
4.22.4.14 caerSampleEventSetType() 391
4.22.4.15 caerSampleEventValidate() 392
4.22.4.16 PACKED_STRUCT() [1/2] 392
4.22.4.17 PACKED_STRUCT() [2/2] 392
4.23 events/special.h File Reference 392
4.23.1 Detailed Description 394
4.23.2 Macro Definition Documentation 394
4.23.2.1 CAER_SPECIAL_CONST_ITERATOR_ALL_START 394
4.23.2.2 CAER_SPECIAL_CONST_ITERATOR_VALID_START 395
4.23.2.3 CAER_SPECIAL_CONST_REVERSE_ITERATOR_ALL_START 395
4.23.2.4 CAER_SPECIAL_CONST_REVERSE_ITERATOR_VALID_START 396
4.23.2.5 CAER_SPECIAL_ITERATOR_ALL_END 396
4.23.2.6 CAER_SPECIAL_ITERATOR_ALL_START 396

4.23.2.7 CAER_SPECIAL_ITERATOR_VALID_END 396
4.23.2.8 CAER_SPECIAL_ITERATOR_VALID_START 397
4.23.2.9 CAER_SPECIAL_REVERSE_ITERATOR_ALL_END 397
4.23.2.10 CAER_SPECIAL_REVERSE_ITERATOR_ALL_START 397
4.23.2.11 CAER_SPECIAL_REVERSE_ITERATOR_VALID_END 397
4.23.2.12 CAER_SPECIAL_REVERSE_ITERATOR_VALID_START 398
4.23.2.13 SPECIAL_DATA_MASK 398
4.23.2.14 SPECIAL_DATA_SHIFT 398
4.23.2.15 SPECIAL_TYPE_MASK 398
4.23.2.16 SPECIAL_TYPE_SHIFT 398
4.23.3 Typedef Documentation 399
4.23.3.1 caerSpecialEvent 399
4.23.3.2 caerSpecialEventPacket 399
4.23.4 Enumeration Type Documentation 399
4.23.4.1 caer_special_event_types 399
4.23.5 Function Documentation 400
4.23.5.1 caerSpecialEventGetData() 400
4.23.5.2 caerSpecialEventGetTimestamp() 400
4.23.5.3 caerSpecialEventGetTimestamp64() 401
4.23.5.4 caerSpecialEventGetType() 401
4.23.5.5 caerSpecialEventInvalidate() 402
4.23.5.6 caerSpecialEventIsValid() 402
4.23.5.7 caerSpecialEventPacketAllocate() 402
4.23.5.8 caerSpecialEventPacketFindEventByType() 403
4.23.5.9 caerSpecialEventPacketFindEventByTypeConst() 403
4.23.5.10 caerSpecialEventPacketFindValidEventByType() 403
4.23.5.11 caerSpecialEventPacketFindValidEventByTypeConst() 404
4.23.5.12 caerSpecialEventPacketFromPacketHeader() 404
4.23.5.13 caerSpecialEventPacketFromPacketHeaderConst() 405
4.23.5.14 caerSpecialEventPacketGetEvent() 405

4.23.5.15 caerSpecialEventPacketGetEventConst() 405
4.23.5.16 caerSpecialEventSetData() 406
4.23.5.17 caerSpecialEventSetTimestamp() 406
4.23.5.18 caerSpecialEventSetType() 406
4.23.5.19 caerSpecialEventValidate() 407
4.23.5.20 PACKED_STRUCT() [1/2] 407
4.23.5.21 PACKED_STRUCT() [2/2] 407
4.24 events/spike.h File Reference 408
4.24.1 Detailed Description 409
4.24.2 Macro Definition Documentation 409
4.24.2.1 CAER_SPIKE_CONST_ITERATOR_ALL_START 409
4.24.2.2 CAER_SPIKE_CONST_ITERATOR_VALID_START 410
4.24.2.3 CAER_SPIKE_CONST_REVERSE_ITERATOR_ALL_START 410
4.24.2.4 CAER_SPIKE_CONST_REVERSE_ITERATOR_VALID_START 410
4.24.2.5 CAER_SPIKE_ITERATOR_ALL_END 411
4.24.2.6 CAER_SPIKE_ITERATOR_ALL_START 411
4.24.2.7 CAER_SPIKE_ITERATOR_VALID_END 411
4.24.2.8 CAER_SPIKE_ITERATOR_VALID_START 411
4.24.2.9 CAER_SPIKE_REVERSE_ITERATOR_ALL_END 412
4.24.2.10 CAER_SPIKE_REVERSE_ITERATOR_ALL_START 412
4.24.2.11 CAER_SPIKE_REVERSE_ITERATOR_VALID_END 412
4.24.2.12 CAER_SPIKE_REVERSE_ITERATOR_VALID_START 412
4.24.2.13 SPIKE_CHIP_ID_MASK 413
4.24.2.14 SPIKE_CHIP_ID_SHIFT 413
4.24.2.15 SPIKE_NEURON_ID_MASK 413
4.24.2.16 SPIKE_NEURON_ID_SHIFT 413
4.24.2.17 SPIKE_SOURCE_CORE_ID_MASK 413
4.24.2.18 SPIKE_SOURCE_CORE_ID_SHIFT 413
4.24.3 Typedef Documentation 413
4.24.3.1 caerSpikeEvent 414

4.24.3.2 caerSpikeEventPacket 414
4.24.4 Function Documentation 414
4.24.4.1 caerSpikeEventGetChipID() 414
4.24.4.2 caerSpikeEventGetNeuronID() 414
4.24.4.3 caerSpikeEventGetSourceCoreID() 415
4.24.4.4 caerSpikeEventGetTimestamp() 415
4.24.4.5 caerSpikeEventGetTimestamp64() 415
4.24.4.6 caerSpikeEventInvalidate() 416
4.24.4.7 caerSpikeEventIsValid() 416
4.24.4.8 caerSpikeEventPacketAllocate() 416
4.24.4.9 caerSpikeEventPacketFromPacketHeader() 417
4.24.4.10 caerSpikeEventPacketFromPacketHeaderConst() 417
4.24.4.11 caerSpikeEventPacketGetEvent() 418
4.24.4.12 caerSpikeEventPacketGetEventConst() 418
4.24.4.13 caerSpikeEventSetChipID() 418
4.24.4.14 caerSpikeEventSetNeuronID() 420
4.24.4.15 caerSpikeEventSetSourceCoreID() 420
4.24.4.16 caerSpikeEventSetTimestamp() 420
4.24.4.17 caerSpikeEventValidate() 421
4.24.4.18 PACKED_STRUCT() [1/2] 421
4.24.4.19 PACKED_STRUCT() [2/2] 421
4.25 filters/dvs_noise.h File Reference 421
4.25.1 Detailed Description 422
4.25.2 Macro Definition Documentation 423
4.25.2.1 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_CHECK_POLARITY 423
4.25.2.2 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_ENABLE 423
4.25.2.3 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS 423
4.25.2.4 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS_OFF 423
4.25.2.5 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS_ON 423
4.25.2.6 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MAX 423

4.25.2.7 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MIN 424
4.25.2.8 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TIME 424
4.25.2.9 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TWO_LEVELS 424
4.25.2.10 CAER_FILTER_DVS_HOTPIXEL_COUNT 424
4.25.2.11 CAER_FILTER_DVS_HOTPIXEL_ENABLE 424
4.25.2.12 CAER_FILTER_DVS_HOTPIXEL_LEARN 424
4.25.2.13 CAER_FILTER_DVS_HOTPIXEL_STATISTICS 424
4.25.2.14 CAER_FILTER_DVS_HOTPIXEL_STATISTICS_OFF 425
4.25.2.15 CAER_FILTER_DVS_HOTPIXEL_STATISTICS_ON 425
4.25.2.16 CAER_FILTER_DVS_HOTPIXEL_TIME 425
4.25.2.17 CAER_FILTER_DVS_LOG_LEVEL 425
4.25.2.18 CAER_FILTER_DVS_REFRACTORY_PERIOD_ENABLE 425
4.25.2.19 CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS 425
4.25.2.20 CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS_OFF 425
4.25.2.21 CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS_ON 425
4.25.2.22 CAER_FILTER_DVS_REFRACTORY_PERIOD_TIME 426
4.25.2.23 CAER_FILTER_DVS_RESET 426
4.25.3 Typedef Documentation 426
4.25.3.1 caerFilterDVSNoise 426
4.25.3.2 caerFilterDVSPixel 426
4.25.4 Function Documentation 426
4.25.4.1 caerFilterDVSNoiseApply() 426
4.25.4.2 caerFilterDVSNoiseConfigGet() 427
4.25.4.3 caerFilterDVSNoiseConfigSet() 427
4.25.4.4 caerFilterDVSNoiseDestroy() 427
4.25.4.5 caerFilterDVSNoiseGetHotPixels() 428
4.25.4.6 caerFilterDVSNoiseInitialize() 428
4.25.4.7 caerFilterDVSNoiseStatsApply() 429
4.26 frame_utils.h File Reference 429
4.26.1 Detailed Description 429

4.27 libcaer.h File Reference 430
4.27.1 Detailed Description 431
4.27.2 Macro Definition Documentation 431
4.27.2.1 CLEAR_NUMBITS16 431
4.27.2.2 CLEAR_NUMBITS32 431
4.27.2.3 CLEAR_NUMBITS8 431
4.27.2.4 GET_NUMBITS16 432
4.27.2.5 GET_NUMBITS32 432
4.27.2.6 GET_NUMBITS8 432
4.27.2.7 I16T 432
4.27.2.8 I32T 432
4.27.2.9 I64T 432
4.27.2.10 I8T 433
4.27.2.11 LIBCAER_HAVE_OPENCV 433
4.27.2.12 LIBCAER_HAVE_SERIALDEV 433
4.27.2.13 LIBCAER_NAME_STRING 433
4.27.2.14 LIBCAER_VERSION 433
4.27.2.15 LIBCAER_VERSION_STRING 433
4.27.2.16 MASK_NUMBITS32 433
4.27.2.17 MASK_NUMBITS64 434
4.27.2.18 SET_NUMBITS16 434
4.27.2.19 SET_NUMBITS32 434
4.27.2.20 SET_NUMBITS8 434
4.27.2.21 SWAP_VAR 434
4.27.2.22 U16T 434
4.27.2.23 U32T 435
4.27.2.24 U64T 435
4.27.2.25 U8T 435
4.27.3 Enumeration Type Documentation 435
4.27.3.1 caer_error_codes 435

4.27.4 Function Documentation 435
4.27.4.1 caerByteArrayToInteger() 435
4.27.4.2 caerIntegerToByteArray() 436
4.27.4.3 caerStrEquals() 436
4.27.4.4 caerStrEqualsUpTo() 437
4.28 log.h File Reference 437
4.28.1 Detailed Description 438
4.28.2 Enumeration Type Documentation 438
4.28.2.1 caer_log_level 438
4.28.3 Function Documentation 438
4.28.3.1 caerLog() 438
4.28.3.2 caerLogDisable() 438
4.28.3.3 caerLogDisabled() 439
4.28.3.4 caerLogFileDescriptorsGetFirst() 439
4.28.3.5 caerLogFileDescriptorsGetSecond() 439
4.28.3.6 caerLogFileDescriptorsSet() 439
4.28.3.7 caerLogLevelGet() 440
4.28.3.8 caerLogLevelSet() 440
4.28.3.9 caerLogVA() 440
4.28.3.10 caerLogVAFull() 441
4.29 network.h File Reference 441
4.29.1 Detailed Description 442
4.30 portable_endian.h File Reference 442
4.30.1 Detailed Description 442

Index **443**

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

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

Chapter 2

File Index

2.1 File List

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

frame_utils.h . . . 429
libcaer.h . . . 430
log.h . . . 437
network.h . . . 441
portable_endian.h . . . 442
ringbuffer.h . . . **??**
devices/davis.h . . . 13
devices/device.h . . . 123
devices/device_discover.h . . . 131
devices/dvs128.h . . . 133
devices/dynapse.h . . . 137
devices/edvs.h . . . 162
devices/serial.h . . . 166
devices/usb.h . . . 168
events/common.h . . . 169
events/config.h . . . 187
events/ear.h . . . 200
events/frame.h . . . 213
events/imu6.h . . . 243
events/imu9.h . . . 259
events/matrix4x4.h . . . 278
events/packetContainer.h . . . 304
events/point1d.h . . . 312
events/point2d.h . . . 325
events/point3d.h . . . 338
events/point4d.h . . . 353
events/polarity.h . . . 368
events/sample.h . . . 381
events/special.h . . . 392
events/spike.h . . . 408
filters/dvs_noise.h . . . 421

Chapter 3

Data Structure Documentation

3.1 caer_bias_coarsefine Struct Reference

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

Data Fields

- uint8_t coarseValue

  Coarse current, from 0 to 7, creates big variations in output current.
- uint8_t fineValue

  Fine current, from 0 to 255, creates small variations in output current.
- bool enabled

  Whether this bias is enabled or not.
- bool sexN

  Bias sex: true for 'N' type, false for 'P' type.
- bool typeNormal

  Bias type: true for 'Normal', false for 'Cascode'.
- bool currentLevelNormal

  Bias current level: true for 'Normal, false for 'Low'.

3.1.1 Detailed Description

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

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

- devices/davis.h

3.2 caer_bias_dynapse Struct Reference

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

Data Fields

- uint8_t biasAddress

    Address of bias to configure, see DYNAPSE_CONFIG_BIAS_∗ defines.
- uint8_t coarseValue

    Coarse current, from 0 to 7, creates big variations in output current.
- uint8_t fineValue

    Fine current, from 0 to 255, creates small variations in output current.
- bool enabled

    Whether this bias is enabled or not.
- bool sexN

    Bias sex: true for 'N' type, false for 'P' type.
- bool typeNormal

    Bias type: true for 'Normal', false for 'Cascode'.
- bool biasHigh

    Bias current level: true for 'HighBias', false for 'LowBias'.

3.2.1 Detailed Description

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

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

- devices/dynapse.h

3.3 caer_bias_shiftedsource Struct Reference

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

Data Fields

- uint8_t refValue

    Shifted-source bias level, from 0 to 63.
- uint8_t regValue

    Shifted-source bias current for buffer amplifier, from 0 to 63.
- enum caer_bias_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').

3.3.1 Detailed Description

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

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

- devices/davis.h

3.4 caer_bias_vdac Struct Reference

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

Data Fields

- uint8_t voltageValue

  Voltage, between 0 and 63, as a fraction of 1/64th of VDD=3.3V.
- uint8_t currentValue

  Current, between 0 and 7, that drives the voltage.

3.4.1 Detailed Description

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

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

- devices/davis.h

3.5 caer_davis_info Struct Reference

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

Data Fields

- int16_t deviceID

  Unique device identifier. Also 'source' for events.
- char deviceSerialNumber [8+1]

  Device serial number.
- uint8_t deviceUSBBusNumber

  Device USB bus number.
- uint8_t deviceUSBDeviceAddress

  Device USB device address.
- char ∗ deviceString
- int16_t firmwareVersion

  USB firmware version.
- int16_t logicVersion

  Logic (FPGA/CPLD) version.
- int16_t chipID

  Chip identifier/type.
- bool deviceIsMaster

  Whether the device is a time-stamp master or slave.
- bool muxHasStatistics

  Feature test: Multiplexer statistics support (event drops).
- int16_t dvsSizeX

DVS X axis resolution.
- int16_t dvsSizeY

  DVS Y axis resolution.
- bool dvsHasPixelFilter

  Feature test: DVS pixel-level filtering.
- bool dvsHasBackgroundActivityFilter

  Feature test: DVS Background Activity filter (and Refractory Period filter).
- bool dvsHasROIFilter

  Feature test: DVS ROI filter.
- bool dvsHasSkipFilter

  Feature test: DVS event skip filter.
- bool dvsHasPolarityFilter

  Feature test: DVS polarity suppression filter.
- bool dvsHasStatistics

  Feature test: DVS statistics support.
- 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.
- enum caer_davis_imu_types imuType

  IMU chip type on device.
- bool extInputHasGenerator

  Feature test: External Input module supports Signal-Generation.

3.5.1 Detailed Description

DAVIS device-related information.

3.5.2 Field Documentation

3.5.2.1 deviceString

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

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

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

- devices/davis.h

3.6 caer_device_discovery_result Struct Reference

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

Data Fields

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

3.6.1 Detailed Description

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

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

- devices/device_discover.h

3.7 caer_dvs128_info Struct Reference

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

Data Fields

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

3.7.1 Detailed Description

DVS128 device-related information.

3.7.2 Field Documentation

3.7.2.1 deviceString

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

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

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

- devices/dvs128.h

3.8 caer_dynapse_info Struct Reference

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

Data Fields

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

3.8.1 Detailed Description

Dynap-se device-related information.

3.8.2 Field Documentation

3.8.2.1 deviceString

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

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

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

- devices/dynapse.h

3.9 caer_edvs_info Struct Reference

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

Data Fields

- int16_t deviceID

  Unique device identifier. Also 'source' for events.
- char ∗ deviceString
- bool 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.
- char serialPortName [64]

  Connected serial port name (OS-specific).
- uint32_t serialBaudRate

  Serial connection baud-rate.

3.9.1 Detailed Description

EDVS device-related information.

3.9.2 Field Documentation

3.9.2.1 deviceString

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

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

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

- devices/edvs.h

3.10 caer_filter_dvs_pixel Struct Reference

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

Data Fields

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

3.10.1 Detailed Description

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

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

- filters/dvs_noise.h

Chapter 4

File Documentation

4.1 devices/davis.h File Reference

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

Data Structures

- struct caer_davis_info
- struct caer_bias_vdac
- struct caer_bias_coarsefine
- struct caer_bias_shiftedsource

Macros

- #define CAER_DEVICE_DAVIS_FX2 1
- #define CAER_DEVICE_DAVIS_FX3 2
- #define CAER_DEVICE_DAVIS 4
- #define CAER_DEVICE_DAVIS_RPI 6
- #define DAVIS_CHIP_DAVIS240A 0
- #define DAVIS_CHIP_DAVIS240B 1
- #define DAVIS_CHIP_DAVIS240C 2
- #define DAVIS_CHIP_DAVIS128 3
- #define DAVIS_CHIP_DAVIS346A 4
- #define DAVIS_CHIP_DAVIS346B 5
- #define DAVIS_CHIP_DAVIS640 6
- #define DAVIS_CHIP_DAVIS640H 7
- #define DAVIS_CHIP_DAVIS208 8
- #define DAVIS_CHIP_DAVIS346C 9
- #define DAVIS_CONFIG_MUX 0
- #define DAVIS_CONFIG_DVS 1
- #define DAVIS_CONFIG_APS 2

- #define DAVIS_CONFIG_IMU 3
- #define DAVIS_CONFIG_EXTINPUT 4
- #define DAVIS_CONFIG_BIAS 5
- #define DAVIS_CONFIG_CHIP 5
- #define DAVIS_CONFIG_SYSINFO 6
- #define DAVIS_CONFIG_USB 9
- #define DAVIS_CONFIG_DDRAER 9
- #define DAVIS_CONFIG_MUX_RUN 0
- #define DAVIS_CONFIG_MUX_TIMESTAMP_RUN 1
- #define DAVIS_CONFIG_MUX_TIMESTAMP_RESET 2
- #define DAVIS_CONFIG_MUX_RUN_CHIP 3
- #define DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL 4
- #define DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL 5
- #define DAVIS_CONFIG_MUX_HAS_STATISTICS 80
- #define DAVIS_CONFIG_MUX_STATISTICS_EXTINPUT_DROPPED 81
- #define DAVIS_CONFIG_MUX_STATISTICS_DVS_DROPPED 83
- #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_WAIT_ON_TRANSFER_STALL 4
- #define DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL 5
- #define DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER 10
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW 11
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN 12
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW 13
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN 14
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW 15
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN 16
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW 17
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN 18
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW 19
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN 20
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW 21
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN 22
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW 23
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN 24
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW 25
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN 26
- #define DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER 30
- #define DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY 31
- #define DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_TIME 32
- #define DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD 33
- #define DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD_TIME 34
- #define DAVIS_CONFIG_DVS_HAS_ROI_FILTER 40
- #define DAVIS_CONFIG_DVS_FILTER_ROI_START_COLUMN 41
- #define DAVIS_CONFIG_DVS_FILTER_ROI_START_ROW 42
- #define DAVIS_CONFIG_DVS_FILTER_ROI_END_COLUMN 43
- #define DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW 44
- #define DAVIS_CONFIG_DVS_HAS_SKIP_FILTER 50
- #define DAVIS_CONFIG_DVS_FILTER_SKIP_EVENTS 51
- #define DAVIS_CONFIG_DVS_FILTER_SKIP_EVENTS_EVERY 52
- #define DAVIS_CONFIG_DVS_HAS_POLARITY_FILTER 60
- #define DAVIS_CONFIG_DVS_FILTER_POLARITY_FLATTEN 61
- #define DAVIS_CONFIG_DVS_FILTER_POLARITY_SUPPRESS 62

- #define DAVIS_CONFIG_DVS_FILTER_POLARITY_SUPPRESS_TYPE 63
- #define DAVIS_CONFIG_DVS_HAS_STATISTICS 80
- #define DAVIS_CONFIG_DVS_STATISTICS_EVENTS_ROW 81
- #define DAVIS_CONFIG_DVS_STATISTICS_EVENTS_COLUMN 83
- #define DAVIS_CONFIG_DVS_STATISTICS_EVENTS_DROPPED 85
- #define DAVIS_CONFIG_DVS_STATISTICS_FILTERED_PIXELS 87
- #define DAVIS_CONFIG_DVS_STATISTICS_FILTERED_BACKGROUND_ACTIVITY 89
- #define DAVIS_CONFIG_DVS_STATISTICS_FILTERED_REFRACTORY_PERIOD 91
- #define DAVIS_CONFIG_DVS_FILTER_PIXEL_AUTO_TRAIN 100
- #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_WAIT_ON_TRANSFER_STALL 5
- #define DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER 6
- #define DAVIS_CONFIG_APS_GLOBAL_SHUTTER 7
- #define DAVIS_CONFIG_APS_START_COLUMN_0 8
- #define DAVIS_CONFIG_APS_START_ROW_0 9
- #define DAVIS_CONFIG_APS_END_COLUMN_0 10
- #define DAVIS_CONFIG_APS_END_ROW_0 11
- #define DAVIS_CONFIG_APS_EXPOSURE 12
- #define DAVIS_CONFIG_APS_FRAME_INTERVAL 13
- #define DAVIS640H_CONFIG_APS_TRANSFER 14
- #define DAVIS640H_CONFIG_APS_RSFDSETTLE 15
- #define DAVIS640H_CONFIG_APS_GSPDRESET 16
- #define DAVIS640H_CONFIG_APS_GSRESETFALL 17
- #define DAVIS640H_CONFIG_APS_GSTXFALL 18
- #define DAVIS640H_CONFIG_APS_GSFDRESET 19
- #define DAVIS_CONFIG_APS_SNAPSHOT 100
- #define DAVIS_CONFIG_APS_AUTOEXPOSURE 101
- #define DAVIS_CONFIG_APS_FRAME_MODE 102
- #define DAVIS_CONFIG_IMU_TYPE 0
- #define DAVIS_CONFIG_IMU_ORIENTATION_INFO 1
- #define DAVIS_CONFIG_IMU_RUN_ACCELEROMETER 2
- #define DAVIS_CONFIG_IMU_RUN_GYROSCOPE 3
- #define DAVIS_CONFIG_IMU_RUN_TEMPERATURE 4
- #define DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER 5
- #define DAVIS_CONFIG_IMU_ACCEL_DLPF 6
- #define **DAVIS_CONFIG_IMU_DIGITAL_LOW_PASS_FILTER** DAVIS_CONFIG_IMU_ACCEL_DLPF
- #define DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE 7
- #define DAVIS_CONFIG_IMU_GYRO_DLPF 9
- #define DAVIS_CONFIG_IMU_GYRO_FULL_SCALE 10
- #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 10
- #define DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR 11
- #define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY 12
- #define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL 13
- #define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH 14
- #define DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_RISING_EDGE 15

- #define DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_FALLING_EDGE 16
- #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_SYSINFO_USB_CLOCK 5
- #define DAVIS_CONFIG_SYSINFO_CLOCK_DEVIATION 6
- #define DAVIS_CONFIG_USB_RUN 0
- #define DAVIS_CONFIG_USB_EARLY_PACKET_DELAY 1
- #define DAVIS_CONFIG_DDRAER_RUN 0

- #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_DAVIS640H(chipID) ((chipID) == DAVIS_CHIP_DAVIS640H)

- #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_PIXINVBN 13
- #define DAVIS128_CONFIG_BIAS_PRBP 14
- #define DAVIS128_CONFIG_BIAS_PRSFBP 15
- #define DAVIS128_CONFIG_BIAS_REFRBP 16
- #define DAVIS128_CONFIG_BIAS_READOUTBUFBP 17
- #define DAVIS128_CONFIG_BIAS_APSROSFBN 18
- #define DAVIS128_CONFIG_BIAS_ADCCOMPBP 19
- #define DAVIS128_CONFIG_BIAS_COLSELLOWBN 20
- #define DAVIS128_CONFIG_BIAS_DACBUFBP 21
- #define DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN 22
- #define DAVIS128_CONFIG_BIAS_AEPDBN 23
- #define DAVIS128_CONFIG_BIAS_AEPUXBP 24
- #define DAVIS128_CONFIG_BIAS_AEPUYBP 25
- #define DAVIS128_CONFIG_BIAS_IFREFRBN 26

- #define DAVIS128_CONFIG_BIAS_IFTHRBN 27
- #define DAVIS128_CONFIG_BIAS_BIASBUFFER 34
- #define DAVIS128_CONFIG_BIAS_SSP 35
- #define DAVIS128_CONFIG_BIAS_SSN 36

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

- #define DAVIS208_CONFIG_BIAS_APSOVERFLOWLEVEL 0
- #define DAVIS208_CONFIG_BIAS_APSCAS 1
- #define DAVIS208_CONFIG_BIAS_ADCREFHIGH 2
- #define DAVIS208_CONFIG_BIAS_ADCREFLOW 3
- #define DAVIS208_CONFIG_BIAS_RESETHIGHPASS 6
- #define DAVIS208_CONFIG_BIAS_REFSS 7
- #define DAVIS208_CONFIG_BIAS_LOCALBUFBN 8
- #define DAVIS208_CONFIG_BIAS_PADFOLLBN 9
- #define DAVIS208_CONFIG_BIAS_DIFFBN 10
- #define DAVIS208_CONFIG_BIAS_ONBN 11
- #define DAVIS208_CONFIG_BIAS_OFFBN 12
- #define DAVIS208_CONFIG_BIAS_PIXINVBN 13
- #define DAVIS208_CONFIG_BIAS_PRBP 14
- #define DAVIS208_CONFIG_BIAS_PRSFBP 15
- #define DAVIS208_CONFIG_BIAS_REFRBP 16
- #define DAVIS208_CONFIG_BIAS_READOUTBUFBP 17
- #define DAVIS208_CONFIG_BIAS_APSROSFBN 18
- #define DAVIS208_CONFIG_BIAS_ADCCOMPBP 19
- #define DAVIS208_CONFIG_BIAS_COLSELLOWBN 20
- #define DAVIS208_CONFIG_BIAS_DACBUFBP 21
- #define DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN 22
- #define DAVIS208_CONFIG_BIAS_AEPDBN 23
- #define DAVIS208_CONFIG_BIAS_AEPUXBP 24
- #define DAVIS208_CONFIG_BIAS_AEPUYBP 25
- #define DAVIS208_CONFIG_BIAS_IFREFRBN 26
- #define DAVIS208_CONFIG_BIAS_IFTHRBN 27
- #define DAVIS208_CONFIG_BIAS_REGBIASBP 28

- #define DAVIS208_CONFIG_BIAS_REFSSBN 30
- #define DAVIS208_CONFIG_BIAS_BIASBUFFER 34
- #define DAVIS208_CONFIG_BIAS_SSP 35
- #define DAVIS208_CONFIG_BIAS_SSN 36

- #define DAVIS208_CONFIG_CHIP_DIGITALMUX0 128
- #define DAVIS208_CONFIG_CHIP_DIGITALMUX1 129
- #define DAVIS208_CONFIG_CHIP_DIGITALMUX2 130
- #define DAVIS208_CONFIG_CHIP_DIGITALMUX3 131
- #define DAVIS208_CONFIG_CHIP_ANALOGMUX0 132
- #define DAVIS208_CONFIG_CHIP_ANALOGMUX1 133
- #define DAVIS208_CONFIG_CHIP_ANALOGMUX2 134
- #define DAVIS208_CONFIG_CHIP_BIASMUX0 135
- #define DAVIS208_CONFIG_CHIP_RESETCALIBNEURON 136
- #define DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON 137
- #define DAVIS208_CONFIG_CHIP_RESETTESTPIXEL 138
- #define DAVIS208_CONFIG_CHIP_AERNAROW 140
- #define DAVIS208_CONFIG_CHIP_USEAOUT 141
- #define DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER 142
- #define DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER 143
- #define DAVIS208_CONFIG_CHIP_SELECTPREAMPAVG 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_APSOVERFLOWLEVELBN 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_APSOVERFLOWLEVEL 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_APSOVERFLOWLEVEL 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 DAVIS640H_CONFIG_BIAS_APSCAS 0
- #define DAVIS640H_CONFIG_BIAS_OVG1LO 1
- #define DAVIS640H_CONFIG_BIAS_OVG2LO 2
- #define DAVIS640H_CONFIG_BIAS_TX2OVG2HI 3
- #define DAVIS640H_CONFIG_BIAS_GND07 4
- #define DAVIS640H_CONFIG_BIAS_ADCTESTVOLTAGE 5
- #define DAVIS640H_CONFIG_BIAS_ADCREFHIGH 6
- #define DAVIS640H_CONFIG_BIAS_ADCREFLOW 7
- #define DAVIS640H_CONFIG_BIAS_IFREFRBN 8
- #define DAVIS640H_CONFIG_BIAS_IFTHRBN 9
- #define DAVIS640H_CONFIG_BIAS_LOCALBUFBN 10
- #define DAVIS640H_CONFIG_BIAS_PADFOLLBN 11
- #define DAVIS640H_CONFIG_BIAS_PIXINVBN 13
- #define DAVIS640H_CONFIG_BIAS_DIFFBN 14
- #define DAVIS640H_CONFIG_BIAS_ONBN 15
- #define DAVIS640H_CONFIG_BIAS_OFFBN 16
- #define DAVIS640H_CONFIG_BIAS_PRBP 17
- #define DAVIS640H_CONFIG_BIAS_PRSFBP 18
- #define DAVIS640H_CONFIG_BIAS_REFRBP 19
- #define DAVIS640H_CONFIG_BIAS_ARRAYBIASBUFFERBN 20
- #define DAVIS640H_CONFIG_BIAS_ARRAYLOGICBUFFERBN 22
- #define DAVIS640H_CONFIG_BIAS_FALLTIMEBN 23
- #define DAVIS640H_CONFIG_BIAS_RISETIMEBP 24
- #define DAVIS640H_CONFIG_BIAS_READOUTBUFBP 25
- #define DAVIS640H_CONFIG_BIAS_APSROSFBN 26
- #define DAVIS640H_CONFIG_BIAS_ADCCOMPBP 27
- #define DAVIS640H_CONFIG_BIAS_DACBUFBP 28
- #define DAVIS640H_CONFIG_BIAS_LCOLTIMEOUTBN 30
- #define DAVIS640H_CONFIG_BIAS_AEPDBN 31
- #define DAVIS640H_CONFIG_BIAS_AEPUXBP 32
- #define DAVIS640H_CONFIG_BIAS_AEPUYBP 33
- #define DAVIS640H_CONFIG_BIAS_BIASBUFFER 34
- #define DAVIS640H_CONFIG_BIAS_SSP 35
- #define DAVIS640H_CONFIG_BIAS_SSN 36

- #define DAVIS640H_CONFIG_CHIP_DIGITALMUX0 128
- #define DAVIS640H_CONFIG_CHIP_DIGITALMUX1 129
- #define DAVIS640H_CONFIG_CHIP_DIGITALMUX2 130
- #define DAVIS640H_CONFIG_CHIP_DIGITALMUX3 131
- #define DAVIS640H_CONFIG_CHIP_ANALOGMUX0 132
- #define DAVIS640H_CONFIG_CHIP_ANALOGMUX1 133
- #define DAVIS640H_CONFIG_CHIP_ANALOGMUX2 134
- #define DAVIS640H_CONFIG_CHIP_BIASMUX0 135
- #define DAVIS640H_CONFIG_CHIP_RESETCALIBNEURON 136
- #define DAVIS640H_CONFIG_CHIP_TYPENCALIBNEURON 137
- #define DAVIS640H_CONFIG_CHIP_RESETTESTPIXEL 138
- #define DAVIS640H_CONFIG_CHIP_AERNAROW 140
- #define DAVIS640H_CONFIG_CHIP_USEAOUT 141
- #define DAVIS640H_CONFIG_CHIP_SELECTGRAYCOUNTER 143
- #define DAVIS640H_CONFIG_CHIP_TESTADC 144
- #define DAVIS640H_CONFIG_CHIP_ADJUSTOVG1LO 145
- #define DAVIS640H_CONFIG_CHIP_ADJUSTOVG2LO 146
- #define DAVIS640H_CONFIG_CHIP_ADJUSTTX2OVG2HI 147

Enumerations

- enum caer_davis_aps_frame_modes { **APS_FRAME_DEFAULT** = 0, **APS_FRAME_GRAYSCALE** = 1, **A↩ PS_FRAME_ORIGINAL** = 2 }
- enum caer_davis_imu_types { **IMU_NONE** = 0, **IMU_INVENSENSE_6050_6150** = 1, **IMU_INVENSENSE↩ _9250** = 2 }
- enum caer_davis_imu_invensense_accel_scale { **ACCEL_2G** = 0, **ACCEL_4G** = 1, **ACCEL_8G** = 2, **ACC↩ EL_16G** = 3 }
- enum caer_davis_imu_invensense_gyro_scale { **GYRO_250DPS** = 0, **GYRO_500DPS** = 1, **GYRO_1000DPS** = 2, **GYRO_2000DPS** = 3 }
- enum caer_bias_shiftedsource_operating_mode { SHIFTED_SOURCE = 0, HI_Z = 1, TIED_TO_RAIL = 2 }
- enum caer_bias_shiftedsource_voltage_level { SPLIT_GATE = 0, SINGLE_DIODE = 1, DOUBLE_DIODE = 2 }

Functions

- struct caer_davis_info caerDavisInfoGet (caerDeviceHandle handle)
- uint16_t caerBiasVDACGenerate (const struct caer_bias_vdac vdacBias)
- struct caer_bias_vdac caerBiasVDACParse (const uint16_t vdacBias)
- uint16_t caerBiasCoarseFineGenerate (const struct caer_bias_coarsefine coarseFineBias)
- struct caer_bias_coarsefine caerBiasCoarseFineParse (const uint16_t coarseFineBias)
- struct caer_bias_coarsefine caerBiasCoarseFineFromCurrent (uint32_t picoAmps)
- uint32_t caerBiasCoarseFineToCurrent (struct caer_bias_coarsefine coarseFineBias)
- uint16_t caerBiasShiftedSourceGenerate (const struct caer_bias_shiftedsource shiftedSourceBias)
- struct caer_bias_shiftedsource caerBiasShiftedSourceParse (const uint16_t shiftedSourceBias)
- bool caerDavisROIConfigure (caerDeviceHandle handle, uint16_t startX, uint16_t startY, uint16_t endX, uint16_t endY)

4.1.1 Detailed Description

DAVIS specific configuration defines and information structures.

4.1.2 Macro Definition Documentation

4.1.2.1 CAER_DEVICE_DAVIS

```
#define CAER_DEVICE_DAVIS 4
```

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

4.1.2.2 CAER_DEVICE_DAVIS_FX2

```
#define CAER_DEVICE_DAVIS_FX2 1
```

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

4.1.2.3 CAER_DEVICE_DAVIS_FX3

```
#define CAER_DEVICE_DAVIS_FX3 2
```

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

4.1.2.4 CAER_DEVICE_DAVIS_RPI

```
#define CAER_DEVICE_DAVIS_RPI 6
```

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

4.1.2.5 DAVIS128_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS128_CONFIG_BIAS_ADCCOMPBP 19
```

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

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

4.1.2.6 DAVIS128_CONFIG_BIAS_ADCREFHIGH

```
#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 '`https://inivation.com/support/hardware/bi`
  for more details.

4.1.2.7 DAVIS128_CONFIG_BIAS_ADCREFLOW

```
#define DAVIS128_CONFIG_BIAS_ADCREFLOW 3
```

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

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

4.1.2.8 DAVIS128_CONFIG_BIAS_AEPDBN

```
#define DAVIS128_CONFIG_BIAS_AEPDBN 23
```

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

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

4.1.2.9 DAVIS128_CONFIG_BIAS_AEPUXBP

```
#define DAVIS128_CONFIG_BIAS_AEPUXBP 24
```

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

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

4.1.2.10 DAVIS128_CONFIG_BIAS_AEPUYBP

```
#define DAVIS128_CONFIG_BIAS_AEPUYBP 25
```

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

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

4.1.2.11 DAVIS128_CONFIG_BIAS_APSCAS

```
#define DAVIS128_CONFIG_BIAS_APSCAS 1
```

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

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

4.1.2.12 DAVIS128_CONFIG_BIAS_APSOVERFLOWLEVEL

```
#define DAVIS128_CONFIG_BIAS_APSOVERFLOWLEVEL 0
```

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

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

4.1.2.13 DAVIS128_CONFIG_BIAS_APSROSFBN

```
#define DAVIS128_CONFIG_BIAS_APSROSFBN 18
```

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

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

4.1.2.14 DAVIS128_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS128_CONFIG_BIAS_BIASBUFFER 34
```

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

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

4.1.2.15 DAVIS128_CONFIG_BIAS_COLSELLOWBN

```
#define DAVIS128_CONFIG_BIAS_COLSELLOWBN 20
```

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

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

4.1.2.16 DAVIS128_CONFIG_BIAS_DACBUFBP

```
#define DAVIS128_CONFIG_BIAS_DACBUFBP 21
```

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

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

4.1.2.17 DAVIS128_CONFIG_BIAS_DIFFBN

```
#define DAVIS128_CONFIG_BIAS_DIFFBN 10
```

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

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

4.1.2.18 DAVIS128_CONFIG_BIAS_IFREFRBN

```
#define DAVIS128_CONFIG_BIAS_IFREFRBN 26
```

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

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

4.1.2.19 DAVIS128_CONFIG_BIAS_IFTHRBN

```
#define DAVIS128_CONFIG_BIAS_IFTHRBN 27
```

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

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

4.1.2.20 DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN 22
```

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

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

4.1.2.21 DAVIS128_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS128_CONFIG_BIAS_LOCALBUFBN 8
```

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

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

4.1.2.22 DAVIS128_CONFIG_BIAS_OFFBN

```
#define DAVIS128_CONFIG_BIAS_OFFBN 12
```

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

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

4.1.2.23 DAVIS128_CONFIG_BIAS_ONBN

```
#define DAVIS128_CONFIG_BIAS_ONBN 11
```

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

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

4.1.2.24 DAVIS128_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS128_CONFIG_BIAS_PADFOLLBN 9
```

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

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

4.1.2.25 DAVIS128_CONFIG_BIAS_PIXINVBN

```
#define DAVIS128_CONFIG_BIAS_PIXINVBN 13
```

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

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

4.1.2.26 DAVIS128_CONFIG_BIAS_PRBP

```
#define DAVIS128_CONFIG_BIAS_PRBP 14
```

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

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

4.1.2.27 DAVIS128_CONFIG_BIAS_PRSFBP

```
#define DAVIS128_CONFIG_BIAS_PRSFBP 15
```

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

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

4.1.2.28 DAVIS128_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS128_CONFIG_BIAS_READOUTBUFBP 17
```

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

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

4.1.2.29 DAVIS128_CONFIG_BIAS_REFRBP

```
#define DAVIS128_CONFIG_BIAS_REFRBP 16
```

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

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

4.1.2.30 DAVIS128_CONFIG_BIAS_SSN

```
#define DAVIS128_CONFIG_BIAS_SSN 36
```

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

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

4.1.2.31 DAVIS128_CONFIG_BIAS_SSP

```
#define DAVIS128_CONFIG_BIAS_SSP 35
```

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

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

4.1.2.32 DAVIS128_CONFIG_CHIP_AERNAROW

```
#define DAVIS128_CONFIG_CHIP_AERNAROW 140
```

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

4.1.2.33 DAVIS128_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS128_CONFIG_CHIP_ANALOGMUX0 132
```

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

4.1.2.34 DAVIS128_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS128_CONFIG_CHIP_ANALOGMUX1 133
```

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

4.1.2.35 DAVIS128_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS128_CONFIG_CHIP_ANALOGMUX2 134
```

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

4.1.2.36 DAVIS128_CONFIG_CHIP_BIASMUX0

```
#define DAVIS128_CONFIG_CHIP_BIASMUX0 135
```

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

4.1.2.37 DAVIS128_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS128_CONFIG_CHIP_DIGITALMUX0 128
```

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

4.1.2.38 DAVIS128_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS128_CONFIG_CHIP_DIGITALMUX1 129
```

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

4.1.2.39 DAVIS128_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS128_CONFIG_CHIP_DIGITALMUX2 130
```

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

4.1.2.40 DAVIS128_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS128_CONFIG_CHIP_DIGITALMUX3 131
```

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

4.1.2.41 DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

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

4.1.2.42 DAVIS128_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS128_CONFIG_CHIP_RESETCALIBNEURON 136
```

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

4.1.2.43 DAVIS128_CONFIG_CHIP_RESETTESTPIXEL

```
#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.

4.1.2.44 DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS128_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

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

4.1.2.45 DAVIS128_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS128_CONFIG_CHIP_TYPENCALIBNEURON 137
```

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

4.1.2.46 DAVIS128_CONFIG_CHIP_USEAOUT

```
#define DAVIS128_CONFIG_CHIP_USEAOUT 141
```

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

4.1.2.47 DAVIS208_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS208_CONFIG_BIAS_ADCCOMPBP 19
```

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

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

4.1.2.48 DAVIS208_CONFIG_BIAS_ADCREFHIGH

```
#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 'https://inivation.com/support/hardware/bi
for more details.

4.1.2.49 DAVIS208_CONFIG_BIAS_ADCREFLOW

```
#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 'https://inivation.com/support/hardware/bi
for more details.

4.1.2.50 DAVIS208_CONFIG_BIAS_AEPDBN

```
#define DAVIS208_CONFIG_BIAS_AEPDBN 23
```

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

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

4.1.2.51 DAVIS208_CONFIG_BIAS_AEPUXBP

```
#define DAVIS208_CONFIG_BIAS_AEPUXBP 24
```

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

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

4.1.2.52 DAVIS208_CONFIG_BIAS_AEPUYBP

```
#define DAVIS208_CONFIG_BIAS_AEPUYBP 25
```

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

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

4.1.2.53 DAVIS208_CONFIG_BIAS_APSCAS

```
#define DAVIS208_CONFIG_BIAS_APSCAS 1
```

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

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

4.1.2.54 DAVIS208_CONFIG_BIAS_APSOVERFLOWLEVEL

```
#define DAVIS208_CONFIG_BIAS_APSOVERFLOWLEVEL 0
```

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

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

4.1.2.55 DAVIS208_CONFIG_BIAS_APSROSFBN

```
#define DAVIS208_CONFIG_BIAS_APSROSFBN 18
```

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

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

4.1.2.56 DAVIS208_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS208_CONFIG_BIAS_BIASBUFFER 34
```

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

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

4.1.2.57 DAVIS208_CONFIG_BIAS_COLSELLOWBN

```
#define DAVIS208_CONFIG_BIAS_COLSELLOWBN 20
```

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

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

4.1.2.58 DAVIS208_CONFIG_BIAS_DACBUFBP

```
#define DAVIS208_CONFIG_BIAS_DACBUFBP 21
```

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

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

4.1.2.59 DAVIS208_CONFIG_BIAS_DIFFBN

```
#define DAVIS208_CONFIG_BIAS_DIFFBN 10
```

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

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

4.1.2.60 DAVIS208_CONFIG_BIAS_IFREFRBN

```
#define DAVIS208_CONFIG_BIAS_IFREFRBN 26
```

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

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

4.1.2.61 DAVIS208_CONFIG_BIAS_IFTHRBN

```
#define DAVIS208_CONFIG_BIAS_IFTHRBN 27
```

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

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

4.1.2.62 DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN 22
```

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

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

4.1.2.63 DAVIS208_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS208_CONFIG_BIAS_LOCALBUFBN 8
```

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

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

4.1.2.64 DAVIS208_CONFIG_BIAS_OFFBN

```
#define DAVIS208_CONFIG_BIAS_OFFBN 12
```

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

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

4.1.2.65 DAVIS208_CONFIG_BIAS_ONBN

```
#define DAVIS208_CONFIG_BIAS_ONBN 11
```

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

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

4.1.2.66 DAVIS208_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS208_CONFIG_BIAS_PADFOLLBN 9
```

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

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

4.1.2.67 DAVIS208_CONFIG_BIAS_PIXINVBN

```
#define DAVIS208_CONFIG_BIAS_PIXINVBN 13
```

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

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

4.1.2.68 DAVIS208_CONFIG_BIAS_PRBP

```
#define DAVIS208_CONFIG_BIAS_PRBP 14
```

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

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

4.1.2.69 DAVIS208_CONFIG_BIAS_PRSFBP

```
#define DAVIS208_CONFIG_BIAS_PRSFBP 15
```

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

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

4.1.2.70 DAVIS208_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS208_CONFIG_BIAS_READOUTBUFBP 17
```

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

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

4.1.2.71 DAVIS208_CONFIG_BIAS_REFRBP

```
#define DAVIS208_CONFIG_BIAS_REFRBP 16
```

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

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

4.1.2.72 DAVIS208_CONFIG_BIAS_REFSS

```
#define DAVIS208_CONFIG_BIAS_REFSS 7
```

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

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

4.1.2.73 DAVIS208_CONFIG_BIAS_REFSSBN

```
#define DAVIS208_CONFIG_BIAS_REFSSBN 30
```

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

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

4.1.2.74 DAVIS208_CONFIG_BIAS_REGBIASBP

```
#define DAVIS208_CONFIG_BIAS_REGBIASBP 28
```

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

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

4.1.2.75 DAVIS208_CONFIG_BIAS_RESETHIGHPASS

```
#define DAVIS208_CONFIG_BIAS_RESETHIGHPASS 6
```

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

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

4.1.2.76 DAVIS208_CONFIG_BIAS_SSN

```
#define DAVIS208_CONFIG_BIAS_SSN 36
```

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

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

4.1.2.77 DAVIS208_CONFIG_BIAS_SSP

```
#define DAVIS208_CONFIG_BIAS_SSP 35
```

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

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

4.1.2.78 DAVIS208_CONFIG_CHIP_AERNAROW

```
#define DAVIS208_CONFIG_CHIP_AERNAROW 140
```

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

4.1.2.79 DAVIS208_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS208_CONFIG_CHIP_ANALOGMUX0 132
```

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

4.1.2.80 DAVIS208_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS208_CONFIG_CHIP_ANALOGMUX1 133
```

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

4.1.2.81 DAVIS208_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS208_CONFIG_CHIP_ANALOGMUX2 134
```

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

4.1.2.82 DAVIS208_CONFIG_CHIP_BIASMUX0

```
#define DAVIS208_CONFIG_CHIP_BIASMUX0 135
```

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

4.1.2.83 DAVIS208_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS208_CONFIG_CHIP_DIGITALMUX0 128
```

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

4.1.2.84 DAVIS208_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS208_CONFIG_CHIP_DIGITALMUX1 129
```

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

4.1.2.85 DAVIS208_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS208_CONFIG_CHIP_DIGITALMUX2 130
```

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

4.1.2.86 DAVIS208_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS208_CONFIG_CHIP_DIGITALMUX3 131
```

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

4.1.2.87 DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

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

4.1.2.88 DAVIS208_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS208_CONFIG_CHIP_RESETCALIBNEURON 136
```

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

4.1.2.89 DAVIS208_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVIS208_CONFIG_CHIP_RESETTESTPIXEL 138
```

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

4.1.2.90 DAVIS208_CONFIG_CHIP_SELECTBIASREFSS

```
#define DAVIS208_CONFIG_CHIP_SELECTBIASREFSS 146
```

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

4.1.2.91 DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

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

4.1.2.92 DAVIS208_CONFIG_CHIP_SELECTHIGHPASS

```
#define DAVIS208_CONFIG_CHIP_SELECTHIGHPASS 149
```

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

4.1.2.93 DAVIS208_CONFIG_CHIP_SELECTPOSFB

```
#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.

4.1.2.94 DAVIS208_CONFIG_CHIP_SELECTPREAMPAVG

```
#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.

4.1.2.95 DAVIS208_CONFIG_CHIP_SELECTSENSE

```
#define DAVIS208_CONFIG_CHIP_SELECTSENSE 147
```

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

4.1.2.96 DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON 137
```

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

4.1.2.97 DAVIS208_CONFIG_CHIP_USEAOUT

```
#define DAVIS208_CONFIG_CHIP_USEAOUT 141
```

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

4.1.2.98 DAVIS240_CONFIG_BIAS_AEPDBN

```
#define DAVIS240_CONFIG_BIAS_AEPDBN 11
```

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

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See '`https://inivation.com/support/hardware/bi`
  for more details.

4.1.2.99 DAVIS240_CONFIG_BIAS_AEPUXBP

```
#define DAVIS240_CONFIG_BIAS_AEPUXBP 13
```

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

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

4.1.2.100 DAVIS240_CONFIG_BIAS_AEPUYBP

```
#define DAVIS240_CONFIG_BIAS_AEPUYBP 14
```

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

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

4.1.2.101 DAVIS240_CONFIG_BIAS_APSCASEPC

```
#define DAVIS240_CONFIG_BIAS_APSCASEPC 3
```

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

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

4.1.2.102 DAVIS240_CONFIG_BIAS_APSOVERFLOWLEVELBN

```
#define DAVIS240_CONFIG_BIAS_APSOVERFLOWLEVELBN 18
```

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

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

4.1.2.103 DAVIS240_CONFIG_BIAS_APSROSFBN

```
#define DAVIS240_CONFIG_BIAS_APSROSFBN 5
```

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

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

4.1.2.104 DAVIS240_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS240_CONFIG_BIAS_BIASBUFFER 19
```

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

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

4.1.2.105 DAVIS240_CONFIG_BIAS_DIFFBN

```
#define DAVIS240_CONFIG_BIAS_DIFFBN 0
```

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

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

4.1.2.106 DAVIS240_CONFIG_BIAS_DIFFCASBNC

```
#define DAVIS240_CONFIG_BIAS_DIFFCASBNC 4
```

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

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

4.1.2.107 DAVIS240_CONFIG_BIAS_IFREFRBN

```
#define DAVIS240_CONFIG_BIAS_IFREFRBN 16
```

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

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

4.1.2.108 DAVIS240_CONFIG_BIAS_IFTHRBN

```
#define DAVIS240_CONFIG_BIAS_IFTHRBN 15
```

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

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

4.1.2.109 DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN 12
```

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

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

4.1.2.110 DAVIS240_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS240_CONFIG_BIAS_LOCALBUFBN 6
```

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

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

4.1.2.111 DAVIS240_CONFIG_BIAS_OFFBN

```
#define DAVIS240_CONFIG_BIAS_OFFBN 2
```

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

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

4.1.2.112 DAVIS240_CONFIG_BIAS_ONBN

```
#define DAVIS240_CONFIG_BIAS_ONBN 1
```

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

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

4.1.2.113 DAVIS240_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS240_CONFIG_BIAS_PADFOLLBN 17
```

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

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

4.1.2.114 DAVIS240_CONFIG_BIAS_PIXINVBN

```
#define DAVIS240_CONFIG_BIAS_PIXINVBN 7
```

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

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

4.1.2.115 DAVIS240_CONFIG_BIAS_PRBP

```
#define DAVIS240_CONFIG_BIAS_PRBP 8
```

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

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

4.1.2.116 DAVIS240_CONFIG_BIAS_PRSFBP

```
#define DAVIS240_CONFIG_BIAS_PRSFBP 9
```

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

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

4.1.2.117 DAVIS240_CONFIG_BIAS_REFRBP

```
#define DAVIS240_CONFIG_BIAS_REFRBP 10
```

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

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

4.1.2.118 DAVIS240_CONFIG_BIAS_SSN

```
#define DAVIS240_CONFIG_BIAS_SSN 21
```

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

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

4.1.2.119 DAVIS240_CONFIG_BIAS_SSP

```
#define DAVIS240_CONFIG_BIAS_SSP 20
```

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

- caerBiasCoarseFineGenerate() for coarse-fine (current) biases.
- caerBiasShiftedSourceGenerate() for shifted-source biases. See '`https://inivation.com/support/hardware/bi`
  for more details.

4.1.2.120 DAVIS240_CONFIG_CHIP_AERNAROW

```
#define DAVIS240_CONFIG_CHIP_AERNAROW 140
```

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

4.1.2.121 DAVIS240_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS240_CONFIG_CHIP_ANALOGMUX0 132
```

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

4.1.2.122 DAVIS240_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS240_CONFIG_CHIP_ANALOGMUX1 133
```

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

4.1.2.123 DAVIS240_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS240_CONFIG_CHIP_ANALOGMUX2 134
```

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

4.1.2.124 DAVIS240_CONFIG_CHIP_BIASMUX0

```
#define DAVIS240_CONFIG_CHIP_BIASMUX0 135
```

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

4.1.2.125 DAVIS240_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS240_CONFIG_CHIP_DIGITALMUX0 128
```

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

4.1.2.126 DAVIS240_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS240_CONFIG_CHIP_DIGITALMUX1 129
```

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

4.1.2.127 DAVIS240_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS240_CONFIG_CHIP_DIGITALMUX2 130
```

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

4.1.2.128 DAVIS240_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS240_CONFIG_CHIP_DIGITALMUX3 131
```

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

4.1.2.129 DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

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

4.1.2.130 DAVIS240_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS240_CONFIG_CHIP_RESETCALIBNEURON 136
```

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

4.1.2.131 DAVIS240_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVIS240_CONFIG_CHIP_RESETTESTPIXEL 138
```

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

4.1.2.132 DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL

```
#define DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL 139
```

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

4.1.2.133 DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON 137
```

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

4.1.2.134 DAVIS240_CONFIG_CHIP_USEAOUT

```
#define DAVIS240_CONFIG_CHIP_USEAOUT 141
```

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

4.1.2.135 DAVIS346_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS346_CONFIG_BIAS_ADCCOMPBP 19
```

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

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

4.1.2.136 DAVIS346_CONFIG_BIAS_ADCREFHIGH

```
#define DAVIS346_CONFIG_BIAS_ADCREFHIGH 2
```

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

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

4.1.2.137 DAVIS346_CONFIG_BIAS_ADCREFLOW

```
#define DAVIS346_CONFIG_BIAS_ADCREFLOW 3
```

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

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

4.1.2.138 DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE

```
#define DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE 4
```

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

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

4.1.2.139 DAVIS346_CONFIG_BIAS_AEPDBN

```
#define DAVIS346_CONFIG_BIAS_AEPDBN 23
```

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

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

4.1.2.140 DAVIS346_CONFIG_BIAS_AEPUXBP

```
#define DAVIS346_CONFIG_BIAS_AEPUXBP 24
```

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

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

4.1.2.141 DAVIS346_CONFIG_BIAS_AEPUYBP

```
#define DAVIS346_CONFIG_BIAS_AEPUYBP 25
```

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

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

4.1.2.142 DAVIS346_CONFIG_BIAS_APSCAS

```
#define DAVIS346_CONFIG_BIAS_APSCAS 1
```

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

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

4.1.2.143 DAVIS346_CONFIG_BIAS_APSOVERFLOWLEVEL

```
#define DAVIS346_CONFIG_BIAS_APSOVERFLOWLEVEL 0
```

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

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

4.1.2.144 DAVIS346_CONFIG_BIAS_APSROSFBN

```
#define DAVIS346_CONFIG_BIAS_APSROSFBN 18
```

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

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

4.1.2.145 DAVIS346_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS346_CONFIG_BIAS_BIASBUFFER 34
```

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

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

4.1.2.146 DAVIS346_CONFIG_BIAS_COLSELLOWBN

```
#define DAVIS346_CONFIG_BIAS_COLSELLOWBN 20
```

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

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

4.1.2.147 DAVIS346_CONFIG_BIAS_DACBUFBP

```
#define DAVIS346_CONFIG_BIAS_DACBUFBP 21
```

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

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

4.1.2.148 DAVIS346_CONFIG_BIAS_DIFFBN

```
#define DAVIS346_CONFIG_BIAS_DIFFBN 10
```

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

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

4.1.2.149 DAVIS346_CONFIG_BIAS_IFREFRBN

```
#define DAVIS346_CONFIG_BIAS_IFREFRBN 26
```

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

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

4.1.2.150 DAVIS346_CONFIG_BIAS_IFTHRBN

```
#define DAVIS346_CONFIG_BIAS_IFTHRBN 27
```

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

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

4.1.2.151 DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN 22
```

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

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

4.1.2.152 DAVIS346_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS346_CONFIG_BIAS_LOCALBUFBN 8
```

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

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

4.1.2.153 DAVIS346_CONFIG_BIAS_OFFBN

```
#define DAVIS346_CONFIG_BIAS_OFFBN 12
```

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

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

4.1.2.154 DAVIS346_CONFIG_BIAS_ONBN

```
#define DAVIS346_CONFIG_BIAS_ONBN 11
```

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

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

4.1.2.155 DAVIS346_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS346_CONFIG_BIAS_PADFOLLBN 9
```

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

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

4.1.2.156 DAVIS346_CONFIG_BIAS_PIXINVBN

```
#define DAVIS346_CONFIG_BIAS_PIXINVBN 13
```

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

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

4.1.2.157 DAVIS346_CONFIG_BIAS_PRBP

```
#define DAVIS346_CONFIG_BIAS_PRBP 14
```

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

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

4.1.2.158 DAVIS346_CONFIG_BIAS_PRSFBP

```
#define DAVIS346_CONFIG_BIAS_PRSFBP 15
```

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

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

4.1.2.159 DAVIS346_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS346_CONFIG_BIAS_READOUTBUFBP 17
```

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

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

4.1.2.160 DAVIS346_CONFIG_BIAS_REFRBP

```
#define DAVIS346_CONFIG_BIAS_REFRBP 16
```

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

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

4.1.2.161 DAVIS346_CONFIG_BIAS_SSN

```
#define DAVIS346_CONFIG_BIAS_SSN 36
```

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

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

4.1.2.162 DAVIS346_CONFIG_BIAS_SSP

```
#define DAVIS346_CONFIG_BIAS_SSP 35
```

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

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

4.1.2.163 DAVIS346_CONFIG_CHIP_AERNAROW

```
#define DAVIS346_CONFIG_CHIP_AERNAROW 140
```

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

4.1.2.164 DAVIS346_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS346_CONFIG_CHIP_ANALOGMUX0 132
```

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

4.1.2.165 DAVIS346_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS346_CONFIG_CHIP_ANALOGMUX1 133
```

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

4.1.2.166 DAVIS346_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS346_CONFIG_CHIP_ANALOGMUX2 134
```

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

4.1.2.167 DAVIS346_CONFIG_CHIP_BIASMUX0

```
#define DAVIS346_CONFIG_CHIP_BIASMUX0 135
```

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

4.1.2.168 DAVIS346_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS346_CONFIG_CHIP_DIGITALMUX0 128
```

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

4.1.2.169 DAVIS346_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS346_CONFIG_CHIP_DIGITALMUX1 129
```

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

4.1.2.170 DAVIS346_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS346_CONFIG_CHIP_DIGITALMUX2 130
```

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

4.1.2.171 DAVIS346_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS346_CONFIG_CHIP_DIGITALMUX3 131
```

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

4.1.2.172 DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

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

4.1.2.173 DAVIS346_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS346_CONFIG_CHIP_RESETCALIBNEURON 136
```

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

4.1.2.174 DAVIS346_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVIS346_CONFIG_CHIP_RESETTESTPIXEL 138
```

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

4.1.2.175 DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

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

4.1.2.176 DAVIS346_CONFIG_CHIP_TESTADC

```
#define DAVIS346_CONFIG_CHIP_TESTADC 144
```

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

4.1.2.177 DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON 137
```

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

4.1.2.178 DAVIS346_CONFIG_CHIP_USEAOUT

```
#define DAVIS346_CONFIG_CHIP_USEAOUT 141
```

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

4.1.2.179 DAVIS640_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS640_CONFIG_BIAS_ADCCOMPBP 19
```

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

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

4.1.2.180 DAVIS640_CONFIG_BIAS_ADCREFHIGH

```
#define DAVIS640_CONFIG_BIAS_ADCREFHIGH 2
```

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

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

4.1.2.181 DAVIS640_CONFIG_BIAS_ADCREFLOW

```
#define DAVIS640_CONFIG_BIAS_ADCREFLOW 3
```

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

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

4.1.2.182 DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE

```
#define DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE 4
```

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

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

4.1.2.183 DAVIS640_CONFIG_BIAS_AEPDBN

```
#define DAVIS640_CONFIG_BIAS_AEPDBN 23
```

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

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

4.1.2.184 DAVIS640_CONFIG_BIAS_AEPUXBP

```
#define DAVIS640_CONFIG_BIAS_AEPUXBP 24
```

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

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

4.1.2.185 DAVIS640_CONFIG_BIAS_AEPUYBP

```
#define DAVIS640_CONFIG_BIAS_AEPUYBP 25
```

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

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

4.1.2.186 DAVIS640_CONFIG_BIAS_APSCAS

```
#define DAVIS640_CONFIG_BIAS_APSCAS 1
```

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

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

4.1.2.187 DAVIS640_CONFIG_BIAS_APSOVERFLOWLEVEL

```
#define DAVIS640_CONFIG_BIAS_APSOVERFLOWLEVEL 0
```

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

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

4.1.2.188 DAVIS640_CONFIG_BIAS_APSROSFBN

```
#define DAVIS640_CONFIG_BIAS_APSROSFBN 18
```

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

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

4.1.2.189 DAVIS640_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS640_CONFIG_BIAS_BIASBUFFER 34
```

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

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

4.1.2.190 DAVIS640_CONFIG_BIAS_COLSELLOWBN

```
#define DAVIS640_CONFIG_BIAS_COLSELLOWBN 20
```

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

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

4.1.2.191 DAVIS640_CONFIG_BIAS_DACBUFBP

```
#define DAVIS640_CONFIG_BIAS_DACBUFBP 21
```

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

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

4.1.2.192 DAVIS640_CONFIG_BIAS_DIFFBN

```
#define DAVIS640_CONFIG_BIAS_DIFFBN 10
```

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

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

4.1.2.193 DAVIS640_CONFIG_BIAS_IFREFRBN

```
#define DAVIS640_CONFIG_BIAS_IFREFRBN 26
```

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

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

4.1.2.194 DAVIS640_CONFIG_BIAS_IFTHRBN

```
#define DAVIS640_CONFIG_BIAS_IFTHRBN 27
```

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

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

4.1.2.195 DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN 22
```

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

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

4.1.2.196 DAVIS640_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS640_CONFIG_BIAS_LOCALBUFBN 8
```

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

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

4.1.2.197 DAVIS640_CONFIG_BIAS_OFFBN

```
#define DAVIS640_CONFIG_BIAS_OFFBN 12
```

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

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

4.1.2.198 DAVIS640_CONFIG_BIAS_ONBN

```
#define DAVIS640_CONFIG_BIAS_ONBN 11
```

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

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

4.1.2.199 DAVIS640_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS640_CONFIG_BIAS_PADFOLLBN 9
```

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

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

4.1.2.200 DAVIS640_CONFIG_BIAS_PIXINVBN

```
#define DAVIS640_CONFIG_BIAS_PIXINVBN 13
```

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

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

4.1.2.201 DAVIS640_CONFIG_BIAS_PRBP

```
#define DAVIS640_CONFIG_BIAS_PRBP 14
```

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

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

4.1.2.202 DAVIS640_CONFIG_BIAS_PRSFBP

```
#define DAVIS640_CONFIG_BIAS_PRSFBP 15
```

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

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

4.1.2.203 DAVIS640_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS640_CONFIG_BIAS_READOUTBUFBP 17
```

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

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

4.1.2.204 DAVIS640_CONFIG_BIAS_REFRBP

```
#define DAVIS640_CONFIG_BIAS_REFRBP 16
```

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

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

4.1.2.205 DAVIS640_CONFIG_BIAS_SSN

```
#define DAVIS640_CONFIG_BIAS_SSN 36
```

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

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

4.1.2.206 DAVIS640_CONFIG_BIAS_SSP

```
#define DAVIS640_CONFIG_BIAS_SSP 35
```

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

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

4.1.2.207 DAVIS640_CONFIG_CHIP_AERNAROW

```
#define DAVIS640_CONFIG_CHIP_AERNAROW 140
```

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

4.1.2.208 DAVIS640_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS640_CONFIG_CHIP_ANALOGMUX0 132
```

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

4.1.2.209 DAVIS640_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS640_CONFIG_CHIP_ANALOGMUX1 133
```

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

4.1.2.210 DAVIS640_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS640_CONFIG_CHIP_ANALOGMUX2 134
```

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

4.1.2.211 DAVIS640_CONFIG_CHIP_BIASMUX0

```
#define DAVIS640_CONFIG_CHIP_BIASMUX0 135
```

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

4.1.2.212 DAVIS640_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS640_CONFIG_CHIP_DIGITALMUX0 128
```

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

4.1.2.213 DAVIS640_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS640_CONFIG_CHIP_DIGITALMUX1 129
```

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

4.1.2.214 DAVIS640_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS640_CONFIG_CHIP_DIGITALMUX2 130
```

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

4.1.2.215 DAVIS640_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS640_CONFIG_CHIP_DIGITALMUX3 131
```

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

4.1.2.216 DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER

```
#define DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER 142
```

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

4.1.2.217 DAVIS640_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS640_CONFIG_CHIP_RESETCALIBNEURON 136
```

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

4.1.2.218 DAVIS640_CONFIG_CHIP_RESETTESTPIXEL

```
#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.

4.1.2.219 DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

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

4.1.2.220 DAVIS640_CONFIG_CHIP_TESTADC

```
#define DAVIS640_CONFIG_CHIP_TESTADC 144
```

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

4.1.2.221 DAVIS640_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS640_CONFIG_CHIP_TYPENCALIBNEURON 137
```

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

4.1.2.222 DAVIS640_CONFIG_CHIP_USEAOUT

```
#define DAVIS640_CONFIG_CHIP_USEAOUT 141
```

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

4.1.2.223 DAVIS640H_CONFIG_APS_GSFDRESET

```
#define DAVIS640H_CONFIG_APS_GSFDRESET 19
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS640H chip): Global Shutter FD reset time in ADCClock cycles.

4.1.2.224 DAVIS640H_CONFIG_APS_GSPDRESET

```
#define DAVIS640H_CONFIG_APS_GSPDRESET 16
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS640H chip): Global Shutter PD reset time in ADCClock cycles.

4.1.2.225 DAVIS640H_CONFIG_APS_GSRESETFALL

```
#define DAVIS640H_CONFIG_APS_GSRESETFALL 17
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS640H chip): Global Shutter Reset Fall time in ADCClock cycles.

4.1.2.226 DAVIS640H_CONFIG_APS_GSTXFALL

```
#define DAVIS640H_CONFIG_APS_GSTXFALL 18
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS640H chip): Global Shutter Transfer Fall time in ADCClock cycles.

4.1.2.227 DAVIS640H_CONFIG_APS_RSFDSETTLE

```
#define DAVIS640H_CONFIG_APS_RSFDSETTLE 15
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS640H chip): Rolling Shutter FD settle time in ADCClock cycles.

4.1.2.228 DAVIS640H_CONFIG_APS_TRANSFER

```
#define DAVIS640H_CONFIG_APS_TRANSFER 14
```

Parameter address for module DAVIS_CONFIG_APS (only for DAVIS640H chip): charge transfer time in ADCClock cycles.

4.1.2.229 DAVIS640H_CONFIG_BIAS_ADCCOMPBP

```
#define DAVIS640H_CONFIG_BIAS_ADCCOMPBP 27
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.230 DAVIS640H_CONFIG_BIAS_ADCREFHIGH

```
#define DAVIS640H_CONFIG_BIAS_ADCREFHIGH 6
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.231 DAVIS640H_CONFIG_BIAS_ADCREFLOW

```
#define DAVIS640H_CONFIG_BIAS_ADCREFLOW 7
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.232 DAVIS640H_CONFIG_BIAS_ADCTESTVOLTAGE

```
#define DAVIS640H_CONFIG_BIAS_ADCTESTVOLTAGE 5
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.233 DAVIS640H_CONFIG_BIAS_AEPDBN

```
#define DAVIS640H_CONFIG_BIAS_AEPDBN 31
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.234 DAVIS640H_CONFIG_BIAS_AEPUXBP

```
#define DAVIS640H_CONFIG_BIAS_AEPUXBP 32
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.235 DAVIS640H_CONFIG_BIAS_AEPUYBP

```
#define DAVIS640H_CONFIG_BIAS_AEPUYBP 33
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.236 DAVIS640H_CONFIG_BIAS_APSCAS

```
#define DAVIS640H_CONFIG_BIAS_APSCAS 0
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.237 DAVIS640H_CONFIG_BIAS_APSROSFBN

```
#define DAVIS640H_CONFIG_BIAS_APSROSFBN 26
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.238 DAVIS640H_CONFIG_BIAS_ARRAYBIASBUFFERBN

```
#define DAVIS640H_CONFIG_BIAS_ARRAYBIASBUFFERBN 20
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.239 DAVIS640H_CONFIG_BIAS_ARRAYLOGICBUFFERBN

```
#define DAVIS640H_CONFIG_BIAS_ARRAYLOGICBUFFERBN 22
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.240 DAVIS640H_CONFIG_BIAS_BIASBUFFER

```
#define DAVIS640H_CONFIG_BIAS_BIASBUFFER 34
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.241 DAVIS640H_CONFIG_BIAS_DACBUFBP

```
#define DAVIS640H_CONFIG_BIAS_DACBUFBP 28
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.242 DAVIS640H_CONFIG_BIAS_DIFFBN

```
#define DAVIS640H_CONFIG_BIAS_DIFFBN 14
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.243 DAVIS640H_CONFIG_BIAS_FALLTIMEBN

```
#define DAVIS640H_CONFIG_BIAS_FALLTIMEBN 23
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.244 DAVIS640H_CONFIG_BIAS_GND07

```
#define DAVIS640H_CONFIG_BIAS_GND07 4
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.245 DAVIS640H_CONFIG_BIAS_IFREFRBN

```
#define DAVIS640H_CONFIG_BIAS_IFREFRBN 8
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.246 DAVIS640H_CONFIG_BIAS_IFTHRBN

```
#define DAVIS640H_CONFIG_BIAS_IFTHRBN 9
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.247 DAVIS640H_CONFIG_BIAS_LCOLTIMEOUTBN

```
#define DAVIS640H_CONFIG_BIAS_LCOLTIMEOUTBN 30
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.248 DAVIS640H_CONFIG_BIAS_LOCALBUFBN

```
#define DAVIS640H_CONFIG_BIAS_LOCALBUFBN 10
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.249 DAVIS640H_CONFIG_BIAS_OFFBN

```
#define DAVIS640H_CONFIG_BIAS_OFFBN 16
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.250 DAVIS640H_CONFIG_BIAS_ONBN

```
#define DAVIS640H_CONFIG_BIAS_ONBN 15
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.251 DAVIS640H_CONFIG_BIAS_OVG1LO

```
#define DAVIS640H_CONFIG_BIAS_OVG1LO 1
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.252 DAVIS640H_CONFIG_BIAS_OVG2LO

```
#define DAVIS640H_CONFIG_BIAS_OVG2LO 2
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.253 DAVIS640H_CONFIG_BIAS_PADFOLLBN

```
#define DAVIS640H_CONFIG_BIAS_PADFOLLBN 11
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.254 DAVIS640H_CONFIG_BIAS_PIXINVBN

```
#define DAVIS640H_CONFIG_BIAS_PIXINVBN 13
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.255 DAVIS640H_CONFIG_BIAS_PRBP

```
#define DAVIS640H_CONFIG_BIAS_PRBP 17
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.256 DAVIS640H_CONFIG_BIAS_PRSFBP

```
#define DAVIS640H_CONFIG_BIAS_PRSFBP 18
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.257 DAVIS640H_CONFIG_BIAS_READOUTBUFBP

```
#define DAVIS640H_CONFIG_BIAS_READOUTBUFBP 25
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.258 DAVIS640H_CONFIG_BIAS_REFRBP

```
#define DAVIS640H_CONFIG_BIAS_REFRBP 19
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.259 DAVIS640H_CONFIG_BIAS_RISETIMEBP

```
#define DAVIS640H_CONFIG_BIAS_RISETIMEBP 24
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.260 DAVIS640H_CONFIG_BIAS_SSN

```
#define DAVIS640H_CONFIG_BIAS_SSN 36
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.261 DAVIS640H_CONFIG_BIAS_SSP

```
#define DAVIS640H_CONFIG_BIAS_SSP 35
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.262 DAVIS640H_CONFIG_BIAS_TX2OVG2HI

```
#define DAVIS640H_CONFIG_BIAS_TX2OVG2HI 3
```

Parameter address for module DAVIS640H_CONFIG_BIAS: DAVIS640H chip biases. Bias configuration values must be generated using the proper functions, which are:

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

4.1.2.263 DAVIS640H_CONFIG_CHIP_ADJUSTOVG1LO

```
#define DAVIS640H_CONFIG_CHIP_ADJUSTOVG1LO 145
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.264 DAVIS640H_CONFIG_CHIP_ADJUSTOVG2LO

```
#define DAVIS640H_CONFIG_CHIP_ADJUSTOVG2LO 146
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.265 DAVIS640H_CONFIG_CHIP_ADJUSTTX2OVG2HI

```
#define DAVIS640H_CONFIG_CHIP_ADJUSTTX2OVG2HI 147
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.266 DAVIS640H_CONFIG_CHIP_AERNAROW

```
#define DAVIS640H_CONFIG_CHIP_AERNAROW 140
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.267 DAVIS640H_CONFIG_CHIP_ANALOGMUX0

```
#define DAVIS640H_CONFIG_CHIP_ANALOGMUX0 132
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.268 DAVIS640H_CONFIG_CHIP_ANALOGMUX1

```
#define DAVIS640H_CONFIG_CHIP_ANALOGMUX1 133
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.269 DAVIS640H_CONFIG_CHIP_ANALOGMUX2

```
#define DAVIS640H_CONFIG_CHIP_ANALOGMUX2 134
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.270 DAVIS640H_CONFIG_CHIP_BIASMUX0

```
#define DAVIS640H_CONFIG_CHIP_BIASMUX0 135
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.271 DAVIS640H_CONFIG_CHIP_DIGITALMUX0

```
#define DAVIS640H_CONFIG_CHIP_DIGITALMUX0 128
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.272 DAVIS640H_CONFIG_CHIP_DIGITALMUX1

```
#define DAVIS640H_CONFIG_CHIP_DIGITALMUX1 129
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.273 DAVIS640H_CONFIG_CHIP_DIGITALMUX2

```
#define DAVIS640H_CONFIG_CHIP_DIGITALMUX2 130
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.274 DAVIS640H_CONFIG_CHIP_DIGITALMUX3

```
#define DAVIS640H_CONFIG_CHIP_DIGITALMUX3 131
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.275 DAVIS640H_CONFIG_CHIP_RESETCALIBNEURON

```
#define DAVIS640H_CONFIG_CHIP_RESETCALIBNEURON 136
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.276 DAVIS640H_CONFIG_CHIP_RESETTESTPIXEL

```
#define DAVIS640H_CONFIG_CHIP_RESETTESTPIXEL 138
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.277 DAVIS640H_CONFIG_CHIP_SELECTGRAYCOUNTER

```
#define DAVIS640H_CONFIG_CHIP_SELECTGRAYCOUNTER 143
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.278 DAVIS640H_CONFIG_CHIP_TESTADC

```
#define DAVIS640H_CONFIG_CHIP_TESTADC 144
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.279 DAVIS640H_CONFIG_CHIP_TYPENCALIBNEURON

```
#define DAVIS640H_CONFIG_CHIP_TYPENCALIBNEURON 137
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.280 DAVIS640H_CONFIG_CHIP_USEAOUT

```
#define DAVIS640H_CONFIG_CHIP_USEAOUT 141
```

Parameter address for module DAVIS640H_CONFIG_CHIP: DAVIS640H 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.

4.1.2.281 DAVIS_CHIP_DAVIS128

```
#define DAVIS_CHIP_DAVIS128 3
```

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

4.1.2.282 DAVIS_CHIP_DAVIS208

```
#define DAVIS_CHIP_DAVIS208 8
```

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

4.1.2.283 DAVIS_CHIP_DAVIS240A

```
#define DAVIS_CHIP_DAVIS240A 0
```

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

4.1.2.284 DAVIS_CHIP_DAVIS240B

```
#define DAVIS_CHIP_DAVIS240B 1
```

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

4.1.2.285 DAVIS_CHIP_DAVIS240C

```
#define DAVIS_CHIP_DAVIS240C 2
```

DAVIS240C chip identifier. 240x180, no color.

4.1.2.286 DAVIS_CHIP_DAVIS346A

```
#define DAVIS_CHIP_DAVIS346A 4
```

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

4.1.2.287 DAVIS_CHIP_DAVIS346B

```
#define DAVIS_CHIP_DAVIS346B 5
```

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

4.1.2.288 DAVIS_CHIP_DAVIS346C

```
#define DAVIS_CHIP_DAVIS346C 9
```

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

4.1.2.289 DAVIS_CHIP_DAVIS640

```
#define DAVIS_CHIP_DAVIS640 6
```

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

4.1.2.290 DAVIS_CHIP_DAVIS640H

```
#define DAVIS_CHIP_DAVIS640H 7
```

DAVIS640H chip identifier. 640x480 APS, 320x240 DVS, color possible, internal ADC.

4.1.2.291 DAVIS_CONFIG_APS

```
#define DAVIS_CONFIG_APS 2
```

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

4.1.2.292 DAVIS_CONFIG_APS_AUTOEXPOSURE

```
#define DAVIS_CONFIG_APS_AUTOEXPOSURE 101
```

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

4.1.2.293 DAVIS_CONFIG_APS_COLOR_FILTER

```
#define DAVIS_CONFIG_APS_COLOR_FILTER 3
```

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

4.1.2.294 DAVIS_CONFIG_APS_END_COLUMN_0

```
#define DAVIS_CONFIG_APS_END_COLUMN_0 10
```

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.

4.1.2.295 DAVIS_CONFIG_APS_END_ROW_0

```
#define DAVIS_CONFIG_APS_END_ROW_0 11
```

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.

4.1.2.296 DAVIS_CONFIG_APS_EXPOSURE

```
#define DAVIS_CONFIG_APS_EXPOSURE 12
```

Parameter address for module DAVIS_CONFIG_APS: frame exposure time. Range: 0-4194303, in microseconds (maximum ∼4s). Very precise for Global Shutter, slightly less exact for Rolling Shutter due to column-based timing constraints.

4.1.2.297 DAVIS_CONFIG_APS_FRAME_INTERVAL

```
#define DAVIS_CONFIG_APS_FRAME_INTERVAL 13
```

Parameter address for module DAVIS_CONFIG_APS: time between consecutive frames. Range: 0-8388607, in microseconds (maximum ∼8s). This can be used to set a frame-rate. Please note the frame-rate is best-effort, and may not be met if readout and exposure times exceed this value.

4.1.2.298 DAVIS_CONFIG_APS_FRAME_MODE

```
#define DAVIS_CONFIG_APS_FRAME_MODE 102
```

Parameter address for module DAVIS_CONFIG_APS: select desired type of frame output. Available are: 0 - Default, meaning grayscale on MONO cameras and RGB color on cameras with color filters. 1 - Grayscale, always a grayscale intensity frame. 2 - Original, send the frame exactly as it comes in from the device (will show grid pattern on color cameras).

4.1.2.299 DAVIS_CONFIG_APS_GLOBAL_SHUTTER

```
#define DAVIS_CONFIG_APS_GLOBAL_SHUTTER 7
```

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).

4.1.2.300 DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER

```
#define DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER 6
```

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.

4.1.2.301 DAVIS_CONFIG_APS_ORIENTATION_INFO

```
#define DAVIS_CONFIG_APS_ORIENTATION_INFO 2
```

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

4.1.2.302 DAVIS_CONFIG_APS_RUN

```
#define DAVIS_CONFIG_APS_RUN 4
```

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

4.1.2.303 DAVIS_CONFIG_APS_SIZE_COLUMNS

```
#define DAVIS_CONFIG_APS_SIZE_COLUMNS 0
```

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

4.1.2.304 DAVIS_CONFIG_APS_SIZE_ROWS

```
#define DAVIS_CONFIG_APS_SIZE_ROWS 1
```

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

4.1.2.305 DAVIS_CONFIG_APS_SNAPSHOT

```
#define DAVIS_CONFIG_APS_SNAPSHOT 100
```

Parameter address for module DAVIS_CONFIG_APS: takes a snapshot (one frame), like a photo-camera. More efficient implementation that just toggling the DAVIS_CONFIG_APS_RUN parameter. The APS module should not be running prior to calling this, as it only makes sense if frames are not being generated at the time. Also, DAVIS←
_CONFIG_APS_FRAME_INTERVAL 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.

4.1.2.306 DAVIS_CONFIG_APS_START_COLUMN_0

```
#define DAVIS_CONFIG_APS_START_COLUMN_0 8
```

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

4.1.2.307 DAVIS_CONFIG_APS_START_ROW_0

```
#define DAVIS_CONFIG_APS_START_ROW_0 9
```

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.

4.1.2.308 DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL 5
```

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.

4.1.2.309 DAVIS_CONFIG_BIAS

```
#define DAVIS_CONFIG_BIAS 5
```

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

4.1.2.310 DAVIS_CONFIG_CHIP

```
#define DAVIS_CONFIG_CHIP 5
```

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

4.1.2.311 DAVIS_CONFIG_DDRAER

```
#define DAVIS_CONFIG_DDRAER 9
```

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

4.1.2.312 DAVIS_CONFIG_DDRAER_RUN

```
#define DAVIS_CONFIG_DDRAER_RUN 0
```

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

4.1.2.313 DAVIS_CONFIG_DVS

```
#define DAVIS_CONFIG_DVS 1
```

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

4.1.2.314 DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL

```
#define DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL 5
```

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.

4.1.2.315 DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY

```
#define DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY 31
```

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.

4.1.2.316 DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_TIME

```
#define DAVIS_CONFIG_DVS_FILTER_BACKGROUND_ACTIVITY_TIME 32
```

Parameter address for module DAVIS_CONFIG_DVS: specify the time difference constant for the background-activity filter. Range: 0 - 4095, in 250µs units. Events that are correlated within this time-frame are let through, while others are filtered out.

4.1.2.317 DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN 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, X axis setting.

4.1.2.318 DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW 11
```

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.

4.1.2.319 DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN 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, X axis setting.

4.1.2.320 DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW 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 1, Y axis setting.

4.1.2.321 DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN 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, X axis setting.

4.1.2.322 DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW 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 2, Y axis setting.

4.1.2.323 DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN 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, X axis setting.

4.1.2.324 DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW 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 3, Y axis setting.

4.1.2.325 DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN 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, X axis setting.

4.1.2.326 DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW 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 4, Y axis setting.

4.1.2.327 DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN 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, X axis setting.

4.1.2.328 DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW 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 5, Y axis setting.

4.1.2.329 DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN 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, X axis setting.

4.1.2.330 DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW 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 6, Y axis setting.

4.1.2.331 DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN 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, X axis setting.

4.1.2.332 DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW 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 7, Y axis setting.

4.1.2.333 DAVIS_CONFIG_DVS_FILTER_PIXEL_AUTO_TRAIN

```
#define DAVIS_CONFIG_DVS_FILTER_PIXEL_AUTO_TRAIN 100
```

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

4.1.2.334 DAVIS_CONFIG_DVS_FILTER_POLARITY_FLATTEN

```
#define DAVIS_CONFIG_DVS_FILTER_POLARITY_FLATTEN 61
```

Parameter address for module DAVIS_CONFIG_DVS: flatten all polarities to OFF (0).

4.1.2.335 DAVIS_CONFIG_DVS_FILTER_POLARITY_SUPPRESS

```
#define DAVIS_CONFIG_DVS_FILTER_POLARITY_SUPPRESS 62
```

Parameter address for module DAVIS_CONFIG_DVS: suppress one of the two ON/OFF polarities completely. Use DAVIS_CONFIG_DVS_FILTER_POLARITY_IGNORE to select which.

4.1.2.336 DAVIS_CONFIG_DVS_FILTER_POLARITY_SUPPRESS_TYPE

```
#define DAVIS_CONFIG_DVS_FILTER_POLARITY_SUPPRESS_TYPE 63
```

Parameter address for module DAVIS_CONFIG_DVS: polarity to suppress (0=OFF, 1=ON). Use DAVIS_CONFI↩
G_DVS_FILTER_POLARITY_IGNORE to enable.

4.1.2.337 DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD

```
#define DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD 33
```

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

4.1.2.338 DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD_TIME

```
#define DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PERIOD_TIME 34
```

Parameter address for module DAVIS_CONFIG_DVS: specify the time constant for the refractory period filter. Range: 0 - 4095, in 250µs units. Pixels will be inhibited from generating new events during this time after the last even has fired.

4.1.2.339 DAVIS_CONFIG_DVS_FILTER_ROI_END_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_ROI_END_COLUMN 43
```

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

4.1.2.340 DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW 44
```

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

4.1.2.341 DAVIS_CONFIG_DVS_FILTER_ROI_START_COLUMN

```
#define DAVIS_CONFIG_DVS_FILTER_ROI_START_COLUMN 41
```

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

4.1.2.342 DAVIS_CONFIG_DVS_FILTER_ROI_START_ROW

```
#define DAVIS_CONFIG_DVS_FILTER_ROI_START_ROW 42
```

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

4.1.2.343 DAVIS_CONFIG_DVS_FILTER_SKIP_EVENTS

```
#define DAVIS_CONFIG_DVS_FILTER_SKIP_EVENTS 51
```

Parameter address for module DAVIS_CONFIG_DVS: enable the event skip filter, which simply throws away one event every N events (decimation filter).

4.1.2.344 DAVIS_CONFIG_DVS_FILTER_SKIP_EVENTS_EVERY

```
#define DAVIS_CONFIG_DVS_FILTER_SKIP_EVENTS_EVERY 52
```

Parameter address for module DAVIS_CONFIG_DVS: number of events to let through before skipping one. Range: 0 - 255 events.

4.1.2.345 DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER

```
#define DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTIVITY_FILTER 30
```

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.

4.1.2.346 DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER

```
#define DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER 10
```

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.

4.1.2.347 DAVIS_CONFIG_DVS_HAS_POLARITY_FILTER

```
#define DAVIS_CONFIG_DVS_HAS_POLARITY_FILTER 60
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, information about the presence of the polarity suppression 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.

4.1.2.348 DAVIS_CONFIG_DVS_HAS_ROI_FILTER

```
#define DAVIS_CONFIG_DVS_HAS_ROI_FILTER 40
```

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

4.1.2.349 DAVIS_CONFIG_DVS_HAS_SKIP_FILTER

```
#define DAVIS_CONFIG_DVS_HAS_SKIP_FILTER 50
```

Parameter address for module DAVIS_CONFIG_DVS: read-only parameter, information about the presence of the event skip 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.

4.1.2.350 DAVIS_CONFIG_DVS_HAS_STATISTICS

```
#define DAVIS_CONFIG_DVS_HAS_STATISTICS 80
```

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

4.1.2.351 DAVIS_CONFIG_DVS_ORIENTATION_INFO

```
#define DAVIS_CONFIG_DVS_ORIENTATION_INFO 2
```

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

4.1.2.352 DAVIS_CONFIG_DVS_RUN

```
#define DAVIS_CONFIG_DVS_RUN 3
```

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

4.1.2.353 DAVIS_CONFIG_DVS_SIZE_COLUMNS

```
#define DAVIS_CONFIG_DVS_SIZE_COLUMNS 0
```

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

4.1.2.354 DAVIS_CONFIG_DVS_SIZE_ROWS

```
#define DAVIS_CONFIG_DVS_SIZE_ROWS 1
```

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

4.1.2.355 DAVIS_CONFIG_DVS_STATISTICS_EVENTS_COLUMN

```
#define DAVIS_CONFIG_DVS_STATISTICS_EVENTS_COLUMN 83
```

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

4.1.2.356 DAVIS_CONFIG_DVS_STATISTICS_EVENTS_DROPPED

```
#define DAVIS_CONFIG_DVS_STATISTICS_EVENTS_DROPPED 85
```

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

4.1.2.357 DAVIS_CONFIG_DVS_STATISTICS_EVENTS_ROW

```
#define DAVIS_CONFIG_DVS_STATISTICS_EVENTS_ROW 81
```

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

4.1.2.358 DAVIS_CONFIG_DVS_STATISTICS_FILTERED_BACKGROUND_ACTIVITY

```
#define DAVIS_CONFIG_DVS_STATISTICS_FILTERED_BACKGROUND_ACTIVITY 89
```

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

4.1.2.359 DAVIS_CONFIG_DVS_STATISTICS_FILTERED_PIXELS

```
#define DAVIS_CONFIG_DVS_STATISTICS_FILTERED_PIXELS 87
```

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

4.1.2.360 DAVIS_CONFIG_DVS_STATISTICS_FILTERED_REFRACTORY_PERIOD

```
#define DAVIS_CONFIG_DVS_STATISTICS_FILTERED_REFRACTORY_PERIOD 91
```

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

4.1.2.361 DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL 4
```

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

4.1.2.362 DAVIS_CONFIG_EXTINPUT

```
#define DAVIS_CONFIG_EXTINPUT 4
```

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

4.1.2.363 DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_EDGES 2
```

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

4.1.2.364 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LENGTH 5
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the minimal length that a pulse must have to trigger the sending of a special event. Range: 1-1048575, in microseconds.

4.1.2.365 DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_POLARITY 4
```

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

4.1.2.366 DAVIS_CONFIG_EXTINPUT_DETECT_PULSES

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_PULSES 3
```

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

4.1.2.367 DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES

```
#define DAVIS_CONFIG_EXTINPUT_DETECT_RISING_EDGES 1
```

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

4.1.2.368 DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_FALLING_EDGE

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_FALLING_EDGE 16
```

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

4.1.2.369 DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_RISING_EDGE

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_ON_RISING_EDGE 15
```

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

4.1.2.370 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_INTERVAL 13
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the interval between the start of two consecutive pulses. Range: 1-1048575, in microseconds. 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.

4.1.2.371 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_LENGTH 14
```

Parameter address for module DAVIS_CONFIG_EXTINPUT: the length a pulse stays active. Range: 1-1048575, in microseconds. 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_GE↩ NERATE_PULSE_INTERVAL.

4.1.2.372 DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY

```
#define DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_POLARITY 12
```

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

4.1.2.373 DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR

```
#define DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR 10
```

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.

4.1.2.374 DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR

```
#define DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR 0
```

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

4.1.2.375 DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR

```
#define DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR 11
```

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.

4.1.2.376 DAVIS_CONFIG_IMU

```
#define DAVIS_CONFIG_IMU 3
```

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

4.1.2.377 DAVIS_CONFIG_IMU_ACCEL_DLPF

```
#define DAVIS_CONFIG_IMU_ACCEL_DLPF 6
```

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

On InvenSense MPU 6050/6150: 0 - Accel: BW=260Hz, Delay=0ms, FS=1kHz - Gyro: BW=256Hz, Delay=0.98ms, FS=8kHz 1 - Accel: BW=184Hz, Delay=2.0ms, FS=1kHz - Gyro: BW=188Hz, Delay=1.9ms, FS=1kHz 2 - Accel: BW=94Hz, Delay=3.0ms, FS=1kHz - Gyro: BW=98Hz, Delay=2.8ms, FS=1kHz 3 - Accel: BW=44Hz, Delay=4.↩ 9ms, FS=1kHz - Gyro: BW=42Hz, Delay=4.8ms, FS=1kHz 4 - Accel: BW=21Hz, Delay=8.5ms, FS=1kHz - Gyro: BW=20Hz, Delay=8.3ms, FS=1kHz 5 - Accel: BW=10Hz, Delay=13.8ms, FS=1kHz - Gyro: BW=10Hz, Delay=13.↩ 4ms, FS=1kHz 6 - Accel: BW=5Hz, Delay=19.0ms, FS=1kHz - Gyro: BW=5Hz, Delay=18.6ms, FS=1kHz 7 - Accel: RESERVED, FS=1kHz - Gyro: RESERVED, FS=8kHz

On InvenSense MPU 9250: 0 - Accel: BW=218.1Hz, Delay=1.88ms, FS=1kHz 1 - Accel: BW=218.1Hz, Delay=1.↩ 88ms, FS=1kHz 2 - Accel: BW=99Hz, Delay=2.88ms, FS=1kHz 3 - Accel: BW=44.8Hz, Delay=4.88ms, FS=1kHz 4 - Accel: BW=21.2Hz, Delay=8.87ms, FS=1kHz 5 - Accel: BW=10.2Hz, Delay=16.83ms, FS=1kHz 6 - Accel: BW=5.05Hz, Delay=32.48ms, FS=1kHz 7 - Accel: BW=420Hz, Delay=1.38ms, FS=1kHz

4.1.2.378 DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE

```
#define DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE 7
```

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

4.1.2.379 DAVIS_CONFIG_IMU_GYRO_DLPF

```
#define DAVIS_CONFIG_IMU_GYRO_DLPF 9
```

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

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

4.1.2.380 DAVIS_CONFIG_IMU_GYRO_FULL_SCALE

```
#define DAVIS_CONFIG_IMU_GYRO_FULL_SCALE 10
```

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

4.1.2.381 DAVIS_CONFIG_IMU_ORIENTATION_INFO

```
#define DAVIS_CONFIG_IMU_ORIENTATION_INFO 1
```

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

4.1.2.382 DAVIS_CONFIG_IMU_RUN_ACCELEROMETER

```
#define DAVIS_CONFIG_IMU_RUN_ACCELEROMETER 2
```

Parameter address for module DAVIS_CONFIG_IMU: enable the IMU's accelerometer. This takes the IMU chip out of sleep.

4.1.2.383 DAVIS_CONFIG_IMU_RUN_GYROSCOPE

```
#define DAVIS_CONFIG_IMU_RUN_GYROSCOPE 3
```

Parameter address for module DAVIS_CONFIG_IMU: enable the IMU's gyroscope. This takes the IMU chip out of sleep.

4.1.2.384 DAVIS_CONFIG_IMU_RUN_TEMPERATURE

```
#define DAVIS_CONFIG_IMU_RUN_TEMPERATURE 4
```

Parameter address for module DAVIS_CONFIG_IMU: enable the IMU's temperature sensor. This takes the IMU chip out of sleep.

4.1.2.385 DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER

```
#define DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER 5
```

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: Sample Rate = Gyroscope Output Rate / (1 + 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.

4.1.2.386 DAVIS_CONFIG_IMU_TYPE

```
#define DAVIS_CONFIG_IMU_TYPE 0
```

Parameter address for module DAVIS_CONFIG_IMU: read-only parameter, contains information on the type of IMU chip being used in this device: 0 - no IMU present 1 - InvenSense MPU 6050/6150 2 - InvenSense MPU 9250 This is reserved for internal use and should not be used by anything other than libcaer.

4.1.2.387 DAVIS_CONFIG_MUX

```
#define DAVIS_CONFIG_MUX 0
```

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

4.1.2.388 DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFER_STALL 5
```

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.

4.1.2.389 DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL

```
#define DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TRANSFER_STALL 4
```

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.

4.1.2.390 DAVIS_CONFIG_MUX_HAS_STATISTICS

```
#define DAVIS_CONFIG_MUX_HAS_STATISTICS 80
```

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

4.1.2.391 DAVIS_CONFIG_MUX_RUN

```
#define DAVIS_CONFIG_MUX_RUN 0
```

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

4.1.2.392 DAVIS_CONFIG_MUX_RUN_CHIP

```
#define DAVIS_CONFIG_MUX_RUN_CHIP 3
```

Parameter address for module DAVIS_CONFIG_MUX: power up the chip's bias generator, enabling the chip to work.

4.1.2.393 DAVIS_CONFIG_MUX_STATISTICS_DVS_DROPPED

```
#define DAVIS_CONFIG_MUX_STATISTICS_DVS_DROPPED 83
```

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

4.1.2.394 DAVIS_CONFIG_MUX_STATISTICS_EXTINPUT_DROPPED

```
#define DAVIS_CONFIG_MUX_STATISTICS_EXTINPUT_DROPPED 81
```

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

4.1.2.395 DAVIS_CONFIG_MUX_TIMESTAMP_RESET

```
#define DAVIS_CONFIG_MUX_TIMESTAMP_RESET 2
```

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

4.1.2.396 DAVIS_CONFIG_MUX_TIMESTAMP_RUN

```
#define DAVIS_CONFIG_MUX_TIMESTAMP_RUN 1
```

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

4.1.2.397 DAVIS_CONFIG_SYSINFO

```
#define DAVIS_CONFIG_SYSINFO 6
```

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

4.1.2.398 DAVIS_CONFIG_SYSINFO_ADC_CLOCK

```
#define DAVIS_CONFIG_SYSINFO_ADC_CLOCK 4
```

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

4.1.2.399 DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER

```
#define DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER 1
```

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

4.1.2.400 DAVIS_CONFIG_SYSINFO_CLOCK_DEVIATION

```
#define DAVIS_CONFIG_SYSINFO_CLOCK_DEVIATION 6
```

Parameter address for module DAVIS_CONFIG_SYSINFO: read-only parameter, the deviation factor for the clocks. Due to how FX3 generates the clocks, which are then used by FPGA/CPLD, they are not integers but have a fractional part. This is reserved for internal use and should not be used by anything other than libcaer.

4.1.2.401 DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER

```
#define DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER 2
```

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

4.1.2.402 DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK

```
#define DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK 3
```

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

4.1.2.403 DAVIS_CONFIG_SYSINFO_LOGIC_VERSION

```
#define DAVIS_CONFIG_SYSINFO_LOGIC_VERSION 0
```

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

4.1.2.404 DAVIS_CONFIG_SYSINFO_USB_CLOCK

```
#define DAVIS_CONFIG_SYSINFO_USB_CLOCK 5
```

Parameter address for module DAVIS_CONFIG_SYSINFO: read-only parameter, the frequency in MHz at which the FPGA/CPLD logic related to USB data transmission is running. This is reserved for internal use and should not be used by anything other than libcaer.

4.1.2.405 DAVIS_CONFIG_USB

```
#define DAVIS_CONFIG_USB 9
```

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

4.1.2.406 DAVIS_CONFIG_USB_EARLY_PACKET_DELAY

```
#define DAVIS_CONFIG_USB_EARLY_PACKET_DELAY 1
```

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

4.1.2.407 DAVIS_CONFIG_USB_RUN

```
#define DAVIS_CONFIG_USB_RUN 0
```

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

4.1.2.408 IS_DAVIS128

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

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

4.1.2.409 IS_DAVIS208

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

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

4.1.2.410 IS_DAVIS240

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

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

4.1.2.411 IS_DAVIS240A

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

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

4.1.2.412 IS_DAVIS240B

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

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

4.1.2.413 IS_DAVIS240C

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

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

4.1.2.414 IS_DAVIS346

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

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

4.1.2.415 IS_DAVIS346A

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

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

4.1.2.416 IS_DAVIS346B

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

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

4.1.2.417 IS_DAVIS346C

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

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

4.1.2.418 IS_DAVIS640

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

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

4.1.2.419 IS_DAVIS640H

```
#define IS_DAVIS640H(
              chipID ) ((chipID) == DAVIS_CHIP_DAVIS640H)
```

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

4.1.3 Enumeration Type Documentation

4.1.3.1 caer_bias_shiftedsource_operating_mode

```
enum caer_bias_shiftedsource_operating_mode
```

Shifted-source bias operating mode.

Enumerator

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

4.1.3.2 caer_bias_shiftedsource_voltage_level

```
enum caer_bias_shiftedsource_voltage_level
```

Shifted-source bias voltage level.

Enumerator

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

4.1.3.3 caer_davis_aps_frame_modes

```
enum caer_davis_aps_frame_modes
```

List of supported APS frame modes.

4.1.3.4 caer_davis_imu_invensense_accel_scale

```
enum caer_davis_imu_invensense_accel_scale
```

List of accelerometer scale settings for InvenSense IMUs.

4.1.3.5 caer_davis_imu_invensense_gyro_scale

```
enum caer_davis_imu_invensense_gyro_scale
```

List of gyroscope scale settings for InvenSense IMUs.

4.1.3.6 caer_davis_imu_types

```
enum caer_davis_imu_types
```

List of supported IMU models.

4.1.4 Function Documentation

4.1.4.1 caerBiasCoarseFineFromCurrent()

```
struct caer_bias_coarsefine caerBiasCoarseFineFromCurrent (
            uint32_t picoAmps )
```

Transform current value in pico-Ampere to coarse-fine bias structure. Limit is 24.8 micro-Ampere.

Parameters

picoAmps	desired current value in pico-Ampere.

Returns

coarse-fine bias structure.

4.1.4.2 caerBiasCoarseFineGenerate()

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

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

Parameters

coarseFineBias	coarse-fine bias structure.

Returns

internal integer representation for device configuration.

4.1.4.3 caerBiasCoarseFineParse()

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

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

Parameters

coarseFineBias	internal integer representation from device.

Returns

coarse-fine bias structure.

4.1.4.4 caerBiasCoarseFineToCurrent()

```
uint32_t caerBiasCoarseFineToCurrent (
            struct caer_bias_coarsefine coarseFineBias )
```

Transform coarse-fine bias structure into corresponding current value in pico-Ampere.

Parameters

coarseFineBias	coarse-fine bias structure.

Returns

corresponding current value in pico-Ampere.

4.1.4.5 caerBiasShiftedSourceGenerate()

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

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

Parameters

shiftedSourceBias	shifted-source bias structure.

Returns

internal integer representation for device configuration.

4.1.4.6 caerBiasShiftedSourceParse()

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

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

Parameters

shiftedSourceBias	internal integer representation from device.

Returns

shifted-source bias structure.

4.1.4.7 caerBiasVDACGenerate()

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

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

Parameters

vdacBias	VDAC bias structure.

Returns

internal integer representation for device configuration.

4.1.4.8 caerBiasVDACParse()

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

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

Parameters

vdacBias	internal integer representation from device.

Returns

VDAC bias structure.

4.1.4.9 caerDavisInfoGet()

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

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

Parameters

handle	a valid device handle.

Returns

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

4.1.4.10 caerDavisROIConfigure()

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

Configure the APS ROI region in one step. This function guarantees efficiency and atomicity (no partial-sized results possible).

Parameters

handle	a valid device handle.
startX	start corner X coordinate (0, 0 is upper left of frame).
startY	start corner Y coordinate (0, 0 is upper left of frame).
endX	end corner X coordinate (0, 0 is upper left of frame). Must be bigger than startX.
endY	end corner Y coordinate (0, 0 is upper left of frame). Must be bigger than startY.

Returns

true on success, false otherwise.

4.2 devices/device.h File Reference

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

Macros

- #define CAER_SUPPORTED_DEVICES_NUMBER 7
- #define CAER_HOST_CONFIG_DATAEXCHANGE -2
- #define CAER_HOST_CONFIG_PACKETS -3
- #define CAER_HOST_CONFIG_LOG -4
- #define CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE 0
- #define CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING 1
- #define CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS 2
- #define CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS 3
- #define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_PACKET_SIZE 0
- #define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL 1
- #define CAER_HOST_CONFIG_LOG_LEVEL 0

Typedefs

- typedef struct caer_device_handle ∗ caerDeviceHandle

Functions

- bool caerDeviceClose (caerDeviceHandle ∗handle)
- bool caerDeviceSendDefaultConfig (caerDeviceHandle handle)
- bool caerDeviceConfigSet (caerDeviceHandle handle, int8_t modAddr, uint8_t paramAddr, uint32_t param)
- bool caerDeviceConfigGet (caerDeviceHandle handle, int8_t modAddr, uint8_t paramAddr, uint32_t ∗param)
- bool caerDeviceConfigGet64 (caerDeviceHandle handle, int8_t modAddr, uint8_t paramAddr, uint64_↩ t ∗param)
- bool caerDeviceDataStart (caerDeviceHandle handle, void(∗dataNotifyIncrease)(void ∗ptr), void(∗data↩ NotifyDecrease)(void ∗ptr), void ∗dataNotifyUserPtr, void(∗dataShutdownNotify)(void ∗ptr), void ∗data↩ ShutdownUserPtr)
- bool caerDeviceDataStop (caerDeviceHandle handle)
- caerEventPacketContainer caerDeviceDataGet (caerDeviceHandle handle)

4.2.1 Detailed Description

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

4.2.2 Macro Definition Documentation

4.2.2.1 CAER_HOST_CONFIG_DATAEXCHANGE

```
#define CAER_HOST_CONFIG_DATAEXCHANGE -2
```

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

4.2.2.2 CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING

```
#define CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKING 1
```

Parameter address for module CAER_HOST_CONFIG_DATAEXCHANGE: when calling caerDeviceDataGet(), the function can either be blocking, meaning it waits until it has a valid EventPacketContainer to return, or not, meaning it returns right away. This behavior can be set with this flag. Please see the caerDeviceDataGet() documentation for more information on its return values.

4.2.2.3 CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE

```
#define CAER_HOST_CONFIG_DATAEXCHANGE_BUFFER_SIZE 0
```

Parameter address for module CAER_HOST_CONFIG_DATAEXCHANGE: set size of elements that can be held by the thread-safe FIFO buffer between the data transfer thread and the main thread. The default values are usually fine, only change them if you're running into lots of dropped/missing packets; you can turn on the INFO log level to see when this is the case.

4.2.2.4 CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS

```
#define CAER_HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS 2
```

Parameter address for module CAER_HOST_CONFIG_DATAEXCHANGE: whether to start all the data producer modules on the device (DVS, APS, Mux, ...) automatically when starting the data transfer thread with caerDeviceDataStart() or not. If disabled, be aware you will have to start the right modules manually, which can be useful if you need precise control over which ones are running at any time.

4.2.2.5 CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS

```
#define CAER_HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS 3
```

Parameter address for module CAER_HOST_CONFIG_DATAEXCHANGE: whether to stop all the data producer modules on the device (DVS, APS, Mux, ...) automatically when stopping the data transfer thread with caerDeviceDataStop() or not. If disabled, be aware you will have to stop the right modules manually, to halt the data flow, which can be useful if you need precise control over which ones are running at any time.

4.2.2.6 CAER_HOST_CONFIG_LOG

```
#define CAER_HOST_CONFIG_LOG -4
```

Module address: host-side logging configuration.

4.2.2.7 CAER_HOST_CONFIG_LOG_LEVEL

```
#define CAER_HOST_CONFIG_LOG_LEVEL 0
```

Parameter address for module CAER_HOST_CONFIG_LOG: set the log-level for this device, to be used when logging messages. Defaults to the value of the global log-level when the device was first opened.

4.2.2.8 CAER_HOST_CONFIG_PACKETS

```
#define CAER_HOST_CONFIG_PACKETS -3
```

Module address: host-side event packets generation configuration.

4.2.2.9 CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL

```
#define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_INTERVAL 1
```

Parameter address for module CAER_HOST_CONFIG_PACKETS: set the time interval between subsequent packet containers. Must be at least 1 microsecond. The value is in microseconds, and is checked across all types of events contained in the EventPacketContainer.

4.2.2.10 CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_PACKET_SIZE

```
#define CAER_HOST_CONFIG_PACKETS_MAX_CONTAINER_PACKET_SIZE 0
```

Parameter address for module CAER_HOST_CONFIG_PACKETS: set the maximum number of events any of a packet container's packets may hold before it's made available to the user. Set to zero to disable. This is checked for each number of events held in each typed EventPacket that is a part of the EventPacketContainer.

4.2.2.11 CAER_SUPPORTED_DEVICES_NUMBER

```
#define CAER_SUPPORTED_DEVICES_NUMBER 7
```

Number of devices supported by this library. 0 - CAER_DEVICE_DVS128 1 - CAER_DEVICE_DAVIS_FX2 2 - C↩
AER_DEVICE_DAVIS_FX3 3 - CAER_DEVICE_DYNAPSE 4 - CAER_DEVICE_DAVIS 5 - CAER_DEVICE_EDVS
6 - CAER_DEVICE_DAVIS_RPI

4.2.3 Typedef Documentation

4.2.3.1 caerDeviceHandle

```
typedef struct caer_device_handle* caerDeviceHandle
```

Pointer to an open device on which to operate.

4.2.4 Function Documentation

4.2.4.1 caerDeviceClose()

```
bool caerDeviceClose (
            caerDeviceHandle * handle )
```

Close a previously opened device and invalidate its handle.

Parameters

handle	pointer to a valid device handle. Will set handle to NULL if closing is successful, to prevent further usage of this handle for other operations.

Returns

true if closing was successful, false on errors.

4.2.4.2 caerDeviceConfigGet()

```
bool caerDeviceConfigGet (
            caerDeviceHandle handle,
            int8_t modAddr,
            uint8_t paramAddr,
            uint32_t * param )
```

Get the value of a configuration parameter.

Parameters

handle	a valid device handle.
modAddr	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.
paramAddr	a parameter address, to select a specific parameter to query from this particular configuration module. Only positive numbers (including zero) are allowed.
param	a pointer to an integer, in which to store the configuration parameter's current value. The integer will always be either set to zero (on failure), or to the current value (on success).

Returns

true if getting the configuration was successful, false on errors.

4.2.4.3 caerDeviceConfigGet64()

```
bool caerDeviceConfigGet64 (
            caerDeviceHandle handle,
            int8_t modAddr,
            uint8_t paramAddr,
            uint64_t * param )
```

Get the value of a 64bit configuration parameter. This is for special read-only configuration parameters only! Use only when required by the parameter's documentation!

Parameters

handle	a valid device handle.
modAddr	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.
paramAddr	a parameter address, to select a specific parameter to query from this particular configuration module. Only positive numbers (including zero) are allowed.
param	a pointer to a 64bit integer, in which to store the configuration parameter's current value. The integer will always be either set to zero (on failure), or to the current value (on success).

Returns

true if getting the configuration was successful, false on errors.

4.2.4.4 caerDeviceConfigSet()

```
bool caerDeviceConfigSet (
            caerDeviceHandle handle,
            int8_t modAddr,
            uint8_t paramAddr,
            uint32_t param )
```

Set a configuration parameter to a given value.

Parameters

handle	a valid device handle.
modAddr	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.
paramAddr	a parameter address, to select a specific parameter to update from this particular configuration module. Only positive numbers (including zero) are allowed.
param	a configuration parameter's new value.

Returns

true if sending the configuration was successful, false on errors.

4.2.4.5 caerDeviceDataGet()

```
caerEventPacketContainer caerDeviceDataGet (
            caerDeviceHandle handle )
```

Get an event packet container, which contains events of various types generated by the device, for further processing. The returned data structures are allocated in memory and will need to be freed. The

caerEventPacketContainerFree() function can be used to correctly free the full container memory. For single caerEventPackets, just use free(). This function can be made blocking with the CAER_HOST_CONFIG_DATAE↩
XCHANGE_BLOCKING configuration parameter. By default it is non-blocking.

Parameters

handle	a valid device handle.

Returns

a valid event packet container. NULL will be returned on errors, such as exceptional device shutdown, or when there is no container available in non-blocking mode. Always check this return value!

4.2.4.6 caerDeviceDataStart()

```
bool caerDeviceDataStart (
            caerDeviceHandle handle,
            void(*)(void *ptr) dataNotifyIncrease,
            void(*)(void *ptr) dataNotifyDecrease,
            void * dataNotifyUserPtr,
            void(*)(void *ptr) dataShutdownNotify,
            void * dataShutdownUserPtr )
```

Start getting data from the device, setting up the data transfers and starting the data producers (see CAER↩ _HOST_CONFIG_DATAEXCHANGE_START_PRODUCERS). Supports notification of new data and exceptional shutdown events via user-defined call-backs.

Parameters

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

Returns

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

4.2.4.7 caerDeviceDataStop()

```
bool caerDeviceDataStop (
            caerDeviceHandle handle )
```

Stop getting data from the device, shutting down the data transfers and stopping the data producers (see CAER_↩ HOST_CONFIG_DATAEXCHANGE_STOP_PRODUCERS). This normal shut-down will not generate a notification (see caerDeviceDataStart()).

Parameters

handle	a valid device handle.

Returns

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

4.2.4.8 caerDeviceSendDefaultConfig()

```
bool caerDeviceSendDefaultConfig (
            caerDeviceHandle handle )
```

Send a set of good default configuration settings to the device. This avoids users having to set every configuration option each time, especially when wanting to get going quickly or just needing to change a few settings to get to the desired operating mode.

Parameters

handle	a valid device handle.

Returns

true if sending the configuration was successful, false on errors.

4.3 devices/device_discover.h File Reference

```
#include "davis.h"
#include "dvs128.h"
#include "dynapse.h"
#include "edvs.h"
```

Data Structures

- struct caer_device_discovery_result

Macros

- #define CAER_DEVICE_DISCOVER_ALL -1

Typedefs

- typedef struct caer_device_discovery_result ∗ caerDeviceDiscoveryResult

Functions

- ssize_t caerDeviceDiscover (int16_t deviceType, caerDeviceDiscoveryResult ∗discoveredDevices)
- caerDeviceHandle caerDeviceDiscoverOpen (uint16_t deviceID, caerDeviceDiscoveryResult discovered←
Device)

4.3.1 Detailed Description

Functions to discover supported devices attached to the current host system, and then open them.

4.3.2 Macro Definition Documentation

4.3.2.1 CAER_DEVICE_DISCOVER_ALL

```
#define CAER_DEVICE_DISCOVER_ALL -1
```

Define for special value to discover all device types.

4.3.3 Typedef Documentation

4.3.3.1 caerDeviceDiscoveryResult

```
typedef struct caer_device_discovery_result* caerDeviceDiscoveryResult
```

Pointer to result of a device discovery operation.

4.3.4 Function Documentation

4.3.4.1 caerDeviceDiscover()

```
ssize_t caerDeviceDiscover (
            int16_t deviceType,
            caerDeviceDiscoveryResult * discoveredDevices )
```

Discover all supported devices that are accessible on this system. Use -1 as 'deviceType' to search for any device, or an actual device type ID to only search for matches of that specific type.

Parameters

deviceType	type of device to search for, use -1 for any.
discoveredDevices	pointer to array of results, memory will be allocated for it automatically. On error, the pointer is set to NULL. Remember to free() the memory once done!

Returns

number of discovered devices, 0 if no device could be found; or -1 if an error occurred.

4.3.4.2 caerDeviceDiscoverOpen()

```
caerDeviceHandle caerDeviceDiscoverOpen (
            uint16_t deviceID,
            caerDeviceDiscoveryResult discoveredDevice )
```

Open a specific device based on information returned by caerDeviceDiscover(), then assign an ID to it and return a handle for further usage.

Parameters

deviceID	a unique ID to identify the device from others. Will be used as the source for EventPackets being generated from its data.
discoveredDevice	pointer to the result of a device discovery operation. Uniquely identifies a particular device.

Returns

a valid device handle that can be used with the other libcaer functions, or NULL on error. Always check for this!

4.4 devices/dvs128.h File Reference

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

Data Structures

- struct caer_dvs128_info

Macros

- #define CAER_DEVICE_DVS128 0
- #define DVS128_CONFIG_DVS 0
- #define DVS128_CONFIG_BIAS 1
- #define DVS128_CONFIG_DVS_RUN 0
- #define DVS128_CONFIG_DVS_TIMESTAMP_RESET 1
- #define DVS128_CONFIG_DVS_ARRAY_RESET 2
- #define DVS128_CONFIG_DVS_TS_MASTER 3
- #define DVS128_CONFIG_BIAS_CAS 0
- #define DVS128_CONFIG_BIAS_INJGND 1
- #define DVS128_CONFIG_BIAS_REQPD 2
- #define DVS128_CONFIG_BIAS_PUX 3
- #define DVS128_CONFIG_BIAS_DIFFOFF 4
- #define DVS128_CONFIG_BIAS_REQ 5
- #define DVS128_CONFIG_BIAS_REFR 6
- #define DVS128_CONFIG_BIAS_PUY 7
- #define DVS128_CONFIG_BIAS_DIFFON 8
- #define DVS128_CONFIG_BIAS_DIFF 9
- #define DVS128_CONFIG_BIAS_FOLL 10
- #define DVS128_CONFIG_BIAS_PR 11

Functions

- struct caer_dvs128_info caerDVS128InfoGet (caerDeviceHandle handle)

4.4.1 Detailed Description

DVS128 specific configuration defines and information structures.

4.4.2 Macro Definition Documentation

4.4.2.1 CAER_DEVICE_DVS128

```
#define CAER_DEVICE_DVS128 0
```

Device type definition for iniVation DVS128.

4.4.2.2 DVS128_CONFIG_BIAS

```
#define DVS128_CONFIG_BIAS 1
```

Module address: device-side chip bias generator configuration.

4.4.2.3 DVS128_CONFIG_BIAS_CAS

```
#define DVS128_CONFIG_BIAS_CAS 0
```

Parameter address for module DVS128_CONFIG_BIAS: First stage amplifier cascode bias. See 'https↩ ://inivation.com/support/hardware/biasing/' for more details.

4.4.2.4 DVS128_CONFIG_BIAS_DIFF

```
#define DVS128_CONFIG_BIAS_DIFF 9
```

Parameter address for module DVS128_CONFIG_BIAS: Differential (second stage amplifier) bias. See 'https↩ ://inivation.com/support/hardware/biasing/' for more details.

4.4.2.5 DVS128_CONFIG_BIAS_DIFFOFF

```
#define DVS128_CONFIG_BIAS_DIFFOFF 4
```

Parameter address for module DVS128_CONFIG_BIAS: Off events threshold bias. See 'https↩ ://inivation.com/support/hardware/biasing/' for more details.

4.4.2.6 DVS128_CONFIG_BIAS_DIFFON

```
#define DVS128_CONFIG_BIAS_DIFFON 8
```

Parameter address for module DVS128_CONFIG_BIAS: On events threshold bias. See 'https↩ ://inivation.com/support/hardware/biasing/' for more details.

4.4.2.7 DVS128_CONFIG_BIAS_FOLL

```
#define DVS128_CONFIG_BIAS_FOLL 10
```

Parameter address for module DVS128_CONFIG_BIAS: Source follower bias. See 'https://inivation.↩ com/support/hardware/biasing/' for more details.

4.4.2.8 DVS128_CONFIG_BIAS_INJGND

```
#define DVS128_CONFIG_BIAS_INJGND 1
```

Parameter address for module DVS128_CONFIG_BIAS: Injected ground bias. See 'https://inivation.↩ com/support/hardware/biasing/' for more details.

4.4.2.9 DVS128_CONFIG_BIAS_PR

```
#define DVS128_CONFIG_BIAS_PR 11
```

Parameter address for module DVS128_CONFIG_BIAS: Photoreceptor bias. See 'https://inivation.↩ com/support/hardware/biasing/' for more details.

4.4.2.10 DVS128_CONFIG_BIAS_PUX

```
#define DVS128_CONFIG_BIAS_PUX 3
```

Parameter address for module DVS128_CONFIG_BIAS: Pull up on request from X arbiter (AER). See 'https://inivation.com/support/hardware/biasing/' for more details.

4.4.2.11 DVS128_CONFIG_BIAS_PUY

```
#define DVS128_CONFIG_BIAS_PUY 7
```

Parameter address for module DVS128_CONFIG_BIAS: Pull up on request from Y arbiter (AER). See 'https://inivation.com/support/hardware/biasing/' for more details.

4.4.2.12 DVS128_CONFIG_BIAS_REFR

```
#define DVS128_CONFIG_BIAS_REFR 6
```

Parameter address for module DVS128_CONFIG_BIAS: Refractory period bias. See 'https://inivation.com/support/hardware/biasing/' for more details.

4.4.2.13 DVS128_CONFIG_BIAS_REQ

```
#define DVS128_CONFIG_BIAS_REQ 5
```

Parameter address for module DVS128_CONFIG_BIAS: Pull down for passive load inverters in digital AER pixel circuitry. See 'https://inivation.com/support/hardware/biasing/' for more details.

4.4.2.14 DVS128_CONFIG_BIAS_REQPD

```
#define DVS128_CONFIG_BIAS_REQPD 2
```

Parameter address for module DVS128_CONFIG_BIAS: Pull down on chip request (AER). See 'https://inivation.com/support/hardware/biasing/' for more details.

4.4.2.15 DVS128_CONFIG_DVS

```
#define DVS128_CONFIG_DVS 0
```

Module address: device-side DVS configuration.

4.4.2.16 DVS128_CONFIG_DVS_ARRAY_RESET

```
#define DVS128_CONFIG_DVS_ARRAY_RESET 2
```

Parameter address for module DVS128_CONFIG_DVS: reset the whole DVS pixel array. This is a temporary configuration switch and will reset itself right away.

4.4.2.17 DVS128_CONFIG_DVS_RUN

```
#define DVS128_CONFIG_DVS_RUN 0
```

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

4.4.2.18 DVS128_CONFIG_DVS_TIMESTAMP_RESET

```
#define DVS128_CONFIG_DVS_TIMESTAMP_RESET 1
```

Parameter address for module DVS128_CONFIG_DVS: reset the time-stamp counter of the device. This is a temporary configuration switch and will reset itself right away.

4.4.2.19 DVS128_CONFIG_DVS_TS_MASTER

```
#define DVS128_CONFIG_DVS_TS_MASTER 3
```

Parameter address for module DVS128_CONFIG_DVS: control if this DVS is a timestamp master device. Default is enabled.

4.4.3 Function Documentation

4.4.3.1 caerDVS128InfoGet()

```
struct caer_dvs128_info caerDVS128InfoGet (
             caerDeviceHandle handle )
```

Return basic information on the device, such as its ID, its resolution, the logic version, and so on. See the 'struct caer_dvs128_info' documentation for more details.

Parameters

handle	a valid device handle.

Returns

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

4.5 devices/dynapse.h File Reference

```
#include "../events/special.h"
#include "../events/spike.h"
#include "usb.h"
```

Data Structures

- struct caer_dynapse_info
- struct caer_bias_dynapse

Macros

- #define CAER_DEVICE_DYNAPSE 3
- #define DYNAPSE_CHIP_DYNAPSE 64
- #define DYNAPSE_CONFIG_MUX 0
- #define DYNAPSE_CONFIG_AER 1
- #define DYNAPSE_CONFIG_CHIP 5
- #define DYNAPSE_CONFIG_SYSINFO 6
- #define DYNAPSE_CONFIG_USB 9
- #define DYNAPSE_CONFIG_CLEAR_CAM 10
- #define DYNAPSE_CONFIG_DEFAULT_SRAM 11
- #define DYNAPSE_CONFIG_MONITOR_NEU 12
- #define DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY 13
- #define DYNAPSE_CONFIG_SRAM 14
- #define DYNAPSE_CONFIG_SYNAPSERECONFIG 15
- #define DYNAPSE_CONFIG_SPIKEGEN 16
- #define DYNAPSE_CONFIG_TAU2_SET 17
- #define DYNAPSE_CONFIG_POISSONSPIKEGEN 18
- #define DYNAPSE_CONFIG_TAU1_RESET 19
- #define DYNAPSE_CONFIG_TAU2_RESET 20
- #define DYNAPSE_CONFIG_POISSONSPIKEGEN_RUN 0
- #define DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEADDRESS 1
- #define DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEDATA 2
- #define DYNAPSE_CONFIG_POISSONSPIKEGEN_CHIPID 3
- #define DYNAPSE_CONFIG_SPIKEGEN_RUN 0
- #define DYNAPSE_CONFIG_SPIKEGEN_VARMODE 1
- #define DYNAPSE_CONFIG_SPIKEGEN_BASEADDR 2
- #define DYNAPSE_CONFIG_SPIKEGEN_STIMCOUNT 3
- #define DYNAPSE_CONFIG_SPIKEGEN_ISI 4
- #define DYNAPSE_CONFIG_SPIKEGEN_ISIBASE 5
- #define DYNAPSE_CONFIG_SPIKEGEN_REPEAT 6
- #define DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN 0
- #define DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOBALKERNEL 1
- #define DYNAPSE_CONFIG_SYNAPSERECONFIG_USESRAMKERNELS 2
- #define DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPSELECT 3
- #define DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAMBASEADDR 4
- #define DYNAPSE_CONFIG_SRAM_ADDRESS 1
- #define DYNAPSE_CONFIG_SRAM_READDATA 2
- #define DYNAPSE_CONFIG_SRAM_WRITEDATA 3
- #define DYNAPSE_CONFIG_SRAM_RWCOMMAND 4
- #define DYNAPSE_CONFIG_SRAM_READ 0
- #define DYNAPSE_CONFIG_SRAM_WRITE 1
- #define DYNAPSE_CONFIG_SRAM_BURSTMODE 5
- #define DYNAPSE_CONFIG_MUX_RUN 0
- #define DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN 1
- #define DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET 2
- #define DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE 3
- #define DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRANSFER_STALL 4
- #define DYNAPSE_CONFIG_MUX_HAS_STATISTICS 10

- #define DYNAPSE_CONFIG_MUX_STATISTICS_AER_DROPPED 11
- #define DYNAPSE_CONFIG_AER_RUN 3
- #define DYNAPSE_CONFIG_AER_ACK_DELAY 4
- #define DYNAPSE_CONFIG_AER_ACK_EXTENSION 6
- #define DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_STALL 8
- #define DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONTROL 10
- #define DYNAPSE_CONFIG_AER_HAS_STATISTICS 40
- #define DYNAPSE_CONFIG_AER_STATISTICS_EVENTS 41
- #define DYNAPSE_CONFIG_AER_STATISTICS_EVENTS_DROPPED 45
- #define DYNAPSE_CONFIG_CHIP_RUN 0
- #define DYNAPSE_CONFIG_CHIP_ID 1
- #define DYNAPSE_CONFIG_CHIP_CONTENT 2
- #define DYNAPSE_CONFIG_CHIP_REQ_DELAY 3
- #define DYNAPSE_CONFIG_CHIP_REQ_EXTENSION 4
- #define DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION 0
- #define DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER 1
- #define DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER 2
- #define DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK 3
- #define DYNAPSE_CONFIG_USB_RUN 0
- #define DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY 1
- #define DYNAPSE_CONFIG_SRAM_DIRECTION_POS 0
- #define **DYNAPSE_CONFIG_SRAM_DIRECTION_NEG** 1
- #define **DYNAPSE_CONFIG_SRAM_DIRECTION_Y_NORTH** 0
- #define **DYNAPSE_CONFIG_SRAM_DIRECTION_Y_SOUTH** 1
- #define **DYNAPSE_CONFIG_SRAM_DIRECTION_X_EAST** 0
- #define **DYNAPSE_CONFIG_SRAM_DIRECTION_X_WEST** 1
- #define DYNAPSE_X4BOARD_NUMCHIPS 4
- #define DYNAPSE_X4BOARD_NEUX 64
- #define DYNAPSE_X4BOARD_NEUY 64
- #define DYNAPSE_X4BOARD_COREX 4
- #define DYNAPSE_X4BOARD_COREY 4
- #define DYNAPSE_CONFIG_DYNAPSE_U0 0

  Chip 0 ID.
- #define DYNAPSE_CONFIG_DYNAPSE_U1 1

  Chip 1 ID.
- #define DYNAPSE_CONFIG_DYNAPSE_U2 2

  Chip 2 ID.
- #define DYNAPSE_CONFIG_DYNAPSE_U3 3

  Chip 3 ID.
- #define DYNAPSE_CONFIG_NUMCORES 4

  Number of cores per chip.
- #define DYNAPSE_CONFIG_NUMNEURONS 1024

  Number of neurons in single chip.
- #define DYNAPSE_CONFIG_NUMNEURONS_CORE 256

  Number of neurons per core.
- #define DYNAPSE_CONFIG_XCHIPSIZE 32

  Number of columns of neurons in a chip.
- #define DYNAPSE_CONFIG_YCHIPSIZE 32

  Number of rows of neurons in a core.
- #define DYNAPSE_CONFIG_NEUCOL 16

  Number of columns of neurons in a core.
- #define DYNAPSE_CONFIG_NEUROW 16

  Number of rows of neurons in a core.

- #define DYNAPSE_CONFIG_CAMCOL 16

    Number of columns of CAMs in a core.
- #define DYNAPSE_CONFIG_NUMCAM_NEU 64

    Number of CAMs per neuron.
- #define DYNAPSE_CONFIG_NUMSRAM_NEU 4

    Number of SRAM cells per neuron.
- #define DYNAPSE_CONFIG_CAMTYPE_F_EXC 3

    Fast excitatory synapse.
- #define DYNAPSE_CONFIG_CAMTYPE_S_EXC 2

    Slow excitatory synapse.
- #define DYNAPSE_CONFIG_CAMTYPE_F_INH 1

    Fast inhibitory synapse.
- #define DYNAPSE_CONFIG_CAMTYPE_S_INH 0

    Slow inhibitory synapse.
- #define DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P 0
- #define **DYNAPSE_CONFIG_BIAS_C0_PS_WEIGHT_INH_S_N** 2
- #define **DYNAPSE_CONFIG_BIAS_C0_PS_WEIGHT_INH_F_N** 4
- #define **DYNAPSE_CONFIG_BIAS_C0_PS_WEIGHT_EXC_S_N** 6
- #define **DYNAPSE_CONFIG_BIAS_C0_PS_WEIGHT_EXC_F_N** 8
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_RFR_N** 10
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_TAU1_N** 12
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_AHTAU_N** 14
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_CASC_N** 16
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_TAU2_N** 18
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_BUF_P** 20
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_AHTHR_N** 22
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_THR_N** 24
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPIE_THR_S_P** 26
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPIE_THR_F_P** 28
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPII_THR_F_P** 30
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPII_THR_S_P** 32
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_NMDA_N** 34
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_DC_P** 36
- #define **DYNAPSE_CONFIG_BIAS_C0_IF_AHW_P** 38
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPII_TAU_S_P** 40
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPII_TAU_F_P** 42
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPIE_TAU_F_P** 44
- #define **DYNAPSE_CONFIG_BIAS_C0_NPDPIE_TAU_S_P** 46
- #define **DYNAPSE_CONFIG_BIAS_C0_R2R_P** 48
- #define **DYNAPSE_CONFIG_BIAS_C1_PULSE_PWLK_P** 1
- #define **DYNAPSE_CONFIG_BIAS_C1_PS_WEIGHT_INH_S_N** 3
- #define **DYNAPSE_CONFIG_BIAS_C1_PS_WEIGHT_INH_F_N** 5
- #define **DYNAPSE_CONFIG_BIAS_C1_PS_WEIGHT_EXC_S_N** 7
- #define **DYNAPSE_CONFIG_BIAS_C1_PS_WEIGHT_EXC_F_N** 9
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_RFR_N** 11
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_TAU1_N** 13
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_AHTAU_N** 15
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_CASC_N** 17
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_TAU2_N** 19
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_BUF_P** 21
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_AHTHR_N** 23
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_THR_N** 25
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPIE_THR_S_P** 27

- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPIE_THR_F_P** 29
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPII_THR_F_P** 31
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPII_THR_S_P** 33
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_NMDA_N** 35
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_DC_P** 37
- #define **DYNAPSE_CONFIG_BIAS_C1_IF_AHW_P** 39
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPII_TAU_S_P** 41
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPII_TAU_F_P** 43
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPIE_TAU_F_P** 45
- #define **DYNAPSE_CONFIG_BIAS_C1_NPDPIE_TAU_S_P** 47
- #define **DYNAPSE_CONFIG_BIAS_C1_R2R_P** 49
- #define **DYNAPSE_CONFIG_BIAS_U_BUFFER** 50
- #define **DYNAPSE_CONFIG_BIAS_U_SSP** 51
- #define **DYNAPSE_CONFIG_BIAS_U_SSN** 52
- #define **DYNAPSE_CONFIG_BIAS_C2_PULSE_PWLK_P** 64
- #define **DYNAPSE_CONFIG_BIAS_C2_PS_WEIGHT_INH_S_N** 66
- #define **DYNAPSE_CONFIG_BIAS_C2_PS_WEIGHT_INH_F_N** 68
- #define **DYNAPSE_CONFIG_BIAS_C2_PS_WEIGHT_EXC_S_N** 70
- #define **DYNAPSE_CONFIG_BIAS_C2_PS_WEIGHT_EXC_F_N** 72
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_RFR_N** 74
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_TAU1_N** 76
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_AHTAU_N** 78
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_CASC_N** 80
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_TAU2_N** 82
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_BUF_P** 84
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_AHTHR_N** 86
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_THR_N** 88
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPIE_THR_S_P** 90
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPIE_THR_F_P** 92
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPII_THR_F_P** 94
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPII_THR_S_P** 96
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_NMDA_N** 98
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_DC_P** 100
- #define **DYNAPSE_CONFIG_BIAS_C2_IF_AHW_P** 102
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPII_TAU_S_P** 104
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPII_TAU_F_P** 106
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPIE_TAU_F_P** 108
- #define **DYNAPSE_CONFIG_BIAS_C2_NPDPIE_TAU_S_P** 110
- #define **DYNAPSE_CONFIG_BIAS_C2_R2R_P** 112
- #define **DYNAPSE_CONFIG_BIAS_C3_PULSE_PWLK_P** 65
- #define **DYNAPSE_CONFIG_BIAS_C3_PS_WEIGHT_INH_S_N** 67
- #define **DYNAPSE_CONFIG_BIAS_C3_PS_WEIGHT_INH_F_N** 69
- #define **DYNAPSE_CONFIG_BIAS_C3_PS_WEIGHT_EXC_S_N** 71
- #define **DYNAPSE_CONFIG_BIAS_C3_PS_WEIGHT_EXC_F_N** 73
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_RFR_N** 75
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_TAU1_N** 77
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_AHTAU_N** 79
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_CASC_N** 81
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_TAU2_N** 83
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_BUF_P** 85
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_AHTHR_N** 87
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_THR_N** 89
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPIE_THR_S_P** 91
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPIE_THR_F_P** 93
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPII_THR_F_P** 95

- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPII_THR_S_P** 97
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_NMDA_N** 99
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_DC_P** 101
- #define **DYNAPSE_CONFIG_BIAS_C3_IF_AHW_P** 103
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPII_TAU_S_P** 105
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPII_TAU_F_P** 107
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPIE_TAU_F_P** 109
- #define **DYNAPSE_CONFIG_BIAS_C3_NPDPIE_TAU_S_P** 111
- #define **DYNAPSE_CONFIG_BIAS_C3_R2R_P** 113
- #define **DYNAPSE_CONFIG_BIAS_D_BUFFER** 114
- #define **DYNAPSE_CONFIG_BIAS_D_SSP** 115
- #define **DYNAPSE_CONFIG_BIAS_D_SSN** 116

Functions

- struct caer_dynapse_info caerDynapseInfoGet (caerDeviceHandle handle)
- uint32_t caerBiasDynapseGenerate (const struct caer_bias_dynapse dynapseBias)
- struct caer_bias_dynapse caerBiasDynapseParse (const uint32_t dynapseBias)
- bool caerDynapseWriteSramWords (caerDeviceHandle handle, const uint16_t ∗data, uint32_t baseAddr, size_t numWords)
- bool caerDynapseWritePoissonSpikeRate (caerDeviceHandle handle, uint16_t neuronAddr, float rateHz)
- bool caerDynapseWriteSram (caerDeviceHandle handle, uint8_t coreId, uint8_t neuronAddrCore, uint8_↩ t virtualCoreId, bool sx, uint8_t dx, bool sy, uint8_t dy, uint8_t sramId, uint8_t destinationCore)
- bool caerDynapseWriteSramN (caerDeviceHandle handle, uint16_t neuronAddr, uint8_t sramId, uint8_↩ t virtualCoreId, bool sx, uint8_t dx, bool sy, uint8_t dy, uint8_t destinationCore)
- bool caerDynapseWriteCam (caerDeviceHandle handle, uint16_t inputNeuronAddr, uint16_t neuronAddr, uint8_t camId, uint8_t synapseType)
- bool caerDynapseSendDataToUSB (caerDeviceHandle handle, const uint32_t ∗data, size_t numConfig)
- uint32_t caerDynapseGenerateCamBits (uint16_t inputNeuronAddr, uint16_t neuronAddr, uint8_t camId, uint8_t synapseType)
- uint32_t caerDynapseGenerateSramBits (uint16_t neuronAddr, uint8_t sramId, uint8_t virtualCoreId, bool sx, uint8_t dx, bool sy, uint8_t dy, uint8_t destinationCore)
- uint16_t caerDynapseCoreXYToNeuronId (uint8_t coreId, uint8_t columnX, uint8_t rowY)
- uint16_t caerDynapseCoreAddrToNeuronId (uint8_t coreId, uint8_t neuronAddrCore)
- uint16_t caerDynapseSpikeEventGetX (caerSpikeEventConst event)
- uint16_t caerDynapseSpikeEventGetY (caerSpikeEventConst event)
- struct caer_spike_event caerDynapseSpikeEventFromXY (uint16_t x, uint16_t y)

4.5.1 Detailed Description

Dynap-se specific configuration defines and information structures.

4.5.2 Macro Definition Documentation

4.5.2.1 CAER_DEVICE_DYNAPSE

```
#define CAER_DEVICE_DYNAPSE 3
```

Device type definition for aiCTX Dynap-se FX2-based boards.

4.5.2.2 DYNAPSE_CHIP_DYNAPSE

```
#define DYNAPSE_CHIP_DYNAPSE 64
```

Dynap-se chip identifier.

4.5.2.3 DYNAPSE_CONFIG_AER

```
#define DYNAPSE_CONFIG_AER 1
```

Module address: device-side AER configuration (from chip). The AER state machine handshakes with the chip's AER bus and gets the spike events from it. It supports various configurable delays.

4.5.2.4 DYNAPSE_CONFIG_AER_ACK_DELAY

```
#define DYNAPSE_CONFIG_AER_ACK_DELAY 4
```

Parameter address for module DYNAPSE_CONFIG_AER: delay capturing the data and acknowledging it on the AER bus for the events by this many LogicClock cycles.

4.5.2.5 DYNAPSE_CONFIG_AER_ACK_EXTENSION

```
#define DYNAPSE_CONFIG_AER_ACK_EXTENSION 6
```

Parameter address for module DYNAPSE_CONFIG_AER: extend the length of the acknowledge on the AER bus for the events by this many LogicClock cycles.

4.5.2.6 DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONTROL

```
#define DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONTROL 10
```

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

4.5.2.7 DYNAPSE_CONFIG_AER_HAS_STATISTICS

```
#define DYNAPSE_CONFIG_AER_HAS_STATISTICS 40
```

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

4.5.2.8 DYNAPSE_CONFIG_AER_RUN

```
#define DYNAPSE_CONFIG_AER_RUN 3
```

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

4.5.2.9 DYNAPSE_CONFIG_AER_STATISTICS_EVENTS

```
#define DYNAPSE_CONFIG_AER_STATISTICS_EVENTS 41
```

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

4.5.2.10 DYNAPSE_CONFIG_AER_STATISTICS_EVENTS_DROPPED

```
#define DYNAPSE_CONFIG_AER_STATISTICS_EVENTS_DROPPED 45
```

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

4.5.2.11 DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_STALL

```
#define DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_STALL 8
```

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

4.5.2.12 DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P

```
#define DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P 0
```

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

- caerBiasDynapseGenerate() for Dynap-se coarse-fine (current) biases. See '`https://ai-ctx.↩ com/support/'` section 'Neuron’s behaviors and parameters tuning'.

4.5.2.13 DYNAPSE_CONFIG_CHIP

```
#define DYNAPSE_CONFIG_CHIP 5
```

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

4.5.2.14 DYNAPSE_CONFIG_CHIP_CONTENT

```
#define DYNAPSE_CONFIG_CHIP_CONTENT 2
```

Parameter address for module DYNAPSE_CONFIG_CHIP: set the configuration content to send to the chip. Every time this changes, the chip ID is appended and the configuration is sent out to the chip.

4.5.2.15 DYNAPSE_CONFIG_CHIP_ID

```
#define DYNAPSE_CONFIG_CHIP_ID 1
```

Parameter address for module DYNAPSE_CONFIG_CHIP: set the chip ID to which configuration content is being sent.

4.5.2.16 DYNAPSE_CONFIG_CHIP_REQ_DELAY

```
#define DYNAPSE_CONFIG_CHIP_REQ_DELAY 3
```

Parameter address for module DYNAPSE_CONFIG_CHIP: delay doing the request after putting out the data by this many LogicClock cycles.

4.5.2.17 DYNAPSE_CONFIG_CHIP_REQ_EXTENSION

```
#define DYNAPSE_CONFIG_CHIP_REQ_EXTENSION 4
```

Parameter address for module DYNAPSE_CONFIG_CHIP: extend the request after receiving the ACK by this many LogicClock cycles.

4.5.2.18 DYNAPSE_CONFIG_CHIP_RUN

```
#define DYNAPSE_CONFIG_CHIP_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_CHIP: enable the configuration AER state machine to send bias and control configuration to the chip.

4.5.2.19 DYNAPSE_CONFIG_CLEAR_CAM

```
#define DYNAPSE_CONFIG_CLEAR_CAM 10
```

Clear CAM content, on all cores of a chip. No arguments are used. Remember to select the chip you want to configure before this!

4.5.2.20 DYNAPSE_CONFIG_DEFAULT_SRAM

```
#define DYNAPSE_CONFIG_DEFAULT_SRAM 11
```

Clear SRAM content, use one SRAM cell (cell 0, out of the four available) to monitor neurons via USB. 'paramAddr' is the chip ID on which to operate, other arguments are unused. Remember to also select the chip you want to configure before this!

4.5.2.21 DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY

```
#define DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY 13
```

Clear SRAM content, route nothing outside (all four SRAM cells zero). No arguments are used. Remember to select the chip you want to configure before this!

4.5.2.22 DYNAPSE_CONFIG_MONITOR_NEU

```
#define DYNAPSE_CONFIG_MONITOR_NEU 12
```

Setup analog neuron monitoring via SMA connectors. 'paramAddr' takes the core ID to be monitored, 'param' the neuron ID. Remember to select the chip you want to configure before this!

4.5.2.23 DYNAPSE_CONFIG_MUX

```
#define DYNAPSE_CONFIG_MUX 0
```

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

4.5.2.24 DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRANSFER_STALL

```
#define DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRANSFER_STALL 4
```

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

4.5.2.25 DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE

```
#define DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE 3
```

Parameter address for module DYNAPSE_CONFIG_MUX: under normal circumstances, the chip's bias generator is only powered up when either the AER or the configuration state machines are running, to save power. This flag forces the bias generator to be powered up all the time.

4.5.2.26 DYNAPSE_CONFIG_MUX_HAS_STATISTICS

```
#define DYNAPSE_CONFIG_MUX_HAS_STATISTICS 10
```

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

4.5.2.27 DYNAPSE_CONFIG_MUX_RUN

```
#define DYNAPSE_CONFIG_MUX_RUN 0
```

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

4.5.2.28 DYNAPSE_CONFIG_MUX_STATISTICS_AER_DROPPED

```
#define DYNAPSE_CONFIG_MUX_STATISTICS_AER_DROPPED 11
```

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

4.5.2.29 DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET

```
#define DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET 2
```

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

4.5.2.30 DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN

```
#define DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN 1
```

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

4.5.2.31 DYNAPSE_CONFIG_POISSONSPIKEGEN

```
#define DYNAPSE_CONFIG_POISSONSPIKEGEN 18
```

Module address: Device side poisson generator configuration Provides run/stop control of poisson spike generation and rate setting for 1024 sources.

4.5.2.32 DYNAPSE_CONFIG_POISSONSPIKEGEN_CHIPID

```
#define DYNAPSE_CONFIG_POISSONSPIKEGEN_CHIPID 3
```

Parameter address for module DYNAPSE_CONFIG_POISSONSPIKEGEN: Chip ID of the chip that will receive events generated by the poisson spike generator.

4.5.2.33 DYNAPSE_CONFIG_POISSONSPIKEGEN_RUN

```
#define DYNAPSE_CONFIG_POISSONSPIKEGEN_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_POISSONSPIKEGEN: Enables or disables generation of poisson spike trains.

4.5.2.34 DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEADDRESS

```
#define DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEADDRESS 1
```

Parameter address for module DYNAPSE_CONFIG_POISSONSPIKEGEN: Selects the address of a poisson spike train source. Writing to this parameter will apply the rate previously written to the WRITEDATA field.

4.5.2.35 DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEDATA

```
#define DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEDATA 2
```

Parameter address for module DYNAPSE_CONFIG_POISSONSPIKEGEN: Holds data that will be written to the address specified by WRITEADDRESS.

4.5.2.36 DYNAPSE_CONFIG_SPIKEGEN

```
#define DYNAPSE_CONFIG_SPIKEGEN 16
```

Module address: Device side spike generator module configuration. Provides start/stop control of spike train application and selection of fixed/variable inter-spike intervals and their location in memory.

4.5.2.37 DYNAPSE_CONFIG_SPIKEGEN_BASEADDR

```
#define DYNAPSE_CONFIG_SPIKEGEN_BASEADDR 2
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Sets the start address of a spike train in memory.

4.5.2.38 DYNAPSE_CONFIG_SPIKEGEN_ISI

```
#define DYNAPSE_CONFIG_SPIKEGEN_ISI 4
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Sets the inter-spike interval that will be used in fixed ISI mode (VARMODE false).

4.5.2.39 DYNAPSE_CONFIG_SPIKEGEN_ISIBASE

```
#define DYNAPSE_CONFIG_SPIKEGEN_ISIBASE 5
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Sets the time base resolution for inter-spike intervals as the number of FPGA clock cycles.

4.5.2.40 DYNAPSE_CONFIG_SPIKEGEN_REPEAT

```
#define DYNAPSE_CONFIG_SPIKEGEN_REPEAT 6
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Sets repeat mode to true or false.

4.5.2.41 DYNAPSE_CONFIG_SPIKEGEN_RUN

```
#define DYNAPSE_CONFIG_SPIKEGEN_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Instructs the spike generator to start applying the configured spike train when the parameter changes from false to true.

4.5.2.42 DYNAPSE_CONFIG_SPIKEGEN_STIMCOUNT

```
#define DYNAPSE_CONFIG_SPIKEGEN_STIMCOUNT 3
```

Paramter address for module DYNAPSE_CONFIG_SPIKEGEN: Sets the number of events to read from memory for a single application of a spike train.

4.5.2.43 DYNAPSE_CONFIG_SPIKEGEN_VARMODE

```
#define DYNAPSE_CONFIG_SPIKEGEN_VARMODE 1
```

Parameter address for module DYNAPSE_CONFIG_SPIKEGEN: Selects variable inter-spike interval mode (true) or fixed inter-spike interval mode (false).

4.5.2.44 DYNAPSE_CONFIG_SRAM

```
#define DYNAPSE_CONFIG_SRAM 14
```

Module address: Device side SRAM controller configuration. The module holds an address, a word to be written to SRAM, the most recent word read using a read command, and a read/write command. Reads/writes are triggered when the address field is changed. Example: caerDynapseWriteSramWords(devHandle, SRAMData, baseAddr, numWords); Writes numWords words from array SRAMData to the SRAM, starting at baseAddr. This define is for internal use of caerDynapseWriteSramWords(); it can be used on its own, but we recommend using the above function that hides all the internal details of writing to the FPGA SRAM.

4.5.2.45 DYNAPSE_CONFIG_SRAM_ADDRESS

```
#define DYNAPSE_CONFIG_SRAM_ADDRESS 1
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Holds the address that will be used for the next read/write. Writing or reading this field will trigger the command contained in the command register to be executed on the FPGA.

4.5.2.46 DYNAPSE_CONFIG_SRAM_BURSTMODE

```
#define DYNAPSE_CONFIG_SRAM_BURSTMODE 5
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Burst mode enable for fast writing. Disables updates on address change and instead updates on data change, while automatically incrementing the writing address. Two 16-bit words are written per 32-bit word sent to the SPI controller starting with the least significant half word.

4.5.2.47 DYNAPSE_CONFIG_SRAM_DIRECTION_POS

```
#define DYNAPSE_CONFIG_SRAM_DIRECTION_POS 0
```

On-chip SRAM for spike routing.

4.5.2.48 DYNAPSE_CONFIG_SRAM_READ

```
#define DYNAPSE_CONFIG_SRAM_READ 0
```

Command for module DYNAPSE_CONFIG_SRAM: Read command for the RWCOMMAND field. Example: caer↩ DeviceConfigSet(devHandle, DYNAPSE_CONFIG_SRAM, DYNAPSE_CONFIG_SRAM_RWCOMMAND, DYNA↩ PSE_CONFIG_SRAM_READ); Sets the SRAM controller up for doing reads.

4.5.2.49 DYNAPSE_CONFIG_SRAM_READDATA

```
#define DYNAPSE_CONFIG_SRAM_READDATA 2
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Holds the most recently read data from the SRAM. Read-only parameter.

4.5.2.50 DYNAPSE_CONFIG_SRAM_RWCOMMAND

```
#define DYNAPSE_CONFIG_SRAM_RWCOMMAND 4
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Holds the command that will be executed when the address field is written to. Example: caerDeviceConfigSet(devHandle, DYNAPSE_CONFIG_SRAM, DYNAPSE↩ _CONFIG_SRAM_RWCOMMAND, DYNAPSE_CONFIG_SRAM_WRITE); Sets the SRAM controller up for doing writes. DYNAPSE_CONFIG_SRAM_READ and DYNAPSE_CONFIG_SRAM_WRITE are supported.

4.5.2.51 DYNAPSE_CONFIG_SRAM_WRITE

```
#define DYNAPSE_CONFIG_SRAM_WRITE 1
```

Command for module DYNAPSE_CONFIG_SRAM: Write command for the RWCOMMAND field. Example: caer↩ DeviceConfigSet(devHandle, DYNAPSE_CONFIG_SRAM, DYNAPSE_CONFIG_SRAM_RWCOMMAND, DYNA↩ PSE_CONFIG_SRAM_WRITE); Sets the SRAM controller up for doing writes.

4.5.2.52 DYNAPSE_CONFIG_SRAM_WRITEDATA

```
#define DYNAPSE_CONFIG_SRAM_WRITEDATA 3
```

Parameter address for module DYNAPSE_CONFIG_SRAM: Holds the data that will be written on the next write. Example: caerDeviceConfigSet(devHandle, DYNAPSE_CONFIG_SRAM, DYNAPSE_CONFIG_SRAM_WRITE↩ DATA, wData); caerDeviceConfigSet(devHandle, DYNAPSE_CONFIG_SRAM, DYNAPSE_CONFIG_SRAM_R↩ WCOMMAND, DYNAPSE_CONFIG_SRAM_WRITE); caerDeviceConfigSet(devHandle, DYNAPSE_CONFIG_S↩ RAM, DYNAPSE_CONFIG_SRAM_ADDRESS, wAddr); Writes wData to the address specified by wAddr.

4.5.2.53 DYNAPSE_CONFIG_SYNAPSERECONFIG

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG 15
```

Module address: Device side Synapse Reconfiguration module configuration. Provides run control, selection between using a single kernel for all neurons and reading per-neuron kernels from SRAM, programming of the global kernel, as well as target output chip ID selection and SRAM kernel table base address.

4.5.2.54 DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPSELECT

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPSELECT 3
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG: Select which chip outputs should go to.

4.5.2.55 DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOBALKERNEL

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOBALKERNEL 1
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG: Bits 16 down to 12 select the address in the global kernel table and bits 11 down to 0 specify the data. The 12 data bits are split into 4∗3 synaptic weight bits which map onto positive/negative polarity events from 2 DVS pixels.

4.5.2.56 DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN 0
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG: Run control. Starts and stops handshaking with DVS.

4.5.2.57 DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAMBASEADDR

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAMBASEADDR 4
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG: SRAM base address configuration in increments of 32 Kib. Setting this to N will place the SRAM kernel LUT in the range [N∗2^15,((N+1)∗2^15)-1]

4.5.2.58 DYNAPSE_CONFIG_SYNAPSERECONFIG_USESRAMKERNELS

```
#define DYNAPSE_CONFIG_SYNAPSERECONFIG_USESRAMKERNELS 2
```

Parameter address for module DYNAPSE_CONFIG_SYNAPSERECONFIG: Boolean parameter for selecting between using kernels stored in SRAM or the global kernel table. 1 for SRAM, 0 for global kernel table.

4.5.2.59 DYNAPSE_CONFIG_SYSINFO

```
#define DYNAPSE_CONFIG_SYSINFO 6
```

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

4.5.2.60 DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER

```
#define DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER 1
```

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

4.5.2.61 DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER

```
#define DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER 2
```

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

4.5.2.62 DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK

```
#define DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK 3
```

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

4.5.2.63 DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION

```
#define DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION 0
```

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

4.5.2.64 DYNAPSE_CONFIG_TAU1_RESET

```
#define DYNAPSE_CONFIG_TAU1_RESET 19
```

Reset all neurons of a core to use the TAU1 neuron leak bias. 'paramAddr' takes the core ID to be reset, other arguments are unused. Remember to select the chip you want to configure before this!

4.5.2.65 DYNAPSE_CONFIG_TAU2_RESET

```
#define DYNAPSE_CONFIG_TAU2_RESET 20
```

Reset all neurons of a core to use the TAU2 neuron leak bias. 'paramAddr' takes the core ID to be reset, other arguments are unused. Remember to select the chip you want to configure before this!

4.5.2.66 DYNAPSE_CONFIG_TAU2_SET

```
#define DYNAPSE_CONFIG_TAU2_SET 17
```

Set certain neurons of a core to use the TAU2 neuron leak bias. By default neurons use the TAU1 neuron leak bias. You can also use DYNAPSE_CONFIG_TAU1_RESET and DYNAPSE_CONFIG_TAU2_RESET to reset all neurons in a core to the same bias. 'paramAddr' takes the core ID to be set, 'param' the neuron ID. Remember to select the chip you want to configure before this!

4.5.2.67 DYNAPSE_CONFIG_USB

```
#define DYNAPSE_CONFIG_USB 9
```

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

4.5.2.68 DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY

```
#define DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY 1
```

Parameter address for module DYNAPSE_CONFIG_USB: the time delay after which a packet of data is committed to USB, even if it is not full yet (short USB packet). The value is in 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).

4.5.2.69 DYNAPSE_CONFIG_USB_RUN

```
#define DYNAPSE_CONFIG_USB_RUN 0
```

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

4.5.2.70 DYNAPSE_X4BOARD_COREX

```
#define DYNAPSE_X4BOARD_COREX 4
```

Number of cores in the x direction of the board.

4.5.2.71 DYNAPSE_X4BOARD_COREY

```
#define DYNAPSE_X4BOARD_COREY 4
```

Number of cores in the y direction of the board.

4.5.2.72 DYNAPSE_X4BOARD_NEUX

```
#define DYNAPSE_X4BOARD_NEUX 64
```

Number of neurons in the x direction of the board.

4.5.2.73 DYNAPSE_X4BOARD_NEUY

```
#define DYNAPSE_X4BOARD_NEUY 64
```

Number of neurons in the y direction of the board.

4.5.2.74 DYNAPSE_X4BOARD_NUMCHIPS

```
#define DYNAPSE_X4BOARD_NUMCHIPS 4
```

Number of chips on the board.

4.5.3 Function Documentation

4.5.3.1 caerBiasDynapseGenerate()

```
uint32_t caerBiasDynapseGenerate (
            const struct caer_bias_dynapse dynapseBias )
```

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

Parameters

dynapseBias	coarse-fine bias structure.

Returns

internal integer representation for device configuration.

4.5.3.2 caerBiasDynapseParse()

```
struct caer_bias_dynapse caerBiasDynapseParse (
            const uint32_t dynapseBias )
```

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

Parameters

dynapseBias	internal integer representation from device.

Returns

coarse-fine bias structure.

4.5.3.3 caerDynapseCoreAddrToNeuronId()

```
uint16_t caerDynapseCoreAddrToNeuronId (
            uint8_t coreId,
            uint8_t neuronAddrCore )
```

Map core ID and per-core neuron address to the correct chip global neuron address.

Parameters

coreId	the chip's core ID, range [0,3].
neuronAddrCore	the neuron's address within this core, range [0,255].

Returns

chip global neuron address.

4.5.3.4 caerDynapseCoreXYToNeuronId()

```
uint16_t caerDynapseCoreXYToNeuronId (
            uint8_t coreId,
            uint8_t columnX,
            uint8_t rowY )
```

Map core ID and column/row address to the correct chip global neuron address.

Parameters

coreId	the chip's core ID, range [0,3].
columnX	the neuron's column address, range [0,15].
rowY	the neuron's row address, range [0,15].

Returns

chip global neuron address.

4.5.3.5 caerDynapseGenerateCamBits()

```
uint32_t caerDynapseGenerateCamBits (
            uint16_t inputNeuronAddr,
            uint16_t neuronAddr,
            uint8_t camId,
            uint8_t synapseType )
```

Generate bits to write a single CAM, to specify which spikes are allowed as input into a neuron.

Parameters

inputNeuronAddr	the neuron address that should be let in as input to this neuron, range [0,1023] (use caerDynapseCoreXYToNeuronId() for a 2D mapping).
neuronAddr	the neuron to program, range [0,1023] (use caerDynapseCoreXYToNeuronId() for a 2D mapping).
camId	CAM address (synapse), each neuron has 64, range [0,63].
synapseType	one of the four possible synaptic weights: [DYNAPSE_CONFIG_CAMTYPE_F_EXC,DYNAPSE_CONFIG_CAMTYPE_S_EXC,DY↵NAPSE_CONFIG_CAMTYPE_F_INH,DYNAPSE_CONFIG_CAMTYPE_S_INH].

Returns

bits to send to device.

4.5.3.6 caerDynapseGenerateSramBits()

```
uint32_t caerDynapseGenerateSramBits (
            uint16_t neuronAddr,
            uint8_t sramId,
            uint8_t virtualCoreId,
            bool sx,
            uint8_t dx,
            bool sy,
            uint8_t dy,
            uint8_t destinationCore )
```

Generate bits to write one of the 4 SRAMs of a single neuron. Writing the SRAM means writing the destination address of where the spikes will be routed to. This works on the on-chip SRAM!

Parameters

neuronAddr	the neuron to program, range [0,1023] (use caerDynapseCoreXYToNeuronId() for a 2D mapping).
sramId	SRAM address (one of four cells), range [0,3].
virtualCoreId	fake source core ID, set it to this value instead of the actual source core ID, range [0,3].
sx	X direction, can be one of: [DYNAPSE_CONFIG_SRAM_DIRECTION_X_EAST,DYNAPS↵E_CONFIG_SRAM_DIRECTION_X_WEST].
dx	X delta, number of chips to jumps before reaching destination, range is [0,3].
sy	Y direction, can be one of: [DYNAPSE_CONFIG_SRAM_DIRECTION_Y_NORTH,DYNA↵PSE_CONFIG_SRAM_DIRECTION_Y_SOUTH].
dy	Y delta, number of chips to jumps before reaching destination, range is [0,3].
destinationCore	spike destination core, uses one-hot coding for the 4 cores: [C3,C2,C1,C0] -> [0,0,0,0] (0 decimal) no core, [1,1,1,1] (15 decimal) all cores

Returns

bits to send to device.

4.5.3.7 caerDynapseInfoGet()

```
struct caer_dynapse_info caerDynapseInfoGet (
            caerDeviceHandle handle )
```

Return basic information on the device, such as its ID, the logic version, and so on. See the 'struct caer_dynapse_info' documentation for more details.

Parameters

handle	a valid device handle.

Returns

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

4.5.3.8 caerDynapseSendDataToUSB()

```
bool caerDynapseSendDataToUSB (
            caerDeviceHandle handle,
            const uint32_t * data,
            size_t numConfig )
```

Send array of configuration parameters to the device via USB.

Remember to select the chip you want to configure before calling this function!

Parameters

handle	a valid device handle.
data	an array of integers holding configuration data.
numConfig	number of configuration parameters to send.

Returns

true on success, false otherwise.

4.5.3.9 caerDynapseSpikeEventFromXY()

```
struct caer_spike_event caerDynapseSpikeEventFromXY (
            uint16_t x,
            uint16_t y )
```

Get the chip ID, core ID and neuron ID from the X and Y coordinates. This is the reverse transform to↩ : caerDynapseSpikeEventGetX() / caerDynapseSpikeEventGetY(). The return value is a 'struct caer_spike_event' because it already has functions to get/set all the needed values.

Parameters

x	a X coordinate as returned by caerDynapseSpikeEventGetX().
y	a Y coordinate as returned by caerDynapseSpikeEventGetY().

Returns

a SpikeEvent struct holding chip ID, core ID and neuron ID.

4.5.3.10 caerDynapseSpikeEventGetX()

```
uint16_t caerDynapseSpikeEventGetX (
            caerSpikeEventConst event )
```

Get the X (column) address for a spike event, in pixels. The (0, 0) address is in the upper left corner.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

the event X address in pixels.

4.5.3.11 caerDynapseSpikeEventGetY()

```
uint16_t caerDynapseSpikeEventGetY (
            caerSpikeEventConst event )
```

Get the Y (row) address for a spike event, in pixels. The (0, 0) address is in the upper left corner.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

the event Y address in pixels.

4.5.3.12 caerDynapseWriteCam()

```
bool caerDynapseWriteCam (
            caerDeviceHandle handle,
            uint16_t inputNeuronAddr,
            uint16_t neuronAddr,
            uint8_t camId,
            uint8_t synapseType )
```

Write a single CAM, to specify which spikes are allowed as input into a neuron.

Remember to select the chip you want to configure before calling this function!

Parameters

handle	a valid device handle.
inputNeuronAddr	the neuron address that should be let in as input to this neuron, range [0,1023].
neuronAddr	the neuron address whose CAM should be programmed, range [0,1023].
camId	CAM address (synapse), each neuron has 64, range [0,63].
synapseType	one of the four possible synaptic weights: [DYNAPSE_CONFIG_CAMTYPE_F_EXC,DYNAPSE_CONFIG_CAMTYPE_S_EXC,DY↩ NAPSE_CONFIG_CAMTYPE_F_INH,DYNAPSE_CONFIG_CAMTYPE_S_INH].

Returns

true on success, false otherwise.

4.5.3.13 caerDynapseWritePoissonSpikeRate()

```
bool caerDynapseWritePoissonSpikeRate (
            caerDeviceHandle handle,
            uint16_t neuronAddr,
            float rateHz )
```

Specifies the poisson spike generator's spike rate.

Parameters

handle	a valid device handle.
neuronAddr	The target neuron of the poisson spike train, range [0,1023].
rateHz	The rate in Hz of the spike train, this will be quantized to the nearest supported level, range [0,4300].

Returns

true on success, false otherwise.

4.5.3.14 caerDynapseWriteSram()

```
bool caerDynapseWriteSram (
            caerDeviceHandle handle,
            uint8_t coreId,
            uint8_t neuronAddrCore,
            uint8_t virtualCoreId,
            bool sx,
            uint8_t dx,
            bool sy,
            uint8_t dy,
            uint8_t sramId,
            uint8_t destinationCore )
```

THIS FUNCTION IS DEPRECATED. USE caerDynapseWriteSramN() INSTEAD! The new function uses the global neuron ID (range [0,1023]) like all others, instead of the separate core ID/neuron ID syntax. Also the arguments are in the same order as caerDynapseGenerateSramBits(), in particular the 'sramId' comes right after 'neuronId'.

Write one of the 4 SRAMs of a single neuron. Writing the SRAM means writing the destination address of where the spikes will be routed to. This works on the on-chip SRAM!

Remember to select the chip you want to configure before calling this function!

Parameters

handle	a valid device handle.
coreId	the chip's core ID, range [0,3].
neuronAddrCore	the neuron's address within this core, range [0,255].
virtualCoreId	fake source core ID, set it to this value instead of the actual source core ID, range [0,3].
sx	X direction, can be one of: [DYNAPSE_CONFIG_SRAM_DIRECTION_X_EAST,DYNAP↩ SE_CONFIG_SRAM_DIRECTION_X_WEST].
dx	X delta, number of chips to jumps before reaching destination, range is [0,3].
sy	Y direction, can be one of: [DYNAPSE_CONFIG_SRAM_DIRECTION_Y_NORTH,DYNA↩ PSE_CONFIG_SRAM_DIRECTION_Y_SOUTH].
dy	Y delta, number of chips to jumps before reaching destination, range is [0,3].
sramId	SRAM address (one of four cells), range [0,3].
destinationCore	spike destination core, uses one-hot coding for the 4 cores: [C3,C2,C1,C0] -> [0,0,0,0] (0 decimal) no core, [1,1,1,1] (15 decimal) all cores

Returns

true on success, false otherwise.

4.5.3.15 caerDynapseWriteSramN()

```
bool caerDynapseWriteSramN (
            caerDeviceHandle handle,
            uint16_t neuronAddr,
            uint8_t sramId,
            uint8_t virtualCoreId,
            bool sx,
            uint8_t dx,
            bool sy,
            uint8_t dy,
            uint8_t destinationCore )
```

Write one of the 4 SRAMs of a single neuron. Writing the SRAM means writing the destination address of where the spikes will be routed to. This works on the on-chip SRAM!

Remember to select the chip you want to configure before calling this function!

Parameters

handle	a valid device handle.
neuronAddr	the neuron to program, range [0,1023] (use caerDynapseCoreXYToNeuronId() for a 2D mapping).
sramId	SRAM address (one of four cells), range [0,3].
virtualCoreId	fake source core ID, set it to this value instead of the actual source core ID, range [0,3].
sx	X direction, can be one of: [DYNAPSE_CONFIG_SRAM_DIRECTION_X_EAST,DYNAPS←E_CONFIG_SRAM_DIRECTION_X_WEST].
dx	X delta, number of chips to jumps before reaching destination, range is [0,3].
sy	Y direction, can be one of: [DYNAPSE_CONFIG_SRAM_DIRECTION_Y_NORTH,DYNA←PSE_CONFIG_SRAM_DIRECTION_Y_SOUTH].
dy	Y delta, number of chips to jumps before reaching destination, range is [0,3].
destinationCore	spike destination core, uses one-hot coding for the 4 cores: [C3,C2,C1,C0] -> [0,0,0,0] (0 decimal) no core, [1,1,1,1] (15 decimal) all cores

Returns

true on success, false otherwise.

4.5.3.16 caerDynapseWriteSramWords()

```
bool caerDynapseWriteSramWords (
            caerDeviceHandle handle,
            const uint16_t * data,
            uint32_t baseAddr,
            size_t numWords )
```

Transfer 16bit words from memory to device SRAM, with configurable starting address and number of words. This works on the FPGA SRAM!

Parameters

handle	a valid device handle.
data	array from which to read data to send to SRAM.
baseAddr	SRAM start address where to put the data.
numWords	number of 16bit words to transfer.

Returns

true on success, false otherwise.

4.6 devices/edvs.h File Reference

```
#include "../events/polarity.h"
#include "../events/special.h"
#include "serial.h"
```

Data Structures

- struct caer_edvs_info

Macros

- #define CAER_DEVICE_EDVS 5
- #define EDVS_CONFIG_DVS 0
- #define EDVS_CONFIG_BIAS 1
- #define EDVS_CONFIG_DVS_RUN 0
- #define EDVS_CONFIG_DVS_TIMESTAMP_RESET 1
- #define EDVS_CONFIG_BIAS_CAS 0
- #define EDVS_CONFIG_BIAS_INJGND 1
- #define EDVS_CONFIG_BIAS_REQPD 2
- #define EDVS_CONFIG_BIAS_PUX 3
- #define EDVS_CONFIG_BIAS_DIFFOFF 4
- #define EDVS_CONFIG_BIAS_REQ 5
- #define EDVS_CONFIG_BIAS_REFR 6
- #define EDVS_CONFIG_BIAS_PUY 7
- #define EDVS_CONFIG_BIAS_DIFFON 8
- #define EDVS_CONFIG_BIAS_DIFF 9
- #define EDVS_CONFIG_BIAS_FOLL 10
- #define EDVS_CONFIG_BIAS_PR 11

Functions

- struct caer_edvs_info caerEDVSInfoGet (caerDeviceHandle handle)

4.6.1 Detailed Description

EDVS-4337 specific configuration defines and information structures.

4.6.2 Macro Definition Documentation

4.6.2.1 CAER_DEVICE_EDVS

```
#define CAER_DEVICE_EDVS 5
```

Device type definition for iniVation EDVS-4337.

4.6.2.2 EDVS_CONFIG_BIAS

```
#define EDVS_CONFIG_BIAS 1
```

Module address: device-side chip bias generator configuration.

4.6.2.3 EDVS_CONFIG_BIAS_CAS

```
#define EDVS_CONFIG_BIAS_CAS 0
```

Parameter address for module EDVS_CONFIG_BIAS: First stage amplifier cascode bias. See 'https↩
://inivation.com/support/hardware/biasing/' for more details.

4.6.2.4 EDVS_CONFIG_BIAS_DIFF

```
#define EDVS_CONFIG_BIAS_DIFF 9
```

Parameter address for module EDVS_CONFIG_BIAS: Differential (second stage amplifier) bias. See 'https↩
://inivation.com/support/hardware/biasing/' for more details.

4.6.2.5 EDVS_CONFIG_BIAS_DIFFOFF

```
#define EDVS_CONFIG_BIAS_DIFFOFF 4
```

Parameter address for module EDVS_CONFIG_BIAS: Off events threshold bias. See 'https://inivation.↩
com/support/hardware/biasing/' for more details.

4.6.2.6 EDVS_CONFIG_BIAS_DIFFON

```
#define EDVS_CONFIG_BIAS_DIFFON 8
```

Parameter address for module EDVS_CONFIG_BIAS: On events threshold bias. See 'https://inivation.↩
com/support/hardware/biasing/' for more details.

4.6.2.7 EDVS_CONFIG_BIAS_FOLL

```
#define EDVS_CONFIG_BIAS_FOLL 10
```

Parameter address for module EDVS_CONFIG_BIAS: Source follower bias. See 'https://inivation.↩
com/support/hardware/biasing/' for more details.

4.6.2.8 EDVS_CONFIG_BIAS_INJGND

```
#define EDVS_CONFIG_BIAS_INJGND 1
```

Parameter address for module EDVS_CONFIG_BIAS: Injected ground bias. See 'https://inivation.↩
com/support/hardware/biasing/' for more details.

4.6.2.9 EDVS_CONFIG_BIAS_PR

```
#define EDVS_CONFIG_BIAS_PR 11
```

Parameter address for module EDVS_CONFIG_BIAS: Photoreceptor bias. See 'https://inivation.↩
com/support/hardware/biasing/' for more details.

4.6.2.10 EDVS_CONFIG_BIAS_PUX

```
#define EDVS_CONFIG_BIAS_PUX 3
```

Parameter address for module EDVS_CONFIG_BIAS: Pull up on request from X arbiter (AER). See 'https↩
://inivation.com/support/hardware/biasing/' for more details.

4.6.2.11 EDVS_CONFIG_BIAS_PUY

```
#define EDVS_CONFIG_BIAS_PUY 7
```

Parameter address for module EDVS_CONFIG_BIAS: Pull up on request from Y arbiter (AER). See 'https↩
://inivation.com/support/hardware/biasing/' for more details.

4.6.2.12 EDVS_CONFIG_BIAS_REFR

```
#define EDVS_CONFIG_BIAS_REFR 6
```

Parameter address for module EDVS_CONFIG_BIAS: Refractory period bias. See 'https://inivation.↩
com/support/hardware/biasing/' for more details.

4.6.2.13 EDVS_CONFIG_BIAS_REQ

```
#define EDVS_CONFIG_BIAS_REQ 5
```

Parameter address for module EDVS_CONFIG_BIAS: Pull down for passive load inverters in digital AER pixel circuitry. See 'https://inivation.com/support/hardware/biasing/' for more details.

4.6.2.14 EDVS_CONFIG_BIAS_REQPD

```
#define EDVS_CONFIG_BIAS_REQPD 2
```

Parameter address for module EDVS_CONFIG_BIAS: Pull down on chip request (AER). See 'https://inivation.com/support/hardware/biasing/' for more details.

4.6.2.15 EDVS_CONFIG_DVS

```
#define EDVS_CONFIG_DVS 0
```

Module address: device-side DVS configuration.

4.6.2.16 EDVS_CONFIG_DVS_RUN

```
#define EDVS_CONFIG_DVS_RUN 0
```

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

4.6.2.17 EDVS_CONFIG_DVS_TIMESTAMP_RESET

```
#define EDVS_CONFIG_DVS_TIMESTAMP_RESET 1
```

Parameter address for module EDVS_CONFIG_DVS: reset the time-stamp counter of the device. This is a temporary configuration switch and will reset itself right away.

4.6.3 Function Documentation

4.6.3.1 caerEDVSInfoGet()

```
struct caer_edvs_info caerEDVSInfoGet (
            caerDeviceHandle handle )
```

Return basic information on the device, such as its ID, its resolution, the logic version, and so on. See the 'struct caer_edvs_info' documentation for more details.

Parameters

handle	a valid device handle.

Returns

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

4.7 devices/serial.h File Reference

```
#include "device.h"
```

Macros

- #define CAER_HOST_CONFIG_SERIAL -1
- #define CAER_HOST_CONFIG_SERIAL_READ_SIZE 0

- #define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_2M 2000000
- #define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_4M 4000000
- #define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_8M 8000000
- #define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_12M 12000000

Functions

- caerDeviceHandle caerDeviceOpenSerial (uint16_t deviceID, uint16_t deviceType, const char ∗serialPort↩ Name, uint32_t serialBaudRate)

4.7.1 Detailed Description

Common functions to access, configure and exchange data with supported serial port devices. Also contains defines for serial port specific configuration options.

4.7.2 Macro Definition Documentation

4.7.2.1 CAER_HOST_CONFIG_SERIAL

```
#define CAER_HOST_CONFIG_SERIAL -1
```

Module address: host-side serial port configuration.

4.7.2.2 CAER_HOST_CONFIG_SERIAL_BAUD_RATE_12M

```
#define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_12M 12000000
```

Parameter values for module CAER_HOST_CONFIG_SERIAL: possible baud-rates for serial port communication.

4.7.2.3 CAER_HOST_CONFIG_SERIAL_BAUD_RATE_2M

```
#define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_2M 2000000
```

Parameter values for module CAER_HOST_CONFIG_SERIAL: possible baud-rates for serial port communication.

4.7.2.4 CAER_HOST_CONFIG_SERIAL_BAUD_RATE_4M

```
#define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_4M 4000000
```

Parameter values for module CAER_HOST_CONFIG_SERIAL: possible baud-rates for serial port communication.

4.7.2.5 CAER_HOST_CONFIG_SERIAL_BAUD_RATE_8M

```
#define CAER_HOST_CONFIG_SERIAL_BAUD_RATE_8M 8000000
```

Parameter values for module CAER_HOST_CONFIG_SERIAL: possible baud-rates for serial port communication.

4.7.2.6 CAER_HOST_CONFIG_SERIAL_READ_SIZE

```
#define CAER_HOST_CONFIG_SERIAL_READ_SIZE 0
```

Parameter address for module CAER_HOST_CONFIG_SERIAL: read size for serial port communication.

4.7.3 Function Documentation

4.7.3.1 caerDeviceOpenSerial()

```
caerDeviceHandle caerDeviceOpenSerial (
             uint16_t deviceID,
             uint16_t deviceType,
             const char * serialPortName,
             uint32_t serialBaudRate )
```

Open a specified serial port device, assign an ID to it and return a handle for further usage. Various means can be employed to limit the selection of the device.

Parameters

deviceID	a unique ID to identify the device from others. Will be used as the source for EventPackets being generated from its data.
deviceType	type of the device to open. Currently supported are: CAER_DEVICE_EDVS
serialPortName	name of the serial port device to open.
serialBaudRate	baud-rate for serial port communication.

Returns

a valid device handle that can be used with the other libcaer functions, or NULL on error. Always check for this! On error, errno is also set to provide more precise information about the failure cause.

4.8 devices/usb.h File Reference

```
#include "device.h"
```

Macros

- #define CAER_HOST_CONFIG_USB -1
- #define CAER_HOST_CONFIG_USB_BUFFER_NUMBER 0
- #define CAER_HOST_CONFIG_USB_BUFFER_SIZE 1

Functions

- caerDeviceHandle caerDeviceOpen (uint16_t deviceID, uint16_t deviceType, uint8_t busNumberRestrict, uint8_t devAddressRestrict, const char ∗serialNumberRestrict)

4.8.1 Detailed Description

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

4.8.2 Macro Definition Documentation

4.8.2.1 CAER_HOST_CONFIG_USB

```
#define CAER_HOST_CONFIG_USB -1
```

Module address: host-side USB configuration.

4.8.2.2 CAER_HOST_CONFIG_USB_BUFFER_NUMBER

```
#define CAER_HOST_CONFIG_USB_BUFFER_NUMBER 0
```

Parameter address for module CAER_HOST_CONFIG_USB: set number of buffers used by libusb for asynchronous data transfers with the USB device. The default values are usually fine, only change them if you're running into I/O limits.

4.8.2.3 CAER_HOST_CONFIG_USB_BUFFER_SIZE

```
#define CAER_HOST_CONFIG_USB_BUFFER_SIZE 1
```

Parameter address for module CAER_HOST_CONFIG_USB: set size of each buffer used by libusb for asynchronous data transfers with the USB device. The default values are usually fine, only change them if you're running into I/O limits.

4.8.3 Function Documentation

4.8.3.1 caerDeviceOpen()

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

Open a specified USB device, assign an ID to it and return a handle for further usage. Various means can be employed to limit the selection of the device.

Parameters

deviceID	a unique ID to identify the device from others. Will be used as the source for EventPackets being generated from its data.
deviceType	type of the device to open. Currently supported are: CAER_DEVICE_DVS128, CAER_DEVICE_DAVIS, CAER_DEVICE_DYNAPSE
busNumberRestrict	restrict the search for viable devices to only this USB bus number.
devAddressRestrict	restrict the search for viable devices to only this USB device address.
serialNumberRestrict	restrict the search for viable devices to only devices which do possess the given Serial Number in their USB SerialNumber descriptor.

Returns

a valid device handle that can be used with the other libcaer functions, or NULL on error. Always check for this! On error, errno is also set to provide more precise information about the failure cause.

4.9 events/common.h File Reference

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

Macros

- #define **caerLogEHO** caerLog

- #define TS_OVERFLOW_SHIFT 31
- #define CAER_DEFAULT_EVENT_TYPES_COUNT 14
- #define CAER_EVENT_PACKET_HEADER_SIZE 28
- #define CAER_ITERATOR_ALL_START(PACKET_HEADER, EVENT_TYPE)
- #define CAER_ITERATOR_ALL_END }
- #define CAER_ITERATOR_VALID_START(PACKET_HEADER, EVENT_TYPE)
- #define CAER_ITERATOR_VALID_END }

- #define VALID_MARK_SHIFT 0
- #define VALID_MARK_MASK 0x00000001

Typedefs

- typedef struct caer_event_packet_header ∗ caerEventPacketHeader
- typedef const struct caer_event_packet_header ∗ **caerEventPacketHeaderConst**

Enumerations

- enum caer_default_event_types {
  SPECIAL_EVENT = 0, POLARITY_EVENT = 1, FRAME_EVENT = 2, IMU6_EVENT = 3,
  IMU9_EVENT = 4, SAMPLE_EVENT = 5, EAR_EVENT = 6, CONFIG_EVENT = 7,
  POINT1D_EVENT = 8, POINT2D_EVENT = 9, POINT3D_EVENT = 10, POINT4D_EVENT = 11,
  SPIKE_EVENT = 12, MATRIX4x4_EVENT = 13 }

Functions

- PACKED_STRUCT (struct caer_event_packet_header { int16_t eventType;int16_t eventSource;int32↩_t eventSize;int32_t eventTSOffset;int32_t eventTSOverflow;int32_t eventCapacity;int32_t event↩Number;int32_t eventValid;})
- static int16_t caerEventPacketHeaderGetEventType (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventType (caerEventPacketHeader header, int16_t eventType)
- static int16_t caerEventPacketHeaderGetEventSource (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventSource (caerEventPacketHeader header, int16_t eventSource)
- static int32_t caerEventPacketHeaderGetEventSize (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventSize (caerEventPacketHeader header, int32_t eventSize)
- static int32_t caerEventPacketHeaderGetEventTSOffset (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventTSOffset (caerEventPacketHeader header, int32_t eventTS↩Offset)
- static int32_t caerEventPacketHeaderGetEventTSOverflow (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventTSOverflow (caerEventPacketHeader header, int32_t eventTS↩Overflow)
- static int32_t caerEventPacketHeaderGetEventCapacity (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventCapacity (caerEventPacketHeader header, int32_t events↩Capacity)
- static int32_t caerEventPacketHeaderGetEventNumber (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventNumber (caerEventPacketHeader header, int32_t events↩Number)
- static int32_t caerEventPacketHeaderGetEventValid (caerEventPacketHeaderConst header)
- static void caerEventPacketHeaderSetEventValid (caerEventPacketHeader header, int32_t eventsValid)

- static const void ∗ caerGenericEventGetEvent (caerEventPacketHeaderConst headerPtr, int32_t n)
- static int32_t caerGenericEventGetTimestamp (const void ∗eventPtr, caerEventPacketHeaderConst headerPtr)
- static int64_t caerGenericEventGetTimestamp64 (const void ∗eventPtr, caerEventPacketHeaderConst headerPtr)
- static bool caerGenericEventIsValid (const void ∗eventPtr)
- static bool caerGenericEventCopy (void ∗eventPtrDestination, const void ∗eventPtrSource, caerEvent↩ PacketHeaderConst headerPtrDestination, caerEventPacketHeaderConst headerPtrSource)
- static int64_t caerEventPacketGetDataSize (caerEventPacketHeaderConst header)
- static int64_t caerEventPacketGetSize (caerEventPacketHeaderConst header)
- static int64_t caerEventPacketGetDataSizeEvents (caerEventPacketHeaderConst header)
- static int64_t caerEventPacketGetSizeEvents (caerEventPacketHeaderConst header)
- static bool caerEventPacketEquals (caerEventPacketHeaderConst firstPacket, caerEventPacketHeaderConst secondPacket)
- static void caerEventPacketClear (caerEventPacketHeader packet)
- static void caerEventPacketClean (caerEventPacketHeader packet)
- **memset** (((uint8_t ∗) packet)+offset, 0,(size_t)((eventCapacity - eventValid) ∗eventSize))
- **caerEventPacketHeaderSetEventNumber** (packet, eventValid)
- static caerEventPacketHeader caerEventPacketResize (caerEventPacketHeader packet, int32_t newEvent↩ Capacity)
- static caerEventPacketHeader caerEventPacketGrow (caerEventPacketHeader packet, int32_t newEvent↩ Capacity)
- static caerEventPacketHeader caerEventPacketAppend (caerEventPacketHeader packet, caerEventPacketHeader appendPacket)
- static caerEventPacketHeader caerEventPacketCopy (caerEventPacketHeaderConst packet)
- static caerEventPacketHeader caerEventPacketCopyOnlyEvents (caerEventPacketHeaderConst packet)
- static caerEventPacketHeader caerEventPacketCopyOnlyValidEvents (caerEventPacketHeaderConst packet)
- **caerEventPacketHeaderSetEventCapacity** (packetCopy, eventValid)
- **caerEventPacketHeaderSetEventNumber** (packetCopy, eventValid)
- **return** (packetCopy)
- static caerEventPacketHeader caerEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32↩ _t tsOverflow, int16_t eventType, int32_t eventSize, int32_t eventTSOffset)

4.9.1 Detailed Description

Common EventPacket header format definition and handling functions. Every EventPacket, of any type, has as a first member a common header, which describes various properties of the contained events. This allows easy parsing of events. See the 'struct caer_event_packet_header' documentation for more details.

4.9.2 Macro Definition Documentation

4.9.2.1 CAER_DEFAULT_EVENT_TYPES_COUNT

```
#define CAER_DEFAULT_EVENT_TYPES_COUNT 14
```

Number of default event types that are part of libcaer. Corresponds to the count of definitions inside the 'enum caer_default_event_types' enumeration.

4.9.2.2 CAER_EVENT_PACKET_HEADER_SIZE

```
#define CAER_EVENT_PACKET_HEADER_SIZE 28
```

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

4.9.2.3 CAER_ITERATOR_ALL_END

```
#define CAER_ITERATOR_ALL_END }
```

Generic iterator close statement.

4.9.2.4 CAER_ITERATOR_ALL_START

```
#define CAER_ITERATOR_ALL_START(
              PACKET_HEADER,
              EVENT_TYPE )
```

Value:

```
for (int32_t caerIteratorCounter = 0; caerIteratorCounter <
      caerEventPacketHeaderGetEventNumber(PACKET_HEADER); \
         caerIteratorCounter++) {
               \
        EVENT_TYPE caerIteratorElement = (EVENT_TYPE) caerGenericEventGetEvent(
      PACKET_HEADER, caerIteratorCounter);
```

Generic iterator over all events in a packet. Returns the current index in the 'caerIteratorCounter' variable of type 'int32_t' and the current event in the 'caerIteratorElement' variable of type EVENT_TYPE.

PACKET_HEADER: a valid EventPacket header pointer. Cannot be NULL. EVENT_TYPE: the event pointer type for this EventPacket (ie. caerPolarityEvent or caerFrameEvent).

4.9.2.5 CAER_ITERATOR_VALID_END

```
#define CAER_ITERATOR_VALID_END }
```

Generic iterator close statement.

4.9.2.6 CAER_ITERATOR_VALID_START

```
#define CAER_ITERATOR_VALID_START(
              PACKET_HEADER,
              EVENT_TYPE )
```

Value:

```
for (int32_t caerIteratorCounter = 0; caerIteratorCounter <
      caerEventPacketHeaderGetEventNumber(PACKET_HEADER); \
         caerIteratorCounter++) {
               \
        EVENT_TYPE caerIteratorElement = (EVENT_TYPE) caerGenericEventGetEvent(
      PACKET_HEADER, caerIteratorCounter); \
        if (!caerGenericEventIsValid(caerIteratorElement)) {
                                       \
            continue;
               \
        }
```

Generic iterator over only the valid events in a packet. Returns the current index in the 'caerIteratorCounter' variable of type 'int32_t' and the current event in the 'caerIteratorElement' variable of type EVENT_TYPE.

PACKET_HEADER: a valid EventPacket header pointer. Cannot be NULL. EVENT_TYPE: the event pointer type for this EventPacket (ie. caerPolarityEvent or caerFrameEvent).

4.9.2.7 TS_OVERFLOW_SHIFT

```
#define TS_OVERFLOW_SHIFT 31
```

64bit timestamp support: since timestamps wrap around after some time, being only 31 bit (32 bit signed int), another timestamp at the packet level provides another 31 bit (32 bit signed int), to enable the generation of a 62 bit (64 bit signed int) microsecond timestamp which is guaranteed to never wrap around (in the next 146'138 years at least). The TSOverflow needs to be shifted by 31 thus when constructing such a timestamp.

4.9.2.8 VALID_MARK_MASK

```
#define VALID_MARK_MASK 0x00000001
```

Generic validity mark: this bit is used to mark whether an event is still valid or not, and can be used to efficiently filter out events from a packet. The caerXXXEventValidate() and caerXXXEventInvalidate() functions should be used to toggle this! 0 in the 0th bit of the first byte means invalid, 1 means valid. This way zeroing-out an event packet sets all its events to invalid. Care must be taken to put the field containing the validity mark always as the first member of an event.

4.9.2.9 VALID_MARK_SHIFT

```
#define VALID_MARK_SHIFT 0
```

Generic validity mark: this bit is used to mark whether an event is still valid or not, and can be used to efficiently filter out events from a packet. The caerXXXEventValidate() and caerXXXEventInvalidate() functions should be used to toggle this! 0 in the 0th bit of the first byte means invalid, 1 means valid. This way zeroing-out an event packet sets all its events to invalid. Care must be taken to put the field containing the validity mark always as the first member of an event.

4.9.3 Typedef Documentation

4.9.3.1 caerEventPacketHeader

```
typedef struct caer_event_packet_header* caerEventPacketHeader
```

Type for pointer to EventPacket header data structure.

4.9.4 Enumeration Type Documentation

4.9.4.1 caer_default_event_types

```
enum caer_default_event_types
```

List of supported event types. Each event type has its own integer representation. All event types below 100 are reserved for use by libcaer and cAER. DO NOT USE THEM FOR YOUR OWN EVENT TYPES!

Enumerator

SPECIAL_EVENT	Special events.
POLARITY_EVENT	Polarity (change, DVS) events.
FRAME_EVENT	Frame (intensity, APS) events.
IMU6_EVENT	6 axes IMU events.
IMU9_EVENT	9 axes IMU events.
SAMPLE_EVENT	ADC sample events.
EAR_EVENT	Ear (cochlea) events.
CONFIG_EVENT	Device configuration events.
POINT1D_EVENT	1D measurement events.
POINT2D_EVENT	2D measurement events.
POINT3D_EVENT	3D measurement events.
POINT4D_EVENT	4D measurement events.
SPIKE_EVENT	Spike events.
MATRIX4x4_EVENT	4D matrix events.

4.9.5 Function Documentation

4.9.5.1 caerEventPacketAllocate()

```
static caerEventPacketHeader caerEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow,
            int16_t eventType,
            int32_t eventSize,
            int32_t eventTSOffset )  [inline], [static]
```

Allocate memory for an event packet and fill its header with the proper information. THIS FUNCTION IS INTENDED FOR INTERNAL USE ONLY BY THE VARIOUS EVENT PACKET TYPES FOR MEMORY ALLOCATION.

Returns

memory for an event packet, NULL on error.

4.9.5.2 caerEventPacketAppend()

```
static caerEventPacketHeader caerEventPacketAppend (
            caerEventPacketHeader packet,
            caerEventPacketHeader appendPacket )  [inline], [static]
```

Appends an event packet to another. This is a simple append operation, no timestamp reordering is done. Please ensure time is monotonically increasing over the two packets! Use free() to reclaim this memory afterwards.

Parameters

packet	the main events packet.
appendPacket	the events packet to append on the main one.

Returns

a valid event packet handle or NULL on error. On success, the old packet handle is to be considered invalid and not to be used anymore. On failure, the old packet handle is not touched in any way. The appendPacket handle is never touched in any way.

4.9.5.3 caerEventPacketClean()

```
static void caerEventPacketClean (
            caerEventPacketHeader packet )  [inline], [static]
```

Clean a packet by removing all invalid events, so that the total number of events is the number of valid events. The packet's capacity doesn't change.

Parameters

packet	an event packet to clean.

4.9.5.4 caerEventPacketClear()

```
static void caerEventPacketClear (
            caerEventPacketHeader packet )  [inline], [static]
```

Clear a packet by zeroing out all events. Capacity doesn't change, event number is set to zero.

Parameters

packet	an event packet to clear out.

4.9.5.5 caerEventPacketCopy()

```
static caerEventPacketHeader caerEventPacketCopy (
            caerEventPacketHeaderConst packet )  [inline], [static]
```

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

Parameters

packet	an event packet to copy.

Returns

a full copy of an event packet.

4.9.5.6 caerEventPacketCopyOnlyEvents()

```
static caerEventPacketHeader caerEventPacketCopyOnlyEvents (
            caerEventPacketHeaderConst packet )  [inline], [static]
```

Make a copy of an event packet, sized down to only include the currently present events (eventNumber, valid+invalid), and not including the possible extra unused events (up to eventCapacity).

Parameters

packet	an event packet to copy.

Returns

a sized down copy of an event packet.

4.9.5.7 caerEventPacketCopyOnlyValidEvents()

```
static caerEventPacketHeader caerEventPacketCopyOnlyValidEvents (
            caerEventPacketHeaderConst packet )  [inline], [static]
```

Make a copy of an event packet, sized down to only include the currently valid events (eventValid), and discarding everything else.

Parameters

packet	an event packet to copy.

Returns

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

4.9.5.8 caerEventPacketEquals()

```
static bool caerEventPacketEquals (
            caerEventPacketHeaderConst firstPacket,
            caerEventPacketHeaderConst secondPacket )  [inline], [static]
```

Verify if two event packets are equal. This means that the header and all events are equal.

Parameters

firstPacket	an event packet to be compared.
secondPacket	the other event packet to compare against.

Returns

true if both are the same, false otherwise.

4.9.5.9 caerEventPacketGetDataSize()

```
static int64_t caerEventPacketGetDataSize (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Get the data size of an event packet, in bytes. This is only the size of the data portion, excluding the header.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the event packet data size in bytes.

4.9.5.10 caerEventPacketGetDataSizeEvents()

```
static int64_t caerEventPacketGetDataSizeEvents (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Get the data size of an event packet, in bytes, up to its last actual event. This means only up to EventNumber, not up to EventCapacity. This is only the size of the data portion, excluding the header.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the event packet data size in bytes (up to event number).

4.9.5.11 caerEventPacketGetSize()

```
static int64_t caerEventPacketGetSize (
              caerEventPacketHeaderConst header )  [inline], [static]
```

Get the full size of an event packet, in bytes. This includes both the header and the data portion.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the event packet size in bytes.

4.9.5.12 caerEventPacketGetSizeEvents()

```
static int64_t caerEventPacketGetSizeEvents (
              caerEventPacketHeaderConst header )  [inline], [static]
```

Get the full size of an event packet, in bytes, up to its last actual event. This means only up to EventNumber, not up to EventCapacity. This includes both the header and the data portion.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the event packet size in bytes (up to event number).

4.9.5.13 caerEventPacketGrow()

```
static caerEventPacketHeader caerEventPacketGrow (
              caerEventPacketHeader packet,
              int32_t newEventCapacity )  [inline], [static]
```

Grows an event packet. This only supports strictly increasing the size of a packet. For a more flexible resize operation, see caerEventPacketResize(). Use free() to reclaim this memory afterwards.

Parameters

packet	the current event packet.
newEventCapacity	the new maximum number of events this packet can hold. Cannot be zero.

Returns

a valid event packet handle or NULL on error. On success, the old packet handle is to be considered invalid and not to be used anymore. On failure, the old packet handle is not touched in any way.

4.9.5.14 caerEventPacketHeaderGetEventCapacity()

```
static int32_t caerEventPacketHeaderGetEventCapacity (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Get the maximum number of events this packet can store.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the number of events this packet can hold.

4.9.5.15 caerEventPacketHeaderGetEventNumber()

```
static int32_t caerEventPacketHeaderGetEventNumber (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Get the number of events currently stored in this packet, considering both valid and invalid events.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the number of events in this packet.

4.9.5.16 caerEventPacketHeaderGetEventSize()

```
static int32_t caerEventPacketHeaderGetEventSize (
             caerEventPacketHeaderConst header )  [inline], [static]
```

Get the size of a single event, in bytes. All events inside an event packet always have the same size.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the event size in bytes.

4.9.5.17 caerEventPacketHeaderGetEventSource()

```
static int16_t caerEventPacketHeaderGetEventSource (
             caerEventPacketHeaderConst header )  [inline], [static]
```

Get the numerical event source ID, representing the event source that generated all the events present in this packet.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the numerical event source ID.

4.9.5.18 caerEventPacketHeaderGetEventTSOffset()

```
static int32_t caerEventPacketHeaderGetEventTSOffset (
             caerEventPacketHeaderConst header )  [inline], [static]
```

Get the offset, in bytes, to where the field with the main 32 bit timestamp is stored. This is useful for generic access to the timestamp field, given that different event types might have it at different offsets or might even have multiple timestamps, in which case this offset references the 'main' timestamp, the most representative one.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the event timestamp offset in bytes.

4.9.5.19 caerEventPacketHeaderGetEventTSOverflow()

```
static int32_t caerEventPacketHeaderGetEventTSOverflow (
             caerEventPacketHeaderConst header )  [inline], [static]
```

Get the 32 bit timestamp overflow counter (in microseconds). This is per-packet and is used to generate a 64 bit timestamp that never wraps around. Since timestamps wrap around after some time, being only 31 bit (32 bit signed int), another timestamp at the packet level provides another 31 bit (32 bit signed int), to enable the generation of a 62 bit (64 bit signed int) microsecond timestamp which is guaranteed to never wrap around (in the next 146'138 years at least).

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the packet-level timestamp overflow counter, in microseconds.

4.9.5.20 caerEventPacketHeaderGetEventType()

```
static int16_t caerEventPacketHeaderGetEventType (
             caerEventPacketHeaderConst header )  [inline], [static]
```

Return the numerical event type ID, representing the event type this EventPacket is containing.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the numerical event type (see 'enum caer_default_event_types').

4.9.5.21 caerEventPacketHeaderGetEventValid()

```
static int32_t caerEventPacketHeaderGetEventValid (
             caerEventPacketHeaderConst header )  [inline], [static]
```

Get the number of valid events in this packet, disregarding invalid ones (where the invalid mark is set).

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.

Returns

the number of valid events in this packet.

4.9.5.22 caerEventPacketHeaderSetEventCapacity()

```
static void caerEventPacketHeaderSetEventCapacity (
             caerEventPacketHeader header,
             int32_t eventsCapacity )  [inline], [static]
```

Set the maximum number of events this packet can store. This is determined at packet allocation time and should not be changed during the life-time of the packet.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventsCapacity	the number of events this packet can hold.

4.9.5.23 caerEventPacketHeaderSetEventNumber()

```
static void caerEventPacketHeaderSetEventNumber (
             caerEventPacketHeader header,
             int32_t eventsNumber )  [inline], [static]
```

Set the number of events currently stored in this packet, considering both valid and invalid events.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventsNumber	the number of events in this packet.

4.9.5.24 caerEventPacketHeaderSetEventSize()

```
static void caerEventPacketHeaderSetEventSize (
             caerEventPacketHeader header,
             int32_t eventSize )  [inline], [static]
```

Set the size of a single event, in bytes. All events inside an event packet always have the same size.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventSize	the event size in bytes.

4.9.5.25 caerEventPacketHeaderSetEventSource()

```
static void caerEventPacketHeaderSetEventSource (
             caerEventPacketHeader header,
             int16_t eventSource )  [inline], [static]
```

Set the numerical event source ID, representing the event source that generated all the events present in this packet. This ID should be unique at least within a process, if not within the whole system, to guarantee correct identification of who generated an event later on.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventSource	the numerical event source ID.

4.9.5.26 caerEventPacketHeaderSetEventTSOffset()

```
static void caerEventPacketHeaderSetEventTSOffset (
             caerEventPacketHeader header,
             int32_t eventTSOffset )  [inline], [static]
```

Set the offset, in bytes, to where the field with the main 32 bit timestamp is stored. This is useful for generic access to the timestamp field, given that different event types might have it at different offsets or might even have multiple timestamps, in which case this offset references the 'main' timestamp, the most representative one.

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventTSOffset	the event timestamp offset in bytes.

4.9.5.27 caerEventPacketHeaderSetEventTSOverflow()

```
static void caerEventPacketHeaderSetEventTSOverflow (
             caerEventPacketHeader header,
             int32_t eventTSOverflow )  [inline], [static]
```

Set the 32 bit timestamp overflow counter (in microseconds). This is per-packet and is used to generate a 64 bit timestamp that never wraps around. Since timestamps wrap around after some time, being only 31 bit (32 bit signed int), another timestamp at the packet level provides another 31 bit (32 bit signed int), to enable the generation of a 62 bit (64 bit signed int) microsecond timestamp which is guaranteed to never wrap around (in the next 146'138 years at least).

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventTSOverflow	the packet-level timestamp overflow counter, in microseconds.

4.9.5.28 caerEventPacketHeaderSetEventType()

```
static void caerEventPacketHeaderSetEventType (
            caerEventPacketHeader header,
            int16_t eventType )  [inline], [static]
```

Set the numerical event type ID, representing the event type this EventPacket will contain. All event types below 100 are reserved for use by libcaer and cAER. DO NOT USE THEM FOR YOUR OWN EVENT TYPES!

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventType	the numerical event type (see 'enum caer_default_event_types').

4.9.5.29 caerEventPacketHeaderSetEventValid()

```
static void caerEventPacketHeaderSetEventValid (
            caerEventPacketHeader header,
            int32_t eventsValid )  [inline], [static]
```

Set the number of valid events in this packet, disregarding invalid ones (where the invalid mark is set).

Parameters

header	a valid EventPacket header pointer. Cannot be NULL.
eventsValid	the number of valid events in this packet.

4.9.5.30 caerEventPacketResize()

```
static caerEventPacketHeader caerEventPacketResize (
            caerEventPacketHeader packet,
            int32_t newEventCapacity )  [inline], [static]
```

Resize an event packet. First, the packet is cleaned (all invalid events removed), then:

- If the old and new event capacity are equal, nothing else changes.
- If the new capacity is bigger, the packet is enlarged and the new events are initialized to all zeros (invalid).
- If the new capacity is smaller, the packet is truncated at the given point. Use free() to reclaim this memory afterwards.

Parameters

packet	the current event packet.
newEventCapacity	the new maximum number of events this packet can hold. Cannot be zero.

Returns

a valid event packet handle or NULL on error. On success, the old packet handle is to be considered invalid and not to be used anymore. On failure, the old packet handle is still valid, but will have been cleaned of all invalid events!

4.9.5.31 caerGenericEventCopy()

```
static bool caerGenericEventCopy (
            void * eventPtrDestination,
            const void * eventPtrSource,
            caerEventPacketHeaderConst headerPtrDestination,
            caerEventPacketHeaderConst headerPtrSource )  [inline], [static]
```

Copy a given event's content to another location in memory.

Parameters

eventPtrDestination	a generic pointer to an event to copy to. Cannot be NULL.
eventPtrSource	a generic pointer to an event to copy from. Cannot be NULL.
headerPtrDestination	a valid EventPacket header pointer from the destination packet. Cannot be NULL.
headerPtrSource	a valid EventPacket header pointer from the source packet. Cannot be NULL.

Returns

true on successful copy, false otherwise.

4.9.5.32 caerGenericEventGetEvent()

```
static const void* caerGenericEventGetEvent (
            caerEventPacketHeaderConst headerPtr,
            int32_t n )  [inline], [static]
```

Get a generic pointer to an event, without having to know what event type the packet is containing.

Parameters

headerPtr	a valid EventPacket header pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventNumber[bounds.

Returns

a generic pointer to the requested event. NULL on error. This points to unmodifiable memory, as it should never be used for anything other than read operations, such as caerGenericEventGetTimestamp(). Don't modify the memory, you have no idea what it is! If you do know, just use the proper typed packet functions.

4.9.5.33 caerGenericEventGetTimestamp()

```
static int32_t caerGenericEventGetTimestamp (
            const void * eventPtr,
            caerEventPacketHeaderConst headerPtr )  [inline], [static]
```

Get the main 32 bit timestamp for a generic event, without having to know what event type the packet is containing.

Parameters

eventPtr	a generic pointer to an event. Cannot be NULL.
headerPtr	a valid EventPacket header pointer. Cannot be NULL.

Returns

the main 32 bit timestamp of this event.

4.9.5.34 caerGenericEventGetTimestamp64()

```
static int64_t caerGenericEventGetTimestamp64 (
            const void * eventPtr,
            caerEventPacketHeaderConst headerPtr )  [inline], [static]
```

Get the main 64 bit timestamp for a generic event, without having to know what event type the packet is containing. This takes the per-packet timestamp into account too, generating a timestamp that doesn't suffer from overflow problems.

Parameters

eventPtr	a generic pointer to an event. Cannot be NULL.
headerPtr	a valid EventPacket header pointer. Cannot be NULL.

Returns

the main 64 bit timestamp of this event.

4.9.5.35 caerGenericEventIsValid()

```
static bool caerGenericEventIsValid (
            const void * eventPtr )  [inline], [static]
```

Check if the given generic event is valid or not.

Parameters

eventPtr	a generic pointer to an event. Cannot be NULL.

Returns

true if the event is valid, false otherwise.

4.9.5.36 PACKED_STRUCT()

```
PACKED_STRUCT (
            struct caer_event_packet_header { int16_t eventType;int16_t eventSource;int32_t
eventSize;int32_t eventTSOffset;int32_t eventTSOverflow;int32_t eventCapacity;int32_t event←
Number;int32_t eventValid;}  )
```

EventPacket header data structure definition. The size, also defined in CAER_EVENT_PACKET_HEADER_SIZE, must always be constant. The header is common to all types of event packets and is always the very first member of an event packet data structure. Signed integers are used for compatibility with languages that do not have unsigned ones, such as Java.

4.10 events/config.h File Reference

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

Macros

- #define CAER_CONFIGURATION_ITERATOR_ALL_START(CONFIGURATION_PACKET)
- #define CAER_CONFIGURATION_CONST_ITERATOR_ALL_START(CONFIGURATION_PACKET)
- #define CAER_CONFIGURATION_ITERATOR_ALL_END }
- #define CAER_CONFIGURATION_ITERATOR_VALID_START(CONFIGURATION_PACKET)
- #define CAER_CONFIGURATION_CONST_ITERATOR_VALID_START(CONFIGURATION_PACKET)
- #define CAER_CONFIGURATION_ITERATOR_VALID_END }

- #define CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_START(CONFIGURATION_PACKET)
- #define CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_ALL_START(CONFIGURATION_P←ACKET)
- #define CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_END }
- #define CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_START(CONFIGURATION_PACKET)
- #define CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_VALID_START(CONFIGURATION_←PACKET)
- #define CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_END }

- #define CONFIG_MODULE_ADDR_SHIFT 1
- #define CONFIG_MODULE_ADDR_MASK 0x0000007F

Typedefs

- typedef struct caer_configuration_event ∗ caerConfigurationEvent
- typedef const struct caer_configuration_event ∗ **caerConfigurationEventConst**
- typedef struct caer_configuration_event_packet ∗ caerConfigurationEventPacket
- typedef const struct caer_configuration_event_packet ∗ **caerConfigurationEventPacketConst**

Functions

- PACKED_STRUCT (struct caer_configuration_event { uint8_t moduleAddress;uint8_t parameter←Address;uint32_t parameter;int32_t timestamp;})
- PACKED_STRUCT (struct caer_configuration_event_packet { struct caer_event_packet_header packet←Header;struct caer_configuration_event events[ ];})
- static caerConfigurationEventPacket caerConfigurationEventPacketAllocate (int32_t eventCapacity, int16_←t eventSource, int32_t tsOverflow)
- static caerConfigurationEventPacket caerConfigurationEventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerConfigurationEventPacketConst caerConfigurationEventPacketFromPacketHeaderConst (caer←EventPacketHeaderConst header)
- static caerConfigurationEvent caerConfigurationEventPacketGetEvent (caerConfigurationEventPacket packet, int32_t n)
- static caerConfigurationEventConst caerConfigurationEventPacketGetEventConst (caerConfiguration←EventPacketConst packet, int32_t n)
- static int32_t caerConfigurationEventGetTimestamp (caerConfigurationEventConst event)
- static int64_t caerConfigurationEventGetTimestamp64 (caerConfigurationEventConst event, caer←ConfigurationEventPacketConst packet)
- static void caerConfigurationEventSetTimestamp (caerConfigurationEvent event, int32_t timestamp)
- static bool caerConfigurationEventIsValid (caerConfigurationEventConst event)
- static void caerConfigurationEventValidate (caerConfigurationEvent event, caerConfigurationEventPacket packet)
- static void caerConfigurationEventInvalidate (caerConfigurationEvent event, caerConfigurationEventPacket packet)
- static uint8_t caerConfigurationEventGetModuleAddress (caerConfigurationEventConst event)
- static void caerConfigurationEventSetModuleAddress (caerConfigurationEvent event, uint8_t module←Address)
- static uint8_t caerConfigurationEventGetParameterAddress (caerConfigurationEventConst event)
- static void caerConfigurationEventSetParameterAddress (caerConfigurationEvent event, uint8_t parameter←Address)
- static uint32_t caerConfigurationEventGetParameter (caerConfigurationEventConst event)
- static void caerConfigurationEventSetParameter (caerConfigurationEvent event, uint32_t parameter)

4.10.1 Detailed Description

Configuration Events format definition and handling functions. This event contains information about the current configuration of the device. By having configuration as a standardized event format, it becomes host-software agnostic, and it also becomes part of the event stream, enabling easy tracking of changes through time, by putting them into the event stream at the moment they happen. While the resolution of the timestamps for these events is in microseconds for compatibility with all other event types, the precision is in the order of ∼1-20 milliseconds, given that these events are generated and injected on the host-side.

4.10.2 Macro Definition Documentation

4.10.2.1 CAER_CONFIGURATION_CONST_ITERATOR_ALL_START

```
#define CAER_CONFIGURATION_CONST_ITERATOR_ALL_START(
                CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter = 0;                                  \
         caerConfigurationIteratorCounter                                              \
         < caerEventPacketHeaderGetEventNumber(&(CONFIGURATION_PACKET)->
      packetHeader); \
         caerConfigurationIteratorCounter++) {                                         \
        caerConfigurationEventConst caerConfigurationIteratorElement                   \
            = caerConfigurationEventPacketGetEventConst(
      CONFIGURATION_PACKET, caerConfigurationIteratorCounter);
```

Const-Iterator over all configuration events in a packet. Returns the current index in the 'caerConfigurationIterator←Counter' variable of type 'int32_t' and the current read-only event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEventConst.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.2 CAER_CONFIGURATION_CONST_ITERATOR_VALID_START

```
#define CAER_CONFIGURATION_CONST_ITERATOR_VALID_START(
                CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter = 0;
        \
         caerConfigurationIteratorCounter
            \
         < caerEventPacketHeaderGetEventNumber(&(CONFIGURATION_PACKET)->
      packetHeader);                             \
         caerConfigurationIteratorCounter++) {
            \
        caerConfigurationEventConst caerConfigurationIteratorElement
            \
            = caerConfigurationEventPacketGetEventConst(
      CONFIGURATION_PACKET, caerConfigurationIteratorCounter); \
        if (!caerConfigurationEventIsValid(caerConfigurationIteratorElement))
      {                                    \
            continue;
            \
        }
```

Const-Iterator over only the valid configuration events in a packet. Returns the current index in the 'caer←ConfigurationIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerConfiguration←IteratorElement' variable of type caerConfigurationEventConst.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.3 CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_ALL_START(
              CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter                                   \
        = caerEventPacketHeaderGetEventNumber(&(CONFIGURATION_PACKET)->
     packetHeader) - 1; \
        caerConfigurationIteratorCounter >= 0; caerConfigurationIteratorCounter--) {      \
       caerConfigurationEventConst caerConfigurationIteratorElement                       \
           = caerConfigurationEventPacketGetEventConst(
     CONFIGURATION_PACKET, caerConfigurationIteratorCounter);
```

Const-Reverse iterator over all configuration events in a packet. Returns the current index in the 'caer⤶ ConfigurationIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerConfiguration⤶ IteratorElement' variable of type caerConfigurationEventConst.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.4 CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_CONFIGURATION_CONST_REVERSE_ITERATOR_VALID_START(
              CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter
       \
        = caerEventPacketHeaderGetEventNumber(&(CONFIGURATION_PACKET)->
     packetHeader) - 1;                          \
        caerConfigurationIteratorCounter >= 0; caerConfigurationIteratorCounter--) {
           \
       caerConfigurationEventConst caerConfigurationIteratorElement
           \
           = caerConfigurationEventPacketGetEventConst(
     CONFIGURATION_PACKET, caerConfigurationIteratorCounter); \
       if (!caerConfigurationEventIsValid(caerConfigurationIteratorElement))
     {                                 \
           continue;
           \
       }
```

Const-Reverse iterator over only the valid configuration events in a packet. Returns the current index in the 'caer⤶ ConfigurationIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerConfiguration⤶ IteratorElement' variable of type caerConfigurationEventConst.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.5 CAER_CONFIGURATION_ITERATOR_ALL_END

```
#define CAER_CONFIGURATION_ITERATOR_ALL_END }
```

Iterator close statement.

4.10.2.6 CAER_CONFIGURATION_ITERATOR_ALL_START

```
#define CAER_CONFIGURATION_ITERATOR_ALL_START(
               CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter = 0;                                    \
        caerConfigurationIteratorCounter                                                  \
        < caerEventPacketHeaderGetEventNumber(&(CONFIGURATION_PACKET)->
     packetHeader); \
        caerConfigurationIteratorCounter++) {                                             \
       caerConfigurationEvent caerConfigurationIteratorElement                            \
           = caerConfigurationEventPacketGetEvent(CONFIGURATION_PACKET
     , caerConfigurationIteratorCounter);
```

Iterator over all configuration events in a packet. Returns the current index in the 'caerConfigurationIterator↩ Counter' variable of type 'int32_t' and the current event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEvent.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.7 CAER_CONFIGURATION_ITERATOR_VALID_END

```
#define CAER_CONFIGURATION_ITERATOR_VALID_END }
```

Iterator close statement.

4.10.2.8 CAER_CONFIGURATION_ITERATOR_VALID_START

```
#define CAER_CONFIGURATION_ITERATOR_VALID_START(
               CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter = 0;                                                    \
        caerConfigurationIteratorCounter
      \
        < caerEventPacketHeaderGetEventNumber(&(CONFIGURATION_PACKET)->
     packetHeader);                      \
        caerConfigurationIteratorCounter++) {
      \
       caerConfigurationEvent caerConfigurationIteratorElement
      \
           = caerConfigurationEventPacketGetEvent(CONFIGURATION_PACKET
     , caerConfigurationIteratorCounter); \
       if (!caerConfigurationEventIsValid(caerConfigurationIteratorElement))
     {                              \
           continue;
      \
       }
```

Iterator over only the valid configuration events in a packet. Returns the current index in the 'caerConfiguration↩ IteratorCounter' variable of type 'int32_t' and the current event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEvent.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.9 CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_END

```
#define CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.10.2.10 CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_START

```
#define CAER_CONFIGURATION_REVERSE_ITERATOR_ALL_START(
                CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter                                          \
         = caerEventPacketHeaderGetEventNumber(&(CONFIGURATION_PACKET)->
      packetHeader) - 1; \
         caerConfigurationIteratorCounter >= 0; caerConfigurationIteratorCounter--) {      \
        caerConfigurationEvent caerConfigurationIteratorElement                            \
            = caerConfigurationEventPacketGetEvent(CONFIGURATION_PACKET
      , caerConfigurationIteratorCounter);
```

Reverse iterator over all configuration events in a packet. Returns the current index in the 'caerConfiguration←IteratorCounter' variable of type 'int32_t' and the current event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEvent.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.11 CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_END

```
#define CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.10.2.12 CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_START

```
#define CAER_CONFIGURATION_REVERSE_ITERATOR_VALID_START(
                CONFIGURATION_PACKET )
```

Value:

```
for (int32_t caerConfigurationIteratorCounter                                                        \
         = caerEventPacketHeaderGetEventNumber(&(CONFIGURATION_PACKET)->
      packetHeader) - 1;                  \
         caerConfigurationIteratorCounter >= 0; caerConfigurationIteratorCounter--) {
       \
        caerConfigurationEvent caerConfigurationIteratorElement
       \
            = caerConfigurationEventPacketGetEvent(CONFIGURATION_PACKET
      , caerConfigurationIteratorCounter); \
        if (!caerConfigurationEventIsValid(caerConfigurationIteratorElement))
      {                             \
            continue;
       \
        }
```

Reverse iterator over only the valid configuration events in a packet. Returns the current index in the 'caer←ConfigurationIteratorCounter' variable of type 'int32_t' and the current event in the 'caerConfigurationIteratorElement' variable of type caerConfigurationEvent.

CONFIGURATION_PACKET: a valid ConfigurationEventPacket pointer. Cannot be NULL.

4.10.2.13 CONFIG_MODULE_ADDR_MASK

```
#define CONFIG_MODULE_ADDR_MASK 0x0000007F
```

Shift and mask values for the module address. Module address is only 7 bits, since the eighth bit is used device-side to differentiate reads from writes. Here we can just re-use it for the validity mark.

4.10.2.14 CONFIG_MODULE_ADDR_SHIFT

```
#define CONFIG_MODULE_ADDR_SHIFT 1
```

Shift and mask values for the module address. Module address is only 7 bits, since the eighth bit is used device-side to differentiate reads from writes. Here we can just re-use it for the validity mark.

4.10.3 Typedef Documentation

4.10.3.1 caerConfigurationEvent

```
typedef struct caer_configuration_event* caerConfigurationEvent
```

Type for pointer to configuration event data structure.

4.10.3.2 caerConfigurationEventPacket

```
typedef struct caer_configuration_event_packet* caerConfigurationEventPacket
```

Type for pointer to configuration event packet data structure.

4.10.4 Function Documentation

4.10.4.1 caerConfigurationEventGetModuleAddress()

```
static uint8_t caerConfigurationEventGetModuleAddress (
             caerConfigurationEventConst event )  [inline], [static]
```

Get the configuration event's module address.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.

Returns

configuration module address.

4.10.4.2 caerConfigurationEventGetParameter()

```
static uint32_t caerConfigurationEventGetParameter (
             caerConfigurationEventConst event )  [inline], [static]
```

Get the configuration event's parameter.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.

Returns

configuration parameter.

4.10.4.3 caerConfigurationEventGetParameterAddress()

```
static uint8_t caerConfigurationEventGetParameterAddress (
             caerConfigurationEventConst event )  [inline], [static]
```

Get the configuration event's parameter address.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.

Returns

configuration parameter address.

4.10.4.4 caerConfigurationEventGetTimestamp()

```
static int32_t caerConfigurationEventGetTimestamp (
             caerConfigurationEventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.10.4.5 caerConfigurationEventGetTimestamp64()

```
static int64_t caerConfigurationEventGetTimestamp64 (
            caerConfigurationEventConst event,
            caerConfigurationEventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
packet	the ConfigurationEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.10.4.6 caerConfigurationEventInvalidate()

```
static void caerConfigurationEventInvalidate (
            caerConfigurationEvent event,
            caerConfigurationEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
packet	the ConfigurationEventPacket pointer for the packet containing this event. Cannot be NULL.

4.10.4.7 caerConfigurationEventIsValid()

```
static bool caerConfigurationEventIsValid (
            caerConfigurationEventConst event )  [inline], [static]
```

Check if this configuration event is valid.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.10.4.8 caerConfigurationEventPacketAllocate()

```
static caerConfigurationEventPacket caerConfigurationEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )  [inline], [static]
```

Allocate a new configuration events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid ConfigurationEventPacket handle or NULL on error.

4.10.4.9 caerConfigurationEventPacketFromPacketHeader()

```
static caerConfigurationEventPacket caerConfigurationEventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into a Configuration event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.10.4.10 caerConfigurationEventPacketFromPacketHeaderConst()

```
static caerConfigurationEventPacketConst caerConfigurationEventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only Configuration event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.10.4.11 caerConfigurationEventPacketGetEvent()

```
static caerConfigurationEvent caerConfigurationEventPacketGetEvent (
            caerConfigurationEventPacket packet,
            int32_t n )  [inline], [static]
```

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

Parameters

packet	a valid ConfigurationEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested configuration event. NULL on error.

4.10.4.12 caerConfigurationEventPacketGetEventConst()

```
static caerConfigurationEventConst caerConfigurationEventPacketGetEventConst (
            caerConfigurationEventPacketConst packet,
            int32_t n )  [inline], [static]
```

Get the configuration event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid ConfigurationEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only configuration event. NULL on error.

4.10.4.13 caerConfigurationEventSetModuleAddress()

```
static void caerConfigurationEventSetModuleAddress (
            caerConfigurationEvent event,
            uint8_t moduleAddress )  [inline], [static]
```

Set the configuration event's module address.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
moduleAddress	configuration module address.

4.10.4.14 caerConfigurationEventSetParameter()

```
static void caerConfigurationEventSetParameter (
            caerConfigurationEvent event,
            uint32_t parameter )  [inline], [static]
```

Set the configuration event's parameter.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
parameter	configuration parameter.

4.10.4.15 caerConfigurationEventSetParameterAddress()

```
static void caerConfigurationEventSetParameterAddress (
            caerConfigurationEvent event,
            uint8_t parameterAddress )  [inline], [static]
```

Set the configuration event's parameter address.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
parameterAddress	configuration parameter address.

4.10.4.16 caerConfigurationEventSetTimestamp()

```
static void caerConfigurationEventSetTimestamp (
            caerConfigurationEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.10.4.17 caerConfigurationEventValidate()

```
static void caerConfigurationEventValidate (
            caerConfigurationEvent event,
            caerConfigurationEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid ConfigurationEvent pointer. Cannot be NULL.
packet	the ConfigurationEventPacket pointer for the packet containing this event. Cannot be NULL.

4.10.4.18 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_configuration_event { uint8_t moduleAddress;uint8_t parameterAddress;uint32←
_t parameter;int32_t timestamp;}  )
```

Configuration event data structure definition. This contains the actual configuration module address, the parameter address and the actual parameter content, as well as the 32 bit event timestamp. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.10.4.19 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
              struct caer_configuration_event_packet { struct caer_event_packet_header packet↩
Header;struct caer_configuration_event events[];}  )
```

Configuration event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.11 events/ear.h File Reference

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

Macros

- #define CAER_EAR_ITERATOR_ALL_START(EAR_PACKET)
- #define CAER_EAR_CONST_ITERATOR_ALL_START(EAR_PACKET)
- #define CAER_EAR_ITERATOR_ALL_END }
- #define CAER_EAR_ITERATOR_VALID_START(EAR_PACKET)
- #define CAER_EAR_CONST_ITERATOR_VALID_START(EAR_PACKET)
- #define CAER_EAR_ITERATOR_VALID_END }
- #define CAER_EAR_REVERSE_ITERATOR_ALL_START(EAR_PACKET)
- #define CAER_EAR_CONST_REVERSE_ITERATOR_ALL_START(EAR_PACKET)
- #define CAER_EAR_REVERSE_ITERATOR_ALL_END }
- #define CAER_EAR_REVERSE_ITERATOR_VALID_START(EAR_PACKET)
- #define CAER_EAR_CONST_REVERSE_ITERATOR_VALID_START(EAR_PACKET)
- #define CAER_EAR_REVERSE_ITERATOR_VALID_END }

- #define EAR_SHIFT 1
- #define EAR_MASK 0x0000000F
- #define EAR_CHANNEL_SHIFT 5
- #define EAR_CHANNEL_MASK 0x000007FF
- #define EAR_NEURON_SHIFT 16
- #define EAR_NEURON_MASK 0x000000FF
- #define EAR_FILTER_SHIFT 24
- #define EAR_FILTER_MASK 0x000000FF

Typedefs

- typedef struct caer_ear_event ∗ caerEarEvent
- typedef const struct caer_ear_event ∗ **caerEarEventConst**
- typedef struct caer_ear_event_packet ∗ caerEarEventPacket
- typedef const struct caer_ear_event_packet ∗ **caerEarEventPacketConst**

Functions

- PACKED_STRUCT (struct caer_ear_event { uint32_t data;int32_t timestamp;})
- PACKED_STRUCT (struct caer_ear_event_packet { struct caer_event_packet_header packetHeader;struct caer_ear_event events[ ];})
- static caerEarEventPacket caerEarEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerEarEventPacket caerEarEventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerEarEventPacketConst caerEarEventPacketFromPacketHeaderConst (caerEventPacketHeader↩ Const header)
- static caerEarEvent caerEarEventPacketGetEvent (caerEarEventPacket packet, int32_t n)
- static caerEarEventConst caerEarEventPacketGetEventConst (caerEarEventPacketConst packet, int32_t n)
- static int32_t caerEarEventGetTimestamp (caerEarEventConst event)
- static int64_t caerEarEventGetTimestamp64 (caerEarEventConst event, caerEarEventPacketConst packet)
- static void caerEarEventSetTimestamp (caerEarEvent event, int32_t timestamp)
- static bool caerEarEventIsValid (caerEarEventConst event)
- static void caerEarEventValidate (caerEarEvent event, caerEarEventPacket packet)
- static void caerEarEventInvalidate (caerEarEvent event, caerEarEventPacket packet)
- static uint8_t caerEarEventGetEar (caerEarEventConst event)
- static void caerEarEventSetEar (caerEarEvent event, uint8_t ear)
- static uint16_t caerEarEventGetChannel (caerEarEventConst event)
- static void caerEarEventSetChannel (caerEarEvent event, uint16_t channel)
- static uint8_t **caerEarEventGetNeuron** (caerEarEventConst event)
- static void **caerEarEventSetNeuron** (caerEarEvent event, uint8_t neuron)
- static uint8_t **caerEarEventGetFilter** (caerEarEventConst event)
- static void **caerEarEventSetFilter** (caerEarEvent event, uint8_t filter)

4.11.1 Detailed Description

Ear (Cochlea) Events format definition and handling functions. This encodes events from a silicon cochlea chip, containing information about which ear (microphone) generated the event, as well as which channel was involved and additional information on filters and neurons.

4.11.2 Macro Definition Documentation

4.11.2.1 CAER_EAR_CONST_ITERATOR_ALL_START

```
#define CAER_EAR_CONST_ITERATOR_ALL_START(
                EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = 0;                                        \
         caerEarIteratorCounter < caerEventPacketHeaderGetEventNumber(&(
      EAR_PACKET)->packetHeader); \
         caerEarIteratorCounter++) {                                                \
        caerEarEventConst caerEarIteratorElement =
      caerEarEventPacketGetEventConst(EAR_PACKET, caerEarIteratorCounter);
```

Const-Iterator over all ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerEarIteratorElement' variable of type caerEarEventConst.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.2 CAER_EAR_CONST_ITERATOR_VALID_START

```
#define CAER_EAR_CONST_ITERATOR_VALID_START(
                EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = 0;                                                    \
         caerEarIteratorCounter < caerEventPacketHeaderGetEventNumber(&(
      EAR_PACKET)->packetHeader); \
         caerEarIteratorCounter++) {                                                          \
        caerEarEventConst caerEarIteratorElement                                              \
            = caerEarEventPacketGetEventConst(EAR_PACKET,
      caerEarIteratorCounter);                    \
        if (!caerEarEventIsValid(caerEarIteratorElement)) {
                  \
            continue;                                                                         \
        }
```

Const-Iterator over only the valid ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerEarIteratorElement' variable of type caerEar←EventConst.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.3 CAER_EAR_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_EAR_CONST_REVERSE_ITERATOR_ALL_START(
                EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = caerEventPacketHeaderGetEventNumber
       (&(EAR_PACKET)->packetHeader) - 1; \
         caerEarIteratorCounter >= 0; caerEarIteratorCounter--) {
           \
        caerEarEventConst caerEarIteratorElement =
      caerEarEventPacketGetEventConst(EAR_PACKET, caerEarIteratorCounter);
```

Const-Reverse iterator over all ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerEarIteratorElement' variable of type caerEar←EventConst.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.4 CAER_EAR_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_EAR_CONST_REVERSE_ITERATOR_VALID_START(
                EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = caerEventPacketHeaderGetEventNumber
       (&(EAR_PACKET)->packetHeader) - 1; \
         caerEarIteratorCounter >= 0; caerEarIteratorCounter--) {
           \
        caerEarEventConst caerEarIteratorElement
           \
            = caerEarEventPacketGetEventConst(EAR_PACKET,
      caerEarIteratorCounter);                                \
        if (!caerEarEventIsValid(caerEarIteratorElement)) {
                                \
            continue;
           \
        }
```

Const-Reverse iterator over only the valid ear events in a packet. Returns the current index in the 'caerEarIterator←Counter' variable of type 'int32_t' and the current read-only event in the 'caerEarIteratorElement' variable of type caerEarEventConst.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.5 CAER_EAR_ITERATOR_ALL_END

```
#define CAER_EAR_ITERATOR_ALL_END }
```

Iterator close statement.

4.11.2.6 CAER_EAR_ITERATOR_ALL_START

```
#define CAER_EAR_ITERATOR_ALL_START(
                EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = 0;                                    \
         caerEarIteratorCounter < caerEventPacketHeaderGetEventNumber(&(
      EAR_PACKET)->packetHeader); \
         caerEarIteratorCounter++) {                                          \
        caerEarEvent caerEarIteratorElement = caerEarEventPacketGetEvent(
      EAR_PACKET, caerEarIteratorCounter);
```

Iterator over all ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current event in the 'caerEarIteratorElement' variable of type caerEarEvent.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.7 CAER_EAR_ITERATOR_VALID_END

```
#define CAER_EAR_ITERATOR_VALID_END }
```

Iterator close statement.

4.11.2.8 CAER_EAR_ITERATOR_VALID_START

```
#define CAER_EAR_ITERATOR_VALID_START(
                EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = 0;                                           \
         caerEarIteratorCounter < caerEventPacketHeaderGetEventNumber(&(
      EAR_PACKET)->packetHeader);           \
         caerEarIteratorCounter++) {
         \
        caerEarEvent caerEarIteratorElement = caerEarEventPacketGetEvent(
      EAR_PACKET, caerEarIteratorCounter); \
        if (!caerEarEventIsValid(caerEarIteratorElement)) {
                            \
            continue;
         \
        }
```

Iterator over only the valid ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current event in the 'caerEarIteratorElement' variable of type caerEarEvent.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.9 CAER_EAR_REVERSE_ITERATOR_ALL_END

```
#define CAER_EAR_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.11.2.10 CAER_EAR_REVERSE_ITERATOR_ALL_START

```
#define CAER_EAR_REVERSE_ITERATOR_ALL_START(
              EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(EAR_PACKET)->packetHeader) - 1; \
         caerEarIteratorCounter >= 0; caerEarIteratorCounter--) {
           \
        caerEarEvent caerEarIteratorElement = caerEarEventPacketGetEvent(
      EAR_PACKET, caerEarIteratorCounter);
```

Reverse iterator over all ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current event in the 'caerEarIteratorElement' variable of type caerEarEvent.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.11 CAER_EAR_REVERSE_ITERATOR_VALID_END

```
#define CAER_EAR_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.11.2.12 CAER_EAR_REVERSE_ITERATOR_VALID_START

```
#define CAER_EAR_REVERSE_ITERATOR_VALID_START(
              EAR_PACKET )
```

Value:

```
for (int32_t caerEarIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(EAR_PACKET)->packetHeader) - 1; \
         caerEarIteratorCounter >= 0; caerEarIteratorCounter--) {
           \
        caerEarEvent caerEarIteratorElement = caerEarEventPacketGetEvent(
      EAR_PACKET, caerEarIteratorCounter);   \
        if (!caerEarEventIsValid(caerEarIteratorElement)) {
                              \
            continue;
           \
        }
```

Reverse iterator over only the valid ear events in a packet. Returns the current index in the 'caerEarIteratorCounter' variable of type 'int32_t' and the current event in the 'caerEarIteratorElement' variable of type caerEarEvent.

EAR_PACKET: a valid EarEventPacket pointer. Cannot be NULL.

4.11.2.13 EAR_CHANNEL_MASK

```
#define EAR_CHANNEL_MASK 0x000007FF
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.11.2.14 EAR_CHANNEL_SHIFT

```
#define EAR_CHANNEL_SHIFT 5
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.11.2.15 EAR_FILTER_MASK

```
#define EAR_FILTER_MASK 0x000000FF
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.11.2.16 EAR_FILTER_SHIFT

```
#define EAR_FILTER_SHIFT 24
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.11.2.17 EAR_MASK

```
#define EAR_MASK 0x0000000F
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.11.2.18 EAR_NEURON_MASK

```
#define EAR_NEURON_MASK 0x000000FF
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.11.2.19 EAR_NEURON_SHIFT

```
#define EAR_NEURON_SHIFT 16
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.11.2.20 EAR_SHIFT

```
#define EAR_SHIFT 1
```

Shift and mask values for the ear event values coming from a cochlea: the ear position (up to 16), the channel number (up to 2048), the ganglion (up to 256) and the filter (up to 256). Bit 0 is the valid mark, see 'common.h' for more details.

4.11.3 Typedef Documentation

4.11.3.1 caerEarEvent

```
typedef struct caer_ear_event* caerEarEvent
```

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

4.11.3.2 caerEarEventPacket

```
typedef struct caer_ear_event_packet* caerEarEventPacket
```

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

4.11.4 Function Documentation

4.11.4.1 caerEarEventGetChannel()

```
static uint16_t caerEarEventGetChannel (
             caerEarEventConst event )  [inline], [static]
```

Get the channel (frequency band) ID. The channels count from 0 upward, where 0 is the highest frequency channel, while higher numbers are progressively lower frequency channels. This is derived from how the actual human ear works.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.

Returns

the channel (frequency band) ID.

4.11.4.2 caerEarEventGetEar()

```
static uint8_t caerEarEventGetEar (
            caerEarEventConst event )  [inline], [static]
```

Get the numerical ID of the ear (microphone). Usually, 0 is left, 1 is right for 2 ear cochleas. For 4 ear cochleas, 0 is front left, 1 is front right, 2 is back left and 3 is back right.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.

Returns

the ear (microphone) ID.

4.11.4.3 caerEarEventGetTimestamp()

```
static int32_t caerEarEventGetTimestamp (
            caerEarEventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.11.4.4 caerEarEventGetTimestamp64()

```
static int64_t caerEarEventGetTimestamp64 (
            caerEarEventConst event,
            caerEarEventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.
packet	the EarEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.11.4.5 caerEarEventInvalidate()

```
static void caerEarEventInvalidate (
            caerEarEvent event,
            caerEarEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid EarEvent pointer. Cannot be NULL.
packet	the EarEventPacket pointer for the packet containing this event. Cannot be NULL.

4.11.4.6 caerEarEventIsValid()

```
static bool caerEarEventIsValid (
            caerEarEventConst event )  [inline], [static]
```

Check if this ear (cochlea) event is valid.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.11.4.7 caerEarEventPacketAllocate()

```
static caerEarEventPacket caerEarEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )  [inline], [static]
```

Allocate a new ear (cochlea) events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid EarEventPacket handle or NULL on error.

4.11.4.8 caerEarEventPacketFromPacketHeader()

```
static caerEarEventPacket caerEarEventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into an Ear event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.11.4.9 caerEarEventPacketFromPacketHeaderConst()

```
static caerEarEventPacketConst caerEarEventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only Ear event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.11.4.10 caerEarEventPacketGetEvent()

```
static caerEarEvent caerEarEventPacketGetEvent (
            caerEarEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the ear (cochlea) event at the given index from the event packet.

Parameters

packet	a valid EarEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested ear (cochlea) event. NULL on error.

4.11.4.11 caerEarEventPacketGetEventConst()

```
static caerEarEventConst caerEarEventPacketGetEventConst (
            caerEarEventPacketConst packet,
            int32_t n )  [inline], [static]
```

Get the ear (cochlea) event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid EarEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only ear (cochlea) event. NULL on error.

4.11.4.12 caerEarEventSetChannel()

```
static void caerEarEventSetChannel (
            caerEarEvent event,
            uint16_t channel )  [inline], [static]
```

Set the channel (frequency band) ID. The channels count from 0 upward, where 0 is the highest frequency channel, while higher numbers are progressively lower frequency channels. This is derived from how the actual human ear works.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.
channel	the channel (frequency band) ID.

4.11.4.13 caerEarEventSetEar()

```
static void caerEarEventSetEar (
            caerEarEvent event,
            uint8_t ear )  [inline], [static]
```

Set the numerical ID of the ear (microphone). Usually, 0 is left, 1 is right for 2 ear cochleas. For 4 ear cochleas, 0 is front left, 1 is front right, 2 is back left and 3 is back right.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.
ear	the ear (microphone) ID.

4.11.4.14 caerEarEventSetTimestamp()

```
static void caerEarEventSetTimestamp (
            caerEarEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.11.4.15 caerEarEventValidate()

```
static void caerEarEventValidate (
            caerEarEvent event,
            caerEarEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid EarEvent pointer. Cannot be NULL.
packet	the EarEventPacket pointer for the packet containing this event. Cannot be NULL.

4.11.4.16 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_ear_event { uint32_t data;int32_t timestamp;}  )
```

Ear (cochlea) event data structure definition. Contains information on events gotten from a cochlea chip: ears, channels, neurons and filters are stored. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.11.4.17 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_ear_event_packet { struct caer_event_packet_header packetHeader;struct
caer_ear_event events[];}  )
```

Ear (cochlea) event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.12 events/frame.h File Reference

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

Macros

- #define CAER_FRAME_ITERATOR_ALL_START(FRAME_PACKET)
- #define CAER_FRAME_CONST_ITERATOR_ALL_START(FRAME_PACKET)
- #define CAER_FRAME_ITERATOR_ALL_END }
- #define CAER_FRAME_ITERATOR_VALID_START(FRAME_PACKET)
- #define CAER_FRAME_CONST_ITERATOR_VALID_START(FRAME_PACKET)
- #define CAER_FRAME_ITERATOR_VALID_END }
- #define CAER_FRAME_REVERSE_ITERATOR_ALL_START(FRAME_PACKET)
- #define CAER_FRAME_CONST_REVERSE_ITERATOR_ALL_START(FRAME_PACKET)
- #define CAER_FRAME_REVERSE_ITERATOR_ALL_END }
- #define CAER_FRAME_REVERSE_ITERATOR_VALID_START(FRAME_PACKET)
- #define CAER_FRAME_CONST_REVERSE_ITERATOR_VALID_START(FRAME_PACKET)
- #define CAER_FRAME_REVERSE_ITERATOR_VALID_END }

- #define FRAME_COLOR_CHANNELS_SHIFT 1
- #define FRAME_COLOR_CHANNELS_MASK 0x00000007
- #define FRAME_COLOR_FILTER_SHIFT 4
- #define FRAME_COLOR_FILTER_MASK 0x0000000F
- #define FRAME_ROI_IDENTIFIER_SHIFT 8
- #define FRAME_ROI_IDENTIFIER_MASK 0x0000007F

Typedefs

- typedef struct caer_frame_event ∗ caerFrameEvent
- typedef const struct caer_frame_event ∗ **caerFrameEventConst**
- typedef struct caer_frame_event_packet ∗ caerFrameEventPacket
- typedef const struct caer_frame_event_packet ∗ **caerFrameEventPacketConst**

Enumerations

- enum caer_frame_event_color_channels { GRAYSCALE = 1, RGB = 3, RGBA = 4 }
- enum caer_frame_event_color_filter {
  MONO = 0, RGBG = 1, GRGB = 2, GBGR = 3,
  BGRG = 4, RGBW = 5, GRWB = 6, WBGR = 7,
  BWRG = 8 }

Functions

- PACKED_STRUCT (struct caer_frame_event { uint32_t info;int32_t ts_startframe;int32_t ts_endframe;int32↩ _t ts_startexposure;int32_t ts_endexposure;int32_t lengthX;int32_t lengthY;int32_t positionX;int32_↩ t positionY;uint16_t pixels[1];})
- PACKED_STRUCT (struct caer_frame_event_packet { struct caer_event_packet_header packetHeader;})
- static caerFrameEventPacket caerFrameEventPacketAllocateNumPixels (int32_t eventCapacity, int16_↩ t eventSource, int32_t tsOverflow, int32_t maxNumPixels, int16_t maxChannelNumber)
- static caerFrameEventPacket caerFrameEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow, int32_t maxLengthX, int32_t maxLengthY, int16_t maxChannelNumber)
- static caerFrameEventPacket caerFrameEventPacketFromPacketHeader (caerEventPacketHeader header)

- static caerFrameEventPacketConst caerFrameEventPacketFromPacketHeaderConst (caerEventPacket↩ HeaderConst header)
- static caerFrameEvent caerFrameEventPacketGetEvent (caerFrameEventPacket packet, int32_t n)
- static caerFrameEventConst caerFrameEventPacketGetEventConst (caerFrameEventPacketConst packet, int32_t n)
- static int32_t caerFrameEventGetTSStartOfFrame (caerFrameEventConst event)
- static int64_t caerFrameEventGetTSStartOfFrame64 (caerFrameEventConst event, caerFrameEvent↩ PacketConst packet)
- static void caerFrameEventSetTSStartOfFrame (caerFrameEvent event, int32_t startFrame)
- static int32_t caerFrameEventGetTSEndOfFrame (caerFrameEventConst event)
- static int64_t caerFrameEventGetTSEndOfFrame64 (caerFrameEventConst event, caerFrameEventPacket↩ Const packet)
- static void caerFrameEventSetTSEndOfFrame (caerFrameEvent event, int32_t endFrame)
- static int32_t caerFrameEventGetTSStartOfExposure (caerFrameEventConst event)
- static int64_t caerFrameEventGetTSStartOfExposure64 (caerFrameEventConst event, caerFrameEvent↩ PacketConst packet)
- static void caerFrameEventSetTSStartOfExposure (caerFrameEvent event, int32_t startExposure)
- static int32_t caerFrameEventGetTSEndOfExposure (caerFrameEventConst event)
- static int64_t caerFrameEventGetTSEndOfExposure64 (caerFrameEventConst event, caerFrameEvent↩ PacketConst packet)
- static void caerFrameEventSetTSEndOfExposure (caerFrameEvent event, int32_t endExposure)
- static int32_t caerFrameEventGetExposureLength (caerFrameEventConst event)
- static int32_t caerFrameEventGetTimestamp (caerFrameEventConst event)
- static int64_t caerFrameEventGetTimestamp64 (caerFrameEventConst event, caerFrameEventPacketConst packet)
- static bool caerFrameEventIsValid (caerFrameEventConst event)
- static void caerFrameEventValidate (caerFrameEvent event, caerFrameEventPacket packet)
- static void caerFrameEventInvalidate (caerFrameEvent event, caerFrameEventPacket packet)
- static size_t caerFrameEventPacketGetPixelsSize (caerFrameEventPacketConst packet)
- static size_t caerFrameEventPacketGetPixelsMaxIndex (caerFrameEventPacketConst packet)
- static uint8_t caerFrameEventGetROIIdentifier (caerFrameEventConst event)
- static void caerFrameEventSetROIIdentifier (caerFrameEvent event, uint8_t roiIdentifier)
- static enum caer_frame_event_color_filter caerFrameEventGetColorFilter (caerFrameEventConst event)
- static void caerFrameEventSetColorFilter (caerFrameEvent event, enum caer_frame_event_color_filter colorFilter)
- static int32_t caerFrameEventGetLengthX (caerFrameEventConst event)
- static int32_t caerFrameEventGetLengthY (caerFrameEventConst event)
- static enum caer_frame_event_color_channels caerFrameEventGetChannelNumber (caerFrameEventConst event)
- static void caerFrameEventSetLengthXLengthYChannelNumber (caerFrameEvent event, int32_t lengthX, int32_t lengthY, enum caer_frame_event_color_channels channelNumber, caerFrameEventPacketConst packet)
- static size_t caerFrameEventGetPixelsMaxIndex (caerFrameEventConst event)
- static size_t caerFrameEventGetPixelsSize (caerFrameEventConst event)
- static int32_t caerFrameEventGetPositionX (caerFrameEventConst event)
- static void caerFrameEventSetPositionX (caerFrameEvent event, int32_t positionX)
- static int32_t caerFrameEventGetPositionY (caerFrameEventConst event)
- static void caerFrameEventSetPositionY (caerFrameEvent event, int32_t positionY)
- static uint16_t caerFrameEventGetPixel (caerFrameEventConst event, int32_t xAddress, int32_t yAddress)
- static void caerFrameEventSetPixel (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint16_t pixelValue)
- static uint16_t caerFrameEventGetPixelForChannel (caerFrameEventConst event, int32_t xAddress, int32_t yAddress, uint8_t channel)
- static void caerFrameEventSetPixelForChannel (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint8_t channel, uint16_t pixelValue)

- static uint16_t caerFrameEventGetPixelUnsafe (caerFrameEventConst event, int32_t xAddress, int32_t y↩ Address)
- static void caerFrameEventSetPixelUnsafe (caerFrameEvent event, int32_t xAddress, int32_t yAddress, uint16_t pixelValue)
- static uint16_t caerFrameEventGetPixelForChannelUnsafe (caerFrameEventConst event, int32_t xAddress, int32_t yAddress, uint8_t channel)
- static void caerFrameEventSetPixelForChannelUnsafe (caerFrameEvent event, int32_t xAddress, int32_t y↩ Address, uint8_t channel, uint16_t pixelValue)
- static uint16_t ∗ caerFrameEventGetPixelArrayUnsafe (caerFrameEvent event)
- static const uint16_t ∗ caerFrameEventGetPixelArrayUnsafeConst (caerFrameEventConst event)

4.12.1 Detailed Description

Frame Events format definition and handling functions. This event type encodes intensity frames, like you would get from a normal APS camera. It supports multiple channels for color, color filter information, as well as multiple Regions of Interest (ROI). The (0, 0) pixel is in the upper left corner of the screen, like in OpenCV/computer graphics. The pixel array is laid out row by row (increasing X axis), going from top to bottom (increasing Y axis). To copy a frame event, the usual assignment operator = cannot be used. Please use caerGenericEventCopy() to copy frame events!

4.12.2 Macro Definition Documentation

4.12.2.1 CAER_FRAME_CONST_ITERATOR_ALL_START

```
#define CAER_FRAME_CONST_ITERATOR_ALL_START(
                FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = 0;                                                    \
         caerFrameIteratorCounter < caerEventPacketHeaderGetEventNumber(
      &(FRAME_PACKET)->packetHeader); \
         caerFrameIteratorCounter++) {                                                             \
        caerFrameEventConst caerFrameIteratorElement                                               \
            = caerFrameEventPacketGetEventConst(FRAME_PACKET,
      caerFrameIteratorCounter);
```

Const-Iterator over all frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerFrameIteratorElement' variable of type caerFrameEvent↩ Const.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.2 CAER_FRAME_CONST_ITERATOR_VALID_START

```
#define CAER_FRAME_CONST_ITERATOR_VALID_START(
                FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = 0;                                                  \
         caerFrameIteratorCounter < caerEventPacketHeaderGetEventNumber(
      &(FRAME_PACKET)->packetHeader); \
         caerFrameIteratorCounter++) {                                                          \
        caerFrameEventConst caerFrameIteratorElement                                            \
            = caerFrameEventPacketGetEventConst(FRAME_PACKET,
      caerFrameIteratorCounter);                    \
        if (!caerFrameEventIsValid(caerFrameIteratorElement)) {
                           \
            continue;                                                                           \
        }
```

Const-Iterator over only the valid frame events in a packet. Returns the current index in the 'caerFrameIterator↩ Counter' variable of type 'int32_t' and the current read-only event in the 'caerFrameIteratorElement' variable of type caerFrameEventConst.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.3 CAER_FRAME_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_FRAME_CONST_REVERSE_ITERATOR_ALL_START(
                FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(FRAME_PACKET)->packetHeader) - 1; \
         caerFrameIteratorCounter >= 0; caerFrameIteratorCounter--) {
               \
        caerFrameEventConst caerFrameIteratorElement
               \
            = caerFrameEventPacketGetEventConst(FRAME_PACKET,
      caerFrameIteratorCounter);
```

Const-Reverse iterator over all frame events in a packet. Returns the current index in the 'caerFrameIterator↩ Counter' variable of type 'int32_t' and the current read-only event in the 'caerFrameIteratorElement' variable of type caerFrameEventConst.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.4 CAER_FRAME_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_FRAME_CONST_REVERSE_ITERATOR_VALID_START(
                FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(FRAME_PACKET)->packetHeader) - 1; \
         caerFrameIteratorCounter >= 0; caerFrameIteratorCounter--) {
               \
        caerFrameEventConst caerFrameIteratorElement
               \
            = caerFrameEventPacketGetEventConst(FRAME_PACKET,
      caerFrameIteratorCounter);                        \
        if (!caerFrameEventIsValid(caerFrameIteratorElement)) {
                                       \
            continue;
               \
        }
```

Const-Reverse iterator over only the valid frame events in a packet. Returns the current index in the 'caerFrame↩ IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerFrameIteratorElement' variable of type caerFrameEventConst.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.5 CAER_FRAME_ITERATOR_ALL_END

```
#define CAER_FRAME_ITERATOR_ALL_END }
```

Iterator close statement.

4.12.2.6 CAER_FRAME_ITERATOR_ALL_START

```
#define CAER_FRAME_ITERATOR_ALL_START(
                FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = 0;                                           \
         caerFrameIteratorCounter < caerEventPacketHeaderGetEventNumber(
      &(FRAME_PACKET)->packetHeader); \
         caerFrameIteratorCounter++) {                                                   \
        caerFrameEvent caerFrameIteratorElement = caerFrameEventPacketGetEvent(
      FRAME_PACKET, caerFrameIteratorCounter);
```

Iterator over all frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current event in the 'caerFrameIteratorElement' variable of type caerFrameEvent.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.7 CAER_FRAME_ITERATOR_VALID_END

```
#define CAER_FRAME_ITERATOR_VALID_END }
```

Iterator close statement.

4.12.2.8 CAER_FRAME_ITERATOR_VALID_START

```
#define CAER_FRAME_ITERATOR_VALID_START(
                FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = 0;                                           \
         caerFrameIteratorCounter < caerEventPacketHeaderGetEventNumber(
      &(FRAME_PACKET)->packetHeader); \
         caerFrameIteratorCounter++) {                                                   \
        caerFrameEvent caerFrameIteratorElement                                          \
            = caerFrameEventPacketGetEvent(FRAME_PACKET,
      caerFrameIteratorCounter);                     \
        if (!caerFrameEventIsValid(caerFrameIteratorElement)) {
                                \
            continue;                                                                    \
        }
```

Iterator over only the valid frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current event in the 'caerFrameIteratorElement' variable of type caerFrameEvent.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.9 CAER_FRAME_REVERSE_ITERATOR_ALL_END

```
#define CAER_FRAME_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.12.2.10 CAER_FRAME_REVERSE_ITERATOR_ALL_START

```
#define CAER_FRAME_REVERSE_ITERATOR_ALL_START(
            FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(FRAME_PACKET)->packetHeader) - 1; \
         caerFrameIteratorCounter >= 0; caerFrameIteratorCounter--) {
               \
        caerFrameEvent caerFrameIteratorElement = caerFrameEventPacketGetEvent(
      FRAME_PACKET, caerFrameIteratorCounter);
```

Reverse iterator over all frame events in a packet. Returns the current index in the 'caerFrameIteratorCounter' variable of type 'int32_t' and the current event in the 'caerFrameIteratorElement' variable of type caerFrameEvent.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.11 CAER_FRAME_REVERSE_ITERATOR_VALID_END

```
#define CAER_FRAME_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.12.2.12 CAER_FRAME_REVERSE_ITERATOR_VALID_START

```
#define CAER_FRAME_REVERSE_ITERATOR_VALID_START(
            FRAME_PACKET )
```

Value:

```
for (int32_t caerFrameIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(FRAME_PACKET)->packetHeader) - 1; \
         caerFrameIteratorCounter >= 0; caerFrameIteratorCounter--) {
               \
        caerFrameEvent caerFrameIteratorElement
               \
            = caerFrameEventPacketGetEvent(FRAME_PACKET,
      caerFrameIteratorCounter);                             \
        if (!caerFrameEventIsValid(caerFrameIteratorElement)) {
                                     \
            continue;
               \
        }
```

Reverse iterator over only the valid frame events in a packet. Returns the current index in the 'caerFrameIterator←Counter' variable of type 'int32_t' and the current event in the 'caerFrameIteratorElement' variable of type caer←FrameEvent.

FRAME_PACKET: a valid FrameEventPacket pointer. Cannot be NULL.

4.12.2.13 FRAME_COLOR_CHANNELS_MASK

```
#define FRAME_COLOR_CHANNELS_MASK 0x00000007
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame←_event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see 'common.h' for more details.

4.12.2.14 FRAME_COLOR_CHANNELS_SHIFT

```
#define FRAME_COLOR_CHANNELS_SHIFT 1
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame←_event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see 'common.h' for more details.

4.12.2.15 FRAME_COLOR_FILTER_MASK

```
#define FRAME_COLOR_FILTER_MASK 0x0000000F
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame←_event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see 'common.h' for more details.

4.12.2.16 FRAME_COLOR_FILTER_SHIFT

```
#define FRAME_COLOR_FILTER_SHIFT 4
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame←_event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see 'common.h' for more details.

4.12.2.17 FRAME_ROI_IDENTIFIER_MASK

```
#define FRAME_ROI_IDENTIFIER_MASK 0x0000007F
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame←_event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see 'common.h' for more details.

4.12.2.18 FRAME_ROI_IDENTIFIER_SHIFT

```
#define FRAME_ROI_IDENTIFIER_SHIFT 8
```

Shift and mask values for the color channels number, the color filter arrangement and the ROI identifier contained in the 'info' field of the frame event. Multiple channels (RGB for example) are possible, see the 'enum caer_frame←
_event_color_channels'. To understand the original color filter arrangement to interpolate color images, see the 'enum caer_frame_event_color_filter'. Also, up to 128 different Regions of Interest (ROI) can be tracked. Bit 0 is the valid mark, see 'common.h' for more details.

4.12.3 Typedef Documentation

4.12.3.1 caerFrameEvent

```
typedef struct caer_frame_event* caerFrameEvent
```

Type for pointer to frame event data structure.

4.12.3.2 caerFrameEventPacket

```
typedef struct caer_frame_event_packet* caerFrameEventPacket
```

Type for pointer to frame event packet data structure.

4.12.4 Enumeration Type Documentation

4.12.4.1 caer_frame_event_color_channels

```
enum caer_frame_event_color_channels
```

List of all frame event color channel identifiers. Used to interpret the frame event color channel field.

Enumerator

GRAYSCALE	Grayscale, one channel only.
RGB	Red Green Blue, 3 color channels.
RGBA	Red Green Blue Alpha, 3 color channels plus transparency.

4.12.4.2 caer_frame_event_color_filter

```
enum caer_frame_event_color_filter
```

List of all frame event color filter identifiers. Used to interpret the frame event color filter field.

Enumerator

MONO	No color filter present, all light passes.
RGBG	Standard Bayer color filter, 1 red 2 green 1 blue. Variation 1.
GRGB	Standard Bayer color filter, 1 red 2 green 1 blue. Variation 2.
GBGR	Standard Bayer color filter, 1 red 2 green 1 blue. Variation 3.
BGRG	Standard Bayer color filter, 1 red 2 green 1 blue. Variation 4.
RGBW	Modified Bayer color filter, with white (pass all light) instead of extra green. Variation 1.
GRWB	Modified Bayer color filter, with white (pass all light) instead of extra green. Variation 2.
WBGR	Modified Bayer color filter, with white (pass all light) instead of extra green. Variation 3.
BWRG	Modified Bayer color filter, with white (pass all light) instead of extra green. Variation 4.

4.12.5 Function Documentation

4.12.5.1 caerFrameEventGetChannelNumber()

```
static enum caer_frame_event_color_channels caerFrameEventGetChannelNumber (
             caerFrameEventConst event )  [inline], [static]
```

Get the actual color channels number for the current frame. This can be used to store RGB frames for example.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

frame color channels number.

4.12.5.2 caerFrameEventGetColorFilter()

```
static enum caer_frame_event_color_filter caerFrameEventGetColorFilter (
             caerFrameEventConst event )  [inline], [static]
```

Get the identifier for the color filter used by the sensor. Useful for interpolating color images.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

color filter identifier.

4.12.5.3 caerFrameEventGetExposureLength()

```
static int32_t caerFrameEventGetExposureLength (
            caerFrameEventConst event )  [inline], [static]
```

The total length, in microseconds, of the frame exposure time.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

the exposure time in microseconds.

4.12.5.4 caerFrameEventGetLengthX()

```
static int32_t caerFrameEventGetLengthX (
            caerFrameEventConst event )  [inline], [static]
```

Get the actual X axis length for the current frame.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

frame X axis length.

4.12.5.5 caerFrameEventGetLengthY()

```
static int32_t caerFrameEventGetLengthY (
            caerFrameEventConst event )  [inline], [static]
```

Get the actual Y axis length for the current frame.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

frame Y axis length.

4.12.5.6 caerFrameEventGetPixel()

```
static uint16_t caerFrameEventGetPixel (
            caerFrameEventConst event,
            int32_t xAddress,
            int32_t yAddress )  [inline], [static]
```

Get the pixel value at the specified (X, Y) address. (X, Y) are checked against the actual possible values for this frame. Different channels are not taken into account! The (0, 0) pixel is in the upper left corner, like in OpenC↩V/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (checked).
yAddress	Y address value (checked).

Returns

pixel value (normalized to 16 bit depth).

4.12.5.7 caerFrameEventGetPixelArrayUnsafe()

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

Get a direct pointer to the underlying pixels array. This can be used to both get and set values. No checks at all are performed at any point, nor any conversions, use this at your own risk! Remember that the 16 bit pixel values are in little-endian! The pixel array is laid out row by row (increasing X axis), going from top to bottom (increasing Y axis).

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

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

4.12.5.8 caerFrameEventGetPixelArrayUnsafeConst()

```
static const uint16_t* caerFrameEventGetPixelArrayUnsafeConst (
            caerFrameEventConst event )  [inline], [static]
```

Get a direct read-only pointer to the underlying pixels array. This can be used to only get values. No checks at all are performed at any point, nor any conversions, use this at your own risk! Remember that the 16 bit pixel values are in little-endian! The pixel array is laid out row by row (increasing X axis), going from top to bottom (increasing Y axis).

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

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

4.12.5.9 caerFrameEventGetPixelForChannel()

```
static uint16_t caerFrameEventGetPixelForChannel (
            caerFrameEventConst event,
            int32_t xAddress,
            int32_t yAddress,
            uint8_t channel )  [inline], [static]
```

Get the pixel value at the specified (X, Y) address, taking into account the specified channel. (X, Y) and the channel number are checked against the actual possible values for this frame. The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (checked).
yAddress	Y address value (checked).
channel	the channel number (checked).

Returns

pixel value (normalized to 16 bit depth).

4.12.5.10 caerFrameEventGetPixelForChannelUnsafe()

```
static uint16_t caerFrameEventGetPixelForChannelUnsafe (
            caerFrameEventConst event,
            int32_t xAddress,
            int32_t yAddress,
            uint8_t channel )  [inline], [static]
```

Get the pixel value at the specified (X, Y) address, taking into account the specified channel. No checks on (X, Y) and the channel number are performed! The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (unchecked).
yAddress	Y address value (unchecked).
channel	the channel number (unchecked).

Returns

pixel value (normalized to 16 bit depth).

4.12.5.11 caerFrameEventGetPixelsMaxIndex()

```
static size_t caerFrameEventGetPixelsMaxIndex (
            caerFrameEventConst event )  [inline], [static]
```

Get the maximum valid index into the pixel array, at which you can still get valid pixels.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

maximum valid pixels array index.

4.12.5.12 caerFrameEventGetPixelsSize()

```
static size_t caerFrameEventGetPixelsSize (
            caerFrameEventConst event )  [inline], [static]
```

Get the maximum size of the pixels array in bytes, in which you can still get valid pixels.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

maximum valid pixels array size in bytes.

4.12.5.13 caerFrameEventGetPixelUnsafe()

```
static uint16_t caerFrameEventGetPixelUnsafe (
            caerFrameEventConst event,
            int32_t xAddress,
            int32_t yAddress )  [inline], [static]
```

Get the pixel value at the specified (X, Y) address. No checks on (X, Y) are performed! The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (unchecked).
yAddress	Y address value (unchecked).

Returns

pixel value (normalized to 16 bit depth).

4.12.5.14 caerFrameEventGetPositionX()

```
static int32_t caerFrameEventGetPositionX (
            caerFrameEventConst event )  [inline], [static]
```

Get the X axis position offset. This is used to place partial frames, like the ones gotten from ROI readouts, in the visual space.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

X axis position offset.

4.12.5.15 caerFrameEventGetPositionY()

```
static int32_t caerFrameEventGetPositionY (
            caerFrameEventConst event )  [inline], [static]
```

Get the Y axis position offset. This is used to place partial frames, like the ones gotten from ROI readouts, in the visual space.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

Y axis position offset.

4.12.5.16 caerFrameEventGetROIIdentifier()

```
static uint8_t caerFrameEventGetROIIdentifier (
            caerFrameEventConst event )  [inline], [static]
```

Get the numerical identifier for the Region of Interest (ROI) region, to distinguish between multiple of them.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

numerical ROI identifier.

4.12.5.17 caerFrameEventGetTimestamp()

```
static int32_t caerFrameEventGetTimestamp (
            caerFrameEventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. This is a median of the exposure timestamps. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WR↩AP' 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

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.12.5.18 caerFrameEventGetTimestamp64()

```
static int64_t caerFrameEventGetTimestamp64 (
            caerFrameEventConst event,
            caerFrameEventPacketConst packet ) [inline], [static]
```

Get the 64bit event timestamp, in microseconds. This is a median of the exposure timestamps. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.12.5.19 caerFrameEventGetTSEndOfExposure()

```
static int32_t caerFrameEventGetTSEndOfExposure (
            caerFrameEventConst event ) [inline], [static]
```

Get the 32bit end of exposure timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond end of exposure timestamp.

4.12.5.20 caerFrameEventGetTSEndOfExposure64()

```
static int64_t caerFrameEventGetTSEndOfExposure64 (
            caerFrameEventConst event,
            caerFrameEventPacketConst packet ) [inline], [static]
```

Get the 64bit end of exposure timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond end of exposure timestamp.

4.12.5.21 caerFrameEventGetTSEndOfFrame()

```
static int32_t caerFrameEventGetTSEndOfFrame (
             caerFrameEventConst event )  [inline], [static]
```

Get the 32bit end of frame capture timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond end of frame timestamp.

4.12.5.22 caerFrameEventGetTSEndOfFrame64()

```
static int64_t caerFrameEventGetTSEndOfFrame64 (
             caerFrameEventConst event,
             caerFrameEventPacketConst packet )  [inline], [static]
```

Get the 64bit end of frame capture timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond end of frame timestamp.

4.12.5.23 caerFrameEventGetTSStartOfExposure()

```
static int32_t caerFrameEventGetTSStartOfExposure (
            caerFrameEventConst event )  [inline], [static]
```

Get the 32bit start of exposure timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond start of exposure timestamp.

4.12.5.24 caerFrameEventGetTSStartOfExposure64()

```
static int64_t caerFrameEventGetTSStartOfExposure64 (
            caerFrameEventConst event,
            caerFrameEventPacketConst packet )  [inline], [static]
```

Get the 64bit start of exposure timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond start of exposure timestamp.

4.12.5.25 caerFrameEventGetTSStartOfFrame()

```
static int32_t caerFrameEventGetTSStartOfFrame (
            caerFrameEventConst event )  [inline], [static]
```

Get the 32bit start of frame capture timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond start of frame timestamp.

4.12.5.26 caerFrameEventGetTSStartOfFrame64()

```
static int64_t caerFrameEventGetTSStartOfFrame64 (
            caerFrameEventConst event,
            caerFrameEventPacketConst packet ) [inline], [static]
```

Get the 64bit start of frame capture timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond start of frame timestamp.

4.12.5.27 caerFrameEventInvalidate()

```
static void caerFrameEventInvalidate (
            caerFrameEvent event,
            caerFrameEventPacket packet ) [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

4.12.5.28 caerFrameEventIsValid()

```
static bool caerFrameEventIsValid (
             caerFrameEventConst event )  [inline], [static]
```

Check if this frame event is valid.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.12.5.29 caerFrameEventPacketAllocate()

```
static caerFrameEventPacket caerFrameEventPacketAllocate (
             int32_t eventCapacity,
             int16_t eventSource,
             int32_t tsOverflow,
             int32_t maxLengthX,
             int32_t maxLengthY,
             int16_t maxChannelNumber )  [inline], [static]
```

Allocate a new frame events packet. Use free() to reclaim this memory. The frame events allocate memory for a maximum sized pixels array, depending on the parameters passed to this function, so that every event occupies the same amount of memory (constant size). The actual frames inside of it might be smaller than that, for example when using ROI, and their actual size is stored inside the frame event and should always be queried from there. The unused part of a pixels array is guaranteed to be zeros.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.
maxLengthX	the maximum expected X axis size for frames in this packet.
maxLengthY	the maximum expected Y axis size for frames in this packet.
maxChannelNumber	the maximum expected number of channels for frames in this packet.

Returns

a valid FrameEventPacket handle or NULL on error.

4.12.5.30 caerFrameEventPacketAllocateNumPixels()

```
static caerFrameEventPacket caerFrameEventPacketAllocateNumPixels (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow,
            int32_t maxNumPixels,
            int16_t maxChannelNumber )  [inline], [static]
```

Allocate a new frame events packet, passing the total number of maximum pixels instead of the maximum X/Y dimensions expected. 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

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.
maxNumPixels	the maximum number of pixels that can be held by a frame event.
maxChannelNumber	the maximum expected number of channels for frames in this packet.

Returns

a valid FrameEventPacket handle or NULL on error.

4.12.5.31 caerFrameEventPacketFromPacketHeader()

```
static caerFrameEventPacket caerFrameEventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into a Frame event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.12.5.32 caerFrameEventPacketFromPacketHeaderConst()

```
static caerFrameEventPacketConst caerFrameEventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only Frame event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.12.5.33 caerFrameEventPacketGetEvent()

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

Get the frame event at the given index from the event packet. To copy a frame event, the usual assignment operator = cannot be used. Please use caerGenericEventCopy() to copy frame events!

Parameters

packet	a valid FrameEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested frame event. NULL on error.

4.12.5.34 caerFrameEventPacketGetEventConst()

```
static caerFrameEventConst caerFrameEventPacketGetEventConst (
            caerFrameEventPacketConst packet,
            int32_t n )  [inline], [static]
```

Get the frame event at the given index from the event packet. This is a read-only event, do not change its contents in any way! To copy a frame event, the usual assignment operator = cannot be used. Please use caerGenericEventCopy() to copy frame events!

Parameters

packet	a valid FrameEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only frame event. NULL on error.

4.12.5.35 caerFrameEventPacketGetPixelsMaxIndex()

```
static size_t caerFrameEventPacketGetPixelsMaxIndex (
            caerFrameEventPacketConst packet )  [inline], [static]
```

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

Parameters

packet	a valid FrameEventPacket pointer. Cannot be NULL.

Returns

maximum pixels array index.

4.12.5.36 caerFrameEventPacketGetPixelsSize()

```
static size_t caerFrameEventPacketGetPixelsSize (
            caerFrameEventPacketConst packet )  [inline], [static]
```

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

Parameters

packet	a valid FrameEventPacket pointer. Cannot be NULL.

Returns

maximum pixels array size in bytes.

4.12.5.37 caerFrameEventSetColorFilter()

```
static void caerFrameEventSetColorFilter (
            caerFrameEvent event,
            enum caer_frame_event_color_filter colorFilter )  [inline], [static]
```

Set the identifier for the color filter used by the sensor. Useful for interpolating color images.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
colorFilter	color filter identifier.

4.12.5.38 caerFrameEventSetLengthXLengthYChannelNumber()

```
static void caerFrameEventSetLengthXLengthYChannelNumber (
            caerFrameEvent event,
            int32_t lengthX,
            int32_t lengthY,
            enum caer_frame_event_color_channels channelNumber,
            caerFrameEventPacketConst packet )  [inline], [static]
```

Set the X and Y axes length and the color channels number for a frame, while taking into account the maximum amount of memory available for the pixel array, as allocated in 'caerFrameEventPacketAllocate()'.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
lengthX	the frame's X axis length.
lengthY	the frame's Y axis length.
channelNumber	the number of color channels for this frame.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

4.12.5.39 caerFrameEventSetPixel()

```
static void caerFrameEventSetPixel (
            caerFrameEvent event,
            int32_t xAddress,
            int32_t yAddress,
            uint16_t pixelValue )  [inline], [static]
```

Set the pixel value at the specified (X, Y) address. (X, Y) are checked against the actual possible values for this frame. Different channels are not taken into account! The (0, 0) pixel is in the upper left corner, like in OpenC↩
V/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (checked).
yAddress	Y address value (checked).
pixelValue	pixel value (normalized to 16 bit depth).

4.12.5.40 caerFrameEventSetPixelForChannel()

```
static void caerFrameEventSetPixelForChannel (
            caerFrameEvent event,
            int32_t xAddress,
            int32_t yAddress,
            uint8_t channel,
            uint16_t pixelValue ) [inline], [static]
```

Set the pixel value at the specified (X, Y) address, taking into account the specified channel. (X, Y) and the channel number are checked against the actual possible values for this frame. The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (checked).
yAddress	Y address value (checked).
channel	the channel number (checked).
pixelValue	pixel value (normalized to 16 bit depth).

4.12.5.41 caerFrameEventSetPixelForChannelUnsafe()

```
static void caerFrameEventSetPixelForChannelUnsafe (
            caerFrameEvent event,
            int32_t xAddress,
            int32_t yAddress,
            uint8_t channel,
            uint16_t pixelValue ) [inline], [static]
```

Set the pixel value at the specified (X, Y) address, taking into account the specified channel. No checks on (X, Y) and the channel number are performed! The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (unchecked).
yAddress	Y address value (unchecked).
channel	the channel number (unchecked).
pixelValue	pixel value (normalized to 16 bit depth).

4.12.5.42 caerFrameEventSetPixelUnsafe()

```
static void caerFrameEventSetPixelUnsafe (
             caerFrameEvent event,
             int32_t xAddress,
             int32_t yAddress,
             uint16_t pixelValue )  [inline], [static]
```

Set the pixel value at the specified (X, Y) address. No checks on (X, Y) are performed! The (0, 0) pixel is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
xAddress	X address value (unchecked).
yAddress	Y address value (unchecked).
pixelValue	pixel value (normalized to 16 bit depth).

4.12.5.43 caerFrameEventSetPositionX()

```
static void caerFrameEventSetPositionX (
             caerFrameEvent event,
             int32_t positionX )  [inline], [static]
```

Set the X axis position offset. This is used to place partial frames, like the ones gotten from ROI readouts, in the visual space.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
positionX	X axis position offset.

4.12.5.44 caerFrameEventSetPositionY()

```
static void caerFrameEventSetPositionY (
             caerFrameEvent event,
             int32_t positionY )  [inline], [static]
```

Set the Y axis position offset. This is used to place partial frames, like the ones gotten from ROI readouts, in the visual space.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
positionY	Y axis position offset.

4.12.5.45 caerFrameEventSetROIIdentifier()

```
static void caerFrameEventSetROIIdentifier (
            caerFrameEvent event,
            uint8_t roiIdentifier )  [inline], [static]
```

Set the numerical identifier for the Region of Interest (ROI) region, to distinguish between multiple of them.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
roiIdentifier	numerical ROI identifier.

4.12.5.46 caerFrameEventSetTSEndOfExposure()

```
static void caerFrameEventSetTSEndOfExposure (
            caerFrameEvent event,
            int32_t endExposure )  [inline], [static]
```

Set the 32bit end of exposure timestamp, the value has to be in microseconds.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
endExposure	a positive 32bit microsecond timestamp.

4.12.5.47 caerFrameEventSetTSEndOfFrame()

```
static void caerFrameEventSetTSEndOfFrame (
            caerFrameEvent event,
            int32_t endFrame )  [inline], [static]
```

Set the 32bit end of frame capture timestamp, the value has to be in microseconds.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
endFrame	a positive 32bit microsecond timestamp.

4.12.5.48 caerFrameEventSetTSStartOfExposure()

```
static void caerFrameEventSetTSStartOfExposure (
            caerFrameEvent event,
            int32_t startExposure )  [inline], [static]
```

Set the 32bit start of exposure timestamp, the value has to be in microseconds.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
startExposure	a positive 32bit microsecond timestamp.

4.12.5.49 caerFrameEventSetTSStartOfFrame()

```
static void caerFrameEventSetTSStartOfFrame (
            caerFrameEvent event,
            int32_t startFrame )  [inline], [static]
```

Set the 32bit start of frame capture timestamp, the value has to be in microseconds.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
startFrame	a positive 32bit microsecond timestamp.

4.12.5.50 caerFrameEventValidate()

```
static void caerFrameEventValidate (
            caerFrameEvent event,
            caerFrameEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid FrameEvent pointer. Cannot be NULL.
packet	the FrameEventPacket pointer for the packet containing this event. Cannot be NULL.

4.12.5.51 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
             struct caer_frame_event { uint32_t info;int32_t ts_startframe;int32_t ts_endframe;int32↩
_t ts_startexposure;int32_t ts_endexposure;int32_t lengthX;int32_t lengthY;int32_t positionX;int32↩
_t positionY;uint16_t pixels[1];}  )
```

Frame event data structure definition. This contains the actual information on the frame (ROI, color channels, color filter), several timestamps to signal start and end of capture and of exposure, as well as the actual pixels, in a 16 bit normalized format. The (0, 0) address is in the upper left corner, like in OpenCV/computer graphics. The pixel array is laid out row by row (increasing X axis), going from top to bottom (increasing Y axis). Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java. To copy a frame event, the usual assignment operator = cannot be used. Please use caerGenericEventCopy() to copy frame events!

4.12.5.52 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
             struct caer_frame_event_packet { struct caer_event_packet_header packetHeader;}
)
```

Frame event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block. Direct access to the events array is not possible for Frame events. To calculate position offsets, use the 'eventSize' field in the packet header.

4.13 events/imu6.h File Reference

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

Macros

- #define CAER_IMU6_ITERATOR_ALL_START(IMU6_PACKET)
- #define CAER_IMU6_CONST_ITERATOR_ALL_START(IMU6_PACKET)
- #define CAER_IMU6_ITERATOR_ALL_END }
- #define CAER_IMU6_ITERATOR_VALID_START(IMU6_PACKET)
- #define CAER_IMU6_CONST_ITERATOR_VALID_START(IMU6_PACKET)
- #define CAER_IMU6_ITERATOR_VALID_END }
- #define CAER_IMU6_REVERSE_ITERATOR_ALL_START(IMU6_PACKET)
- #define CAER_IMU6_CONST_REVERSE_ITERATOR_ALL_START(IMU6_PACKET)
- #define CAER_IMU6_REVERSE_ITERATOR_ALL_END }
- #define CAER_IMU6_REVERSE_ITERATOR_VALID_START(IMU6_PACKET)
- #define CAER_IMU6_CONST_REVERSE_ITERATOR_VALID_START(IMU6_PACKET)
- #define CAER_IMU6_REVERSE_ITERATOR_VALID_END }

Typedefs

- typedef struct caer_imu6_event ∗ caerIMU6Event
- typedef const struct caer_imu6_event ∗ **caerIMU6EventConst**
- typedef struct caer_imu6_event_packet ∗ caerIMU6EventPacket
- typedef const struct caer_imu6_event_packet ∗ **caerIMU6EventPacketConst**

Functions

- PACKED_STRUCT (struct caer_imu6_event { uint32_t info;int32_t timestamp;float accel_x;float accel_y;float accel_z;float gyro_x;float gyro_y;float gyro_z;float temp;})
- PACKED_STRUCT (struct caer_imu6_event_packet { struct caer_event_packet_header packetHeader;struct caer_imu6_event events[ ];})
- static caerIMU6EventPacket caerIMU6EventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerIMU6EventPacket caerIMU6EventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerIMU6EventPacketConst caerIMU6EventPacketFromPacketHeaderConst (caerEventPacket←HeaderConst header)
- static caerIMU6Event caerIMU6EventPacketGetEvent (caerIMU6EventPacket packet, int32_t n)
- static caerIMU6EventConst caerIMU6EventPacketGetEventConst (caerIMU6EventPacketConst packet, int32_t n)
- static int32_t caerIMU6EventGetTimestamp (caerIMU6EventConst event)
- static int64_t caerIMU6EventGetTimestamp64 (caerIMU6EventConst event, caerIMU6EventPacketConst packet)
- static void caerIMU6EventSetTimestamp (caerIMU6Event event, int32_t timestamp)
- static bool caerIMU6EventIsValid (caerIMU6EventConst event)
- static void caerIMU6EventValidate (caerIMU6Event event, caerIMU6EventPacket packet)
- static void caerIMU6EventInvalidate (caerIMU6Event event, caerIMU6EventPacket packet)
- static float caerIMU6EventGetAccelX (caerIMU6EventConst event)
- static void caerIMU6EventSetAccelX (caerIMU6Event event, float accelX)
- static float caerIMU6EventGetAccelY (caerIMU6EventConst event)
- static void caerIMU6EventSetAccelY (caerIMU6Event event, float accelY)
- static float caerIMU6EventGetAccelZ (caerIMU6EventConst event)
- static void caerIMU6EventSetAccelZ (caerIMU6Event event, float accelZ)
- static float caerIMU6EventGetGyroX (caerIMU6EventConst event)
- static void caerIMU6EventSetGyroX (caerIMU6Event event, float gyroX)
- static float caerIMU6EventGetGyroY (caerIMU6EventConst event)
- static void caerIMU6EventSetGyroY (caerIMU6Event event, float gyroY)
- static float caerIMU6EventGetGyroZ (caerIMU6EventConst event)
- static void caerIMU6EventSetGyroZ (caerIMU6Event event, float gyroZ)
- static float caerIMU6EventGetTemp (caerIMU6EventConst event)
- static void caerIMU6EventSetTemp (caerIMU6Event event, float temp)

4.13.1 Detailed Description

IMU6 (6 axes) Events format definition and handling functions. This contains data coming from the Inertial Measurement Unit chip, with the 3-axes accelerometer and 3-axes gyroscope. Temperature is also included.

4.13.2 Macro Definition Documentation

4.13.2.1 CAER_IMU6_CONST_ITERATOR_ALL_START

```
#define CAER_IMU6_CONST_ITERATOR_ALL_START(
                IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = 0;                                                  \
         caerIMU6IteratorCounter < caerEventPacketHeaderGetEventNumber(&
      (IMU6_PACKET)->packetHeader); \
         caerIMU6IteratorCounter++) {                                                          \
        caerIMU6EventConst caerIMU6IteratorElement                                             \
            = caerIMU6EventPacketGetEventConst(IMU6_PACKET,
      caerIMU6IteratorCounter);
```

Const-Iterator over all IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU6IteratorElement' variable of type caerIMU6Event←Const.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.2 CAER_IMU6_CONST_ITERATOR_VALID_START

```
#define CAER_IMU6_CONST_ITERATOR_VALID_START(
                IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = 0;                                                  \
         caerIMU6IteratorCounter < caerEventPacketHeaderGetEventNumber(&
      (IMU6_PACKET)->packetHeader); \
         caerIMU6IteratorCounter++) {                                                          \
        caerIMU6EventConst caerIMU6IteratorElement                                             \
            = caerIMU6EventPacketGetEventConst(IMU6_PACKET,
      caerIMU6IteratorCounter);                  \
        if (!caerIMU6EventIsValid(caerIMU6IteratorElement)) {
                     \
            continue;                                                                          \
        }
```

Const-Iterator over only the valid IMU6 events in a packet. Returns the current index in the 'caerIMU6Iterator←Counter' variable of type 'int32_t' and the current read-only event in the 'caerIMU6IteratorElement' variable of type caerIMU6EventConst.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.3 CAER_IMU6_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_IMU6_CONST_REVERSE_ITERATOR_ALL_START(
                IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(IMU6_PACKET)->packetHeader) - 1; \
         caerIMU6IteratorCounter >= 0; caerIMU6IteratorCounter--) {
             \
        caerIMU6EventConst caerIMU6IteratorElement
             \
            = caerIMU6EventPacketGetEventConst(IMU6_PACKET,
      caerIMU6IteratorCounter);
```

Const-Reverse iterator over all IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU6IteratorElement' variable of type caerIM←U6EventConst.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.4 CAER_IMU6_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_IMU6_CONST_REVERSE_ITERATOR_VALID_START(
              IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(IMU6_PACKET)->packetHeader) - 1; \
         caerIMU6IteratorCounter >= 0; caerIMU6IteratorCounter--) {
             \
        caerIMU6EventConst caerIMU6IteratorElement
             \
            = caerIMU6EventPacketGetEventConst(IMU6_PACKET,
      caerIMU6IteratorCounter);                             \
        if (!caerIMU6EventIsValid(caerIMU6IteratorElement)) {
                                 \
            continue;
             \
        }
```

Const-Reverse iterator over only the valid IMU6 events in a packet. Returns the current index in the 'caerIMU6←IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU6IteratorElement' variable of type caerIMU6EventConst.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.5 CAER_IMU6_ITERATOR_ALL_END

```
#define CAER_IMU6_ITERATOR_ALL_END }
```

Iterator close statement.

4.13.2.6 CAER_IMU6_ITERATOR_ALL_START

```
#define CAER_IMU6_ITERATOR_ALL_START(
              IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = 0;                                           \
         caerIMU6IteratorCounter < caerEventPacketHeaderGetEventNumber(&
      (IMU6_PACKET)->packetHeader); \
         caerIMU6IteratorCounter++) {                                                   \
        caerIMU6Event caerIMU6IteratorElement = caerIMU6EventPacketGetEvent(
      IMU6_PACKET, caerIMU6IteratorCounter);
```

Iterator over all IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU6IteratorElement' variable of type caerIMU6Event.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.7 CAER_IMU6_ITERATOR_VALID_END

```
#define CAER_IMU6_ITERATOR_VALID_END }
```

Iterator close statement.

4.13.2.8 CAER_IMU6_ITERATOR_VALID_START

```
#define CAER_IMU6_ITERATOR_VALID_START(
                IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = 0;
          \
         caerIMU6IteratorCounter < caerEventPacketHeaderGetEventNumber(&
      (IMU6_PACKET)->packetHeader);                \
         caerIMU6IteratorCounter++) {
              \
        caerIMU6Event caerIMU6IteratorElement = caerIMU6EventPacketGetEvent(
      IMU6_PACKET, caerIMU6IteratorCounter); \
        if (!caerIMU6EventIsValid(caerIMU6IteratorElement)) {
                                  \
            continue;
              \
        }
```

Iterator over only the valid IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU6IteratorElement' variable of type caerIMU6Event.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.9 CAER_IMU6_REVERSE_ITERATOR_ALL_END

```
#define CAER_IMU6_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.13.2.10 CAER_IMU6_REVERSE_ITERATOR_ALL_START

```
#define CAER_IMU6_REVERSE_ITERATOR_ALL_START(
                IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(IMU6_PACKET)->packetHeader) - 1; \
         caerIMU6IteratorCounter >= 0; caerIMU6IteratorCounter--) {
             \
        caerIMU6Event caerIMU6IteratorElement = caerIMU6EventPacketGetEvent(
      IMU6_PACKET, caerIMU6IteratorCounter);
```

Reverse iterator over all IMU6 events in a packet. Returns the current index in the 'caerIMU6IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU6IteratorElement' variable of type caerIMU6Event.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.2.11 CAER_IMU6_REVERSE_ITERATOR_VALID_END

```
#define CAER_IMU6_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.13.2.12 CAER_IMU6_REVERSE_ITERATOR_VALID_START

```
#define CAER_IMU6_REVERSE_ITERATOR_VALID_START(
                IMU6_PACKET )
```

Value:

```
for (int32_t caerIMU6IteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(IMU6_PACKET)->packetHeader) - 1;  \
         caerIMU6IteratorCounter >= 0; caerIMU6IteratorCounter--) {
              \
        caerIMU6Event caerIMU6IteratorElement = caerIMU6EventPacketGetEvent(
      IMU6_PACKET, caerIMU6IteratorCounter); \
        if (!caerIMU6EventIsValid(caerIMU6IteratorElement)) {
                                  \
            continue;
              \
        }
```

Reverse iterator over only the valid IMU6 events in a packet. Returns the current index in the 'caerIMU6Iterator↩ Counter' variable of type 'int32_t' and the current event in the 'caerIMU6IteratorElement' variable of type caerIM↩ U6Event.

IMU6_PACKET: a valid IMU6EventPacket pointer. Cannot be NULL.

4.13.3 Typedef Documentation

4.13.3.1 caerIMU6Event

```
typedef struct caer_imu6_event* caerIMU6Event
```

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

4.13.3.2 caerIMU6EventPacket

```
typedef struct caer_imu6_event_packet* caerIMU6EventPacket
```

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

4.13.4 Function Documentation

4.13.4.1 caerIMU6EventGetAccelX()

```
static float caerIMU6EventGetAccelX (
              caerIMU6EventConst event )  [inline], [static]
```

Get the X axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

acceleration on the X axis.

4.13.4.2 caerIMU6EventGetAccelY()

```
static float caerIMU6EventGetAccelY (
             caerIMU6EventConst event )  [inline], [static]
```

Get the Y axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

acceleration on the Y axis.

4.13.4.3 caerIMU6EventGetAccelZ()

```
static float caerIMU6EventGetAccelZ (
             caerIMU6EventConst event )  [inline], [static]
```

Get the Z axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

acceleration on the Z axis.

4.13.4.4 caerIMU6EventGetGyroX()

```
static float caerIMU6EventGetGyroX (
             caerIMU6EventConst event )  [inline], [static]
```

Get the X axis (roll) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

angular velocity on the X axis (roll).

4.13.4.5 caerIMU6EventGetGyroY()

```
static float caerIMU6EventGetGyroY (
            caerIMU6EventConst event )  [inline], [static]
```

Get the Y axis (pitch) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

angular velocity on the Y axis (pitch).

4.13.4.6 caerIMU6EventGetGyroZ()

```
static float caerIMU6EventGetGyroZ (
            caerIMU6EventConst event )  [inline], [static]
```

Get the Z axis (yaw) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

angular velocity on the Z axis (yaw).

4.13.4.7 caerIMU6EventGetTemp()

```
static float caerIMU6EventGetTemp (
            caerIMU6EventConst event )  [inline], [static]
```

Get the temperature reading. This is in ℃.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

temperature in ℃.

4.13.4.8 caerIMU6EventGetTimestamp()

```
static int32_t caerIMU6EventGetTimestamp (
            caerIMU6EventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.13.4.9 caerIMU6EventGetTimestamp64()

```
static int64_t caerIMU6EventGetTimestamp64 (
            caerIMU6EventConst event,
            caerIMU6EventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
packet	the IMU6EventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.13.4.10 caerIMU6EventInvalidate()

```
static void caerIMU6EventInvalidate (
            caerIMU6Event event,
            caerIMU6EventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
packet	the IMU6EventPacket pointer for the packet containing this event. Cannot be NULL.

4.13.4.11 caerIMU6EventIsValid()

```
static bool caerIMU6EventIsValid (
            caerIMU6EventConst event )  [inline], [static]
```

Check if this IMU 6-axes event is valid.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.13.4.12 caerIMU6EventPacketAllocate()

```
static caerIMU6EventPacket caerIMU6EventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )  [inline], [static]
```

Allocate a new IMU 6-axes events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid IMU6EventPacket handle or NULL on error.

4.13.4.13 caerIMU6EventPacketFromPacketHeader()

```
static caerIMU6EventPacket caerIMU6EventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into an IMU 6-axes event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.13.4.14 caerIMU6EventPacketFromPacketHeaderConst()

```
static caerIMU6EventPacketConst caerIMU6EventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only IMU 6-axes event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.13.4.15 caerIMU6EventPacketGetEvent()

```
static caerIMU6Event caerIMU6EventPacketGetEvent (
            caerIMU6EventPacket packet,
            int32_t n )  [inline], [static]
```

Get the IMU 6-axes event at the given index from the event packet.

Parameters

packet	a valid IMU6EventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested IMU 6-axes event. NULL on error.

4.13.4.16 caerIMU6EventPacketGetEventConst()

```
static caerIMU6EventConst caerIMU6EventPacketGetEventConst (
            caerIMU6EventPacketConst packet,
            int32_t n )  [inline], [static]
```

Get the IMU 6-axes event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid IMU6EventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only IMU 6-axes event. NULL on error.

4.13.4.17 caerIMU6EventSetAccelX()

```
static void caerIMU6EventSetAccelX (
            caerIMU6Event event,
            float accelX )  [inline], [static]
```

Set the X axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
accelX	acceleration on the X axis.

4.13.4.18 caerIMU6EventSetAccelY()

```
static void caerIMU6EventSetAccelY (
             caerIMU6Event event,
             float accelY )  [inline], [static]
```

Set the Y axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
accelY	acceleration on the Y axis.

4.13.4.19 caerIMU6EventSetAccelZ()

```
static void caerIMU6EventSetAccelZ (
             caerIMU6Event event,
             float accelZ )  [inline], [static]
```

Set the Z axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
accelZ	acceleration on the Z axis.

4.13.4.20 caerIMU6EventSetGyroX()

```
static void caerIMU6EventSetGyroX (
             caerIMU6Event event,
             float gyroX )  [inline], [static]
```

Set the X axis (roll) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
gyroX	angular velocity on the X axis (roll).

4.13.4.21 caerIMU6EventSetGyroY()

```
static void caerIMU6EventSetGyroY (
             caerIMU6Event event,
             float gyroY )  [inline], [static]
```

Set the Y axis (pitch) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
gyroY	angular velocity on the Y axis (pitch).

4.13.4.22 caerIMU6EventSetGyroZ()

```
static void caerIMU6EventSetGyroZ (
             caerIMU6Event event,
             float gyroZ )  [inline], [static]
```

Set the Z axis (yaw) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
gyroZ	angular velocity on the Z axis (yaw).

4.13.4.23 caerIMU6EventSetTemp()

```
static void caerIMU6EventSetTemp (
             caerIMU6Event event,
             float temp )  [inline], [static]
```

Set the temperature reading. This is in ℃.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
temp	temperature in ℃.

4.13.4.24 caerIMU6EventSetTimestamp()

```
static void caerIMU6EventSetTimestamp (
             caerIMU6Event event,
             int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.13.4.25 caerIMU6EventValidate()

```
static void caerIMU6EventValidate (
            caerIMU6Event event,
            caerIMU6EventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid IMU6Event pointer. Cannot be NULL.
packet	the IMU6EventPacket pointer for the packet containing this event. Cannot be NULL.

4.13.4.26 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_imu6_event { uint32_t info;int32_t timestamp;float accel_x;float
accel_y;float accel_z;float gyro_x;float gyro_y;float gyro_z;float temp;}  )
```

IMU 6-axes event data structure definition. This contains accelerometer and gyroscope headings, plus temperature. The X, Y and Z axes are referred to the camera plane. X increases to the right, Y going up and Z towards where the lens is pointing. Rotation for the gyroscope is counter-clockwise along the increasing axis, for all three axes. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.13.4.27 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_imu6_event_packet { struct caer_event_packet_header packetHeader;struct
caer_imu6_event events[];}  )
```

IMU 6-axes event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.14 events/imu9.h File Reference

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

Macros

- #define CAER_IMU9_ITERATOR_ALL_START(IMU9_PACKET)
- #define CAER_IMU9_CONST_ITERATOR_ALL_START(IMU9_PACKET)
- #define CAER_IMU9_ITERATOR_ALL_END }
- #define CAER_IMU9_ITERATOR_VALID_START(IMU9_PACKET)
- #define CAER_IMU9_CONST_ITERATOR_VALID_START(IMU9_PACKET)
- #define CAER_IMU9_ITERATOR_VALID_END }
- #define CAER_IMU9_REVERSE_ITERATOR_ALL_START(IMU9_PACKET)
- #define CAER_IMU9_CONST_REVERSE_ITERATOR_ALL_START(IMU9_PACKET)
- #define CAER_IMU9_REVERSE_ITERATOR_ALL_END }
- #define CAER_IMU9_REVERSE_ITERATOR_VALID_START(IMU9_PACKET)
- #define CAER_IMU9_CONST_REVERSE_ITERATOR_VALID_START(IMU9_PACKET)
- #define CAER_IMU9_REVERSE_ITERATOR_VALID_END }

Typedefs

- typedef struct caer_imu9_event ∗ caerIMU9Event
- typedef const struct caer_imu9_event ∗ **caerIMU9EventConst**
- typedef struct caer_imu9_event_packet ∗ caerIMU9EventPacket
- typedef const struct caer_imu9_event_packet ∗ **caerIMU9EventPacketConst**

Functions

- PACKED_STRUCT (struct caer_imu9_event { uint32_t info;int32_t timestamp;float accel_x;float accel_y;float accel_z;float gyro_x;float gyro_y;float gyro_z;float temp;float comp_x;float comp_y;float comp_z;})
- PACKED_STRUCT (struct caer_imu9_event_packet { struct caer_event_packet_header packetHeader;struct caer_imu9_event events[ ];})
- static caerIMU9EventPacket caerIMU9EventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerIMU9EventPacket caerIMU9EventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerIMU9EventPacketConst caerIMU9EventPacketFromPacketHeaderConst (caerEventPacket←HeaderConst header)
- static caerIMU9Event caerIMU9EventPacketGetEvent (caerIMU9EventPacket packet, int32_t n)
- static caerIMU9EventConst caerIMU9EventPacketGetEventConst (caerIMU9EventPacketConst packet, int32_t n)
- static int32_t caerIMU9EventGetTimestamp (caerIMU9EventConst event)
- static int64_t caerIMU9EventGetTimestamp64 (caerIMU9EventConst event, caerIMU9EventPacketConst packet)
- static void caerIMU9EventSetTimestamp (caerIMU9Event event, int32_t timestamp)
- static bool caerIMU9EventIsValid (caerIMU9EventConst event)
- static void caerIMU9EventValidate (caerIMU9Event event, caerIMU9EventPacket packet)
- static void caerIMU9EventInvalidate (caerIMU9Event event, caerIMU9EventPacket packet)
- static float caerIMU9EventGetAccelX (caerIMU9EventConst event)
- static void caerIMU9EventSetAccelX (caerIMU9Event event, float accelX)
- static float caerIMU9EventGetAccelY (caerIMU9EventConst event)
- static void caerIMU9EventSetAccelY (caerIMU9Event event, float accelY)
- static float caerIMU9EventGetAccelZ (caerIMU9EventConst event)
- static void caerIMU9EventSetAccelZ (caerIMU9Event event, float accelZ)
- static float caerIMU9EventGetGyroX (caerIMU9EventConst event)
- static void caerIMU9EventSetGyroX (caerIMU9Event event, float gyroX)
- static float caerIMU9EventGetGyroY (caerIMU9EventConst event)
- static void caerIMU9EventSetGyroY (caerIMU9Event event, float gyroY)

- static float caerIMU9EventGetGyroZ (caerIMU9EventConst event)
- static void caerIMU9EventSetGyroZ (caerIMU9Event event, float gyroZ)
- static float caerIMU9EventGetTemp (caerIMU9EventConst event)
- static void caerIMU9EventSetTemp (caerIMU9Event event, float temp)
- static float caerIMU9EventGetCompX (caerIMU9EventConst event)
- static void caerIMU9EventSetCompX (caerIMU9Event event, float compX)
- static float caerIMU9EventGetCompY (caerIMU9EventConst event)
- static void caerIMU9EventSetCompY (caerIMU9Event event, float compY)
- static float caerIMU9EventGetCompZ (caerIMU9EventConst event)
- static void caerIMU9EventSetCompZ (caerIMU9Event event, float compZ)

4.14.1 Detailed Description

IMU9 (9 axes) Events format definition and handling functions. This contains data coming from the Inertial Measurement Unit chip, with the 3-axes accelerometer and 3-axes gyroscope. Temperature is also included. Further, 3-axes from the magnetometer are included, which can be used to get a compass-like heading.

4.14.2 Macro Definition Documentation

4.14.2.1 CAER_IMU9_CONST_ITERATOR_ALL_START

```
#define CAER_IMU9_CONST_ITERATOR_ALL_START(
              IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = 0;                                               \
        caerIMU9IteratorCounter < caerEventPacketHeaderGetEventNumber(&
      (IMU9_PACKET)->packetHeader); \
        caerIMU9IteratorCounter++) {                                                        \
       caerIMU9EventConst caerIMU9IteratorElement                                           \
           = caerIMU9EventPacketGetEventConst(IMU9_PACKET,
      caerIMU9IteratorCounter);
```

Const-Iterator over all IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU9IteratorElement' variable of type caerIMU9Event↩Const.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.2 CAER_IMU9_CONST_ITERATOR_VALID_START

```
#define CAER_IMU9_CONST_ITERATOR_VALID_START(
                IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = 0;                                                   \
         caerIMU9IteratorCounter < caerEventPacketHeaderGetEventNumber(&
      (IMU9_PACKET)->packetHeader); \
         caerIMU9IteratorCounter++) {                                                          \
        caerIMU9EventConst caerIMU9IteratorElement                                             \
            = caerIMU9EventPacketGetEventConst(IMU9_PACKET,
      caerIMU9IteratorCounter);                 \
        if (!caerIMU9EventIsValid(caerIMU9IteratorElement)) {
                    \
            continue;                                                                          \
        }
```

Const-Iterator over only the valid IMU9 events in a packet. Returns the current index in the 'caerIMU9Iterator↩ Counter' variable of type 'int32_t' and the current read-only event in the 'caerIMU9IteratorElement' variable of type caerIMU9EventConst.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.3 CAER_IMU9_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_IMU9_CONST_REVERSE_ITERATOR_ALL_START(
                IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(IMU9_PACKET)->packetHeader) - 1; \
         caerIMU9IteratorCounter >= 0; caerIMU9IteratorCounter--) {
             \
        caerIMU9EventConst caerIMU9IteratorElement
             \
            = caerIMU9EventPacketGetEventConst(IMU9_PACKET,
      caerIMU9IteratorCounter);
```

Const-Reverse iterator over all IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU9IteratorElement' variable of type caerIM↩ U9EventConst.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.4 CAER_IMU9_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_IMU9_CONST_REVERSE_ITERATOR_VALID_START(
                IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(IMU9_PACKET)->packetHeader) - 1; \
         caerIMU9IteratorCounter >= 0; caerIMU9IteratorCounter--) {
             \
        caerIMU9EventConst caerIMU9IteratorElement
             \
            = caerIMU9EventPacketGetEventConst(IMU9_PACKET,
      caerIMU9IteratorCounter);                              \
        if (!caerIMU9EventIsValid(caerIMU9IteratorElement)) {
                                 \
            continue;
             \
        }
```

Const-Reverse iterator over only the valid IMU9 events in a packet. Returns the current index in the 'caerIMU9↩ IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerIMU9IteratorElement' variable of type caerIMU9EventConst.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.5 CAER_IMU9_ITERATOR_ALL_END

```
#define CAER_IMU9_ITERATOR_ALL_END }
```

Iterator close statement.

4.14.2.6 CAER_IMU9_ITERATOR_ALL_START

```
#define CAER_IMU9_ITERATOR_ALL_START(
              IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = 0;                                          \
        caerIMU9IteratorCounter < caerEventPacketHeaderGetEventNumber(&
      (IMU9_PACKET)->packetHeader); \
        caerIMU9IteratorCounter++) {                                                   \
       caerIMU9Event caerIMU9IteratorElement = caerIMU9EventPacketGetEvent(
      IMU9_PACKET, caerIMU9IteratorCounter);
```

Iterator over all IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU9IteratorElement' variable of type caerIMU9Event.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.7 CAER_IMU9_ITERATOR_VALID_END

```
#define CAER_IMU9_ITERATOR_VALID_END }
```

Iterator close statement.

4.14.2.8 CAER_IMU9_ITERATOR_VALID_START

```
#define CAER_IMU9_ITERATOR_VALID_START(
              IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = 0;
         \
        caerIMU9IteratorCounter < caerEventPacketHeaderGetEventNumber(&
      (IMU9_PACKET)->packetHeader);               \
        caerIMU9IteratorCounter++) {
             \
       caerIMU9Event caerIMU9IteratorElement = caerIMU9EventPacketGetEvent(
      IMU9_PACKET, caerIMU9IteratorCounter); \
       if (!caerIMU9EventIsValid(caerIMU9IteratorElement)) {
                                 \
           continue;
             \
       }
```

Iterator over only the valid IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU9IteratorElement' variable of type caerIMU9Event.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.9 CAER_IMU9_REVERSE_ITERATOR_ALL_END

```
#define CAER_IMU9_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.14.2.10 CAER_IMU9_REVERSE_ITERATOR_ALL_START

```
#define CAER_IMU9_REVERSE_ITERATOR_ALL_START(
                IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(IMU9_PACKET)->packetHeader) - 1; \
         caerIMU9IteratorCounter >= 0; caerIMU9IteratorCounter--) {
             \
        caerIMU9Event caerIMU9IteratorElement = caerIMU9EventPacketGetEvent(
      IMU9_PACKET, caerIMU9IteratorCounter);
```

Reverse iterator over all IMU9 events in a packet. Returns the current index in the 'caerIMU9IteratorCounter' variable of type 'int32_t' and the current event in the 'caerIMU9IteratorElement' variable of type caerIMU9Event.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.2.11 CAER_IMU9_REVERSE_ITERATOR_VALID_END

```
#define CAER_IMU9_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.14.2.12 CAER_IMU9_REVERSE_ITERATOR_VALID_START

```
#define CAER_IMU9_REVERSE_ITERATOR_VALID_START(
                IMU9_PACKET )
```

Value:

```
for (int32_t caerIMU9IteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(IMU9_PACKET)->packetHeader) - 1;  \
         caerIMU9IteratorCounter >= 0; caerIMU9IteratorCounter--) {
              \
        caerIMU9Event caerIMU9IteratorElement = caerIMU9EventPacketGetEvent(
      IMU9_PACKET, caerIMU9IteratorCounter); \
        if (!caerIMU9EventIsValid(caerIMU9IteratorElement)) {
                                  \
            continue;
              \
        }
```

Reverse iterator over only the valid IMU9 events in a packet. Returns the current index in the 'caerIMU9Iterator↵ Counter' variable of type 'int32_t' and the current event in the 'caerIMU9IteratorElement' variable of type caerIM↵ U9Event.

IMU9_PACKET: a valid IMU9EventPacket pointer. Cannot be NULL.

4.14.3 Typedef Documentation

4.14.3.1 caerIMU9Event

```
typedef struct caer_imu9_event* caerIMU9Event
```

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

4.14.3.2 caerIMU9EventPacket

```
typedef struct caer_imu9_event_packet* caerIMU9EventPacket
```

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

4.14.4 Function Documentation

4.14.4.1 caerIMU9EventGetAccelX()

```
static float caerIMU9EventGetAccelX (
              caerIMU9EventConst event )  [inline], [static]
```

Get the X axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

acceleration on the X axis.

4.14.4.2 caerIMU9EventGetAccelY()

```
static float caerIMU9EventGetAccelY (
              caerIMU9EventConst event )  [inline], [static]
```

Get the Y axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

acceleration on the Y axis.

4.14.4.3 caerIMU9EventGetAccelZ()

```
static float caerIMU9EventGetAccelZ (
            caerIMU9EventConst event )  [inline], [static]
```

Get the Z axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

acceleration on the Z axis.

4.14.4.4 caerIMU9EventGetCompX()

```
static float caerIMU9EventGetCompX (
            caerIMU9EventConst event )  [inline], [static]
```

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

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

X axis compass heading.

4.14.4.5 caerIMU9EventGetCompY()

```
static float caerIMU9EventGetCompY (
            caerIMU9EventConst event )  [inline], [static]
```

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

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

Y axis compass heading.

4.14.4.6 caerIMU9EventGetCompZ()

```
static float caerIMU9EventGetCompZ (
             caerIMU9EventConst event )  [inline], [static]
```

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

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

Z axis compass heading.

4.14.4.7 caerIMU9EventGetGyroX()

```
static float caerIMU9EventGetGyroX (
             caerIMU9EventConst event )  [inline], [static]
```

Get the X axis (roll) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

angular velocity on the X axis (roll).

4.14.4.8 caerIMU9EventGetGyroY()

```
static float caerIMU9EventGetGyroY (
             caerIMU9EventConst event )  [inline], [static]
```

Get the Y axis (pitch) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

angular velocity on the Y axis (pitch).

4.14.4.9 caerIMU9EventGetGyroZ()

```
static float caerIMU9EventGetGyroZ (
            caerIMU9EventConst event )  [inline], [static]
```

Get the Z axis (yaw) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

angular velocity on the Z axis (yaw).

4.14.4.10 caerIMU9EventGetTemp()

```
static float caerIMU9EventGetTemp (
            caerIMU9EventConst event )  [inline], [static]
```

Get the temperature reading. This is in °C.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

temperature in °C.

4.14.4.11 caerIMU9EventGetTimestamp()

```
static int32_t caerIMU9EventGetTimestamp (
            caerIMU9EventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.14.4.12 caerIMU9EventGetTimestamp64()

```
static int64_t caerIMU9EventGetTimestamp64 (
            caerIMU9EventConst event,
            caerIMU9EventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
packet	the IMU9EventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.14.4.13 caerIMU9EventInvalidate()

```
static void caerIMU9EventInvalidate (
            caerIMU9Event event,
            caerIMU9EventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
packet	the IMU9EventPacket pointer for the packet containing this event. Cannot be NULL.

4.14.4.14 caerIMU9EventIsValid()

```
static bool caerIMU9EventIsValid (
             caerIMU9EventConst event )  [inline], [static]
```

Check if this IMU 9-axes event is valid.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.14.4.15 caerIMU9EventPacketAllocate()

```
static caerIMU9EventPacket caerIMU9EventPacketAllocate (
             int32_t eventCapacity,
             int16_t eventSource,
             int32_t tsOverflow )  [inline], [static]
```

Allocate a new IMU 9-axes events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid IMU9EventPacket handle or NULL on error.

4.14.4.16 caerIMU9EventPacketFromPacketHeader()

```
static caerIMU9EventPacket caerIMU9EventPacketFromPacketHeader (
             caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into an IMU 9-axes event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.14.4.17 caerIMU9EventPacketFromPacketHeaderConst()

```
static caerIMU9EventPacketConst caerIMU9EventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only IMU 9-axes event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.14.4.18 caerIMU9EventPacketGetEvent()

```
static caerIMU9Event caerIMU9EventPacketGetEvent (
            caerIMU9EventPacket packet,
            int32_t n )  [inline], [static]
```

Get the IMU 9-axes event at the given index from the event packet.

Parameters

packet	a valid IMU9EventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested IMU 9-axes event. NULL on error.

4.14.4.19 caerIMU9EventPacketGetEventConst()

```
static caerIMU9EventConst caerIMU9EventPacketGetEventConst (
            caerIMU9EventPacketConst packet,
            int32_t n )  [inline], [static]
```

Get the IMU 9-axes event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid IMU9EventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only IMU 9-axes event. NULL on error.

4.14.4.20 caerIMU9EventSetAccelX()

```
static void caerIMU9EventSetAccelX (
            caerIMU9Event event,
            float accelX )  [inline], [static]
```

Set the X axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
accelX	acceleration on the X axis.

4.14.4.21 caerIMU9EventSetAccelY()

```
static void caerIMU9EventSetAccelY (
            caerIMU9Event event,
            float accelY )  [inline], [static]
```

Set the Y axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
accelY	acceleration on the Y axis.

4.14.4.22 caerIMU9EventSetAccelZ()

```
static void caerIMU9EventSetAccelZ (
             caerIMU9Event event,
             float accelZ )  [inline], [static]
```

Set the Z axis acceleration reading (from accelerometer). This is in g (1 g = 9.81 m/s²).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
accelZ	acceleration on the Z axis.

4.14.4.23 caerIMU9EventSetCompX()

```
static void caerIMU9EventSetCompX (
             caerIMU9Event event,
             float compX )  [inline], [static]
```

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

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
compX	X axis compass heading.

4.14.4.24 caerIMU9EventSetCompY()

```
static void caerIMU9EventSetCompY (
             caerIMU9Event event,
             float compY )  [inline], [static]
```

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

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
compY	Y axis compass heading.

4.14.4.25 caerIMU9EventSetCompZ()

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

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

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
compZ	Z axis compass heading.

4.14.4.26 caerIMU9EventSetGyroX()

```
static void caerIMU9EventSetGyroX (
            caerIMU9Event event,
            float gyroX )  [inline], [static]
```

Set the X axis (roll) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
gyroX	angular velocity on the X axis (roll).

4.14.4.27 caerIMU9EventSetGyroY()

```
static void caerIMU9EventSetGyroY (
            caerIMU9Event event,
            float gyroY )  [inline], [static]
```

Set the Y axis (pitch) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
gyroY	angular velocity on the Y axis (pitch).

4.14.4.28 caerIMU9EventSetGyroZ()

```
static void caerIMU9EventSetGyroZ (
            caerIMU9Event event,
            float gyroZ )  [inline], [static]
```

Set the Z axis (yaw) angular velocity reading (from gyroscope). This is in °/s (deg/sec).

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
gyroZ	angular velocity on the Z axis (yaw).

4.14.4.29 caerIMU9EventSetTemp()

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

Set the temperature reading. This is in ℃.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
temp	temperature in ℃.

4.14.4.30 caerIMU9EventSetTimestamp()

```
static void caerIMU9EventSetTimestamp (
            caerIMU9Event event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.14.4.31 caerIMU9EventValidate()

```
static void caerIMU9EventValidate (
            caerIMU9Event event,
            caerIMU9EventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid IMU9Event pointer. Cannot be NULL.
packet	the IMU9EventPacket pointer for the packet containing this event. Cannot be NULL.

4.14.4.32 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_imu9_event { uint32_t info;int32_t timestamp;float accel_x;float
accel_y;float accel_z;float gyro_x;float gyro_y;float gyro_z;float temp;float comp_x;float
comp_y;float comp_z;}  )
```

IMU 9-axes event data structure definition. This contains accelerometer and gyroscope headings, plus temperature, and magnetometer readings. The X, Y and Z axes are referred to the camera plane. X increases to the right, Y going up and Z towards where the lens is pointing. Rotation for the gyroscope is counter-clockwise along the increasing axis, for all three axes. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.14.4.33 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_imu9_event_packet { struct caer_event_packet_header packetHeader;struct
caer_imu9_event events[];}  )
```

IMU 9-axes event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.15 events/matrix4x4.h File Reference

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

Macros

- #define CAER_MATRIX4x4_ITERATOR_ALL_START(MATRIX4x4_PACKET)
- #define CAER_MATRIX4x4_CONST_ITERATOR_ALL_START(MATRIX4x4_PACKET)
- #define CAER_MATRIX4x4_ITERATOR_ALL_END }
- #define CAER_MATRIX4x4_ITERATOR_VALID_START(MATRIX4x4_PACKET)
- #define CAER_MATRIX4x4_CONST_ITERATOR_VALID_START(MATRIX4x4_PACKET)
- #define CAER_MATRIX4x4_ITERATOR_VALID_END }
- #define CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_START(MATRIX4x4_PACKET)
- #define CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_ALL_START(MATRIX4x4_PACKET)
- #define CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_END }
- #define CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_START(MATRIX4x4_PACKET)
- #define CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_VALID_START(MATRIX4x4_PACKET)
- #define CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_END }

- #define MATRIX4x4_TYPE_SHIFT 1
- #define MATRIX4x4_TYPE_MASK 0x0000007F
- #define MATRIX4x4_SCALE_SHIFT 8
- #define MATRIX4x4_SCALE_MASK 0x000000FF

Typedefs

- typedef struct caer_matrix4x4_event ∗ caerMatrix4x4Event
- typedef const struct caer_matrix4x4_event ∗ **caerMatrix4x4EventConst**
- typedef struct caer_matrix4x4_event_packet ∗ caerMatrix4x4EventPacket
- typedef const struct caer_matrix4x4_event_packet ∗ **caerMatrix4x4EventPacketConst**

Functions

- PACKED_STRUCT (struct caer_matrix4x4_event { uint32_t info;float m[4][4];int32_t timestamp;})
- PACKED_STRUCT (struct caer_matrix4x4_event_packet { struct caer_event_packet_header packet↩ Header;struct caer_matrix4x4_event events[ ];})
- static caerMatrix4x4EventPacket caerMatrix4x4EventPacketAllocate (int32_t eventCapacity, int16_t event↩ Source, int32_t tsOverflow)
- static caerMatrix4x4EventPacket caerMatrix4x4EventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerMatrix4x4EventPacketConst caerMatrix4x4EventPacketFromPacketHeaderConst (caerEvent↩ PacketHeaderConst header)
- static caerMatrix4x4Event caerMatrix4x4EventPacketGetEvent (caerMatrix4x4EventPacket packet, int32_t n)
- static caerMatrix4x4EventConst caerMatrix4x4EventPacketGetEventConst (caerMatrix4x4EventPacketConst packet, int32_t n)
- static int32_t caerMatrix4x4EventGetTimestamp (caerMatrix4x4EventConst event)
- static int64_t caerMatrix4x4EventGetTimestamp64 (caerMatrix4x4EventConst event, caerMatrix4x4Event↩ PacketConst packet)
- static void caerMatrix4x4EventSetTimestamp (caerMatrix4x4Event event, int32_t timestamp)
- static bool caerMatrix4x4EventIsValid (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventValidate (caerMatrix4x4Event event, caerMatrix4x4EventPacket packet)
- static void caerMatrix4x4EventInvalidate (caerMatrix4x4Event event, caerMatrix4x4EventPacket packet)
- static uint8_t caerMatrix4x4EventGetType (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetType (caerMatrix4x4Event event, uint8_t type)
- static int8_t caerMatrix4x4EventGetScale (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetScale (caerMatrix4x4Event event, int8_t scale)
- static float caerMatrix4x4EventGetM00 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM00 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM01 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM01 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM02 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM02 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM03 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM03 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM10 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM10 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM11 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM11 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM12 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM12 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM13 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM13 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM20 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM20 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM21 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM21 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM22 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM22 (caerMatrix4x4Event event, float x)

- static float caerMatrix4x4EventGetM23 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM23 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM30 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM30 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM31 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM31 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM32 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM32 (caerMatrix4x4Event event, float x)
- static float caerMatrix4x4EventGetM33 (caerMatrix4x4EventConst event)
- static void caerMatrix4x4EventSetM33 (caerMatrix4x4Event event, float x)

4.15.1 Detailed Description

THIS EVENT DEFINITION IS STILL TO BE CONSIDERED EXPERIMENTAL AND IS SUBJECT TO FUTURE C↩
HANGES AND REVISIONS!

Matrix4x4 Events format definition and handling functions. This contains a matrix of dimensions 4x4 with floats entries, together with support for distinguishing type and scale. Useful for homogeneous coordinates for example.

m00 m01 m02 m03 m10 m11 m12 m13 m20 m21 m22 m23 m30 m31 m32 m33

4.15.2 Macro Definition Documentation

4.15.2.1 CAER_MATRIX4x4_CONST_ITERATOR_ALL_START

```
#define CAER_MATRIX4x4_CONST_ITERATOR_ALL_START(
                MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter = 0;
       \
         caerMatrix4x4IteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(MATRIX4x4_PACKET)->packetHeader); \
         caerMatrix4x4IteratorCounter++) {
           \
        caerMatrix4x4EventConst caerMatrix4x4IteratorElement
           \
            = caerMatrix4x4EventPacketGetEventConst(MATRIX4x4_PACKET,
      caerMatrix4x4IteratorCounter);
```

Const-Iterator over all Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerMatrix4x4IteratorElement' variable of type caer↩
Matrix4x4EventConst.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.2 CAER_MATRIX4x4_CONST_ITERATOR_VALID_START

```
#define CAER_MATRIX4x4_CONST_ITERATOR_VALID_START(
                MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter = 0;
       \
         caerMatrix4x4IteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(MATRIX4x4_PACKET)->packetHeader); \
         caerMatrix4x4IteratorCounter++) {
           \
        caerMatrix4x4EventConst caerMatrix4x4IteratorElement
           \
            = caerMatrix4x4EventPacketGetEventConst(MATRIX4x4_PACKET,
      caerMatrix4x4IteratorCounter);              \
        if (!caerMatrix4x4EventIsValid(caerMatrix4x4IteratorElement)) {
                                    \
            continue;
           \
        }
```

Const-Iterator over only the valid Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4↩ IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerMatrix4x4IteratorElement' variable of type caerMatrix4x4EventConst.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.3 CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_ALL_START(
                MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter                                   \
         = caerEventPacketHeaderGetEventNumber(&(MATRIX4x4_PACKET)->
      packetHeader) - 1; \
         caerMatrix4x4IteratorCounter >= 0; caerMatrix4x4IteratorCounter--) {          \
        caerMatrix4x4EventConst caerMatrix4x4IteratorElement                           \
            = caerMatrix4x4EventPacketGetEventConst(MATRIX4x4_PACKET,
      caerMatrix4x4IteratorCounter);
```

Const-Reverse iterator over all Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4↩ IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerMatrix4x4IteratorElement' variable of type caerMatrix4x4EventConst.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.4 CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_MATRIX4x4_CONST_REVERSE_ITERATOR_VALID_START(
                MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter                                                  \
         = caerEventPacketHeaderGetEventNumber(&(MATRIX4x4_PACKET)->
      packetHeader) - 1;                         \
         caerMatrix4x4IteratorCounter >= 0; caerMatrix4x4IteratorCounter--) {                  \
        caerMatrix4x4EventConst caerMatrix4x4IteratorElement                                   \
            = caerMatrix4x4EventPacketGetEventConst(MATRIX4x4_PACKET,
      caerMatrix4x4IteratorCounter); \
        if (!caerMatrix4x4EventIsValid(caerMatrix4x4IteratorElement)) {
                          \
            continue;                                                                          \
        }
```

Const-Reverse iterator over only the valid Matrix4x4 events in a packet. Returns the current index in the 'caer←Matrix4x4IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerMatrix4x4Iterator←Element' variable of type caerMatrix4x4EventConst.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.5 CAER_MATRIX4x4_ITERATOR_ALL_END

```
#define CAER_MATRIX4x4_ITERATOR_ALL_END }
```

Iterator close statement.

4.15.2.6 CAER_MATRIX4x4_ITERATOR_ALL_START

```
#define CAER_MATRIX4x4_ITERATOR_ALL_START(
                MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter = 0;
       \
         caerMatrix4x4IteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(MATRIX4x4_PACKET)->packetHeader); \
         caerMatrix4x4IteratorCounter++) {
           \
        caerMatrix4x4Event caerMatrix4x4IteratorElement
           \
            = caerMatrix4x4EventPacketGetEvent(MATRIX4x4_PACKET,
      caerMatrix4x4IteratorCounter);
```

Iterator over all Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4IteratorCounter' variable of type 'int32_t' and the current event in the 'caerMatrix4x4IteratorElement' variable of type caerMatrix4x4Event.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.7 CAER_MATRIX4x4_ITERATOR_VALID_END

```
#define CAER_MATRIX4x4_ITERATOR_VALID_END }
```

Iterator close statement.

4.15.2.8 CAER_MATRIX4x4_ITERATOR_VALID_START

```
#define CAER_MATRIX4x4_ITERATOR_VALID_START(
                MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter = 0;
       \
         caerMatrix4x4IteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(MATRIX4x4_PACKET)->packetHeader); \
         caerMatrix4x4IteratorCounter++) {
           \
        caerMatrix4x4Event caerMatrix4x4IteratorElement
           \
            = caerMatrix4x4EventPacketGetEvent(MATRIX4x4_PACKET,
      caerMatrix4x4IteratorCounter);                 \
        if (!caerMatrix4x4EventIsValid(caerMatrix4x4IteratorElement)) {
                                        \
            continue;
           \
        }
```

Iterator over only the valid Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4Iterator↵Counter' variable of type 'int32_t' and the current event in the 'caerMatrix4x4IteratorElement' variable of type caer↵Matrix4x4Event.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.9 CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_END

```
#define CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.15.2.10 CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_START

```
#define CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_START(
                MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter                                     \
         = caerEventPacketHeaderGetEventNumber(&(MATRIX4x4_PACKET)->
      packetHeader) - 1; \
         caerMatrix4x4IteratorCounter >= 0; caerMatrix4x4IteratorCounter--) {          \
        caerMatrix4x4Event caerMatrix4x4IteratorElement                                \
            = caerMatrix4x4EventPacketGetEvent(MATRIX4x4_PACKET,
      caerMatrix4x4IteratorCounter);
```

Reverse iterator over all Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4Iterator↵Counter' variable of type 'int32_t' and the current event in the 'caerMatrix4x4IteratorElement' variable of type caer↵Matrix4x4Event.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.11 CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_END

```
#define CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.15.2.12 CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_START

```
#define CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_START(
             MATRIX4x4_PACKET )
```

Value:

```
for (int32_t caerMatrix4x4IteratorCounter                                                   \
        = caerEventPacketHeaderGetEventNumber(&(MATRIX4x4_PACKET)->
     packetHeader) - 1;          \
        caerMatrix4x4IteratorCounter >= 0; caerMatrix4x4IteratorCounter--) {                   \
       caerMatrix4x4Event caerMatrix4x4IteratorElement                                        \
           = caerMatrix4x4EventPacketGetEvent(MATRIX4x4_PACKET,
     caerMatrix4x4IteratorCounter); \
       if (!caerMatrix4x4EventIsValid(caerMatrix4x4IteratorElement)) {
                   \
           continue;                                                                          \
       }
```

Reverse iterator over only the valid Matrix4x4 events in a packet. Returns the current index in the 'caerMatrix4x4←IteratorCounter' variable of type 'int32_t' and the current event in the 'caerMatrix4x4IteratorElement' variable of type caerMatrix4x4Event.

MATRIX4x4_PACKET: a valid Matrix4x4EventPacket pointer. Cannot be NULL.

4.15.2.13 MATRIX4x4_SCALE_MASK

```
#define MATRIX4x4_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Matrix4x4 event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.15.2.14 MATRIX4x4_SCALE_SHIFT

```
#define MATRIX4x4_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Matrix4x4 event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.15.2.15 MATRIX4x4_TYPE_MASK

```
#define MATRIX4x4_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Matrix4x4 event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.15.2.16 MATRIX4x4_TYPE_SHIFT

```
#define MATRIX4x4_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Matrix4x4 event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.15.3 Typedef Documentation

4.15.3.1 caerMatrix4x4Event

```
typedef struct caer_matrix4x4_event* caerMatrix4x4Event
```

Type for pointer to Matrix4x4 event data structure.

4.15.3.2 caerMatrix4x4EventPacket

```
typedef struct caer_matrix4x4_event_packet* caerMatrix4x4EventPacket
```

Type for pointer to Matrix4x4 event packet data structure.

4.15.4 Function Documentation

4.15.4.1 caerMatrix4x4EventGetM00()

```
static float caerMatrix4x4EventGetM00 (
             caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M00 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M00 element.

4.15.4.2 caerMatrix4x4EventGetM01()

```
static float caerMatrix4x4EventGetM01 (
            caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M01 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M01 element.

4.15.4.3 caerMatrix4x4EventGetM02()

```
static float caerMatrix4x4EventGetM02 (
            caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M02 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M02 element.

4.15.4.4 caerMatrix4x4EventGetM03()

```
static float caerMatrix4x4EventGetM03 (
            caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M03 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M01 element.

4.15.4.5 caerMatrix4x4EventGetM10()

```
static float caerMatrix4x4EventGetM10 (
            caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M10 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M10 element.

4.15.4.6 caerMatrix4x4EventGetM11()

```
static float caerMatrix4x4EventGetM11 (
            caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M11 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M11 element.

4.15.4.7 caerMatrix4x4EventGetM12()

```
static float caerMatrix4x4EventGetM12 (
            caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M12 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M12 element.

4.15.4.8 caerMatrix4x4EventGetM13()

```
static float caerMatrix4x4EventGetM13 (
             caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M13 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M13 element.

4.15.4.9 caerMatrix4x4EventGetM20()

```
static float caerMatrix4x4EventGetM20 (
             caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M20 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M20 element.

4.15.4.10 caerMatrix4x4EventGetM21()

```
static float caerMatrix4x4EventGetM21 (
             caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M21 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M21 element.

4.15.4.11 caerMatrix4x4EventGetM22()

```
static float caerMatrix4x4EventGetM22 (
             caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M22 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M22 element.

4.15.4.12 caerMatrix4x4EventGetM23()

```
static float caerMatrix4x4EventGetM23 (
             caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M23 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M23 element.

4.15.4.13 caerMatrix4x4EventGetM30()

```
static float caerMatrix4x4EventGetM30 (
             caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M30 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M30 element.

4.15.4.14 caerMatrix4x4EventGetM31()

```
static float caerMatrix4x4EventGetM31 (
            caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M31 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M31 element.

4.15.4.15 caerMatrix4x4EventGetM32()

```
static float caerMatrix4x4EventGetM32 (
            caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M32 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M32 element.

4.15.4.16 caerMatrix4x4EventGetM33()

```
static float caerMatrix4x4EventGetM33 (
            caerMatrix4x4EventConst event )  [inline], [static]
```

Get the M33 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

M33 element.

4.15.4.17 caerMatrix4x4EventGetScale()

```
static int8_t caerMatrix4x4EventGetScale (
            caerMatrix4x4EventConst event )  [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

the Matrix4x4 measurement scale.

4.15.4.18 caerMatrix4x4EventGetTimestamp()

```
static int32_t caerMatrix4x4EventGetTimestamp (
            caerMatrix4x4EventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.15.4.19 caerMatrix4x4EventGetTimestamp64()

```
static int64_t caerMatrix4x4EventGetTimestamp64 (
             caerMatrix4x4EventConst event,
             caerMatrix4x4EventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
packet	the Matrix4x4EventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.15.4.20 caerMatrix4x4EventGetType()

```
static uint8_t caerMatrix4x4EventGetType (
             caerMatrix4x4EventConst event )  [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

the Matrix4x4 measurement type.

4.15.4.21 caerMatrix4x4EventInvalidate()

```
static void caerMatrix4x4EventInvalidate (
             caerMatrix4x4Event event,
             caerMatrix4x4EventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
packet	the Matrix4x4EventPacket pointer for the packet containing this event. Cannot be NULL.

4.15.4.22 caerMatrix4x4EventIsValid()

```
static bool caerMatrix4x4EventIsValid (
            caerMatrix4x4EventConst event )  [inline], [static]
```

Check if this Matrix4x4 event is valid.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.15.4.23 caerMatrix4x4EventPacketAllocate()

```
static caerMatrix4x4EventPacket caerMatrix4x4EventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )  [inline], [static]
```

Allocate a new Matrix4x4 events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid Matrix4x4EventPacket handle or NULL on error.

4.15.4.24 caerMatrix4x4EventPacketFromPacketHeader()

```
static caerMatrix4x4EventPacket caerMatrix4x4EventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into a Matrix4x4 event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.15.4.25 caerMatrix4x4EventPacketFromPacketHeaderConst()

```
static caerMatrix4x4EventPacketConst caerMatrix4x4EventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only Matrix4x4 event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.15.4.26 caerMatrix4x4EventPacketGetEvent()

```
static caerMatrix4x4Event caerMatrix4x4EventPacketGetEvent (
            caerMatrix4x4EventPacket packet,
            int32_t n )  [inline], [static]
```

Get the Matrix4x4 event at the given index from the event packet.

Parameters

packet	a valid Matrix4x4EventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Matrix4x4 event. NULL on error.

4.15.4.27 caerMatrix4x4EventPacketGetEventConst()

```
static caerMatrix4x4EventConst caerMatrix4x4EventPacketGetEventConst (
             caerMatrix4x4EventPacketConst packet,
             int32_t n )  [inline], [static]
```

Get the Matrix4x4 event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid Matrix4x4EventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only Matrix4x4 event. NULL on error.

4.15.4.28 caerMatrix4x4EventSetM00()

```
static void caerMatrix4x4EventSetM00 (
             caerMatrix4x4Event event,
             float x )  [inline], [static]
```

Set the M00 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	m00 value.

4.15.4.29 caerMatrix4x4EventSetM01()

```
static void caerMatrix4x4EventSetM01 (
             caerMatrix4x4Event event,
             float x )  [inline], [static]
```

Set the M01 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	m01 value.

4.15.4.30 caerMatrix4x4EventSetM02()

```
static void caerMatrix4x4EventSetM02 (
            caerMatrix4x4Event event,
            float x )  [inline], [static]
```

Set the M02 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	m02 value.

4.15.4.31 caerMatrix4x4EventSetM03()

```
static void caerMatrix4x4EventSetM03 (
            caerMatrix4x4Event event,
            float x )  [inline], [static]
```

Set the M03 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	m03 value.

4.15.4.32 caerMatrix4x4EventSetM10()

```
static void caerMatrix4x4EventSetM10 (
            caerMatrix4x4Event event,
            float x )  [inline], [static]
```

Set the M10 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	m10 value.

4.15.4.33 caerMatrix4x4EventSetM11()

```
static void caerMatrix4x4EventSetM11 (
            caerMatrix4x4Event event,
            float x )  [inline], [static]
```

Set the M11 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	m11 value.

4.15.4.34 caerMatrix4x4EventSetM12()

```
static void caerMatrix4x4EventSetM12 (
            caerMatrix4x4Event event,
            float x )  [inline], [static]
```

Set the M12 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	m12 value.

4.15.4.35 caerMatrix4x4EventSetM13()

```
static void caerMatrix4x4EventSetM13 (
            caerMatrix4x4Event event,
            float x )  [inline], [static]
```

Set the M13 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	m13 value.

4.15.4.36 caerMatrix4x4EventSetM20()

```
static void caerMatrix4x4EventSetM20 (
            caerMatrix4x4Event event,
            float x )  [inline], [static]
```

Set the M20 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	M20 value.

4.15.4.37 caerMatrix4x4EventSetM21()

```
static void caerMatrix4x4EventSetM21 (
             caerMatrix4x4Event event,
             float x )  [inline], [static]
```

Set the M21 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	M21 value.

4.15.4.38 caerMatrix4x4EventSetM22()

```
static void caerMatrix4x4EventSetM22 (
             caerMatrix4x4Event event,
             float x )  [inline], [static]
```

Set the M22 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	M22 value.

4.15.4.39 caerMatrix4x4EventSetM23()

```
static void caerMatrix4x4EventSetM23 (
             caerMatrix4x4Event event,
             float x )  [inline], [static]
```

Set the M23 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	M23 value.

4.15.4.40 caerMatrix4x4EventSetM30()

```
static void caerMatrix4x4EventSetM30 (
             caerMatrix4x4Event event,
             float x )  [inline], [static]
```

Set the M30 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	M30 value.

4.15.4.41 caerMatrix4x4EventSetM31()

```
static void caerMatrix4x4EventSetM31 (
             caerMatrix4x4Event event,
             float x )  [inline], [static]
```

Set the M31 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	M31 value.

4.15.4.42 caerMatrix4x4EventSetM32()

```
static void caerMatrix4x4EventSetM32 (
             caerMatrix4x4Event event,
             float x )  [inline], [static]
```

Set the M32 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	M32 value.

4.15.4.43 caerMatrix4x4EventSetM33()

```
static void caerMatrix4x4EventSetM33 (
            caerMatrix4x4Event event,
            float x )  [inline], [static]
```

Set the M33 element.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
x	M33 value.

4.15.4.44 caerMatrix4x4EventSetScale()

```
static void caerMatrix4x4EventSetScale (
            caerMatrix4x4Event event,
            int8_t scale )  [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
scale	the Matrix4x4 measurement scale.

4.15.4.45 caerMatrix4x4EventSetTimestamp()

```
static void caerMatrix4x4EventSetTimestamp (
            caerMatrix4x4Event event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.15.4.46 caerMatrix4x4EventSetType()

```
static void caerMatrix4x4EventSetType (
            caerMatrix4x4Event event,
            uint8_t type )  [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
type	the Matrix4x4 measurement type.

4.15.4.47 caerMatrix4x4EventValidate()

```
static void caerMatrix4x4EventValidate (
            caerMatrix4x4Event event,
            caerMatrix4x4EventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid Matrix4x4Event pointer. Cannot be NULL.
packet	the Matrix4x4EventPacket pointer for the packet containing this event. Cannot be NULL.

4.15.4.48 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_matrix4x4_event { uint32_t info;float m[4][4];int32_t timestamp;}  )
```

Matrix4x4 event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The measurements are stored as floats. m00 m01 m02 m03 m10 m11 m12 m13 m20 m21 m22 m23 m30 m31 m32 m33 Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.15.4.49 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_matrix4x4_event_packet { struct caer_event_packet_header packet↩
Header;struct caer_matrix4x4_event events[];}  )
```

Matrix4x4 event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.16 events/packetContainer.h File Reference

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

Macros

- #define CAER_EVENT_PACKET_CONTAINER_ITERATOR_START(PACKET_CONTAINER)
- #define CAER_EVENT_PACKET_CONTAINER_CONST_ITERATOR_START(PACKET_CONTAINER)
- #define CAER_EVENT_PACKET_CONTAINER_ITERATOR_END

Typedefs

- typedef struct caer_event_packet_container ∗ caerEventPacketContainer
- typedef const struct caer_event_packet_container ∗ **caerEventPacketContainerConst**

Functions

- PACKED_STRUCT (struct caer_event_packet_container { int64_t lowestEventTimestamp;int64_←t highestEventTimestamp;int32_t eventsNumber;int32_t eventsValidNumber;int32_t eventPackets←Number;caerEventPacketHeader eventPackets[ ];})
- static caerEventPacketContainer caerEventPacketContainerAllocate (int32_t eventPacketsNumber)
- static void caerEventPacketContainerUpdateStatistics (caerEventPacketContainer container)
- static int32_t caerEventPacketContainerGetEventPacketsNumber (caerEventPacketContainerConst container)
- static void caerEventPacketContainerSetEventPacketsNumber (caerEventPacketContainer container, int32←_t eventPacketsNumber)
- static caerEventPacketHeader caerEventPacketContainerGetEventPacket (caerEventPacketContainerConst container, int32_t n)
- static caerEventPacketHeaderConst caerEventPacketContainerGetEventPacketConst (caerEventPacket←ContainerConst container, int32_t n)
- static void caerEventPacketContainerSetEventPacket (caerEventPacketContainer container, int32_t n, caerEventPacketHeader packetHeader)
- static void caerEventPacketContainerFree (caerEventPacketContainer container)
- static int64_t caerEventPacketContainerGetLowestEventTimestamp (caerEventPacketContainerConst container)
- static int64_t caerEventPacketContainerGetHighestEventTimestamp (caerEventPacketContainerConst container)
- static int32_t caerEventPacketContainerGetEventsNumber (caerEventPacketContainerConst container)
- static int32_t caerEventPacketContainerGetEventsValidNumber (caerEventPacketContainerConst container)
- static caerEventPacketHeader caerEventPacketContainerFindEventPacketByType (caerEventPacket←ContainerConst container, int16_t typeID)
- static caerEventPacketHeaderConst caerEventPacketContainerFindEventPacketByTypeConst (caerEvent←PacketContainerConst container, int16_t typeID)
- static caerEventPacketContainer caerEventPacketContainerCopyAllEvents (caerEventPacketContainerConst container)
- static caerEventPacketContainer caerEventPacketContainerCopyValidEvents (caerEventPacketContainer←Const container)

4.16.1 Detailed Description

EventPacketContainer format definition and handling functions. An EventPacketContainer is a logical construct that contains packets of events (EventPackets) of different event types, with the aim of keeping related events of differing types, such as DVS and IMU data, together. Such a relation is usually based on time intervals, trying to keep groups of event happening in a certain time-slice together. This time-order is based on the *main* time-stamp of an event, the one whose offset is referenced in the event packet header and that is used by the caerGenericEvent∗() functions. It's guaranteed that all conforming input modules keep to this rule, generating containers that include all events from all types within the given time-slice. The smallest and largest timestamps are tracked at the packet container level as a convenience, to avoid having to examine all packets for this often useful piece of information. All integers are in their native host format, as this is a purely internal, in-memory data structure, never meant for exchange between different systems (and different endianness).

== Packet Containers and Input Modules == The "packeting system" works in this way: events are accumulated by type in a packet, and that packet is part of a packet container, by an input module. The packet container is then sent out for processing when either the configured time limit or the size limit are hit. The time limit is always active, in microseconds, and basically tells you the time-span an event packet covers. This enables regular, constant delivery of packets, that cover a period of time. The size limit is an addon to prevent packets to grow to immense sizes (like if the time limit is high and there is lots of activity). As soon as a packet hits the number of events in the size limit, it is sent out. The regular time limit is not reset in this case. This size limit can be disabled by setting it to 0. The cAER DVS128/DAVIS/File/Network input modules call these two configuration variables "PacketContainerInterval" and "PacketContainerMaxPacketSize". Too small packet sizes or intervals simply mean more packets, which may negatively affect performance. It's usually a good idea to set the size to something around 4-8K, and the time to a good value based on the application you're building, so if you need ms-reaction-time, you probably want to set it to 1000µs, so that you do get new data every ms. If on the other hand you're looking at a static scene and just want to detect that something is passing by once every while, a higher number like 100ms might also be perfectly appropriate.

4.16.2 Macro Definition Documentation

4.16.2.1 CAER_EVENT_PACKET_CONTAINER_CONST_ITERATOR_START

```
#define CAER_EVENT_PACKET_CONTAINER_CONST_ITERATOR_START(
                PACKET_CONTAINER )
```

Value:

```
if ((PACKET_CONTAINER) != NULL) {                                          \
        for (int32_t caerEventPacketContainerIteratorCounter = 0;              \
             caerEventPacketContainerIteratorCounter                           \
             < caerEventPacketContainerGetEventPacketsNumber(
      PACKET_CONTAINER); \
             caerEventPacketContainerIteratorCounter++) {                      \
            caerEventPacketHeaderConst caerEventPacketContainerIteratorElement \
                = caerEventPacketContainerGetEventPacketConst(
                      \
                       PACKET_CONTAINER, caerEventPacketContainerIteratorCounter); \
            if (caerEventPacketContainerIteratorElement == NULL) {            \
                continue;                                                      \
            }
```

Const-Iterator over all event packets in an event packet container. Returns the current index in the 'caerEvent←PacketContainerIteratorCounter' variable of type 'int32_t' and the current read-only event packet in the 'caerEvent←PacketContainerIteratorElement' variable of type caerEventPacketHeaderConst. The current packet may be NULL, in which case it is skipped during iteration.

PACKET_CONTAINER: a valid EventPacketContainer handle. If NULL, no iteration is performed.

4.16.2.2 CAER_EVENT_PACKET_CONTAINER_ITERATOR_END

```
#define CAER_EVENT_PACKET_CONTAINER_ITERATOR_END
```

Value:

```
}                                            \
    }
```

Iterator close statement.

4.16.2.3 CAER_EVENT_PACKET_CONTAINER_ITERATOR_START

```
#define CAER_EVENT_PACKET_CONTAINER_ITERATOR_START(
                PACKET_CONTAINER )
```

Value:

```
if ((PACKET_CONTAINER) != NULL) {
            \
        for (int32_t caerEventPacketContainerIteratorCounter = 0;
                \
             caerEventPacketContainerIteratorCounter
                \
             < caerEventPacketContainerGetEventPacketsNumber(
      PACKET_CONTAINER);                                       \
             caerEventPacketContainerIteratorCounter++) {
                \
            caerEventPacketHeader caerEventPacketContainerIteratorElement
                \
                = caerEventPacketContainerGetEventPacket(
      PACKET_CONTAINER, caerEventPacketContainerIteratorCounter); \
            if (caerEventPacketContainerIteratorElement == NULL) {
                \
                continue;
                \
            }
```

Iterator over all event packets in an event packet container. Returns the current index in the 'caerEventPacket←ContainerIteratorCounter' variable of type 'int32_t' and the current event packet in the 'caerEventPacketContainer←IteratorElement' variable of type caerEventPacketHeader. The current packet may be NULL, in which case it is skipped during iteration.

PACKET_CONTAINER: a valid EventPacketContainer handle. If NULL, no iteration is performed.

4.16.3 Typedef Documentation

4.16.3.1 caerEventPacketContainer

```
typedef struct caer_event_packet_container* caerEventPacketContainer
```

Type for pointer to EventPacketContainer data structure.

4.16.4 Function Documentation

4.16.4.1 caerEventPacketContainerAllocate()

```
static caerEventPacketContainer caerEventPacketContainerAllocate (
              int32_t eventPacketsNumber )  [inline], [static]
```

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

Parameters

eventPacketsNumber	the maximum number of EventPacket pointers that can be stored in this container.

Returns

a valid EventPacketContainer handle or NULL on error.

4.16.4.2 caerEventPacketContainerCopyAllEvents()

```
static caerEventPacketContainer caerEventPacketContainerCopyAllEvents (
             caerEventPacketContainerConst container )  [inline], [static]
```

Make a deep copy of an event packet container and all of its event packets and their current events.

Parameters

container	an event packet container to copy.

Returns

a deep copy of an event packet container, containing all events.

4.16.4.3 caerEventPacketContainerCopyValidEvents()

```
static caerEventPacketContainer caerEventPacketContainerCopyValidEvents (
             caerEventPacketContainerConst container )  [inline], [static]
```

Make a deep copy of an event packet container, with its event packets sized down to only include the currently valid events (eventValid), and discarding everything else.

Parameters

container	an event packet container to copy.

Returns

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

4.16.4.4 caerEventPacketContainerFindEventPacketByType()

```
static caerEventPacketHeader caerEventPacketContainerFindEventPacketByType (
            caerEventPacketContainerConst container,
            int16_t typeID )  [inline], [static]
```

Get the pointer to an EventPacket stored in this container with the given event type. This returns the first found event packet with that type ID, or NULL if we get to the end without finding any such event packet.

Parameters

container	a valid EventPacketContainer handle. If NULL, returns NULL too.
typeID	the event type to search for.

Returns

a pointer to an EventPacket with a certain type or NULL if none found.

4.16.4.5 caerEventPacketContainerFindEventPacketByTypeConst()

```
static caerEventPacketHeaderConst caerEventPacketContainerFindEventPacketByTypeConst (
            caerEventPacketContainerConst container,
            int16_t typeID )  [inline], [static]
```

Get the pointer to a read-only EventPacket stored in this container with the given event type. This returns the first found event packet with that type ID, or NULL if we get to the end without finding any such event packet.

Parameters

container	a valid EventPacketContainer handle. If NULL, returns NULL too.
typeID	the event type to search for.

Returns

a pointer to a read-only EventPacket with a certain type or NULL if none found.

4.16.4.6 caerEventPacketContainerFree()

```
static void caerEventPacketContainerFree (
            caerEventPacketContainer container )  [inline], [static]
```

Free the memory occupied by an EventPacketContainer, as well as freeing all of its contained EventPackets and their memory. If you don't want the contained EventPackets to be freed, make sure that you set their pointers to NULL before calling this.

Parameters

container	the container to be freed.

4.16.4.7 caerEventPacketContainerGetEventPacket()

```
static caerEventPacketHeader caerEventPacketContainerGetEventPacket (
            caerEventPacketContainerConst container,
            int32_t n )  [inline], [static]
```

Get the pointer to the EventPacket stored in this container at the given index.

Parameters

container	a valid EventPacketContainer handle. If NULL, returns NULL too.
n	the index of the EventPacket to get.

Returns

a pointer to an EventPacket or NULL on error.

4.16.4.8 caerEventPacketContainerGetEventPacketConst()

```
static caerEventPacketHeaderConst caerEventPacketContainerGetEventPacketConst (
            caerEventPacketContainerConst container,
            int32_t n )  [inline], [static]
```

Get the pointer to the EventPacket stored in this container at the given index. This is a read-only EventPacket, do not change its contents in any way!

Parameters

container	a valid EventPacketContainer handle. If NULL, returns NULL too.
n	the index of the EventPacket to get.

Returns

a pointer to a read-only EventPacket or NULL on error.

4.16.4.9 caerEventPacketContainerGetEventPacketsNumber()

```
static int32_t caerEventPacketContainerGetEventPacketsNumber (
            caerEventPacketContainerConst container )  [inline], [static]
```

Get the maximum number of EventPacket pointers that can be stored in this particular EventPacketContainer.

Parameters

container	a valid EventPacketContainer handle. If NULL, zero is returned.

Returns

the number of EventPacket pointers that can be contained.

4.16.4.10 caerEventPacketContainerGetEventsNumber()

```
static int32_t caerEventPacketContainerGetEventsNumber (
            caerEventPacketContainerConst container )  [inline], [static]
```

Get the number of events contained in this event packet container.

Parameters

container	a valid EventPacketContainer handle. If NULL, 0 is returned.

Returns

the number of events in this container.

4.16.4.11 caerEventPacketContainerGetEventsValidNumber()

```
static int32_t caerEventPacketContainerGetEventsValidNumber (
            caerEventPacketContainerConst container )  [inline], [static]
```

Get the number of valid events contained in this event packet container.

Parameters

container	a valid EventPacketContainer handle. If NULL, 0 is returned.

Returns

the number of valid events in this container.

4.16.4.12 caerEventPacketContainerGetHighestEventTimestamp()

```
static int64_t caerEventPacketContainerGetHighestEventTimestamp (
             caerEventPacketContainerConst container )  [inline], [static]
```

Get the highest timestamp contained in this event packet container.

Parameters

container	a valid EventPacketContainer handle. If NULL, -1 is returned.

Returns

the highest timestamp (in µs) or -1 if not initialized.

4.16.4.13 caerEventPacketContainerGetLowestEventTimestamp()

```
static int64_t caerEventPacketContainerGetLowestEventTimestamp (
             caerEventPacketContainerConst container )  [inline], [static]
```

Get the lowest timestamp contained in this event packet container.

Parameters

container	a valid EventPacketContainer handle. If NULL, -1 is returned.

Returns

the lowest timestamp (in µs) or -1 if not initialized.

4.16.4.14 caerEventPacketContainerSetEventPacket()

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

Set the pointer to the EventPacket stored in this container at the given index.

Parameters

container	a valid EventPacketContainer handle. If NULL, nothing happens.
n	the index of the EventPacket to set.
packetHeader	a pointer to an EventPacket's header. Can be NULL.

4.16.4.15 caerEventPacketContainerSetEventPacketsNumber()

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

Set the maximum number of EventPacket pointers that can be stored in this particular EventPacketContainer. This should never be used directly, caerEventPacketContainerAllocate() sets this for you.

Parameters

container	a valid EventPacketContainer handle. If NULL, nothing happens.
eventPacketsNumber	the number of EventPacket pointers that can be contained.

4.16.4.16 caerEventPacketContainerUpdateStatistics()

```
static void caerEventPacketContainerUpdateStatistics (
            caerEventPacketContainer container )  [inline], [static]
```

Recalculates and updates all the packet-container level statistics (event counts and timestamps).

Parameters

container	a valid EventPacketContainer handle. If NULL, nothing happens.

4.16.4.17 PACKED_STRUCT()

```
PACKED_STRUCT (
            struct caer_event_packet_container { int64_t lowestEventTimestamp;int64_t highest↩
EventTimestamp;int32_t eventsNumber;int32_t eventsValidNumber;int32_t eventPacketsNumber;caerEventPacketHeader
eventPackets[];}  )
```

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

4.17 events/point1d.h File Reference

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

Macros

- #define CAER_POINT1D_ITERATOR_ALL_START(POINT1D_PACKET)
- #define CAER_POINT1D_CONST_ITERATOR_ALL_START(POINT1D_PACKET)
- #define CAER_POINT1D_ITERATOR_ALL_END }
- #define CAER_POINT1D_ITERATOR_VALID_START(POINT1D_PACKET)
- #define CAER_POINT1D_CONST_ITERATOR_VALID_START(POINT1D_PACKET)
- #define CAER_POINT1D_ITERATOR_VALID_END }
- #define CAER_POINT1D_REVERSE_ITERATOR_ALL_START(POINT1D_PACKET)
- #define CAER_POINT1D_CONST_REVERSE_ITERATOR_ALL_START(POINT1D_PACKET)
- #define CAER_POINT1D_REVERSE_ITERATOR_ALL_END }
- #define CAER_POINT1D_REVERSE_ITERATOR_VALID_START(POINT1D_PACKET)
- #define CAER_POINT1D_CONST_REVERSE_ITERATOR_VALID_START(POINT1D_PACKET)
- #define CAER_POINT1D_REVERSE_ITERATOR_VALID_END }

- #define POINT1D_TYPE_SHIFT 1
- #define POINT1D_TYPE_MASK 0x0000007F
- #define POINT1D_SCALE_SHIFT 8
- #define POINT1D_SCALE_MASK 0x000000FF

Typedefs

- typedef struct caer_point1d_event ∗ caerPoint1DEvent
- typedef const struct caer_point1d_event ∗ **caerPoint1DEventConst**
- typedef struct caer_point1d_event_packet ∗ caerPoint1DEventPacket
- typedef const struct caer_point1d_event_packet ∗ **caerPoint1DEventPacketConst**

Functions

- PACKED_STRUCT (struct caer_point1d_event { uint32_t info;float x;int32_t timestamp;})
- PACKED_STRUCT (struct caer_point1d_event_packet { struct caer_event_packet_header packet↩ Header;struct caer_point1d_event events[ ];})
- static caerPoint1DEventPacket caerPoint1DEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerPoint1DEventPacket caerPoint1DEventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerPoint1DEventPacketConst caerPoint1DEventPacketFromPacketHeaderConst (caerEventPacket↩ HeaderConst header)
- static caerPoint1DEvent caerPoint1DEventPacketGetEvent (caerPoint1DEventPacket packet, int32_t n)
- static caerPoint1DEventConst caerPoint1DEventPacketGetEventConst (caerPoint1DEventPacketConst packet, int32_t n)
- static int32_t caerPoint1DEventGetTimestamp (caerPoint1DEventConst event)
- static int64_t caerPoint1DEventGetTimestamp64 (caerPoint1DEventConst event, caerPoint1DEventPacket↩ Const packet)
- static void caerPoint1DEventSetTimestamp (caerPoint1DEvent event, int32_t timestamp)
- static bool caerPoint1DEventIsValid (caerPoint1DEventConst event)
- static void caerPoint1DEventValidate (caerPoint1DEvent event, caerPoint1DEventPacket packet)
- static void caerPoint1DEventInvalidate (caerPoint1DEvent event, caerPoint1DEventPacket packet)
- static uint8_t caerPoint1DEventGetType (caerPoint1DEventConst event)
- static void caerPoint1DEventSetType (caerPoint1DEvent event, uint8_t type)
- static int8_t caerPoint1DEventGetScale (caerPoint1DEventConst event)
- static void caerPoint1DEventSetScale (caerPoint1DEvent event, int8_t scale)
- static float caerPoint1DEventGetX (caerPoint1DEventConst event)
- static void caerPoint1DEventSetX (caerPoint1DEvent event, float x)

4.17.1 Detailed Description

Point1D Events format definition and handling functions. This contains one dimensional data points as floats, together with support for distinguishing type and scale.

4.17.2 Macro Definition Documentation

4.17.2.1 CAER_POINT1D_CONST_ITERATOR_ALL_START

```
#define CAER_POINT1D_CONST_ITERATOR_ALL_START(
              POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter = 0;                                   \
        caerPoint1DIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(POINT1D_PACKET)->packetHeader); \
        caerPoint1DIteratorCounter++) {
      \
       caerPoint1DEventConst caerPoint1DIteratorElement
      \
           = caerPoint1DEventPacketGetEventConst(POINT1D_PACKET,
     caerPoint1DIteratorCounter);
```

Const-Iterator over all Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint1DIteratorElement' variable of type caer←Point1DEventConst.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.2 CAER_POINT1D_CONST_ITERATOR_VALID_START

```
#define CAER_POINT1D_CONST_ITERATOR_VALID_START(
              POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter = 0;                                   \
        caerPoint1DIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(POINT1D_PACKET)->packetHeader); \
        caerPoint1DIteratorCounter++) {
      \
       caerPoint1DEventConst caerPoint1DIteratorElement
      \
           = caerPoint1DEventPacketGetEventConst(POINT1D_PACKET,
     caerPoint1DIteratorCounter);                  \
       if (!caerPoint1DEventIsValid(caerPoint1DIteratorElement)) {
                                \
           continue;
      \
       }
```

Const-Iterator over only the valid Point1D events in a packet. Returns the current index in the 'caerPoint1DIterator←Counter' variable of type 'int32_t' and the current read-only event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEventConst.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.3 CAER_POINT1D_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT1D_CONST_REVERSE_ITERATOR_ALL_START(
              POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter                                             \
        = caerEventPacketHeaderGetEventNumber(&(POINT1D_PACKET)->
     packetHeader) - 1; \
        caerPoint1DIteratorCounter >= 0; caerPoint1DIteratorCounter--) {               \
       caerPoint1DEventConst caerPoint1DIteratorElement                                \
           = caerPoint1DEventPacketGetEventConst(POINT1D_PACKET,
     caerPoint1DIteratorCounter);
```

Const-Reverse iterator over all Point1D events in a packet. Returns the current index in the 'caerPoint1DIterator← Counter' variable of type 'int32_t' and the current read-only event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEventConst.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.4 CAER_POINT1D_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT1D_CONST_REVERSE_ITERATOR_VALID_START(
              POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter                                                       \
        = caerEventPacketHeaderGetEventNumber(&(POINT1D_PACKET)->
     packetHeader) - 1;           \
        caerPoint1DIteratorCounter >= 0; caerPoint1DIteratorCounter--) {                         \
       caerPoint1DEventConst caerPoint1DIteratorElement                                          \
           = caerPoint1DEventPacketGetEventConst(POINT1D_PACKET,
     caerPoint1DIteratorCounter); \
       if (!caerPoint1DEventIsValid(caerPoint1DIteratorElement)) {
                \
           continue;                                                                             \
       }
```

Const-Reverse iterator over only the valid Point1D events in a packet. Returns the current index in the 'caer← Point1DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEventConst.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.5 CAER_POINT1D_ITERATOR_ALL_END

```
#define CAER_POINT1D_ITERATOR_ALL_END }
```

Iterator close statement.

4.17.2.6 CAER_POINT1D_ITERATOR_ALL_START

```
#define CAER_POINT1D_ITERATOR_ALL_START(
              POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter = 0;                                                  \
        caerPoint1DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT1D_PACKET)->packetHeader); \
        caerPoint1DIteratorCounter++) {
       \
       caerPoint1DEvent caerPoint1DIteratorElement
       \
           = caerPoint1DEventPacketGetEvent(POINT1D_PACKET,
      caerPoint1DIteratorCounter);
```

Iterator over all Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEvent.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.7 CAER_POINT1D_ITERATOR_VALID_END

```
#define CAER_POINT1D_ITERATOR_VALID_END }
```

Iterator close statement.

4.17.2.8 CAER_POINT1D_ITERATOR_VALID_START

```
#define CAER_POINT1D_ITERATOR_VALID_START(
              POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter = 0;                                                  \
        caerPoint1DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT1D_PACKET)->packetHeader); \
        caerPoint1DIteratorCounter++) {
       \
       caerPoint1DEvent caerPoint1DIteratorElement
       \
           = caerPoint1DEventPacketGetEvent(POINT1D_PACKET,
      caerPoint1DIteratorCounter);                   \
        if (!caerPoint1DEventIsValid(caerPoint1DIteratorElement)) {
                               \
            continue;
       \
        }
```

Iterator over only the valid Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEvent.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.9 CAER_POINT1D_REVERSE_ITERATOR_ALL_END

```
#define CAER_POINT1D_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.17.2.10 CAER_POINT1D_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT1D_REVERSE_ITERATOR_ALL_START(
              POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter                                        \
        = caerEventPacketHeaderGetEventNumber(&(POINT1D_PACKET)->
      packetHeader) - 1; \
         caerPoint1DIteratorCounter >= 0; caerPoint1DIteratorCounter--) {            \
        caerPoint1DEvent caerPoint1DIteratorElement                                  \
            = caerPoint1DEventPacketGetEvent(POINT1D_PACKET,
      caerPoint1DIteratorCounter);
```

Reverse iterator over all Point1D events in a packet. Returns the current index in the 'caerPoint1DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEvent.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.11 CAER_POINT1D_REVERSE_ITERATOR_VALID_END

```
#define CAER_POINT1D_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.17.2.12 CAER_POINT1D_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT1D_REVERSE_ITERATOR_VALID_START(
              POINT1D_PACKET )
```

Value:

```
for (int32_t caerPoint1DIteratorCounter                                             \
        = caerEventPacketHeaderGetEventNumber(&(POINT1D_PACKET)->
      packetHeader) - 1;      \
         caerPoint1DIteratorCounter >= 0; caerPoint1DIteratorCounter--) {                 \
        caerPoint1DEvent caerPoint1DIteratorElement                                       \
            = caerPoint1DEventPacketGetEvent(POINT1D_PACKET,
      caerPoint1DIteratorCounter); \
        if (!caerPoint1DEventIsValid(caerPoint1DIteratorElement)) {
            \
            continue;                                                                     \
        }
```

Reverse iterator over only the valid Point1D events in a packet. Returns the current index in the 'caerPoint1D↩ IteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint1DIteratorElement' variable of type caerPoint1DEvent.

POINT1D_PACKET: a valid Point1DEventPacket pointer. Cannot be NULL.

4.17.2.13 POINT1D_SCALE_MASK

```
#define POINT1D_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Point1D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.17.2.14 POINT1D_SCALE_SHIFT

```
#define POINT1D_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Point1D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.17.2.15 POINT1D_TYPE_MASK

```
#define POINT1D_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Point1D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.17.2.16 POINT1D_TYPE_SHIFT

```
#define POINT1D_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Point1D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.17.3 Typedef Documentation

4.17.3.1 caerPoint1DEvent

```
typedef struct caer_point1d_event* caerPoint1DEvent
```

Type for pointer to Point1D event data structure.

4.17.3.2 caerPoint1DEventPacket

```
typedef struct caer_point1d_event_packet* caerPoint1DEventPacket
```

Type for pointer to Point1D event packet data structure.

4.17.4 Function Documentation

4.17.4.1 caerPoint1DEventGetScale()

```
static int8_t caerPoint1DEventGetScale (
             caerPoint1DEventConst event )  [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.

Returns

the Point1D measurement scale.

4.17.4.2 caerPoint1DEventGetTimestamp()

```
static int32_t caerPoint1DEventGetTimestamp (
            caerPoint1DEventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.17.4.3 caerPoint1DEventGetTimestamp64()

```
static int64_t caerPoint1DEventGetTimestamp64 (
            caerPoint1DEventConst event,
            caerPoint1DEventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
packet	the Point1DEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.17.4.4 caerPoint1DEventGetType()

```
static uint8_t caerPoint1DEventGetType (
            caerPoint1DEventConst event )  [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.

Returns

the Point1D measurement type.

4.17.4.5 caerPoint1DEventGetX()

```
static float caerPoint1DEventGetX (
            caerPoint1DEventConst event )  [inline], [static]
```

Get the X axis measurement.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.

Returns

X axis measurement.

4.17.4.6 caerPoint1DEventInvalidate()

```
static void caerPoint1DEventInvalidate (
            caerPoint1DEvent event,
            caerPoint1DEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
packet	the Point1DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.17.4.7 caerPoint1DEventIsValid()

```
static bool caerPoint1DEventIsValid (
            caerPoint1DEventConst event )  [inline], [static]
```

Check if this Point1D event is valid.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.17.4.8 caerPoint1DEventPacketAllocate()

```
static caerPoint1DEventPacket caerPoint1DEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )  [inline], [static]
```

Allocate a new Point1D events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid Point1DEventPacket handle or NULL on error.

4.17.4.9 caerPoint1DEventPacketFromPacketHeader()

```
static caerPoint1DEventPacket caerPoint1DEventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into a Point1D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.17.4.10 caerPoint1DEventPacketFromPacketHeaderConst()

```
static caerPoint1DEventPacketConst caerPoint1DEventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only Point1D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.17.4.11 caerPoint1DEventPacketGetEvent()

```
static caerPoint1DEvent caerPoint1DEventPacketGetEvent (
            caerPoint1DEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the Point1D event at the given index from the event packet.

Parameters

packet	a valid Point1DEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Point1D event. NULL on error.

4.17.4.12 caerPoint1DEventPacketGetEventConst()

```
static caerPoint1DEventConst caerPoint1DEventPacketGetEventConst (
            caerPoint1DEventPacketConst packet,
            int32_t n )  [inline], [static]
```

Get the Point1D event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid Point1DEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only Point1D event. NULL on error.

4.17.4.13 caerPoint1DEventSetScale()

```
static void caerPoint1DEventSetScale (
            caerPoint1DEvent event,
            int8_t scale )  [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
scale	the Point1D measurement scale.

4.17.4.14 caerPoint1DEventSetTimestamp()

```
static void caerPoint1DEventSetTimestamp (
            caerPoint1DEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.17.4.15 caerPoint1DEventSetType()

```
static void caerPoint1DEventSetType (
            caerPoint1DEvent event,
            uint8_t type )  [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
type	the Point1D measurement type.

4.17.4.16 caerPoint1DEventSetX()

```
static void caerPoint1DEventSetX (
            caerPoint1DEvent event,
            float x )  [inline], [static]
```

Set the X axis measurement.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
x	X axis measurement.

4.17.4.17 caerPoint1DEventValidate()

```
static void caerPoint1DEventValidate (
            caerPoint1DEvent event,
            caerPoint1DEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid Point1DEvent pointer. Cannot be NULL.
packet	the Point1DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.17.4.18 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_point1d_event { uint32_t info;float x;int32_t timestamp;}  )
```

Point1D event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The one measurement (x) is stored as a float. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.17.4.19 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_point1d_event_packet { struct caer_event_packet_header packetHeader;struct
caer_point1d_event events[];}  )
```

Point1D event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.18 events/point2d.h File Reference

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

Macros

- #define CAER_POINT2D_ITERATOR_ALL_START(POINT2D_PACKET)
- #define CAER_POINT2D_CONST_ITERATOR_ALL_START(POINT2D_PACKET)
- #define CAER_POINT2D_ITERATOR_ALL_END }
- #define CAER_POINT2D_ITERATOR_VALID_START(POINT2D_PACKET)
- #define CAER_POINT2D_CONST_ITERATOR_VALID_START(POINT2D_PACKET)
- #define CAER_POINT2D_ITERATOR_VALID_END }
- #define CAER_POINT2D_REVERSE_ITERATOR_ALL_START(POINT2D_PACKET)
- #define CAER_POINT2D_CONST_REVERSE_ITERATOR_ALL_START(POINT2D_PACKET)
- #define CAER_POINT2D_REVERSE_ITERATOR_ALL_END }
- #define CAER_POINT2D_REVERSE_ITERATOR_VALID_START(POINT2D_PACKET)
- #define CAER_POINT2D_CONST_REVERSE_ITERATOR_VALID_START(POINT2D_PACKET)
- #define CAER_POINT2D_REVERSE_ITERATOR_VALID_END }

- #define POINT2D_TYPE_SHIFT 1
- #define POINT2D_TYPE_MASK 0x0000007F
- #define POINT2D_SCALE_SHIFT 8
- #define POINT2D_SCALE_MASK 0x000000FF

Typedefs

- typedef struct caer_point2d_event ∗ caerPoint2DEvent
- typedef const struct caer_point2d_event ∗ **caerPoint2DEventConst**
- typedef struct caer_point2d_event_packet ∗ caerPoint2DEventPacket
- typedef const struct caer_point2d_event_packet ∗ **caerPoint2DEventPacketConst**

Functions

- PACKED_STRUCT (struct caer_point2d_event { uint32_t info;float x;float y;int32_t timestamp;})
- PACKED_STRUCT (struct caer_point2d_event_packet { struct caer_event_packet_header packet↩ Header;struct caer_point2d_event events[ ];})
- static caerPoint2DEventPacket caerPoint2DEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerPoint2DEventPacket caerPoint2DEventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerPoint2DEventPacketConst caerPoint2DEventPacketFromPacketHeaderConst (caerEventPacket↩ HeaderConst header)
- static caerPoint2DEvent caerPoint2DEventPacketGetEvent (caerPoint2DEventPacket packet, int32_t n)
- static caerPoint2DEventConst caerPoint2DEventPacketGetEventConst (caerPoint2DEventPacketConst packet, int32_t n)
- static int32_t caerPoint2DEventGetTimestamp (caerPoint2DEventConst event)
- static int64_t caerPoint2DEventGetTimestamp64 (caerPoint2DEventConst event, caerPoint2DEventPacket↩ Const packet)
- static void caerPoint2DEventSetTimestamp (caerPoint2DEvent event, int32_t timestamp)
- static bool caerPoint2DEventIsValid (caerPoint2DEventConst event)
- static void caerPoint2DEventValidate (caerPoint2DEvent event, caerPoint2DEventPacket packet)
- static void caerPoint2DEventInvalidate (caerPoint2DEvent event, caerPoint2DEventPacket packet)
- static uint8_t caerPoint2DEventGetType (caerPoint2DEventConst event)
- static void caerPoint2DEventSetType (caerPoint2DEvent event, uint8_t type)
- static int8_t caerPoint2DEventGetScale (caerPoint2DEventConst event)
- static void caerPoint2DEventSetScale (caerPoint2DEvent event, int8_t scale)
- static float caerPoint2DEventGetX (caerPoint2DEventConst event)
- static void caerPoint2DEventSetX (caerPoint2DEvent event, float x)
- static float caerPoint2DEventGetY (caerPoint2DEventConst event)
- static void caerPoint2DEventSetY (caerPoint2DEvent event, float y)

4.18.1 Detailed Description

Point2D Events format definition and handling functions. This contains two dimensional data points as floats, together with support for distinguishing type and scale.

4.18.2 Macro Definition Documentation

4.18.2.1 CAER_POINT2D_CONST_ITERATOR_ALL_START

```
#define CAER_POINT2D_CONST_ITERATOR_ALL_START(
                POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter = 0;                                                    \
         caerPoint2DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT2D_PACKET)->packetHeader); \
         caerPoint2DIteratorCounter++) {
       \
        caerPoint2DEventConst caerPoint2DIteratorElement
       \
            = caerPoint2DEventPacketGetEventConst(POINT2D_PACKET,
      caerPoint2DIteratorCounter);
```

Const-Iterator over all Point2D events in a packet. Returns the current index in the 'caerPoint2DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint2DIteratorElement' variable of type caer←Point2DEventConst.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.2 CAER_POINT2D_CONST_ITERATOR_VALID_START

```
#define CAER_POINT2D_CONST_ITERATOR_VALID_START(
                POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter = 0;                                                    \
         caerPoint2DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT2D_PACKET)->packetHeader); \
         caerPoint2DIteratorCounter++) {
       \
        caerPoint2DEventConst caerPoint2DIteratorElement
       \
            = caerPoint2DEventPacketGetEventConst(POINT2D_PACKET,
      caerPoint2DIteratorCounter);                \
        if (!caerPoint2DEventIsValid(caerPoint2DIteratorElement)) {
                                \
            continue;
       \
        }
```

Const-Iterator over only the valid Point2D events in a packet. Returns the current index in the 'caerPoint2DIterator←Counter' variable of type 'int32_t' and the current read-only event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEventConst.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.3 CAER_POINT2D_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT2D_CONST_REVERSE_ITERATOR_ALL_START(
                POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter                                              \
         = caerEventPacketHeaderGetEventNumber(&(POINT2D_PACKET)->
      packetHeader) - 1; \
         caerPoint2DIteratorCounter >= 0; caerPoint2DIteratorCounter--) {             \
        caerPoint2DEventConst caerPoint2DIteratorElement                              \
            = caerPoint2DEventPacketGetEventConst(POINT2D_PACKET,
      caerPoint2DIteratorCounter);
```

Const-Reverse iterator over all Point2D events in a packet. Returns the current index in the 'caerPoint2DIterator←Counter' variable of type 'int32_t' and the current read-only event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEventConst.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.4 CAER_POINT2D_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT2D_CONST_REVERSE_ITERATOR_VALID_START(
                POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter                                                    \
         = caerEventPacketHeaderGetEventNumber(&(POINT2D_PACKET)->
      packetHeader) - 1;           \
         caerPoint2DIteratorCounter >= 0; caerPoint2DIteratorCounter--) {                      \
        caerPoint2DEventConst caerPoint2DIteratorElement                                       \
            = caerPoint2DEventPacketGetEventConst(POINT2D_PACKET,
      caerPoint2DIteratorCounter); \
        if (!caerPoint2DEventIsValid(caerPoint2DIteratorElement)) {
                 \
            continue;                                                                          \
        }
```

Const-Reverse iterator over only the valid Point2D events in a packet. Returns the current index in the 'caer↵Point2DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEventConst.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.5 CAER_POINT2D_ITERATOR_ALL_END

```
#define CAER_POINT2D_ITERATOR_ALL_END }
```

Iterator close statement.

4.18.2.6 CAER_POINT2D_ITERATOR_ALL_START

```
#define CAER_POINT2D_ITERATOR_ALL_START(
                POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter = 0;                                                          \
         caerPoint2DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT2D_PACKET)->packetHeader); \
         caerPoint2DIteratorCounter++) {
       \
        caerPoint2DEvent caerPoint2DIteratorElement
       \
            = caerPoint2DEventPacketGetEvent(POINT2D_PACKET,
      caerPoint2DIteratorCounter);
```

Iterator over all Point2D events in a packet. Returns the current index in the 'caerPoint2DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEvent.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.7 CAER_POINT2D_ITERATOR_VALID_END

```
#define CAER_POINT2D_ITERATOR_VALID_END }
```

Iterator close statement.

4.18.2.8 CAER_POINT2D_ITERATOR_VALID_START

```
#define CAER_POINT2D_ITERATOR_VALID_START(
              POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter = 0;                                                 \
        caerPoint2DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT2D_PACKET)->packetHeader); \
        caerPoint2DIteratorCounter++) {
       \
       caerPoint2DEvent caerPoint2DIteratorElement
       \
           = caerPoint2DEventPacketGetEvent(POINT2D_PACKET,
      caerPoint2DIteratorCounter);                 \
       if (!caerPoint2DEventIsValid(caerPoint2DIteratorElement)) {
                                    \
           continue;
       \
       }
```

Iterator over only the valid Point2D events in a packet. Returns the current index in the 'caerPoint2DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEvent.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.9 CAER_POINT2D_REVERSE_ITERATOR_ALL_END

```
#define CAER_POINT2D_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.18.2.10 CAER_POINT2D_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT2D_REVERSE_ITERATOR_ALL_START(
              POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter                                         \
        = caerEventPacketHeaderGetEventNumber(&(POINT2D_PACKET)->
      packetHeader) - 1; \
        caerPoint2DIteratorCounter >= 0; caerPoint2DIteratorCounter--) {            \
       caerPoint2DEvent caerPoint2DIteratorElement                                  \
           = caerPoint2DEventPacketGetEvent(POINT2D_PACKET,
      caerPoint2DIteratorCounter);
```

Reverse iterator over all Point2D events in a packet. Returns the current index in the 'caerPoint2DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEvent.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.11 CAER_POINT2D_REVERSE_ITERATOR_VALID_END

```
#define CAER_POINT2D_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.18.2.12 CAER_POINT2D_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT2D_REVERSE_ITERATOR_VALID_START(
              POINT2D_PACKET )
```

Value:

```
for (int32_t caerPoint2DIteratorCounter                                       \
         = caerEventPacketHeaderGetEventNumber(&(POINT2D_PACKET)->
      packetHeader) - 1;      \
         caerPoint2DIteratorCounter >= 0; caerPoint2DIteratorCounter--) {         \
        caerPoint2DEvent caerPoint2DIteratorElement                                \
            = caerPoint2DEventPacketGetEvent(POINT2D_PACKET,
      caerPoint2DIteratorCounter); \
        if (!caerPoint2DEventIsValid(caerPoint2DIteratorElement)) {
            \
            continue;                                                              \
        }
```

Reverse iterator over only the valid Point2D events in a packet. Returns the current index in the 'caerPoint2D↩ IteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint2DIteratorElement' variable of type caerPoint2DEvent.

POINT2D_PACKET: a valid Point2DEventPacket pointer. Cannot be NULL.

4.18.2.13 POINT2D_SCALE_MASK

```
#define POINT2D_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Point2D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.18.2.14 POINT2D_SCALE_SHIFT

```
#define POINT2D_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Point2D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.18.2.15 POINT2D_TYPE_MASK

```
#define POINT2D_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Point2D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.18.2.16 POINT2D_TYPE_SHIFT

```
#define POINT2D_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Point2D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.18.3 Typedef Documentation

4.18.3.1 caerPoint2DEvent

```
typedef struct caer_point2d_event* caerPoint2DEvent
```

Type for pointer to Point2D event data structure.

4.18.3.2 caerPoint2DEventPacket

```
typedef struct caer_point2d_event_packet* caerPoint2DEventPacket
```

Type for pointer to Point2D event packet data structure.

4.18.4 Function Documentation

4.18.4.1 caerPoint2DEventGetScale()

```
static int8_t caerPoint2DEventGetScale (
             caerPoint2DEventConst event )  [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.

Returns

the Point2D measurement scale.

4.18.4.2 caerPoint2DEventGetTimestamp()

```
static int32_t caerPoint2DEventGetTimestamp (
             caerPoint2DEventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.18.4.3 caerPoint2DEventGetTimestamp64()

```
static int64_t caerPoint2DEventGetTimestamp64 (
            caerPoint2DEventConst event,
            caerPoint2DEventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
packet	the Point2DEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.18.4.4 caerPoint2DEventGetType()

```
static uint8_t caerPoint2DEventGetType (
            caerPoint2DEventConst event )  [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.

Returns

the Point2D measurement type.

4.18.4.5 caerPoint2DEventGetX()

```
static float caerPoint2DEventGetX (
             caerPoint2DEventConst event )  [inline], [static]
```

Get the X axis measurement.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.

Returns

X axis measurement.

4.18.4.6 caerPoint2DEventGetY()

```
static float caerPoint2DEventGetY (
             caerPoint2DEventConst event )  [inline], [static]
```

Get the Y axis measurement.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.

Returns

Y axis measurement.

4.18.4.7 caerPoint2DEventInvalidate()

```
static void caerPoint2DEventInvalidate (
             caerPoint2DEvent event,
             caerPoint2DEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
packet	the Point2DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.18.4.8 caerPoint2DEventIsValid()

```
static bool caerPoint2DEventIsValid (
            caerPoint2DEventConst event )  [inline], [static]
```

Check if this Point2D event is valid.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.18.4.9 caerPoint2DEventPacketAllocate()

```
static caerPoint2DEventPacket caerPoint2DEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )  [inline], [static]
```

Allocate a new Point2D events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid Point2DEventPacket handle or NULL on error.

4.18.4.10 caerPoint2DEventPacketFromPacketHeader()

```
static caerPoint2DEventPacket caerPoint2DEventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into a Point2D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.18.4.11 caerPoint2DEventPacketFromPacketHeaderConst()

```
static caerPoint2DEventPacketConst caerPoint2DEventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only Point2D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.18.4.12 caerPoint2DEventPacketGetEvent()

```
static caerPoint2DEvent caerPoint2DEventPacketGetEvent (
            caerPoint2DEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the Point2D event at the given index from the event packet.

Parameters

packet	a valid Point2DEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Point2D event. NULL on error.

4.18.4.13 caerPoint2DEventPacketGetEventConst()

```
static caerPoint2DEventConst caerPoint2DEventPacketGetEventConst (
            caerPoint2DEventPacketConst packet,
            int32_t n ) [inline], [static]
```

Get the Point2D event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid Point2DEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only Point2D event. NULL on error.

4.18.4.14 caerPoint2DEventSetScale()

```
static void caerPoint2DEventSetScale (
            caerPoint2DEvent event,
            int8_t scale ) [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
scale	the Point2D measurement scale.

4.18.4.15 caerPoint2DEventSetTimestamp()

```
static void caerPoint2DEventSetTimestamp (
            caerPoint2DEvent event,
            int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.18.4.16 caerPoint2DEventSetType()

```
static void caerPoint2DEventSetType (
            caerPoint2DEvent event,
            uint8_t type ) [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
type	the Point2D measurement type.

4.18.4.17 caerPoint2DEventSetX()

```
static void caerPoint2DEventSetX (
            caerPoint2DEvent event,
            float x ) [inline], [static]
```

Set the X axis measurement.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
x	X axis measurement.

4.18.4.18 caerPoint2DEventSetY()

```
static void caerPoint2DEventSetY (
            caerPoint2DEvent event,
            float y ) [inline], [static]
```

Set the Y axis measurement.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
y	Y axis measurement.

4.18.4.19 caerPoint2DEventValidate()

```
static void caerPoint2DEventValidate (
            caerPoint2DEvent event,
            caerPoint2DEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid Point2DEvent pointer. Cannot be NULL.
packet	the Point2DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.18.4.20 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_point2d_event { uint32_t info;float x;float y;int32_t timestamp;}  )
```

Point2D event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The two measurements (x, y) are stored as floats. Floats are in IE↩EE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.18.4.21 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_point2d_event_packet { struct caer_event_packet_header packetHeader;struct
caer_point2d_event events[];}  )
```

Point2D event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.19 events/point3d.h File Reference

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

Macros

- #define CAER_POINT3D_ITERATOR_ALL_START(POINT3D_PACKET)
- #define CAER_POINT3D_CONST_ITERATOR_ALL_START(POINT3D_PACKET)
- #define CAER_POINT3D_ITERATOR_ALL_END }
- #define CAER_POINT3D_ITERATOR_VALID_START(POINT3D_PACKET)
- #define CAER_POINT3D_CONST_ITERATOR_VALID_START(POINT3D_PACKET)
- #define CAER_POINT3D_ITERATOR_VALID_END }
- #define CAER_POINT3D_REVERSE_ITERATOR_ALL_START(POINT3D_PACKET)
- #define CAER_POINT3D_CONST_REVERSE_ITERATOR_ALL_START(POINT3D_PACKET)
- #define CAER_POINT3D_REVERSE_ITERATOR_ALL_END }
- #define CAER_POINT3D_REVERSE_ITERATOR_VALID_START(POINT3D_PACKET)
- #define CAER_POINT3D_CONST_REVERSE_ITERATOR_VALID_START(POINT3D_PACKET)
- #define CAER_POINT3D_REVERSE_ITERATOR_VALID_END }

- #define POINT3D_TYPE_SHIFT 1
- #define POINT3D_TYPE_MASK 0x0000007F
- #define POINT3D_SCALE_SHIFT 8
- #define POINT3D_SCALE_MASK 0x000000FF

Typedefs

- typedef struct caer_point3d_event ∗ caerPoint3DEvent
- typedef const struct caer_point3d_event ∗ **caerPoint3DEventConst**
- typedef struct caer_point3d_event_packet ∗ caerPoint3DEventPacket
- typedef const struct caer_point3d_event_packet ∗ **caerPoint3DEventPacketConst**

Functions

- PACKED_STRUCT (struct caer_point3d_event { uint32_t info;float x;float y;float z;int32_t timestamp;})
- PACKED_STRUCT (struct caer_point3d_event_packet { struct caer_event_packet_header packet←Header;struct caer_point3d_event events[ ];})
- static caerPoint3DEventPacket caerPoint3DEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerPoint3DEventPacket caerPoint3DEventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerPoint3DEventPacketConst caerPoint3DEventPacketFromPacketHeaderConst (caerEventPacket←HeaderConst header)
- static caerPoint3DEvent caerPoint3DEventPacketGetEvent (caerPoint3DEventPacket packet, int32_t n)
- static caerPoint3DEventConst caerPoint3DEventPacketGetEventConst (caerPoint3DEventPacketConst packet, int32_t n)
- static int32_t caerPoint3DEventGetTimestamp (caerPoint3DEventConst event)
- static int64_t caerPoint3DEventGetTimestamp64 (caerPoint3DEventConst event, caerPoint3DEventPacket←Const packet)
- static void caerPoint3DEventSetTimestamp (caerPoint3DEvent event, int32_t timestamp)
- static bool caerPoint3DEventIsValid (caerPoint3DEventConst event)
- static void caerPoint3DEventValidate (caerPoint3DEvent event, caerPoint3DEventPacket packet)
- static void caerPoint3DEventInvalidate (caerPoint3DEvent event, caerPoint3DEventPacket packet)
- static uint8_t caerPoint3DEventGetType (caerPoint3DEventConst event)
- static void caerPoint3DEventSetType (caerPoint3DEvent event, uint8_t type)

- static int8_t caerPoint3DEventGetScale (caerPoint3DEventConst event)
- static void caerPoint3DEventSetScale (caerPoint3DEvent event, int8_t scale)
- static float caerPoint3DEventGetX (caerPoint3DEventConst event)
- static void caerPoint3DEventSetX (caerPoint3DEvent event, float x)
- static float caerPoint3DEventGetY (caerPoint3DEventConst event)
- static void caerPoint3DEventSetY (caerPoint3DEvent event, float y)
- static float caerPoint3DEventGetZ (caerPoint3DEventConst event)
- static void caerPoint3DEventSetZ (caerPoint3DEvent event, float z)

4.19.1 Detailed Description

Point3D Events format definition and handling functions. This contains three dimensional data points as floats, together with support for distinguishing type and scale.

4.19.2 Macro Definition Documentation

4.19.2.1 CAER_POINT3D_CONST_ITERATOR_ALL_START

```
#define CAER_POINT3D_CONST_ITERATOR_ALL_START(
                POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter = 0;                                          \
         caerPoint3DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT3D_PACKET)->packetHeader); \
         caerPoint3DIteratorCounter++) {
       \
        caerPoint3DEventConst caerPoint3DIteratorElement
       \
            = caerPoint3DEventPacketGetEventConst(POINT3D_PACKET,
      caerPoint3DIteratorCounter);
```

Const-Iterator over all Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint3DIteratorElement' variable of type caer↩ Point3DEventConst.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.2 CAER_POINT3D_CONST_ITERATOR_VALID_START

```
#define CAER_POINT3D_CONST_ITERATOR_VALID_START(
                POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter = 0;                                                 \
         caerPoint3DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT3D_PACKET)->packetHeader); \
         caerPoint3DIteratorCounter++) {
       \
        caerPoint3DEventConst caerPoint3DIteratorElement
       \
            = caerPoint3DEventPacketGetEventConst(POINT3D_PACKET,
      caerPoint3DIteratorCounter);                \
        if (!caerPoint3DEventIsValid(caerPoint3DIteratorElement)) {
                                    \
            continue;
       \
        }
```

Const-Iterator over only the valid Point3D events in a packet. Returns the current index in the 'caerPoint3DIterator←Counter' variable of type 'int32_t' and the current read-only event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEventConst.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.3 CAER_POINT3D_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT3D_CONST_REVERSE_ITERATOR_ALL_START(
                POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter                                          \
         = caerEventPacketHeaderGetEventNumber(&(POINT3D_PACKET)->
      packetHeader) - 1; \
         caerPoint3DIteratorCounter >= 0; caerPoint3DIteratorCounter--) {            \
        caerPoint3DEventConst caerPoint3DIteratorElement                             \
            = caerPoint3DEventPacketGetEventConst(POINT3D_PACKET,
      caerPoint3DIteratorCounter);
```

Const-Reverse iterator over all Point3D events in a packet. Returns the current index in the 'caerPoint3DIterator←Counter' variable of type 'int32_t' and the current read-only event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEventConst.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.4 CAER_POINT3D_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT3D_CONST_REVERSE_ITERATOR_VALID_START(
                POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter                                                    \
         = caerEventPacketHeaderGetEventNumber(&(POINT3D_PACKET)->
      packetHeader) - 1;           \
         caerPoint3DIteratorCounter >= 0; caerPoint3DIteratorCounter--) {                      \
        caerPoint3DEventConst caerPoint3DIteratorElement                                       \
            = caerPoint3DEventPacketGetEventConst(POINT3D_PACKET,
      caerPoint3DIteratorCounter); \
        if (!caerPoint3DEventIsValid(caerPoint3DIteratorElement)) {
                  \
            continue;                                                                          \
        }
```

Const-Reverse iterator over only the valid Point3D events in a packet. Returns the current index in the 'caer←Point3DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEventConst.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.5 CAER_POINT3D_ITERATOR_ALL_END

```
#define CAER_POINT3D_ITERATOR_ALL_END }
```

Iterator close statement.

4.19.2.6 CAER_POINT3D_ITERATOR_ALL_START

```
#define CAER_POINT3D_ITERATOR_ALL_START(
              POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter = 0;                                                   \
        caerPoint3DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT3D_PACKET)->packetHeader); \
        caerPoint3DIteratorCounter++) {
       \
       caerPoint3DEvent caerPoint3DIteratorElement
       \
           = caerPoint3DEventPacketGetEvent(POINT3D_PACKET,
      caerPoint3DIteratorCounter);
```

Iterator over all Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEvent.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.7 CAER_POINT3D_ITERATOR_VALID_END

```
#define CAER_POINT3D_ITERATOR_VALID_END }
```

Iterator close statement.

4.19.2.8 CAER_POINT3D_ITERATOR_VALID_START

```
#define CAER_POINT3D_ITERATOR_VALID_START(
              POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter = 0;                                                   \
        caerPoint3DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT3D_PACKET)->packetHeader); \
        caerPoint3DIteratorCounter++) {
       \
       caerPoint3DEvent caerPoint3DIteratorElement
       \
           = caerPoint3DEventPacketGetEvent(POINT3D_PACKET,
      caerPoint3DIteratorCounter);                      \
         if (!caerPoint3DEventIsValid(caerPoint3DIteratorElement)) {
                                       \
             continue;
       \
        }
```

Iterator over only the valid Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEvent.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.9 CAER_POINT3D_REVERSE_ITERATOR_ALL_END

```
#define CAER_POINT3D_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.19.2.10 CAER_POINT3D_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT3D_REVERSE_ITERATOR_ALL_START(
              POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter                                    \
        = caerEventPacketHeaderGetEventNumber(&(POINT3D_PACKET)->
     packetHeader) - 1; \
        caerPoint3DIteratorCounter >= 0; caerPoint3DIteratorCounter--) {          \
       caerPoint3DEvent caerPoint3DIteratorElement                                \
           = caerPoint3DEventPacketGetEvent(POINT3D_PACKET,
     caerPoint3DIteratorCounter);
```

Reverse iterator over all Point3D events in a packet. Returns the current index in the 'caerPoint3DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEvent.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.11 CAER_POINT3D_REVERSE_ITERATOR_VALID_END

```
#define CAER_POINT3D_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.19.2.12 CAER_POINT3D_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT3D_REVERSE_ITERATOR_VALID_START(
              POINT3D_PACKET )
```

Value:

```
for (int32_t caerPoint3DIteratorCounter                                         \
        = caerEventPacketHeaderGetEventNumber(&(POINT3D_PACKET)->
     packetHeader) - 1;      \
        caerPoint3DIteratorCounter >= 0; caerPoint3DIteratorCounter--) {               \
       caerPoint3DEvent caerPoint3DIteratorElement                                     \
           = caerPoint3DEventPacketGetEvent(POINT3D_PACKET,
     caerPoint3DIteratorCounter); \
       if (!caerPoint3DEventIsValid(caerPoint3DIteratorElement)) {
           \
           continue;                                                                   \
       }
```

Reverse iterator over only the valid Point3D events in a packet. Returns the current index in the 'caerPoint3D↩ IteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint3DIteratorElement' variable of type caerPoint3DEvent.

POINT3D_PACKET: a valid Point3DEventPacket pointer. Cannot be NULL.

4.19.2.13 POINT3D_SCALE_MASK

```
#define POINT3D_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Point3D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.19.2.14 POINT3D_SCALE_SHIFT

```
#define POINT3D_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Point3D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.19.2.15 POINT3D_TYPE_MASK

```
#define POINT3D_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Point3D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.19.2.16 POINT3D_TYPE_SHIFT

```
#define POINT3D_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Point3D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.19.3 Typedef Documentation

4.19.3.1 caerPoint3DEvent

```
typedef struct caer_point3d_event* caerPoint3DEvent
```

Type for pointer to Point3D event data structure.

4.19.3.2 caerPoint3DEventPacket

```
typedef struct caer_point3d_event_packet* caerPoint3DEventPacket
```

Type for pointer to Point3D event packet data structure.

4.19.4 Function Documentation

4.19.4.1 caerPoint3DEventGetScale()

```
static int8_t caerPoint3DEventGetScale (
            caerPoint3DEventConst event )  [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

the Point3D measurement scale.

4.19.4.2 caerPoint3DEventGetTimestamp()

```
static int32_t caerPoint3DEventGetTimestamp (
            caerPoint3DEventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.19.4.3 caerPoint3DEventGetTimestamp64()

```
static int64_t caerPoint3DEventGetTimestamp64 (
            caerPoint3DEventConst event,
            caerPoint3DEventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
packet	the Point3DEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.19.4.4 caerPoint3DEventGetType()

```
static uint8_t caerPoint3DEventGetType (
            caerPoint3DEventConst event )  [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

the Point3D measurement type.

4.19.4.5 caerPoint3DEventGetX()

```
static float caerPoint3DEventGetX (
            caerPoint3DEventConst event )  [inline], [static]
```

Get the X axis measurement.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

X axis measurement.

4.19.4.6 caerPoint3DEventGetY()

```
static float caerPoint3DEventGetY (
             caerPoint3DEventConst event )  [inline], [static]
```

Get the Y axis measurement.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

Y axis measurement.

4.19.4.7 caerPoint3DEventGetZ()

```
static float caerPoint3DEventGetZ (
             caerPoint3DEventConst event )  [inline], [static]
```

Get the Z axis measurement.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

Z axis measurement.

4.19.4.8 caerPoint3DEventInvalidate()

```
static void caerPoint3DEventInvalidate (
             caerPoint3DEvent event,
             caerPoint3DEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
packet	the Point3DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.19.4.9 caerPoint3DEventIsValid()

```
static bool caerPoint3DEventIsValid (
            caerPoint3DEventConst event )  [inline], [static]
```

Check if this Point3D event is valid.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.19.4.10 caerPoint3DEventPacketAllocate()

```
static caerPoint3DEventPacket caerPoint3DEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )  [inline], [static]
```

Allocate a new Point3D events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid Point3DEventPacket handle or NULL on error.

4.19.4.11 caerPoint3DEventPacketFromPacketHeader()

```
static caerPoint3DEventPacket caerPoint3DEventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into a Point3D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.19.4.12 caerPoint3DEventPacketFromPacketHeaderConst()

```
static caerPoint3DEventPacketConst caerPoint3DEventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only Point3D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.19.4.13 caerPoint3DEventPacketGetEvent()

```
static caerPoint3DEvent caerPoint3DEventPacketGetEvent (
            caerPoint3DEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the Point3D event at the given index from the event packet.

Parameters

packet	a valid Point3DEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Point3D event. NULL on error.

4.19.4.14 caerPoint3DEventPacketGetEventConst()

```
static caerPoint3DEventConst caerPoint3DEventPacketGetEventConst (
            caerPoint3DEventPacketConst packet,
            int32_t n ) [inline], [static]
```

Get the Point3D event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid Point3DEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only Point3D event. NULL on error.

4.19.4.15 caerPoint3DEventSetScale()

```
static void caerPoint3DEventSetScale (
            caerPoint3DEvent event,
            int8_t scale ) [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
scale	the Point3D measurement scale.

4.19.4.16 caerPoint3DEventSetTimestamp()

```
static void caerPoint3DEventSetTimestamp (
            caerPoint3DEvent event,
            int32_t timestamp ) [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.19.4.17 caerPoint3DEventSetType()

```
static void caerPoint3DEventSetType (
            caerPoint3DEvent event,
            uint8_t type )  [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
type	the Point3D measurement type.

4.19.4.18 caerPoint3DEventSetX()

```
static void caerPoint3DEventSetX (
            caerPoint3DEvent event,
            float x )  [inline], [static]
```

Set the X axis measurement.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
x	X axis measurement.

4.19.4.19 caerPoint3DEventSetY()

```
static void caerPoint3DEventSetY (
            caerPoint3DEvent event,
            float y )  [inline], [static]
```

Set the Y axis measurement.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
y	Y axis measurement.

4.19.4.20 caerPoint3DEventSetZ()

```
static void caerPoint3DEventSetZ (
            caerPoint3DEvent event,
            float z )  [inline], [static]
```

Set the Z axis measurement.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
z	Z axis measurement.

4.19.4.21 caerPoint3DEventValidate()

```
static void caerPoint3DEventValidate (
            caerPoint3DEvent event,
            caerPoint3DEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid Point3DEvent pointer. Cannot be NULL.
packet	the Point3DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.19.4.22 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_point3d_event { uint32_t info;float x;float y;float z;int32_t timestamp;}
)
```

Point3D event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The three measurements (x, y, z) are stored as floats. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.19.4.23 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_point3d_event_packet { struct caer_event_packet_header packetHeader;struct
caer_point3d_event events[];}  )
```

Point3D event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.20 events/point4d.h File Reference

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

Macros

- #define CAER_POINT4D_ITERATOR_ALL_START(POINT4D_PACKET)
- #define CAER_POINT4D_CONST_ITERATOR_ALL_START(POINT4D_PACKET)
- #define CAER_POINT4D_ITERATOR_ALL_END }
- #define CAER_POINT4D_ITERATOR_VALID_START(POINT4D_PACKET)
- #define CAER_POINT4D_CONST_ITERATOR_VALID_START(POINT4D_PACKET)
- #define CAER_POINT4D_ITERATOR_VALID_END }
- #define CAER_POINT4D_REVERSE_ITERATOR_ALL_START(POINT4D_PACKET)
- #define CAER_POINT4D_CONST_REVERSE_ITERATOR_ALL_START(POINT4D_PACKET)
- #define CAER_POINT4D_REVERSE_ITERATOR_ALL_END }
- #define CAER_POINT4D_REVERSE_ITERATOR_VALID_START(POINT4D_PACKET)
- #define CAER_POINT4D_CONST_REVERSE_ITERATOR_VALID_START(POINT4D_PACKET)
- #define CAER_POINT4D_REVERSE_ITERATOR_VALID_END }

- #define POINT4D_TYPE_SHIFT 1
- #define POINT4D_TYPE_MASK 0x0000007F
- #define POINT4D_SCALE_SHIFT 8
- #define POINT4D_SCALE_MASK 0x000000FF

Typedefs

- typedef struct caer_point4d_event ∗ caerPoint4DEvent
- typedef const struct caer_point4d_event ∗ **caerPoint4DEventConst**
- typedef struct caer_point4d_event_packet ∗ caerPoint4DEventPacket
- typedef const struct caer_point4d_event_packet ∗ **caerPoint4DEventPacketConst**

Functions

- PACKED_STRUCT (struct caer_point4d_event { uint32_t info;float x;float y;float z;float w;int32_t timestamp;})
- PACKED_STRUCT (struct caer_point4d_event_packet { struct caer_event_packet_header packet←Header;struct caer_point4d_event events[ ];})
- static caerPoint4DEventPacket caerPoint4DEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerPoint4DEventPacket caerPoint4DEventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerPoint4DEventPacketConst caerPoint4DEventPacketFromPacketHeaderConst (caerEventPacket←HeaderConst header)
- static caerPoint4DEvent caerPoint4DEventPacketGetEvent (caerPoint4DEventPacket packet, int32_t n)
- static caerPoint4DEventConst caerPoint4DEventPacketGetEventConst (caerPoint4DEventPacketConst packet, int32_t n)
- static int32_t caerPoint4DEventGetTimestamp (caerPoint4DEventConst event)

- static int64_t caerPoint4DEventGetTimestamp64 (caerPoint4DEventConst event, caerPoint4DEventPacket←Const packet)
- static void caerPoint4DEventSetTimestamp (caerPoint4DEvent event, int32_t timestamp)
- static bool caerPoint4DEventIsValid (caerPoint4DEventConst event)
- static void caerPoint4DEventValidate (caerPoint4DEvent event, caerPoint4DEventPacket packet)
- static void caerPoint4DEventInvalidate (caerPoint4DEvent event, caerPoint4DEventPacket packet)
- static uint8_t caerPoint4DEventGetType (caerPoint4DEventConst event)
- static void caerPoint4DEventSetType (caerPoint4DEvent event, uint8_t type)
- static int8_t caerPoint4DEventGetScale (caerPoint4DEventConst event)
- static void caerPoint4DEventSetScale (caerPoint4DEvent event, int8_t scale)
- static float caerPoint4DEventGetX (caerPoint4DEventConst event)
- static void caerPoint4DEventSetX (caerPoint4DEvent event, float x)
- static float caerPoint4DEventGetY (caerPoint4DEventConst event)
- static void caerPoint4DEventSetY (caerPoint4DEvent event, float y)
- static float caerPoint4DEventGetZ (caerPoint4DEventConst event)
- static void caerPoint4DEventSetZ (caerPoint4DEvent event, float z)
- static float caerPoint4DEventGetW (caerPoint4DEventConst event)
- static void caerPoint4DEventSetW (caerPoint4DEvent event, float w)

4.20.1 Detailed Description

Point4D Events format definition and handling functions. This contains four dimensional data points as floats, together with support for distinguishing type and scale. Useful for homogeneous coordinates for example.

4.20.2 Macro Definition Documentation

4.20.2.1 CAER_POINT4D_CONST_ITERATOR_ALL_START

```
#define CAER_POINT4D_CONST_ITERATOR_ALL_START(
              POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter = 0;                                              \
         caerPoint4DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT4D_PACKET)->packetHeader); \
         caerPoint4DIteratorCounter++) {
       \
        caerPoint4DEventConst caerPoint4DIteratorElement
       \
            = caerPoint4DEventPacketGetEventConst(POINT4D_PACKET,
      caerPoint4DIteratorCounter);
```

Const-Iterator over all Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint4DIteratorElement' variable of type caer←Point4DEventConst.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.2 CAER_POINT4D_CONST_ITERATOR_VALID_START

```
#define CAER_POINT4D_CONST_ITERATOR_VALID_START(
              POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter = 0;                                                  \
        caerPoint4DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT4D_PACKET)->packetHeader); \
        caerPoint4DIteratorCounter++) {
       \
       caerPoint4DEventConst caerPoint4DIteratorElement
       \
           = caerPoint4DEventPacketGetEventConst(POINT4D_PACKET,
      caerPoint4DIteratorCounter);                \
       if (!caerPoint4DEventIsValid(caerPoint4DIteratorElement)) {
                              \
           continue;
       \
       }
```

Const-Iterator over only the valid Point4D events in a packet. Returns the current index in the 'caerPoint4DIterator↩ Counter' variable of type 'int32_t' and the current read-only event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEventConst.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.3 CAER_POINT4D_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT4D_CONST_REVERSE_ITERATOR_ALL_START(
              POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter                                        \
        = caerEventPacketHeaderGetEventNumber(&(POINT4D_PACKET)->
      packetHeader) - 1; \
        caerPoint4DIteratorCounter >= 0; caerPoint4DIteratorCounter--) {           \
       caerPoint4DEventConst caerPoint4DIteratorElement                             \
           = caerPoint4DEventPacketGetEventConst(POINT4D_PACKET,
      caerPoint4DIteratorCounter);
```

Const-Reverse iterator over all Point4D events in a packet. Returns the current index in the 'caerPoint4DIterator↩ Counter' variable of type 'int32_t' and the current read-only event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEventConst.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.4 CAER_POINT4D_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT4D_CONST_REVERSE_ITERATOR_VALID_START(
              POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter                                                \
        = caerEventPacketHeaderGetEventNumber(&(POINT4D_PACKET)->
      packetHeader) - 1;          \
        caerPoint4DIteratorCounter >= 0; caerPoint4DIteratorCounter--) {                   \
       caerPoint4DEventConst caerPoint4DIteratorElement                                     \
           = caerPoint4DEventPacketGetEventConst(POINT4D_PACKET,
      caerPoint4DIteratorCounter); \
       if (!caerPoint4DEventIsValid(caerPoint4DIteratorElement)) {
                \
           continue;                                                                         \
       }
```

Const-Reverse iterator over only the valid Point4D events in a packet. Returns the current index in the 'caer↩ Point4DIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEventConst.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.5 CAER_POINT4D_ITERATOR_ALL_END

```
#define CAER_POINT4D_ITERATOR_ALL_END }
```

Iterator close statement.

4.20.2.6 CAER_POINT4D_ITERATOR_ALL_START

```
#define CAER_POINT4D_ITERATOR_ALL_START(
                POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter = 0;                                                    \
         caerPoint4DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT4D_PACKET)->packetHeader); \
         caerPoint4DIteratorCounter++) {
       \
        caerPoint4DEvent caerPoint4DIteratorElement
       \
            = caerPoint4DEventPacketGetEvent(POINT4D_PACKET,
      caerPoint4DIteratorCounter);
```

Iterator over all Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEvent.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.7 CAER_POINT4D_ITERATOR_VALID_END

```
#define CAER_POINT4D_ITERATOR_VALID_END }
```

Iterator close statement.

4.20.2.8 CAER_POINT4D_ITERATOR_VALID_START

```
#define CAER_POINT4D_ITERATOR_VALID_START(
                POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter = 0;                                                    \
         caerPoint4DIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POINT4D_PACKET)->packetHeader); \
         caerPoint4DIteratorCounter++) {
       \
        caerPoint4DEvent caerPoint4DIteratorElement
       \
            = caerPoint4DEventPacketGetEvent(POINT4D_PACKET,
      caerPoint4DIteratorCounter);                       \
        if (!caerPoint4DEventIsValid(caerPoint4DIteratorElement)) {
                                  \
            continue;
       \
        }
```

Iterator over only the valid Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEvent.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.9 CAER_POINT4D_REVERSE_ITERATOR_ALL_END

```
#define CAER_POINT4D_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.20.2.10 CAER_POINT4D_REVERSE_ITERATOR_ALL_START

```
#define CAER_POINT4D_REVERSE_ITERATOR_ALL_START(
                POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter                                     \
         = caerEventPacketHeaderGetEventNumber(&(POINT4D_PACKET)->
      packetHeader) - 1; \
         caerPoint4DIteratorCounter >= 0; caerPoint4DIteratorCounter--) {        \
        caerPoint4DEvent caerPoint4DIteratorElement                              \
            = caerPoint4DEventPacketGetEvent(POINT4D_PACKET,
      caerPoint4DIteratorCounter);
```

Reverse iterator over all Point4D events in a packet. Returns the current index in the 'caerPoint4DIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEvent.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.11 CAER_POINT4D_REVERSE_ITERATOR_VALID_END

```
#define CAER_POINT4D_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.20.2.12 CAER_POINT4D_REVERSE_ITERATOR_VALID_START

```
#define CAER_POINT4D_REVERSE_ITERATOR_VALID_START(
                POINT4D_PACKET )
```

Value:

```
for (int32_t caerPoint4DIteratorCounter                                          \
         = caerEventPacketHeaderGetEventNumber(&(POINT4D_PACKET)->
      packetHeader) - 1;      \
         caerPoint4DIteratorCounter >= 0; caerPoint4DIteratorCounter--) {             \
        caerPoint4DEvent caerPoint4DIteratorElement                                   \
            = caerPoint4DEventPacketGetEvent(POINT4D_PACKET,
      caerPoint4DIteratorCounter); \
        if (!caerPoint4DEventIsValid(caerPoint4DIteratorElement)) {
            \
            continue;                                                                 \
        }
```

Reverse iterator over only the valid Point4D events in a packet. Returns the current index in the 'caerPoint4D↩IteratorCounter' variable of type 'int32_t' and the current event in the 'caerPoint4DIteratorElement' variable of type caerPoint4DEvent.

POINT4D_PACKET: a valid Point4DEventPacket pointer. Cannot be NULL.

4.20.2.13 POINT4D_SCALE_MASK

```
#define POINT4D_SCALE_MASK 0x000000FF
```

Shift and mask values for type and scale information associated with a Point4D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.20.2.14 POINT4D_SCALE_SHIFT

```
#define POINT4D_SCALE_SHIFT 8
```

Shift and mask values for type and scale information associated with a Point4D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.20.2.15 POINT4D_TYPE_MASK

```
#define POINT4D_TYPE_MASK 0x0000007F
```

Shift and mask values for type and scale information associated with a Point4D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.20.2.16 POINT4D_TYPE_SHIFT

```
#define POINT4D_TYPE_SHIFT 1
```

Shift and mask values for type and scale information associated with a Point4D event. Up to 128 types are supported. The scale is given as orders of magnitude, from 10^-128 to 10^127. Bit 0 is the valid mark, see 'common.h' for more details.

4.20.3 Typedef Documentation

4.20.3.1 caerPoint4DEvent

```
typedef struct caer_point4d_event* caerPoint4DEvent
```

Type for pointer to Point4D event data structure.

4.20.3.2 caerPoint4DEventPacket

```
typedef struct caer_point4d_event_packet* caerPoint4DEventPacket
```

Type for pointer to Point4D event packet data structure.

4.20.4 Function Documentation

4.20.4.1 caerPoint4DEventGetScale()

```
static int8_t caerPoint4DEventGetScale (
             caerPoint4DEventConst event )  [inline], [static]
```

Get the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

the Point4D measurement scale.

4.20.4.2 caerPoint4DEventGetTimestamp()

```
static int32_t caerPoint4DEventGetTimestamp (
             caerPoint4DEventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.20.4.3 caerPoint4DEventGetTimestamp64()

```
static int64_t caerPoint4DEventGetTimestamp64 (
             caerPoint4DEventConst event,
             caerPoint4DEventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
packet	the Point4DEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.20.4.4 caerPoint4DEventGetType()

```
static uint8_t caerPoint4DEventGetType (
            caerPoint4DEventConst event )  [inline], [static]
```

Get the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

the Point4D measurement type.

4.20.4.5 caerPoint4DEventGetW()

```
static float caerPoint4DEventGetW (
            caerPoint4DEventConst event )  [inline], [static]
```

Get the W axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

W axis measurement.

4.20.4.6 caerPoint4DEventGetX()

```
static float caerPoint4DEventGetX (
             caerPoint4DEventConst event )  [inline], [static]
```

Get the X axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

X axis measurement.

4.20.4.7 caerPoint4DEventGetY()

```
static float caerPoint4DEventGetY (
             caerPoint4DEventConst event )  [inline], [static]
```

Get the Y axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

Y axis measurement.

4.20.4.8 caerPoint4DEventGetZ()

```
static float caerPoint4DEventGetZ (
             caerPoint4DEventConst event )  [inline], [static]
```

Get the Z axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

Z axis measurement.

4.20.4.9 caerPoint4DEventInvalidate()

```
static void caerPoint4DEventInvalidate (
             caerPoint4DEvent event,
             caerPoint4DEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
packet	the Point4DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.20.4.10 caerPoint4DEventIsValid()

```
static bool caerPoint4DEventIsValid (
             caerPoint4DEventConst event )  [inline], [static]
```

Check if this Point4D event is valid.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.20.4.11 caerPoint4DEventPacketAllocate()

```
static caerPoint4DEventPacket caerPoint4DEventPacketAllocate (
             int32_t eventCapacity,
             int16_t eventSource,
             int32_t tsOverflow )  [inline], [static]
```

Allocate a new Point4D events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid Point4DEventPacket handle or NULL on error.

4.20.4.12 caerPoint4DEventPacketFromPacketHeader()

```
static caerPoint4DEventPacket caerPoint4DEventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into a Point4D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.20.4.13 caerPoint4DEventPacketFromPacketHeaderConst()

```
static caerPoint4DEventPacketConst caerPoint4DEventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only Point4D event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.20.4.14 caerPoint4DEventPacketGetEvent()

```
static caerPoint4DEvent caerPoint4DEventPacketGetEvent (
            caerPoint4DEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the Point4D event at the given index from the event packet.

Parameters

packet	a valid Point4DEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Point4D event. NULL on error.

4.20.4.15 caerPoint4DEventPacketGetEventConst()

```
static caerPoint4DEventConst caerPoint4DEventPacketGetEventConst (
            caerPoint4DEventPacketConst packet,
            int32_t n )  [inline], [static]
```

Get the Point4D event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid Point4DEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only Point4D event. NULL on error.

4.20.4.16 caerPoint4DEventSetScale()

```
static void caerPoint4DEventSetScale (
            caerPoint4DEvent event,
            int8_t scale )  [inline], [static]
```

Set the measurement scale. This allows order of magnitude shifts on the measured value to be applied automatically, such as having measurements of type Distance (meters) and storing the values as centimeters (10^-2) for higher precision, but keeping that information around to allow easy changes of unit.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
scale	the Point4D measurement scale.

4.20.4.17 caerPoint4DEventSetTimestamp()

```
static void caerPoint4DEventSetTimestamp (
            caerPoint4DEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.20.4.18 caerPoint4DEventSetType()

```
static void caerPoint4DEventSetType (
            caerPoint4DEvent event,
            uint8_t type )  [inline], [static]
```

Set the measurement event type. This is useful to distinguish between different measurements, for example distance or weight.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
type	the Point4D measurement type.

4.20.4.19 caerPoint4DEventSetW()

```
static void caerPoint4DEventSetW (
            caerPoint4DEvent event,
            float w )  [inline], [static]
```

Set the W axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
w	W axis measurement.

4.20.4.20 caerPoint4DEventSetX()

```
static void caerPoint4DEventSetX (
            caerPoint4DEvent event,
            float x )  [inline], [static]
```

Set the X axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
x	X axis measurement.

4.20.4.21 caerPoint4DEventSetY()

```
static void caerPoint4DEventSetY (
            caerPoint4DEvent event,
            float y )  [inline], [static]
```

Set the Y axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
y	Y axis measurement.

4.20.4.22 caerPoint4DEventSetZ()

```
static void caerPoint4DEventSetZ (
            caerPoint4DEvent event,
            float z )  [inline], [static]
```

Set the Z axis measurement.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
z	Z axis measurement.

4.20.4.23 caerPoint4DEventValidate()

```
static void caerPoint4DEventValidate (
            caerPoint4DEvent event,
            caerPoint4DEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid Point4DEvent pointer. Cannot be NULL.
packet	the Point4DEventPacket pointer for the packet containing this event. Cannot be NULL.

4.20.4.24 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_point4d_event { uint32_t info;float x;float y;float z;float w;int32_t
timestamp;}  )
```

Point4D event data structure definition. This contains information about the measurement, such as a type and a scale field, together with the usual validity mark. The four measurements (x, y, z, w) are stored as floats. Floats are in IEEE 754-2008 binary32 format. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.20.4.25 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_point4d_event_packet { struct caer_event_packet_header packetHeader;struct
caer_point4d_event events[];}  )
```

Point4D event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.21 events/polarity.h File Reference

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

Macros

- #define CAER_POLARITY_ITERATOR_ALL_START(POLARITY_PACKET)
- #define CAER_POLARITY_CONST_ITERATOR_ALL_START(POLARITY_PACKET)
- #define CAER_POLARITY_ITERATOR_ALL_END }
- #define CAER_POLARITY_ITERATOR_VALID_START(POLARITY_PACKET)
- #define CAER_POLARITY_CONST_ITERATOR_VALID_START(POLARITY_PACKET)
- #define CAER_POLARITY_ITERATOR_VALID_END }
- #define CAER_POLARITY_REVERSE_ITERATOR_ALL_START(POLARITY_PACKET)
- #define CAER_POLARITY_CONST_REVERSE_ITERATOR_ALL_START(POLARITY_PACKET)
- #define CAER_POLARITY_REVERSE_ITERATOR_ALL_END }
- #define CAER_POLARITY_REVERSE_ITERATOR_VALID_START(POLARITY_PACKET)
- #define CAER_POLARITY_CONST_REVERSE_ITERATOR_VALID_START(POLARITY_PACKET)
- #define CAER_POLARITY_REVERSE_ITERATOR_VALID_END }

- #define POLARITY_SHIFT 1
- #define POLARITY_MASK 0x00000001
- #define POLARITY_Y_ADDR_SHIFT 2
- #define POLARITY_Y_ADDR_MASK 0x00007FFF
- #define POLARITY_X_ADDR_SHIFT 17
- #define POLARITY_X_ADDR_MASK 0x00007FFF

Typedefs

- typedef struct caer_polarity_event ∗ caerPolarityEvent
- typedef const struct caer_polarity_event ∗ **caerPolarityEventConst**
- typedef struct caer_polarity_event_packet ∗ caerPolarityEventPacket
- typedef const struct caer_polarity_event_packet ∗ **caerPolarityEventPacketConst**

Functions

- PACKED_STRUCT (struct caer_polarity_event { uint32_t data;int32_t timestamp;})
- PACKED_STRUCT (struct caer_polarity_event_packet { struct caer_event_packet_header packet←Header;struct caer_polarity_event events[ ];})
- static caerPolarityEventPacket caerPolarityEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerPolarityEventPacket caerPolarityEventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerPolarityEventPacketConst caerPolarityEventPacketFromPacketHeaderConst (caerEventPacket←HeaderConst header)
- static caerPolarityEvent caerPolarityEventPacketGetEvent (caerPolarityEventPacket packet, int32_t n)
- static caerPolarityEventConst caerPolarityEventPacketGetEventConst (caerPolarityEventPacketConst packet, int32_t n)
- static int32_t caerPolarityEventGetTimestamp (caerPolarityEventConst event)
- static int64_t caerPolarityEventGetTimestamp64 (caerPolarityEventConst event, caerPolarityEventPacket←Const packet)
- static void caerPolarityEventSetTimestamp (caerPolarityEvent event, int32_t timestamp)
- static bool caerPolarityEventIsValid (caerPolarityEventConst event)
- static void caerPolarityEventValidate (caerPolarityEvent event, caerPolarityEventPacket packet)
- static void caerPolarityEventInvalidate (caerPolarityEvent event, caerPolarityEventPacket packet)
- static bool caerPolarityEventGetPolarity (caerPolarityEventConst event)
- static void caerPolarityEventSetPolarity (caerPolarityEvent event, bool polarity)
- static uint16_t caerPolarityEventGetY (caerPolarityEventConst event)
- static void caerPolarityEventSetY (caerPolarityEvent event, uint16_t yAddress)
- static uint16_t caerPolarityEventGetX (caerPolarityEventConst event)
- static void caerPolarityEventSetX (caerPolarityEvent event, uint16_t xAddress)

4.21.1 Detailed Description

Polarity Events format definition and handling functions. This event contains change information, with an X/Y address and an ON/OFF polarity. The (0, 0) address is in the upper left corner of the screen, like in OpenCV/computer graphics.

4.21.2 Macro Definition Documentation

4.21.2.1 CAER_POLARITY_CONST_ITERATOR_ALL_START

```
#define CAER_POLARITY_CONST_ITERATOR_ALL_START(
                POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter = 0;                                                  \
         caerPolarityIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POLARITY_PACKET)->packetHeader); \
         caerPolarityIteratorCounter++) {
         \
        caerPolarityEventConst caerPolarityIteratorElement
         \
            = caerPolarityEventPacketGetEventConst(POLARITY_PACKET,
      caerPolarityIteratorCounter);
```

Const-Iterator over all polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPolarityIteratorElement' variable of type caer←PolarityEventConst.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.2 CAER_POLARITY_CONST_ITERATOR_VALID_START

```
#define CAER_POLARITY_CONST_ITERATOR_VALID_START(
                POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter = 0;                                                  \
         caerPolarityIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POLARITY_PACKET)->packetHeader); \
         caerPolarityIteratorCounter++) {
         \
        caerPolarityEventConst caerPolarityIteratorElement
         \
            = caerPolarityEventPacketGetEventConst(POLARITY_PACKET,
      caerPolarityIteratorCounter);                \
        if (!caerPolarityEventIsValid(caerPolarityIteratorElement)) {
                                     \
            continue;
         \
        }
```

Const-Iterator over only the valid polarity events in a packet. Returns the current index in the 'caerPolarityIterator←Counter' variable of type 'int32_t' and the current read-only event in the 'caerPolarityIteratorElement' variable of type caerPolarityEventConst.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.3 CAER_POLARITY_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_POLARITY_CONST_REVERSE_ITERATOR_ALL_START(
                POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter                                         \
         = caerEventPacketHeaderGetEventNumber(&(POLARITY_PACKET)->
      packetHeader) - 1; \
         caerPolarityIteratorCounter >= 0; caerPolarityIteratorCounter--) {          \
        caerPolarityEventConst caerPolarityIteratorElement                           \
            = caerPolarityEventPacketGetEventConst(POLARITY_PACKET,
      caerPolarityIteratorCounter);
```

Const-Reverse iterator over all polarity events in a packet. Returns the current index in the 'caerPolarityIterator←Counter' variable of type 'int32_t' and the current read-only event in the 'caerPolarityIteratorElement' variable of type caerPolarityEventConst.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.4 CAER_POLARITY_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_POLARITY_CONST_REVERSE_ITERATOR_VALID_START(
                POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter                                                         \
         = caerEventPacketHeaderGetEventNumber(&(POLARITY_PACKET)->
      packetHeader) - 1;             \
         caerPolarityIteratorCounter >= 0; caerPolarityIteratorCounter--) {                          \
        caerPolarityEventConst caerPolarityIteratorElement                                           \
            = caerPolarityEventPacketGetEventConst(POLARITY_PACKET,
      caerPolarityIteratorCounter); \
        if (!caerPolarityEventIsValid(caerPolarityIteratorElement)) {
                     \
            continue;                                                                                 \
        }
```

Const-Reverse iterator over only the valid polarity events in a packet. Returns the current index in the 'caerPolarity↩ IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerPolarityIteratorElement' variable of type caerPolarityEventConst.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.5 CAER_POLARITY_ITERATOR_ALL_END

```
#define CAER_POLARITY_ITERATOR_ALL_END }
```

Iterator close statement.

4.21.2.6 CAER_POLARITY_ITERATOR_ALL_START

```
#define CAER_POLARITY_ITERATOR_ALL_START(
                POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter = 0;                                                          \
         caerPolarityIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POLARITY_PACKET)->packetHeader); \
         caerPolarityIteratorCounter++) {
         \
        caerPolarityEvent caerPolarityIteratorElement
         \
            = caerPolarityEventPacketGetEvent(POLARITY_PACKET,
      caerPolarityIteratorCounter);
```

Iterator over all polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPolarityIteratorElement' variable of type caerPolarityEvent.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.7 CAER_POLARITY_ITERATOR_VALID_END

```
#define CAER_POLARITY_ITERATOR_VALID_END }
```

Iterator close statement.

4.21.2.8 CAER_POLARITY_ITERATOR_VALID_START

```
#define CAER_POLARITY_ITERATOR_VALID_START(
              POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter = 0;                                                   \
        caerPolarityIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(POLARITY_PACKET)->packetHeader); \
        caerPolarityIteratorCounter++) {
        \
       caerPolarityEvent caerPolarityIteratorElement
        \
           = caerPolarityEventPacketGetEvent(POLARITY_PACKET,
      caerPolarityIteratorCounter);                  \
       if (!caerPolarityEventIsValid(caerPolarityIteratorElement)) {
                                   \
           continue;
        \
       }
```

Iterator over only the valid polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPolarityIteratorElement' variable of type caerPolarityEvent.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.9 CAER_POLARITY_REVERSE_ITERATOR_ALL_END

```
#define CAER_POLARITY_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.21.2.10 CAER_POLARITY_REVERSE_ITERATOR_ALL_START

```
#define CAER_POLARITY_REVERSE_ITERATOR_ALL_START(
              POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter                                          \
        = caerEventPacketHeaderGetEventNumber(&(POLARITY_PACKET)->
      packetHeader) - 1; \
        caerPolarityIteratorCounter >= 0; caerPolarityIteratorCounter--) {            \
       caerPolarityEvent caerPolarityIteratorElement                                  \
           = caerPolarityEventPacketGetEvent(POLARITY_PACKET,
      caerPolarityIteratorCounter);
```

Reverse iterator over all polarity events in a packet. Returns the current index in the 'caerPolarityIteratorCounter' variable of type 'int32_t' and the current event in the 'caerPolarityIteratorElement' variable of type caerPolarityEvent.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.11 CAER_POLARITY_REVERSE_ITERATOR_VALID_END

```
#define CAER_POLARITY_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.21.2.12 CAER_POLARITY_REVERSE_ITERATOR_VALID_START

```
#define CAER_POLARITY_REVERSE_ITERATOR_VALID_START(
                POLARITY_PACKET )
```

Value:

```
for (int32_t caerPolarityIteratorCounter                                          \
         = caerEventPacketHeaderGetEventNumber(&(POLARITY_PACKET)->
      packetHeader) - 1;        \
         caerPolarityIteratorCounter >= 0; caerPolarityIteratorCounter--) {          \
        caerPolarityEvent caerPolarityIteratorElement                                 \
           = caerPolarityEventPacketGetEvent(POLARITY_PACKET,
      caerPolarityIteratorCounter); \
        if (!caerPolarityEventIsValid(caerPolarityIteratorElement)) {
                \
            continue;                                                                 \
        }
```

Reverse iterator over only the valid polarity events in a packet. Returns the current index in the 'caerPolarity←IteratorCounter' variable of type 'int32_t' and the current event in the 'caerPolarityIteratorElement' variable of type caerPolarityEvent.

POLARITY_PACKET: a valid PolarityEventPacket pointer. Cannot be NULL.

4.21.2.13 POLARITY_MASK

```
#define POLARITY_MASK 0x00000001
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 15 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see 'common.h' for more details.

4.21.2.14 POLARITY_SHIFT

```
#define POLARITY_SHIFT 1
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 15 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see 'common.h' for more details.

4.21.2.15 POLARITY_X_ADDR_MASK

```
#define POLARITY_X_ADDR_MASK 0x00007FFF
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 15 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see 'common.h' for more details.

4.21.2.16 POLARITY_X_ADDR_SHIFT

```
#define POLARITY_X_ADDR_SHIFT 17
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 15 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see 'common.h' for more details.

4.21.2.17 POLARITY_Y_ADDR_MASK

```
#define POLARITY_Y_ADDR_MASK 0x00007FFF
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 15 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see 'common.h' for more details.

4.21.2.18 POLARITY_Y_ADDR_SHIFT

```
#define POLARITY_Y_ADDR_SHIFT 2
```

Shift and mask values for the polarity, X and Y addresses of a polarity event. Addresses up to 15 bit are supported. Polarity is ON(=1) or OFF(=0). Bit 0 is the valid mark, see 'common.h' for more details.

4.21.3 Typedef Documentation

4.21.3.1 caerPolarityEvent

```
typedef struct caer_polarity_event* caerPolarityEvent
```

Type for pointer to polarity event data structure.

4.21.3.2 caerPolarityEventPacket

```
typedef struct caer_polarity_event_packet* caerPolarityEventPacket
```

Type for pointer to polarity event packet data structure.

4.21.4 Function Documentation

4.21.4.1 caerPolarityEventGetPolarity()

```
static bool caerPolarityEventGetPolarity (
              caerPolarityEventConst event )  [inline], [static]
```

Get the change event polarity. 1 is ON, 0 is OFF.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.

Returns

event polarity value.

4.21.4.2 caerPolarityEventGetTimestamp()

```
static int32_t caerPolarityEventGetTimestamp (
              caerPolarityEventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.21.4.3 caerPolarityEventGetTimestamp64()

```
static int64_t caerPolarityEventGetTimestamp64 (
              caerPolarityEventConst event,
              caerPolarityEventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
packet	the PolarityEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.21.4.4 caerPolarityEventGetX()

```
static uint16_t caerPolarityEventGetX (
              caerPolarityEventConst event )  [inline], [static]
```

Get the X (column) address for a change event, in pixels. The (0, 0) address is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.

Returns

the event X address.

4.21.4.5 caerPolarityEventGetY()

```
static uint16_t caerPolarityEventGetY (
            caerPolarityEventConst event )  [inline], [static]
```

Get the Y (row) address for a change event, in pixels. The (0, 0) address is in the upper left corner, like in OpenC↩ V/computer graphics.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.

Returns

the event Y address.

4.21.4.6 caerPolarityEventInvalidate()

```
static void caerPolarityEventInvalidate (
            caerPolarityEvent event,
            caerPolarityEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
packet	the PolarityEventPacket pointer for the packet containing this event. Cannot be NULL.

4.21.4.7 caerPolarityEventIsValid()

```
static bool caerPolarityEventIsValid (
            caerPolarityEventConst event )  [inline], [static]
```

Check if this polarity event is valid.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.21.4.8 caerPolarityEventPacketAllocate()

```
static caerPolarityEventPacket caerPolarityEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )  [inline], [static]
```

Allocate a new polarity events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid PolarityEventPacket handle or NULL on error.

4.21.4.9 caerPolarityEventPacketFromPacketHeader()

```
static caerPolarityEventPacket caerPolarityEventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into a Polarity event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.21.4.10 caerPolarityEventPacketFromPacketHeaderConst()

```
static caerPolarityEventPacketConst caerPolarityEventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only Polarity event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.21.4.11 caerPolarityEventPacketGetEvent()

```
static caerPolarityEvent caerPolarityEventPacketGetEvent (
            caerPolarityEventPacket packet,
            int32_t n )  [inline], [static]
```

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

Parameters

packet	a valid PolarityEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested polarity event. NULL on error.

4.21.4.12 caerPolarityEventPacketGetEventConst()

```
static caerPolarityEventConst caerPolarityEventPacketGetEventConst (
            caerPolarityEventPacketConst packet,
            int32_t n )  [inline], [static]
```

Get the polarity event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid PolarityEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only polarity event. NULL on error.

4.21.4.13 caerPolarityEventSetPolarity()

```
static void caerPolarityEventSetPolarity (
             caerPolarityEvent event,
             bool polarity )  [inline], [static]
```

Set the change event polarity. 1 is ON, 0 is OFF.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
polarity	event polarity value.

4.21.4.14 caerPolarityEventSetTimestamp()

```
static void caerPolarityEventSetTimestamp (
             caerPolarityEvent event,
             int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.21.4.15 caerPolarityEventSetX()

```
static void caerPolarityEventSetX (
             caerPolarityEvent event,
             uint16_t xAddress )  [inline], [static]
```

Set the X (column) address for a change event, in pixels. The (0, 0) address is in the upper left corner, like in OpenCV/computer graphics.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
xAddress	the event X address.

4.21.4.16 caerPolarityEventSetY()

```
static void caerPolarityEventSetY (
            caerPolarityEvent event,
            uint16_t yAddress )  [inline], [static]
```

Set the Y (row) address for a change event, in pixels. The (0, 0) address is in the upper left corner, like in OpenC↩ V/computer graphics.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
yAddress	the event Y address.

4.21.4.17 caerPolarityEventValidate()

```
static void caerPolarityEventValidate (
            caerPolarityEvent event,
            caerPolarityEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid PolarityEvent pointer. Cannot be NULL.
packet	the PolarityEventPacket pointer for the packet containing this event. Cannot be NULL.

4.21.4.18 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_polarity_event { uint32_t data;int32_t timestamp;}  )
```

Polarity event data structure definition. This contains the actual X/Y addresses, the polarity, as well as the 32 bit event timestamp. The (0, 0) address is in the upper left corner of the screen, like in OpenCV/computer graphics. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.21.4.19 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_polarity_event_packet { struct caer_event_packet_header packet↩
Header;struct caer_polarity_event events[];}  )
```

Polarity event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.22 events/sample.h File Reference

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

Macros

- #define CAER_SAMPLE_ITERATOR_ALL_START(SAMPLE_PACKET)
- #define CAER_SAMPLE_CONST_ITERATOR_ALL_START(SAMPLE_PACKET)
- #define CAER_SAMPLE_ITERATOR_ALL_END }
- #define CAER_SAMPLE_ITERATOR_VALID_START(SAMPLE_PACKET)
- #define CAER_SAMPLE_CONST_ITERATOR_VALID_START(SAMPLE_PACKET)
- #define CAER_SAMPLE_ITERATOR_VALID_END }
- #define CAER_SAMPLE_REVERSE_ITERATOR_ALL_START(SAMPLE_PACKET)
- #define CAER_SAMPLE_CONST_REVERSE_ITERATOR_ALL_START(SAMPLE_PACKET)
- #define CAER_SAMPLE_REVERSE_ITERATOR_ALL_END }
- #define CAER_SAMPLE_REVERSE_ITERATOR_VALID_START(SAMPLE_PACKET)
- #define CAER_SAMPLE_CONST_REVERSE_ITERATOR_VALID_START(SAMPLE_PACKET)
- #define CAER_SAMPLE_REVERSE_ITERATOR_VALID_END }

- #define SAMPLE_TYPE_SHIFT 1
- #define SAMPLE_TYPE_MASK 0x0000007F
- #define SAMPLE_SHIFT 8
- #define SAMPLE_MASK 0x00FFFFFF

Typedefs

- typedef struct caer_sample_event ∗ caerSampleEvent
- typedef const struct caer_sample_event ∗ **caerSampleEventConst**
- typedef struct caer_sample_event_packet ∗ caerSampleEventPacket
- typedef const struct caer_sample_event_packet ∗ **caerSampleEventPacketConst**

Functions

- PACKED_STRUCT (struct caer_sample_event { uint32_t data;int32_t timestamp;})
- PACKED_STRUCT (struct caer_sample_event_packet { struct caer_event_packet_header packet↩ Header;struct caer_sample_event events[ ];})
- static caerSampleEventPacket caerSampleEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerSampleEventPacket caerSampleEventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerSampleEventPacketConst caerSampleEventPacketFromPacketHeaderConst (caerEventPacket↩ HeaderConst header)
- static caerSampleEvent caerSampleEventPacketGetEvent (caerSampleEventPacket packet, int32_t n)
- static caerSampleEventConst caerSampleEventPacketGetEventConst (caerSampleEventPacketConst packet, int32_t n)
- static int32_t caerSampleEventGetTimestamp (caerSampleEventConst event)
- static int64_t caerSampleEventGetTimestamp64 (caerSampleEventConst event, caerSampleEventPacket↩ Const packet)
- static void caerSampleEventSetTimestamp (caerSampleEvent event, int32_t timestamp)
- static bool caerSampleEventIsValid (caerSampleEventConst event)
- static void caerSampleEventValidate (caerSampleEvent event, caerSampleEventPacket packet)
- static void caerSampleEventInvalidate (caerSampleEvent event, caerSampleEventPacket packet)
- static uint8_t caerSampleEventGetType (caerSampleEventConst event)
- static void caerSampleEventSetType (caerSampleEvent event, uint8_t type)
- static uint32_t caerSampleEventGetSample (caerSampleEventConst event)
- static void caerSampleEventSetSample (caerSampleEvent event, uint32_t sample)

4.22.1 Detailed Description

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

4.22.2 Macro Definition Documentation

4.22.2.1 CAER_SAMPLE_CONST_ITERATOR_ALL_START

```
#define CAER_SAMPLE_CONST_ITERATOR_ALL_START(
              SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = 0;                                    \
         caerSampleIteratorCounter < caerEventPacketHeaderGetEventNumber
       (&(SAMPLE_PACKET)->packetHeader); \
         caerSampleIteratorCounter++) {                                          \
        caerSampleEventConst caerSampleIteratorElement                           \
            = caerSampleEventPacketGetEventConst(SAMPLE_PACKET,
       caerSampleIteratorCounter);
```

Const-Iterator over all sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSampleIteratorElement' variable of type caer↩ SampleEventConst.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.2 CAER_SAMPLE_CONST_ITERATOR_VALID_START

```
#define CAER_SAMPLE_CONST_ITERATOR_VALID_START(
                SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = 0;                                                    \
         caerSampleIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(SAMPLE_PACKET)->packetHeader); \
         caerSampleIteratorCounter++) {                                                            \
        caerSampleEventConst caerSampleIteratorElement                                             \
            = caerSampleEventPacketGetEventConst(SAMPLE_PACKET,
      caerSampleIteratorCounter);                    \
        if (!caerSampleEventIsValid(caerSampleIteratorElement)) {
                           \
            continue;                                                                              \
        }
```

Const-Iterator over only the valid sample events in a packet. Returns the current index in the 'caerSampleIterator↩ Counter' variable of type 'int32_t' and the current read-only event in the 'caerSampleIteratorElement' variable of type caerSampleEventConst.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.3 CAER_SAMPLE_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_SAMPLE_CONST_REVERSE_ITERATOR_ALL_START(
                SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(SAMPLE_PACKET)->packetHeader) - 1; \
         caerSampleIteratorCounter >= 0; caerSampleIteratorCounter--) {
                 \
        caerSampleEventConst caerSampleIteratorElement
                 \
            = caerSampleEventPacketGetEventConst(SAMPLE_PACKET,
      caerSampleIteratorCounter);
```

Const-Reverse iterator over all sample events in a packet. Returns the current index in the 'caerSampleIterator↩ Counter' variable of type 'int32_t' and the current read-only event in the 'caerSampleIteratorElement' variable of type caerSampleEventConst.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.4 CAER_SAMPLE_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_SAMPLE_CONST_REVERSE_ITERATOR_VALID_START(
                SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(SAMPLE_PACKET)->packetHeader) - 1; \
         caerSampleIteratorCounter >= 0; caerSampleIteratorCounter--) {
                 \
        caerSampleEventConst caerSampleIteratorElement
                 \
            = caerSampleEventPacketGetEventConst(SAMPLE_PACKET,
      caerSampleIteratorCounter);                        \
        if (!caerSampleEventIsValid(caerSampleIteratorElement)) {
                                       \
            continue;
                 \
        }
```

Const-Reverse iterator over only the valid sample events in a packet. Returns the current index in the 'caerSample↩ IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSampleIteratorElement' variable of type caerSampleEventConst.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.5 CAER_SAMPLE_ITERATOR_ALL_END

```
#define CAER_SAMPLE_ITERATOR_ALL_END }
```

Iterator close statement.

4.22.2.6 CAER_SAMPLE_ITERATOR_ALL_START

```
#define CAER_SAMPLE_ITERATOR_ALL_START(
                SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = 0;                                                    \
        caerSampleIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(SAMPLE_PACKET)->packetHeader); \
        caerSampleIteratorCounter++) {                                                            \
       caerSampleEvent caerSampleIteratorElement                                                  \
           = caerSampleEventPacketGetEvent(SAMPLE_PACKET,
      caerSampleIteratorCounter);
```

Iterator over all sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSampleIteratorElement' variable of type caerSampleEvent.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.7 CAER_SAMPLE_ITERATOR_VALID_END

```
#define CAER_SAMPLE_ITERATOR_VALID_END }
```

Iterator close statement.

4.22.2.8 CAER_SAMPLE_ITERATOR_VALID_START

```
#define CAER_SAMPLE_ITERATOR_VALID_START(
                SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = 0;                                                    \
        caerSampleIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(SAMPLE_PACKET)->packetHeader); \
        caerSampleIteratorCounter++) {                                                            \
       caerSampleEvent caerSampleIteratorElement                                                  \
           = caerSampleEventPacketGetEvent(SAMPLE_PACKET,
      caerSampleIteratorCounter);                          \
       if (!caerSampleEventIsValid(caerSampleIteratorElement)) {
                              \
           continue;                                                                              \
       }
```

Iterator over only the valid sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSampleIteratorElement' variable of type caerSampleEvent.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.9 CAER_SAMPLE_REVERSE_ITERATOR_ALL_END

```
#define CAER_SAMPLE_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.22.2.10 CAER_SAMPLE_REVERSE_ITERATOR_ALL_START

```
#define CAER_SAMPLE_REVERSE_ITERATOR_ALL_START(
                SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(SAMPLE_PACKET)->packetHeader) - 1; \
         caerSampleIteratorCounter >= 0; caerSampleIteratorCounter--) {
                 \
        caerSampleEvent caerSampleIteratorElement
                 \
            = caerSampleEventPacketGetEvent(SAMPLE_PACKET,
      caerSampleIteratorCounter);
```

Reverse iterator over all sample events in a packet. Returns the current index in the 'caerSampleIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSampleIteratorElement' variable of type caerSampleEvent.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.11 CAER_SAMPLE_REVERSE_ITERATOR_VALID_END

```
#define CAER_SAMPLE_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.22.2.12 CAER_SAMPLE_REVERSE_ITERATOR_VALID_START

```
#define CAER_SAMPLE_REVERSE_ITERATOR_VALID_START(
                SAMPLE_PACKET )
```

Value:

```
for (int32_t caerSampleIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(SAMPLE_PACKET)->packetHeader) - 1; \
         caerSampleIteratorCounter >= 0; caerSampleIteratorCounter--) {
                 \
        caerSampleEvent caerSampleIteratorElement
                 \
            = caerSampleEventPacketGetEvent(SAMPLE_PACKET,
      caerSampleIteratorCounter);                              \
        if (!caerSampleEventIsValid(caerSampleIteratorElement)) {
                                                    \
            continue;
                 \
        }
```

Reverse iterator over only the valid sample events in a packet. Returns the current index in the 'caerSample↩ IteratorCounter' variable of type 'int32_t' and the current event in the 'caerSampleIteratorElement' variable of type caerSampleEvent.

SAMPLE_PACKET: a valid SampleEventPacket pointer. Cannot be NULL.

4.22.2.13 SAMPLE_MASK

```
#define SAMPLE_MASK 0x00FFFFFF
```

Shift and mask values for the sample type and the actual sample value of an ADC sample. Up to 128 sample types are supported, with 24 bits of data per sample. Higher values mean a higher voltage, 0 is ground. Bit 0 is the valid mark, see 'common.h' for more details.

4.22.2.14 SAMPLE_SHIFT

```
#define SAMPLE_SHIFT 8
```

Shift and mask values for the sample type and the actual sample value of an ADC sample. Up to 128 sample types are supported, with 24 bits of data per sample. Higher values mean a higher voltage, 0 is ground. Bit 0 is the valid mark, see 'common.h' for more details.

4.22.2.15 SAMPLE_TYPE_MASK

```
#define SAMPLE_TYPE_MASK 0x0000007F
```

Shift and mask values for the sample type and the actual sample value of an ADC sample. Up to 128 sample types are supported, with 24 bits of data per sample. Higher values mean a higher voltage, 0 is ground. Bit 0 is the valid mark, see 'common.h' for more details.

4.22.2.16 SAMPLE_TYPE_SHIFT

```
#define SAMPLE_TYPE_SHIFT 1
```

Shift and mask values for the sample type and the actual sample value of an ADC sample. Up to 128 sample types are supported, with 24 bits of data per sample. Higher values mean a higher voltage, 0 is ground. Bit 0 is the valid mark, see 'common.h' for more details.

4.22.3 Typedef Documentation

4.22.3.1 caerSampleEvent

```
typedef struct caer_sample_event* caerSampleEvent
```

Type for pointer to ADC sample event data structure.

4.22.3.2 caerSampleEventPacket

```
typedef struct caer_sample_event_packet* caerSampleEventPacket
```

Type for pointer to ADC sample event packet data structure.

4.22.4 Function Documentation

4.22.4.1 caerSampleEventGetSample()

```
static uint32_t caerSampleEventGetSample (
             caerSampleEventConst event )  [inline], [static]
```

Get the ADC sample value. Up to 24 bits of resolution are possible. Higher values mean a higher voltage, 0 is ground.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.

Returns

the ADC sample value.

4.22.4.2 caerSampleEventGetTimestamp()

```
static int32_t caerSampleEventGetTimestamp (
             caerSampleEventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.22.4.3 caerSampleEventGetTimestamp64()

```
static int64_t caerSampleEventGetTimestamp64 (
             caerSampleEventConst event,
             caerSampleEventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.
packet	the SampleEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.22.4.4 caerSampleEventGetType()

```
static uint8_t caerSampleEventGetType (
             caerSampleEventConst event )  [inline], [static]
```

Get the ADC sample event type. This is useful to distinguish between different measurements, for example from two separate microphones on a device.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.

Returns

the ADC sample type.

4.22.4.5 caerSampleEventInvalidate()

```
static void caerSampleEventInvalidate (
             caerSampleEvent event,
             caerSampleEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.
packet	the SampleEventPacket pointer for the packet containing this event. Cannot be NULL.

4.22.4.6 caerSampleEventIsValid()

```
static bool caerSampleEventIsValid (
            caerSampleEventConst event )  [inline], [static]
```

Check if this ADC sample event is valid.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.22.4.7 caerSampleEventPacketAllocate()

```
static caerSampleEventPacket caerSampleEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )  [inline], [static]
```

Allocate a new ADC sample events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid SampleEventPacket handle or NULL on error.

4.22.4.8 caerSampleEventPacketFromPacketHeader()

```
static caerSampleEventPacket caerSampleEventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into a Sample event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.22.4.9 caerSampleEventPacketFromPacketHeaderConst()

```
static caerSampleEventPacketConst caerSampleEventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only Sample event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.22.4.10 caerSampleEventPacketGetEvent()

```
static caerSampleEvent caerSampleEventPacketGetEvent (
            caerSampleEventPacket packet,
            int32_t n )  [inline], [static]
```

Get the ADC sample event at the given index from the event packet.

Parameters

packet	a valid SampleEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested ADC sample event. NULL on error.

4.22.4.11 caerSampleEventPacketGetEventConst()

```
static caerSampleEventConst caerSampleEventPacketGetEventConst (
            caerSampleEventPacketConst packet,
            int32_t n )  [inline], [static]
```

Get the ADC sample event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid SampleEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only ADC sample event. NULL on error.

4.22.4.12 caerSampleEventSetSample()

```
static void caerSampleEventSetSample (
            caerSampleEvent event,
            uint32_t sample )  [inline], [static]
```

Set the ADC sample value. Up to 24 bits of resolution are possible. Higher values mean a higher voltage, 0 is ground.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.
sample	the ADC sample value.

4.22.4.13 caerSampleEventSetTimestamp()

```
static void caerSampleEventSetTimestamp (
            caerSampleEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.22.4.14 caerSampleEventSetType()

```
static void caerSampleEventSetType (
            caerSampleEvent event,
            uint8_t type )  [inline], [static]
```

Set the ADC sample event type. This is useful to distinguish between different measurements, for example from two separate microphones on a device.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.
type	the ADC sample type.

4.22.4.15 caerSampleEventValidate()

```
static void caerSampleEventValidate (
            caerSampleEvent event,
            caerSampleEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid SampleEvent pointer. Cannot be NULL.
packet	the SampleEventPacket pointer for the packet containing this event. Cannot be NULL.

4.22.4.16 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_sample_event { uint32_t data;int32_t timestamp;}  )
```

ADC sample event data structure definition. Contains a type indication to separate different ADC readouts, as well as a value for that readout, up to 24 bits resolution. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.22.4.17 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_sample_event_packet { struct caer_event_packet_header packetHeader;struct
caer_sample_event events[];}  )
```

ADC sample event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.23 events/special.h File Reference

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

Macros

- #define CAER_SPECIAL_ITERATOR_ALL_START(SPECIAL_PACKET)
- #define CAER_SPECIAL_CONST_ITERATOR_ALL_START(SPECIAL_PACKET)
- #define CAER_SPECIAL_ITERATOR_ALL_END }
- #define CAER_SPECIAL_ITERATOR_VALID_START(SPECIAL_PACKET)
- #define CAER_SPECIAL_CONST_ITERATOR_VALID_START(SPECIAL_PACKET)
- #define CAER_SPECIAL_ITERATOR_VALID_END }
- #define CAER_SPECIAL_REVERSE_ITERATOR_ALL_START(SPECIAL_PACKET)
- #define CAER_SPECIAL_CONST_REVERSE_ITERATOR_ALL_START(SPECIAL_PACKET)
- #define CAER_SPECIAL_REVERSE_ITERATOR_ALL_END }
- #define CAER_SPECIAL_REVERSE_ITERATOR_VALID_START(SPECIAL_PACKET)
- #define CAER_SPECIAL_CONST_REVERSE_ITERATOR_VALID_START(SPECIAL_PACKET)
- #define CAER_SPECIAL_REVERSE_ITERATOR_VALID_END }

- #define SPECIAL_TYPE_SHIFT 1
- #define SPECIAL_TYPE_MASK 0x0000007F
- #define SPECIAL_DATA_SHIFT 8
- #define SPECIAL_DATA_MASK 0x00FFFFFF

Typedefs

- typedef struct caer_special_event ∗ caerSpecialEvent
- typedef const struct caer_special_event ∗ **caerSpecialEventConst**
- typedef struct caer_special_event_packet ∗ caerSpecialEventPacket
- typedef const struct caer_special_event_packet ∗ **caerSpecialEventPacketConst**

Enumerations

- enum caer_special_event_types {
  TIMESTAMP_WRAP = 0, TIMESTAMP_RESET = 1, EXTERNAL_INPUT_RISING_EDGE = 2,
  EXTERNAL_INPUT_FALLING_EDGE = 3,
  EXTERNAL_INPUT_PULSE = 4, DVS_ROW_ONLY = 5, EXTERNAL_INPUT1_RISING_EDGE = 6,
  EXTERNAL_INPUT1_FALLING_EDGE = 7,
  EXTERNAL_INPUT1_PULSE = 8, EXTERNAL_INPUT2_RISING_EDGE = 9, EXTERNAL_INPUT2_FALLING_EDGE
  = 10, EXTERNAL_INPUT2_PULSE = 11,
  EXTERNAL_GENERATOR_RISING_EDGE = 12, EXTERNAL_GENERATOR_FALLING_EDGE = 13,
  APS_FRAME_START = 14, APS_FRAME_END = 15,
  APS_EXPOSURE_START = 16, APS_EXPOSURE_END = 17 }

Functions

- PACKED_STRUCT (struct caer_special_event { uint32_t data;int32_t timestamp;})
- PACKED_STRUCT (struct caer_special_event_packet { struct caer_event_packet_header packet↩ Header;struct caer_special_event events[ ];})
- static caerSpecialEventPacket caerSpecialEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerSpecialEventPacket caerSpecialEventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerSpecialEventPacketConst caerSpecialEventPacketFromPacketHeaderConst (caerEventPacket↩ HeaderConst header)
- static caerSpecialEvent caerSpecialEventPacketGetEvent (caerSpecialEventPacket packet, int32_t n)
- static caerSpecialEventConst caerSpecialEventPacketGetEventConst (caerSpecialEventPacketConst packet, int32_t n)
- static int32_t caerSpecialEventGetTimestamp (caerSpecialEventConst event)
- static int64_t caerSpecialEventGetTimestamp64 (caerSpecialEventConst event, caerSpecialEventPacket↩ Const packet)
- static void caerSpecialEventSetTimestamp (caerSpecialEvent event, int32_t timestamp)
- static bool caerSpecialEventIsValid (caerSpecialEventConst event)
- static void caerSpecialEventValidate (caerSpecialEvent event, caerSpecialEventPacket packet)
- static void caerSpecialEventInvalidate (caerSpecialEvent event, caerSpecialEventPacket packet)
- static uint8_t caerSpecialEventGetType (caerSpecialEventConst event)
- static void caerSpecialEventSetType (caerSpecialEvent event, uint8_t type)
- static uint32_t caerSpecialEventGetData (caerSpecialEventConst event)
- static void caerSpecialEventSetData (caerSpecialEvent event, uint32_t data)
- static caerSpecialEvent caerSpecialEventPacketFindEventByType (caerSpecialEventPacket packet, uint8_t type)
- static caerSpecialEventConst caerSpecialEventPacketFindEventByTypeConst (caerSpecialEventPacket↩ Const packet, uint8_t type)
- static caerSpecialEvent caerSpecialEventPacketFindValidEventByType (caerSpecialEventPacket packet, uint8_t type)
- static caerSpecialEventConst caerSpecialEventPacketFindValidEventByTypeConst (caerSpecialEvent↩ PacketConst packet, uint8_t type)

4.23.1 Detailed Description

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

4.23.2 Macro Definition Documentation

4.23.2.1 CAER_SPECIAL_CONST_ITERATOR_ALL_START

```
#define CAER_SPECIAL_CONST_ITERATOR_ALL_START(
              SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter = 0;                                                  \
        caerSpecialIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(SPECIAL_PACKET)->packetHeader); \
        caerSpecialIteratorCounter++) {
       \
       caerSpecialEventConst caerSpecialIteratorElement
       \
           = caerSpecialEventPacketGetEventConst(SPECIAL_PACKET,
      caerSpecialIteratorCounter);
```

Const-Iterator over all special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpecialIteratorElement' variable of type caer↩ SpecialEventConst.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.2 CAER_SPECIAL_CONST_ITERATOR_VALID_START

```
#define CAER_SPECIAL_CONST_ITERATOR_VALID_START(
              SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter = 0;                                                  \
        caerSpecialIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(SPECIAL_PACKET)->packetHeader); \
        caerSpecialIteratorCounter++) {
       \
       caerSpecialEventConst caerSpecialIteratorElement
       \
           = caerSpecialEventPacketGetEventConst(SPECIAL_PACKET,
      caerSpecialIteratorCounter);                    \
         if (!caerSpecialEventIsValid(caerSpecialIteratorElement)) {
                                 \
           continue;
       \
        }
```

Const-Iterator over only the valid special events in a packet. Returns the current index in the 'caerSpecialIterator↩ Counter' variable of type 'int32_t' and the current read-only event in the 'caerSpecialIteratorElement' variable of type caerSpecialEventConst.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.3 CAER_SPECIAL_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_SPECIAL_CONST_REVERSE_ITERATOR_ALL_START(
              SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter                                        \
        = caerEventPacketHeaderGetEventNumber(&(SPECIAL_PACKET)->
      packetHeader) - 1; \
        caerSpecialIteratorCounter >= 0; caerSpecialIteratorCounter--) {             \
       caerSpecialEventConst caerSpecialIteratorElement                              \
           = caerSpecialEventPacketGetEventConst(SPECIAL_PACKET,
      caerSpecialIteratorCounter);
```

Const-Reverse iterator over all special events in a packet. Returns the current index in the 'caerSpecialIterator↩ Counter' variable of type 'int32_t' and the current read-only event in the 'caerSpecialIteratorElement' variable of type caerSpecialEventConst.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.4 CAER_SPECIAL_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_SPECIAL_CONST_REVERSE_ITERATOR_VALID_START(
                SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter                                              \
         = caerEventPacketHeaderGetEventNumber(&(SPECIAL_PACKET)->
      packetHeader) - 1;           \
         caerSpecialIteratorCounter >= 0; caerSpecialIteratorCounter--) {                  \
        caerSpecialEventConst caerSpecialIteratorElement                                   \
            = caerSpecialEventPacketGetEventConst(SPECIAL_PACKET,
      caerSpecialIteratorCounter); \
        if (!caerSpecialEventIsValid(caerSpecialIteratorElement)) {
                 \
            continue;                                                                      \
        }
```

Const-Reverse iterator over only the valid special events in a packet. Returns the current index in the 'caerSpecial↩IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpecialIteratorElement' variable of type caerSpecialEventConst.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.5 CAER_SPECIAL_ITERATOR_ALL_END

```
#define CAER_SPECIAL_ITERATOR_ALL_END }
```

Iterator close statement.

4.23.2.6 CAER_SPECIAL_ITERATOR_ALL_START

```
#define CAER_SPECIAL_ITERATOR_ALL_START(
                SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter = 0;                                                 \
         caerSpecialIteratorCounter < caerEventPacketHeaderGetEventNumber
      (&(SPECIAL_PACKET)->packetHeader); \
         caerSpecialIteratorCounter++) {
       \
        caerSpecialEvent caerSpecialIteratorElement
       \
            = caerSpecialEventPacketGetEvent(SPECIAL_PACKET,
      caerSpecialIteratorCounter);
```

Iterator over all special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpecialIteratorElement' variable of type caerSpecialEvent.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.7 CAER_SPECIAL_ITERATOR_VALID_END

```
#define CAER_SPECIAL_ITERATOR_VALID_END }
```

Iterator close statement.

4.23.2.8 CAER_SPECIAL_ITERATOR_VALID_START

```
#define CAER_SPECIAL_ITERATOR_VALID_START(
                SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter = 0;                                                   \
        caerSpecialIteratorCounter < caerEventPacketHeaderGetEventNumber
     (&(SPECIAL_PACKET)->packetHeader); \
        caerSpecialIteratorCounter++) {
      \
       caerSpecialEvent caerSpecialIteratorElement
      \
           = caerSpecialEventPacketGetEvent(SPECIAL_PACKET,
     caerSpecialIteratorCounter);                    \
       if (!caerSpecialEventIsValid(caerSpecialIteratorElement)) {
                                 \
           continue;
      \
       }
```

Iterator over only the valid special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpecialIteratorElement' variable of type caerSpecialEvent.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.9 CAER_SPECIAL_REVERSE_ITERATOR_ALL_END

```
#define CAER_SPECIAL_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.23.2.10 CAER_SPECIAL_REVERSE_ITERATOR_ALL_START

```
#define CAER_SPECIAL_REVERSE_ITERATOR_ALL_START(
                SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter                                    \
        = caerEventPacketHeaderGetEventNumber(&(SPECIAL_PACKET)->
     packetHeader) - 1; \
        caerSpecialIteratorCounter >= 0; caerSpecialIteratorCounter--) {            \
       caerSpecialEvent caerSpecialIteratorElement                                  \
           = caerSpecialEventPacketGetEvent(SPECIAL_PACKET,
     caerSpecialIteratorCounter);
```

Reverse iterator over all special events in a packet. Returns the current index in the 'caerSpecialIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpecialIteratorElement' variable of type caerSpecialEvent.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.11 CAER_SPECIAL_REVERSE_ITERATOR_VALID_END

```
#define CAER_SPECIAL_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.23.2.12 CAER_SPECIAL_REVERSE_ITERATOR_VALID_START

```
#define CAER_SPECIAL_REVERSE_ITERATOR_VALID_START(
                SPECIAL_PACKET )
```

Value:

```
for (int32_t caerSpecialIteratorCounter                                    \
         = caerEventPacketHeaderGetEventNumber(&(SPECIAL_PACKET)->
      packetHeader) - 1;      \
         caerSpecialIteratorCounter >= 0; caerSpecialIteratorCounter--) {       \
        caerSpecialEvent caerSpecialIteratorElement                            \
            = caerSpecialEventPacketGetEvent(SPECIAL_PACKET,
      caerSpecialIteratorCounter); \
        if (!caerSpecialEventIsValid(caerSpecialIteratorElement)) {
            \
            continue;                                                          \
        }
```

Reverse iterator over only the valid special events in a packet. Returns the current index in the 'caerSpecial↩ IteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpecialIteratorElement' variable of type caerSpecialEvent.

SPECIAL_PACKET: a valid SpecialEventPacket pointer. Cannot be NULL.

4.23.2.13 SPECIAL_DATA_MASK

```
#define SPECIAL_DATA_MASK 0x00FFFFFF
```

Shift and mask values for the type and data portions of a special event. Up to 128 types, with 24 bits of data each, are possible. Bit 0 is the valid mark, see 'common.h' for more details.

4.23.2.14 SPECIAL_DATA_SHIFT

```
#define SPECIAL_DATA_SHIFT 8
```

Shift and mask values for the type and data portions of a special event. Up to 128 types, with 24 bits of data each, are possible. Bit 0 is the valid mark, see 'common.h' for more details.

4.23.2.15 SPECIAL_TYPE_MASK

```
#define SPECIAL_TYPE_MASK 0x0000007F
```

Shift and mask values for the type and data portions of a special event. Up to 128 types, with 24 bits of data each, are possible. Bit 0 is the valid mark, see 'common.h' for more details.

4.23.2.16 SPECIAL_TYPE_SHIFT

```
#define SPECIAL_TYPE_SHIFT 1
```

Shift and mask values for the type and data portions of a special event. Up to 128 types, with 24 bits of data each, are possible. Bit 0 is the valid mark, see 'common.h' for more details.

4.23.3 Typedef Documentation

4.23.3.1 caerSpecialEvent

```
typedef struct caer_special_event* caerSpecialEvent
```

Type for pointer to special event data structure.

4.23.3.2 caerSpecialEventPacket

```
typedef struct caer_special_event_packet* caerSpecialEventPacket
```

Type for pointer to special event packet data structure.

4.23.4 Enumeration Type Documentation

4.23.4.1 caer_special_event_types

```
enum caer_special_event_types
```

List of all special event type identifiers. Used to interpret the special event type field.

Enumerator

TIMESTAMP_WRAP	A 32 bit timestamp wrap occurred.
TIMESTAMP_RESET	A timestamp reset occurred.
EXTERNAL_INPUT_RISING_EDGE	A rising edge was detected (External Input module on device).
EXTERNAL_INPUT_FALLING_EDGE	A falling edge was detected (External Input module on device).
EXTERNAL_INPUT_PULSE	A pulse was detected (External Input module on device).
DVS_ROW_ONLY	A DVS row-only event was detected (a row address without any following column addresses).
EXTERNAL_INPUT1_RISING_EDGE	A rising edge was detected (External Input 1 module on device).
EXTERNAL_INPUT1_FALLING_EDGE	A falling edge was detected (External Input 1 module on device).
EXTERNAL_INPUT1_PULSE	A pulse was detected (External Input 1 module on device).
EXTERNAL_INPUT2_RISING_EDGE	A rising edge was detected (External Input 2 module on device).
EXTERNAL_INPUT2_FALLING_EDGE	A falling edge was detected (External Input 2 module on device).
EXTERNAL_INPUT2_PULSE	A pulse was detected (External Input 2 module on device).

Enumerator

EXTERNAL_GENERATOR_RISING_EDGE	A rising edge was generated (External Input Generator module on device).
EXTERNAL_GENERATOR_FALLING_EDGE	A falling edge was generated (External Input Generator module on device).
APS_FRAME_START	An APS frame capture has started (Frame Event will follow).
APS_FRAME_END	An APS frame capture has completed (Frame Event is alongside).
APS_EXPOSURE_START	An APS frame exposure has started (Frame Event will follow).
APS_EXPOSURE_END	An APS frame exposure has completed (Frame Event will follow).

4.23.5 Function Documentation

4.23.5.1 caerSpecialEventGetData()

```
static uint32_t caerSpecialEventGetData (
             caerSpecialEventConst event )  [inline], [static]
```

Get the special event data. Its meaning depends on the type. Current types that make use of it are (see 'enum caer_special_event_types'):

- DVS_ROW_ONLY: encodes the address of the row from the row-only event.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.

Returns

the special event data.

4.23.5.2 caerSpecialEventGetTimestamp()

```
static int32_t caerSpecialEventGetTimestamp (
             caerSpecialEventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.23.5.3 caerSpecialEventGetTimestamp64()

```
static int64_t caerSpecialEventGetTimestamp64 (
            caerSpecialEventConst event,
            caerSpecialEventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.
packet	the SpecialEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.23.5.4 caerSpecialEventGetType()

```
static uint8_t caerSpecialEventGetType (
            caerSpecialEventConst event )  [inline], [static]
```

Get the numerical special event type.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.

Returns

the special event type (see 'enum caer_special_event_types').

4.23.5.5 caerSpecialEventInvalidate()

```
static void caerSpecialEventInvalidate (
            caerSpecialEvent event,
            caerSpecialEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.
packet	the SpecialEventPacket pointer for the packet containing this event. Cannot be NULL.

4.23.5.6 caerSpecialEventIsValid()

```
static bool caerSpecialEventIsValid (
            caerSpecialEventConst event )  [inline], [static]
```

Check if this special event is valid.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.23.5.7 caerSpecialEventPacketAllocate()

```
static caerSpecialEventPacket caerSpecialEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )  [inline], [static]
```

Allocate a new special events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid SpecialEventPacket handle or NULL on error.

4.23.5.8 caerSpecialEventPacketFindEventByType()

```
static caerSpecialEvent caerSpecialEventPacketFindEventByType (
            caerSpecialEventPacket packet,
            uint8_t type )  [inline], [static]
```

Get the first special event with the given event type in this event packet. This returns the first found event with that type ID, or NULL if we get to the end without finding any such event.

Parameters

packet	a valid SpecialEventPacket pointer. Cannot be NULL.
type	the special event type to search for.

Returns

the requested special event or NULL on error/not found.

4.23.5.9 caerSpecialEventPacketFindEventByTypeConst()

```
static caerSpecialEventConst caerSpecialEventPacketFindEventByTypeConst (
            caerSpecialEventPacketConst packet,
            uint8_t type )  [inline], [static]
```

Get the first special event with the given event type in this event packet. This returns the first found event with that type ID, or NULL if we get to the end without finding any such event. The returned event is read-only!

Parameters

packet	a valid SpecialEventPacket pointer. Cannot be NULL.
type	the special event type to search for.

Returns

the requested read-only special event or NULL on error/not found.

4.23.5.10 caerSpecialEventPacketFindValidEventByType()

```
static caerSpecialEvent caerSpecialEventPacketFindValidEventByType (
            caerSpecialEventPacket packet,
            uint8_t type )  [inline], [static]
```

Get the first valid special event with the given event type in this event packet. This returns the first found valid event with that type ID, or NULL if we get to the end without finding any such event.

Parameters

packet	a valid SpecialEventPacket pointer. Cannot be NULL.
type	the special event type to search for.

Returns

the requested valid special event or NULL on error/not found.

4.23.5.11 caerSpecialEventPacketFindValidEventByTypeConst()

```
static caerSpecialEventConst caerSpecialEventPacketFindValidEventByTypeConst (
            caerSpecialEventPacketConst packet,
            uint8_t type )  [inline], [static]
```

Get the first valid special event with the given event type in this event packet. This returns the first found valid event with that type ID, or NULL if we get to the end without finding any such event. The returned event is read-only!

Parameters

packet	a valid SpecialEventPacket pointer. Cannot be NULL.
type	the special event type to search for.

Returns

the requested read-only valid special event or NULL on error/not found.

4.23.5.12 caerSpecialEventPacketFromPacketHeader()

```
static caerSpecialEventPacket caerSpecialEventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into a Special event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.23.5.13 caerSpecialEventPacketFromPacketHeaderConst()

```
static caerSpecialEventPacketConst caerSpecialEventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only Special event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.23.5.14 caerSpecialEventPacketGetEvent()

```
static caerSpecialEvent caerSpecialEventPacketGetEvent (
            caerSpecialEventPacket packet,
            int32_t n )  [inline], [static]
```

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

Parameters

packet	a valid SpecialEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested special event. NULL on error.

4.23.5.15 caerSpecialEventPacketGetEventConst()

```
static caerSpecialEventConst caerSpecialEventPacketGetEventConst (
            caerSpecialEventPacketConst packet,
            int32_t n )  [inline], [static]
```

Get the special event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid SpecialEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only special event. NULL on error.

4.23.5.16 caerSpecialEventSetData()

```
static void caerSpecialEventSetData (
            caerSpecialEvent event,
            uint32_t data )  [inline], [static]
```

Set the special event data. Its meaning depends on the type. Current types that make use of it are (see 'enum caer_special_event_types'):

- DVS_ROW_ONLY: encodes the address of the row from the row-only event.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.
data	the special event data.

4.23.5.17 caerSpecialEventSetTimestamp()

```
static void caerSpecialEventSetTimestamp (
            caerSpecialEvent event,
            int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.23.5.18 caerSpecialEventSetType()

```
static void caerSpecialEventSetType (
            caerSpecialEvent event,
            uint8_t type )  [inline], [static]
```

Set the numerical special event type.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.
type	the special event type (see 'enum caer_special_event_types').

4.23.5.19 caerSpecialEventValidate()

```
static void caerSpecialEventValidate (
            caerSpecialEvent event,
            caerSpecialEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid SpecialEvent pointer. Cannot be NULL.
packet	the SpecialEventPacket pointer for the packet containing this event. Cannot be NULL.

4.23.5.20 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
            struct caer_special_event { uint32_t data;int32_t timestamp;}  )
```

Special event data structure definition. This contains the actual data, as well as the 32 bit event timestamp. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.23.5.21 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
            struct caer_special_event_packet { struct caer_event_packet_header packetHeader;struct
caer_special_event events[];}  )
```

Special event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.24 events/spike.h File Reference

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

Macros

- #define CAER_SPIKE_ITERATOR_ALL_START(SPIKE_PACKET)
- #define CAER_SPIKE_CONST_ITERATOR_ALL_START(SPIKE_PACKET)
- #define CAER_SPIKE_ITERATOR_ALL_END }
- #define CAER_SPIKE_ITERATOR_VALID_START(SPIKE_PACKET)
- #define CAER_SPIKE_CONST_ITERATOR_VALID_START(SPIKE_PACKET)
- #define CAER_SPIKE_ITERATOR_VALID_END }
- #define CAER_SPIKE_REVERSE_ITERATOR_ALL_START(SPIKE_PACKET)
- #define CAER_SPIKE_CONST_REVERSE_ITERATOR_ALL_START(SPIKE_PACKET)
- #define CAER_SPIKE_REVERSE_ITERATOR_ALL_END }
- #define CAER_SPIKE_REVERSE_ITERATOR_VALID_START(SPIKE_PACKET)
- #define CAER_SPIKE_CONST_REVERSE_ITERATOR_VALID_START(SPIKE_PACKET)
- #define CAER_SPIKE_REVERSE_ITERATOR_VALID_END }

- #define SPIKE_SOURCE_CORE_ID_SHIFT 1
- #define SPIKE_SOURCE_CORE_ID_MASK 0x0000001F
- #define SPIKE_CHIP_ID_SHIFT 6
- #define SPIKE_CHIP_ID_MASK 0x0000003F
- #define SPIKE_NEURON_ID_SHIFT 12
- #define SPIKE_NEURON_ID_MASK 0x000FFFFF

Typedefs

- typedef struct caer_spike_event ∗ caerSpikeEvent
- typedef const struct caer_spike_event ∗ **caerSpikeEventConst**
- typedef struct caer_spike_event_packet ∗ caerSpikeEventPacket
- typedef const struct caer_spike_event_packet ∗ **caerSpikeEventPacketConst**

Functions

- PACKED_STRUCT (struct caer_spike_event { uint32_t data;int32_t timestamp;})
- PACKED_STRUCT (struct caer_spike_event_packet { struct caer_event_packet_header packetHeader;struct caer_spike_event events[ ];})
- static caerSpikeEventPacket caerSpikeEventPacketAllocate (int32_t eventCapacity, int16_t eventSource, int32_t tsOverflow)
- static caerSpikeEventPacket caerSpikeEventPacketFromPacketHeader (caerEventPacketHeader header)
- static caerSpikeEventPacketConst caerSpikeEventPacketFromPacketHeaderConst (caerEventPacket↩HeaderConst header)
- static caerSpikeEvent caerSpikeEventPacketGetEvent (caerSpikeEventPacket packet, int32_t n)
- static caerSpikeEventConst caerSpikeEventPacketGetEventConst (caerSpikeEventPacketConst packet, int32_t n)
- static int32_t caerSpikeEventGetTimestamp (caerSpikeEventConst event)

- static int64_t caerSpikeEventGetTimestamp64 (caerSpikeEventConst event, caerSpikeEventPacketConst packet)
- static void caerSpikeEventSetTimestamp (caerSpikeEvent event, int32_t timestamp)
- static bool caerSpikeEventIsValid (caerSpikeEventConst event)
- static void caerSpikeEventValidate (caerSpikeEvent event, caerSpikeEventPacket packet)
- static void caerSpikeEventInvalidate (caerSpikeEvent event, caerSpikeEventPacket packet)
- static uint8_t caerSpikeEventGetSourceCoreID (caerSpikeEventConst event)
- static void caerSpikeEventSetSourceCoreID (caerSpikeEvent event, uint8_t sourceCoreID)
- static uint8_t caerSpikeEventGetChipID (caerSpikeEventConst event)
- static void caerSpikeEventSetChipID (caerSpikeEvent event, uint8_t chipID)
- static uint32_t caerSpikeEventGetNeuronID (caerSpikeEventConst event)
- static void caerSpikeEventSetNeuronID (caerSpikeEvent event, uint32_t neuronID)

4.24.1 Detailed Description

Spike Events format definition and handling functions. This contains spikes generated by a neuron-array chip.

4.24.2 Macro Definition Documentation

4.24.2.1 CAER_SPIKE_CONST_ITERATOR_ALL_START

```
#define CAER_SPIKE_CONST_ITERATOR_ALL_START(
                SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = 0;                                                    \
         caerSpikeIteratorCounter < caerEventPacketHeaderGetEventNumber(
      &(SPIKE_PACKET)->packetHeader); \
         caerSpikeIteratorCounter++) {                                                            \
        caerSpikeEventConst caerSpikeIteratorElement                                              \
            = caerSpikeEventPacketGetEventConst(SPIKE_PACKET,
      caerSpikeIteratorCounter);
```

Const-Iterator over all Spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpikeIteratorElement' variable of type caerSpikeEvent↩Const.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.2 CAER_SPIKE_CONST_ITERATOR_VALID_START

```
#define CAER_SPIKE_CONST_ITERATOR_VALID_START(
                SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = 0;                                                   \
         caerSpikeIteratorCounter < caerEventPacketHeaderGetEventNumber(
      &(SPIKE_PACKET)->packetHeader); \
         caerSpikeIteratorCounter++) {                                                            \
        caerSpikeEventConst caerSpikeIteratorElement                                              \
            = caerSpikeEventPacketGetEventConst(SPIKE_PACKET,
      caerSpikeIteratorCounter);                    \
        if (!caerSpikeEventIsValid(caerSpikeIteratorElement)) {
                                \
            continue;                                                                             \
        }
```

Const-Iterator over only the valid Spike events in a packet. Returns the current index in the 'caerSpikeIterator↩ Counter' variable of type 'int32_t' and the current read-only event in the 'caerSpikeIteratorElement' variable of type caerSpikeEventConst.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.3 CAER_SPIKE_CONST_REVERSE_ITERATOR_ALL_START

```
#define CAER_SPIKE_CONST_REVERSE_ITERATOR_ALL_START(
                SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(SPIKE_PACKET)->packetHeader) - 1; \
         caerSpikeIteratorCounter >= 0; caerSpikeIteratorCounter--) {
               \
        caerSpikeEventConst caerSpikeIteratorElement
               \
            = caerSpikeEventPacketGetEventConst(SPIKE_PACKET,
      caerSpikeIteratorCounter);
```

Const-Reverse iterator over all spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpikeIteratorElement' variable of type caer↩ SpikeEventConst.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.4 CAER_SPIKE_CONST_REVERSE_ITERATOR_VALID_START

```
#define CAER_SPIKE_CONST_REVERSE_ITERATOR_VALID_START(
                SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(SPIKE_PACKET)->packetHeader) - 1; \
         caerSpikeIteratorCounter >= 0; caerSpikeIteratorCounter--) {
               \
        caerSpikeEventConst caerSpikeIteratorElement
               \
            = caerSpikeEventPacketGetEventConst(SPIKE_PACKET,
      caerSpikeIteratorCounter);                  \
        if (!caerSpikeEventIsValid(caerSpikeIteratorElement)) {
                                    \
            continue;
               \
        }
```

Const-Reverse iterator over only the valid spike events in a packet. Returns the current index in the 'caerSpike↩ IteratorCounter' variable of type 'int32_t' and the current read-only event in the 'caerSpikeIteratorElement' variable of type caerSpikeEventConst.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.5 CAER_SPIKE_ITERATOR_ALL_END

```
#define CAER_SPIKE_ITERATOR_ALL_END }
```

Iterator close statement.

4.24.2.6 CAER_SPIKE_ITERATOR_ALL_START

```
#define CAER_SPIKE_ITERATOR_ALL_START(
                SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = 0;                                              \
         caerSpikeIteratorCounter < caerEventPacketHeaderGetEventNumber(
      &(SPIKE_PACKET)->packetHeader); \
         caerSpikeIteratorCounter++) {                                                      \
        caerSpikeEvent caerSpikeIteratorElement = caerSpikeEventPacketGetEvent(
      SPIKE_PACKET, caerSpikeIteratorCounter);
```

Iterator over all Spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpikeIteratorElement' variable of type caerSpikeEvent.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.7 CAER_SPIKE_ITERATOR_VALID_END

```
#define CAER_SPIKE_ITERATOR_VALID_END }
```

Iterator close statement.

4.24.2.8 CAER_SPIKE_ITERATOR_VALID_START

```
#define CAER_SPIKE_ITERATOR_VALID_START(
                SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = 0;                                              \
         caerSpikeIteratorCounter < caerEventPacketHeaderGetEventNumber(
      &(SPIKE_PACKET)->packetHeader); \
         caerSpikeIteratorCounter++) {                                                      \
        caerSpikeEvent caerSpikeIteratorElement                                             \
            = caerSpikeEventPacketGetEvent(SPIKE_PACKET,
      caerSpikeIteratorCounter);                         \
        if (!caerSpikeEventIsValid(caerSpikeIteratorElement)) {
                        \
            continue;                                                                       \
        }
```

Iterator over only the valid Spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpikeIteratorElement' variable of type caerSpikeEvent.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.9 CAER_SPIKE_REVERSE_ITERATOR_ALL_END

```
#define CAER_SPIKE_REVERSE_ITERATOR_ALL_END }
```

Reverse iterator close statement.

4.24.2.10 CAER_SPIKE_REVERSE_ITERATOR_ALL_START

```
#define CAER_SPIKE_REVERSE_ITERATOR_ALL_START(
                SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(SPIKE_PACKET)->packetHeader) - 1; \
         caerSpikeIteratorCounter >= 0; caerSpikeIteratorCounter--) {
               \
        caerSpikeEvent caerSpikeIteratorElement = caerSpikeEventPacketGetEvent(
      SPIKE_PACKET, caerSpikeIteratorCounter);
```

Reverse iterator over all spike events in a packet. Returns the current index in the 'caerSpikeIteratorCounter' variable of type 'int32_t' and the current event in the 'caerSpikeIteratorElement' variable of type caerSpikeEvent.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.11 CAER_SPIKE_REVERSE_ITERATOR_VALID_END

```
#define CAER_SPIKE_REVERSE_ITERATOR_VALID_END }
```

Reverse iterator close statement.

4.24.2.12 CAER_SPIKE_REVERSE_ITERATOR_VALID_START

```
#define CAER_SPIKE_REVERSE_ITERATOR_VALID_START(
                SPIKE_PACKET )
```

Value:

```
for (int32_t caerSpikeIteratorCounter = caerEventPacketHeaderGetEventNumber
      (&(SPIKE_PACKET)->packetHeader) - 1; \
         caerSpikeIteratorCounter >= 0; caerSpikeIteratorCounter--) {
               \
        caerSpikeEvent caerSpikeIteratorElement
               \
            = caerSpikeEventPacketGetEvent(SPIKE_PACKET,
      caerSpikeIteratorCounter);                            \
        if (!caerSpikeEventIsValid(caerSpikeIteratorElement)) {
                                     \
            continue;
               \
        }
```

Reverse iterator over only the valid spike events in a packet. Returns the current index in the 'caerSpikeIterator↩ Counter' variable of type 'int32_t' and the current event in the 'caerSpikeIteratorElement' variable of type caer↩ SpikeEvent.

SPIKE_PACKET: a valid SpikeEventPacket pointer. Cannot be NULL.

4.24.2.13 SPIKE_CHIP_ID_MASK

```
#define SPIKE_CHIP_ID_MASK 0x0000003F
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see 'common.h' for more details.

4.24.2.14 SPIKE_CHIP_ID_SHIFT

```
#define SPIKE_CHIP_ID_SHIFT 6
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see 'common.h' for more details.

4.24.2.15 SPIKE_NEURON_ID_MASK

```
#define SPIKE_NEURON_ID_MASK 0x000FFFFF
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see 'common.h' for more details.

4.24.2.16 SPIKE_NEURON_ID_SHIFT

```
#define SPIKE_NEURON_ID_SHIFT 12
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see 'common.h' for more details.

4.24.2.17 SPIKE_SOURCE_CORE_ID_MASK

```
#define SPIKE_SOURCE_CORE_ID_MASK 0x0000001F
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see 'common.h' for more details.

4.24.2.18 SPIKE_SOURCE_CORE_ID_SHIFT

```
#define SPIKE_SOURCE_CORE_ID_SHIFT 1
```

Shift and mask values for spike information associated with a Spike event. 32 core IDs, 64 chip IDs and up to a million neuron IDs are supported. Bit 0 is the valid mark, see 'common.h' for more details.

4.24.3 Typedef Documentation

4.24.3.1 caerSpikeEvent

```
typedef struct caer_spike_event* caerSpikeEvent
```

Type for pointer to Spike event data structure.

4.24.3.2 caerSpikeEventPacket

```
typedef struct caer_spike_event_packet* caerSpikeEventPacket
```

Type for pointer to Spike event packet data structure.

4.24.4 Function Documentation

4.24.4.1 caerSpikeEventGetChipID()

```
static uint8_t caerSpikeEventGetChipID (
             caerSpikeEventConst event )  [inline], [static]
```

Get the chip ID.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

the Spike's chip ID.

4.24.4.2 caerSpikeEventGetNeuronID()

```
static uint32_t caerSpikeEventGetNeuronID (
             caerSpikeEventConst event )  [inline], [static]
```

Get the neuron ID.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

the Spike's neuron ID.

4.24.4.3 caerSpikeEventGetSourceCoreID()

```
static uint8_t caerSpikeEventGetSourceCoreID (
            caerSpikeEventConst event )  [inline], [static]
```

Get the source core ID.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

the Spike's source core ID.

4.24.4.4 caerSpikeEventGetTimestamp()

```
static int32_t caerSpikeEventGetTimestamp (
            caerSpikeEventConst event )  [inline], [static]
```

Get the 32bit event timestamp, in microseconds. Be aware that this wraps around! You can either ignore this fact, or handle the special 'TIMESTAMP_WRAP' event that is generated when this happens, or use the 64bit timestamp which never wraps around. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

this event's 32bit microsecond timestamp.

4.24.4.5 caerSpikeEventGetTimestamp64()

```
static int64_t caerSpikeEventGetTimestamp64 (
            caerSpikeEventConst event,
            caerSpikeEventPacketConst packet )  [inline], [static]
```

Get the 64bit event timestamp, in microseconds. See 'caerEventPacketHeaderGetEventTSOverflow()' documentation for more details on the 64bit timestamp.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
packet	the SpikeEventPacket pointer for the packet containing this event. Cannot be NULL.

Returns

this event's 64bit microsecond timestamp.

4.24.4.6 caerSpikeEventInvalidate()

```
static void caerSpikeEventInvalidate (
            caerSpikeEvent event,
            caerSpikeEventPacket packet )  [inline], [static]
```

Invalidate the current event by setting its valid bit to false and decreasing the number of valid events held in the packet. Only works with events that are already valid!

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
packet	the SpikeEventPacket pointer for the packet containing this event. Cannot be NULL.

4.24.4.7 caerSpikeEventIsValid()

```
static bool caerSpikeEventIsValid (
            caerSpikeEventConst event )  [inline], [static]
```

Check if this Spike event is valid.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.

Returns

true if valid, false if not.

4.24.4.8 caerSpikeEventPacketAllocate()

```
static caerSpikeEventPacket caerSpikeEventPacketAllocate (
            int32_t eventCapacity,
            int16_t eventSource,
            int32_t tsOverflow )  [inline], [static]
```

Allocate a new Spike events packet. Use free() to reclaim this memory.

Parameters

eventCapacity	the maximum number of events this packet will hold.
eventSource	the unique ID representing the source/generator of this packet.
tsOverflow	the current timestamp overflow counter value for this packet.

Returns

a valid SpikeEventPacket handle or NULL on error.

4.24.4.9 caerSpikeEventPacketFromPacketHeader()

```
static caerSpikeEventPacket caerSpikeEventPacketFromPacketHeader (
            caerEventPacketHeader header )  [inline], [static]
```

Transform a generic event packet header into a Spike event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid event packet header pointer. Cannot be NULL.

Returns

a properly converted, typed event packet pointer.

4.24.4.10 caerSpikeEventPacketFromPacketHeaderConst()

```
static caerSpikeEventPacketConst caerSpikeEventPacketFromPacketHeaderConst (
            caerEventPacketHeaderConst header )  [inline], [static]
```

Transform a generic read-only event packet header into a read-only Spike event packet. This takes care of proper casting and checks that the packet type really matches the intended conversion type.

Parameters

header	a valid read-only event packet header pointer. Cannot be NULL.

Returns

a properly converted, read-only typed event packet pointer.

4.24.4.11 caerSpikeEventPacketGetEvent()

```
static caerSpikeEvent caerSpikeEventPacketGetEvent (
            caerSpikeEventPacket packet,
            int32_t n )  [inline], [static]
```

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

Parameters

packet	a valid SpikeEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested Spike event. NULL on error.

4.24.4.12 caerSpikeEventPacketGetEventConst()

```
static caerSpikeEventConst caerSpikeEventPacketGetEventConst (
            caerSpikeEventPacketConst packet,
            int32_t n )  [inline], [static]
```

Get the Spike event at the given index from the event packet. This is a read-only event, do not change its contents in any way!

Parameters

packet	a valid SpikeEventPacket pointer. Cannot be NULL.
n	the index of the returned event. Must be within [0,eventCapacity[bounds.

Returns

the requested read-only Spike event. NULL on error.

4.24.4.13 caerSpikeEventSetChipID()

```
static void caerSpikeEventSetChipID (
            caerSpikeEvent event,
            uint8_t chipID )  [inline], [static]
```

Set the chip ID.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
chipID	the Spike's chip ID.

4.24.4.14 caerSpikeEventSetNeuronID()

```
static void caerSpikeEventSetNeuronID (
             caerSpikeEvent event,
             uint32_t neuronID )  [inline], [static]
```

Set the neuron ID.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
neuronID	the Spike's neuron ID.

4.24.4.15 caerSpikeEventSetSourceCoreID()

```
static void caerSpikeEventSetSourceCoreID (
             caerSpikeEvent event,
             uint8_t sourceCoreID )  [inline], [static]
```

Set the source core ID.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
sourceCoreID	the Spike's source core ID.

4.24.4.16 caerSpikeEventSetTimestamp()

```
static void caerSpikeEventSetTimestamp (
             caerSpikeEvent event,
             int32_t timestamp )  [inline], [static]
```

Set the 32bit event timestamp, the value has to be in microseconds.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
timestamp	a positive 32bit microsecond timestamp.

4.24.4.17 caerSpikeEventValidate()

```
static void caerSpikeEventValidate (
             caerSpikeEvent event,
             caerSpikeEventPacket packet )  [inline], [static]
```

Validate the current event by setting its valid bit to true and increasing the event packet's event count and valid event count. Only works on events that are invalid. DO NOT CALL THIS AFTER HAVING PREVIOUSLY ALREADY INVALIDATED THIS EVENT, the total count will be incorrect.

Parameters

event	a valid SpikeEvent pointer. Cannot be NULL.
packet	the SpikeEventPacket pointer for the packet containing this event. Cannot be NULL.

4.24.4.18 PACKED_STRUCT() [1/2]

```
PACKED_STRUCT (
             struct caer_spike_event { uint32_t data;int32_t timestamp;}  )
```

Spike event data structure definition. This contains the core ID, the neuron ID and the timestamp of the received spike, together with the usual validity mark. Signed integers are used for fields that are to be interpreted directly, for compatibility with languages that do not have unsigned integer types, such as Java.

4.24.4.19 PACKED_STRUCT() [2/2]

```
PACKED_STRUCT (
             struct caer_spike_event_packet { struct caer_event_packet_header packetHeader;struct
caer_spike_event events[];}  )
```

Spike event packet data structure definition. EventPackets are always made up of the common packet header, followed by 'eventCapacity' events. Everything has to be in one contiguous memory block.

4.25 filters/dvs_noise.h File Reference

```
#include "../events/polarity.h"
```

Data Structures

- struct caer_filter_dvs_pixel

Macros

- #define CAER_FILTER_DVS_HOTPIXEL_LEARN 0
- #define CAER_FILTER_DVS_HOTPIXEL_TIME 1
- #define CAER_FILTER_DVS_HOTPIXEL_COUNT 2
- #define CAER_FILTER_DVS_HOTPIXEL_ENABLE 3
- #define CAER_FILTER_DVS_HOTPIXEL_STATISTICS 4
- #define CAER_FILTER_DVS_HOTPIXEL_STATISTICS_ON 17
- #define CAER_FILTER_DVS_HOTPIXEL_STATISTICS_OFF 18
- #define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_ENABLE 5
- #define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TIME 6
- #define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS 7
- #define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS_ON 19
- #define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS_OFF 20
- #define CAER_FILTER_DVS_REFRACTORY_PERIOD_ENABLE 8
- #define CAER_FILTER_DVS_REFRACTORY_PERIOD_TIME 9
- #define CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS 10
- #define CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS_ON 21
- #define CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS_OFF 22
- #define CAER_FILTER_DVS_LOG_LEVEL 11
- #define CAER_FILTER_DVS_RESET 12
- #define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TWO_LEVELS 13
- #define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MIN 14
- #define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MAX 15
- #define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_CHECK_POLARITY 16

Typedefs

- typedef struct caer_filter_dvs_pixel ∗ caerFilterDVSPixel
- typedef struct caer_filter_dvs_noise ∗ caerFilterDVSNoise

Functions

- caerFilterDVSNoise caerFilterDVSNoiseInitialize (uint16_t sizeX, uint16_t sizeY)
- void caerFilterDVSNoiseDestroy (caerFilterDVSNoise noiseFilter)
- void caerFilterDVSNoiseApply (caerFilterDVSNoise noiseFilter, caerPolarityEventPacket polarity)
- void caerFilterDVSNoiseStatsApply (caerFilterDVSNoise noiseFilter, caerPolarityEventPacketConst polarity)
- bool caerFilterDVSNoiseConfigSet (caerFilterDVSNoise noiseFilter, uint8_t paramAddr, uint64_t param)
- bool caerFilterDVSNoiseConfigGet (caerFilterDVSNoise noiseFilter, uint8_t paramAddr, uint64_t ∗param)
- ssize_t caerFilterDVSNoiseGetHotPixels (caerFilterDVSNoise noiseFilter, caerFilterDVSPixel ∗hotPixels)

4.25.1 Detailed Description

The DVS noise filter combines a HotPixel filter (high activity pixels), a Background-Activity filter (uncorrelated events), and a Refractory Period filter (limit event rate of a pixel). The HotPixel and Background-Activity filters reduce noise due to transistor mismatch, the Refractory Period filter can reduce the event rate and is efficient to implement together with the Background-Activity filter, requiring only one pixel memory map for both. Please note that the filter is not thread-safe, all function calls should happen on the same thread, unless you take care that they never overlap.

4.25.2 Macro Definition Documentation

4.25.2.1 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_CHECK_POLARITY

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_CHECK_POLARITY 16
```

DVS Background-Activity Filter: whether polarity is considered when searching the neighbors for supporting activity.

4.25.2.2 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_ENABLE

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_ENABLE 5
```

DVS Background-Activity Filter: enable the background-activity filter, which tries to remove events caused by transistor leakage, by rejecting uncorrelated events.

4.25.2.3 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS 7
```

DVS Background-Activity Filter: number of events filtered out by the background-activity filter.

4.25.2.4 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS_OFF

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS_OFF 20
```

DVS Background-Activity Filter: number of OFF events filtered out by the background-activity filter.

4.25.2.5 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS_ON

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_STATISTICS_ON 19
```

DVS Background-Activity Filter: number of ON events filtered out by the background-activity filter.

4.25.2.6 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MAX

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MAX 15
```

DVS Background-Activity Filter: maximum number of pixels in the immediate neighborhood that can support the current pixel for it to be considered valid.

4.25.2.7 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MIN

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_SUPPORT_MIN 14
```

DVS Background-Activity Filter: minimum number of pixels in the immediate neighborhood that must support the current pixel for it to be considered valid.

4.25.2.8 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TIME

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TIME 6
```

DVS Background-Activity Filter: specify the time difference constant for the background-activity filter in microseconds. Events that do correlated within this time-frame are let through, while others are filtered out.

4.25.2.9 CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TWO_LEVELS

```
#define CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TWO_LEVELS 13
```

DVS Background-Activity Filter: repeat the background-activity check, that at least one neighbor pixel supports this pixel, on each pixel that supported the current pixel in turn, basically repeating the check for a second level of pixels.

4.25.2.10 CAER_FILTER_DVS_HOTPIXEL_COUNT

```
#define CAER_FILTER_DVS_HOTPIXEL_COUNT 2
```

DVS HotPixel Filter: Minimum number of events, during the given learning time, for a pixel to be considered hot.

4.25.2.11 CAER_FILTER_DVS_HOTPIXEL_ENABLE

```
#define CAER_FILTER_DVS_HOTPIXEL_ENABLE 3
```

DVS HotPixel Filter: Enable the hot pixel filter, filtering out the last learned hot pixels.

4.25.2.12 CAER_FILTER_DVS_HOTPIXEL_LEARN

```
#define CAER_FILTER_DVS_HOTPIXEL_LEARN 0
```

DVS HotPixel Filter: Turn on learning to determine which pixels are hot, meaning abnormally active within a certain time period. In the absence of external stimuli, the only pixels behaving as such must be noise. Once learning is enabled, do not disable it until completed. To verify completion, query this parameter and wait for it to switch from 'true' back to 'false'.

4.25.2.13 CAER_FILTER_DVS_HOTPIXEL_STATISTICS

```
#define CAER_FILTER_DVS_HOTPIXEL_STATISTICS 4
```

DVS HotPixel Filter: Number of events filtered out by the hot pixel filter.

4.25.2.14 CAER_FILTER_DVS_HOTPIXEL_STATISTICS_OFF

```
#define CAER_FILTER_DVS_HOTPIXEL_STATISTICS_OFF 18
```

DVS HotPixel Filter: Number of OFF events filtered out by the hot pixel filter.

4.25.2.15 CAER_FILTER_DVS_HOTPIXEL_STATISTICS_ON

```
#define CAER_FILTER_DVS_HOTPIXEL_STATISTICS_ON 17
```

DVS HotPixel Filter: Number of ON events filtered out by the hot pixel filter.

4.25.2.16 CAER_FILTER_DVS_HOTPIXEL_TIME

```
#define CAER_FILTER_DVS_HOTPIXEL_TIME 1
```

DVS HotPixel Filter: Minimum time (in µs) to accumulate events for during learning.

4.25.2.17 CAER_FILTER_DVS_LOG_LEVEL

```
#define CAER_FILTER_DVS_LOG_LEVEL 11
```

DVS Noise Filter: set a custom log-level for an instance of the DVS Noise filter.

4.25.2.18 CAER_FILTER_DVS_REFRACTORY_PERIOD_ENABLE

```
#define CAER_FILTER_DVS_REFRACTORY_PERIOD_ENABLE 8
```

DVS Refractory Period Filter: enable the refractory period filter, which limits the firing rate of pixels.

4.25.2.19 CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS

```
#define CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS 10
```

DVS Refractory Period Filter: number of events filtered out by the refractory period filter.

4.25.2.20 CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS_OFF

```
#define CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS_OFF 22
```

DVS Refractory Period Filter: number of OFF events filtered out by the refractory period filter.

4.25.2.21 CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS_ON

```
#define CAER_FILTER_DVS_REFRACTORY_PERIOD_STATISTICS_ON 21
```

DVS Refractory Period Filter: number of ON events filtered out by the refractory period filter.

4.25.2.22 CAER_FILTER_DVS_REFRACTORY_PERIOD_TIME

```
#define CAER_FILTER_DVS_REFRACTORY_PERIOD_TIME 9
```

DVS Refractory Period Filter: specify the time constant for the refractory period filter. Pixels will be inhibited from generating new events during this time after the last even has fired.

4.25.2.23 CAER_FILTER_DVS_RESET

```
#define CAER_FILTER_DVS_RESET 12
```

DVS Noise Filter: reset this instance of the filter to its initial state, forgetting any learned hot pixels and clearing the timestamp map and the statistics. This does not change or reset the configuration.

4.25.3 Typedef Documentation

4.25.3.1 caerFilterDVSNoise

```
typedef struct caer_filter_dvs_noise* caerFilterDVSNoise
```

Pointer to DVS noise filter structure (private).

4.25.3.2 caerFilterDVSPixel

```
typedef struct caer_filter_dvs_pixel* caerFilterDVSPixel
```

Pointer to DVS pixel address structure.

4.25.4 Function Documentation

4.25.4.1 caerFilterDVSNoiseApply()

```
void caerFilterDVSNoiseApply (
             caerFilterDVSNoise noiseFilter,
             caerPolarityEventPacket polarity )
```

Apply the DVS noise filter to the given polarity events packet. This will filter out events by marking them as invalid, depending on the given filter configuration.

Parameters

noiseFilter	a valid DVS noise filter instance.
polarity	a valid polarity event packet. If NULL, no operation is performed.

4.25.4.2 caerFilterDVSNoiseConfigGet()

```
bool caerFilterDVSNoiseConfigGet (
            caerFilterDVSNoise noiseFilter,
            uint8_t paramAddr,
            uint64_t * param )
```

Get DVS noise filter configuration parameters.

Parameters

noiseFilter	a valid DVS noise filter instance.
paramAddr	a configuration parameter address, see defines CAER_FILTER_DVS_*.
param	a pointer to a configuration parameter value integer, in which to store the current value.

Returns

true if operation successful, false otherwise.

4.25.4.3 caerFilterDVSNoiseConfigSet()

```
bool caerFilterDVSNoiseConfigSet (
            caerFilterDVSNoise noiseFilter,
            uint8_t paramAddr,
            uint64_t param )
```

Set DVS noise filter configuration parameters.

Parameters

noiseFilter	a valid DVS noise filter instance.
paramAddr	a configuration parameter address, see defines CAER_FILTER_DVS_*.
param	a configuration parameter value integer.

Returns

true if operation successful, false otherwise.

4.25.4.4 caerFilterDVSNoiseDestroy()

```
void caerFilterDVSNoiseDestroy (
            caerFilterDVSNoise noiseFilter )
```

Destroy a DVS noise filter instance and free its memory.

Parameters

noiseFilter	a valid DVS noise filter instance.

4.25.4.5 caerFilterDVSNoiseGetHotPixels()

```
ssize_t caerFilterDVSNoiseGetHotPixels (
            caerFilterDVSNoise noiseFilter,
            caerFilterDVSPixel * hotPixels )
```

Get an array of currently learned hot pixels, in order of activity (most active first, least active last). Useful for working with hardware-based pixel filtering (FPGA/CPLD).

Parameters

noiseFilter	a valid DVS noise filter instance.
hotPixels	array of DVS pixel addresses, sorted by activity (most active first). Memory will be allocated for it automatically. On error, the pointer is set to NULL. Remember to free() the memory once done!

Returns

number of hot pixels in array, 0 if no hot pixels were found; or -1 if an error occurred.

4.25.4.6 caerFilterDVSNoiseInitialize()

```
caerFilterDVSNoise caerFilterDVSNoiseInitialize (
            uint16_t sizeX,
            uint16_t sizeY )
```

Allocate memory and initialize the DVS noise filter. This filter combines a HotPixel filter (high activity pixels), a Background-Activity filter (uncorrelated events), and a Refractory Period filter (limit event rate of a pixel). The HotPixel and Background-Activity filters reduce noise due to transistor mismatch, the Refractory Period filter can reduce the event rate and is efficient to implement together with the Background-Activity filter, requiring only one pixel memory map for both. At initialization, all filters are disabled. You must configure and enable them using caerFilterDVSNoiseConfigSet(). You must specify the maximum resolution at initialization, as it is used to set up efficient lookup tables.

Parameters

sizeX	maximum X axis resolution.
sizeY	maximum Y axis resolution.

Returns

DVS noise filter instance, NULL on error.

4.25.4.7 caerFilterDVSNoiseStatsApply()

```
void caerFilterDVSNoiseStatsApply (
            caerFilterDVSNoise noiseFilter,
            caerPolarityEventPacketConst polarity )
```

Apply the DVS noise filter to the given polarity events packet. This will only gather statistics on the noise, without changing the event packet at all!

Parameters

noiseFilter	a valid DVS noise filter instance.
polarity	a valid polarity event packet. If NULL, no operation is performed.

4.26 frame_utils.h File Reference

```
#include "events/frame.h"
```

Enumerations

- enum **caer_frame_utils_demosaic_types** {
  DEMOSAIC_STANDARD = 0, **DEMOSAIC_TO_GRAY** = 1, **DEMOSAIC_OPENCV_STANDARD** = 2, **DE↩ MOSAIC_OPENCV_EDGE_AWARE** = 3,
  DEMOSAIC_OPENCV_TO_GRAY = 4 }
- enum **caer_frame_utils_contrast_types** { **CONTRAST_STANDARD** = 0, **CONTRAST_OPENCV_NOR↩ MALIZATION** = 1, **CONTRAST_OPENCV_HISTOGRAM_EQUALIZATION** = 2, **CONTRAST_OPENCV_↩ CLAHE** = 3 }

Functions

- void **caerFrameUtilsDemosaic** (caerFrameEventConst inputFrame, caerFrameEvent outputFrame, enum caer_frame_utils_demosaic_types demosaicType)
- void **caerFrameUtilsContrast** (caerFrameEventConst inputFrame, caerFrameEvent outputFrame, enum caer_frame_utils_contrast_types contrastType)

4.26.1 Detailed Description

Functions for frame enhancement and demosaicing. Basic variants that don't require any external dependencies, such as OpenCV. Use of the OpenCV variants is recommended for quality and performance, and can optionally be enabled at build-time.

4.27 libcaer.h File Reference

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

Macros

- #define **PACKED_STRUCT**(STRUCT_DECLARATION) STRUCT_DECLARATION
- #define **DEPRECATED_FUNCTION**(DEPR_MSG)
- #define LIBCAER_VERSION ((3 ∗ 10000) + (0 ∗ 100) + 0)
- #define LIBCAER_NAME_STRING "libcaer"
- #define LIBCAER_VERSION_STRING "3.0.0"
- #define LIBCAER_HAVE_SERIALDEV 1
- #define LIBCAER_HAVE_OPENCV 1
- #define U8T(X) ((uint8_t) (X))
- #define U16T(X) ((uint16_t) (X))
- #define U32T(X) ((uint32_t) (X))
- #define U64T(X) ((uint64_t) (X))
- #define I8T(X) ((int8_t) (X))
- #define I16T(X) ((int16_t) (X))
- #define I32T(X) ((int32_t) (X))
- #define I64T(X) ((int64_t) (X))
- #define MASK_NUMBITS32(X) U32T(U32T(U32T(1) << X) - 1)
- #define MASK_NUMBITS64(X) U64T(U64T(U64T(1) << X) - 1)
- #define SWAP_VAR(type, x, y) { type tmpv; tmpv = (x); (x) = (y); (y) = tmpv; }

- #define CLEAR_NUMBITS32(VAR, SHIFT, MASK) (VAR) &= 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))

Enumerations

- enum caer_error_codes {
  CAER_ERROR_MEMORY_ALLOCATION = -1, **CAER_ERROR_RESOURCE_ALLOCATION** = -2, **CAE↩R_ERROR_OPEN_ACCESS** = -3, **CAER_ERROR_COMMUNICATION** = -4,
  CAER_ERROR_FW_VERSION = -5, **CAER_ERROR_LOGIC_VERSION** = -6 }

Functions

- static bool caerStrEquals (const char ∗s1, const char ∗s2)
- static bool caerStrEqualsUpTo (const char ∗s1, const char ∗s2, size_t len)
- static void caerIntegerToByteArray (const uint32_t integer, uint8_t ∗byteArray, const uint8_t byteArrayLength)
- static uint32_t caerByteArrayToInteger (const uint8_t ∗byteArray, const uint8_t byteArrayLength)

4.27.1 Detailed Description

Main libcaer header; provides inclusions for common system functions and definitions for useful macros used often in the code. Also includes the logging functions and definitions and several useful static inline functions for string comparison and byte array manipulation. When including libcaer, please make sure to always use the full path, ie. #include <libcaer/libcaer.h> and not just #include <libcaer.h>.

4.27.2 Macro Definition Documentation

4.27.2.1 CLEAR_NUMBITS16

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

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

4.27.2.2 CLEAR_NUMBITS32

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

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

4.27.2.3 CLEAR_NUMBITS8

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

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

4.27.2.4 GET_NUMBITS16

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

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

4.27.2.5 GET_NUMBITS32

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

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

4.27.2.6 GET_NUMBITS8

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

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

4.27.2.7 I16T

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

Cast argument to int16_t (16bit signed integer).

4.27.2.8 I32T

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

Cast argument to int32_t (32bit signed integer).

4.27.2.9 I64T

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

Cast argument to int64_t (64bit signed integer).

4.27.2.10 I8T

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

Cast argument to int8_t (8bit signed integer).

4.27.2.11 LIBCAER_HAVE_OPENCV

```
#define LIBCAER_HAVE_OPENCV 1
```

libcaer OpenCV support.

4.27.2.12 LIBCAER_HAVE_SERIALDEV

```
#define LIBCAER_HAVE_SERIALDEV 1
```

libcaer serial devices support.

4.27.2.13 LIBCAER_NAME_STRING

```
#define LIBCAER_NAME_STRING "libcaer"
```

libcaer name string.

4.27.2.14 LIBCAER_VERSION

```
#define LIBCAER_VERSION ((3 * 10000) + (0 * 100) + 0)
```

libcaer version (MAJOR ∗ 10000 + MINOR ∗ 100 + PATCH).

4.27.2.15 LIBCAER_VERSION_STRING

```
#define LIBCAER_VERSION_STRING "3.0.0"
```

libcaer version string.

4.27.2.16 MASK_NUMBITS32

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

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

4.27.2.17 MASK_NUMBITS64

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

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

4.27.2.18 SET_NUMBITS16

```
#define SET_NUMBITS16(
              VAR,
              SHIFT,
              MASK,
              VALUE ) (VAR) |= htole16(U16T((U16T(VALUE) & (MASK)) << (SHIFT)))
```

Set bits given by mask (amount) and shift (position) to a value.

4.27.2.19 SET_NUMBITS32

```
#define SET_NUMBITS32(
              VAR,
              SHIFT,
              MASK,
              VALUE ) (VAR) |= htole32(U32T((U32T(VALUE) & (MASK)) << (SHIFT)))
```

Set bits given by mask (amount) and shift (position) to a value.

4.27.2.20 SET_NUMBITS8

```
#define SET_NUMBITS8(
              VAR,
              SHIFT,
              MASK,
              VALUE ) (VAR) |= U8T((U8T(VALUE) & (MASK)) << (SHIFT))
```

Set bits given by mask (amount) and shift (position) to a value.

4.27.2.21 SWAP_VAR

```
#define SWAP_VAR(
              type,
              x,
              y ) { type tmpv; tmpv = (x); (x) = (y); (y) = tmpv; }
```

Swap the two values of the two variables X and Y, of a common type TYPE.

4.27.2.22 U16T

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

Cast argument to uint16_t (16bit unsigned integer).

4.27.2.23 U32T

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

Cast argument to uint32_t (32bit unsigned integer).

4.27.2.24 U64T

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

Cast argument to uint64_t (64bit unsigned integer).

4.27.2.25 U8T

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

Cast argument to uint8_t (8bit unsigned integer).

4.27.3 Enumeration Type Documentation

4.27.3.1 caer_error_codes

```
enum caer_error_codes
```

Error codes, used for the errno variable to give more precise information on errors, in addition to the logging output. All functions setting errno do note so in their documentation.

4.27.4 Function Documentation

4.27.4.1 caerByteArrayToInteger()

```
static uint32_t caerByteArrayToInteger (
             const uint8_t * byteArray,
             const uint8_t byteArrayLength )  [inline], [static]
```

Convert an unsigned byte array of up to four bytes into a 32bit unsigned integer. The byte array length decides how many resulting bits in the integer are set, and the single bytes are placed in the integer following big-endian ordering.

Parameters

byteArray	pointer to the byte array with parts of the value stored.
byteArrayLength	length of the array from which to convert.

Returns

integer representing the value stored in the byte array.

4.27.4.2 caerIntegerToByteArray()

```
static void caerIntegerToByteArray (
            const uint32_t integer,
            uint8_t * byteArray,
            const uint8_t byteArrayLength )  [inline], [static]
```

Convert a 32bit unsigned integer into an unsigned byte array of up to four bytes. The integer will be stored in big-endian order, and the length will specify how many bits to convert, starting from the lowest bit.

Parameters

integer	the integer to convert.
byteArray	pointer to the byte array in which to store the converted values.
byteArrayLength	length of the byte array to convert to.

4.27.4.3 caerStrEquals()

```
static bool caerStrEquals (
            const char * s1,
            const char * s2 )  [inline], [static]
```

Compare two strings for equality.

Parameters

s1	the first string, cannot be NULL.
s2	the second string, cannot be NULL.

Returns

true if equal, false otherwise.

4.27.4.4 caerStrEqualsUpTo()

```
static bool caerStrEqualsUpTo (
            const char * s1,
            const char * s2,
            size_t len ) [inline], [static]
```

Compare two strings for equality, up to a specified maximum length.

Parameters

s1	the first string, cannot be NULL.
s2	the second string, cannot be NULL.
len	maximum comparison length, cannot be zero.

Returns

true if equal, false otherwise.

4.28 log.h File Reference

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

Macros

- #define **ATTRIBUTE_FORMAT**(N)
- #define **ATTRIBUTE_FORMAT_VA**(N)

Enumerations

- enum caer_log_level {
  CAER_LOG_EMERGENCY = 0, **CAER_LOG_ALERT** = 1, **CAER_LOG_CRITICAL** = 2, **CAER_LOG_E↩RROR** = 3,
  CAER_LOG_WARNING = 4, **CAER_LOG_NOTICE** = 5, **CAER_LOG_INFO** = 6, **CAER_LOG_DEBUG** = 7
  }

Functions

- void caerLogLevelSet (enum caer_log_level logLevel)
- enum caer_log_level caerLogLevelGet (void)
- void caerLogFileDescriptorsSet (int fd1, int fd2)
- int caerLogFileDescriptorsGetFirst (void)
- int caerLogFileDescriptorsGetSecond (void)
- void caerLogDisable (bool disableLogging)
- bool caerLogDisabled (void)
- void caerLog (enum caer_log_level logLevel, const char ∗subSystem, const char ∗format,...) ATTRIBUTE↩_FORMAT(3)
- void caerLogVA (enum caer_log_level logLevel, const char ∗subSystem, const char ∗format, va_list args) ATTRIBUTE_FORMAT_VA(3)
- void caerLogVAFull (int logFileDescriptor1, int logFileDescriptor2, uint8_t systemLogLevel, enum caer_log_level logLevel, const char ∗subSystem, const char ∗format, va_list args) ATTRIBUTE_FOR↩MAT_VA(6)

4.28.1 Detailed Description

Logging functions to print useful messages for the user.

4.28.2 Enumeration Type Documentation

4.28.2.1 caer_log_level

```
enum caer_log_level
```

Log levels for caerLog() logging function. Log messages only get printed if their log level is equal or above the global system log level, which can be set with caerLogLevelSet(). The default log level is CAER_LOG_ERROR. CAER_LOG_EMERGENCY is the most urgent log level and will always be printed, while CAER_LOG_DEBUG is the least urgent log level and will only be delivered if configured by the user.

4.28.3 Function Documentation

4.28.3.1 caerLog()

```
void caerLog (
            enum caer_log_level logLevel,
            const char * subSystem,
            const char * format,
             ... )
```

Main logging function. This function takes messages, formats them and sends them out to a file descriptor, respecting the system-wide log level setting and prepending the current time, the log level and a user-specified common string to the actual formatted output. The format is specified exactly as with the printf() family of functions. Please see their manual-page for more information.

Parameters

logLevel	the message-specific log level.
subSystem	a common, user-specified string to prepend before the message.
format	the message format string (see printf()).
...	the parameters to be formatted according to the format string (see printf()).

4.28.3.2 caerLogDisable()

```
void caerLogDisable (
            bool disableLogging )
```

Disable all logging for this thread only. Call again with different argument to re-enable.

Parameters

disableLogging	true to disable logging for this thread, false to enable it again.

4.28.3.3 caerLogDisabled()

```
bool caerLogDisabled (
             void  )
```

Status of logging for this thread.

Returns

true if logging is disabled for this thread, false if it is enabled.

4.28.3.4 caerLogFileDescriptorsGetFirst()

```
int caerLogFileDescriptorsGetFirst (
             void  )
```

Get the current output file descriptor 1.

Returns

the current output file descriptor 1.

4.28.3.5 caerLogFileDescriptorsGetSecond()

```
int caerLogFileDescriptorsGetSecond (
             void  )
```

Get the current output file descriptor 2.

Returns

the current output file descriptor 2.

4.28.3.6 caerLogFileDescriptorsSet()

```
void caerLogFileDescriptorsSet (
             int fd1,
             int fd2 )
```

Set to which file descriptors log messages are sent. Up to two different file descriptors can be configured here. By default logging to STDERR only is enabled. If both file descriptors are identical, logging to it will only happen once, as if the second one was disabled.

Parameters

fd1	first file descriptor to log to. A negative value will disable it.
fd2	second file descriptor to log to. A negative value will disable it.

4.28.3.7 caerLogLevelGet()

```
enum caer_log_level caerLogLevelGet (
            void  )
```

Get the current system-wide log level. Log messages are only printed if their level is equal or above this level.

Returns

the current system-wide log level.

4.28.3.8 caerLogLevelSet()

```
void caerLogLevelSet (
            enum caer_log_level logLevel )
```

Set the system-wide log level. Log messages will only be printed if their level is equal or above this level.

Parameters

logLevel	the system-wide log level.

4.28.3.9 caerLogVA()

```
void caerLogVA (
            enum caer_log_level logLevel,
            const char * subSystem,
            const char * format,
            va_list args )
```

Secondary logging function. This function takes messages, formats them and sends them out to a file descriptor, respecting the system-wide log level setting and prepending the current time, the log level and a user-specified common string to the actual formatted output. The format is specified exactly as with the printf() family of functions. The argument list is a va_list as returned by va_start(), following the vprintf() family of functions in its functionality. Please see their manual-page for more information.

Parameters

logLevel	the message-specific log level.
subSystem	a common, user-specified string to prepend before the message.
format	the message format string (see printf()).
args	the parameters to be formatted according to the format string (see printf()). This is an argument list as returned by va_start().

4.28.3.10 caerLogVAFull()

```
void caerLogVAFull (
            int logFileDescriptor1,
            int logFileDescriptor2,
            uint8_t systemLogLevel,
            enum caer_log_level logLevel,
            const char * subSystem,
            const char * format,
            va_list args )
```

Tertiary logging function. This function takes messages, formats them and sends them out to up to two file descriptors, fully specified by the user; allows a user-given system log level setting to also be specified, and then prepends the current time, the message log level and a user-specified common string to the actual formatted output. The format is specified exactly as with the printf() family of functions. The argument list is a va_list as returned by va_start(), following the vprintf() family of functions in its functionality. Please see their manual-page for more information.

Parameters

logFileDescriptor1	first output file descriptor.
logFileDescriptor2	second output file descriptor.
systemLogLevel	the system-wide log level.
logLevel	the message-specific log level.
subSystem	a common, user-specified string to prepend before the message.
format	the message format string (see printf()).
args	the parameters to be formatted according to the format string (see printf()). This is an argument list as returned by va_start().

4.29 network.h File Reference

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

Macros

- #define **AEDAT3_NETWORK_HEADER_LENGTH** 20
- #define **AEDAT3_NETWORK_MAGIC_NUMBER** 0x1D378BC90B9A6658
- #define **AEDAT3_NETWORK_VERSION** 0x01
- #define **AEDAT3_FILE_VERSION** "3.1"
- #define **AEDAT3_MAX_UDP_SIZE** (1472 - AEDAT3_NETWORK_HEADER_LENGTH)

Functions

- **PACKED_STRUCT** (struct aedat3_network_header { int64_t magicNumber;int64_t sequenceNumber;int8↩ _t versionNumber;int8_t formatNumber;int16_t sourceID;})
- static struct aedat3_network_header **caerParseNetworkHeader** (const uint8_t ∗dataBuffer)

4.29.1 Detailed Description

Useful functions for AEDAT 3.X network streams.

4.30 portable_endian.h File Reference

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

Functions

- static float **htobeflt** (float val)
- static float **htoleflt** (float val)
- static float **befltoh** (float val)
- static float **lefltoh** (float val)

4.30.1 Detailed Description

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

Index

CAER_CONFIGURATION_CONST_ITERATOR_ALL↩
_START
config.h, 189
CAER_CONFIGURATION_CONST_ITERATOR_VA↩
LID_START
config.h, 189
CAER_CONFIGURATION_CONST_REVERSE_ITE↩
RATOR_ALL_START
config.h, 189
CAER_CONFIGURATION_CONST_REVERSE_ITE↩
RATOR_VALID_START
config.h, 190
CAER_CONFIGURATION_ITERATOR_ALL_END
config.h, 190
CAER_CONFIGURATION_ITERATOR_ALL_START
config.h, 190
CAER_CONFIGURATION_ITERATOR_VALID_END
config.h, 191
CAER_CONFIGURATION_ITERATOR_VALID_START
config.h, 191
CAER_CONFIGURATION_REVERSE_ITERATOR_↩
ALL_END
config.h, 191
CAER_CONFIGURATION_REVERSE_ITERATOR_↩
ALL_START
config.h, 192
CAER_CONFIGURATION_REVERSE_ITERATOR_↩
VALID_END
config.h, 192
CAER_CONFIGURATION_REVERSE_ITERATOR_↩
VALID_START
config.h, 192
CAER_DEFAULT_EVENT_TYPES_COUNT
common.h, 171
CAER_DEVICE_DAVIS_FX2
davis.h, 23
CAER_DEVICE_DAVIS_FX3
davis.h, 23
CAER_DEVICE_DAVIS_RPI
davis.h, 23
CAER_DEVICE_DAVIS
davis.h, 23
CAER_DEVICE_DISCOVER_ALL
device_discover.h, 132
CAER_DEVICE_DVS128
dvs128.h, 134
CAER_DEVICE_DYNAPSE
dynapse.h, 142
CAER_DEVICE_EDVS
edvs.h, 163
CAER_EAR_CONST_ITERATOR_ALL_START
ear.h, 201
CAER_EAR_CONST_ITERATOR_VALID_START
ear.h, 201
CAER_EAR_CONST_REVERSE_ITERATOR_ALL_↩
START
ear.h, 202
CAER_EAR_CONST_REVERSE_ITERATOR_VALI↩
D_START
ear.h, 202
CAER_EAR_ITERATOR_ALL_END
ear.h, 202
CAER_EAR_ITERATOR_ALL_START
ear.h, 203
CAER_EAR_ITERATOR_VALID_END
ear.h, 203
CAER_EAR_ITERATOR_VALID_START
ear.h, 203
CAER_EAR_REVERSE_ITERATOR_ALL_END
ear.h, 203
CAER_EAR_REVERSE_ITERATOR_ALL_START
ear.h, 204
CAER_EAR_REVERSE_ITERATOR_VALID_END
ear.h, 204
CAER_EAR_REVERSE_ITERATOR_VALID_START
ear.h, 204
CAER_EVENT_PACKET_CONTAINER_CONST_IT↩
ERATOR_START
packetContainer.h, 305
CAER_EVENT_PACKET_CONTAINER_ITERATOR↩
_END
packetContainer.h, 305
CAER_EVENT_PACKET_CONTAINER_ITERATOR↩
_START
packetContainer.h, 306
CAER_EVENT_PACKET_HEADER_SIZE
common.h, 171
CAER_FILTER_DVS_BACKGROUND_ACTIVITY_C↩
HECK_POLARITY
dvs_noise.h, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVITY_E↩
NABLE
dvs_noise.h, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVITY_S↩
TATISTICS_OFF
dvs_noise.h, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVITY_S↩
TATISTICS_ON

dvs_noise.h, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVITY_S↔TATISTICS
dvs_noise.h, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVITY_S↔UPPORT_MAX
dvs_noise.h, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVITY_S↔UPPORT_MIN
dvs_noise.h, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVITY_TI↔ME
dvs_noise.h, 424
CAER_FILTER_DVS_BACKGROUND_ACTIVITY_T↔WO_LEVELS
dvs_noise.h, 424
CAER_FILTER_DVS_HOTPIXEL_COUNT
dvs_noise.h, 424
CAER_FILTER_DVS_HOTPIXEL_ENABLE
dvs_noise.h, 424
CAER_FILTER_DVS_HOTPIXEL_LEARN
dvs_noise.h, 424
CAER_FILTER_DVS_HOTPIXEL_STATISTICS_OFF
dvs_noise.h, 424
CAER_FILTER_DVS_HOTPIXEL_STATISTICS_ON
dvs_noise.h, 425
CAER_FILTER_DVS_HOTPIXEL_STATISTICS
dvs_noise.h, 424
CAER_FILTER_DVS_HOTPIXEL_TIME
dvs_noise.h, 425
CAER_FILTER_DVS_LOG_LEVEL
dvs_noise.h, 425
CAER_FILTER_DVS_REFRACTORY_PERIOD_EN↔ABLE
dvs_noise.h, 425
CAER_FILTER_DVS_REFRACTORY_PERIOD_STA↔TISTICS_OFF
dvs_noise.h, 425
CAER_FILTER_DVS_REFRACTORY_PERIOD_STA↔TISTICS_ON
dvs_noise.h, 425
CAER_FILTER_DVS_REFRACTORY_PERIOD_STA↔TISTICS
dvs_noise.h, 425
CAER_FILTER_DVS_REFRACTORY_PERIOD_TIME
dvs_noise.h, 425
CAER_FILTER_DVS_RESET
dvs_noise.h, 426
CAER_FRAME_CONST_ITERATOR_ALL_START
frame.h, 216
CAER_FRAME_CONST_ITERATOR_VALID_START
frame.h, 216
CAER_FRAME_CONST_REVERSE_ITERATOR_AL↔L_START
frame.h, 217
CAER_FRAME_CONST_REVERSE_ITERATOR_VA↔LID_START
frame.h, 217
CAER_FRAME_ITERATOR_ALL_END
frame.h, 217
CAER_FRAME_ITERATOR_ALL_START
frame.h, 218
CAER_FRAME_ITERATOR_VALID_END
frame.h, 218
CAER_FRAME_ITERATOR_VALID_START
frame.h, 218
CAER_FRAME_REVERSE_ITERATOR_ALL_END
frame.h, 218
CAER_FRAME_REVERSE_ITERATOR_ALL_START
frame.h, 219
CAER_FRAME_REVERSE_ITERATOR_VALID_END
frame.h, 219
CAER_FRAME_REVERSE_ITERATOR_VALID_STA↔RT
frame.h, 219
CAER_HOST_CONFIG_DATAEXCHANGE_BLOCKI↔NG
device.h, 124
CAER_HOST_CONFIG_DATAEXCHANGE_BUFFE↔R_SIZE
device.h, 125
CAER_HOST_CONFIG_DATAEXCHANGE_START↔_PRODUCERS
device.h, 125
CAER_HOST_CONFIG_DATAEXCHANGE_STOP_↔PRODUCERS
device.h, 125
CAER_HOST_CONFIG_DATAEXCHANGE
device.h, 124
CAER_HOST_CONFIG_LOG_LEVEL
device.h, 125
CAER_HOST_CONFIG_LOG
device.h, 125
CAER_HOST_CONFIG_PACKETS_MAX_CONTAIN↔ER_INTERVAL
device.h, 126
CAER_HOST_CONFIG_PACKETS_MAX_CONTAIN↔ER_PACKET_SIZE
device.h, 126
CAER_HOST_CONFIG_PACKETS
device.h, 125
CAER_HOST_CONFIG_SERIAL_BAUD_RATE_12M
serial.h, 166
CAER_HOST_CONFIG_SERIAL_BAUD_RATE_2M
serial.h, 166
CAER_HOST_CONFIG_SERIAL_BAUD_RATE_4M
serial.h, 167
CAER_HOST_CONFIG_SERIAL_BAUD_RATE_8M
serial.h, 167
CAER_HOST_CONFIG_SERIAL_READ_SIZE
serial.h, 167
CAER_HOST_CONFIG_SERIAL
serial.h, 166
CAER_HOST_CONFIG_USB_BUFFER_NUMBER
usb.h, 168
CAER_HOST_CONFIG_USB_BUFFER_SIZE

usb.h, 168
CAER_HOST_CONFIG_USB
usb.h, 168
CAER_IMU6_CONST_ITERATOR_ALL_START
imu6.h, 244
CAER_IMU6_CONST_ITERATOR_VALID_START
imu6.h, 245
CAER_IMU6_CONST_REVERSE_ITERATOR_ALL_↵
START
imu6.h, 245
CAER_IMU6_CONST_REVERSE_ITERATOR_VALI↵
D_START
imu6.h, 245
CAER_IMU6_ITERATOR_ALL_END
imu6.h, 246
CAER_IMU6_ITERATOR_ALL_START
imu6.h, 246
CAER_IMU6_ITERATOR_VALID_END
imu6.h, 246
CAER_IMU6_ITERATOR_VALID_START
imu6.h, 246
CAER_IMU6_REVERSE_ITERATOR_ALL_END
imu6.h, 247
CAER_IMU6_REVERSE_ITERATOR_ALL_START
imu6.h, 247
CAER_IMU6_REVERSE_ITERATOR_VALID_END
imu6.h, 247
CAER_IMU6_REVERSE_ITERATOR_VALID_START
imu6.h, 247
CAER_IMU9_CONST_ITERATOR_ALL_START
imu9.h, 261
CAER_IMU9_CONST_ITERATOR_VALID_START
imu9.h, 261
CAER_IMU9_CONST_REVERSE_ITERATOR_ALL_↵
START
imu9.h, 262
CAER_IMU9_CONST_REVERSE_ITERATOR_VALI↵
D_START
imu9.h, 262
CAER_IMU9_ITERATOR_ALL_END
imu9.h, 262
CAER_IMU9_ITERATOR_ALL_START
imu9.h, 263
CAER_IMU9_ITERATOR_VALID_END
imu9.h, 263
CAER_IMU9_ITERATOR_VALID_START
imu9.h, 263
CAER_IMU9_REVERSE_ITERATOR_ALL_END
imu9.h, 263
CAER_IMU9_REVERSE_ITERATOR_ALL_START
imu9.h, 264
CAER_IMU9_REVERSE_ITERATOR_VALID_END
imu9.h, 264
CAER_IMU9_REVERSE_ITERATOR_VALID_START
imu9.h, 264
CAER_ITERATOR_ALL_END
common.h, 172
CAER_ITERATOR_ALL_START
common.h, 172
CAER_ITERATOR_VALID_END
common.h, 172
CAER_ITERATOR_VALID_START
common.h, 172
CAER_MATRIX4x4_CONST_ITERATOR_ALL_START
matrix4x4.h, 280
CAER_MATRIX4x4_CONST_ITERATOR_VALID_ST↵
ART
matrix4x4.h, 280
CAER_MATRIX4x4_CONST_REVERSE_ITERATOR↵
_ALL_START
matrix4x4.h, 281
CAER_MATRIX4x4_CONST_REVERSE_ITERATOR↵
_VALID_START
matrix4x4.h, 281
CAER_MATRIX4x4_ITERATOR_ALL_END
matrix4x4.h, 282
CAER_MATRIX4x4_ITERATOR_ALL_START
matrix4x4.h, 282
CAER_MATRIX4x4_ITERATOR_VALID_END
matrix4x4.h, 282
CAER_MATRIX4x4_ITERATOR_VALID_START
matrix4x4.h, 282
CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_END
matrix4x4.h, 283
CAER_MATRIX4x4_REVERSE_ITERATOR_ALL_S↵
TART
matrix4x4.h, 283
CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_↵
END
matrix4x4.h, 283
CAER_MATRIX4x4_REVERSE_ITERATOR_VALID_↵
START
matrix4x4.h, 284
CAER_POINT1D_CONST_ITERATOR_ALL_START
point1d.h, 314
CAER_POINT1D_CONST_ITERATOR_VALID_START
point1d.h, 314
CAER_POINT1D_CONST_REVERSE_ITERATOR_↵
ALL_START
point1d.h, 314
CAER_POINT1D_CONST_REVERSE_ITERATOR_↵
VALID_START
point1d.h, 315
CAER_POINT1D_ITERATOR_ALL_END
point1d.h, 315
CAER_POINT1D_ITERATOR_ALL_START
point1d.h, 315
CAER_POINT1D_ITERATOR_VALID_END
point1d.h, 316
CAER_POINT1D_ITERATOR_VALID_START
point1d.h, 316
CAER_POINT1D_REVERSE_ITERATOR_ALL_END
point1d.h, 316
CAER_POINT1D_REVERSE_ITERATOR_ALL_STA↵
RT
point1d.h, 316

CAER_POINT1D_REVERSE_ITERATOR_VALID_END
point1d.h, 317
CAER_POINT1D_REVERSE_ITERATOR_VALID_S↩
TART
point1d.h, 317
CAER_POINT2D_CONST_ITERATOR_ALL_START
point2d.h, 326
CAER_POINT2D_CONST_ITERATOR_VALID_START
point2d.h, 327
CAER_POINT2D_CONST_REVERSE_ITERATOR_↩
ALL_START
point2d.h, 327
CAER_POINT2D_CONST_REVERSE_ITERATOR_↩
VALID_START
point2d.h, 327
CAER_POINT2D_ITERATOR_ALL_END
point2d.h, 328
CAER_POINT2D_ITERATOR_ALL_START
point2d.h, 328
CAER_POINT2D_ITERATOR_VALID_END
point2d.h, 328
CAER_POINT2D_ITERATOR_VALID_START
point2d.h, 328
CAER_POINT2D_REVERSE_ITERATOR_ALL_END
point2d.h, 329
CAER_POINT2D_REVERSE_ITERATOR_ALL_STA↩
RT
point2d.h, 329
CAER_POINT2D_REVERSE_ITERATOR_VALID_END
point2d.h, 329
CAER_POINT2D_REVERSE_ITERATOR_VALID_S↩
TART
point2d.h, 329
CAER_POINT3D_CONST_ITERATOR_ALL_START
point3d.h, 340
CAER_POINT3D_CONST_ITERATOR_VALID_START
point3d.h, 340
CAER_POINT3D_CONST_REVERSE_ITERATOR_↩
ALL_START
point3d.h, 341
CAER_POINT3D_CONST_REVERSE_ITERATOR_↩
VALID_START
point3d.h, 341
CAER_POINT3D_ITERATOR_ALL_END
point3d.h, 341
CAER_POINT3D_ITERATOR_ALL_START
point3d.h, 342
CAER_POINT3D_ITERATOR_VALID_END
point3d.h, 342
CAER_POINT3D_ITERATOR_VALID_START
point3d.h, 342
CAER_POINT3D_REVERSE_ITERATOR_ALL_END
point3d.h, 342
CAER_POINT3D_REVERSE_ITERATOR_ALL_STA↩
RT
point3d.h, 343
CAER_POINT3D_REVERSE_ITERATOR_VALID_END
point3d.h, 343
CAER_POINT3D_REVERSE_ITERATOR_VALID_S↩
TART
point3d.h, 343
CAER_POINT4D_CONST_ITERATOR_ALL_START
point4d.h, 354
CAER_POINT4D_CONST_ITERATOR_VALID_START
point4d.h, 354
CAER_POINT4D_CONST_REVERSE_ITERATOR_↩
ALL_START
point4d.h, 355
CAER_POINT4D_CONST_REVERSE_ITERATOR_↩
VALID_START
point4d.h, 355
CAER_POINT4D_ITERATOR_ALL_END
point4d.h, 355
CAER_POINT4D_ITERATOR_ALL_START
point4d.h, 356
CAER_POINT4D_ITERATOR_VALID_END
point4d.h, 356
CAER_POINT4D_ITERATOR_VALID_START
point4d.h, 356
CAER_POINT4D_REVERSE_ITERATOR_ALL_END
point4d.h, 356
CAER_POINT4D_REVERSE_ITERATOR_ALL_STA↩
RT
point4d.h, 357
CAER_POINT4D_REVERSE_ITERATOR_VALID_END
point4d.h, 357
CAER_POINT4D_REVERSE_ITERATOR_VALID_S↩
TART
point4d.h, 357
CAER_POLARITY_CONST_ITERATOR_ALL_START
polarity.h, 369
CAER_POLARITY_CONST_ITERATOR_VALID_ST↩
ART
polarity.h, 370
CAER_POLARITY_CONST_REVERSE_ITERATOR↩
_ALL_START
polarity.h, 370
CAER_POLARITY_CONST_REVERSE_ITERATOR↩
_VALID_START
polarity.h, 370
CAER_POLARITY_ITERATOR_ALL_END
polarity.h, 371
CAER_POLARITY_ITERATOR_ALL_START
polarity.h, 371
CAER_POLARITY_ITERATOR_VALID_END
polarity.h, 371
CAER_POLARITY_ITERATOR_VALID_START
polarity.h, 371
CAER_POLARITY_REVERSE_ITERATOR_ALL_END
polarity.h, 372
CAER_POLARITY_REVERSE_ITERATOR_ALL_ST↩
ART
polarity.h, 372
CAER_POLARITY_REVERSE_ITERATOR_VALID_↩
END
polarity.h, 372

CAER_POLARITY_REVERSE_ITERATOR_VALID_↩START
  polarity.h, 372
CAER_SAMPLE_CONST_ITERATOR_ALL_START
  sample.h, 382
CAER_SAMPLE_CONST_ITERATOR_VALID_START
  sample.h, 382
CAER_SAMPLE_CONST_REVERSE_ITERATOR_A↩LL_START
  sample.h, 383
CAER_SAMPLE_CONST_REVERSE_ITERATOR_V↩ALID_START
  sample.h, 383
CAER_SAMPLE_ITERATOR_ALL_END
  sample.h, 383
CAER_SAMPLE_ITERATOR_ALL_START
  sample.h, 384
CAER_SAMPLE_ITERATOR_VALID_END
  sample.h, 384
CAER_SAMPLE_ITERATOR_VALID_START
  sample.h, 384
CAER_SAMPLE_REVERSE_ITERATOR_ALL_END
  sample.h, 384
CAER_SAMPLE_REVERSE_ITERATOR_ALL_START
  sample.h, 385
CAER_SAMPLE_REVERSE_ITERATOR_VALID_END
  sample.h, 385
CAER_SAMPLE_REVERSE_ITERATOR_VALID_ST↩ART
  sample.h, 385
CAER_SPECIAL_CONST_ITERATOR_ALL_START
  special.h, 394
CAER_SPECIAL_CONST_ITERATOR_VALID_START
  special.h, 395
CAER_SPECIAL_CONST_REVERSE_ITERATOR_↩ALL_START
  special.h, 395
CAER_SPECIAL_CONST_REVERSE_ITERATOR_↩VALID_START
  special.h, 395
CAER_SPECIAL_ITERATOR_ALL_END
  special.h, 396
CAER_SPECIAL_ITERATOR_ALL_START
  special.h, 396
CAER_SPECIAL_ITERATOR_VALID_END
  special.h, 396
CAER_SPECIAL_ITERATOR_VALID_START
  special.h, 396
CAER_SPECIAL_REVERSE_ITERATOR_ALL_END
  special.h, 397
CAER_SPECIAL_REVERSE_ITERATOR_ALL_START
  special.h, 397
CAER_SPECIAL_REVERSE_ITERATOR_VALID_END
  special.h, 397
CAER_SPECIAL_REVERSE_ITERATOR_VALID_S↩TART
  special.h, 397
CAER_SPIKE_CONST_ITERATOR_ALL_START
  spike.h, 409
CAER_SPIKE_CONST_ITERATOR_VALID_START
  spike.h, 409
CAER_SPIKE_CONST_REVERSE_ITERATOR_ALL↩_START
  spike.h, 410
CAER_SPIKE_CONST_REVERSE_ITERATOR_VAL↩ID_START
  spike.h, 410
CAER_SPIKE_ITERATOR_ALL_END
  spike.h, 410
CAER_SPIKE_ITERATOR_ALL_START
  spike.h, 411
CAER_SPIKE_ITERATOR_VALID_END
  spike.h, 411
CAER_SPIKE_ITERATOR_VALID_START
  spike.h, 411
CAER_SPIKE_REVERSE_ITERATOR_ALL_END
  spike.h, 411
CAER_SPIKE_REVERSE_ITERATOR_ALL_START
  spike.h, 412
CAER_SPIKE_REVERSE_ITERATOR_VALID_END
  spike.h, 412
CAER_SPIKE_REVERSE_ITERATOR_VALID_START
  spike.h, 412
CAER_SUPPORTED_DEVICES_NUMBER
  device.h, 126
CLEAR_NUMBITS16
  libcaer.h, 431
CLEAR_NUMBITS32
  libcaer.h, 431
CLEAR_NUMBITS8
  libcaer.h, 431
CONFIG_MODULE_ADDR_MASK
  config.h, 192
CONFIG_MODULE_ADDR_SHIFT
  config.h, 193
caer_bias_coarsefine, 5
caer_bias_dynapse, 5
caer_bias_shiftedsource, 6
caer_bias_shiftedsource_operating_mode
  davis.h, 118
caer_bias_shiftedsource_voltage_level
  davis.h, 119
caer_bias_vdac, 7
caer_davis_aps_frame_modes
  davis.h, 119
caer_davis_imu_invensense_accel_scale
  davis.h, 119
caer_davis_imu_invensense_gyro_scale
  davis.h, 119
caer_davis_imu_types
  davis.h, 119
caer_davis_info, 7
  deviceString, 8
caer_default_event_types
  common.h, 173
caer_device_discovery_result, 9

caer_dvs128_info, 9
    deviceString, 10
caer_dynapse_info, 10
    deviceString, 11
caer_edvs_info, 11
    deviceString, 12
caer_error_codes
    libcaer.h, 435
caer_filter_dvs_pixel, 12
caer_frame_event_color_channels
    frame.h, 221
caer_frame_event_color_filter
    frame.h, 221
caer_log_level
    log.h, 438
caer_special_event_types
    special.h, 399
caerBiasCoarseFineFromCurrent
    davis.h, 120
caerBiasCoarseFineGenerate
    davis.h, 120
caerBiasCoarseFineParse
    davis.h, 120
caerBiasCoarseFineToCurrent
    davis.h, 121
caerBiasDynapseGenerate
    dynapse.h, 154
caerBiasDynapseParse
    dynapse.h, 154
caerBiasShiftedSourceGenerate
    davis.h, 121
caerBiasShiftedSourceParse
    davis.h, 121
caerBiasVDACGenerate
    davis.h, 122
caerBiasVDACParse
    davis.h, 122
caerByteArrayToInteger
    libcaer.h, 435
caerConfigurationEvent
    config.h, 193
caerConfigurationEventGetModuleAddress
    config.h, 193
caerConfigurationEventGetParameter
    config.h, 194
caerConfigurationEventGetParameterAddress
    config.h, 194
caerConfigurationEventGetTimestamp
    config.h, 194
caerConfigurationEventGetTimestamp64
    config.h, 195
caerConfigurationEventInvalidate
    config.h, 195
caerConfigurationEventIsValid
    config.h, 195
caerConfigurationEventPacket
    config.h, 193
caerConfigurationEventPacketAllocate
    config.h, 196
caerConfigurationEventPacketFromPacketHeader
    config.h, 196
caerConfigurationEventPacketFromPacketHeaderConst
    config.h, 197
caerConfigurationEventPacketGetEvent
    config.h, 197
caerConfigurationEventPacketGetEventConst
    config.h, 197
caerConfigurationEventSetModuleAddress
    config.h, 198
caerConfigurationEventSetParameter
    config.h, 198
caerConfigurationEventSetParameterAddress
    config.h, 198
caerConfigurationEventSetTimestamp
    config.h, 199
caerConfigurationEventValidate
    config.h, 199
caerDVS128InfoGet
    dvs128.h, 137
caerDavisInfoGet
    davis.h, 122
caerDavisROIConfigure
    davis.h, 123
caerDeviceClose
    device.h, 126
caerDeviceConfigGet
    device.h, 127
caerDeviceConfigGet64
    device.h, 127
caerDeviceConfigSet
    device.h, 128
caerDeviceDataGet
    device.h, 128
caerDeviceDataStart
    device.h, 130
caerDeviceDataStop
    device.h, 130
caerDeviceDiscover
    device_discover.h, 132
caerDeviceDiscoverOpen
    device_discover.h, 133
caerDeviceDiscoveryResult
    device_discover.h, 132
caerDeviceHandle
    device.h, 126
caerDeviceOpen
    usb.h, 169
caerDeviceOpenSerial
    serial.h, 167
caerDeviceSendDefaultConfig
    device.h, 131
caerDynapseCoreAddrToNeuronId
    dynapse.h, 155
caerDynapseCoreXYToNeuronId
    dynapse.h, 155
caerDynapseGenerateCamBits

dynapse.h, 155
caerDynapseGenerateSramBits
dynapse.h, 156
caerDynapseInfoGet
dynapse.h, 157
caerDynapseSendDataToUSB
dynapse.h, 157
caerDynapseSpikeEventFromXY
dynapse.h, 157
caerDynapseSpikeEventGetX
dynapse.h, 158
caerDynapseSpikeEventGetY
dynapse.h, 158
caerDynapseWriteCam
dynapse.h, 159
caerDynapseWritePoissonSpikeRate
dynapse.h, 159
caerDynapseWriteSram
dynapse.h, 160
caerDynapseWriteSramWords
dynapse.h, 161
caerDynapseWriteSramN
dynapse.h, 160
caerEDVSInfoGet
edvs.h, 165
caerEarEvent
ear.h, 206
caerEarEventGetChannel
ear.h, 206
caerEarEventGetEar
ear.h, 207
caerEarEventGetTimestamp
ear.h, 207
caerEarEventGetTimestamp64
ear.h, 207
caerEarEventInvalidate
ear.h, 208
caerEarEventIsValid
ear.h, 208
caerEarEventPacket
ear.h, 206
caerEarEventPacketAllocate
ear.h, 209
caerEarEventPacketFromPacketHeader
ear.h, 209
caerEarEventPacketFromPacketHeaderConst
ear.h, 209
caerEarEventPacketGetEvent
ear.h, 211
caerEarEventPacketGetEventConst
ear.h, 211
caerEarEventSetChannel
ear.h, 211
caerEarEventSetEar
ear.h, 212
caerEarEventSetTimestamp
ear.h, 212
caerEarEventValidate
ear.h, 212
caerEventPacketAllocate
common.h, 174
caerEventPacketAppend
common.h, 174
caerEventPacketClean
common.h, 175
caerEventPacketClear
common.h, 175
caerEventPacketContainer
packetContainer.h, 306
caerEventPacketContainerAllocate
packetContainer.h, 306
caerEventPacketContainerCopyAllEvents
packetContainer.h, 307
caerEventPacketContainerCopyValidEvents
packetContainer.h, 307
caerEventPacketContainerFindEventPacketByType
packetContainer.h, 307
caerEventPacketContainerFindEventPacketByType←
Const
packetContainer.h, 308
caerEventPacketContainerFree
packetContainer.h, 308
caerEventPacketContainerGetEventPacket
packetContainer.h, 309
caerEventPacketContainerGetEventPacketConst
packetContainer.h, 309
caerEventPacketContainerGetEventPacketsNumber
packetContainer.h, 309
caerEventPacketContainerGetEventsNumber
packetContainer.h, 310
caerEventPacketContainerGetEventsValidNumber
packetContainer.h, 310
caerEventPacketContainerGetHighestEventTimestamp
packetContainer.h, 310
caerEventPacketContainerGetLowestEventTimestamp
packetContainer.h, 311
caerEventPacketContainerSetEventPacket
packetContainer.h, 311
caerEventPacketContainerSetEventPacketsNumber
packetContainer.h, 312
caerEventPacketContainerUpdateStatistics
packetContainer.h, 312
caerEventPacketCopy
common.h, 175
caerEventPacketCopyOnlyEvents
common.h, 176
caerEventPacketCopyOnlyValidEvents
common.h, 176
caerEventPacketEquals
common.h, 176
caerEventPacketGetDataSize
common.h, 177
caerEventPacketGetDataSizeEvents
common.h, 177
caerEventPacketGetSize
common.h, 178

caerEventPacketGetSizeEvents
common.h, 178
caerEventPacketGrow
common.h, 178
caerEventPacketHeader
common.h, 173
caerEventPacketHeaderGetEventCapacity
common.h, 179
caerEventPacketHeaderGetEventNumber
common.h, 179
caerEventPacketHeaderGetEventSize
common.h, 179
caerEventPacketHeaderGetEventSource
common.h, 180
caerEventPacketHeaderGetEventTSOffset
common.h, 180
caerEventPacketHeaderGetEventTSOverflow
common.h, 181
caerEventPacketHeaderGetEventType
common.h, 181
caerEventPacketHeaderGetEventValid
common.h, 181
caerEventPacketHeaderSetEventCapacity
common.h, 182
caerEventPacketHeaderSetEventNumber
common.h, 182
caerEventPacketHeaderSetEventSize
common.h, 182
caerEventPacketHeaderSetEventSource
common.h, 183
caerEventPacketHeaderSetEventTSOffset
common.h, 183
caerEventPacketHeaderSetEventTSOverflow
common.h, 183
caerEventPacketHeaderSetEventType
common.h, 184
caerEventPacketHeaderSetEventValid
common.h, 184
caerEventPacketResize
common.h, 184
caerFilterDVSNoise
dvs_noise.h, 426
caerFilterDVSNoiseApply
dvs_noise.h, 426
caerFilterDVSNoiseConfigGet
dvs_noise.h, 427
caerFilterDVSNoiseConfigSet
dvs_noise.h, 427
caerFilterDVSNoiseDestroy
dvs_noise.h, 427
caerFilterDVSNoiseGetHotPixels
dvs_noise.h, 428
caerFilterDVSNoiseInitialize
dvs_noise.h, 428
caerFilterDVSNoiseStatsApply
dvs_noise.h, 428
caerFilterDVSPixel
dvs_noise.h, 426
caerFrameEvent
frame.h, 221
caerFrameEventGetChannelNumber
frame.h, 222
caerFrameEventGetColorFilter
frame.h, 222
caerFrameEventGetExposureLength
frame.h, 223
caerFrameEventGetLengthX
frame.h, 223
caerFrameEventGetLengthY
frame.h, 223
caerFrameEventGetPixel
frame.h, 225
caerFrameEventGetPixelArrayUnsafe
frame.h, 225
caerFrameEventGetPixelArrayUnsafeConst
frame.h, 226
caerFrameEventGetPixelForChannel
frame.h, 226
caerFrameEventGetPixelForChannelUnsafe
frame.h, 226
caerFrameEventGetPixelUnsafe
frame.h, 228
caerFrameEventGetPixelsMaxIndex
frame.h, 227
caerFrameEventGetPixelsSize
frame.h, 227
caerFrameEventGetPositionX
frame.h, 228
caerFrameEventGetPositionY
frame.h, 228
caerFrameEventGetROIIdentifier
frame.h, 229
caerFrameEventGetTSEndOfExposure
frame.h, 230
caerFrameEventGetTSEndOfExposure64
frame.h, 230
caerFrameEventGetTSEndOfFrame
frame.h, 231
caerFrameEventGetTSEndOfFrame64
frame.h, 231
caerFrameEventGetTSStartOfExposure
frame.h, 232
caerFrameEventGetTSStartOfExposure64
frame.h, 232
caerFrameEventGetTSStartOfFrame
frame.h, 232
caerFrameEventGetTSStartOfFrame64
frame.h, 233
caerFrameEventGetTimestamp
frame.h, 229
caerFrameEventGetTimestamp64
frame.h, 230
caerFrameEventInvalidate
frame.h, 233
caerFrameEventIsValid
frame.h, 234

caerFrameEventPacket
frame.h, 221
caerFrameEventPacketAllocate
frame.h, 234
caerFrameEventPacketAllocateNumPixels
frame.h, 234
caerFrameEventPacketFromPacketHeader
frame.h, 235
caerFrameEventPacketFromPacketHeaderConst
frame.h, 235
caerFrameEventPacketGetEvent
frame.h, 236
caerFrameEventPacketGetEventConst
frame.h, 236
caerFrameEventPacketGetPixelsMaxIndex
frame.h, 237
caerFrameEventPacketGetPixelsSize
frame.h, 237
caerFrameEventSetColorFilter
frame.h, 237
caerFrameEventSetLengthXLengthYChannelNumber
frame.h, 238
caerFrameEventSetPixel
frame.h, 238
caerFrameEventSetPixelForChannel
frame.h, 239
caerFrameEventSetPixelForChannelUnsafe
frame.h, 239
caerFrameEventSetPixelUnsafe
frame.h, 240
caerFrameEventSetPositionX
frame.h, 240
caerFrameEventSetPositionY
frame.h, 240
caerFrameEventSetROIIdentifier
frame.h, 241
caerFrameEventSetTSEndOfExposure
frame.h, 241
caerFrameEventSetTSEndOfFrame
frame.h, 241
caerFrameEventSetTSStartOfExposure
frame.h, 242
caerFrameEventSetTSStartOfFrame
frame.h, 242
caerFrameEventValidate
frame.h, 242
caerGenericEventCopy
common.h, 185
caerGenericEventGetEvent
common.h, 185
caerGenericEventGetTimestamp
common.h, 186
caerGenericEventGetTimestamp64
common.h, 186
caerGenericEventIsValid
common.h, 187
caerIMU6Event
imu6.h, 248
caerIMU6EventGetAccelX
imu6.h, 248
caerIMU6EventGetAccelY
imu6.h, 249
caerIMU6EventGetAccelZ
imu6.h, 249
caerIMU6EventGetGyroX
imu6.h, 249
caerIMU6EventGetGyroY
imu6.h, 251
caerIMU6EventGetGyroZ
imu6.h, 251
caerIMU6EventGetTemp
imu6.h, 251
caerIMU6EventGetTimestamp
imu6.h, 253
caerIMU6EventGetTimestamp64
imu6.h, 253
caerIMU6EventInvalidate
imu6.h, 253
caerIMU6EventIsValid
imu6.h, 254
caerIMU6EventPacket
imu6.h, 248
caerIMU6EventPacketAllocate
imu6.h, 254
caerIMU6EventPacketFromPacketHeader
imu6.h, 255
caerIMU6EventPacketFromPacketHeaderConst
imu6.h, 255
caerIMU6EventPacketGetEvent
imu6.h, 255
caerIMU6EventPacketGetEventConst
imu6.h, 256
caerIMU6EventSetAccelX
imu6.h, 256
caerIMU6EventSetAccelY
imu6.h, 256
caerIMU6EventSetAccelZ
imu6.h, 257
caerIMU6EventSetGyroX
imu6.h, 257
caerIMU6EventSetGyroY
imu6.h, 257
caerIMU6EventSetGyroZ
imu6.h, 258
caerIMU6EventSetTemp
imu6.h, 258
caerIMU6EventSetTimestamp
imu6.h, 258
caerIMU6EventValidate
imu6.h, 259
caerIMU9Event
imu9.h, 265
caerIMU9EventGetAccelX
imu9.h, 265
caerIMU9EventGetAccelY
imu9.h, 265

caerIMU9EventGetAccelZ
imu9.h, 266
caerIMU9EventGetCompX
imu9.h, 266
caerIMU9EventGetCompY
imu9.h, 266
caerIMU9EventGetCompZ
imu9.h, 268
caerIMU9EventGetGyroX
imu9.h, 268
caerIMU9EventGetGyroY
imu9.h, 268
caerIMU9EventGetGyroZ
imu9.h, 270
caerIMU9EventGetTemp
imu9.h, 270
caerIMU9EventGetTimestamp
imu9.h, 270
caerIMU9EventGetTimestamp64
imu9.h, 271
caerIMU9EventInvalidate
imu9.h, 271
caerIMU9EventIsValid
imu9.h, 272
caerIMU9EventPacket
imu9.h, 265
caerIMU9EventPacketAllocate
imu9.h, 272
caerIMU9EventPacketFromPacketHeader
imu9.h, 272
caerIMU9EventPacketFromPacketHeaderConst
imu9.h, 273
caerIMU9EventPacketGetEvent
imu9.h, 273
caerIMU9EventPacketGetEventConst
imu9.h, 273
caerIMU9EventSetAccelX
imu9.h, 274
caerIMU9EventSetAccelY
imu9.h, 274
caerIMU9EventSetAccelZ
imu9.h, 274
caerIMU9EventSetCompX
imu9.h, 275
caerIMU9EventSetCompY
imu9.h, 275
caerIMU9EventSetCompZ
imu9.h, 275
caerIMU9EventSetGyroX
imu9.h, 276
caerIMU9EventSetGyroY
imu9.h, 276
caerIMU9EventSetGyroZ
imu9.h, 276
caerIMU9EventSetTemp
imu9.h, 277
caerIMU9EventSetTimestamp
imu9.h, 277
caerIMU9EventValidate
imu9.h, 277
caerIntegerToByteArray
libcaer.h, 436
caerLog
log.h, 438
caerLogDisable
log.h, 438
caerLogDisabled
log.h, 439
caerLogFileDescriptorsGetFirst
log.h, 439
caerLogFileDescriptorsGetSecond
log.h, 439
caerLogFileDescriptorsSet
log.h, 439
caerLogLevelGet
log.h, 440
caerLogLevelSet
log.h, 440
caerLogVAFull
log.h, 441
caerLogVA
log.h, 440
caerMatrix4x4Event
matrix4x4.h, 285
caerMatrix4x4EventGetM00
matrix4x4.h, 285
caerMatrix4x4EventGetM01
matrix4x4.h, 285
caerMatrix4x4EventGetM02
matrix4x4.h, 286
caerMatrix4x4EventGetM03
matrix4x4.h, 286
caerMatrix4x4EventGetM10
matrix4x4.h, 287
caerMatrix4x4EventGetM11
matrix4x4.h, 287
caerMatrix4x4EventGetM12
matrix4x4.h, 287
caerMatrix4x4EventGetM13
matrix4x4.h, 288
caerMatrix4x4EventGetM20
matrix4x4.h, 288
caerMatrix4x4EventGetM21
matrix4x4.h, 288
caerMatrix4x4EventGetM22
matrix4x4.h, 289
caerMatrix4x4EventGetM23
matrix4x4.h, 289
caerMatrix4x4EventGetM30
matrix4x4.h, 289
caerMatrix4x4EventGetM31
matrix4x4.h, 291
caerMatrix4x4EventGetM32
matrix4x4.h, 291
caerMatrix4x4EventGetM33
matrix4x4.h, 291

caerMatrix4x4EventGetScale
matrix4x4.h, 293
caerMatrix4x4EventGetTimestamp
matrix4x4.h, 293
caerMatrix4x4EventGetTimestamp64
matrix4x4.h, 293
caerMatrix4x4EventGetType
matrix4x4.h, 294
caerMatrix4x4EventInvalidate
matrix4x4.h, 294
caerMatrix4x4EventIsValid
matrix4x4.h, 295
caerMatrix4x4EventPacket
matrix4x4.h, 285
caerMatrix4x4EventPacketAllocate
matrix4x4.h, 295
caerMatrix4x4EventPacketFromPacketHeader
matrix4x4.h, 295
caerMatrix4x4EventPacketFromPacketHeaderConst
matrix4x4.h, 296
caerMatrix4x4EventPacketGetEvent
matrix4x4.h, 296
caerMatrix4x4EventPacketGetEventConst
matrix4x4.h, 296
caerMatrix4x4EventSetM00
matrix4x4.h, 297
caerMatrix4x4EventSetM01
matrix4x4.h, 297
caerMatrix4x4EventSetM02
matrix4x4.h, 297
caerMatrix4x4EventSetM03
matrix4x4.h, 298
caerMatrix4x4EventSetM10
matrix4x4.h, 298
caerMatrix4x4EventSetM11
matrix4x4.h, 298
caerMatrix4x4EventSetM12
matrix4x4.h, 299
caerMatrix4x4EventSetM13
matrix4x4.h, 299
caerMatrix4x4EventSetM20
matrix4x4.h, 299
caerMatrix4x4EventSetM21
matrix4x4.h, 300
caerMatrix4x4EventSetM22
matrix4x4.h, 300
caerMatrix4x4EventSetM23
matrix4x4.h, 300
caerMatrix4x4EventSetM30
matrix4x4.h, 301
caerMatrix4x4EventSetM31
matrix4x4.h, 301
caerMatrix4x4EventSetM32
matrix4x4.h, 301
caerMatrix4x4EventSetM33
matrix4x4.h, 301
caerMatrix4x4EventSetScale
matrix4x4.h, 302
caerMatrix4x4EventSetTimestamp
matrix4x4.h, 302
caerMatrix4x4EventSetType
matrix4x4.h, 302
caerMatrix4x4EventValidate
matrix4x4.h, 303
caerPoint1DEvent
point1d.h, 318
caerPoint1DEventGetScale
point1d.h, 318
caerPoint1DEventGetTimestamp
point1d.h, 319
caerPoint1DEventGetTimestamp64
point1d.h, 319
caerPoint1DEventGetType
point1d.h, 319
caerPoint1DEventGetX
point1d.h, 320
caerPoint1DEventInvalidate
point1d.h, 320
caerPoint1DEventIsValid
point1d.h, 321
caerPoint1DEventPacket
point1d.h, 318
caerPoint1DEventPacketAllocate
point1d.h, 321
caerPoint1DEventPacketFromPacketHeader
point1d.h, 321
caerPoint1DEventPacketFromPacketHeaderConst
point1d.h, 322
caerPoint1DEventPacketGetEvent
point1d.h, 322
caerPoint1DEventPacketGetEventConst
point1d.h, 322
caerPoint1DEventSetScale
point1d.h, 323
caerPoint1DEventSetTimestamp
point1d.h, 323
caerPoint1DEventSetType
point1d.h, 324
caerPoint1DEventSetX
point1d.h, 324
caerPoint1DEventValidate
point1d.h, 324
caerPoint2DEvent
point2d.h, 331
caerPoint2DEventGetScale
point2d.h, 331
caerPoint2DEventGetTimestamp
point2d.h, 331
caerPoint2DEventGetTimestamp64
point2d.h, 332
caerPoint2DEventGetType
point2d.h, 332
caerPoint2DEventGetX
point2d.h, 332
caerPoint2DEventGetY
point2d.h, 333

caerPoint2DEventInvalidate
point2d.h, 333
caerPoint2DEventIsValid
point2d.h, 334
caerPoint2DEventPacket
point2d.h, 331
caerPoint2DEventPacketAllocate
point2d.h, 334
caerPoint2DEventPacketFromPacketHeader
point2d.h, 334
caerPoint2DEventPacketFromPacketHeaderConst
point2d.h, 335
caerPoint2DEventPacketGetEvent
point2d.h, 335
caerPoint2DEventPacketGetEventConst
point2d.h, 335
caerPoint2DEventSetScale
point2d.h, 336
caerPoint2DEventSetTimestamp
point2d.h, 336
caerPoint2DEventSetType
point2d.h, 337
caerPoint2DEventSetX
point2d.h, 337
caerPoint2DEventSetY
point2d.h, 337
caerPoint2DEventValidate
point2d.h, 337
caerPoint3DEvent
point3d.h, 344
caerPoint3DEventGetScale
point3d.h, 345
caerPoint3DEventGetTimestamp
point3d.h, 345
caerPoint3DEventGetTimestamp64
point3d.h, 345
caerPoint3DEventGetType
point3d.h, 346
caerPoint3DEventGetX
point3d.h, 346
caerPoint3DEventGetY
point3d.h, 346
caerPoint3DEventGetZ
point3d.h, 347
caerPoint3DEventInvalidate
point3d.h, 347
caerPoint3DEventIsValid
point3d.h, 348
caerPoint3DEventPacket
point3d.h, 344
caerPoint3DEventPacketAllocate
point3d.h, 348
caerPoint3DEventPacketFromPacketHeader
point3d.h, 348
caerPoint3DEventPacketFromPacketHeaderConst
point3d.h, 349
caerPoint3DEventPacketGetEvent
point3d.h, 349
caerPoint3DEventPacketGetEventConst
point3d.h, 349
caerPoint3DEventSetScale
point3d.h, 350
caerPoint3DEventSetTimestamp
point3d.h, 350
caerPoint3DEventSetType
point3d.h, 351
caerPoint3DEventSetX
point3d.h, 351
caerPoint3DEventSetY
point3d.h, 351
caerPoint3DEventSetZ
point3d.h, 351
caerPoint3DEventValidate
point3d.h, 352
caerPoint4DEvent
point4d.h, 358
caerPoint4DEventGetScale
point4d.h, 359
caerPoint4DEventGetTimestamp
point4d.h, 359
caerPoint4DEventGetTimestamp64
point4d.h, 359
caerPoint4DEventGetType
point4d.h, 360
caerPoint4DEventGetW
point4d.h, 360
caerPoint4DEventGetX
point4d.h, 360
caerPoint4DEventGetY
point4d.h, 361
caerPoint4DEventGetZ
point4d.h, 361
caerPoint4DEventInvalidate
point4d.h, 362
caerPoint4DEventIsValid
point4d.h, 362
caerPoint4DEventPacket
point4d.h, 358
caerPoint4DEventPacketAllocate
point4d.h, 362
caerPoint4DEventPacketFromPacketHeader
point4d.h, 363
caerPoint4DEventPacketFromPacketHeaderConst
point4d.h, 363
caerPoint4DEventPacketGetEvent
point4d.h, 363
caerPoint4DEventPacketGetEventConst
point4d.h, 364
caerPoint4DEventSetScale
point4d.h, 364
caerPoint4DEventSetTimestamp
point4d.h, 364
caerPoint4DEventSetType
point4d.h, 365
caerPoint4DEventSetW
point4d.h, 365

caerPoint4DEventSetX
point4d.h, 365
caerPoint4DEventSetY
point4d.h, 366
caerPoint4DEventSetZ
point4d.h, 366
caerPoint4DEventValidate
point4d.h, 366
caerPolarityEvent
polarity.h, 374
caerPolarityEventGetPolarity
polarity.h, 374
caerPolarityEventGetTimestamp
polarity.h, 375
caerPolarityEventGetTimestamp64
polarity.h, 375
caerPolarityEventGetX
polarity.h, 375
caerPolarityEventGetY
polarity.h, 376
caerPolarityEventInvalidate
polarity.h, 376
caerPolarityEventIsValid
polarity.h, 376
caerPolarityEventPacket
polarity.h, 374
caerPolarityEventPacketAllocate
polarity.h, 377
caerPolarityEventPacketFromPacketHeader
polarity.h, 377
caerPolarityEventPacketFromPacketHeaderConst
polarity.h, 378
caerPolarityEventPacketGetEvent
polarity.h, 378
caerPolarityEventPacketGetEventConst
polarity.h, 378
caerPolarityEventSetPolarity
polarity.h, 379
caerPolarityEventSetTimestamp
polarity.h, 379
caerPolarityEventSetX
polarity.h, 379
caerPolarityEventSetY
polarity.h, 380
caerPolarityEventValidate
polarity.h, 380
caerSampleEvent
sample.h, 386
caerSampleEventGetSample
sample.h, 387
caerSampleEventGetTimestamp
sample.h, 387
caerSampleEventGetTimestamp64
sample.h, 387
caerSampleEventGetType
sample.h, 388
caerSampleEventInvalidate
sample.h, 388
caerSampleEventIsValid
sample.h, 388
caerSampleEventPacket
sample.h, 386
caerSampleEventPacketAllocate
sample.h, 389
caerSampleEventPacketFromPacketHeader
sample.h, 389
caerSampleEventPacketFromPacketHeaderConst
sample.h, 390
caerSampleEventPacketGetEvent
sample.h, 390
caerSampleEventPacketGetEventConst
sample.h, 390
caerSampleEventSetSample
sample.h, 391
caerSampleEventSetTimestamp
sample.h, 391
caerSampleEventSetType
sample.h, 391
caerSampleEventValidate
sample.h, 392
caerSpecialEvent
special.h, 399
caerSpecialEventGetData
special.h, 400
caerSpecialEventGetTimestamp
special.h, 400
caerSpecialEventGetTimestamp64
special.h, 401
caerSpecialEventGetType
special.h, 401
caerSpecialEventInvalidate
special.h, 401
caerSpecialEventIsValid
special.h, 402
caerSpecialEventPacket
special.h, 399
caerSpecialEventPacketAllocate
special.h, 402
caerSpecialEventPacketFindEventByType
special.h, 403
caerSpecialEventPacketFindEventByTypeConst
special.h, 403
caerSpecialEventPacketFindValidEventByType
special.h, 403
caerSpecialEventPacketFindValidEventByTypeConst
special.h, 404
caerSpecialEventPacketFromPacketHeader
special.h, 404
caerSpecialEventPacketFromPacketHeaderConst
special.h, 405
caerSpecialEventPacketGetEvent
special.h, 405
caerSpecialEventPacketGetEventConst
special.h, 405
caerSpecialEventSetData
special.h, 406

caerSpecialEventSetTimestamp
    special.h, 406
caerSpecialEventSetType
    special.h, 406
caerSpecialEventValidate
    special.h, 407
caerSpikeEvent
    spike.h, 413
caerSpikeEventGetChipID
    spike.h, 414
caerSpikeEventGetNeuronID
    spike.h, 414
caerSpikeEventGetSourceCoreID
    spike.h, 415
caerSpikeEventGetTimestamp
    spike.h, 415
caerSpikeEventGetTimestamp64
    spike.h, 415
caerSpikeEventInvalidate
    spike.h, 416
caerSpikeEventIsValid
    spike.h, 416
caerSpikeEventPacket
    spike.h, 414
caerSpikeEventPacketAllocate
    spike.h, 416
caerSpikeEventPacketFromPacketHeader
    spike.h, 417
caerSpikeEventPacketFromPacketHeaderConst
    spike.h, 417
caerSpikeEventPacketGetEvent
    spike.h, 418
caerSpikeEventPacketGetEventConst
    spike.h, 418
caerSpikeEventSetChipID
    spike.h, 418
caerSpikeEventSetNeuronID
    spike.h, 420
caerSpikeEventSetSourceCoreID
    spike.h, 420
caerSpikeEventSetTimestamp
    spike.h, 420
caerSpikeEventValidate
    spike.h, 421
caerStrEquals
    libcaer.h, 436
caerStrEqualsUpTo
    libcaer.h, 436
common.h
    CAER_DEFAULT_EVENT_TYPES_COUNT, 171
    CAER_EVENT_PACKET_HEADER_SIZE, 171
    CAER_ITERATOR_ALL_END, 172
    CAER_ITERATOR_ALL_START, 172
    CAER_ITERATOR_VALID_END, 172
    CAER_ITERATOR_VALID_START, 172
    caer_default_event_types, 173
    caerEventPacketAllocate, 174
    caerEventPacketAppend, 174
    caerEventPacketClean, 175
    caerEventPacketClear, 175
    caerEventPacketCopy, 175
    caerEventPacketCopyOnlyEvents, 176
    caerEventPacketCopyOnlyValidEvents, 176
    caerEventPacketEquals, 176
    caerEventPacketGetDataSize, 177
    caerEventPacketGetDataSizeEvents, 177
    caerEventPacketGetSize, 178
    caerEventPacketGetSizeEvents, 178
    caerEventPacketGrow, 178
    caerEventPacketHeader, 173
    caerEventPacketHeaderGetEventCapacity, 179
    caerEventPacketHeaderGetEventNumber, 179
    caerEventPacketHeaderGetEventSize, 179
    caerEventPacketHeaderGetEventSource, 180
    caerEventPacketHeaderGetEventTSOffset, 180
    caerEventPacketHeaderGetEventTSOverflow, 181
    caerEventPacketHeaderGetEventType, 181
    caerEventPacketHeaderGetEventValid, 181
    caerEventPacketHeaderSetEventCapacity, 182
    caerEventPacketHeaderSetEventNumber, 182
    caerEventPacketHeaderSetEventSize, 182
    caerEventPacketHeaderSetEventSource, 183
    caerEventPacketHeaderSetEventTSOffset, 183
    caerEventPacketHeaderSetEventTSOverflow, 183
    caerEventPacketHeaderSetEventType, 184
    caerEventPacketHeaderSetEventValid, 184
    caerEventPacketResize, 184
    caerGenericEventCopy, 185
    caerGenericEventGetEvent, 185
    caerGenericEventGetTimestamp, 186
    caerGenericEventGetTimestamp64, 186
    caerGenericEventIsValid, 187
    PACKED_STRUCT, 187
    TS_OVERFLOW_SHIFT, 172
    VALID_MARK_MASK, 173
    VALID_MARK_SHIFT, 173
config.h
    CAER_CONFIGURATION_CONST_ITERATOR↩
        _ALL_START, 189
    CAER_CONFIGURATION_CONST_ITERATOR↩
        _VALID_START, 189
    CAER_CONFIGURATION_CONST_REVERSE↩
        _ITERATOR_ALL_START, 189
    CAER_CONFIGURATION_CONST_REVERSE↩
        _ITERATOR_VALID_START, 190
    CAER_CONFIGURATION_ITERATOR_ALL_E↩
        ND, 190
    CAER_CONFIGURATION_ITERATOR_ALL_ST↩
        ART, 190
    CAER_CONFIGURATION_ITERATOR_VALID_↩
        END, 191
    CAER_CONFIGURATION_ITERATOR_VALID_↩
        START, 191
    CAER_CONFIGURATION_REVERSE_ITERAT↩
        OR_ALL_END, 191

CAER_CONFIGURATION_REVERSE_ITERAT←
OR_ALL_START, 192
CAER_CONFIGURATION_REVERSE_ITERAT←
OR_VALID_END, 192
CAER_CONFIGURATION_REVERSE_ITERAT←
OR_VALID_START, 192
CONFIG_MODULE_ADDR_MASK, 192
CONFIG_MODULE_ADDR_SHIFT, 193
caerConfigurationEvent, 193
caerConfigurationEventGetModuleAddress, 193
caerConfigurationEventGetParameter, 194
caerConfigurationEventGetParameterAddress, 194
caerConfigurationEventGetTimestamp, 194
caerConfigurationEventGetTimestamp64, 195
caerConfigurationEventInvalidate, 195
caerConfigurationEventIsValid, 195
caerConfigurationEventPacket, 193
caerConfigurationEventPacketAllocate, 196
caerConfigurationEventPacketFromPacketHeader, 196
caerConfigurationEventPacketFromPacket←
HeaderConst, 197
caerConfigurationEventPacketGetEvent, 197
caerConfigurationEventPacketGetEventConst, 197
caerConfigurationEventSetModuleAddress, 198
caerConfigurationEventSetParameter, 198
caerConfigurationEventSetParameterAddress, 198
caerConfigurationEventSetTimestamp, 199
caerConfigurationEventValidate, 199
PACKED_STRUCT, 199

DAVIS128_CONFIG_BIAS_ADCCOMPBP
davis.h, 23
DAVIS128_CONFIG_BIAS_ADCREFHIGH
davis.h, 23
DAVIS128_CONFIG_BIAS_ADCREFLOW
davis.h, 24
DAVIS128_CONFIG_BIAS_AEPDBN
davis.h, 24
DAVIS128_CONFIG_BIAS_AEPUXBP
davis.h, 24
DAVIS128_CONFIG_BIAS_AEPUYBP
davis.h, 25
DAVIS128_CONFIG_BIAS_APSCAS
davis.h, 25
DAVIS128_CONFIG_BIAS_APSOVERFLOWLEVEL
davis.h, 25
DAVIS128_CONFIG_BIAS_APSROSFBN
davis.h, 26
DAVIS128_CONFIG_BIAS_BIASBUFFER
davis.h, 26
DAVIS128_CONFIG_BIAS_COLSELLOWBN
davis.h, 26
DAVIS128_CONFIG_BIAS_DACBUFBP
davis.h, 27
DAVIS128_CONFIG_BIAS_DIFFBN
davis.h, 27
DAVIS128_CONFIG_BIAS_IFREFRBN
davis.h, 27
DAVIS128_CONFIG_BIAS_IFTHRBN
davis.h, 28
DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN
davis.h, 28
DAVIS128_CONFIG_BIAS_LOCALBUFBN
davis.h, 28
DAVIS128_CONFIG_BIAS_OFFBN
davis.h, 29
DAVIS128_CONFIG_BIAS_ONBN
davis.h, 29
DAVIS128_CONFIG_BIAS_PADFOLLBN
davis.h, 29
DAVIS128_CONFIG_BIAS_PIXINVBN
davis.h, 30
DAVIS128_CONFIG_BIAS_PRBP
davis.h, 30
DAVIS128_CONFIG_BIAS_PRSFBP
davis.h, 30
DAVIS128_CONFIG_BIAS_READOUTBUFBP
davis.h, 31
DAVIS128_CONFIG_BIAS_REFRBP
davis.h, 31
DAVIS128_CONFIG_BIAS_SSN
davis.h, 31
DAVIS128_CONFIG_BIAS_SSP
davis.h, 32
DAVIS128_CONFIG_CHIP_AERNAROW
davis.h, 32
DAVIS128_CONFIG_CHIP_ANALOGMUX0
davis.h, 32
DAVIS128_CONFIG_CHIP_ANALOGMUX1
davis.h, 32
DAVIS128_CONFIG_CHIP_ANALOGMUX2
davis.h, 33
DAVIS128_CONFIG_CHIP_BIASMUX0
davis.h, 33
DAVIS128_CONFIG_CHIP_DIGITALMUX0
davis.h, 33
DAVIS128_CONFIG_CHIP_DIGITALMUX1
davis.h, 33
DAVIS128_CONFIG_CHIP_DIGITALMUX2
davis.h, 33
DAVIS128_CONFIG_CHIP_DIGITALMUX3
davis.h, 33
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, 34
DAVIS128_CONFIG_CHIP_TYPENCALIBNEURON
davis.h, 34
DAVIS128_CONFIG_CHIP_USEAOUT
davis.h, 34
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_APSOVERFLOWLEVEL
davis.h, 37
DAVIS208_CONFIG_BIAS_APSROSFBN
davis.h, 37
DAVIS208_CONFIG_BIAS_BIASBUFFER
davis.h, 37
DAVIS208_CONFIG_BIAS_COLSELLOWBN
davis.h, 38
DAVIS208_CONFIG_BIAS_DACBUFBP
davis.h, 38
DAVIS208_CONFIG_BIAS_DIFFBN
davis.h, 38
DAVIS208_CONFIG_BIAS_IFREFRBN
davis.h, 39
DAVIS208_CONFIG_BIAS_IFTHRBN
davis.h, 39
DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN
davis.h, 39
DAVIS208_CONFIG_BIAS_LOCALBUFBN
davis.h, 40
DAVIS208_CONFIG_BIAS_OFFBN
davis.h, 40
DAVIS208_CONFIG_BIAS_ONBN
davis.h, 40
DAVIS208_CONFIG_BIAS_PADFOLLBN
davis.h, 41
DAVIS208_CONFIG_BIAS_PIXINVBN
davis.h, 41
DAVIS208_CONFIG_BIAS_PRBP
davis.h, 41
DAVIS208_CONFIG_BIAS_PRSFBP
davis.h, 42
DAVIS208_CONFIG_BIAS_READOUTBUFBP
davis.h, 42
DAVIS208_CONFIG_BIAS_REFRBP
davis.h, 42
DAVIS208_CONFIG_BIAS_REFSSBN
davis.h, 43
DAVIS208_CONFIG_BIAS_REFSS
davis.h, 43
DAVIS208_CONFIG_BIAS_REGBIASBP
davis.h, 43
DAVIS208_CONFIG_BIAS_RESETHIGHPASS
davis.h, 44
DAVIS208_CONFIG_BIAS_SSN
davis.h, 44
DAVIS208_CONFIG_BIAS_SSP
davis.h, 44
DAVIS208_CONFIG_CHIP_AERNAROW
davis.h, 45
DAVIS208_CONFIG_CHIP_ANALOGMUX0
davis.h, 45
DAVIS208_CONFIG_CHIP_ANALOGMUX1
davis.h, 45
DAVIS208_CONFIG_CHIP_ANALOGMUX2
davis.h, 45
DAVIS208_CONFIG_CHIP_BIASMUX0
davis.h, 45
DAVIS208_CONFIG_CHIP_DIGITALMUX0
davis.h, 46
DAVIS208_CONFIG_CHIP_DIGITALMUX1
davis.h, 46
DAVIS208_CONFIG_CHIP_DIGITALMUX2
davis.h, 46
DAVIS208_CONFIG_CHIP_DIGITALMUX3
davis.h, 46
DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER
davis.h, 46
DAVIS208_CONFIG_CHIP_RESETCALIBNEURON
davis.h, 46
DAVIS208_CONFIG_CHIP_RESETTESTPIXEL
davis.h, 47
DAVIS208_CONFIG_CHIP_SELECTBIASREFSS
davis.h, 47
DAVIS208_CONFIG_CHIP_SELECTGRAYCOUNTER
davis.h, 47
DAVIS208_CONFIG_CHIP_SELECTHIGHPASS
davis.h, 47
DAVIS208_CONFIG_CHIP_SELECTPOSFB
davis.h, 47
DAVIS208_CONFIG_CHIP_SELECTPREAMPAVG
davis.h, 47
DAVIS208_CONFIG_CHIP_SELECTSENSE
davis.h, 48
DAVIS208_CONFIG_CHIP_TYPENCALIBNEURON
davis.h, 48
DAVIS208_CONFIG_CHIP_USEAOUT
davis.h, 48
DAVIS240_CONFIG_BIAS_AEPDBN
davis.h, 48
DAVIS240_CONFIG_BIAS_AEPUXBP
davis.h, 48
DAVIS240_CONFIG_BIAS_AEPUYBP
davis.h, 49
DAVIS240_CONFIG_BIAS_APSCASEPC
davis.h, 49
DAVIS240_CONFIG_BIAS_APSOVERFLOWLEVELBN
davis.h, 49
DAVIS240_CONFIG_BIAS_APSROSFBN
davis.h, 49
DAVIS240_CONFIG_BIAS_BIASBUFFER
davis.h, 50
DAVIS240_CONFIG_BIAS_DIFFBN
davis.h, 50

DAVIS240_CONFIG_BIAS_DIFFCASBNC
davis.h, 50
DAVIS240_CONFIG_BIAS_IFREFRBN
davis.h, 50
DAVIS240_CONFIG_BIAS_IFTHRBN
davis.h, 51
DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN
davis.h, 51
DAVIS240_CONFIG_BIAS_LOCALBUFBN
davis.h, 51
DAVIS240_CONFIG_BIAS_OFFBN
davis.h, 51
DAVIS240_CONFIG_BIAS_ONBN
davis.h, 52
DAVIS240_CONFIG_BIAS_PADFOLLBN
davis.h, 52
DAVIS240_CONFIG_BIAS_PIXINVBN
davis.h, 52
DAVIS240_CONFIG_BIAS_PRBP
davis.h, 52
DAVIS240_CONFIG_BIAS_PRSFBP
davis.h, 53
DAVIS240_CONFIG_BIAS_REFRBP
davis.h, 53
DAVIS240_CONFIG_BIAS_SSN
davis.h, 53
DAVIS240_CONFIG_BIAS_SSP
davis.h, 53
DAVIS240_CONFIG_CHIP_AERNAROW
davis.h, 54
DAVIS240_CONFIG_CHIP_ANALOGMUX0
davis.h, 54
DAVIS240_CONFIG_CHIP_ANALOGMUX1
davis.h, 54
DAVIS240_CONFIG_CHIP_ANALOGMUX2
davis.h, 54
DAVIS240_CONFIG_CHIP_BIASMUX0
davis.h, 54
DAVIS240_CONFIG_CHIP_DIGITALMUX0
davis.h, 55
DAVIS240_CONFIG_CHIP_DIGITALMUX1
davis.h, 55
DAVIS240_CONFIG_CHIP_DIGITALMUX2
davis.h, 55
DAVIS240_CONFIG_CHIP_DIGITALMUX3
davis.h, 55
DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER
davis.h, 55
DAVIS240_CONFIG_CHIP_RESETCALIBNEURON
davis.h, 55
DAVIS240_CONFIG_CHIP_RESETTESTPIXEL
davis.h, 56
DAVIS240_CONFIG_CHIP_SPECIALPIXELCONTROL
davis.h, 56
DAVIS240_CONFIG_CHIP_TYPENCALIBNEURON
davis.h, 56
DAVIS240_CONFIG_CHIP_USEAOUT
davis.h, 56
DAVIS346_CONFIG_BIAS_ADCCOMPBP
davis.h, 56
DAVIS346_CONFIG_BIAS_ADCREFHIGH
davis.h, 57
DAVIS346_CONFIG_BIAS_ADCREFLOW
davis.h, 57
DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE
davis.h, 57
DAVIS346_CONFIG_BIAS_AEPDBN
davis.h, 58
DAVIS346_CONFIG_BIAS_AEPUXBP
davis.h, 58
DAVIS346_CONFIG_BIAS_AEPUYBP
davis.h, 58
DAVIS346_CONFIG_BIAS_APSCAS
davis.h, 59
DAVIS346_CONFIG_BIAS_APSOVERFLOWLEVEL
davis.h, 59
DAVIS346_CONFIG_BIAS_APSROSFBN
davis.h, 59
DAVIS346_CONFIG_BIAS_BIASBUFFER
davis.h, 60
DAVIS346_CONFIG_BIAS_COLSELLOWBN
davis.h, 60
DAVIS346_CONFIG_BIAS_DACBUFBP
davis.h, 60
DAVIS346_CONFIG_BIAS_DIFFBN
davis.h, 61
DAVIS346_CONFIG_BIAS_IFREFRBN
davis.h, 61
DAVIS346_CONFIG_BIAS_IFTHRBN
davis.h, 61
DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN
davis.h, 62
DAVIS346_CONFIG_BIAS_LOCALBUFBN
davis.h, 62
DAVIS346_CONFIG_BIAS_OFFBN
davis.h, 62
DAVIS346_CONFIG_BIAS_ONBN
davis.h, 63
DAVIS346_CONFIG_BIAS_PADFOLLBN
davis.h, 63
DAVIS346_CONFIG_BIAS_PIXINVBN
davis.h, 63
DAVIS346_CONFIG_BIAS_PRBP
davis.h, 64
DAVIS346_CONFIG_BIAS_PRSFBP
davis.h, 64
DAVIS346_CONFIG_BIAS_READOUTBUFBP
davis.h, 64
DAVIS346_CONFIG_BIAS_REFRBP
davis.h, 65
DAVIS346_CONFIG_BIAS_SSN
davis.h, 65
DAVIS346_CONFIG_BIAS_SSP
davis.h, 65
DAVIS346_CONFIG_CHIP_AERNAROW
davis.h, 66

DAVIS346_CONFIG_CHIP_ANALOGMUX0
davis.h, 66
DAVIS346_CONFIG_CHIP_ANALOGMUX1
davis.h, 66
DAVIS346_CONFIG_CHIP_ANALOGMUX2
davis.h, 66
DAVIS346_CONFIG_CHIP_BIASMUX0
davis.h, 66
DAVIS346_CONFIG_CHIP_DIGITALMUX0
davis.h, 67
DAVIS346_CONFIG_CHIP_DIGITALMUX1
davis.h, 67
DAVIS346_CONFIG_CHIP_DIGITALMUX2
davis.h, 67
DAVIS346_CONFIG_CHIP_DIGITALMUX3
davis.h, 67
DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER
davis.h, 67
DAVIS346_CONFIG_CHIP_RESETCALIBNEURON
davis.h, 67
DAVIS346_CONFIG_CHIP_RESETTESTPIXEL
davis.h, 68
DAVIS346_CONFIG_CHIP_SELECTGRAYCOUNTER
davis.h, 68
DAVIS346_CONFIG_CHIP_TESTADC
davis.h, 68
DAVIS346_CONFIG_CHIP_TYPENCALIBNEURON
davis.h, 68
DAVIS346_CONFIG_CHIP_USEAOUT
davis.h, 68
DAVIS640_CONFIG_BIAS_ADCCOMPBP
davis.h, 68
DAVIS640_CONFIG_BIAS_ADCREFHIGH
davis.h, 69
DAVIS640_CONFIG_BIAS_ADCREFLOW
davis.h, 69
DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE
davis.h, 69
DAVIS640_CONFIG_BIAS_AEPDBN
davis.h, 70
DAVIS640_CONFIG_BIAS_AEPUXBP
davis.h, 70
DAVIS640_CONFIG_BIAS_AEPUYBP
davis.h, 70
DAVIS640_CONFIG_BIAS_APSCAS
davis.h, 71
DAVIS640_CONFIG_BIAS_APSOVERFLOWLEVEL
davis.h, 71
DAVIS640_CONFIG_BIAS_APSROSFBN
davis.h, 71
DAVIS640_CONFIG_BIAS_BIASBUFFER
davis.h, 72
DAVIS640_CONFIG_BIAS_COLSELLOWBN
davis.h, 72
DAVIS640_CONFIG_BIAS_DACBUFBP
davis.h, 72
DAVIS640_CONFIG_BIAS_DIFFBN
davis.h, 73
DAVIS640_CONFIG_BIAS_IFREFRBN
davis.h, 73
DAVIS640_CONFIG_BIAS_IFTHRBN
davis.h, 73
DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN
davis.h, 74
DAVIS640_CONFIG_BIAS_LOCALBUFBN
davis.h, 74
DAVIS640_CONFIG_BIAS_OFFBN
davis.h, 74
DAVIS640_CONFIG_BIAS_ONBN
davis.h, 75
DAVIS640_CONFIG_BIAS_PADFOLLBN
davis.h, 75
DAVIS640_CONFIG_BIAS_PIXINVBN
davis.h, 75
DAVIS640_CONFIG_BIAS_PRBP
davis.h, 76
DAVIS640_CONFIG_BIAS_PRSFBP
davis.h, 76
DAVIS640_CONFIG_BIAS_READOUTBUFBP
davis.h, 76
DAVIS640_CONFIG_BIAS_REFRBP
davis.h, 77
DAVIS640_CONFIG_BIAS_SSN
davis.h, 77
DAVIS640_CONFIG_BIAS_SSP
davis.h, 77
DAVIS640_CONFIG_CHIP_AERNAROW
davis.h, 78
DAVIS640_CONFIG_CHIP_ANALOGMUX0
davis.h, 78
DAVIS640_CONFIG_CHIP_ANALOGMUX1
davis.h, 78
DAVIS640_CONFIG_CHIP_ANALOGMUX2
davis.h, 78
DAVIS640_CONFIG_CHIP_BIASMUX0
davis.h, 78
DAVIS640_CONFIG_CHIP_DIGITALMUX0
davis.h, 79
DAVIS640_CONFIG_CHIP_DIGITALMUX1
davis.h, 79
DAVIS640_CONFIG_CHIP_DIGITALMUX2
davis.h, 79
DAVIS640_CONFIG_CHIP_DIGITALMUX3
davis.h, 79
DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER
davis.h, 79
DAVIS640_CONFIG_CHIP_RESETCALIBNEURON
davis.h, 79
DAVIS640_CONFIG_CHIP_RESETTESTPIXEL
davis.h, 80
DAVIS640_CONFIG_CHIP_SELECTGRAYCOUNTER
davis.h, 80
DAVIS640_CONFIG_CHIP_TESTADC
davis.h, 80
DAVIS640_CONFIG_CHIP_TYPENCALIBNEURON
davis.h, 80

DAVIS640_CONFIG_CHIP_USEAOUT
davis.h, 80
DAVIS640H_CONFIG_APS_GSFDRESET
davis.h, 80
DAVIS640H_CONFIG_APS_GSPDRESET
davis.h, 81
DAVIS640H_CONFIG_APS_GSRESETFALL
davis.h, 81
DAVIS640H_CONFIG_APS_GSTXFALL
davis.h, 81
DAVIS640H_CONFIG_APS_RSFDSETTLE
davis.h, 81
DAVIS640H_CONFIG_APS_TRANSFER
davis.h, 81
DAVIS640H_CONFIG_BIAS_ADCCOMPBP
davis.h, 81
DAVIS640H_CONFIG_BIAS_ADCREFHIGH
davis.h, 82
DAVIS640H_CONFIG_BIAS_ADCREFLOW
davis.h, 82
DAVIS640H_CONFIG_BIAS_ADCTESTVOLTAGE
davis.h, 82
DAVIS640H_CONFIG_BIAS_AEPDBN
davis.h, 83
DAVIS640H_CONFIG_BIAS_AEPUXBP
davis.h, 83
DAVIS640H_CONFIG_BIAS_AEPUYBP
davis.h, 83
DAVIS640H_CONFIG_BIAS_APSCAS
davis.h, 84
DAVIS640H_CONFIG_BIAS_APSROSFBN
davis.h, 84
DAVIS640H_CONFIG_BIAS_ARRAYBIASBUFFERBN
davis.h, 84
DAVIS640H_CONFIG_BIAS_ARRAYLOGICBUFFE↩
RBN
davis.h, 85
DAVIS640H_CONFIG_BIAS_BIASBUFFER
davis.h, 85
DAVIS640H_CONFIG_BIAS_DACBUFBP
davis.h, 85
DAVIS640H_CONFIG_BIAS_DIFFBN
davis.h, 86
DAVIS640H_CONFIG_BIAS_FALLTIMEBN
davis.h, 86
DAVIS640H_CONFIG_BIAS_GND07
davis.h, 86
DAVIS640H_CONFIG_BIAS_IFREFRBN
davis.h, 87
DAVIS640H_CONFIG_BIAS_IFTHRBN
davis.h, 87
DAVIS640H_CONFIG_BIAS_LCOLTIMEOUTBN
davis.h, 87
DAVIS640H_CONFIG_BIAS_LOCALBUFBN
davis.h, 88
DAVIS640H_CONFIG_BIAS_OFFBN
davis.h, 88
DAVIS640H_CONFIG_BIAS_ONBN
davis.h, 88
DAVIS640H_CONFIG_BIAS_OVG1LO
davis.h, 89
DAVIS640H_CONFIG_BIAS_OVG2LO
davis.h, 89
DAVIS640H_CONFIG_BIAS_PADFOLLBN
davis.h, 89
DAVIS640H_CONFIG_BIAS_PIXINVBN
davis.h, 90
DAVIS640H_CONFIG_BIAS_PRBP
davis.h, 90
DAVIS640H_CONFIG_BIAS_PRSFBP
davis.h, 90
DAVIS640H_CONFIG_BIAS_READOUTBUFBP
davis.h, 91
DAVIS640H_CONFIG_BIAS_REFRBP
davis.h, 91
DAVIS640H_CONFIG_BIAS_RISETIMEBP
davis.h, 91
DAVIS640H_CONFIG_BIAS_SSN
davis.h, 92
DAVIS640H_CONFIG_BIAS_SSP
davis.h, 92
DAVIS640H_CONFIG_BIAS_TX2OVG2HI
davis.h, 92
DAVIS640H_CONFIG_CHIP_ADJUSTOVG1LO
davis.h, 93
DAVIS640H_CONFIG_CHIP_ADJUSTOVG2LO
davis.h, 93
DAVIS640H_CONFIG_CHIP_ADJUSTTX2OVG2HI
davis.h, 93
DAVIS640H_CONFIG_CHIP_AERNAROW
davis.h, 93
DAVIS640H_CONFIG_CHIP_ANALOGMUX0
davis.h, 93
DAVIS640H_CONFIG_CHIP_ANALOGMUX1
davis.h, 94
DAVIS640H_CONFIG_CHIP_ANALOGMUX2
davis.h, 94
DAVIS640H_CONFIG_CHIP_BIASMUX0
davis.h, 94
DAVIS640H_CONFIG_CHIP_DIGITALMUX0
davis.h, 94
DAVIS640H_CONFIG_CHIP_DIGITALMUX1
davis.h, 94
DAVIS640H_CONFIG_CHIP_DIGITALMUX2
davis.h, 94
DAVIS640H_CONFIG_CHIP_DIGITALMUX3
davis.h, 95
DAVIS640H_CONFIG_CHIP_RESETCALIBNEURON
davis.h, 95
DAVIS640H_CONFIG_CHIP_RESETTESTPIXEL
davis.h, 95
DAVIS640H_CONFIG_CHIP_SELECTGRAYCOUNT↩
ER
davis.h, 95
DAVIS640H_CONFIG_CHIP_TESTADC
davis.h, 95

DAVIS640H_CONFIG_CHIP_TYPENCALIBNEURON
davis.h, 95
DAVIS640H_CONFIG_CHIP_USEAOUT
davis.h, 96
DAVIS_CHIP_DAVIS128
davis.h, 96
DAVIS_CHIP_DAVIS208
davis.h, 96
DAVIS_CHIP_DAVIS240A
davis.h, 96
DAVIS_CHIP_DAVIS240B
davis.h, 96
DAVIS_CHIP_DAVIS240C
davis.h, 96
DAVIS_CHIP_DAVIS346A
davis.h, 96
DAVIS_CHIP_DAVIS346B
davis.h, 97
DAVIS_CHIP_DAVIS346C
davis.h, 97
DAVIS_CHIP_DAVIS640
davis.h, 97
DAVIS_CHIP_DAVIS640H
davis.h, 97
DAVIS_CONFIG_APS_AUTOEXPOSURE
davis.h, 97
DAVIS_CONFIG_APS_COLOR_FILTER
davis.h, 97
DAVIS_CONFIG_APS_END_COLUMN_0
davis.h, 98
DAVIS_CONFIG_APS_END_ROW_0
davis.h, 98
DAVIS_CONFIG_APS_EXPOSURE
davis.h, 98
DAVIS_CONFIG_APS_FRAME_INTERVAL
davis.h, 98
DAVIS_CONFIG_APS_FRAME_MODE
davis.h, 98
DAVIS_CONFIG_APS_GLOBAL_SHUTTER
davis.h, 98
DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTTER
davis.h, 99
DAVIS_CONFIG_APS_ORIENTATION_INFO
davis.h, 99
DAVIS_CONFIG_APS_RUN
davis.h, 99
DAVIS_CONFIG_APS_SIZE_COLUMNS
davis.h, 99
DAVIS_CONFIG_APS_SIZE_ROWS
davis.h, 99
DAVIS_CONFIG_APS_SNAPSHOT
davis.h, 99
DAVIS_CONFIG_APS_START_COLUMN_0
davis.h, 100
DAVIS_CONFIG_APS_START_ROW_0
davis.h, 100
DAVIS_CONFIG_APS_WAIT_ON_TRANSFER_STALL
davis.h, 100
DAVIS_CONFIG_APS
davis.h, 97
DAVIS_CONFIG_BIAS
davis.h, 100
DAVIS_CONFIG_CHIP
davis.h, 100
DAVIS_CONFIG_DDRAER_RUN
davis.h, 101
DAVIS_CONFIG_DDRAER
davis.h, 100
DAVIS_CONFIG_DVS_EXTERNAL_AER_CONTROL
davis.h, 101
DAVIS_CONFIG_DVS_FILTER_BACKGROUND_AC↵
TIVITY_TIME
davis.h, 101
DAVIS_CONFIG_DVS_FILTER_BACKGROUND_AC↵
TIVITY
davis.h, 101
DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COLUMN
davis.h, 101
DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW
davis.h, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COLUMN
davis.h, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW
davis.h, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COLUMN
davis.h, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW
davis.h, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COLUMN
davis.h, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW
davis.h, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COLUMN
davis.h, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW
davis.h, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COLUMN
davis.h, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW
davis.h, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COLUMN
davis.h, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW
davis.h, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COLUMN
davis.h, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW
davis.h, 104
DAVIS_CONFIG_DVS_FILTER_PIXEL_AUTO_TRAIN
davis.h, 104
DAVIS_CONFIG_DVS_FILTER_POLARITY_FLATTEN
davis.h, 104
DAVIS_CONFIG_DVS_FILTER_POLARITY_SUPPR↵
ESS_TYPE
davis.h, 104

DAVIS_CONFIG_DVS_FILTER_POLARITY_SUPPR↩ESS
davis.h, 104
DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PE↩RIOD_TIME
davis.h, 104
DAVIS_CONFIG_DVS_FILTER_REFRACTORY_PE↩RIOD
davis.h, 104
DAVIS_CONFIG_DVS_FILTER_ROI_END_COLUMN
davis.h, 105
DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW
davis.h, 105
DAVIS_CONFIG_DVS_FILTER_ROI_START_COLU↩MN
davis.h, 105
DAVIS_CONFIG_DVS_FILTER_ROI_START_ROW
davis.h, 105
DAVIS_CONFIG_DVS_FILTER_SKIP_EVENTS_EV↩ERY
davis.h, 105
DAVIS_CONFIG_DVS_FILTER_SKIP_EVENTS
davis.h, 105
DAVIS_CONFIG_DVS_HAS_BACKGROUND_ACTI↩VITY_FILTER
davis.h, 106
DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER
davis.h, 106
DAVIS_CONFIG_DVS_HAS_POLARITY_FILTER
davis.h, 106
DAVIS_CONFIG_DVS_HAS_ROI_FILTER
davis.h, 106
DAVIS_CONFIG_DVS_HAS_SKIP_FILTER
davis.h, 106
DAVIS_CONFIG_DVS_HAS_STATISTICS
davis.h, 106
DAVIS_CONFIG_DVS_ORIENTATION_INFO
davis.h, 107
DAVIS_CONFIG_DVS_RUN
davis.h, 107
DAVIS_CONFIG_DVS_SIZE_COLUMNS
davis.h, 107
DAVIS_CONFIG_DVS_SIZE_ROWS
davis.h, 107
DAVIS_CONFIG_DVS_STATISTICS_EVENTS_COL↩UMN
davis.h, 107
DAVIS_CONFIG_DVS_STATISTICS_EVENTS_DRO↩PPED
davis.h, 107
DAVIS_CONFIG_DVS_STATISTICS_EVENTS_ROW
davis.h, 108
DAVIS_CONFIG_DVS_STATISTICS_FILTERED_BA↩CKGROUND_ACTIVITY
davis.h, 108
DAVIS_CONFIG_DVS_STATISTICS_FILTERED_PI↩XELS
davis.h, 108
DAVIS_CONFIG_DVS_STATISTICS_FILTERED_RE↩FRACTORY_PERIOD
davis.h, 108
DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER_STALL
davis.h, 108
DAVIS_CONFIG_DVS
davis.h, 101
DAVIS_CONFIG_EXTINPUT_DETECT_FALLING_E↩DGES
davis.h, 109
DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_LE↩NGTH
davis.h, 109
DAVIS_CONFIG_EXTINPUT_DETECT_PULSE_PO↩LARITY
davis.h, 109
DAVIS_CONFIG_EXTINPUT_DETECT_PULSES
davis.h, 109
DAVIS_CONFIG_EXTINPUT_DETECT_RISING_ED↩GES
davis.h, 109
DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_↩ON_FALLING_EDGE
davis.h, 109
DAVIS_CONFIG_EXTINPUT_GENERATE_INJECT_↩ON_RISING_EDGE
davis.h, 110
DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_I↩NTERVAL
davis.h, 110
DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_↩LENGTH
davis.h, 110
DAVIS_CONFIG_EXTINPUT_GENERATE_PULSE_↩POLARITY
davis.h, 110
DAVIS_CONFIG_EXTINPUT_HAS_GENERATOR
davis.h, 110
DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR
davis.h, 110
DAVIS_CONFIG_EXTINPUT_RUN_GENERATOR
davis.h, 111
DAVIS_CONFIG_EXTINPUT
davis.h, 108
DAVIS_CONFIG_IMU_ACCEL_DLPF
davis.h, 111
DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE
davis.h, 111
DAVIS_CONFIG_IMU_GYRO_DLPF
davis.h, 111
DAVIS_CONFIG_IMU_GYRO_FULL_SCALE
davis.h, 112
DAVIS_CONFIG_IMU_ORIENTATION_INFO
davis.h, 112
DAVIS_CONFIG_IMU_RUN_ACCELEROMETER
davis.h, 112
DAVIS_CONFIG_IMU_RUN_GYROSCOPE
davis.h, 112

DAVIS_CONFIG_IMU_RUN_TEMPERATURE
davis.h, 112
DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVIDER
davis.h, 112
DAVIS_CONFIG_IMU_TYPE
davis.h, 113
DAVIS_CONFIG_IMU
davis.h, 111
DAVIS_CONFIG_MUX_DROP_DVS_ON_TRANSFE←
R_STALL
davis.h, 113
DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON_TR←
ANSFER_STALL
davis.h, 113
DAVIS_CONFIG_MUX_HAS_STATISTICS
davis.h, 113
DAVIS_CONFIG_MUX_RUN_CHIP
davis.h, 114
DAVIS_CONFIG_MUX_RUN
davis.h, 113
DAVIS_CONFIG_MUX_STATISTICS_DVS_DROPPED
davis.h, 114
DAVIS_CONFIG_MUX_STATISTICS_EXTINPUT_D←
ROPPED
davis.h, 114
DAVIS_CONFIG_MUX_TIMESTAMP_RESET
davis.h, 114
DAVIS_CONFIG_MUX_TIMESTAMP_RUN
davis.h, 114
DAVIS_CONFIG_MUX
davis.h, 113
DAVIS_CONFIG_SYSINFO_ADC_CLOCK
davis.h, 115
DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER
davis.h, 115
DAVIS_CONFIG_SYSINFO_CLOCK_DEVIATION
davis.h, 115
DAVIS_CONFIG_SYSINFO_DEVICE_IS_MASTER
davis.h, 115
DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK
davis.h, 115
DAVIS_CONFIG_SYSINFO_LOGIC_VERSION
davis.h, 115
DAVIS_CONFIG_SYSINFO_USB_CLOCK
davis.h, 116
DAVIS_CONFIG_SYSINFO
davis.h, 114
DAVIS_CONFIG_USB_EARLY_PACKET_DELAY
davis.h, 116
DAVIS_CONFIG_USB_RUN
davis.h, 116
DAVIS_CONFIG_USB
davis.h, 116
DVS128_CONFIG_BIAS_CAS
dvs128.h, 134
DVS128_CONFIG_BIAS_DIFFOFF
dvs128.h, 135
DVS128_CONFIG_BIAS_DIFFON
dvs128.h, 135
DVS128_CONFIG_BIAS_DIFF
dvs128.h, 135
DVS128_CONFIG_BIAS_FOLL
dvs128.h, 135
DVS128_CONFIG_BIAS_INJGND
dvs128.h, 135
DVS128_CONFIG_BIAS_PUX
dvs128.h, 135
DVS128_CONFIG_BIAS_PUY
dvs128.h, 136
DVS128_CONFIG_BIAS_PR
dvs128.h, 135
DVS128_CONFIG_BIAS_REFR
dvs128.h, 136
DVS128_CONFIG_BIAS_REQPD
dvs128.h, 136
DVS128_CONFIG_BIAS_REQ
dvs128.h, 136
DVS128_CONFIG_BIAS
dvs128.h, 134
DVS128_CONFIG_DVS_ARRAY_RESET
dvs128.h, 136
DVS128_CONFIG_DVS_RUN
dvs128.h, 136
DVS128_CONFIG_DVS_TIMESTAMP_RESET
dvs128.h, 137
DVS128_CONFIG_DVS_TS_MASTER
dvs128.h, 137
DVS128_CONFIG_DVS
dvs128.h, 136
DYNAPSE_CHIP_DYNAPSE
dynapse.h, 142
DYNAPSE_CONFIG_AER_ACK_DELAY
dynapse.h, 143
DYNAPSE_CONFIG_AER_ACK_EXTENSION
dynapse.h, 143
DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONT←
ROL
dynapse.h, 143
DYNAPSE_CONFIG_AER_HAS_STATISTICS
dynapse.h, 143
DYNAPSE_CONFIG_AER_RUN
dynapse.h, 143
DYNAPSE_CONFIG_AER_STATISTICS_EVENTS_←
DROPPED
dynapse.h, 144
DYNAPSE_CONFIG_AER_STATISTICS_EVENTS
dynapse.h, 143
DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_←
STALL
dynapse.h, 144
DYNAPSE_CONFIG_AER
dynapse.h, 143
DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P
dynapse.h, 144
DYNAPSE_CONFIG_CHIP_CONTENT
dynapse.h, 144

DYNAPSE_CONFIG_CHIP_ID
dynapse.h, 144
DYNAPSE_CONFIG_CHIP_REQ_DELAY
dynapse.h, 145
DYNAPSE_CONFIG_CHIP_REQ_EXTENSION
dynapse.h, 145
DYNAPSE_CONFIG_CHIP_RUN
dynapse.h, 145
DYNAPSE_CONFIG_CHIP
dynapse.h, 144
DYNAPSE_CONFIG_CLEAR_CAM
dynapse.h, 145
DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY
dynapse.h, 145
DYNAPSE_CONFIG_DEFAULT_SRAM
dynapse.h, 145
DYNAPSE_CONFIG_MONITOR_NEU
dynapse.h, 145
DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRAN↩SFER_STALL
dynapse.h, 146
DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_E↩NABLE
dynapse.h, 146
DYNAPSE_CONFIG_MUX_HAS_STATISTICS
dynapse.h, 146
DYNAPSE_CONFIG_MUX_RUN
dynapse.h, 146
DYNAPSE_CONFIG_MUX_STATISTICS_AER_DRO↩PPED
dynapse.h, 146
DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET
dynapse.h, 147
DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN
dynapse.h, 147
DYNAPSE_CONFIG_MUX
dynapse.h, 146
DYNAPSE_CONFIG_POISSONSPIKEGEN_CHIPID
dynapse.h, 147
DYNAPSE_CONFIG_POISSONSPIKEGEN_RUN
dynapse.h, 147
DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITE↩ADDRESS
dynapse.h, 147
DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITE↩DATA
dynapse.h, 147
DYNAPSE_CONFIG_POISSONSPIKEGEN
dynapse.h, 147
DYNAPSE_CONFIG_SPIKEGEN_BASEADDR
dynapse.h, 148
DYNAPSE_CONFIG_SPIKEGEN_ISIBASE
dynapse.h, 148
DYNAPSE_CONFIG_SPIKEGEN_ISI
dynapse.h, 148
DYNAPSE_CONFIG_SPIKEGEN_REPEAT
dynapse.h, 148
DYNAPSE_CONFIG_SPIKEGEN_RUN
dynapse.h, 148
DYNAPSE_CONFIG_SPIKEGEN_STIMCOUNT
dynapse.h, 148
DYNAPSE_CONFIG_SPIKEGEN_VARMODE
dynapse.h, 149
DYNAPSE_CONFIG_SPIKEGEN
dynapse.h, 148
DYNAPSE_CONFIG_SRAM_ADDRESS
dynapse.h, 149
DYNAPSE_CONFIG_SRAM_BURSTMODE
dynapse.h, 149
DYNAPSE_CONFIG_SRAM_DIRECTION_POS
dynapse.h, 149
DYNAPSE_CONFIG_SRAM_READDATA
dynapse.h, 150
DYNAPSE_CONFIG_SRAM_READ
dynapse.h, 149
DYNAPSE_CONFIG_SRAM_RWCOMMAND
dynapse.h, 150
DYNAPSE_CONFIG_SRAM_WRITEDATA
dynapse.h, 150
DYNAPSE_CONFIG_SRAM_WRITE
dynapse.h, 150
DYNAPSE_CONFIG_SRAM
dynapse.h, 149
DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPS↩ELECT
dynapse.h, 150
DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOB↩ALKERNEL
dynapse.h, 151
DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN
dynapse.h, 151
DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAM↩BASEADDR
dynapse.h, 151
DYNAPSE_CONFIG_SYNAPSERECONFIG_USES↩RAMKERNELS
dynapse.h, 151
DYNAPSE_CONFIG_SYNAPSERECONFIG
dynapse.h, 150
DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER
dynapse.h, 151
DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER
dynapse.h, 152
DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK
dynapse.h, 152
DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION
dynapse.h, 152
DYNAPSE_CONFIG_SYSINFO
dynapse.h, 151
DYNAPSE_CONFIG_TAU1_RESET
dynapse.h, 152
DYNAPSE_CONFIG_TAU2_RESET
dynapse.h, 152
DYNAPSE_CONFIG_TAU2_SET
dynapse.h, 152
DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY

dynapse.h, 153
DYNAPSE_CONFIG_USB_RUN
dynapse.h, 153
DYNAPSE_CONFIG_USB
dynapse.h, 153
DYNAPSE_X4BOARD_COREX
dynapse.h, 153
DYNAPSE_X4BOARD_COREY
dynapse.h, 153
DYNAPSE_X4BOARD_NEUX
dynapse.h, 153
DYNAPSE_X4BOARD_NEUY
dynapse.h, 154
DYNAPSE_X4BOARD_NUMCHIPS
dynapse.h, 154
davis.h
CAER_DEVICE_DAVIS_FX2, 23
CAER_DEVICE_DAVIS_FX3, 23
CAER_DEVICE_DAVIS_RPI, 23
CAER_DEVICE_DAVIS, 23
caer_bias_shiftedsource_operating_mode, 118
caer_bias_shiftedsource_voltage_level, 119
caer_davis_aps_frame_modes, 119
caer_davis_imu_invensense_accel_scale, 119
caer_davis_imu_invensense_gyro_scale, 119
caer_davis_imu_types, 119
caerBiasCoarseFineFromCurrent, 120
caerBiasCoarseFineGenerate, 120
caerBiasCoarseFineParse, 120
caerBiasCoarseFineToCurrent, 121
caerBiasShiftedSourceGenerate, 121
caerBiasShiftedSourceParse, 121
caerBiasVDACGenerate, 122
caerBiasVDACParse, 122
caerDavisInfoGet, 122
caerDavisROIConfigure, 123
DAVIS128_CONFIG_BIAS_ADCCOMPBP, 23
DAVIS128_CONFIG_BIAS_ADCREFHIGH, 23
DAVIS128_CONFIG_BIAS_ADCREFLOW, 24
DAVIS128_CONFIG_BIAS_AEPDBN, 24
DAVIS128_CONFIG_BIAS_AEPUXBP, 24
DAVIS128_CONFIG_BIAS_AEPUYBP, 25
DAVIS128_CONFIG_BIAS_APSCAS, 25
DAVIS128_CONFIG_BIAS_APSOVERFLOWLE↩
VEL, 25
DAVIS128_CONFIG_BIAS_APSROSFBN, 26
DAVIS128_CONFIG_BIAS_BIASBUFFER, 26
DAVIS128_CONFIG_BIAS_COLSELLOWBN, 26
DAVIS128_CONFIG_BIAS_DACBUFBP, 27
DAVIS128_CONFIG_BIAS_DIFFBN, 27
DAVIS128_CONFIG_BIAS_IFREFRBN, 27
DAVIS128_CONFIG_BIAS_IFTHRBN, 28
DAVIS128_CONFIG_BIAS_LCOLTIMEOUTBN,
28
DAVIS128_CONFIG_BIAS_LOCALBUFBN, 28
DAVIS128_CONFIG_BIAS_OFFBN, 29
DAVIS128_CONFIG_BIAS_ONBN, 29
DAVIS128_CONFIG_BIAS_PADFOLLBN, 29
DAVIS128_CONFIG_BIAS_PIXINVBN, 30
DAVIS128_CONFIG_BIAS_PRBP, 30
DAVIS128_CONFIG_BIAS_PRSFBP, 30
DAVIS128_CONFIG_BIAS_READOUTBUFBP, 31
DAVIS128_CONFIG_BIAS_REFRBP, 31
DAVIS128_CONFIG_BIAS_SSN, 31
DAVIS128_CONFIG_BIAS_SSP, 32
DAVIS128_CONFIG_CHIP_AERNAROW, 32
DAVIS128_CONFIG_CHIP_ANALOGMUX0, 32
DAVIS128_CONFIG_CHIP_ANALOGMUX1, 32
DAVIS128_CONFIG_CHIP_ANALOGMUX2, 33
DAVIS128_CONFIG_CHIP_BIASMUX0, 33
DAVIS128_CONFIG_CHIP_DIGITALMUX0, 33
DAVIS128_CONFIG_CHIP_DIGITALMUX1, 33
DAVIS128_CONFIG_CHIP_DIGITALMUX2, 33
DAVIS128_CONFIG_CHIP_DIGITALMUX3, 33
DAVIS128_CONFIG_CHIP_GLOBAL_SHUTTER,
34
DAVIS128_CONFIG_CHIP_RESETCALIBNEU↩
RON, 34
DAVIS128_CONFIG_CHIP_RESETTESTPIXEL,
34
DAVIS128_CONFIG_CHIP_SELECTGRAYCO↩
UNTER, 34
DAVIS128_CONFIG_CHIP_TYPENCALIBNEU↩
RON, 34
DAVIS128_CONFIG_CHIP_USEAOUT, 34
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_APSOVERFLOWLE↩
VEL, 37
DAVIS208_CONFIG_BIAS_APSROSFBN, 37
DAVIS208_CONFIG_BIAS_BIASBUFFER, 37
DAVIS208_CONFIG_BIAS_COLSELLOWBN, 38
DAVIS208_CONFIG_BIAS_DACBUFBP, 38
DAVIS208_CONFIG_BIAS_DIFFBN, 38
DAVIS208_CONFIG_BIAS_IFREFRBN, 39
DAVIS208_CONFIG_BIAS_IFTHRBN, 39
DAVIS208_CONFIG_BIAS_LCOLTIMEOUTBN,
39
DAVIS208_CONFIG_BIAS_LOCALBUFBN, 40
DAVIS208_CONFIG_BIAS_OFFBN, 40
DAVIS208_CONFIG_BIAS_ONBN, 40
DAVIS208_CONFIG_BIAS_PADFOLLBN, 41
DAVIS208_CONFIG_BIAS_PIXINVBN, 41
DAVIS208_CONFIG_BIAS_PRBP, 41
DAVIS208_CONFIG_BIAS_PRSFBP, 42
DAVIS208_CONFIG_BIAS_READOUTBUFBP, 42
DAVIS208_CONFIG_BIAS_REFRBP, 42
DAVIS208_CONFIG_BIAS_REFSSBN, 43
DAVIS208_CONFIG_BIAS_REFSS, 43
DAVIS208_CONFIG_BIAS_REGBIASBP, 43

DAVIS208_CONFIG_BIAS_RESETHIGHPASS, 44
DAVIS208_CONFIG_BIAS_SSN, 44
DAVIS208_CONFIG_BIAS_SSP, 44
DAVIS208_CONFIG_CHIP_AERNAROW, 45
DAVIS208_CONFIG_CHIP_ANALOGMUX0, 45
DAVIS208_CONFIG_CHIP_ANALOGMUX1, 45
DAVIS208_CONFIG_CHIP_ANALOGMUX2, 45
DAVIS208_CONFIG_CHIP_BIASMUX0, 45
DAVIS208_CONFIG_CHIP_DIGITALMUX0, 46
DAVIS208_CONFIG_CHIP_DIGITALMUX1, 46
DAVIS208_CONFIG_CHIP_DIGITALMUX2, 46
DAVIS208_CONFIG_CHIP_DIGITALMUX3, 46
DAVIS208_CONFIG_CHIP_GLOBAL_SHUTTER, 46
DAVIS208_CONFIG_CHIP_RESETCALIBNEU←RON, 46
DAVIS208_CONFIG_CHIP_RESETTESTPIXEL, 47
DAVIS208_CONFIG_CHIP_SELECTBIASREFSS, 47
DAVIS208_CONFIG_CHIP_SELECTGRAYCO←UNTER, 47
DAVIS208_CONFIG_CHIP_SELECTHIGHPASS, 47
DAVIS208_CONFIG_CHIP_SELECTPOSFB, 47
DAVIS208_CONFIG_CHIP_SELECTPREAMPA←VG, 47
DAVIS208_CONFIG_CHIP_SELECTSENSE, 48
DAVIS208_CONFIG_CHIP_TYPENCALIBNEU←RON, 48
DAVIS208_CONFIG_CHIP_USEAOUT, 48
DAVIS240_CONFIG_BIAS_AEPDBN, 48
DAVIS240_CONFIG_BIAS_AEPUXBP, 48
DAVIS240_CONFIG_BIAS_AEPUYBP, 49
DAVIS240_CONFIG_BIAS_APSCASEPC, 49
DAVIS240_CONFIG_BIAS_APSOVERFLOWLE←VELBN, 49
DAVIS240_CONFIG_BIAS_APSROSFBN, 49
DAVIS240_CONFIG_BIAS_BIASBUFFER, 50
DAVIS240_CONFIG_BIAS_DIFFBN, 50
DAVIS240_CONFIG_BIAS_DIFFCASBNC, 50
DAVIS240_CONFIG_BIAS_IFREFRBN, 50
DAVIS240_CONFIG_BIAS_IFTHRBN, 51
DAVIS240_CONFIG_BIAS_LCOLTIMEOUTBN, 51
DAVIS240_CONFIG_BIAS_LOCALBUFBN, 51
DAVIS240_CONFIG_BIAS_OFFBN, 51
DAVIS240_CONFIG_BIAS_ONBN, 52
DAVIS240_CONFIG_BIAS_PADFOLLBN, 52
DAVIS240_CONFIG_BIAS_PIXINVBN, 52
DAVIS240_CONFIG_BIAS_PRBP, 52
DAVIS240_CONFIG_BIAS_PRSFBP, 53
DAVIS240_CONFIG_BIAS_REFRBP, 53
DAVIS240_CONFIG_BIAS_SSN, 53
DAVIS240_CONFIG_BIAS_SSP, 53
DAVIS240_CONFIG_CHIP_AERNAROW, 54
DAVIS240_CONFIG_CHIP_ANALOGMUX0, 54
DAVIS240_CONFIG_CHIP_ANALOGMUX1, 54
DAVIS240_CONFIG_CHIP_ANALOGMUX2, 54
DAVIS240_CONFIG_CHIP_BIASMUX0, 54
DAVIS240_CONFIG_CHIP_DIGITALMUX0, 55
DAVIS240_CONFIG_CHIP_DIGITALMUX1, 55
DAVIS240_CONFIG_CHIP_DIGITALMUX2, 55
DAVIS240_CONFIG_CHIP_DIGITALMUX3, 55
DAVIS240_CONFIG_CHIP_GLOBAL_SHUTTER, 55
DAVIS240_CONFIG_CHIP_RESETCALIBNEU←RON, 55
DAVIS240_CONFIG_CHIP_RESETTESTPIXEL, 56
DAVIS240_CONFIG_CHIP_SPECIALPIXELCO←NTROL, 56
DAVIS240_CONFIG_CHIP_TYPENCALIBNEU←RON, 56
DAVIS240_CONFIG_CHIP_USEAOUT, 56
DAVIS346_CONFIG_BIAS_ADCCOMPBP, 56
DAVIS346_CONFIG_BIAS_ADCREFHIGH, 57
DAVIS346_CONFIG_BIAS_ADCREFLOW, 57
DAVIS346_CONFIG_BIAS_ADCTESTVOLTAGE, 57
DAVIS346_CONFIG_BIAS_AEPDBN, 58
DAVIS346_CONFIG_BIAS_AEPUXBP, 58
DAVIS346_CONFIG_BIAS_AEPUYBP, 58
DAVIS346_CONFIG_BIAS_APSCAS, 59
DAVIS346_CONFIG_BIAS_APSOVERFLOWLE←VEL, 59
DAVIS346_CONFIG_BIAS_APSROSFBN, 59
DAVIS346_CONFIG_BIAS_BIASBUFFER, 60
DAVIS346_CONFIG_BIAS_COLSELLOWBN, 60
DAVIS346_CONFIG_BIAS_DACBUFBP, 60
DAVIS346_CONFIG_BIAS_DIFFBN, 61
DAVIS346_CONFIG_BIAS_IFREFRBN, 61
DAVIS346_CONFIG_BIAS_IFTHRBN, 61
DAVIS346_CONFIG_BIAS_LCOLTIMEOUTBN, 62
DAVIS346_CONFIG_BIAS_LOCALBUFBN, 62
DAVIS346_CONFIG_BIAS_OFFBN, 62
DAVIS346_CONFIG_BIAS_ONBN, 63
DAVIS346_CONFIG_BIAS_PADFOLLBN, 63
DAVIS346_CONFIG_BIAS_PIXINVBN, 63
DAVIS346_CONFIG_BIAS_PRBP, 64
DAVIS346_CONFIG_BIAS_PRSFBP, 64
DAVIS346_CONFIG_BIAS_READOUTBUFBP, 64
DAVIS346_CONFIG_BIAS_REFRBP, 65
DAVIS346_CONFIG_BIAS_SSN, 65
DAVIS346_CONFIG_BIAS_SSP, 65
DAVIS346_CONFIG_CHIP_AERNAROW, 66
DAVIS346_CONFIG_CHIP_ANALOGMUX0, 66
DAVIS346_CONFIG_CHIP_ANALOGMUX1, 66
DAVIS346_CONFIG_CHIP_ANALOGMUX2, 66
DAVIS346_CONFIG_CHIP_BIASMUX0, 66
DAVIS346_CONFIG_CHIP_DIGITALMUX0, 67
DAVIS346_CONFIG_CHIP_DIGITALMUX1, 67
DAVIS346_CONFIG_CHIP_DIGITALMUX2, 67
DAVIS346_CONFIG_CHIP_DIGITALMUX3, 67

DAVIS346_CONFIG_CHIP_GLOBAL_SHUTTER, 67
DAVIS346_CONFIG_CHIP_RESETCALIBNEU↩RON, 67
DAVIS346_CONFIG_CHIP_RESETTESTPIXEL, 68
DAVIS346_CONFIG_CHIP_SELECTGRAYCO↩UNTER, 68
DAVIS346_CONFIG_CHIP_TESTADC, 68
DAVIS346_CONFIG_CHIP_TYPENCALIBNEU↩RON, 68
DAVIS346_CONFIG_CHIP_USEAOUT, 68
DAVIS640_CONFIG_BIAS_ADCCOMPBP, 68
DAVIS640_CONFIG_BIAS_ADCREFHIGH, 69
DAVIS640_CONFIG_BIAS_ADCREFLOW, 69
DAVIS640_CONFIG_BIAS_ADCTESTVOLTAGE, 69
DAVIS640_CONFIG_BIAS_AEPDBN, 70
DAVIS640_CONFIG_BIAS_AEPUXBP, 70
DAVIS640_CONFIG_BIAS_AEPUYBP, 70
DAVIS640_CONFIG_BIAS_APSCAS, 71
DAVIS640_CONFIG_BIAS_APSOVERFLOWLE↩VEL, 71
DAVIS640_CONFIG_BIAS_APSROSFBN, 71
DAVIS640_CONFIG_BIAS_BIASBUFFER, 72
DAVIS640_CONFIG_BIAS_COLSELLOWBN, 72
DAVIS640_CONFIG_BIAS_DACBUFBP, 72
DAVIS640_CONFIG_BIAS_DIFFBN, 73
DAVIS640_CONFIG_BIAS_IFREFRBN, 73
DAVIS640_CONFIG_BIAS_IFTHRBN, 73
DAVIS640_CONFIG_BIAS_LCOLTIMEOUTBN, 74
DAVIS640_CONFIG_BIAS_LOCALBUFBN, 74
DAVIS640_CONFIG_BIAS_OFFBN, 74
DAVIS640_CONFIG_BIAS_ONBN, 75
DAVIS640_CONFIG_BIAS_PADFOLLBN, 75
DAVIS640_CONFIG_BIAS_PIXINVBN, 75
DAVIS640_CONFIG_BIAS_PRBP, 76
DAVIS640_CONFIG_BIAS_PRSFBP, 76
DAVIS640_CONFIG_BIAS_READOUTBUFBP, 76
DAVIS640_CONFIG_BIAS_REFRBP, 77
DAVIS640_CONFIG_BIAS_SSN, 77
DAVIS640_CONFIG_BIAS_SSP, 77
DAVIS640_CONFIG_CHIP_AERNAROW, 78
DAVIS640_CONFIG_CHIP_ANALOGMUX0, 78
DAVIS640_CONFIG_CHIP_ANALOGMUX1, 78
DAVIS640_CONFIG_CHIP_ANALOGMUX2, 78
DAVIS640_CONFIG_CHIP_BIASMUX0, 78
DAVIS640_CONFIG_CHIP_DIGITALMUX0, 79
DAVIS640_CONFIG_CHIP_DIGITALMUX1, 79
DAVIS640_CONFIG_CHIP_DIGITALMUX2, 79
DAVIS640_CONFIG_CHIP_DIGITALMUX3, 79
DAVIS640_CONFIG_CHIP_GLOBAL_SHUTTER, 79
DAVIS640_CONFIG_CHIP_RESETCALIBNEU↩RON, 79
DAVIS640_CONFIG_CHIP_RESETTESTPIXEL, 80
DAVIS640_CONFIG_CHIP_SELECTGRAYCO↩UNTER, 80
DAVIS640_CONFIG_CHIP_TESTADC, 80
DAVIS640_CONFIG_CHIP_TYPENCALIBNEU↩RON, 80
DAVIS640_CONFIG_CHIP_USEAOUT, 80
DAVIS640H_CONFIG_APS_GSFDRESET, 80
DAVIS640H_CONFIG_APS_GSPDRESET, 81
DAVIS640H_CONFIG_APS_GSRESETFALL, 81
DAVIS640H_CONFIG_APS_GSTXFALL, 81
DAVIS640H_CONFIG_APS_RSFDSETTLE, 81
DAVIS640H_CONFIG_APS_TRANSFER, 81
DAVIS640H_CONFIG_BIAS_ADCCOMPBP, 81
DAVIS640H_CONFIG_BIAS_ADCREFHIGH, 82
DAVIS640H_CONFIG_BIAS_ADCREFLOW, 82
DAVIS640H_CONFIG_BIAS_ADCTESTVOLTA↩GE, 82
DAVIS640H_CONFIG_BIAS_AEPDBN, 83
DAVIS640H_CONFIG_BIAS_AEPUXBP, 83
DAVIS640H_CONFIG_BIAS_AEPUYBP, 83
DAVIS640H_CONFIG_BIAS_APSCAS, 84
DAVIS640H_CONFIG_BIAS_APSROSFBN, 84
DAVIS640H_CONFIG_BIAS_ARRAYBIASBUF↩FERBN, 84
DAVIS640H_CONFIG_BIAS_ARRAYLOGICBU↩FFERBN, 85
DAVIS640H_CONFIG_BIAS_BIASBUFFER, 85
DAVIS640H_CONFIG_BIAS_DACBUFBP, 85
DAVIS640H_CONFIG_BIAS_DIFFBN, 86
DAVIS640H_CONFIG_BIAS_FALLTIMEBN, 86
DAVIS640H_CONFIG_BIAS_GND07, 86
DAVIS640H_CONFIG_BIAS_IFREFRBN, 87
DAVIS640H_CONFIG_BIAS_IFTHRBN, 87
DAVIS640H_CONFIG_BIAS_LCOLTIMEOUTBN, 87
DAVIS640H_CONFIG_BIAS_LOCALBUFBN, 88
DAVIS640H_CONFIG_BIAS_OFFBN, 88
DAVIS640H_CONFIG_BIAS_ONBN, 88
DAVIS640H_CONFIG_BIAS_OVG1LO, 89
DAVIS640H_CONFIG_BIAS_OVG2LO, 89
DAVIS640H_CONFIG_BIAS_PADFOLLBN, 89
DAVIS640H_CONFIG_BIAS_PIXINVBN, 90
DAVIS640H_CONFIG_BIAS_PRBP, 90
DAVIS640H_CONFIG_BIAS_PRSFBP, 90
DAVIS640H_CONFIG_BIAS_READOUTBUFBP, 91
DAVIS640H_CONFIG_BIAS_REFRBP, 91
DAVIS640H_CONFIG_BIAS_RISETIMEBP, 91
DAVIS640H_CONFIG_BIAS_SSN, 92
DAVIS640H_CONFIG_BIAS_SSP, 92
DAVIS640H_CONFIG_BIAS_TX2OVG2HI, 92
DAVIS640H_CONFIG_CHIP_ADJUSTOVG1LO, 93
DAVIS640H_CONFIG_CHIP_ADJUSTOVG2LO, 93
DAVIS640H_CONFIG_CHIP_ADJUSTTX2OV↩G2HI, 93
DAVIS640H_CONFIG_CHIP_AERNAROW, 93

DAVIS640H_CONFIG_CHIP_ANALOGMUX0, 93
DAVIS640H_CONFIG_CHIP_ANALOGMUX1, 94
DAVIS640H_CONFIG_CHIP_ANALOGMUX2, 94
DAVIS640H_CONFIG_CHIP_BIASMUX0, 94
DAVIS640H_CONFIG_CHIP_DIGITALMUX0, 94
DAVIS640H_CONFIG_CHIP_DIGITALMUX1, 94
DAVIS640H_CONFIG_CHIP_DIGITALMUX2, 94
DAVIS640H_CONFIG_CHIP_DIGITALMUX3, 95
DAVIS640H_CONFIG_CHIP_RESETCALIBNE←URON, 95
DAVIS640H_CONFIG_CHIP_RESETTESTPIXEL, 95
DAVIS640H_CONFIG_CHIP_SELECTGRAYC←OUNTER, 95
DAVIS640H_CONFIG_CHIP_TESTADC, 95
DAVIS640H_CONFIG_CHIP_TYPENCALIBNE←URON, 95
DAVIS640H_CONFIG_CHIP_USEAOUT, 96
DAVIS_CHIP_DAVIS128, 96
DAVIS_CHIP_DAVIS208, 96
DAVIS_CHIP_DAVIS240A, 96
DAVIS_CHIP_DAVIS240B, 96
DAVIS_CHIP_DAVIS240C, 96
DAVIS_CHIP_DAVIS346A, 96
DAVIS_CHIP_DAVIS346B, 97
DAVIS_CHIP_DAVIS346C, 97
DAVIS_CHIP_DAVIS640, 97
DAVIS_CHIP_DAVIS640H, 97
DAVIS_CONFIG_APS_AUTOEXPOSURE, 97
DAVIS_CONFIG_APS_COLOR_FILTER, 97
DAVIS_CONFIG_APS_END_COLUMN_0, 98
DAVIS_CONFIG_APS_END_ROW_0, 98
DAVIS_CONFIG_APS_EXPOSURE, 98
DAVIS_CONFIG_APS_FRAME_INTERVAL, 98
DAVIS_CONFIG_APS_FRAME_MODE, 98
DAVIS_CONFIG_APS_GLOBAL_SHUTTER, 98
DAVIS_CONFIG_APS_HAS_GLOBAL_SHUTT←ER, 99
DAVIS_CONFIG_APS_ORIENTATION_INFO, 99
DAVIS_CONFIG_APS_RUN, 99
DAVIS_CONFIG_APS_SIZE_COLUMNS, 99
DAVIS_CONFIG_APS_SIZE_ROWS, 99
DAVIS_CONFIG_APS_SNAPSHOT, 99
DAVIS_CONFIG_APS_START_COLUMN_0, 100
DAVIS_CONFIG_APS_START_ROW_0, 100
DAVIS_CONFIG_APS_WAIT_ON_TRANSFER←_STALL, 100
DAVIS_CONFIG_APS, 97
DAVIS_CONFIG_BIAS, 100
DAVIS_CONFIG_CHIP, 100
DAVIS_CONFIG_DDRAER_RUN, 101
DAVIS_CONFIG_DDRAER, 100
DAVIS_CONFIG_DVS_EXTERNAL_AER_CON←TROL, 101
DAVIS_CONFIG_DVS_FILTER_BACKGROUN←D_ACTIVITY_TIME, 101
DAVIS_CONFIG_DVS_FILTER_BACKGROUN←D_ACTIVITY, 101
DAVIS_CONFIG_DVS_FILTER_PIXEL_0_COL←UMN, 101
DAVIS_CONFIG_DVS_FILTER_PIXEL_0_ROW, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_1_COL←UMN, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_1_ROW, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_2_COL←UMN, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_2_ROW, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_3_COL←UMN, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_3_ROW, 102
DAVIS_CONFIG_DVS_FILTER_PIXEL_4_COL←UMN, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_4_ROW, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_5_COL←UMN, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_5_ROW, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_6_COL←UMN, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_6_ROW, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_7_COL←UMN, 103
DAVIS_CONFIG_DVS_FILTER_PIXEL_7_ROW, 104
DAVIS_CONFIG_DVS_FILTER_PIXEL_AUTO_←TRAIN, 104
DAVIS_CONFIG_DVS_FILTER_POLARITY_FL←ATTEN, 104
DAVIS_CONFIG_DVS_FILTER_POLARITY_SU←PPRESS_TYPE, 104
DAVIS_CONFIG_DVS_FILTER_POLARITY_SU←PPRESS, 104
DAVIS_CONFIG_DVS_FILTER_REFRACTORY←_PERIOD_TIME, 104
DAVIS_CONFIG_DVS_FILTER_REFRACTORY←_PERIOD, 104
DAVIS_CONFIG_DVS_FILTER_ROI_END_CO←LUMN, 105
DAVIS_CONFIG_DVS_FILTER_ROI_END_ROW, 105
DAVIS_CONFIG_DVS_FILTER_ROI_START_C←OLUMN, 105
DAVIS_CONFIG_DVS_FILTER_ROI_START_R←OW, 105
DAVIS_CONFIG_DVS_FILTER_SKIP_EVENTS←_EVERY, 105
DAVIS_CONFIG_DVS_FILTER_SKIP_EVENTS, 105
DAVIS_CONFIG_DVS_HAS_BACKGROUND_←ACTIVITY_FILTER, 106

DAVIS_CONFIG_DVS_HAS_PIXEL_FILTER, 106
DAVIS_CONFIG_DVS_HAS_POLARITY_FILTER, 106
DAVIS_CONFIG_DVS_HAS_ROI_FILTER, 106
DAVIS_CONFIG_DVS_HAS_SKIP_FILTER, 106
DAVIS_CONFIG_DVS_HAS_STATISTICS, 106
DAVIS_CONFIG_DVS_ORIENTATION_INFO, 107
DAVIS_CONFIG_DVS_RUN, 107
DAVIS_CONFIG_DVS_SIZE_COLUMNS, 107
DAVIS_CONFIG_DVS_SIZE_ROWS, 107
DAVIS_CONFIG_DVS_STATISTICS_EVENTS_↩COLUMN, 107
DAVIS_CONFIG_DVS_STATISTICS_EVENTS_↩DROPPED, 107
DAVIS_CONFIG_DVS_STATISTICS_EVENTS_↩ROW, 108
DAVIS_CONFIG_DVS_STATISTICS_FILTERE↩D_BACKGROUND_ACTIVITY, 108
DAVIS_CONFIG_DVS_STATISTICS_FILTERE↩D_PIXELS, 108
DAVIS_CONFIG_DVS_STATISTICS_FILTERE↩D_REFRACTORY_PERIOD, 108
DAVIS_CONFIG_DVS_WAIT_ON_TRANSFER↩_STALL, 108
DAVIS_CONFIG_DVS, 101
DAVIS_CONFIG_EXTINPUT_DETECT_FALLIN↩G_EDGES, 109
DAVIS_CONFIG_EXTINPUT_DETECT_PULSE↩_LENGTH, 109
DAVIS_CONFIG_EXTINPUT_DETECT_PULSE↩_POLARITY, 109
DAVIS_CONFIG_EXTINPUT_DETECT_PULSES, 109
DAVIS_CONFIG_EXTINPUT_DETECT_RISING↩_EDGES, 109
DAVIS_CONFIG_EXTINPUT_GENERATE_INJ↩ECT_ON_FALLING_EDGE, 109
DAVIS_CONFIG_EXTINPUT_GENERATE_INJ↩ECT_ON_RISING_EDGE, 110
DAVIS_CONFIG_EXTINPUT_GENERATE_PUL↩SE_INTERVAL, 110
DAVIS_CONFIG_EXTINPUT_GENERATE_PUL↩SE_LENGTH, 110
DAVIS_CONFIG_EXTINPUT_GENERATE_PUL↩SE_POLARITY, 110
DAVIS_CONFIG_EXTINPUT_HAS_GENERAT↩OR, 110
DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR, 110
DAVIS_CONFIG_EXTINPUT_RUN_GENERAT↩OR, 111
DAVIS_CONFIG_EXTINPUT, 108
DAVIS_CONFIG_IMU_ACCEL_DLPF, 111
DAVIS_CONFIG_IMU_ACCEL_FULL_SCALE, 111
DAVIS_CONFIG_IMU_GYRO_DLPF, 111
DAVIS_CONFIG_IMU_GYRO_FULL_SCALE, 112
DAVIS_CONFIG_IMU_ORIENTATION_INFO, 112
DAVIS_CONFIG_IMU_RUN_ACCELEROMETER, 112
DAVIS_CONFIG_IMU_RUN_GYROSCOPE, 112
DAVIS_CONFIG_IMU_RUN_TEMPERATURE, 112
DAVIS_CONFIG_IMU_SAMPLE_RATE_DIVID↩ER, 112
DAVIS_CONFIG_IMU_TYPE, 113
DAVIS_CONFIG_IMU, 111
DAVIS_CONFIG_MUX_DROP_DVS_ON_TRA↩NSFER_STALL, 113
DAVIS_CONFIG_MUX_DROP_EXTINPUT_ON↩_TRANSFER_STALL, 113
DAVIS_CONFIG_MUX_HAS_STATISTICS, 113
DAVIS_CONFIG_MUX_RUN_CHIP, 114
DAVIS_CONFIG_MUX_RUN, 113
DAVIS_CONFIG_MUX_STATISTICS_DVS_DR↩OPPED, 114
DAVIS_CONFIG_MUX_STATISTICS_EXTINPU↩T_DROPPED, 114
DAVIS_CONFIG_MUX_TIMESTAMP_RESET, 114
DAVIS_CONFIG_MUX_TIMESTAMP_RUN, 114
DAVIS_CONFIG_MUX, 113
DAVIS_CONFIG_SYSINFO_ADC_CLOCK, 115
DAVIS_CONFIG_SYSINFO_CHIP_IDENTIFIER, 115
DAVIS_CONFIG_SYSINFO_CLOCK_DEVIATI↩ON, 115
DAVIS_CONFIG_SYSINFO_DEVICE_IS_MAST↩ER, 115
DAVIS_CONFIG_SYSINFO_LOGIC_CLOCK, 115
DAVIS_CONFIG_SYSINFO_LOGIC_VERSION, 115
DAVIS_CONFIG_SYSINFO_USB_CLOCK, 116
DAVIS_CONFIG_SYSINFO, 114
DAVIS_CONFIG_USB_EARLY_PACKET_DELAY, 116
DAVIS_CONFIG_USB_RUN, 116
DAVIS_CONFIG_USB, 116
IS_DAVIS128, 116
IS_DAVIS208, 116
IS_DAVIS240, 117
IS_DAVIS240A, 117
IS_DAVIS240B, 117
IS_DAVIS240C, 117
IS_DAVIS346, 117
IS_DAVIS346A, 117
IS_DAVIS346B, 118
IS_DAVIS346C, 118
IS_DAVIS640, 118
IS_DAVIS640H, 118
device.h
CAER_HOST_CONFIG_DATAEXCHANGE_BL↩OCKING, 124
CAER_HOST_CONFIG_DATAEXCHANGE_BU↩FFER_SIZE, 125

CAER_HOST_CONFIG_DATAEXCHANGE_ST↩ART_PRODUCERS, 125
CAER_HOST_CONFIG_DATAEXCHANGE_ST↩OP_PRODUCERS, 125
CAER_HOST_CONFIG_DATAEXCHANGE, 124
CAER_HOST_CONFIG_LOG_LEVEL, 125
CAER_HOST_CONFIG_LOG, 125
CAER_HOST_CONFIG_PACKETS_MAX_CON↩TAINER_INTERVAL, 126
CAER_HOST_CONFIG_PACKETS_MAX_CON↩TAINER_PACKET_SIZE, 126
CAER_HOST_CONFIG_PACKETS, 125
CAER_SUPPORTED_DEVICES_NUMBER, 126
caerDeviceClose, 126
caerDeviceConfigGet, 127
caerDeviceConfigGet64, 127
caerDeviceConfigSet, 128
caerDeviceDataGet, 128
caerDeviceDataStart, 130
caerDeviceDataStop, 130
caerDeviceHandle, 126
caerDeviceSendDefaultConfig, 131

device_discover.h
CAER_DEVICE_DISCOVER_ALL, 132
caerDeviceDiscover, 132
caerDeviceDiscoverOpen, 133
caerDeviceDiscoveryResult, 132

deviceString
caer_davis_info, 8
caer_dvs128_info, 10
caer_dynapse_info, 11
caer_edvs_info, 12

devices/davis.h, 13
devices/device.h, 123
devices/device_discover.h, 131
devices/dvs128.h, 133
devices/dynapse.h, 137
devices/edvs.h, 162
devices/serial.h, 166
devices/usb.h, 168

dvs128.h
CAER_DEVICE_DVS128, 134
caerDVS128InfoGet, 137
DVS128_CONFIG_BIAS_CAS, 134
DVS128_CONFIG_BIAS_DIFFOFF, 135
DVS128_CONFIG_BIAS_DIFFON, 135
DVS128_CONFIG_BIAS_DIFF, 135
DVS128_CONFIG_BIAS_FOLL, 135
DVS128_CONFIG_BIAS_INJGND, 135
DVS128_CONFIG_BIAS_PUX, 135
DVS128_CONFIG_BIAS_PUY, 136
DVS128_CONFIG_BIAS_PR, 135
DVS128_CONFIG_BIAS_REFR, 136
DVS128_CONFIG_BIAS_REQPD, 136
DVS128_CONFIG_BIAS_REQ, 136
DVS128_CONFIG_BIAS, 134
DVS128_CONFIG_DVS_ARRAY_RESET, 136
DVS128_CONFIG_DVS_RUN, 136
DVS128_CONFIG_DVS_TIMESTAMP_RESET, 137
DVS128_CONFIG_DVS_TS_MASTER, 137
DVS128_CONFIG_DVS, 136

dvs_noise.h
CAER_FILTER_DVS_BACKGROUND_ACTIVIT↩Y_CHECK_POLARITY, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVIT↩Y_ENABLE, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVIT↩Y_STATISTICS_OFF, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVIT↩Y_STATISTICS_ON, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVIT↩Y_STATISTICS, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVIT↩Y_SUPPORT_MAX, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVIT↩Y_SUPPORT_MIN, 423
CAER_FILTER_DVS_BACKGROUND_ACTIVIT↩Y_TIME, 424
CAER_FILTER_DVS_BACKGROUND_ACTIVIT↩Y_TWO_LEVELS, 424
CAER_FILTER_DVS_HOTPIXEL_COUNT, 424
CAER_FILTER_DVS_HOTPIXEL_ENABLE, 424
CAER_FILTER_DVS_HOTPIXEL_LEARN, 424
CAER_FILTER_DVS_HOTPIXEL_STATISTICS↩_OFF, 424
CAER_FILTER_DVS_HOTPIXEL_STATISTICS↩_ON, 425
CAER_FILTER_DVS_HOTPIXEL_STATISTICS, 424
CAER_FILTER_DVS_HOTPIXEL_TIME, 425
CAER_FILTER_DVS_LOG_LEVEL, 425
CAER_FILTER_DVS_REFRACTORY_PERIOD↩_ENABLE, 425
CAER_FILTER_DVS_REFRACTORY_PERIOD↩_STATISTICS_OFF, 425
CAER_FILTER_DVS_REFRACTORY_PERIOD↩_STATISTICS_ON, 425
CAER_FILTER_DVS_REFRACTORY_PERIOD↩_STATISTICS, 425
CAER_FILTER_DVS_REFRACTORY_PERIOD↩_TIME, 425
CAER_FILTER_DVS_RESET, 426
caerFilterDVSNoise, 426
caerFilterDVSNoiseApply, 426
caerFilterDVSNoiseConfigGet, 427
caerFilterDVSNoiseConfigSet, 427
caerFilterDVSNoiseDestroy, 427
caerFilterDVSNoiseGetHotPixels, 428
caerFilterDVSNoiseInitialize, 428
caerFilterDVSNoiseStatsApply, 428
caerFilterDVSPixel, 426

dynapse.h
CAER_DEVICE_DYNAPSE, 142
caerBiasDynapseGenerate, 154
caerBiasDynapseParse, 154

caerDynapseCoreAddrToNeuronId, 155
caerDynapseCoreXYToNeuronId, 155
caerDynapseGenerateCamBits, 155
caerDynapseGenerateSramBits, 156
caerDynapseInfoGet, 157
caerDynapseSendDataToUSB, 157
caerDynapseSpikeEventFromXY, 157
caerDynapseSpikeEventGetX, 158
caerDynapseSpikeEventGetY, 158
caerDynapseWriteCam, 159
caerDynapseWritePoissonSpikeRate, 159
caerDynapseWriteSram, 160
caerDynapseWriteSramWords, 161
caerDynapseWriteSramN, 160
DYNAPSE_CHIP_DYNAPSE, 142
DYNAPSE_CONFIG_AER_ACK_DELAY, 143
DYNAPSE_CONFIG_AER_ACK_EXTENSION, 143
DYNAPSE_CONFIG_AER_EXTERNAL_AER_CONTROL, 143
DYNAPSE_CONFIG_AER_HAS_STATISTICS, 143
DYNAPSE_CONFIG_AER_RUN, 143
DYNAPSE_CONFIG_AER_STATISTICS_EVENTS_DROPPED, 144
DYNAPSE_CONFIG_AER_STATISTICS_EVENTS, 143
DYNAPSE_CONFIG_AER_WAIT_ON_TRANSFER_STALL, 144
DYNAPSE_CONFIG_AER, 143
DYNAPSE_CONFIG_BIAS_C0_PULSE_PWLK_P, 144
DYNAPSE_CONFIG_CHIP_CONTENT, 144
DYNAPSE_CONFIG_CHIP_ID, 144
DYNAPSE_CONFIG_CHIP_REQ_DELAY, 145
DYNAPSE_CONFIG_CHIP_REQ_EXTENSION, 145
DYNAPSE_CONFIG_CHIP_RUN, 145
DYNAPSE_CONFIG_CHIP, 144
DYNAPSE_CONFIG_CLEAR_CAM, 145
DYNAPSE_CONFIG_DEFAULT_SRAM_EMPTY, 145
DYNAPSE_CONFIG_DEFAULT_SRAM, 145
DYNAPSE_CONFIG_MONITOR_NEU, 145
DYNAPSE_CONFIG_MUX_DROP_AER_ON_TRANSFER_STALL, 146
DYNAPSE_CONFIG_MUX_FORCE_CHIP_BIAS_ENABLE, 146
DYNAPSE_CONFIG_MUX_HAS_STATISTICS, 146
DYNAPSE_CONFIG_MUX_RUN, 146
DYNAPSE_CONFIG_MUX_STATISTICS_AER_DROPPED, 146
DYNAPSE_CONFIG_MUX_TIMESTAMP_RESET, 147
DYNAPSE_CONFIG_MUX_TIMESTAMP_RUN, 147
DYNAPSE_CONFIG_MUX, 146
DYNAPSE_CONFIG_POISSONSPIKEGEN_CHIPID, 147
DYNAPSE_CONFIG_POISSONSPIKEGEN_RUN, 147
DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEADDRESS, 147
DYNAPSE_CONFIG_POISSONSPIKEGEN_WRITEDATA, 147
DYNAPSE_CONFIG_POISSONSPIKEGEN, 147
DYNAPSE_CONFIG_SPIKEGEN_BASEADDR, 148
DYNAPSE_CONFIG_SPIKEGEN_ISIBASE, 148
DYNAPSE_CONFIG_SPIKEGEN_ISI, 148
DYNAPSE_CONFIG_SPIKEGEN_REPEAT, 148
DYNAPSE_CONFIG_SPIKEGEN_RUN, 148
DYNAPSE_CONFIG_SPIKEGEN_STIMCOUNT, 148
DYNAPSE_CONFIG_SPIKEGEN_VARMODE, 149
DYNAPSE_CONFIG_SPIKEGEN, 148
DYNAPSE_CONFIG_SRAM_ADDRESS, 149
DYNAPSE_CONFIG_SRAM_BURSTMODE, 149
DYNAPSE_CONFIG_SRAM_DIRECTION_POS, 149
DYNAPSE_CONFIG_SRAM_READDATA, 150
DYNAPSE_CONFIG_SRAM_READ, 149
DYNAPSE_CONFIG_SRAM_RWCOMMAND, 150
DYNAPSE_CONFIG_SRAM_WRITEDATA, 150
DYNAPSE_CONFIG_SRAM_WRITE, 150
DYNAPSE_CONFIG_SRAM, 149
DYNAPSE_CONFIG_SYNAPSERECONFIG_CHIPSELECT, 150
DYNAPSE_CONFIG_SYNAPSERECONFIG_GLOBALKERNEL, 151
DYNAPSE_CONFIG_SYNAPSERECONFIG_RUN, 151
DYNAPSE_CONFIG_SYNAPSERECONFIG_SRAMBASEADDR, 151
DYNAPSE_CONFIG_SYNAPSERECONFIG_USESRAMKERNELS, 151
DYNAPSE_CONFIG_SYNAPSERECONFIG, 150
DYNAPSE_CONFIG_SYSINFO_CHIP_IDENTIFIER, 151
DYNAPSE_CONFIG_SYSINFO_DEVICE_IS_MASTER, 152
DYNAPSE_CONFIG_SYSINFO_LOGIC_CLOCK, 152
DYNAPSE_CONFIG_SYSINFO_LOGIC_VERSION, 152
DYNAPSE_CONFIG_SYSINFO, 151
DYNAPSE_CONFIG_TAU1_RESET, 152
DYNAPSE_CONFIG_TAU2_RESET, 152
DYNAPSE_CONFIG_TAU2_SET, 152
DYNAPSE_CONFIG_USB_EARLY_PACKET_DELAY, 153
DYNAPSE_CONFIG_USB_RUN, 153
DYNAPSE_CONFIG_USB, 153
DYNAPSE_X4BOARD_COREX, 153

DYNAPSE_X4BOARD_COREY, 153
DYNAPSE_X4BOARD_NEUX, 153
DYNAPSE_X4BOARD_NEUY, 154
DYNAPSE_X4BOARD_NUMCHIPS, 154

EAR_CHANNEL_MASK
ear.h, 204
EAR_CHANNEL_SHIFT
ear.h, 205
EAR_FILTER_MASK
ear.h, 205
EAR_FILTER_SHIFT
ear.h, 205
EAR_MASK
ear.h, 205
EAR_NEURON_MASK
ear.h, 205
EAR_NEURON_SHIFT
ear.h, 205
EAR_SHIFT
ear.h, 206
EDVS_CONFIG_BIAS_CAS
edvs.h, 163
EDVS_CONFIG_BIAS_DIFFOFF
edvs.h, 163
EDVS_CONFIG_BIAS_DIFFON
edvs.h, 163
EDVS_CONFIG_BIAS_DIFF
edvs.h, 163
EDVS_CONFIG_BIAS_FOLL
edvs.h, 163
EDVS_CONFIG_BIAS_INJGND
edvs.h, 164
EDVS_CONFIG_BIAS_PUX
edvs.h, 164
EDVS_CONFIG_BIAS_PUY
edvs.h, 164
EDVS_CONFIG_BIAS_PR
edvs.h, 164
EDVS_CONFIG_BIAS_REFR
edvs.h, 164
EDVS_CONFIG_BIAS_REQPD
edvs.h, 164
EDVS_CONFIG_BIAS_REQ
edvs.h, 164
EDVS_CONFIG_BIAS
edvs.h, 163
EDVS_CONFIG_DVS_RUN
edvs.h, 165
EDVS_CONFIG_DVS_TIMESTAMP_RESET
edvs.h, 165
EDVS_CONFIG_DVS
edvs.h, 165
ear.h
CAER_EAR_CONST_ITERATOR_ALL_START, 201
CAER_EAR_CONST_ITERATOR_VALID_STA↩RT, 201
CAER_EAR_CONST_REVERSE_ITERATOR_↩ALL_START, 202
CAER_EAR_CONST_REVERSE_ITERATOR_↩VALID_START, 202
CAER_EAR_ITERATOR_ALL_END, 202
CAER_EAR_ITERATOR_ALL_START, 203
CAER_EAR_ITERATOR_VALID_END, 203
CAER_EAR_ITERATOR_VALID_START, 203
CAER_EAR_REVERSE_ITERATOR_ALL_END, 203
CAER_EAR_REVERSE_ITERATOR_ALL_STA↩RT, 204
CAER_EAR_REVERSE_ITERATOR_VALID_E↩ND, 204
CAER_EAR_REVERSE_ITERATOR_VALID_S↩TART, 204
caerEarEvent, 206
caerEarEventGetChannel, 206
caerEarEventGetEar, 207
caerEarEventGetTimestamp, 207
caerEarEventGetTimestamp64, 207
caerEarEventInvalidate, 208
caerEarEventIsValid, 208
caerEarEventPacket, 206
caerEarEventPacketAllocate, 209
caerEarEventPacketFromPacketHeader, 209
caerEarEventPacketFromPacketHeaderConst, 209
caerEarEventPacketGetEvent, 211
caerEarEventPacketGetEventConst, 211
caerEarEventSetChannel, 211
caerEarEventSetEar, 212
caerEarEventSetTimestamp, 212
caerEarEventValidate, 212
EAR_CHANNEL_MASK, 204
EAR_CHANNEL_SHIFT, 205
EAR_FILTER_MASK, 205
EAR_FILTER_SHIFT, 205
EAR_MASK, 205
EAR_NEURON_MASK, 205
EAR_NEURON_SHIFT, 205
EAR_SHIFT, 206
PACKED_STRUCT, 213
edvs.h
CAER_DEVICE_EDVS, 163
caerEDVSInfoGet, 165
EDVS_CONFIG_BIAS_CAS, 163
EDVS_CONFIG_BIAS_DIFFOFF, 163
EDVS_CONFIG_BIAS_DIFFON, 163
EDVS_CONFIG_BIAS_DIFF, 163
EDVS_CONFIG_BIAS_FOLL, 163
EDVS_CONFIG_BIAS_INJGND, 164
EDVS_CONFIG_BIAS_PUX, 164
EDVS_CONFIG_BIAS_PUY, 164
EDVS_CONFIG_BIAS_PR, 164
EDVS_CONFIG_BIAS_REFR, 164
EDVS_CONFIG_BIAS_REQPD, 164
EDVS_CONFIG_BIAS_REQ, 164
EDVS_CONFIG_BIAS, 163

EDVS_CONFIG_DVS_RUN, 165
EDVS_CONFIG_DVS_TIMESTAMP_RESET, 165
EDVS_CONFIG_DVS, 165
events/common.h, 169
events/config.h, 187
events/ear.h, 200
events/frame.h, 213
events/imu6.h, 243
events/imu9.h, 259
events/matrix4x4.h, 278
events/packetContainer.h, 304
events/point1d.h, 312
events/point2d.h, 325
events/point3d.h, 338
events/point4d.h, 353
events/polarity.h, 368
events/sample.h, 381
events/special.h, 392
events/spike.h, 408

FRAME_COLOR_CHANNELS_MASK
frame.h, 219
FRAME_COLOR_CHANNELS_SHIFT
frame.h, 220
FRAME_COLOR_FILTER_MASK
frame.h, 220
FRAME_COLOR_FILTER_SHIFT
frame.h, 220
FRAME_ROI_IDENTIFIER_MASK
frame.h, 220
FRAME_ROI_IDENTIFIER_SHIFT
frame.h, 220
filters/dvs_noise.h, 421
frame.h
CAER_FRAME_CONST_ITERATOR_ALL_STA↩RT, 216
CAER_FRAME_CONST_ITERATOR_VALID_S↩TART, 216
CAER_FRAME_CONST_REVERSE_ITERATO↩R_ALL_START, 217
CAER_FRAME_CONST_REVERSE_ITERATO↩R_VALID_START, 217
CAER_FRAME_ITERATOR_ALL_END, 217
CAER_FRAME_ITERATOR_ALL_START, 218
CAER_FRAME_ITERATOR_VALID_END, 218
CAER_FRAME_ITERATOR_VALID_START, 218
CAER_FRAME_REVERSE_ITERATOR_ALL_E↩ND, 218
CAER_FRAME_REVERSE_ITERATOR_ALL_S↩TART, 219
CAER_FRAME_REVERSE_ITERATOR_VALID↩_END, 219
CAER_FRAME_REVERSE_ITERATOR_VALID↩_START, 219
caer_frame_event_color_channels, 221
caer_frame_event_color_filter, 221
caerFrameEvent, 221
caerFrameEventGetChannelNumber, 222
caerFrameEventGetColorFilter, 222
caerFrameEventGetExposureLength, 223
caerFrameEventGetLengthX, 223
caerFrameEventGetLengthY, 223
caerFrameEventGetPixel, 225
caerFrameEventGetPixelArrayUnsafe, 225
caerFrameEventGetPixelArrayUnsafeConst, 226
caerFrameEventGetPixelForChannel, 226
caerFrameEventGetPixelForChannelUnsafe, 226
caerFrameEventGetPixelUnsafe, 228
caerFrameEventGetPixelsMaxIndex, 227
caerFrameEventGetPixelsSize, 227
caerFrameEventGetPositionX, 228
caerFrameEventGetPositionY, 228
caerFrameEventGetROIIdentifier, 229
caerFrameEventGetTSEndOfExposure, 230
caerFrameEventGetTSEndOfExposure64, 230
caerFrameEventGetTSEndOfFrame, 231
caerFrameEventGetTSEndOfFrame64, 231
caerFrameEventGetTSStartOfExposure, 232
caerFrameEventGetTSStartOfExposure64, 232
caerFrameEventGetTSStartOfFrame, 232
caerFrameEventGetTSStartOfFrame64, 233
caerFrameEventGetTimestamp, 229
caerFrameEventGetTimestamp64, 230
caerFrameEventInvalidate, 233
caerFrameEventIsValid, 234
caerFrameEventPacket, 221
caerFrameEventPacketAllocate, 234
caerFrameEventPacketAllocateNumPixels, 234
caerFrameEventPacketFromPacketHeader, 235
caerFrameEventPacketFromPacketHeaderConst, 235
caerFrameEventPacketGetEvent, 236
caerFrameEventPacketGetEventConst, 236
caerFrameEventPacketGetPixelsMaxIndex, 237
caerFrameEventPacketGetPixelsSize, 237
caerFrameEventSetColorFilter, 237
caerFrameEventSetLengthXLengthYChannel↩Number, 238
caerFrameEventSetPixel, 238
caerFrameEventSetPixelForChannel, 239
caerFrameEventSetPixelForChannelUnsafe, 239
caerFrameEventSetPixelUnsafe, 240
caerFrameEventSetPositionX, 240
caerFrameEventSetPositionY, 240
caerFrameEventSetROIIdentifier, 241
caerFrameEventSetTSEndOfExposure, 241
caerFrameEventSetTSEndOfFrame, 241
caerFrameEventSetTSStartOfExposure, 242
caerFrameEventSetTSStartOfFrame, 242
caerFrameEventValidate, 242
FRAME_COLOR_CHANNELS_MASK, 219
FRAME_COLOR_CHANNELS_SHIFT, 220
FRAME_COLOR_FILTER_MASK, 220
FRAME_COLOR_FILTER_SHIFT, 220
FRAME_ROI_IDENTIFIER_MASK, 220
FRAME_ROI_IDENTIFIER_SHIFT, 220
PACKED_STRUCT, 242, 243

frame_utils.h, 429

GET_NUMBITS16
libcaer.h, 431
GET_NUMBITS32
libcaer.h, 432
GET_NUMBITS8
libcaer.h, 432

I16T
libcaer.h, 432
I32T
libcaer.h, 432
I64T
libcaer.h, 432
I8T
libcaer.h, 432
IS_DAVIS128
davis.h, 116
IS_DAVIS208
davis.h, 116
IS_DAVIS240
davis.h, 117
IS_DAVIS240A
davis.h, 117
IS_DAVIS240B
davis.h, 117
IS_DAVIS240C
davis.h, 117
IS_DAVIS346
davis.h, 117
IS_DAVIS346A
davis.h, 117
IS_DAVIS346B
davis.h, 118
IS_DAVIS346C
davis.h, 118
IS_DAVIS640
davis.h, 118
IS_DAVIS640H
davis.h, 118
imu6.h
CAER_IMU6_CONST_ITERATOR_ALL_START, 244
CAER_IMU6_CONST_ITERATOR_VALID_STA←RT, 245
CAER_IMU6_CONST_REVERSE_ITERATOR_←ALL_START, 245
CAER_IMU6_CONST_REVERSE_ITERATOR_←VALID_START, 245
CAER_IMU6_ITERATOR_ALL_END, 246
CAER_IMU6_ITERATOR_ALL_START, 246
CAER_IMU6_ITERATOR_VALID_END, 246
CAER_IMU6_ITERATOR_VALID_START, 246
CAER_IMU6_REVERSE_ITERATOR_ALL_END, 247
CAER_IMU6_REVERSE_ITERATOR_ALL_STA←RT, 247
CAER_IMU6_REVERSE_ITERATOR_VALID_E←ND, 247
CAER_IMU6_REVERSE_ITERATOR_VALID_S←TART, 247
caerIMU6Event, 248
caerIMU6EventGetAccelX, 248
caerIMU6EventGetAccelY, 249
caerIMU6EventGetAccelZ, 249
caerIMU6EventGetGyroX, 249
caerIMU6EventGetGyroY, 251
caerIMU6EventGetGyroZ, 251
caerIMU6EventGetTemp, 251
caerIMU6EventGetTimestamp, 253
caerIMU6EventGetTimestamp64, 253
caerIMU6EventInvalidate, 253
caerIMU6EventIsValid, 254
caerIMU6EventPacket, 248
caerIMU6EventPacketAllocate, 254
caerIMU6EventPacketFromPacketHeader, 255
caerIMU6EventPacketFromPacketHeaderConst, 255
caerIMU6EventPacketGetEvent, 255
caerIMU6EventPacketGetEventConst, 256
caerIMU6EventSetAccelX, 256
caerIMU6EventSetAccelY, 256
caerIMU6EventSetAccelZ, 257
caerIMU6EventSetGyroX, 257
caerIMU6EventSetGyroY, 257
caerIMU6EventSetGyroZ, 258
caerIMU6EventSetTemp, 258
caerIMU6EventSetTimestamp, 258
caerIMU6EventValidate, 259
PACKED_STRUCT, 259
imu9.h
CAER_IMU9_CONST_ITERATOR_ALL_START, 261
CAER_IMU9_CONST_ITERATOR_VALID_STA←RT, 261
CAER_IMU9_CONST_REVERSE_ITERATOR_←ALL_START, 262
CAER_IMU9_CONST_REVERSE_ITERATOR_←VALID_START, 262
CAER_IMU9_ITERATOR_ALL_END, 262
CAER_IMU9_ITERATOR_ALL_START, 263
CAER_IMU9_ITERATOR_VALID_END, 263
CAER_IMU9_ITERATOR_VALID_START, 263
CAER_IMU9_REVERSE_ITERATOR_ALL_END, 263
CAER_IMU9_REVERSE_ITERATOR_ALL_STA←RT, 264
CAER_IMU9_REVERSE_ITERATOR_VALID_E←ND, 264
CAER_IMU9_REVERSE_ITERATOR_VALID_S←TART, 264
caerIMU9Event, 265
caerIMU9EventGetAccelX, 265
caerIMU9EventGetAccelY, 265
caerIMU9EventGetAccelZ, 266

caerIMU9EventGetCompX, 266
caerIMU9EventGetCompY, 266
caerIMU9EventGetCompZ, 268
caerIMU9EventGetGyroX, 268
caerIMU9EventGetGyroY, 268
caerIMU9EventGetGyroZ, 270
caerIMU9EventGetTemp, 270
caerIMU9EventGetTimestamp, 270
caerIMU9EventGetTimestamp64, 271
caerIMU9EventInvalidate, 271
caerIMU9EventIsValid, 272
caerIMU9EventPacket, 265
caerIMU9EventPacketAllocate, 272
caerIMU9EventPacketFromPacketHeader, 272
caerIMU9EventPacketFromPacketHeaderConst, 273
caerIMU9EventPacketGetEvent, 273
caerIMU9EventPacketGetEventConst, 273
caerIMU9EventSetAccelX, 274
caerIMU9EventSetAccelY, 274
caerIMU9EventSetAccelZ, 274
caerIMU9EventSetCompX, 275
caerIMU9EventSetCompY, 275
caerIMU9EventSetCompZ, 275
caerIMU9EventSetGyroX, 276
caerIMU9EventSetGyroY, 276
caerIMU9EventSetGyroZ, 276
caerIMU9EventSetTemp, 277
caerIMU9EventSetTimestamp, 277
caerIMU9EventValidate, 277
PACKED_STRUCT, 278

LIBCAER_HAVE_OPENCV
libcaer.h, 433
LIBCAER_HAVE_SERIALDEV
libcaer.h, 433
LIBCAER_NAME_STRING
libcaer.h, 433
LIBCAER_VERSION_STRING
libcaer.h, 433
LIBCAER_VERSION
libcaer.h, 433
libcaer.h, 430
CLEAR_NUMBITS16, 431
CLEAR_NUMBITS32, 431
CLEAR_NUMBITS8, 431
caer_error_codes, 435
caerByteArrayToInteger, 435
caerIntegerToByteArray, 436
caerStrEquals, 436
caerStrEqualsUpTo, 436
GET_NUMBITS16, 431
GET_NUMBITS32, 432
GET_NUMBITS8, 432
I16T, 432
I32T, 432
I64T, 432
I8T, 432
LIBCAER_HAVE_OPENCV, 433
LIBCAER_HAVE_SERIALDEV, 433
LIBCAER_NAME_STRING, 433
LIBCAER_VERSION_STRING, 433
LIBCAER_VERSION, 433
MASK_NUMBITS32, 433
MASK_NUMBITS64, 433
SET_NUMBITS16, 434
SET_NUMBITS32, 434
SET_NUMBITS8, 434
SWAP_VAR, 434
U16T, 434
U32T, 434
U64T, 435
U8T, 435
log.h, 437
caer_log_level, 438
caerLog, 438
caerLogDisable, 438
caerLogDisabled, 439
caerLogFileDescriptorsGetFirst, 439
caerLogFileDescriptorsGetSecond, 439
caerLogFileDescriptorsSet, 439
caerLogLevelGet, 440
caerLogLevelSet, 440
caerLogVAFull, 441
caerLogVA, 440

MASK_NUMBITS32
libcaer.h, 433
MASK_NUMBITS64
libcaer.h, 433
MATRIX4x4_SCALE_MASK
matrix4x4.h, 284
MATRIX4x4_SCALE_SHIFT
matrix4x4.h, 284
MATRIX4x4_TYPE_MASK
matrix4x4.h, 284
MATRIX4x4_TYPE_SHIFT
matrix4x4.h, 284
matrix4x4.h
CAER_MATRIX4x4_CONST_ITERATOR_ALL_↩ START, 280
CAER_MATRIX4x4_CONST_ITERATOR_VALI↩ D_START, 280
CAER_MATRIX4x4_CONST_REVERSE_ITER↩ ATOR_ALL_START, 281
CAER_MATRIX4x4_CONST_REVERSE_ITER↩ ATOR_VALID_START, 281
CAER_MATRIX4x4_ITERATOR_ALL_END, 282
CAER_MATRIX4x4_ITERATOR_ALL_START, 282
CAER_MATRIX4x4_ITERATOR_VALID_END, 282
CAER_MATRIX4x4_ITERATOR_VALID_START, 282
CAER_MATRIX4x4_REVERSE_ITERATOR_AL↩ L_END, 283
CAER_MATRIX4x4_REVERSE_ITERATOR_AL↩ L_START, 283

CAER_MATRIX4x4_REVERSE_ITERATOR_V↩
ALID_END, 283
CAER_MATRIX4x4_REVERSE_ITERATOR_V↩
ALID_START, 284
caerMatrix4x4Event, 285
caerMatrix4x4EventGetM00, 285
caerMatrix4x4EventGetM01, 285
caerMatrix4x4EventGetM02, 286
caerMatrix4x4EventGetM03, 286
caerMatrix4x4EventGetM10, 287
caerMatrix4x4EventGetM11, 287
caerMatrix4x4EventGetM12, 287
caerMatrix4x4EventGetM13, 288
caerMatrix4x4EventGetM20, 288
caerMatrix4x4EventGetM21, 288
caerMatrix4x4EventGetM22, 289
caerMatrix4x4EventGetM23, 289
caerMatrix4x4EventGetM30, 289
caerMatrix4x4EventGetM31, 291
caerMatrix4x4EventGetM32, 291
caerMatrix4x4EventGetM33, 291
caerMatrix4x4EventGetScale, 293
caerMatrix4x4EventGetTimestamp, 293
caerMatrix4x4EventGetTimestamp64, 293
caerMatrix4x4EventGetType, 294
caerMatrix4x4EventInvalidate, 294
caerMatrix4x4EventIsValid, 295
caerMatrix4x4EventPacket, 285
caerMatrix4x4EventPacketAllocate, 295
caerMatrix4x4EventPacketFromPacketHeader,
295
caerMatrix4x4EventPacketFromPacketHeader↩
Const, 296
caerMatrix4x4EventPacketGetEvent, 296
caerMatrix4x4EventPacketGetEventConst, 296
caerMatrix4x4EventSetM00, 297
caerMatrix4x4EventSetM01, 297
caerMatrix4x4EventSetM02, 297
caerMatrix4x4EventSetM03, 298
caerMatrix4x4EventSetM10, 298
caerMatrix4x4EventSetM11, 298
caerMatrix4x4EventSetM12, 299
caerMatrix4x4EventSetM13, 299
caerMatrix4x4EventSetM20, 299
caerMatrix4x4EventSetM21, 300
caerMatrix4x4EventSetM22, 300
caerMatrix4x4EventSetM23, 300
caerMatrix4x4EventSetM30, 301
caerMatrix4x4EventSetM31, 301
caerMatrix4x4EventSetM32, 301
caerMatrix4x4EventSetM33, 301
caerMatrix4x4EventSetScale, 302
caerMatrix4x4EventSetTimestamp, 302
caerMatrix4x4EventSetType, 302
caerMatrix4x4EventValidate, 303
MATRIX4x4_SCALE_MASK, 284
MATRIX4x4_SCALE_SHIFT, 284
MATRIX4x4_TYPE_MASK, 284
MATRIX4x4_TYPE_SHIFT, 284
PACKED_STRUCT, 303

network.h, 441

PACKED_STRUCT
common.h, 187
config.h, 199
ear.h, 213
frame.h, 242, 243
imu6.h, 259
imu9.h, 278
matrix4x4.h, 303
packetContainer.h, 312
point1d.h, 325
point2d.h, 338
point3d.h, 352
point4d.h, 368
polarity.h, 380
sample.h, 392
special.h, 407
spike.h, 421
POINT1D_SCALE_MASK
point1d.h, 317
POINT1D_SCALE_SHIFT
point1d.h, 317
POINT1D_TYPE_MASK
point1d.h, 318
POINT1D_TYPE_SHIFT
point1d.h, 318
POINT2D_SCALE_MASK
point2d.h, 330
POINT2D_SCALE_SHIFT
point2d.h, 330
POINT2D_TYPE_MASK
point2d.h, 330
POINT2D_TYPE_SHIFT
point2d.h, 330
POINT3D_SCALE_MASK
point3d.h, 343
POINT3D_SCALE_SHIFT
point3d.h, 344
POINT3D_TYPE_MASK
point3d.h, 344
POINT3D_TYPE_SHIFT
point3d.h, 344
POINT4D_SCALE_MASK
point4d.h, 357
POINT4D_SCALE_SHIFT
point4d.h, 358
POINT4D_TYPE_MASK
point4d.h, 358
POINT4D_TYPE_SHIFT
point4d.h, 358
POLARITY_MASK
polarity.h, 373
POLARITY_SHIFT
polarity.h, 373
POLARITY_X_ADDR_MASK

polarity.h, 373
POLARITY_X_ADDR_SHIFT
polarity.h, 373
POLARITY_Y_ADDR_MASK
polarity.h, 373
POLARITY_Y_ADDR_SHIFT
polarity.h, 374
packetContainer.h
CAER_EVENT_PACKET_CONTAINER_CONS←
T_ITERATOR_START, 305
CAER_EVENT_PACKET_CONTAINER_ITERA←
TOR_END, 305
CAER_EVENT_PACKET_CONTAINER_ITERA←
TOR_START, 306
caerEventPacketContainer, 306
caerEventPacketContainerAllocate, 306
caerEventPacketContainerCopyAllEvents, 307
caerEventPacketContainerCopyValidEvents, 307
caerEventPacketContainerFindEventPacketBy←
Type, 307
caerEventPacketContainerFindEventPacketBy←
TypeConst, 308
caerEventPacketContainerFree, 308
caerEventPacketContainerGetEventPacket, 309
caerEventPacketContainerGetEventPacketConst,
309
caerEventPacketContainerGetEventPackets←
Number, 309
caerEventPacketContainerGetEventsNumber, 310
caerEventPacketContainerGetEventsValidNumber,
310
caerEventPacketContainerGetHighestEvent←
Timestamp, 310
caerEventPacketContainerGetLowestEvent←
Timestamp, 311
caerEventPacketContainerSetEventPacket, 311
caerEventPacketContainerSetEventPackets←
Number, 312
caerEventPacketContainerUpdateStatistics, 312
PACKED_STRUCT, 312
point1d.h
CAER_POINT1D_CONST_ITERATOR_ALL_ST←
ART, 314
CAER_POINT1D_CONST_ITERATOR_VALID_←
START, 314
CAER_POINT1D_CONST_REVERSE_ITERAT←
OR_ALL_START, 314
CAER_POINT1D_CONST_REVERSE_ITERAT←
OR_VALID_START, 315
CAER_POINT1D_ITERATOR_ALL_END, 315
CAER_POINT1D_ITERATOR_ALL_START, 315
CAER_POINT1D_ITERATOR_VALID_END, 316
CAER_POINT1D_ITERATOR_VALID_START,
316
CAER_POINT1D_REVERSE_ITERATOR_ALL_←
END, 316
CAER_POINT1D_REVERSE_ITERATOR_ALL_←
START, 316
CAER_POINT1D_REVERSE_ITERATOR_VALI←
D_END, 317
CAER_POINT1D_REVERSE_ITERATOR_VALI←
D_START, 317
caerPoint1DEvent, 318
caerPoint1DEventGetScale, 318
caerPoint1DEventGetTimestamp, 319
caerPoint1DEventGetTimestamp64, 319
caerPoint1DEventGetType, 319
caerPoint1DEventGetX, 320
caerPoint1DEventInvalidate, 320
caerPoint1DEventIsValid, 321
caerPoint1DEventPacket, 318
caerPoint1DEventPacketAllocate, 321
caerPoint1DEventPacketFromPacketHeader, 321
caerPoint1DEventPacketFromPacketHeaderConst,
322
caerPoint1DEventPacketGetEvent, 322
caerPoint1DEventPacketGetEventConst, 322
caerPoint1DEventSetScale, 323
caerPoint1DEventSetTimestamp, 323
caerPoint1DEventSetType, 324
caerPoint1DEventSetX, 324
caerPoint1DEventValidate, 324
PACKED_STRUCT, 325
POINT1D_SCALE_MASK, 317
POINT1D_SCALE_SHIFT, 317
POINT1D_TYPE_MASK, 318
POINT1D_TYPE_SHIFT, 318
point2d.h
CAER_POINT2D_CONST_ITERATOR_ALL_ST←
ART, 326
CAER_POINT2D_CONST_ITERATOR_VALID_←
START, 327
CAER_POINT2D_CONST_REVERSE_ITERAT←
OR_ALL_START, 327
CAER_POINT2D_CONST_REVERSE_ITERAT←
OR_VALID_START, 327
CAER_POINT2D_ITERATOR_ALL_END, 328
CAER_POINT2D_ITERATOR_ALL_START, 328
CAER_POINT2D_ITERATOR_VALID_END, 328
CAER_POINT2D_ITERATOR_VALID_START,
328
CAER_POINT2D_REVERSE_ITERATOR_ALL_←
END, 329
CAER_POINT2D_REVERSE_ITERATOR_ALL_←
START, 329
CAER_POINT2D_REVERSE_ITERATOR_VALI←
D_END, 329
CAER_POINT2D_REVERSE_ITERATOR_VALI←
D_START, 329
caerPoint2DEvent, 331
caerPoint2DEventGetScale, 331
caerPoint2DEventGetTimestamp, 331
caerPoint2DEventGetTimestamp64, 332
caerPoint2DEventGetType, 332
caerPoint2DEventGetX, 332
caerPoint2DEventGetY, 333

caerPoint2DEventInvalidate, 333
caerPoint2DEventIsValid, 334
caerPoint2DEventPacket, 331
caerPoint2DEventPacketAllocate, 334
caerPoint2DEventPacketFromPacketHeader, 334
caerPoint2DEventPacketFromPacketHeaderConst, 335
caerPoint2DEventPacketGetEvent, 335
caerPoint2DEventPacketGetEventConst, 335
caerPoint2DEventSetScale, 336
caerPoint2DEventSetTimestamp, 336
caerPoint2DEventSetType, 337
caerPoint2DEventSetX, 337
caerPoint2DEventSetY, 337
caerPoint2DEventValidate, 337
PACKED_STRUCT, 338
POINT2D_SCALE_MASK, 330
POINT2D_SCALE_SHIFT, 330
POINT2D_TYPE_MASK, 330
POINT2D_TYPE_SHIFT, 330
point3d.h
CAER_POINT3D_CONST_ITERATOR_ALL_START, 340
CAER_POINT3D_CONST_ITERATOR_VALID_START, 340
CAER_POINT3D_CONST_REVERSE_ITERATOR_ALL_START, 341
CAER_POINT3D_CONST_REVERSE_ITERATOR_VALID_START, 341
CAER_POINT3D_ITERATOR_ALL_END, 341
CAER_POINT3D_ITERATOR_ALL_START, 342
CAER_POINT3D_ITERATOR_VALID_END, 342
CAER_POINT3D_ITERATOR_VALID_START, 342
CAER_POINT3D_REVERSE_ITERATOR_ALL_END, 342
CAER_POINT3D_REVERSE_ITERATOR_ALL_START, 343
CAER_POINT3D_REVERSE_ITERATOR_VALID_END, 343
CAER_POINT3D_REVERSE_ITERATOR_VALID_START, 343
caerPoint3DEvent, 344
caerPoint3DEventGetScale, 345
caerPoint3DEventGetTimestamp, 345
caerPoint3DEventGetTimestamp64, 345
caerPoint3DEventGetType, 346
caerPoint3DEventGetX, 346
caerPoint3DEventGetY, 346
caerPoint3DEventGetZ, 347
caerPoint3DEventInvalidate, 347
caerPoint3DEventIsValid, 348
caerPoint3DEventPacket, 344
caerPoint3DEventPacketAllocate, 348
caerPoint3DEventPacketFromPacketHeader, 348
caerPoint3DEventPacketFromPacketHeaderConst, 349
caerPoint3DEventPacketGetEvent, 349
caerPoint3DEventPacketGetEventConst, 349
caerPoint3DEventSetScale, 350
caerPoint3DEventSetTimestamp, 350
caerPoint3DEventSetType, 351
caerPoint3DEventSetX, 351
caerPoint3DEventSetY, 351
caerPoint3DEventSetZ, 351
caerPoint3DEventValidate, 352
PACKED_STRUCT, 352
POINT3D_SCALE_MASK, 343
POINT3D_SCALE_SHIFT, 344
POINT3D_TYPE_MASK, 344
POINT3D_TYPE_SHIFT, 344
point4d.h
CAER_POINT4D_CONST_ITERATOR_ALL_START, 354
CAER_POINT4D_CONST_ITERATOR_VALID_START, 354
CAER_POINT4D_CONST_REVERSE_ITERATOR_ALL_START, 355
CAER_POINT4D_CONST_REVERSE_ITERATOR_VALID_START, 355
CAER_POINT4D_ITERATOR_ALL_END, 355
CAER_POINT4D_ITERATOR_ALL_START, 356
CAER_POINT4D_ITERATOR_VALID_END, 356
CAER_POINT4D_ITERATOR_VALID_START, 356
CAER_POINT4D_REVERSE_ITERATOR_ALL_END, 356
CAER_POINT4D_REVERSE_ITERATOR_ALL_START, 357
CAER_POINT4D_REVERSE_ITERATOR_VALID_END, 357
CAER_POINT4D_REVERSE_ITERATOR_VALID_START, 357
caerPoint4DEvent, 358
caerPoint4DEventGetScale, 359
caerPoint4DEventGetTimestamp, 359
caerPoint4DEventGetTimestamp64, 359
caerPoint4DEventGetType, 360
caerPoint4DEventGetW, 360
caerPoint4DEventGetX, 360
caerPoint4DEventGetY, 361
caerPoint4DEventGetZ, 361
caerPoint4DEventInvalidate, 362
caerPoint4DEventIsValid, 362
caerPoint4DEventPacket, 358
caerPoint4DEventPacketAllocate, 362
caerPoint4DEventPacketFromPacketHeader, 363
caerPoint4DEventPacketFromPacketHeaderConst, 363
caerPoint4DEventPacketGetEvent, 363
caerPoint4DEventPacketGetEventConst, 364
caerPoint4DEventSetScale, 364
caerPoint4DEventSetTimestamp, 364
caerPoint4DEventSetType, 365
caerPoint4DEventSetW, 365
caerPoint4DEventSetX, 365

caerPoint4DEventSetY, 366
caerPoint4DEventSetZ, 366
caerPoint4DEventValidate, 366
PACKED_STRUCT, 368
POINT4D_SCALE_MASK, 357
POINT4D_SCALE_SHIFT, 358
POINT4D_TYPE_MASK, 358
POINT4D_TYPE_SHIFT, 358
polarity.h
CAER_POLARITY_CONST_ITERATOR_ALL_S↩
TART, 369
CAER_POLARITY_CONST_ITERATOR_VALID↩
_START, 370
CAER_POLARITY_CONST_REVERSE_ITERA↩
TOR_ALL_START, 370
CAER_POLARITY_CONST_REVERSE_ITERA↩
TOR_VALID_START, 370
CAER_POLARITY_ITERATOR_ALL_END, 371
CAER_POLARITY_ITERATOR_ALL_START, 371
CAER_POLARITY_ITERATOR_VALID_END, 371
CAER_POLARITY_ITERATOR_VALID_START,
371
CAER_POLARITY_REVERSE_ITERATOR_AL↩
L_END, 372
CAER_POLARITY_REVERSE_ITERATOR_AL↩
L_START, 372
CAER_POLARITY_REVERSE_ITERATOR_VA↩
LID_END, 372
CAER_POLARITY_REVERSE_ITERATOR_VA↩
LID_START, 372
caerPolarityEvent, 374
caerPolarityEventGetPolarity, 374
caerPolarityEventGetTimestamp, 375
caerPolarityEventGetTimestamp64, 375
caerPolarityEventGetX, 375
caerPolarityEventGetY, 376
caerPolarityEventInvalidate, 376
caerPolarityEventIsValid, 376
caerPolarityEventPacket, 374
caerPolarityEventPacketAllocate, 377
caerPolarityEventPacketFromPacketHeader, 377
caerPolarityEventPacketFromPacketHeaderConst,
378
caerPolarityEventPacketGetEvent, 378
caerPolarityEventPacketGetEventConst, 378
caerPolarityEventSetPolarity, 379
caerPolarityEventSetTimestamp, 379
caerPolarityEventSetX, 379
caerPolarityEventSetY, 380
caerPolarityEventValidate, 380
PACKED_STRUCT, 380
POLARITY_MASK, 373
POLARITY_SHIFT, 373
POLARITY_X_ADDR_MASK, 373
POLARITY_X_ADDR_SHIFT, 373
POLARITY_Y_ADDR_MASK, 373
POLARITY_Y_ADDR_SHIFT, 374
portable_endian.h, 442
SAMPLE_MASK
sample.h, 385
SAMPLE_SHIFT
sample.h, 386
SAMPLE_TYPE_MASK
sample.h, 386
SAMPLE_TYPE_SHIFT
sample.h, 386
SET_NUMBITS16
libcaer.h, 434
SET_NUMBITS32
libcaer.h, 434
SET_NUMBITS8
libcaer.h, 434
SPECIAL_DATA_MASK
special.h, 398
SPECIAL_DATA_SHIFT
special.h, 398
SPECIAL_TYPE_MASK
special.h, 398
SPECIAL_TYPE_SHIFT
special.h, 398
SPIKE_CHIP_ID_MASK
spike.h, 412
SPIKE_CHIP_ID_SHIFT
spike.h, 413
SPIKE_NEURON_ID_MASK
spike.h, 413
SPIKE_NEURON_ID_SHIFT
spike.h, 413
SPIKE_SOURCE_CORE_ID_MASK
spike.h, 413
SPIKE_SOURCE_CORE_ID_SHIFT
spike.h, 413
SWAP_VAR
libcaer.h, 434
sample.h
CAER_SAMPLE_CONST_ITERATOR_ALL_ST↩
ART, 382
CAER_SAMPLE_CONST_ITERATOR_VALID_↩
START, 382
CAER_SAMPLE_CONST_REVERSE_ITERAT↩
OR_ALL_START, 383
CAER_SAMPLE_CONST_REVERSE_ITERAT↩
OR_VALID_START, 383
CAER_SAMPLE_ITERATOR_ALL_END, 383
CAER_SAMPLE_ITERATOR_ALL_START, 384
CAER_SAMPLE_ITERATOR_VALID_END, 384
CAER_SAMPLE_ITERATOR_VALID_START, 384
CAER_SAMPLE_REVERSE_ITERATOR_ALL_↩
END, 384
CAER_SAMPLE_REVERSE_ITERATOR_ALL_↩
START, 385
CAER_SAMPLE_REVERSE_ITERATOR_VALI↩
D_END, 385
CAER_SAMPLE_REVERSE_ITERATOR_VALI↩
D_START, 385
caerSampleEvent, 386

caerSampleEventGetSample, 387
caerSampleEventGetTimestamp, 387
caerSampleEventGetTimestamp64, 387
caerSampleEventGetType, 388
caerSampleEventInvalidate, 388
caerSampleEventIsValid, 388
caerSampleEventPacket, 386
caerSampleEventPacketAllocate, 389
caerSampleEventPacketFromPacketHeader, 389
caerSampleEventPacketFromPacketHeaderConst, 390
caerSampleEventPacketGetEvent, 390
caerSampleEventPacketGetEventConst, 390
caerSampleEventSetSample, 391
caerSampleEventSetTimestamp, 391
caerSampleEventSetType, 391
caerSampleEventValidate, 392
PACKED_STRUCT, 392
SAMPLE_MASK, 385
SAMPLE_SHIFT, 386
SAMPLE_TYPE_MASK, 386
SAMPLE_TYPE_SHIFT, 386

serial.h
CAER_HOST_CONFIG_SERIAL_BAUD_RATE↩_12M, 166
CAER_HOST_CONFIG_SERIAL_BAUD_RATE↩_2M, 166
CAER_HOST_CONFIG_SERIAL_BAUD_RATE↩_4M, 167
CAER_HOST_CONFIG_SERIAL_BAUD_RATE↩_8M, 167
CAER_HOST_CONFIG_SERIAL_READ_SIZE, 167
CAER_HOST_CONFIG_SERIAL, 166
caerDeviceOpenSerial, 167

special.h
CAER_SPECIAL_CONST_ITERATOR_ALL_ST↩ART, 394
CAER_SPECIAL_CONST_ITERATOR_VALID_↩START, 395
CAER_SPECIAL_CONST_REVERSE_ITERAT↩OR_ALL_START, 395
CAER_SPECIAL_CONST_REVERSE_ITERAT↩OR_VALID_START, 395
CAER_SPECIAL_ITERATOR_ALL_END, 396
CAER_SPECIAL_ITERATOR_ALL_START, 396
CAER_SPECIAL_ITERATOR_VALID_END, 396
CAER_SPECIAL_ITERATOR_VALID_START, 396
CAER_SPECIAL_REVERSE_ITERATOR_ALL_↩END, 397
CAER_SPECIAL_REVERSE_ITERATOR_ALL_↩START, 397
CAER_SPECIAL_REVERSE_ITERATOR_VALI↩D_END, 397
CAER_SPECIAL_REVERSE_ITERATOR_VALI↩D_START, 397
caer_special_event_types, 399
caerSpecialEvent, 399
caerSpecialEventGetData, 400
caerSpecialEventGetTimestamp, 400
caerSpecialEventGetTimestamp64, 401
caerSpecialEventGetType, 401
caerSpecialEventInvalidate, 401
caerSpecialEventIsValid, 402
caerSpecialEventPacket, 399
caerSpecialEventPacketAllocate, 402
caerSpecialEventPacketFindEventByType, 403
caerSpecialEventPacketFindEventByTypeConst, 403
caerSpecialEventPacketFindValidEventByType, 403
caerSpecialEventPacketFindValidEventByType↩Const, 404
caerSpecialEventPacketFromPacketHeader, 404
caerSpecialEventPacketFromPacketHeaderConst, 405
caerSpecialEventPacketGetEvent, 405
caerSpecialEventPacketGetEventConst, 405
caerSpecialEventSetData, 406
caerSpecialEventSetTimestamp, 406
caerSpecialEventSetType, 406
caerSpecialEventValidate, 407
PACKED_STRUCT, 407
SPECIAL_DATA_MASK, 398
SPECIAL_DATA_SHIFT, 398
SPECIAL_TYPE_MASK, 398
SPECIAL_TYPE_SHIFT, 398

spike.h
CAER_SPIKE_CONST_ITERATOR_ALL_START, 409
CAER_SPIKE_CONST_ITERATOR_VALID_ST↩ART, 409
CAER_SPIKE_CONST_REVERSE_ITERATOR↩_ALL_START, 410
CAER_SPIKE_CONST_REVERSE_ITERATOR↩_VALID_START, 410
CAER_SPIKE_ITERATOR_ALL_END, 410
CAER_SPIKE_ITERATOR_ALL_START, 411
CAER_SPIKE_ITERATOR_VALID_END, 411
CAER_SPIKE_ITERATOR_VALID_START, 411
CAER_SPIKE_REVERSE_ITERATOR_ALL_END, 411
CAER_SPIKE_REVERSE_ITERATOR_ALL_ST↩ART, 412
CAER_SPIKE_REVERSE_ITERATOR_VALID_↩END, 412
CAER_SPIKE_REVERSE_ITERATOR_VALID_↩START, 412
caerSpikeEvent, 413
caerSpikeEventGetChipID, 414
caerSpikeEventGetNeuronID, 414
caerSpikeEventGetSourceCoreID, 415
caerSpikeEventGetTimestamp, 415
caerSpikeEventGetTimestamp64, 415
caerSpikeEventInvalidate, 416

caerSpikeEventIsValid, 416
caerSpikeEventPacket, 414
caerSpikeEventPacketAllocate, 416
caerSpikeEventPacketFromPacketHeader, 417
caerSpikeEventPacketFromPacketHeaderConst, 417
caerSpikeEventPacketGetEvent, 418
caerSpikeEventPacketGetEventConst, 418
caerSpikeEventSetChipID, 418
caerSpikeEventSetNeuronID, 420
caerSpikeEventSetSourceCoreID, 420
caerSpikeEventSetTimestamp, 420
caerSpikeEventValidate, 421
PACKED_STRUCT, 421
SPIKE_CHIP_ID_MASK, 412
SPIKE_CHIP_ID_SHIFT, 413
SPIKE_NEURON_ID_MASK, 413
SPIKE_NEURON_ID_SHIFT, 413
SPIKE_SOURCE_CORE_ID_MASK, 413
SPIKE_SOURCE_CORE_ID_SHIFT, 413

TS_OVERFLOW_SHIFT
common.h, 172

U16T
libcaer.h, 434
U32T
libcaer.h, 434
U64T
libcaer.h, 435
U8T
libcaer.h, 435
usb.h
CAER_HOST_CONFIG_USB_BUFFER_NUMB↩ER, 168
CAER_HOST_CONFIG_USB_BUFFER_SIZE, 168
CAER_HOST_CONFIG_USB, 168
caerDeviceOpen, 169

VALID_MARK_MASK
common.h, 173
VALID_MARK_SHIFT
common.h, 173