

nmpp

Создано системой Doxygen 1.8.13



# Оглавление

1	Введение	1
1.1	Introduction . . . . .	1
1.2	Installation . . . . .	1
1.2.1	Step 1: Opening the box . . . . .	1
2	Ошибки	3
3	Алфавитный указатель групп	5
3.1	Группы . . . . .	5
4	Иерархический список классов	11
4.1	Иерархия классов . . . . .	11
5	Алфавитный указатель классов	13
5.1	Классы . . . . .	13
6	Список файлов	15
6.1	Файлы . . . . .	15

7	Группы	19
7.1	DFT-8	19
7.1.1	Подробное описание	19
7.1.2	Функции	19
7.1.2.1	nmppsDFT8Fwd_32fcr()	19
7.2	FFT-16	21
7.2.1	Подробное описание	21
7.2.2	Функции	21
7.2.2.1	nmppsFFT16Fwd_32fcr()	21
7.2.2.2	nmppsFFT16FwdInitAlloc_32fcr()	21
7.3	FFT-32	23
7.3.1	Подробное описание	23
7.3.2	Функции	23
7.3.2.1	nmppsFFT32Fwd_32fcr()	23
7.3.2.2	nmppsFFT32FwdInitAlloc_32fcr()	23
7.4	FFT-64	25
7.4.1	Подробное описание	25
7.4.2	Функции	25
7.4.2.1	nmppsFFT64Fwd_32fcr()	25
7.4.2.2	nmppsFFT64FwdInitAlloc_32fcr()	25
7.5	FFT-128	27
7.5.1	Подробное описание	27
7.5.2	Функции	27
7.5.2.1	nmppsFFT128Fwd_32fcr()	27
7.5.2.2	nmppsFFT128FwdInitAlloc_32fcr()	27
7.6	FFT-256	29
7.6.1	Подробное описание	29
7.6.2	Функции	29
7.6.2.1	nmppsFFT256Fwd_32fcr()	29
7.6.2.2	nmppsFFT256FwdInitAlloc_32fcr()	29

7.7	FFT-512 . . . . .	31
7.7.1	Подробное описание . . . . .	31
7.7.2	Функции . . . . .	31
7.7.2.1	nmppsFFT512Fwd_32fcr() . . . . .	31
7.7.2.2	nmppsFFT512FwdInitAlloc_32fcr() . . . . .	31
7.8	FFT-1024 . . . . .	33
7.8.1	Подробное описание . . . . .	33
7.8.2	Функции . . . . .	33
7.8.2.1	nmppsFFT1024Fwd_32fcr() . . . . .	33
7.8.2.2	nmppsFFT1024FwdInitAlloc_32fcr() . . . . .	33
7.9	FFT-2048 . . . . .	35
7.9.1	Подробное описание . . . . .	35
7.9.2	Функции . . . . .	35
7.9.2.1	nmppsFFT2048Fwd_32fcr() . . . . .	35
7.9.2.2	nmppsFFT2048FwdInitAlloc_32fcr() . . . . .	35
7.10	FFT-4096 . . . . .	37
7.10.1	Подробное описание . . . . .	37
7.10.2	Функции . . . . .	37
7.10.2.1	nmppsFFT4096Fwd_32fcr() . . . . .	37
7.10.2.2	nmppsFFT4096FwdInitAlloc_32fcr() . . . . .	37
7.11	IDFT-8 . . . . .	39
7.11.1	Подробное описание . . . . .	39
7.11.2	Функции . . . . .	39
7.11.2.1	nmppsDFT8Inv_32fcr() . . . . .	39
7.12	IFFT-16 . . . . .	40
7.12.1	Подробное описание . . . . .	40
7.12.2	Функции . . . . .	40
7.12.2.1	nmppsFFT16Inv_32fcr() . . . . .	40
7.12.2.2	nmppsFFT16InvInitAlloc_32fcr() . . . . .	40
7.13	IFFT-32 . . . . .	42

7.13.1	Подробное описание . . . . .	42
7.13.2	Функции . . . . .	42
7.13.2.1	nmppsFFT32Inv_32fcr() . . . . .	42
7.13.2.2	nmppsFFT32InvInitAlloc_32fcr() . . . . .	42
7.14	IFFT-64 . . . . .	44
7.14.1	Подробное описание . . . . .	44
7.14.2	Функции . . . . .	44
7.14.2.1	nmppsFFT64Inv_32fcr() . . . . .	44
7.14.2.2	nmppsFFT64InvInitAlloc_32fcr() . . . . .	44
7.15	IFFT-128 . . . . .	46
7.15.1	Подробное описание . . . . .	46
7.15.2	Функции . . . . .	46
7.15.2.1	nmppsFFT128Inv_32fcr() . . . . .	46
7.15.2.2	nmppsFFT128InvInitAlloc_32fcr() . . . . .	46
7.16	IFFT-256 . . . . .	48
7.16.1	Подробное описание . . . . .	48
7.16.2	Функции . . . . .	48
7.16.2.1	nmppsFFT256Inv_32fcr() . . . . .	48
7.16.2.2	nmppsFFT256InvInitAlloc_32fcr() . . . . .	48
7.17	IFFT-512 . . . . .	50
7.17.1	Подробное описание . . . . .	50
7.17.2	Функции . . . . .	50
7.17.2.1	nmppsFFT512Inv_32fcr() . . . . .	50
7.17.2.2	nmppsFFT512InvInitAlloc_32fcr() . . . . .	50
7.18	IFFT-1024 . . . . .	52
7.18.1	Подробное описание . . . . .	52
7.18.2	Функции . . . . .	52
7.18.2.1	nmppsFFT1024Inv_32fcr() . . . . .	52
7.18.2.2	nmppsFFT1024InvInitAlloc_32fcr() . . . . .	52
7.19	IFFT-2048 . . . . .	54

7.19.1	Подробное описание . . . . .	54
7.19.2	Функции . . . . .	54
7.19.2.1	nmppsFFT2048Inv_32fcr() . . . . .	54
7.19.2.2	nmppsFFT2048InvInitAlloc_32fcr() . . . . .	54
7.20	IFFT-4096 . . . . .	56
7.20.1	Подробное описание . . . . .	56
7.20.2	Функции . . . . .	56
7.20.2.1	nmppsFFT4096Inv_32fcr() . . . . .	56
7.20.2.2	nmppsFFT4096InvInitAlloc_32fcr() . . . . .	56
7.21	FFT-Common . . . . .	58
7.21.1	Подробное описание . . . . .	58
7.21.2	Функции . . . . .	58
7.21.2.1	nmppsFFTFwd_32fcr() . . . . .	58
7.21.2.2	nmppsFFTFwdInitAlloc_32fcr() . . . . .	58
7.22	IFFT-Common . . . . .	61
7.22.1	Подробное описание . . . . .	61
7.22.2	Функции . . . . .	61
7.22.2.1	nmppsFFTInv_32fcr() . . . . .	61
7.22.2.2	nmppsFFTInvInitAlloc_32fcr() . . . . .	61
7.23	nmppsMalloc . . . . .	64
7.23.1	Подробное описание . . . . .	64
7.24	nmppsFree . . . . .	65
7.24.1	Подробное описание . . . . .	65
7.25	BLASS-LEVEL1 . . . . .	66
7.25.1	Подробное описание . . . . .	66
7.25.2	Перечисления . . . . .	66
7.25.2.1	nm__trans . . . . .	66
7.25.3	Функции . . . . .	66
7.25.3.1	nmblas_dasum() . . . . .	67
7.26	nmppcDivC . . . . .	68

7.26.1	Подробное описание . . . . .	68
7.27	nmppcProdC . . . . .	69
7.27.1	Подробное описание . . . . .	69
7.28	nmppcFixExp32 . . . . .	70
7.28.1	Подробное описание . . . . .	70
7.29	nmppcFixSinCos32 . . . . .	71
7.29.1	Подробное описание . . . . .	71
7.30	nmppcFixArcTan32 . . . . .	72
7.30.1	Подробное описание . . . . .	72
7.31	nmppcDoubleToFix32 . . . . .	73
7.31.1	Подробное описание . . . . .	73
7.32	nmppcFix32ToDouble . . . . .	74
7.32.1	Подробное описание . . . . .	74
7.33	nmppcFixSqrt32 . . . . .	75
7.33.1	Подробное описание . . . . .	75
7.34	nmppcFixMul32 . . . . .	76
7.34.1	Подробное описание . . . . .	76
7.35	nmppcFixInv32 . . . . .	77
7.35.1	Подробное описание . . . . .	77
7.36	nmppcTblFixArcSin32 . . . . .	78
7.36.1	Подробное описание . . . . .	78
7.37	nmppcTblFixArcCos32 . . . . .	79
7.37.1	Подробное описание . . . . .	79
7.38	nmppcTblFixCos32 . . . . .	80
7.38.1	Подробное описание . . . . .	80
7.39	nmppcTblFixSin32 . . . . .	81
7.39.1	Подробное описание . . . . .	81
7.40	nmppcFixDivMod32 . . . . .	82
7.40.1	Подробное описание . . . . .	82
7.41	nmppcFixSqrt64 . . . . .	83



7.41.1	Подробное описание . . . . .	83
7.42	nmppcDoubleToFix64 . . . . .	84
7.42.1	Подробное описание . . . . .	84
7.43	nmppcFix64ToDouble . . . . .	85
7.43.1	Подробное описание . . . . .	85
7.44	nmppcFixDiv64 . . . . .	86
7.44.1	Подробное описание . . . . .	86
7.45	nmppcFixSinCos64 . . . . .	87
7.45.1	Подробное описание . . . . .	87
7.46	nmppcFixArcTan64 . . . . .	88
7.46.1	Подробное описание . . . . .	88
7.47	nmppcFix64Exp01 . . . . .	89
7.47.1	Подробное описание . . . . .	89
7.48	nmppsRand . . . . .	90
7.48.1	Подробное описание . . . . .	90
7.49	nmppcSqrt . . . . .	91
7.49.1	Подробное описание . . . . .	91
7.50	Инициализация . . . . .	92
7.50.1	Подробное описание . . . . .	92
7.51	Integer operations . . . . .	93
7.51.1	Подробное описание . . . . .	93
7.52	Fix-point 64 . . . . .	94
7.52.1	Подробное описание . . . . .	94
7.53	Fix-point 32 . . . . .	95
7.53.1	Подробное описание . . . . .	95
7.54	Арифметические операции . . . . .	96
7.54.1	Подробное описание . . . . .	96
7.55	Функции деинтерлейсинга . . . . .	97
7.55.1	Подробное описание . . . . .	97
7.55.2	Функции . . . . .	97

7.55.2.1	IMG_DeinterlaceBlend()	97
7.55.2.2	IMG_DeinterlaceSplit()	97
7.56	КИХ-фильтрация	99
7.56.1	Подробное описание	99
7.57	Floodfill	100
7.57.1	Подробное описание	100
7.57.2	Функции	100
7.57.2.1	FloodFill8()	101
7.58	IMG_Convert	106
7.58.1	Подробное описание	106
7.59	IMG_RGB32ToGray	107
7.59.1	Подробное описание	107
7.60	Переупорядочивание изображений	108
7.60.1	Подробное описание	108
7.61	Блочное переупорядочивание	109
7.61.1	Подробное описание	109
7.62	IMG_SplitIntoBlocks	110
7.62.1	Подробное описание	110
7.63	IMG_MergeFromBlocks	111
7.63.1	Подробное описание	111
7.64	IMG_Free	112
7.64.1	Подробное описание	112
7.65	IMG_Release	113
7.65.1	Подробное описание	113
7.66	Арифметические действия	114
7.67	Масочная фильтрация	115
7.67.1	Подробное описание	115
7.68	Инициализация и копирование	116
7.68.1	Подробное описание	116
7.69	Функции поддержки	117

7.69.1	Подробное описание . . . . .	117
7.70	Функции графического вывода текста . . . . .	118
7.70.1	Подробное описание . . . . .	118
7.70.2	Функции . . . . .	118
7.70.2.1	hex2ascii() [1/2] . . . . .	118
7.70.2.2	hex2ascii() [2/2] . . . . .	118
7.70.2.3	IMG_Print8x15() . . . . .	119
7.71	Инициализация и копирование . . . . .	120
7.71.1	Подробное описание . . . . .	120
7.72	nmppmCopyua . . . . .	121
7.72.1	Подробное описание . . . . .	121
7.73	MTR_Copyua . . . . .	122
7.73.1	Подробное описание . . . . .	122
7.74	MTR_Copy . . . . .	123
7.74.1	Подробное описание . . . . .	123
7.75	nmppmMul_mm . . . . .	124
7.75.1	Подробное описание . . . . .	124
7.76	nmppmMul_mv_ . . . . .	126
7.76.1	Подробное описание . . . . .	126
7.77	nmppmMul_mv__AddC . . . . .	127
7.77.1	Подробное описание . . . . .	127
7.78	MTR_ProdUnitV . . . . .	128
7.78.1	Подробное описание . . . . .	128
7.79	MTR_Malloc . . . . .	129
7.80	MTR_Free . . . . .	130
7.80.1	Подробное описание . . . . .	130
7.81	MTR_Addr . . . . .	131
7.81.1	Подробное описание . . . . .	131
7.82	MTR_SetVal . . . . .	132
7.82.1	Подробное описание . . . . .	132

7.83 MTR_GetVal . . . . .	133
7.83.1 Подробное описание . . . . .	133
7.84 Функции поддержки . . . . .	134
7.84.1 Подробное описание . . . . .	134
7.85 Векторно-матричные операции . . . . .	135
7.85.1 Подробное описание . . . . .	135
7.86 FFT-256 . . . . .	136
7.86.1 Подробное описание . . . . .	136
7.86.2 Функции . . . . .	136
7.86.2.1 FFT_Fwd256() . . . . .	136
7.87 IFFT-256 . . . . .	138
7.87.1 Подробное описание . . . . .	138
7.87.2 Функции . . . . .	138
7.87.2.1 FFT_Inv256() . . . . .	138
7.88 FFT-512 . . . . .	141
7.88.1 Подробное описание . . . . .	141
7.88.2 Функции . . . . .	141
7.88.2.1 FFT_Fwd512() . . . . .	141
7.89 IFFT-512 . . . . .	143
7.89.1 Подробное описание . . . . .	143
7.89.2 Функции . . . . .	143
7.89.2.1 FFT_Inv512() . . . . .	143
7.90 FFT-1024 . . . . .	146
7.90.1 Подробное описание . . . . .	146
7.90.2 Функции . . . . .	146
7.90.2.1 FFT_Fwd1024() . . . . .	146
7.91 IFFT-1024 . . . . .	148
7.91.1 Подробное описание . . . . .	148
7.91.2 Функции . . . . .	148
7.91.2.1 FFT_Inv1024() . . . . .	148

7.92 FFT-2048 . . . . .	151
7.92.1 Подробное описание . . . . .	151
7.92.2 Функции . . . . .	151
7.92.2.1 FFT_Fwd2048() . . . . .	151
7.93 IFFT-2048 . . . . .	153
7.93.1 Подробное описание . . . . .	153
7.93.2 Функции . . . . .	153
7.93.2.1 FFT_Inv2048() . . . . .	153
7.94 FFT-4096 . . . . .	156
7.94.1 Подробное описание . . . . .	156
7.94.2 Функции . . . . .	156
7.94.2.1 FFT_Fwd4096() . . . . .	156
7.95 IFFT-4096 . . . . .	158
7.95.1 Подробное описание . . . . .	158
7.95.2 Функции . . . . .	158
7.95.2.1 FFT_Inv4096() . . . . .	158
7.96 FFT-8192 . . . . .	160
7.96.1 Подробное описание . . . . .	160
7.96.2 Функции . . . . .	160
7.96.2.1 FFT_Fwd8192() . . . . .	160
7.97 IFFT-8192 . . . . .	162
7.97.1 Подробное описание . . . . .	162
7.97.2 Функции . . . . .	162
7.97.2.1 FFT_Inv8192() . . . . .	162
7.98 Свертка . . . . .	164
7.98.1 Подробное описание . . . . .	164
7.99 Масочная фильтрация . . . . .	165
7.99.1 Подробное описание . . . . .	165
7.100 Изменение размеров . . . . .	166
7.100.1 Подробное описание . . . . .	166

7.101 Быстрое преобразование Фурье . . . . .	167
7.102 SIG_XCorr . . . . .	168
7.102.1 Подробное описание . . . . .	168
7.103 SIG_Median3 . . . . .	170
7.103.1 Подробное описание . . . . .	170
7.104 КИХ-фильтрация . . . . .	171
7.104.1 Подробное описание . . . . .	171
7.105 nmppsFIR_Xs . . . . .	172
7.105.1 Подробное описание . . . . .	172
7.106 nmppsFIRInit_Xs . . . . .	173
7.106.1 Подробное описание . . . . .	173
7.107 nmppsFIRInitAlloc_Xs . . . . .	174
7.107.1 Подробное описание . . . . .	174
7.108 nmppsFIRGetStateSize_Xs . . . . .	175
7.108.1 Подробное описание . . . . .	175
7.109 nmppsFIRFree . . . . .	176
7.109.1 Подробное описание . . . . .	176
7.110 SIG_ResampleDown2 . . . . .	177
7.110.1 Подробное описание . . . . .	177
7.111 SIG_ResampleUp3Down2 . . . . .	178
7.111.1 Подробное описание . . . . .	178
7.112 SIG_CreateResample . . . . .	179
7.112.1 Подробное описание . . . . .	179
7.113 SIG_SetResample . . . . .	180
7.113.1 Подробное описание . . . . .	180
7.114 SIG_Resample_perf . . . . .	181
7.114.1 Подробное описание . . . . .	181
7.115 Типы векторных данных . . . . .	182
7.115.1 Подробное описание . . . . .	183
7.115.2 Типы . . . . .	184

7.115.2.1 nm1 . . . . .	184
7.115.2.2 nm16s . . . . .	184
7.115.2.3 nm16s15b . . . . .	185
7.115.2.4 nm16u . . . . .	185
7.115.2.5 nm16u15b . . . . .	185
7.115.2.6 nm2s . . . . .	186
7.115.2.7 nm2u . . . . .	186
7.115.2.8 nm32s . . . . .	186
7.115.2.9 nm32s30b . . . . .	187
7.115.2.10nm32s31b . . . . .	187
7.115.2.11nm32u . . . . .	187
7.115.2.12nm32u31b . . . . .	188
7.115.2.13nm4s . . . . .	188
7.115.2.14nm4u . . . . .	188
7.115.2.15nm4u3b . . . . .	189
7.115.2.16nm64s . . . . .	189
7.115.2.17nm64s63b . . . . .	189
7.115.2.18nm64u . . . . .	190
7.115.2.19nm8s . . . . .	190
7.115.2.20nm8s7b . . . . .	190
7.115.2.21nm8u . . . . .	191
7.115.2.22nm8u7b . . . . .	191
7.115.2.23v16nm16s . . . . .	191
7.115.2.24v16nm16u . . . . .	191
7.115.2.25v16nm32s . . . . .	191
7.115.2.26v16nm32u . . . . .	192
7.115.2.27v16nm4b3u . . . . .	192
7.115.2.28v16nm4u . . . . .	192
7.115.2.29v16nm8s . . . . .	192
7.115.2.30v16nm8s7b . . . . .	192

7.115.2.31v16nm8u . . . . .	192
7.115.2.32v2nm32s . . . . .	193
7.115.2.33v2nm32u . . . . .	193
7.115.2.34v4nm16s . . . . .	193
7.115.2.35v4nm16u . . . . .	193
7.115.2.36v4nm32s . . . . .	193
7.115.2.37v4nm32u . . . . .	193
7.115.2.38v4nm8u . . . . .	193
7.115.2.39v8nm16s . . . . .	193
7.115.2.40v8nm16u . . . . .	194
7.115.2.41v8nm32s . . . . .	194
7.115.2.42v8nm32u . . . . .	194
7.115.2.43v8nm8s . . . . .	194
7.115.2.44v8nm8u . . . . .	194
7.116Типы скалярных данных . . . . .	195
7.116.1 Подробное описание . . . . .	195
7.116.2 Типы . . . . .	195
7.116.2.1 int15b . . . . .	196
7.116.2.2 int16b . . . . .	196
7.116.2.3 int1b . . . . .	196
7.116.2.4 int2b . . . . .	196
7.116.2.5 int30b . . . . .	197
7.116.2.6 int31b . . . . .	197
7.116.2.7 int32b . . . . .	197
7.116.2.8 int3b . . . . .	197
7.116.2.9 int4b . . . . .	198
7.116.2.10nt63b . . . . .	198
7.116.2.11nt64b . . . . .	198
7.116.2.12nt7b . . . . .	198
7.116.2.13nt8b . . . . .	199



7.116.2.14	uint15b . . . . .	199
7.116.2.15	uint16b . . . . .	199
7.116.2.16	uint1b . . . . .	199
7.116.2.17	uint2b . . . . .	200
7.116.2.18	uint31b . . . . .	200
7.116.2.19	uint32b . . . . .	200
7.116.2.20	uint3b . . . . .	200
7.116.2.21	uint4b . . . . .	201
7.116.2.22	uint63b . . . . .	201
7.116.2.23	uint64b . . . . .	201
7.116.2.24	uint7b . . . . .	201
7.116.2.25	uint8b . . . . .	201
7.117	Функции поддержки . . . . .	202
7.117.1	Подробное описание . . . . .	202
7.118	Инициализация и копирование . . . . .	203
7.118.1	Подробное описание . . . . .	203
7.119	Арифметические операции . . . . .	204
7.119.1	Подробное описание . . . . .	204
7.120	Логические и бинарные операции . . . . .	205
7.120.1	Подробное описание . . . . .	205
7.121	Операции сравнения . . . . .	206
7.121.1	Подробное описание . . . . .	206
7.122	Переупорядочивание и сортировка . . . . .	207
7.122.1	Подробное описание . . . . .	207
7.123	nmppsAbs . . . . .	208
7.123.1	Подробное описание . . . . .	208
7.124	nmppsAbs1 . . . . .	209
7.124.1	Подробное описание . . . . .	209
7.125	nmppsNeg . . . . .	210
7.125.1	Подробное описание . . . . .	210

7.126nmppsAddC . . . . .	211
7.126.1 Подробное описание . . . . .	211
7.127nmppsAdd . . . . .	212
7.127.1 Подробное описание . . . . .	212
7.128nmppsAdd_AddC . . . . .	213
7.128.1 Подробное описание . . . . .	213
7.129nmppsSubC . . . . .	214
7.129.1 Подробное описание . . . . .	214
7.130nmppsSubCRev . . . . .	215
7.130.1 Подробное описание . . . . .	215
7.131nmppsSub . . . . .	216
7.131.1 Подробное описание . . . . .	216
7.132nmppsAbsDiff . . . . .	217
7.132.1 Подробное описание . . . . .	217
7.133nmppsAbsDiff1 . . . . .	218
7.133.1 Подробное описание . . . . .	218
7.134nmppsMulC . . . . .	219
7.134.1 Подробное описание . . . . .	219
7.135nmppsMul_AddC . . . . .	220
7.135.1 Подробное описание . . . . .	220
7.136nmppsMulC_AddC . . . . .	221
7.136.1 Подробное описание . . . . .	221
7.136.2 Функции . . . . .	221
7.136.2.1 nmppsMulC_AddC_2x32s() . . . . .	222
7.137nmppsRShiftC_MulC_AddC . . . . .	223
7.137.1 Подробное описание . . . . .	223
7.138nmppsMulC_AddV_AddC . . . . .	224
7.138.1 Подробное описание . . . . .	224
7.139nmppsSumN . . . . .	225
7.139.1 Подробное описание . . . . .	225

7.140nmppsDivC . . . . .	226
7.140.1 Подробное описание . . . . .	226
7.141nmppsSum . . . . .	228
7.141.1 Подробное описание . . . . .	228
7.142nmppsDotProd . . . . .	229
7.142.1 Подробное описание . . . . .	229
7.143nmppsWeightedSum . . . . .	231
7.143.1 Подробное описание . . . . .	231
7.144nmppsNot_ . . . . .	232
7.144.1 Подробное описание . . . . .	232
7.145nmppsAndC . . . . .	233
7.145.1 Подробное описание . . . . .	233
7.146nmppsAnd . . . . .	234
7.146.1 Подробное описание . . . . .	234
7.147nmppsAnd4V_ . . . . .	235
7.147.1 Подробное описание . . . . .	235
7.148nmppsAndNotV_ . . . . .	236
7.148.1 Подробное описание . . . . .	236
7.149nmppsOrC . . . . .	237
7.149.1 Подробное описание . . . . .	237
7.150nmppsOr . . . . .	238
7.150.1 Подробное описание . . . . .	238
7.151nmppsOr3V_ . . . . .	239
7.151.1 Подробное описание . . . . .	239
7.152nmppsOr4V_ . . . . .	240
7.152.1 Подробное описание . . . . .	240
7.153nmppsXorC . . . . .	241
7.153.1 Подробное описание . . . . .	241
7.154nmppsXor . . . . .	242
7.154.1 Подробное описание . . . . .	242

7.155nmppsMaskV_ . . . . .	243
7.155.1 Подробное описание . . . . .	243
7.156nmppsRShiftC . . . . .	244
7.156.1 Подробное описание . . . . .	244
7.157nmppsRShiftC_ . . . . .	245
7.157.1 Подробное описание . . . . .	245
7.158nmppsRShiftC_AddC_ . . . . .	246
7.159nmppsDisplaceBits . . . . .	247
7.159.1 Подробное описание . . . . .	247
7.160nmppsSet-инициализация . . . . .	249
7.160.1 Подробное описание . . . . .	249
7.161nmppsRandUniform . . . . .	250
7.161.1 Подробное описание . . . . .	250
7.161.2 Функции . . . . .	250
7.161.2.1 nmppsRandUniform_64s() . . . . .	250
7.162nmppsRandUniform_ . . . . .	251
7.162.1 Подробное описание . . . . .	251
7.163nmppsRamp_ . . . . .	252
7.163.1 Подробное описание . . . . .	252
7.164nmppsConvert . . . . .	253
7.164.1 Подробное описание . . . . .	254
7.164.2 Функции . . . . .	254
7.164.2.1 nmppsConvert_1s2s() . . . . .	254
7.164.2.2 nmppsConvert_1u2u() . . . . .	254
7.164.2.3 nmppsConvert_32s32fcr() . . . . .	255
7.164.2.4 nmppsConvert_32sc32fcr() . . . . .	255
7.164.2.5 nmppsConvert_32u32fcr() . . . . .	255
7.164.2.6 nmppsConvertRisc_32u8u() . . . . .	256
7.164.2.7 nmppsConvertRisc_8u32u() . . . . .	256
7.164.2.8 nmppsJoin_32f() . . . . .	257

7.165nmppsCory_ . . . . .	258
7.165.1 Подробное описание . . . . .	258
7.166nmppsCoryua_ . . . . .	259
7.166.1 Подробное описание . . . . .	259
7.167nmppsSwap_ . . . . .	260
7.167.1 Подробное описание . . . . .	260
7.168nmppsMax_ . . . . .	261
7.168.1 Подробное описание . . . . .	261
7.168.2 Функции . . . . .	261
7.168.2.1 nmppsMax_16s15b() . . . . .	262
7.168.2.2 nmppsMax_32s31b() . . . . .	262
7.168.2.3 nmppsMax_8s7b() . . . . .	262
7.169nmppsMin . . . . .	263
7.169.1 Подробное описание . . . . .	263
7.169.2 Функции . . . . .	263
7.169.2.1 nmppsMin_16s15b() . . . . .	264
7.169.2.2 nmppsMin_32s31b() . . . . .	264
7.169.2.3 nmppsMin_8s7b() . . . . .	264
7.170nmppsMaxIndx_ . . . . .	265
7.170.1 Подробное описание . . . . .	265
7.171nmppsMinIndx_ . . . . .	267
7.171.1 Подробное описание . . . . .	267
7.172nmppsMinIndxVN_ . . . . .	269
7.172.1 Подробное описание . . . . .	269
7.173nmppsFirstZeroIndx . . . . .	270
7.173.1 Подробное описание . . . . .	270
7.174nmppsFirstNonZeroIndx . . . . .	271
7.174.1 Подробное описание . . . . .	271
7.175nmppsLastZeroIndx . . . . .	272
7.175.1 Подробное описание . . . . .	272

7.176nmppsLastNonZeroIdx . . . . .	273
7.176.1 Подробное описание . . . . .	273
7.177nmppsMinEvery_ . . . . .	274
7.177.1 Подробное описание . . . . .	274
7.178nmppsMaxEvery_ . . . . .	275
7.178.1 Подробное описание . . . . .	275
7.179nmppsMinCmpLtV_ . . . . .	277
7.179.1 Подробное описание . . . . .	277
7.180nmppsCmpLt0 . . . . .	279
7.180.1 Подробное описание . . . . .	279
7.181nmppsCmpEq0 . . . . .	280
7.181.1 Подробное описание . . . . .	280
7.181.2 Функции . . . . .	280
7.181.2.1 nmppsCmpEq0_16u15b() . . . . .	281
7.181.2.2 nmppsCmpEq0_32u31b() . . . . .	281
7.181.2.3 nmppsCmpEq0_8u7b() . . . . .	281
7.182nmppsCmpMinMaxV_ . . . . .	282
7.182.1 Подробное описание . . . . .	282
7.183nmppsClipPowC_ . . . . .	284
7.183.1 Подробное описание . . . . .	284
7.184nmppsClipCC_ . . . . .	285
7.184.1 Подробное описание . . . . .	285
7.185nmppsClipRShiftConvert_AddC_ . . . . .	286
7.185.1 Подробное описание . . . . .	286
7.186nmppsClipConvert_AddC_ . . . . .	287
7.186.1 Подробное описание . . . . .	287
7.187nmppsCmpEqC . . . . .	289
7.187.1 Подробное описание . . . . .	289
7.188nmppsCmpNe0 . . . . .	290
7.188.1 Подробное описание . . . . .	290

7.189nmppsCmpNeC . . . . .	291
7.189.1 Подробное описание . . . . .	292
7.190nmppsCmpEqV_ . . . . .	294
7.190.1 Подробное описание . . . . .	294
7.191nmppsCmpNeV_ . . . . .	295
7.191.1 Подробное описание . . . . .	295
7.192nmppsAddr_ . . . . .	296
7.192.1 Подробное описание . . . . .	296
7.193nmppsSetVal_ . . . . .	297
7.193.1 Подробное описание . . . . .	297
7.194nmppsGetVal_ . . . . .	298
7.194.1 Подробное описание . . . . .	298
7.195nmppsGetVal_ (return) . . . . .	299
7.195.1 Подробное описание . . . . .	299
7.196VEC_ QSort . . . . .	300
7.196.1 Подробное описание . . . . .	300
7.197nmppsRemap_ . . . . .	301
7.197.1 Подробное описание . . . . .	301
7.198nmppSplitTmp . . . . .	303
7.199nmppSplit . . . . .	304
7.200nmppMerge . . . . .	305
7.201nmppSplit_ 32fcr . . . . .	306
7.202nmppsDecimate . . . . .	307
7.203Типы данных . . . . .	308
7.203.1 Подробное описание . . . . .	308
7.204Векторные функции . . . . .	309
7.204.1 Подробное описание . . . . .	309
7.205Матричные функции . . . . .	310
7.205.1 Подробное описание . . . . .	310
7.206Функции обработки сигналов . . . . .	311

7.206.1 Подробное описание . . . . .	311
7.207 Функции обработки изображений . . . . .	312
7.207.1 Подробное описание . . . . .	312
7.208 Скалярные функции . . . . .	313
7.208.1 Подробное описание . . . . .	313
7.208.1.1 Введение . . . . .	313
7.209 Базовые регистровые функции библиотеки . . . . .	315
7.209.1 Подробное описание . . . . .	315
7.210 контроль переполнения . . . . .	316
7.210.1 Подробное описание . . . . .	316
7.210.2 Макросы . . . . .	316
7.210.2.1 GetVec . . . . .	316
7.210.3 Функции . . . . .	317
7.210.3.1 operator<<() . . . . .	317
7.211 Элементарные функции . . . . .	318
7.211.1 Подробное описание . . . . .	318
7.212 функции взвешенного суммирования . . . . .	319
7.212.1 Подробное описание . . . . .	319
7.213 Целевые функции . . . . .	320
7.213.1 Подробное описание . . . . .	320
7.214 Vec_0_sub_data . . . . .	321
7.214.1 Подробное описание . . . . .	321
7.215 Vec_activate_data . . . . .	322
7.215.1 Подробное описание . . . . .	322
7.216 Vec_activate_data_add_0 . . . . .	323
7.216.1 Подробное описание . . . . .	323
7.217 Vec_activate_data_xor_data . . . . .	324
7.217.1 Подробное описание . . . . .	324
7.218 Vec_activate_data_add_ram . . . . .	325
7.218.1 Подробное описание . . . . .	325



7.219Vec_Add_VV_shift . . . . .	326
7.219.1 Подробное описание . . . . .	326
7.220Vec_afifo . . . . .	327
7.220.1 Подробное описание . . . . .	327
7.221Vec_data . . . . .	328
7.221.1 Подробное описание . . . . .	328
7.222Vec_data_add_afifo . . . . .	329
7.222.1 Подробное описание . . . . .	329
7.223Vec_data_add_ram . . . . .	330
7.223.1 Подробное описание . . . . .	330
7.224Vec_data_and_ram . . . . .	331
7.224.1 Подробное описание . . . . .	331
7.225Vec_data_or_ram . . . . .	332
7.225.1 Подробное описание . . . . .	332
7.226Vec_data_sub_ram . . . . .	333
7.226.1 Подробное описание . . . . .	333
7.227Vec_data_xor_ram . . . . .	334
7.227.1 Подробное описание . . . . .	334
7.228Vec_FilterCoreRow2 . . . . .	335
7.228.1 Подробное описание . . . . .	335
7.229Vec_FilterCoreRow4 . . . . .	336
7.229.1 Подробное описание . . . . .	336
7.230Vec_FilterCoreRow8 . . . . .	337
7.230.1 Подробное описание . . . . .	337
7.231Vec_And . . . . .	338
7.231.1 Подробное описание . . . . .	338
7.232Vec_Mask . . . . .	339
7.232.1 Подробное описание . . . . .	339
7.233Vec_Or . . . . .	340
7.233.1 Подробное описание . . . . .	340

7.234Vec_Xor . . . . .	341
7.234.1 Подробное описание . . . . .	341
7.235Vec_Abs . . . . .	342
7.235.1 Подробное описание . . . . .	342
7.236Vec_Add . . . . .	343
7.236.1 Подробное описание . . . . .	343
7.237Vec_ClipExt . . . . .	344
7.237.1 Подробное описание . . . . .	344
7.238Vec_ClipMul2D2W8_AddVr . . . . .	345
7.238.1 Подробное описание . . . . .	345
7.239Vec_ClipMulNDNW2_AddVr . . . . .	346
7.239.1 Подробное описание . . . . .	346
7.240Vec_ClipMulNDNW4_AddVr . . . . .	347
7.240.1 Подробное описание . . . . .	347
7.241Vec_ClipMulNDNW8_AddVr . . . . .	348
7.241.1 Подробное описание . . . . .	348
7.242Vec_IncNeg . . . . .	349
7.242.1 Подробное описание . . . . .	349
7.243Vec_Mul2D2W1_AddVr . . . . .	350
7.243.1 Подробное описание . . . . .	350
7.244Vec_Mul2D2W2_AddVr . . . . .	351
7.244.1 Подробное описание . . . . .	351
7.245Vec_Mul2D2W4_AddVr . . . . .	352
7.245.1 Подробное описание . . . . .	352
7.246Vec_Mul2D2W8_AddVr . . . . .	353
7.246.1 Подробное описание . . . . .	353
7.247Vec_Mul3D3W2_AddVr . . . . .	354
7.247.1 Подробное описание . . . . .	354
7.248Vec_Mul3D3W8_AddVr . . . . .	355
7.248.1 Подробное описание . . . . .	355

7.249Vec_Mul4D4W2_AddVr . . . . .	356
7.249.1 Подробное описание . . . . .	356
7.250Vec_MulVN_AddVN . . . . .	357
7.250.1 Подробное описание . . . . .	357
7.251Vec_Sub . . . . .	358
7.251.1 Подробное описание . . . . .	358
7.252Vec_SubAbs . . . . .	359
7.252.1 Подробное описание . . . . .	359
7.253Vec_SubVN_Abs . . . . .	360
7.253.1 Подробное описание . . . . .	360
7.254Vec_Swap . . . . .	361
7.254.1 Подробное описание . . . . .	361
7.255Vec_MUL_2V4toW8_shift . . . . .	362
7.255.1 Подробное описание . . . . .	362
7.256Vec_MUL_2V8toW16_shift . . . . .	363
7.256.1 Подробное описание . . . . .	363
7.257Vec_not_data . . . . .	364
7.257.1 Подробное описание . . . . .	364
7.258Vec_ram . . . . .	365
7.258.1 Подробное описание . . . . .	365
7.259Vec_ram_sub_data . . . . .	366
7.259.1 Подробное описание . . . . .	366
7.260Vec_vsum_activate_data_0 . . . . .	367
7.260.1 Подробное описание . . . . .	367
7.261Vec_vsum_data_0 . . . . .	368
7.261.1 Подробное описание . . . . .	368
7.262Vec_vsum_data_affo . . . . .	369
7.262.1 Подробное описание . . . . .	369
7.263Vec_vsum_data_vr . . . . .	370
7.263.1 Подробное описание . . . . .	370

7.264Vec_vsum_shift_data_0 . . . . .	371
7.264.1 Подробное описание . . . . .	371
7.265Vec_vsum_shift_data_vr . . . . .	372
7.265.1 Подробное описание . . . . .	372
7.266Vec_vsum_shift_data_afifo . . . . .	373
7.266.1 Подробное описание . . . . .	373
7.267Vec_CompareMinV . . . . .	374
7.267.1 Подробное описание . . . . .	374
7.268Vec_CompareMaxV . . . . .	375
7.268.1 Подробное описание . . . . .	375
7.269Vec_DupValueInVector8 . . . . .	376
7.269.1 Подробное описание . . . . .	376
7.270Vec_DupValueInVector16 . . . . .	377
7.270.1 Подробное описание . . . . .	377
7.271Vec_BuildDiagWeights8 . . . . .	378
7.271.1 Подробное описание . . . . .	378
7.272Vec_BuildDiagWeights16 . . . . .	379
7.272.1 Подробное описание . . . . .	379
7.273Vec_MaxVal_v8nm8s . . . . .	380
7.273.1 Подробное описание . . . . .	380
7.274Vec_MaxVal_v4nm16s . . . . .	381
7.274.1 Подробное описание . . . . .	381
7.275Vec_MaxVal . . . . .	382
7.275.1 Подробное описание . . . . .	382
7.276Vec_MinVal_v8nm8s . . . . .	383
7.276.1 Подробное описание . . . . .	383
7.277Vec_MinVal_v4nm16s . . . . .	384
7.277.1 Подробное описание . . . . .	384
7.278Vec_MinVal . . . . .	385
7.278.1 Подробное описание . . . . .	385
7.279Vec_AccMul1D1W32_AddVr . . . . .	386
7.279.1 Подробное описание . . . . .	386

8	Классы	387
8.1	Класс <code>C_2DSubPixelMinPosition</code>	387
8.1.1	Подробное описание	387
8.2	Класс <code>C_2DTrigSubPixelMinPosition</code>	387
8.2.1	Подробное описание	388
8.3	Класс <code>C_Allocator32</code>	388
8.3.1	Подробное описание	388
8.4	Шаблон класса <code>C_BoxImg&lt; T &gt;</code>	389
8.4.1	Подробное описание	389
8.5	Шаблон класса <code>C_BoxVec&lt; T &gt;</code>	389
8.5.1	Подробное описание	390
8.6	Класс <code>C_Heap</code>	390
8.6.1	Подробное описание	391
8.6.2	Методы	391
8.6.2.1	<code>AllocateMaxAvail()</code>	391
8.7	Шаблон класса <code>C_Img&lt; T &gt;</code>	391
8.7.1	Подробное описание	392
8.8	Класс <code>C_MultiHeap</code>	392
8.8.1	Подробное описание	393
8.9	Класс <code>C_PlessyCornerDetector</code>	393
8.9.1	Подробное описание	394
8.10	Класс <code>C_PlessyCornerDetector_16s</code>	394
8.10.1	Подробное описание	395
8.11	Класс <code>C_PlessyCornerDetector_32f</code>	395
8.11.1	Подробное описание	396
8.12	Шаблон класса <code>C_RingBufferRemote&lt; T &gt;</code>	396
8.12.1	Подробное описание	397
8.13	Шаблон класса <code>C_WarpImg&lt; T &gt;</code>	397
8.13.1	Подробное описание	398
8.14	Шаблон класса <code>CIMG_FIR&lt; nmbits_in, nmbits_out &gt;</code>	398

8.14.1	Подробное описание . . . . .	399
8.14.2	Конструктор(ы) . . . . .	399
8.14.2.1	CIMG_FIR() . . . . .	399
8.14.3	Методы . . . . .	400
8.14.3.1	Filter() . . . . .	400
8.14.3.2	SetWeights() . . . . .	400
8.15	Структура ds_struct . . . . .	401
8.15.1	Подробное описание . . . . .	401
8.16	Класс EnterHardMode . . . . .	401
8.16.1	Подробное описание . . . . .	401
8.17	Класс I_2DSubPixelMinPosition . . . . .	401
8.17.1	Подробное описание . . . . .	402
8.18	Класс I_PlessyCornerDetector . . . . .	402
8.18.1	Подробное описание . . . . .	402
8.19	Структура int15in16x4 . . . . .	402
8.19.1	Подробное описание . . . . .	403
8.20	Структура int30in32x2 . . . . .	403
8.20.1	Подробное описание . . . . .	403
8.21	Структура int31in32x2 . . . . .	403
8.21.1	Подробное описание . . . . .	403
8.22	Шаблон класса mtr< T > . . . . .	404
8.22.1	Подробное описание . . . . .	405
8.23	Структура nm16sc . . . . .	405
8.23.1	Подробное описание . . . . .	405
8.24	Класс nmchar . . . . .	406
8.24.1	Подробное описание . . . . .	406
8.25	Шаблон класса nmchar1D< N > . . . . .	406
8.25.1	Подробное описание . . . . .	406
8.26	Шаблон класса nmchar2D< Y, X > . . . . .	407
8.26.1	Подробное описание . . . . .	407

8.27	Шаблон класса <code>nmintpack&lt; T &gt;</code>	407
8.27.1	Подробное описание	407
8.28	Шаблон класса <code>nmmtr&lt; T &gt;</code>	408
8.28.1	Подробное описание	409
8.29	Структура <code>NmprriFFTSpec_32fcr</code>	409
8.29.1	Подробное описание	409
8.30	Структура <code>NmprpsFFTSpec</code>	409
8.30.1	Подробное описание	410
8.31	Структура <code>NmprpsFFTSpec_32fcr</code>	410
8.31.1	Подробное описание	410
8.32	Структура <code>NmprpsFrame_16s</code>	410
8.32.1	Подробное описание	410
8.33	Структура <code>NmprpsFrame_16u</code>	410
8.33.1	Подробное описание	411
8.34	Структура <code>NmprpsFrame_32s</code>	411
8.34.1	Подробное описание	411
8.35	Структура <code>NmprpsFrame_32u</code>	411
8.35.1	Подробное описание	411
8.36	Структура <code>NmprpsFrame_64s</code>	411
8.36.1	Подробное описание	412
8.37	Структура <code>NmprpsFrame_64u</code>	412
8.37.1	Подробное описание	412
8.38	Структура <code>NmprpsFrame_8s</code>	412
8.38.1	Подробное описание	412
8.39	Структура <code>NmprpsFrame_8u</code>	412
8.39.1	Подробное описание	413
8.40	Структура <code>NmprpsMallocSpec</code>	413
8.40.1	Подробное описание	413
8.41	Структура <code>NmprpsTmpSpec</code>	413
8.41.1	Подробное описание	413

8.42 Структура <code>nmreg</code> . . . . .	414
8.42.1 Подробное описание . . . . .	414
8.43 Класс <code>nmshort</code> . . . . .	414
8.43.1 Подробное описание . . . . .	414
8.44 Шаблон класса <code>nmshort2D&lt; Y, X &gt;</code> . . . . .	415
8.44.1 Подробное описание . . . . .	415
8.45 Шаблон класса <code>nmvespack&lt; T &gt;</code> . . . . .	415
8.45.1 Подробное описание . . . . .	416
8.46 Структура <code>RGB32_nm10s</code> . . . . .	417
8.46.1 Подробное описание . . . . .	417
8.47 Структура <code>RGB32_nm10u</code> . . . . .	417
8.47.1 Подробное описание . . . . .	417
8.48 Структура <code>RGB32_nm8s</code> . . . . .	417
8.48.1 Подробное описание . . . . .	418
8.49 Структура <code>RGB32_nm8u</code> . . . . .	418
8.49.1 Подробное описание . . . . .	418
8.50 Структура <code>RGB64_nm16u</code> . . . . .	418
8.50.1 Подробное описание . . . . .	418
8.51 Класс <code>RPoint</code> . . . . .	419
8.51.1 Подробное описание . . . . .	419
8.52 Структура <code>S_BufferInfo</code> . . . . .	419
8.52.1 Подробное описание . . . . .	420
8.53 Структура <code>S_IMG_FilterKernel</code> . . . . .	420
8.53.1 Подробное описание . . . . .	420
8.54 Структура <code>S_IMG_FilterKernel_32s32s</code> . . . . .	420
8.54.1 Подробное описание . . . . .	420
8.55 Структура <code>s_int32x2</code> . . . . .	421
8.55.1 Подробное описание . . . . .	421
8.56 Структура <code>s_nm32fc</code> . . . . .	421
8.56.1 Подробное описание . . . . .	421



8.57 Структура <code>s_nm32fcr</code> . . . . .	421
8.57.1 Подробное описание . . . . .	421
8.58 Структура <code>s_nm32sc</code> . . . . .	422
8.58.1 Подробное описание . . . . .	422
8.59 Структура <code>s_nm64sc</code> . . . . .	422
8.59.1 Подробное описание . . . . .	422
8.59.2 Данные класса . . . . .	422
8.59.2.1 <code>im</code> . . . . .	422
8.59.2.2 <code>re</code> . . . . .	423
8.60 Структура <code>s_v16nm16s</code> . . . . .	423
8.60.1 Подробное описание . . . . .	423
8.61 Структура <code>s_v16nm16u</code> . . . . .	423
8.61.1 Подробное описание . . . . .	423
8.62 Структура <code>s_v16nm32s</code> . . . . .	424
8.62.1 Подробное описание . . . . .	424
8.63 Структура <code>s_v16nm32u</code> . . . . .	424
8.63.1 Подробное описание . . . . .	424
8.64 Структура <code>s_v16nm4u</code> . . . . .	424
8.64.1 Подробное описание . . . . .	425
8.65 Структура <code>s_v16nm8s</code> . . . . .	425
8.65.1 Подробное описание . . . . .	425
8.66 Структура <code>s_v16nm8u</code> . . . . .	425
8.66.1 Подробное описание . . . . .	425
8.67 Структура <code>s_v2nm32s</code> . . . . .	426
8.67.1 Подробное описание . . . . .	426
8.68 Структура <code>s_v2nm32u</code> . . . . .	426
8.68.1 Подробное описание . . . . .	426
8.69 Структура <code>s_v4nm16s</code> . . . . .	426
8.69.1 Подробное описание . . . . .	427
8.70 Структура <code>s_v4nm16u</code> . . . . .	427

8.70.1	Подробное описание . . . . .	427
8.71	Структура <code>s_v4nm32s</code> . . . . .	427
8.71.1	Подробное описание . . . . .	427
8.72	Структура <code>s_v4nm32u</code> . . . . .	428
8.72.1	Подробное описание . . . . .	428
8.73	Структура <code>s_v4nm8u</code> . . . . .	428
8.73.1	Подробное описание . . . . .	428
8.74	Структура <code>s_v8nm16s</code> . . . . .	428
8.74.1	Подробное описание . . . . .	429
8.75	Структура <code>s_v8nm16u</code> . . . . .	429
8.75.1	Подробное описание . . . . .	429
8.76	Структура <code>s_v8nm32s</code> . . . . .	429
8.76.1	Подробное описание . . . . .	429
8.77	Структура <code>s_v8nm32u</code> . . . . .	430
8.77.1	Подробное описание . . . . .	430
8.78	Структура <code>s_v8nm8s</code> . . . . .	430
8.78.1	Подробное описание . . . . .	430
8.79	Структура <code>s_v8nm8u</code> . . . . .	430
8.79.1	Подробное описание . . . . .	431
8.80	Структура <code>SpecTmp1</code> . . . . .	431
8.80.1	Подробное описание . . . . .	431
8.81	Структура <code>spot_struct</code> . . . . .	431
8.81.1	Подробное описание . . . . .	432
8.82	Структура <code>tagSegmentInfo</code> . . . . .	432
8.82.1	Подробное описание . . . . .	432
8.83	Шаблон класса <code>tfxpoint&lt; T, point &gt;</code> . . . . .	432
8.83.1	Подробное описание . . . . .	433
8.84	Структура <code>Tmp2BuffSpec</code> . . . . .	433
8.84.1	Подробное описание . . . . .	434
8.85	Класс <code>uint16ptr</code> . . . . .	434
8.85.1	Подробное описание . . . . .	434
8.86	Класс <code>uint8ptr</code> . . . . .	435
8.86.1	Подробное описание . . . . .	435
8.87	Структура <code>v16nm4s</code> . . . . .	435
8.87.1	Подробное описание . . . . .	436
8.88	Структура <code>v4nm8s</code> . . . . .	436
8.88.1	Подробное описание . . . . .	436
8.89	Шаблон класса <code>vec&lt; T &gt;</code> . . . . .	436
8.89.1	Подробное описание . . . . .	438

9	Файлы	439
9.1	Файл D:/GIT/nmpp/include/nmblas.h	439
9.1.1	Подробное описание	440
9.2	Файл D:/GIT/nmpp/include/nmtype.h	441
9.2.1	Подробное описание	443
9.2.2	Макросы	444
9.2.2.1	CAPACITY_nm64s	444
9.2.2.2	VEC_NM16S	444
9.2.2.3	VEC_NM16U	444
9.2.2.4	VEC_NM32S	445
9.2.2.5	VEC_NM4U	445
9.2.2.6	VEC_NM8S	445
	Алфавитный указатель	447



# Глава 1

## Введение

### 1.1 Introduction

This is the introduction.

### 1.2 Installation

#### 1.2.1 Step 1: Opening the box

etc...



## Глава 2

## Ошибки

Файл [nmblas.h](#)





## Глава 3

# Алфавитный указатель групп

### 3.1 Группы

Полный список групп.

DFT-8 . . . . .	19
FFT-16 . . . . .	21
FFT-32 . . . . .	23
FFT-64 . . . . .	25
FFT-128 . . . . .	27
FFT-256 . . . . .	29
FFT-512 . . . . .	31
FFT-1024 . . . . .	33
FFT-2048 . . . . .	35
FFT-4096 . . . . .	37
IDFT-8 . . . . .	39
IFFT-16 . . . . .	40
IFFT-32 . . . . .	42
IFFT-64 . . . . .	44
IFFT-128 . . . . .	46
IFFT-256 . . . . .	48
IFFT-512 . . . . .	50
IFFT-1024 . . . . .	52
IFFT-2048 . . . . .	54
IFFT-4096 . . . . .	56
FFT-Common . . . . .	58
IFFT-Common . . . . .	61
BLASS-LEVEL1 . . . . .	66
FFT-256 . . . . .	136
IFFT-256 . . . . .	138
FFT-512 . . . . .	141
IFFT-512 . . . . .	143
FFT-1024 . . . . .	146
IFFT-1024 . . . . .	148
FFT-2048 . . . . .	151
IFFT-2048 . . . . .	153
FFT-4096 . . . . .	156
IFFT-4096 . . . . .	158
FFT-8192 . . . . .	160
IFFT-8192 . . . . .	162

Типы данных . . . . .	308
Типы векторных данных . . . . .	182
Типы скалярных данных . . . . .	195
Векторные функции . . . . .	309
Функции поддержки . . . . .	202
nmppsMalloc . . . . .	64
nmppsFree . . . . .	65
nmppsAddr_ . . . . .	296
nmppsSetVal_ . . . . .	297
nmppsGetVal_ . . . . .	298
nmppsGetVal_ (return) . . . . .	299
Инициализация и копирование . . . . .	203
nmppsSet-инициализация . . . . .	249
nmppsRandUniform . . . . .	250
nmppsRandUniform_ . . . . .	251
nmppsRamp_ . . . . .	252
nmppsConvert . . . . .	253
nmppsCopy_ . . . . .	258
nmppsCopyua_ . . . . .	259
nmppsSwap_ . . . . .	260
Арифметические операции . . . . .	204
nmppsAbs . . . . .	208
nmppsAbs1 . . . . .	209
nmppsNeg . . . . .	210
nmppsAddC . . . . .	211
nmppsAdd . . . . .	212
nmppsAdd_AddC . . . . .	213
nmppsSubC . . . . .	214
nmppsSubCRev . . . . .	215
nmppsSub . . . . .	216
nmppsAbsDiff . . . . .	217
nmppsAbsDiff1 . . . . .	218
nmppsMulC . . . . .	219
nmppsMul_AddC . . . . .	220
nmppsMulC_AddC . . . . .	221
nmppsRShiftC_MulC_AddC . . . . .	223
nmppsMulC_AddV_AddC . . . . .	224
nmppsSumN . . . . .	225
nmppsDivC . . . . .	226
nmppsSum . . . . .	228
nmppsDotProd . . . . .	229
nmppsWeightedSum . . . . .	231
Логические и бинарные операции . . . . .	205
nmppsNot_ . . . . .	232
nmppsAndC . . . . .	233
nmppsAnd . . . . .	234
nmppsAnd4V_ . . . . .	235
nmppsAndNotV_ . . . . .	236
nmppsOrC . . . . .	237
nmppsOr . . . . .	238
nmppsOr3V_ . . . . .	239
nmppsOr4V_ . . . . .	240
nmppsXorC . . . . .	241
nmppsXor . . . . .	242
nmppsMaskV_ . . . . .	243
nmppsRShiftC . . . . .	244
nmppsRShiftC_ . . . . .	245
nmppsRShiftC_AddC_ . . . . .	246

nmppsDisplaceBits . . . . .	247
Операции сравнения . . . . .	206
nmppsMax_ . . . . .	261
nmppsMin . . . . .	263
nmppsMaxIndx_ . . . . .	265
nmppsMinIndx_ . . . . .	267
nmppsMinIndxVN_ . . . . .	269
nmppsFirstZeroIndx . . . . .	270
nmppsFirstNonZeroIndx . . . . .	271
nmppsLastZeroIndx . . . . .	272
nmppsLastNonZeroIndx . . . . .	273
nmppsMinEvery_ . . . . .	274
nmppsMaxEvery_ . . . . .	275
nmppsMinCmpLtV_ . . . . .	277
nmppsCmpLt0 . . . . .	279
nmppsCmpEq0 . . . . .	280
nmppsCmpMinMaxV_ . . . . .	282
nmppsClipPowC_ . . . . .	284
nmppsClipCC_ . . . . .	285
nmppsClipRShiftConvert_ AddC_ . . . . .	286
nmppsClipConvert_ AddC_ . . . . .	287
nmppsCmpEqC . . . . .	289
nmppsCmpNe0 . . . . .	290
nmppsCmpNeC . . . . .	291
nmppsCmpEqV_ . . . . .	294
nmppsCmpNeV_ . . . . .	295
Переупорядочивание и сортировка . . . . .	207
VEC_QSort . . . . .	300
nmppsRemap_ . . . . .	301
nmppSplitTmp . . . . .	303
nmppSplit . . . . .	304
nmppMerge . . . . .	305
nmppSplit_32fcr . . . . .	306
nmppsDecimate . . . . .	307
Матричные функции . . . . .	310
Инициализация и копирование . . . . .	120
nmppmCopyua . . . . .	121
MTR_Copyau . . . . .	122
MTR_Copy . . . . .	123
Функции поддержки . . . . .	134
MTR_Malloc . . . . .	129
MTR_Free . . . . .	130
MTR_Addr . . . . .	131
MTR_SetVal . . . . .	132
MTR_GetVal . . . . .	133
Векторно-матричные операции . . . . .	135
nmppmMul_mm . . . . .	124
nmppmMul_mv_ . . . . .	126
nmppmMul_mv_ AddC . . . . .	127
MTR_ProdUnitV . . . . .	128
Функции обработки сигналов . . . . .	311
Свертка . . . . .	164
SIG_XCorr . . . . .	168
Масочная фильтрация . . . . .	165
SIG_Median3 . . . . .	170
КИХ-фильтрация . . . . .	171
nmppsFIR_Xs . . . . .	172

nmppsFIRInit_Xs . . . . .	173
nmppsFIRInitAlloc_Xs . . . . .	174
nmppsFIRGetStateSize_Xs . . . . .	175
nmppsFIRFree . . . . .	176
Изменение размеров . . . . .	166
SIG_ResampleDown2 . . . . .	177
SIG_ResampleUp3Down2 . . . . .	178
SIG_CreateResample . . . . .	179
SIG_SetResample . . . . .	180
SIG_Resample_perf . . . . .	181
Быстрое преобразование Фурье . . . . .	167
Функции обработки изображений . . . . .	312
Floodfill . . . . .	100
Переупорядочивание изображений . . . . .	108
Блочное переупорядочивание . . . . .	109
IMG_SplitIntoBlocks . . . . .	110
IMG_MergeFromBlocks . . . . .	111
Арифметические действия . . . . .	114
Масочная фильтрация . . . . .	115
КИХ-фильтрация . . . . .	99
Инициализация и копирование . . . . .	116
IMG_Convert . . . . .	106
IMG_RGB32ToGray . . . . .	107
Функции поддержки . . . . .	117
IMG_Free . . . . .	112
IMG_Release . . . . .	113
Функции графического вывода текста . . . . .	118
Скалярные функции . . . . .	313
Инициализация . . . . .	92
nmppsRand . . . . .	90
Integer operations . . . . .	93
nmppcSqrt . . . . .	91
Fix-point 64 . . . . .	94
nmppcFixSqrt64 . . . . .	83
nmppcDoubleToFix64 . . . . .	84
nmppcFix64ToDouble . . . . .	85
nmppcFixDiv64 . . . . .	86
nmppcFixSinCos64 . . . . .	87
nmppcFixArcTan64 . . . . .	88
Fix-point 32 . . . . .	95
nmppcFixExp32 . . . . .	70
nmppcFixSinCos32 . . . . .	71
nmppcFixArcTan32 . . . . .	72
nmppcDoubleToFix32 . . . . .	73
nmppcFix32ToDouble . . . . .	74
nmppcFixSqrt32 . . . . .	75
nmppcFixMul32 . . . . .	76
nmppcFixInv32 . . . . .	77
nmppcTblFixArcSin32 . . . . .	78
nmppcTblFixArcCos32 . . . . .	79
nmppcTblFixCos32 . . . . .	80
nmppcTblFixSin32 . . . . .	81
nmppcFixDivMod32 . . . . .	82
nmppcFix64Exp01 . . . . .	89
Арифметические операции . . . . .	96
nmppcDivC . . . . .	68

nmprcProdC . . . . .	69
Функции деинтерлейсинга . . . . .	97
Базовые регистровые функции библиотеки . . . . .	315
Элементарные функции . . . . .	318
Vec_0_sub_data . . . . .	321
Vec_activate_data . . . . .	322
Vec_activate_data_add_0 . . . . .	323
Vec_activate_data_xor_data . . . . .	324
Vec_activate_data_add_ram . . . . .	325
Vec_afifo . . . . .	327
Vec_data . . . . .	328
Vec_data_add_ram . . . . .	330
Vec_data_and_ram . . . . .	331
Vec_data_or_ram . . . . .	332
Vec_data_sub_ram . . . . .	333
Vec_data_xor_ram . . . . .	334
Vec_And . . . . .	338
Vec_Mask . . . . .	339
Vec_Or . . . . .	340
Vec_Xor . . . . .	341
Vec_Add . . . . .	343
Vec_Sub . . . . .	358
Vec_not_data . . . . .	364
Vec_ram . . . . .	365
Vec_ram_sub_data . . . . .	366
Vec_vsum_activate_data_0 . . . . .	367
функции взвешенного суммирования . . . . .	319
Vec_ClipMul2D2W8_AddVr . . . . .	345
Vec_ClipMulNDNW2_AddVr . . . . .	346
Vec_ClipMulNDNW4_AddVr . . . . .	347
Vec_ClipMulNDNW8_AddVr . . . . .	348
Vec_Mul2D2W1_AddVr . . . . .	350
Vec_Mul2D2W2_AddVr . . . . .	351
Vec_Mul2D2W4_AddVr . . . . .	352
Vec_Mul2D2W8_AddVr . . . . .	353
Vec_Mul3D3W2_AddVr . . . . .	354
Vec_Mul3D3W8_AddVr . . . . .	355
Vec_Mul4D4W2_AddVr . . . . .	356
Vec_MulVN_AddVN . . . . .	357
Vec_vsum_data_0 . . . . .	368
Vec_vsum_data_vr . . . . .	370
Vec_vsum_shift_data_0 . . . . .	371
Vec_vsum_shift_data_vr . . . . .	372
Vec_vsum_shift_data_afifo . . . . .	373
Целевые функции . . . . .	320
Vec_Add_VV_shift . . . . .	326
Vec_data_add_afifo . . . . .	329
Vec_FilterCoreRow2 . . . . .	335
Vec_FilterCoreRow4 . . . . .	336
Vec_FilterCoreRow8 . . . . .	337
Vec_Abs . . . . .	342
Vec_ClipExt . . . . .	344
Vec_IncNeg . . . . .	349
Vec_SubAbs . . . . .	359
Vec_SubVN_Abs . . . . .	360
Vec_Swap . . . . .	361
Vec_MUL_2V4toW8_shift . . . . .	362
Vec_MUL_2V8toW16_shift . . . . .	363

Vec_vsum_data_afifo . . . . .	369
Vec_CompareMinV . . . . .	374
Vec_CompareMaxV . . . . .	375
Vec_DupValueInVector8 . . . . .	376
Vec_DupValueInVector16 . . . . .	377
Vec_BuildDiagWeights8 . . . . .	378
Vec_BuildDiagWeights16 . . . . .	379
Vec_MaxVal_v8nm8s . . . . .	380
Vec_MaxVal_v4nm16s . . . . .	381
Vec_MaxVal . . . . .	382
Vec_MinVal_v8nm8s . . . . .	383
Vec_MinVal_v4nm16s . . . . .	384
Vec_MinVal . . . . .	385
Vec_AccMul1D1W32_AddVr . . . . .	386
контроль переполнения . . . . .	316

## Глава 4

# Иерархический список классов

### 4.1 Иерархия классов

Иерархия классов.

C_Allocator32 . . . . .	388
C_MultiHeap . . . . .	392
C_BoxImg< T > . . . . .	389
C_BoxVec< T > . . . . .	389
C_Heap . . . . .	390
C_Img< T > . . . . .	391
C_RingBufferRemote< T > . . . . .	396
C_WarpImg< T > . . . . .	397
CIMG_FIR< nmbits_in, nmbits_out > . . . . .	398
ds_struct . . . . .	401
EnterHardMode . . . . .	401
I_2DSubPixelMinPosition . . . . .	401
C_2DSubPixelMinPosition . . . . .	387
C_2DTrigSubPixelMinPosition . . . . .	387
I_PlessyCornerDetector . . . . .	402
C_PlessyCornerDetector . . . . .	393
C_PlessyCornerDetector_16s . . . . .	394
C_PlessyCornerDetector_32f . . . . .	395
int15in16x4 . . . . .	402
int30in32x2 . . . . .	403
int31in32x2 . . . . .	403
mtr< T > . . . . .	404
nm16sc . . . . .	405
nmchar . . . . .	406
nmchar1D< N > . . . . .	406
nmchar2D< Y, X > . . . . .	407
nmintpack< T > . . . . .	407
nmnmtr< T > . . . . .	408
NmppiFFTSpec_32fcr . . . . .	409
NmppsFFTSpec . . . . .	409
NmppsFFTSpec_32fcr . . . . .	410
NmppsFrame_16s . . . . .	410
NmppsFrame_16u . . . . .	410
NmppsFrame_32s . . . . .	411

NmppsFrame_32u . . . . .	411
NmppsFrame_64s . . . . .	411
NmppsFrame_64u . . . . .	412
NmppsFrame_8s . . . . .	412
NmppsFrame_8u . . . . .	412
NmppsMallocSpec . . . . .	413
NmppsTmpSpec . . . . .	413
nmreg . . . . .	414
nmshort . . . . .	414
nmshort2D< Y, X > . . . . .	415
nmvecpack< T > . . . . .	415
RGB32_nm10s . . . . .	417
RGB32_nm10u . . . . .	417
RGB32_nm8s . . . . .	417
RGB32_nm8u . . . . .	418
RGB64_nm16u . . . . .	418
RPoint . . . . .	419
S_BufferInfo . . . . .	419
S_IMG_FilterKernel . . . . .	420
S_IMG_FilterKernel_32s32s . . . . .	420
s_int32x2 . . . . .	421
s_nm32fc . . . . .	421
s_nm32fcr . . . . .	421
s_nm32sc . . . . .	422
s_nm64sc . . . . .	422
s_v16nm16s . . . . .	423
s_v16nm16u . . . . .	423
s_v16nm32s . . . . .	424
s_v16nm32u . . . . .	424
s_v16nm4u . . . . .	424
s_v16nm8s . . . . .	425
s_v16nm8u . . . . .	425
s_v2nm32s . . . . .	426
s_v2nm32u . . . . .	426
s_v4nm16s . . . . .	426
s_v4nm16u . . . . .	427
s_v4nm32s . . . . .	427
s_v4nm32u . . . . .	428
s_v4nm8u . . . . .	428
s_v8nm16s . . . . .	428
s_v8nm16u . . . . .	429
s_v8nm32s . . . . .	429
s_v8nm32u . . . . .	430
s_v8nm8s . . . . .	430
s_v8nm8u . . . . .	430
SpecTmp1 . . . . .	431
spot_struct . . . . .	431
tagSegmentInfo . . . . .	432
tfixpoint< T, point > . . . . .	432
Tmp2BuffSpec . . . . .	433
uint16ptr . . . . .	434
uint8ptr . . . . .	435
v16nm4s . . . . .	435
v4nm8s . . . . .	436
vec< T > . . . . .	436



## Глава 5

# Алфавитный указатель классов

### 5.1 Классы

Классы с их кратким описанием.

<a href="#">C_2DSubPixelMinPosition</a>	387
<a href="#">C_2DTrigSubPixelMinPosition</a>	387
<a href="#">C_Allocator32</a>	388
<a href="#">C_BoxImg&lt; T &gt;</a>	389
<a href="#">C_BoxVec&lt; T &gt;</a>	389
<a href="#">C_Heap</a>	
класс - куча	390
<a href="#">C_Img&lt; T &gt;</a>	391
<a href="#">C_MultiHeap</a>	392
<a href="#">C_PlessyCornerDetector</a>	393
<a href="#">C_PlessyCornerDetector_16s</a>	394
<a href="#">C_PlessyCornerDetector_32f</a>	395
<a href="#">C_RingBufferRemote&lt; T &gt;</a>	396
<a href="#">C_WarpImg&lt; T &gt;</a>	397
<a href="#">CIMG_FIR&lt; nmbits_in, nmbits_out &gt;</a>	398
<a href="#">ds_struct</a>	401
<a href="#">EnterHardMode</a>	401
<a href="#">I_2DSubPixelMinPosition</a>	401
<a href="#">I_PlessyCornerDetector</a>	402
<a href="#">int15in16x4</a>	402
<a href="#">int30in32x2</a>	403
<a href="#">int31in32x2</a>	403
<a href="#">mtr&lt; T &gt;</a>	404
<a href="#">nm16sc</a>	405
<a href="#">nmchar</a>	406
<a href="#">nmchar1D&lt; N &gt;</a>	406
<a href="#">nmchar2D&lt; Y, X &gt;</a>	407
<a href="#">nmintpack&lt; T &gt;</a>	407
<a href="#">nmmtr&lt; T &gt;</a>	408
<a href="#">NmppiFFTSpec_32fcr</a>	409
<a href="#">NmppsFFTSpec</a>	409
<a href="#">NmppsFFTSpec_32fcr</a>	410
<a href="#">NmppsFrame_16s</a>	410
<a href="#">NmppsFrame_16u</a>	410
<a href="#">NmppsFrame_32s</a>	411

NmppsFrame_32u	411
NmppsFrame_64s	411
NmppsFrame_64u	412
NmppsFrame_8s	412
NmppsFrame_8u	412
NmppsMallocSpec	413
NmppsTmpSpec	413
nmreg	414
nmshort	414
nmshort2D< Y, X >	415
nmvecpack< T >	415
RGB32_nm10s	417
RGB32_nm10u	417
RGB32_nm8s	417
RGB32_nm8u	418
RGB64_nm16u	418
RPoint	419
S_BufferInfo	
класс буфер - заголовок в начале выделяемой динамической памяти	419
S_IMG_FilterKernel	420
S_IMG_FilterKernel_32s32s	420
s_int32x2	421
s_nm32fc	421
s_nm32fcr	421
s_nm32sc	422
s_nm64sc	422
s_v16nm16s	423
s_v16nm16u	423
s_v16nm32s	424
s_v16nm32u	424
s_v16nm4u	424
s_v16nm8s	425
s_v16nm8u	425
s_v2nm32s	426
s_v2nm32u	426
s_v4nm16s	426
s_v4nm16u	427
s_v4nm32s	427
s_v4nm32u	428
s_v4nm8u	428
s_v8nm16s	428
s_v8nm16u	429
s_v8nm32s	429
s_v8nm32u	430
s_v8nm8s	430
s_v8nm8u	430
SpecTmp1	431
spot_struct	431
tagSegmentInfo	432
tfixpoint< T, point >	432
Tmp2BuffSpec	433
uint16ptr	434
uint8ptr	435
v16nm4s	435
v4nm8s	436
vec< T >	436

## Глава 6

# Список файлов

### 6.1 Файлы

Полный список документированных файлов.

D:/GIT/nmpp/include/crtdbg2.h	??
D:/GIT/nmpp/include/fft.h	??
D:/GIT/nmpp/include/fft_32fcr.h	??
D:/GIT/nmpp/include/fftexp.h	??
D:/GIT/nmpp/include/macros_fpu.h	??
D:/GIT/nmpp/include/malloc32.h	??
D:/GIT/nmpp/include/metric.h	??
D:/GIT/nmpp/include/minrep.h	??
D:/GIT/nmpp/include/multiheap.h	??
D:/GIT/nmpp/include/nmblas.h	
Nmblas (NeuroMatrix Basic Linear Algebra Subroutines)	439
D:/GIT/nmpp/include/nmchar.h	??
D:/GIT/nmpp/include/nmdef.h	??
D:/GIT/nmpp/include/nmplc.h	??
D:/GIT/nmpp/include/nmpli.h	??
D:/GIT/nmpp/include/nmplm.h	??
D:/GIT/nmpp/include/nmpls.h	??
D:/GIT/nmpp/include/nmplv.h	??
D:/GIT/nmpp/include/nmpp.h	??
D:/GIT/nmpp/include/nmshort.h	??
D:/GIT/nmpp/include/nmtl.h	??
D:/GIT/nmpp/include/nmtype.h	441
D:/GIT/nmpp/include/ringremote.h	??
D:/GIT/nmpp/include/test.h	??
D:/GIT/nmpp/include/tfixpoint.h	??
D:/GIT/nmpp/include/nmplc/cArithmetic.h	??
D:/GIT/nmpp/include/nmplc/cfixpnt32.h	??
D:/GIT/nmpp/include/nmplc/cfixpnt64.h	??
D:/GIT/nmpp/include/nmplc/cInit.h	??
D:/GIT/nmpp/include/nmplc/cInteger.h	??
D:/GIT/nmpp/include/nmplc/nmplc.h	??
D:/GIT/nmpp/include/nmpli/filter.h	??
D:/GIT/nmpp/include/nmpli/iArithmetics.h	??
D:/GIT/nmpp/include/nmpli/iCellTexture.h	??
D:/GIT/nmpp/include/nmpli/iDef.h	??

D:/GIT/nmpp/include/nmpli/iDeinterlace.h	??
D:/GIT/nmpp/include/nmpli/iFilter.h	??
D:/GIT/nmpp/include/nmpli/iFiltration.h	??
D:/GIT/nmpp/include/nmpli/iFloodFill.h	??
D:/GIT/nmpp/include/nmpli/iInit.h	??
D:/GIT/nmpp/include/nmpli/iPlessy.h	??
D:/GIT/nmpp/include/nmpli/iPlessyDetector.h	??
D:/GIT/nmpp/include/nmpli/iPrint.h	??
D:/GIT/nmpp/include/nmpli/iReordering.h	??
D:/GIT/nmpp/include/nmpli/iResample.h	??
D:/GIT/nmpp/include/nmpli/iSelect.h	??
D:/GIT/nmpp/include/nmpli/isubpixel2d.h	??
D:/GIT/nmpp/include/nmpli/isubpixel2dimpl.h	??
D:/GIT/nmpp/include/nmpli/iSupport.h	??
D:/GIT/nmpp/include/nmpli/nmpli.h	??
D:/GIT/nmpp/include/nmpli/warping.h	??
D:/GIT/nmpp/include/nmplm/mInit.h	??
D:/GIT/nmpp/include/nmplm/mInverse.h	??
D:/GIT/nmpp/include/nmplm/mMatrixVector.h	??
D:/GIT/nmpp/include/nmplm/mMatrixVectorDev.h	??
D:/GIT/nmpp/include/nmplm/mSupport.h	??
D:/GIT/nmpp/include/nmplm/nmplm.h	??
D:/GIT/nmpp/include/nmpls/fft.h	??
D:/GIT/nmpp/include/nmpls/fft_old.h	??
D:/GIT/nmpp/include/nmpls/fftext.h	??
D:/GIT/nmpp/include/nmpls/nmpls.h	??
D:/GIT/nmpp/include/nmpls/sArithmetics.h	??
D:/GIT/nmpp/include/nmpls/sCorrelation.h	??
D:/GIT/nmpp/include/nmpls/sFiltration.h	??
D:/GIT/nmpp/include/nmpls/sfir.h	??
D:/GIT/nmpp/include/nmpls/sResample.h	??
D:/GIT/nmpp/include/nmplv/nmplv.h	??
D:/GIT/nmpp/include/nmplv/nmtl.h	??
D:/GIT/nmpp/include/nmplv/vArithmetics.h	??
D:/GIT/nmpp/include/nmplv/vArithmeticsDev.h	??
D:/GIT/nmpp/include/nmplv/vBitwise.h	??
D:/GIT/nmpp/include/nmplv/vInit.h	??
D:/GIT/nmpp/include/nmplv/vInitDev.h	??
D:/GIT/nmpp/include/nmplv/vSelect.h	??
D:/GIT/nmpp/include/nmplv/vStat.h	??
D:/GIT/nmpp/include/nmplv/vSupport.h	??
D:/GIT/nmpp/include/nmplv/vTransform.h	??
D:/GIT/nmpp/include/nmtl/nmtl.h	??
D:/GIT/nmpp/include/nmtl/nmtlio.h	??
D:/GIT/nmpp/include/nmtl/tcmplx.h	??
D:/GIT/nmpp/include/nmtl/tcmplx_spec.h	??
D:/GIT/nmpp/include/nmtl/tfixpoint.h	??
D:/GIT/nmpp/include/nmtl/tfixpointmath.h	??
D:/GIT/nmpp/include/nmtl/tmatrix.h	??
D:/GIT/nmpp/include/nmtl/tmatrix_spec.h	??
D:/GIT/nmpp/include/nmtl/tnmcvec.h	??
D:/GIT/nmpp/include/nmtl/tnmint.h	??
D:/GIT/nmpp/include/nmtl/tnmmtr.h	??
D:/GIT/nmpp/include/nmtl/tnmvec.h	??
D:/GIT/nmpp/include/nmtl/tnmvecpack.h	??
D:/GIT/nmpp/include/nmtl/tvector.h	??
D:/GIT/nmpp/include/nmvc/vCore.h	??
D:/GIT/nmpp/include/rpc/aura-exports.c	??

---

D:/GIT/nmpp/include/rpc/rpc-host.h	??
D:/GIT/nmpp/include/rpc/rpc-nmc-func.h	??
D:/GIT/nmpp/include/rpc/rpc-nmc.h	??



## Глава 7

# Группы

### 7.1 DFT-8

Функция для вычисления прямого ДПФ с плавающей точкой над вектором, состоящим из 8 комплексных чисел

#### Функции

- void `nmppsDFT8Fwd_32fcr` (const `nm32fcr` \*x, `nm32fcr` \*X, `NmppsFFTSpec_32fcr` \*spec)  
Функция для вычисления прямого ДПФ с плавающей точкой над вектором, состоящим из 8 комплексных чисел

#### 7.1.1 Подробное описание

Функция для вычисления прямого ДПФ с плавающей точкой над вектором, состоящим из 8 комплексных чисел

#### 7.1.2 Функции

##### 7.1.2.1 `nmppsDFT8Fwd_32fcr()`

```
void nmppsDFT8Fwd_32fcr (  
    const nm32fcr * x,  
    nm32fcr * X,  
    NmppsFFTSpec_32fcr * spec )
```

Функция для вычисления прямого ДПФ с плавающей точкой над вектором, состоящим из 8 комплексных чисел

## Аргументы

in	x	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	---	---

## Возвращаемые значения

[out]	X выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	--

## Аргументы

in	spec	структура, содержащая необходимые коэффициенты, для вычисления прямого БПФ определенного размера
----	------	---



## 7.2 FFT-16

### Функции

- void `nmppsFFT16Fwd_32fcr` (const `nm32fcr` \*x, `nm32fcr` \*X, `NmppsFFTSpec_32fcr` \*spec)  
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 16 комплексных чисел
- int `nmppsFFT16FwdInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*addr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-16.

### 7.2.1 Подробное описание

### 7.2.2 Функции

#### 7.2.2.1 `nmppsFFT16Fwd_32fcr()`

```
void nmppsFFT16Fwd_32fcr (
    const nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fcr * spec )
```

Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 16 комплексных чисел

#### Аргументы

in	x	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	---	---

#### Возвращаемые значения

[out]	X выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	--

#### Аргументы

in	spec	структура, содержащая необходимые коэффициенты, для вычисления прямого БПФ определенного размера
----	------	--

#### 7.2.2.2 `nmppsFFT16FwdInitAlloc_32fcr()`

```
int nmppsFFT16FwdInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** addr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-16.

Аргументы

in	addr	двойной указатель на структуру коэффициентов
----	------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибки

## 7.3 FFT-32

### Функции

- void `nmppsFFT32Fwd_32fcr` (const `nm32fcr` \*x, `nm32fcr` \*X, `NmppsFFTSpec_32fcr` \*spec)  
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 32 комплексных чисел
- int `nmppsFFT32FwdInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*addr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-32.

### 7.3.1 Подробное описание

### 7.3.2 Функции

#### 7.3.2.1 `nmppsFFT32Fwd_32fcr()`

```
void nmppsFFT32Fwd_32fcr (
    const nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fcr * spec )
```

Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 32 комплексных чисел

#### Аргументы

in	x	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	---	---

#### Возвращаемые значения

[out]	X выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	--

#### Аргументы

in	spec	структура, содержащая необходимые коэффициенты, для вычисления прямого БПФ определенного размера
----	------	--

#### 7.3.2.2 `nmppsFFT32FwdInitAlloc_32fcr()`

```
int nmppsFFT32FwdInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** addr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-32.

Аргументы

in	addr	двойной указатель на структуру коэффициентов
----	------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.4 FFT-64

### Функции

- void `nmppsFFT64Fwd_32fcr` (const `nm32fcr` \*x, `nm32fcr` \*X, `NmppsFFTSpec_32fcr` \*spec)  
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 64 комплексных чисел
- int `nmppsFFT64FwdInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*addr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-64.

### 7.4.1 Подробное описание

### 7.4.2 Функции

#### 7.4.2.1 `nmppsFFT64Fwd_32fcr()`

```
void nmppsFFT64Fwd_32fcr (
    const nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fcr * spec )
```

Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 64 комплексных чисел

#### Аргументы

in	x	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	---	---

#### Возвращаемые значения

[out]	X	выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---	--

#### Аргументы

in	spec	структура, содержащая необходимые коэффициенты, для вычисления прямого БПФ определенного размера
----	------	--

#### 7.4.2.2 `nmppsFFT64FwdInitAlloc_32fcr()`

```
int nmppsFFT64FwdInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** addr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-64.

Аргументы

in	addr	двойной указатель на структуру коэффициентов
----	------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.5 FFT-128

### Функции

- void `nmppsFFT128Fwd_32fcr` (const `nm32fcr` \*x, `nm32fcr` \*X, `NmppsFFTSpec_32fcr` \*spec)  
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 128 комплексных чисел
- int `nmppsFFT128FwdInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*addr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-128.

### 7.5.1 Подробное описание

### 7.5.2 Функции

#### 7.5.2.1 `nmppsFFT128Fwd_32fcr()`

```
void nmppsFFT128Fwd_32fcr (
    const nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fcr * spec )
```

Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 128 комплексных чисел

#### Аргументы

in	x	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	---	---

#### Возвращаемые значения

[out]	X	выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---	--

#### Аргументы

in	spec	структура, содержащая необходимые коэффициенты, для вычисления прямого БПФ определенного размера
----	------	--

#### 7.5.2.2 `nmppsFFT128FwdInitAlloc_32fcr()`

```
int nmppsFFT128FwdInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** addr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-128.

Аргументы

in	addr	двойной указатель на структуру коэффициентов
----	------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок



## 7.6 FFT-256

### Функции

- void `nmppsFFT256Fwd_32fcr` (const `nm32fcr` \*x, `nm32fcr` \*X, `NmppsFFTSpec_32fcr` \*spec)  
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 256 комплексных чисел
- int `nmppsFFT256FwdInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*addr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-256.

### 7.6.1 Подробное описание

### 7.6.2 Функции

#### 7.6.2.1 `nmppsFFT256Fwd_32fcr()`

```
void nmppsFFT256Fwd_32fcr (
    const nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fcr * spec )
```

Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 256 комплексных чисел

#### Аргументы

in	x	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	---	---

#### Возвращаемые значения

[out]	X	выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---	--

#### Аргументы

in	spec	структура, содержащая необходимые коэффициенты, для вычисления прямого БПФ определенного размера
----	------	--

Для достижения максимальной производительности (1763 такта) необходимо положить входной вектор в 1-ый банк, выходной вектор в 5-ый банк

#### 7.6.2.2 `nmppsFFT256FwdInitAlloc_32fcr()`

```
int nmppsFFT256FwdInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** addr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-256.

Аргументы

in	addr	двойной указатель на структуру коэффициентов
----	------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.7 FFT-512

### Функции

- void `nmppsFFT512Fwd_32fcr` (const `nm32fcr` \*x, `nm32fcr` \*X, `NmppsFFTSpec_32fcr` \*spec)  
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 512 комплексных чисел
- int `nmppsFFT512FwdInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*addr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-512.

### 7.7.1 Подробное описание

### 7.7.2 Функции

#### 7.7.2.1 `nmppsFFT512Fwd_32fcr()`

```
void nmppsFFT512Fwd_32fcr (
    const nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fcr * spec )
```

Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 512 комплексных чисел

#### Аргументы

in	x	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	---	---

#### Возвращаемые значения

[out]	X	выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---	--

#### Аргументы

in	spec	структура, содержащая необходимые коэффициенты, для вычисления прямого БПФ определенного размера
----	------	--

Для достижения максимальной производительности (3675 такта) необходимо положить входной вектор в 2-ый банк, выходной вектор в 5-ый банк

#### 7.7.2.2 `nmppsFFT512FwdInitAlloc_32fcr()`

```
int nmppsFFT512FwdInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** addr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-512.

Аргументы

in	addr	двойной указатель на структуру коэффициентов
----	------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.8 FFT-1024

### Функции

- void `nmppsFFT1024Fwd_32fcr` (const `nm32fcr` \*x, `nm32fcr` \*X, `NmppsFFTSpec_32fcr` \*spec)  
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 1024 комплексных чисел
- int `nmppsFFT1024FwdInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*addr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-1024.

### 7.8.1 Подробное описание

### 7.8.2 Функции

#### 7.8.2.1 `nmppsFFT1024Fwd_32fcr()`

```
void nmppsFFT1024Fwd_32fcr (
    const nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fcr * spec )
```

Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 1024 комплексных чисел

#### Аргументы

in	x	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	---	---

#### Возвращаемые значения

[out]	X	выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---	--

#### Аргументы

in	spec	структура, содержащая необходимые коэффициенты, для вычисления прямого БПФ определенного размера
----	------	--

Для достижения максимальной производительности (8655 такта) необходимо положить входной вектор в 6-ый банк, выходной вектор в 5-ый банк

#### 7.8.2.2 `nmppsFFT1024FwdInitAlloc_32fcr()`

```
int nmppsFFT1024FwdInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** addr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-1024.

Аргументы

in	addr	двойной указатель на структуру коэффициентов
----	------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.9 FFT-2048

### Функции

- void `nmppsFFT2048Fwd_32fcr` (const `nm32fcr` \*x, `nm32fcr` \*X, `NmppsFFTSpec_32fcr` \*spec)  
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 2048 комплексных чисел
- int `nmppsFFT2048FwdInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*addr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-2048.

### 7.9.1 Подробное описание

### 7.9.2 Функции

#### 7.9.2.1 `nmppsFFT2048Fwd_32fcr()`

```
void nmppsFFT2048Fwd_32fcr (
    const nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fcr * spec )
```

Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 2048 комплексных чисел

#### Аргументы

in	x	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	---	---

#### Возвращаемые значения

[out]	X	выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---	--

#### Аргументы

in	spec	структура, содержащая необходимые коэффициенты, для вычисления прямого БПФ определенного размера
----	------	--

Для достижения максимальной производительности (19504 такта) необходимо положить входной вектор в 5-ый банк, выходной вектор в 5-ый банк

#### 7.9.2.2 `nmppsFFT2048FwdInitAlloc_32fcr()`

```
int nmppsFFT2048FwdInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** addr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-2048.

Аргументы

in	addr	двойной указатель на структуру коэффициентов
----	------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок



## 7.10 FFT-4096

### Функции

- void `nmppsFFT4096Fwd_32fcr` (const `nm32fcr` \*x, `nm32fcr` \*X, `NmppsFFTSpec_32fcr` \*spec)  
Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 4096 комплексных чисел
- int `nmppsFFT4096FwdInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*addr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-4096.

### 7.10.1 Подробное описание

### 7.10.2 Функции

#### 7.10.2.1 `nmppsFFT4096Fwd_32fcr()`

```
void nmppsFFT4096Fwd_32fcr (
    const nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fcr * spec )
```

Функция для вычисления прямого БПФ с плавающей точкой над вектором, состоящим из 4096 комплексных чисел

#### Аргументы

in	x	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	---	---

#### Возвращаемые значения

[out]	X	выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---	--

#### Аргументы

in	spec	структура, содержащая необходимые коэффициенты, для вычисления прямого БПФ определенного размера
----	------	--

Для достижения максимальной производительности (54258 такта) необходимо положить входной вектор в 5-ый банк, выходной вектор в 5-ый банк

#### 7.10.2.2 `nmppsFFT4096FwdInitAlloc_32fcr()`

```
int nmppsFFT4096FwdInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** addr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ-4096.

Аргументы

in	addr	двойной указатель на структуру коэффициентов
----	------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.11 IDFT-8

### Функции

- void `nmppsDFT8Inv_32fcr` (const `nm32fcr` \*ix, `nm32fcr` \*iX, `NmppsFFTSpec_32fcr` \*ispec)

Функция для вычисления обратного ДПФ с плавающей точкой над вектором, состоящим из 8 комплексных чисел

#### 7.11.1 Подробное описание

#### 7.11.2 Функции

##### 7.11.2.1 `nmppsDFT8Inv_32fcr()`

```
void nmppsDFT8Inv_32fcr (
    const nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fcr * ispec )
```

Функция для вычисления обратного ДПФ с плавающей точкой над вектором, состоящим из 8 комплексных чисел

#### Аргументы

in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	----	---

#### Возвращаемые значения

[out]	iX выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---

#### Аргументы

in	ispec	структура, содержащая необходимые коэффициенты, для вычисления обратного БПФ определенного размера
----	-------	--

## 7.12 IFFT-16

### Функции

- void `nmppsFFT16Inv_32fcr` (const `nm32fcr` \*ix, `nm32fcr` \*iX, `NmppsFFTSpec_32fcr` \*ispec)  
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 16 комплексных чисел
- int `nmppsFFT16InvInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*iaddr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-16.

### 7.12.1 Подробное описание

### 7.12.2 Функции

#### 7.12.2.1 `nmppsFFT16Inv_32fcr()`

```
void nmppsFFT16Inv_32fcr (
    const nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fcr * ispec )
```

Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 16 комплексных чисел

#### Аргументы

in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	----	---

#### Возвращаемые значения

[out]	iX выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---

#### Аргументы

in	ispec	структура, содержащая необходимые коэффициенты, для вычисления обратного БПФ определенного размера
----	-------	--

#### 7.12.2.2 `nmppsFFT16InvInitAlloc_32fcr()`

```
int nmppsFFT16InvInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** iaddr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-16.

Аргументы

in	iaddr	двойной указатель на структуру коэффициентов
----	-------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.13 IFFT-32

### Функции

- void `nmppsFFT32Inv_32fcr` (const `nm32fcr` \*ix, `nm32fcr` \*iX, `NmppsFFTSpec_32fcr` \*ispec)  
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 32 комплексных чисел
- int `nmppsFFT32InvInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*iaddr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-32.

### 7.13.1 Подробное описание

### 7.13.2 Функции

#### 7.13.2.1 `nmppsFFT32Inv_32fcr()`

```
void nmppsFFT32Inv_32fcr (
    const nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fcr * ispec )
```

Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 32 комплексных чисел

#### Аргументы

in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	----	---

#### Возвращаемые значения

[out]	iX выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---

#### Аргументы

in	ispec	структура, содержащая необходимые коэффициенты, для вычисления обратного БПФ определенного размера
----	-------	--

#### 7.13.2.2 `nmppsFFT32InvInitAlloc_32fcr()`

```
int nmppsFFT32InvInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** iaddr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-32.

Аргументы

in	iaddr	двойной указатель на структуру коэффициентов
----	-------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.14 IFFT-64

### Функции

- void `nmppsFFT64Inv_32fcr` (const `nm32fcr` \*ix, `nm32fcr` \*iX, `NmppsFFTSpec_32fcr` \*ispec)  
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 64 комплексных чисел
- int `nmppsFFT64InvInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*iaddr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-64.

### 7.14.1 Подробное описание

### 7.14.2 Функции

#### 7.14.2.1 `nmppsFFT64Inv_32fcr()`

```
void nmppsFFT64Inv_32fcr (
    const nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fcr * ispec )
```

Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 64 комплексных чисел

#### Аргументы

in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	----	---

#### Возвращаемые значения

[out]	iX выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---

#### Аргументы

in	ispec	структура, содержащая необходимые коэффициенты, для вычисления обратного БПФ определенного размера
----	-------	--

#### 7.14.2.2 `nmppsFFT64InvInitAlloc_32fcr()`

```
int nmppsFFT64InvInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** iaddr )
```



Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-64.

Аргументы

in	iaddr	двойной указатель на структуру коэффициентов
----	-------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.15 IFFT-128

### Функции

- void `nmppsFFT128Inv_32fcr` (const `nm32fcr` \*ix, `nm32fcr` \*iX, `NmppsFFTSpec_32fcr` \*ispec)  
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 128 комплексных чисел
- int `nmppsFFT128InvInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*iaddr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-128.

### 7.15.1 Подробное описание

### 7.15.2 Функции

#### 7.15.2.1 `nmppsFFT128Inv_32fcr()`

```
void nmppsFFT128Inv_32fcr (
    const nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fcr * ispec )
```

Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 128 комплексных чисел

#### Аргументы

in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	----	---

#### Возвращаемые значения

[out]	iX выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---

#### Аргументы

in	ispec	структура, содержащая необходимые коэффициенты, для вычисления обратного БПФ определенного размера
----	-------	--

#### 7.15.2.2 `nmppsFFT128InvInitAlloc_32fcr()`

```
int nmppsFFT128InvInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** iaddr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-128.

Аргументы

in	iaddr	двойной указатель на структуру коэффициентов
----	-------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.16 IFFT-256

### Функции

- void `nmppsFFT256Inv_32fcr` (const `nm32fcr` \*ix, `nm32fcr` \*iX, `NmppsFFTSpec_32fcr` \*ispec)  
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 256 комплексных чисел
- int `nmppsFFT256InvInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*iaddr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-256.

### 7.16.1 Подробное описание

### 7.16.2 Функции

#### 7.16.2.1 `nmppsFFT256Inv_32fcr()`

```
void nmppsFFT256Inv_32fcr (
    const nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fcr * ispec )
```

Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 256 комплексных чисел

#### Аргументы

in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	----	---

#### Возвращаемые значения

[out]	iX выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---

#### Аргументы

in	ispec	структура, содержащая необходимые коэффициенты, для вычисления обратного БПФ определенного размера
----	-------	--

Для достижения максимальной производительности (1763 такта) необходимо положить входной вектор в 1-ый банк, выходной вектор в 5-ый банк

#### 7.16.2.2 `nmppsFFT256InvInitAlloc_32fcr()`

```
int nmppsFFT256InvInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** iaddr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-256.

Аргументы

in	iaddr	двойной указатель на структуру коэффициентов
----	-------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.17 IFFT-512

### Функции

- void `nmppsFFT512Inv_32fcr` (const `nm32fcr` \*ix, `nm32fcr` \*iX, `NmppsFFTSpec_32fcr` \*ispec)  
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 512 комплексных чисел
- int `nmppsFFT512InvInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*iaddr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-512.

### 7.17.1 Подробное описание

### 7.17.2 Функции

#### 7.17.2.1 `nmppsFFT512Inv_32fcr()`

```
void nmppsFFT512Inv_32fcr (
    const nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fcr * ispec )
```

Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 512 комплексных чисел

#### Аргументы

in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	----	---

#### Возвращаемые значения

[out]	iX выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---

#### Аргументы

in	ispec	структура, содержащая необходимые коэффициенты, для вычисления обратного БПФ определенного размера
----	-------	--

Для достижения максимальной производительности (3675 такта) необходимо положить входной вектор в 2-ый банк, выходной вектор в 5-ый банк

#### 7.17.2.2 `nmppsFFT512InvInitAlloc_32fcr()`

```
int nmppsFFT512InvInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** iaddr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-512.

Аргументы

in	iaddr	двойной указатель на структуру коэффициентов
----	-------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.18 IFFT-1024

### Функции

- void `nmppsFFT1024Inv_32fcr` (const `nm32fcr` \*ix, `nm32fcr` \*iX, `NmppsFFTSpec_32fcr` \*ispec)  
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 1024 комплексных чисел
- int `nmppsFFT1024InvInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*iaddr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-1024.

### 7.18.1 Подробное описание

### 7.18.2 Функции

#### 7.18.2.1 `nmppsFFT1024Inv_32fcr()`

```
void nmppsFFT1024Inv_32fcr (
    const nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fcr * ispec )
```

Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 1024 комплексных чисел

#### Аргументы

in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	----	---

#### Возвращаемые значения

[out]	iX выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---

#### Аргументы

in	ispec	структура, содержащая необходимые коэффициенты, для вычисления обратного БПФ определенного размера
----	-------	--

Для достижения максимальной производительности (8655 такта) необходимо положить входной вектор в 6-ый банк, выходной вектор в 5-ый банк

#### 7.18.2.2 `nmppsFFT1024InvInitAlloc_32fcr()`

```
int nmppsFFT1024InvInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** iaddr )
```



Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-1024.

Аргументы

in	iaddr	двойной указатель на структуру коэффициентов
----	-------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.19 IFFT-2048

### Функции

- void `nmppsFFT2048Inv_32fcr` (const `nm32fcr` \*ix, `nm32fcr` \*iX, `NmppsFFTSpec_32fcr` \*ispec)  
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 2048 комплексных чисел
- int `nmppsFFT2048InvInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*iaddr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-2048.

### 7.19.1 Подробное описание

### 7.19.2 Функции

#### 7.19.2.1 `nmppsFFT2048Inv_32fcr()`

```
void nmppsFFT2048Inv_32fcr (
    const nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fcr * ispec )
```

Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 2048 комплексных чисел

#### Аргументы

in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	----	---

#### Возвращаемые значения

[out]	iX выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---

#### Аргументы

in	ispec	структура, содержащая необходимые коэффициенты, для вычисления обратного БПФ определенного размера
----	-------	--

Для достижения максимальной производительности (19504 такта) необходимо положить входной вектор в 5-ый банк, выходной вектор в 5-ый банк

#### 7.19.2.2 `nmppsFFT2048InvInitAlloc_32fcr()`

```
int nmppsFFT2048InvInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** iaddr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-2048.

Аргументы

in	iaddr	двойной указатель на структуру коэффициентов
----	-------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

## 7.20 IFFT-4096

### Функции

- void `nmppsFFT4096Inv_32fcr` (const `nm32fcr` \*ix, `nm32fcr` \*iX, `NmppsFFTSpec_32fcr` \*ispec)  
Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 4096 комплексных чисел
- int `nmppsFFT4096InvInitAlloc_32fcr` (`NmppsFFTSpec_32fcr` \*\*iaddr)  
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-4096.

### 7.20.1 Подробное описание

### 7.20.2 Функции

#### 7.20.2.1 nmppsFFT4096Inv\_32fcr()

```
void nmppsFFT4096Inv_32fcr (
    const nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fcr * ispec )
```

Функция для вычисления обратного БПФ с плавающей точкой над вектором, состоящим из 4096 комплексных чисел

#### Аргументы

in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	----	---

#### Возвращаемые значения

[out]	iX выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---

#### Аргументы

in	ispec	структура, содержащая необходимые коэффициенты, для вычисления обратного БПФ определенного размера
----	-------	--

#### 7.20.2.2 nmppsFFT4096InvInitAlloc\_32fcr()

```
int nmppsFFT4096InvInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** iaddr )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ-4096.

Аргументы

in	iaddr	двойной указатель на структуру коэффициентов
----	-------	--

Возвращает

Функция возвращают 0 в случае успешной инициализации и число, отличное от нуля, в случае ошибок

Для достижения максимальной производительности (54258 такта) необходимо положить входной вектор в 5-ый банк, выходной вектор в 5-ый банк

## 7.21 FFT-Common

### Функции

- `int nmppsFFTFwd_32fcr (const nm32fcr *x, nm32fcr *X, NmppsFFTSpec_32fcr *Spec)`  
Функция для вычисления прямого БПФ с плавающей точкой над вектором длины от 8 до 2048.
- `int nmppsFFTFwdInitAlloc_32fcr (NmppsFFTSpec_32fcr **Spec, int Order)`  
Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ с плавающей точкой над вектором длины от 8 до 2048.

### 7.21.1 Подробное описание

### 7.21.2 Функции

#### 7.21.2.1 nmppsFFTFwd\_32fcr()

```
int nmppsFFTFwd_32fcr (
    const nm32fcr * x,
    nm32fcr * X,
    NmppsFFTSpec_32fcr * Spec )
```

Функция для вычисления прямого БПФ с плавающей точкой над вектором длины от 8 до 2048.

#### Аргументы

in	x	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	---	---

#### Возвращаемые значения

[out]	X	выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---	--

#### Аргументы

in	спец	структра, содержащая необходимые коэффициенты, для вычисления прямого БПФ определенного размера
----	------	---

#### 7.21.2.2 nmppsFFTFwdInitAlloc\_32fcr()

```
int nmppsFFTFwdInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** Spec,
    int Order )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления прямого БПФ с плавающей точкой над вектором длины от 8 до 2048.

## Аргументы

in	Spec	двойной указатель на структуру коэффициентов
in	Order	размерность БПФ, которое нужно вычислить, например, для БПФ256 этот параметр равен 8 (т.к. $2^8 = 256$ )

## Возвращает

Функция возвращают 0 в случае успешной инициализации и отрицательное число (от -1 и меньше) в случае ошибок



## 7.22 IFFT-Common

### Функции

- `int nmppsFFTInv_32fcr (const nm32fcr *ix, nm32fcr *iX, NmppsFFTSpec_32fcr *iSpec)`  
Функция для вычисления обратного БПФ с плавающей точкой над вектором длины от 8 до 2048.
- `int nmppsFFTInvInitAlloc_32fcr (NmppsFFTSpec_32fcr **iSpec, int iOrder)`  
Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ с плавающей точкой над вектором длины от 8 до 2048.

### 7.22.1 Подробное описание

### 7.22.2 Функции

#### 7.22.2.1 nmppsFFTInv\_32fcr()

```
int nmppsFFTInv_32fcr (
    const nm32fcr * ix,
    nm32fcr * iX,
    NmppsFFTSpec_32fcr * iSpec )
```

Функция для вычисления обратного БПФ с плавающей точкой над вектором длины от 8 до 2048.

#### Аргументы

in	ix	входной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
----	----	---

#### Возвращаемые значения

[out]	iX выходной вектор комплексных чисел (на мнимая и действительная части имеют тип float)
-------	---

#### Аргументы

in	ispec	структра, содержащая необходимые коэффициенты, для вычисления обратного БПФ определенного размера
----	-------	---

#### 7.22.2.2 nmppsFFTInvInitAlloc\_32fcr()

```
int nmppsFFTInvInitAlloc_32fcr (
    NmppsFFTSpec_32fcr ** iSpec,
    int iOrder )
```

Функция инициализации структуры коэффициентов, необходимых для вычисления обратного БПФ с плавающей точкой над вектором длины от 8 до 2048.

## Аргументы

in	iSpec	двойной указатель на структуру коэффициентов
in	iOrder	размерность БПФ, которое нужно вычислить, например, для БПФ256 этот параметр равен 8 (т.к. $2^8 = 256$ )

## Возвращает

Функция возвращают 0 в случае успешной инициализации и отрицательное число (от -1 и меньше) в случае ошибок

## 7.23 nmppsMalloc

Распределение памяти для векторов библиотеки.

### Функции

- `void * nmppsMalloc32 (unsigned sizeInt32)`
- `nm64s * nmppsMalloc_64s (unsigned nSize)`
- `nm1 * nmppsMalloc_1 (unsigned nSize)`
- `nm2s * nmppsMalloc_2s (unsigned nSize)`
- `nm2u * nmppsMalloc_2u (unsigned nSize)`
- `nm4s * nmppsMalloc_4s (unsigned nSize)`
- `nm4u * nmppsMalloc_4u (unsigned nSize)`
- `nm8u * nmppsMalloc_8u (unsigned nSize)`
- `nm8s * nmppsMalloc_8s (unsigned nSize)`
- `nm16u * nmppsMalloc_16u (unsigned nSize)`
- `nm16s * nmppsMalloc_16s (unsigned nSize)`
- `nm32u * nmppsMalloc_32u (unsigned nSize)`
- `nm32s * nmppsMalloc_32s (unsigned nSize)`
- `nm64u * nmppsMalloc_64u (unsigned nSize)`
- `nm32sc * nmppsMalloc_32sc (unsigned sizeCmplxInt32)`
- `nm32fc * nmppsMalloc_32fc (unsigned sizeCmplxFloat)`
- `nm32fcr * nmppsMalloc_32fcr (unsigned sizeCmplxFloat)`
- `float * nmppsMalloc_32f (unsigned sizeFloat)`
- `double * nmppsMalloc_64f (unsigned sizeDouble)`

### 7.23.1 Подробное описание

Распределение памяти для векторов библиотеки.

#### Аргументы

nSize	Число элементов в векторе.
hint	Номер банка памяти. Может принимать значения MEM_LOCAL, MEM_GLOBAL.

#### Заметки

Память, распределенная с помощью функций `nmppsMalloc_` должна освобождаться с помощью функции `nmppsFree()`.

## 7.24 nmppsFree

Освобождение памяти для векторов.

Функции

- void nmppsFree (void \*ptr)

### 7.24.1 Подробное описание

Освобождение памяти для векторов.

Заметки

Данная функция должна вызываться только для векторов, распределенных с помощью функций nmppsMalloc\_.

## 7.25 BLASS-LEVEL1

### Перечисления

- enum `nm_trans` { `nm_n` =0, `nm_t` =1 }  
Brief description.

### Функции

- double `nmblas_dasum` (const int `n`, const double \*`x`, const int `inc_x`)  
Brief description.

#### 7.25.1 Подробное описание

#### 7.25.2 Перечисления

##### 7.25.2.1 `nm_trans`

enum `nm_trans`

Brief description.

#### Аргументы

in	<code>n</code>	Description for <code>n</code>
in	<code>alpha</code>	Description for <code>alpha</code>
in	<code>arr↔ _x</code>	Description for <code>arr_x</code>
in	<code>inc↔ _x</code>	Description for <code>inc_x</code>
in	<code>arr↔ _y</code>	Description for <code>arr_y</code>
in	<code>inc↔ _y</code>	Description for <code>inc_y</code>

#### Возвращает

Return description

#### More details

См. определение в файле `nmblas.h` строка 61

#### 7.25.3 Функции

## 7.25.3.1 nmblas\_dasum()

```
double nmblas_dasum (
    const int n,
    const double * x,
    const int inc_x )
```

Brief description.

Аргументы

in	n	Description for n
in,out	x	Description for x
in	inc $\leftrightarrow$ _x	Description for inc_x

Возвращает

Return description

More details

## 7.26 nmppcDivC

частное двух комплексных чисел

Функции

- void nmppcDivC (nm64sc \*pnSrcA, nm64s \*pnSrcB, nm64sc \*Dst)

### 7.26.1 Подробное описание

частное двух комплексных чисел

<>

Аргументы

*pnSrcA	указатель на делимое. указатель на делитель. указатель на частное.
---------	--



## 7.27 nmppcProdC

произведение двух комплексных чисел.

### Функции

- void nmppcProdC (nm64sc \*pnSrcA, nm64sc \*pnSrcB, nm64sc \*Dst)

### 7.27.1 Подробное описание

произведение двух комплексных чисел.

#### Аргументы

*pnSrcA	указатель на первый множитель. указатель на второй множитель. указатель на произведение.
---------	--

## 7.28 nmppsFixExp32

Вычисление вычисления экспоненты числа в формате fixed-point (16.16)

Функции

- int nmppsFixExp32 (int nVal)

### 7.28.1 Подробное описание

Вычисление вычисления экспоненты числа в формате fixed-point (16.16)

Аргументы

nVal	Входное число с фиксированной точкой в формате (16.16)
------	--

Возвращает

Экспонента числа в формате с фиксированной точкой (16.16)

## 7.29 nmppcFixSinCos32

Вычисление синуса и косинуса от аргумента в формате fixed-point (16.16)

Функции

- void nmppcFixSinCos32 (int nArg, int \*pnSin, int \*pnCos)

### 7.29.1 Подробное описание

Вычисление синуса и косинуса от аргумента в формате fixed-point (16.16)

Аргументы

nArg	Угол в радианах. Угол должен быть в диапазоне от $-\pi/2$ до $+\pi/2$
------	---

Возвращаемые значения

pnSin	указатель на синус
pnCos	указатель на косинус

## 7.30 nmppsFixArcTan32

Вычисление арктангенса от аргумента в формате fixed-point (16.16)

Функции

- `int nmppsFixArcTan32 (int nArg)`

### 7.30.1 Подробное описание

Вычисление арктангенса от аргумента в формате fixed-point (16.16)

Аргументы

nArg	Угол в радианах
------	-----------------

Возвращает

Арктангенс

## 7.31 nmppcDoubleToFix32

Функция перевода из Fixed-Point (16.16) в Double.

Функции

- `int nmppcDoubleToFix32 (double arg)`

### 7.31.1 Подробное описание

Функция перевода из Fixed-Point (16.16) в Double.

Аргументы

<code>arg</code>	Входное число с плавающей точкой
<code>fixpoint</code>	позиция двоичной точки

Возвращает

Число с фиксированной точкой

## 7.32 nmppcFix32ToDouble

Преобразование 32р. числа с фиксированной точкой (16.16) в число с плавающей точкой типа double.

### Функции

- double nmppcFix32ToDouble (int arg)

### 7.32.1 Подробное описание

Преобразование 32р. числа с фиксированной точкой (16.16) в число с плавающей точкой типа double.

#### Аргументы

arg	Входное 32р. число в формате с фиксированной точкой (16.16)
-----	---

#### Возвращает

Число с плавающей точкой

## 7.33 nmppcFixSqrt32

Вычисление квадратного корня числа в формате fixed-point (16.16)

Функции

- unsigned int nmppcFixSqrt32 (unsigned int nVal)

### 7.33.1 Подробное описание

Вычисление квадратного корня числа в формате fixed-point (16.16)

Аргументы

nVal	Входное число с фиксированной точкой в формате (16.16)
------	--

Возвращает

Квадратный корень в формате с фиксированной точкой (16.16)

## 7.34 nmppcFixMul32

Вычисление произведения двух числе в формате fixed-point (16.16)

Функции

- int nmppcFixMul32 (int nX, int nY)
- int nmppcFixDiv32 (int nX, int nY)

### 7.34.1 Подробное описание

Вычисление произведения двух числе в формате fixed-point (16.16)

Деление двух целых чисел с записью результата в формате fixed-point (16.16)

Аргументы

nX	Первое входное число с фиксированной точкой в формате (16.16)
nY	Второе входное число с фиксированной точкой в формате (16.16)

Возвращает

Произведение в формате с фиксированной точкой (16.16)

Аргументы

nX	Делимое
nY	Делитель

Возвращает

Частное от деления в формате с фиксированной точкой (16.16)



## 7.35 nmppcFixInv32

Вычисление обратного значения целого числа с записью результата в формате fixed-point.

Функции

- int nmppcFixInv32 (int nVal, int nFixpoint)

### 7.35.1 Подробное описание

Вычисление обратного значения целого числа с записью результата в формате fixed-point.

$$Res = 2^n Fixpoint / nVal$$

\param nVal

Делитель

Аргументы

nFixpoint	Позиция бинарной точки в результирующем слове
-----------	---

Возвращает

Частное от деления нав формате с фиксированной точкой (16.16)

## 7.36 nmppcTblFixArcSin32

Вычисление функции  $\arcsin$  по таблице. Входные и выходные значения задаются в формате fixed-point (16.16)

### Функции

- `int nmppcTblFixArcSin32 (int nArg)`

#### 7.36.1 Подробное описание

Вычисление функции  $\arcsin$  по таблице. Входные и выходные значения задаются в формате fixed-point (16.16)

#### Аргументы

nArg	Входное значение.
------	-------------------

#### Возвращает

Угол в диапазоне от  $-\pi/2$  до  $+\pi/2$  в формате fixed-point (16.16)

## 7.37 nmppcTblFixArcCos32

Вычисление функции `arccos` по таблице. Входные и выходные значения задаются в формате `fixed-point (16.16)`

Функции

- `int nmppcTblFixArcCos32 (int nArg)`

### 7.37.1 Подробное описание

Вычисление функции `arccos` по таблице. Входные и выходные значения задаются в формате `fixed-point (16.16)`

Аргументы

nArg	Входное значение.
------	-------------------

Возвращает

Угол в диапазоне от 0 до  $\pi$  в формате `fixed-point (16.16)`

## 7.38 nmppcTblFixCos32

Вычисление функции  $\cos$  по таблице. Входные и выходные значения задаются в формате fixed-point (16.16)

Функции

- `int nmppcTblFixCos32 (int nArg)`

### 7.38.1 Подробное описание

Вычисление функции  $\cos$  по таблице. Входные и выходные значения задаются в формате fixed-point (16.16)

Аргументы

nArg	Угол в диапазоне от 0 до $\pi$ в формате fixed-point (16.16)
------	--

Возвращает

значение  $\cos$  в формате fixed-point (16.16)

## 7.39 nmppcTblFixSin32

Вычисление функции  $\sin$  по таблице. Входные и выходные значения задаются в формате fixed-point (16.16)

Функции

- `int nmppcTblFixSin32 (int nArg)`

### 7.39.1 Подробное описание

Вычисление функции  $\sin$  по таблице. Входные и выходные значения задаются в формате fixed-point (16.16)

Аргументы

nArg	Угол в диапазоне от $-\pi/2$ до $+\pi/2$ в формате fixed-point (16.16)
------	--

Возвращает

значение  $\sin$  в формате fixed-point (16.16)

## 7.40 nmppcFixDivMod32

Вычисление частного и остатка при делении чисел с фиксированной запятой в формате fixed-point (16.16)

### Функции

- void nmppcFixDivMod32 (int nDividend, int nDivisor, int \*pnQuotient, int \*pnReminder)
- void nmppcFixDivPosMod32 (unsigned int nDividend, unsigned int nDivisor, int \*pnQuotient, int \*pnReminder)

### 7.40.1 Подробное описание

Вычисление частного и остатка при делении чисел с фиксированной запятой в формате fixed-point (16.16)

#### Аргументы

nDividend	Делимое в формате fixed-point (16.16)
nDivisor	Делитель в формате fixed-point (16.16)

#### Возвращаемые значения

pnQuotient	Частное от деления в формате fixed-point (16.16)
pnReminder	Остаток от деления в формате fixed-point (16.16)

## 7.41 nmppcFixSqrt64

Вычисление квадратного корня числа в формате fixed-point (32.32)

Функции

- unsigned long nmppcFixSqrt64 (unsigned long x)

### 7.41.1 Подробное описание

Вычисление квадратного корня числа в формате fixed-point (32.32)

Аргументы

x	Входное число с фиксированной точкой в формате (32.32)
---	--

Возвращает

Квадратный корень в формате с фиксированной точкой (32.32)

## 7.42 nmppcDoubleToFix64

Функция перевода из Fixed-Point 64 в Double.

Функции

- long nmppcDoubleToFix64 (double arg, int fixpoint)

### 7.42.1 Подробное описание

Функция перевода из Fixed-Point 64 в Double.

Аргументы

arg	Входное число с плавающей точкой
fixpoint	позиция двоичной точки

Возвращает

Число с фиксированной точкой



## 7.43 nmppcFix64ToDouble

Преобразование 64р. числа с фиксированной точкой в число с плавающей точкой типа double.

### Функции

- double nmppcFix64ToDouble (long arg, int fixpoint)

#### 7.43.1 Подробное описание

Преобразование 64р. числа с фиксированной точкой в число с плавающей точкой типа double.

#### Аргументы

arg	Входное 64р. число в формате с фиксированной точкой
fixpoint	Позиция двоичной точки

#### Возвращает

Число с плавающей точкой

## 7.44 nmppcFixDiv64

Деление двух целых чисел с записью результата в формате fixed-point.

### Функции

- void nmppcFixDiv64 (long \*nDividend, long \*nDivisor, int nFixpoint, long \*nQuotient)

#### 7.44.1 Подробное описание

Деление двух целых чисел с записью результата в формате fixed-point.

##### Аргументы

nDividend	Делимое
nDivisor	Делитель. Делитель должен быть по модулю больше чем делимое.
nFixpoint	Позиция двоичной точки

##### Возвращаемые значения

nQuotient	Частное в формате числа с фиксированной точкой
-----------	--

## 7.45 nmppcFixSinCos64

Вычисление синуса и косинуса от аргумента в формате fixed-point (32.32)

Функции

- void nmppcFixSinCos64 (long nArg, long \*pnSin, long \*pnCos)

### 7.45.1 Подробное описание

Вычисление синуса и косинуса от аргумента в формате fixed-point (32.32)

Аргументы

nArg	Угол в радианах. Угол должен быть в диапазоне от $-\pi/2$ до $+\pi/2$
------	---

Возвращаемые значения

pnSin	указатель на синус
pnCos	указатель на косинус

## 7.46 nmppsFixArcTan64

Вычисление арктангенса от аргумента в формате fixed-point (32.32)

### Функции

- long nmppsFixArcTan64 (long nArg)

#### 7.46.1 Подробное описание

Вычисление арктангенса от аргумента в формате fixed-point (32.32)

#### Аргументы

nArg	Угол в радианах
------	-----------------

#### Арктангенс

## 7.47 nmppcFix64Exp01

Вычисление вычисления экспоненты числа в формате fixed-point (4.60)

Функции

- long nmppcFix64Exp01 (long nArg)

### 7.47.1 Подробное описание

Вычисление вычисления экспоненты числа в формате fixed-point (4.60)

Аргументы

nVal	Входное число с фиксированной точкой в формате (4.60)
------	---

Возвращает

Экспонента числа в формате с фиксированной точкой (4.60)

## 7.48 nmppsRand

Генерация случайного числа с равномерным распределением.

### Функции

- int nmppcRandMinMaxDiv (int nMin, int nMax, int nDivisible)
- int nmppcRandMinMax (int nMin, int nMax)
- int nmppcRand ()

### 7.48.1 Подробное описание

Генерация случайного числа с равномерным распределением.

#### Аргументы

nMin	Минимальное возможное значение случайного числа.
nMax	Максимальное возможное значение случайного числа.
nDivisible	Значение, которому будет кратно случайное число.

#### Возвращает

int Случайное число в диапазоне либо [nMin, nMax]. Для функции без параметров данный диапазон  $[-2^{31}; 2^{31}-1]$ .

## 7.49 nmppcSqrt

Вычисление квадратного корня

Функции

- unsigned int nmppcSqrt\_64u (unsigned long long x)

### 7.49.1 Подробное описание

Вычисление квадратного корня

Аргументы

x	Входное число
---	---------------

Возвращает

Квадратный корень

## 7.50 Инициализация

### Группы

- [nmppsRand](#)

Генерация случайного числа с равномерным распределением.

#### 7.50.1 Подробное описание



## 7.51 Integer operations

### Группы

- [nmppsSqrt](#)

Вычисление квадратного корня

#### 7.51.1 Подробное описание

## 7.52 Fix-point 64

### Группы

- [nmppcFixSqrt64](#)  
Вычисление квадратного корня числа в формате fixed-point (32.32)
- [nmppcDoubleToFix64](#)  
Функция перевода из Fixed-Point 64 в Double.
- [nmppcFix64ToDouble](#)  
Преобразование 64p. числа с фиксированной точкой в число с плавающей точкой типа double.
- [nmppcFixDiv64](#)  
Деление двух целых чисел с записью результата в формате fixed-point.
- [nmppcFixSinCos64](#)  
Вычисление синуса и косинуса от аргумента в формате fixed-point (32.32)
- [nmppcFixArcTan64](#)  
Вычисление арктангенса от аргумента в формате fixed-point (32.32)

#### 7.52.1 Подробное описание

## 7.53 Fix-point 32

### Группы

- [nmppcFixExp32](#)  
Вычисление вычисления экспоненты числа в формате fixed-point (16.16)
- [nmppcFixSinCos32](#)  
Вычисление синуса и косинуса от аргумента в формате fixed-point (16.16)
- [nmppcFixArcTan32](#)  
Вычисление арктангенса от аргумента в формате fixed-point (16.16)
- [nmppcDoubleToFix32](#)  
Функция перевода из Fixed-Point (16.16) в Double.
- [nmppcFix32ToDouble](#)  
Преобразование 32р. числа с фиксированной точкой (16.16) в число с плавающей точкой типа double.
- [nmppcFixSqrt32](#)  
Вычисление квадратного корня числа в формате fixed-point (16.16)
- [nmppcFixMul32](#)  
Вычисление произведения двух числе в формате fixed-point (16.16)
- [nmppcFixInv32](#)  
Вычисление обратного значения целого числа с записью результата в формате fixed-point.
- [nmppcTblFixArcSin32](#)  
Вычисление функции arcsin по таблице. Входные и выходные значения задаются в формате fixed-point (16.16)
- [nmppcTblFixArcCos32](#)  
Вычисление функции arccos по таблице. Входные и выходные значения задаются в формате fixed-point (16.16)
- [nmppcTblFixCos32](#)  
Вычисление функции cos по таблице. Входные и выходные значения задаются в формате fixed-point (16.16)
- [nmppcTblFixSin32](#)  
Вычисление функции sin по таблице. Входные и выходные значения задаются в формате fixed-point (16.16)
- [nmppcFixDivMod32](#)  
Вычисление частного и остатка при делении чисел с фиксированной запятой в формате fixed-point (16.16)
- [nmppcFix64Exp01](#)  
Вычисление вычисления экспоненты числа в формате fixed-point (4.60)

#### 7.53.1 Подробное описание

## 7.54 Арифметические операции

### Группы

- [nmprcDivC](#)  
частное двух комплексных чисел
- [nmprcProdC](#)  
произведение двух комплексных чисел.

#### 7.54.1 Подробное описание

## 7.55 Функции деинтерлейсинга

### Функции

- void **IMG\_DeinterlaceSplit** (nm8u \*pSrcImg, int nSrcWidth, int nSrcHeight, nm8u \*pDstEven, nm8u \*pDstOdd)
- void **IMG\_DeinterlaceBlend** (nm8u \*pSrcEven, nm8u \*pSrcOdd, int nSrcWidth, int nSrcHeight, nm8u \*pDst)

### 7.55.1 Подробное описание

### 7.55.2 Функции

#### 7.55.2.1 IMG\_DeinterlaceBlend()

```
void IMG_DeinterlaceBlend (  
    nm8u * pSrcEven,  
    nm8u * pSrcOdd,  
    int nSrcWidth,  
    int nSrcHeight,  
    nm8u * pDst )
```

Функция объединяет два полукдра в один полный кадр.

#### Аргументы

pSrcEven	указатель на четный полукادر.
nSrcOdd	указатель на нечетный полукادر.
nSrcWidth	ширина исходных полукадров в пикселях.
pSrcHeight	высота исходных полукадров в пикселях.
pDst	указатель на буфер результирующего кадр

#### Возвращает

void

#### 7.55.2.2 IMG\_DeinterlaceSplit()

```
void IMG_DeinterlaceSplit (  
    nm8u * pSrcImg,  
    int nSrcWidth,  
    int nSrcHeight,  
    nm8u * pDstEven,  
    nm8u * pDstOdd )
```

Функция разделяет кадр на два полукдра.

## Аргументы

pSrcImg	указатель на исходный кадр.
nSrcWidth	ширина исходного кадра в пикселях.
nSrcHeight	высота исходного кадра в пикселях.
pDstEven	указатель на буфер четного полукадра
pDstOdd	указатель на буфер нечетного полукадра

## Возвращает

void

## 7.56 КИХ-фильтрация

Двумерная КИХ фильтрация

Классы

- class `CIMG_FIR`< `nmbits_in`, `nmbits_out` >

### 7.56.1 Подробное описание

Двумерная КИХ фильтрация

## 7.57 Floodfill

Исполняет разделение бинарной картинки на односвязные области. Пример вызова: `no=VL_↵ FloodFill32b(pSrcImage, Tetr,Image, pTmpBuff, nSrcWidth, nSrcHeight);`.

### Функции

- `int IMG_FloodFill (unsigned int *pSrcImage, SegmentInfo *pSegmentInfo, unsigned int *p↵ SegmentImage, int nWidth, int nHeight, unsigned int *pTmpBuff)`
- `int FloodFill8 (void *src, void *dst, int nWidth, int nHeight, spot\_struct *spot, int lenSpot, unsigned *pixels, int mSpot, int dtFull, int dtSpot, int lDiag, int lDropSpot, ds\_struct *drop↵ Spot, int nPxlMin, int nPxlMax, int dXYmin, int dXYmax)`

Функция `FloodFill8` выполняет поиск пятен (сегментов, односвязных областей) во входной 8-битной матрице (изображения в градациях серого от 0 до 255), и строит такие же пятна в выходной матрице, заполняя их одним и тем же значением (цветом, соответствующим номеру пятна).

### 7.57.1 Подробное описание

Исполняет разделение бинарной картинки на односвязные области. Пример вызова: `no=VL_↵ FloodFill32b(pSrcImage, Tetr,Image, pTmpBuff, nSrcWidth, nSrcHeight);`.

#### Аргументы

<code>pSrcImage</code>	Входное изображение
<code>pSegmentInfo</code>	массив структур, где содержатся минимальные и максимальные координаты прямоугольника, описывающего сегментированную область.
<code>nWidth</code>	ширина изображения
<code>nHeight</code>	высота изображения
<code>pTmpBuff</code>	Временный массив. Его размер должен быть $2*nWidth*nHeight$

#### Возвращаемые значения

<code>pSegmentImage</code>	Результирующее "изображение", где все точки одного сегмента имеют одинаковое значение (1,...)
----------------------------	---

#### Возвращает

Число сегментов на изображении

#### Заметки

Массив `pSegmentImage` должен быть обнулен. Функция изменяет массив `pSrcImage`.

### 7.57.2 Функции



## 7.57.2.1 FloodFill8()

```
int FloodFill8 (
    void * src,
    void * dst,
    int nWidth,
    int nHeight,
    spot_struct * spot,
    int lenSpot,
    unsigned * pixels,
    int mSpot,
    int dtFull,
    int dtSpot,
    int lDiag,
    int lDropSpot,
    ds_struct * dropSpot,
    int nPxlMin,
    int nPxlMax,
    int dXYmin,
    int dXYmax )
```

Функция FloodFill8 выполняет поиск пятен (сегментов, односвязных областей) во входной 8-битной матрице (изображения в градациях серого от 0 до 255), и строит такие же пятна в выходной матрице, заполняя их одним и тем же значением (цветом, соответствующим номеру пятна).

## Аргументы

src	входная 8-битная матрица размером nHeight x nWidth. Внимание. Входная матрица src модифицируется: <ul style="list-style-type: none"> <li>• в целях оптимизации программы обнуляются верхняя и нижняя строки, и крайний левый и крайний правый столбцы (границы матрицы),</li> <li>• в процессе обработки найденные пятна обнуляются.</li> </ul>
dst	результатирующая 8-битная матрица того же размера nHeight x nWidth. Внимание. Перед обращением к функции выходная матрица dst должна быть обнулена.
nWidth	ширина входной (и выходной) матрицы в 8-битных элементах. Ограничения. Ширина матрицы должна быть кратна 32-битным словам: nWidth - положительное, (nWidth & 3) = 0.
nHeight	высота входной (и выходной) матрицы. Ограничения. Высота матрицы не должна превышать 1080 строк: nHeight - положительное, (nHeight <= 1080). Это ограничение можно обойти задав в начале файла FloodFill8.asm вместо 1080 нужное значение константы: const NHEIGHT = 1080;
spot	массив, содержащий обобщенную информацию о найденных пятнах – минимально 6 параметров формата int для каждого пятна: Замечание1. Чтобы гарантированно избежать переполнения массива spot в общем случае (когда характер матрицы, количество, размер пятен неизвестны, и не задан параметр mSpot>0), размер массива spot в 32-битных словах должен быть: ( nSpotMax + 1 + [(nSpotMax-1)/255] ) * lenSpot, nSpotMax = [(nHeight-2)*(nWidth-2)/2 + 0.5], при lDiag=0, = [(nHeight-2)/2+0.5] * [(nWidth-2)/2+0.5], при lDiag=1, квадратные скобки [...] обозначают целую часть числа, напомним: 2 строки и 2 столбца входной матрицы обнулены. Замечание2. Если задан параметр mSpot>0, то массив spot достаточно определить для mSpot пятен (mSpot задается с учетом служебного и фиктивных пятен).

## Аргументы

lenSpot	размер пятна в 32-битных словах, позволяет задать размер пятна более 6 слов, резервируя место для дополнительных параметров пятна довычисляемых другими функциями (например, угол наклона пятна к горизонтальной оси... ). Замечание. Если параметр lenSpot задан некорректно lenSpot<6, то ему присваивается минимальное возможное значение lenSpot=6.
pixels	массив координат пикселей, принадлежащих найденным пятнам, каждый элемент массива состоит из одного 32 битного слова, включающего в себя координаты пикселя в матрице src (или dst): unsigned int ij; биты [31..16] - i - номер строки, биты [15..0] - j - номер столбца. (Отсюда ограничение: nHeight, nWidth не могут быть больше 2**16-1 = 65535). Замечание. В общем случае (когда заранее неизвестно о количестве и размерах пятен в матрице), чтобы гарантированно избежать переполнения массива pixels, его размер должен быть (nHeight-2)*(nWidth-2) 32-битных слов (что почти в 4 раза больше входной 8-битной матрицы, напомним: 2 строки и 2 столбца входной матрицы обнулены).
mSpot	число пятен, ограничивающее поиск. Если задано mSpot>0, то после обработки очередного пятна проверяется: достигло ли количество найденных пятен nSpot заданного mSpot (mSpot<=nSpot)? Если «Да», то происходит (принудительный) выход из программы с записью в spot[0].Ymin=2. Если задано mSpot=0, то проверок на количество найденных пятен не производится. Замечание1. Если задать mSpot<=255, то пятна будут однозначно соответствовать своим номерам, то есть не будет пятен с повторяющимися номерами. Замечание2. mSpot следует задавать с учетом служебного пятна (с номером 0) и фиктивных пятен (с номерами 256,512,768...), то есть, если поиск надо ограничить нахождением первых mSpotReal>0 пятен, то mSpot = 1 + mSpotReal + [(mSpotReal-1)/255] квадратные скобки [...] обозначают целую часть числа
dtFull	время, выделенное на работу программы (в тактах процессора). Если задано dtFull>0, то после обработки очередного пятна (и в конце каждой строки) проверяется: осталось ли время на обработку следующего пятна dtRest<dtSpot? (dtRest=dtFull-(T-T0), T – текущее время, T0 – начало работы программы). Если «Нет», то происходит выход из программы с записью в spot[0].Ymin=1. Если задано dtFull=0, то проверок на достаточность времени на обработку следующего пятна не производится. Замечание1. Задание dtFull>0 маленьким может значительно сократить время работы программы, однако, не гарантирует корректного завершения программы (например, когда обрабатываемое пятно занимает всю оставшуюся часть матрицы). Замечание2. При выборе значения dtFull следует учитывать время обработки типового пятна dtSpot.
dtSpot	время обработки типового пятна (в тактах процессора). Используется совместно с параметром dtFull при оценке достаточного времени на обработку следующего пятна. Ограничение. Если задано dtFull>0, то dtSpot должно быть меньше него: dtSpot - неотрицательное, (dtFull>0)&&(dtSpot<dtFull).

## Аргументы

lDiag	<p>параметр, определяющий связность пятна по диагоналям:</p> <ul style="list-style-type: none"> <li>• lDiag=0 - элементы являются соседними, если они граничат или по вертикали, или по горизонтали (по ходам ладьи, но не по ходам слона),</li> <li>• lDiag=1 - элементы являются соседними, если они граничат или по вертикали, или по горизонтали, или по диагоналям (по ходам ферзя).</li> </ul> <p>Замечание. Как показывает практика, результат обработки видеоизображений существенно не зависит от параметра lDiag, но при lDiag=1 значительно увеличивает время работы функции. Если нет особой необходимости, рекомендуется задавать lDiag=0.</p>
lDropSpot	<p>параметр, задающий необходимость отбраковки найденных пятен по условиям НЕвхождения числа пикселей или размеров пятна в заданные диапазоны (см. ниже параметры nPxLmin, nPxLmax, dXYmin, dXYmax), статистика по отбракованным пятнам ведется в массиве dropSpot.</p> <p>lDropSpot=0 – найденные пятна НЕ отбраковываются, то есть все найденные пятна учитываются в массивах spot, pixels и отмечаются в выходной матрице dst, lDropSpot=1 – пятна с числом пикселей или размеров пятна не попадающими в заданные диапазоны отбраковываются, то есть не учитываются в массивах spot, pixels и не отмечаются в выходной матрице dst, статистика по отбракованным пятнам ведется в массиве dropSpot.</p> <p>Замечание. Задание параметра lDropSpot=1 приводит к увеличению работы функции и требует выделения памяти для массива dropSpot. Если нет особой необходимости, задавайте lDropSpot=0.</p>
dropSpot	<p>– массив, в который опционально собирается статистика по отбракованным пятнам: на каждый из 4 признаков отбраковки: nPxLMin, nPxLMax, dXYmin, dXYmax, - отведено по 3 параметра формата int: dropSpot[4*3] +0 +1 +2</p> <ul style="list-style-type: none"> <li>• 0 – dropSpot(nPxLMin): [ nnSpot, nnPxL, dttSpot]</li> <li>• 3 - dropSpot(nPxLMax): [ nnSpot, nnPxL, dttSpot]</li> <li>• 6 - dropSpot(dXYmin) : [ nnSpot, nnPxL, dttSpot]</li> <li>• 9 - dropSpot(dXYmax) : [ nnSpot, nnPxL, dttSpot]</li> </ul> <p>Перед обращением к функции массив dropSpot (размером 12 int) должен быть обнулен.</p> <p>При ldropSpot=0 статистика не ведется и массив dropSpot не требуется, то есть вместо него можно задать фиктивный параметр, например, 0.</p>
nPxLMin	<p>минимальное число пикселей в пятне, при lDropSpot=1: пятно отбраковывается, если число его пикселей меньше nPxLMin, при lDropSpot=0 проверок не производится, параметр можно задать 0.</p>
nPxLMax	<p>максимальное число пикселей в пятне,</p> <ul style="list-style-type: none"> <li>• при lDropSpot=1: пятно отбраковывается, если число его пикселей больше nPxLMax,</li> <li>• при lDropSpot=0 проверок не производится, параметр можно задать 0.</li> </ul>

## Аргументы

dXYmin	<p>минимальная протяженность пятна по X или по Y в пикселах,</p> <ul style="list-style-type: none"> <li>при IDropSpot=1: пятно отбраковывается, если его протяженность меньше dXYMax,</li> <li>при IDropSpot=0 проверок не производится, но параметр используется для определения шага поиска по строкам входной матрицы src (и начальной строки поиска): <math>dI = \max(1, dXYmin)</math></li> </ul> <p>Замечание. Задание dXYmin&gt;1 приводит к просмотру входной матрицы src через каждые dXYmin строк, что может значительно сократить время работы функции, так как пятна меньшего размера, попавшие между строк поиска "обходятся" (однако, не гарантирует этого, например, когда пятно занимает всю матрицу). Поэтому, независимо от параметра IDropSpot надо задавать dXYmin, по возможности, большим.</p>
dXYmax	<p>максимальная протяженность пятна по X или по Y в пикселах,</p> <ul style="list-style-type: none"> <li>при IDropSpot=1: пятно отбраковывается, если его протяженность больше dXYMax,</li> <li>при IDropSpot=0 проверок не производится, параметр можно задать 0.</li> </ul> <p>Ограничения: nPxMin, nPxMax, dXYmin, dXYmax - неотрицательные, (nPxMin &lt;= nPxMax), (dXYmin &lt;= dXYmax).</p>

## Возвращает

Функция возвращает целое число:

- либо положительное целое, тогда это - nSpot - число найденных пятен,
- либо отрицательное целое, тогда это - nError - код ошибки.

Число найденных пятен nSpot включает в себя также служебное и фиктивные пятна (с номерами 0,256,512,768...). Таким образом, число реальных пятен равно  $nSpotReal = nSpot - 1 - [nSpot/256]$

Если не найдено ни одного реального пятна, то nSpot=1 – одно служебное пятно.

Код ошибки возвращается, если в начале работы функции при проверке входных данных обнаружится недопустимая комбинация параметров: тогда происходит (принудительный) выход из программы с кодом ошибки:

```

dI = max(1,dXYmin);
if ( (nHeight<=0) || (nHeight < dI +2) || (1080<nHeight) ) return -1;
if ( (nWidth <=0) || (nWidth < dI +2) || ((nWidth & 3)!=0)) return -2;
if ( (dtSpot <0) || ((dtFull != 0) && (dtFull < dtSpot)) ) return -3;
if ( (nPxMin <0) || (nPxMax < nPxMin) ) return -4;
if ( (dXYmin <0) || (dXYmax < dXYmin) ) return -5;
if ( mSpot <0) return -6;

```

При нормальном выходе из функции (без ошибок) в параметре Ymin служебного пятна сообщается дополнительный признак завершения программы nExitCode:

- spot[0].Ymin=0 – вся матрица src просмотрена, найдены все пятна, естественный выход из программы.
- spot[0].Ymin=1 – принудительный выход из программы по исчерпанию заданного времени работы программы dtFull>0.
- spot[0].Ymin=2 – принудительный выход из программы по нахождению заданного количества пятен mSpot>0.

## Заметки

## Предельные случаи

Следует выделить предельные случаи:

- одно сплошное пятно на всю матрицу - максимальное время работы функции и максимальный размер массива `pixels` (но минимальный размер массива `spot`),
- «шахматная доска» - однопиксельные пятна, расположенные по диагоналям - максимальный размер массива `spot`.

## Алгоритм

В процессе поиска координаты всех пикселей каждого пятна сохраняются в массиве `pixels`:

- пиксели каждого пятна занимают непрерывную область в массиве `pixels`,
- параметр `spot[i].noPx1` *i*-го пятна в массиве `spot` указывает на начальный пиксел (*i*+1)-го пятна. В начале работы программы устанавливается указатель на начальный пиксел следующего пятна `noPx1 = 0`.

1. Цикл поиск пятна (осуществляется построчным попиксельным проходом матрицы `src`), если очередной пиксел нулевой, то переход к следующему пикселу иначе (встретился ненулевой пиксел)

2. начало обработки (очередного) пятна:

- засечь время начала обработки пятна `tSpot0=clock()`;
- устанавливаются указатели начала и конца очереди пикселей пятна в массиве `pixels`: `noPx10 = noPx11 = noPx1`.
- пиксел в матрице `src` обнуляется,
- координаты пиксела заносятся в массив `pixels`, указатель `noPx11` увеличивается на 1

3. while-цикл обработки пятна:

- выбирается пиксел `noPx10` из массива `pixels`, указатель `noPx10` увеличивается на 1
  - просматриваются все его соседние элементы (с учетом параметра `lDiag`) координаты ненулевых пикселей записываются в массив `pixels`
  - пиксели в матрице `src` обнуляются
  - если указатели `noPx10`, `noPx11` не равны (не все пиксели пятна и их соседи просмотрены), то переход на начало while-цикла
- // пятно найдено! `noPx10=noPx11` - указывают на начальный пиксел  
//следующего пятна в массиве `pixels`.

Вычисление минимального прямоугольника (`Xmin`, `Ymin`, `Xmax`, `Ymax`), объемлющего пятно

Если задан параметр `lDropSpot=1`, то проверка условий отбраковки пятна по признакам `nPx1Min`, `nPx1Max`, `dXYmin`, `dXYmax` если пятно отбраковано, то

формирование записи массива `dropSpot` по соответствующему признаку, переход на поиск следующего пятна (`noPx1` не изменяется).

Меняется указатель `noPx1=noPx11`.

Формирование параметров данного пятна в массиве `spot[i] =`

(`Xmin`, `Ymin`, `Xmax`, `Ymax`, `noPx1`, 0, 0, 0)

Если задан параметр вычисления угла наклона пятна (`lAngleSpot=1`), то

вычисление `codeAngle`, `lBadSpot` и запись в `spot[i] =`

(., ., ., ., ., `codeAngle`, `lBadSpot`, .)

Запись `dtSpot=clock()-tSpot0` в `spot[i] =` (., ., ., ., ., ., ., `dtSpot`).

Выбор следующего пиксела, переход на начало цикла поиска очередного пятна (1).

4. Конец программы.

## 7.58 IMG\_Convert

Преобразование типов для элементов изображения.

Функция конвертирует элементы RGB между 8-м разрядным представлением и 10-ти разрядным.

Функции

- void IMG\_Convert (RGB32\_nm8u \*pSrcImg, RGB32\_nm10u \*pDstImg, int nSize)
- void IMG\_Convert (RGB32\_nm10u \*pSrcImg, RGB32\_nm8u \*pDstImg, int nSize)

### 7.58.1 Подробное описание

Преобразование типов для элементов изображения.

Функция конвертирует элементы RGB между 8-м разрядным представлением и 10-ти разрядным.

Аргументы

pSrcImg	Указатель на первый элемент исходного изображения.
pDstImg	Указатель на первый элемент результирующего изображения.
nSize	Количество элементов в изображении.

Возвращает

void

Restrictions:

- Указатели на изображения должны быть выровнены в памяти на границу 64-х разрядного слова;

## 7.59 IMG\_RGB32ToGray

Преобразование пикселей из RGB в яркость

### Функции

- void IMG\_RGB32ToGray (RGB32\_nm8u \*pRGB, nm32u \*pDstGray, int nSize)
- void IMG\_RGB32ToGray (RGB32\_nm8u \*pRGB, nm32s \*pDstGray, int nSize)
- void IMG\_RGB32ToGray (RGB32\_nm8u \*pRGB, nm8s \*pDstGray, int nSize, void \*pTmpBuf)
- void IMG\_RGB32ToGray (RGB32\_nm10s \*pRGB, nm32s \*pDstGray, int nSize)
- void IMG\_RGB32ToGray (RGB32\_nm10s \*pRGB, nm32u \*pDstGray, int nSize)
- void IMG\_RGB32ToGray (RGB32\_nm10s \*pRGB, nm8s \*pDstGray, int nSize, void \*pTmpBuf)

### 7.59.1 Подробное описание

Преобразование пикселей из RGB в яркость

#### Аргументы

pRGB	Вход, по 4 байта на пиксель. Порядок байтов B, G, R, 0. Результат в виде 32 битных целых чисел, в которых полезные данные занимают младшие 24 бита. Для получения восьмибитовых пикселей необходимо вырезать биты 16..23, например, с помощью nmppsClipRShiftConvert_Add_32s(nm32s* pSrcVec, int nClipFactor,int nShift, nm64u* nAddValue,nm8s* pDstVec, int nSize); с параметрами nClipFactor=24, nShift=16. Количество пикселей на входе и выходе. nSize=[64,128,192,...] Временный массив размером nm32s[nSize] .
------	--

## 7.60 Переупорядочивание изображений

### Группы

- [Блочное переупорядочивание](#)

Блочное перупорядочивание изображений.

#### 7.60.1 Подробное описание



## 7.61 Блочное переупорядочивание

Блочное переупорядочивание изображений.

Группы

- [IMG\\_SplitIntoBlocks](#)  
Преобразует изображение в последовательность квадратных блоков.
- [IMG\\_MergeFromBlocks](#)  
Объединяет последовательность квадратных блоков в изображение.

### 7.61.1 Подробное описание

Блочное переупорядочивание изображений.

## 7.62 IMG\_SplitIntoBlocks

Преобразует изображение в последовательность квадратных блоков.

Функции

- void IMG\_SplitIntoBlocks8x8 (nm8s \*pSrcImg, nm8s \*pDstBlockSeq, int nWidth, int nHeight)

### 7.62.1 Подробное описание

Преобразует изображение в последовательность квадратных блоков.

Аргументы

pSrcImg	Исходное изображение Выходная последовательность блоков Ширина исходного изображения в пикселах. nWidth =[8,16,24...] Высота исходного изображения в пикселах. nHeight =[8,16,24...] Исходное изображение имеет вид
---------	---

```
[A00|A01|A02|A03|A04|A05|A06|A07|B00|B01|..|B07|C00|....H07]
[A10|A11|A12|A13|A14|A15|A16|A17|B10|B11|..|B17|C10|....H17]
[A20|A21|A22|A23|A24|A25|A26|A27|B20|B21|..|B27|C20|....H27]
[A30|A31|A32|A33|A34|A35|A36|A37|B30|B31|..|B37|C30|....H37]
.....
[A70|A71|A72|A73|A74|A75|A76|A77|B70|B71|..|B77|C70|....H77]
.....
[I00|I01|I02|I03|I04|I05|I06|I07|J00|J01|.....]
[I10|I11|I12|I13|I14|I15|I16|I17|J00|J09|.....]
.....Z77]
```

Выходная последовательность блоков имеет вид

```
[A00|A01|A02|...|A06|A07|A20|A21|...|A77|B00|B01|B02|...|B07|B10|...|H77|
[I00|I01|I02|...|I06|I07|I10|I11|...|I77|J00|J01|J02|.....|Z77|
```

## 7.63 IMG\_MergeFromBlocks

Объединяет последовательность квадратных блоков в изображение.

### Функции

- void IMG\_MergeFromBlocks8x8 (nm8s \*pSrcBlockSeq, nm8s \*pDstImg, int nWidth, int nHeight)

#### 7.63.1 Подробное описание

Объединяет последовательность квадратных блоков в изображение.

#### Аргументы

pSrcBlockSeq	Входная последовательность блоков . Результирующее изображение Ширина исходного изображения в пикселах. nWidth=[8,16,24...] Высота исходного изображения в пикселах. nHeight=[8,16,24..] Входная последовательность блоков имеет вид
--------------	--

```
[A00|A01|A02|...|A06|A07|A20|A21|...|A77|B00|B01|B02|...|B07|B10|...|H77|
[I00|I01|I02|...|I06|I07|I10|I11|...|I77|J00|J01|J02|.....|Z77]
```

Результирующее изображение имеет вид

```
[A00|A01|A02|A03|A04|A05|A06|A07|B00|B01|..|B07|C00|...|H07|
[A10|A11|A12|A13|A14|A15|A16|A17|B10|B11|..|B17|C10|...|H17|
[A20|A21|A22|A23|A24|A25|A26|A27|B20|B21|..|B27|C20|...|H27|
[A30|A31|A32|A33|A34|A35|A36|A37|B30|B31|..|B37|C30|...|H37|
.....
[A70|A71|A72|A73|A74|A75|A76|A77|B70|B71|..|B77|C70|...|H77|
.....
[I00|I01|I02|I03|I04|I05|I06|I07|J00|J01|.....]
[I10|I11|I12|I13|I14|I15|I16|I17|J00|I09|.....
.....Z77]
```

## 7.64 IMG\_Free

Освобождение памяти для изображений.

Функции

- void IMG\_Free (void \*ptr)

### 7.64.1 Подробное описание

Освобождение памяти для изображений.

Заметки

Данная функция должна вызываться только для векторов, распределенных с помощью функций IMG\_Malloc.

## 7.65 IMG\_Release

Освобождение блоков памяти, выделенных функциями IMG\_Create\*\*\*.

Функции

- `__INLINE__ void IMG_ReleaseObject (nm64s *pKernel)`

### 7.65.1 Подробное описание

Освобождение блоков памяти, выделенных функциями IMG\_Create\*\*\*.

## 7.66 Арифметические действия

## 7.67 Масочная фильтрация

### Группы

- [КИХ-фильтрация](#)

Двумерная КИХ фильтрация

#### 7.67.1 Подробное описание

## 7.68 Инициализация и копирование

### Группы

- [IMG\\_Convert](#)

Преобразование типов для элементов изображения.

Функция конвертирует элементы RGB между 8-м разрядным представлением и 10-ти разрядным.

- [IMG\\_RGB32ToGray](#)

Преобразование пикселей из RGB в яркость

### 7.68.1 Подробное описание



## 7.69 Функции поддержки

### Группы

- [IMG\\_Free](#)  
Освобождение памяти для изображений.
- [IMG\\_Release](#)  
Освобождение блоков памяти, выделенных функциями IMG\_Create\*\*\*.

### 7.69.1 Подробное описание

## 7.70 Функции графического вывода текста

- `int IMG_Print8x15 (char *str, void *img, int imgWidth, int x, int y, int FGcolor, int BGcolor)`  
Prints text over 8-bit grayscale image buffer with 8x15 font size.
- `char * hex2ascii (int value, char *str)`  
Converts integer to 8-byte string in hexadecimal base.
- `char * hex2ascii (int value)`  
Converts integer to 8-byte string in hexadecimal base.

### 7.70.1 Подробное описание

### 7.70.2 Функции

#### 7.70.2.1 hex2ascii() [1/2]

```
char* hex2ascii (
    int value,
    char * str )
```

Converts integer to 8-byte string in hexadecimal base.

Аргументы

in	value	Value to be converted to a string
in	str	Array in memory where to store the resulting null-terminated string

Возвращает

A pointer to the resulting null-terminated string, same as parameter str.

Details

#### 7.70.2.2 hex2ascii() [2/2]

```
char* hex2ascii (
    int value )
```

Converts integer to 8-byte string in hexadecimal base.

Аргументы

in	value	Value to be converted to a string
----	-------	-----------------------------------

Возвращает

A pointer to the global resulting null-terminated string

Заметки

Function uses internal string buffer for result. Previous result of call this function will be overwritten.

### 7.70.2.3 IMG\_Print8x15()

```
int IMG_Print8x15 (  
    char * str,  
    void * img,  
    int imgWidth,  
    int x,  
    int y,  
    int FGcolor,  
    int BGcolor )
```

Prints text over 8-bit grayscale image buffer with 8x15 font size.

Аргументы

in	str	String to be printed
in	img	Pointer to 8-bit grayscale image , where text should be printed
in	imgWidth	Image width
in	x	Horizontal positon in image where text should be printed
in	y	Vertical positon in image where text should be printed
in	FGcolor	Foreground text color
in	BGcolor	Background text color

Возвращает

x-coordinate behind printed text

Заметки

Russian text is supported in win-1251 codepage

## 7.71 Инициализация и копирование

### Группы

- [nmprpmCoryua](#)

Копирование подматрицы с невыровненной по границам 64- разрядного слова позиции в выровненную.

- [MTR\\_Coryau](#)

Копирование подматрицы с выровненной по границе 64-х разрядных слов позиции в невыровненную позицию.

- [MTR\\_Coryu](#)

Функции копирования прямоугольных областей памяти между двумерными массивами.

### 7.71.1 Подробное описание

## 7.72 nmppmCopyua

Копирование подматрицы с невыровненной по границам 64- разрядного слова позиции в выровненную.

### Функции

- void nmppmCopyua\_8s (nm8s \*pSrcMtr, int nSrcStride, int nSrcOffset, nm8s \*pDstMtr, int nDstStride, int nHeight, int nWidth)
- void nmppmCopyua\_16s (nm16s \*pSrcMtr, int nSrcStride, int nSrcOffset, nm16s \*pDstMtr, int nDstStride, int nHeight, int nWidth)
- void nmppmCopyua\_32s (nm32s \*pSrcMtr, int nSrcStride, int nSrcOffset, nm32s \*pDstMtr, int nDstStride, int nHeight, int nWidth)
- void nmppmCopy\_32fc (double \*SrcMtr, int nSrcStride, double \*DstMtr, int nDstStride, int nHeight, int nWidth)

### 7.72.1 Подробное описание

Копирование подматрицы с невыровненной по границам 64- разрядного слова позиции в выровненную.

#### Аргументы

pSrcMtr	Исходная матрица.
nSrcStride	Ширина исходной матрицы в элементах.
nSrcOffset	Смещение в элементах от начала матрицы-источника.
nDstStride	Ширина результирующей матрицы в элементах.
nHeight	Число строк подматрицы.
nWidth	Число столбцов подматрицы.

#### Возвращаемые значения

pDstMtr	Результирующая матрица.
---------	-------------------------

#### Возвращает

void

#### Restrictions:

- Входная и выходная матрицы не должны перекрываться в памяти
- Число столбцов копируемой подматрицы nWidth должна быть кратно числу элементов в 64-х разрядном слове.

## 7.73 MTR\_Соруаu

Копирование подматрицы с выровненной по границе 64-х разрядных слов позиции в невыровненную позицию.

### Функции

- void nmppmCopyau\_8s (nm8s \*pSrcMtr, int nSrcStride, nm8s \*pDstMtr, int nDstStride, int nDstOffset, int nHeight, int nWidth)
- void nmppmCopyau\_16s (nm16s \*pSrcMtr, int nSrcStride, nm16s \*pDstMtr, int nDstStride, int nDstOffset, int nHeight, int nWidth)
- void nmppmCopyau\_32s (nm32s \*pSrcMtr, int nSrcStride, nm32s \*pDstMtr, int nDstStride, int nDstOffset, int nHeight, int nWidth)

### 7.73.1 Подробное описание

Копирование подматрицы с выровненной по границе 64-х разрядных слов позиции в невыровненную позицию.

#### Аргументы

pSrcMtr	Исходная матрица.
nSrcStride	Ширина исходной матрицы в элементах.

#### Возвращаемые значения

pDstMtr	Результирующая матрица.
---------	-------------------------

#### Аргументы

nDstStride	Ширина результирующей матрицы в элементах.
nDstOffset	Индекс столбца (в элементах) куда производится вставка.
nHeight	Число строк подматрицы.
nWidth	Число столбцов подматрицы.

#### Возвращает

void

#### Restrictions:

- Входная и выходная матрицы не должны перекрываться в памяти
- Число столбцов копируемой подматрицы nWidth должна быть кратной 64-битам.

## 7.74 MTR\_Copy

Функции копирования прямоугольных областей памяти между двумерными массивами.

### Функции

- void nmppmCopy\_1 (nm1 \*pSrcMtr, int nSrcStride, nm1 \*pDstMtr, int nDstStride, int nHeight, int nWidth)
- void nmppmCopy\_2s (nm2s \*pSrcMtr, int nSrcStride, nm2s \*pDstMtr, int nDstStride, int nHeight, int nWidth)
- void nmppmCopy\_4s (nm4s \*pSrcMtr, int nSrcStride, nm4s \*pDstMtr, int nDstStride, int nHeight, int nWidth)
- void nmppmCopy\_8s (nm8s \*pSrcMtr, int nSrcStride, nm8s \*pDstMtr, int nDstStride, int nHeight, int nWidth)
- void nmppmCopy\_16s (nm16s \*pSrcMtr, int nSrcStride, nm16s \*pDstMtr, int nDstStride, int nHeight, int nWidth)
- void nmppmCopy\_32s (nm32s \*pSrcMtr, int nSrcStride, nm32s \*pDstMtr, int nDstStride, int nHeight, int nWidth)
- void nmppmCopy\_64s (nm64s \*pSrcMtr, int nSrcStride, nm64s \*pDstMtr, int nDstStride, int nHeight, int nWidth)

### 7.74.1 Подробное описание

Функции копирования прямоугольных областей памяти между двумерными массивами.

#### Аргументы

pSrcMtr	Исходная матрица.
nSrcStride	Ширина исходной матрицы в элементах.
nDstStride	Ширина результирующей матрицы в элементах.
nHeight	Число строк подматрицы.
nWidth	Число столбцов подматрицы.

#### Возвращаемые значения

pDstMtr	Результирующая матрица.
---------	-------------------------

#### Возвращает

void

#### Restrictions:

Выходная матрица не может быть перезаписана (т.е. помещена непосредственно на место входной)

## 7.75 nmppmMul\_mm

Умножение матрицы на матрицу.

Функции

- void nmppmMul\_mm\_8s8s (nm8s \*pSrcMtr1, int nHeight1, int nWidth1, nm8s \*pSrcMtr2, nm8s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_8s16s (nm8s \*pSrcMtr1, int nHeight1, int nWidth1, nm16s \*pSrcMtr2, nm16s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_8s32s (nm8s \*pSrcMtr1, int nHeight1, int nWidth1, nm32s \*pSrcMtr2, nm32s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_8s64s (nm8s \*pSrcMtr1, int nHeight1, int nWidth1, nm64s \*pSrcMtr2, nm64s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_16s16s (nm16s \*pSrcMtr1, int nHeight1, int nWidth1, nm16s \*pSrcMtr2, nm16s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_16s32s (nm16s \*pSrcMtr1, int nHeight1, int nWidth1, nm32s \*pSrcMtr2, nm32s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_16s64s (nm16s \*pSrcMtr1, int nHeight1, int nWidth1, nm64s \*pSrcMtr2, nm64s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_32s32s (nm32s \*pSrcMtr1, int nHeight1, int nWidth1, nm32s \*pSrcMtr2, nm32s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_32s64s (nm32s \*pSrcMtr1, int nHeight1, int nWidth1, nm64s \*pSrcMtr2, nm64s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_colmajor\_8s8s (const nm8s \*pSrcMtr1, int nHeight1, int nWidth1, const nm8s \*pSrcMtr2, nm8s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_colmajor\_8s16s (const nm8s \*pSrcMtr1, int nHeight1, int nWidth1, const nm16s \*pSrcMtr2, nm16s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_colmajor\_8s32s (const nm8s \*pSrcMtr1, int nHeight1, int nWidth1, const nm32s \*pSrcMtr2, nm32s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_colmajor\_8s64s (const nm8s \*pSrcMtr1, int nHeight1, int nWidth1, const nm64s \*pSrcMtr2, nm64s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_colmajor\_16s16s (const nm16s \*pSrcMtr1, int nHeight1, int nWidth1, const nm16s \*pSrcMtr2, nm16s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_colmajor\_16s32s (const nm16s \*pSrcMtr1, int nHeight1, int nWidth1, const nm32s \*pSrcMtr2, nm32s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_colmajor\_16s64s (const nm16s \*pSrcMtr1, int nHeight1, int nWidth1, const nm64s \*pSrcMtr2, nm64s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_colmajor\_32s32s (const nm32s \*pSrcMtr1, int nHeight1, int nWidth1, const nm32s \*pSrcMtr2, nm32s \*pDstMtr, int nWidth2)
- void nmppmMul\_mm\_colmajor\_32s64s (const nm32s \*pSrcMtr1, int nHeight1, int nWidth1, const nm64s \*pSrcMtr2, nm64s \*pDstMtr, int nWidth2)

### 7.75.1 Подробное описание

Умножение матрицы на матрицу.

Аргументы

pSrcMtr1	Исходная матрица.
pSrcMtr2	Матрица-множитель.
nHeight1	Число строк исходной матрицы.
nWidth1	Число столбцов исходной матрицы.
nWidth2	Число столбцов матрицы множителя.



Возвращаемые значения

pDstMtr	Результирующая матрица.
---------	-------------------------

Возвращает

void

## 7.76 nmppmMul\_mv\_

Умножение матрицы на вектор.

### Функции

- void nmppmMul\_mv\_8s64s (nm8s \*pSrcMtr, nm64s \*pSrcVec, nm64s \*pDstVec, int nHeight, int nWidth)
- void nmppmMul\_mv\_16s64s (nm16s \*pSrcMtr, nm64s \*pSrcVec, nm64s \*pDstVec, int nHeight, int nWidth)
- void nmppmMul\_mv\_32s64s (nm32s \*pSrcMtr, nm64s \*pSrcVec, nm64s \*pDstVec, int nHeight, int nWidth)
- void nmppmMul\_mv\_8s16s\_8xH (v8nm8s \*pSrcMtr, v8nm16s \*pSrcVec, nm16s \*pDstVec, int nHeight)
- void nmppmMul\_mv\_16s16s\_8xH (v8nm16s \*pSrcMtr, v8nm16s \*pSrcVec, nm16s \*pDstVec, int nHeight)
- void nmppmMul\_mv\_colmajor\_8s64s (const nm8s \*pSrcMtr, const nm64s \*pSrcVec, nm64s \*pDstVec, int nHeight, int nWidth)
- void nmppmMul\_mv\_colmajor\_16s64s (const nm16s \*pSrcMtr, const nm64s \*pSrcVec, nm64s \*pDstVec, int nHeight, int nWidth)
- void nmppmMul\_mv\_colmajor\_32s64s (const nm32s \*pSrcMtr, const nm64s \*pSrcVec, nm64s \*pDstVec, int nHeight, int nWidth)

### 7.76.1 Подробное описание

Умножение матрицы на вектор.

#### Аргументы

pSrcMtr	Исходная матрица.
pSrcVec	Вектор-множитель.
pSrcVec8	Вектор-множитель размерности 8.
nHeight	Число строк исходной матрицы.
nWidth	Число столбцов исходной матрицы.

#### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

#### Возвращает

void

## 7.77 nmppmMul\_mv\_\_AddC

Умножение матрицы на вектор с добавлением константы.

Функции

- void nmppmMul\_mv\_\_AddC (v2nm32s \*pSrcMtr, v2nm32s \*pnSrcVec, int nAddVal, nm32s \*pDstVec, int nHeight)

### 7.77.1 Подробное описание

Умножение матрицы на вектор с добавлением константы.

Аргументы

pSrcMtr	Исходная матрица.
pSrcVec	Вектор-множитель.
pSrcVec	Вектор-множитель размерности 2.
nAddVal	Константа.
nHeight	Число строк исходной матрицы. nHeight = [0, 2, 4, ...].
nWidth	Число столбцов исходной матрицы.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.78 MTR\_ProdUnitV

Умножение матрицы на единичный вектор.

Функции

- void MTR\_ProdUnitV\_16s\_4xH (v4nm16s \*pSrcMtr, nm16s \*pDstVec, int nHeight)
- void MTR\_ProdUnitV\_16s\_16xH (v16nm8s \*pSrcMtr, nm16s \*pDstVec, int nHeight)

### 7.78.1 Подробное описание

Умножение матрицы на единичный вектор.

$$pDstVec(i) = \sum_{j=0}^{w-1} pSrcMtr(i, j)$$

Данная функция эквивалентна суммированию столбцов матрицы. Ширины матрицы, для которых имеется реализация данной функции указываются в ее названии.

Аргументы

pSrcMtr	Матрица.
nHeight	Число строк матрицы. nHeight=[128,256,...]

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.79 MTR\_Malloc

Распределение памяти для матриц библиотеки.

Начало и конец распределяемой памяти выравнивается на начало 64-х разрядного слова.

Распределение памяти для матриц библиотеки.

Начало и конец распределяемой памяти выравнивается на начало 64-х разрядного слова.

Аргументы

nHeight	Число строк в матрице.
nWidth	Число столбцов в матрице.
hint	Номер банка памяти. Может принимать значения MEM_LOCAL, MEM_GLOBAL.

Заметки

Память, распределенная с помощью функций MTR\_Malloc должна освобождаться с помощью функции MTR\_Free.

## 7.80 MTR\_Free

Освобождение памяти для матриц.

Функции

- `__INLINE__ void MTR_Free (void *ptr)`

### 7.80.1 Подробное описание

Освобождение памяти для матриц.

Заметки

Данная функция должна вызываться только для векторов, распределенных с помощью функций MTR\_Malloc.

## 7.81 MTR\_Addr

Возвращает адрес ячейки памяти, содержащей указанный элемент.

Реализация для процессора NeuroMatrix возвращает адрес, выровненный в памяти на 32 бита.

### Функции

- `__INLINE__ nm1 * MTR_Addr_1 (nm1 *pMTR, int nWidth, int nY, int nX)`
- `__INLINE__ nm2s * MTR_Addr_2s (nm2s *pMTR, int nWidth, int nY, int nX)`
- `__INLINE__ nm4s * MTR_Addr_4s (nm4s *pMTR, int nWidth, int nY, int nX)`
- `__INLINE__ nm8s * MTR_Addr_8s (nm8s *pMTR, int nWidth, int nY, int nX)`
- `__INLINE__ nm16s * MTR_Addr_16s (nm16s *pMTR, int nWidth, int nY, int nX)`
- `__INLINE__ nm32s * MTR_Addr_32s (nm32s *pMTR, int nWidth, int nY, int nX)`
- `__INLINE__ nm64s * MTR_Addr_64s (nm64s *pMTR, int nWidth, int nY, int nX)`
- `__INLINE__ nm2u * MTR_Addr_2u (nm2u *pMTR, int nWidth, int nY, int nX)`
- `__INLINE__ nm4u * MTR_Addr_4u (nm4u *pMTR, int nWidth, int nY, int nX)`
- `__INLINE__ nm8u * MTR_Addr_8u (nm8u *pMTR, int nWidth, int nY, int nX)`
- `__INLINE__ nm16u * MTR_Addr_16u (nm16u *pMTR, int nWidth, int nY, int nX)`
- `__INLINE__ nm32u * MTR_Addr_32u (nm32u *pMTR, int nWidth, int nY, int nX)`
- `__INLINE__ nm64u * MTR_Addr_64u (nm64u *pMTR, int nWidth, int nY, int nX)`

### 7.81.1 Подробное описание

Возвращает адрес ячейки памяти, содержащей указанный элемент.

Реализация для процессора NeuroMatrix возвращает адрес, выровненный в памяти на 32 бита.

#### Аргументы

pMtr	Входная матрица.
nWidth	Ширина таблицы в элементах.
nY	Номер строки.
nX	Номер колонки.

Возвращает

Адрес ячейки памяти.

#### Заметки

Для ускорения работы на PC возможно использование макроса ADDR(ptr, index), который раскрывается на PC как (ptr+index), а на NM как вызов функции MTR\_Addr.

## 7.82 MTR\_SetVal

Записывают число в элемент матрицы.

### Функции

- `__INLINE__ void MTR_SetVal_1 (nm1 *pMtr, int nWidth, int nY, int nX, int1b nVal)`
- `__INLINE__ void MTR_SetVal_2s (nm2s *pMtr, int nWidth, int nY, int nX, int2b nVal)`
- `__INLINE__ void MTR_SetVal_4s (nm4s *pMtr, int nWidth, int nY, int nX, int4b nVal)`
- `__INLINE__ void MTR_SetVal_8s (nm8s *pMtr, int nWidth, int nY, int nX, int8b nVal)`
- `__INLINE__ void MTR_SetVal_16s (nm16s *pMtr, int nWidth, int nY, int nX, int16b nVal)`
- `__INLINE__ void MTR_SetVal_32s (nm32s *pMtr, int nWidth, int nY, int nX, int32b nVal)`
- `__INLINE__ void MTR_SetVal_64s (nm64s *pMtr, int nWidth, int nY, int nX, int64b nVal)`
- `__INLINE__ void MTR_SetVal_2u (nm2u *pMtr, int nWidth, int nY, int nX, uint2b nVal)`
- `__INLINE__ void MTR_SetVal_4u (nm4u *pMtr, int nWidth, int nY, int nX, uint4b nVal)`
- `__INLINE__ void MTR_SetVal_8u (nm8u *pMtr, int nWidth, int nY, int nX, uint8b nVal)`
- `__INLINE__ void MTR_SetVal_16u (nm16u *pMtr, int nWidth, int nY, int nX, uint16b nVal)`
- `__INLINE__ void MTR_SetVal_32u (nm32u *pMtr, int nWidth, int nY, int nX, uint32b nVal)`
- `__INLINE__ void MTR_SetVal_64u (nm64u *pMtr, int nWidth, int nY, int nX, uint64b nVal)`

### 7.82.1 Подробное описание

Записывают число в элемент матрицы.

$$pMtr[nY][nX] = nVal$$

### Аргументы

pMtr	Матрица.
nWidth	Ширина матрицы в элементах
nY	Номер строки
nX	Номер столбца
nVal	Значение элемента

### Возвращает

void



## 7.83 MTR\_GetVal

Считывает значение элемента матрицы.

### Функции

- `__INLINE__ void MTR_GetVal_1 (nm1 *pMtr, int nWidth, int nY, int nX, int1b *nVal)`
- `__INLINE__ void MTR_GetVal_2s (nm2s *pMtr, int nWidth, int nY, int nX, int2b *nVal)`
- `__INLINE__ void MTR_GetVal_4s (nm4s *pMtr, int nWidth, int nY, int nX, int4b *nVal)`
- `__INLINE__ void MTR_GetVal_8s (nm8s *pMtr, int nWidth, int nY, int nX, int8b *nVal)`
- `__INLINE__ void MTR_GetVal_16s (nm16s *pMtr, int nWidth, int nY, int nX, int16b *nVal)`
- `__INLINE__ void MTR_GetVal_32s (nm32s *pMtr, int nWidth, int nY, int nX, int32b *nVal)`
- `__INLINE__ void MTR_GetVal_64s (nm64s *pMtr, int nWidth, int nY, int nX, int64b *nVal)`
- `__INLINE__ void MTR_GetVal_2u (nm2u *pMtr, int nWidth, int nY, int nX, uint2b *nVal)`
- `__INLINE__ void MTR_GetVal_4u (nm4u *pMtr, int nWidth, int nY, int nX, uint4b *nVal)`
- `__INLINE__ void MTR_GetVal_8u (nm8u *pMtr, int nWidth, int nY, int nX, uint8b *nVal)`
- `__INLINE__ void MTR_GetVal_16u (nm16u *pMtr, int nWidth, int nY, int nX, uint16b *nVal)`
- `__INLINE__ void MTR_GetVal_32u (nm32u *pMtr, int nWidth, int nY, int nX, uint32b *nVal)`
- `__INLINE__ void MTR_GetVal_64u (nm64u *pMtr, int nWidth, int nY, int nX, uint64b *nVal)`

### 7.83.1 Подробное описание

Считывает значение элемента матрицы.

$$nVal = pMtr[nY][nX]$$

### Аргументы

pMtr	Матрица.
nWidth	Ширина матрицы в элементах
nY	Номер строки
nX	Номер столбца

### Возвращаемые значения

nVal	Значение элемента
------	-------------------

### Возвращает

void

## 7.84 Функции поддержки

### Группы

- [MTR\\_Malloc](#)

Распределение памяти для матриц библиотеки.

Начало и конец распределяемой памяти выравнивается на начало 64-х разрядного слова.

- [MTR\\_Free](#)

Освобождение памяти для матриц.

- [MTR\\_Addr](#)

Возвращает адрес ячейки памяти, содержащей указанный элемент.

Реализация для процессора NeuroMatrix возвращает адрес, выровненный в памяти на 32 бита.

- [MTR\\_SetVal](#)

Записывает число в элемент матрицы.

- [MTR\\_GetVal](#)

Считывает значение элемента матрицы.

### 7.84.1 Подробное описание

## 7.85 Векторно-матричные операции

### Группы

- [nmppmMul\\_mm](#)  
Умножение матрицы на матрицу.
- [nmppmMul\\_mv\\_](#)  
Умножение матрицы на вектор.
- [nmppmMul\\_mv\\_\\_AddC](#)  
Умножение матрицы на вектор с добавлением константы.
- [MTR\\_ProdUnitV](#)  
Умножение матрицы на единичный вектор.

#### 7.85.1 Подробное описание

## 7.86 FFT-256

### Функции

- void [FFT\\_Fwd256Set6bit](#) ()  
Устанавливает 6-битную точность вычислений
- void [FFT\\_Fwd256Set7bit](#) ()  
Устанавливает 7-битную точность вычислений
- void [FFT\\_Fwd256](#) (nm32sc \*GSrcBuffer, nm32sc \*LDstBuffer, void \*LBuffer, void \*GBuffer, int ShiftR=-1)  
Прямое быстрое преобразование Фурье-256.

### 7.86.1 Подробное описание

### 7.86.2 Функции

#### 7.86.2.1 FFT\_Fwd256()

```
void FFT_Fwd256 (
    nm32sc * GSrcBuffer,
    nm32sc * LDstBuffer,
    void * LBuffer,
    void * GBuffer,
    int ShiftR = -1 )
```

Прямое быстрое преобразование Фурье-256.

Функция выполняет дискретное комплексное 256-точечное преобразование Фурье на базе алгоритма БПФ по основанию 16-16

#### Аргументы

in	GSrcBuffer	Входной массив размером 256 64-р. слов
out	LDstBuffer	Результирующий массив размером 256*3 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 256*3 64-р. слов
in	GBuffer	Временный массив на глобальной шине (Global Bus) размером 256*2 64-р. слов
in	ShiftR	Коэффициент нормализации, выполняет арифметический сдвиг результирующего массива на ShiftR бит вправо для получения нормализованного массива LDstBuffer. При передаче значения по умолчанию (-1) ShiftR автоматически принимается равным 14 если ранее установлена точность 7-бит функцией <a href="#">FFT_Fwd256Set7bit</a> () и 12 - если ранее установлена точность 6-бит функцией <a href="#">FFT_Fwd256Set6bit</a> () .

Возвращает

void

Заметки

Использование inplace параметров не допускается ( все указатели должны быть разными)

GSrcBuffer	LDstBuffer	LBuffer	GBuffer	Shift R	clocks
L	L	L	L	-1	21.54
L	L	L	G	-1	16.60
L	L	G	L	-1	22.17
L	L	G	G	-1	20.98
L	G	L	L	-1	20.53
L	G	L	G	-1	17.56
L	G	G	L	-1	21.17
L	G	G	G	-1	21.94
G	L	L	L	-1	20.57
G	L	L	G	-1	15.64
G	L	G	L	-1	21.21
G	L	G	G	-1	20.02
G	G	L	L	-1	19.57
G	G	L	G	-1	16.59
G	G	G	L	-1	20.20
G	G	G	G	-1	20.97
L	L	L	L	0	21.51
L	L	L	G	0	16.58
L	L	G	L	0	22.15
L	L	G	G	0	20.96
L	G	L	L	0	20.51
L	G	L	G	0	17.53
L	G	G	L	0	21.14
L	G	G	G	0	21.91
G	L	L	L	0	20.55
G	L	L	G	0	15.62
G	L	G	L	0	21.19
G	L	G	G	0	20.00
G	G	L	L	0	19.54
G	G	L	G	0	16.57
G	G	G	L	0	20.18
G	G	G	G	0	20.95

## 7.87 IFFT-256

### Функции

- void [FFT\\_Inv256Set6bit](#) ()  
Устанавливает 6-битную точность вычислений
- void [FFT\\_Inv256Set7bit](#) ()  
Устанавливает 7-битную точность вычислений
- void [FFT\\_Inv256](#) (nm32sc \*GSrcBuffer, nm32sc \*GDstBuffer, void \*LBuffer, void \*GBuffer, int ShiftR1=8, int ShiftR2=-1)  
Обратное быстрое преобразование Фурье. ОБПФ-256.

### 7.87.1 Подробное описание

### 7.87.2 Функции

#### 7.87.2.1 FFT\_Inv256()

```
void FFT_Inv256 (
    nm32sc * GSrcBuffer,
    nm32sc * GDstBuffer,
    void * LBuffer,
    void * GBuffer,
    int ShiftR1 = 8,
    int ShiftR2 = -1 )
```

Обратное быстрое преобразование Фурье. ОБПФ-256.

Функция выполняет обратное дискретное комплексное 256-точечное быстрое преобразование Фурье на базе алгоритма ОБПФ по онованию 16-16.

#### Аргументы

in	GSrcBuffer	Входной массив размером 256 64-р. слов
out	GDstBuffer	Результирующий массив размером 256 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 256*3 64-р. слов
in	GBuffer	Временный массив на глобальной шине (Global Bus) размером 256*2 64-р. слов
in	ShiftR1	Промежуточный сдвиг результатов на ShiftR1 бит вправо (первая нормализация). Необходимо для предотвращения переполнения. По умолчанию равен 8
in	ShiftR2	Заключительный сдвиг результатов на ShiftR2 бит вправо (вторая нормализация) в конце вычисления обратного БПФ. По умолчанию ShiftR2 принимается равным 14 при установленной точности 7-бит с помощью функции <a href="#">FFT_Inv256Set7bit()</a> и 12 - при точности 6-бит, установленной с помощью функции <a href="#">FFT_Inv256Set6bit()</a> .

Возвращает

void

Заметки

Использование inplace параметров не допускается ( все указатели должны быть разными)

GSrcBuffer	GDstBuffer	LBuffer	GBuffer	ShiftR1	ShiftR2	clocks
------------	------------	---------	---------	---------	---------	--------

L	L	L	L	8	-1	26.76
L	L	L	L	0	-1	26.76
L	L	L	G	8	-1	19.07
L	L	L	G	0	-1	19.07
L	L	G	L	8	-1	22.87
L	L	G	L	0	-1	22.87
L	L	G	G	8	-1	20.77
L	L	G	G	0	-1	20.77
L	G	L	L	8	-1	25.75
L	G	L	L	0	-1	25.75
L	G	L	G	8	-1	18.06
L	G	L	G	0	-1	18.06
L	G	G	L	8	-1	23.82
L	G	G	L	0	-1	23.82
L	G	G	G	8	-1	21.72
L	G	G	G	0	-1	21.72
G	L	L	L	8	-1	25.80
G	L	L	L	0	-1	25.80
G	L	L	G	8	-1	18.10
G	L	L	G	0	-1	18.10
G	L	G	L	8	-1	21.91
G	L	G	L	0	-1	21.91
G	L	G	G	8	-1	19.80
G	L	G	G	0	-1	19.80
G	G	L	L	8	-1	24.79
G	G	L	L	0	-1	24.79
G	G	L	G	8	-1	17.09
G	G	L	G	0	-1	17.09
G	G	G	L	8	-1	22.86
G	G	G	L	0	-1	22.86
G	G	G	G	8	-1	20.76
G	G	G	G	0	-1	20.76
L	L	L	L	8	0	26.74
L	L	L	L	0	0	26.74
L	L	L	G	8	0	19.04
L	L	L	G	0	0	19.04
L	L	G	L	8	0	22.85
L	L	G	L	0	0	22.85
L	L	G	G	8	0	20.75
L	L	G	G	0	0	20.75
L	G	L	L	8	0	25.73
L	G	L	L	0	0	25.73
L	G	L	G	8	0	18.03
L	G	L	G	0	0	18.03
L	G	G	L	8	0	23.80
L	G	G	L	0	0	23.80
L	G	G	G	8	0	21.70
L	G	G	G	0	0	21.70
G	L	L	L	8	0	25.77
G	L	L	L	0	0	25.77
G	L	L	G	8	0	18.08
G	L	L	G	0	0	18.08
G	L	G	L	8	0	21.88
G	L	G	L	0	0	21.88
G	L	G	G	8	0	19.78
G	L	G	G	0	0	19.78
G	G	L	L	8	0	24.76
G	G	L	L	0	0	24.76
G	G	L	G	8	0	17.07
G	G	L	G	0	0	17.07
G	G	G	L	8	0	22.83

G		G		G		L		0		0		22.83
G		G		G		G		8		0		20.73
G		G		G		G		0		0		20.73



## 7.88 FFT-512

### Функции

- void [FFT\\_Fwd512Set6bit](#) ()  
Устанавливает 6-битную точность вычислений
- void [FFT\\_Fwd512Set7bit](#) ()  
Устанавливает 7-битную точность вычислений
- void [FFT\\_Fwd512](#) ([nm32sc](#) \*GSrcBuffer, [nm32sc](#) \*GDstBuffer, void \*LBuffer, void \*GBuffer, int ShiftR=-1)  
Прямое быстрое преобразование Фурье-512.

### 7.88.1 Подробное описание

### 7.88.2 Функции

#### 7.88.2.1 FFT\_Fwd512()

```
void FFT_Fwd512 (
    nm32sc * GSrcBuffer,
    nm32sc * GDstBuffer,
    void * LBuffer,
    void * GBuffer,
    int ShiftR = -1 )
```

Прямое быстрое преобразование Фурье-512.

Функция выполняет дискретное комплексное 512-точечное преобразование Фурье на базе алгоритма БПФ по основанию 2-16-16

#### Аргументы

in	GSrcBuffer	Входной массив размером 512 64-р. слов
out	GDstBuffer	Результирующий массив размером 512 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 512*2 64-р. слов
in	GBuffer	Временный массив на глобальной шине (Global Bus) размером 512*3 64-р. слов
in	ShiftR	Коэффициент нормализации, выполняет арифметический сдвиг результирующего массива на ShiftR бит вправо для получения нормализованного массива LDstBuffer. При передаче значения по умолчанию (-1) ShiftR автоматически принимается равным 14 если ранее установлена точность 7-бит функцией <a href="#">FFT_Fwd512Set7bit()</a> и 12 - если ранее установлена точность 6-бит функцией <a href="#">FFT_Fwd512Set6bit()</a> .

Возвращает

void

Заметки

Использование inplace параметров не допускается ( все указатели должны быть разными)

GSrcBuffer	GDstBuffer	LBuffer	GBuffer	ShiftR	clocks
L	L	L	L	-1	24.12
L	L	L	G	-1	19.29
L	L	G	L	-1	22.81
L	L	G	G	-1	21.62
L	G	L	L	-1	23.10
L	G	L	G	-1	18.27
L	G	G	L	-1	23.77
L	G	G	G	-1	22.58
G	L	L	L	-1	23.06
G	L	L	G	-1	18.23
G	L	G	L	-1	23.79
G	L	G	G	-1	22.60
G	G	L	L	-1	22.04
G	G	L	G	-1	17.21
G	G	G	L	-1	24.75
G	G	G	G	-1	23.56
L	L	L	L	0	24.11
L	L	L	G	0	19.28
L	L	G	L	0	22.80
L	L	G	G	0	21.61
L	G	L	L	0	23.09
L	G	L	G	0	18.26
L	G	G	L	0	23.76
L	G	G	G	0	22.57
G	L	L	L	0	23.05
G	L	L	G	0	18.22
G	L	G	L	0	23.78
G	L	G	G	0	22.59
G	G	L	L	0	22.03
G	G	L	G	0	17.20
G	G	G	L	0	24.74
G	G	G	G	0	23.55

## 7.89 IFFT-512

### Функции

- void [FFT\\_Inv512Set6bit](#) ()  
Устанавливает 6-битную точность вычислений
- void [FFT\\_Inv512Set7bit](#) ()  
Устанавливает 7-битную точность вычислений
- void [FFT\\_Inv512](#) (nm32sc \*GSrcBuffer, nm32sc \*LDstBuffer, void \*LBuffer, void \*GBuffer, int ShiftR1=9, int ShiftR2=-1)  
Обратное быстрое преобразование Фурье. ОБПФ-512.

### 7.89.1 Подробное описание

### 7.89.2 Функции

#### 7.89.2.1 FFT\_Inv512()

```
void FFT_Inv512 (
    nm32sc * GSrcBuffer,
    nm32sc * LDstBuffer,
    void * LBuffer,
    void * GBuffer,
    int ShiftR1 = 9,
    int ShiftR2 = -1 )
```

Обратное быстрое преобразование Фурье. ОБПФ-512.

Функция выполняет обратное дискретное комплексное 512-точечное быстрое преобразование Фурье на базе алгоритма ОБПФ по онованию 2-16-16.

#### Аргументы

in	GSrcBuffer	Входной массив размером 512 64-р. слов
out	LDstBuffer	Результирующий массив размером 512 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 512*3 64-р. слов
in	GBuffer	Временный массив на глобальной шине (Global Bus) размером 512*3 64-р. слов
in	ShiftR1	Промежуточный сдвиг результатов на ShiftR1 бит вправо (первая нормализация). Необходимо для предотвращения переполнения. По умолчанию равен 9
in	ShiftR2	Заключительный сдвиг результатов на ShiftR2 бит вправо (вторая нормализация) в конце вычисления обратного БПФ. По умолчанию ShiftR2 принимается равным 14 при установленной точности 7-бит с помощью функции <a href="#">FFT_Inv512Set7bit</a> () и 12 - при точности 6-бит, установленной с помощью функции <a href="#">FFT_Inv512Set6bit</a> () .

Возвращает

void

Заметки

Использование inplace параметров не допускается ( все указатели должны быть разными)

GSrcBuffer	LDstBuffer	LBuffer	GBuffer	ShiftR1	ShiftR2	clocks
------------	------------	---------	---------	---------	---------	--------

L	L	L	L	9	-1	23.41
L	L	L	L	0	-1	23.40
L	L	L	G	9	-1	19.39
L	L	L	G	0	-1	19.38
L	L	G	L	9	-1	24.86
L	L	G	L	0	-1	24.86
L	L	G	G	9	-1	26.47
L	L	G	G	0	-1	26.46
L	G	L	L	9	-1	22.39
L	G	L	L	0	-1	22.38
L	G	L	G	9	-1	20.35
L	G	L	G	0	-1	20.34
L	G	G	L	9	-1	23.84
L	G	G	L	0	-1	23.84
L	G	G	G	9	-1	27.43
L	G	G	G	0	-1	27.42
G	L	L	L	9	-1	22.35
G	L	L	L	0	-1	22.34
G	L	L	G	9	-1	18.34
G	L	L	G	0	-1	18.33
G	L	G	L	9	-1	25.84
G	L	G	L	0	-1	25.84
G	L	G	G	9	-1	27.45
G	L	G	G	0	-1	27.44
G	G	L	L	9	-1	21.33
G	G	L	L	0	-1	21.33
G	G	L	G	9	-1	19.30
G	G	L	G	0	-1	19.29
G	G	G	L	9	-1	24.83
G	G	G	L	0	-1	24.82
G	G	G	G	9	-1	28.41
G	G	G	G	0	-1	28.41
L	L	L	L	9	0	23.40
L	L	L	L	0	0	23.39
L	L	L	G	9	0	19.38
L	L	L	G	0	0	19.37
L	L	G	L	9	0	24.85
L	L	G	L	0	0	24.85
L	L	G	G	9	0	26.45
L	L	G	G	0	0	26.45
L	G	L	L	9	0	22.38
L	G	L	L	0	0	22.37
L	G	L	G	9	0	20.34
L	G	L	G	0	0	20.33
L	G	G	L	9	0	23.83
L	G	G	L	0	0	23.83
L	G	G	G	9	0	27.41
L	G	G	G	0	0	27.41
G	L	L	L	9	0	22.34
G	L	L	L	0	0	22.33
G	L	L	G	9	0	18.33
G	L	L	G	0	0	18.32
G	L	G	L	9	0	25.83
G	L	G	L	0	0	25.83
G	L	G	G	9	0	27.43
G	L	G	G	0	0	27.43
G	G	L	L	9	0	21.32
G	G	L	L	0	0	21.31
G	G	L	G	9	0	19.29
G	G	L	G	0	0	19.28
G	G	G	L	9	0	24.81

G		G		G		L		0		0		24.81
G		G		G		G		9		0		28.40
G		G		G		G		0		0		28.39

## 7.90 FFT-1024

### Функции

- void [FFT\\_Fwd1024Set6bit](#) ()  
Устанавливает 6-битную точность вычислений
- void [FFT\\_Fwd1024Set7bit](#) ()  
Устанавливает 7-битную точность вычислений
- void [FFT\\_Fwd1024](#) (nm32sc \*GSrcBuffer, nm32sc \*LDstBuffer, void \*LBuffer, void \*GBuffer, int ShiftR=-1)  
Прямое быстрое преобразование Фурье-1024.

### 7.90.1 Подробное описание

### 7.90.2 Функции

#### 7.90.2.1 FFT\_Fwd1024()

```
void FFT_Fwd1024 (
    nm32sc * GSrcBuffer,
    nm32sc * LDstBuffer,
    void * LBuffer,
    void * GBuffer,
    int ShiftR = -1 )
```

Прямое быстрое преобразование Фурье-1024.

Функция выполняет дискретное комплексное 1024-точечное преобразование Фурье на базе алгоритма БПФ по основанию 2-32-16

#### Аргументы

in	GSrcBuffer	Входной массив размером 1024 64-р. слов
out	LDstBuffer	Результирующий массив размером 1024 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 1024*3 64-р. слов
in	GBuffer	Временный массив на глобальной шине (Global Bus) размером 1024*3 64-р. слов
in	ShiftR	Коэффициент нормализации, выполняет арифметический сдвиг результирующего массива на ShiftR бит вправо для получения нормализованного массива LDstBuffer. При передаче значения по умолчанию (-1) ShiftR автоматически принимается равным 14 если ранее установлена точность 7-бит функцией <a href="#">FFT_Fwd1024Set7bit()</a> и 12 - если ранее установлена точность 6-бит функцией <a href="#">FFT_Fwd1024Set6bit()</a> .

Возвращает

void

Заметки

Использование inplace параметров не допускается ( все указатели должны быть разными)

GSrcBuffer	LDstBuffer	LBuffer	GBuffer	ShiftR	clocks
-----					
L	L	L	L	-1	22.55
L	L	L	G	-1	20.52
L	L	G	L	-1	26.08
L	L	G	G	-1	25.93
L	G	L	L	-1	21.53
L	G	L	G	-1	21.48
L	G	G	L	-1	25.06
L	G	G	G	-1	26.89
G	L	L	L	-1	21.55
G	L	L	G	-1	19.51
G	L	G	L	-1	27.03
G	L	G	G	-1	26.87
G	G	L	L	-1	20.52
G	G	L	G	-1	20.48
G	G	G	L	-1	26.00
G	G	G	G	-1	27.84
L	L	L	L	0	22.55
L	L	L	G	0	20.51
L	L	G	L	0	26.08
L	L	G	G	0	25.92
L	G	L	L	0	21.52
L	G	L	G	0	21.48
L	G	G	L	0	25.05
L	G	G	G	0	26.89
G	L	L	L	0	21.54
G	L	L	G	0	19.51
G	L	G	L	0	27.02
G	L	G	G	0	26.87
G	G	L	L	0	20.52
G	G	L	G	0	20.47
G	G	G	L	0	26.00
G	G	G	G	0	27.83

## 7.91 IFFT-1024

### Функции

- void [FFT\\_Inv1024Set6bit](#) ()  
Устанавливает 6-битную точность вычислений
- void [FFT\\_Inv1024Set7bit](#) ()  
Устанавливает 7-битную точность вычислений
- void [FFT\\_Inv1024](#) (nm32sc \*GSrcBuffer, nm32sc \*GDstBuffer, void \*LBuffer, void \*GBuffer, int ShiftR1=10, int ShiftR2=-1)  
Обратное быстрое преобразование Фурье. ОБПФ-1024.

### 7.91.1 Подробное описание

### 7.91.2 Функции

#### 7.91.2.1 FFT\_Inv1024()

```
void FFT_Inv1024 (
    nm32sc * GSrcBuffer,
    nm32sc * GDstBuffer,
    void * LBuffer,
    void * GBuffer,
    int ShiftR1 = 10,
    int ShiftR2 = -1 )
```

Обратное быстрое преобразование Фурье. ОБПФ-1024.

Функция выполняет обратное дискретное комплексное 1024-точечное быстрое преобразование Фурье на базе алгоритма ОБПФ по онованию 2-16-16.

#### Аргументы

in	GSrcBuffer	Входной массив размером 1024 64-р. слов
out	GDstBuffer	Результирующий массив размером 1024 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 1024*3 64-р. слов
in	GBuffer	Временный массив на глобальной шине (Global Bus) размером 1024*3 64-р. слов
in	ShiftR1	Промежуточный сдвиг результатов на ShiftR1 бит вправо (первая нормализация). Необходимо для предотвращения переполнения. По умолчанию равен 10
in	ShiftR2	Заключительный сдвиг результатов на ShiftR2 бит вправо (вторая нормализация) в конце вычисления обратного БПФ. По умолчанию ShiftR2 принимается равным 14 при установленной точности 7-бит с помощью функции <a href="#">FFT_Inv1024Set7bit</a> () и 12 - при точности 6-бит, установленной с помощью функции <a href="#">FFT_Inv1024Set6bit</a> ()



Возвращает

void

Заметки

Использование inplace параметров не допускается ( все указатели должны быть разными)

GSrcBuffer	GDstBuffer	LBuffer	GBuffer	ShiftR1	ShiftR2	clocks
------------	------------	---------	---------	---------	---------	--------

L	L	L	L	10	-1	28.82
L	L	L	L	0	-1	28.82
L	L	L	G	10	-1	23.90
L	L	L	G	0	-1	23.90
L	L	G	L	10	-1	26.60
L	L	G	L	0	-1	26.60
L	L	G	G	10	-1	25.55
L	L	G	G	0	-1	25.55
L	G	L	L	10	-1	27.80
L	G	L	L	0	-1	27.80
L	G	L	G	10	-1	22.88
L	G	L	G	0	-1	22.88
L	G	G	L	10	-1	27.57
L	G	G	L	0	-1	27.57
L	G	G	G	10	-1	26.52
L	G	G	G	0	-1	26.52
G	L	L	L	10	-1	27.81
G	L	L	L	0	-1	27.81
G	L	L	G	10	-1	22.89
G	L	L	G	0	-1	22.89
G	L	G	L	10	-1	27.55
G	L	G	L	0	-1	27.55
G	L	G	G	10	-1	26.50
G	L	G	G	0	-1	26.50
G	G	L	L	10	-1	26.79
G	G	L	L	0	-1	26.79
G	G	L	G	10	-1	21.87
G	G	L	G	0	-1	21.87
G	G	G	L	10	-1	28.51
G	G	G	L	0	-1	28.51
G	G	G	G	10	-1	27.46
G	G	G	G	0	-1	27.46
L	L	L	L	10	0	28.82
L	L	L	L	0	0	28.82
L	L	L	G	10	0	23.90
L	L	L	G	0	0	23.90
L	L	G	L	10	0	26.60
L	L	G	L	0	0	26.60
L	L	G	G	10	0	25.54
L	L	G	G	0	0	25.54
L	G	L	L	10	0	27.79
L	G	L	L	0	0	27.79
L	G	L	G	10	0	22.87
L	G	L	G	0	0	22.87
L	G	G	L	10	0	27.56
L	G	G	L	0	0	27.56
L	G	G	G	10	0	26.51
L	G	G	G	0	0	26.51
G	L	L	L	10	0	27.81
G	L	L	L	0	0	27.81
G	L	L	G	10	0	22.89
G	L	L	G	0	0	22.89
G	L	G	L	10	0	27.54
G	L	G	L	0	0	27.54
G	L	G	G	10	0	26.49
G	L	G	G	0	0	26.49
G	G	L	L	10	0	26.78
G	G	L	L	0	0	26.78
G	G	L	G	10	0	21.86
G	G	L	G	0	0	21.86
G	G	G	L	10	0	28.51

G		G		G		L		0		0		28.51
G		G		G		G		10		0		27.46
G		G		G		G		0		0		27.46

## 7.92 FFT-2048

### Функции

- void [FFT\\_Fwd2048Set6bit](#) ()  
Устанавливает 6-битную точность вычислений
- void [FFT\\_Fwd2048Set7bit](#) ()  
Устанавливает 7-битную точность вычислений
- void [FFT\\_Fwd2048](#) ([nm32sc](#) \*GSrcBuffer, [nm32sc](#) \*GDstBuffer, void \*LBuffer, int ShiftR=-1)  
Прямое быстрое преобразование Фурье-2048.

### 7.92.1 Подробное описание

### 7.92.2 Функции

#### 7.92.2.1 FFT\_Fwd2048()

```
void FFT_Fwd2048 (
    nm32sc * GSrcBuffer,
    nm32sc * GDstBuffer,
    void * LBuffer,
    int ShiftR = -1 )
```

Прямое быстрое преобразование Фурье-2048.

Функция выполняет дискретное комплексное 2048-точечное преобразование Фурье на базе алгоритма БПФ по основанию 2-32-32

#### Аргументы

in	GSrcBuffer	Входной массив размером 2048 64-р. слов
out	GDstBuffer	Результирующий массив размером 2048 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 2048*4 64-р. слов
in	ShiftR	Коэффициент нормализации, выполняет арифметический сдвиг результирующего массива на ShiftR бит вправо для получения нормализованного массива GDstBuffer. При передаче значения по умолчанию (-1) ShiftR автоматически принимается равным 14 если ранее установлена точность 7-бит функцией <a href="#">FFT_Fwd2048Set7bit()</a> и 12 - если ранее установлена точность 6-бит функцией <a href="#">FFT_Fwd2048Set6bit()</a> .

Возвращает

void

## Заметки

Использование inplace параметров не допускается ( все указатели должны быть разными)

GSrcBuffer	GDstBuffer	LBuffer	Shift R	clocks
<hr/>				
L	L	L	-1	26.41
L	L	G	-1	28.97
L	G	L	-1	25.38
L	G	G	-1	29.93
G	L	L	-1	25.38
G	L	G	-1	29.93
G	G	L	-1	24.36
G	G	G	-1	30.89
L	L	L	0	26.40
L	L	G	0	28.96
L	G	L	0	25.38
L	G	G	0	29.93
G	L	L	0	25.38
G	L	G	0	29.93
G	G	L	0	24.35
G	G	G	0	30.89

## 7.93 IFFT-2048

### Функции

- void [FFT\\_Inv2048Set6bit](#) ()  
Устанавливает 6-битную точность вычислений
- void [FFT\\_Inv2048Set7bit](#) ()  
Устанавливает 7-битную точность вычислений
- void [FFT\\_Inv2048](#) (nm32sc \*GSrcBuffer, nm32sc \*LDstBuffer, void \*LBuffer, void \*GBuffer, int ShiftR1=11, int ShiftR2=-1)  
Обратное быстрое преобразование Фурье. ОБПФ-2048.

### 7.93.1 Подробное описание

### 7.93.2 Функции

#### 7.93.2.1 FFT\_Inv2048()

```
void FFT_Inv2048 (
    nm32sc * GSrcBuffer,
    nm32sc * LDstBuffer,
    void * LBuffer,
    void * GBuffer,
    int ShiftR1 = 11,
    int ShiftR2 = -1 )
```

Обратное быстрое преобразование Фурье. ОБПФ-2048.

Функция выполняет обратное дискретное комплексное 2048-точечное быстрое преобразование Фурье на базе алгоритма ОБПФ по онованию 2-32-32.

#### Аргументы

in	GSrcBuffer	Входной массив размером 2048 64-р. слов
out	LDstBuffer	Результирующий массив размером 2048 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 2048*4 64-р. слов
in	GBuffer	Временный массив на глобальной шине (Global Bus) размером 2048*4 64-р. слов
in	ShiftR1	Промежуточный сдвиг результатов на ShiftR1 бит вправо (первая нормализация). Необходимо для предотвращения переполнения. По умолчанию равен 11
in	ShiftR2	Заключительный сдвиг результатов на ShiftR2 бит вправо (вторая нормализация) в конце вычисления обратного БПФ. По умолчанию ShiftR2 принимается равным 14 при установленной точности 7-бит с помощью функции <a href="#">FFT_Inv2048Set7bit</a> () и 12 - при точности 6-бит, установленной с помощью функции <a href="#">FFT_Inv2048Set6bit</a> ()).

Возвращает

void

Заметки

Использование inplace параметров не допускается ( все указатели должны быть разными)

GSrcBuffer| LDstBuffer| LBuffer | GBuffer | ShiftR1 | ShiftR2 | clocks

L	L	L	L	11	-1	30.58
L	L	L	L	0	-1	30.58
L	L	L	G	11	-1	26.52
L	L	L	G	0	-1	26.52
L	L	G	L	11	-1	31.33
L	L	G	L	0	-1	31.33
L	L	G	G	11	-1	29.26
L	L	G	G	0	-1	29.26
L	G	L	L	11	-1	29.56
L	G	L	L	0	-1	29.55
L	G	L	G	11	-1	27.48
L	G	L	G	0	-1	27.48
L	G	G	L	11	-1	30.30
L	G	G	L	0	-1	30.30
L	G	G	G	11	-1	30.22
L	G	G	G	0	-1	30.22
G	L	L	L	11	-1	29.56
G	L	L	L	0	-1	29.56
G	L	L	G	11	-1	25.49
G	L	L	G	0	-1	25.49
G	L	G	L	11	-1	32.29
G	L	G	L	0	-1	32.29
G	L	G	G	11	-1	30.22
G	L	G	G	0	-1	30.22
G	G	L	L	11	-1	28.53
G	G	L	L	0	-1	28.53
G	G	L	G	11	-1	26.46
G	G	L	G	0	-1	26.46
G	G	G	L	11	-1	31.26
G	G	G	L	0	-1	31.26
G	G	G	G	11	-1	31.19
G	G	G	G	0	-1	31.18
L	L	L	L	11	0	30.58
L	L	L	L	0	0	30.58
L	L	L	G	11	0	26.51
L	L	L	G	0	0	26.51
L	L	G	L	11	0	31.33
L	L	G	L	0	0	31.32
L	L	G	G	11	0	29.25
L	L	G	G	0	0	29.25
L	G	L	L	11	0	29.55
L	G	L	L	0	0	29.55
L	G	L	G	11	0	27.48
L	G	L	G	0	0	27.48
L	G	G	L	11	0	30.30
L	G	G	L	0	0	30.30
L	G	G	G	11	0	30.22
L	G	G	G	0	0	30.22
G	L	L	L	11	0	29.56
G	L	L	L	0	0	29.56
G	L	L	G	11	0	25.49
G	L	L	G	0	0	25.49
G	L	G	L	11	0	32.29
G	L	G	L	0	0	32.29
G	L	G	G	11	0	30.22
G	L	G	G	0	0	30.22
G	G	L	L	11	0	28.53
G	G	L	L	0	0	28.53
G	G	L	G	11	0	26.46
G	G	L	G	0	0	26.46
G	G	G	L	11	0	31.26

G		G		G		L		0		0		31.26
G		G		G		G		11		0		31.18
G		G		G		G		0		0		31.18

## 7.94 FFT-4096

### Функции

- void `FFT_Fwd4096` (`nm32sc` \*GSrcBuffer, `nm32sc` \*GDstBuffer, void \*LBuffer, void \*GBuffer)

Прямое быстрое преобразование Фурье-4096.

#### 7.94.1 Подробное описание

#### 7.94.2 Функции

##### 7.94.2.1 FFT\_Fwd4096()

```
void FFT_Fwd4096 (
    nm32sc * GSrcBuffer,
    nm32sc * GDstBuffer,
    void * LBuffer,
    void * GBuffer )
```

Прямое быстрое преобразование Фурье-4096.

Функция выполняет дискретное комплексное 4096-точечное преобразование Фурье на базе алгоритма БПФ по основанию 16-16-16

#### Аргументы

in	GSrcBuffer	Входной массив размером 4096 64-р. слов
out	GDstBuffer	Результирующий массив размером 4096 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 4096*2 64-р. слов
in	GBuffer	Временный массив на глобальной шине (Global Bus) размером 4096*3 64-р. слов

#### Возвращает

void

#### Заметки

Использование inplace параметров не допускается (все указатели должны быть разными) Диапазон входных данных: -4096..4096

\perf

GSrcBuffer | GDstBuffer | LBuffer | GBuffer | CLOCKS

L		L		L		L		38.25
L		L		L		G		26.82
L		L		G		L		32.21
L		L		G		G		30.74



L		G		L		L		37.22
L		G		L		G		25.79
L		G		G		L		33.17
L		G		G		G		31.71
G		L		L		L		37.26
G		L		L		G		25.83
G		L		G		L		31.21
G		L		G		G		29.75
G		G		L		L		36.23
G		G		L		G		24.80
G		G		G		L		32.18
G		G		G		G		30.71

## 7.95 IFFT-4096

### Функции

- void `FFT_Inv4096` (`nm32sc *GSrcBuffer`, `nm32sc *GDstBuffer`, `void *LBuffer`, `void *GBuffer`)

Обратное быстрое преобразование Фурье. ОБПФ-4096.

#### 7.95.1 Подробное описание

#### 7.95.2 Функции

##### 7.95.2.1 `FFT_Inv4096()`

```
void FFT_Inv4096 (
    nm32sc * GSrcBuffer,
    nm32sc * GDstBuffer,
    void * LBuffer,
    void * GBuffer )
```

Обратное быстрое преобразование Фурье. ОБПФ-4096.

Функция выполняет обратное дискретное комплексное 4096-точечное быстрое преобразование Фурье на базе алгоритма ОБПФ по онованию 16-16-16.

#### Аргументы

in	GSrcBuffer	Входной массив размером 4096 64-р. слов
out	GDstBuffer	Результирующий массив размером 4096 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 4096*2 64-р. слов
in	GBuffer	Временный массив на глобальной шине (Global Bus) размером 4096*3 64-р. слов

#### Возвращает

void

#### Заметки

Использование inplace параметров не допускается (все указатели должны быть разными) Диапазон входных данных: -4096..4096

\perf

GSrcBuffer | GDstBuffer | LBuffer | GBuffer | Clocks

L		L		L		L		38.25
L		L		L		G		26.82
L		L		G		L		32.21
L		L		G		G		30.74

L		G		L		L		37.22
L		G		L		G		25.79
L		G		G		L		33.17
L		G		G		G		31.71
G		L		L		L		37.26
G		L		L		G		25.83
G		L		G		L		31.21
G		L		G		G		29.75
G		G		L		L		36.23
G		G		L		G		24.80
G		G		G		L		32.18
G		G		G		G		30.71

## 7.96 FFT-8192

### Функции

- void `FFT_Fwd8192` (`nm32sc` \*LSrcBuffer, `nm32sc` \*GDstBuffer, void \*LBuffer, void \*GBuffer)  
Прямое быстрое преобразование Фурье-8192.

### 7.96.1 Подробное описание

### 7.96.2 Функции

#### 7.96.2.1 FFT\_Fwd8192()

```
void FFT_Fwd8192 (
    nm32sc * LSrcBuffer,
    nm32sc * GDstBuffer,
    void * LBuffer,
    void * GBuffer )
```

Прямое быстрое преобразование Фурье-8192.

Функция выполняет дискретное комплексное 8192-точечное преобразование Фурье на базе алгоритма БПФ по основанию 2-16-16-16

#### Аргументы

in	LSrcBuffer	Входной массив размером 8192 64-р. слов
out	GDstBuffer	Результирующий массив размером 8192 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 8192 64-р. слов
in	GBuffer	Временный массив на глобальной шине (Global Bus) размером 8192*3 64-р. слов

#### Возвращает

void

#### Заметки

Использование inplace параметров не допускается (все указатели должны быть разными) Диапазон входных данных: -2048..2048

\perf

LSrcBuffer | GDstBuffer | LBuffer | GBuffer | Clocks

L	L	L	L	40.70
L	L	L	G	28.89
L	L	G	L	35.55
L	L	G	G	31.88
L	G	L	L	39.67

L		G		L		G		27.86
L		G		G		L		36.52
L		G		G		G		32.85
G		L		L		L		40.17
G		L		L		G		29.40
G		L		G		L		35.02
G		L		G		G		32.39
G		G		L		L		39.14
G		G		L		G		28.37
G		G		G		L		35.99
G		G		G		G		33.36

## 7.97 IFFT-8192

### Функции

- void `FFT_Inv8192` (`nm32sc` \*LSrcBuffer, `nm32sc` \*GDstBuffer, void \*LBuffer, void \*GBuffer)

Обратное быстрое преобразование Фурье. ОБПФ-8192.

#### 7.97.1 Подробное описание

#### 7.97.2 Функции

##### 7.97.2.1 `FFT_Inv8192()`

```
void FFT_Inv8192 (
    nm32sc * LSrcBuffer,
    nm32sc * GDstBuffer,
    void * LBuffer,
    void * GBuffer )
```

Обратное быстрое преобразование Фурье. ОБПФ-8192.

Функция выполняет обратное дискретное комплексное 8192-точечное быстрое преобразование Фурье на базе алгоритма ОБПФ по онованию 2-16-16-16.

#### Аргументы

in	LSrcBuffer	Входной массив размером 8192 64-р. слов
out	GDstBuffer	Результирующий массив размером 8192 64-р. слов
in	LBuffer	Временный массив на локальной шине (Local Bus) размером 8192 64-р. слов
in	GBuffer	Временный массив на глобальной шине (Global Bus) размером 8192*3 64-р. слов

#### Возвращает

void

#### Заметки

Использование inplace параметров не допускается (все указатели должны быть разными) Диапазон входных данных: -2048..2048

\perf

GSrcBuffer | GDstBuffer | LBuffer | GBuffer | Clocks

L		L		L		L		40.70
L		L		L		G		28.89

L		L		G		L		35.55
L		L		G		G		31.88
L		G		L		L		39.67
L		G		L		G		27.86
L		G		G		L		36.52
L		G		G		G		32.85
G		L		L		L		40.17
G		L		L		G		29.40
G		L		G		L		35.02
G		L		G		G		32.39
G		G		L		L		39.14
G		G		L		G		28.37
G		G		G		L		35.99
G		G		G		G		33.36

## 7.98 Свертка

### Группы

- [SIG\\_XCorr](#)

Свертка двух векторов.

#### 7.98.1 Подробное описание



## 7.99 Масочная фильтрация

### Группы

- [SIG\\_Median3](#)  
Вычисление медианы трех чисел
- [КИХ-фильтрация](#)  
Одномерная КИХ-фильтрация.

### 7.99.1 Подробное описание

## 7.100 Изменение размеров

### Группы

- [SIG\\_ResampleDown2](#)  
Уменьшение числа отсчетов в двое.
- [SIG\\_ResampleUp3Down2](#)  
Передискретизации сигнала в  $3/2$  раза  
Передискретизации сигнала осуществляется методом Polyphase:
- [SIG\\_CreateResample](#)  
Создание ядра для функции передискретизации SIG\_Resample().  
Функции выделяют память и инициализируют таблицы весовых коэффициентов для использования в функциях передискретизации.
- [SIG\\_SetResample](#)  
Создание ядра для функции передискретизации SIG\_Resample().  
Функции инициализируют таблицы весовых коэффициентов для использования в функциях передискретизации.
- [SIG\\_Resample\\_perf](#)  
Функции для оценки производительности функций фильтрации SIG\_Resample()  
Функция эквивалентно следующим псевдоинструкциям:

### 7.100.1 Подробное описание

## 7.101 Быстрое преобразование Фурье

## 7.102 SIG\_XCorr

Свертка двух векторов.

### Функции

- void SIG\_XCorr\_32s (nm32s \*pSrcVec, int nSrcVecSize, nm32s \*pKernel, int nKernelSize, nm32s \*pDstVec, void \*pTmpBuf)
- void SIG\_XCorr\_16s32s (nm16s \*pSrcVec, int nSrcVecSize, nm32s \*pKernel, int nKernelSize, nm32s \*pDstVec, void \*pTmpBuf)
- void SIG\_XCorr\_8s32s (nm8s \*pSrcVec, int nSrcVecSize, nm32s \*pKernel, int nKernelSize, nm32s \*pDstVec, void \*pTmpBuf)

### 7.102.1 Подробное описание

Свертка двух векторов.

$$DstVec_i = \sum_{j=0}^{nKernelSize-1} pSrcVec[i+j] \cdot pKernel[j]$$

$$i = \overline{0 \dots nSrcVecSize - nKernelSize + 1}$$

### Аргументы

pSrcVec	Входной вектор.
pKernel	Вектор коэффициентов окна свертки.
pTmpBuf	Указатель на временный буфер размера 2*nKernelSize + 32 32-битных слов;

2\*nKernelSize + 32 32-bit words;

### Аргументы

nKernelSize	Размер окна свертки [1,2,3,4...nSrcVecSize-1].
nSrcVecSize	Размер входного вектора в элементах .Размер кратен 8,4 или 2 согласно типу данных.

### Возвращаемые значения

pDstVec	Результирующий вектор, размером nSrcVecSize-nKernelSize+1. после которого могут записываться еще до 7 незначачих 32p слова.
---------	---

## Заметки

По сути функции осуществляют фильтрацию данных окном свертки.

## 7.103 SIG\_Median3

Вычисление медианы трех чисел

Функции

- `int SIG_Median3 (int a, int b, int c)`
- `uint32b SIG_Median3 (uint32b a, uint32b b, uint32b c)`

### 7.103.1 Подробное описание

Вычисление медианы трех чисел

Аргументы

a	Первое число
b	Второе число
c	Третье число

Возвращает

Медианное значение

## 7.104 КИХ-фильтрация

Одномерная КИХ-фильтрация.

### Группы

- [nmppsFIR\\_Xs](#)  
Одномерная КИХ-фильтрация
- [nmppsFIRInit\\_Xs](#)  
Инициализация функции одномерной фильтрации
- [nmppsFIRInitAlloc\\_Xs](#)  
Выделение и инициализация служебной структуры для функции одномерной фильтрации
- [nmppsFIRGetStateSize\\_Xs](#)  
Возвращает размер памяти (в 32р.-словах) необходимый для хранения служебной структуры
- [nmppsFIRFree](#)  
освобождает структуру pState в куче

### 7.104.1 Подробное описание

Одномерная КИХ-фильтрация.

## 7.105 nmppsFIR\_Xs

Одномерная КИХ-фильтрация

### Функции

- void nmppsFIR\_8s (nm8s \*pSrc, nm8s \*pDst, int srcSize, NmppsFIRState \*pState)
- void nmppsFIR\_8s16s (nm8s \*pSrc, nm16s \*pDst, int srcSize, NmppsFIRState \*pState)
- void nmppsFIR\_8s32s (nm8s \*pSrc, nm32s \*pDst, int srcSize, NmppsFIRState \*pState)
- void nmppsFIR\_16s (nm16s \*pSrc, nm16s \*pDst, int srcSize, NmppsFIRState \*pState)
- void nmppsFIR\_16s32s (nm16s \*pSrc, nm32s \*pDst, int srcSize, NmppsFIRState \*pState)
- void nmppsFIR\_32s (nm32s \*pSrc, nm32s \*pDst, int srcSize, NmppsFIRState \*pState)

### 7.105.1 Подробное описание

Одномерная КИХ-фильтрация

#### Аргументы

in	pSrc	Входной вектор
in	srcSize	Размер входного вектора в элементах. Размер вектора должен быть кратен количеству элементов в 64-р. слове.
out	pDst	Результирующий вектор
in	NmppsFIRState	Служебная структура, содержащая весовые коэффициенты фильтра во внутреннем формате.

#### Заметки

Инициализация служебной структуры производится соответствующей функцией nmppsFIR↔Init\_Xs() или nmppsFIRInitAlloc\_Xs(). Максимальная производительность достигается при размещении pSrc, pDst и pPstate в разных банках памяти .



## 7.106 nmppsFIRInit\_Xs

Инициализация функции одномерной фильтрации

Функции

- int nmppsFIRInit\_8s (NmppsFIRState \*pState, int \*pTaps, int tapsLen)
- int nmppsFIRInit\_8s16s (NmppsFIRState \*pState, int \*pTaps, int tapsLen)
- int nmppsFIRInit\_8s32s (NmppsFIRState \*pState, int \*pTaps, int tapsLen)
- int nmppsFIRInit\_16s (NmppsFIRState \*pState, int \*pTaps, int tapsLen)
- int nmppsFIRInit\_16s32s (NmppsFIRState \*pState, int \*pTaps, int tapsLen)
- int nmppsFIRInit\_32s (NmppsFIRState \*pState, int \*pTaps, int tapsLen)

### 7.106.1 Подробное описание

Инициализация функции одномерной фильтрации

Функция преобразует таблицу весовых коэффициентов окна фильтра во внутренний формат

Аргументы

in	pTaps	Указатель на коэффициенты фильтра
in	tapsLen	Размер окна фильтра. tapsLen=[3,5,7,9....]
out	pState	Указатель на служебную структуру, содержащую весовые коэффициенты фильтра во внутреннем формате. Размер памяти (в 32р.-словах) необходимый для хранения служебной структуры можно получить с помощью функции nmppsFIRGetSize_Xs

Возвращает

Размер проинициализированной структуры pState в 32р. словах

## 7.107 nmppsFIRInitAlloc\_Xs

Выделение и инициализация служебной структуры для функции одномерной фильтрации

### Функции

- int nmppsFIRInitAlloc\_8s (NmppsFIRState \*\*ppState, int \*pTaps, int tapsLen)
- int nmppsFIRInitAlloc\_8s16s (NmppsFIRState \*\*ppState, int \*pTaps, int tapsLen)
- int nmppsFIRInitAlloc\_8s32s (NmppsFIRState \*\*ppState, int \*pTaps, int tapsLen)
- int nmppsFIRInitAlloc\_16s (NmppsFIRState \*\*ppState, int \*pTaps, int tapsLen)
- int nmppsFIRInitAlloc\_16s32s (NmppsFIRState \*\*ppState, int \*pTaps, int tapsLen)
- int nmppsFIRInitAlloc\_32s (NmppsFIRState \*\*ppState, int \*pTaps, int tapsLen)

### 7.107.1 Подробное описание

Выделение и инициализация служебной структуры для функции одномерной фильтрации

Функция выделяет структуру в куче и преобразует таблицу весовых коэффициентов окна фильтра во внутренний формат

#### Аргументы

in	pTaps	Указатель на коэффициенты фильтра
in	tapsLen	Размер окна фильтра. nWeights=[3,5,7,9....]

#### Возвращаемые значения

[out]	ppState Возвращает указатель на служебную структуру, содержащую весовые коэффициенты фильтра во внутреннем формате
-------	--

#### Возвращает

Размер проинициализированной структуры pState в 32р. словах

## 7.108 nmppsFIRGetStateSize\_Xs

Возвращает размер памяти (в 32р.-словах) необходимый для хранения служебной структуры

### Функции

- int nmppsFIRGetStateSize\_8s (int tapsLen)
- int nmppsFIRGetStateSize\_8s16s (int tapsLen)
- int nmppsFIRGetStateSize\_8s32s (int tapsLen)
- int nmppsFIRGetStateSize\_16s (int tapsLen)
- int nmppsFIRGetStateSize\_16s32s (int tapsLen)
- int nmppsFIRGetStateSize\_32s (int tapsLen)

### 7.108.1 Подробное описание

Возвращает размер памяти (в 32р.-словах) необходимый для хранения служебной структуры

#### Аргументы

in	tapsLen	Размер окна фильтра. tapsLen=[3,5,7,9....]
----	---------	--

#### Возвращает

Возвращает размер памяти (в 32р.-словах), необходимый для хранения служебной структуры  
NmppsFIRState

## 7.109 nmppsFIRFree

освобождает структуру pState в куче

### Функции

- void nmppsFIRFree (NmppsFIRState \*pState)

### 7.109.1 Подробное описание

освобождает структуру pState в куче

### Аргументы

in	pState	указатель на служебную структуру NmppsFIRState
----	--------	--

## 7.110 SIG\_ResampleDown2

Уменьшение числа отсчетов в двое.

### Функции

- void SIG\_ResampleDown2\_8u (nm8u7b \*pSrcVec, nm8u7b \*pDstVec, int nSrcVecSize, nm64s \*pKernel)
- void SIG\_ResampleDown2\_16u (nm16u15b \*pSrcVec, nm16u15b \*pDstVec, int nSrcVecSize, nm64s \*pKernel)

### 7.110.1 Подробное описание

Уменьшение числа отсчетов в двое.

$$pDstVec = \frac{1}{2} (pSrcVec(2 * x) + pSrcVec(2 * x + 1))$$

### Аргументы

pSrcVec	Входной сигнал.
nSize	Размер массива входных данных.

### Возвращаемые значения

pDstVec	Результирующий сигнал.
---------	------------------------

### Заметки

Для того чтобы избежать переполнения при усреднении, динамический диапазон исходного изображения должен принадлежать диапазону, определенному типом.

## 7.111 SIG\_ResampleUp3Down2

Передискретизации сигнала в 3/2 раза

Передискретизации сигнала осуществляется методом Polyphase:

.

### Функции

- void SIG\_ResampleUp3Down2 (nm8s \*pSrcVec, nm16s \*pDstVec, int nSrcVecSize, nm64s \*pKernel)

### 7.111.1 Подробное описание

Передискретизации сигнала в 3/2 раза

Передискретизации сигнала осуществляется методом Polyphase:

.

- Между отсчетами входного сигнала вставляется по 2 нуля
- Полученный сигнал пропускается через фильтр ФНЧ. Длина фильтра 17
- Из полученного сигнала выбирается каждый 2 отсчет

### Аргументы

pSrcVec	Входной вектор. Элементы вектора - целые числа со знаком.
nSrcVecSize	Размер входного вектора.

### Возвращаемые значения

pDstVec	Результирующий вектор. Элементы вектора возвращаются в формате fixed-point: [12.4] (целая часть-12 бит, дробная -4бита)
---------	---

### Возвращает

void

## 7.112 SIG\_CreateResample

Создание ядра для функции передискретизации SIG\_Resample().

Функции выделяют память и инициализируют таблицы весовых коэффициентов для использования в функциях передискретизации.

### Функции

- void SIG\_CreateResampleUp3Down2\_8s16s (nm64s \*\*pKernel, int nHint=MEM\_LOCAL)
- void SIG\_CreateResampleDown2\_8u8u (nm64s \*\*pKernel, int nHint=MEM\_LOCAL)
- void SIG\_CreateResampleDown2\_16u16u (nm64s \*\*pKernel, int nHint=MEM\_LOCAL)

### 7.112.1 Подробное описание

Создание ядра для функции передискретизации SIG\_Resample().

Функции выделяют память и инициализируют таблицы весовых коэффициентов для использования в функциях передискретизации.

#### Аргументы

nHint	Определяет память(Local или Global) в которой создается служебная структура. nHint=[MEM_LOCAL, MEM_GLOBAL].
-------	--

#### Возвращаемые значения

pKernel	Указатель на служебную структуру, содержащую весовые коэффициенты фильтра во внутреннем формате.
---------	--

#### Заметки

Используется перед вызовом функции SIG\_Filter.

## 7.113 SIG\_SetResample

Создание ядра для функции передискретизации SIG\_Resample().

Функции инициализируют таблицы весовых коэффициентов для использования в функциях передискретизации.

### Функции

- int SIG\_SetResampleUp3Down2\_8s16s (nm64s \*pKernel)
- int SIG\_SetResampleDown2\_8u8u (nm64s \*pKernel)
- int SIG\_SetResampleDown2\_16u16u (nm64s \*pKernel)

### 7.113.1 Подробное описание

Создание ядра для функции передискретизации SIG\_Resample().

Функции инициализируют таблицы весовых коэффициентов для использования в функциях передискретизации.

### Аргументы

pKernel	Указатель на служебную структуру, содержащую весовые коэффициенты фильтра во внутреннем формате.
---------	--

### Возвращает

Размер проинициализированной структуры pKernel в 32р. словах

### Заметки

Используется перед вызовом функции SIG\_Filter.

Используется перед вызовом функции SIG\_Resample().



## 7.114 SIG\_Resample\_perf

Функции для оценки производительности функций фильтрации SIG\_Resample()

Функция эквивалентно следующим псевдоинструкциям:

Функции

- void SIG\_ResampleUp3Down2\_perf (nm8s \*pSrcVec, nm16s \*pDstVec, int nSrcVecSize, nm64s \*pKernel)
- void SIG\_ResampleDown2\_perf\_8u (nm8u7b \*pSrcVec, nm8u7b \*pDstVec, int nSrcVecSize, nm64s \*pKernel)
- void SIG\_ResampleDown2\_perf\_16u (nm16u15b \*pSrcVec, nm16u15b \*pDstVec, int nSrcVecSize, nm64s \*pKernel)

### 7.114.1 Подробное описание

Функции для оценки производительности функций фильтрации SIG\_Resample()

Функция эквивалентно следующим псевдоинструкциям:

```
void SIG_Resample***_perf(nm16u15b* pSrcVec, nm16u15b* pDstVec, int nSrcVecSize,
nm64s* pKernel);
{
    SIG_SetResample***_***(pKernel);
    clock_t t0,t1;
    t0=clock();
    SIG_Resample***(pSrcVec, pDstVec, nSize,pKernel);
    t1=clock();
    exit(t1-t0);
}
```

\retval pKernel

Указатель на служебную структуру, содержащую весовые коэффициенты фильтра во внутреннем формате.

Аргументы

pSrcVec	Входной вектор.
nSrcVecSize	Размер входного вектора в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Заметки

Инициализация служебной структуры производится соответствующей функцией SIG\_Set↵Filter() и SIG\_CreateFilter().

## 7.115 Типы векторных данных

### Классы

- struct [v16nm4s](#)
- struct [v4nm8s](#)
- struct [s\\_v8nm8s](#)
- struct [s\\_v16nm8s](#)
- struct [s\\_v4nm16s](#)
- struct [s\\_v8nm16s](#)
- struct [s\\_v16nm16s](#)
- struct [s\\_v2nm32s](#)
- struct [s\\_v4nm32s](#)
- struct [s\\_v8nm32s](#)
- struct [s\\_v16nm32s](#)
- struct [s\\_v16nm4u](#)
- struct [s\\_v4nm8u](#)
- struct [s\\_v8nm8u](#)
- struct [s\\_v16nm8u](#)
- struct [s\\_v4nm16u](#)
- struct [s\\_v8nm16u](#)
- struct [s\\_v16nm16u](#)
- struct [s\\_v2nm32u](#)
- struct [s\\_v4nm32u](#)
- struct [s\\_v8nm32u](#)
- struct [s\\_v16nm32u](#)

### Определения типов

- typedef void [nm1](#)
- typedef void [nm2s](#)
- typedef void [nm4s](#)
- typedef void [nm8s](#)
- typedef signed char [nm8s7b](#)
- typedef void [nm16s](#)
- typedef [nm16s](#) [nm16s15b](#)
- typedef int [nm32s](#)
- typedef int [nm32s31b](#)
- typedef int [nm32s30b](#)
- typedef long long [nm64s](#)
- typedef [nm64s](#) [nm64s63b](#)
- typedef void [nm2u](#)
- typedef void [nm4u](#)
- typedef [nm4u](#) [nm4u3b](#)
- typedef void [nm8u](#)
- typedef unsigned char [nm8u7b](#)
- typedef void [nm16u](#)
- typedef [nm16u](#) [nm16u15b](#)
- typedef unsigned int [nm32u](#)
- typedef unsigned int [nm32u31b](#)
- typedef unsigned long long [nm64u](#)
- typedef struct [s\\_v8nm8s](#) [v8nm8s](#)
- typedef struct [s\\_v16nm8s](#) [v16nm8s](#)

- typedef struct [s\\_v4nm16s](#) v4nm16s
- typedef struct [s\\_v8nm16s](#) v8nm16s
- typedef struct [s\\_v16nm16s](#) v16nm16s
- typedef struct [s\\_v2nm32s](#) v2nm32s
- typedef struct [s\\_v4nm32s](#) v4nm32s
- typedef struct [s\\_v8nm32s](#) v8nm32s
- typedef struct [s\\_v16nm32s](#) v16nm32s
- typedef [v16nm8s](#) v16nm8s7b
- typedef struct [s\\_v16nm4u](#) v16nm4u
- typedef struct [s\\_v4nm8u](#) v4nm8u
- typedef struct [s\\_v8nm8u](#) v8nm8u
- typedef struct [s\\_v16nm8u](#) v16nm8u
- typedef struct [s\\_v4nm16u](#) v4nm16u
- typedef struct [s\\_v8nm16u](#) v8nm16u
- typedef struct [s\\_v16nm16u](#) v16nm16u
- typedef struct [s\\_v2nm32u](#) v2nm32u
- typedef struct [s\\_v4nm32u](#) v4nm32u
- typedef struct [s\\_v8nm32u](#) v8nm32u
- typedef struct [s\\_v16nm32u](#) v16nm32u
- typedef [v16nm4u](#) v16nm4b3u

### 7.115.1 Подробное описание

В данном разделе описываются типы векторных данных с которыми могут работать функции библиотеки, задействующие векторный узел. Также рассматриваются соглашения о передаче параметров.

Поскольку векторный узел работает с данными произвольной разрядности, упакованными в 64-разрядные слова, то это накладывает следующие ограничения на работу с массивами данных и их типами:

1. Указатель на векторные данные всегда является четным адресом. Т.е. выровнен в памяти по границе 64р. слов.
2. Размер массива, передаваемый на вход функций, как правило, исчисляется в отдельных элементах, составляющих этот массив.  
Кратность этого размера по умолчанию определяется кол-вом чисел, упакованных в 64р. слово.  
Например:  
для nm8s кратность-8  
для nm16s кратность-4  
для nm32s кратность-2  
для nm64s кратность-1  
Если в описании указаны другие условия кратности, как например [32,64,96,128...], то это означает, что допустимые размеры могут только из этого диапазона с кратностью 32.
3. Типы nm8s , nm16s, nm32s... хоть и созданы для обозначения разрядности упакованных данных, но с точки зрения C++ таковыми не являются , так как определяются через typedef как производные от стандартных типов char, short и int, которые все три в свою очередь являются 32-разрядными типами в архитектуре NeuroMatrix. Поэтому эти векторные типы можно использовать только с оператором \* (nm8s\*,nm16s\*,... ). Операции же sizeof() к массивам этих типов будут выполняться некорректно.

Расшифровка мнемоники в названии типа:

1. Префикс `nm` - означает что данные являются векторными ,упакованными в 64р слова (`nm8s,nm8u,nm16s....` ).
2. Разрядность данных указывается после префикса `nm` (`nm8s,nm8u` - байтовые массивы, `nm16s,nm16u` - 16р. массивы ).
3. суффикс `s` или `u` оначает знаковый или беззнаковый тип данных.
4. Для работы некоторых функций во избежании переполнения требуется суженный диапазон возможных значений, чем позволяет разрядность. Такие данные имеют суффикс в виде кол-ва значащих бит в слове и символом `b`. (`nm8s7b`)

## 7.115.2 Типы

### 7.115.2.1 `nm1`

```
typedef void nm1
```

Большинство функций библиотеки получают и возвращают массивы упакованных чисел. Обращение к элементам данных массивов должно производиться с помощью функций доступа к элементам `Getval()` and `Setval()`.

Тип характеризует векторные данные как массив одноразрядных чисел. Начальный адрес массива должен быть выровнен по границе 64р слова. Предполагается , что размер массива данного типа как минимум кратен 64.

Диапазон значений:

$[-1, 0]$

См. определение в файле `nmtype.h` строка 95

### 7.115.2.2 `nm16s`

```
typedef void nm16s
```

Тип характеризует векторные данные как массив 16-ти разрядных чисел со знаком. Начальный адрес массива должен быть выровнен по границе 64р слова. Предполагается , что размер массива данного типа как минимум кратен 4.

Диапазон значений:

$[-2^{15}, \dots, +2^{15} - 1]$ .

См. определение в файле `nmtype.h` строка 209

## 7.115.2.3 nm16s15b

```
typedef nm16s nm16s15b
```

Тип характеризует векторные данные как массив 16-ти разрядных чисел со знаком с ограниченным диапазоном принимаемых значений.

Начальный адрес массива должен быть выровнен по границе 64р слова.

Предполагается , что размер массива данного типа как минимум кратен 4.

Диапазон значений:

$$[-2^{14}, \dots, +2^{14} - 1]$$

См. определение в файле nmtypе.h строка 231

## 7.115.2.4 nm16u

```
typedef void nm16u
```

Тип характеризует векторные данные как массив 16-ти разрядных чисел без знака.

Начальный адрес массива должен быть выровнен по границе 64р слова.

Предполагается , что размер массива данного типа как минимум кратен 4.

Диапазон значений:

$$[0, \dots, 2^{16} - 1].$$

См. определение в файле nmtypе.h строка 444

## 7.115.2.5 nm16u15b

```
typedef nm16u nm16u15b
```

Тип характеризует векторные данные как массив 16-ти разрядных чисел без знака.

Начальный адрес массива должен быть выровнен по границе 64р слова.

Предполагается , что размер массива данного типа как минимум кратен 4.

Диапазон значений:

$$[-2^{14}, \dots, +2^{14} - 1].$$

См. определение в файле nmtypе.h строка 460

## 7.115.2.6 nm2s

```
typedef void nm2s
```

Тип характеризует векторные данные как массив 2-х разрядных чисел со знаком.  
Начальный адрес массива должен быть выровнен по границе 64р слова.  
Предполагается , что размер массива данного типа как минимум кратен 32.

Диапазон значений:

$$[-2^1, \dots, +2^1 - 1] = [-2, \dots, +1]$$

См. определение в файле nmtypе.h строка 113

## 7.115.2.7 nm2u

```
typedef void nm2u
```

Тип характеризует векторные данные как массив 2-х разрядных чисел без знака.  
Начальный адрес массива должен быть выровнен по границе 64р слова.  
Предполагается , что размер массива данного типа как минимум кратен 32.

Диапазон значений:

$$[0, \dots, +2^2 - 1] = [0, \dots, 3]$$

См. определение в файле nmtypе.h строка 369

## 7.115.2.8 nm32s

```
typedef int nm32s
```

Тип характеризует векторные данные как массив 32-х разрядных чисел со знаком.  
Начальный адрес массива должен быть выровнен по границе 64р слова.  
Предполагается , что размер массива данного типа как минимум кратен 2.

Диапазон значений:

$$[-2^{31}, \dots, +2^{31} - 1].$$

См. определение в файле nmtypе.h строка 257

## 7.115.2.9 nm32s30b

```
typedef int nm32s30b
```

Тип характеризует векторные данные как массив 32-х разрядных чисел со знаком. Начальный адрес массива должен быть выровнен по границе 64-х слова. Предполагается, что размер массива данного типа как минимум кратен 2.

Диапазон значений:

$$[-2^{29}, \dots, 2^{29} - 1].$$

См. определение в файле nmtypе.h строка 319

## 7.115.2.10 nm32s31b

```
typedef int nm32s31b
```

Тип характеризует векторные данные как массив 32-х разрядных чисел со знаком. Начальный адрес массива должен быть выровнен по границе 64-х слова. Предполагается, что размер массива данного типа как минимум кратен 2.

Диапазон значений:

$$[-2^{30}, \dots, 2^{30} - 1].$$

См. определение в файле nmtypе.h строка 299

## 7.115.2.11 nm32u

```
typedef unsigned int nm32u
```

Тип характеризует векторные данные как массив 32-х разрядных чисел без знака. Начальный адрес массива должен быть выровнен по границе 64-х слова. Предполагается, что размер массива данного типа как минимум кратен 2.

Диапазон значений

$$[0, \dots, 2^{32} - 1].$$

См. определение в файле nmtypе.h строка 474

## 7.115.2.12 nm32u31b

```
typedef unsigned int nm32u31b
```

Тип характеризует векторные данные как массив 32-х разрядных чисел без знака. Начальный адрес массива должен быть выровнен по границе 64-х слова. Предполагается, что размер массива данного типа как минимум кратен 2.

Диапазон значений

$$[0, \dots, 2^{31} - 1].$$

См. определение в файле nmtypе.h строка 488

## 7.115.2.13 nm4s

```
typedef void nm4s
```

Тип характеризует векторные данные как массив 4-х разрядных чисел со знаком. Начальный адрес массива должен быть выровнен по границе 64-х слова. Предполагается, что размер массива данного типа как минимум кратен 16.

Диапазон значений:

$$[-2^3, \dots, +2^3 - 1] = [-8, \dots, +7]$$

См. определение в файле nmtypе.h строка 126

## 7.115.2.14 nm4u

```
typedef void nm4u
```

Тип характеризует векторные данные как массив 4-х разрядных чисел без знака. Начальный адрес массива должен быть выровнен по границе 64-х слова. Предполагается, что размер массива данного типа как минимум кратен 16.

Диапазон значений:

$$[0, \dots, +2^4 - 1] = [0, \dots, 15]$$

См. определение в файле nmtypе.h строка 382



## 7.115.2.15 nm4u3b

```
typedef nm4u nm4u3b
```

Тип характеризует векторные данные как массив 4-х разрядных чисел без знака.  
Начальный адрес массива должен быть выровнен по границе 64р слова.  
Предполагается , что размер массива данного типа как минимум кратен 16.

Диапазон значений:

$$[0, \dots, +2^3 - 1] = [0, \dots, 7]$$

См. определение в файле nmtype.h строка 395

## 7.115.2.16 nm64s

```
typedef long long nm64s
```

Тип характеризует векторные данные как массив 64-х разрядных чисел со знаком.  
Начальный адрес массива должен быть выровнен по границе 64р слова.  
По умолчанию размер массива произвольный .

Диапазон значений:

$$[-2^{63}, \dots, +2^{63} - 1]$$

См. определение в файле nmtype.h строка 340

## 7.115.2.17 nm64s63b

```
typedef nm64s nm64s63b
```

Тип характеризует векторные данные как массив 64-х разрядных чисел со знаком.  
Начальный адрес массива должен быть выровнен по границе 64р слова.  
По умолчанию размер массива произвольный .

Диапазон значений:

$$[-2^{62}, \dots, +2^{62} - 1]$$

См. определение в файле nmtype.h строка 355

## 7.115.2.18 nm64u

```
typedef unsigned long long nm64u
```

Тип характеризует векторные данные как массив 64-х разрядных чисел без знака.  
Начальный адрес массива должен быть выровнен по границе 64-х слова.  
По умолчанию размер массива произвольный .

Диапазон значений

$$[0, \dots, 2^{64} - 1].$$

См. определение в файле nmtypе.h строка 501

## 7.115.2.19 nm8s

```
typedef void nm8s
```

Тип характеризует векторные данные как массив 8-ми разрядных чисел со знаком.  
Начальный адрес массива должен быть выровнен по границе 64-х слова.  
Предполагается , что размер массива данного типа как минимум кратен 8.

Диапазон значений:

$$[-2^7, \dots, +2^7 - 1] = [-128, \dots, +127]$$

См. определение в файле nmtypе.h строка 149

## 7.115.2.20 nm8s7b

```
typedef signed char nm8s7b
```

Тип характеризует векторные данные как массив 8-ми разрядных чисел со знаком.  
Начальный адрес массива должен быть выровнен по границе 64-х слова.  
Предполагается , что размер массива данного типа как минимум кратен 8.

Диапазон значений:

$$[-2^6, \dots, +2^6 - 1] = [-64, \dots, +63]$$

См. определение в файле nmtypе.h строка 178

## 7.115.2.21 nm8u

```
typedef void nm8u
```

Тип характеризует векторные данные как массив 8-ми разрядных чисел без знака. Начальный адрес массива должен быть выровнен по границе 64р слова. Предполагается , что размер массива данного типа как минимум кратен 8.

Диапазон значений:

$$[0, \dots, +2^8 - 1] = [0, \dots, 255]$$

См. определение в файле nmtype.h строка 411

## 7.115.2.22 nm8u7b

```
typedef unsigned char nm8u7b
```

Тип характеризует векторные данные как массив 8-ми разрядных чисел без знака. Начальный адрес массива должен быть выровнен по границе 64р слова. Предполагается , что размер массива данного типа как минимум кратен 8.

Диапазон значений:

$$[0, \dots, +2^7 - 1] = [0, \dots, 127]$$

См. определение в файле nmtype.h строка 428

## 7.115.2.23 v16nm16s

```
typedef struct s_v16nm16s v16nm16s
```

Тип векторной структуры, состоящей из 16-ти 16р. чисел со знаком.

## 7.115.2.24 v16nm16u

```
typedef struct s_v16nm16u v16nm16u
```

Тип векторной структуры, состоящей из 16-ти 16р. чисел без знака.

## 7.115.2.25 v16nm32s

```
typedef struct s_v16nm32s v16nm32s
```

Тип векторной структуры, состоящей из 16-ти 32р. чисел со знаком.

7.115.2.26 v16nm32u

```
typedef struct s_v16nm32u v16nm32u
```

Тип векторной структуры, состоящей из 16-ти 32р. чисел без знака.

7.115.2.27 v16nm4b3u

```
typedef v16nm4u v16nm4b3u
```

Тип векторной структуры, состоящей из 16-ти 32р. чисел со знаком.

Диапазон значений элементов структуры:

$[0, \dots, 7]$

См. определение в файле nmtype.h строка 1222

7.115.2.28 v16nm4u

```
typedef struct s_v16nm4u v16nm4u
```

Тип векторной структуры, состоящей из 16-ти 4-р. чисел без знака.

7.115.2.29 v16nm8s

```
typedef struct s_v16nm8s v16nm8s
```

Тип векторной структуры, состоящей из 16-ти 8р. чисел со знаком.

7.115.2.30 v16nm8s7b

```
typedef v16nm8s v16nm8s7b
```

Тип векторной структуры, состоящей из 16-ти 32р. чисел со знаком.

Диапазон значений элементов структуры:

$[-64, \dots, +63]$

См. определение в файле nmtype.h строка 1084

7.115.2.31 v16nm8u

```
typedef struct s_v16nm8u v16nm8u
```

Тип векторной структуры, состоящей из 16-ти 8р. чисел без знака.

7.115.2.32 `v2nm32s`

```
typedef struct s\_v2nm32s v2nm32s
```

Тип векторной структуры, состоящей из 2-х 32р. чисел со знаком.

7.115.2.33 `v2nm32u`

```
typedef struct s\_v2nm32u v2nm32u
```

Тип векторной структуры, состоящей из 2-х 32р. чисел без знака.

7.115.2.34 `v4nm16s`

```
typedef struct s\_v4nm16s v4nm16s
```

Тип векторной структуры, состоящей из 4-х 16р. чисел со знаком.

7.115.2.35 `v4nm16u`

```
typedef struct s\_v4nm16u v4nm16u
```

Тип векторной структуры, состоящей из 4-х 16р. чисел без знака.

7.115.2.36 `v4nm32s`

```
typedef struct s\_v4nm32s v4nm32s
```

Тип векторной структуры, состоящей из 4-х 32р. чисел со знаком.

7.115.2.37 `v4nm32u`

```
typedef struct s\_v4nm32u v4nm32u
```

Тип векторной структуры, состоящей из 4-х 32р. чисел без знака.

7.115.2.38 `v4nm8u`

```
typedef struct s\_v4nm8u v4nm8u
```

Тип векторной структуры, состоящей из 4-х 8р. чисел без знака.

7.115.2.39 `v8nm16s`

```
typedef struct s\_v8nm16s v8nm16s
```

Тип векторной структуры, состоящей из 8-ми 16р. чисел со знаком.

7.115.2.40    `v8nm16u`

```
typedef struct s\_v8nm16u v8nm16u
```

Тип векторной структуры, состоящей из 8-ми 16р. чисел без знака.

7.115.2.41    `v8nm32s`

```
typedef struct s\_v8nm32s v8nm32s
```

Тип векторной структуры, состоящей из 8-ми 32р. чисел со знаком.

7.115.2.42    `v8nm32u`

```
typedef struct s\_v8nm32u v8nm32u
```

Тип векторной структуры, состоящей из 8-ми 32р. чисел без знака.

7.115.2.43    `v8nm8s`

```
typedef struct s\_v8nm8s v8nm8s
```

Тип векторной структуры, состоящей из 8-ми 8р. чисел со знаком.

7.115.2.44    `v8nm8u`

```
typedef struct s\_v8nm8u v8nm8u
```

Тип векторной структуры, состоящей из 8-ми 8р. чисел без знака.

## 7.116 Типы скалярных данных

### Определения типов

- typedef int [int1b](#)
- typedef int [int2b](#)
- typedef int [int3b](#)
- typedef int [int4b](#)
- typedef int [int7b](#)
- typedef int [int8b](#)
- typedef int [int15b](#)
- typedef int [int16b](#)
- typedef int [int30b](#)
- typedef int [int31b](#)
- typedef int [int32b](#)
- typedef INT64 [int63b](#)
- typedef INT64 [int64b](#)
- typedef unsigned int [uint1b](#)
- typedef unsigned int [uint2b](#)
- typedef unsigned int [uint3b](#)
- typedef unsigned int [uint4b](#)
- typedef unsigned int [uint7b](#)
- typedef unsigned int [uint8b](#)
- typedef unsigned int [uint15b](#)
- typedef unsigned int [uint16b](#)
- typedef unsigned int [uint31b](#)
- typedef unsigned int [uint32b](#)
- typedef UINT64 [uint63b](#)
- typedef [nm64u](#) [uint64b](#)

### 7.116.1 Подробное описание

Назначением данной библиотеки является предоставление базовых операций по обработке одномерных массивов (векторов) для процессоров семейства NeroMatrix.

В состав библиотеки входят логические и арифметические функции, операции сравнения, инициализации, копирования, преобразования разрядностей и т.п. Библиотека предназначена для быстрой разработки эффективных пользовательских программ как на языке высокого уровня(C++), так и на языке ассемблера с помощью прилагаемой библиотеки ядра низкоуровневых функций. Функции библиотеки имеют C++ интерфейс.

Большинство функций библиотеки реализованы на языке ассемблера с использованием векторных инструкций и оптимизированы под архитектуру процессоров семейства NeuroMatrix. Для удобства разработки прикладных программ библиотека содержит аналогичные реализации функций для процессоров серии x86, выполненных на языке C++. Данные реализации позволяют выполнять написанные с использованием данной библиотеки прикладные программы на персональном компьютере.

### 7.116.2 Типы

## 7.116.2.1 int15b

```
typedef int int15b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[-2^{14}, \dots, +2^{14} - 1]$$

См. определение в файле nmtypes.h строка 596

## 7.116.2.2 int16b

```
typedef int int16b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[-2^{15}, \dots, +2^{15} - 1]$$

См. определение в файле nmtypes.h строка 609

## 7.116.2.3 int1b

```
typedef int int1b
```

Тип для 32-разрядных скалярных переменных с ограниченным диапазоном значений.

Диапазон значений:

$$[-1, 0]$$

См. определение в файле nmtypes.h строка 518

## 7.116.2.4 int2b

```
typedef int int2b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[-2^1, \dots, +2^1 - 1] = [-2, \dots, +1]$$

См. определение в файле nmtypes.h строка 531



## 7.116.2.5 int30b

```
typedef int int30b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[-2^{29}, \dots, +2^{29} - 1]$$

См. определение в файле nmtypes.h строка 622

## 7.116.2.6 int31b

```
typedef int int31b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[-2^{30}, \dots, +2^{30} - 1]$$

См. определение в файле nmtypes.h строка 635

## 7.116.2.7 int32b

```
typedef int int32b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[-2^{31}, \dots, +2^{31} - 1]$$

См. определение в файле nmtypes.h строка 648

## 7.116.2.8 int3b

```
typedef int int3b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[-2^2, \dots, +2^2 - 1] = [-4, \dots, +3]$$

См. определение в файле nmtypes.h строка 544

## 7.116.2.9 int4b

```
typedef int int4b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[-2^3, \dots, +2^3 - 1] = [-8, \dots, +7]$$

См. определение в файле nmtypes.h строка 557

## 7.116.2.10 int63b

```
typedef INT64 int63b
```

Тип для 64-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[-2^{62}, \dots, +2^{62} - 1]$$

См. определение в файле nmtypes.h строка 661

## 7.116.2.11 int64b

```
typedef INT64 int64b
```

Тип для 64-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[-2^{63}, \dots, +2^{63} - 1]$$

См. определение в файле nmtypes.h строка 674

## 7.116.2.12 int7b

```
typedef int int7b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[-2^6, \dots, +2^6 - 1] = [-64, \dots, +63]$$

См. определение в файле nmtypes.h строка 570

## 7.116.2.13 int8b

```
typedef int int8b
```

Тип для 32-разрядных скалярных переменных с ограниченным диапазоном значений.

Диапазон значений:

$$[-2^7, \dots, +2^7 - 1] = [-128, \dots, +127]$$

См. определение в файле nmtypе.h строка 583

## 7.116.2.14 uint15b

```
typedef unsigned int uint15b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[0, \dots, 2^{15} - 1]$$

См. определение в файле nmtypе.h строка 765

## 7.116.2.15 uint16b

```
typedef unsigned int uint16b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[0, \dots, 2^{16} - 1]$$

См. определение в файле nmtypе.h строка 778

## 7.116.2.16 uint1b

```
typedef unsigned int uint1b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[0, 1] = [0, 1]$$

См. определение в файле nmtypе.h строка 687

## 7.116.2.17 uint2b

```
typedef unsigned int uint2b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[0, \dots, 2^2 - 1] = [0, \dots, 3]$$

См. определение в файле nmtypes.h строка 700

## 7.116.2.18 uint31b

```
typedef unsigned int uint31b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[0, \dots, 2^{31} - 1]$$

См. определение в файле nmtypes.h строка 791

## 7.116.2.19 uint32b

```
typedef unsigned int uint32b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[0, \dots, 2^{32} - 1]$$

См. определение в файле nmtypes.h строка 804

## 7.116.2.20 uint3b

```
typedef unsigned int uint3b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[0, \dots, 2^3 - 1] = [0, \dots, 7]$$

См. определение в файле nmtypes.h строка 713

## 7.116.2.21 uint4b

```
typedef unsigned int uint4b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[0, \dots, 2^4 - 1] = [0, \dots, 15]$$

См. определение в файле nmtypes.h строка 726

## 7.116.2.22 uint63b

```
typedef UINT64 uint63b
```

Тип для 64-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[0, \dots, 2^{63} - 1]$$

См. определение в файле nmtypes.h строка 817

## 7.116.2.23 uint64b

```
typedef nm64u uint64b
```

Тип для 64-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[0, \dots, 2^{64} - 1]$$

См. определение в файле nmtypes.h строка 830

## 7.116.2.24 uint7b

```
typedef unsigned int uint7b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[0, \dots, 2^7 - 1] = [0, \dots, 127]$$

См. определение в файле nmtypes.h строка 739

## 7.116.2.25 uint8b

```
typedef unsigned int uint8b
```

Тип для 32-разрядных скалярных переменных с ограниченным допустимым диапазоном значений.

Диапазон значений:

$$[0, \dots, 2^8 - 1] = [0, \dots, 255]$$

См. определение в файле nmtypes.h строка 752

## 7.117 Функции поддержки

### Группы

- [nmppsMalloc](#)  
Распределение памяти для векторов библиотеки.
- [nmppsFree](#)  
Освобождение памяти для векторов.
- [nmppsAddr\\_](#)  
Возвращает адрес ячейки памяти, содержащей указанный элемент.  
Реализация для процессора NeuroMatrix возвращает адрес, выровненный в памяти на 32 бита.
- [nmppsSetVal\\_](#)  
Модификация элемента вектора.
- [nmppsGetVal\\_](#)  
Извлекает значение элемента вектора.
- [nmppsGetVal\\_\(return\)](#)  
Извлекает значение элемента вектора.

### 7.117.1 Подробное описание

## 7.118 Инициализация и копирование

### Группы

- [nmppsSet-инициализация](#)  
Функция инициализации элементов массива постоянным значением.
- [nmppsRandUniform](#)  
Инициализация массива случайными числами.
- [nmppsRandUniform\\_](#)  
Генерация случайного числа с равномерным распределением.
- [nmppsRamp\\_](#)  
Инициализация массива элементами арифметической прогрессии.
- [nmppsConvert](#)  
Изменение разрядности элементов вектора.  
Преобразование знаковых данных к меньшей разрядности осуществляется отбрасыванием старших битов. Преобразование знаковых данных к большей разрядности осуществляется с распространением влево старшего (знакового) бита. Преобразование беззнаковых данных к большей разрядности осуществляется добавлением слева старших нулевых битов.
- [nmppsCopy\\_](#)  
Копирование вектора.
- [nmppsCopyua\\_](#)  
Копирование вектора с невыровненной байтовой позиции в выровненную.
- [nmppsSwap\\_](#)  
Перестановка двух векторов.

### 7.118.1 Подробное описание

## 7.119 Арифметические операции

### Группы

- [nmppsAbs](#)  
Вычисление абсолютных значений для элементов вектора.
- [nmppsAbs1](#)  
Функция логического вычисления модулей элементов вектора.
- [nmppsNeg](#)  
Изменение знака элементов вектора на противоположный.
- [nmppsAddC](#)  
Добавление к вектору константы.
- [nmppsAdd](#)  
Сложение двух векторов.
- [nmppsAdd\\_AddC](#)  
Сложение двух векторов с прибавлением константы.
- [nmppsSubC](#)  
Вычитание константы из вектора.
- [nmppsSubCRev](#)  
Вычитание константы из вектора с переменной знака элементов вектора.
- [nmppsSub](#)  
Вычитание двух векторов.
- [nmppsAbsDiff](#)  
Вычисление вектора модулей разности элементов двух векторов.
- [nmppsAbsDiff1](#)  
Функция логического вычисления модулей разностей элементов двух векторов.
- [nmppsMulC](#)  
Умножение вектора на константу.
- [nmppsMul\\_AddC](#)  
Поэлементное умножение векторов с прибавлением константы.
- [nmppsMulC\\_AddC](#)  
Умножение вектора на константу с прибавлением константы.
- [nmppsRShiftC\\_MulC\\_AddC](#)
- [nmppsMulC\\_AddV\\_AddC](#)  
Умножение вектора на константу с прибавлением вектора и константы.
- [nmppsSumN](#)  
Сложение нескольких векторов.
- [nmppsDivC](#)  
Деление вектора на константу.
- [nmppsSum](#)  
Возвращает сумму всех элементов вектора.
- [nmppsDotProd](#)  
Скалярное умножение двух векторов.
- [nmppsWeightedSum](#)  
Поэлементное взвешенное суммирование элементов двух векторов

### 7.119.1 Подробное описание



## 7.120 Логические и бинарные операции

### Группы

- [nmppsNot\\_](#)  
Функция логического "НЕ".
- [nmppsAndC](#)  
Функция логического "И" между вектором и константой.
- [nmppsAnd](#)  
Функция логического "И" между двумя векторами.
- [nmppsAnd4V\\_](#)  
Функция логического "И" между четырьмя векторами.
- [nmppsAndNotV\\_](#)  
Функция логического "И-НЕ" между двумя векторами.
- [nmppsOrC](#)  
Функция логического "ИЛИ" между вектором и константой.
- [nmppsOr](#)  
Функция логического "ИЛИ" между двумя векторами.
- [nmppsOr3V\\_](#)  
Функция логического "ИЛИ" между четырьмя векторами.
- [nmppsOr4V\\_](#)  
Функция логического "ИЛИ" между четырьмя векторами.
- [nmppsXorC](#)  
Функция логического "Исключающего ИЛИ" между вектором и константой.
- [nmppsXor](#)  
Функция логического "Исключающего ИЛИ" между двумя векторами.
- [nmppsMaskV\\_](#)  
Функция логического ИЛИ с предварительным маскированием двух векторов.
- [nmppsRShiftC](#)  
Операция арифметического сдвига вправо.
- [nmppsRShiftC\\_](#)  
Операция логического сдвига.
- [nmppsRShiftC\\_AddC\\_](#)  
Операция логического сдвига.
- [nmppsDisplaceBits](#)  
Непрерывное смещение битов внутри бинарного массива в сторону конца массива  
Функция смещает биты внутри бинарного массива на несколько позиций (nBits) в сторону конца массива.  
Внутри 64р. слова младшие биты сдвигаются на старшие позиции того же слова, а старшие биты перемещаются в младшие позиции следующего 64р. слова. Освободившееся место в первом 64р. слове заполняется старшими битами 64р. слова с адреса pnBits. Сдвинутые биты сохраняются в массиве pDst.  
Пример сдвига на 8 бит :

### 7.120.1 Подробное описание

## 7.121 Операции сравнения

### Группы

- [nmppsMax\\_](#)  
Поиск значения максимального элемента вектора.
- [nmppsMin](#)  
Поиск значения минимального элемента вектора.
- [nmppsMaxIndx\\_](#)  
Поиск значения максимального элемента вектора и его положения (положений) в векторе.
- [nmppsMinIndx\\_](#)  
Поиск значения минимального элемента вектора и его положения (положений) в векторе.
- [nmppsMinIndxVN\\_](#)  
Поиск значения минимального элемента вектора длины N и его положения в векторе.
- [nmppsFirstZeroIndx](#)  
Поиск позиции первого нулевого элемента в векторе .
- [nmppsFirstNonZeroIndx](#)  
Поиск позиции первого ненулевого элемента в векторе .
- [nmppsLastZeroIndx](#)  
Поиск позиции последнего нулевого элемента в векторе .
- [nmppsLastNonZeroIndx](#)  
Поиск позиции последнего ненулевого элемента в векторе .
- [nmppsMinEvery\\_](#)  
Поэлементный минимум из двух векторов.
- [nmppsMaxEvery\\_](#)  
Поэлементный максимум из двух векторов.
- [nmppsMinCmpLtV\\_](#)  
Поэлементный минимум из двух векторов.
- [nmppsCmpLt0](#)  
Сравнивает элементы массива на меньше нуля.
- [nmppsCmpEq0](#)  
Сравнивает элементы массива на признак равенства нулю.
- [nmppsCmpMinMaxV\\_](#)  
Поэлементное сравнение двух векторов.
- [nmppsClipPowC\\_](#)  
Функция насыщения.
- [nmppsClipCC\\_](#)  
Функция насыщения с произвольными порогами.
- [nmppsClipRShiftConvert\\_AddC\\_](#)  
Сокращение разрядности данных с предварительной их обработкой.
- [nmppsClipConvert\\_AddC\\_](#)  
Сокращение разрядности данных с предварительной их обработкой.
- [nmppsCmpEqC](#)  
Сравнивает элементы массива на признак равенства константе.
- [nmppsCmpNe0](#)  
Сравнивает элементы массива на признак неравенства нулю.
- [nmppsCmpNeC](#)  
Сравнивает элементы массива на признак неравенства константе.

### 7.121.1 Подробное описание

## 7.122 Переупорядочивание и сортировка

### Группы

- [VEC\\_QSort](#)  
Сортировка массива по убыванию.
- [nmppsRemap\\_](#)  
Переупорядочивание элементов вектора по таблице.
- [nmppSplitTmp](#)  
Расщепляет массив на два, группируя по четным и нечетным элементам
- [nmppSplit](#)  
Расщепляет массив на два массива, группируя по четным и нечетным элементам
- [nmppMerge](#)  
Собирает массив из двух, чередуя элементы из каждого. Функция обратная nmppsSplit.
- [nmppSplit\\_32fcr](#)  
Расщепляет массив на два, группируя по четным и нечетным элементам
- [nmppsDecimate](#)  
Делает выборку элементов из массива с некоторым шагом

### 7.122.1 Подробное описание

## 7.123 nmppsAbs

Вычисление абсолютных значений для элементов вектора.

Функции

- void nmppsAbs\_4s (const nm4s \*pSrcVec, nm4s \*pDstVec, int nSize)
- void nmppsAbs\_8s (const nm8s \*pSrcVec, nm8s \*pDstVec, int nSize)
- void nmppsAbs\_16s (const nm16s \*pSrcVec, nm16s \*pDstVec, int nSize)
- void nmppsAbs\_32s (const nm32s \*pSrcVec, nm32s \*pDstVec, int nSize)
- void nmppsAbs\_64s (const nm64s \*pSrcVec, nm64s \*pDstVec, int nSize)

### 7.123.1 Подробное описание

Вычисление абсолютных значений для элементов вектора.

$$pDstVec[i] = abs\{pSrcVec[i]\},$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec	Входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

Restrictions:

Значения элементов вектора не должны быть равны минимальному значению для соответствующего типа (т.е. -128, -2<sup>15</sup> и т.д). В противном случае, абсолютное значение для таких элементов вычисляется неверно, давая на выходе то же самое число.

## 7.124 nmppsAbs1

Функция логического вычисления модулей элементов вектора.

Функции

- void nmppsAbs1\_4s (const nm4s \*pSrcVec, nm4s \*pDstVec, int nSize)
- void nmppsAbs1\_8s (const nm8s \*pSrcVec, nm8s \*pDstVec, int nSize)
- void nmppsAbs1\_16s (const nm16s \*pSrcVec, nm16s \*pDstVec, int nSize)
- void nmppsAbs1\_32s (const nm32s \*pSrcVec, nm32s \*pDstVec, int nSize)
- void nmppsAbs1\_64s (const nm64s \*pSrcVec, nm64s \*pDstVec, int nSize)

### 7.124.1 Подробное описание

Функция логического вычисления модулей элементов вектора.

$$pDstVec[i] = \begin{cases} pSrcVec[i], & \text{if } pSrcVec[i] \geq 0 \\ -pSrcVec[i] - 1, & \text{if } pSrcVec[i] < 0 \end{cases}$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec	Входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.125 nmppsNeg

Изменение знака элементов вектора на противоположный.

### Функции

- void nmppsNeg\_8s (const nm8s \*pSrcVec, nm8s \*pDstVec, int nSize)
- void nmppsNeg\_16s (const nm16s \*pSrcVec, nm16s \*pDstVec, int nSize)
- void nmppsNeg\_32s (const nm32s \*pSrcVec, nm32s \*pDstVec, int nSize)
- void nmppsNeg\_64s (const nm64s \*pSrcVec, nm64s \*pDstVec, int nSize)

### 7.125.1 Подробное описание

Изменение знака элементов вектора на противоположный.

$$pDstVec[i] = -pSrcVec[i]$$

$$i = \overline{0 \dots nSize - 1}$$

### Аргументы

pSrcVec	Входной вектор.
nSize	Размер векторов в элементах.

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

### Возвращает

void

## 7.126 nmppsAddC

Добавление к вектору константы.

### Функции

- void nmppsAddC\_8s (const nm8s \*pSrcVec, int8b nVal, nm8s \*pDstVec, int nSize)
- void nmppsAddC\_16s (const nm16s \*pSrcVec, int16b nVal, nm16s \*pDstVec, int nSize)
- void nmppsAddC\_32s (const nm32s \*pSrcVec, int32b nVal, nm32s \*pDstVec, int nSize)
- void nmppsAddC\_64s (const nm64s \*pSrcVec, int64b pnVal, nm64s \*pDstVec, int nSize)
- void nmppsAddC\_p64s (const nm64s \*pSrcVec, int64b \*pnVal, nm64s \*pDstVec, int nSize)
- void nmppsAddC\_32fcr (const nm32fcr \*pSrcVec, nm32fcr \*pDstVec, float C, int nSize)

### 7.126.1 Подробное описание

Добавление к вектору константы.

$$pDstVec[i] = pSrcVec[i] + nVal,$$

$$i = \overline{0 \dots nSize - 1}$$

### Аргументы

pSrcVec	Входной вектор.
nVal	Добавляемая константа.
pnVal	Указатель на добавляемую константу.
nSize	Размер векторов в элементах.

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

### Возвращает

void

## 7.127 nmppsAdd

Сложение двух векторов.

Функции

- void nmppsAdd\_4s (const nm4s \*pSrcVec1, const nm4s \*pSrcVec2, nm4s \*pDstVec, int nSize)
- void nmppsAdd\_8s (const nm8s \*pSrcVec1, const nm8s \*pSrcVec2, nm8s \*pDstVec, int nSize)
- void nmppsAdd\_16s (const nm16s \*pSrcVec1, const nm16s \*pSrcVec2, nm16s \*pDstVec, int nSize)
- void nmppsAdd\_32s (const nm32s \*pSrcVec1, const nm32s \*pSrcVec2, nm32s \*pDstVec, int nSize)
- void nmppsAdd\_64s (const nm64s \*pSrcVec1, const nm64s \*pSrcVec2, nm64s \*pDstVec, int nSize)
- void nmppsAdd\_32f (const nm32f \*pSrcVec1, const nm32f \*pSrcVec2, nm32f \*pDstVec, int nSize)
- void nmppsAddEx\_64s (const nm64s \*pSrcVec1, int srcStep1, const nm64s \*pSrcVec2, int srcStep2, nm64s \*pDstVec, int dstStep, int nSize)

### 7.127.1 Подробное описание

Сложение двух векторов.

$$pDstVec[i] = pSrcVec1[i] + pSrcVec2[i],$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
nSize	Размер вектора в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void



## 7.128 nmppsAdd\_AddC

Сложение двух векторов с прибавлением константы.

Функции

- void nmppsAdd\_AddC\_32s (nm32s \*pSrcVec1, nm32s \*pSrcVec2, int nVal, nm32s \*pDstVec, int nSize)

### 7.128.1 Подробное описание

Сложение двух векторов с прибавлением константы.

$$pDstVec[i] = pSrcVec1[i] + pSrcVec2[i] + nVal,$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
nVal	Добавляемая константа.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.129 nmppsSubC

Вычитание константы из вектора.

### Функции

- void nmppsSubC\_4s (const nm4s \*pSrcVec, int4b nVal, nm4s \*pDstVec, int nSize)
- void nmppsSubC\_8s (const nm8s \*pSrcVec, int8b nVal, nm8s \*pDstVec, int nSize)
- void nmppsSubC\_16s (const nm16s \*pSrcVec, int16b nVal, nm16s \*pDstVec, int nSize)
- void nmppsSubC\_32s (const nm32s \*pSrcVec, int32b nVal, nm32s \*pDstVec, int nSize)
- void nmppsSubC\_64s (const nm64s \*pSrcVec, int64b nVal, nm64s \*pDstVec, int nSize)

### 7.129.1 Подробное описание

Вычитание константы из вектора.

$$pDstVec[i] = pSrcVec[i] - nVal$$

$$i = \overline{0 \dots nSize - 1}$$

### Аргументы

pSrcVec	Входной вектор.
nVal	Вычитаемая константа.
pnVal	Указатель на вычитаемую константу.
nSize	Размер векторов в элементах.

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

### Возвращает

void

## 7.130 nmppsSubCRev

Вычитание константы из вектора с переменной знака элементов вектора.

### Функции

- void nmppsSubCRev\_8s (const nm8s \*pSrcVec, int8b nVal, nm8s \*pDstVec, int nSize)
- void nmppsSubCRev\_16s (const nm16s \*pSrcVec, int16b nVal, nm16s \*pDstVec, int nSize)
- void nmppsSubCRev\_32s (const nm32s \*pSrcVec, int32b nVal, nm32s \*pDstVec, int nSize)
- void nmppsSubCRev\_64s (const nm64s \*pSrcVec, int64b nVal, nm64s \*pDstVec, int nSize)

### 7.130.1 Подробное описание

Вычитание константы из вектора с переменной знака элементов вектора.

$$pDstVec[i] = nVal - pSrcVec[i],$$

$$i = \overline{0 \dots nSize - 1}$$

### Аргументы

pSrcVec	Входной вектор.
nVal	Константа.
pnVal	Указатель на константу.
nSize	Размер векторов в элементах.

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

### Возвращает

void

## 7.131 nmppsSub

Вычитание двух вектров.

Функции

- void nmppsSub\_4s (const nm4s \*pSrcVec1, nm4s \*pSrcVec2, nm4s \*pDstVec, int nSize)
- void nmppsSub\_8s (const nm8s \*pSrcVec1, nm8s \*pSrcVec2, nm8s \*pDstVec, int nSize)
- void nmppsSub\_16s (const nm16s \*pSrcVec1, nm16s \*pSrcVec2, nm16s \*pDstVec, int nSize)
- void nmppsSub\_32s (const nm32s \*pSrcVec1, nm32s \*pSrcVec2, nm32s \*pDstVec, int nSize)
- void nmppsSub\_64s (const nm64s \*pSrcVec1, nm64s \*pSrcVec2, nm64s \*pDstVec, int nSize)

### 7.131.1 Подробное описание

Вычитание двух вектров.

$$pDstVec[i] = pSrcVec1[i] - pSrcVec2[i]$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Уменьшаемый вектор.
pSrcVec2	Вычитаемый вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.132 nmppsAbsDiff

Вычисление вектора модулей разности элементов двух векторов.

Функции

- void nmppsAbsDiff\_8s (const nm8s \*pSrcVec1, nm8s \*pSrcVec2, nm8s \*pDstVec, int nSize)
- void nmppsAbsDiff\_16s (const nm16s \*pSrcVec1, nm16s \*pSrcVec2, nm16s \*pDstVec, int nSize)
- void nmppsAbsDiff\_32s (const nm32s \*pSrcVec1, nm32s \*pSrcVec2, nm32s \*pDstVec, int nSize)
- void nmppsAbsDiff\_64s (const nm64s \*pSrcVec1, nm64s \*pSrcVec2, nm64s \*pDstVec, int nSize)

### 7.132.1 Подробное описание

Вычисление вектора модулей разности элементов двух векторов.

$$pDstVec[i] = abs\{pSrcVec1[i] - pSrcVec2[i]\},$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Входной вектор.
pSrcVec2	Вычитаемый вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

Restrictions:

Разность элементов векторов не должна быть равна минимальному значению для соответствующего типа (т.е. -128, -2<sup>15</sup> и т.д.). В противном случае, абсолютное значение для таких элементов вычисляется не верно, давая на выходе то же самое число.

## 7.133 nmppsAbsDiff1

Функция логического вычисления модулей разностей элементов двух векторов.

Функции

- void nmppsAbsDiff1\_8s (nm8s \*pSrcVec1, nm8s \*pSrcVec2, nm8s \*pDstVec, int nSize)

### 7.133.1 Подробное описание

Функция логического вычисления модулей разностей элементов двух векторов.

$$pDstVec[i] = \begin{cases} pSrcVec1[i] - pSrcVec2[i], & \text{if } pSrcVec1[i] - pSrcVec2[i] \geq 0 \\ pSrcVec1[i] - pSrcVec2[i] - 1, & \text{if } pSrcVec1[i] - pSrcVec2[i] < 0 \end{cases}$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Входной вектор.
pSrcVec2	Вычитаемый вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

Restrictions:

Разность элементов векторов не должна быть равна минимальному значению для соответствующего типа (т.е. -128, -2<sup>15</sup> и т.д.). В противном случае, абсолютное значение для таких элементов вычисляется не верно, давая на выходе то же самое число.

## 7.134 nmppsMulC

Умножение вектора на константу.

### Функции

- void nmppsMulC\_8s (const nm8s \*pSrcVec, int8b nVal, nm8s \*pDstVec, int nSize)
- void nmppsMulC\_8s16s (const nm8s \*pSrcVec, int16b nVal, nm16s \*pDstVec, int nSize)
- void nmppsMulC\_16s (const nm16s \*pSrcVec, int16b nVal, nm16s \*pDstVec, int nSize)
- void nmppsMulC\_16s32s (const nm16s \*pSrcVec, int32b nVal, nm32s \*pDstVec, int nSize)
- void nmppsMulC\_32s (const nm32s \*pSrcVec, int32b nVal, nm32s \*pDstVec, int nSize)
- void nmppsMulC\_32s64s (const nm32s \*pSrcVec, int64b nVal, nm64s \*pDstVec, int nSize)
- void nmppsMulC\_64s (const nm64s \*pSrcVec, int64b nVal, nm64s \*pDstVec, int nSize)
- void nmppsMulC\_2s16s (const nm2s \*pSrcVec, int16b nVal, nm16s \*pDstVec, int nSize)

### 7.134.1 Подробное описание

Умножение вектора на константу.

$$pDstVec[i] = nVal \cdot pSrcVec[i],$$

$$i = \overline{0 \dots nSize - 1}$$

### Аргументы

pSrcVec	Входной вектор.
nVal	Константа-множитель.
nSize	Размер векторов в элементах.

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

### Возвращает

void

## 7.135 nmppsMul\_AddC

Поэлементное умножение векторов с прибавлением константы.

Функции

- void nmppsMul\_AddC\_64s (const nm64s \*pSrcVec1, const nm64s \*pSrcVec2, const nm64s \*pnVal, nm64s \*pDstVec, int nSize)

### 7.135.1 Подробное описание

Поэлементное умножение векторов с прибавлением константы.

$$pDstVec[i] = pSrcVec1[i] \cdot pSrcVec2[i] + nVal,$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Входной вектор.
pSrcVec2	Входной вектор.
nVal	указаель на константу-инкремент.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void



## 7.136 nmppsMulC\_AddC

Умножение вектора на константу с прибавлением константы.

## Функции

- void nmppsMulC\_AddC\_32s (const nm32s \*pSrcVec, int nMulVal, int nAddVal, nm32s \*pDstVec, int nSize)
- void nmppsMulC\_AddC\_2x32s (int32x2 \*dataSparseSrc, int32x2 \*mulArg, int32x2 \*addArg, int32x2 \*dataSparseDst, int size, int stepSparseSrc, int stepSparseDst)

Sparse vector by constant multiplication with addition of constant.

$$dataSparseDst[i \cdot stepSparseDst][k] = dataSparseSrc[i \cdot stepSparseSrc][k] \cdot mulArg[k] + addArg[k],$$

$$i = \overline{0 \dots size - 1}; k = \overline{0 \dots K - 1},$$

where K is value of intWxK type.

## 7.136.1 Подробное описание

Умножение вектора на константу с прибавлением константы.

$$pDstVec[i] = nMulVal \cdot pSrcVec[i] + nAddVal,$$

$$i = \overline{0 \dots nSize - 1}$$

## Аргументы

pSrcVec	Входной вектор.
nMulVal	Константа-множитель.
nAddVal	Добавляемая константа.
nSize	Размер векторов в элементах.

## Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

## Возвращает

void

## 7.136.2 Функции

## 7.136.2.1 nmppsMulC\_AddC\_2x32s()

```
void nmppsMulC_AddC_2x32s (
    int32x2 * dataSparseSrc,
    int32x2 * mulArg,
    int32x2 * addArg,
    int32x2 * dataSparseDst,
    int size,
    int stepSparseSrc,
    int stepSparseDst )
```

Sparse vector by constant multiplication with addition of constant.

$$dataSparseDst[i \cdot stepSparseDst][k] = dataSparseSrc[i \cdot stepSparseSrc][k] \cdot mulArg[k] + addArg[k],$$

$$i = \overline{0 \dots size - 1}; k = \overline{0 \dots K - 1},$$

where K is value of int WxK type.

## Аргументы

in	dataSparseSrc	Input sparse vector of 64-bit packed words
in	mulArg	Packed 64-bit word with values to multiply
in	addArg	Packed 64-bit word with values to add
in	dataSparseDst	Ouput sparse vector of 64-bit packed words
in	size	actual amount of 64-bit packed words in sparse vector to be processed
in	stepSparseSrc	64-bit step between input packed words in memory . By default=1 means that input vector is continuous
in	stepSparseDst	64-bit step between output packed words in memory. By default=1 means that output vector is continuous

## Возвращает

void

## 7.137 nmppsRShiftC\_MulC\_AddC

## Функции

- void nmppsRShiftC\_MulC\_AddC\_2x32s (int32x2 \*dataSparseSrc, int32x2 \*preshiftArg, int32x2 \*mulArg, int32x2 \*addArg, int32x2 \*dataSparseDst, int size, int stepSparseSrc, int stepSparseDst)

## 7.137.1 Подробное описание

Sparse vector by constant multiplication with addition of constant.

$$dataSparseDst[i \cdot stepSparseDst][k] = (dataSparseSrc[i \cdot stepSparseSrc][k] \gg preshiftArg[k]) \cdot mulArg[k] + addArg[k],$$

$$i = 0 \dots size - 1; k = 0 \dots K - 1,$$

where K is value of int WxK type

## Аргументы

in	dataSparseSrc	Input sparse vector of 64-bit packed words
in	preshiftArg	Packed 64-bit word of values for preshifting of input data = [2,4,6,8...28,30]
in	mulArg	Packed 64-bit word with values to multiply
in	addArg	Packed 64-bit word with values to add
in	dataSparseDst	Ouput sparse vector of 64-bit packed words
in	size	actual amount of 64-bit packed words in sparse vector to be processed
in	stepSparseSrc	64-bit step between input packed words in memory . By default=1 means that input vector is continuous
in	stepSparseDst	64-bit step between output packed words in memory. By default=1 means that output vector is continuous

## Возвращает

void

## 7.138 nmppsMulC\_AddV\_AddC

Умножение вектора на константу с прибавлением вектора и константы.

Функции

- void nmppsMulC\_AddV\_AddC\_32s (nm32s \*pSrcVec1, int nMulVal, nm32s \*pSrcVec2, int nAddVal, nm32s \*pDstVec, int nSize)

### 7.138.1 Подробное описание

Умножение вектора на константу с прибавлением вектора и константы.

$$pDstVec[i] = nMulVal \cdot pSrcVec1[i] + pSrcVec2[i] + nAddVal,$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Первый входной вектор.
nMulVal	Константа-множитель.
pSrcVec2	Второй входной вектор.
nAddVal	Добавляемая константа.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.139 nmppsSumN

Сложение нескольких векторов.

### Функции

- void nmppsSumN\_8s16s (nm8s \*\*ppSrcVec, nm16s \*pDstVec, int nSize, int nNumberOfVectors)
- void nmppsSumN\_16s (nm16s \*\*ppSrcVec, nm16s \*pDstVec, int nSize, int nNumberOfVectors)

### 7.139.1 Подробное описание

Сложение нескольких векторов.

$$pDstVec[i] = \sum_{j=0}^{(nNumberOfVectors-1)} ppSrcVec(j)(i)$$

### Аргументы

ppSrcVec	Массив указателей на суммируемые вектора.
nNumberOfVectors	Число суммируемых векторов.
nSize	Размер векторов в элементах = [32*PACK]

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

### Возвращает

void

## 7.140 nmppsDivC

Деление вектора на константу.

## Функции

- void nmppsDivC\_32s (nm32s \*pSrcVec, int nDivisor, nm32s \*pDstVec, int nSize, void \*pTmpBuf1, void \*pTmpBuf2)

## 7.140.1 Подробное описание

Деление вектора на константу.

$$pDstVec[i] = \frac{pSrcVec[i]}{Divisor},$$

$$i = \overline{0 \dots nSize - 1}$$

## Аргументы

pSrcVec	Входной вектор.
nDivisor	Константа-делитель.
nSize	Размер входного вектора в элементах.
pTmpBuf1	Временный массив размером nSize 64-х разрядных слов.
pTmpBuf2	Временный массив размером nSize 64-х разрядных слов.

## Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

## Возвращает

void

## Restrictions:

- Допустимые значения для элементов входного вектора лежат в диапазоне [-4095,...,4095];
- Допустимые значения для делителя лежат в диапазоне [1,2,..145].

## Заметки

Для корректного использования in-place параметров следует учитывать порядок получения промежуточных результатов:

the order of obtaining intermediate results:

pSrcVec => pTmpBuf1 (1cpl:L<=>G) - 1/x Multiplying (in-place is supported)

pTmpBuf1 => pTmpBuf2 (1cpl:G<=>L) - Scaling down (in-place is supported)

pTmpBuf2 => pDstVec (2cpl:L<=>G) - Result correction (in-place is supported)

Примеры использования in-place параметров:

```
nmppsDiv_(L0,G0,10240,3,G0,L0);
```

```
nmppsDiv_(L0,L0,10240,3,L0,L0);
```

## 7.141 nmppsSum

Возвращает сумму всех элементов вектора.

### Функции

- void nmppsSum\_1s (const nm1 \*pSrcVec, int nSize, int32b \*pnRes, void \*pTmpBuf)
- void nmppsSum\_8s (const nm8s \*pSrcVec, int nSize, int32b \*pnRes)
- void nmppsSum\_16s (const nm16s \*pSrcVec, int nSize, int64b \*pnRes)
- void nmppsSum\_32s (const nm32s \*pSrcVec, int nSize, int64b \*pnRes)
- void nmppsSum\_64s (const nm64s \*pSrcVec, int nSize, int64b \*pnRes)

### 7.141.1 Подробное описание

Возвращает сумму всех элементов вектора.

$$return = \sum_{i=0}^{(nSize-1)} pSrcVec[i]$$

### Аргументы

pSrcVec	Входной вектор.
pTmpBuf	Временный массив размера nSize 64-х разрядных слов.
nSize	Размер векторов в элементах.

Возвращает

Сумма элементов вектора.



## 7.142 nmppsDotProd

Скалярное умножение двух векторов.

### Функции

- `int nmppsDotProd_8s8sm (const nm8s *pSrcVec1, const nm8s *pSrcVec2, int nSize, int64b *pnRes, nm64s *tmp)`
- `int nmppsDotProd_8s16sm (const nm8s *pSrcVec1, const nm16s *pSrcVec2, int nSize, int64b *pnRes, nm64s *tmp)`
- `int nmppsDotProd_8s32sm (const nm8s *pSrcVec1, const nm32s *pSrcVec2, int nSize, int64b *pnRes, nm64s *tmp)`
- `void nmppsDotProd_8s64s (const nm8s *pSrcVec1, const nm64s *pSrcVec2, int nSize, int64b *pnRes)`
- `int nmppsDotProd_16s16sm (const nm16s *pSrcVec1, const nm16s *pSrcVec2, int nSize, int64b *pnRes, nm64s *tmp)`
- `int nmppsDotProd_16s32sm (const nm16s *pSrcVec1, const nm32s *pSrcVec2, int nSize, int64b *pnRes, nm64s *tmp)`
- `void nmppsDotProd_16s64s (const nm16s *pSrcVec1, const nm64s *pSrcVec2, int nSize, int64b *pnRes)`
- `int nmppsDotProd_32s32sm (const nm32s *pSrcVec1, const nm32s *pSrcVec2, int nSize, int64b *pnRes, nm64s *tmp)`
- `void nmppsDotProd_32s64s (const nm32s *pSrcVec1, const nm64s *pSrcVec2, int nSize, int64b *pnRes)`
- `void nmppsDotProd_64s64s (const nm64s *pSrcVec1, const nm64s *pSrcVec2, int nSize, int64b *pnRes)`

### 7.142.1 Подробное описание

Скалярное умножение двух векторов.

$$nRes = \sum_{i=0}^{nSize-1} pSrcVec1[i] \cdot pSrcVec2[i]$$

#### Аргументы

pSrcVec1	Первый вектор.
pSrcVec2	Второй вектор.
nSize	Размер векторов в элементах.

#### Возвращаемые значения

pnRes	Указатель на результирующее значение.
-------	---------------------------------------

#### Возвращает

pTmpBuff Временный массив из nSize элементов.  
void



## 7.143 nmppsWeightedSum

Поэлементное взвешенное суммирование элементов двух векторов

Функции

- void nmppsWeightedSum\_8s16s (nm8s \*pSrcVec1, int nW1, nm8s \*pSrcVec2, int nW2, nm16s \*pDstVec, int nSize)
- void nmppsWeightedSum\_16s32s (nm16s \*pSrcVec1, int nW1, nm16s \*pSrcVec2, int nW2, nm32s \*pDstVec, int nSize)
- void nmppsWeightedSum\_32s64s (nm32s \*pSrcVec1, nm64s nW1, nm32s \*pSrcVec2, nm64s nW2, nm64s \*pDstVec, int nSize)

### 7.143.1 Подробное описание

Поэлементное взвешенное суммирование элементов двух векторов

$$pDstVec[i] = nW1 \cdot pSrcVec1[i] + nW2 \cdot pSrcVec2[i],$$

$$= \overline{0 \dots nSize - 1}$$

\param pSrcVec1

1-ый входной вектор.

\param nW1

1-ый весовой коэффициент

\param pSrcVec2

2-ой входной вектор.

\param nW2

2-ой весовой коэффициент

\param nSize

Размер векторов в элементах.

\retval pDstVec

Результирующий вектор.

Возвращает

void

## 7.144 nmppsNot\_

Функция логического "НЕ".

Функции

- void nmppsNot\_2u (const nm2u \*pSrcVec, nm2u \*pDstVec, int nSize)
- void nmppsNot\_8u (const nm8u \*pSrcVec, nm8u \*pDstVec, int nSize)
- void nmppsNot\_16u (const nm16u \*pSrcVec, nm16u \*pDstVec, int nSize)
- void nmppsNot\_32u (const nm32u \*pSrcVec, nm32u \*pDstVec, int nSize)
- void nmppsNot\_64u (const nm64u \*pSrcVec, nm64u \*pDstVec, int nSize)

### 7.144.1 Подробное описание

Функция логического "НЕ".

$$pDstVec[i] = \overline{pSrcVec[i]},$$

$$i = \overline{0 \dots nSize - 1}$$

Функция изменяет значения всех битов входного вектора на противоположные.

Аргументы

pSrcVec	Входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.145 nmppsAndC

Функция логического "И" между вектором и константой.

Функции

- void nmppsAndC\_p64u (nm64u \*pSrcVec, nm64u \*pnVal, nm64u \*pDstVec, int nSize)
- void nmppsAndC\_4u (const nm4u \*pSrcVec, uint4b nVal, nm4u \*pDstVec, int nSize)
- void nmppsAndC\_8u (const nm8u \*pSrcVec, uint8b nVal, nm8u \*pDstVec, int nSize)
- void nmppsAndC\_16u (const nm16u \*pSrcVec, uint16b nVal, nm16u \*pDstVec, int nSize)
- void nmppsAndC\_32u (const nm32u \*pSrcVec, uint32b nVal, nm32u \*pDstVec, int nSize)
- void nmppsAndC\_64u (const nm64u \*pSrcVec, uint64b nVal, nm64u \*pDstVec, int nSize)

### 7.145.1 Подробное описание

Функция логического "И" между вектором и константой.

$$pDstVec[i] = pSrcVec[i] \wedge nVal,$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec	Входной вектор.
nVal	Константа.
pnVal	Указатель на константу.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.146 nmppsAnd

Функция логического "И" между двумя векторами.

Функции

- void nmppsAnd\_1 (const nm1 \*pSrcVec1, const nm1 \*pSrcVec2, nm1 \*pDstVec, int nSize)
- void nmppsAnd\_2u (const nm2u \*pSrcVec1, const nm2u \*pSrcVec2, nm2u \*pDstVec, int nSize)
- void nmppsAnd\_4u (const nm4u \*pSrcVec1, const nm4u \*pSrcVec2, nm4u \*pDstVec, int nSize)
- void nmppsAnd\_8u (const nm8u \*pSrcVec1, const nm8u \*pSrcVec2, nm8u \*pDstVec, int nSize)
- void nmppsAnd\_16u (const nm16u \*pSrcVec1, const nm16u \*pSrcVec2, nm16u \*pDstVec, int nSize)
- void nmppsAnd\_32u (const nm32u \*pSrcVec1, const nm32u \*pSrcVec2, nm32u \*pDstVec, int nSize)
- void nmppsAnd\_64u (const nm64u \*pSrcVec1, const nm64u \*pSrcVec2, nm64u \*pDstVec, int nSize)

### 7.146.1 Подробное описание

Функция логического "И" между двумя векторами.

$$pDstVec[i] = pSrcVec1[i] \wedge pSrcVec2[i]$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.147 nmppsAnd4V\_

Функция логического "И" между четырьмя векторами.

Функции

- void nmppsAnd4V\_64u (nm64u \*pSrcVec1, nm64u \*pSrcVec2, nm64u \*pSrcVec3, nm64u \*pSrcVec4, nm64u \*pDstVec, int nSize)

## 7.147.1 Подробное описание

Функция логического "И" между четырьмя векторами.

$$pDstVec[i] = pSrcVec1[i] \wedge pSrcVec2[i] \wedge pSrcVec3[i] \wedge pSrcVec4[i]$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
pSrcVec3	Третий входной вектор.
pSrcVec4	Четвертый входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.148 nmppsAndNotV\_

Функция логического "И-НЕ" между двумя векторами.

Функции

- void nmppsAndNotV\_64u (nm64u \*pSrcVec1, nm64u \*pSrcVec2, nm64u \*pDstVec, int nSize)

## 7.148.1 Подробное описание

Функция логического "И-НЕ" между двумя векторами.

$$pDstVec[i] = pSrcVec1[i] \wedge not pSrcVec2[i]$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void



## 7.149 nmppsOrC

Функция логического "ИЛИ" между вектором и константой.

Функции

- void nmppsOrC\_8u (const nm8u \*pSrcVec, uint8b nVal, nm8u \*pDstVec, int nSize)
- void nmppsOrC\_16u (const nm16u \*pSrcVec, uint16b nVal, nm16u \*pDstVec, int nSize)
- void nmppsOrC\_32u (const nm32u \*pSrcVec, uint32b nVal, nm32u \*pDstVec, int nSize)
- void nmppsOrC\_64u (const nm64u \*pSrcVec, uint64b nVal, nm64u \*pDstVec, int nSize)

## 7.149.1 Подробное описание

Функция логического "ИЛИ" между вектором и константой.

$$pDstVec[i] = pSrcVec[i] \vee nVal$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec	Входной вектор.
nVal	Константа.
pnVal	Указатель на константу.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.150 nmppsOr

Функция логического "ИЛИ" между двумя векторами.

Функции

- void nmppsOr\_1 (const nm1 \*pSrcVec1, const nm1 \*pSrcVec2, nm1 \*pDstVec, int nSize)
- void nmppsOr\_2u (const nm2u \*pSrcVec1, const nm2u \*pSrcVec2, nm2u \*pDstVec, int nSize)
- void nmppsOr\_4u (const nm4u \*pSrcVec1, const nm4u \*pSrcVec2, nm4u \*pDstVec, int nSize)
- void nmppsOr\_8u (const nm8u \*pSrcVec1, const nm8u \*pSrcVec2, nm8u \*pDstVec, int nSize)
- void nmppsOr\_16u (const nm16u \*pSrcVec1, const nm16u \*pSrcVec2, nm16u \*pDstVec, int nSize)
- void nmppsOr\_32u (const nm32u \*pSrcVec1, const nm32u \*pSrcVec2, nm32u \*pDstVec, int nSize)
- void nmppsOr\_64u (const nm64u \*pSrcVec1, const nm64u \*pSrcVec2, nm64u \*pDstVec, int nSize)

### 7.150.1 Подробное описание

Функция логического "ИЛИ" между двумя векторами.

$$pDstVec[i] = pSrcVec1[i] \vee pSrcVec2[i]$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.151 nmppsOr3V\_

Функция логического "ИЛИ" между четырьмя векторами.

Функции

- void nmppsOr3V\_64u (nm64u \*pSrcVec1, nm64u \*pSrcVec2, nm64u \*pSrcVec3, nm64u \*pDstVec, int nSize)

### 7.151.1 Подробное описание

Функция логического "ИЛИ" между четырьмя векторами.

$$pDstVec[i] = pSrcVec1[i] \vee pSrcVec2[i] \vee pSrcVec3[i]$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
pSrcVec3	Третий входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.152 nmppsOr4V\_

Функция логического "ИЛИ" между четырьмя векторами.

Функции

- void nmppsOr4V\_64u (nm64u \*pSrcVec1, nm64u \*pSrcVec2, nm64u \*pSrcVec3, nm64u \*pSrcVec4, nm64u \*pDstVec, int nSize)

### 7.152.1 Подробное описание

Функция логического "ИЛИ" между четырьмя векторами.

$$pDstVec[i] = pSrcVec1[i] \vee pSrcVec2[i] \vee pSrcVec3[i] \vee pSrcVec4[i]$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
pSrcVec3	Третий входной вектор.
pSrcVec4	Четвертый входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.153 nmppsXorC

Функция логического "Исключающего ИЛИ" между вектором и константой.

Функции

- void nmppsXorC\_8u (const nm8u \*pSrcVec, uint8b nVal, nm8u \*pDstVec, int nSize)
- void nmppsXorC\_16u (const nm16u \*pSrcVec, uint16b nVal, nm16u \*pDstVec, int nSize)
- void nmppsXorC\_32u (const nm32u \*pSrcVec, uint32b nVal, nm32u \*pDstVec, int nSize)
- void nmppsXorC\_64u (const nm64u \*pSrcVec, uint64b nVal, nm64u \*pDstVec, int nSize)

### 7.153.1 Подробное описание

Функция логического "Исключающего ИЛИ" между вектором и константой.

$$pDstVec[i] = pSrcVec[i] \vee nVal$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec	Входной вектор.
nVal	Константа.
pnVal	Указатель на константу.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.154 nmppsXor

Функция логического "Исключающего ИЛИ" между двумя векторами.

Функции

- void nmppsXor\_8u (const nm8u \*pSrcVec1, const nm8u \*pSrcVec2, nm8u \*pDstVec, int nSize)
- void nmppsXor\_16u (const nm16u \*pSrcVec1, const nm16u \*pSrcVec2, nm16u \*pDstVec, int nSize)
- void nmppsXor\_32u (const nm32u \*pSrcVec1, const nm32u \*pSrcVec2, nm32u \*pDstVec, int nSize)
- void nmppsXor\_64u (const nm64u \*pSrcVec1, const nm64u \*pSrcVec2, nm64u \*pDstVec, int nSize)

### 7.154.1 Подробное описание

Функция логического "Исключающего ИЛИ" между двумя векторами.

$$pDstVec[i] = pSrcVec1[i] \vee pSrcVec2[i]$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй Входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.155 nmppsMaskV\_

Функция логического ИЛИ с предварительным маскированием двух векторов.

Функции

- void nmppsMaskV\_64u (nm64u \*pSrcVec1, nm64u \*pSrcVec2, nm64u \*pMaskVec, nm64u \*pDstVec, int nSize)

## 7.155.1 Подробное описание

Функция логического ИЛИ с предварительным маскированием двух векторов.

$$pDstVec[i] = (pSrcVec1[i] \quad and \quad pMaskVec[i]) \quad or \quad (pSrcVec2[i] \quad and \quad \overline{pMaskVec[i]})$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй Входной вектор.
pMaskVec	Вектор маски.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.156 nmppsRShiftC

Операция арифметического сдвига вправо.

### Функции

- void nmppsRShiftC\_8s (const nm8s \*pSrcVec, int nShift, nm8s \*pDstVec, int nSize)
- void nmppsRShiftC\_16s (const nm16s \*pSrcVec, int nShift, nm16s \*pDstVec, int nSize)
- void nmppsRShiftC\_32s (const nm32s \*pSrcVec, int nShift, nm32s \*pDstVec, int nSize)
- void nmppsRShiftC\_64s (const nm64s \*pSrcVec, int nShift, nm64s \*pDstVec, int nSize)

### 7.156.1 Подробное описание

Операция арифметического сдвига вправо.

$$pDstVec[i] = pSrcVec[i] >> nShift,$$

$$i = \overline{0 \dots nSize - 1}$$

Функции реализуют операции арифметического сдвига вправо битов элементов вектора. Освободившиеся биты заполняются знаковым битом - старшим битом.

### Аргументы

pSrcVec	Входной вектор.
nSize	Размер вектора в элементах.
nShift	Параметр определяет на сколько позиций нужно сдвинуть биты элементов вектора.

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

### Возвращает

void



## 7.157 nmppsRShiftC\_

Операция логического сдвига.

### Функции

- void nmppsRShiftC\_8u (const nm8u \*pSrcVec, int nShift, nm8u \*pDstVec, int nSize)
- void nmppsRShiftC\_16u (const nm16u \*pSrcVec, int nShift, nm16u \*pDstVec, int nSize)
- void nmppsRShiftC\_32u (const nm32u \*pSrcVec, int nShift, nm32u \*pDstVec, int nSize)
- void nmppsRShiftC\_64u (const nm64u \*pSrcVec, int nShift, nm64u \*pDstVec, int nSize)

### 7.157.1 Подробное описание

Операция логического сдвига.

$$pDstVec[i] = pSrcVec[i] >> nShift,$$

$$i = \overline{0 \dots nSize - 1}$$

Функции реализуют операции логического сдвига вправо битов элементов вектора. Сдвиг осуществляется на число бит, указанных в соответствующем операнде. Освободившиеся биты заполняются нулями.

### Аргументы

pSrcVec	Входной вектор.
nSize	Размер векторов в элементах.
nShift	Определяет на сколько позиций нужно сдвинуть биты элемента вектора.

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

### Возвращает

void

## 7.158 nmppsRShiftC\_AddC\_

Операция логического сдвига.

Операция логического сдвига.

$$pDstVec[i] = (pSrcVec[i] >> nShift) + nAddVal,$$

$$i = \overline{0 \dots nSize - 1}$$

Функции реализуют операции логического сдвига вправо битов элементов вектора с прибавлением константы. Сдвиг осуществляется на число бит, указанных в соответствующем операнде. Освободившиеся биты заполняются нулями.

Аргументы

pSrcVec	Входной вектор.
nAddVal	Константа для суммирования.
nSize	Размер векторов в элементах.
nShift	Определяет на сколько позиций нужно сдвинуть биты элемента вектора.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.159 nmppsDisplaceBits

Непрерывное смещение битов внутри бинарного массива в сторону конца массива

Функция смещает биты внутри бинарного массива на несколько позиций (nBits) в сторону конца массива. Внутри 64р. слова младшие биты сдвигаются на старшие позиции того же слова, а старшие биты перемещаются в младшие позиции следующего 64р. слова. Освободившееся место в первом 64р. слове заполняется старшими битами 64р. слова с адреса pnBits. Сдвинутые биты сохраняются в массиве pDst. Пример сдвига на 8 бит :

### Функции

- void nmppsFwdShiftBitstream (const nm64u \*pSrcVec, nm64u \*pDstVec, nm64u \*pnBits, int n←Bits, int nSize)

### 7.159.1 Подробное описание

Непрерывное смещение битов внутри бинарного массива в сторону конца массива

Функция смещает биты внутри бинарного массива на несколько позиций (nBits) в сторону конца массива. Внутри 64р. слова младшие биты сдвигаются на старшие позиции того же слова, а старшие биты перемещаются в младшие позиции следующего 64р. слова. Освободившееся место в первом 64р. слове заполняется старшими битами 64р. слова с адреса pnBits. Сдвинутые биты сохраняются в массиве pDst. Пример сдвига на 8 бит :

```
pnBits=[AB00000000000000]
pSrcVec=[0807060504030201][FF0F0E0D0C0B0A09]
pDstVec=[07060504030201AB][0F0E0D0C0B0A0908]
```

Последние 8 бит массива pDstVec будут потеряны. Если же указатель pBits установить на последнее 64р. слово в результате получится цикличическое перемещение бит.

### Аргументы

pSrcVec	Входной вектор.
nSize	Размер векторов в 64р. элементах.
pnBits	Указатель на 64р-слово, старшие биты которого записываются на освобождающуюся при сдвиге младшую часть первого 64р. слова
nBits	Кол-во позиций на которое происходит смещение бит :nBits=[2,4,6....62].

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.160 nmppsSet-инициализация

Функция инициализации элементов массива постоянным значением.

### Функции

- void nmppsSet\_8s (nm8s \*pVec, int8b nVal, int nSize)
- void nmppsSet\_16s (nm16s \*pVec, int16b nVal, int nSize)
- void nmppsSet\_32s (nm32s \*pVec, int32b nVal, int nSize)
- void nmppsSet\_64sp (nm64s \*pVec, int64b \*nVal, int nSize)
- \_\_INLINE\_\_ void nmppsSet\_64s (nm64s \*pVec, int64b nVal, int nSize)
- \_\_INLINE\_\_ void nmppsSet\_8u (nm8u \*pVec, uint8b nVal, int nSize)
- \_\_INLINE\_\_ void nmppsSet\_16u (nm16u \*pVec, uint16b nVal, int nSize)
- \_\_INLINE\_\_ void nmppsSet\_32u (nm32u \*pVec, uint32b nVal, int nSize)
- \_\_INLINE\_\_ void nmppsSet\_64u (nm64u \*pVec, uint64b nVal, int nSize)
- \_\_INLINE\_\_ void nmppsSet\_64up (nm64u \*pVec, uint64b \*nVal, int nSize)

### 7.160.1 Подробное описание

Функция инициализации элементов массива постоянным значением.

$$pVec[i] = nVal,$$

$$i = \overline{0 \dots nSize - 1}$$

### Аргументы

nSize	Размер вектора в элементах.
nVal	Константа. Диапазон значений nVal должен соответствовать типу результирующего вектора.

### Возвращаемые значения

pVec	Результирующий вектор.
------	------------------------

### Возвращает

void

## 7.161 nmppsRandUniform

Инициализация массива случайными числами.

### Функции

- void [nmppsRandUniform\\_64s](#) ([nm64s](#) \*pDstVec, int nSize)
- `__INLINE__` void nmppsRandUniform\_8s ([nm8s](#) \*pDstVec, int nSize)
- `__INLINE__` void nmppsRandUniform\_16s ([nm16s](#) \*pDstVec, int nSize)
- `__INLINE__` void nmppsRandUniform\_32s ([nm32s](#) \*pDstVec, int nSize)
- `__INLINE__` void nmppsRandUniform\_8u ([nm8u](#) \*pDstVec, int nSize)
- `__INLINE__` void nmppsRandUniform\_16u ([nm16u](#) \*pDstVec, int nSize)
- `__INLINE__` void nmppsRandUniform\_32u ([nm32u](#) \*pDstVec, int nSize)
- `__INLINE__` void nmppsRandUniform\_64u ([nm64u](#) \*pDstVec, int nSize)
- void nmppsRandUniform\_64f (nm64f \*pDstVec, int nSize, double low, double hi)
- void nmppsRand\_32f (nm32f \*pDstVec, int nSize, float low, float hi)
- void nmppsRandUniform\_32f\_integer (nm32f \*pDstVec, int nSize, int hi, int low)

### 7.161.1 Подробное описание

Инициализация массива случайными числами.

#### Аргументы

nSize	Размер вектора.
nRandomize	Произвольное число для инициализации генератора случайных чисел.

#### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

#### Возвращает

void

### 7.161.2 Функции

#### 7.161.2.1 nmppsRandUniform\_64s()

```
void nmppsRandUniform_64s (
    nm64s * pDstVec,
    int nSize )
```

/ru Инициализация массива 32-разрядными случайными числами. /en Random initialization of 32-bit buffer /~

## 7.162 nmppsRandUniform\_

Генерация случайного числа с равномерным распределением.

### Функции

- `int nmppsRandUniform2_32s (int nMin, int nMax, int nDivisible)`
- `int nmppsRandUniform3_32s (int nMin, int nMax)`
- `int nmppsRandUniform ()`

### 7.162.1 Подробное описание

Генерация случайного числа с равномерным распределением.

/ru Инициализация массива 64-разрядными случайными числами. /en Random initialization of 64-bit buffer /~

### Аргументы

nMin	Минимальное возможное значение случайного числа.
nMax	Максимальное возможное значение случайного числа.
nDivisible	Значение, которому будет кратно случайное число.

### Возвращает

`int` Случайное число в диапазоне либо `[nMin, nMax]`. Для функции без параметров данный диапазон `[-231; 231-1]`.

## 7.163 nmppsRamp\_

Инициализация массива элементами арифметической прогрессии.

Функции

- void nmppsRamp\_8s (nm8s \*pVec, int8b nOffset, int8b nSlope, int nSize)
- void nmppsRamp\_16s (nm16s \*pVec, int16b nOffset, int16b nSlope, int nSize)
- void nmppsRamp\_32s (nm32s \*pVec, int32b nOffset, int32b nSlope, int nSize)
- void nmppsRamp\_64s (nm64s \*pVec, int64b nOffset, int64b nSlope, int nSize)

### 7.163.1 Подробное описание

Инициализация массива элементами арифметической прогрессии.

$$pVec[i] = nOffset + nSlope \cdot i$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

nOffset	Первый член арифметической прогрессии.
nSlope	Разность арифметической прогрессии.
nSize	Размер вектора.

Возвращаемые значения

pVec	Результирующий массив.
------	------------------------

Возвращает

void



## 7.164 nmppsConvert

Изменение разрядности элементов вектора.

Преобразование знаковых данных к меньшей разрядности осуществляется отбрасыванием старших битов. Преобразование знаковых данных к большей разрядности осуществляется с распространением влево старшего (знакового) бита. Преобразование беззнаковых данных к большей разрядности осуществляется добавлением слева старших нулевых битов.

## Функции

- void nmppsConvert\_1s2s (const nm1 \*pSrcVec, nm2s \*pDstVec, int nSize)
- void nmppsConvert\_1u2u (const nm1 \*pSrcVec, nm2u \*pDstVec, int nSize)
- void nmppsConvert\_1u4u (const nm1 \*pSrcVec, nm4u \*pDstVec, int nSize)
- void nmppsConvert\_2s1s (const nm2s \*pSrcVec, nm1 \*pDstVec, int nSize)
- void nmppsConvert\_2s4s (const nm2s \*pSrcVec, nm4s \*pDstVec, int nSize)
- void nmppsConvert\_2u4u (const nm2u \*pSrcVec, nm4u \*pDstVec, int nSize)
- void nmppsConvert\_4s1s (const nm4s \*pSrcVec, nm1 \*pDstVec, int nSize)
- void nmppsConvert\_4s2s (const nm4s \*pSrcVec, nm2s \*pDstVec, int nSize)
- void nmppsConvert\_4s8s (const nm4s \*pSrcVec, nm8s \*pDstVec, int nSize)
- void nmppsConvert\_4u8u (const nm4u \*pSrcVec, nm8u \*pDstVec, int nSize)
- void nmppsConvert\_8s4s (const nm8s \*pSrcVec, nm4s \*pDstVec, int nSize)
- void nmppsConvert\_8s16s (const nm8s \*pSrcVec, nm16s \*pDstVec, int nSize)
- void nmppsConvert\_8s32s (const nm8s \*pSrcVec, nm32s \*pDstVec, int nSize)
- void nmppsConvert\_8s64s (const nm8s \*pSrcVec, nm64s \*pDstVec, int nSize)
- void nmppsConvert\_8u16u (const nm8u \*pSrcVec, nm16u \*pDstVec, int nSize)
- void nmppsConvert\_8u32u (const nm8u \*pSrcVec, nm32u \*pDstVec, int nSize)
- void nmppsConvert\_8u64u (const nm8u \*pSrcVec, nm64u \*pDstVec, int nSize)
- void nmppsConvert\_16s4s (const nm16s \*pSrcVec, nm4s \*pDstVec, int nSize)
- void nmppsConvert\_16s8s (const nm16s \*pSrcVec, nm8s \*pDstVec, int nSize)
- void nmppsConvert\_16s32s (const nm16s \*pSrcVec, nm32s \*pDstVec, int nSize)
- void nmppsConvert\_16s64s (const nm16s \*pSrcVec, nm64s \*pDstVec, int nSize)
- void nmppsConvert\_16u32u (const nm16u \*pSrcVec, nm32u \*pDstVec, int nSize)
- void nmppsConvert\_16u64u (const nm16u \*pSrcVec, nm64u \*pDstVec, int nSize)
- void nmppsConvert\_32s8s (const nm32s \*pSrcVec, nm8s \*pDstVec, int nSize)
- void nmppsConvert\_32s16s (const nm32s \*pSrcVec, nm16s \*pDstVec, int nSize)
- void nmppsConvert\_32s64s (const nm32s \*pSrcVec, nm64s \*pDstVec, int nSize)
- void nmppsConvert\_32u64u (const nm32u \*pSrcVec, nm64u \*pDstVec, int nSize)
- void nmppsConvert\_64s32s (const nm64s \*pSrcVec, nm32s \*pDstVec, int nSize)
- void nmppsConvert\_64s16s (const nm64s \*pSrcVec, nm16s \*pDstVec, int nSize)
- void nmppsConvert\_32u32fcr (const nm32u \*pSrcVec, nm32fcr \*pDstVec, int nSize)

Функция конвертации вектора целых беззнаковых чисел в вектор комплексных чисел, где мнимая(равна 0) и действительная части - 32-битные числа с плавающей точкой

- void nmppsConvert\_32s32fcr (const nm32s \*pSrcVec, nm32fcr \*pDstVec, int nSize)

Функция конвертации вектора целых чисел в вектор комплексных чисел, где мнимая(равна 0) и действительная части - 32-битные числа с плавающей точкой

- void nmppsConvert\_32sc32fcr (const nm32sc \*pSrcVec, nm32fcr \*pDstVec, int nSize)

Функция конвертации вектора комплексных чисел с целыми действительной и мнимой частью (32 бита) в вектор комплексных чисел, где мнимая и действительная части - 32-битные числа с плавающей точкой

- void nmppsJoin\_32f (const nm32f \*pSrcVec1, const nm32f \*pSrcVec2, nm32f \*pDstVec, int nSize)

Функция соединяет 2 массива 32-х чисел с плавающей точкой (float) в один. Результирующий массив { pSrcVec1[0], pSrcVec2[0], pSrcVec1[1], pSrcVec2[1], pSrcVec1[2], pSrcVec2[2] .... }.

- void nmppsConvertRisc\_8u32u (const nm8u \*pSrcVec, nm32u \*pDstVec, int nSize)

Функция конвертации вектора беззнаковых байт в вектор беззнаковых целых чисел

- void nmppsConvertRisc\_32u8u (const nm32u \*pSrcVec, nm8u \*pDstVec, int nSize)

Функция конвертации вектора беззнаковых байт в вектор беззнаковых целых чисел

### 7.164.1 Подробное описание

Изменение разрядности элементов вектора.

Преобразование знаковых данных к меньшей разрядности осуществляется отбрасыванием старших битов. Преобразование знаковых данных к большей разрядности осуществляется с распространением влево старшего (знакового) бита. Преобразование беззнаковых данных к большей разрядности осуществляется добавлением слева старших нулевых битов.

Аргументы

pSrcVec	Входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

### 7.164.2 Функции

#### 7.164.2.1 nmppsConvert\_1s2s()

```
void nmppsConvert_1s2s (
    const nm1 * pSrcVec,
    nm2s * pDstVec,
    int nSize )
```

Restrictions: nSize =[32\*64,32\*64\*2,32\*64\*3,...]

#### 7.164.2.2 nmppsConvert\_1u2u()

```
void nmppsConvert_1u2u (
    const nm1 * pSrcVec,
    nm2u * pDstVec,
    int nSize )
```

Restrictions: nSize =[32\*64,32\*64\*2,32\*64\*3,...]

## 7.164.2.3 nmppsConvert\_32s32fcr()

```
void nmppsConvert_32s32fcr (
    const nm32s * pSrcVec,
    nm32fcr * pDstVec,
    int nSize )
```

Функция конвертации вектора целых чисел в вектор комплексных чисел, где мнимая(равна 0) и действительная части - 32-битные числа с плавающей точкой

## Аргументы

pSrcVec	указатель на входной вектор целых чисел
pDstVec	указатель на выходной вектор комплексных чисел с плавающей точкой
nSize	число элементов во входном векторе

После конвертации мнимая часть каждого комплексного числа в выходном векторе будет равна 0

Функция выполняется на сопроцессоре (процессор 1879ВМ6Я) с плавающей точкой с использованием переупаковщика данных

## 7.164.2.4 nmppsConvert\_32sc32fcr()

```
void nmppsConvert_32sc32fcr (
    const nm32sc * pSrcVec,
    nm32fcr * pDstVec,
    int nSize )
```

Функция конвертации вектора комплексных чисел с целыми действительной и мнимой частью (32 бита) в вектор комплексных чисел, где мнимая и действительная части - 32-битные числа с плавающей точкой

## Аргументы

pSrcVec	указатель на входной вектор комплексных чисел с плавающей точкой
pDstVec	указатель на выходной вектор комплексных чисел с плавающей точкой
nSize	число элементов во входном векторе

Функция выполняется на сопроцессоре (процессор 1879ВМ6Я) с плавающей точкой с использованием переупаковщика данных

## 7.164.2.5 nmppsConvert\_32u32fcr()

```
void nmppsConvert_32u32fcr (
    const nm32u * pSrcVec,
    nm32fcr * pDstVec,
    int nSize )
```

Функция конвертации вектора целых беззнаковых чисел в вектор комплексных чисел, где мнимая(равна 0) и действительная части - 32-битные числа с плавающей точкой

## Аргументы

pSrcVec	указатель на входной вектор целых беззнаковых чисел
pDstVec	указатель на выходной вектор комплексных чисел с плавающей точкой
nSize	число элементов во входном векторе

После конвертации мнимая часть каждого комплексного числа в выходном векторе будет равна 0

Функция выполняется на сопроцессоре (процессор 1879ВМ6Я) с плавающей точкой с использованием переупаковщика данных

## 7.164.2.6 nmppsConvertRisc\_32u8u()

```
void nmppsConvertRisc_32u8u (
    const nm32u * pSrcVec,
    nm8u * pDstVec,
    int nSize )
```

Функция конвертации вектора беззнаковых байт в вектор беззнаковых целых чисел

## Аргументы

pSrcVec	указатель на входной вектор беззнаковых целых чисел
pDstVec	указатель на выходной вектор беззнаковых байт
nSize	число элементов во входном векторе (должно быть кратно 4 и не может быть меньше 4)

Функция выполняется на RISC-процессоре

## 7.164.2.7 nmppsConvertRisc\_8u32u()

```
void nmppsConvertRisc_8u32u (
    const nm8u * pSrcVec,
    nm32u * pDstVec,
    int nSize )
```

Функция конвертации вектора беззнаковых байт в вектор беззнаковых целых чисел

## Аргументы

pSrcVec	указатель на входной вектор беззнаковых байт
pDstVec	указатель на выходной вектор беззнаковых целых чисел
nSize	число элементов во входном векторе (должно быть кратно 8 и не может быть меньше 8)

Функция выполняется на RISC-процессоре

## 7.164.2.8 nmppsJoin\_32f()

```
void nmppsJoin_32f (  
    const nm32f * pSrcVec1,  
    const nm32f * pSrcVec2,  
    nm32f * pDstVec,  
    int nSize )
```

Функция соединяет 2 массива 32-х чисел с плавающей точкой (float) в один. Результирующий массив { pSrcVec1[0], pSrcVec2[0], pSrcVec1[1], pSrcVec2[1], pSrcVec1[2], pSrcVec2[2] .... }.

## Аргументы

pSrcVec1	указатель на входной массив чисел float
pSrcVec2	указатель на входной массив чисел float
pDstVec	указатель на выходной массив чисел float
nSize	количество элементов в массиве pSrcVec (в массиве pDstVec элементов будет в 2 раза больше)

## 7.165 nmppsCopy\_

Копирование вектора.

Функции

- void nmppsCopy\_2s (const [nm2s](#) \*pSrcVec, [nm2s](#) \*pDstVec, int nSize)
- void nmppsCopy\_8s (const [nm8s](#) \*pSrcVec, [nm8s](#) \*pDstVec, int nSize)
- void nmppsCopy\_16s (const [nm16s](#) \*pSrcVec, [nm16s](#) \*pDstVec, int nSize)
- void nmppsCopy\_32s (const [nm32s](#) \*pSrcVec, [nm32s](#) \*pDstVec, int nSize)
- void nmppsCopy\_64s (const [nm64s](#) \*pSrcVec, [nm64s](#) \*pDstVec, int nSize)
- \_\_INLINE\_\_ void nmppsCopy\_8u (const [nm8u](#) \*pSrcVec, [nm8u](#) \*pDstVec, int nSize)
- \_\_INLINE\_\_ void nmppsCopy\_16u (const [nm16u](#) \*pSrcVec, [nm16u](#) \*pDstVec, int nSize)
- \_\_INLINE\_\_ void nmppsCopy\_32u (const [nm32u](#) \*pSrcVec, [nm32u](#) \*pDstVec, int nSize)
- \_\_INLINE\_\_ void nmppsCopy\_64u (const [nm64u](#) \*pSrcVec, [nm64u](#) \*pDstVec, int nSize)

### 7.165.1 Подробное описание

Копирование вектора.

$$pDstVec[i] = pSrcVec[i],$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec	Входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.166 nmppsCoryua\_

Копирование вектора с невыровненной байтовой позиции в выровненную.

Функции

- void nmppsCoryua\_8s (const nm8s \*pSrcVec, int nSrcOffset, nm8s \*pDstVec, int nSize)

## 7.166.1 Подробное описание

Копирование вектора с невыровненной байтовой позиции в выровненную.

$$pDstVec[i] = pSrcVec[nSrcOffset + i]$$

$$i = \overline{0 \dots nSize - 1}$$

Позиция байта считается выровненной если она совпадает с границей 64р. слов в памяти.

\param pSrcVec

Входной вектор.

Аргументы

pDstVec	Результирующий вектор.
nSrcOffset	Смещение в элементах относительно начала вектора. nSrcOffset Может принимать любое значение.
nSize	Кол-во копируемых элементов.

Возвращает

void

## 7.167 nmppsSwap\_

Перестановка двух векторов.

Функции

- void nmppsSwap\_64s (nm64s \*pSrcVec1, nm64s \*pSrcVec2, int nSize)

### 7.167.1 Подробное описание

Перестановка двух векторов.

Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
nSize	Размер векторов в элементах.

Возвращает

void



## 7.168 nmppsMax\_

Поиск значения максимального элемента вектора.

## Функции

- void nmppsMax\_8s7b (const nm8s7b \*pSrcVec, int nSize, int8b \*nMaxValue)
- void nmppsMax\_16s15b (const nm16s15b \*pSrcVec, int nSize, int16b \*nMaxValue)
- void nmppsMax\_32s31b (const nm32s31b \*pSrcVec, int nSize, int \*nMaxValue)
- void nmppsMax\_64s63b (const nm64s63b \*pSrcVec, int nSize, int64b \*nMaxValue)
- int nmppsMax\_8sm (const nm8s \*srcVec, int size, int8b \*maxValue, nm16s \*tmp)
- int nmppsMax\_16sm (const nm16s \*srcVec, int size, int16b \*maxValue, nm32s \*tmp)
- int nmppsMax\_32sm (const nm32s \*srcVec, int size, int32b \*maxValue, nm64s \*tmp)

## 7.168.1 Подробное описание

Поиск значения максимального элемента вектора.

$$nMaxValue = \max_i (pSrcVec[i])$$

## Аргументы

pSrcVec	Входной вектор.
nSize	Размер вектора в элементах.

## Возвращаемые значения

nMaxValue	значение максимального элемент вектора.
-----------	---

## Возвращает

void

## Restrictions:

Ограничения на параметры приводятся в описании каждой из функций.

## 7.168.2 Функции

## 7.168.2.1 nmppsMax\_16s15b()

```
void nmppsMax_16s15b (
    const nm16s15b * pSrcVec,
    int nSize,
    int16b * nMaxValue )
```

## Restrictions:

Максимальный и минимальный элементы массива должны отличаться не более чем на  $2^{15}-1$ .  
Примеры допустимых диапазонов входных чисел:

## 7.168.2.2 nmppsMax\_32s31b()

```
void nmppsMax_32s31b (
    const nm32s31b * pSrcVec,
    int nSize,
    int * nMaxValue )
```

## Restrictions:

Максимальный и минимальный элементы массива должны отличаться не более чем на  $2^{31}-1$ .  
Примеры допустимых диапазонов входных чисел:  
[00000000h..7FFFFFFFh]=[ 0.. $2^{31}-1$ ] [FFFFFFFFh..7FFFFFFEh]=[ -1.. $2^{31}-2$ ] [C0000000h..3←  
FFFFFFFFh]=[ - $2^{30}$ .. $2^{30}-1$ ] [80000000h..00000000h]=[ - $2^{31}$ ..0 ]

## 7.168.2.3 nmppsMax\_8s7b()

```
void nmppsMax_8s7b (
    const nm8s7b * pSrcVec,
    int nSize,
    int8b * nMaxValue )
```

## Restrictions:

Физический размер вектора должен быть кратен блоку из 32-х 64р. слов

Максимальный и минимальный элементы массива должны отличаться не более чем на 127.  
Примеры допустимых диапазонов входных чисел:  
[00h..7Fh]=[ 0.. $2^7-1$ ] [FFh..7Eh]=[ -1.. $2^7-1$ ] [C0h..3Fh]=[ - $2^6$ .. $2^6-1$ ] [80h..00h]=[ - $2^7$ ..0 ]

## 7.169 nmppsMin

Поиск значения минимального элемента вектора.

### Функции

- void nmppsMin\_8s7b (const nm8s7b \*pSrcVec, int nSize, int8b \*nMinValue)
- void nmppsMin\_16s15b (const nm16s15b \*pSrcVec, int nSize, int16b \*nMinValue)
- void nmppsMin\_32s31b (const nm32s31b \*pSrcVec, int nSize, int \*nMinValue)
- void nmppsMin\_64s63b (const nm64s63b \*pSrcVec, int nSize, int64b \*nMinValue)
- int nmppsMin\_8sm (const nm8s \*srcVec, int size, int8b \*minValue, nm16s \*tmp)
- int nmppsMin\_16sm (const nm16s \*srcVec, int size, int16b \*minValue, nm32s \*tmp)
- int nmppsMin\_32sm (const nm32s \*srcVec, int size, int32b \*minValue, nm64s \*tmp)

### 7.169.1 Подробное описание

Поиск значения минимального элемента вектора.

$$nMinValue = \min_i (pSrcVec[i])$$

### Аргументы

pSrcVec	Входной вектор.
nSize	Размер вектора в элементах.

### Возвращаемые значения

nMinValue	значение минимального элемент.
-----------	--------------------------------

### Возвращает

void

### Restrictions:

Ограничения на параметры приводятся в описании каждой из функций.

### 7.169.2 Функции

## 7.169.2.1 nmppsMin\_16s15b()

```
void nmppsMin_16s15b (
    const nm16s15b * pSrcVec,
    int nSize,
    int16b * nMinValue )
```

## Restrictions:

Максимальный и минимальный элементы массива должны отличаться не более чем на  $2^{15}-1$ .  
 Примеры допустимых диапазонов входных чисел:  
 $[0000h..7FFFh] = [0..+32767]$   $[FFFFh..7FFEh] = [-1..+32766]$   $[C000h..3FFFh] = [-16384..+16383]$   $[8000h..0000h] = [-32768..0]$

## 7.169.2.2 nmppsMin\_32s31b()

```
void nmppsMin_32s31b (
    const nm32s31b * pSrcVec,
    int nSize,
    int * nMinValue )
```

## Restrictions:

Максимальный и минимальный элементы массива должны отличаться не более чем на  $2^{31}-1$ .  
 Примеры допустимых диапазонов входных чисел:  
 $[00000000h..7FFFFFFFh] = [0..+2^{31}-1]$   $[FFFFFFFFh..7FFFFFFEh] = [-1..+2^{31}-2]$   $[C0000000h..3FFFFFFFh] = [-2^{30}..+2^{30}-1]$   $[80000000h..00000000h] = [-2^{31}..0]$

## 7.169.2.3 nmppsMin\_8s7b()

```
void nmppsMin_8s7b (
    const nm8s7b * pSrcVec,
    int nSize,
    int8b * nMinValue )
```

## Restrictions:

Максимальный и минимальный элементы массива должны отличаться не более чем на 127.  
 Примеры допустимых диапазонов входных чисел:  
 $[00h..7Fh] = [0..+127]$   $[FFh..7Eh] = [-1..+126]$   $[C0h..3Fh] = [-64..+63]$   $[80h..00h] = [-128..0]$

## 7.170 nmppsMaxIndx\_

Поиск значения максимального элемента вектора и его положения (положений) в векторе.

## Функции

- void nmppsMaxIndx\_8s (nm8s7b \*pSrcVec, int nSize, int \*nIndex, int8b \*nMaxValue, void \*pLTmpBuf, void \*pGTmpBuf, int nSearchDir)
- void nmppsMaxIndx\_16s (nm16s15b \*pSrcVec, int nSize, int \*nIndex, int16b \*nMaxValue, void \*pLTmpBuf, void \*pGTmpBuf, int nSearchDir)
- void nmppsMaxIndx\_32s (nm32s31b \*pSrcVec, int nSize, int \*nIndex, int32b \*nMaxValue, void \*pLTmpBuf, void \*pGTmpBuf, int nSearchDir)

## 7.170.1 Подробное описание

Поиск значения максимального элемента вектора и его положения (положений) в векторе.

$$nMaxValue = \max_i (pSrcVec[i])$$

## Аргументы

pSrcVec	Входной вектор.
nSize	Размер вектора в элементах. Занимаемый этим вектором объем памяти должен быть кратен 64 длинным словам (nm64s[64,128,...]).
pLTmpBuf	Временный массив на локальной шине из nSize элементов.
pGTmpBuf	Временный массив на глобальной шине .
nSearchDir	Направление поиска максимума. При nSearchDir = 1, поиск ведется от начала массива. При nSearchDir = -1, поиск ведется от конца массива. Значение максимального элемента.

## Возвращаемые значения

nIndex	Индекс первого найденного максимума среди равных.
--------	---

## Возвращает

void

## Restrictions:

Ограничения на параметры приводятся в описании каждой из функций.

## Restrictions:

Диапазоны входных элементов могут быть "плавающими"  
Например для данных nm16s15b максимальный и минимальный элементы массива должны отличаться не более чем на  $2^{15}-1$ . Примеры допустимых диапазонов входных чисел для типа nm16s15b :

Here are some examples of admissible ranges for input numbers: \n

[0000h..7FFFh]=[ 0..+32767 ] [FFFFh..7FFEh]=[ -1..+32766 ] [C000h..3FFFh]=[-16384..+16383 ]  
[8000h..0000h]=[-32768..0 ]

## 7.171 nmppsMinIndx\_

Поиск значения минимального элемента вектора и его положения (положений) в векторе.

## Функции

- void nmppsMinIndx\_8s (nm8s7b \*pSrcVec, int nSize, int \*nIndex, int8b \*nMinValue, void \*pLTmpBuf, void \*pGTmpBuf, int nSearchDir)
- void nmppsMinIndx\_16s (nm16s15b \*pSrcVec, int nSize, int \*nIndex, int16b \*nMinValue, void \*pLTmpBuf, void \*pGTmpBuf, int nSearchDir)
- void nmppsMinIndx\_32s (nm32s31b \*pSrcVec, int nSize, int \*nIndex, int32b \*nMinValue, void \*pLTmpBuf, void \*pGTmpBuf, int nSearchDir)

## 7.171.1 Подробное описание

Поиск значения минимального элемента вектора и его положения (положений) в векторе.

$$nMinValue = \min_i (pSrcVec[i])$$

## Аргументы

pSrcVec	Входной вектор.
nSize	Размер вектора в элементах. Занимаемый этим вектором объем памяти должен быть кратен 64 длинным словам (nm64s[64,128,...]).
pLTmpBuf	Временный массив на локальной шине .
pGTmpBuf	Временный массив на глобальной шине из nSize элементов.
nSearchDir	Направление поиска минимума. При nSearchDir = 1, поиск ведется от начала массива. При nSearchDir = -1, поиск ведется от конца массива. Значение минимального элемента.

## Возвращаемые значения

nIndex	Индекс первого найденного минимума среди равных.
--------	--

## Возвращает

void

## Restrictions:

Ограничения на параметры приводятся в описании каждой из функций.

## Restrictions:

Диапазоны входных элементов могут быть "плавающими"  
Например для данных nm16s15b максимальный и минимальный элементы массива должны отличаться не более чем на  $2^{15}-1$ . Примеры допустимых диапазонов входных чисел для типа nm16s15b :

Here are some examples of admissible ranges for input numbers: \n

[0000h..7FFFh]=[ 0..+32767 ] [FFFFh..7FFEh]=[ -1..+32766 ] [C000h..3FFFh]=[-16384..+16383 ]  
[8000h..0000h]=[-32768..0 ]



## 7.172 nmppsMinIndxVN\_\_

Поиск значения минимального элемента вектора длины N и его положения в векторе.

### Функции

- int nmppsMinIndxV9\_32s (int \*pSrcVec, int nStride, int \*nPos)
- int nmppsMinIndxV16\_32s (int \*pSrcVec, int nStride, int \*nPos)
- int nmppsMinIndxV25\_32s (int \*pSrcVec, int nStride, int \*nPos)
- int nmppsMinIndxV256\_32s (int \*pSrcVec, int nStride, int \*nPos)
- int nmppsMinIndxV1024\_32s (int \*pSrcVec, int nStride, int \*nPos)

### 7.172.1 Подробное описание

Поиск значения минимального элемента вектора длины N и его положения в векторе.

#### Аргументы

pSrcVec	Массив из N элементов, где N определяется числом в имени функции.
nStride	шаг между элементами в 32р. словах

#### Возвращаемые значения

Индекс	первого найденного минимума.
--------	------------------------------

#### Возвращает

Значение минимального элемента.

#### Restrictions:

Вычисления проиводятся на скалярном ядре, поэтому указатель на данные pSrcVec может ссылаться на нечетный адрес.

## 7.173 nmppsFirstZeroIndx

Поиск позиции первого нулевого элемента в векторе .

Функции

- `int nmppsFirstZeroIndx_32s (int *pSrcVec, int nSize)`

### 7.173.1 Подробное описание

Поиск позиции первого нулевого элемента в векторе .

Аргументы

<code>pSrcVec</code>	Входной массив.
<code>nIndex</code>	Размер массива.

Возвращает

Позиция нулевого элемента или -1 в случае если нулевой элемент не найден

Restrictions:

Вычисления проиводятся на скалярном ядре, поэтому указатель на данные `pSrcVec` может ссылаться на нечетный адрес. Поиск производится до первого нулевого элемента.

## 7.174 nmppsFirstNonZeroIndx

Поиск позиции первого ненулевого элемента в векторе .

Функции

- `int nmppsFirstNonZeroIndx_32s (int *pSrcVec, int nSize)`

### 7.174.1 Подробное описание

Поиск позиции первого ненулевого элемента в векторе .

Аргументы

<code>pSrcVec</code>	Входной массив.
<code>nIndex</code>	Размер массива.

Возвращает

Позиция ненулевого элемента или -1 в случае если ненулевой элемент не найден

Restrictions:

Вычисления проиводятся на скалярном ядре, поэтому указатель на данные `pSrcVec` может ссылаться на нечетный адрес. Поиск производится до первого ненулевого элемента.

## 7.175 nmppsLastZeroIndx

Поиск позиции последнего нулевого элемента в векторе .

Функции

- `int nmppsLastZeroIndx_32s (int *pSrcVec, int nSize)`

### 7.175.1 Подробное описание

Поиск позиции последнего нулевого элемента в векторе .

Аргументы

<code>pSrcVec</code>	Входной массив.
<code>nIndex</code>	Размер массива.

Возвращает

Позиция нулевого элемента или -1 в случае если нулевой элемент не найден

Restrictions:

Вычисления проиводятся на скалярном ядре, поэтому указатель на данные `pSrcVec` может ссылаться на нечетный адрес. Поиск производится с конца до первого нулевого элемента.

## 7.176 nmppsLastNonZeroIndx

Поиск позиции последнего ненулевого элемента в векторе .

Функции

- `int nmppsLastNonZeroIndx_32s (int *pSrcVec, int nSize)`

### 7.176.1 Подробное описание

Поиск позиции последнего ненулевого элемента в векторе .

Аргументы

<code>pSrcVec</code>	Входной массив.
<code>nIndex</code>	Размер массива.

Возвращает

Позиция нулевого элемента или -1 в случае если нулевой элемент не найден

Restrictions:

Вычисления производятся на скалярном ядре, поэтому указатель на данные `pSrcVec` может ссылаться на нечетный адрес. Поиск производится с конца до первого ненулевого элемента.

## 7.177 nmppsMinEvery\_

Поэлементный минимум из двух векторов.

## Функции

- void nmppsMinEvery\_8s (nm8s7b \*pSrcVec1, nm8s7b \*pSrcVec2, nm8s7b \*pDstMinVec, int nSize)
- void nmppsMinEvery\_16s (nm16s15b \*pSrcVec1, nm16s15b \*pSrcVec2, nm16s15b \*pDstMinVec, int nSize)
- void nmppsMinEvery\_32s (nm32s31b \*pSrcVec1, nm32s31b \*pSrcVec2, nm32s31b \*pDstMinVec, int nSize)
- void nmppsMinEvery\_64s (nm64s63b \*pSrcVec1, nm64s63b \*pSrcVec2, nm64s63b \*pDstMinVec, int nSize)

## 7.177.1 Подробное описание

Поэлементный минимум из двух векторов.

$$pDstMinVec[i] = \min(pSrcVec1[i], pSrcVec2[i])$$

$$i = \overline{0 \dots nSize - 1}$$

## Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй Входной вектор.
nSize	Размер векторов в элементах.

## Возвращаемые значения

pDstMinVec	Массив поэлементных минимумов для входных массивов.
------------	---

## Возвращает

void

## Restrictions:

Сравниваемые пары чисел двух массивов должны отличаться не более чем на  $2^{15} - 1$ .

Примеры допустимых диапазонов сравниваемых пар чисел:

should not be more than  $2^{15} - 1$ .

Here are some examples of admissible ranges for comparable pairs of numbers:

[0000h..7FFFh]=[ 0..+32767 ]

[FFFFh..7FFEh]=[ -1..+32766 ]

[C000h..3FFFh]=[ -16384..+16383 ]

[8000h..0000h]=[ -32768..0 ]

## 7.178 nmppsMaxEvery\_

Поэлементный максимум из двух векторов.

## Функции

- void nmppsMaxEvery\_8s (nm8s7b \*pSrcVec1, nm8s7b \*pSrcVec2, nm8s7b \*pDstMaxVec, int nSize)
- void nmppsMaxEvery\_16s (nm16s15b \*pSrcVec1, nm16s15b \*pSrcVec2, nm16s15b \*pDstMaxVec, int nSize)
- void nmppsMaxEvery\_32s (nm32s31b \*pSrcVec1, nm32s31b \*pSrcVec2, nm32s31b \*pDstMaxVec, int nSize)
- void nmppsMaxEvery\_64s (nm64s63b \*pSrcVec1, nm64s63b \*pSrcVec2, nm64s63b \*pDstMaxVec, int nSize)

## 7.178.1 Подробное описание

Поэлементный максимум из двух векторов.

$$pDstMaxVec[i] = \max(pSrcVec1[i], pSrcVec2[i])$$

$$i = \overline{0 \dots nSize - 1}$$

## Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй Входной вектор.
nSize	Размер векторов в элементах.

## Возвращаемые значения

pDstMaxVec	Массив поэлементных минимумов для входных массивов.
------------	---

## Возвращает

void

## Restrictions:

Сравниваемые пары чисел двух массивов должны отличаться не более чем на  $2^{15} - 1$ .

Примеры допустимых диапазонов сравниваемых пар чисел:

should not be more than  $2^{15} - 1$ .

Here are some examples of admissible ranges for comparable pairs of numbers:

[0000h..7FFFh]=[ 0..+32767 ]

[FFFFh..7FFEh]=[ -1..+32766 ]

[C000h..3FFFh]=[ -16384..+16383 ]

[8000h..0000h]=[ -32768..0 ]





## 7.179 nmppsMinCmpLtV\_

Поэлементный минимум из двух векторов.

Функции

- void nmppsMinCmpLtV\_16s (nm16s15b \*pSrcVec1, nm16s15b \*pSrcVec2, nm16s15b \*pDstMin, nm16s15b \*pDstSignMask, int nSize)

## 7.179.1 Подробное описание

Поэлементный минимум из двух векторов.

$$pDstMin[i] = \min(pSrcVec1[i], pSrcVec2[i])$$

$$pDstSignMask[i] = \begin{cases} 11 \dots 1b, & pSrcVec1[i] < pSrcVec2[i] \\ 00 \dots 0b, & pSrcVec1[i] \leq pSrcVec2[i] \end{cases}$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй Входной вектор.
nSize	Размер векторов в элементах.

Возвращаемые значения

pDstMin	Массив поэлементных минимумов для входных массивов.
pDstSignMask	Массив знаков поэлементных разностей первого и второго векторов.

Возвращает

void

Restrictions:

Сравниваемые пары чисел двух массивов должны отличаться не более чем на  $2^{15} - 1$ .

Примеры допустимых диапазонов сравниваемых пар чисел:

should not be more than  $2^{15} - 1$ .

Here are some examples of admissible ranges for comparable pairs of numbers:

[0000h..7FFFh]=[ 0..+32767 ]

[FFFFh..7FFEh]=[ -1..+32766 ]

[C000h..3FFFh]=[ -16384..+16383 ]

[8000h..0000h]=[ -32768..0 ]



## 7.180 nmppsCmpLt0

Сравнивает элементы массива на меньше нуля.

### Функции

- void nmppsCmpLt0\_8s (const nm8s \*pSrcVec, nm8s \*pDstVec, int nSize)
- void nmppsCmpLt0\_16s (const nm16s \*pSrcVec, nm16s \*pDstVec, int nSize)
- void nmppsCmpLt0\_32s (const nm32s \*pSrcVec, nm32s \*pDstVec, int nSize)
- void nmppsCmpLt0\_64s (const nm64s \*pSrcVec, nm64s \*pDstVec, int nSize)

### 7.180.1 Подробное описание

Сравнивает элементы массива на меньше нуля.

$$pDstVec(i) = \begin{cases} 11 \dots 1b, & \text{if } pSrcVec(i) < 0 \\ 00 \dots 0b, & \text{if } pSrcVec(i) \geq 0 \end{cases}$$

$$i = \overline{0 \dots nSize - 1}$$

Если элемент входного вектора меньше 0, во все биты соответствующего элемента выходного вектора записывается 1.

### Аргументы

pSrcVec	Входной вектор.
nSize	Размер векторов в элементах.

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

### Возвращает

void

## 7.181 nmppsCmpEq0

Сравнивает элементы массива на признак равенства нулю.

### Функции

- void `nmppsCmpEq0_8u7b` (`nm8u7b` \*pSrcVec, `nm1` \*pDstVec, int nSize, int nTrueFlag)
- void `nmppsCmpEq0_16u15b` (`nm16u15b` \*pSrcVec, `nm1` \*pDstVec, int nSize, int nTrueFlag)
- void `nmppsCmpEq0_32u31b` (`nm32u31b` \*pSrcVec, `nm1` \*pDstVec, int nSize, int nTrueFlag)

### 7.181.1 Подробное описание

Сравнивает элементы массива на признак равенства нулю.

$$pDstVec(i) = \begin{cases} 1, & \text{if } pSrcVec(i) = 0 \\ 0, & \text{if } pSrcVec(i) \neq 0 \end{cases}, \text{if } nTrueFlag = 1$$

$$pDstVec(i) = \begin{cases} 0, & \text{if } pSrcVec(i) = 0 \\ 1, & \text{if } pSrcVec(i) \neq 0 \end{cases}, \text{if } nTrueFlag = 0$$

$$i = \overline{0 \dots nSize - 1}$$

### Аргументы

pSrcVec	Входной вектор.
nSize	Размер векторов в элементах. Vector size in elements.
nTrueFlag	Младший бит nTrueFlag определяет значение выходного бита при выполнении условия.

### Возвращаемые значения

pDstVec	Результирующий бинарный вектор.
---------	---------------------------------

### Возвращает

void

### 7.181.2 Функции

## 7.181.2.1 nmppsCmpEq0\_16u15b()

```
void nmppsCmpEq0_16u15b (
    nm16u15b * pSrcVec,
    nm1 * pDstVec,
    int nSize,
    int nTrueFlag )
```

Заметки

nSize =[1,2,3,4...]

## 7.181.2.2 nmppsCmpEq0\_32u31b()

```
void nmppsCmpEq0_32u31b (
    nm32u31b * pSrcVec,
    nm1 * pDstVec,
    int nSize,
    int nTrueFlag )
```

Заметки

nSize =[1,2,3,4...]

## 7.181.2.3 nmppsCmpEq0\_8u7b()

```
void nmppsCmpEq0_8u7b (
    nm8u7b * pSrcVec,
    nm1 * pDstVec,
    int nSize,
    int nTrueFlag )
```

Заметки

nSize =[1,2,3,4...]

## 7.182 nmppsCmpMinMaxV\_

Поэлементное сравнение двух векторов.

## Функции

- void nmppsCmpMinMaxV\_8s (nm8s \*pSrcVec1, nm8s \*pSrcVec2, nm8s \*pDstMin, nm8s \*pDstMax, int nSize)
- void nmppsCmpMinMaxV\_16s (nm16s \*pSrcVec1, nm16s \*pSrcVec2, nm16s \*pDstMin, nm16s \*pDstMax, int nSize)
- void nmppsCmpMinMaxV\_32s (nm32s \*pSrcVec1, nm32s \*pSrcVec2, nm32s \*pDstMin, nm32s \*pDstMax, int nSize)

## 7.182.1 Подробное описание

Поэлементное сравнение двух векторов.

$$pDstMin[i] = \min(pSrcVec1[i], pSrcVec2[i])$$

$$pDstMax[i] = \max(pSrcVec1[i], pSrcVec2[i])$$

$$i = \overline{0 \dots nSize - 1}$$

## Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй Входной вектор.
nSize	Размер векторов в элементах.

## Возвращаемые значения

pDstMin	Массив поэлементных минимумов для входных массивов.
pDstMax	Массив поэлементных максимумов для входных массивов.

## Возвращает

void

## Restrictions:

Сравниваемые пары чисел двух массивов должны отличаться не более чем на  $2^{15} - 1$ .

Примеры допустимых диапазонов сравниваемых пар чисел:

should not be more than  $2^{15} - 1$ .

Here are some examples of admissible ranges for comparable pairs of numbers:

[0000h..7FFFh]=[ 0..+32767 ]

[FFFFh..7FFEh]=[ -1..+32766 ]  
[C000h..3FFFh]=[-16384..+16383 ]  
[8000h..0000h]=[-32768..0 ]

## 7.183 nmppsClipPowC\_

Функция насыщения.

Функции

- void nmppsClipPowC\_8s (nm8s \*pSrcVec, int nClipFactor, nm8s \*pDstVec, int nSize)
- void nmppsClipPowC\_16s (nm16s \*pSrcVec, int nClipFactor, nm16s \*pDstVec, int nSize)
- void nmppsClipPowC\_32s (nm32s \*pSrcVec, int nClipFactor, nm32s \*pDstVec, int nSize)
- void nmppsClipPowC\_64s (nm64s \*pSrcVec, int nClipFactor, nm64s \*pDstVec, int nSize)

### 7.183.1 Подробное описание

Функция насыщения.

$$pDstVec[i] = \begin{cases} -2^{nClipFactor}, & \text{if } pSrcVec[i] < -2^{nClipFactor} \\ pSrcVec[i], & \text{if } -2^{nClipFactor} \leq pSrcVec[i] \leq 2^{nClipFactor} - 1 \\ 2^{nClipFactor} - 1, & \text{if } pSrcVec[i] > 2^{nClipFactor} - 1 \end{cases}$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec	Входной вектор.
nClipFactor	Показатель степени, определяющий верхний и нижний пороги насыщения.
nSize	Размер вектора в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void



## 7.184 nmppsClipCC\_

Функция насыщения с произвольными порогами.

Функции

- void nmppsClipCC\_32s (nm32s30b \*pSrcVec, int30b nNegThresh, int30b nPosThresh, nm32s30b \*pDstVec, int nSize)

## 7.184.1 Подробное описание

Функция насыщения с произвольными порогами.

$$pDstVec[i] = \begin{cases} nPosThresh, & \text{if } pSrcVec[i] > nPosThresh \\ pSrcVec[i], & \text{if } nNegThresh \leq pSrcVec[i] \leq nPosThresh \\ nNegThresh, & \text{if } pSrcVec[i] < nNegThresh \end{cases}$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec	Входной вектор.
nNegThresh	Нижний порог насыщения.
nPosThresh	Верхний порог насыщения.
nSize	Размер вектора в элементах.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

Restrictions:

- Диапазон изменения для \form#65.
- Диапазон изменения для \form#66. \n

## 7.185 nmppsClipRShiftConvert\_AddC\_

Сокращение разрядности данных с предварительной их обработкой.

### Функции

- void nmppsClipRShiftConvertAddC\_16s8s (nm16s \*pSrcVec, int nClipFactor, int nShift, int8b nAddValue, nm8s \*pDstVec, int nSize)
- void nmppsClipRShiftConvertAddC\_32s8s (nm32s \*pSrcVec, int nClipFactor, int nShift, int8b nAddValue, nm8s \*pDstVec, int nSize)

### 7.185.1 Подробное описание

Сокращение разрядности данных с предварительной их обработкой.

$$pDstVec[i] = Convert(Clip(pSrcVec[i], nClipFactor) >> nShift) + nAddValue$$

$$i = \overline{0 \dots nSize - 1}$$

Сокращение разрядности данных выполняется после предварительной обработки и осуществляется путем отбрасывания старших битов.

### Аргументы

pSrcVec	Входной вектор.
nClipFactor	Показатель степени, определяющий верхний и нижний пороги насыщения. =[1,2,3...15]
nShift	Параметр определяет на сколько позиций =[2,4,6...14] нужно сдвинуть биты элементов вектора;
nAddValue	Добавляемая константа.
nSize	Размер векторов в элементах.

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

### Возвращает

void

## 7.186 nmppsClipConvert\_AddC\_

Сокращение разрядности данных с предварительной их обработкой.

Определения типов

- typedef [nm64u](#) NmppsWeightState

Функции

- void nmppsClipConvertAddCInitAlloc\_16s8s (NmppsWeightState \*\*ppState)
- void nmppsClipConvertAddC\_16s8s ([nm16s](#) \*pSrcVec, int nClipFactor, [int8b](#) nAddValue, [nm8s](#) \*pDstVec, int nSize, NmppsWeightState \*pState)
- void nmppsClipConvertAddCFree (NmppsWeightState \*pState)

### 7.186.1 Подробное описание

Сокращение разрядности данных с предварительной их обработкой.

$$pDstVec[i] = Convert(Clip(pSrcVec[i], nClipFactor)) + nAddValue$$

$$i = \overline{0 \dots nSize - 1}$$

Сокращение разрядности данных выполняется после предварительной обработки и осуществляется путем отбрасывания старших битов.

\param pSrcVec

Входной вектор.

Аргументы

nClipFactor	Показатель степени, определяющий верхний и нижний пороги насыщения. = [1,2,3...15]
nAddValue	Добавляемая константа.
nSize	Размер векторов в элементах. указатель на матрицу коэффициентов для векторного умножителя. Для ускорения.

Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.187 nmppsCmpEqC

Сравнивает элементы массива на признак равенства константе.

## Функции

- void nmppsCmpEqC\_16u15b (nm16u15b \*pSrcVec, uint15b nCmpVal, nm16s \*pDstVec, int nSize, int16b nTrueFlag)
- void nmppsCmpEqC\_8u7b (nm8u7b \*pSrcVec, uint7b nCmpVal, nm8s \*pDstVec, int nSize, int8b nTrueFlag)
- void nmppsCmpEqC\_4u3b (nm4u3b \*pSrcVec, uint3b nCmpVal, nm4s \*pDstVec, int nSize, int4b nTrueFlag)

## 7.187.1 Подробное описание

Сравнивает элементы массива на признак равенства константе.

$$pDstVec(i) = \begin{cases} nTrueFlag, & \text{if } pSrcVec(i) = nCmpVal \\ 0, & \text{if } pSrcVec(i) \neq nCmpVal \end{cases}$$

$$i = \overline{0 \dots nSize - 1}$$

## Аргументы

pSrcVec	Входной вектор.
nCmpVal	Значение константы для сравнения
nSize	Размер векторов в элементах.
nTrueFlag	Значение флага, устанавливаемого при выполнении условия

## Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

## Возвращает

void

## 7.188 nmppsCmpNe0

Сравнивает элементы массива на признак неравенства нулю.

### Функции

- void nmppsCmpNe0\_8s (const nm8s \*pSrcVec, nm8s \*pDstVec, int nSize)
- void nmppsCmpNe0\_16s (const nm16s \*pSrcVec, nm16s \*pDstVec, int nSize)
- void nmppsCmpNe0\_32s (const nm32s \*pSrcVec, nm32s \*pDstVec, int nSize)
- void nmppsCmpNe0\_64s (const nm64s \*pSrcVec, nm64s \*pDstVec, int nSize)

### 7.188.1 Подробное описание

Сравнивает элементы массива на признак неравенства нулю.

$$pDstVec(i) = \begin{cases} nTrueFlag, & \text{if } pSrcVec(i) \neq 0 \\ 0, & \text{if } pSrcVec(i) = 0 \end{cases}$$

$$i = \overline{0 \dots nSize - 1}$$

### Аргументы

pSrcVec	Входной вектор.
nCmpVal	Значение константы для сравнения
nSize	Размер векторов в элементах.
nTrueFlag	Значение флага, устанавливаемого при выполнении условия

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

### Возвращает

void

## 7.189 nmppsCmpNeC

Сравнивает элементы массива на признак неравенства константе.

### Группы

- [nmppsCmpEqV\\_](#)  
Поэлементное сравнение элементов двух векторов на признак равенства.
- [nmppsCmpNeV\\_](#)  
Поэлементное сравнение элементов двух векторов на признак неравенства.

### Функции

- void nmppsCmpNeC\_8s (const [nm8s](#) \*pSrcVec, [int8b](#) nCmpVal, [nm8s](#) \*pDstVec, int nSize)
- void nmppsCmpNeC\_16s (const [nm16s](#) \*pSrcVec, [int16b](#) nCmpVal, [nm16s](#) \*pDstVec, int nSize)
- void nmppsCmpNeC\_32s (const [nm32s](#) \*pSrcVec, [int32b](#) nCmpVal, [nm32s](#) \*pDstVec, int nSize)
- void nmppsCmpNeC\_64s (const [nm64s](#) \*pSrcVec, [int64b](#) nCmpVal, [nm64s](#) \*pDstVec, int nSize)
- int nmppsCmpNeC\_8s8um (const [nm8s](#) \*pSrcVec, [int8b](#) nCmpVal, [nm8u](#) \*pDstVec, int nSize, struct [NmppsTmpSpec](#) \*spec)
- int nmppsCmpNeC\_16s8um (const [nm16s](#) \*pSrcVec, [int16b](#) nCmpVal, [nm8u](#) \*pDstVec, int nSize, struct [NmppsTmpSpec](#) \*spec)
- int nmppsCmpNeC\_32s8um (const [nm32s](#) \*pSrcVec, [int32b](#) nCmpVal, [nm8u](#) \*pDstVec, int nSize, struct [NmppsTmpSpec](#) \*spec)
- int nmppsCmpNeC\_64s8um (const [nm64s](#) \*pSrcVec, [int64b](#) nCmpVal, [nm8u](#) \*pDstVec, int nSize, struct [NmppsTmpSpec](#) \*spec)
- void nmppsCmpNeC\_8u7b ([nm8u7b](#) \*pSrcVec, [uint7b](#) nCmpVal, [nm8s](#) \*pDstVec, int nSize, [int8b](#) nTrueFlag)
- void nmppsCmpNeC\_16u15b ([nm16u15b](#) \*pSrcVec, [uint15b](#) nCmpVal, [nm16s](#) \*pDstVec, int nSize, [int16b](#) nTrueFlag)
- int nmppsCmpLtC\_8s8um (const [nm8s](#) \*pSrcVec, [int8b](#) nCmpVal, [nm8u](#) \*pDstVec, int nSize, struct [NmppsTmpSpec](#) \*spec)
- int nmppsCmpLtC\_16s8um (const [nm16s](#) \*pSrcVec, [int16b](#) nCmpVal, [nm8u](#) \*pDstVec, int nSize, struct [NmppsTmpSpec](#) \*spec)
- int nmppsCmpLtC\_32s8um (const [nm32s](#) \*pSrcVec, [int32b](#) nCmpVal, [nm8u](#) \*pDstVec, int nSize, struct [NmppsTmpSpec](#) \*spec)
- int nmppsCmpLtC\_64s8um (const [nm64s](#) \*pSrcVec, [int64b](#) nCmpVal, [nm8u](#) \*pDstVec, int nSize, struct [NmppsTmpSpec](#) \*spec)
- void nmppsCmpLtC\_8s7b (const [nm8s7b](#) \*pSrcVec, [int8b](#) nCmpVal, [nm8s](#) \*pDstVec, int nSize)
- void nmppsCmpLtC\_16s15b (const [nm16s15b](#) \*pSrcVec, [int16b](#) nCmpVal, [nm16s](#) \*pDstVec, int nSize)
- void nmppsCmpLtC\_32s31b (const [nm32s31b](#) \*pSrcVec, [int32b](#) nCmpVal, [nm32s](#) \*pDstVec, int nSize)
- void nmppsCmpLtC\_64s63b (const [nm64s63b](#) \*pSrcVec, [int64b](#) nCmpVal, [nm64s](#) \*pDstVec, int nSize)
- void nmppsCmpGtC\_8s7b (const [nm8s7b](#) \*pSrcVec, [int8b](#) nCmpVal, [nm8s](#) \*pDstVec, int nSize)
- void nmppsCmpGtC\_16s15b (const [nm16s15b](#) \*pSrcVec, [int16b](#) nCmpVal, [nm16s](#) \*pDstVec, int nSize)
- void nmppsCmpGtC\_32s31b (const [nm32s31b](#) \*pSrcVec, [int32b](#) nCmpVal, [nm32s](#) \*pDstVec, int nSize)
- void nmppsCmpGtC\_64s63b (const [nm64s63b](#) \*pSrcVec, [int64b](#) nCmpVal, [nm64s](#) \*pDstVec, int nSize)
- void nmppsCmpNe\_8s (const [nm8s](#) \*pSrcVec1, const [nm8s](#) \*pSrcVec2, [nm8s](#) \*pDstVec, int nSize)

- void nmppsCmpNe\_16s (const nm16s \*pSrcVec1, const nm16s \*pSrcVec2, nm16s \*pDstVec, int nSize)
- void nmppsCmpNe\_32s (const nm32s \*pSrcVec1, const nm32s \*pSrcVec2, nm32s \*pDstVec, int nSize)
- void nmppsCmpNe\_64s (const nm64s \*pSrcVec1, const nm64s \*pSrcVec2, nm64s \*pDstVec, int nSize)
- int nmppsCmpNe\_8s8um (const nm8s \*pSrcVec1, const nm8s \*pSrcVec2, nm8u \*pDstVec, int nSize, struct NmppsTmpSpec \*spec)
- int nmppsCmpNe\_16s8um (const nm16s \*pSrcVec1, const nm16s \*pSrcVec2, nm8u \*pDstVec, int nSize, struct NmppsTmpSpec \*spec)
- int nmppsCmpNe\_32s8um (const nm32s \*pSrcVec1, const nm32s \*pSrcVec2, nm8u \*pDstVec, int nSize, struct NmppsTmpSpec \*spec)
- int nmppsCmpNe\_64s8um (const nm64s \*pSrcVec1, const nm64s \*pSrcVec2, nm8u \*pDstVec, int nSize, struct NmppsTmpSpec \*spec)
- void nmppsCmpLt\_8s7b (const nm8s \*pSrcVec1, const nm8s \*pSrcVec2, nm8s \*pDstVec, int nSize)
- void nmppsCmpLt\_16s15b (const nm16s \*pSrcVec1, const nm16s \*pSrcVec2, nm16s \*pDstVec, int nSize)
- void nmppsCmpLt\_32s31b (const nm32s \*pSrcVec1, const nm32s \*pSrcVec2, nm32s \*pDstVec, int nSize)
- void nmppsCmpLt\_64s63b (const nm64s \*pSrcVec1, const nm64s \*pSrcVec2, nm64s \*pDstVec, int nSize)
- int nmppsCmpLt\_8s8um (const nm8s \*pSrcVec1, const nm8s \*pSrcVec2, nm8u \*pDstVec, int nSize, struct NmppsTmpSpec \*spec)
- int nmppsCmpLt\_16s8um (const nm16s \*pSrcVec1, const nm16s \*pSrcVec2, nm8u \*pDstVec, int nSize, struct NmppsTmpSpec \*spec)
- int nmppsCmpLt\_32s8um (const nm32s \*pSrcVec1, const nm32s \*pSrcVec2, nm8u \*pDstVec, int nSize, struct NmppsTmpSpec \*spec)
- int nmppsCmpLt\_64s8um (const nm64s \*pSrcVec1, const nm64s \*pSrcVec2, nm8u \*pDstVec, int nSize, struct NmppsTmpSpec \*spec)
- void nmppsCmpNeV\_8s8u (nm8s \*src1, nm8s \*src2, nm8u \*dst, int nSize, int8b nTrueFlag)
- void Vec\_ClipRShiftConvert\_AddC (nm32s \*pSrcVec, nm8u \*pDstVec, int nSize)

### 7.189.1 Подробное описание

Сравнивает элементы массива на признак неравенства константе.

$$pDstVec(i) = \begin{cases} nTrueFlag, & \text{if } pSrcVec(i) \neq nCmpVal \\ 0, & \text{if } pSrcVec(i) = nCmpVal \end{cases}$$

$$i = \overline{0 \dots nSize - 1}$$

Аргументы

pSrcVec	Входной вектор.
nCmpVal	Значение константы для сравнения
nSize	Размер векторов в элементах.
nTrueFlag	Значение флага, устанавливаемого при выполнении условия



Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

Возвращает

void

## 7.190 nmppsCmpEqV\_

Поэлементное сравнение элементов двух векторов на признак равенства.

### Функции

- void nmppsCmpEqV\_16u15b (nm16u15b \*pSrcVec1, nm16u15b \*pSrcVec2, nm16s \*pDstVec, int nSize, int16b nTrueFlag)
- void nmppsCmpEqV\_8u7b (nm8u7b \*pSrcVec1, nm8u7b \*pSrcVec2, nm8s \*pDstVec, int nSize, int8b nTrueFlag)

### 7.190.1 Подробное описание

Поэлементное сравнение элементов двух векторов на признак равенства.

$$pDstVec(i) = \begin{cases} nTrueFlag, & \text{if } pSrcVec(i) = nCmpVal \\ 0, & \text{if } pSrcVec(i) \neq nCmpVal \end{cases}$$

$$i = \overline{0 \dots nSize - 1}$$

### Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
nSize	Размер векторов в элементах.
nTrueFlag	Значение флага, устанавливаемого при выполнении условия

### Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

### Возвращает

void

## 7.191 nmppsCmpNeV\_

Поэлементное сравнение элементов двух векторов на признак неравенства.

## Функции

- void nmppsCmpNeV\_16u (nm16u15b \*pSrcVec1, nm16u15b \*pSrcVec2, nm16s \*pDstVec, int nSize, int16b nTrueFlag)
- void nmppsCmpNeV\_8u (nm8u7b \*pSrcVec1, nm8u7b \*pSrcVec2, nm8s \*pDstVec, int nSize, int8b nTrueFlag)

## 7.191.1 Подробное описание

Поэлементное сравнение элементов двух векторов на признак неравенства.

$$pDstVec(i) = \begin{cases} nTrueFlag, & \text{if } pSrcVec(i) \neq nCmpVal \\ 0, & \text{if } pSrcVec(i) = nCmpVal \end{cases}$$

$$i = \overline{0 \dots nSize - 1}$$

## Аргументы

pSrcVec1	Первый входной вектор.
pSrcVec2	Второй входной вектор.
nSize	Размер векторов в элементах.
nTrueFlag	Значение флага, устанавливаемого при выполнении условия

## Возвращаемые значения

pDstVec	Результирующий вектор.
---------	------------------------

## Возвращает

void

## 7.192 nmppsAddr\_

Возвращает адрес ячейки памяти, содержащей указанный элемент.

Реализация для процессора NeuroMatrix возвращает адрес, выровненный в памяти на 32 бита.

### Функции

- `__INLINE__ nm1 * nmppsAddr_1 (const nm1 *pVec, int nIndex)`
- `__INLINE__ nm2s * nmppsAddr_2s (const nm2s *pVec, int nIndex)`
- `__INLINE__ nm4s * nmppsAddr_4s (const nm4s *pVec, int nIndex)`
- `__INLINE__ nm8s * nmppsAddr_8s (const nm8s *pVec, int nIndex)`
- `__INLINE__ nm16s * nmppsAddr_16s (const nm16s *pVec, int nIndex)`
- `__INLINE__ nm32s * nmppsAddr_32s (const nm32s *pVec, int nIndex)`
- `__INLINE__ nm64s * nmppsAddr_64s (const nm64s *pVec, int nIndex)`
- `__INLINE__ nm2u * nmppsAddr_2u (const nm2u *pVec, int nIndex)`
- `__INLINE__ nm4u * nmppsAddr_4u (const nm4u *pVec, int nIndex)`
- `__INLINE__ nm8u * nmppsAddr_8u (const nm8u *pVec, int nIndex)`
- `__INLINE__ nm16u * nmppsAddr_16u (const nm16u *pVec, int nIndex)`
- `__INLINE__ nm32u * nmppsAddr_32u (const nm32u *pVec, int nIndex)`
- `__INLINE__ nm64u * nmppsAddr_64u (const nm64u *pVec, int nIndex)`

### 7.192.1 Подробное описание

Возвращает адрес ячейки памяти, содержащей указанный элемент.

Реализация для процессора NeuroMatrix возвращает адрес, выровненный в памяти на 32 бита.

#### Аргументы

pVec	Входной вектор.
nIndex	Индекс элемента.

#### Возвращает

Адрес ячейки памяти.

#### Заметки

Для ускорения работы на PC возможно использование макроса `ADDR(ptr, index)`, который раскрывается на PC как `(ptr+index)`, а на NM как вызов функции `nmppsAddr_`.

## 7.193 nmppsSetVal\_

Модификация элемента вектора.

## Функции

- void nmppsPut\_1 (nm1 \*pVec, int nIndex, int1b nVal)
- void nmppsPut\_2s (nm2s \*pVec, int nIndex, int2b nVal)
- void nmppsPut\_4s (nm4s \*pVec, int nIndex, int4b nVal)
- void nmppsPut\_8s (nm8s \*pVec, int nIndex, int8b nVal)
- void nmppsPut\_16s (nm16s \*pVec, int nIndex, int16b nVal)
- \_\_INLINE\_\_ void nmppsPut\_32s (nm32s \*pVec, int nIndex, int32b nVal)
- \_\_INLINE\_\_ void nmppsPut\_64s (nm64s \*pVec, int nIndex, int64b nVal)
- \_\_INLINE\_\_ void nmppsPut\_2u (nm2u \*pVec, int nIndex, uint2b nVal)
- \_\_INLINE\_\_ void nmppsPut\_4u (nm4u \*pVec, int nIndex, uint4b nVal)
- \_\_INLINE\_\_ void nmppsPut\_8u (nm8u \*pVec, int nIndex, uint8b nVal)
- \_\_INLINE\_\_ void nmppsPut\_16u (nm16u \*pVec, int nIndex, uint16b nVal)
- \_\_INLINE\_\_ void nmppsPut\_32u (nm32u \*pVec, int nIndex, uint32b nVal)
- \_\_INLINE\_\_ void nmppsPut\_64u (nm64u \*pVec, int nIndex, uint64b nVal)

## 7.193.1 Подробное описание

Модификация элемента вектора.

$$pVec(nIndex) = Val$$

## Аргументы

pVec	Вектор.
nIndex	Позиция элемента
nVal	Значение элемента

Возвращает

void

## 7.194 nmppsGetVal\_

Извлекает значение элемента вектора.

### Функции

- void nmppsGetVal\_1 (const nm1 \*pVec, int nIndex, int1b \*nVal)
- void nmppsGetVal\_2s (const nm2s \*pVec, int nIndex, int2b \*nVal)
- void nmppsGetVal\_4s (const nm4s \*pVec, int nIndex, int4b \*nVal)
- void nmppsGetVal\_8s (const nm8s \*pVec, int nIndex, int8b \*nVal)
- void nmppsGetVal\_16s (const nm16s \*pVec, int nIndex, int16b \*nVal)
- \_\_INLINE\_\_ void nmppsGetVal\_32s (const nm32s \*pVec, int nIndex, int32b \*nVal)
- \_\_INLINE\_\_ void nmppsGetVal\_64s (const nm64s \*pVec, int nIndex, int64b \*nVal)
- void nmppsGetVal\_2u (const nm2u \*pVec, int nIndex, uint2b \*nVal)
- void nmppsGetVal\_4u (const nm4u \*pVec, int nIndex, uint4b \*nVal)
- void nmppsGetVal\_8u (const nm8u \*pVec, int nIndex, uint8b \*nVal)
- void nmppsGetVal\_16u (const nm16u \*pVec, int nIndex, uint16b \*nVal)
- \_\_INLINE\_\_ void nmppsGetVal\_32u (const nm32u \*pVec, int nIndex, uint32b \*nVal)
- \_\_INLINE\_\_ void nmppsGetVal\_64u (const nm64u \*pVec, int nIndex, uint64b \*nVal)

### 7.194.1 Подробное описание

Извлекает значение элемента вектора.

#### Аргументы

pVec	Вектор.
nIndex	Позиция элемента.

#### Возвращаемые значения

nVal	Значение элемента.
------	--------------------

#### Возвращает

void

## 7.195 nmppsGetVal\_(return)

Извлекает значение элемента вектора.

## Функции

- [int2b](#) nmppsGet\_2s (const [nm2s](#) \*pVec, int nIndex)
- [int4b](#) nmppsGet\_4s (const [nm4s](#) \*pVec, int nIndex)
- [int8b](#) nmppsGet\_8s (const [nm8s](#) \*pVec, int nIndex)
- [int16b](#) nmppsGet\_16s (const [nm16s](#) \*pVec, int nIndex)
- `__INLINE__` [int32b](#) nmppsGet\_32s (const [nm32s](#) \*pVec, int nIndex)
- [uint1b](#) nmppsGet\_1 (const [nm1](#) \*pVec, int nIndex)
- [uint2b](#) nmppsGet\_2u (const [nm2u](#) \*pVec, int nIndex)
- [uint4b](#) nmppsGet\_4u (const [nm4u](#) \*pVec, int nIndex)
- [uint8b](#) nmppsGet\_8u (const [nm8u](#) \*pVec, int nIndex)
- [uint16b](#) nmppsGet\_16u (const [nm16u](#) \*pVec, int nIndex)
- `__INLINE__` [uint32b](#) nmppsGet\_32u (const [nm32u](#) \*pVec, int nIndex)

## 7.195.1 Подробное описание

Извлекает значение элемента вектора.

## Аргументы

pVec	Вектор.
nIndex	Позиция элемента.

## Возвращает

Значение элемента.

## 7.196 VEC\_QSort

Сортировка массива по убыванию.

Функции

- void nmppsQSort\_32s (nm32s \*pSrcDstVec, int nSize)

### 7.196.1 Подробное описание

Сортировка массива по убыванию.

Аргументы

pSrcDstVec	Входной и результирующий вектор.
nSize	Размер вектора в элементах.

Возвращает

void

Restrictions:

Функция работает рекурсивно, передавая параметры через стек, поэтому размер стека должен быть больше  $4 \cdot \log_2(nSize)$  32-битных слов в лучшем случае (элементы массива расположены беспорядочно) и больше  $6 \cdot nSize$  32-битных слов в худшем случае (элементы массива уже упорядочены)



## 7.197 nmppsRemap\_

Переупорядочивание элементов вектора по таблице.

## Функции

- void nmppsRemap\_32u (nm32u \*pSrcVec, nm32u \*pDstVec, nm32s \*pRemapTable, int nDstVecSize)
- void nmppsRemap\_8u (nm8u \*pSrcVec, nm8u \*pDstVec, nm32s \*pRemapTable, int nSrcVecSize, int nDstVecSize, void \*pTmpBuf1, void \*pTmpBuf2)

## 7.197.1 Подробное описание

Переупорядочивание элементов вектора по таблице.

$$pDstVec[i] = pSrcVec[pRemapTable[i]],$$

$$i = \overline{0 \dots nSize - 1}$$

## Аргументы

pSrcVec	Входной вектор.
pRemapTable	Таблица новых индексов для переупорядочивания.
nDstVecSize	Размер результирующего вектора в элементах.
pTmpBuf1	Временный массив nm32u pTmpBuf1[nSrcVecSize].
pTmpBuf2	Временный массив nm32u pTmpBuf2[nDstVecSize]. Результирующий вектор nm8u pDstVec[nDstVecSize].

## Возвращает

void

```
// Функция
// void nmppsRemap_8u(nm8u* pSrcVec, nm8u* pDstVec, nm32s* pRemapTable, int nSrcVecSize, int nDstVecSize,
// void* pTmpBuf1, void* pTmpBuf2);
// выполняет следующие действия:
nmppsConvert_8u((nm8u*) pSrcVec, (nm32u*)pTmpBuf1,nSrcVecSize);
nmppsRemap_32u((nm32u*)pTmpBuf1,(nm32u*)pTmpBuf2,RemapTable,DstVecSize);
nmppsConvert_32s((nm32s*)pTmpBuf2,(nm8s*) pDstVec, DstVecSize);
```

## Заметки

Возможность использования inplace параметров определяется исходя из последовательности процессов чтения/записи:

the folowing sequence of actions :

- pSrcVec => pTmpBuf1 - inplace запрещен;

- $pTmpBuf1 \Rightarrow pTmpBuf2$  - inplace запрещен;
- $pTmpBuf2 \Rightarrow pDstVec$  - inplace разрешен;

## 7.198 nmppSplitTmp

Расщепляет массив на два, группируя по четным и нечетным элементам

Расщепляет массив на два, группируя по четным и нечетным элементам

Аргументы

in	src	Входной массив
out	dst1	Выходной массив размера size/2
out	dst2	Выходной массив размера size/2
in	size	Размер исходного массива в элементах. Кратность параметра size должна соответствовать двум длинным 64-р. словам.
in	tmpSizeOfDst	Временный массив размера size/2

Возвращает

Details Максимальная производительность достигается при размещении входных, выходных и временных массивов в разных банках памяти. Массивы dst1 и dst2 могут находиться в одном банке. Макс производительность на 64-р. слово результата = 2.1 такта (при size=10240 байт) и 2.5 такта (при size=4096 байт)

## 7.199 nmppSplit

Расщепляет массив на два массива, группируя по четным и нечетным элементам

Расщепляет массив на два массива, группируя по четным и нечетным элементам

Аргументы

in	src	Входной массив
out	dst1	Выходной массив размера size/2
out	dst2	Выходной массив размера size/2
in	size	Размер исходного массива в элементах. Кратность параметра size должна соответствовать двум длинным 64-р. словам.

Возвращает

Details Максимальная производительность достигается при размещении входных, выходных массивов в разных банках памяти. Массивы dst1 и dst2 могут находиться в одном банке. Макс производительность на 64-р. слово результата = 2.14 такта (при size=10240 байт) и 2.6 такта (при size=4096 байт)

## 7.200 nmppMerge

Собирает массив из двух, чередуя элементы из каждого. Функция обратная nmppsSplit.

Собирает массив из двух, чередуя элементы из каждого. Функция обратная nmppsSplit.

Аргументы

in	src0	Входной массив размера size
in	src1	Входной массив размера size
out	dst	Выходной массив размера 2*size
in	size	Размер выходного массива в элементах. Кратность параметра size должна соответствовать 64-р. слову.

Возвращает

Details

## 7.201 nmppSplit\_32fcr

Расщепляет массив на два, группируя по четным и нечетным элементам

Расщепляет массив на два, группируя по четным и нечетным элементам

Аргументы

in	pSrcVec	Входной массив
out	pDstVec1	Выходной массив размера size/2
out	pDstVec2	Выходной массив размера size/2
in	sizeSrc	Размер исходного массива в элементах (должен быть четным)

Возвращает

Details Максимальная производительность достигается при размещении входных, выходных массивов в разных банках памяти. Массивы dst1 и dst2 могут находиться в одном банке. Макс производительность на 64-р. слово результата = 1 такт

## 7.202 nmppsDecimate

Делает выборку элементов из массива с некоторым шагом

Делает выборку элементов из массива с некоторым шагом

Аргументы

in	pSrc	Входной массив
in	startPos	Положение элемента в 64-р. слове
out	step	Шаг выборки. Кратность параметра step должна соответствовать длинному 64-р. слову.
out	pDst	Выходной массив
in	size	Размер исходного массива в элементах. Кратность параметра size должна соответствовать длинному 64-р. слову.

Возвращает

Details Максимальная производительность достигается при размещении входных, выходных массивов в разных банках памяти.

## 7.203 Типы данных

### Группы

- [Типы векторных данных](#)
- [Типы скалярных данных](#)

#### 7.203.1 Подробное описание



## 7.204 Векторные функции

### Группы

- [Функции поддержки](#)
- [Инициализация и копирование](#)
- [Арифметические операции](#)
- [Логические и бинарные операции](#)
- [Операции сравнения](#)
- [Переупорядочивание и сортировка](#)

### 7.204.1 Подробное описание

## 7.205 Матричные функции

### Группы

- [Инициализация и копирование](#)
- [Функции поддержки](#)
- [Векторно-матричные операции](#)

#### 7.205.1 Подробное описание

## 7.206 Функции обработки сигналов

### Группы

- [Свертка](#)
- [Масочная фильтрация](#)
- [Изменение размеров](#)
- [Быстрое преобразование Фурье](#)

### 7.206.1 Подробное описание

## 7.207 Функции обработки изображений

### Группы

- [Floodfill](#)

Исполняет разделение бинарной картинки на односвязные области. Пример вызова: `no=VL_↵ FloodFill32b(pSrcImage, Tetr,Image, pTmpBuff, nSrcWidth, nSrcHeight);`.

- [Переупорядочивание изображений](#)
- [Арифметические действия](#)
- [Масочная фильтрация](#)
- [Инициализация и копирование](#)
- [Функции поддержки](#)
- [Функции графического вывода текста](#)

### 7.207.1 Подробное описание

## 7.208 Скалярные функции

### Группы

- [Инициализация](#)
- [Integer operations](#)
- [Fix-point 64](#)
- [Fix-point 32](#)
- [Арифметические операции](#)
- [Функции деинтерлейсинга](#)

### 7.208.1 Подробное описание

#### 7.208.1.1 Введение

Назначением данной библиотеки является предоставление базовых операций по работе со скалярными данными для процессора NM6403, NM6404, NM6405.

В состав библиотеки входят арифметические, тригонометрические функции, функции для работы с данными в формате с фиксированной точкой.

Библиотека предназначена для быстрой разработки эффективных пользовательских программ на языке высокого уровня(C++).

Назначением данной библиотеки является предоставление базовых операций обработки изображений для процессора NM6403, NM6404, NM6405. В состав библиотеки входят функции двумерной фильтрации, арифметические действия и цветовые преобразования. Библиотека предназначена для быстрой разработки эффективных пользовательских программ на языке высокого уровня с использованием преимуществ архитектуры данного процессора.

Функции библиотеки имеют C++ интерфейс. Большинство функций библиотеки реализованы на языке ассемблера с использованием векторных инструкций и оптимизированы под архитектуру процессора NM6403.

Для удобства разработки прикладных программ библиотека содержит аналогичные реализации функций для процессоров серии x86, выполненных на языке C++. Данные реализации позволяют выполнять написанные с использованием данной библиотеки прикладные программы на персональном компьютере.

Назначением данной библиотеки является предоставление базовых операций по обработке матриц для процессорах NM6403, NM6404, NM6405. В состав библиотеки входят арифметические операции над матрицами. Библиотека предназначена для быстрой разработки эффективных пользовательских программ как на языке высокого уровня(C++).

Функции библиотеки имеют C++ интерфейс. Большинство функций библиотеки реализованы на языке ассемблера с использованием векторных инструкций и оптимизированы под архитектуру процессора NM6403.

Для удобства разработки прикладных программ библиотека содержит аналогичные реализации функций для процессоров серии x86, выполненных на языке C++. Данные реализации позволяют выполнять написанные с использованием данной библиотеки прикладные программы на персональном компьютере.

Назначением данной библиотеки является предоставление базовых функций по обработке сигналов для процессоров NM6403,NM6404,NM6405. В состав библиотеки входят функции одномерной КИХ

фильтрации, нелинейной фильтрации, передискретизации. Библиотека предназначена для быстрой разработки эффективных пользовательских программ как на языке высокого уровня(C++).

Функции библиотеки имеют C++ интерфейс. Большинство функций библиотеки реализованы на языке ассемблера с использованием векторных инструкций и оптимизированы под архитектуру процессоров NM6403.

Для удобства разработки прикладных программ библиотека содержит аналогичные реализации функций для процессоров серии x86, выполненных на языке C++. Данные реализации позволяют выполнять написанные с использованием данной библиотеки прикладные программы на персональном компьютере.

Назначением данной библиотеки является предоставление базовых операций по обработке одномерных массивов (векторов) для процессоров NM6405, NM6406, систем на кристалле с ядром NMC.

В состав библиотеки входят логические и арифметические функции, операции сравнения, инициализации, копирования, преобразования разрядностей и т.п. Библиотека предназначена для быстрой разработки эффективных пользовательских программ как на языке высокого уровня(C++), так и на языке ассемблера с помощью прилагаемой библиотеки ядра низкоуровневых функций. Функции библиотеки имеют C++ интерфейс.

Большинство функций библиотеки реализованы на языке ассемблера с использованием векторных инструкций и оптимизированы под архитектуру процессоров NMC. Для удобства разработки прикладных программ библиотека содержит аналогичные реализации функций для процессоров серии x86, выполненных на языке C++. Данные реализации позволяют выполнять написанные с использованием данной библиотеки прикладные программы на персональном компьютере.

Функции векторного ядра библиотеки

Функции различных библиотек: nmplv, nmpls, nmpli, nmplm и др. , имеющие C++ интерфейсы, в своей реализации используют вызовы функций ядра. Функции ядра не имеют C++ интерфейса. Их вызов возможен только из ассемблера процессора NeuroMatrix. Передача параметров и настройка функций производится через регистры.

Одна и та же функция ядра может использоваться при реализации одной

или нескольких функций библиотеки. Функции ядра также могут быть использованы для реализации пользовательских функций. Использование функций ядра позволяет минимизировать время разработки, уменьшить размер кода и получить максимальную производительность.

## 7.209 Базовые регистровые функции библиотеки

### Группы

- [Элементарные функции](#)
- [функции взвешенного суммирования](#)
- [Целевые функции](#)

### 7.209.1 Подробное описание

## 7.210 контроль переполнения

### Классы

- class `nmmtr< T >`
- class `nmvecpack< T >`
- class `vec< T >`

### Макросы

- `#define GetVec getvec`

### Функции

- `__INLINE__ ostream & operator<< (ostream &s, mtr< unsigned char > &mtr)`

### 7.210.1 Подробное описание

определяет классы, предназначенные для контроля переполнения при реализации библиотеки на РС.

—реализации библиотеки на РС производит контроль переполнения с выдачей сообщения об ошибке пользователю библиотеки. Для этой цели определены шаблонные классы для вектора, матрицы и скалярного числа, позволяющие производить базовые операции над их элементами.

### 7.210.2 Макросы

#### 7.210.2.1 GetVec

```
#define GetVec getvec
```

Класс матриц.

Примеры:

```
int Test[10]={1,125,3,4,5,6,7,8,9,10};
int Res [10];

mtr<int> AA0(3,3);
vec<int> A0(3);
scalar<int> a0(2);

mtr<int> BB0(3,3);
mtr<int> CC0(3,3);
vec<int> C0(3);

BB0.SetDataa(Test);

BB0=AA0;
A0 =AA0[2];
a0 =AA0[2][2];

CC0=AA0+BB0;
CC0=AA0*a0;
C0 =AA0*A0;
CC0=AA0*BB0;
```

См. определение в файле `tmatrix.h` строка 60



### 7.210.3 Функции

#### 7.210.3.1 operator<<()

```
__INLINE__ ostream& operator<< (  
    ostream & s,  
    mtr< unsigned char > & mtr )
```

Класс матриц.

Примеры:

```
int Test[10]={1,125,3,4,5,6,7,8,9,10};  
int Res [10];  
  
mtr<int> AA0(3,3);  
vec<int> A0(3);  
scalar<int> a0(2);  
  
mtr<int> BB0(3,3);  
mtr<int> CC0(3,3);  
vec<int> C0(3);  
  
BB0.SetData( Test);  
  
BB0=AA0;  
A0 =AA0[2];  
a0 =AA0[2][2];  
  
CC0=AA0+BB0;  
CC0=AA0*a0;  
C0 =AA0*A0;  
CC0=AA0*BB0;
```

См. определение в файле nmtlio.h строка 64

## 7.211 Элементарные функции

### Группы

- [Vec\\_0\\_sub\\_data](#)
- [Vec\\_activate\\_data](#)
- [Vec\\_activate\\_data\\_add\\_0](#)
- [Vec\\_activate\\_data\\_xor\\_data](#)
- [Vec\\_activate\\_data\\_add\\_ram](#)
- [Vec\\_affo](#)
- [Vec\\_data](#)
- [Vec\\_data\\_add\\_ram](#)
- [Vec\\_data\\_and\\_ram](#)
- [Vec\\_data\\_or\\_ram](#)
- [Vec\\_data\\_sub\\_ram](#)
- [Vec\\_data\\_xor\\_ram](#)
- [Vec\\_And](#)
- [Vec\\_Mask](#)
- [Vec\\_Or](#)
- [Vec\\_Xor](#)
- [Vec\\_Add](#)
- [Vec\\_Sub](#)
- [Vec\\_not\\_data](#)
- [Vec\\_ram](#)
- [Vec\\_ram\\_sub\\_data](#)
- [Vec\\_vsum\\_activate\\_data\\_0](#)

### 7.211.1 Подробное описание

## 7.212 функции взвешенного суммирования

### Группы

- [Vec\\_ClipMul2D2W8\\_AddVr](#)
- [Vec\\_ClipMulNDNW2\\_AddVr](#)
- [Vec\\_ClipMulNDNW4\\_AddVr](#)
- [Vec\\_ClipMulNDNW8\\_AddVr](#)
- [Vec\\_Mul2D2W1\\_AddVr](#)
- [Vec\\_Mul2D2W2\\_AddVr](#)
- [Vec\\_Mul2D2W4\\_AddVr](#)
- [Vec\\_Mul2D2W8\\_AddVr](#)
- [Vec\\_Mul3D3W2\\_AddVr](#)
- [Vec\\_Mul3D3W8\\_AddVr](#)
- [Vec\\_Mul4D4W2\\_AddVr](#)
- [Vec\\_MulVN\\_AddVN](#)
- [Vec\\_vsum\\_data\\_0](#)
- [Vec\\_vsum\\_data\\_vr](#)
- [Vec\\_vsum\\_shift\\_data\\_0](#)
- [Vec\\_vsum\\_shift\\_data\\_vr](#)
- [Vec\\_vsum\\_shift\\_data\\_afifo](#)

### 7.212.1 Подробное описание

## 7.213 Целевые функции

### Группы

- [Vec\\_Add\\_VV\\_shift](#)
- [Vec\\_data\\_add\\_afifo](#)
- [Vec\\_FilterCoreRow2](#)
- [Vec\\_FilterCoreRow4](#)
- [Vec\\_FilterCoreRow8](#)
- [Vec\\_Abs](#)
- [Vec\\_ClipExt](#)
- [Vec\\_IncNeg](#)
- [Vec\\_SubAbs](#)
- [Vec\\_SubVN\\_Abs](#)
- [Vec\\_Swap](#)
- [Vec\\_MUL\\_2V4toW8\\_shift](#)
- [Vec\\_MUL\\_2V8toW16\\_shift](#)
- [Vec\\_vsum\\_data\\_afifo](#)
- [Vec\\_CompareMinV](#)

Поэлементный поиск минимального

Действие функции эквивалентно следующим псевдоинструкциям:

- [Vec\\_CompareMaxV](#)

Поэлементный поиск максимального

Действие функции эквивалентно следующим псевдоинструкциям:

- [Vec\\_DupValueInVector8](#)

Размножение 8-ми битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

- [Vec\\_DupValueInVector16](#)

Размножение 16-ти битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

- [Vec\\_BuildDiagWeights8](#)

Построение диагональной матрицы весовых коэффициентов (8x8).

- [Vec\\_BuildDiagWeights16](#)

Построение диагональной матрицы весовых коэффициентов (16x16).

- [Vec\\_MaxVal\\_v8nm8s](#)

Поиск максимума в 8 байтах

- [Vec\\_MaxVal\\_v4nm16s](#)

Поиск максимума в 4-х 16р. элементах

- [Vec\\_MaxVal](#)

Поиск максимумов в колонках матрицы SrcMtr1.

- [Vec\\_MinVal\\_v8nm8s](#)

Поиск минимума в 8 байтах

- [Vec\\_MinVal\\_v4nm16s](#)

Поиск минимума в 4-х 16р. элементах

- [Vec\\_MinVal](#)

Поиск минимумов в колонках матрицы SrcMtr1.

- [Vec\\_AccMul1D1W32\\_AddVr](#)

Умножение с накоплением

Действие функции эквивалентно следующим псевдоинструкциям:

### 7.213.1 Подробное описание

## 7.214 Vec\_0\_sub\_data

## Функции

- void vec\_0\_sub\_data (nmreg nb1, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.214.1 Подробное описание

Ядро функции nmppsNeg().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with 0-data;
rep N [ar6++gr6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.215 Vec\_activate\_data

### Функции

- void vec\_activate\_data (nmreg f1cr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.215.1 Подробное описание

Ядро функции nmppsCmpLt0\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with activate data;
rep N [ar6++gr6]=afifo;
```

### Аргументы

f1cr	задает функцию активации
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.216 Vec\_activate\_data\_add\_0

### Функции

- void vec\_activate\_data\_add\_0 (nmreg flcr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.216.1 Подробное описание

Функция производит арифметическую активацию.

Ядро функции nmppsClipPowC\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with activate data + 0;
rep N [ar6++gr6]=aifo;
```

### Аргументы

flcr	задает функцию активации
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.217 Vec\_activate\_data\_xor\_data

### Функции

- void vec\_activate\_data\_xor\_data (nmreg flcr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.217.1 Подробное описание

Функция позволяет вычислить приближенное значение модуля.

Ядро функции nmppsAbs1().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with activate data xor data;
rep N [ar6++gr6]=aifo;
```

### Аргументы

flcr	задает функцию активации
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц $N = [0,1,2...31,32,33,...]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.



## 7.218 Vec\_activate\_data\_add\_ram

### Функции

- void vec\_activate\_data\_add\_ram (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.218.1 Подробное описание

Функция выполняет арифметическую активацию с прибавлением константы.

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram = [ar1]
rep N data = [ar0 ++ gr0] with activate data + ram;
rep N [ar6 ++ gr6] = afifo;
```

### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
flcr	задает функцию активации
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на 64р. слово
gr5	Высота матриц $N = [0,1,2...31,32,33,...]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0, ar6.

## 7.219 Vec\_Add\_VV\_shift

### Функции

- void vec\_Add\_VV\_shift (nmreg nb1, nmreg sb, nmreg woper, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.219.1 Подробное описание

Функция служит для суммирования двух массивов со сдвигом результата на 1 бит вправо.

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram = [ar4];
rep N data = [ar0++gr0] with data + 0;
rep N data = [ar1++gr1] with data + aifo;
rep N with mask ram,shift aifo,0;
rep N [ar6++gr6] = aifo;
```

### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на строки
woper	в рабочей матрице должны быть загружены весовые коэффициенты
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar4	указатель на 64р. слово (маска)
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar2,ar6,gr2,gr5.

## 7.220 Vec\_afifo

### Функции

- void vec\_afifo (nmreg ar0, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.220.1 Подробное описание

Функция служит для заполнения массива константой.

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 1 data=[ar0] with data;  
rep 1*N [ar6++gr6]=afifo with afifo;
```

### Аргументы

ar0	указатель на 64р. слово
gr5	Высота матриц $N = [0,1,2...31,32,33,...]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.221 Vec\_data

### Функции

- void vec\_data (nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.221.1 Подробное описание

Ядро функции nmppsCopy\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data;
rep N [ar6++gr6]=aifo;
```

### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.222 Vec\_data\_add\_afifo

## Функции

- void vec\_data\_add\_afifo (nmreg nb1, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6)

## 7.222.1 Подробное описание

Ядро функции nmppsSum().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 1  data=vfalse;
.repeat N;
rep 1  data=[ar0++gr0] with data + afifo; (rep1 N times)
.endrepeat;
rep 1  [ar6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на 64р. слово

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0.

## 7.223 Vec\_data\_add\_ram

### Функции

- void vec\_data\_add\_ram (nmreg nb1, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.223.1 Подробное описание

Ядро функции nmppsAddC().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar1];
rep N data=[ar0++gr0] with data+ram;
rep N [ar6++gr6]=aiffo;
```

### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr stride
ar1	указатель на 64р. слово-константу
gr5	Высота матриц $N = [0,1,2...31,32,33,...]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.224 Vec\_data\_and\_ram

## Функции

- void vec\_data\_and\_ram (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.224.1 Подробное описание

Ядро функции nmppsAndC\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar1];
rep N data=[ar0++gr0] with data and ram;
rep N [ar6++gr6]=afifo;
```

## Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на 64р. слово-константу
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.225 Vec\_data\_or\_ram

### Функции

- void vec\_data\_or\_ram (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.225.1 Подробное описание

Ядро функции nmppsOrC\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar1];
rep N data=[ar0++gr0] with data or ram;
rep N [ar6++gr6]=afifo;
```

### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на 64-е слово-константу
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.



## 7.226 Vec\_data\_sub\_ram

## Функции

- void vec\_data\_sub\_ram (nmreg nb1, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.226.1 Подробное описание

Ядро функции nmppsSubC().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar1];
rep N data=[ar0++gr0] with data-ram;
rep N [ar6++gr6]=aiffo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
ar1	указатель на 64р. слово-константу
gr5	Высота матриц $N = [0,1,2...31,32,33,...]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.227 Vec\_data\_xor\_ram

### Функции

- void vec\_data\_xor\_ram (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.227.1 Подробное описание

Ядро функции nmppsXorC\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram = [ar1];
rep N data = [ar0++gr0] with data xor ram;
rep N [ar6++gr6] = afifo;
```

### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на 64р. слово-константу
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.228 Vec\_FilterCoreRow2

### Функции

- void vec\_FilterCoreRow2 (nmreg ar0, nmreg ar4, nmreg ar6, nmreg gr1, nmreg gr4, nmreg gr6)

#### 7.228.1 Подробное описание

## 7.229 Vec\_FilterCoreRow4

### Функции

- void vec\_FilterCoreRow4 (nmreg ar0, nmreg ar4, nmreg ar6, nmreg gr1, nmreg gr4, nmreg gr6)

#### 7.229.1 Подробное описание

## 7.230 Vec\_FilterCoreRow8

### Функции

- void vec\_FilterCoreRow8 (nmreg ar0, nmreg ar4, nmreg ar6, nmreg gr1, nmreg gr4, nmreg gr6)

#### 7.230.1 Подробное описание

## 7.231 Vec\_And

### Функции

- void vec\_And (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.231.1 Подробное описание

Ядро функции nmppsAnd\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data;
rep N data=[ar1++gr1] with data and aifo;
rep N [ar6++gr6]=aifo;
```

### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar6.

## 7.232 Vec\_Mask

### Функции

- void vec\_Mask (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar2, nmreg gr2, nmreg gr5, nmreg ar6, nmreg gr6)

#### 7.232.1 Подробное описание

Ядро функции nmppsMaskV\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar0++gr0];
rep N data=[ar1++gr1] with data;
rep N data=[ar2++gr2] with mask data,ram,afifo;
rep N [ar6++gr6]=afifo;
```

### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar2	указатель на столбец SrcMtr2 (маска)
gr2	SrcMtr3 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar2,ar6.

## 7.233 Vec\_Or

### Функции

- void vec\_Or (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.233.1 Подробное описание

Ядро функции nmppsOr\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data;
rep N data=[ar1++gr1] with data or aifo;
rep N [ar6++gr6]=aifo;
```

### Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar6.



## 7.234 Vec\_Xor

## Функции

- void vec\_Xor (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.234.1 Подробное описание

Ядро функции nmppsXor\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data;
rep N data=[ar1++gr1] with data xor affo;
rep N [ar6++gr6]=affo;
```

## Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar6.

## 7.235 Vec\_Abs

### Функции

- void vec\_Abs (nmreg nb1, nmreg sb, nmreg flcr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.235.1 Подробное описание

Ядро функции nmppsAbs().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram = [ar0++gr0] with activate data;
rep N          with vsum afifo,ram,ram;
rep N [ar6++gr6] = afifo;
```

### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
flcr	задает функцию активации
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
gr5	Высота матриц $N = [0,1,2...31,32,33,...]$
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.236 Vec\_Add

## Функции

- void vec\_Add (nmreg nb1, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.236.1 Подробное описание

Ядро функции nmppsAdd().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data;
rep N data=[ar1++gr1] with data + aifo;
rep N [ar6++gr6]=aifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar6.

## 7.237 Vec\_ClipExt

### Функции

- `void vec_ClipExt (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg ar2, nmreg ar3, nmreg gr5, nmreg ar6, nmreg gr6)`

### 7.237.1 Подробное описание

Ядро функции `nmppsClipCC_()`.

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram = [ar1];
rep N data = [ar0++gr0] with data-ram;
rep N data = [ar2]          with activate afifo+data;
rep N data = [ar3]          with activate afifo-data;
rep N [ar6++gr6] = afifo;
```

### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
flcr	задает функцию активации
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
ar1	указатель на 64р. слово-константу
ar2	указатель на 64р. слово-константу
ar3	указатель на 64р. слово-константу
gr5	Высота матриц $N = [0,1,2...31,32,33,...]$
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

Restrictions:

При выходе из функции изменяется содержимое регистров: `ar0,ar6`.

`nmppsClipCC_`

## 7.238 Vec\_ClipMul2D2W8\_AddVr

## Функции

- void vec\_ClipMul2D2W8\_AddVr (nmreg nb1, nmreg sb, nmreg flcr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.238.1 Подробное описание

Взвешенное умножение двух массивов с накоплением и активацией

Ядро функции nmppsClipPowC\_RShift\_Convert\_AddC\_().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 8 wfifo=[ar4++],ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,vr;
rep 8 wfifo=[ar4++],ftw,wtw;
rep N data =[ar1++gr1] with vsum ,activate data,afifo;
rep N [ar6++gr6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки
sb	задает разбиение на 8 строк
flcr	задает функцию активации
vr	константа для суммирования
ar0	SrcMtr1
gr0	SrcMtr1 stride
ar1	SrcMtr2
gr1	SrcMtr2 stride
ar4	2 матрицы весовых коэффициентов по 8 64р. слов
gr4	дублирует nb1
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar6,gr7.

## 7.239 Vec\_ClipMulNDNW2\_AddVr

### Функции

- void vec\_ClipMulNDNW2\_AddVr (nmreg nb1, nmreg sb, nmreg flcr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.239.1 Подробное описание

Взвешенное умножение нескольких массивов с накоплением и активацией

Ядро функции SIG\_Filter().

Действие функции эквивалентно следующим псевдоинструкциям:

```
ar2=ar0;

gr2=[ar1++];
ar0=ar2+gr2;
rep 2 wfifo=[ar4++].ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,vr;

.repeat K-1;
gr2=[ar1++];
ar0=ar2+gr2;
rep 2 wfifo=[ar4++].ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,afifo;
.endrepeat;
rep N [ar6++gr6]=afifo;
```

### Аргументы

nb1	задает разбиение на колонки
sb	задает разбиение на 2 строки (sb=2)
flcr	задает функцию активации
vr	константа для суммирования
ar0	задает базовый адрес для входных массивов (как правило адрес первого массива)
gr0	шаг чтения входного массива stride for input arrays
ar1	массив адресных смещений входных массивов относительно ar0
gr1	количество массив - K
ar4	массив из K матриц весовых коэффициентов по 2 64р. слов
gr4	дублирует nb1
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,gr1,ar2,gr2,ar3,gr3,ar4,ar6,gr7.

## 7.240 Vec\_ClipMulNDNW4\_AddVr

## Функции

- void vec\_ClipMulNDNW4\_AddVr (nmreg nb1, nmreg sb, nmreg flcr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.240.1 Подробное описание

Взвешенное умножение нескольких массивов с накоплением и активацией

Ядро функции SIG\_Filter().

Действие функции эквивалентно следующим псевдоинструкциям:

```
ar2=ar0;

gr2=[ar1++];
ar0=ar2+gr2;
rep 4 wfifo=[ar4++].ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,vr;

.repeat K-1;
gr2=[ar1++];
ar0=ar2+gr2;
rep 4 wfifo=[ar4++].ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,afifo;
.endrepeat;
rep N [ar6++gr6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки
sb	задает разбиение на 4 строки (sb=20002h)
flcr	задает функцию активации
vr	константа для суммирования
ar0	задает базовый адрес для входных массивов (как правило адрес первого массива)
gr0	stride for input arrays
ar1	массив адресных смещений входных массивов относительно ar0
gr1	количество массив - K
ar4	массив из K матриц весовых коэффициентов по 4 64р. слов
gr4	дублирует nb1
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,gr1,ar2,gr2,ar3,gr3,ar4,ar6,gr7.

## 7.241 Vec\_ClipMulNDNW8\_AddVr

### Функции

- void vec\_ClipMulNDNW8\_AddVr (nmreg nb1, nmreg sb, nmreg flcr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.241.1 Подробное описание

Взвешенное умножение нескольких массивов с накоплением и активацией

Ядро функции SIG\_Filter().

Действие функции эквивалентно следующим псевдоинструкциям:

```
ar2=ar0;

gr2=[ar1++];
ar0=ar2+gr2;
rep 8 wfifo=[ar4++].ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,vr;

.repeat K-1;
gr2=[ar1++];
ar0=ar2+gr2;
rep 8 wfifo=[ar4++].ftw,wtw;
rep N data =[ar0++gr0] with vsum ,activate data,affo;
.endrepeat;
rep N [ar6++gr6]=affo;
```

### Аргументы

nb1	задает разбиение на колонки
sb	задает разбиение на 8 строки (sb=2020202h)
flcr	задает функцию активации
vr	константа для суммирования
ar0	\ задает базовый адрес для входных массивов (как правило адрес первого массива) \ set base address for input arrays
gr0	stride for input arrays
ar1	массив адресных смещений входных массивов относительно ar0
gr1	\ количество массив - K \ number of arrays - K
ar4	массив из K матриц весовых коэффициентов по 8 64р. слов
gr4	дублирует nb1
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на результирующий столбец
gr6	межстрочный шаг для ar6

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,gr1,ar2,gr2,ar3,gr3,ar4,ar6,gr7.



## 7.242 Vec\_IncNeg

## Функции

- void vec\_IncNeg (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.242.1 Подробное описание

Увеличивает отрицательные числа на 1.

Применяется в nmppsDivC().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram = [ar0++gr0] with activate data;
rep N          with ram - aifo;
rep N [ar6++gr6] = aifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
flcr	задает функцию активации
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.243 Vec\_Mul2D2W1\_AddVr

### Функции

- void vec\_Mul2D2W1\_AddVr (nmreg nb1, nmreg sb, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.243.1 Подробное описание

Ядро функции nmppsConvert\_64s(nm64s\* ,nm32s\* ,int ).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 1 wfifo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 1 wfifo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep N [ar6++gr6]=afifo;
```

### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение в 1 строку
vr	константа для суммирования
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar4	указатель на 2 матрицы весовых коэффициентов по 2 строки в каждой
gr4	дублирует nb1
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar6,gr7.

## 7.244 Vec\_Mul2D2W2\_AddVr

## Функции

- void vec\_Mul2D2W2\_AddVr (nmreg nb1, nmreg sb, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.244.1 Подробное описание

Ядро функции nmppsConvert\_32s(nm32s\* ,nm16s\* ,int ).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 2 wffo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 2 wffo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep N [ar6++gr6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 2 строки
vr	константа для суммирования
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar4	указатель на 2 матрицы весовых коэффициентов по 2 строки в каждой
gr4	дублирует nb1
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar6,gr7.

## 7.245 Vec\_Mul2D2W4\_AddVr

### Функции

- void vec\_Mul2D2W4\_AddVr (nmreg nb1, nmreg sb, nmreg flcr, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.245.1 Подробное описание

Ядро функции nmppsConvert\_16s(nm16s\* pSrcVec, nm8s\* pDstVec, int nSize).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 4 wffo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 4 wffo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep N [ar6++gr6]=afifo;
```

### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 4 строки
flcr	задает функцию активации
vr	константа для суммирования
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar4	указатель на 2 матрицы весовых коэффициентов по 4 строки в каждой
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar6,gr7.

## 7.246 Vec\_Mul2D2W8\_AddVr

## Функции

- void vec\_Mul2D2W8\_AddVr (nmreg nb1, nmreg sb, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.246.1 Подробное описание

Применяется в MTR\_Cоруua().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 8 wffo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 8 wffo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,affo;
rep N [ar6++gr6]=affo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
vr	константа для суммирования
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar4	указатель на 2 матрицы весовых коэффициентов по 8 строк в каждой
gr4	дублирует nb1
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar6,gr7.

## 7.247 Vec\_Mul3D3W2\_AddVr

## Функции

- void vec\_Mul3D3W2\_AddVr (nmreg nb1, nmreg sb, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar2, nmreg gr2, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.247.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar2++gr2] with vsum ,data,afifo;
rep N [ar6++gr6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 2 строки
vr	константа для суммирования
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar2	указатель на столбец SrcMtr3
gr2	SrcMtr3 stride
ar4	указатель на 3 матрицы весовых коэффициентов по 2 строки в каждой
gr4	дублирует nb1
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar2,ar6,gr7.

## 7.248 Vec\_Mul3D3W8\_AddVr

## Функции

- void vec\_Mul3D3W8\_AddVr (nmreg nb1, nmreg sb, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar2, nmreg gr2, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.248.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 8 wfifo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 8 wfifo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep 8 wfifo=[ar4++],ftw,wtw;
rep N data=[ar2++gr2] with vsum ,data,afifo;
rep N [ar6++gr6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
vr	константа для суммирования
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar2	указатель на столбец SrcMtr3
gr2	SrcMtr3 stride
ar4	указатель на 3 матрицы весовых коэффициентов по 8 строк в каждой
gr4	дублирует nb1
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar2,ar6,gr7.

## 7.249 Vec\_Mul4D4W2\_AddVr

## Функции

- void vec\_Mul4D4W2\_AddVr (nmreg nb1, nmreg sb, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar2, nmreg gr2, nmreg ar3, nmreg gr3, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.249.1 Подробное описание

Ядро функции nmppsConvert\_32s(nm32s\* pSrcVec, nm8s\* pDstVec, int nSize).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar0++gr0] with vsum ,data,vr;
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar2++gr2] with vsum ,data,afifo;
rep 2 wfifo=[ar4++],ftw,wtw;
rep N data=[ar3++gr3] with vsum ,data,afifo;
rep N [ar6++gr6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 2 строки
vr	константа для суммирования
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar2	указатель на столбец SrcMtr3
gr2	SrcMtr3 stride
ar3	указатель на столбец SrcMtr4
gr3	SrcMtr4 stride
ar4	указатель на 4 матрицы весовых коэффициентов по 2 строки в каждой
gr4	дублирует nb1
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar2,ar3,ar6,gr7.



## 7.250 Vec\_MulVN\_AddVN

## Функции

- void vec\_MulVN\_AddVN (nmreg nb1, nmreg sb, nmreg flcr, nmreg woper, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.250.1 Подробное описание

Ядро функции MTR\_MulC\_AddVsVc( ).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with vsum ,data, vr;
rep N data=[ar1++gr1] with affo+data;
rep N [ar6++gr6]=affo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
flcr	задает функцию активации
woper	в рабочей матрице должны быть загружены весовые коэффициенты
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
gr5	Высота матриц $N = [0,1,2...31,32,33,...]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar6.

## 7.251 Vec\_Sub

### Функции

- void vec\_Sub (nmreg nb1, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.251.1 Подробное описание

Ядро функции nmppsSub().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data;
rep N data=[ar1++gr1] with data - affo;
rep N [ar6++gr6]=affo;
```

### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar6.

## 7.252 Vec\_SubAbs

## Функции

- void vec\_SubAbs (nmreg nb1, nmreg sb, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.252.1 Подробное описание

Ядро функции nmppsAbsDiff().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with data
rep N data=[ar1++gr1] with afifo-data;
rep N [ar4],ram =afifo with activate afifo;
rep N with vsum afifo,ram,ram;
rep N [ar6++gr6]= afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
flcr	задает функцию активации
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar4	указатель на временный буфер (1 64р. слово)
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.253 Vec\_SubVN\_Abs

### Функции

- void vec\_SubVN\_Abs (nmreg nb1, nmreg sb, nmreg flcr, nmreg woper, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.253.1 Подробное описание

Ядро функции mtr\_SubMV\_Abs().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar1]
rep N data=[ar0++gr0] with data-ram;
rep N [ar2++]=afifo with activate afifo;
// ar2,ar5- internal pointers to temporary buffer size of long[32]
rep N data=[ar5++] with vsum afifo,data,data;
rep N [ar6++gr6]=afifo;
```

### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
flcr	задает функцию активации
woper	в рабочей матрице должны быть загружены весовые коэффициенты
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
ar1	указатель на маску (1 64p. слово)
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.254 Vec\_Swap

## Функции

- void vec\_Swap (nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.254.1 Подробное описание

Ядро функции mtr\_SubVN\_Abs().

Функция осуществляет два одновременных копирования:

[ar0++gr0] => [ar4++gr4]

[ar1++gr1] => [ar6++gr6]

если ar6=ar0,gr6=gr0, ar4=ar1,gr4=gr1

то выполняется перестановка двух векторов

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar0++gr0];
rep N data=[ar1++gr1] with data;
rep N [ar6++gr6]=afifo with ram;
rep N [ar4++gr4]=afifo;
```

## Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar4	указатель на столбец DstMtr1
gr4	DstVec1 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr2
gr6	DstVec2 stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar4,ar6.

## 7.255 Vec\_MUL\_2V4toW8\_shift

### Функции

- void vec\_MUL\_2V4toW8\_shift (nmreg nb1, nmreg sb, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar3, nmreg gr4, nmreg ar5, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.255.1 Подробное описание

Ядро функции SIG\_ResizeDown2(nm8u7b\* pSrcVec, nm8u7b\* pDstVec, int nSize).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar3];
rep 8 wifo=[ar5++],ftw,wtw;
rep N data =[ar0++gr0],ftw,wtw with vsum ,data,0;
rep N data =[ar1++gr1]          with vsum ,data,afifo;
rep N          with mask ram,shift afifo,0;
rep N [ar6++gr6]=afifo;
```

### Аргументы

nb1	задает разбиение на колонки
sb	задает разбиение на 4 строки
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar3	указатель на 64р. маску
gr4	дублирует nb1
ar5	указатель на матрицу весовых коэффициентов (16 64р.слов)
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar2,ar4,ar6,gr2,gr5.

## 7.256 Vec\_MUL\_2V8toW16\_shift

## Функции

- void vec\_MUL\_2V8toW16\_shift (nmreg nb1, nmreg sb, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar3, nmreg gr4, nmreg ar5, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.256.1 Подробное описание

Ядро функции SIG\_ResizeDown2(nm16u15b\* pSrcVec, nm16u15b\* pDstVec, int nSize).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar3];
rep 16 wifo=[ar5++],ftw,wtw;
rep N data=[ar0++gr0],ftw,wtw with vsum ,data,0;
rep N data=[ar1++gr1] with vsum ,data,afifo;
rep N with mask ram,shift afifo,0;
rep N [ar6++gr6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки
sb	задает разбиение на 2 строки
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
ar3	указатель на 64р. маску
gr4	дублирует nb1
ar5	указатель на матрицу весовых коэффициентов (16 64р.слов)
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar2,ar4,ar6,gr2,gr5.

## 7.257 Vec\_not\_data

## Функции

- void vec\_not\_data (nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.257.1 Подробное описание

Ядро функции nmppsNot\_ ( ).

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with not data;
rep N [ar6++gr6]=aiffo;
```

## Аргументы

ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar6.



## 7.258 Vec\_ram

### Функции

- void vec\_ram (nmreg ar0, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.258.1 Подробное описание

Ядро функции nmppsSet\_().

Функция служит для заполнения массива константой.

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar0] with data;  
rep N [ar6++gr6]=aifo;
```

### Аргументы

ar0	указатель на столбец SrcMtr1
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.259 Vec\_ram\_sub\_data

### Функции

- void vec\_ram\_sub\_data (nmreg nb1, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.259.1 Подробное описание

Ядро функции nmppsSubCRev().

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar1];
rep N data=[ar0++gr0] with ram-data;
rep N [ar6++gr6]=afifo;
```

### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.260 Vec\_vsum\_activate\_data\_0

## Функции

- void vec\_vsum\_activate\_data\_0 (nmreg nb1, nmreg sb, nmreg flcr, nmreg woper, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.260.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with vsum ,activate,0;
rep N [ar6++gr6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
flcr	задает функцию активации
woper	в рабочей матрице должны быть загружены весовые коэффициенты
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1	указатель на столбец SrcMtr2
gr1	SrcMtr2 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.261 Vec\_vsum\_data\_0

## Функции

- void vec\_vsum\_data\_0 (nmreg nb1, nmreg sb, nmreg woper, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.261.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with vsum ,data,0;
rep N [ar6++gr6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
woper	в рабочей матрице должны быть загружены весовые коэффициенты
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
gr5	Высота матриц $N = [0,1,2...31,32,33,...]$
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.262 Vec\_vsum\_data\_afifo

## Функции

- void vec\_vsum\_data\_afifo (nmreg nb1, nmreg sb, nmreg woper, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6)

## 7.262.1 Подробное описание

Используется в nmppsSum(nm1\* pSrcVec, void\* pTmpBuf, int nSize)

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 1  data=vfalse;
.repeat N;
rep 1  data=[ar0++gr0] with vsum ,data,afifo;
.endrepeat;
rep 1  [ar6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на строки
woper	в рабочей матрице должны быть загружены весовые коэффициенты
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0.

## 7.263 Vec\_vsum\_data\_vr

### Функции

- void vec\_vsum\_data\_vr (nmreg nb1, nmreg sb, nmreg woper, nmreg vr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.263.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with vsum ,data,vr;
rep N [ar6++gr6]=afifo;
```

### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
flcr	задает функцию активации
woper	в рабочей матрице должны быть загружены весовые коэффициенты
vr	константа для суммирования
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.264 Vec\_vsum\_shift\_data\_0

## Функции

- void vec\_vsum\_shift\_data\_0 (nmreg nb1, nmreg sb, nmreg woper, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.264.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with vsum ,shift data,0;
rep N [ar6++gr6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
woper	в рабочей матрице должны быть загружены весовые коэффициенты
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.

## 7.265 Vec\_vsum\_shift\_data\_vr

## Функции

- void vec\_vsum\_shift\_data\_vr (nmreg nb1, nmreg sb, nmreg woper, nmreg vr, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6, nmreg gr6)

## 7.265.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N data=[ar0++gr0] with vsum ,shift data, vr;
rep N [ar6++gr6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
woper	в рабочей матрице должны быть загружены весовые коэффициенты
vr	константа для суммирования
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar6.



## 7.266 Vec\_vsum\_shift\_data\_afifo

## Функции

- void vec\_vsum\_shift\_data\_afifo (nmreg nb1, nmreg sb, nmreg flcr, nmreg woper, nmreg ar0, nmreg gr0, nmreg gr5, nmreg ar6)

## 7.266.1 Подробное описание

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 1  data=vfalse;
rep 1*N data=[ar0++gr0] with vsum , shift data,afifo; (rep1 N times)
rep 1  [ar6]=afifo;
```

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
sb	задает разбиение на 8 строк
flcr	задает функцию активации
woper	в рабочей матрице должны быть загружены весовые коэффициенты
ar0	указатель на столбец SrcMtr
gr0	SrcMtr stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0.

## 7.267 Vec\_CompareMinV

Поэлементный поиск минимального

Действие функции эквивалентно следующим псевдоинструкциям:

Функции

- void vec\_CompareMinV (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar3, nmreg gr3, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.267.1 Подробное описание

Поэлементный поиск минимального

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar0++gr0];
rep N data=[ar1++gr1] with ram-data;
rep N      with activate afifo;
rep N data=[ar3++gr3] with mask afifo, ram, data;
rep N [ar6++gr6]=afifo;
```

\param nb1

задает разбиение на колонки (необходимо wtw)

Аргументы

flcr	задает функцию активации
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1,ar3	указатель на столбец SrcMtr2
gr1,gr3	SrcMtr2 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar3,ar6.

## 7.268 Vec\_CompareMaxV

Поэлементный поиск максимального

Действие функции эквивалентно следующим псевдоинструкциям:

Функции

- void vec\_CompareMaxV (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar1, nmreg gr1, nmreg ar3, nmreg gr3, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.268.1 Подробное описание

Поэлементный поиск максимального

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep N ram=[ar0++gr0];
rep N data=[ar1++gr1] with ram-data;
rep N      with activate aifo;
rep N data=[ar3++gr3] with mask aifo, data, ram;
rep N [ar6++gr6]=aifo;
```

\param nb1

задаёт разбиение на колонки (необходимо wtw)

Аргументы

flcr	задаёт функцию активации
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr1 stride
ar1,ar3	указатель на столбец SrcMtr2
gr1,gr3	SrcMtr2 stride
gr5	Высота матриц N = [0,1,2...31,32,33,...]
ar6	указатель на столбец DstMtr
gr6	DstMtr stride

Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,ar1,ar3,ar6.

## 7.269 Vec\_DupValueInVector8

Размножение 8-ми битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

Функции

- void vec\_DupValueInVector8 (nmreg ar1, nmreg gr1)

### 7.269.1 Подробное описание

Размножение 8-ми битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

```
gr1 = gr1 and 0FFh;
[ar1] = gr1 + (gr1 << 8) + (gr1 << 16) + (gr1 << 24) +
          (gr1 << 32) + (gr1 << 40) + (gr1 << 48) + (gr1 << 56).
ar1 += 2;
```

\param ar1

Адрес 64-х битового вектора.

Аргументы

gr1	Значение (8 бит).
-----	-------------------

Restrictions:

При выходе из функции изменяется содержимое регистров: ar1, gr1.

## 7.270 Vec\_DupValueInVector16

Размножение 16-ти битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

Функции

- void vec\_DupValueInVector16 (nmreg ar1, nmreg gr1)

### 7.270.1 Подробное описание

Размножение 16-ти битового значения по всему вектору.

Действие функции эквивалентно следующим псевдоинструкциям:

```
gr1 = gr1 and 0FFFFh;
[ar1] = gr1 + (gr1 << 16) + (gr1 << 32) + (gr1 << 48).
ar1 += 2;
```

\param ar1

Адрес 64-х битового вектора.

Аргументы

gr1	Значение (8 бит).
-----	-------------------

Restrictions:

При выходе из функции изменяется содержимое регистров: ar1, gr1.

## 7.271 Vec\_BuildDiagWeights8

Построение диагональной матрицы весовых коэффициентов (8x8).

Функции

- void vec\_BuildDiagWeights8 (nmreg ar1, nmreg gr1)

### 7.271.1 Подробное описание

Построение диагональной матрицы весовых коэффициентов (8x8).

Аргументы

ar1	Адрес 64-х буфера весовых коэффициентов (8x64 бит).
gr1	Значение (8 бит).

Restrictions:

При выходе из функции изменяется содержимое регистров: ar1, gr1.

## 7.272 Vec\_BuildDiagWeights16

Построение диагональной матрицы весовых коэффициентов (16x16).

Функции

- void vec\_BuildDiagWeights16 (nmreg ar1, nmreg gr1)

### 7.272.1 Подробное описание

Построение диагональной матрицы весовых коэффициентов (16x16).

Аргументы

ar1	Адрес 64-х буфера весовых коэффициентов (4x64 бит).
gr1	Значение (16 бит).

Restrictions:

При выходе из функции изменяется содержимое регистров: ar1, gr1.

## 7.273 Vec\_MaxVal\_v8nm8s

Поиск максимума в 8 байтах

Функции

- void vec\_MaxVal\_v8nm8s ([nmreg](#) ar0, [nmreg](#) gr7)

### 7.273.1 Подробное описание

Поиск максимума в 8 байтах

Аргументы

ar0	Адрес 64р. слова
-----	------------------

Возвращаемые значения

gr7	Максимум из 8 байт
-----	--------------------

Restrictions:

При выходе из функции изменяется содержимое регистров: gr0, gr1, gr2, gr3, ar5, gr5, gr7.



## 7.274 Vec\_MaxVal\_v4nm16s

Поиск максимума в 4-х 16р. элементах

Функции

- void vec\_MaxVal\_v4nm16s ([nmreg](#) ar0, [nmreg](#) gr7)

### 7.274.1 Подробное описание

Поиск максимума в 4-х 16р. элементах

Аргументы

ar0	Адрес 64р. слова
-----	------------------

Возвращаемые значения

gr7	Максимум из 8 байт
-----	--------------------

Restrictions:

При выходе из функции изменяется содержимое регистров: gr0, gr1, gr2, gr3, ar5, gr5, gr7.

## 7.275 Vec\_MaxVal

Поиск максимумов в колонках матрицы SrcMtr1.

### Функции

- void vec\_MaxVal (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar4, nmreg gr5, nmreg ar6)

### 7.275.1 Подробное описание

Поиск максимумов в колонках матрицы SrcMtr1.

#### Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
flcr	задает функцию активации

\param ar0

указатель на столбец SrcMtr1

#### Аргументы

gr0	SrcMtr stride
ar4	указатель на временный массив размером nm64s[64]
gr5	Высота матрицы SrcMtr1 N = [32,64,...]

#### Возвращаемые значения

ar6	указатель на 64р. слово результатов (максимумов)
-----	--

#### Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,gr0,ar4,ar3,ar6,gr7.

## 7.276 Vec\_MinVal\_v8nm8s

Поиск минимума в 8 байтах

Функции

- void vec\_MinVal\_v8nm8s ([nmreg](#) ar0, [nmreg](#) gr7)

### 7.276.1 Подробное описание

Поиск минимума в 8 байтах

Аргументы

ar0	Адрес 64р. слова
-----	------------------

Возвращаемые значения

gr7	Максимум из 8 байт
-----	--------------------

Restrictions:

При выходе из функции изменяется содержимое регистров: gr0, gr1, gr2, gr3, ar5, gr5, gr7.

## 7.277 Vec\_MinVal\_v4nm16s

Поиск минимума в 4-х 16р. элементах

Функции

- void vec\_MinVal\_v4nm16s ([nmreg](#) ar0, [nmreg](#) gr7)

### 7.277.1 Подробное описание

Поиск минимума в 4-х 16р. элементах

Аргументы

ar0	Адрес 64р. слова
-----	------------------

Возвращаемые значения

gr7	Максимум из 8 байт
-----	--------------------

Restrictions:

При выходе из функции изменяется содержимое регистров: gr0, gr1, gr2, gr3, ar5, gr5, gr7.

## 7.278 Vec\_MinVal

Поиск минимумов в колонках матрицы SrcMtr1.

## Функции

- void vec\_MinVal (nmreg nb1, nmreg flcr, nmreg ar0, nmreg gr0, nmreg ar4, nmreg gr5, nmreg ar6)

## 7.278.1 Подробное описание

Поиск минимумов в колонках матрицы SrcMtr1.

## Аргументы

nb1	задает разбиение на колонки (необходимо wtw)
flcr	задает функцию активации
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr stride
ar4	указатель на временный массив размером nm64s[64]
gr5	Высота матрицы SrcMtr1 N = [32,64,...]

## Возвращаемые значения

ar6	указатель на 64р. слово результатов (максимумов)
-----	--

## Restrictions:

При выходе из функции изменяется содержимое регистров: ar0,gr0,ar4,ar3,ar6,gr7.

## 7.279 Vec\_AccMul1D1W32\_AddVr

Умножение с накоплением

Действие функции эквивалентно следующим псевдоинструкциям:

Функции

- void vec\_AccMul1D1W32\_AddVr (nmreg nb1, nmreg sb, nmreg vr, nmreg ar0, nmreg gr0, nmreg ar4, nmreg gr4, nmreg gr5, nmreg ar6, nmreg gr6)

### 7.279.1 Подробное описание

Умножение с накоплением

Действие функции эквивалентно следующим псевдоинструкциям:

```
rep 32 wfifo=[ar4++],ftw,wtw;
rep 32 wfifo data = [ar0++gr0] with vsum ,data,vr;
with gr5--;
with gr5--;
<Loop>
    rep 32 wfifo=[ar4++],ftw,wtw;
    rep 32 data=[ar0++gr0] with vsum ,data,afifo;
if <>0 goto Loop with gr5--;
rep 32 [ar6++gr6]=afifo;
```

\param nb1

задает разбиение на колонки (необходимо wtw)

Аргументы

sb	задает разбиение на 32 строки
vr	константа для суммирования
ar0	указатель на столбец SrcMtr1
gr0	SrcMtr stride
ar4	матрицы весовых коэффициентов
gr4	дублирует nb1
gr5	кол-во итераций умножений с накоплением

Возвращаемые значения

ar6	указатель на столбец DstMtr, состоящий из 32 длинных слов
-----	---

Аргументы

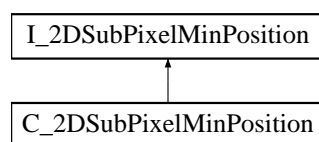
gr6	DstMtr stride
-----	---------------

## Глава 8

# Классы

### 8.1 Класс C\_2DSubPixelMinPosition

Граф наследования: C\_2DSubPixelMinPosition:



Открытые члены

- virtual void Find (float \*S9, float &dx, float &dy)
- virtual void Release ()

#### 8.1.1 Подробное описание

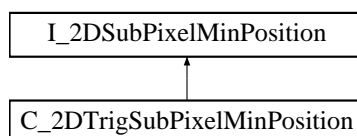
См. определение в файле isubpixel2dimpl.h строка 7

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/isubpixel2dimpl.h

### 8.2 Класс C\_2DTrigSubPixelMinPosition

Граф наследования: C\_2DTrigSubPixelMinPosition:



### Открытые члены

- virtual void Find (float \*S9, float &dx, float &dy)
- virtual void Release ()

### Защищенные члены

- float S9Interpolation (float x, float y, float \*S9)

### Защищенные данные

- float Teta [8]

#### 8.2.1 Подробное описание

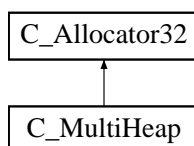
См. определение в файле isubpixel2dimpl.h строка 20

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/isubpixel2dimpl.h

### 8.3 Класс C\_Allocator32

Граф наследования: C\_Allocator32:



### Открытые члены

- void \* Allocate ()
- int Release ()

#### 8.3.1 Подробное описание

См. определение в файле multiheap.h строка 330

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/multiheap.h



## 8.4 Шаблон класса C\_BoxImg< T >

### Открытые члены

- C\_BoxImg (C\_MultiHeap &MultiHeap, int Width, int Height, int BorderHeight=0, int FillMode=BOX\_IMG\_FILL\_FF)
- void Lock ()
- void Unlock ()
- void Fill (int FillMode)
- C\_BoxImg (T \*Data, int Width, int Height, int BorderHeight=0)
- void Init (T \*Data, int Width, int Height, int BorderHeight=0)
- T \* Addr (int y, int x)
- T \* Allocate (C\_MultiHeap &MultiHeap, int Width, int Height, int BorderHeight=0)
- T \* Allocate (C\_MultiHeap &MultiHeap)
- int Release ()
- \_\_INLINE\_\_ T \* operator[] (int idx)

### Открытые атрибуты

- int nWidth
- int nHeight
- int sizeBox
- int sizeData
- int nBorder
- T \* pBox
- T \* pData
- C\_MultiHeap \* pHeap

#### 8.4.1 Подробное описание

```
template<class T>
class C_BoxImg< T >
```

См. определение в файле multiheap.h строка 717

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/multiheap.h

## 8.5 Шаблон класса C\_BoxVec< T >

### Открытые члены

- C\_BoxVec (T \*Data, int SizeData, int SizeBorder=0)
- C\_BoxVec (C\_MultiHeap &MultiHeap, int SizeData, int Border=0)
- T \* Addr (int idx)
- int Assign (T \*Data, int SizeData, int SizeBorder=0)
- T \* Allocate (C\_MultiHeap &MultiHeap, int SizeData, int Border=0)
- T \* Allocate (C\_MultiHeap &MultiHeap)
- int Release ()

## Открытые атрибуты

- int sizeData
- int sizeBox
- int nBorder
- T \* pBox
- T \* pData
- [C\\_MultiHeap](#) \* pHeap

## 8.5.1 Подробное описание

```
template<class T>
class C_BoxVec< T >
```

См. определение в файле multiheap.h строка 634

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/multiheap.h

## 8.6 Класс C\_Heap

класс - куча

```
#include <multiheap.h>
```

## Открытые члены

- [C\\_Heap](#) (void \*addrHeap, size\_t size32Heap)  
конструктор - создает кучу в указанной памяти
- void [Create](#) (void \*addrHeap, size\_t size32Heap)  
создает кучу в указанной памяти
- int [IsMine](#) (void \*addr)  
устанавливает принадлежность к куче
- size\_t [AllocateMaxAvail](#) ()  
Возвращает объем свободной памяти в пуле в 32-х словах
- int \* [Allocate](#) (size\_t size32Buffer)  
Выделяет буфер в куче
- int [ReleaseBuffer](#) ([S\\_BufferInfo](#) \*pDelBuffer)  
удаляет структуру буфера из списка
- int [Release](#) (void \*p)  
освобождает память по адресу
- void [Lock](#) (void \*p)  
блокирует указатель от удаления через Release.
- void [LockAll](#) ()  
блокирует все указатели от удаления через Release.
- void [UnlockAll](#) ()  
разблокирует все указатели для удаления через Release.
- void [Unlock](#) (void \*p)  
разблокирует все указатели для удаления через Release.
- void [ReleaseAll](#) ()  
удаляет все указатели из кучи
- void [LockHeap](#) ()  
Запрещает операции с кучей
- void [UnlockHeap](#) ()  
Разрешает операции с кучей
- int [Check](#) ()

## Открытые атрибуты

- [S\\_BufferInfo \\* pZeroBuffer](#)  
< указатель на нулевой буфер в списке (с нулевым размером)
- [int \\* pHeapEnd](#)  
< указатель на слово следующее за концом кучи
- [int size32HeapAvail](#)  
< размер общей свободной памяти в куче
- [bool isHeapLocked](#)  
< запрещает операции с кучей
- [int status](#)

## 8.6.1 Подробное описание

класс - куча

См. определение в файле multiheap.h строка 64

## 8.6.2 Методы

## 8.6.2.1 AllocateMaxAvail()

```
size_t32 C_Heap::AllocateMaxAvail ( ) [inline]
```

Возвращает объем свободной памяти в пуле в 32р. словах

Возвращает максимальный размер буфера в 32р. словах, который можно выделить в куче

См. определение в файле multiheap.h строка 115

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/multiheap.h

## 8.7 Шаблон класса C\_Img&lt; T &gt;

## Открытые члены

- [C\\_Img \(int nWidth, int nHeight, int nStride, int nBorder, void \\*\(\\*allocator32\)\(int\)\)](#)
- [C\\_Img \(T \\*pData, int nWidth, int nHeight, int nStride, int nBorder\)](#)
- [void Fill \(T color\)](#)

## Открытые атрибуты

- int m\_nBorder
- T \* m\_pContainer
- T \* m\_pData
- int m\_nWidth
- int m\_nHeight
- int m\_nStride
- int m\_nSize

## 8.7.1 Подробное описание

```
template<class T>
class C_Img< T >
```

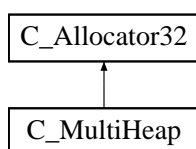
См. определение в файле iSupport.h строка 37

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/iSupport.h

## 8.8 Класс C\_MultiHeap

Граф наследования: C\_MultiHeap:



## Открытые члены

- C\_MultiHeap (int Mode=ALLOCATE\_FORWARD)
- int Error ()
- void Mode (int mode, void \*\*legend=0)
- unsigned Rand ()
  - Генератор случайных чисел
- unsigned Rand (unsigned min, unsigned max)
- C\_Heap & operator[] (int idxHeap)
- int CreateHeap (void \*addrHeap, size\_t size32Heap)
  - создает кучу по адресу указанного размера (полный размер со служебными данными)
- void \* Allocate (size\_t size32Buffer)
  - обходит кучи в заданном в AllocateMode порядке и выделяет память заданного размера
- void \* Allocate (size\_t size32Buffer, int nPriorHeap0, int nPriorHeap1=-1, int nPriorHeap2=-1, int nPriorHeap3=-1, int nPriorHeap4=-1, int nPriorHeap5=-1)
  - обходит кучи в заданном порядке и выделяет память заданного размера
- void \* AllocateWith (size\_t size32Buffer, void \*addrInTheSameHeap)
  - выделяет массив в той же куче где и указатель

- int Which (void \*addr)  
Возвращает номер кучи к которой принадлежит адрес
- void Lock (void \*addr)
- int Unlock (void \*addr)
- int LockAll ()
- int UnlockAll ()
- int Release (void \*addr)
- void ReleaseAll ()  
Удвояет все назаблокированные указатели из куч
- void LockHeap (int idxHeap)  
Запрещает операции Allocate и Release с кучей
- void UnlockHeap (int idxHeap)  
Разрешает операции Allocate и Release с кучей
- int Check ()

### Открытые атрибуты

- C\_Heap pHeap [MAX\_NUM\_BANKS]  
массив куч
- unsigned numHeaps  
число проинициализированных куч
- unsigned numAllocateFails  
число ошибок выделения куч
- unsigned AllocateMode  
порядок обхода куч при поиске свободного места
- void \*\* pAllocateLegend  
история номеров куч использованных в последних 8 Allocate.
- unsigned idxAllocateLegend
- long long allocateHistory

#### 8.8.1 Подробное описание

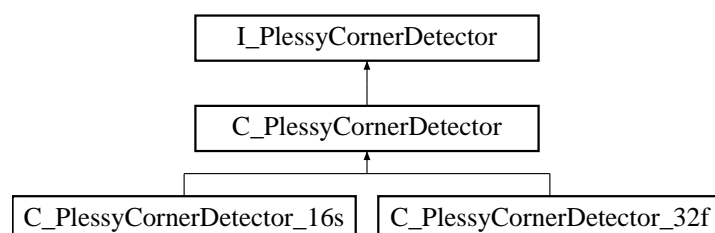
См. определение в файле multiheap.h строка 336

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/multiheap.h

## 8.9 Класс C\_PlessyCornerDetector

Граф наследования: C\_PlessyCornerDetector:



## Открытые члены

- virtual void Allocate (int w, int h, int ww)
- virtual void DeAllocate ()
- virtual void FindCorners (unsigned char \*Picture, int w, int h, int ww, float \*px, float \*py, int &nc)
- virtual void Release ()
- virtual void SetThreshold (float \_threshold)=0

## Защищенные члены

- virtual void CountDer (unsigned char \*Picture, int w, int h, int ww)
- virtual void CountPlessy (int w, int h, int ww)

## Защищенные данные

- [I\\_2DSubPixelMinPosition](#) \* SubPixelMinPosition
- short \* fxi
- short \* fyi
- short \* Picture16
- float \* SumSxxSyy
- float \* Subxy
- float S9 [9]
- float threshold
- unsigned char \* cres

## 8.9.1 Подробное описание

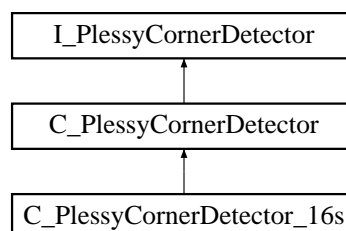
См. определение в файле iPlessyDetector.h строка 8

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/iPlessyDetector.h

## 8.10 Класс C\_PlessyCornerDetector\_16s

Граф наследования:C\_PlessyCornerDetector\_16s:



## Открытые члены

- virtual void Allocate (int w, int h, int ww)
- virtual void DeAllocate ()
- virtual void SetThreshold (float \_threshold)

## Защищенные члены

- virtual void CountPlessy (int w, int h, int ww)

## Защищенные данные

- short \* sxxi
- short \* sxyi
- short \* syyi
- short \* Sxxi
- short \* Sxyi
- short \* Syyi
- short \* Sxxit
- short \* Syyit
- short \* SumSxxSyyi
- short \* MulSxxSyyi
- short \* MulSxySxyi
- short \* Subxyi
- short \* cSubxyi
- short threshold

## 8.10.1 Подробное описание

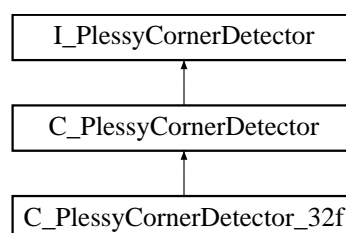
См. определение в файле iPlessyDetector.h строка 56

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/iPlessyDetector.h

## 8.11 Класс C\_PlessyCornerDetector\_32f

Граф наследования: C\_PlessyCornerDetector\_32f:



## Открытые члены

- virtual void Allocate (int w, int h, int ww)
- virtual void DeAllocate ()
- virtual void SetThreshold (float \_threshold)

## Защищенные члены

- virtual void CountPlessy (int w, int h, int ww)
- virtual void CountDer (unsigned char \*Picture, int w, int h, int ww)

## Защищенные данные

- float \* fx
- float \* fy
- float \* sxx
- float \* sxy
- float \* syx
- float \* Sxx
- float \* Sxy
- float \* Syy
- float \* MulSxxSyy
- float \* MulSxySxy
- float \* cSubxy
- float threshold

## 8.11.1 Подробное описание

См. определение в файле iPlessyDetector.h строка 34

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/iPlessyDetector.h

## 8.12 Шаблон класса C\_RingBufferRemote&lt; T &gt;

## Открытые члены

- C\_RingBufferRemote (size\_t ringbuffer\_addr, t\_bytecpy push\_memcpy, t\_bytecpy pop\_memcpy)
- size\_t GetHead ()
- size\_t GetTail ()
- void SetHead ()
- void SetTail ()
- bool Init (size\_t ringbuffer\_addr, t\_bytecpy push\_memcpy, t\_bytecpy pop\_memcpy)
- \_\_INLINE\_\_ bool IsFull ()
- \_\_INLINE\_\_ bool IsEmpty ()
- \_\_INLINE\_\_ size\_t GetWriteAvail ()
- \_\_INLINE\_\_ size\_t GetReadAvail ()
- bool Push (int numElements)
- bool Pop (int numElements)
- size\_t Push (T \*pSrcElements, size\_t numElements, int ExitMode=EXIT\_ON\_COMPLETED)
- size\_t Pop (T \*pDstElements, size\_t numElements, int ExitMode=EXIT\_ON\_COMPLETED)
- size\_t View (T \*pDstElements, size\_t numElements, int ExitMode=EXIT\_ON\_COMPLETED)



## Открытые атрибуты

- `size_t data_addr`  
физический адрес кольцевого буфера входных данных
- `size_t head_addr`  
сколько элементов ОТ НАЧАЛА ПОТОКА код MASTER уже записал в буфер входных данных [заполняется MASTER].
- `size_t tail_addr`  
сколько элементов ОТ НАЧАЛА ПОТОКА код SLAVE уже прочитал (обработал) [заполняется SLAVE].
- `size_t size`  
размер кольцевого буфера входных данных (в элементах; гарантируется что это степень двойки)
- `size_t head`  
сколько элементов ОТ НАЧАЛА ПОТОКА код MASTER уже записал в буфер входных данных [заполняется MASTER].
- `size_t tail`  
сколько элементов ОТ НАЧАЛА ПОТОКА код SLAVE уже прочитал (обработал) [заполняется SLAVE].
- `size_t id`  
сколько элементов ОТ НАЧАЛА ПОТОКА код SLAVE уже прочитал (обработал) [заполняется SLAVE].
- `bool isConnected`
- `t_bytecpy push_memcpy`
- `t_bytecpy pop_memcpy`
- `t_memcpy dma_init`
- `size_t(* dma_check )()`
- `T * dma_ptr`
- `size_t dma_left`
- `size_t dma_size`
- `unsigned timeout`
- `unsigned time2sleep`
- `int pad [16-3-5 *sizeof(t_bytecpy)/sizeof(int)]`  
резервные поля

## 8.12.1 Подробное описание

```
template<class T>
class C_RingBufferRemote< T >
```

См. определение в файле ringremote.h строка 39

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/ringremote.h

## 8.13 Шаблон класса C\_WarpImg&lt; T &gt;

## Открытые члены

- `C_WarpImg (unsigned width, unsigned height, unsigned border, void *(*malloc32)(unsigned), void(*free32)(void *))`
- `C_WarpImg (unsigned width, unsigned height, unsigned border, void *buffer, int mode=IMG_↔ AT_BUFFER)`
- `T * addr (int x, int y)`
- `T * end ()`

## Открытые атрибуты

- unsigned nWidth
- unsigned nHeight
- unsigned warpHeight
- T \* pWarp
- unsigned nWarpSize
- T \* pImg
- unsigned nImgSize

## 8.13.1 Подробное описание

```
template<class T>
class C_WarpImg< T >
```

См. определение в файле `warpimg.h` строка 5

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/warpimg.h

## 8.14 Шаблон класса CIMG\_FIR&lt; nmbits\_in, nmbits\_out &gt;

```
#include <iFilter.h>
```

## Открытые члены

- **CIMG\_FIR** (int nKerWidth, int nKerHeight, void \*(\*malloc32\_func)(unsigned), void(\*free32\_func)(void \*))  
Конструктор КИХ фильтра
- void \* **SetWeights** (int \*pWeights, int nImgWidth)  
Загружает коэффициенты фильтра и инициализирует внутреннюю структуру хранения коэффициентов в pKernel.
- void **Filter** (nmbits\_in \*pSrcImg, nmbits\_out \*pDstImg, int nImgWidth, int nImgHeight)  
Функция одномерной фильтрации
- **~CIMG\_FIR** ()  
Освобождает динамическую область памяти pKernel.

## Открытые атрибуты

- void(\* **pfFree32** )(void \*)  
Указатель на функции освобождения памяти (pKernel)
- int **nKerWidth**  
Ширина окна коэффициентов КИХ фильтра
- int **nKerHeight**  
Высота окна коэффициентов КИХ фильтра
- **nm64s** \* **pKernel**  
Указатель на внутреннюю структуру коэффициентов
- int **nKernelSize**  
Размер памяти необходимый для хранения внутренней структуры коэффициентов

## 8.14.1 Подробное описание

```
template<class nmbits_in, class nmbits_out>
class CIMG_FIR< nmbits_in, nmbits_out >
```

Класс КИХ фильтра

Template Parameters

nmbits_in	Тип указывающий разрядность входного изображения. Допустимые типы : nm8s,nm16s,nm32s,nm64s
nmbits_out	Тип указывающий разрядность выходного изображения. Допустимые типы : nm8s,nm16s,nm32s,nm64s . Разрядность входного вектора не должна превышать разрядности выходного.

Пример

См. определение в файле iFilter.h строка 302

## 8.14.2 Конструктор(ы)

## 8.14.2.1 CIMG\_FIR()

```
template<class nmbits_in , class nmbits_out >
CIMG_FIR< nmbits_in, nmbits_out >::CIMG_FIR (
    int nKerWidth,
    int nKerHeight,
    void (*)(unsigned) malloc32_func,
    void (*)(void *) free32_func )
```

Конструктор КИХ фильтра

Выделяет область памяти под внутреннюю структуру коэффициентов

Аргументы

nKerWidth	Ширина окна фильтра. nKerWidth=[3,5,7,...]
nKerHeight	Высота окна фильтра. nKerHeight=[1,3,5,7,...]
malloc32_func	указатель на функцию выделения динамической памяти 32-разрядными словами.
free32_func	указатель на функцию динамического освобождения памяти

### 8.14.3 Методы

#### 8.14.3.1 Filter()

```
template<class nmbits_in , class nmbits_out >
void CIMG_FIR< nmbits_in, nmbits_out >::Filter (
    nmbits_in * pSrcImg,
    nmbits_out * pDstImg,
    int nImgWidth,
    int nImgHeight )
```

Функция одномерной фильтрации

$$pDstImg[y][x] = \sum_{i=0}^{nKerHeight-1} \sum_{j=0}^{nKerWidth-1} pSrcImg[y+i-nKerHeight/2][x+j-nKerWidth/2] \cdot pWeights[i][j], x =$$

Аргументы

pSrcImg	входное изображение
pDstImg	выходное изображение
nImgWidth	Ширина изображения к которому данный фильтр будет применен. Кратность согласно входному типу.
nImgHeight	Высота изображения (измеряется в пикселях).

Предупреждения

При вычислении первых и последних  $nKerHeight/2$  сторк производится выход за границы входного массива pSrcImg . Для корректного поведения функции необходимо дополнительно резервировать поля размером не менее  $nImgWidth(nKernHeight/2+1)$  нулевых элементов перед началом и в конце массива pSrcImg.

#### 8.14.3.2 SetWeights()

```
template<class nmbits_in , class nmbits_out >
void* CIMG_FIR< nmbits_in, nmbits_out >::SetWeights (
    int * pWeights,
    int nImgWidth )
```

Загружает коэффициенты фильтра и инициализирует внутреннюю структуру хранения коэффициентов в pKernel.

Аргументы

pWeights	коэффициенты фильтра
nImgWidth	Ширина изображения к которому данный фильтр будет применен. Кратность согласно входному типу.

Возвращает

указатель на внутреннюю структуру коэффициентов. 0- Если память под структуру не была выделена.

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/iFilter.h

## 8.15 Структура ds\_struct

Открытые атрибуты

- int nnSpot  
общее число отбракованных пятен (по данному признаку)
- int nnPxl  
суммарное число пикселей в отбракованных пятнах
- int dttSpot  
общее время обработки отбракованных пятен (в тактах процессора)

### 8.15.1 Подробное описание

См. определение в файле iFloodFill.h строка 94

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iFloodFill.h

## 8.16 Класс EnterHardMode

### 8.16.1 Подробное описание

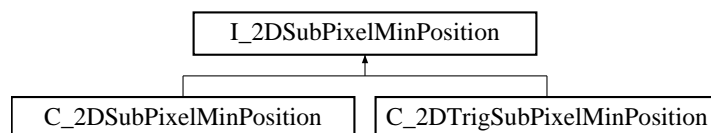
См. определение в файле macros\_fpu.h строка 21

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/macros\_fpu.h

## 8.17 Класс I\_2DSubPixelMinPosition

Граф наследования:I\_2DSubPixelMinPosition:



## Открытые члены

- virtual void Find (float \*S9, float &dx, float &dy)=0
- virtual void Release ()=0

## 8.17.1 Подробное описание

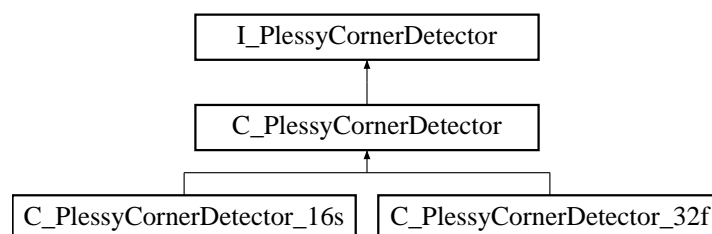
См. определение в файле isubpixel2d.h строка 7

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/isubpixel2d.h

## 8.18 Класс I\_PlessyCornerDetector

Граф наследования:I\_PlessyCornerDetector:



## Открытые члены

- virtual void Allocate (int w, int h, int ww)=0
- virtual void DeAllocate ()=0
- virtual void FindCorners (unsigned char \*Picture, int w, int h, int ww, float \*px, float \*py, int &nc)=0
- virtual void Release ()=0
- virtual void SetThreshold (float \_threshold)=0

## 8.18.1 Подробное описание

См. определение в файле iPlessy.h строка 6

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/iPlessy.h

## 8.19 Структура int15in16x4

## Открытые атрибуты

- unsigned long items

### 8.19.1 Подробное описание

См. определение в файле `nmtype.h` строка 234

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

## 8.20 Структура int30in32x2

Открытые атрибуты

- `unsigned long items`

### 8.20.1 Подробное описание

См. определение в файле `nmtype.h` строка 320

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

## 8.21 Структура int31in32x2

Открытые атрибуты

- `unsigned long items`

### 8.21.1 Подробное описание

См. определение в файле `nmtype.h` строка 300

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

## 8.22 Шаблон класса `mtr< T >`

### Открытые члены

- `void origin (int y, int x)`
- `void setval (int y, int x, const T &val)`
- `void setvalx (int y, int x, const T &val)`
- `T getval (int y, int x) const`
- `T getvalx (int y, int x) const`
- `mtr (int nHeight, int nWidth, int nBorder=0)`
- `void resize (int nHeight, int nWidth, int nBorder=0)`
- `mtr (const mtr< T > &matr)`
- `void assign (T *Data, int nHeight, int nWidth, int nStride=0)`
- `mtr (T *Data, int nHeight, int nWidth, int nStride=0)`
- `mtr< T > & operator= (mtr< T > &matr)`
- `mtr< T > & operator= (T &val)`
- `T * operator[] (int row) const`
- `T & index (int idx)`
- `mtr< T > & operator*= (const T val)`
- `mtr< T > operator* (const T &val) const`
- `vec< T > operator* (const vec< T > &vect)`
- `mtr< T > operator* (const mtr< T > &matr)`
- `mtr< T > & operator+= (const T &val)`
- `mtr< T > & operator+= (const mtr< T > &matr)`
- `mtr< T > operator+ (const mtr< T > &matr) const`
- `mtr< T > & operator-= (const mtr< T > &matr)`
- `mtr< T > & operator-= (const T &val)`
- `mtr< T > operator- (const mtr< T > &matr) const`
- `mtr< T > operator- () const`
- `mtr< T > & operator/= (const T val)`
- `mtr< T > operator/ (const T val) const`
- `mtr< T > & operator>>= (const int shr)`
- `mtr< T > operator>> (const int shr) const`
- `mtr< T > & operator<<= (const int shl)`
- `mtr< T > operator<< (const int shl) const`
- `mtr< T > & operator &= (const T &val)`
- `mtr< T > & operator &= (const mtr< T > &matr)`
- `mtr< T > operator & (const mtr< T > &matr) const`
- `mtr< T > & operator|= (const mtr< T > &matr)`
- `mtr< T > operator| (const mtr< T > &matr) const`
- `mtr< T > & operator^= (const mtr< T > &matr)`
- `mtr< T > & operator^= (const T &val)`
- `mtr< T > operator^ (const mtr< T > &matr) const`
- `mtr< T > operator^ (const T &val) const`
- `mtr< T > operator~ ()`
- `void set (const T val)`
- `mtr< T > transpose ()`
- `mtr< T > & diag (T val)`
- `T * addr (int y, int x)`
- `void reset ()`
- `int sum ()`
- `vec< T > getvec (int y) const`
- `vec< T > getcol (int x) const`
- `T minpos (int &ypos, int &xpos)`



- T maxpos (int &ypos, int &xpos)
- void CopyTo (T \*pData)
- void CopyFrom (T \*pData)
- template<class T2 >  
void ConvertTo (T2 \*pData)
- template<class T2 >  
void ConvertFrom (T2 \*pData)
- template<>  
[mtr](#)< double > & operator<<= (const int Shl)

#### Открытые атрибуты

- int m\_border
- int m\_stride
- int m\_height
- int m\_width
- int m\_size
- int m\_x0
- int m\_y0
- T \* m\_data

#### Защищенные данные

- T \* m\_container

##### 8.22.1 Подробное описание

```
template<class T>
class mtr< T >
```

См. определение в файле `tmatrix.h` строка 87

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmtl/tmatrix.h`

## 8.23 Структура nm16sc

#### Открытые атрибуты

- signed short r
- signed short c

##### 8.23.1 Подробное описание

См. определение в файле `nmtypе.h` строка 1240

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtypе.h`

## 8.24 Класс nmchar

### Открытые члены

- nmchar (unsigned int \*p, int offset)
- nmchar (nmchar &ch)
- nmchar & operator= (nmchar ch)
- unsigned int operator+ (nmchar &ch)
- nmchar & operator&= (unsigned int val)
- nmchar & operator|= (unsigned int val)
- nmchar & operator= (unsigned int val)
- operator unsigned char ()
- uint8ptr operator& ()

### Открытые атрибуты

- unsigned int \* adr
- int idx

#### 8.24.1 Подробное описание

См. определение в файле nmchar.h строка 13

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmchar.h

## 8.25 Шаблон класса nmchar1D< N >

### Открытые члены

- nmchar & operator[] (int idx)
- operator uint8ptr ()
- unsigned int \* ptr ()

### Открытые атрибуты

- nmchar deref
- unsigned int data [(N+3)/4]

#### 8.25.1 Подробное описание

```
template<int N>
class nmchar1D< N >
```

См. определение в файле nmchar.h строка 363

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmchar.h

## 8.26 Шаблон класса nmchar2D< Y, X >

### Открытые члены

- `uint8ptr & operator[] (int idx)`
- `unsigned int * ptr ()`

### Открытые атрибуты

- `uint8ptr arr`
- `unsigned int data [Y *X/4]`

#### 8.26.1 Подробное описание

```
template<int Y, int X>
class nmchar2D< Y, X >
```

См. определение в файле nmchar.h строка 345

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmchar.h`

## 8.27 Шаблон класса nmintpack< T >

### Открытые члены

- `nmintpack (T *base, int idx)`
- `nmintpack< T > & operator= (const nmintpack< T > &val)`
- `nmintpack< T > & operator= (const int &val)`
- `operator int (void) const`
- `__INLINE__ int intdisp (int indx)`
- `__INLINE__ int bitdisp (int indx)`

### Открытые атрибуты

- `T * m_container`
- `int m_disp`

#### 8.27.1 Подробное описание

```
template<class T>
class nmintpack< T >
```

См. определение в файле tnmvecpack.h строка 28

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmtl/tnmvecpack.h`

8.28 Шаблон класса `nmmtr< T >`

```
#include <tnmmtr.h>
```

## Открытые члены

- `nmmtr (int Height, int Width, int Border=0)`
- `nmvec< T > operator[] (int y) const`
- `nmmtr (nmmtr< T > &mtr)`
- `nmmtr (T *Data, int Height, int Width, int Stride=0)`
- `nmmtr (const T *Data, int Height, int Width, int Stride=0)`
- `nmmtr< T > & operator= (const nmmtr< T > &mtr)`
- `nmmtr< T > & operator*= (const nmint< T > &val)`
- `template<class T2 >`  
`nmmtr< T2 > operator* (const nmint< T2 > &val)`
- `template<class T2 >`  
`nmvec< T2 > operator* (const nmvec< T2 > &vec)`
- `template<class T2 >`  
`nmmtr< T2 > operator* (const nmmtr< T2 > &mtr)`
- `nmvec< T > & operator+= (const nmint< T > &val)`
- `nmmtr< T > & operator+= (const nmmtr< T > &mtr)`
- `nmmtr< T > operator+ (const nmmtr< T > &mtr) const`
- `nmmtr< T > & operator-= (const nmmtr< T > &mtr)`
- `nmmtr< T > operator- (const nmmtr< T > &mtr) const`
- `nmmtr< T > operator- () const`
- `nmmtr< T > & operator/= (const T val)`
- `nmmtr< T > operator/ (const nmint< T > val) const`
- `nmmtr< T > & operator>>= (const int shr)`
- `nmmtr< T > operator>> (const int shr) const`
- `nmmtr< T > & operator<<= (const int shl)`
- `nmmtr< T > operator<< (const int shl) const`
- `nmmtr< T > & operator|= (const nmmtr< T > &mtr)`
- `nmmtr< T > operator| (const nmmtr< T > &mtr) const`
- `nmmtr< T > & operator &= (const nmmtr< T > &mtr)`
- `nmmtr< T > operator & (const nmmtr< T > &mtr) const`
- `nmmtr< T > & operator^= (const nmint< T > &val)`
- `nmmtr< T > operator^ (const nmint< T > &val) const`
- `nmmtr< T > operator~ () const`
- `T * addr (int y, int x)`
- `nmvec< T > vec (int y)`
- `void fill (nmint< T > &nVal)`
- `nmmtr< T > transpose ()`
- `template<class T2 >`  
`void set (nmmtr< T2 > &mSrcMtr) const`
- `template<class T2 >`  
`void set (mtr< T2 > &Mtr)`
- `void set (const T val)`
- `void reset ()`

## Открытые атрибуты

- int m\_height
- int m\_width
- int m\_size
- int m\_stride
- int m\_border
- T \* m\_data

## Защищенные данные

- T \* m\_container

## 8.28.1 Подробное описание

```
template<class T>
class nmmtr< T >
```

класс матриц.

См. определение в файле tnmtr.h строка 37

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmtl/tnmmtr.h

## 8.29 Структура NmppiFFTSpec\_32fcr

## Открытые атрибуты

- NmppsFFTSpec\_32fcr \* factors
- nm32fcr \* bufferFFT

## 8.29.1 Подробное описание

См. определение в файле fft\_32fcr.h строка 24

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/fft\_32fcr.h

## 8.30 Структура NmppsFFTSpec

## Открытые атрибуты

- nm32sc \* buffer [FFT\_SPEC\_NUM\_BUFFERS]
- void \* fftTable [FFT\_SPEC\_NUM\_TABLES]
- int shift [FFT\_SPEC\_NUM\_SHIFTS]
- int amp [FFT\_SPEC\_NUM\_AMPLITUDES]
- Free32Func \* free

### 8.30.1 Подробное описание

См. определение в файле `fft.h` строка 119

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmpls/fft.h`

## 8.31 Структура `NmppsFFTSpec_32fcr`

Открытые атрибуты

- `nm32fcr` \* Buffers [NUMBUFF1]
- `nm32fcr` \* Buffs [NUMBUFF2]
- `int` order

### 8.31.1 Подробное описание

См. определение в файле `fft_32fcr.h` строка 17

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/fft_32fcr.h`

## 8.32 Структура `NmppsFrame_16s`

Открытые атрибуты

- `void` \* pull
- `nm16s` \* data

### 8.32.1 Подробное описание

См. определение в файле `malloc32.h` строка 97

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/malloc32.h`

## 8.33 Структура `NmppsFrame_16u`

Открытые атрибуты

- `void` \* pull
- `nm16u` \* data

### 8.33.1 Подробное описание

См. определение в файле malloc32.h строка 92

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

## 8.34 Структура NmppsFrame\_32s

Открытые атрибуты

- void \* pull
- [nm32s](#) \* data

### 8.34.1 Подробное описание

См. определение в файле malloc32.h строка 107

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

## 8.35 Структура NmppsFrame\_32u

Открытые атрибуты

- void \* pull
- [nm32u](#) \* data

### 8.35.1 Подробное описание

См. определение в файле malloc32.h строка 102

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

## 8.36 Структура NmppsFrame\_64s

Открытые атрибуты

- void \* pull
- [nm64s](#) \* data

### 8.36.1 Подробное описание

См. определение в файле malloc32.h строка 117

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

## 8.37 Структура NmppsFrame\_64u

Открытые атрибуты

- void \* pull
- nm64u \* data

### 8.37.1 Подробное описание

См. определение в файле malloc32.h строка 112

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

## 8.38 Структура NmppsFrame\_8s

Открытые атрибуты

- void \* pull
- nm8s \* data

### 8.38.1 Подробное описание

См. определение в файле malloc32.h строка 87

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

## 8.39 Структура NmppsFrame\_8u

Открытые атрибуты

- void \* pull
- nm8u \* data



### 8.39.1 Подробное описание

См. определение в файле malloc32.h строка 82

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

## 8.40 Структура NmppsMallocSpec

Открытые атрибуты

- Malloc32Func \* allocator [4]
- enum MALLOC32\_MODE mode
- uint32 random
- fseq32 priority
- uint32 status
- uint32 time
- uint32 timeBest
- uint32 routePos
- fseq64 route [NMPPS\_MALLOC\_LIMIT/16]
- fseq64 bestRoute [NMPPS\_MALLOC\_LIMIT/16]
- void \* allocHistory [NMPPS\_MALLOC\_LIMIT]
- void \* freeHistory [NMPPS\_MALLOC\_LIMIT]
- uint32 allocHistoryPos
- uint32 freeHistoryPos
- uint32 firstPass

### 8.40.1 Подробное описание

См. определение в файле malloc32.h строка 42

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

## 8.41 Структура NmppsTmpSpec

Открытые атрибуты

- void \* buffer0
- void \* buffer1

### 8.41.1 Подробное описание

См. определение в файле malloc32.h строка 60

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/malloc32.h

## 8.42 Структура nmreg

```
#include <nmtype.h>
```

Открытые атрибуты

- `int nVal`

### 8.42.1 Подробное описание

NM регистр.

См. определение в файле `nmtype.h` строка 54

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

## 8.43 Класс nmshort

Открытые члены

- `__INLINE__ nmshort (nmshort &ch)`
- `__INLINE__ nmshort & operator= (nmshort ch)`
- `__INLINE__ unsigned int operator+ (nmshort &ch)`
- `__INLINE__ nmshort & operator= (unsigned int val)`
- `__INLINE__ operator unsigned char ()`
- `__INLINE__ uint16ptr operator& ()`

Открытые атрибуты

- `unsigned int * adr`
- `int idx`

### 8.43.1 Подробное описание

См. определение в файле `nmshort.h` строка 8

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmshort.h`

## 8.44 Шаблон класса nmshort2D< Y, X >

### Открытые члены

- `uint16ptr` & operator[] (int idx)
- `unsigned int * ptr` ()

### Открытые атрибуты

- `uint16ptr` arr
- `unsigned int data [Y *X/2]`

#### 8.44.1 Подробное описание

```
template<int Y, int X>
class nmshort2D< Y, X >
```

См. определение в файле nmshort.h строка 292

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmshort.h

## 8.45 Шаблон класса nmvecpack< T >

```
#include <tnmvecpack.h>
```

### Открытые члены

- `nmvecpack` (void \*Data, int Size, int Border=0)
- `nmvecpack` (int Size, int Border=0)
- `nmvecpack` (const `nmvecpack`< T > &vec)
- `nmvecpack`< T > & operator= (const `nmvecpack`< T > &vec)
- `__INLINE__ nmintpack`< T > operator[] (int idx)
- `template<class T2 >`  
`nmvecpack`< T2 > & operator\*= (const `nmint`< T2 > val)
- `template<class T2 >`  
`nmvecpack`< T2 > operator\* (const `nmint`< T2 > &val) const
- `template<class T2 >`  
`nmint`< T2 > operator\* (const `nmvecpack`< T2 > &vec) const
- `nmvecpack`< T > & operator+= (const `nmint`< T > &val)
- `nmvecpack`< T > & operator+= (const `nmvecpack`< T > &vec)
- `nmvecpack`< T > operator+ (const `nmint`< T > &val) const
- `nmvecpack`< T > operator+ (const `nmvecpack`< T > &vec) const
- `nmvecpack`< T > & operator-= (const `nmint`< T > &val)
- `nmvecpack`< T > & operator-= (const `nmvecpack`< T > &vec)
- `nmvecpack`< T > operator- () const
- `nmvecpack`< T > operator- (const `nmint`< T > &val) const

- `nmvecpack< T > operator- (const nmvecpack< T > &vec) const`
- `nmvecpack< T > & operator/= (const nmint< T > val)`
- `nmvecpack< T > operator/ (const T val) const`
- `nmvecpack< T > & operator>>= (const int shr)`
- `nmvecpack< T > operator>> (const int shr) const`
- `nmvecpack< T > & operator<<= (const int shl)`
- `nmvecpack< T > operator<< (const int shl) const`
- `nmvecpack< T > & operator|= (const nmint< T > &val)`
- `nmvecpack< T > & operator|= (const nmvecpack< T > &vec)`
- `nmvecpack< T > operator| (const nmint< T > &val) const`
- `nmvecpack< T > operator| (const nmvecpack< T > &vec) const`
- `nmvecpack< T > & operator &= (const nmint< T > &val)`
- `nmvecpack< T > & operator &= (const nmvecpack< T > &vec)`
- `nmvecpack< T > operator & (const nmint< T > &val) const`
- `nmvecpack< T > operator & (const nmvecpack< T > &vec) const`
- `nmvecpack< T > & operator^= (const nmint< T > &val)`
- `nmvecpack< T > & operator^= (const nmvecpack< T > &vec)`
- `nmvecpack< T > operator^ (const nmint< T > &val) const`
- `nmvecpack< T > operator^ (const nmvecpack< T > &vec) const`
- `nmvecpack< T > & operator~ () const`
- `bool operator== (const nmvecpack< T > &vec) const`
- `bool operator!= (const nmvecpack< T > &vec) const`
- `template<class T2 >`  
`void SetData (T2 *Data)`
- `template<class T2 >`  
`void GetData (T2 *Data)`
- `void reset ()`

### Открытые атрибуты

- `int m_size`
- `T * m_data`

### Защищенные данные

- `T * m_container`
- `int m_border`

### 8.45.1 Подробное описание

```
template<class T>
class nmvecpack< T >
```

Класс векторов.

Примеры:

```
int Test[10]={1,125,3,4,5,6,7,8,9,10};
int Res [10];
nmvecpack<int> A0(3);
nmvecpack<int> B0(3);
nmvecpack<int> C0(3);
```

См. определение в файле `tnmvecpack.h` строка 137

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmtl/tnmvecpack.h`

## 8.46 Структура RGB32\_nm10s

Открытые атрибуты

- int nB:10
- int nG:10
- int nR:10
- int nA:2

### 8.46.1 Подробное описание

См. определение в файле iDef.h строка 59

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iDef.h

## 8.47 Структура RGB32\_nm10u

Открытые атрибуты

- unsigned int nB:10
- unsigned int nG:10
- unsigned int nR:10
- unsigned int nA:2

### 8.47.1 Подробное описание

См. определение в файле iDef.h строка 51

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iDef.h

## 8.48 Структура RGB32\_nm8s

Открытые атрибуты

- int nB:8
- int nG:8
- int nR:8
- int nA:8

#### 8.48.1 Подробное описание

См. определение в файле iDef.h строка 43

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iDef.h

### 8.49 Структура RGB32\_nm8u

Открытые атрибуты

- unsigned int nB:8
- unsigned int nG:8
- unsigned int nR:8
- unsigned int nA:8

#### 8.49.1 Подробное описание

См. определение в файле iDef.h строка 35

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iDef.h

### 8.50 Структура RGB64\_nm16u

Открытые атрибуты

- unsigned int nB:16
- unsigned int nG:16
- unsigned int nR:16
- unsigned int nA:16

#### 8.50.1 Подробное описание

См. определение в файле iDef.h строка 67

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iDef.h

## 8.51 Класс RPoint

### Открытые члены

- RPoint (const [RPoint](#) &p)
- RPoint (double \_x, double \_y)
- [RPoint](#) & operator= (const [RPoint](#) &p)

### Открытые атрибуты

- double x
- double y

#### 8.51.1 Подробное описание

См. определение в файле iCellTexture.h строка 25

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmpli/iCellTexture.h

## 8.52 Структура S\_BufferInfo

класс буфер - заголовок в начале выделяемой динамической памяти

```
#include <multiheap.h>
```

### Открытые члены

- int \* [DataBegin](#) ()  
Возвращает указатель на данные в буфере
- int \* [DataEnd](#) ()  
Возвращает указатель на конец данных в буфере (следу)
- int \* [EndGuardBits](#) ()  
Возвращает указатель на конечные защитные поля (2 слова)
- unsigned [CheckGuardBits](#) ()

### Открытые атрибуты

- int guardInfoBits0
- size\_t [size32Buffer](#)  
< размер массива данных в буфере в 32-х словах
- [S\\_BufferInfo](#) \* [pPrevBuffer](#)  
< указатель на предыдущий буфер в списке
- [S\\_BufferInfo](#) \* [pNextBuffer](#)  
< указатель на следующий буфер в списке
- bool [isLocked](#)  
< флаг блокировки буфера, запрещающий его удаление с помощью Release()
- int guardInfoBits1

### 8.52.1 Подробное описание

класс буфер - заголовок в начале выделяемой динамической памяти

См. определение в файле multiheap.h строка 14

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/multiheap.h

## 8.53 Структура S\_IMG\_FilterKernel

Открытые атрибуты

- `nm32s * pDispArray`
- `nm32s * pWeightMatrix`

### 8.53.1 Подробное описание

См. определение в файле iFilter.h строка 24

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iFilter.h

## 8.54 Структура S\_IMG\_FilterKernel\_32s32s

Открытые атрибуты

- `nm32s * pDispArray`
- `nm32s * pWeightMatrix`
- `int nKerWidth`
- `int nKerHeight`

### 8.54.1 Подробное описание

См. определение в файле iFilter.h строка 37

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iFilter.h



## 8.55 Структура `s_int32x2`

Открытые атрибуты

- `int hi`
- `int lo`

### 8.55.1 Подробное описание

См. определение в файле `nmtype.h` строка 258

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

## 8.56 Структура `s_nm32fc`

Открытые атрибуты

- `float re`
- `float im`

### 8.56.1 Подробное описание

См. определение в файле `nmtype.h` строка 1275

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

## 8.57 Структура `s_nm32fcr`

Открытые атрибуты

- `float im`
- `float re`

### 8.57.1 Подробное описание

См. определение в файле `nmtype.h` строка 1282

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

## 8.58 Структура `s_nm32sc`

Открытые атрибуты

- `nm32s re`
- `nm32s im`

### 8.58.1 Подробное описание

См. определение в файле `nmtype.h` строка 1264

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

## 8.59 Структура `s_nm64sc`

Открытые атрибуты

- `long long re`
- `long long im`

### 8.59.1 Подробное описание

См. определение в файле `nmtype.h` строка 1311

### 8.59.2 Данные класса

#### 8.59.2.1 `im`

`long long s_nm64sc::im`

Мнимая часть комплексного числа.

См. определение в файле `nmtype.h` строка 1324

## 8.59.2.2 re

```
long long s_nm64sc::re
```

Вещественная часть комплексного числа.

См. определение в файле nmtype.h строка 1318

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

## 8.60 Структура s\_v16nm16s

```
#include <nmtype.h>
```

Открытые атрибуты

- unsigned long long vec [4]

## 8.60.1 Подробное описание

Тип векторной структуры, состоящей из 16-ти 16р. чисел со знаком.

См. определение в файле nmtype.h строка 1023

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

## 8.61 Структура s\_v16nm16u

```
#include <nmtype.h>
```

Открытые атрибуты

- unsigned long long vec [4]

## 8.61.1 Подробное описание

Тип векторной структуры, состоящей из 16-ти 16р. чисел без знака.

См. определение в файле nmtype.h строка 1161

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

## 8.62 Структура `s_v16nm32s`

```
#include <nmtypes.h>
```

Открытые атрибуты

- `unsigned long long vec [8]`

### 8.62.1 Подробное описание

Тип векторной структуры, состоящей из 16-ти 32-бит. чисел со знаком.

См. определение в файле `nmtypes.h` строка 1068

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtypes.h`

## 8.63 Структура `s_v16nm32u`

```
#include <nmtypes.h>
```

Открытые атрибуты

- `unsigned long long vec [8]`

### 8.63.1 Подробное описание

Тип векторной структуры, состоящей из 16-ти 32-бит. чисел без знака.

См. определение в файле `nmtypes.h` строка 1205

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtypes.h`

## 8.64 Структура `s_v16nm4u`

```
#include <nmtypes.h>
```

Открытые атрибуты

- `unsigned long long vec [1]`

#### 8.64.1 Подробное описание

Тип векторной структуры, состоящей из 16-ти 4-р. чисел без знака.

См. определение в файле nmtype.h строка 1095

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

### 8.65 Структура s\_v16nm8s

```
#include <nmtype.h>
```

Открытые атрибуты

- unsigned long long vec [2]

#### 8.65.1 Подробное описание

Тип векторной структуры, состоящей из 16-ти 8р. чисел со знаком.

См. определение в файле nmtype.h строка 988

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

### 8.66 Структура s\_v16nm8u

```
#include <nmtype.h>
```

Открытые атрибуты

- unsigned long long vec [2]

#### 8.66.1 Подробное описание

Тип векторной структуры, состоящей из 16-ти 8р. чисел без знака.

См. определение в файле nmtype.h строка 1128

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

## 8.67 Структура s\_v2nm32s

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [1]

### 8.67.1 Подробное описание

Тип векторной структуры, состоящей из 2-х 32р. чисел со знаком.

См. определение в файле nmtypes.h строка 1035

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtypes.h](#)

## 8.68 Структура s\_v2nm32u

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [1]

### 8.68.1 Подробное описание

Тип векторной структуры, состоящей из 2-х 32р. чисел без знака.

См. определение в файле nmtypes.h строка 1172

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtypes.h](#)

## 8.69 Структура s\_v4nm16s

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [1]

### 8.69.1 Подробное описание

Тип векторной структуры, состоящей из 4-х 16р. чисел со знаком.

См. определение в файле nmtype.h строка 999

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

## 8.70 Структура s\_v4nm16u

```
#include <nmtype.h>
```

Открытые атрибуты

- unsigned long long vec [1]

### 8.70.1 Подробное описание

Тип векторной структуры, состоящей из 4-х 16р. чисел без знака.

См. определение в файле nmtype.h строка 1139

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

## 8.71 Структура s\_v4nm32s

```
#include <nmtype.h>
```

Открытые атрибуты

- unsigned long long vec [2]

### 8.71.1 Подробное описание

Тип векторной структуры, состоящей из 4-х 32р. чисел со знаком.

См. определение в файле nmtype.h строка 1046

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/[nmtype.h](#)

## 8.72 Структура `s_v4nm32u`

```
#include <nmtypes.h>
```

Открытые атрибуты

- `unsigned long long vec [2]`

### 8.72.1 Подробное описание

Тип векторной структуры, состоящей из 4-х 32-бит. чисел без знака.

См. определение в файле `nmtypes.h` строка 1183

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtypes.h`

## 8.73 Структура `s_v4nm8u`

```
#include <nmtypes.h>
```

Открытые атрибуты

- `unsigned long long vec [1]`

### 8.73.1 Подробное описание

Тип векторной структуры, состоящей из 4-х 8-бит. чисел без знака.

См. определение в файле `nmtypes.h` строка 1106

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtypes.h`

## 8.74 Структура `s_v8nm16s`

```
#include <nmtypes.h>
```

Открытые атрибуты

- `unsigned long long vec [2]`



#### 8.74.1 Подробное описание

Тип векторной структуры, состоящей из 8-ми 16р. чисел со знаком.

См. определение в файле `nmtype.h` строка 1012

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

### 8.75 Структура s\_v8nm16u

```
#include <nmtype.h>
```

Открытые атрибуты

- `unsigned long long vec [2]`

#### 8.75.1 Подробное описание

Тип векторной структуры, состоящей из 8-ми 16р. чисел без знака.

См. определение в файле `nmtype.h` строка 1150

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

### 8.76 Структура s\_v8nm32s

```
#include <nmtype.h>
```

Открытые атрибуты

- `unsigned long long vec [4]`

#### 8.76.1 Подробное описание

Тип векторной структуры, состоящей из 8-ми 32р. чисел со знаком.

См. определение в файле `nmtype.h` строка 1057

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

## 8.77 Структура `s_v8nm32u`

```
#include <nmtypes.h>
```

Открытые атрибуты

- `unsigned long long vec [4]`

### 8.77.1 Подробное описание

Тип векторной структуры, состоящей из 8-ми 32р. чисел без знака.

См. определение в файле `nmtypes.h` строка 1194

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtypes.h`

## 8.78 Структура `s_v8nm8s`

```
#include <nmtypes.h>
```

Открытые атрибуты

- `unsigned long long vec [1]`

### 8.78.1 Подробное описание

Тип векторной структуры, состоящей из 8-ми 8р. чисел со знаком.

См. определение в файле `nmtypes.h` строка 977

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtypes.h`

## 8.79 Структура `s_v8nm8u`

```
#include <nmtypes.h>
```

Открытые атрибуты

- `unsigned long long vec [1]`

### 8.79.1 Подробное описание

Тип векторной структуры, состоящей из 8-ми 8p. чисел без знака.

См. определение в файле `nmtype.h` строка 1117

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

## 8.80 Структура SpecTmp1

Открытые атрибуты

- `void * buffer`
- `int status`
- `int mode`
- `fseq64 route`

### 8.80.1 Подробное описание

См. определение в файле `malloc32.h` строка 153

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/malloc32.h`

## 8.81 Структура `spot_struct`

Открытые атрибуты

- `int Xmin`  
координаты минимального прямоугольника, содержащего пятно
- `int Ymin`  
координаты минимального прямоугольника, содержащего пятно
- `int Xmax`  
координаты минимального прямоугольника, содержащего пятно
- `int Ymax`  
координаты минимального прямоугольника, содержащего пятно
- `int noPxl`  
номер начального пиксела следующего пятна в массиве `pixels`.
- `int dtSpot`  
время обработки пятна (в тактах процессора)

### 8.81.1 Подробное описание

См. определение в файле iFloodFill.h строка 84

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iFloodFill.h

## 8.82 Структура tagSegmentInfo

Открытые атрибуты

- int xMin
- int yMin
- int xMax
- int yMax
- int N

### 8.82.1 Подробное описание

См. определение в файле iFloodFill.h строка 22

Объявления и описания членов структуры находятся в файле:

- D:/GIT/nmpp/include/nmpli/iFloodFill.h

## 8.83 Шаблон класса tfixpoint< T, point >

Открытые члены

- `INLINE tfixpoint (const int val)`
- `INLINE tfixpoint (const float val)`
- `INLINE tfixpoint (const double val)`
- `INLINE tfixpoint< T, point > & operator= (const int val)`
- `INLINE tfixpoint< T, point > & operator= (const float val)`
- `tfixpoint< T, point > & operator= (const double val)`
- `INLINE tfixpoint< T, point > & operator= (const tfixpoint< T, point > &val)`
- `INLINE tfixpoint< T, point > & operator-= (const tfixpoint< T, point > &val)`
- `INLINE tfixpoint< T, point > & operator+= (const tfixpoint< T, point > &val)`
- `INLINE tfixpoint< T, point > operator- (const tfixpoint< T, point > &val) const`
- `INLINE tfixpoint< T, point > operator+ (const tfixpoint< T, point > &val) const`
- `INLINE tfixpoint< T, point > & operator++ (int)`
- `INLINE tfixpoint< T, point > & operator-- (int)`
- `template<class T2 , int point2>`  
`INLINE tfixpoint< T, point > & operator*= (const tfixpoint< T2, point2 > val)`
- `INLINE tfixpoint< T, point > & operator*= (const int val)`
- `INLINE tfixpoint< T, point > & operator*= (const float val)`
- `INLINE tfixpoint< T, point > & operator*= (const double val)`

- `template<int point2>`  
`INLINE tfixpoint< T, point > operator* (const tfixpoint< T, point2 > val) const`
- `INLINE tfixpoint< T, point > operator* (const int val) const`
- `INLINE tfixpoint< T, point > & operator/= (const tfixpoint< T, point > val)`
- `INLINE tfixpoint< T, point > operator/ (const tfixpoint< T, point > val) const`
- `INLINE tfixpoint< T, point > & operator>>= (const int y)`
- `INLINE tfixpoint< T, point > operator>> (const int y) const`
- `INLINE tfixpoint< T, point > & operator<<= (const int y)`
- `INLINE tfixpoint< T, point > operator<< (const int n) const`
- `INLINE tfixpoint< T, point > & operator^= (tfixpoint< T, point > &val)`
- `INLINE tfixpoint< T, point > operator^ (tfixpoint< T, point > &val) const`
- `INLINE tfixpoint< T, point > operator- () const`
- `bool operator> (const tfixpoint< T, point > &y) const`
- `bool operator>= (const tfixpoint< T, point > &y) const`
- `bool operator< (const tfixpoint< T, point > &y) const`
- `bool operator<= (const tfixpoint< T, point > &y) const`
- `bool operator== (const tfixpoint< T, point > &y) const`
- `bool operator!= (const tfixpoint< T, point > &y) const`
- `float flt ()`

## Открытые атрибуты

- `T m_value`

### 8.83.1 Подробное описание

```
template<class T, int point>
class tfixpoint< T, point >
```

См. определение в файле `tfixpoint.h` строка 52

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmtl/tfixpoint.h`

## 8.84 Структура Tmp2BuffSpec

### Открытые атрибуты

- `void * buffer0`
- `void * buffer1`
- `fseq64 route`
- `int mode`
- `int status`

### 8.84.1 Подробное описание

См. определение в файле `nmtype.h` строка 1350

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtype.h`

## 8.85 Класс `uint16ptr`

Открытые члены

- `__INLINE__ unsigned char * x86addr ()`
- `__INLINE__ uint16ptr (void *p)`
- `__INLINE__ uint16ptr (unsigned short *p)`
- `__INLINE__ uint16ptr (short *p)`
- `__INLINE__ uint16ptr (const uint16ptr &p)`
- `__INLINE__ uint16ptr (unsigned int *p)`
- `__INLINE__ uint16ptr (unsigned int *p, int offset)`
- `__INLINE__ nmshort & operator[] (int idx)`
- `__INLINE__ uint16ptr & operator= (unsigned int *ptr)`
- `__INLINE__ bool operator< (uint16ptr ptr)`
- `__INLINE__ bool operator>= (uint16ptr ptr)`
- `__INLINE__ int operator- (uint16ptr ptr)`
- `__INLINE__ uint16ptr & operator= (const uint16ptr &p)`
- `__INLINE__ unsigned int * ptr ()`
- `__INLINE__ nmshort & operator* ()`
- `__INLINE__ uint16ptr operator+ (int idx)`
- `__INLINE__ uint16ptr operator- (int idx)`
- `__INLINE__ uint16ptr & operator+= (int idx)`
- `__INLINE__ uint16ptr & operator-= (int idx)`
- `__INLINE__ uint16ptr operator++ (int)`
- `__INLINE__ uint16ptr & operator++ ()`
- `__INLINE__ unsigned int operator== (unsigned int N)`
- `__INLINE__ bool operator== (uint16ptr ptr)`

Открытые атрибуты

- `unsigned int * addr`
- `int idx`
- `nmshort arref`

### 8.85.1 Подробное описание

См. определение в файле `nmshort.h` строка 57

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmshort.h`

## 8.86 Класс uint8ptr

### Открытые члены

- unsigned char \* x86addr ()
- uint8ptr (void \*p)
- uint8ptr (unsigned char \*p)
- uint8ptr (const unsigned char \*p)
- uint8ptr (char \*p)
- uint8ptr (const uint8ptr &p)
- uint8ptr (unsigned int \*p)
- uint8ptr (unsigned int \*p, int offset)
- nmchar & operator[] (int idx)
- uint8ptr & operator= (unsigned int \*ptr)
- bool operator< (uint8ptr ptr)
- bool operator> (uint8ptr ptr)
- bool operator>= (uint8ptr ptr)
- int operator- (uint8ptr ptr)
- uint8ptr & operator= (const uint8ptr &p)
- unsigned int \* ptr ()
- nmchar & operator\* ()
- uint8ptr operator+ (int idx)
- uint8ptr operator- (int idx)
- uint8ptr & operator+= (int idx)
- uint8ptr & operator-= (int idx)
- uint8ptr operator++ (int)
- uint8ptr & operator++ ()
- bool operator== (uint8ptr ptr)

### Открытые атрибуты

- unsigned int \* addr
- int indx
- nmchar arref

#### 8.86.1 Подробное описание

См. определение в файле nmchar.h строка 79

Объявления и описания членов класса находятся в файле:

- D:/GIT/nmpp/include/nmchar.h

## 8.87 Структура v16nm4s

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [1]

#### 8.87.1 Подробное описание

Тип векторной структуры, состоящей из 16-ти 4-р. чисел со знаком.

См. определение в файле `nmtypes.h` строка 954

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtypes.h`

### 8.88 Структура `v4nm8s`

```
#include <nmtypes.h>
```

Открытые атрибуты

- unsigned long long vec [1]

#### 8.88.1 Подробное описание

Тип векторной структуры, состоящей из 4-х 8р. чисел со знаком.

См. определение в файле `nmtypes.h` строка 966

Объявления и описания членов структуры находятся в файле:

- `D:/GIT/nmpp/include/nmtypes.h`

### 8.89 Шаблон класса `vec< T >`

```
#include <tvector.h>
```



## Открытые члены

- `void resize (int Size, int Border=0)`
- `vec (T *Data, int Size, int Border=0)`
- `vec (int Size, int Border=0)`
- `vec (const vec< T > &vect)`
- `void reset ()`
- `vec< T > & InitRamp (T StartValue, T Increment)`
- `int MaxPos (T &maxval)`
- `int MinPos (T &minval)`
- `double Mean ()`
- `T & CustomMax ()`
- `vec< T > & operator= (const T &val)`
- `vec< T > & operator= (const vec< T > &vect)`
- `T * addr (int idx)`
- `T & operator[] (size_t idx)`
- `vec< T > & operator*= (const T &val)`
- `vec< T > operator* (const T &val) const`
- `T operator* (const vec< T > &vect) const`
- `vec< T > operator* (const mtr< T > matr) const`
- `vec< T > & operator+= (const T &val)`
- `vec< T > & operator+= (const vec< T > &vect)`
- `vec< T > operator+ (const T &val) const`
- `vec< T > operator+ (const vec< T > &vect) const`
- `vec< T > & operator-= (const T &val)`
- `vec< T > & operator-= (const vec< T > &vect)`
- `vec< T > operator- (const vec< T > &vect) const`
- `vec< T > operator- () const`
- `vec< T > & operator/= (const T val)`
- `vec< T > operator/ (const T val) const`
- `vec< T > & operator>>= (const int shr)`
- `vec< T > operator>> (const int shr) const`
- `vec< T > & operator<<= (const int shl)`
- `vec< T > operator<< (const int shl) const`
- `vec< T > & operator &= (const T &val)`
- `vec< T > & operator &= (const vec< T > &vect)`
- `vec< T > operator & (const T &val) const`
- `vec< T > operator & (const vec< T > &vect) const`
- `vec< T > & operator|= (const T &val)`
- `vec< T > & operator|= (const vec< T > &vect)`
- `vec< T > operator| (const T &val) const`
- `vec< T > operator| (const vec< T > &vect) const`
- `vec< T > & operator^= (const T &val)`
- `vec< T > & operator^= (const vec< T > &vect)`
- `vec< T > operator^ (const T &val) const`
- `vec< T > operator^ (const vec< T > &vect) const`
- `int sum ()`
- `bool operator== (const vec< T > &vect)`
- `bool operator!= (const vec< T > &vect)`

## Открытые атрибуты

- `int m_border`
- `int size`
- `T * m_data`

## Защищенные данные

- `T * m_container`

### 8.89.1 Подробное описание

```
template<class T>
class vec< T >
```

Класс векторов.

Примеры:

```
int Test[10]={1,125,3,4,5,6,7,8,9,10};
int Res [10];
vec<int> A0(3);
vec<int> B0(3);
vec<int> C0(3);
scalar<int> a0(2);
A0[0]=1;
A0[1]=2;
A0[2]=3;
B0.SetData(Test);
B0=A0;
a0=A0[1];

C0=A0+B0;
C0=A0*a0;
a0=A0*A0;
```

См. определение в файле `tvector.h` строка 79

Объявления и описания членов класса находятся в файле:

- `D:/GIT/nmpp/include/nmtl/tvector.h`

## Глава 9

# Файлы

### 9.1 Файл D:/GIT/nmpp/include/nmblas.h

nmblas (NeuroMatrix Basic Linear Algebra Subroutines)

#### Перечисления

- enum `nm_trans` { `nm_n` =0, `nm_t` =1 }  
Brief description.

#### Функции

- double `nmblas_dasum` (const int n, const double \*x, const int inc\_x)  
Brief description.
- void `nmblas_daxpy` (const int n, const double alpha, const double \*arr\_x, const int inc\_x, double \*arr\_y, const int inc\_y)
- void `nmblas_dcopy` (const int N, const double \*X, const int INCX, double \*Y, const int INCY)
- double `nmblas_ddot` (const int N, const double \*X, const int INCX, const double \*Y, const int INCY)
- double `nmblas_dnorm2` (const int n, const double \*x, const int inc\_x)
- void `nmblas_drot` (const int N, double \*X, const int INCX, double \*Y, const int INCY, const double C, const double S)
- void `nmblas_drotg` (double \*a, double \*b, double \*c, double \*s)
- void `nmblas_drotm` (const int N, double \*X, const int INCX, double \*Y, const int INCY, double \*param)
- void `nmblas_dscal` (const int N, const double ALPHA, double \*X, const int INCX)
- double `nmblas_dsdot` (const int N, const float \*X, const int INCX, const float \*Y, const int INCY)
- void `nmblas_dswap` (const int N, double \*X, const int INCX, double \*Y, const int INCY)
- double `nmblas_dznrm2` (const int n, const double \*x, const int inc\_x)
- int `nmblas_idamax` (const int N, const double \*X, const int INCX)
- int `nmblas_isamax` (const int N, const float \*X, const int INCX)
- float `nmblas_sasum` (const int n, const float \*x, const int inc)
- void `nmblas_saxpy` (int n, const float alpha, const float \*arr\_x, const int inc\_x, float \*arr\_y, const int inc\_y)
- float `nmblas_scnrm2` (const int N, const float \*X, const int INCX)
- void `nmblas_scopy` (const int N, const float \*X, const int INCX, float \*Y, const int INCY)

- float nmblas\_sdot (const int n, const float \*x, const int inc\_x, const float \*y, const int inc\_y)
- float nmblas\_sdsdot (const int N, const float B, const float \*X, const int INCX, const float \*Y, const int INCY)
- float nmblas\_snrm2 (const int N, const float \*X, const int INCX)
- void nmblas\_srot (const int N, float \*X, const int INCX, float \*Y, const int INCY, const float C, const float S)
- void nmblas\_srotm (const int N, float \*X, const int INCX, float \*Y, const int INCY, float \*param)
- void nmblas\_sscal (const int n, const float alpha, const float \*x, const int inc\_x)
- void nmblas\_sswap (const int N, const float \*X, const int INCX, const float \*Y, const int INCY)
- void nmblas\_sgemv (const enum nm\_trans TRANS, const int M, const int N, const float ALPHA, const float \*A, const int LDA, const float \*X, const int INCX, const float BETA, float \*Y, const int INCY)
- void nmblas\_dgemv (const enum nm\_trans TRANS, const int M, const int N, const double ALPHA, const double \*A, const int LDA, const double \*X, const int INCX, const double BETA, double \*Y, const int INCY)
- void MullMatrix\_f (void \*A, int pI, int ldA, void \*B, int pK, int ldB, void \*C, int pJ, int ldC, bool plusC)

### 9.1.1 Подробное описание

nmblas (NeuroMatrix Basic Linear Algebra Subroutines)

This class is used to demonstrate a number of section commands.

Автор

Alexandr Bolotnikov

Версия

1.0

Дата

2017-05-23T15:13:13

[Ошибка](#)

Предупреждения

Авторство

(c) RC Module Inc.

## 9.2 Файл D:/GIT/nmpp/include/nmtype.h

## Классы

- struct `nmreg`
- struct `int15in16x4`
- struct `s_int32x2`
- struct `int31in32x2`
- struct `int30in32x2`
- struct `v16nm4s`
- struct `v4nm8s`
- struct `s_v8nm8s`
- struct `s_v16nm8s`
- struct `s_v4nm16s`
- struct `s_v8nm16s`
- struct `s_v16nm16s`
- struct `s_v2nm32s`
- struct `s_v4nm32s`
- struct `s_v8nm32s`
- struct `s_v16nm32s`
- struct `s_v16nm4u`
- struct `s_v4nm8u`
- struct `s_v8nm8u`
- struct `s_v16nm8u`
- struct `s_v4nm16u`
- struct `s_v8nm16u`
- struct `s_v16nm16u`
- struct `s_v2nm32u`
- struct `s_v4nm32u`
- struct `s_v8nm32u`
- struct `s_v16nm32u`
- struct `nm16sc`
- struct `s_nm32sc`
- struct `s_nm32fc`
- struct `s_nm32fer`
- struct `s_nm64sc`
- struct `Tmp2BuffSpec`

## Макросы

- `#define MEM_LOCAL 0`
- `#define MEM_GLOBAL 1`
- `#define HEAP0 0`
- `#define HEAP1 1`
- `#define HEAP2 2`
- `#define HEAP3 3`
- `#define sizeof32(t) sizeof(t)`
- `#define VEC_NM1(X) unsigned data[(X)/32];`
- `#define VEC_NM2U(X) unsigned data[(X)/16];`
- `#define VEC_NM2S(X) int data[(X)/16];`
- `#define VEC_NM4U(X)`
- `#define VEC_NM4S(X) int data[(X)/8];`
- `#define VEC_NM8U(X) unsigned int data[(X)/4];`

- `#define VEC_NM8S(X)`
- `#define VEC_NM16U(X)`
- `#define VEC_NM16S(X)`
- `#define VEC_NM32U(X) nm32u data[(X)];`
- `#define VEC_NM32S(X)`
- `#define CAPACITY_nm64s 1`
- `#define CAPACITY_nm32s 2`
- `#define CAPACITY_nm16s 4`
- `#define CAPACITY_nm8s 8`
- `#define CAPACITY_nm4s 16`
- `#define CAPACITY_nm2s 32`
- `#define CAPACITY_nm1 64`
- `#define __INLINE__ static inline`

### Определения типов

- `typedef signed long long INT64`
- `typedef unsigned long long UINT64`
- `typedef void nm1`
- `typedef void nm2s`
- `typedef void nm4s`
- `typedef void nm8s`
- `typedef signed char nm8s7b`
- `typedef void nm16s`
- `typedef nm16s nm16s15b`
- `typedef int nm32s`
- `typedef struct s_int32x2 int32x2`
- `typedef int nm32s31b`
- `typedef int nm32s30b`
- `typedef long long nm64s`
- `typedef nm64s nm64s63b`
- `typedef void nm2u`
- `typedef void nm4u`
- `typedef nm4u nm4u3b`
- `typedef void nm8u`
- `typedef unsigned char nm8u7b`
- `typedef void nm16u`
- `typedef nm16u nm16u15b`
- `typedef unsigned int nm32u`
- `typedef unsigned int nm32u31b`
- `typedef unsigned long long nm64u`
- `typedef int int1b`
- `typedef int int2b`
- `typedef int int3b`
- `typedef int int4b`
- `typedef int int7b`
- `typedef int int8b`
- `typedef int int15b`
- `typedef int int16b`
- `typedef int int30b`
- `typedef int int31b`
- `typedef int int32b`
- `typedef INT64 int63b`
- `typedef INT64 int64b`

- typedef unsigned int uint1b
- typedef unsigned int uint2b
- typedef unsigned int uint3b
- typedef unsigned int uint4b
- typedef unsigned int uint7b
- typedef unsigned int uint8b
- typedef unsigned int uint15b
- typedef unsigned int uint16b
- typedef unsigned int uint31b
- typedef unsigned int uint32b
- typedef UINT64 uint63b
- typedef nm64u uint64b
- typedef struct s\_v8nm8s v8nm8s
- typedef struct s\_v16nm8s v16nm8s
- typedef struct s\_v4nm16s v4nm16s
- typedef struct s\_v8nm16s v8nm16s
- typedef struct s\_v16nm16s v16nm16s
- typedef struct s\_v2nm32s v2nm32s
- typedef struct s\_v4nm32s v4nm32s
- typedef struct s\_v8nm32s v8nm32s
- typedef struct s\_v16nm32s v16nm32s
- typedef v16nm8s v16nm8s7b
- typedef struct s\_v16nm4u v16nm4u
- typedef struct s\_v4nm8u v4nm8u
- typedef struct s\_v8nm8u v8nm8u
- typedef struct s\_v16nm8u v16nm8u
- typedef struct s\_v4nm16u v4nm16u
- typedef struct s\_v8nm16u v8nm16u
- typedef struct s\_v16nm16u v16nm16u
- typedef struct s\_v2nm32u v2nm32u
- typedef struct s\_v4nm32u v4nm32u
- typedef struct s\_v8nm32u v8nm32u
- typedef struct s\_v16nm32u v16nm32u
- typedef v16nm4u v16nm4b3u
- typedef struct s\_nm32sc nm32sc
- typedef struct s\_nm32fc nm32fc
- typedef struct s\_nm32fcr nm32fcr
- typedef float nm32f
- typedef double nm64f
- typedef struct s\_nm64sc nm64sc
- typedef unsigned long long uint64
- typedef unsigned int uint32
- typedef uint64 fifo64
- typedef uint32 fseq32
- typedef uint64 seq64
- typedef uint64 fseq64

### 9.2.1 Подробное описание

Файл, содержащий определения типов упакованных данных.

## 9.2.2 Макросы

### 9.2.2.1 CAPACITY\_nm64s

```
#define CAPACITY_nm64s 1
```

•

См. определение в файле nmtype.h строка 1376

### 9.2.2.2 VEC\_NM16S

```
#define VEC_NM16S(  
    X )
```

Макроопределение:

```
int data[(X)/2]; \
void set(int i, int val){ ((unsigned short*)data)[i]=val;} \
short operator[] (int index){ return ((unsigned short*)data)[index];}
```

См. определение в файле nmtype.h строка 933

### 9.2.2.3 VEC\_NM16U

```
#define VEC_NM16U(  
    X )
```

Макроопределение:

```
unsigned int data[(X)/2]; \
void set(int i, int val){ ((unsigned short*)data)[i]=val;} \
unsigned short operator[] (int index){ return ((unsigned short*)data)[index];}
```

См. определение в файле nmtype.h строка 928



## 9.2.2.4 VEC\_NM32S

```
#define VEC_NM32S(  
    X )
```

Макроопределение:

```
int data[(X)]; \n  
void set(int i, int val){ ((int*)data)[i]=val;} \n  
int operator[] (int index){ return ((int*)data)[index];}
```

См. определение в файле nmtype.h строка 939

## 9.2.2.5 VEC\_NM4U

```
#define VEC_NM4U(  
    X )
```

Макроопределение:

```
unsigned data[(X)/8]; \n  
uint4b get(int nIndex){return nmget((nm4u*)data,nIndex); } \n  
uint4b operator[] (int index){ return nmget((nm4u*)data,index);}
```

См. определение в файле nmtype.h строка 915

## 9.2.2.6 VEC\_NM8S

```
#define VEC_NM8S(  
    X )
```

Макроопределение:

```
int data[(X)/4]; \n  
void set(int i, int val){ ((char*)data)[i]=val;} \n  
char operator[] (int index){ return ((char*)data)[index];}
```

См. определение в файле nmtype.h строка 923



# Предметный указатель

- Арифметические действия, [114](#)
- Арифметические операции, [96](#), [204](#)
- Базовые регистровые функции библиотеки, [315](#)
- Быстрое преобразование Фурье, [167](#)
- Блочное переупорядочивание, [109](#)
- Целевые функции, [320](#)
- Элементарные функции, [318](#)
- Функции деинтерлейсинга, [97](#)
  - IMG\_DeinterlaceBlend, [97](#)
  - IMG\_DeinterlaceSplit, [97](#)
- Функции графического вывода текста, [118](#)
  - hex2ascii, [118](#)
  - IMG\_Print8x15, [119](#)
- Функции обработки изображений, [312](#)
- Функции обработки сигналов, [311](#)
- Функции поддержки, [117](#), [134](#), [202](#)
- Инициализация, [92](#)
- Инициализация и копирование, [116](#), [120](#), [203](#)
- Изменение размеров, [166](#)
- КИХ-фильтрация, [99](#), [171](#)
- Логические и бинарные операции, [205](#)
- Масочная фильтрация, [115](#), [165](#)
- Матричные функции, [310](#)
- Операции сравнения, [206](#)
- Переупорядочивание и сортировка, [207](#)
- Переупорядочивание изображений, [108](#)
- Скалярные функции, [313](#)
- Свертка, [164](#)
- Типы данных, [308](#)
- Типы скалярных данных, [195](#)
  - int15b, [195](#)
  - int16b, [196](#)
  - int1b, [196](#)
  - int2b, [196](#)
  - int30b, [196](#)
  - int31b, [197](#)
  - int32b, [197](#)
  - int3b, [197](#)
  - int4b, [197](#)
  - int63b, [198](#)
  - int64b, [198](#)
  - int7b, [198](#)
  - int8b, [198](#)
  - uint15b, [199](#)
  - uint16b, [199](#)
  - uint1b, [199](#)
  - uint2b, [199](#)
  - uint31b, [200](#)
  - uint32b, [200](#)
  - uint3b, [200](#)
  - uint4b, [200](#)
  - uint63b, [201](#)
  - uint64b, [201](#)
  - uint7b, [201](#)
  - uint8b, [201](#)
- Типы векторных данных, [182](#)
  - nm1, [184](#)
  - nm16s, [184](#)
  - nm16s15b, [184](#)
  - nm16u, [185](#)
  - nm16u15b, [185](#)
  - nm2s, [185](#)
  - nm2u, [186](#)
  - nm32s, [186](#)
  - nm32s30b, [186](#)
  - nm32s31b, [187](#)
  - nm32u, [187](#)
  - nm32u31b, [187](#)
  - nm4s, [188](#)
  - nm4u, [188](#)
  - nm4u3b, [188](#)
  - nm64s, [189](#)
  - nm64s63b, [189](#)
  - nm64u, [189](#)
  - nm8s, [190](#)
  - nm8s7b, [190](#)
  - nm8u, [190](#)
  - nm8u7b, [191](#)
  - v16nm16s, [191](#)
  - v16nm16u, [191](#)
  - v16nm32s, [191](#)
  - v16nm32u, [191](#)
  - v16nm4b3u, [192](#)
  - v16nm4u, [192](#)
  - v16nm8s, [192](#)
  - v16nm8s7b, [192](#)
  - v16nm8u, [192](#)
  - v2nm32s, [192](#)
  - v2nm32u, [193](#)
  - v4nm16s, [193](#)
  - v4nm16u, [193](#)
  - v4nm32s, [193](#)
  - v4nm32u, [193](#)
  - v4nm8u, [193](#)
  - v8nm16s, [193](#)
  - v8nm16u, [193](#)
  - v8nm32s, [194](#)
  - v8nm32u, [194](#)

- v8nm8s, [194](#)
  - v8nm8u, [194](#)
- Векторные функции, [309](#)
- Векторно-матричные операции, [135](#)
- функции взвешенного суммирования, [319](#)
- контроль переполнения, [316](#)
  - GetVec, [316](#)
  - operator<<, [317](#)
- AllocateMaxAvail
  - C\_Heap, [391](#)
- BLASS-LEVEL1, [66](#)
  - nm\_trans, [66](#)
  - nmblas\_dasum, [66](#)
- C\_2DSubPixelMinPosition, [387](#)
- C\_2DTrigSubPixelMinPosition, [387](#)
- C\_Allocator32, [388](#)
- C\_BoxImg< T >, [389](#)
- C\_BoxVec< T >, [389](#)
- C\_Heap, [390](#)
  - AllocateMaxAvail, [391](#)
- C\_Img< T >, [391](#)
- C\_MultiHeap, [392](#)
- C\_PlessyCornerDetector, [393](#)
- C\_PlessyCornerDetector\_16s, [394](#)
- C\_PlessyCornerDetector\_32f, [395](#)
- C\_RingBufferRemote< T >, [396](#)
- C\_WarpImg< T >, [397](#)
- CAPACITY\_nm64s
  - nmtype.h, [444](#)
- CIMG\_FIR< nmbits\_in, nmbits\_out >, [398](#)
- CIMG\_FIR
  - CIMG\_FIR, [399](#)
  - Filter, [400](#)
  - SetWeights, [400](#)
- D:/GIT/nmpp/include/nmblas.h, [439](#)
- D:/GIT/nmpp/include/nmtype.h, [441](#)
- DFT-8, [19](#)
  - nmppsDFT8Fwd\_32fcr, [19](#)
- ds\_struct, [401](#)
- EnterHardMode, [401](#)
- FFT-1024, [33](#), [146](#)
  - FFT\_Fwd1024, [146](#)
  - nmppsFFT1024Fwd\_32fcr, [33](#)
  - nmppsFFT1024FwdInitAlloc\_32fcr, [33](#)
- FFT-128, [27](#)
  - nmppsFFT128Fwd\_32fcr, [27](#)
  - nmppsFFT128FwdInitAlloc\_32fcr, [27](#)
- FFT-16, [21](#)
  - nmppsFFT16Fwd\_32fcr, [21](#)
  - nmppsFFT16FwdInitAlloc\_32fcr, [21](#)
- FFT-2048, [35](#), [151](#)
  - FFT\_Fwd2048, [151](#)
  - nmppsFFT2048Fwd\_32fcr, [35](#)
  - nmppsFFT2048FwdInitAlloc\_32fcr, [35](#)
- FFT-256, [29](#), [136](#)
  - FFT\_Fwd256, [136](#)
  - nmppsFFT256Fwd\_32fcr, [29](#)
  - nmppsFFT256FwdInitAlloc\_32fcr, [29](#)
- FFT-32, [23](#)
  - nmppsFFT32Fwd\_32fcr, [23](#)
  - nmppsFFT32FwdInitAlloc\_32fcr, [23](#)
- FFT-4096, [37](#), [156](#)
  - FFT\_Fwd4096, [156](#)
  - nmppsFFT4096Fwd\_32fcr, [37](#)
  - nmppsFFT4096FwdInitAlloc\_32fcr, [37](#)
- FFT-512, [31](#), [141](#)
  - FFT\_Fwd512, [141](#)
  - nmppsFFT512Fwd\_32fcr, [31](#)
  - nmppsFFT512FwdInitAlloc\_32fcr, [31](#)
- FFT-64, [25](#)
  - nmppsFFT64Fwd\_32fcr, [25](#)
  - nmppsFFT64FwdInitAlloc\_32fcr, [25](#)
- FFT-8192, [160](#)
  - FFT\_Fwd8192, [160](#)
- FFT-Common, [58](#)
  - nmppsFFTFwd\_32fcr, [58](#)
  - nmppsFFTFwdInitAlloc\_32fcr, [58](#)
- FFT\_Fwd1024
  - FFT-1024, [146](#)
- FFT\_Fwd2048
  - FFT-2048, [151](#)
- FFT\_Fwd256
  - FFT-256, [136](#)
- FFT\_Fwd4096
  - FFT-4096, [156](#)
- FFT\_Fwd512
  - FFT-512, [141](#)
- FFT\_Fwd8192
  - FFT-8192, [160](#)
- FFT\_Inv1024
  - IFFT-1024, [148](#)
- FFT\_Inv2048
  - IFFT-2048, [153](#)
- FFT\_Inv256
  - IFFT-256, [138](#)
- FFT\_Inv4096
  - IFFT-4096, [158](#)
- FFT\_Inv512
  - IFFT-512, [143](#)
- FFT\_Inv8192
  - IFFT-8192, [162](#)
- Filter
  - CIMG\_FIR, [400](#)
- Fix-point 32, [95](#)
- Fix-point 64, [94](#)
- FloodFill8
  - Floodfill, [100](#)
- Floodfill, [100](#)
  - FloodFill8, [100](#)
- GetVec
  - контроль переполнения, [316](#)

- hex2ascii
  - Функции графического вывода текста, 118
- I\_2DSubPixelMinPosition, 401
- I\_PlessyCornerDetector, 402
- IDFT-8, 39
  - nmppsDFT8Inv\_32fcr, 39
- IFFT-1024, 52, 148
  - FFT\_Inv1024, 148
  - nmppsFFT1024Inv\_32fcr, 52
  - nmppsFFT1024InvInitAlloc\_32fcr, 52
- IFFT-128, 46
  - nmppsFFT128Inv\_32fcr, 46
  - nmppsFFT128InvInitAlloc\_32fcr, 46
- IFFT-16, 40
  - nmppsFFT16Inv\_32fcr, 40
  - nmppsFFT16InvInitAlloc\_32fcr, 40
- IFFT-2048, 54, 153
  - FFT\_Inv2048, 153
  - nmppsFFT2048Inv\_32fcr, 54
  - nmppsFFT2048InvInitAlloc\_32fcr, 54
- IFFT-256, 48, 138
  - FFT\_Inv256, 138
  - nmppsFFT256Inv\_32fcr, 48
  - nmppsFFT256InvInitAlloc\_32fcr, 48
- IFFT-32, 42
  - nmppsFFT32Inv\_32fcr, 42
  - nmppsFFT32InvInitAlloc\_32fcr, 42
- IFFT-4096, 56, 158
  - FFT\_Inv4096, 158
  - nmppsFFT4096Inv\_32fcr, 56
  - nmppsFFT4096InvInitAlloc\_32fcr, 56
- IFFT-512, 50, 143
  - FFT\_Inv512, 143
  - nmppsFFT512Inv\_32fcr, 50
  - nmppsFFT512InvInitAlloc\_32fcr, 50
- IFFT-64, 44
  - nmppsFFT64Inv\_32fcr, 44
  - nmppsFFT64InvInitAlloc\_32fcr, 44
- IFFT-8192, 162
  - FFT\_Inv8192, 162
- IFFT-Common, 61
  - nmppsFFTInv\_32fcr, 61
  - nmppsFFTInvInitAlloc\_32fcr, 61
- IMG\_Convert, 106
- IMG\_DeinterlaceBlend
  - Функции деинтерлейсинга, 97
- IMG\_DeinterlaceSplit
  - Функции деинтерлейсинга, 97
- IMG\_Free, 112
- IMG\_MergeFromBlocks, 111
- IMG\_Print8x15
  - Функции графического вывода текста, 119
- IMG\_RGB32ToGray, 107
- IMG\_Release, 113
- IMG\_SplitIntoBlocks, 110
- im
  - s\_nm64sc, 422
- int15b
  - Типы скалярных данных, 195
- int15in16x4, 402
- int16b
  - Типы скалярных данных, 196
- int1b
  - Типы скалярных данных, 196
- int2b
  - Типы скалярных данных, 196
- int30b
  - Типы скалярных данных, 196
- int30in32x2, 403
- int31b
  - Типы скалярных данных, 197
- int31in32x2, 403
- int32b
  - Типы скалярных данных, 197
- int3b
  - Типы скалярных данных, 197
- int4b
  - Типы скалярных данных, 197
- int63b
  - Типы скалярных данных, 198
- int64b
  - Типы скалярных данных, 198
- int7b
  - Типы скалярных данных, 198
- int8b
  - Типы скалярных данных, 198
- Integer operations, 93
- MTR\_Addr, 131
- MTR\_Copy, 123
- MTR\_Copyau, 122
- MTR\_Free, 130
- MTR\_GetVal, 133
- MTR\_Malloc, 129
- MTR\_ProdUnitV, 128
- MTR\_SetVal, 132
- mtr< T >, 404
- nm1
  - Типы векторных данных, 184
- nm16s
  - Типы векторных данных, 184
- nm16s15b
  - Типы векторных данных, 184
- nm16sc, 405
- nm16u
  - Типы векторных данных, 185
- nm16u15b
  - Типы векторных данных, 185
- nm2s
  - Типы векторных данных, 185
- nm2u
  - Типы векторных данных, 186
- nm32s
  - Типы векторных данных, 186
- nm32s30b
  - Типы векторных данных, 186

nm32s31b  
     Типы векторных данных, 187  
 nm32u  
     Типы векторных данных, 187  
 nm32u31b  
     Типы векторных данных, 187  
 nm4s  
     Типы векторных данных, 188  
 nm4u  
     Типы векторных данных, 188  
 nm4u3b  
     Типы векторных данных, 188  
 nm64s  
     Типы векторных данных, 189  
 nm64s63b  
     Типы векторных данных, 189  
 nm64u  
     Типы векторных данных, 189  
 nm8s  
     Типы векторных данных, 190  
 nm8s7b  
     Типы векторных данных, 190  
 nm8u  
     Типы векторных данных, 190  
 nm8u7b  
     Типы векторных данных, 191  
 nm\_trans  
     BLASS-LEVEL1, 66  
 nmblas\_dasum  
     BLASS-LEVEL1, 66  
 nmchar, 406  
 nmchar1D< N >, 406  
 nmchar2D< Y, X >, 407  
 nmintpack< T >, 407  
 nmmtr< T >, 408  
 nmppMerge, 305  
 nmppSplit, 304  
 nmppSplit\_32fcr, 306  
 nmppSplitTmp, 303  
 nmppcDivC, 68  
 nmppcDoubleToFix32, 73  
 nmppcDoubleToFix64, 84  
 nmppcFix32ToDouble, 74  
 nmppcFix64Exp01, 89  
 nmppcFix64ToDouble, 85  
 nmppcFixArcTan32, 72  
 nmppcFixArcTan64, 88  
 nmppcFixDiv64, 86  
 nmppcFixDivMod32, 82  
 nmppcFixExp32, 70  
 nmppcFixInv32, 77  
 nmppcFixMul32, 76  
 nmppcFixSinCos32, 71  
 nmppcFixSinCos64, 87  
 nmppcFixSqrt32, 75  
 nmppcFixSqrt64, 83  
 nmppcProdC, 69  
 nmppcSqrt, 91

nmppcTblFixArcCos32, 79  
 nmppcTblFixArcSin32, 78  
 nmppcTblFixCos32, 80  
 nmppcTblFixSin32, 81  
 NmppiFFTSpec\_32fcr, 409  
 nmppmCopyua, 121  
 nmppmMul\_mm, 124  
 nmppmMul\_mv\_, 126  
 nmppmMul\_mv\_\_AddC, 127  
 nmppsAbs, 208  
 nmppsAbs1, 209  
 nmppsAbsDiff, 217  
 nmppsAbsDiff1, 218  
 nmppsAdd, 212  
 nmppsAdd\_AddC, 213  
 nmppsAddC, 211  
 nmppsAddr\_, 296  
 nmppsAnd, 234  
 nmppsAnd4V\_, 235  
 nmppsAndNotV\_, 236  
 nmppsAndC, 233  
 nmppsClipCC\_, 285  
 nmppsClipConvert\_AddC\_, 287  
 nmppsClipPowC\_, 284  
 nmppsClipRShiftConvert\_AddC\_, 286  
 nmppsCmpEq0, 280  
     nmppsCmpEq0\_16u15b, 280  
     nmppsCmpEq0\_32u31b, 281  
     nmppsCmpEq0\_8u7b, 281  
 nmppsCmpEq0\_16u15b  
     nmppsCmpEq0, 280  
 nmppsCmpEq0\_32u31b  
     nmppsCmpEq0, 281  
 nmppsCmpEq0\_8u7b  
     nmppsCmpEq0, 281  
 nmppsCmpEqV\_, 294  
 nmppsCmpEqC, 289  
 nmppsCmpLt0, 279  
 nmppsCmpMinMaxV\_, 282  
 nmppsCmpNe0, 290  
 nmppsCmpNeV\_, 295  
 nmppsCmpNeC, 291  
 nmppsConvert, 253  
     nmppsConvert\_1s2s, 254  
     nmppsConvert\_1u2u, 254  
     nmppsConvert\_32s32fcr, 254  
     nmppsConvert\_32sc32fcr, 255  
     nmppsConvert\_32u32fcr, 255  
     nmppsConvertRisc\_32u8u, 256  
     nmppsConvertRisc\_8u32u, 256  
     nmppsJoin\_32f, 256  
 nmppsConvert\_1s2s  
     nmppsConvert, 254  
 nmppsConvert\_1u2u  
     nmppsConvert, 254  
 nmppsConvert\_32s32fcr  
     nmppsConvert, 254  
 nmppsConvert\_32sc32fcr

- nmppsConvert, [255](#)
- nmppsConvert\_32u32fcr
  - nmppsConvert, [255](#)
- nmppsConvertRisc\_32u8u
  - nmppsConvert, [256](#)
- nmppsConvertRisc\_8u32u
  - nmppsConvert, [256](#)
- nmppsCopy\_, [258](#)
- nmppsCopyua\_, [259](#)
- nmppsDFT8Fwd\_32fcr
  - DFT-8, [19](#)
- nmppsDFT8Inv\_32fcr
  - IDFT-8, [39](#)
- nmppsDecimate, [307](#)
- nmppsDisplaceBits, [247](#)
- nmppsDivC, [226](#)
- nmppsDotProd, [229](#)
- nmppsFFT1024Fwd\_32fcr
  - FFT-1024, [33](#)
- nmppsFFT1024FwdInitAlloc\_32fcr
  - FFT-1024, [33](#)
- nmppsFFT1024Inv\_32fcr
  - IFFT-1024, [52](#)
- nmppsFFT1024InvInitAlloc\_32fcr
  - IFFT-1024, [52](#)
- nmppsFFT128Fwd\_32fcr
  - FFT-128, [27](#)
- nmppsFFT128FwdInitAlloc\_32fcr
  - FFT-128, [27](#)
- nmppsFFT128Inv\_32fcr
  - IFFT-128, [46](#)
- nmppsFFT128InvInitAlloc\_32fcr
  - IFFT-128, [46](#)
- nmppsFFT16Fwd\_32fcr
  - FFT-16, [21](#)
- nmppsFFT16FwdInitAlloc\_32fcr
  - FFT-16, [21](#)
- nmppsFFT16Inv\_32fcr
  - IFFT-16, [40](#)
- nmppsFFT16InvInitAlloc\_32fcr
  - IFFT-16, [40](#)
- nmppsFFT2048Fwd\_32fcr
  - FFT-2048, [35](#)
- nmppsFFT2048FwdInitAlloc\_32fcr
  - FFT-2048, [35](#)
- nmppsFFT2048Inv\_32fcr
  - IFFT-2048, [54](#)
- nmppsFFT2048InvInitAlloc\_32fcr
  - IFFT-2048, [54](#)
- nmppsFFT256Fwd\_32fcr
  - FFT-256, [29](#)
- nmppsFFT256FwdInitAlloc\_32fcr
  - FFT-256, [29](#)
- nmppsFFT256Inv\_32fcr
  - IFFT-256, [48](#)
- nmppsFFT256InvInitAlloc\_32fcr
  - IFFT-256, [48](#)
- nmppsFFT32Fwd\_32fcr
  - FFT-32, [23](#)
- nmppsFFT32FwdInitAlloc\_32fcr
  - FFT-32, [23](#)
- nmppsFFT32Inv\_32fcr
  - IFFT-32, [42](#)
- nmppsFFT32InvInitAlloc\_32fcr
  - IFFT-32, [42](#)
- nmppsFFT4096Fwd\_32fcr
  - FFT-4096, [37](#)
- nmppsFFT4096FwdInitAlloc\_32fcr
  - FFT-4096, [37](#)
- nmppsFFT4096Inv\_32fcr
  - IFFT-4096, [56](#)
- nmppsFFT4096InvInitAlloc\_32fcr
  - IFFT-4096, [56](#)
- nmppsFFT512Fwd\_32fcr
  - FFT-512, [31](#)
- nmppsFFT512FwdInitAlloc\_32fcr
  - FFT-512, [31](#)
- nmppsFFT512Inv\_32fcr
  - IFFT-512, [50](#)
- nmppsFFT512InvInitAlloc\_32fcr
  - IFFT-512, [50](#)
- nmppsFFT64Fwd\_32fcr
  - FFT-64, [25](#)
- nmppsFFT64FwdInitAlloc\_32fcr
  - FFT-64, [25](#)
- nmppsFFT64Inv\_32fcr
  - IFFT-64, [44](#)
- nmppsFFT64InvInitAlloc\_32fcr
  - IFFT-64, [44](#)
- nmppsFFTFwd\_32fcr
  - FFT-Common, [58](#)
- nmppsFFTFwdInitAlloc\_32fcr
  - FFT-Common, [58](#)
- nmppsFFTInv\_32fcr
  - IFFT-Common, [61](#)
- nmppsFFTInvInitAlloc\_32fcr
  - IFFT-Common, [61](#)
- NmppsFFTSpec, [409](#)
- NmppsFFTSpec\_32fcr, [410](#)
- nmppsFIR\_Xs, [172](#)
- nmppsFIRFree, [176](#)
- nmppsFIRGetStateSize\_Xs, [175](#)
- nmppsFIRInit\_Xs, [173](#)
- nmppsFIRInitAlloc\_Xs, [174](#)
- nmppsFirstNonZeroIndx, [271](#)
- nmppsFirstZeroIndx, [270](#)
- NmppsFrame\_16s, [410](#)
- NmppsFrame\_16u, [410](#)
- NmppsFrame\_32s, [411](#)
- NmppsFrame\_32u, [411](#)
- NmppsFrame\_64s, [411](#)
- NmppsFrame\_64u, [412](#)
- NmppsFrame\_8s, [412](#)
- NmppsFrame\_8u, [412](#)
- nmppsFree, [65](#)
- nmppsGetVal\_, [298](#)

- nmppsGetVal\_(return), 299
- nmppsJoin\_32f
  - nmppsConvert, 256
- nmppsLastNonZeroIndx, 273
- nmppsLastZeroIndx, 272
- nmppsMalloc, 64
- NmppsMallocSpec, 413
- nmppsMaskV\_, 243
- nmppsMax\_, 261
  - nmppsMax\_16s15b, 261
  - nmppsMax\_32s31b, 262
  - nmppsMax\_8s7b, 262
- nmppsMax\_16s15b
  - nmppsMax\_, 261
- nmppsMax\_32s31b
  - nmppsMax\_, 262
- nmppsMax\_8s7b
  - nmppsMax\_, 262
- nmppsMaxEvery\_, 275
- nmppsMaxIndx\_, 265
- nmppsMin, 263
  - nmppsMin\_16s15b, 263
  - nmppsMin\_32s31b, 264
  - nmppsMin\_8s7b, 264
- nmppsMin\_16s15b
  - nmppsMin, 263
- nmppsMin\_32s31b
  - nmppsMin, 264
- nmppsMin\_8s7b
  - nmppsMin, 264
- nmppsMinCmpLtV\_, 277
- nmppsMinEvery\_, 274
- nmppsMinIndx\_, 267
- nmppsMinIndxVN\_, 269
- nmppsMul\_AddC, 220
- nmppsMulC\_AddC\_2x32s
  - nmppsMulC\_AddC, 221
- nmppsMulC\_AddV\_AddC, 224
- nmppsMulC\_AddC, 221
  - nmppsMulC\_AddC\_2x32s, 221
- nmppsMulC, 219
- nmppsNeg, 210
- nmppsNot\_, 232
- nmppsOr, 238
- nmppsOr3V\_, 239
- nmppsOr4V\_, 240
- nmppsOrC, 237
- nmppsRShiftC\_, 245
- nmppsRShiftC\_AddC\_, 246
- nmppsRShiftC\_MulC\_AddC, 223
- nmppsRShiftC, 244
- nmppsRamp\_, 252
- nmppsRand, 90
- nmppsRandUniform, 250
  - nmppsRandUniform\_64s, 250
- nmppsRandUniform\_, 251
- nmppsRandUniform\_64s
  - nmppsRandUniform, 250
- nmppsRemap\_, 301
- nmppsSet-инициализация, 249
- nmppsSetVal\_, 297
- nmppsSub, 216
- nmppsSubCRev, 215
- nmppsSubC, 214
- nmppsSum, 228
- nmppsSumN, 225
- nmppsSwap\_, 260
- NmppsTmpSpec, 413
- nmppsWeightedSum, 231
- nmppsXor, 242
- nmppsXorC, 241
- nmreg, 414
- nmshort, 414
- nmshort2D< Y, X >, 415
- nmtype.h
  - CAPACITY\_nm64s, 444
  - VEC\_NM16S, 444
  - VEC\_NM16U, 444
  - VEC\_NM32S, 444
  - VEC\_NM4U, 445
  - VEC\_NM8S, 445
- nmvecpack< T >, 415
- operator<<
  - контроль переполнения, 317
- RGB32\_nm10s, 417
- RGB32\_nm10u, 417
- RGB32\_nm8s, 417
- RGB32\_nm8u, 418
- RGB64\_nm16u, 418
- RPoint, 419
- re
  - s\_nm64sc, 422
- S\_BufferInfo, 419
- S\_IMG\_FilterKernel, 420
- S\_IMG\_FilterKernel\_32s32s, 420
- s\_int32x2, 421
- s\_nm32fc, 421
- s\_nm32fcr, 421
- s\_nm32sc, 422
- s\_nm64sc, 422
  - im, 422
  - re, 422
- s\_v16nm16s, 423
- s\_v16nm16u, 423
- s\_v16nm32s, 424
- s\_v16nm32u, 424
- s\_v16nm4u, 424
- s\_v16nm8s, 425
- s\_v16nm8u, 425
- s\_v2nm32s, 426
- s\_v2nm32u, 426
- s\_v4nm16s, 426
- s\_v4nm16u, 427
- s\_v4nm32s, 427



- s\_v4nm32u, 428
- s\_v4nm8u, 428
- s\_v8nm16s, 428
- s\_v8nm16u, 429
- s\_v8nm32s, 429
- s\_v8nm32u, 430
- s\_v8nm8s, 430
- s\_v8nm8u, 430
- SIG\_CreateResample, 179
- SIG\_Median3, 170
- SIG\_Resample\_perf, 181
- SIG\_ResampleDown2, 177
- SIG\_ResampleUp3Down2, 178
- SIG\_SetResample, 180
- SIG\_XCorr, 168
- SetWeights
  - CIMG\_FIR, 400
- SpecTmp1, 431
- spot\_struct, 431
  
- tagSegmentInfo, 432
- tfixpoint< T, point >, 432
- Tmp2BuffSpec, 433
  
- uint15b
  - Типы скалярных данных, 199
- uint16b
  - Типы скалярных данных, 199
- uint16ptr, 434
- uint1b
  - Типы скалярных данных, 199
- uint2b
  - Типы скалярных данных, 199
- uint31b
  - Типы скалярных данных, 200
- uint32b
  - Типы скалярных данных, 200
- uint3b
  - Типы скалярных данных, 200
- uint4b
  - Типы скалярных данных, 200
- uint63b
  - Типы скалярных данных, 201
- uint64b
  - Типы скалярных данных, 201
- uint7b
  - Типы скалярных данных, 201
- uint8b
  - Типы скалярных данных, 201
- uint8ptr, 435
  
- v16nm16s
  - Типы векторных данных, 191
- v16nm16u
  - Типы векторных данных, 191
- v16nm32s
  - Типы векторных данных, 191
- v16nm32u
  - Типы векторных данных, 191
- v16nm4b3u
  - Типы векторных данных, 192
- v16nm4s, 435
- v16nm4u
  - Типы векторных данных, 192
- v16nm8s
  - Типы векторных данных, 192
- v16nm8s7b
  - Типы векторных данных, 192
- v16nm8u
  - Типы векторных данных, 192
- v2nm32s
  - Типы векторных данных, 192
- v2nm32u
  - Типы векторных данных, 193
- v4nm16s
  - Типы векторных данных, 193
- v4nm16u
  - Типы векторных данных, 193
- v4nm32s
  - Типы векторных данных, 193
- v4nm32u
  - Типы векторных данных, 193
- v4nm8s, 436
- v4nm8u
  - Типы векторных данных, 193
- v8nm16s
  - Типы векторных данных, 193
- v8nm16u
  - Типы векторных данных, 193
- v8nm32s
  - Типы векторных данных, 194
- v8nm32u
  - Типы векторных данных, 194
- v8nm8s
  - Типы векторных данных, 194
- v8nm8u
  - Типы векторных данных, 194
- VEC\_NM16S
  - nmtypes.h, 444
- VEC\_NM16U
  - nmtypes.h, 444
- VEC\_NM32S
  - nmtypes.h, 444
- VEC\_NM4U
  - nmtypes.h, 445
- VEC\_NM8S
  - nmtypes.h, 445
- VEC\_QSort, 300
- vec< T >, 436
- Vec\_0\_sub\_data, 321
- Vec\_Abs, 342
- Vec\_AccMul1D1W32\_AddVr, 386
- Vec\_Add, 343
- Vec\_Add\_VV\_shift, 326
- Vec\_And, 338
- Vec\_BuildDiagWeights16, 379
- Vec\_BuildDiagWeights8, 378

Vec\_ClipExt, [344](#)  
Vec\_ClipMul2D2W8\_AddVr, [345](#)  
Vec\_ClipMulNDNW2\_AddVr, [346](#)  
Vec\_ClipMulNDNW4\_AddVr, [347](#)  
Vec\_ClipMulNDNW8\_AddVr, [348](#)  
Vec\_CompareMaxV, [375](#)  
Vec\_CompareMinV, [374](#)  
Vec\_DupValueInVector16, [377](#)  
Vec\_DupValueInVector8, [376](#)  
Vec\_FilterCoreRow2, [335](#)  
Vec\_FilterCoreRow4, [336](#)  
Vec\_FilterCoreRow8, [337](#)  
Vec\_IncNeg, [349](#)  
Vec\_MUL\_2V4toW8\_shift, [362](#)  
Vec\_MUL\_2V8toW16\_shift, [363](#)  
Vec\_Mask, [339](#)  
Vec\_MaxVal, [382](#)  
Vec\_MaxVal\_v4nm16s, [381](#)  
Vec\_MaxVal\_v8nm8s, [380](#)  
Vec\_MinVal, [385](#)  
Vec\_MinVal\_v4nm16s, [384](#)  
Vec\_MinVal\_v8nm8s, [383](#)  
Vec\_Mul2D2W1\_AddVr, [350](#)  
Vec\_Mul2D2W2\_AddVr, [351](#)  
Vec\_Mul2D2W4\_AddVr, [352](#)  
Vec\_Mul2D2W8\_AddVr, [353](#)  
Vec\_Mul3D3W2\_AddVr, [354](#)  
Vec\_Mul3D3W8\_AddVr, [355](#)  
Vec\_Mul4D4W2\_AddVr, [356](#)  
Vec\_MulVN\_AddVN, [357](#)  
Vec\_Or, [340](#)  
Vec\_Sub, [358](#)  
Vec\_SubAbs, [359](#)  
Vec\_SubVN\_Abs, [360](#)  
Vec\_Swap, [361](#)  
Vec\_Xor, [341](#)  
Vec\_activate\_data, [322](#)  
Vec\_activate\_data\_add\_0, [323](#)  
Vec\_activate\_data\_add\_ram, [325](#)  
Vec\_activate\_data\_xor\_data, [324](#)  
Vec\_afifo, [327](#)  
Vec\_data, [328](#)  
Vec\_data\_add\_afifo, [329](#)  
Vec\_data\_add\_ram, [330](#)  
Vec\_data\_and\_ram, [331](#)  
Vec\_data\_or\_ram, [332](#)  
Vec\_data\_sub\_ram, [333](#)  
Vec\_data\_xor\_ram, [334](#)  
Vec\_not\_data, [364](#)  
Vec\_ram, [365](#)  
Vec\_ram\_sub\_data, [366](#)  
Vec\_vsum\_activate\_data\_0, [367](#)  
Vec\_vsum\_data\_0, [368](#)  
Vec\_vsum\_data\_afifo, [369](#)  
Vec\_vsum\_data\_vr, [370](#)  
Vec\_vsum\_shift\_data\_0, [371](#)  
Vec\_vsum\_shift\_data\_afifo, [373](#)  
Vec\_vsum\_shift\_data\_vr, [372](#)